Einführung in die Erweiterte Realität

- 1. Introduction -

will be held on Oct 28, 2003

- 2. Coordinate Systems -

Gudrun Klinker

Oct 21, 2003

Agenda

1. Adminstrative Issues
2. Coordinate Systems

1. Administrative Issues

- Lecture:
  Tuesdays 9:15 - 10:45
- Schein:
  Oral exam
- Office hours:
  By appointment
- Infos:
  wwwbruegge.in.tum.de/projects/lehrstuhl/twiki/bin/view/Lehrstuhl/AugmentedRealityIntroductionWiSe2003

  or via:
  www.in.tum.de/~klinker -> Teaching -> Erweiterte Realität I

1. Administrative Issues
- Schedule -

- Oct. 21 Coordinate systems
- Oct. 28 Introduction
- Nov. 4 Computer graphics: OpenGL, VRML
- Nov. 11 Computer graphics: VRML
- Nov. 18 Mixed reality, information presentation
- Nov. 25 Displays (HMDs)
- Dec. 2 System architectures for AR: DWARF
- Dec. 9 System architectures for AR: Context Toolkit
- Dec. 16 Demos

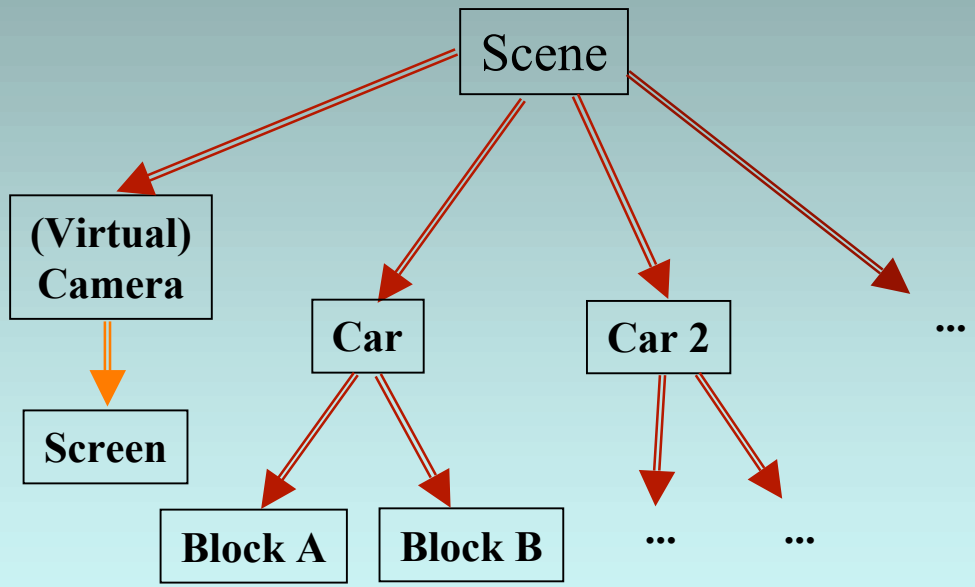
- Jan. 13 Trackers
- Jan. 20 Camera calibration
- Jan. 27 Computer vision
- Feb. 3 Intelligent environments
- Feb. 10 Discussion and conclusions

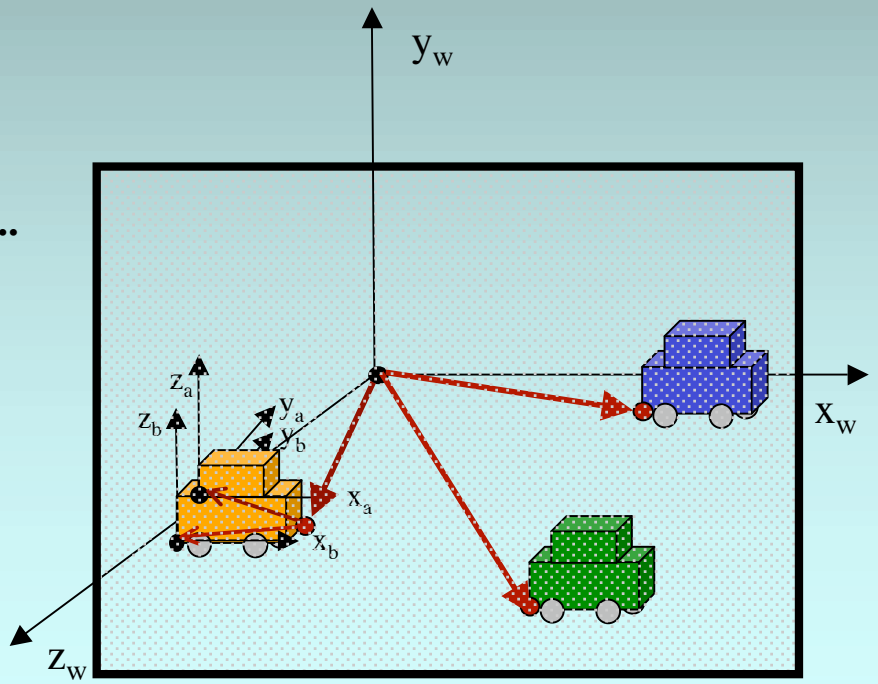
Einführung in die Erweiterte Realität
- 2. Coordinate Systems -

Gudrun Klinker

Oct 21, 2003

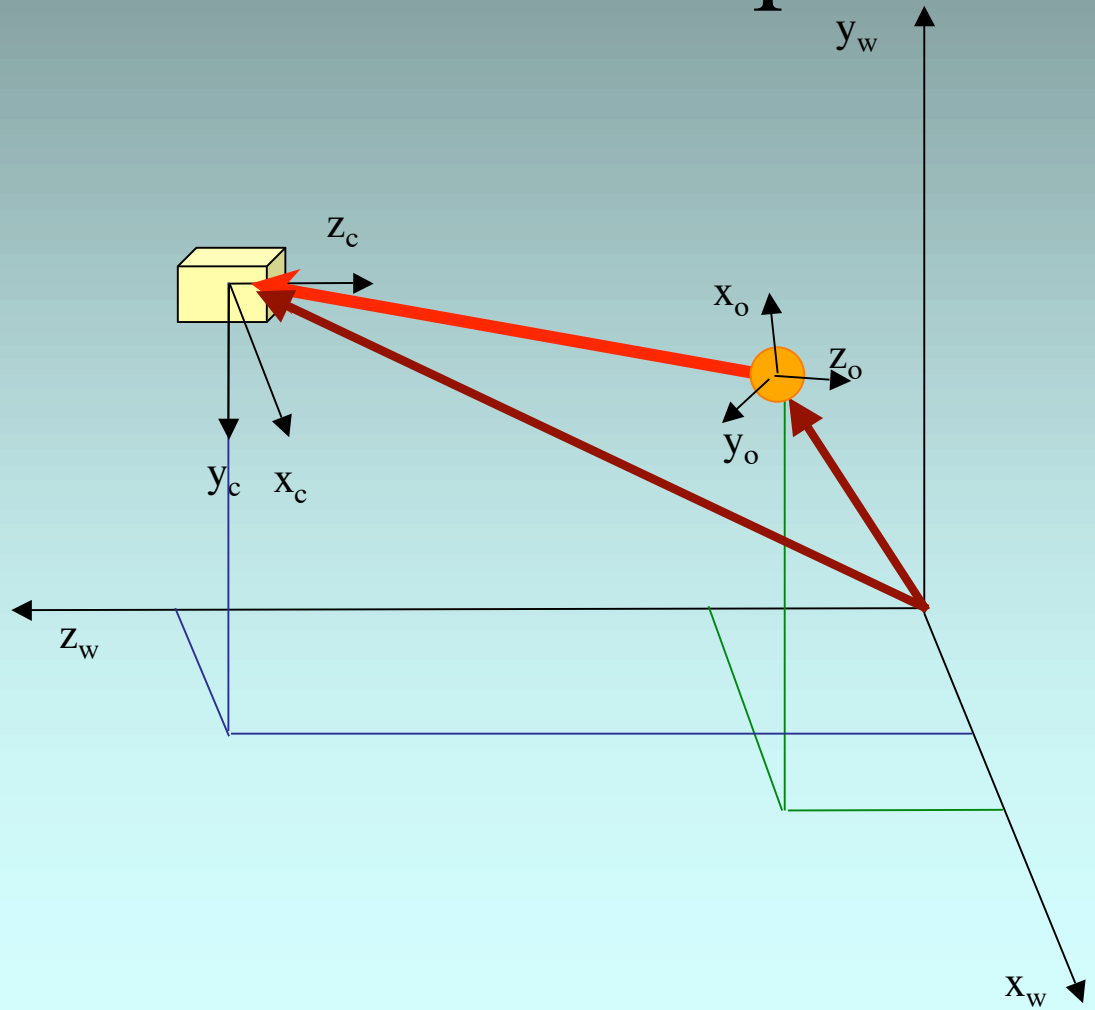
Scene Graph


Scene
(Virtual) Camera
Screen
Car
Car 2
Block A
Block B
...
...
...


y_w
x_w
z_w
z_a
z_b
y_a
y_b
x_a
x_b

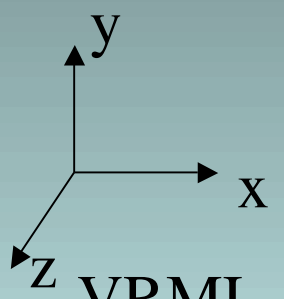
Rendering = Traversal of the Scene Graph

- **Homogeneous coordinates**
  3D Transformations between
  - Object coordinates
  - World coordinates
  - Camera coordinates
- **Geometry Pipeline in OpenGL**

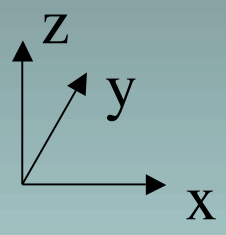


Different Coordinate Systems

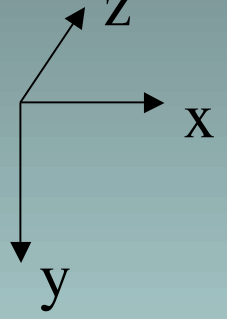
- Righthanded Systems



VRML, openGL

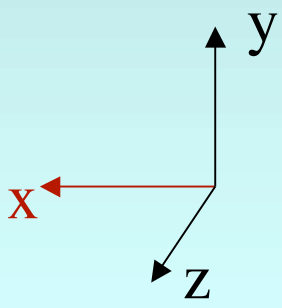


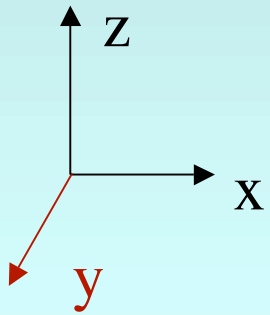
Apps

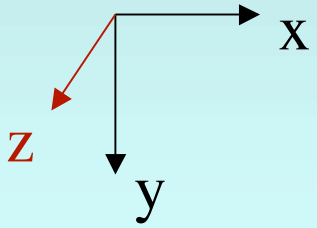


Video

- Lefthanded Systems



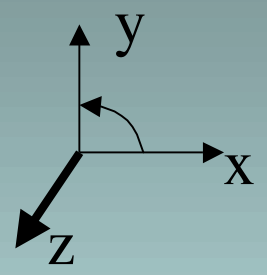




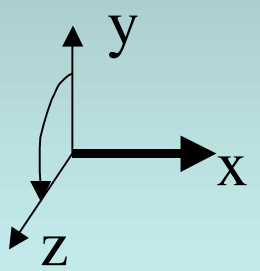
Video

Righthanded Rotations

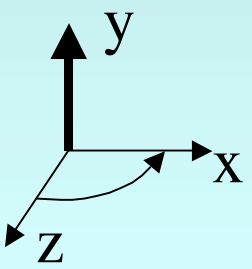
- Around z: x ⟶ y



- Around x: y ⟶ z



- Around y: z ⟶ x

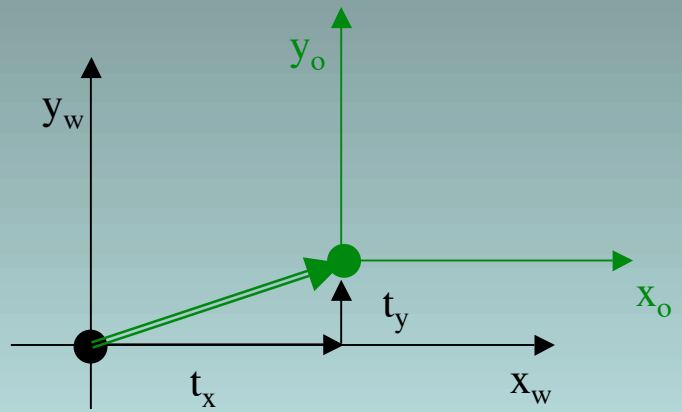


X-y-z-x-Y-z-x-y-Z-x-y-z

2D Transformations

- Translation:

$$x_w = x_o + t_x$$
$$y_w = y_o + t_y$$

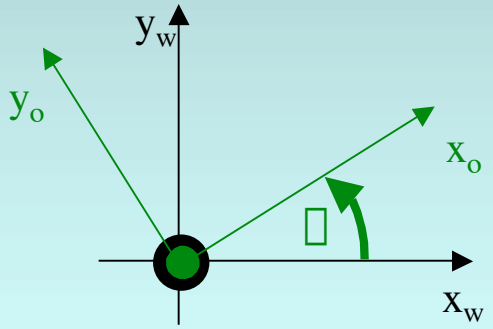


- Scaling:

$$x_w = s_x * x_o$$
$$y_w = s_y * y_o$$

- Rotation:

$$x_w = \cos\theta \cdot x_o - \sin\theta \cdot y_o$$
$$y_w = \sin\theta \cdot x_o + \cos\theta \cdot y_o$$



θ	0	30	45	60	90
sin	0.0	0.5	0.7071	0.8660	1.0
cos	1.0	0.8660	0.7071	0.5	0.0

2D Transformations Using Homogeneous Coordinates

- Translation:

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} x + wt_x \\ y + wt_y \\ w \end{bmatrix}$$

- Scaling:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ w \end{bmatrix}$$

- Rotation:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta \cdot x - \sin\theta \cdot y \\ \sin\theta \cdot x + \cos\theta \cdot y \\ w \end{bmatrix}$$

3D Transformations Using Homogeneous Coordinates

- Translation: glTranslate* (t_x, t_y, t_z)

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x + wt_x \\ y + wt_y \\ z + wt_z \\ w \end{bmatrix}$$

- Scaling: glScale* (s_x, s_y, s_z)

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_y z \\ w \end{bmatrix}$$

3D Transformations Using Homogeneous Coordinates

- Rotation: glRotate* (θ, e_x, e_y, e_z)
  - around x

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ \cos\theta\, y - \sin\theta\, z \\ \sin\theta\, y + \cos\theta\, z \\ w \end{bmatrix}$$

  - around y

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta\, x + \sin\theta\, z \\ y \\ -\sin\theta\, x + \cos\theta\, z \\ w \end{bmatrix}$$

  - around z

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta\, x - \sin\theta\, y \\ \sin\theta\, x + \cos\theta\, y \\ z \\ w \end{bmatrix}$$

Composition of Transformations

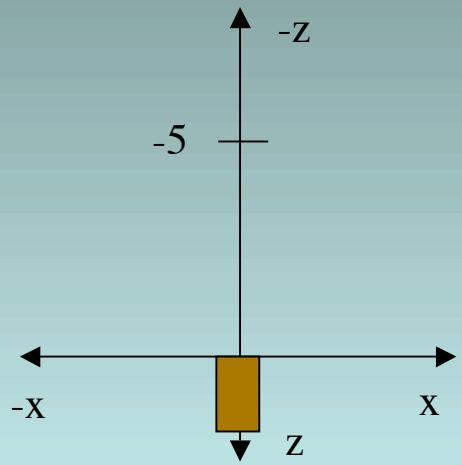
- Initialize accumulative matrix C_0 with identity matrix I
- Post-multiply accumulative matrix C_i with transformation matrix M_i according to command. $C_{i+1} := C_i M_i$
- Accumulated result represents complete transformation (independent of the number of transformation steps)
- Transformation sequence not commutative!!!

Example: translate1 - scale1 - rotate1 - translate2 - rotate2

$$I * T1 * S1 * R1 * T2 * R2 * \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

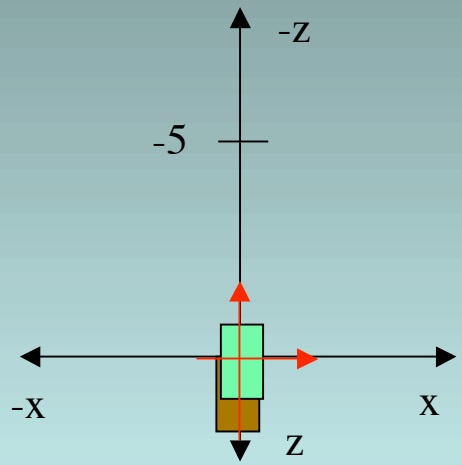
Viewing/Modelling Transformation Examples

glTranslatef (0.0, 0.0, -5.0); // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis



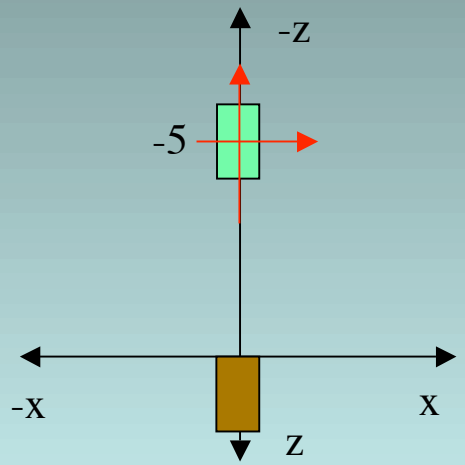
Viewing/Modelling Transformation Examples

```
glTranslatef (0.0, 0.0, -5.0);    // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis
```



Viewing/Modelling Transformation Examples

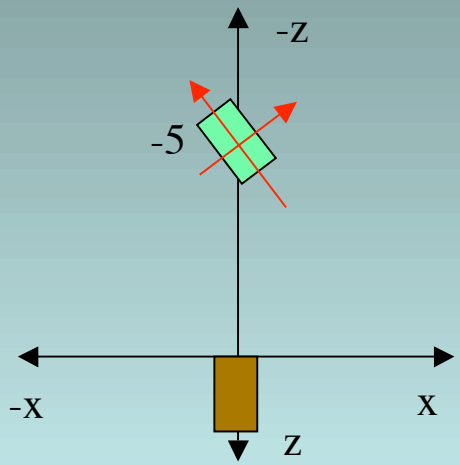
glTranslatef (0.0, 0.0, -5.0); // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis



Viewing/Modelling Transformation Examples

glTranslatef (0.0, 0.0, -5.0); // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis
...

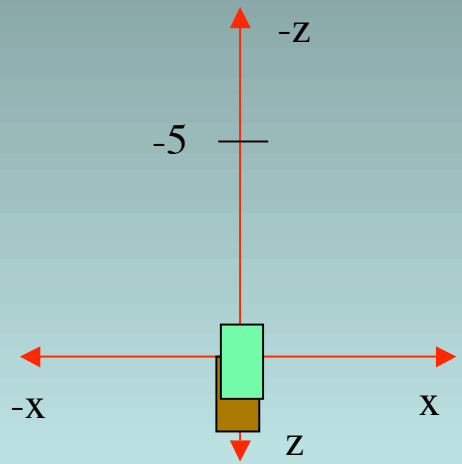
“Examine”



Viewing/Modelling Transformation Examples

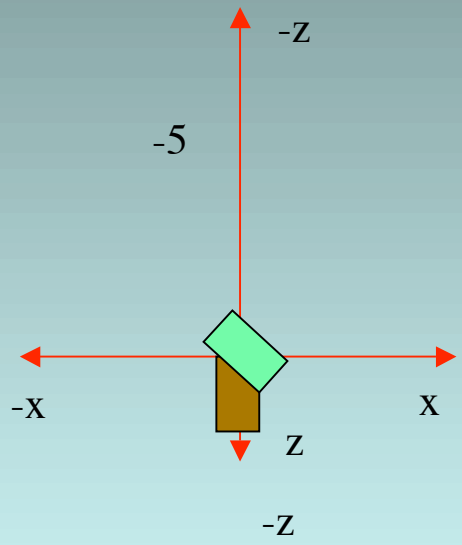
```
glTranslatef (0.0, 0.0, -5.0);    // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis
```

ALTERNATIVE:
“Fly-Through”



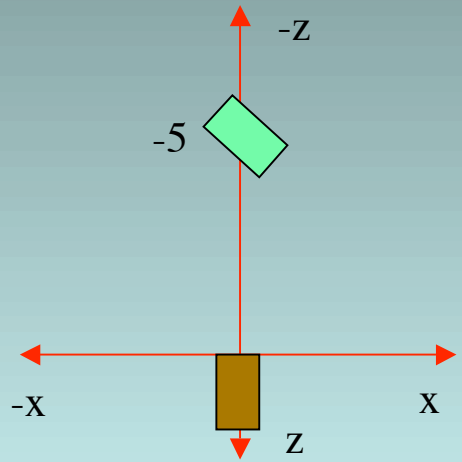
Viewing/Modelling Transformation Examples

glTranslatef (0.0, 0.0, -5.0); // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis



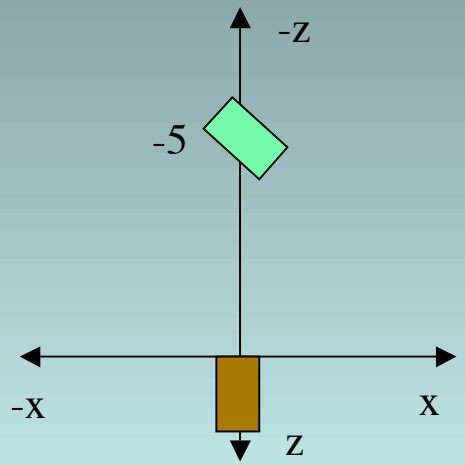
Viewing/Modelling Transformation Examples

glTranslatef (0.0, 0.0, -5.0); // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis

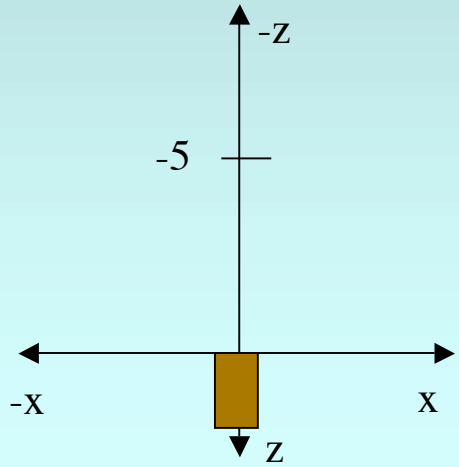


Viewing/Modelling Transformation Examples

glTranslatef (0.0, 0.0, -5.0); // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis

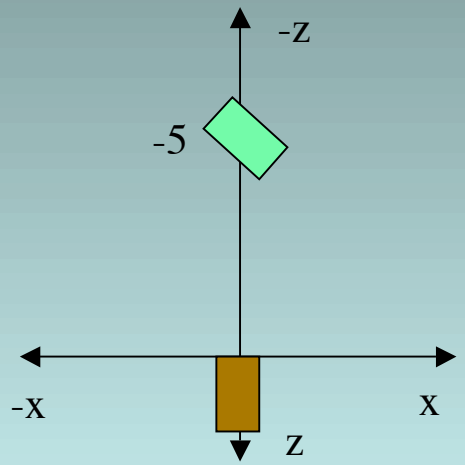


Question:
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis
glTranslatef (0.0, 0.0, -5.0); // along z-axis

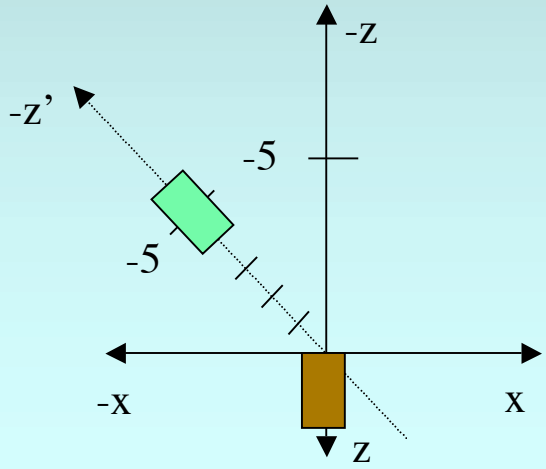


Viewing/Modelling Transformation Examples

```
glTranslatef (0.0, 0.0, -5.0);    // along z-axis
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis
```



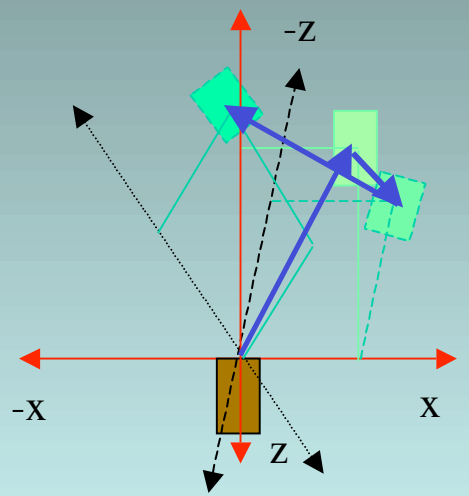

```
glRotatef (45.0, 0.0, 1.0, 0.0); // around y-axis
glTranslatef (0.0, 0.0, -5.0);    // along z-axis
```



Transformation Examples

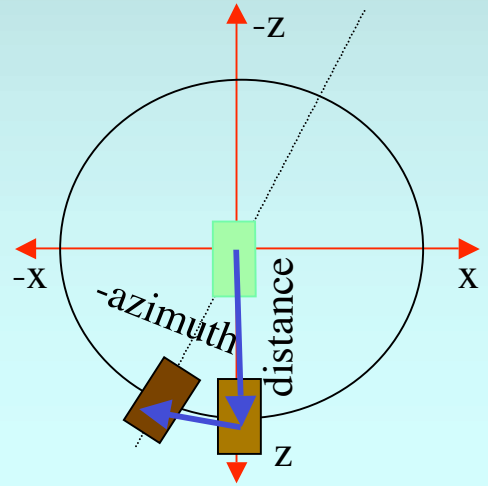
// Flight Simulator, Pilot View ("fly-through")
// world rotates around stationary camera

…
glRotated (roll, 0.0, 0.0, 1.0); // z (4)
glRotated (pitch, 0.0, 1.0, 0.0); // y (3)
glRotated (heading, 1.0, 0.0, 0.0); // x (2)
glTranslated (-planex, -planey, -planez); (1)



// Polar View: rotating camera (= examine object)
// camera rotates around stationary object

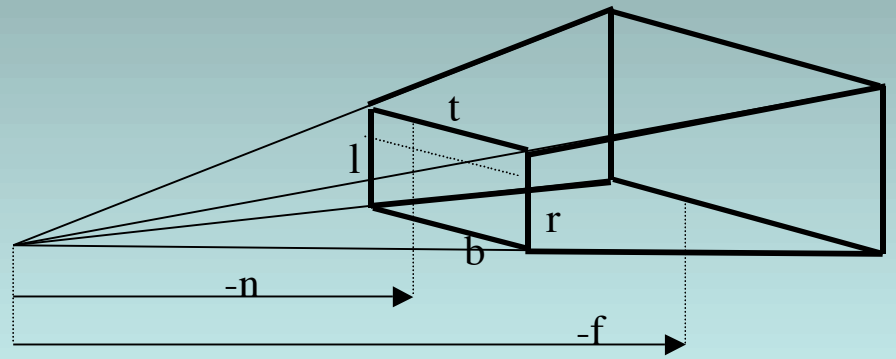
glTranslated (0.0, 0.0, -distance); (1)
glRotated (-twist, 0.0, 0.0, 1.0); // z (2)
glRotated (-elevation, 1.0, 0.0, 0.0); // x (3)
glRotated (-azimuth, 0.0, 1.0, 0.0); // y (4)
...



3D Projections

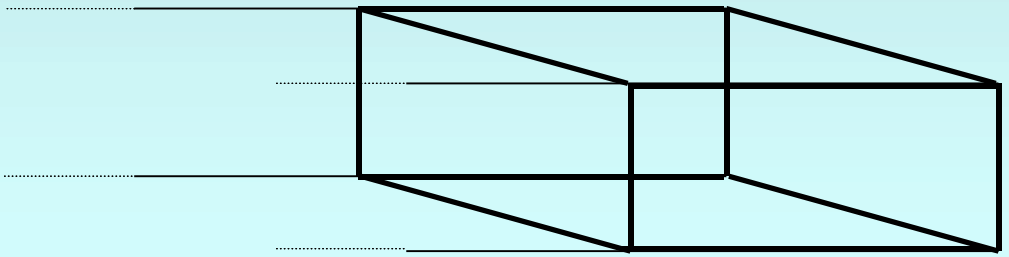
- Perspective Projection: glFrustum* (l,r,b,t,n,f)

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



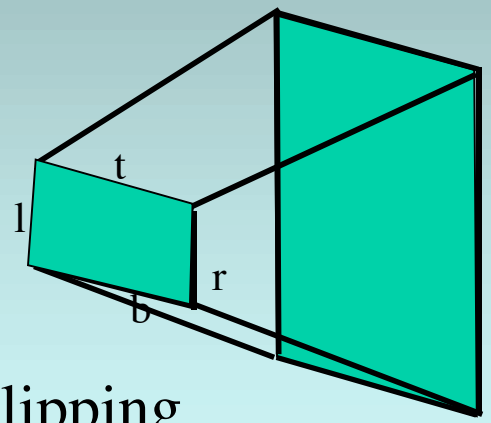
- Orthographic Projection: glOrtho* (l,r,b,t,n,f)

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



Viewport Transformation

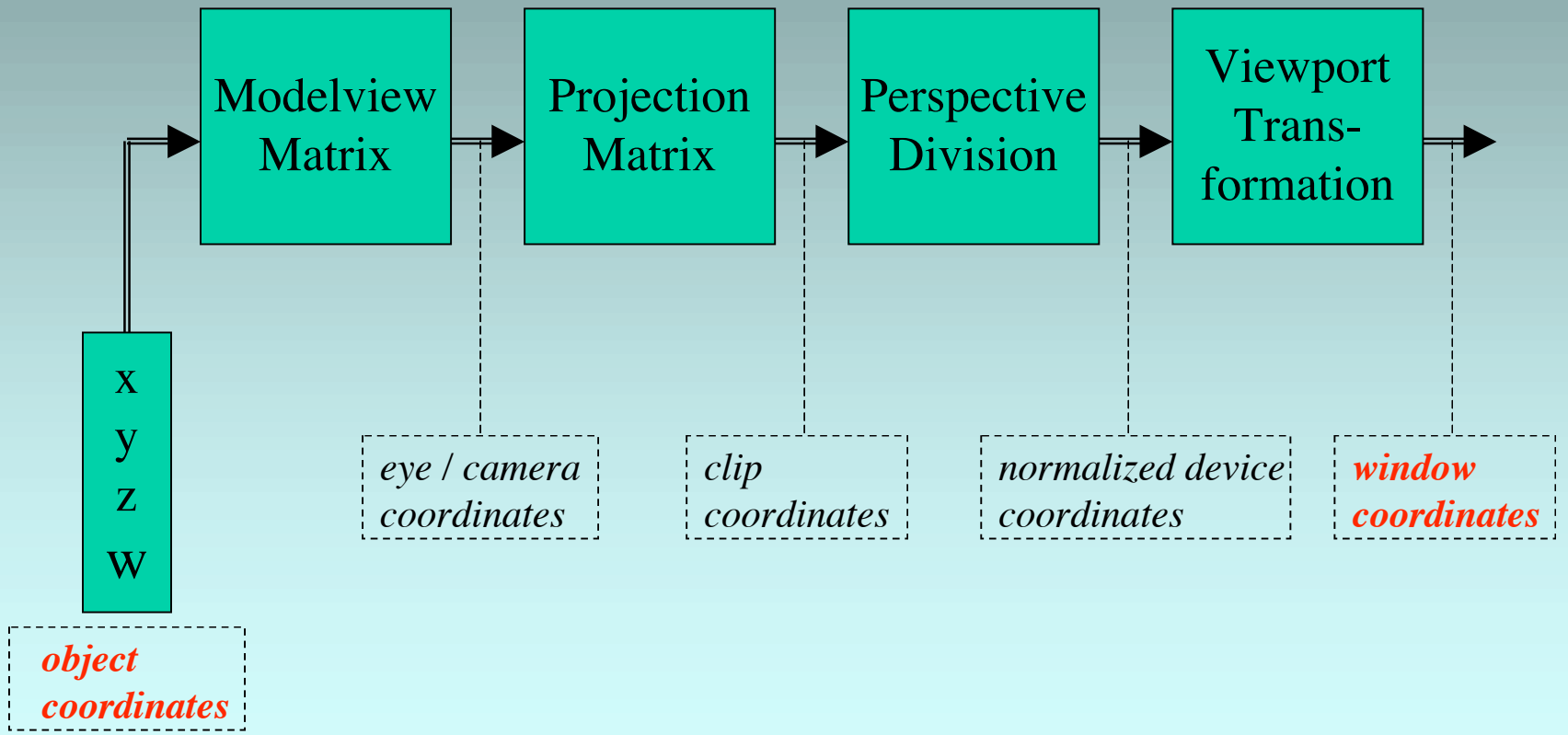
- glViewport (x, y, width, height)



window on
the screen

near clipping
plane of
frustum

Geometry Pipeline



Sample Code

```
// Viewport Transformations
glViewport (0.0, 0.0, w, h);

// Projection Transformations
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
glFrustum (l,r,t,b,n,f);   or gluPerspective (60.0, w/h, 1.0, 20.0);

// Viewing Transformations
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
glTranslatef (0.0, 0.0, -5.0);  // move world away from camera
glRotatef (45.0, 0.0, 1.0, 0.0);  // rotate world in front of the camera around y-axis
…
// Modeling Transformations
...
// Draw Object
```