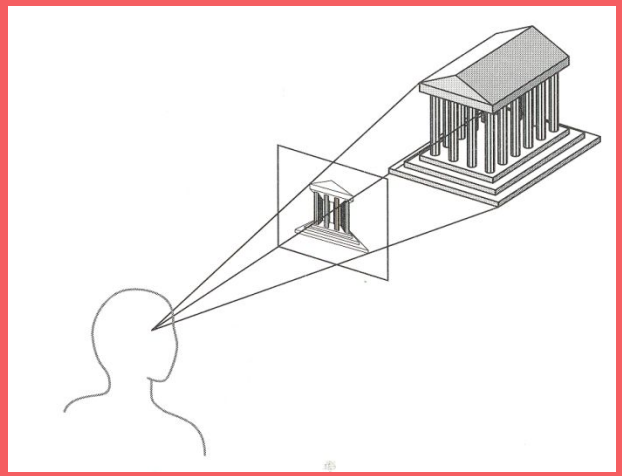


Angel: Interactive Computer Graphics

CSE 409: Computer Graphics

Camera Transformations and Projection

Acknowledgements: parts of information and pictures has been collected from the University of Virginia.

Projection

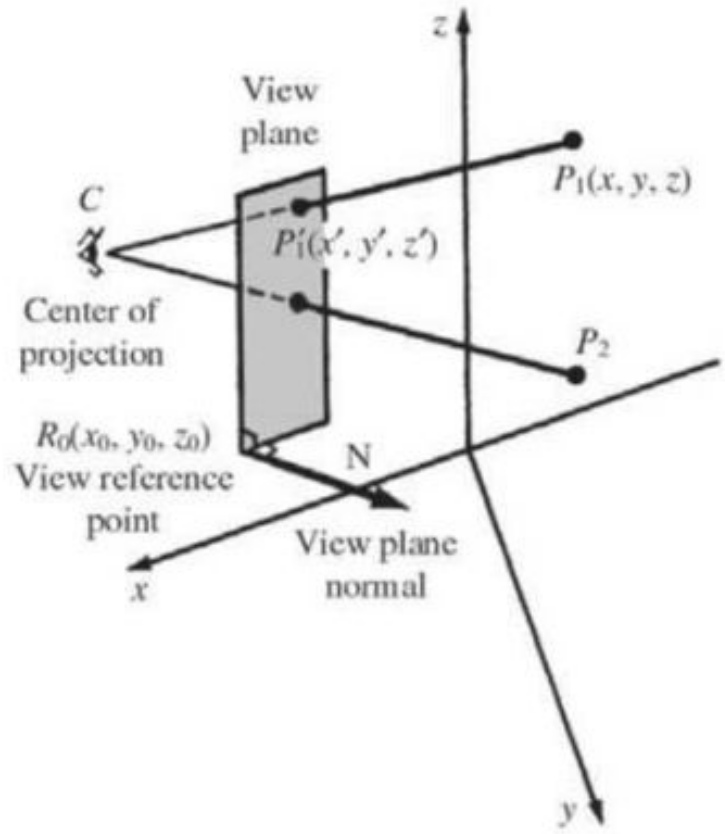
In general, projections transform points in a coordinate system of dimension n into points in a coordinate system of dimension less than n.

We shall limit ourselves to the projection from 3D to 2D.

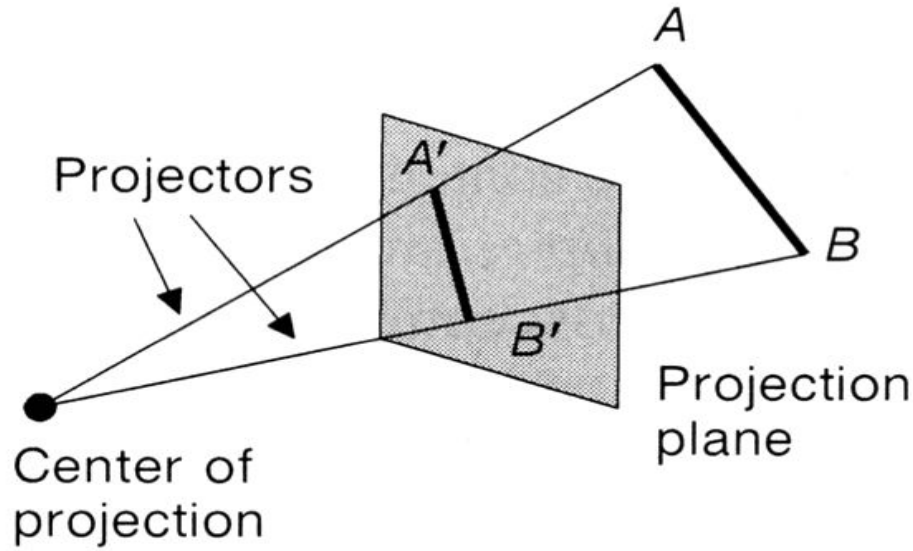
We will deal with planar geometric projections where:

- The projection is onto a plane rather than a curved surface
- The projectors are straight lines rather than curves

Projection


z
View
plane
C
$P_1(x, y, z)$
$P_1'(x', y', z')$
Center of
projection
P_2
$R_0(x_0, y_0, z_0)$
View reference
point
N
View plane
normal
x
y

Projection



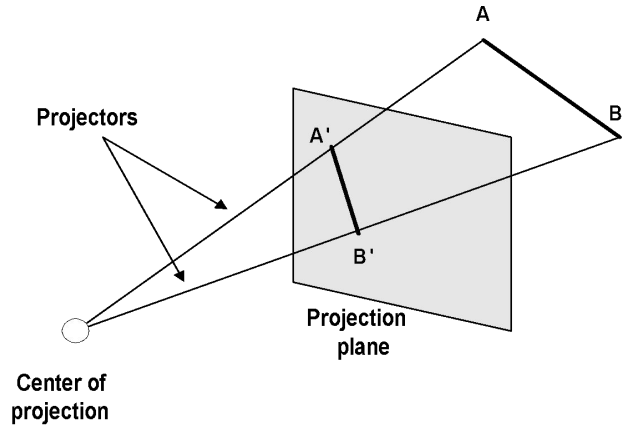
Planer Geometric Projection

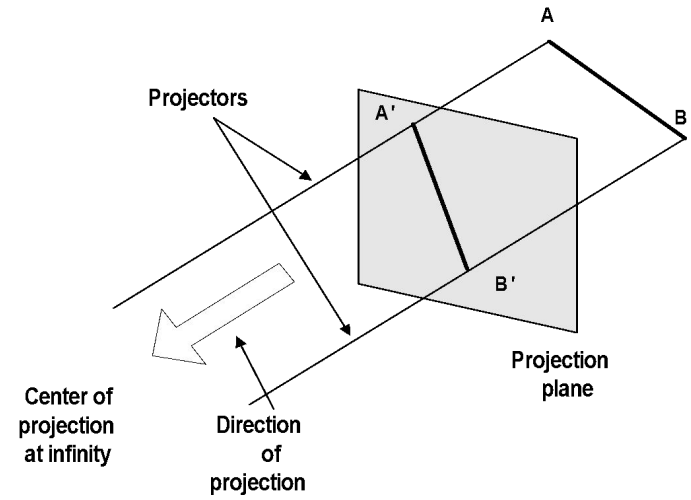
2 types of projections

- *perspective* and *parallel*.

Key factor is the center of projection.

- if distance to center of projection is finite : perspective
- if infinite : parallel

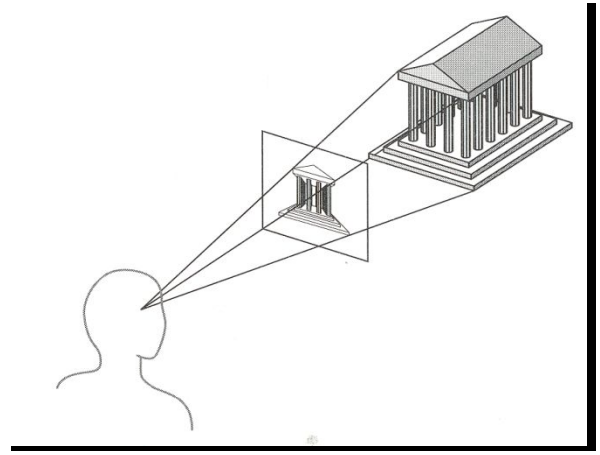




Perspective v Parallel

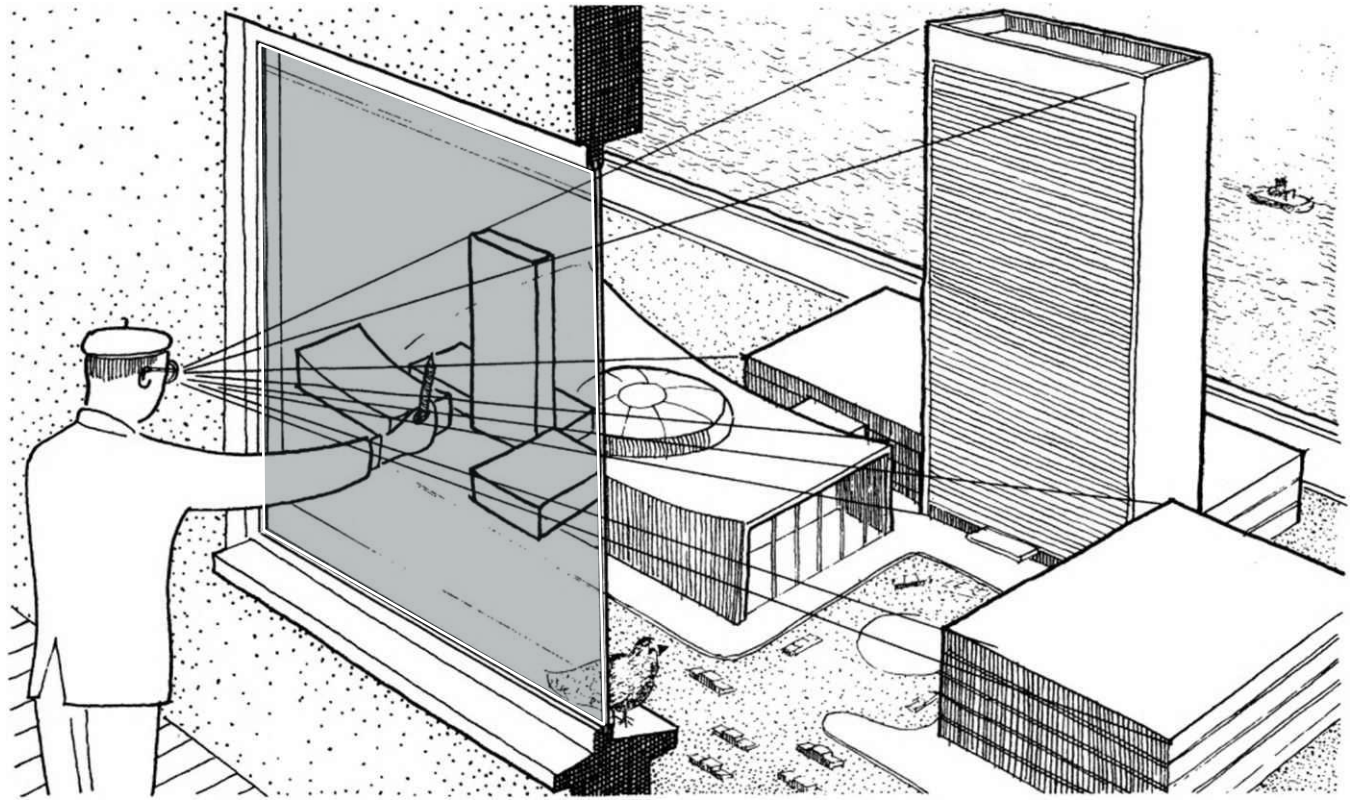
Perspective:

- visual effect is similar to human visual system...
- has 'perspective foreshortening'
  - size of object varies inversely with distance from the center of projection.
- Parallel lines do not in general project to parallel lines
- angles only remain intact for faces parallel to projection plane.
- Vanishing point



Perspective v Parallel

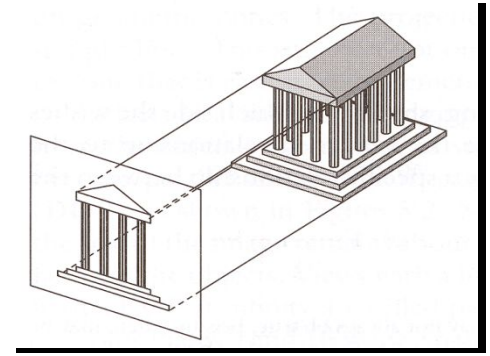
Perspective:



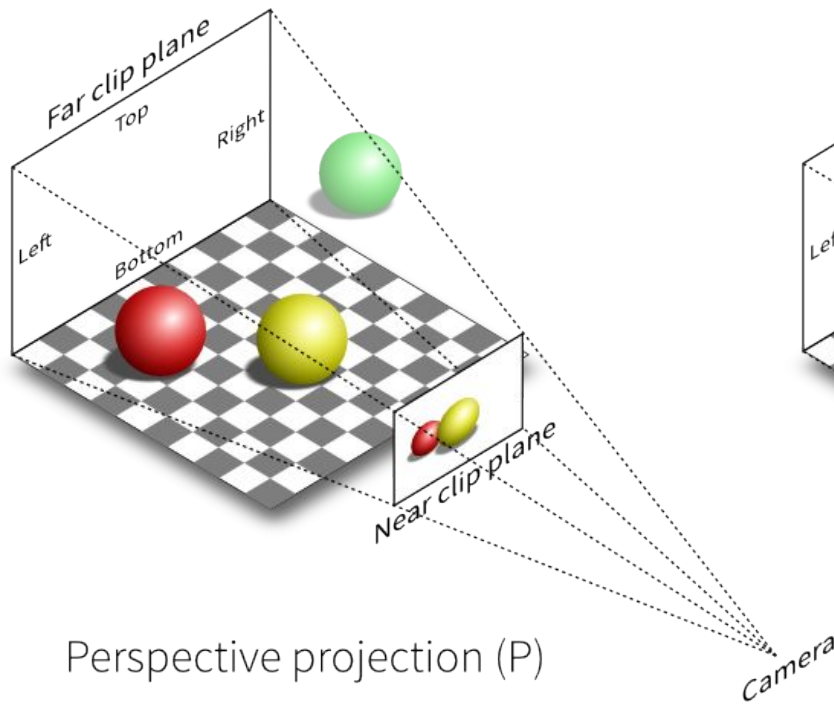
Parallel

Parallel:

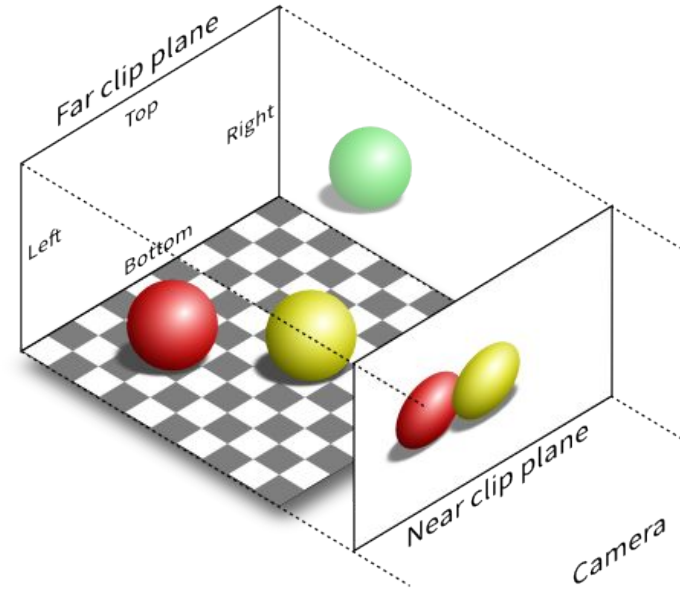
- less realistic view because of no foreshortening
- however, parallel lines remain parallel.
- angles only remain intact for faces parallel to projection plane.



Perspective v Parallel



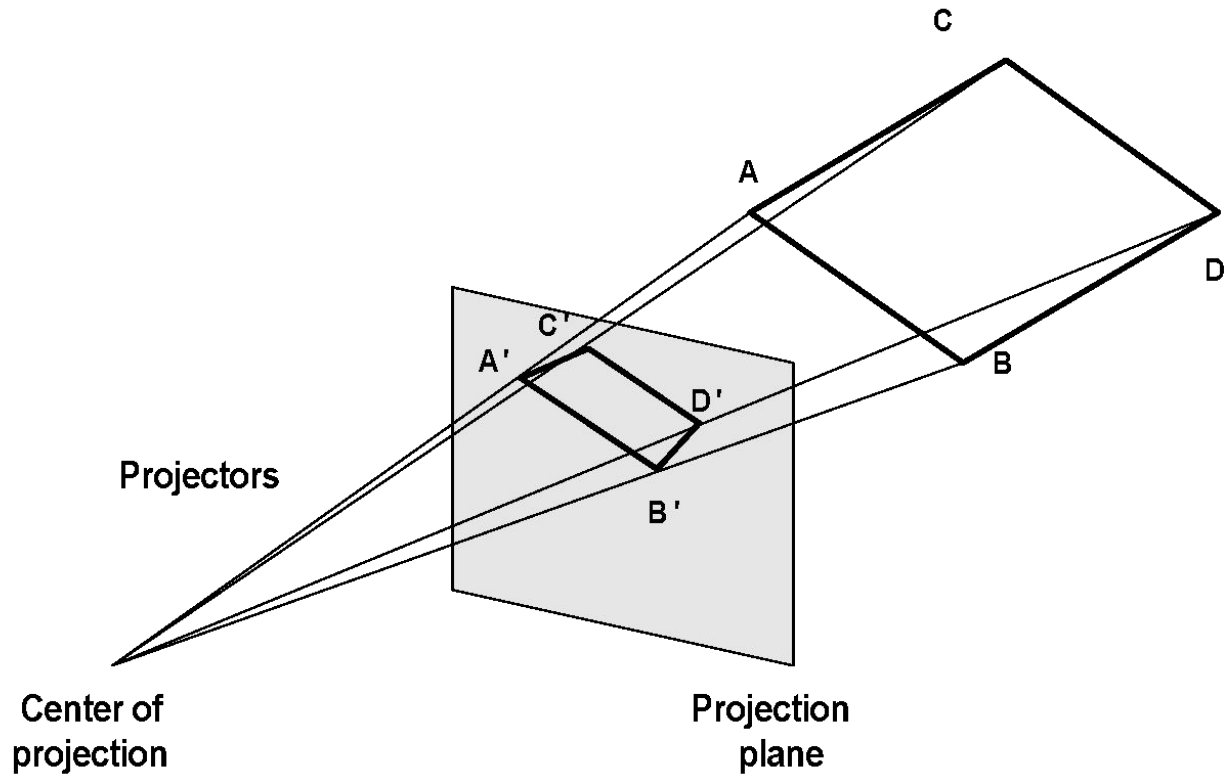
Perspective projection (P)



Orthographic projection (O)

Perspective projection- anomalies

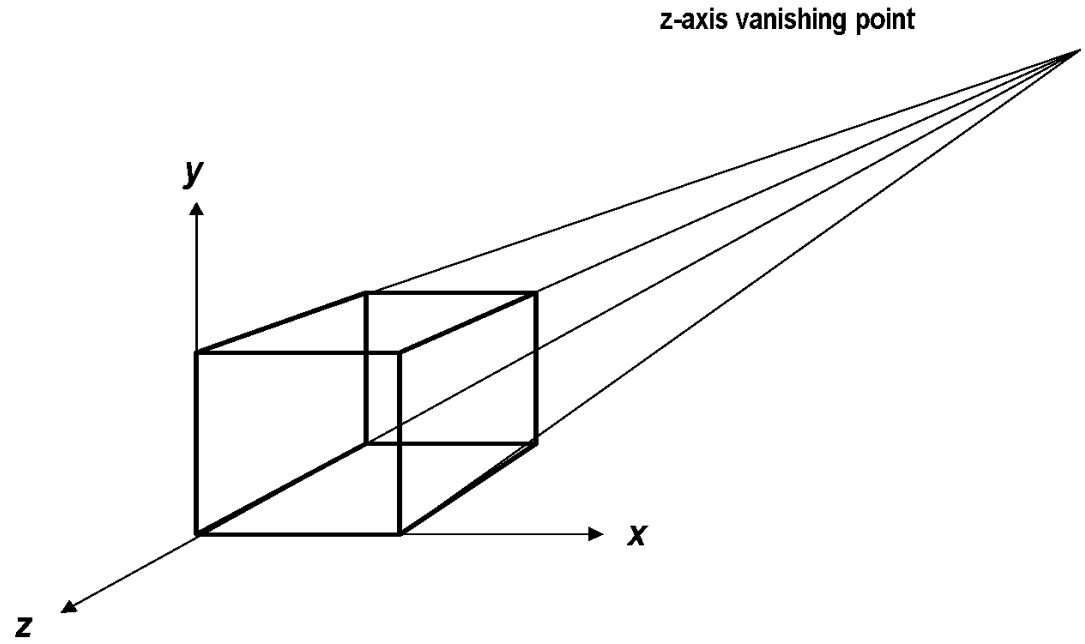
Perspective foreshortening The farther an object is from COP the smaller it appears



Perspective projection- anomalies

Vanishing Points: Any set of parallel lines not parallel to the view plane appear to meet at some point.

- There are an infinite number of these, 1 for each of the infinite amount of directions line can be oriented



Vanishing Point

Vanishing Point

If a set of lines are parallel to one of the three axes, the vanishing point is called an axis vanishing point (Principal Vanishing Point).

- There are at most 3 such points, corresponding to the number of axes cut by the projection plane

One-point:

- One principle axis cut by projection plane
- One axis vanishing point

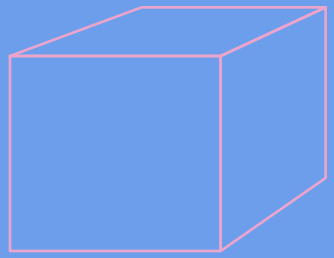
Two-point:

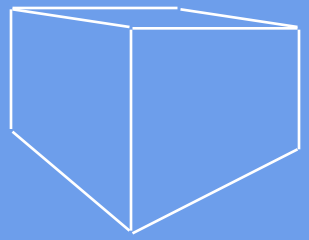
- Two principal axes cut by projection plane
- Two axis vanishing points

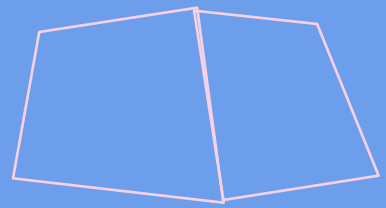
Three-point:

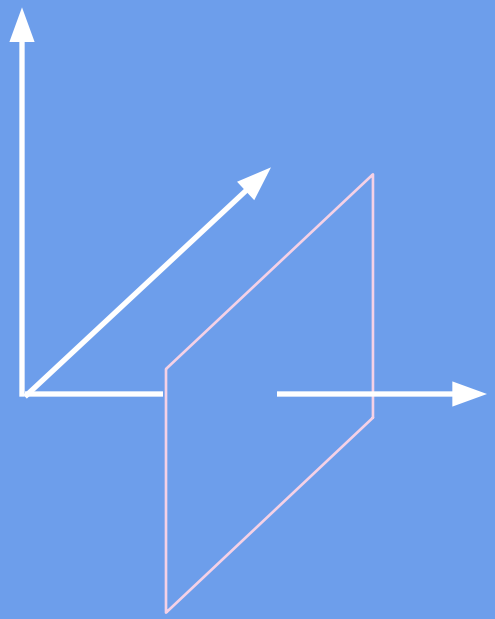
- Three principle axes cut by projection plane
- Three axis vanishing points

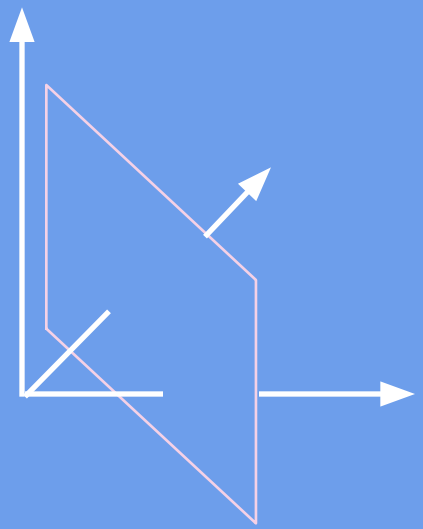
Vanishing Point

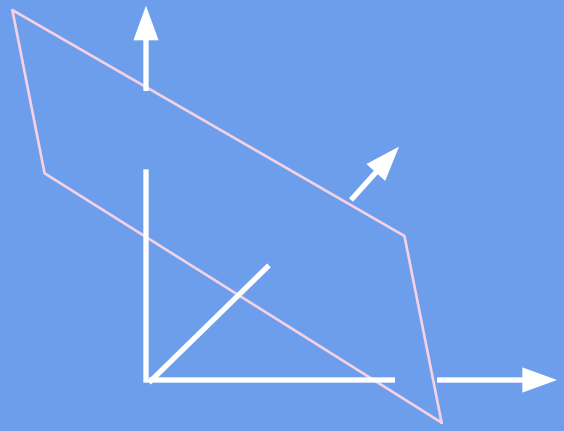






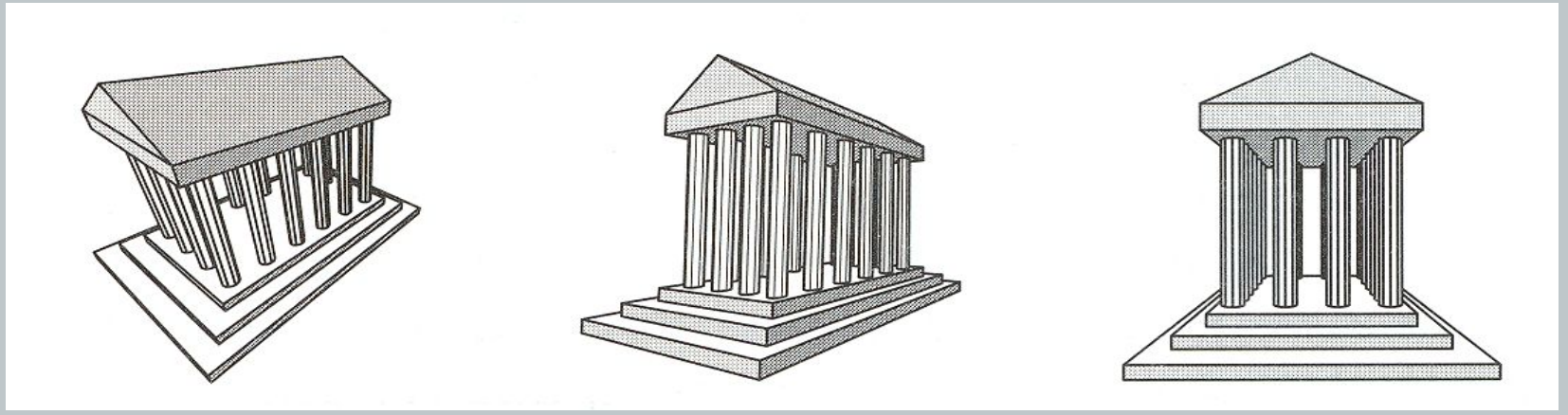






Perspective Projection

How many vanishing points?



3-Point Perspective

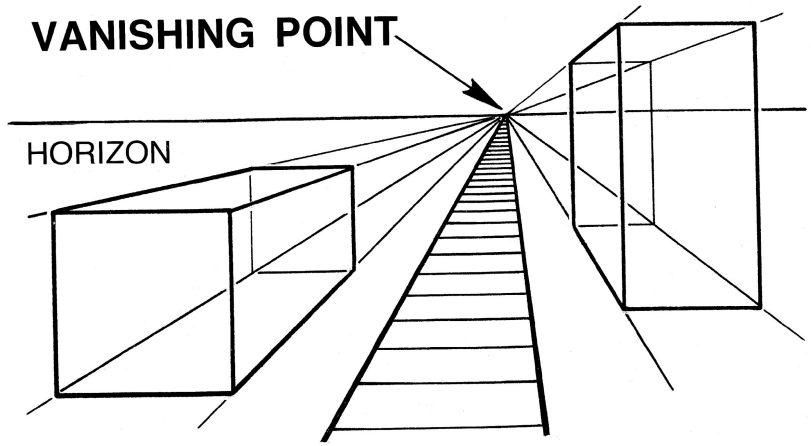
2-Point Perspective

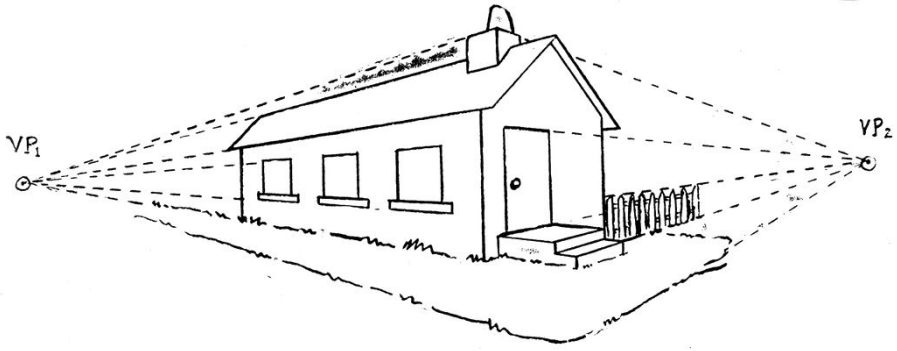
1-Point Perspective

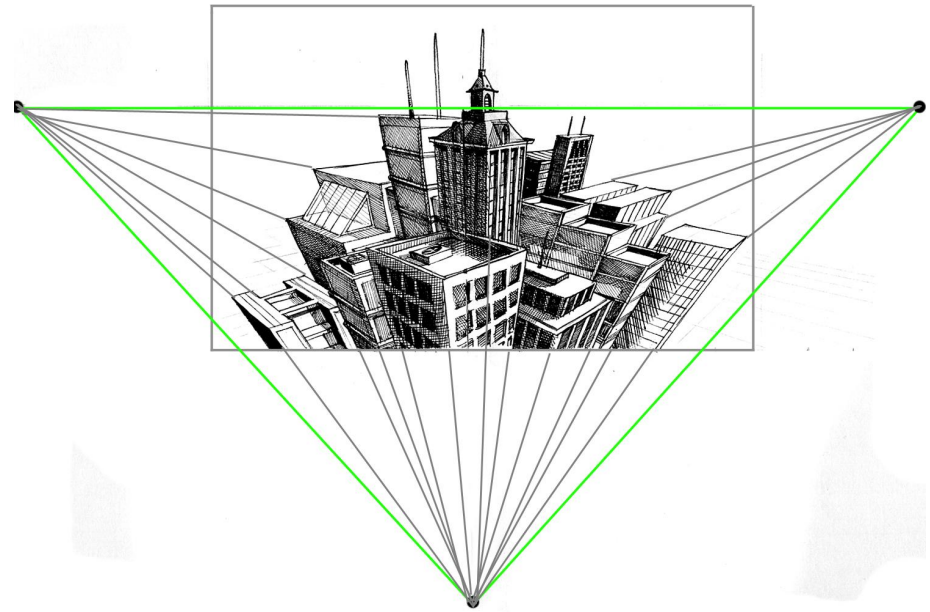
Angel Figure 5.10

Perspective Projection

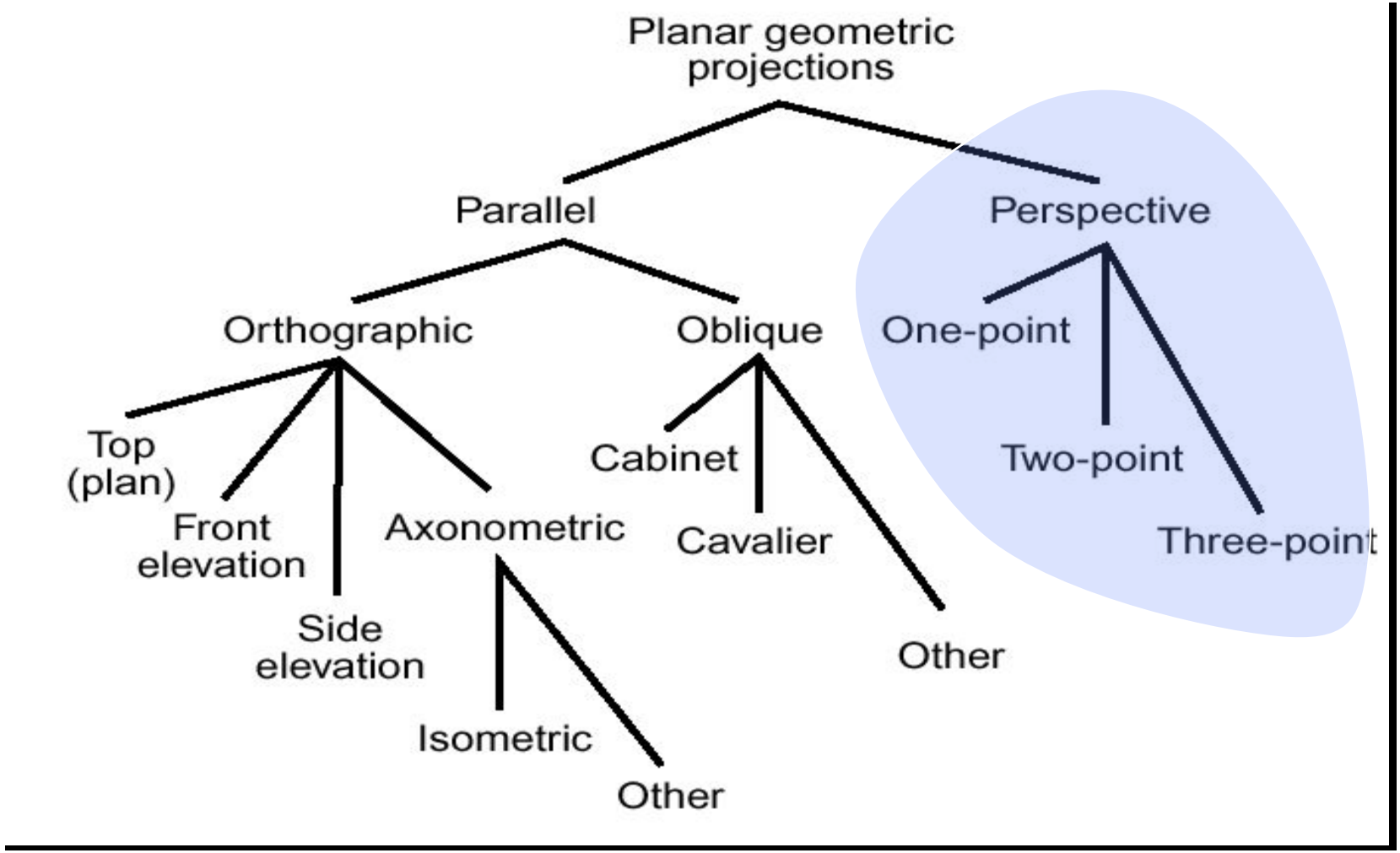
How many vanishing points?







Taxonomy of Projection XM



Parallel projection

2 principle types:

- *orthographic* and *oblique*.

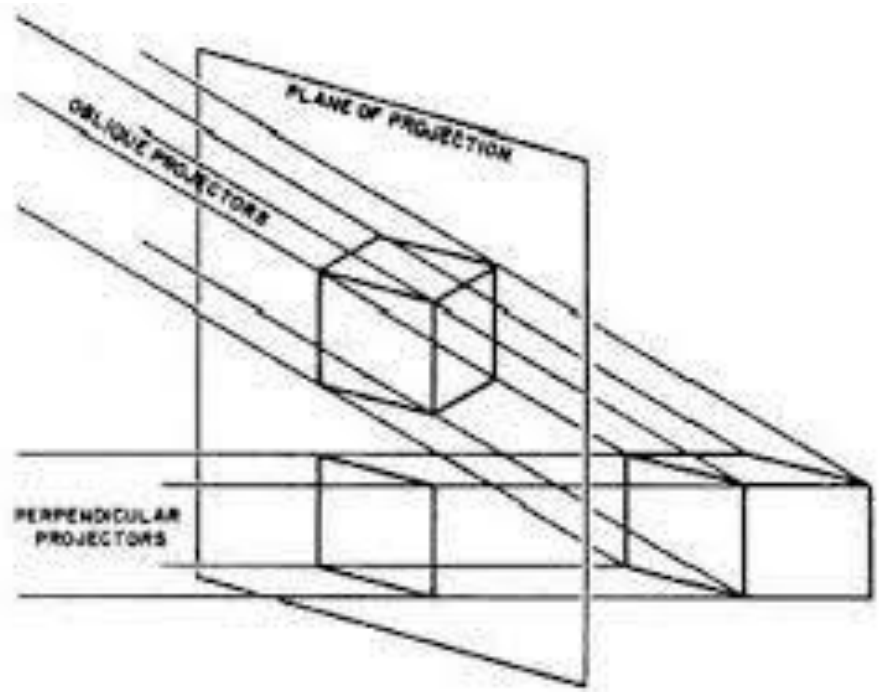
Orthographic :

- direction of projection = normal to the projection plane.

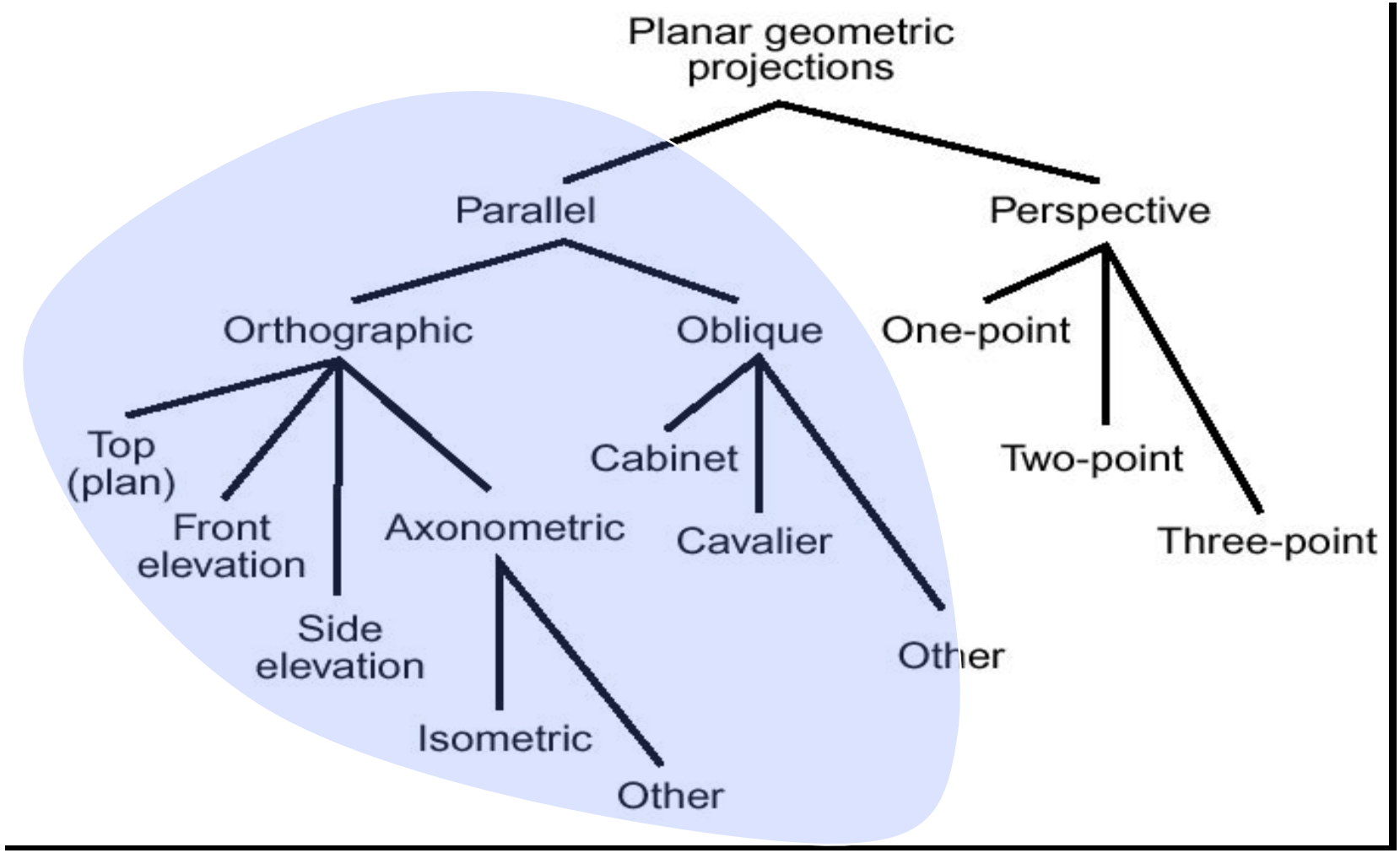
Oblique :

- direction of projection != normal to the projection plane.

Oblique vs. Orthographic



Taxonomy of Projection



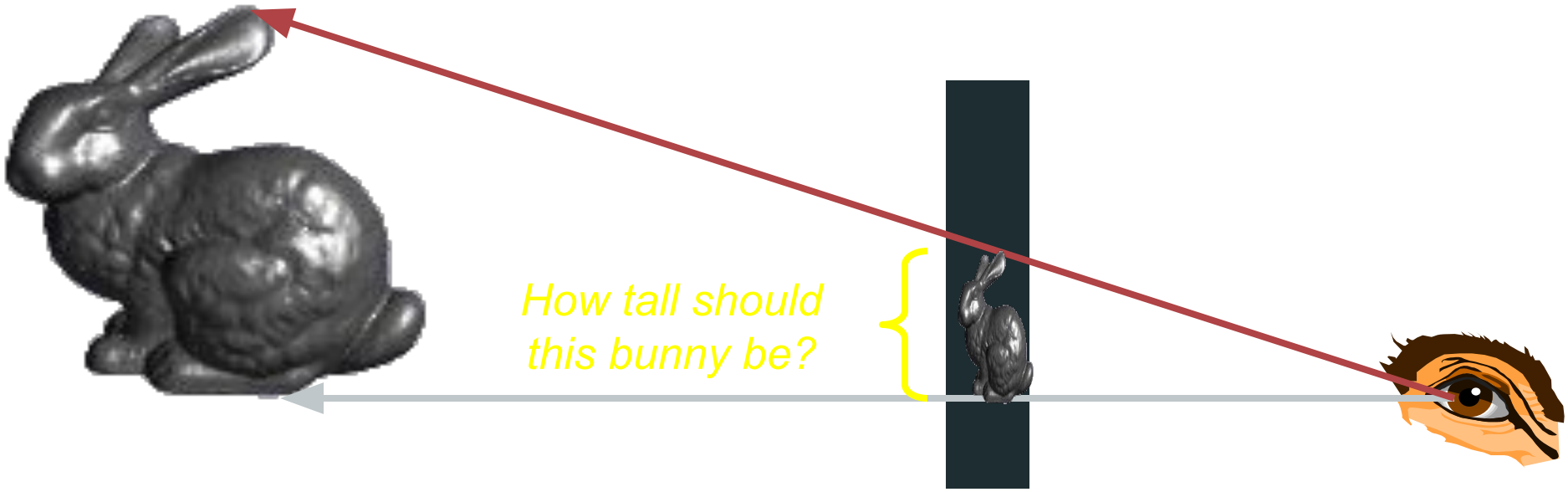
Perspective Projection

In the real world, objects exhibit perspective foreshortening: distant objects appear smaller

The basic situation:

Perspective Projection

When we do 3-D graphics, we think of the screen as a 2-D window onto the 3-D world:



Perspective Projection

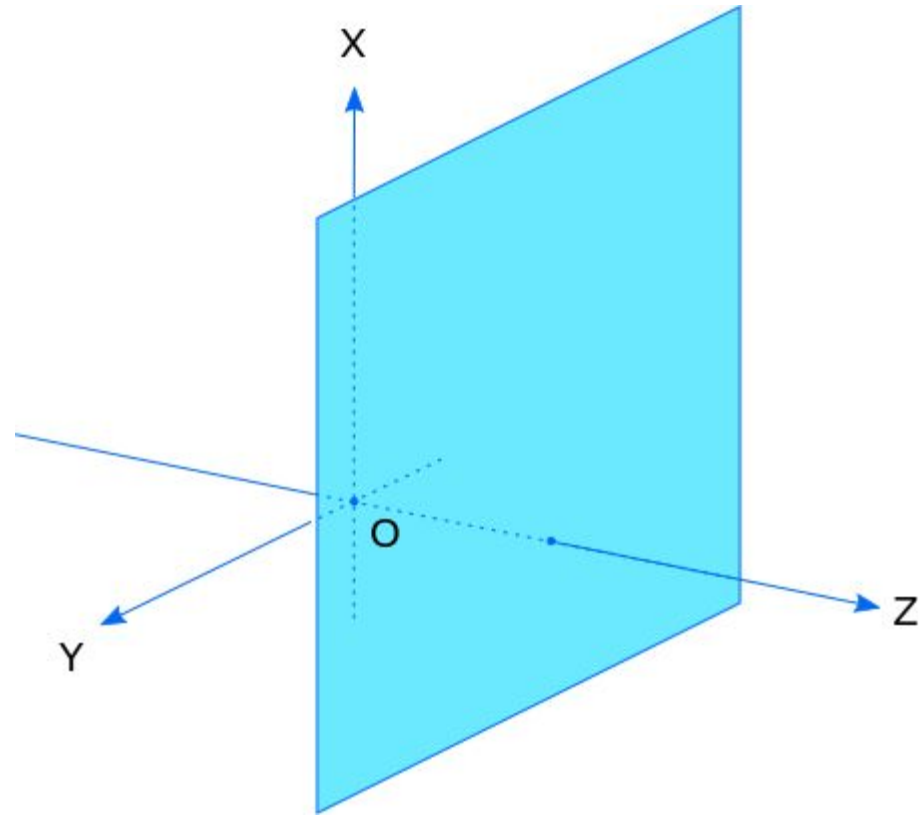
Simple Case:

COP: Origin

View Plane:

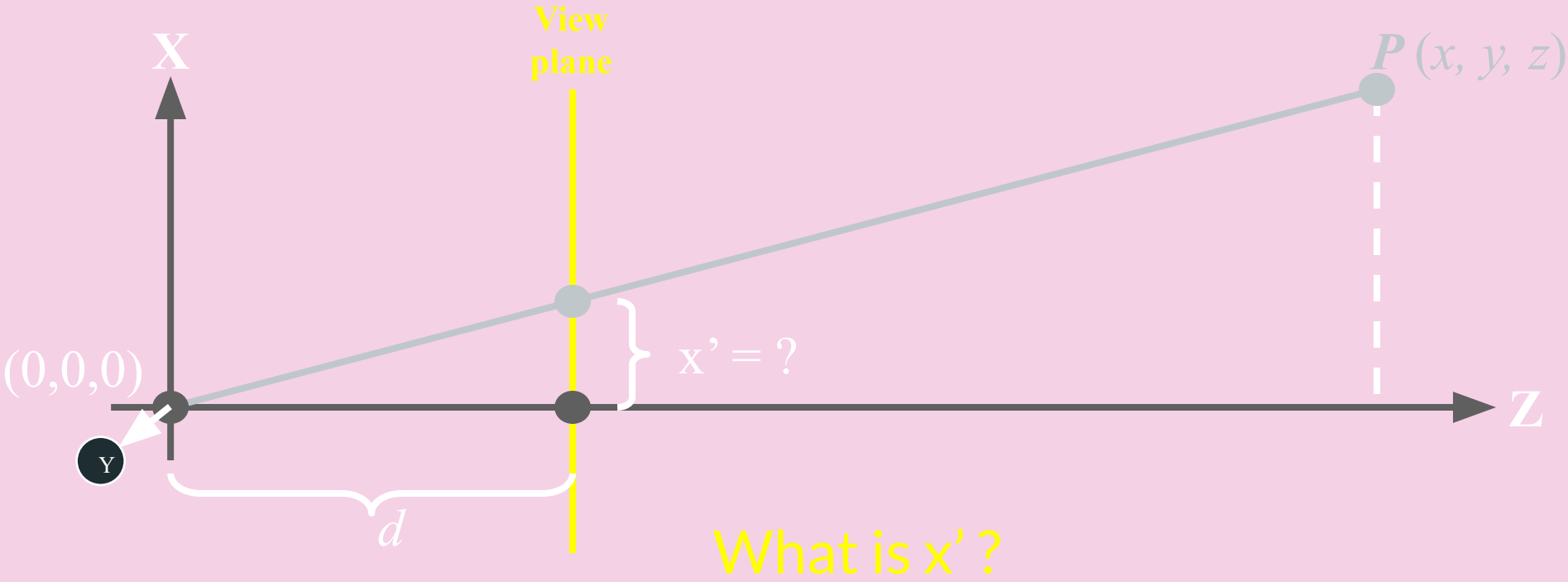
Parallel to XY plane

d distance afar from the origin



Perspective Projection

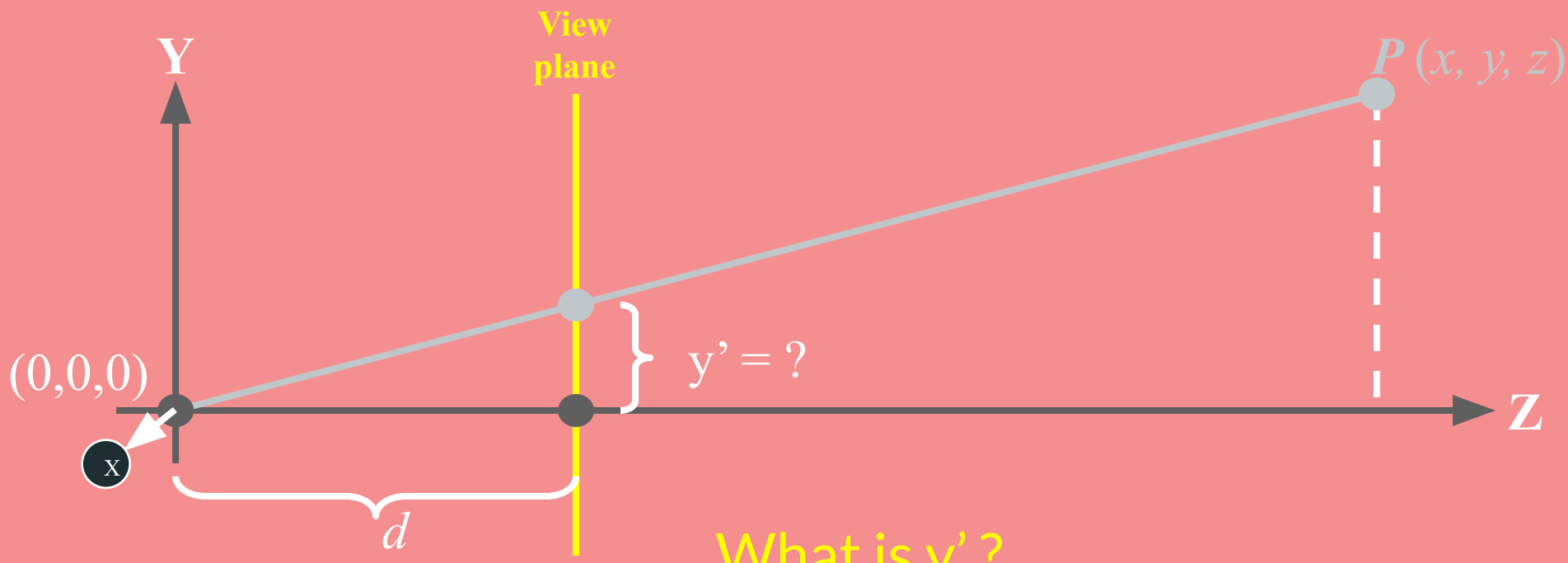
The geometry of the situation is that of similar triangles.
View from Y-axis:



What is x' ?

Perspective Projection

The geometry of the situation is that of similar triangles.
View from X-axis:



What is y' ?

Perspective Projection

Desired result for a point $[x, y, z, 1]^T$ projected onto the view plane:

$$\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z}$$

$$x' = \frac{d \cdot x}{z} = \frac{x}{z/d}, \quad y' = \frac{d \cdot y}{z} = \frac{y}{z/d}, \quad z = d$$

What could a matrix look like to do this?

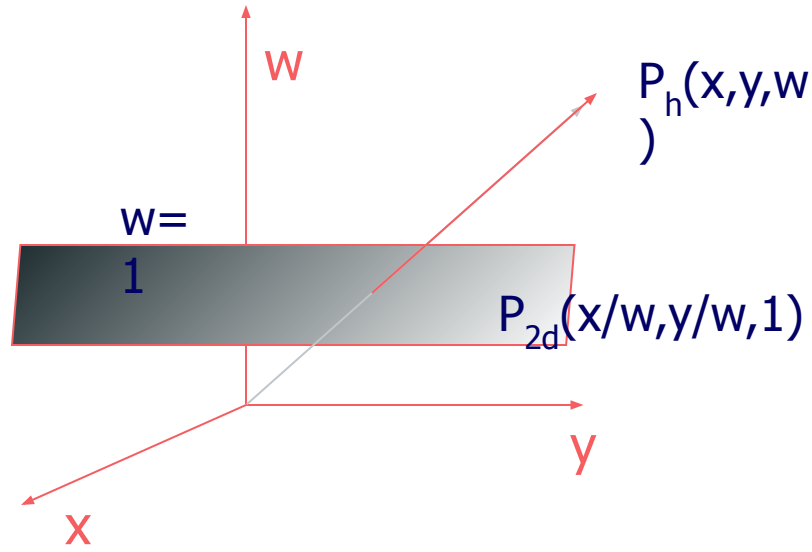
A Perspective Projection Matrix

Answer:

$$M_{perspective} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

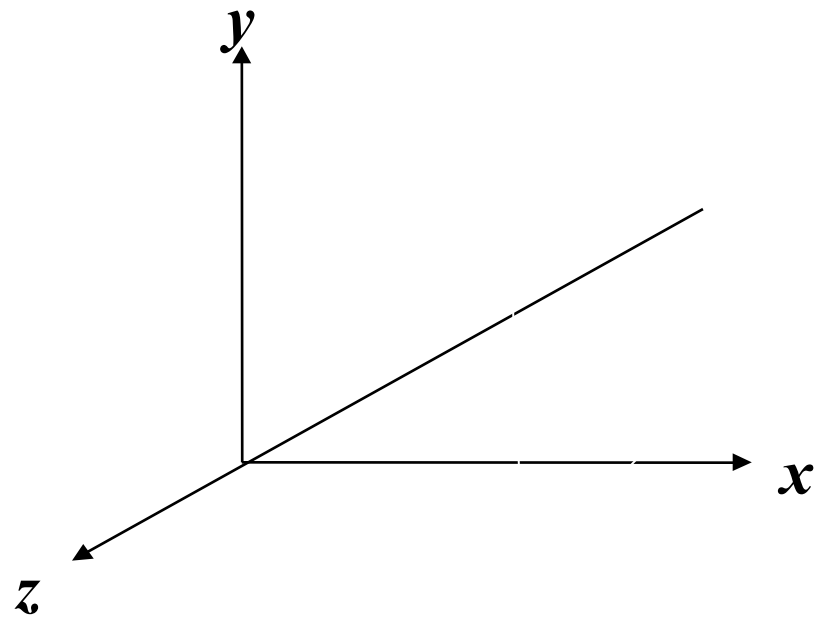
Homogeneous Co-ordinates

- P_{2d} is a projection of P_h onto the $w = 1$ plane
- So an infinite number of points correspond to : they constitute the whole line (tx, ty, tw)



Generalized Projection

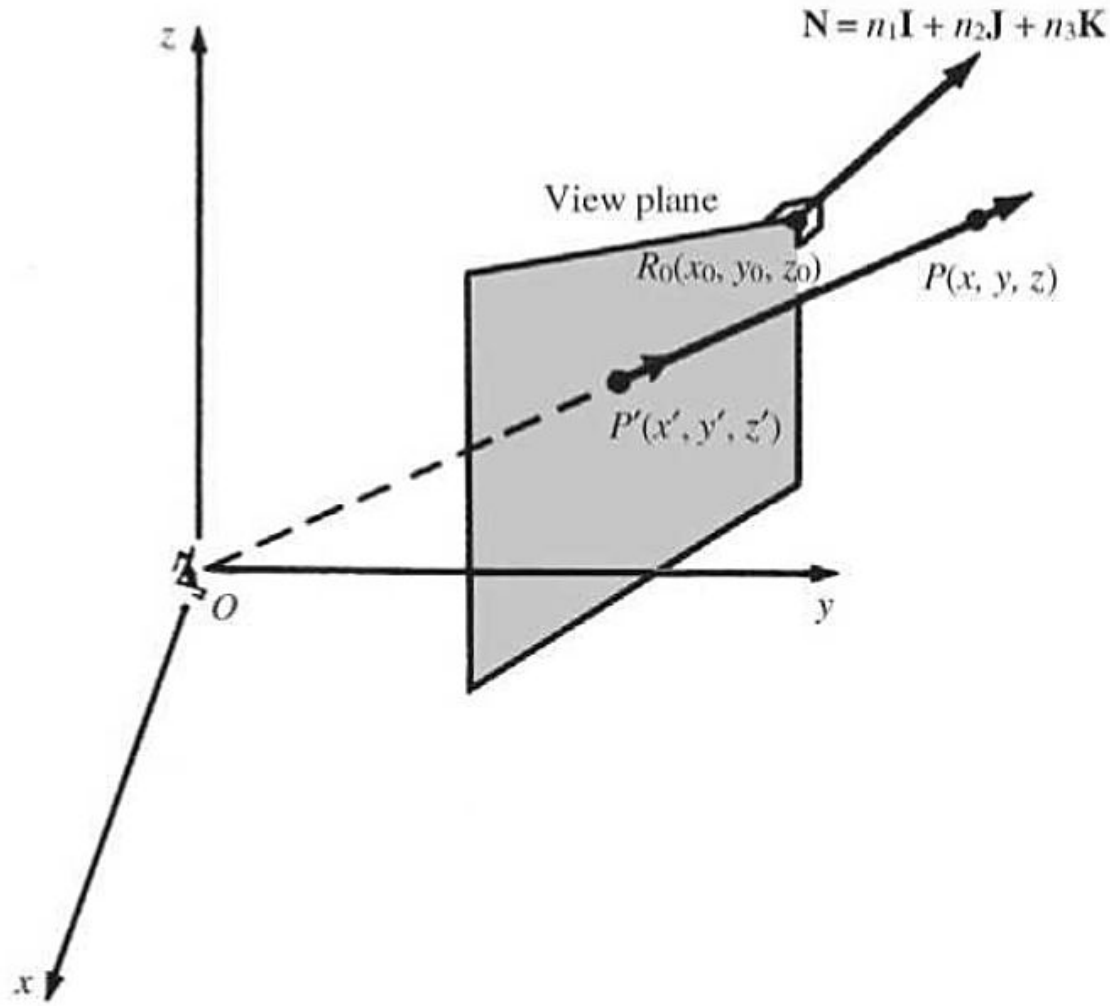
- Using the origin as the center of projection, derive the perspective transformation onto the plane passing through the point $R_0(x_0, y_0, z_0)$ and having the normal vector $N = n_1I + n_2J + n_3K$.



SOLUTION

Let $P(x, y, z)$ be projected onto $P'(x', y', z')$. From Fig. 7-16, the vectors $\overline{\mathbf{PO}}$ and $\overline{\mathbf{P'O}}$ have the same direction. Thus there is a number α so that $\overline{\mathbf{P'O}} = \alpha\overline{\mathbf{PO}}$. Comparing components, we have

$$x' = \alpha x \qquad y' = \alpha y \qquad z' = \alpha z$$



We now find the value of α. Since any point $P'(x', y', z')$ lying on the plane satisfies the equation (App. 2)

$$n_1x' + n_2y' + n_3z' = d_0$$

(where $d_0 = n_1x_0 + n_2y_0 + n_3z_0$), substitution of $x' = \alpha x$, $y' = \alpha y$, and $z' = \alpha z$ into this equation gives

$$\alpha = \frac{d_0}{n_1x + n_2y + n_3z}$$

This projection transformation cannot be represented as a 3×3 matrix transformation. However, by using the homogeneous coordinate representation for three-dimensional points, we can write the projection transformation as a 4×4 matrix:

$$Per_{\mathbf{N},R_0} = \begin{pmatrix} d_0 & 0 & 0 & 0 \\ 0 & d_0 & 0 & 0 \\ 0 & 0 & d_0 & 0 \\ n_1 & n_2 & n_3 & 0 \end{pmatrix}$$

Application of this matrix to the homogeneous representation $P(x, y, z, 1)$ of points P gives $P'(d_0x, d_0y, d_0z, n_1x + n_2y + n_3z)$, which is the homogeneous representation of $P'(x', y', z')$ found above.

Generalized Projection

$$P'O = \alpha\, PO$$

$$x' = \alpha x,\ y' = \alpha y,\ z' = \alpha z$$

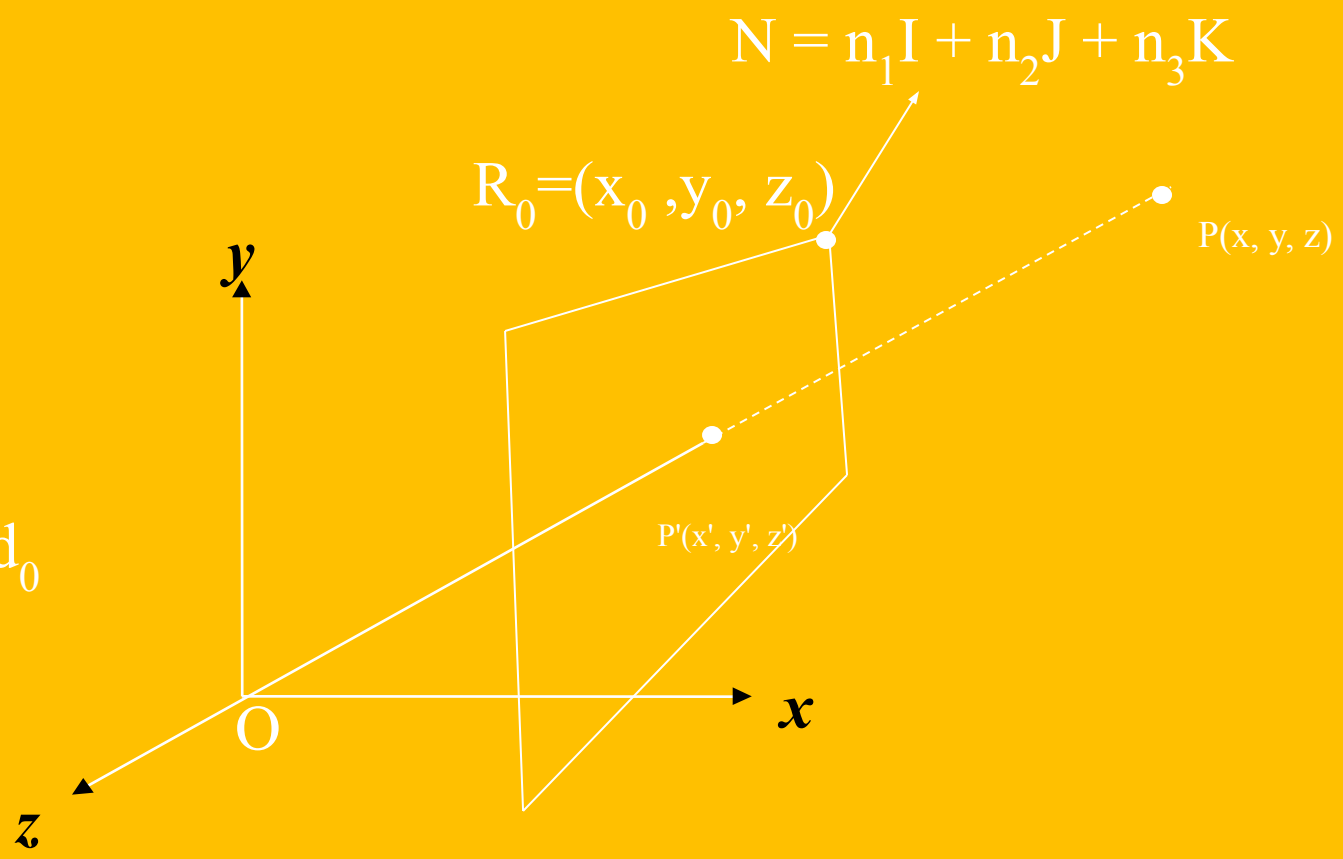
$$N.\ R_0P' = 0$$

$$n_1x' + n_2y' + n_3z'$$

$$= n_1x_0 + n_2y_0 + n_3z_0 = d_0$$

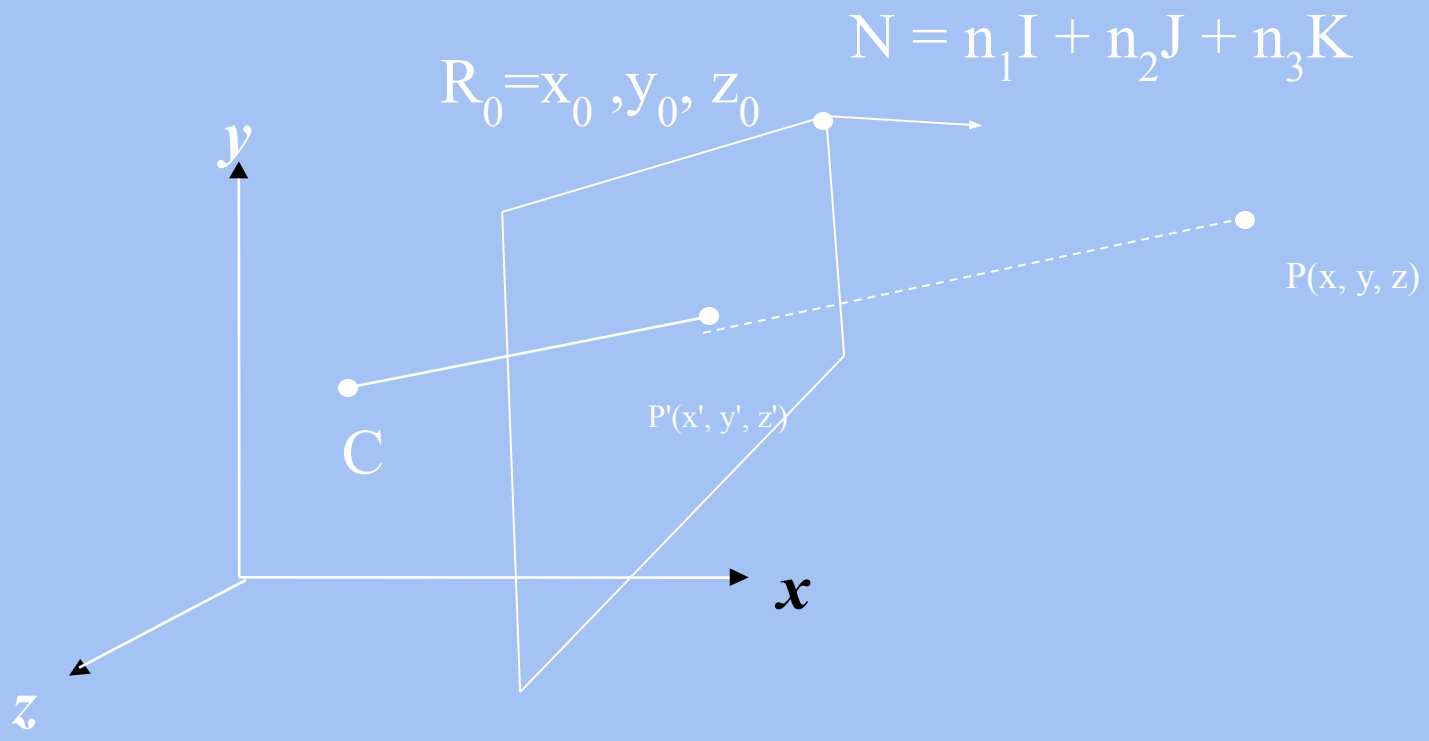
$$\alpha = \frac{d_0}{n_1x + n_2y + n_3z}$$

$$Per = \begin{pmatrix} d_0 & 0 & 0 & 0 \\ 0 & d_0 & 0 & 0 \\ 0 & 0 & d_0 & 0 \\ n_1 & n_2 & n_3 & 0 \end{pmatrix}$$



Generalized Projection

- Derive the general perspective transformation onto a plane with reference point R_0 and normal vector N and using C(a,b,c) as the center of projection.



Generalized Projection

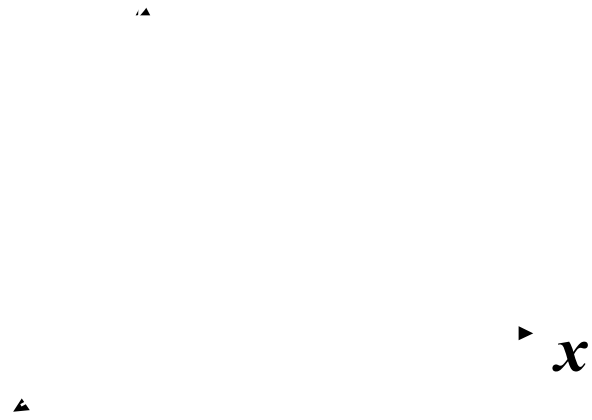
- Follow the steps –
  - Translate so that C lies at the origin
  - Perform projection as the previous problem
  - Translate back

$$Per_{\mathbf{N},R_0,C} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ n_1 & n_2 & n_3 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} d + an_1 & an_2 & an_3 & -ad_0 \\ bn_1 & d + bn_2 & bn_3 & -bd_0 \\ cn_1 & cn_2 & d + cn_3 & -cd_0 \\ n_1 & n_2 & n_3 & -d_1 \end{pmatrix}$$

Generalized Projection

$$\alpha = \frac{d}{n_1(x-a)+n_2(y-b)+n_3(z-c)}$$



Viewing in OpenGL

OpenGL has multiple matrix stacks - transformation functions right-multiply the top of the stack

Two most important stacks: `GL_MODELVIEW` and `GL_PROJECTION`

Points get multiplied by the modelview matrix first, and then the projection matrix

`GL_MODELVIEW`: Object->Camera

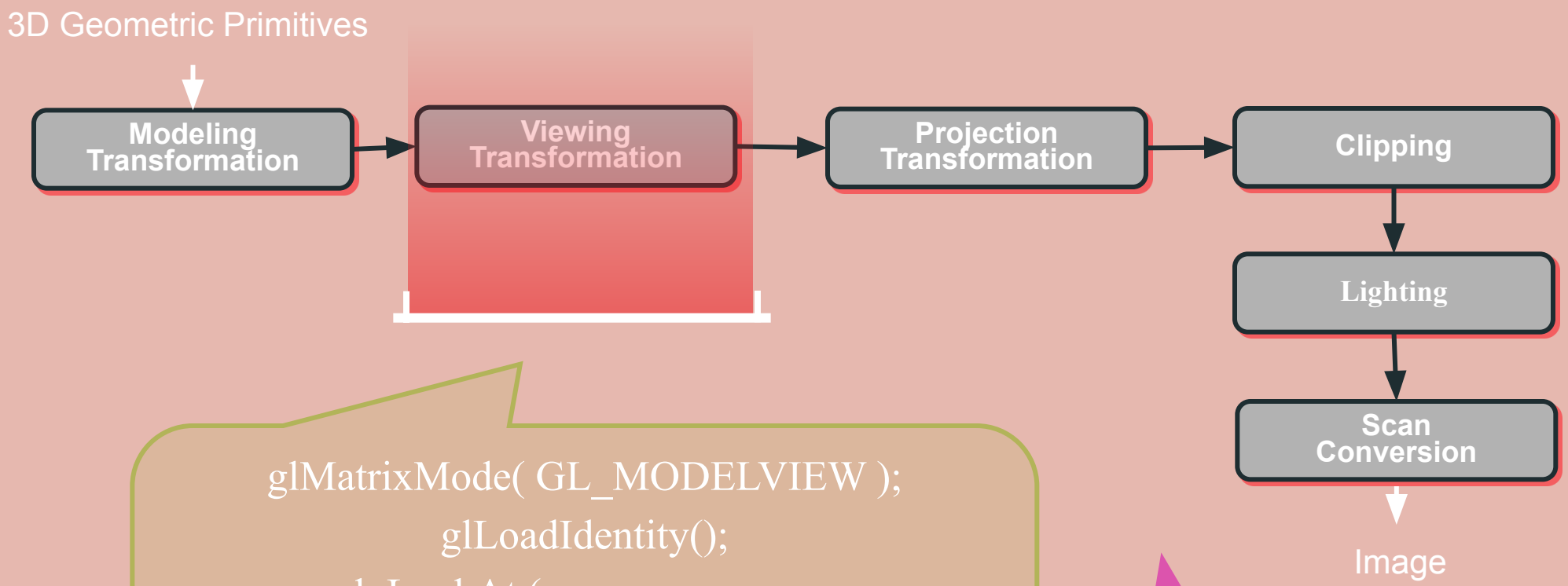
`GL_PROJECTION`: Camera->Screen

glViewport(0,0,w,h): Screen->Device

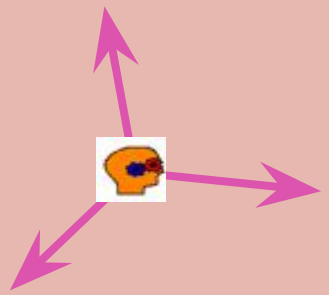
OpenGL Example

```
void SetUpViewing()
{
    // The viewport isn't a matrix, it's just state...
    glViewport( 0, 0, window_width, window_height );

    // Set up camera->screen transformation first
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluPerspective( 60, 1, 1, 1000 ); // fov, aspect, near, far


    // Set up the model->camera transformation
    glMatrixMode( GL_MODELVIEW );
    gluLookAt( 3, 3, 2,   // eye point
               0, 0, 0,   // look at point
               0, 0, 1 ); // up vector
    glRotatef( theta, 0, 0, 1 ); // rotate the model
    glScalef( zoom, zoom, zoom ); // scale the model
}
```

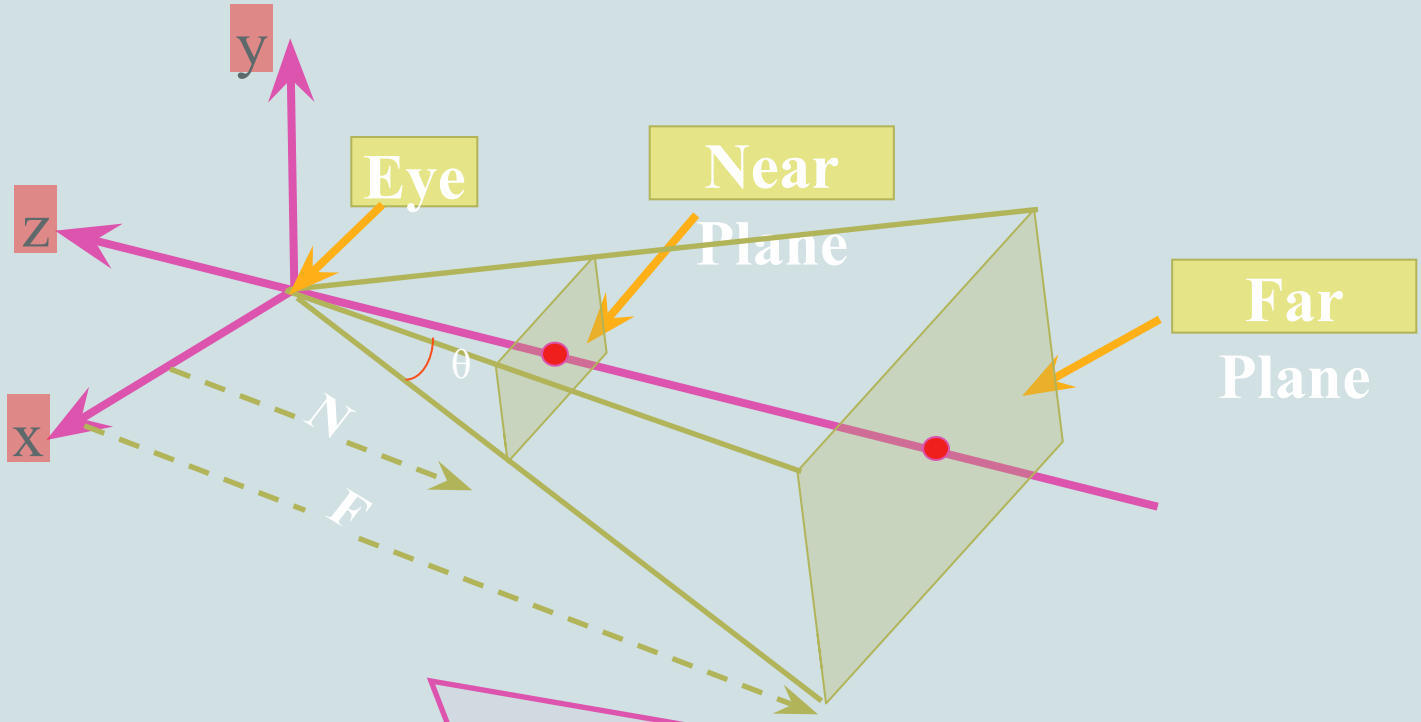
Graphics Pipeline Revisited




```
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
gluLookAt ( eyex, eyey, eyez,
            lookx, looky, lookz,
            upx, upy, upz );
```

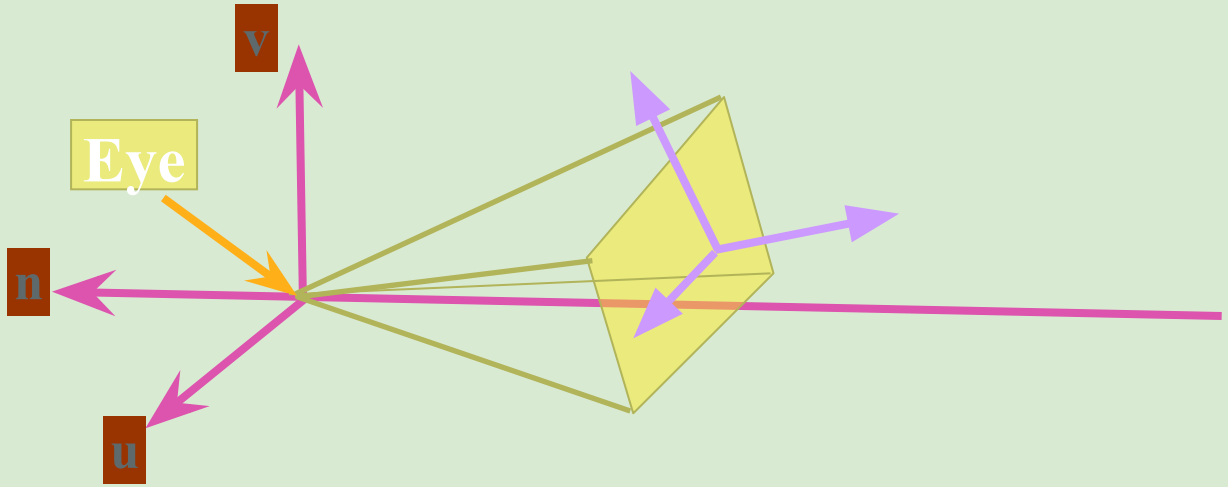


Default Camera Position

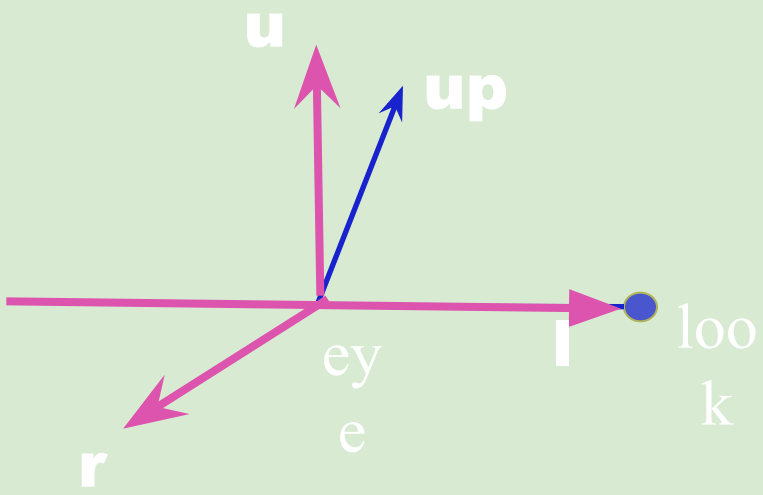



```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(viewAngle,
               aspectRatio, N, F);
```

General Camera Position



General Camera Position



l = look - eye

r = **l** X **up**

u = **r** X **l**

- To transform world coordinate into camera's coordinate, transform
  - *eye* into the origin
  - **r** into the vector **i**
  - **u** into the vector **j**
  - **-l** into the vector **k**

So what is the matrix for viewing transform?

Apply the following translation *T* to move the eye/camera to origin.

$$\begin{vmatrix} 1 & 0 & 0 & -eyeX \\ 0 & 1 & 0 & -eyeY \\ 0 & 0 & 1 & -eyeZ \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Apply the following rotation *R* such that the `l` aligns with the -Z axis, `r` with X axis, and `u` with Y axis. Remember that, the rows of the rotation matrix contain the unit vectors that align with the unit vectors along the principal axes after transformation.

$$\begin{vmatrix} r.x & r.y & r.z & 0 \\ u.x & u.y & u.z & 0 \\ -l.x & -l.y & -l.z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Thus the view transformation matrix $V=RT$.

A 3D Scene

Notice the presence of the camera, the projection plane, and the world coordinate axes

Viewing transformations define how to acquire the image on the projection plane

Viewing Transformations

Create a camera-centered view

Camera is at origin

Camera is looking along negative z-axis

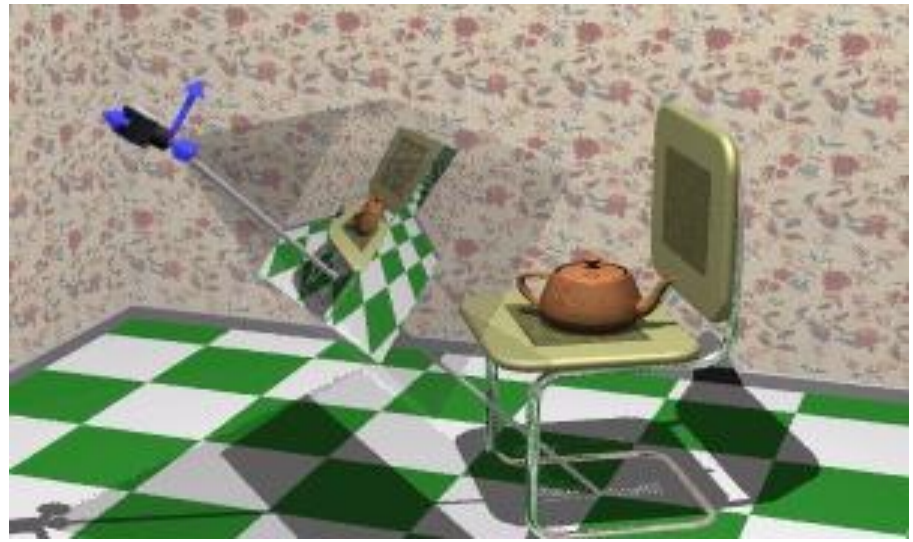
Camera's "up" is aligned with y-axis

2 Basic Steps

Align the two coordinate frames by rotation

2 Basic Steps

Translate to align origins



Creating Camera Coordinate Space

- Specify a point where the camera is located in world space, the eye point
- Specify a point in world space that we wish to become the center of view, the lookat point
- Specify a vector in world space that we wish to point up in camera image, the up vector
- Intuitive camera movement

Constructing Viewing Transformation, V

Create a vector from eye-point to lookat-point

$$\begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix} = \begin{bmatrix} lookat_x \\ lookat_y \\ lookat_z \end{bmatrix} - \begin{bmatrix} eye_x \\ eye_y \\ eye_z \end{bmatrix}$$

Normalize the v

$$\hat{l} = \frac{\bar{l}}{\sqrt{l_x^2 + l_y^2 + l_z^2}}$$

Desired rotation matrix should map th

vector to $[0, 0, -1]^T$ Why?

$$\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \mathbf{v}\hat{l}$$

Constructing Viewing Transformation, V

Construct another important vector from the cross product of the lookat-vector and the vup-vector

This vector, $\bar{r} = \bar{l} \times \overline{up}$ hould align with $[1, 0, 0]^T$ Why?

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{V} \frac{\bar{r}}{\sqrt{r_x^2 + r_y^2 + r_z^2}}$$

Constructing Viewing Transformation, V

One more vector to define....

$$\bar{u} = \bar{r} \times \bar{l}$$

This vector, when normalized, should align with $[0, 1, 0]^T$

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \mathbf{V} \frac{\bar{u}}{\sqrt{u_x^2 + u_y^2 + u_z^2}}$$

Now let's compose the results

Compositing Vectors to Form V

We know the three world axis vectors ((x, y, z)

We know the three camera axis vectors ((r, u, l)

Viewing transformation, V, must convert from world to camera coordinate systems

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{V}\begin{bmatrix} \hat{r} & \hat{u} & -\hat{l} \end{bmatrix}$$

Compositing Vectors to Form V

Remember

- Each camera axis vector is unit length.
- Each camera axis vector is perpendicular to others

Camera matrix is orthogonal and normalized

- Orthonormal

Therefore, $M^{-1} = M^{T}$

Compositing Vectors to Form V

Therefore, rotation component of viewing transformation is just transpose of computed vectors

$$\mathbf{V}_{rotate} = \begin{bmatrix} \hat{r} \\ \hat{u} \\ -\hat{l} \end{bmatrix}$$

Compositing Vectors to Form V

Translation component too

Multi

$$\begin{bmatrix} \hat{r} \\ \hat{u} \\ -\hat{l} \end{bmatrix} \begin{bmatrix} x - eye_x \\ y - eye_y \\ z - eye_z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

$$\begin{bmatrix} \hat{r} \\ \hat{u} \\ -\hat{l} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} \hat{r} \\ \hat{u} \\ -\hat{l} \end{bmatrix} \begin{bmatrix} eye_x \\ eye_y \\ eye_z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

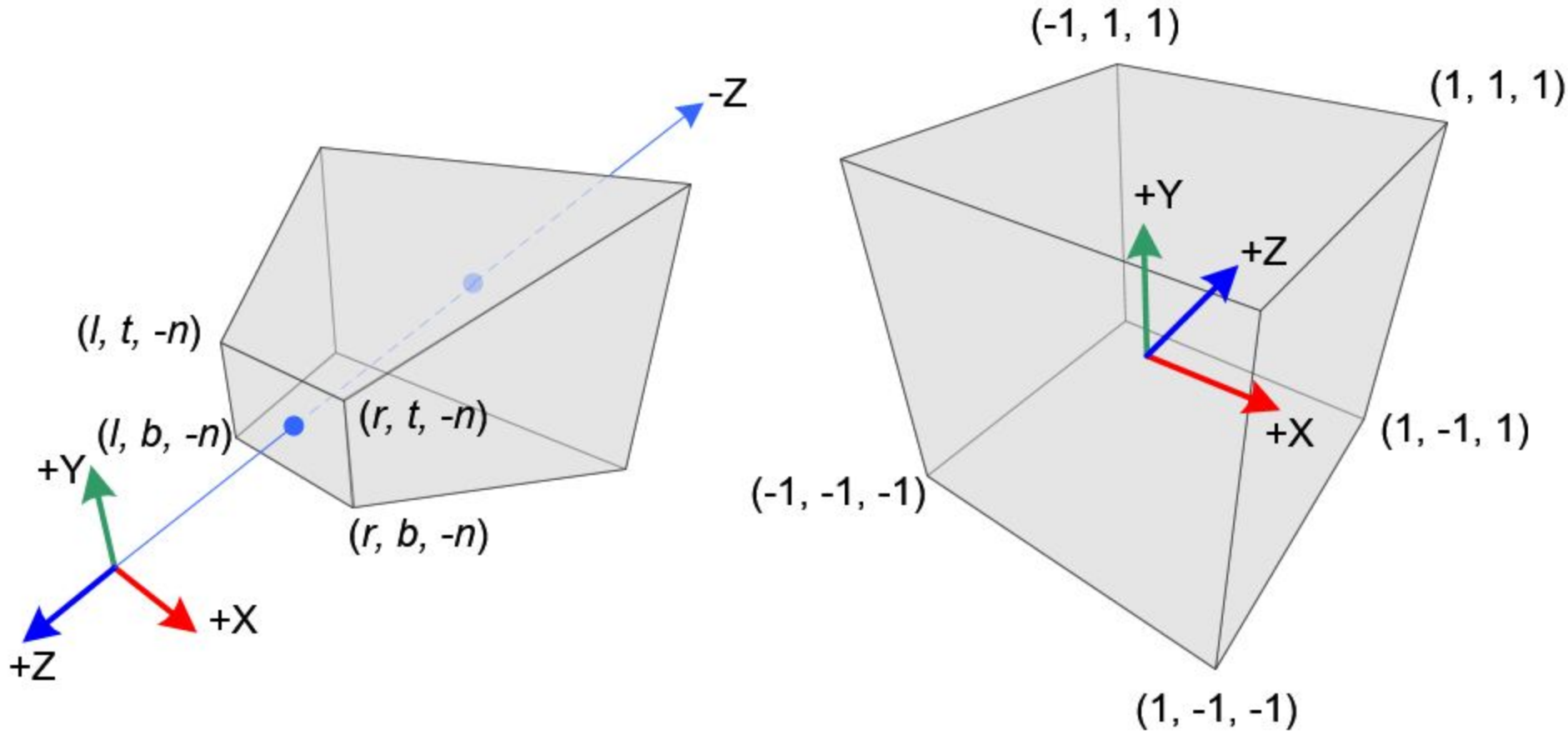
Final Viewing Transformation, V

To transform vertices, use this matrix:

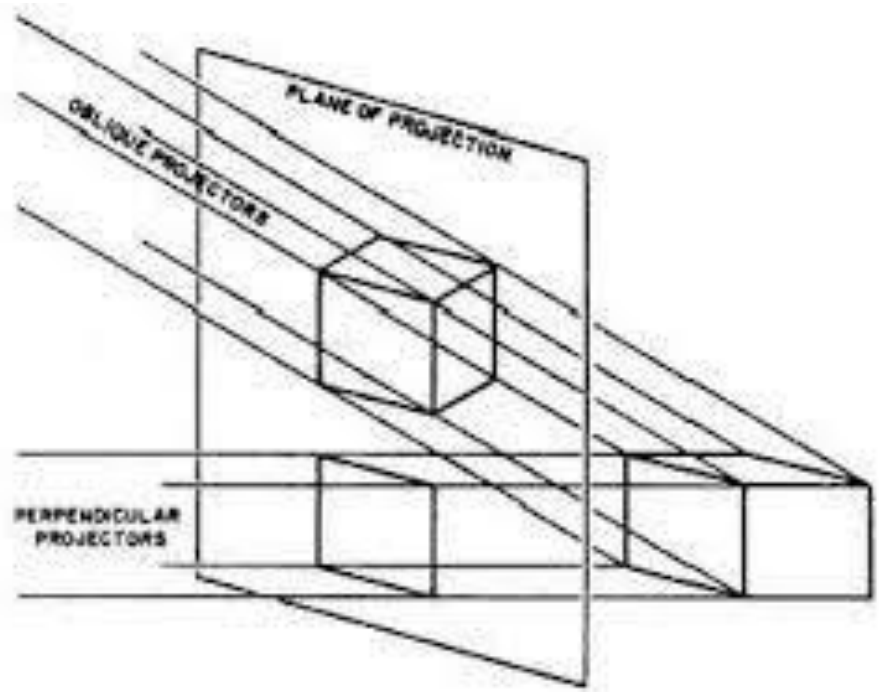
$$\begin{bmatrix} \hat{r} & -\hat{r}\cdot\overline{eye} \\ \hat{u} & -\hat{u}\cdot\overline{eye} \\ -\hat{l} & \hat{l}\cdot\overline{eye} \\ 0\ \ 0\ \ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

And you get this:

OpenGL Projection Transformation

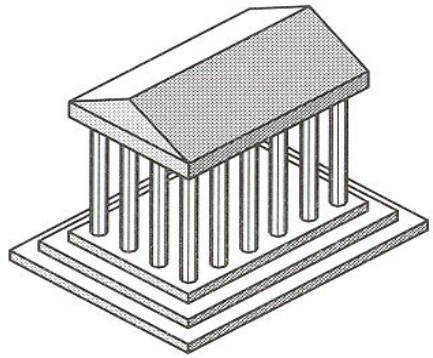


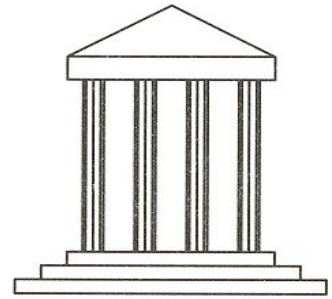
Oblique vs. Orthographic



Orthographic Projections

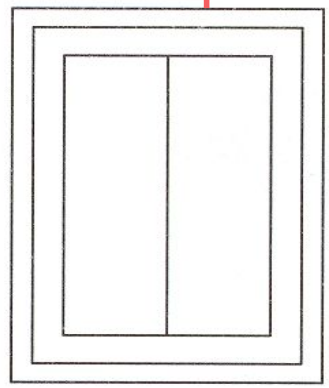
DOP perpendicular to view plane



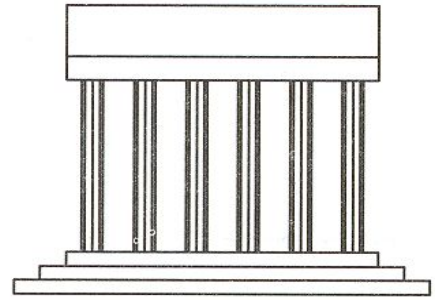


Top

Front



Side



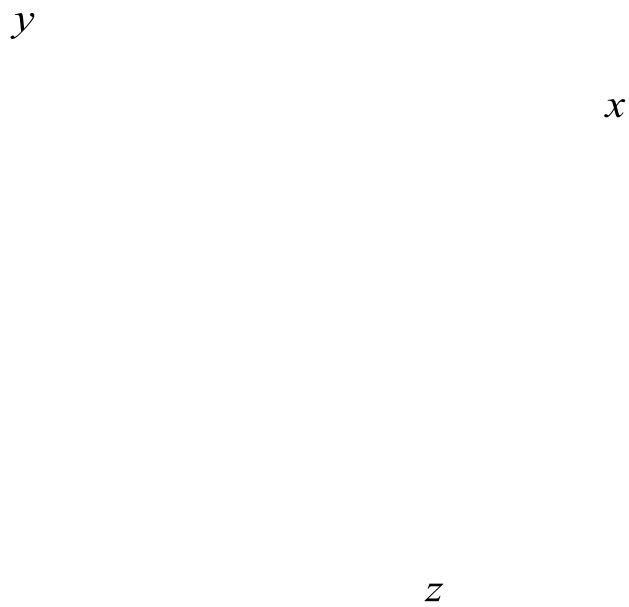
Axonometric Orthogonal Projection

(XM) Isometric Axonometric Orthogonal Projection:

Axonometric vs Perspective

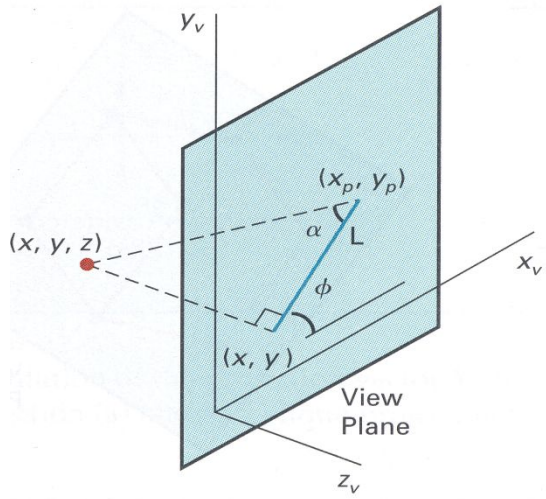
Axonometric projection shows several faces of an object at once like perspective projection.

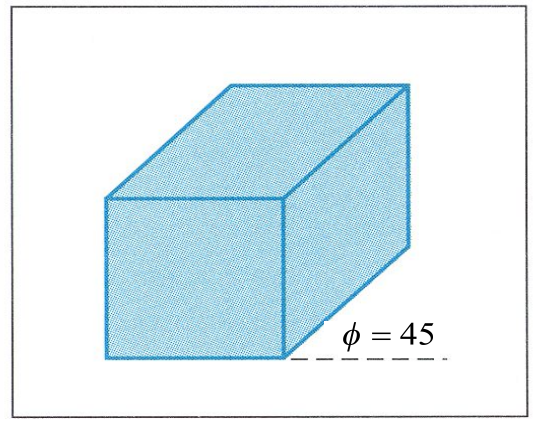
But the foreshortening is uniform rather than being related to the distance from the COP.

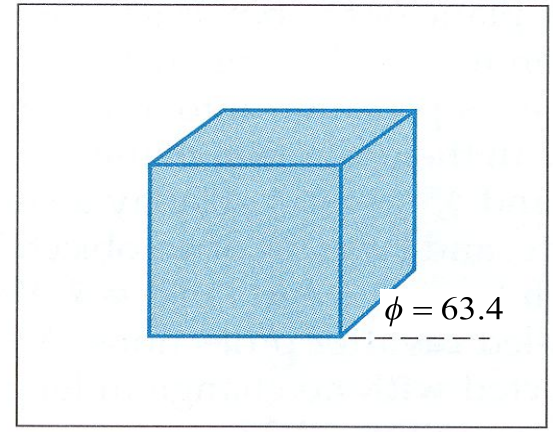


Oblique Projections

DOP not perpendicular to view plane







H&B

Oblique parallel projection

Cavalier, cabinet and orthogonal projections can alll be specified in terms of (α, β) or (λ, β) since

- $\tan(\alpha) = 1/\lambda$

Oblique parallel projection

Oblique parallel projection

y

(x_p, y_p)

λ

(x, y) α

x

$$\therefore x_p = x + \lambda \cos\alpha$$
$$y_p = y + \lambda \sin\alpha$$

$$\begin{pmatrix} x' \\ y' \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \lambda\cos\alpha & 0 \\ 0 & 1 & \lambda\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Ref.

- Chapter 7, Schaum's Outline of Computer Graphics (2nd Edition) by Zhigang Xiang, Roy A. Plastock