# Computer Graphics and Programming
# Lecture 3

# Perspective Projection Matrix

Jeong-Yean Yang

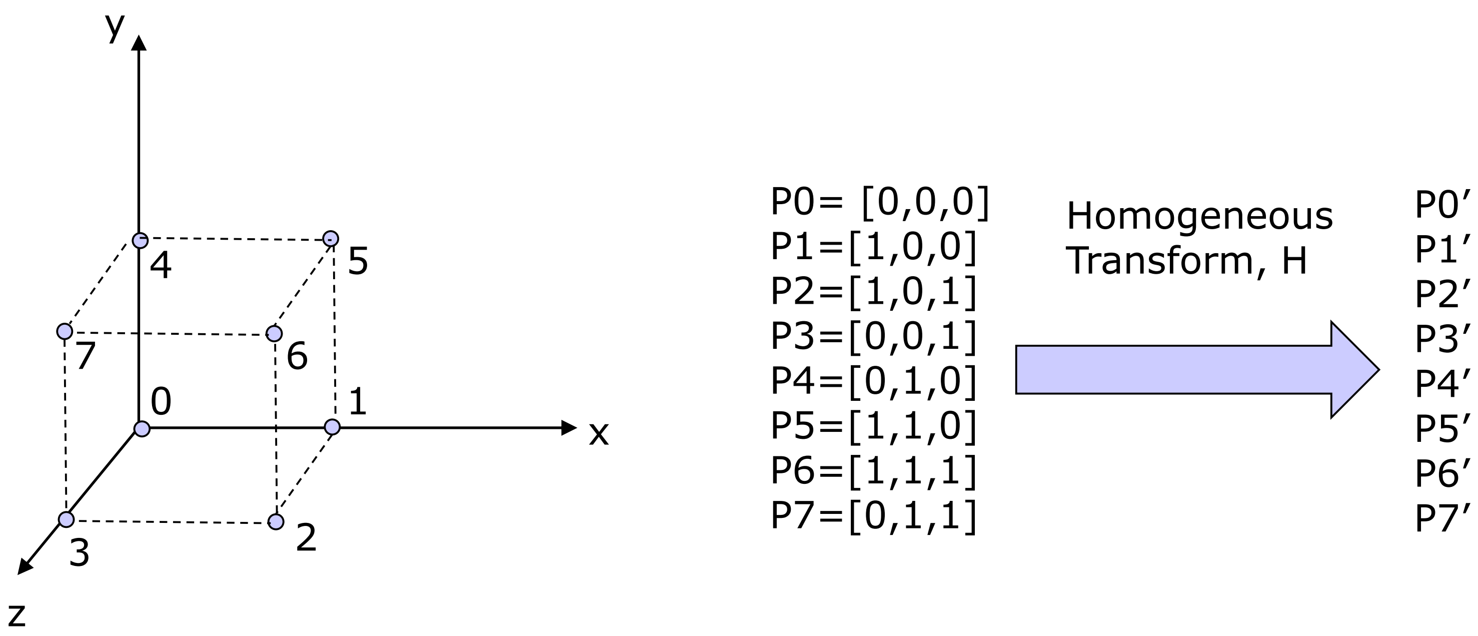
2020/10/22



Dept. of Intelligent Robot Eng. MU

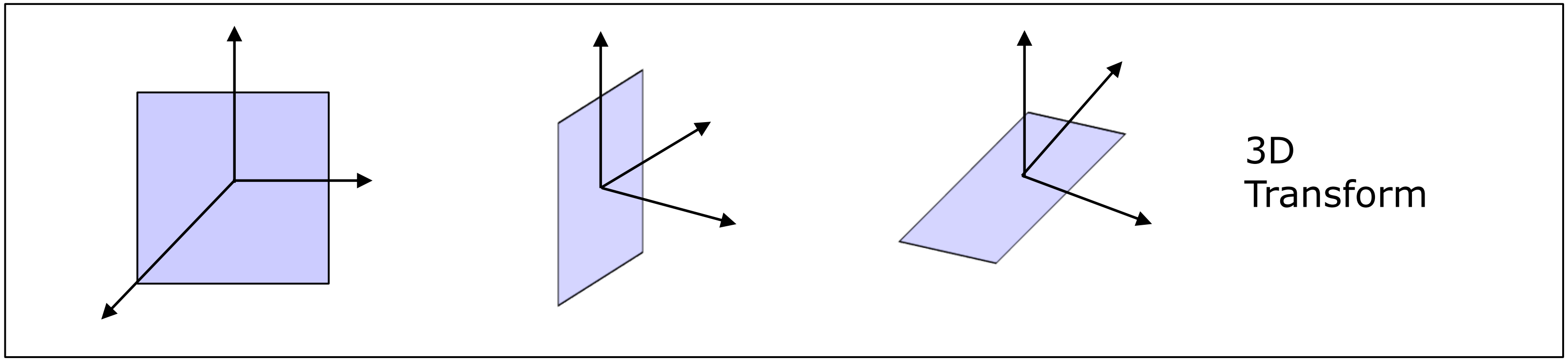
# Projection from 3D into 2D

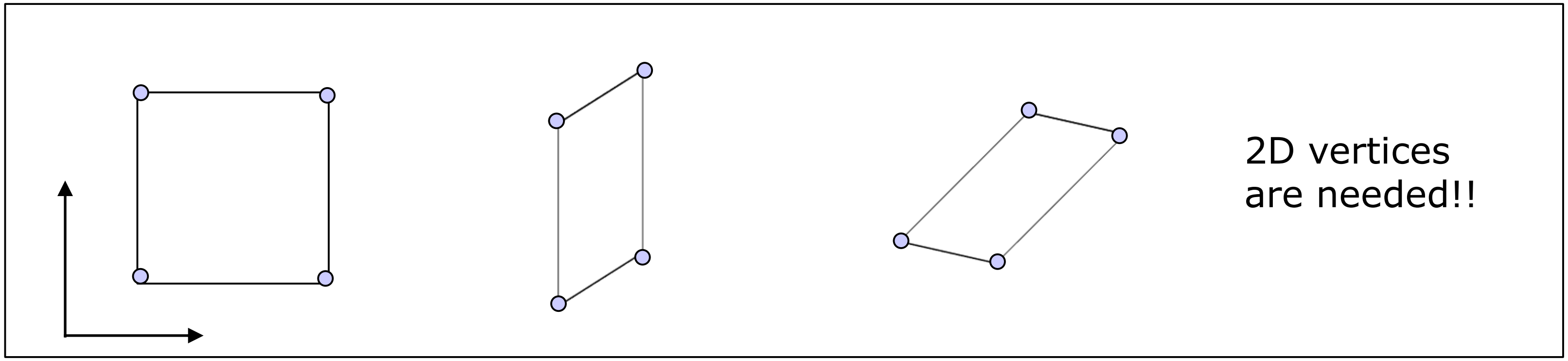
- Think box in 3D




**Question: How we get 2D points for line drawing?**

# Box Projection onto 2D Space

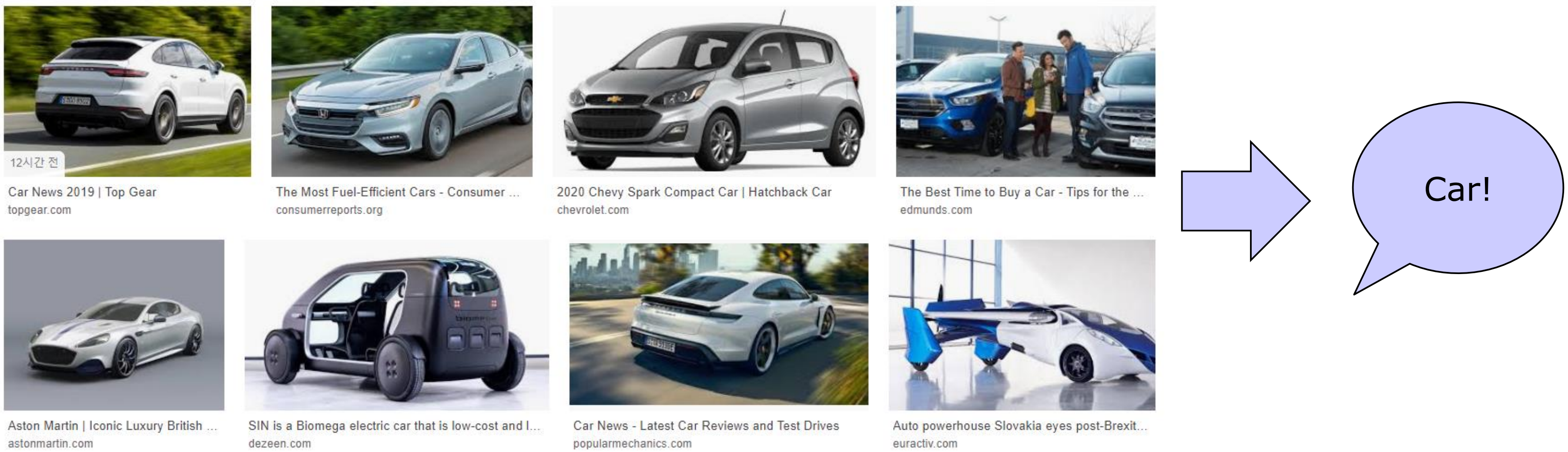



- Projection: Mapping from N dimensional space to N-1 dimensional space

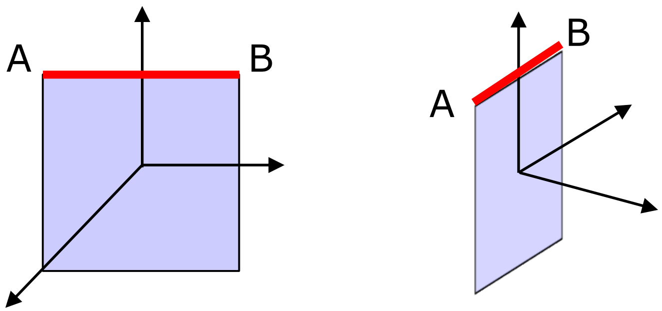

Dept. of Intelligent Robot Eng. MU

# Ref. Projection in Machine Learning

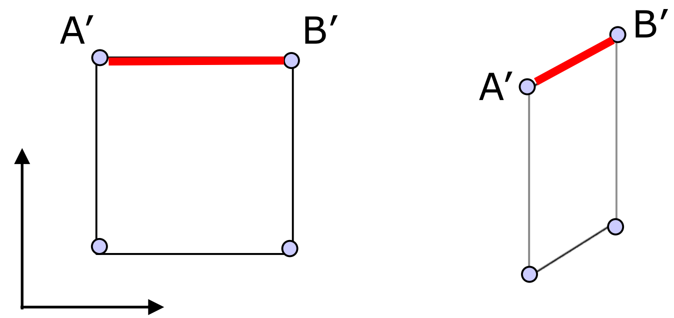



- Information in a High dimensional space is projected onto a lower dimensional space
  - Why? 4 wheels, engine, and so on.
- Starting point for Structuring Hierarchy

# Characteristics of Projection




$$\| A - B \| = \text{const}$$

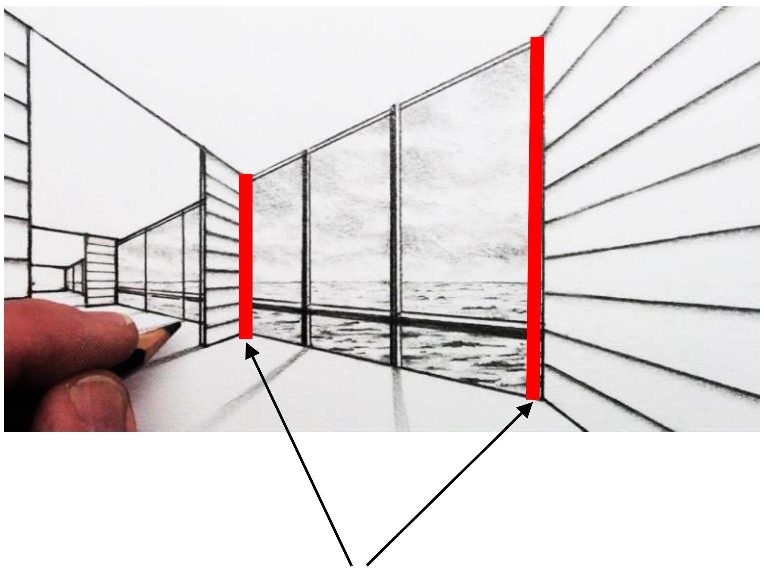



$$\| A' - B' \| \neq \text{const}$$

- 3D Space is a Euclidean Space.
- But, Projected Space is NOT a Euclidean Space.

# Perspective Projection Matrix

- Perspective View



Two lines have Same Distance,
but these **look Different** in Perspective View

Who is the closest one?

It is not clear….
Big head…

# Back to Homogeneous Transform

# **Why we call Homogeneity, 1?**

$$H = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

$$X' = HX = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} X = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} Rx + T \\ 1 \end{bmatrix}$$

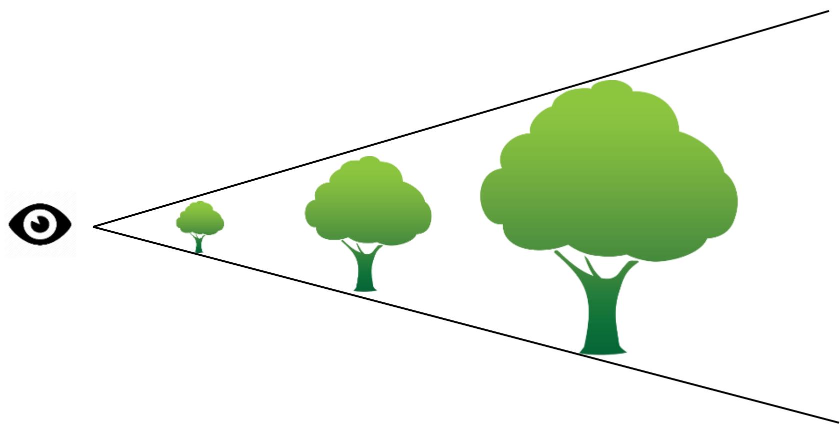
- We learn X has 1, which is needed for Homogeneous transform.
- **Why 1 is required?**

# Homogeneity, 1 for Projection

- In a Homogeneous Space,

$$X \square \begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} 2x \\ 2 \end{bmatrix} = \begin{bmatrix} hx \\ h \end{bmatrix}$$

  - It is NOT a general vector space
  - It is a Affine Space



**Everything looks same!**

# Vector or Point

- Homogeneous Transform requires

$$X = \begin{bmatrix} x \\ h \end{bmatrix} \qquad \text{h=0 or 1}$$

$$X_1 = \begin{bmatrix} x_1 \\ 1 \end{bmatrix}, X_2 = \begin{bmatrix} x_2 \\ 1 \end{bmatrix}$$

$$X_1 - X_2 = \begin{bmatrix} x_1 - x_2 \\ 1-1 \end{bmatrix} = \begin{bmatrix} x_1 - x_2 \\ 0 \end{bmatrix}$$

Point-Point = Vector

$$X_1 = \begin{bmatrix} x_1 \\ 0 \end{bmatrix}, X_2 = \begin{bmatrix} x_2 \\ 0 \end{bmatrix}$$

$$X_1 - X_2 = \begin{bmatrix} x_1 - x_2 \\ 0 \end{bmatrix}$$

Vector-Vector = Vector

# Definition of Vector and Point

$$Vector, V = \begin{bmatrix} x \\ 0 \end{bmatrix}, \qquad \mathrm{Point, P} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

- So, 0 or 1 is the only case?

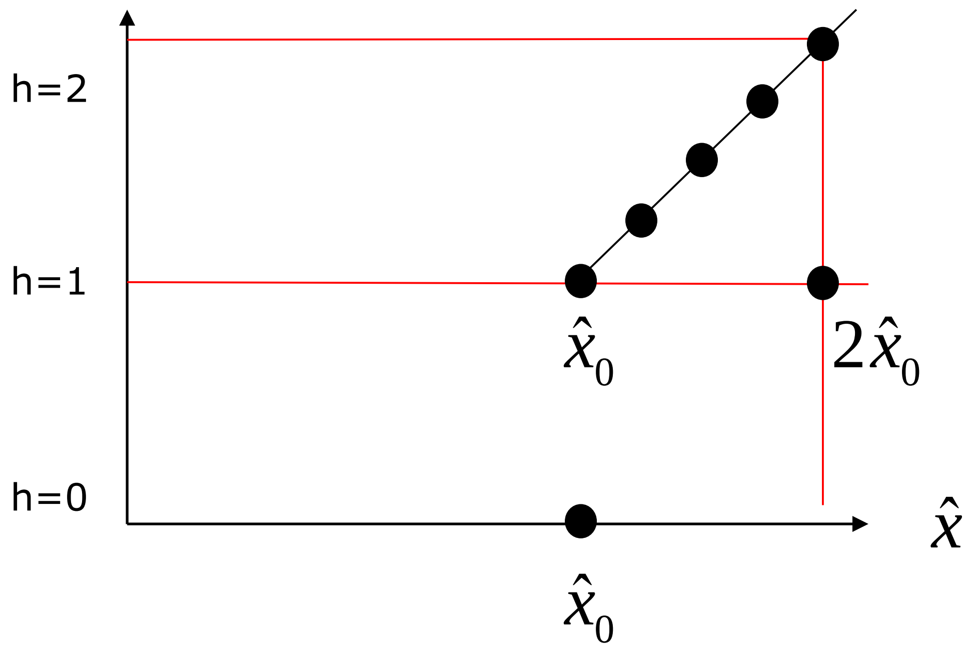
$$\mathrm{P} = \begin{bmatrix} x \\ 0.3 \end{bmatrix}$$

What is it?

- It is a **Point**.

$$\mathrm{P} = \begin{bmatrix} x \\ 0.3 \end{bmatrix} = \begin{bmatrix} x/0.3 \\ 1 \end{bmatrix}$$

# Truth of Homogeneous Transform

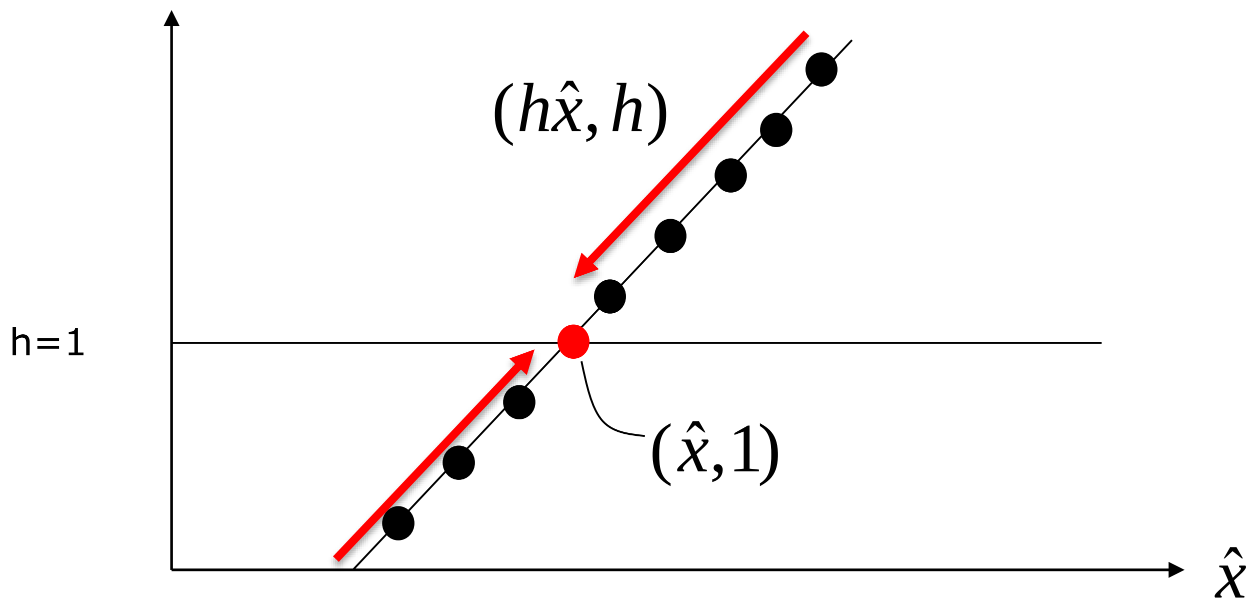



$$X = \begin{bmatrix} \hat{x}_0 \\ h \end{bmatrix}$$

- In a Homogeneous Space,
  - Design this space as,
  - Every Vector is projected on h=0
  - Every point is projected on h=1

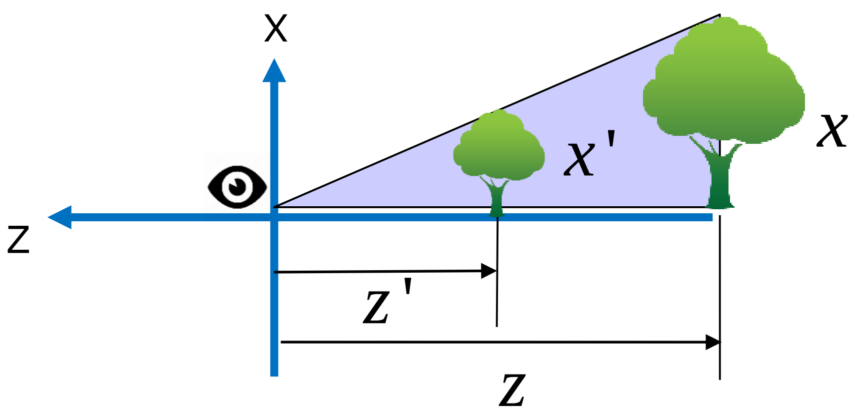
$$X = \begin{bmatrix} \hat{x}_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2\hat{x}_0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.1\hat{x}_0 \\ 0.1 \end{bmatrix}$$

# Homogeneous → Projection on h=1




- All points passing $(\hat{x},1)$ are same.
- All points passing $(h\hat{x}, h)$ are projected on $(\hat{x},1)$

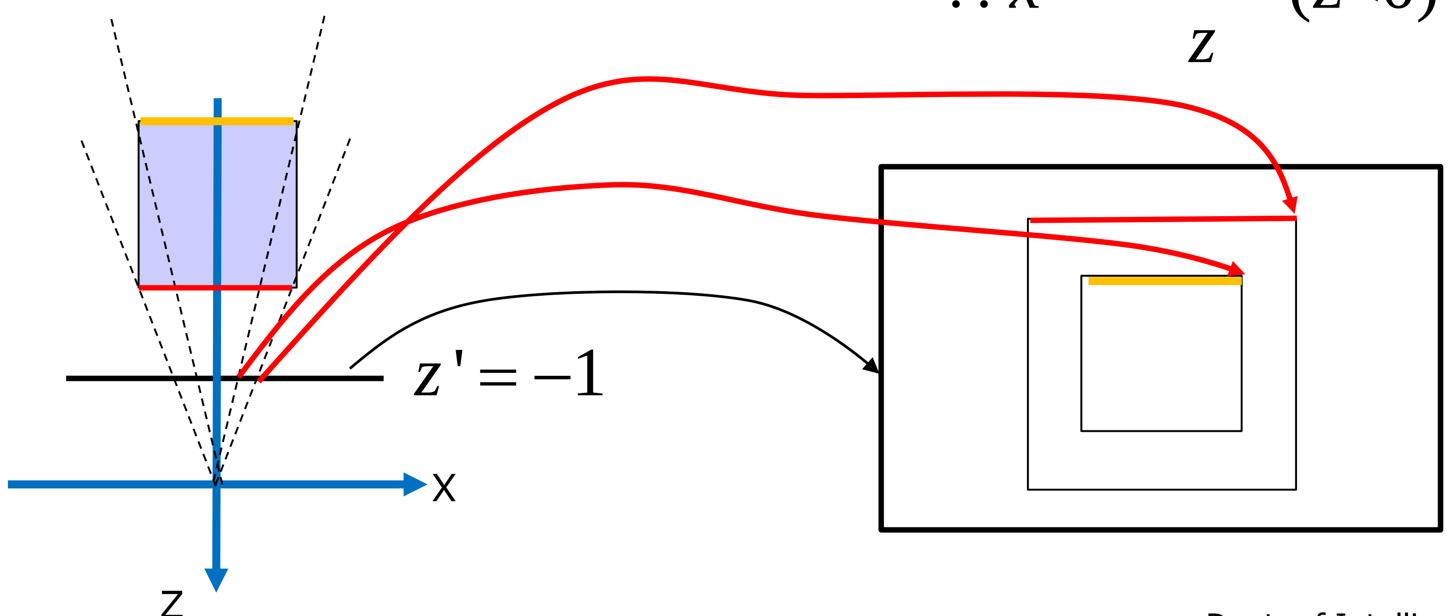
# Example) Perspective View




$$\frac{x'}{z'} = \frac{x}{z} \quad (z', z<0)$$

*if* we define $z' = -1$,

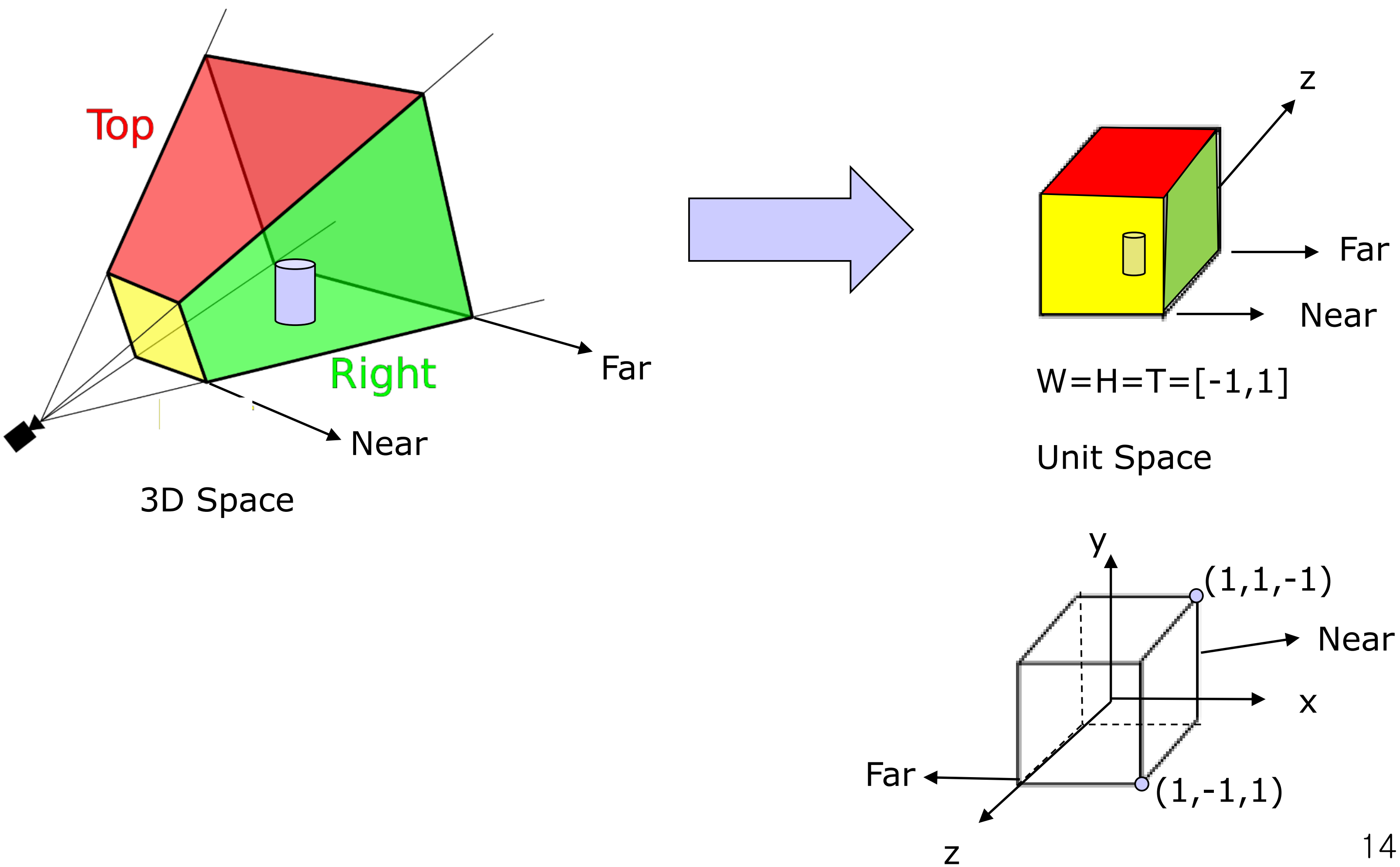
$$\therefore x' = -\frac{x}{z} \quad (z<0)$$






Dept. of Intelligent Robot Eng. MU

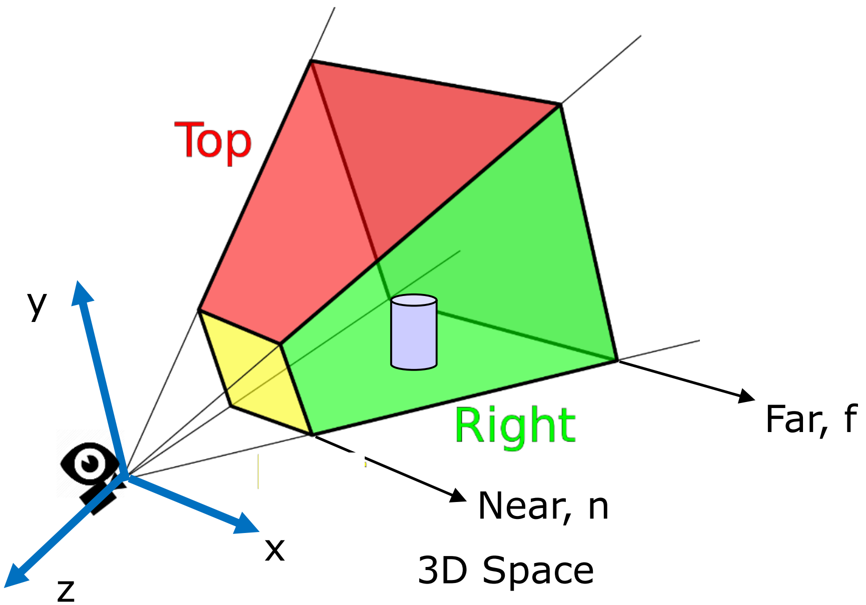
# Modeling of Perspective View uses Viewing Pyramid(Frustum)

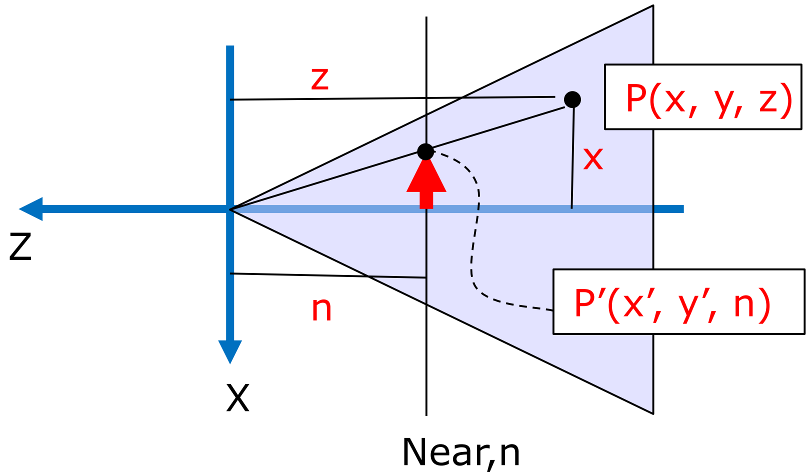




Dept. of Intelligent Robot Eng. MU

# Basic Perspective View to Z axis



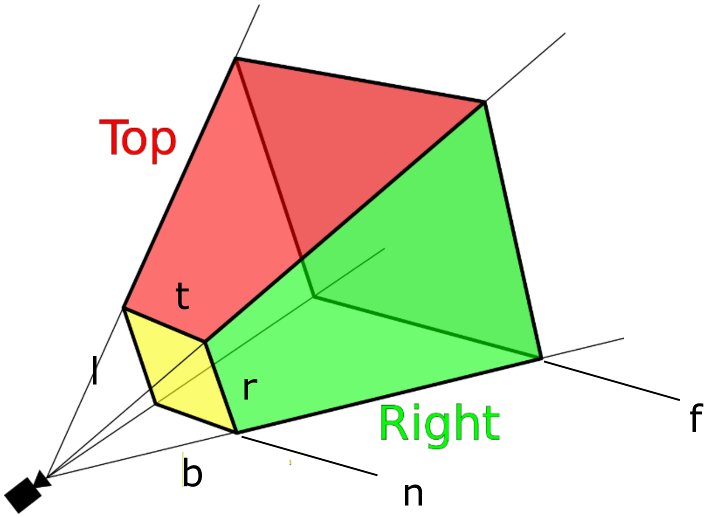

- Calculate P'(x',y',n) on near plane from P(x,y,z)

$$\frac{\uparrow}{n} = \frac{x'}{n} = \frac{x}{z} \quad (n, z < 0) \qquad \therefore x' = n\frac{x}{z}$$

- Top of Pyramid is "Origin", at which the Eye locates.

# Definition of Perspective Projection

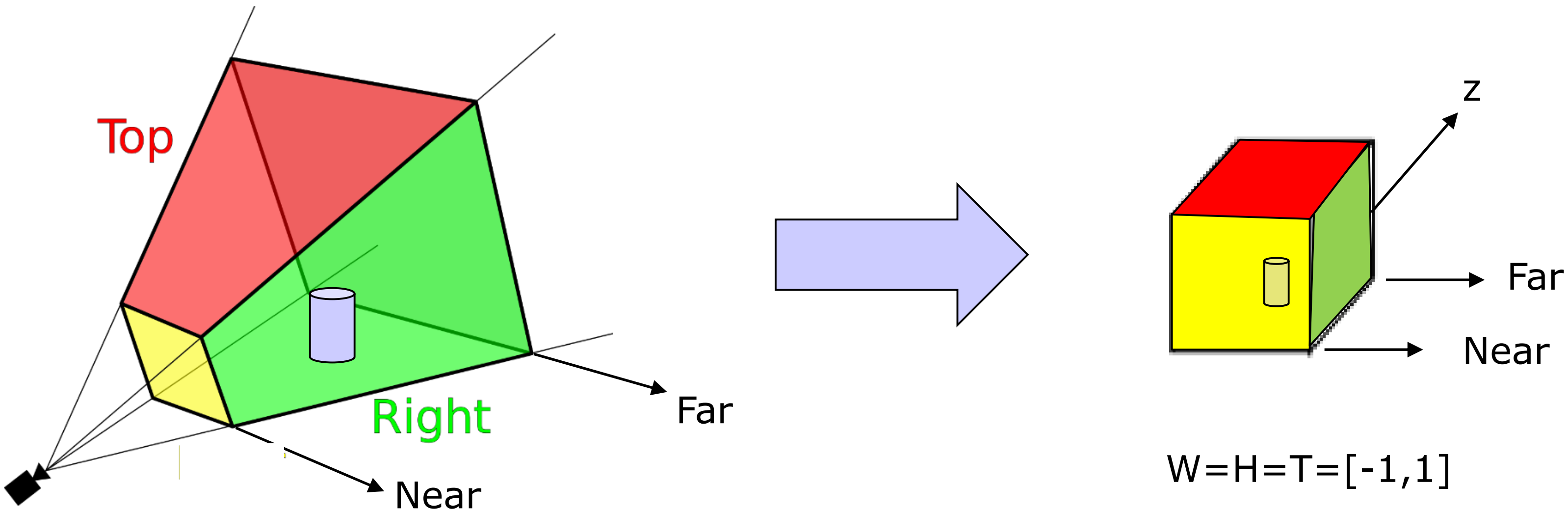
- Perspective Projection
  - Window size of **near plane** is **l, r, t, b.**



# 1. Perspective Mapping

**Objects in the Frustum are mapped into the unit space**

- Recall that




3D Space

W=H=T=[-1,1]

Unit Space

- Near plane : z= -1 (→ n= -1)
- Map x to $\frac{x}{-z}$ and map y to $\frac{y}{-z}$

$$x' = -\frac{x}{z}$$

$pp.\ 13$

# Derivation of Perspective Mapping (Original Derivation)

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} e & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & a & c \\ 0 & 0 & b & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} ex \\ fy \\ az+c \\ bz+d \end{pmatrix} = \begin{pmatrix} \frac{ex}{bz+d} \\ \frac{fy}{bz+d} \\ \frac{az+c}{bz+d} \\ 1 \end{pmatrix}$$

$$a, b, c, d, e, f = ?$$

To map $x$ to $x' = \dfrac{x}{-z} \quad (z < 0)$, $\qquad \dfrac{ex}{bz+d} \rightarrow e = 1, b = -1, d = 0$

To map $y$ to $y' = \dfrac{y}{-z} \quad (z < 0)$, $\qquad \dfrac{fy}{bz+d} \rightarrow f = 1, b = -1, d = 0$

$$\therefore z' = \frac{az+c}{bz+d} = \frac{az+c}{-z}$$

# Derivation of Perspective Mapping (Short Derivation)

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & c \\ 0 & 0 & b & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ az+c \\ bz+d \end{pmatrix} = \begin{pmatrix} \frac{x}{bz+d} \\ \frac{y}{bz+d} \\ \frac{az+c}{bz+d} \\ 1 \end{pmatrix}$$

$$a, b, c, d = ?$$

$$\text{To map } x \text{ to } x' = \frac{x}{-z}, \qquad \frac{x}{bz+d} \rightarrow b = -1, d = 0$$

$$\text{To map } y \text{ to } y' = \frac{y}{-z}, \qquad \frac{y}{bz+d} \rightarrow b = -1, d = 0$$

$$\therefore z' = \frac{az+c}{bz+d} = \frac{az+c}{-z}$$

# Think Z in Near Plane and Far plane

$$\therefore z' = \frac{az+c}{bz+d} = \frac{az+c}{-z} \quad (z<0)$$

$$if \;\; n, f > 0$$

$$if \;\; z = -n, \text{ then } z' = -1$$

$$\therefore z' = -1 = \frac{-an+c}{n}$$

$$-an + c = -n$$

$$if \;\; z = -f, \text{ then } z' = 1$$

$$\therefore z' = 1 = \frac{-af+c}{f}$$

$$f = -af + c$$

$$f = -af - n + an$$

$$\therefore a = \frac{n+f}{n-f}$$

$$c = (a-1)n$$

$$\therefore c = \left(\frac{n+f}{n-f} - 1\right) n = \frac{2nf}{n-f}$$

# Perspective Matrix

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & c \\ 0 & 0 & b & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{n+f}{n-f} & \dfrac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$Perspective\ Matrix = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{n+f}{n-f} & \dfrac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

# Normalization of Perspective Matrix Result

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n-f} + \frac{2nf}{n-f} \\ -z \end{pmatrix}$$

$$= \begin{pmatrix} \frac{x}{-z} & \frac{y}{-z} & \frac{z(n+f)+2nf}{-z(n-f)} & 1 \end{pmatrix}^T \quad (z < 0)$$

# See uWnd-19-3D-Perspective-Projection

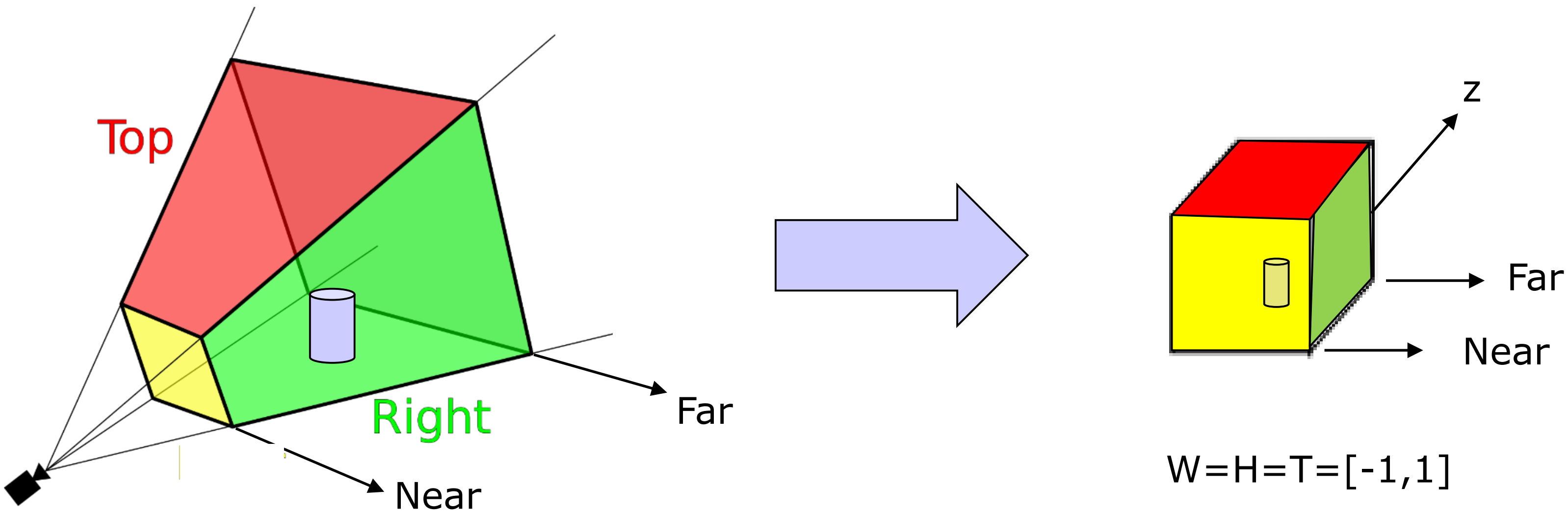
- uObj::Draw line 39

$$\hat{x}' = P\hat{x} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n-f} + \frac{2nf}{n-f} \\ -z \end{pmatrix} \quad (z<0)$$

$$= \begin{pmatrix} \frac{x}{-z} & \frac{y}{-z} & \frac{z(n+f)+2nf}{-z(n-f)} & 1 \end{pmatrix}^T \quad (z<0)$$

```
// transform vertex
for (i=0;i<nMax;i++)
{
    temp[i] = H*vertex[i];

    float z = temp[i].z;
    temp[i] = P*temp[i];
    temp[i].x   = -temp[i].x/z;
    temp[i].y   = -temp[i].y/z;
    temp[i].z   = -temp[i].z/z;

    temp[i] = S*temp[i];
}
```

# Example 1.
## Calculation of Perspective Projection Mapping




3D Space

Unit Space

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

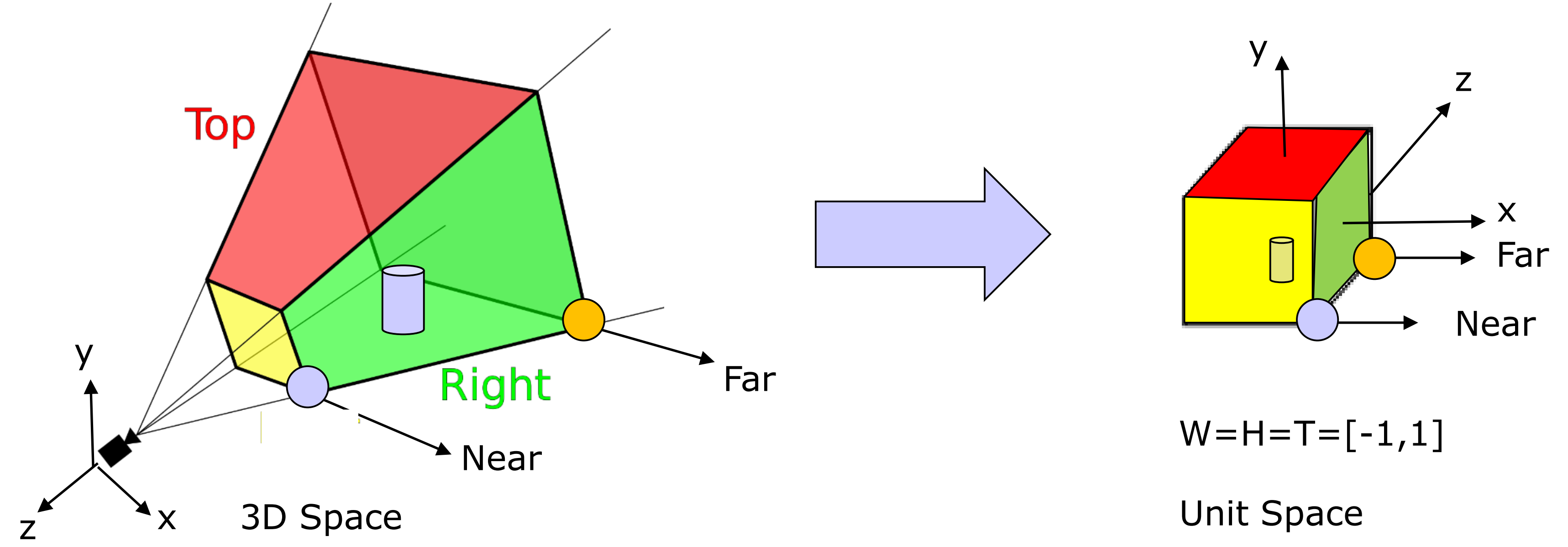
$n, f > 0$

*Assume* that

$n = 1$

$f = 65535$

$$P \cong \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$




$$X = (1, -1, -1, 1)^T$$

$$X = PX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ \underline{-1} \\ 1 \end{pmatrix}$$

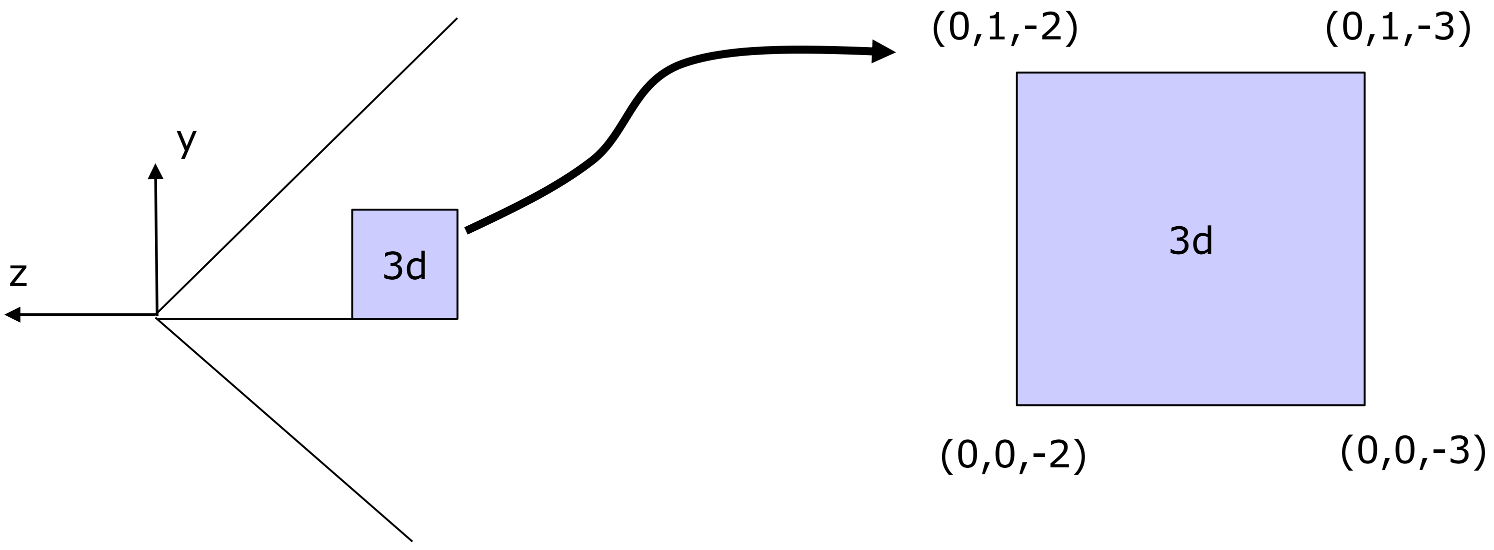
$$X = (65535, -65535, -65535, 1)^T$$

$$= (f, -f, -f, 1)^T$$

$$X = PX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} f \\ -f \\ -f \\ 1 \end{pmatrix} = \begin{pmatrix} f \\ -f \\ f-2 \\ f \end{pmatrix}$$

$$= \begin{pmatrix} \frac{f}{f} & \frac{-f}{f} & \frac{f-2}{f} & 1 \end{pmatrix}^T \cong (1 \quad -1 \quad \underline{1} \quad 1)^T$$

# Example 2.
# From 3D to 2D



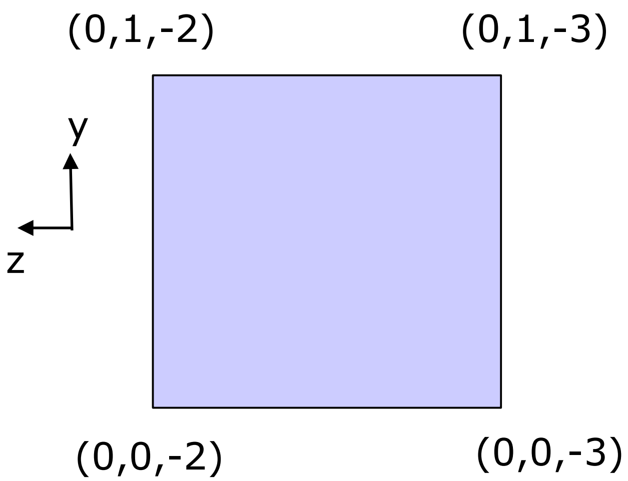

$$P \cong \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$P\begin{pmatrix}0 & 1 & -2 & 1\end{pmatrix}^T = \begin{pmatrix}0 & 1 & 0 & 2\end{pmatrix}^T$$

$$P\begin{pmatrix}0 & 0 & -2 & 1\end{pmatrix}^T = \begin{pmatrix}0 & 0 & 0 & 2\end{pmatrix}^T$$

$$P\begin{pmatrix}0 & 1 & -3 & 1\end{pmatrix}^T = \begin{pmatrix}0 & 1 & 1 & 3\end{pmatrix}^T$$

$$P\begin{pmatrix}0 & 0 & -3 & 1\end{pmatrix}^T = \begin{pmatrix}0 & 0 & 1 & 3\end{pmatrix}^T$$




*pp.*23

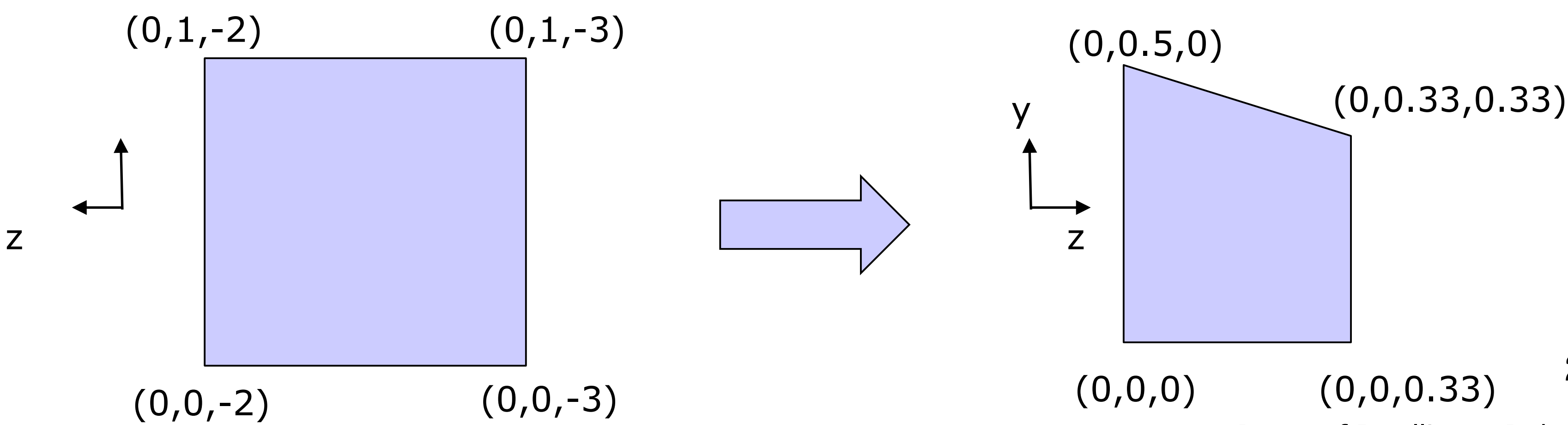
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n-f} + \frac{2nf}{n-f} \\ -z \end{pmatrix} \quad (z < 0)$$

$$P\begin{pmatrix} 0 & 1 & -2 & 1 \end{pmatrix}^T = \begin{pmatrix} 0 & 1 & 0 & 2 \end{pmatrix}^T = \begin{pmatrix} 0 & 0.5 & 0 & 1 \end{pmatrix}^T \quad (z = \text{-}2 < 0)$$

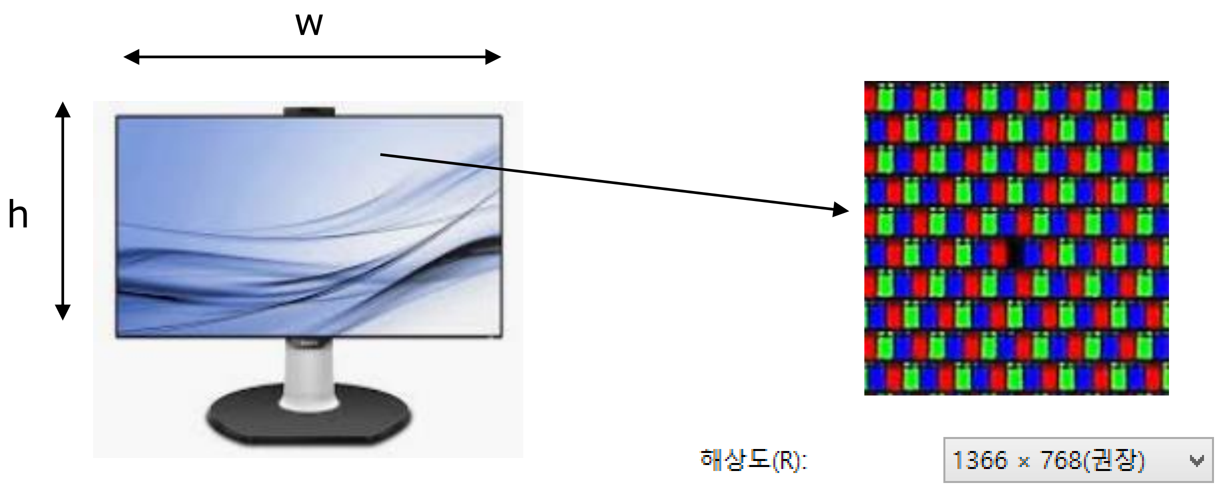
$$P\begin{pmatrix} 0 & 0 & -2 & 1 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 0 & 2 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T \quad (z = \text{-}2 < 0)$$

$$P\begin{pmatrix} 0 & 1 & -3 & 1 \end{pmatrix}^T = \begin{pmatrix} 0 & 1 & 1 & 3 \end{pmatrix}^T = \begin{pmatrix} 0 & 0.33 & 0.33 & 1 \end{pmatrix}^T \quad (z = \text{-}3 < 0)$$

$$P\begin{pmatrix} 0 & 0 & -3 & 1 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 1 & 3 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 0.33 & 1 \end{pmatrix}^T \quad (z = \text{-}3 < 0)$$



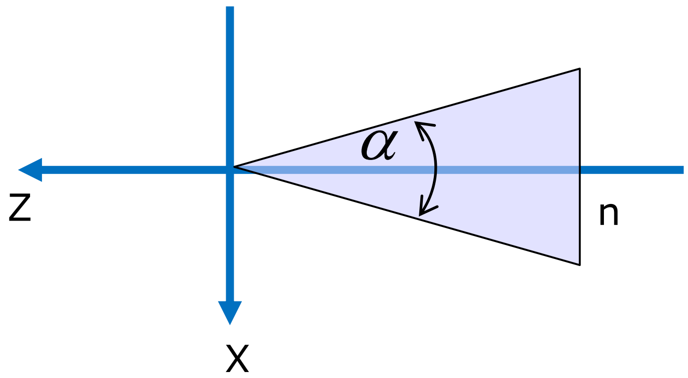
# 2. Aspect Ratio

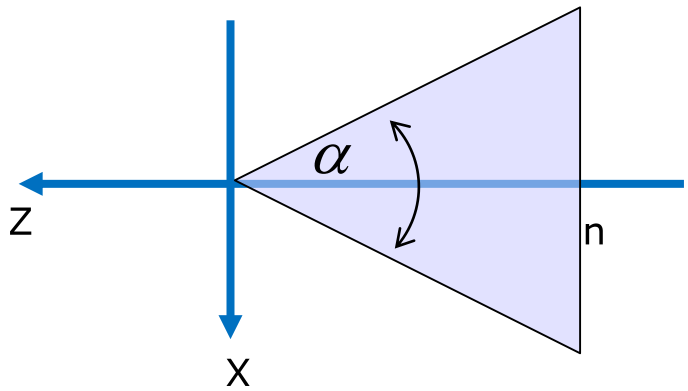



It is NOT a Square!

- Aspect ratio = width/height
  - Ex) AR = 1366/768

# 3. Field of View (FOV)




$$x' = \frac{n}{r-l}x = x\cot\frac{\alpha}{2}$$

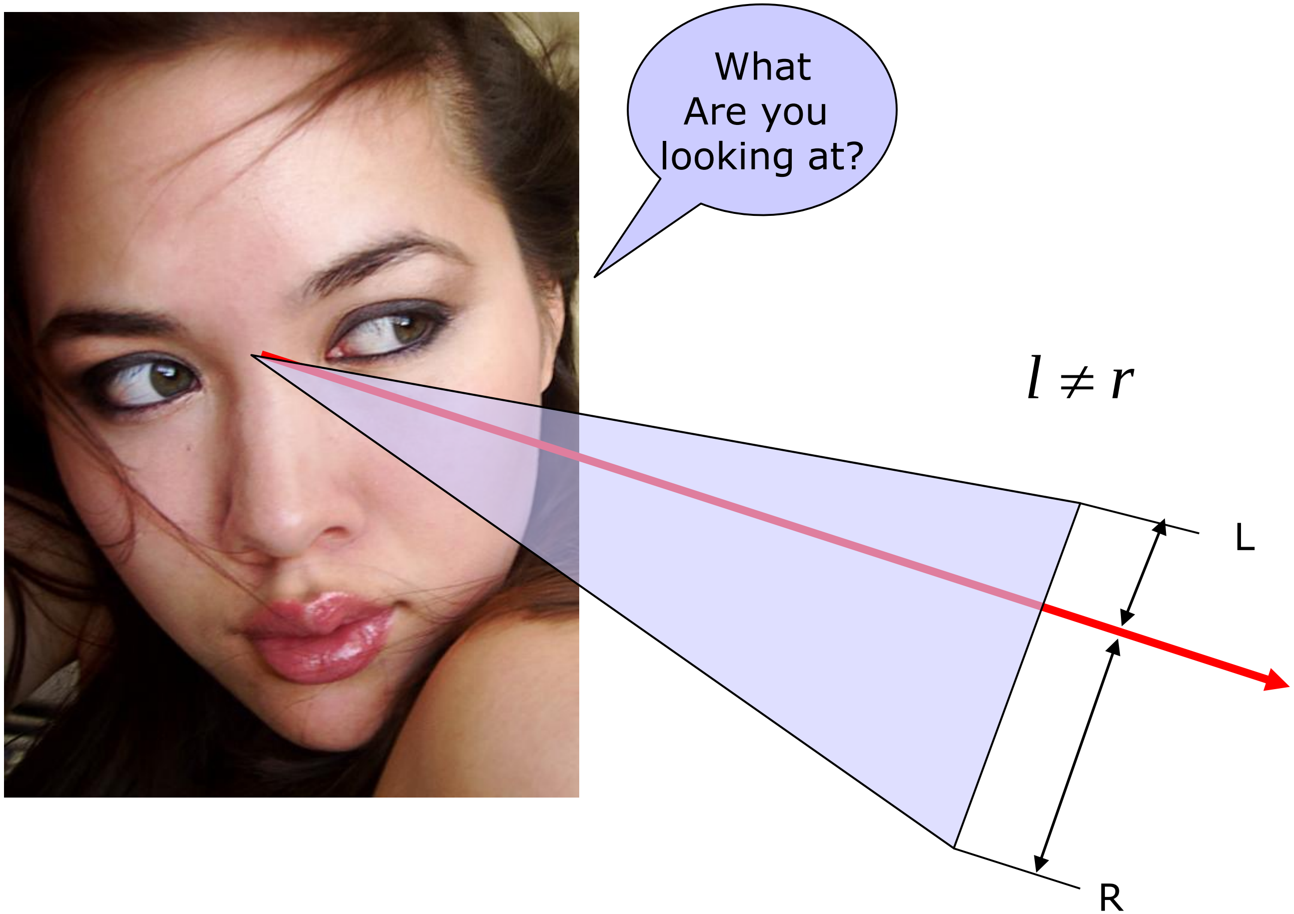
$$y' = \frac{n}{t-b}y = y\cot\frac{\alpha}{2}$$

# 4. Aspect Ratio and FOV are applied

$$P = \begin{pmatrix} \frac{\cot(\alpha/2)}{W/H} & 0 & 0 & 0 \\ 0 & \cot(\alpha/2) & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

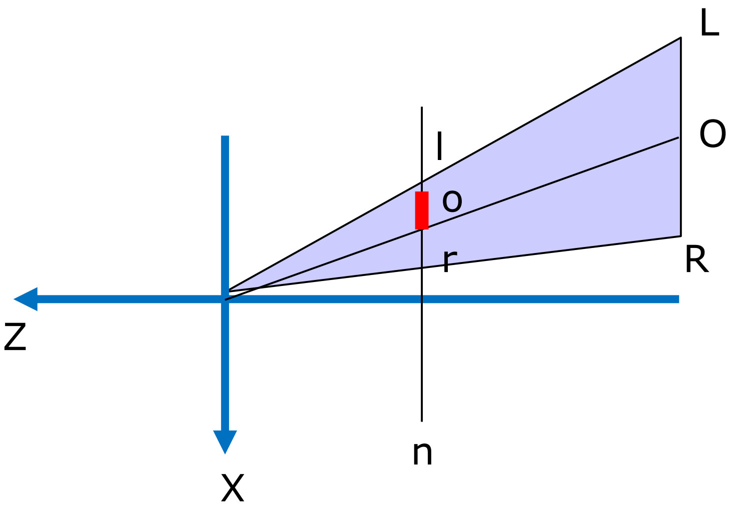
- This matrix is same with gluPerspective in OpenGL
  - n = 1 and f=65535
- Keep in mind → gluPerspective is deprecated in OpenGL ES.

# Special Cases: Condition that Frustum is Skewed, Shearing and Clipping are Needed.




- Skewed projection becomes Popular in VR.

# 5. Shearing Window to Z axis



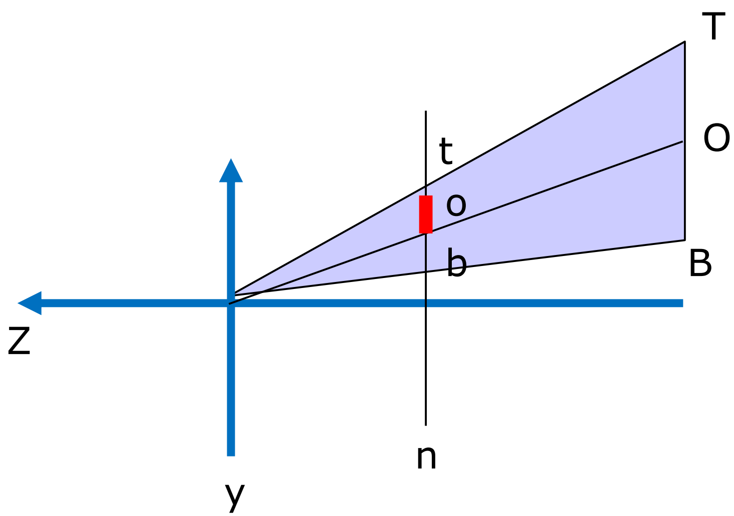

$$\frac{l}{n}=\frac{L}{z},\ \frac{r}{n}=\frac{R}{z}$$

$$o=\frac{l+r}{2}=n\frac{\frac{L}{z}+\frac{R}{z}}{2}=\frac{n}{z}\frac{L+R}{2}=\frac{n}{z}O$$

$$x' \,\square\, x-|O|$$

$$x'=x+O=\mathrm{x}+\frac{z}{n}o=x+\frac{l+r}{2}\frac{z}{n}$$

# 5. Shearing Window to Z axis




$$\frac{t}{n} = \frac{T}{z}, \frac{b}{n} = \frac{B}{z}$$

$$o = \frac{t+b}{2} = n\frac{\frac{T}{z} + \frac{B}{z}}{2} = \frac{n}{z}\frac{T+B}{2} = \frac{n}{z}O$$

$$y' \square\ y - |O|$$

$$y' = y + O = y + \frac{z}{n}o = y + \frac{t+b}{2}\frac{z}{n}$$

# 5. Shearing Window Matrix

$$x' = x + O_x = x + \frac{z}{n} o_x = x + \frac{l+r}{2} \frac{z}{n}$$

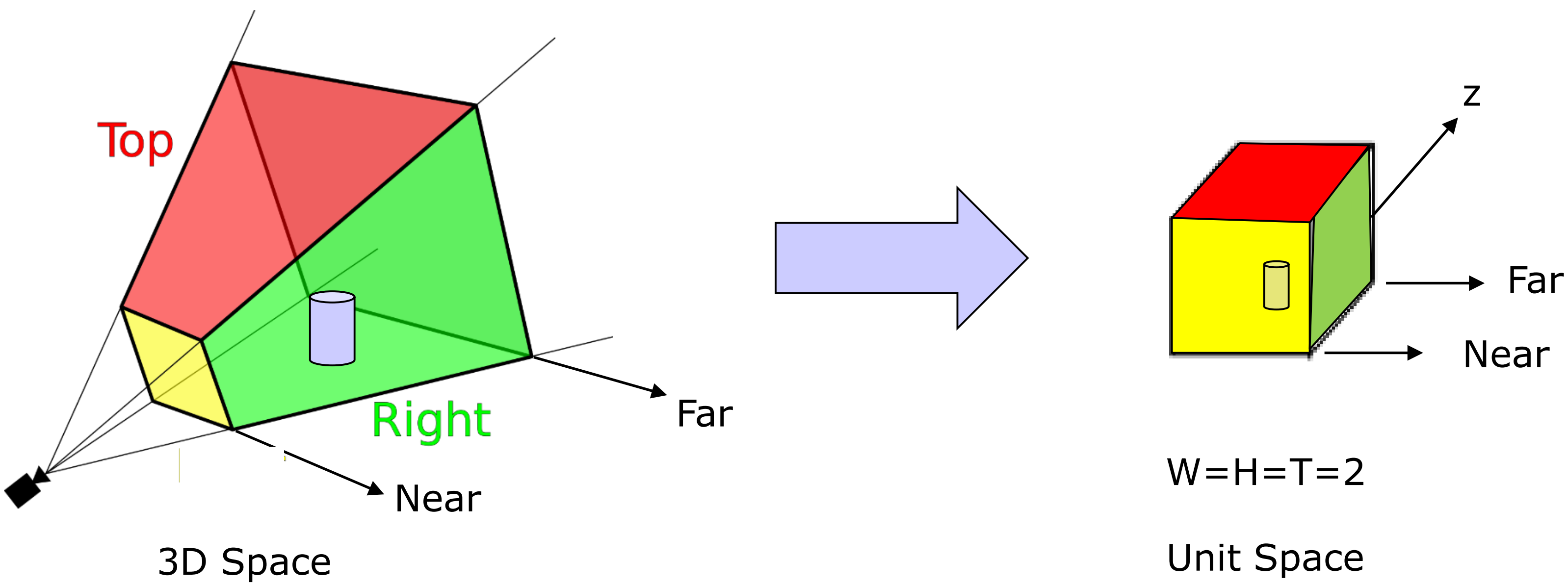
$$y' = y + O_y = y + \frac{z}{n} o_y = y + \frac{t+b}{2} \frac{z}{n}$$

$$z' = z$$

- Shearing Window Matrix in Homogeneous Transform

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{l+r}{2n} & 0 \\ 0 & 1 & \frac{t+b}{2n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

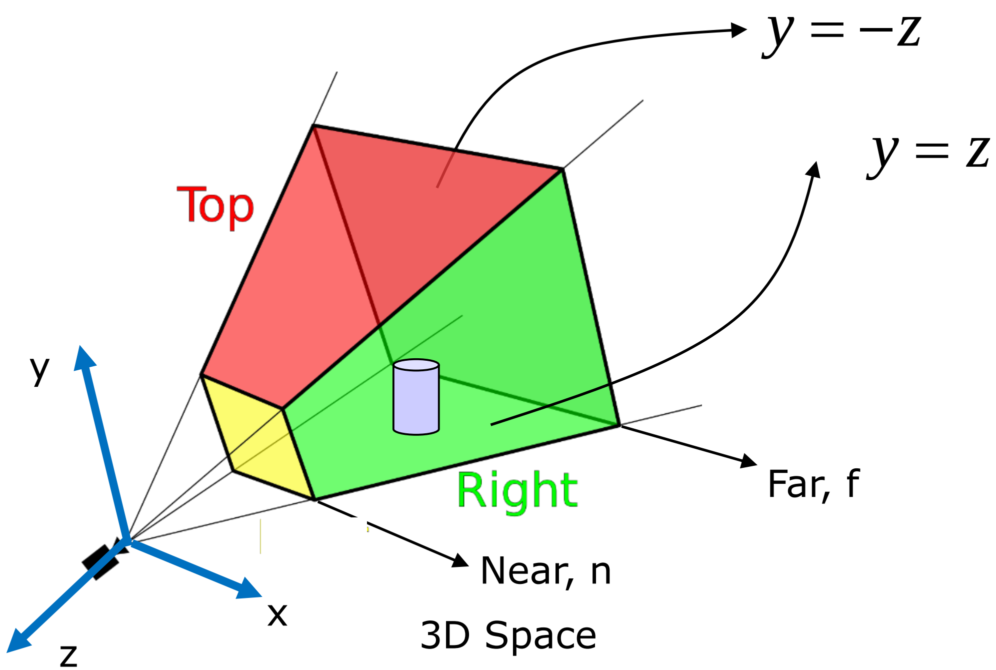
# 6. Clipping Boundaries




- Clipping object and Mapping intro Unit space
  - Width=[-1,1]
  - Height=[-1,1]
  - Thickness=[-1,1]

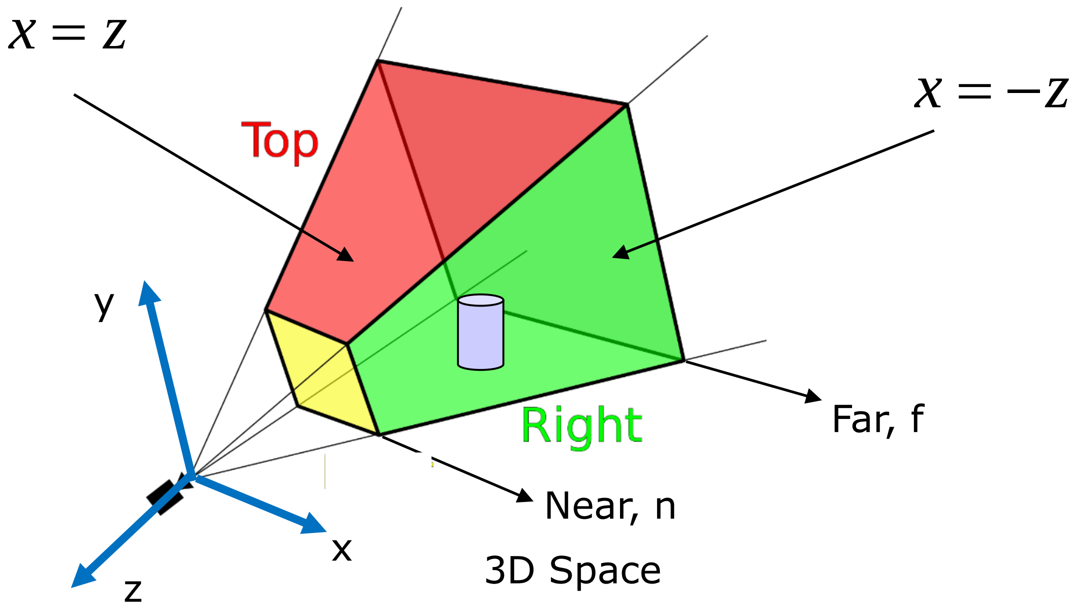
# 6. Clipping Boundaries

- Think plane



# 6. Clipping Boundaries

- Think plane



# 6. Clipping Boundaries

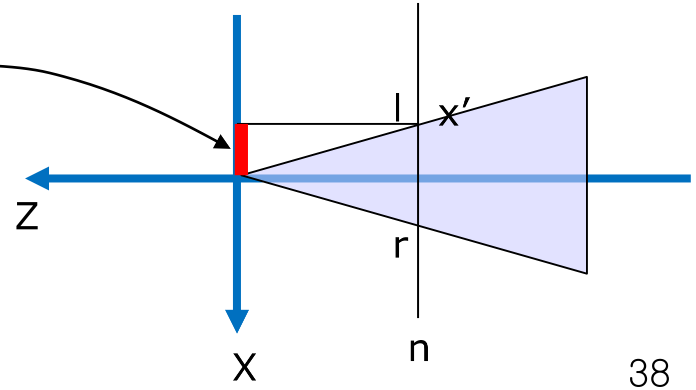
- Clipping boundaries into Unit Space [-1,1]

$$x' = \frac{2n}{r-l}x \qquad y' = \frac{2n}{t-b}y \qquad z' = z$$

$$r = 1,\quad l = -1,\quad t = 1,\quad b = -1,\quad n = 1$$

$$x' = x \qquad y' = y \qquad z' = z$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



# 7. Skewed Perspective Projection matrix

$$P = Perspective \times Clipping \times Shearing$$

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

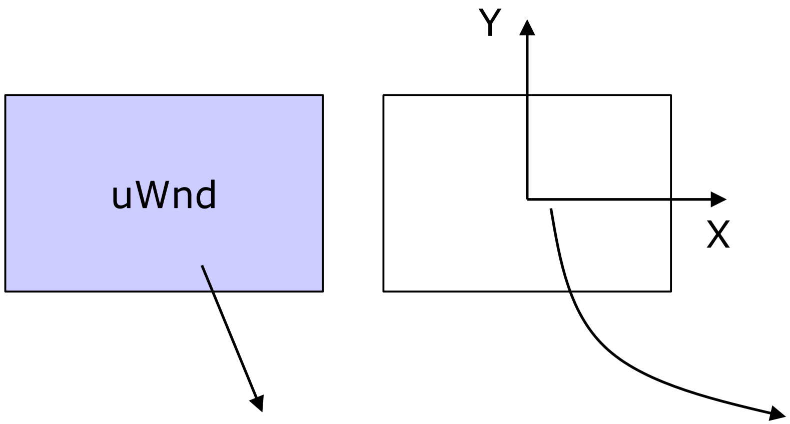
$$= \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Skewed Perspective Projection

$$\begin{aligned} r &= 1 \\ l &= -1 \\ t &= 1 \\ b &= -1 \\ n &= 1 \end{aligned} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Perspective Projection

# Ex) uWnd-19-3D-Perspective-Projection




hMat P : Perspective Projection

```
float n=1;
float f=65535;
float angle     = 90;
float aspect    = 1366./768;

float z1 = (n+f)/(f-n);
float z2 = (n*f)/(f-n);
float ct  = 1./tan(RAD(angle)/2);

P.v[0]  = ct/aspect;
P.v[5]  = ct;
P.v[10] = -z1;
P.v[11] = -1;
P.v[14] = -2*z2;
P.v[15] = 1;
```

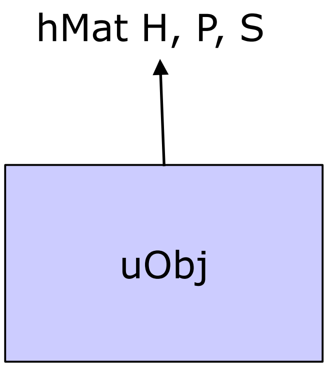
```
void uWnd::OnPaint()
{
    CPaintDC dc(this);
    CDC *pDC = &dc;

    CRect rect;
    GetClientRect(rect);
    pDC->SetMapMode(MM_ANISOTROPIC);
    pDC->SetWindowExt(rect.Width()/2,rect.Height()/2);
    pDC->SetViewportExt( rect.Width()/2,-rect.Height()/2);
    pDC->SetViewportOrg( rect.Width()/2, rect.Height()/2);

    Draw(pDC);
}
```

$$hMat = \begin{bmatrix} v[0] & v[4] & v[8] & v[12] \\ v[1] & v[5] & v[9] & v[13] \\ v[2] & v[6] & v[10] & v[14] \\ v[3] & v[7] & v[11] & v[15] \end{bmatrix}$$

pp. 23

$$P = \begin{pmatrix} \frac{\cot(\alpha/2)}{W/H} & 0 & 0 & 0 \\ 0 & \cot(\alpha/2) & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

# 3D Object Building

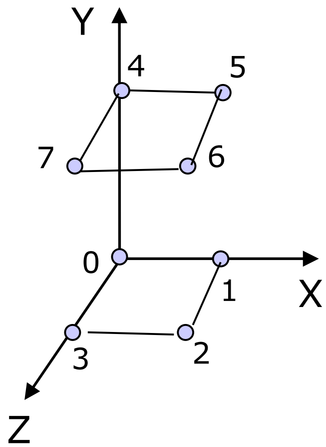



H: Object transform

P: Perspective Projection

S: Scaling

```
vertex[0]   = uVector(0,0,0);
vertex[1]   = uVector(1,0,0);
vertex[2]   = uVector(1,1,0);
vertex[3]   = uVector(0,1,0);
vertex[4]   = uVector(0,0,1);
vertex[5]   = uVector(1,0,1);
vertex[6]   = uVector(1,1,1);
vertex[7]   = uVector(0,1,1);
```




Polygon with 3 vertices

```
void uObj::DrawPolygon(CDC *pDC, uVector f,uVector s,uVector t)
{
    pDC->MoveTo(f.x,f.y);
    pDC->LineTo(s.x,s.y);
    pDC->LineTo(t.x,t.y);
    pDC->LineTo(f.x,f.y);
}
```

Draw Counter Clock Wise

```
DrawPolygon(pDC, temp[6],temp[2],temp[1]);//right
DrawPolygon(pDC, temp[6],temp[1],temp[5]);
```

# Drawing with Perspective Projection

$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ → **uObj::H** → **uObj::P Perspective Projection** → $\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$ → **uObj::S Scale** →

vertex

Remind

$x', y' = [-1, 1]$

$x', y'$

$= 640 \text{ x } 480$

$$X' = \begin{pmatrix} \dfrac{x}{-z} & \dfrac{y}{-z} & \dfrac{z(n+f)+2nf}{-z(n-f)} & 1 \end{pmatrix}^T \quad (z<0)$$

$$\frac{x'}{n} = \frac{x}{z}$$

$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ →

3D vertex

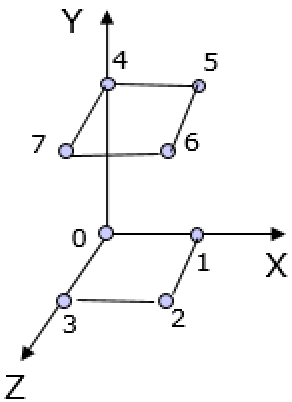
```
// transform vertex
for (i=0;i<nMax;i++)
{
    temp[i] = H*vertex[i];

    float z = temp[i].z;
    temp[i] = P*temp[i];
    temp[i].x   = -temp[i].x/z;
    temp[i].y   = -temp[i].y/z;
    temp[i].z   = -temp[i].z/z;

    temp[i] = S*temp[i];
}
```

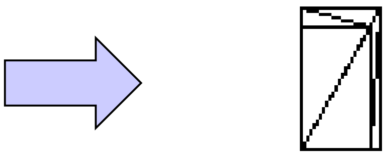
→ $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$

2D temp

# Why Width and Height are Different?

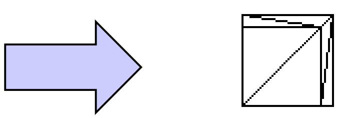
```
vertex[0]   = uVector(0,0,0);
vertex[1]   = uVector(1,0,0);
vertex[2]   = uVector(1,1,0);
vertex[3]   = uVector(0,1,0);
vertex[4]   = uVector(0,0,1);
vertex[5]   = uVector(1,0,1);
vertex[6]   = uVector(1,1,1);
vertex[7]   = uVector(0,1,1);
```



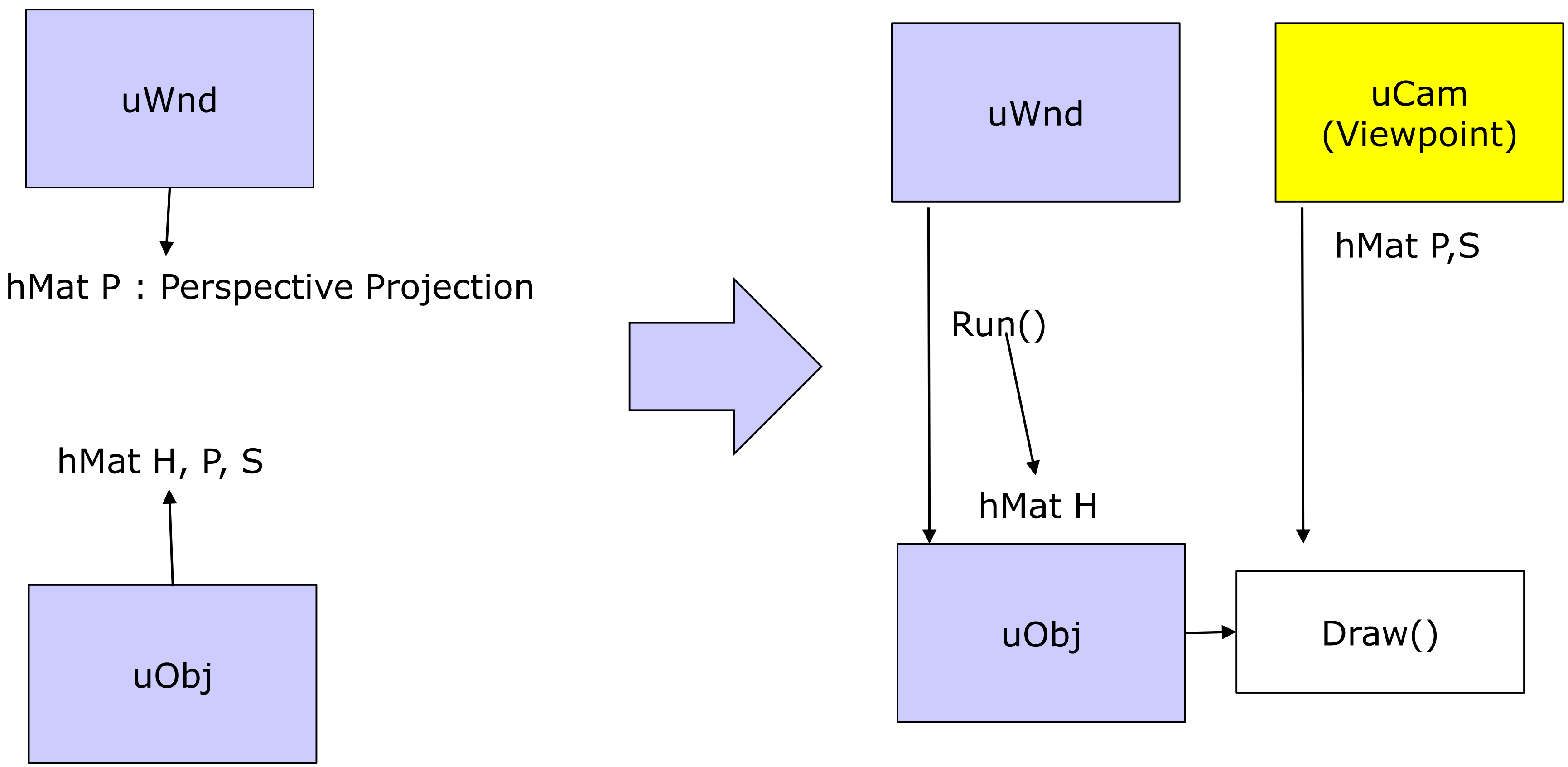
- Answer: That is aspect ratio.

- Windows CDC already changes aspect ratio
- Thus, modify aspect ratio=1



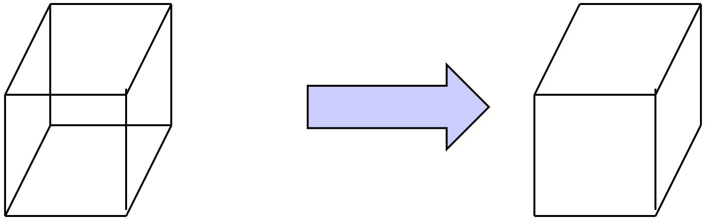
# Ex) uWnd-20-3D-PP-Camera
# New Class: uCam for projection

- Think, this structure is somewhat bad.



# Next week

- Hidden surface removal



- Various types of Objects