Computer Graphics (CSE2066)

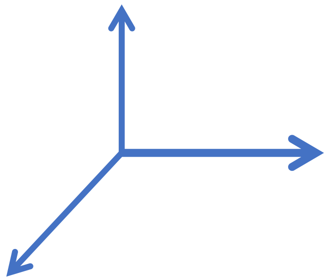
3D Geometric Transformations

Translation, Scaling, Rotation in computer Graphics

PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

3D Geometric Transformations:

- In Computer graphics, Transformation is a process of modifying and re-positioning the existing graphics.
- 3D Transformations take place in a three dimensional plane



- Transformations are helpful in changing the position, size, orientation, shape etc of the object.
- A three-dimensional position, expressed in **homogeneous coordinates, is represented as a four-element column vector**

1. Three-Dimensional Translation:

- In Computer graphics, 3D Translation is a process of **moving** an object from one position to another in a two dimensional plane.

- A position P = (x, y, z) in three-dimensional space is translated to a location P'= (x', y', z') by adding translation distances *tx*, *ty*, and *tz* to the Cartesian coordinates of P:

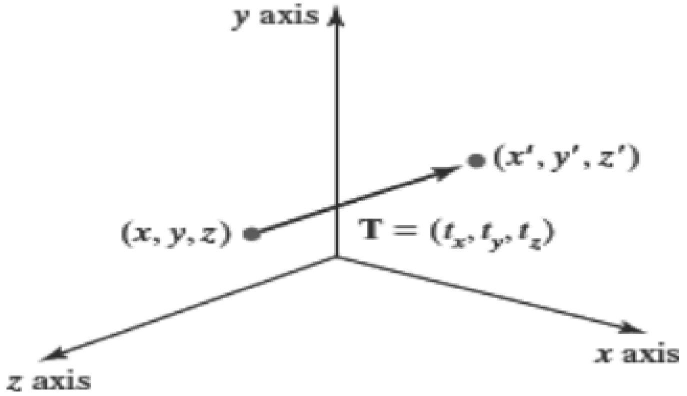
$$x' = x + t_x, \qquad y' = y + t_y, \qquad z' = z + t_z$$

- We can express these three-dimensional translation operations in matrix form

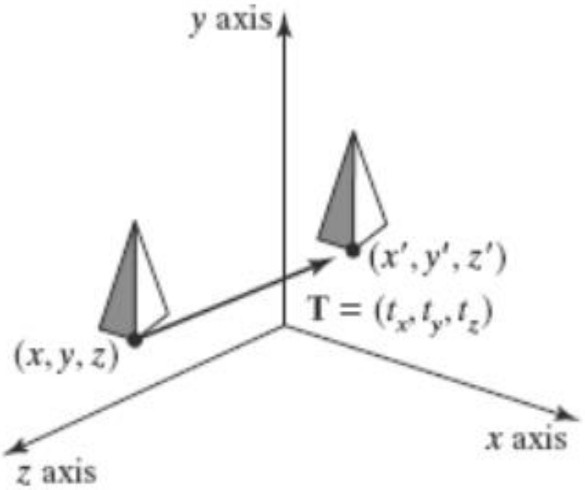
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Or $$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$

- Moving a coordinate position with translation vector T = (tx , ty , tz).



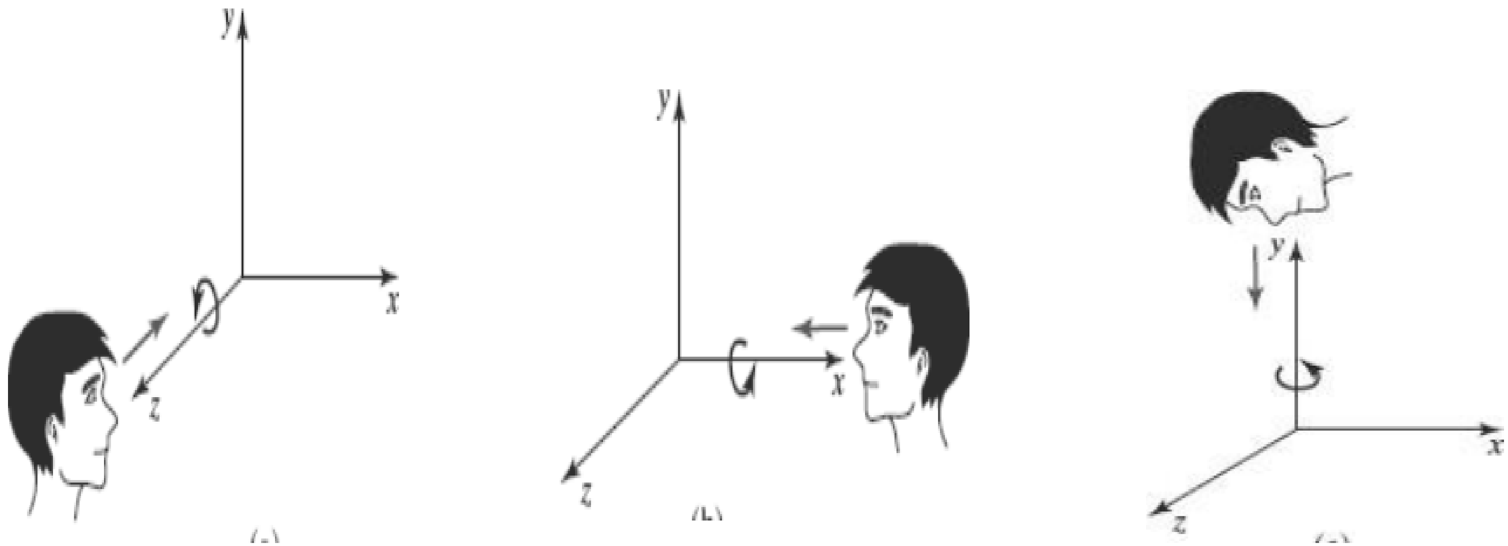
- Shifting the position of a three-dimensional object using translation vector **T**.



- **An inverse of a three-dimensional translation matrix is obtained by <u>negating</u> the translation distances *tx*, *ty*, and *tz***

2. Three-Dimensional Rotation

- By convention, **positive rotation angles produce counterclockwise rotations** about a coordinate axis.
- Positive rotations about a coordinate axis are counterclockwise, when looking along the positive half of the axis toward the origin.



Three-Dimensional Coordinate-Axis Rotations

Along z axis:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

- In homogeneous-coordinate form, the three-dimensional z-axis rotation equations are

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Transformation equations for rotations about the other two coordinate axes can be obtained with a cyclic permutation of the coordinate parameters x, y, and z

$$x \rightarrow y \rightarrow z \rightarrow x$$

Along x axis

$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$
$$x' = x$$

Along y axis

$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$

- **An inverse three-dimensional rotation matrix is obtained in the same by replacing θ with $-\theta$ *(Negating)*.**

Matrix for representing three-dimensional rotations about the Z axis

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

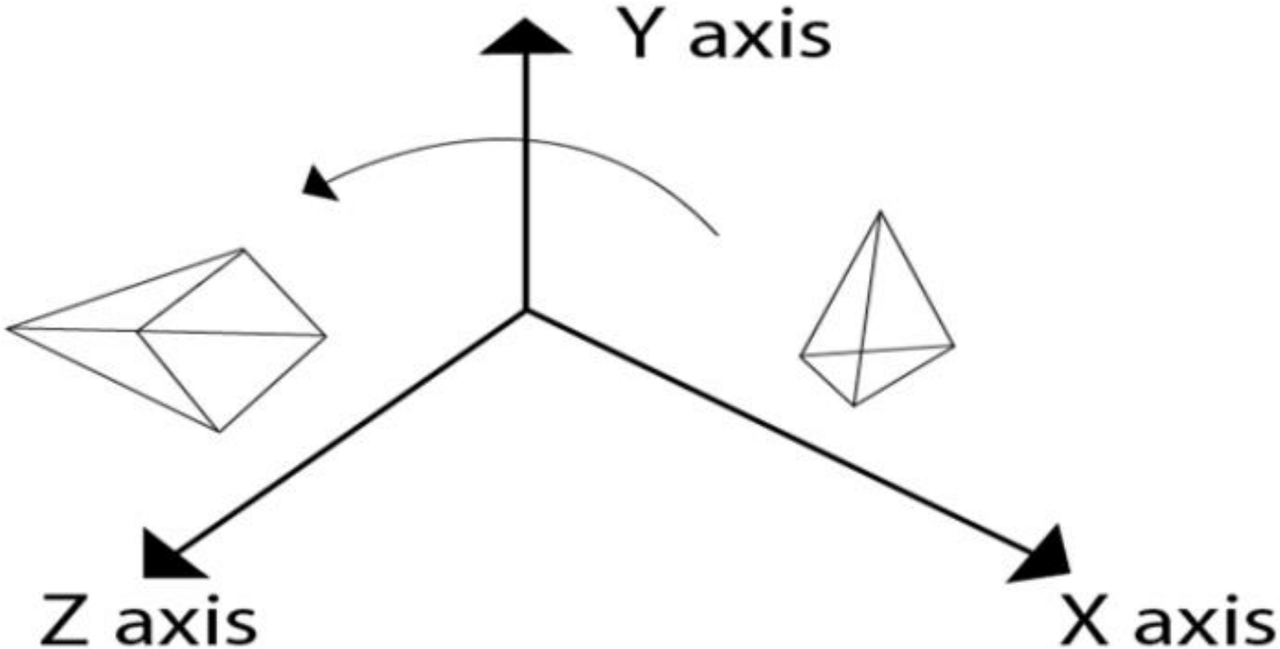
Matrix for representing three-dimensional rotations about the X axis

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

Matrix for representing three-dimensional rotations about the Y axis

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

Following figure show the original position of object and position of object after rotation about the x-axis



3. Three-Dimensional Scaling

- The matrix expression for the three-dimensional scaling transformation of a position **P** = *(x, y, z) is given by*

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- The three-dimensional scaling transformation for a point position can be represented as

$$\mathrm{P}' = \mathrm{S} \cdot \mathrm{P}$$

where scaling parameters s_x, s_y, and s_z are assigned any positive values.

- Explicit expressions for the scaling transformation relative to the origin are

$$x' = x \cdot s_x, \qquad y' = y \cdot s_y, \qquad z' = z \cdot s_z$$

Composite 3D Translations

- If two successive translation vectors (t_{1x}, t_{1y}, t_{1z}), (t_{2x}, t_{2y}, t_{2z}) are applied to a three dimensional coordinate position **P**, the final transformed location P′ is calculated as

$$\mathbf{P}' = \mathbf{T}(\mathbf{t_{2x}}, \mathbf{t_{2y}}, \mathbf{t_{2z}}).\{\mathbf{T}(\mathbf{t_{1x}}, \mathbf{t_{1y}}, \mathbf{t_{1z}}).\mathbf{P}\}$$
$$= \{\mathbf{T}(\mathbf{t_{2x}}, \mathbf{t_{2y}}, \mathbf{t_{2z}}).\mathbf{T}(\mathbf{t_{1x}}, \mathbf{t_{1y}}, \mathbf{t_{1z}})\}.\mathbf{P}$$

- Also, the composite transformation matrix for this sequence of translations is

$$\begin{bmatrix} 1 & 0 & 0 & t_{2x} \\ 0 & 1 & 0 & t_{2y} \\ 0 & 0 & 1 & t_{2z} \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & t_{1x} \\ 0 & 1 & 0 & t_{1y} \\ 0 & 0 & 1 & t_{1z} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & 0 & t_{1y} + t_{2y} \\ 0 & 0 & 1 & t_{1z} + t_{2z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T(t_{2x}, t_{2y}, t_{2z}) . T(t_{1x}, t_{1y}, t_{1z}) = T(t_{1x}+t_{2x}, t_{1y}+t_{2y}, t_{1z}+t_{2z})}$$

- **This demonstrates 2 successive translation matrix are additive.**

Composite 3D Rotation

- Two successive rotations applied to a point **P** produce the transformed position

$$\mathbf{P}' = \mathbf{R}(\theta_2) \cdot \{\mathbf{R}(\theta_1) \cdot \mathbf{P}\}$$
$$= \{\mathbf{R}(\theta_2) \cdot \mathbf{R}(\theta_1)\} \cdot \mathbf{P}$$

- By multiplying the two rotation matrices, we can verify that two successive rotations are additive:

$$\mathbf{R}(\vartheta 2) \cdot \mathbf{R}(\vartheta 1) = \mathbf{R}(\vartheta 1 + \vartheta 2)$$

- So that the final rotated coordinates of a point can be calculated with the composite rotation matrix

$$\mathbf{P}' = \mathbf{R}(\vartheta 1 + \vartheta 2) \cdot \mathbf{P}$$

- **This demonstrates 2 successive rotation matrix are additive.**

Composite 3D Scaling

- Concatenating transformation matrices for two successive scaling operations in three dimensions produces the following composite scaling

$$\begin{bmatrix} s_{2x} & 0 & 0 & 0 \\ 0 & s_{2y} & 0 & 0 \\ 0 & 0 & s_{2z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{1x} & 0 & 0 & 0 \\ 0 & s_{1y} & 0 & 0 \\ 0 & 0 & s_{1z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{2x}\cdot s_{1x} & 0 & 0 & 0 \\ 0 & s_{2y}\cdot s_{1y} & 0 & 0 \\ 0 & 0 & s_{2z}\cdot s_{1z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S(s_{2x}, s_{2y}, s_{2z}) \cdot S'(s_{1x}, s_{1y}, s_{1z}) = S(s_{1x}\cdot s_{2x}, s_{1y}\cdot s_{2y}, s_{1z}\cdot s_{2z})$$

- **This demonstrates two successive scaling matrix are multiplicative.**

OpenGL 3D Geometric Transformation Functions

4 x 4 Translation matrix is constructed with the following routine:

glTranslate*(tx, ty, tz);

- Translation parameters **tx**, **ty**, and **tz** can be assigned any real-number values, and the single suffix code to be affixed to this function is either **f** (float) or **d** (double).
- example: **glTranslatef (25.0, -10.0, 10.0);**

4 x 4 Rotation matrix is generated with

glRotate*(theta, vx, vy, vz);

- where the vector **v** = (**vx**, **vy**, **vz**) can have any floating-point values for its components defines the orientation for a rotation axis that passes through the coordinate origin.
- The suffix code can be either **f** or **d**, and parameter **theta** is to be assigned a rotation angle in degree
- For example, the statement: **glRotatef (90.0, 0.0, 0.0, 1.0);**

4 x 4 Scaling matrix with respect to the coordinate origin with the following routine:

glScale* (sx, sy, sz);

- The suffix code is again either **f** or **d**, and the scaling parameters can be assigned any real-number values.
- Scaling in a three-dimensional system involves changes in the *x*, *y* & *z* dimensions, so a typical three -dimensional scaling operation has a *z* scaling factor of 1.0
- **Example: glScalef (2.0, -3.0, 1.0);**

Basics of 3D viewing and Clipping:

Basics of 3D viewing

Three-Dimensional Viewing

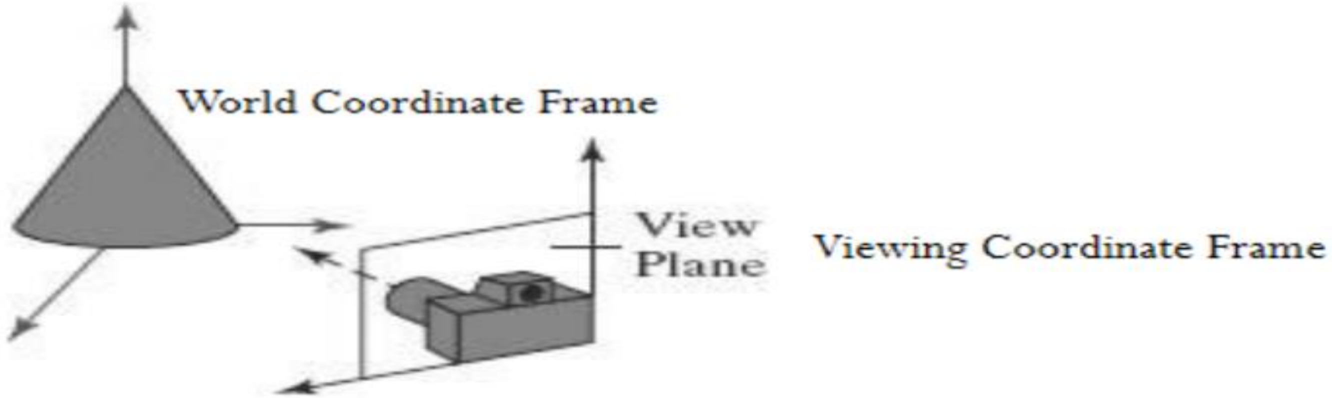
Overview of Three-Dimensional Viewing Concepts

- The 3D viewing process is inherently more complex than the 2D viewing process.
- In 2D viewing, we have 2D window and 2D viewport and objects in the world coordinates are clipped against the window and are then transformed into the viewport for display.
- The complexity added in the three dimensional viewing is because of the added dimension and the fact that even though objects are 3D. The display devices are only 2D
- The mismatch between 3D objects and 2D displays is compensated by introducing projections. The projections transform 3D objects into a 2D projection plane.

Viewing a Three-Dimensional Scene

- To obtain a display of a three-dimensional world-coordinate scene, we first set up a coordinate reference for the viewing, or "camera," parameters.

- This coordinate reference defines the position and orientation for a *view plane* (or *projection plane*) that corresponds to a camera film plane as shown in below figure.

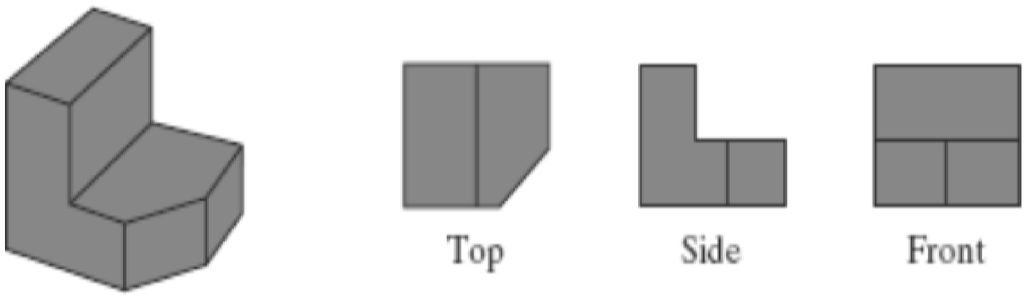


Coordinate reference for obtaining a selected view of a three-dimensional scene.

- We can generate a view of an object on the output device in wireframe (outline) form, or we can apply lighting and surface-rendering techniques to obtain a realistic shading of the visible surfaces Projections

Two methods:

1. One method for getting the description of a solid object onto a view plane is to project points on the object surface along parallel lines. This technique, called *parallel projection*



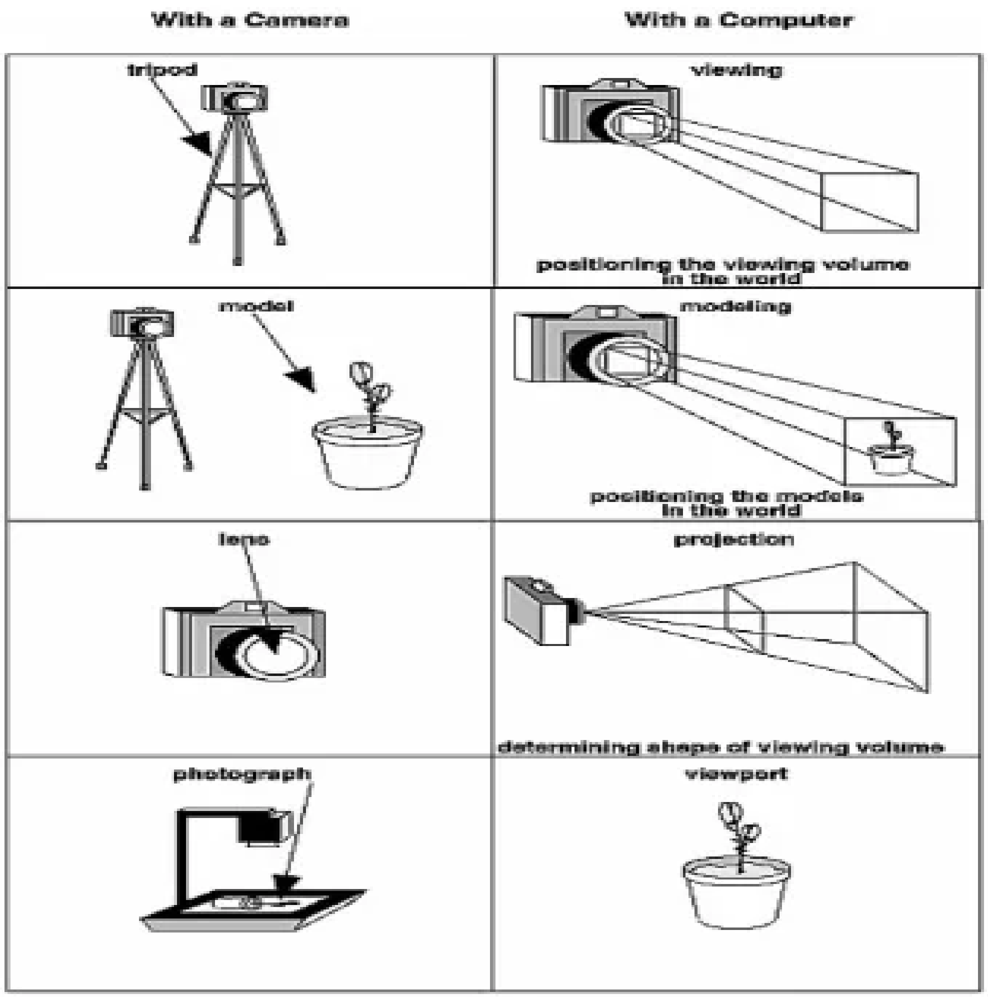
Three parallel-projection views of an object, showing relative proportions from different viewing positions

2. Another method for generating a view of a three-dimensional scene is to project points to the view plane along converging paths. This process, called a *perspective projection,* causes objects farther from the viewing position to be displayed smaller than objects of the same size that are nearer to the viewing position.

The Three-Dimensional Viewing Pipeline

3D viewing:Camera analogy

- Choose the position of the camera and pointing it at the scene (viewing transformation).
- Arranging the scene to be photographed into the desired composition (modeling transformation).
- Choosing a camera lens or adjusting the zoom (projection transformation).
- Determining how large you want the final photograph to be (viewport transformation).



PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013

Pipeline Architecture:

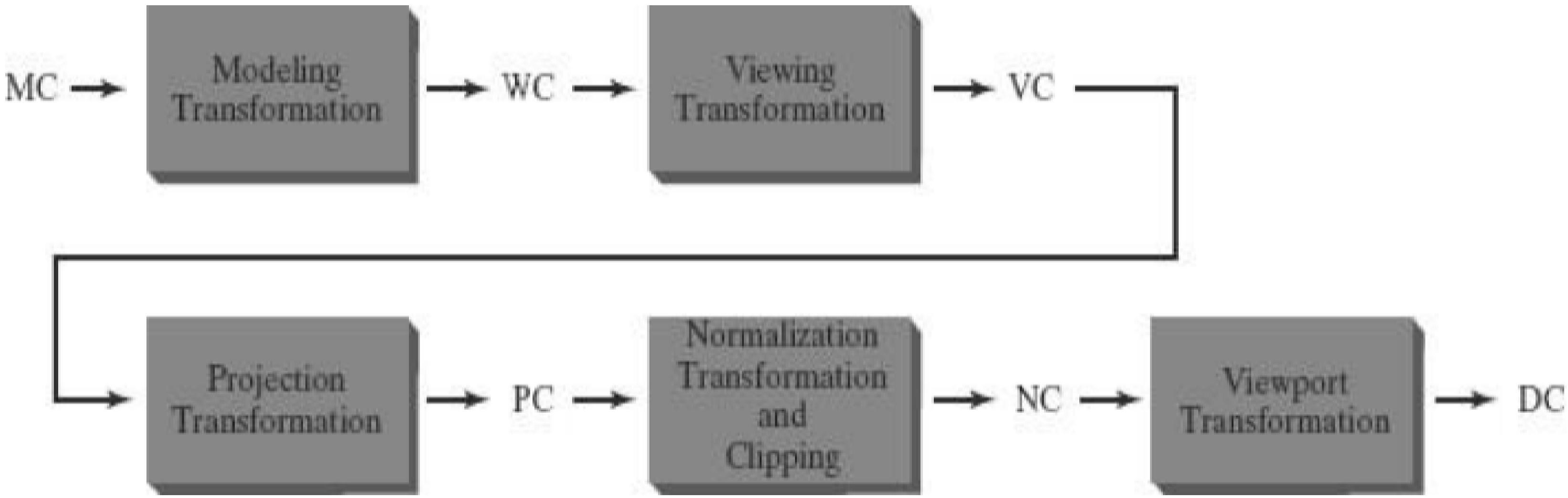


Figure above shows the general processing steps for creating and transforming a three-dimensional scene to device coordinates.

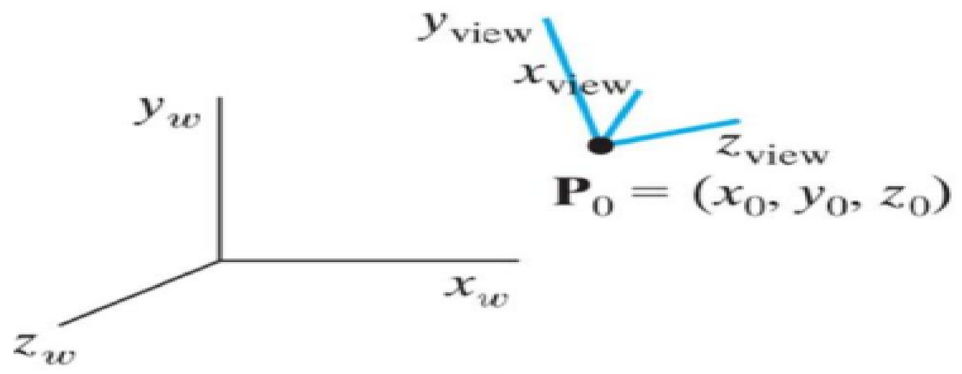
- Once the scene has been modeled in world coordinates, a viewing-coordinate system is selected and the description of the scene is converted to viewing coordinates
- A two-dimensional clipping window, corresponding to a selected camera lens, is defined on the projection plane, and a three-dimensional clipping region is established. This clipping region is called the view volume.
- Projection operations are performed to convert the viewing-coordinate description of the scene to coordinate positions on the projection plane.

- Objects are mapped to normalized coordinates, and all parts of the scene outside the view volume are clipped off. The clipping operations can be applied after all device-independent coordinate transformations.
- Assume that the viewport is to be specified in device coordinates and that normalized coordinates are transferred to viewport coordinates, following the clipping operations.
- The final step is to map viewport coordinates to device coordinates within a selected display window

Three-Dimensional Viewing-Coordinate Parameters

- *Establish a 3D viewing reference frame*

Right-handed



A right-handed viewing-coordinate system, with axes x $_{view}$, y $_{view}$, and z $_{view}$, relative to a right-handed world coordinate frame.

a. The **viewing origin**

Define the view point or viewing position (sometimes is referred to as the eye position or the camera position)

b. $\mathbf{y}_{view}$ -- view-up vector **V**

Defines y_{view} direction

- *Viewing direction and view plane*

c. $\mathbf{z}_{view}$: viewing direction

Along the z_{view} axis, often in the negative z_{view} direction

d. The view plane (also called projection plane) Perpendicular to $\mathbf{z}_{view}$ axis

The orientation of the view plane can be defined by a view-plane **normal vector N**

The different position of the view-plane along the $\mathbf{z}_{view}$ axis

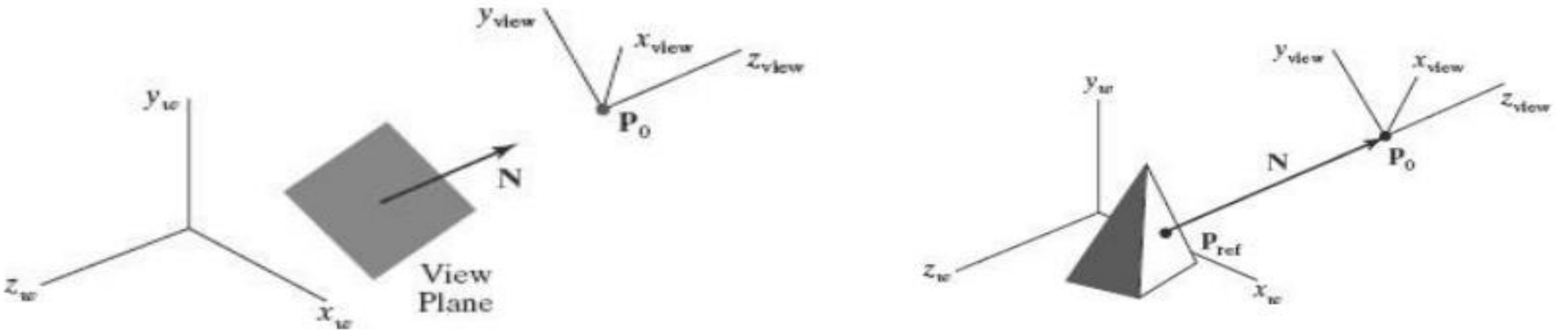


Fig: Orientation of the view plane and view-plane normal vector **N**

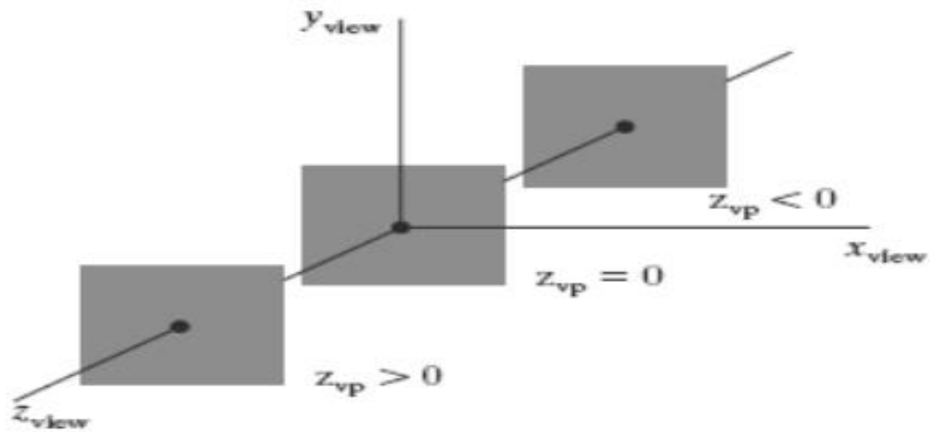
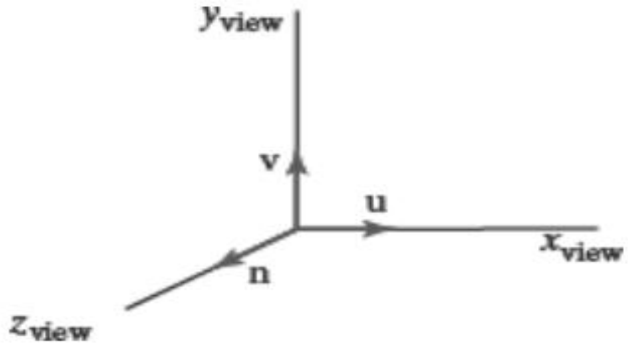


Fig: Three possible positions for the **view plane** along the z_{view} axis

- The coordinate system formed with these unit vectors is often described as a uvn viewing-coordinate reference frame



The uvn Viewing-Coordinate Reference Frame

- The **uvn** *Viewing-Coordinate Reference Frame* (*Viewing Coordinate System*)

  Direction of $\mathbf{z_{view}}$ axis: the view-plane normal vector **N**;

  Direction of $\mathbf{y_{view}}$ axis: the view-up vector **V**;

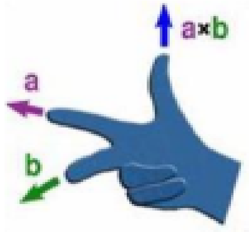
  Direction of $\mathbf{x_{view}}$ axis: taking the vector cross product of **V** and **N** to get **U**.

- Following these procedures, we obtain the following set of unit axis vectors for a right- handed viewing coordinate system.

$$\mathbf{n} = \frac{\mathbf{N}}{|\mathbf{N}|} = (n_x, n_y, n_z)$$

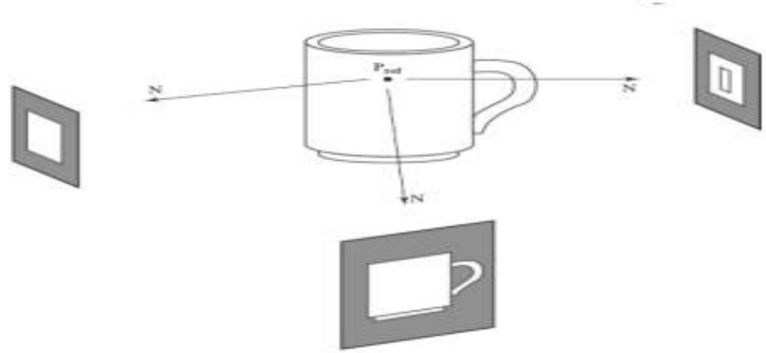
$$\mathbf{u} = \frac{\mathbf{V} \times \mathbf{n}}{|\mathbf{V} \times \mathbf{n}|} = (u_x, u_y, u_z)$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u} = (v_x, v_y, v_z)$$

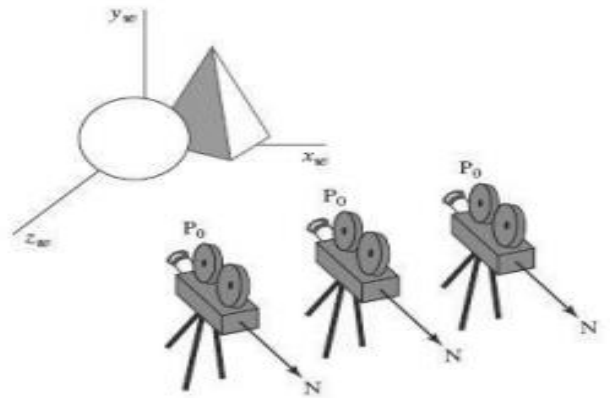


Generating Three-Dimensional Viewing Effects

To obtain a series of view of a scene, we can keep the view reference point fixed and change the direction of N



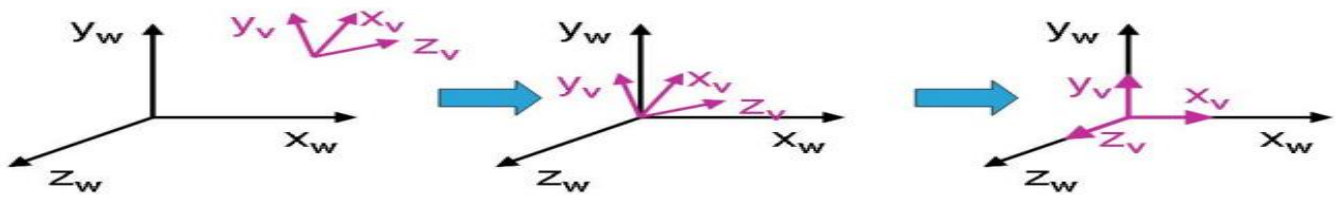
To simulate camera motion through a scene, we can keep N fixed and move the view reference point around.



Transformation from World to Viewing Coordinates

- Transformation viewing from world to viewing coordinates includes

  1. Translate the viewing-coordinate origin to the origin of the world coordinate system.
  2. Apply rotations to align the x_{view}, y_{view}, and z_{view} axes with the world x_w, y_w, and z_w axes, respectively.



- The viewing-coordinate origin is at world position $P_0 = (x_0, y_0, z_0)$. Therefore, the matrix for translating the viewing origin to the world origin is.

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- For the rotation transformation, we can use the unit vectors u, v, and n to form the composite rotation matrix that superimposes the viewing axes onto the world frame. This transformation matrix is

$$\mathbf{R} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where the elements of matrix R are the components of the uvn axis vectors.

- The coordinate transformation matrix is then obtained as the product of the preceding translation and rotation matrices:

$$\mathbf{M}_{WC,\,VC} = \mathbf{R} \cdot \mathbf{T}$$
$$= \begin{bmatrix} u_x & u_y & u_z & -\mathbf{u} \cdot \mathbf{P}_0 \\ v_x & v_y & v_z & -\mathbf{v} \cdot \mathbf{P}_0 \\ n_x & n_y & n_z & -\mathbf{n} \cdot \mathbf{P}_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- These matrix elements are evaluated as

$$-\mathbf{u} \cdot \mathbf{P}_0 = -x_0 u_x - y_0 u_y - z_0 u_z$$
$$-\mathbf{v} \cdot \mathbf{P}_0 = -x_0 v_x - y_0 v_y - z_0 v_z$$
$$-\mathbf{n} \cdot \mathbf{P}_0 = -x_0 n_x - y_0 n_y - z_0 n_z$$

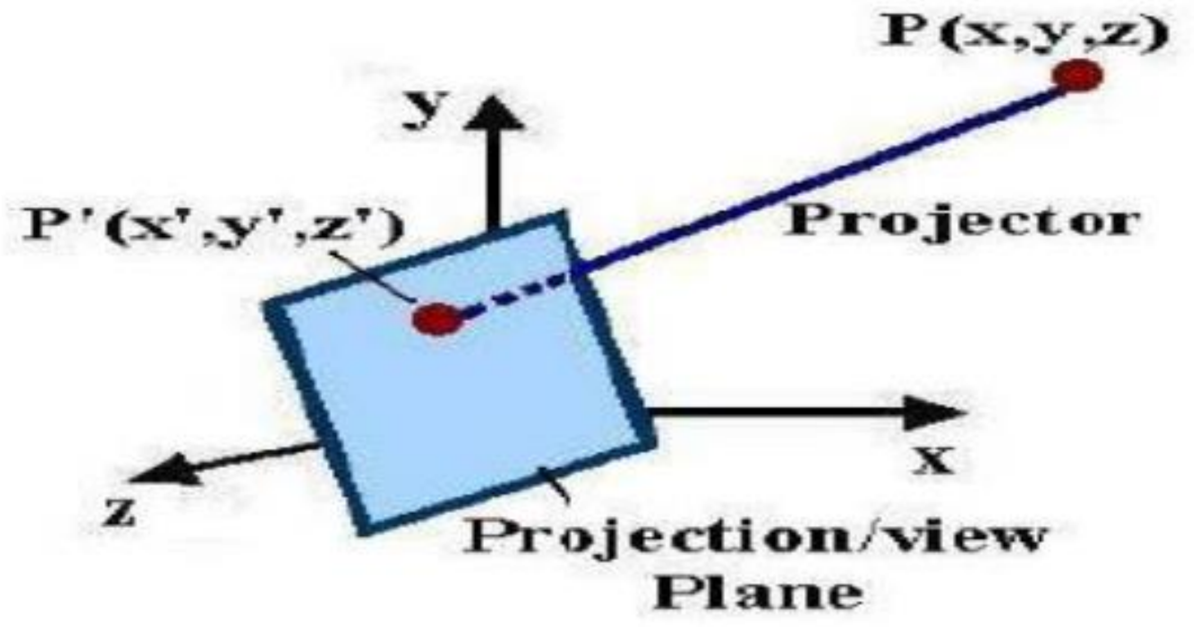
Projection Transformations :

Projection:

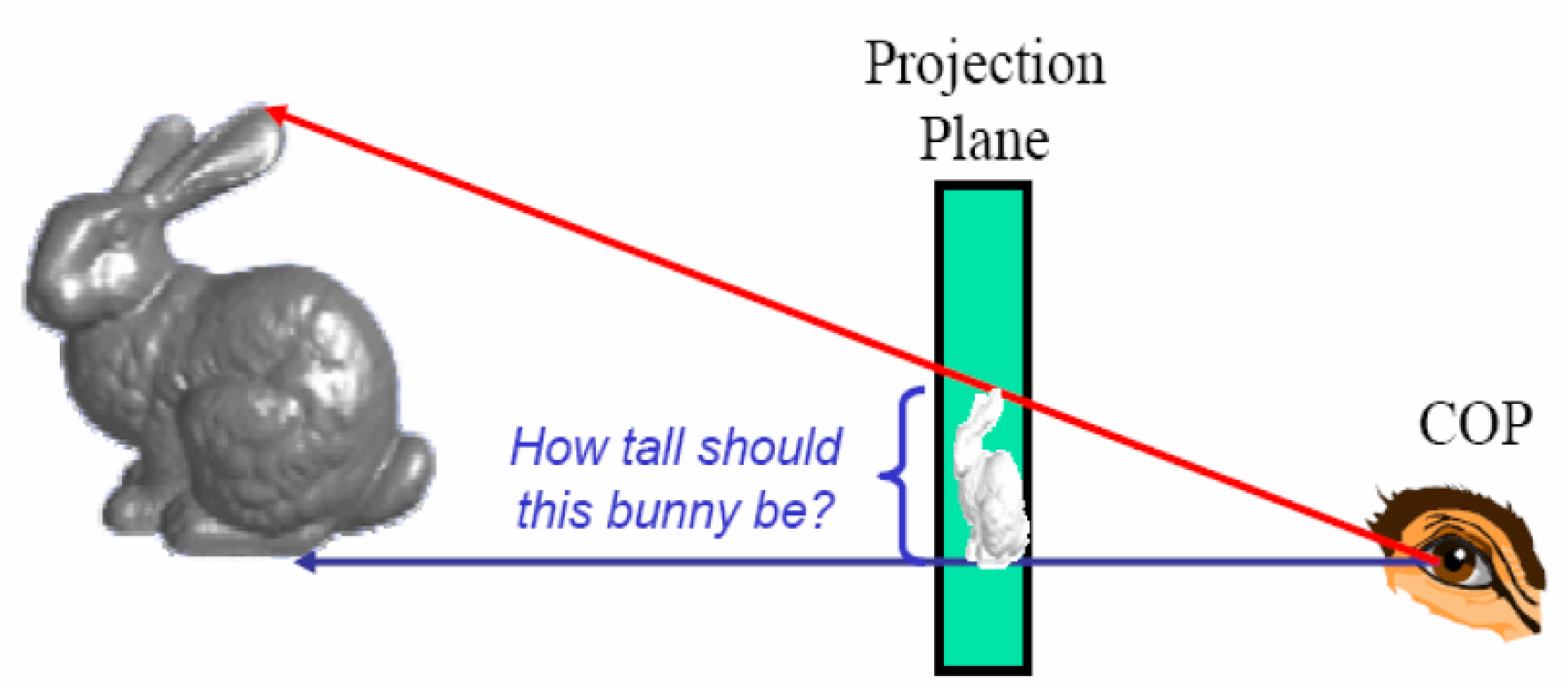
It is the process of converting a 3D object into a 2D object. It is also defined as mapping or transformation of the object in projection plane or view plane. The view plane is displayed surface.

Projection can be defined as a mapping of a point p(x,y,z) onto its image $p' = (x', y', z')$ in the projection plane (View plane).

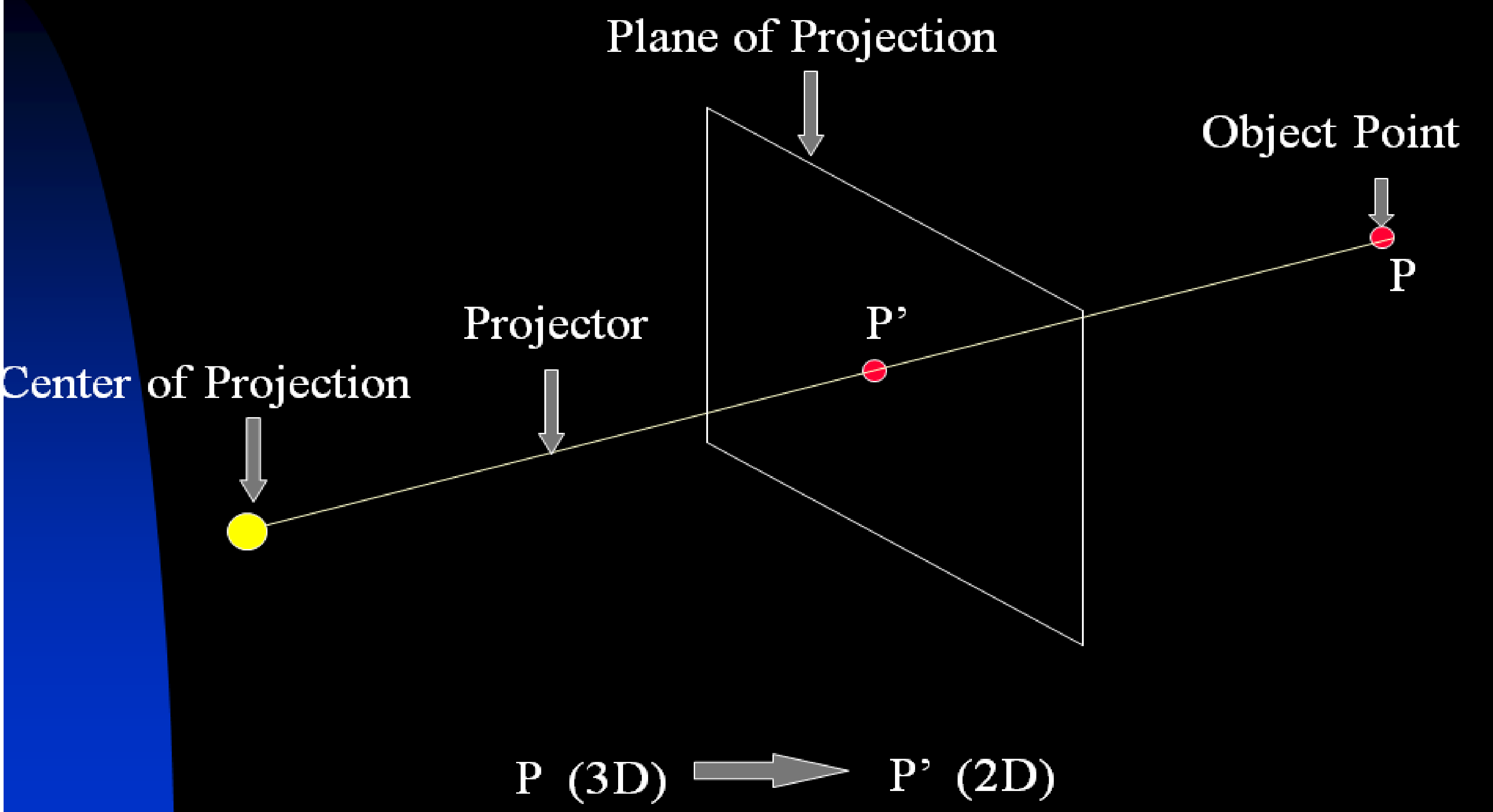
The mapping is determined by Projector that passes through P and intersects the view plane p'



Projections

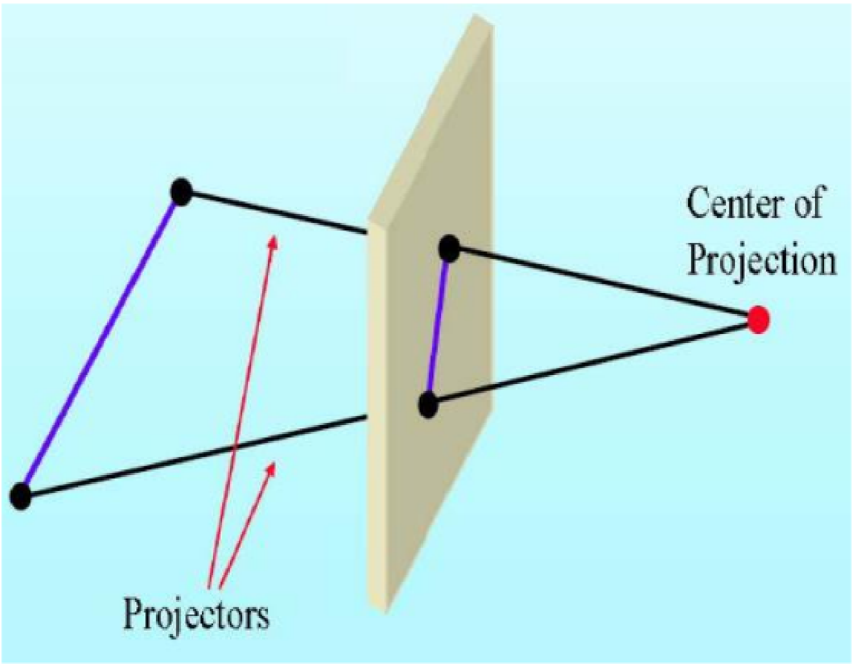


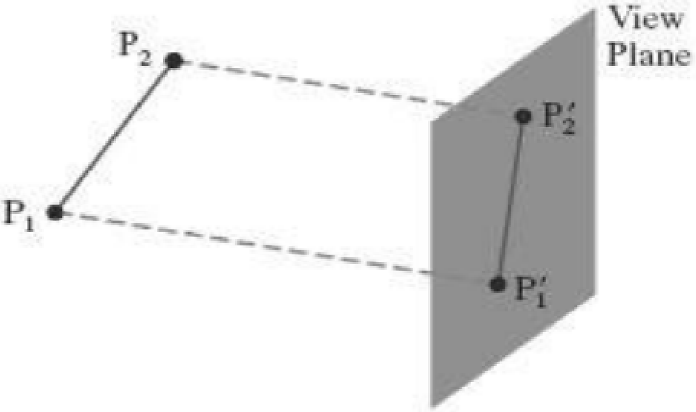
Projection Geometry

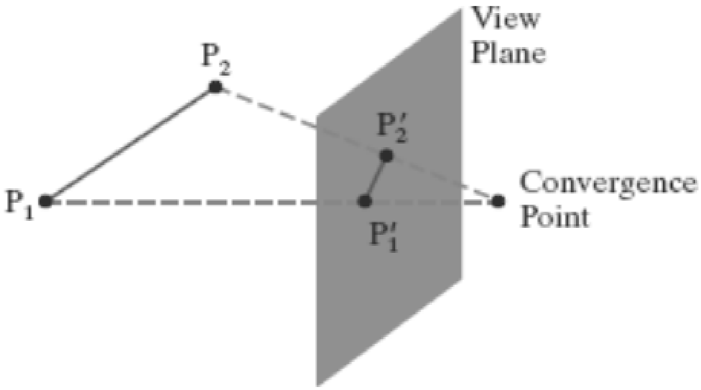


Projectors are lines from COP (center/reference of projection) through each point in the object.

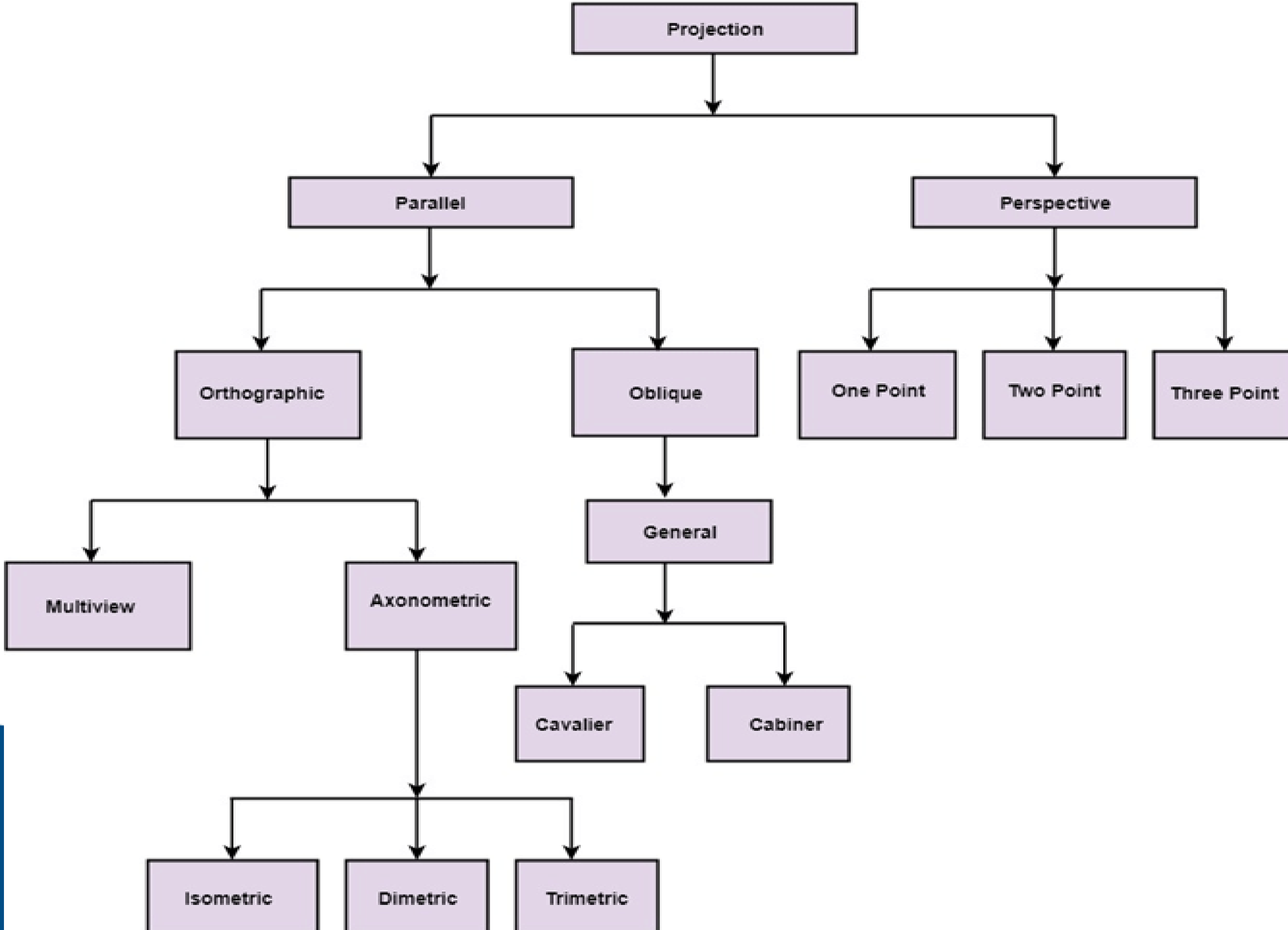
The result of projecting an object is dependent on spatial relationship among the projectors and the view plane.



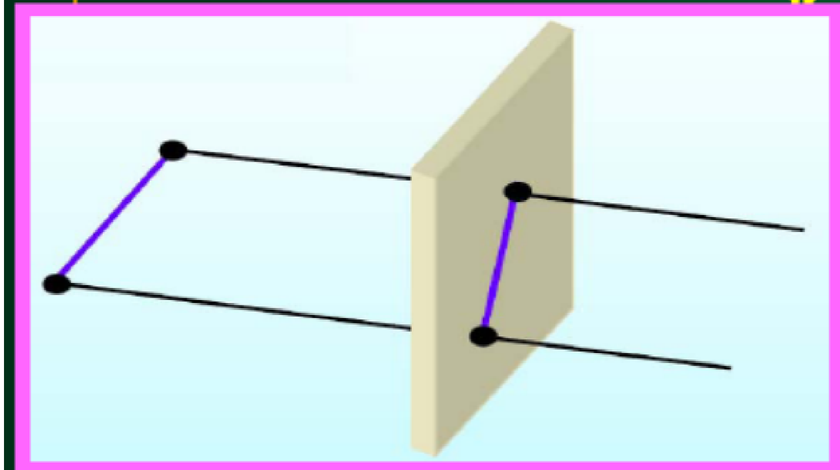


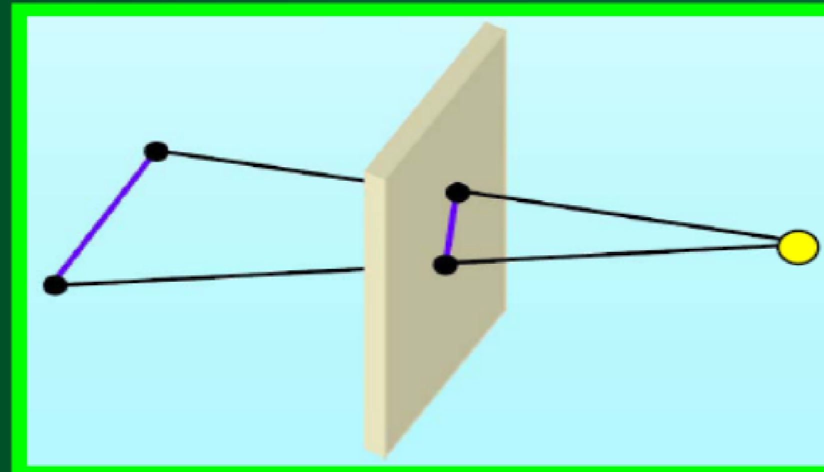


Types of Projection



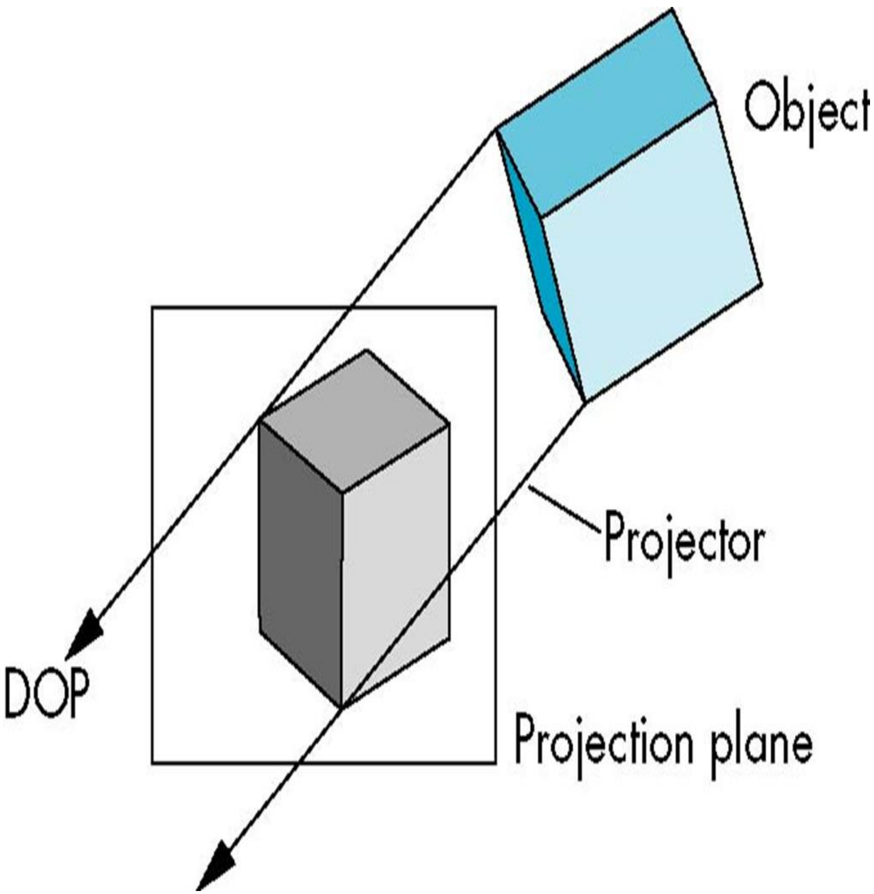
Projection

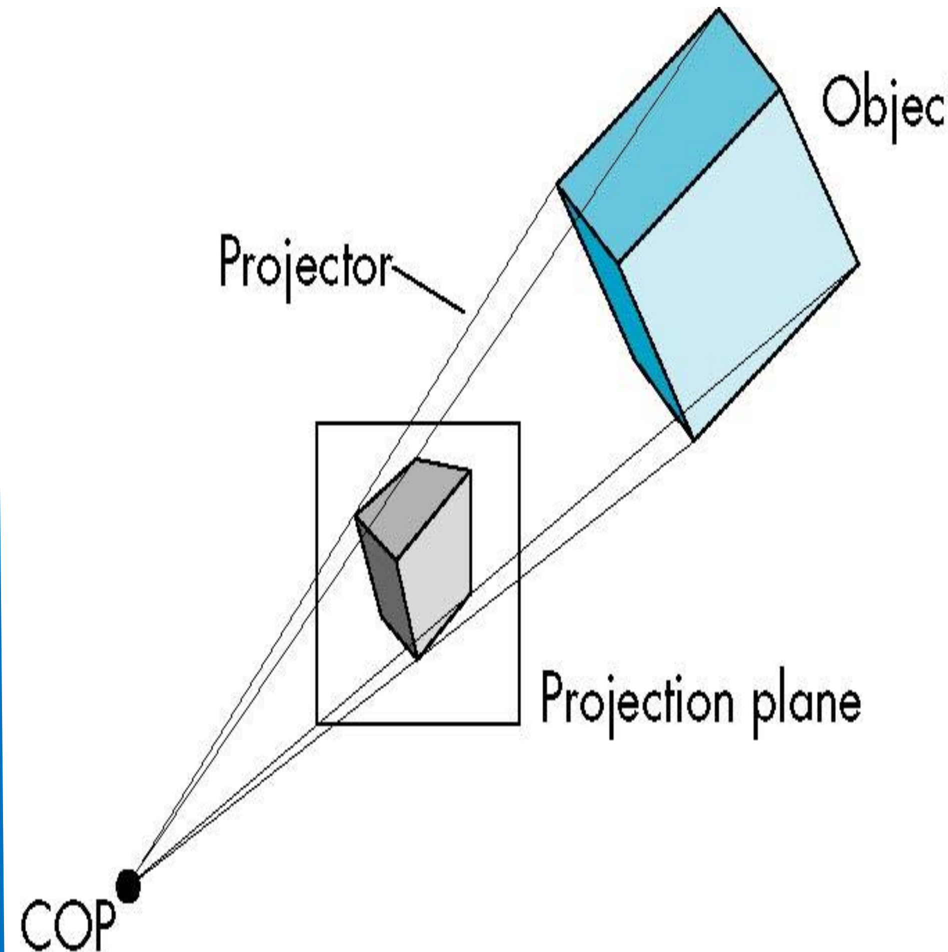




Parallel Projection :
Coordinate position are transformed to the view plane along **parallel lines**.

Perspective Projection:
Object positions are transformed to the view plane along lines that converge to the **projection reference (center) point**.


Object
Projector
DOP
Projection plane


Object
Projector
Projection plane
COP

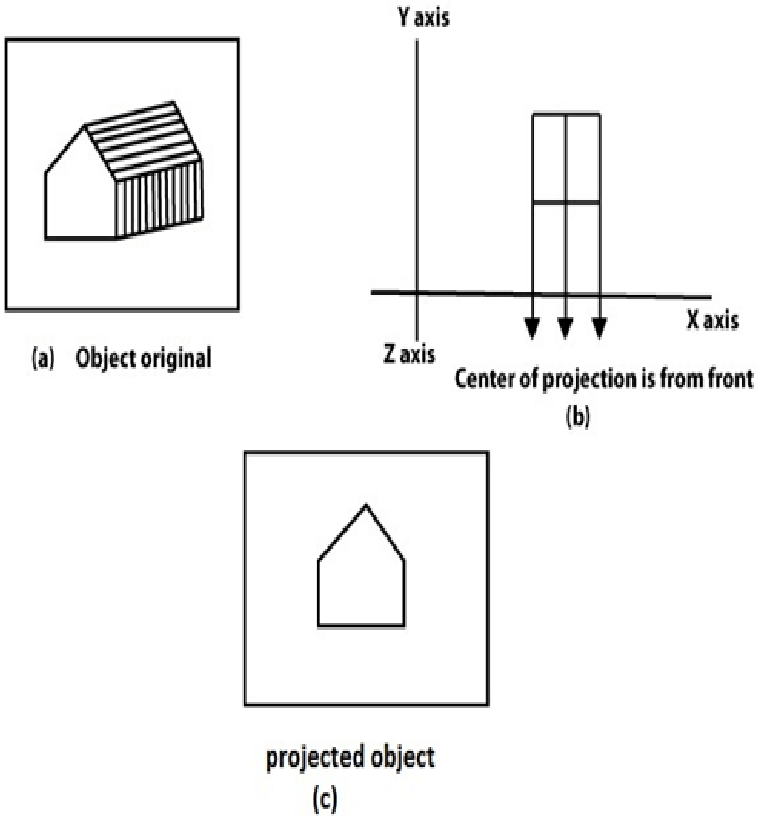


Fig (a) shows original object. Fig (b) shows object when projection is taken. Fig (c) gives projected object.

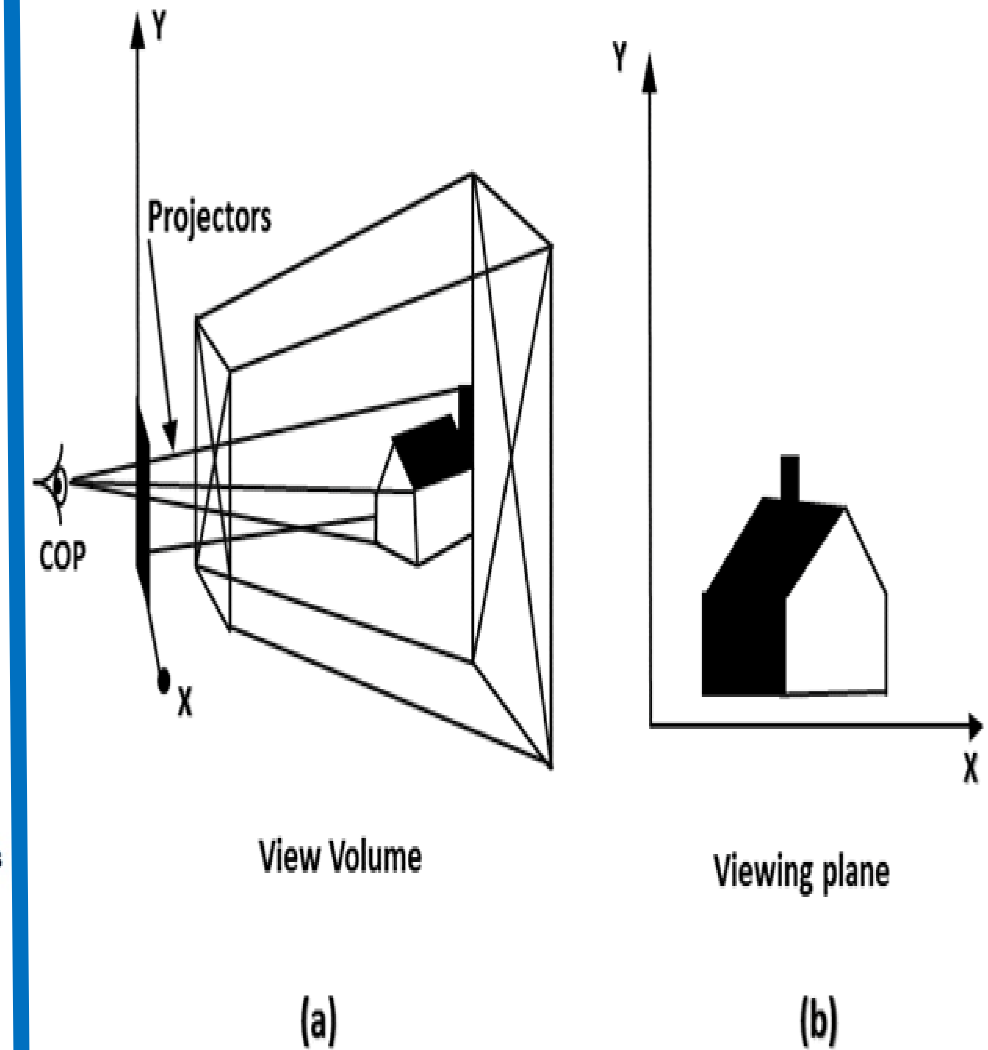
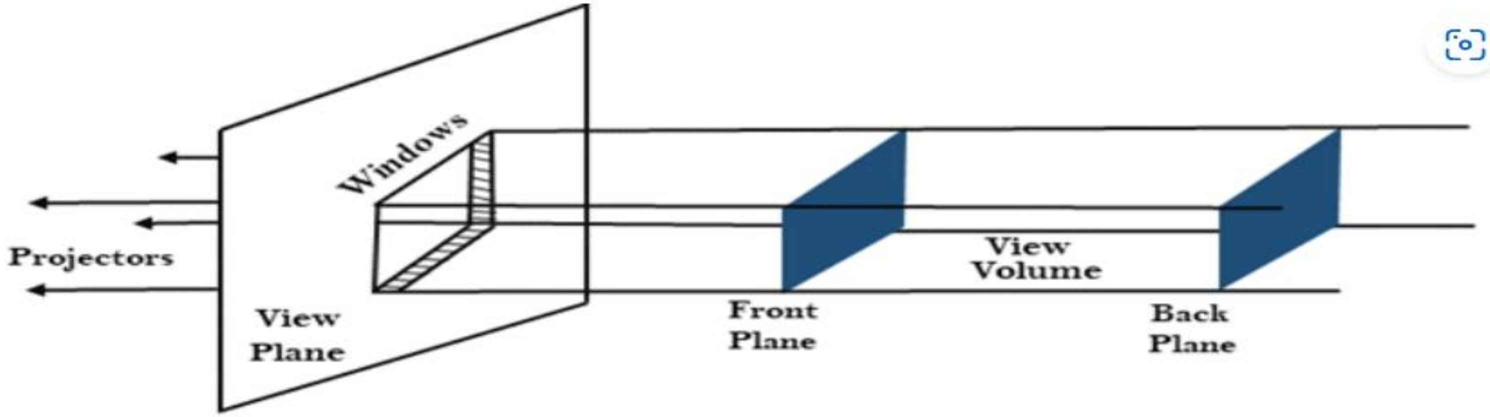
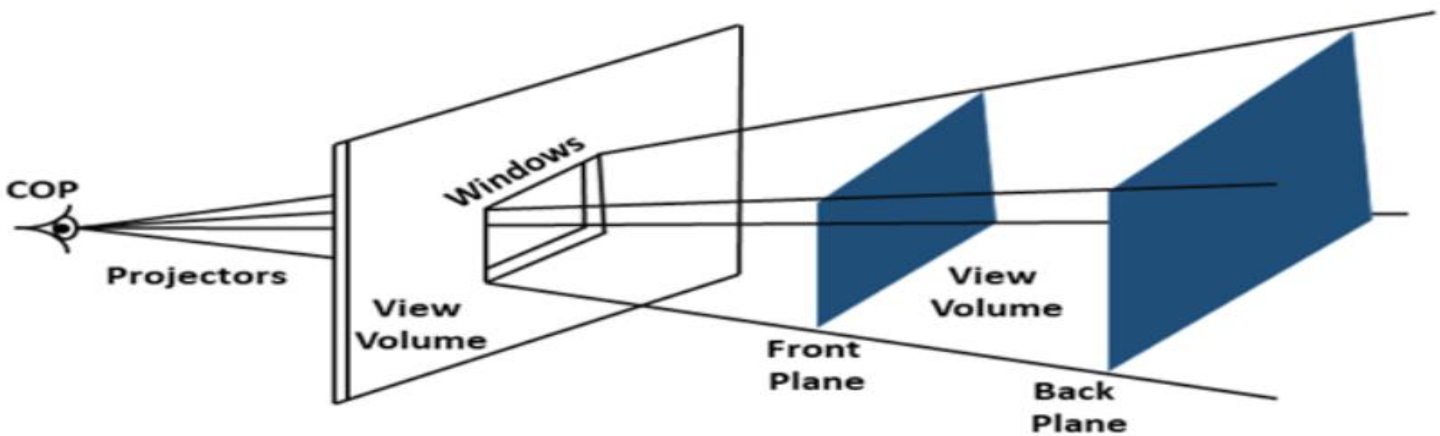


Fig: Prespective Projection



(a) Viewing Volume in orthographic projection

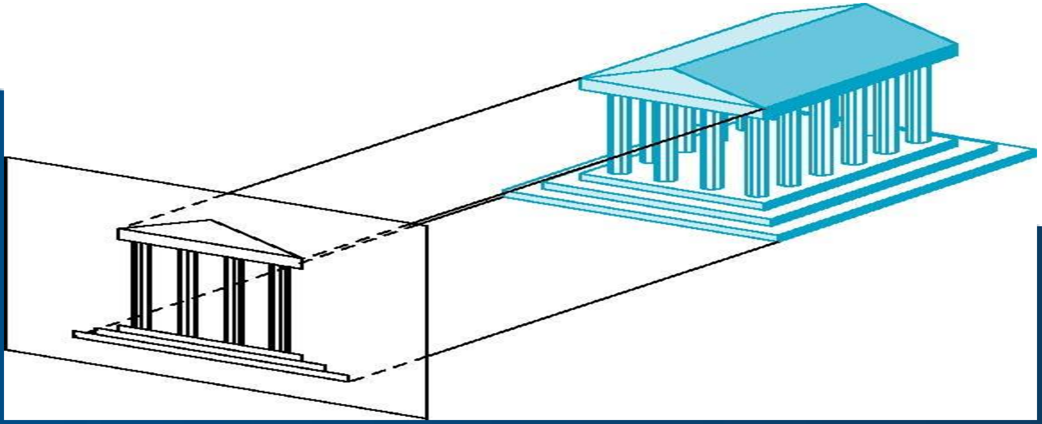


Difference Between perspective projection and parallel projection

Parallel projection	Perspective projection
The Center of projection at infinity results with a parallel projection	The center of projection is at a finite distance from the viewing plane
Direction of projection is specified	Explicitly specify: center of projection
No change in the size of object	Size of the object is inversely proportional to the distance of the object from the center of projection
A parallel projection observes relative proportion of objects, but does not give us a realistic representation of the appearance of object.	Produces realistic views but does not preserve relative proportion of objects
Used for exact measurement	Not useful for recording exact shape and measurements of the object
Parallel lines do remain parallel	Parallel lines do not in general project as Parallel

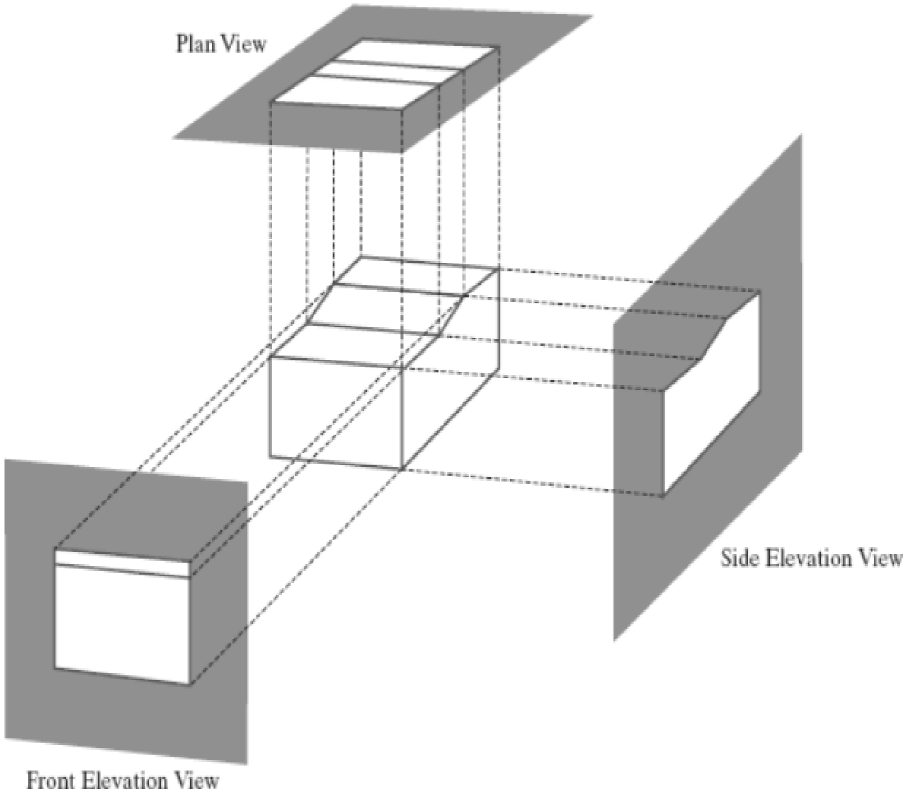
Parallel Projection:

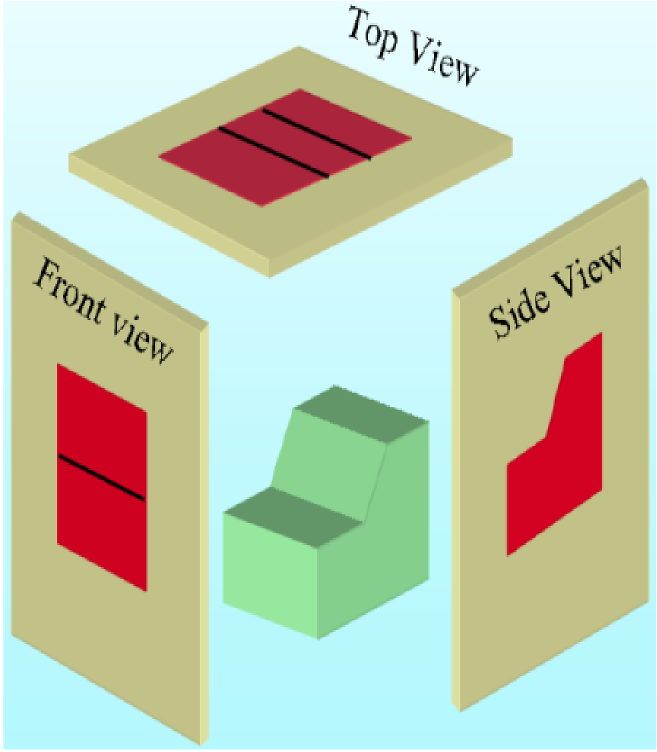
- Parallel Projection use to display picture in its true shape and size.
- Oblique and Orthographic drawings are somehow similar to each other except that **Oblique drawings** shows only one face of the object in true shape whereas **Orthographic drawings** shows all the faces.
- The parallel projection is formed by extending parallel lines from each vertex on the object until they intersect the plane of the screen. The point of intersection is the projection of vertex.
- Parallel projections are used by architects and engineers for creating working drawing of the object, for complete representations require two or more views of an object using different planes.



Orthogonal Projections:

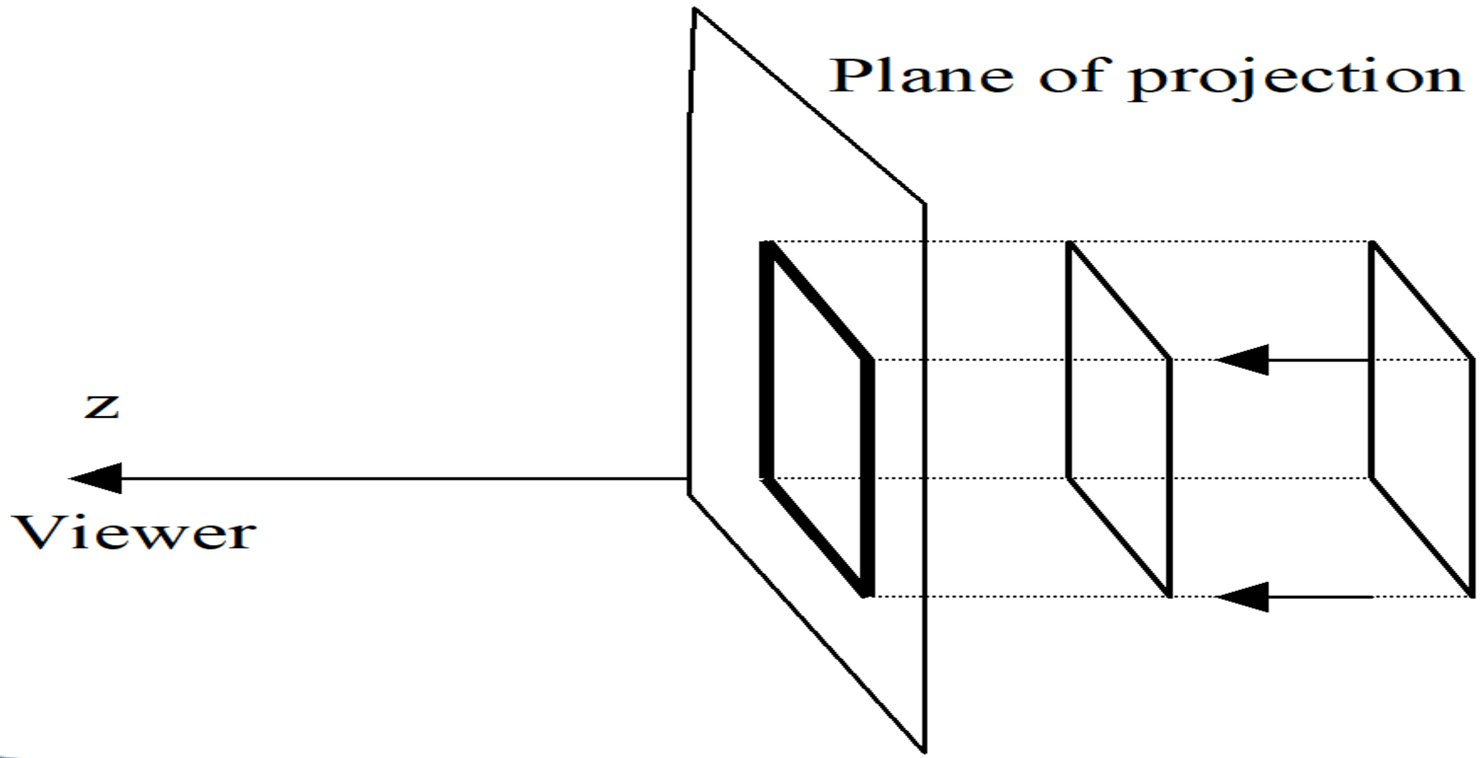
- **A transformation of object descriptions to a view plane along lines that are all parallel to the view-plane normal vector N is called an orthogonal projection also termed as orthographic projection.**
- This produces a parallel-projection transformation in which the projection lines are perpendicular to the view plane.
- Orthogonal projections are most often used to produce the front, side, and top views of an object
- Rear orthogonal projections of an object are called elevations; and a top orthogonal projection is called a plan view


Plan View
Side Elevation View
Front Elevation View


Top View
Front view
Side View

Orthographic Projection

The lines of projection are parallel, and at the same time orthogonal to the plane of projection.



Advantages and Disadvantages of Orthogonal Projections

Advantages:

- Preserves both distances and angles
  - Shapes preserved
  - Can be used for measurements
    - Building plans
    - Manuals

DisAdvantages:

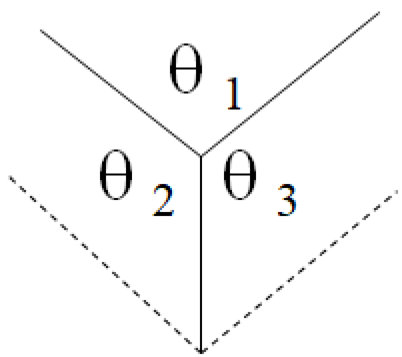
- Cannot see what object really looks like because many surfaces hidden from view
  - Often we add the isometric

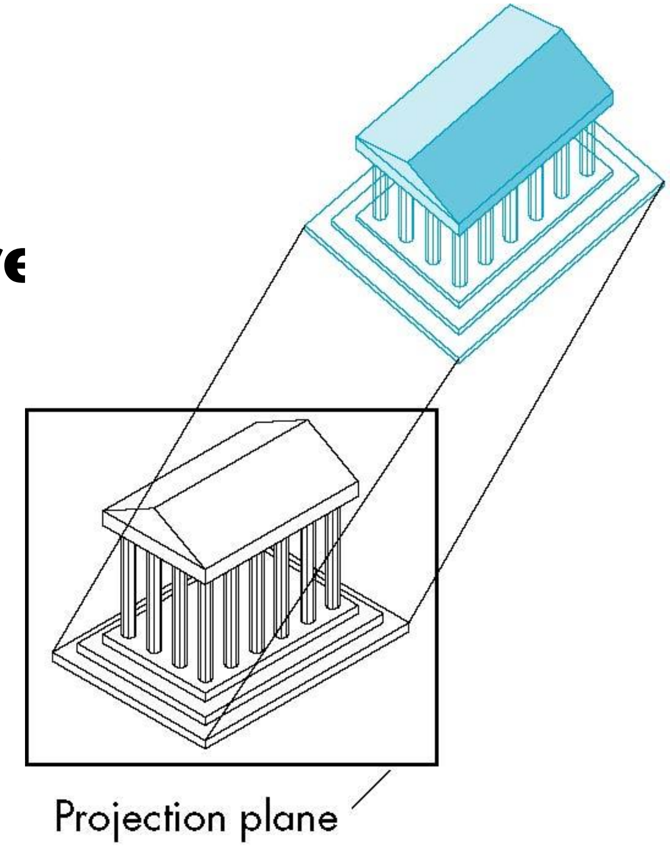
Axonometric Orthogonal Projections

- Orthogonal projections that display more than one face of an object. Such views are called axonometric orthogonal projections.
- Allow projection plane to move relative to object

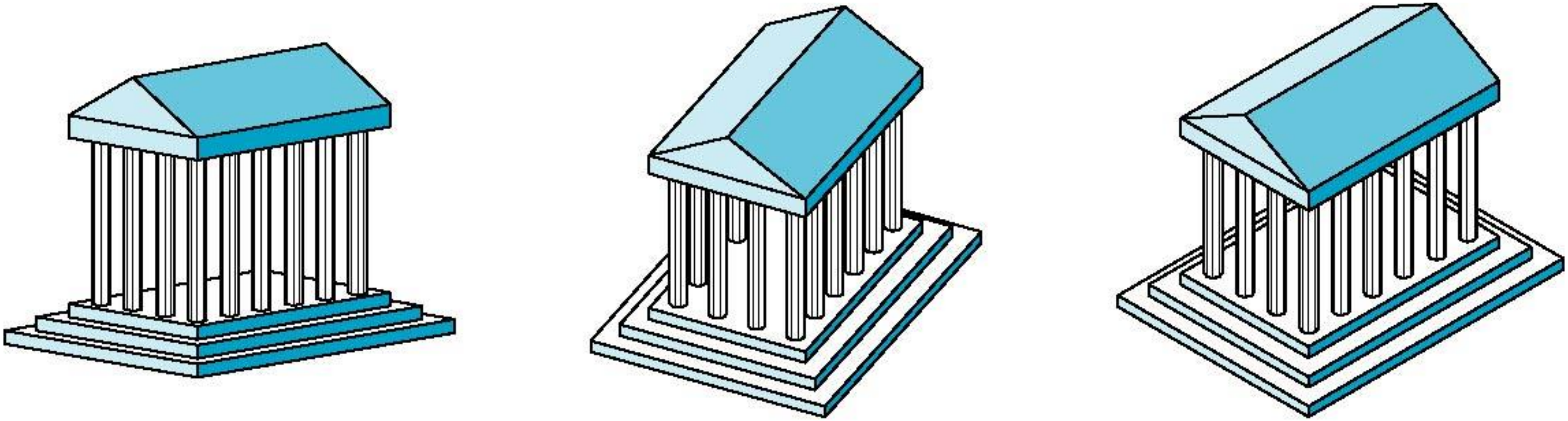
classify by how many angles of a corner of a projected cube are the same

none: trimetric
two: dimetric
three: isometric



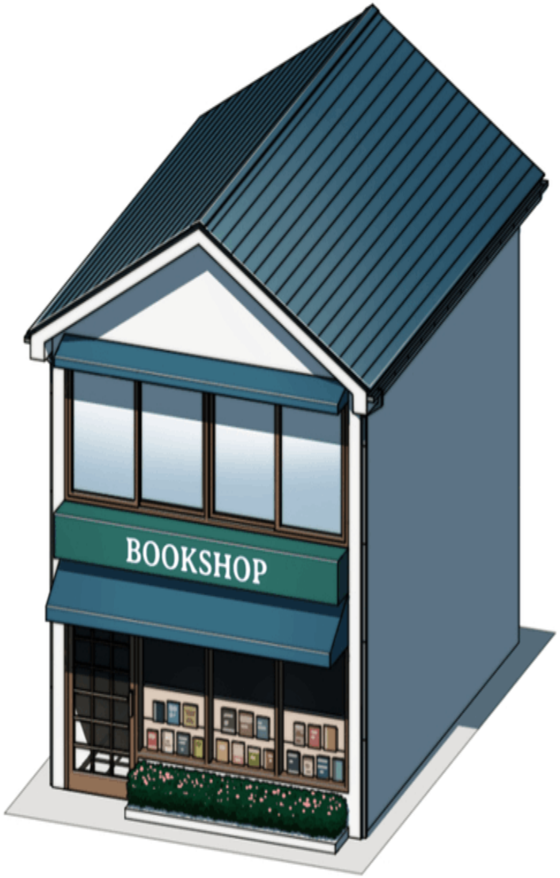


Types of Axonometric Projections



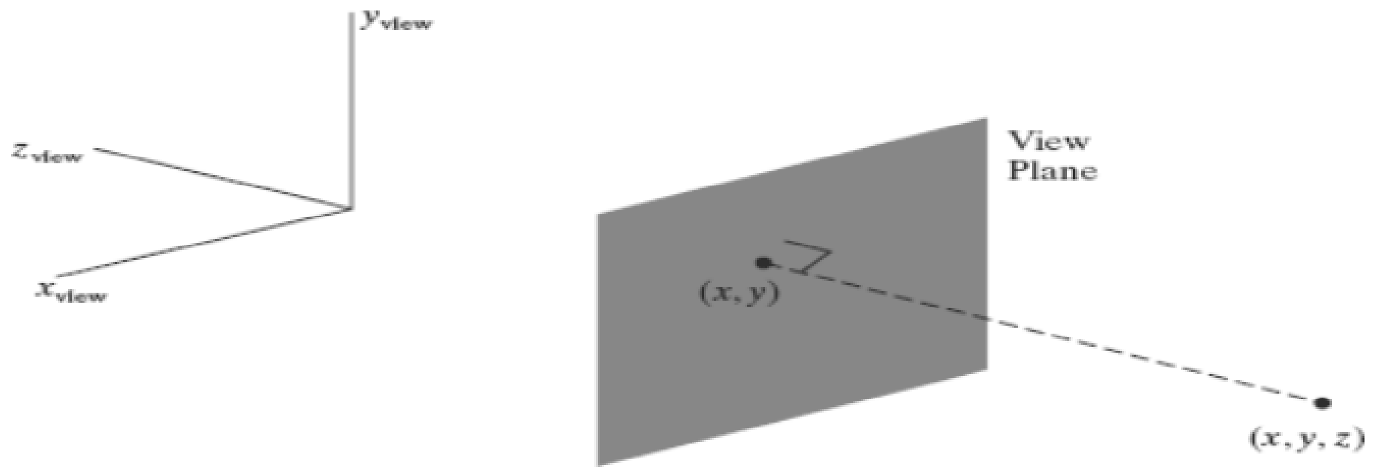
Dimetric Trimetric Isometric

- **Isometric projection:** In a three-axis direction, the amount of the foreshortening is equal.
- **Dimetric projection:** In this projection, two axes are the body is foreshortened in the same amount and the third axis is foreshortened in different amount.
- **Trimetric projection:** Foreshortenings in three directions are different.


BOOKSHOP

Orthogonal Projection Coordinates

- With the projection direction parallel to the z_{view} axis, the transformation equations for an orthogonal projection are trivial. For any position *(x, y, z)* in viewing coordinates, as in Figure below, the projection coordinates are $x_p = x, \; y_p = y$



$$\mathbf{M}_{Orthographic\ Parallel} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic Projection

Projection along z axis:

Transformation: $x' = x$

$y' = y$

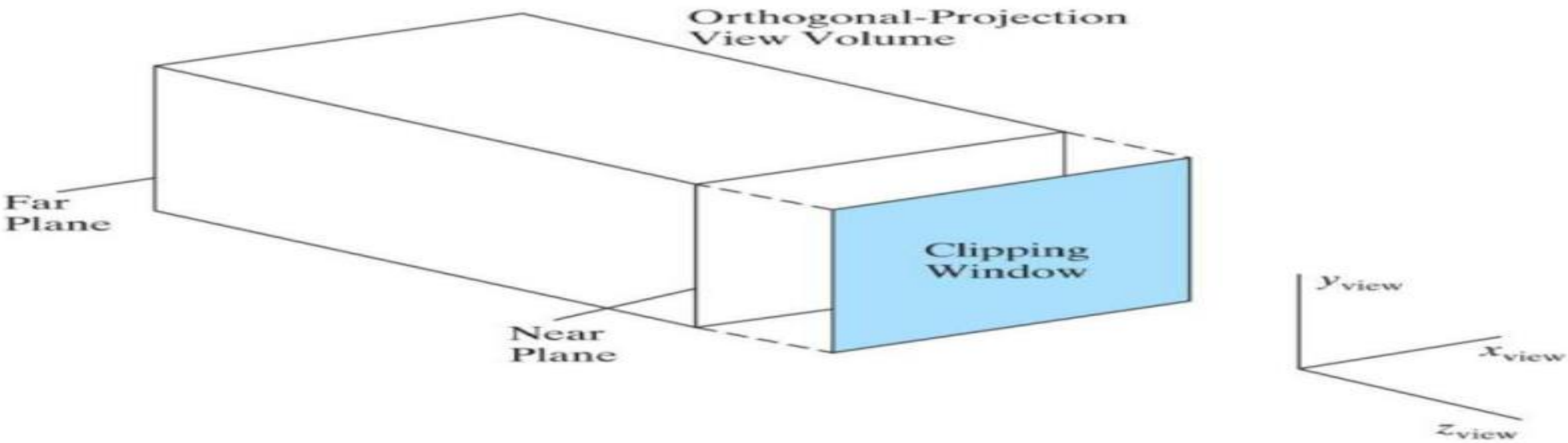
z-coordinate information is lost!

Orthographic Projection Matrix:

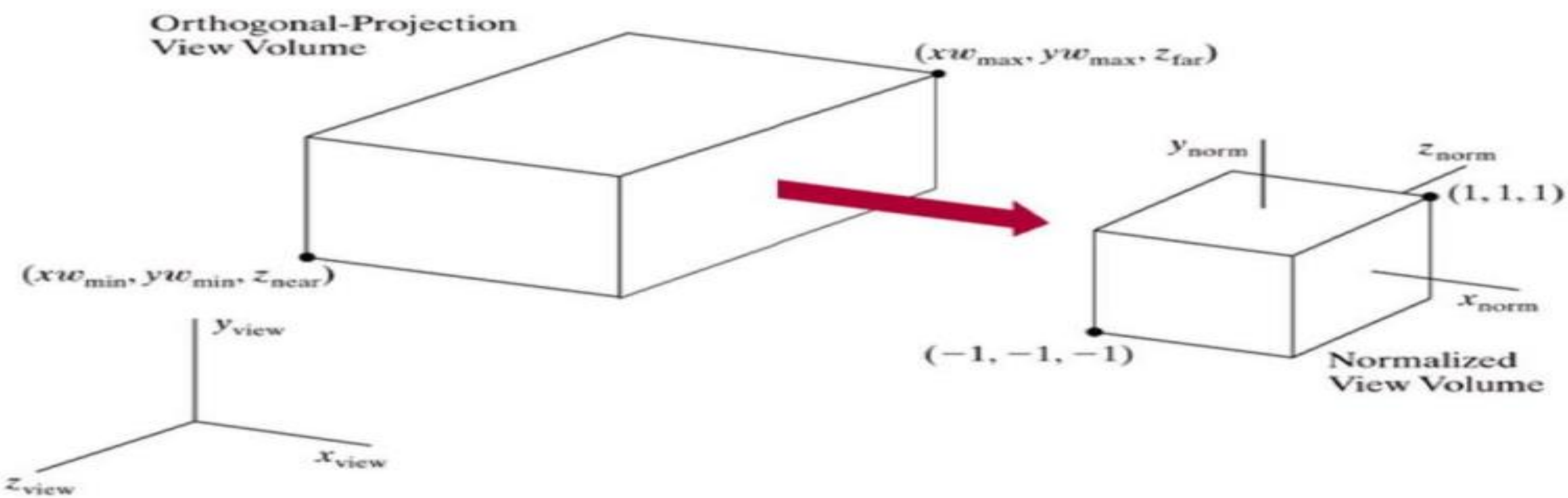
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Clipping and Orthogonal projection view :

- **The clipping window**: the **x** and **y** limits of the scene you want to display
- These form the orthogonal-projection view volume
- The depth is limited by near and far clipping planes in z_{view}

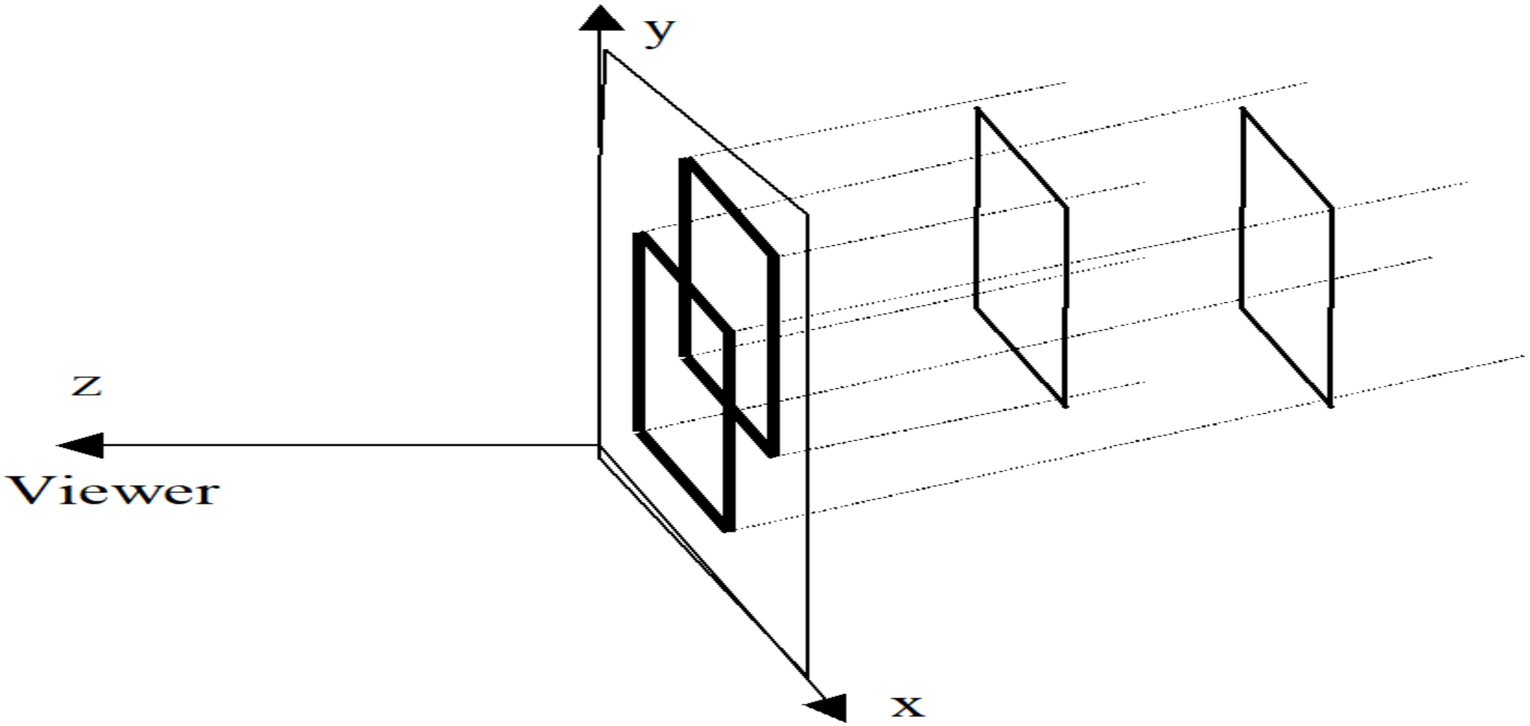


- Map the view volume into a normalized view volume.



Oblique Projection

The lines of projection are parallel, but not orthogonal to the plane of projection.



Oblique Projection

Transformation:

$$x' = x + k_1 z$$
$$y' = y + k_2 z$$

The z-coordinate value of the object point, leads to a shift of x, y coordinates of the projected point, proportional to z.

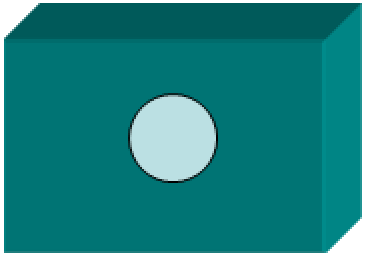
Oblique Projection Matrix:

$$\begin{bmatrix} 1 & 0 & k_1 & 0 \\ 0 & 1 & k_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Advantages and Disadvantages of Orthogonal Projections

Advantages:

- Can pick the angles to emphasize a particular face
  - Architecture: plan oblique, elevation oblique
- Angles in faces parallel to projection plane are preserved while we can still see "around" side



DisAdvantages:

- In physical world, cannot create with simple camera; possible with bellows camera or special lens (architectural)