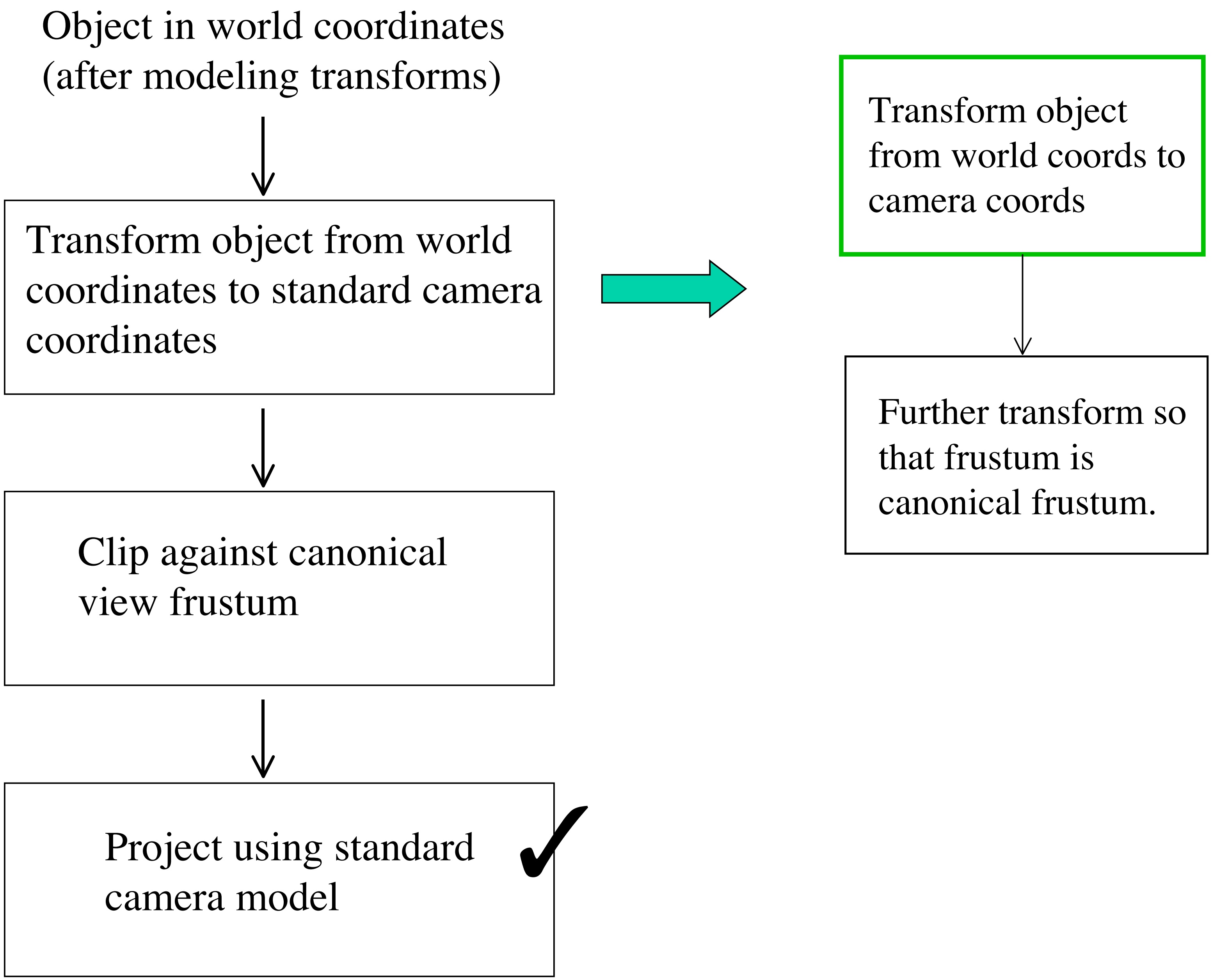

Object in world coordinates (after modeling transforms)
Transform object from world coordinates to standard camera coordinates
Clip against canonical view frustum
Project using standard camera model
Transform object from world coords to camera coords
Further transform so that frustum is canonical frustum.

Transform object
from world coords to
camera coords

Step 1. Translate the camera at VRP to the world origin. Call this T_1.

Translation vector is simply negative VRP.

(We are changing the coordinate system of the world, which is the same thing mathematically as moving the camera. We want object world coordinates to **change** so that the camera location **becomes** the origin).

Transform object
from world coords to
camera coords

Step 2. Rotate camera coordinate frame (in w.c.) so that so that **u** is **x**, **v** is **y**, and **n** is **z**. The matrix is ?

(We are changing the coordinate system of the world, which is the same thing mathematically as moving the camera. We want object world coordinates to **change** so that the camera axis **becomes** the standard axis—e.g, **u** becomes (1,0,0), **v** becomes (0,1,0) and **n** becomes (0,0,1)).

Transform object from world coords to camera coords

Step 2. Rotate camera coordinate frame (in w.c.) so that so that **u** is **x**, **v** is **y**, and **n** is **z**. The matrix is:

$$\begin{vmatrix} \mathbf{u}^T & 0 \\ \mathbf{v}^T & 0 \\ \mathbf{n}^T & 0 \\ 0\ 0\ 0 & 1 \end{vmatrix}$$

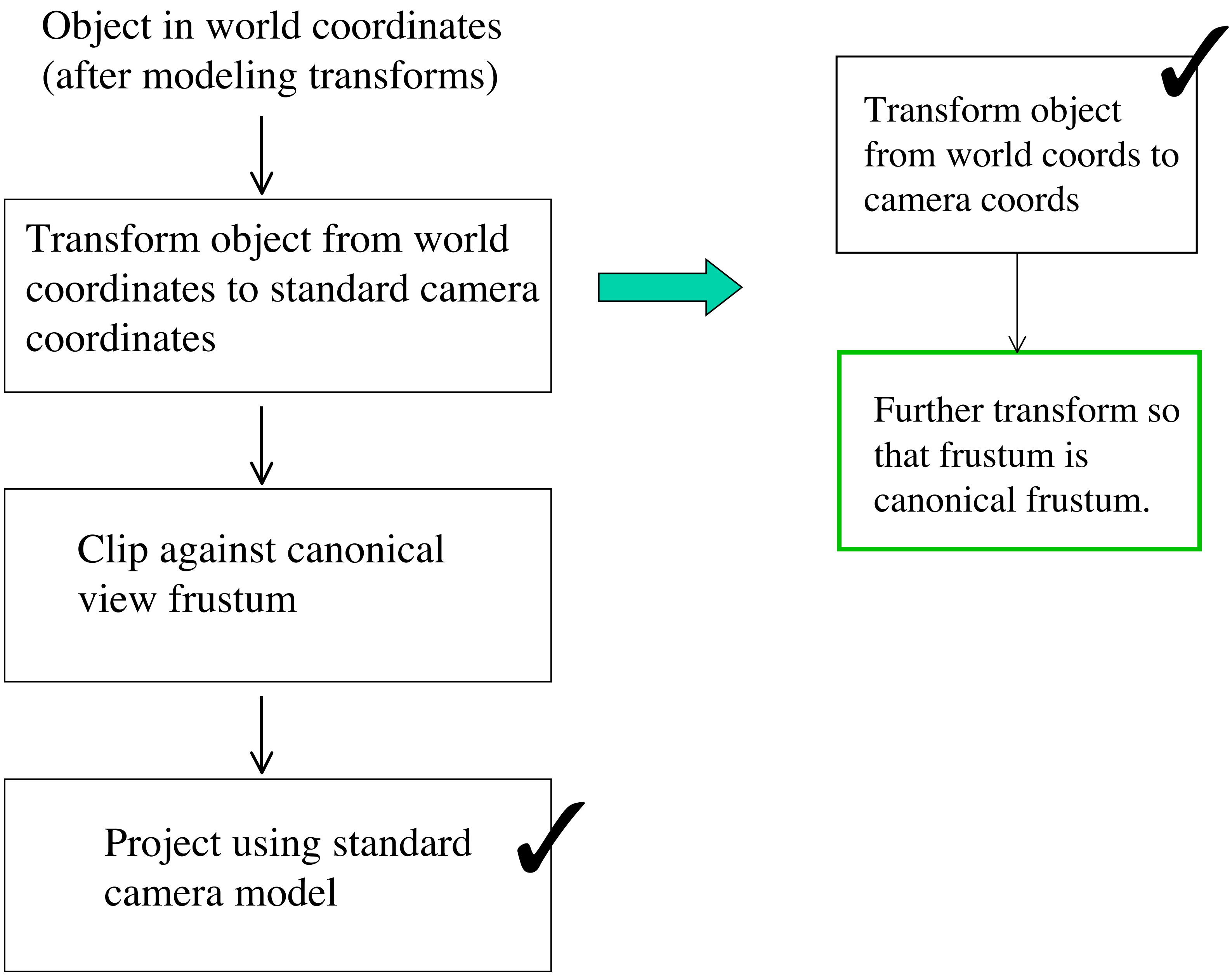
(why?)

Transform object from world coords to camera coords

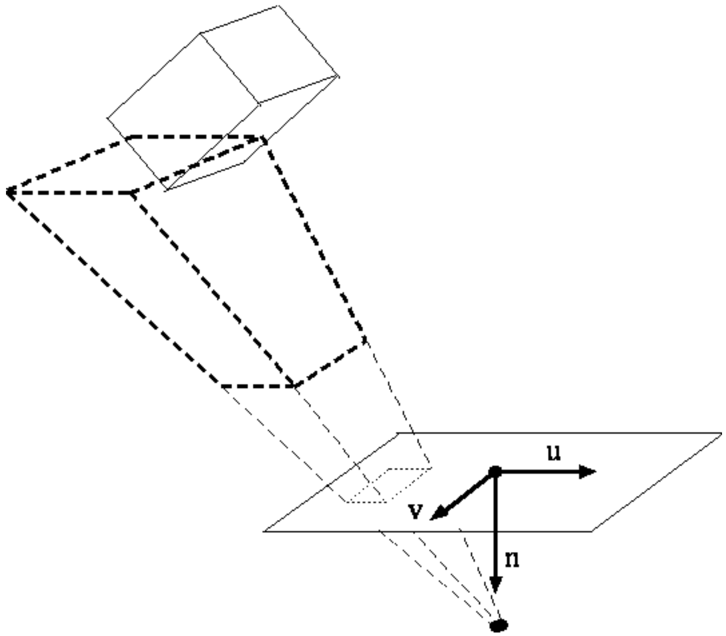
$$\begin{vmatrix} \mathbf{u}^T & 0 \\ \mathbf{v}^T & 0 \\ \mathbf{n}^T & 0 \\ 0\ 0\ 0 & 1 \end{vmatrix} \mathbf{u} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

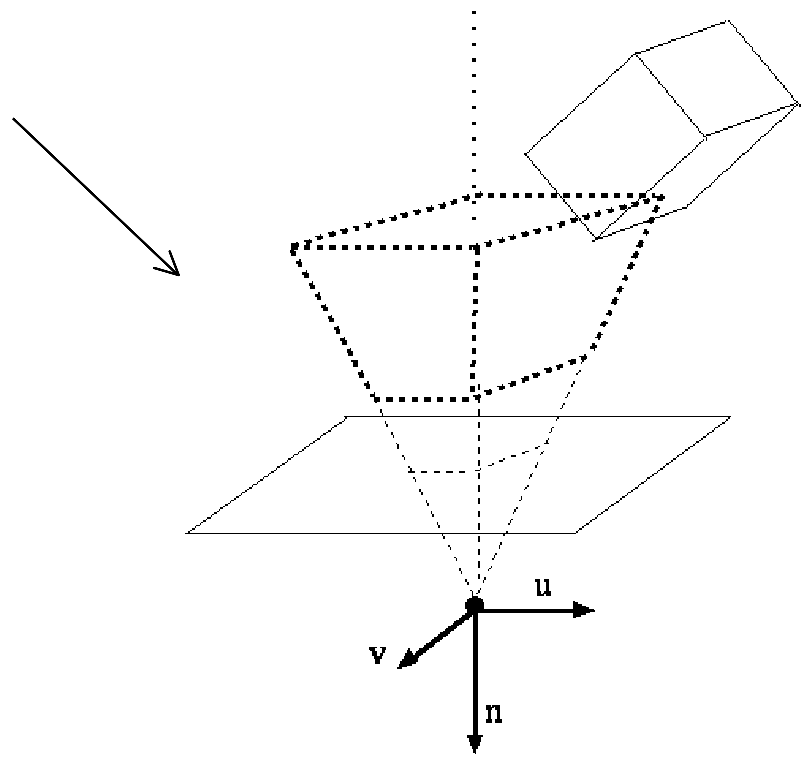
In the current coords (world shifted so that VPR is at origin): **u** maps into the X-axis unit vector (1,0,0,0) which is what we want.

(Similarly, **v**-->Y-axis unit vector, **n**-->Z-axis unit vector)


Object in world coordinates (after modeling transforms)
Transform object from world coordinates to standard camera coordinates
Clip against canonical view frustum
Project using standard camera model
Transform object from world coords to camera coords
Further transform so that frustum is canonical frustum.

Mapping the view frustum to the canonical view frustum





Further transform so that frustum is canonical frustum.

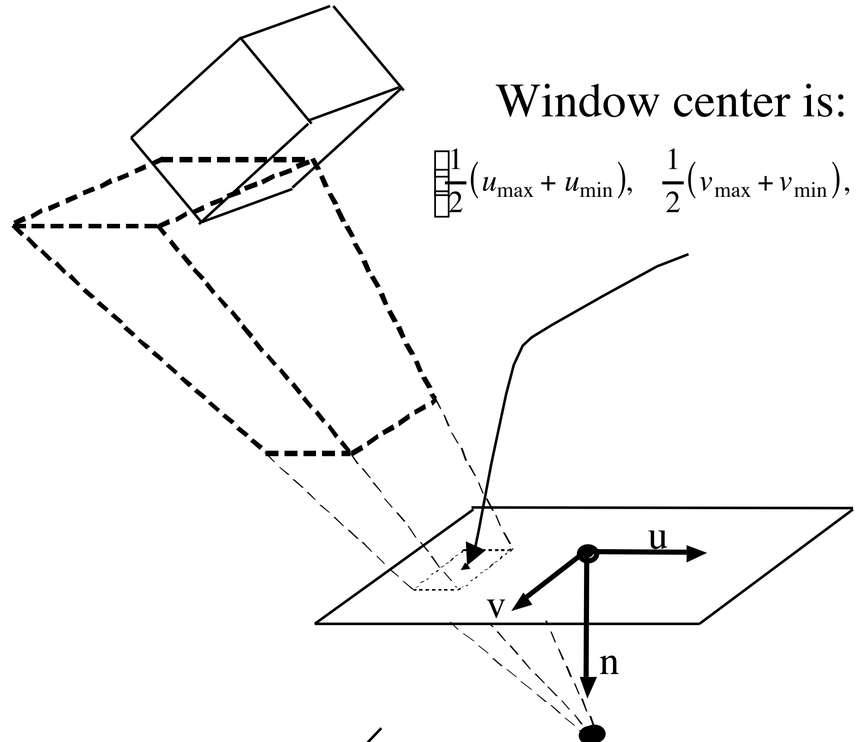
Since we are now in camera coordinates, we will often refer to them as (x,y,z) not (u,v,n).

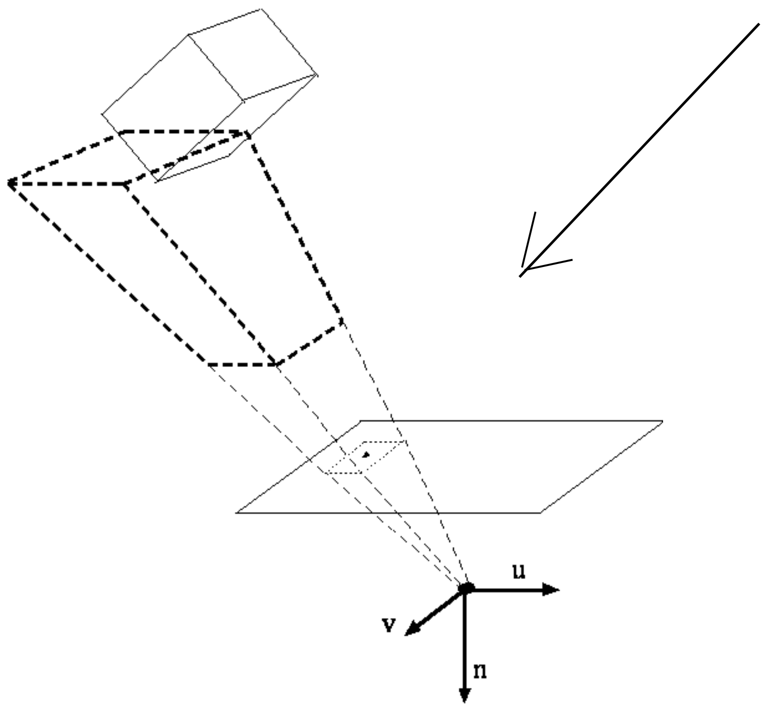
1. Translate focal point to origin
2. Shear so that central axis of frustum lies along the z axis
3. Scale x, y so that faces of frustum lie on conical planes
4. Isotropic scale so that back clipping plane lies at z=-1

Step 1: Translate focal point (PRP) to origin; call this translation T_2. Since we have PRP in camera coordinates (where we now are), the translation vector is simply negative PRP. In particular, in the very common case where PRP is (0,0,f), the translation vector is (0,0,-f)

Window center is:

$$\left[\frac{1}{2}(u_{\max}+u_{\min}),\quad \frac{1}{2}(v_{\max}+v_{\min}),\quad 0\right]$$

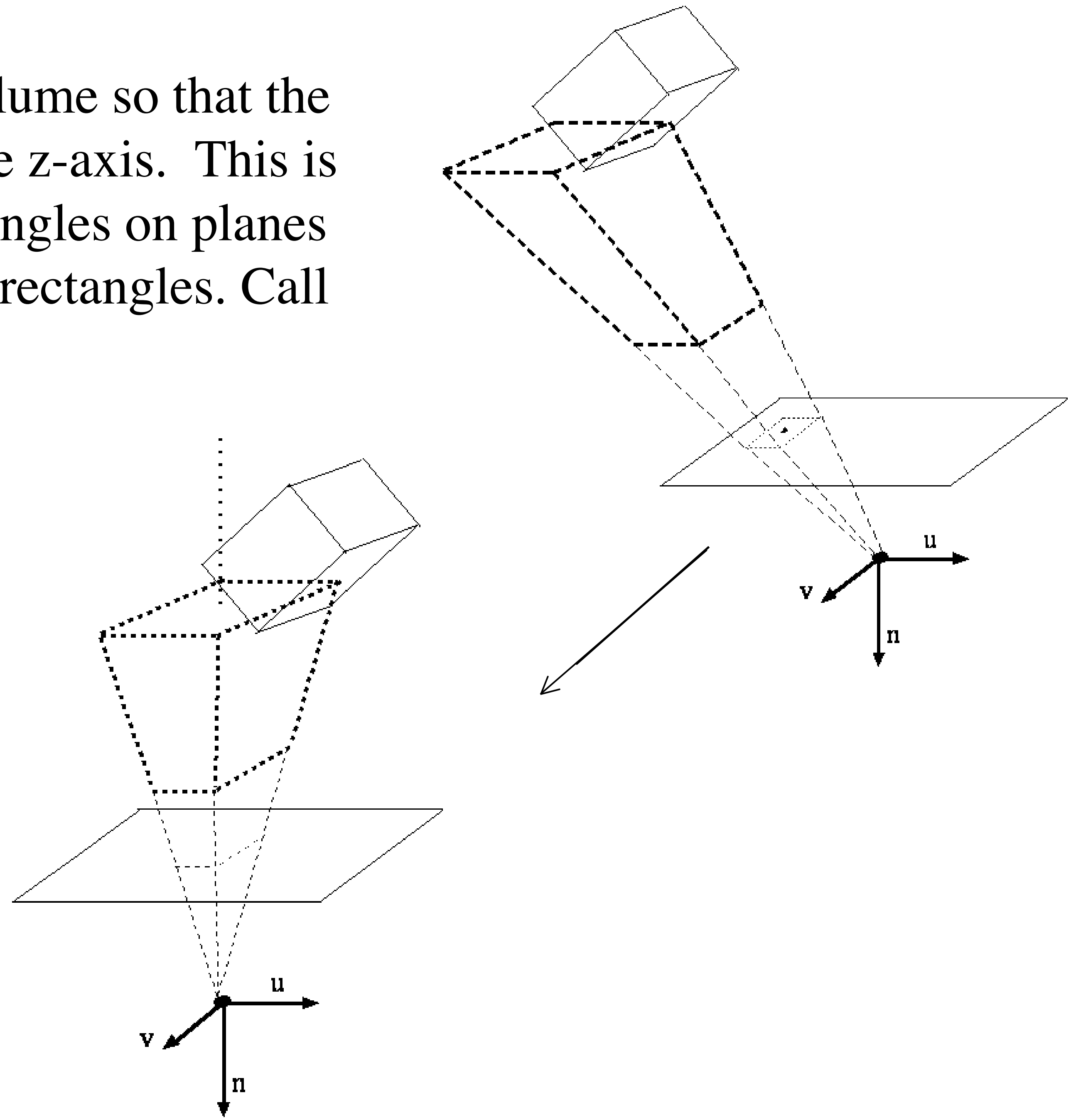




Window center is now:

$$\left[\frac{1}{2}(u_{\max}+u_{\min}),\quad \frac{1}{2}(v_{\max}+v_{\min}),\quad -f\right]$$

Step 2: Shear this volume so that the central axis lies on the z-axis. This is a shear, because rectangles on planes z=constant must stay rectangles. Call this shear S_1



Shear S_1 takes previous window midpoint $\left[\frac{1}{2}(u_{\max}+u_{\min}),\ \frac{1}{2}(v_{\max}+v_{\min}),\ -f\right]$ to (0, 0, -f) - this means that matrix is

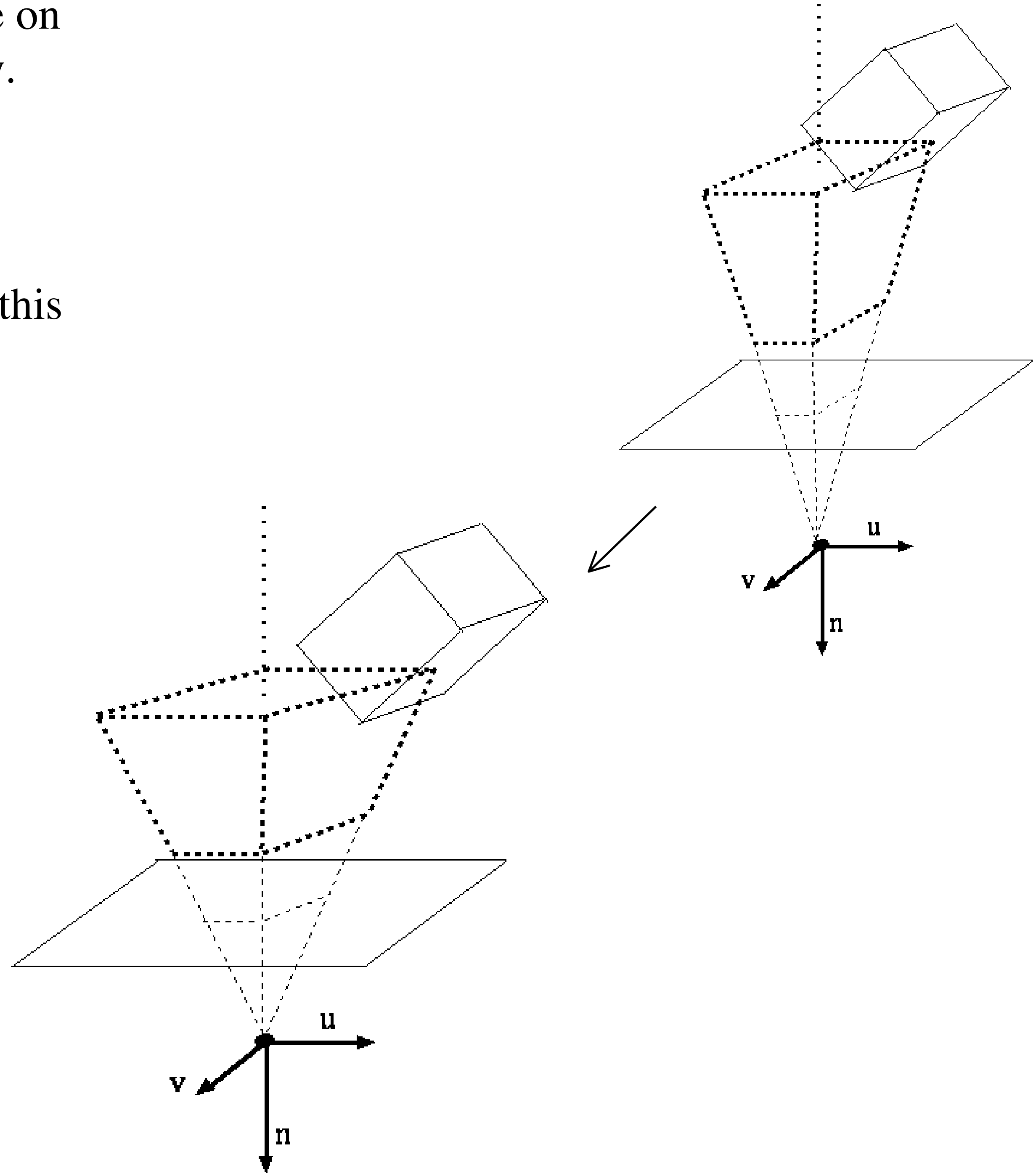
?

Shear S_1 takes previous window midpoint $\left(\frac{1}{2}(u_{\max}+u_{\min}),\ \frac{1}{2}(v_{\max}+v_{\min}),\ -f\right)$ to (0, 0, -f) - this means that matrix is:

$$\begin{bmatrix} 1 & 0 & \frac{(u_{\min}+u_{\max})}{2f} & 0 \\ 0 & 1 & \frac{(v_{\min}+v_{\max})}{2f} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

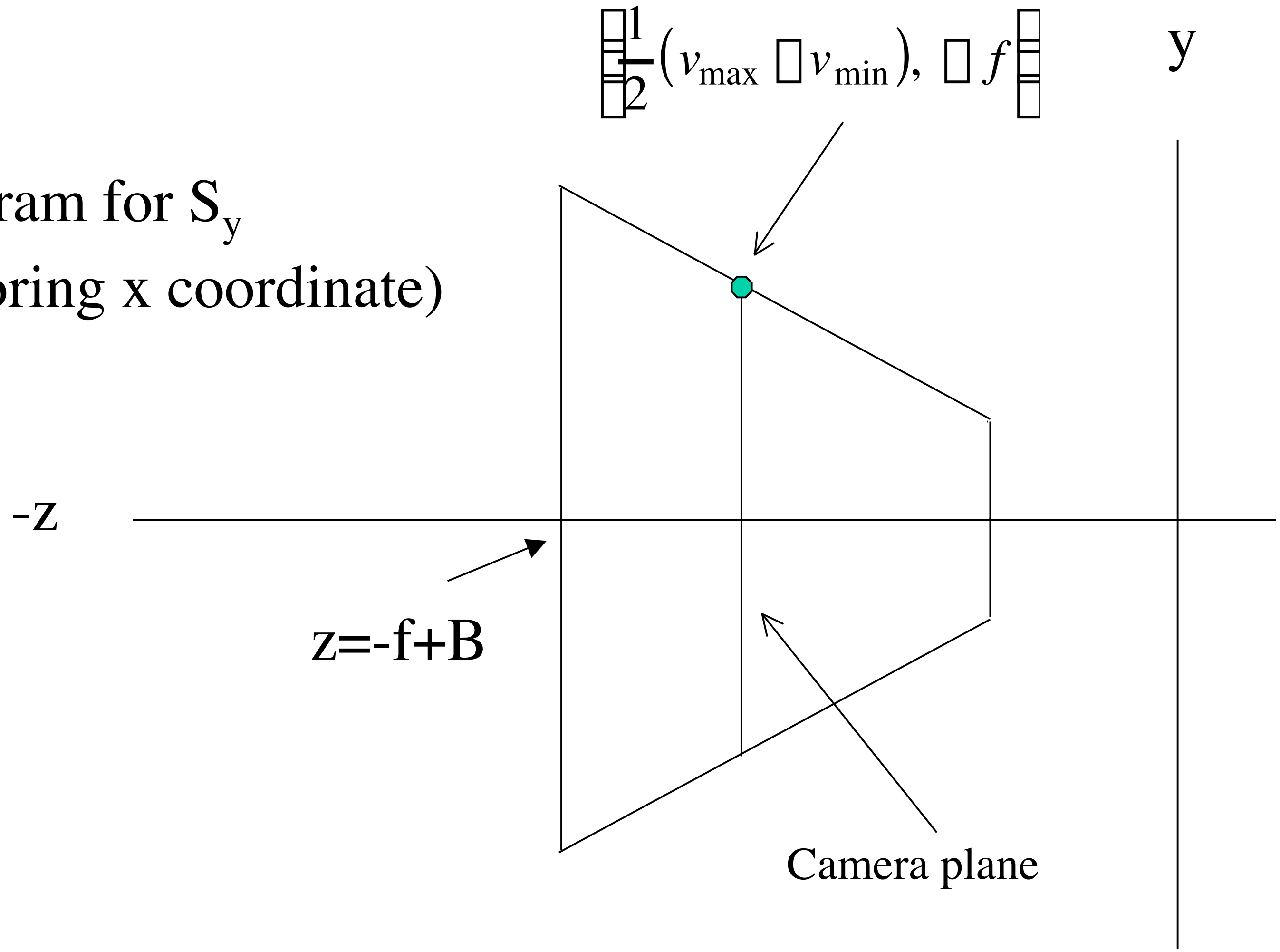
3. Scale x, y so that planes are on z=x, z=-x and z=y and z=-y. Call this scale Sc_1

4. Isotropic scale so that far clipping plane is z=-1; call this scale Sc_2



3. Scale x, y so that planes are on z=x, z=-x and z=y and z=-y. Call this scale Sc_1

Diagram for S_y
(ignoring x coordinate)



4. Scale x, y so that planes are on z=x, z=-x and z=y and z=-y. Call this scale Sc_1

$$\left[\frac{1}{2}\left(v_{max} - v_{min}\right),\ -f\right] \Longrightarrow \quad y=-z$$

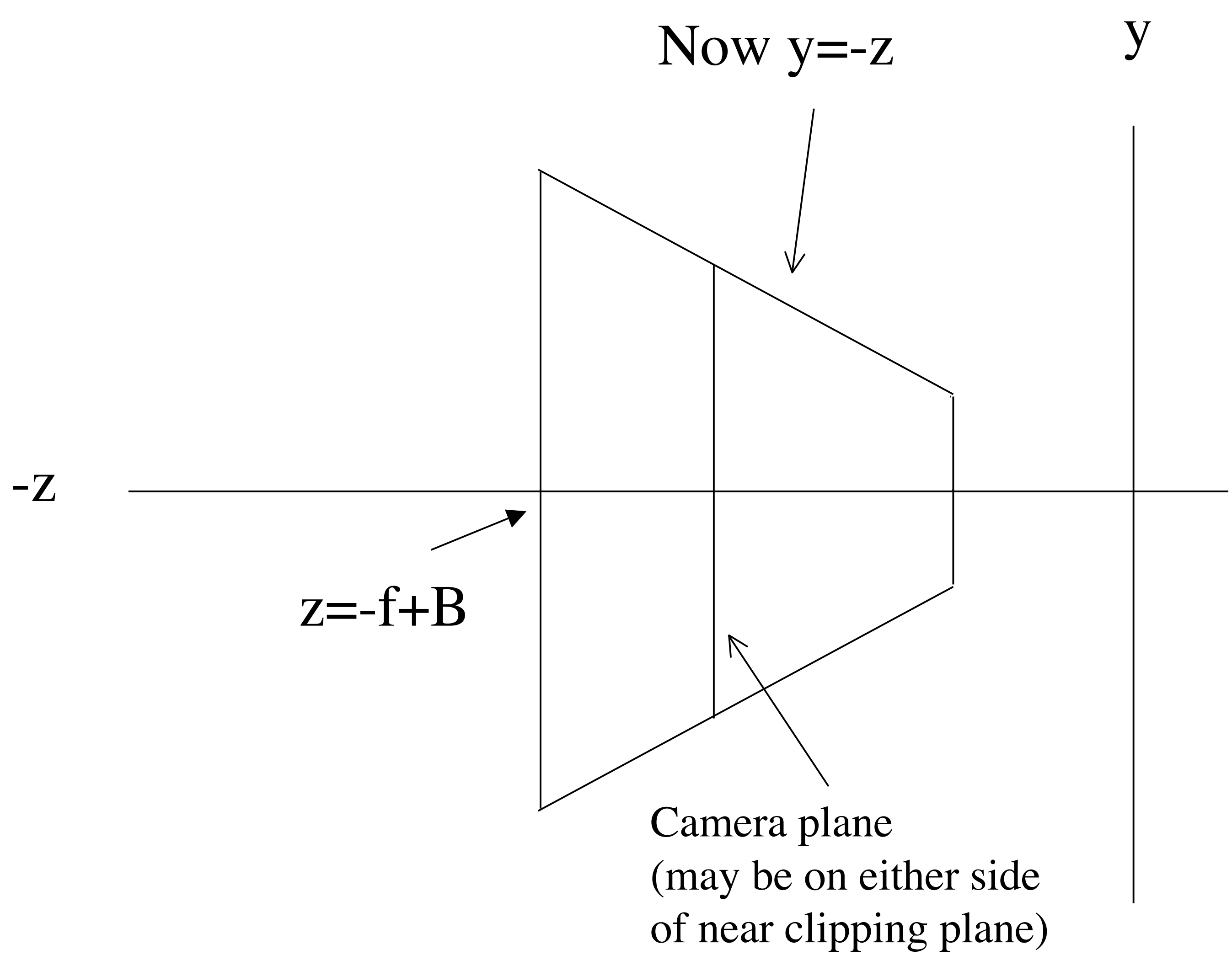
$$k_y \frac{1}{2}\left(v_{max} - v_{min}\right) = f$$

$$k_y = \frac{2f}{\left(v_{max} - v_{min}\right)}$$

(k_y is y scale factor)

$$\mathbf{Sc}_1 = \begin{vmatrix} \dfrac{2f}{(u_{\max} - u_{\min})} & 0 & 0 & 0 \\ 0 & \dfrac{2f}{(v_{\max} - v_{\min})} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

5. Now isotropic scale so that far clipping plane is z=-1; call this scale Sc_2



5. Now isotropic scale so that far clipping plane is $z=-1$; call this scale Sc_2

Currently, at far clipping plane, $z=-f+B$

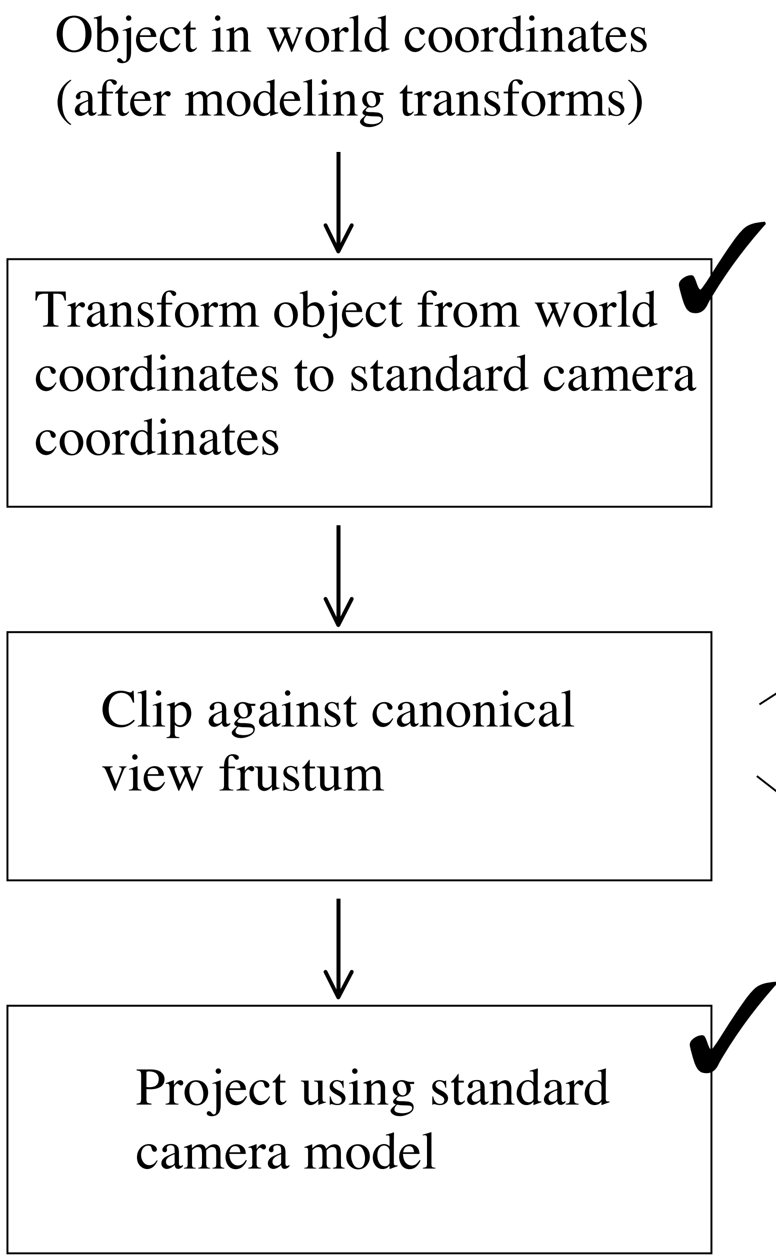
Want a factor k so that $k(-f+B)=-1$

So, $k = -1 / (-f + B) = 1 / (f - B)$

(Note that B is negative, and k is positive)

3D Viewing Pipeline

$$\begin{pmatrix}\text{Point in}\\ \text{canonical}\\ \text{camera}\\ \text{coordinates}\end{pmatrix} \quad Sc_2 Sc_1 S_1 T_2 R_1 T_1 \quad \begin{pmatrix}\text{Point in}\\ \text{world}\\ \text{coordinates}\end{pmatrix}$$



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.

Plan A: Clipping against the canonical frustum

2D algorithms are easily extended. For example, for Cohen Sutherland we use the following 6 out codes:

$y>-z \quad y<z \quad x>-z \quad x<z \quad z<-1 \quad z>z_{min}$

($z_{min} = (f-F)/(B-f)$)

Recall C.S for segments

Compute out codes for endpoints

While not trivial accept and not trivial reject:

 Clip against a problem edge (one point in, one out)

 Compute out codes again

Return appropriate data structure

Clipping against the canonical frustum

Clipping polygons in 3D against canonical frustum planes is simpler and more efficient than the general case.

Recall the S.C. gives four cases:

Polygon edge crosses clip **plane** going from out to in

- emit crossing, next vertex

Polygon edge crosses clip **plane** going from in to out

- emit crossing

Polygon edge goes from out to out

- emit nothing

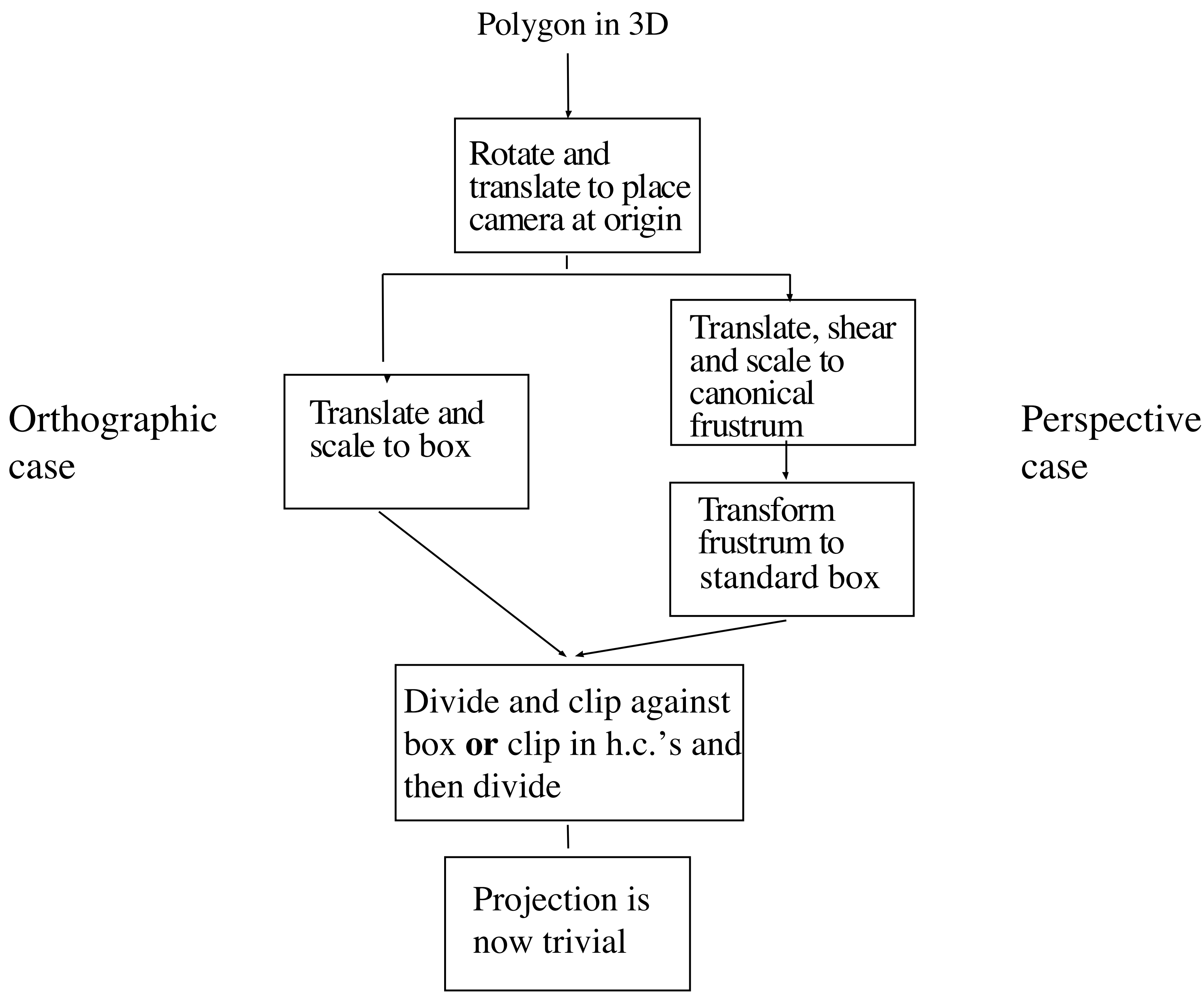
Polygon edge goes from in to in

- emit next vertex

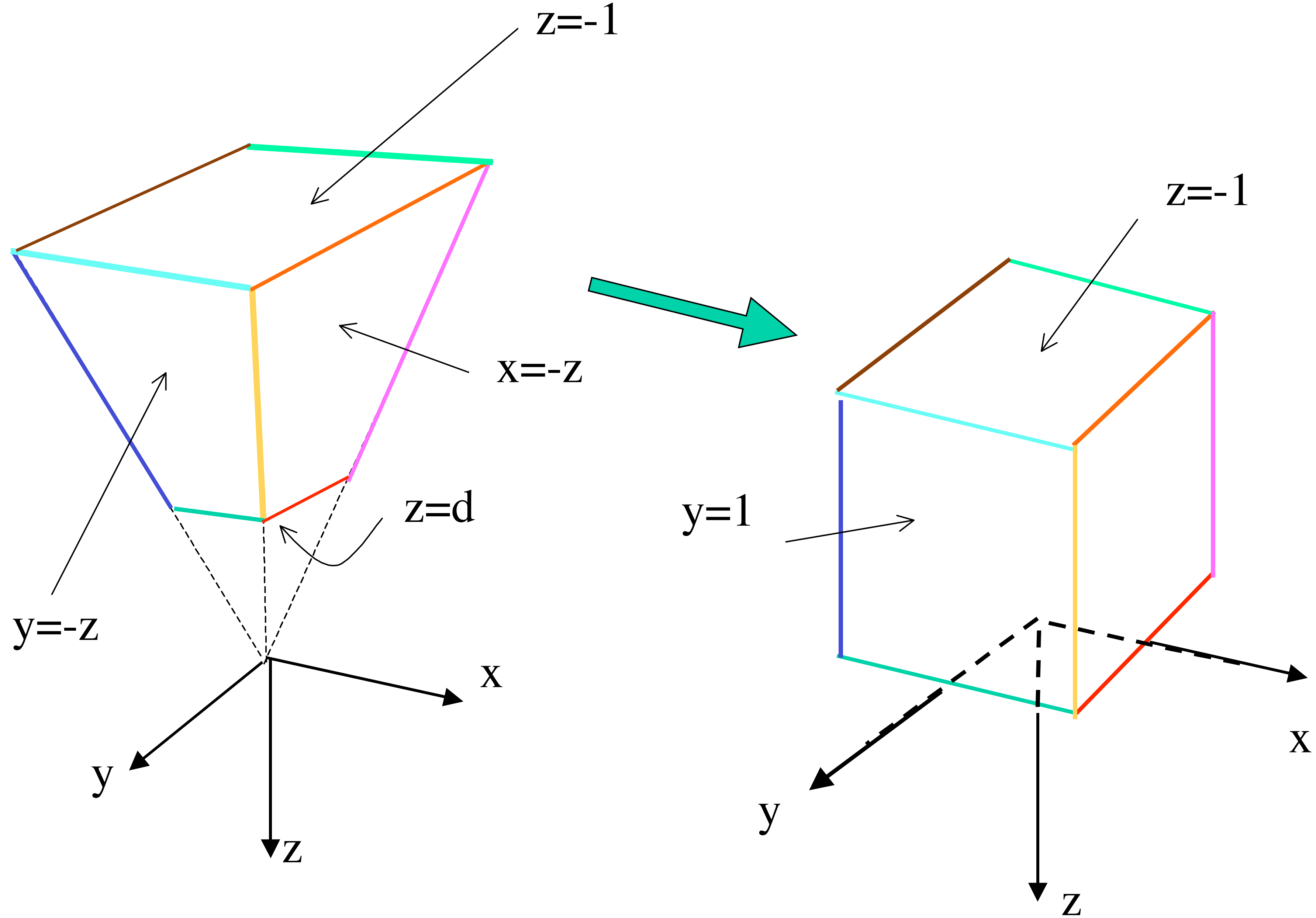
(The above is from before, just change “edge” to “plane”)

Plan B: Clipping in homogenous coords

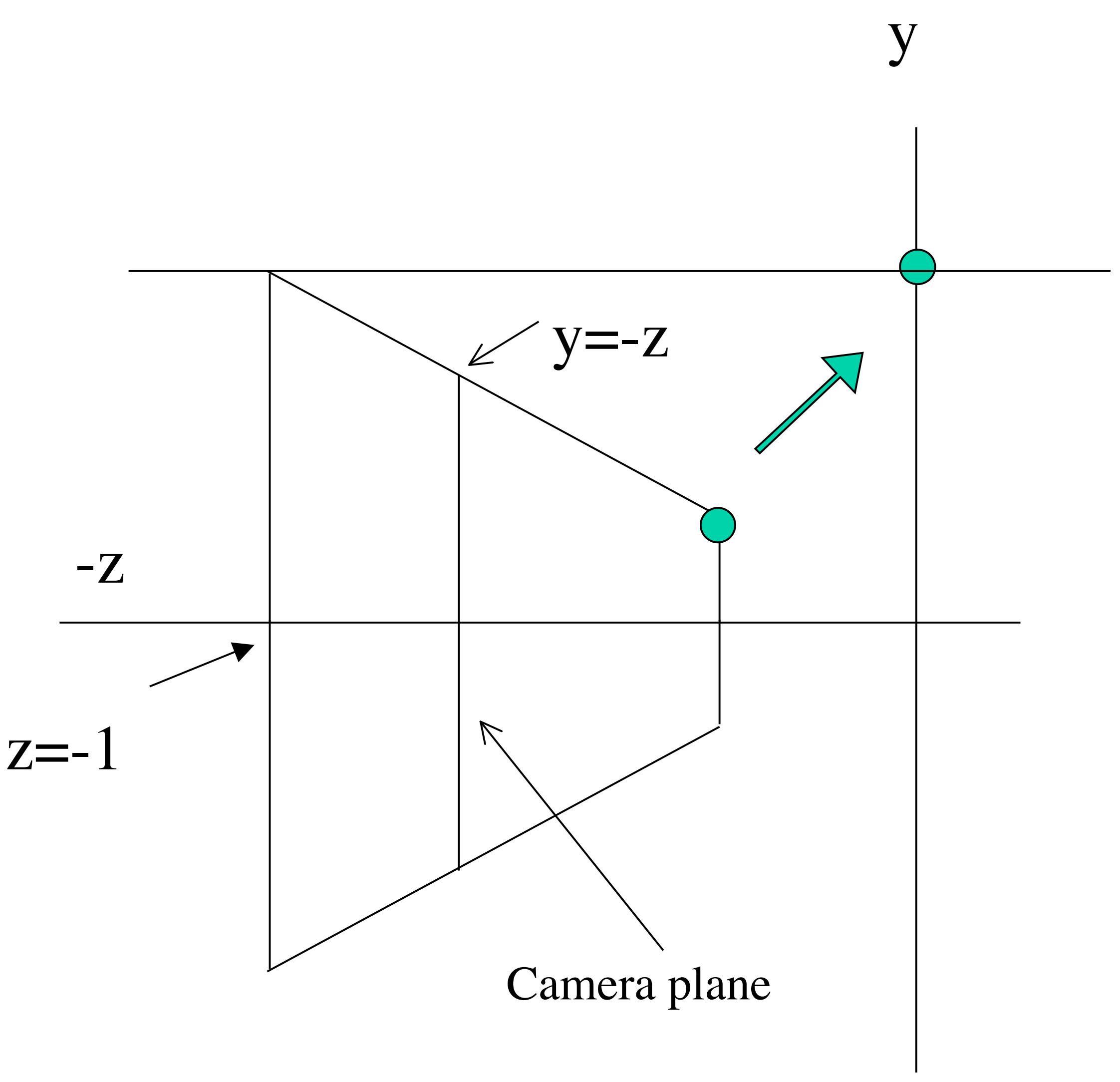
- For any camera, can turn the view frustrum into a regular parallelepiped (box). We will use the box bounded by $x = \pm 1$, $y = \pm 1$, $z = -1$, and $z = 0$.
- Advantages
  - Simplified clipping in homogenous coordinates
  - Extends to cases where we use homogenous coordinates to represent additional 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
Translate and scale to box
Translate, shear and scale to canonical frustrum
Orthographic case
Perspective case
Transform frustrum to standard box
Divide and clip against box **or** clip in h.c.'s and then divide
Projection is now trivial

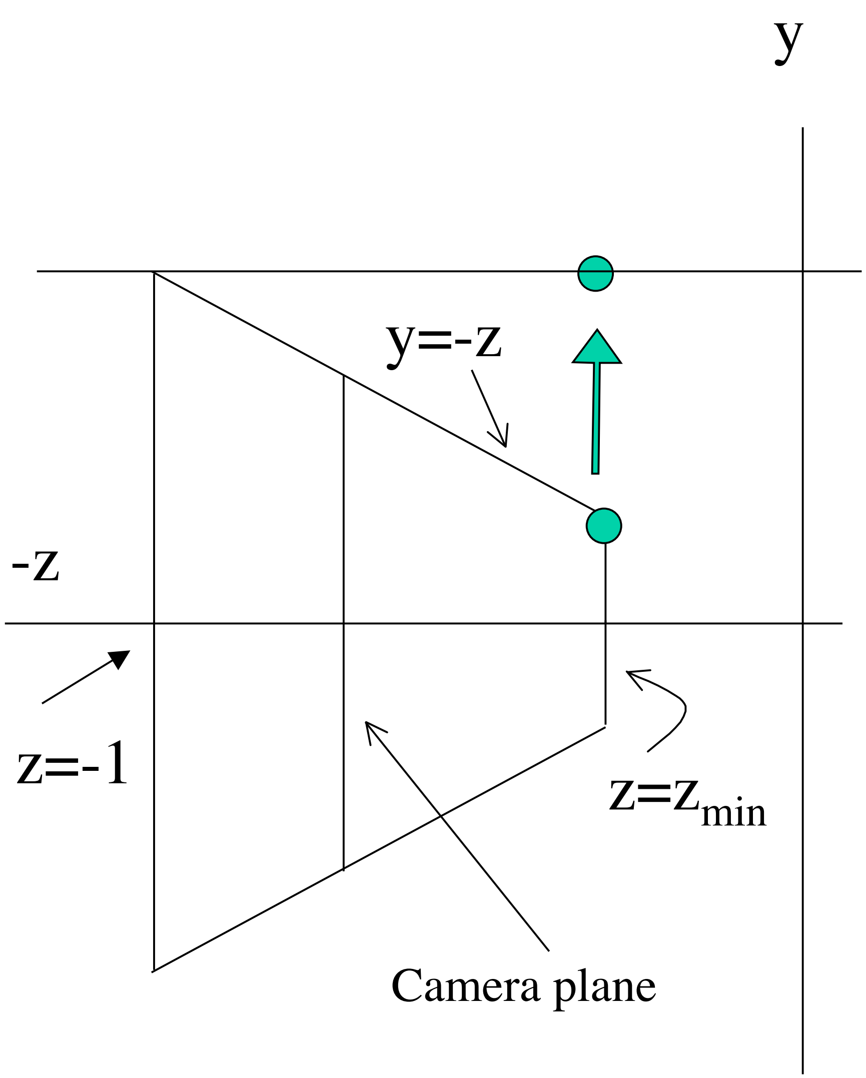
Transforming canonical frustum to box



Transforming canonical frustum to box



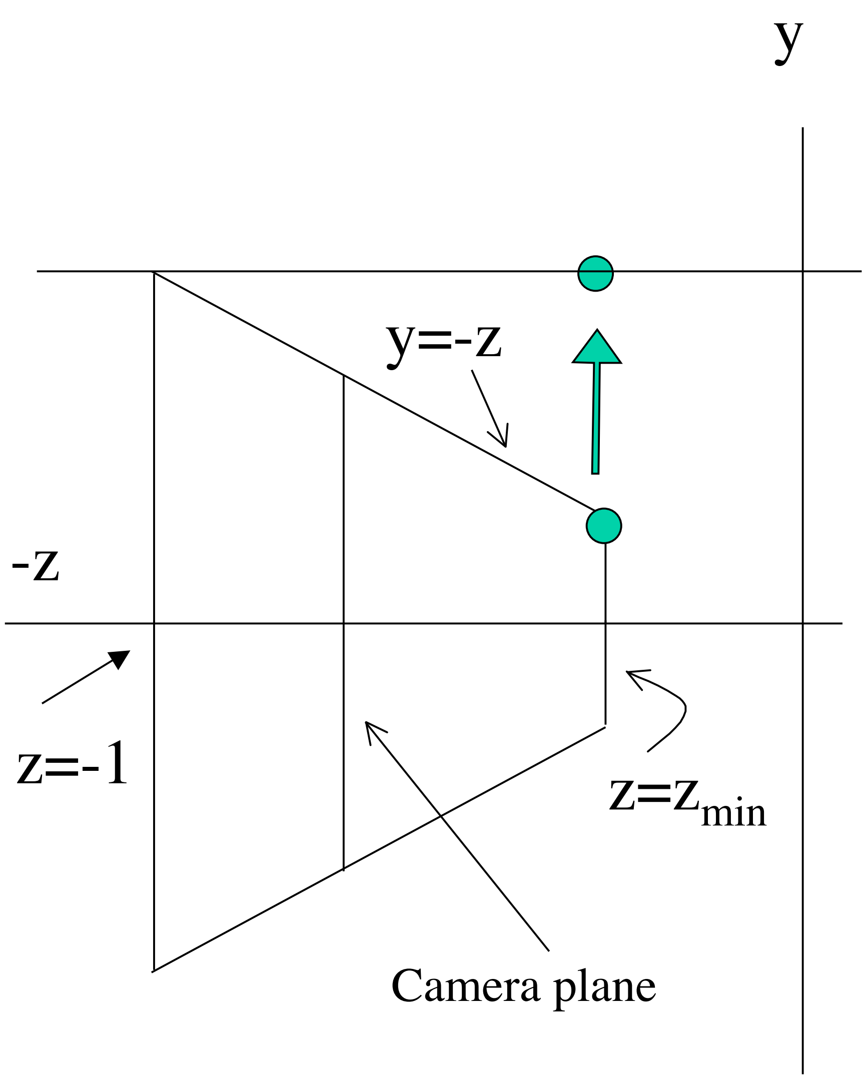
Transforming canonical frustum to box



The picture should suggest an appropriate scaling for y.

It is ?

Transforming canonical frustum to box



On top, y—> 1, so scaling is (1/y) Recall that y=-z there.

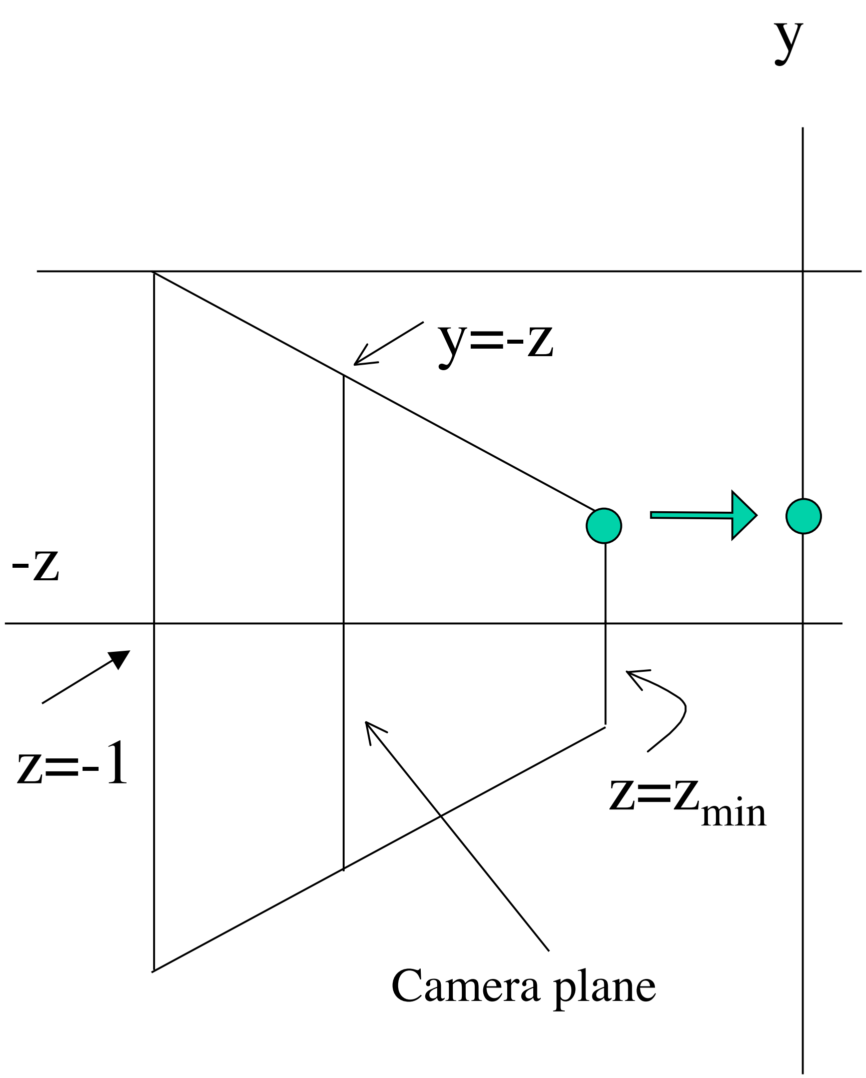
On bottom, y—> -1 so scaling is (-1/y). Recall that y=z there.

So scaling is y'= y/(-z)

Similarly, x' = x/(-z)

Transformation is **non-linear**, but in h.c., we can make w = (-z).

Transforming canonical frustum to box



For z, we translate near plane to origin. But now box is too small. Specifically it has z dimension $(1 + z_{min})$ (recall z_{min} is negative)

So we have an extra scale factor $1 / (1 + z_{min})$ and thus $z' = (z - z_{min}) / (1 + z_{min})$

But we want x and y to work nicely in h.c., with w=-z, so we use

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

(Thus in our box, depth transforms **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=>x

y=>y

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

1=>-z

So, the matrix is

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

Mapping to standard view volume (additional comments)

- The mapping from $[z_{min}, -1]$ to $[0,-1]$ is non-linear. (Of course, there exists a linear mapping, but not if we want everything else to work out nicely in h.c.).
- So a change in depth of $\triangle$ D at the near plane maps to a larger depth difference in screen coordinates than the same $\triangle$ D at the far plane.
- But order is preserved (important!); the function is monotonic (proof?).
- And lines are still lines (proof?) and planes are still planes (important!).

Transforming canonical frustum to box