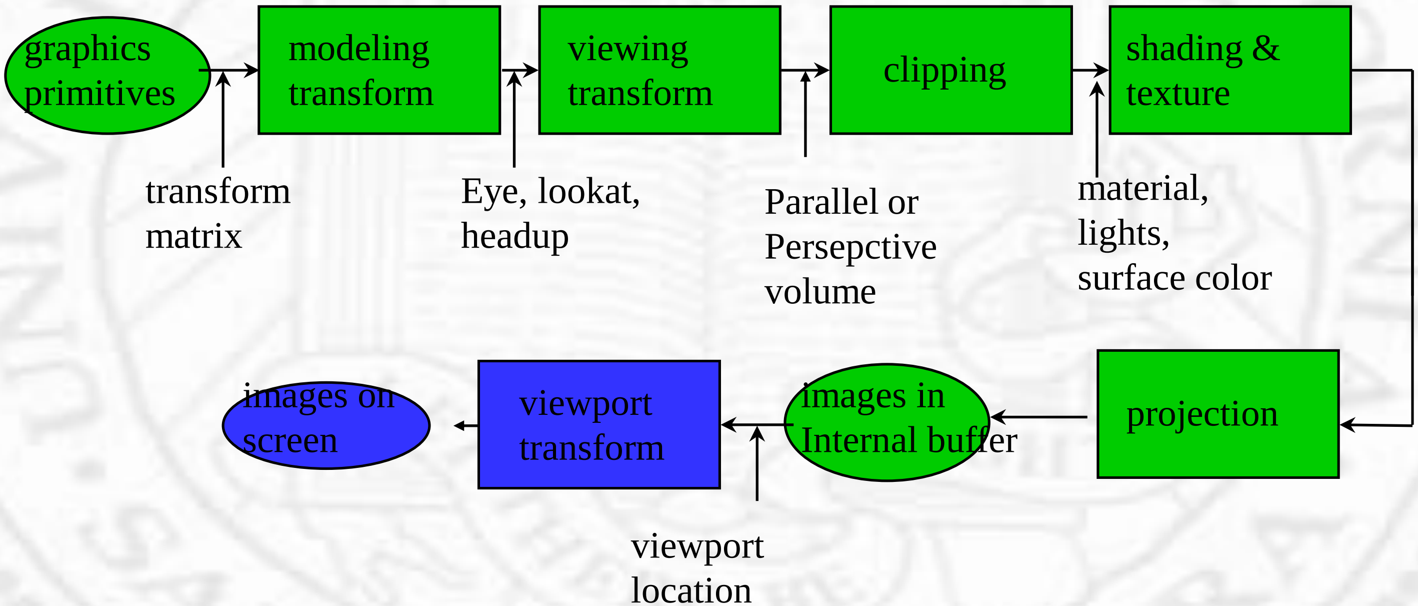
Assignment #3

- As a graphics engineer, can you implement a “skeletal” rendering pipeline?



Modeling Transform

- As a global system
  - objects move but coordinates stay the same
  - apply in the *reverse* order

glMatrixMode(GL_MODELVIEW);

glLoadIdentity(T_1);

glMultiMatrixf(T_2);

…

glMultiMatrixf(T_n);

draw_the_object(v);

v' = $\mathbf{I T_1 T_2 T_n v}$

Tasks

- Need a stack (CS16, 24)
  - Push & pop
- Need to create matrices (4x4)
- Need to multiply matrices (4x4)

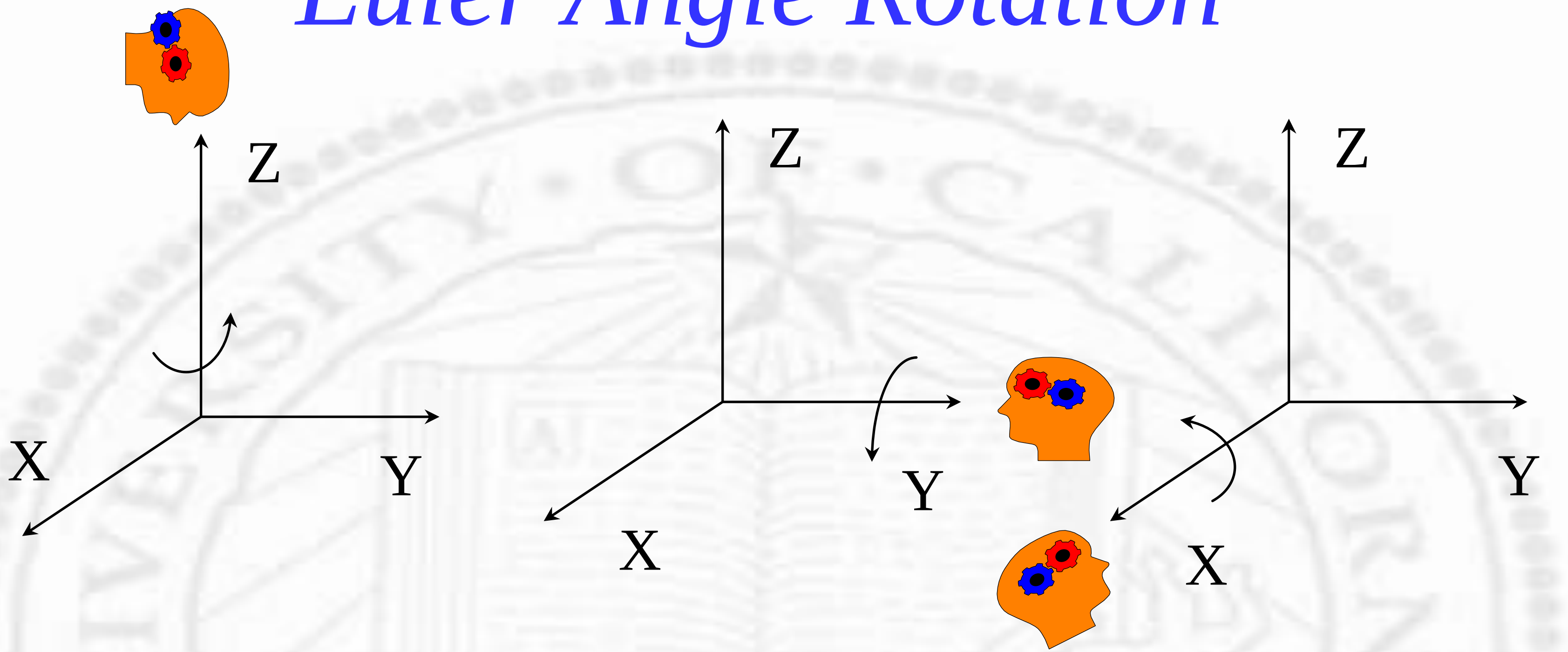
Why 4x4 not 3x3?

- Use of homogeneous coordinates
  - [x,y] -> [wx,wy,w]
  - [wx,wy,w] -> [wx/w, wy/w, w] -> [x,y]
  - [x,y,z] -> [wx, wy, wz, w]
  - [wx, wy, wz, w] -> [wx/w, wy/w, wz/w] ->[x,y,z]
- One dimension up (w!=0)

Reason #1

- All operations (including translation) are now matrix operations
- Hierarchical transforms (multiple T, R, S) are computed once (top matrix in the stack) and applied

Euler Angle Rotation



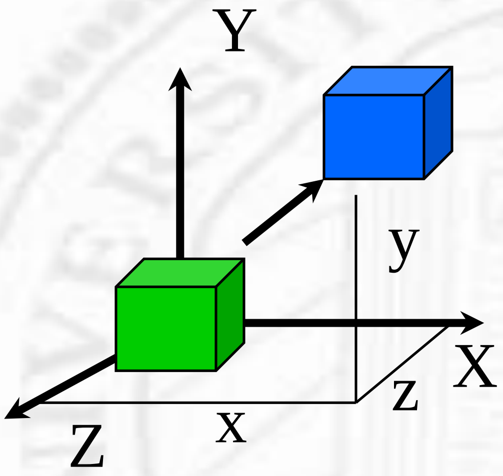
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

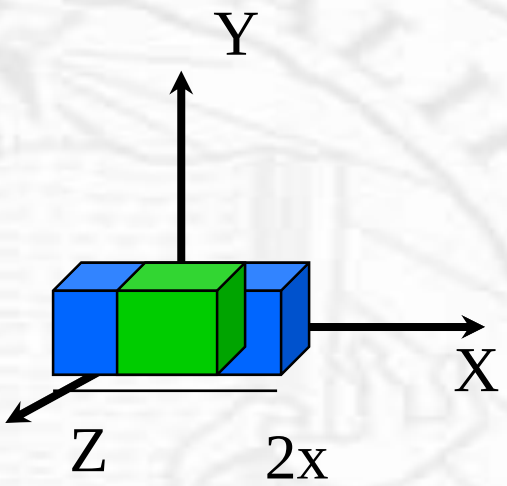
Transformations

- Translation



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

- Scaling



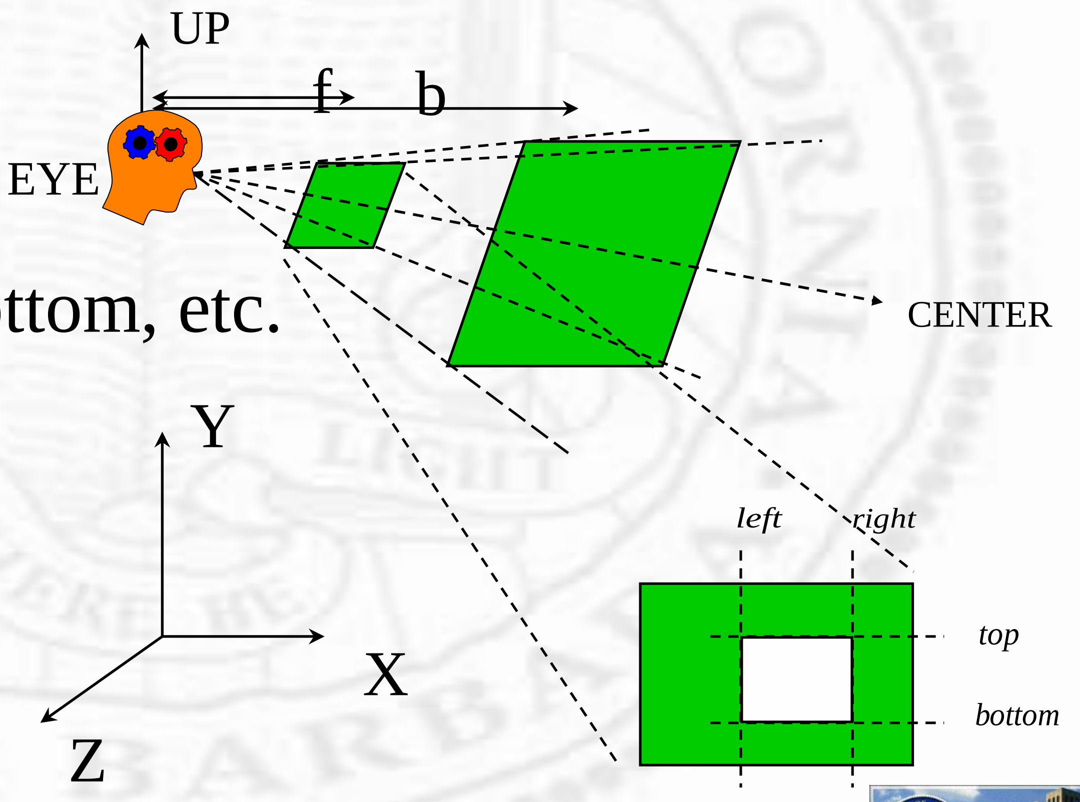
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Modeling Transform Your Way

- Implement a stack with push and pop
- Replace OpenGL codes with your own codes

Viewing Transform

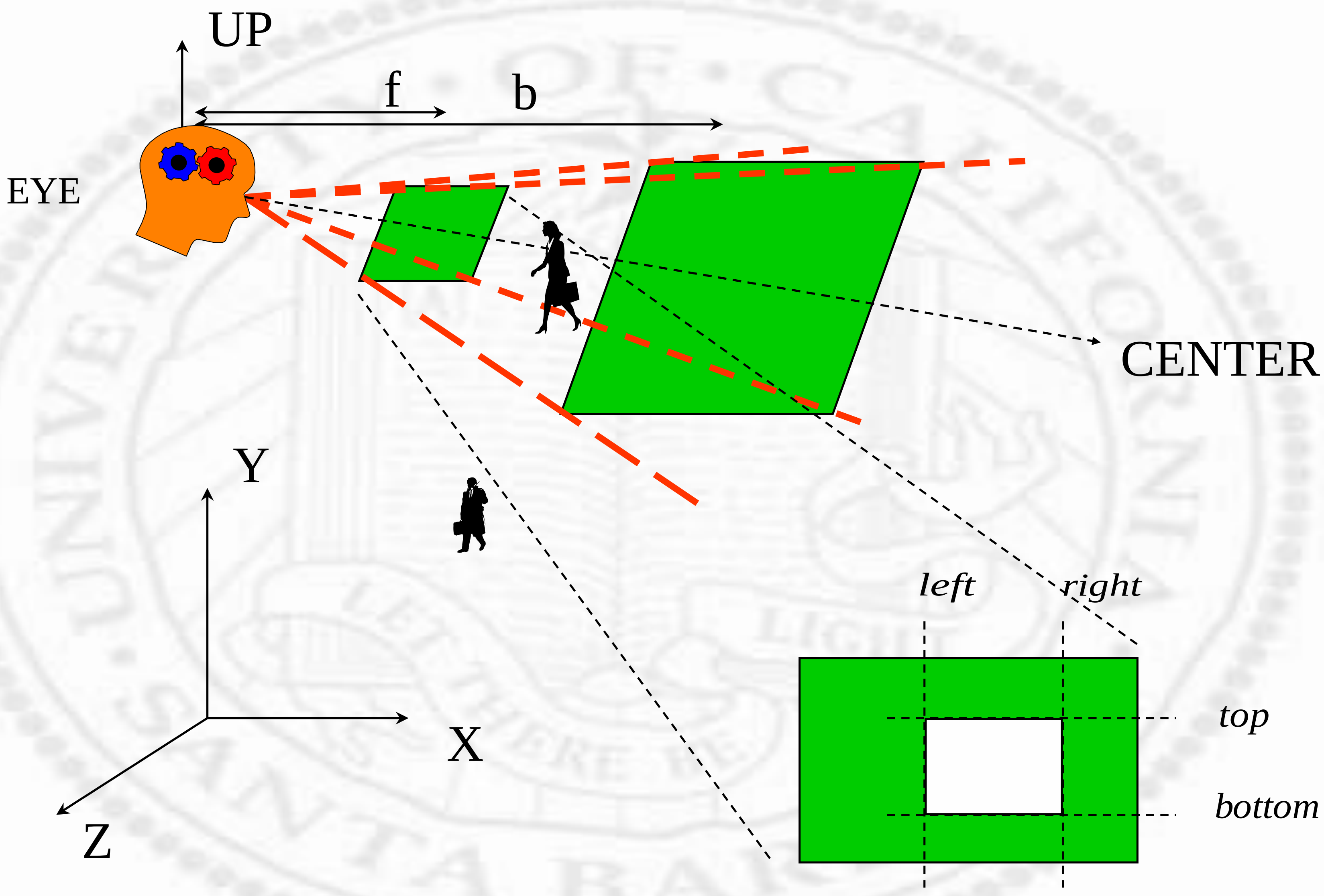
- So the user specifies a lot of information
  - Eye
  - Center
  - Up
  - Near, far,
  - Left, right top, bottom, etc.



What does OpenGL do?

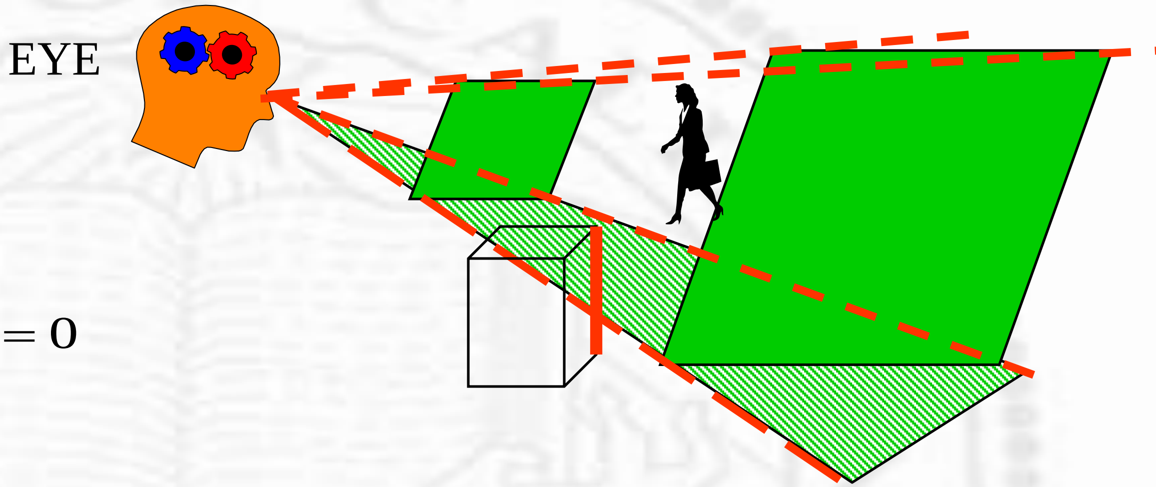
- What does a system programmer do with those numbers?
- Generate screen coordinates *correctly* and *efficiently*
  - Inside/outside test
  - Projection
- *Here comes the part which contains math which you may not like*
- But all you need to know is matrix operation

Arbitrary View Volume



Inside-Outside Test (Clipping)

- Intersection of
  - A plane and
  - A Line



$$plane: ax + by + cz + d = 0$$

$$line: \begin{cases} x_1 + t(x_2 - x_1) \\ y_1 + t(y_2 - y_1) \\ z_1 + t(z_2 - z_1) \end{cases}$$

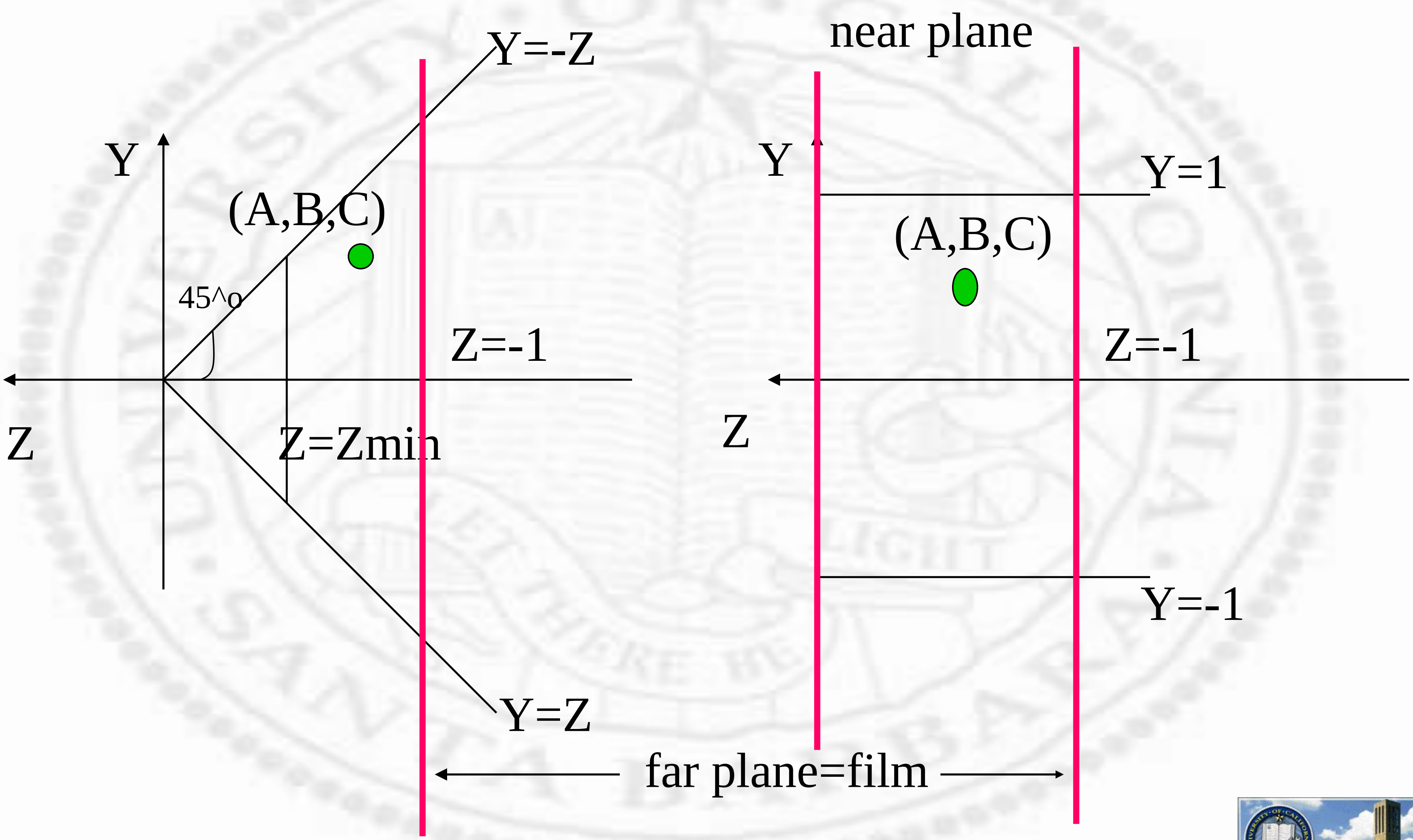
$$a[x_1 + t(x_2 - x_1)] + b[y_1 + t(y_2 - y_1)] + c[z_1 + t(z_2 - z_1)] + d = 0$$

$$t = -\frac{ax_1 + by_1 + cz_1 + d}{a(x_2 - x_1) + b(y_2 - y_1) + c(z_2 - z_1)}$$

$$0 \le t \le 1$$

$$(x_1, y_1, z_1), (x_2, y_2, z_2): \text{end points of line}$$

Clipping in Canonical Volumes



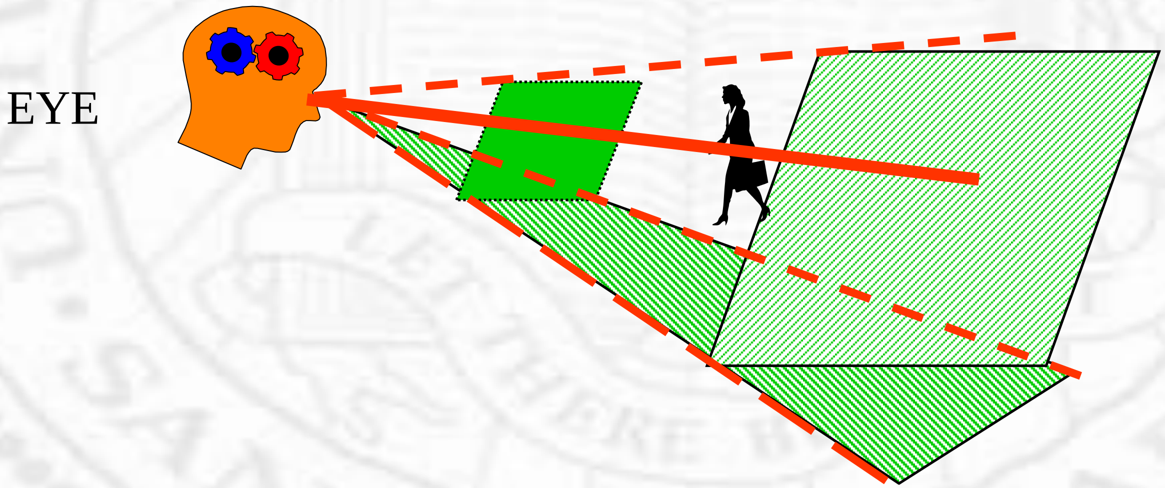
Clipping with 6-bit outcode

- Perspective
- Above y>-z
- Below y<z
- Right x>-z
- Left   x<z
- Behind z<-1
- In front z >zmin

- Parallel
- Above y>1
- Below y<-1
- Right x>1
- Left   x<-1
- Behind z<-1
- In front z >0

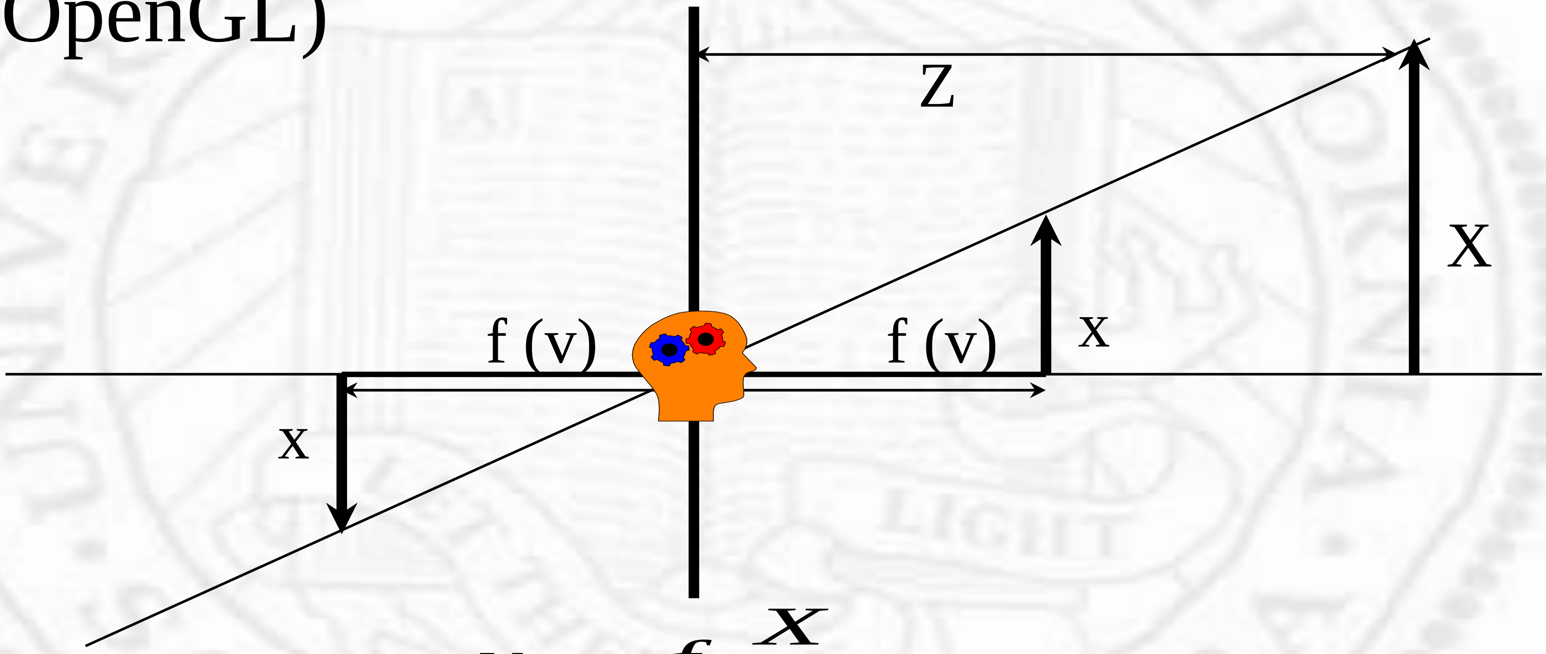
Projection

- Again, an intersection of
  - A plane and
  - A Line



Sidebar: Reason #2

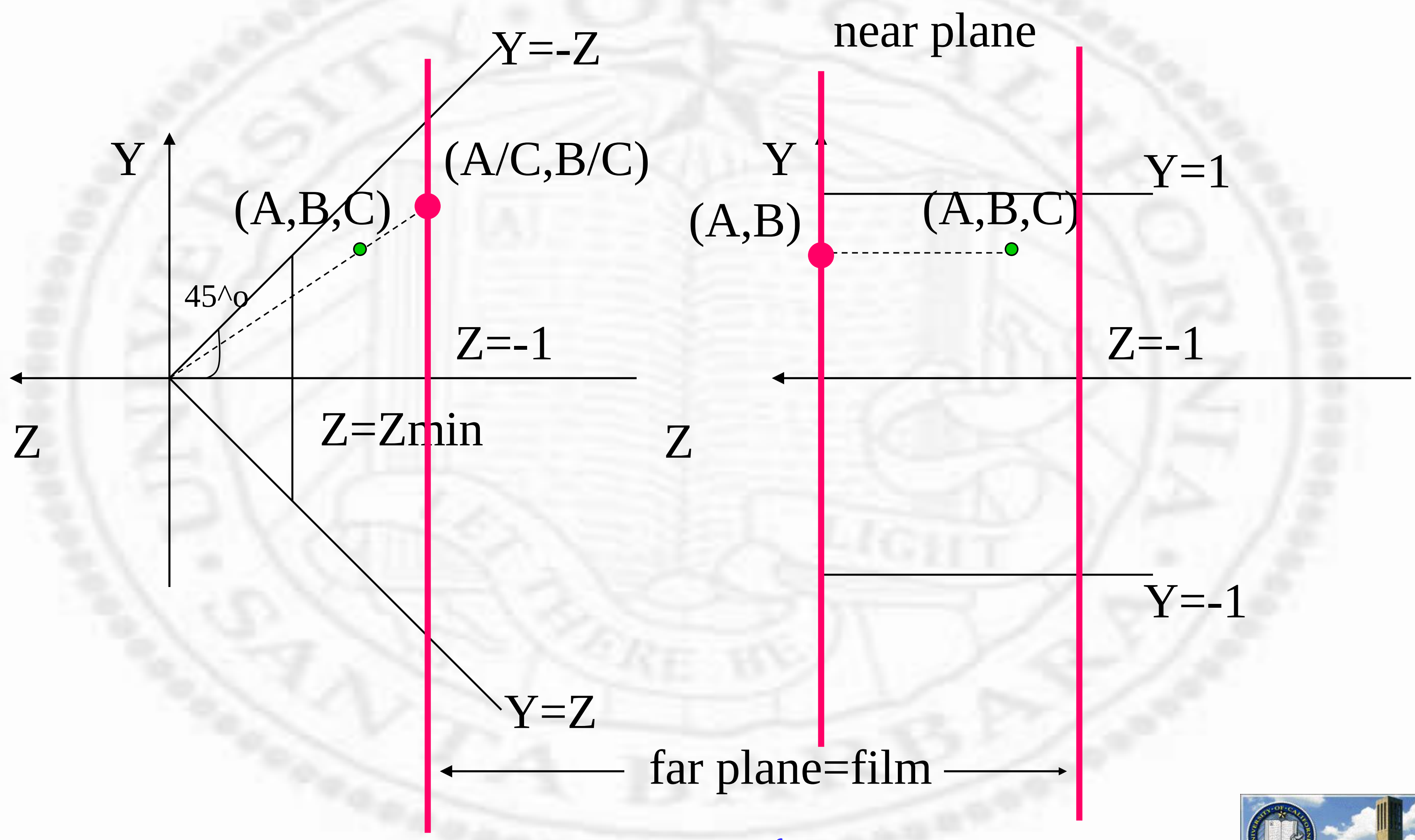
- A pin-hole model without inversion (f=1 in OpenGL)



$$x = f\,\frac{X}{Z}$$

$$y = f\,\frac{Y}{Z}$$

Canonical Volumes



Sidebar: Reason #2

- Traditional Way

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$

- Matrix Way

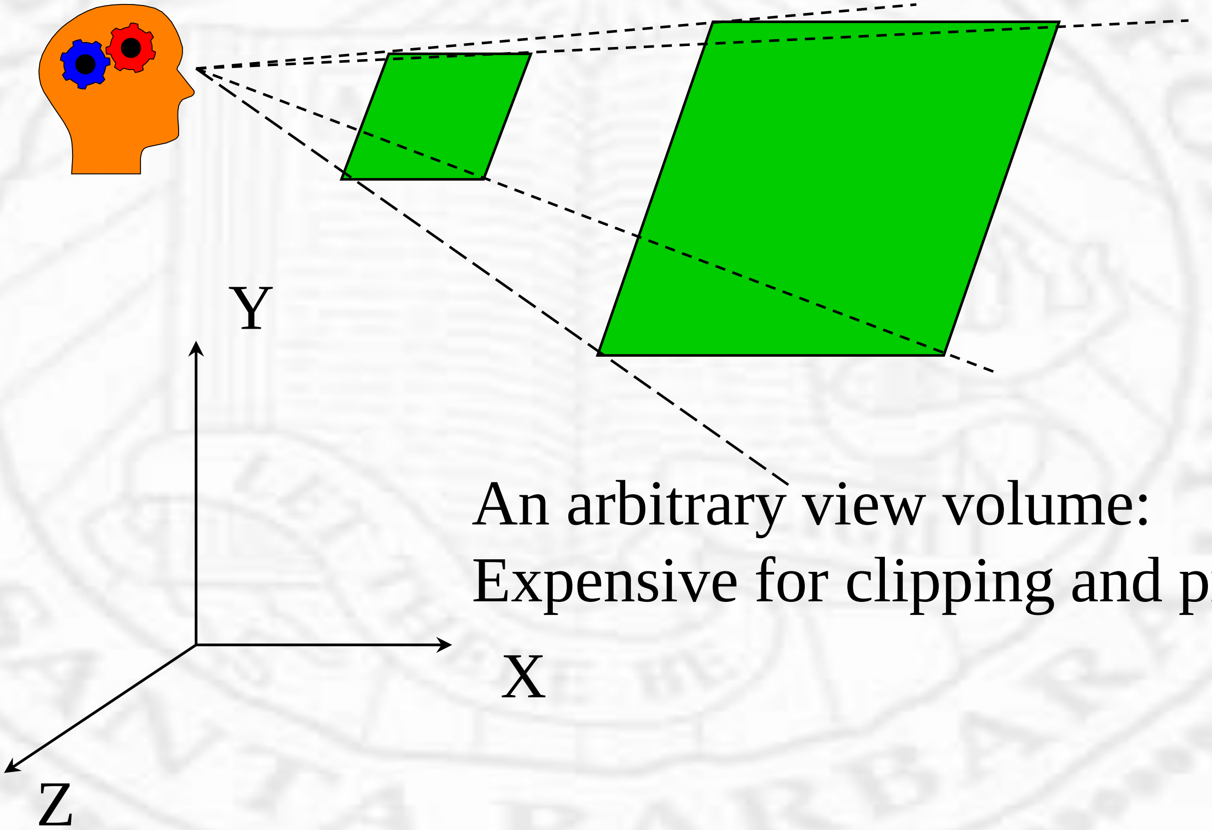
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Problem

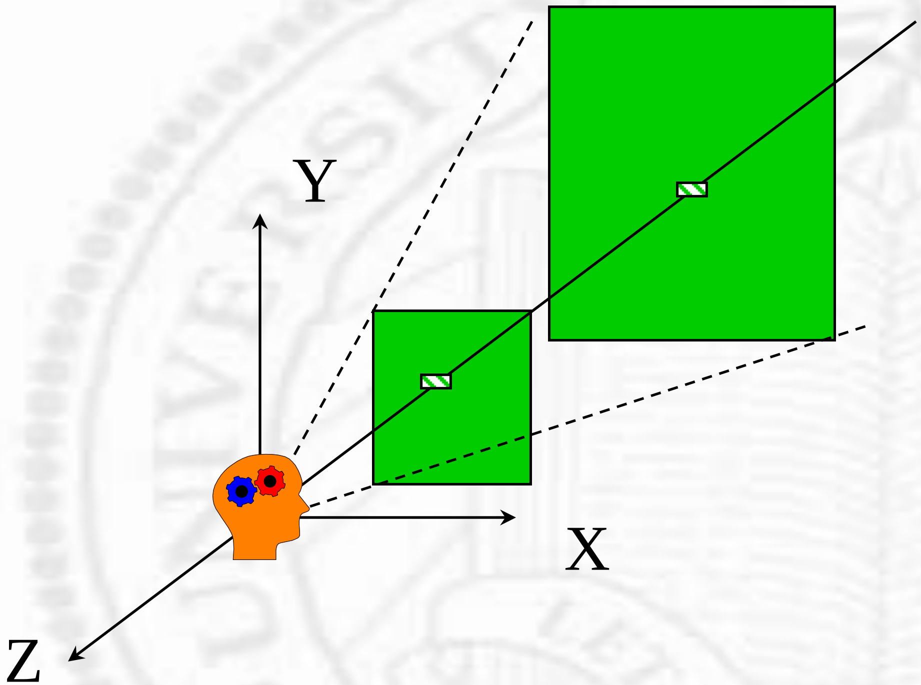
- Both clipping and projection can be done efficiently in a canonical volume
- But we do not have a canonical volume in general
- Solution: Normalization transform
  - A single matrix operation to bring objects in any arbitrary volume into a canonical volume
  - *Cannot change what the user sees*

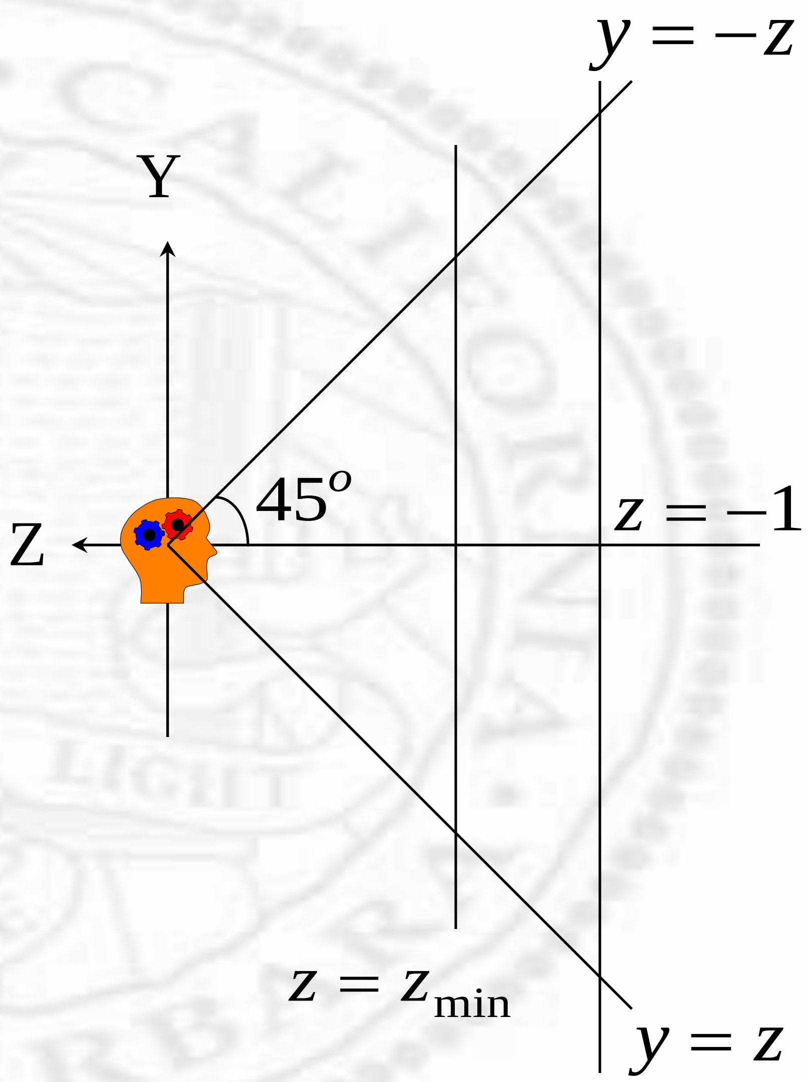
Normalization Transform

- A transformation to facilitate clipping and projection



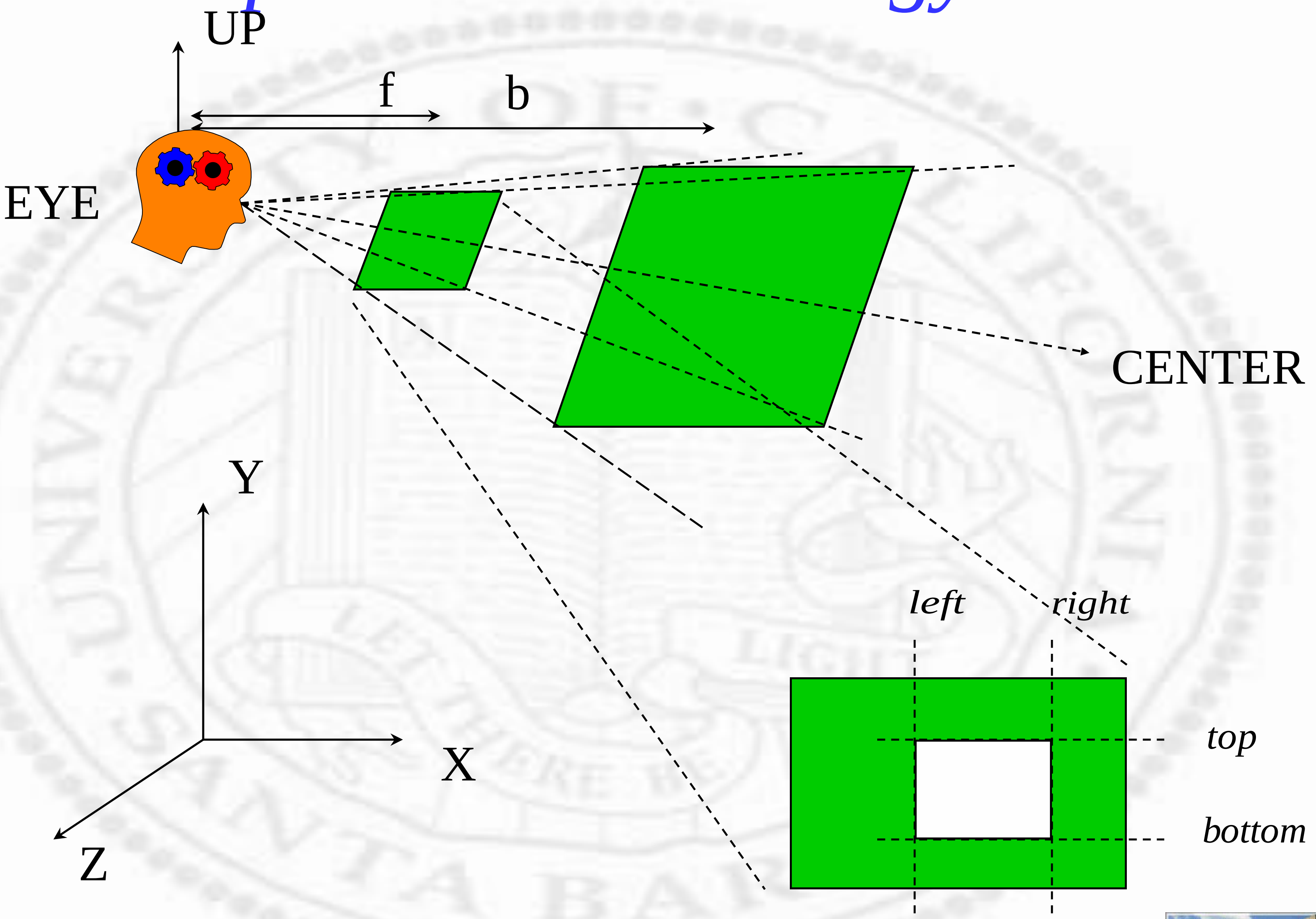
An arbitrary view volume:
Expensive for clipping and projection





The canonical view volume:
Simple clipping (six-bit outcode)
Simple projection (x/z, y/z)

OpenGL Terminology

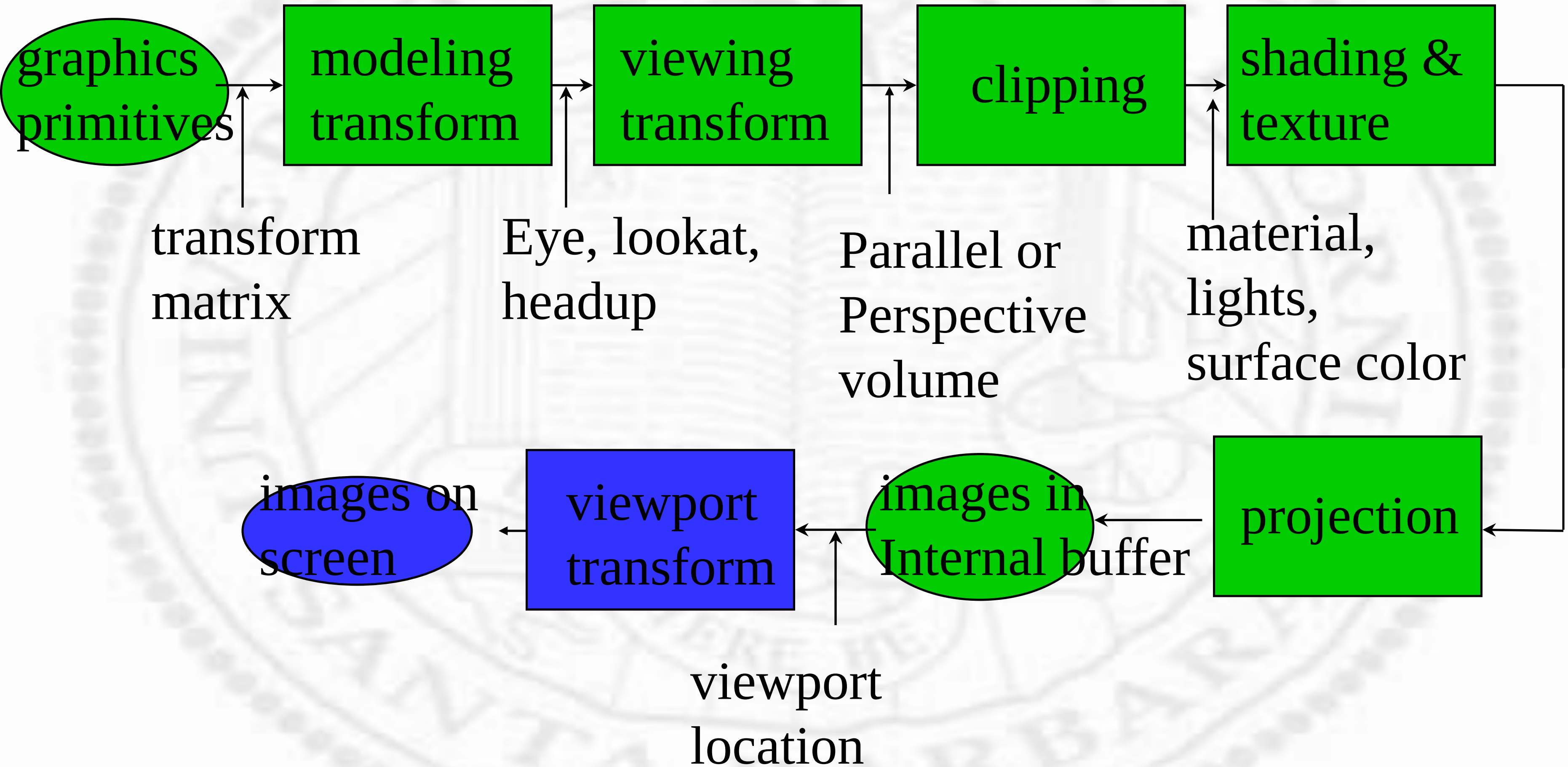


Normalization Transform

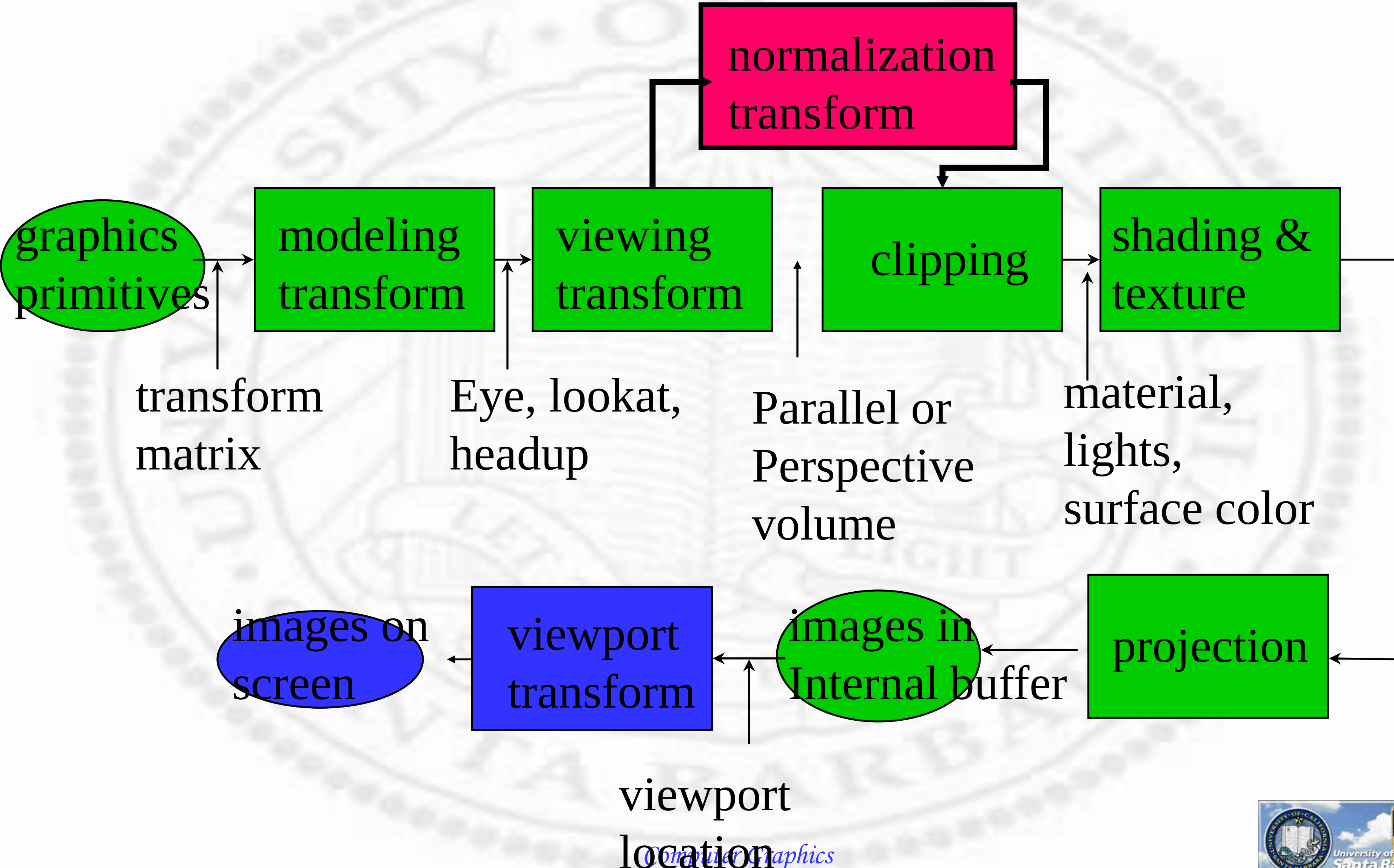
Perspective - OpenGL

- External parameters
  - Translate EYE into origin
  - Rotate the EYE coordinate system such that
    - w (e-c) becomes z
    - u becomes x
    - v becomes y
- Internal parameters
  - Shear to have centerline of the view volume aligning with z
  - Scale into canonical truncated pyramid

Existing Rendering Pipeline



Rendering Pipeline with Normalization Transform



Changes

- Modeling + Viewing + Normalization get concatenated into ONE transform before applying to any primitives
- Confusion: normalization does not just push the eye frame back to origin and line up with world frame, it pushes objects away too
- Purpose: to make clipping and projection much more efficient

Clipping?

- Polygon clipping algorithm discussed earlier
- For inside/outside determination

What Else?

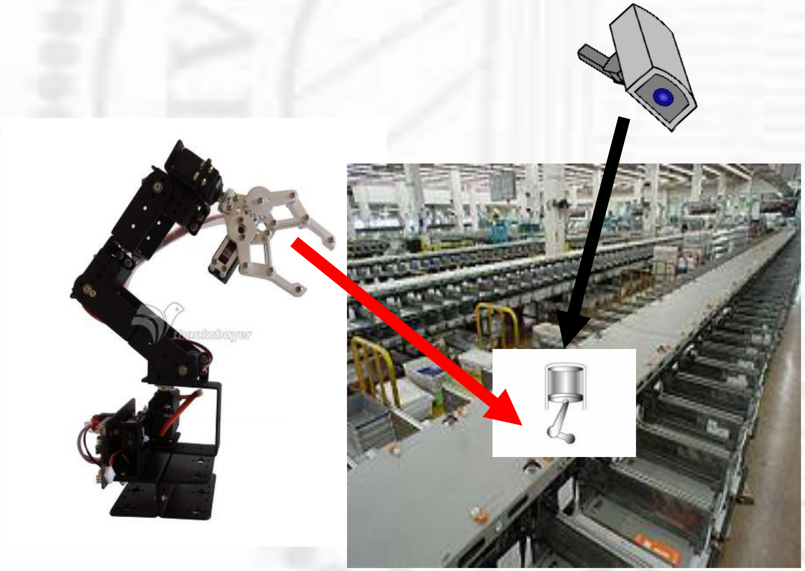
- Start from polygon with vertices
- End with polygon with vertices
- Visible surface determination
  - Painter's algorithm (sort by depth)
- 2D painting
  - Interior: Scan conversion
  - Exterior: Line drawing (Bresemheim)
- That is!

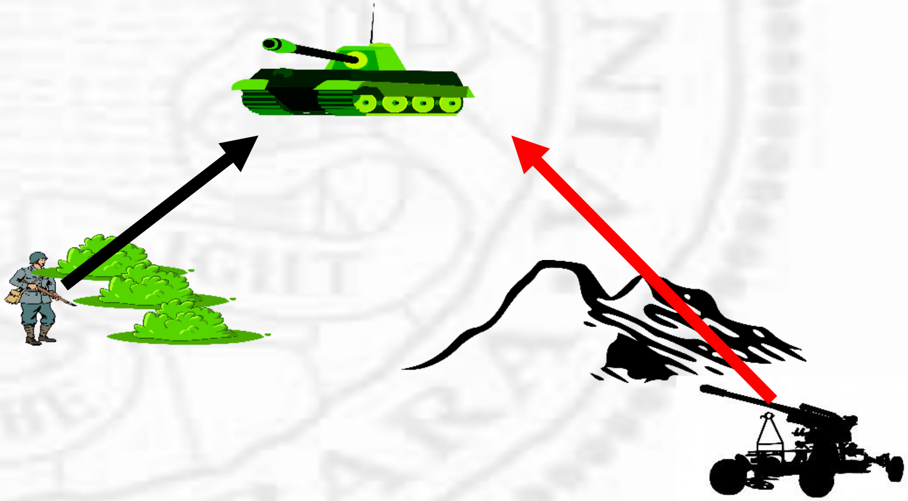
Viewing Normalization

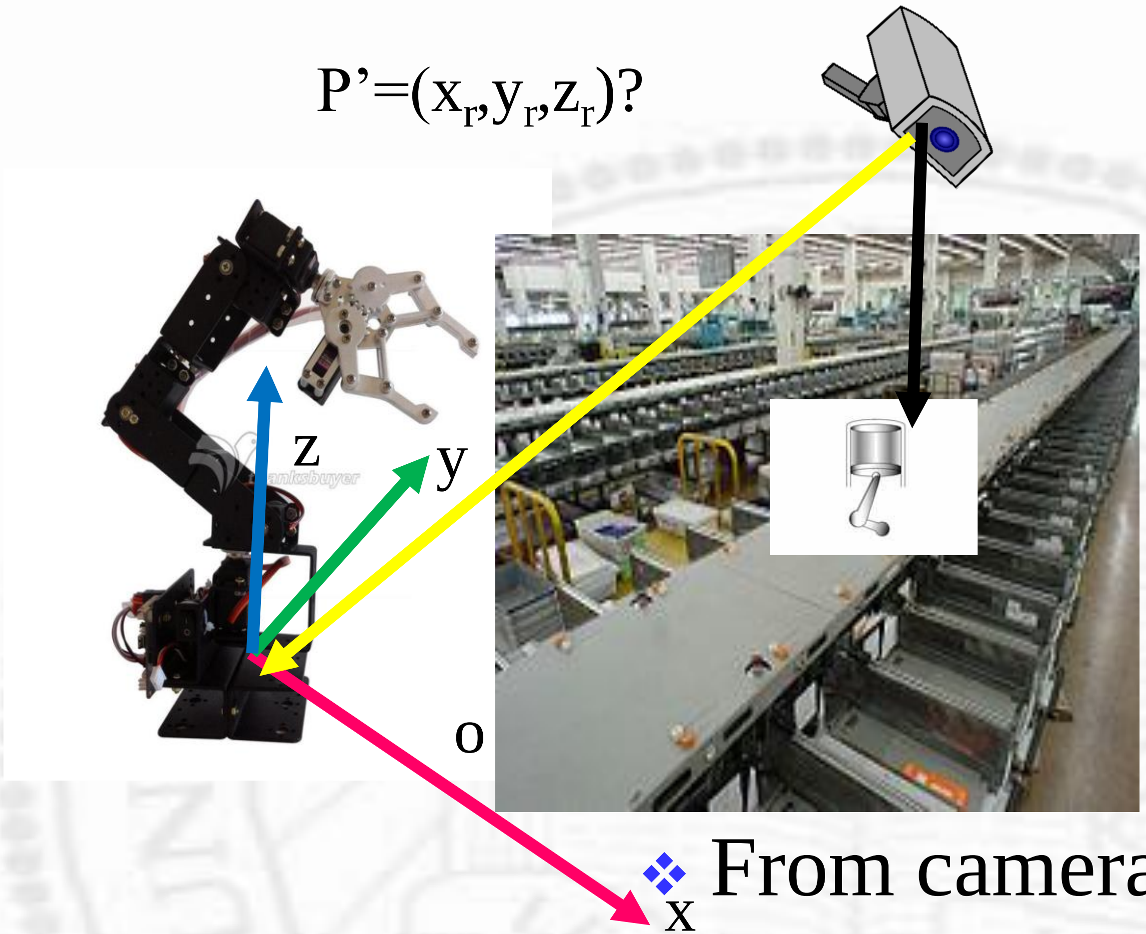
- Line up (X-Y-Z) and (U-V-W)
- Initially, (U-V-W) are specified in (X-Y-Z) system (In fact, everything is specified in X-Y-Z system)
- Some point in time, want to specify things in (U-V-W) system, or U becomes (1,0,0), V becomes (0,1,0), W becomes (0,0,1)
- Translation (easy) + Rotation (hard)

Hand-eye Coordination

- A common problem in graphics, robotics and vision
- The camera and effector are not co-located

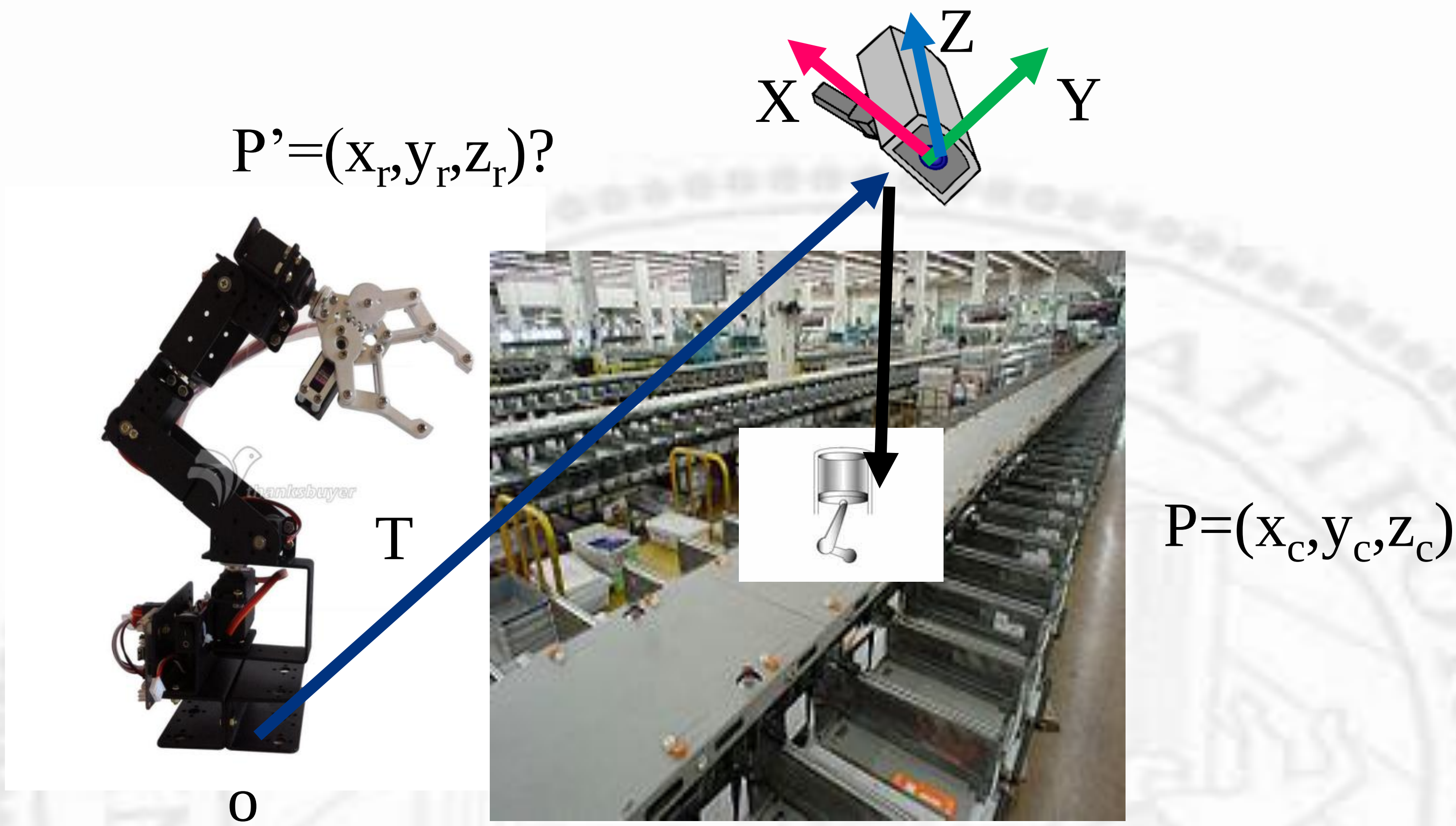






- From camera to robot
- **R** and **O** are in camera’s frame
- Rows are robot’s frame in camera’s system

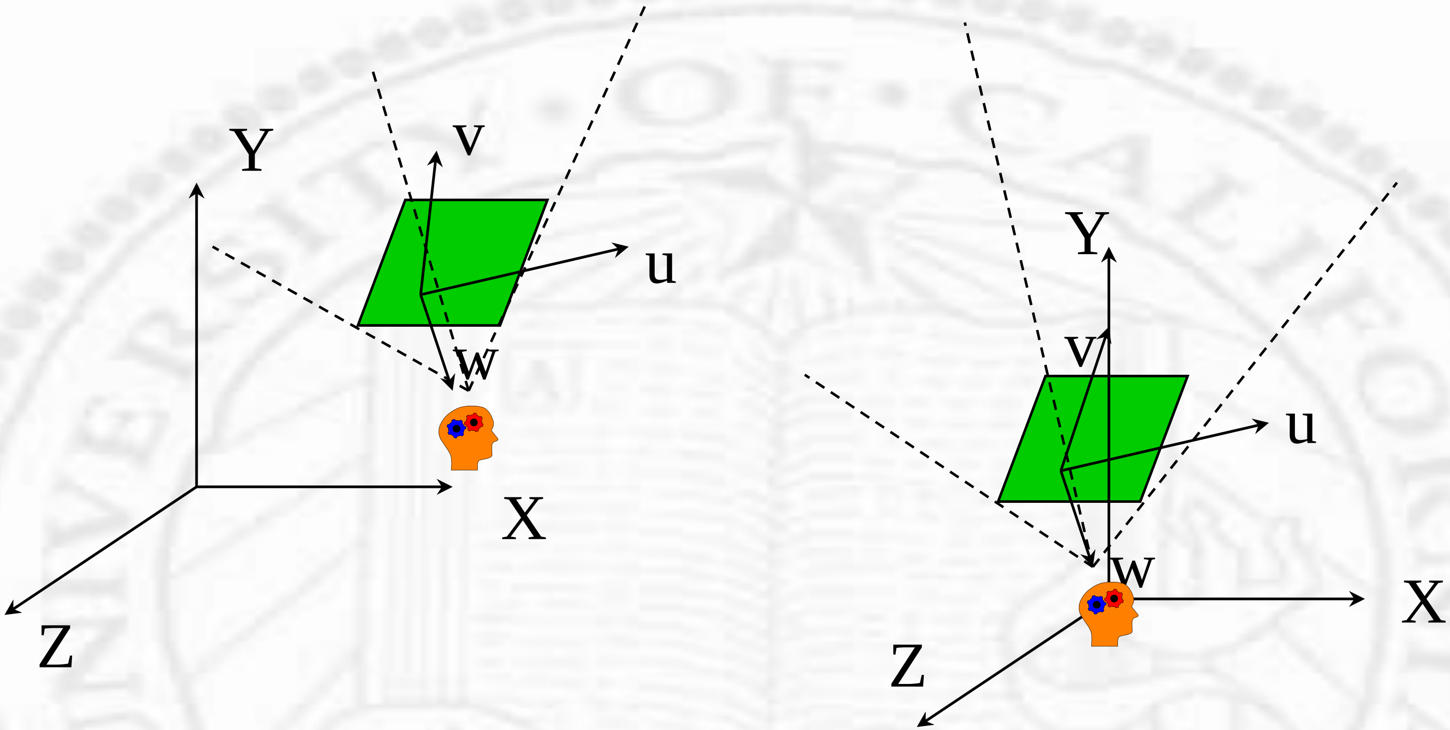
$$\mathbf{P}' = \begin{bmatrix} - & \mathbf{x}' & - \\ - & \mathbf{y}' & - \\ - & \mathbf{z}' & - \end{bmatrix} (\mathbf{P} - \mathbf{O})$$



$$\mathbf{P}' = \begin{bmatrix} - & \mathbf{x}' & - \\ - & \mathbf{y}' & - \\ - & \mathbf{z}' & - \end{bmatrix}(\mathbf{P}-\mathbf{O}) = \begin{bmatrix} | & | & | \\ X & Y & Z \\ | & | & | \end{bmatrix}(\mathbf{P}-\mathbf{O}) = \begin{bmatrix} | & | & | \\ X & Y & Z \\ | & | & | \end{bmatrix}\mathbf{P} - \begin{bmatrix} | & | & | \\ X & Y & Z \\ | & | & | \end{bmatrix}\mathbf{O} = \begin{bmatrix} | & | & | \\ X & Y & Z \\ | & | & | \end{bmatrix}\mathbf{P} + \mathbf{T}$$

- From robot to camera
- **R** and **T** are in robot's frame
- Cols are camera's frame in robot's system

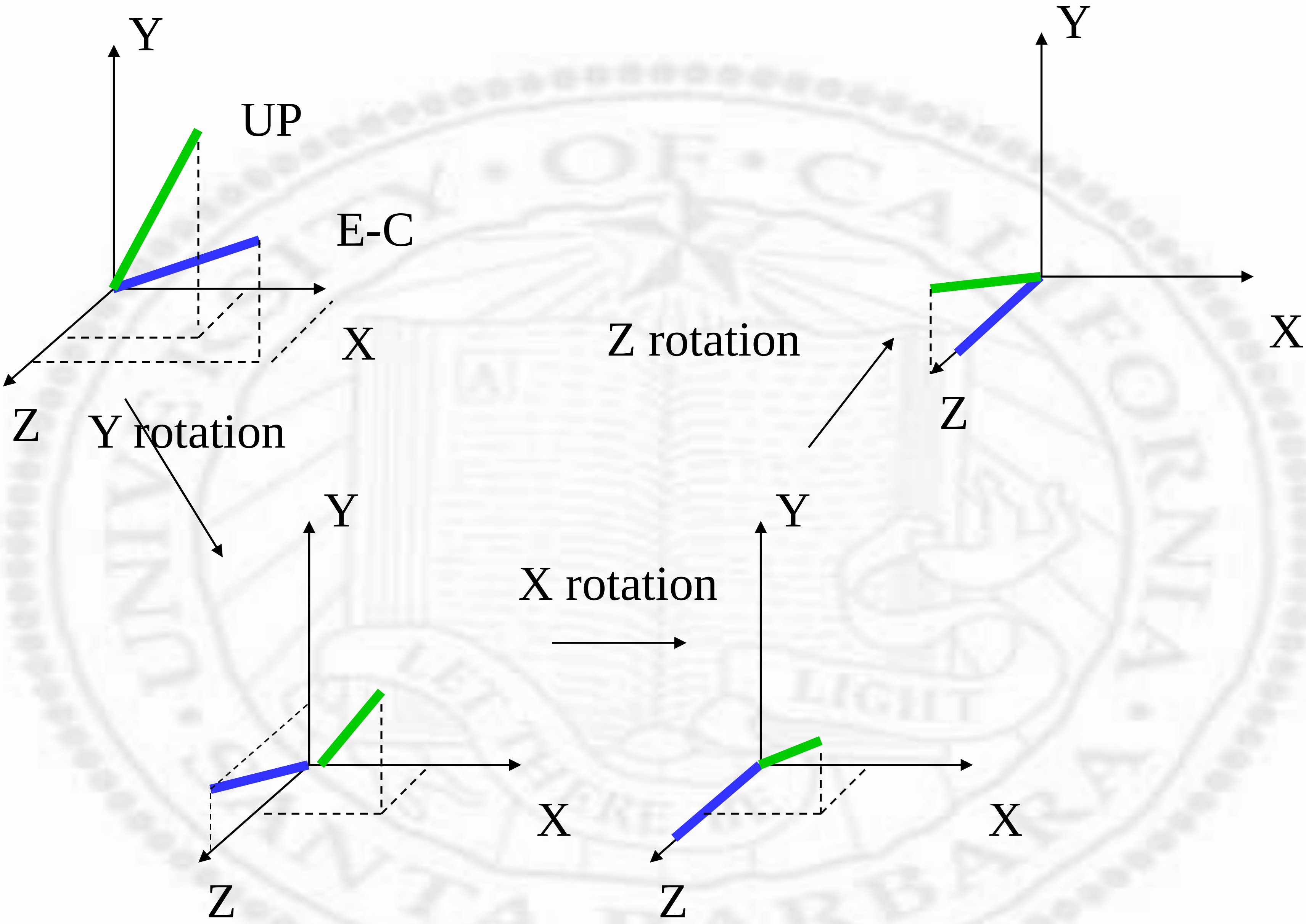
Translate EYE into the origin



$$T_1 = \begin{bmatrix} 1 & 0 & 0 & -EYE_x \\ 0 & 1 & 0 & -EYE_y \\ 0 & 0 & 1 & -EYE_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewing Normalization

- Three rotations
  - Rotate about Y
  - Rotate about X
  - Rotate about Z


Y
UP
E-C
X
Z
Y rotation
Z rotation
Y
X
Z
Y
X rotation
X
Z
Y
X
Z

Viewing Normalization

- Figuring out **[u, v, w]** in **[x, y, z]** system
- Applying a rotation to transform **[x, y, z]** coordinates into **[u, v, w]** coordinates

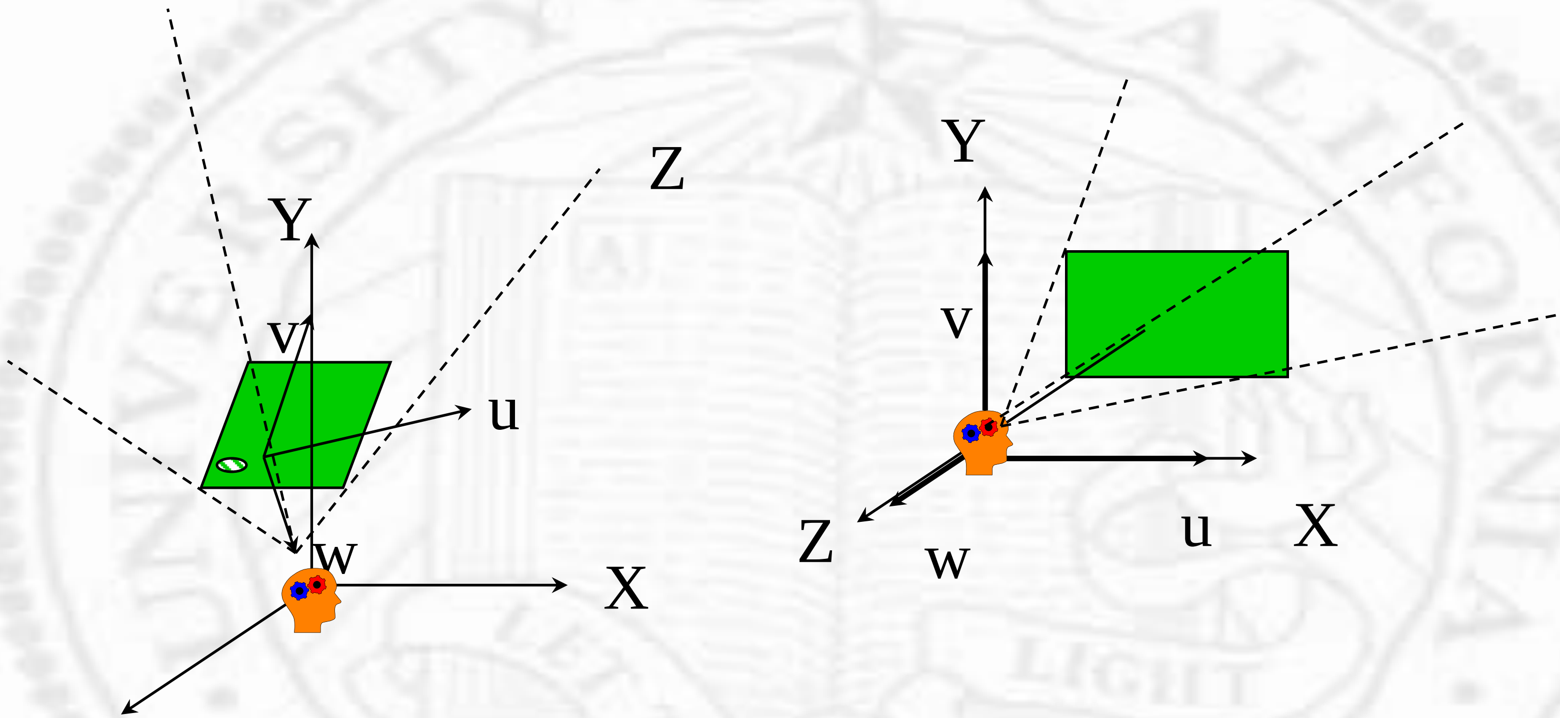
$$w = \frac{e - c}{|e - c|}$$

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

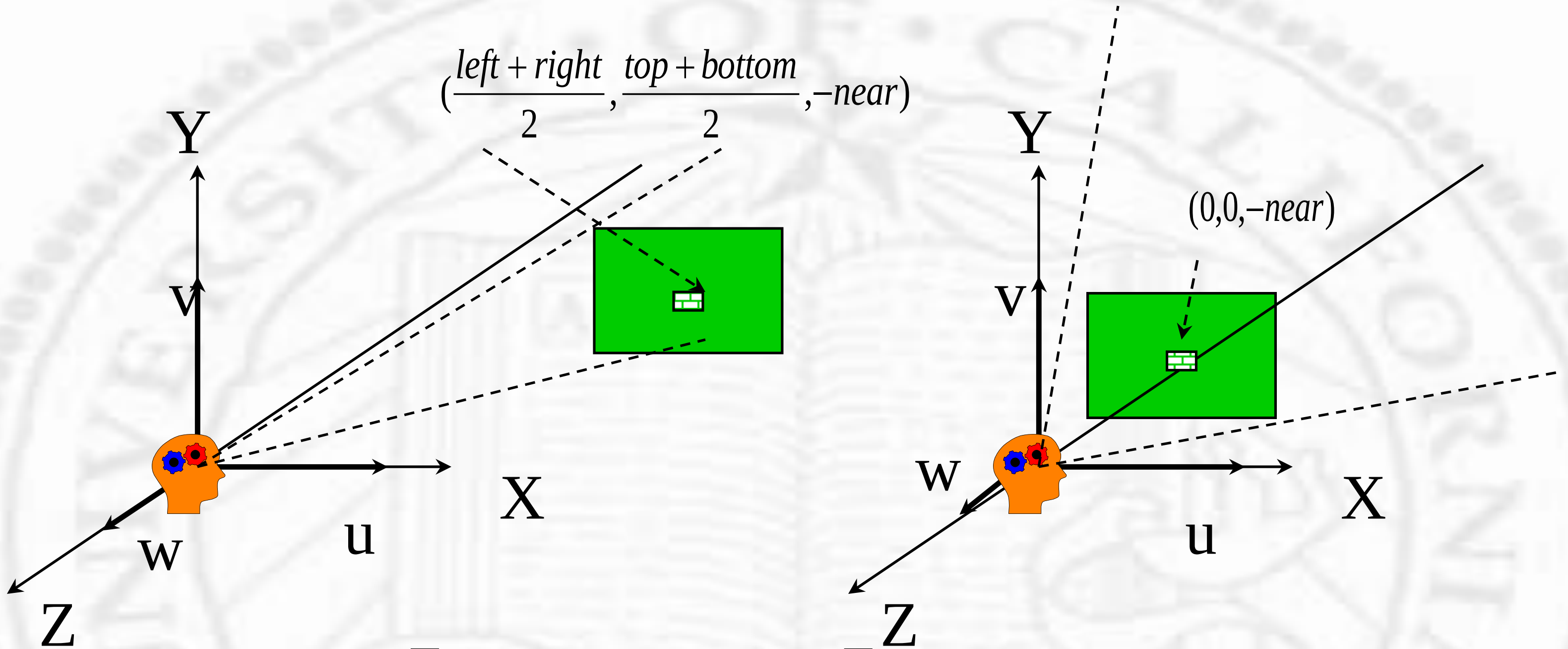
$$v = w \times u$$

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotate EYE coordinate to align w. world system



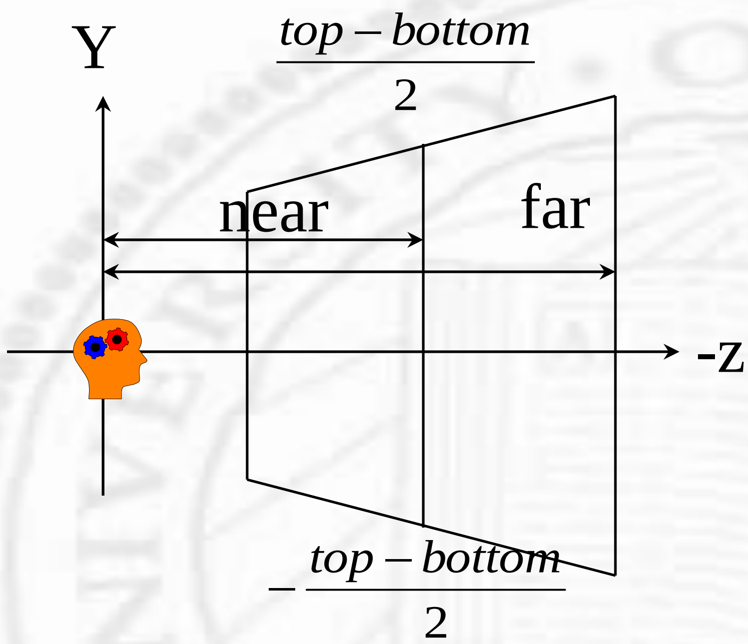
Shear

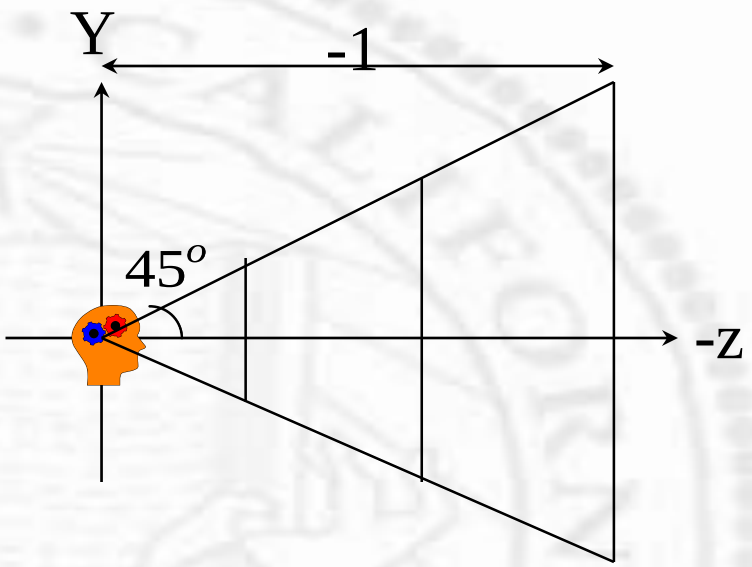


$$SH = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$a = \frac{\frac{left+right}{2}}{near}, b = \frac{\frac{top+bottom}{2}}{near}$$

Scale into canonical volume





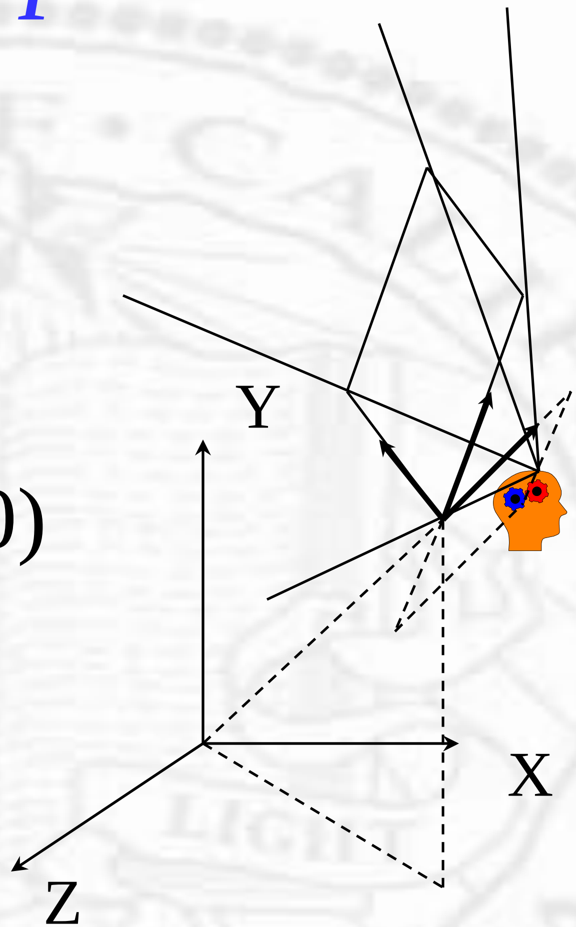
- scale in x and y
- scale in z

$$S_1 = \left(\frac{near}{\frac{right-left}{2}}, \frac{near}{\frac{top-bottom}{2}}, 1\right)$$

$$S_2 = \left(\frac{1}{far}, \frac{1}{far}, \frac{1}{far}\right)$$

Example

$$EYE = (10,10,10)$$
$$CENTER = (0,0,0)$$
$$UP = (0,1,0)$$
$$(right, left) = (20,0)$$
$$(top, bottom) = (20,0)$$
$$F = 1$$
$$B = 10$$



- Translate EYE into the origin

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & -10 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotate EYE to align with the world system

$$w = \frac{e-c}{|e-c|} = \frac{(1,1,1)}{\sqrt{3}}$$

$$u = \frac{UP \times w}{|UP \times w|} = \frac{(0,1,0)\times(1,1,1)}{|(0,1,0)\times(1,1,1)|} = \frac{(1,0,-1)}{\sqrt{2}}$$

$$v = w \times u = \frac{1}{\sqrt{6}}(1,1,1)\times(1,0,-1) = \frac{1}{\sqrt{6}}(-1,2,-1)$$

$$R = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Shear

$$SH = \begin{bmatrix} 1 & 0 & 10 & 0 \\ 0 & 1 & 10 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$a = \frac{\dfrac{left + right}{2}}{near} = 10, b = \frac{\dfrac{top + bottom}{2}}{near} = 10$$

- Scale into canonical volume

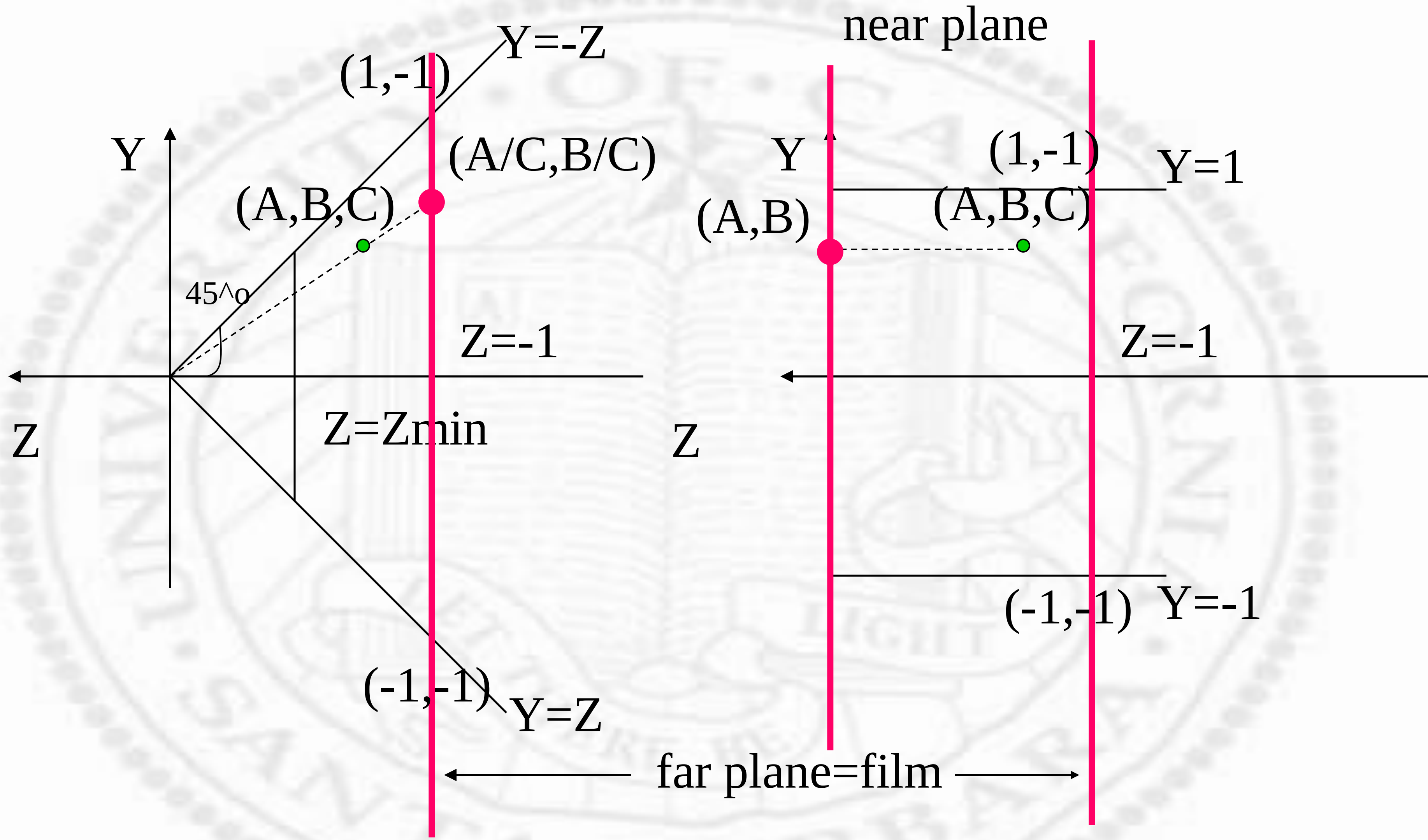
- scale in x and y

$$S_1 = \begin{bmatrix} \frac{1}{10} & 0 & 0 & 0 \\ 0 & \frac{1}{10} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- scale in z

$$S_1 = \begin{bmatrix} \frac{1}{10} & 0 & 0 & 0 \\ 0 & \frac{1}{10} & 0 & 0 \\ 0 & 0 & \frac{1}{10} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Comparing Canonical Volumes



- One additional division is needed in perspective

Matrix Magic

- Try this:
  - Map perspective volume into parallel volume to save the division (note Zmin is NEGATIVE)

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{1}{1+z_{\min}} & \dfrac{-z_{\min}}{1+z_{\min}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$N'_{per} = MN_{per}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{1}{1+z_{\min}} & \dfrac{-z_{\min}}{1+z_{\min}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ z_{\min} \\ z_{\min} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ z_{\min} \\ 0 \\ -z_{\min} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{1}{1+z_{\min}} & \dfrac{-z_{\min}}{1+z_{\min}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -z_{\min} \\ z_{\min} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -z_{\min} \\ 0 \\ -z_{\min} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Normalization Transform

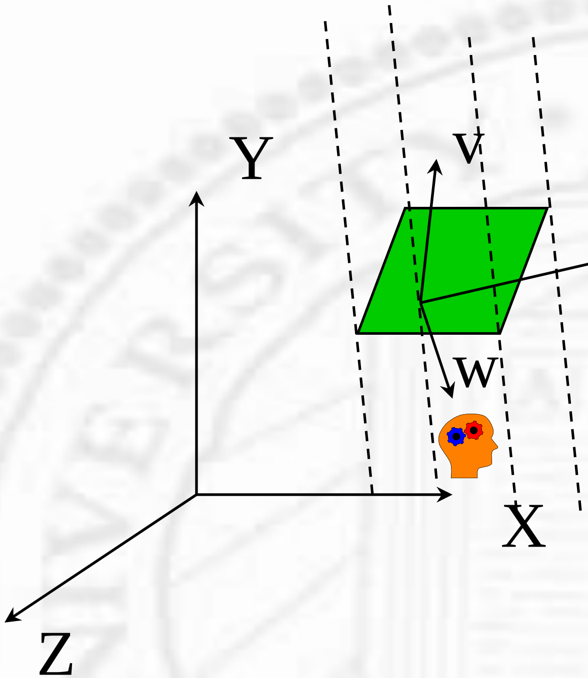
Parallel (othographic) - OpenGL

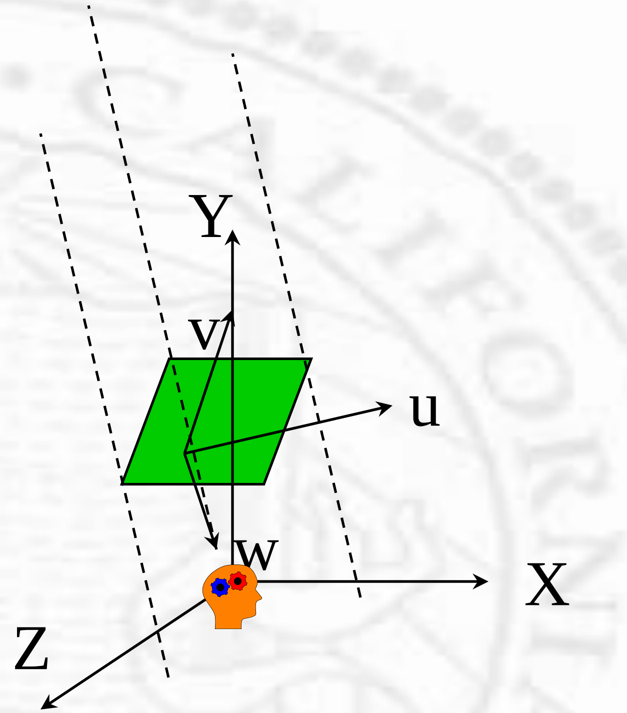
- External parameters
  - Translate EYE into origin
    - *Even though eye is not really where the viewer is*
  - Rotate the EYE coordinate system such that
    - w (e-c) becomes z
    - u becomes x
    - v becomes y
- Internal parameters
  - *Translate* to have centerline of the view volume aligning with z, and near plane at z=0
  - Scale into canonical *rectangular piped*

Viewing Normalization

- Line up (X-Y-Z) and (U-V-W)
- Initially, (U-V-W) are specified in (X-Y-Z) system (In fact, everything is specified in X-Y-Z system)
- Some point in time, want to specify things in (U-V-W) system, or U becomes (1,0,0), V becomes (0,1,0), W becomes (0,0,1)
- Translation (easy) + Rotation (hard)

Translate EYE into the origin

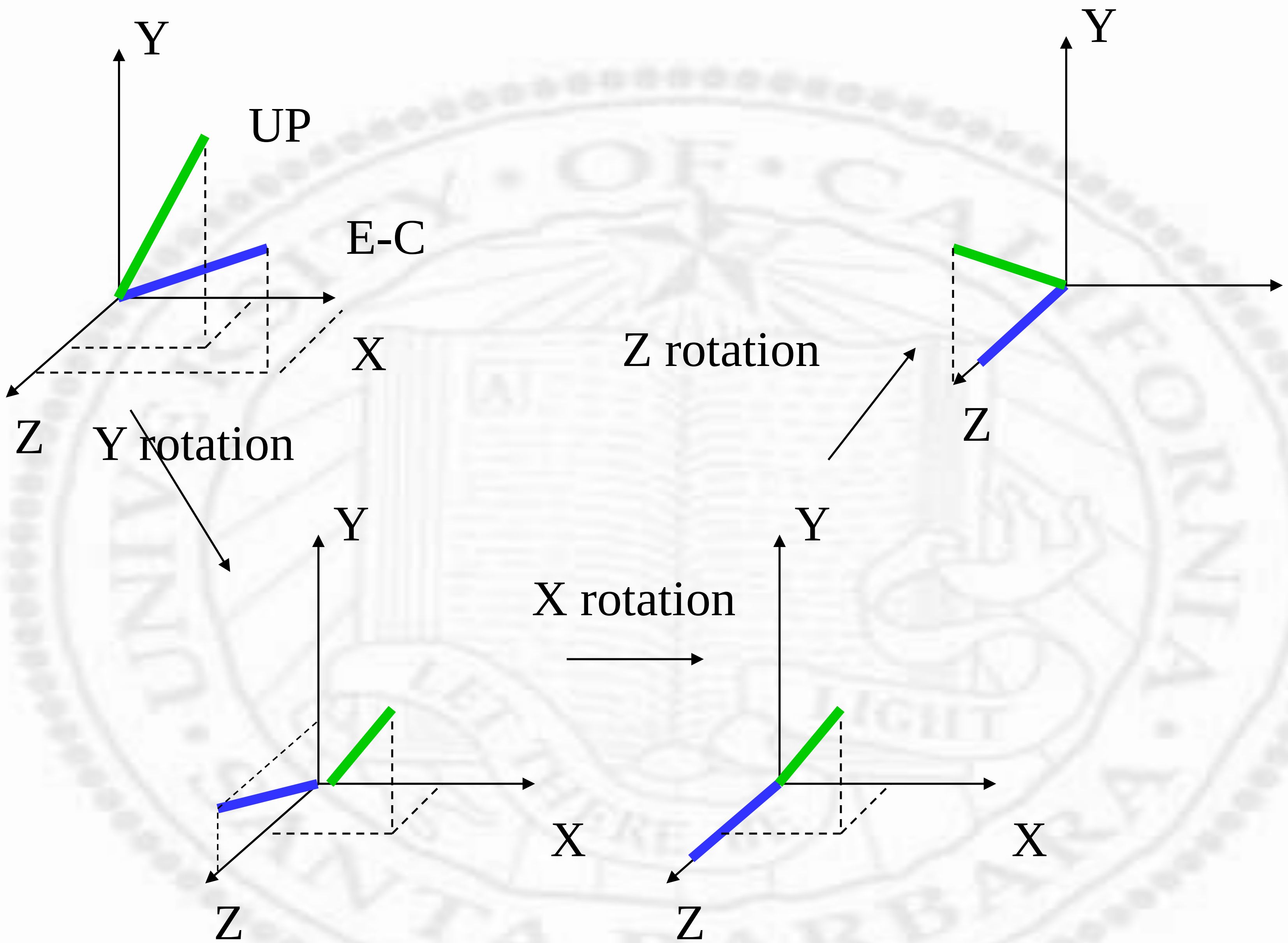




$$T_1 = \begin{bmatrix} 1 & 0 & 0 & -EYE_x \\ 0 & 1 & 0 & -EYE_y \\ 0 & 0 & 1 & -EYE_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewing Normalization

- Three rotations
  - Rotate about Y
  - Rotate about X
  - Rotate about Z


Y
UP
E-C
X
Z
Y rotation
Z rotation
Y
X
Z
Y
X rotation
X
Z
Y
X
Z

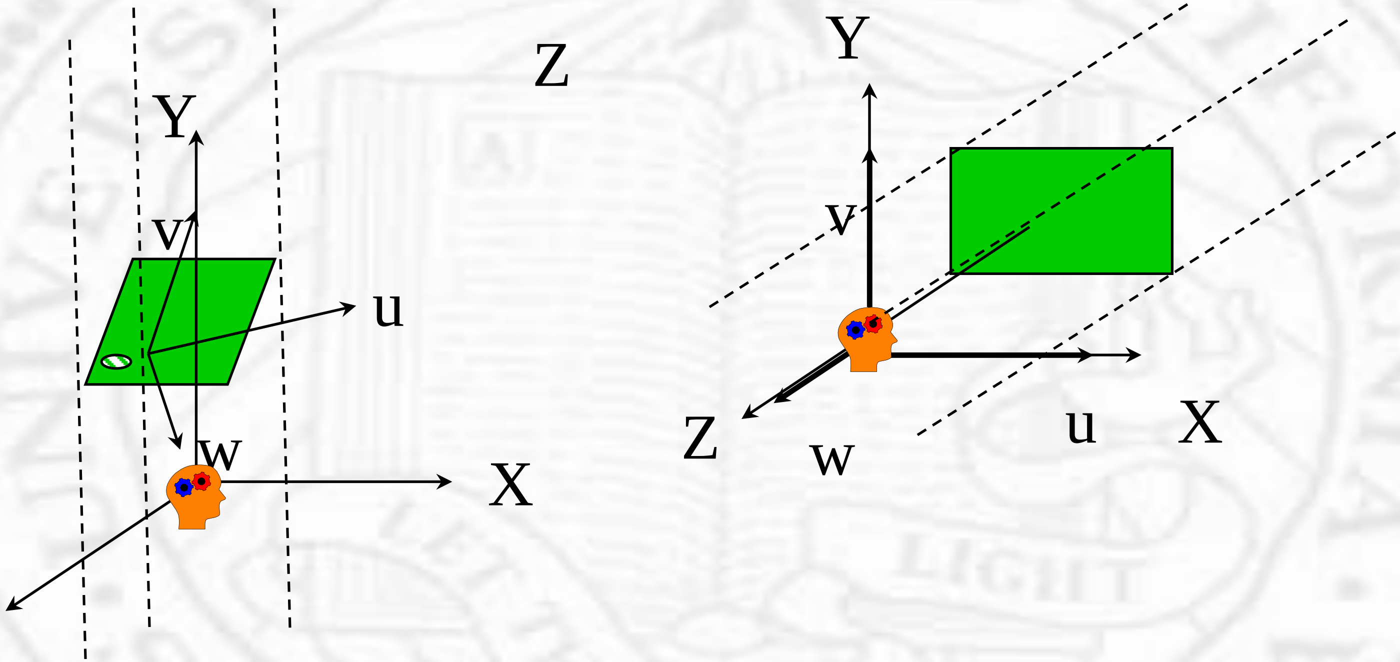
Viewing Normalization

- Figuring out **[u, v, w]** in **[x, y, z]** system
- Applying a rotation to transform **[x, y, z]** coordinates into **[u, v, w]** coordinates

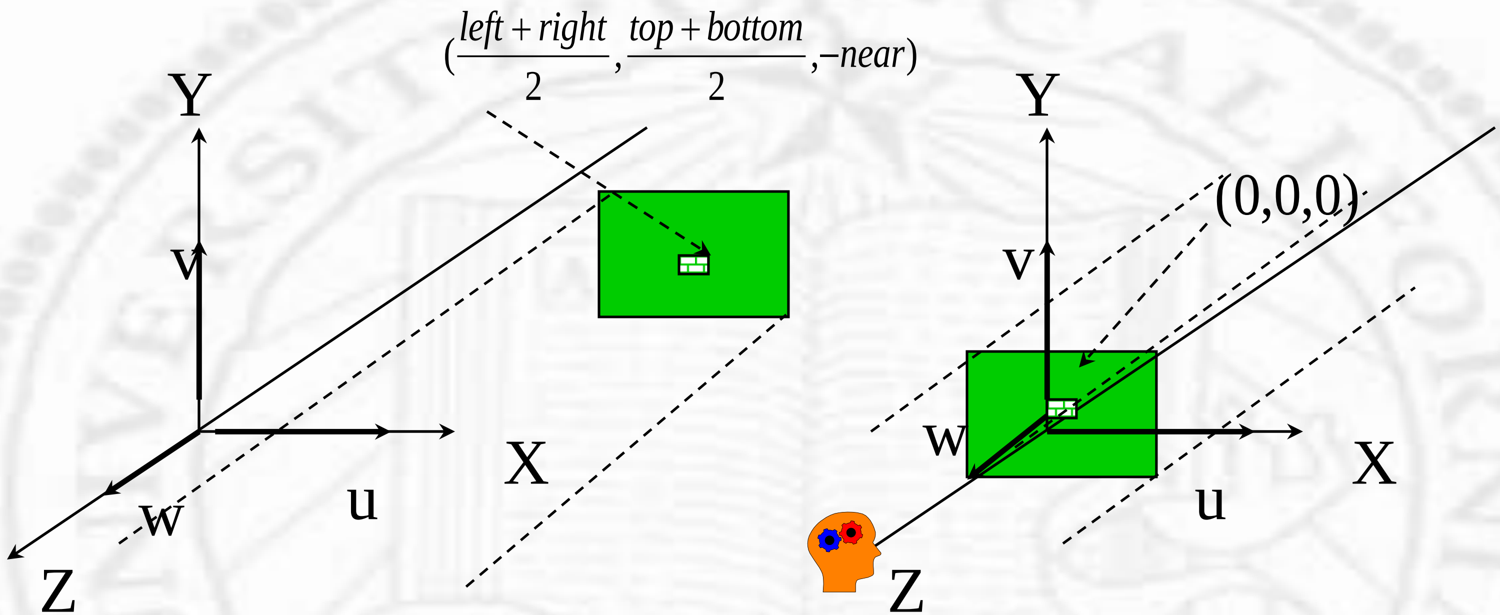
$$w = \frac{e - c}{|e - c|}$$
$$u = \frac{up \times w}{|up \times w|}$$
$$v = w \times u$$

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotate EYE coordinate to align w. world system



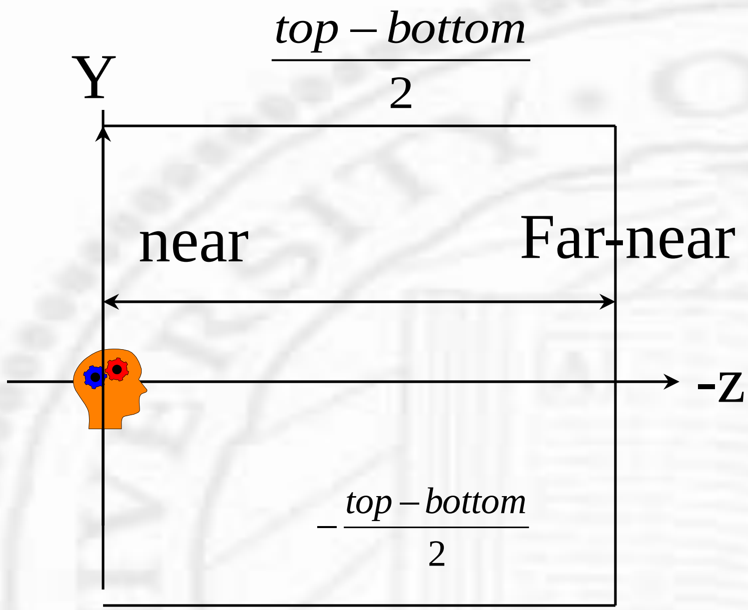
Translation

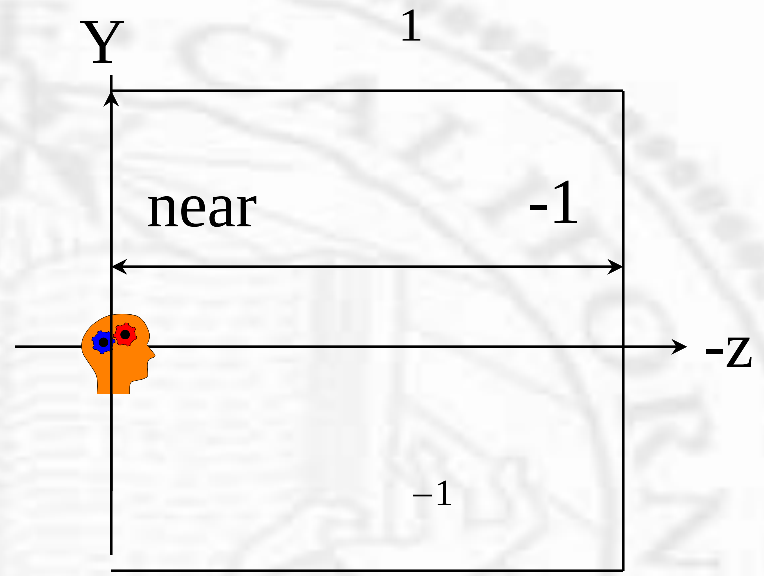


$$T = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$a = -\frac{left + right}{2}, b = -\frac{top + bottom}{2}, c = near$$

Scale into canonical volume



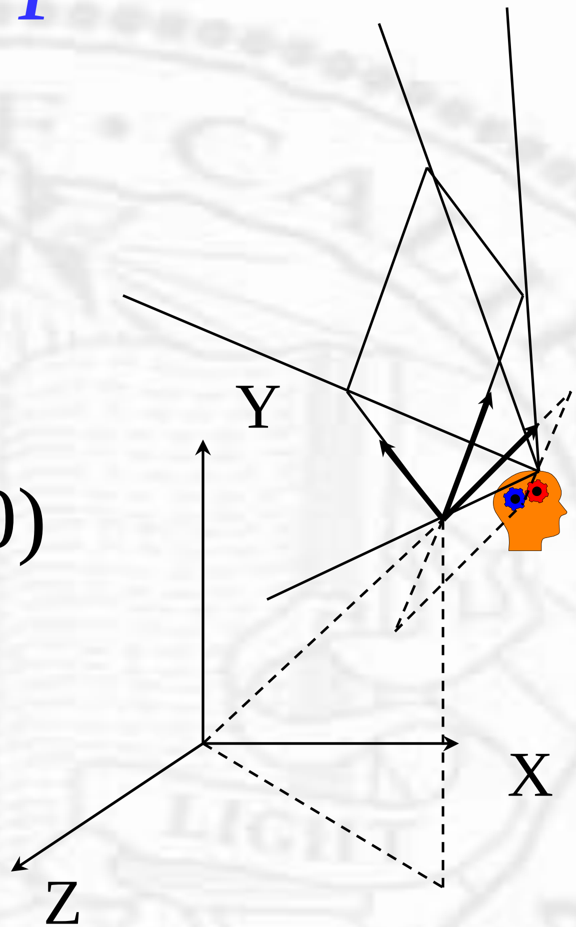


- scale in x, y, and z

$$S = \left(\frac{1}{\frac{right-left}{2}}, \frac{1}{\frac{top-bottom}{2}}, \frac{1}{far-near}\right)$$

Example

$$EYE = (10,10,10)$$
$$CENTER = (0,0,0)$$
$$UP = (0,1,0)$$
$$(right, left) = (20,0)$$
$$(top, bottom) = (20,0)$$
$$F = 1$$
$$B = 10$$



- Translate EYE into the origin

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & -10 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotate EYE to align with the world system

$$w = \frac{e-c}{|e-c|} = \frac{(1,1,1)}{\sqrt{3}}$$

$$u = \frac{UP \times w}{|UP \times w|} = \frac{(0,1,0)\times(1,1,1)}{|(0,1,0)\times(1,1,1)|} = \frac{(1,0,-1)}{\sqrt{2}}$$

$$v = w \times u = \frac{1}{\sqrt{6}}(1,1,1)\times(1,0,-1) = \frac{1}{\sqrt{6}}(-1,2,-1)$$

$$R = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Translation

$$SH = \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & -10 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$a = \frac{\frac{left + right}{2}}{near} = 10, b = \frac{\frac{top + bottom}{2}}{near} = 10$$

- Scale into canonical volume

- scale in x, y, and z

$$S_1 = \begin{bmatrix} \frac{1}{10} & 0 & 0 & 0 \\ 0 & \frac{1}{10} & 0 & 0 \\ 0 & 0 & \frac{1}{9} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$