Fundamentals of
Computer Graphics and Image Processing

Transformations, Projections (02)

doc. RNDr. Martin Madaras, PhD.
martin.madaras@fmph.uniba.sk

Overview

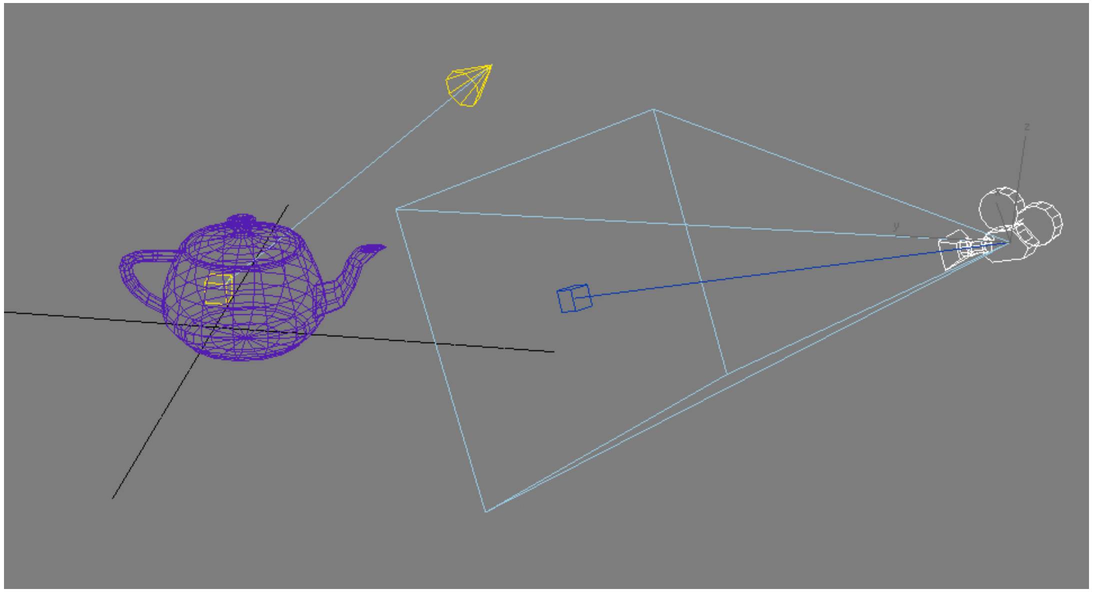
- 2D Transformations
  - **Basic 2D transformations**
  - Matrix representation
  - Matrix composition
- 3D Transformations
  - Basic 3D transformations
  - Same as 2D

How the lectures should look like #1

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!
- We will go into depth as far, as there are no questions

Geometry space

- Scene
  - Virtual representation of world
- Objects
  - Visible objects ("real world")
  - Invisible objects (e.g. lights, cameras, etc.)

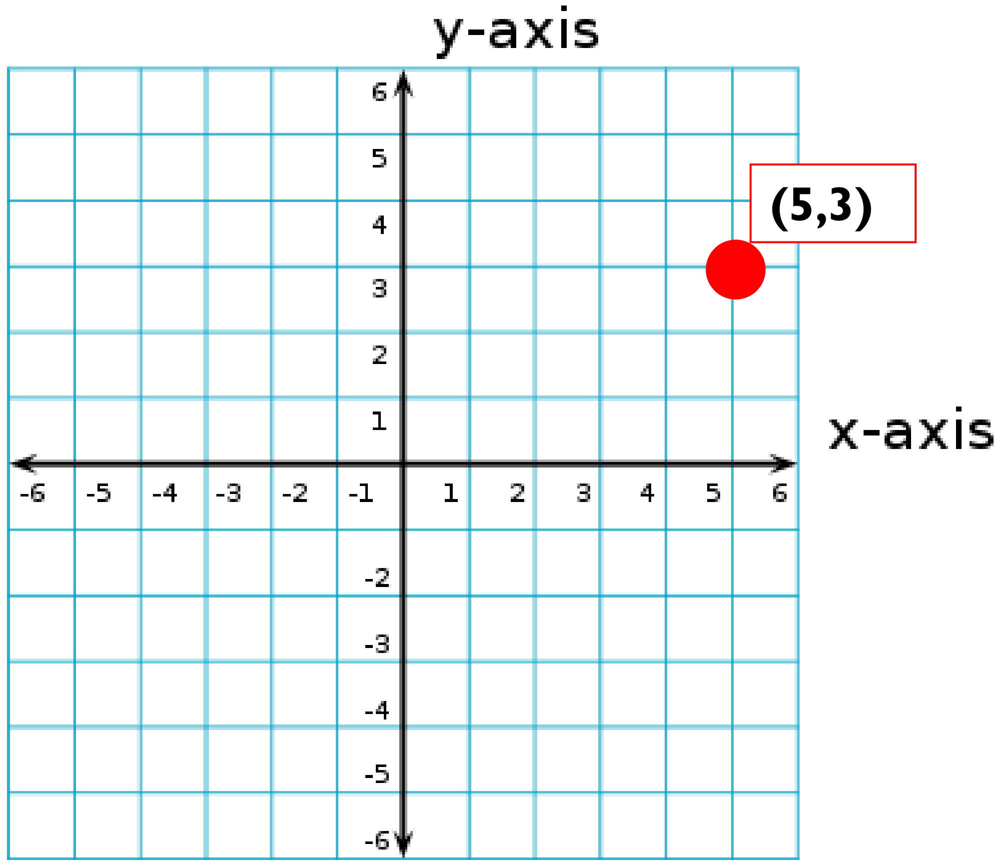


Full scene definition

- Objects
  - What objects, where, how transformed
    - To be discussed early during course
  - How they look – color, material, texture...
    - To be discussed later during course
- Camera
  - Position, target, camera parameters

Coordinate system

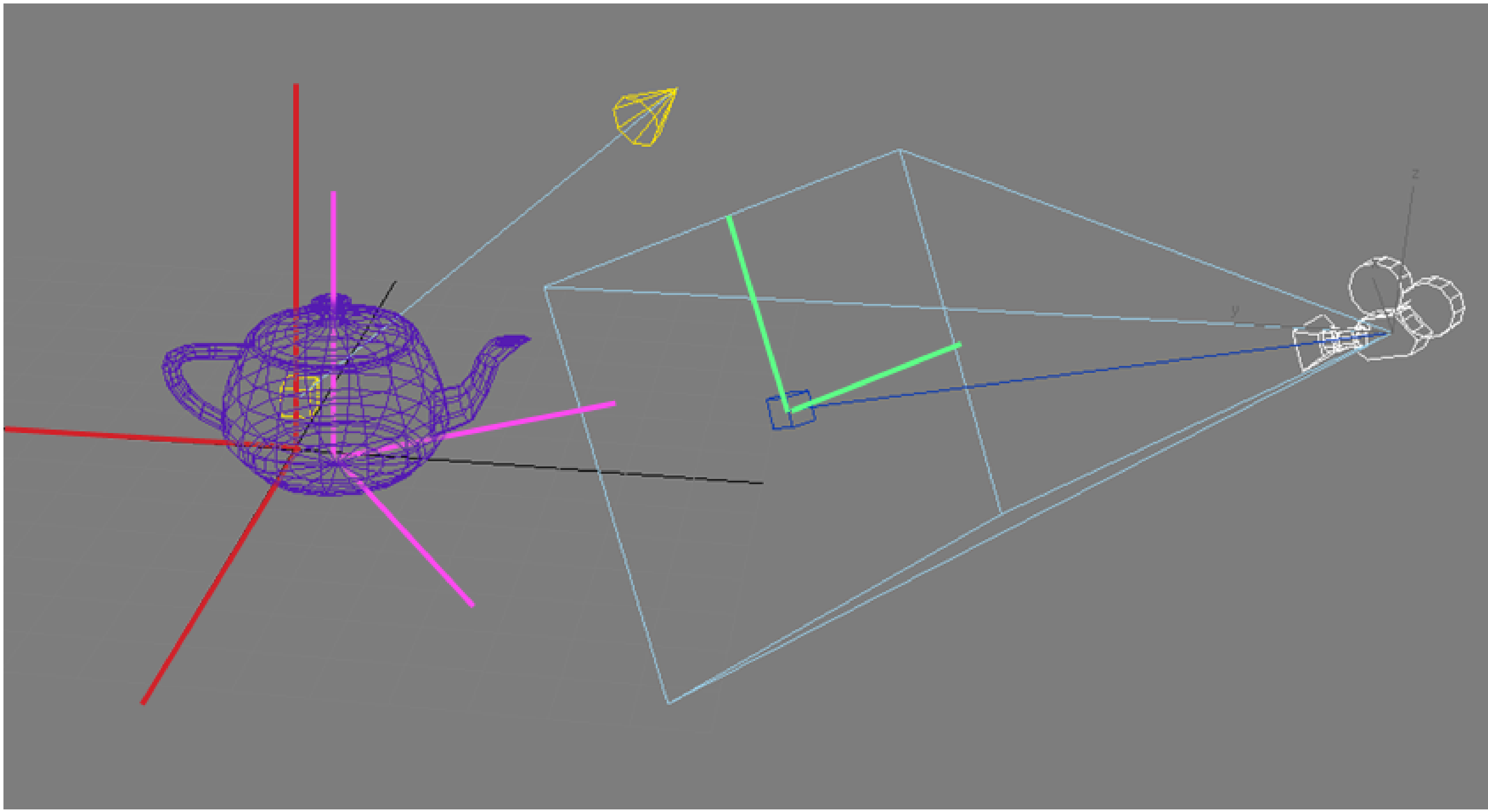
- Cartesian coordinates in 2D
  - Origin
  - x axis
  - y axis



Coordinate systems

- Global
  - One for whole scene
- Local
  - Individual for every model
  - Pivot point
- Camera coordinates
- Window coordinates
- Units may differ
- Conversion between coordinate spaces

Global/local/camera coords.



Essential basic algebra

Point

- Position in space
- Cartesian coordinates

$$(x, y)$$
$$(x, y, z)$$

- Homogeneous coordinates

$$(x, y, 1)$$
$$(x, y, z, 1)$$

- Notation: **P**, **A**, …

Vector

- Direction in space,
- Has no position
- Subtraction of points

- Cartesian coordinates

$$(x, y)$$
$$(x, y, z)$$

- Homogeneous coordinates

$$(x, y, 0)$$
$$(x, y, z, 0)$$

- Notation: $\vec{u}, \vec{v}, \vec{n}$

Vector

- Addition

  Point + vector = point

  Vector + vector = vector

- Subtraction

  Point – point = vector

  Point – vector = point + (-vector) = point

  Vector – vector = vector + (-vector) = vector

- Multiplication

  Multiplier * vector = vector

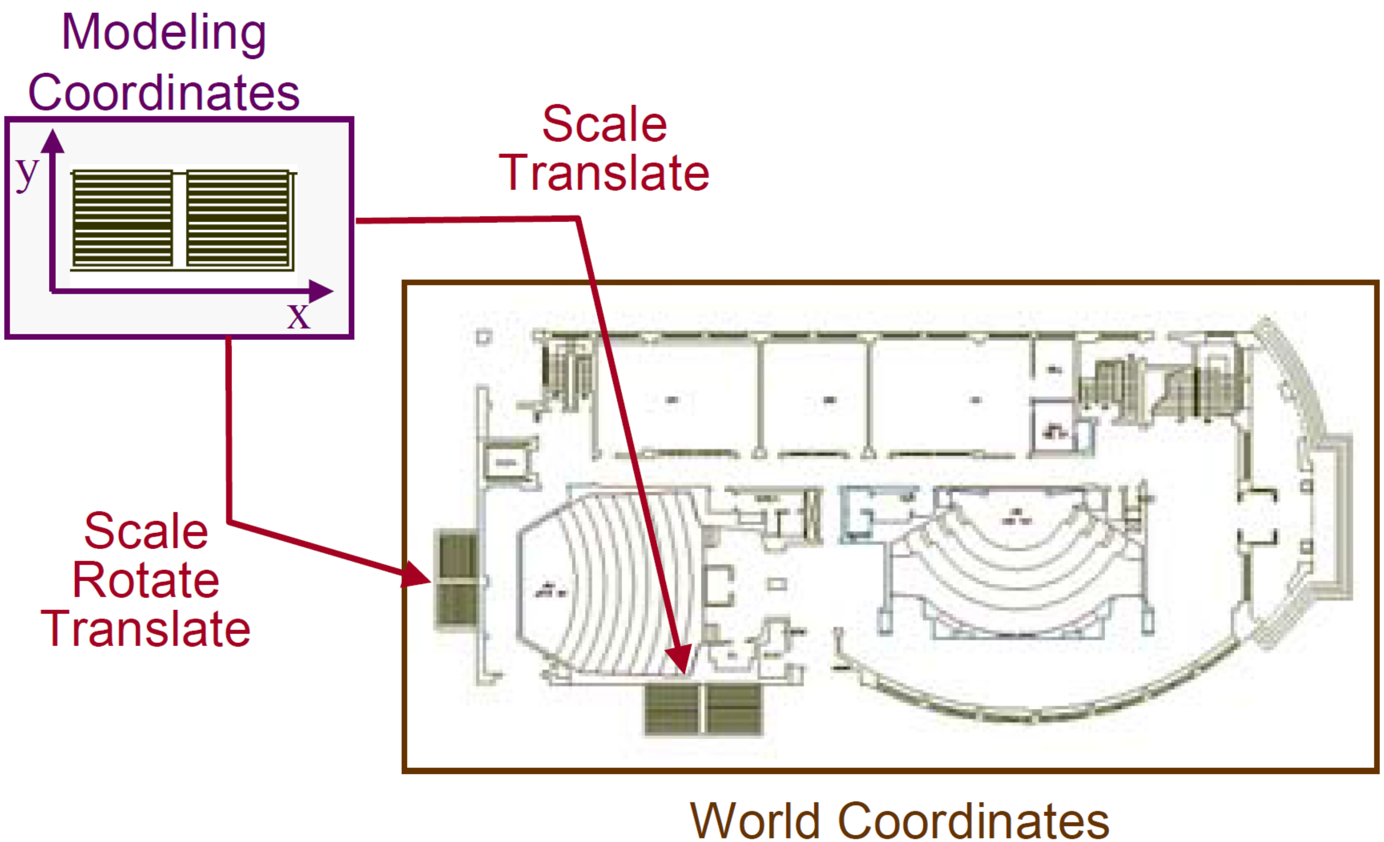
Ask questions

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!

Transformations

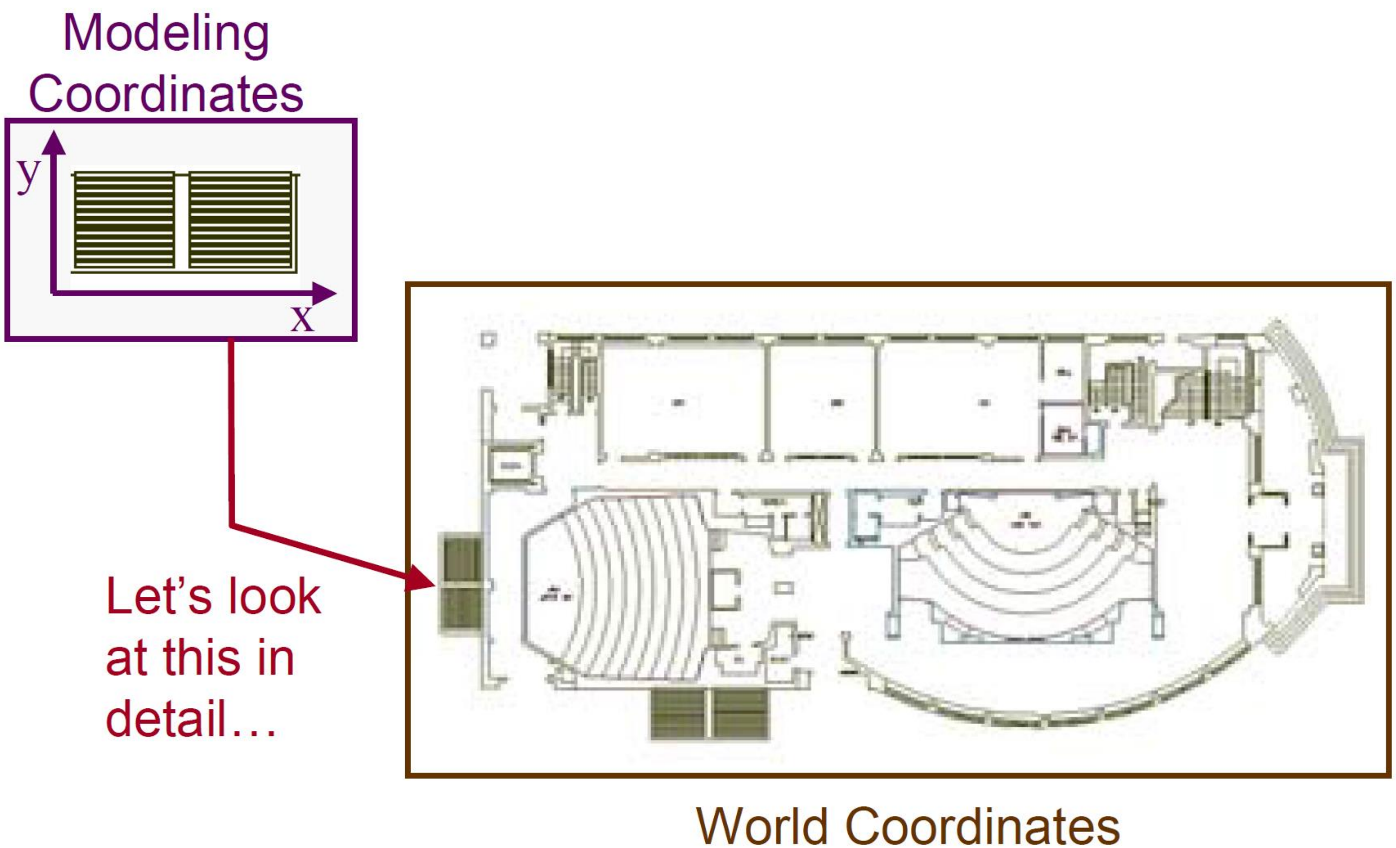
2D Modeling Transformations

- Modeling space and World space



2D Modeling Transformations

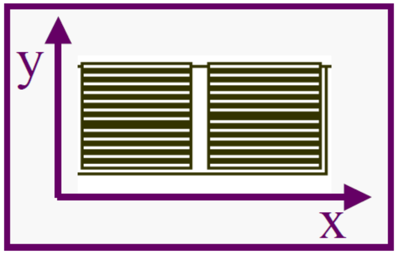
- Transformed instances

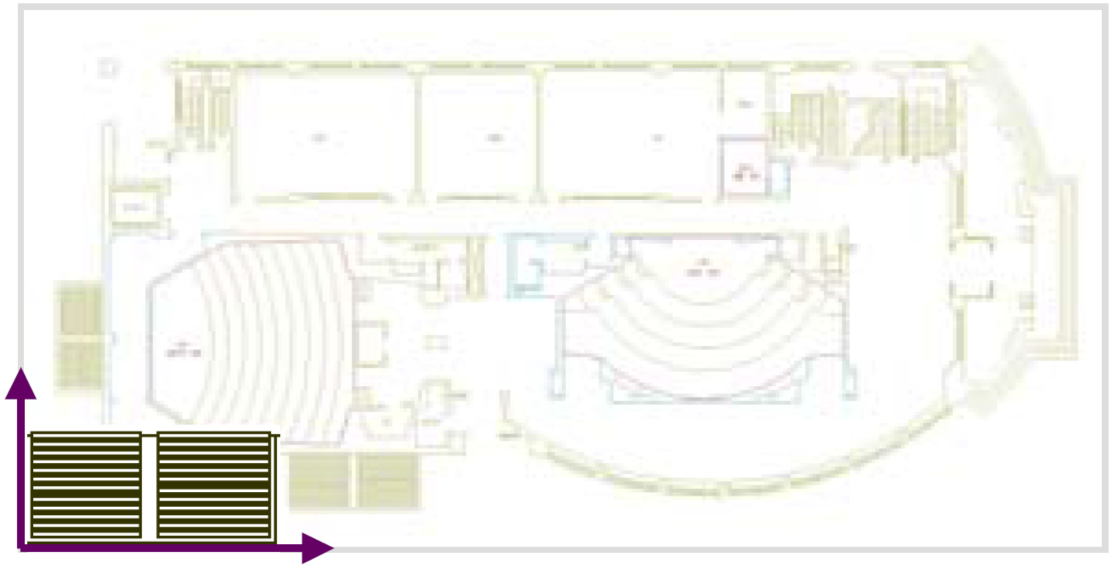


2D Modeling Transformations

- Transformation identity

Modeling
Coordinates

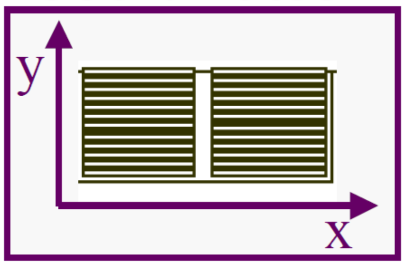




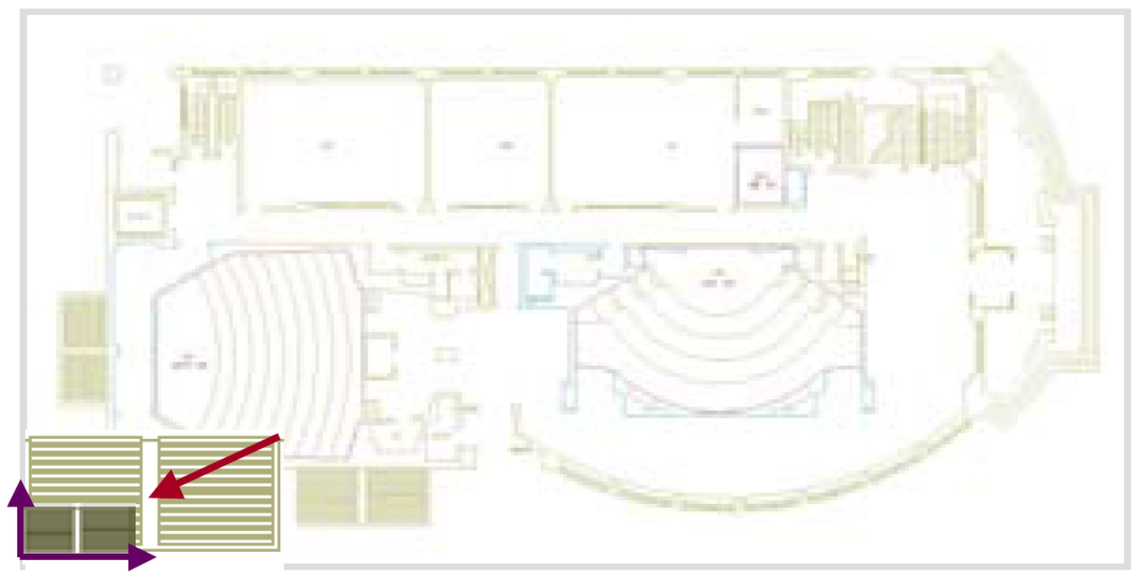
2D Modeling Transformations

- Scaling applied…

Modeling Coordinates



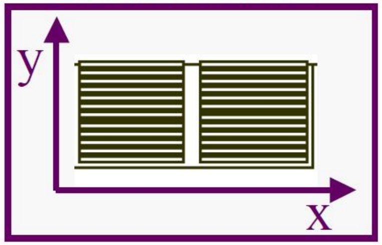
Scale .3, .3



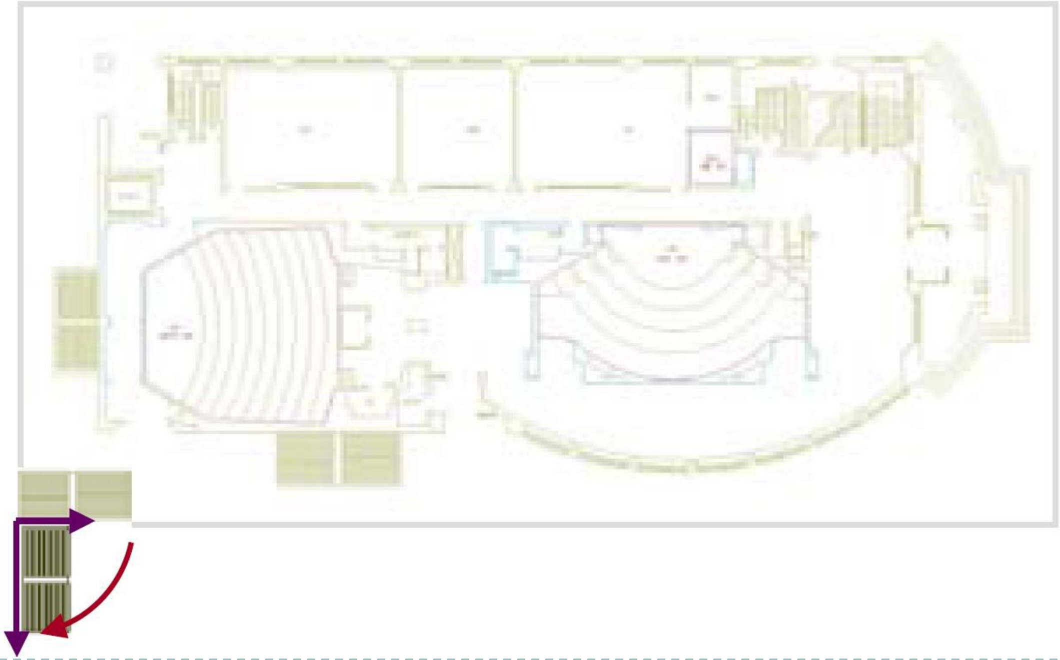
2D Modeling Transformations

- Scaling and rotation applied…

Modeling
Coordinates

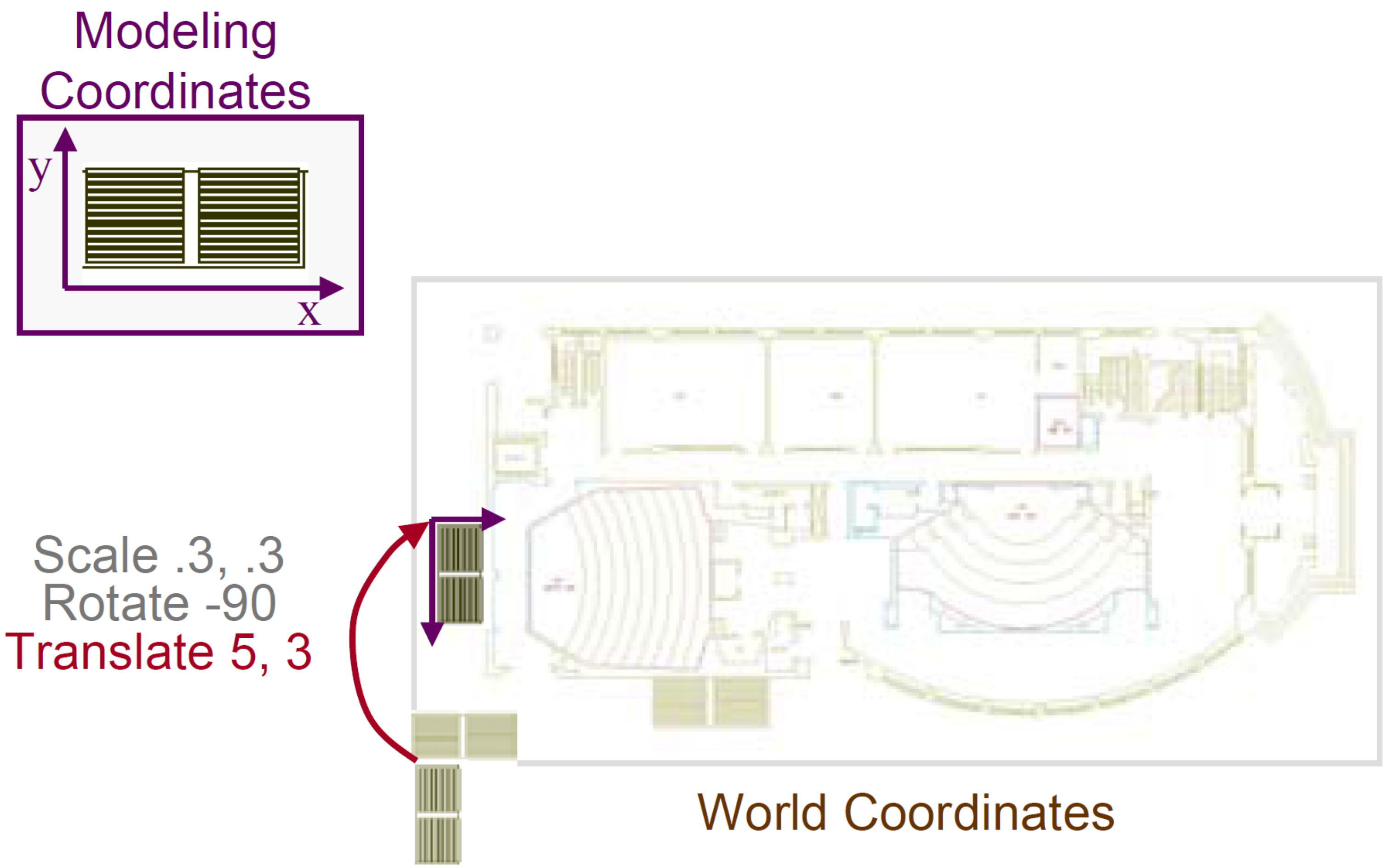


Scale .3, .3
Rotate -90



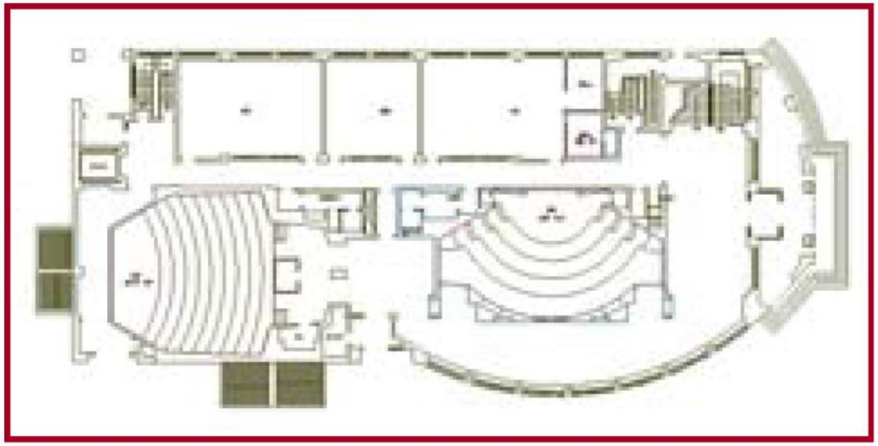
2D Modeling Transformations

- Scaling, rotation and translation applied…



2D Modeling Transformations

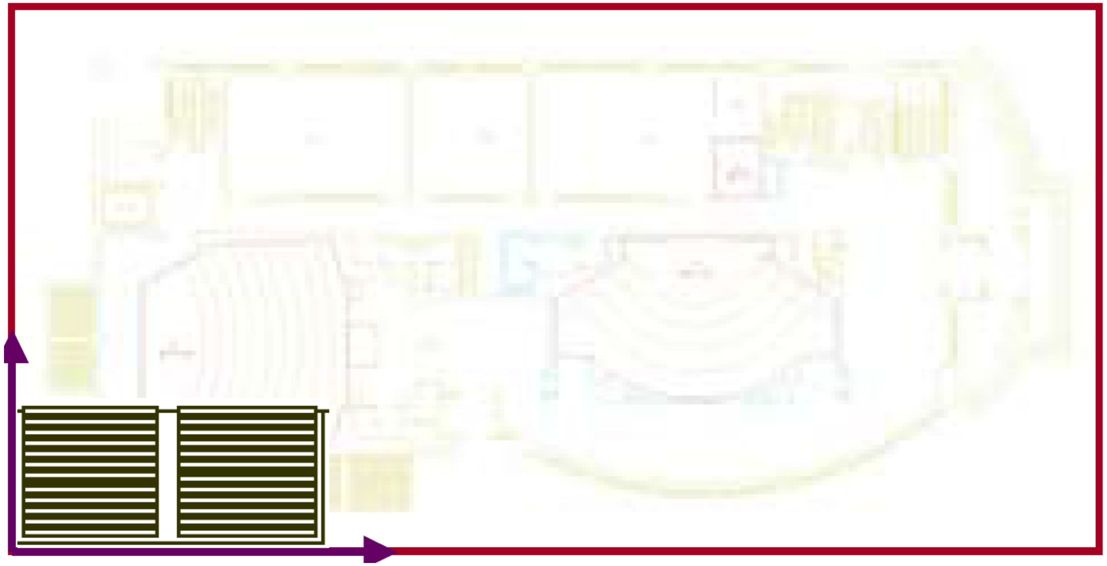
- Translation:
  - $x' = x + t_x$
  - $y' = y + t_y$
- Scale:
  - $x' = x * s_x$
  - $y' = y * s_y$
- Rotation:
  - $x' = x*\cos\Theta - y*\sin\Theta$
  - $y' = x*\sin\Theta + y*\cos\Theta$



Transformations can be combined (with simple algebra)

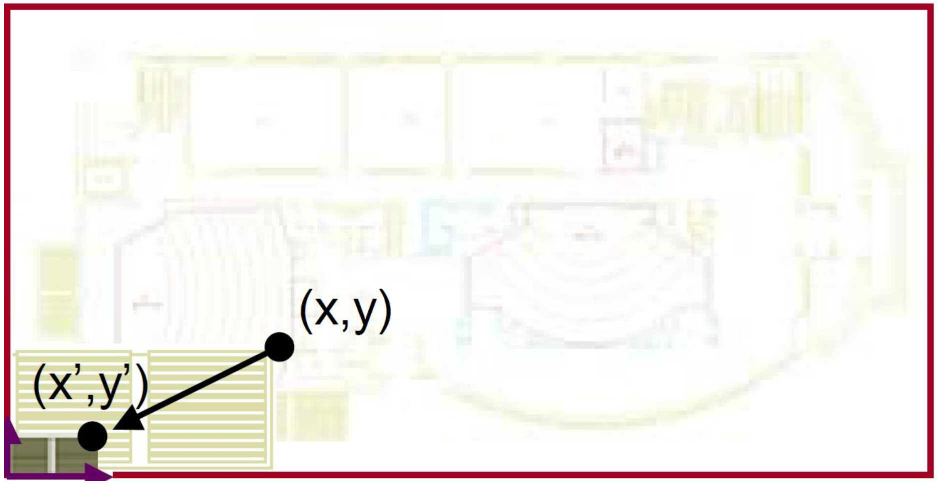
2D Modeling Transformations

- Translation:
  - $x' = x + t_x$
  - $y' = y + t_y$
- Scale:
  - $x' = x * s_x$
  - $y' = y * s_y$
- Rotation:
  - $x' = x*\cos\Theta - y*\sin\Theta$
  - $y' = x*\sin\Theta + y*\cos\Theta$



2D Modeling Transformations

- Translation:
  - x' = x + t_x
  - y' = y + t_y
- **Scale**:
  - x' = x * s_x
  - y' = y * s_y
- Rotation:
  - x' = x*cosΘ - y*sinΘ
  - y' = x*sinΘ + y*cosΘ



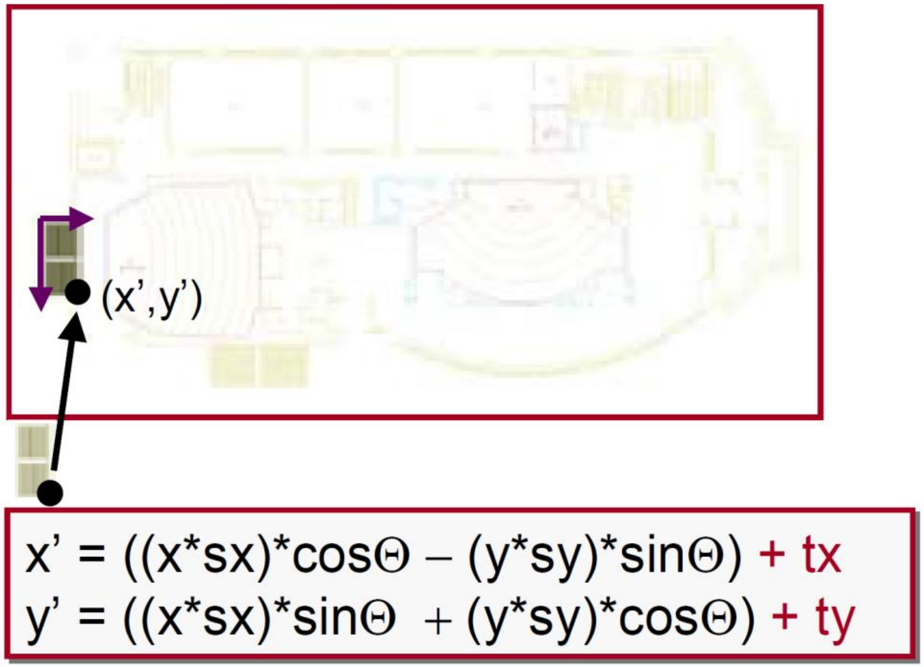
2D Modeling Transformations

- Translation:
  - $x' = x + t_x$
  - $y' = y + t_y$
- Scale:
  - $x' = x * s_x$
  - $y' = y * s_y$
- **Rotation**:
  - x' = x*cosΘ - y*sinΘ
  - y' = x*sinΘ + y*cosΘ

x' = (x*sx)*cosΘ – (y*sy)*sinΘ
y' = (x*sx)*sinΘ + (y*sy)*cosΘ

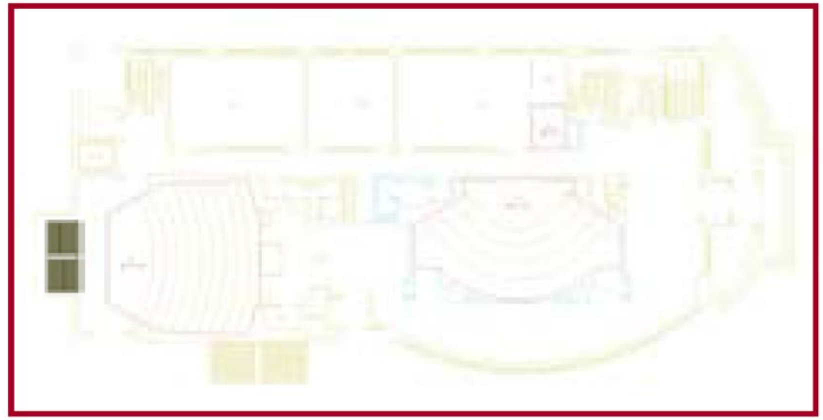
2D Modeling Transformations

- **Translation**:
  - x’ = x $+ t_x$
  - y’ = y $+ t_y$
- Scale:
  - x’ = x $* s_x$
  - y’ = y $* s_y$
- Rotation:
  - x’ = x*cosΘ - y*sinΘ
  - y’ = x*sinΘ + y*cosΘ



2D Modeling Transformations

- Translation:
  - $x' = x + t_x$
  - $y' = y + t_y$
- Scale:
  - $x' = x * s_x$
  - $y' = y * s_y$
- Rotation:
  - x' = x*cosΘ - y*sinΘ
  - y' = x*sinΘ + y*cosΘ



x' = ((x*sx)*cosΘ – (y*sy)*sinΘ) + tx
y' = ((x*sx)*sinΘ + (y*sy)*cosΘ) + ty

Overview

- 2D Transformations
  - Basic 2D transformations
  - **Matrix representation**
  - Matrix composition
- 3D Transformations
  - Basic 3D transformations
  - Same as 2D

Matrix Representation

- Represent 2D transformation by a matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- Multiply matrix by column vector $\Leftrightarrow$ transformation of a point

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = ax + by$$

$$y' = cx + dy$$

Matrix Representation

- Transformations combined by multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrices are a convenient and efficient way to represent a sequence of transformations!

2x2 Matrices

- What transformations can be represented by a 2x2 matrix?
  - 2D Identity?

$$x' = x \qquad y = y \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

  - 2D Scale around origin (0,0)?

$$x' = s_x * x \qquad y = s_y * y \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- What transformations can be represented by a 2x2 matrix?
  - 2D Rotate around origin (0,0)?

$$x' = \mathrm{x} * \cos\theta - \mathrm{y} * \sin\theta$$

$$y' = \mathrm{x} * sin\,\theta + \mathrm{y} * cos\,\theta$$

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

  - 2D Shear?

$$x' = x + sh_x * y$$

$$y' = sh_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- What transformations can be represented by a 2x2 matrix?
  - 2D Mirror over Y axis?

$$x' = -x$$

$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

  - 2D Mirror over (0,0)?

$$x' = -x$$

$$y' = -y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- What transformations can be represented by a 2x2 matrix?
  - 2D Translation?

$$x' = x + t_x$$

$$y' = y + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Only a linear 2D transformations can be represented by a 2x2 matrix

Linear Transformations

- Linear transformations are combinations of ..
  - Scale
  - Rotation
  - Shear
  - Mirror
- Properties of linear transformations:
  - Satisfies: $T(s_1 p_1 + s_2 p_2) = s_1 T(p_1) + s_2 T(p_1)$
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

2D Translation

- 2D translation represented by a 3x3 matrix
- Point represented in *homogenous coordinates*

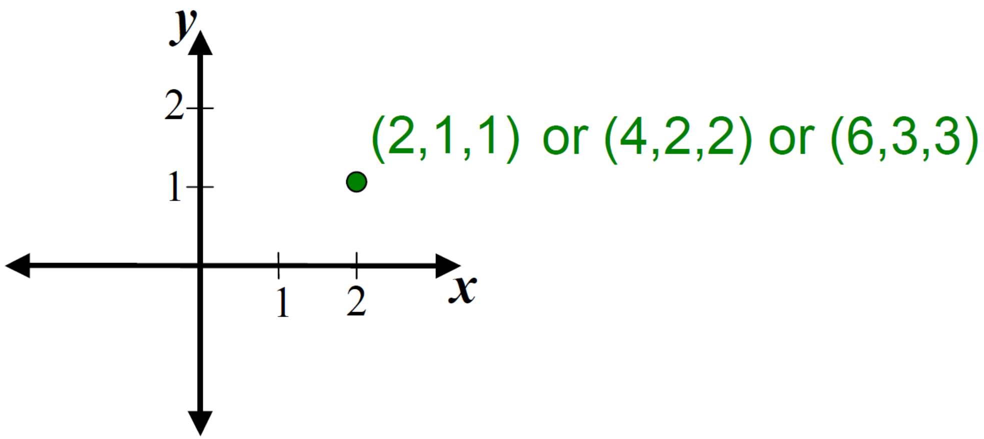
$$x' = x + t_x$$

$$y' = y + t_y$$

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

Homogenous Coordinates

- Add a 3rd coordinate to every 2D point
  - (x,y,w) represents a point at location (x/w, y/w)
  - (x,y,0) represents a point at infinity
  - (0,0,0) is not allowed



Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

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

Translate

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

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Affine Transformations

- Affine transformations are combinations of…
  - Linear transformations, and
  - Translations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

Projective Transformations

- Projective transformations:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition

Overview

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
  - **Matrix composition**
- 3D Transformations
  - Basic 3D transformations
  - Same as 2D

Matrix Composition

- Transformations can be combined by matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

P' = T(t_x, t_y) R(θ) S(s_x, s_y) P

Matrix Composition

- Matrices are a convenient and efficient way to represent a sequence of transformations
- General purpose representation
- Hardware matrix multiply
- Efficiency with premultiplication
  - Matrix multiplication is associative

$$P' = (T * (S * (R * p)))$$
$$P' = (T * S * R) * p$$

Matrix Composition

- Be aware: order of transformations matters
  - Matrix multiplication is not commutative

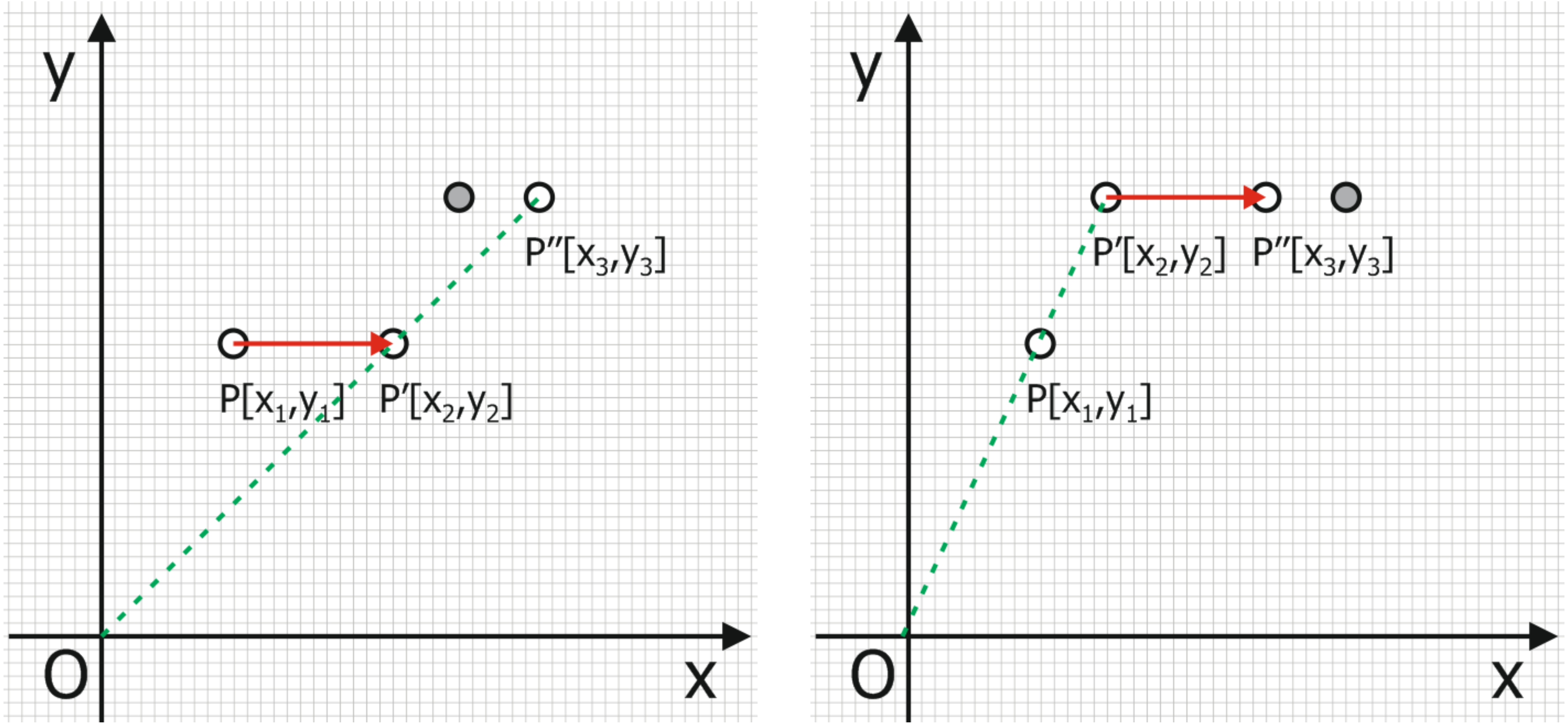
transformation order

$$P' = T * S * R * p$$

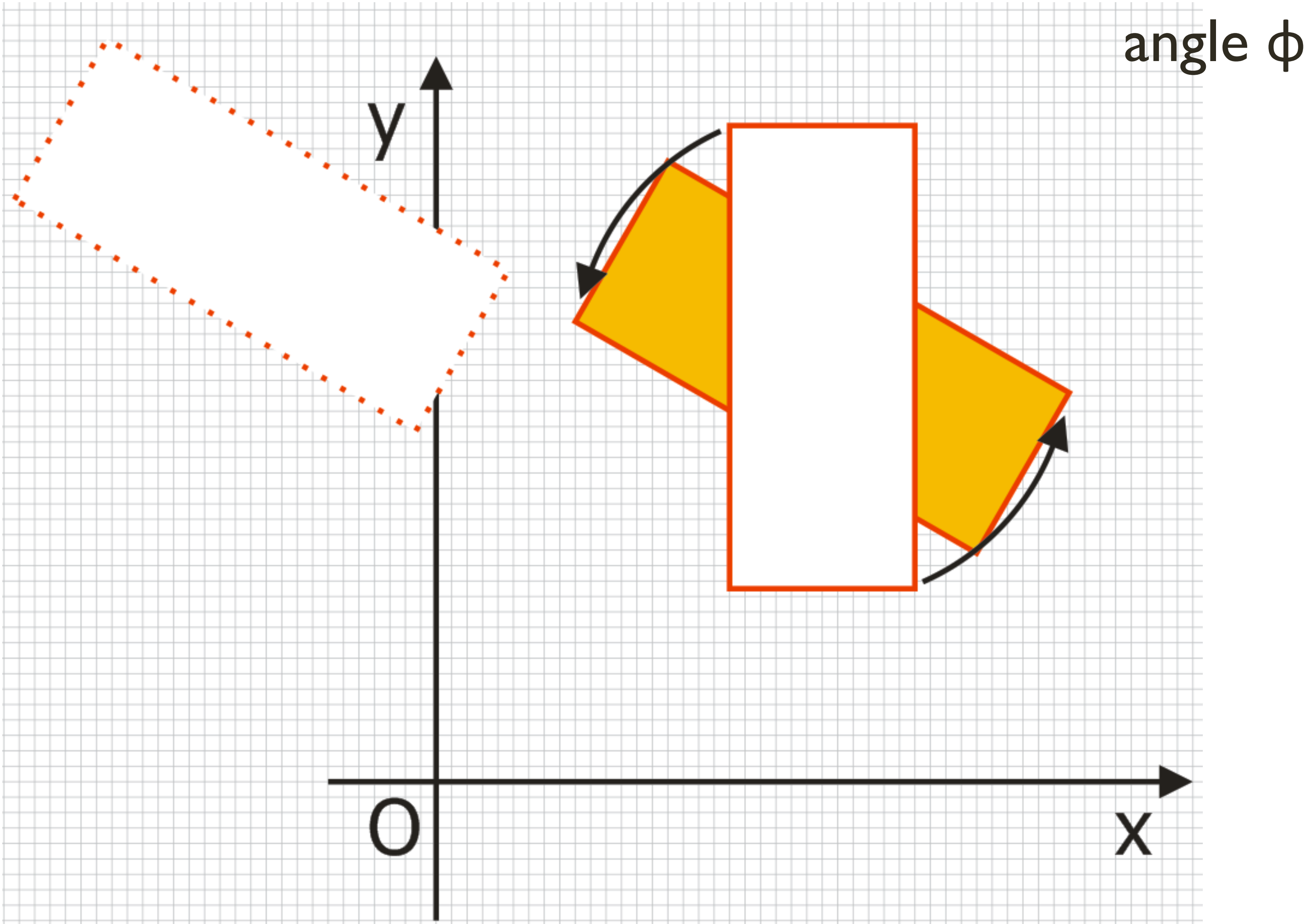
"Global" "Local"

Matrix Composition

- Be aware: order of transformations matters
  - Matrix multiplication is not commutative

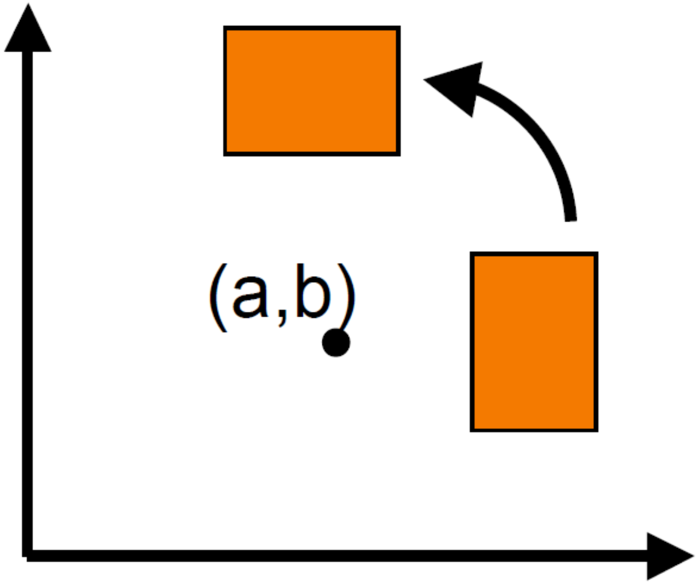


Problem: local rotation



Matrix Composition

- Rotate around an arbitrary point (a,b)
  - Translate (a,b) to the origin
  - Rotate around origin
  - Translate back



Matrix Composition

- Rotate around an arbitrary point (a,b)
  - Translate (a,b) to the origin
  - Rotate around origin
  - Translate back

$$M = T(a, b) * R(\theta) * T(-a, -b)$$

Matrix Composition

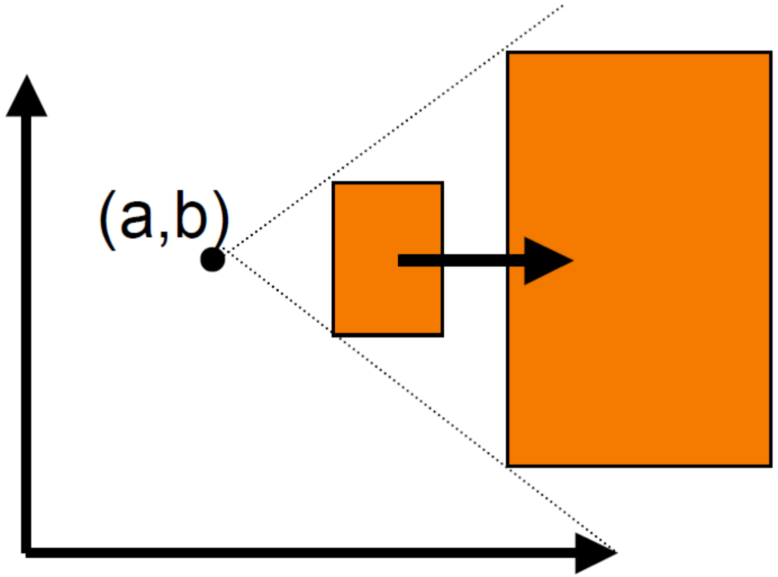
1. translate rotation center to origin: $t(t_x, t_y)$
2. rotate by ϕ
3. inverse translate by $t'(-t_x, -t_y)$

Matrix notation:

$$(x', y', 1) = (x, y, 1)\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix}\begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_x & -t_y & 1 \end{pmatrix}$$

Matrix Composition

- Scale by sx, sy around arbitrary point (a, b)
  - Use the same approach …



Matrix Composition

- Scale by sx, sy around arbitrary point (a, b)
  - Use the same approach …

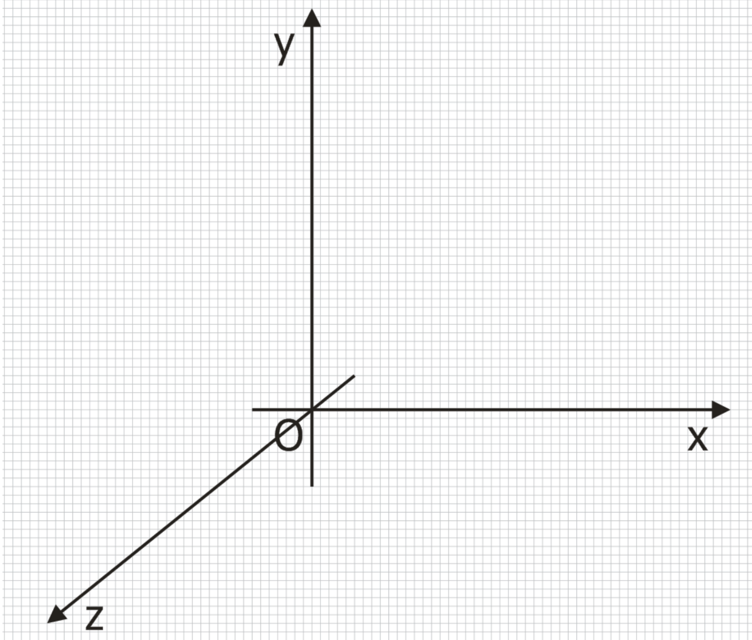
M = T(a, b) * S(s_x, s_y) * T(-a, -b)

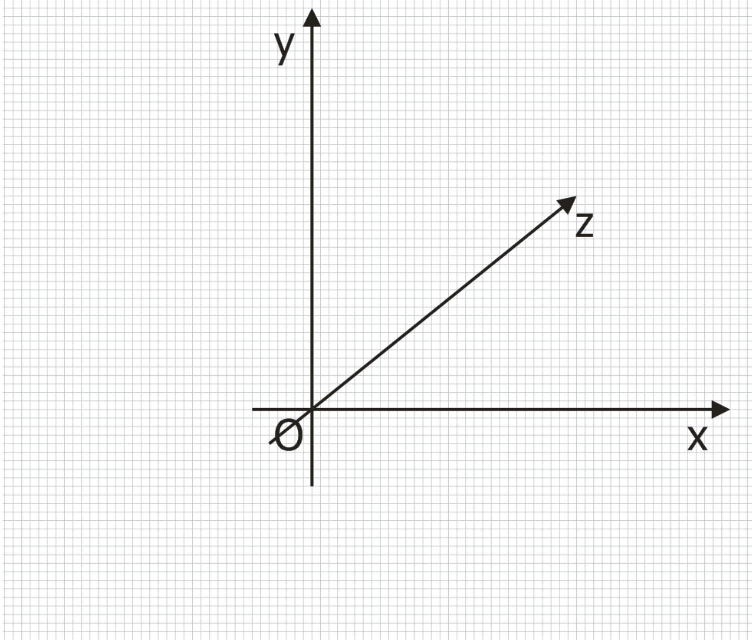
Overview

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
  - Matrix composition
- 3D Transformations
  - **Basic 3D transformations**
  - Same as 2D

3D Transformations

- Right-handed coordinate system
- Left-handed coordinate system





- rotation direction

3D Transformations

- Same idea as 2D transformations
  - Homogenous coordinates (x,y,z,w)
  - 4x4 transformation matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Basic 3D Transformations

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Identity

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

Scale

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

Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Mirror over X

Basic 3D Transformations

- Rotation around Z axis

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

- Rotation around Y axis

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

- Rotation around X axis

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

Same as 2D

- Everything else is the same as 2D
  - In fact 2D is actually 3D in OpenGL
    - z is either 0 or ignored

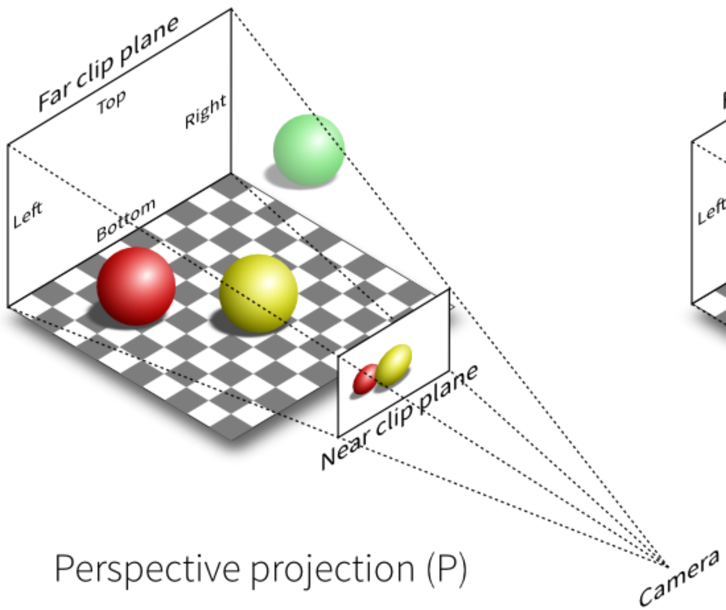
How the lectures should look like #3

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!

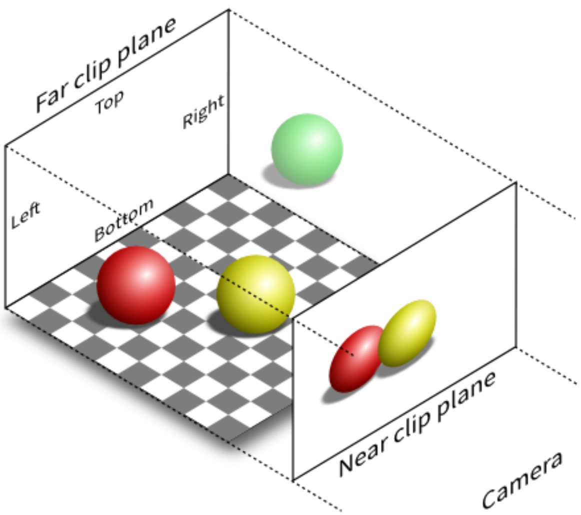
Projections

Projection

- General definition:
  - Maps points in n-space to m-space (m<n)
- In Computer Graphics:
  - Map 3D camera coordinates to 2D screen coordinates



Perspective projection (P)



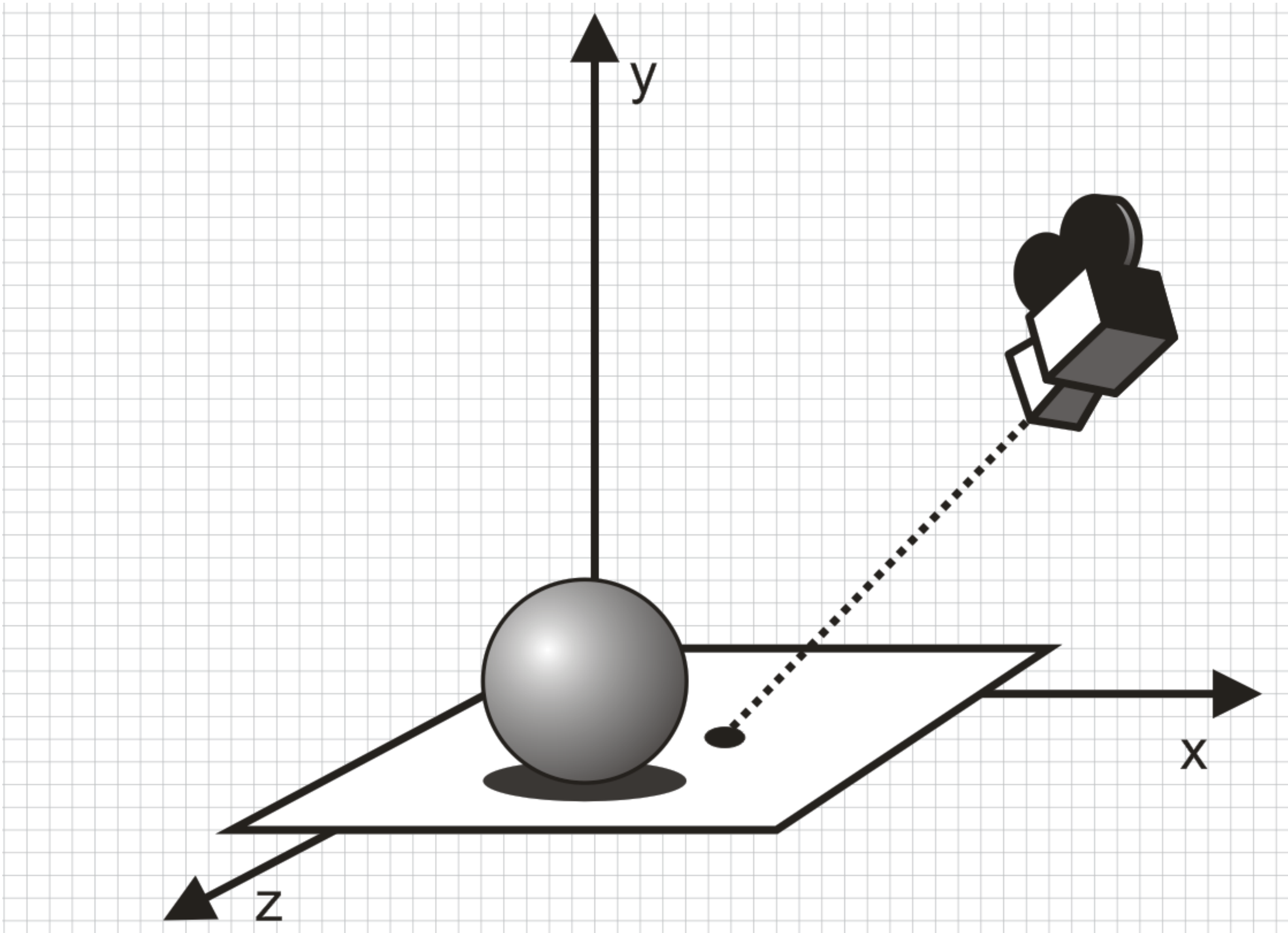
Orthographic projection (O)

Viewing transformation

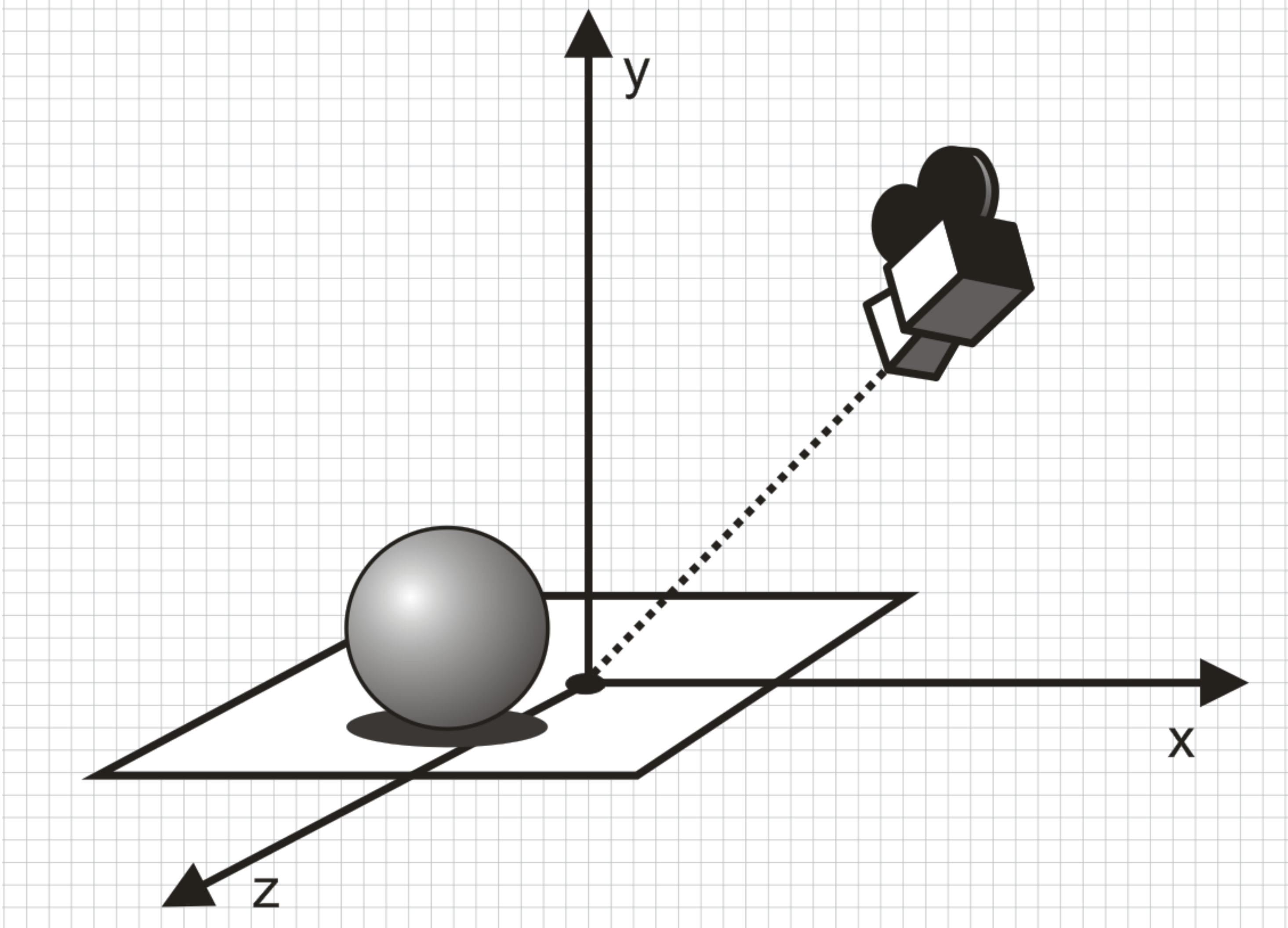
- Convert from local/world coordinates to camera/viewport coordinates

1. rotate scene so that camera lies in z-axis
2. projection transformation
3. viewport transformation

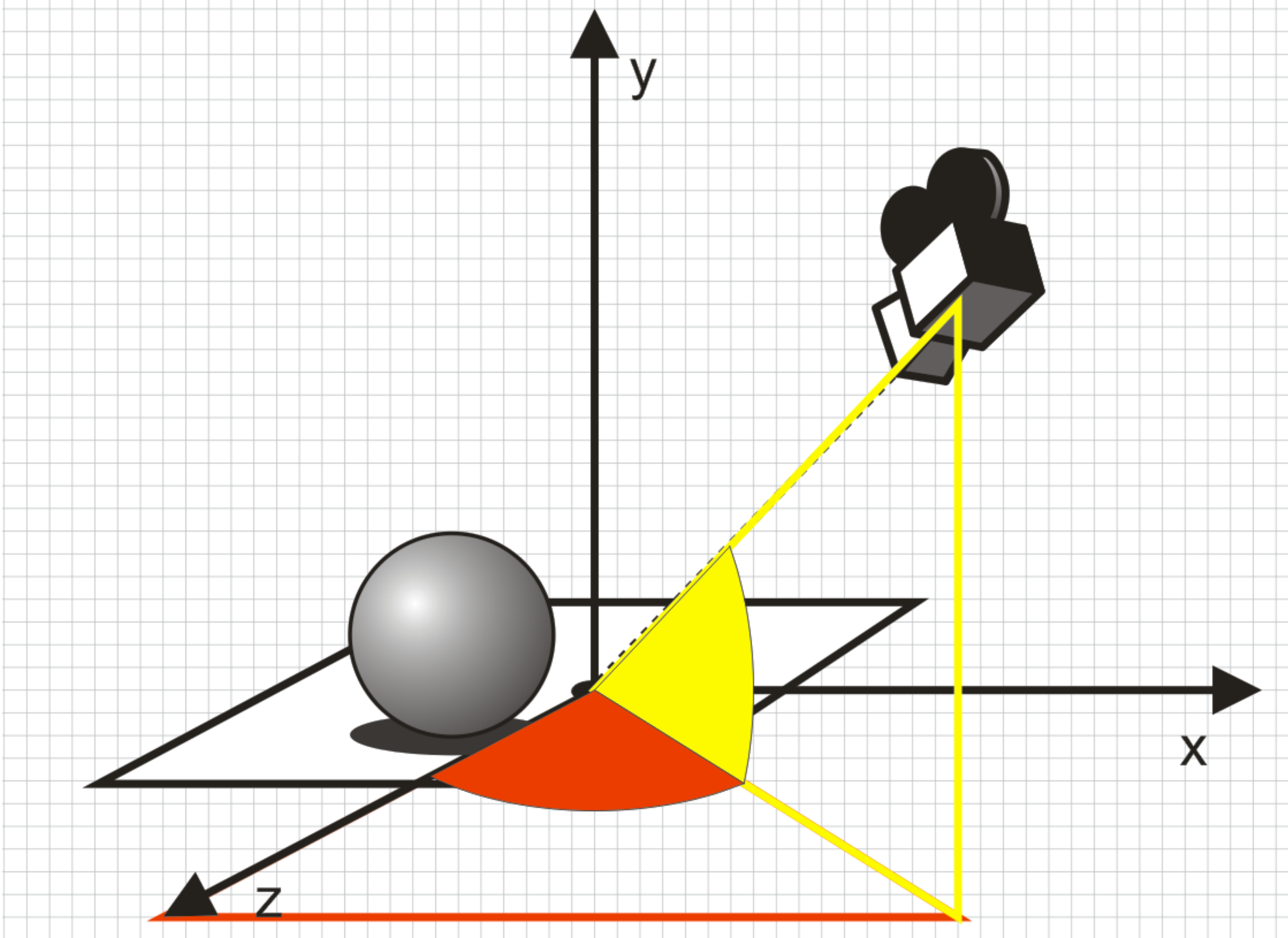
Stage 0



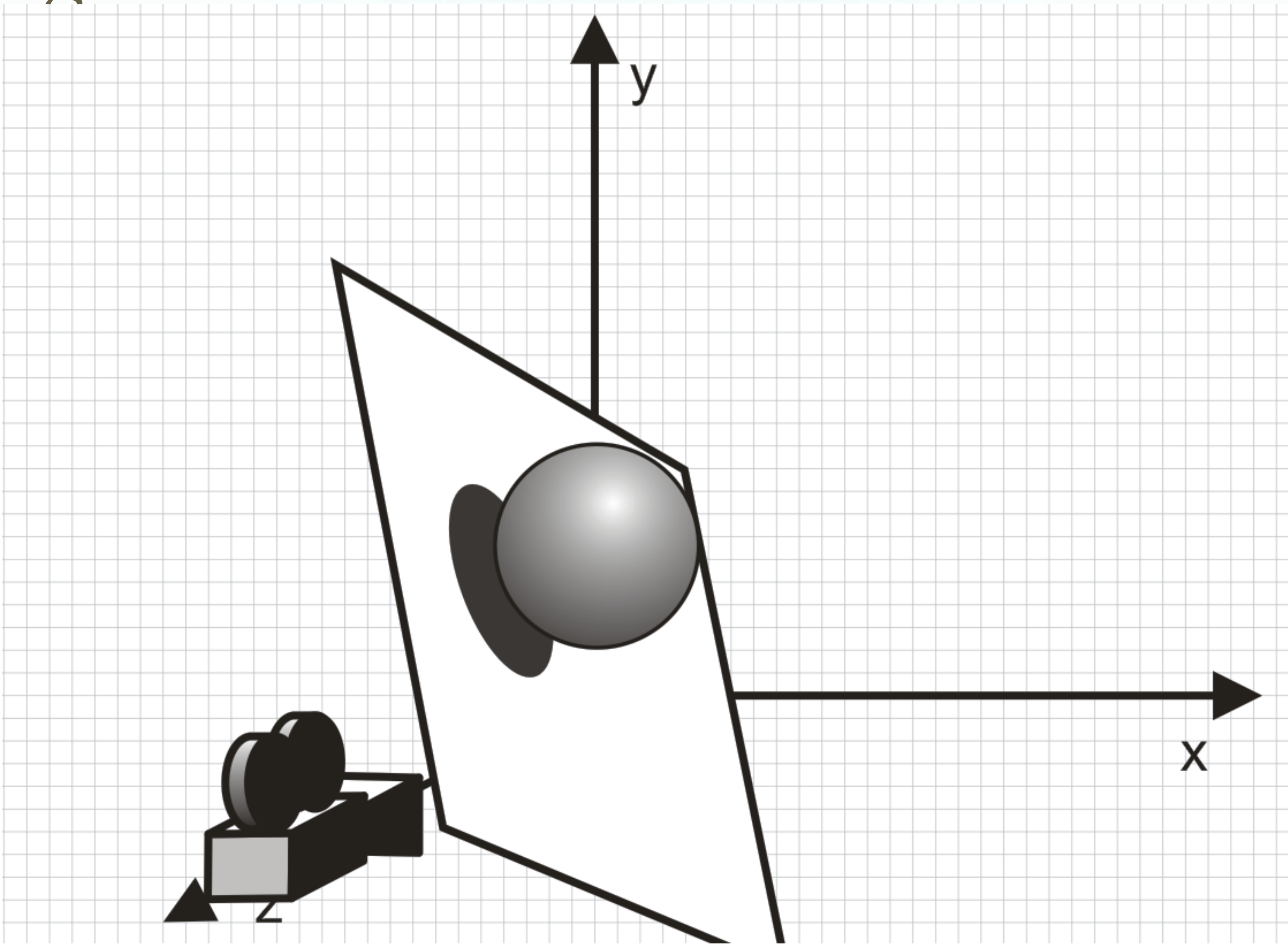
Stage 1 - translate P→P’



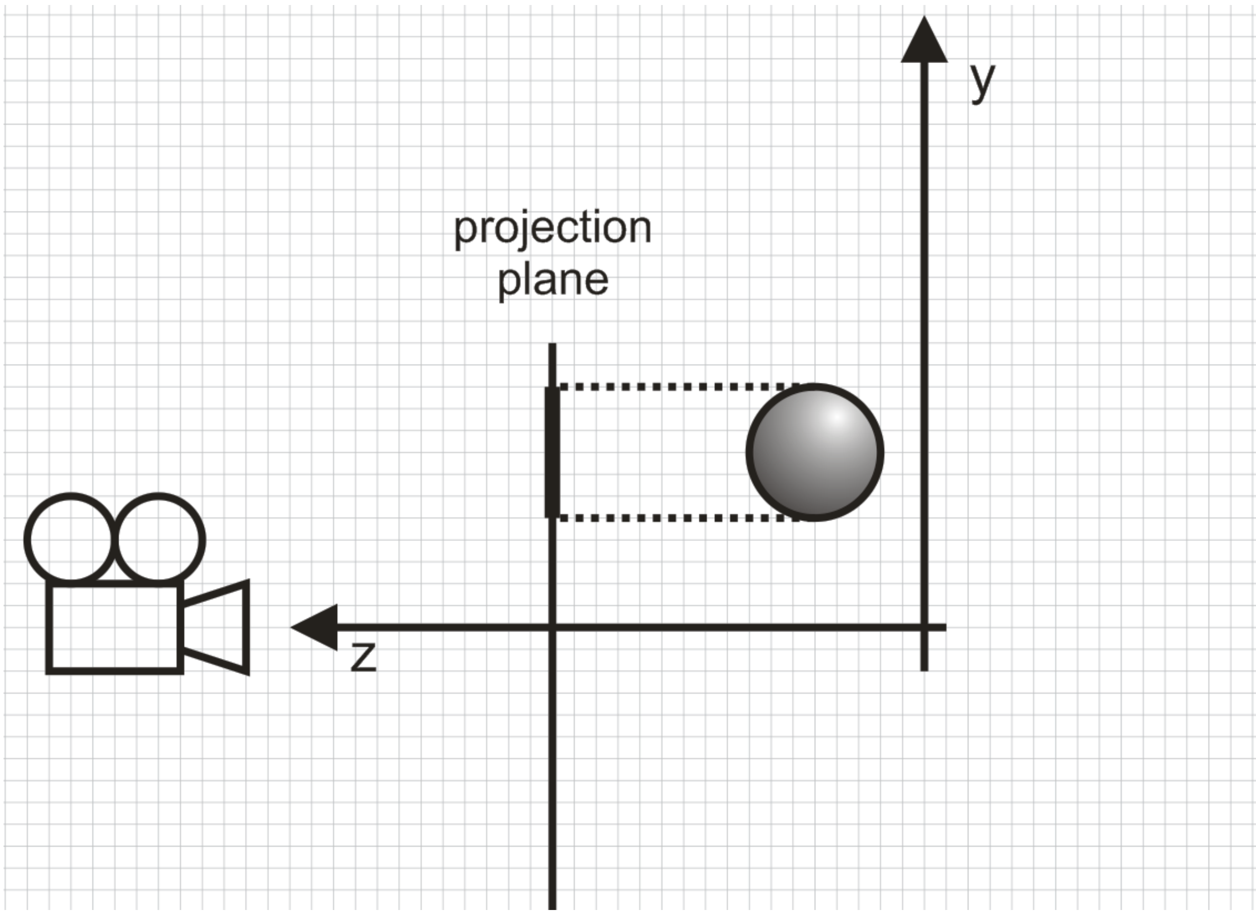
Stage 2 - rotate P’→P”→P’’’



Stage 2 - rotate P’→P”→P’’’



Orthogonal projection



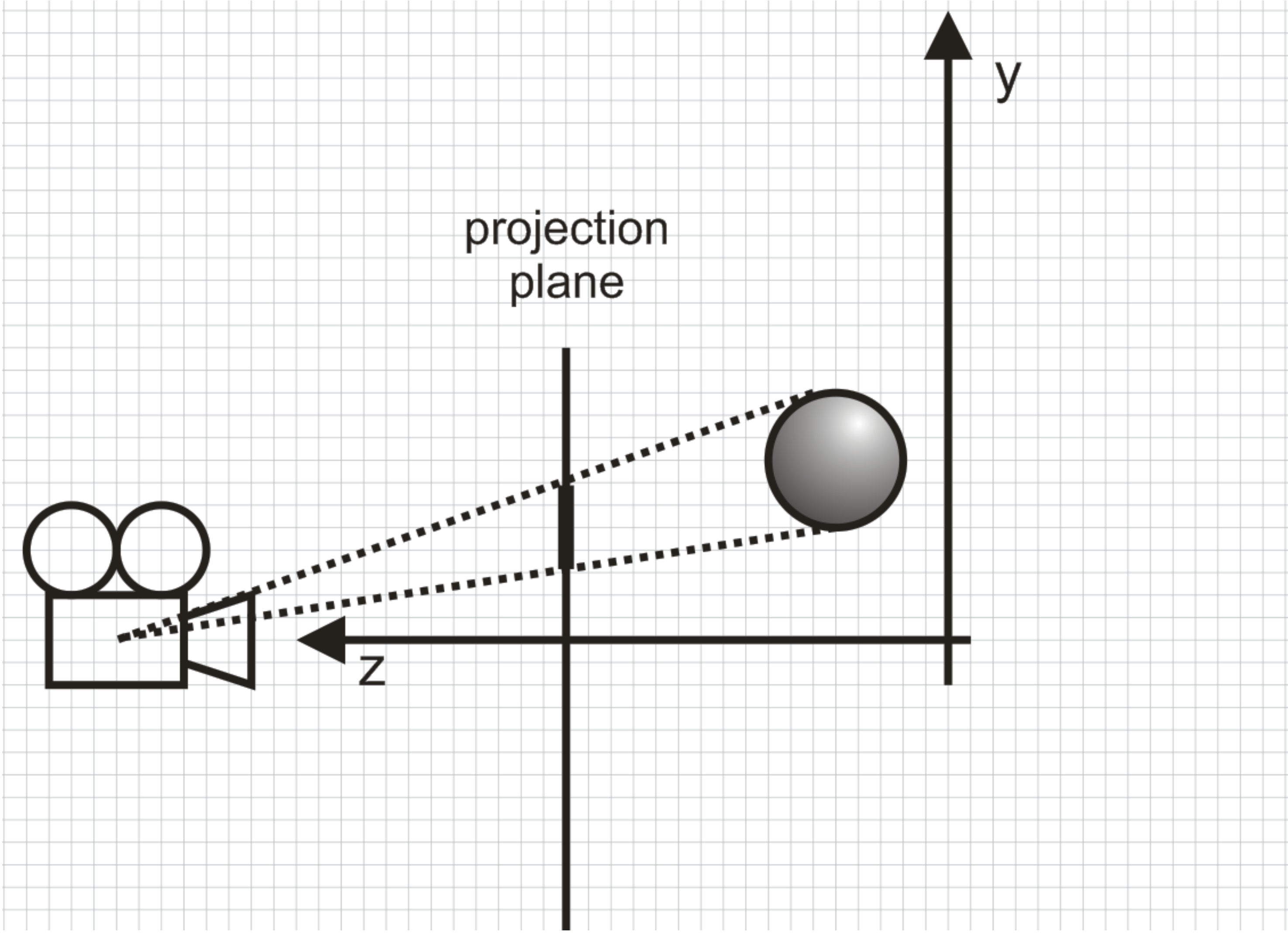
Orthogonal projection

- x_P = x'''
- y_P = y'''
- z''' is simply left out

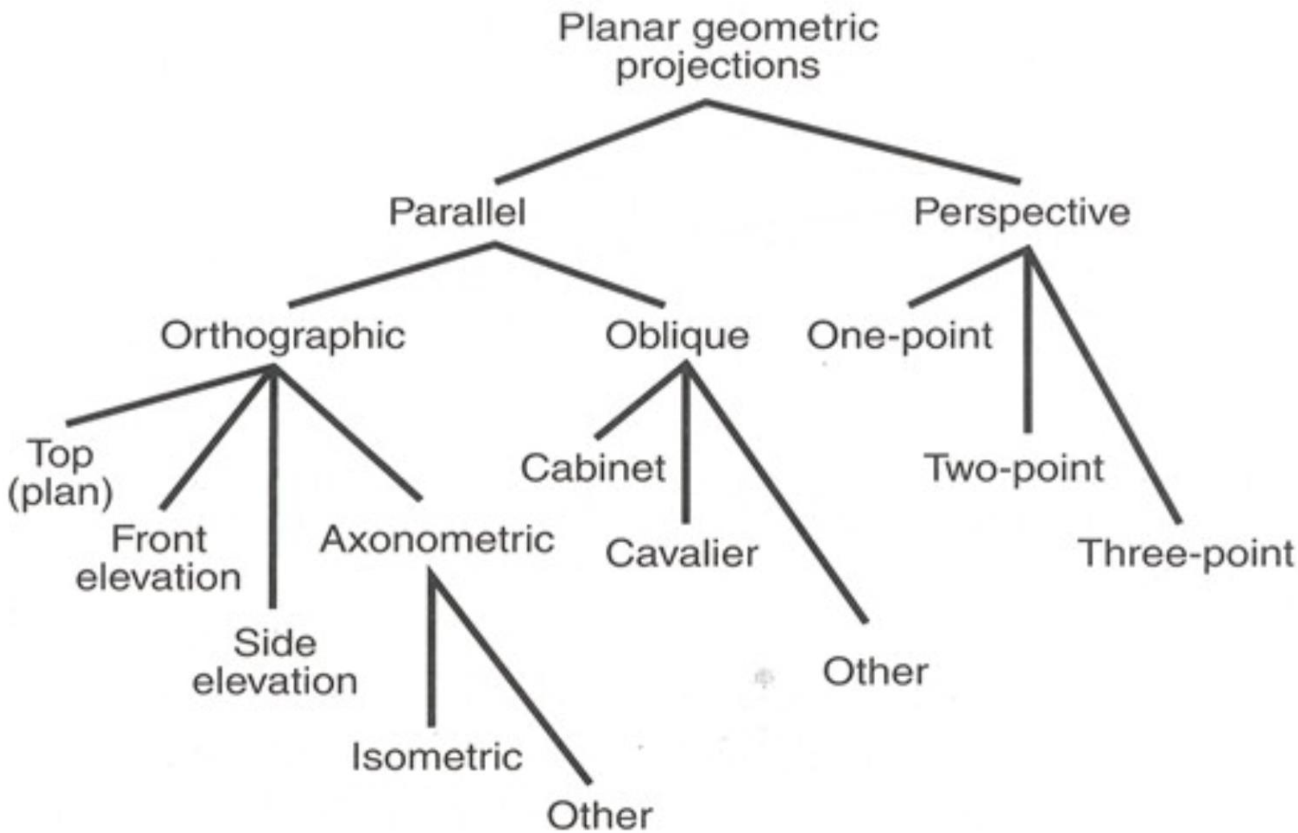
- Matrix notation

$$(x_P, y_P, z_p, 1) = (x''', y''', z''', 1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

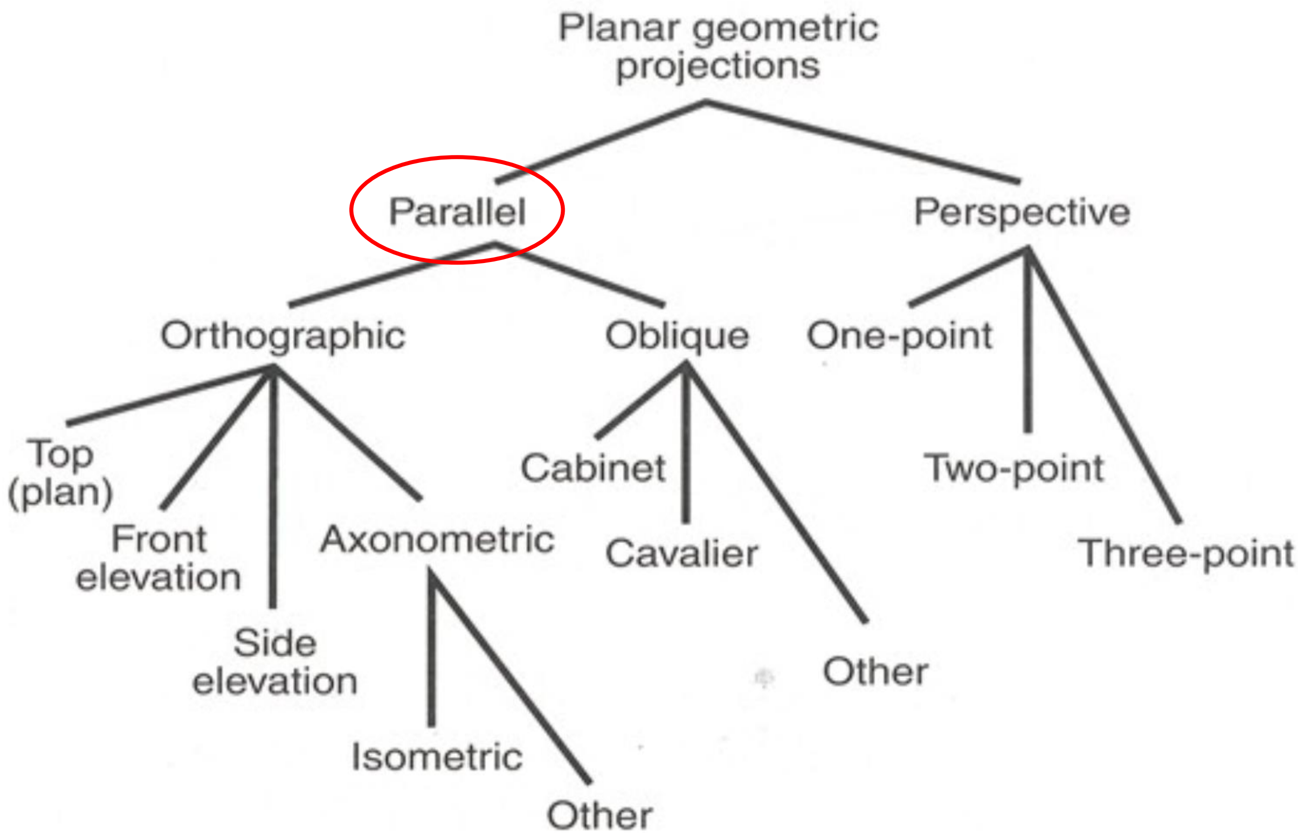
Perspective projection



Taxonomy of Projections

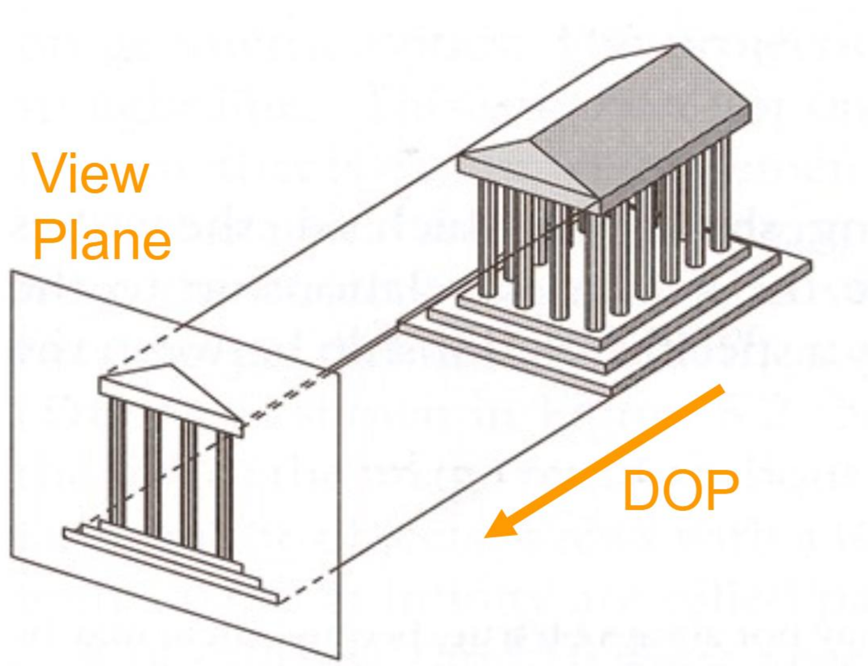


Taxonomy of Projections



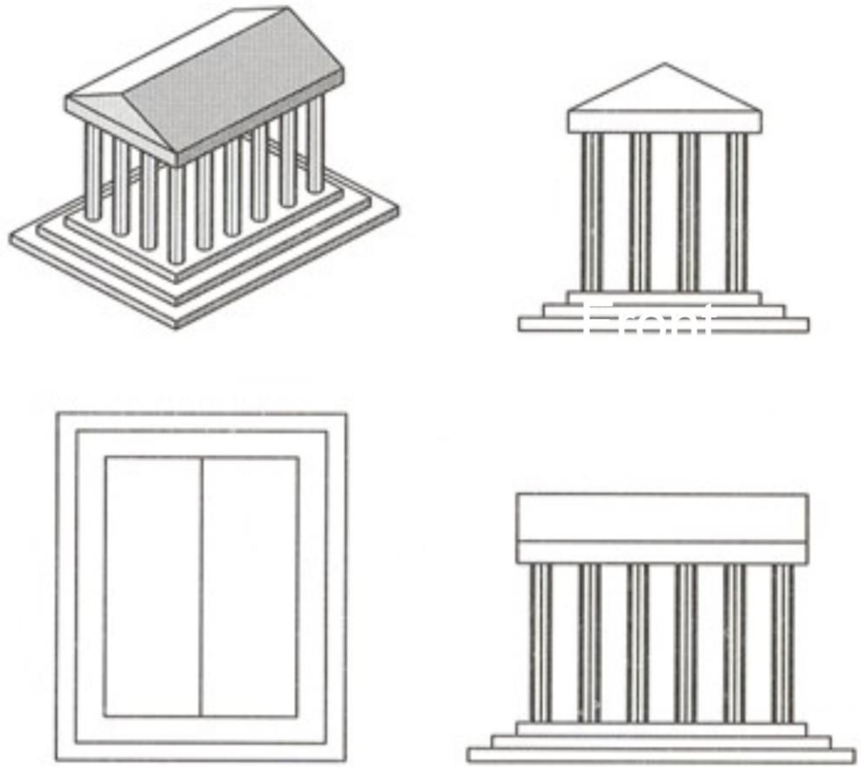
Parallel Projection

- Center of projection is at infinity
- Direction of projection (DOP) same for all points



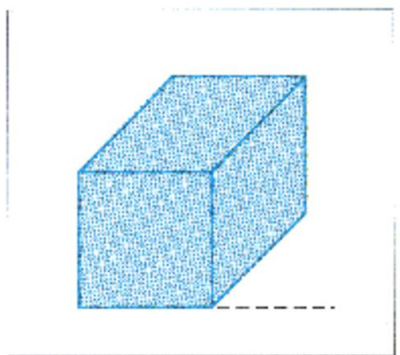
Orthographic Projections

- DOP perpendicular to view plane

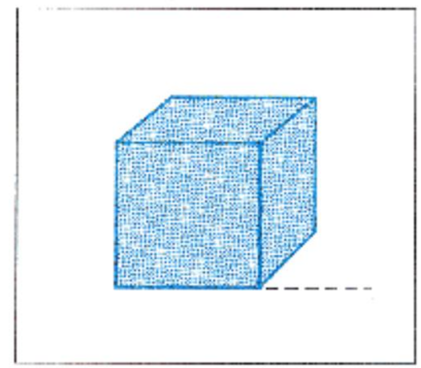


Oblique Projections

- DOP **not** perpendicular to view plane

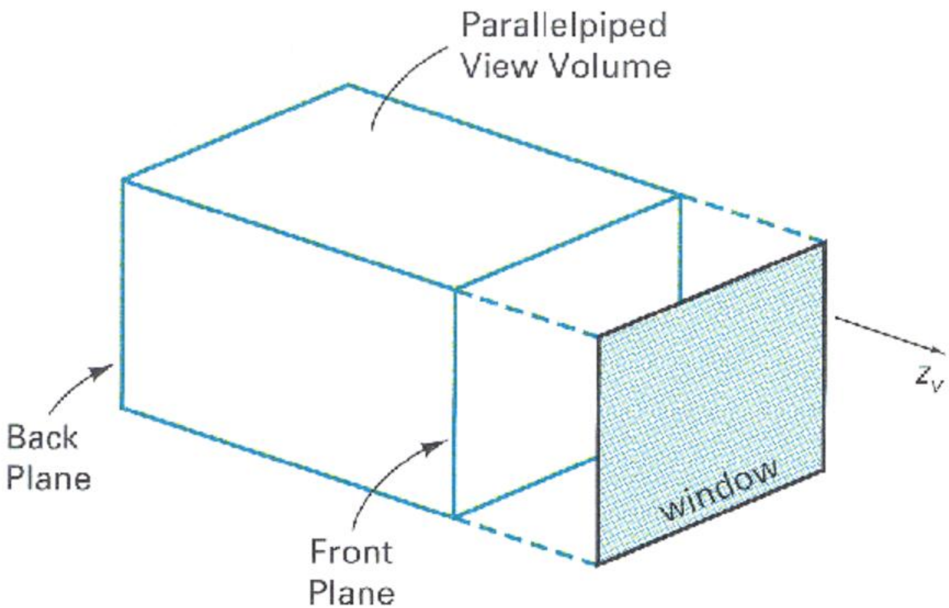


Cavalier
(DOP at 45°)



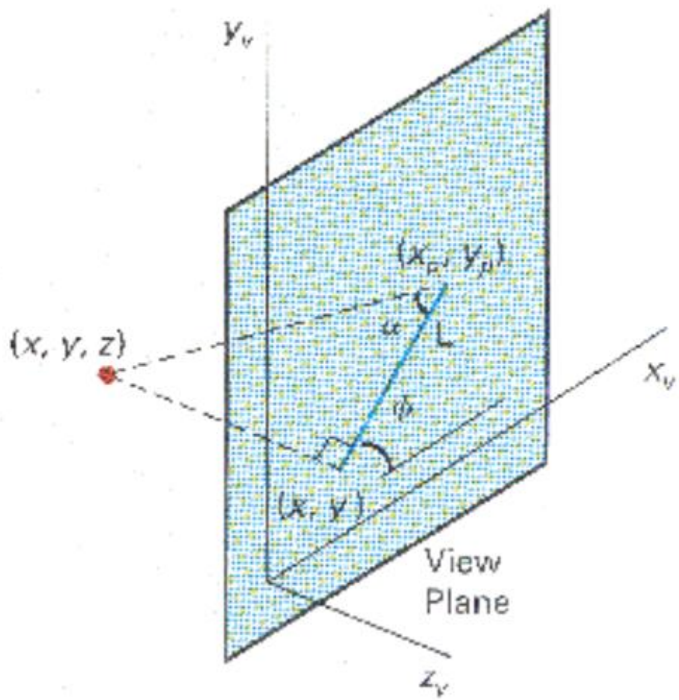
Cabinet
(DOP at 63.4°)

Parallel Projection View Volume



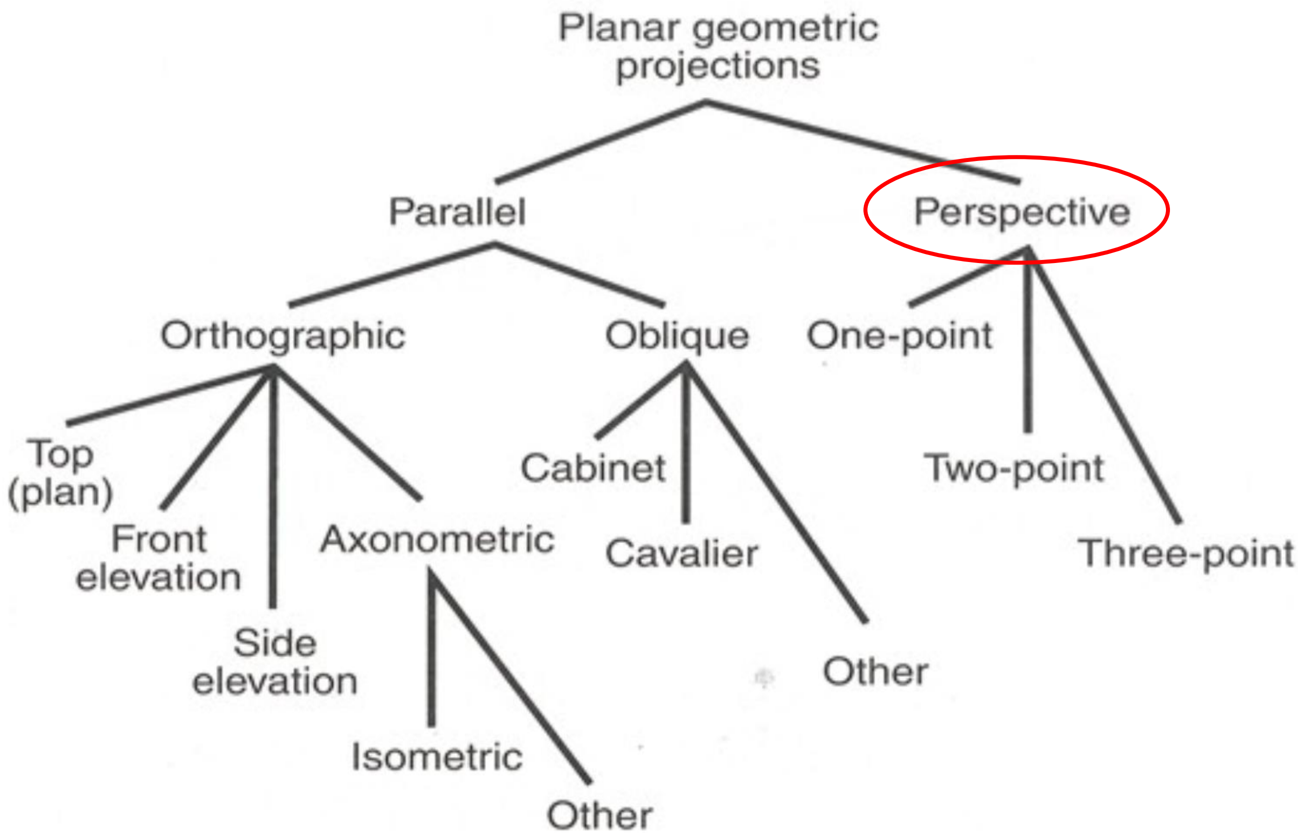
Parallel Projection Matrix

- General parallel projection transformation



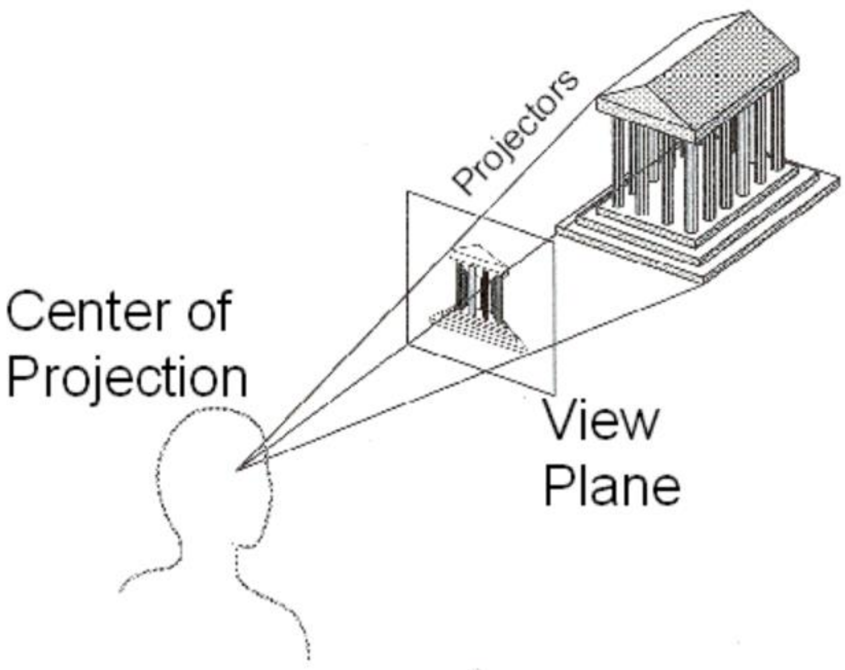
$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & L_1 \cos\phi & 0 \\ 0 & 1 & L_1 \sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Taxonomy of Projections



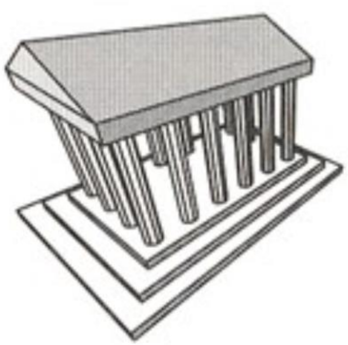
Perspective Projection

- Maps points onto a view plane along projectors emitting from center of projection (COP)



Perspective Projection

- N-point perspective
  - How many vanishing points?



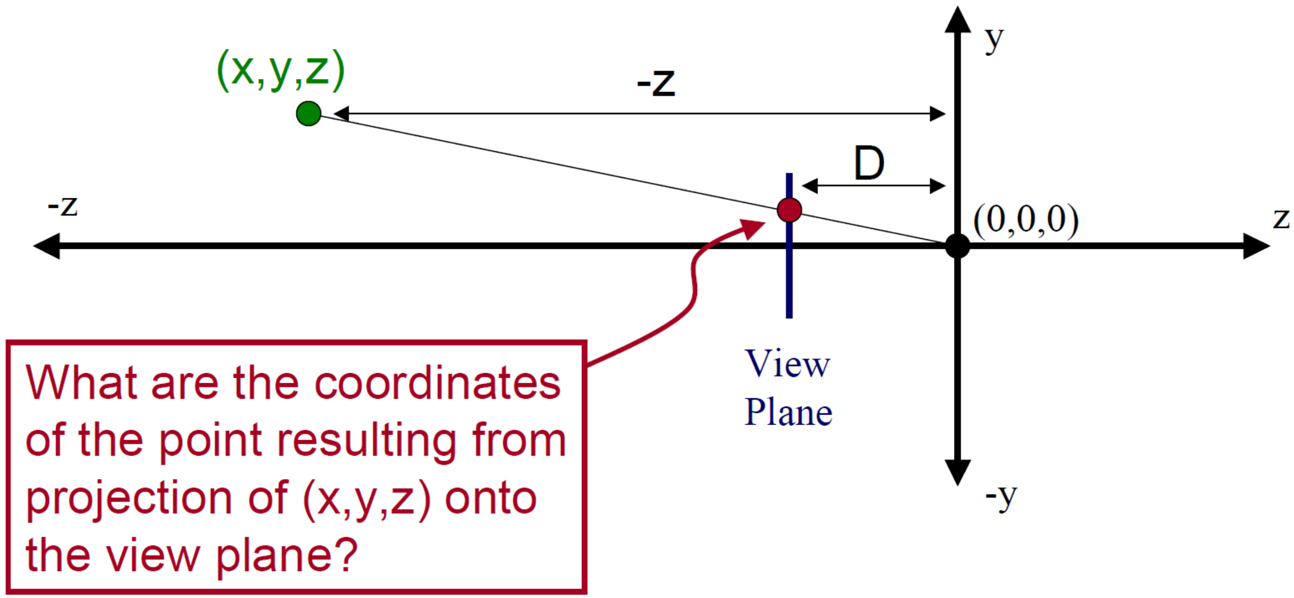
3-point perspective

2-point perspective

1-point perspective

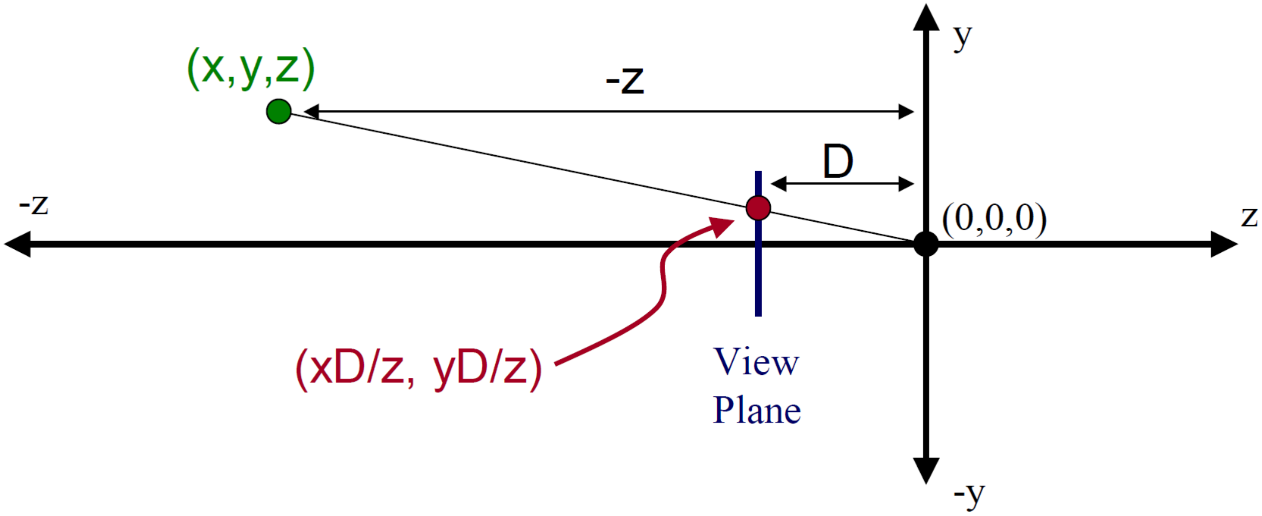
Perspective Projection

- Compute 2D coordinates from 3D coordinates using triangle similarity principle



Perspective Projection

- Compute 2D coordinates from 3D coordinates using triangle similarity principle



Perspective Projection

- 4x4 matrix representation
  - $x_S = x_C D / z_C$
  - $y_S = y_C D / z_C$
  - $z_S = D$
  - $w_S = 1$

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

- 4x4 matrix representation
  - $x_S = x_C D / z_C$
  - $y_S = y_C D / z_C$
  - $z_S = D$
  - $w_S = 1$

$x_S = x_C$

$y_S = y_C$

depth is stored $z_S = z_C$

$w_S = z_C / D$

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

- 4x4 matrix representation
  - $x_S = x_C D / z_C$ $\quad$ $x_S = x_C$
  - $y_S = y_C D / z_C$ $\quad$ $y_S = y_C$
  - $z_S = D$ $\quad$ $z_S = z_C$
  - $w_S = 1$ $\quad$ $w_S = z_C / D$

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

- 4x4 matrix representation
  - $x_S = x_C D / z_C$
  - $y_S = y_C D / z_C$
  - $z_S = D$
  - $w_S = 1$

$x_S = x_C$

$y_S = y_C$

$z_S = z_C$

$w_S = z_C / D$

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

- 4x4 matrix representation
  - $x_S = x_C D / z_C$ $\qquad$ $x_S = x_C$
  - $y_S = y_C D / z_C$ $\qquad$ $y_S = y_C$
  - $z_S = D$ $\qquad$ $z_S = z_C$
  - $w_S = 1$ $\qquad$ $w_S = z_C / D$

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

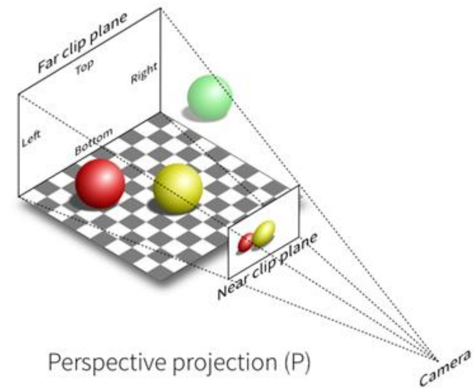
Perspective Projection

- 4x4 matrix representation
  - $x_S = x_C D/z_C$ — $x_S = x_C$
  - $y_S = y_C D/z_C$ — $y_S = y_C$
  - $z_S = D$ — $z_S = z_C$
  - $w_S = 1$ — $w_S = z_C/D$

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

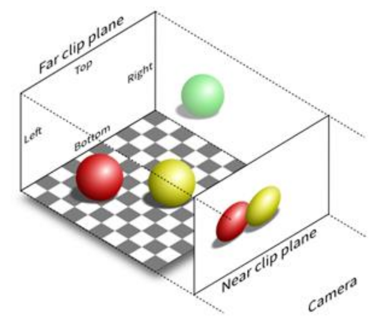
Perspective vs. Parallel

- Perspective Projection
  - + Size varies inversely with distance - looks realistic
  - - Distance and angles are not (in general) preserved
  - - Parallel lines do not (in general) remain parallel



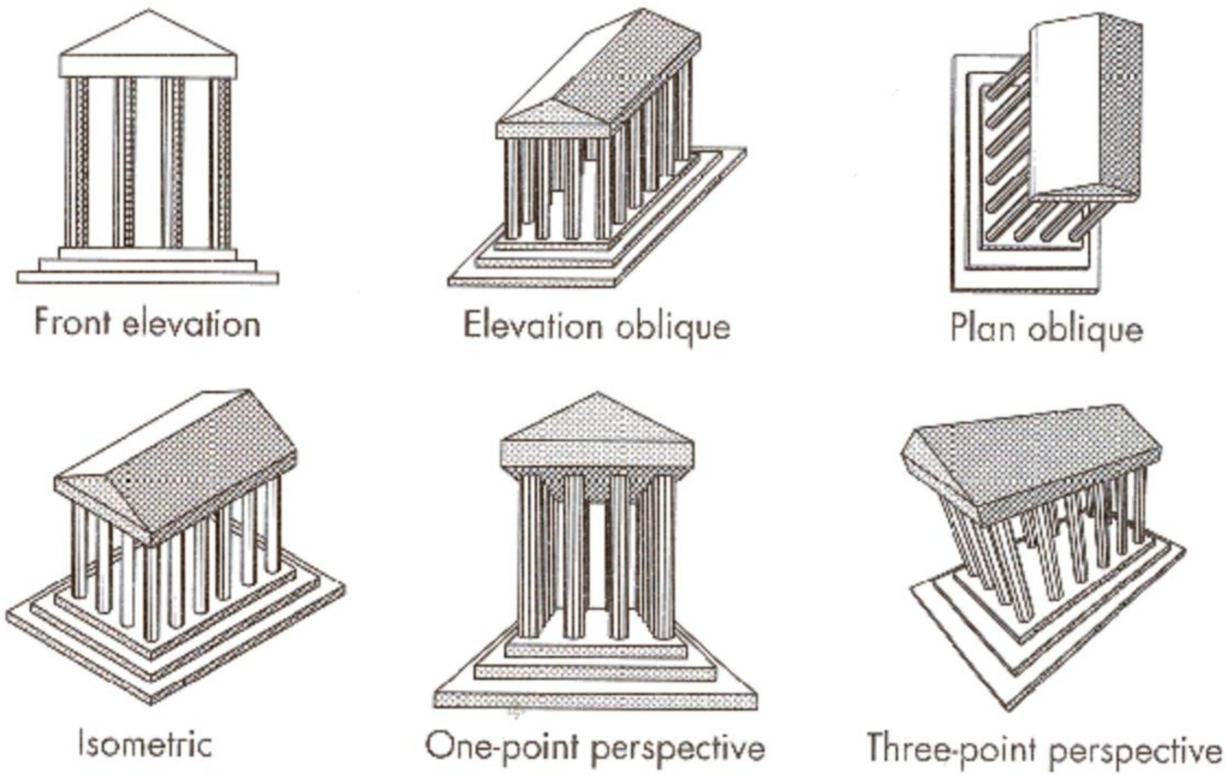
Perspective projection (P)

- Parallel Projection
  - + Good for exact measurements
  - + Parallel lines remain parallel
  - - Angles are not (in general) preserved
  - - Less realistic looking

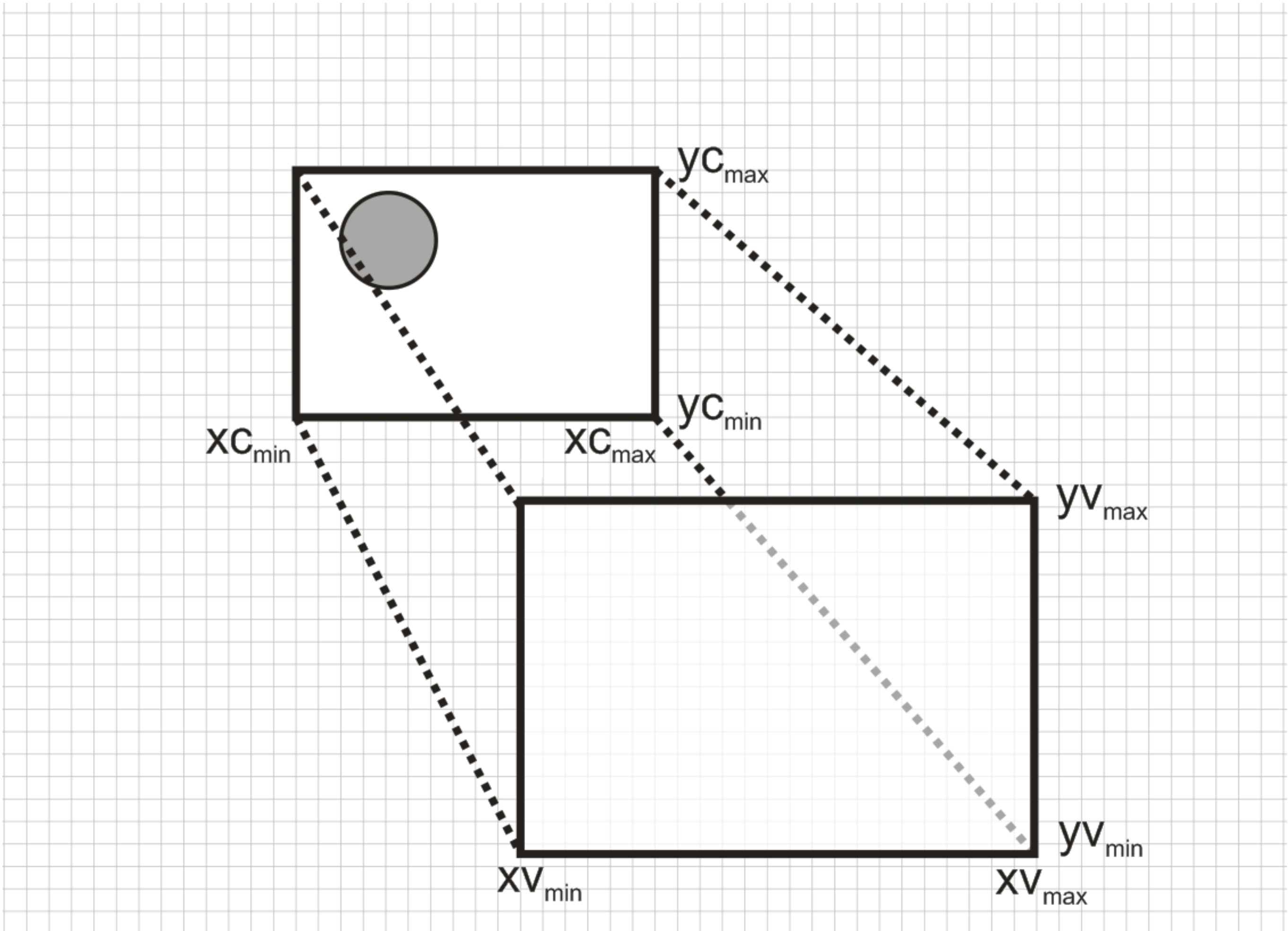


Orthographic projection (O)

Classical Projections



Viewport transformation



Viewport transformation

- s_x, s_y – scale factors

$$s_x = \frac{xv_{\max} - xv_{\min}}{xc_{\max} - xc_{\min}} \qquad s_y = \frac{yv_{\max} - yv_{\min}}{yc_{\max} - yc_{\min}}$$

- Matrix notation

$$(x_v, y_v, 1) = (x_p, y_p, 1)\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ -s_x xc_{\min} + xv_{\min} & -s_y yc_{\min} + yv_{\min} & 1 \end{pmatrix}$$

Welcome to the matrix!

1. local → global coordinates
   - translate, rotate, scale, translate
2. global → camera
   - translate, rotate, rotate, project
3. camera → viewport
   - translate, scale, translate

- Transformation combine = matrix multiply

3D rendering pipeline

3D polygons

Modeling Transformation
↓
Lighting
↓
Viewing Transformation
↓
Projection Transformation
↓
Clipping
↓
Scan Conversion

2D Image

Model transformation
    local → global / world coordinates
Viewport transformation
    global → camera
Projection transformation
    global → normalized device
Clipping
Rasterization
Texturing & Lighting

How the lectures should look like #2

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!

Next Lecture

Rasterization

Rendering Pipeline

Acknowledgements

- Thanks to all the people, whose work is shown here and whose slides were used as a material for creation of these slides:

Matej Novotný, GSVM lectures at FMFI UK

Peter Drahoš, PPGSO lectures at FIIT STU

Output of all the publications and great team work

Very best data from 3D cameras

Questions ?!

Skeletex
RESEARCH

www.skeletex.xyz

madaras@skeletex.xyz

martin.madaras@fmph.uniba.sk