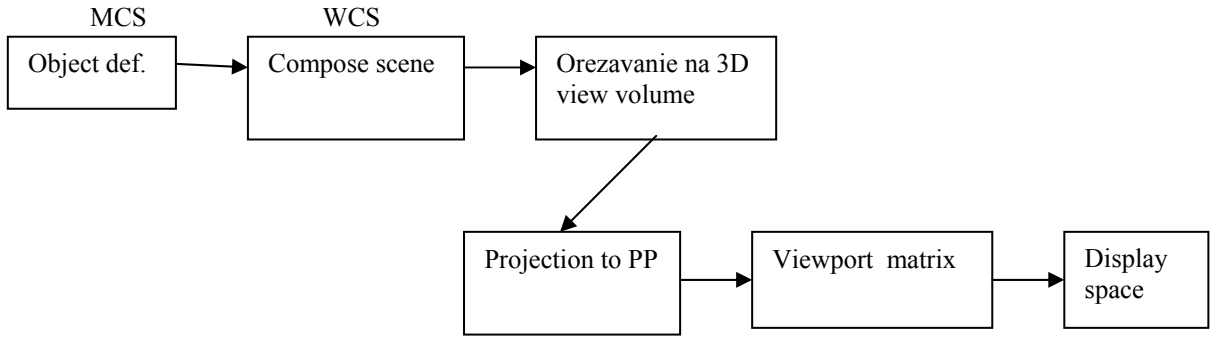
prednáška **Počítačová grafika 1**

Súradnicové systémy

1. Obsah

- Graphics pipeline
- Súradnicové systémy
- Pozícia kamery
  - LookUp vector
- Zobrazenie na kanonicky objem
  - pre rovnobežné premietanie
  - pre stredové premietanie
- View volumes (in OpenGl a DX)
- Orezávanie vo viewing volumes
- Stereo viewes

2. 3D viewing pipeline



There could be also:

- Hidden surface removal
- Transformation into viewpoint in 2D coordinates
- …

3. Súradnicové systémy

- Modelovací súradnicový systém -Modeling (Local) Coordinate System (**MCS**)
  - For easy modeling of objects
  - Storing of object data in "user friendly" coordinates
  - Every object has your own coordinates

- Svetový súradnicový system - Word Coordinate System (**WCS**)
  - Common space
  - Transformation from MCS to WCS

- Súranicový system kamery - Camera, eye or view Coordinate System (**CCS**)
  - We need to know:
  2. View point (viewer position in WCS, position of camera) **VP**
  3. View coordinate system (**UV**) with respect to the VP
  4. View plane (Projected plane) **PP**
  5. View frustum (to define the field (depth) of view) **VF**

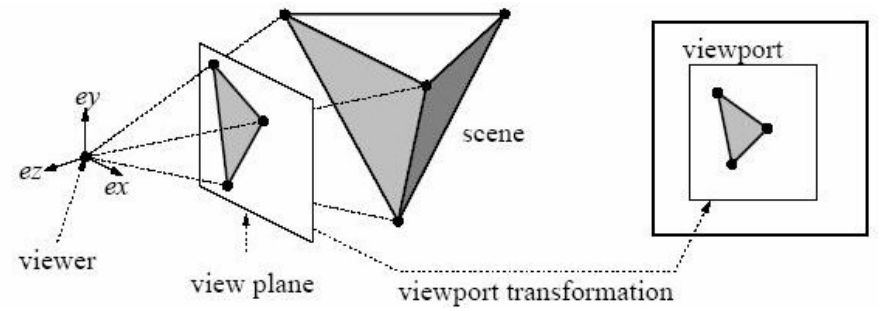
6. Perspective or parallel projection?

$$\begin{pmatrix} x_v \\ y_v \\ z_v \end{pmatrix} = T_{view} \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix}$$

where

$$T_{view} = RT$$

$$T = \begin{pmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} U_x & U_y & U_z & 0 \\ V_x & V_y & V_z & 0 \\ N_x & N_y & N_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



4. Konverzia do suradnicoveho systemu kamery

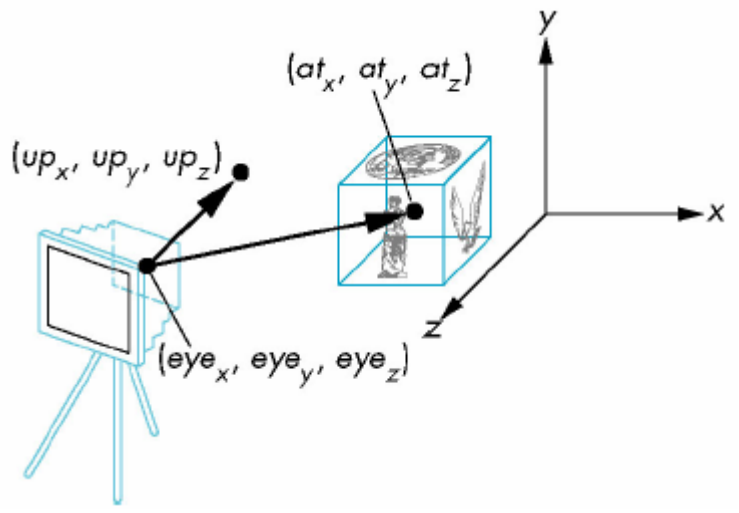
V opengl predstavuje danu funkcionalitu metoda gluLookAt()

Priklad:

```
// assuming: glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(eye_x, eye_y, eye_z, center_x, center_y, center_z, up_x, up_y, up_z);
```

4.1. Parametre kamery a implementacia

Pozicia kamery **E** a vektory ***u, v,*** a smer ***n***
Defaultne je nastavna kamera v smere –***n***.



n = eye – look
u = up x n
v = n x u

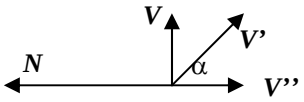
$$V = \begin{pmatrix} u_x & u_y & u_z & d_x \\ v_x & v_y & v_z & d_y \\ n_x & n_y & n_z & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

kde (dx,dy,dz) = (-eye.u, -eye.v, -eye.n)

Little problem:
Specifying a user interface for the system.
Dane: Bod Eye- **E**, Bod look-**L**, a vector Up-***U***
Potrebujeme nájsť poziciu kamery **C ,**a vektorov ***N, V, U***.
C = **E**
N = **E – L**

V (up) is problematic, because UP in WCS ≠ V(0, 0, 1)
- because ***V*** has to be perpendicular to ***N***

Solution: We allow V' (UP in WCS) but we'll transform it to ⊥ to ***N***
V = V' – (V'.N).N and normalize it



V = V' – V''
|V''| = |V'| cosα = |V'|.(W.V / |W|.|V|) = W.V
(-N.V') = | -N| . |V'| cosα => cosα = (-N.V')/|V'|
potom teda:
|V''| = |V'| cosα = (-N.V')
V'' = (-N) |V''|
V = V' – (-N.V').(-N)
V = V' – (N.V').N lebo |N|=1

Dalej:
U = NxV
(for left-hand coordinate system)

5. Culling or Back-Face Elimination

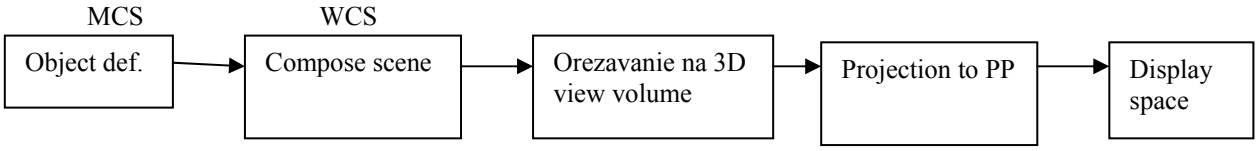
- removing back-face – 0br. 5.5 str. 148
- it's not general hidden surface

visibility := N_p*N > 0
N_p – polygon normal
N – line of sight

6. Zobrazenie na kanonicky objem:

6.1. 3D viewing pipeline 2

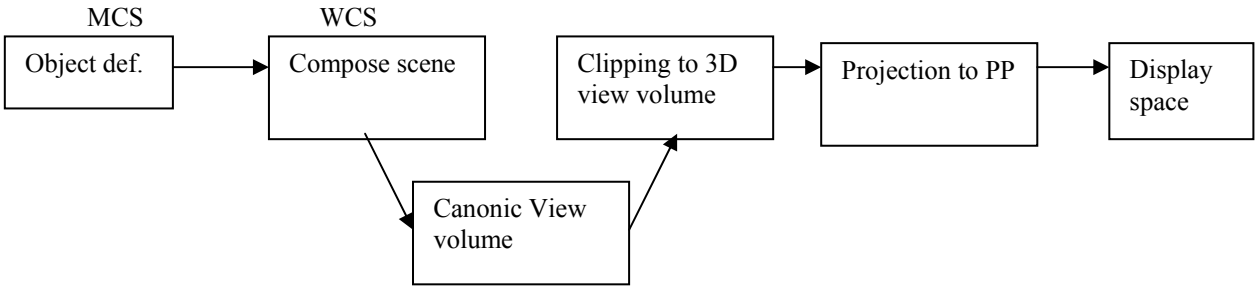
Previose:



There could be also:

- Hidden surface removal
- Transformation into viewpoint in 2D coordinates
- …

But in application is better:



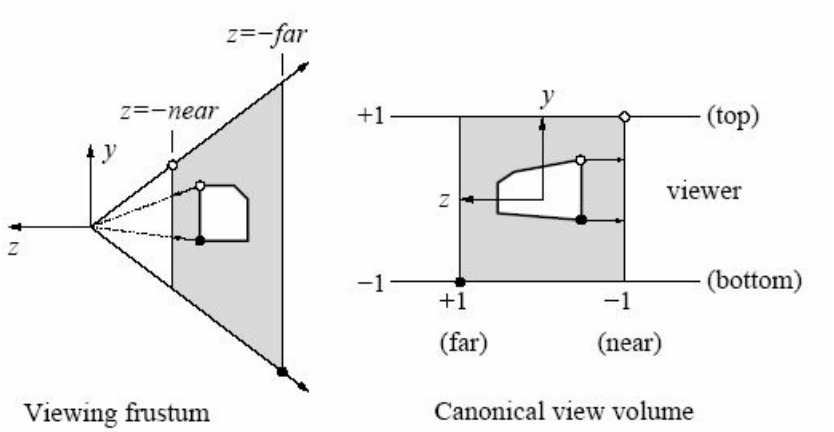
V praxi sa používa ako medzi krok zobrazenie na kanonicky objem.

6.2. Pre parallel objem

Priemetna je zadefinovana bodom **R** a vektormi ***u, v***
Zadefinujme 2D okno

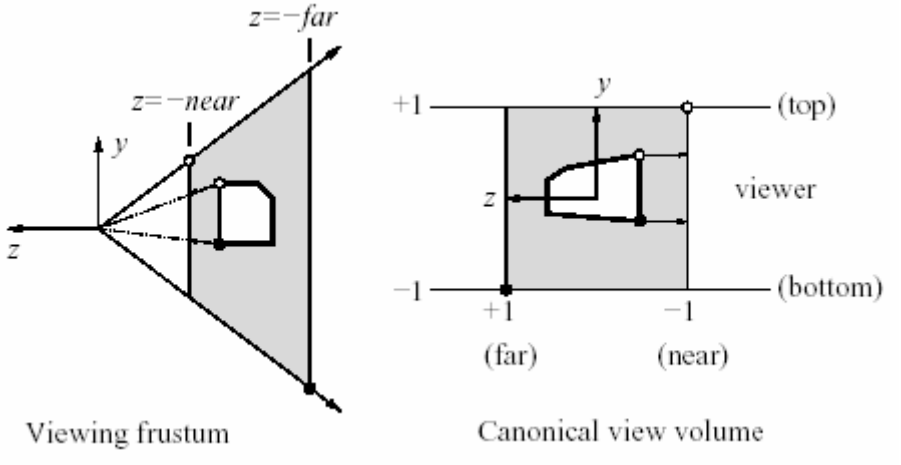
1.	posunutie R do pociatku	...	T
2.	otocenie okolo y, aby obraz lezal v xy	…	R_y
3.	otocenie okolo z, aby vysledny obraz n lezal na kladnej osi x	…	R_z
4.	otocenie okolo x, aby sa v stotoznil s osou y	…	R_x
5.	pripadna sumernost podla xy, aby lavotociva suradnicova sustava presla do pravotocivej	…	S_{xy}
6.	vyrovnanie sikmeho k-bokeho hranola do kvadra	…	D

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{d'_x}{d'_z} & -\frac{d'_y}{d'_z} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



6.3. Zobrazenie na kanonicky objem pre persp.

x = z, x = -z,
y = z, y = -z,
z= -z min, z = -1



Postup
1. posun o -(*right* + *left*)/2 v smere *x*
2. posun o -(*top* + *bott*)/2 v smere *y*
3. Skalovanie 2/(*right* - *left*) v smere *x*
4. Skalovanie 2/(*top* - *bott*) v smere y

Finalna matica - **projection matrix**:

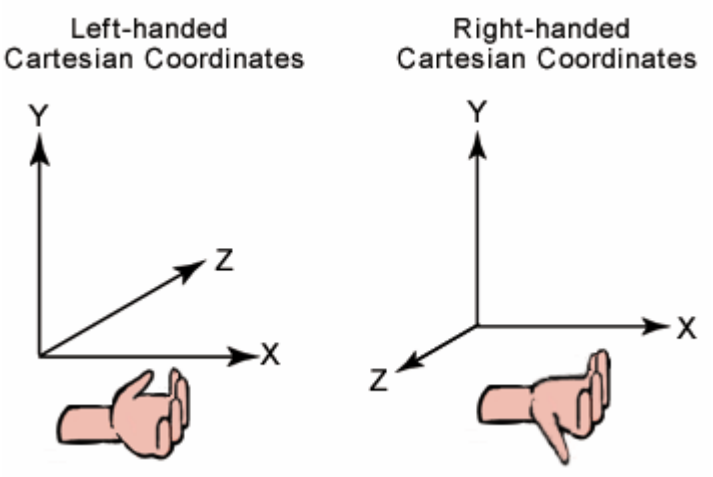
$$R = \begin{pmatrix} \frac{2N}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2N}{top-bottt} & \frac{top+bott}{top-bott} & 0 \\ 0 & 0 & \frac{-(F+N)}{F-N} & \frac{-2FN}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

6.4. CLIPING in 3D – generally Cohen – Sutherland

6 bit code

1. bit = 1 x < 0
2. bit = 1 x > a
3. bit = 1 x < 0
4. bit = 1 x > b
5. bit = 1 x < 0
6. bit = 1 x > c

7. Viewing in OpenGL



OpenGl as right-handed coordinate system: e_x x $e_y = e_z$,
and left handed otherwise (e_x x $e_y = -e_z$)

Notes: Microsoft® Direct3D® uses a left-handed coordinate system.
If you are porting an application that is based on a right-handed coordinate system, you must make two changes to the data passed to Direct3D.

- Flip the order of triangle vertices so that the system traverses them clockwise from the front. In other words, if the vertices are v0, v1, v2, pass them to Direct3D as v0, v2, v1.
- Use the view matrix to scale world space by –1 in the z-direction. To do this, flip the sign of the _31, _32, _33, and _34 member of the **D3DMATRIX** structure that you use for your view matrix.

Modelview transformation
GL_MODELVIEW

Perspective projection
GL_PROJECTION

Mapping to the viewpoint

7.1. Orthoghonal Projection

glOrtho(left, right, bottom, top, near far)

$$\begin{pmatrix} \frac{2}{right-left} & 0 & 0 & t_x \\ 0 & \frac{2}{top-bottom} & 0 & t_y \\ 0 & 0 & \frac{-2}{far-near} & t_z \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

where

$$t_x = -\frac{right+left}{right-left}$$

$$t_y = -\frac{top+bottom}{top-bottom}$$

$$t_z = -\frac{far+near}{far-near}$$

HW: Preco je to inac ako sme hovorili v casti kanonicke zobrazenie?

7.2. Perspective projection

Assuming that the eye is located at (0, 0, 0)

glFrustum(left, right, bottom, up, N, F)

$$\begin{pmatrix} \frac{2\ near}{right-left} & 0 & A & 0 \\ 0 & \frac{2\ near}{top-bottom} & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

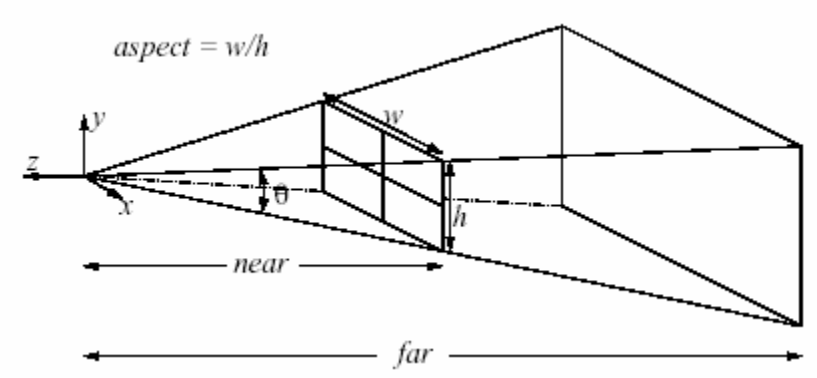
$$A = \frac{right+left}{right-left}$$

$$B = \frac{top+bottom}{top-bottom}$$

$$C = -\frac{far+near}{far-near}$$

$$D = -\frac{2\ far\ near}{far-near}$$

BUT better is (user friendly):
gluPerspective(viewAngle, aspectRadio, N, F)
- view angle – field of view (easy zoom)
- aspectRadio = pomer stran near obdlznika w/h
  Analogous to the size of film used in a camera
  Determines proportion of width to height of image displayed on screen



7.3. Converting to Viewer-Centered Coordinate System

gluLookAt(eye, center, ůp)

- CoP is orign
- View plane (or projection plane) is orthogonal to one of the axes (z-axis)

Eye – point
Center – point
UP - vector

7.4. Stereo Viewes