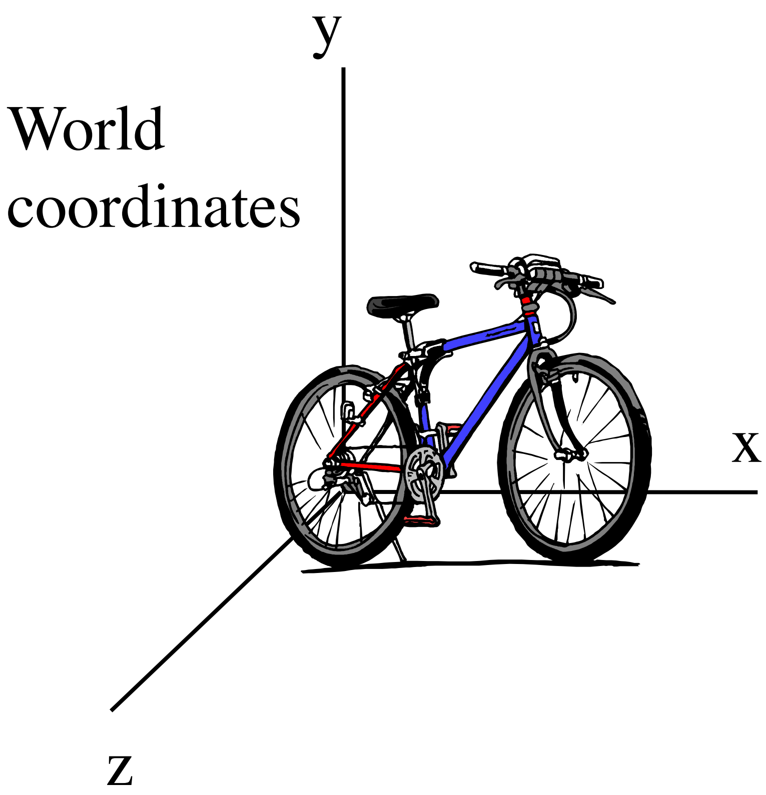
Review from previous lecture.

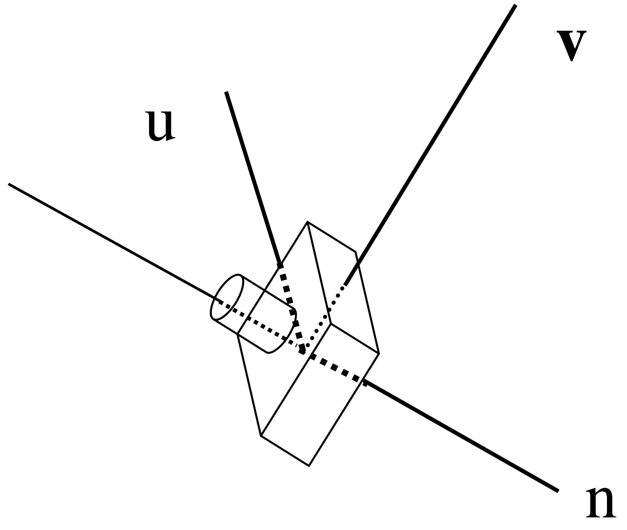
Geometric Camera Model

- Let **P**=(X,Y,Z) be a point in space.
- Let (u,v) be image coordinates.
- A geometric camera model, G, tells us where P goes in the image.
- (u,v) = G(**P**)

Review from previous lecture.

World and camera coordinates





Review from previous lecture.

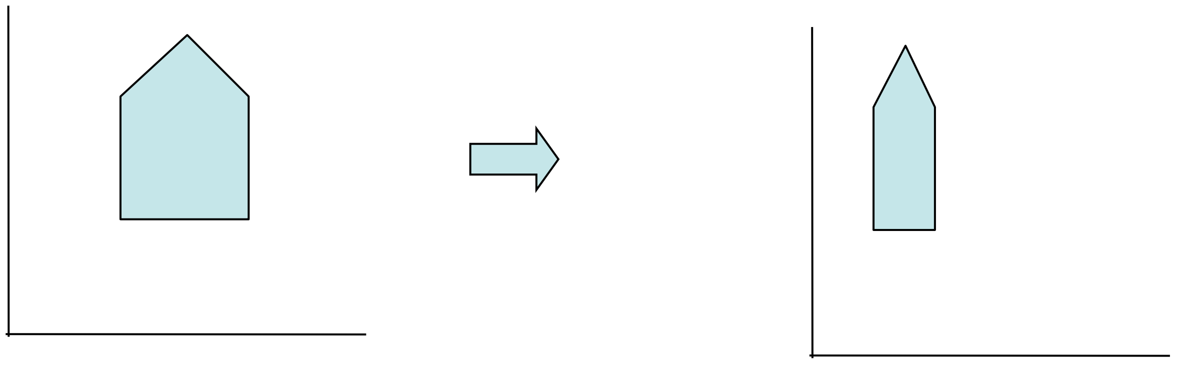
Geometric Camera Model

- Transform world coordinates to standard camera coordinates
  - (Extrinsic parameters)
- Project onto standard camera plane
  - (3D becomes 2D)
- Transform into pixel locations
  - (Intrinsic camera parameters)

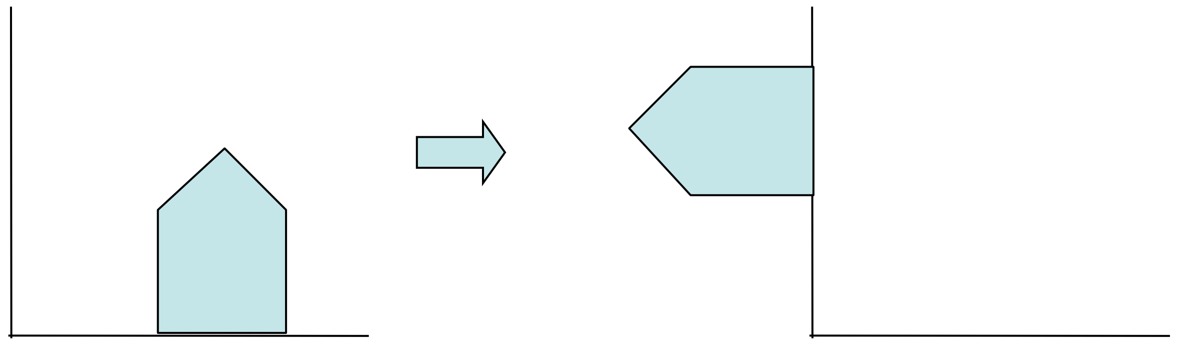
Review from previous lecture.

Representing Transformations

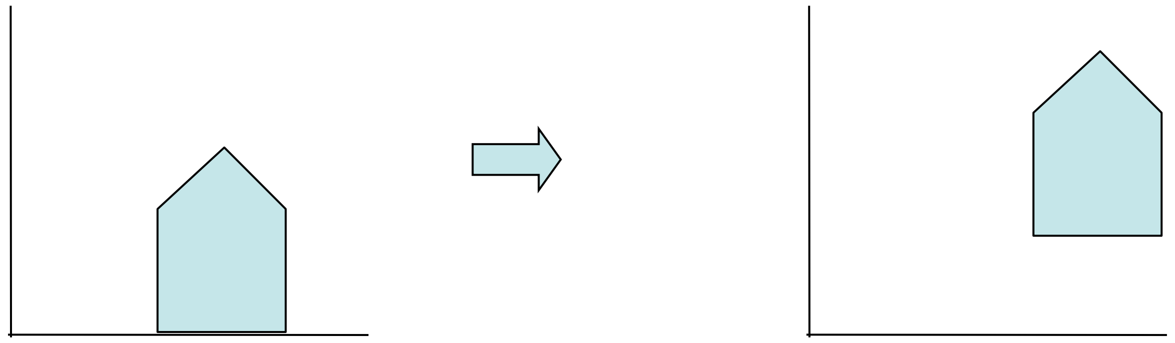
- Need mathematical representation for mapping points from the world to an image (and later, from an image taken by one camera to another).
- Represent linear transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
- To transform line segments, transform endpoints
- To transform polygons, transform vertices



$$M = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$



$$M = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$



M =?

Homogenous Coordinates

- Represent 2D points by 3D vectors
- (x,y)-->(x,y,1)
- Now a multitude of 3D points (x,y,W) represent the same 2D point, (x/W, y/W, 1)
- Represent 2D transforms with 3 by 3 matrices
- Can now represent translations by matrix multiplications

2D Scale in H.C.

$$M = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Rotation in H.C.

$$M = \left| \begin{array}{ccc} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{array} \right|$$

2D Translation in H.C.

- $\mathbf{P}_{\text{new}} = \mathbf{P} + \mathbf{T}$
- $(x', y') = (x, y) + (t_x, t_y)$

$$M = \left| \quad ? \quad \right|$$

Transformations in 3D

- Homogeneous coordinates now have four components (x, y, z, w)
  - ordinary to homogeneous: (x, y, z) -> (x, y, z, 1)
  - homogeneous to ordinary: (x, y, z, w) -> (x/w, y/w, z/w)
- Again, translation can be expressed as a multiplication.

Transformation examples in 3D

- Translation:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Anisotropic scaling:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Transformation examples in 3D

- Rotation about x-axis:

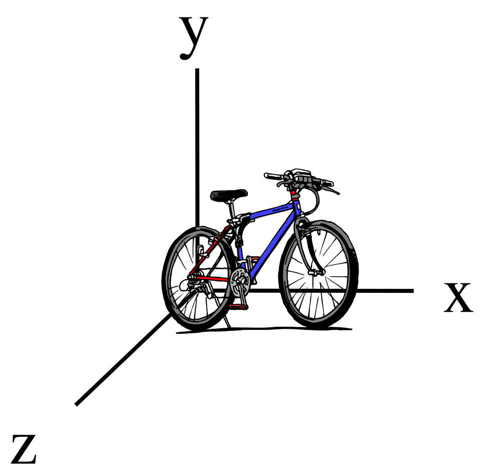
$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

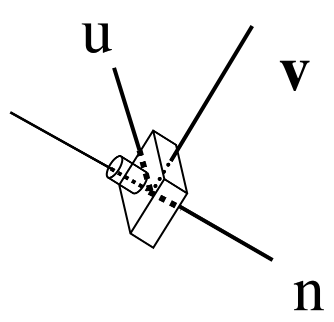
- See supplementary material for rotation about an arbitrary axis.

- A rotation matrix can be thought of as either a rotation about an axis, or a rigid transformation represented by an orthogonal matrix

First step of geometric camera model

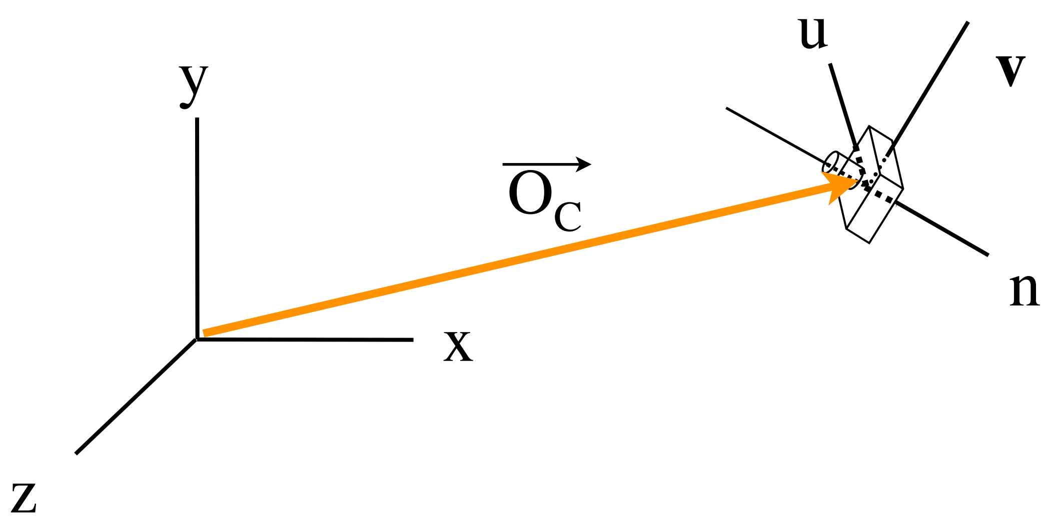
- Rewrite world coordinates as camera centric coordinates
  - Note that the origins are not the same and the axis are not aligned
  - Note that our rotation matrices are about an axis.
  - Hence we need to translate the world coordinates, and then rotate them.





Transform object expressed in
world coords to camera coords

Step 1. Rewrite world coordinates so that the camera at $\overrightarrow{O_C}$ in original coordinates is at the new origin. Call this T_1.



Transform object expressed in world coords to camera coords

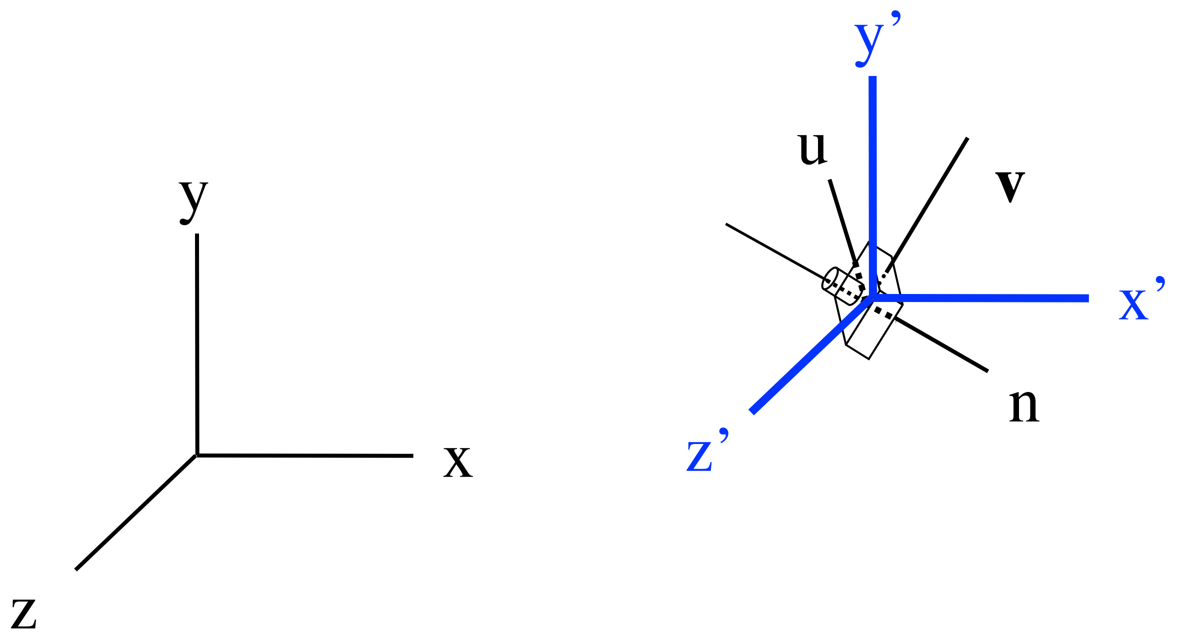
Step 1. T Rewrite world coordinates so that the camera at $\overrightarrow{O_C}$ in original coordinates is at the new origin. Call this T_1.

Translation vector is simply negative O_C.

(We want world coordinates to **change** so that the camera location **becomes** the origin).

Transform object expressed in world coords to camera coords

Step 2. Rewrite coordinates in the representation in step 1 so that **u** is x”, **v** is y”, and **n** is z”. The matrix is ?



Transform object expressed in world coords to camera coords

Step 2. Rewrite coordinates in the representation in step 1 so that **u** is **x"**, **v** is **y"**, and **n** is **z"**. The matrix is ?

(We want the previous object 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 expressed in world coords to camera coords

Step 2. Rewrite coordinates in the representation in step 1 so that **u** is **x”**, **v** is **y”**, and **n** is **z”**. The matrix is:

$$\left| \begin{array}{cc} \mathbf{u}^T & 0 \\ \mathbf{v}^T & 0 \\ \mathbf{n}^T & 0 \\ 0\ 0\ 0 & 1 \end{array} \right|$$

(why?)

Transform object expressed in world coords to camera coords

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

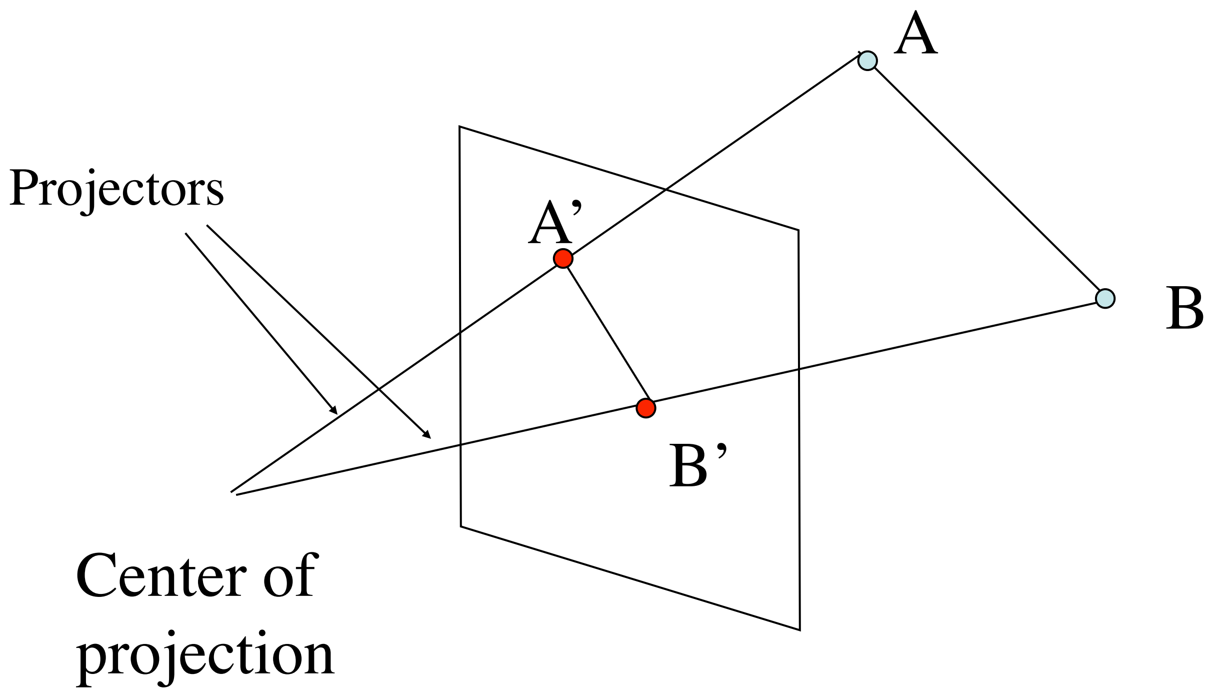
In the current coords **u** maps into the X-axis unit vector (1,0,0,1) which is what we want.

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

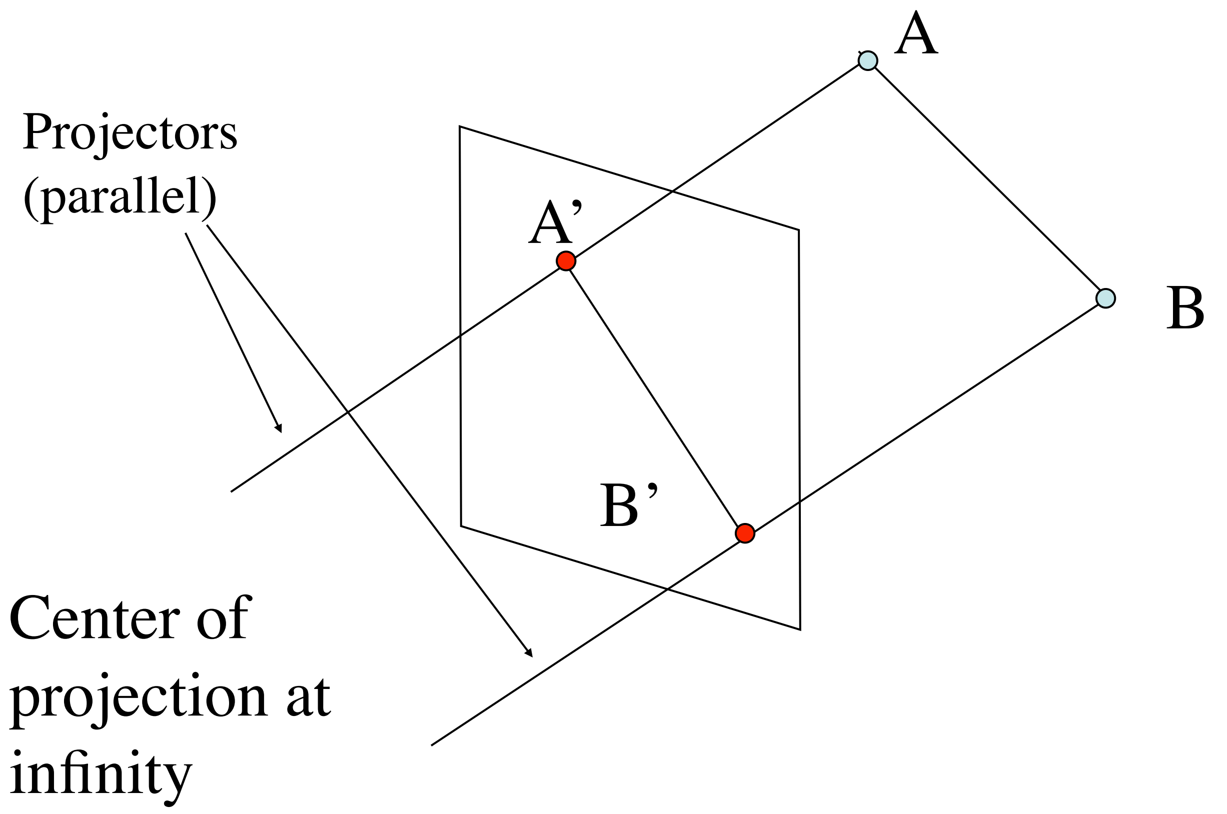
Projections

- Want to think about geometric image formation as a mathematical transformation taking points in the 3D world and mapping them into an image plane.
- 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

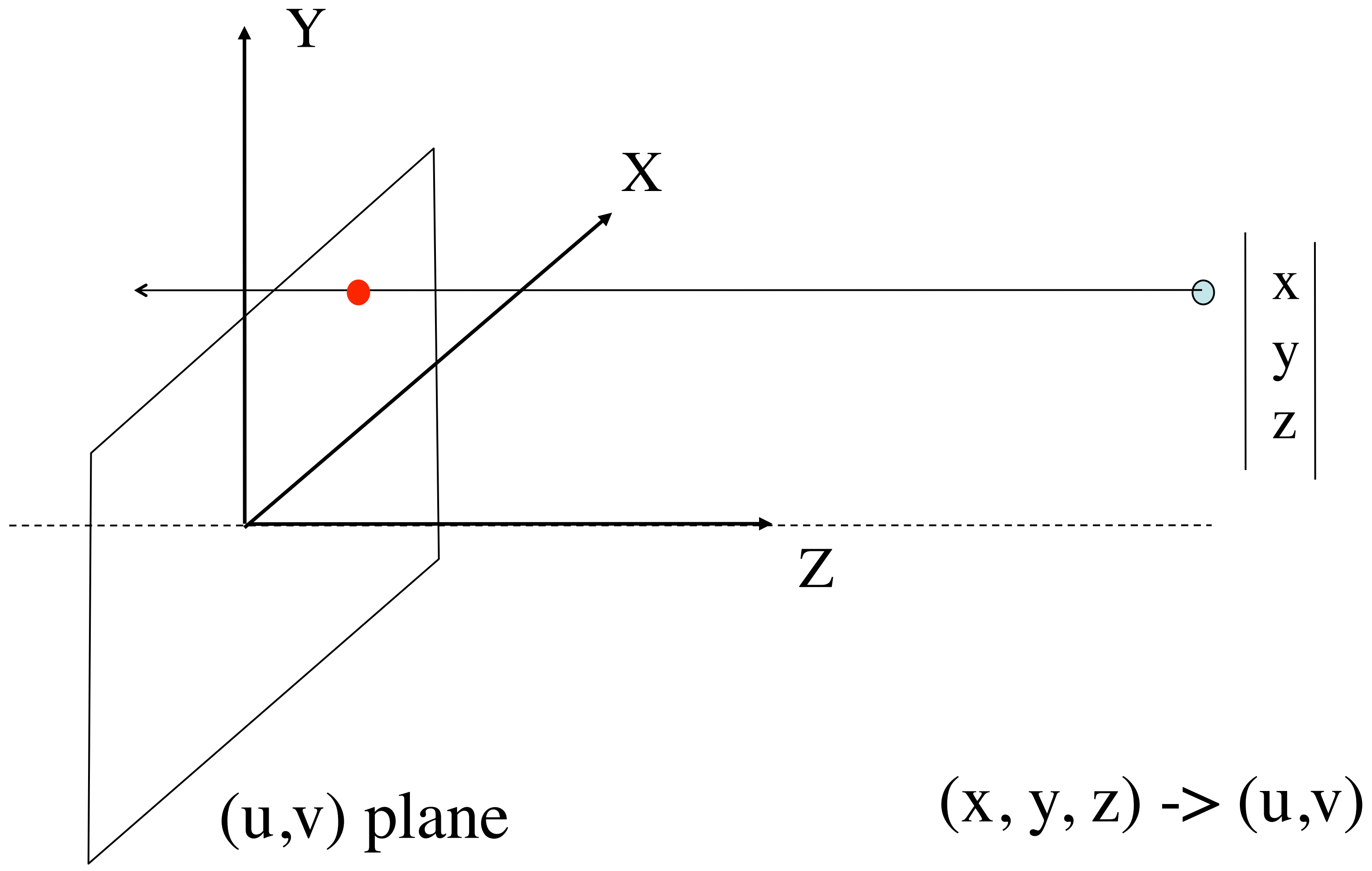


Parallel Projection

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

If projection plane is perpendicular to projectors the projection is orthographic

Orthographic example (onto z=0)



(x, y, z) -> (u,v)

The equation of projection (orthographic, onto z=0)

- In homogeneous coordinates

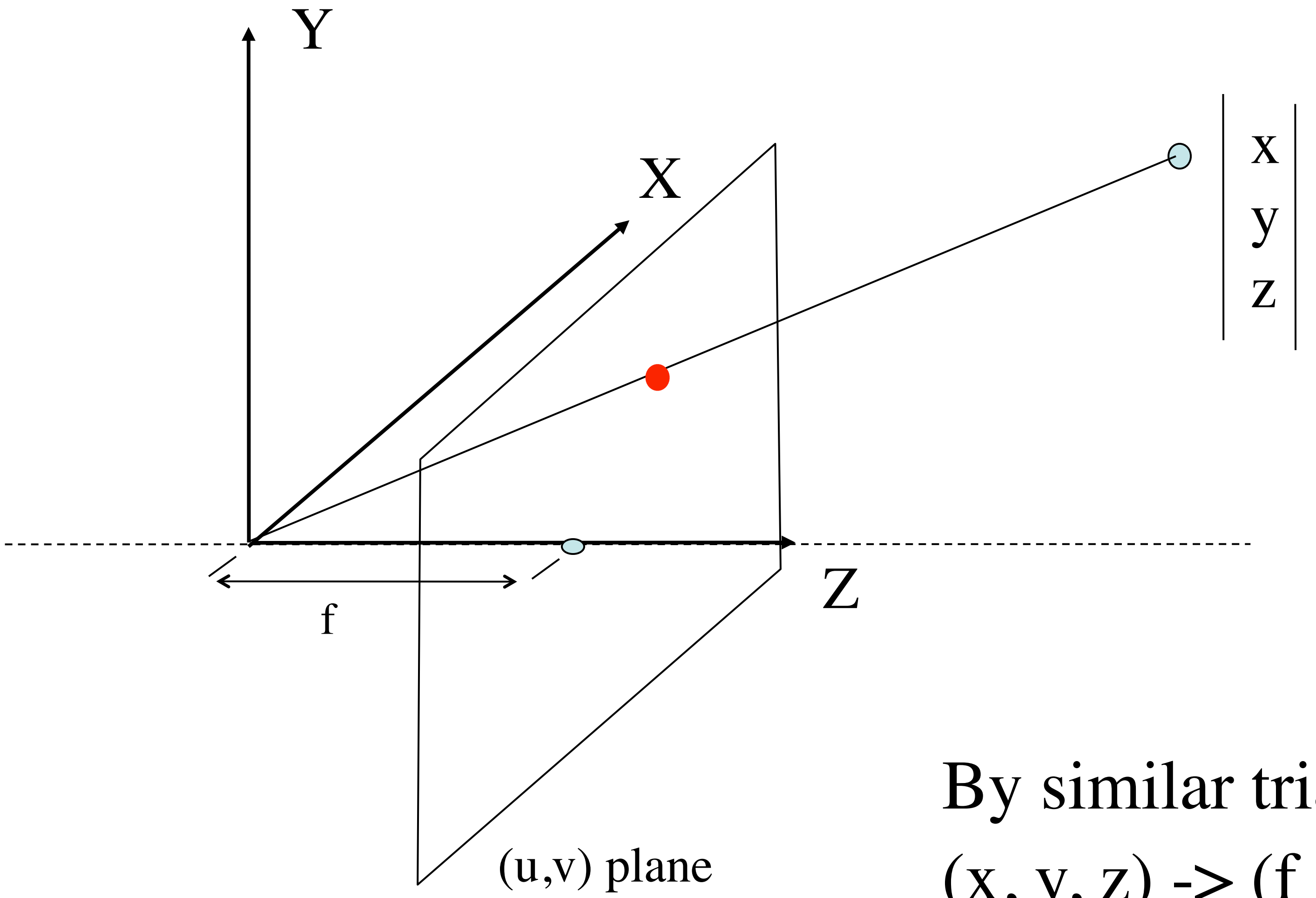
$$(x, y, z, 1) \Rightarrow (x, y, 1)$$

- Graphics course survivors: You might notice slight changes in style to be consistent with the book. Perhaps most notably we will explicitly, rather than implicitly, ignore the third projected coordinate, so projection matrices will be 3 by 4 not 4 by 4.

The projection matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective example (onto z=f)



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

The equation of projection

- In homogeneous coordinates

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

- Equivalently

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

- Homogeneous coordinates are being used to store foreshortening

- Note that using regular coordinates does **not** yield a linear transformation (inconvenient!).

The projection matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix}$$

Camera matrix, M

Actual pixel coords are
(u,v) = (U/W, V/W)

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \overbrace{\begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{intrinsic parameters} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{extrinsic parameters} \end{pmatrix}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Projection. We use **f=1** and
let the intrinsic parameters
absorb the focal length.

First part makes it so that we are
in standard camera coords
where we know how to project.