PV112 Programování grafických aplikací

Jaro 2017

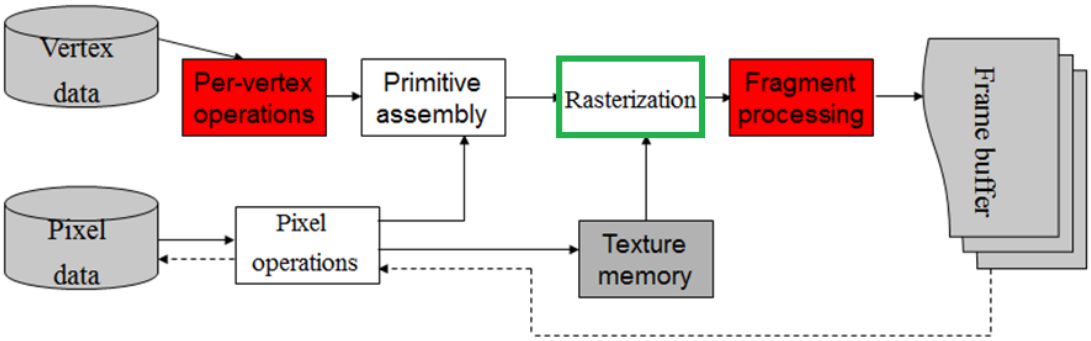
Výukový materiál

---

3. přednáška: Rasterizace, Homogenní souřadnice, Transformace

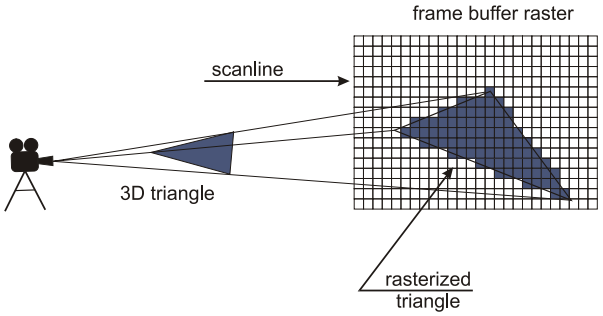
Rasterizace

Nejdříve se znovu vrátíme k nejdražší části blokového diagramu OpenGL a to rasterizaci. Rasterizace je vlastně konverze grafických primitiv do dvoudimenzionálních obrázků.

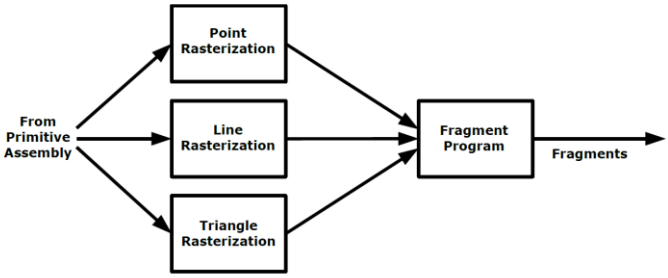


Proces rasterizace má dvě části. V první zjistíme, které obdélníky (pixely) mřížky okna jsou obsazeny. Ve druhé části pak každému obdélníku přiřadíme příslušnou hodnotu barvy a hloubky. Výsledek je poté poslán dále ke zpracování fragment shaderem.

Princip rasterizace



Na tomto obrázku vidíte ilustraci principu rasterizačního procesu, kdy dochází k promítnutí 3D trojúhelníku do rastru framebufferu. Přitom zásadní je pozice pozorovatele vůči zobrazovanému trojúhelníku.



Veškeré vstupní objekty jsou zpracovány rasterizační jednotkou a vznikají fragmenty.

Než jsou hodnoty skutečně uloženy do framebufferu, je nad fragmenty provedena sada operací, které mohou změnit vlastnosti nebo i zcela odstranit dané fragmenty z vykreslování. Tyto změny probíhají ve fragment shaderu, kde probíhá například texturování a výpočet mlhy. Dále zde mohou probíhat operace jako discard pro výpočet průhlednosti, stencil test, depth test (odstranění skrytých částí). Pokud některý z „povolených“ (enabled) testů selže, může to způsobit ukončení další zpracování daného čtverce fragmentu. Pokud testy projdou, následuje blending (míchání), dithering (roztřesení) a logické operace. Nakonec je zpracovaný fragment vykreslen do příslušného bufferu.

Fragment

Fragment je vlastně zlomek objektu v mřížce společně s přiřazenými parametry, jako je například barva nebo hloubka (hodnota Z). Tyto parametry se nazývají „fragment’s associated data“.

Fragment je definován dolním levým rohem, který leží v mřížce celočíselných souřadnic. Operace v rasterizačním procesu rovněž pracují se středem fragmentu, který je posunut o ½ od levého dolního rohu.

Rasterizace bodů

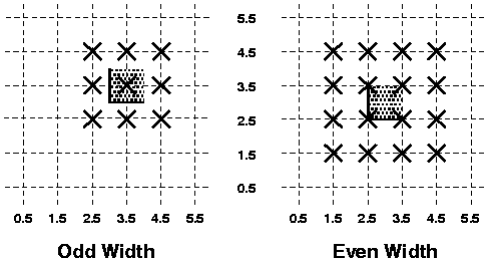
Rasterizace bodů produkuje fragment pro každý pixel framebufferu, jehož střed leží uvnitř čtverce se středem v bodu $[x_w,y_w]$, jehož délka strany je shodná s aktuální velikostí bodu.

Defaultně je bod rasterizován ořezáním jeho x_w a y_w souřadnic na hodnoty typu integer.

Rasterizace bodů je řízena funkcí void glPointsize(float size), kde *size* určuje velikost vykreslovaného bodu. Defaultní hodnota je 1.0. Hodnota menší nebo rovna nule způsobí chybu INVALID_VALUE. Velikost bodu lze nastavit alternativně i povolením změny velikosti bodu pomocí glEnable(GL_PROGRAM_POINT_SIZE) a poté zavoláním gl_Pointsize = size, kde *size* je opět požadovaná velikost vykreslovaného bodu.

Při rasterizaci bodu o velikosti větší než 1.0 je skutečná šířka bodu určena zaokrouhlením zadané šířky na nejbližší hodnotu typu integer a dále uzavřením na maximální možnou hodnotu šířky bodu (liší se podle dané implementace).

Jestliže je výsledek zaokrouhlení 0, pak se výsledek bere jako hodnota 1. Když je výsledek **lichý**, pak je střed fragmentu (x,y) = ($\lfloor x_w \rfloor + ½$, $\lfloor y_w \rfloor + ½$) spočten z pozice $[x_w,y_w]$ vrcholu a čtvercová mřížka liché šířky se středem [x,y] definuje středy rasterizovaných fragmentů (jinak řečeno, středy fragmentů leží v tzv. half-integer souřadném systému). Je-li výsledek **sudý**, je středový bod definován jako (x,y) = ($\lfloor x_w + ½ \rfloor$, $\lfloor y_w + ½ \rfloor$).



Na obrázku je znázorněn příklad rasterizace „velkých“ bodů, kde křížky znázorňují středy fragmentů vytvořené rasterizací pro jakýkoliv bod ležící v tečkované oblasti. Tečkovaná oblast leží v prostoru half-integer souřadnic.

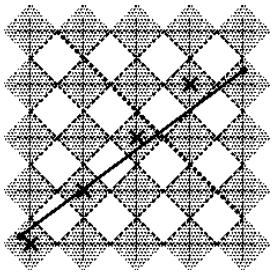
Rasterizace úseček

Rasterizace úsečky je opět řízena šířkou úsečky. Rasterizace úsečky začíná určením typu úsečky – jestli je tzv. x-hlavní (x-major) nebo y-hlavní (y-major). Toto rozdělení závisí na sklonu úsečky. x-hlavní úsečka má sklon v rozmezí uzavřeného intervalu [-1, 1]. Všechny ostatní úsečky jsou y-hlavní.

OpenGL využívá tzv. „diamond-exit“ pravidlo pro určení fragmentů, které vznikají rasterizací úsečky. Pro každý fragment *f* se středem $[x_f,y_f]$ definujeme region tvaru diamantu, který je průsečíkem čtyř polorovin:

$R_f = \{(x,y) \mid |x - x_f| + |y - y_f| < ½\}$

Každá úsečka začínající v bodě p_a a končící v bodě p_b vytváří takové fragmenty *f*, pro které úsečka protíná R_f.



Pro generování úseček je použit Bressenhamův algoritmus s jednou modifikací – takto vytvořená úsečka je tzv. „polootevřená“, což znamená, že koncový fragment (odpovídající p_b) není vykreslen. To odpovídá situaci, kdy vykreslujeme sadu na sebe navazujících úseček. Tímto přístupem vykreslíme koncové body pouze jednou (klasický Bressenham vykreslí v tomto případě koncové body dvakrát).

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

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

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

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.

Homogenní souřadnice umožňují reprezentovat všechny grafické operace pomocí násobení matic. Rotaci a scaling ve 2D můžeme reprezentovat násobení maticí 2x2, pro translaci musíme zavést třetí, homogenní souřadnici.

Důvodem jejich použití je skutečnost, že v klasickém Euklidovském nebo Kartézském prostoru se nemohou dvě rovnoběžky protnout. Pokud se ale dostaneme do prostoru projektivní roviny (viz obrázek), vidíme, že se zvětšující se vzdáleností od pozorovatele se dvě

rovnoběžné přímky protínají v nekonečnu. Problém definice tohoto jevu byl vyřešen zavedením homogenních souřadnic.

Homogenní souřadnice umožňují reprezentovat n-dimenzionální bod n+1 hodnotami. Pro vytvoření 2D homogenních souřadnic stačí k souřadnicím (x, y) přidat třetí proměnnou w. Tedy bod v kartézských souřadnicích (X,Y) odpovídá bodu (x, y, w) v homogenních souřadnicích. Převod z homogenních souřadnic na kartézské probíhá následovně:

$X = x/w$

$Y = y/w$

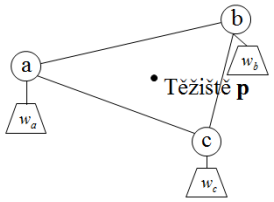
Jako příklad si uveďme bod (1, 2), který odpovídá homogenním souřadnicím (1, 2, 1). Pokud se tento bod pohybuje směrem k nekonečnu, v kartézských souřadnicích bychom dostali nesmyslné vyjádření (∞, ∞). V homogenních souřadnicích takovýto bod vyjádříme jednoduše jako (1, 2, 0).

Definice homogenních souřadnic

Nejdříve začneme s příklady v dvourozměrném prostoru a nabyté znalosti využijeme k vysvětlení homogenních souřadnic ve třech dimenzích.

Definice 1:

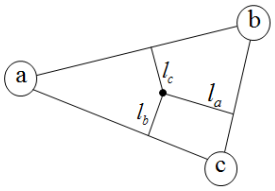
Jako první zavedl v projektivní geometrii homogenních souřadnice August Ferdinand Möbius (1790 - 1868). Mějme pevně daný trojúhelník v rovině. Množina homogenních souřadnic pro bod *p* je definována jako hmotnosti přiřazené vrcholům trojúhelníka takové, že *p* se stane těžištěm trojúhelníka.



Bod *p* je spočten jako (w_aa, w_bb, w_cc), kde w_a, w_b a w_c se nazývají barycentrické souřadnice a platí $w_a + w_b + w_c = 1$.

Definice 2:

Další definici homogenních souřadnic vytvořil německý matematik a fyzik Julius Plücker (1801 - 1868). Ve své definici pracuje se vzdálenostmi z daného bodu ke hranám pevného trojúhelníka.



Plückerovy souřadnice: **p** = $(l_a \quad l_b \quad l_c)$

n+1 souřadnic reprezentuje n-dimenzionální bod.

Jednou z nejdůležitějších vlastností tohoto systému je jeho invariance vůči scalingu (změně velikosti). Scaling Möbiových vah ani velikosti Plückerových souřadnic nezmění pozici bodu ***p***. To si ukážeme na následujícím příkladě:

Převod homogenních souřadnic (x, y, w) na kartézské (X, Y) probíhá pomocí již známého převodu

X = x/w

Y = y/w

Vezmeme-li například souřadnice:

(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)

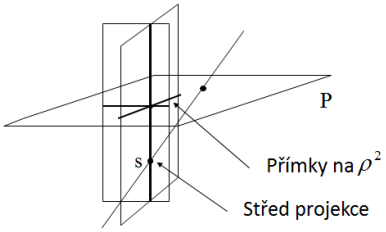
Z toho je vidět, že všechny zmíněné body odpovídají stejnému bodu v euklidovských souřadnicích. Proto jsou nazývány „homogenní“. Jinak řečeno, homogenní souřadnice jsou invariantní vůči scalingu.

Projektivní rovina

Plücker si uvědomil tuto invarianci homogenních souřadnic vůči scalingu a dále považoval homogenní bod s hodnotou w = 0 odpovídající bodu v normální rovině, který je nekonečně daleko.

Normální rovina doplněná o body v nekonečnu se nazývá **projektivní rovina**. V projektivní rovině se libovolné dvě přímky protínají (tedy i ty rovnoběžné).

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 geometrie je ve skutečnosti geometrií "snímání".



Projektivní rovinu můžeme popsat 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***.

Podle konvence paralelní přímky protínají **P** v ideálním (nekonečně vzdáleném) bodě.

Pak projektivní rovina ρ^2 obsahuje současně ideální body a ideální přímky.

Dvojrozměrný projektivní prostor

Každé z následujících tvrzení definuje 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.

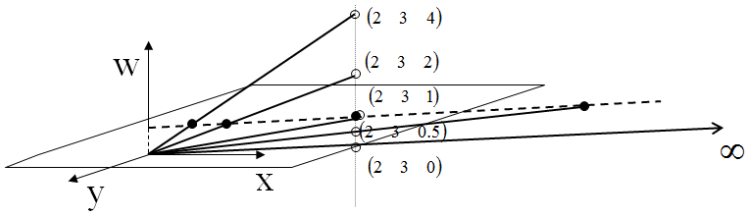
Prakticky to znamená:

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 $x' \approx x/w$, $y' \approx y/w$.

- 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:

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

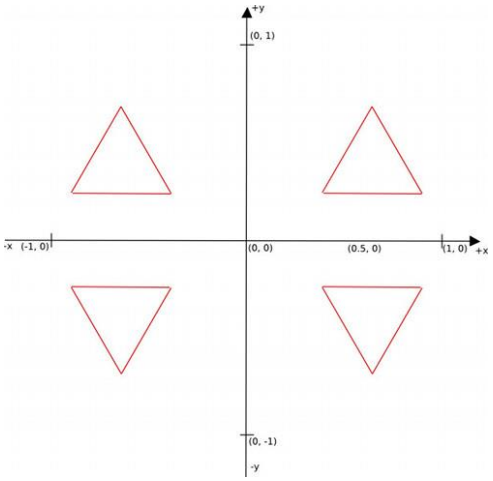
- Protože homogenní body reprezentují projekci, mohou také reprezentovat body v nekonečnu.



Důležitý a praktický aspekt systému homogenních souřadnic je unifikace translace, scalingu a rotace geometrických objektů.

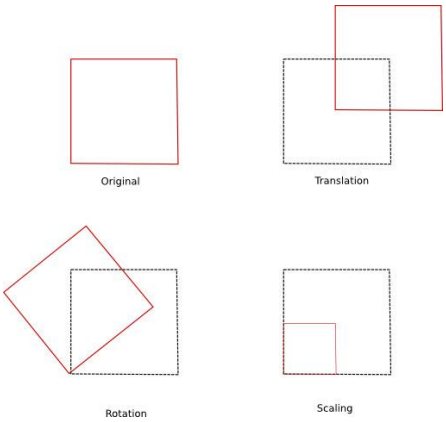
Motivace pro transformace

Nyní si na jednoduchém příkladě ukážeme, proč jsou důležité v OpenGL matice a k čemu se používají. Jsou na nich založeny veškeré transformace s objekty ve scéně. Představme si situaci, že chceme vykreslit čtyři trojúhelníky jako na obrázku.

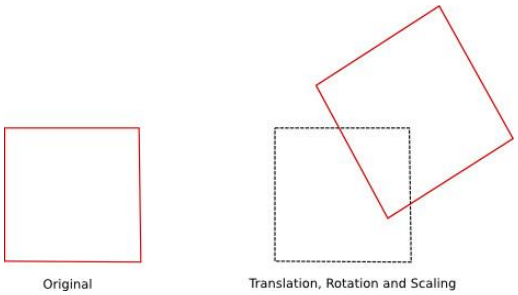


Trojúhelníky mají různou pozici a orientaci. Jedním z možných řešení je samozřejmě uložit si souřadnice vrcholů všech trojúhelníků a použít postup, jaký jsme si ukazovali na minulé přednášce. Ale co kdybychom měli ve scéně trojúhelníků například 20? V tom případě bychom museli do našeho pole uložit 60 pozic vrcholů. A obecně objekty bývají mnohem složitější. Pokud tedy chceme zkopírovat daný objekt na mnoho různých pozic na obrazovce a v různých orientacích, potřebujeme lepší způsob než zadávání všech vrcholů do pole.

Tento problém lze vyřešit za použití maticové algebry. Místo uložení daného objektů N-krát jej uložíme pouze jednou a použijeme geometrické transformace posunutí, rotace a škálování, abychom jej umístili na požadovaným způsobem (na požadovanou pozici, správně orotovaný a se správnou velikostí). Obrázek znázorňuje prinicip těchto transformací na jednoduchém příkladě čtverce.



Tyto jednoduché transformace můžeme skládat a tím dosáhneme složitějších efektů (jako například na obrázku).



Jak tedy dosáhneme těchto transformací v OpenGL? Jako obvykle přesuneme souřadnice objektu do OpenGL bufferu a ve vertex shaderu vynásobíme tyto souřadnice *transformační maticí*.

Výklad teorie maticové algebry je mimo rozsah předmětu, nicméně je nutné ji znát. Zde uvedeme pouze některé důležité vlastnosti maticového násobení, protože právě násobení matic nás bude v OpenGL zajímat nejvíc.

- Maticové násobení není komutativní, záleží tedy na pořadí matic při jejich násobení.
  $A \cdot B \neq B \cdot A$
- Násobení je asociativní, můžeme tedy operaci provádět po různých částech (ale při zachování pořadí matic).
  $A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$
- Pokud transponujeme výslednou matici vzniklou vynásobením dvou matic, pak výsledek je stejný jako kdybychom nejdříve matice transponovali a pak je vynásobili, ale v opačném pořadí.
  $(A \cdot B)^T = B^T \cdot A^T$
- Podobné pravidlo platí i pro inverzní matice.
  $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$
- Násobení maticí identity nemění původní matici.
  $A \cdot I = I \cdot A = A$

Modelovací matice (model matrix)

Jak již bylo řečeno, násobení matic není komutativní, proto je naprosto **zásadní** pořadí, v jakém transformace aplikujeme na objekt. Z matematického pohledu se transformace provádí postupným násobením souřadnic objektu základními transformačními maticemi. Alternativně se dá proces transformace provést i takovým způsobem, že si nejdříve připravíme matici složenou ze základních transformací (jejich vynásobením) a poté tuto finální transformační matici aplikovat na objekt (jeho vrcholy ve formě sloupcového vektoru, v rovnici označené jako *v*).

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

Aby byla transformace korektní, je třeba ji aplikovat na všechny vrcholy objektu.

Matice M, kterou jsme v předchozí rovnici násobili každý vrchol objektu, se v OpenGL nazývá **modelovací matice (model matrix)**. Místo toho, abychom do pipeline posílali více transformačních matic, pošleme pouze matici M a tím dojde k zefektivnění.

Základní (affinní) transformace

Mezi základní transformace patří posun, rotace a scaling. Při posunu dojde k vynásobení vektoru v pozic vrcholu maticí daného tvaru, což způsobí posun v o hodnotu T_x v ose x, T_y v ose y a T_z v ose z.

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

Pro rotaci máme tři různé matice lišící se tím, podle které osy rotujeme. Například matice R_z orotuje vektor v o *alpha* stupňů okolo osy z proti směru hodinových ručiček.

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

Matice škálování pak naškáluje vektor *v* hodnotou S_x ve směru osy x, S_y ve směru osy y a S_z ve směru osy z.

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

Všimněte si, že matice jsou o rozměrech 4x4. Vektor *v* je tedy vyjádřen v homogenních souřadnicích.

Uveďme si příklad rotace vrcholu na pozici x = 0.5 a y = 0.5 o 90° okolo osy z.

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

Následně provedeme ještě posun o 0.1 v ose x, -0.2 v ose y a 0.5 v ose z.

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

Tyto transformace jsme provedli postupným násobením s vektorem reprezentujícím vrchol.

Totéž lze dosáhnout tím, že nejdříve sestavíme modelovací matici složenou z individuálních transformací, kterou následně v jediném kroku aplikujeme na vrchol.

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

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

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

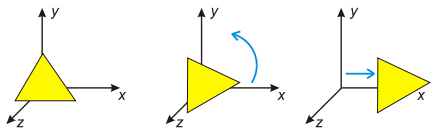
Jak již bylo řečeno, pořadí prováděných transformací je velmi důležité, protože násobení matic není obecně komutativní. Transformace můžeme chápat dvěma různými způsoby: Prvním přístupem je **globální souřadný systém** (the grand fixed coordinate system), druhý přístup je použití **lokálního souřadného systému** (local coordinate systém tied to object).

Pokud máme v globálním souřadném systému matici C=NML a aplikujeme ji na vrchol *v*, transformovaný vrchol *v'* vypadá následovně: *v'* = N(M(L*v*)). Vrchol *v* je tedy nejprve transformován transformací L! Tedy **poslední volaná transformace je ta, která se jako první aplikuje na vrcholy**.

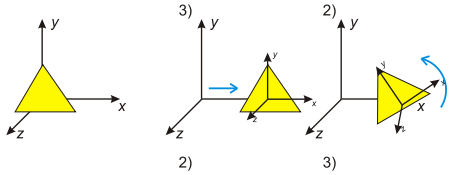
Oproti tomu v **lokálním souřadném systému** se s operacemi počítá v přirozeném pořadí, jak jdou za sebou. Tedy v předchozím příkladě se na vrchol *v* nejdříve aplikuje transformace N. S tímto konceptem pracuje i OpenGL, tedy v pořadí, v jakém jsou zadány transformační matice, se aplikují na jednotlivé vrcholy.

Pro porovnání globálního a lokálního přístupu si uvedeme následující obrázek:

Globální

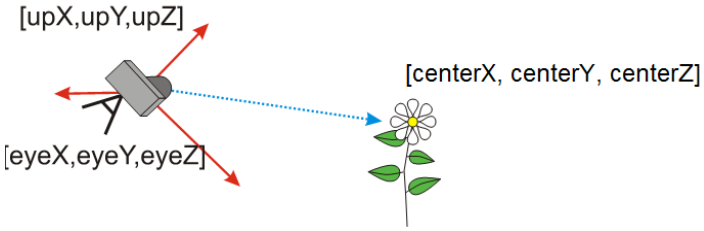


Lokální

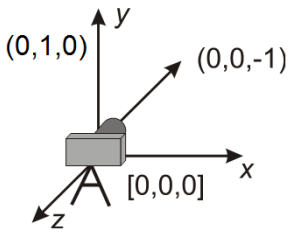


Pohledová matice (view matrix)

Pohledová matice (view matrix) v OpenGL kontroluje způsob náhledu na scénu. Pohledová matice je velikosti 4x4 a je jednoznačně určena třemi parametry a nastavuje se funkcí LookAt, která je přítomna v některé z nadstavbových knihoven pro zvolený programovací jazyk (více na cvičeních). Tyto parametry jsou pozice pozorovatele (*eye*), bod, do kterého se kamera dívá (*center*) a směr nahoru pozorovatele (*up*)(definující orientaci pozorovatele).



V OpenGL je pozorovatel implicitně umístěn na ose z a směr pohledu je k počátku souřadné soustavy. Vektor definující orientaci pohledu (up vektor) směřuje v kladném směru osy y.



Pohledovou matici V aplikujeme na modelovací matici a tu pak na vektor reprezentující vrchol. Rovnice tedy definuje aplikování pohledové a modelovací matice na vrchol.

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

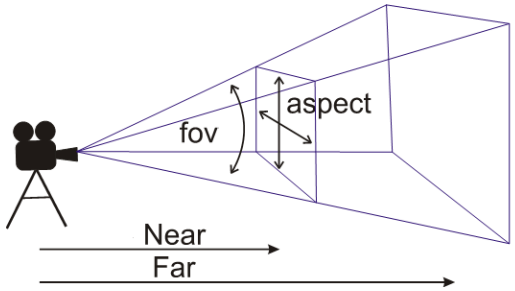
Projekční matice (projection matrix)

V OpenGL má objekt implicitně stále stejnou velikost, bez ohledu na pozici kamery. To ale odporuje reálnému světu, kdy se objekt blíže pozorovateli (kameře) jeví větší. Dalším problémem defaultního nastavení OpenGL je skutečnost, že pro vykreslení objektu na obrazovku je nutné tento objekt umístit dovnitř krychle o rozměrech

[-1,+1] x [-1,+1] x [-1,+1]. Cokoliv mimo tento rozsah bude ořezáno. Tento defaultní ořezávací objem je možné virtuálně zvětšit nebo zmenšit pomocí projekční matice. Tento objem může mít podobu komolého jehlanu (v případě perspektivní projekce) nebo kvádru (v případě ortografické projekce). Ortografická projekce nemění velikost objektu s ohledem na pozici kamery. Toto chování je vhodné u CAD systémů či 2D her.

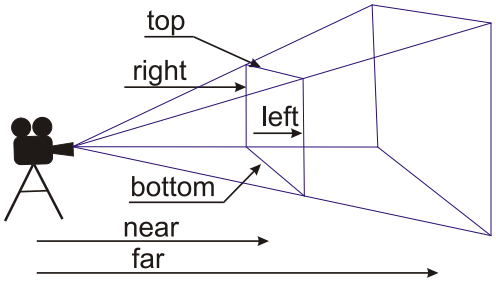
Perspektivní projekce

Matice perspektivní projekce je často specifikována čtyřmi parametry. Parametr *fov* určuje pohledový úhel, *aspect* je poměr výšky a šířky pohledového jehlanu a parametry *Near* a *Far* určují pozici přední a zadní ořezávací roviny. Vztah parametrů je následující:



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

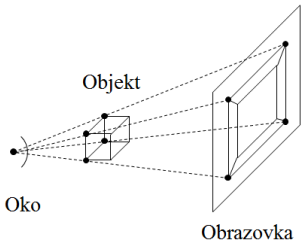
Matice perspektivní projekce vypadá následujícím způsobem. Jednotlivé parametry představují hranice ořezávacího objemu.



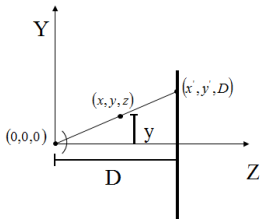
$$P = \begin{bmatrix} \frac{2 \cdot near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2 \cdot near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2 \cdot far \cdot near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspektivní projekce slouží pro přirozeně vypadající promítnutí 3D objektu na 2D obrazovku. Perspektivní projekce je nezbytná pro zobrazení jednoduchých grafických primitiv, jako například polygonů či úseček.

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



Pohled zboku:



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

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

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

Bod obrazu (x', y') je vypočten z bodu objektu (x, y, z) pomocí podobnosti trojúhelníků.

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

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

Perspektivní transformace

Díky lineární závislosti třetího a čtvrtého sloupce matice ztrácí perspektivní projekce hloubkovou informaci.

$$(x', y', z') = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Zavedením nenulového výrazu, např. -1, do třetího sloupce neovlivní x'a y', ale z' bude $D - D / z$

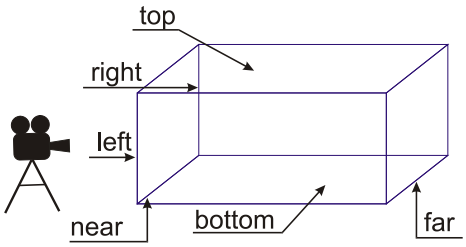
Smysl tohoto dalšího výrazu je komprese Euklidovského prostoru $z \in [1, \infty]$ na $z' \in [0, D]$.

$$(x', y', z') = (xD / z, yD / z, D - D / z) \cong [x \quad y \quad z - 1 \quad z / D] =$$

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

Ortografická projekce

Matice ortografické projekce vypadá následujícím způsobem. Parametry *right*, *left*, *far*, *near*, *top*, *bottom* reprezentují pozice ořezávacích rovin, které definují pohledový kvádr.



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