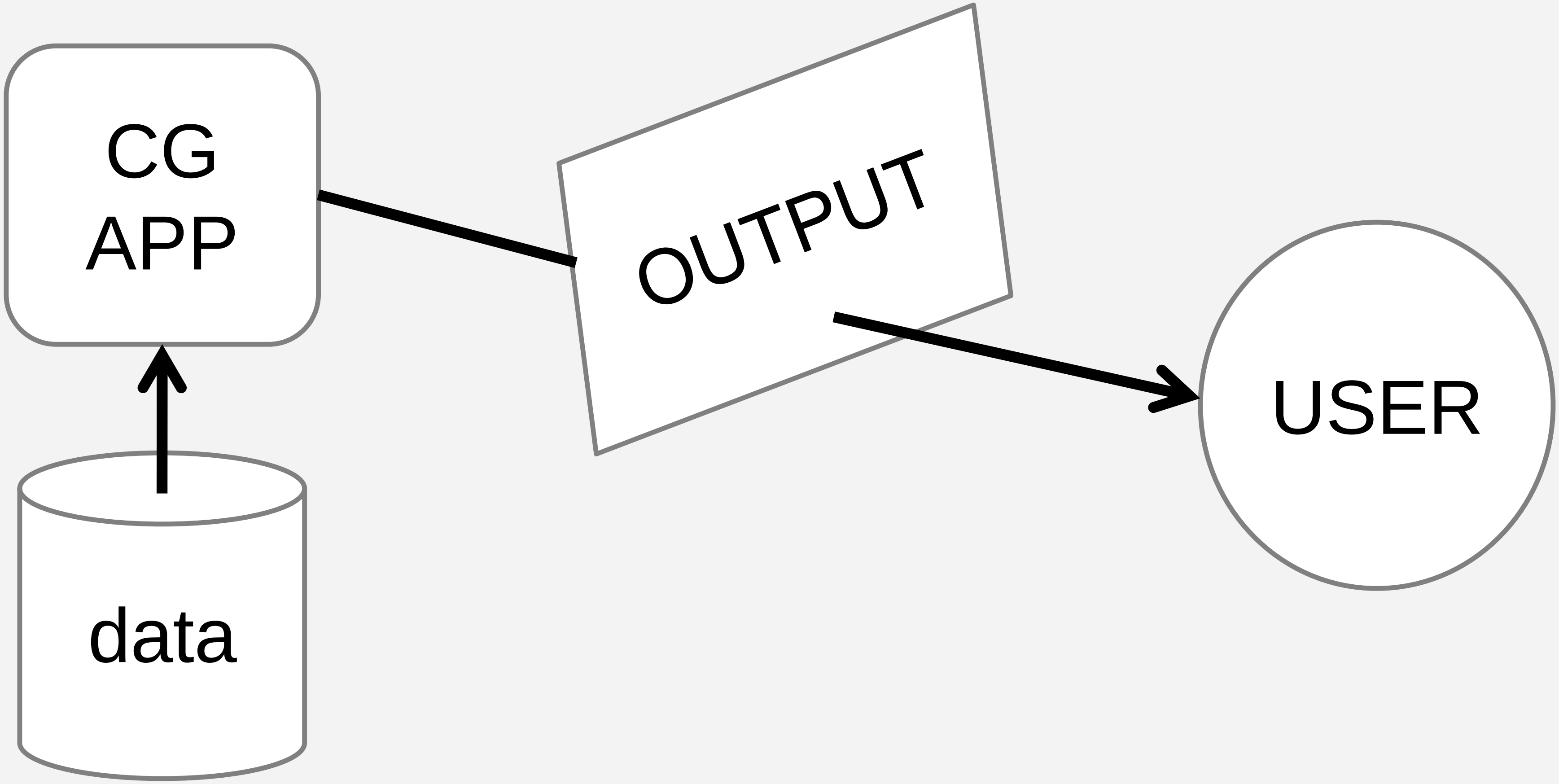
Graphical systems, visualization and multimedia

Computer graphics task

- Deliver images from computer to user



Example process

Program

3D model, 2D shape, animation, CT scan….

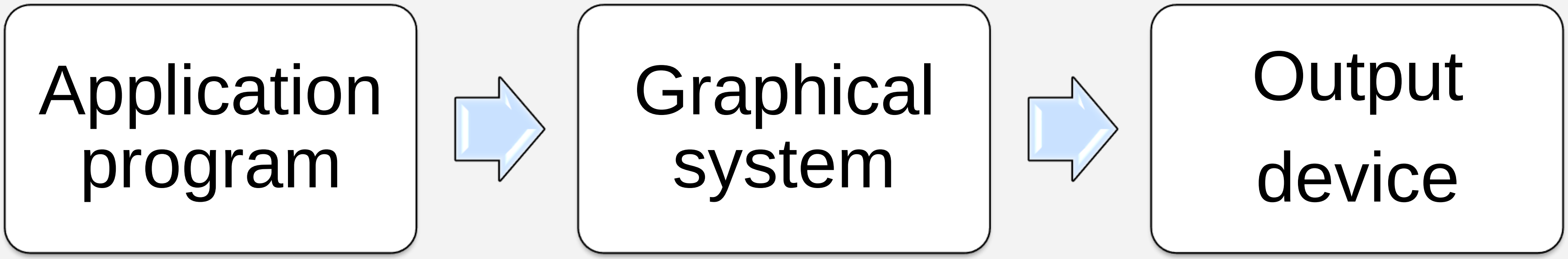
Monitor

Printer, projector, plotter, movie file, picture file, stereolitograph..

Platform

PC Win, PC Lin, Mac, SGI...
PS, XBOX, Wii, ...

CG reference model



- Inside the boxes – standards
- Between the boxes – standard interfaces
- Separate modeling and rendering
- Separate device-dependent and device-independent parts

Reference model – detailed

Application program

- Graphical data
  - – Models, textures, description, mapping…
- Animation
  - – Scripted, procedural (physics), interactive
- Application logic

Data sources

- Modeling, capturing, simulation…

Reference model – detailed

Graphical system

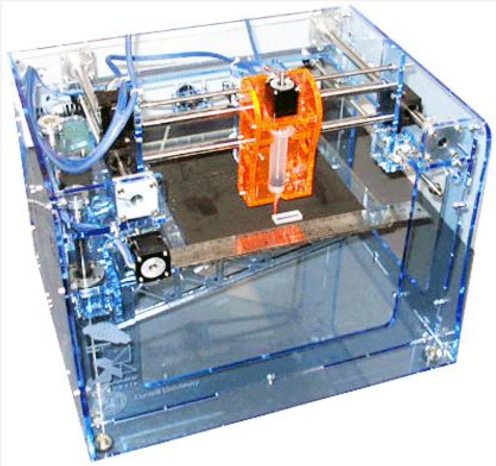
- Data processing (input, conversion)
- Transformations
- Projection
- Clipping, visibility, lighting
- Rasterization

Reference model – detailed

Output device

- Device driver
- Physical device
- Output format





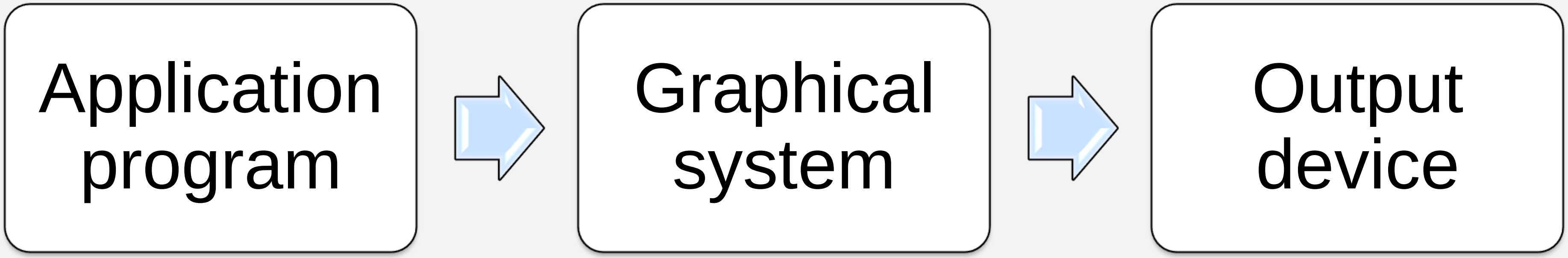
CGRM example

Application program → Graphical system → Output device

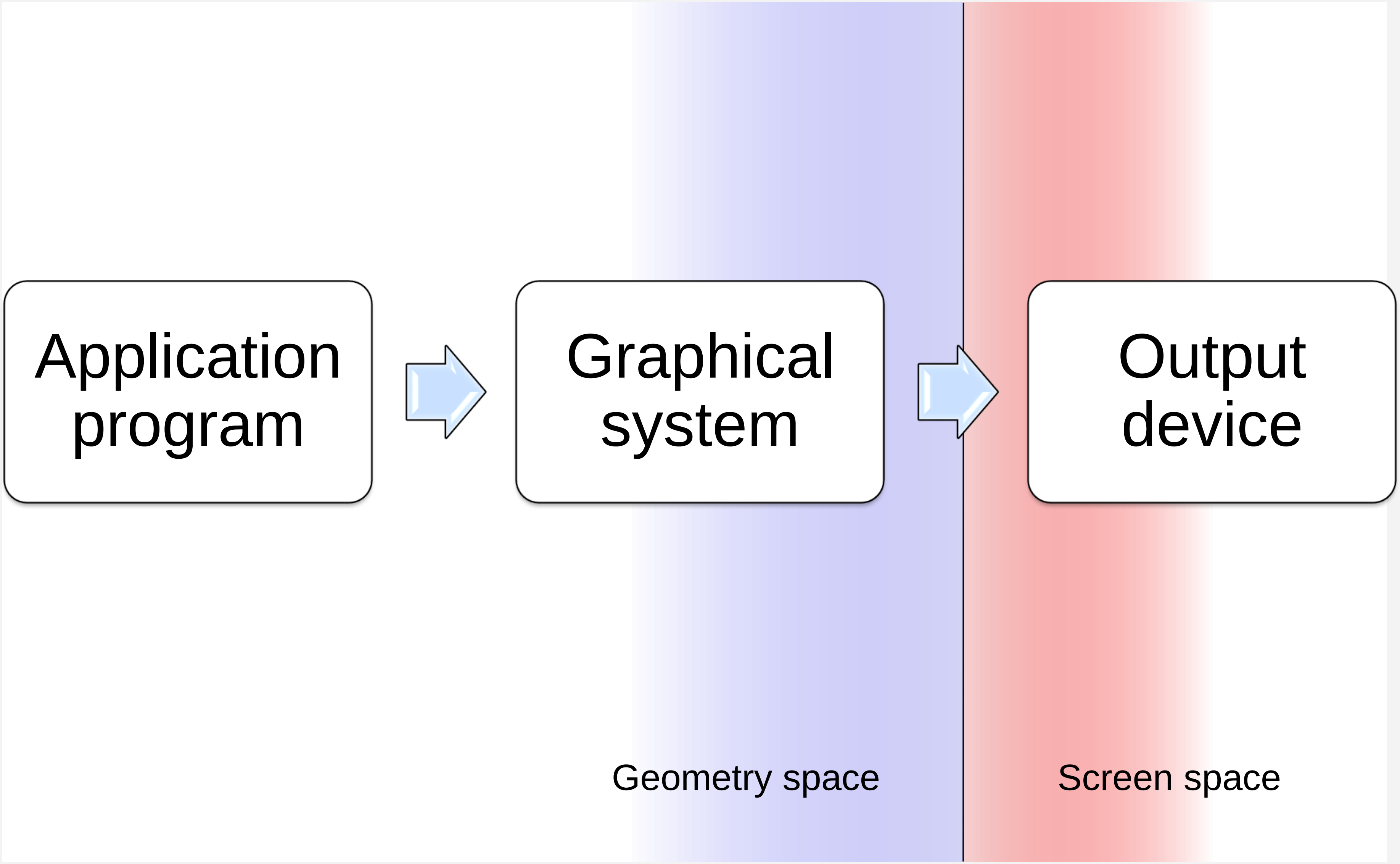
Advantages of CGRM

- Device-independent application development
- Application-independent device development
- Standard interface GS ↔ device
  - Hardware acceleration, optimization
- Standard interface APP ↔ GS
  - Rapid development, transferrable code
  - Translation from APP language to GS language

CG reference model



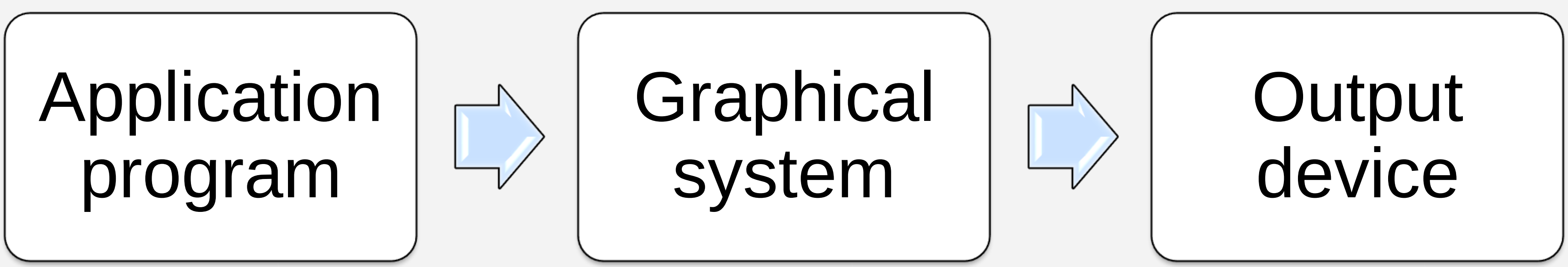
CG reference model



Graphical information and rendering

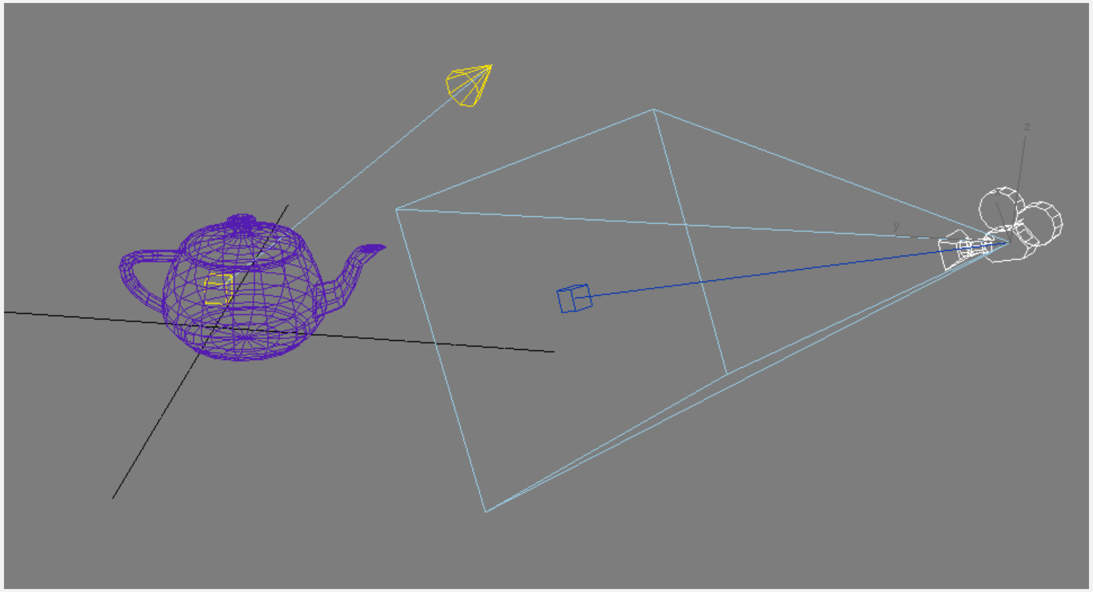
Our focus

- 3D objects in geometry space
  - some concepts explained in 2D, then extended
- Object representation (inside APP, GS)
- Object rendering (GS → Output device)



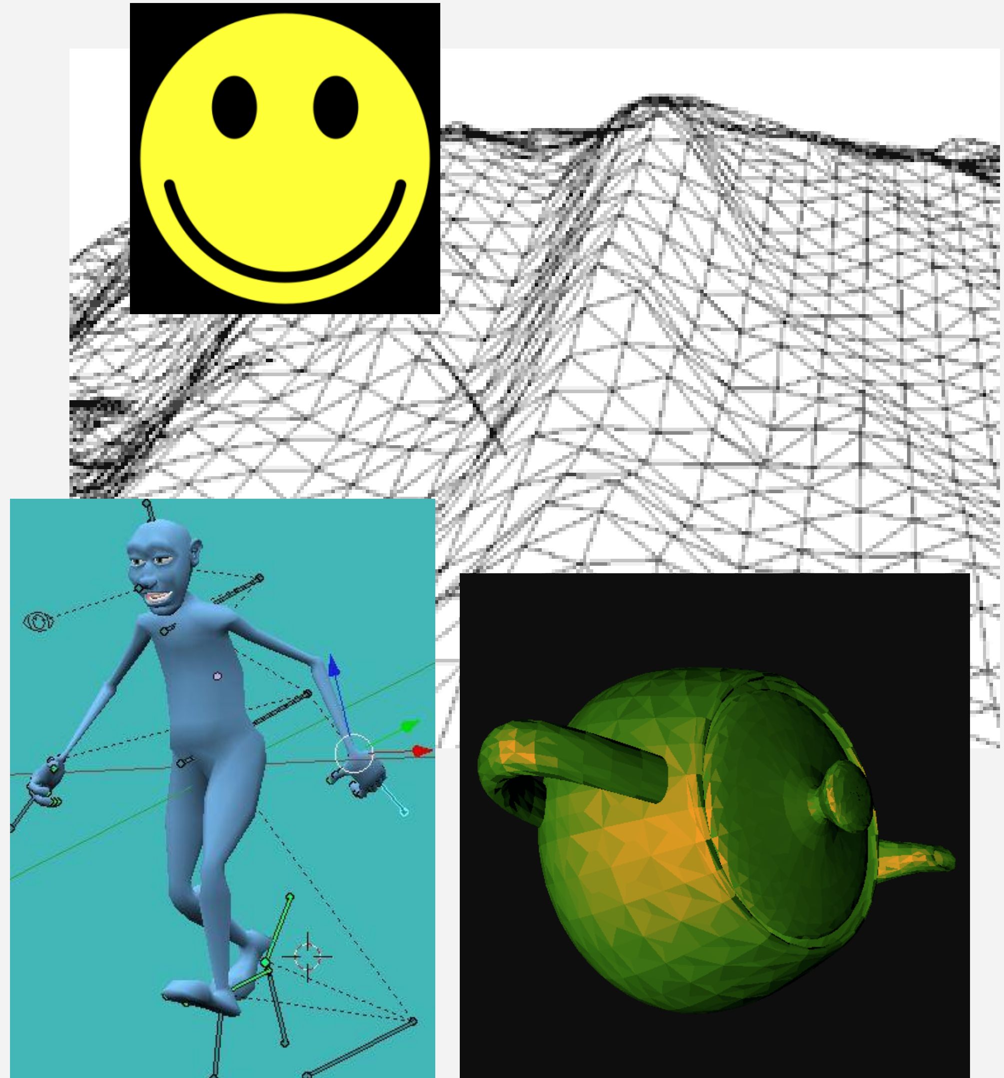
Geometry space

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



Dimensionality

- 2D
  - Shapes, images
- 2.5D
  - Surfaces, terrains
- **3D**
  - **Objects, scenes**
- 4D
  - Animation

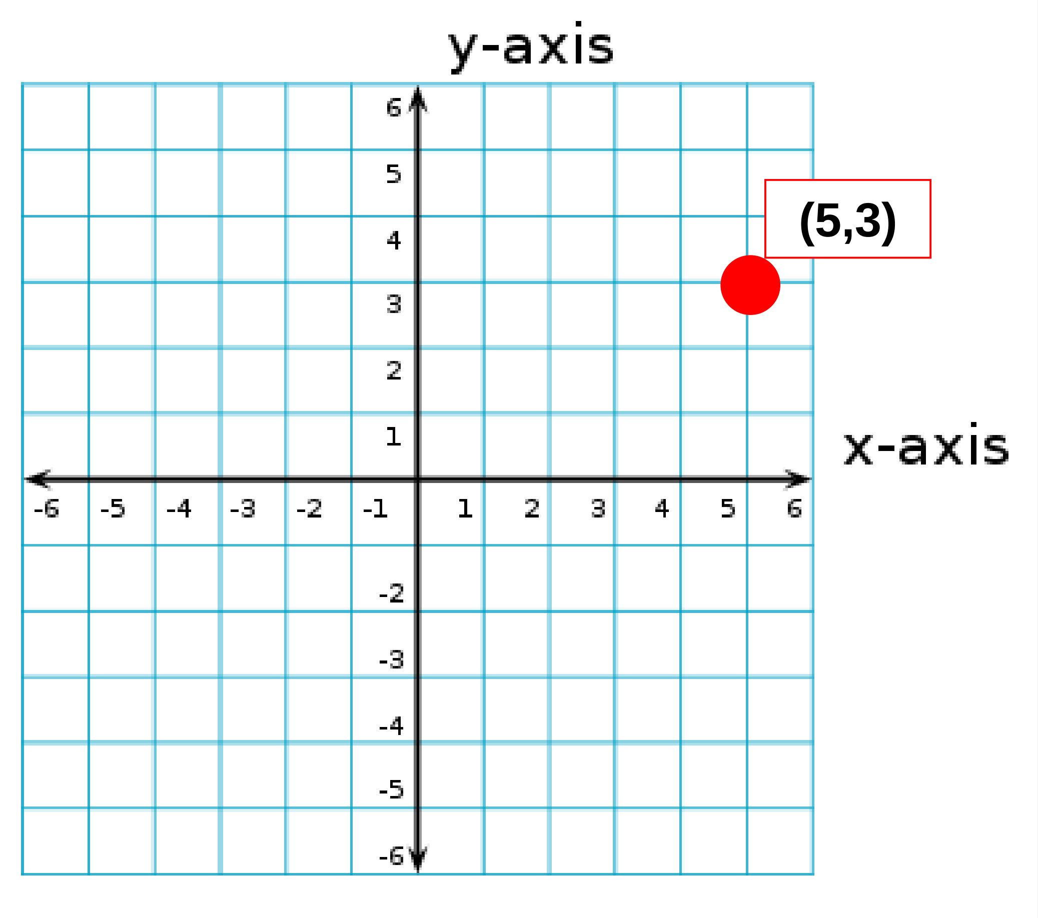


Full scene definition

- 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

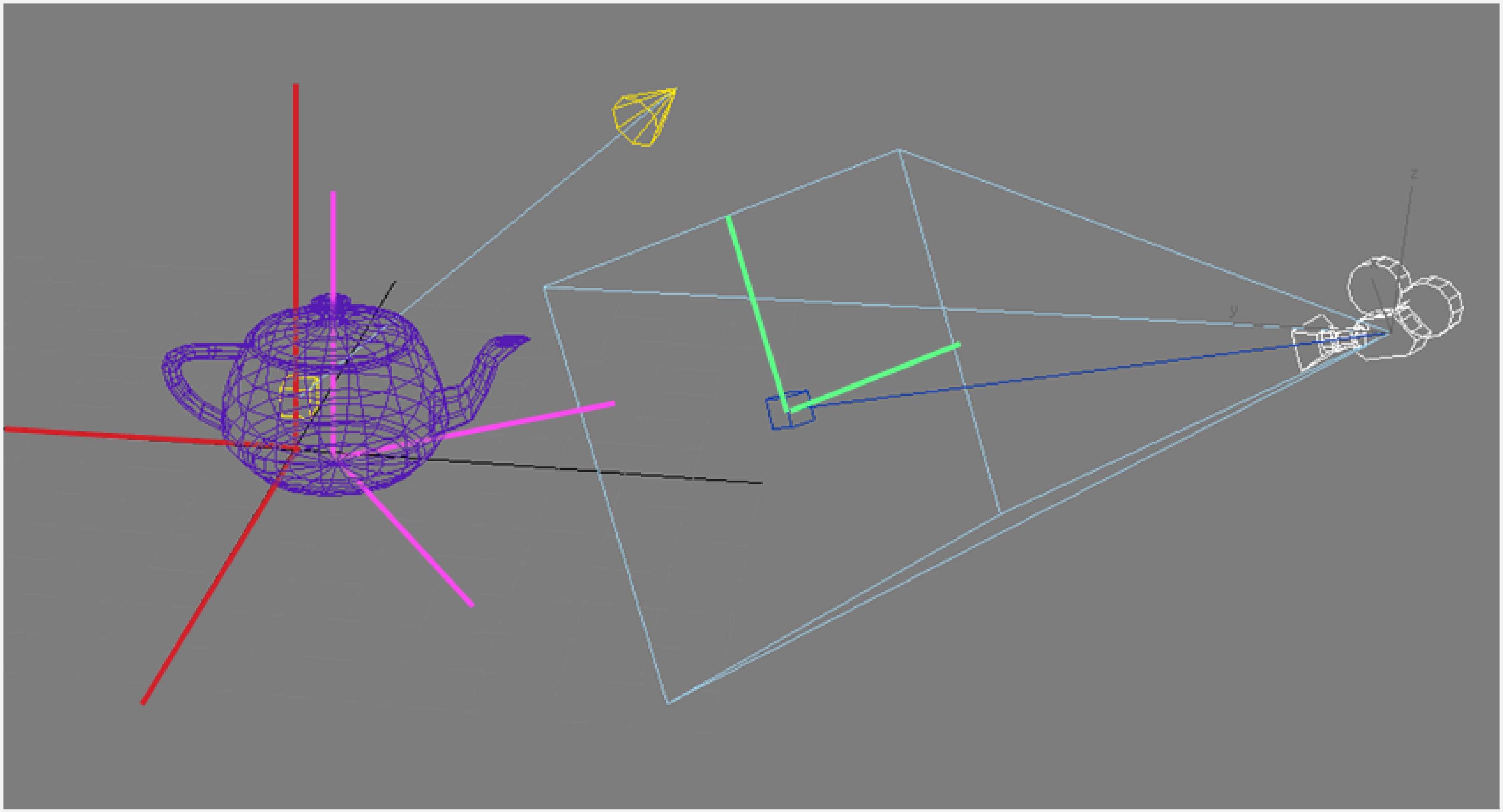
- Cartesian coordinates in 2D
  - Origin
  - x axis
  - y axis



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 geometry

Point

- Position in space
- Cartesian coordinates (x, y)

  (x, y, z)
- Homogeneous coordinates $(x, y, 1)$ $(x, y, z, 1)$
  - Subtraction of points
  - Translation
- Notation: **P**, **A**, …

Vector

- Direction in space
- Has no position
- Subtraction of 2 points
- Cartesian coordinates (x, y)

  (x, y, z)
- Homogeneous coordinates $(x, y, 0)$

  $(x, y, z, 0)$
- Notation: $\vec{u}, \vec{v}, \vec{n}$

Basic operations

- Addition

  Point + vector = point

  Vector + vector = vector

- Subtraction

  Point – point = vector

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

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

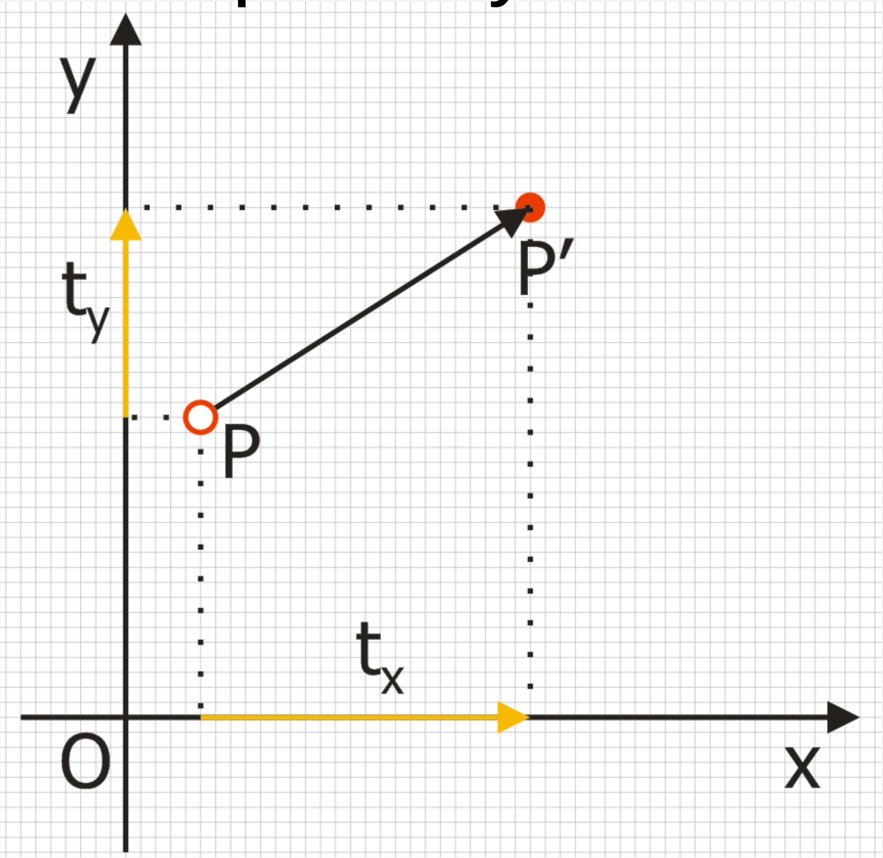
- Multiplication

  Multiplier * vector = vector

Transformations

Example: translation

- Move point by a vector



$$P(x, y) + \vec{v}(t_x, t_y) = P'(x + t_x, y + t_y)$$

Transformation matrix

- Unified way of performing transformations in 3D/2D spaces
- Translation, rotation, scaling, projections…
- GPUs are optimized for matrix operations
- Applying a transformation
  = Matrix multiplication

Transformations – translate

$\mathbf{P(x,y)} \rightarrow \mathbf{P'(x',y')}$

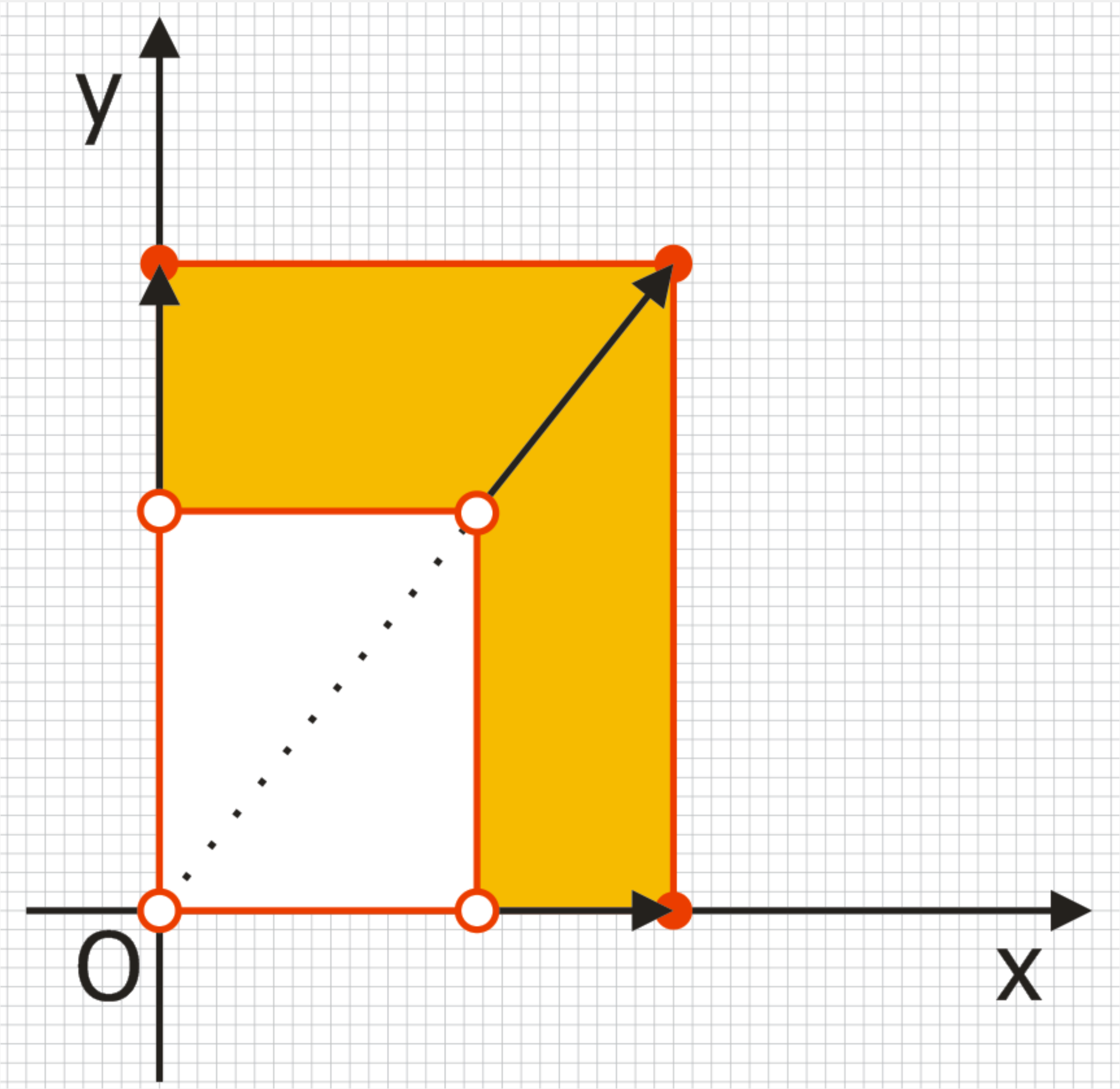
$\mathbf{x' = x + t_x}$

$\mathbf{y' = y + t_y}$

Matrix notation:

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

Transformations – scale



factor s

Transformations – scale

$P(x,y) \rightarrow P'(x',y')$

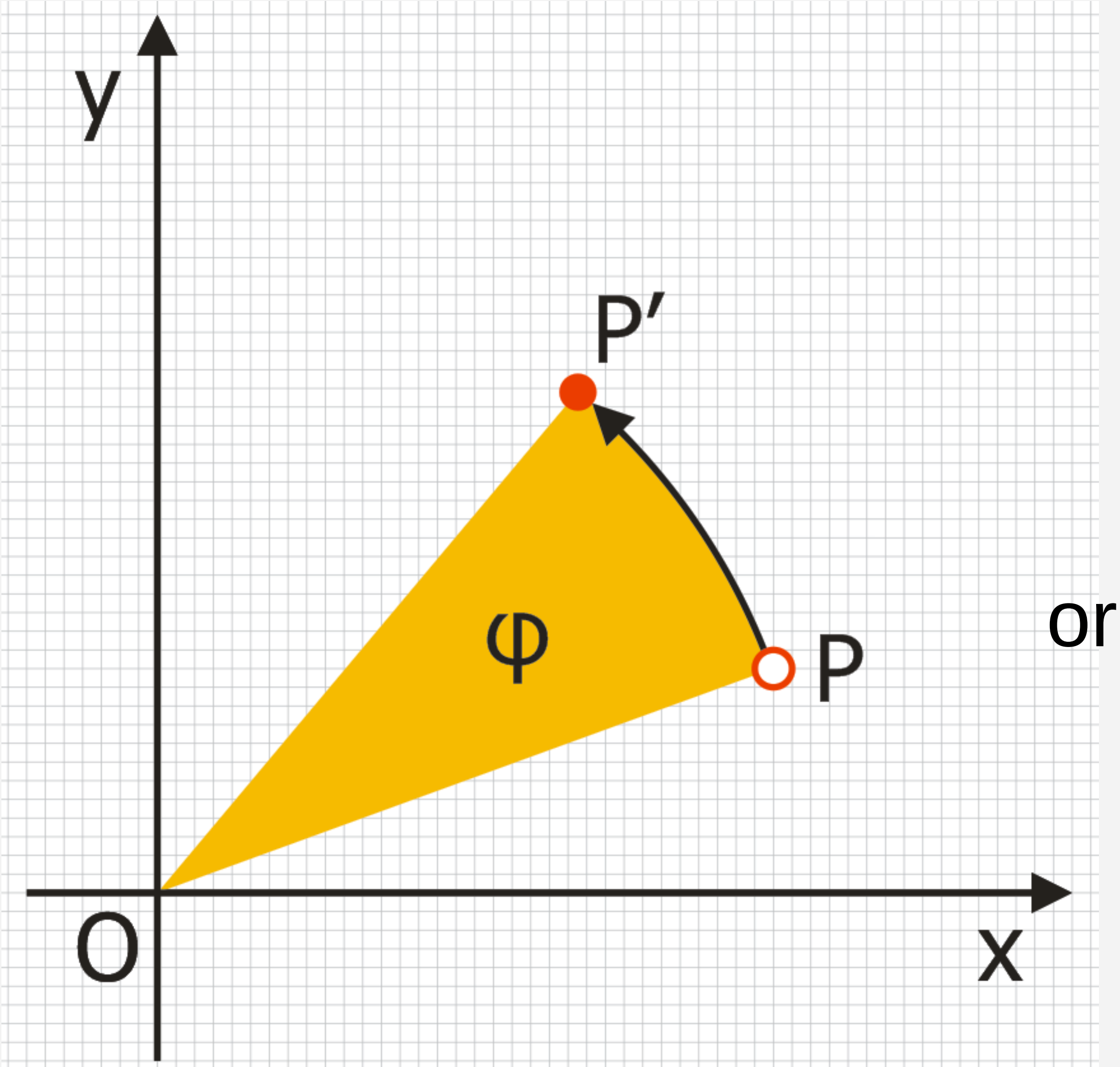
$x' = x.s_x$

$y' = y.s_y$

Matrix notation:

$$(x', y', 1) = (x, y, 1)\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Transformations – rotate



angle φ
‹0..360°›
‹0..2π›

Angle orientation!

Transformations – rotate

P(x,y) → P′(x′,y′)

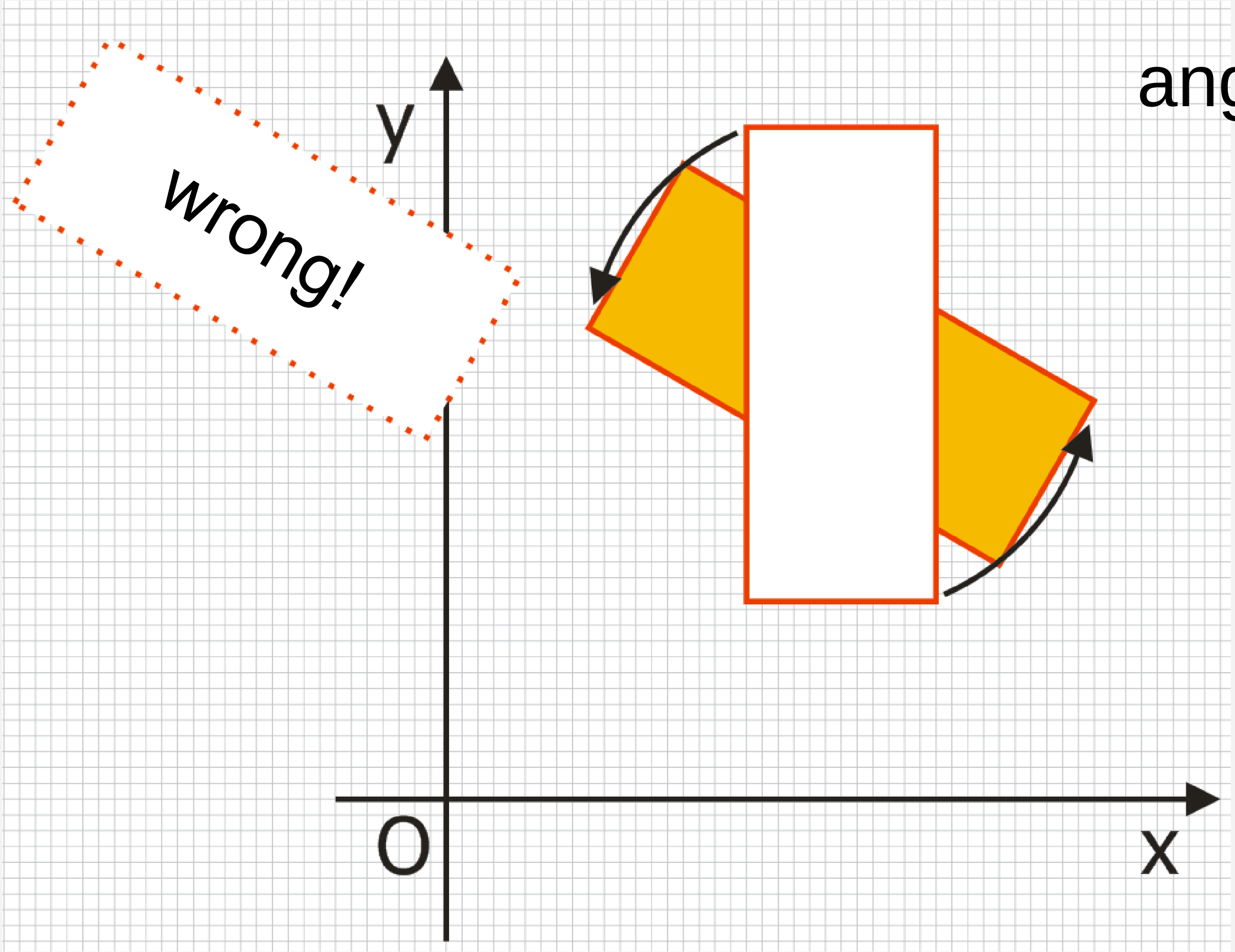
x′ = x.cos φ - y.sin φ

y′ = y.cos φ + x.sin φ

Matrix notation:

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

Problem: local rotation



angle φ

Transformation 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:

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

Transformation composition

- Matrix multiplication is associative

  A.B.C = (A.B).C = A.(B.C)

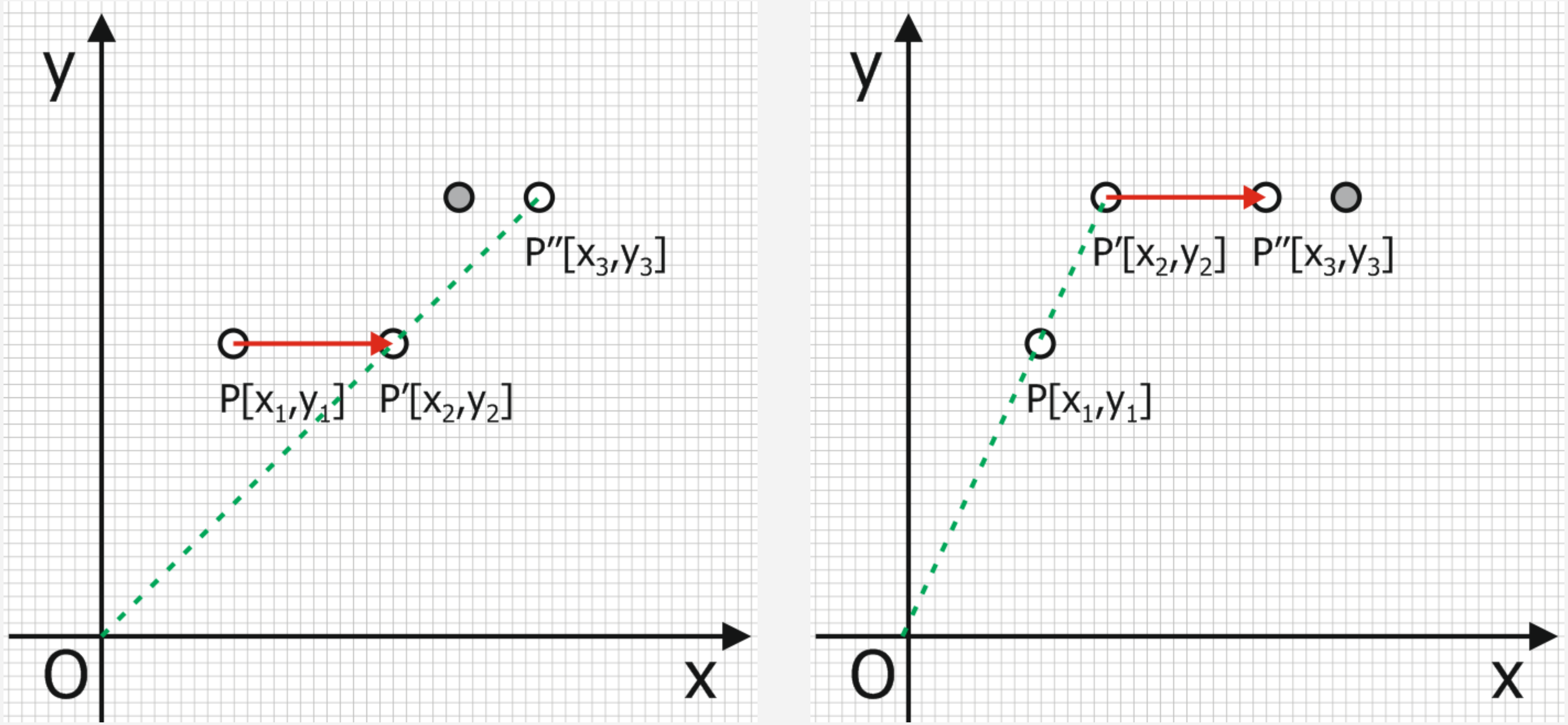
- Combined transformations can be re-used

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

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

Transformation order

- Matrix multiplication is not commutative
  - Order of transformations plays role



3D transformations

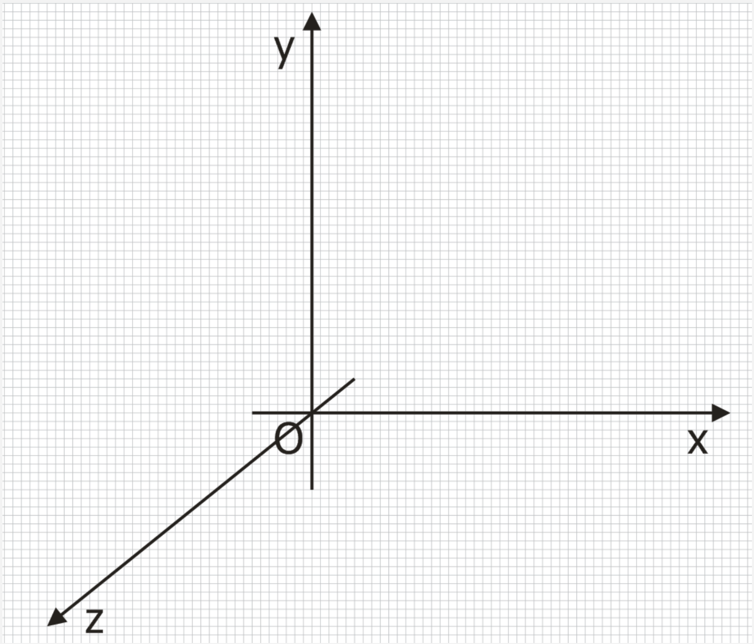
- scale $\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ translate $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix}$

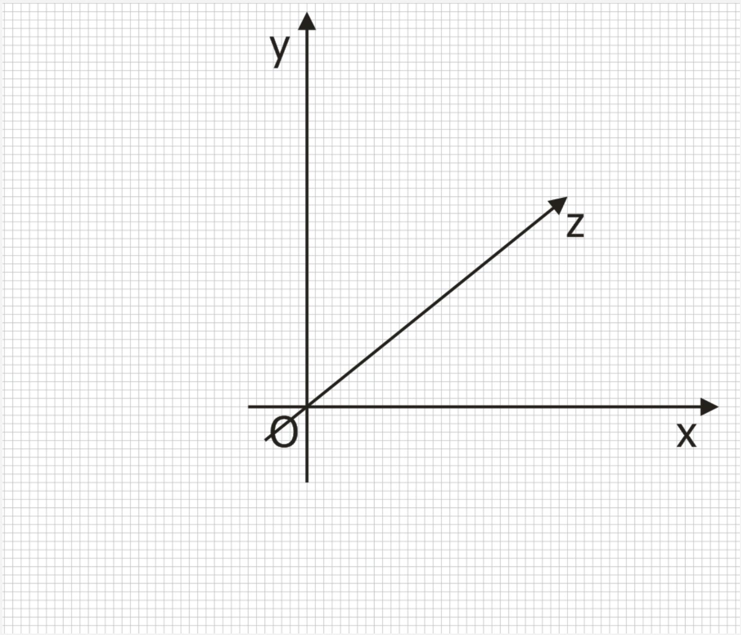
- rotate

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi_x & -\sin\varphi_x & 0 \\ 0 & \sin\varphi_x & \cos\varphi_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi_y & 0 & \sin\varphi_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi_y & 0 & \cos\varphi_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi_z & \sin\varphi_z & 0 & 0 \\ -\sin\varphi_z & \cos\varphi_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D coordinate systems

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





- rotation direction

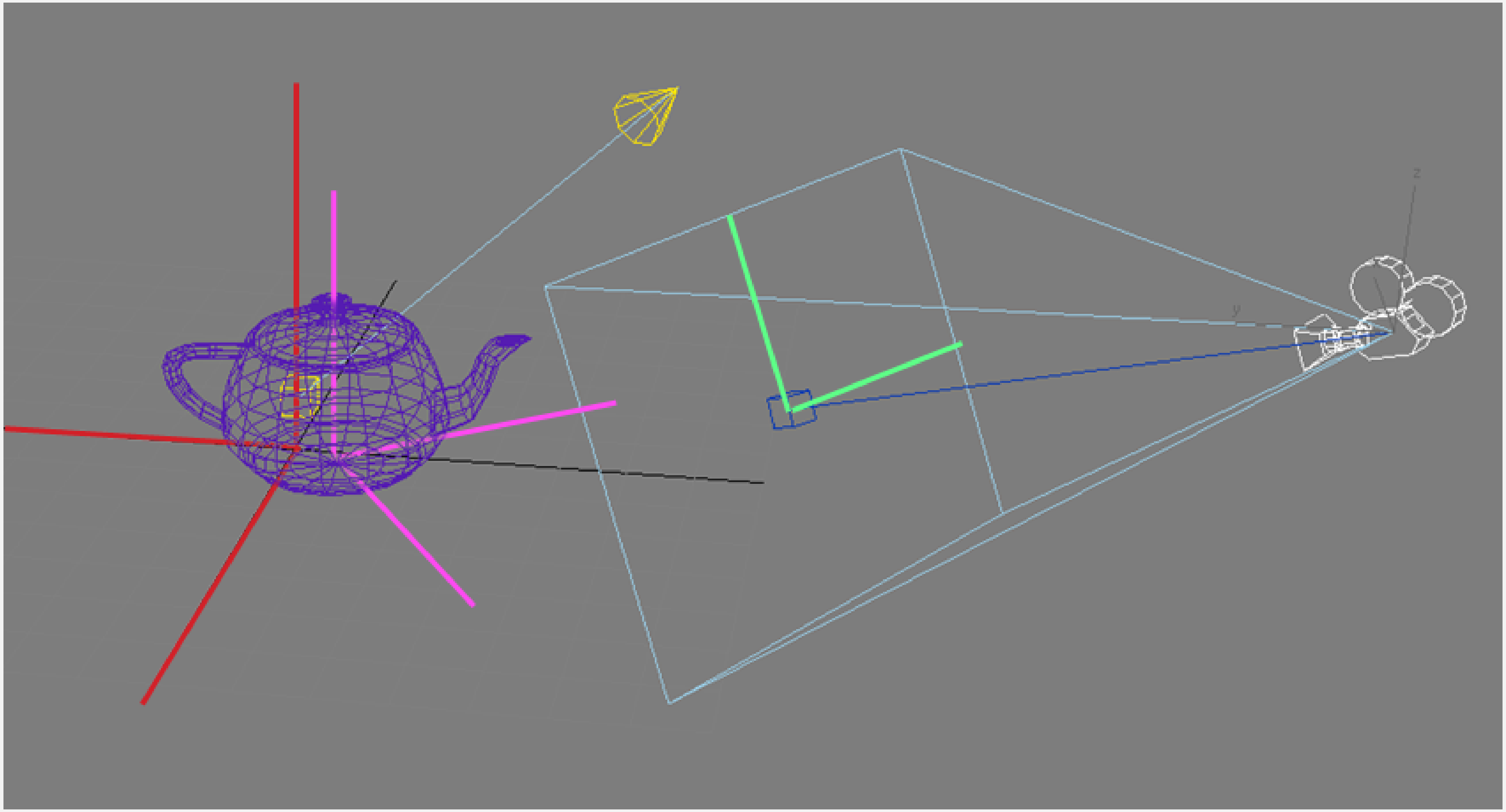
Row/column vector notation

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

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

Projection

Global/local/camera coords.

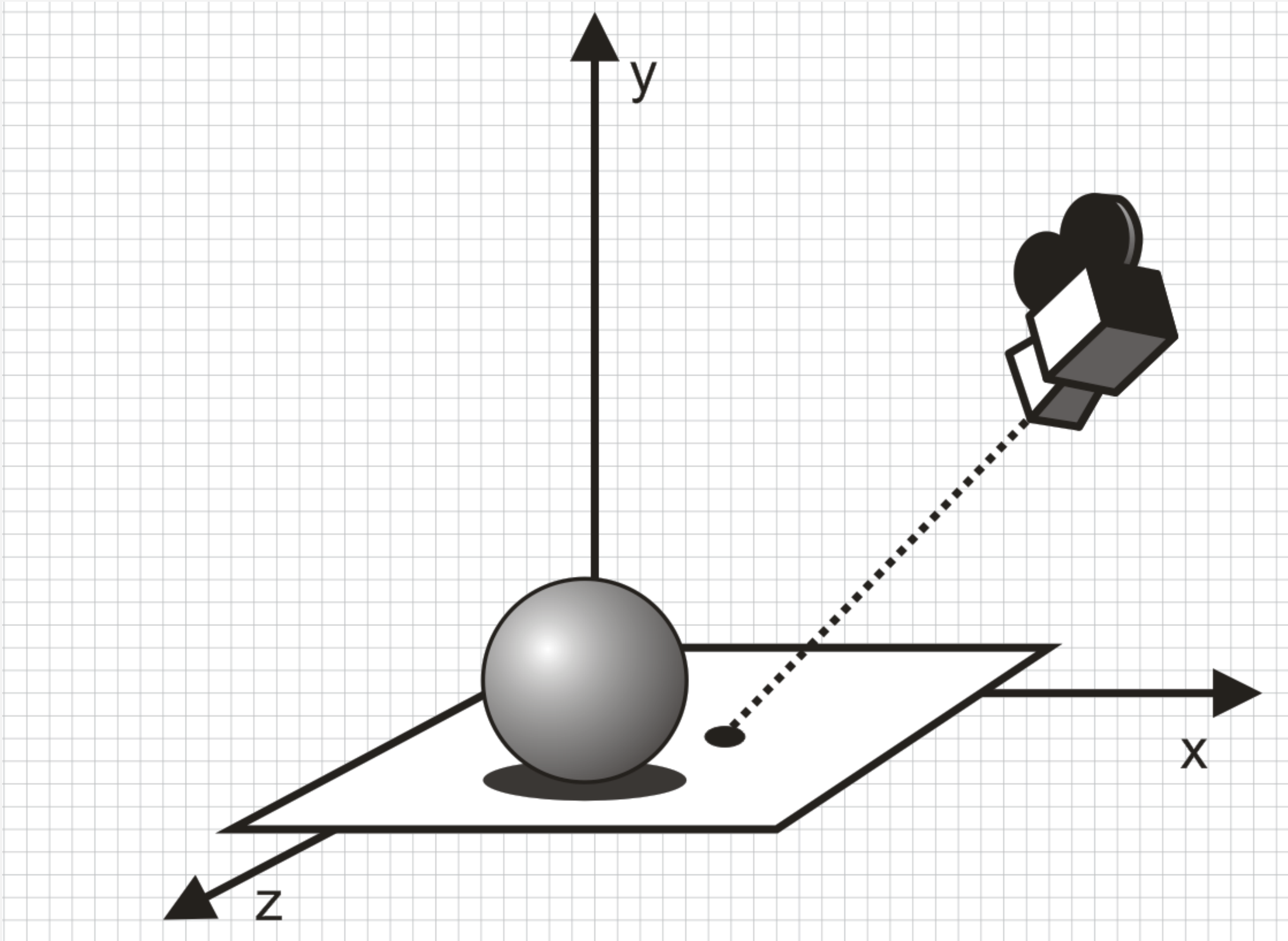


Viewing transformation

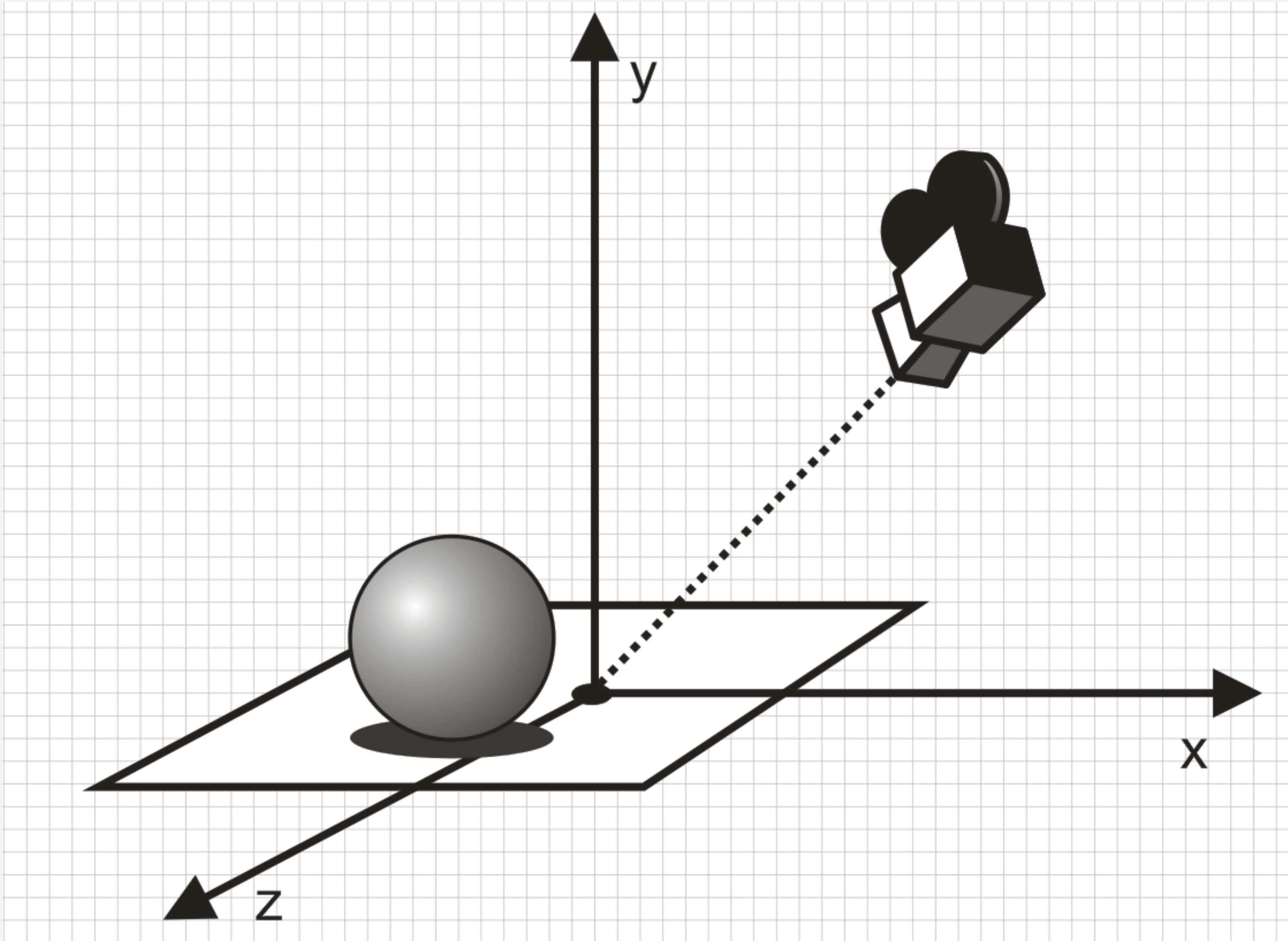
- Convert from local/world coordinates to camera/viewport coordinates

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

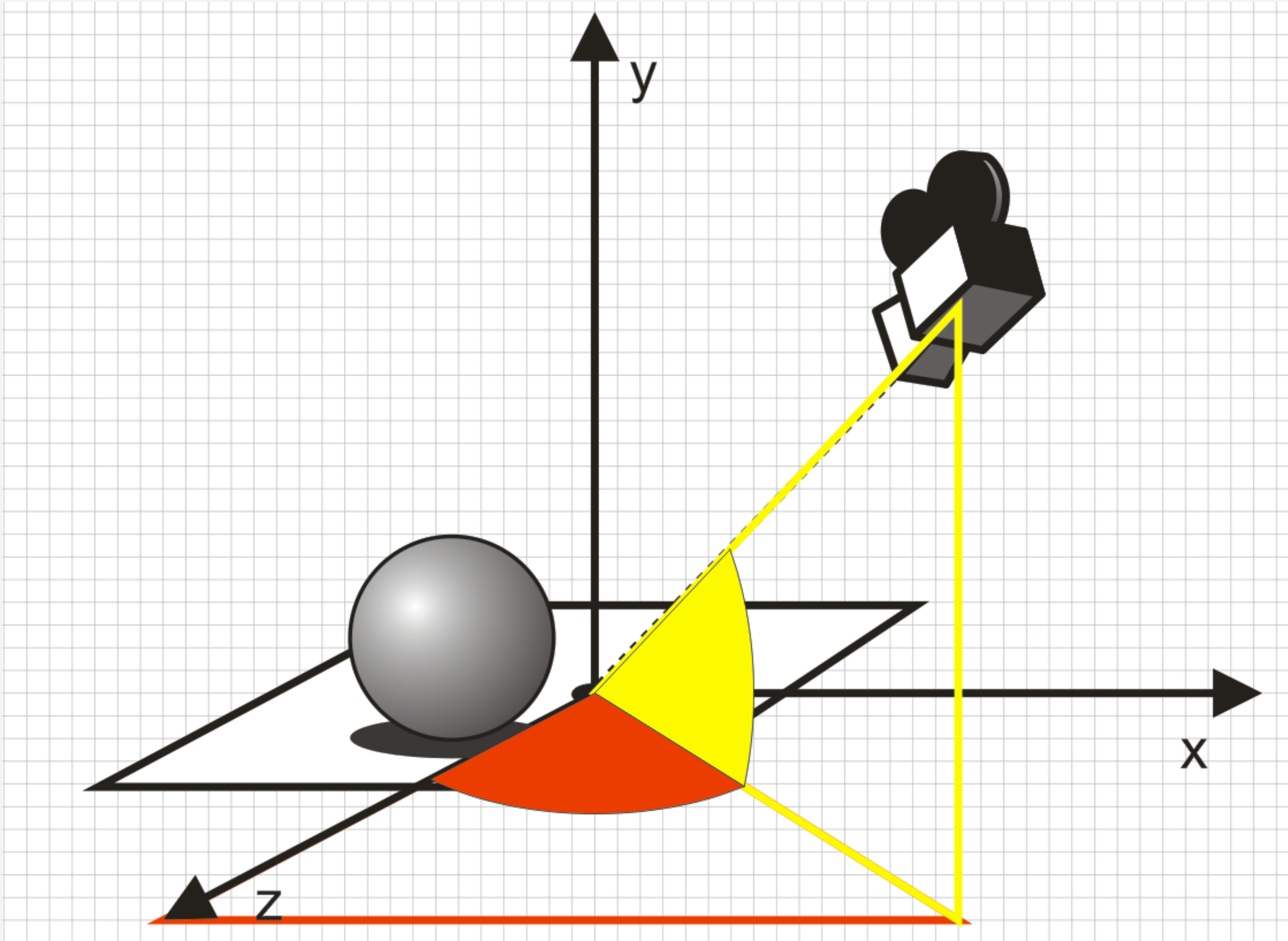
Stage 0



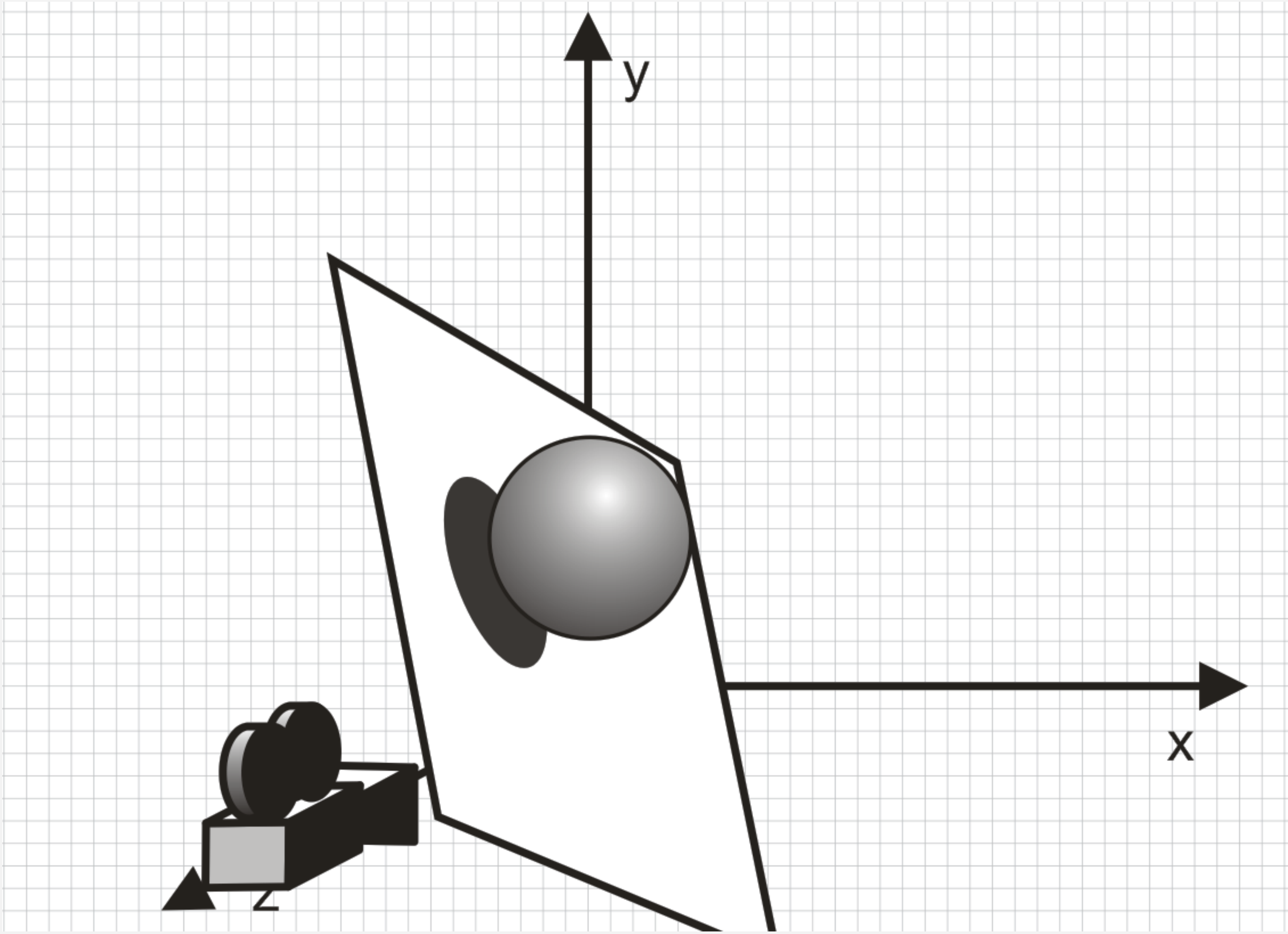
Stage 1 – translate P→P’



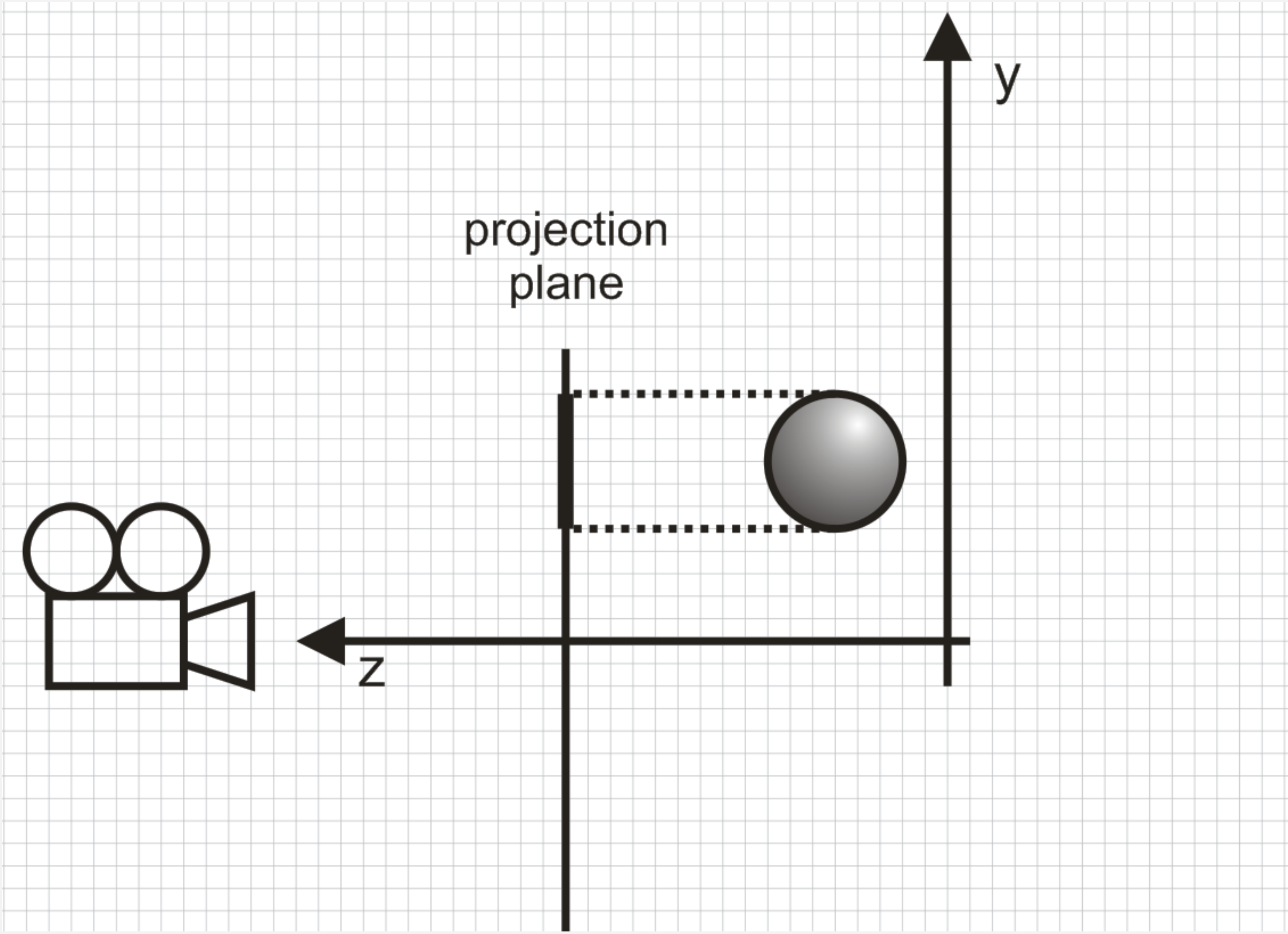
Stage 2 – rotate P'→P''→P'''



Rotated scene



Orthogonal projection



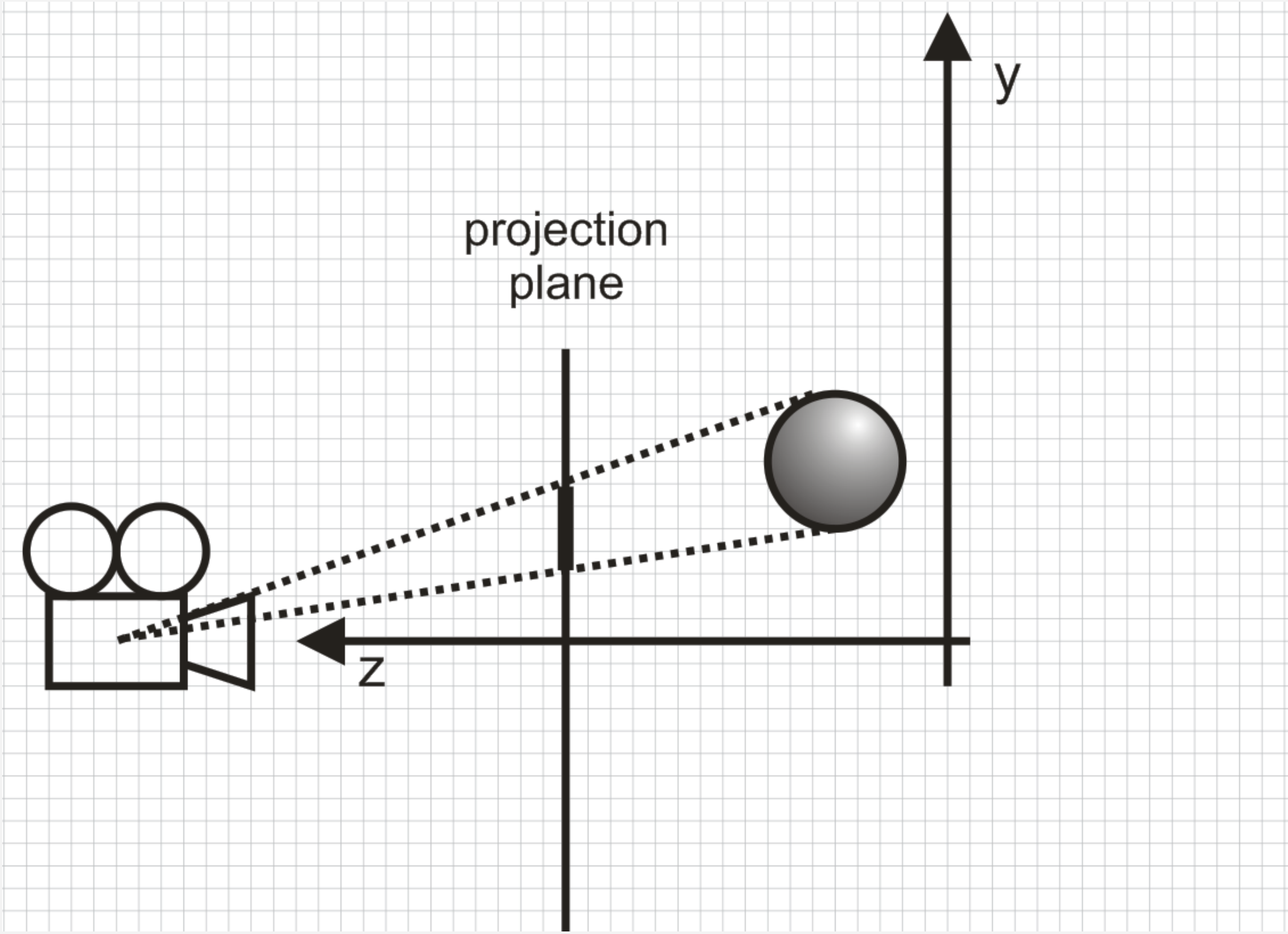
Orthogonal projection

- $x_p = x'''$
- $y_p = y'''$
- 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

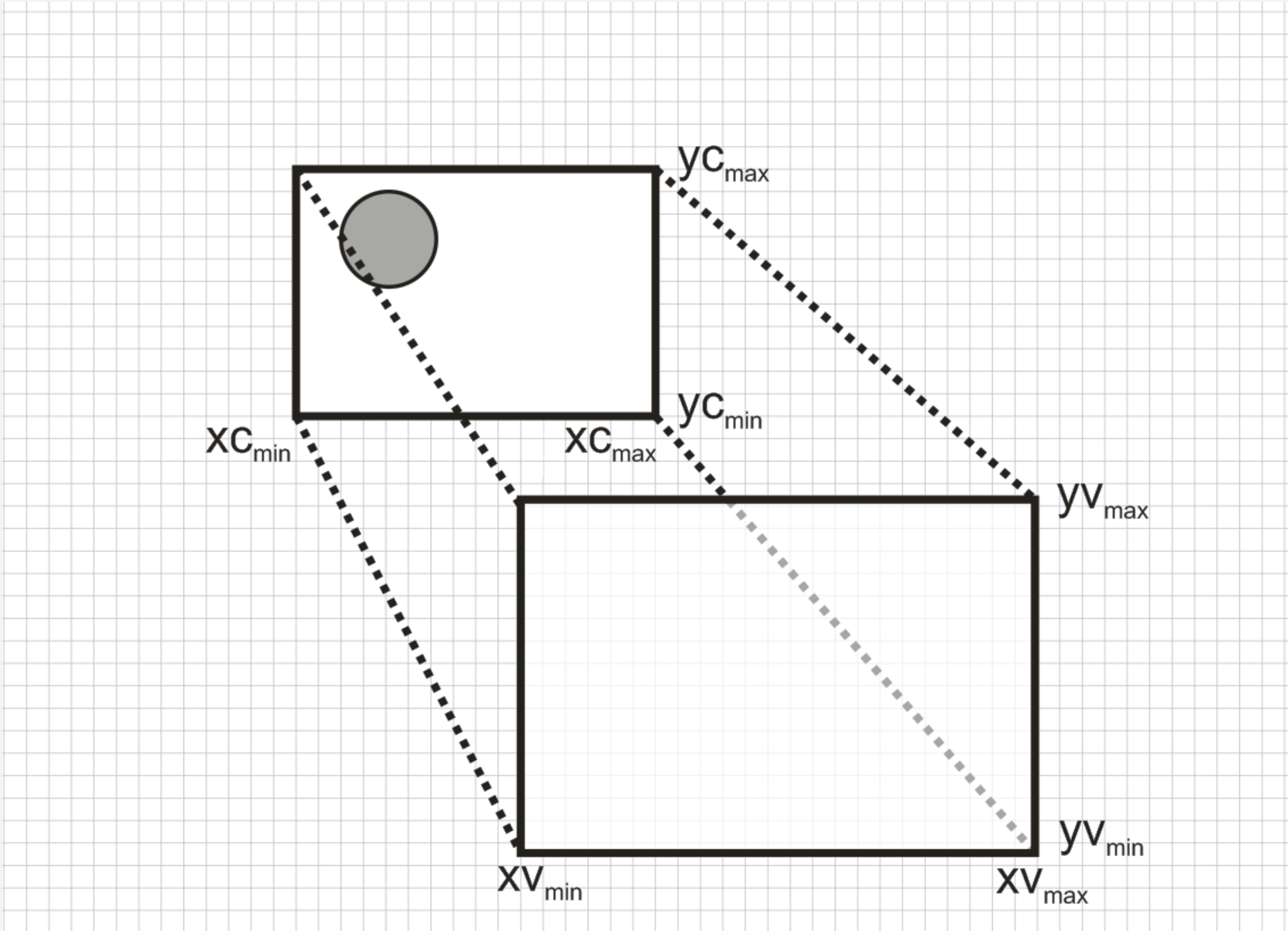


Perspective projection

- x_p = ?
- y_p = ?
- Necessary info:
  - distance between camera and projection plane
- Matrix notation

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

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

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

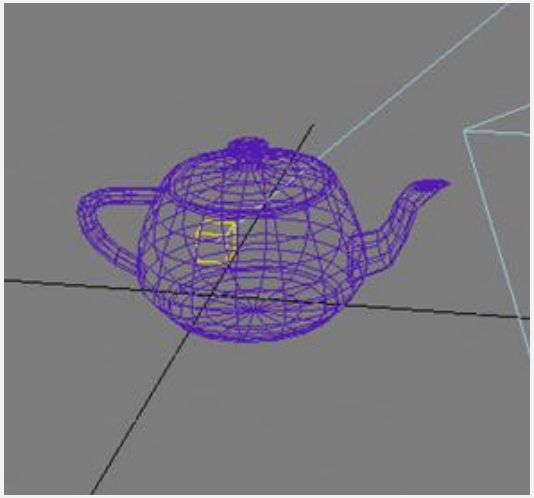
Welcome to the matrix!

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

Rendering pipeline

- Model transformation
  - local → global coordinates
- Viewport transformation
  - global → camera
- Clipping
- Rasterization
- Texturing & Lighting



Next week:

Rasterization, culling, clipping