CMP205: Computer Graphics

Lecture 4: Viewing and Projection

Ahmed S. Kaseb
Fall 2018

kol object lazm ykon leh origin
mokhtlf 3n el origin bta3 el camera.
el origin bta3hom da bysa3dny eny a7otohom fe mkanhom el mazbot fl scene
lakn el origin bta3 el camera bysa3dny eny a7dd el view bta3 el object el ana 3auz abos mn 3eneh

Slides by: Dr. Mohamed Alaa El-Dien Aly

Agenda

el 5twat el bnmshy 3leha 34an ne2dr nersm object 3l screen

1. eny an2l el verticies bta3t el 3D model lel mkan bta3hom with respect to the origin.
2. eny a3rdhom with respect to the view of the eye bta3t el camera 34an nzbot el view.

----

kol el fat da kan fl 3D system, lakn ehna mehtagen n3rdhom 3la shasha baa el hya aslun 2D

----

3. b3d keda b2a bn3ml projection lel 3D models 3la el shasha 3n tre2 eny a3ml projection.
4.

- Viewing
- Projections
  - Orthographic
  - Perspective
- Transformations Pipeline

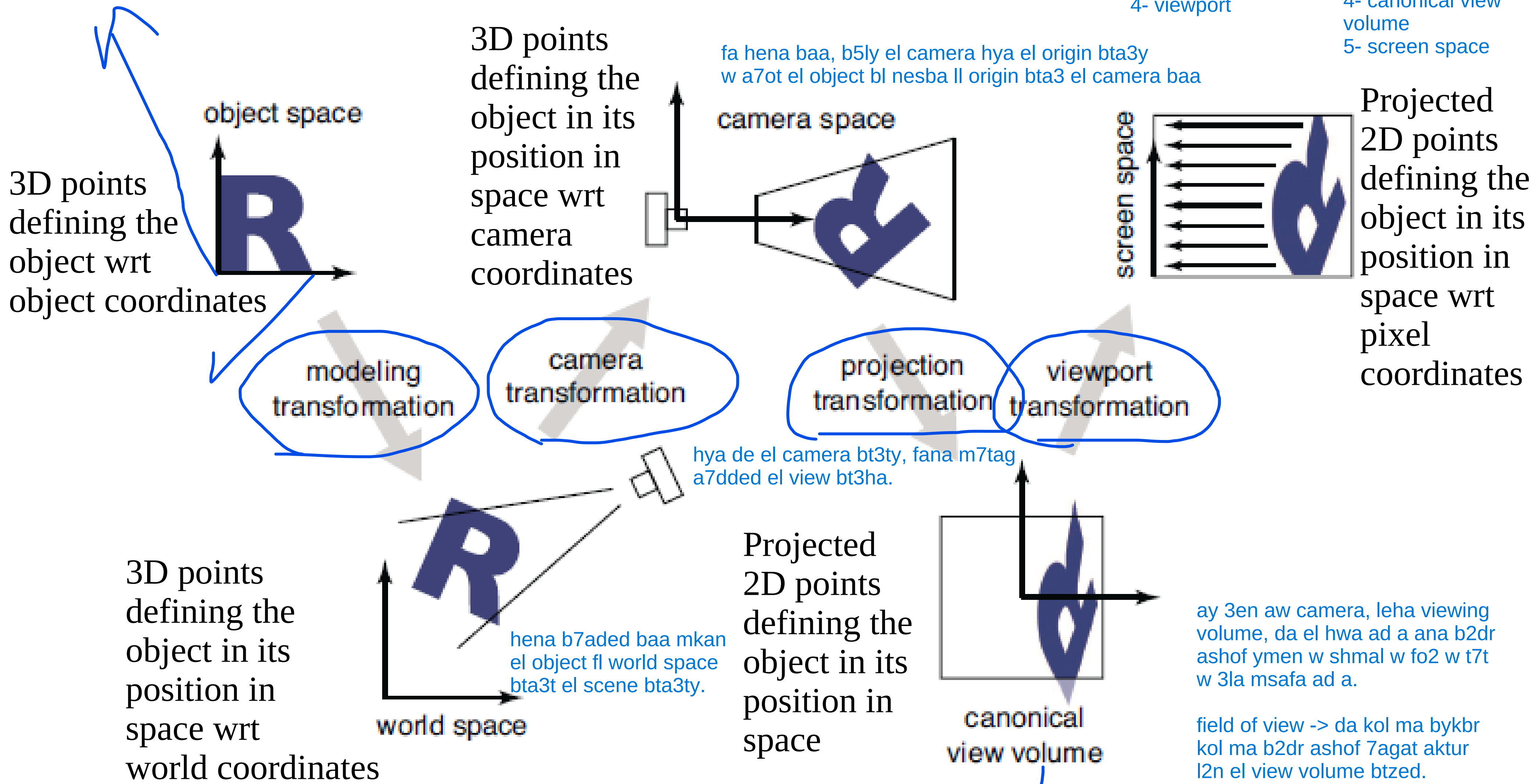
Acknowledgment: Some slides adapted from Steve Marschner and Maneesh Agrawala

3D Viewing

hwa da el origin bta3 kol object.

4 transformations
1- modeling
2- camera
3- projection
4- viewport

5 spaces
1- object space
2- world space
3- camera space
4- canonical view volume
5- screen space



ay 3en aw camera, leha viewing volume, da el hwa ad a ana b2dr ashof ymen w shmal w fo2 w t7t w 3la msafa ad a.

field of view -> da kol ma bykbr kol ma b2dr ashof 7agat aktur l2n el view volume btzed.

intermediate step to simplify the process of conversion from camera space to the screen space to be able to apply the projection.

de nfs el fekra bta3t el wide angle camera, enk lama btgbha, hya el field of view bta3ha akbur, fa btgeb kol el nas.

Convert from 3D points in space to 2D points on screen

Projections

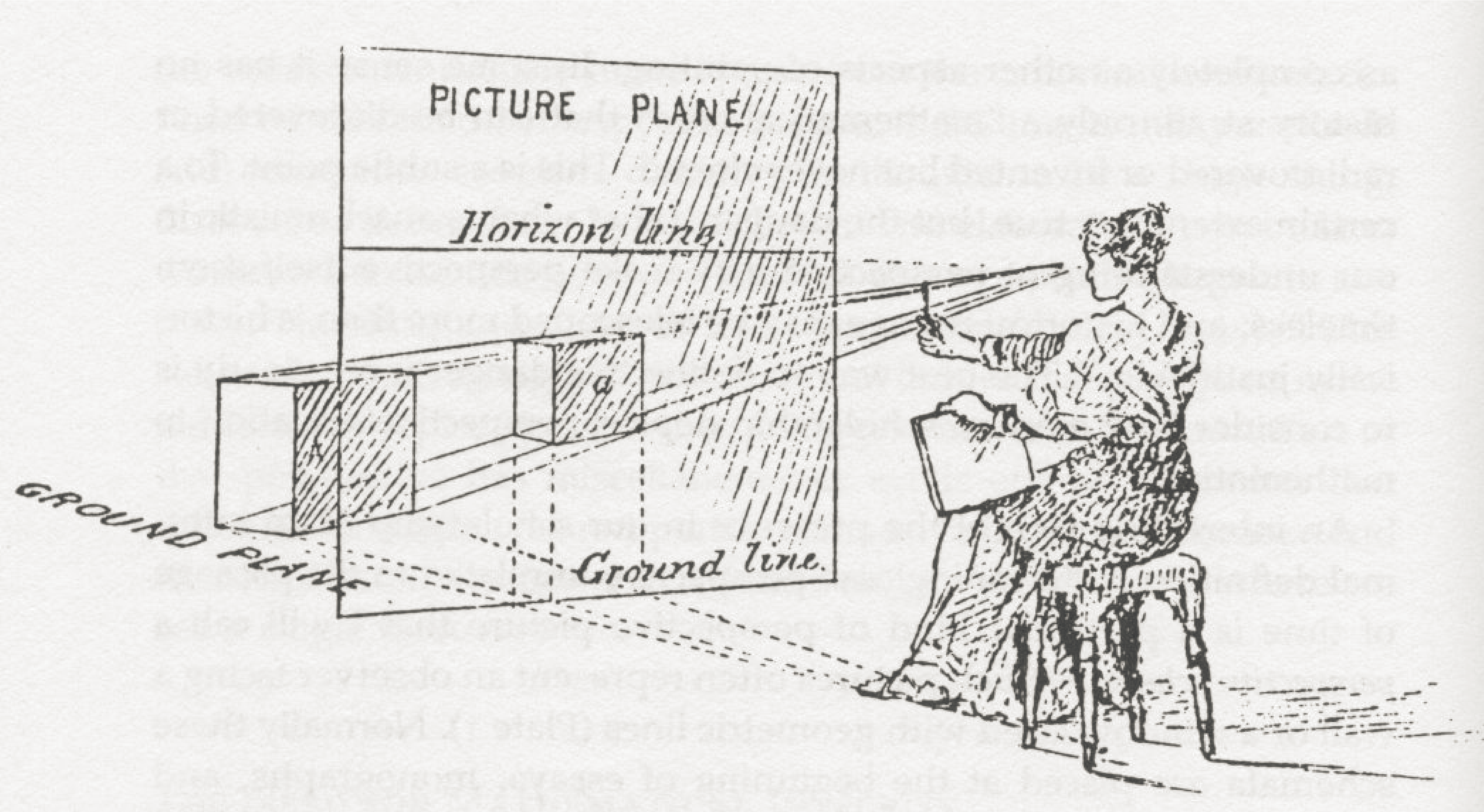
tb from 2D to 3D esmo a?

3D reconstruction

banzl one dimension
from 3D to 2D

aw 2D to 1D w hakaza

transformation is changing
positions in the same level
3D -.> 3D
2D -> 2D

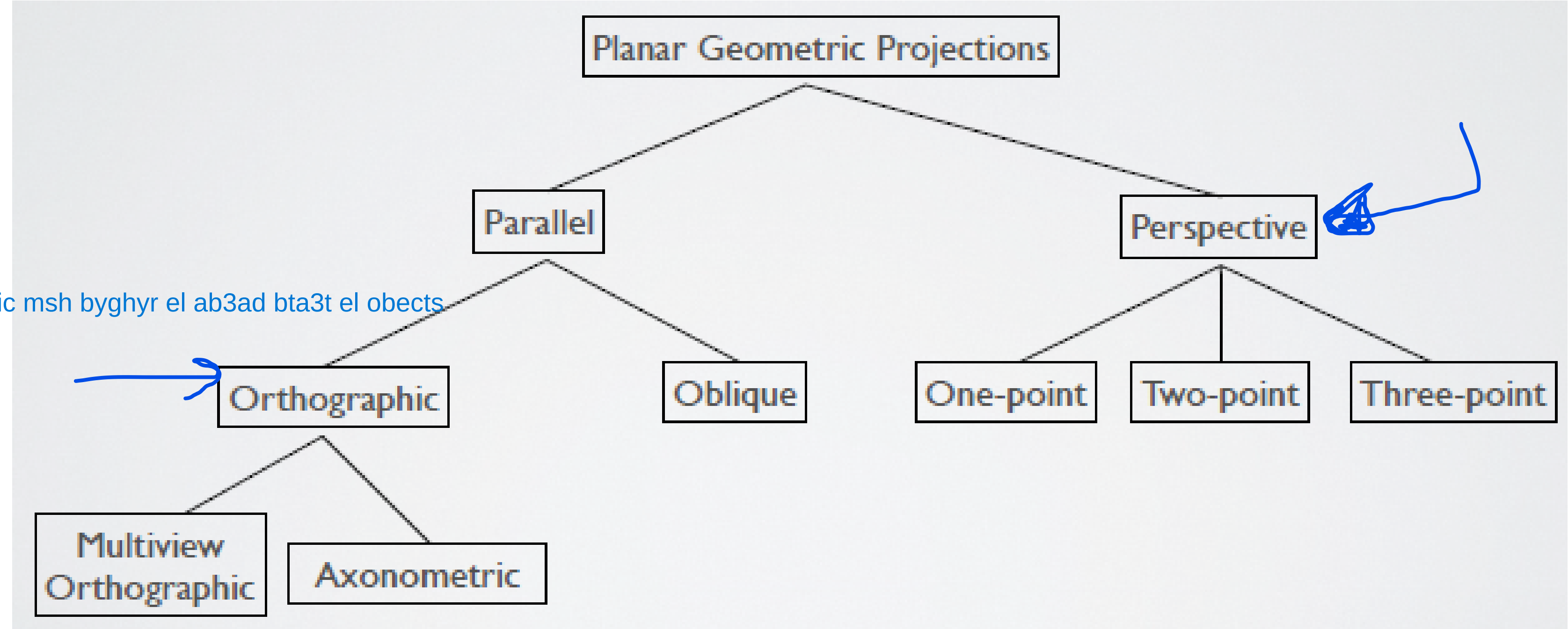


Project points in the world onto a plane

Projections

lama ykon el camera leha field volume
el projection byb2a esmo prespective
da bykhlene ashof el haga el oryba kbera wl b3eda soghyra.

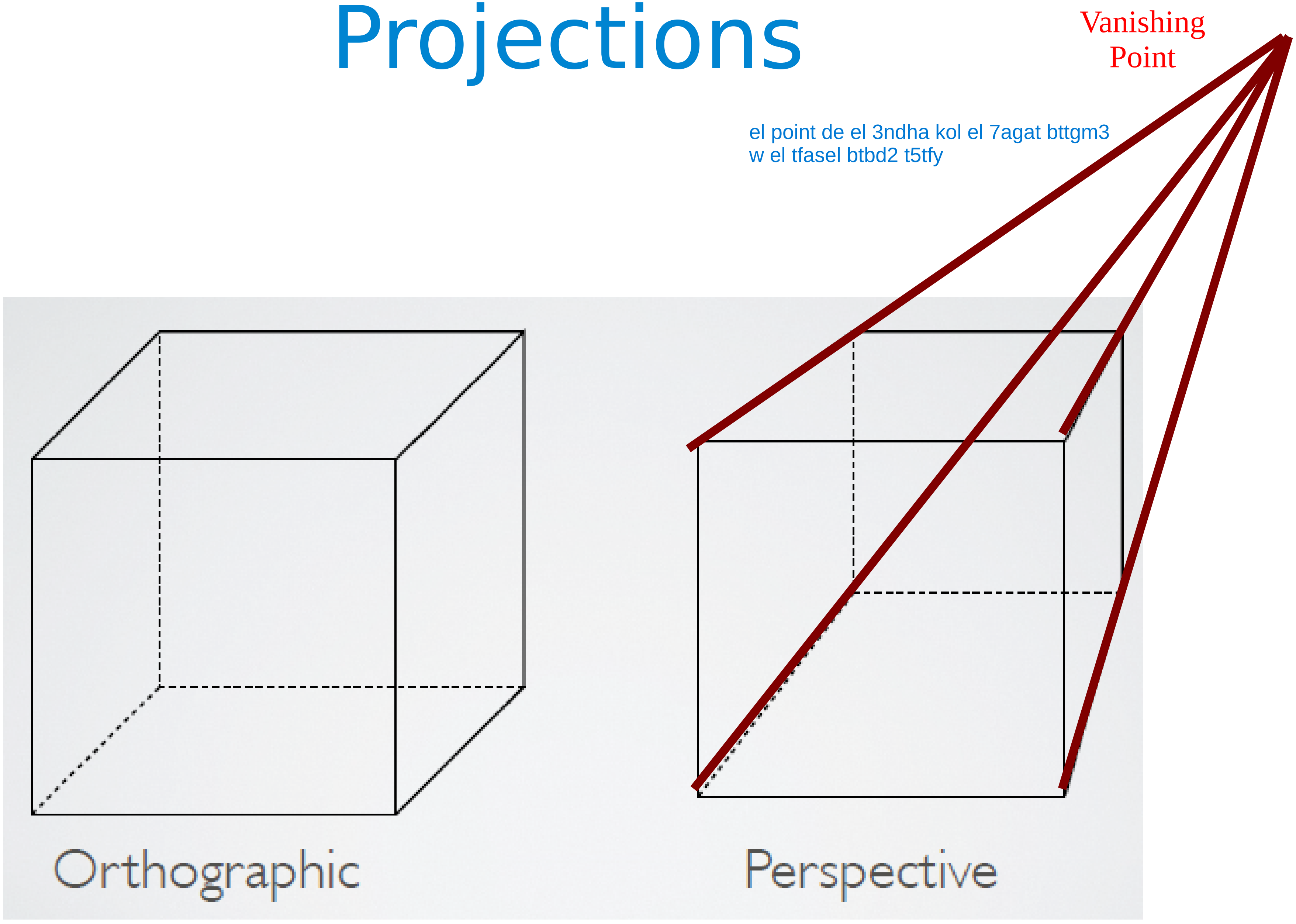
el orthographic msh byghyr el ab3ad bta3t el obects



There are many kinds of projections, but we will only talk about two types...

Projections

el point de el 3ndha kol el 7agat bttgm3
w el tfasel btbd2 t5tfy



Parallel lines remain parallel

Parallel lines intersect at a *vanishing* point

el orthographic lw fe 7agat kant parallel, hya btfdl dayman parallel,
da natega l2n ana b7afz 3la el dimensions

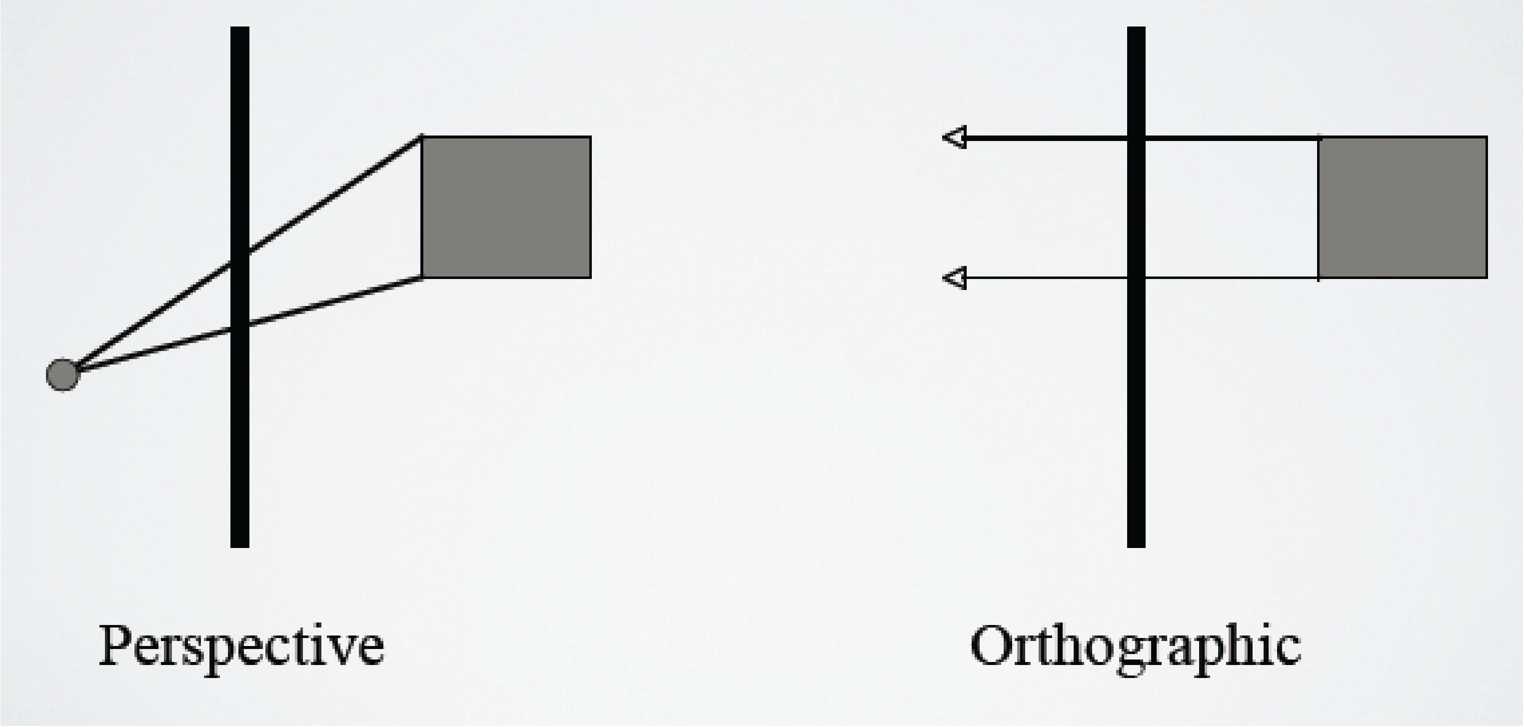
Perspective Projection

Railway tracks intersect at a vanishing point

[www.edupic.net/math_pics.htm]

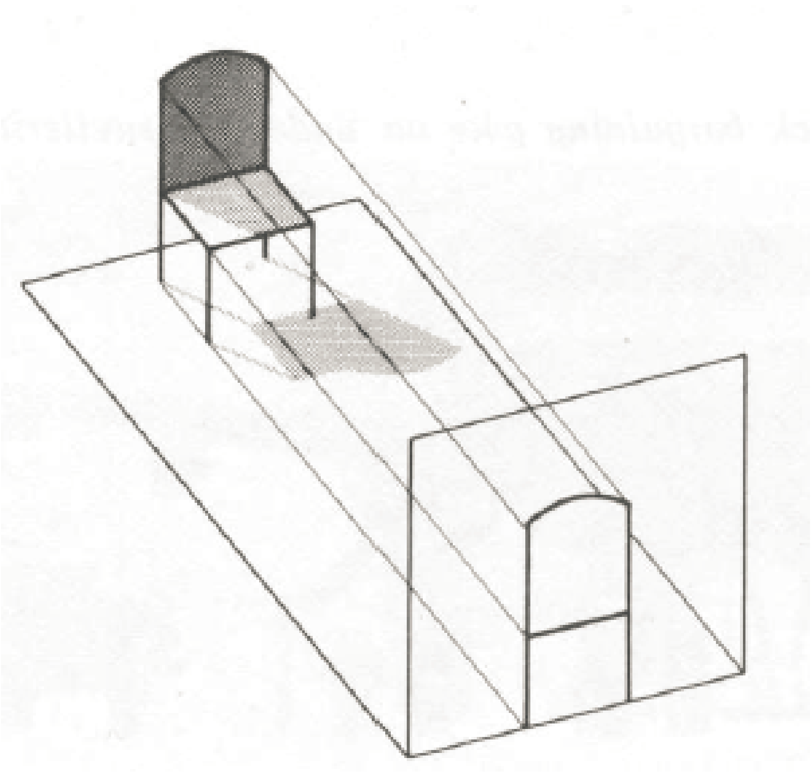
Projections in 2D

In 2D, project is done on a *projection line*



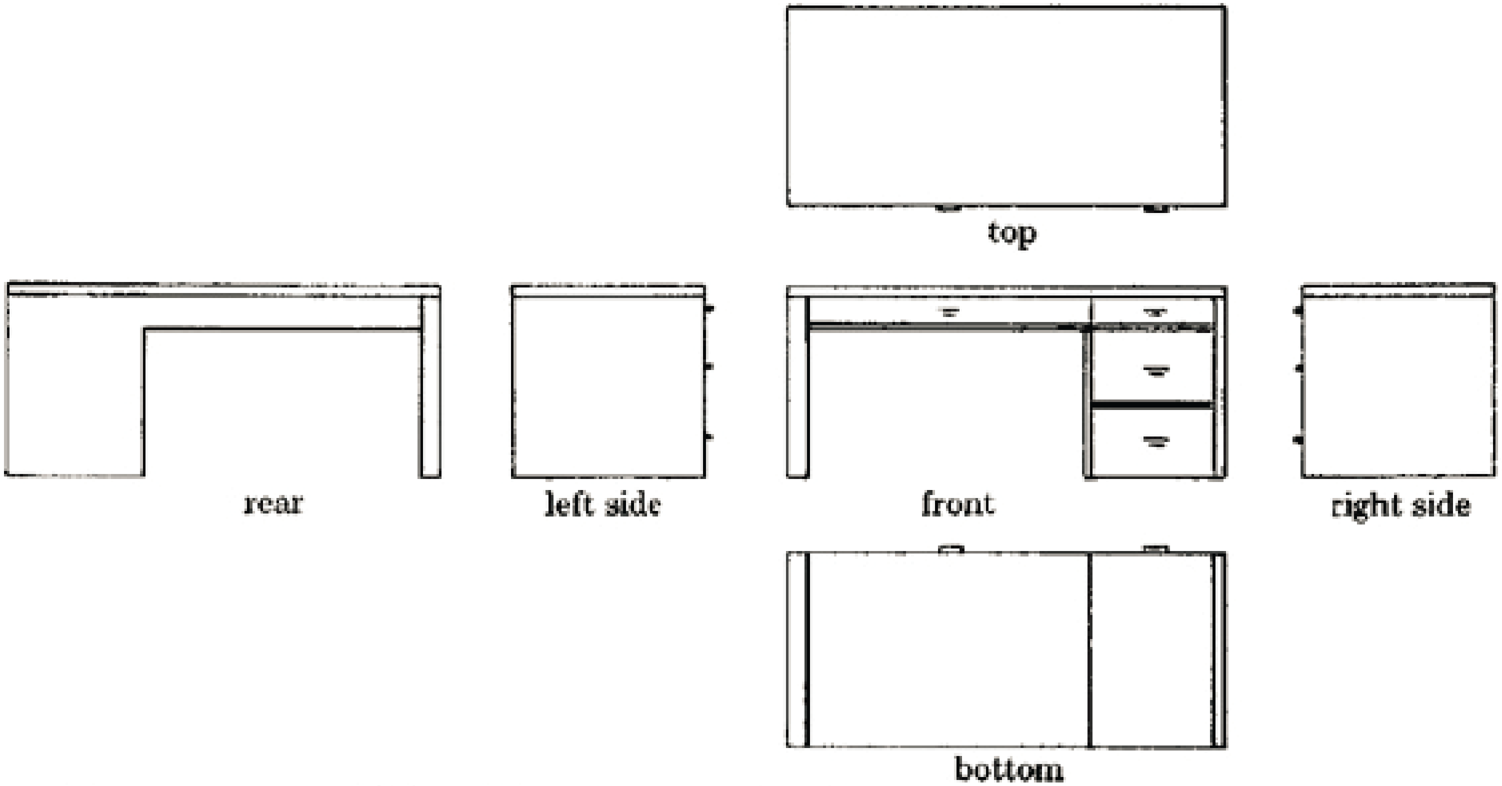
Orthographic Projection

el prespective wl orthographic msh byfr2o fe awl 3 steps, lakn byfr2o fe akher 2 steps bs.



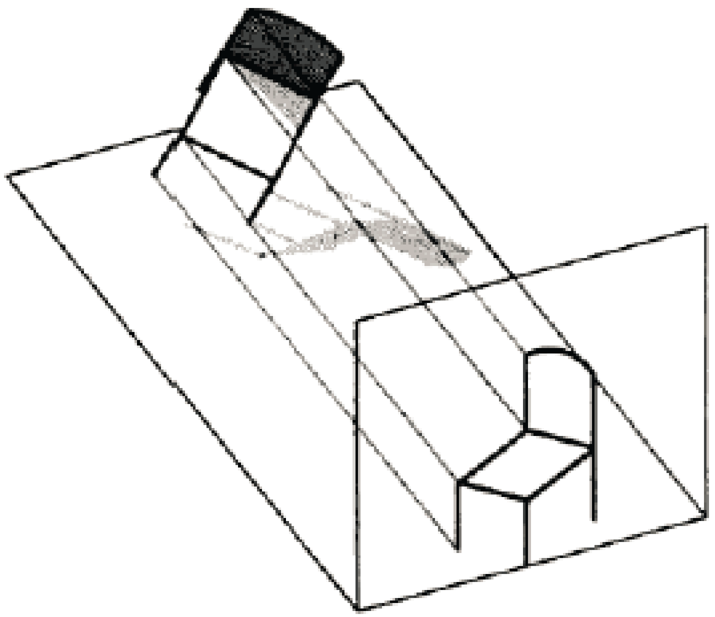
- Projection plane parallel to a Coordinate plane
- Projection direction perpendicular to projection plane

Multiview Orthographic



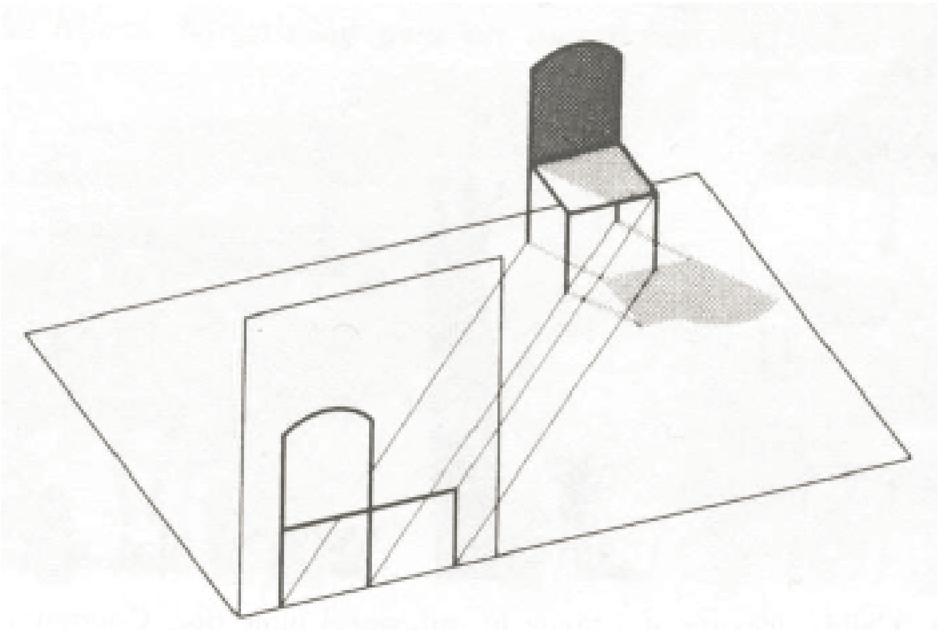
Similar to the engineering drawing of the preparatory year!

Off-Axis Projections



Axonometric Projection

Projection plane not parallel to coordinate planes



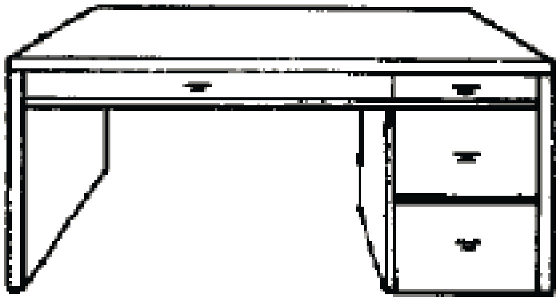
Oblique Projection

Projection lines not perpendicular to projection plane

Perspective Projection

One-Point Perspective

Projection plane parallel to
a coordinate plane



one-point

Two-Point Perspective

Projection plane parallel to
a coordinate axis

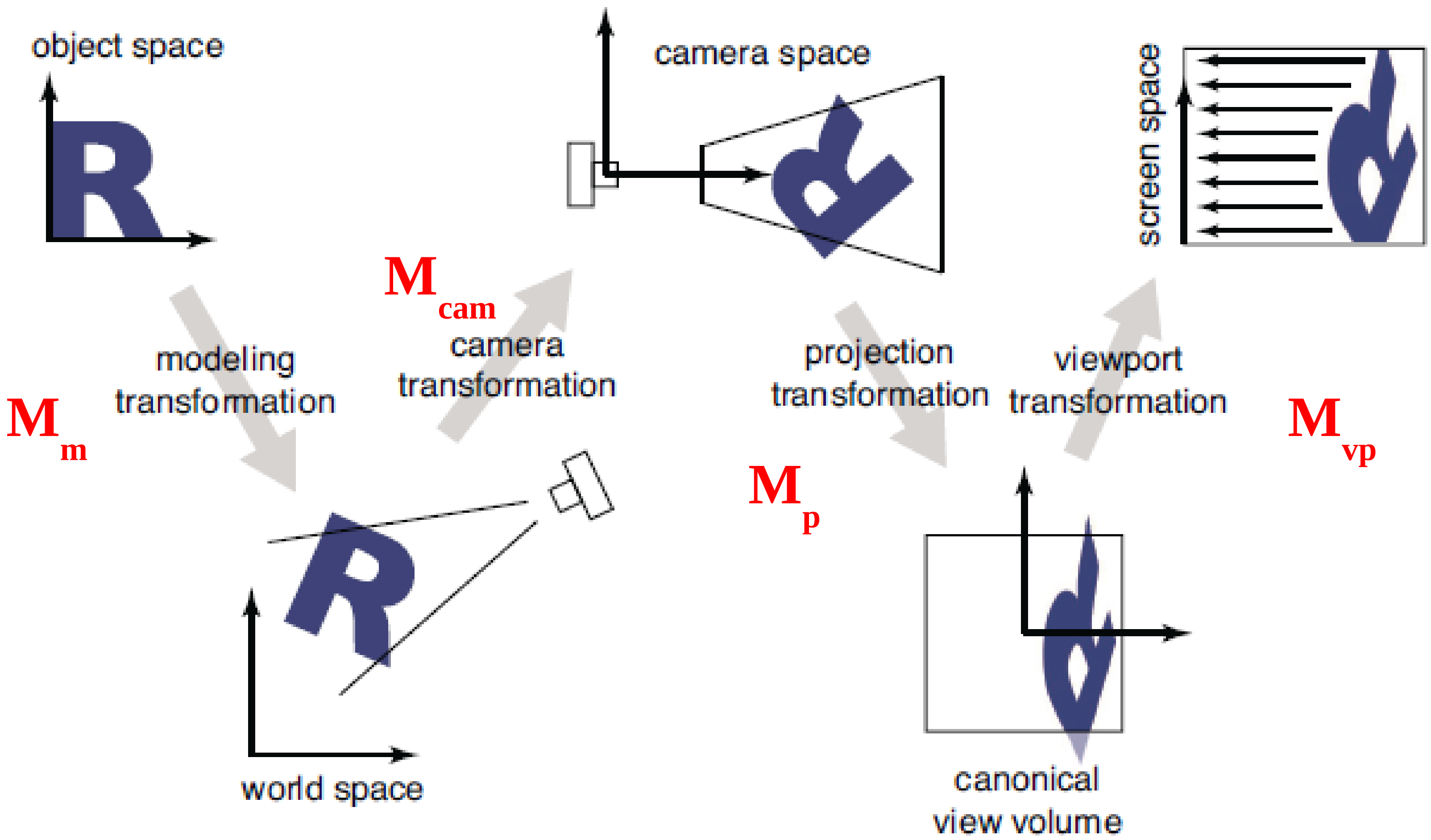
two-point

Three-Point Perspective

Projection plane not parallel to
any coordinate axes

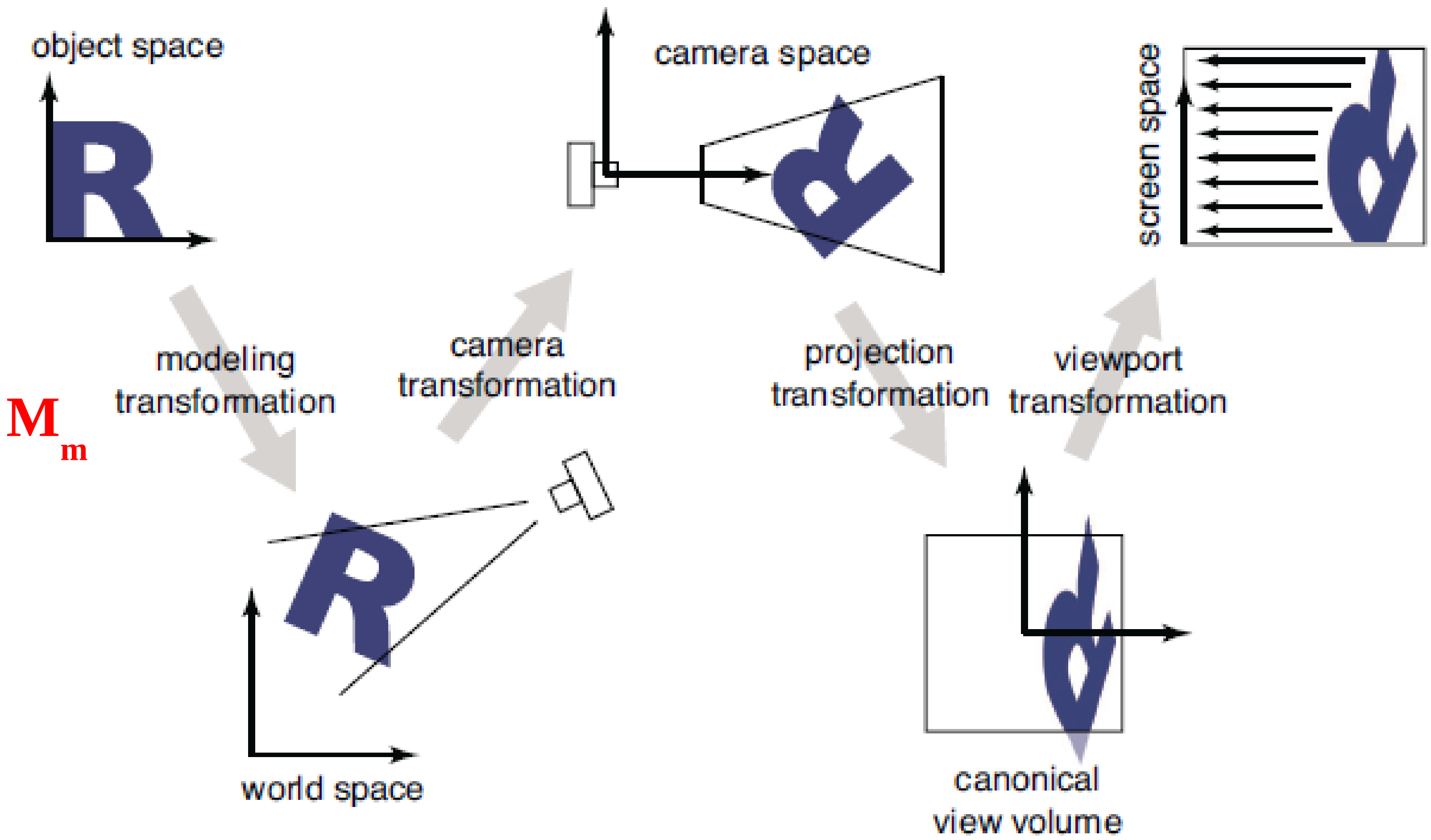
three-point

Transformations Pipeline



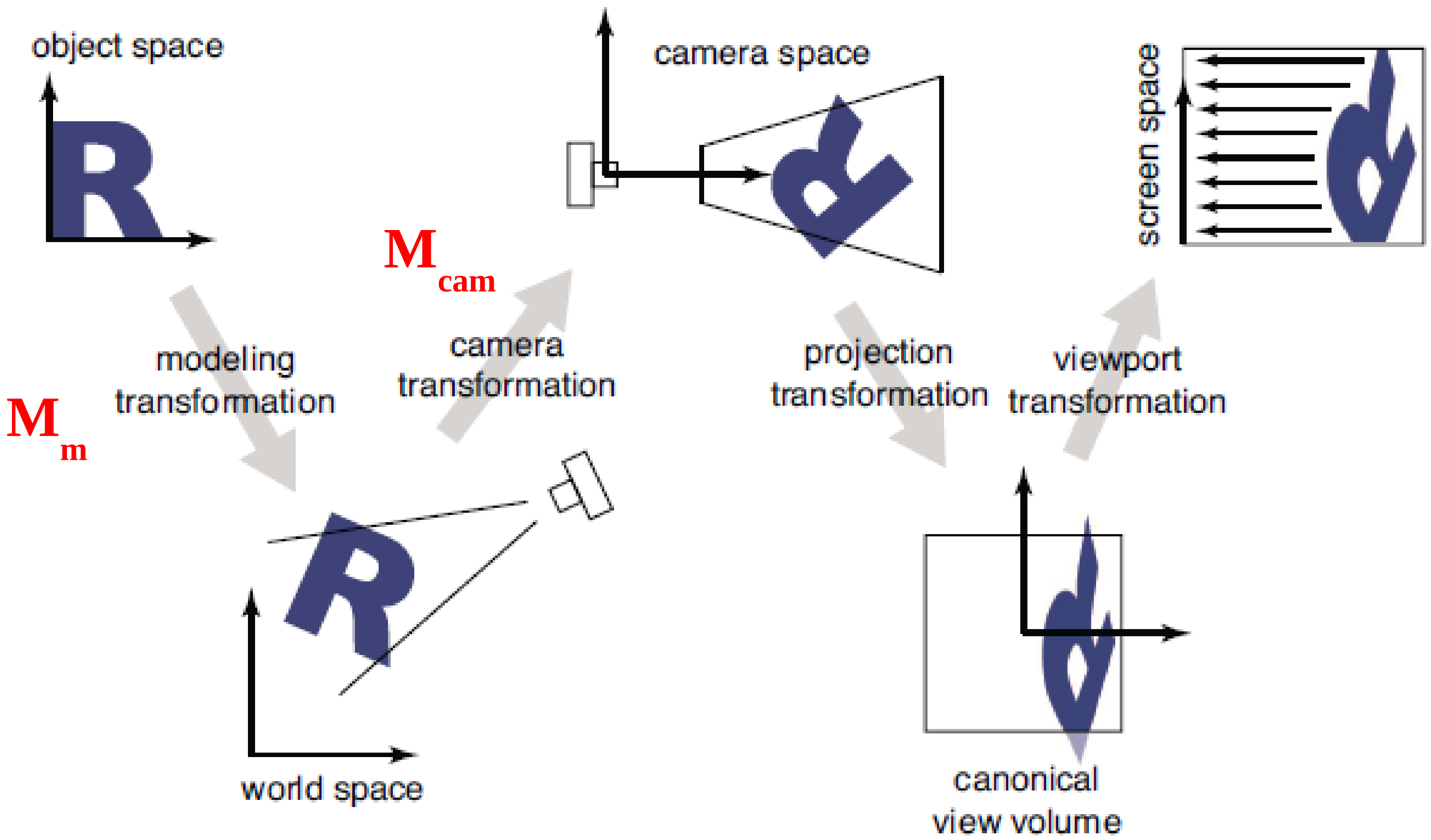
Converts 3D points in object space to 2D pixels on the screen through a series of transformations

Modeling Transformation



Converts 3D points from the object coordinates to the world coordinates
i.e. place the object in the world

Camera Transformation



Converts 3D points from the world coordinates to the camera coordinates
i.e. place the origin at the camera center

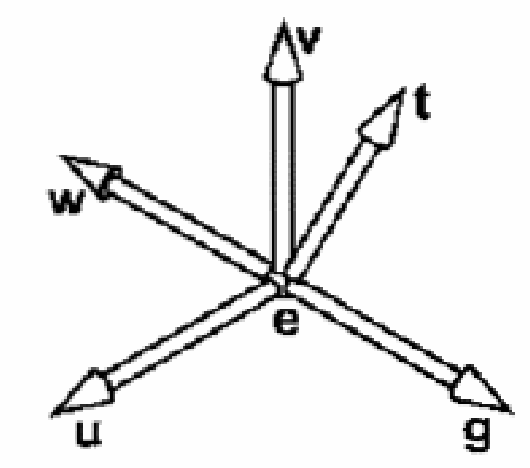
Arbitrary Views

Camera frame is usually defined by:

e : eye position

g : gaze direction   gaze ----> bases ezay

t : view up vector   el angle bta3t el camera



Using this information, we can construct three coordinate axes centered at e as follows:

$$w = \frac{-g}{\|g\|}$$

hwa byftrd en el w dayman fl -g

$$u = \frac{t \times w}{\|t \times w\|}$$

cross product to get a perpendicular axis on the w

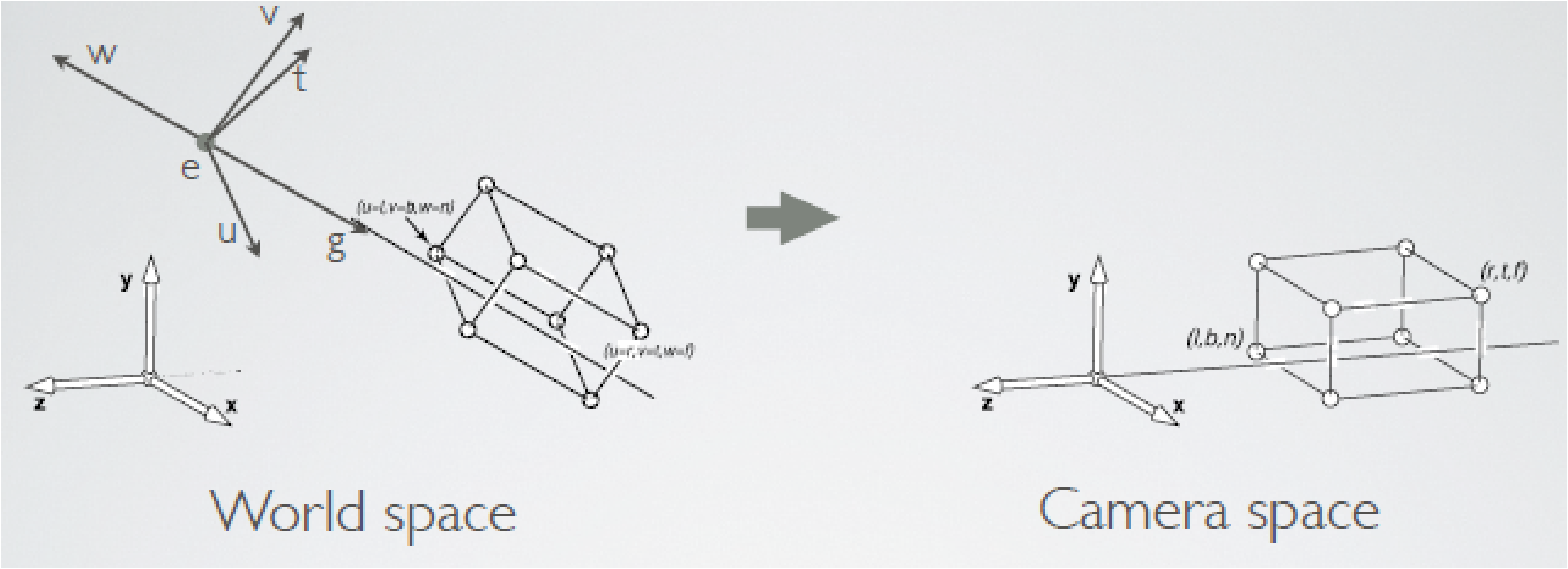
$$v = w \times u$$

el w,u,v dol el axis el bn3mlhom 34an nkwn el world bta3 el camera

awl haga bn7sb el w 3n tre2 enna ngebha -g

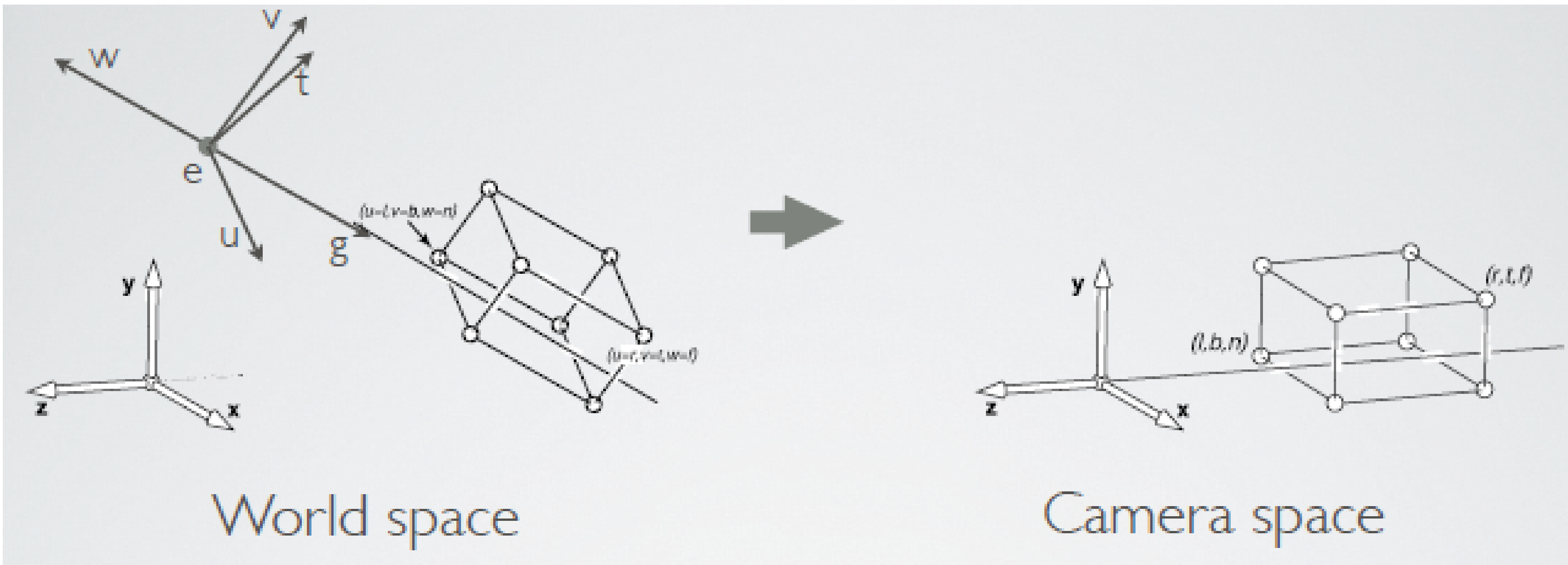
w b3d keda bngeb 2 axes perpendicular 3leha.

Camera Transformation



Convert from World Coordinates to Camera Coordinates

Camera Transformation



camera origin

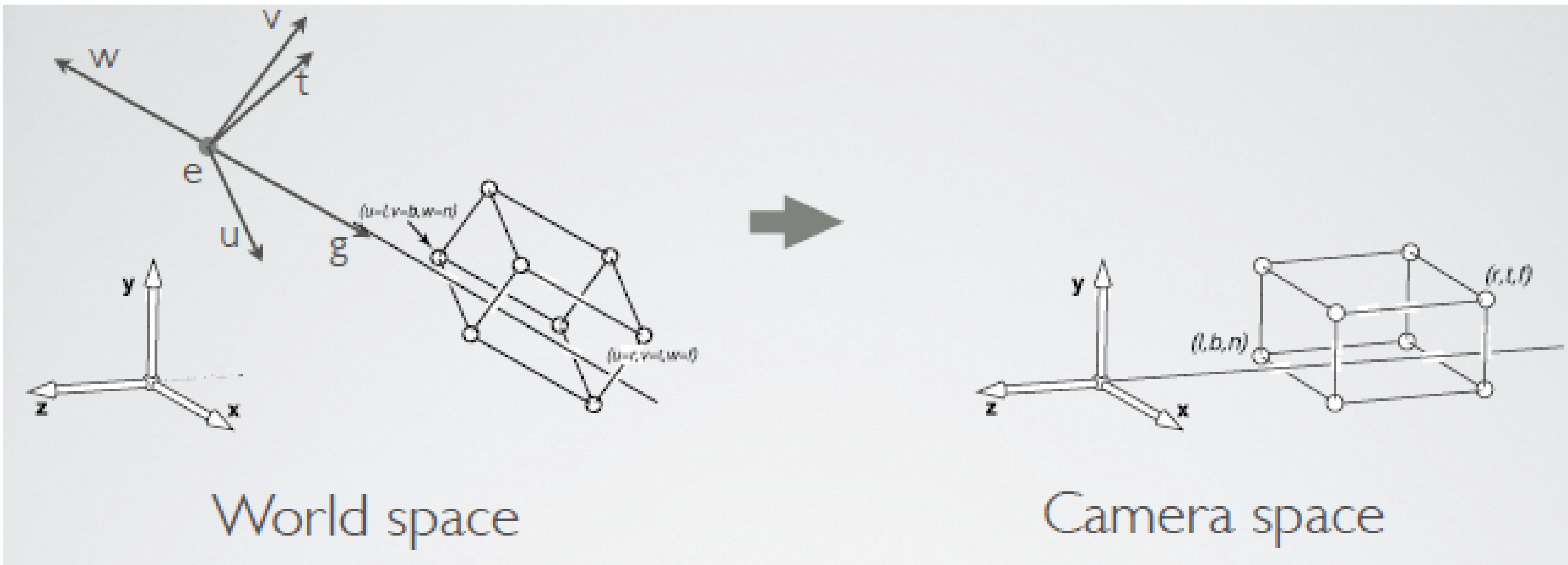
e -> eye position

$$M_{cam} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u & v & w & e \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

Aligns Camera Coordinates with World Coordinates

Moves Camera to World Origin

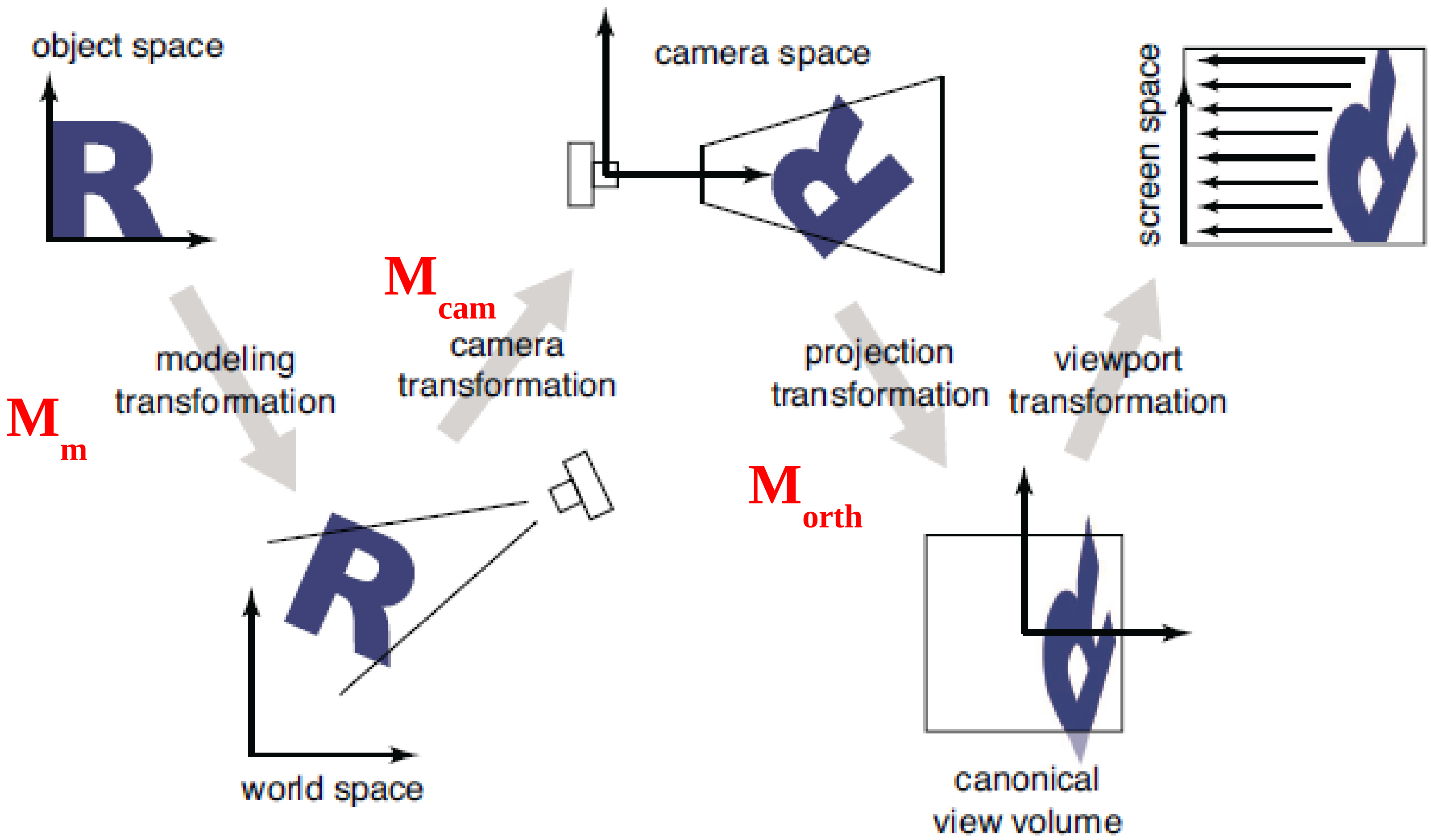
Camera Transformation



$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Converts coordinates from the world frame to the camera frame

Projection Transformation

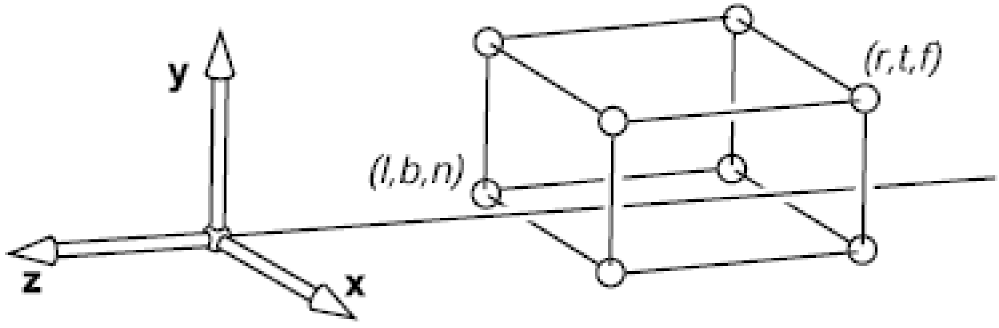


Converts 3D points in the camera space to “2D” points in the *canonical* view volume

Orthographic Projection

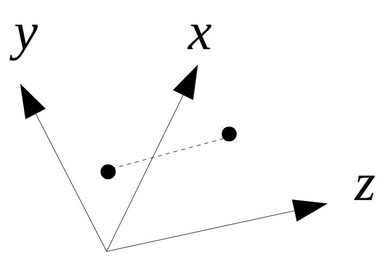
- Consider a camera at the *origin* looking at the –ve *z* direction
- We want to project the image on the *near* plane
- We want to project only the points defined in the "view volume" defined by (*left*, *bottom*, *near*) and (*right*, *top*, *far*)

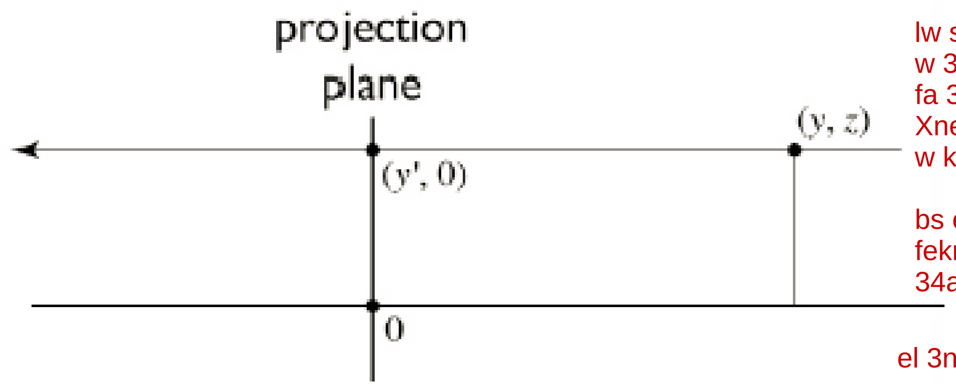
el orthograpghic view
bytl3 motawazy
mostatelat l2n kol el
projection lines byb2o
parallel lb3d



Orthographic Projection

First consider the *y* coordinate by looking along the +ve *x* axis





lw shghalen orthographic projection
w 3ndna fl 3d verticies (x,y,z)
fa 34an n3ml projection
Xnew = Xold bzbt w nfs el klam ll Ynew = Yold
w kol el bn3mllo enna bn3ml neglect lel z

bs ehna msh hanhmlha tmaman, ehna bs kol el fekra, han3rf mn el z anhy a2rb w anhy ab3d, 34an a3rf men el hy3ml overwrite 3l tany.

el 3ndo z a2rab, hwa el hy3ml overwrite 3l b3ed

How do we get y' given (x, y, z) through projecting on the xy plane?

$$y' = y$$

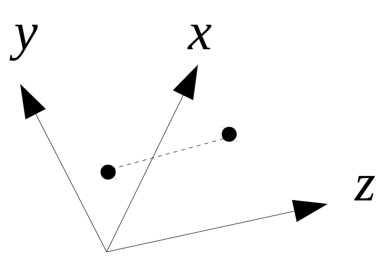
Similarly,

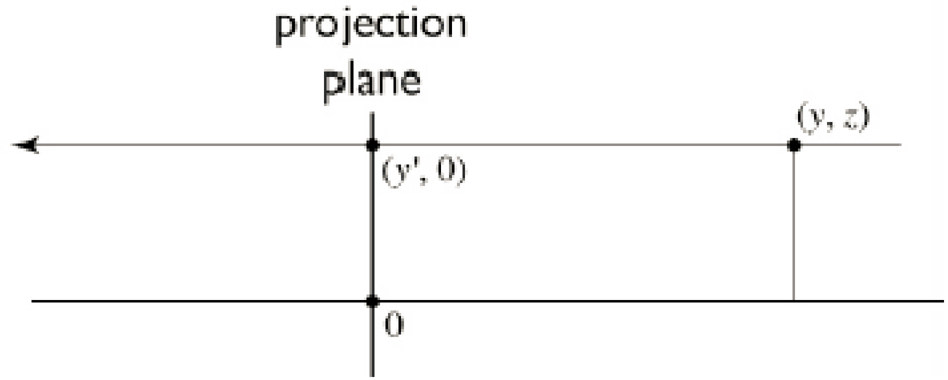
$$x' = x$$

Drop z-coordinate!

since we use -g, so the element with less absolute z will be more closer.

Orthographic Projection





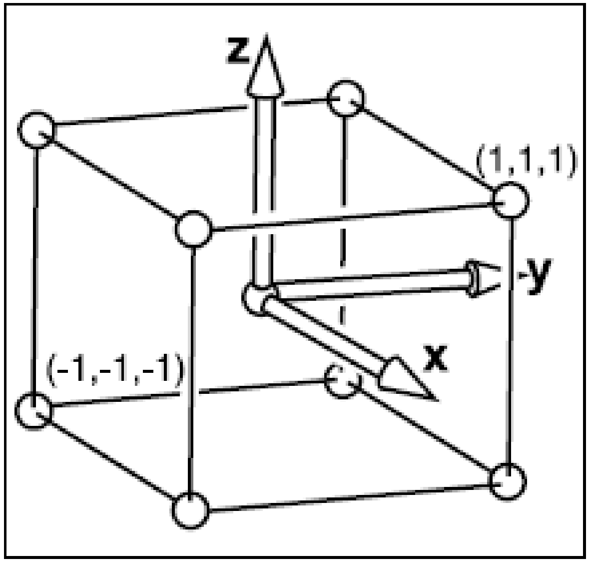
How do we get (x', y') given (x, y, z)?

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

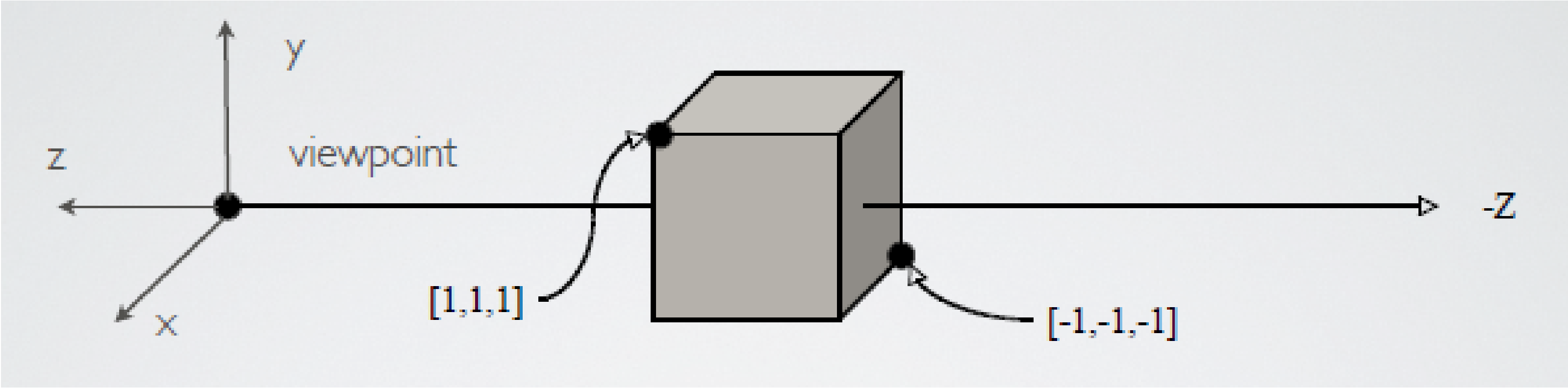
Drop z-coordinate

3D Canonical View Volume

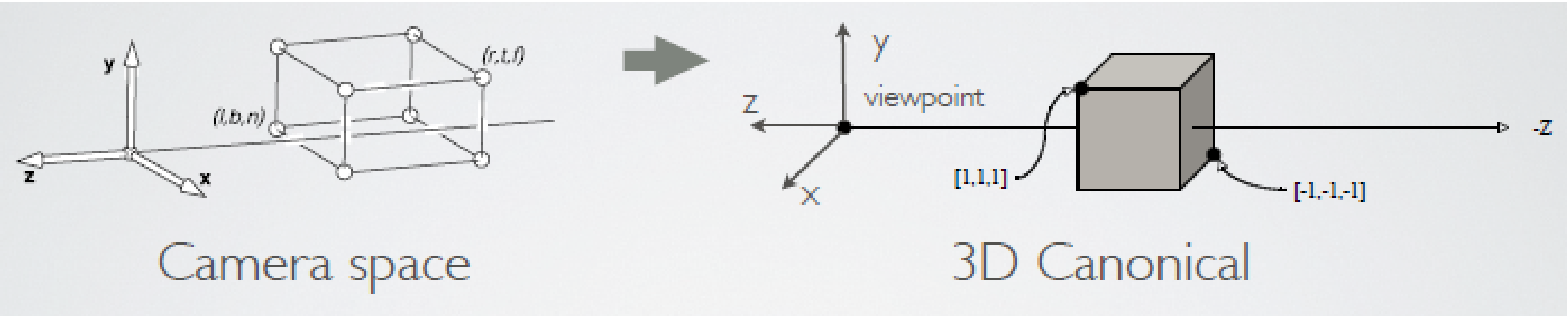
A camera independent volume that extends from –1 to +1 in both the *x*, *y*, and *z* directions i.e. camera at *origin*



windowing



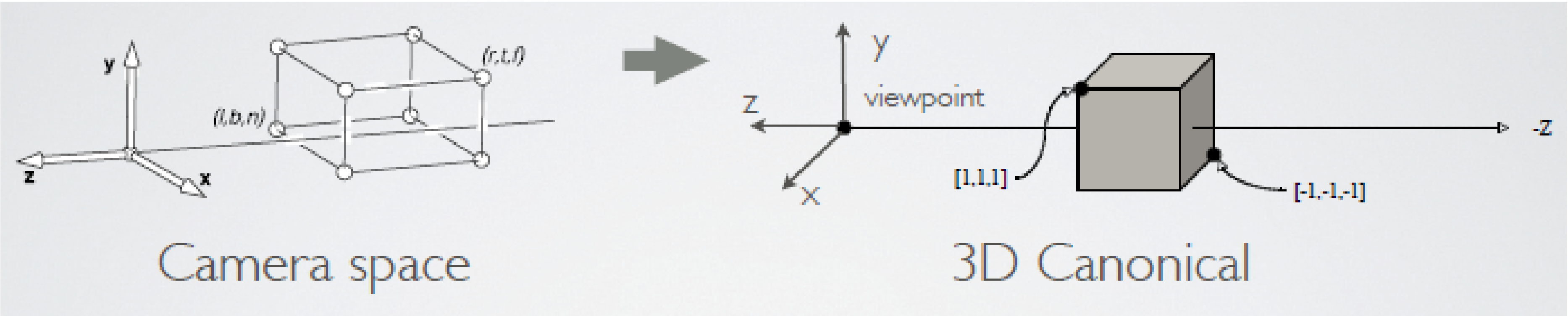
Orthographic Projection



How do we transform from a general view volume defined by (l, b, n) and (r, t, f)?

Windowing Transform!

Orthographic Projection



this make the scaling process

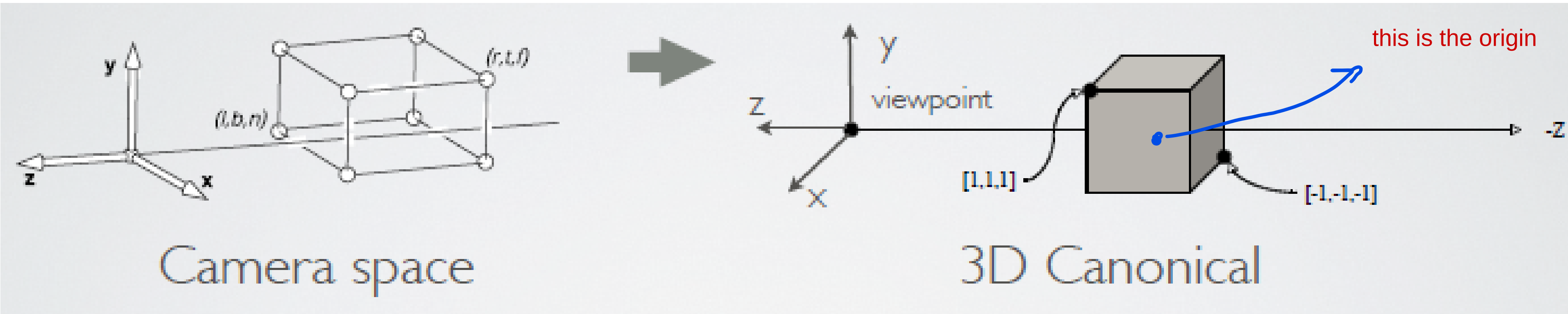
this which make the translation

$$M_{orth} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \frac{-(l+r)}{2} \\ 0 & 1 & 0 & \frac{-(b+t)}{2} \\ 0 & 0 & 1 & \frac{-(n+f)}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, scale the sides to have the right lengths

First, translate so the origin is at the center

Orthographic Projection



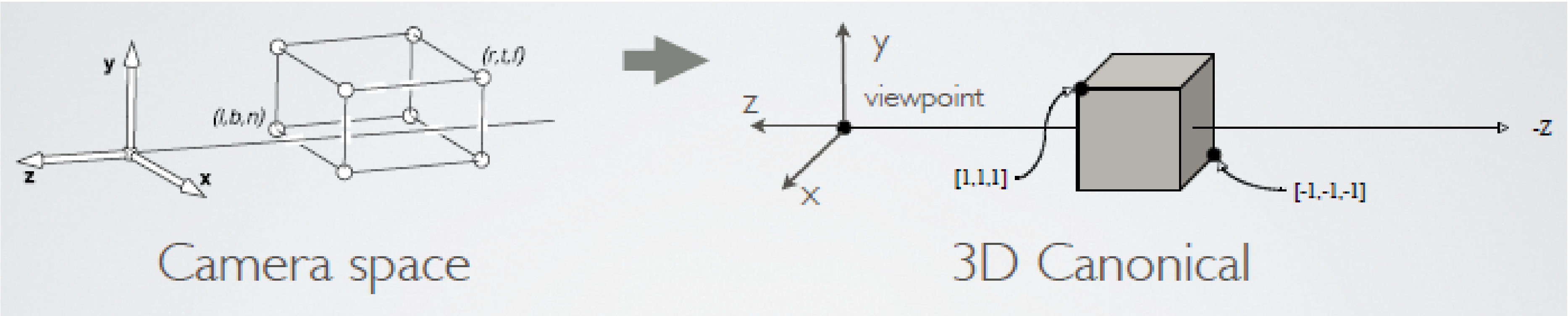
$$
M_{orth} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

el center bta3 el canonical view hwa el origin bta3 el camera

Putting them together

Orthographic Projection

ay orthographic projection
by5ly shakl el object
cuboid.



da el gowa el canonical

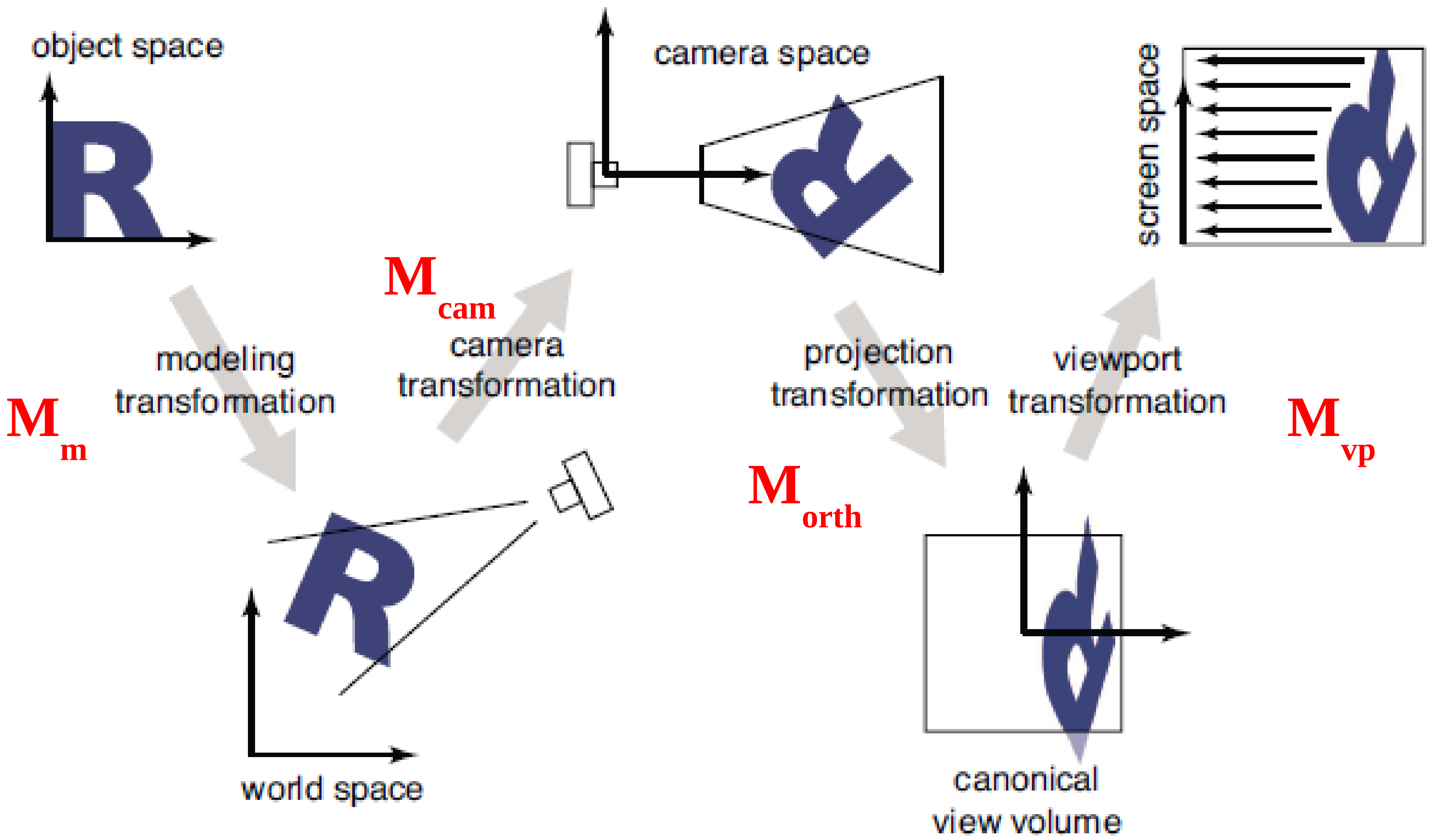
$$\begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ 1 \end{bmatrix}$$

el values lazm teb2a in range = [-1,1]
lw 7aga gt brahom b3ml trim w ba neglect

Why do we keep the *z* coordinate?

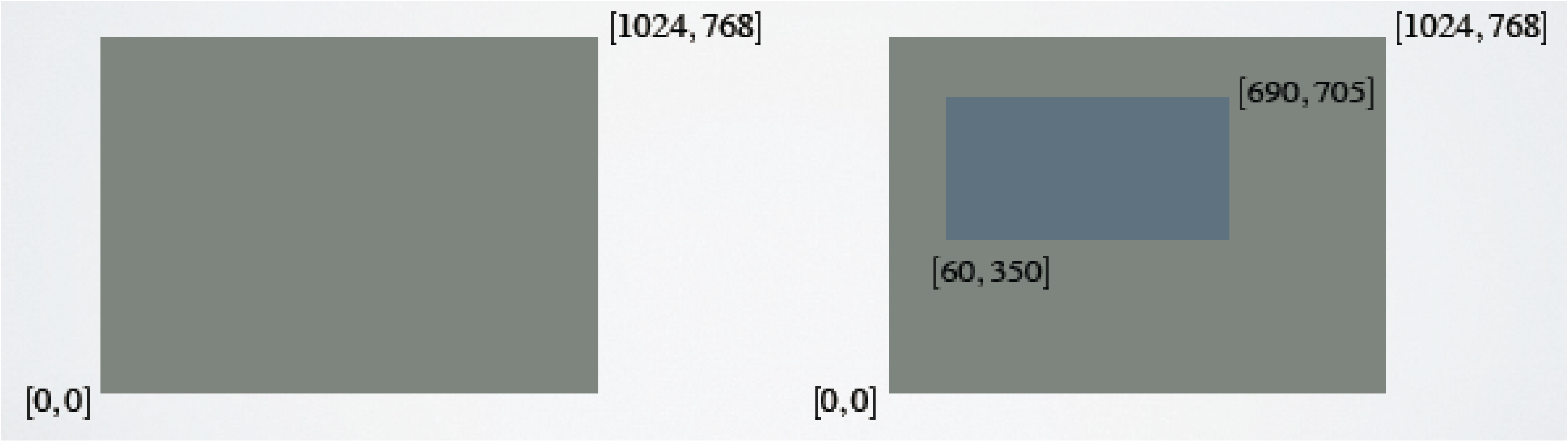
to detect which will overwrite which

Viewport Transformation



Convert from 3D points in canonical space to 2D points on screen

Screen Space

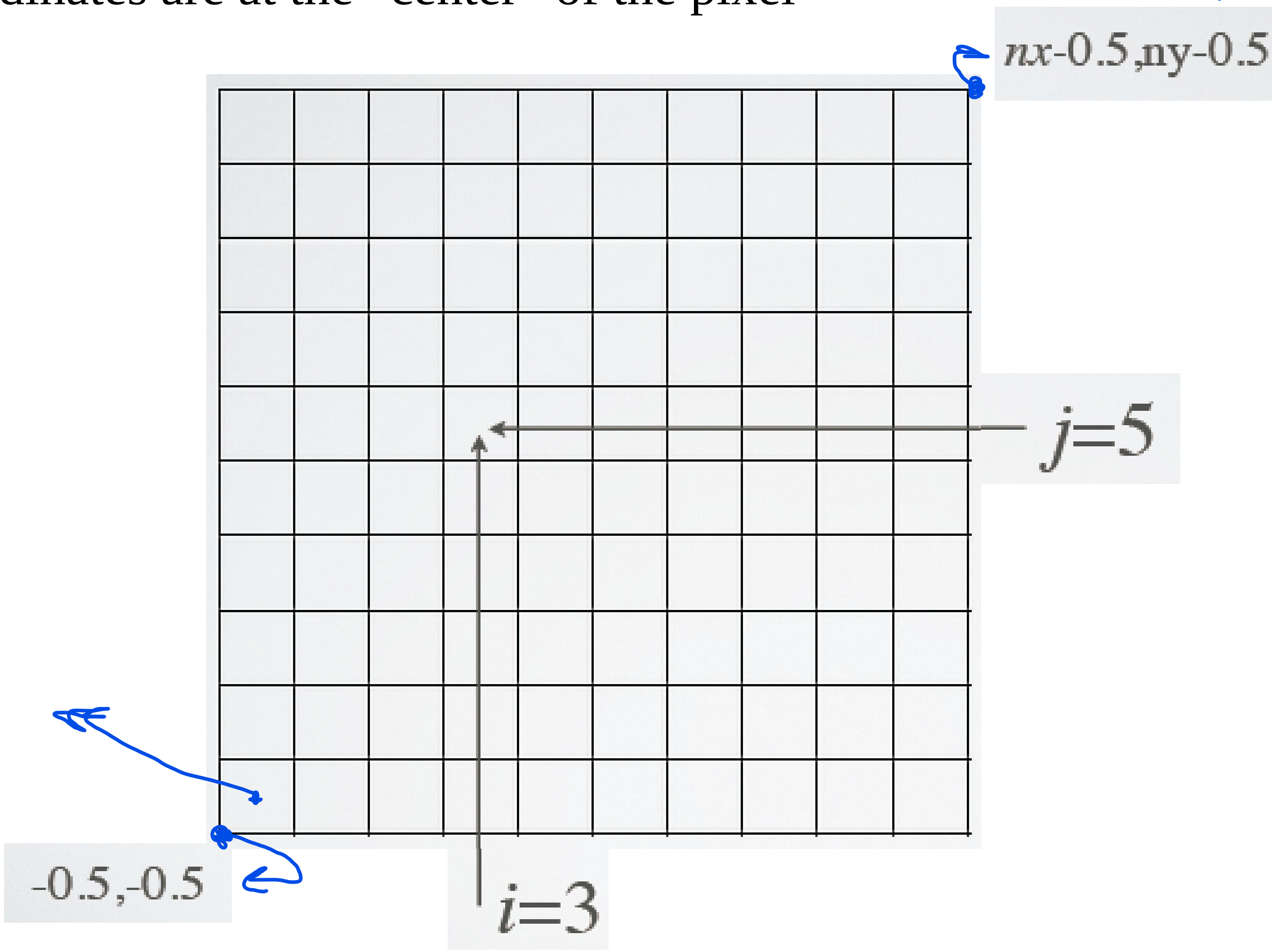


Screen

Viewport i.e. any “window” inside the screen

Screen Space

Pixels coordinates are at the “center” of the pixel



el fekra enk shghal 0 index, fa akhrak aslun (n-1) w (n-1)
la ehna shaghalen 3la el center bta3 el square htl2eh baa (9,9) msln

fa 34an baa tgeb el upper right corner, htzwd nos w nos fhyb2a (9.5, 9.5)
w since en n = 10 yeb2a fe3ln nx - 0.5

fa 0.5 msh bt3br eny nzlt fl y aw shmal fl x, la de el equation el f3lya

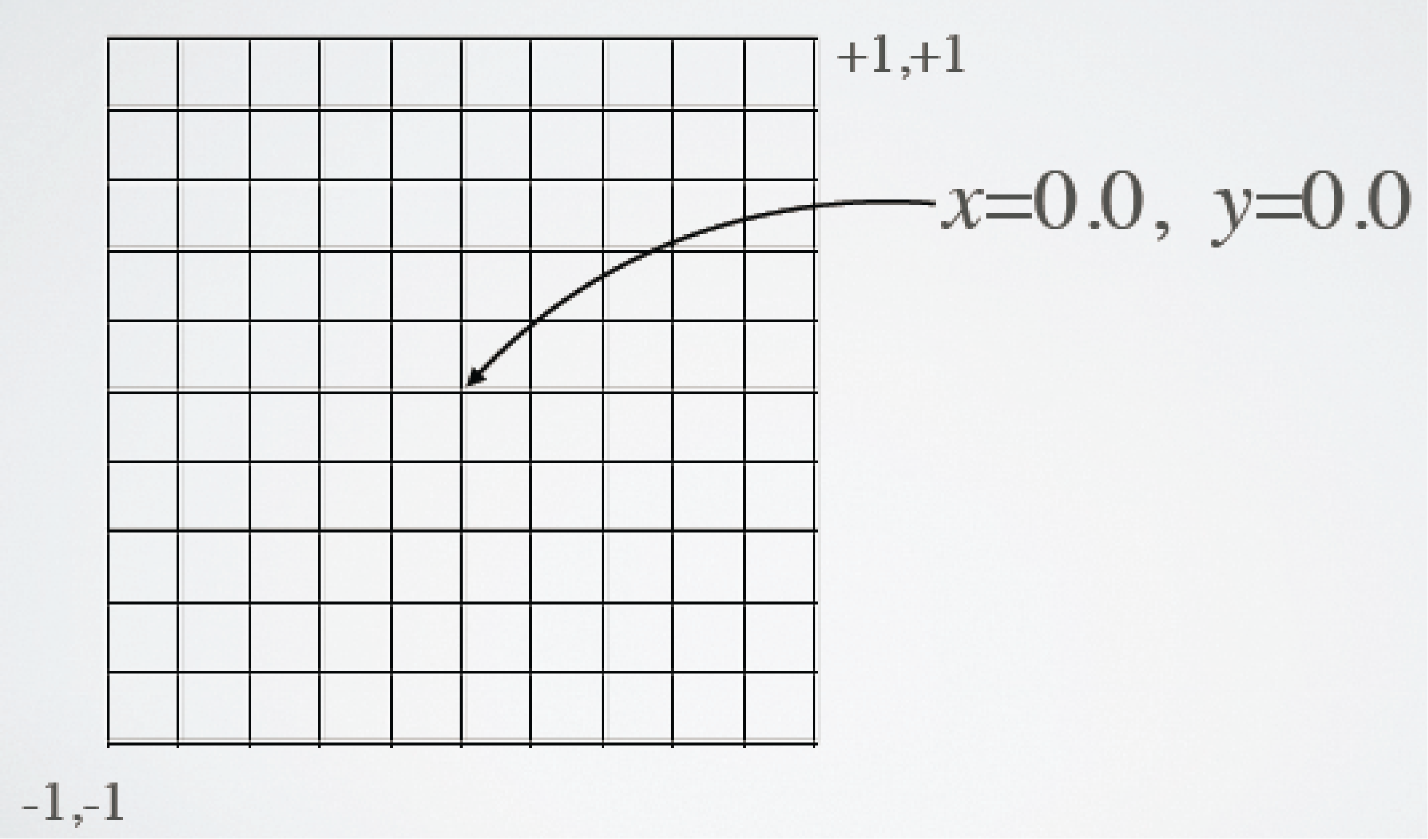
this is the origin (0,0)

mokhtalef 3n el canonical

Screen of width n_x and height n_y pixels

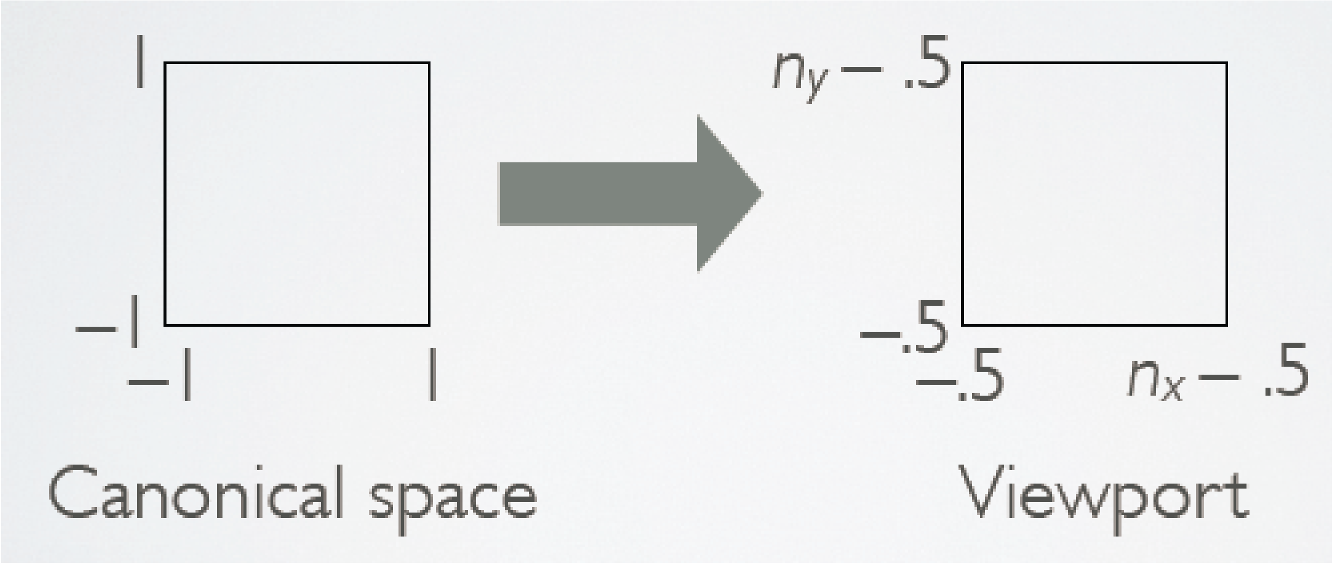
2D Canonical View Space

The *xy* plane of the canonical view volume



2D Viewport Transformation

Converts 2D points from the canonical space to screen space i.e. pixels

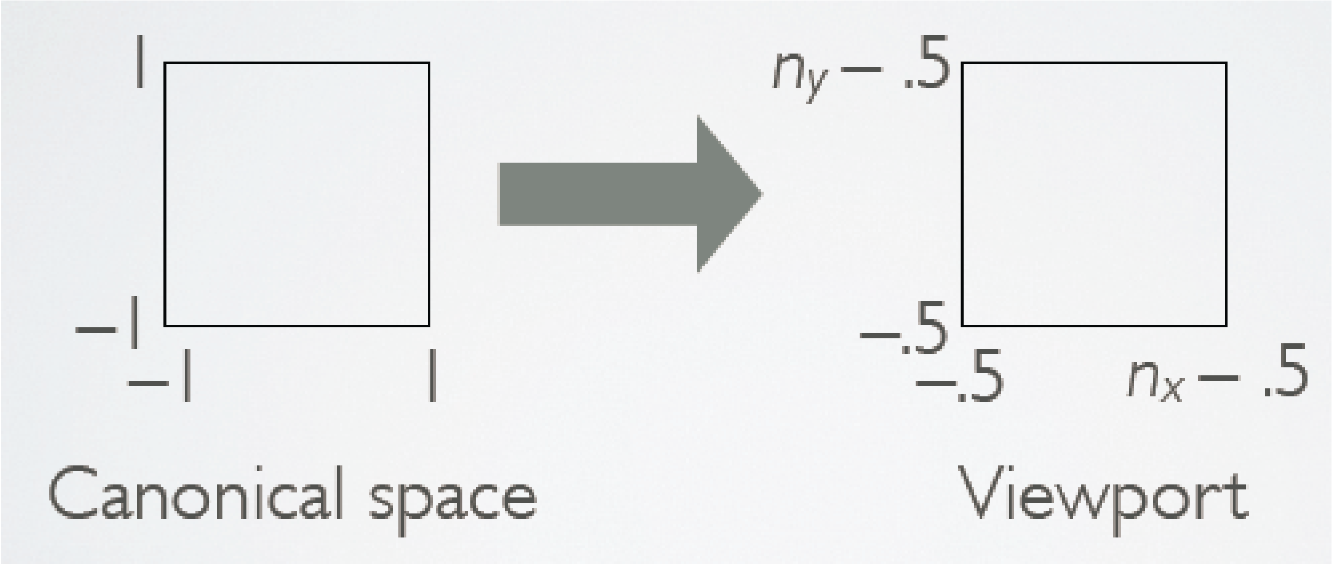


How do we do that?

Another windowing transform!

2D Viewport Transformation

Converts 2D points from the canonical space to screen space i.e. pixels

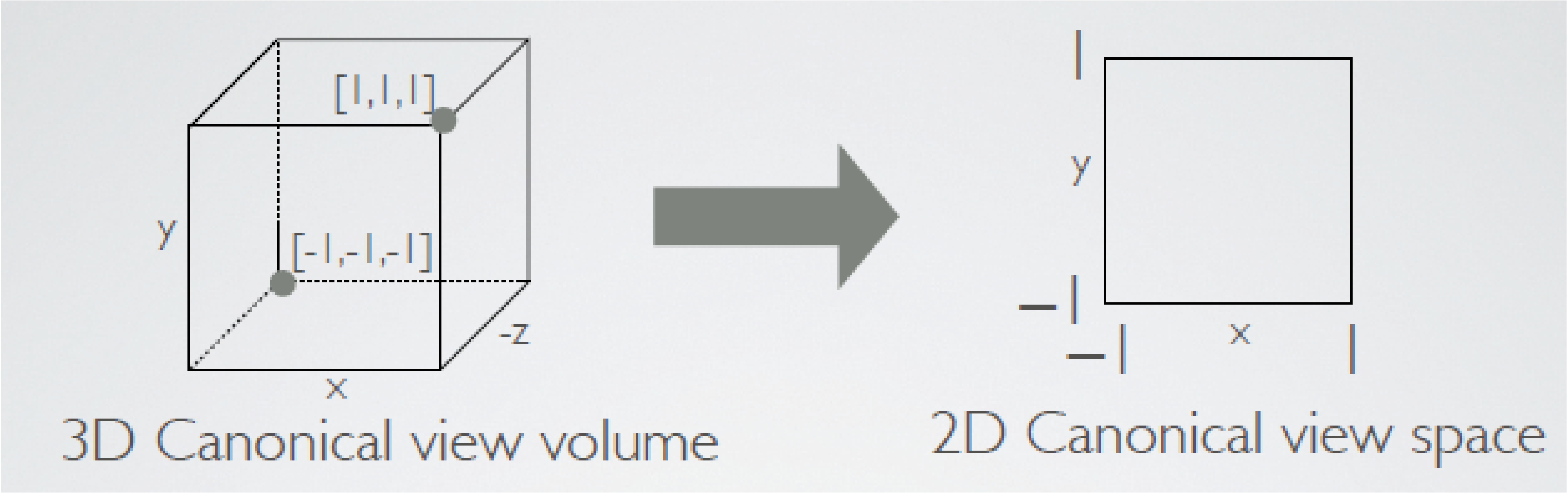


$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y - 1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ 1 \end{bmatrix}$$

here we dropped the z component

3D Viewport Transformation

Converts points from the 3D canonical space to the 2D canonical space



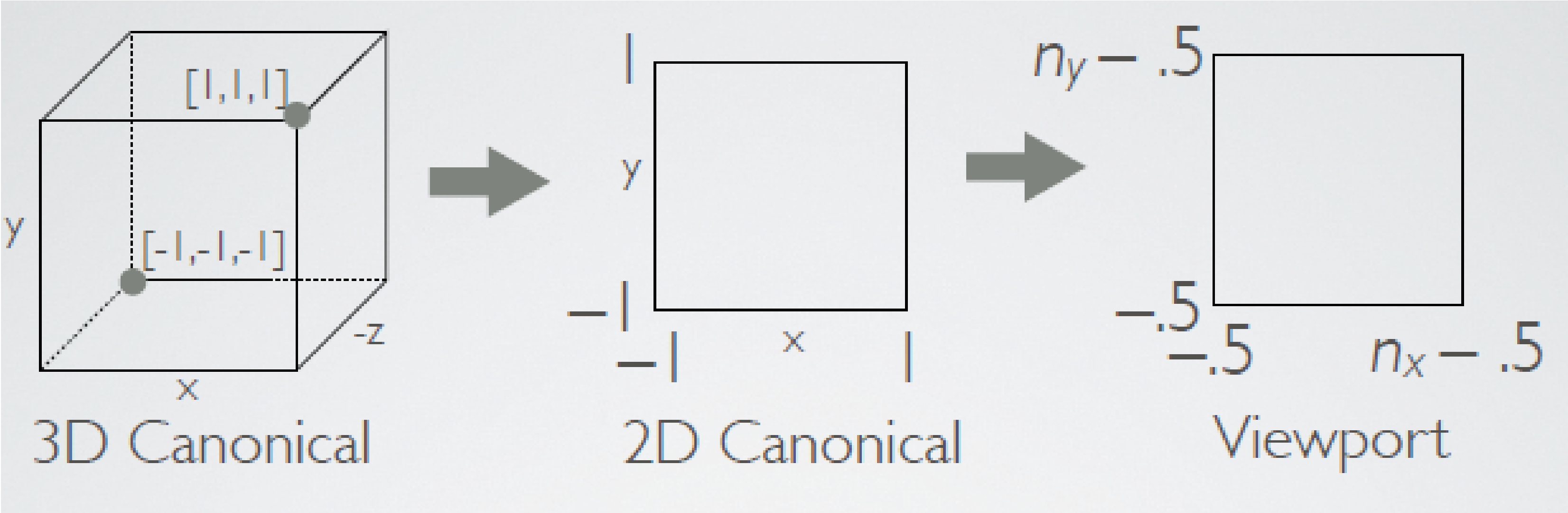
$$\begin{bmatrix} x_{canonical} \\ y_{canonical} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix}$$

here we keep the z component

Drop z-coordinate. Orthographic Projection!

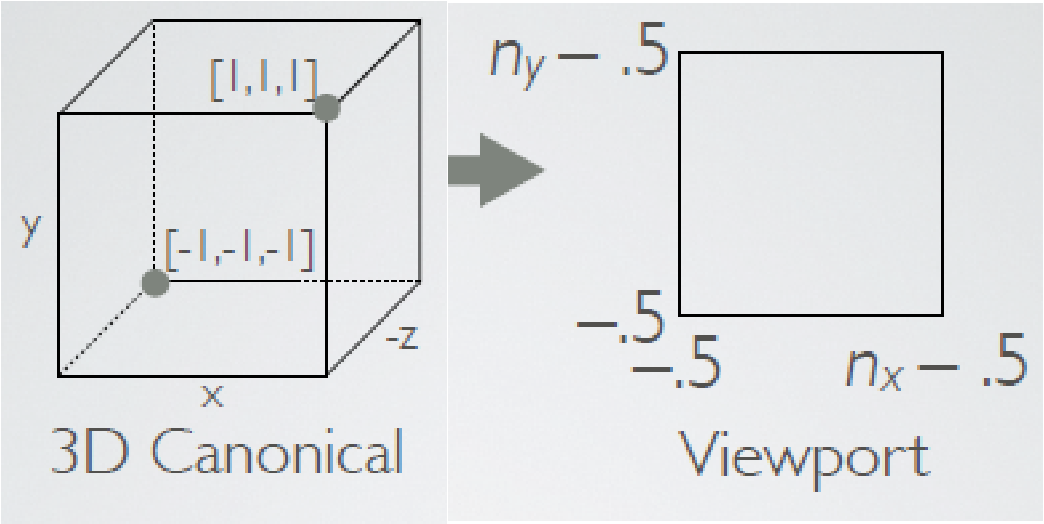
3D Viewport Transformation

Converts points from the 3D canonical space to the screen space



$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y - 1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix}$$

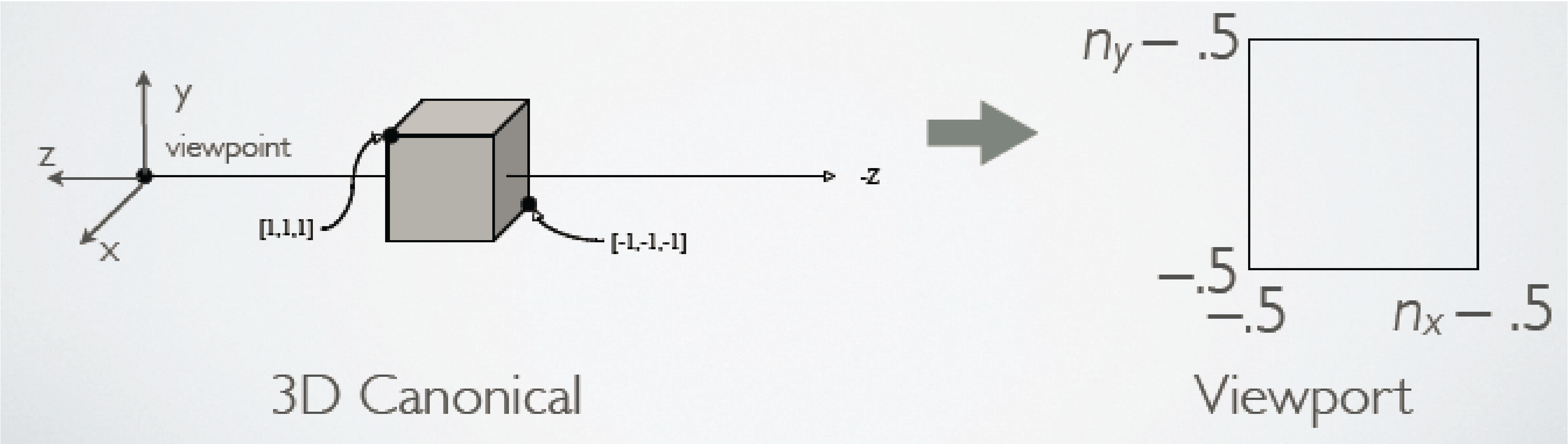
3D Viewport Transformation



$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z_{canonical} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y - 1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix}$$

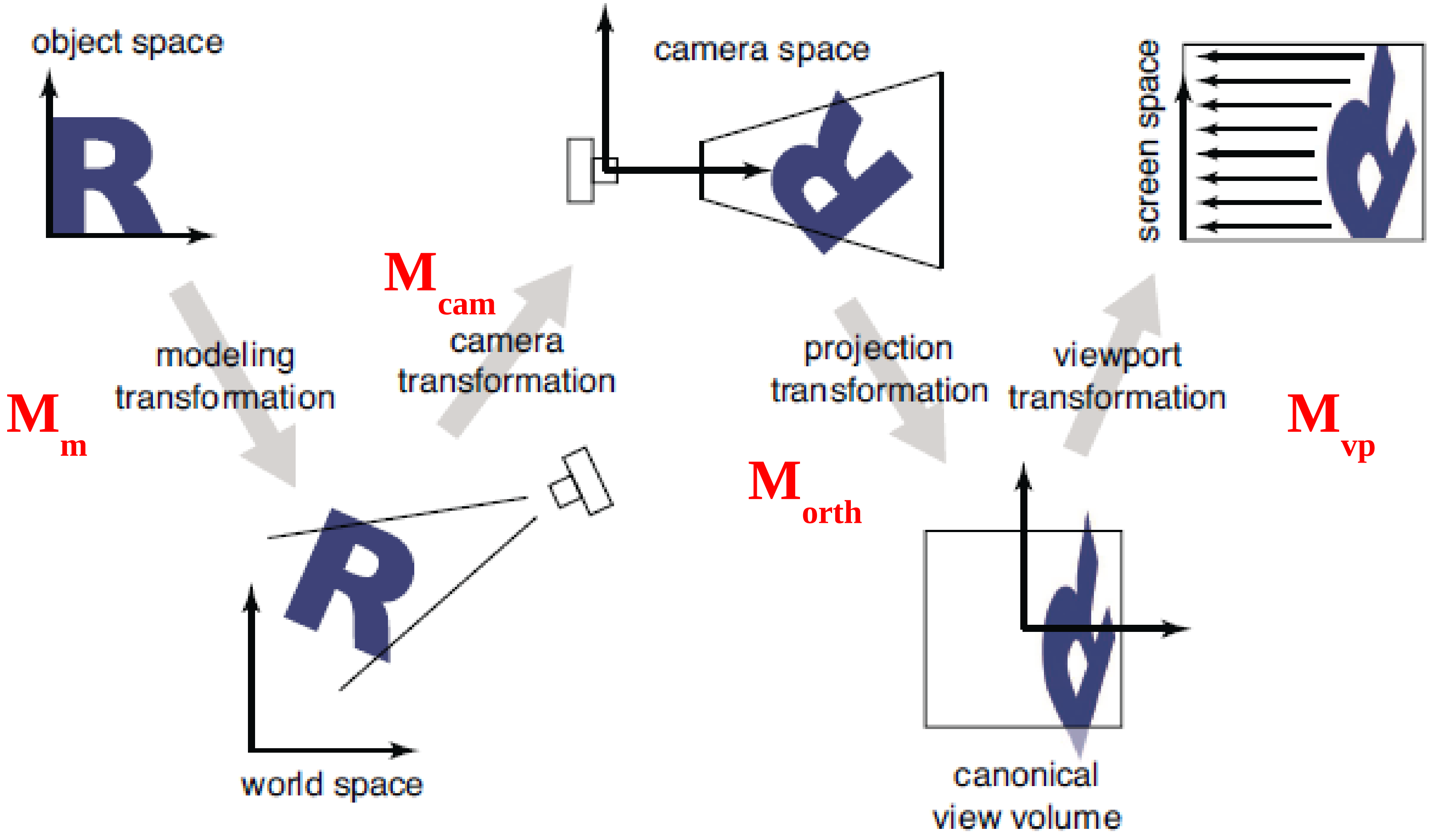
Why do we keep the *z* coordinate?

3D Viewport Transformation



$$M_{vp} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y - 1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic Transformation Pipeline



Orthographic Transformation

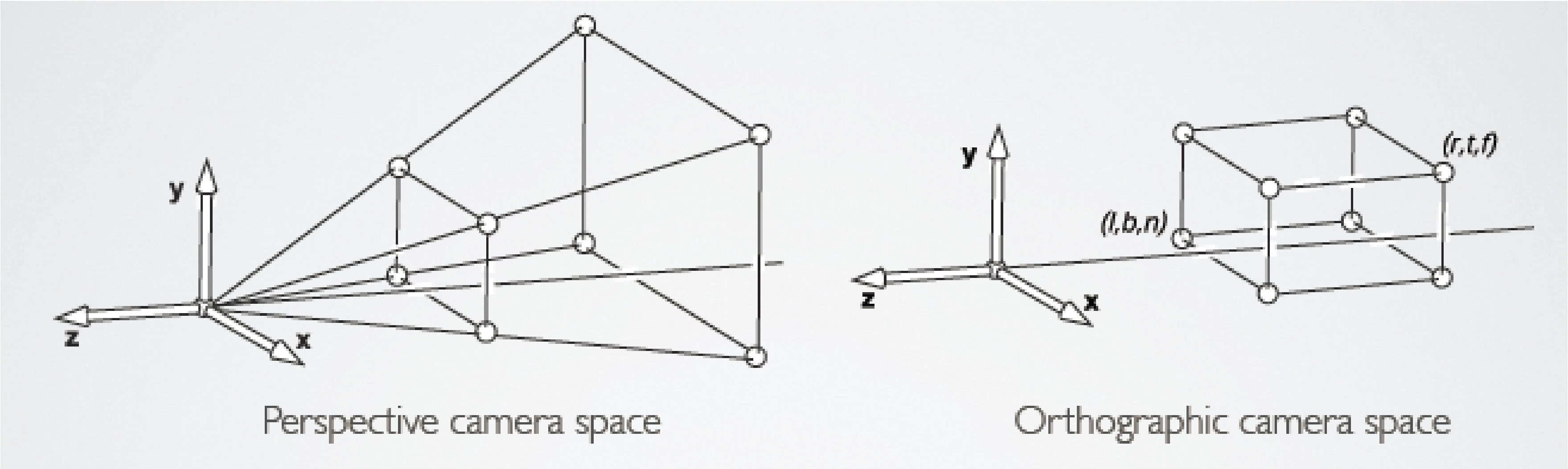
- Start with point in Object coordinates
- Convert to World Coordinates: M_m
- Convert to Camera Coordinates: M_{cam}
- Perform Orthographic Projection: M_{orth}
- Convert to Screen Coordinates: M_{vp}

$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z_{canonical} \\ 1 \end{bmatrix} = M_{vp} M_{orth} M_{cam} M_m \begin{bmatrix} x_{object} \\ y_{object} \\ z_{object} \\ 1 \end{bmatrix}$$

just indicator to know which is far and which is close

mid term l7d el slide de.

Perspective Projection



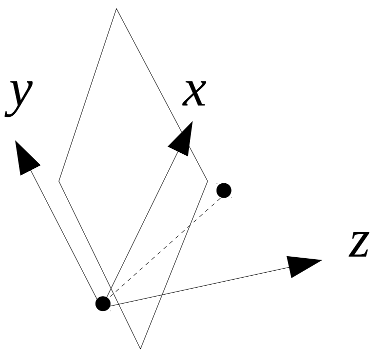
Perspective View Volume: Frustum   Orthographic View Volume

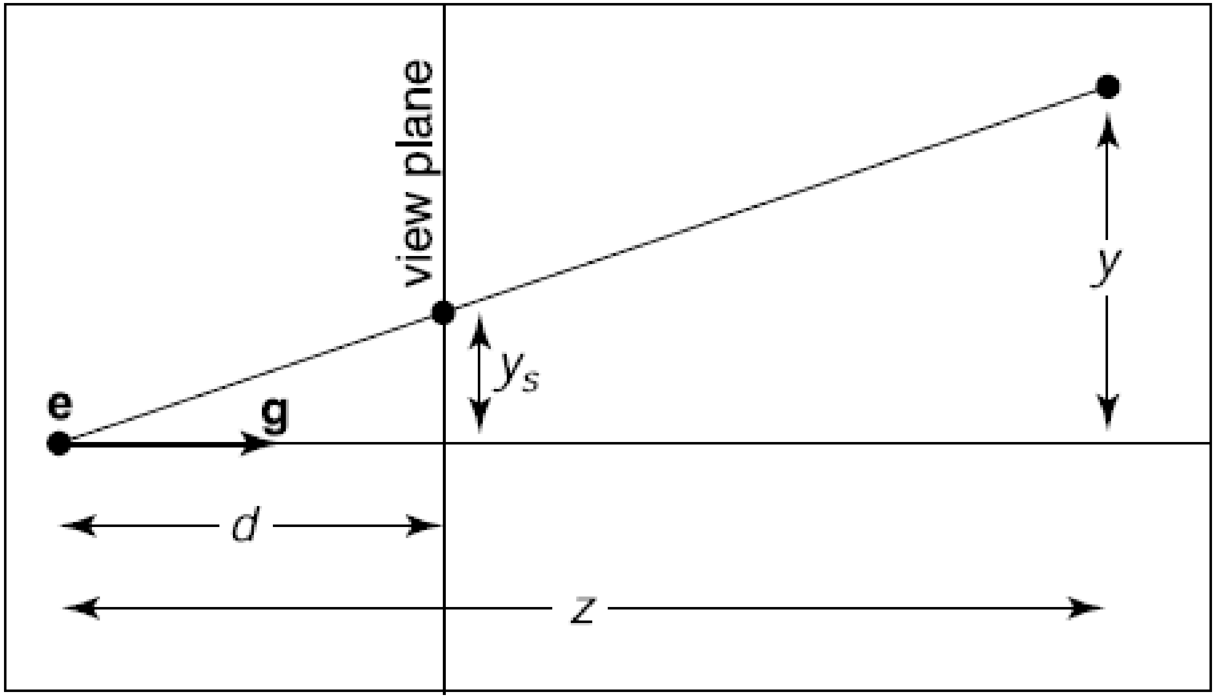
Projection lines go through the camera center !

Want to map the perspective view frustum onto the orthographic view volume

Perspective Projection

Consider first the y coordinate





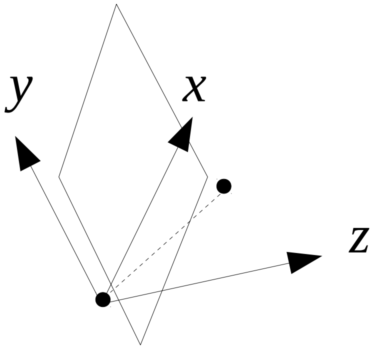
Similar Triangles

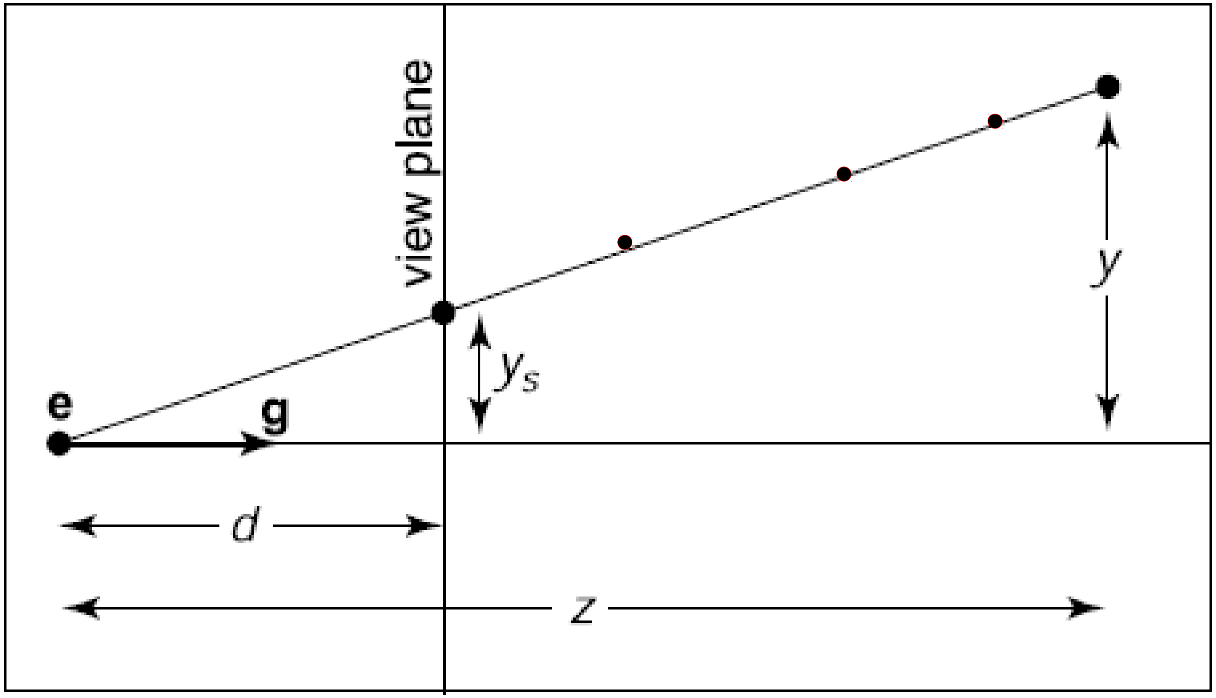
$$\frac{y_s}{d} = \frac{y}{z}$$

$$y_s = \frac{dy}{z}$$

Perspective Projection

Consider first the *y* coordinate



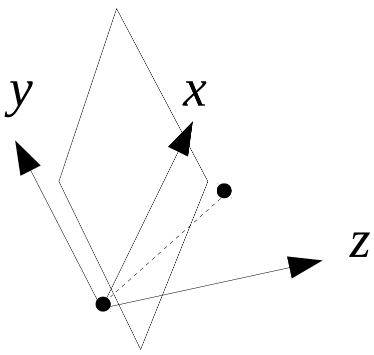


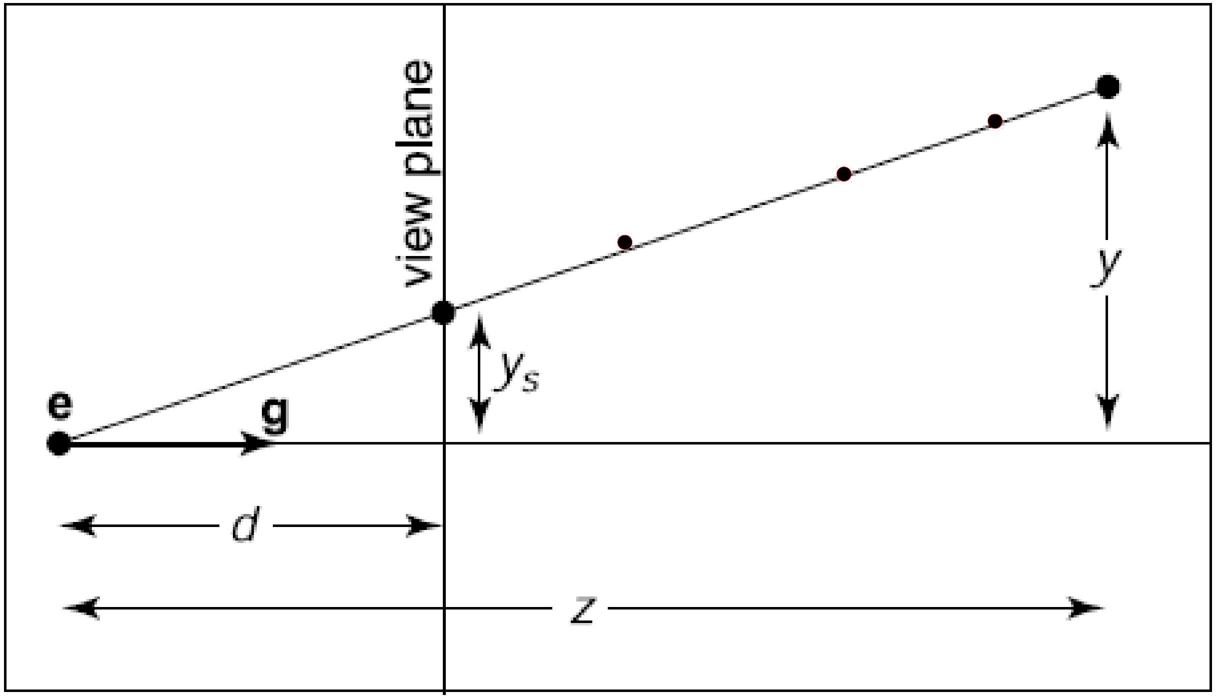
$$y_s = \frac{dy}{z}$$

Notice that all points on the line joining the point with *e* have the same project y_s e.g. *y*/2 and *z*/2

Perspective Projection

Consider first the y coordinate





$$y_s = \frac{dy}{z}$$

How do we perform this division using a transformation matrix?

Homogeneous Coordinates

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Represent 3D points as 4D vectors
such that scale does not matter

$$p \sim w\, p$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \sim \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

Allow any w

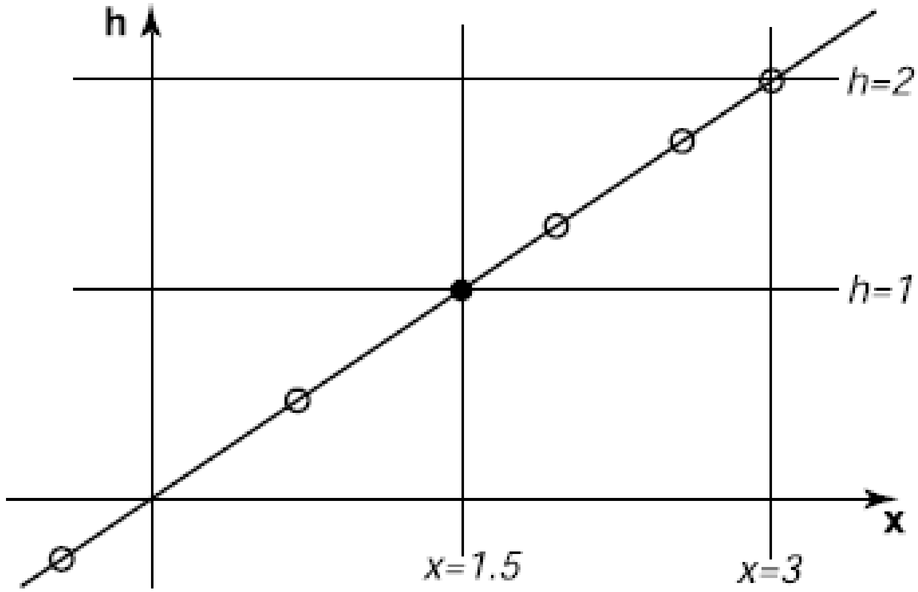
$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \sim \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

Divide by w to go back to 3D

What if w is zero? Then it's a *vector* not a *point*!

Homogeneous Coordinates

In 1D, we represent a point as a 2D vector $[x, h]$



The point $x = 1.5$ is equivalent to all points $[1.5\,h, h]$ for all h

For example $[3, 2]$ is the same as the point $[1.5, 1]$ which is $x = 1.5$

Perspective Projection

$$y_s = \frac{dy}{z} \;\&\; x_s = \frac{dx}{z}$$

So how do we divide by z?

By putting z in the homogeneous coordinate ...

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \begin{bmatrix} dx/z \\ dy/z \\ 1 \end{bmatrix} \sim \begin{bmatrix} dx \\ dy \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

What about z_s ?

Perspective Projection

What happens if we add the row (0, 0, 1, 0)?

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\tilde{z} = z \rightarrow z_s = 1$$

Z coordinate is lost ! How do we preserve it i.e. keep relative depth information?

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Solve for *a* and *b* such that the depth information is saved

Perspective Projection

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\tilde{z} = az + b \;\; \& \;\; z_s = \frac{az+b}{z}$$

Set $d = n$ and find a & b such that:

- when $z = n$ we get $z_s = n$
- when $z = f$ we get $z_s = f$

$$a = n + f \;\; \& \;\; b = -fn$$

Perspective Projection

The perspective projection matrix becomes:

$$P = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

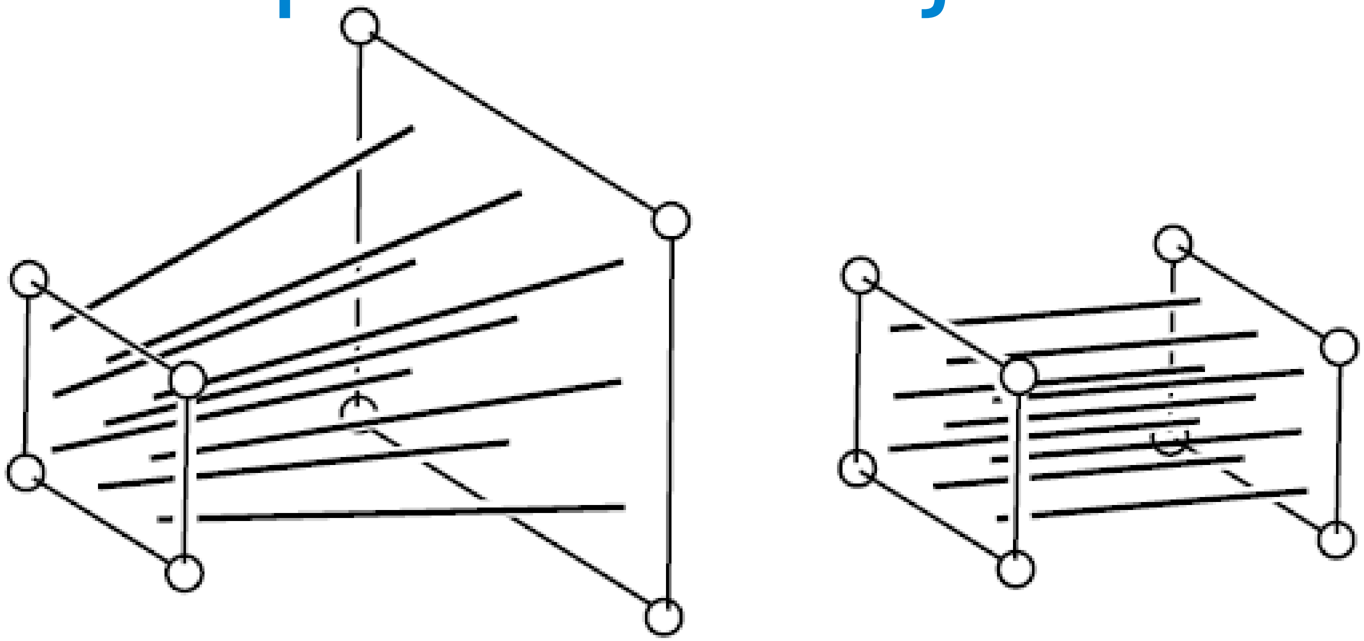
Notice that points with $z = n$ don't change.

$$x_s = \frac{nx}{n} = x$$

$$y_s = \frac{ny}{n} = y$$

$$z_s = \frac{(n+f)n - fn}{n} = n$$

Perspective Projection

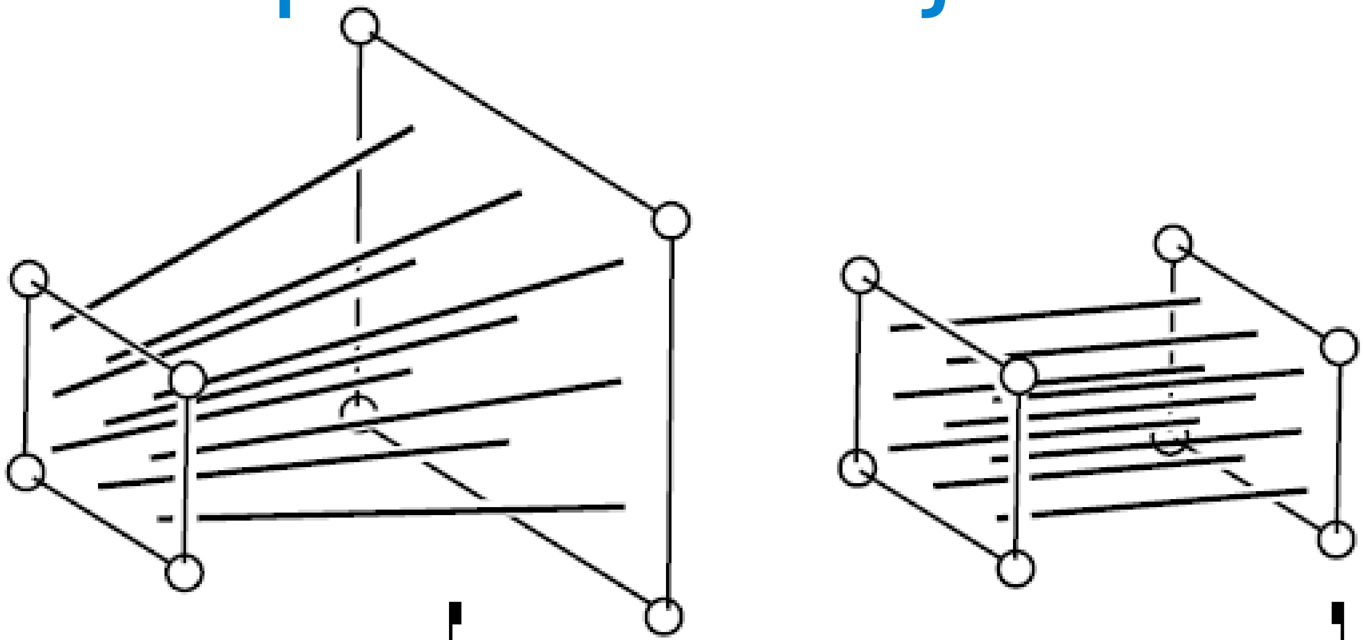


$$P = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Maps lines through the origin to lines parallel to z-axis preserving the point at z=n

Now that we have transformed the view frustum into the orthographic view volume, we can perform the rest of the pipeline starting at the orthographic projection

Perspective Projection



$$M_{per} = M_{orth} P = \begin{bmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{-2nf}{n-f} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The complete Perspective Projection matrix (including the orthographic transformation)

Perspective Transformation

- Start with point in Object coordinates
- Convert to World Coordinates: M_m
- Convert to Camera Coordinates: M_{cam}
- Perform Perspective Projection: P
- Perform Orthographic Projection: M_{orth}
- Convert to Screen Coordinates: M_{vp}

$$\begin{bmatrix} x_s \\ y_s \\ z_c \\ 1 \end{bmatrix} = M_{vp} M_{orth} P M_{cam} M_m \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$

Drawing Lines

```
Compute M=M_vp M_orth P M_cam M_m
For each line segment (a, b)
  p = Ma
  q = Mb
  drawline(xp/hp, yp/hp, xq/hq, yq/hq)
```

Compute $M = M_{vp} M_{orth} P M_{cam} M_m$

Recap

- Viewing
- Projections
  - Orthographic
  - Perspective
- Transformations Pipeline