# CMP205: Computer Graphics


## Lecture 4: Viewing and Projection

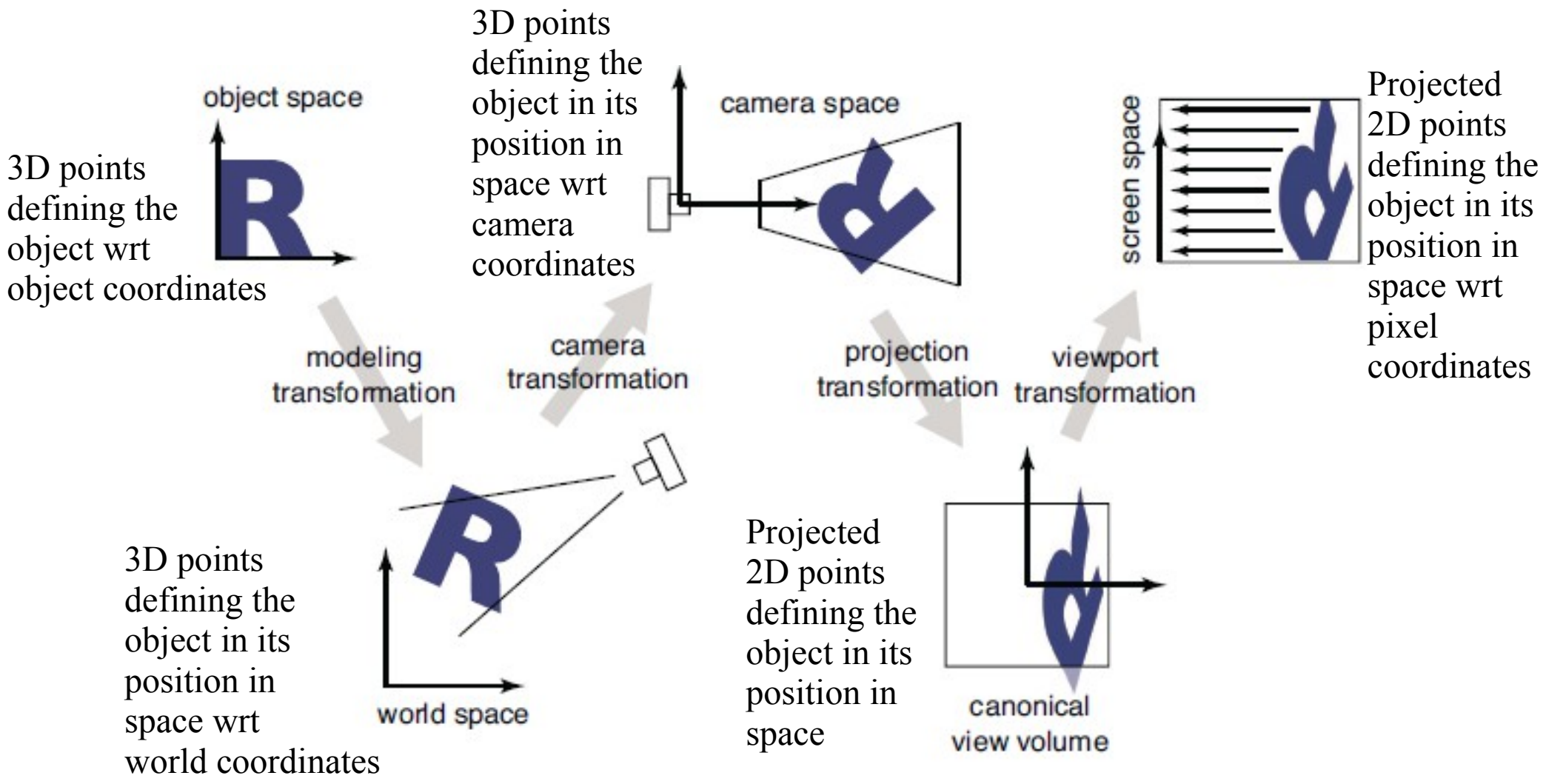
Mohamed Alaa El-Dien Aly
Computer Engineering Department
Cairo University
Fall 2013

# Agenda

- Viewing
- Projections
  - Orthographic
  - Perspective
- Transformations Pipeline

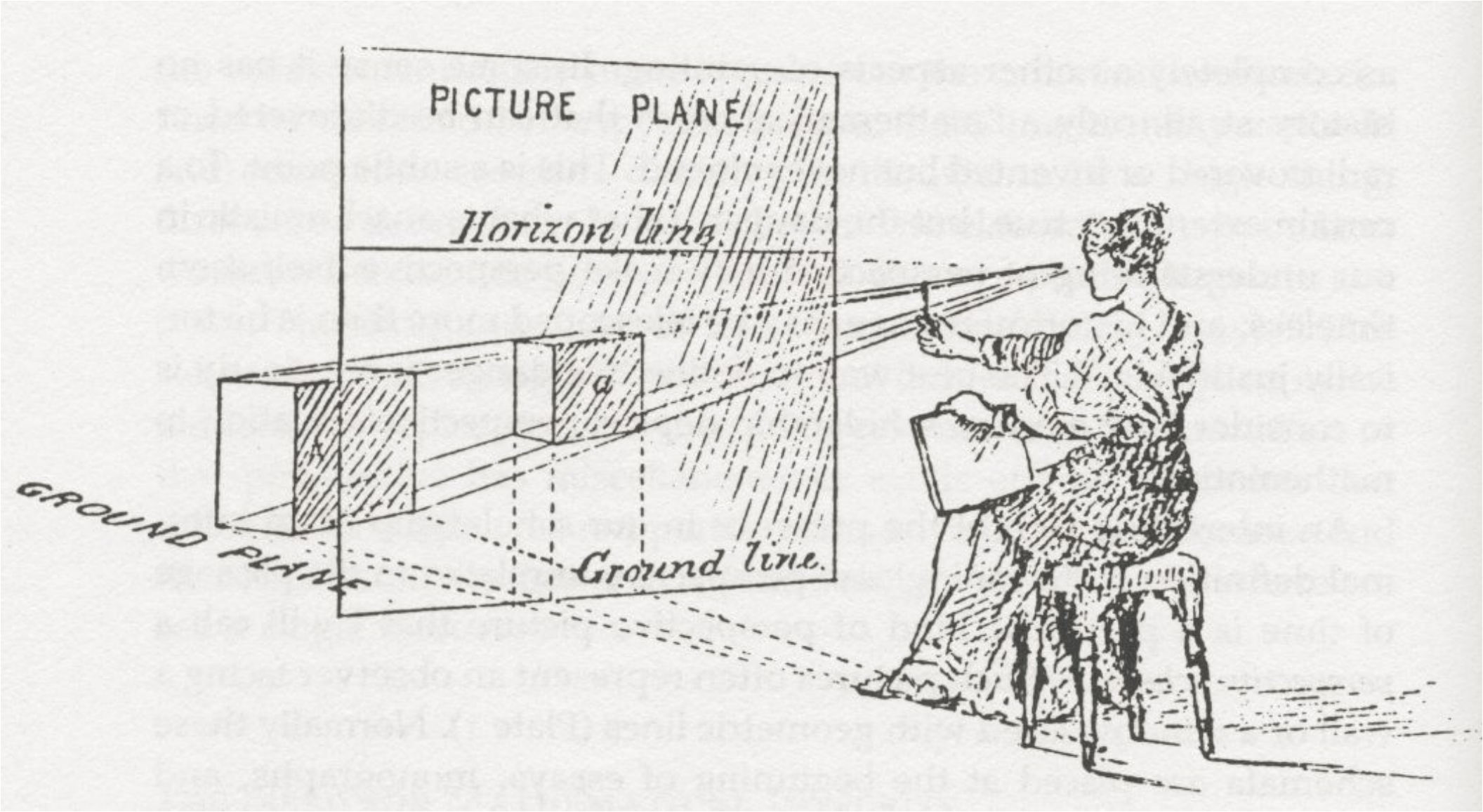
**Acknowledgment**: Some slides adapted from Steve Marschner and Maneesh Agrawala

# 3D Viewing



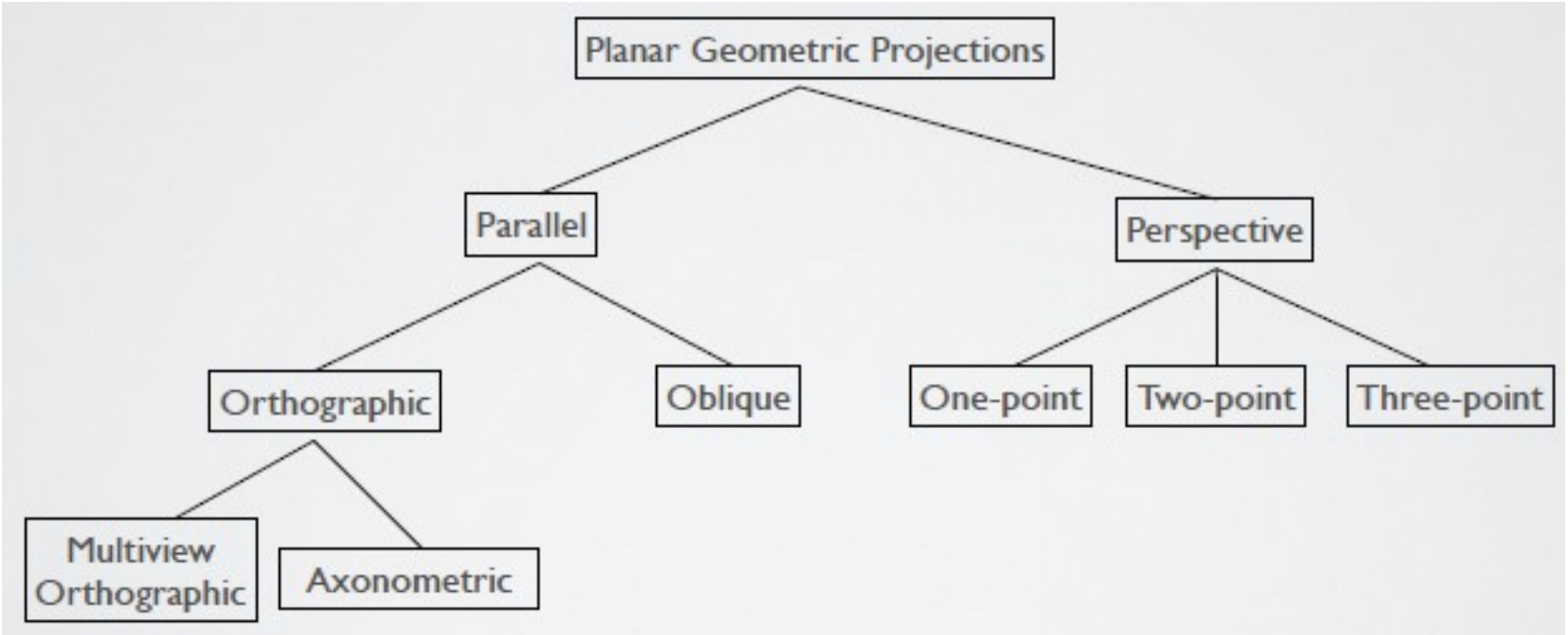

Convert from 3D points in space to 2D points on screen

# Projections



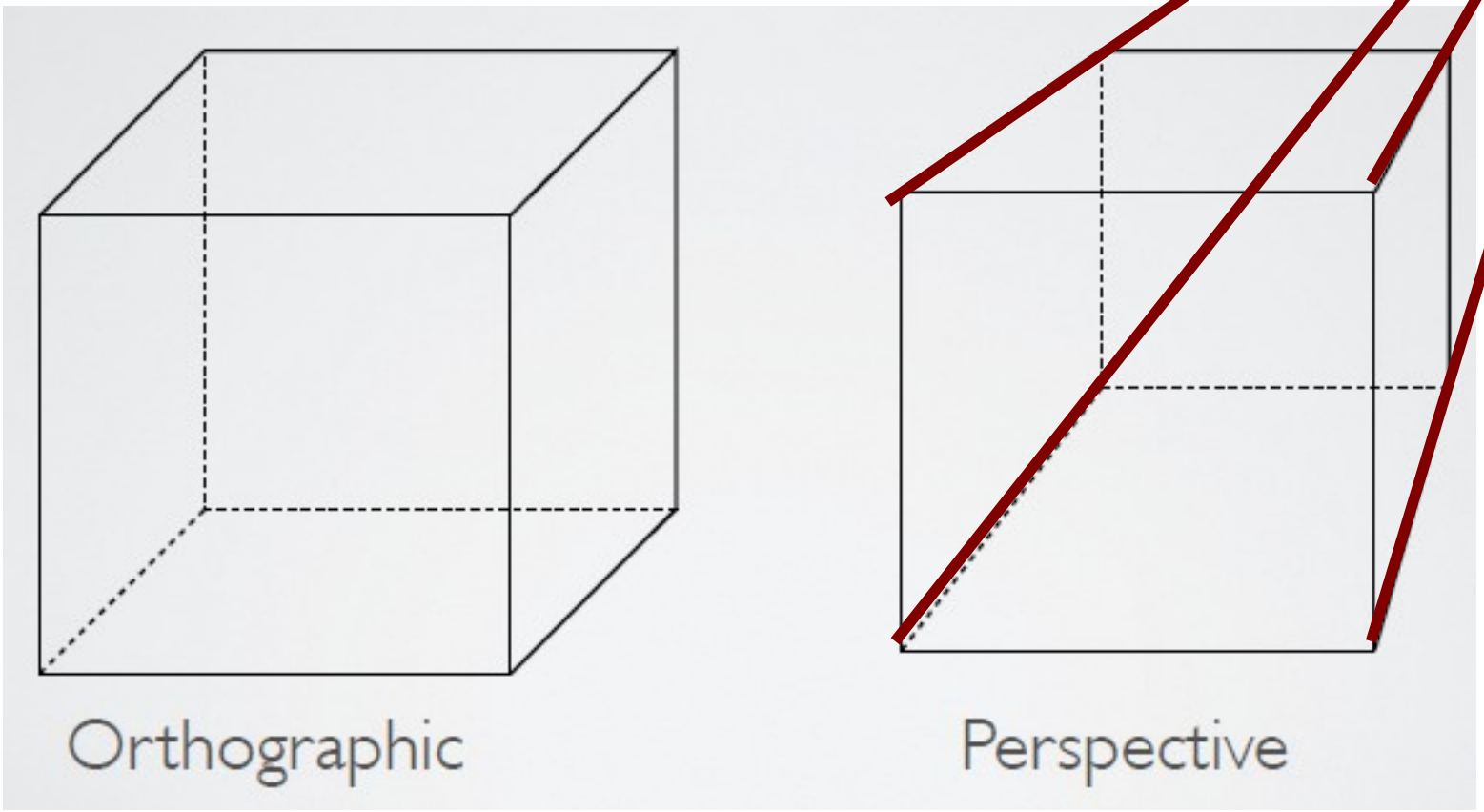

Project points in the world onto a plane

# Projections




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

# Projections




Parallel lines remain parallel

Parallel lines intersect at a *vanishing* point

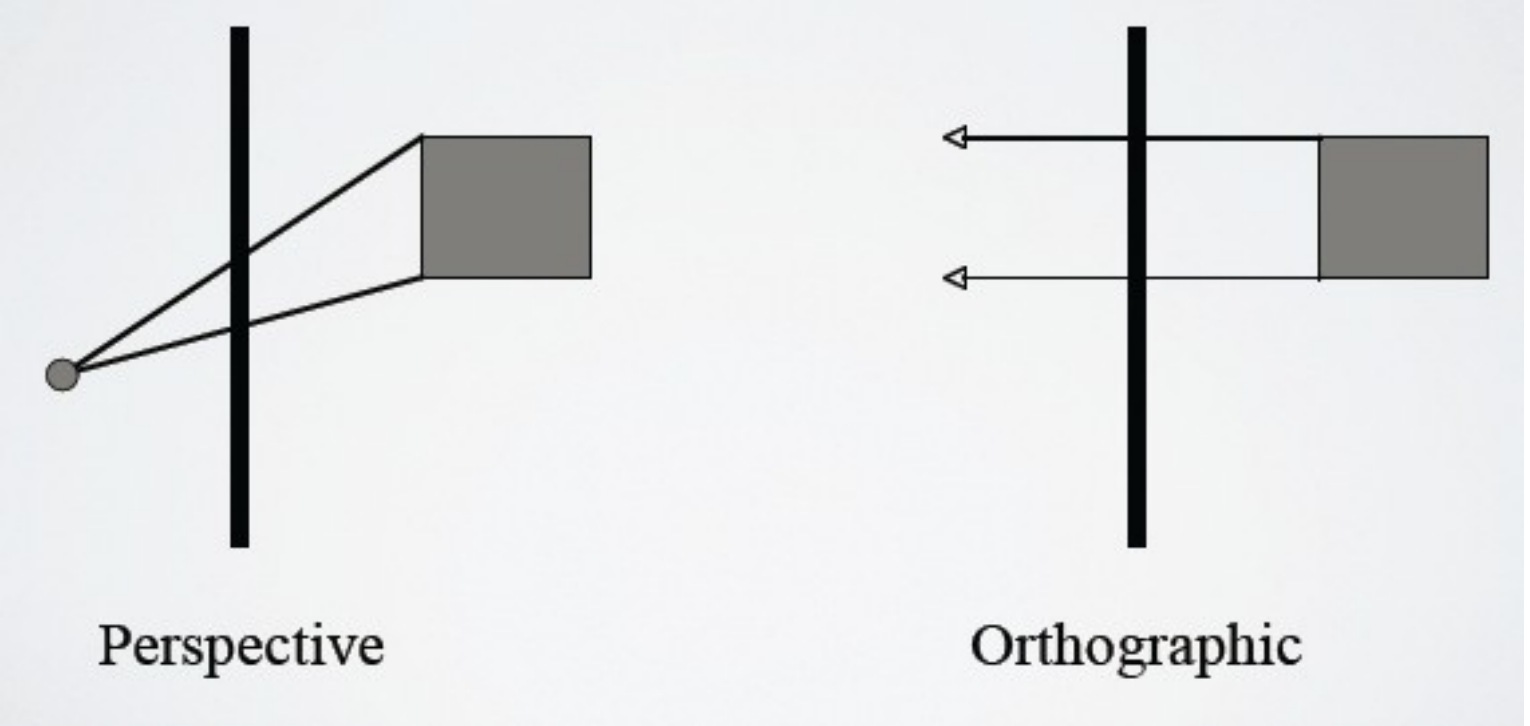
# Perspective Projection

Railway tracks intersect at a vanishing point

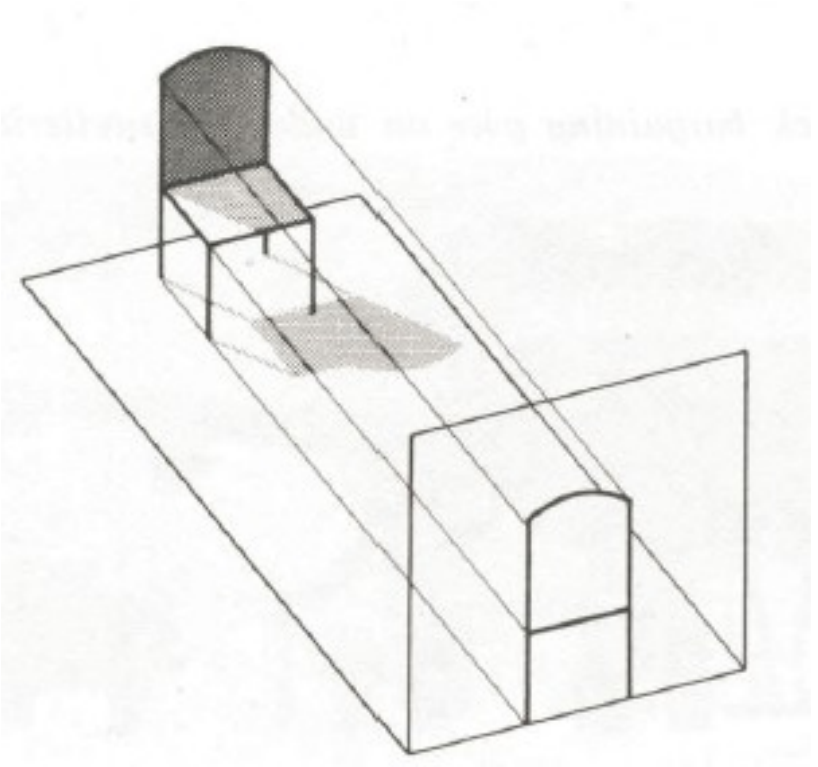
[www.edupic.net/math_pics.htm]

# Projections in 2D

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

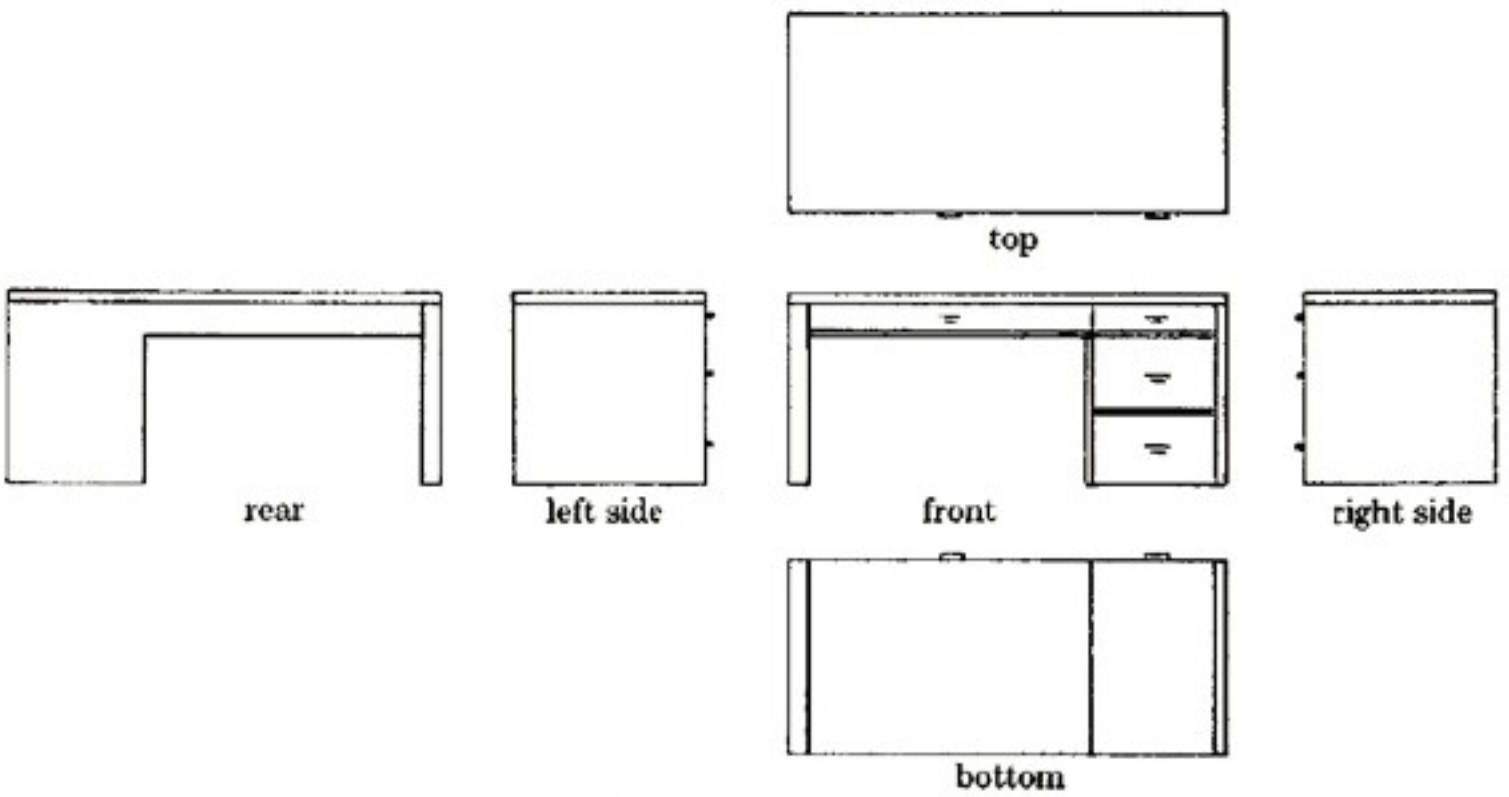


# Orthographic Projection



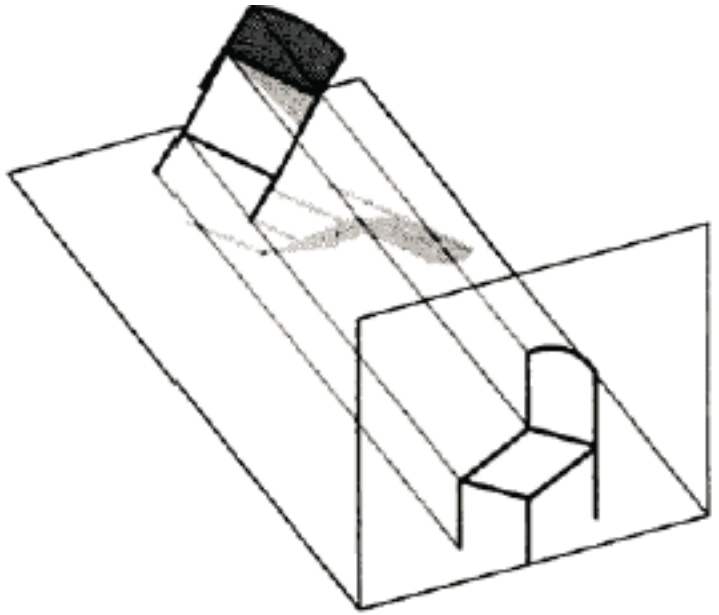
- Projection plane parallel to a Coordinate plane
- Projection direction perpendicular to projection plane

# Multiview Orthographic



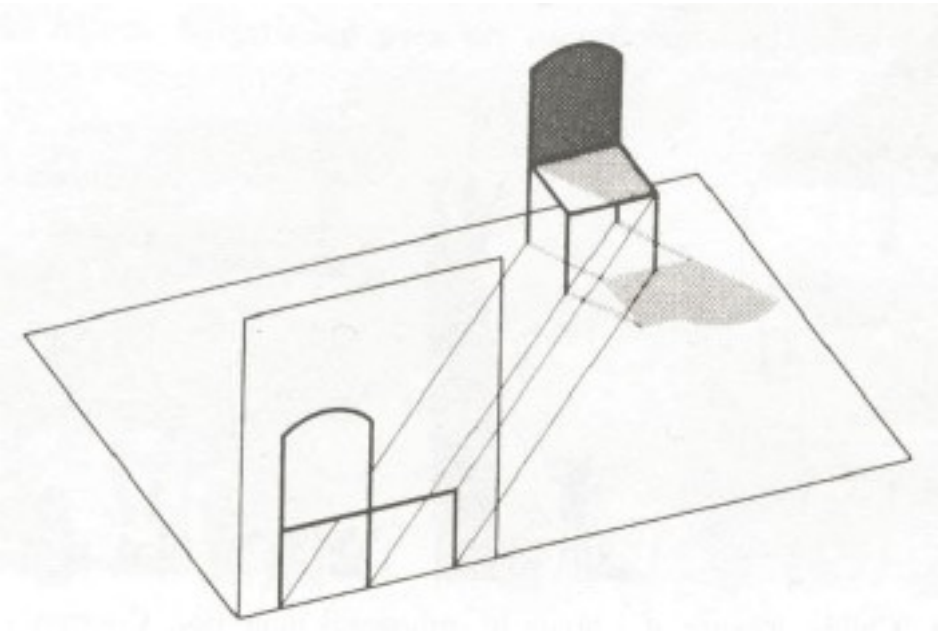

Similar to the engineering drawing of the preparatory year!

# Off-Axis Projections



Axonometric Projection

Projection plane not parallel to coordinate planes



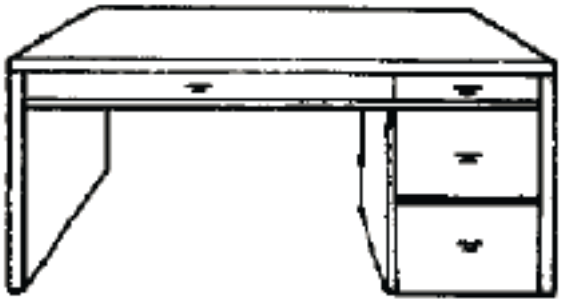
Oblique Projection

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

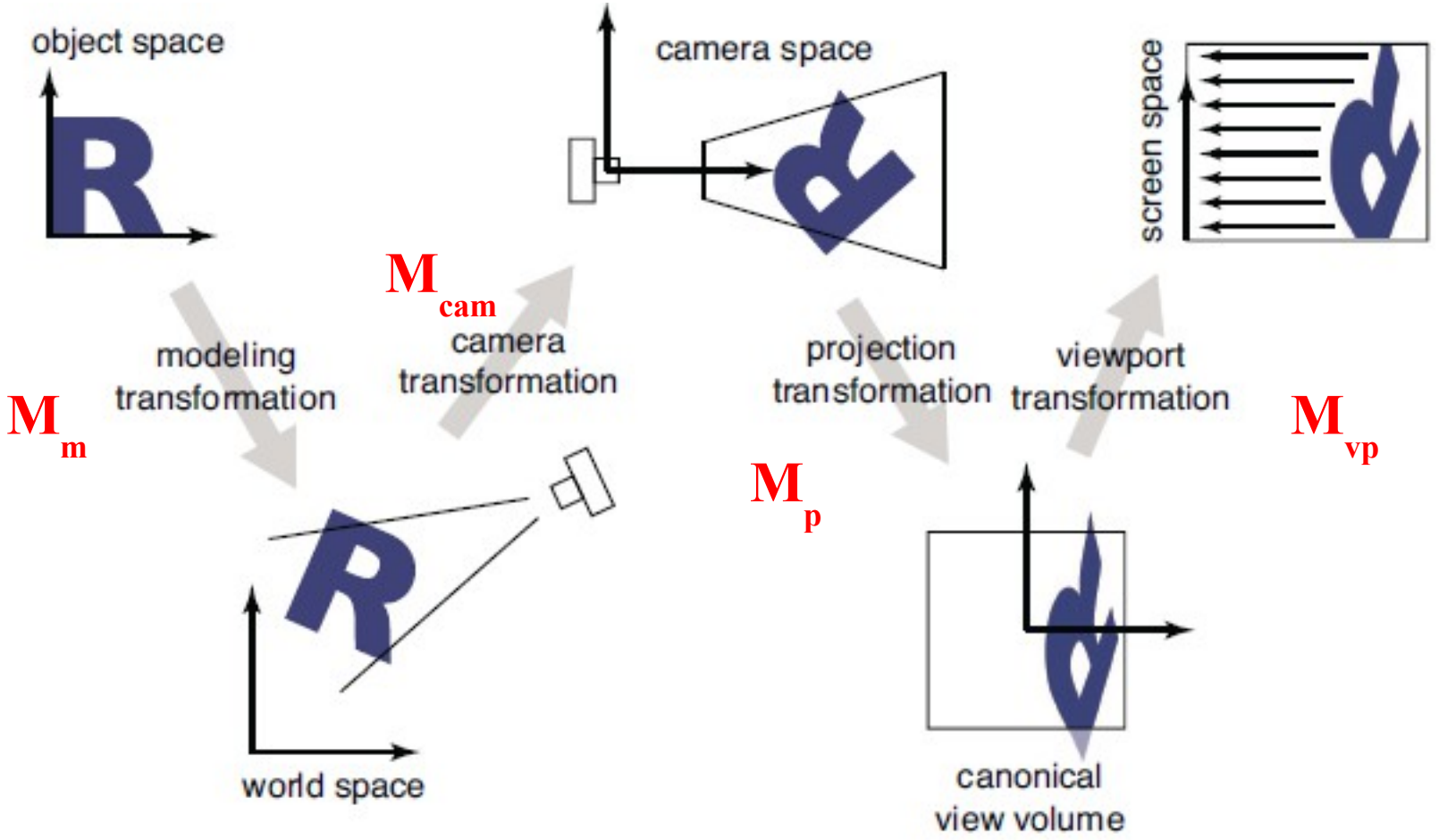
two-point

## Three-Point Perspective

Projection plane not parallel to
any coordinate axes

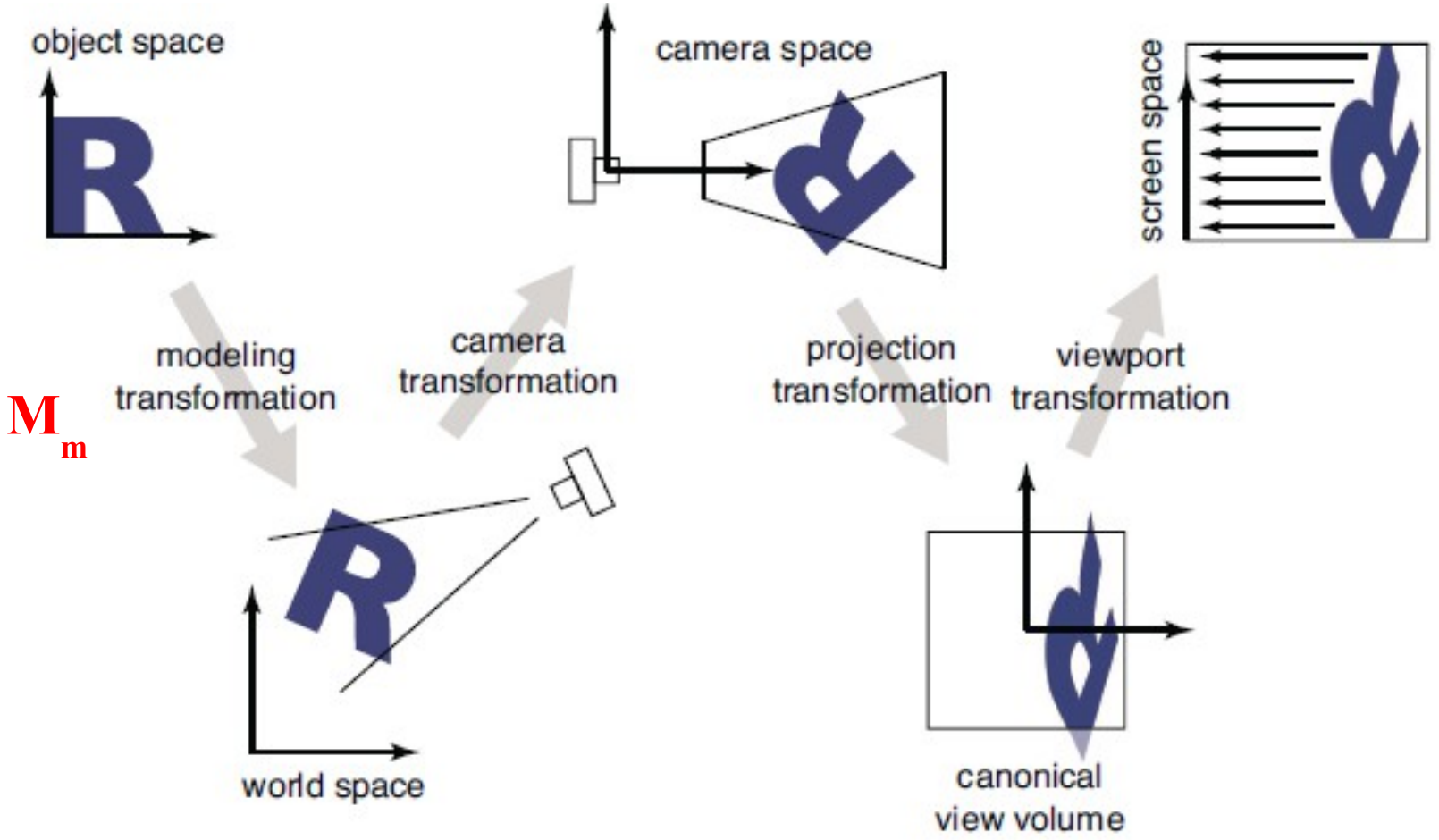
three-point

# Transformations Pipeline



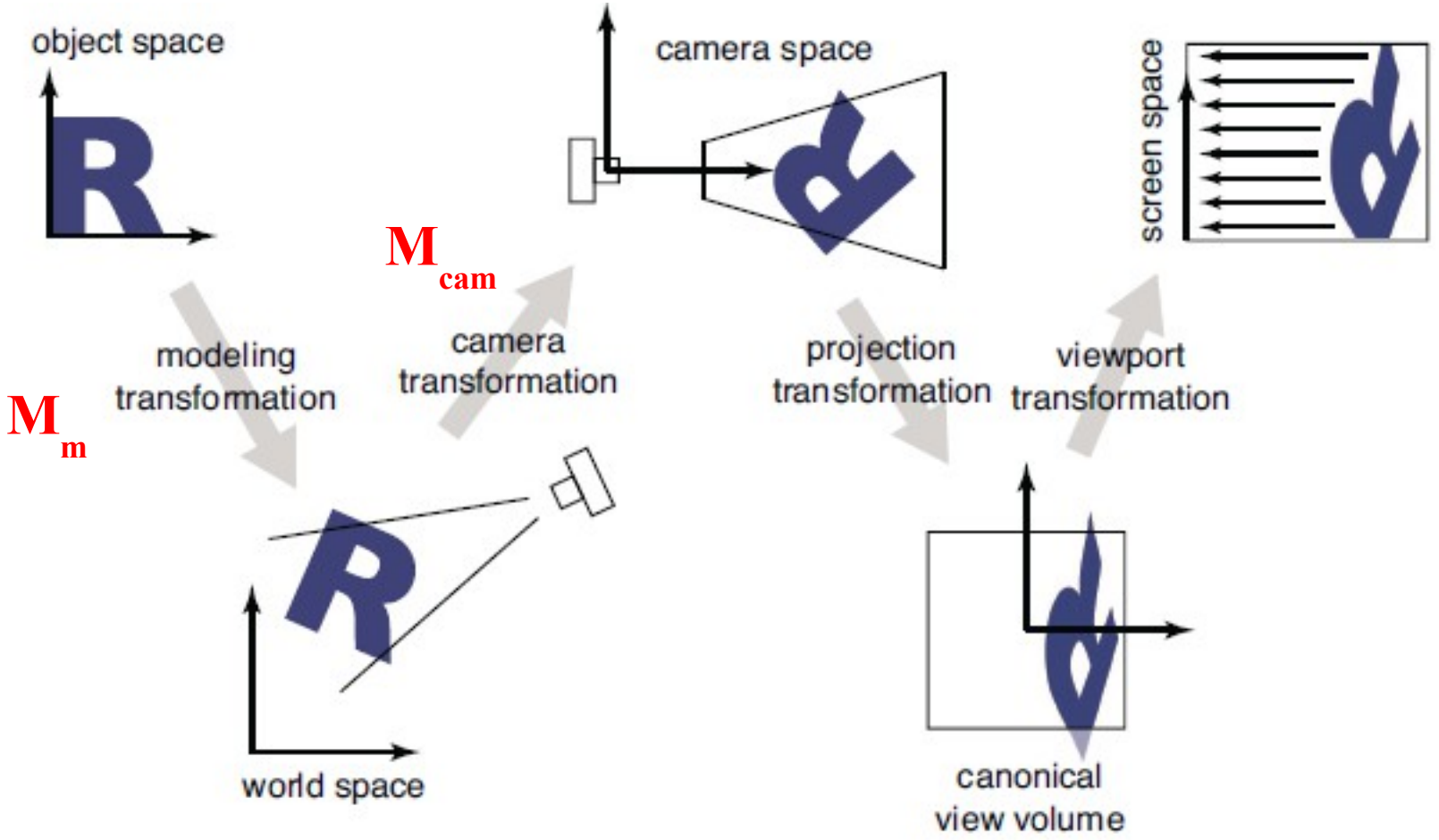

Converts 3D points in object space to 2D pixels on the screen through a series of transformations

# 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

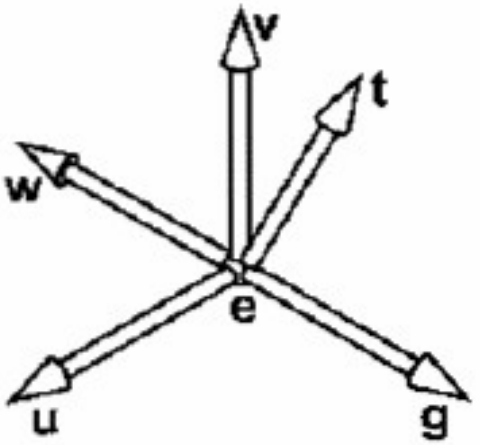
# Arbitrary Views

Camera frame is usually defined by:

$e$ : eye position

$g$ : gaze direction

$t$ : view up vector



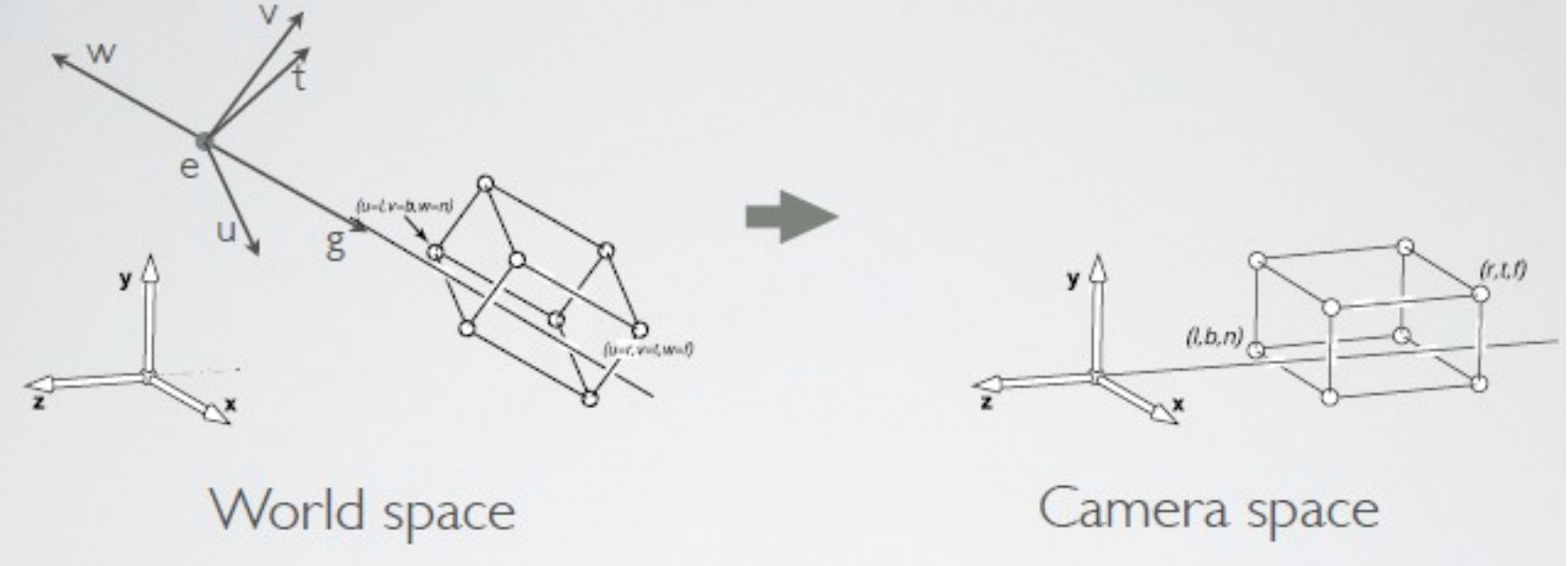

Using this information, we can construct three coordinate axes centered at $e$ as follows:

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

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

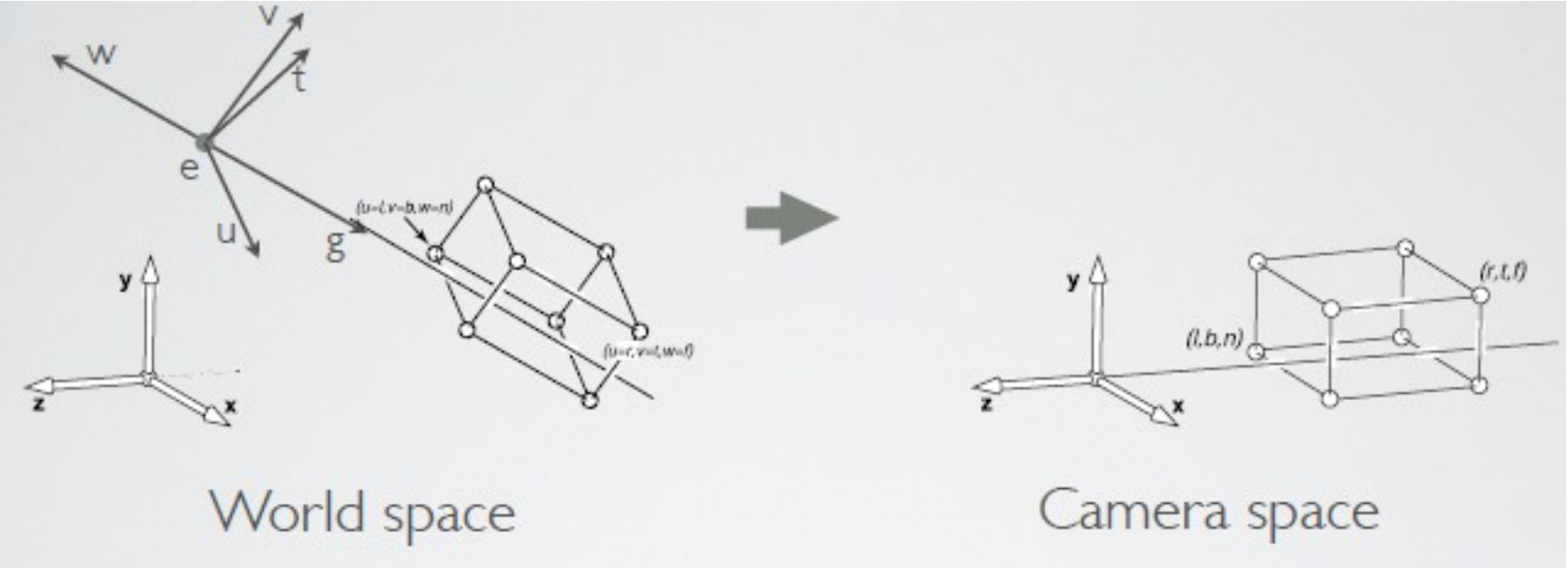
$$v = w \times u$$

# Camera Transformation




Convert from World Coordinates to Camera Coordinates

# Camera Transformation

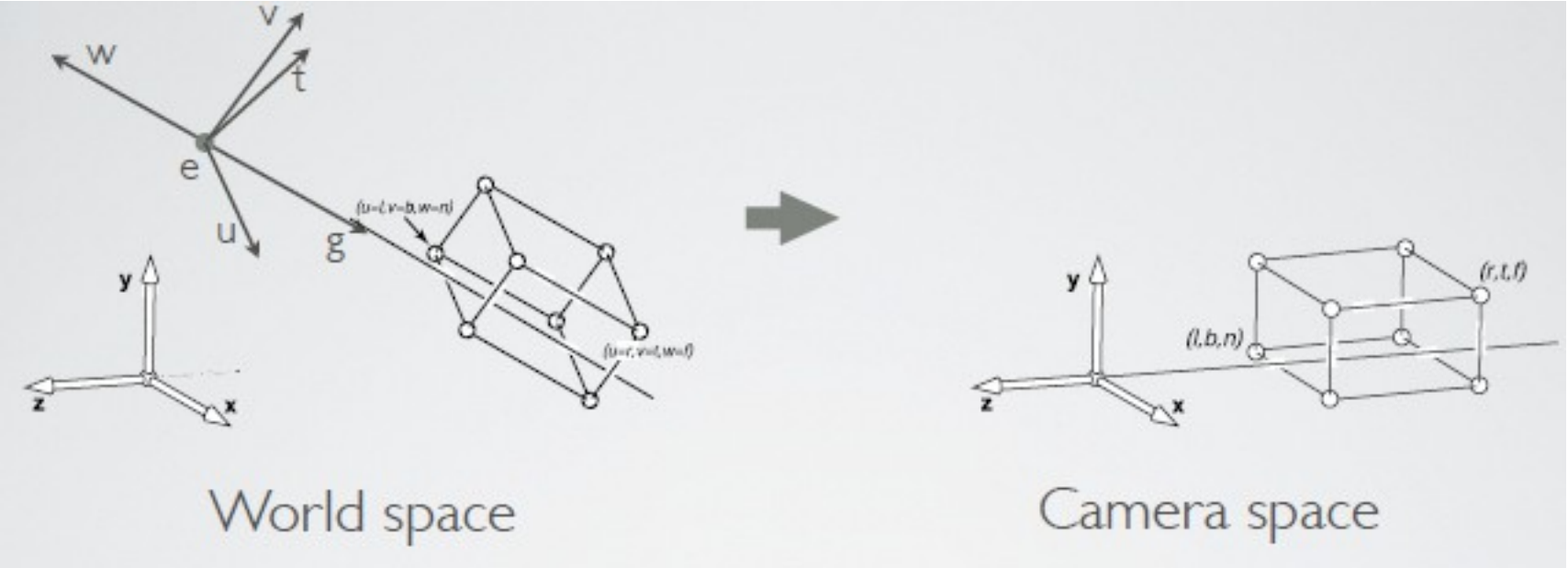



$$\boldsymbol{M}_{cam} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u & v & w & e \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

Aligns Camera Coordinates with World Coordinates

Moves Camera to World Origin

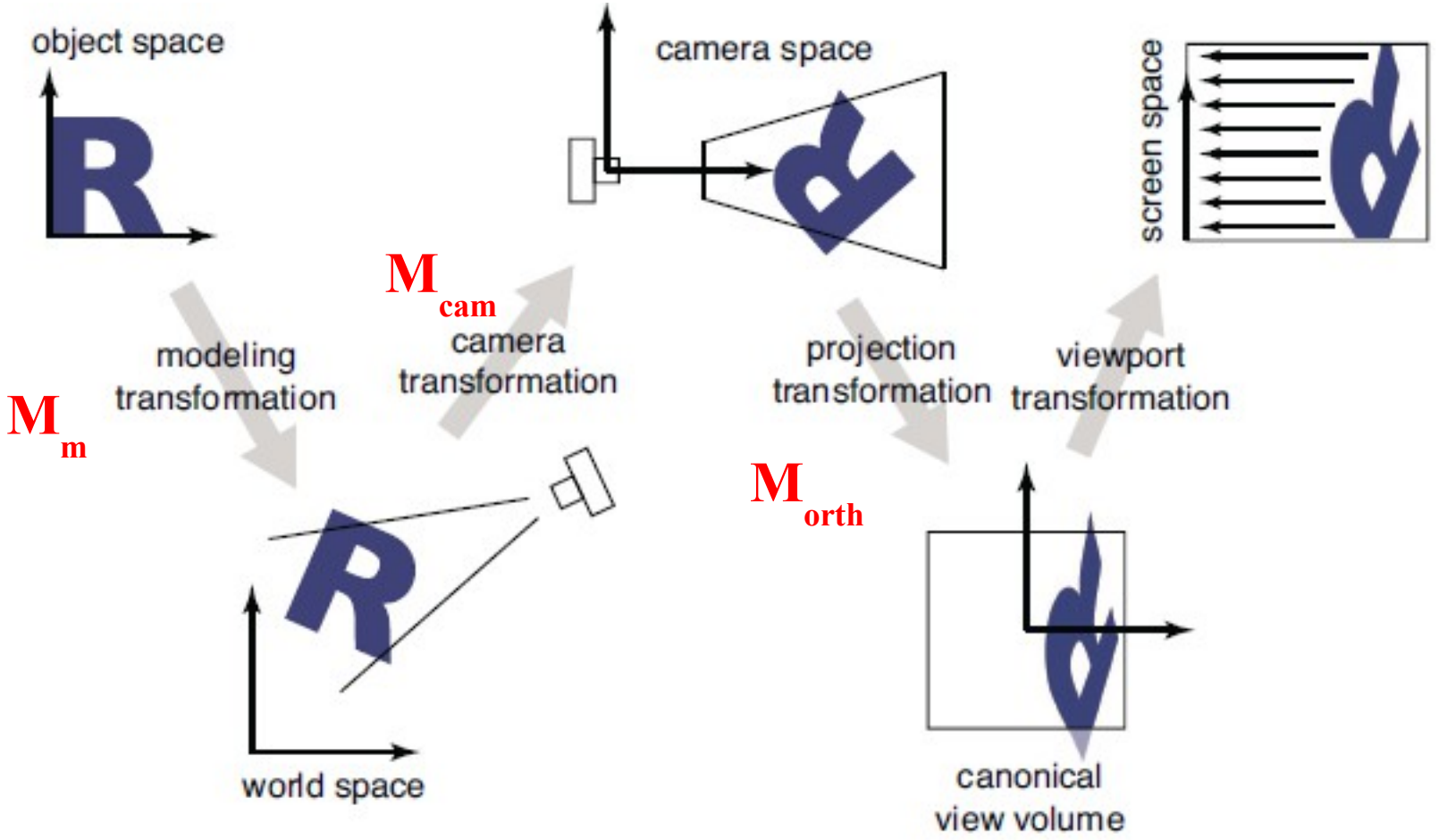
# Camera Transformation




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

Converts coordinates from the world frame to the camera frame

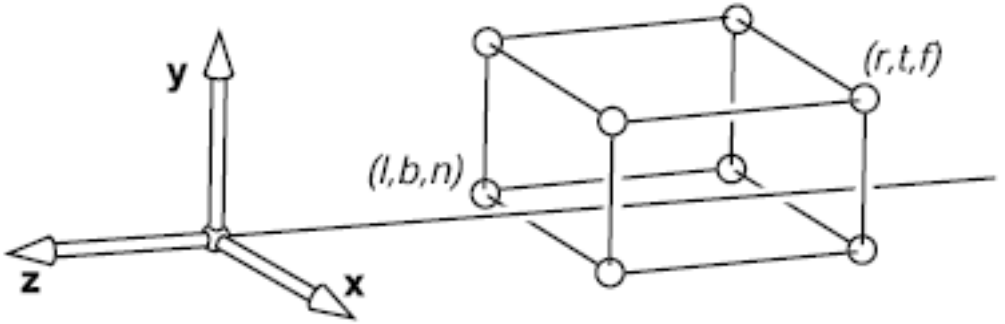
# Projection Transformation




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

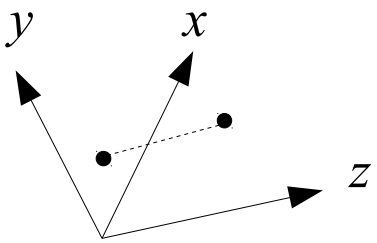
# Orthographic Projection

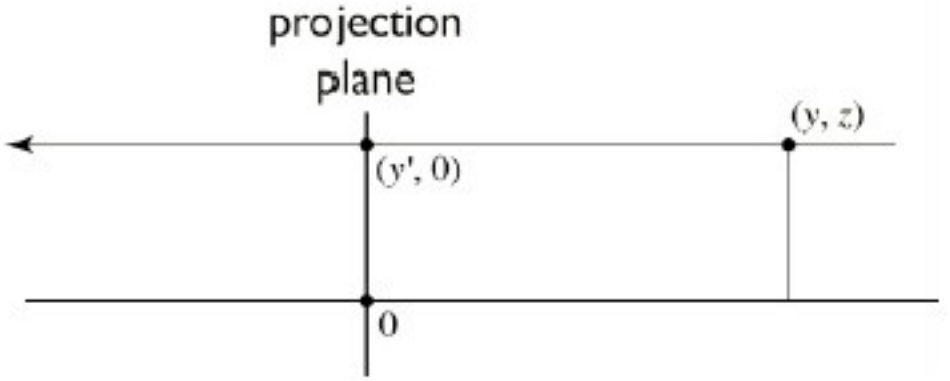
- Consider a camera at the *origin* looking at the –ve *z* direction
- We want to project the image on the *near* plane
- We want to project only the points defined in the "view volume" defined by (*left*, *bottom*, *near*) and (*right*, *top*, *far*)



# Orthographic Projection

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




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

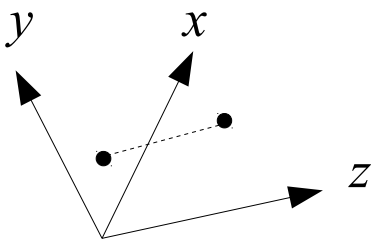
$$y' = y$$

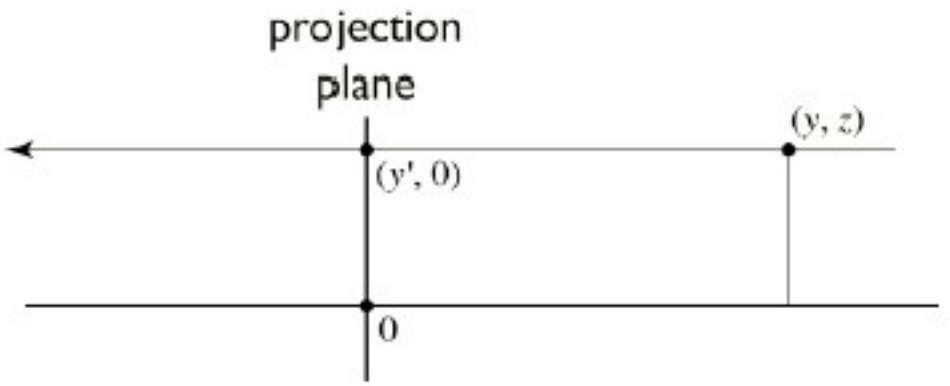
Similarly,

$$x' = x$$

Drop z-coordinate!

# Orthographic Projection



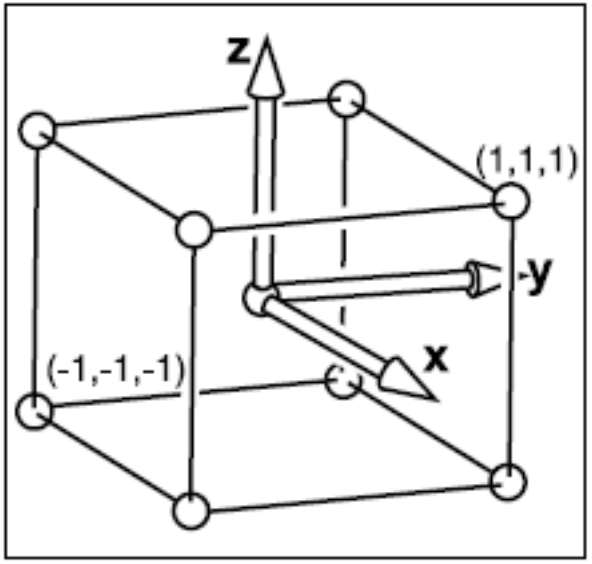

How do we get $(x',y')$ given $(x, y, z)$?

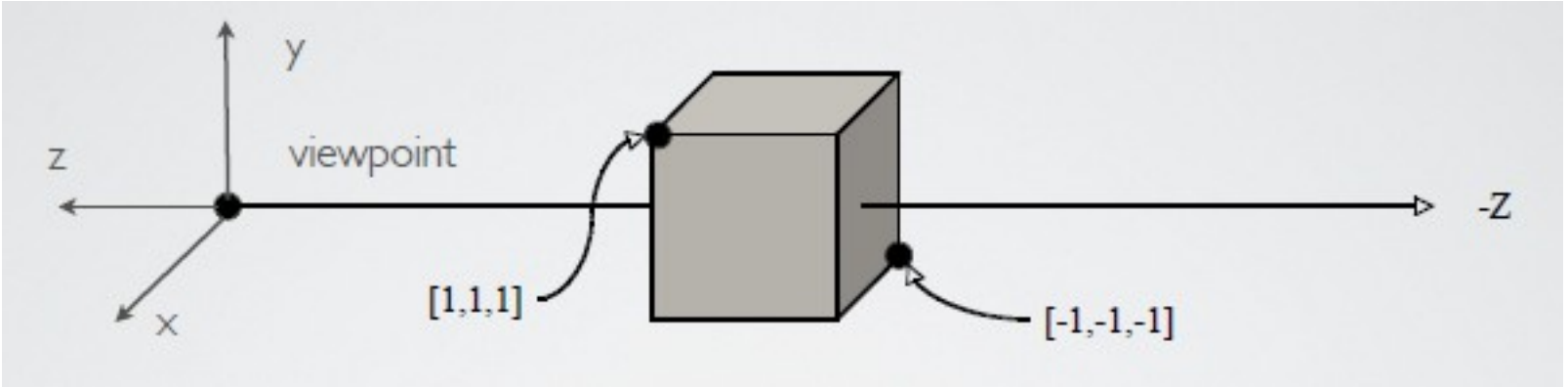
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Drop z-coordinate

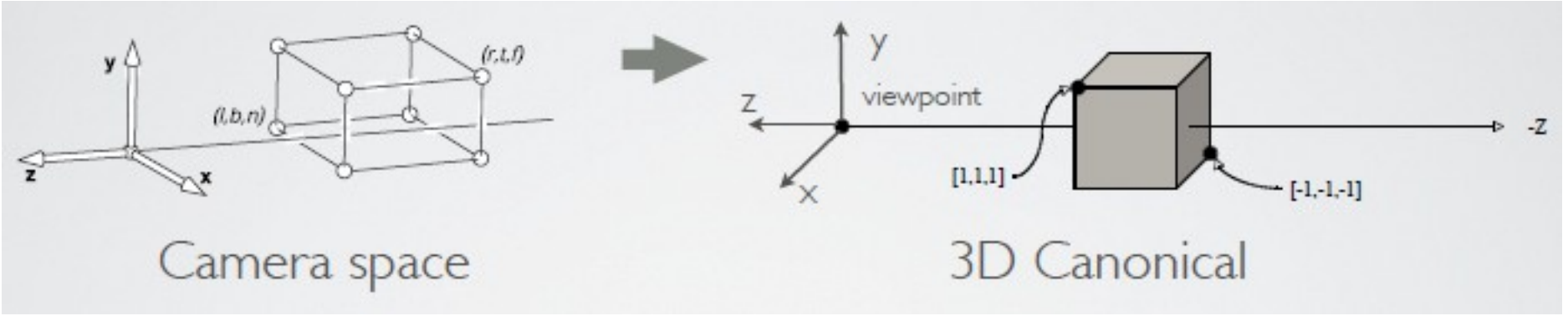
# 3D Canonical View Volume

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



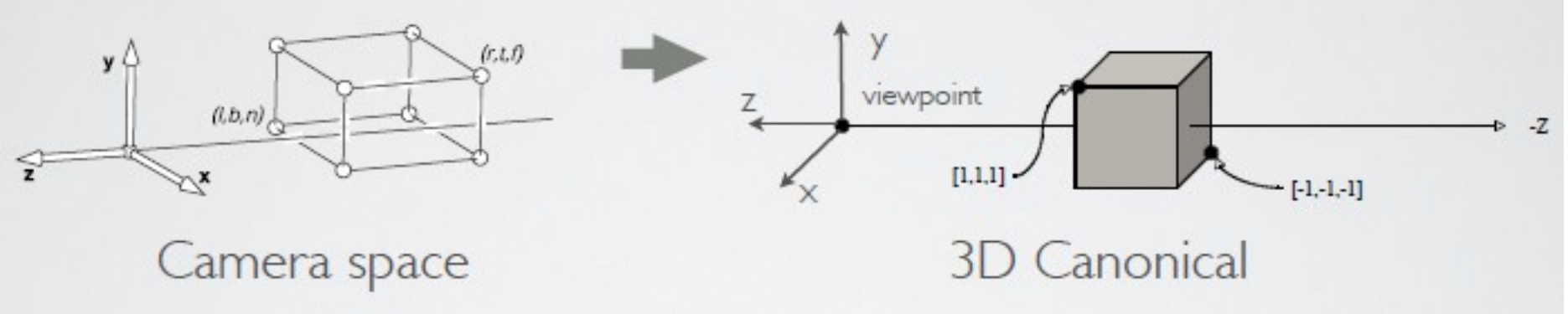
# Orthographic Projection




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

Windowing Transform!

# Orthographic Projection

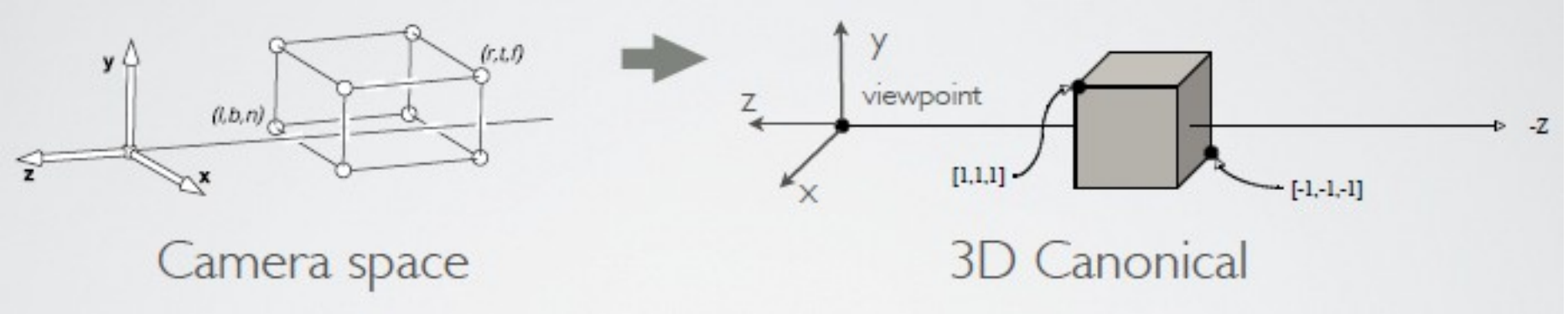



$$M_{orth} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \frac{-(l+r)}{2} \\ 0 & 1 & 0 & \frac{-(b+t)}{2} \\ 0 & 0 & 1 & \frac{-(n+f)}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, scale the sides to have the right lengths

First, translate so the origin is at the center

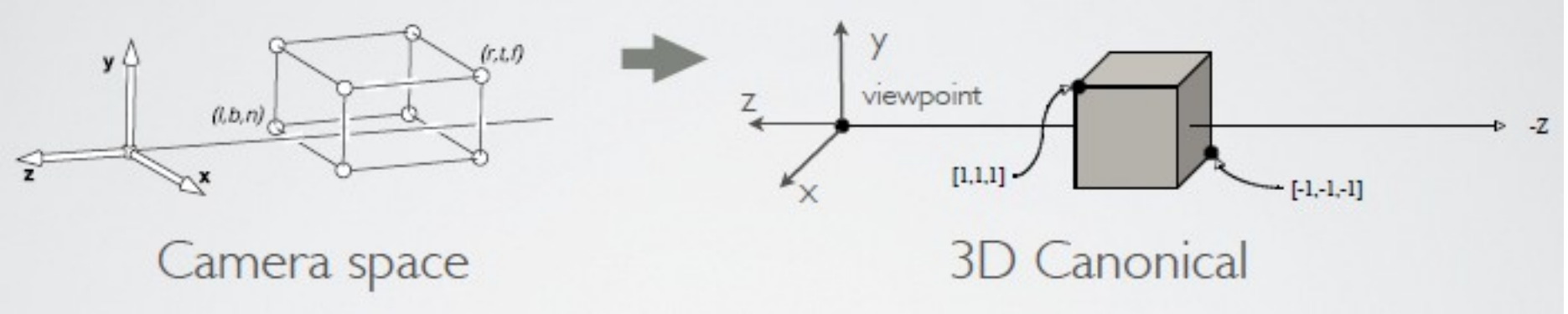
# Orthographic Projection




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

Putting them together

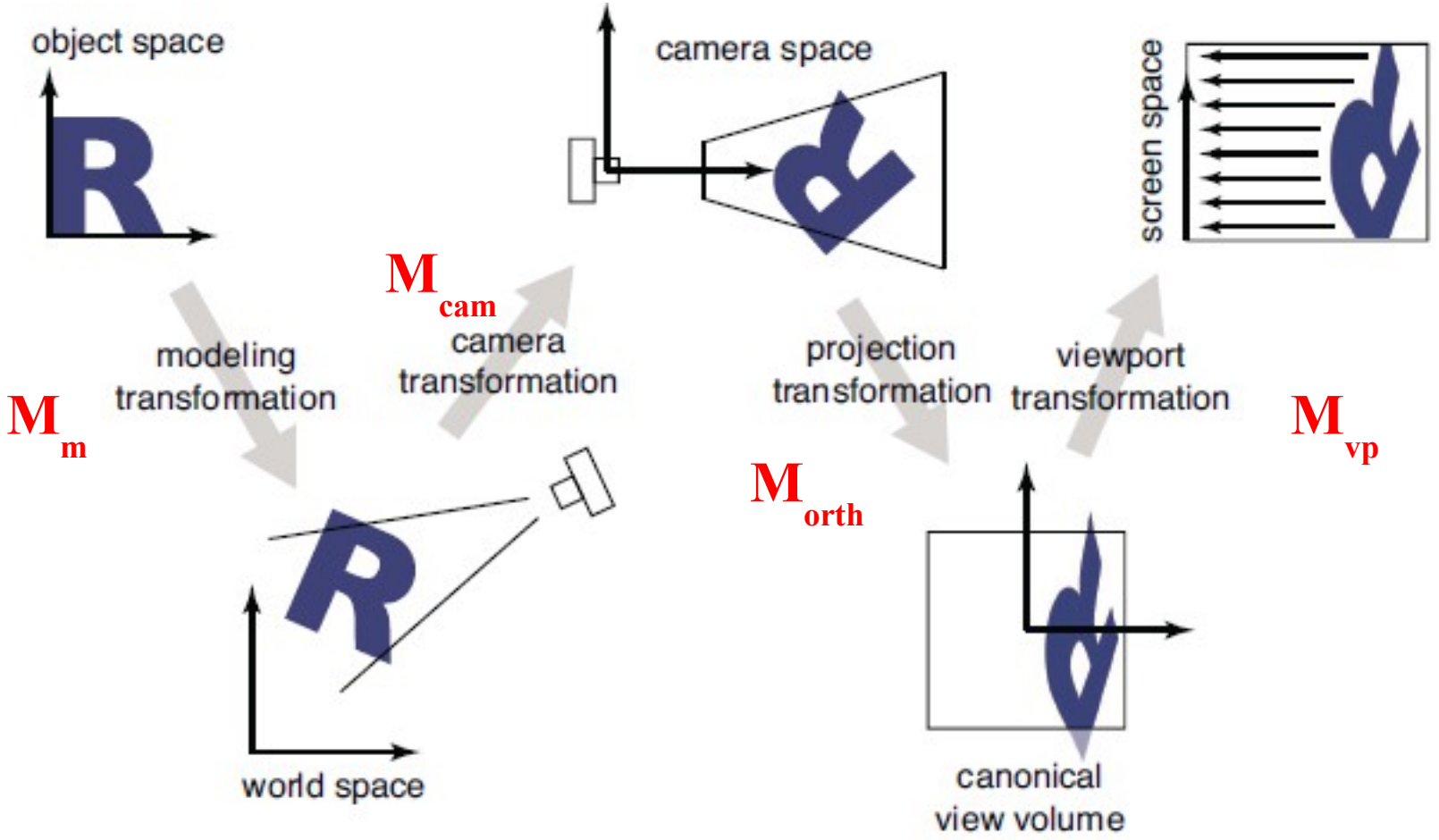
# Orthographic Projection




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

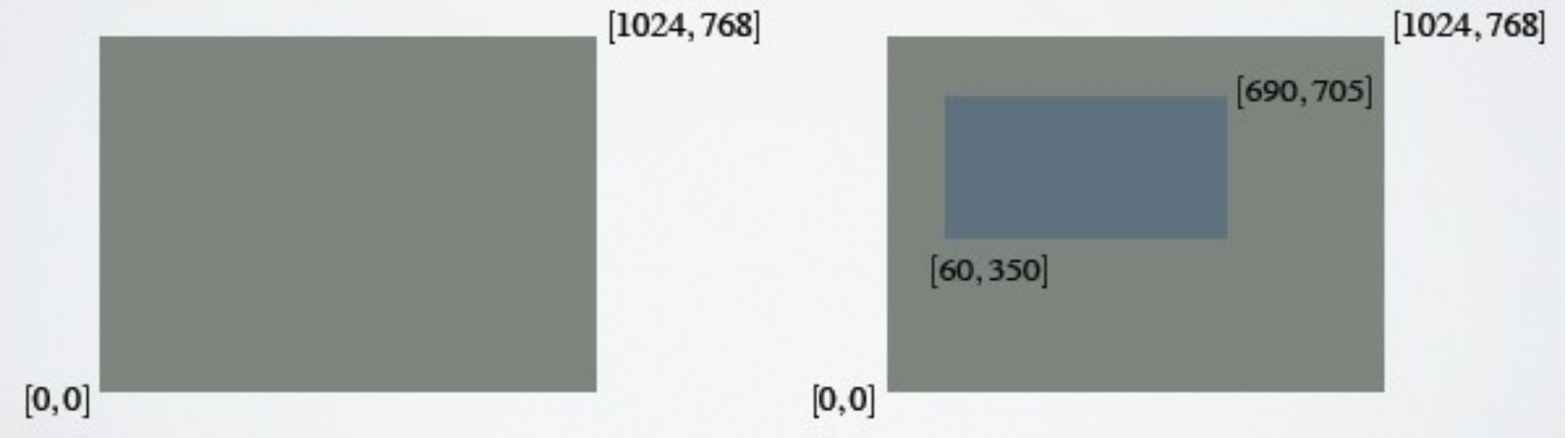
Why do we keep the $z$ coordinate?

# Viewport Transformation




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

# Screen Space

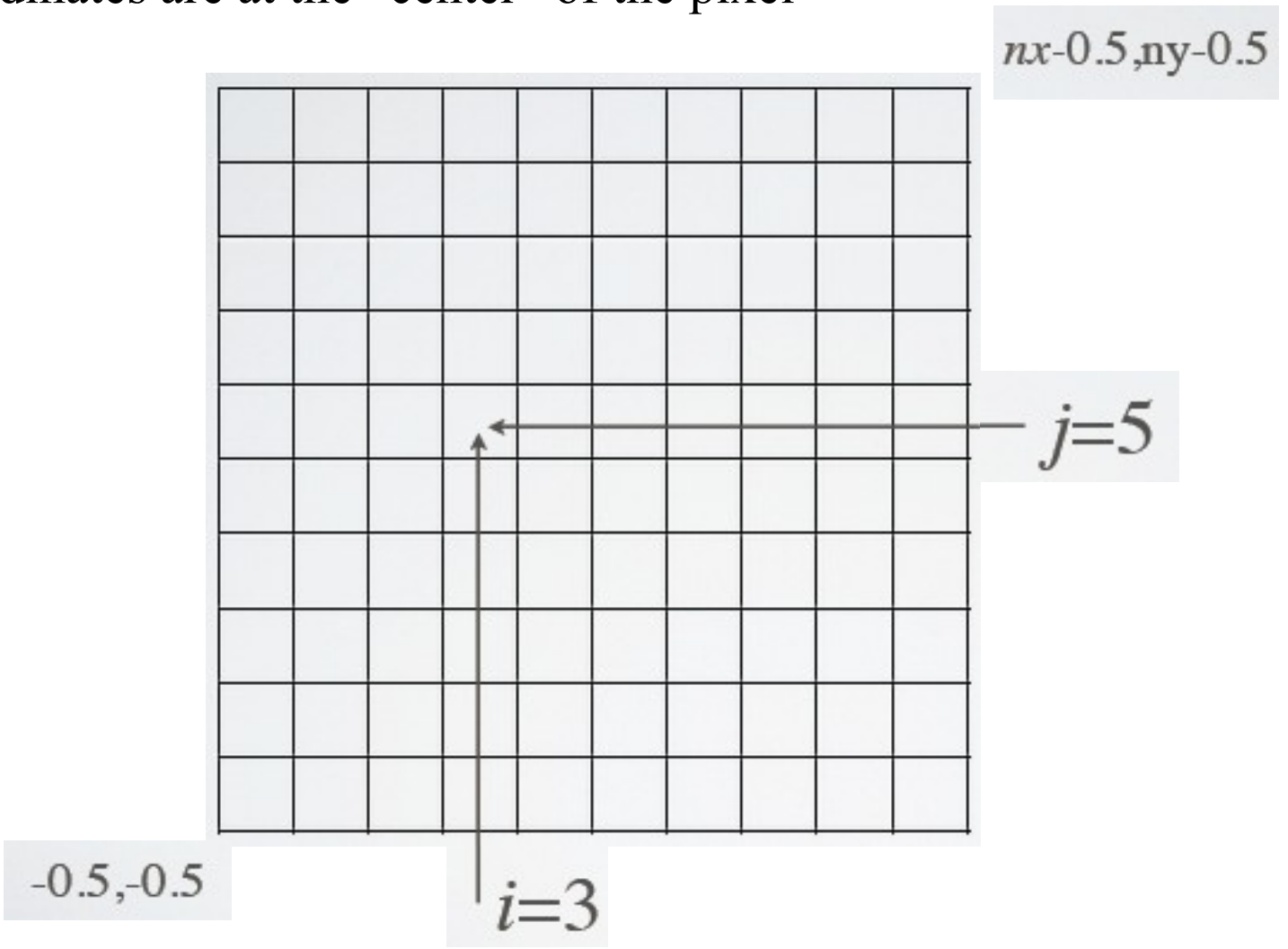



Screen

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

# Screen Space

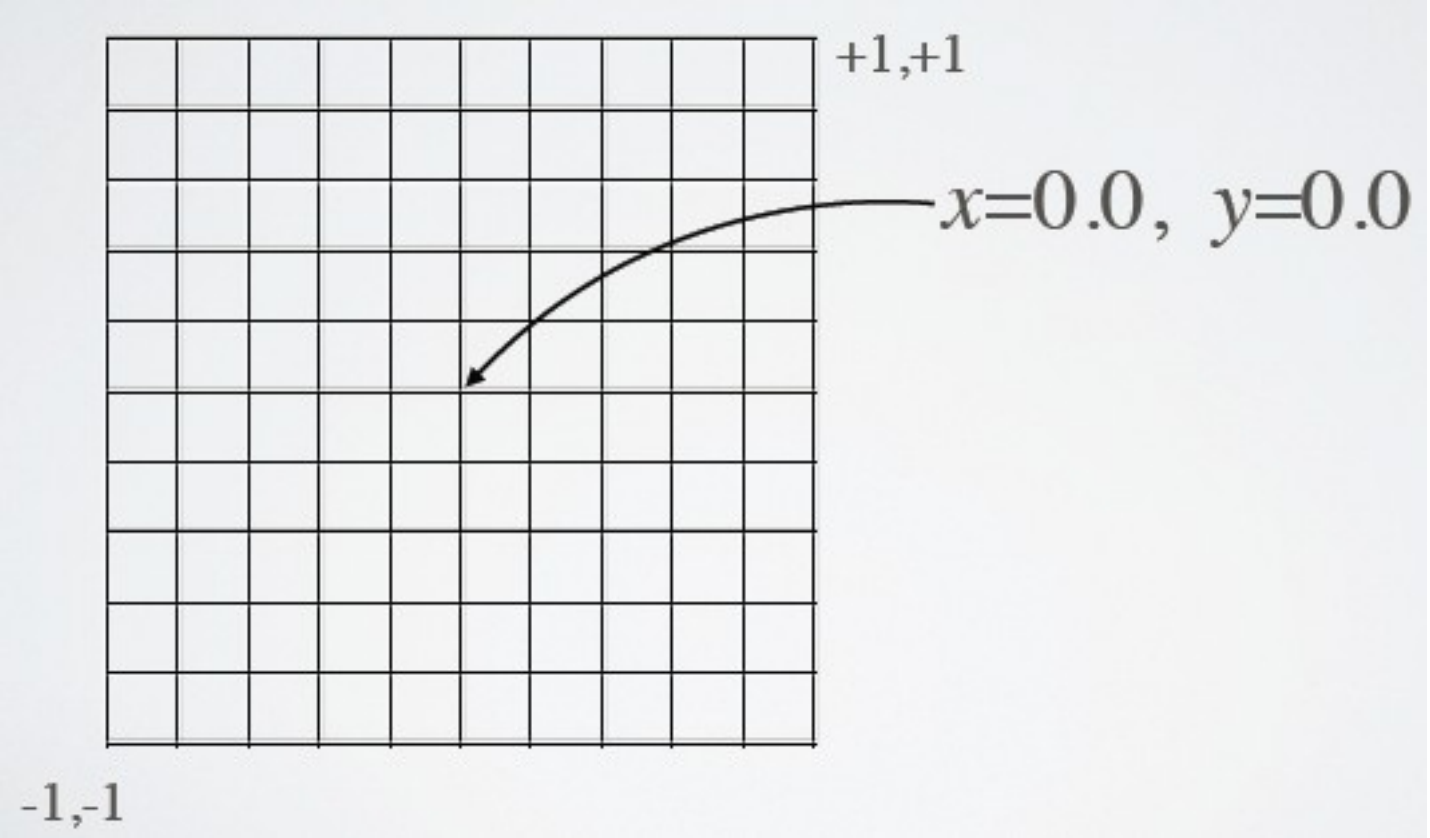
Pixels coordinates are at the “center” of the pixel




Screen of width $n_x$ and height $n_y$ pixels

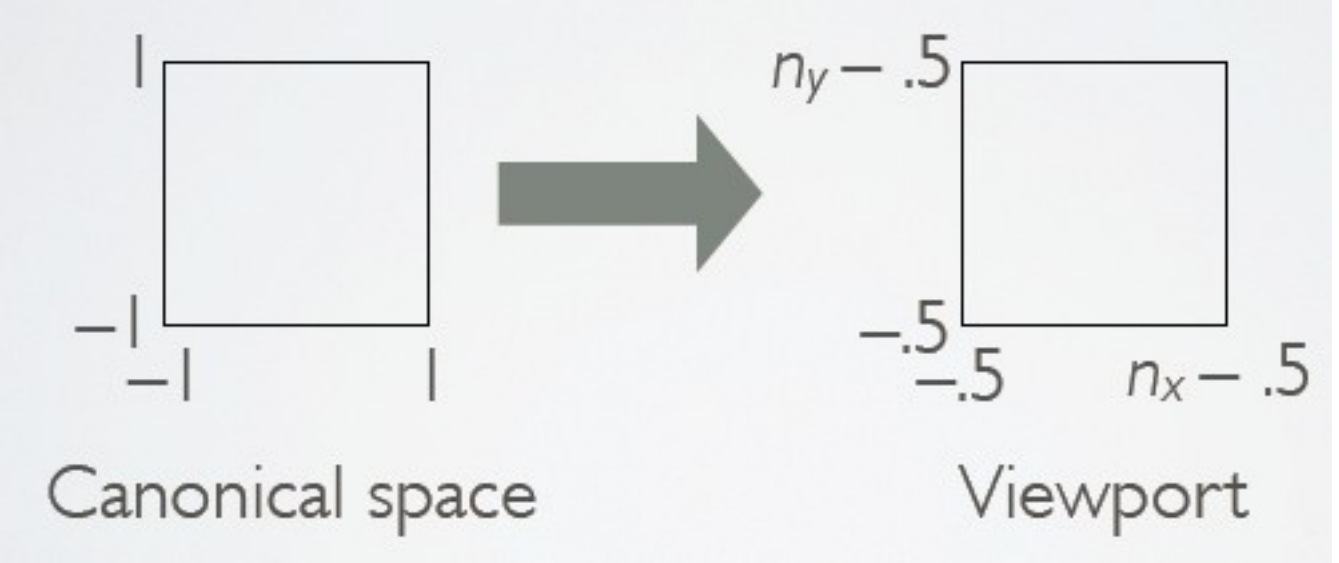
# 2D Canonical View Space

The *xy* plane of the canonical view volume



# 2D Viewport Transformation

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

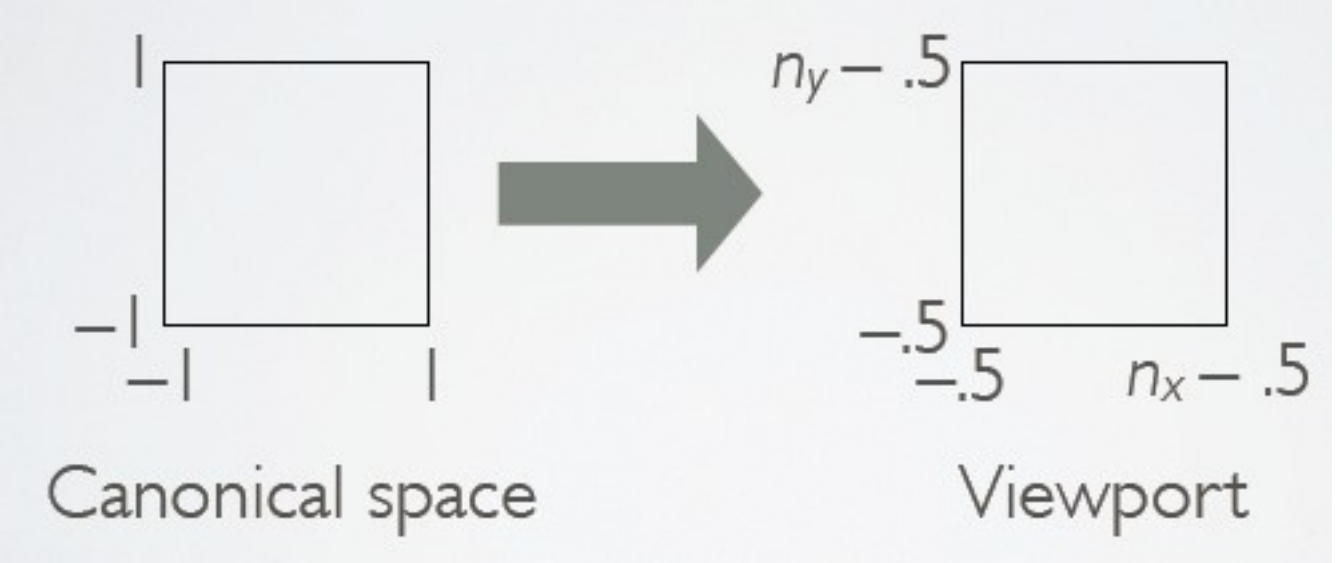



How do we do that?

Another windowing transform!

# 2D Viewport Transformation

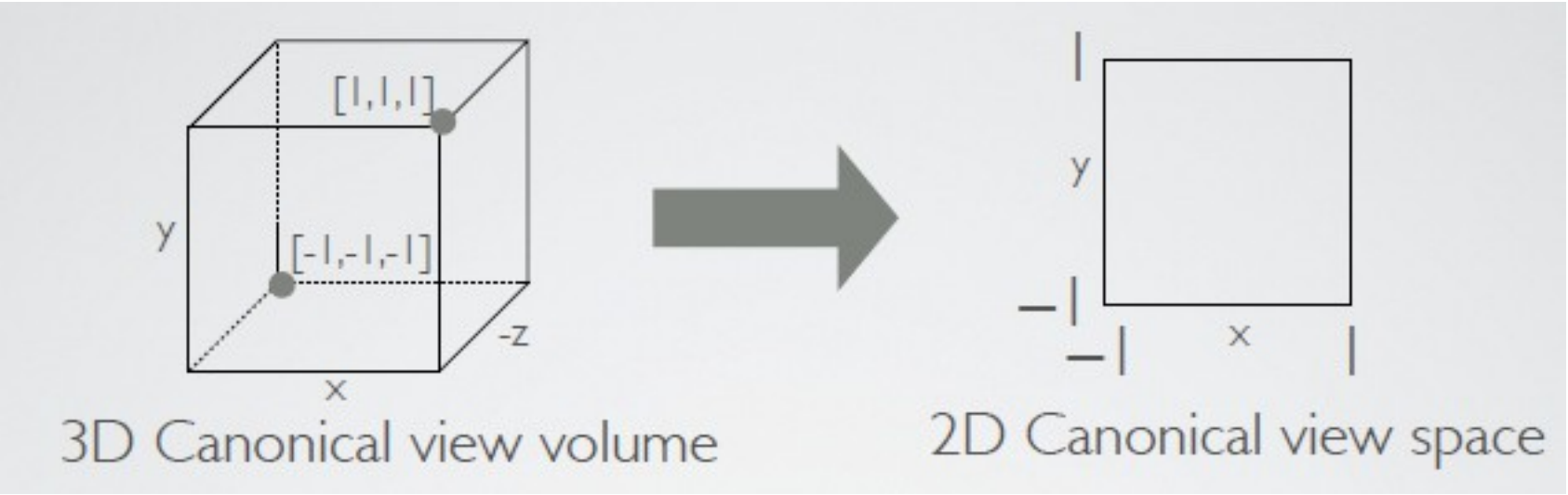
Converts 2D points from the canonical space to screen space i.e. pixels




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

# 3D Viewport Transformation

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

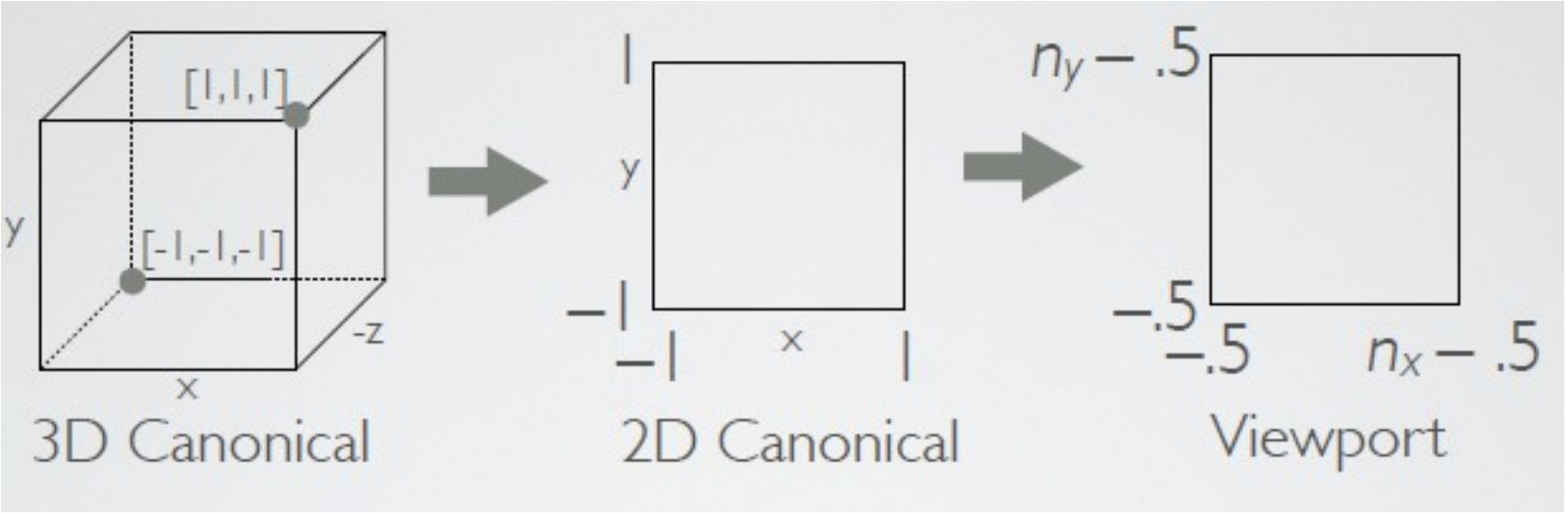



$$\begin{bmatrix} x_{canonical} \\ y_{canonical} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix}$$

Drop z-coordinate. Orthographic Projection!

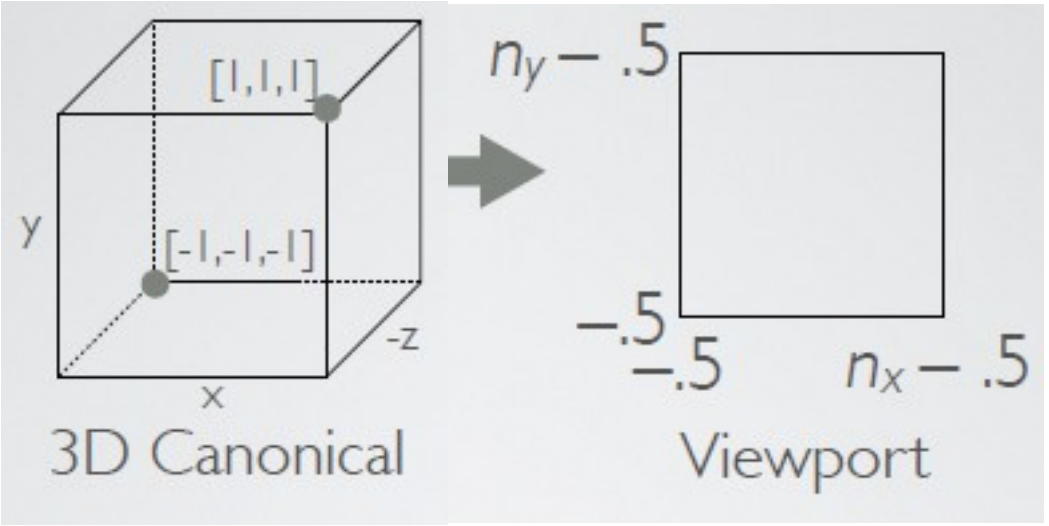
# 3D Viewport Transformation

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




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

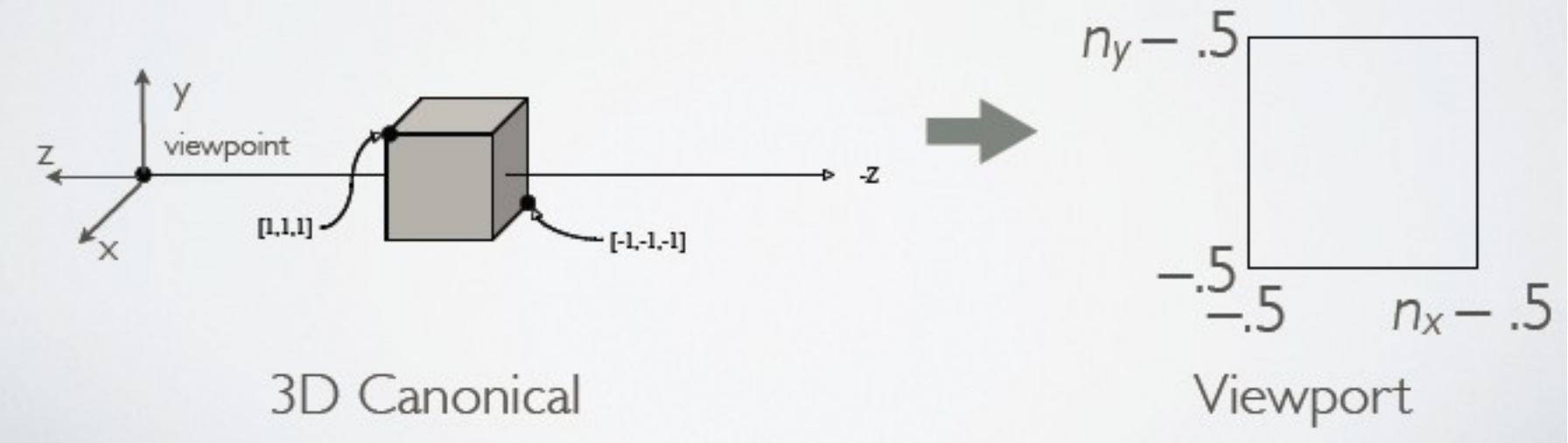
# 3D Viewport Transformation




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

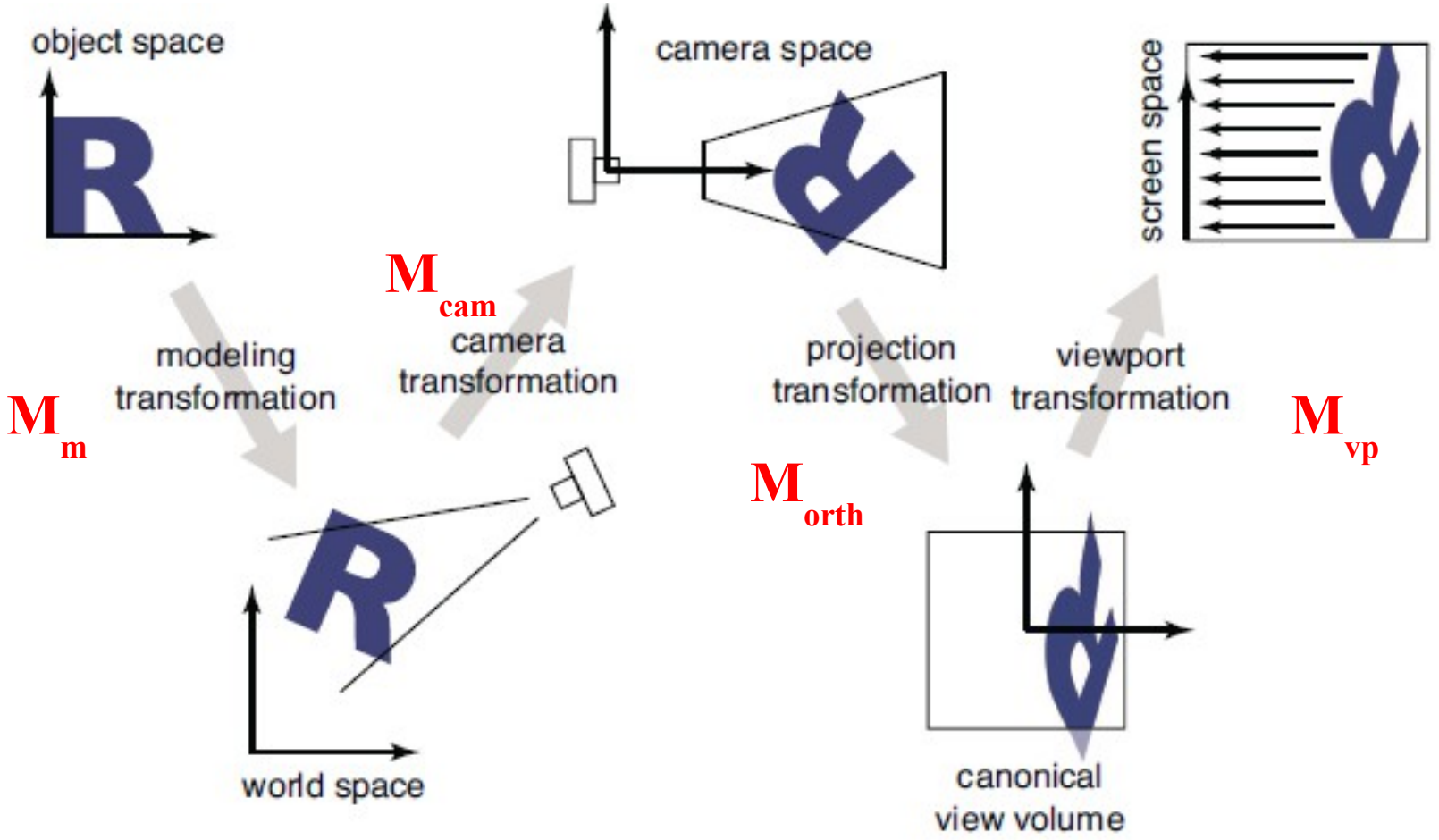
Why do we keep the $z$ coordinate?

# 3D Viewport Transformation




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

# Orthographic Transformation Pipeline

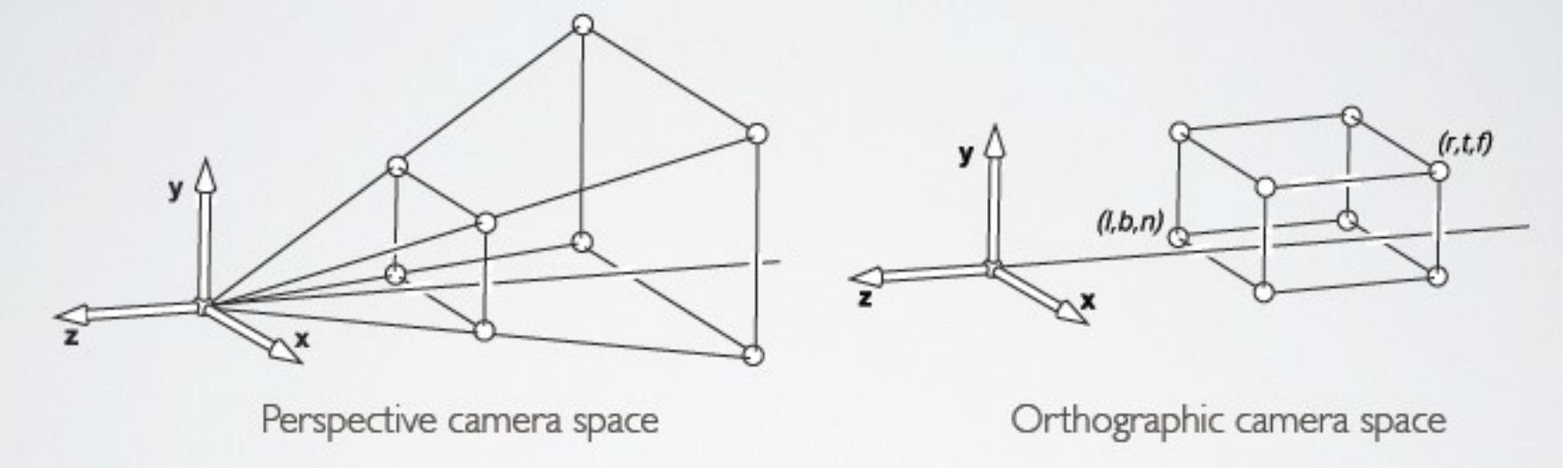


# Orthographic Transformation

- 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: $M_{vp}$

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

# Perspective Projection




Perspective View Volume: Frustum

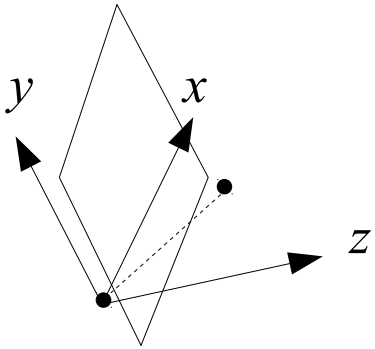
Orthographic View Volume

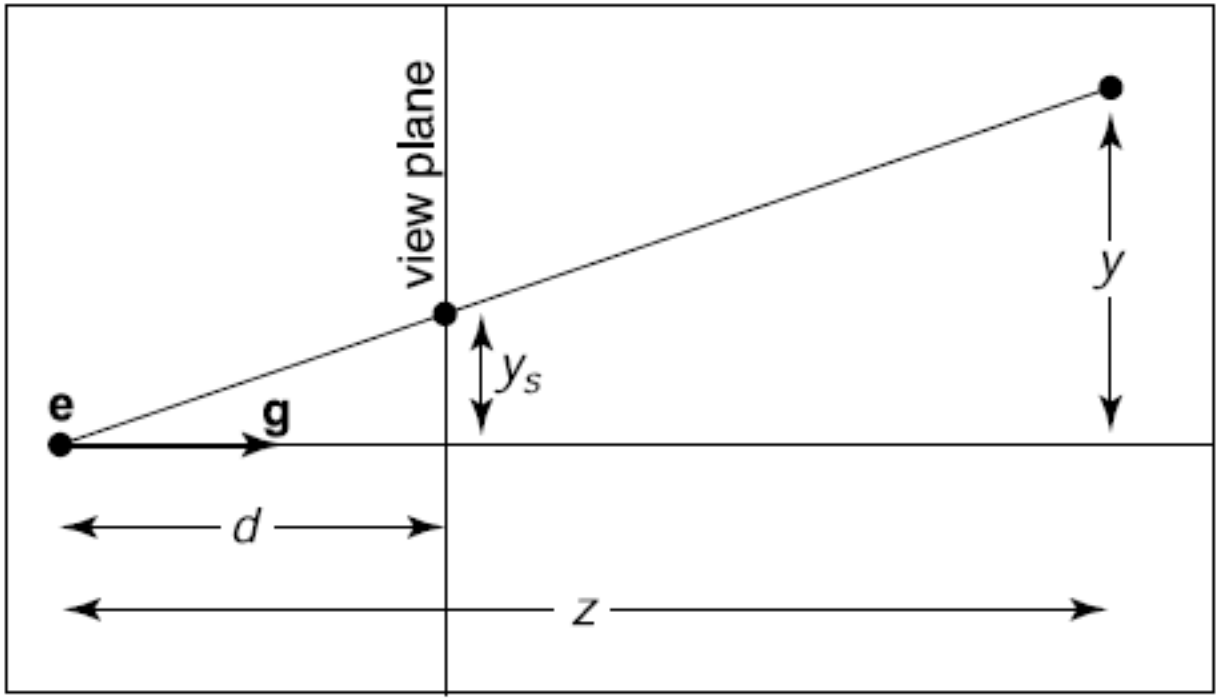
Projection lines go through the camera center !

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

# Perspective Projection

Consider first the $y$ coordinate



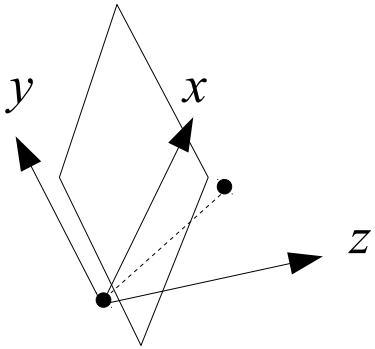

Similar Triangles

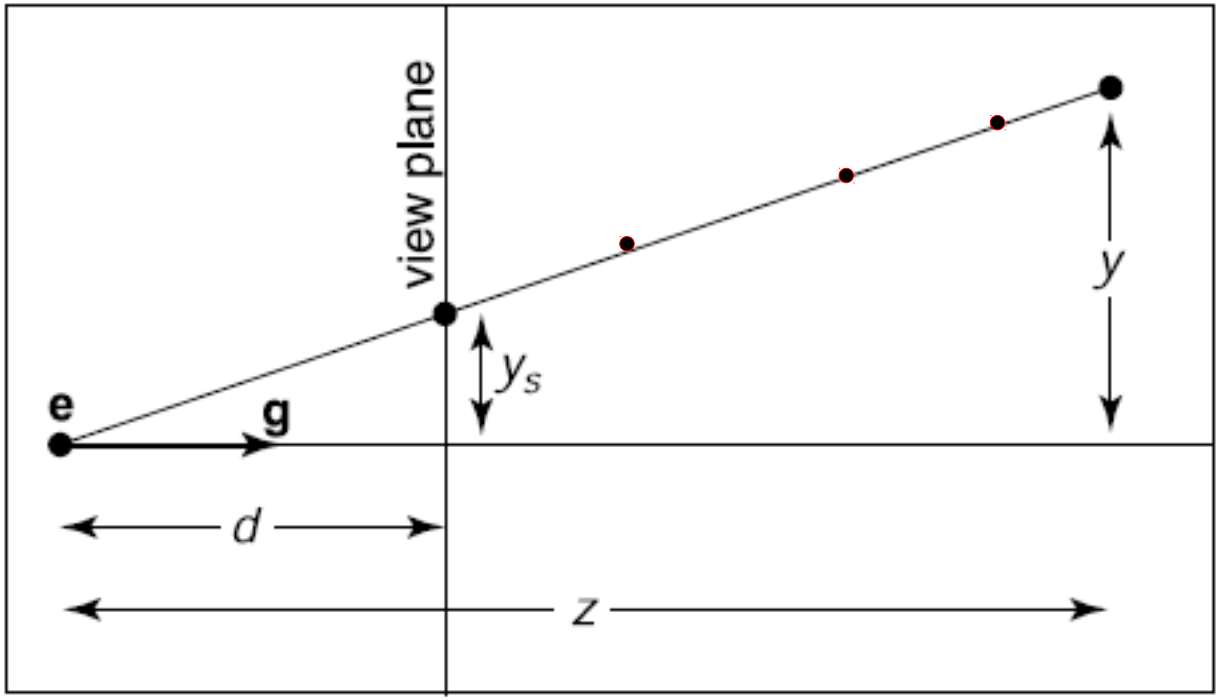
$$\frac{y_s}{d} = \frac{y}{z}$$

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

# Perspective Projection

Consider first the $y$ coordinate



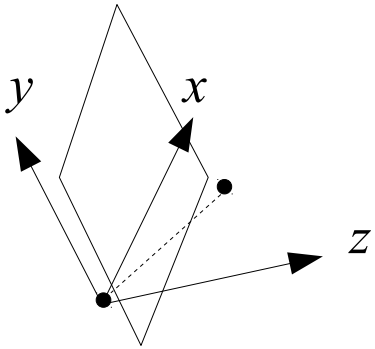



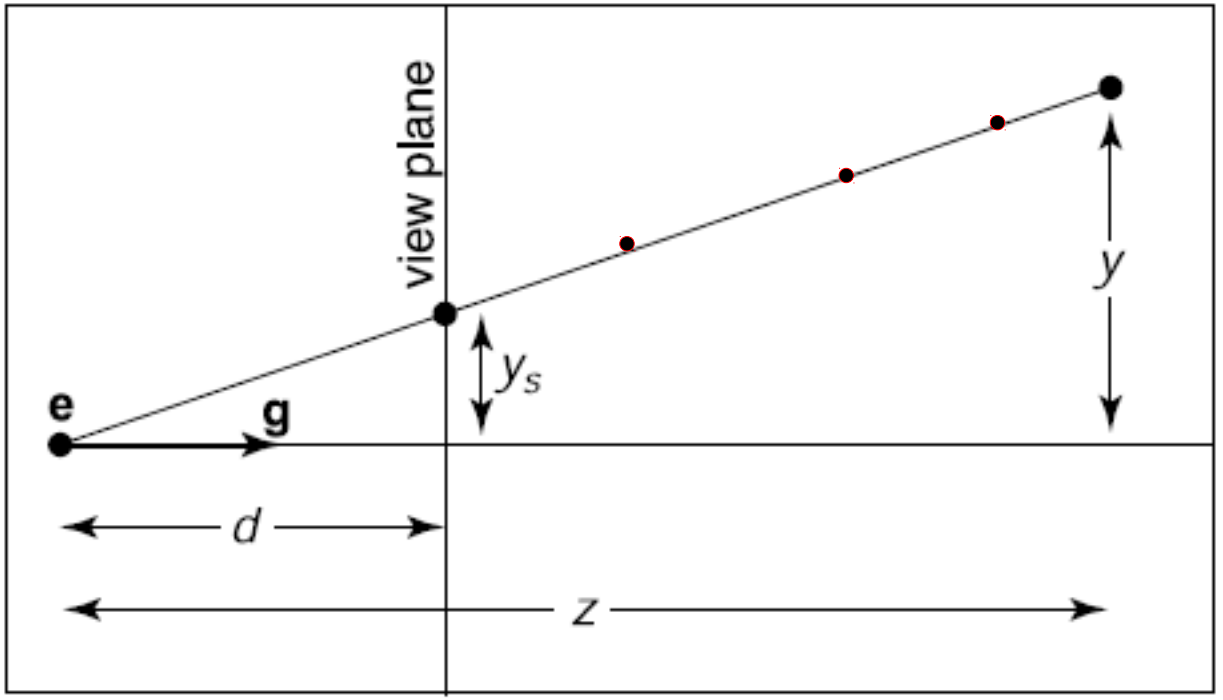

$$y_s = \frac{dy}{z}$$

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 $w$

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

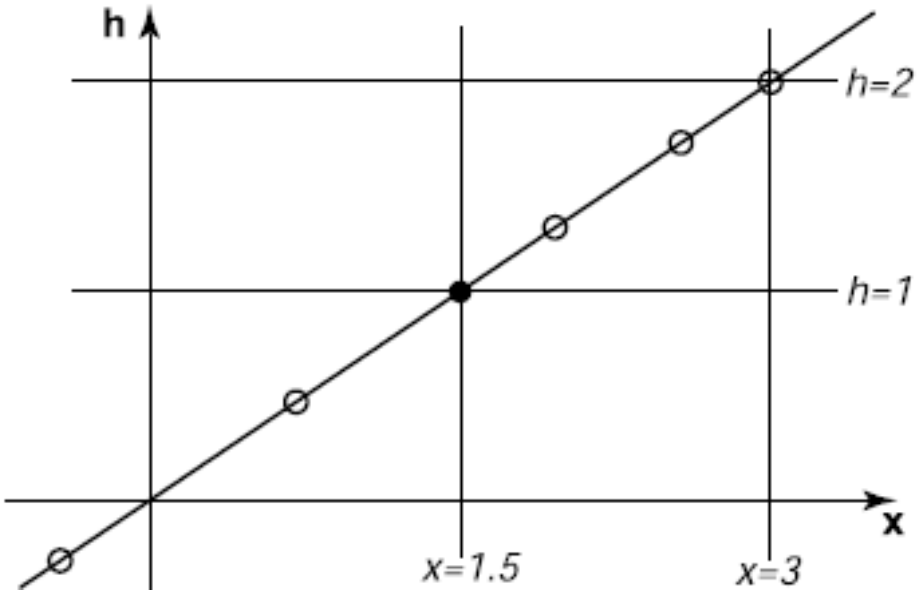
Divide by $w$ to go back to 3D

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, 1.5 $h$] for all $h$

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

# Perspective Projection

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

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:

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

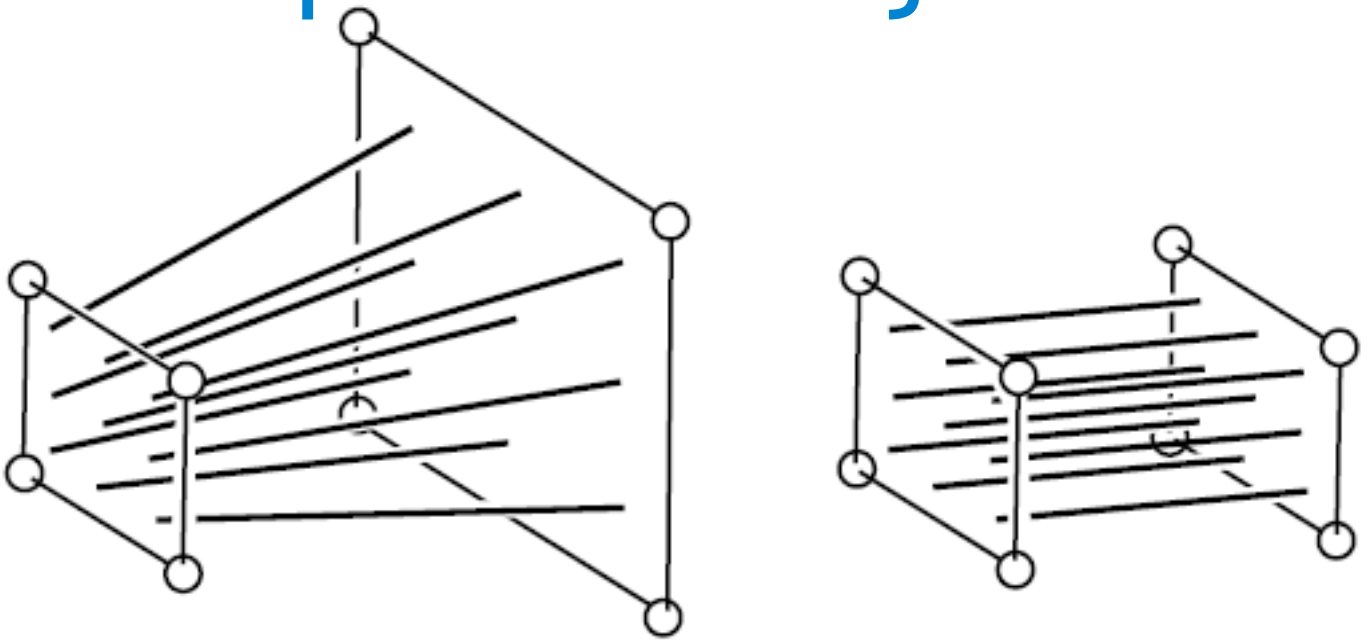
Notice that points with $z = n$ don't change.

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

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

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

# Perspective Projection

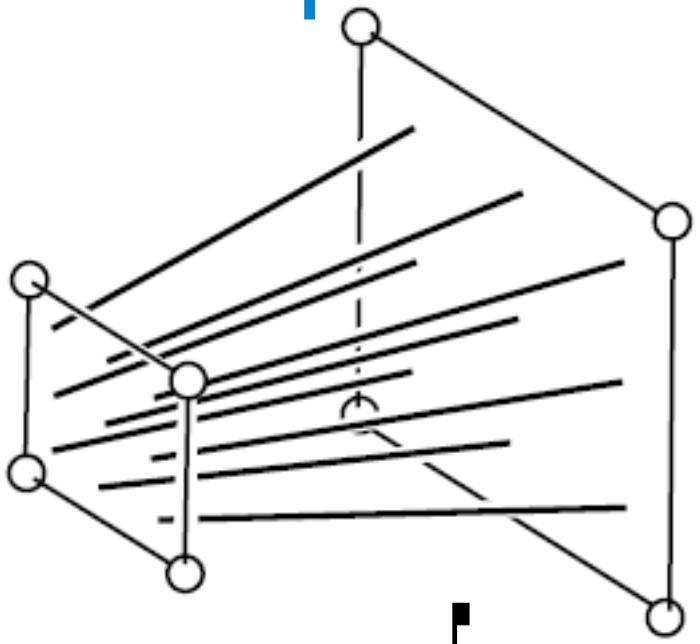


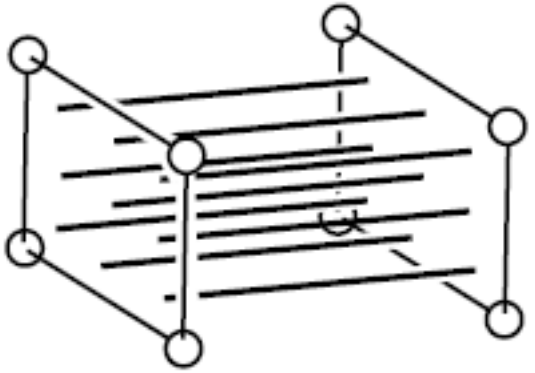
$$P = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

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

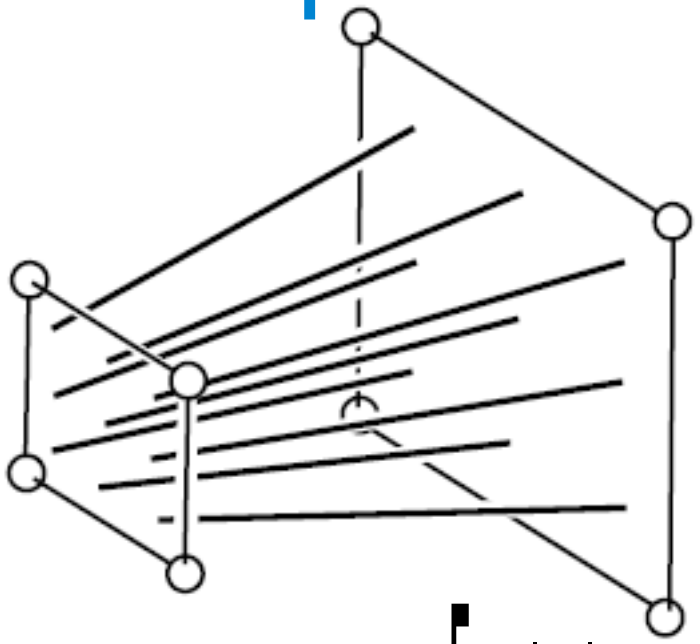


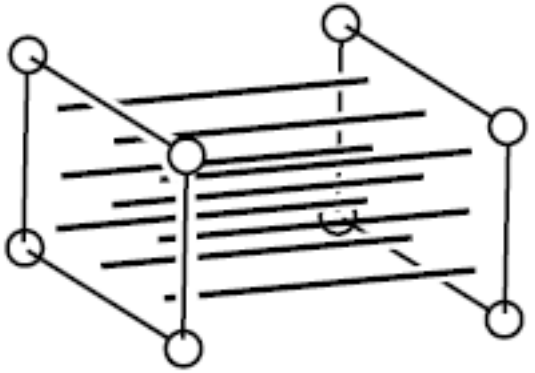


$$M_{per} = M_{orth} P = \begin{bmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{-2nf}{n-f} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The complete Perspective Projection matrix (including the orthographic transformation)

# Perspective Projection





When we are given $|n|$ and $|f|$

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

The Perspective Matrix for OpenGL (and the homework)

# 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  M   PM  M
          vp orth  cam m
For each line segment (a, b)
  p = Ma
  q = Mb
  drawline(xp/hp, yp/hp, xq/hq, yq/hq)
```

# Recap

- Viewing
- Projections
  - Orthographic
  - Perspective
- Transformations Pipeline