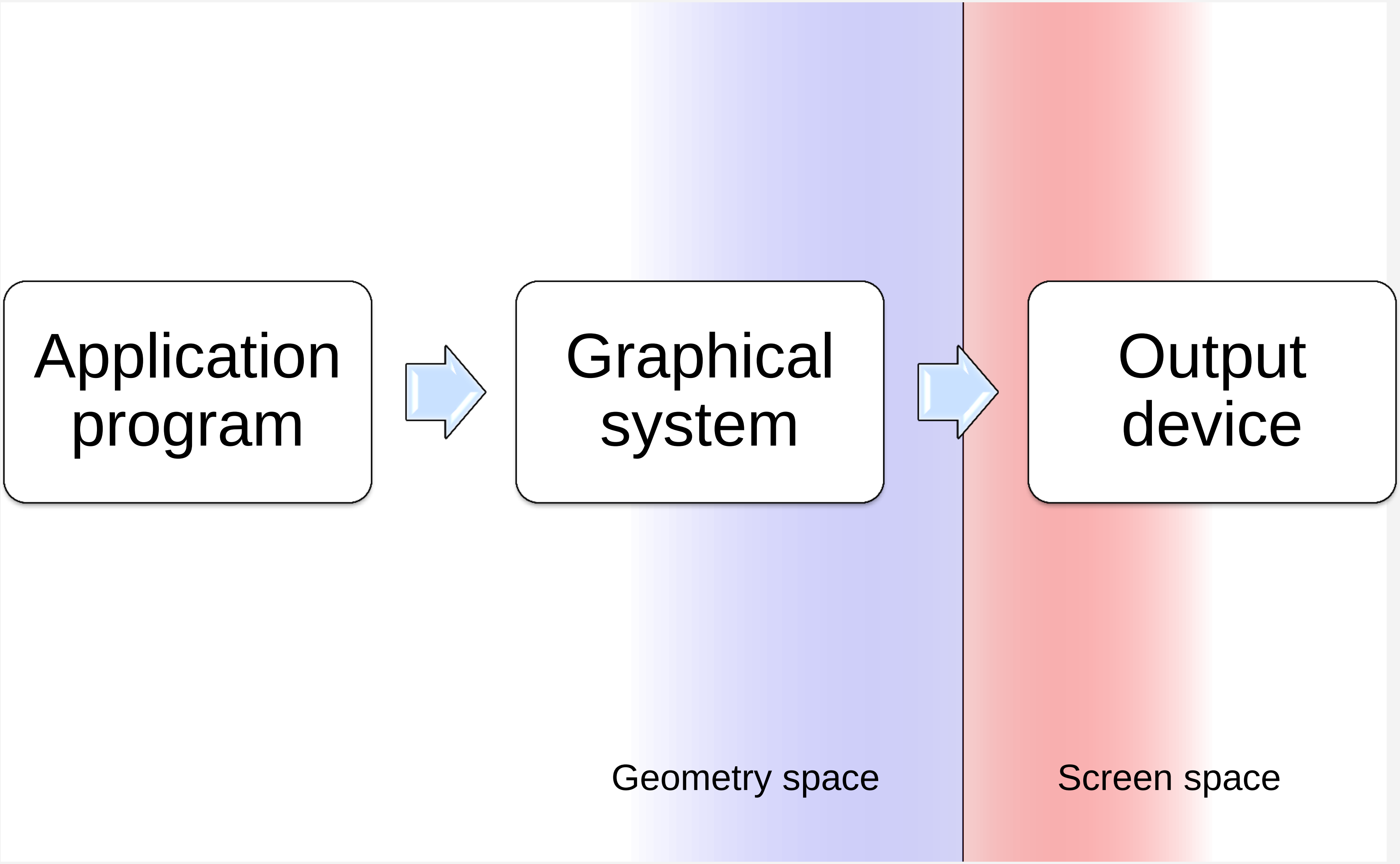
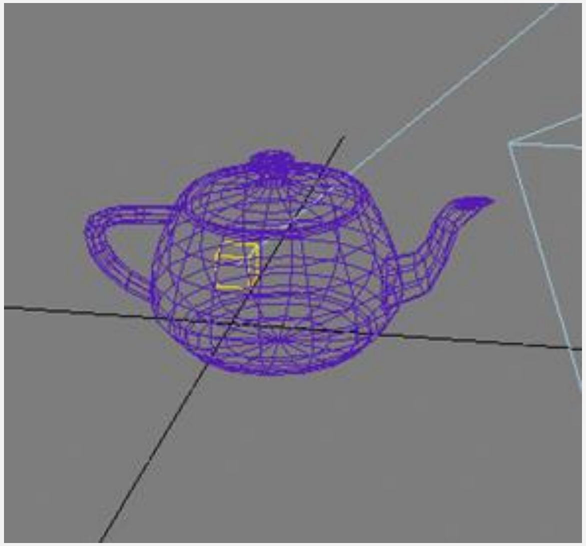
Last lesson summary

CG reference model

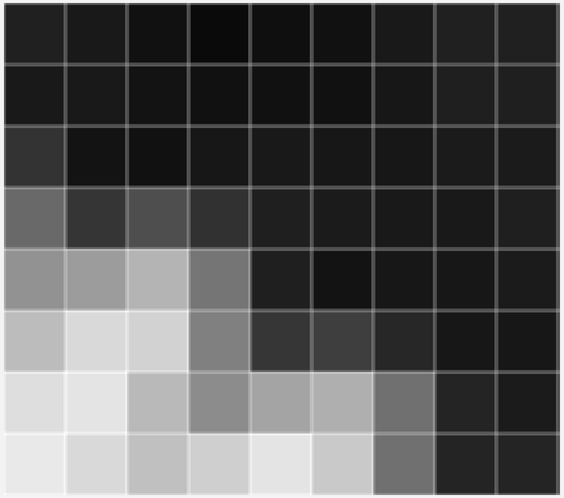


Recollections

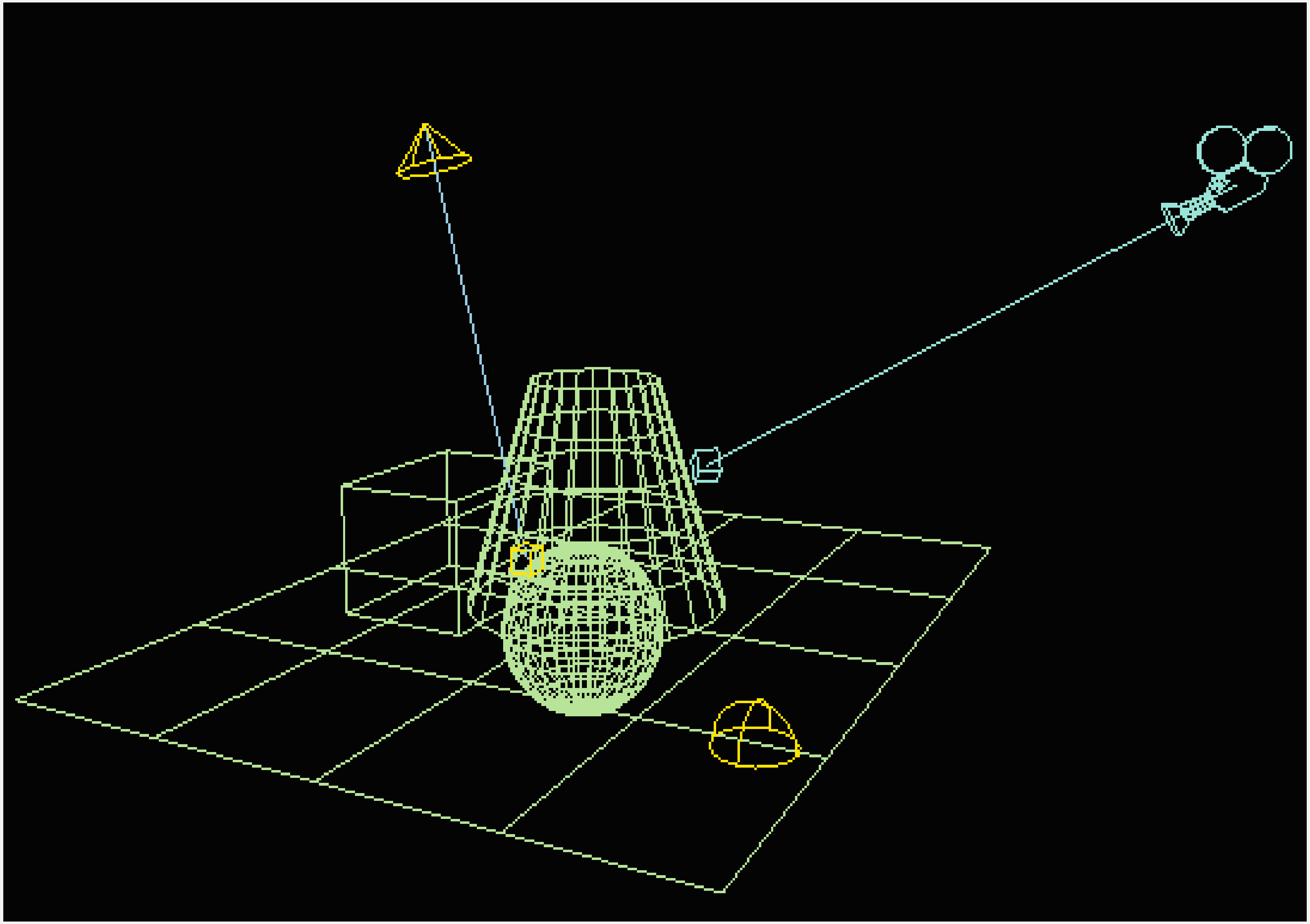
- Geometry space
  - continuous
  - 3Dimensional



- Screen space
  - discrete
  - 2Dimensional



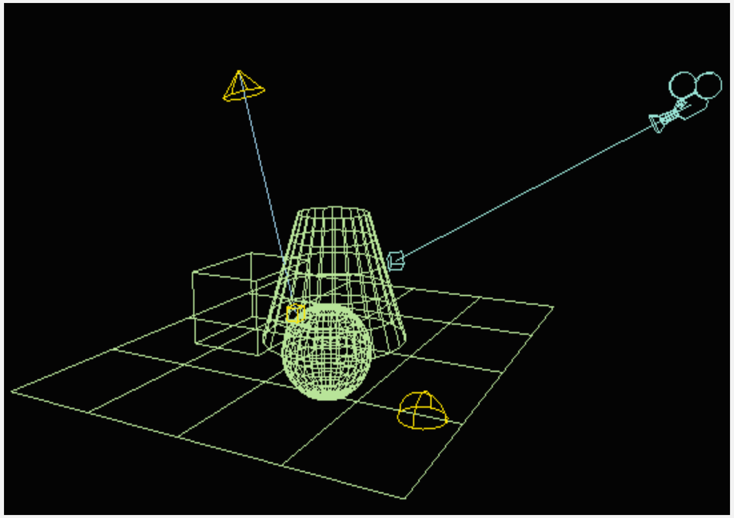
3D Scene vs. 2D image

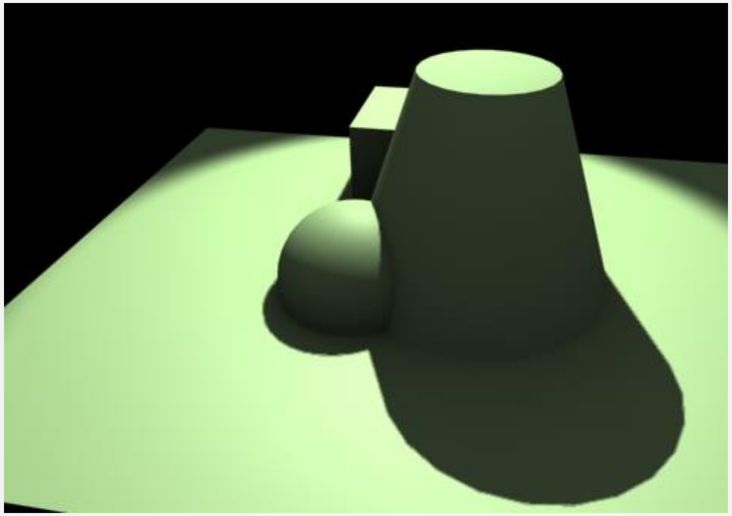


Geometry vs. screen space

- 3D
- Continuous
- Parametric
- Models

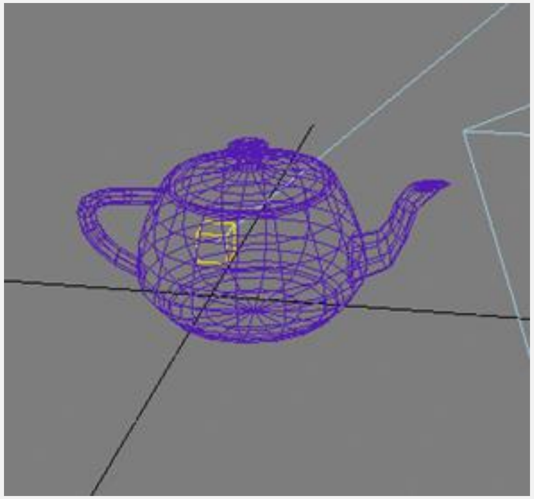
- 2D
- Discrete
- Non-parametric
- Pixels





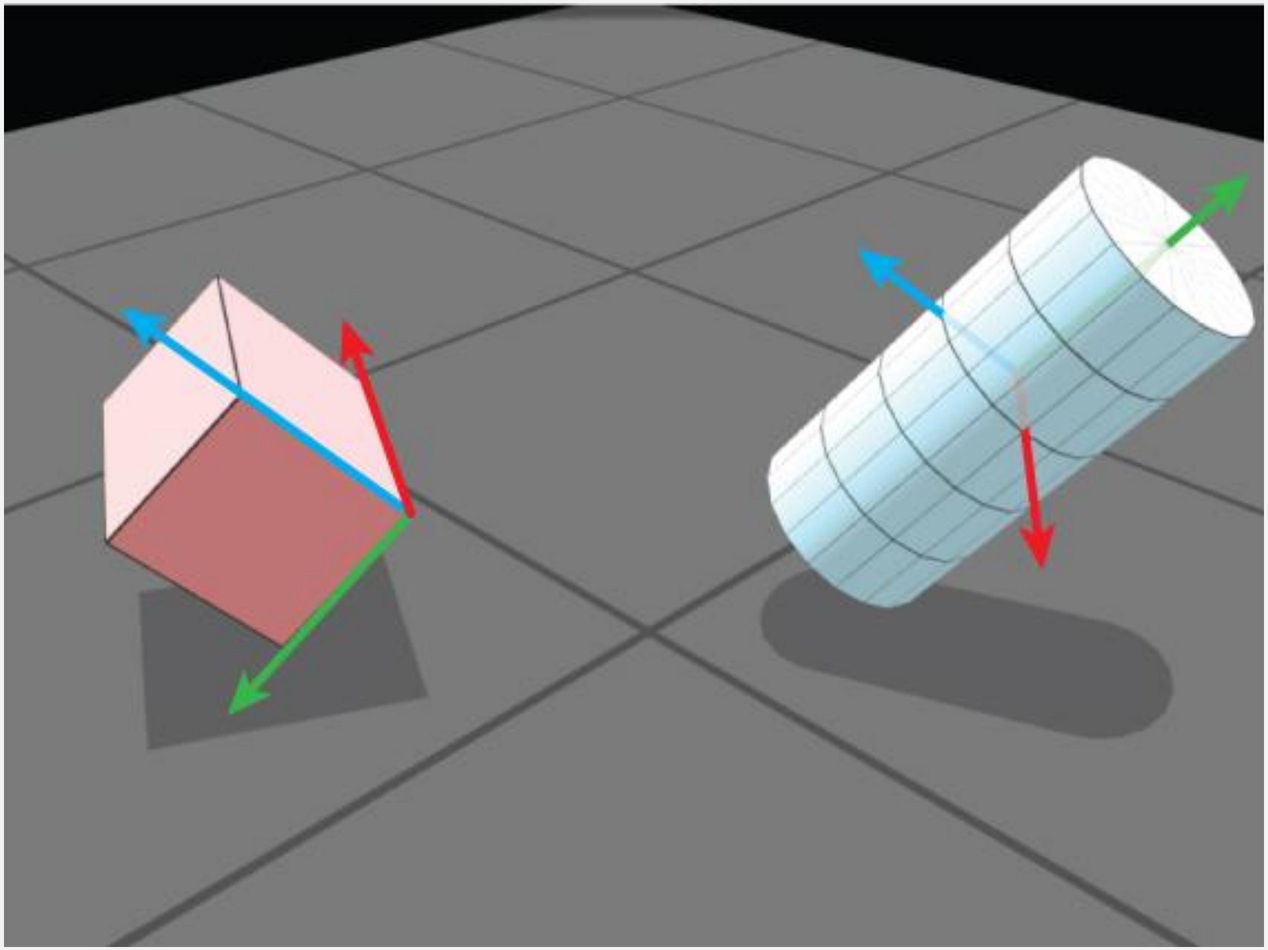
Rendering pipeline

- Model transformation
  - local → global coordinates
- View transformation
  - global → camera
- Projection transformation
  - camera → screen
- Clipping, rasterization, texturing & Lighting
  - might take place earlier



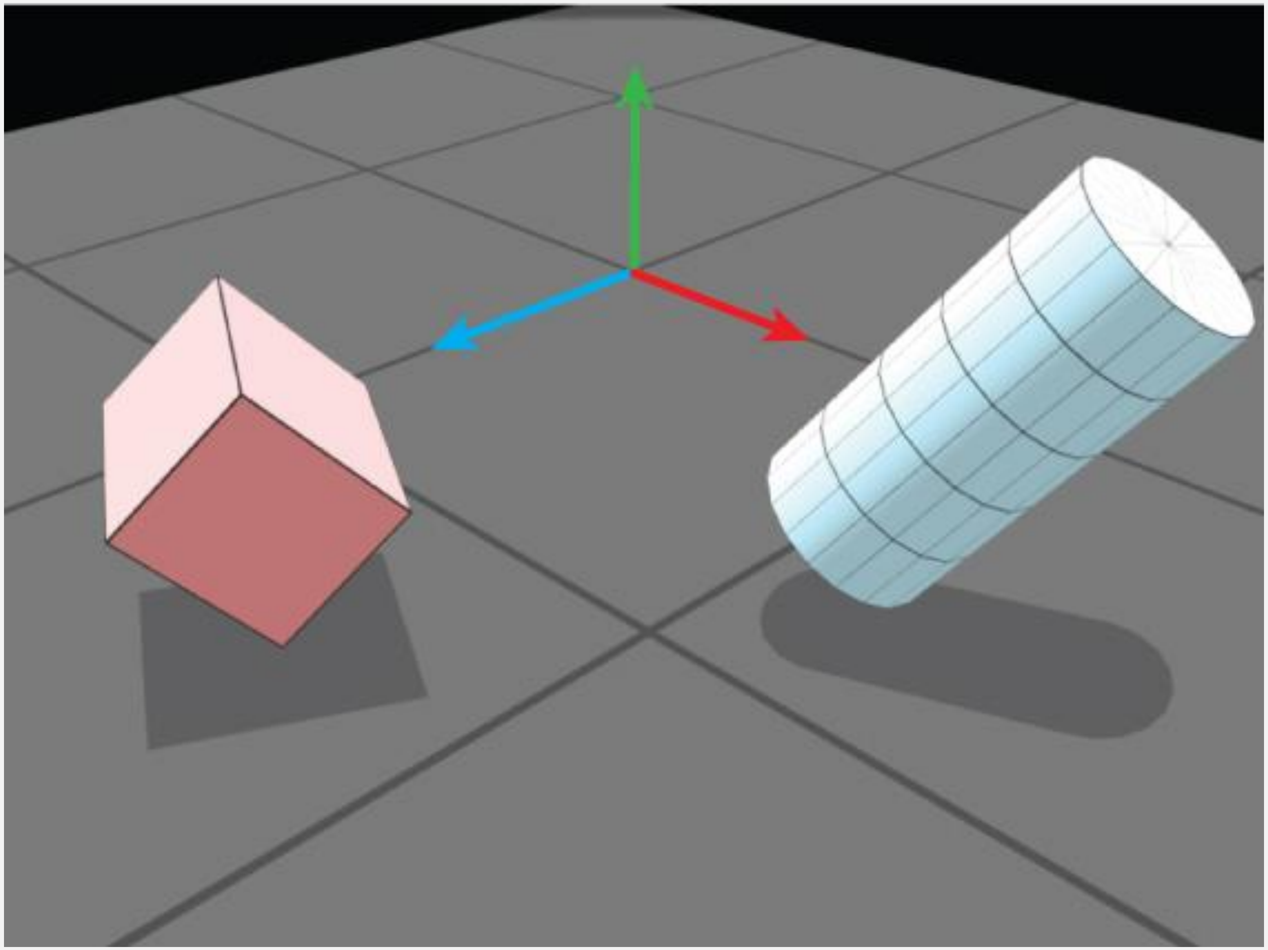
Local coordinates

- Each object has its own coordinate system



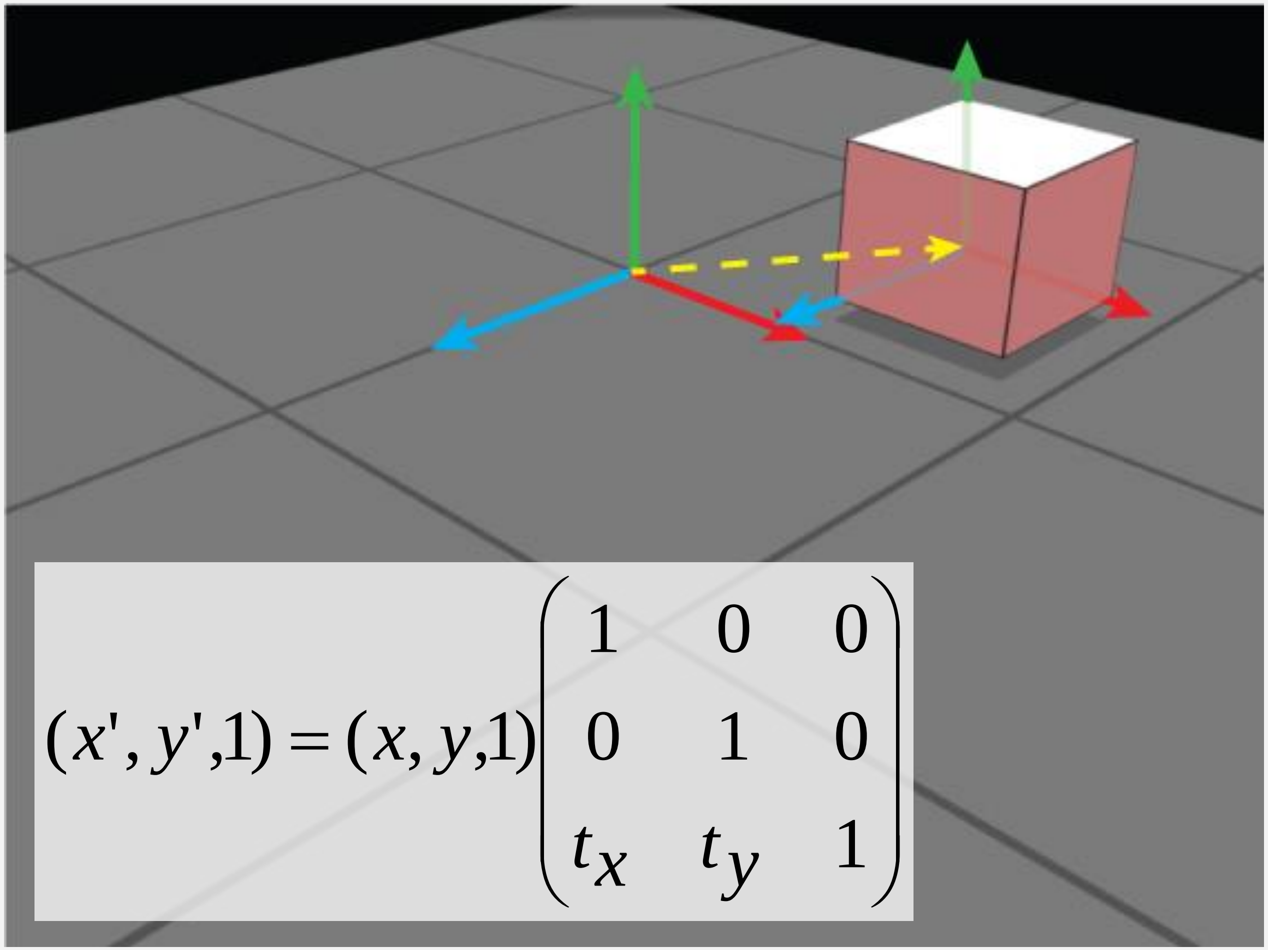
Global coordinates

- One system for the whole scene



Local → Global coordinates

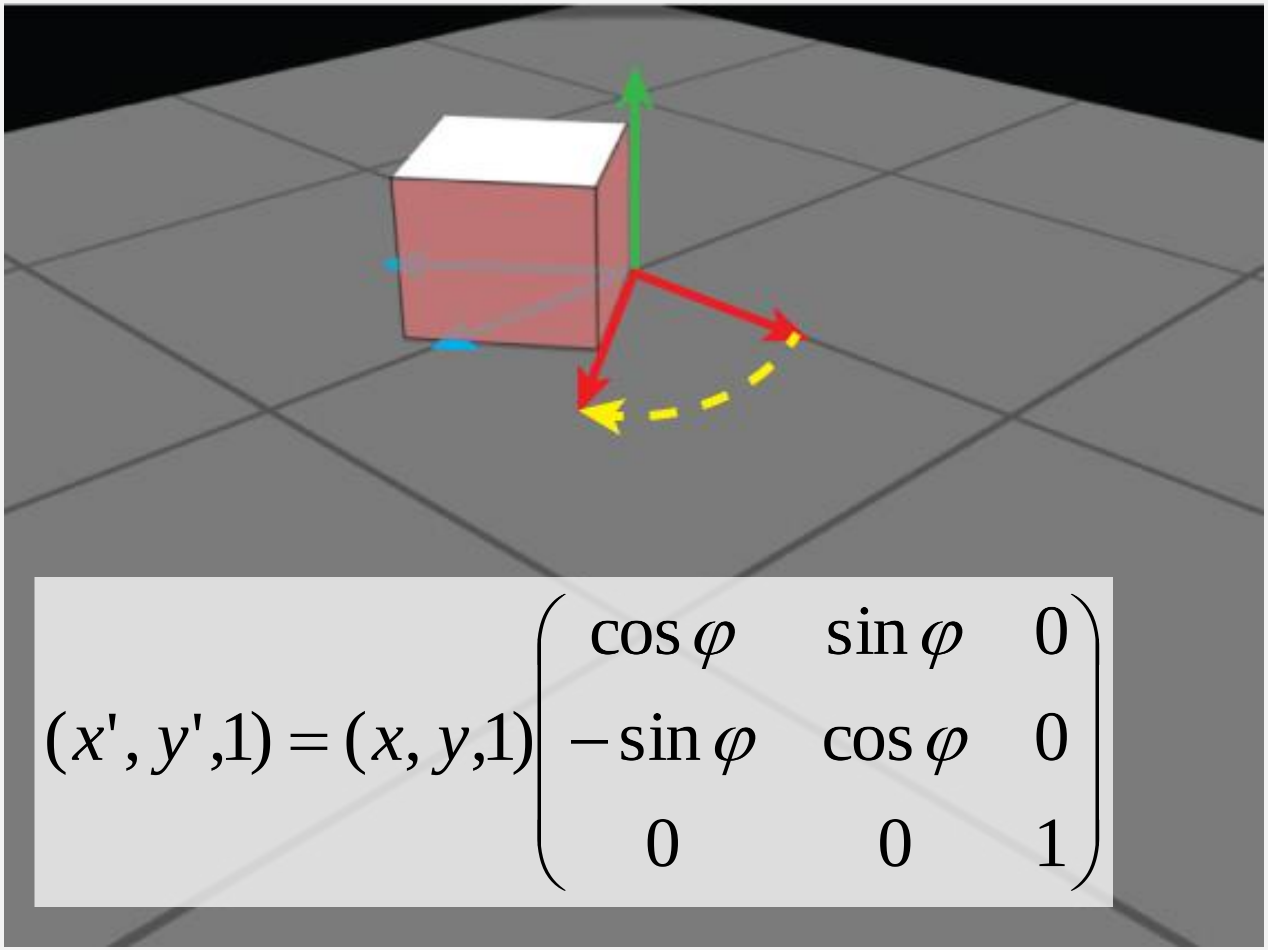
- Translation



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

Local → Global coordinates

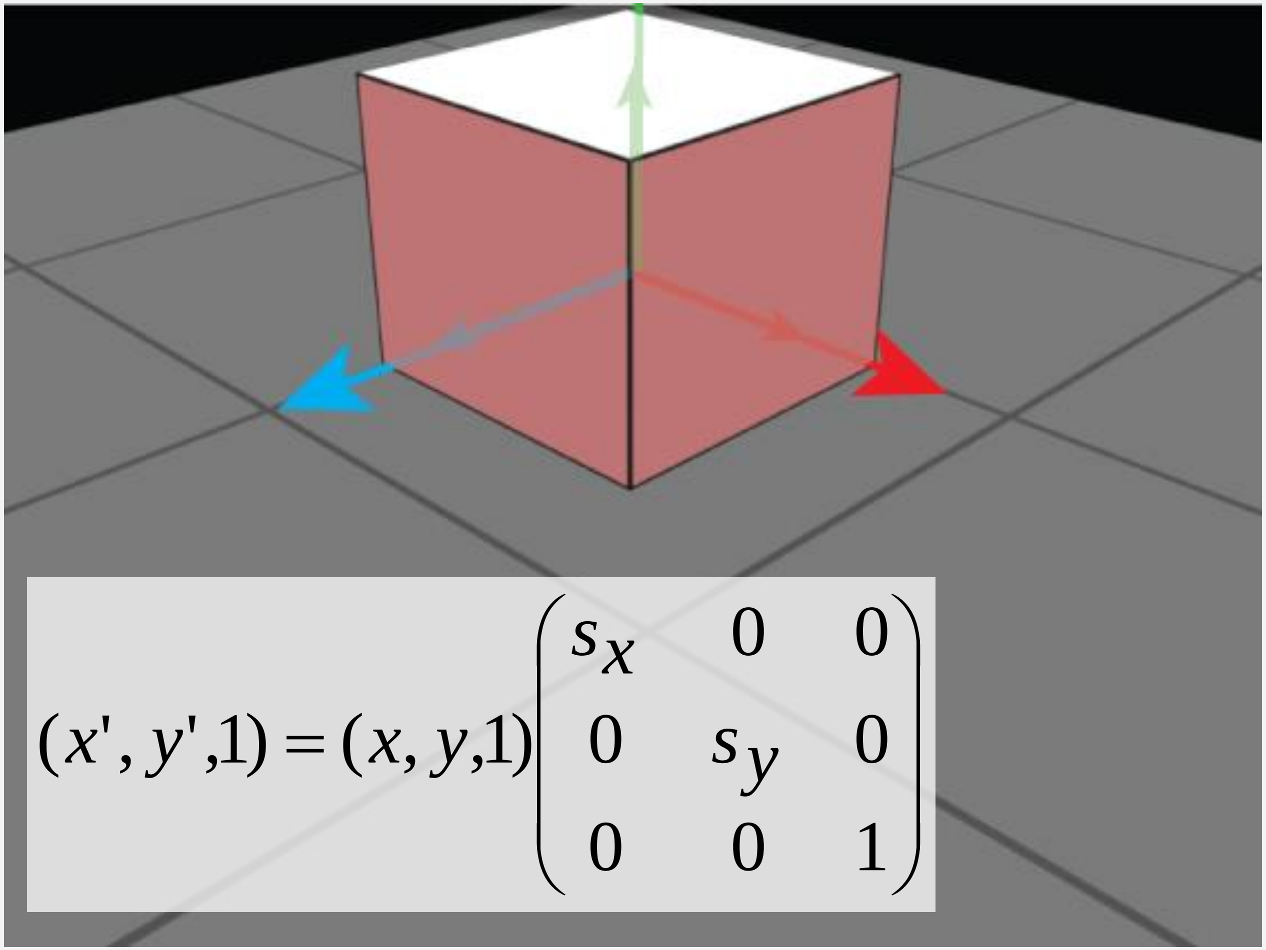
- Rotation



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

Local → Global coordinates

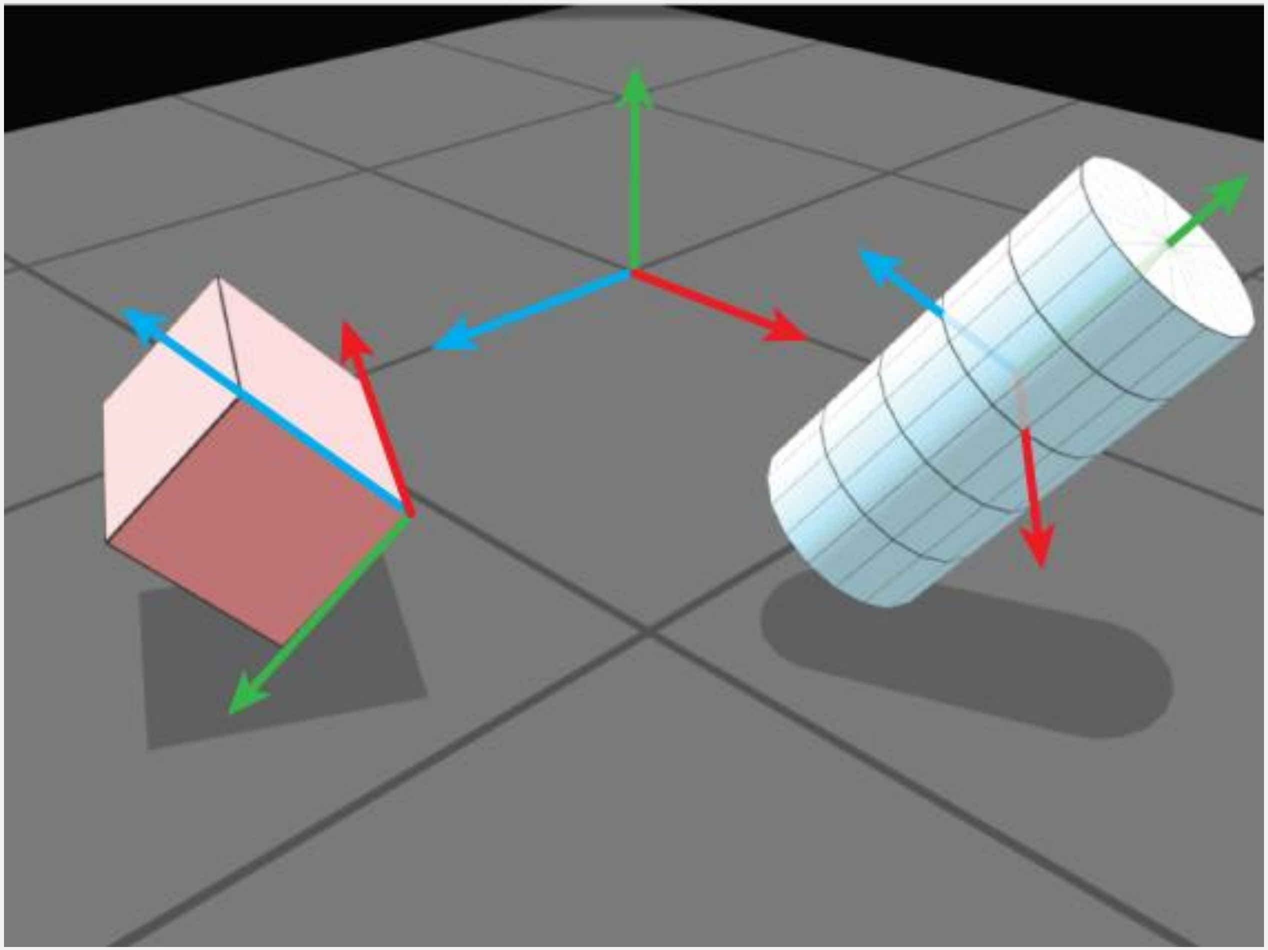
- Scaling



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

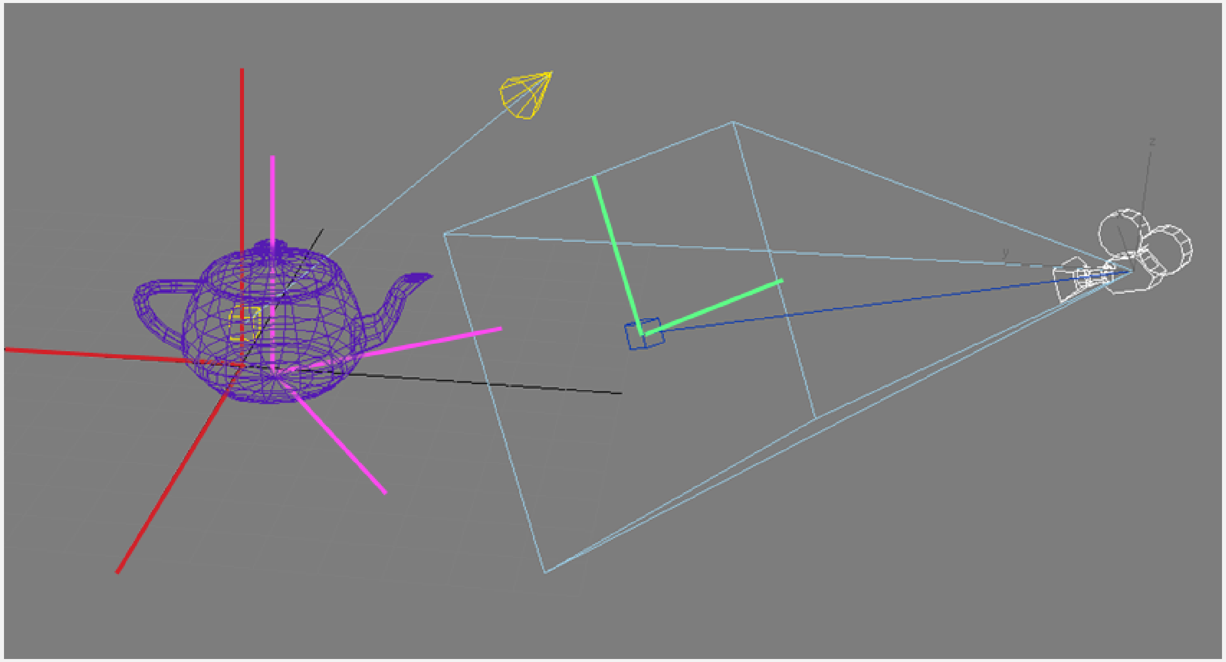
Local → Global coordinates

- All transformations combined



Transformations

- Transformation from one coordinate system to another one is a composition of partial transformations:
  - Translation
  - Rotation
  - Scaling

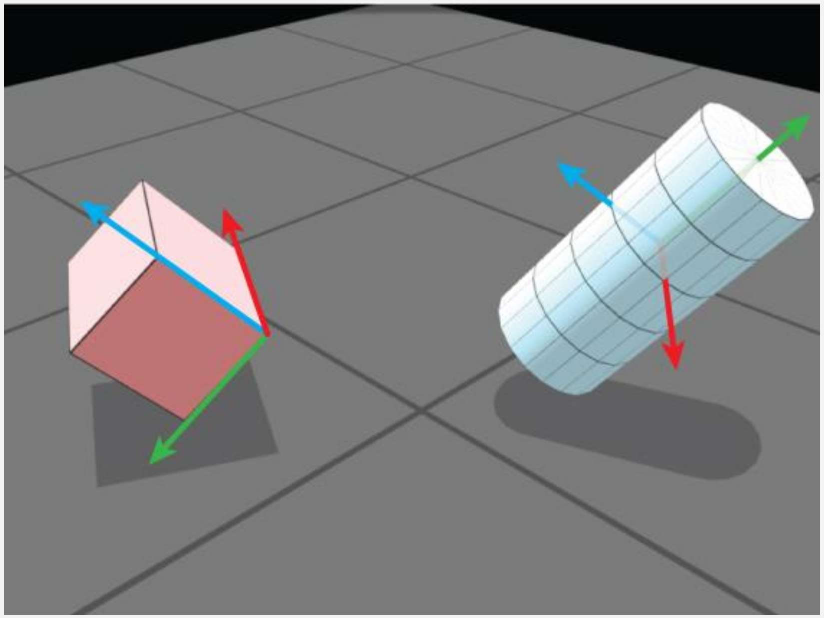


All transformations

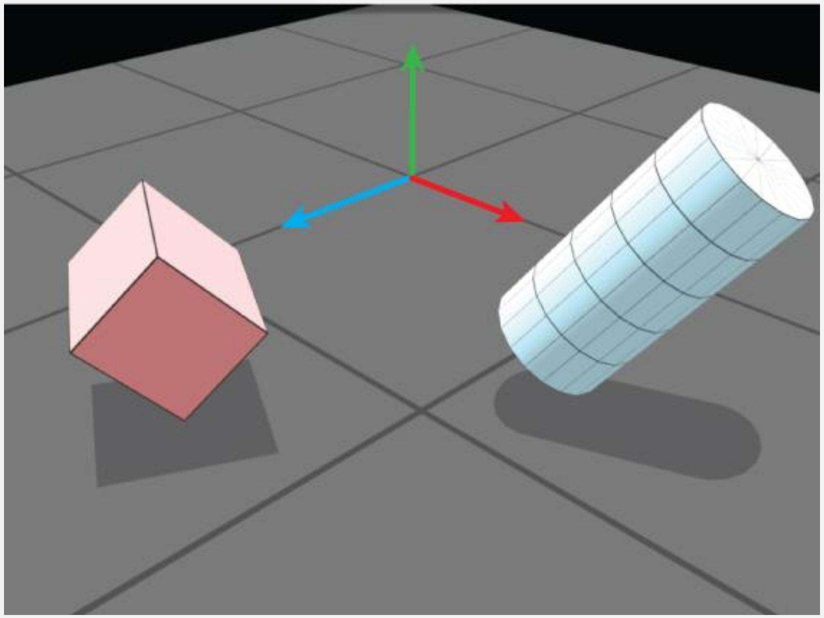
- Model transformation
  - Unify coordinates by transforming local to global coordinates
- View transformation
  - Transform global coordinates so that they are aligned with camera coordinates
  - To make projection computable

Model transformation

- Transformation local → global
- Combination of rotate, translate, scale
- Matrix multiplication

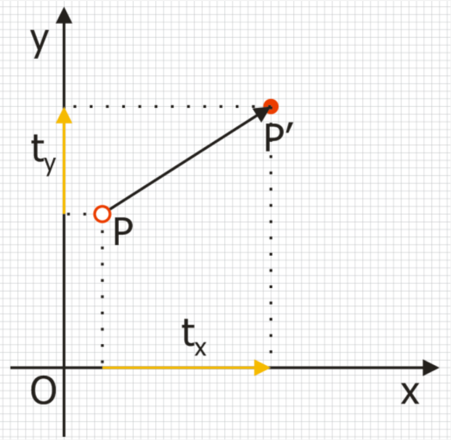


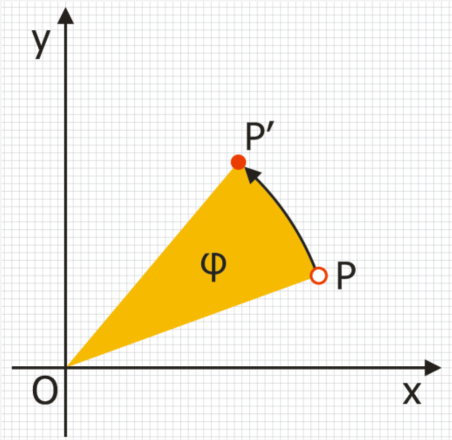
→

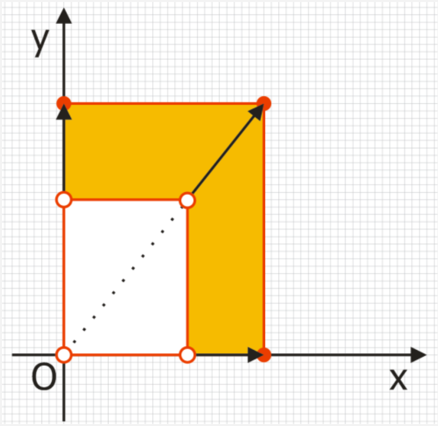


Transformations

- Translation, rotation, scaling





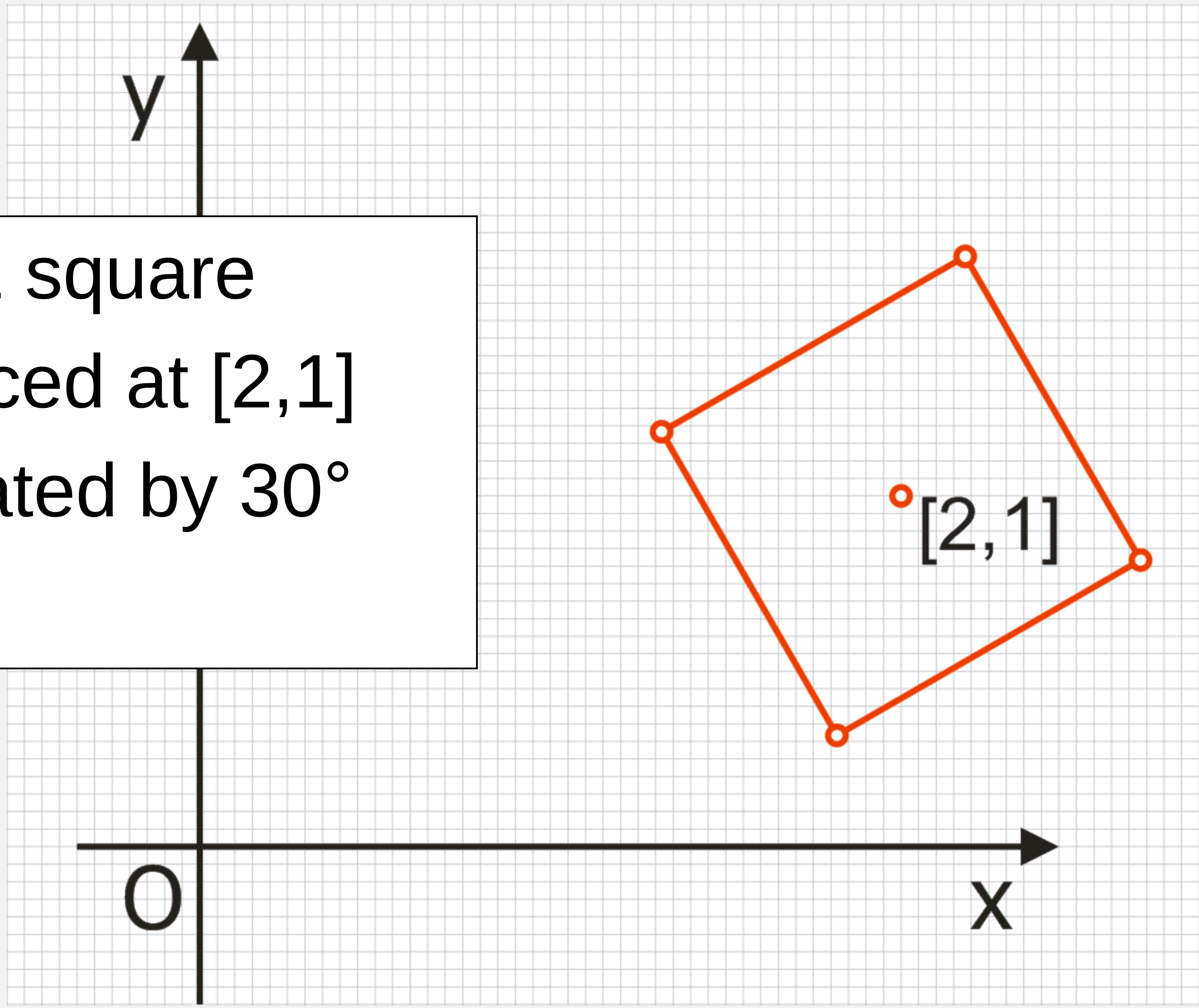


$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix} \qquad \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Example

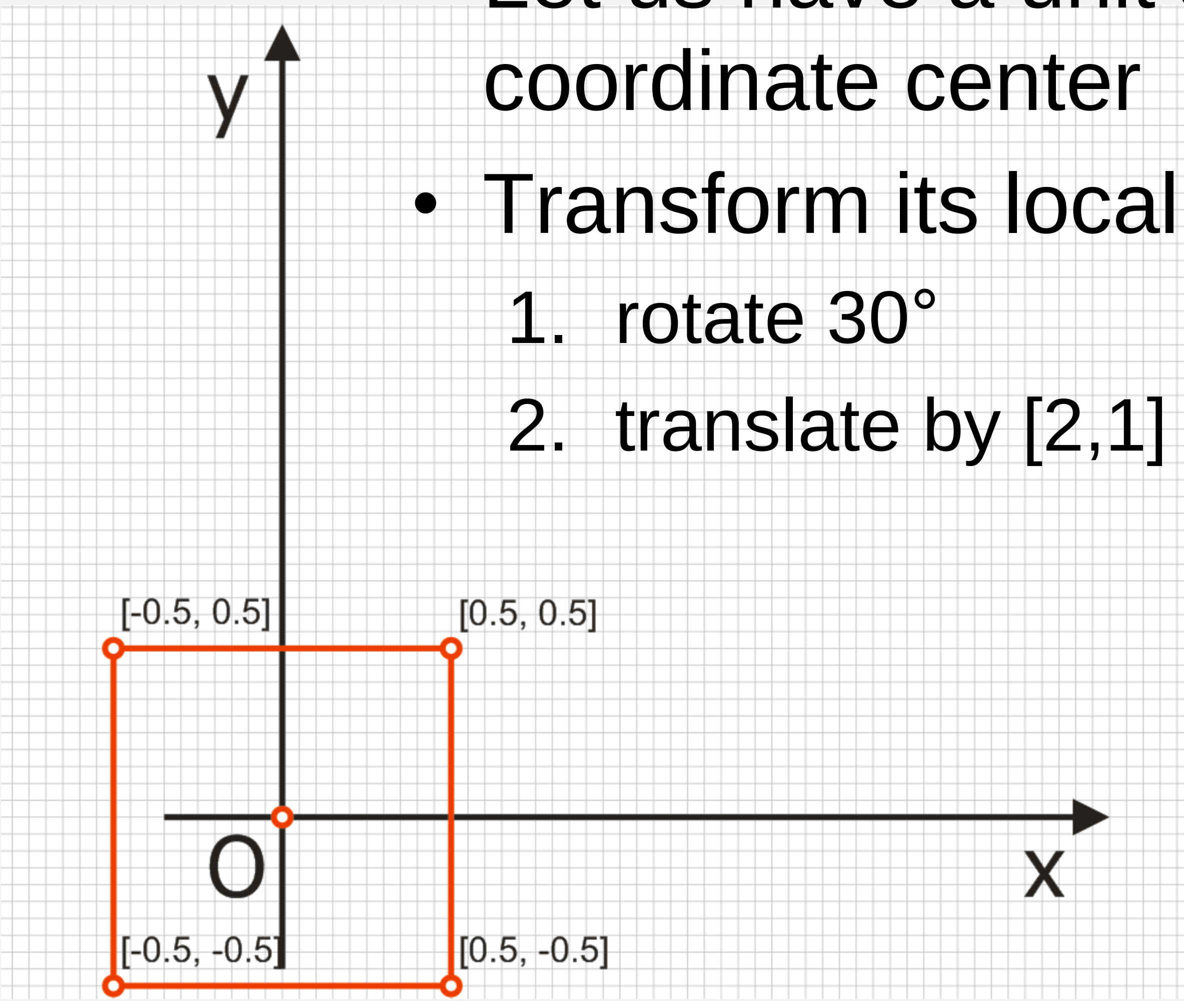
Goal

- 1x1 square
- placed at [2,1]
- rotated by 30°



Model transformations

- Let us have a unit square around coordinate center
- Transform its local coordinates:
  1. rotate 30°
  2. translate by [2,1]



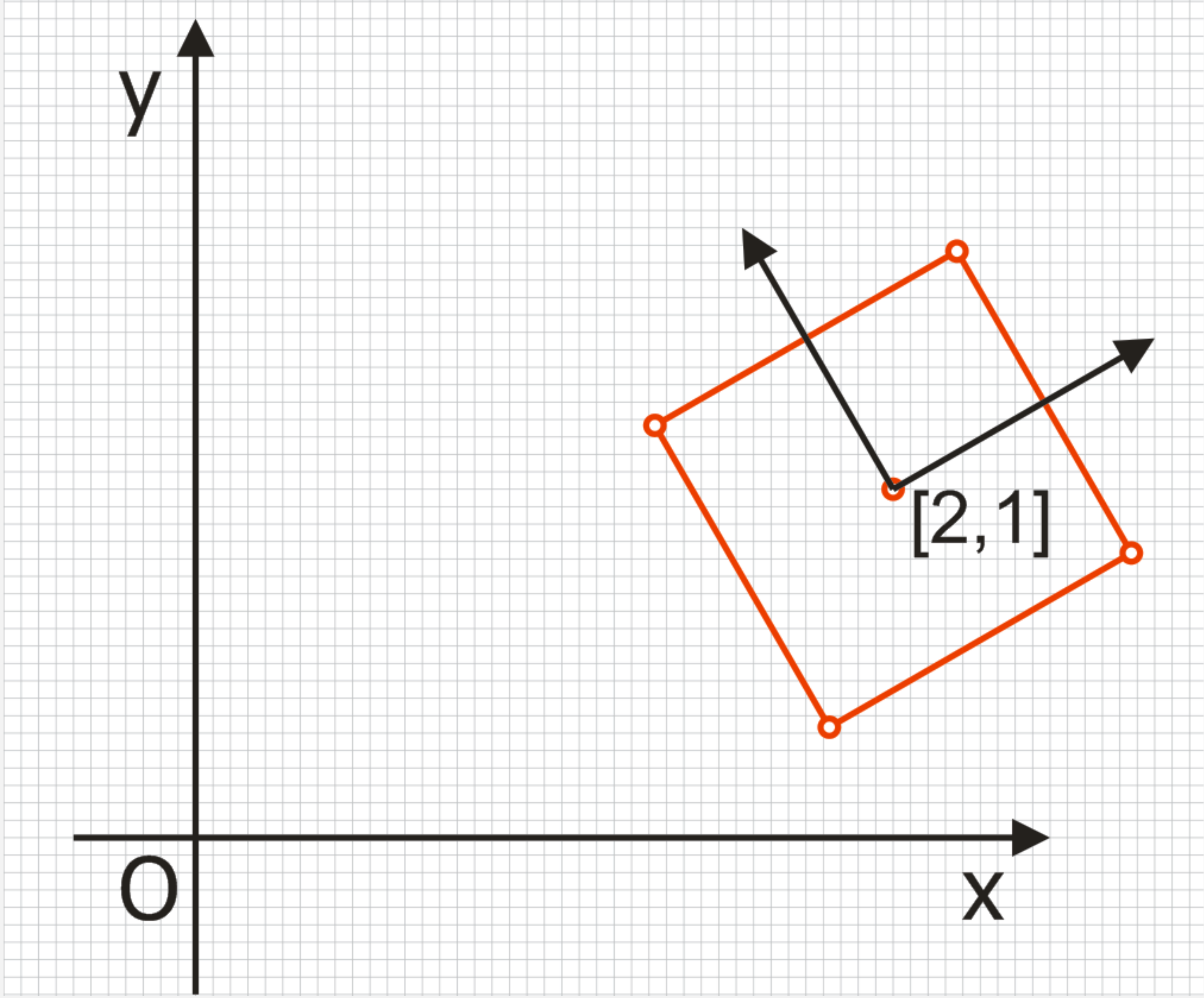
Model transformations

Rotation by 30° + Translation by [2,1] =

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 2 & 1 & 1 \end{pmatrix}$$

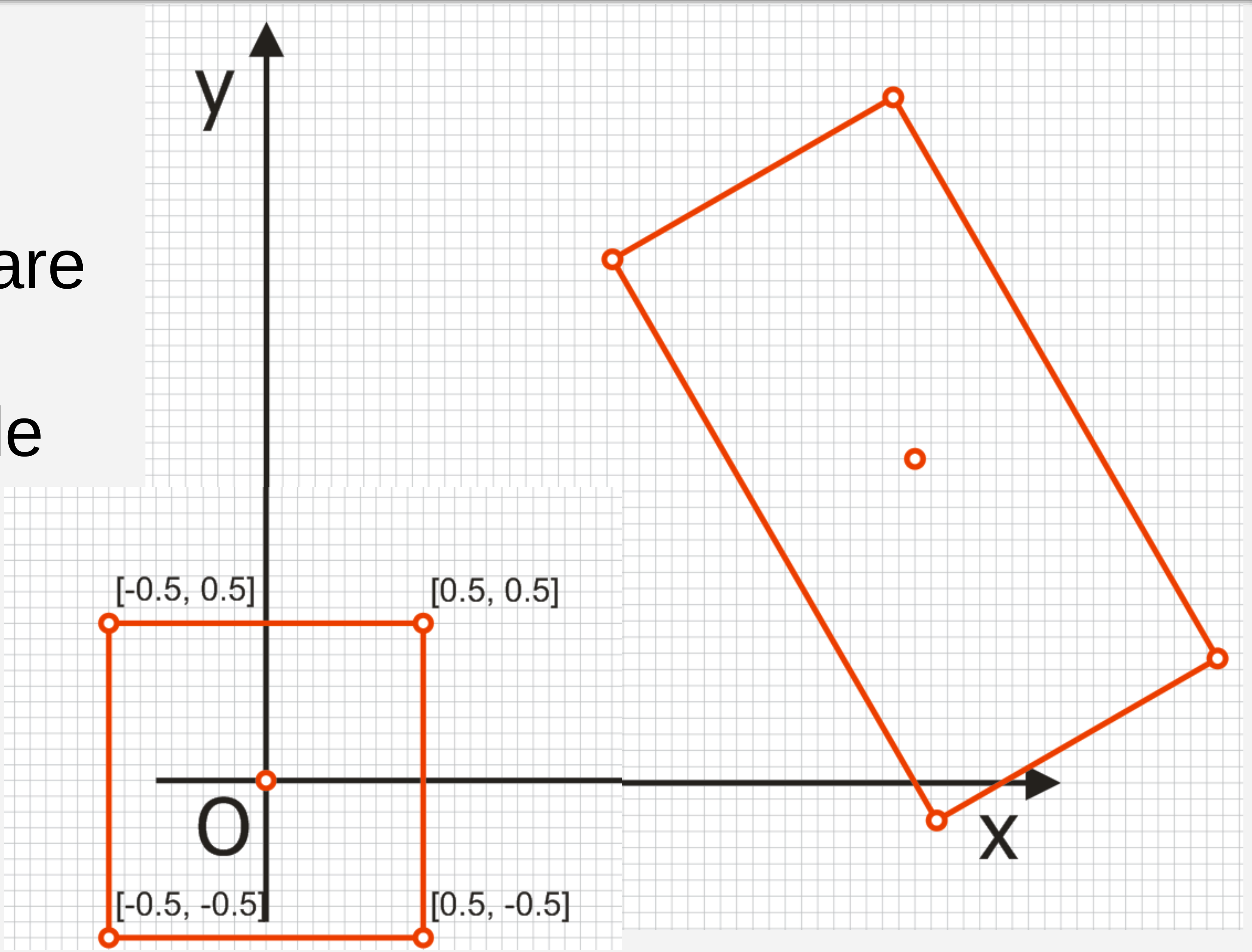
R . T = M

Result:



Why local coordinates?

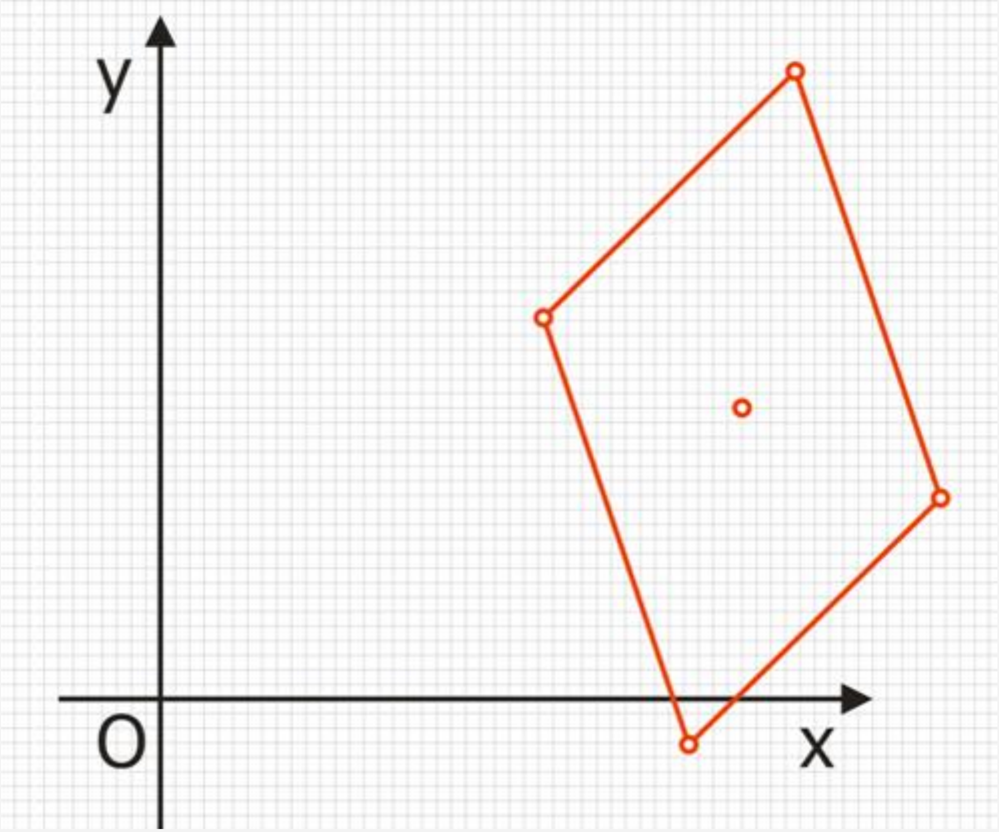
Goal 2: stretch the square to a rectangle



Scale *y* by 2

- $\boldsymbol{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

R * T * S = M

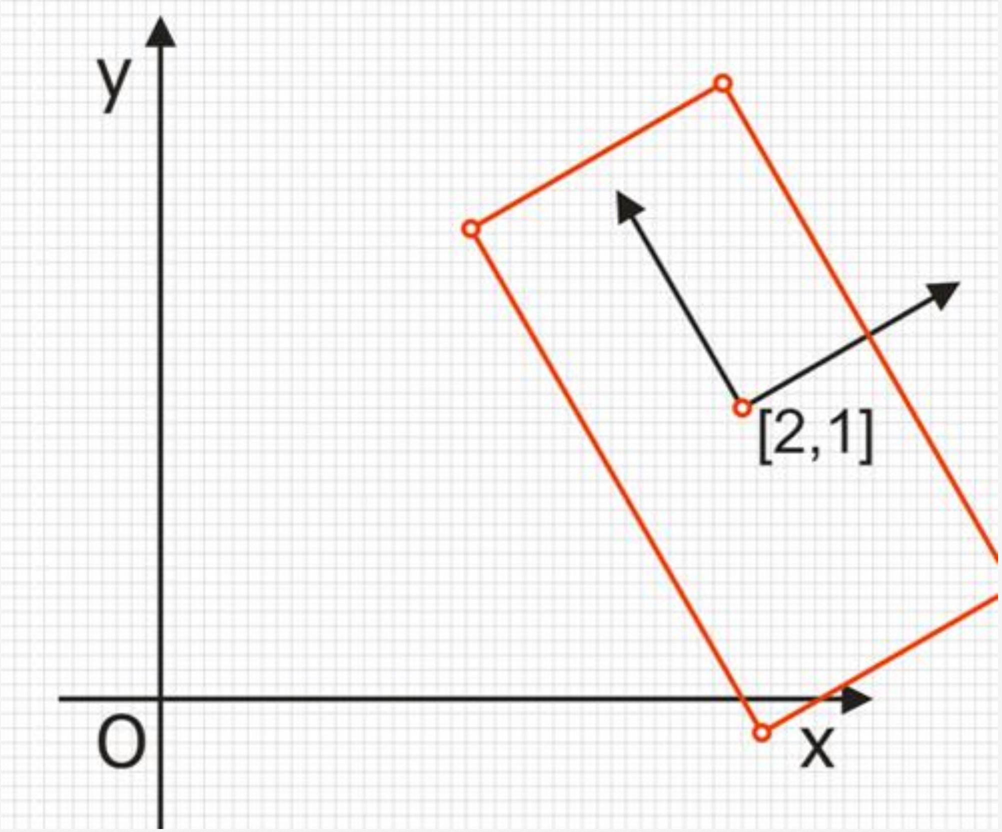


- Result =

Local scaling

- $S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

S * R * T = M



- Result =

Final model transformation

$$S * R * T = M$$

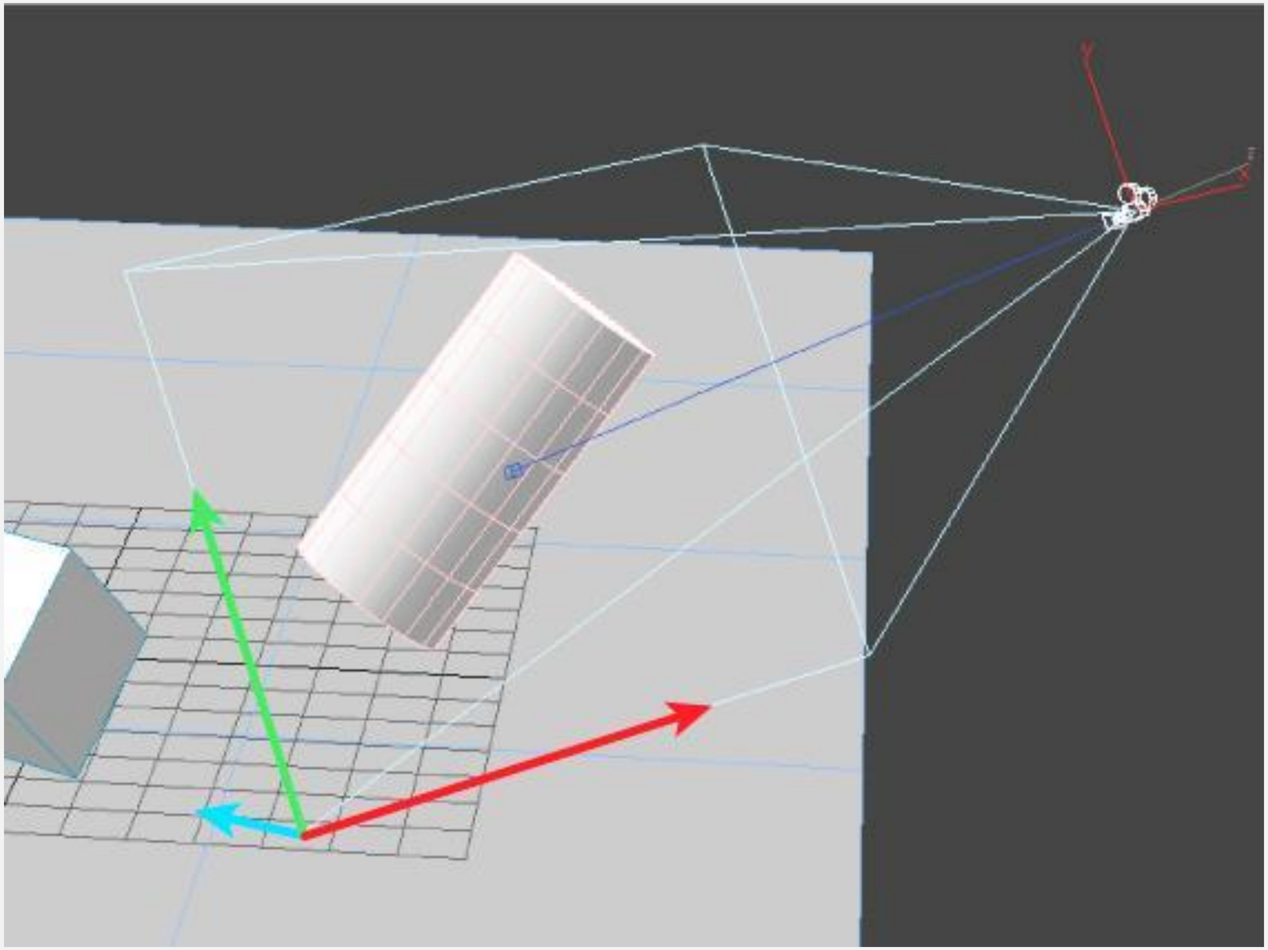
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ -1 & \sqrt{3} & 0 \\ 2 & 1 & 1 \end{pmatrix}$$

Remember! $A * B \neq B * A$

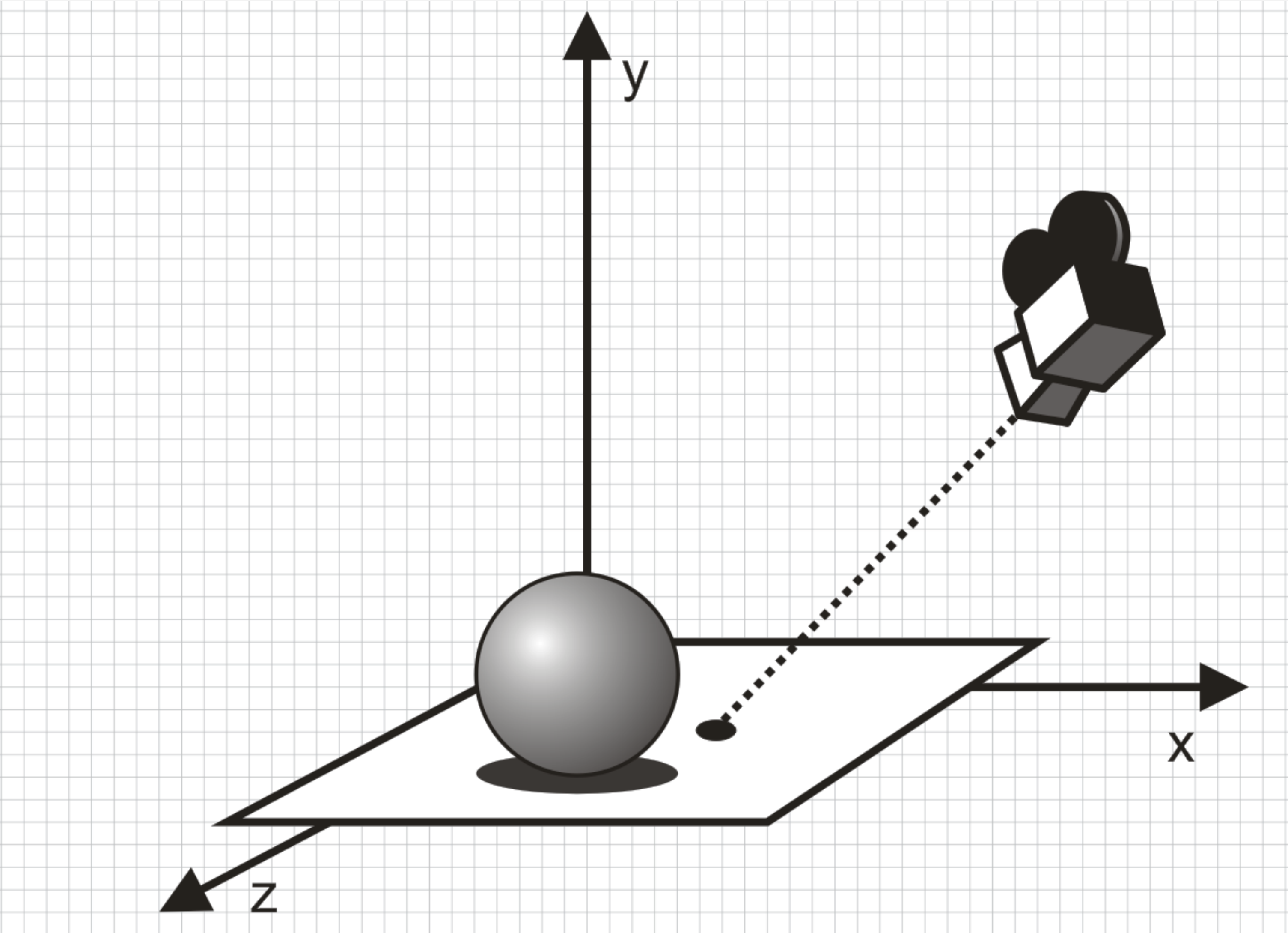
Summary continued

Camera coordinates

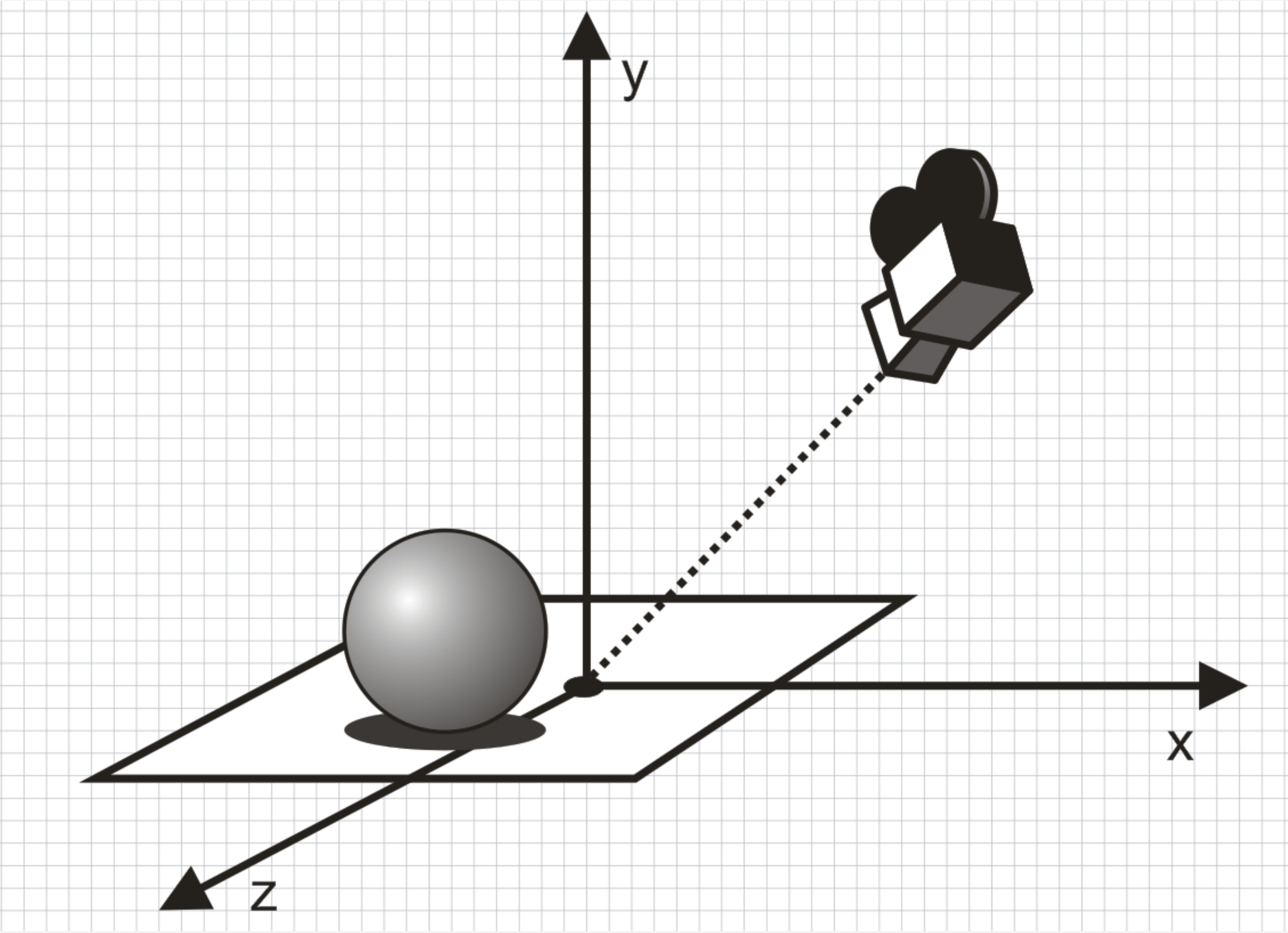
- XY of screen + Z as direction of view



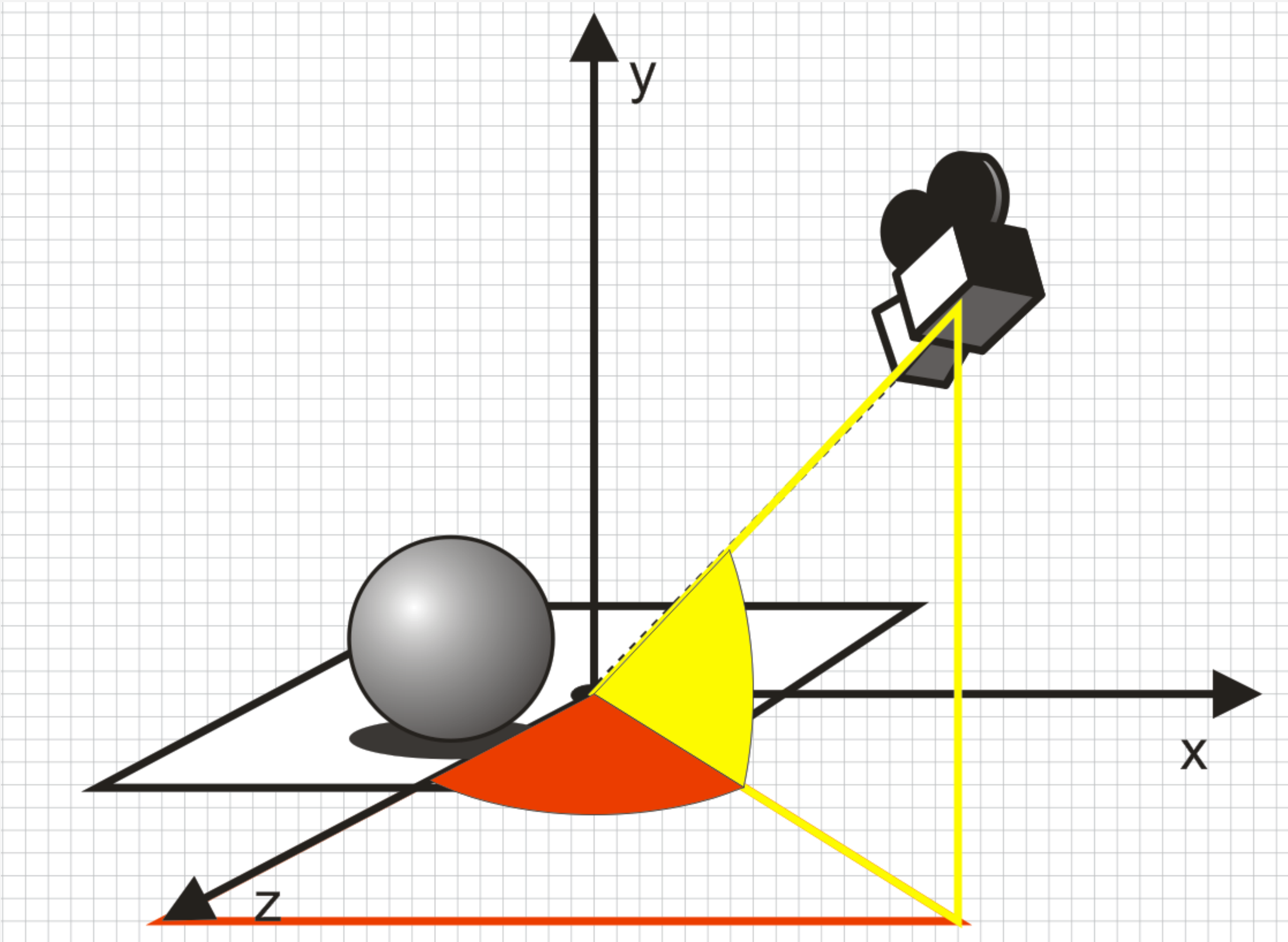
Stage 0



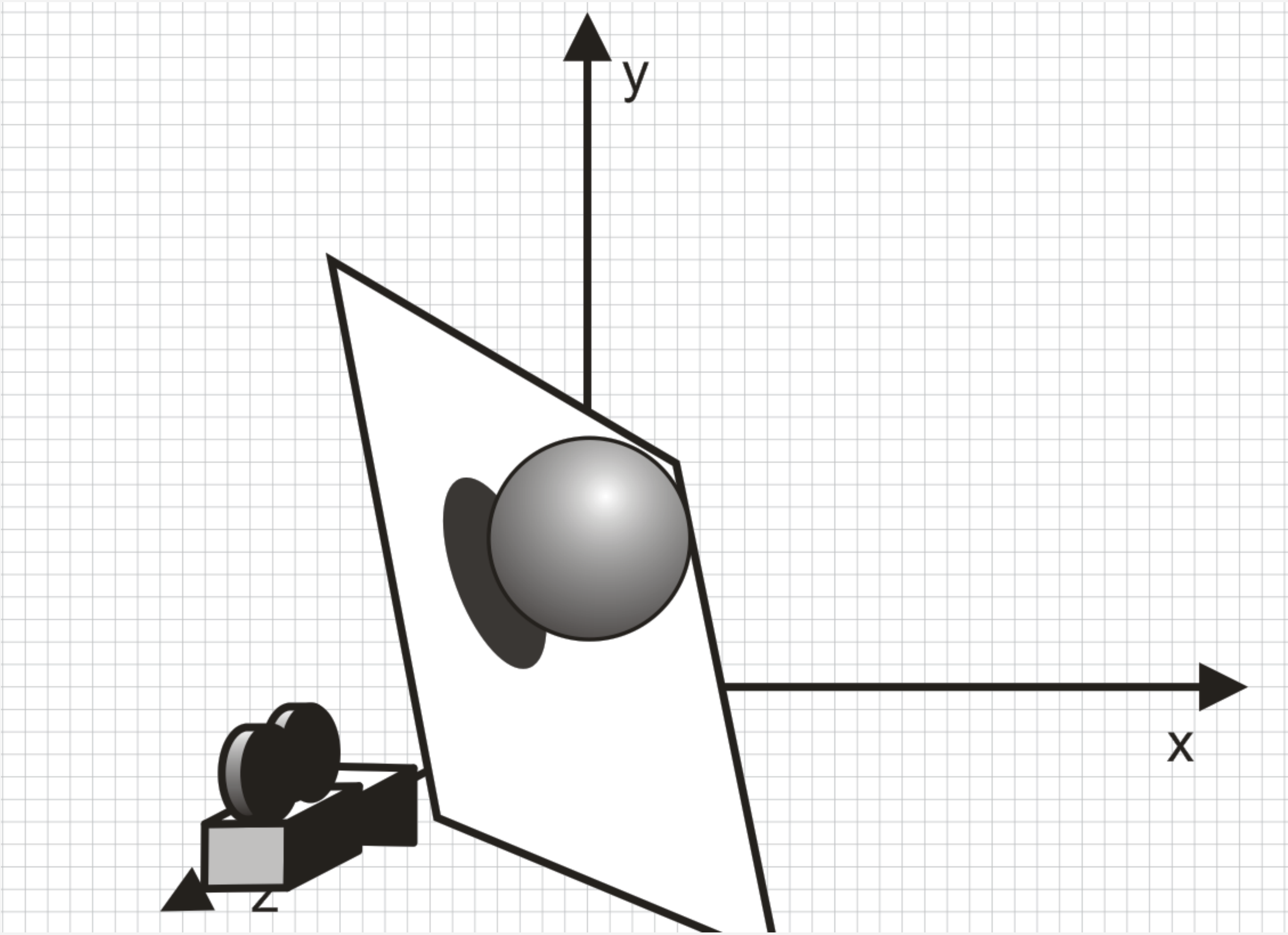
Stage 1 – translate P→P’



Stage 2 – rotate P'→P''→P'''



Rotated scene



Global→camera coordinates

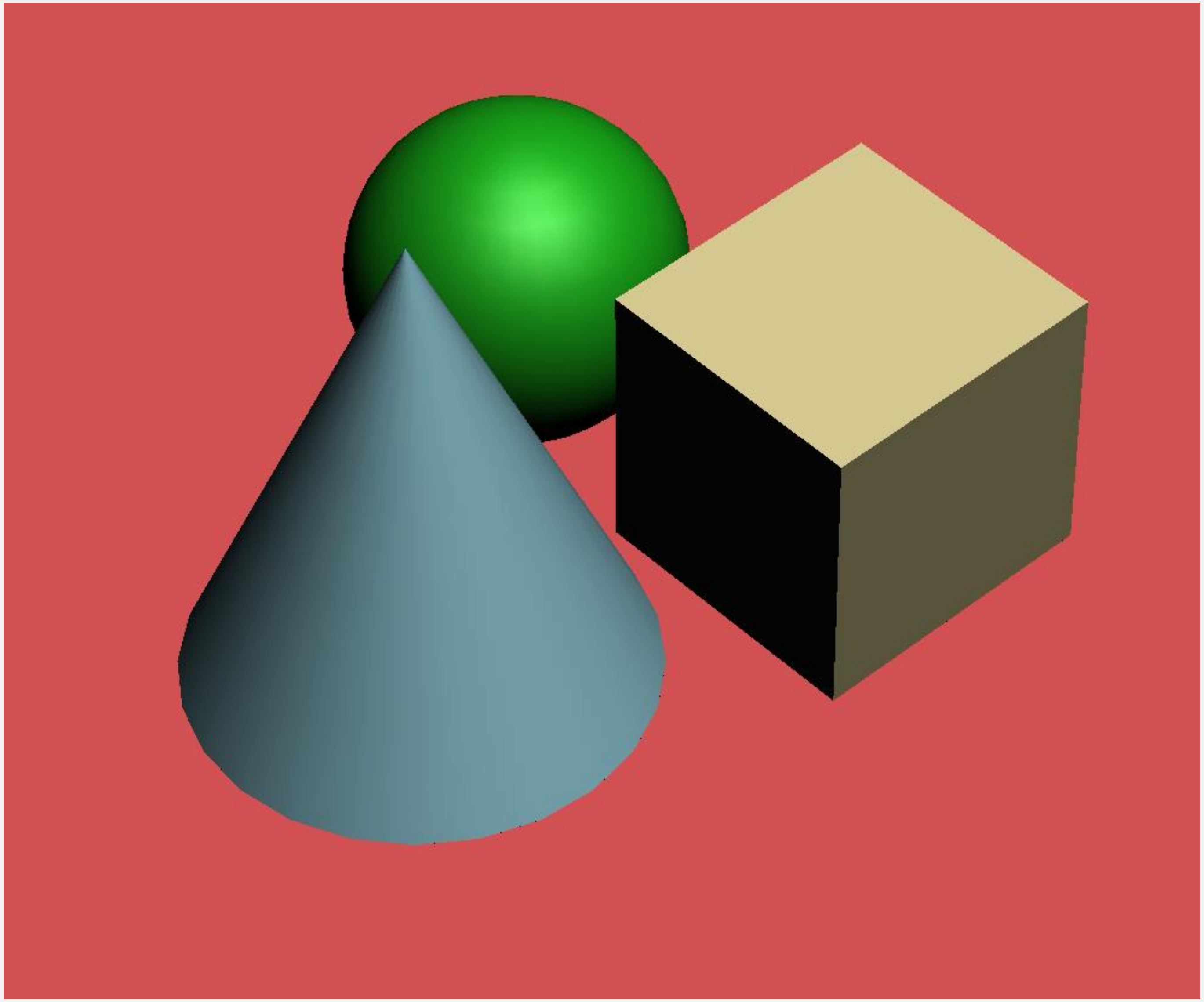
- $T * R_y * R_x$
  - Translation, rotation, rotation, projection
- $T * R_y * R_x * R_z$
  - if the camera is rolled

- Projection P
  - orthogonal, perspective, isometric ...

Projection types

- Orthogonal

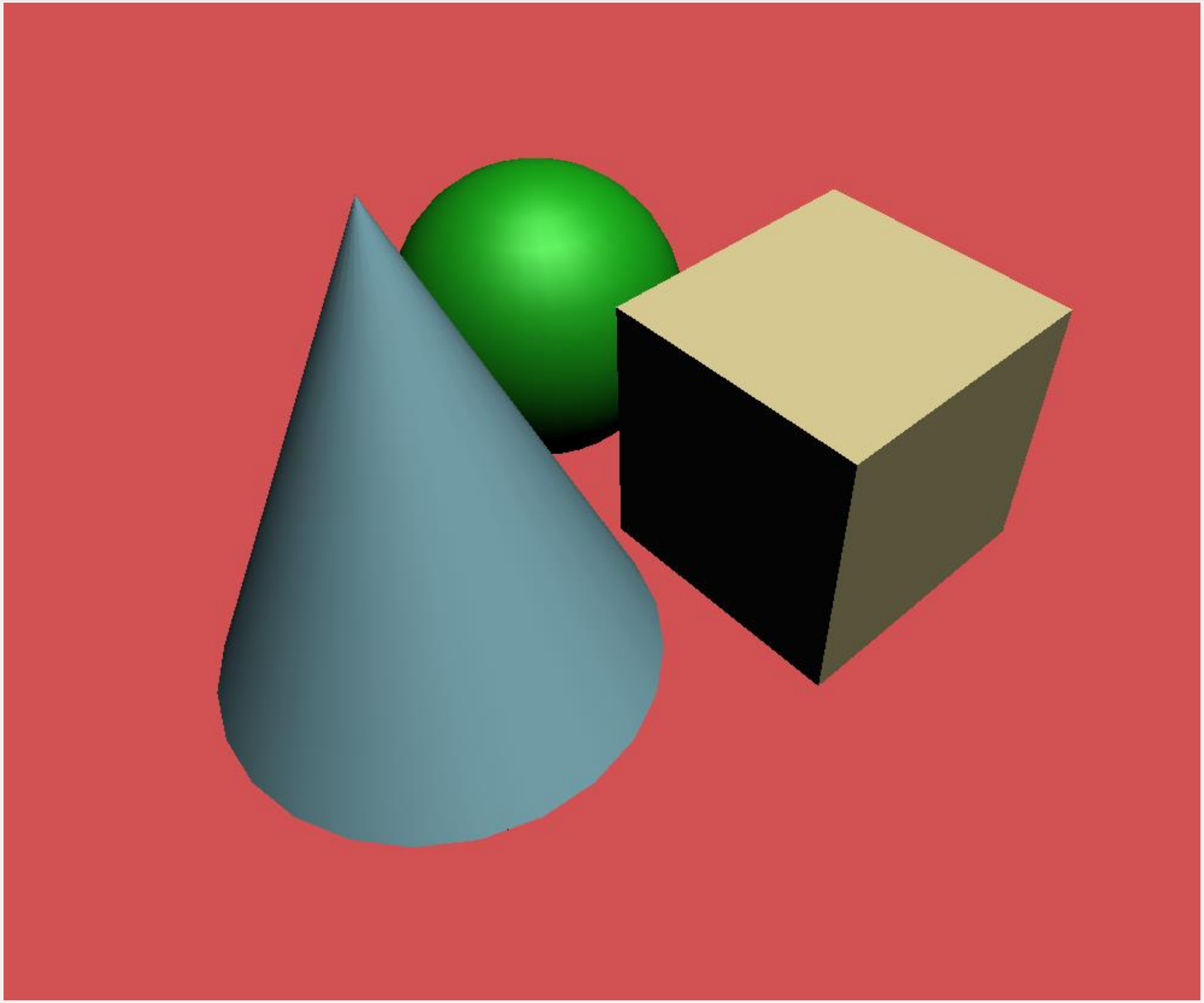
Projection types – parallel



Projection types – parallel

- Isometric (parallel but not orthogonal)

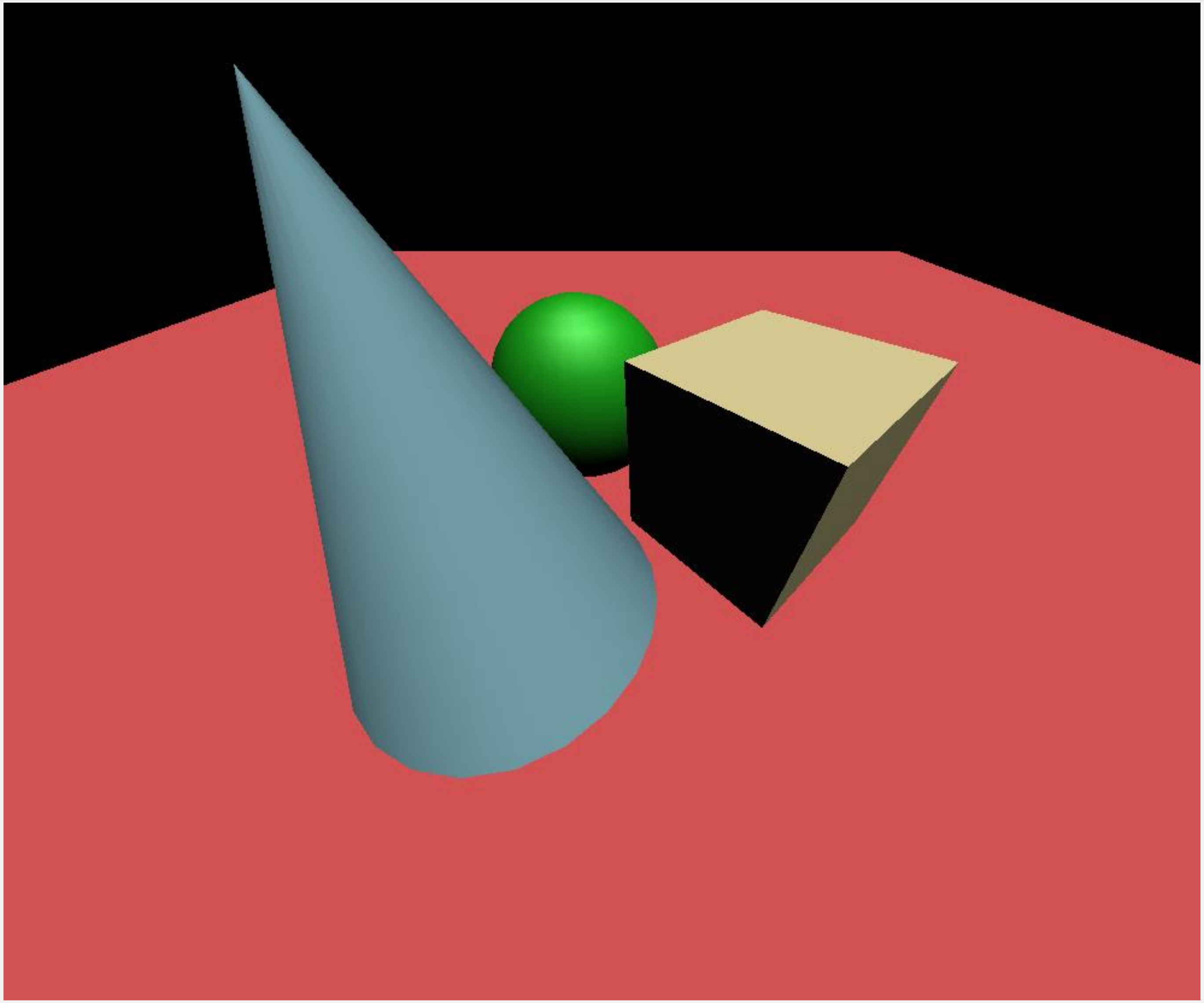
Projection – perspective



Projection types

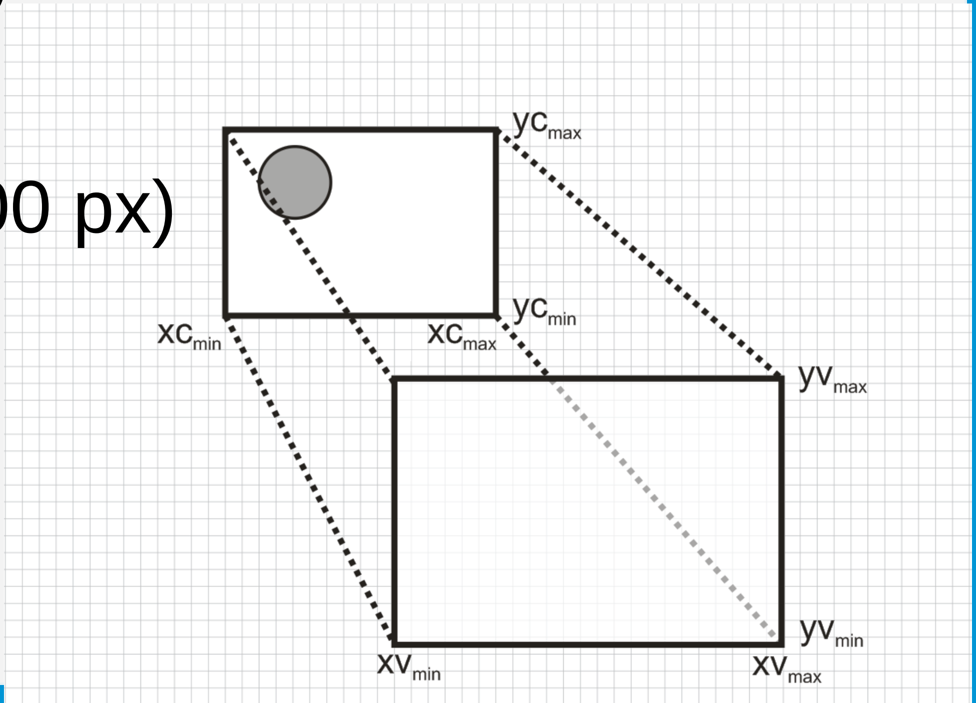
- Perspective

Distorted perspective



Viewport transformation

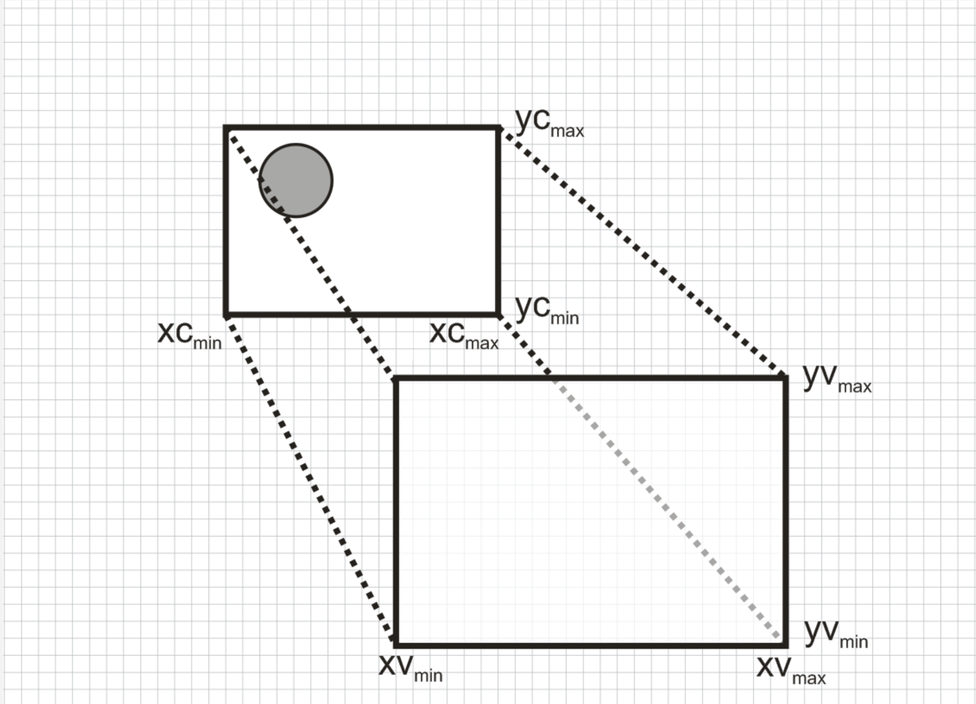
- Global coordinates
  - e.g. (-50..50 cm, -50..50 cm, -50..50 cm)
- Camera coordinates
  - e.g. (-1..1, -1..1, -1..1)
- Viewport (window)
  - e.g. (0..1200 px, 0..800 px)



Viewport transformation

$$s_x = \frac{xv_{max} - xv_{min}}{xc_{max} - xc_{min}}$$

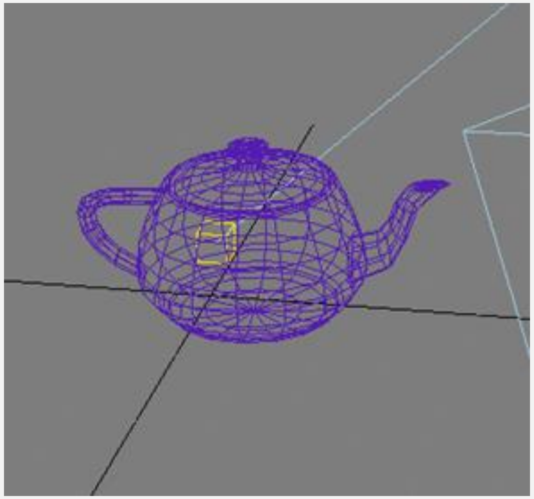
$$s_y = \frac{yv_{max} - yv_{min}}{yc_{max} - yc_{min}}$$



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

Rendering pipeline

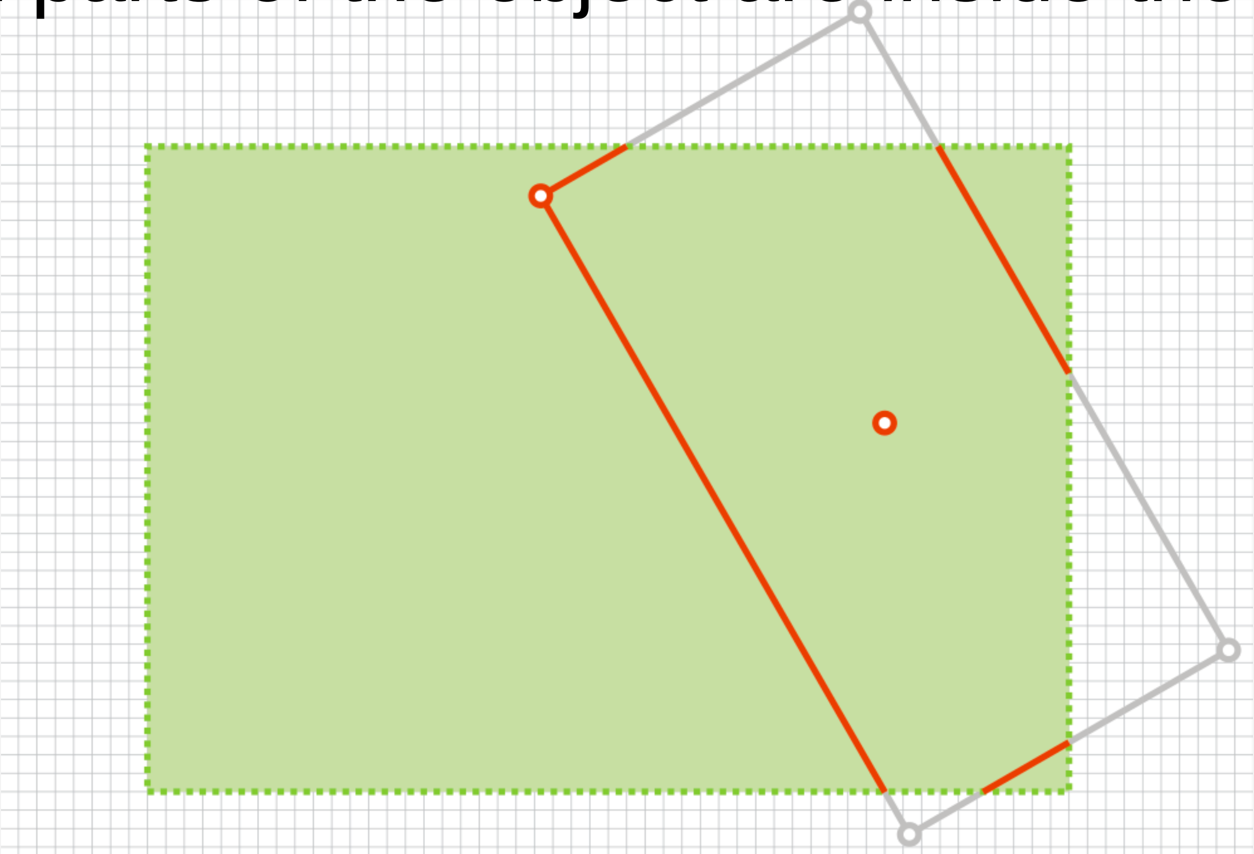
- Model transformation
  - local → global coordinates
- View transformation
  - global → camera
- Projection transformation
  - camera → screen
- Clipping, rasterization, texturing & Lighting
  - might take place earlier



Clipping

General problem:

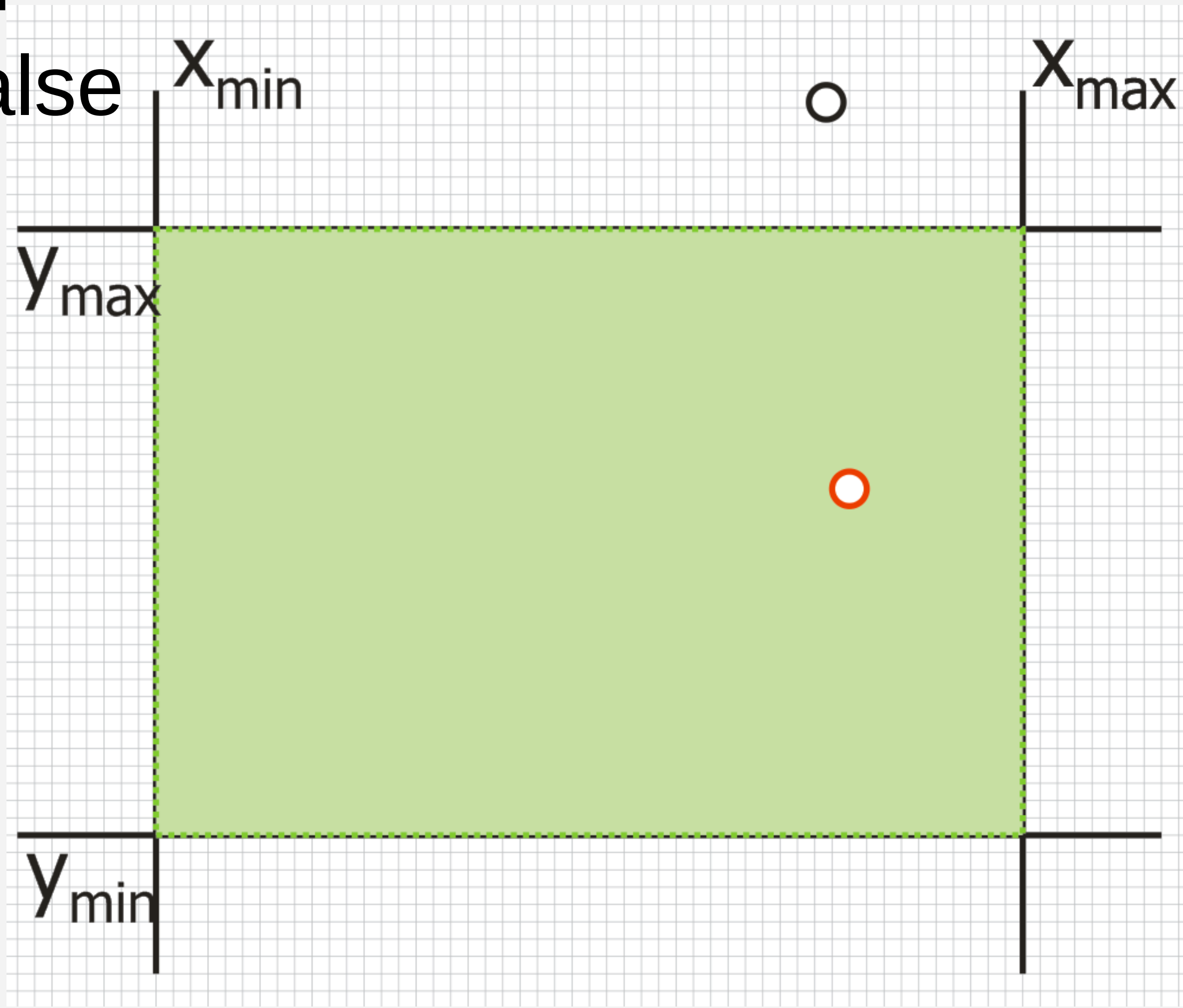
- Which parts of the object are inside the view



- Points, lines, polygons, text

Point clipping

- Trivial – 4 comparisons
- Result: true / false



- $x_{min} < x < x_{max}$
- $y_{min} < y < y_{max}$

Line clipping

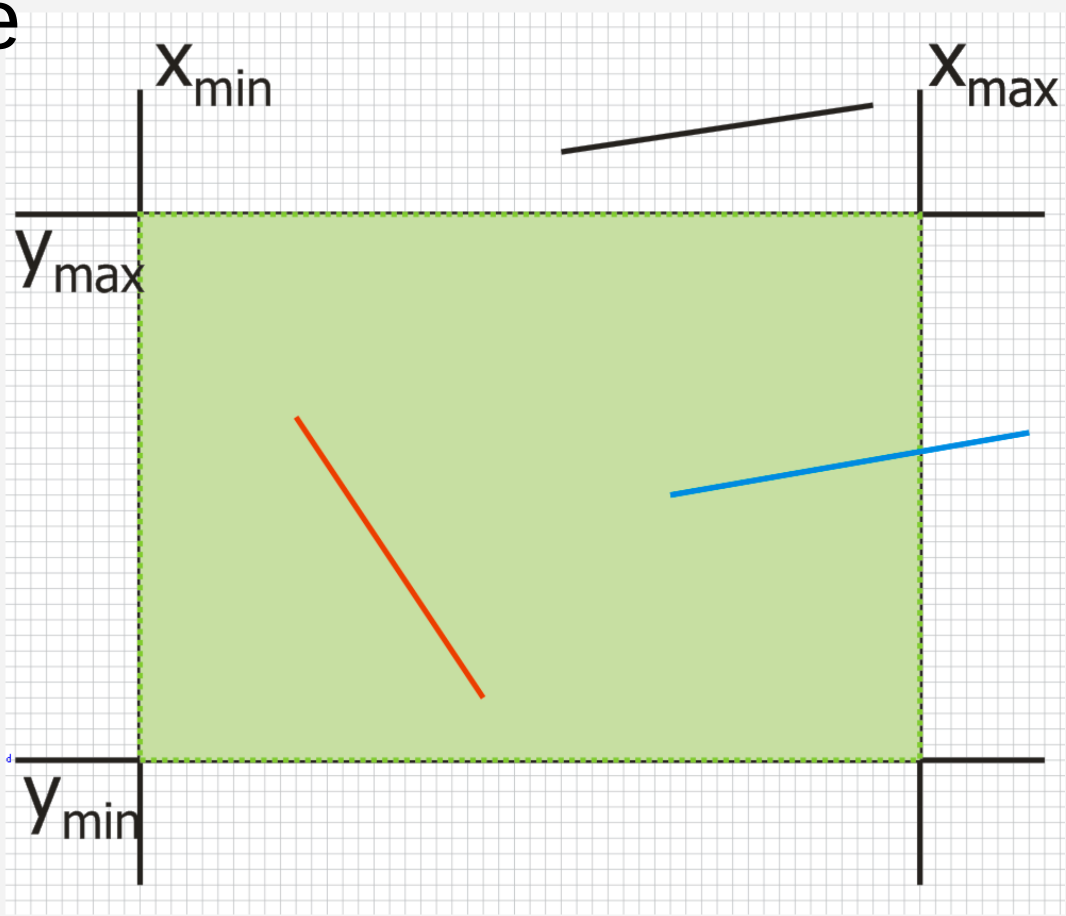
- 2 trivial cases

a)whole line outside

b)whole line inside

- non-trivial case

c)line partly inside

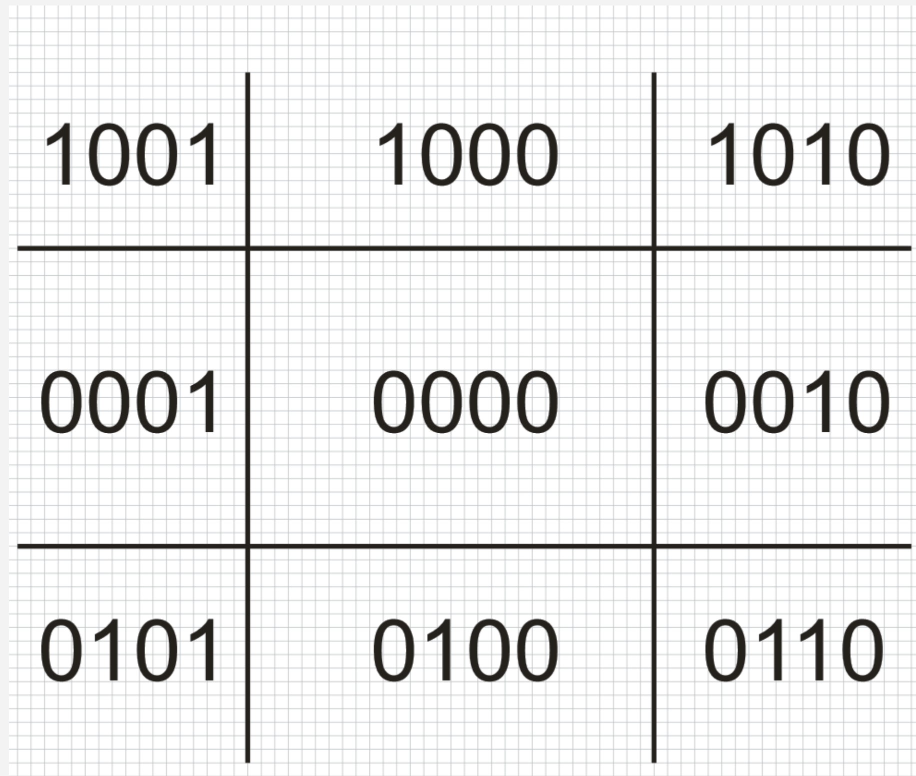


Cohen-Sutherland

- 4 bits code for each endpoint

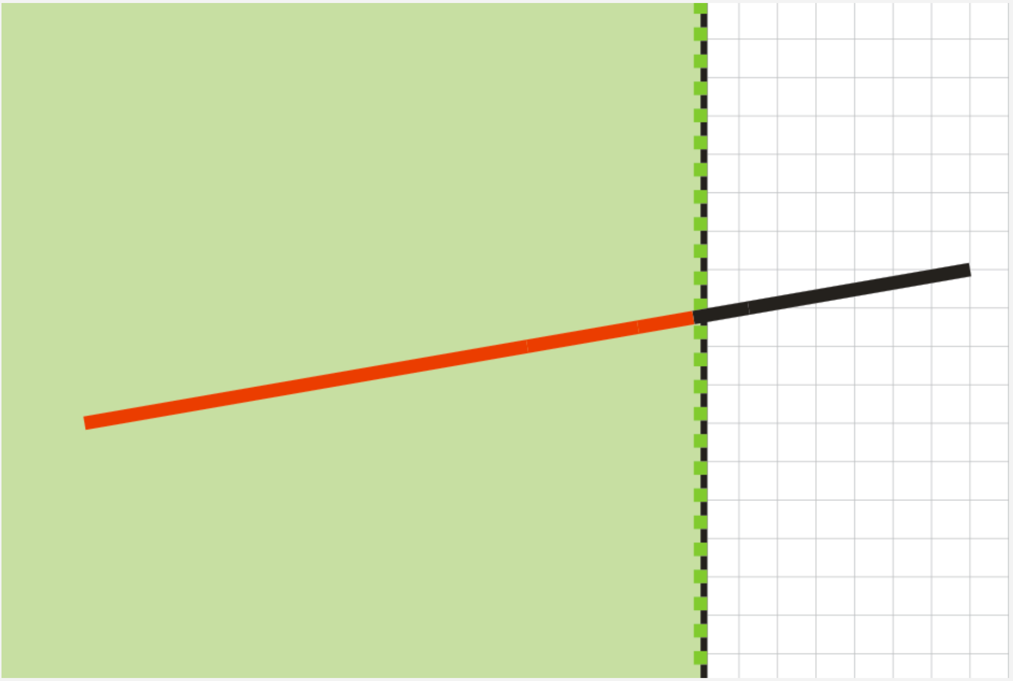
$y > y_{max}$	$y < y_{min}$	$x > x_{max}$	$x < x_{min}$

- bitwise OR == 0
  - – whole line inside
- bitwise AND != 0
  - – whole line outside
- otherwise
  - – line partially inside



Line partially inside

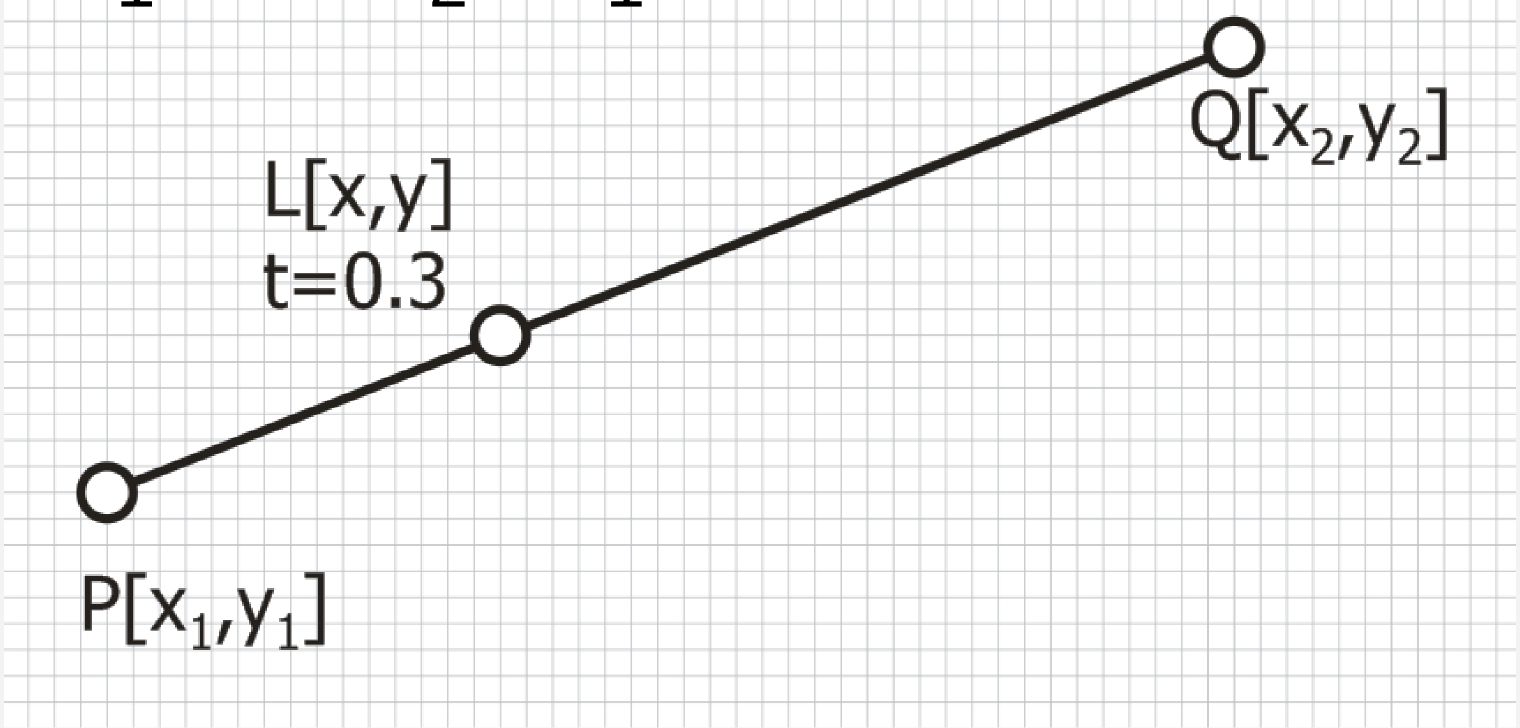
1. split into segments
2. test segments for trivial cases
   a) if segment inside – draw it
   b) if segment outside – reject it
   c) if non-trivial case – repeat recursively from 1



Parametric line equation

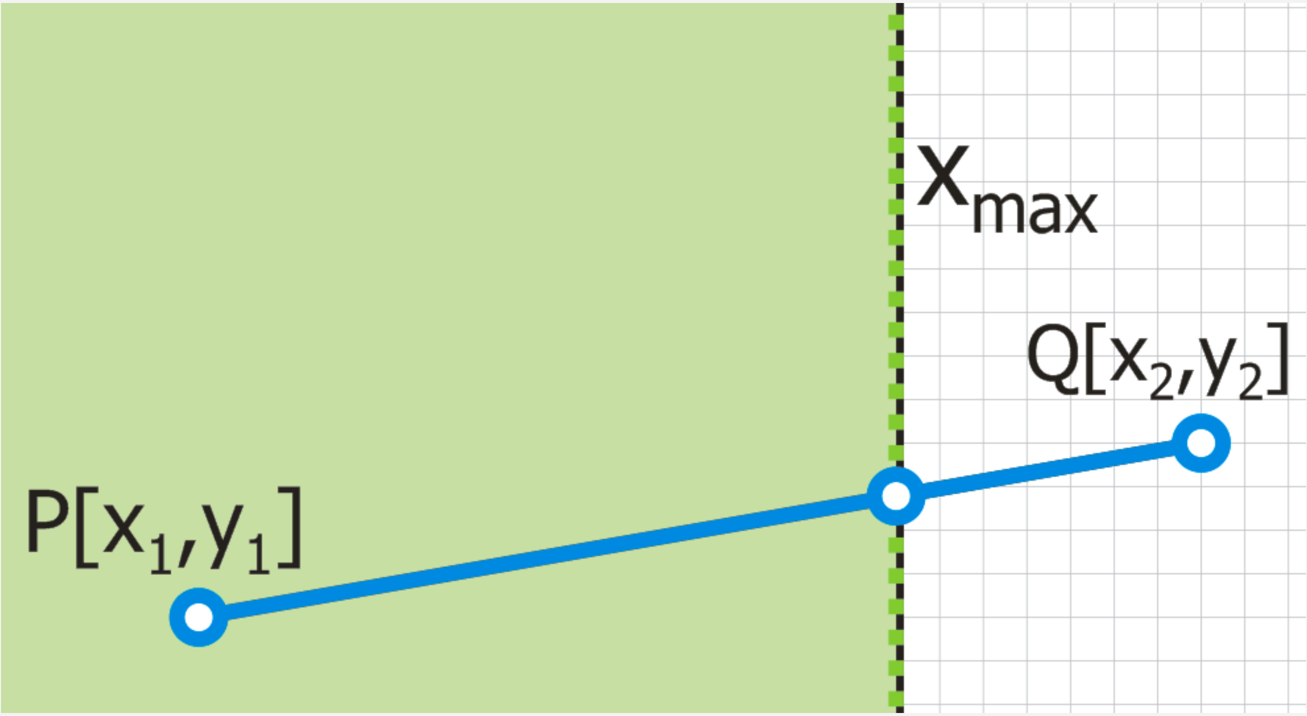
- Line **P-Q** where P = $[x_1,y_1]$, Q = $[x_2,y_2]$
- $x = x_1 + t * (x_2 – x_1)$
- $y = y_1 + t * (y_2 – y_1)$

} **L** = **P** + t*(**Q**-**P**)

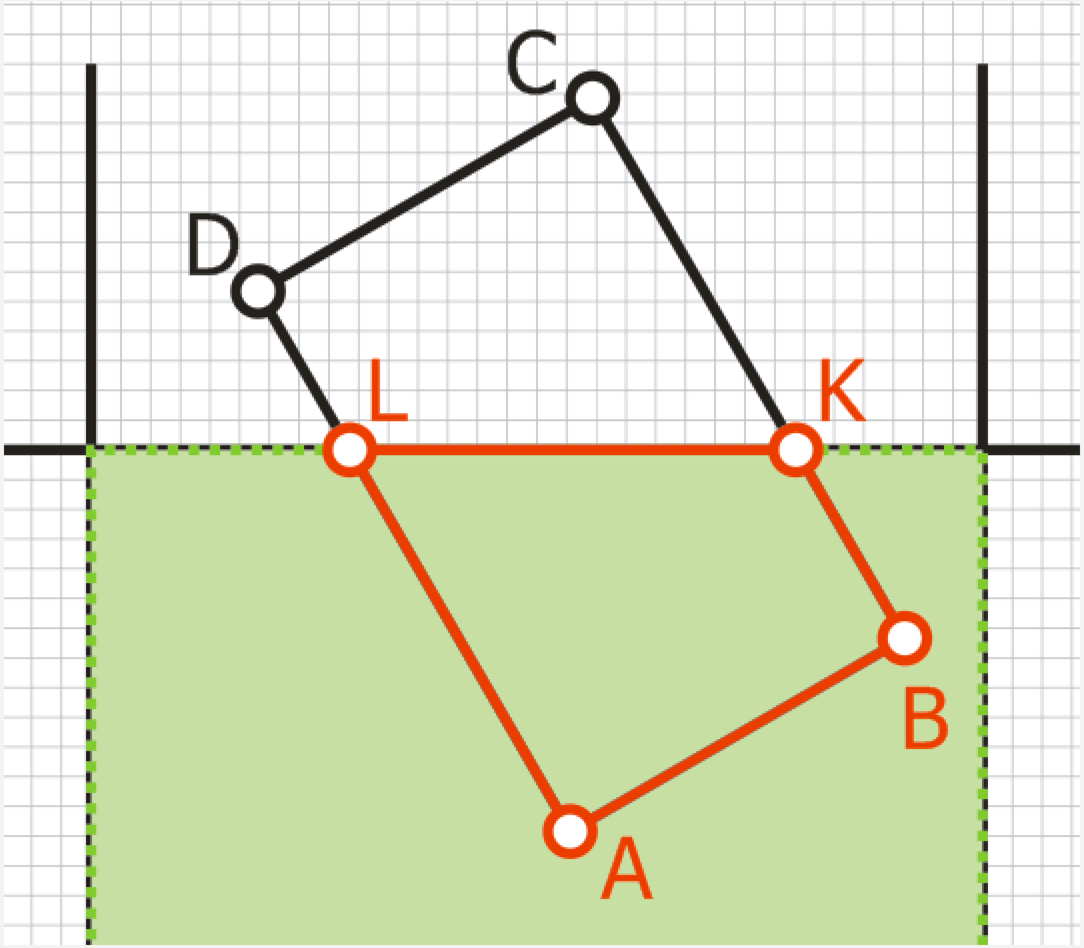


Line-edge intersection

- Look for t
- $t = (x - x_1)/(x_2 - x_1)$ where $x = x_{min}$ or x_{max}
- $t = (y - y_1)/(y_2 - y_1)$ where $y = y_{min}$ or y_{max}

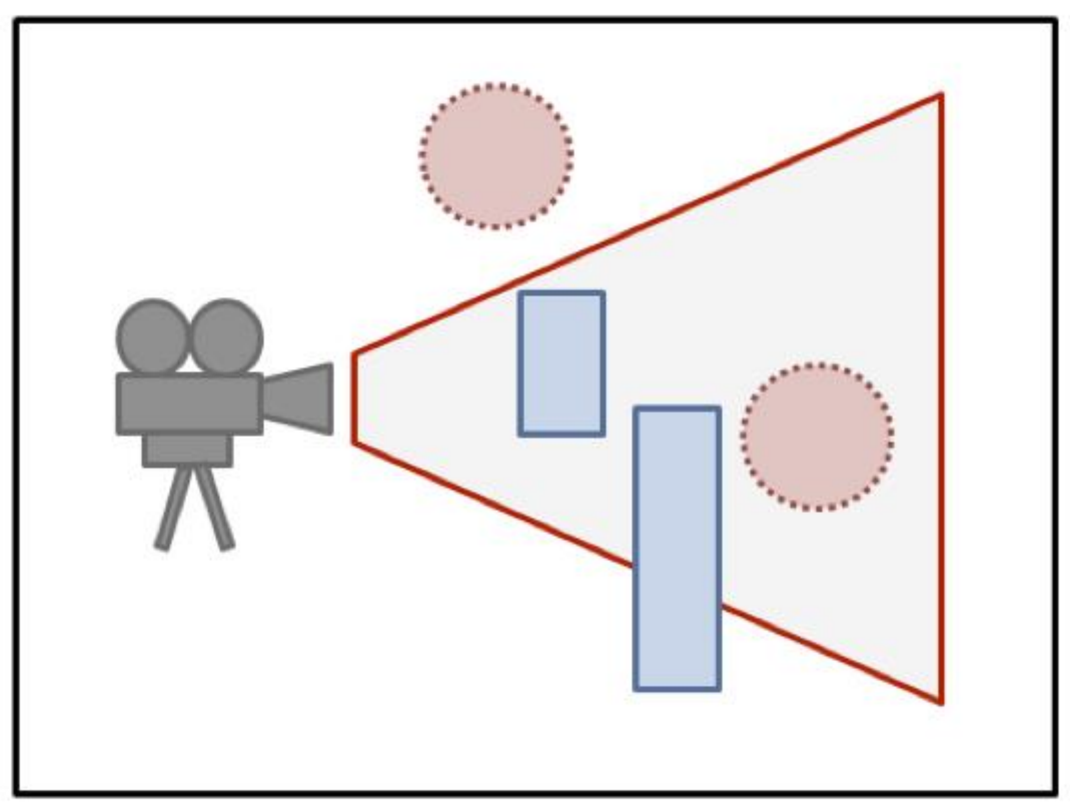


Polygon clipping



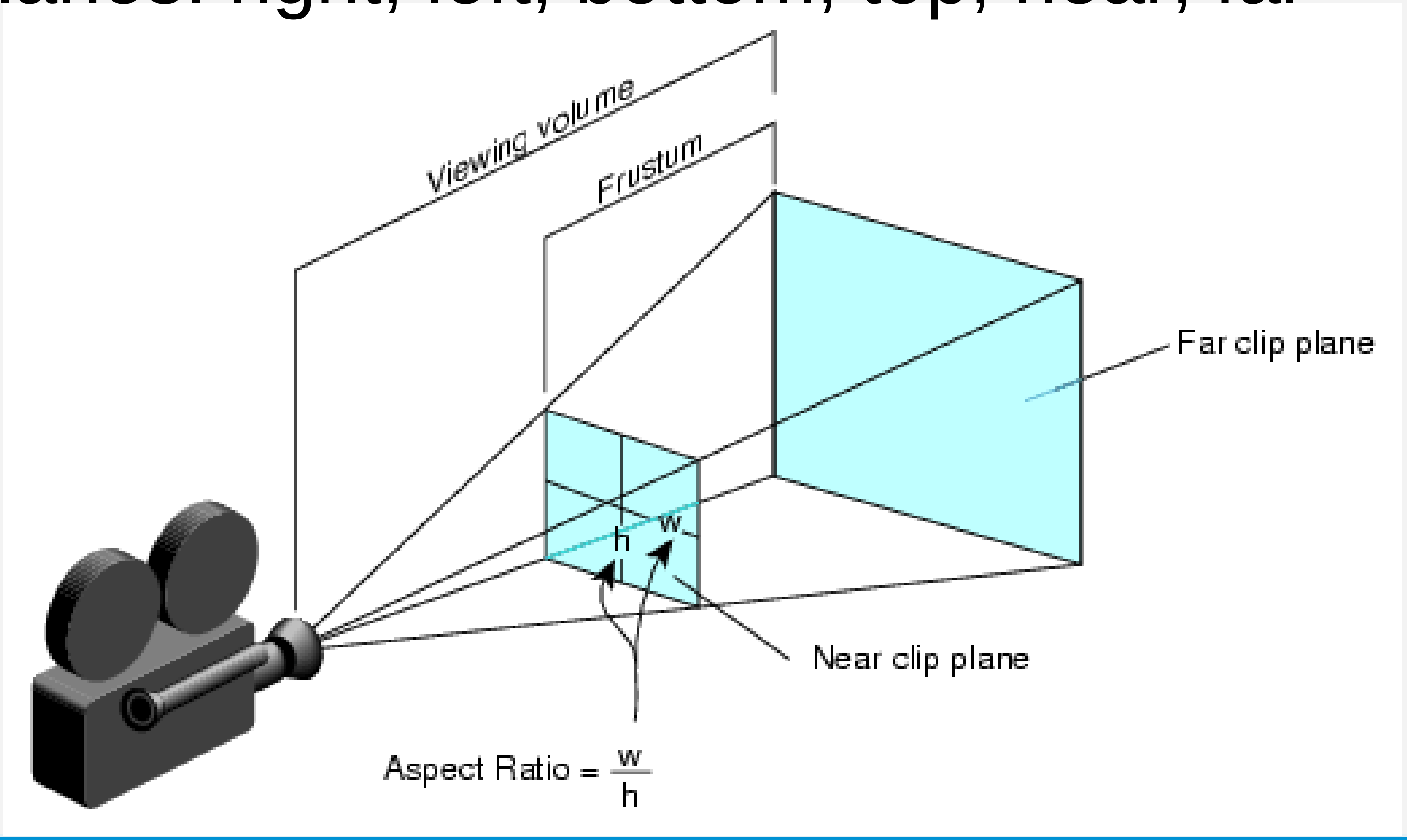
General problem in 3D:

- Which objects / object parts are visible?
- Objects outside the view can be ignored
- Speeding up the rendering



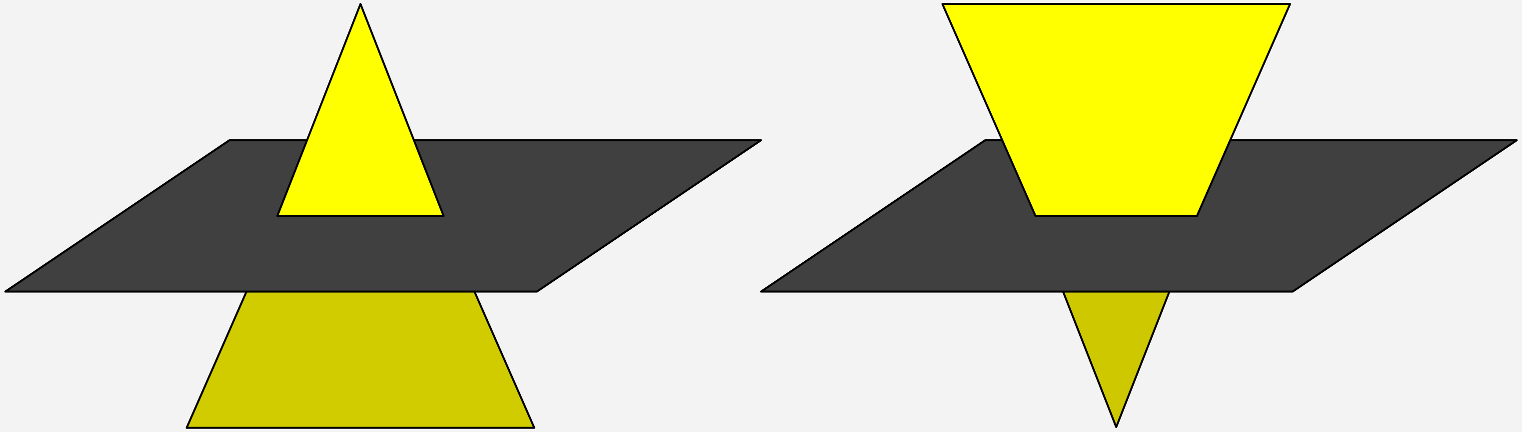
Clipping in 3D

- Viewing volume (or frustum)
- 6 planes: right, left, bottom, top, near, far



Clipping in 3D

- Usually the primitives are triangles
- Triangle-plane intersection
  - = 0 or 2 line-plane intersections



Line-plane intersection in 3D

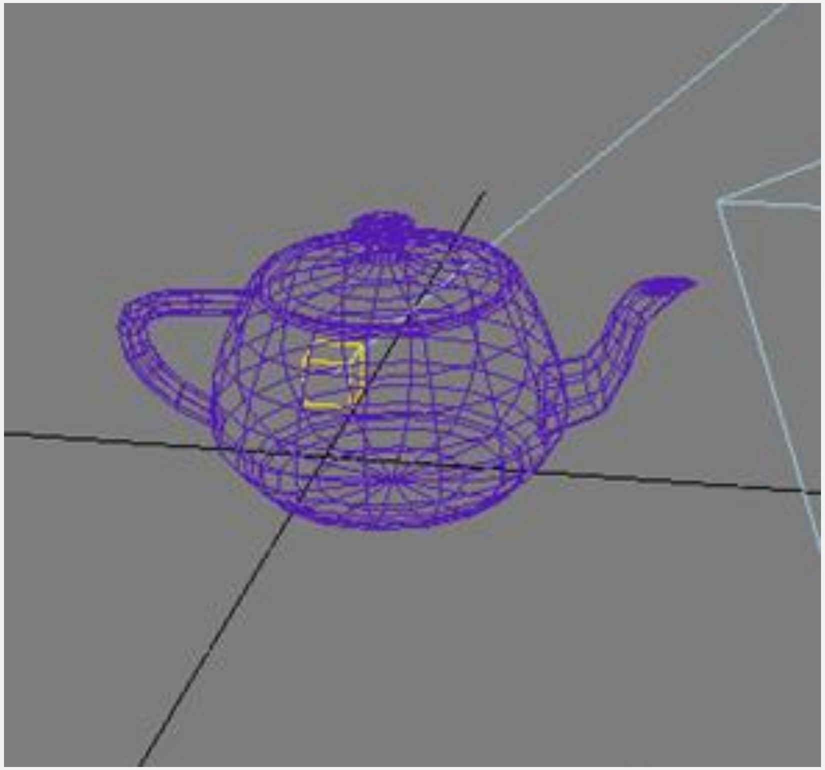
- Plane: $P = W + u(U - W) + v(V - W)$
- Line: $L = A + t(B - A)$
- Find t: $L = P$

$$A + t(B - A) = W + u(U - W) + v(V - W)$$

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \begin{pmatrix} A_x - B_x & U_x - W_x & V_x - W_x \\ A_y - B_y & U_y - W_y & V_y - W_y \\ A_z - B_z & U_z - W_z & V_z - W_z \end{pmatrix}^{-1} \begin{pmatrix} A_x - W_x \\ A_y - W_y \\ A_z - W_z \end{pmatrix}$$

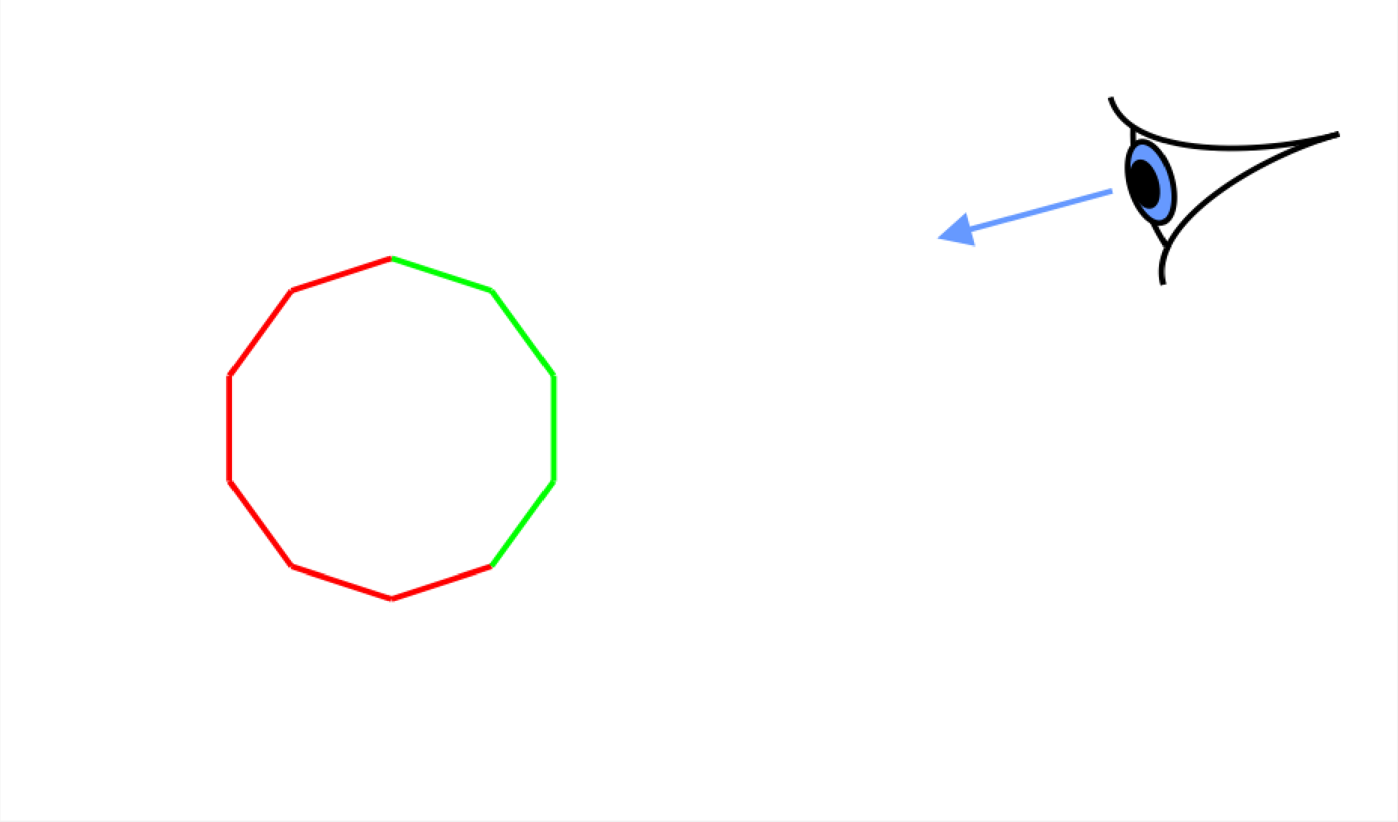
Back-face culling

- Parts of object not facing the camera are also invisible
  - Except for semi-transparency, mirrors etc.



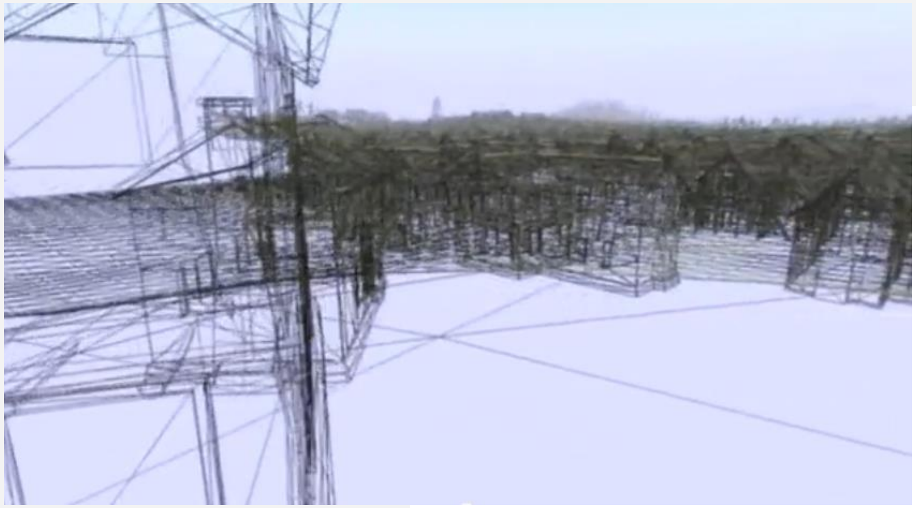
Backface culling

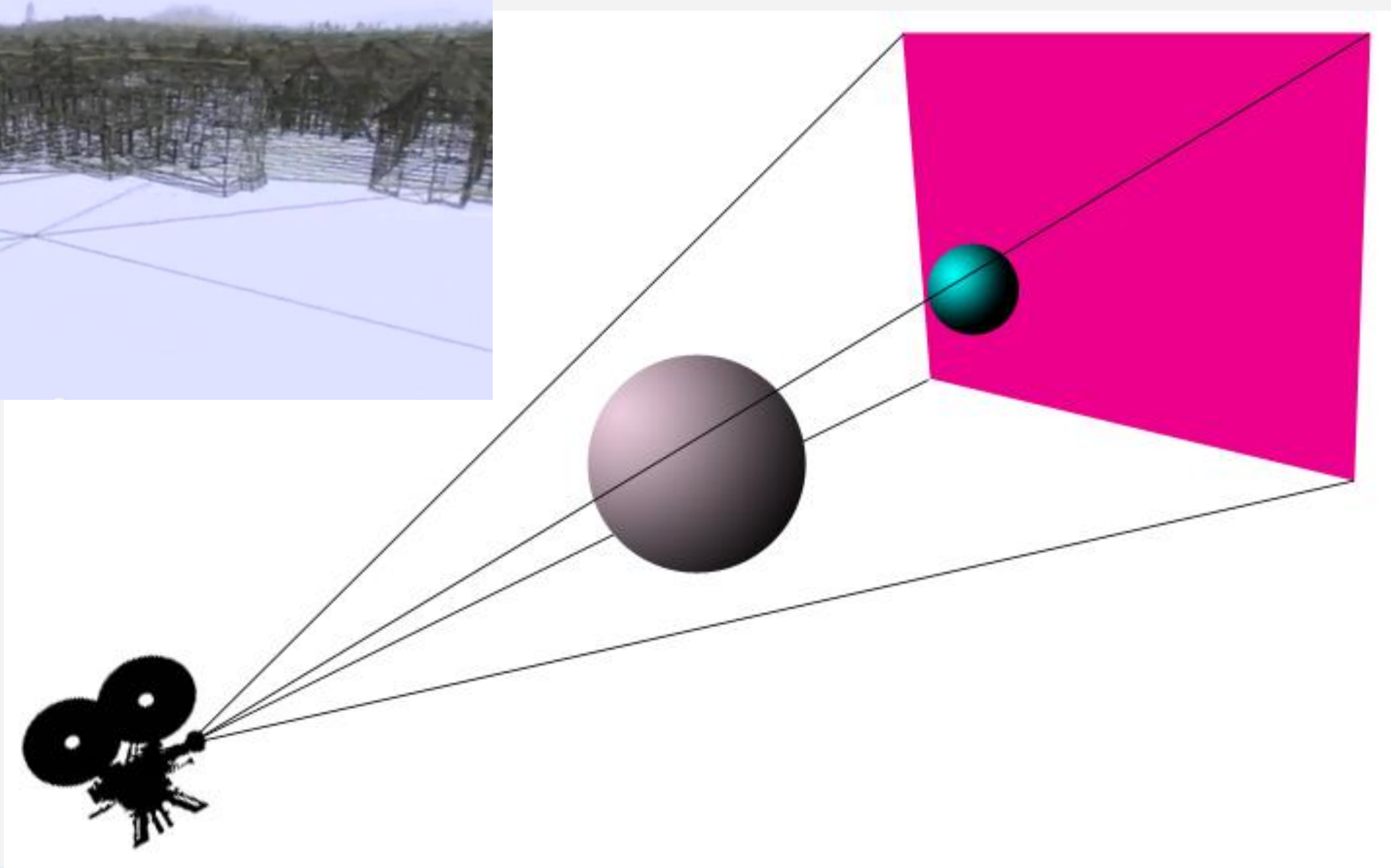
- Which object faces are visible?
- Remember normal vector (face orientation)



Occlusion culling

- Some objects are fully occluded by others





Portal culling

- Some parts of the scene are not visible from some other parts of the scene