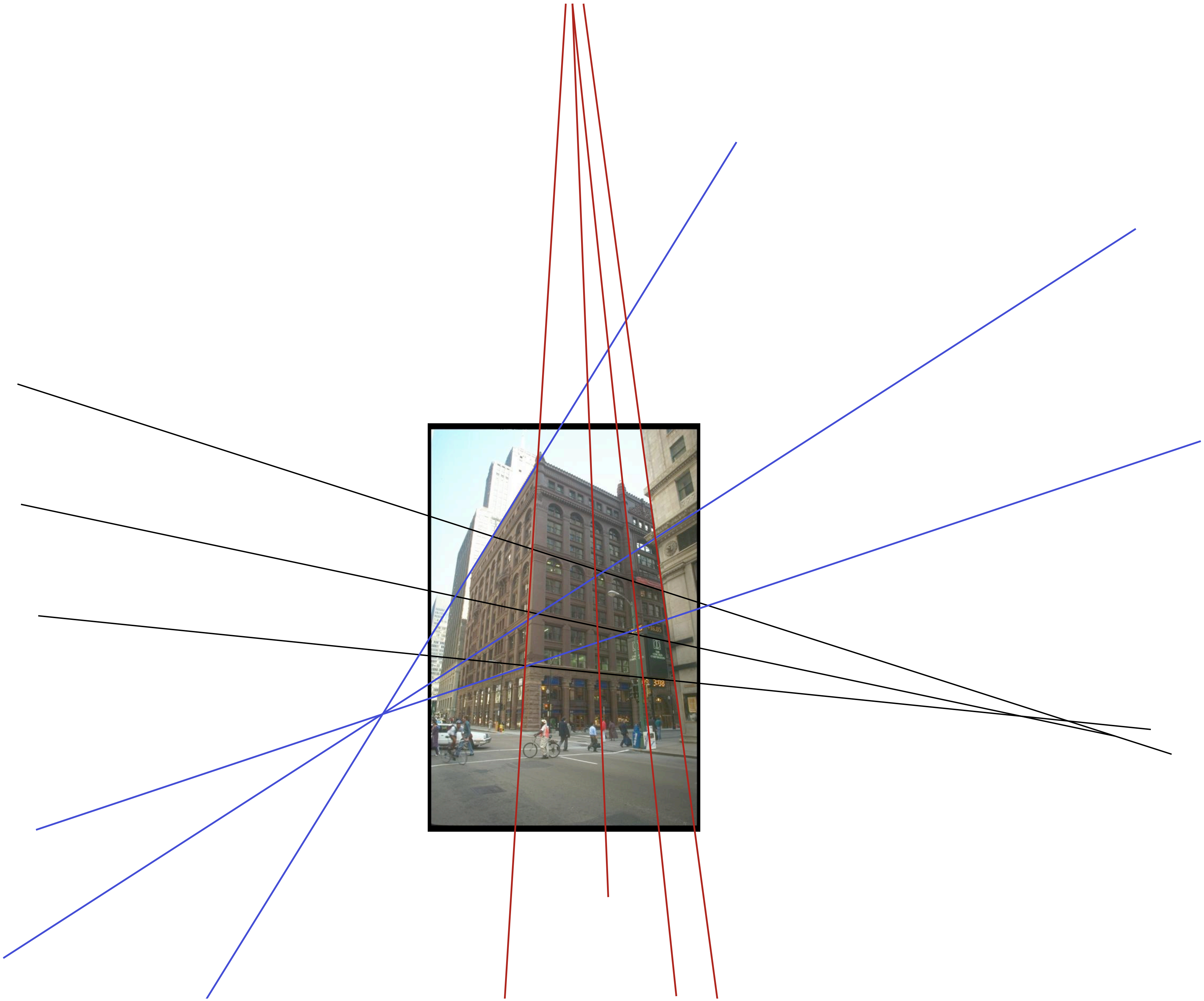
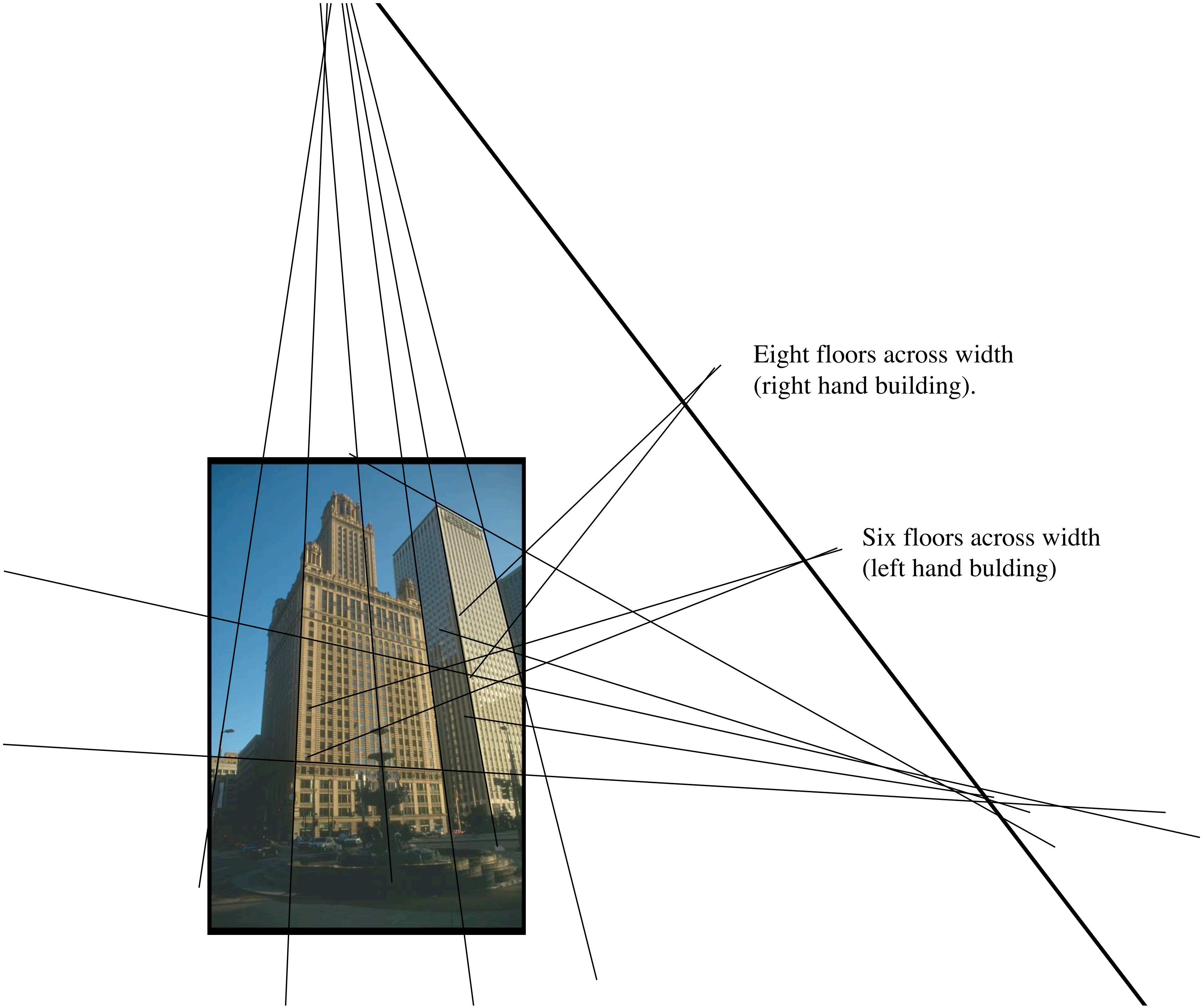




Eight floors across width
(right hand building).
Six floors across width
(left hand bulding)

Object in world coordinates
(after modeling transforms)

↓

Transform object from world coordinates to standard camera coordinates

Next topic

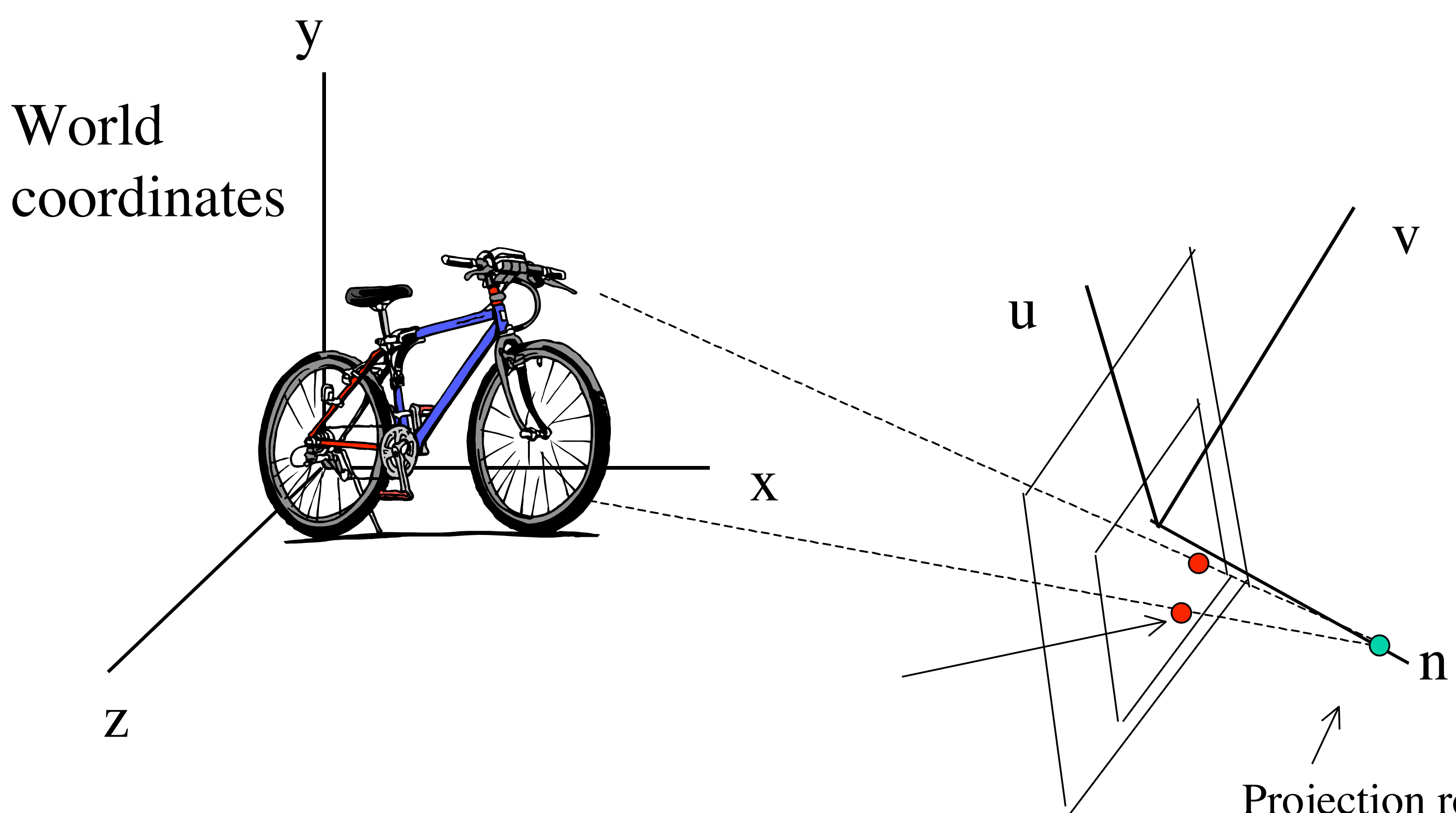
↓

Clip against canonical view frustum

↓

Project using standard camera model ✓

Camera coordinate system



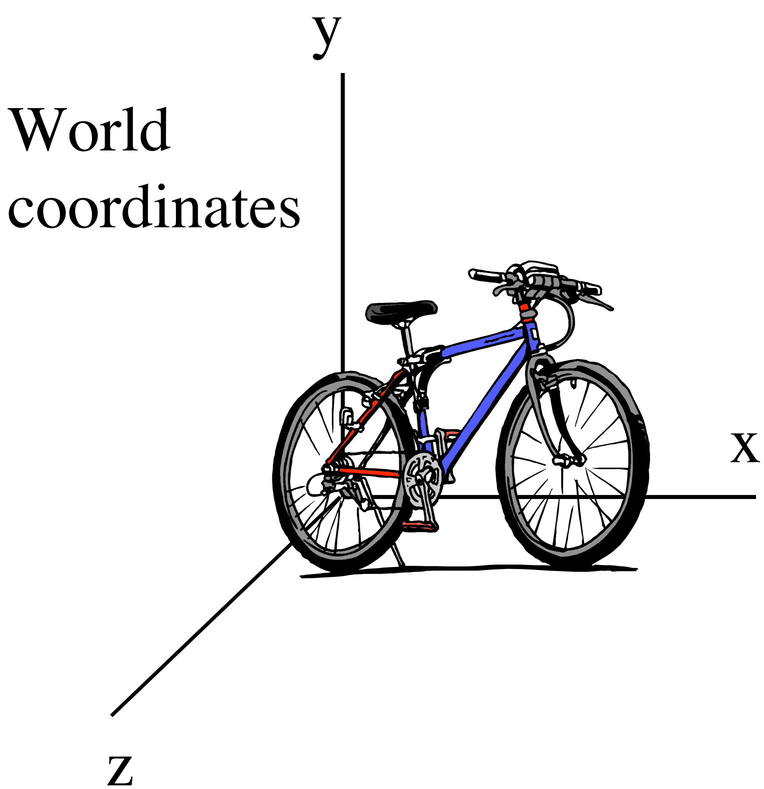
Window given by (u_{min}, u_{max}), (v_{min}, v_{max}), denote center by CW

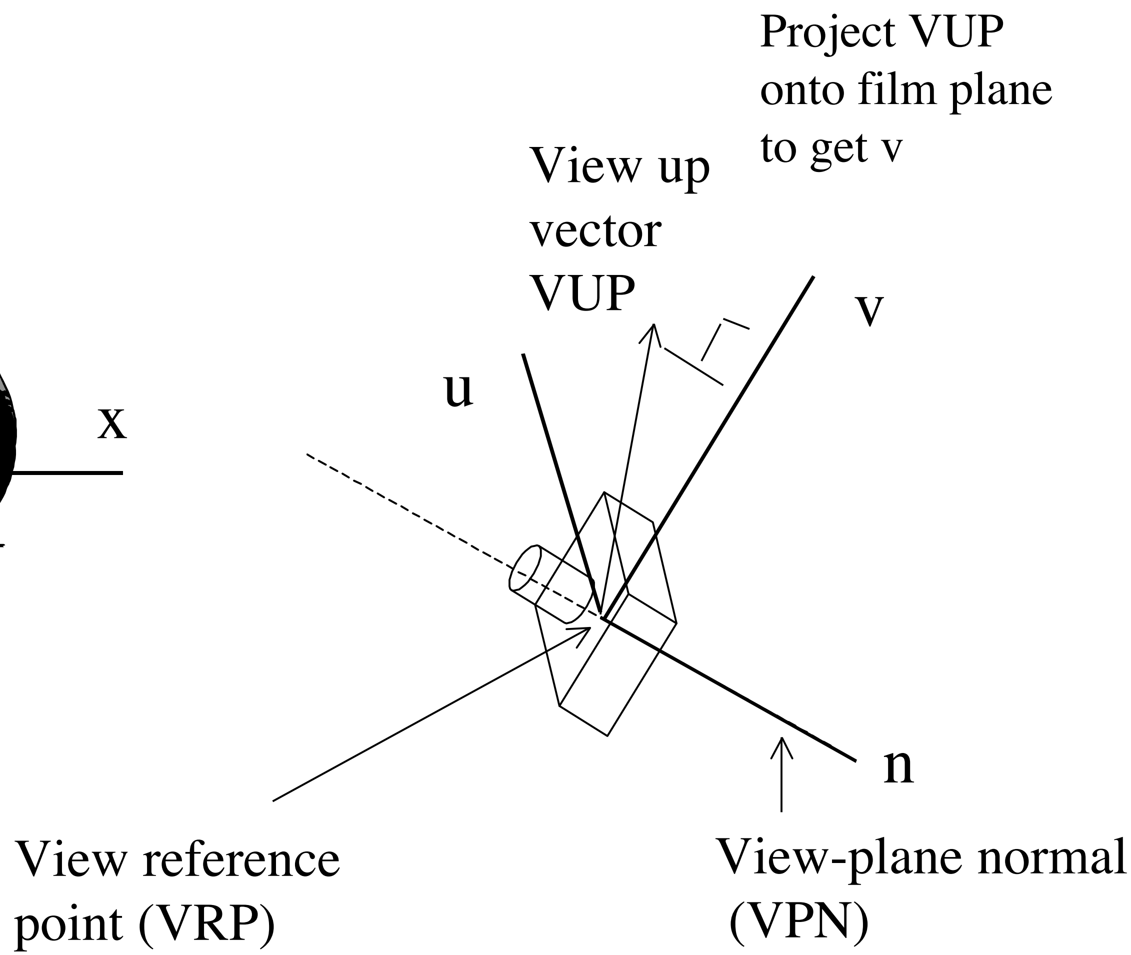
Projection reference point (PRP)

Specifying a camera

- We link camera specifications that relate to the user or the application to the (**u, n, v**) coordinate system
- There are many ways for the user to specify the camera. OpenGL has several. We will study one common and flexible one based on world coordinate entities “VUP” and “VPN”
- Note the difference between camera specification and the camera coordinate system
- Our treatment is very similar to H&S (§7.3), but note that they call VUP, **V**

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 one which is very much how we are doing it).
  - A world centric coordinate system is natural for the user. In particular, the user may think in terms of the camera rotation around the axis (**n**) relative to a natural horizon and/or "up" direction.
  - This will mean that VUP cannot be parallel to **n**. Often one will fix VUP (e.g. to the Y-axis) but this is too restrictive for some applications.
- Why use a "backwards" pointing **n**?
  - It is more natural to make the camera direction point the other way, but this makes the camera coordinates left handed. (You will see it done both ways).

Specifying a camera

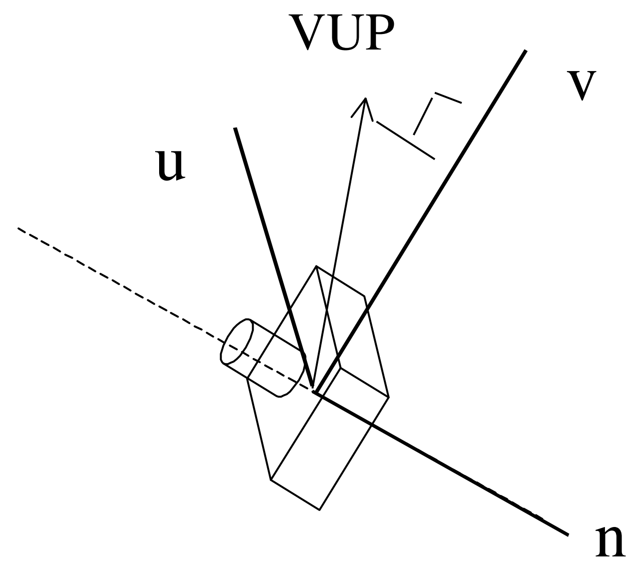
- View reference point, VRP, and view plane normal, VPN=**n**, specify image plane.
- Up vector, VUP, gives an “up” direction in the image plane, providing for user twist of camera about **n**. **v** is projection of up vector into image plane.
- We construct (**u**, **v**, **n**) so that it is a right handed coordinate system
- Thus it is possible to map the world coordinates (**x, y, z**) to (**u**, **v**, **n**) so that (**x**->**u**, **y**->**v**, **z**->**n**) using only translations and rotations.

Computing (u,v,n) in world coordinates

v is the projection of VUP into the view plane which is perpendicular to **n**

u is perpendicular to the plane formed by **n** and VUP (equivalently, **v**)

So, we can easily compute a vector parallel to **u**.

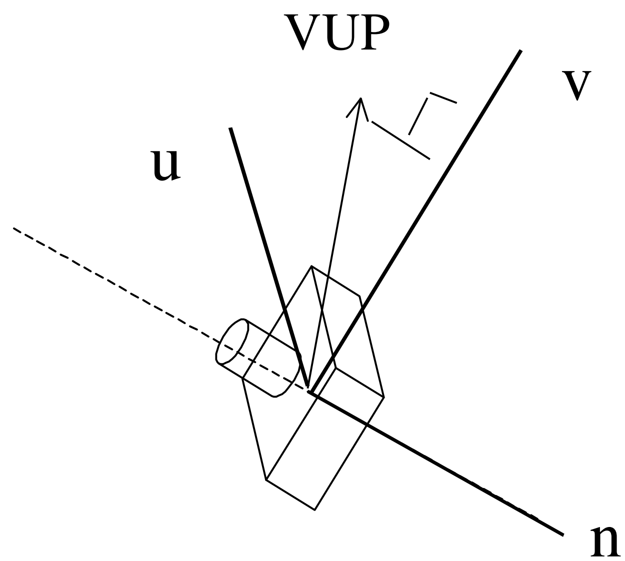


Computing (u,v,n) in world coordinates

$$\mathbf{u} \parallel \mathrm{VUP} \times \mathbf{n}$$

$$\mathbf{u} = \frac{\mathrm{VUP} \times \mathbf{n}}{\left|\mathrm{VUP} \times \mathbf{n}\right|} = \frac{\mathrm{VUP} \times \mathbf{n}}{\left|\mathrm{VUP}\right|}$$

What about **v**?

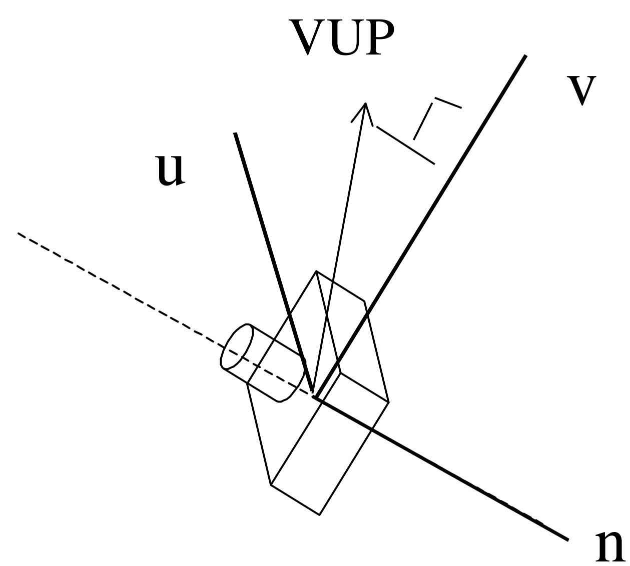


Computing (u,v,n) in world coordinates

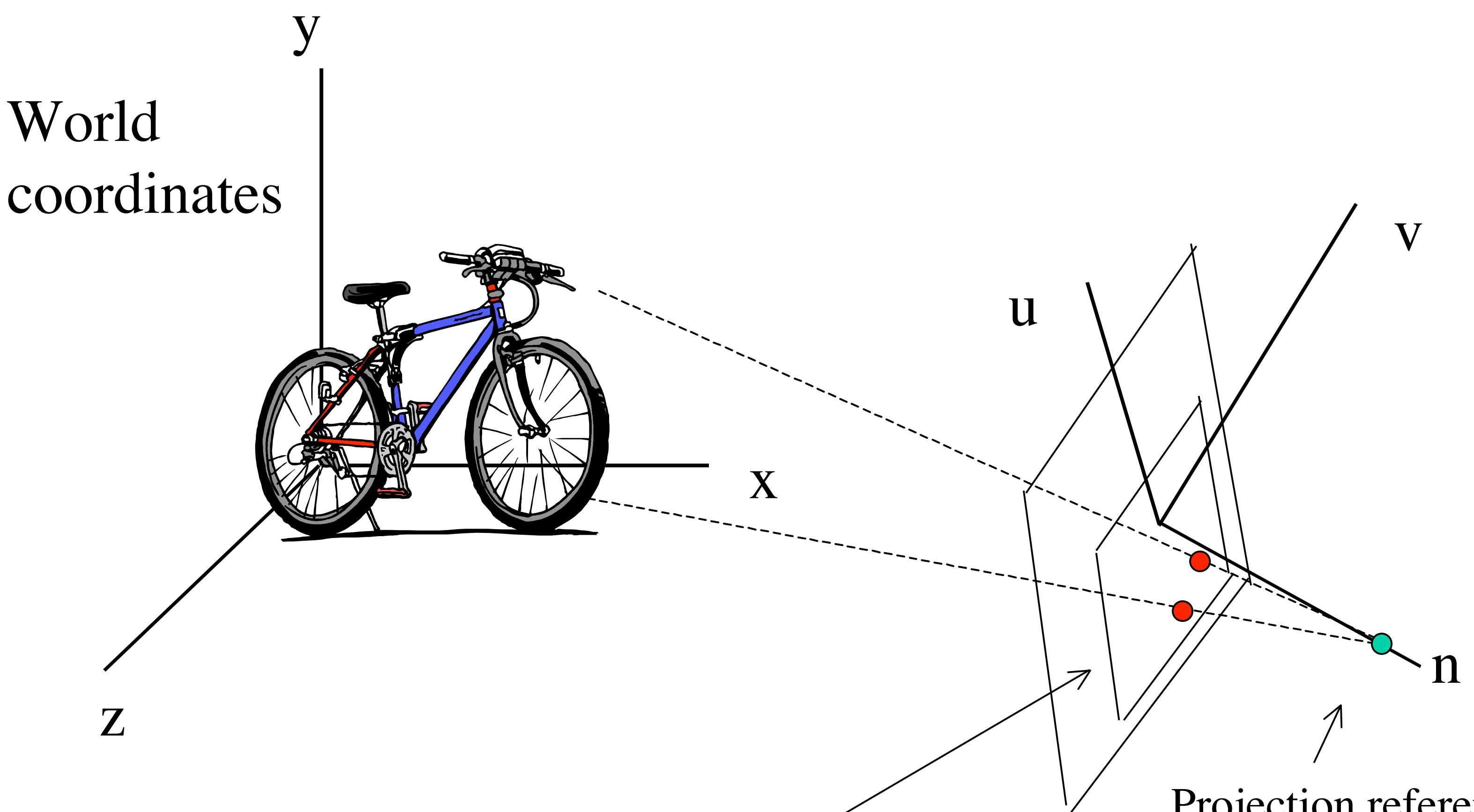
$$\mathbf{u} \parallel \mathrm{VUP} \times \mathbf{n}$$

$$\mathbf{u} = \frac{\mathrm{VUP} \times \mathbf{n}}{\left|\mathrm{VUP} \times \mathbf{n}\right|} = \frac{\mathrm{VUP} \times \mathbf{n}}{\left|\mathrm{VUP}\right|}$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u}$$



Specifying a camera



Window given by (u_{min}, u_{max}), (v_{min}, v_{max}), denote center by CW

Projection reference point (PRP)--assume on **n**

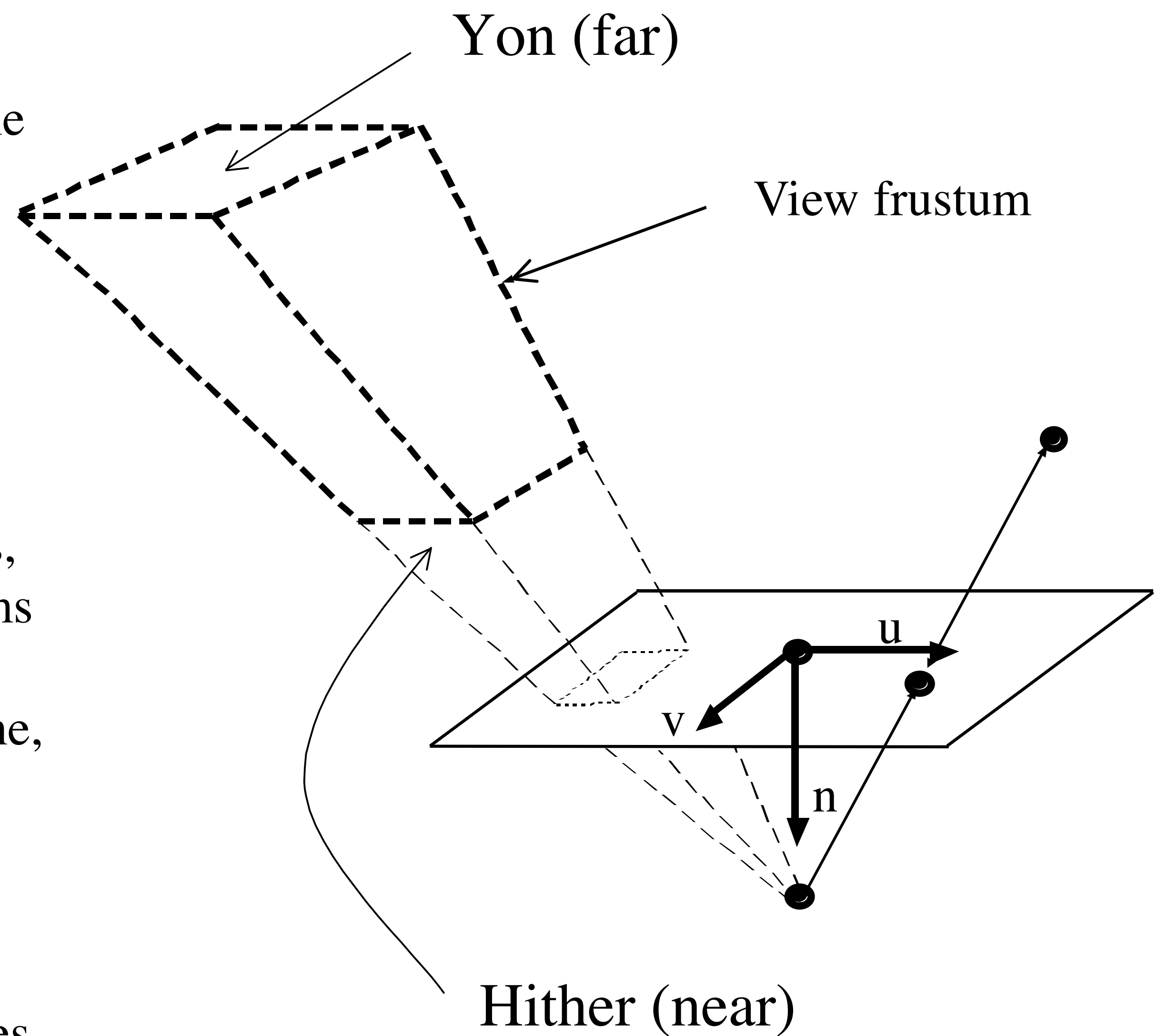
- VRP, VPN, VUP must be in world coords;
- PRP (focal point) could be in world coords, but more commonly, camera coords
- We will use camera coords, and further assume that it is simply (0,0,f).
- What follows works fine for an off-axis PRP, but this is rarely needed.

u and **v** can be used to specify a window in the image plane; only this section of image plane 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).



We will first map world coordinates to the camera coordinates (top figure)

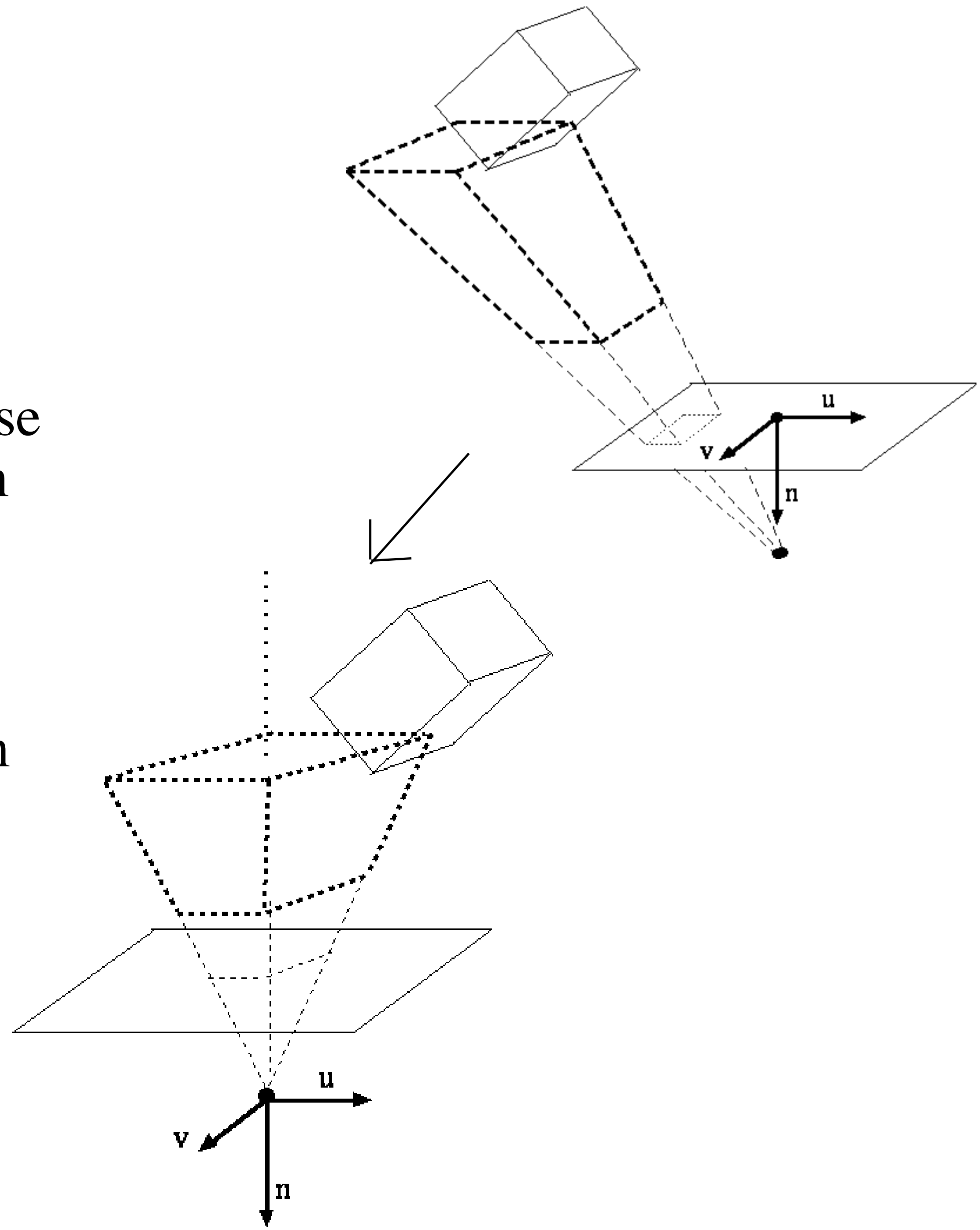
However, if we clipping against the frustum is because planes bounding the frustum have a complex form

Solution: further transform frustrum to a canonical form where clip planes are easy form to deal with. Specifically:

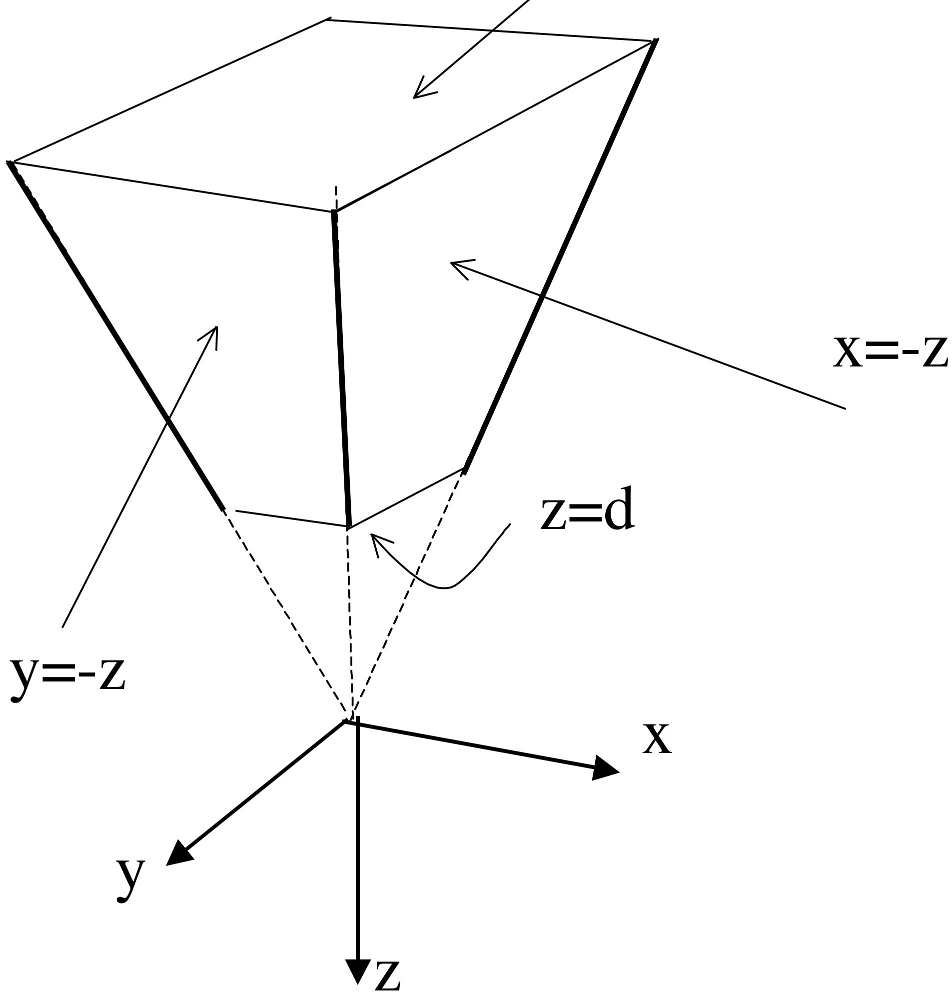
z=x, z=-x

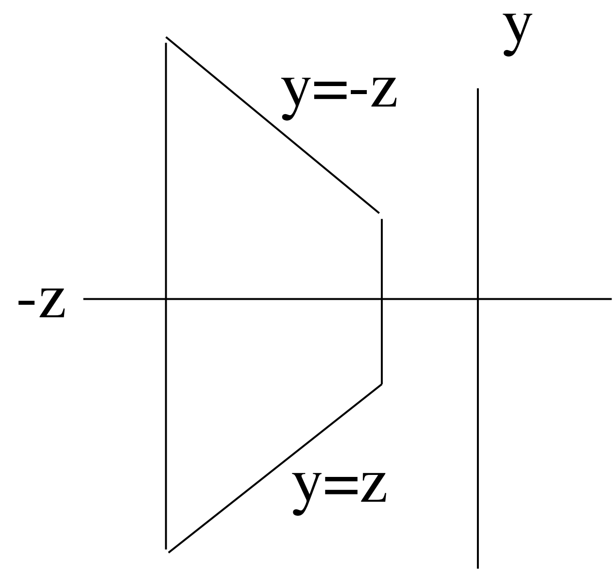
z=y, z=-y

z=-1, z=d



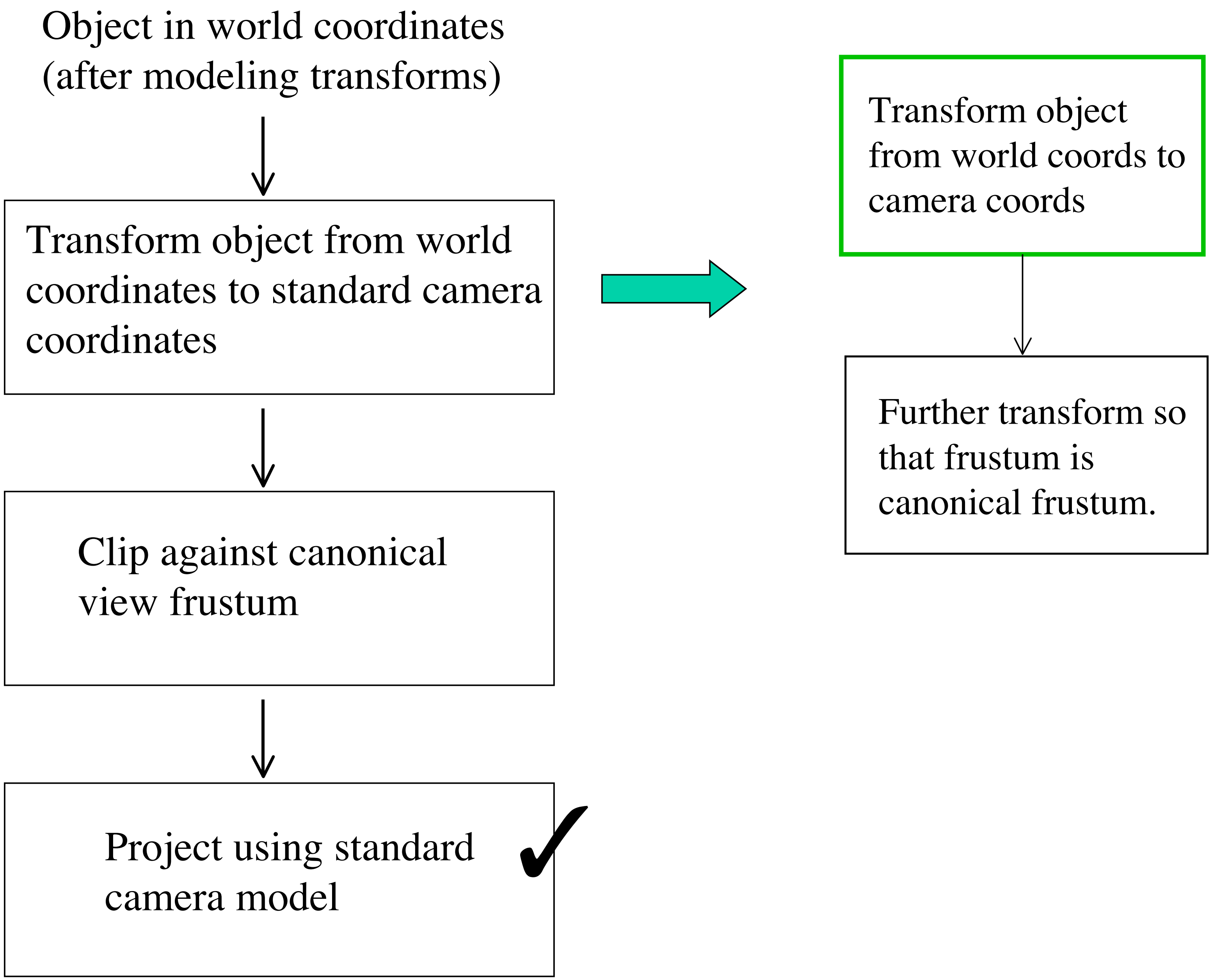
Canonical Frustum





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

$(x, y, z) \rightarrow (m\,x/z, m\,y/z)$


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)