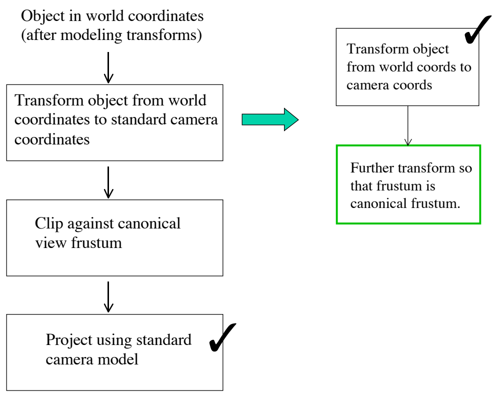
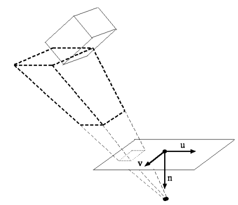
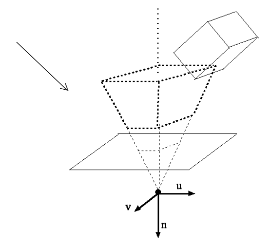



Mapping the view frustum to the canonical view frustum




Further transform so that frustum is canonical frustum.

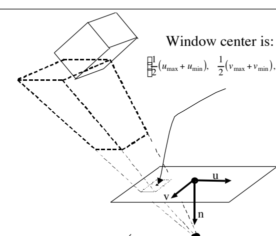
Since we are now in camera coordinates, we will often refer to them as (x,y,z) not (u,v,n).

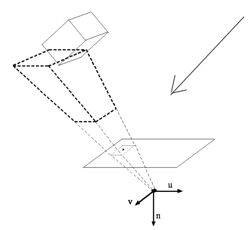
1. Translate focal point to origin
2. Shear so that central axis of frustum lies along the z axis
3. Scale x, y so that faces of frustum lie on conical planes
4. Isotropic scale so that back clipping plane lies at z=-1

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:

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

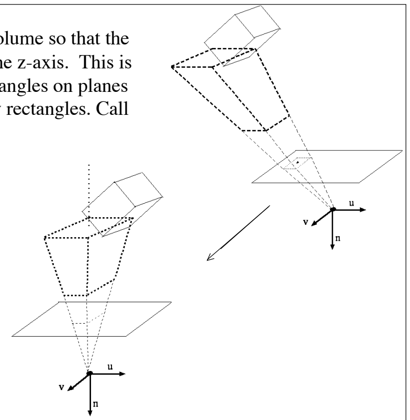



Window center is now:

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

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

?

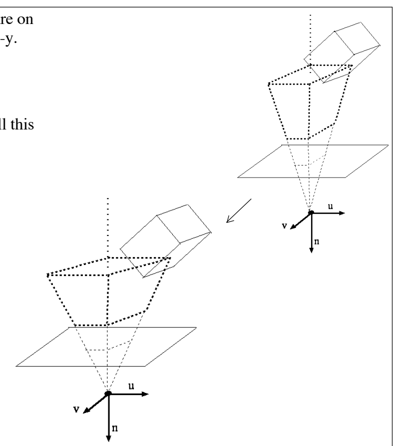
Shear $S_1$ takes previous window midpoint $\left(\frac{1}{2}(u_{max}+u_{min}),\ \frac{1}{2}(v_{max}+v_{min}),\ -f\right)$ to (0, 0, -f) - this means that matrix is:

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

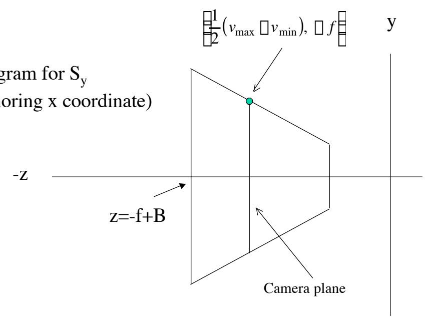
Note that the size of a rectangle in the image plane does not change.

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 $Sc_1$




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

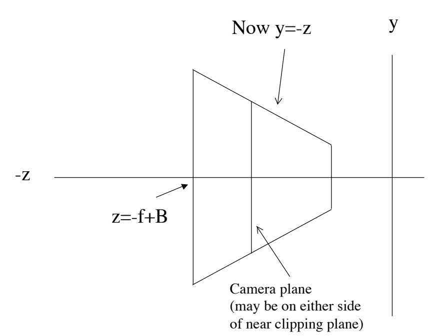
$$\left(\frac{1}{2}(v_{max} - v_{min}), -f\right) \Rightarrow y=-z$$

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

$$k_y = \frac{2f}{(v_{max} - v_{min})}$$ ($k_y$ is y scale factor)

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

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)

$$\mathbf{Sc}_2 = \begin{vmatrix} \frac{1}{f - B} & 0 & 0 & 0 \\ 0 & \frac{1}{f - B} & 0 & 0 \\ 0 & 0 & \frac{1}{f - B} & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

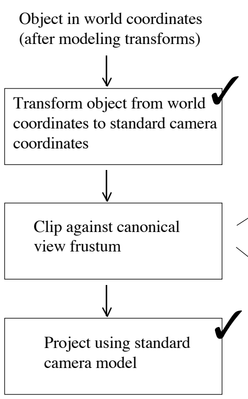
Note that the focal length, f, also gets transformed (needed for the perspective transformation coming up).

It is: $$f' = \frac{f}{f - B}$$

## 3D Viewing Pipeline

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

←




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:

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

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

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”)