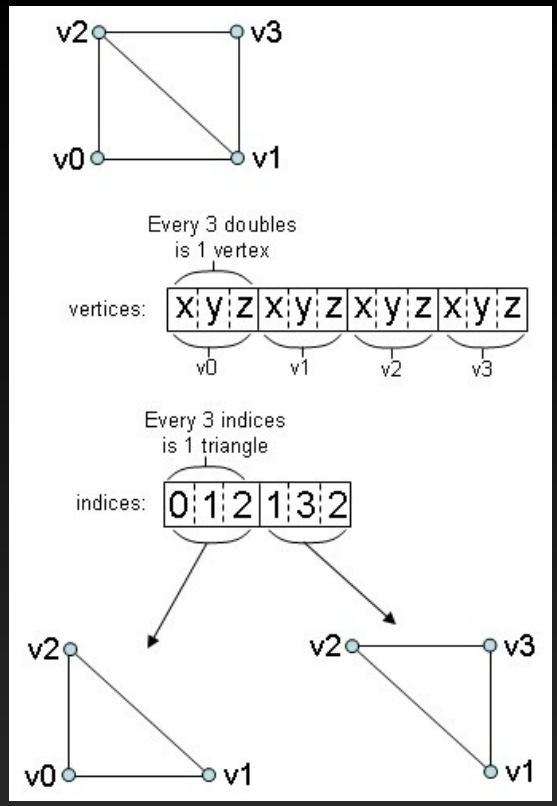
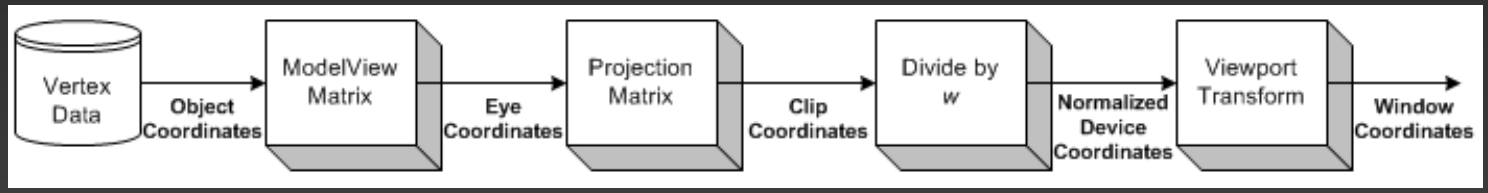
OpenGL transformace

Opakování



- Reprezentace
  - 3D rastr
  - obálka
    - vertex
    - hrana
    - ploška
- 3D zobrazování
  - načtení a transformace souřadnic zadaných vertexů



  - rasterizace
  - výpočet barvy fragmentu
  - průhlednost a zakrývání podle Z

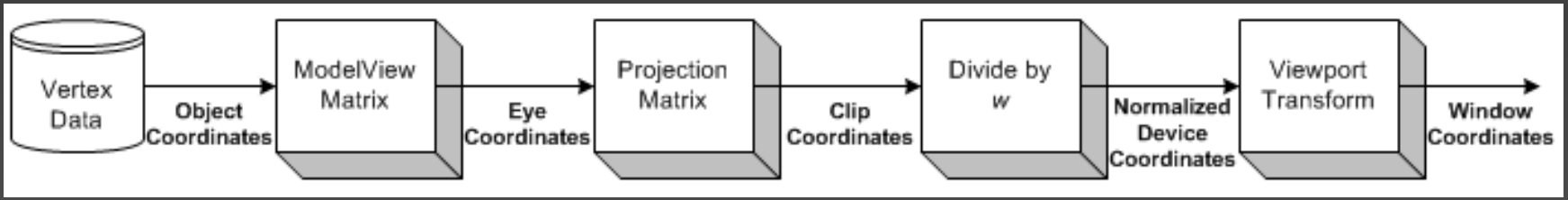
Homogenní souřadnice

- Homogenní souřadnice
  - (x, y, z, w), standardně w = 1
  - homogenní vrchol (x, y, z, w)
    odpovídá trojrozměrnému (x/w, y/w, z/w)
  - pokud *w=0.0,* dělení nulou, body jsou v nekonečnu
- Vynechaná hodnota se doplňuje
  - z = 0.0
  - w = 1.0

Transformace

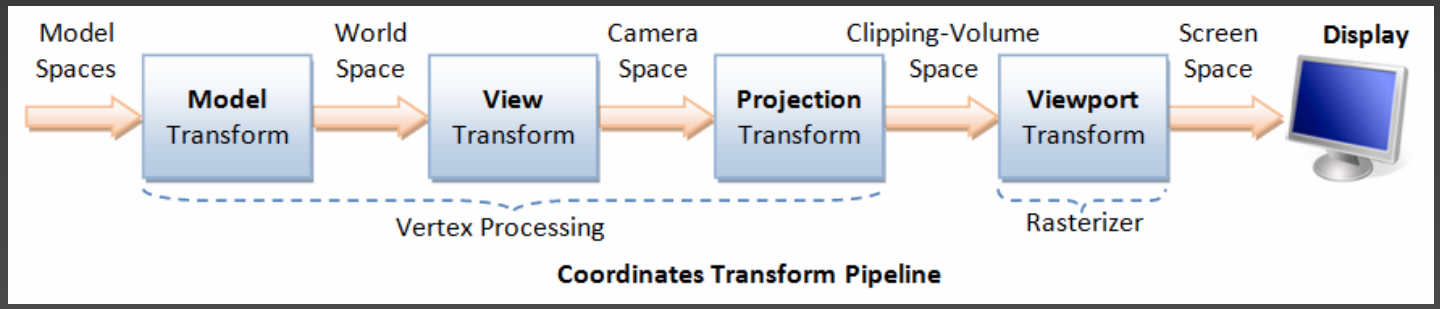
- Jak rozmístit a orientovat tělesa v prostoru?
- Jak určit bod pohledu a pozorovaný prostor?

- Modelování a pohledy – modelovací a pohledová (modelview) matice
  - posun, rotace, změna měřítka, zrcadlení
    - relativita: pohyb kamery X pohyb scény
- Projekce – projekční (projection) matice
  - ortografická (zachovává rozměry, CAD)
  - perspektivní (deformuje, přirozené vidění)
  - ořez pomocí ořezových rovin
- Zobrazení na 2D monitoru – viewport matice



Analogie k fotoaparátu

1) Nastavení stativu, zaměření na scénu
   - pohledová transformace
2) Naaranžování scény, kompozice
   - modelovací transformace
3) Nastavení zoomu, výběr objektivu
   - projekční transformace
4) Volba rozlišení a formátu na digit. fotoaparátu
   - zobrazovací transformace



Transformační matice

- Vytvoření prázdné transformace (identita)

  glm::mat4 my_m;
- Vytvoření transformační matice

  glm::mat4 m_proj = glm::perspective(…);
  glm::mat4 m_translate = glm::translate(…);
  glm::mat4 m_rotate = glm::rotate(…);
  glm::mat4 m_scale = glm::scale(…);
- Nahrání matice do GPU
  - Fixed pipeline
    - Nutné vybrat správnou matici
      glMatrixMode(typ_matice)
      - Modelování a pohledy = GL_MODELVIEW
      - Projekce = GL_PROJECTION
    - Nahrát pomocí glLoadMatrixf(glm::value_ptr(my_m));
  - Shadery
    - glUniformMatrix4fv(location,count, transpose,value_ptr);
    - glUniformMatrix4fv(my_m, 1, GL_FALSE, glm::value_ptr(trans));

Viewport matice

- Zobrazovací transformace

  glViewPort(lev_dol_x, lev_dol_y, šířka, výška)
- Transformace ze souřadnic -1..1 do rozsahu definovaného viewportem
- Obdélník pixelů v okně, do kterého je mapován výsledný obraz

Trocha matematiky

- V rovině

$$x' = x \cdot a_{11} + y \cdot a_{12}$$
$$y' = x \cdot a_{21} + y \cdot a_{22}$$

$$x' = x + t_x$$
$$y' = y + t_y$$

- Maticově

$$v' = v + t$$
$$v' = \boldsymbol{M} \cdot v$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$v' = \boldsymbol{M} \cdot v$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

v prostoru je transformační matice **M** rozměru 4x4

- Maticové násobení není komutativní **A**x**B** ≠ **B**x**A**
  - Pozor na skládání, rozdílný výsledek pokud
    - nejprve posunem a pak rotujem
    - nejprve rotujem a pak posunem

Transformace

$$\underset{(translation)}{Posun}: \boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underset{(scale)}{Měřítko}: \boldsymbol{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformace

Rotace kolem osy x , osy y , osy z :

$$glm::rotate(a,1,0,0):\boldsymbol{R}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(a) & -\sin(a) & 0\\ 0 & \sin(a) & \cos(a) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$glm::rotate(a,0,1,0):\boldsymbol{R}=\begin{bmatrix}\cos(a) & 0 & \sin(a) & 0\\ 0 & 1 & 0 & 0\\ -\sin(a) & 0 & \cos(a) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$glm::rotate(a,0,0,1):\boldsymbol{R}=\begin{bmatrix}\cos(a) & -\sin(a) & 0 & 0\\ \sin(a) & \cos(a) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Obecná rotace kolem jednotkového vektoru $[r_x, r_y, r_z]$:

$$glm::rotate(a,r_x,r_y,r_z):\boldsymbol{R}=\begin{bmatrix}\cos(a)+r_x^2(1-\cos(a)) & r_x r_y(1-\cos(a))-r_z\sin(a) & r_x r_z(1-\cos(a))+r_y\sin(a) & 0\\ r_y r_x(1-\cos(a))+r_z\sin(a) & \cos(a)+r_y^2(1-\cos(a)) & r_y r_z(1-\cos(a))-r_x\sin(a) & 0\\ r_z r_x(1-\cos(a))-r_y\sin(a) & r_z r_y(1-\cos(a))+r_x\sin(a) & \cos(a)+r_z^2(1-\cos(a)) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Skládání transformací

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = N \times M \times L \times \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = NMLv = N(M(Lv))$$

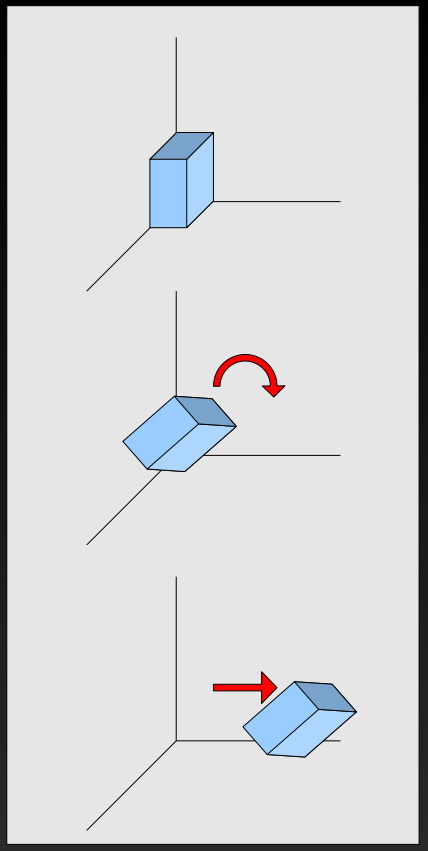
Transformace aplikovány „pozadu“!

Nejprve L, pak M, nakonec N
(na rozdíl od pořadí ve zdrojovém kódu).

- Skládání GLM

```
glm::mat4 L = glm::...;
glm::mat4 M = glm::...;
glm::mat4 N = glm::...;
glm::mat4 all = N * M * L;

glUniformMatrix4fv(location, 1, GL_FALSE,glm::value_ptr(m));
(glLoadMatrixf(glm::value_ptr(my_m));)

//emit vertex
//...
```

Pořadí skládání



- Úkol: mít kvádr natočený a posunutý podle x
- Myšlený základní, pevný systém souřadnic
  - rotujem kvádřík
  - posunem podle x
  - ZADAT V OPAČNÉM POŘADÍ
- Myšlený lokální systém souřadnic
  - vykreslíme kvádr v počátku
  - odsuneme soustavu i s objektem podél osy
  - rotujeme celou soustavu souřadnou
  - myšlená souřadná soustava spojená s kresleným objektem, všechny operace relativně vůči ní
  - ZADAT V TOMTO POŘADÍ

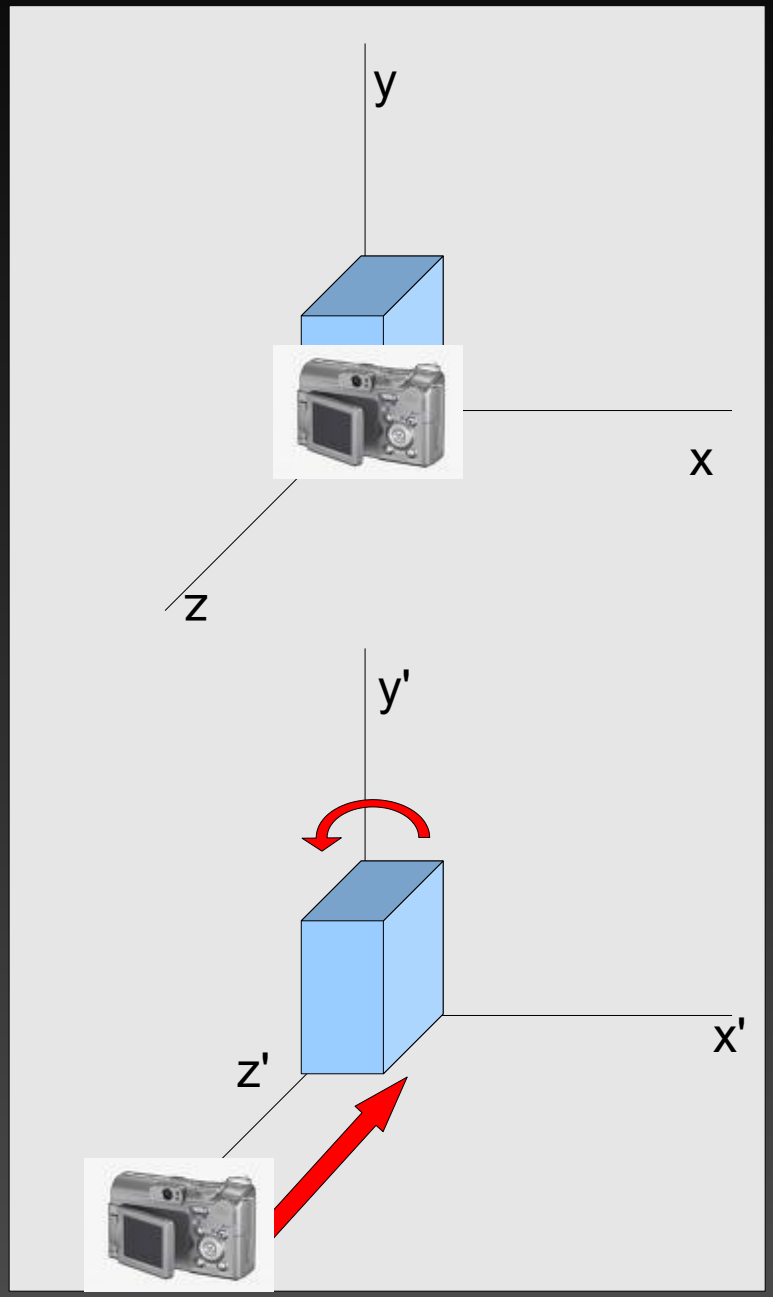
```
glm::mat4 m = glm::mat4(1.0f) *
                  transl_m * rot_m;

(glMatrixMode(GL_MODELVIEW);)
(glLoadMatrixf(glm::value_ptr(my_m));)

glUniformMatrix4fv(location,1,
          GL_FALSE,glm::value_ptr(m));

kresli_kvadr();
```

Příklad modelování

- Kamera na počátku
  - v bodě [0,0,0]
  - orientace směrem -Z

```
m = glm::translate(m,0,0,-5)

glm::rotate(m,90,0,1,0)
```

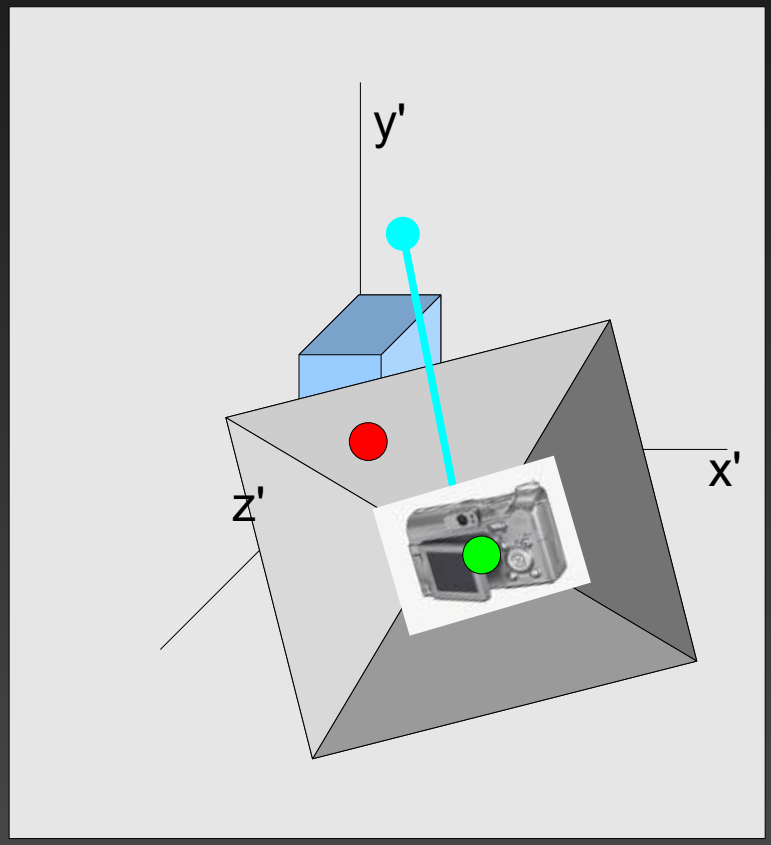
- Model/View ekvivalence
  - stejný výsledek
    - posun kamery
    - posun objektu (scény)



Příklad pohledu

glm::mat4 glm::lookAt(glm::vec3(camposx, camposy, camposz),
glm::vec3(centerx, centery, centerz),
glm::vec3(upx, upy, upz))

- umístí kameru v prostoru
- orientuje ji určitým směrem
- natočí okolo osy objektivu
- ve skutečnosti zapouzdřená série translací a rotací



Projekční transformace

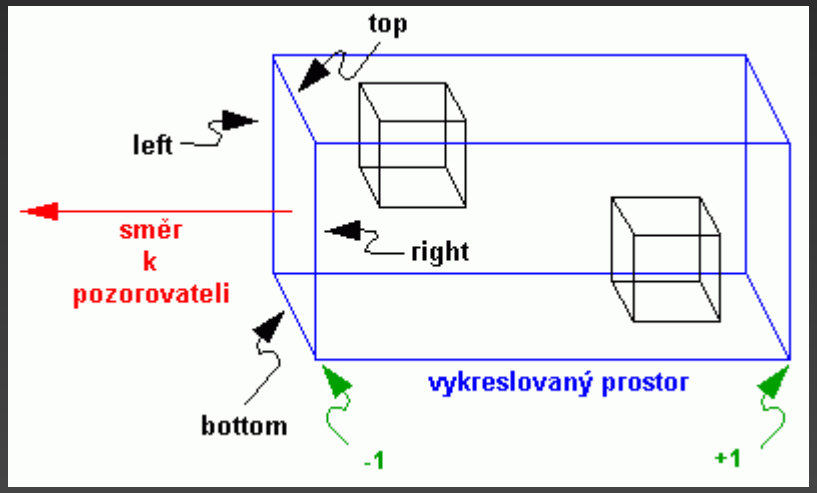
Ortografická projekce

- Zachovává vzdálenosti

  glm::mat4 = glm::ortho(left, right, bottom, top, near, far)

- Speciální případ: ve 2D není potřeba zadávat near a far

  glm::mat4 = glm::ortho(left, right, bottom, top)

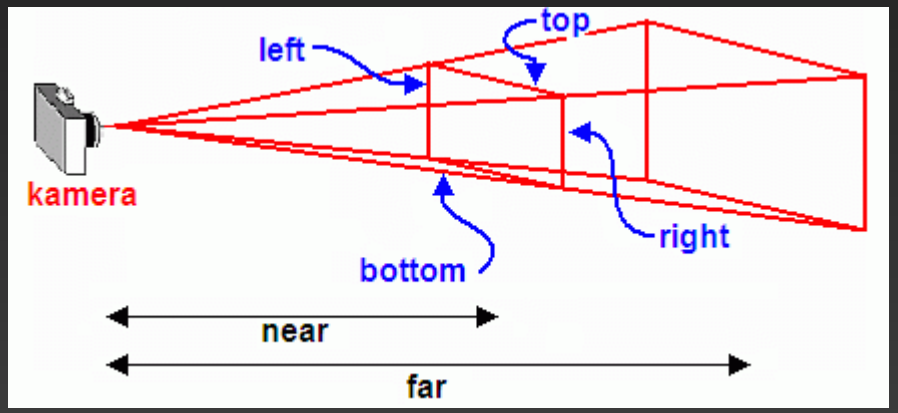


$$M_{proj} = \begin{pmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{-2}{zFar-zNear} & -\frac{zFar+zNear}{zFar-zNear} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspektivní projekce

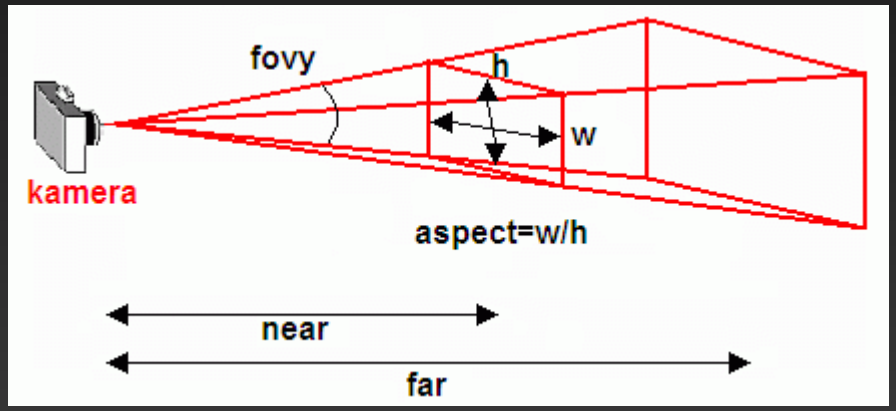
glm::mat4 glm::frustum(left, right, bottom, top, znear, zfar)

- určuje ořezové roviny a deformaci
- zmenšuje vzdálenější objekty

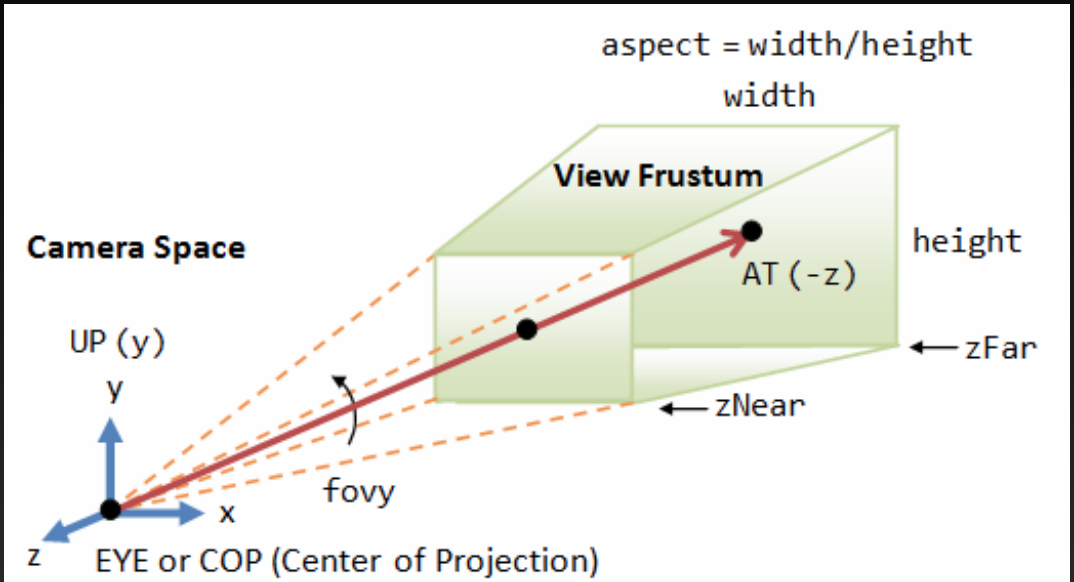


Perspektiva jednodušeji

glm::mat4 glm::perspective(fovy, aspect, near, far)



Projekční matice perspektivy

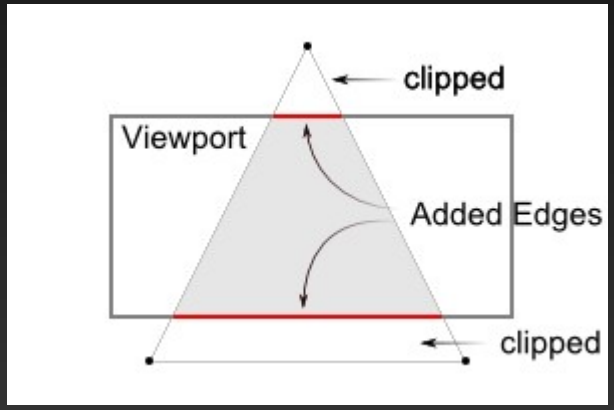


Perspective Projection: The camera's view frustum is specified via 4 view parameters: fovy, aspect, zNear and zFar.

$$M_{proj} = \begin{pmatrix} \frac{\cot\left(\frac{fovy}{2}\right)}{aspect} & 0 & 0 & 0 \\ 0 & \cot\left(\frac{fovy}{2}\right) & 0 & 0 \\ 0 & 0 & -\frac{zFar}{zFar - zNear} & -\frac{zNear \times zFar}{zFar - zNear} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Ořezové roviny

- Projekční transformace nastavují 6 ořez. rovin

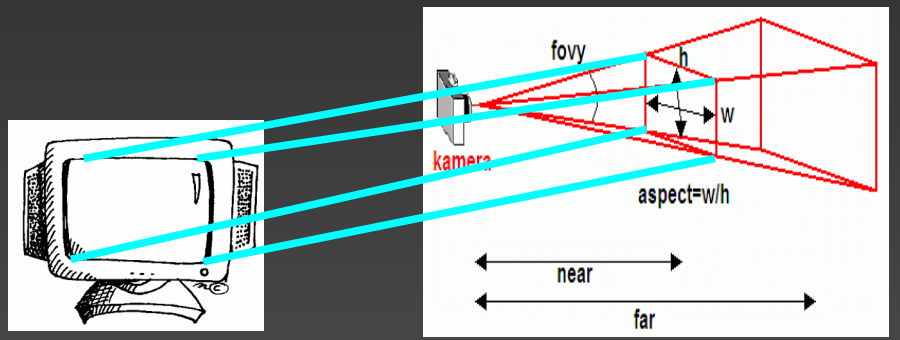


Zobrazovací transformace

- Mapování z relativních OpenGL souřadnic na obdélníkovou oblast pixelů na obrazovce

  glViewPort(x, y, width, height)
  - [x, y] levý dolní roh
  - Složená z 3 transformací: otočení osy y, škálování XYZ, posunutí počátku
- Poměr stran by měl odpovídat projekci
  - Jinak deformace obrazu

$$M_{viewport} = \begin{pmatrix} \frac{w}{2} & 0 & 0 & minX + \frac{w}{2} \\ 0 & \frac{-h}{2} & 0 & minY + \frac{h}{2} \\ 0 & 0 & maxZ - minZ & minZ \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



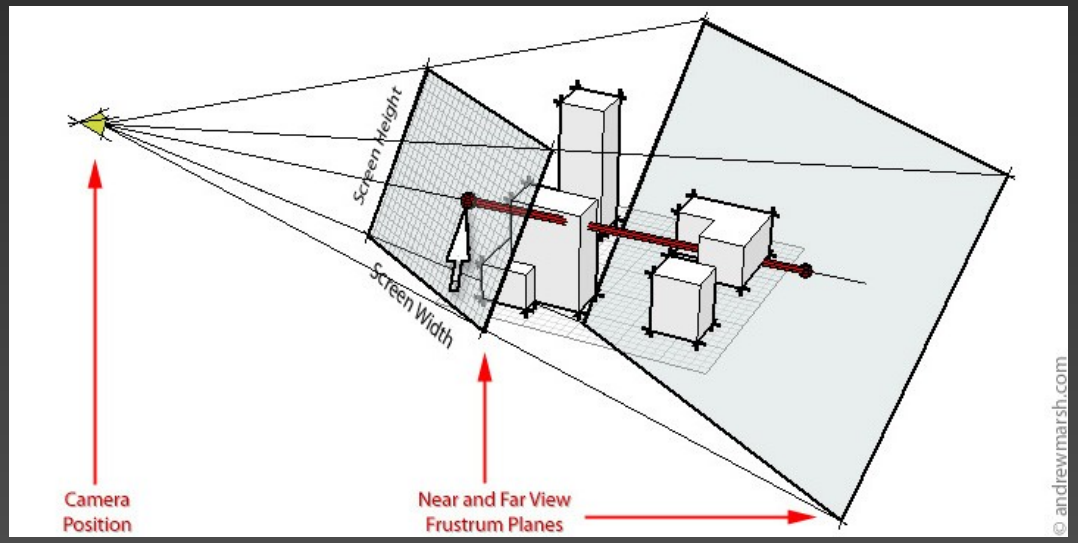
Další možnosti manipulace

- Především zapamatování pozice a obnovení, např.:
  - zobrazení 3D scény s perspektivou
  - uložit matici(e)
    - informace o scéně ortograficky (FPS apod.)
  - obnovit matici(e)


```
std::stack<glm::mat4> stack_mv;
glm::mat4 m_mv = glm::identity<glm::mat4>(); //null transformation

//... some draw code ...

stack_mv.push(m); //save current ModelView

// apply additional transformations
m = glm::translate(m, glm::vec3(x, y, z));
m = glm::scale(m, glm::vec3(scaleValue));

// send matrix to shaders
glUniformMatrix4fv(glGetUniformLocation(basic_shaders.ID, "uMV_m"), 1, GL_FALSE, glm::value_ptr(m_mv));

draw_object();

mv = stack_mv.top(); // restore ModelView
stack_mv.pop();      // remove from stack

// send restored matrix to shaders
glUniformMatrix4fv(glGetUniformLocation(basic_shaders.ID, "uMV_m"), 1, GL_FALSE, glm::value_ptr(m_mv));

//continue draw...
```

Zpětná vazba

- Při kliknutí do okna zjistit, kam se kliklo ve 3D
  - kliknutí je událost v okně → [x,y] okna OS
  - transformované GL souřadnice „odtransformuje“
  - glm::vec3 $objekt_{xyz}$ = glm::unProject($window_{xyz}$, modelview matice, projekční matice, viewport)
    - inverzní transformace, vyžaduje všechny údaje, neošetřuje některé krajní stavy...



Nastavení kamery

- Méně intuitivní

  glm::rotate(), glm::translate(), glm::scale(), …
  - špatné nastavení obvykle vykreslí černou plochu (barva pozadí), kamera hledí mimo scénu

- Pochopitelnější

  glm::lookAt()
  glm::ortho()
  glm::perspective()

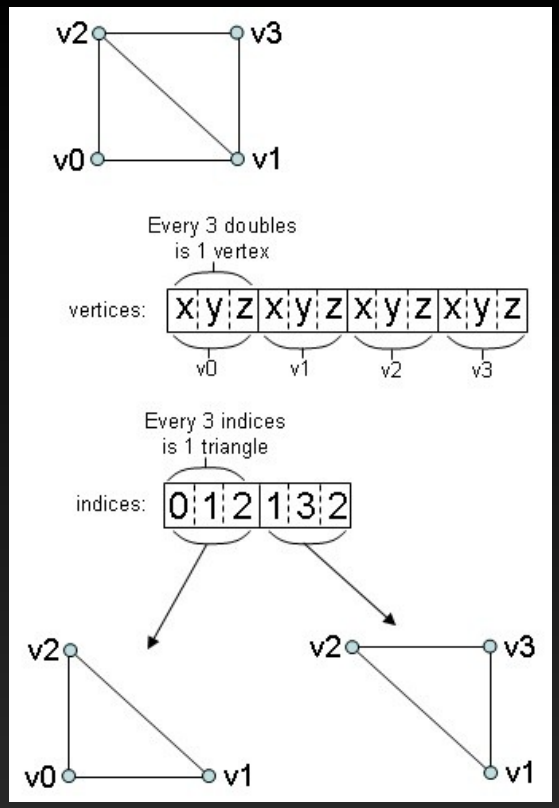
- https://www.opengl.org/wiki/Object_Mouse_Trackball

Některé funkce GLM

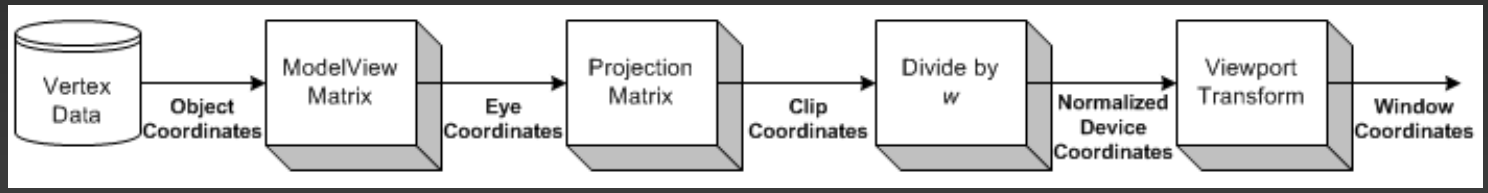
- rotace
- změna měřítka
- posun
- prázdná transformace
- násobení (skládání transformací)
- transpozice
- násobení s transpozicí
- perspektivní projekce (6 rovin)
- rovinná projekce (6 rovin)
- nastavení kamery
- rovinná projekce
- perspektivní projekce
- výpočet proj.m. pro kreslení do části obrazovky
- obj[xyz] → viewport[xy]
- viewport[xy] → obj[xyz]

- glm::rotate
- glm::scale
- glm::translate
- glm::identity<mat4>()
- * (C++ operator overload)
- glm::transpose(m)
- * (C++ operator overload)
- glm::frustum
- glm::ortho
- glm::lookAt
- glm::ortho
- glm::perspective
- glm::pickMatrix
- glm::project
- glm::unProject

Opakování



- Reprezentace
  - 3D rastr
  - obálka
    - vertex
    - hrana
    - ploška
- 3D zobrazování
  - načtení a transformace souřadnic zadaných vertexů



  - rasterizace
  - výpočet barvy fragmentu
  - průhlednost a zakrývání podle Z

Rasterisation

- DDA, Bresenham, scanline algorithm, edge walk, edge eq., …
  - coord. interpolation