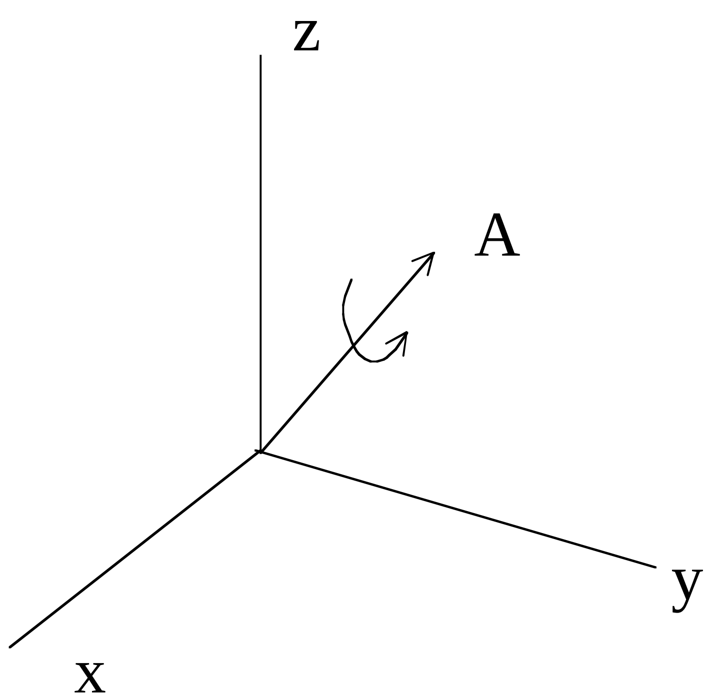
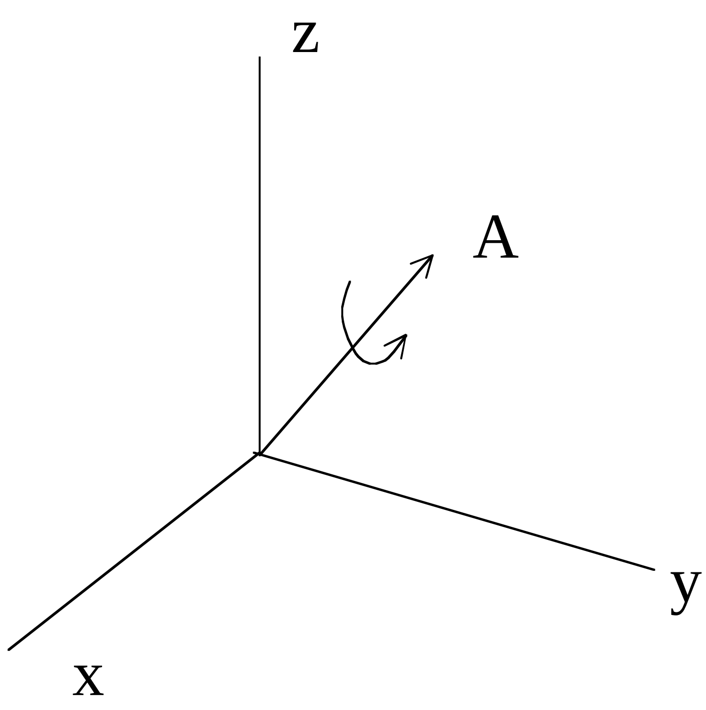
Rotation about an arbitrary axis (assignment hint)

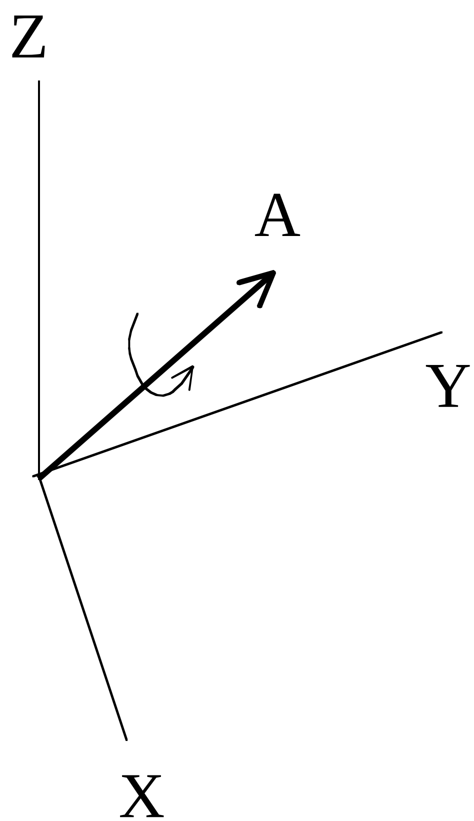


Rotation about an arbitrary axis (assignment hint)



Strategy--rotate A to Z axis, rotate about Z axis, rotate Z back to A.

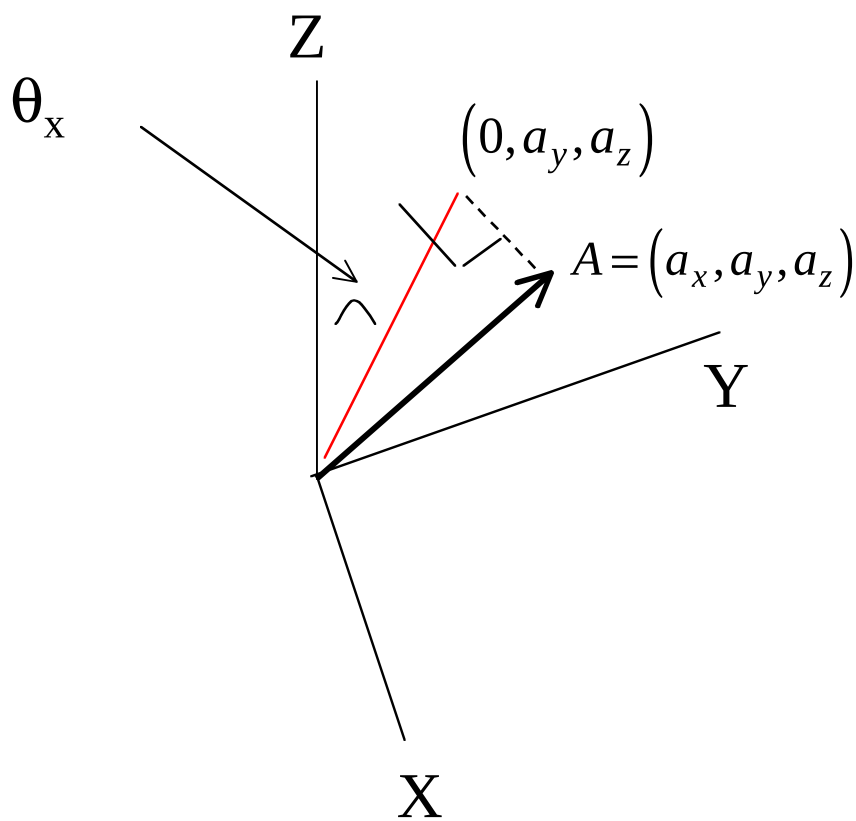
Rotation about an arbitrary axis (assignment hint)



Tricky part:
rotate A to Z axis

Two steps.
1) Rotate about x to xz plane
2) Rotate about y to Z axis.

Rotation about an arbitrary axis (assignment hint)

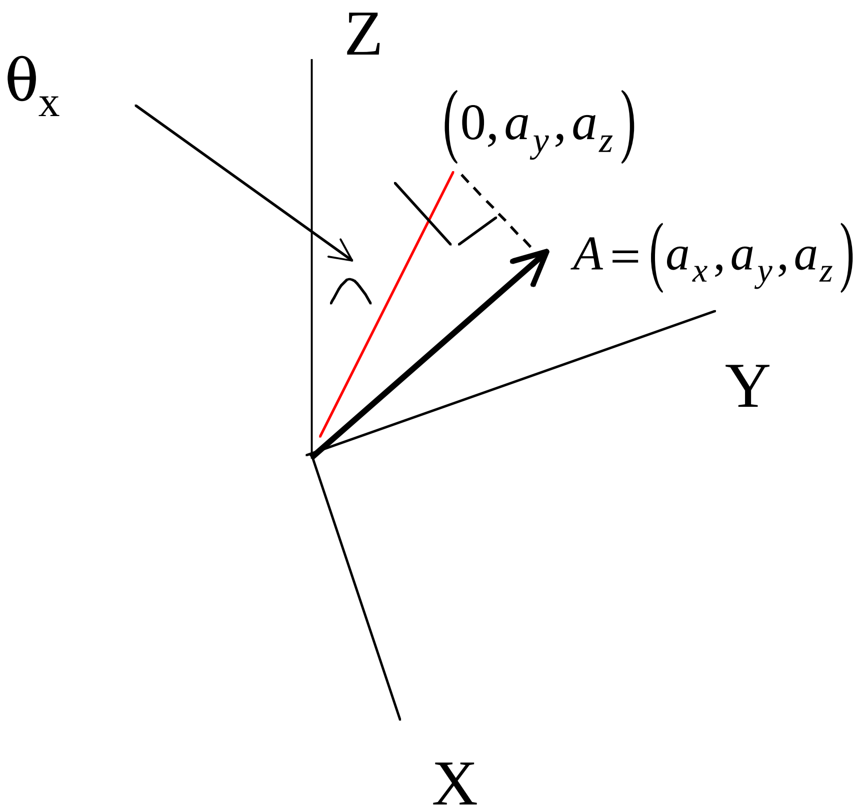


Tricky part:
rotate A to Z axis

Two steps.

1) Rotate about X to xz plane
2) Rotate about Y to Z axis.

Rotation about an arbitrary axis (assignment hint)



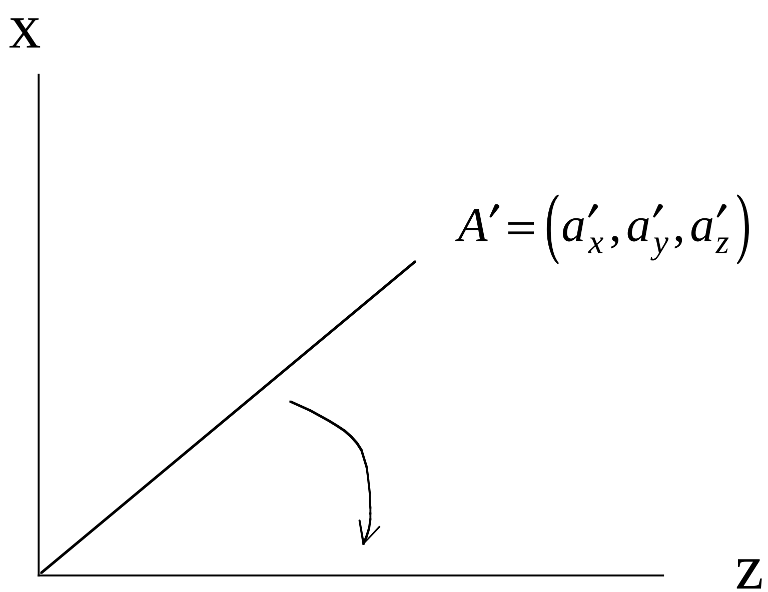
$$d = \sqrt{{a_y}^2 + {a_z}^2}$$

$$\sin\theta_x = a_y / d$$

$$\cos\theta_x = a_z / d$$

No need to compute angles, just put sines and cosines into rotation matrices

Rotation about an arbitrary axis (assignment hint)



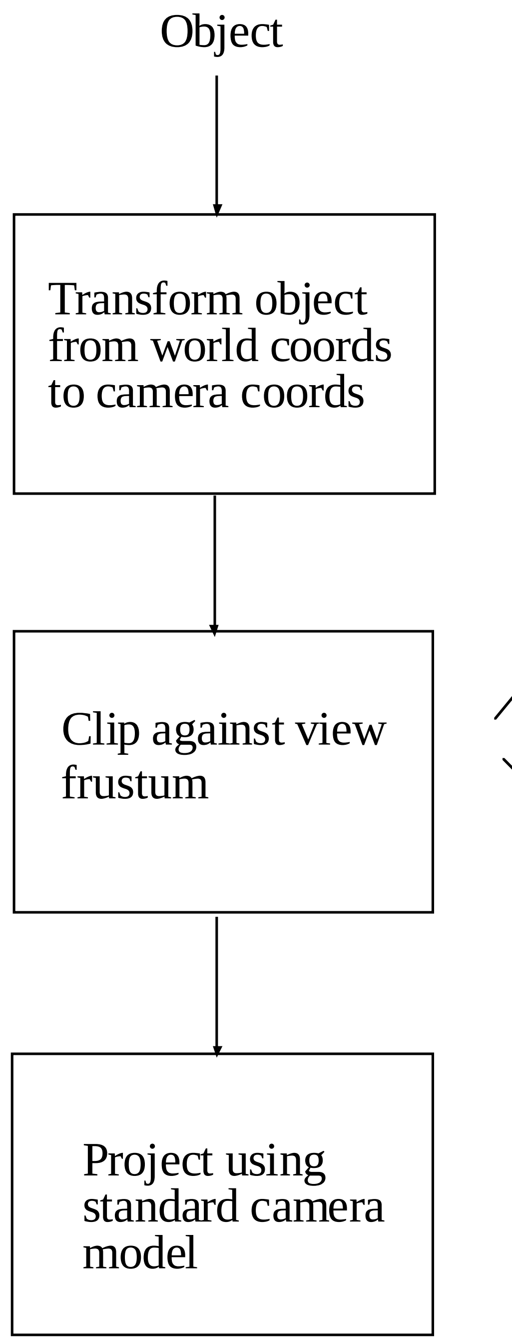
Apply $R_x(\theta_x)$ to A and renormalize to get A'

$R_y(\theta_y)$ should be easy, but note that it is clockwise.

Rotation about an arbitrary axis

Final form is

$$R_x(-\theta_x)R_y(-\theta_y)R_z(\theta_z)R_y(\theta_y)R_x(\theta_x)$$

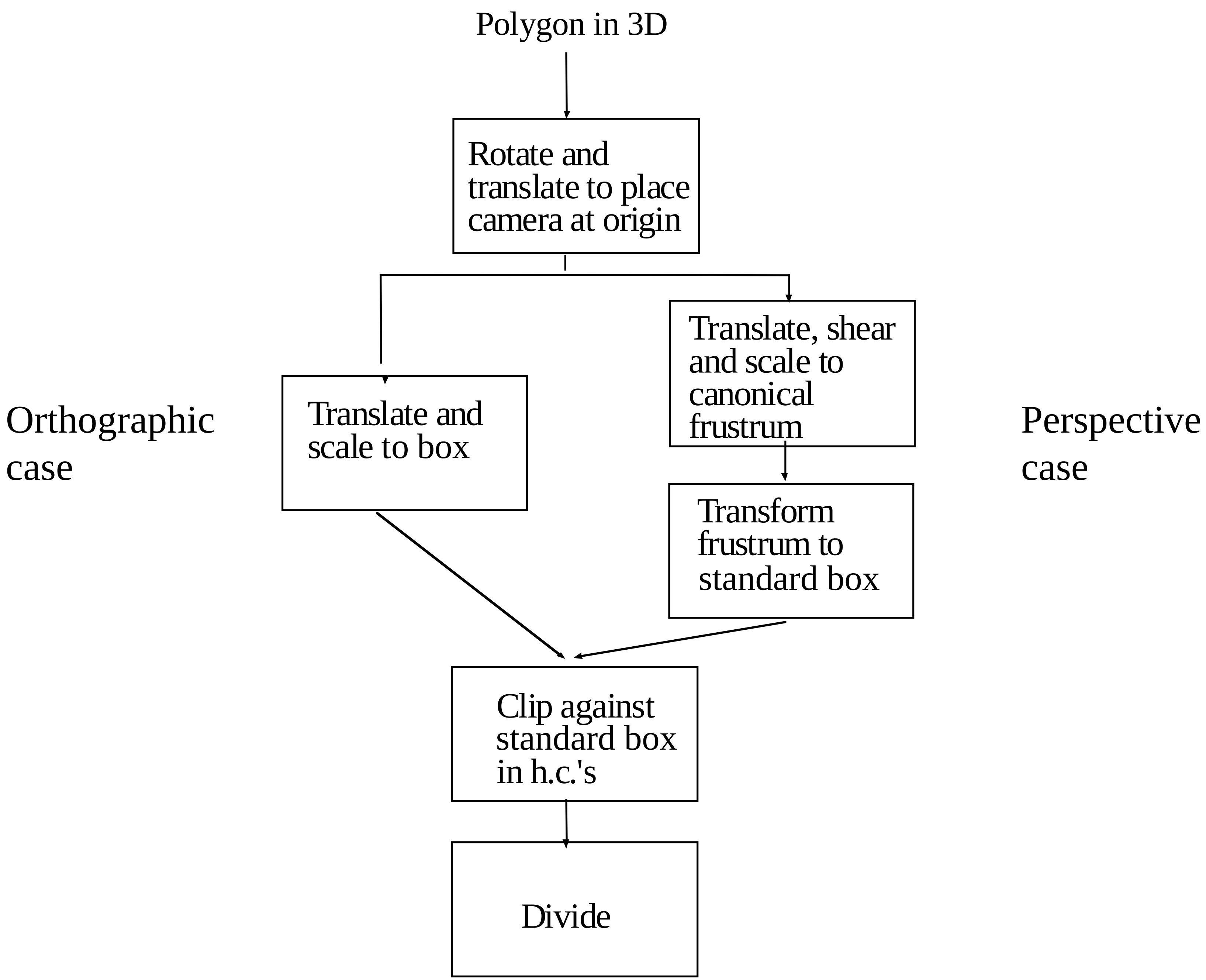


Plan A: Clip against canonical frustum (relatively easy--we chose the canonical frustum so that it would be easy)

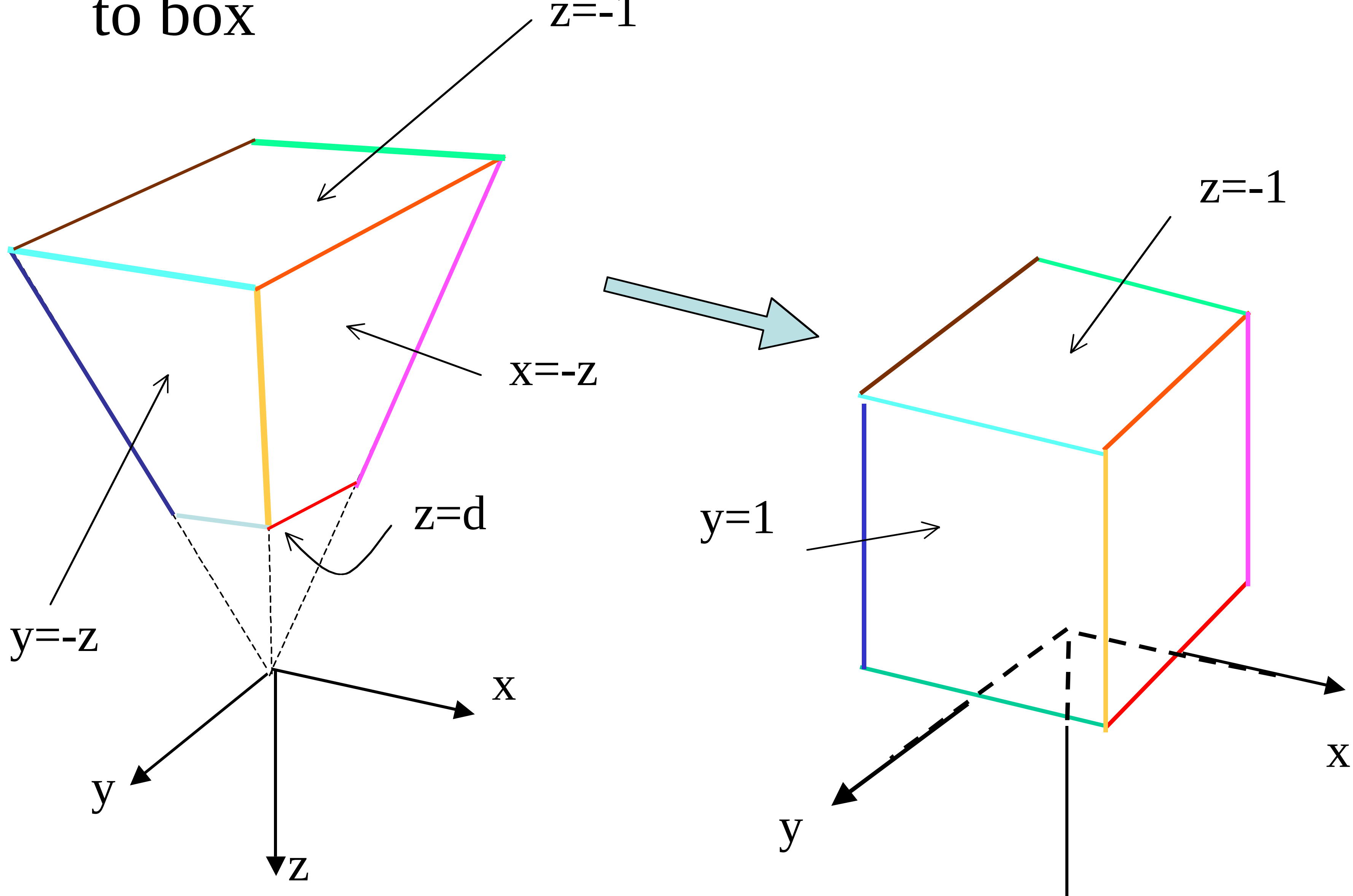
Plan B: Be even more clever. Further transform to cube and clip in homogenous coordinates.

Clipping in homogenous coords

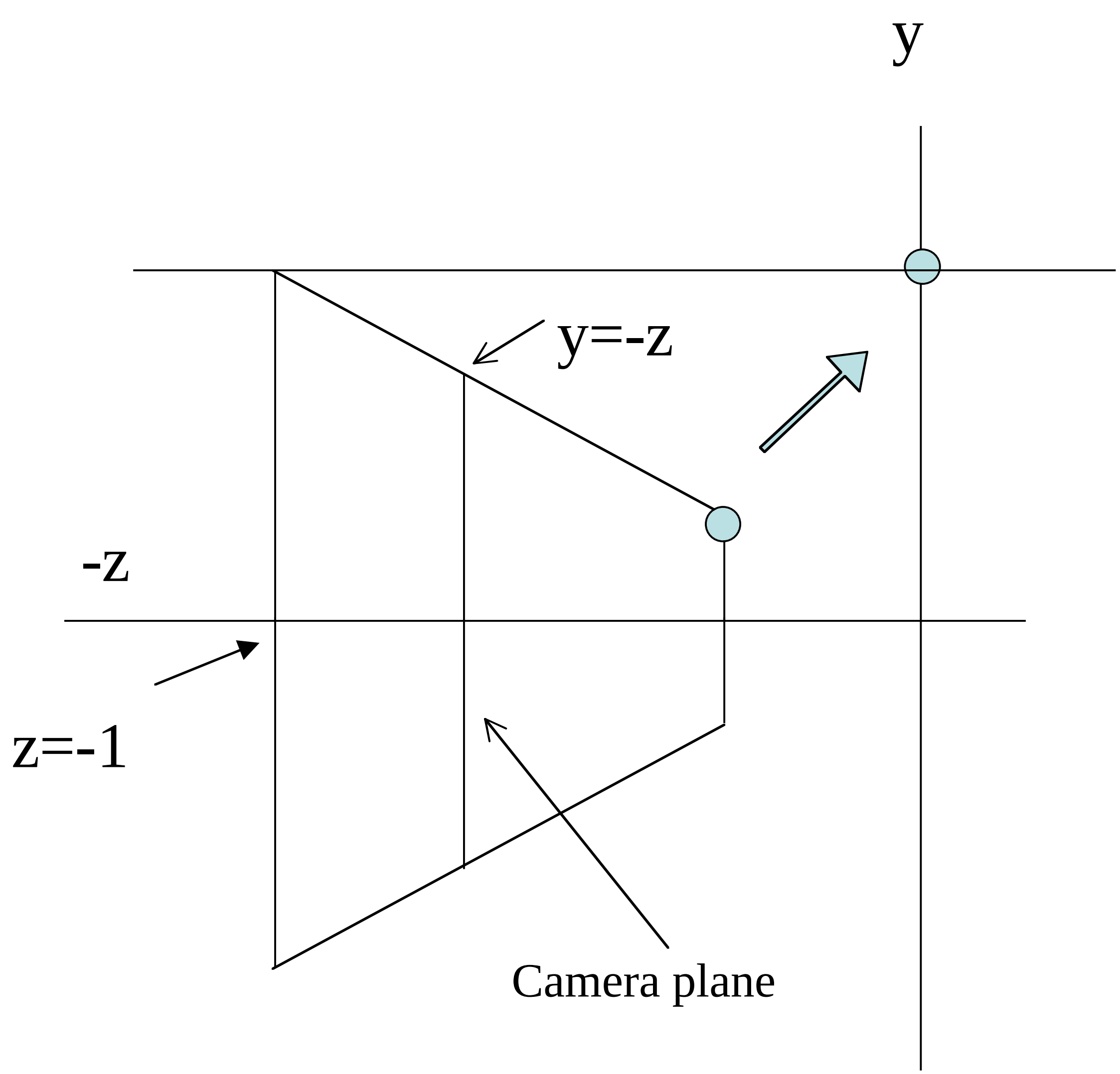
- For any camera, can turn the view frustrum into a regular parallelpiped (box). We will use the box bounded by $x = \pm 1$, $y = \pm 1$, $z = -1$, and $z = 0$.
- Advantages
  - Simplified clipping in homogenous coordinates
  - Extends to cases where we use homogenouse coordinates to represeent additional information (and W could be negative).
  - Can simplify visibility algorithms.
- Approach: clever use of homogenous coordinates


Polygon in 3D
Rotate and translate to place camera at origin
Orthographic case
Translate and scale to box
Translate, shear and scale to canonical frustrum
Perspective case
Transform frustrum to standard box
Clip against standard box in h.c.'s
Divide

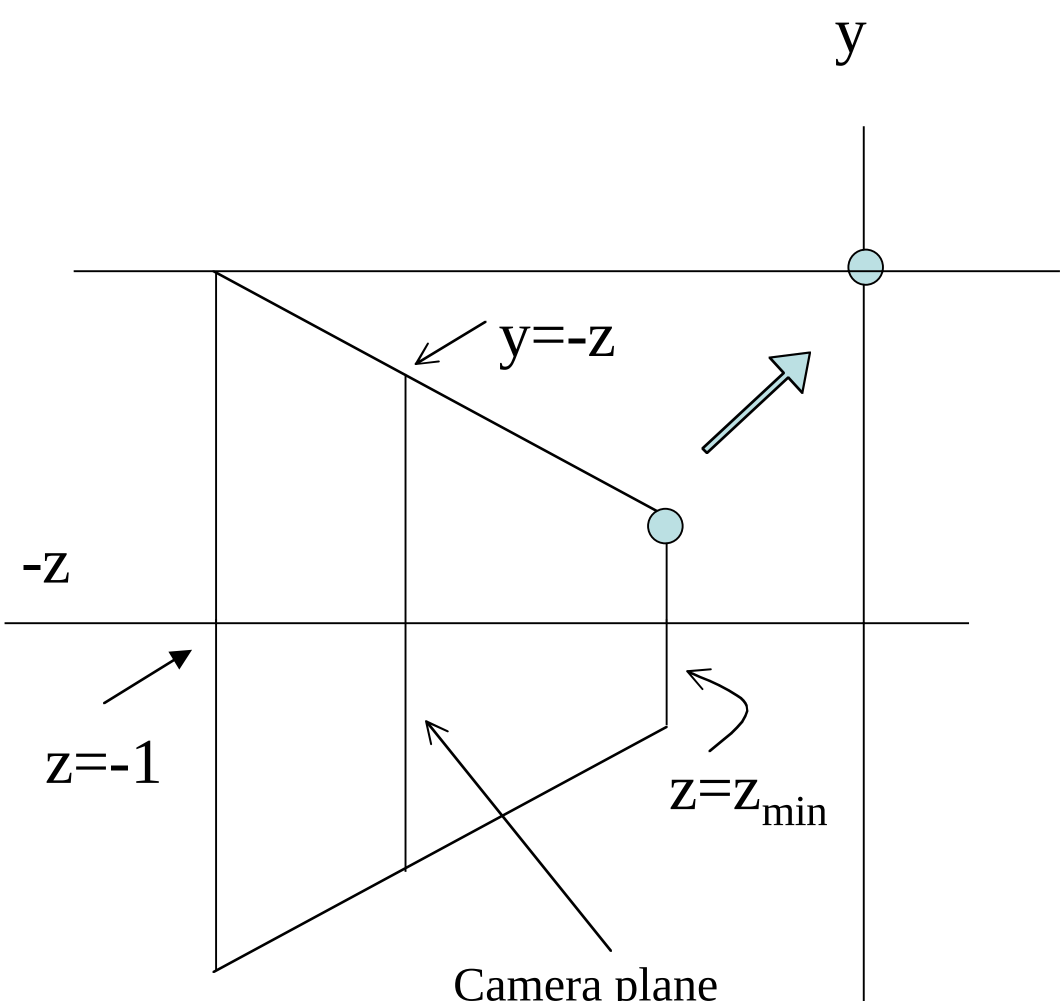
Transforming canonical frustum to box



Transforming canonical frustum to box



Transforming canonical frustum to box

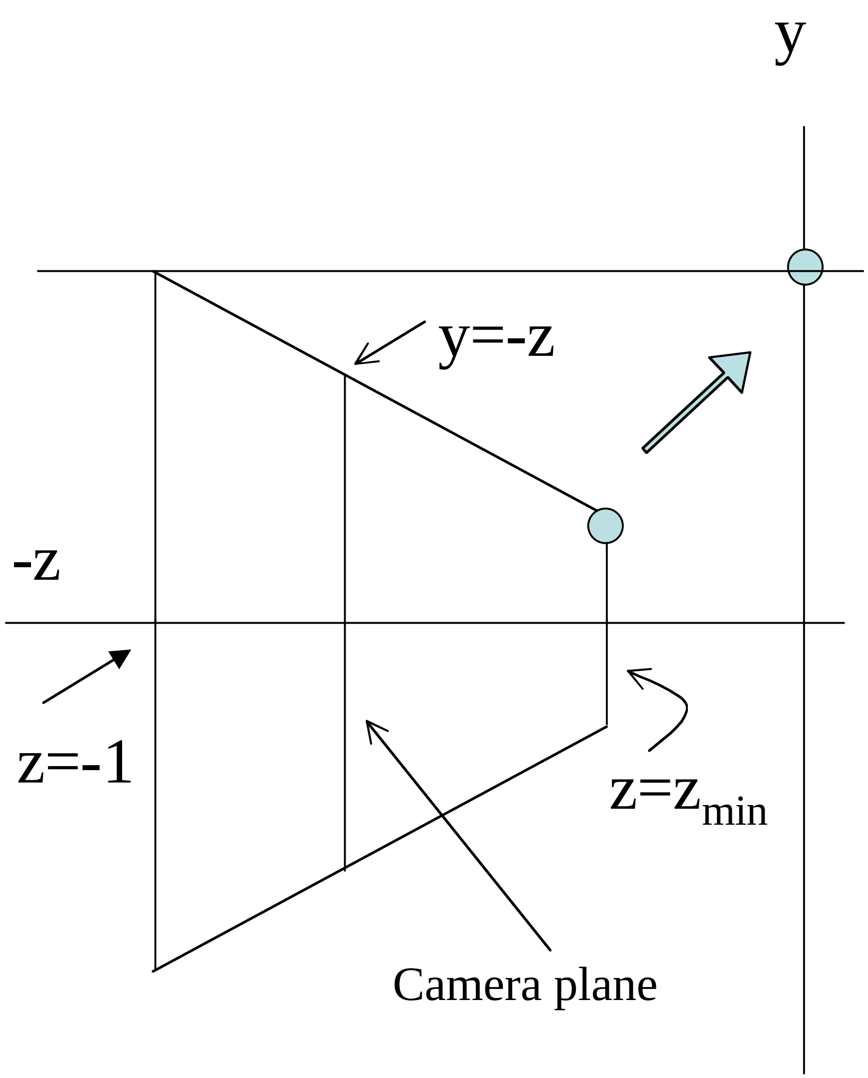


Mapping

x and y scale as a function of z.

We need $z= z_{min}$ to become z=0, so translate; but then need to rescale so that box has correct dimensions.

Transforming canonical frustum to box



$x' = x/(-z)$
$y' = y/(-z)$

Transformation is non-linear, but in h.c., can use w=-z.

Intuitively we would like
$z' = (z - z_{min}) / (1 + z_{min})$

But because we want x and y to work nicely in h.c., we accept

$z' = ((z - z_{min}) / (1 + z_{min}))/(-z)$

(Depths transform non-linearly)

In h.c.,

$x \Rightarrow x$

$y \Rightarrow y$

$z \Rightarrow (z - z_{min}) / (1 + z_{min})$

$1 \Rightarrow -z$

So, the matrix is ?

In h.c.,

$x \Rightarrow x$

$y \Rightarrow y$

$z \Rightarrow (z - z_{min}) / (1 + z_{min})$

$1 \Rightarrow -z$

So, the matrix is

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

Clipping in homogeneous coord.’s

- We have a cube, but its representation is in h.c., so we must divide
- Clipping requires a test for inside or out; in 2D, we used x>xmax, x<xmin, etc.
- In 3D, clipping against the cube, we could use x>1, x<-1, etc.
- But to do this, we would have to convert to cartesian coords by division.
- Dividing before clipping is inefficient if many points are excluded, so we clip in h.c.’s.

Clipping in homogeneous coord.'s

- Write h.c.'s in caps, ordinary coord's in lowercase.
- Consider case x>1, x<-1
- Rearrange clipping inequalities:

$$\begin{array}{cccc} \left(\dfrac{X}{W}\right) > 1 & & X > W, & & X < W, \\ & \text{becomes} & X < -W, & \text{AND} & X > -W, \\ \left(\dfrac{X}{W}\right) < -1 & & W > 0 & & W < 0 \end{array}$$