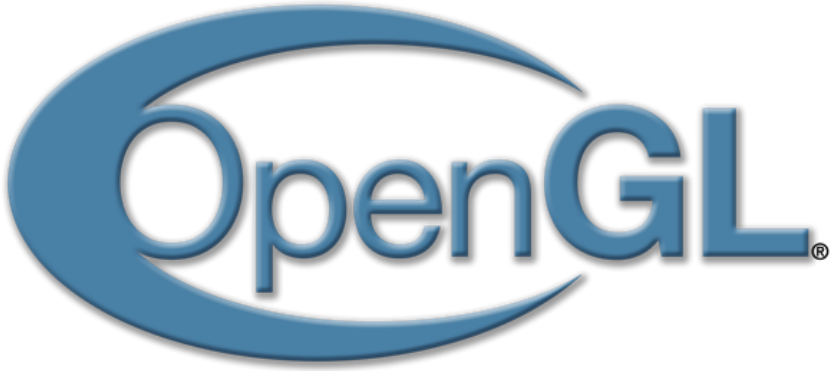


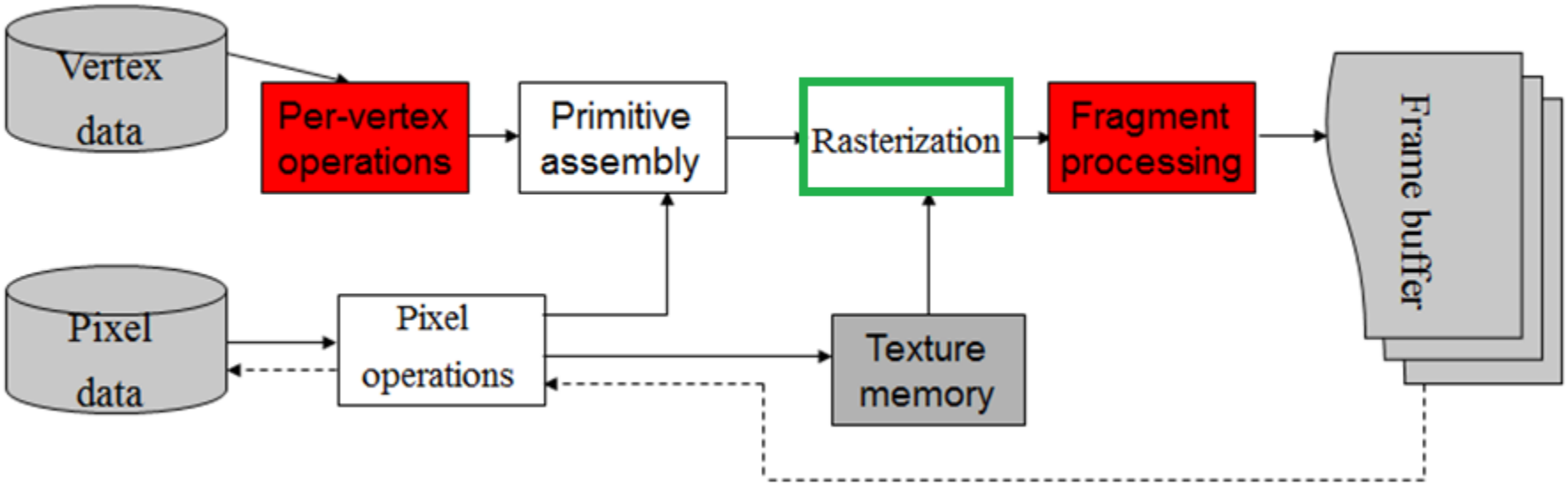
PV112 – Programování grafických aplikací

3. přednáška – Rasterizace, homogenní souřadnice, transformace souřadnic

https://solarianprogrammer.com/2013/05/22/opengl-101-matrices-projection-view-model/

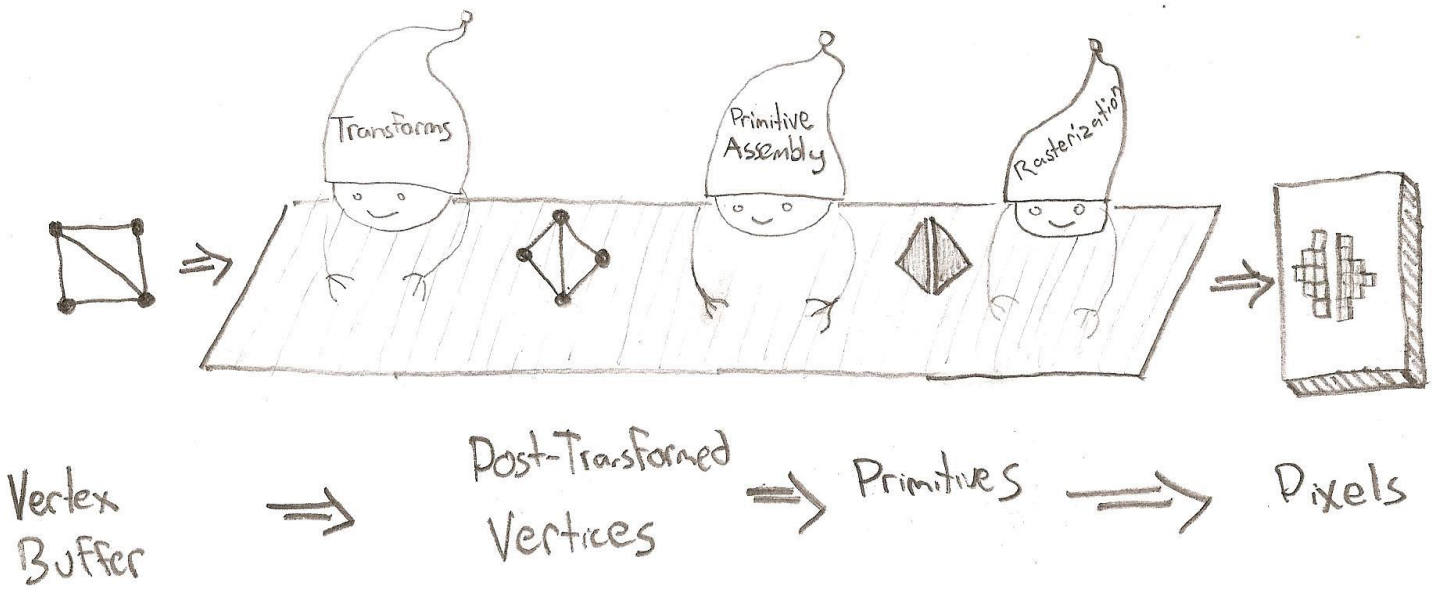
Rasterizace

- Proces konverze primitiv do 2D obrázku



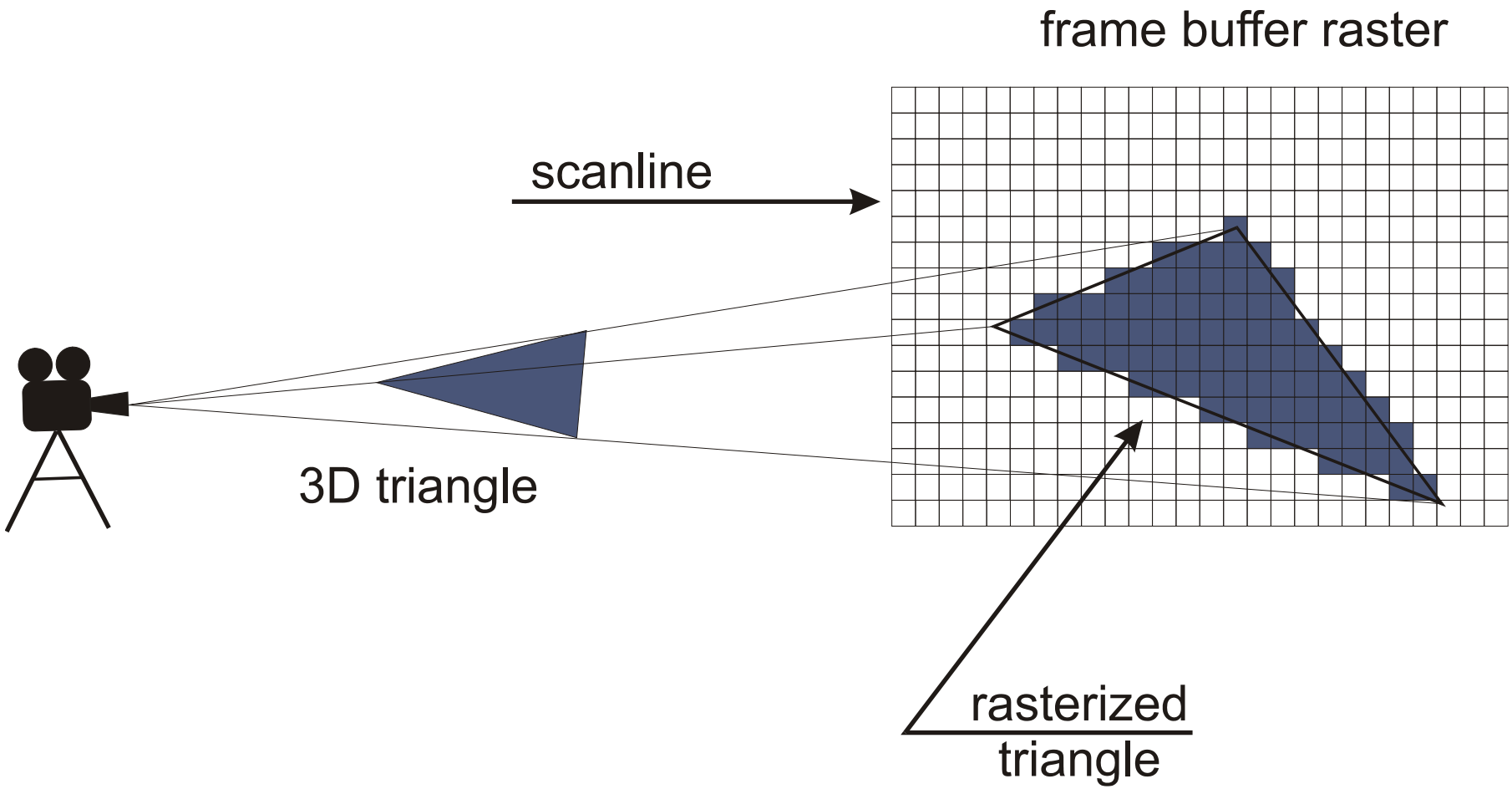
Rasterizace

- Dvě části:
  - Které obdélníky (pixely) mřížky okna jsou obsazeny
  - Každému obdélníku přiřadíme příslušnou hodnotu barvy a hloubky

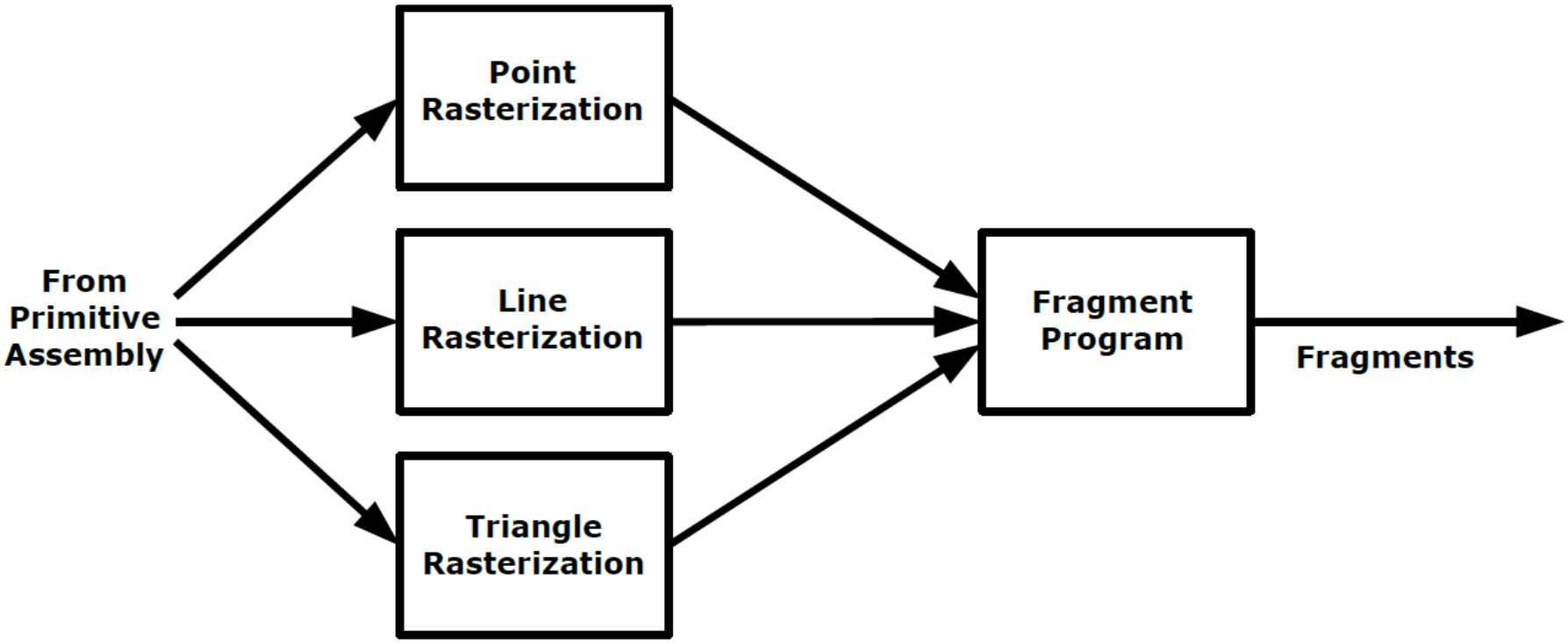


ofps.oreilly.com

Princip rasterizace



Detail rasterizace

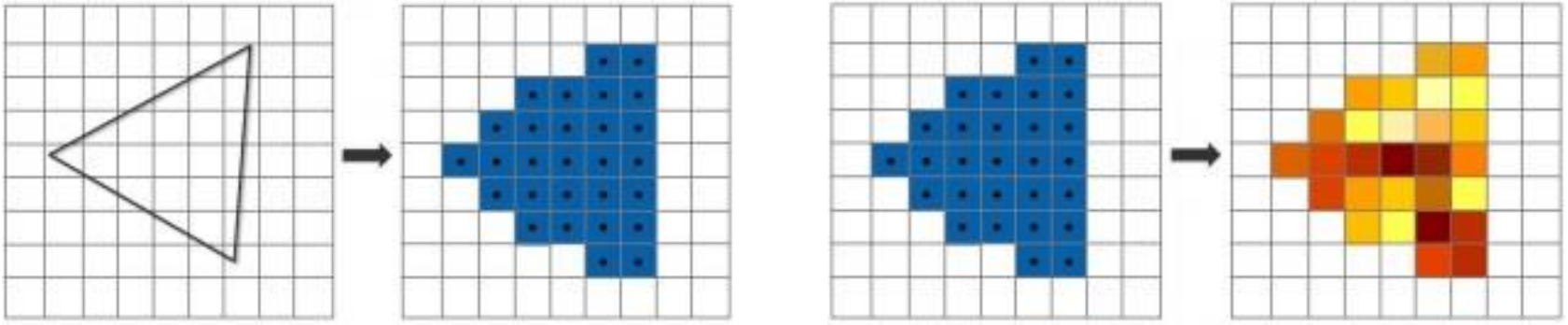


Fragment

- Fragment = zlomek objektu v mřížce společně s přiřazenými parametry, např.:
  - Barva
  - Hloubka (hodnota Z)
- Parametry se nazývají „fragment’s associated data“

Fragment

- Fragment je definován dolním levým rohem, který leží v mřížce celočíselných souřadnic



http://gpudesign.bafree.net/computer-graphics/graphics-pipeline

Rasterizace bodů

- Rasterizace bodů produkuje fragment pro každý pixel framebufferu, jehož střed leží uvnitř čtverce se středem v bodu $[x_w,y_w]$, jehož délka strany je shodná s aktuální velikostí bodu.
- Souřadnice bodů jsou ořezány na souřadnice okna $[x_w,y_w]$.

Rasterizace bodů

- Řízena pomocí

```
glEnable(GL_PROGRAM_POINT_SIZE);
gl_Pointsize = size;
```

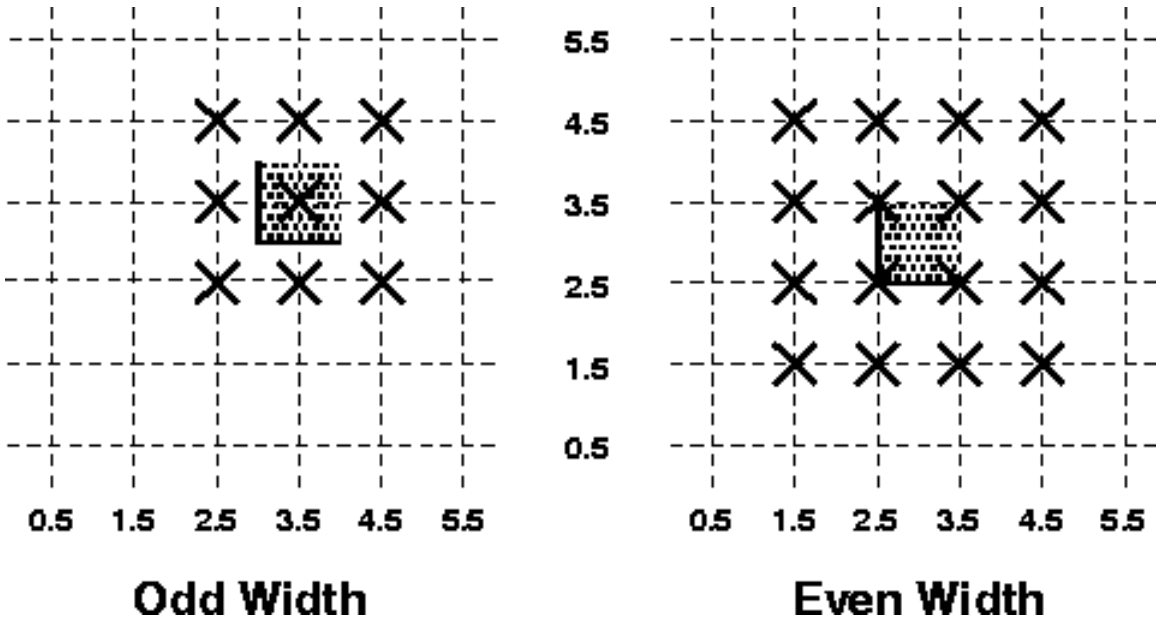
  kde *size* je velikost vykreslovaného bodu.

- Pokud je `GL_PROGRAM_POINT_SIZE` disabled, lze pro nastavení použít funkci `glPointSize(float size)`

Rasterizace velkých bodů

Je-li výsledek po zaokrouhlení **sudý**:

$$(x,y) = (\lfloor x_w + ½ \rfloor , \lfloor y_w + ½ \rfloor)$$

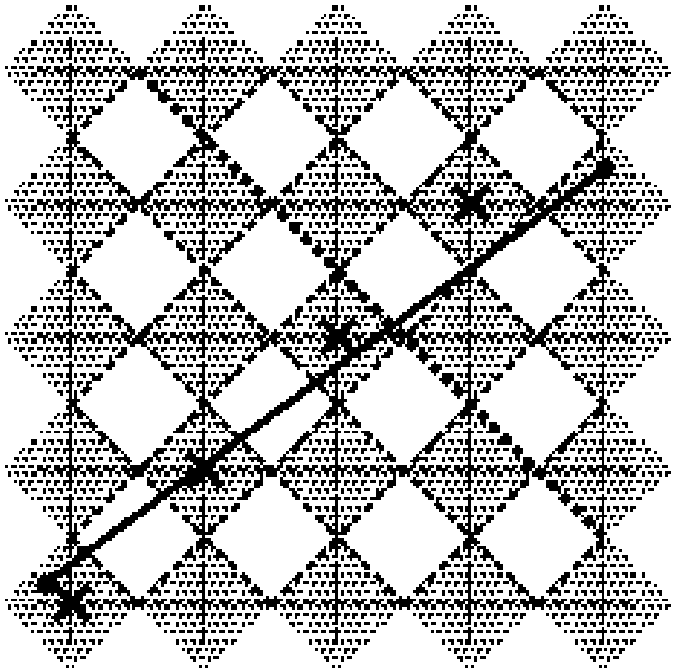


Rasterizace úseček

- Řízena šířkou úsečky
- Určení typu úsečky
  - x-hlavní – sklon v rozmezí uzavřeného intervalu [-1, 1]
  - y-hlavní – ostatní

Rasterizace úseček

- „diamond-exit“ pravidlo pro určení fragmentů:
  - Pro každý fragment *f* se středem $[x_f,y_f]$ definujeme region tvaru diamantu, který je průsečíkem čtyř polorovin: $R_f = \{(x,y) \mid |x - x_f| + |y - y_f| < ½\}$



Rasterizace úseček

- Použití modifikovaného Bressenhamova algoritmu:
  - Vytvořená úsečka je „polootevřená“ = koncový fragment není vykreslen
  - Výhodné při vykreslování sady na sebe navazujících úseček

Rasterizace úseček

- Určení interpolace barev a hloubky:

$p = (x, y)$ aktuální bod na úsečce

$a = (x_a, y_a)$ počáteční bod

$b = (x_b, y_b)$ koncový bod

$$t = \frac{(p-a)(b-a)}{\|b-a\|^2}$$

Rasterizace úseček

kde:

$(p-a)(b-a) = (x-x_a)(x_b-x_a) + (y-y_a)(y_b-y_a)$ je skalární součin

$\|b-a\|^2 = (x_b-x_a)^2 + (y_b-y_a)^2$ je druhá mocnina vektorové délky

t je v intervalu [0, 1] a ***t*** = 0 pro *a*, ***t*** = 1 pro *b*

Homogenní souřadnice

- Problém – dvě úsečky protínající se v nekonečnu
- Převod kartézských souřadnic na homogenní = přidání další dimenze:
  - (X, Y) → (x, y, w)

    X = x/w

    Y = y/w

www.art.com

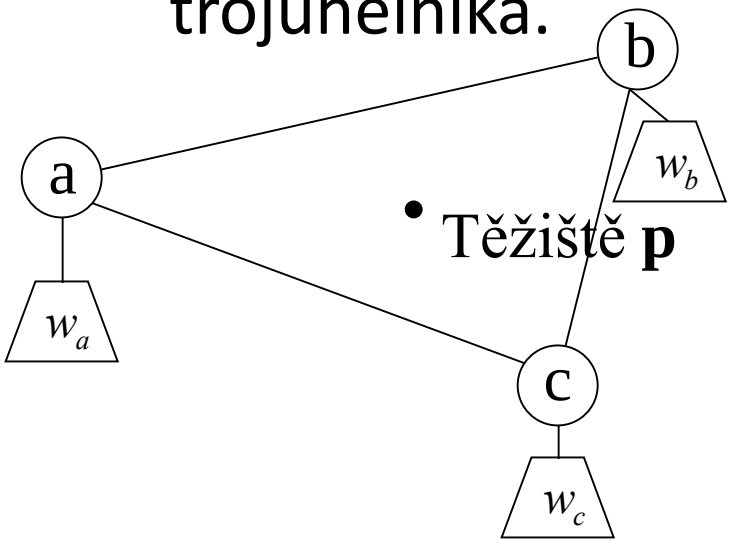
Homogenní souřadnice

- Homogenní souřadnice mají přirozené aplikace v počítačové grafice
- Vytváří základ pro projektivní geometrii, která je široce použita pro projekci třírozměrných scén do dvourozměrné roviny obrazu
- Homogenní souřadnice také umožňují jednotné použití běžných grafických transformací a operací

Definice homogenních souřadnic

- Definice 1: **Möbius**

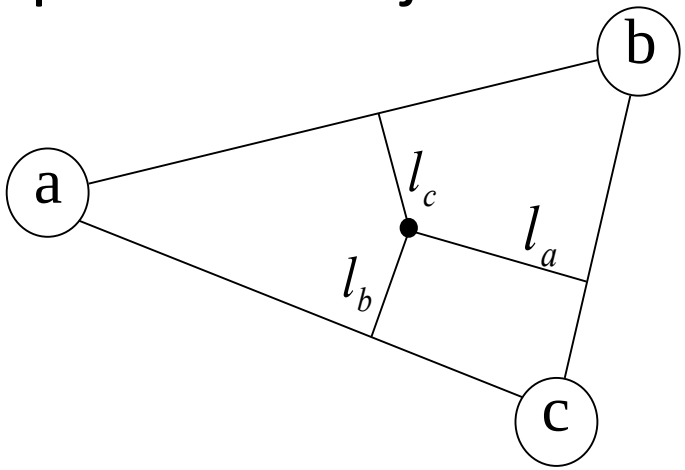
  Mějme pevně daný trojúhelník v rovině.
  Množina homogenních souřadnic pro bod ***p*** je definována jako hmotnosti přiřazené vrcholům trojúhelníka takové, že ***p*** se stane těžištěm trojúhelníka.



Bod ***p*** je vypočten jako
$(w_a a, w_b b, w_c c)$
Kde platí:
$w_a + w_b + w_c = 1$
w_a, w_b a w_c se nazývají barycentrické souřadnice

Definice homogenních souřadnic

- Definice 2: **Plücker**
  - Pracuje se vzdálenostmi z daného bodu ke hranám pevného trojúhelníka



Plückerovy souřadnice

$$\mathbf{p} = \begin{pmatrix} l_a & l_b & l_c \end{pmatrix}$$

  - n+1 souřadnic reprezentuje n-dimenzionální bod

Proč homogenní?

- Invariance při „scalingu“
  - Scaling Möbiových vah ani velikosti Plückerových souřadnic nezmění pozici bodu ***p***
- Příklad:

  (1, 2, 3) = (1/3, 2/3)

  (2, 4, 6) = (2/6, 4/6) = (1/3, 2/3)

  (4, 8, 12) = (4/12, 8/12) = (1/3, 2/3)

  A obecně tedy:

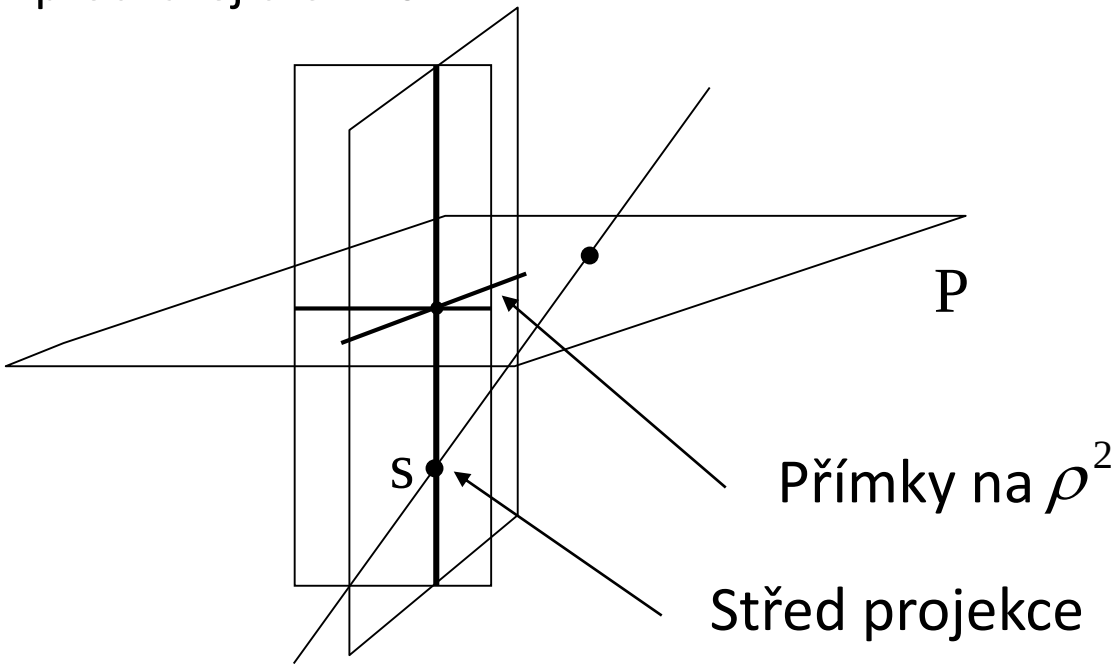
  (1a, 2a, 3a) = (1a/3a, 2a/3a) = (1/3, 2/3)

Projektivní rovina

- Normální rovina doplněná o body v nekonečnu
- Body v nekonečnu jsou definovány homogenními souřadnicemi, kdy w = 0.
- Projektivní rovina nemůže být reprezentována v konečných Euklidovských souřadnicích, může však být reprezentována v homogenních souřadnicích, což je základní důvod jejího použití v projektivní geometrii

Projektivní rovina

- Projektivní rovina je definována pomocí všech přímek a rovin procházejících skrz daný bod ***s***, který nazýváme střed projekce. Jestliže jsou tyto přímky a roviny protnuty rovinou **P**, která neprochází skrz ***s***, pak každý bod (nebo přímka na **P**) může být asociován s přímkou (nebo rovinou) procházející skrz ***s***.



Dvojrozměrný projektivní prostor ρ^2

1. Množina všech tříd ekvivalence uspořádaných trojic nenulových vektorů v ε^3, kde ekvivalence je vzájemná úměra dvou vektorů.
2. Množina všech přímek procházejících skrz počátek ε^3.
3. Množina všech dvojic protilehlých bodů S^2 jednotkové koule v ε^3.

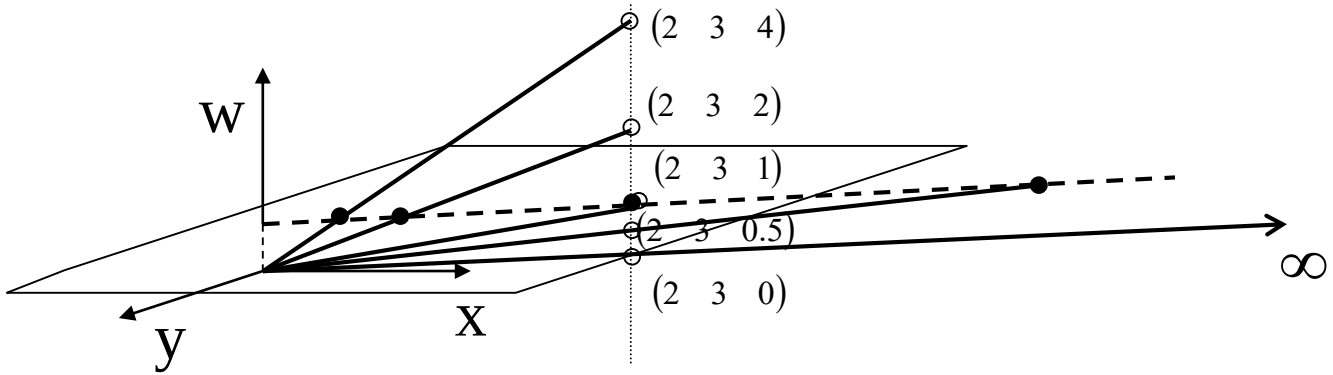
Homogenní souřadnice reprezentují projektivní rovinu ρ^2 zobrazením každého Euklidovského bodu $(x', y') \in \varepsilon^2$ na $[x\ y\ w] \in \varepsilon^3$, $w<>0$, který je členem třídy ekvivalence bodů v ρ^2. Zobrazení je dosaženo ekvivalencí $x' \approx x/w$, $y' \approx y/w$.

Dvojrozměrný projektivní prostor ρ^2

- Skalární násobení vytváří nového člena dané třídy ekvivalence.
- Jestliže jsou vybráni zástupci z průniku s rovinou **w=1**, pak vektorové sčítání dvou bodů v ρ^2 vypočte zástupce středového bodu mezi těmito dvěma vektory. Například:

$$[1, 2, 1] + [3, 4, 1] = [4, 6, 2] \approx (2, 3)$$

- Protože homogenní body reprezentují projekci, mohou také reprezentovat body v nekonečnu.



Homogenní souřadnice ve 2D

- Nejobecnější afinní mapování v Euklidovském prostoru je:

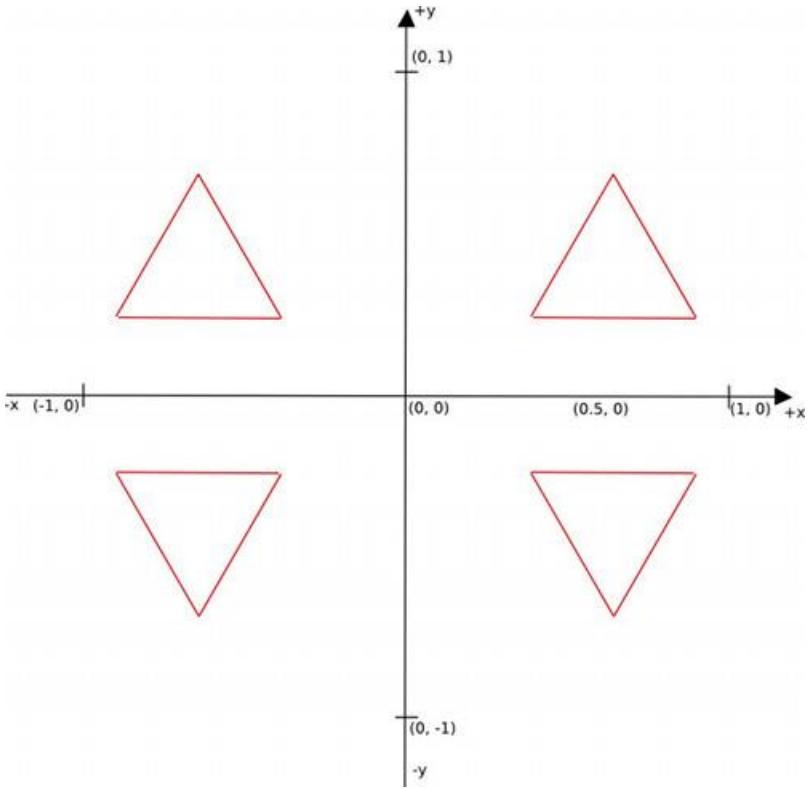
$$p' = pA + c \quad \text{nebo} \quad \left(p'_x, \quad p'_y\right) = \left(p_x, \quad p_y\right)A + \left(c_x, \quad c_y\right)$$

kde

  - p’ a p jsou body v $\mathcal{E}^2$
  - A je 2x2 matice reprezentující *scaling* a *rotaci*
  - c reprezentuje *translaci* (posun)

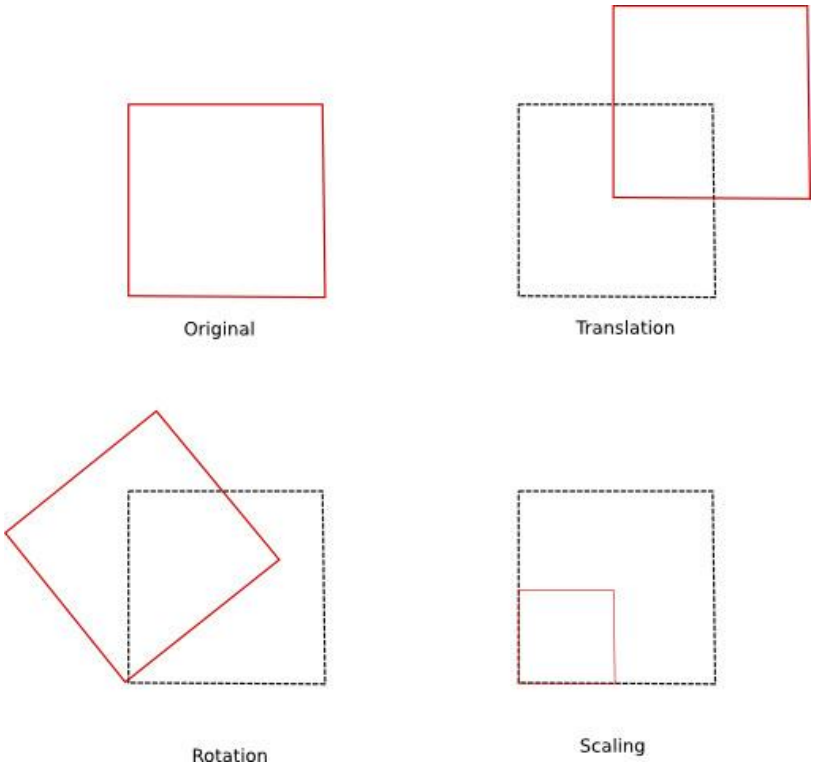
Transformace – motivace

- Chceme vykreslit 4 trojúhelníky s různými pozicemi a orientacemi



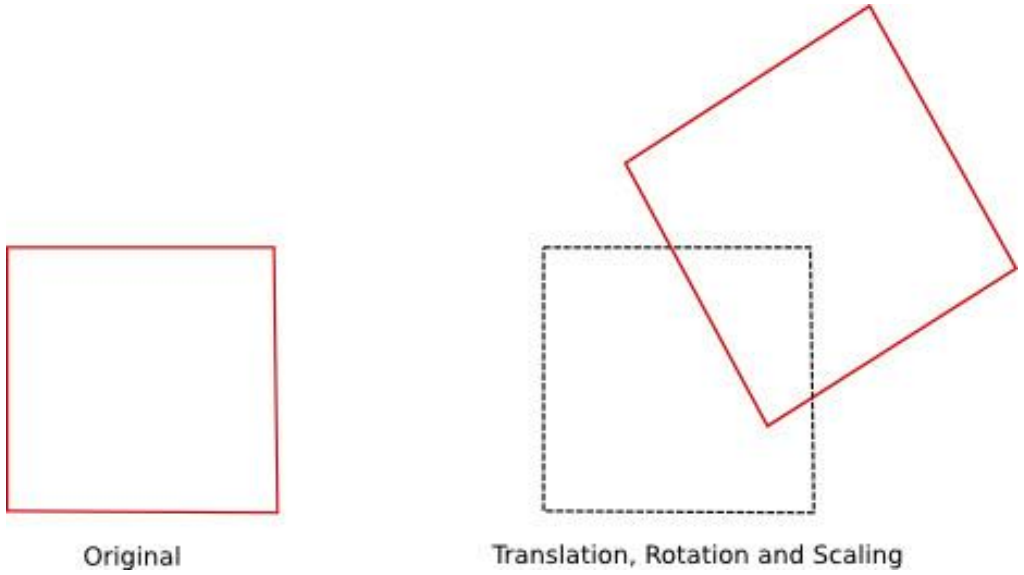
Transformace

- Transformace založeny na maticové algebře
- Objekt uložíme pouze jednou a provedeme transformace:
  - Posun, rotace, scaling



Kombinace transformací

- Skládáním transformací dosáhneme složitějších efektů



Transformace v OpenGL

- Přesun souřadnic objektu do OpenGL bufferu
- Vynásobení souřadnic *transformační maticí* ve vertex shaderu

Základní vlastnosti násobení matic

- Násobení **není komutativní** $A \cdot B \neq B \cdot A$
- Násobení **je asociativní** $A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$
- Transponovaná znásobená matice odpovídá transponování původních matic zvlášť, ale v opačném pořadí

$$(A \cdot B)^T = B^T \cdot A^T$$

- Podobně pro inverzní matici $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$
- Násobení maticí identity $A \cdot I = I \cdot A = A$

Modelovací matice

- Násobení není komutativní => pořadí aplikování transformací je **zásadní**
- Z matematického pohledu se transformace provádí postupným násobením souřadnic objektu základními transformačními maticemi
- Alternativně ze použít jednu matici, která vznikla ze základních matic

$$R \cdot T \cdot v = M \cdot v \qquad \text{kde} \qquad M = R \cdot T$$

Modelovací matice

- Tato matice *M* složená z elementárních transformací se v OpenGL nazývá **modelovací matice (model matrix)**
- Do pipeline posíláme pouze jednu matici = efektivita

Základní (afinní) transformace

- Posun (translace)

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotace

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; R_y = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; R_z = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Škálování (scaling)

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Příklad

- Rotace 2D vrcholu (x = 0.5, y = 0.5) o 90° okolo osy Z

$$\begin{bmatrix} \cos\left(90 \cdot \frac{\pi}{180}\right) & -\sin\left(90 \cdot \frac{\pi}{180}\right) & 0 & 0 \\ \sin\left(90 \cdot \frac{\pi}{180}\right) & \cos\left(90 \cdot \frac{\pi}{180}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix}$$

- Následný posun (0.1 v *x*, -0.2 v *y*, 0.5 v *z*)

$$\begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & -0.2 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.4 \\ 0.3 \\ 0.5 \\ 1 \end{bmatrix}$$

Za použití modelovací matice …

$$M = \begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & -0.2 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\left(90 \cdot \frac{\pi}{180}\right) & -\sin\left(90 \cdot \frac{\pi}{180}\right) & 0 & 0 \\ \sin\left(90 \cdot \frac{\pi}{180}\right) & \cos\left(90 \cdot \frac{\pi}{180}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & -0.2 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0.1 \\ 1 & 0 & 0 & -0.2 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 & 0.1 \\ 1 & 0 & 0 & -0.2 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.4 \\ 0.3 \\ 0.5 \\ 1 \end{bmatrix}$$

Pořadí transformací

- **Je velmi důležité!**
- **Protože násobení matic není komutativní**

$$C' = RT \quad C'' = TR \qquad C' \neq C''$$

- Jak můžeme transformace chápat?
  - 2 přístupy:
  - Globální souřadný systém
  - Lokální souřadný systém

Globální souřadný systém

- Aktuální matice je **C** = **NML**
- Transformovaný vrchol v’ = **C**v = **NML**v

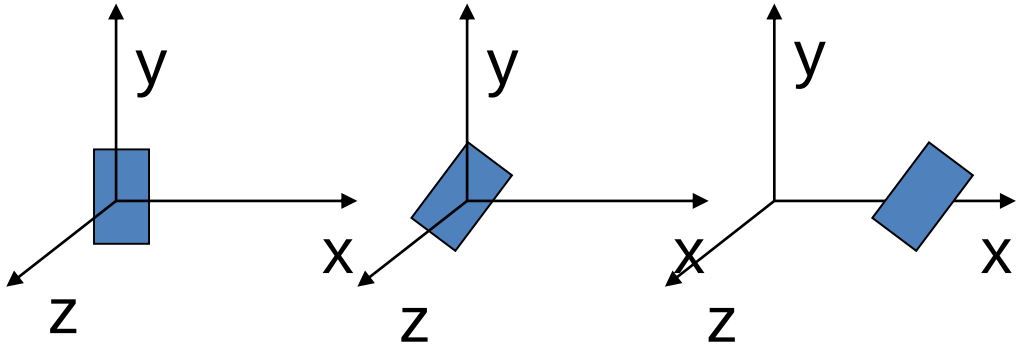
  tj., v’ = **N**(**M**(**L**v))

⇒Vrchol je nejprve transformován transformací **L**!

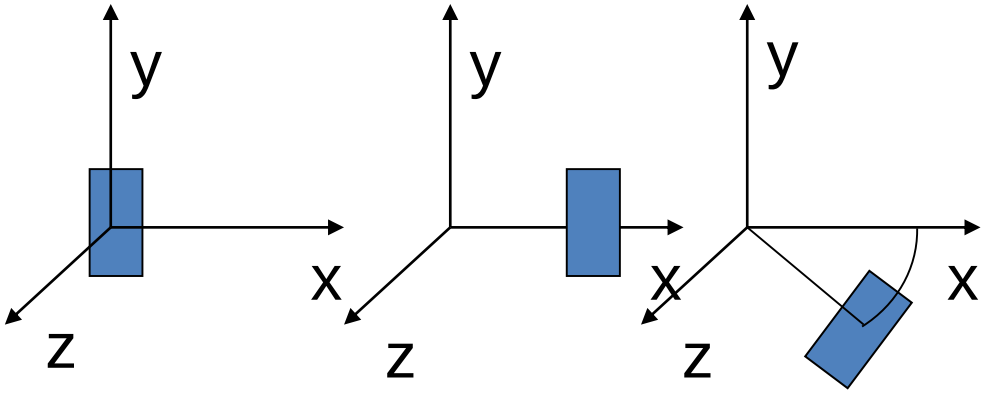
Poslední volaná transformace je ta, která se první aplikuje na vrcholy

Pořadí transformací

- Rotace a poté posun



- Posun a poté rotace

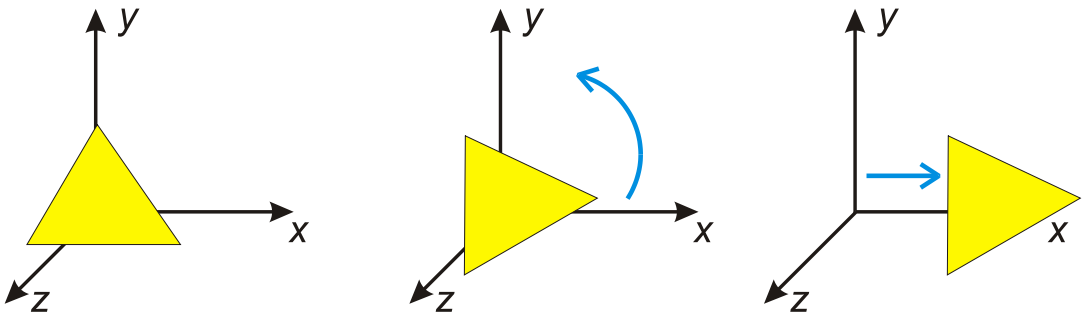


Lokální souřadný systém

- Operace jsou aplikovány na lokální souřadnicový systém
- S operacemi se počítá v přirozeném pořadí

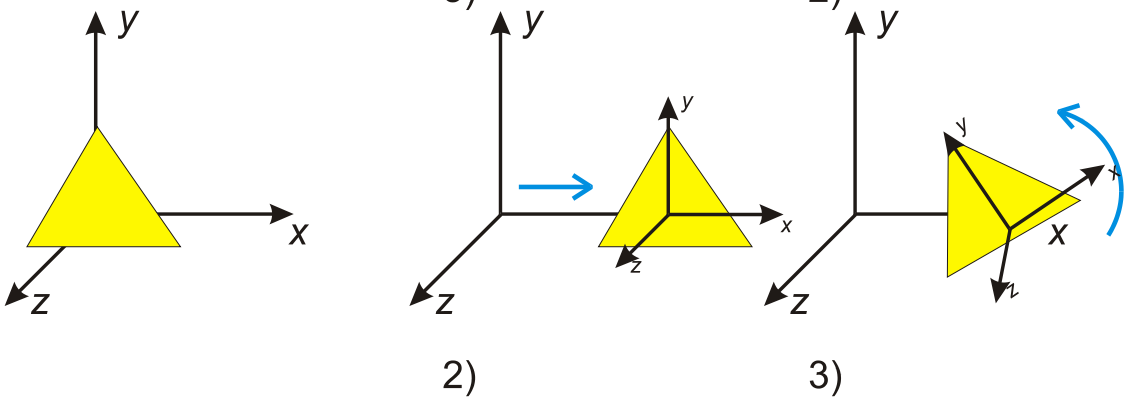
Lokální vs. globální přístup

Globální



3) 2)

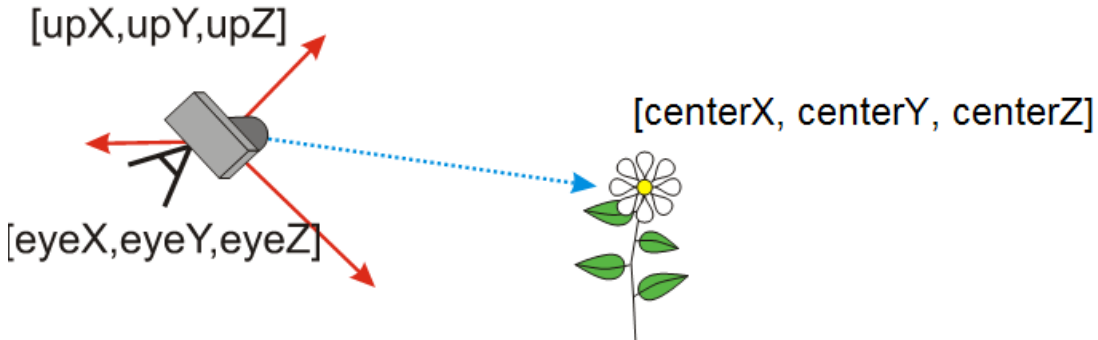
Lokální



2) 3)

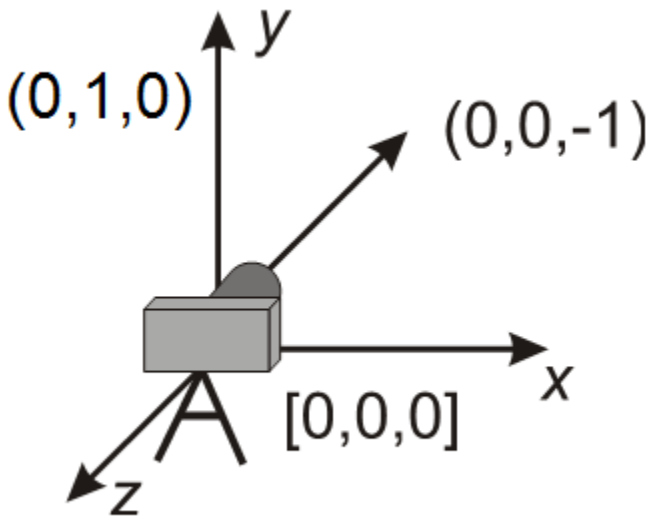
Pohledová matice

- Pohledová matice (view matrix) velikosti 4x4
- Pro vytvoření pohledové matice obsahují knihovny často funkci *LookAt* definující matici pomocí tří parametrů:
  - *eye* = pozice pozorovatele
  - *center* = bod, do kterého se kamera dívá
  - *up* = směr nahoru pozorovatele



Pohledová matice

- Pozorovatel je na ose z, směr pohledu směrem k počátku souřadné soustavy
- Vektor směrem nahoru v kladném směru osy y



Pohledová matice

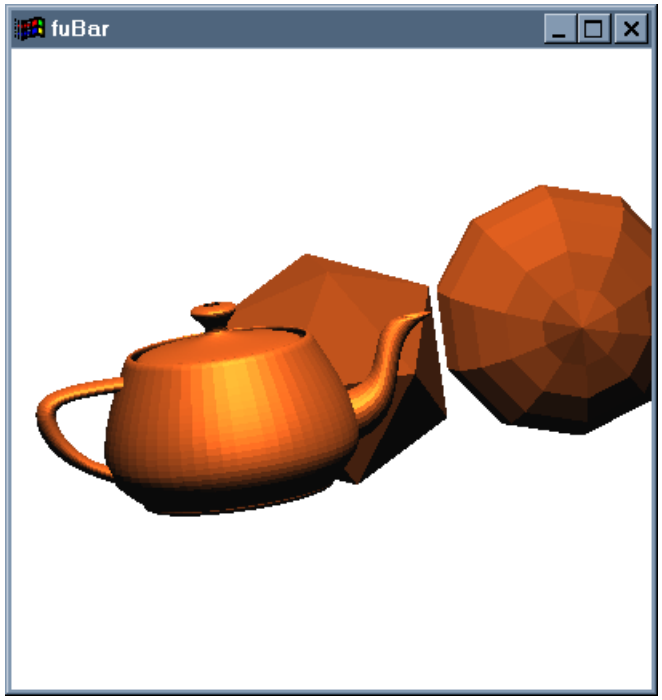
- Aplikace pohledové matice *V*:

$$v' = V \cdot M \cdot v$$

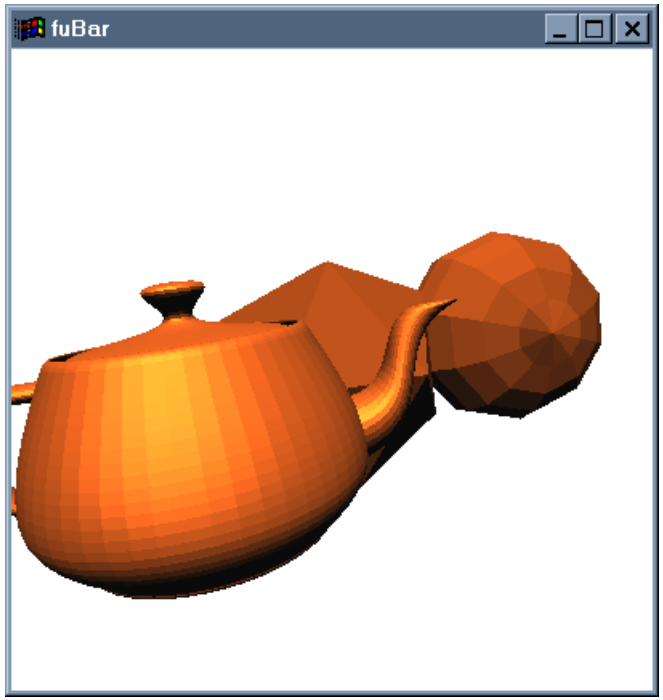
Projekční matice

- Simulace reálného chování scény
  - Objekty blíže kameře se jeví větší
- Objekt pro vykreslení musí být umístěn v pohledovém ořezávacím objemu
- Projekční matice ovlivňuje tento objem
- Dva typy projekce:
  - Perspektivní
  - Ortografická

Projekce



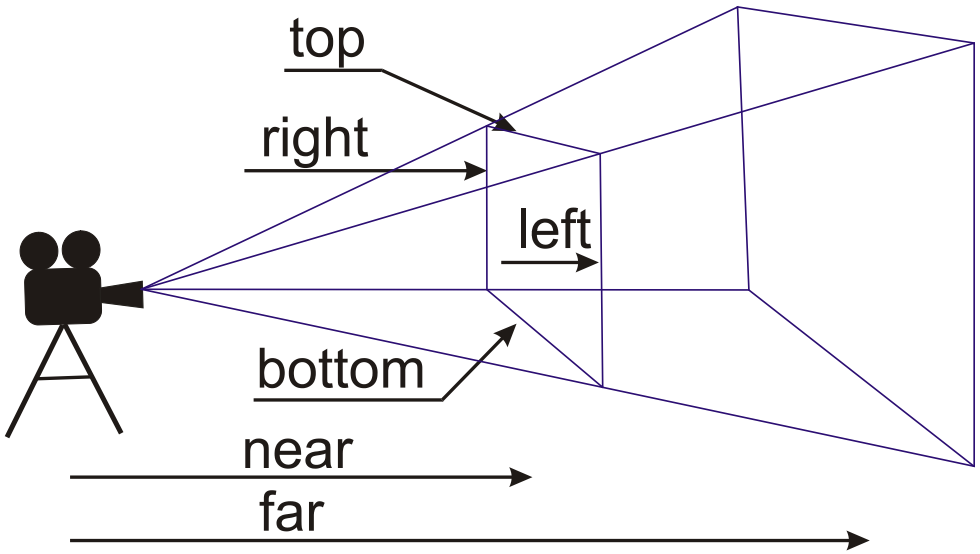
Ortografická



Perspektivní

Perspektivní projekce

$$P = \begin{bmatrix} \frac{2 \cdot near}{right - left} & 0 & \frac{right + left}{right - left} & 0 \\ 0 & \frac{2 \cdot near}{top - bottom} & \frac{top + bottom}{top - bottom} & 0 \\ 0 & 0 & -\frac{far + near}{far - near} & -\frac{2 \cdot far \cdot near}{far - near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



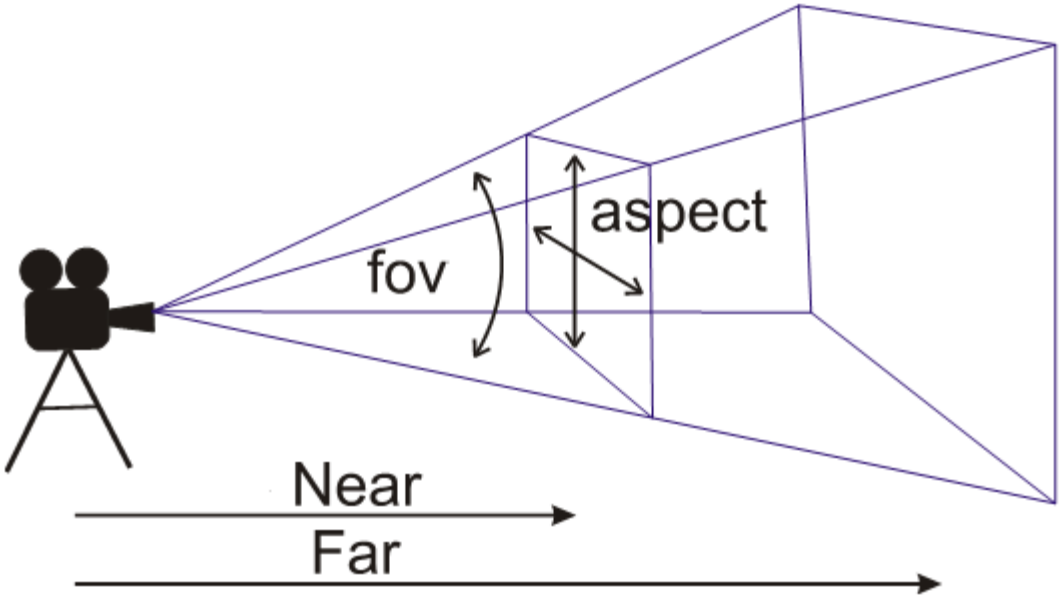
Perspektivní projekce

$$top = near \cdot \tan\left(\frac{\pi}{180} \cdot FOV/2\right)$$

$$bottom = -top$$

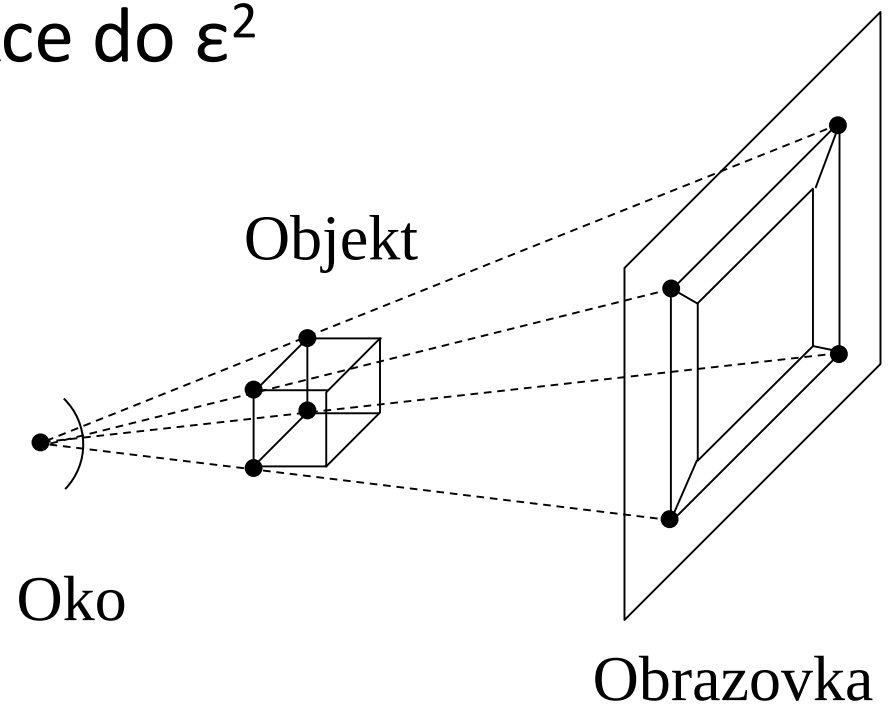
$$right = top \cdot aspect$$

$$left = -right$$



Perspektivní projekce

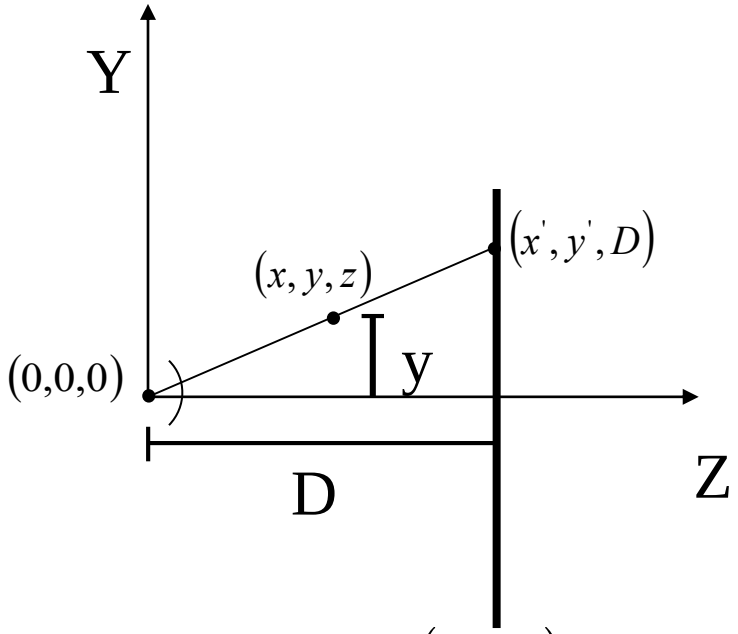
- Vykreslování objektů v ε^3 je prováděno přirozeně použitím jejich projekce do ε^2



- Homogenní souřadnice reprezentují projekci

Perspektivní projekce

- Pohled zboku:



$$y' / y = D / z$$

$$x' / x = D / z$$

$$\left(x', y'\right) = \left(xD / z, yD / z\right)$$

- Bod obrazu $\left(x', y'\right)$ je vypočten z bodu objektu (x, y, z) pomocí podobnosti trojúhelníků.

Perspektivní projekce

$$\left(x', y', z'\right) = \left(xD/z, yD/z, D\right) = \left[xD/z \quad yD/z \quad D \quad 1\right] =$$
$$\cong \left[x \quad y \quad z \quad z/D\right] =$$
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- V okamžiku, kdy se D blíží k nekonečnu, $w' = 1/D$ se blíží k 0 a transformované body se stanou nekonečně vzdálenými.

Matice perspektivní projekce

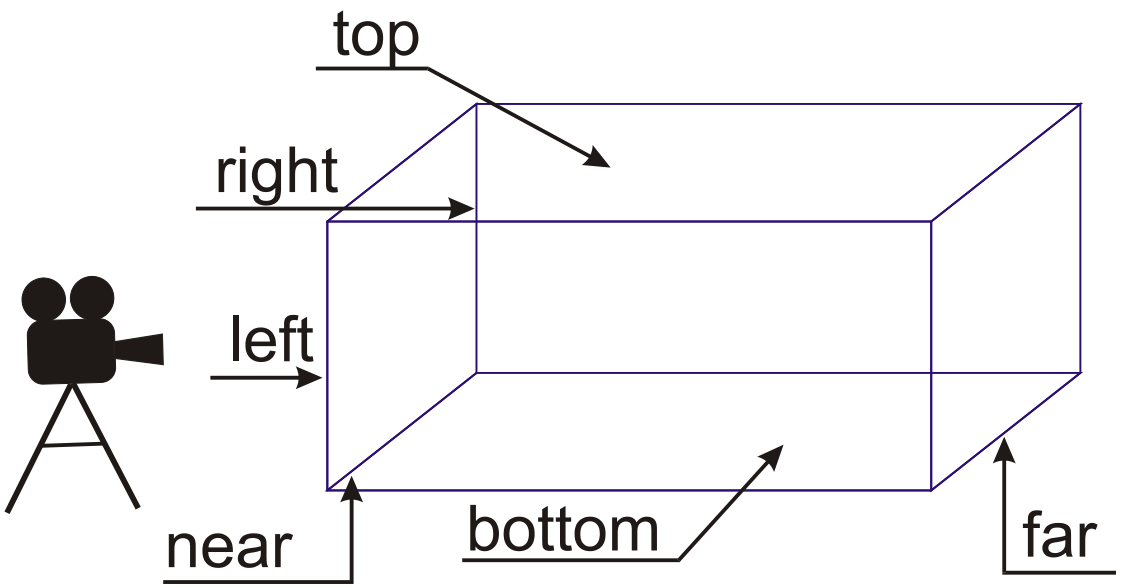
- Vzniká násobením matice pro perspektivu a pro projekci

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & D \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1/D & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{bmatrix}$$

Projekce Perspektiva **Perspektivní projekce**

Ortografická projekce

$$P = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Aplikování projekční matice

- Vynásobení pohledové a modelovací matice

$$v' = P \cdot V \cdot M \cdot v$$

Použití transformací v praxi

- Ukázky používají knihovnu GLM (C++)
- Pro Javu např. knihovna JOML

- Vytvoření modelovací, pohledové a projekční matice

```
glm::mat4 Model, View, Projection;
```

- Po jejich vytvoření všechny obsahují jednotkovou matici

Přesun matic do vertex shaderu

- Na konci inicializace:

```
// Transfer the transformation matrices to the shader program
GLint model = glGetUniformLocation(shaderProgram, "Model" );
glUniformMatrix4fv(model, 1, GL_FALSE, glm::value_ptr(Model));

GLint view = glGetUniformLocation(shaderProgram, "View" );
glUniformMatrix4fv(view, 1, GL_FALSE, glm::value_ptr(View));

GLint projection = glGetUniformLocation(shaderProgram, "Projection" );
glUniformMatrix4fv(projection, 1, GL_FALSE, glm::value_ptr(Projection));
```

Úprava vertex shaderu

```
in vec4 position;
uniform mat4 Model;
uniform mat4 View;
uniform mat4 Projection;

void main() {
  gl_Position = Projection * View * Model * position;
}
```

Nastavení projekce

- Pomocí GLM:

```
glm::mat4 Model, View, Projection;

// Set the projection matrix
Projection = glm::ortho(-4.0f/3.0f, 4.0f/3.0f, -1.0f, 1.0f, -1.0f, 1.0f);
```

Nastavení posunu a rotace

- Pomocí GLM:

```
// Translation
Model = glm::translate(Model, glm::vec3(0.1f, 0.2f, 0.5f));

// Rotation around Oz with 45 degrees
Model = glm::rotate(Model, 0.78f, glm::vec3(0.0f, 0.0f, 1.0f));
```