Projections

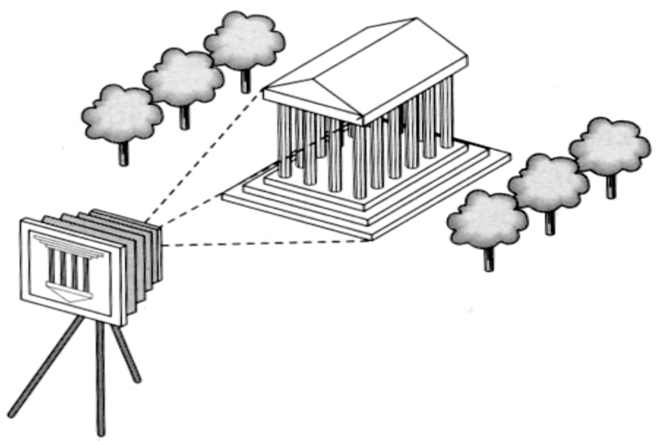
Reading

Angel. Chapter 5

Optional

David F. Rogers and J. Alan Adams, *Mathematical Elements for Computer Graphics, Second edition*, McGraw-Hill, New York, 1990, Chapter 3.

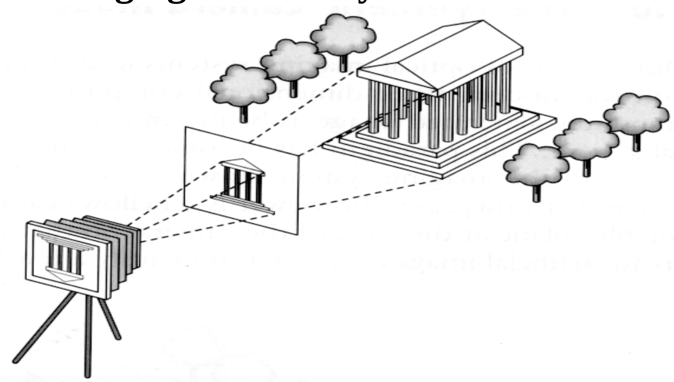
The 3D synthetic camera model



The **synthetic camera model** involves two components, specified *independently:*

- objects (a.k.a. **geometry**)
- viewer (a.k.a. **camera**)

Imaging with the synthetic camera

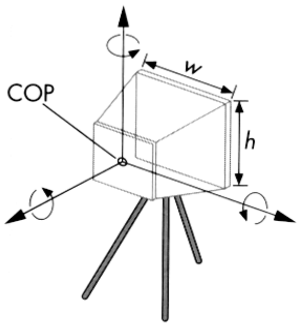


The image is rendered onto an **image plane** or **projection plane** (usually in front of the camera).

Projectors emanate from the **center of projection** (COP) at the center of the lens (or pinhole).

The image of an object point *P* is at the intersection of the projector through *P* and the image plane.

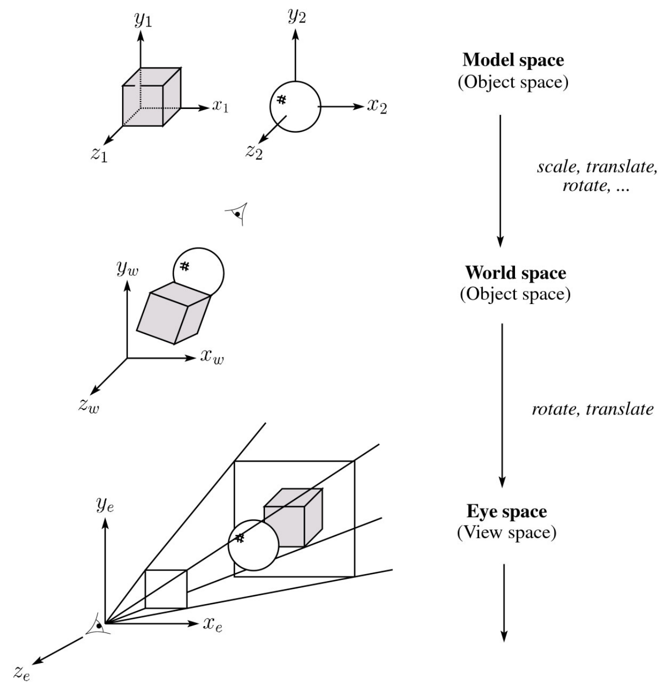
Specifying a viewer

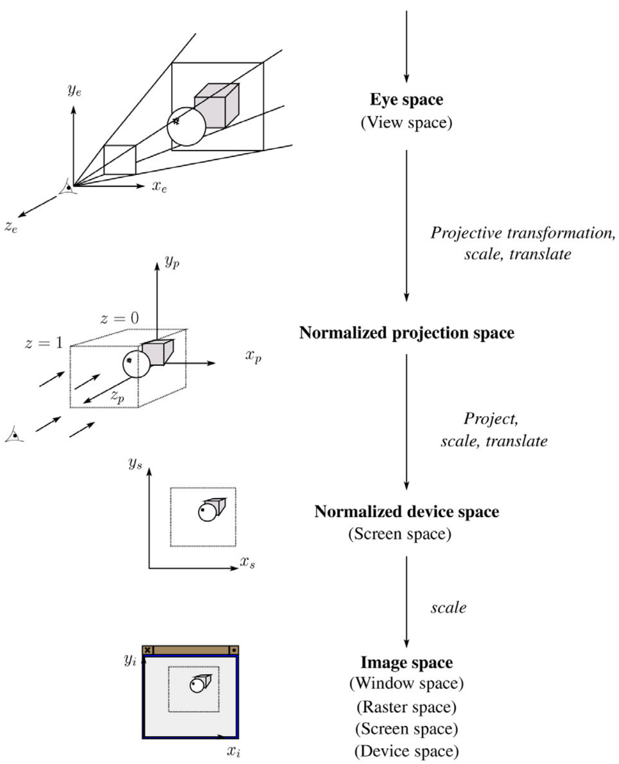


Camera specification requires four kinds of parameters:

- *Position:* the COP.
- *Orientation:* rotations about axes with origin at the COP.
- *Focal length:* determines the size of the image on the film plane, or the **field of view**.
- *Film plane:* its width and height, and possibly orientation.

3D Geometry Pipeline

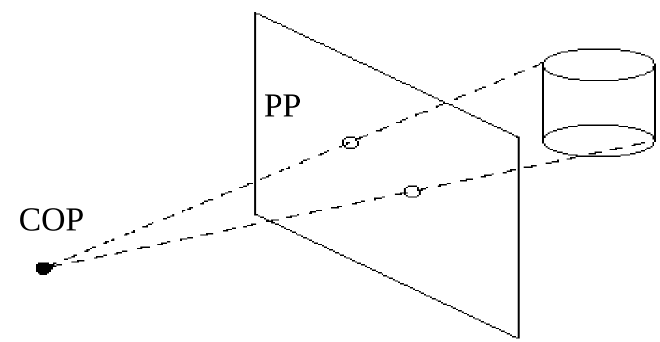




Projections

Projections transform points in *n*-space to *m*-space, where $m < n$.

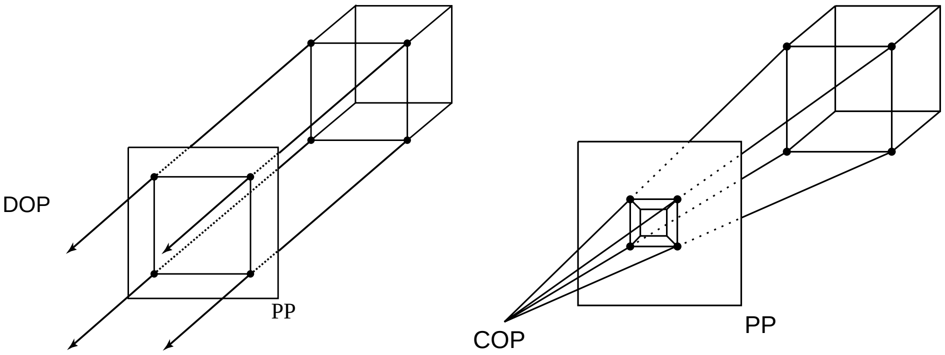
In 3D, we map points from 3-space to the **projection plane (PP)** along **projectors** emanating from the **center of projection (COP)**.



There are two basic types of projections:

- **Perspective** - distance from COP to PP finite
- **Parallel** - distance from COP to PP infinite

Parallel and Perspective Projection



Perspective vs. parallel projections

Perspective projections pros and cons:

- \+ Size varies inversely with distance - looks realistic
- – Distance and angles are not (in general) preserved
- – Parallel lines do not (in general) remain parallel

Parallel projection pros and cons:

- – Less realistic looking
- \+ Good for exact measurements
- \+ Parallel lines remain parallel
- – Angles not (in general) preserved

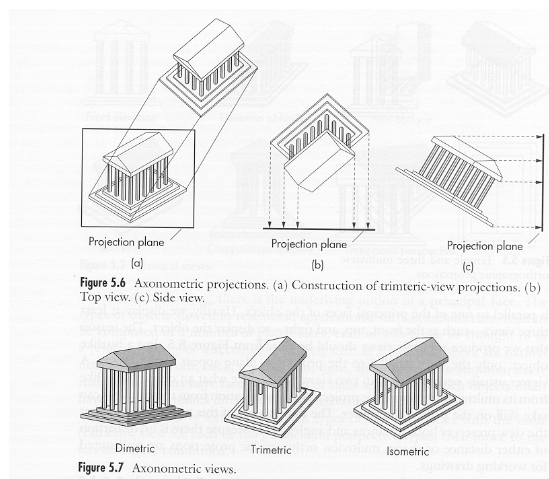
Parallel projections

For parallel projections, we specify a **direction of projection (DOP)** instead of a COP.

There are two types of parallel projections:

- **Orthographic projection** — DOP perpendicular to PP
- **Oblique projection** — DOP not perpendicular to PP

Orthographic Projections



Orthographic transformation

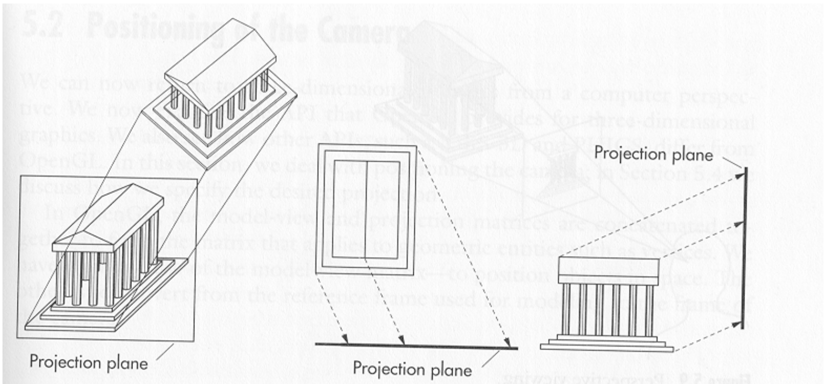
For parallel projections, we specify a **direction of projection** (DOP) instead of a COP.

We can write orthographic projection onto the *z=0* plane with a simple matrix.

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

Normally, we do not drop the z value right away. Why not?

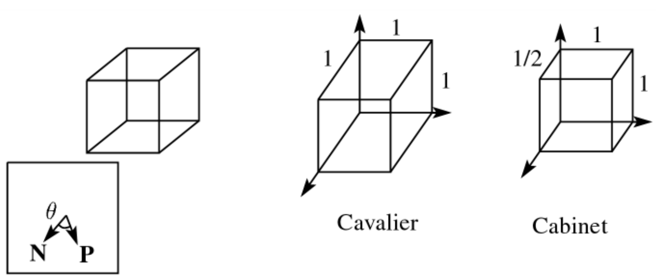
Oblique Projections



Oblique projections

Two standard oblique projections:

- Cavalier projection
  DOP makes 45 angle with PP
  Does not foreshorten lines perpendicular to PP
- Cabinet projection
  DOP makes 63.4 angle with PP
  Foreshortens lines perpendicular to PP by one-half

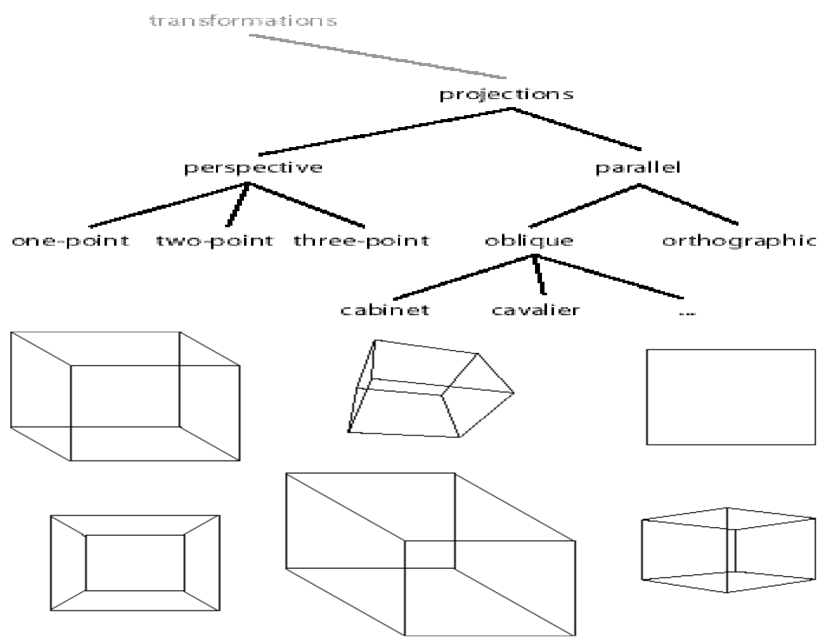


Oblique projection geometry

Cavalier

Cabinet

Projection taxonomy

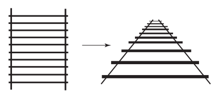


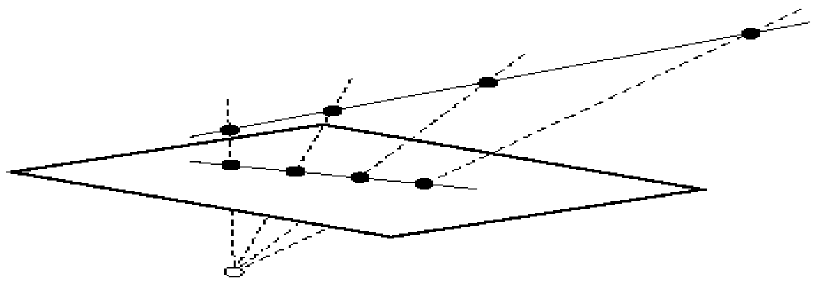
Properties of projections

The perspective projection is an example of a **projective transformation**.

Here are some properties of projective transformations:

- Lines map to lines
- Parallel lines *don't* necessarily remain parallel
- Ratios are *not* preserved

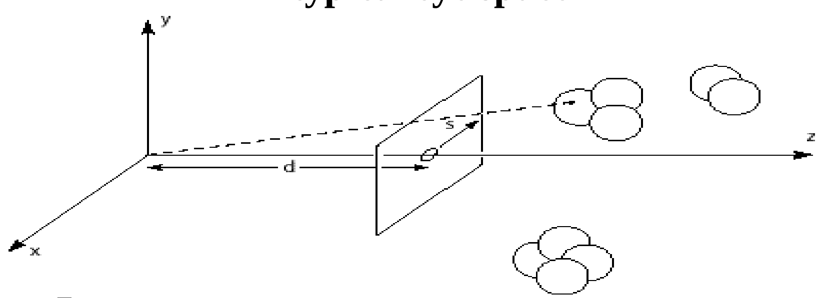




Coordinate systems for CG

- **Model space** — for describing the objections (aka "object space", "world space")
- **World space** — for assembling collections of objects (aka "object space", "problem space", "application space")
- **Eye space** — a canonical space for viewing (aka "camera space")
- **Screen space** — the result of perspective transformation (aka "normalized device coordinate space", "normalized projection space")
- **Image space** — a 2D space that uses device coordinates (aka "window space", "screen space", "normalized device coordinate space", "raster space")

A typical eye space

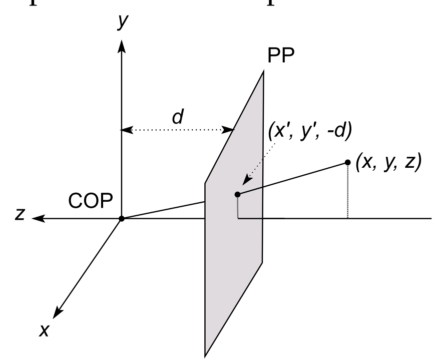


- **Eye**
  - Acts as the COP
  - Placed at the origin
  - Looks down the *z*-axis
- **Screen**
  - Lies in the PP
  - Perpendicular to *z*-axis
  - At distance *d* from the eye
  - Centered on *z*-axis, with radius *s*

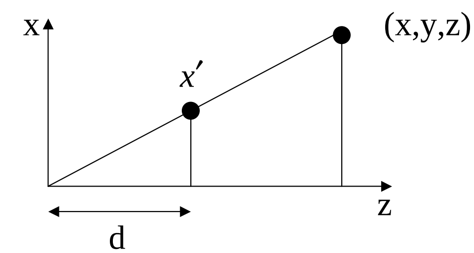
Q: Which objects are visible?

Eye space ➔ screen space

Q: How do we perform the perspective projection from eye space into screen space?



Using similar triangles gives:



Eye space → screen space, cont.

We can write this transformation in matrix form:

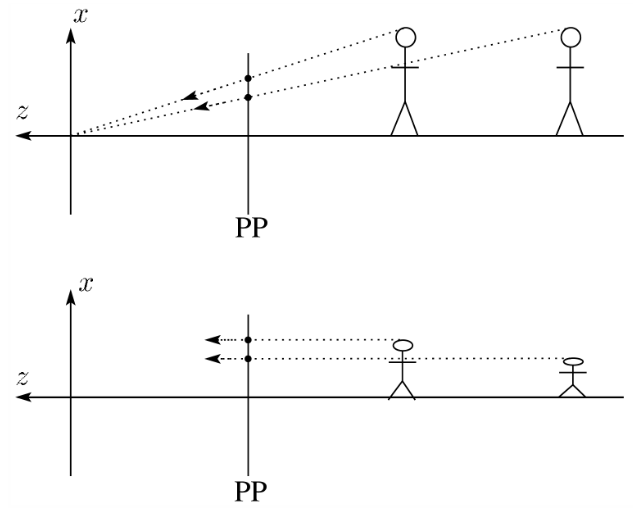
$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = MP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

Perspective divide:

$$\begin{bmatrix} X/W \\ Y/W \\ Z/W \\ W/W \end{bmatrix} = \begin{bmatrix} \dfrac{x}{z/d} \\ \dfrac{y}{z/d} \\ d \\ 1 \end{bmatrix}$$

Projective Normalization

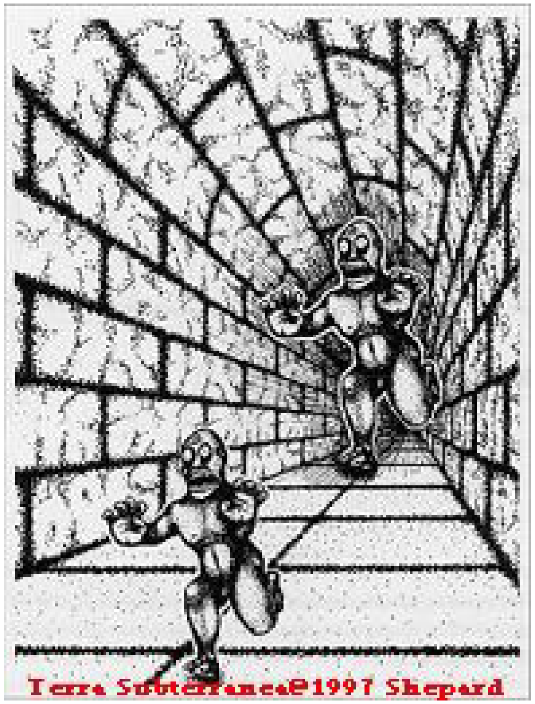
After perspective transformation and perspective divide, we apply parallel projection (drop the z) to get a 2D image.

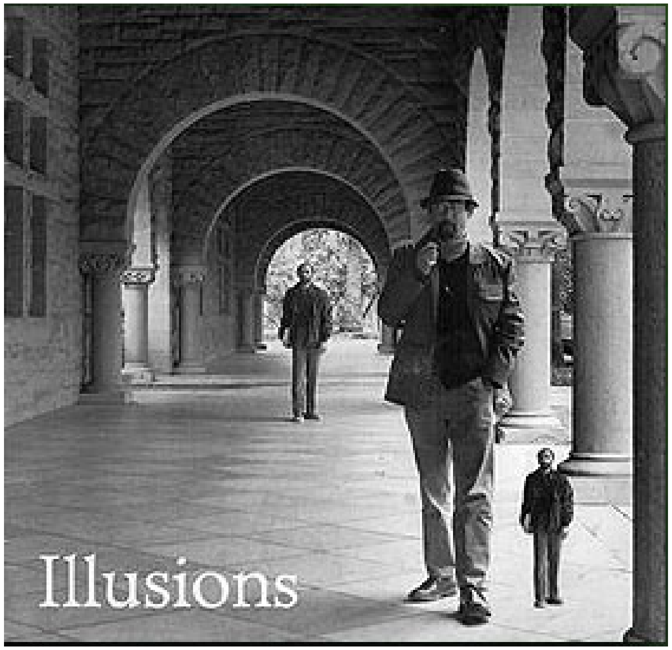


Perspective depth

Q: What did our perspective projection do to *z*?

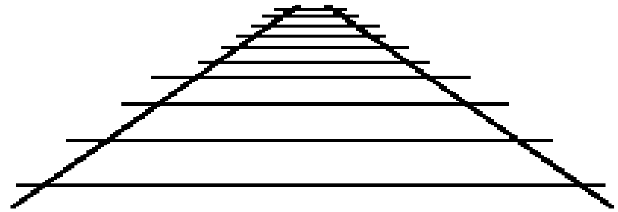
Often, it's useful to have a *z* around — e.g., for hidden surface calculations.





Vanishing points

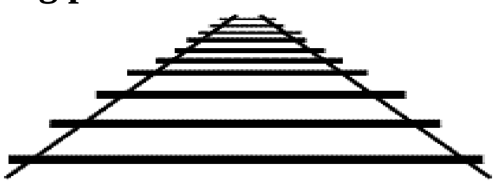
Under perspective projections, any set of parallel lines that are not parallel to the PP will converge to a **vanishing point**.



Vanishing points of lines parallel to a principal axis *x*, *y*, or *z* are called **principal vanishing points**.

How many of these can there be?

Vanishing points



The equation for a line is:

$$\mathbf{l} = \mathbf{p} + t\mathbf{v} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} + t \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$$

After perspective transformation we get:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} p_x + tv_x \\ p_y + tv_y \\ -(p_z + tv_z)/d \end{bmatrix}$$

Vanishing points (cont'd)

Dividing by *w*:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} -\dfrac{p_x + tv_x}{p_z + tv_z} d \\ -\dfrac{p_y + tv_y}{p_z + tv_z} d \\ 1 \end{bmatrix}$$

Letting *t* go to infinity:

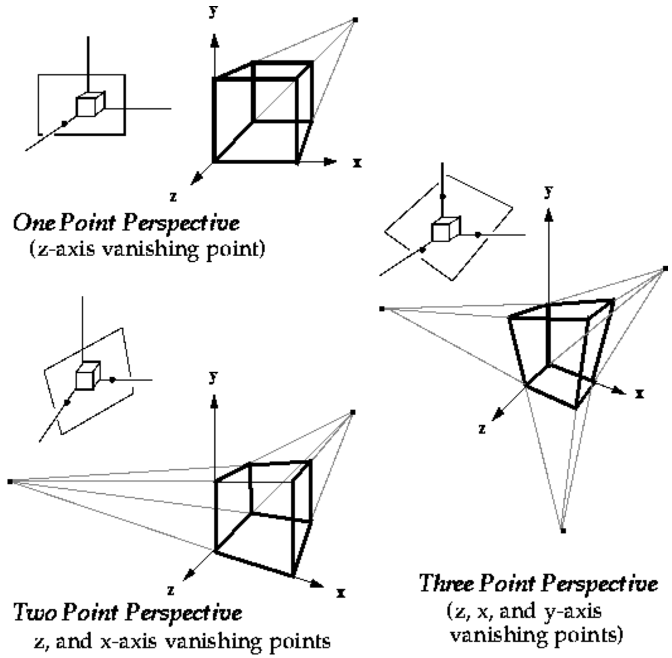
We get a point!

What happens to the line **l** = **q** + *t***v**?

Each set of parallel lines intersect at a **vanishing point** on the PP.

Q: How many vanishing points are there?

Vanishing Points



Types of perspective drawing

If we define a set of **principal axes** in world coordinates, i.e., the x_w, y_w, and z_w axes, then it's possible to choose the viewpoint such that these axes will converge to different vanishing points.

The vanishing points of the principal axes are called the **principal vanishing points**.

Perspective drawings are often classified by the number of principal vanishing points.

- One-point perspective — simplest to draw
- Two-point perspective — gives better impression of depth
- Three-point perspective — most difficult to draw

All three types are equally simple with computer graphics.

General perspective projection

In general, the matrix

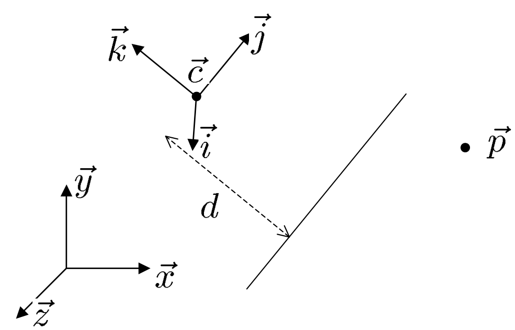
$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ p & q & r & s \end{bmatrix}$$

performs a perspective projection into the plane
$px + qy + rz + s = 1$.

Q: Suppose we have a cube C whose edges are aligned with the principal axes. Which matrices give drawings of C with

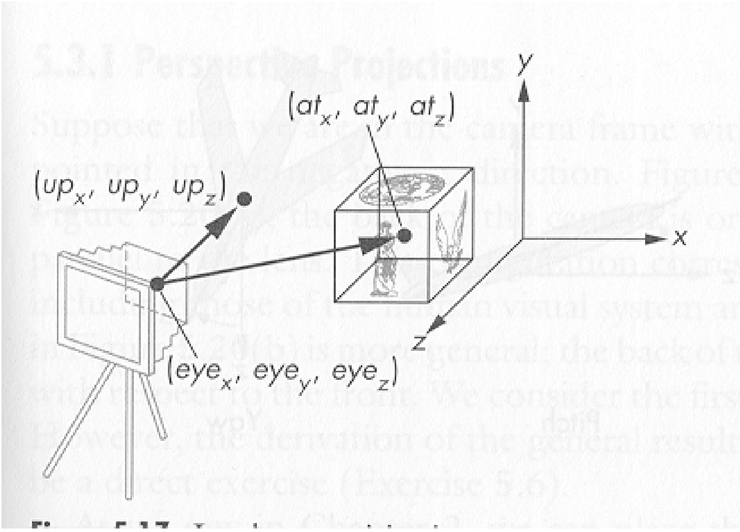
- one-point perspective?
- two-point perspective?
- three-point perspective?

General Projections



Suppose you have a camera with COP c, and x, y, and z axes are unit vectors i, j and k respectively. How do we compute the projection?

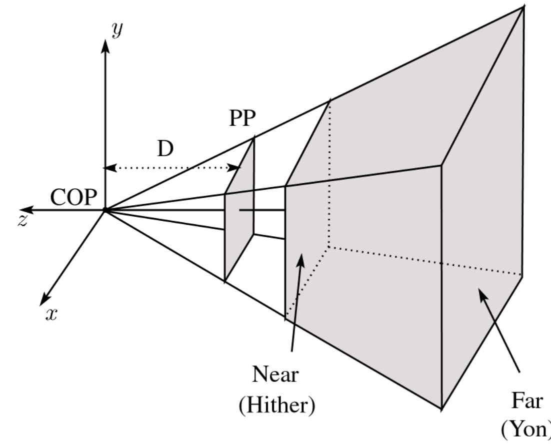
World Space Camera



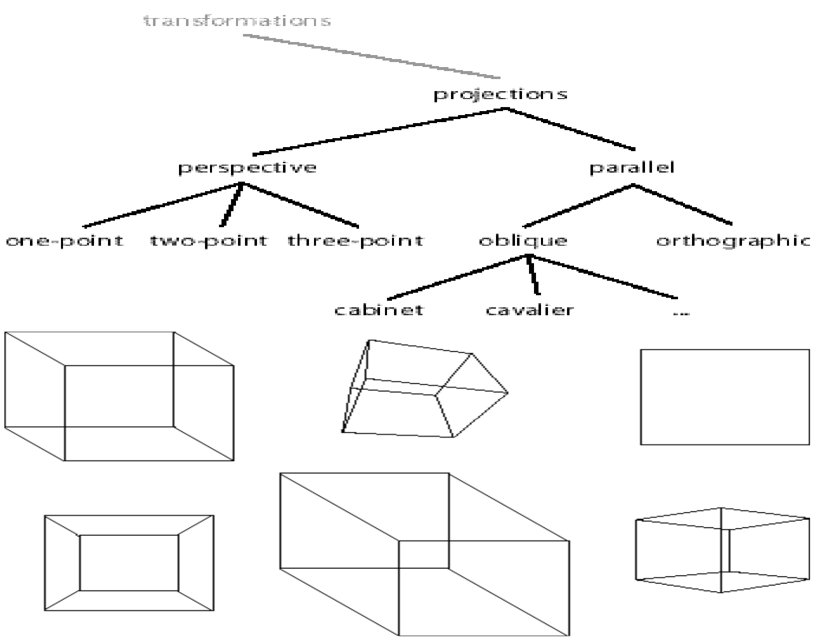
Hither and yon planes

In order to preserve depth, we set up two planes:

- The **hither** (near) plane
- The **yon** (far) plane



Projection taxonomy



Summary

Here's what you should take home from this lecture:

- The classification of different types of projections.
- The concepts of vanishing points and one-, two-, and three-point perspective.
- An appreciation for the various coordinate systems used in computer graphics.
- How the perspective transformation works.