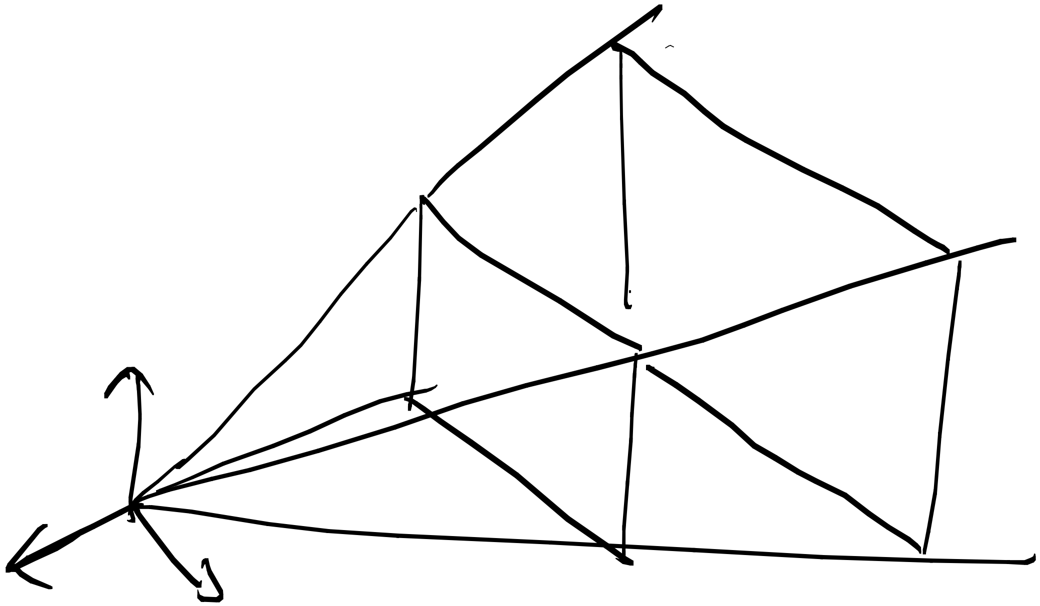
lecture 5

- projective transformation
- normalized view volume
- GL_PROJECTION matrix
- clip coordinates
- normalized device coordinates

- planes and normals in projective space
- Assignment 1 (python and pyopengl)

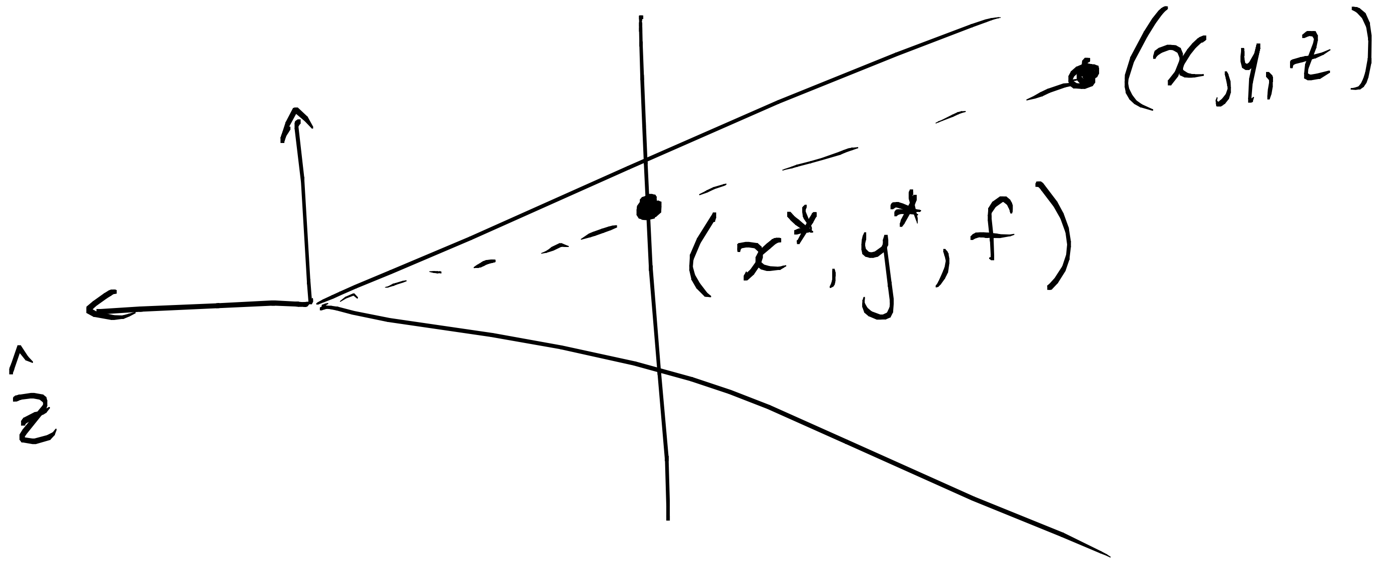
Recall last lecture: view volume (view frustum)



gluPerspective(θy, θx / θy , near, far)

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

Recall last lecture: projection



f = -near

$$
\begin{bmatrix} fx \\ fy \\ fz \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

You might think.....

$$M_{image \leftarrow eye} \qquad \underbrace{M_{eye \leftarrow world}}_{\text{gluLookat}} \; \underbrace{M_{world \leftarrow obj}}_{\substack{\text{glTranslate} \\ \text{glRotate} \\ \text{glScale}}}$$

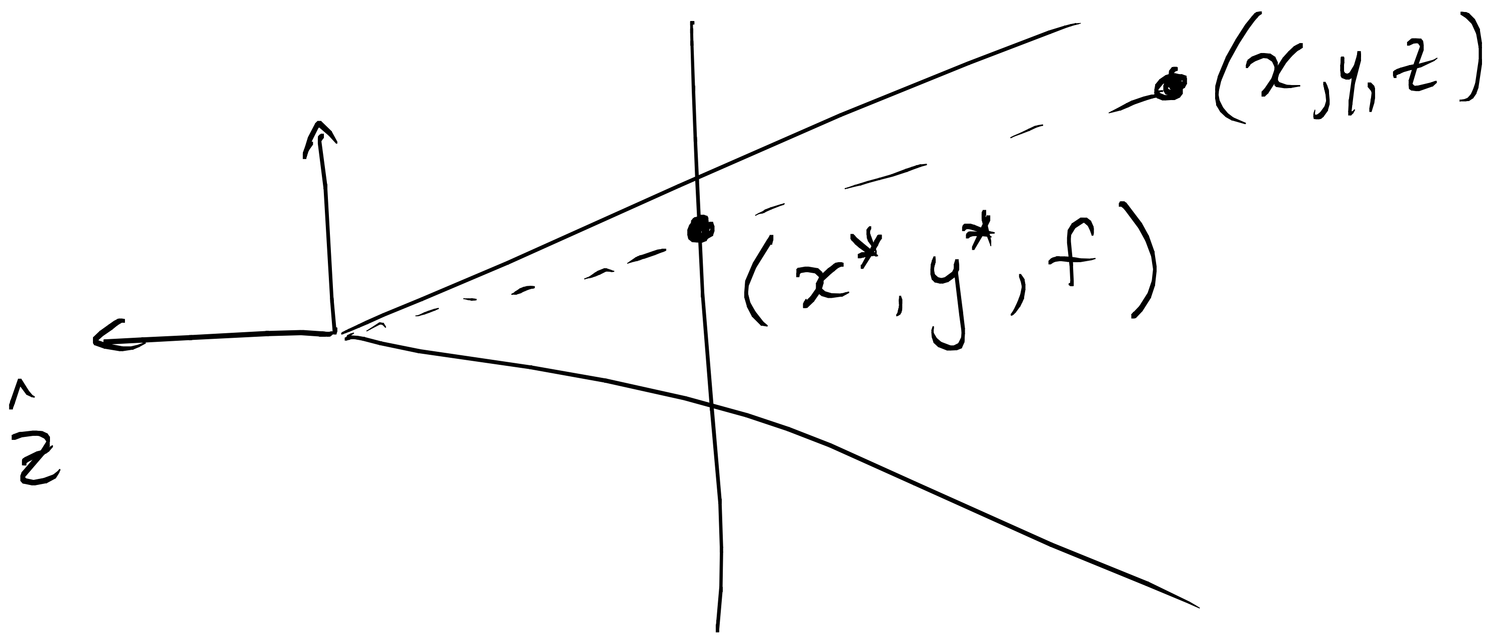
gluPerspective

gl Frustum

But that is <u>not</u> what glFrustrum and gluPerspective do.

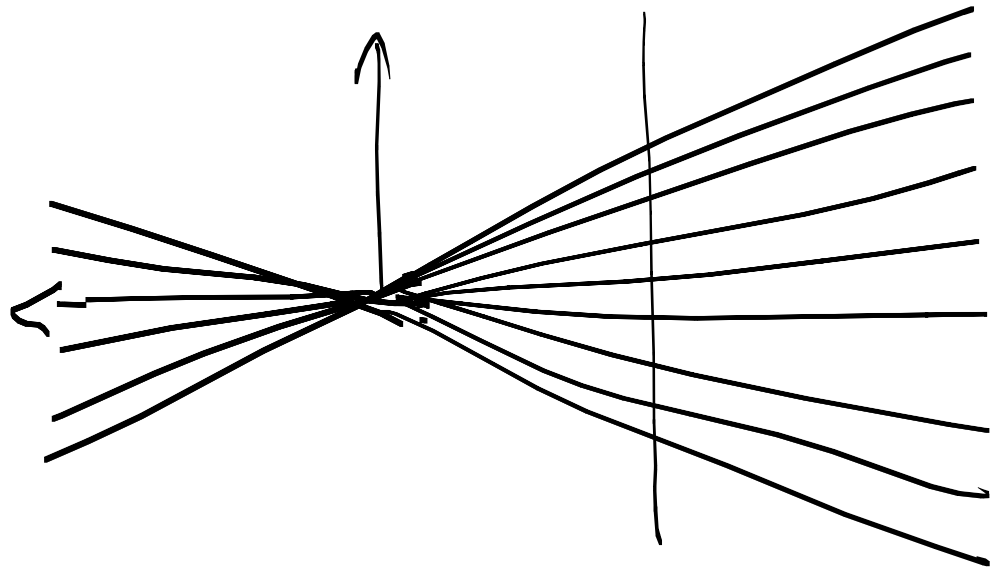
Why not ?   What do they do?

The problem with projection:

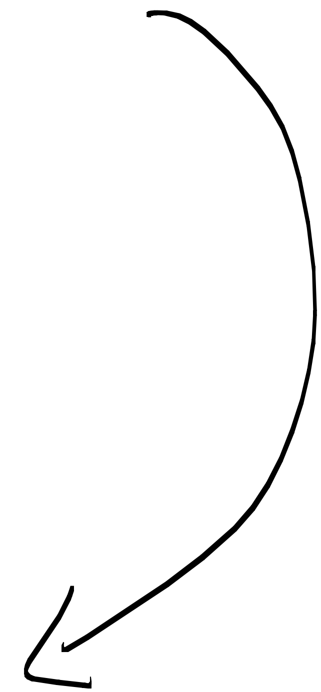


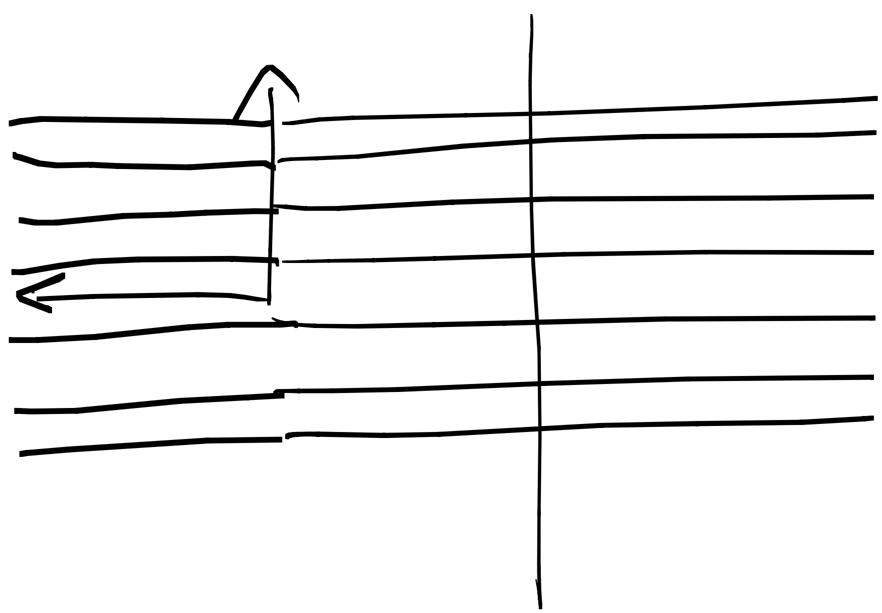
If we discard the z information, then we don't know which objects are in front of which.

Projective Transformation

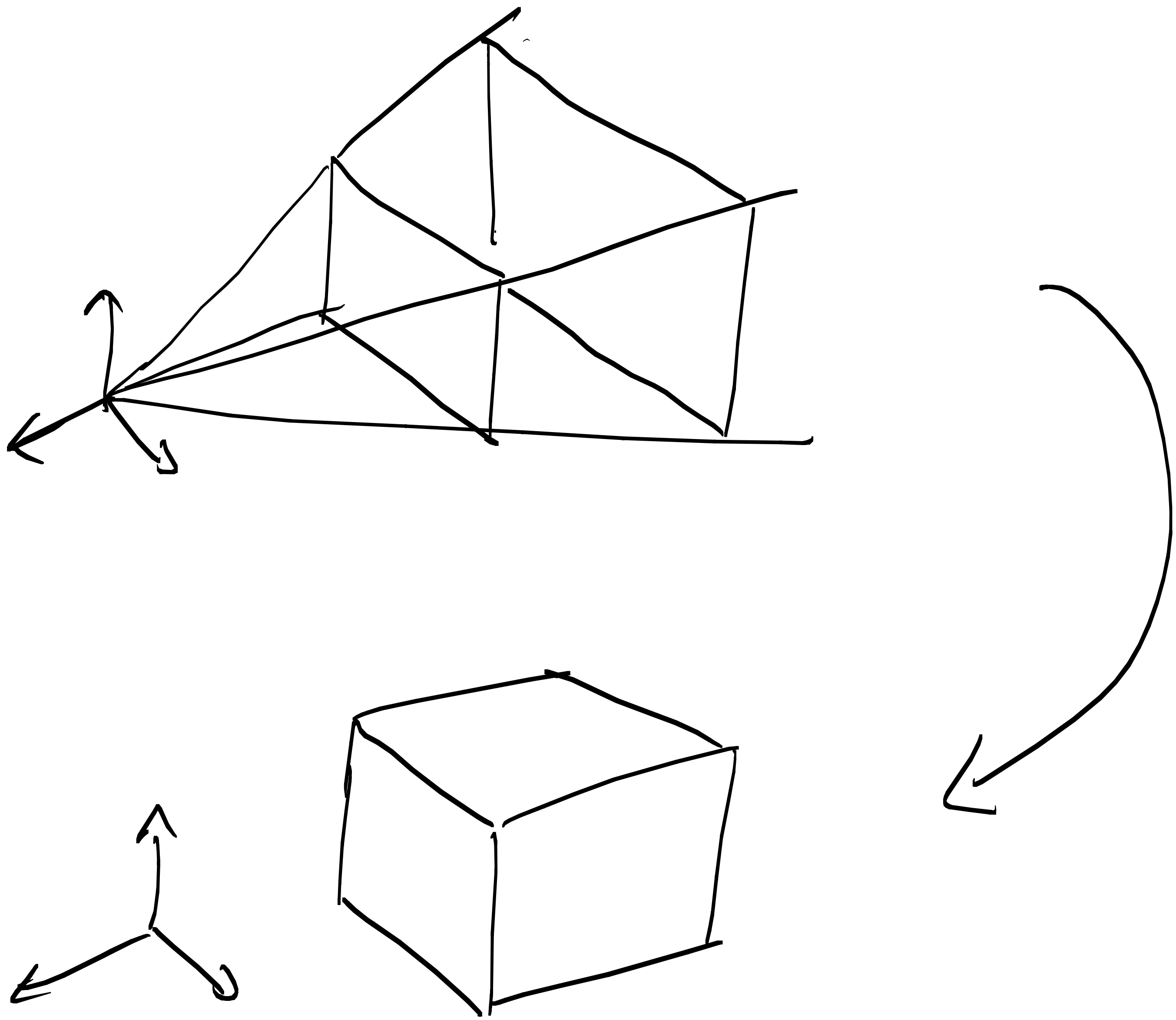


projection plane (near)

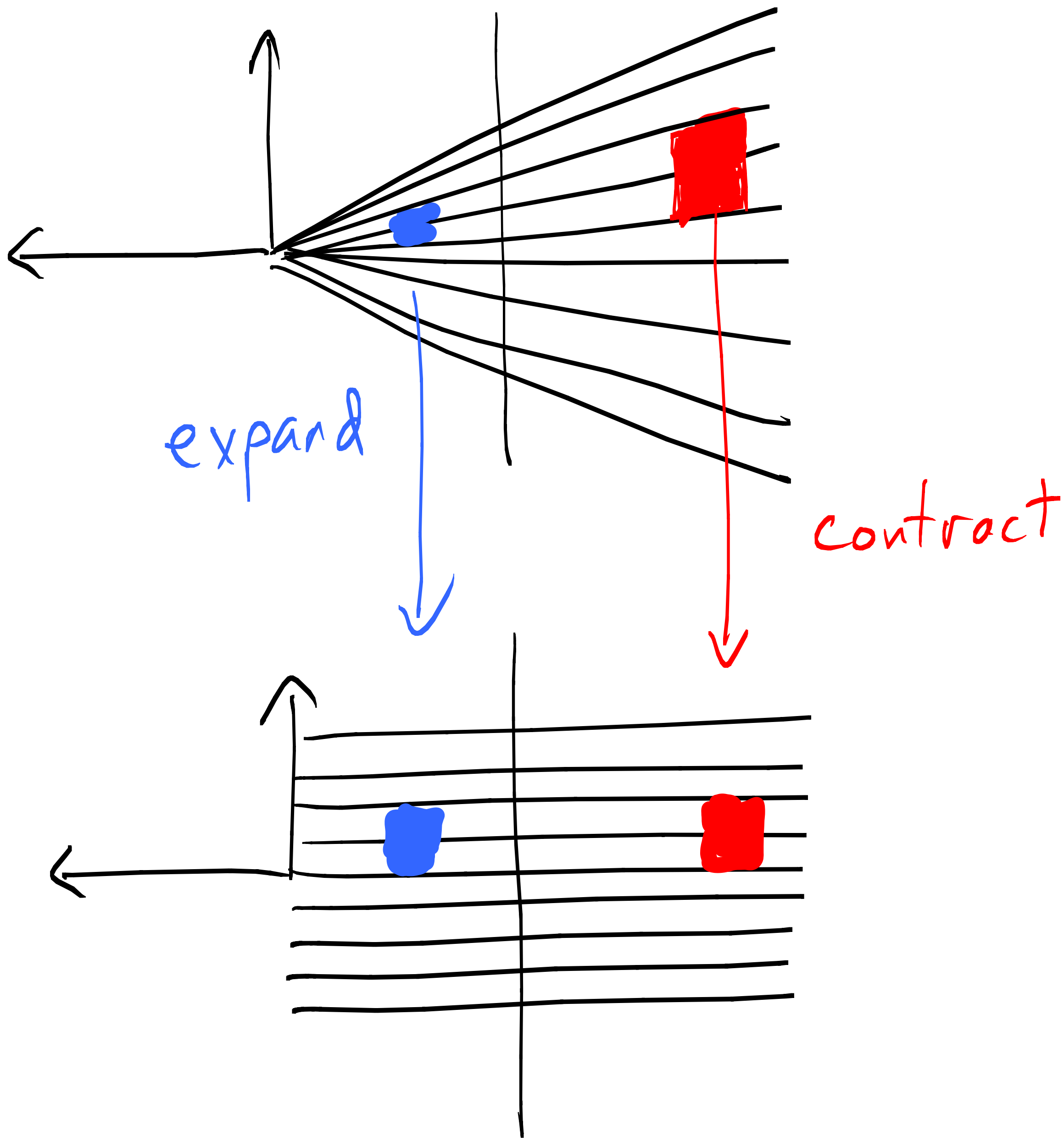




Projective Transformation



Objects that are further away look smaller.



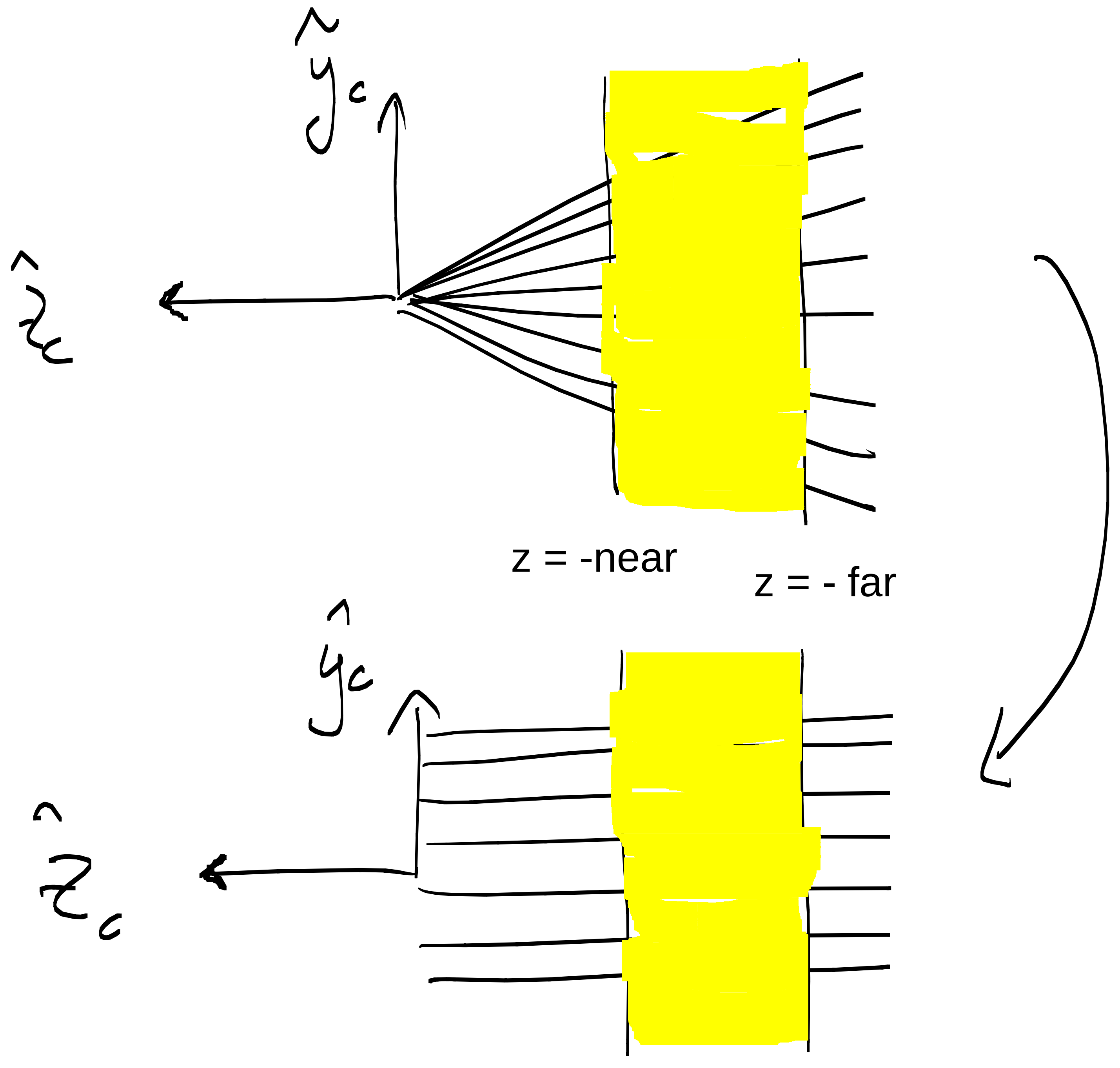
How to define a projective transformation that does this ?

Previously, we considered projection :

$$\begin{bmatrix} fx \\ fy \\ fz \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

But a projection matrix is not invertible
(3rd and 4th rows are linearly dependent)

Projective Transformation



We choose α and β to satisfy desired map illustrated on previous slide.

$$\begin{bmatrix} fx \\ fy \\ fz \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \cancel{f}\ \alpha & \cancel{0}\ \beta \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where $\beta \neq 0$

In Appendix to lecture notes, I derive (easy) :

$$\alpha = f_0 + f_1 , \quad \beta = - f_0 f_1$$

$$\begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & f_0 + f_1 & -f_0 f_1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where $f_0 = -near , \quad f_1 = -far .$

$$f_0 = -\text{near} \qquad f_1 = -\text{far}$$

$$\begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & f_0+f_1 & -f_0 f_1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ f_0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} f_0 x \\ f_0 y \\ f_0 f_0 \\ f_0 \end{bmatrix} \equiv \begin{bmatrix} x \\ y \\ f_0 \\ 1 \end{bmatrix}$$

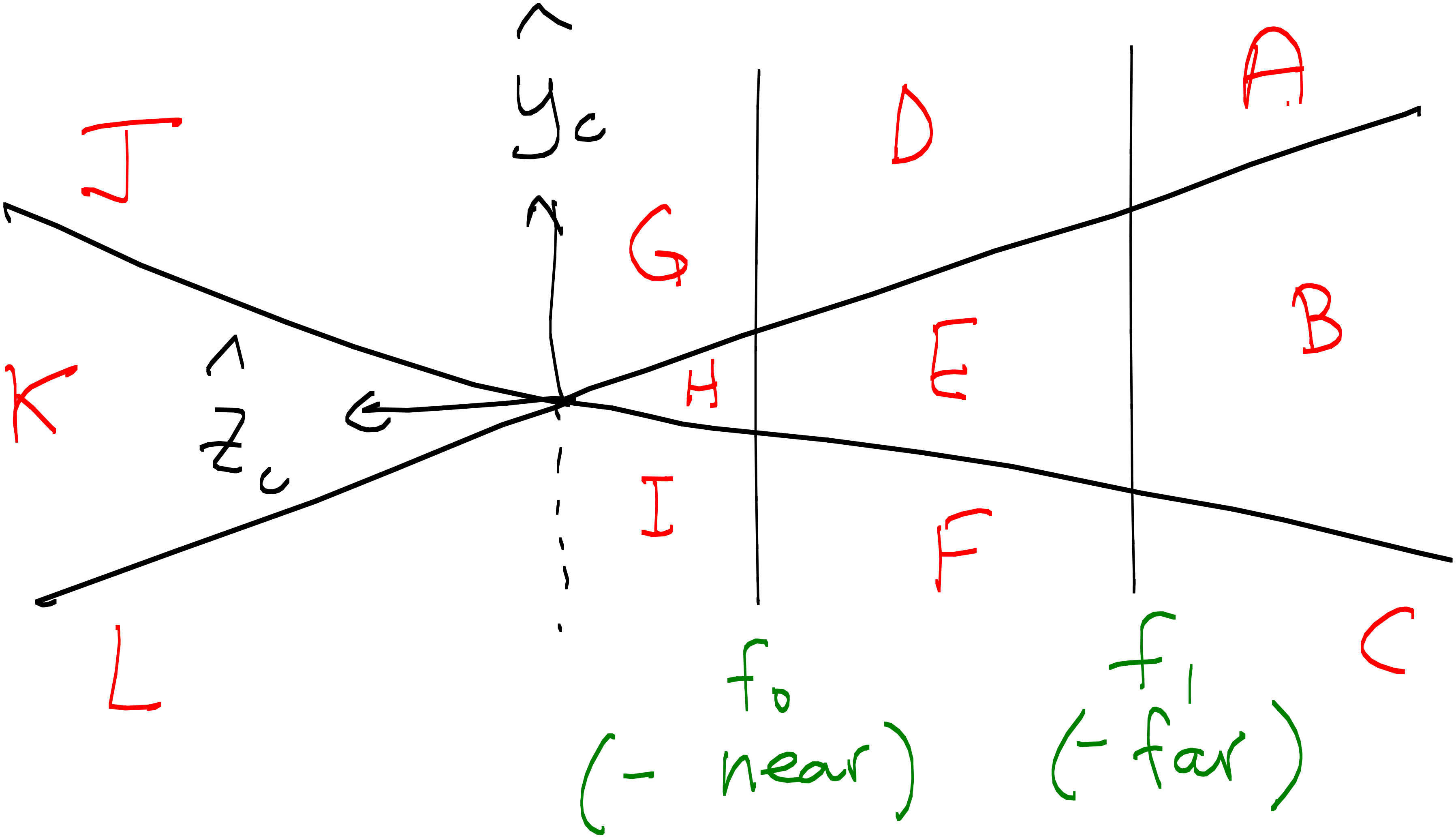
Thus, $z = f_0$ maps to $z = f_0$.

$$f_0 = -\text{near} \qquad f_1 = -\text{far}$$

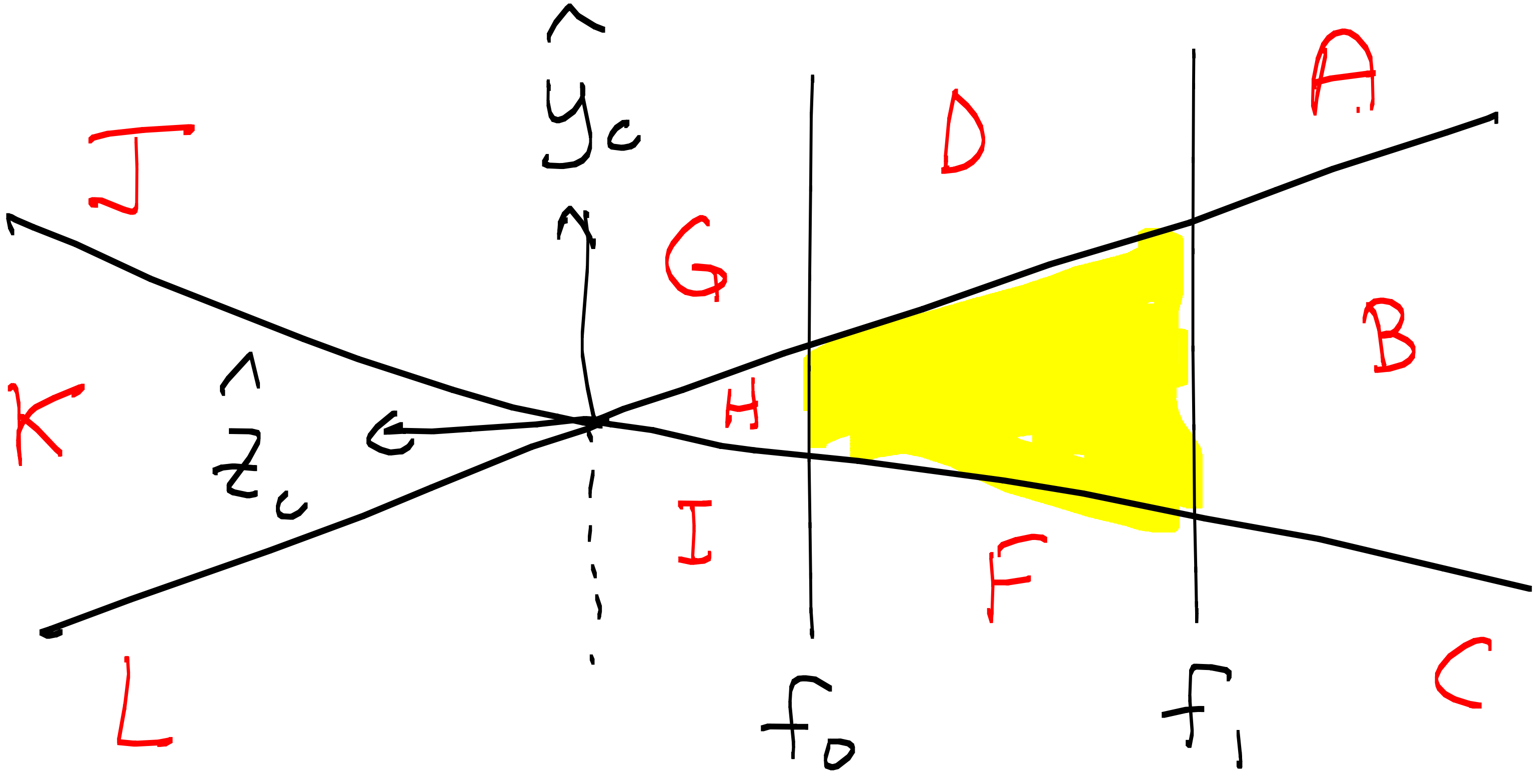
$$\begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & f_0+f_1 & -f_0 f_1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ f_1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} f_0\, x \\ f_0\, y \\ f_1 f_1 \\ f_1 \end{bmatrix} = \begin{bmatrix} x\, f_0/f_1 \\ y\, f_0/f_1 \\ f_1 \\ 1 \end{bmatrix}$$

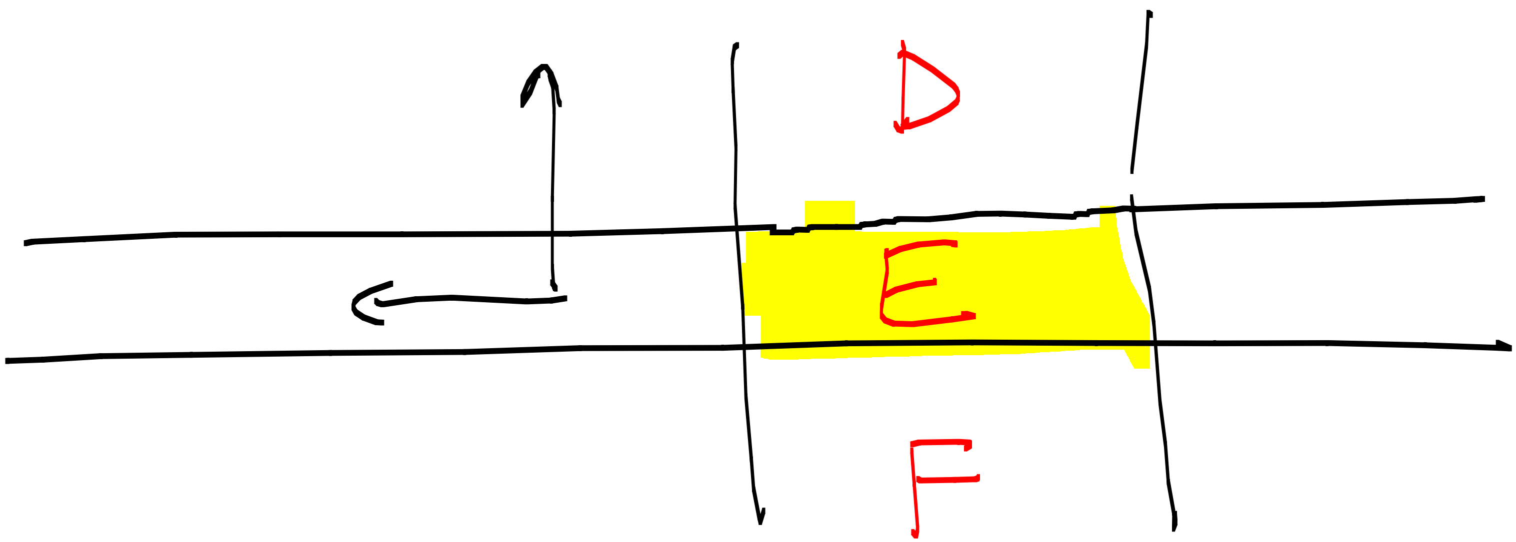
Thus, $z = f_1$ maps to $z = f_1$ and x, y are compressed.



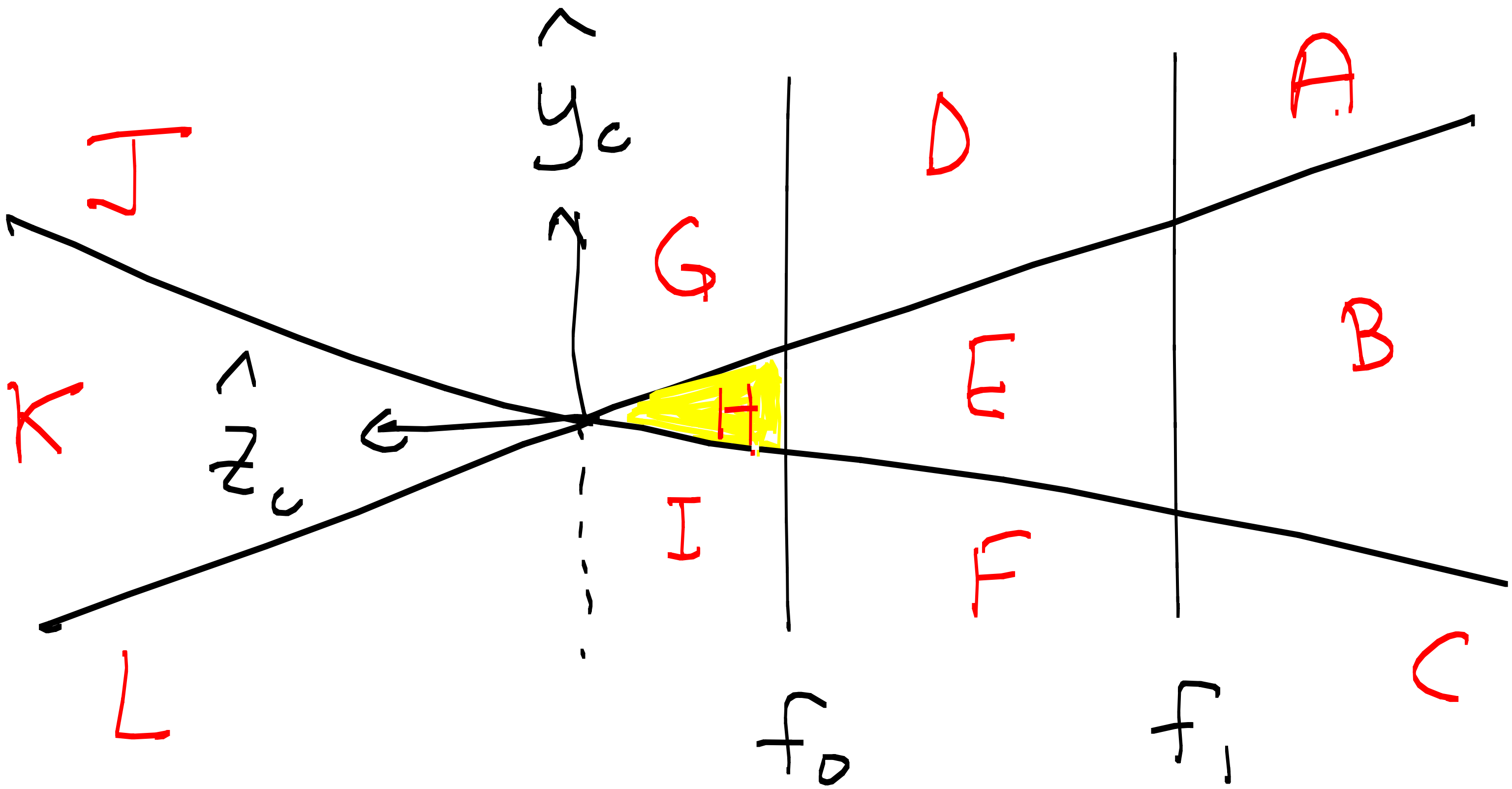
Where do various regions map to ?



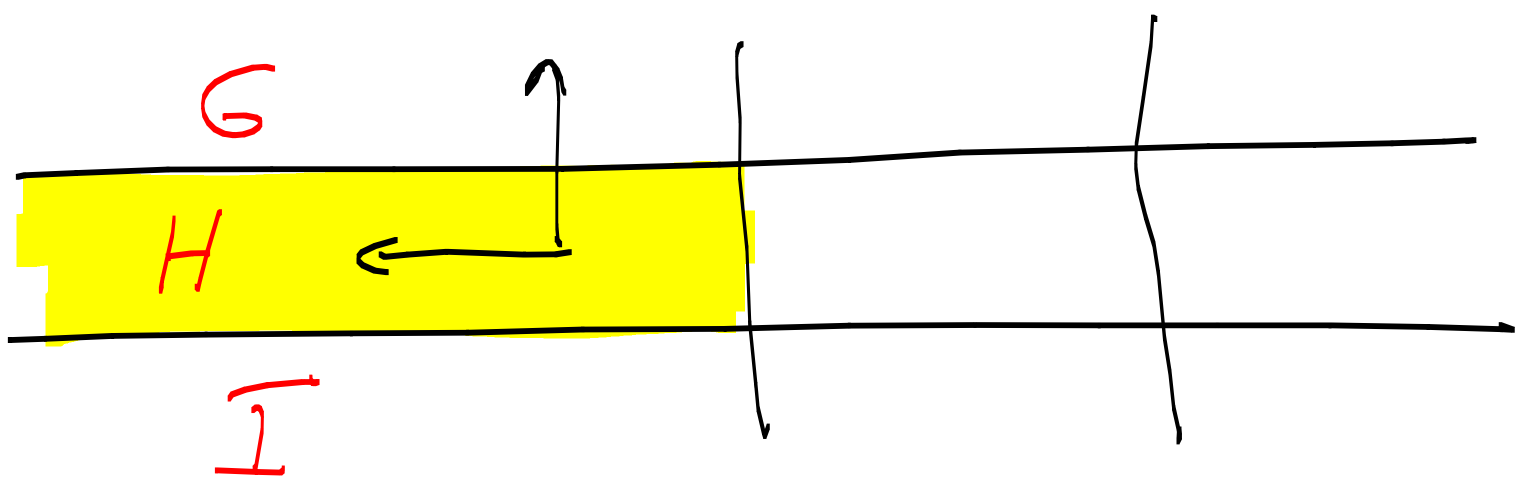
$\hat{y}_c$
$\hat{z}_c$
J
K
L
G
H
I
D
A
B
C
F
f_0
f_1



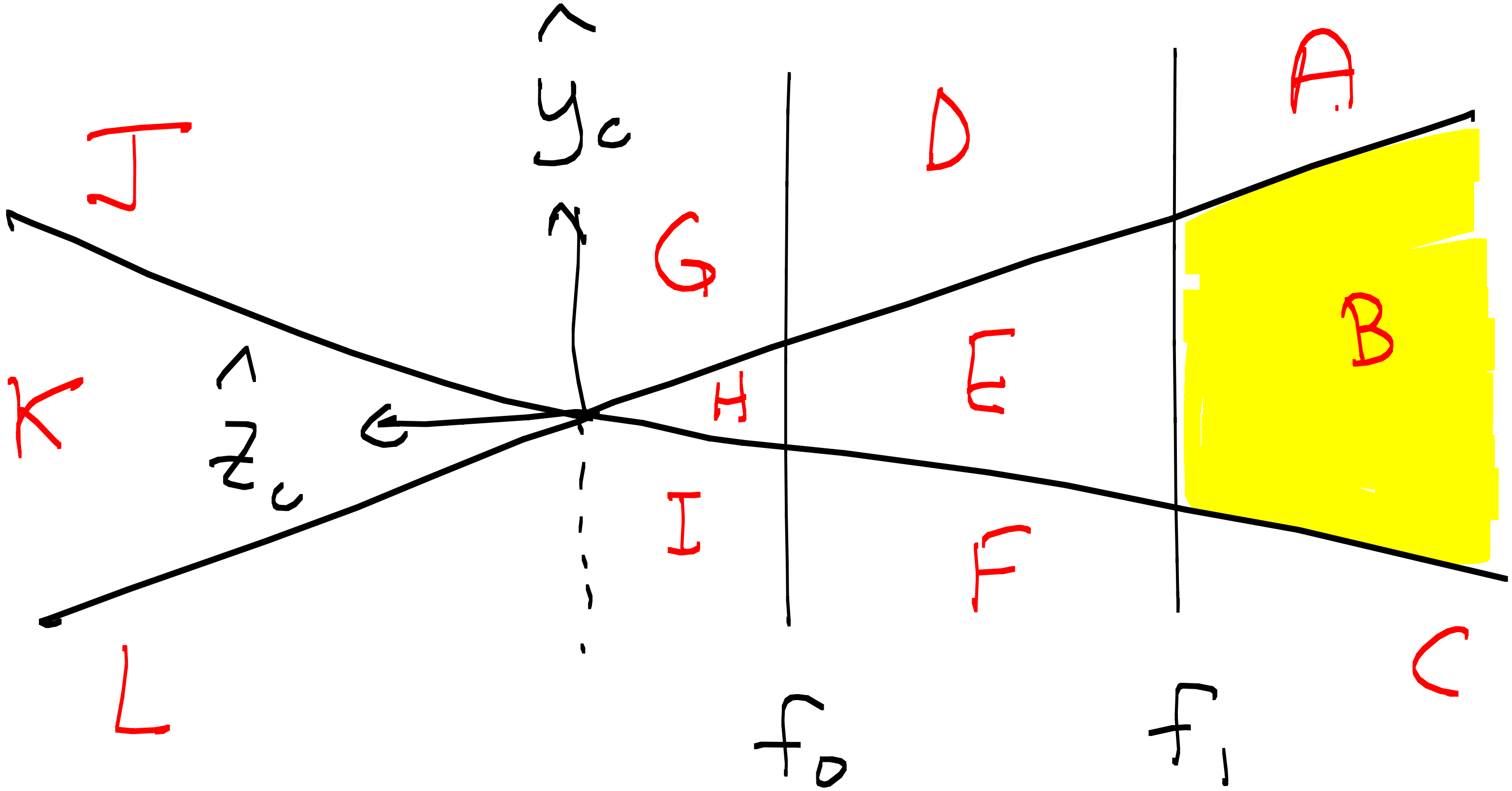
D
E
F

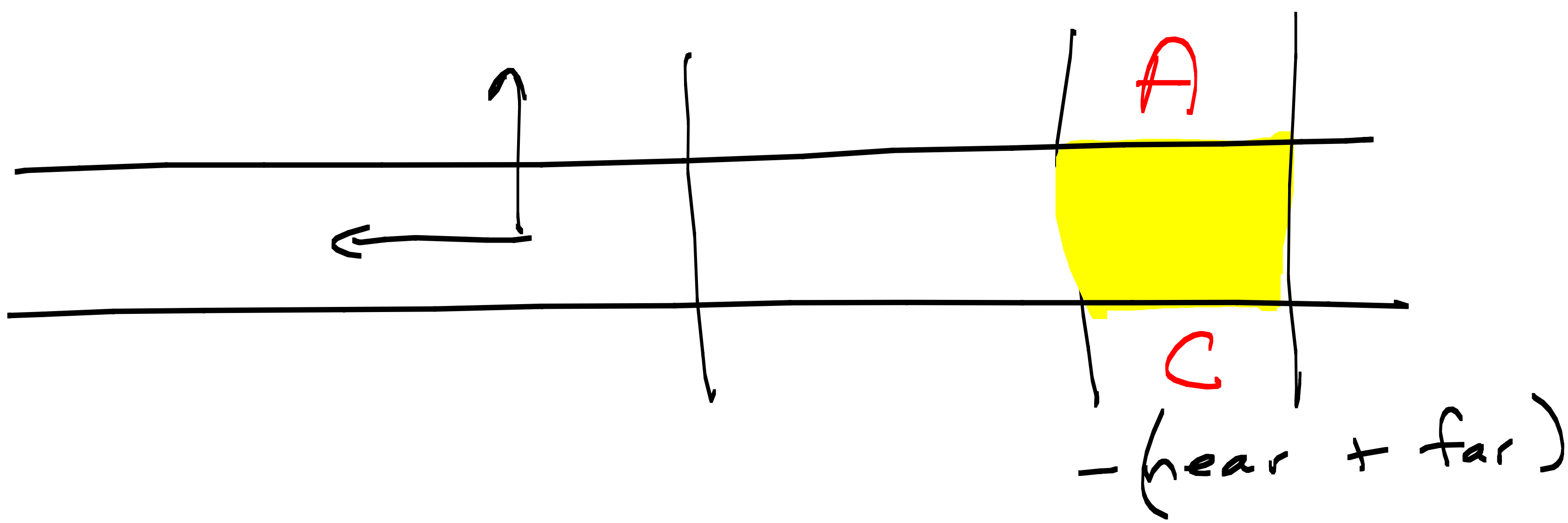


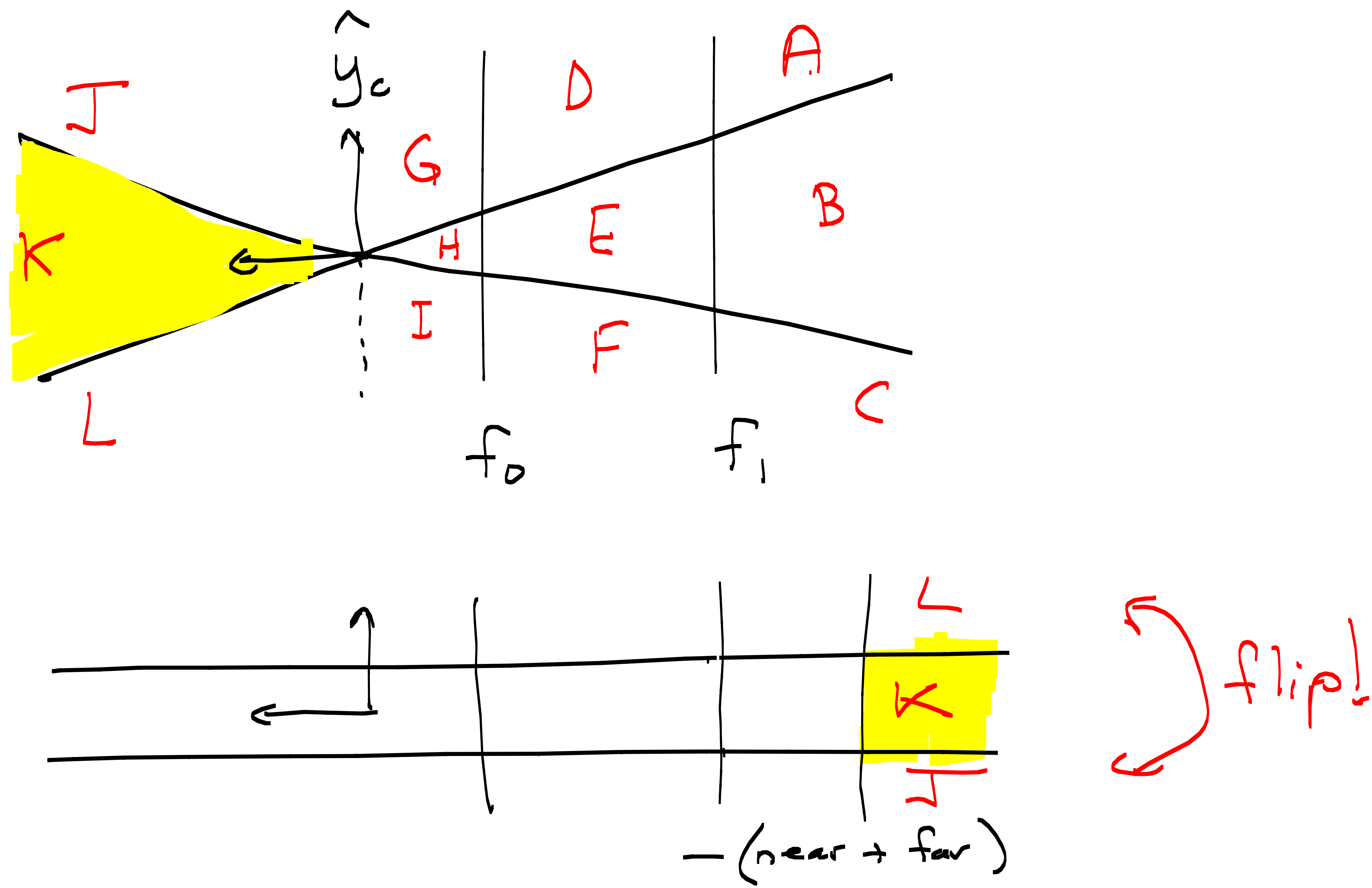
A
B
C
D
E
F
G
H
I
J
K
L
$\hat{y}_c$
$\hat{z}_c$
f_0
f_1



G
H
I

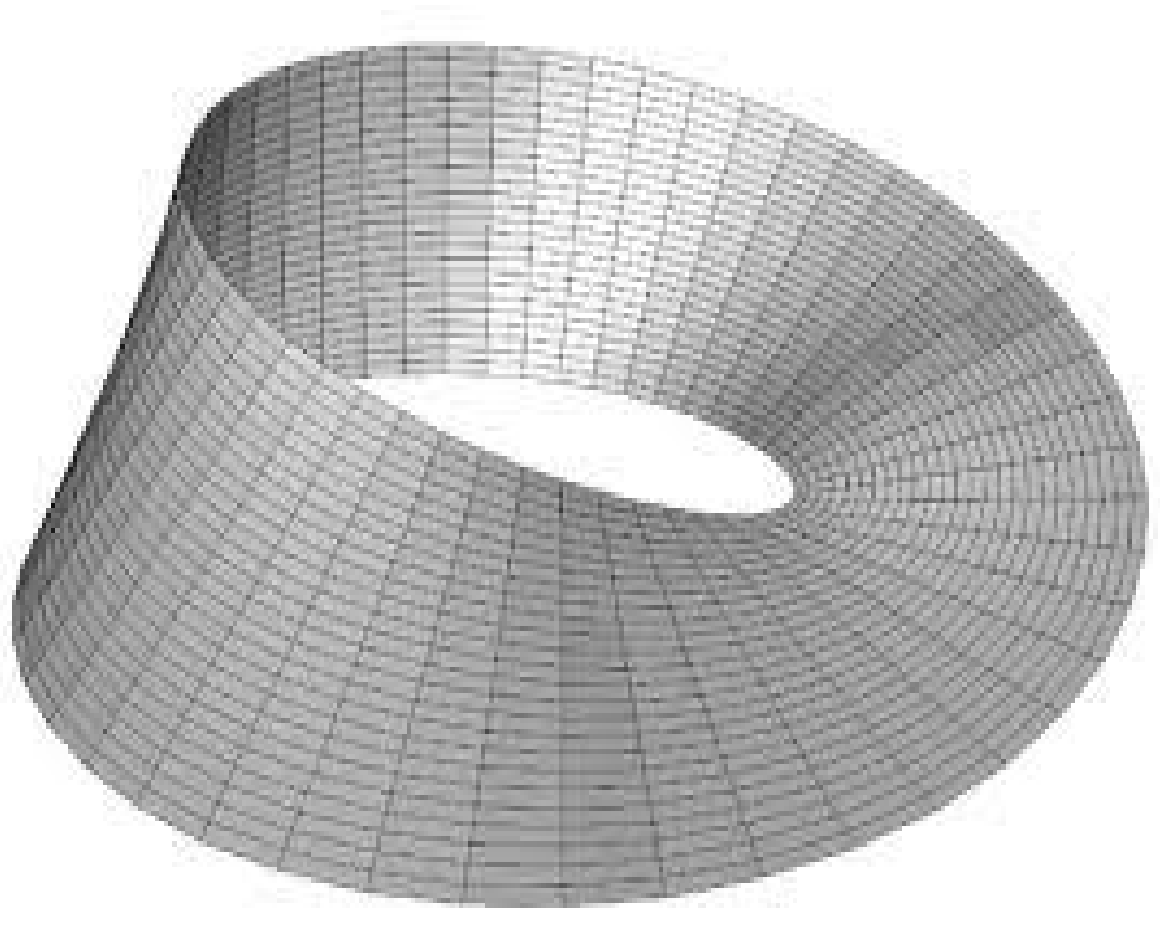

$\hat{y}_c$
J
K
$\hat{z}_c$
L
G
H
I
D
E
F
A
B
C
f_0
f_1


A
C
$-(near + far)$



$\hat{y}_c$
A
B
C
D
E
F
G
H
I
J
K
L
f_0
f_1
L
K
J
$-(near + far)$
flip!

Curious !   The JKL flipping is analogous to a Mobius strip.



Why is the above detail important ?

We decide whether or not points lie in the view volume using projective space representation.

To make these decisions correctly, we need to be careful about inequalities and signs.

More about this later...

Another (surprisingly) important detail:

$$\begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & f_0+f_1 & -f_0 f_1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\equiv \begin{bmatrix} -f_0 & 0 & 0 & 0 \\ 0 & -f_0 & 0 & 0 \\ 0 & 0 & -(f_0+f_1) & f_0 f_1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

OpenGL uses the 2nd matrix above, not the first.

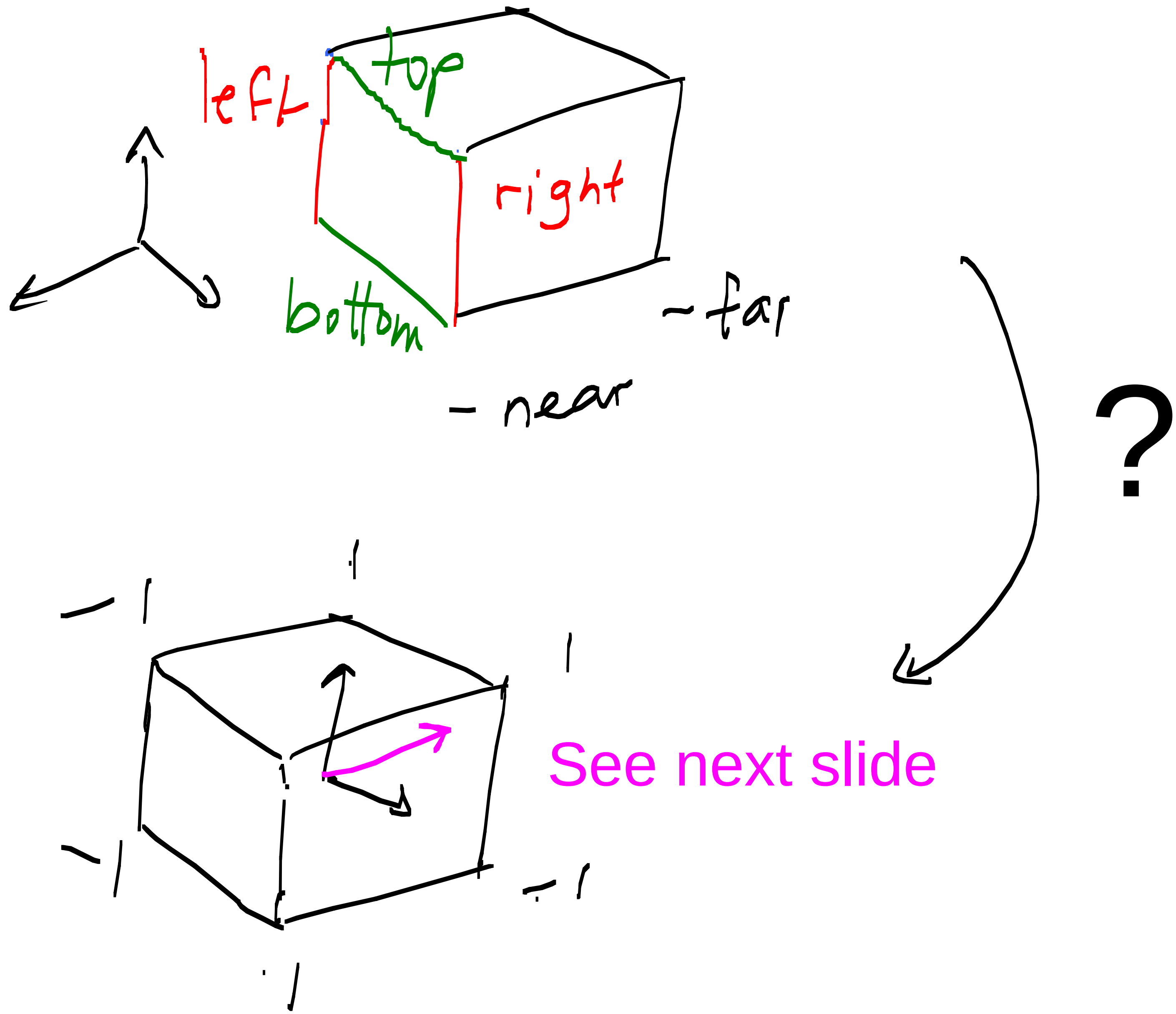
Why ?

$$f_0 = -\text{near} \qquad f_1 = -\text{far}$$

$$\begin{bmatrix} -f_0 & 0 & 0 & 0 \\ 0 & -f_0 & 0 & 0 \\ 0 & 0 & -(f_0+f_1) & f_0 f_1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} * \\ * \\ * \\ -z \end{bmatrix}$$

The "w" coordinate is positive $\iff z < 0$.

Soon we will see why this is desirable.



"Normalized view volume"

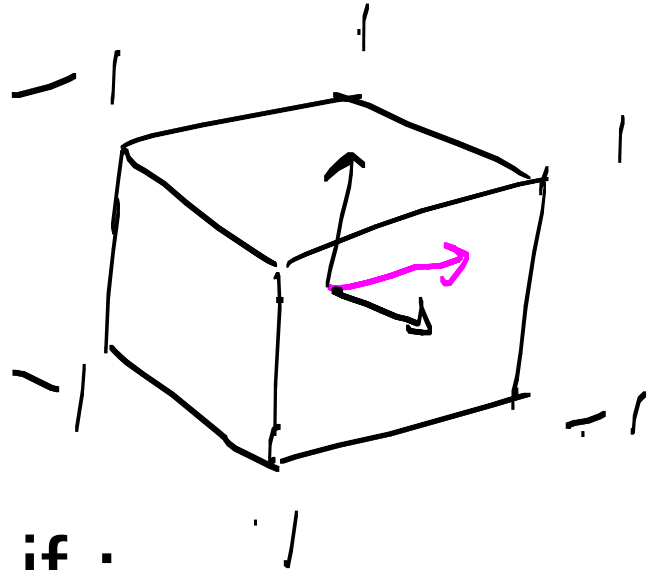
Map to normalized view volume

1) translate (left, bottom, -near) to (0,0,0)
2.) rescale x, y, z so volume is 2 x 2 x 2
and **flip z axis**
(into left handed coordinates !)

3.) translate so volume is centered at origin

$$M_{normalize} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & 0 \\ 0 & \frac{2}{top-bottom} & 0 & 0 \\ 0 & 0 & \frac{-2}{far-near} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -left \\ 0 & 1 & 0 & -bottom \\ 0 & 0 & 1 & near \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(w x, w y, w z, w)



is in the

normalized view volume if :

w > 0   (Recall a few slides ago)

$- w <= w x <= w$

$- w <= w y <= w$

$- w <= w z <= w$

Putting in all together ...

last few lectures

$M_{eye \leftarrow world} \quad M_{world \leftarrow object}$

this lecture :

$M_{normalize} \quad M_{projective}$

$$\underbrace{M_{normalize} \; M_{projective}} \quad \underbrace{M_{eye \leftarrow world}} \quad \underbrace{M_{world \leftarrow obj}}$$

or {
gluPerspective
glFrustum
}

gluLookat

glTranslate
glRotate
glScale

$M_{GL\_PROJECTION}$ $M_{GL\_MODELVIEW}$

Object Coordinates $\qquad$ "Clip Coordinates"

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

$$\begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix} = M_{\text{normalize}} \, M_{\text{projective}} \, M_{\text{eye} \leftarrow \text{world}} \, M_{\text{world} \leftarrow \text{obj}} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Object Coordinates

"Clip Coordinates"

"Normalized Device Coordinates"

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

"Perspective Division"

In OpenGL, this happens after clipping (next lecture)

A few final details:

- projective transform of a plane?
- projective transform of a surface normal?

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

$$(a, b, c, d) \cdot (x, y, z, 1) = 0$$

$$\overbrace{M^{-1} \quad M}$$

for any invertible 4×4 M.

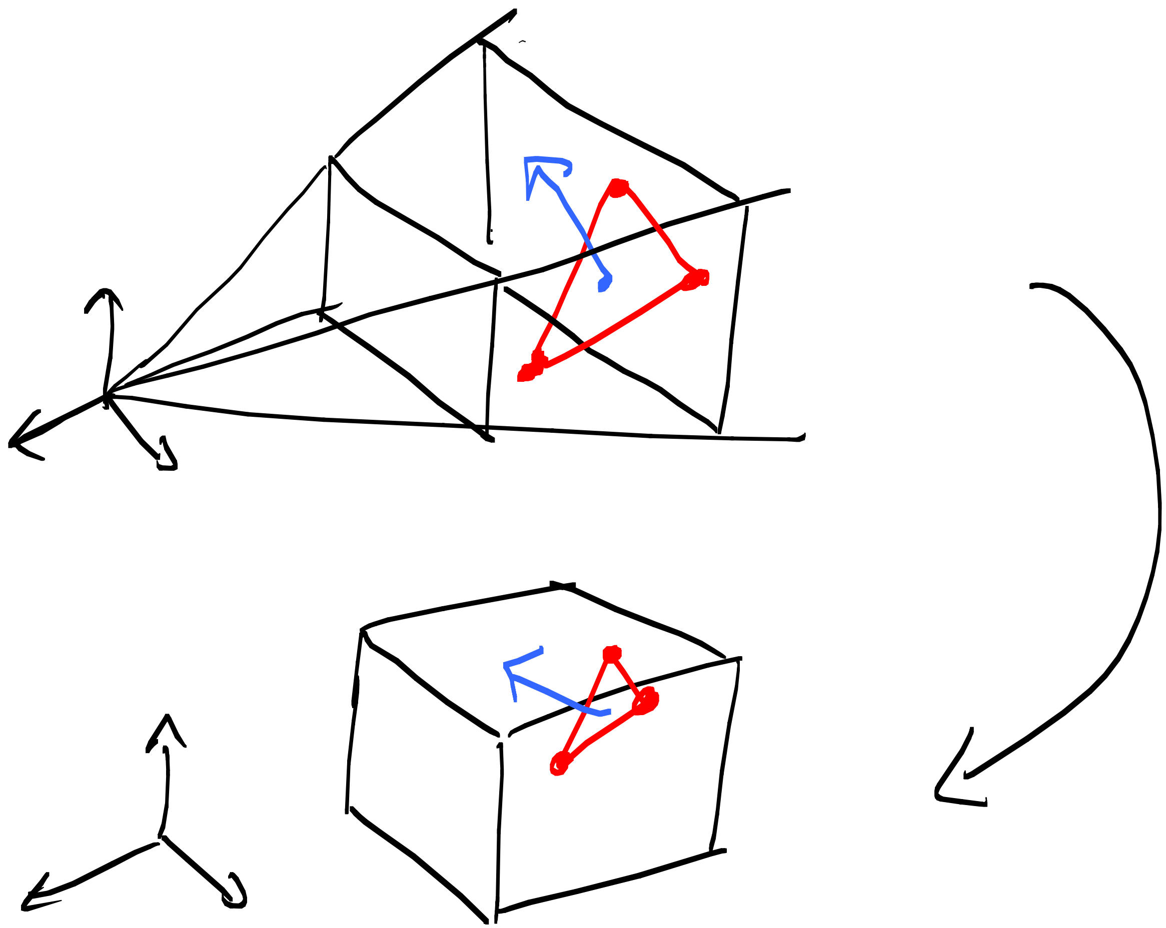
$$\underbrace{(a, b, c, d)\, M^{-1}}\ \underbrace{M \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}} = 0$$

$$(a', b', c', d') \cdot (wx', wy', wz', w) = 0$$

gives equation of plane

in (x', y', z') space.

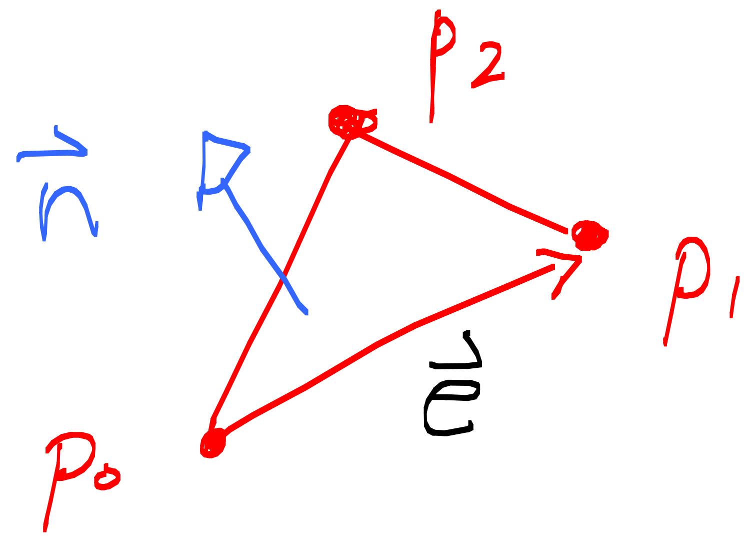
The surface normal of a plane (triangle) does not necessarily get mapped to the surface normal of the mapped plane (triangle).



Why not ? (Space is deformed, and so right angles are not preserved.)

Let $\vec{p}_0$ $\vec{p}_1$ be two points on a plane. Let

- $\vec{n}$ be normal to the plane.
- $\vec{e} = \vec{p}_1 - \vec{p}_0$



$$(n_x, n_y, n_z) \cdot (e_x \; e_y \; e_z) = 0$$

Write $\vec{n}$ and $\vec{e}$ as direction vectors.

Then,

$$[n_x, n_y, n_z, 0]\,[e_x, e_y, e_z, 0]^T = 0$$

Insert $\overbrace{M^{-1}\,M}$

$$\underbrace{\underset{1\times 4}{\vec{n}}\ \underset{4\times 4}{M^{-1}}}_{\text{surface normal of transformed plane}}\ \underbrace{\underset{4\times 4}{M}\ \underset{4\times 1}{\vec{e}^{T}}}_{\text{transformed direction vector lies in transformed plane}} = 0$$

Surface normal of transformed plane

$$(\vec{n} M^{-1})^T = (M^{-1})^T \vec{n}^T$$

except in special situations

$$\neq M \vec{n}^T$$

transformed surface normal of plane

Announcements

- Today is ADD/DROP deadline
  http://www.mcgill.ca/importantdates/key-dates

- For the Assignments, we will use PyOpenGL (Python). It is already installed on the lab computers.

  Fahim (T.A.) has posted instructions for you to install it on your computer:
  http://cim.mcgill.ca/~fmannan/comp557/Python%20and%20PyOpenGL%20Installation.html

  If you need help with the installation, see him (or help each other -- please use the discussion board).

  We will try to get the assignment out Thursday as originally planned.