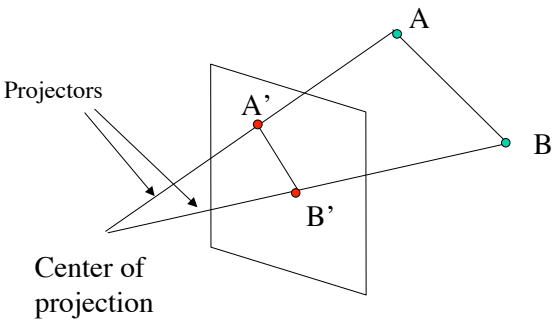
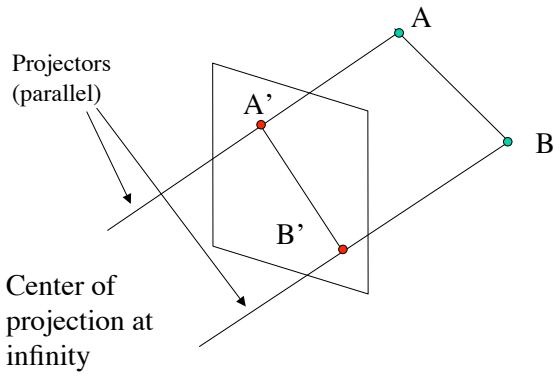
Projections

- Mathematical definition of a projection: PP=P
- (Doing it a second time has no effect).
- Generally rank deficient (non-invertable)--exception is P=I
- Transformation looses information (e.g., depth)
- Given a 2D image, there are many 3D worlds that could have lead to it.

Projections



Parallel Projection



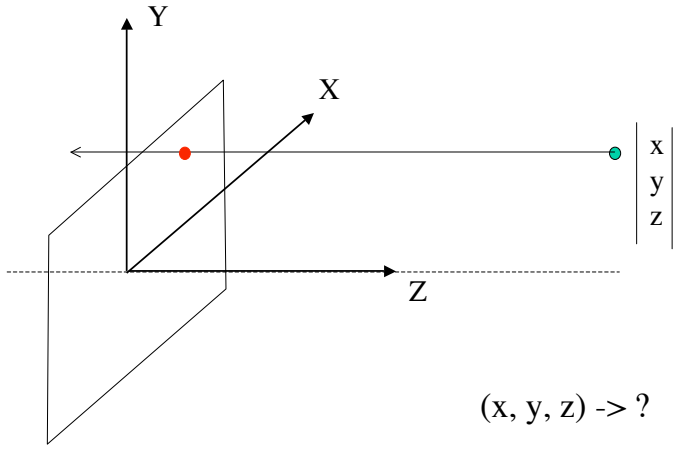
Parallel Projection

Parallel lines remain parallel, some 3D measurements can be made using 2D picture

Does not give realistic 3D view because eye is more like perspective projection.

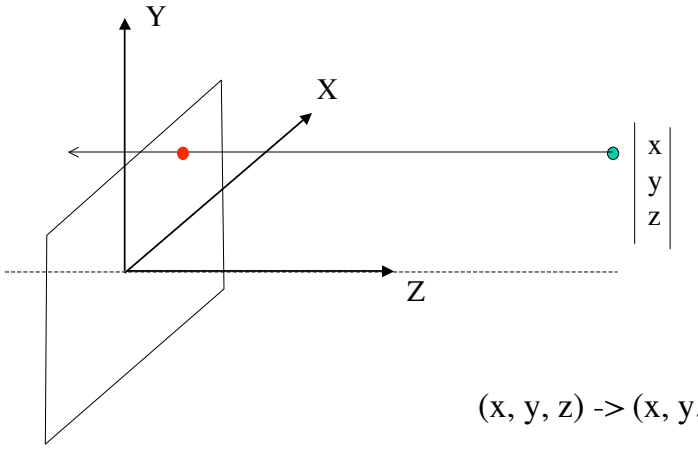
If projection plane is perpendicular to projectors the projection is *orthographic*

Orthographic example (onto z=0)



(x, y, z) -> ?

Orthographic example (onto z=0)



(x, y, z) -> (x, y, 0)

Can we do this with a linear transformation?

$$\begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{bmatrix} & ? & \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

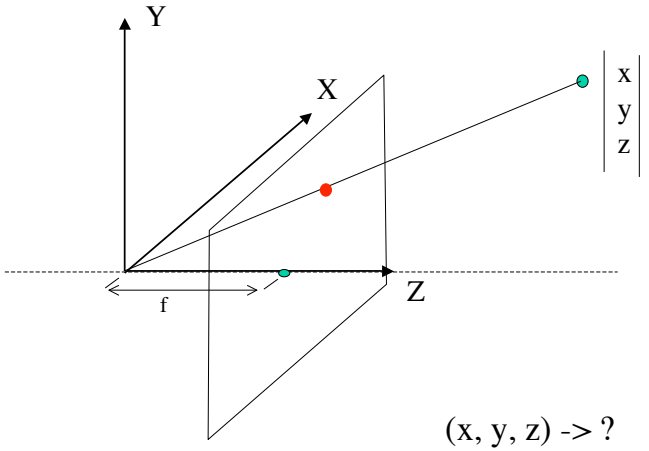
Can we do this with a linear transformation?

$$\begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Camera matrix in homogeneous coordinates

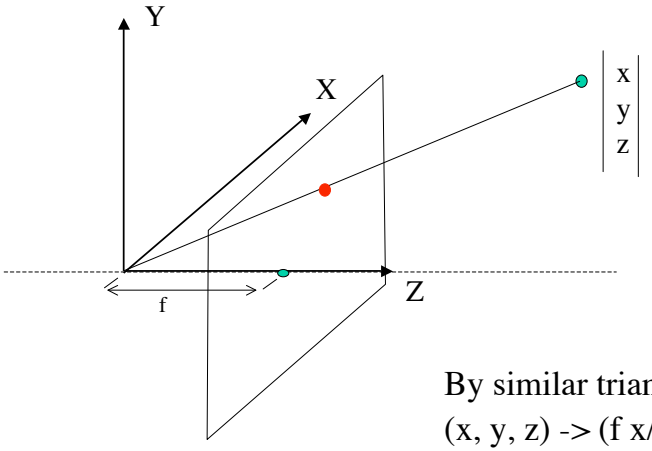
$$\begin{pmatrix} x \\ y \\ 0 \\ w \end{pmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

Perspective example (onto z=f)



(x, y, z) -> ?

Perspective example (onto z=f)



By similar triangles,
(x, y, z) -> (f x/z, f y/z, f)

Can we do this with a linear transformation?

$$f\begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} = \begin{bmatrix} ? & & \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Now try homogeneous coordinates

- In homogeneous coordinates

$$(x,\ y,\ z,\ 1) \Rightarrow (f\frac{x}{z},\ f\frac{y}{z},\ f,\ 1)$$

- Equivalently

$$(x,\ y,\ z,\ 1) \Rightarrow (x,\ y,\ z,\ \frac{z}{f})$$

- (Now H.C. are being used to store foreshortening)

Is there a linear transformation?

$$\begin{pmatrix} x \\ y \\ z \\ \frac{z}{f} \end{pmatrix} = \begin{bmatrix} & ? & \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

The camera matrix

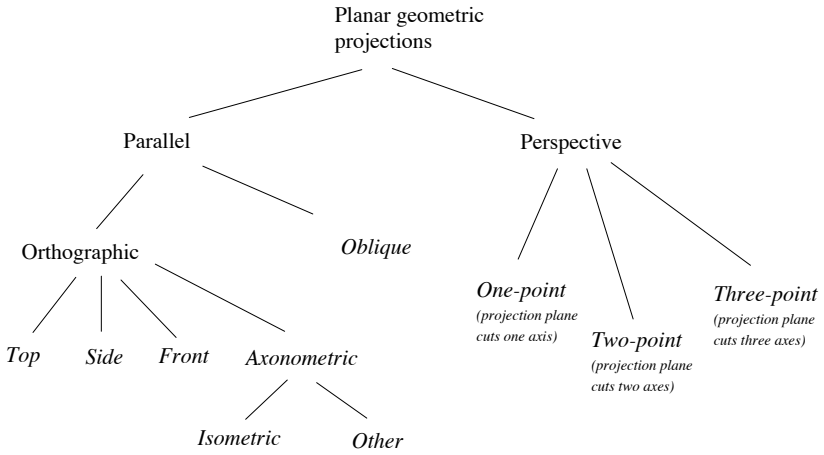
$$\begin{pmatrix} x \\ y \\ z \\ \frac{z}{f} \end{pmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & \frac{1}{f} & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Summary so far

- Parallel and perspective are projections (non-invertable)
- Perspective viewing is not a linear transform in (x,y,z)
- Perspective viewing can be encoded as a linear tranformation in homogenous coordinates
- Perspective becomes parallel projection as *f* becomes infinite

Projection Taxonomy

Terminology in italics is optional

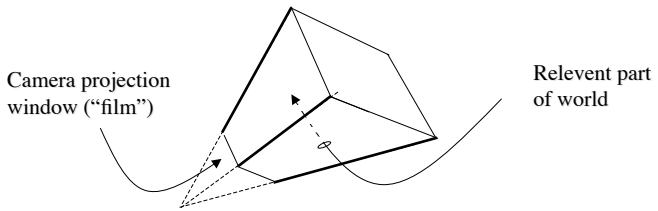


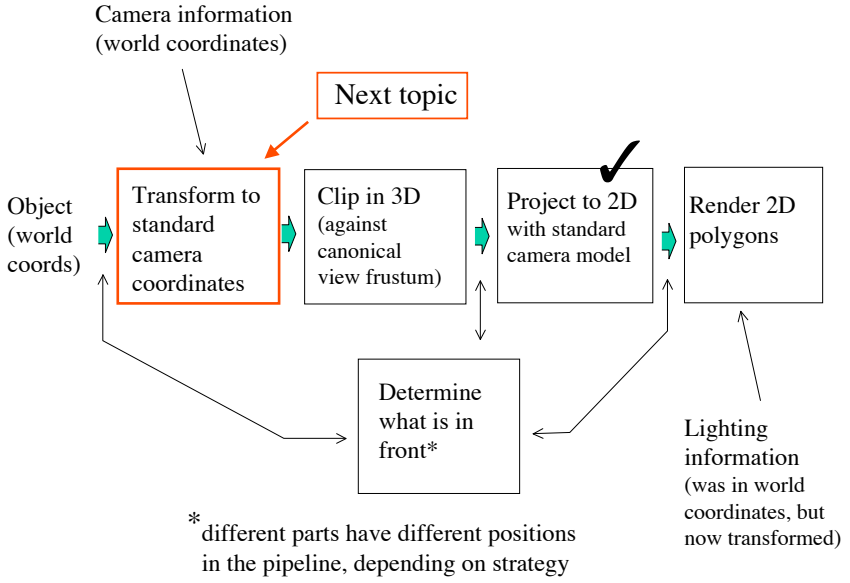
Specifying a camera

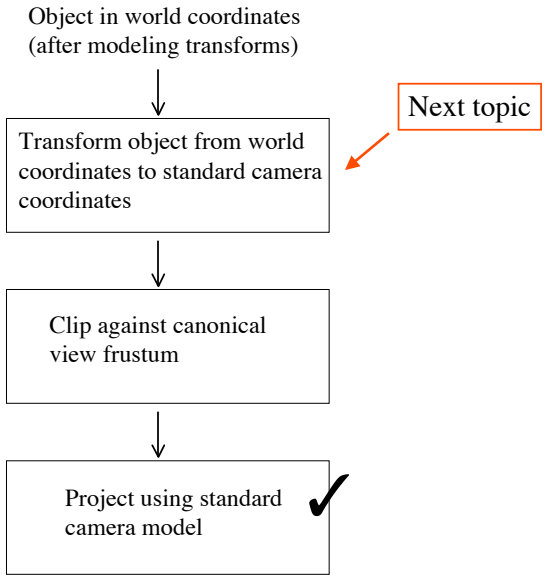
- Makes sense to tell rendering system where camera is in world coordinates
- We want to transform the world into camera coordinates so that projection is easy
  - In particular, we want to use the projection matrix from a few slides back.
- Need to specify focal point and film plane.
- Convenient to construct a coordinate system for the camera with origin on film plane

Clipping volume

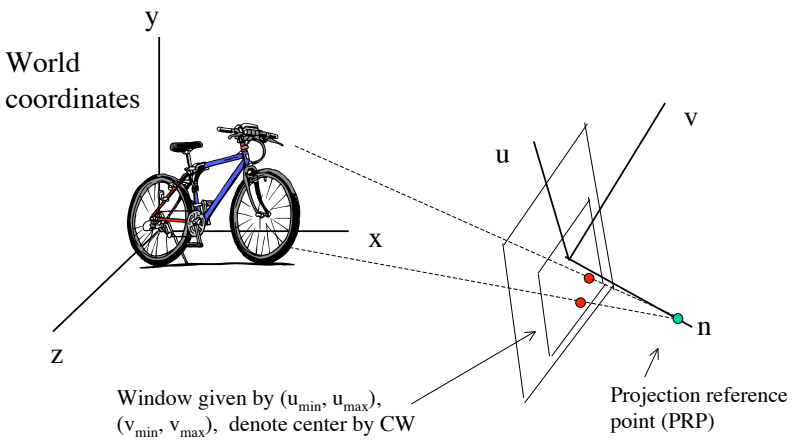
- We render only a window in the film plane
- Things beyond any of four sides don't get rendered
- Things that are too far away don't get rendered
- Things that are too near don't get rendered
- Taken together, this means that everying outside a truncated pyramid (frustum) is irrelavent.







Camera coordinate system



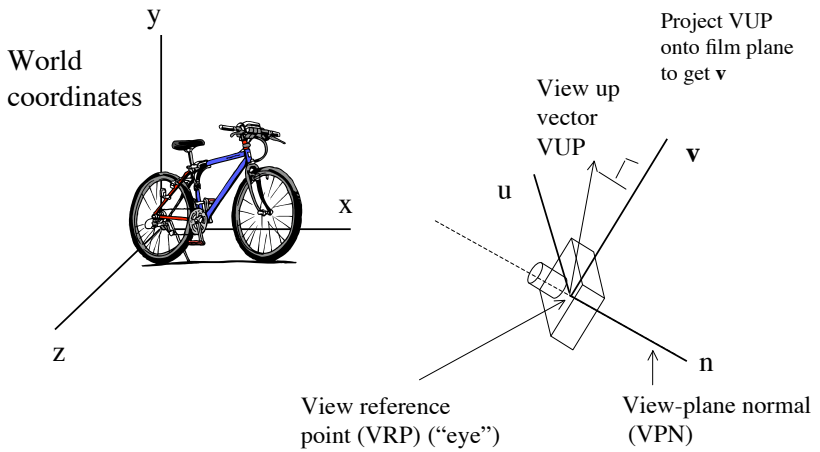
Specifying a camera

- We link camera specifications that relate to the user or the application to the (**u, v, n**) coordinate system
- The picture on the previous page is the most general case. Often one assumes that **n** goes through the center of the window (CW).

Specifying a camera

- 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"
- 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

- Together, the view reference point, VRP, and view plane normal, VPN=**n**, specify image plane.
- The up vector, VUP, gives an "up" direction in the image plane, providing for a user twist of camera about **n**.
- The camera coordinate system axis, **v,** is the projection of the up vector, VUP, into image plane.

Specifying a camera

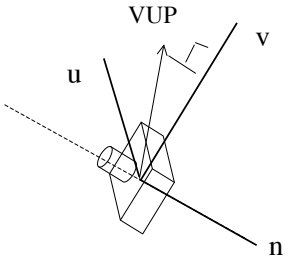
- We construct (**u**, **v**, **n**) so that it is a right handed coordinate system.
- This means that it is possible to map the world coordinates (**x, y, z**) to (**u**, **v**, **n**) so that (**x**->**u**, **y**->**v**, **z**->**n**) using only translations and rotations.

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

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

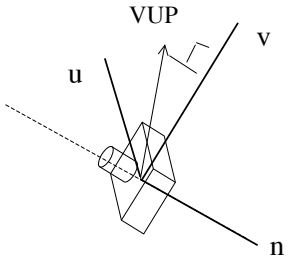


Computing (u,v,n) in world coordinates

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

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

What about **v**?

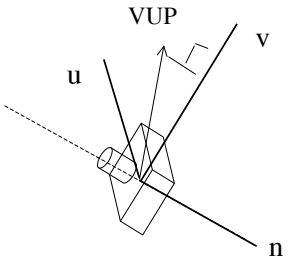


Computing (u,v,n) in world coordinates

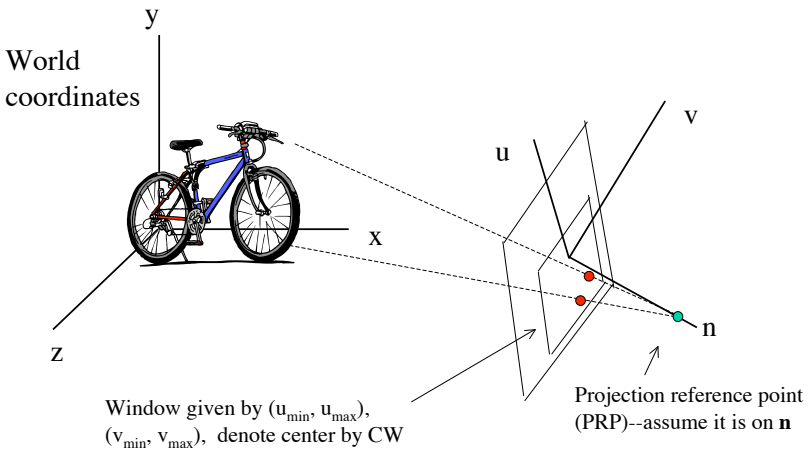
$$\mathbf{u} \parallel \text{VUP} \times \mathbf{n}$$

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

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



Specifying a camera



- VRP, VPN, VUP must be in world coords;
- PRP (focal point) could be in world coords, but more commonly, camera coords (which are the same scale as world coords)
- We will use camera coords, and further assume that PRP = (0,0,f).