Master of intelligent systems

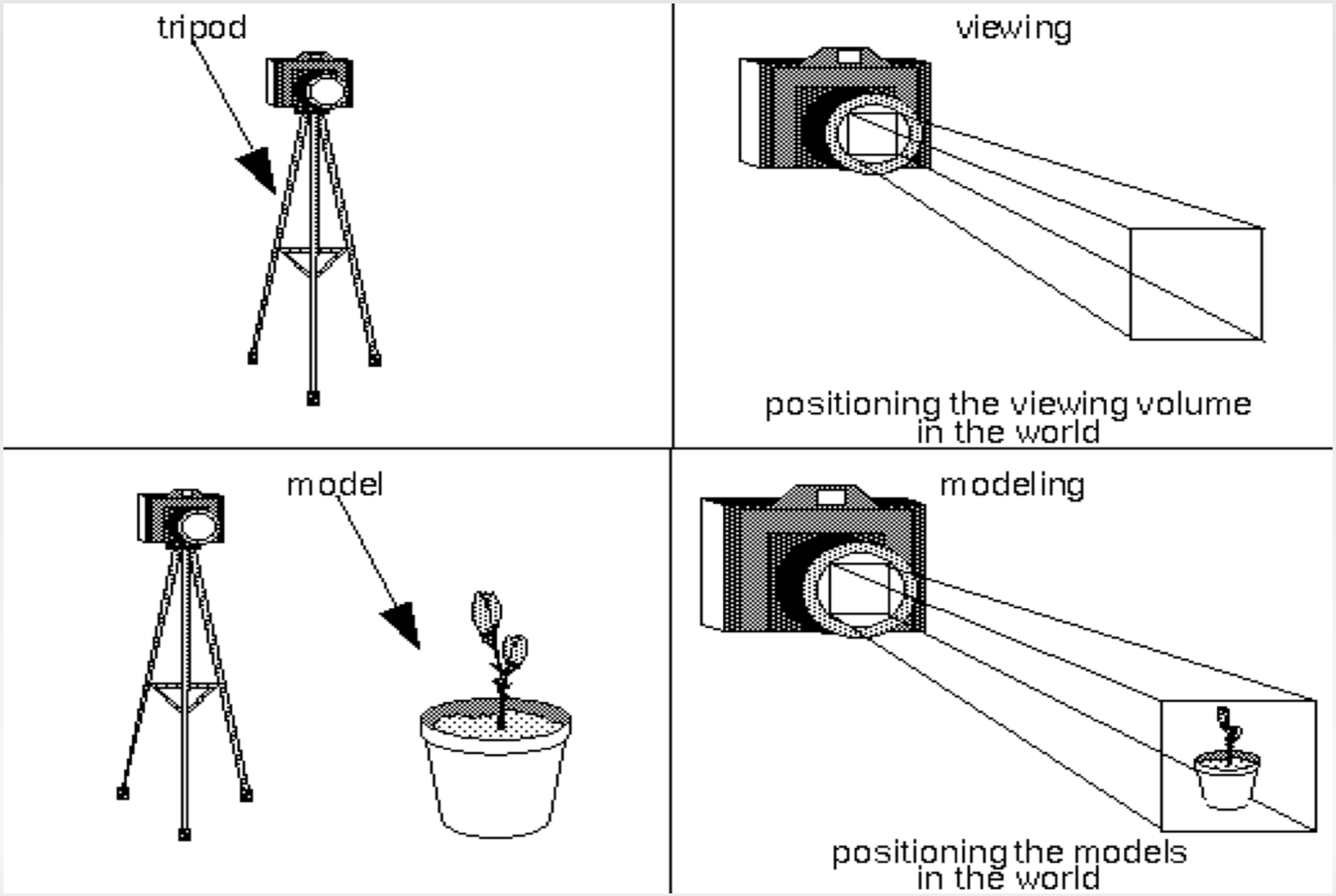
Image synthesis

Geometric transformations

Olivier Gourmel

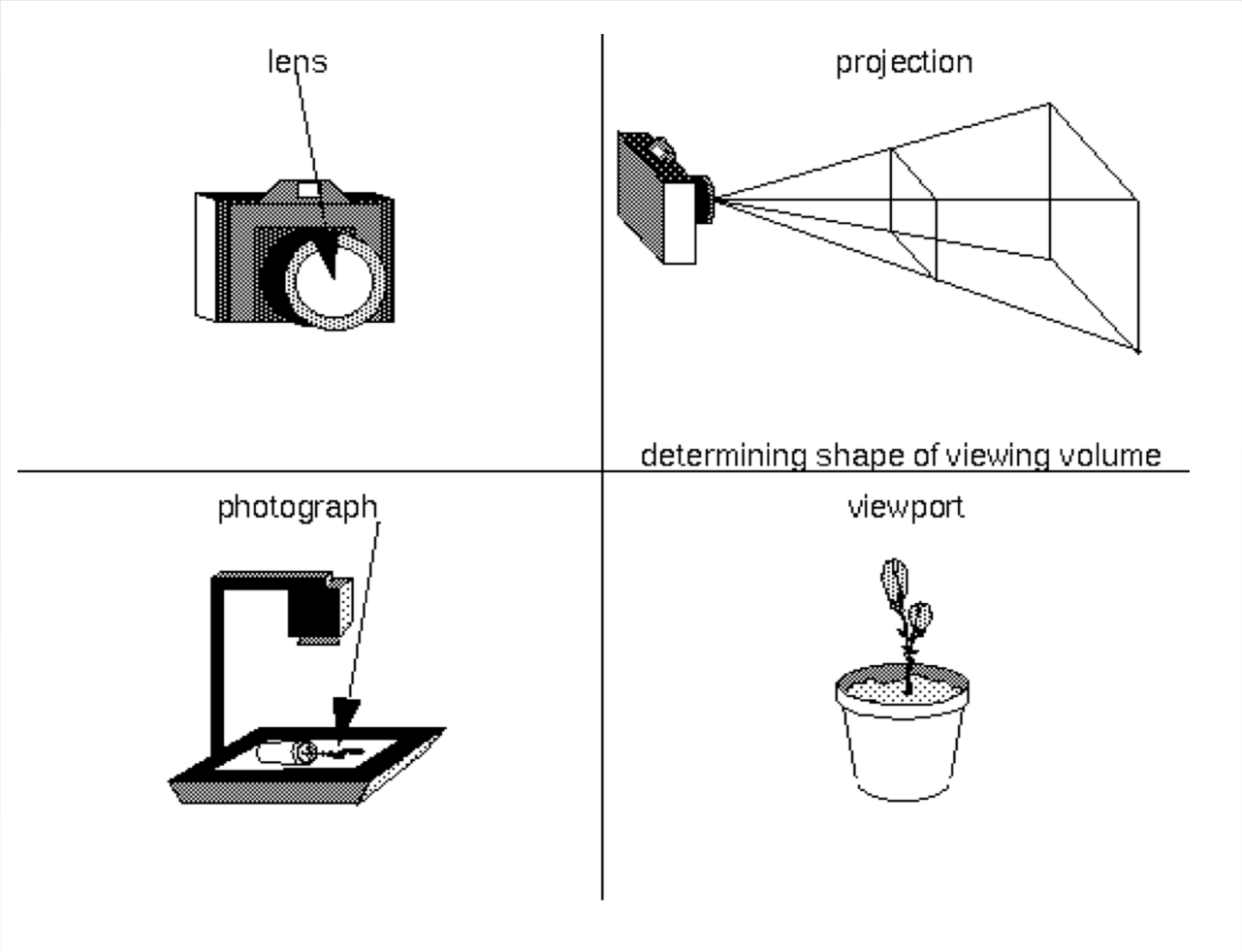
The transformation process

- Analogy with photography



The transformation process

- Analogy with photography



Geometric transformations

- Usual frames

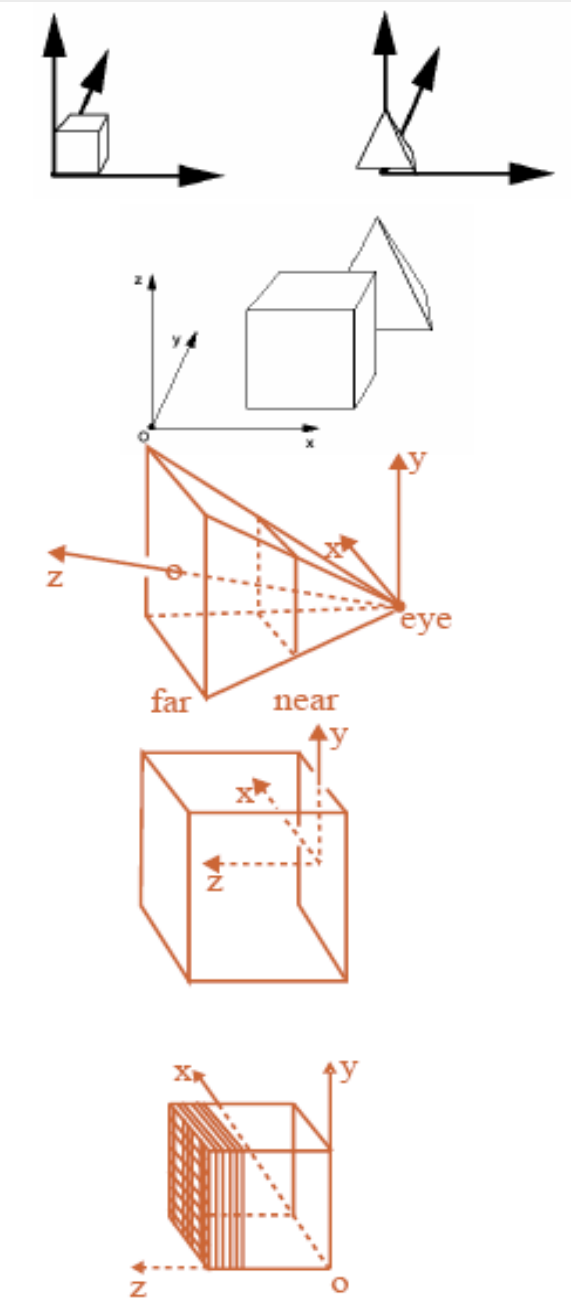
Object space

World space

Eye space

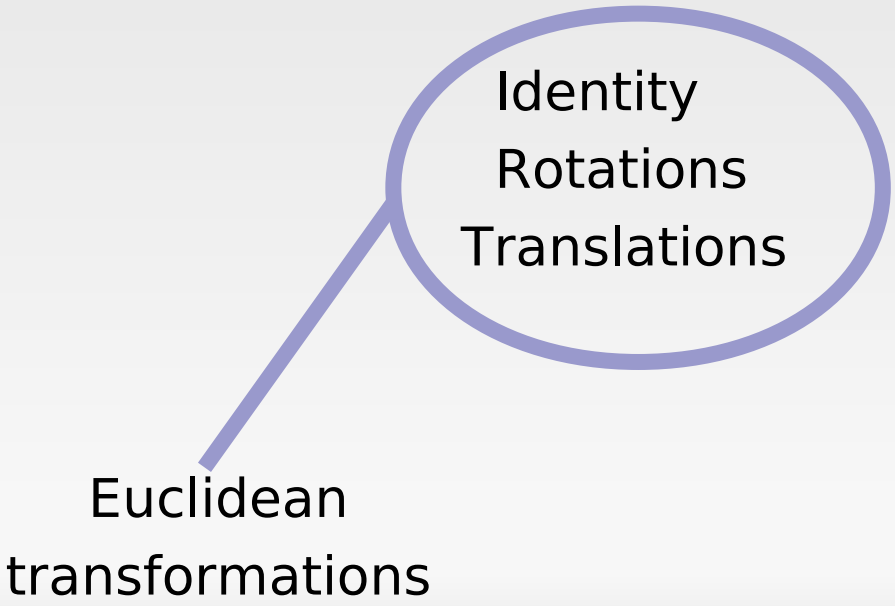
Normalized Device Coordinates

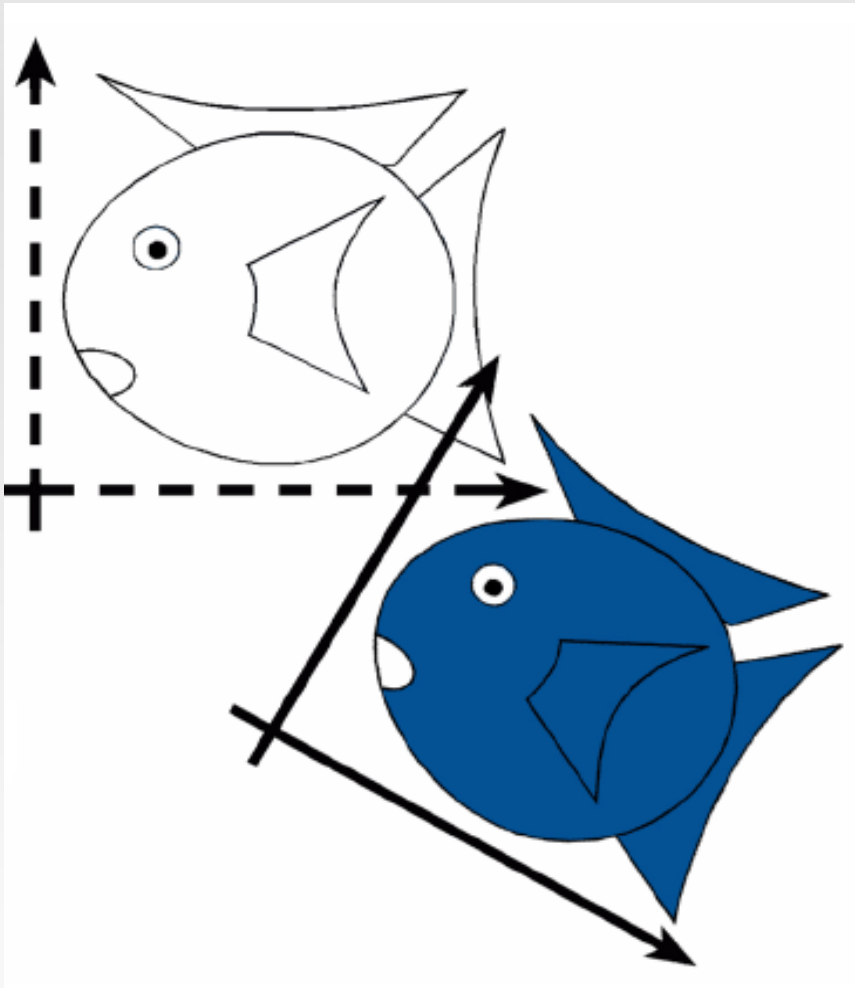
Screen Space



Geometric transformations

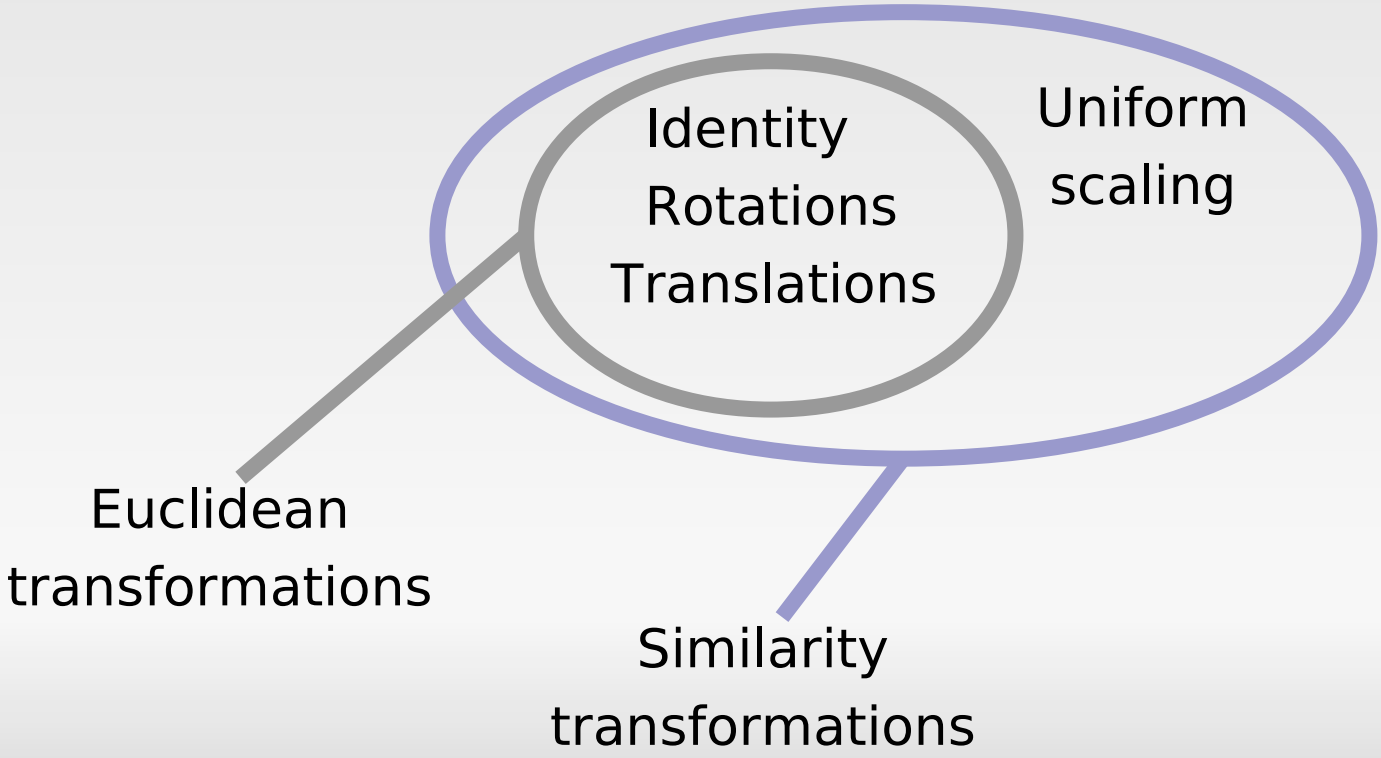
- Euclidean transformations (rigid)
  - Preserve angle measure
  - Preserve length measure
  - Invertible

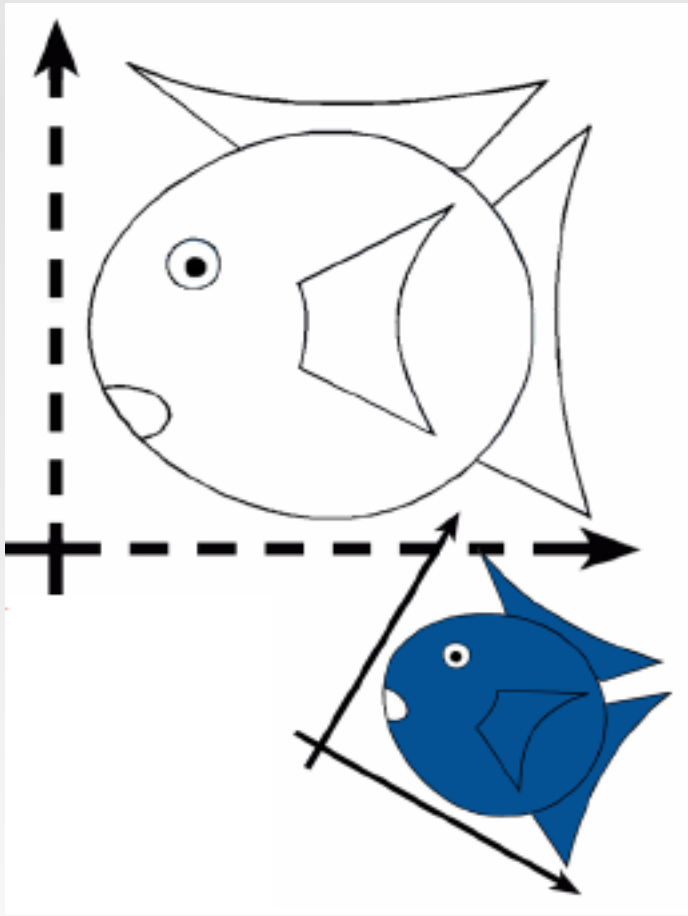




Geometric transformations

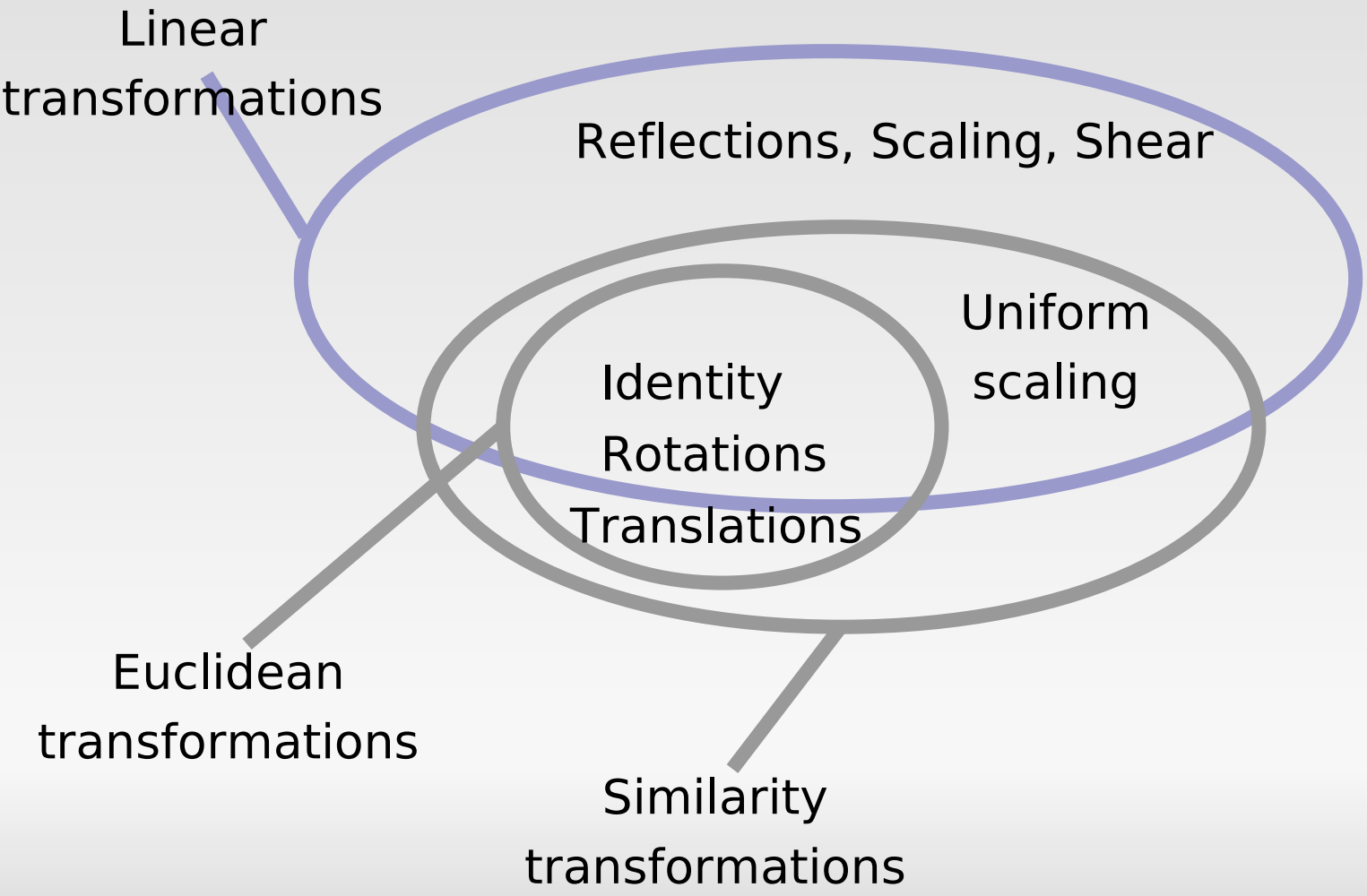
- Similarity transformations
  - Preserve angle measure
  - Invertible (except scale to zero)

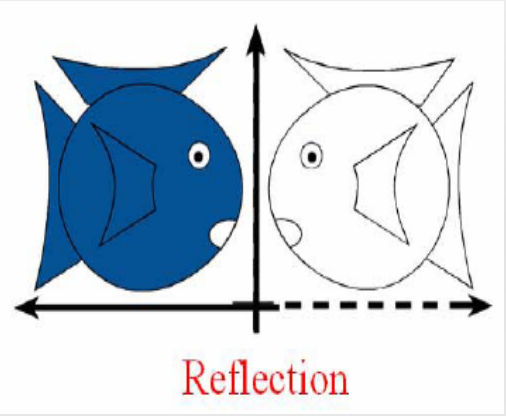


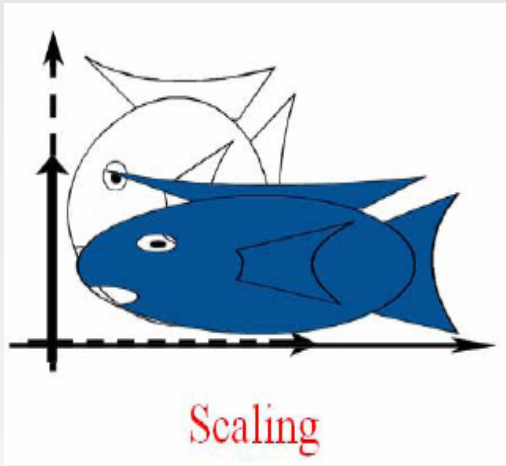


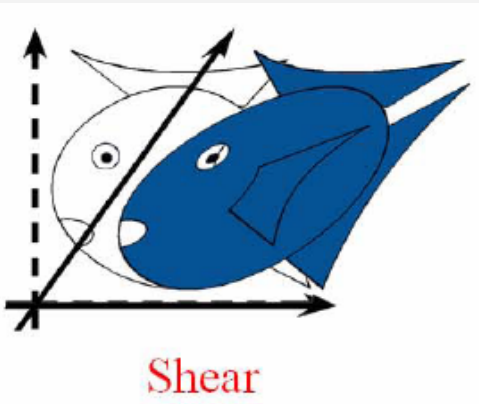
Geometric transformations

- Linear transformations
  - f(P+Q) = f(P) + f(Q) and f(a.P) = a.f(P)



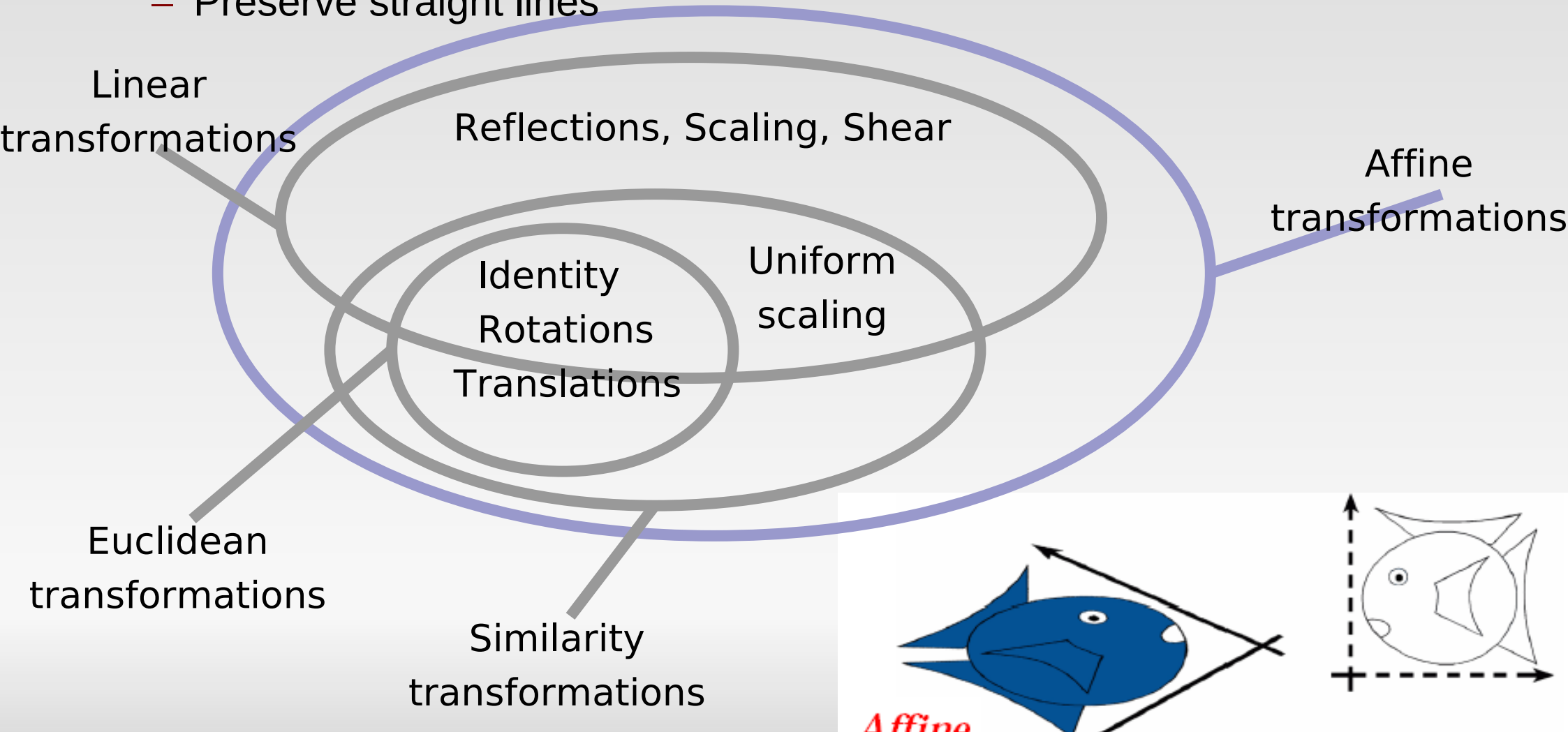






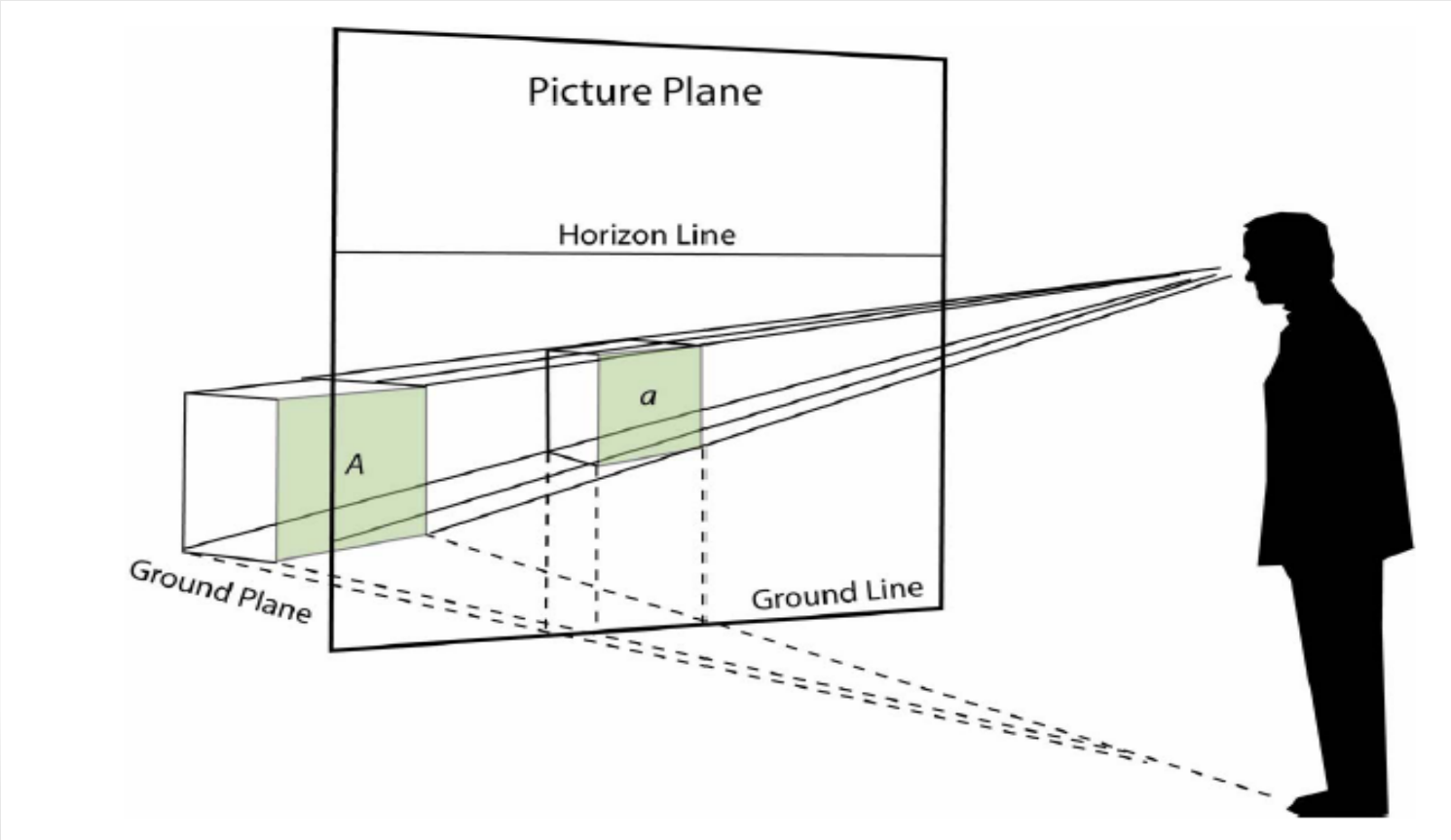
Geometric transformations

- Affine transformations
  - Preserve parallel lines / planes
  - Preserve straight lines



Geometric transformations

- Projections :
  - Affine transformations
  - **Non invertible**



Vector properties (reminder)

- Vector : position / direction in 3D space $V = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
- Operators $V + V' = \begin{bmatrix} x + x' \\ y + y' \\ z + z' \end{bmatrix}$ $V - V' = \begin{bmatrix} x - x' \\ y - y' \\ z - z' \end{bmatrix}$ $a.V = \begin{bmatrix} a.x \\ a.y \\ a.z \end{bmatrix}$
- Norm $\|V\| = x^2 + y^2 + z^2$
- Dot product $V.V' = x.x' + y.y' + z.z' = \|V\| \, . \, \|V'\| \, .cos(\theta)$
- Cross product $VxV' = \begin{bmatrix} y.z' - z.y' \\ z.x' - x.z' \\ x.y' - y.x' \end{bmatrix}$

2D geometric transformations

- Homothetic transformation (scaling):

$$\begin{Bmatrix} x' = x.s_x \\ y' = y.s_y \end{Bmatrix} \quad \text{with} \quad \begin{Bmatrix} s_x = 2 \\ s_y = 3 \end{Bmatrix}$$

- Matrix form:

$$p' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \qquad p = \begin{bmatrix} x \\ y \end{bmatrix}$$

2D geometric transformations

- Rotation

$$\begin{cases} x' = x.cos\theta - y.sin\theta \\ y' = x.sin\theta + y.cos\theta \end{cases}$$

- Matrix form:

$$p' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}$$

$$p' = M\ p$$

$$M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \qquad p = \begin{bmatrix} x \\ y \end{bmatrix}$$

2D geometric transformation

- Translation $\begin{Bmatrix} x' = x + t_x \\ y' = y + t_y \end{Bmatrix}$ with $\begin{Bmatrix} t_x = 4 \\ t_y = -4 \end{Bmatrix}$

- Matrix form ? Not a linear transformation...

$$T = [?]$$

We cannot write p' = T. p as before.

Geometric transformations

- What we want :
  - Unified notations
  - Composition of matrices to represent composition of transformations
- Solution : homogeneous coordinates (here in 2D)
  - (x, y, w) : Point P = (x/w, y/w) from $\Re^2$ if $w \neq 0$

    Vector V = (x, y) if w = 0
- Transformation Matrix : (in 2D 3x3, in 3D 4x4)

$$\begin{matrix} Current \\ Transformation \\ Matrix \end{matrix} = CTM = \begin{bmatrix} m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \\ m_4 & m_8 & m_{12} & m_{16} \end{bmatrix}$$

Geometric transformations

- Translation $\begin{Bmatrix} x' = x + t_x \\ y' = y + t_y \end{Bmatrix}$ with $\begin{Bmatrix} t_x = 4 \\ t_y = -4 \end{Bmatrix}$

- Matrix form in homogeneous coordinates:

$$p = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad p = \begin{bmatrix} x \\ y \end{bmatrix}$$

Geometric transformations

- Translation in homogeneous coordinates:

$$p = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$p' = \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} x + w.t_x \\ y + w.t_y \\ w \end{bmatrix} \rightarrow p' = \begin{bmatrix} \frac{x'}{w} \\ \frac{y'}{w} \\ \frac{w}{w} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} + t_x \\ \frac{y}{w} + t_y \\ 1 \end{bmatrix}$$

Geometric transformations (in 3D)

- Translation $T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Scaling $S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Rotations around an axis of the frame:

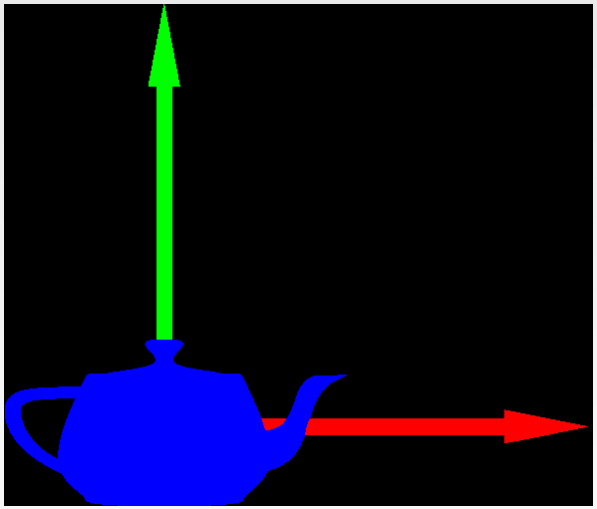
$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D geometric transformations

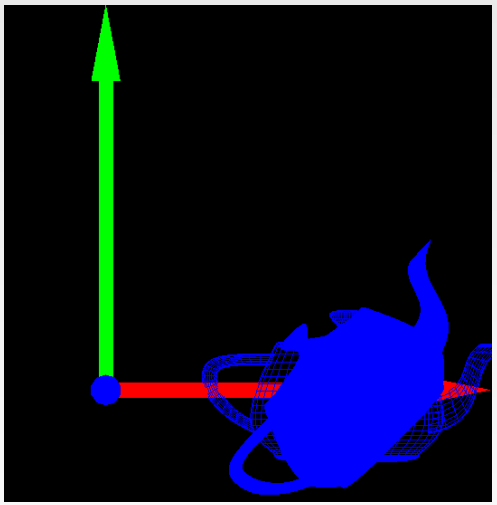
- successive transformations : matrix product
  - Associative :

$p_0 = (T0*(T1*T2))p = ((T0*T1)*T2)p = (T0*T1*T2)p$

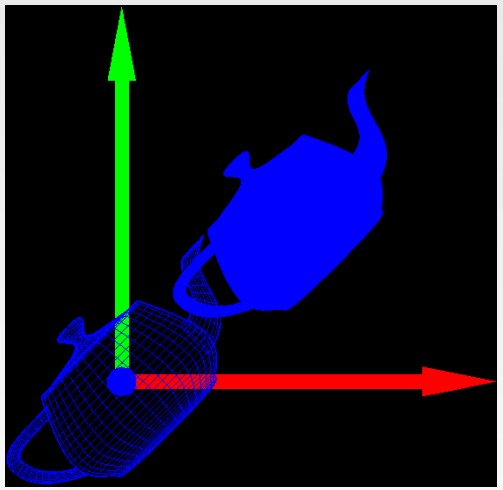
  - Caution !! Matrix product is not commutative (most of the time)



Initial position



Rotation followed by a translation along the x axis



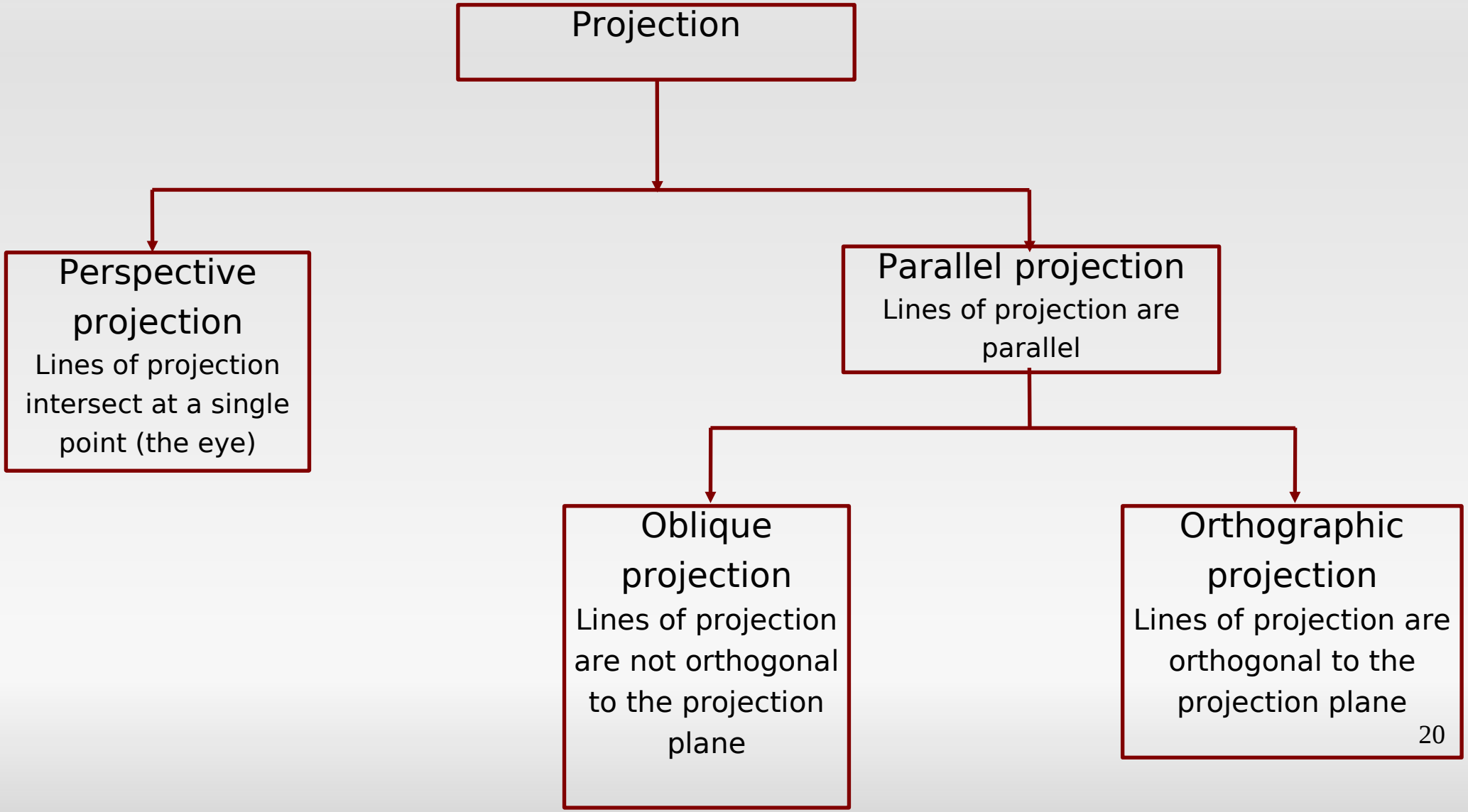
Translation along the x axis followed by a rotation

Projections

- Transformation from the world space to the image space
  - 3D world space coordinates → 2D image space coordinates
- Camera space
  - Point of view : position of the observer
  - Viewing direction : the direction in which the observer is looking
  - « Up » direction: defines the vertical direction of the camera
- Viewing volume (viewing frustum)
  - Two kinds of projections
    - Parallel projection
    - Perspective projection

Projections

- Projection family



Projection matrices

- Orthographic projection on plane ***(z = 0)*** $x' = x, y' = y, z' = 0, w' = w$

  Projection matrix: $P_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Orthographic projection on plane ***(y = 0)*** $x' = x, y' = 0, z' = z, w' = w$

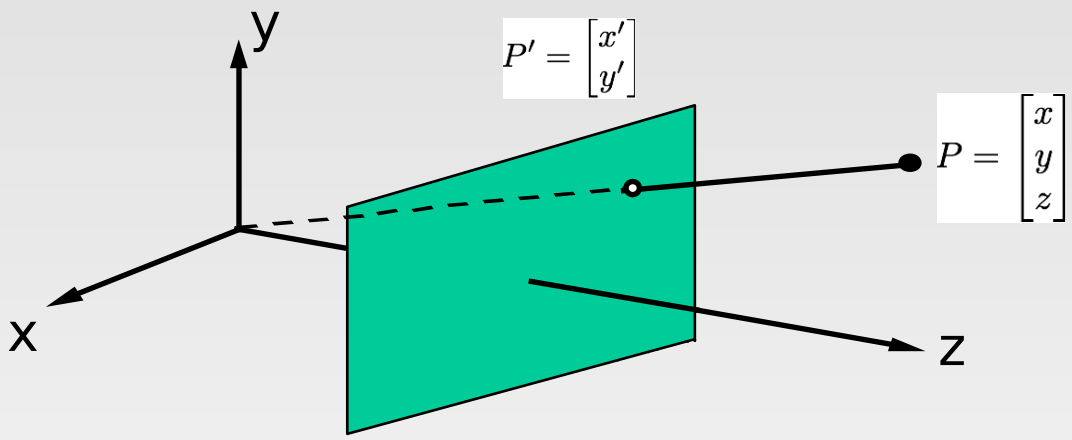
  Projection matrix: $P_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Orthographic projection on plane ***(x = 0)*** $x' = 0, y' = y, z' = z, w' = w$

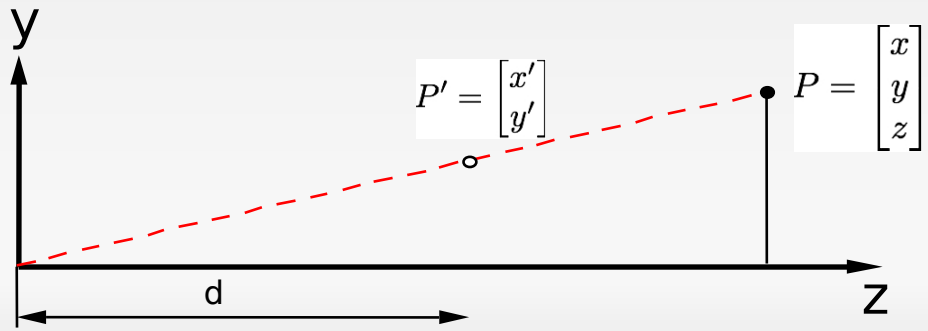
  Projection matrix $P_x = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Projection matrices

- Perspective projection on plane ***(z = d)***
  The eye is at the origin



$$P' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \frac{d}{z}. \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



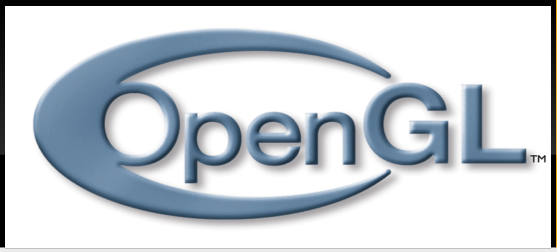
Projection matrices

- Perspective projection on plane ***(z = d)***
  - In homogeneous coordinates:

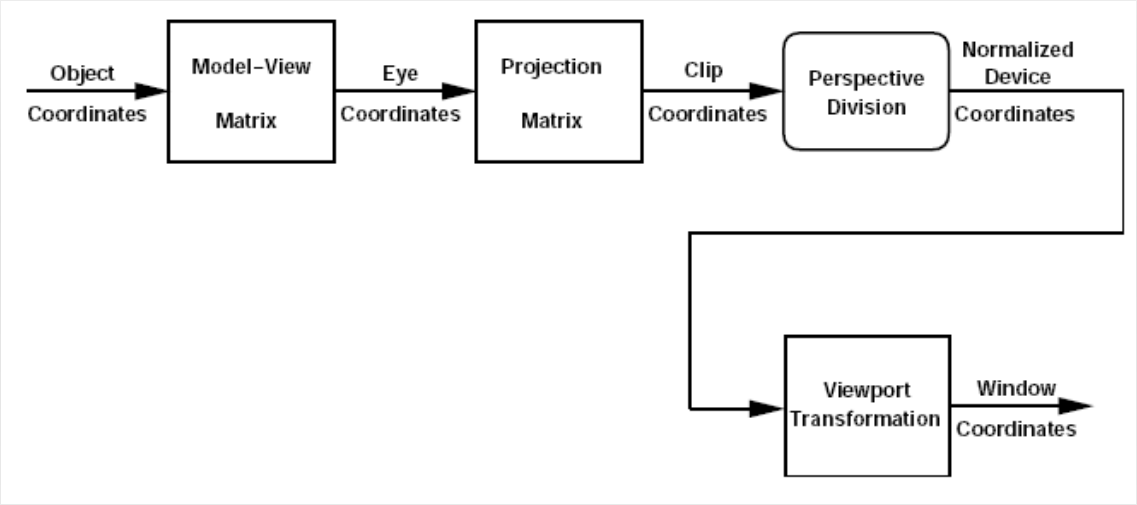
$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

$$P' = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = M_p.P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} = \begin{bmatrix} x.\frac{d}{z} \\ y.\frac{d}{z} \\ z.\frac{d}{z} \\ 1 \end{bmatrix}$$

Transformations in

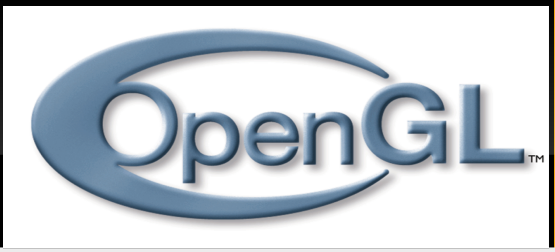


- OpenGL transformation pipeline:



- Two main transformation matrices:
  - GL_MODELVIEW
    - Positions the viewing volume
    - Positions models in the world
      - Rotations, translations, scaling, etc.
  - GL_PROJECTION
    - Determines the shape of the viewing volume
      - Orthogonal projection
      - Perspective projection

Transformations in



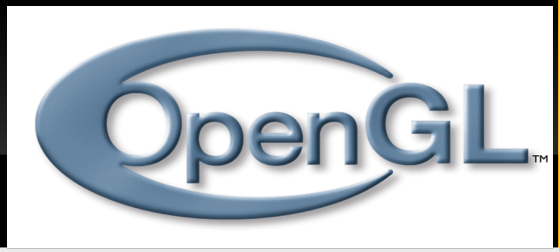
- Setting the current transformation matrix (CTM):
  - void glMatrixMode(GLenum matrix);

    'matrix' is either GL_PROJECTION or GL_MODELVIEW
- Initializing the value of the current matrix:
  - void glLoadIdentity();

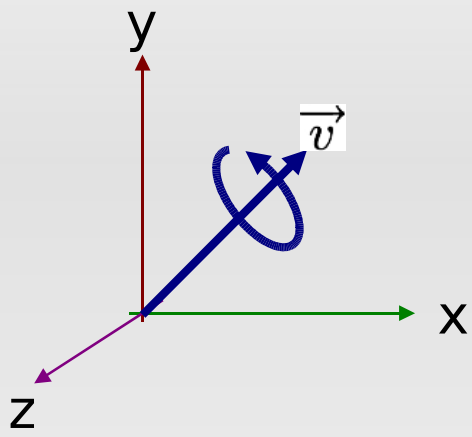
    sets the CTM to the identity matrix
- Translation:
  - void glTranslatef(GLfloat x, Glfloat y, Glfloat z);

    multiplies the CTM with a translation matrix
  - (x, y, z): coordinates of the translation vector

Transformations in



- Rotation
  - void glRotatef (GLfloat angle, Glfloat x, GLfloat y, Glfloat z); multiplies the CTM with a rotation matrix
  - The rotation is counterclockwise around vector v = (x,y,z)



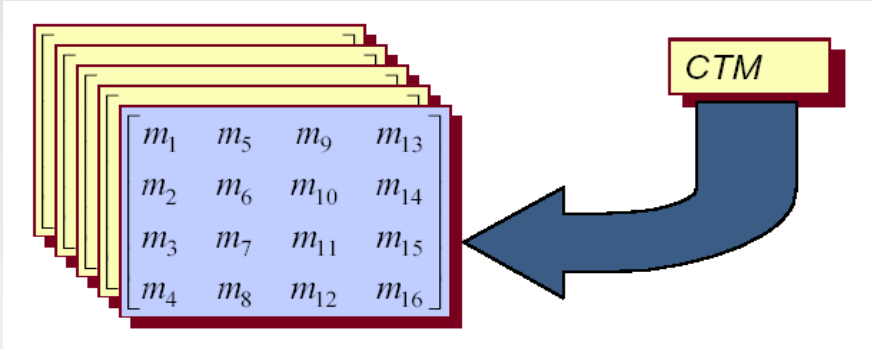
- Scaling
  - void glScalef (GLfloat scale_x, GLfloat scale_y, Glfloat scale_z); multiplies the CTM with a scaling matrix

Tutorial : non-commutativity of the transformations

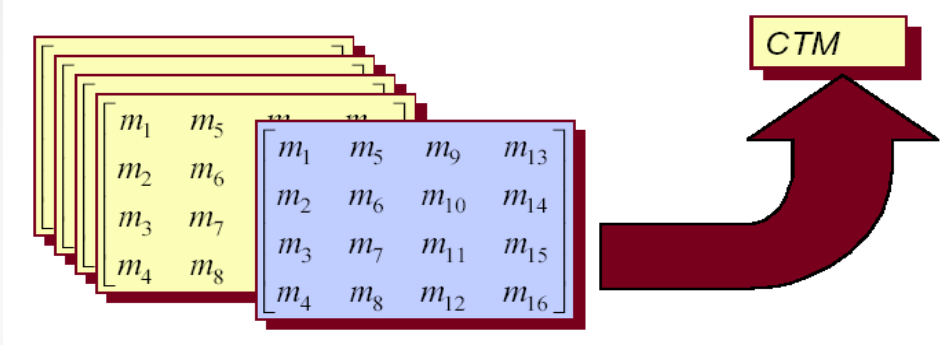
- Write a program that tests the non-commutativity of the transformations.
- A global variable 'order' will be used to determine which of two transformations (a rotation and a translation) will be applied first.
  - Keyboard event function:
    - void key (unsigned char c, int mouseX, int mouseY)
    - Key <q> to quit
    - Key <o> to switch the order of the transformations
  - Display function:
    - void display(void)
    - glutSolidTeapot(int size); // draws a teapot centered at the origin, aligned on the x axis

Geometric transformations

- OpenGL defines two matrix stacks:
  - One stack for the modelview matrices
    One stack for the projection matrices
  - void glPushMatrix(void)
    push the CTM in the current stack

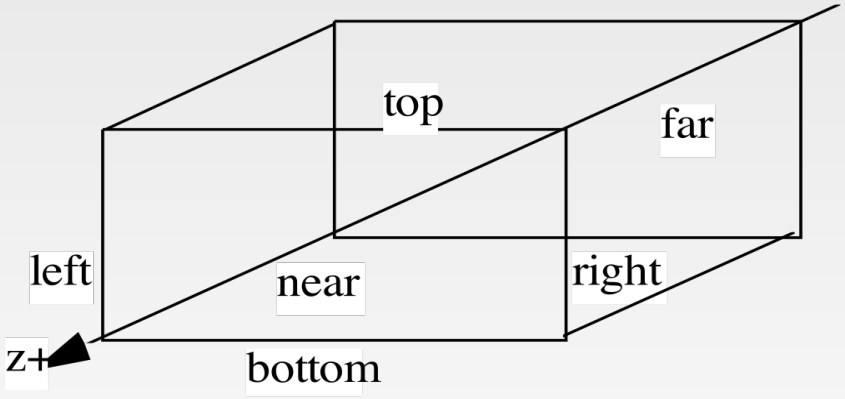


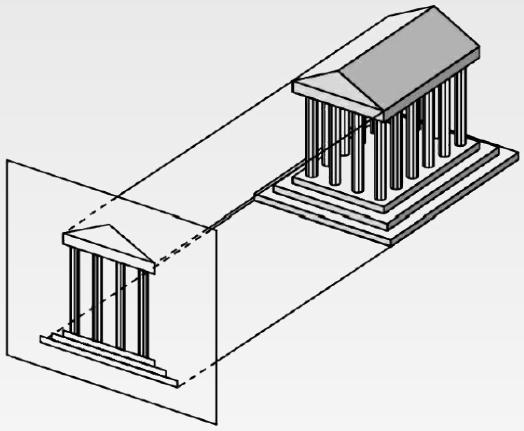
  - void glPopMatrix(void)
    restore the last state of the CTM
    stored in the current stack



Projections & viewing frustum

- Orthographic projection:
  - void glOrtho(GLfloat left, GLfloat right, GLfloat bottom, GLfloat up, GLfloat near, Glfloat far)
    multiplies the CTM by an orthographic projection matrix
  - The projection is done along the z axis

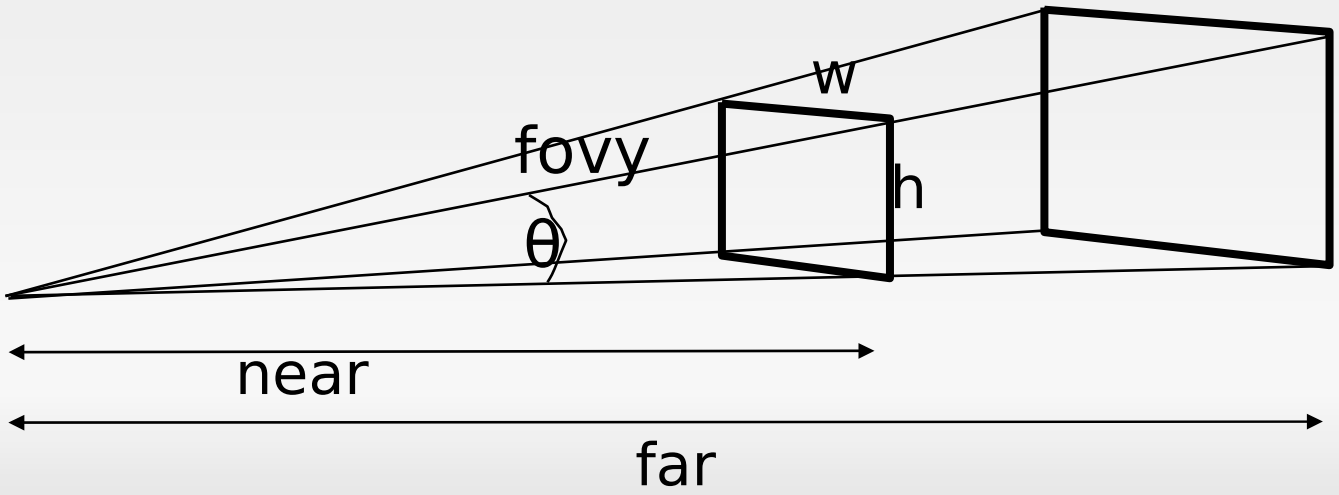


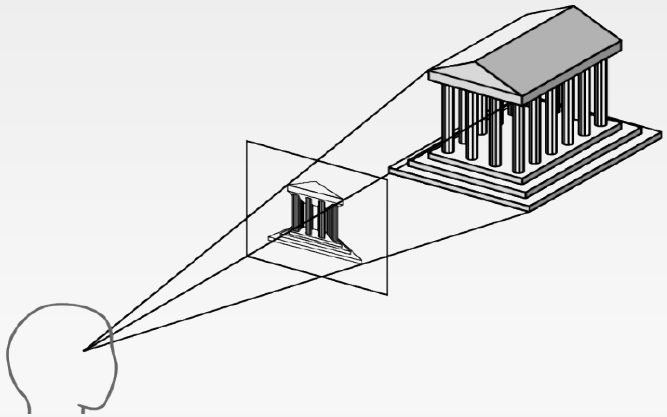


  - By default: GL_PROJECTION = Id
    Equivalent to an orthographic projection with parameters (-1, 1, -1, 1, -1, 1)

Projections & viewing frustum

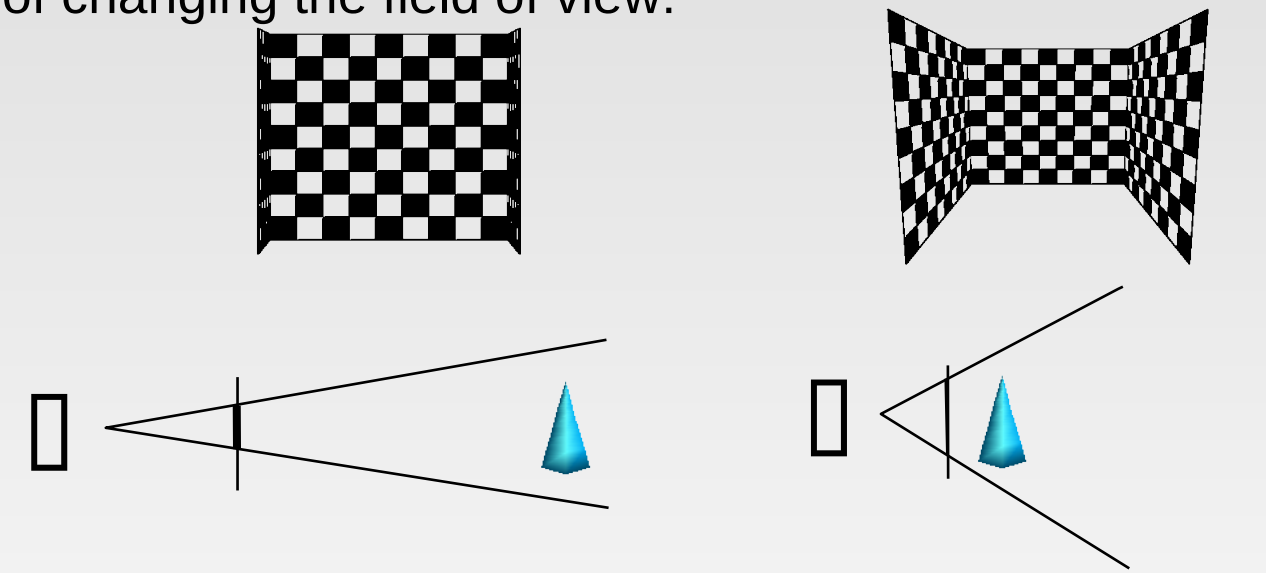
- Perspective projection:
  - void gluPerspective(GLfloat fovy, Glfloat aspect_ratio, GLfloat zNear, GLfloat zFar) multiplies the CTM by a perspective projection matrix
  - The projection is done along the z axis
  - fovy = field of view (in degrees) in the y direction
  - aspect_ratio = w/h determines the field of view in the x direction





Projections & viewing frustum

- Perspective projection:
  - Effects of changing the field of view:

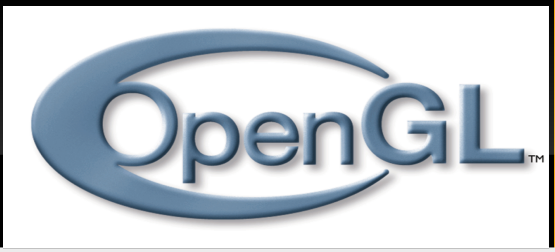


Far object, narrow fov

Close object, wide fov

  - The narrower the fov, the closer to an Orthographic projection !

Geometric transformations



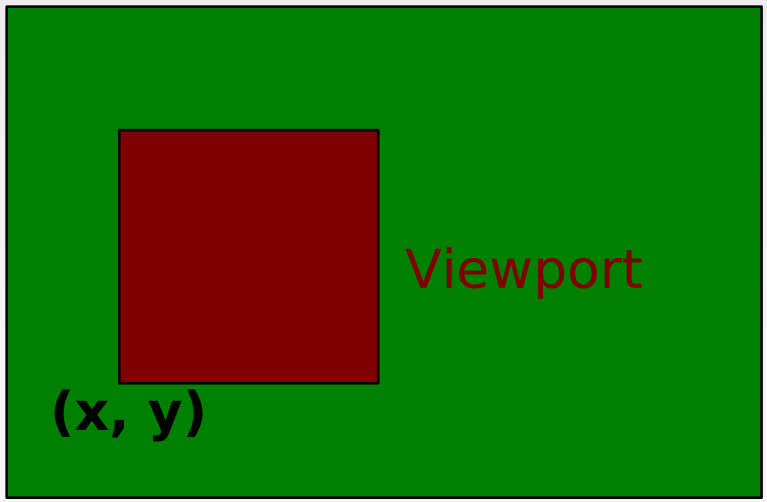
- Setting the point of view
  - void gluLookAt(GLfloat eye_x, eye_y, eye_z,
    center_x, center_y, center_z,
    up_x, up_y, up_z);

    multiplies the CTM by a viewing matrix
  - (eye_x, eye_y, eye_z): position of the camera in the world
  - (center_x, center_y, center_z): a point in the scene, aimed at by the camera (defines the direction of the camera)
  - (up_x, up_y, up_z): defines the verticality of the camera

Geometric transformations

- Setting the viewport
  - void glViewport(GLuint x, GLuint y, GLuint width, GLuint height);
  - Maps the image plane to an area of the window

    Defines the area of the window in which the rendering will be done



Tutorial: viewing volume and viewport

- Draw a teapot of size 1 using the default viewing volume
  - Explain the result with a figure
- Modify the viewing volume in order to see a well-proportionned teapot, then change it again so that the teapot fits the screen
- Using glViewport
  - What parameters should we give to glViewport so that the teapot is drawn in a 300*300 frame at the bottom left of the window? (3)
  - Change the viewing volume so that the teapot is well proportionned again (4)
  - Draw 2 teapots in 2 different viewports (5)
- Move the teapot by 100 along the z axis (6)
  - Change the viewing volume in order to see the teapot again (7)
  - Apply some modeling transformations to the teapot (8,9)
- Perspective