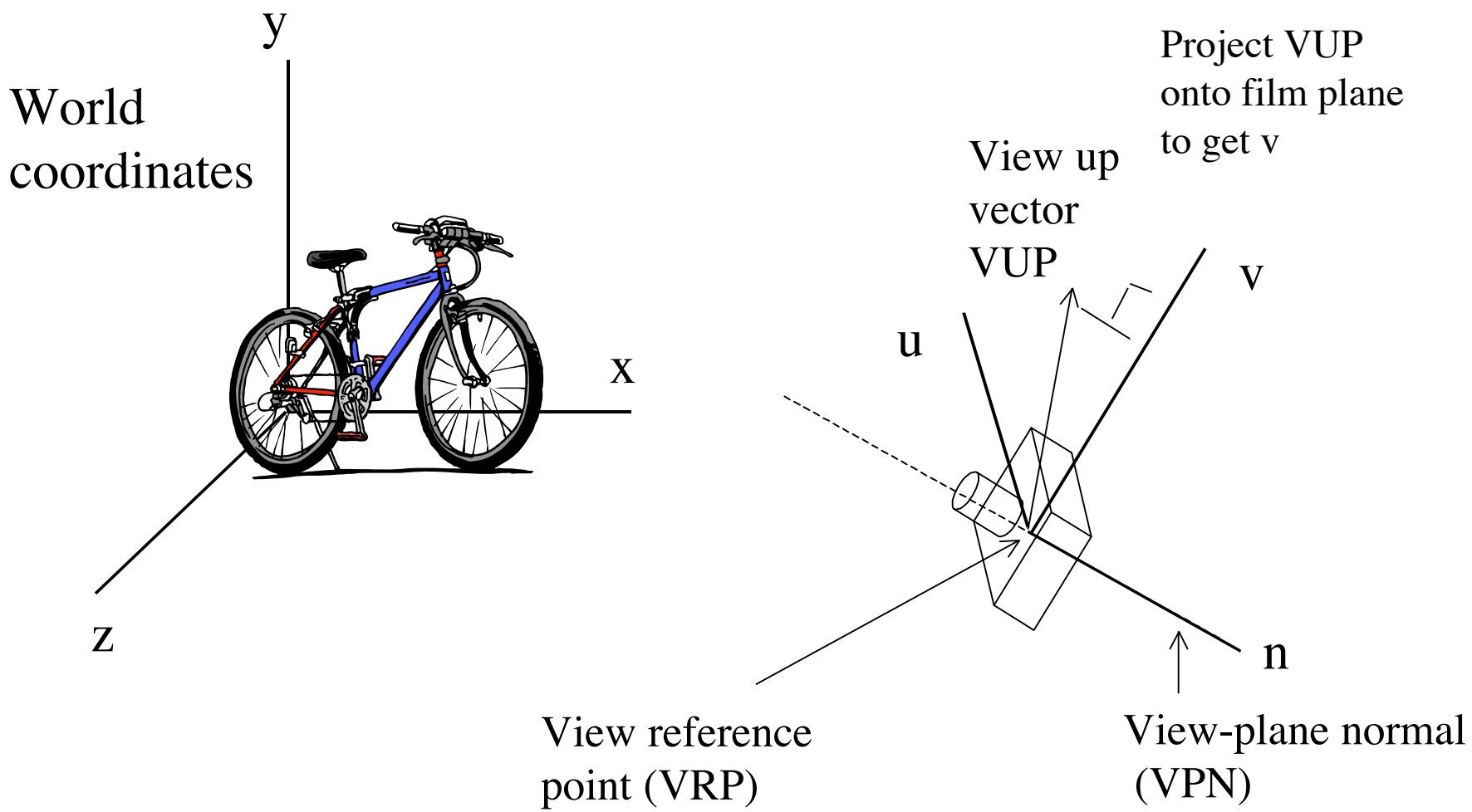
Admin

- Please make your assignments work on graphics machines.
- Kevin can help you with makefiles, etc. if you are new to unix
- Proposed midterm data -- October 28 ?
- Projects --- quick meeting at end of class
- New course next term “computer vision”

Specifying a camera



Specifying a camera

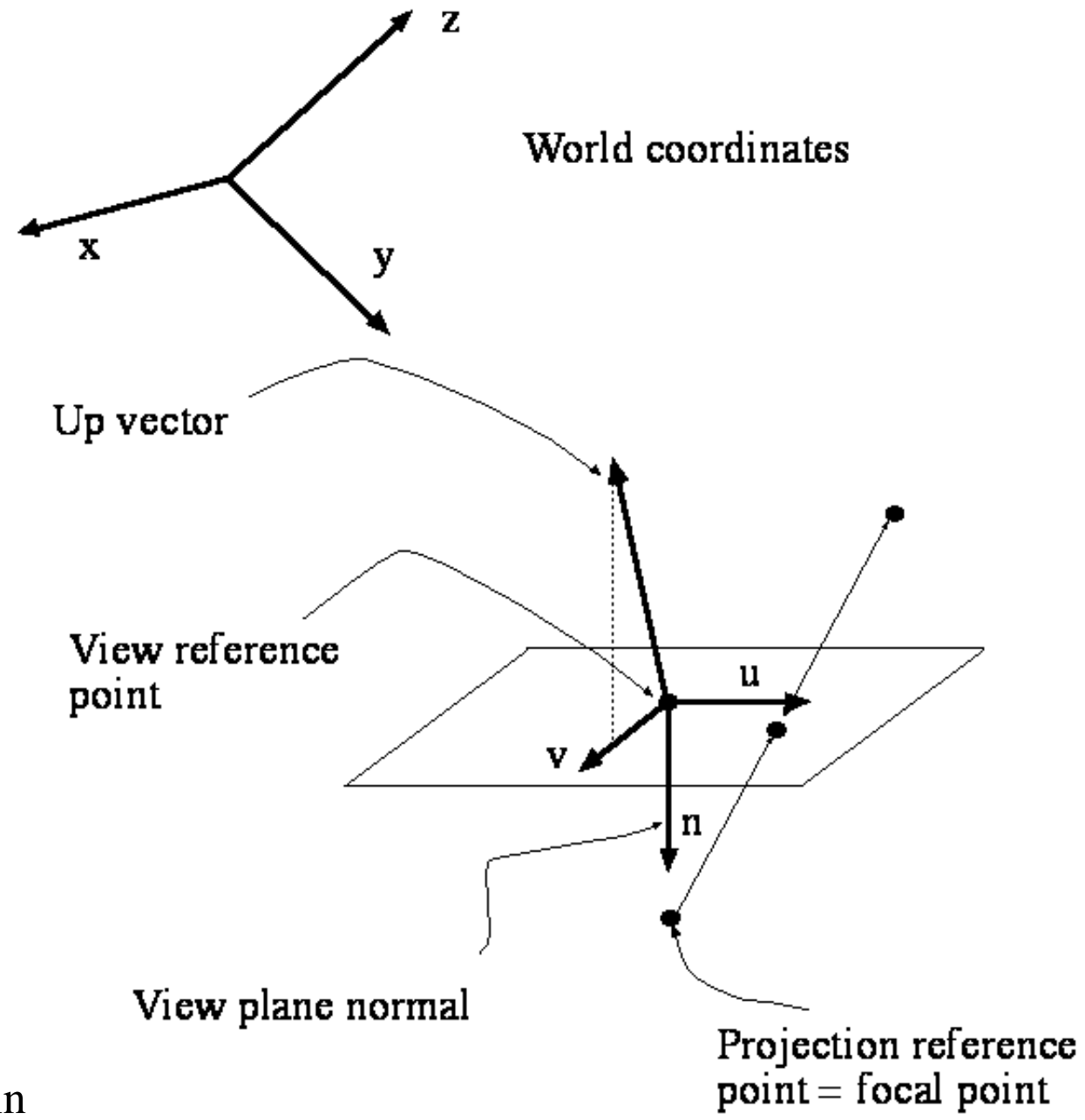
- Why use VUP?
  - Convenient for the user but there are other ways (OpenGL has several ways to negotiate camera parameters, including very much how we are doing it).
  - A world centric coordinate system is natural for the user.
  - Watt develops the material very much like we are doing, but note that the camera direction is opposite to ours. Slightly more natural to but this makes the camera coordinates left handed. (You will see it both ways).

View reference point and view plane normal specify film plane.

Up vector gives an "up" direction in the film plane. vector **v** is projection of up vector into film plane (= **n** ⌊VUP ⌋ **n**).

u is chosen so that (**u**, **v**, **n**) is a right handed coordinate system; i.e. it is possible to rotate (**x->u**, **y->v**, **z->n**) (and we'll do this shortly).

VRP, VPN, VUP must be in world coords; PRP could be in world coords or in camera coords



$$\mathbf{v} \parallel \mathbf{n} \rceil \mathrm{VUP} \rceil \mathbf{n}$$

(We write ∥ for parallel to---we still need to make the RHS into a unit vector to get **v**.)

Why does this work?

Want **v** to be in plane of VUP and **n** and perpendicular to **n**

n | | VUP □ **n** is perpendicular to (VUP x **n**) and **n**

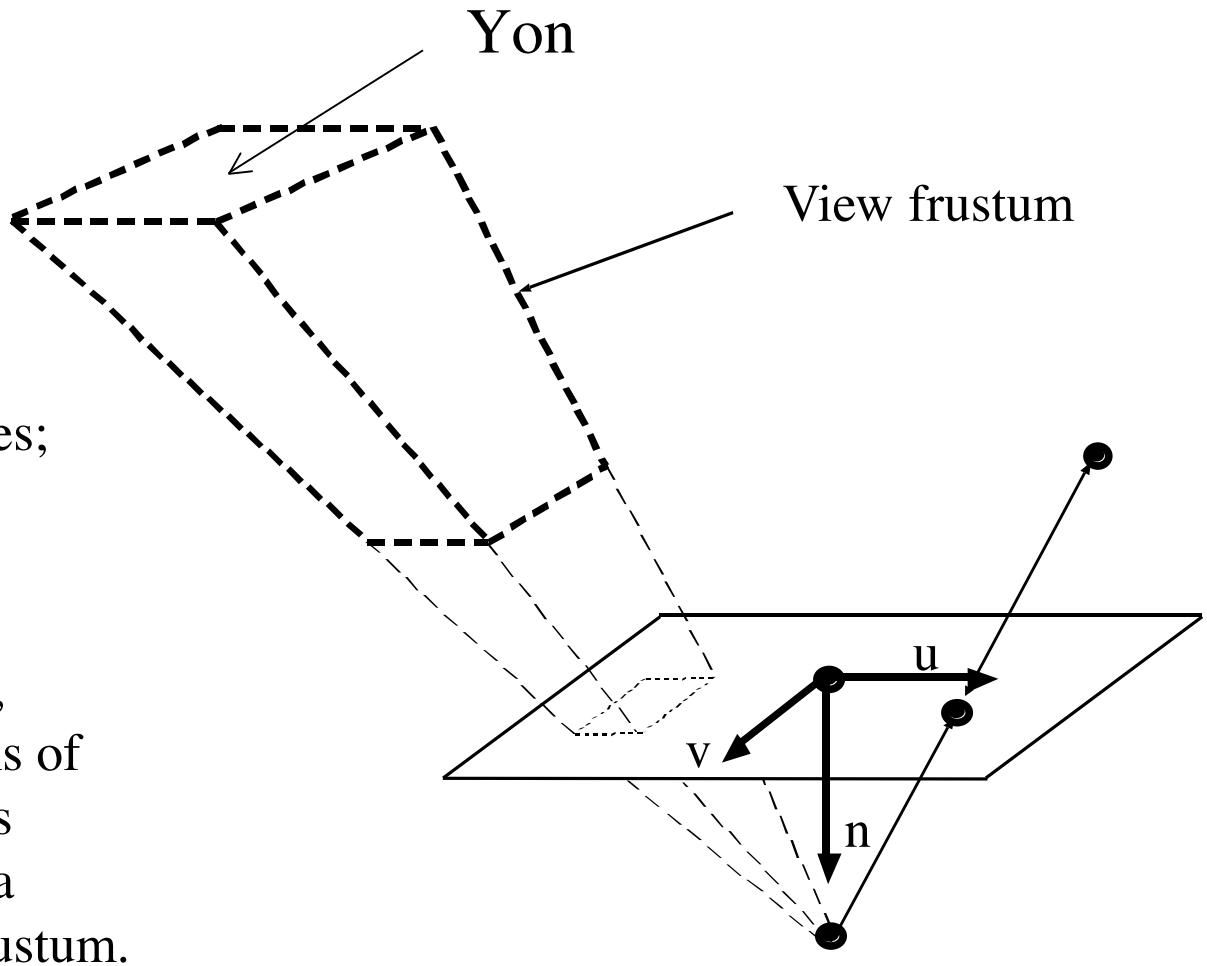
(VUP x **n**) gives a direction perpendicular to both VUP and **n**. So if you are perpendicular to that, then you must be back in the plane.

U, V can be used to specify a window in the film plane; only this section of film ends up on the screen.

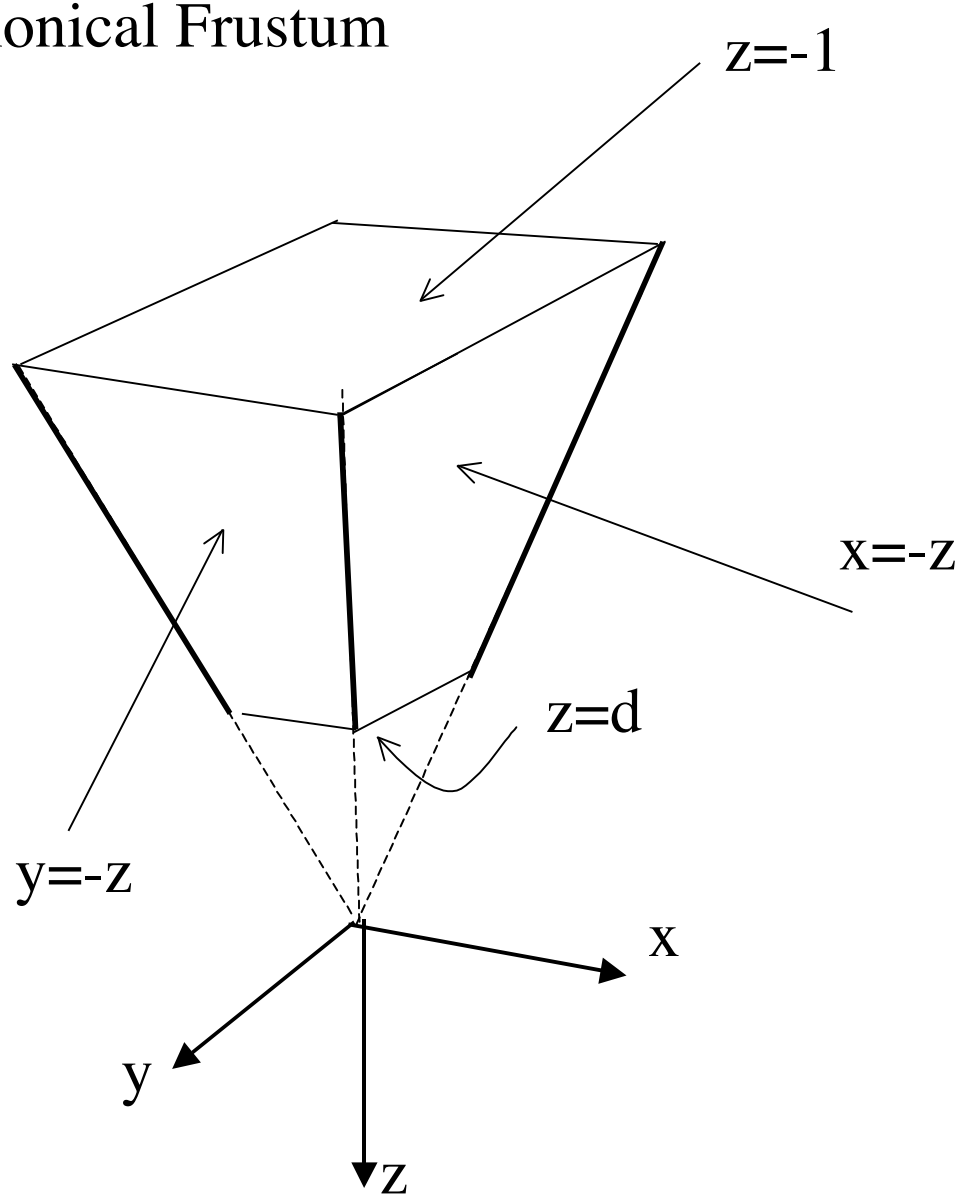
This window defines four planes; points outside these planes are not rendered.

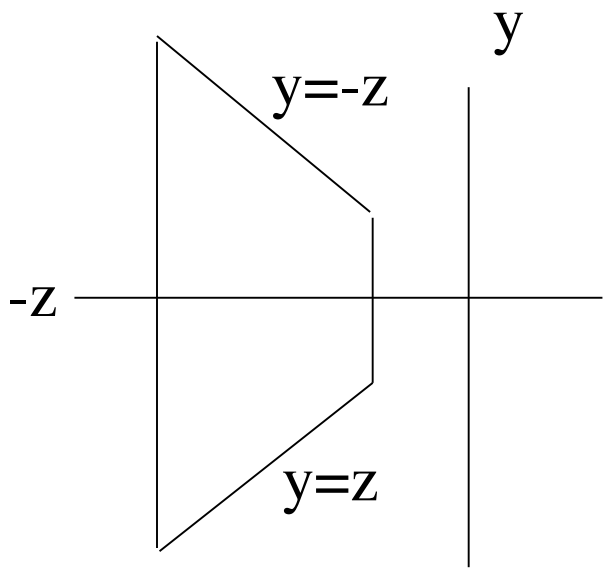
Hither and yon clipping planes, which are always given in terms of camera coordinates, and always parallel to the film plane, give a volume - known as the view frustum.

Orthographic case: - view frustum is cuboid (i.e. all angles right angles, but edges not necessarily of equal length).



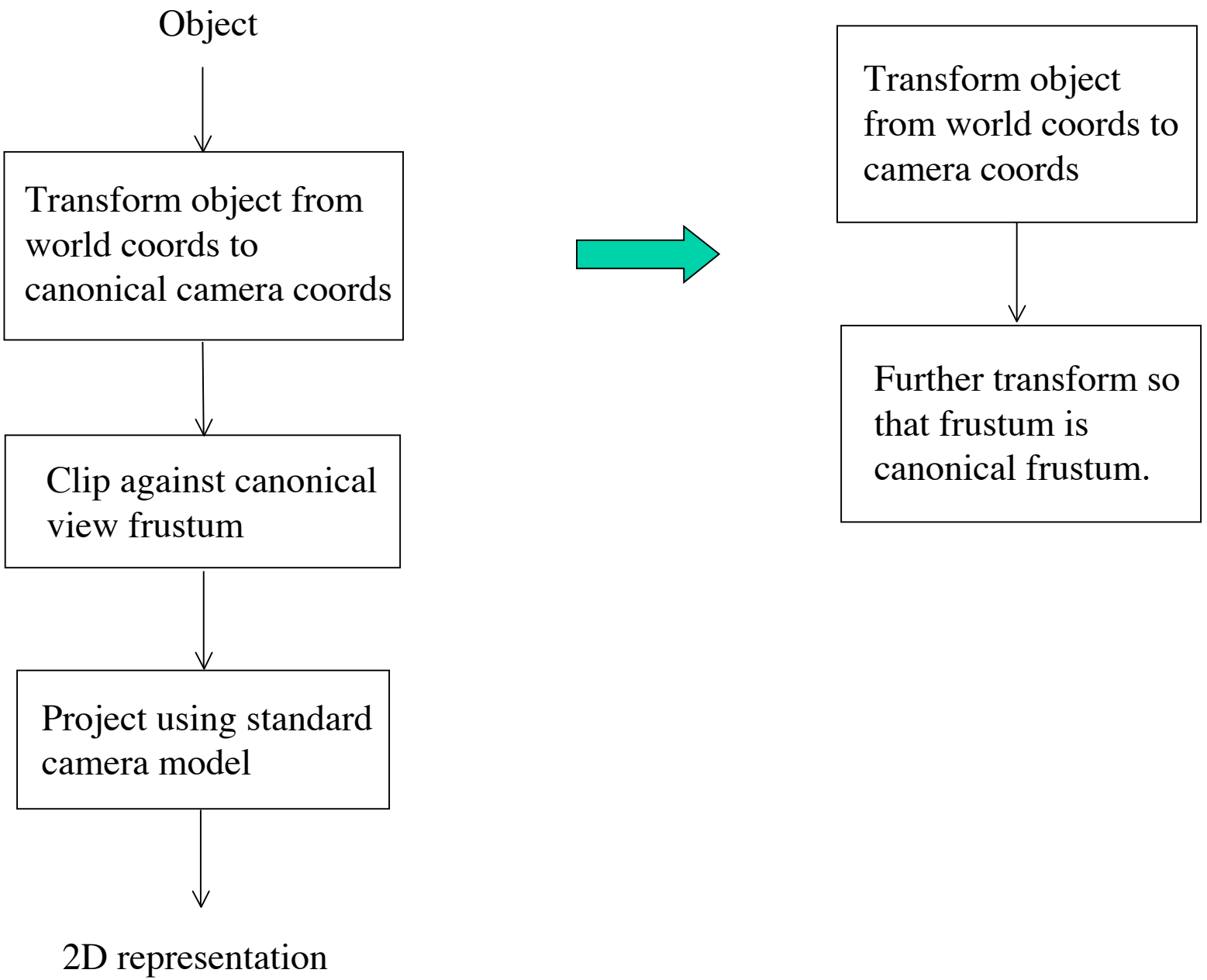
Canonical Frustum





If image plane transforms to z=m then in new frame, projection is easy:

(x, y, z) -> (m x /z, m y/z)


Object
Transform object from world coords to canonical camera coords
Clip against canonical view frustum
Project using standard camera model
2D representation
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 VRP to world origin. Call this T_1. T_1 maps world points to camera coordinates.

VRP must end up with coordinates (0,0,0) in the **new** frame, so the translation must be like subtracting VRP coords from every point.

(Note opposite transformations for object and coordinate frame).

Transform object from world coords to camera coords

Step 2. Rotate camera coordinate frame so that u is x, v is y, and n is z. The matrix is ?

Transform object from world coords to camera coords

Step 2. Rotate camera coordinate frame so that u is x, v is y, and n is z. The matrix is:

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

(why?)

Transform object from world coords to camera coords

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

u in current coords (world shifted so that VPR is at origin) 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)

Further transform so that frustum is canonical frustum.

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

Step 1: Translate focal point to origin; call translation T_2. This takes center of window to:

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