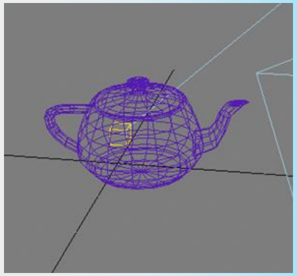
# RENDERING PART ONE

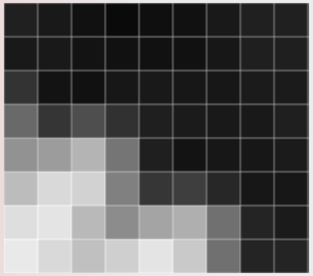
# RECOLLECTION

## CG REFERENCE MODEL

Application program → Graphical system → Output device

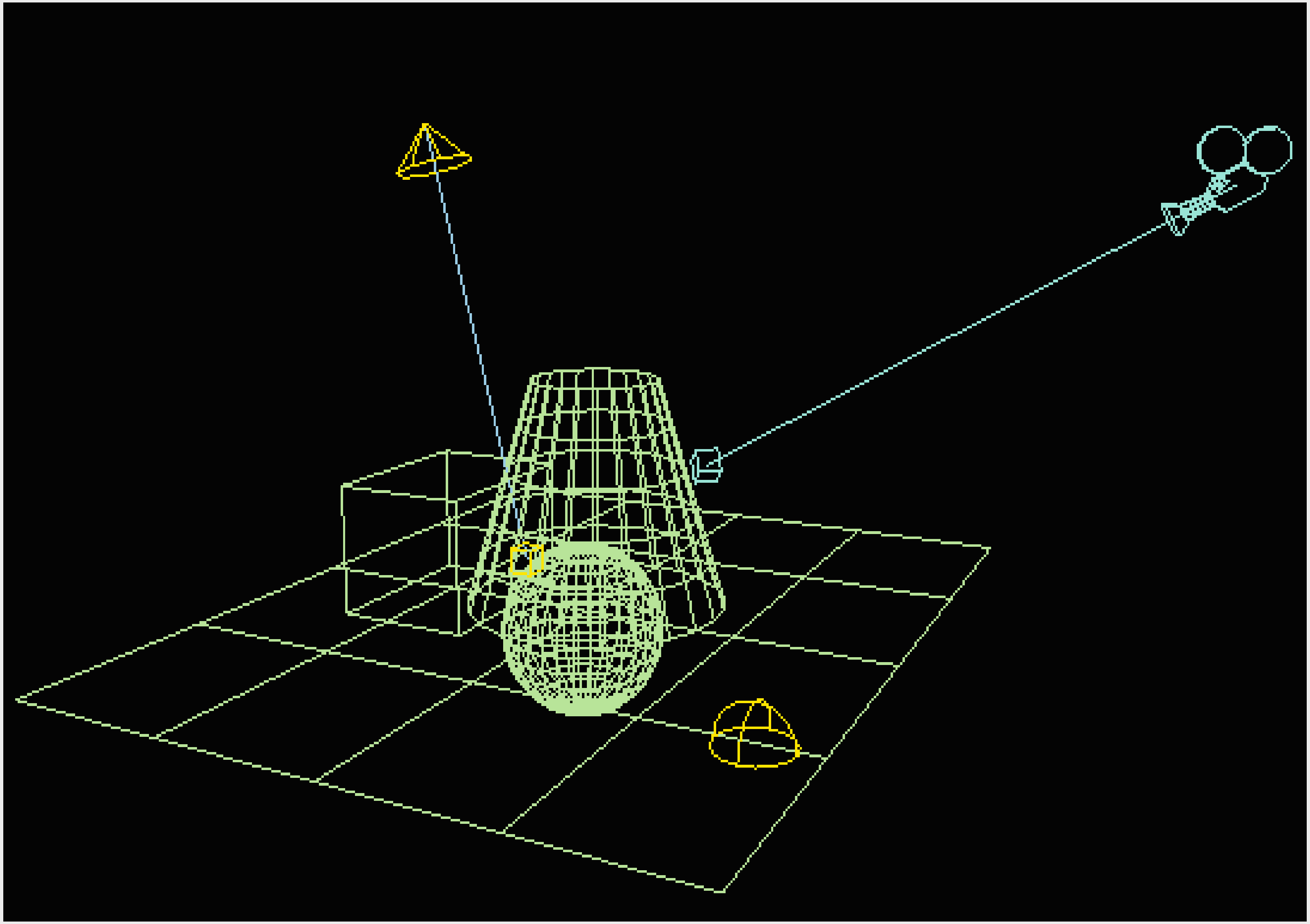


Geometry space – 3D



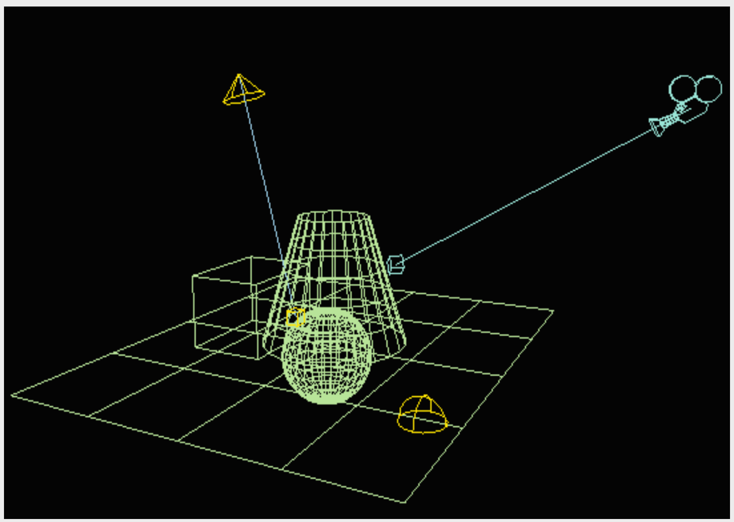
Screen space – 2D

# GEOMETRY SPACE – 3D SCENE

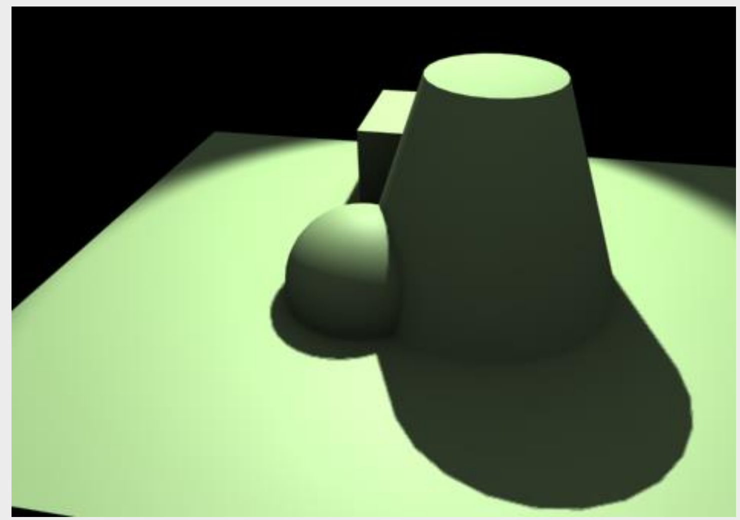


# GEOMETRY VS. SCREEN SPACE

3D
CONTINUOUS
PARAMETRIC
MODELS



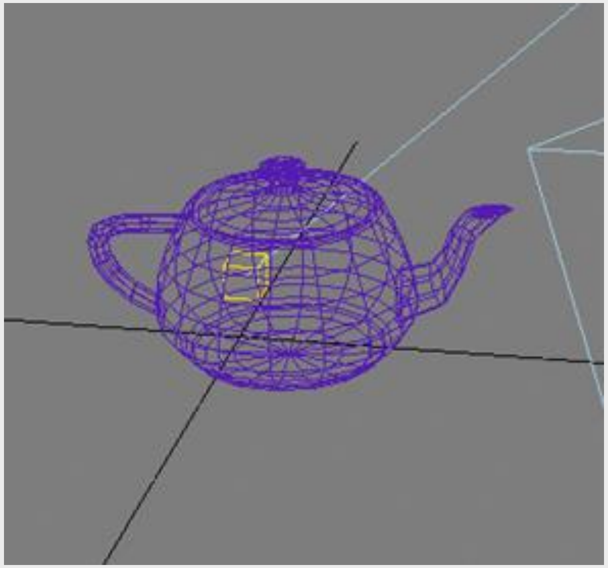
2D
DISCRETE
NON-PARAMETRIC
PIXELS



# RENDERING

MODELS ⟶ PIXELS

GEOMETRY TRANSFORMATIONS
CLIPPING
RASTERIZATION
TEXTURING
+ LIGHTS & SHADOWS



IT HURTS
BUT IT HAS
TO BE DONE

# ESSENTIAL GEOMETRY

# COORDINATE SYSTEM IN 3D

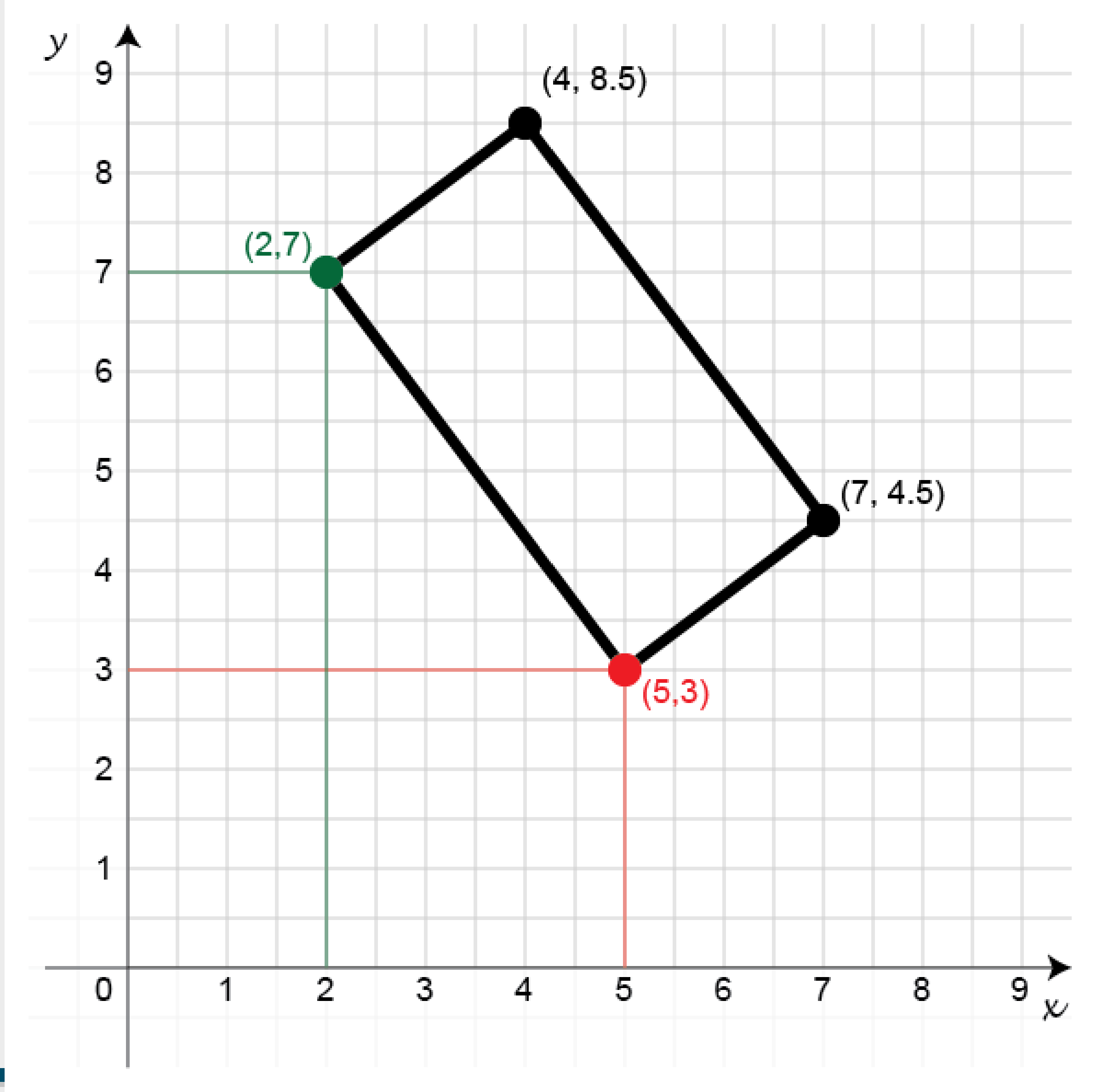
CARTESIAN COORDINATES IN 2D

Origin (0)

x axis

y axis

ORTHOGONAL



# COORDINATE SYSTEM IN 3D

## CARTESIAN COORDINATES IN 3D

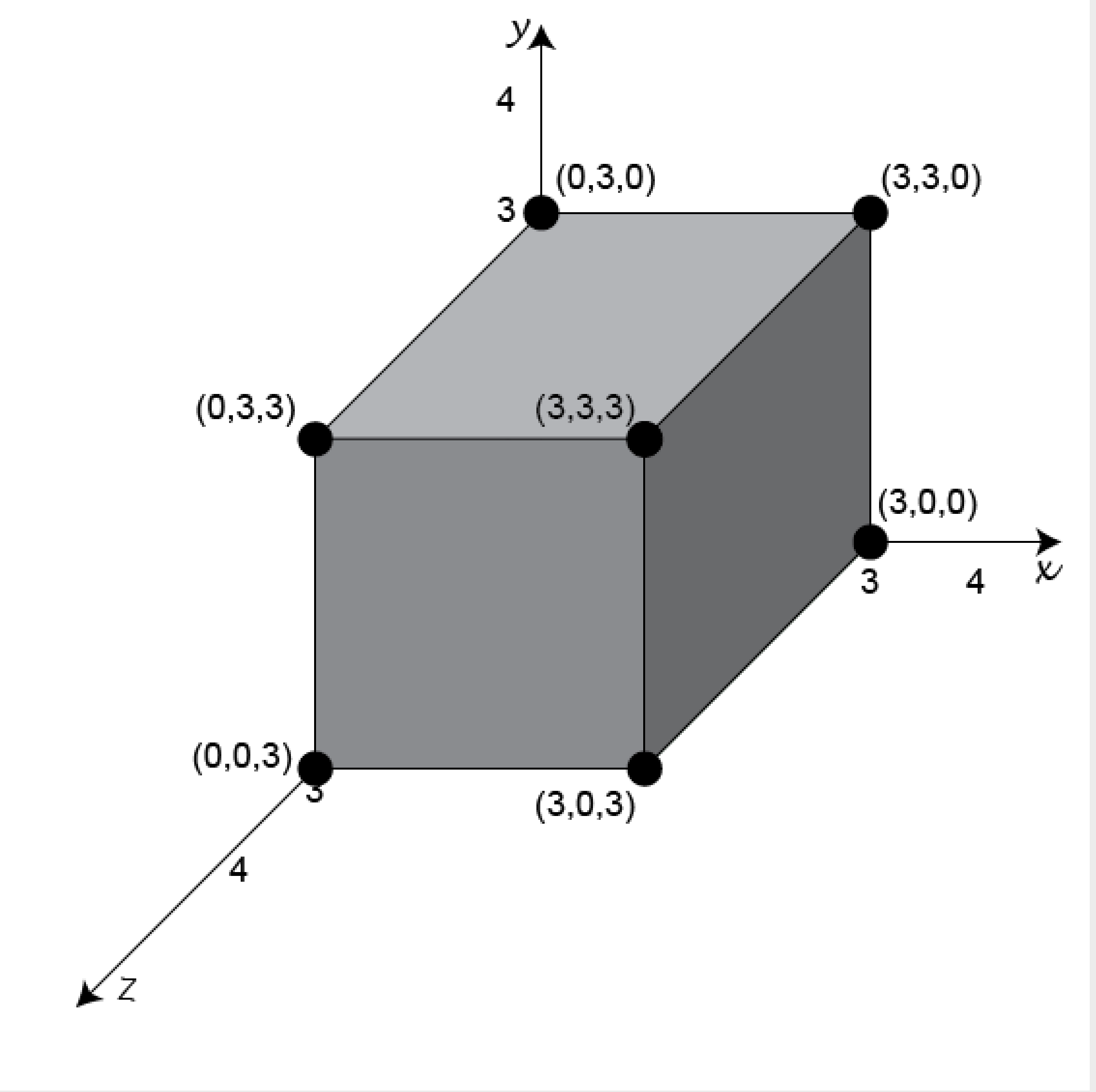
Origin (0)

x axis

y axis

z axis

## ORTHOGONAL

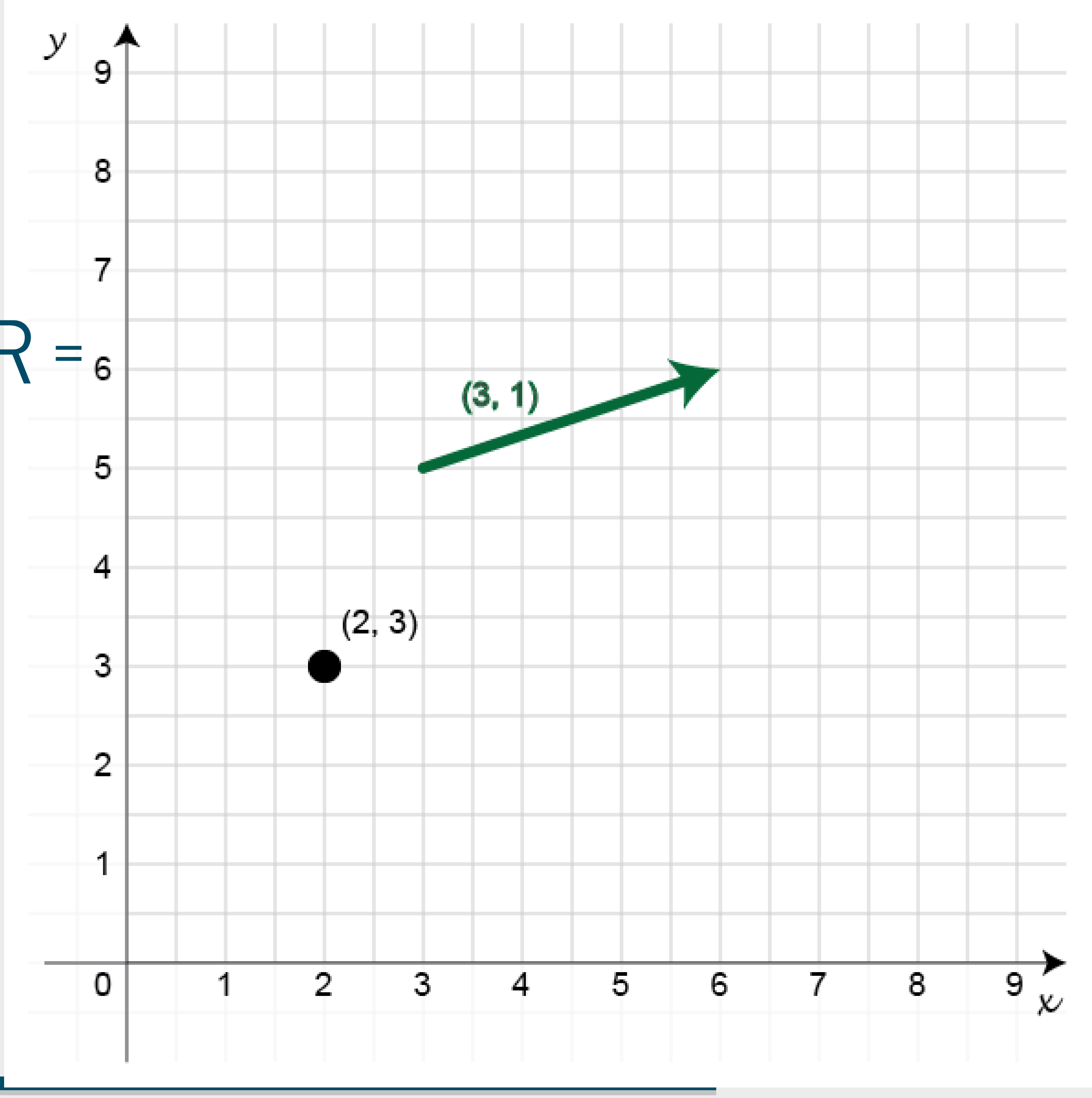


# POINTS & VECTORS

POINT + VECTOR = POINT

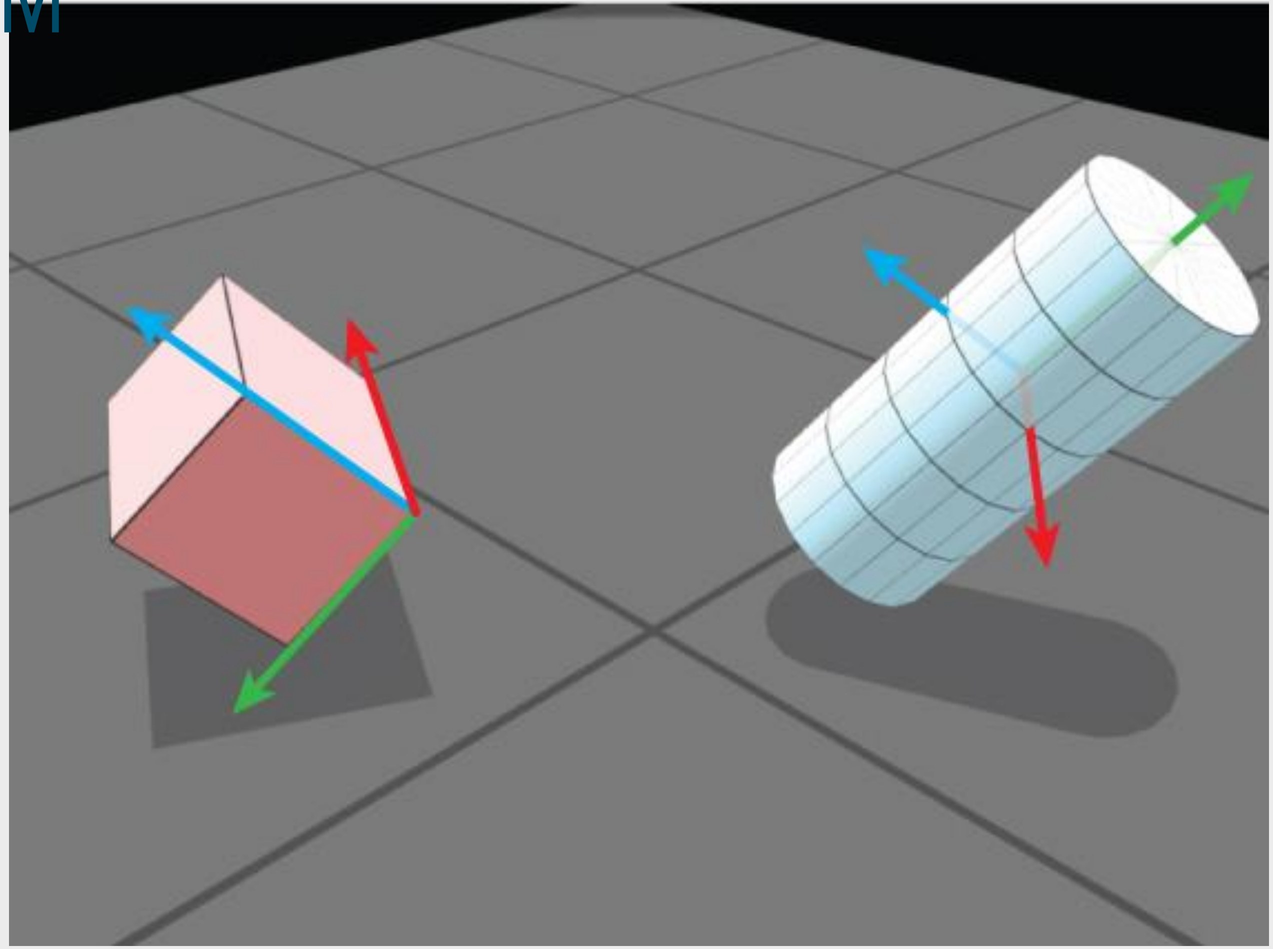
VECTOR + VECTOR = VECTOR

POINT - POINT = VECTOR



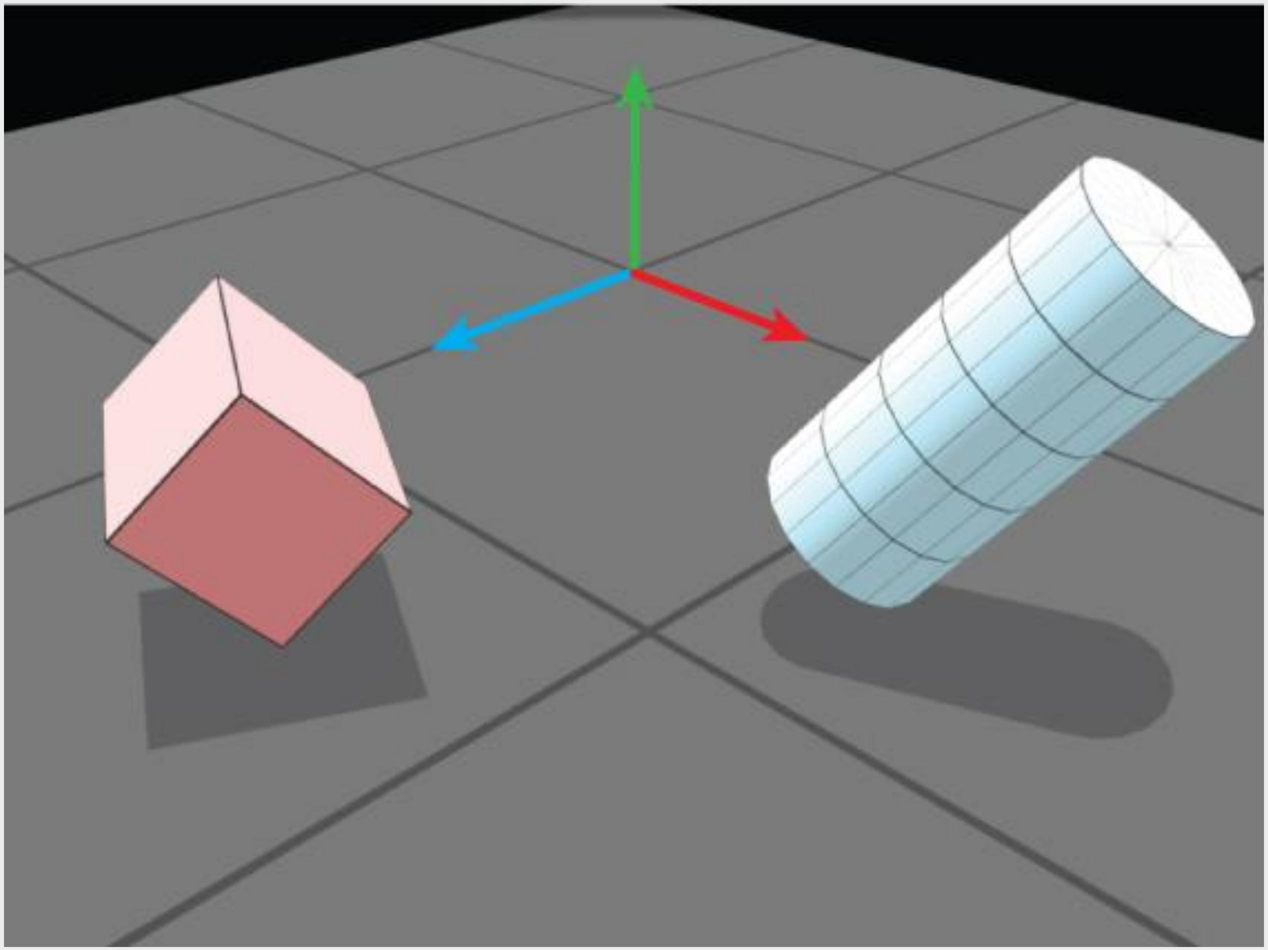
# LOCAL COORDINATES

EACH OBJECT HAS ITS OWN COORDINATE SYSTEM



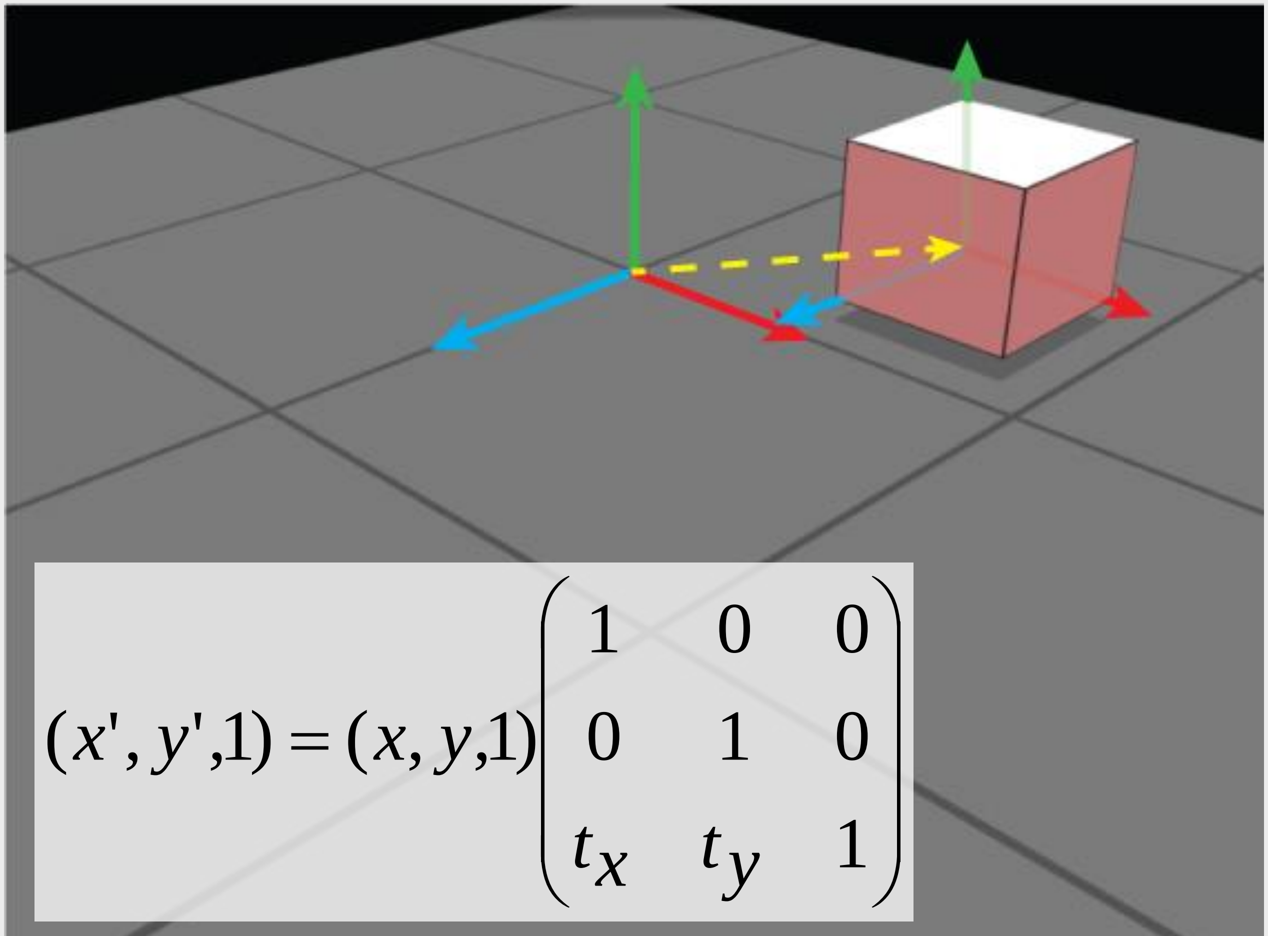
# GLOBAL COORDINATES

## ONE SYSTEM FOR THE WHOLE SCENE



# LOCAL → GLOBAL COORDINATES

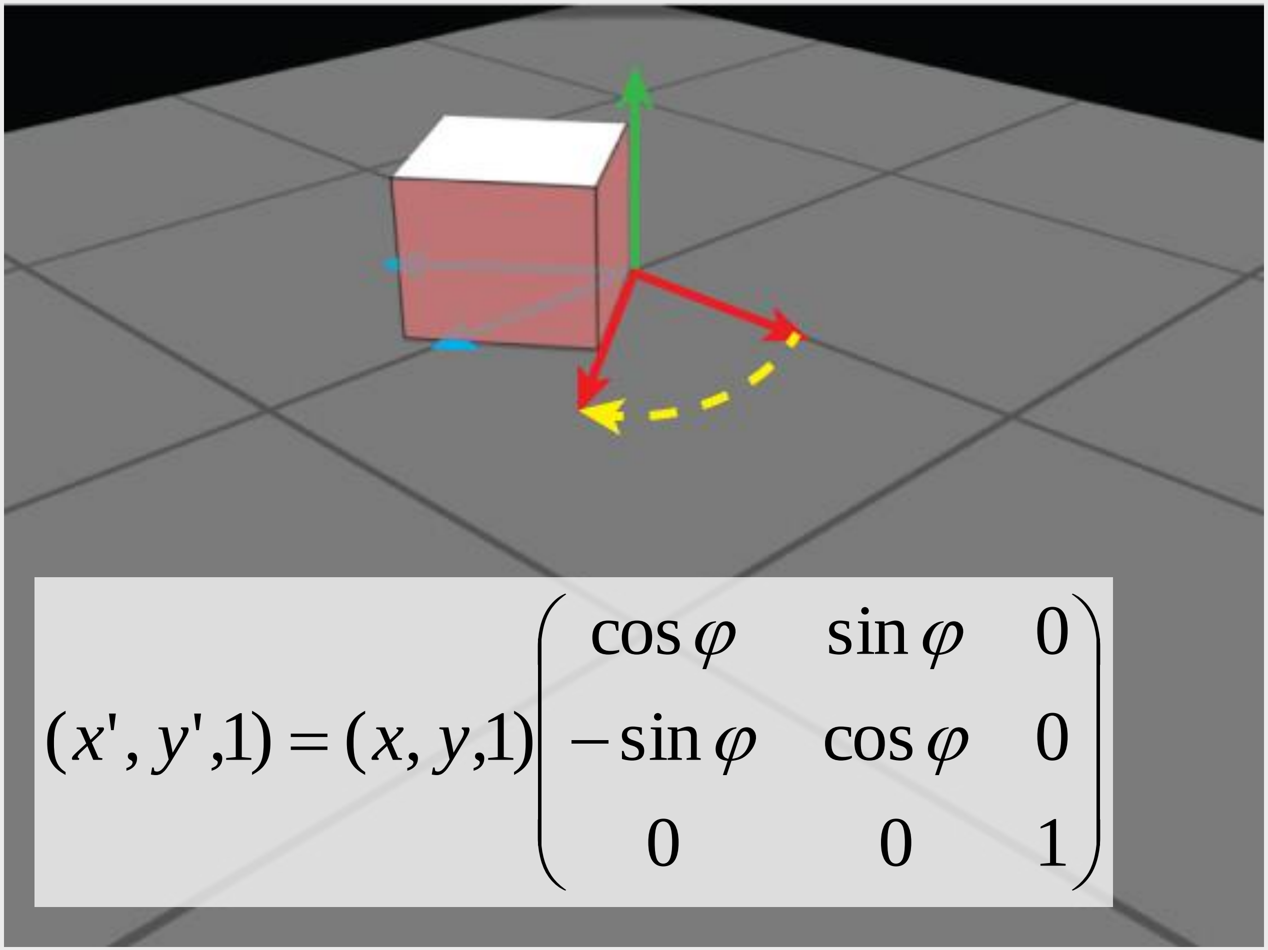
## TRANSLATION



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

# LOCAL $\longrightarrow$ GLOBAL COORDINATES

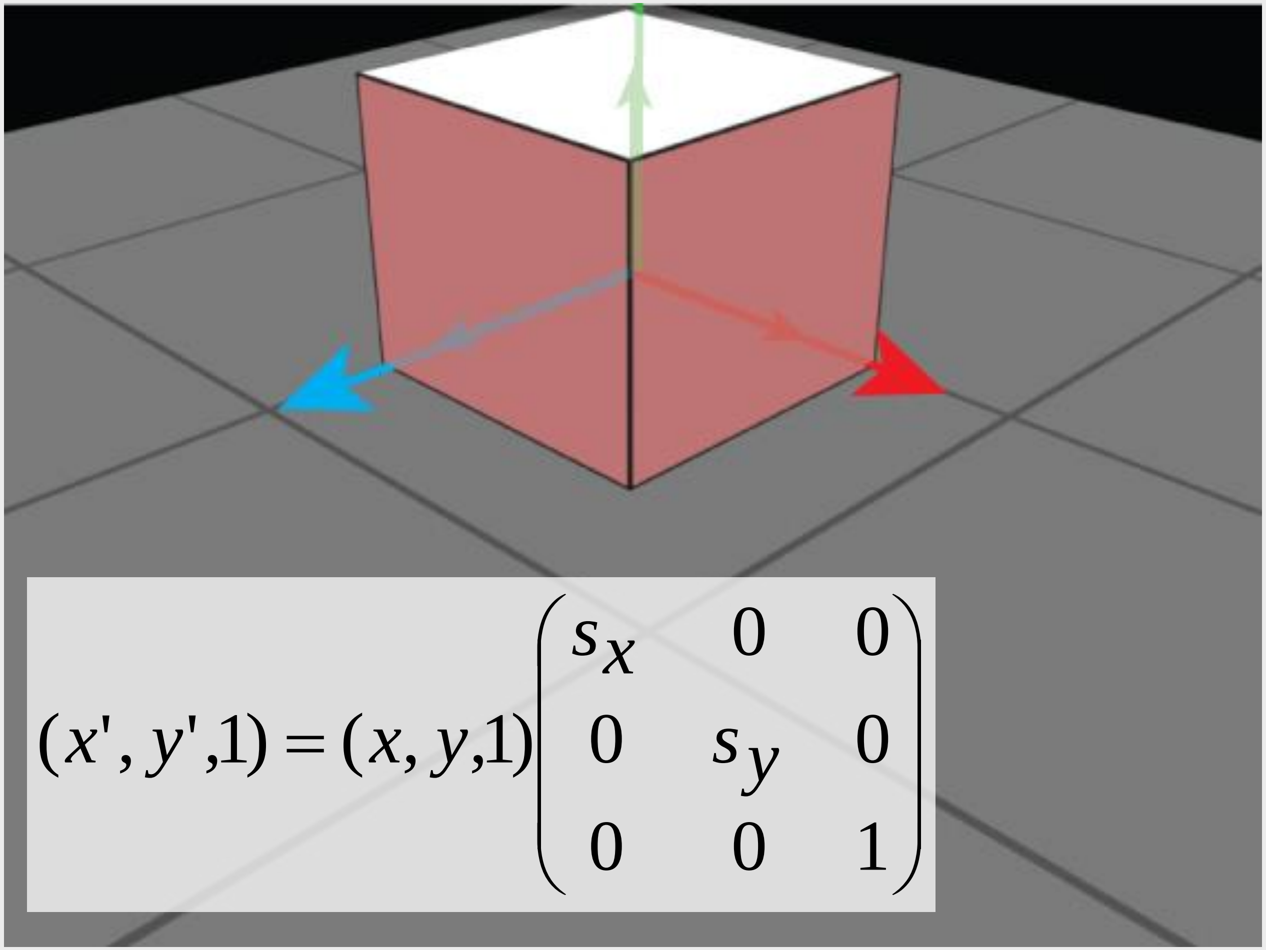
## ROTATION



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

# LOCAL ⟶ GLOBAL COORDINATES

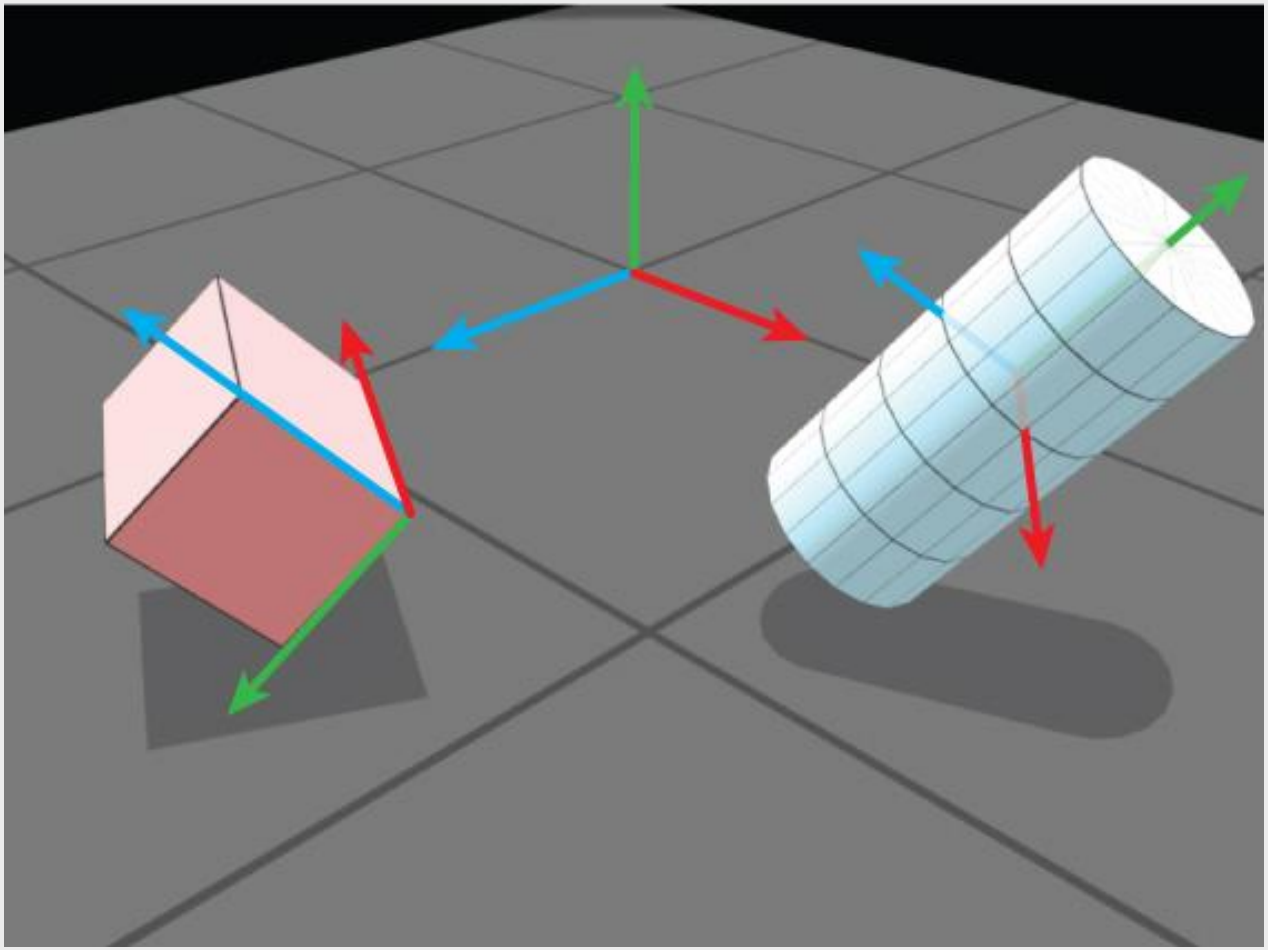
## SCALING



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

# LOCAL → GLOBAL COORDINATES

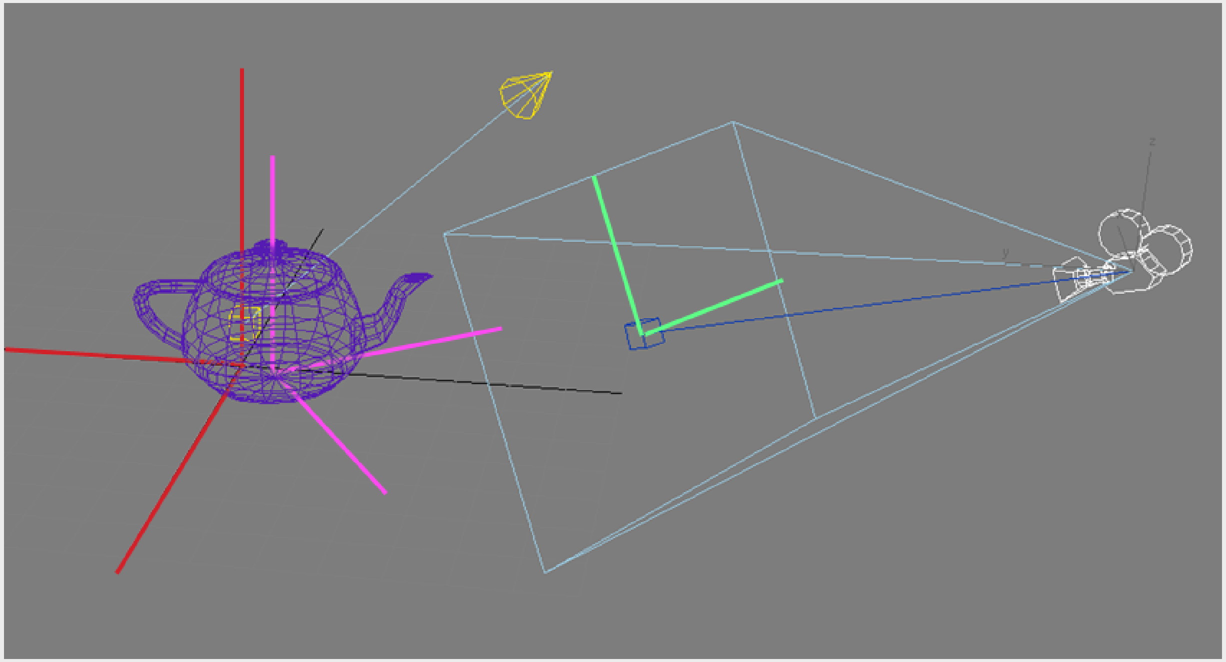
## ALL TRANSFORMATIONS COMBINED



# TRANSFORMATIONS

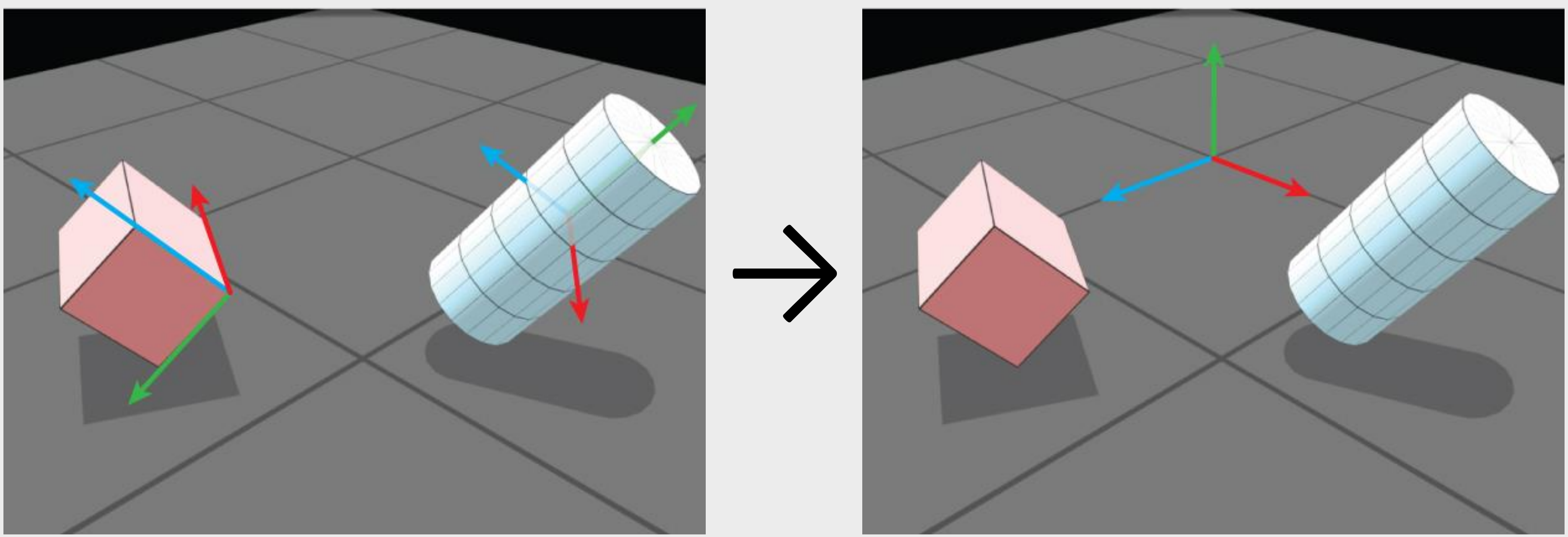
## TRANSFORMATION FROM ONE COORDINATE SYSTEM TO ANOTHER ONE IS A COMPOSITION OF PARTIAL TRANSFORMATIONS:

Translation
Rotation
Scaling



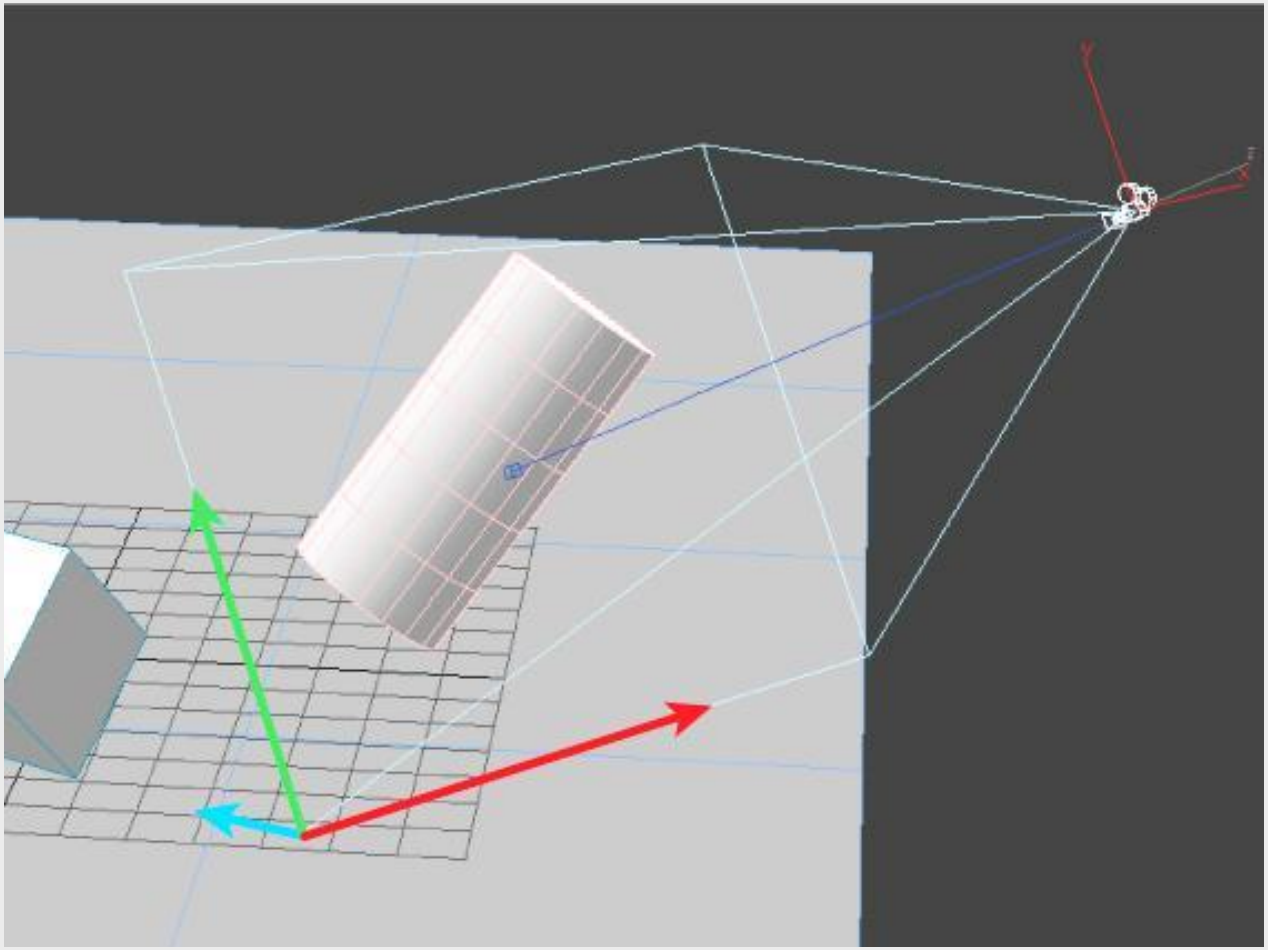
# MODEL TRANSFORMATION

TRANSFORMATION LOCAL → GLOBAL
COMBINATION OF ROTATE, TRANSLATE, SCALE
MATRIX MULTIPLICATION



# CAMERA COORDINATES

## XY OF SCREEN + Z AS DIRECTION OF VIEW



# ALL TRANSFORMATIONS

## MODEL TRANSFORMATION

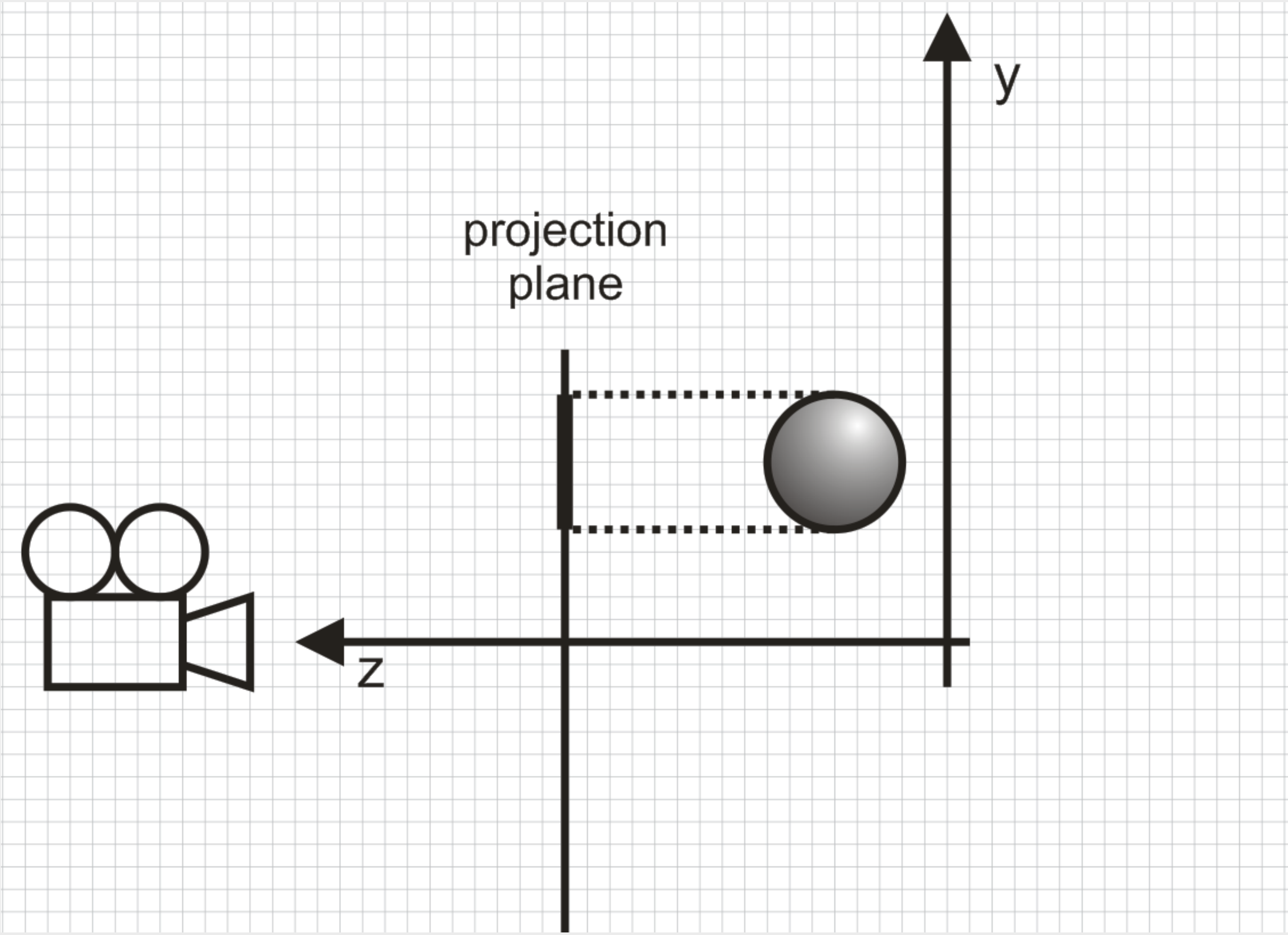
Unify coordinates by transforming local
to global coordinates

## VIEW TRANSFORMATION

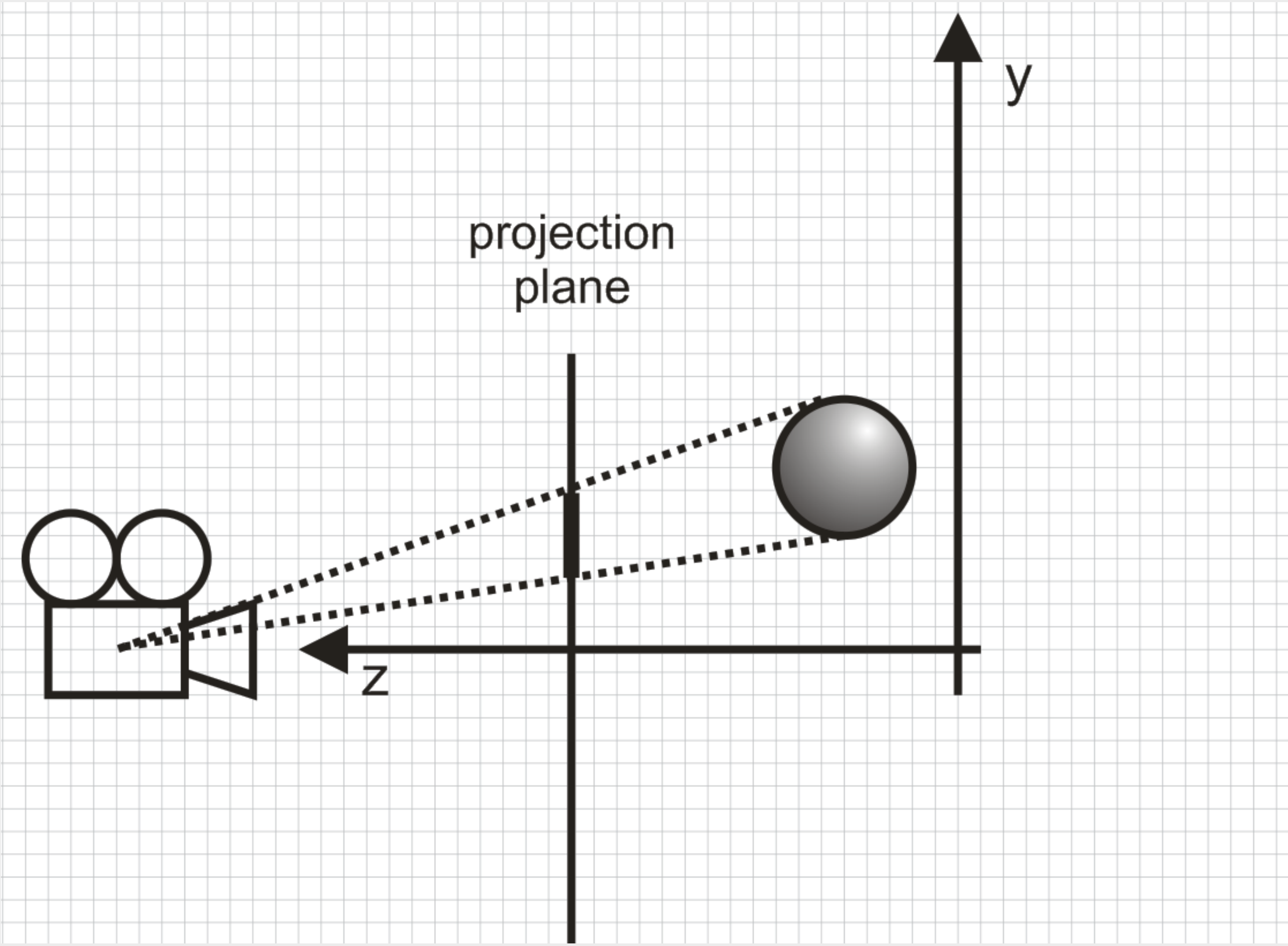
Transform global coordinates so that they are
aligned with camera coordinates
To make projection computable

# PROJECTION

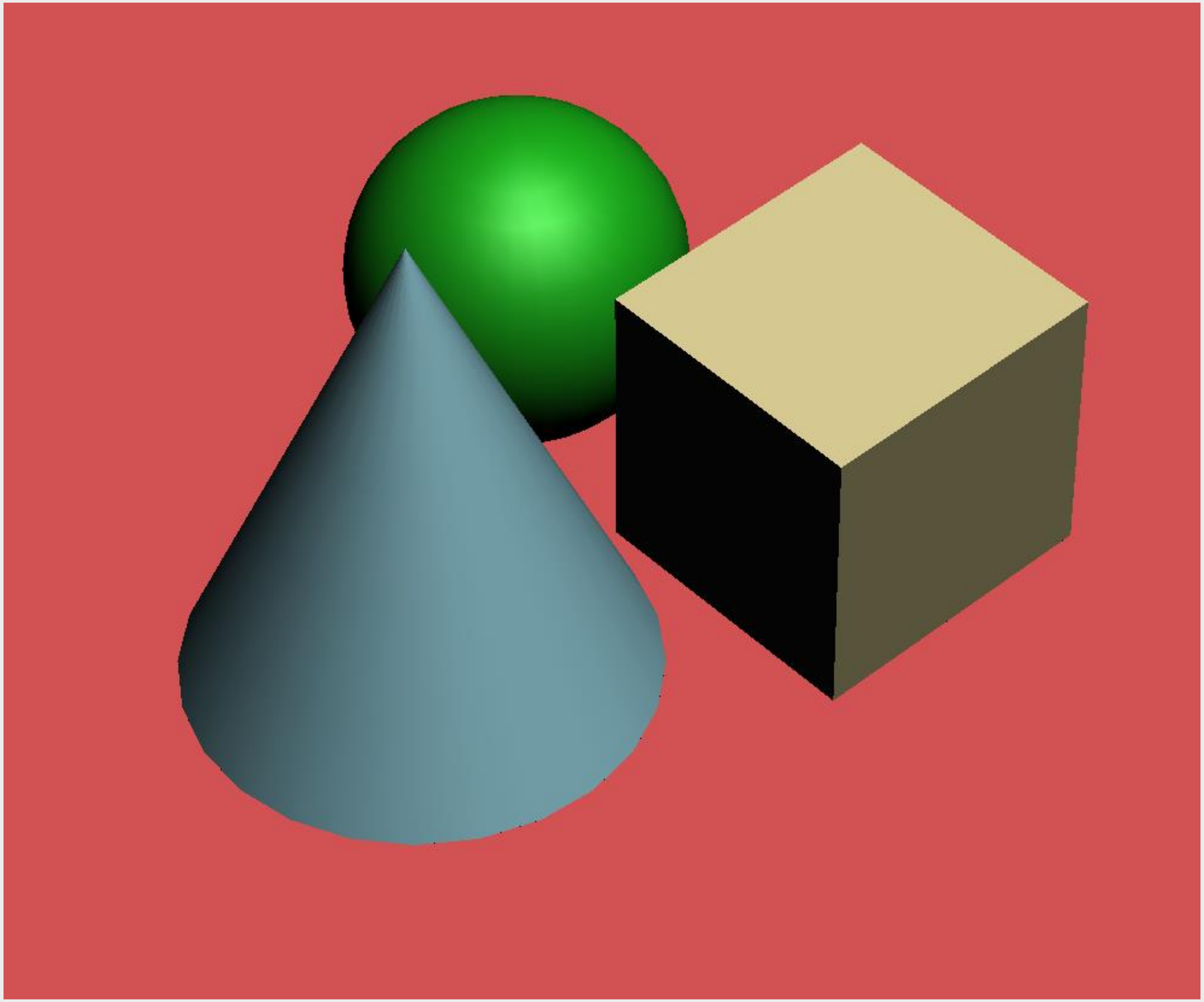
# ORTHOGONAL PROJECTION



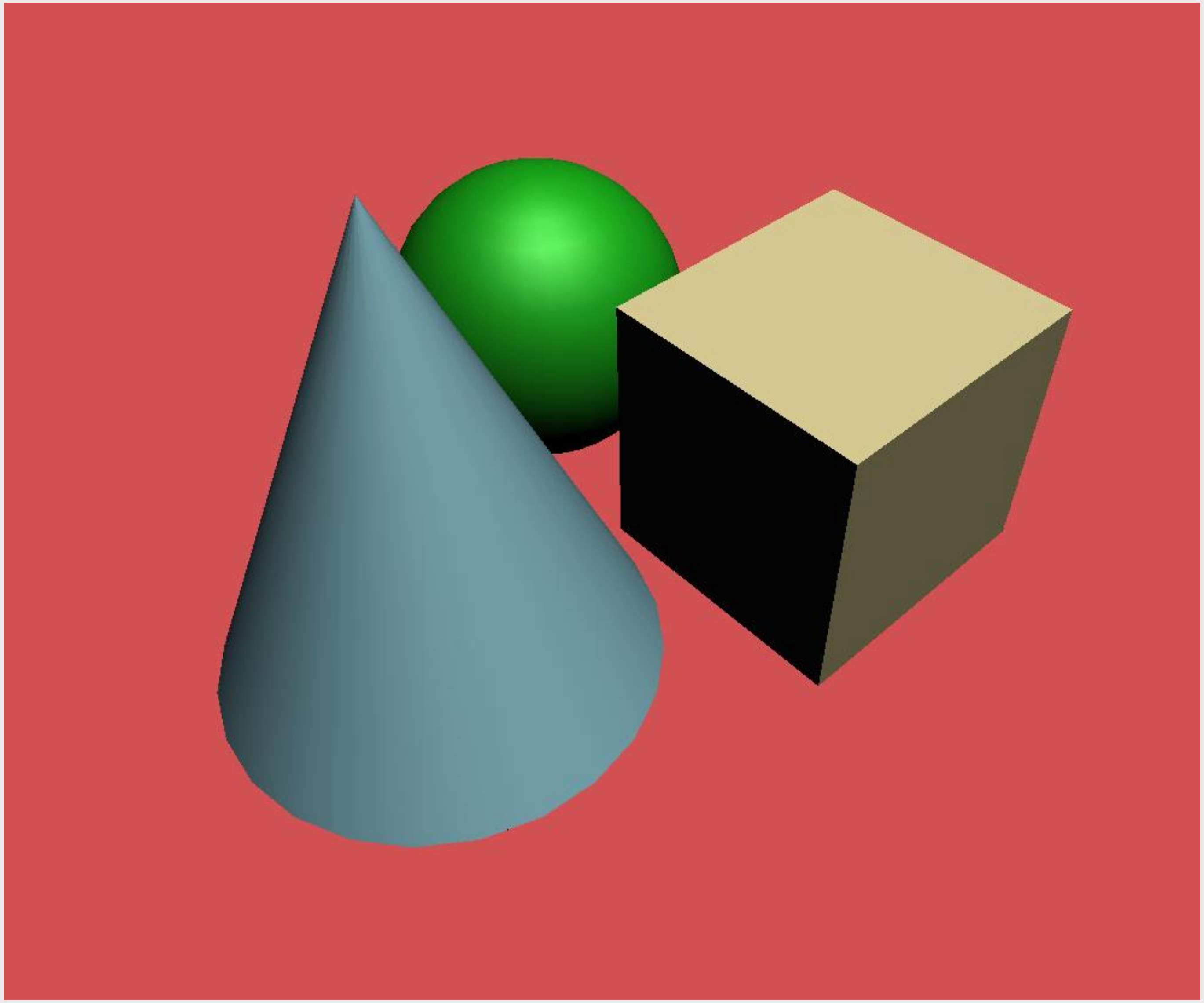
# PERSPECTIVE PROJECTION



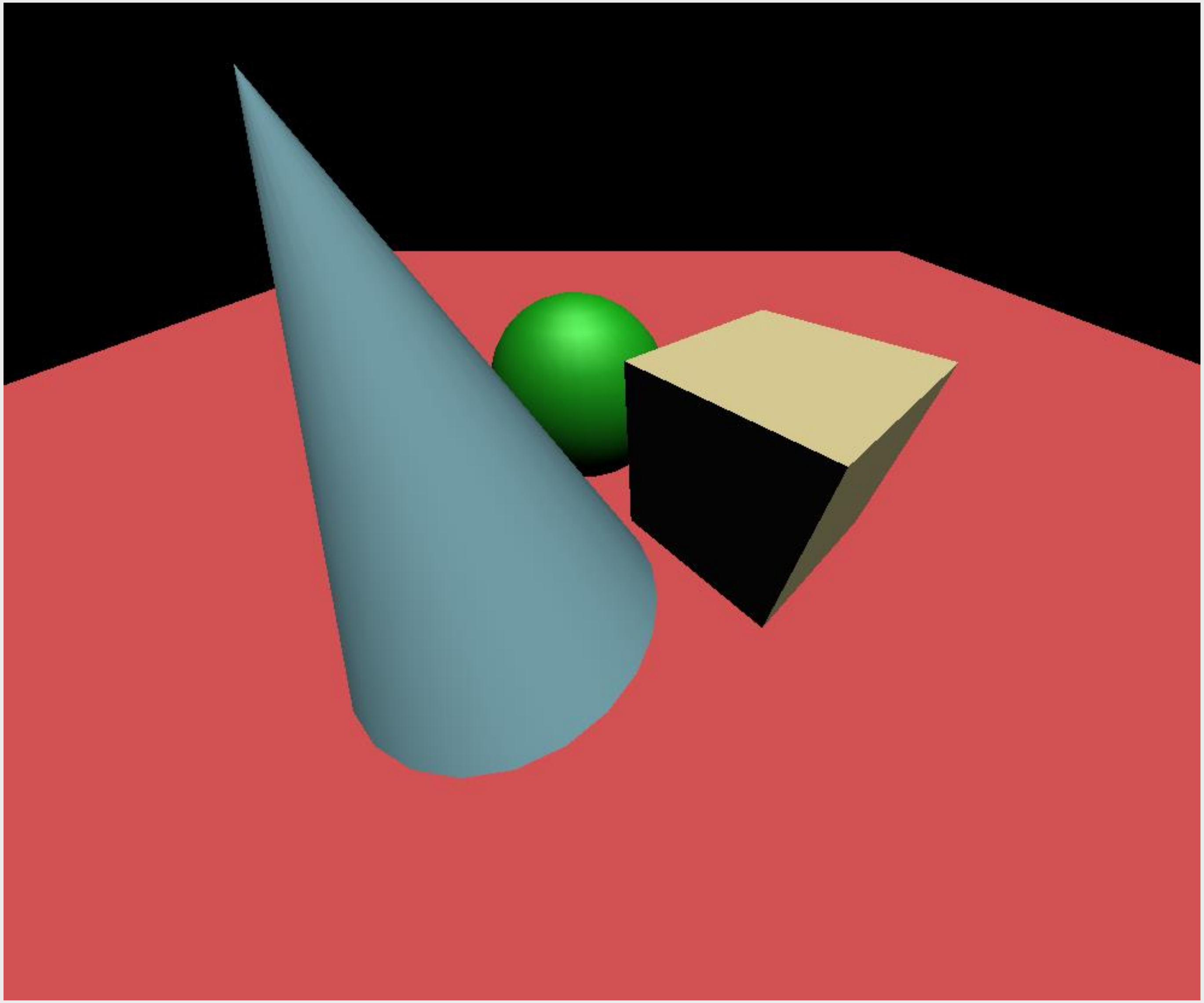
# PROJECTIONS - PARALLEL



# PROJECTIONS - PERSPECTIVE



# PROJECTIONS – DISTORTED PERS.



# ORTHOGONAL PROJECTION

## ONLY 2 DIMENSIONS ARE VISIBLE

# PERSPECTIVE

## 3 DIMENSIONS, NEAR OBJECTS ARE LARGER

# ISOMETRIC

## 3 DIMENSIONS, ALL OBJECTS SAME SIZE

# VIEWPORT TRANSFORMATION

## GLOBAL COORDINATES

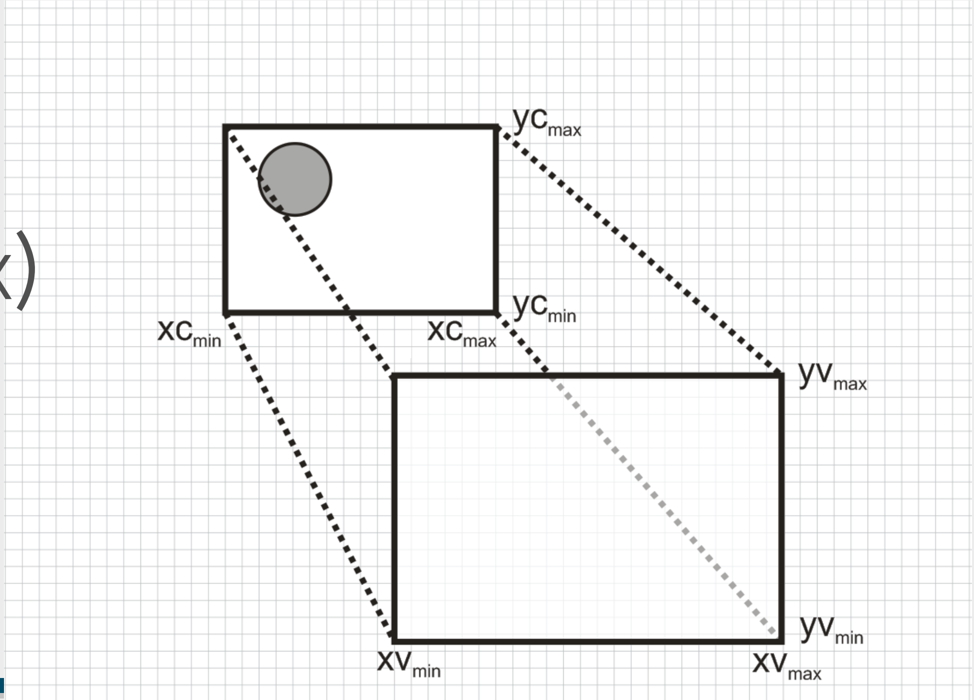
e.g. (-50..50 cm, -50..50 cm, -50..50 cm)

## CAMERA COORDINATES

e.g. (-1..1, -1..1, -1..1)

## VIEWPORT (WINDOW)

e.g. (0..1200 px, 0..800 px)



# VIEWPORT TRANSFORMATION

## AND AGAIN, THERE'S A MATRIX FOR THAT!

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

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

# WELCOME TO THE MATRIX!

1. LOCAL → GLOBAL COORDINATES

   translate, rotate, scale, translate

2. GLOBAL → CAMERA

   translate, rotate, rotate

3. PROJECTION

   e.g. perspective or orthogonal

4. CAMERA → VIEWPORT

   translate, scale, translate

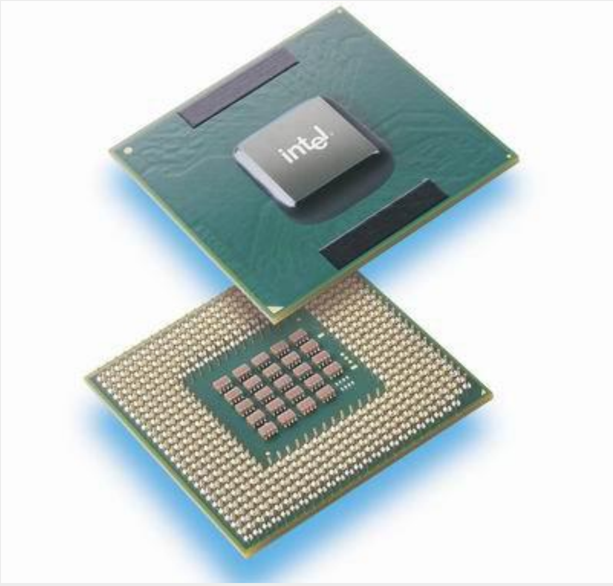
COMBINE TRANSFORMATIONS

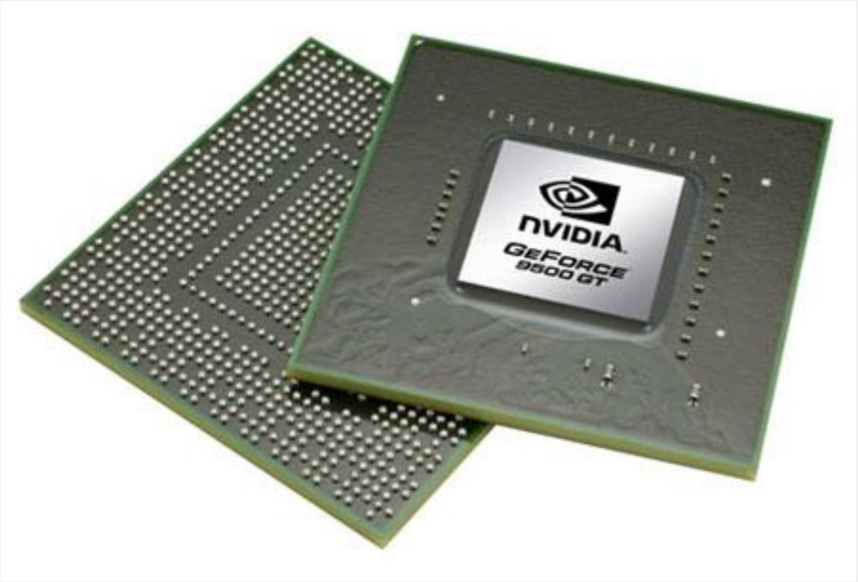
= MULTIPLY MATRICES

# WHO DOES ALL THE WORK?

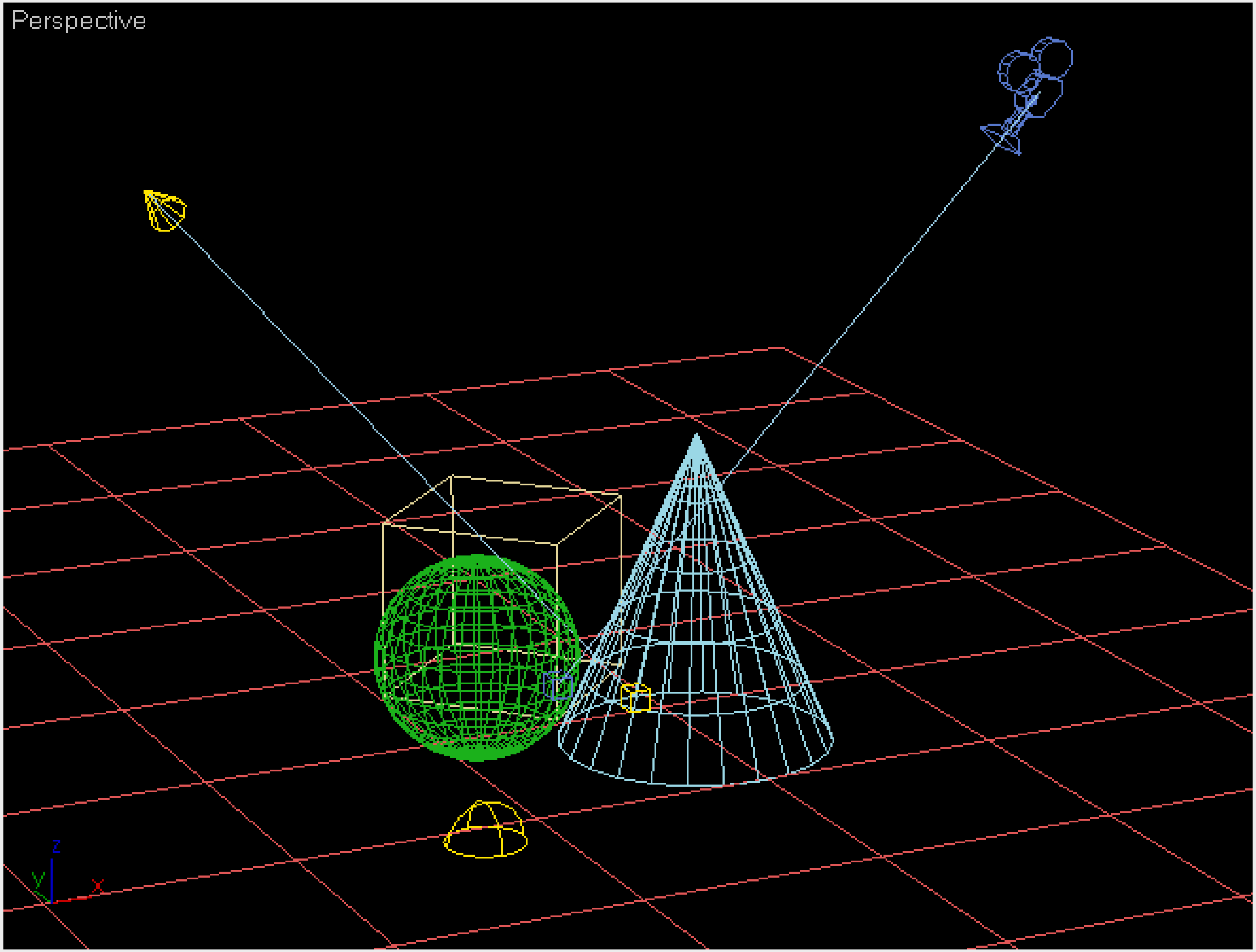
MATRIX OPERATIONS SLOW ON CPU,

GPU BUILT TO ACCELERATE MATRIX CALCULATIONS

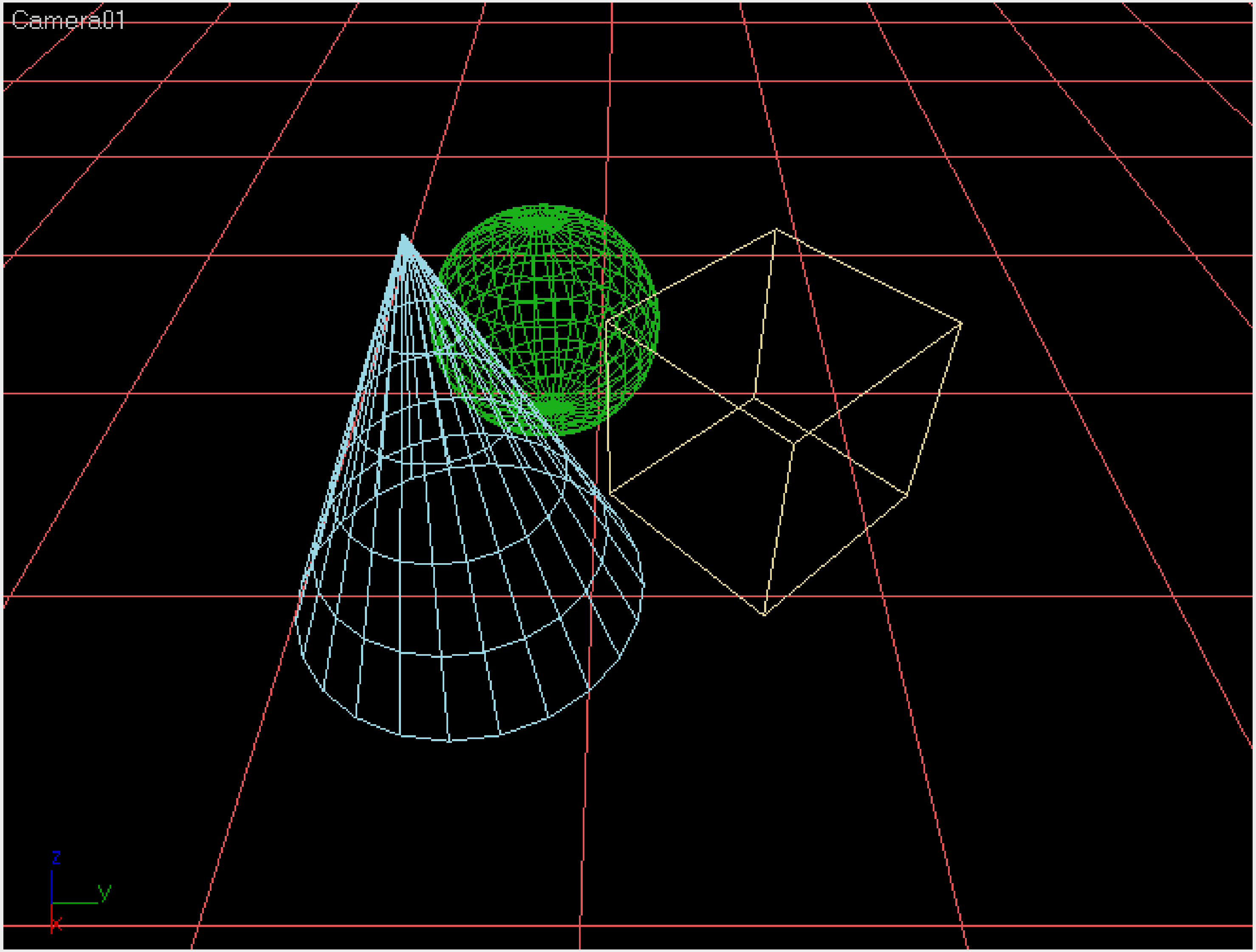


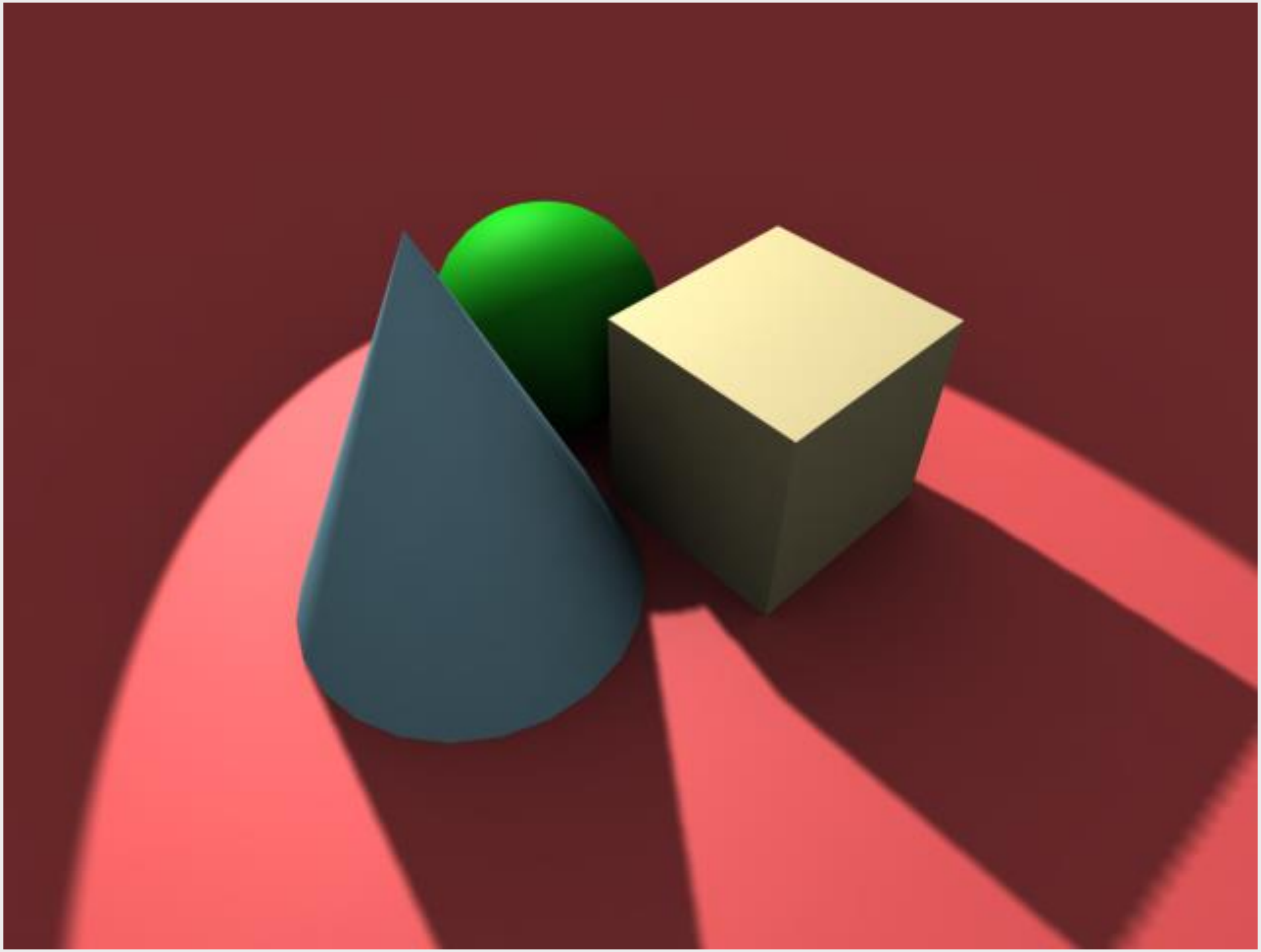
# EXAMPLE – WORLD



# EXAMPLE – FROM CAMERA



# EXAMPLE – OUTPUT



# RENDERING

MODELS → PIXELS

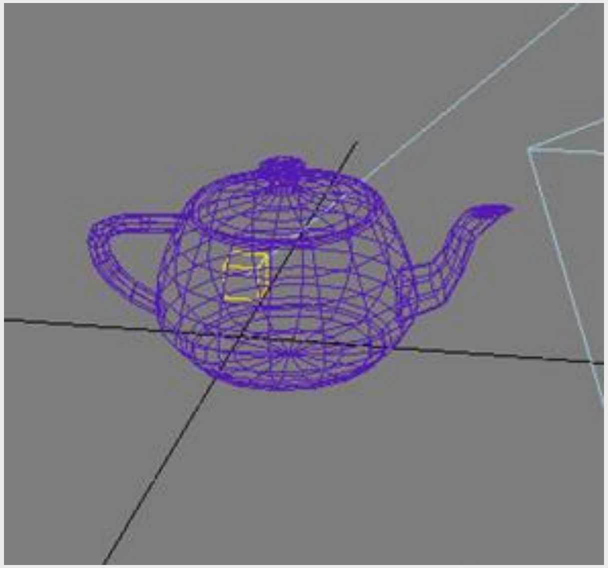
MODEL TRANSFORMATION

VIEWPORT TRANSFORMATION

CLIPPING, CULLING

RASTERIZATION

TEXTURING & LIGHTING



WHAT THE EYE
DOESN'T SEE,
THE HEART
DOESN'T GRIEVE
OVER

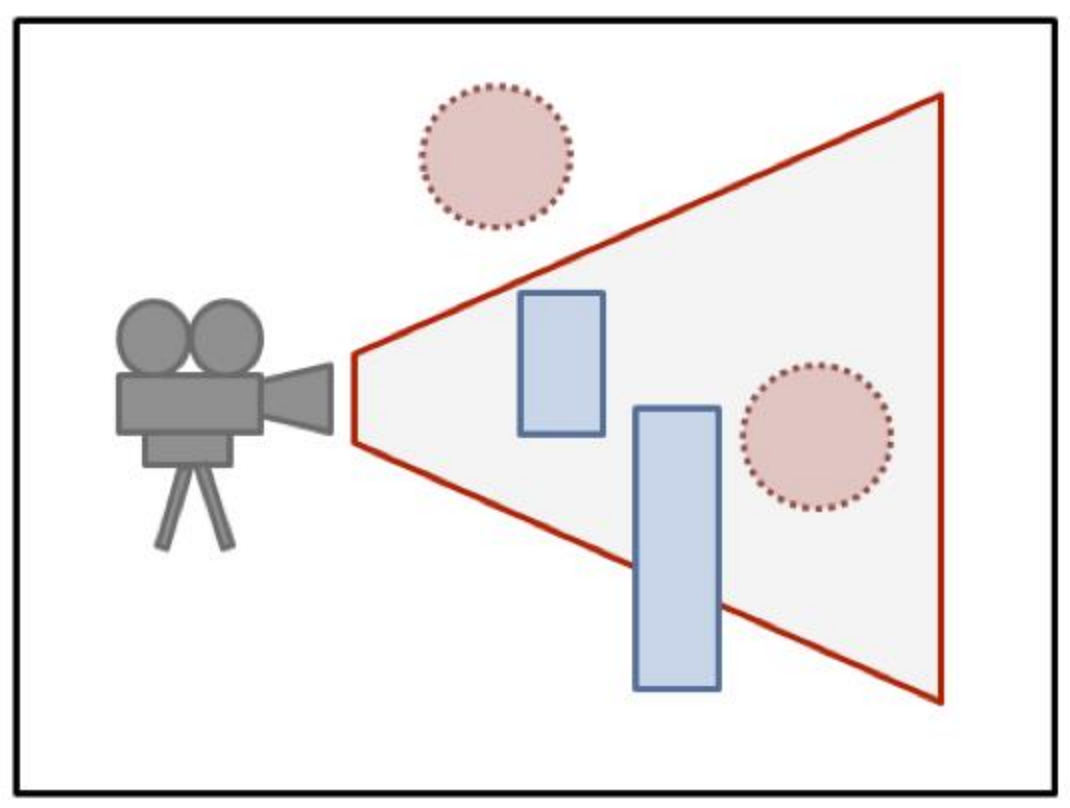
# CLIPPING AND CULLING

# GENERAL PROBLEM:

WHICH OBJECTS / OBJECT PARTS ARE VISIBLE?

OBJECTS OUTSIDE THE VIEW CAN BE IGNORED

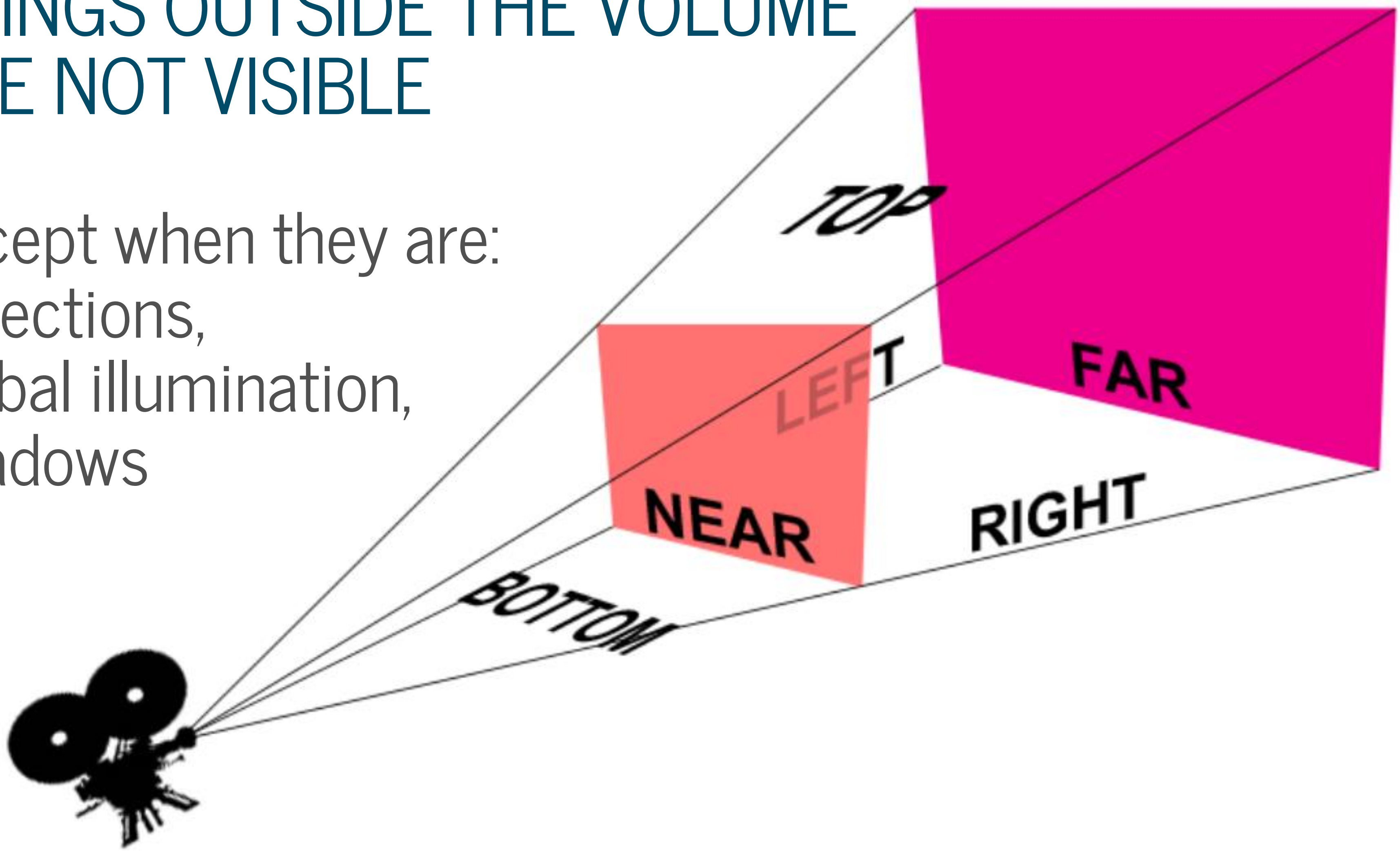
SPEEDS UP THE RENDERING



# VISIBLE VOLUME

## THINGS OUTSIDE THE VOLUME ARE NOT VISIBLE

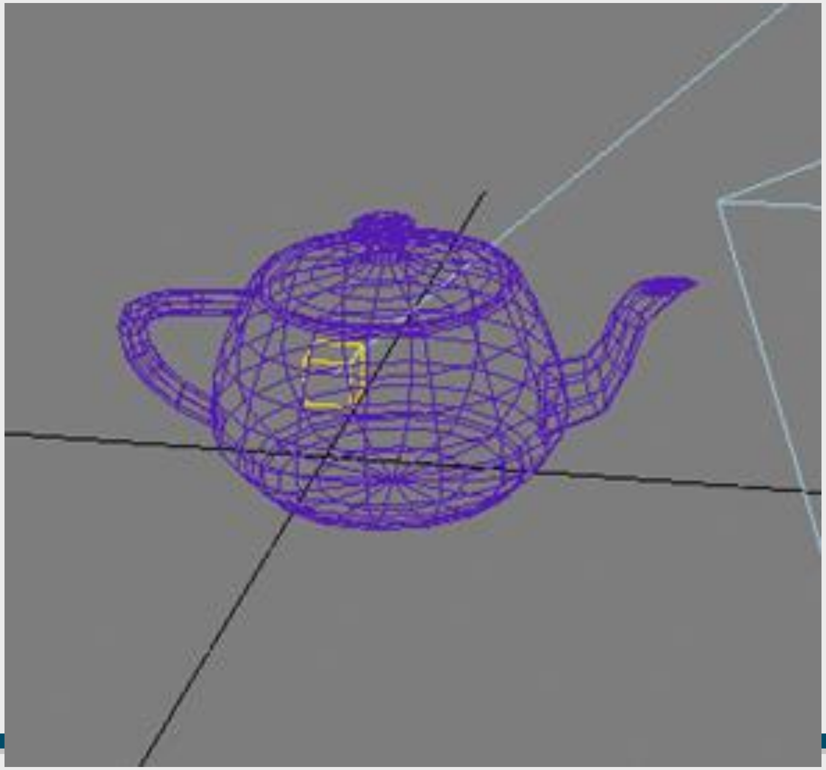
Except when they are:
reflections,
global illumination,
shadows



# BACK-FACE CULLING

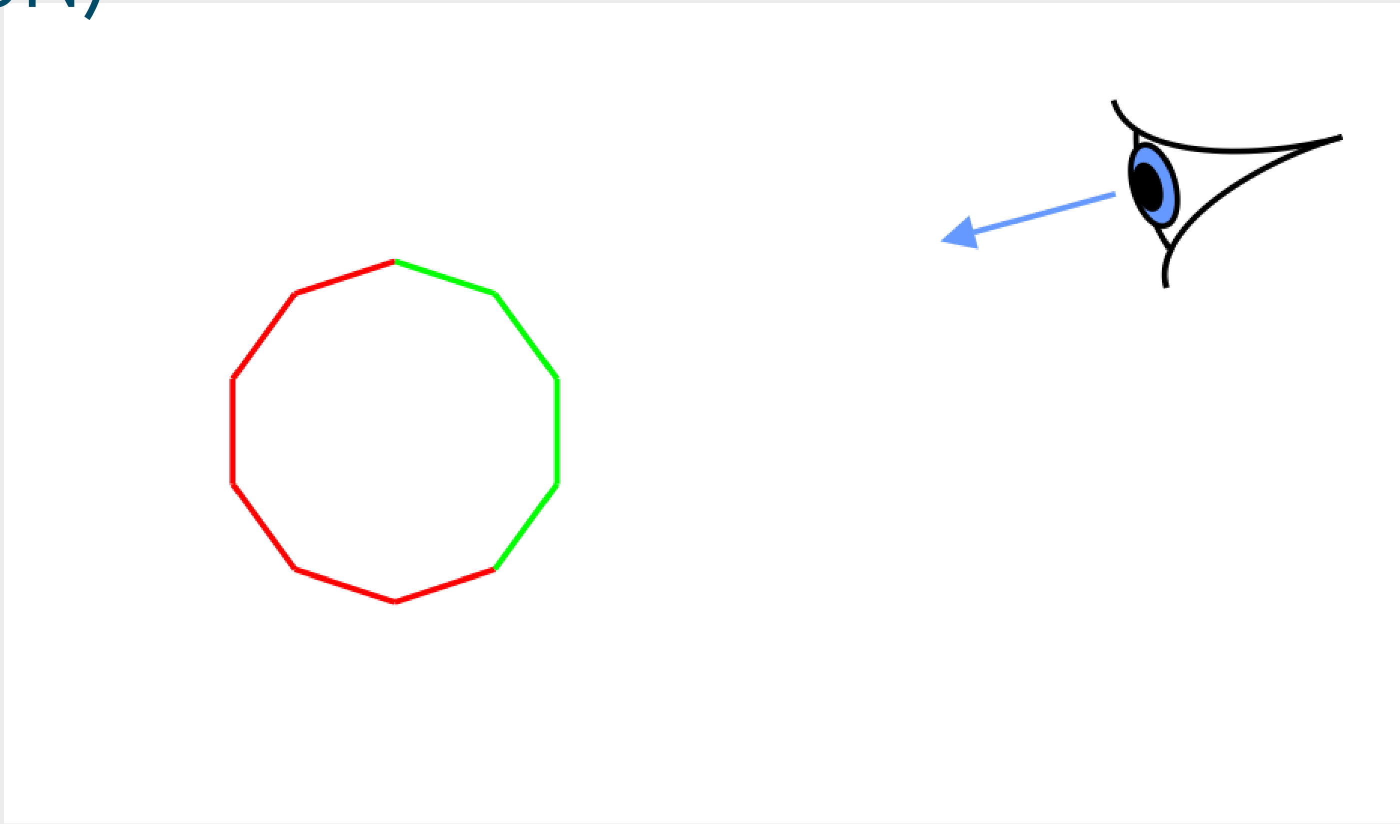
## PARTS OF OBJECT NOT FACING THE CAMERA ARE
## ALSO INVISIBLE

Except for semi-transparency, mirrors etc.



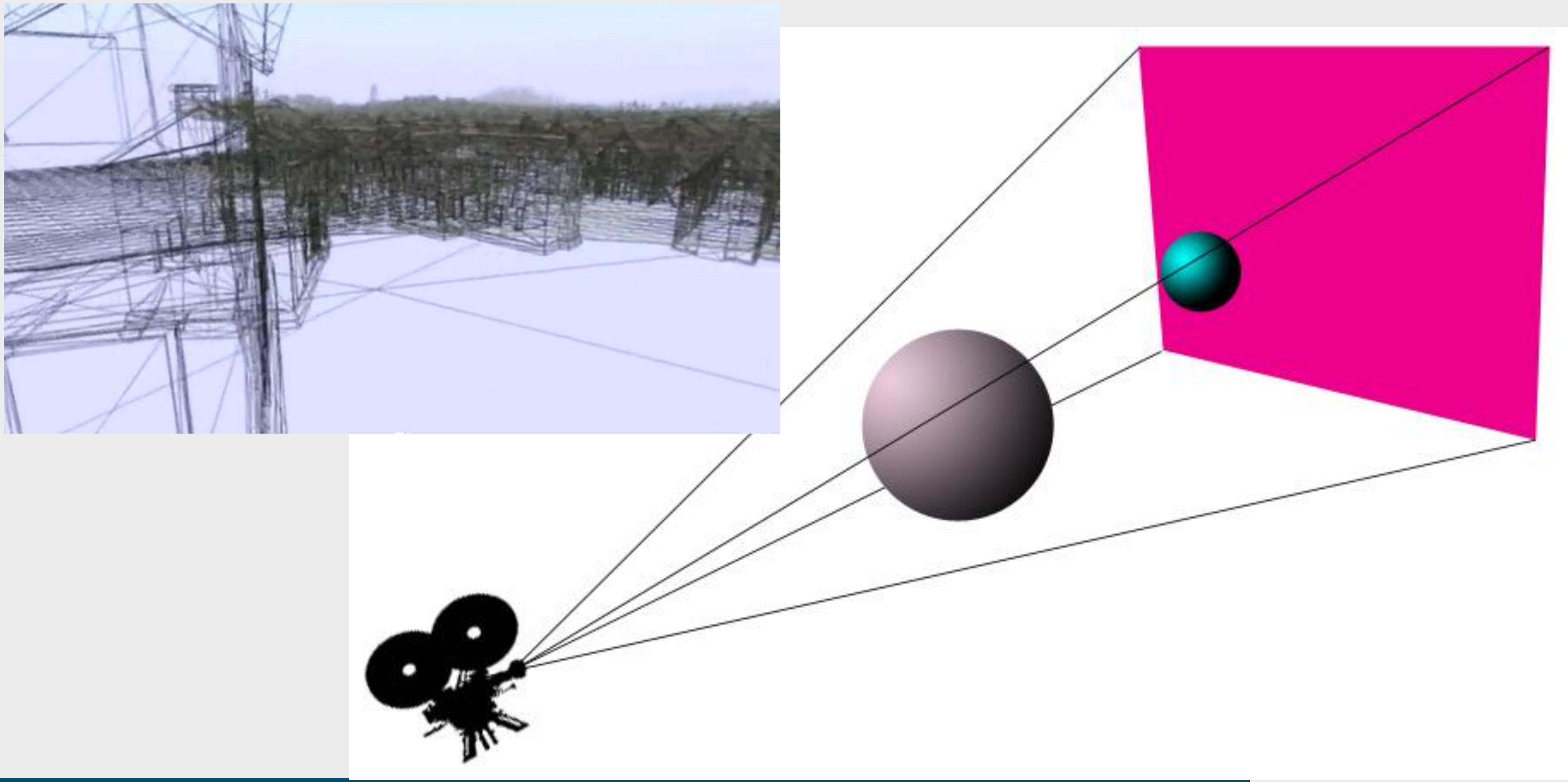
# BACKFACE CULLING

WHICH OBJECT FACES ARE VISIBLE?
REMEMBER NORMAL VECTOR (FACE ORIENTATION)



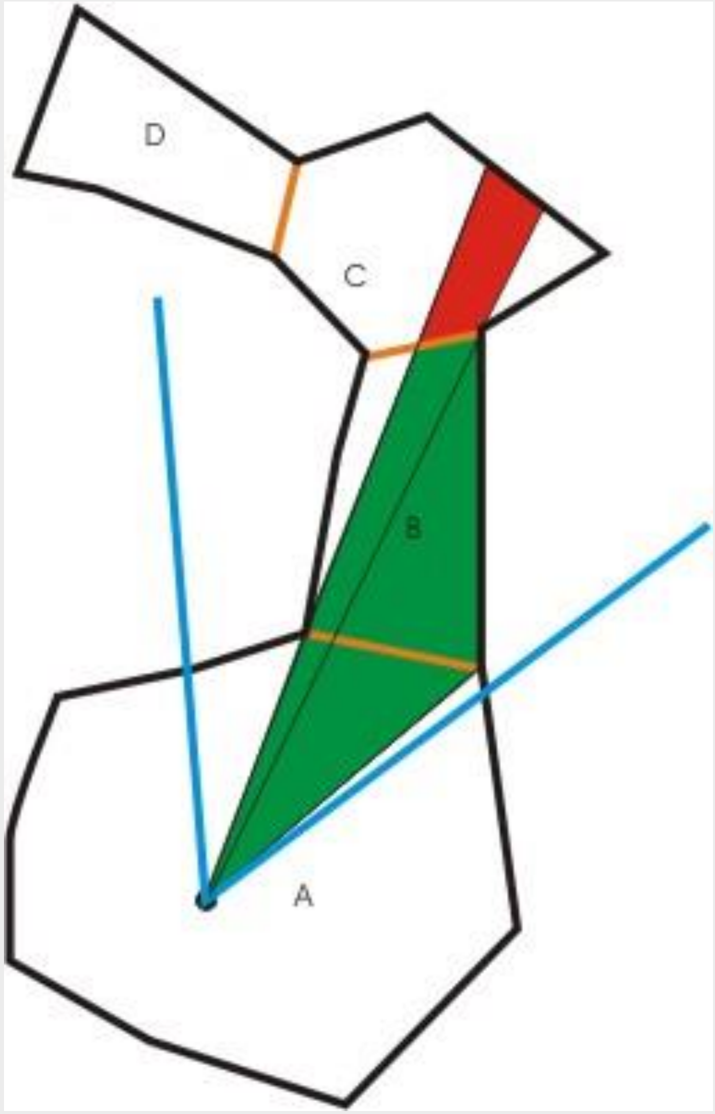
# OCCLUSION CULLING

## SOME OBJECTS ARE FULLY OCCLUDED BY OTHERS



# PORTAL CULLING

## SOME PARTS OF THE SCENE ARE NOT VISIBLE FROM SOME OTHER PARTS OF THE SCENE

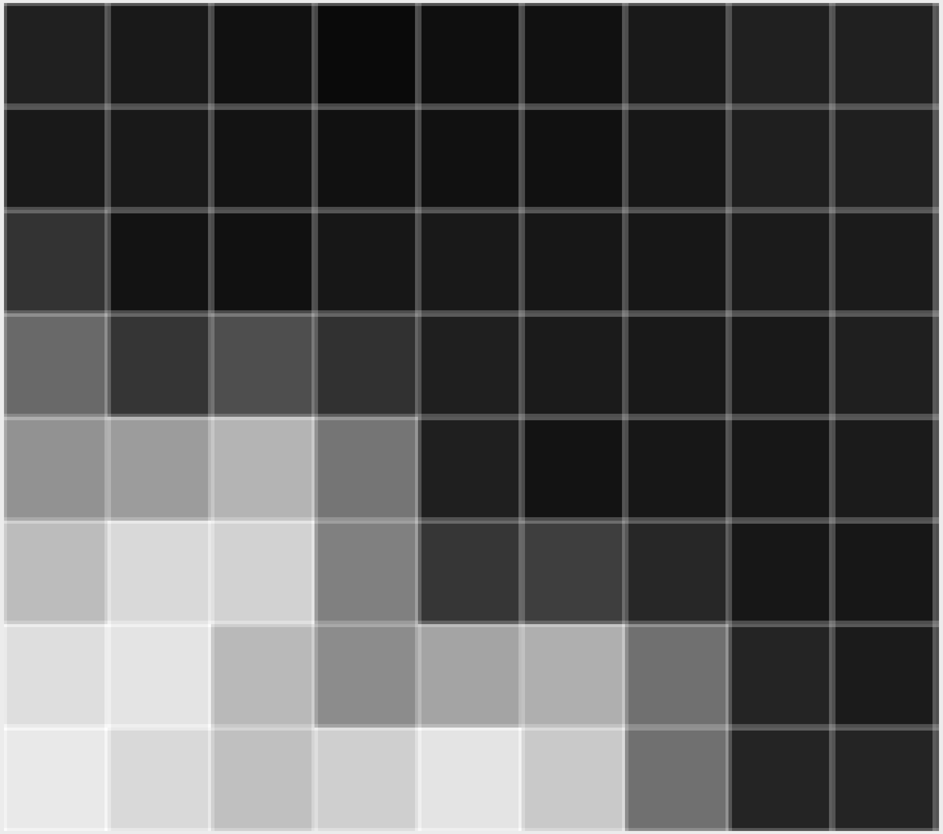



# RASTERIZATION

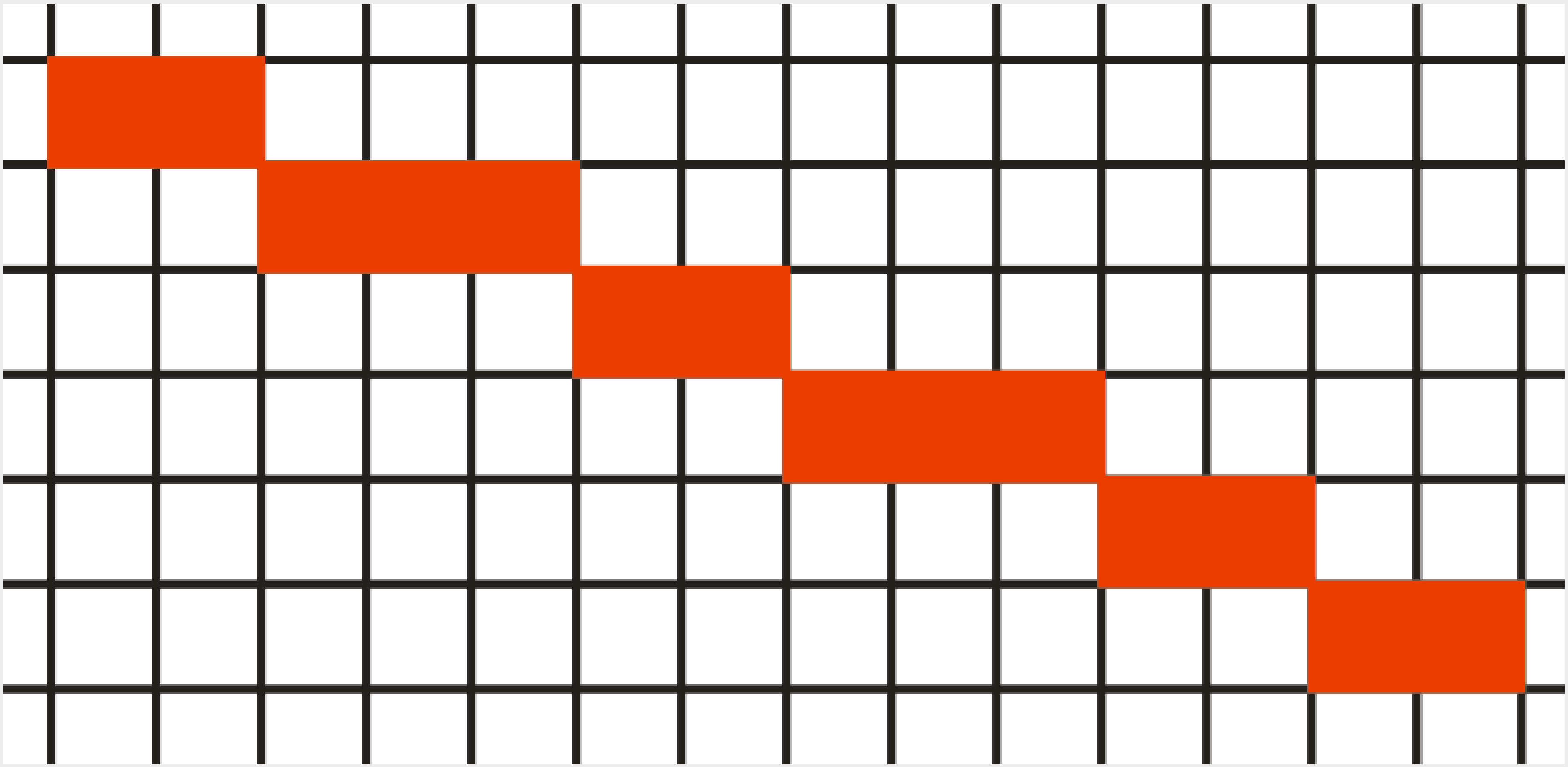
# GENERAL PROBLEM

GIVEN A CONTINUOUS GEOMETRIC REPRESENTATION OF AN OBJECT

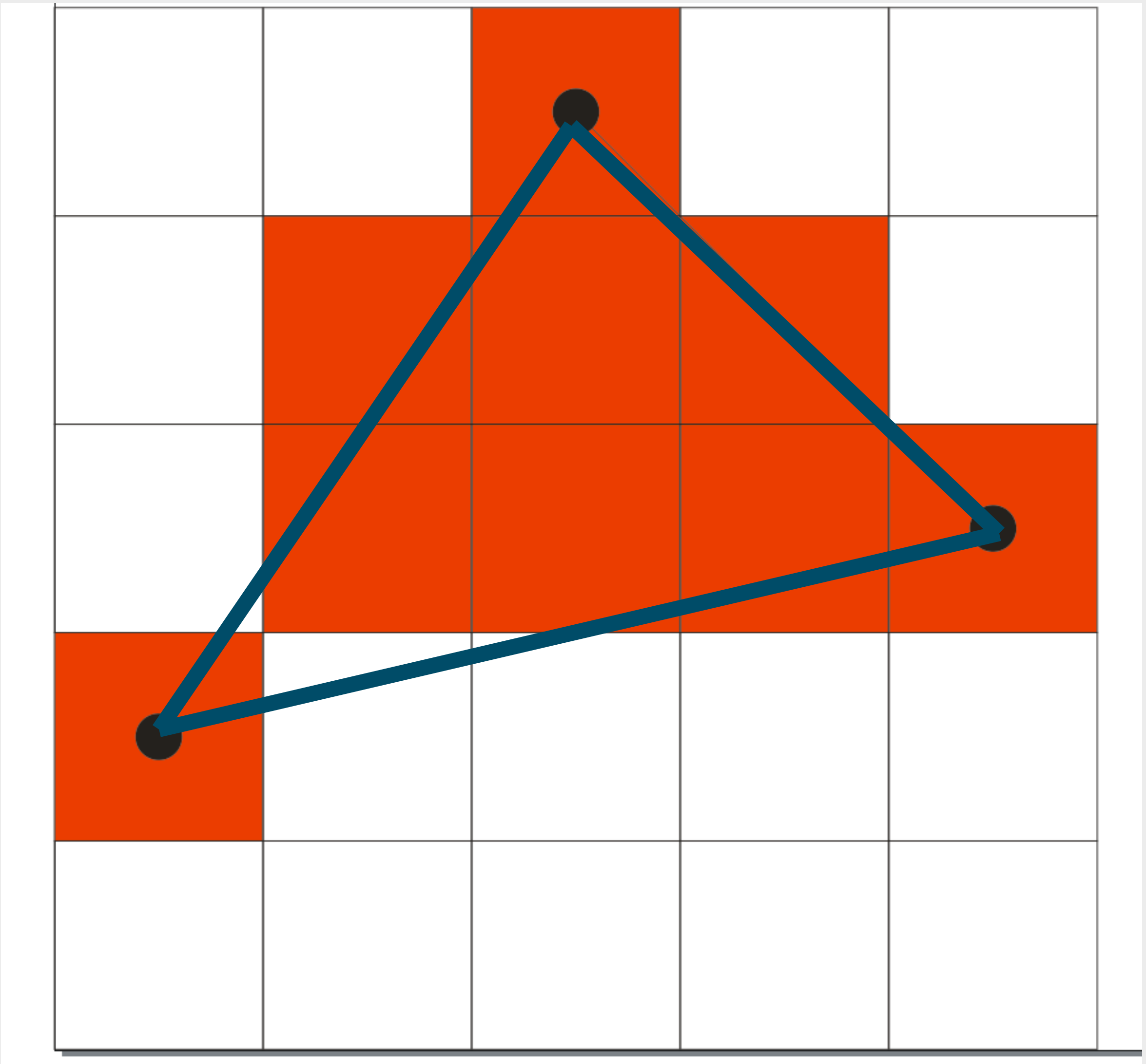
DECIDE WHICH PIXELS ARE OCCUPIED BY THE OBJECT



# LINE RASTERIZATION



# SCAN-LINE ALGORITHM

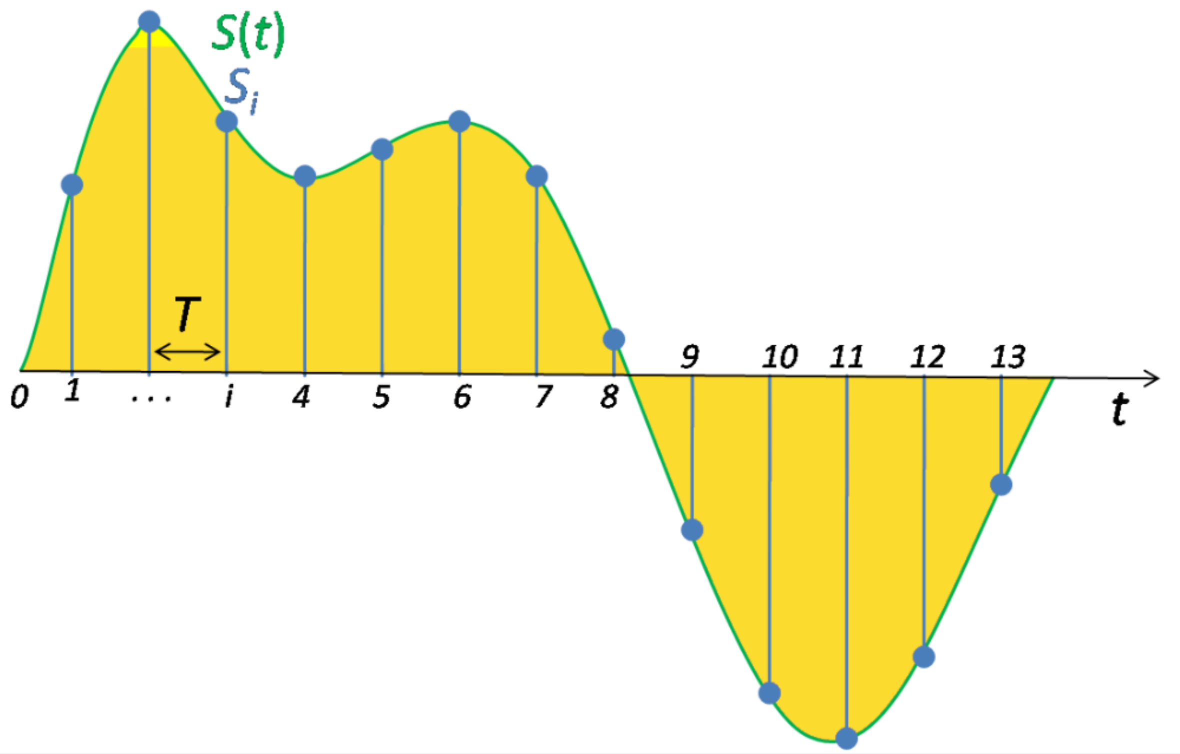


# ALIAS

## CONTINUOUS → DISCRETE
## ARTIFACTS MIGHT APPEAR
## RASTERIZATION ALIAS - JAGGED EDGES

## SAMPLING

creating observation of a continuous phenomenon at discrete points



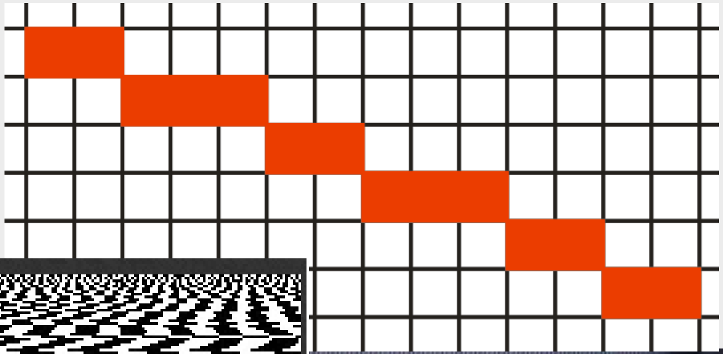
# FORMS OF ALIAS

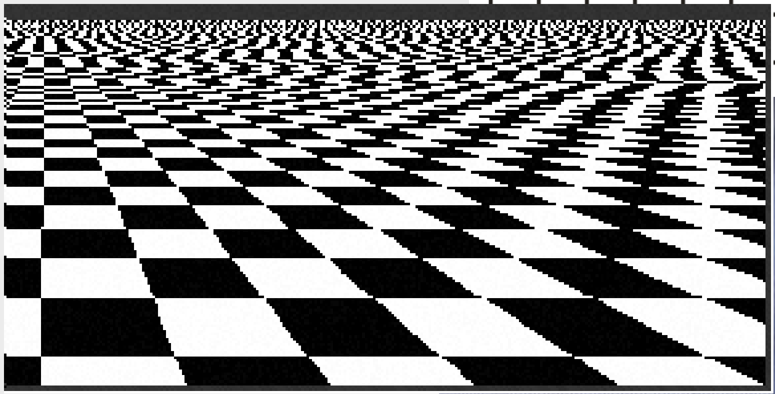
## SPATIAL ALIAS

jaggy edges
moiré
texture
distortion

## TEMPORAL

"wagon wheel"





# ANTI-ALIASING

## GENERAL (GLOBAL) ANTI-ALIASING - SUPERSAMPLING

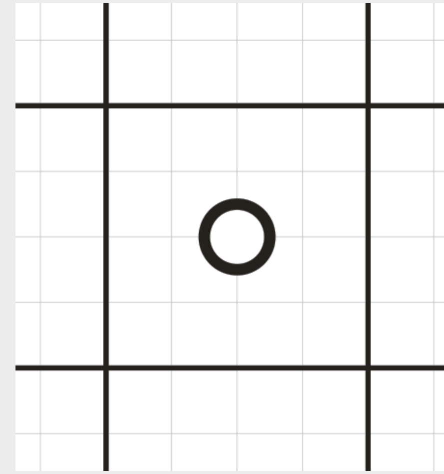
works on all objects - that's good
works on ALL objects - that's bad

## OBJECT (LOCAL) ANTI-ALIASING

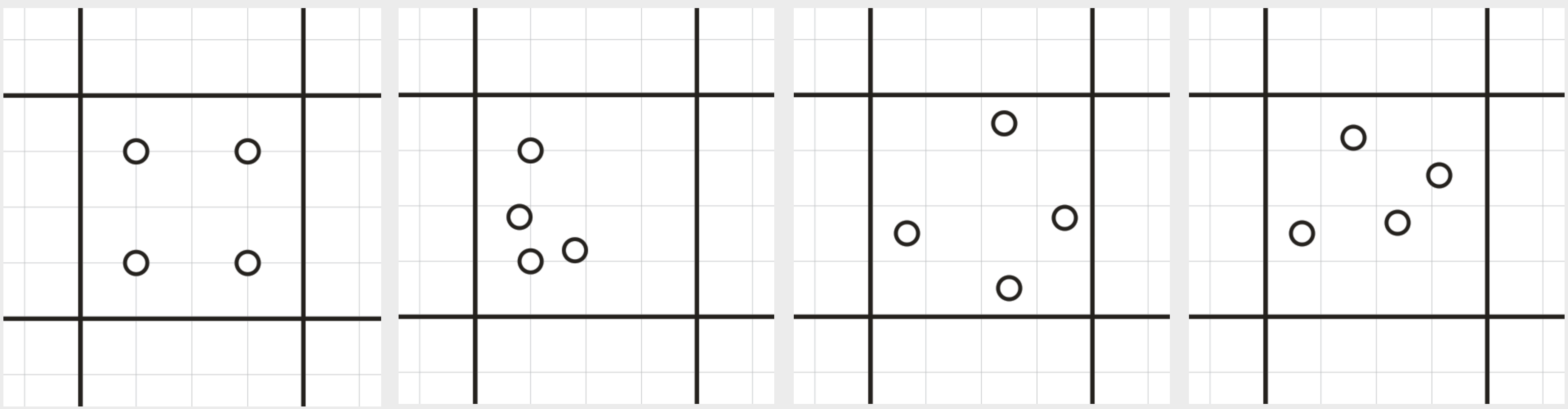
line anti-aliasing
silhouette anti-aliasing
texture anti-aliasing

# SUPER-SAMPLING

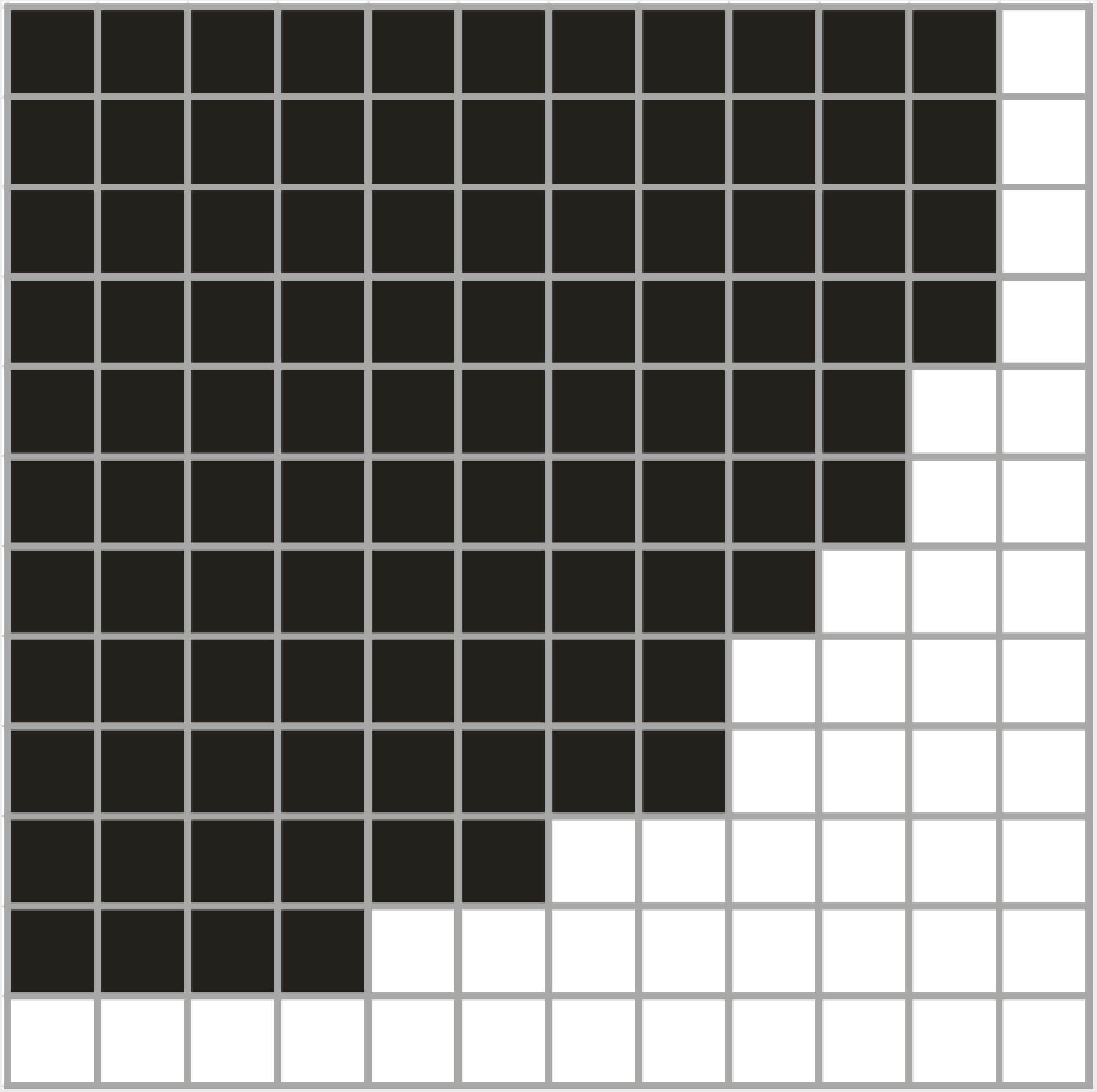
## FOR EACH PIXEL PERFORM MULTIPLE SUB-PIXEL SAMPLINGS AND COMBINE THE RESULTS



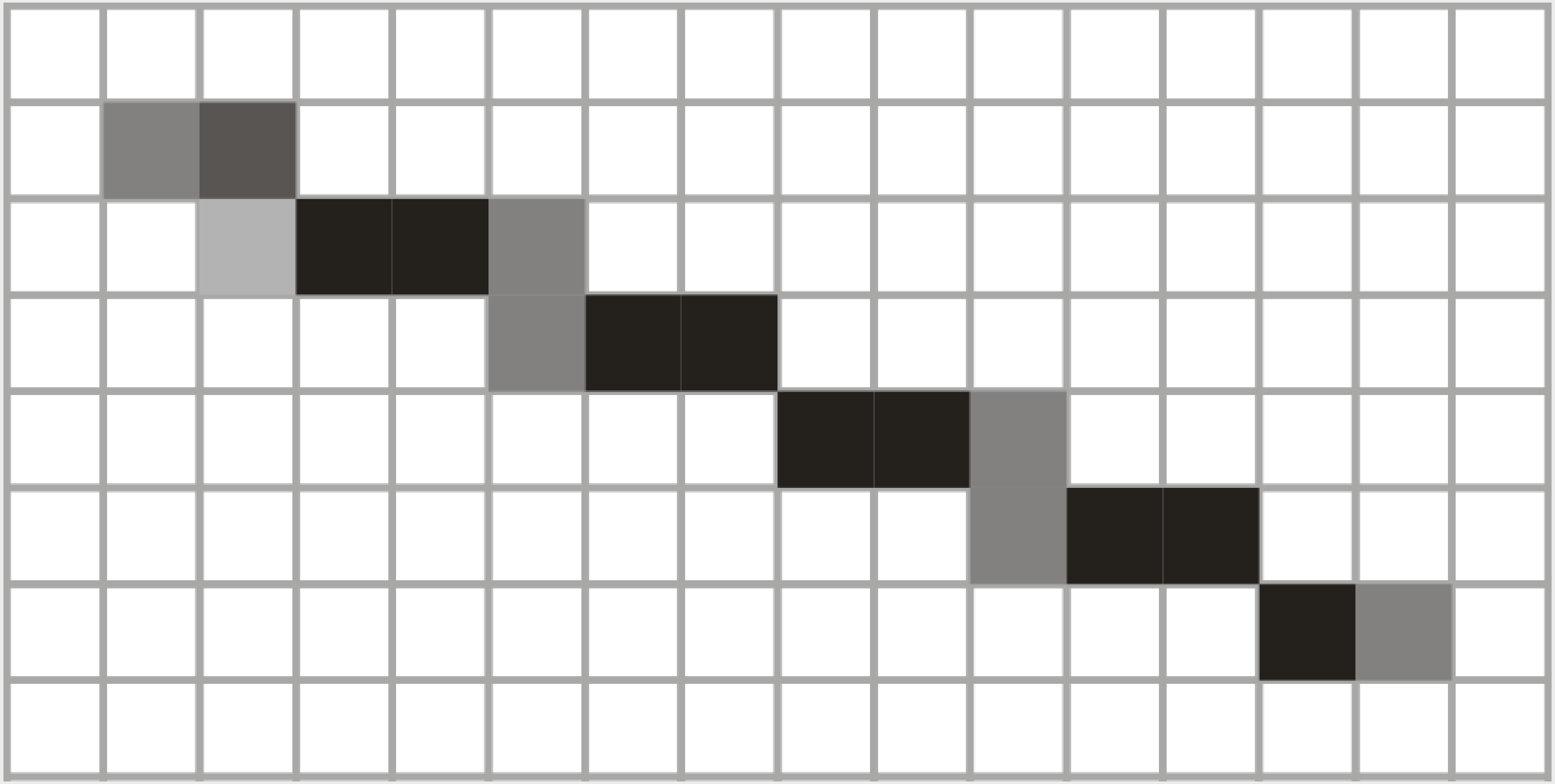
Various sub-pixel distributions:



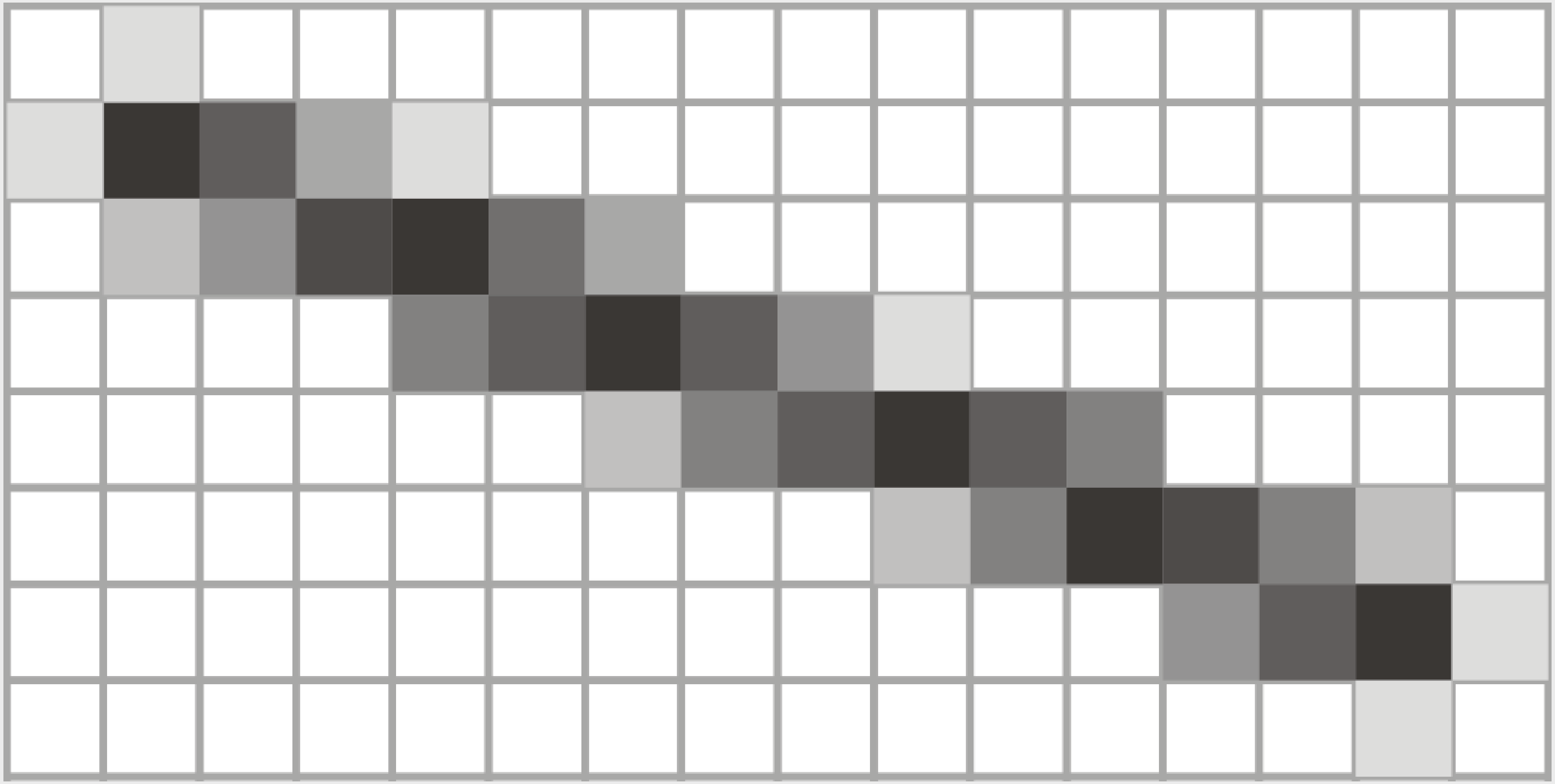
# SUPER-SAMPLING EXAMPLE



# LINE SUPER-SAMPLING



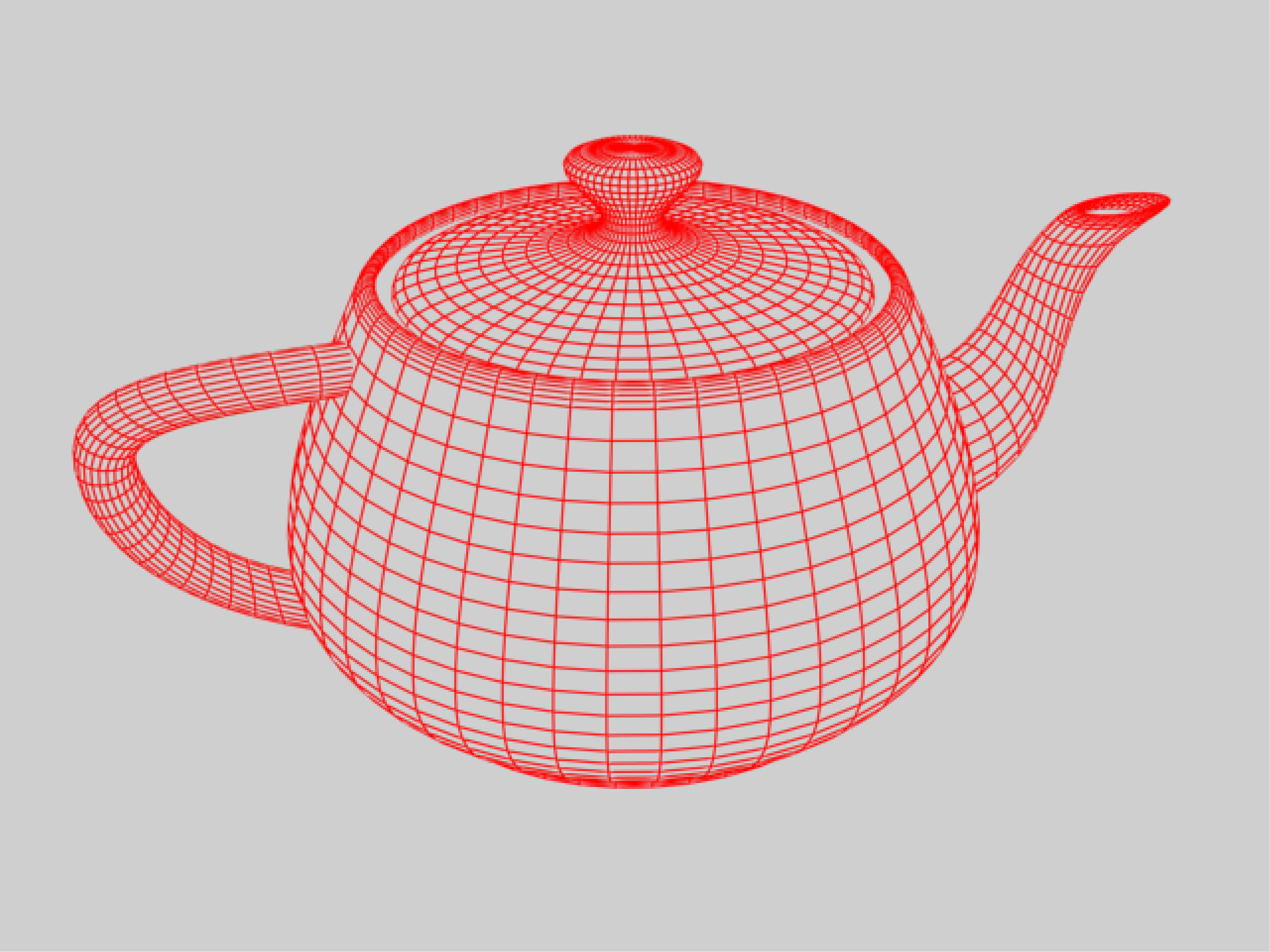
# OBJECT ANTI-ALIASING: LINE



# AND WE DRAW WIREFRAMES!

**Not satisfied?**
**What comes next...**

# VISIBILITY



# TEXTURE

# SHADING

# SHADOWS