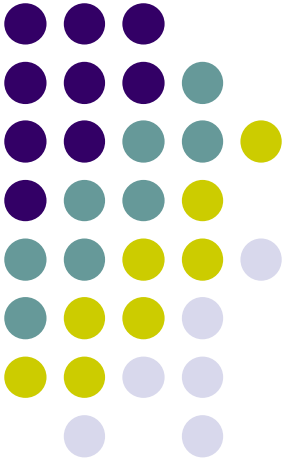
Computer Graphics (CS 4731)
Projection

Joshua Cuneo

Computer Science Dept.

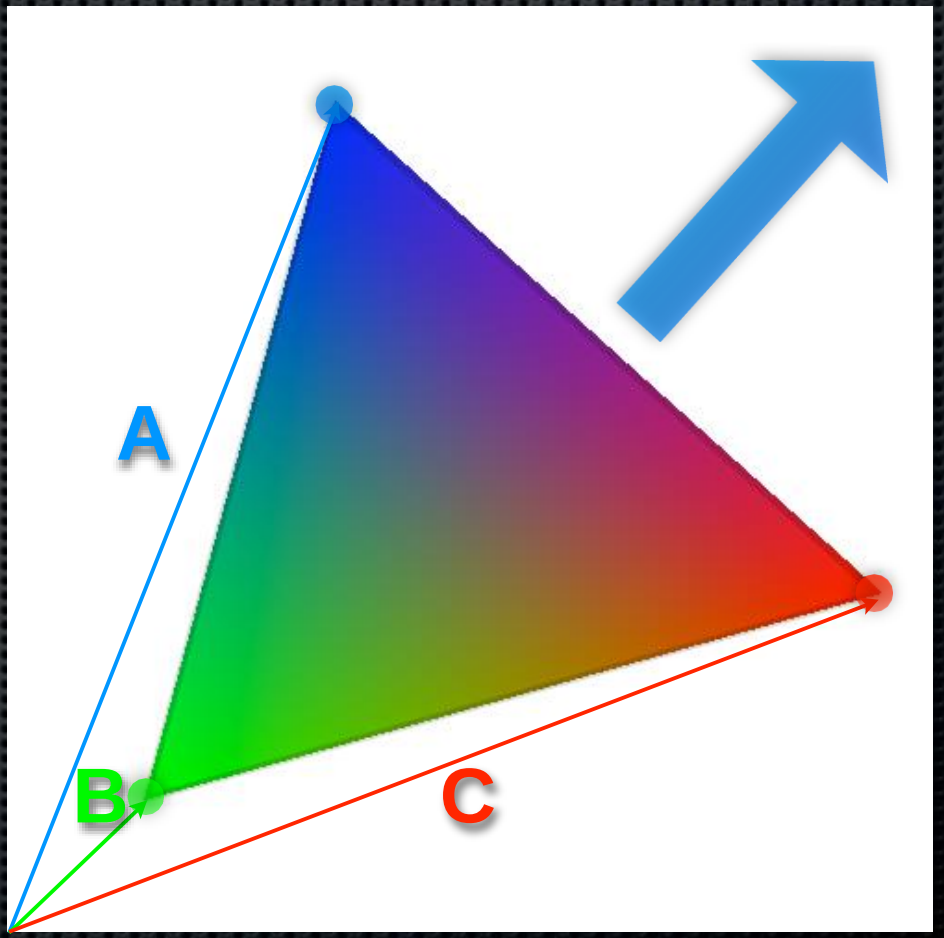
Worcester Polytechnic Institute (WPI)



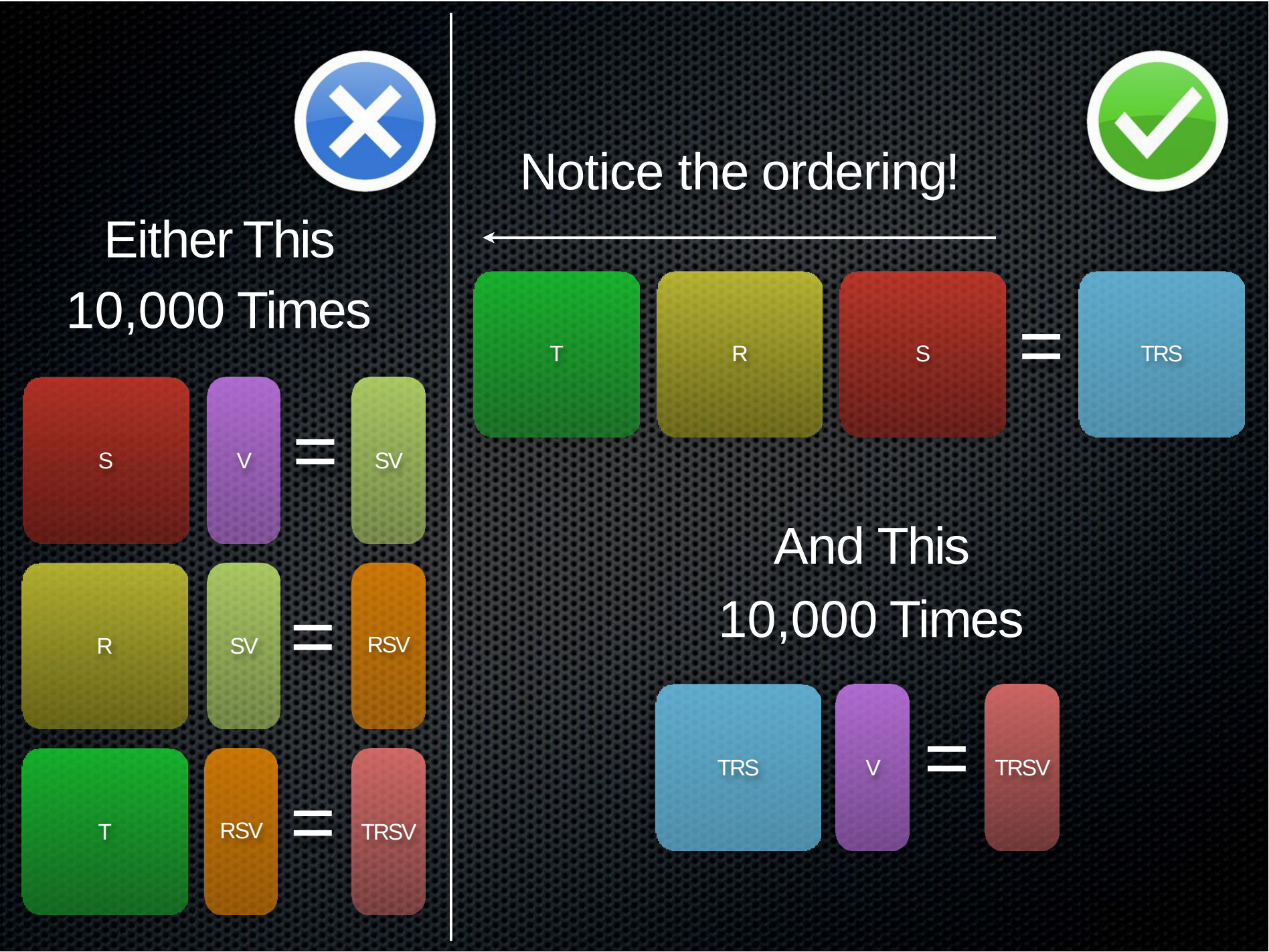
Slides Courtesy of Emmanuel Agu (WPI) and Matt Stoker (Utah)

What Can We Do So Far?

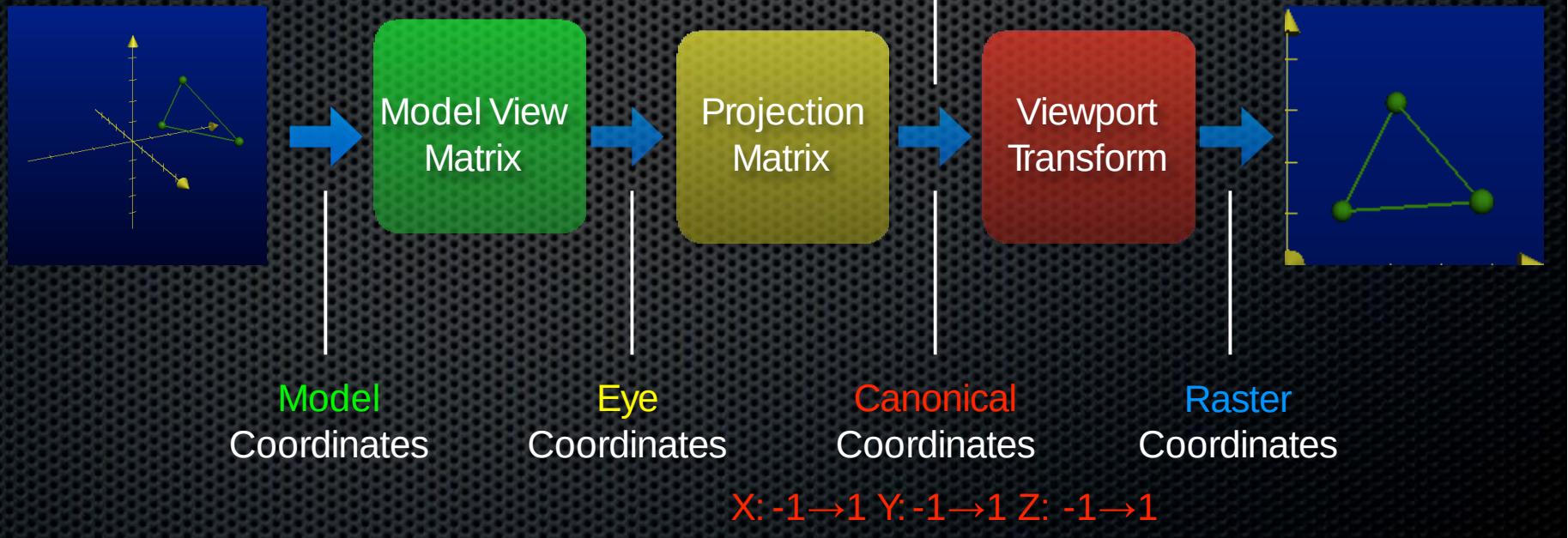
- Create and save rasters
- Define colors, vectors, and triangles
- Draw triangles using interpolated colors
- Transform 3D input geometry

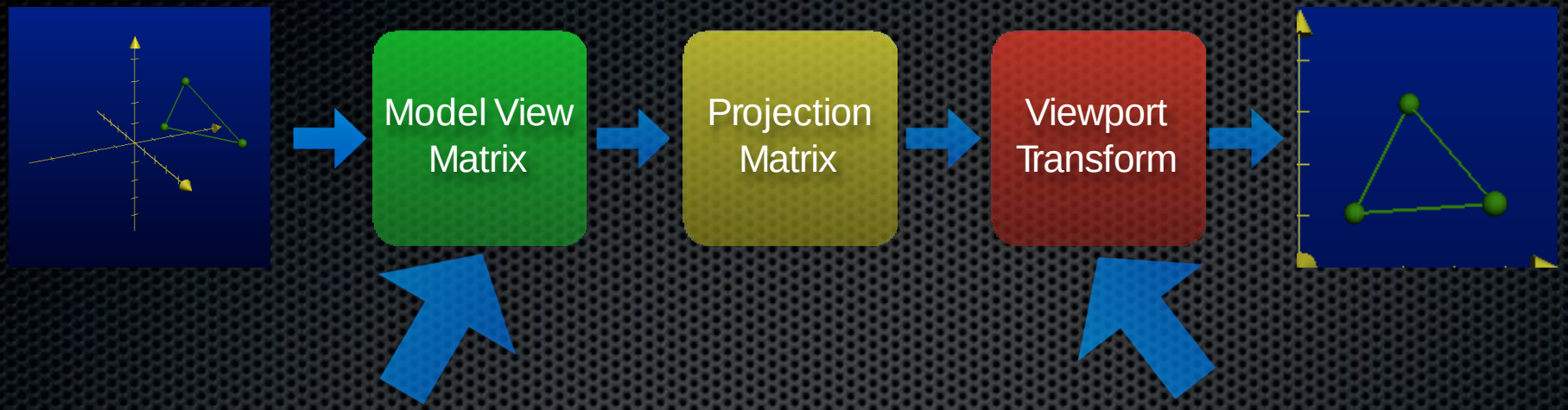


3D Transformations


Either This
10,000 Times
S
V
SV
R
SV
RSV
T
RSV
TRSV
Notice the ordering!
T
R
S
TRS
And This
10,000 Times
TRS
V
TRSV

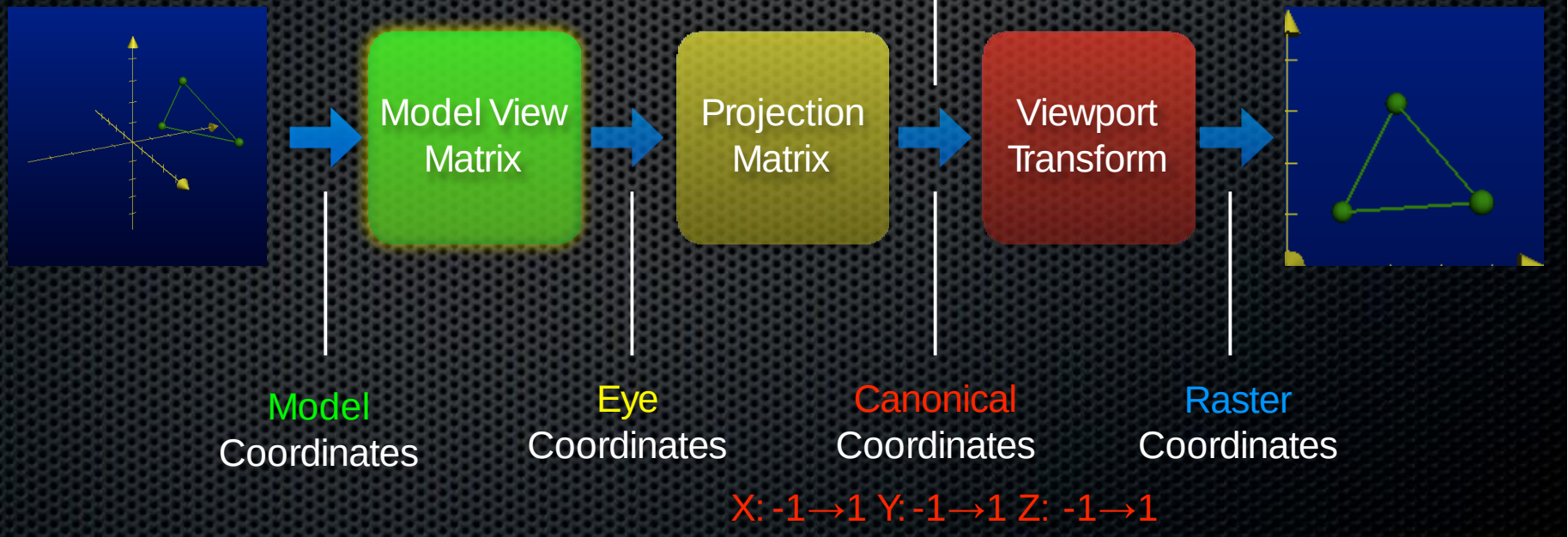
Typical Matrices





To define the projection matrix, lets first define the input and output

Typical Matrices



Model Coordinates

0.9 -0.3 0.4
0.4 0.1 -0.6
-0.8 1.0 0.3

Model Coordinates

Where is the origin?

0.9 -0.3 0.4
0.4 0.1 -0.6
-0.8 1.0 0.3

Model Coordinates

Where is the origin?

0.9 -0.3 0.4
0.4 0.1 -0.6
-0.8 1.0 0.3

Implicitly at 0,0,0

Model Coordinates

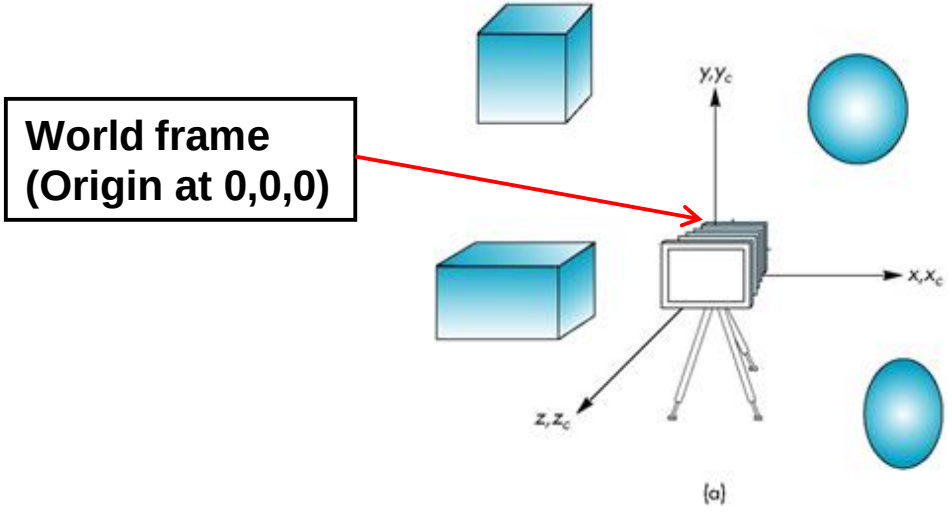
Where is the origin?

0.9 -0.3 0.4
0.4 0.1 -0.6
-0.8 1.0 0.3

Implicitly at 0,0,0
Axes are implicit as well

The World Frame

- Objects/scene initially defined in **world frame**
- **World Frame origin** at (0,0,0)
- Objects positioned, oriented (translate, scale, rotate transformations) applied to objects in **world frame**

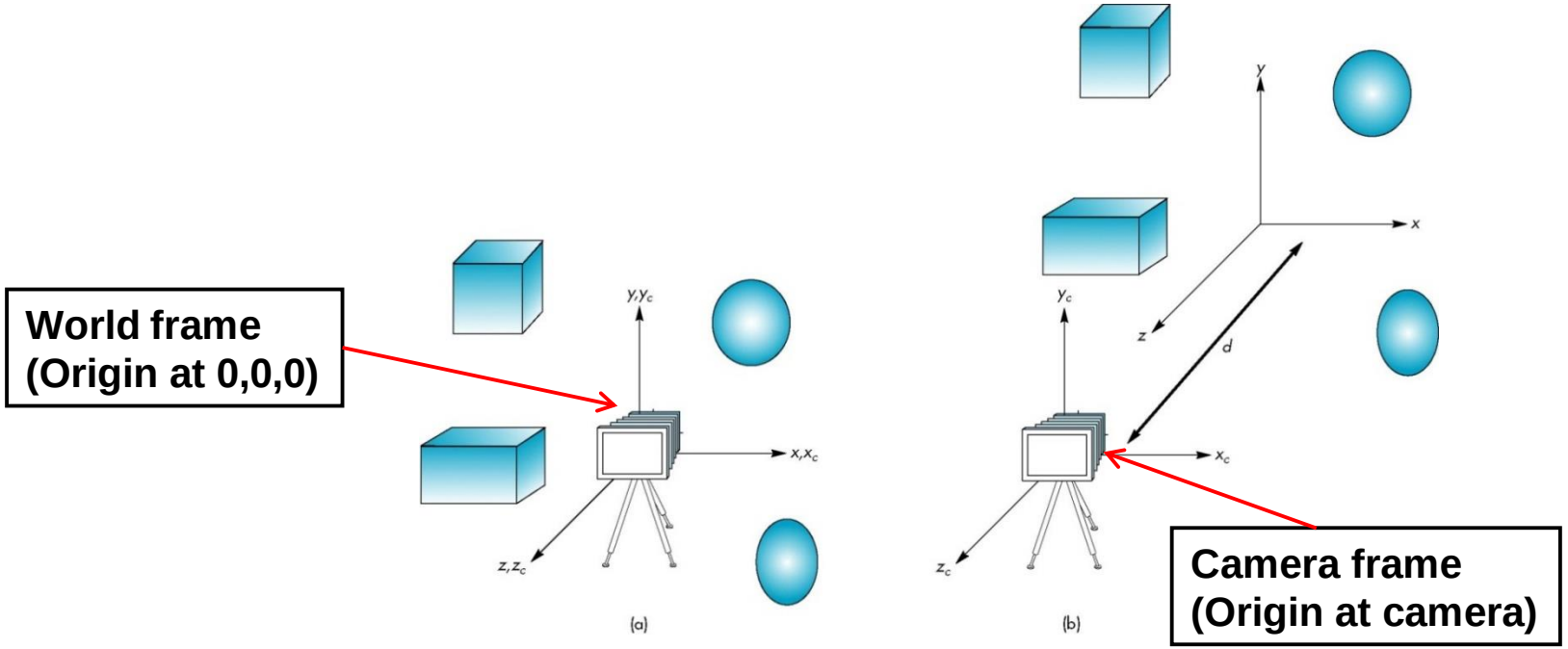


Camera Frame

- More natural to describe object positions relative to camera (eye)
- Why?
  - Our view of the world
  - First person shooter games

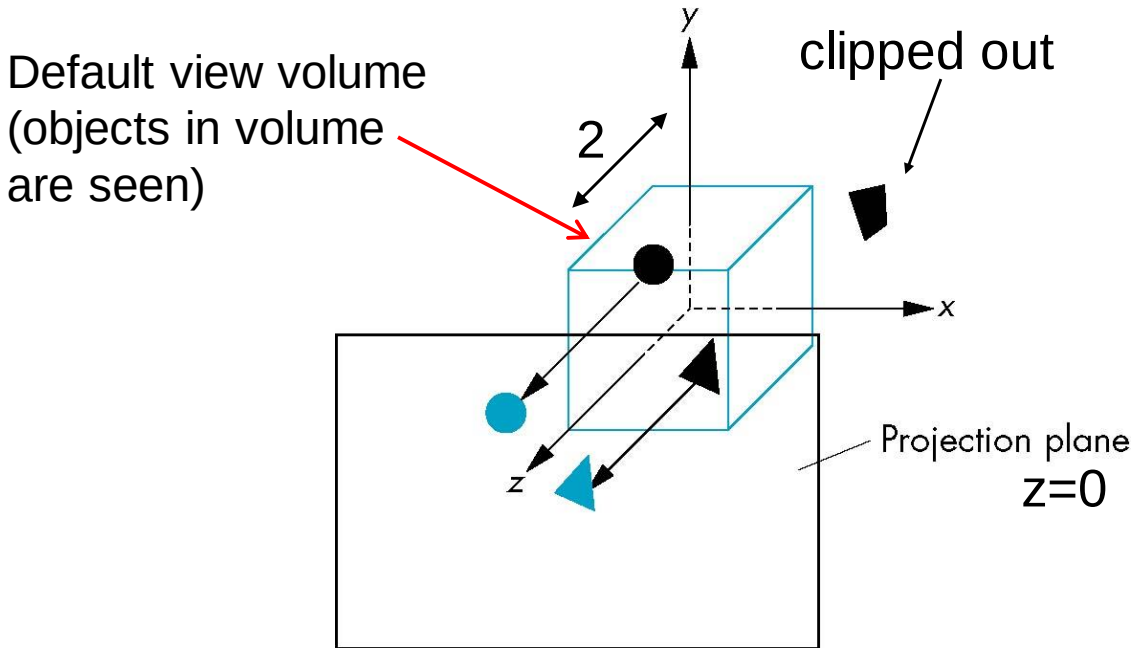
Camera Frame

- **Viewing:** After user chooses camera (eye) position, represent objects in **camera frame** (origin at eye position)
- **Viewing transformation:** Converts object (x,y,z) positions in world frame to positions in camera frame



Default WebGL Camera

- Initially Camera at origin: object and camera frames same
- Points in negative z direction
- Default view volume is cube with sides of length 2

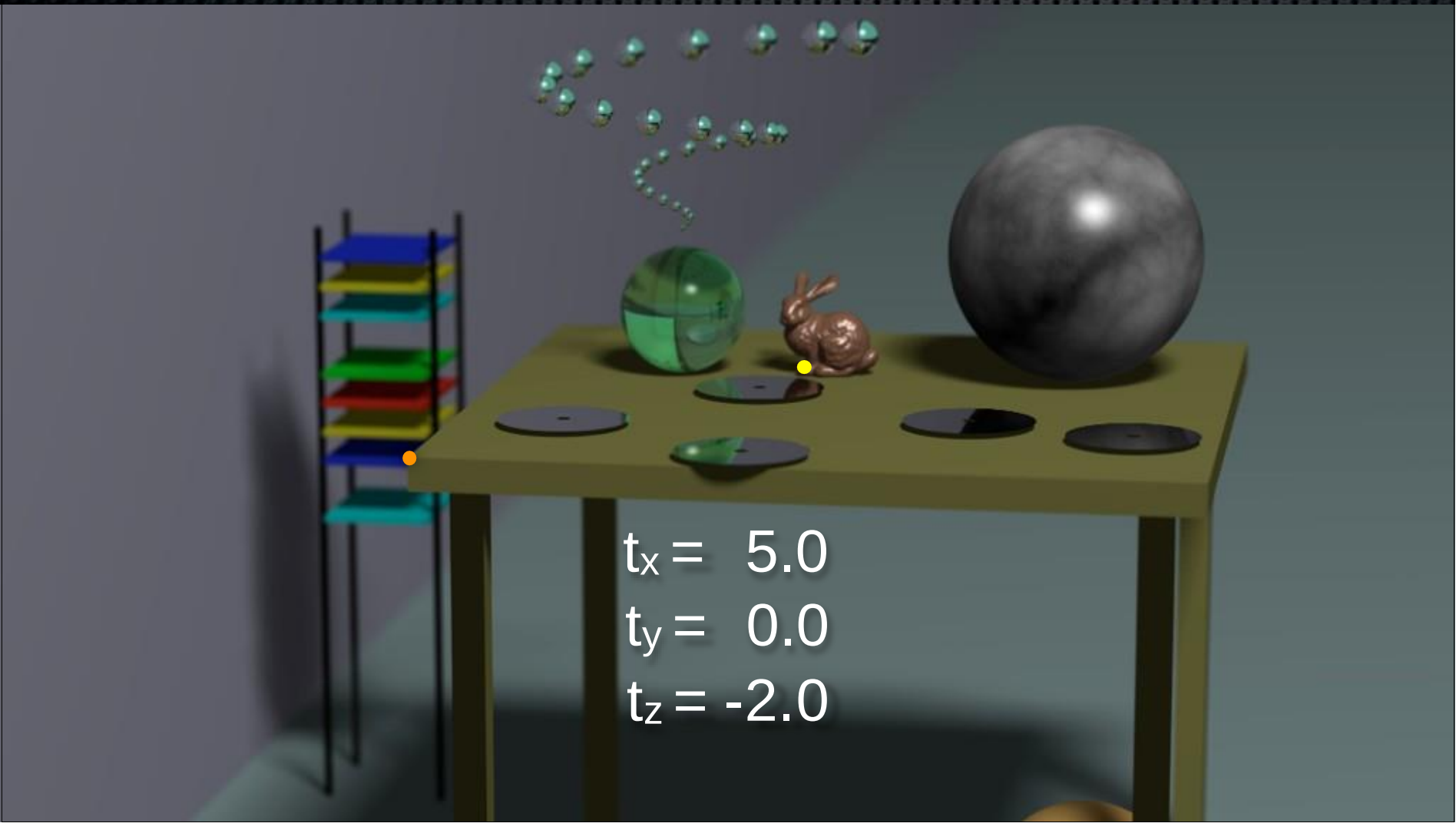


Model Coordinates

Model Coordinates

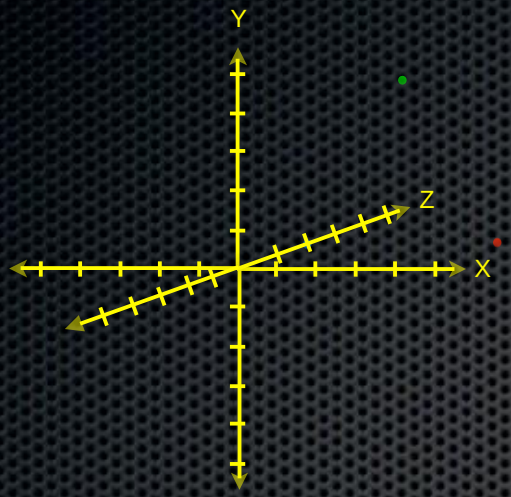
Model Coordinates

Model Coordinates

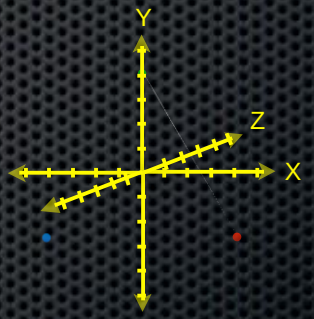


Model Transformations

All Points

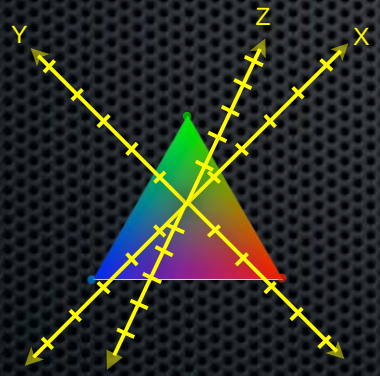


$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



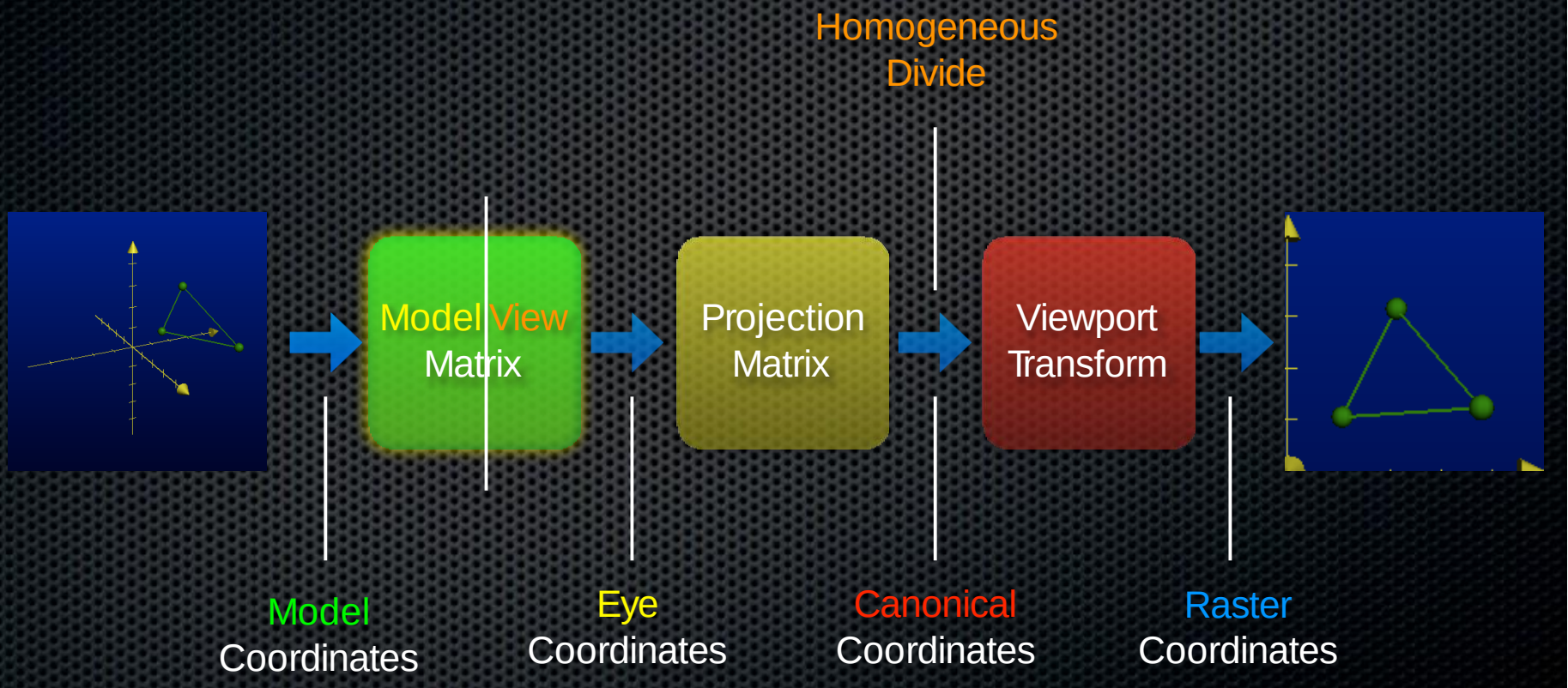
$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

M = vector rotate matrix



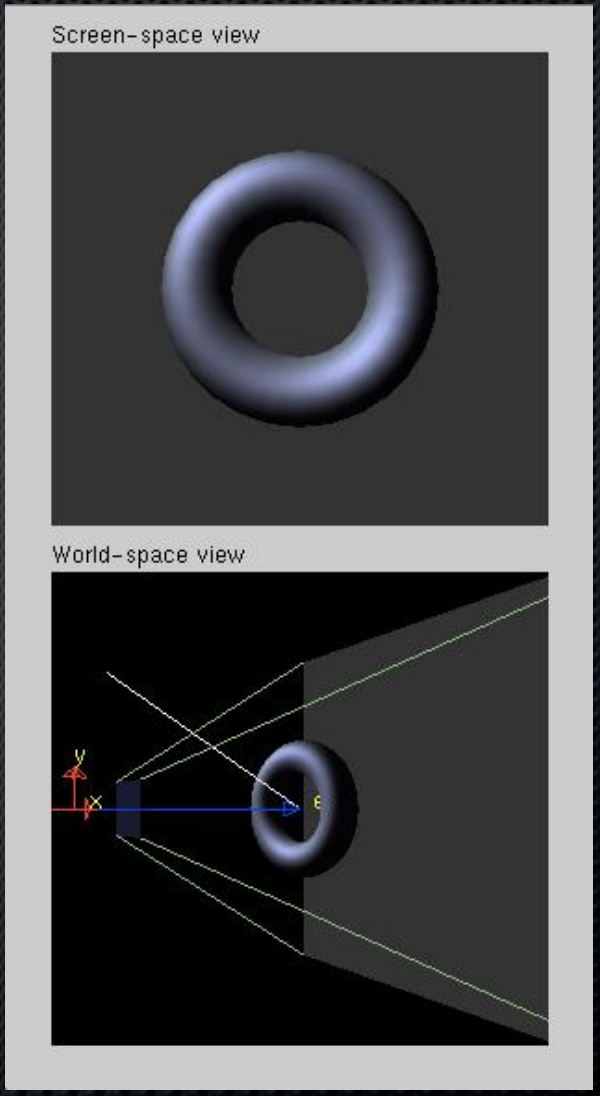
$$\begin{bmatrix} m & m & m & 0 \\ m & m & m & 0 \\ m & m & m & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Typical Matrices



Eye Coordinates

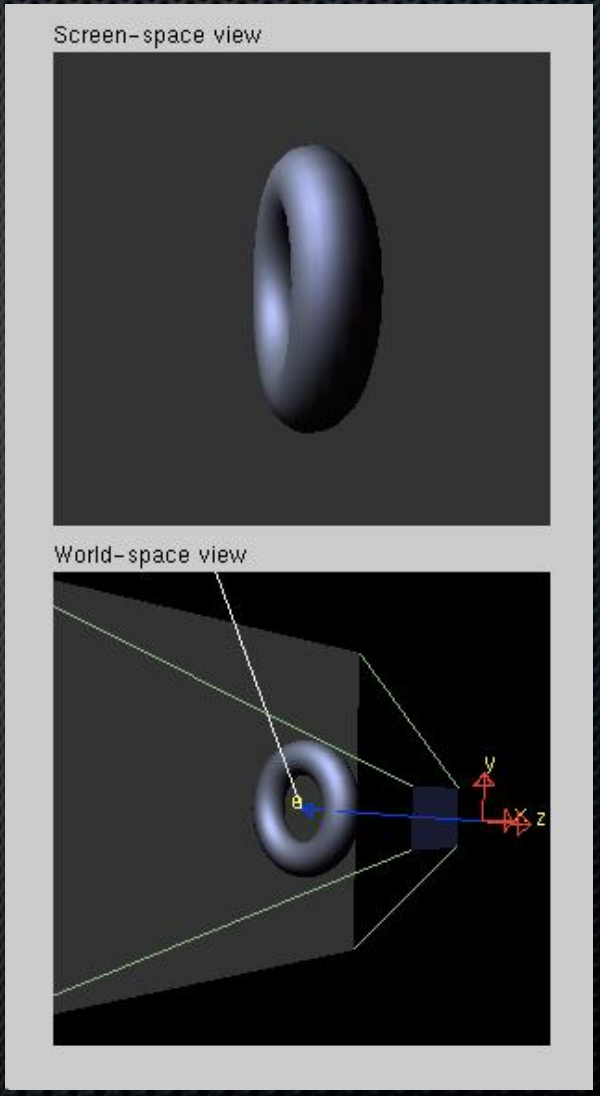
- User positioned at (0,0,0)
- +X axis to the user's right
- +Y axis points up
- User looks down -Z axis by the right hand rule



Eye Coordinates

- User positioned at (0,0,0)
- +X axis to the user's right
- +Y axis points up
- User looks down -Z axis by the right hand rule

The world literally must revolve around you!



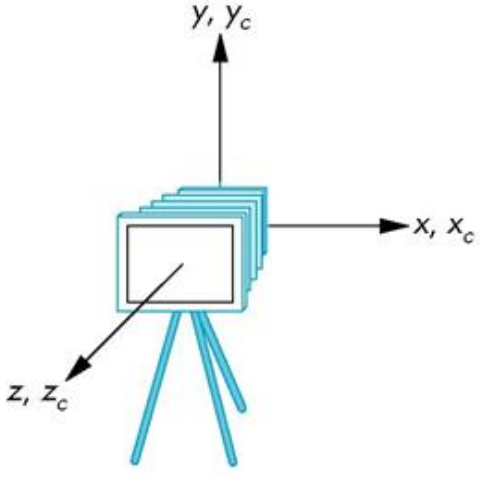
Moving Camera Frame

Same relative distance after
Same result/look

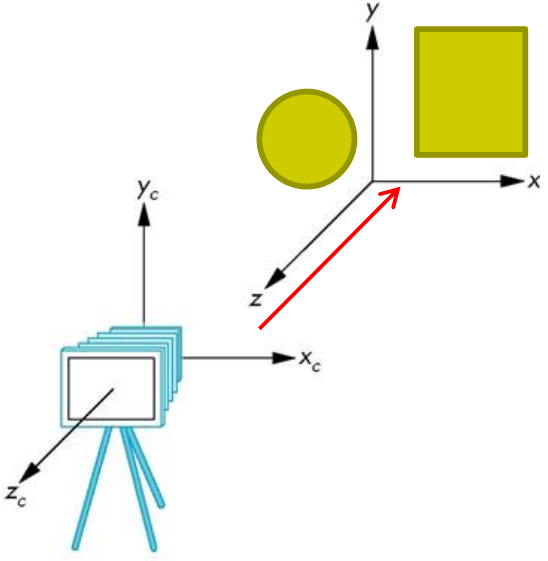
Translate objects -5
away from camera

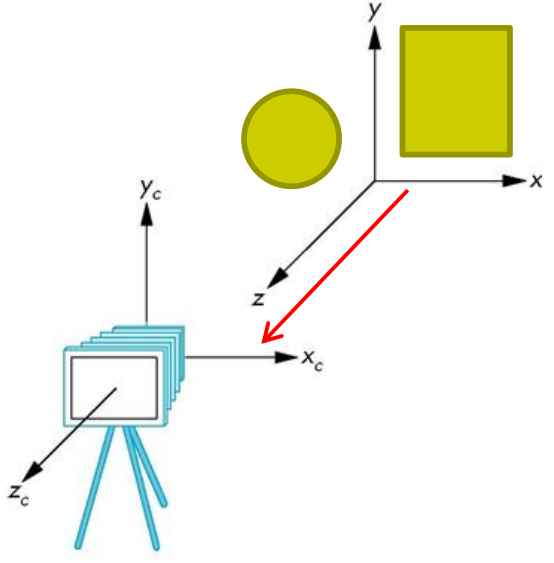
Translate camera +5
away from objects

default frames



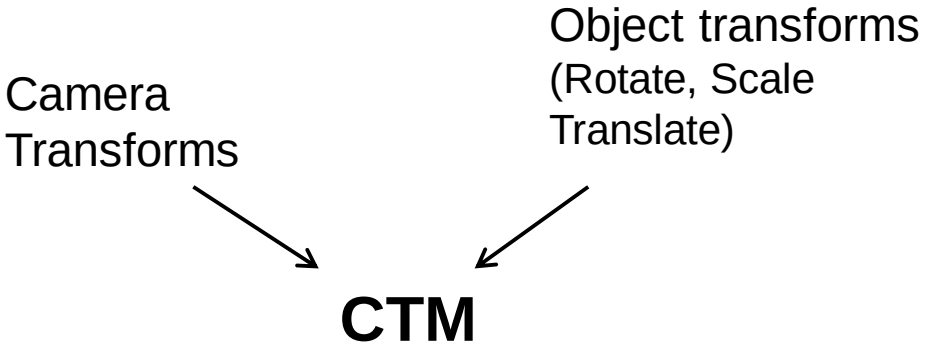
(a)





Moving the Camera Frame

- Object distances **relative to camera** determined by the model-view matrix
  - Transforms (scale, translate, rotate) go into **modelview matrix**
  - Camera transforms also go in **modelview matrix (CTM)**



Model-View Transformations

Translation

$$T = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and } T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scale

$$S = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and } S^{-1} = \begin{bmatrix} \frac{1}{x} & 0 & 0 & 0 \\ 0 & \frac{1}{y} & 0 & 0 \\ 0 & 0 & \frac{1}{z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation

X-Rotation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & -\sin a & 0 \\ 0 & \sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Y-Rotation

$$\begin{bmatrix} \cos a & 0 & \sin a & 0 \\ 0 & 1 & 0 & 0 \\ -\sin a & 0 & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Z-Rotation

$$\begin{bmatrix} \cos a & -\sin a & 0 & 0 \\ \sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let $v = (x, y, z)^T$, and $u = v/\|v\| = (x', y', z')^T$.

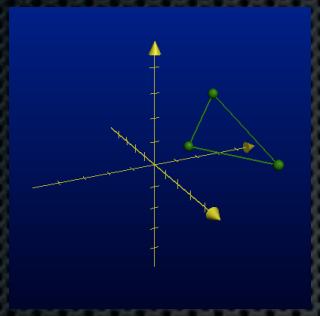
Also let

$$S = \begin{bmatrix} 0 & -z' & y' \\ z' & 0 & -x' \\ -y' & x' & 0 \end{bmatrix} \quad \text{and } M = uu^T + (\cos a)(I - uu^T) + (\sin a)S$$

Then

$$R = \begin{bmatrix} m & m & m & 0 \\ m & m & m & 0 \\ m & m & m & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ where m represents elements from M, which is a 3x3 matrix.

Typical Matrices

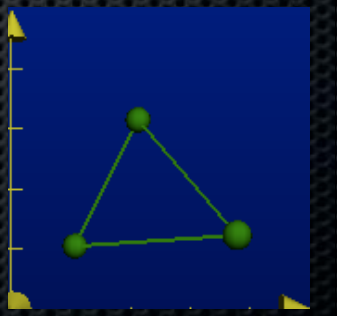


Model View Matrix

Projection Matrix

Homogeneous Divide

Viewport Transform



Model Coordinates

Eye Coordinates

Canonical Coordinates

X: -1→1 Y: -1→1 Z: -1→1

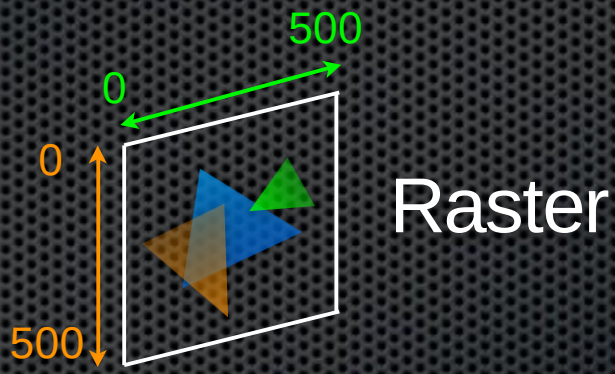
Raster Coordinates

Viewport Transform

X: 0→500

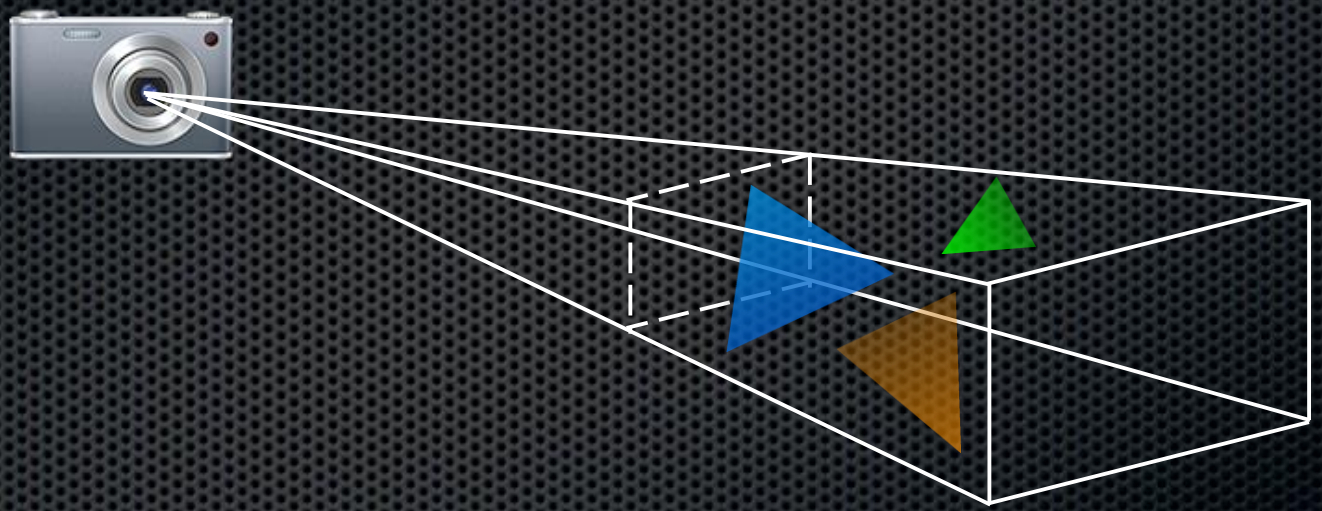
Y: 0→500

Z: 0→0

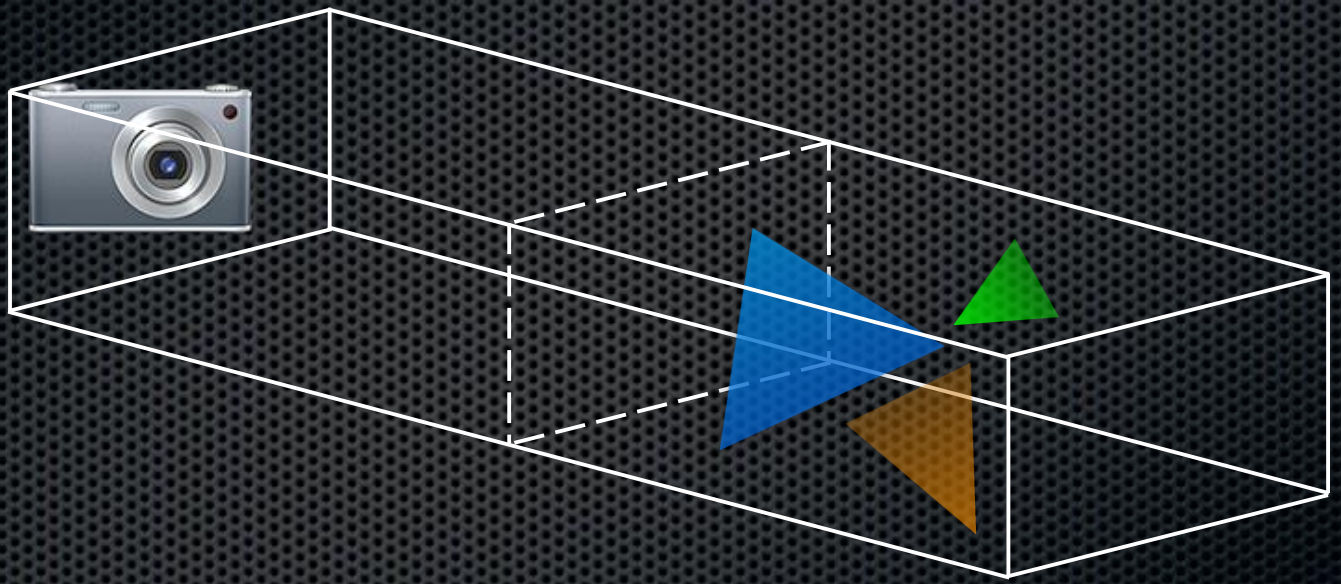


Transformation Process

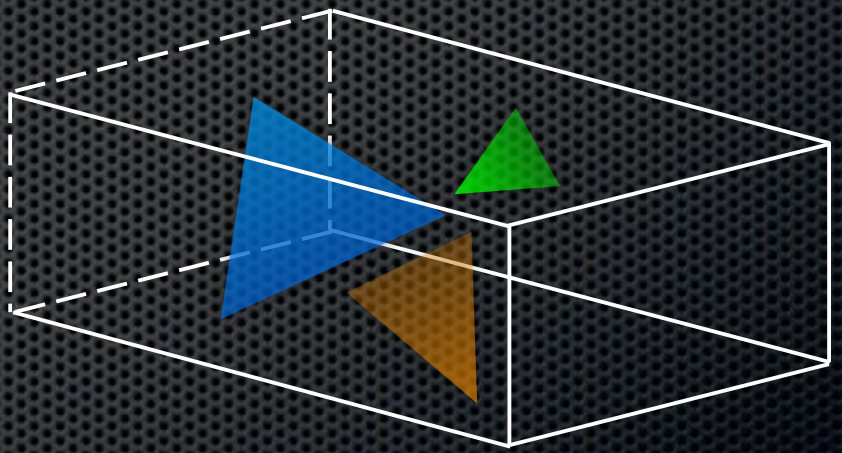
Transformation Process



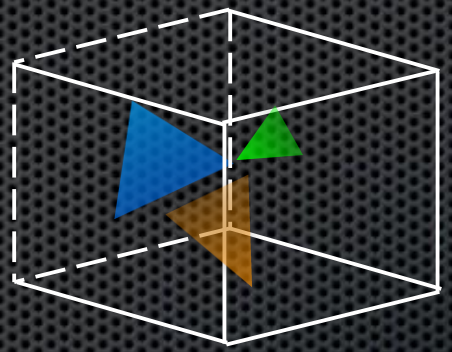
Transformation Process



Transformation Process



Transformation Process



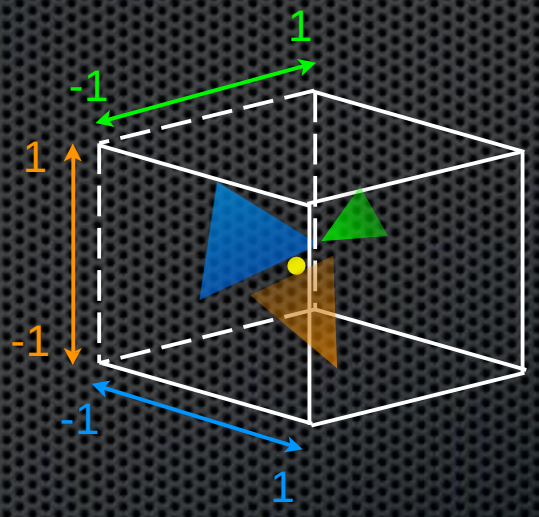
Canonical Coordinates

X: -1→1

Y: -1→1

Z: -1→1

Origin



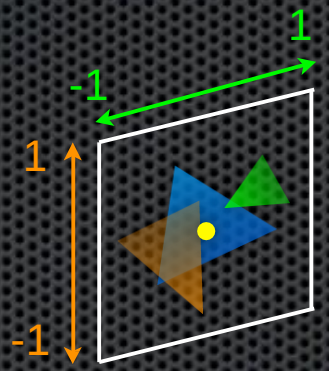
Discard Z

X: -1→1

Y: -1→1

Z: 0→0

Origin



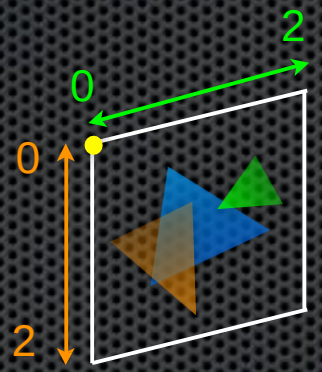
Move Origin

X: 0→2

Y: 0→2

Z: 0→0

Origin



$t_x = 1$

$t_y = 1$

Discard Z

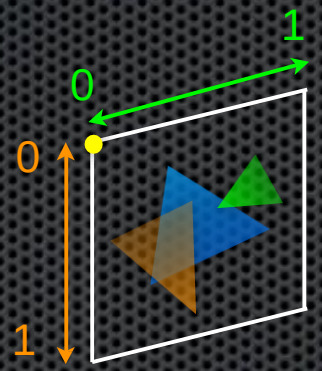
Normalize

X: 0→1

Y: 0→1

Z: 0→0

Origin



$s_x = 1/2$

$s_y = 1/2$

Discard Z

Move Origin

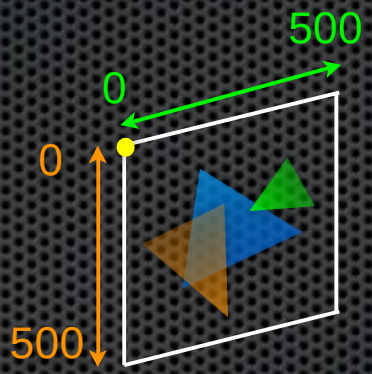
Scale to Viewport Size

X: 0→500

Y: 0→500

Z: 0→0

Origin



$s_x = 500$

$s_y = 500$

Discard Z

Move Origin

Normalize

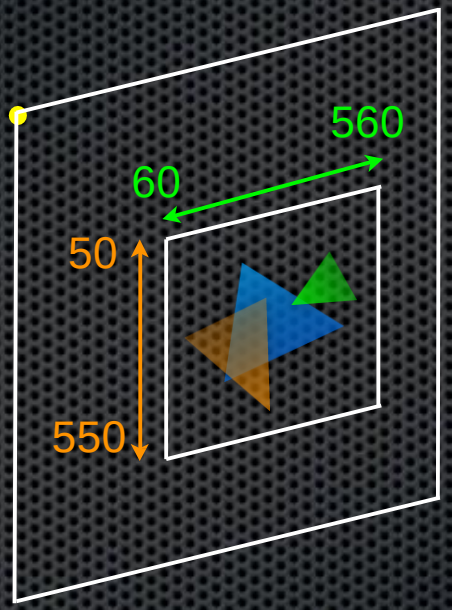
Position on Raster

X: 60→560

Y: 50→550

Z: 0→0

Origin



Raster

$t_x = 60$

$t_y = 50$

Discard Z

Move Origin

Normalize

Scale to Viewport

Viewport Transform

Viewport Transform as a Matrix

Discard Z

Move Origin

Normalize

Scale to Viewport

Position on Raster

$$R = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} + x \\ 0 & \frac{h}{2} & 0 & \frac{h}{2} + y \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewport Transform

Viewport Transform as a Matrix

Discard Z

Move Origin

Normalize

Scale to Viewport

Position on Raster

$$R = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} + x \\ 0 & \frac{h}{2} & 0 & \frac{h}{2} + y \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewport Transform

Viewport Transform as a Matrix

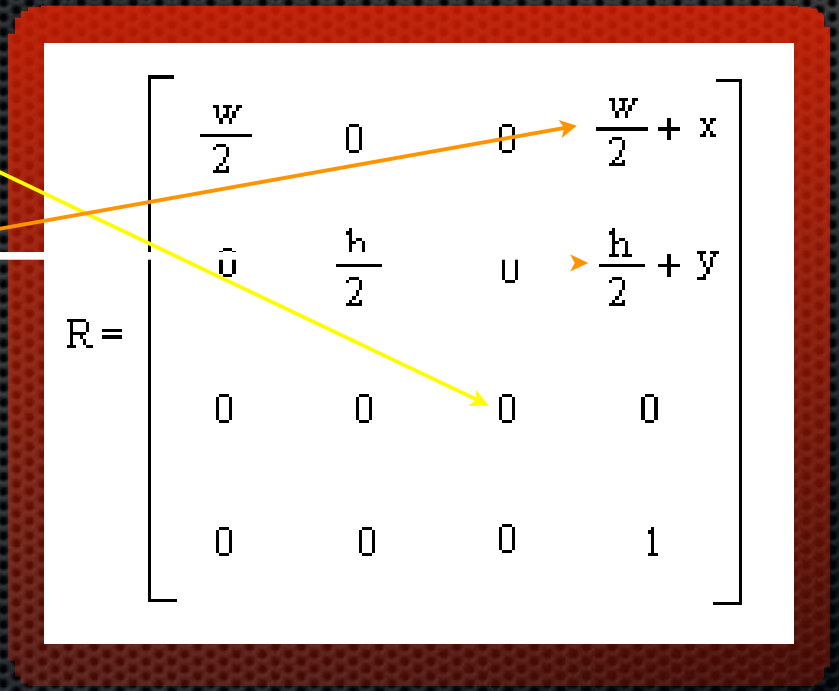
Discard Z

Move Origin

Normalize

Scale to Viewport

Position on Raster



Viewport Transform

Viewport Transform as a Matrix

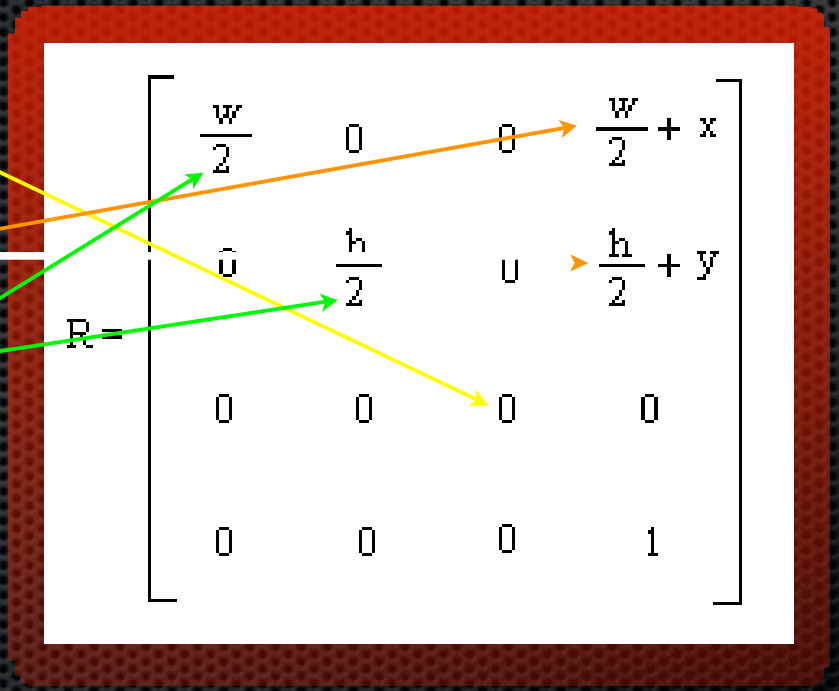
Discard Z

Move Origin

Normalize

Scale to Viewport

Position on Raster



Viewport Transform

Viewport Transform as a Matrix

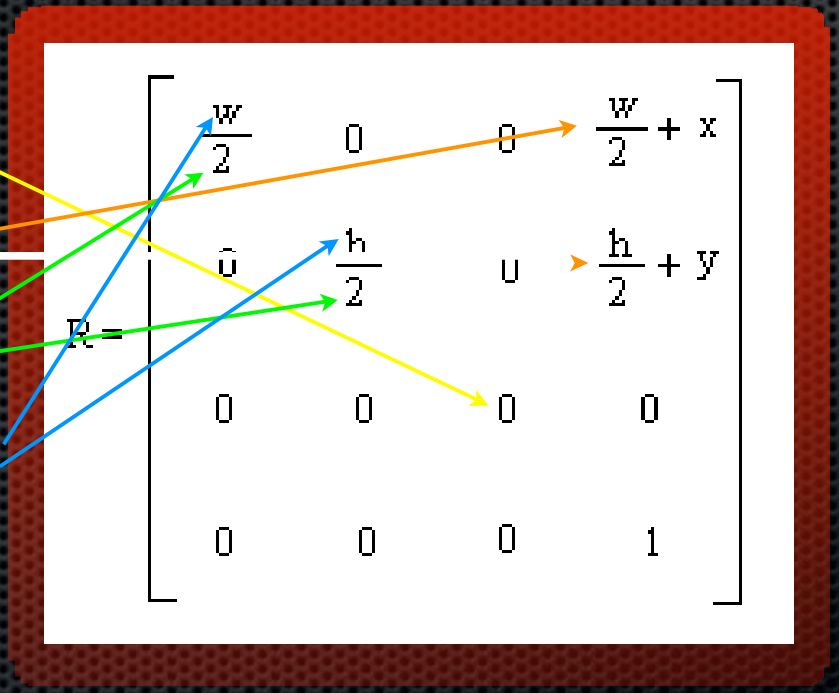
Discard Z

Move Origin

Normalize

Scale to Viewport

Position on Raster



Viewport Transform

Viewport Transform as a Matrix

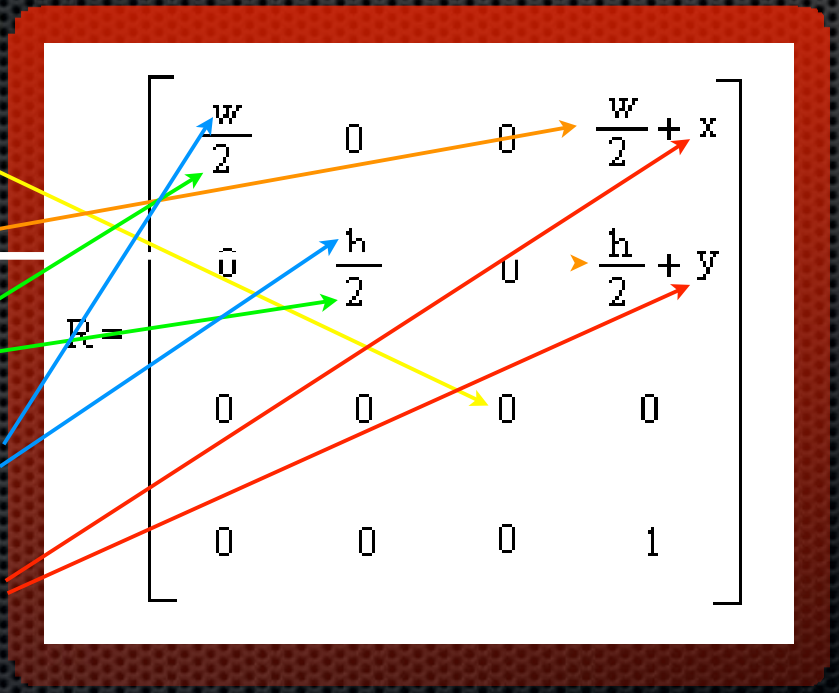
Discard Z

Move Origin

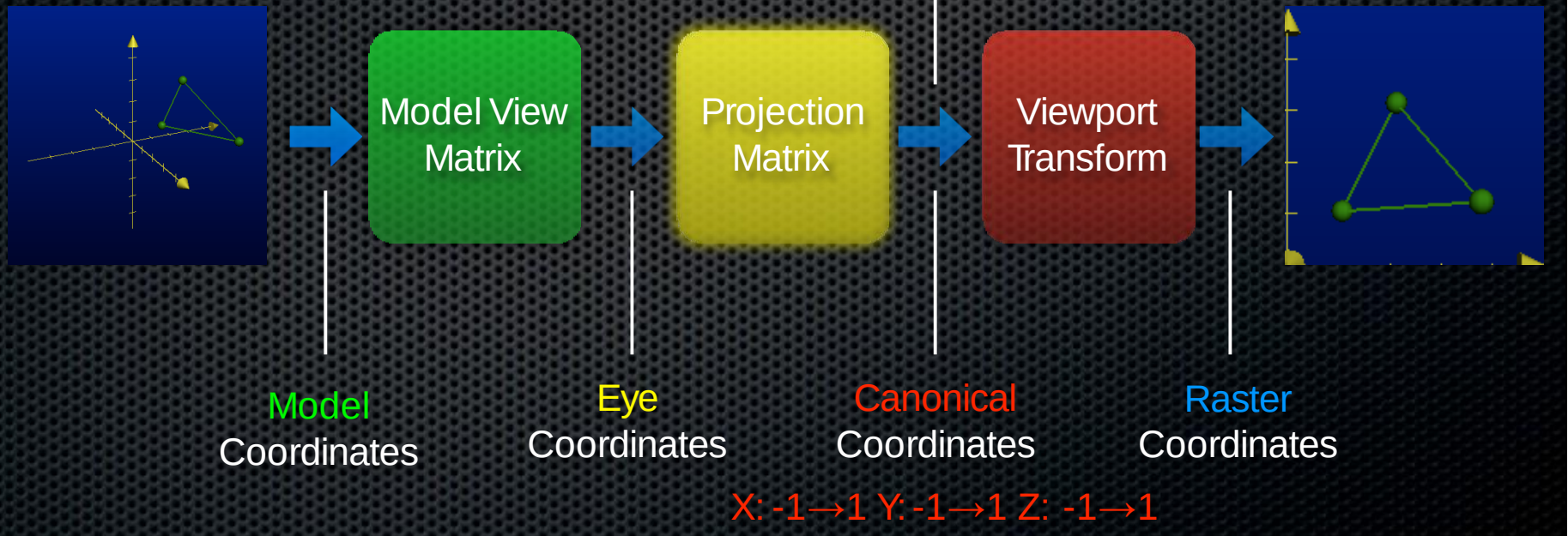
Normalize

Scale to Viewport

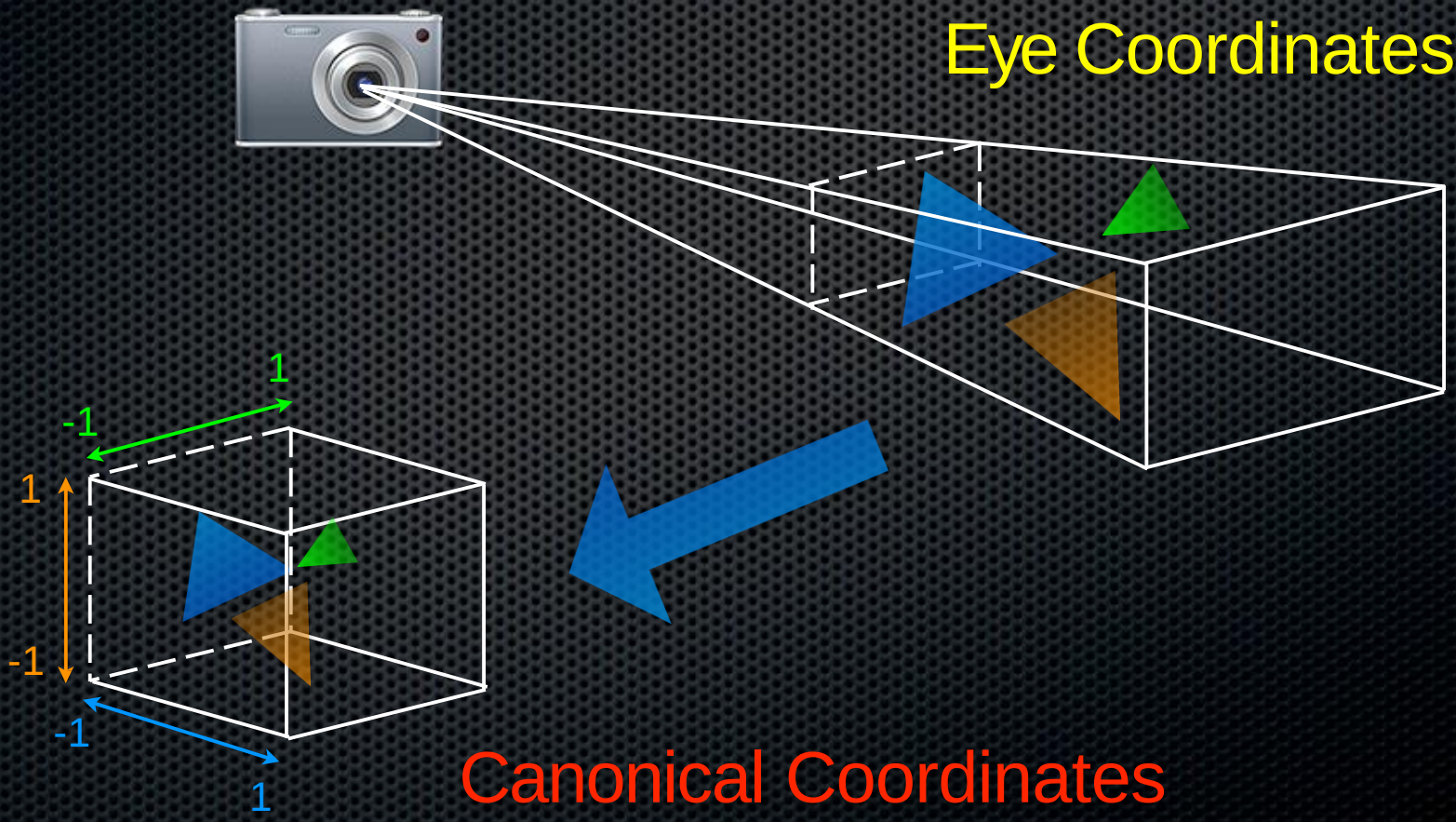
Position on Raster



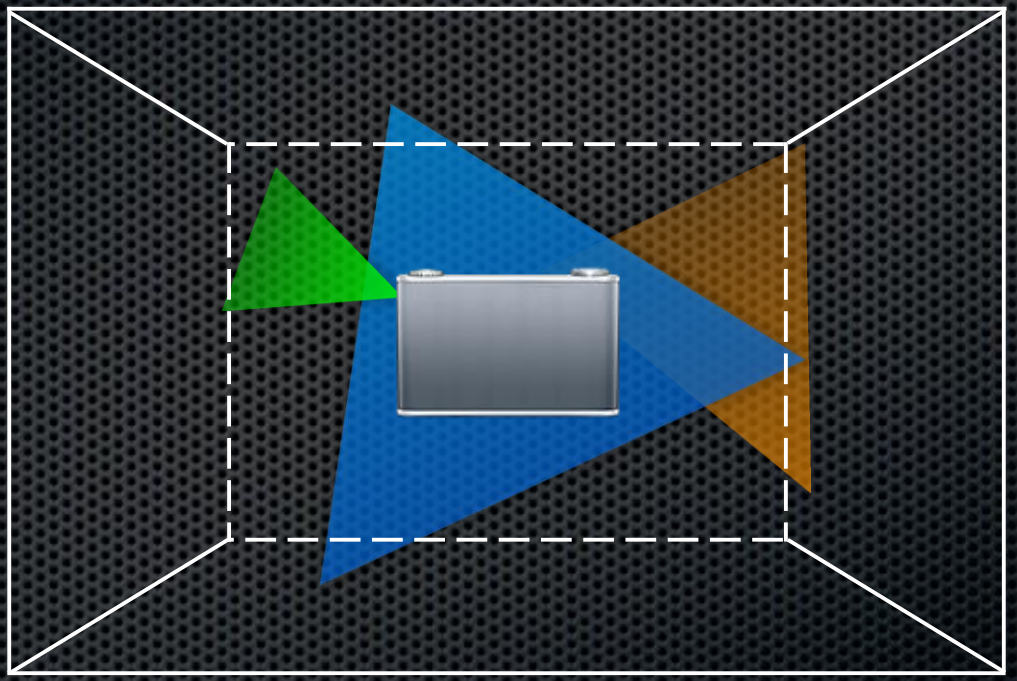
Typical Matrices



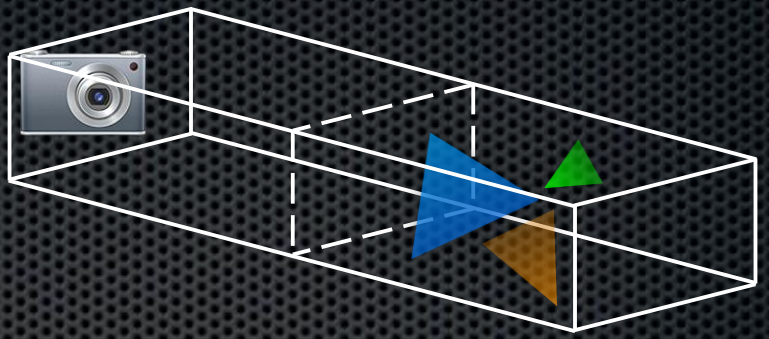
Projection Matrix

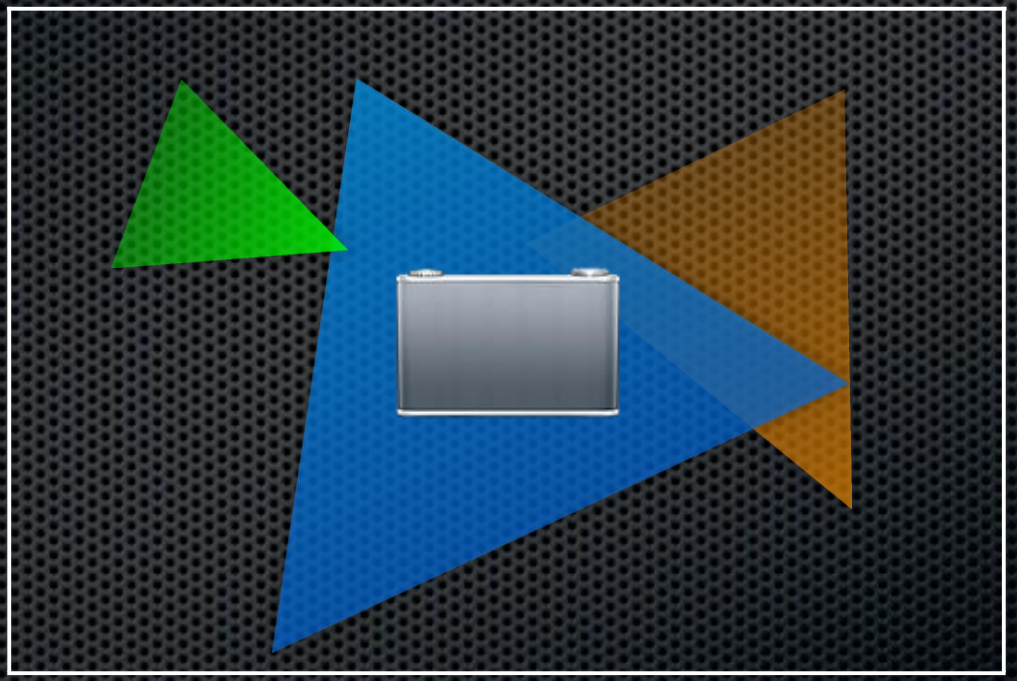


Projection

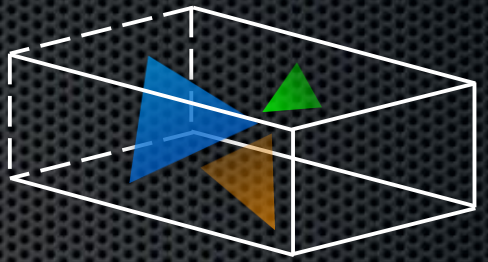


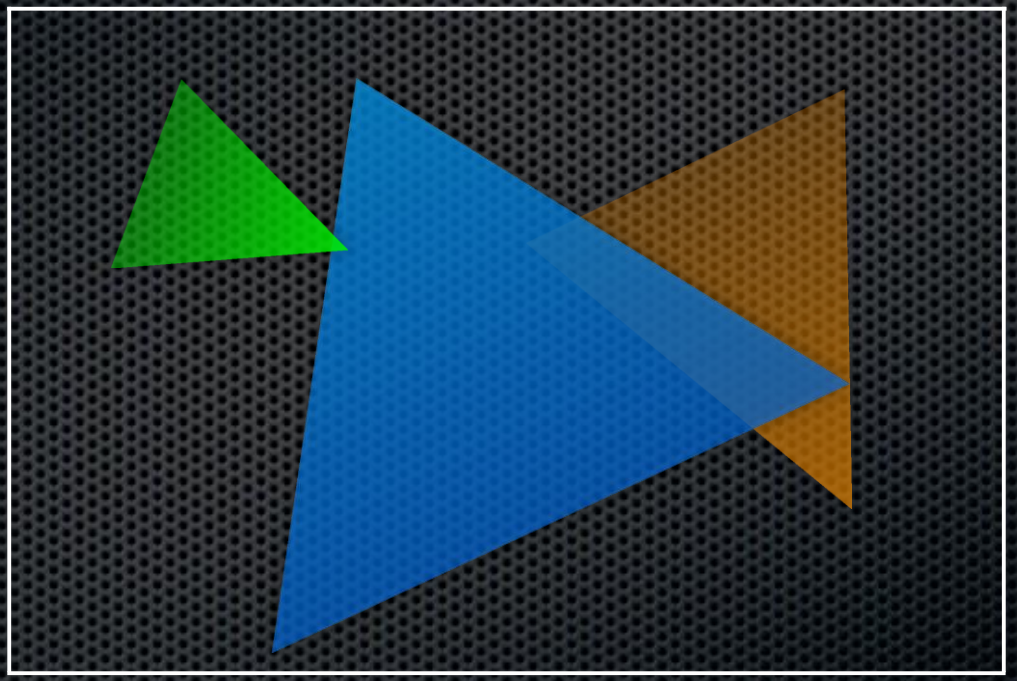
Projection



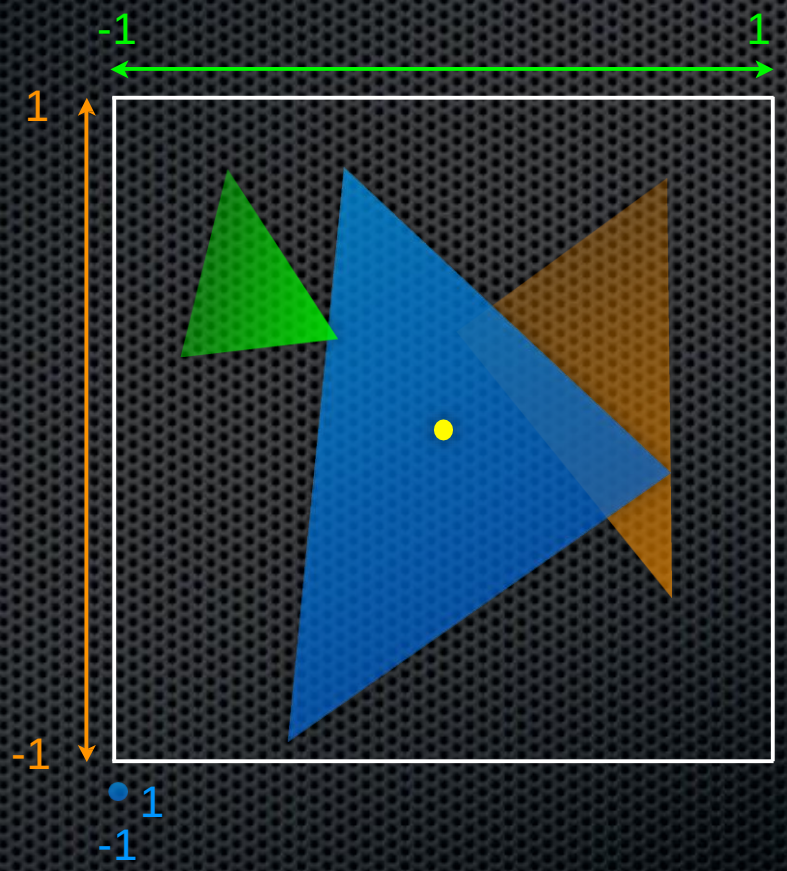


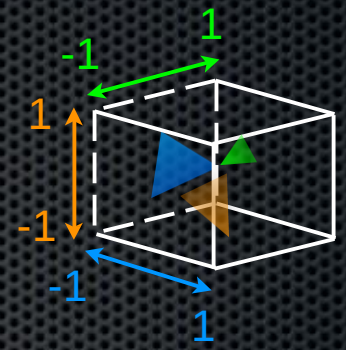
Projection





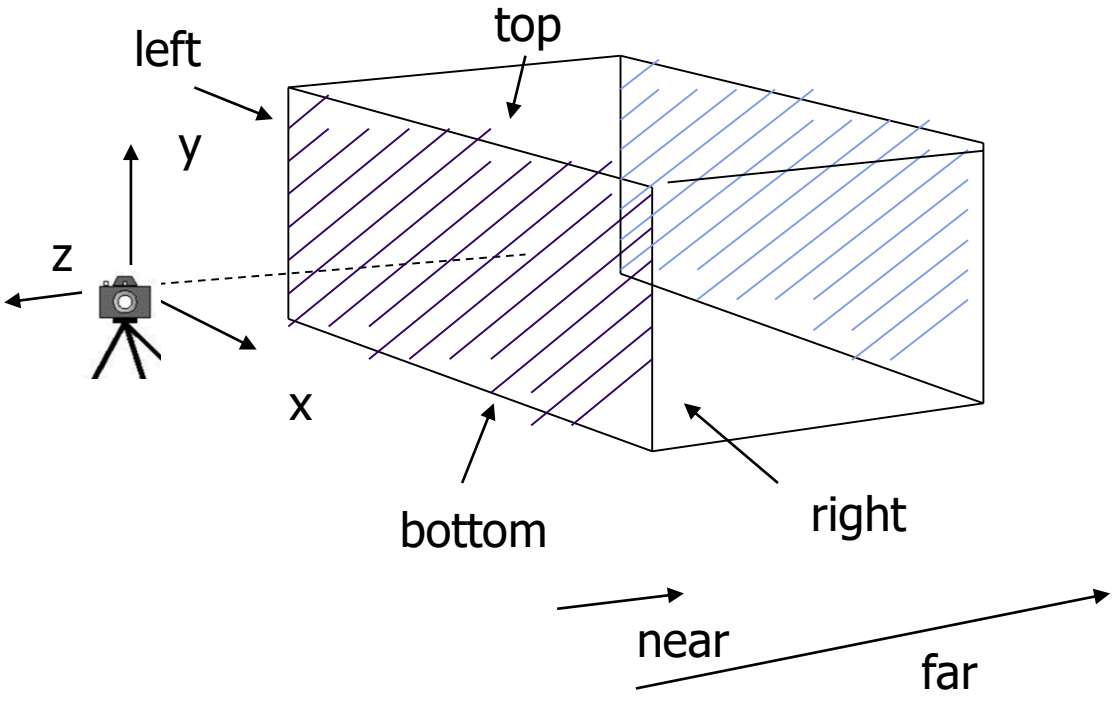
Projection





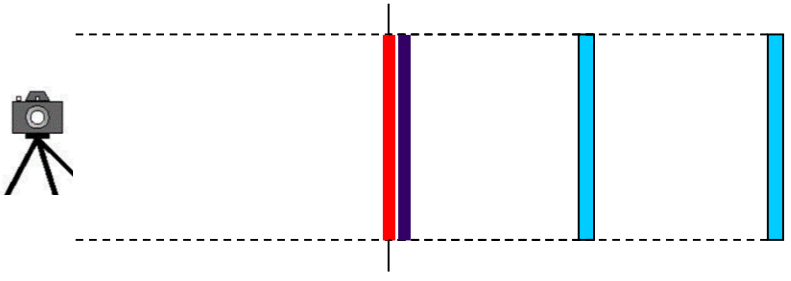
Ortho(left, right, bottom, top, near, far)

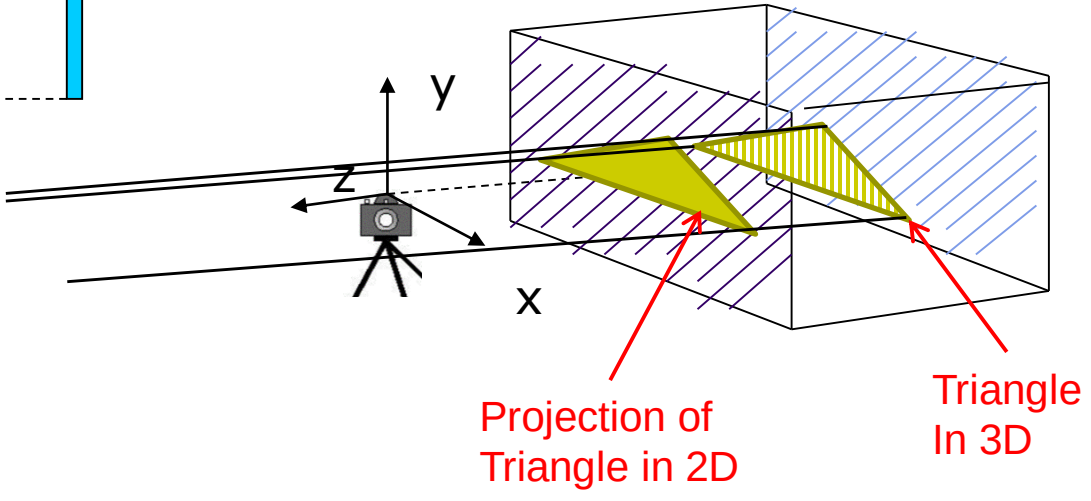
- For orthographic projection



`near` and `far` measured from camera

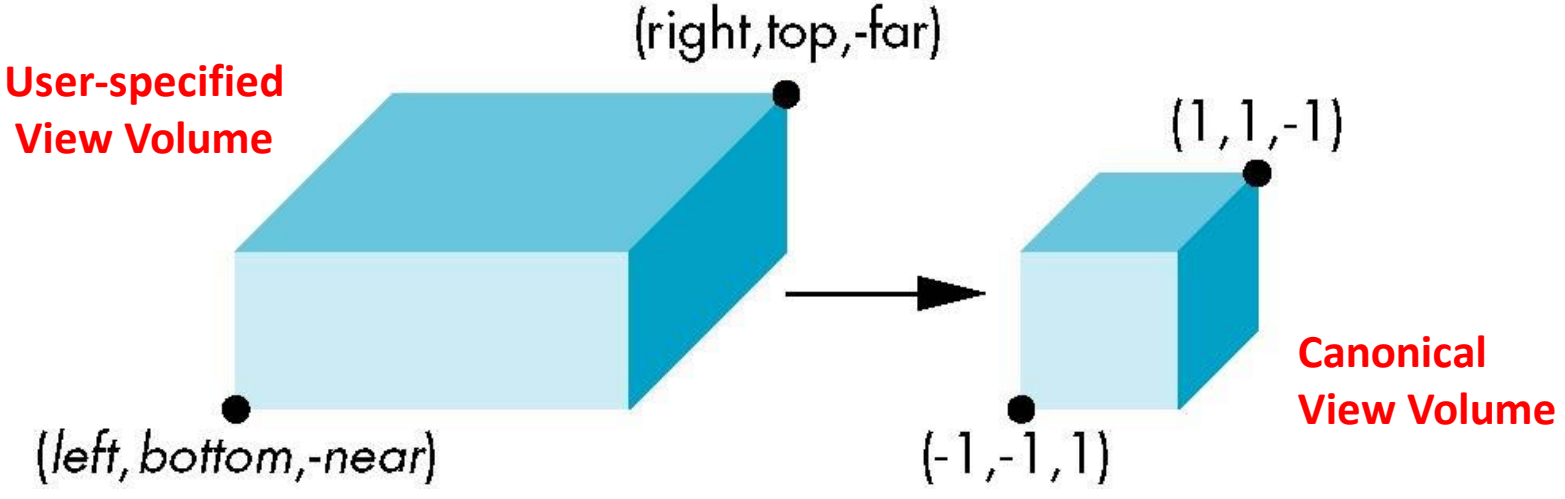
Orthographic Projection





Parallel Projection

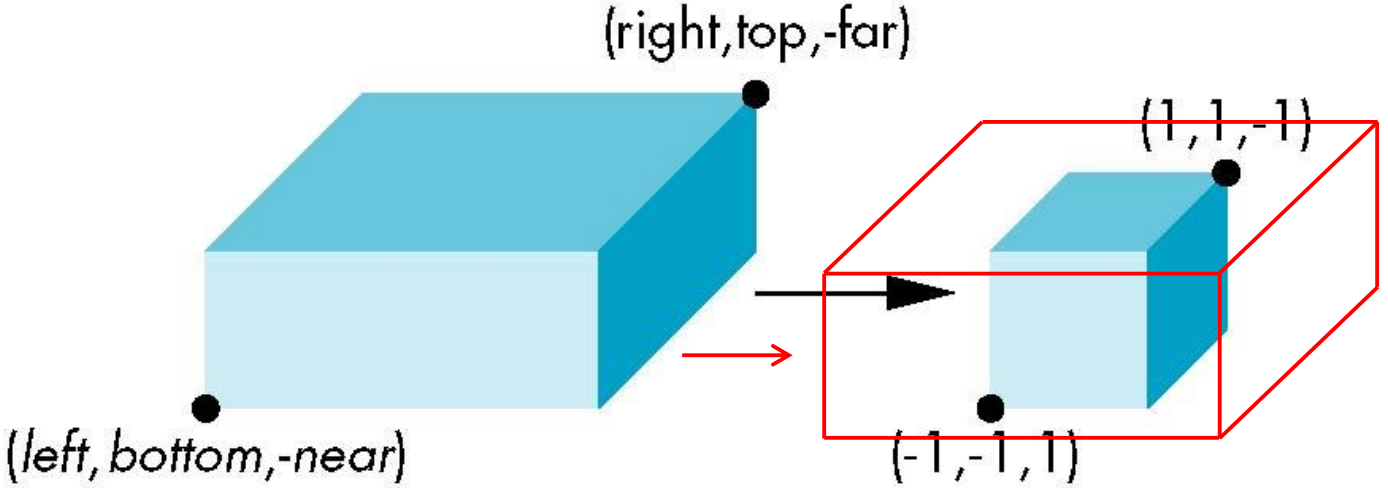
- **normalization** ⇒ find 4x4 matrix to transform **user-specified view volume** to **canonical view volume (cube)**



ortho(left, right, bottom, top,near, far)

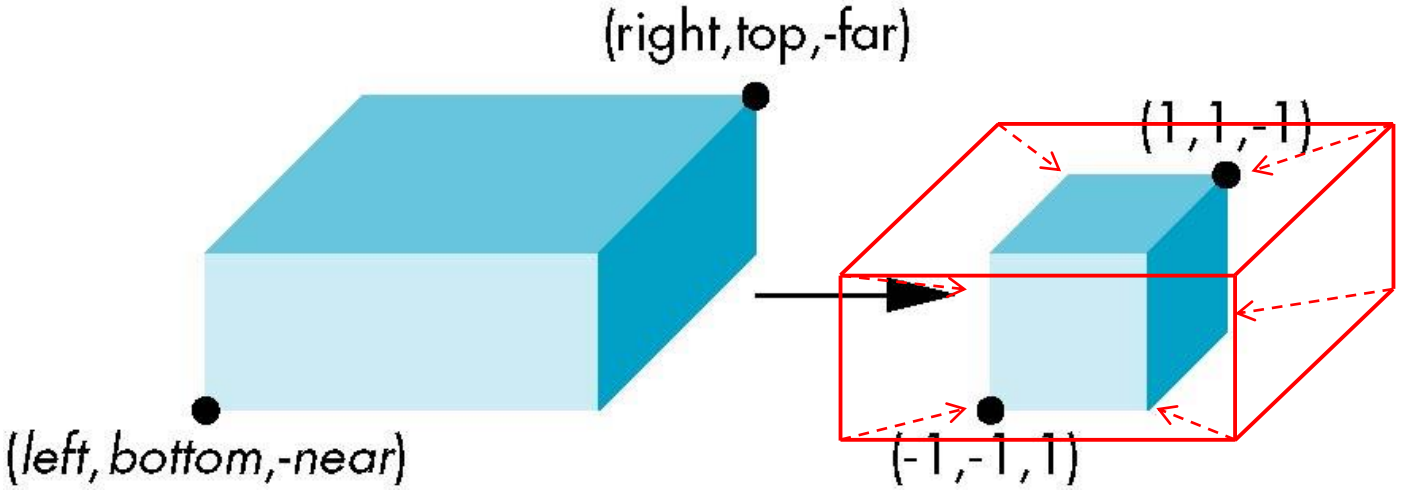
Parallel Projection: Ortho

- Parallel projection: 2 parts
  1. **Translation:** centers view volume at origin



Parallel Projection: Ortho

2. **Scaling:** reduces user-selected cuboid to canonical cube (dimension 2, centered at origin)

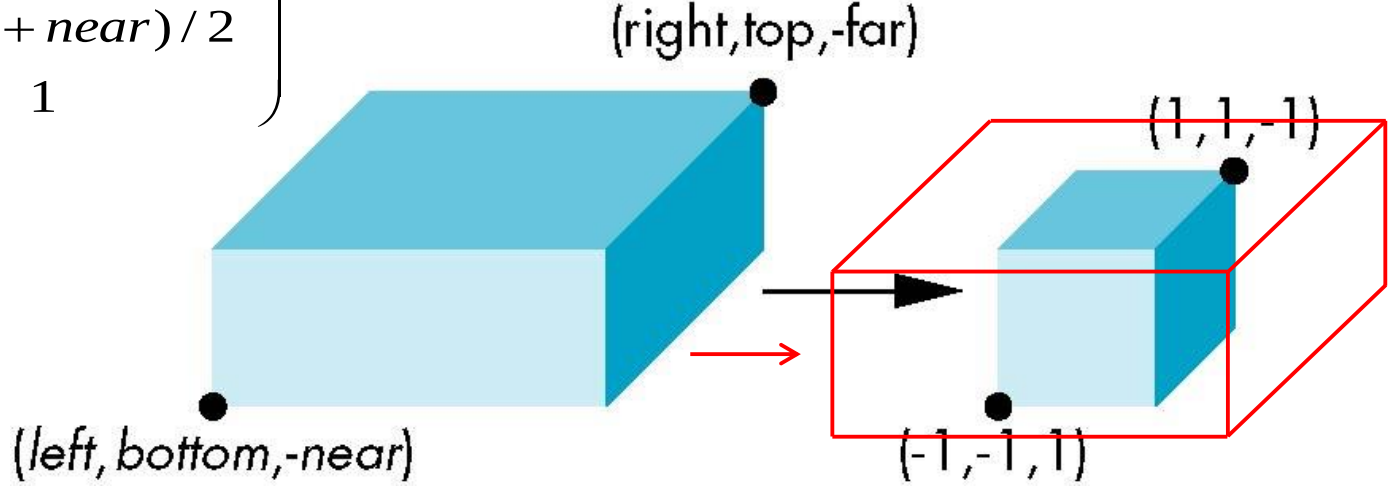


Parallel Projection: Ortho

- Translation lines up midpoints: E.g. midpoint of x = (right + left)/2
- Thus translation factors:

  -(right + left)/2, -(top + bottom)/2, -(far+near)/2
- Translation matrix:

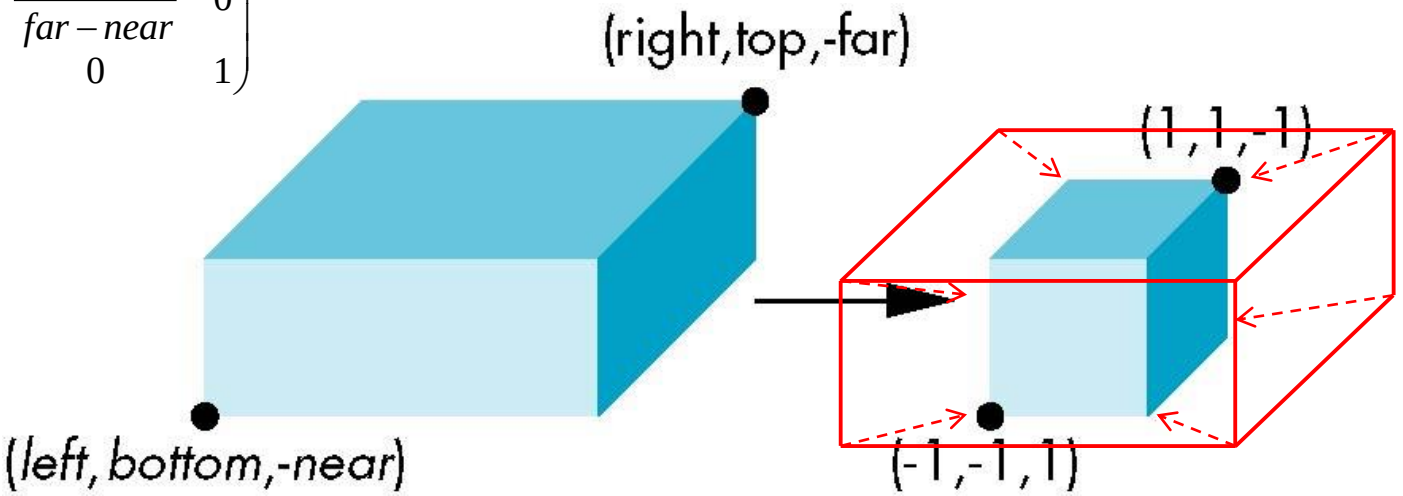
$$\begin{pmatrix} 1 & 0 & 0 & -(right+left)/2 \\ 0 & 1 & 0 & -(top+bottom)/2 \\ 0 & 0 & 1 & -(far+near)/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Parallel Projection: Ortho

- Scaling factor: ratio of ortho view volume to cube dimensions
- Scaling factors: 2/(right - left), 2/(top - bottom), 2/(far - near)
- Scaling Matrix M2:

$$\begin{pmatrix} \frac{2}{right-left} & 0 & 0 & 0 \\ 0 & \frac{2}{top-bottom} & 0 & 0 \\ 0 & 0 & \frac{2}{far-near} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Parallel Projection: Ortho

Concatenating **Translation** x **Scaling**, we get Ortho Projection matrix

$$\begin{pmatrix} \frac{2}{right-left} & 0 & 0 & 0 \\ 0 & \frac{2}{top-bottom} & 0 & 0 \\ 0 & 0 & \frac{2}{far-near} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} 1 & 0 & 0 & -(right+left)/2 \\ 0 & 1 & 0 & -(top+bottom)/2 \\ 0 & 0 & 1 & -(far+near)/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

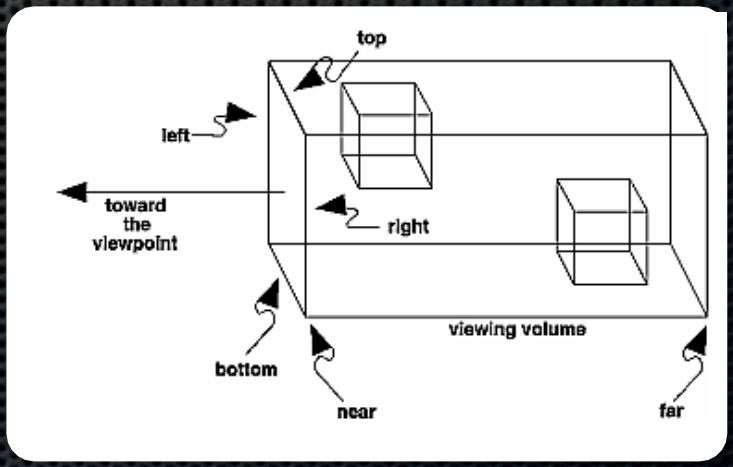
$$\mathbf{P = ST =} \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right-left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{near-far} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection - Orthographic

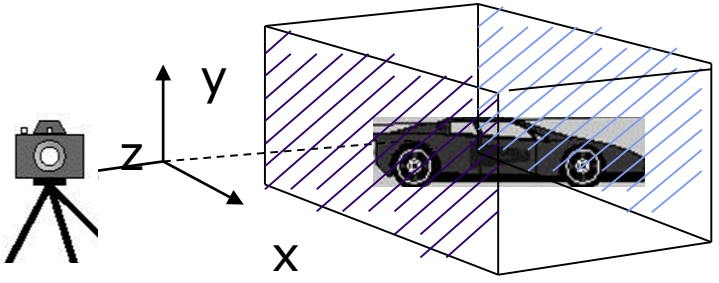
Combination Scale & Translation

ortho(l,r,b,t,n,f)

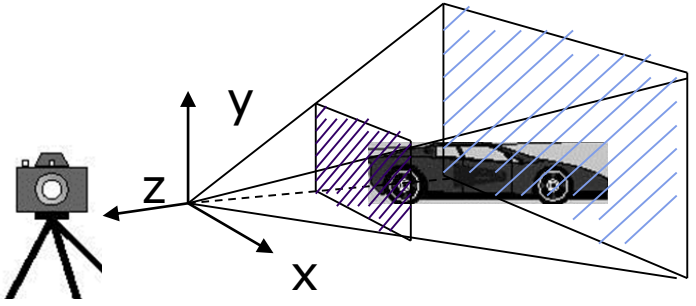
$$R = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Different View Volume Shapes



Orthogonal view volume

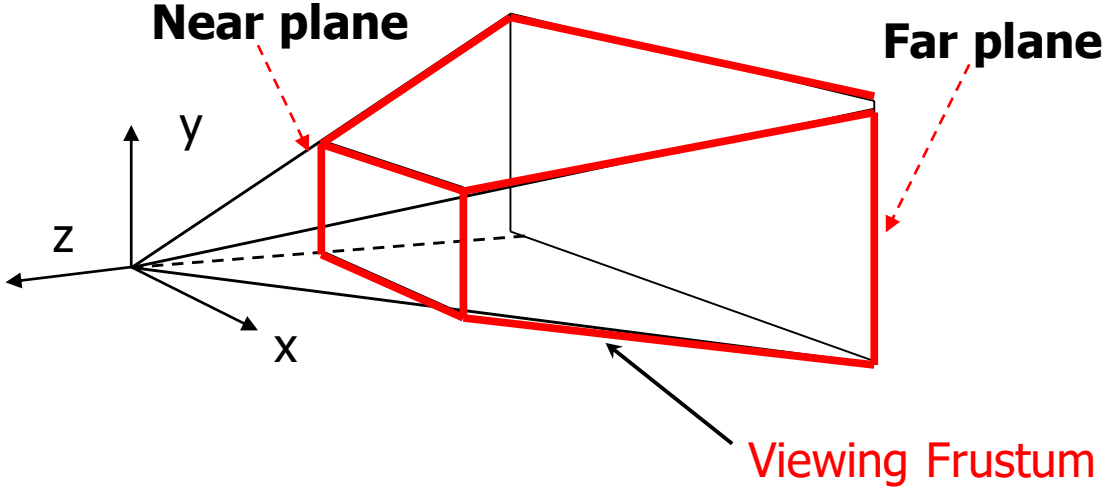


Perspective view volume

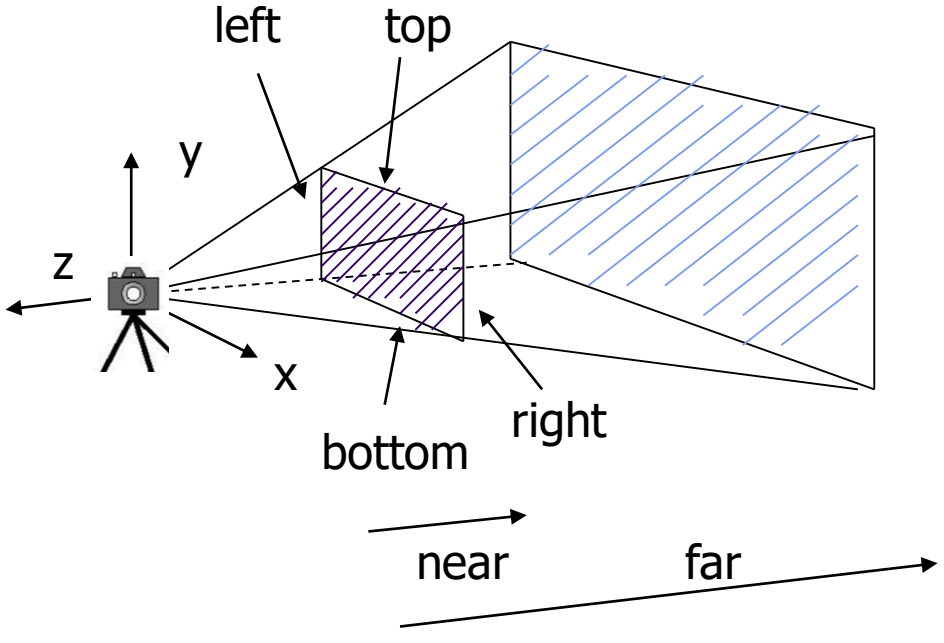
- Different view volume => different look
- **Foreshortening?** Near objects bigger
  - Perpective projection has **foreshortening**
  - Orthogonal projection: no foreshortening

Viewing Frustrum

- Near plane + far plane + field of view = **Viewing Frustum**
- Objects outside the frustum are clipped

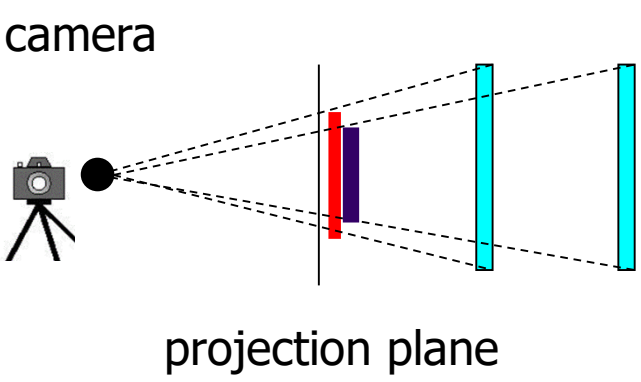


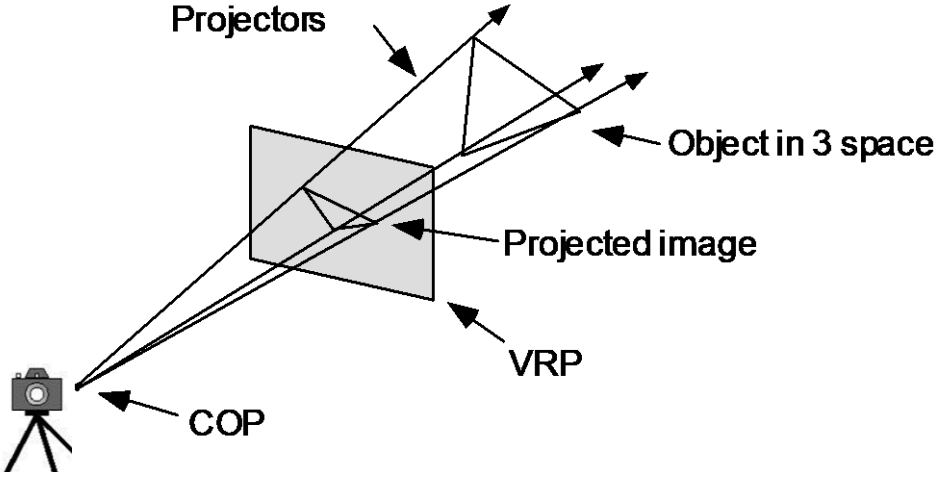
Frustum(left, right, bottom, top, near, far)



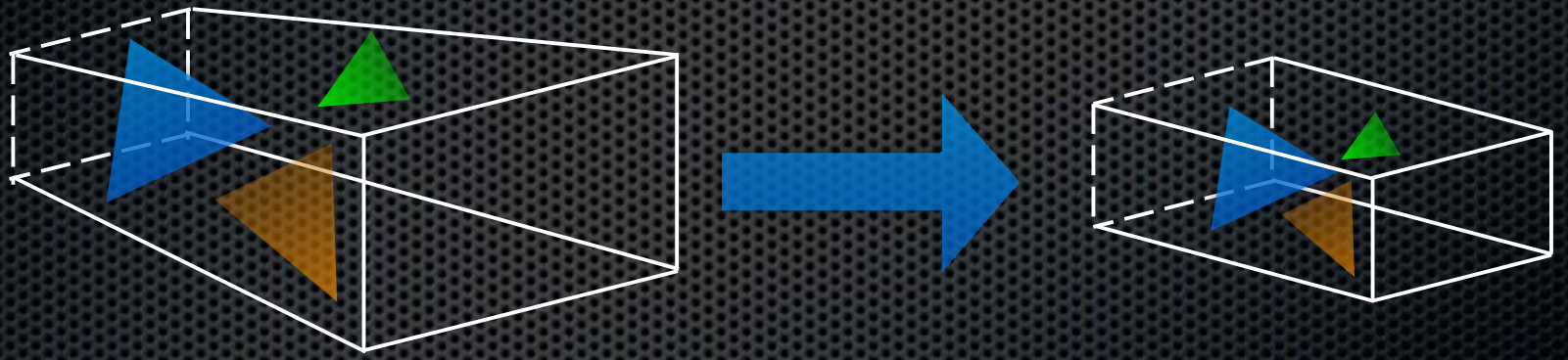
`near` and `far` measured from camera

Perspective Projection





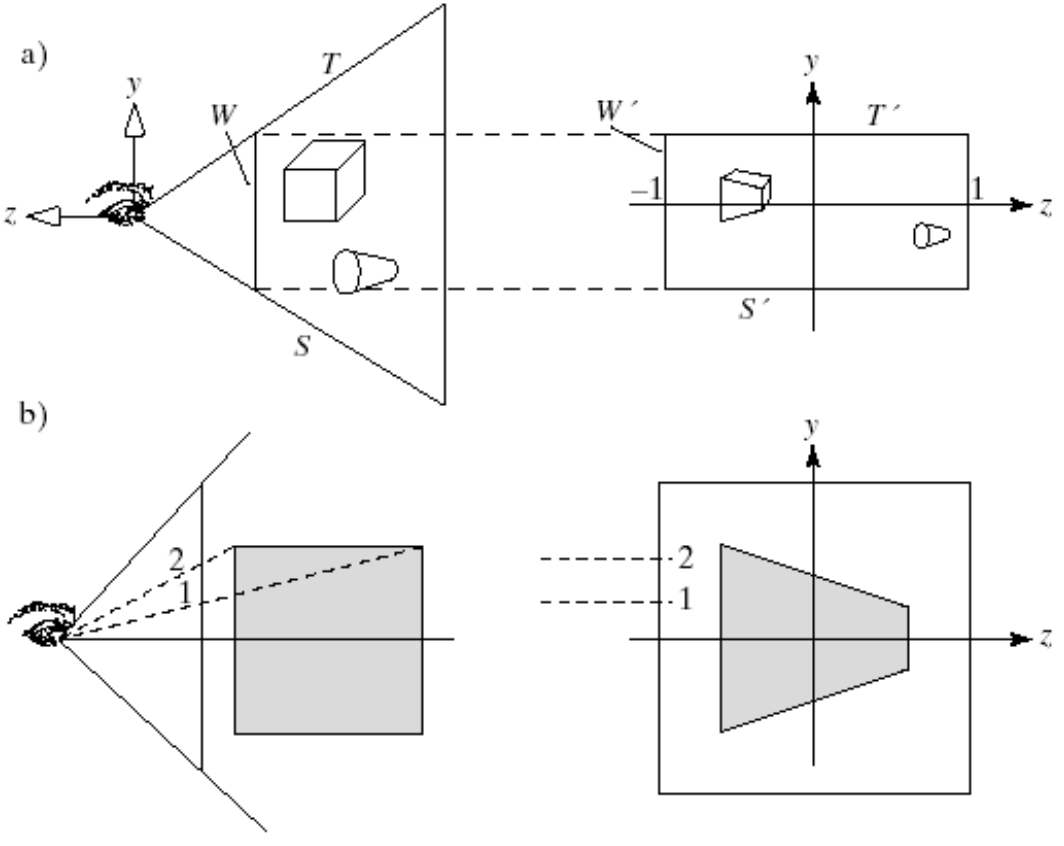
Projection - Perspective



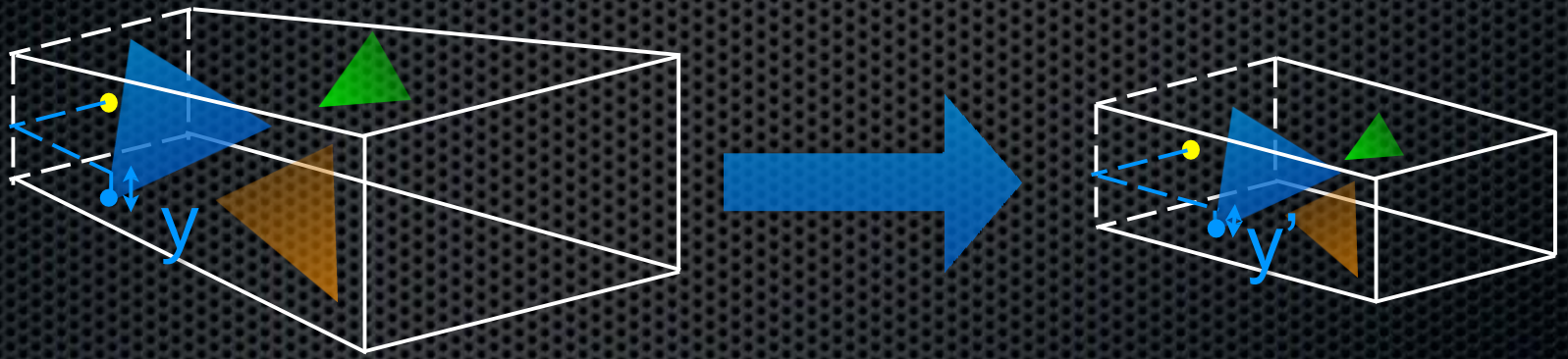
Geometric Nature of Perspective Transform

a) Lines through eye map into lines parallel to z axis after transform

b) Lines perpendicular to z axis map to lines perp to z axis after transform

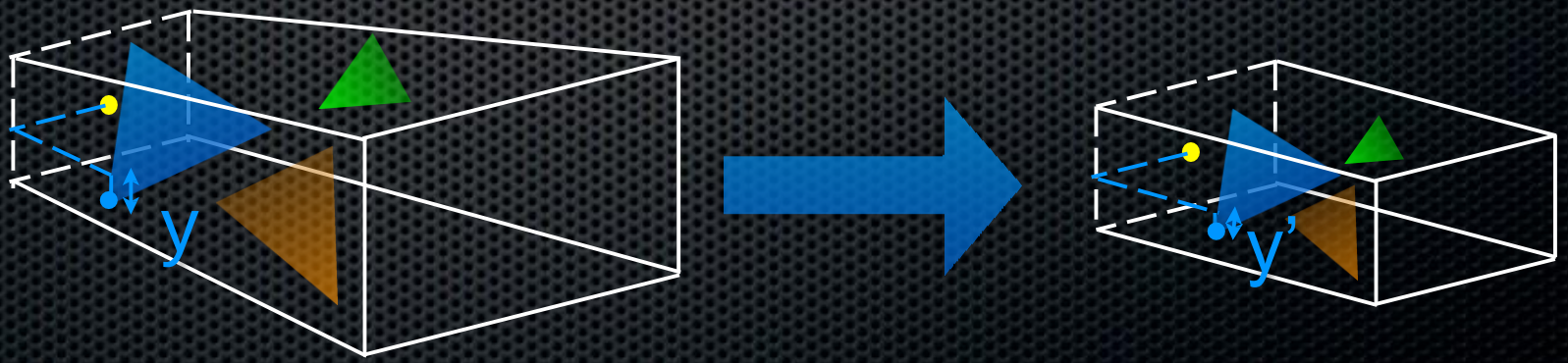


Projection - Perspective

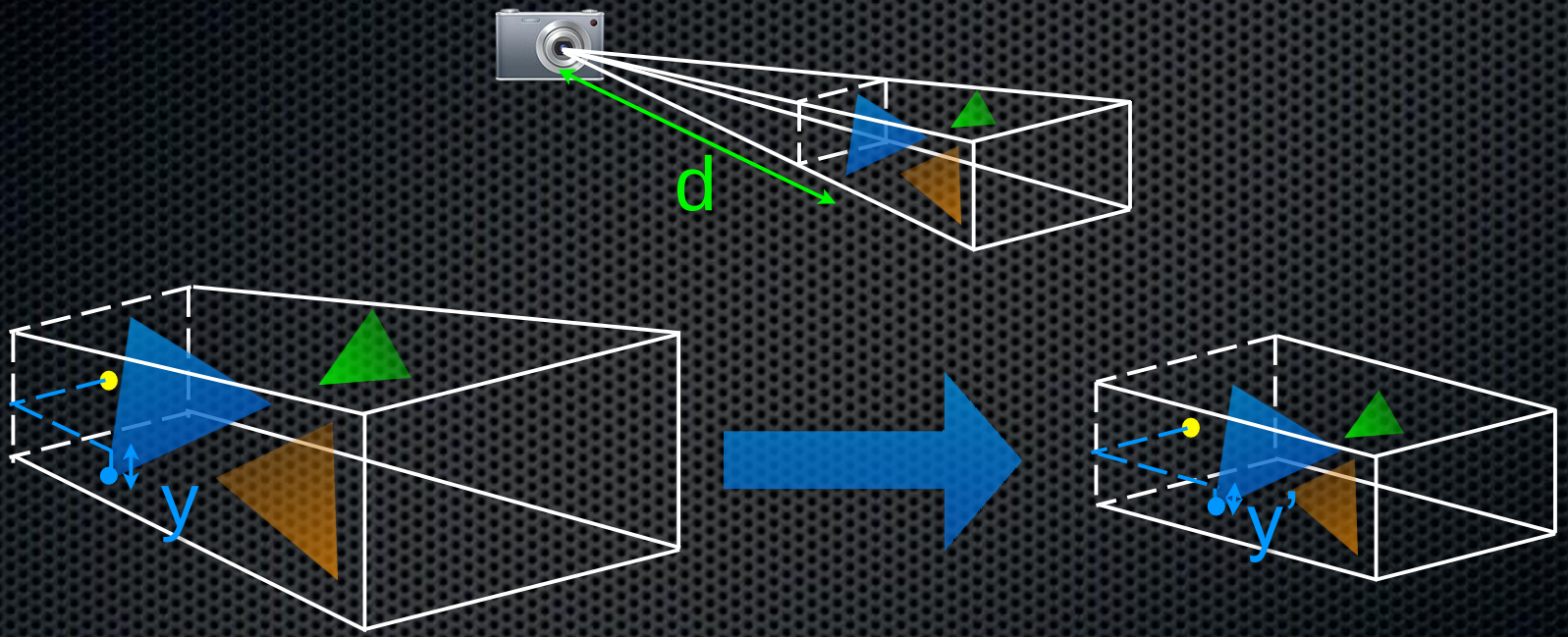


Think-Pair-Share

How is y’ related to z? What is x’? How does this correspond to how we view the world?

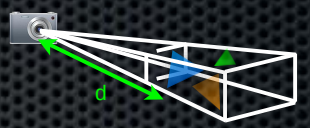


Projection - Perspective



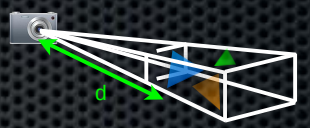
$$y' = \frac{y}{d}, \; d > 1$$

Projection - Perspective



$$y' = \frac{y}{d}$$

Projection - Perspective



In eye coordinates, d = z

$$y' = \frac{y}{d}$$

Projection - Perspective

In eye coordinates, d = z

$$y' = \frac{y}{z}$$

Projection - Perspective

$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

$$\begin{bmatrix} ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ z \\ 1 \end{bmatrix}$$

Projection - Perspective

$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

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

Think-Pair-Share

Fill in the top two rows of this matrix.

$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

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

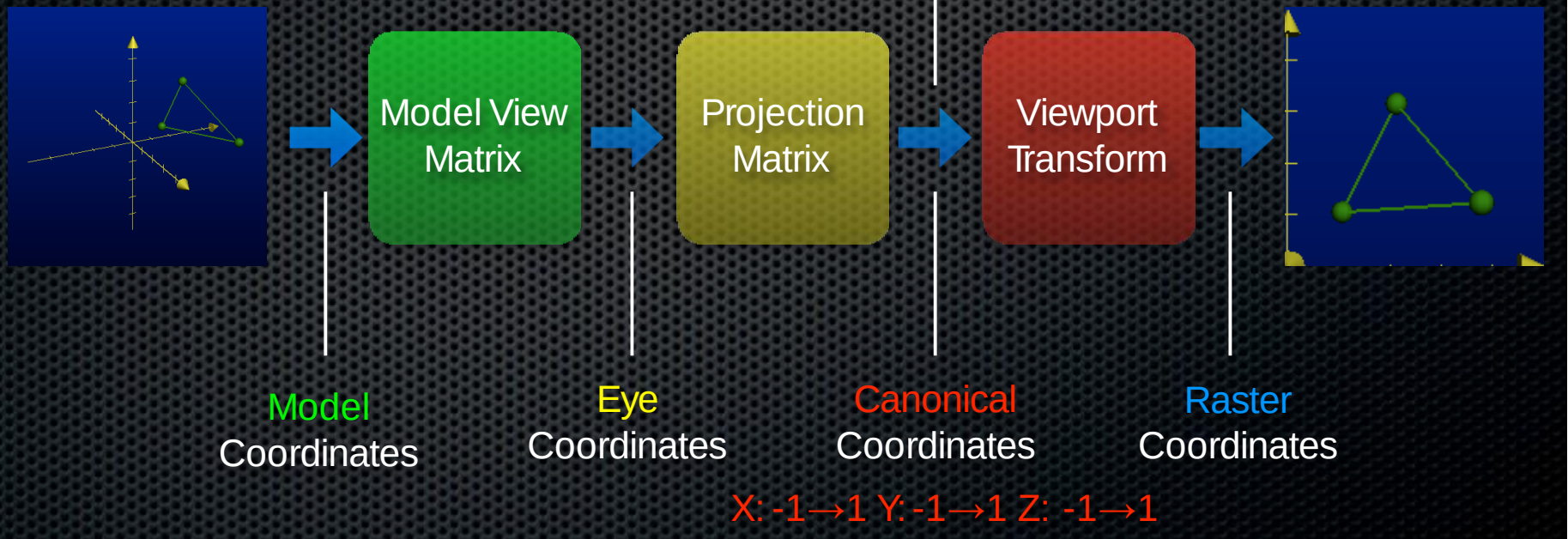
Projection - Perspective

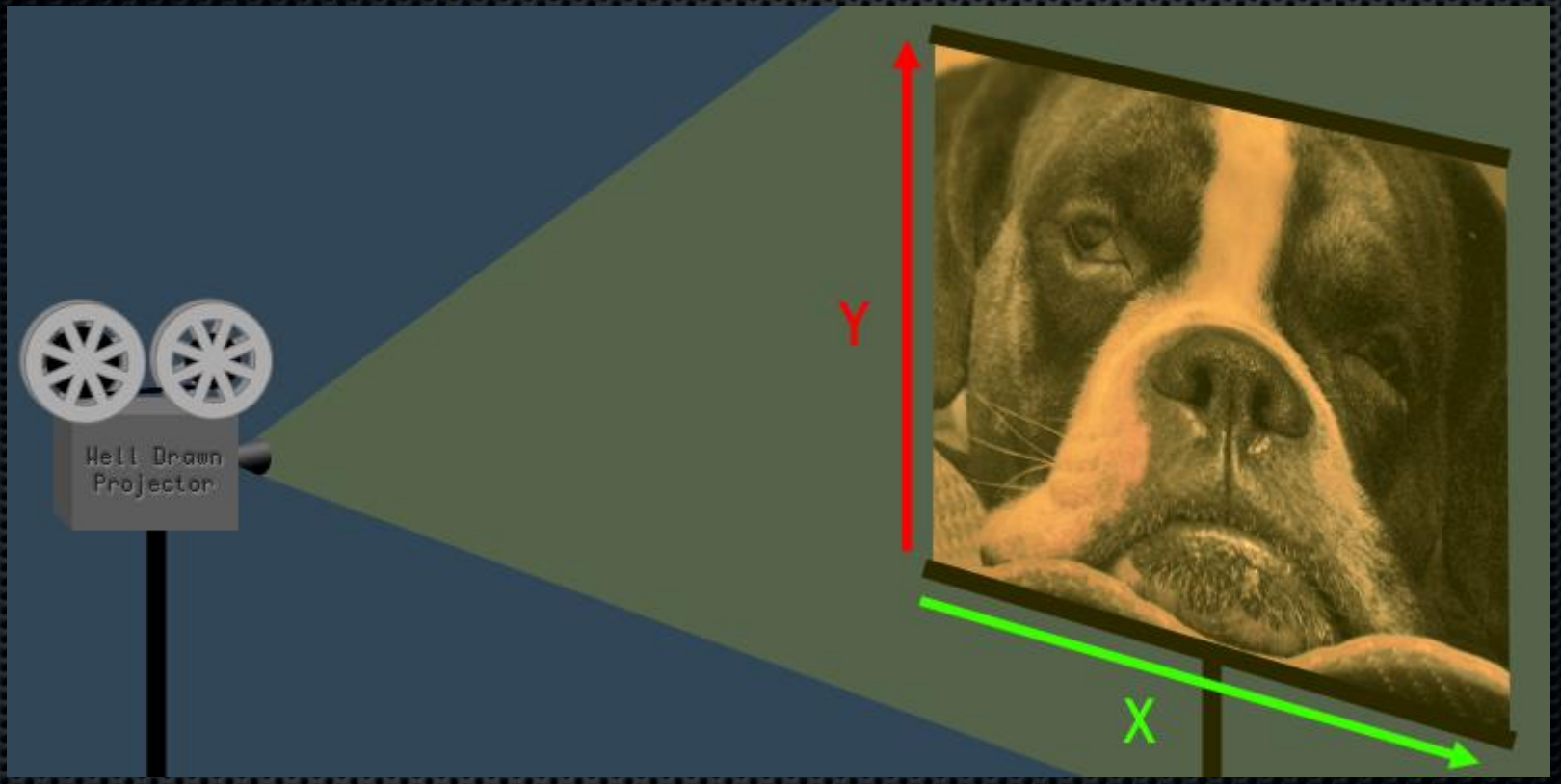
$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

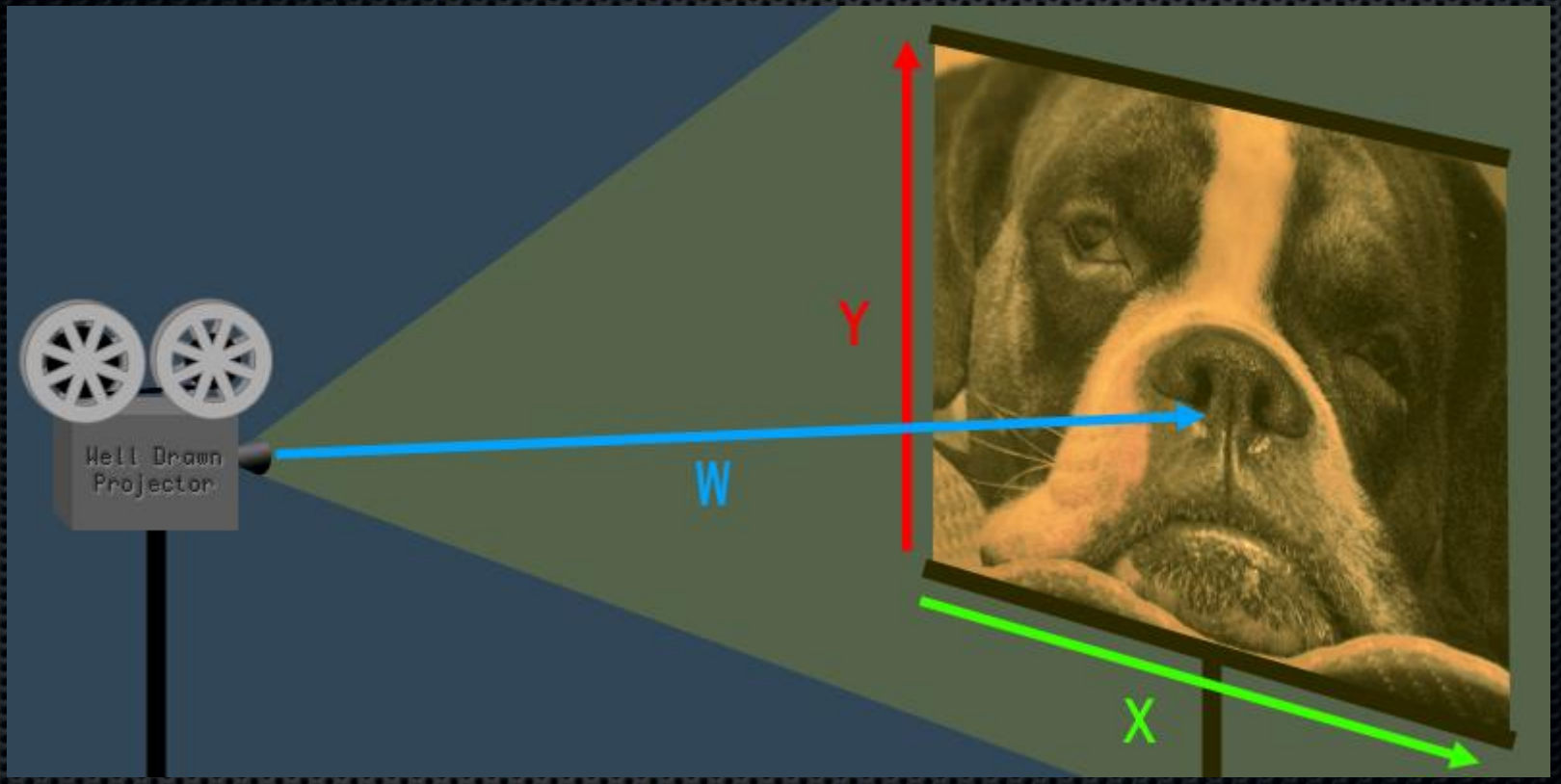
$$\begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ z \\ 1 \end{bmatrix}$$

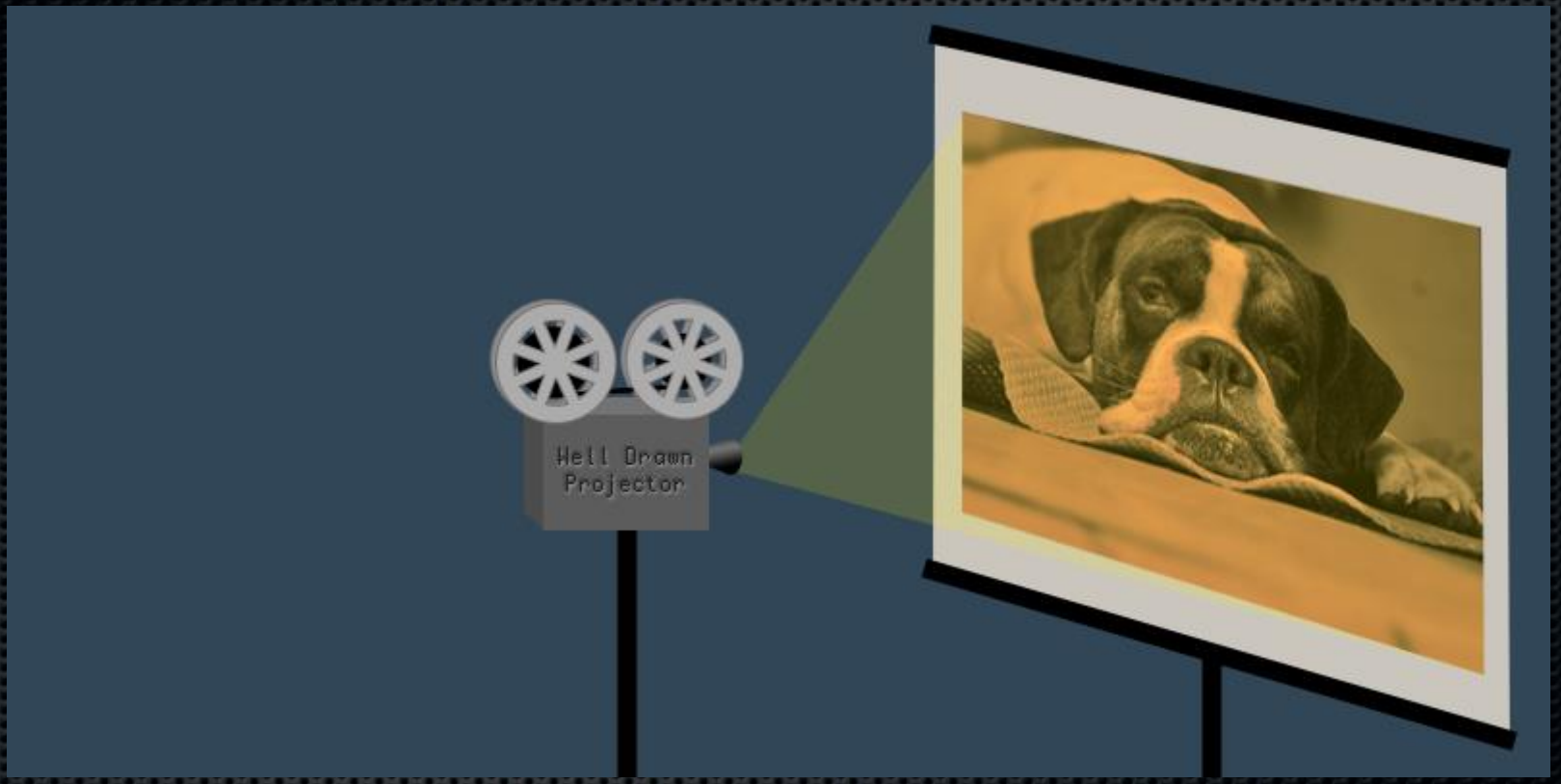
No possible matrix, because we don't have z!

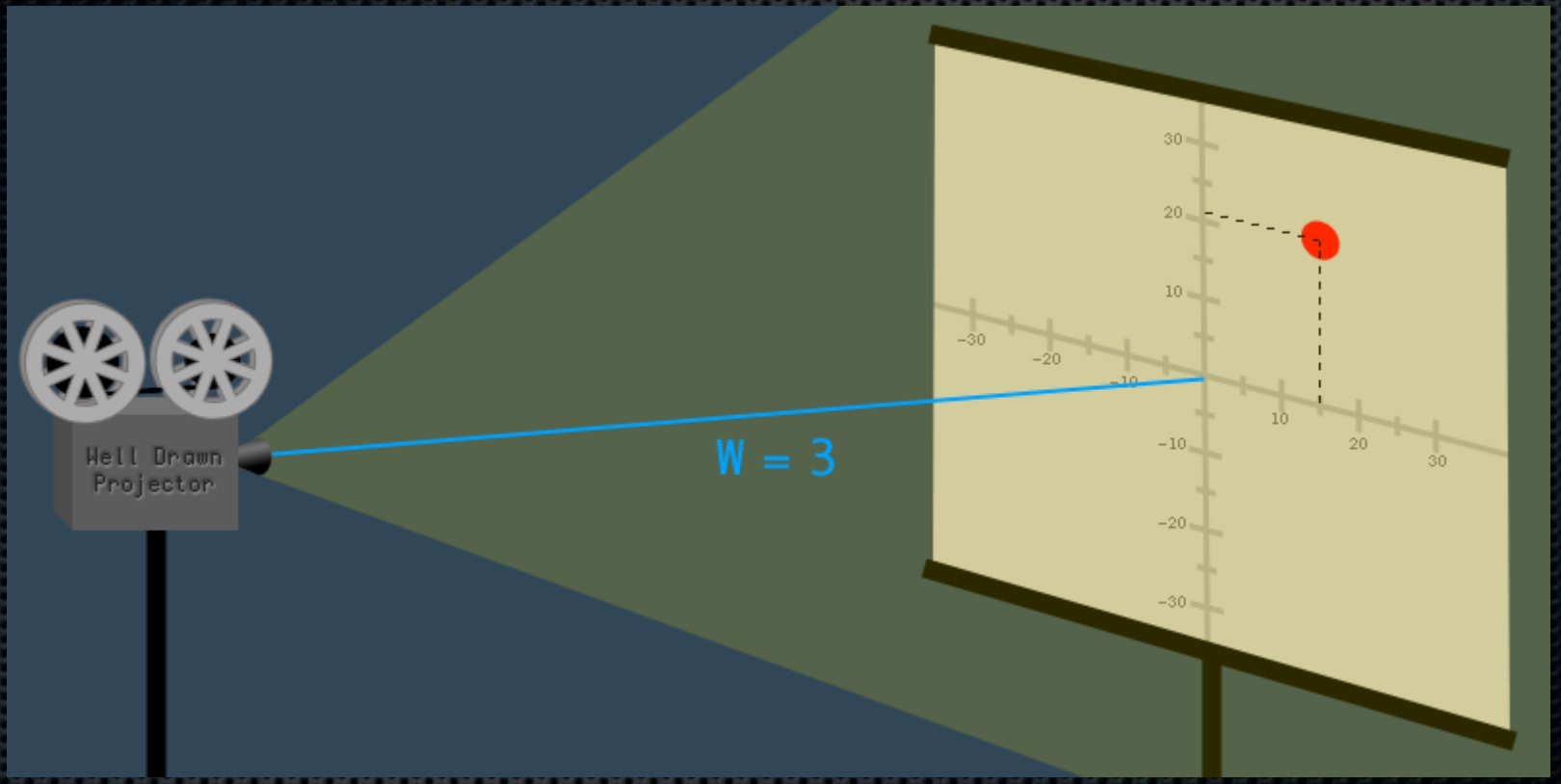
Typical Matrices

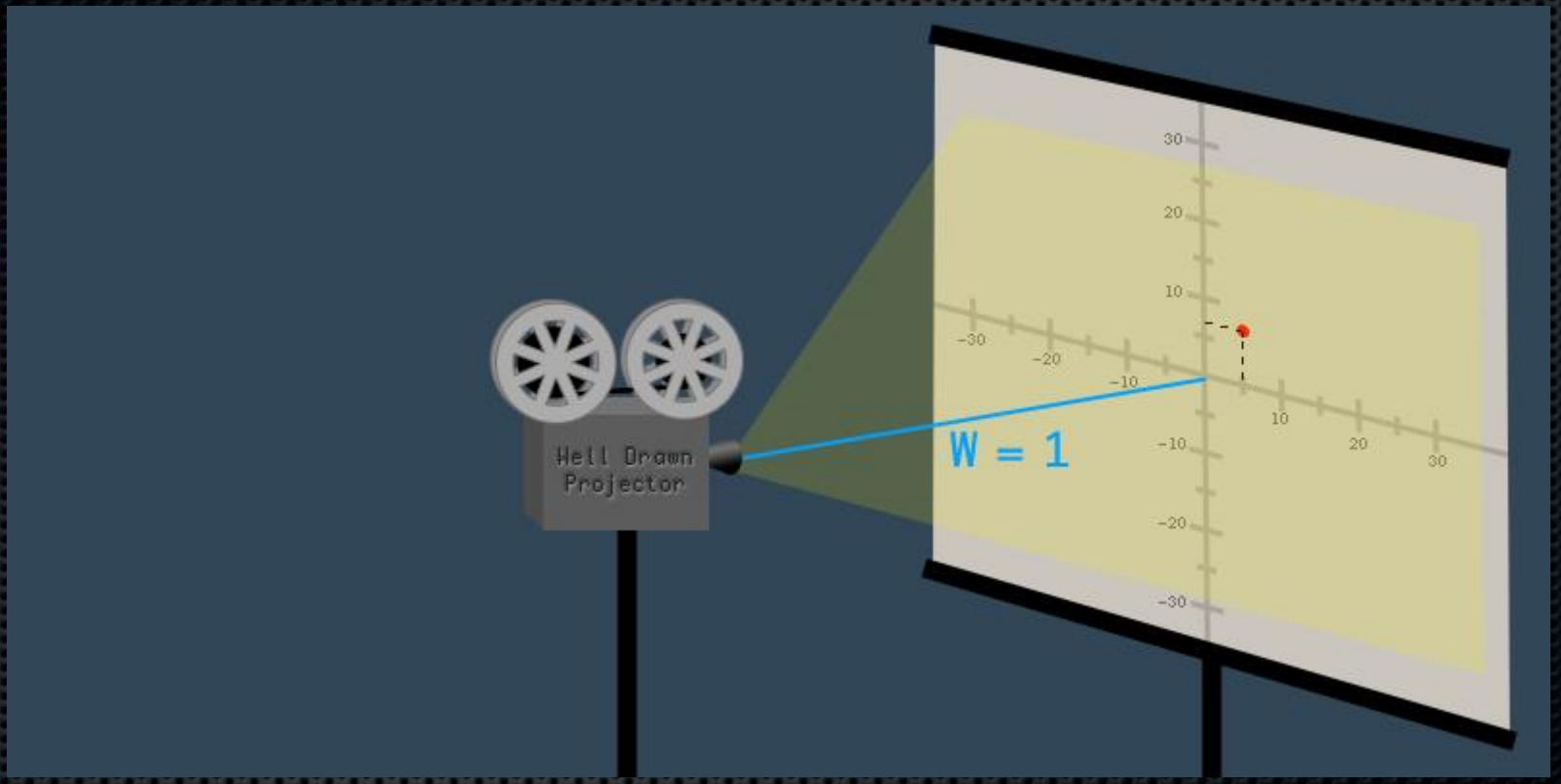



Well Drawn
Projector
Y
X


Well Drawn
Projector
Y
W
X


Hell Drawn
Projector


Well Drawn
Projector
W = 3
30
20
10
-30
-20
-10
10
20
30
-10
-20
-30


Hell Drawn
Projector
W = 1
30
20
10
-30
-20
-10
10
20
30
-10
-20
-30

Homogeneous Divide

$$v = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad v' = \begin{bmatrix} x/w \\ y/w \\ z/w \\ w/w \end{bmatrix}$$

Homogeneous Divide

$$v = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad v' = \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

Projection - Perspective

$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \qquad v = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \quad v' = \begin{bmatrix} x/z \\ y/z \\ z/z \\ z/z \end{bmatrix}$$

With homogeneous divide, just copy z to w

Projection - Perspective

$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \qquad v = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \quad v' = \begin{bmatrix} x/z \\ y/z \\ 1 \\ 1 \end{bmatrix}$$

With homogeneous divide, just copy z to w

Projection - Perspective

$$x' = \frac{x}{z} \qquad y' = \frac{y}{z}$$

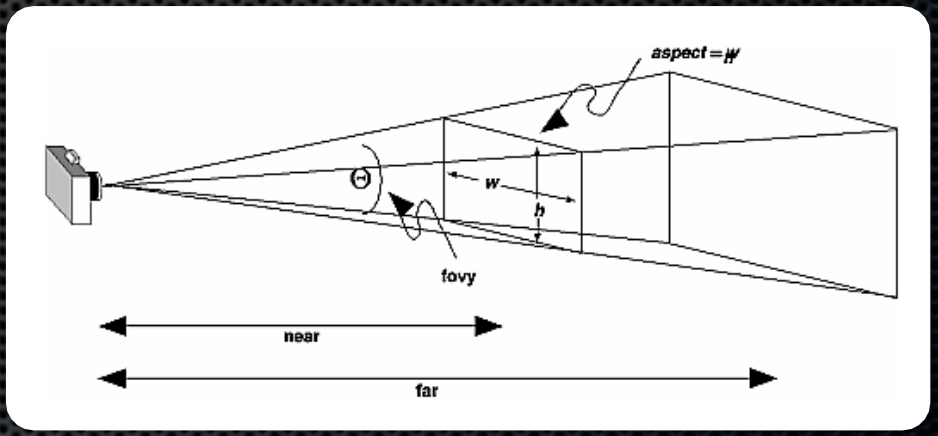
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \qquad v = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \quad v' = \begin{bmatrix} x/z \\ y/z \\ 1 \\ 1 \end{bmatrix}$$

Getting from frustum to canonical cube fills out the rest

Projection - Perspective

frustum(l,r,b,t,n,f)

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$



Z-Copy and Right Hand Rule

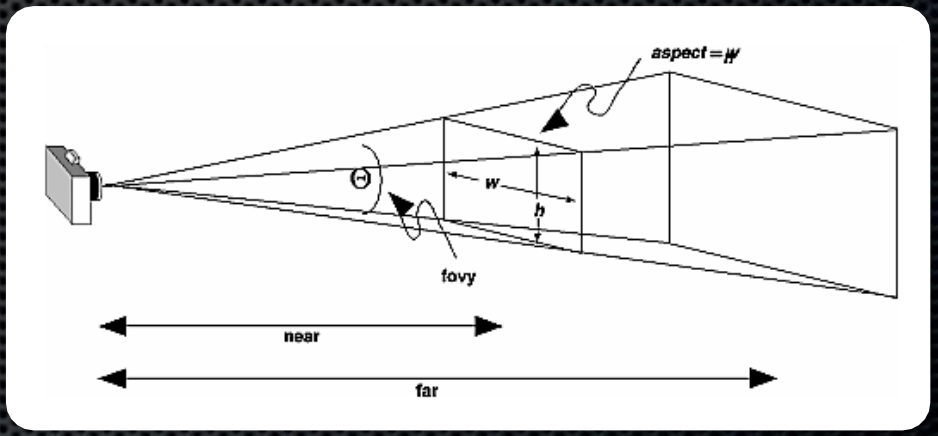
```
void perspective(float fovy, float aspect, float zNear, float  zFar)
{
    float xmin, xmax, ymin, ymax;

    ymax = zNear * tanf(fovy * M_PI / 360.0f);
    ymin = -ymax;
    xmin = ymin * aspect;
    xmax = ymax * aspect;


    frustum(xmin, xmax, ymin, ymax, zNear, zFar);
}
```

Projection - Perspective

Other adjustments due to definition of frustum

frustum(l,r,b,t,n,f)

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$



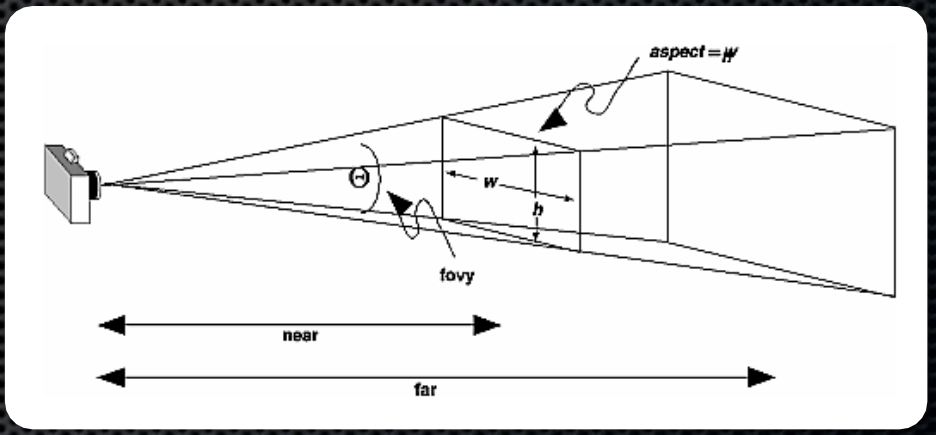
Z-Copy and Right Hand Rule

```
void perspective(float fovy, float aspect, float zNear, float  zFar)
{
    float xmin, xmax, ymin, ymax;

    ymax = zNear * tanf(fovy * M_PI / 360.0f);
    ymin = -ymax;
    xmin = ymin * aspect;
    xmax = ymax * aspect;


    frustum(xmin, xmax, ymin, ymax, zNear, zFar);
}
```

Projection - Perspective

frustum(l,r,b,t,n,f)

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$

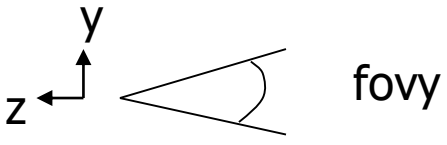


v = fovY
a = aspect
n = near
f = far

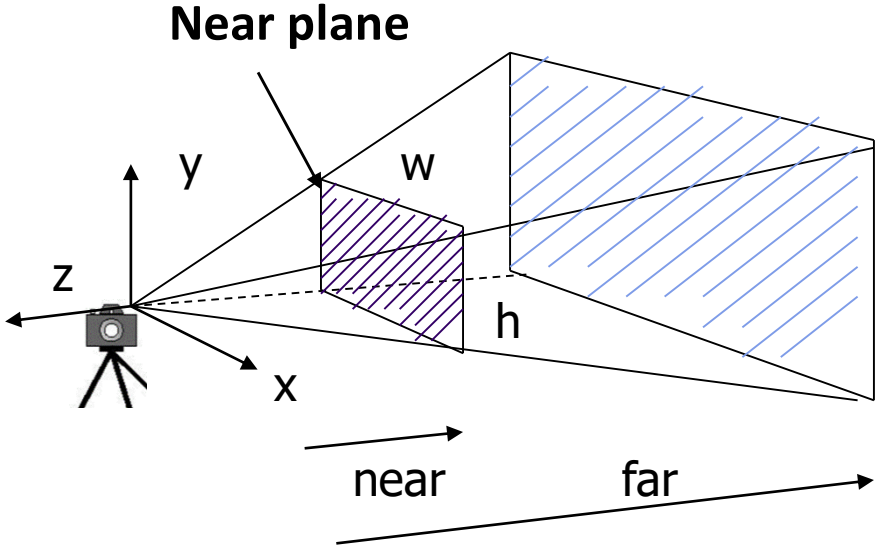
$$\begin{bmatrix} 1/(\tan(v/2)a) & 0 & 0 & 0 \\ 0 & 1/\tan(v/2) & 0 & 0 \\ 0 & 0 & -(f+n)/(f-n) & -2nf/(f-n) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspective(fovy, aspect, near, far)

- Aspect ratio used to calculate window width



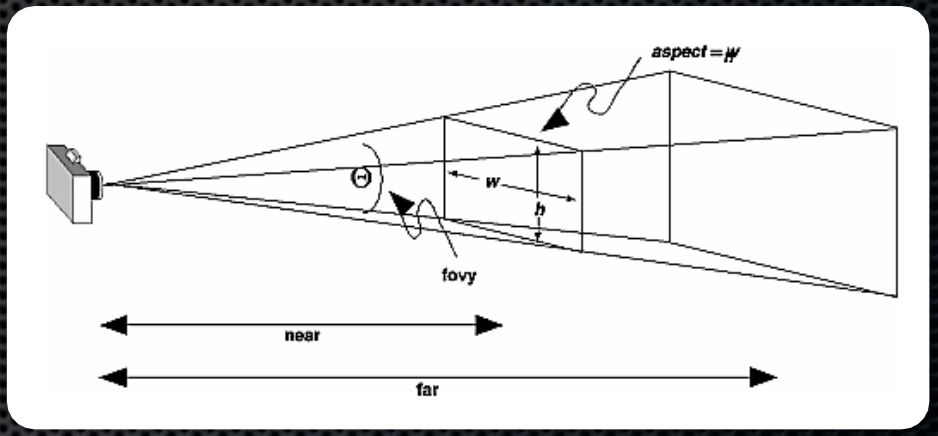
Aspect = w / h



Projection - Perspective

frustum(l,r,b,t,n,f)

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$



v = fovY
a = aspect
n = near
f = far
copy Z

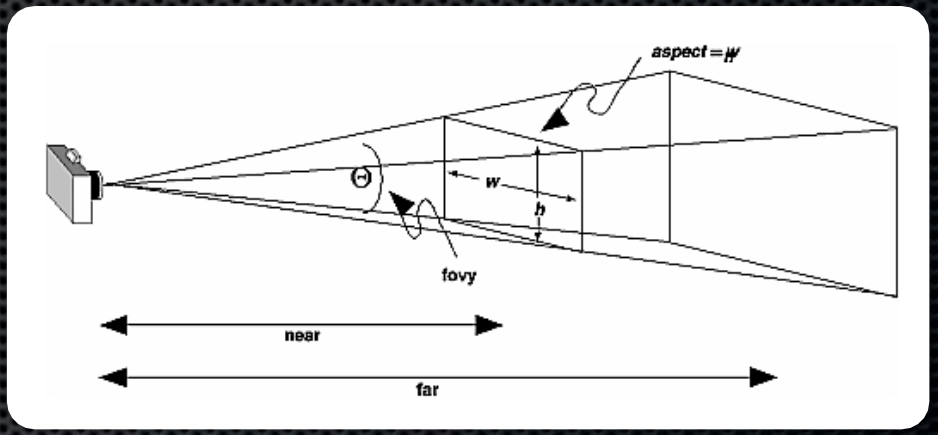
$$\begin{bmatrix} 1/(\tan(v/2)a) & 0 & 0 & 0 \\ 0 & 1/\tan(v/2) & 0 & 0 \\ 0 & 0 & -(f+n)/(f-n) & -2nf/(f-n) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Projection - Perspective

scales near plane to -1 → 1 in x,y

frustum(l,r,b,t,n,f)

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$



v = fovY
a = aspect
n = near
f = far
copy Z

$$\begin{bmatrix} 1/(\tan(v/2)a) & 0 & 0 & 0 \\ 0 & 1/\tan(v/2) & 0 & 0 \\ 0 & 0 & -(f+n)/(f-n) & -2nf/(f-n) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

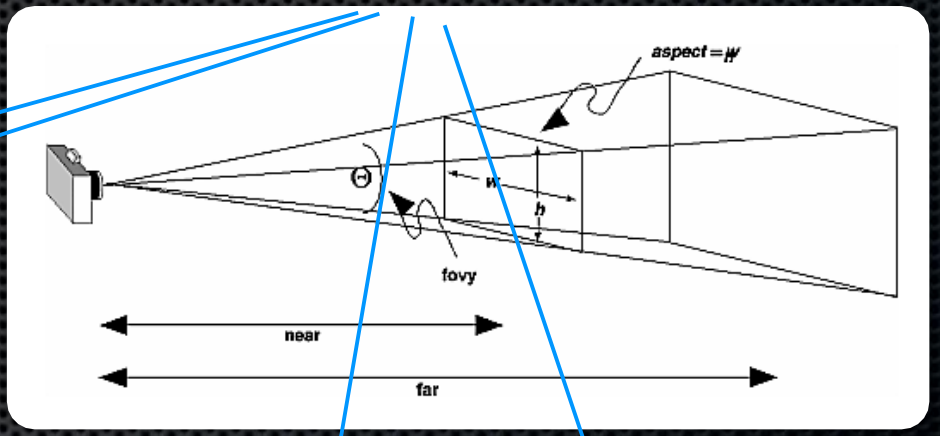
Projection - Perspective

scales near plane to -1 → 1 in x,y

scales and biases z to -1 → 1

frustum(l,r,b,t,n,f)

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$



v = fovY

a = aspect

n = near

f = far

copy Z

$$\begin{bmatrix} 1/(\tan(v/2)a) & 0 & 0 & 0 \\ 0 & 1/\tan(v/2) & 0 & 0 \\ 0 & 0 & -(f+n)/(f-n) & -2nf/(f-n) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Typical Matrices