Computer Graphics

Unit 4 – Part 1

-By Manjula. S

Classical Viewing

- Viewing requires three basic elements
  - One or more objects
  - A viewer with a projection surface
  - Projectors that go from the object(s) to the projection surface
- Classical views are based on the relationship among these elements
  - The viewer picks up the object and orients it according to how she would like to see it
- Each object is assumed to be constructed from flat principal faces
  - Buildings, polyhedral, manufactured objects

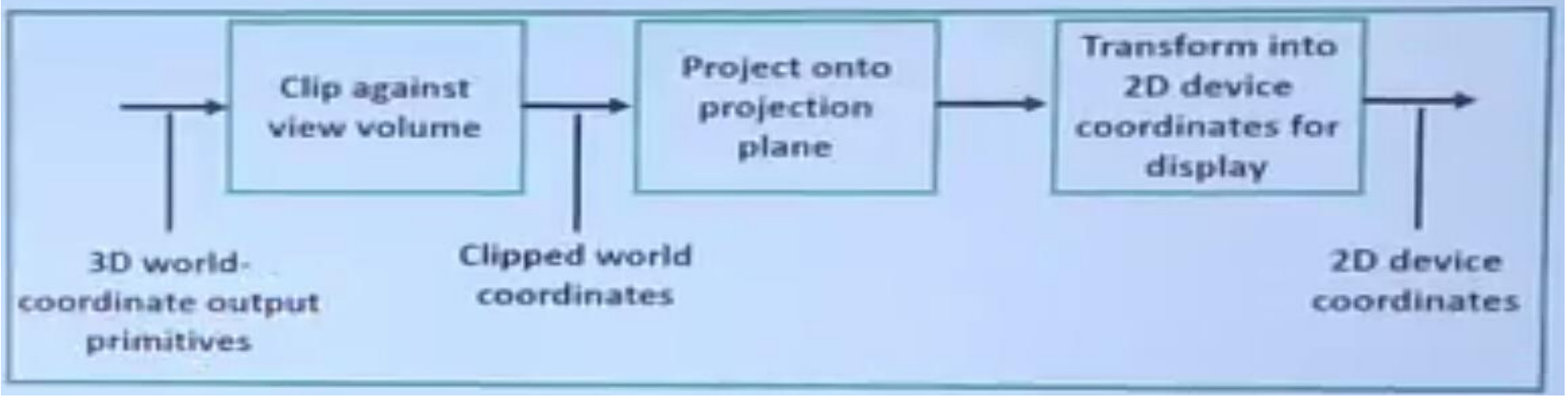
Classical and Computer Viewing

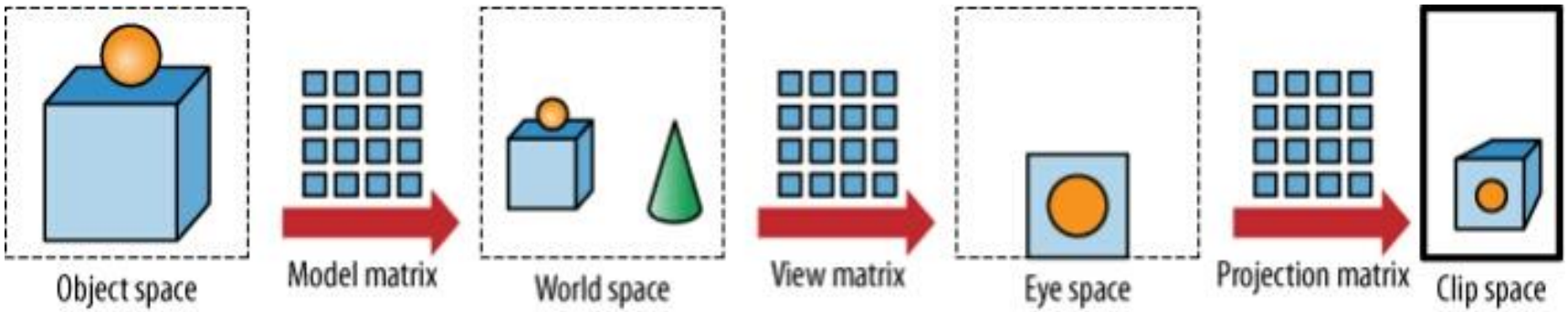
- Viewing requires three basic elements
  - One or more **objects**
  - A **viewer** with a projection surface
  - **Projectors** that go from the object(s) to the **projection surface**

Classical Viewing

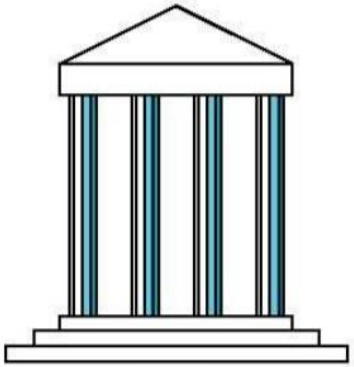
- Classical views are based on the **relationship** among basic elements
  - **The viewer picks up the object and orients it how she would like to see it**
- Each object is assumed to constructed from flat *principal faces*
  - **Buildings, polyhedra, manufactured objects**

3D Viewing Process

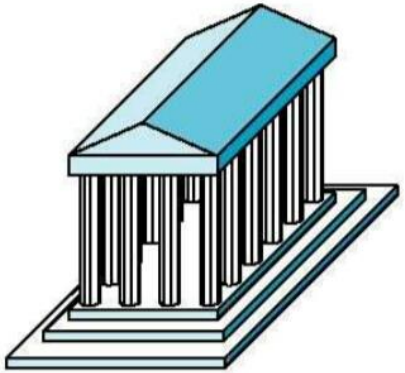




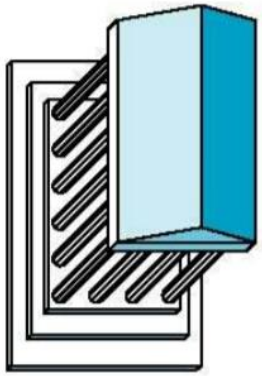
Classical Projections



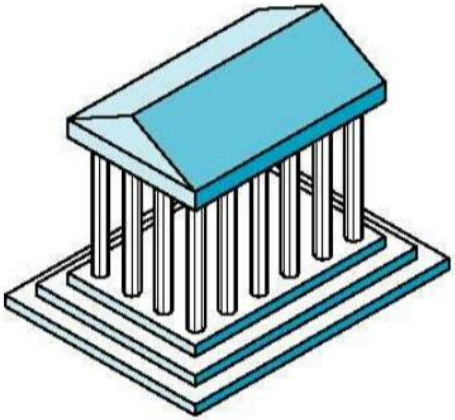
Front elevation



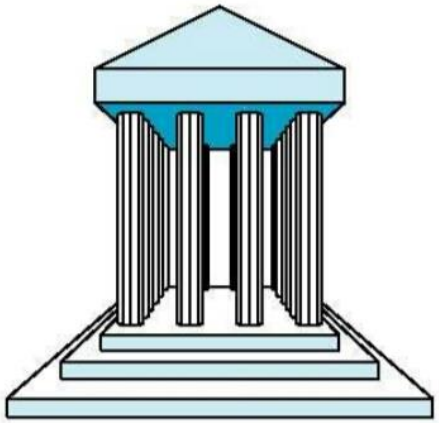
Elevation oblique



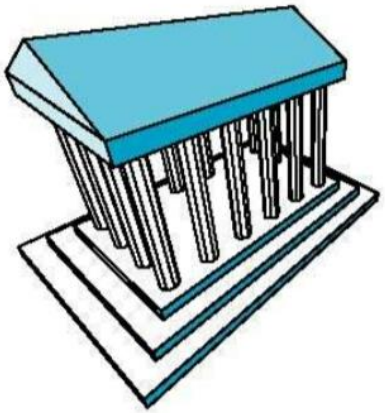
Plan oblique



Isometric



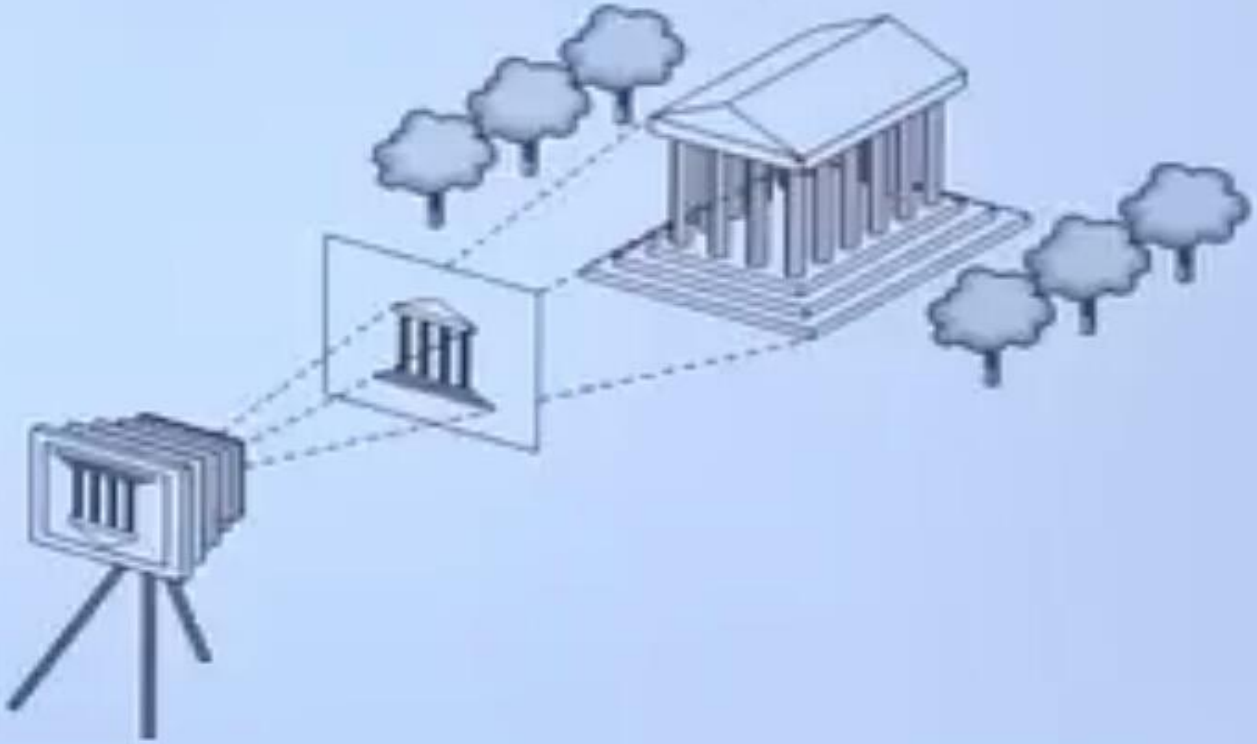
One-point perspective



Three-point perspective

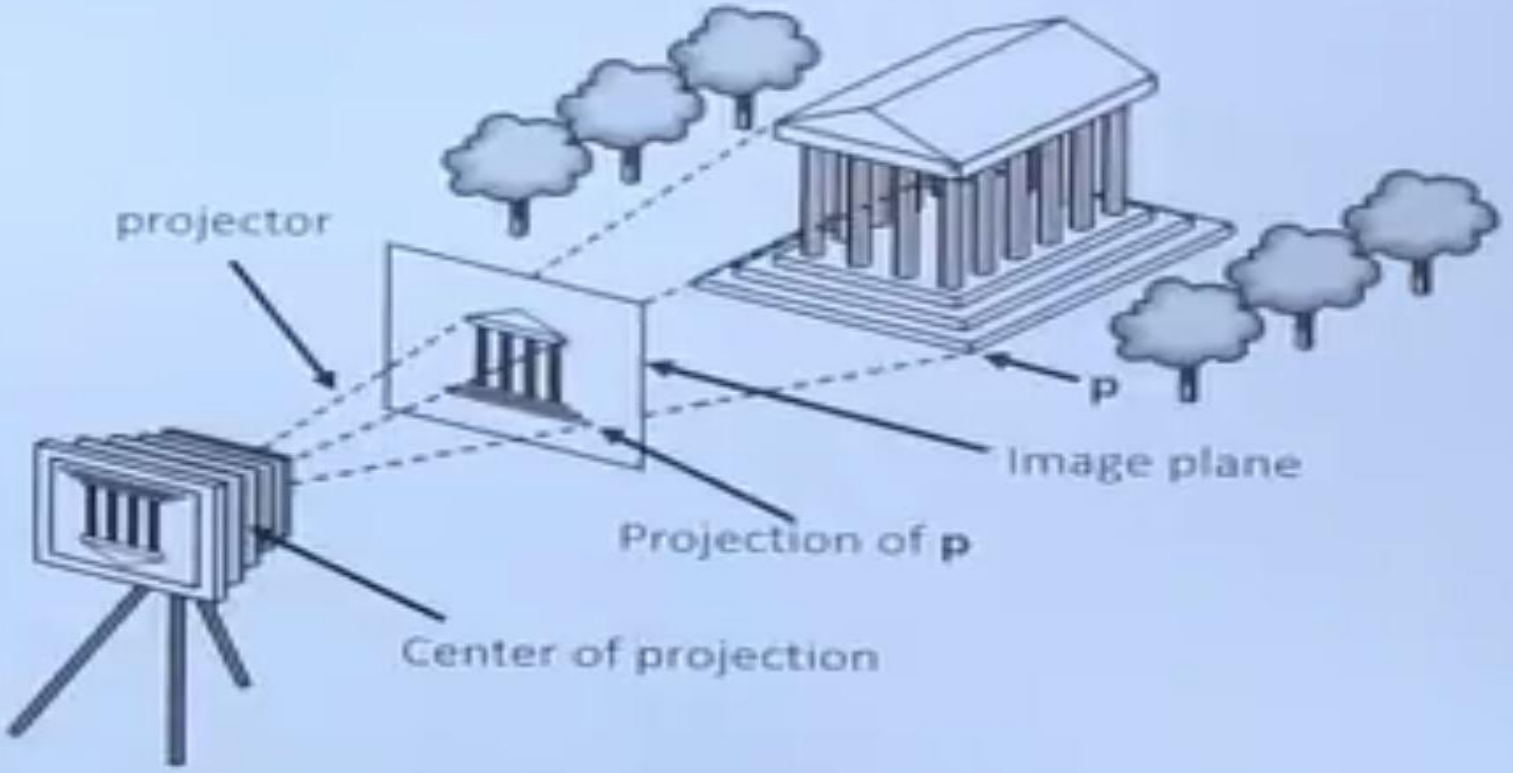
3D Synthetic Camera Model

- The synthetic camera model involves two components, specified independently:
  - objects (a.k.a geometry)
  - viewer (a.k.a camera)



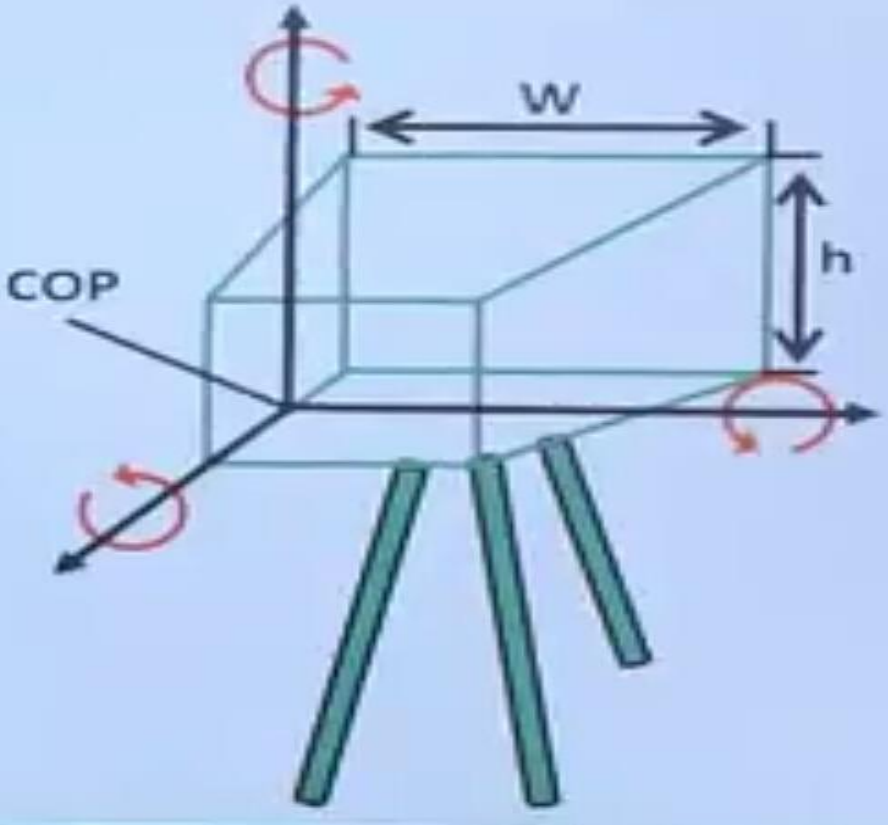
Imaging with the Synthetic Camera

- The image is rendered onto an image plane or project 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.



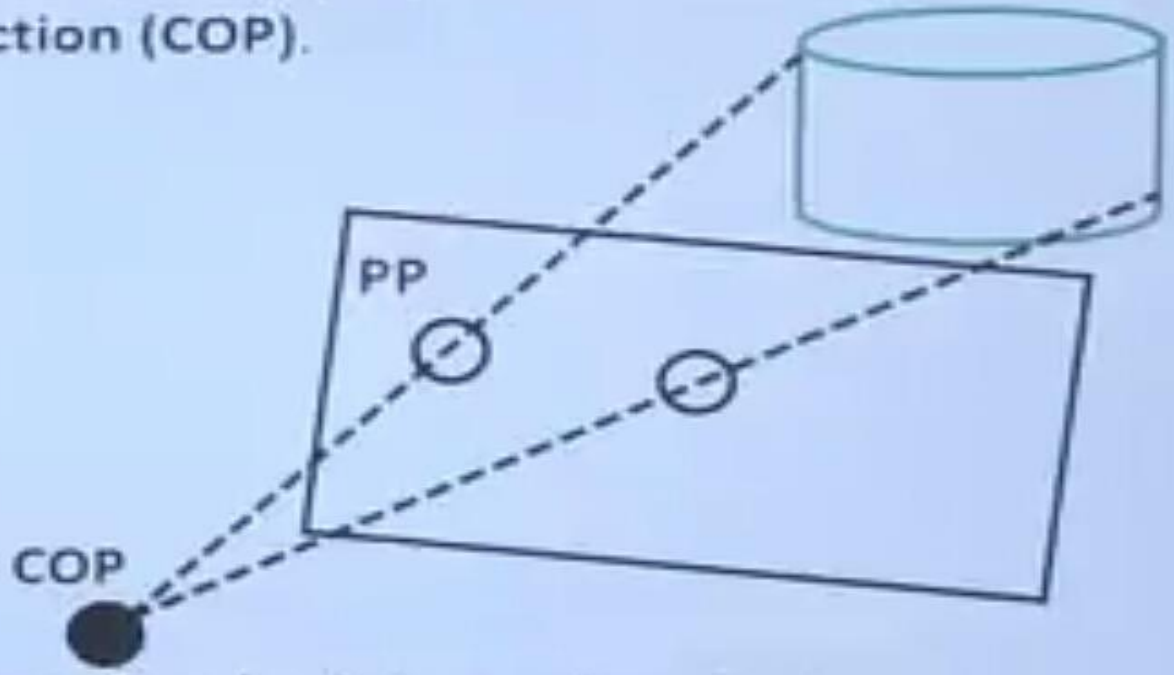
Specifying a Viewer

- **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.



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)**.

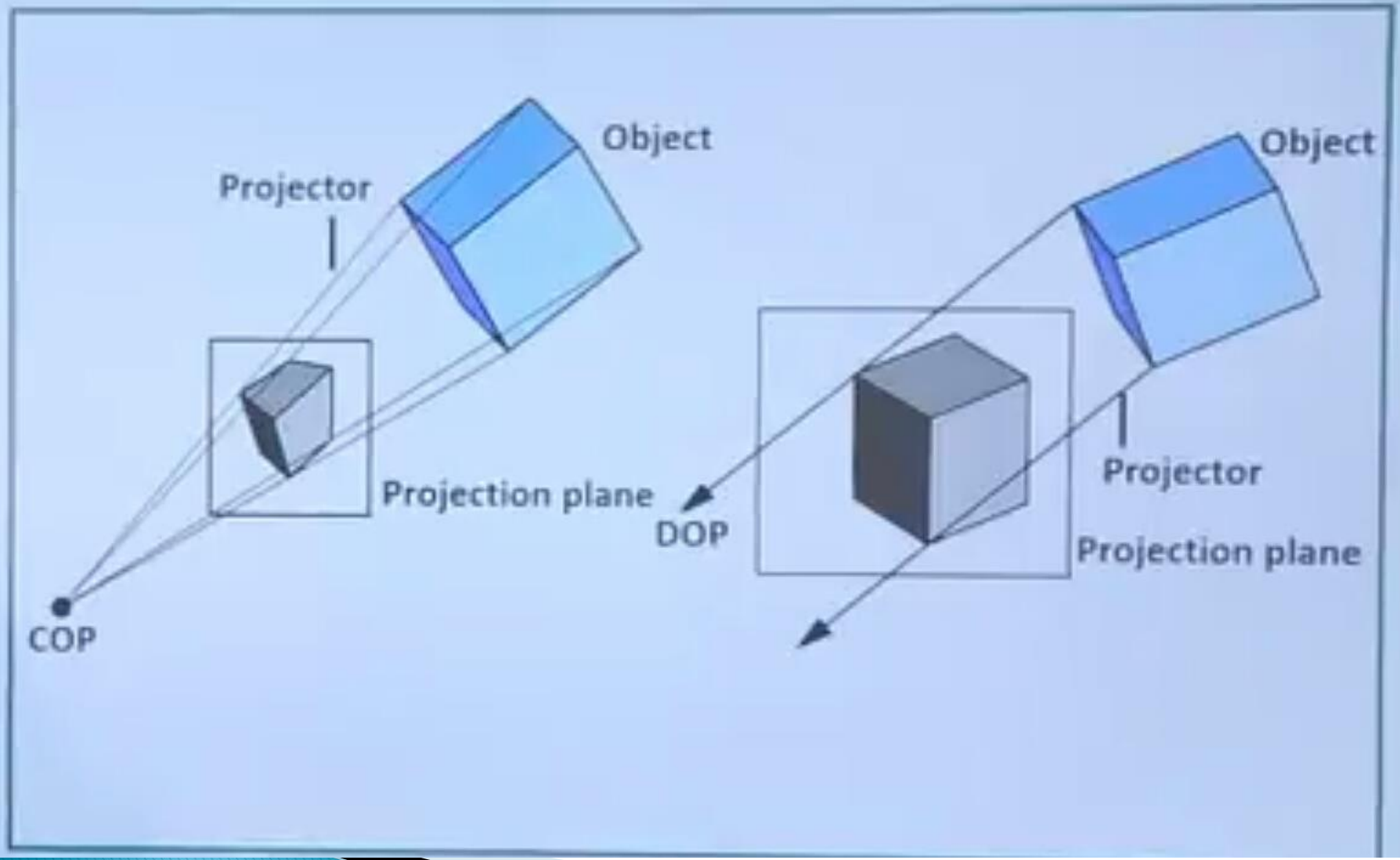


- **There are two basic type of projections:**
  - **Perspective** – distance from COP to PP finite
  - **Parallel** – distance from COP to PP infinite

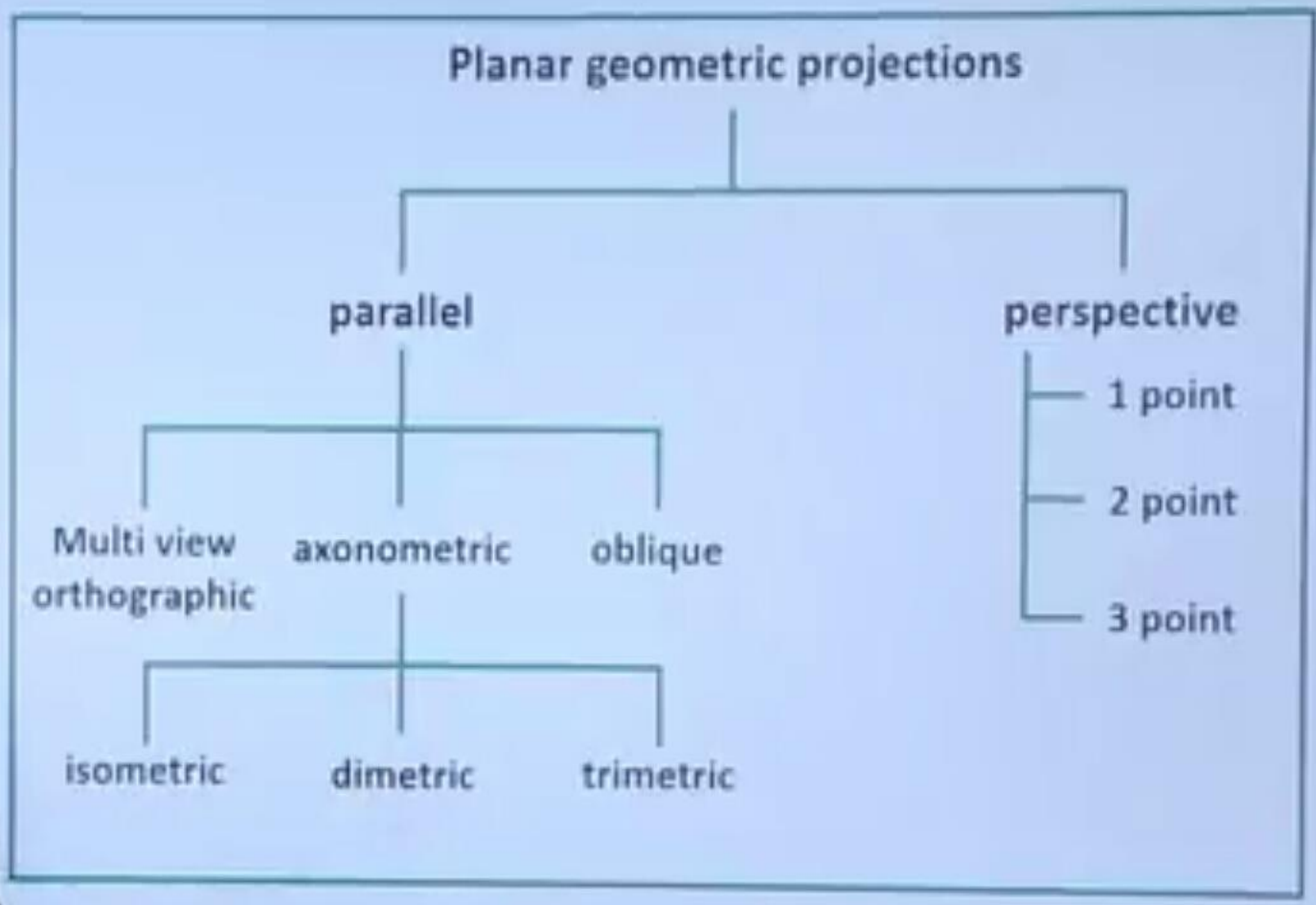
Perspective vs. Parallel Projections

- Computer graphics treats all projections the same and implements them with a single pipeline
- Classical viewing developed different techniques for drawing each type of projection
- Fundamental distinction is between parallel and perspective viewing even though mathematically parallel viewing is the limit of perspective viewing

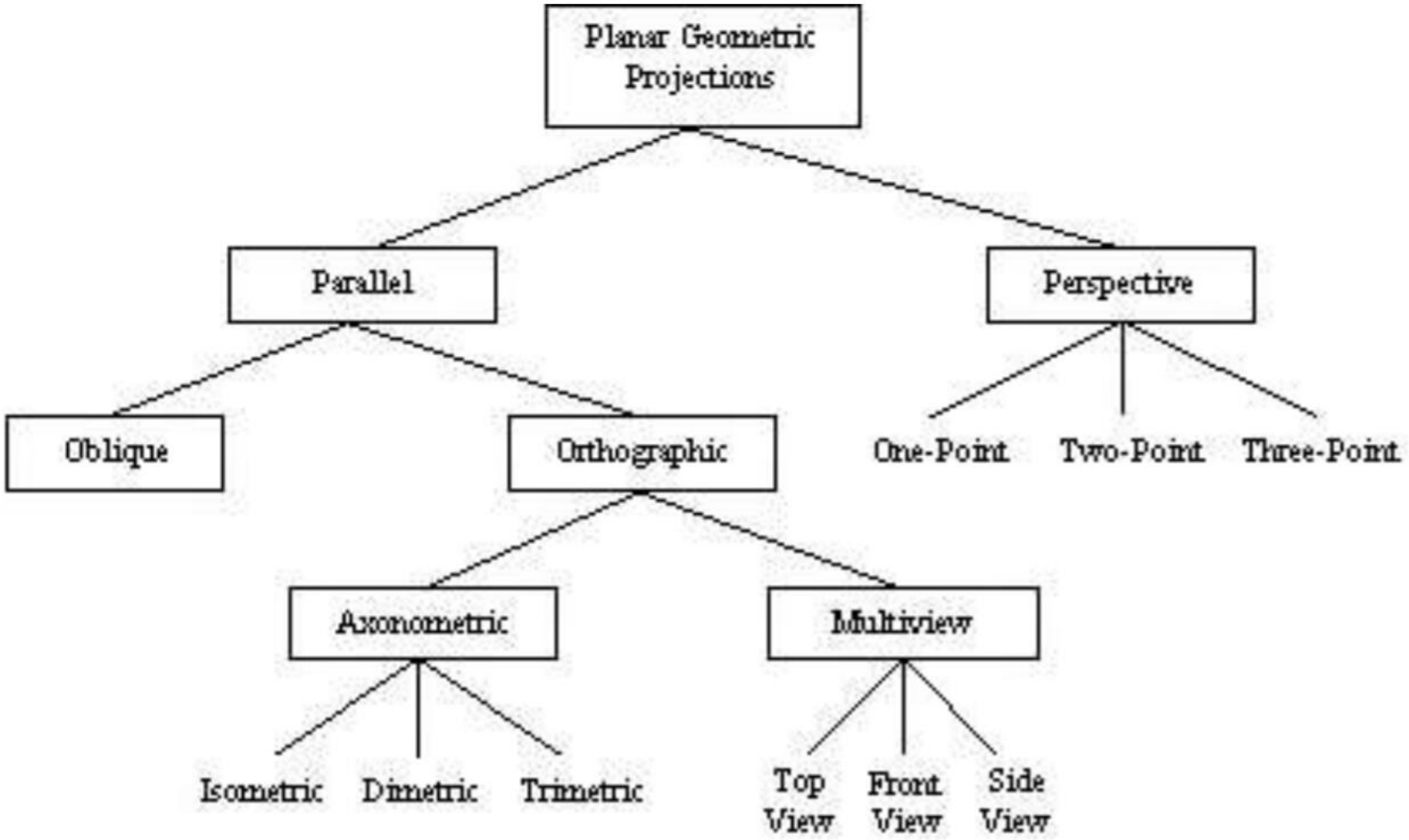
Perspective vs. Parallel Projections



Taxonomy of Planar Geometric Projections



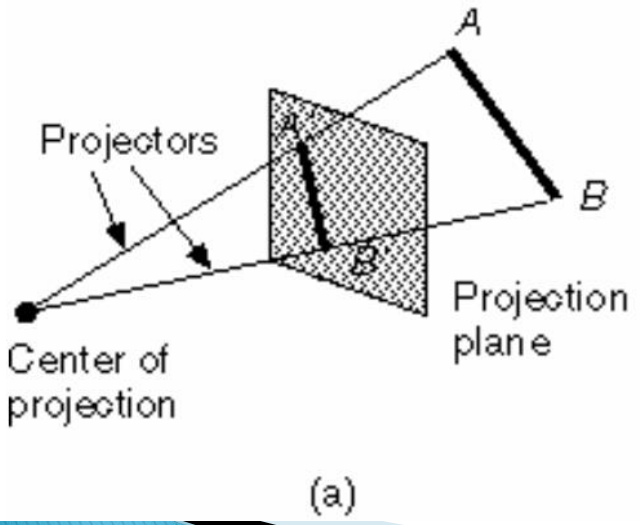
Classical Viewing projections

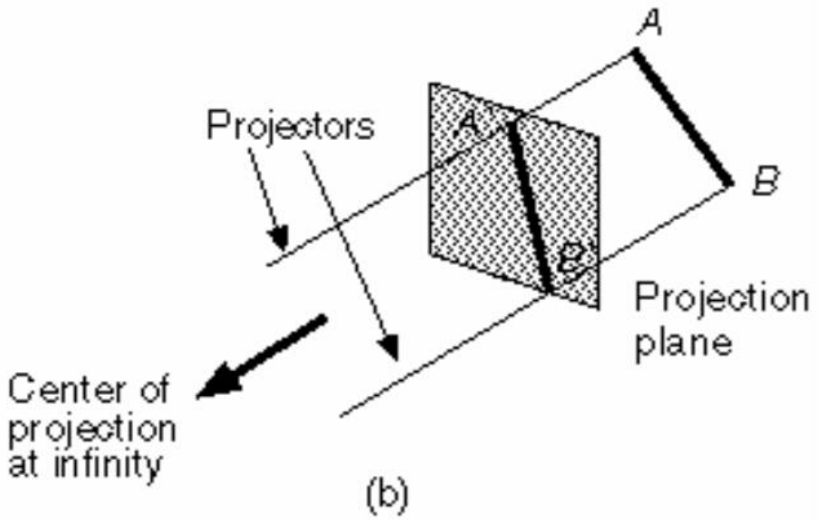


Main Classes of Planar Geometric Projections

a) Perspective Projection: determined by Center of Projection (COP)

b) Parallel Projection: determined by Direction of Projection (DOP)

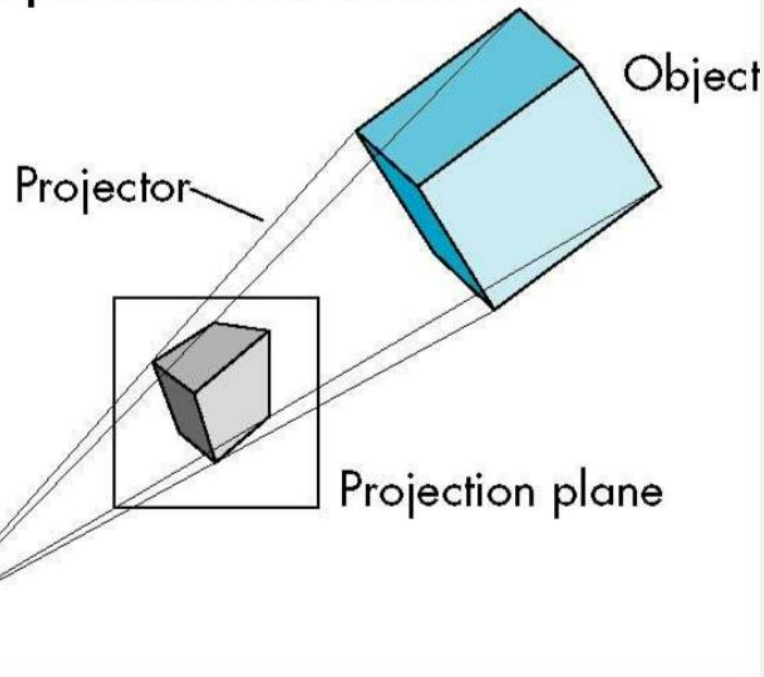




Main Classes of Planar Geometric Projections

Perspective Projection:

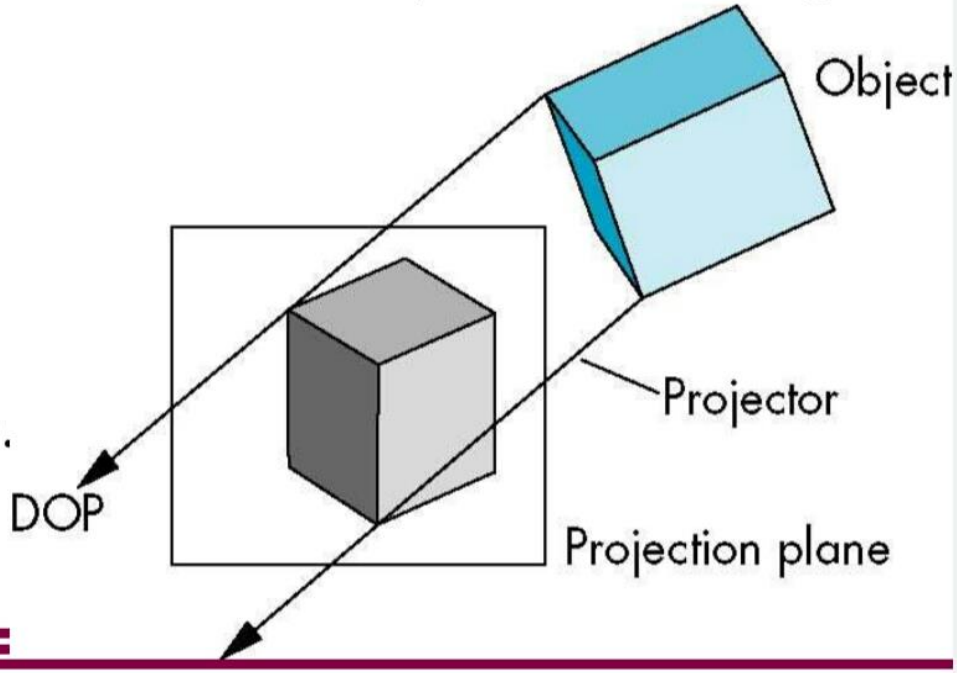
- The projectors meet at the centre of projection.
- COP corresponds to Centre of the lens in the camera.
- Origin of the camera frame for perspective views in the graphics system.
- Projection surface is plane.
- Projectors are straight lines.



Main Classes of Planar Geometric Projections

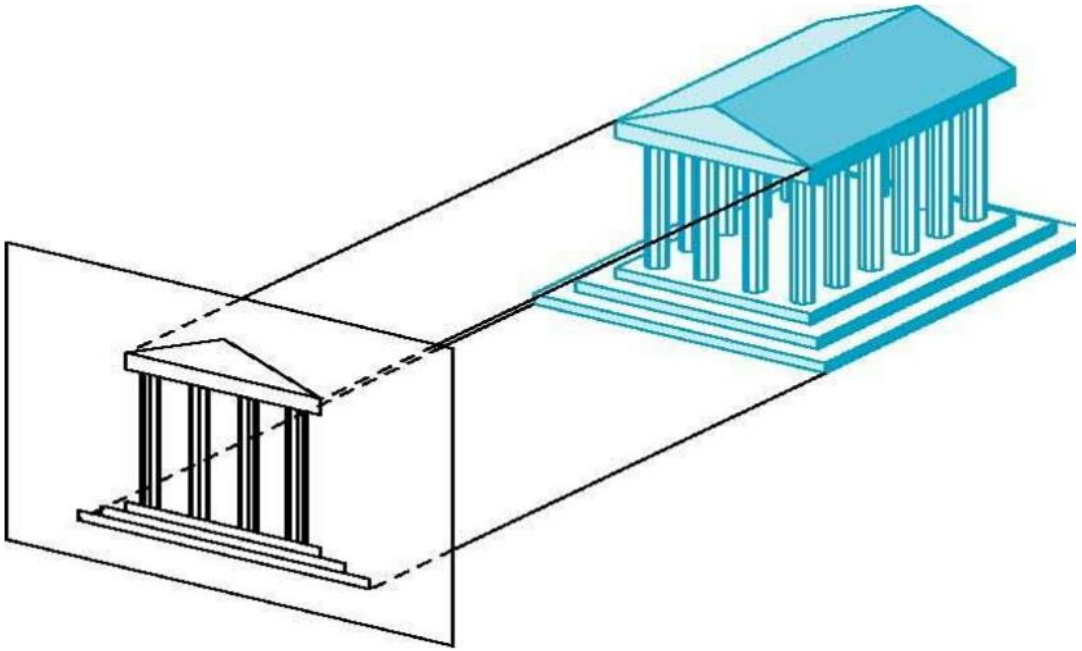
Parallel Projection:

- As COP moves to infinity, the projectors become parallel and COP is replaced by DOP(Direction of Projection).
- Size of the image remains as that of the object even though COP is infinitely far from the objects.
- Origin of camera frame lies in the projection plane.



Orthographic Projection

- Projectors are parallel to each other and orthogonal (perpendicular) to projection plane
- Preserves both distances and angles



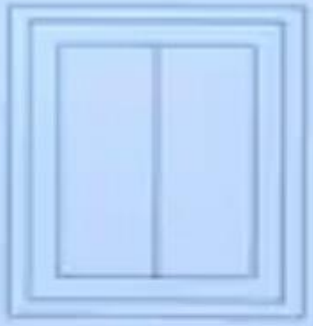
Multiview Orthographic Projection

- Projection plane parallel to principal face
- Usually form front, top, side views

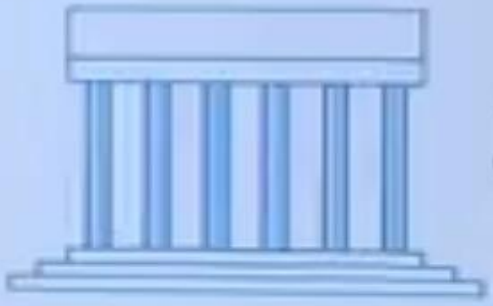
Isometric (Not Multiview Orthographic)

Front View

In CAD & architecture, we often display thee multiviews plus isometric



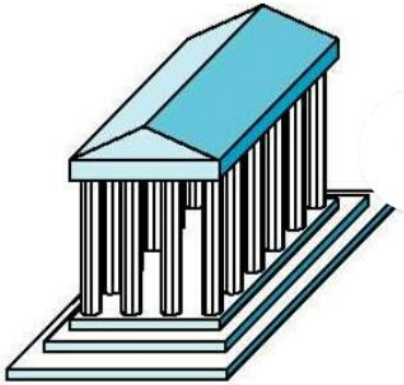
Top View

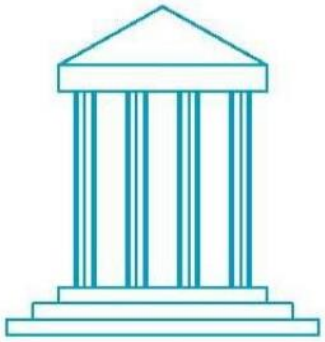


Side View

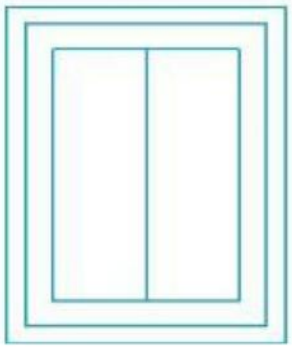
Multiview Projection

- Type of orthographic projection
- Projection plane **parallel** to principal face
- Usually form front, top, side views
- Preserve both distance and angle

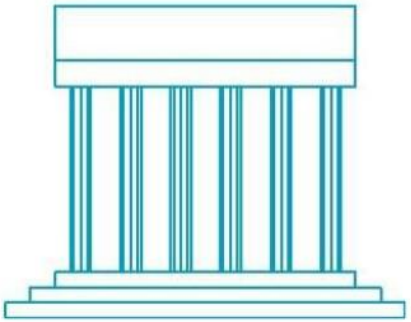




Front view



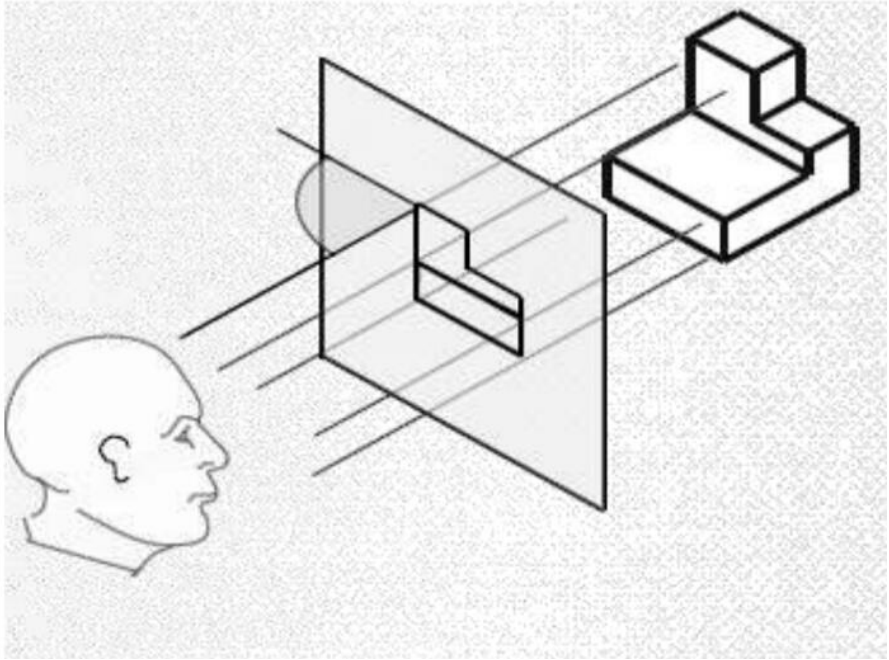
Top view



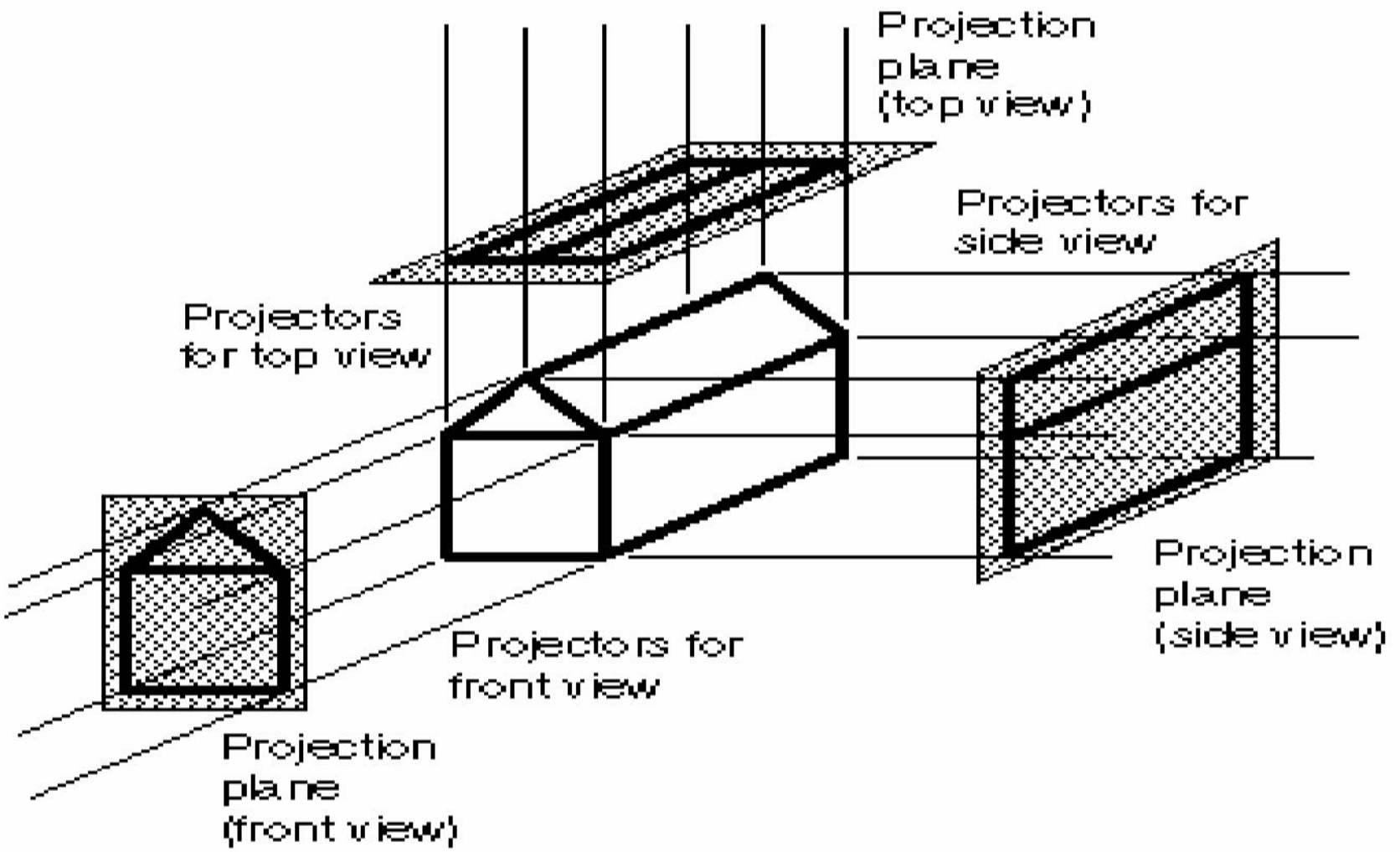
Side view

Multiview Projection

- One of the faces of the object to be drawn is placed parallel to the projection plane.
- One must draw several views of the object to portray it completely.
- The observer can only see one side at a time.



Multiview Projection



Multiview Projection

Advantages

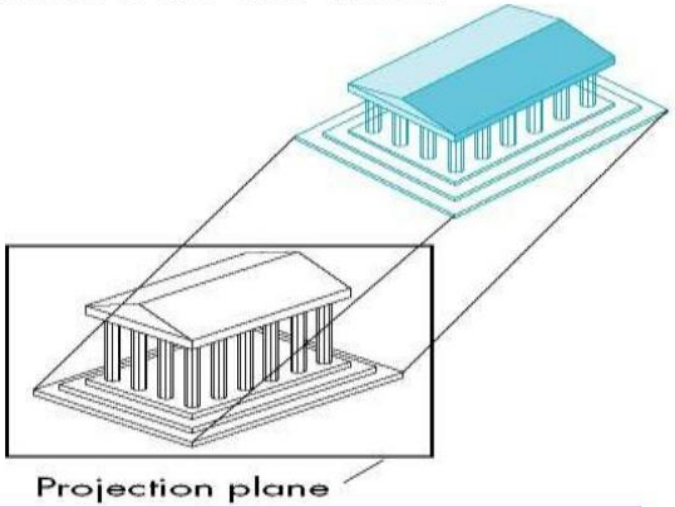
- Preserves both distances and angles
- All views are at same scale
- engineering drawings of machines, machine parts and working architectural drawings.

Disadvantages

- Cannot see what object really looks like because many surfaces hidden from view

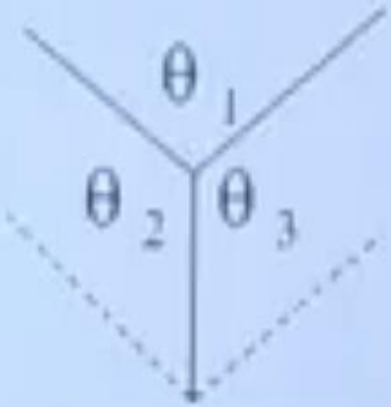
Axonometric Projections

- The projectors are still orthogonal to the projection plane, but the projection plane can have any orientation with respect to the object.
- Used to create a pictorial drawing of an object, where the object is rotated along one or more of its axes relative to the plane of projection.



Axonometric 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

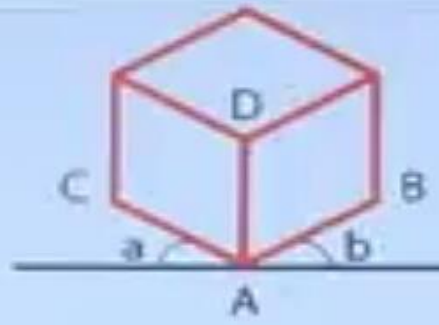


Types of Axonometric Projections

Dimetric projection

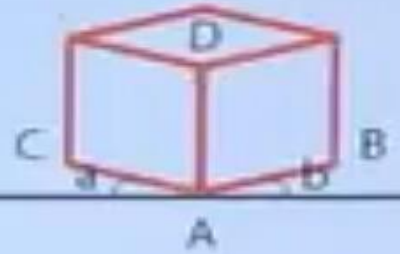
- Projection plane is placed symmetrically with respect to two of the principal faces.
- The angles between the projection of the axes , two of the three to be equal.
- To draw the outline of an object in dimetric projection, two scales are required.

Types of Axonometric Projections



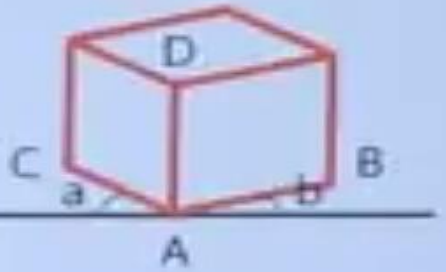
$a = b = 30$
$AB = AC = AD$

Isometric Projection



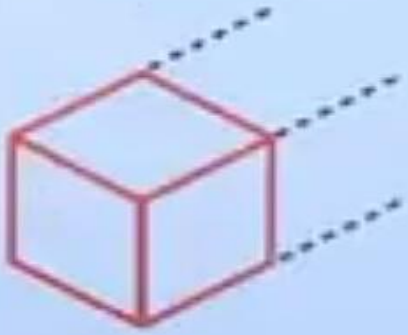
$a = b$
$AB = AC \neq AD$

Dimetric Projection

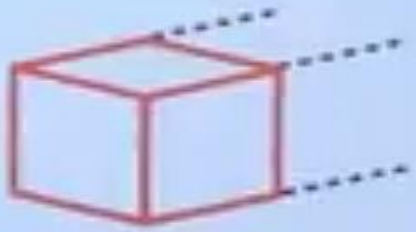


$a \neq b$
$AB \neq AC \neq AD$

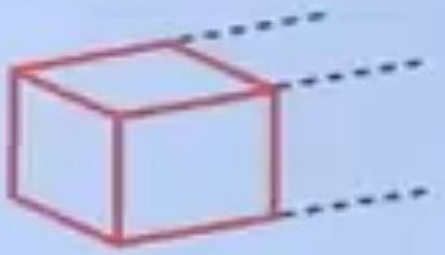
Trimetric Projection



Isometric Projection

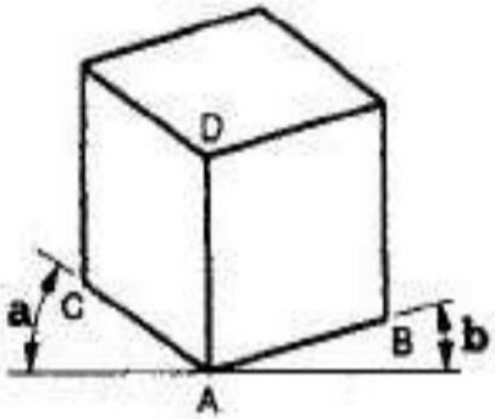


Dimetric Projection



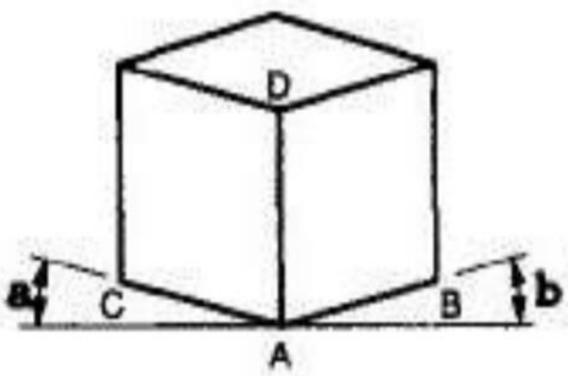
Trimetric Projection

Types of Axonometric Projections



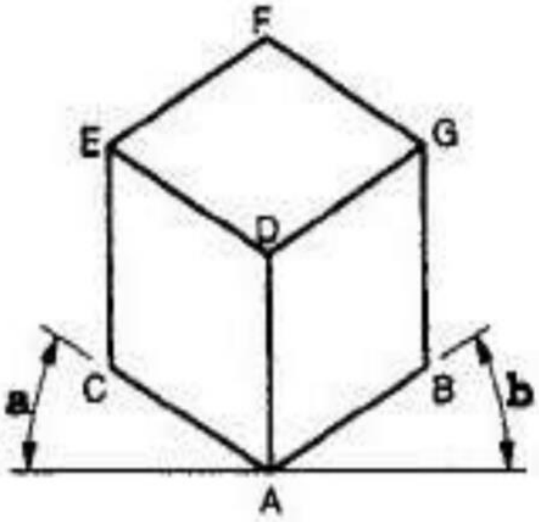
Note: $a \neq b$ and
$AB \neq AC \neq AD$

Trimetric Projection



Note: $a = b$ and
$AB = AC \neq AD$

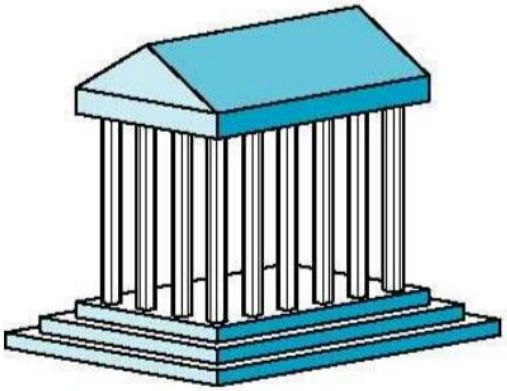
Dimetric Projection



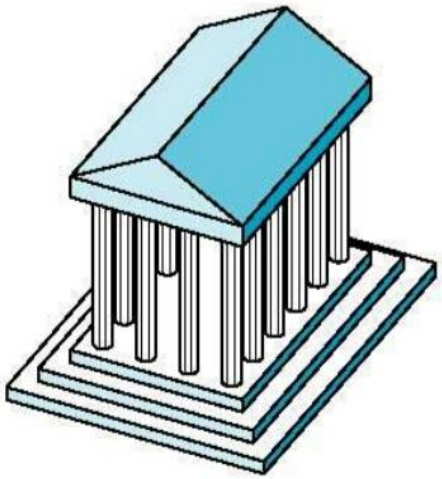
Note: $a = b = 30°$ and
$AB = AC = AD$

Isometric Projection

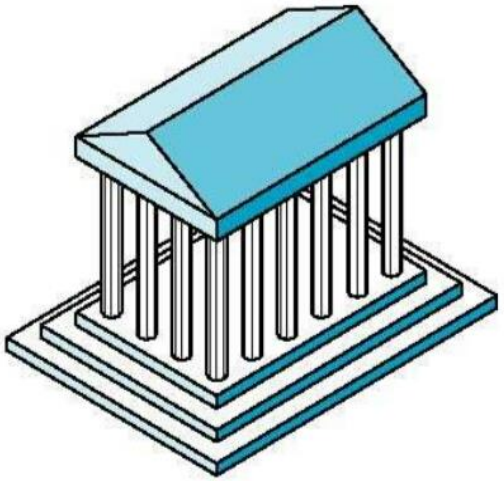
Types of Axonometric Projections



Dimetric



Trimetric



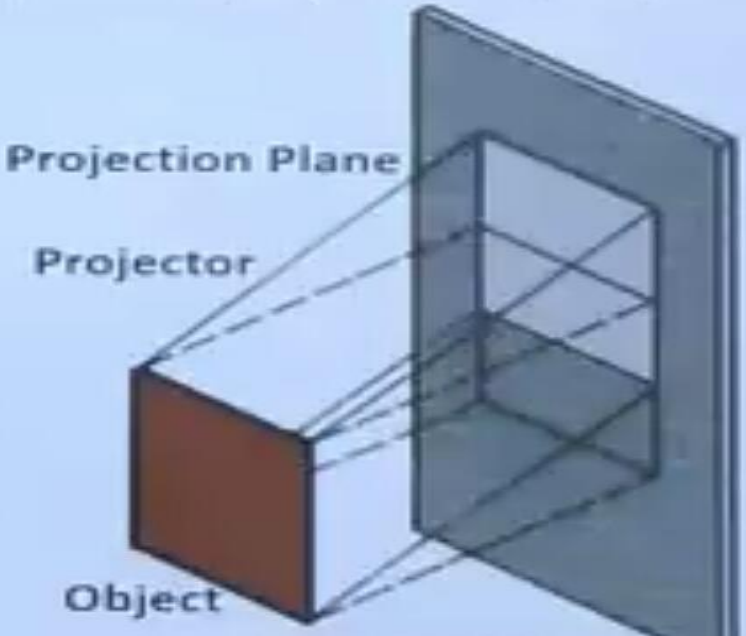
Isometric

Advantages & Disadvantages

- Lines are scaled (foreshortened) but can find scaling factors
- Lines preserved but angles are not
  - Projection of a circle in a plane not parallel to the projection plane is an ellipse
- Can see three principal faces of a box-like object
- Some optical illusions possible
  - Parallel lines appear to diverge
- Does not look real because far objects are scaled the same as near objects
- Used in CAD applications

Oblique Projection

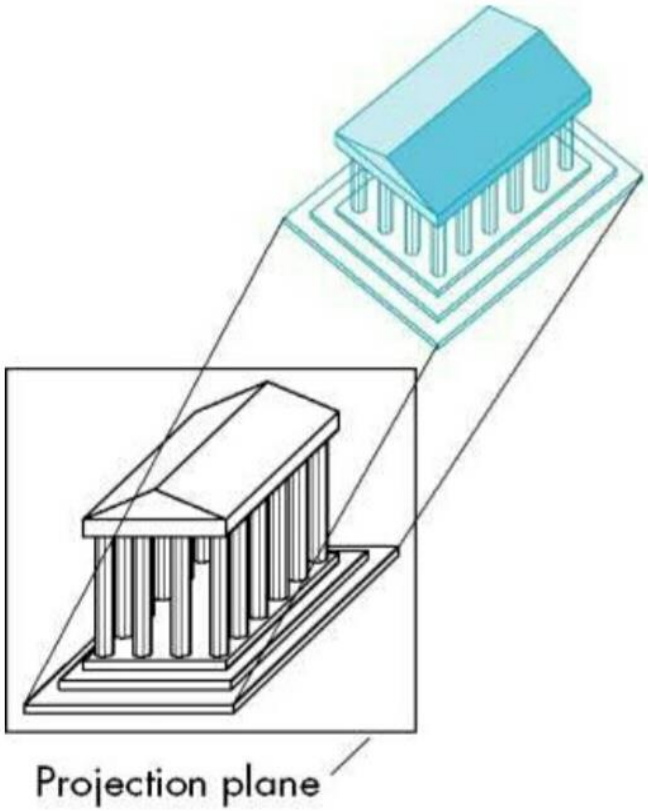
- Arbitrary relationship between projectors and projection plane
- Projectors are parallel to each other but not perpendicular to projection plane
- An oblique projection shows front and top surfaces that include the three dimensions of height, width and depth.
- The front or principal surface of an object (the surface toward the plane of projection) is parallel to the plane of projection.
- Effective in pictorially representing objects.

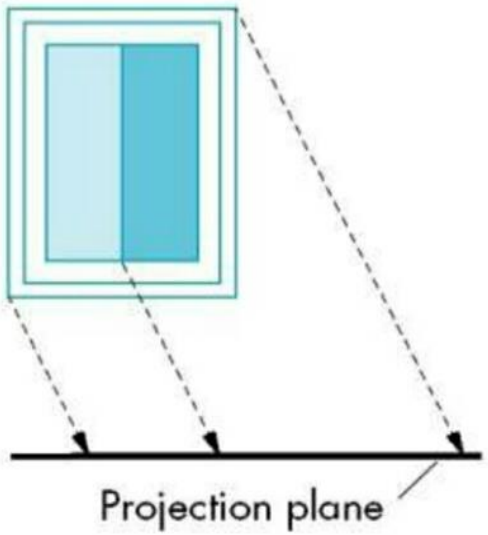


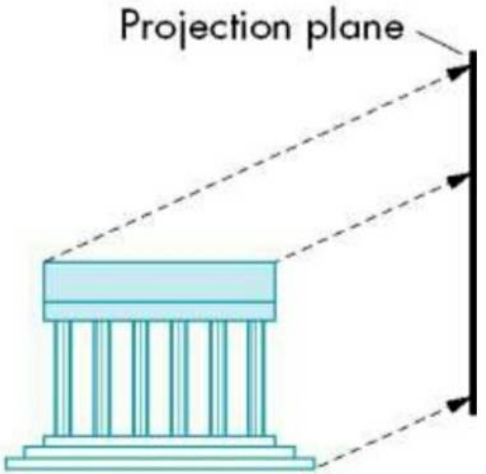
OBLIQUE PROJECTIONS

Oblique Projection

Arbitrary relationship between projectors and projection plane

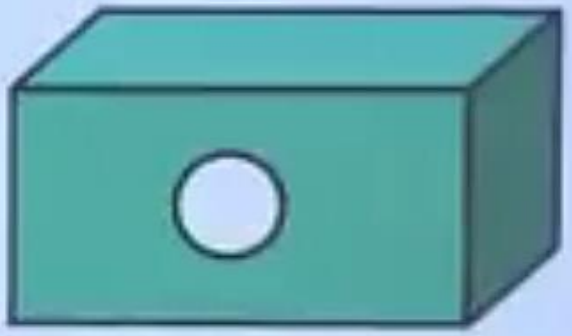






Advantages & Disadvantages

- 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

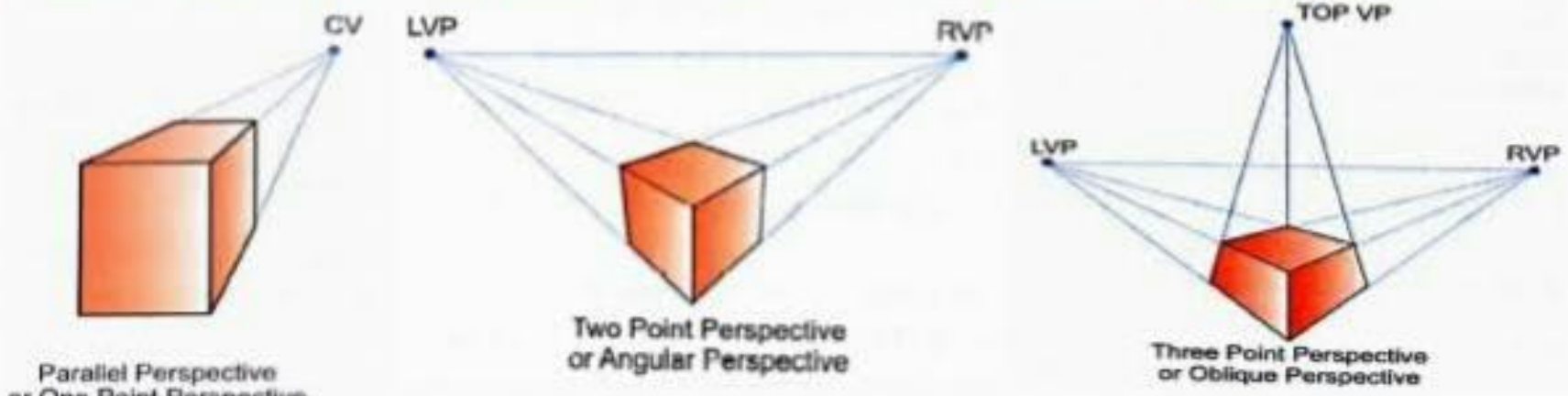


- In physical world, cannot create with simple camera; possible with bellows camera or special lens (architectural)

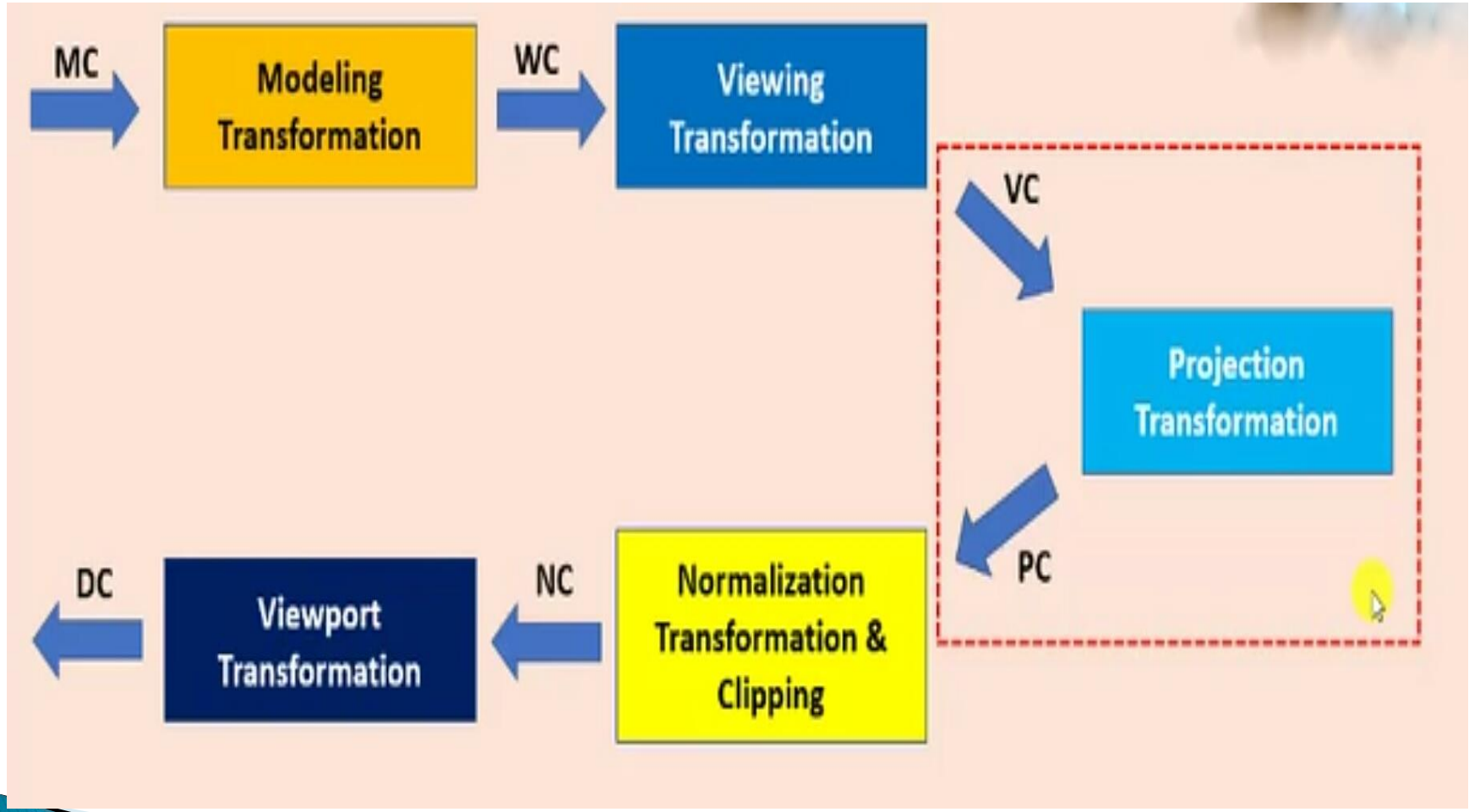
Classification of Perspective Projections

Perspective projections can be broadly classified into three categories.

1. Parallel perspective or single point perspective.
2. Angular perspective or two point perspective.
3. Oblique perspective or three point perspective.



3D viewing Pipeline



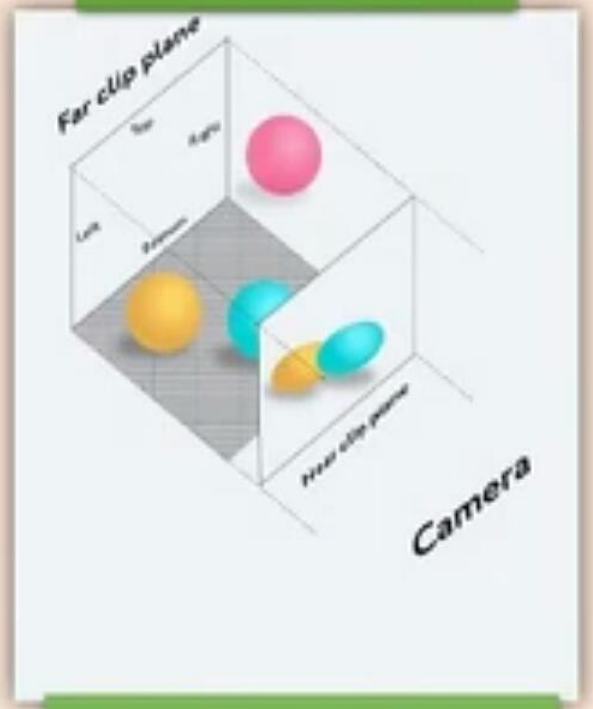
Simple projections

- Perspective Projection
- Parallel Projection

Projection Transformations

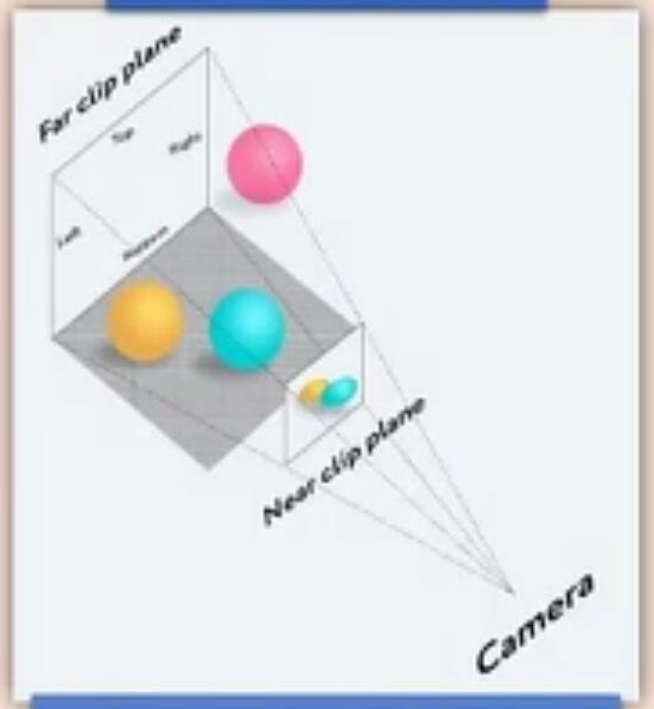
Next Object Descriptions to be Projected to View Plane

Parallel Lines



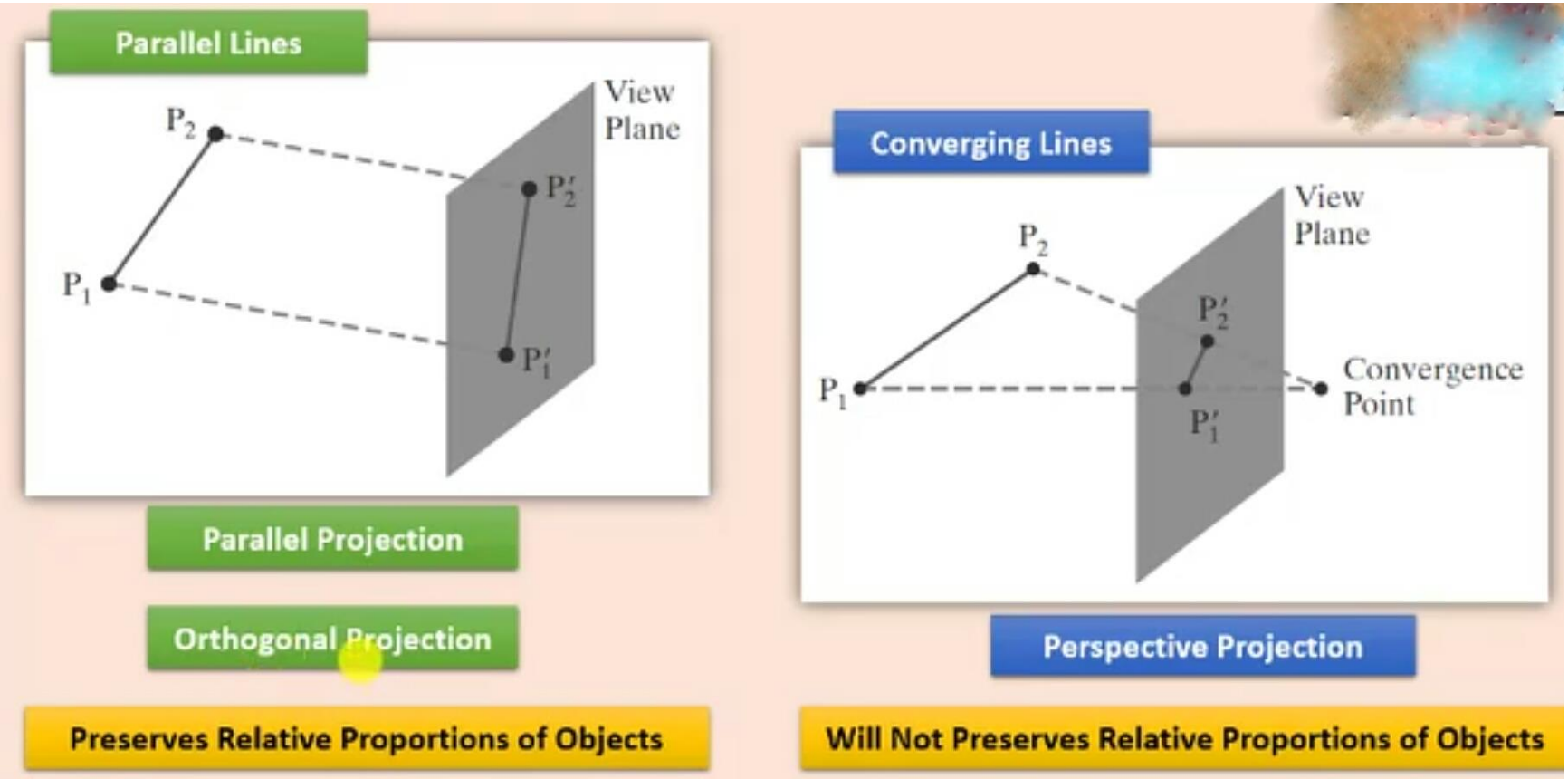
Parallel Projection

Converging Lines



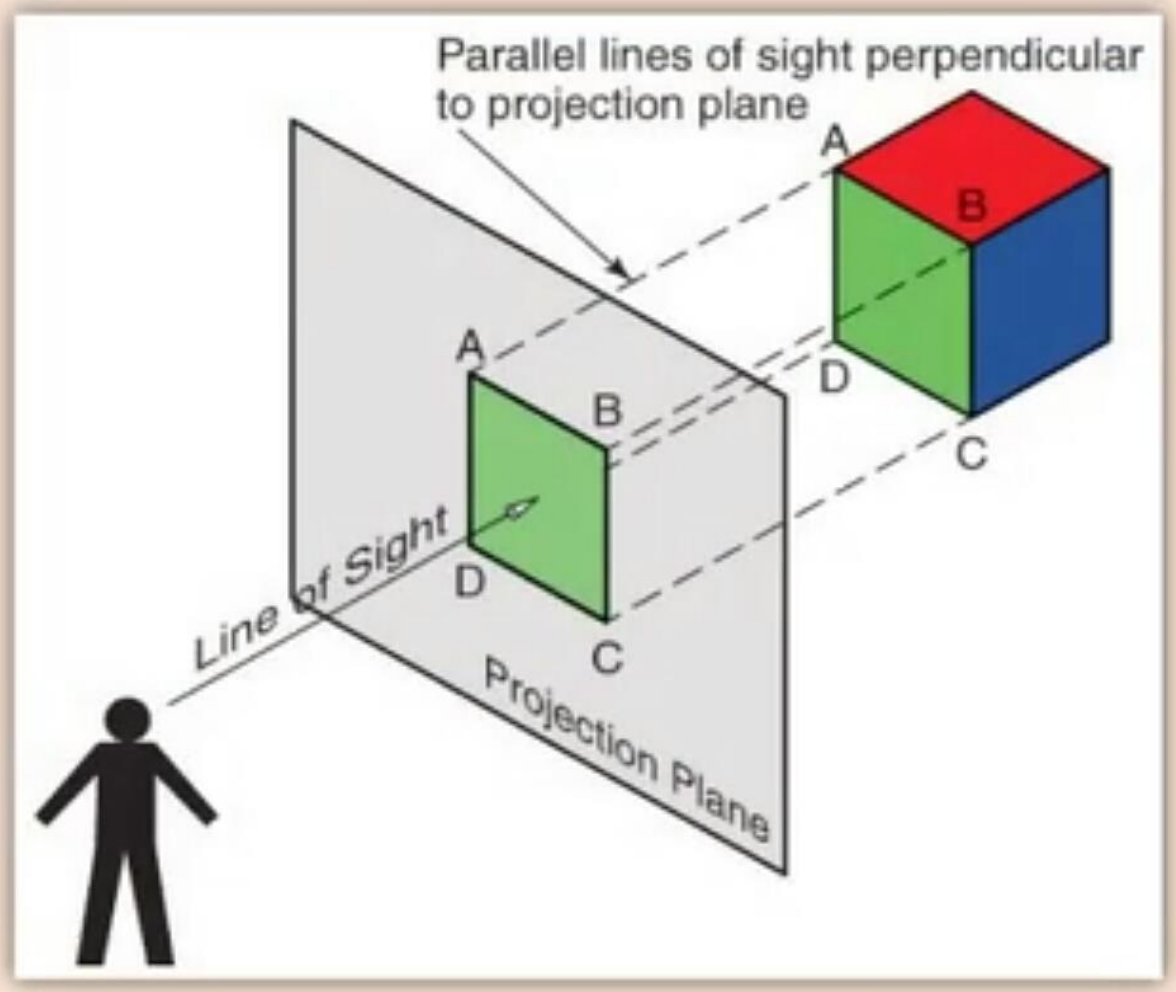
Perspective Projection

Projection Transformation

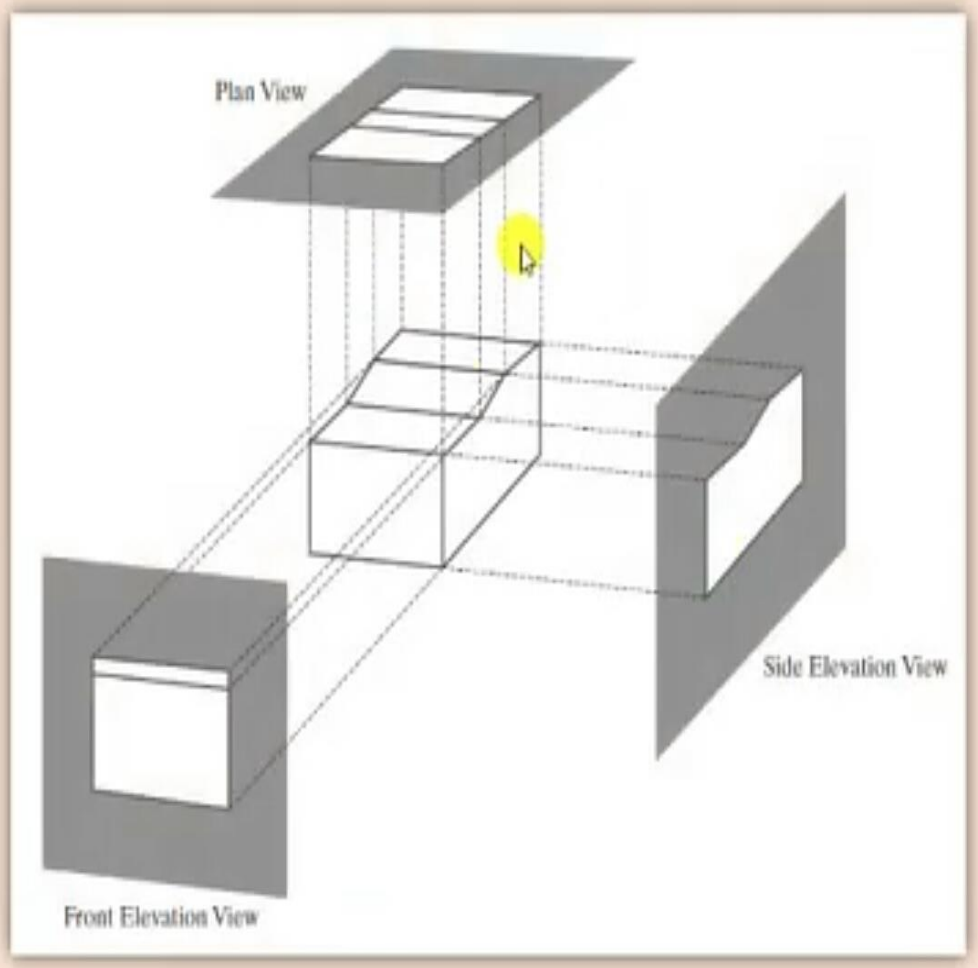


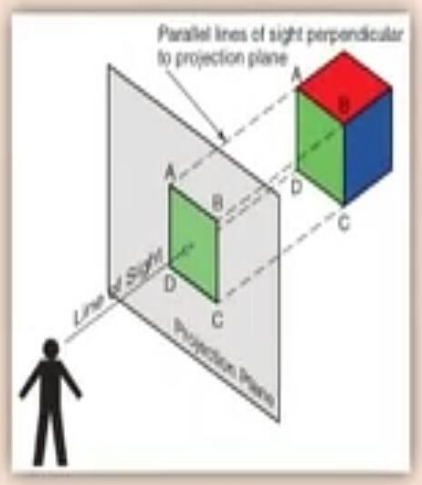
Orthogonal Projection

Object to a View Plane along Lines that all are Parallel to Vector N

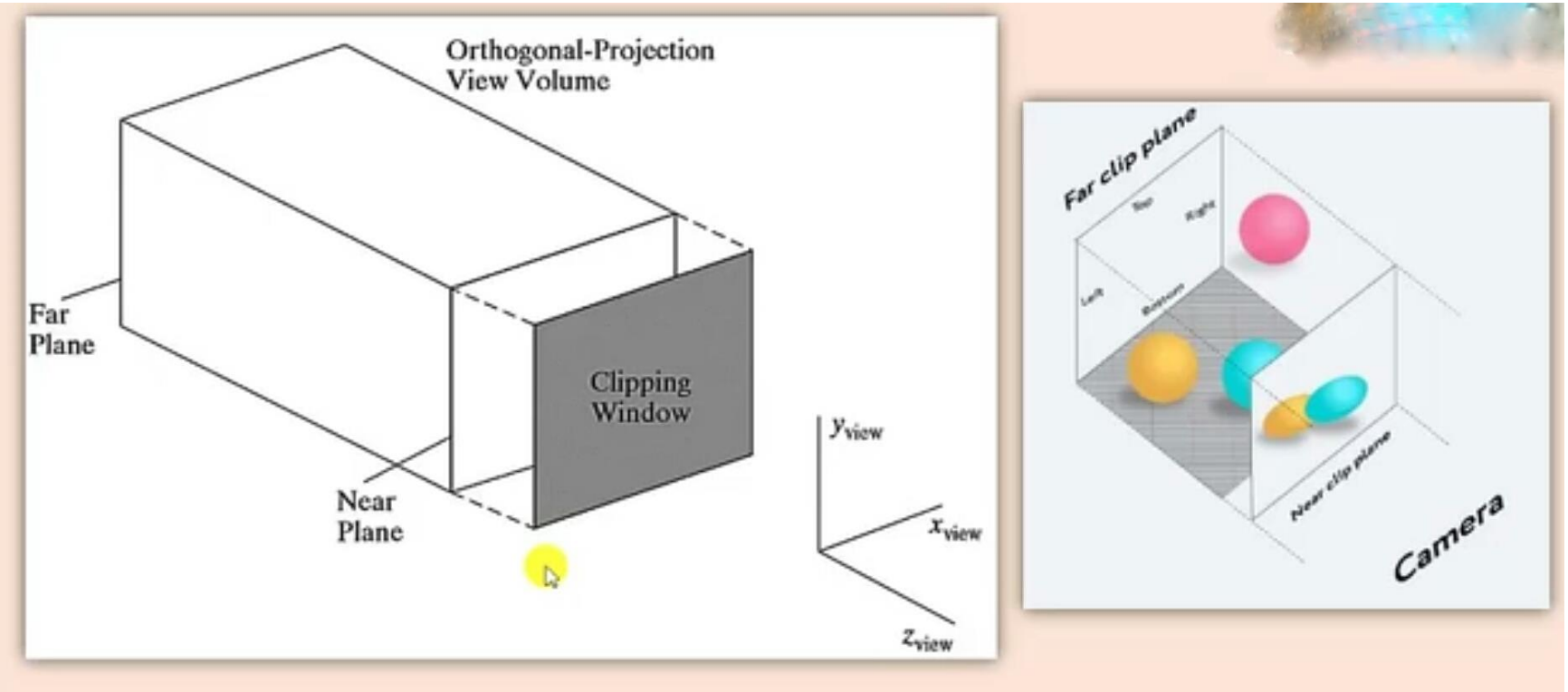


Orthogonal Projection

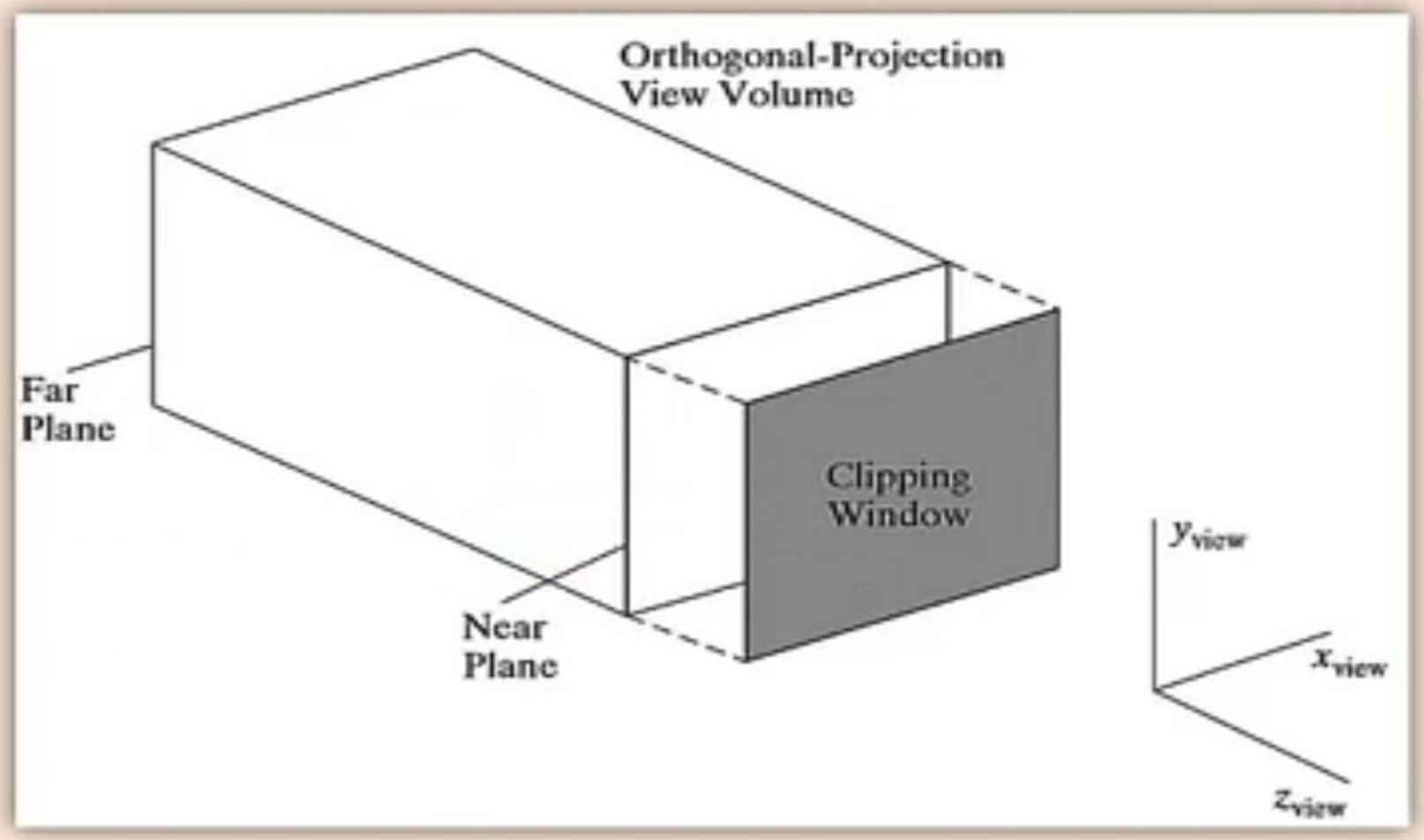




Orthogonal Projection

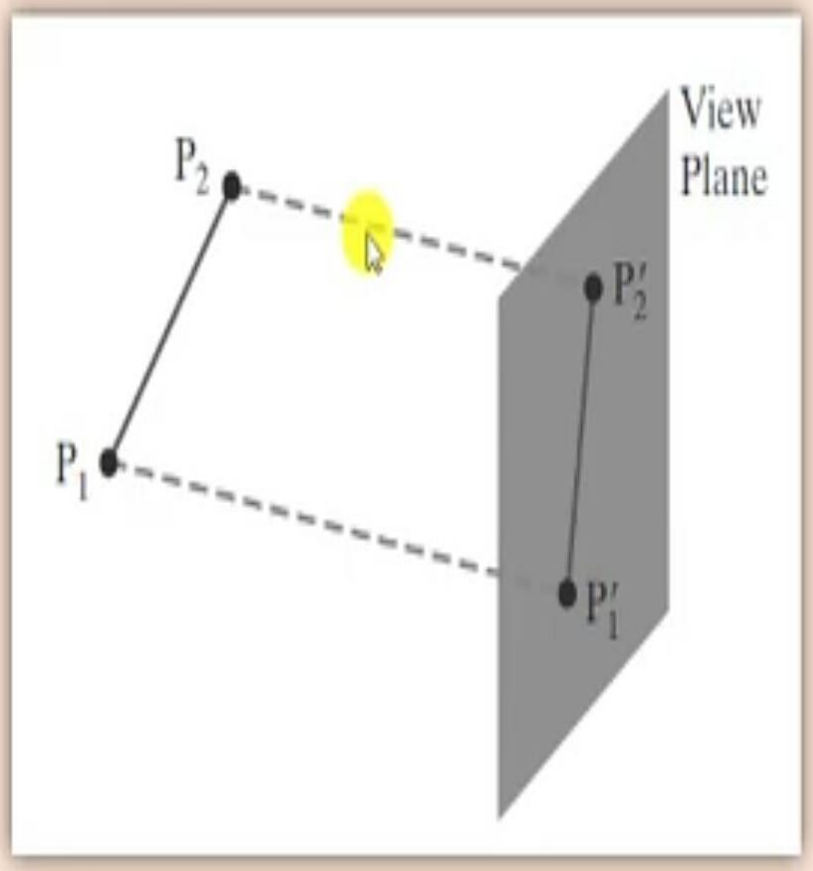


Orthogonal-Projection View Volume

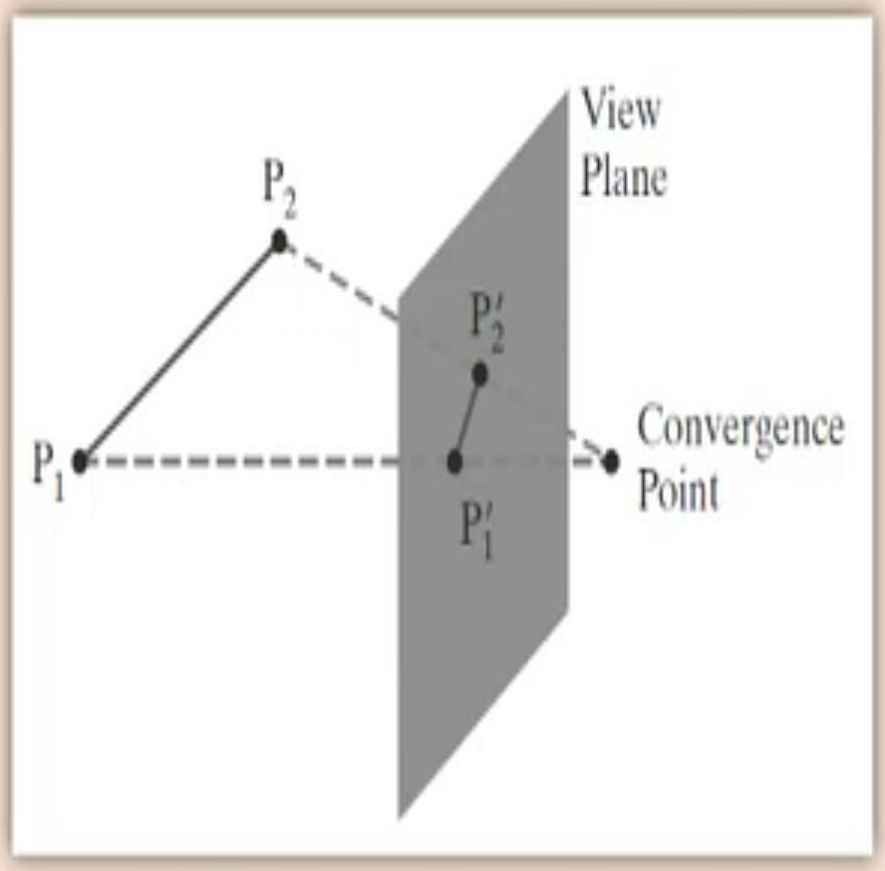


Rectangular Parallelepiped

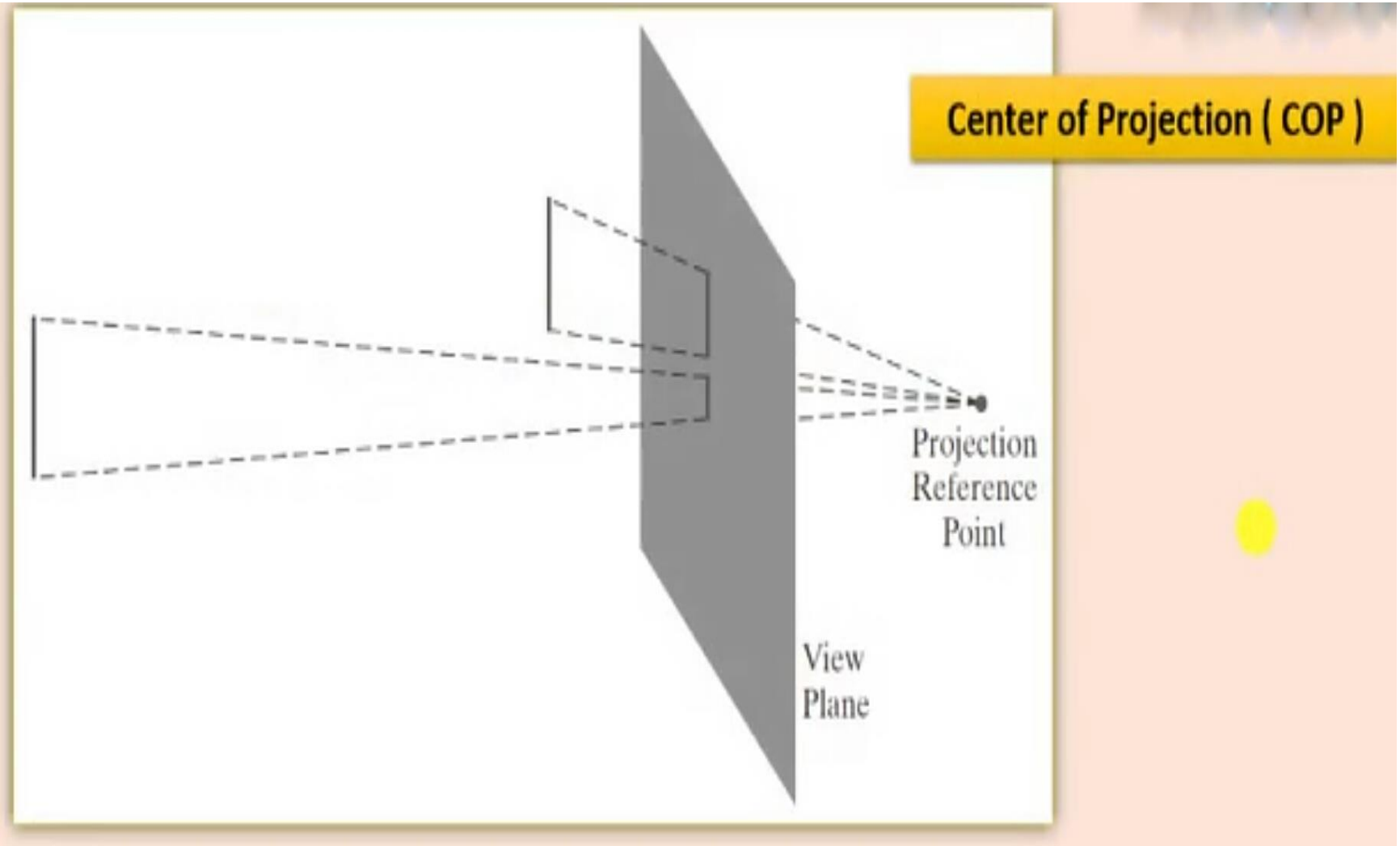
Orthogonal Projection



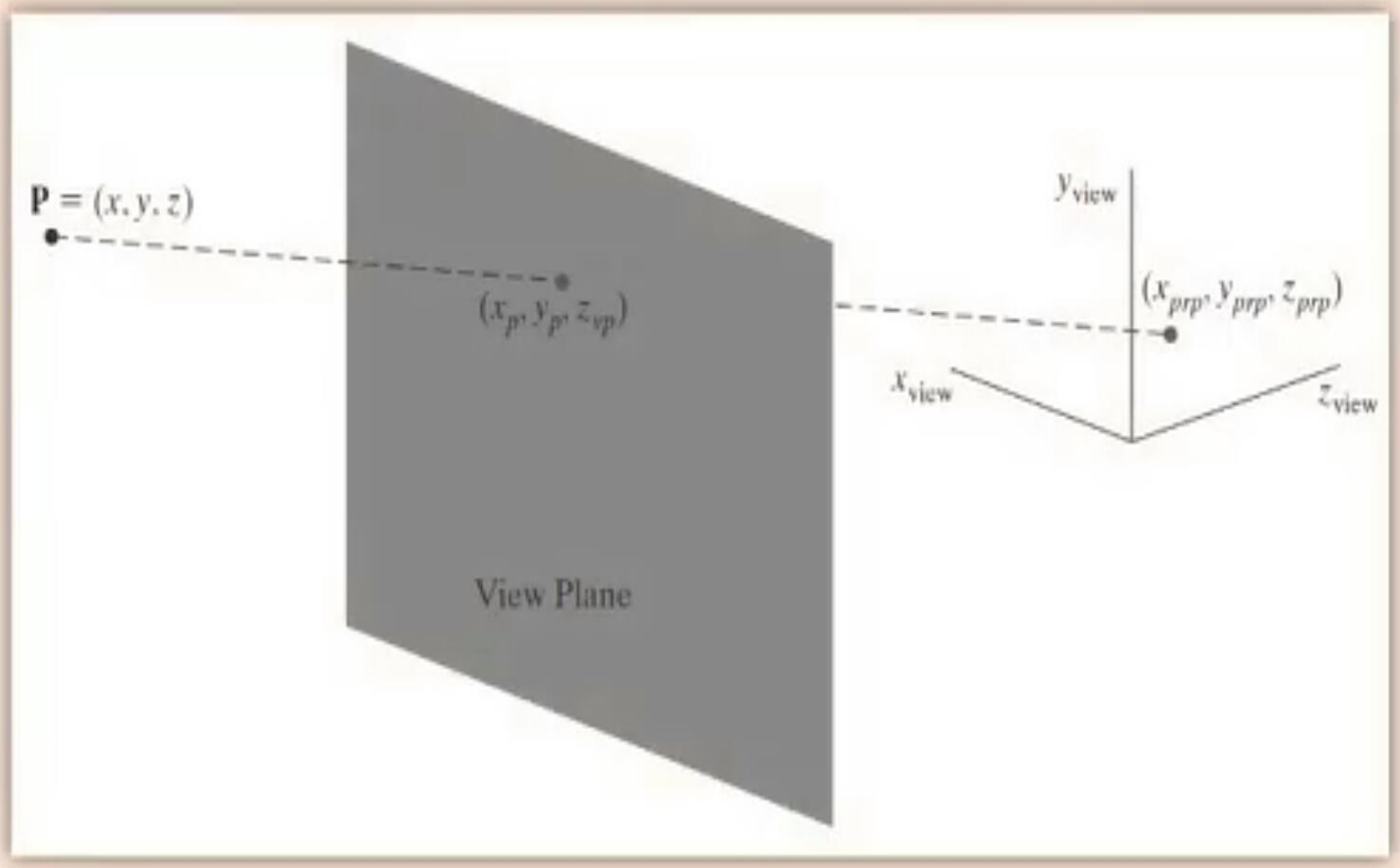
Perspective Projection



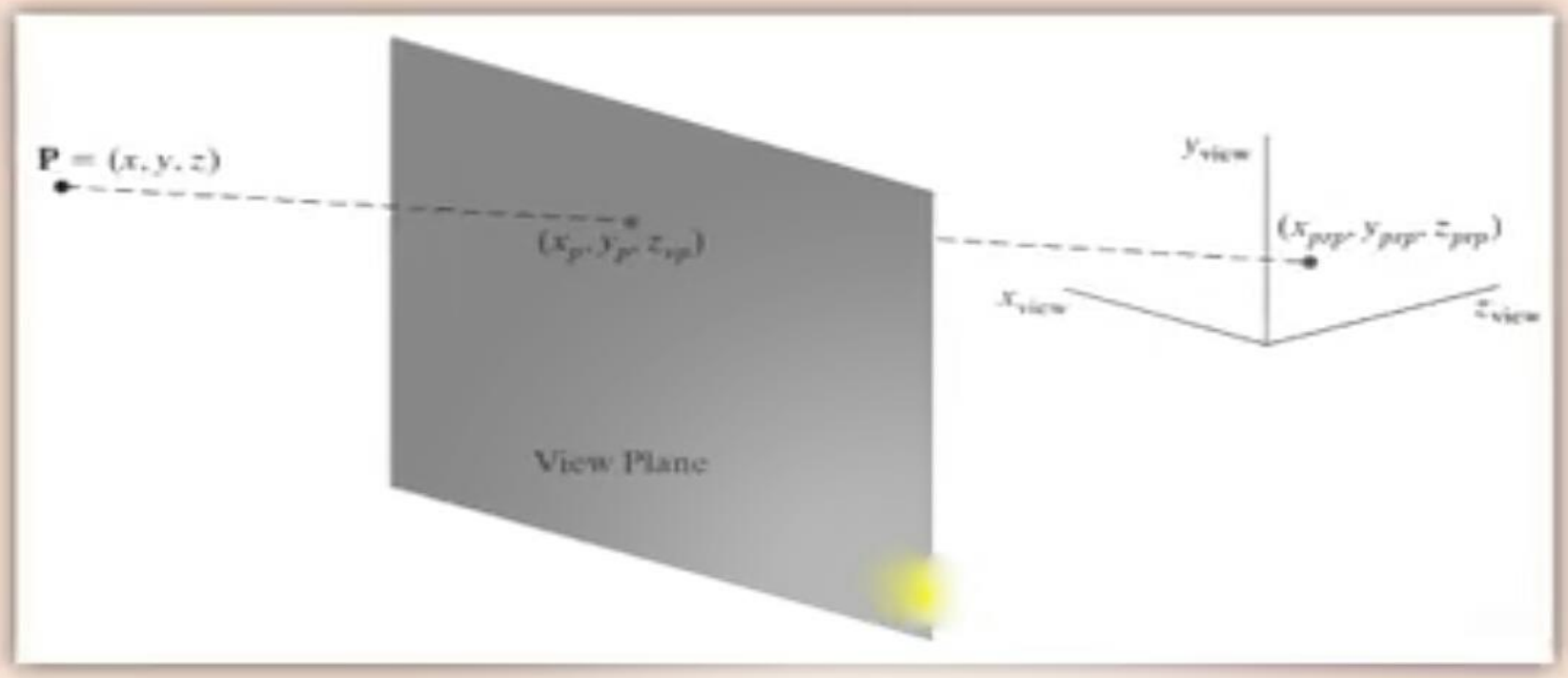
Perspective Projection



Perspective-Projection Transformation



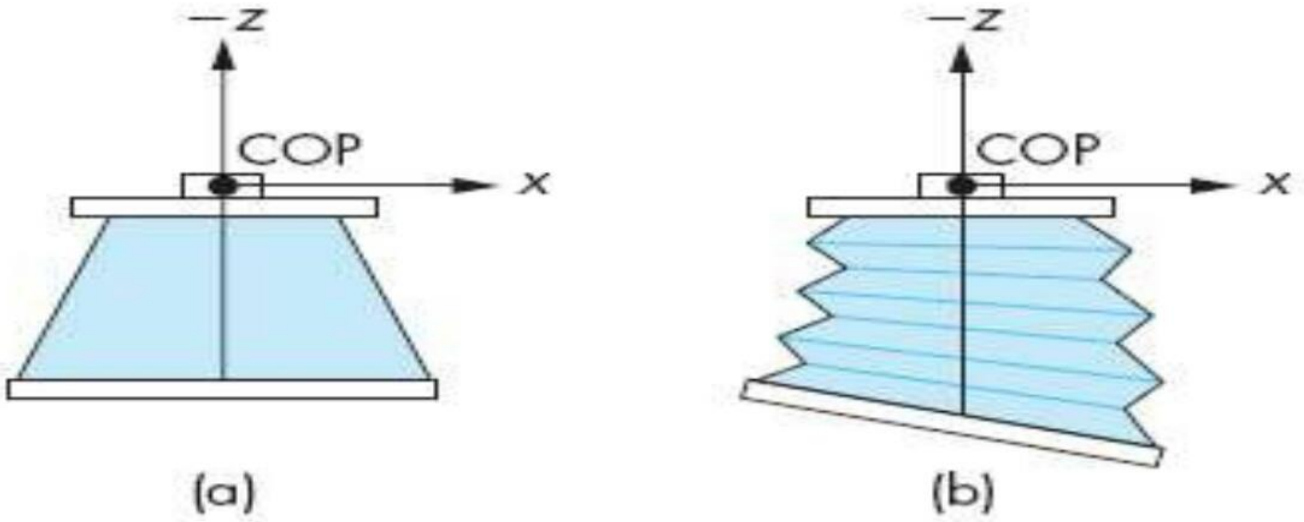
Perspective-Projection Transformation



$$x' = x - (x - x_{prp})u$$
$$y' = y - (y - y_{prp})u \qquad 0 \leq u \leq 1$$
$$z' = z - (z - z_{prp})u$$

Perspective projections

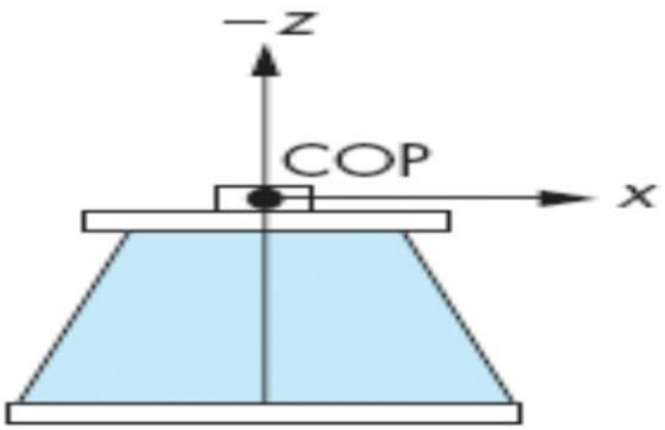
- Suppose that we are in the camera frame with the camera located at the origin, pointed in the negative *z-direction*.
- *Two possibilities*



Perspective projections

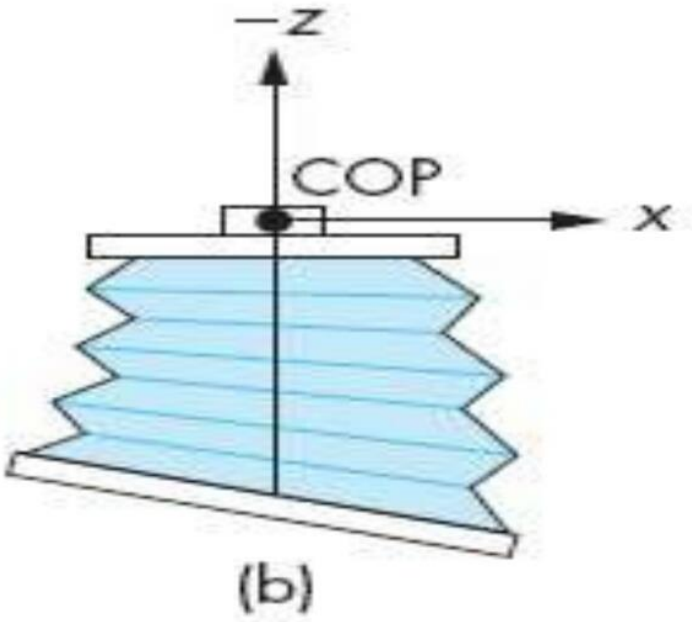
- Back of the camera parallel to front face

  Back of the camera is orthogonal to the *z-direction and is parallel* to the lens. (human visual system and



Perspective projections

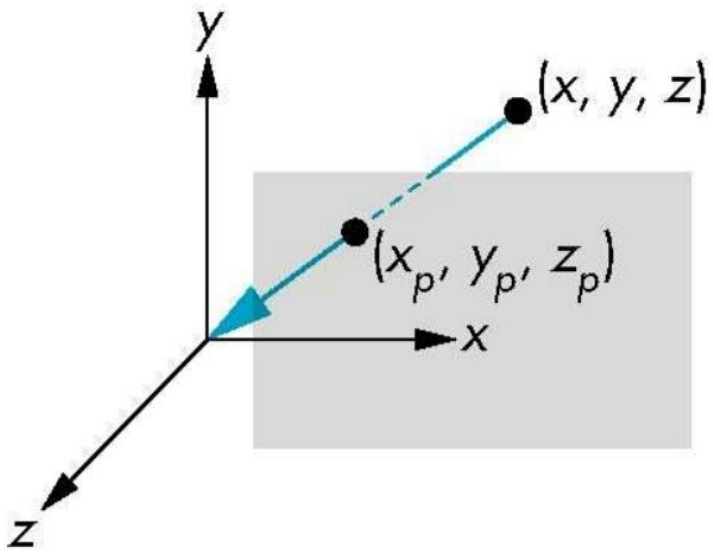
- Back of the camera not parallel to front face the back of the camera can have any orientation with respect to the front.



(b)

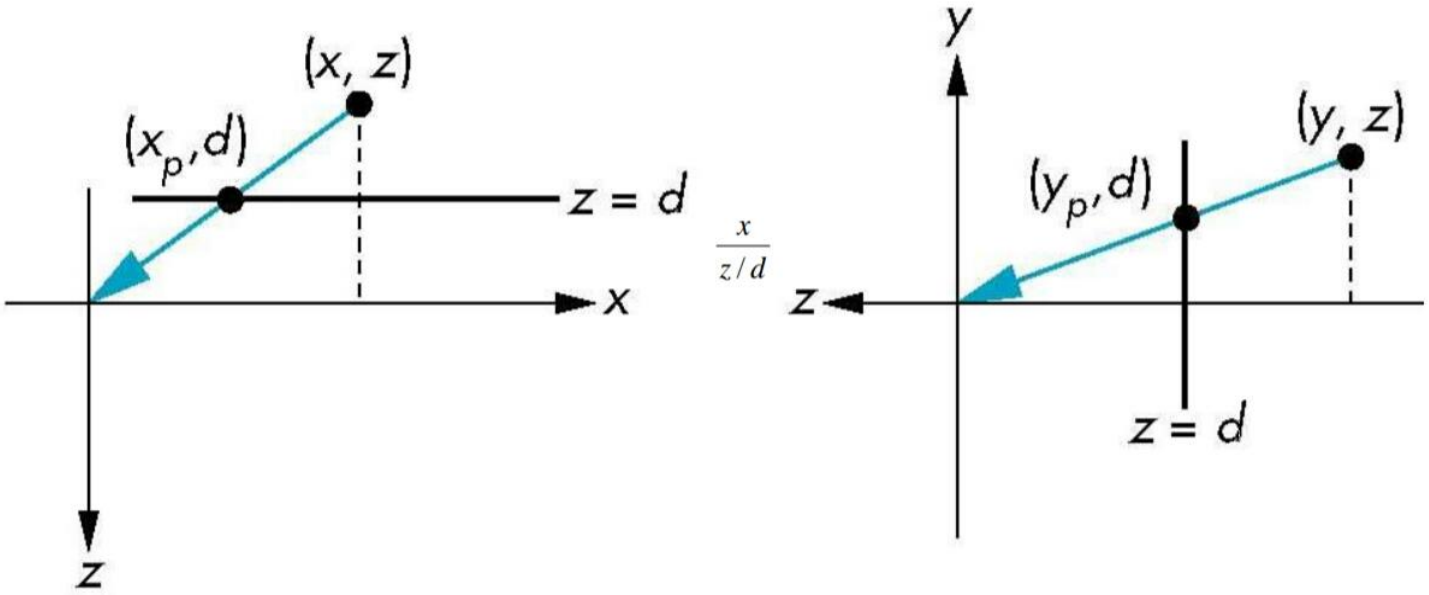
Simple Perspective

- Center of projection at the origin
- Projection plane $z = d$, $d < 0$



Perspective Equations

Consider top and side views



$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

Homogeneous Coordinate Form

consider $\mathbf{q} = \mathbf{M}\mathbf{p}$ where $\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

Perspective Division

- However $w \neq 1$, so we must divide by w to return from homogeneous coordinates
- This *perspective division* yields

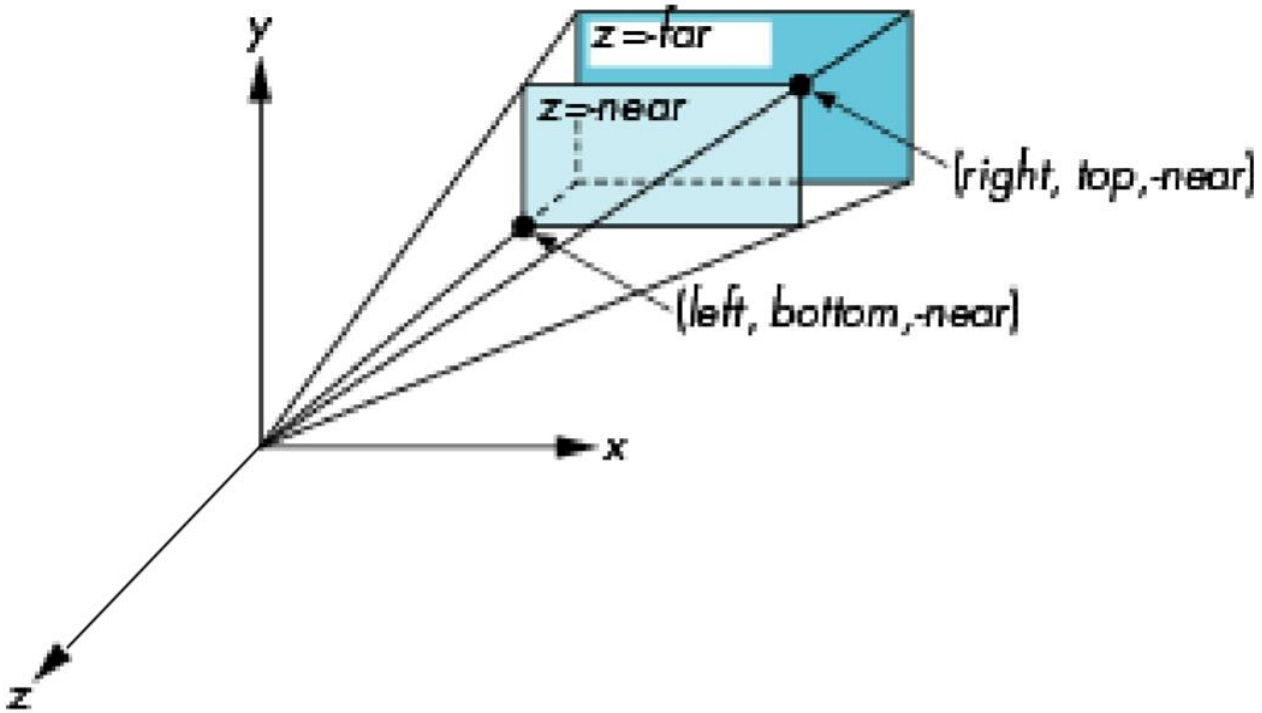
$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

the desired perspective equations

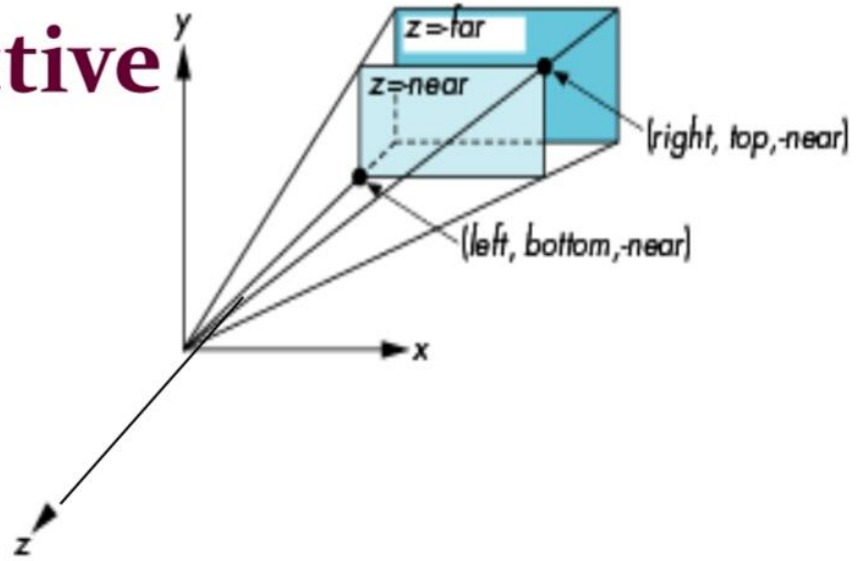
- We will consider the corresponding clipping volume with the OpenGL functions

OpenGL Perspective

```
glFrustum(left,right,bottom,top,near,far)
```



OpenGL Perspective



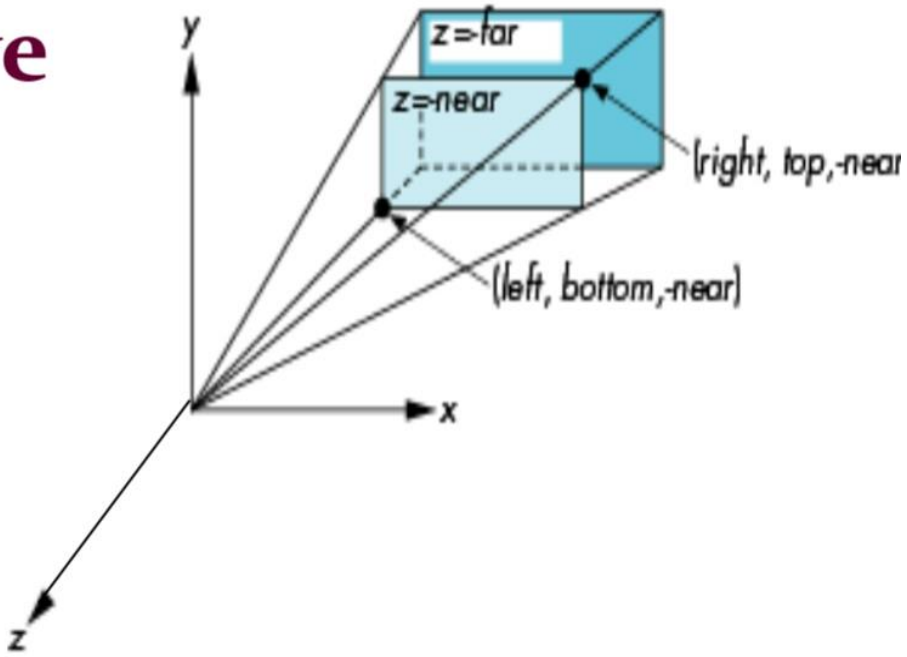
left and right

Specify the coordinates for the left and right vertical clipping planes.

bottom and top

Specify the coordinates for the bottom and top horizontal clippin planes.

OpenGL Perspective



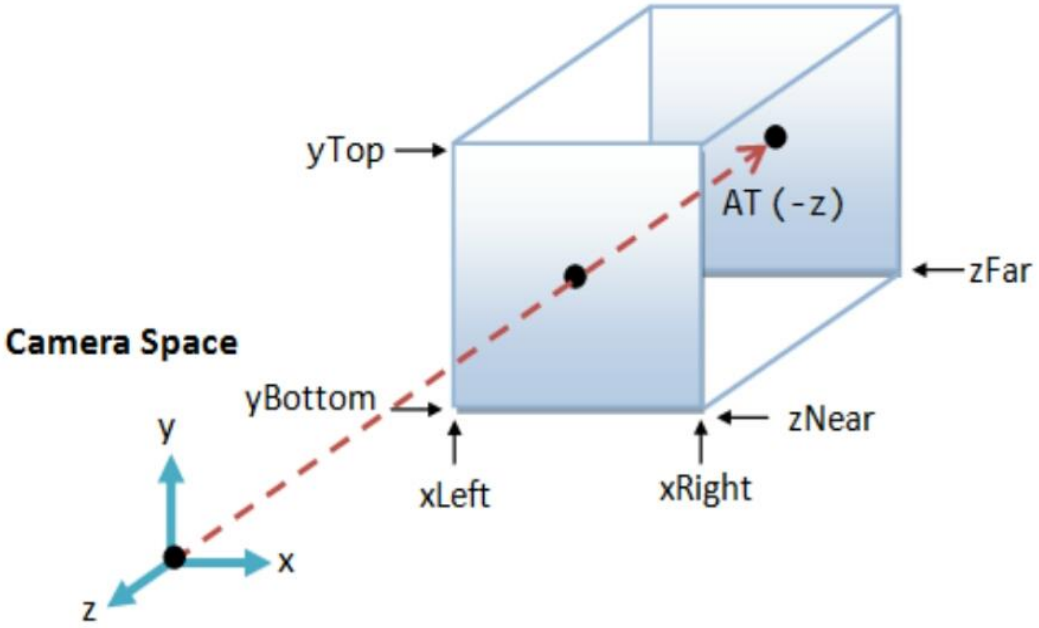
near and far

Measured from the COP (origin in eye coordinates) to front and back clipping planes.

Specify the distances to the near and far depth clipping planes. Both distances must be positive.

OpenGL Orthogonal Viewing

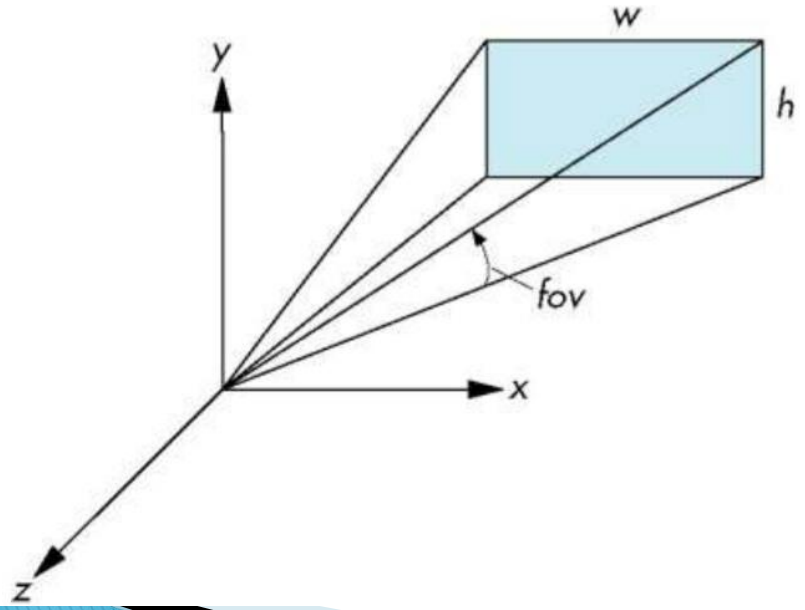
```
glOrtho(left,right,bottom,top,near,far)
```



Orthographic Projection: Camera positioned infinitely far away at $z = \infty$

Using Field of View

- With **glFrustum** it is often difficult to get the desired view
- **gluPerpective(fovy, aspect, near, far)** often provides a better interface

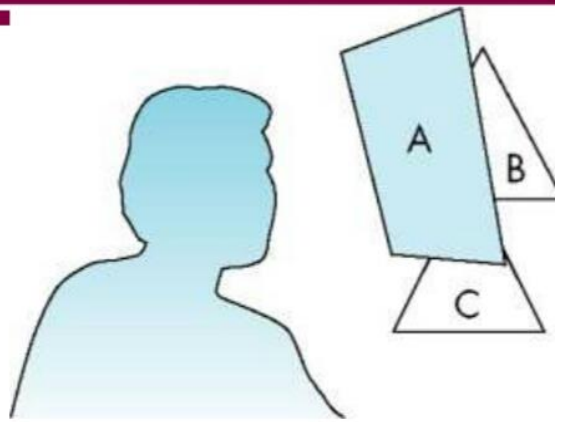


aspect = w/h

Normalization

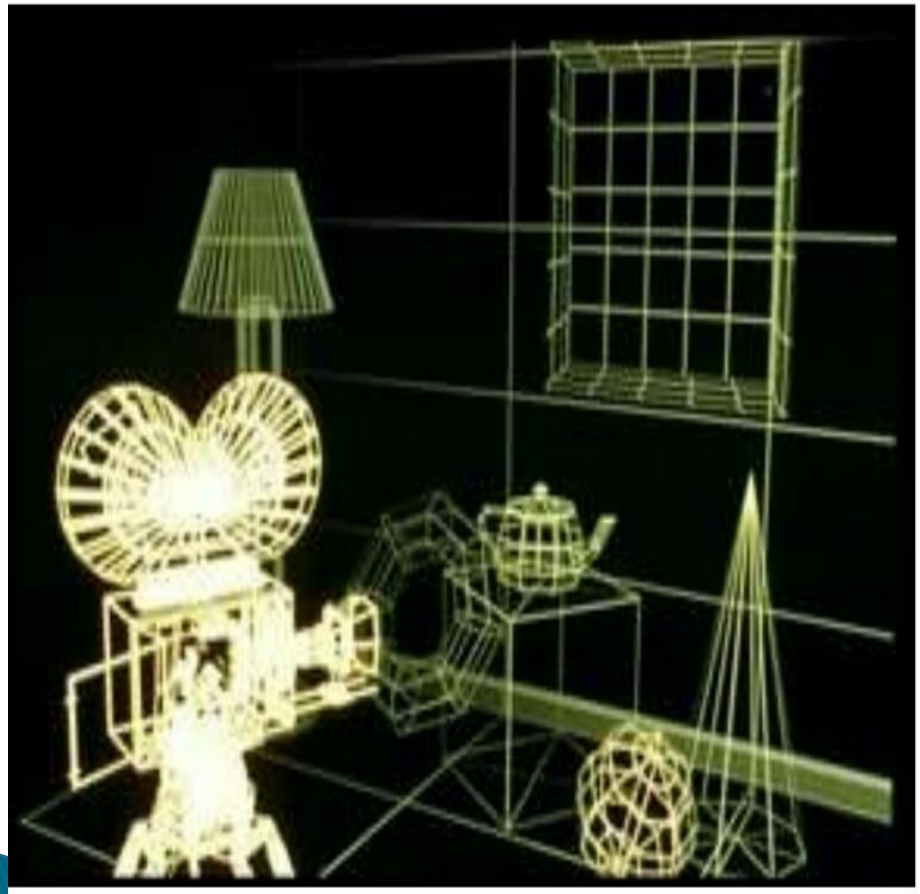
- Rather than derive a different projection matrix for each type of projection, we can convert all projections to orthogonal projections with the default view volume
- This strategy allows us to use standard transformations in the pipeline and makes for efficient clipping

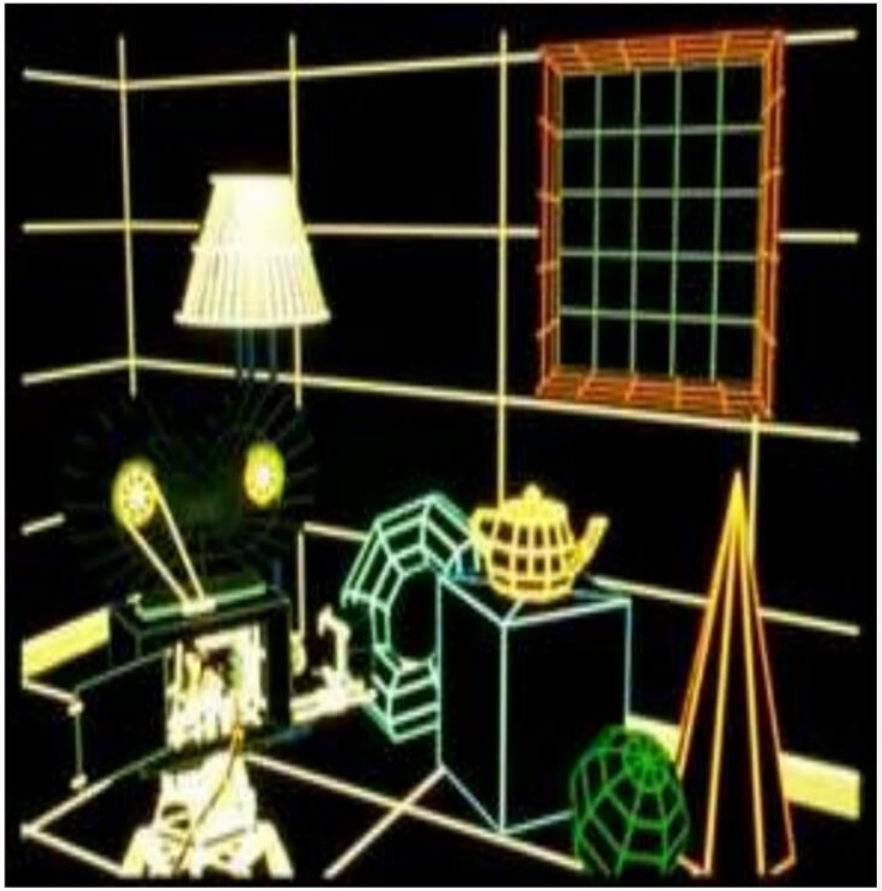
Hidden Surface Removal



- A correct rendering requires correct visibility calculations.
- Correct visibility: When multiple opaque polygons cover the same screen space, only the closest one is visible by removing the other hidden surfaces.

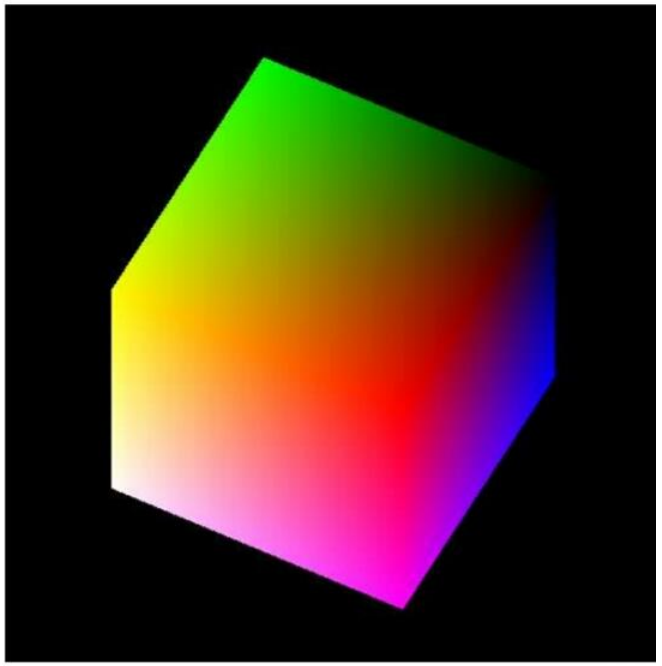
Hidden Surface Removal





Hidden Surface Removal

- Consider rotating cube program.
- From the perspective of the computer Graphics, all 6faces of the cube have been specified and travel down the graphics pipeline.
- Graphics system must be careful about which surface it displays.



Hidden Surface Removal

- We seek algorithms that either remove those surfaces that should not be visible to the viewer called **Hidden Surface Removal Algorithms**.

OR

- Find which surfaces are visible called **Visible Surfaces Algorithms.**
- Open GL has a particular algorithm called **z-buffer algorithm.**

Hidden Surface Removal

- Object Space Algorithms
- Image Space Algorithms

Hidden Surface Removal

Object Space Algorithms

- Attempt to order the surfaces of the objects such that rendering surfaces in particular order provides the correct image.
- This does not work well with the pipeline architecture.

Object space algorithms do their work on the objects them selves before they are converted to pixels in the frame buffer.

There solution of the display device is irrelevant here as this calculation is done at the mathematical level of the objects

Pseudocode…

- for each object A in the scene
- determine which parts of object A are visible
- draw these parts in the appropriate color

Image Space Algorithms

Works as a part of the Projection process.

Z-buffer algorithm fits in well with rendering pipeline in most of the graphics systems as we can save partial information as each object is rendered.

Image Space Algorithms

- Image space algorithms do their work as the objects are being converted to pixels in the frame buffer.
- There solution of the display device is important here as this is done on a pixel by pixel basis.
- Pseudocode...: for each pixel in the frame buffer
- determine which polygon is closest to the viewer at that pixel location.
- color the pixel with the color of that polygon at that location.

Z buffer Algorithm

z-buffering also known as depth buffering is the management of image depth coordinates in3Dgraphics.

- It is a solution to the visibility problem, which is the problem of deciding which elements of a rendered scene are visible, and which are hidden.

Z buffer Algorithm

- If another object of the scene must be rendered in the same pixel, the method compares the two depths and overrides the current pixel if the object is closer to the observer.
- The chosen depth is then saved to the z buffer, replacing the old one.
- At the end, the z-buffer will allow the method to correctly reproduce the usual depth perception :a close object hides a farther one

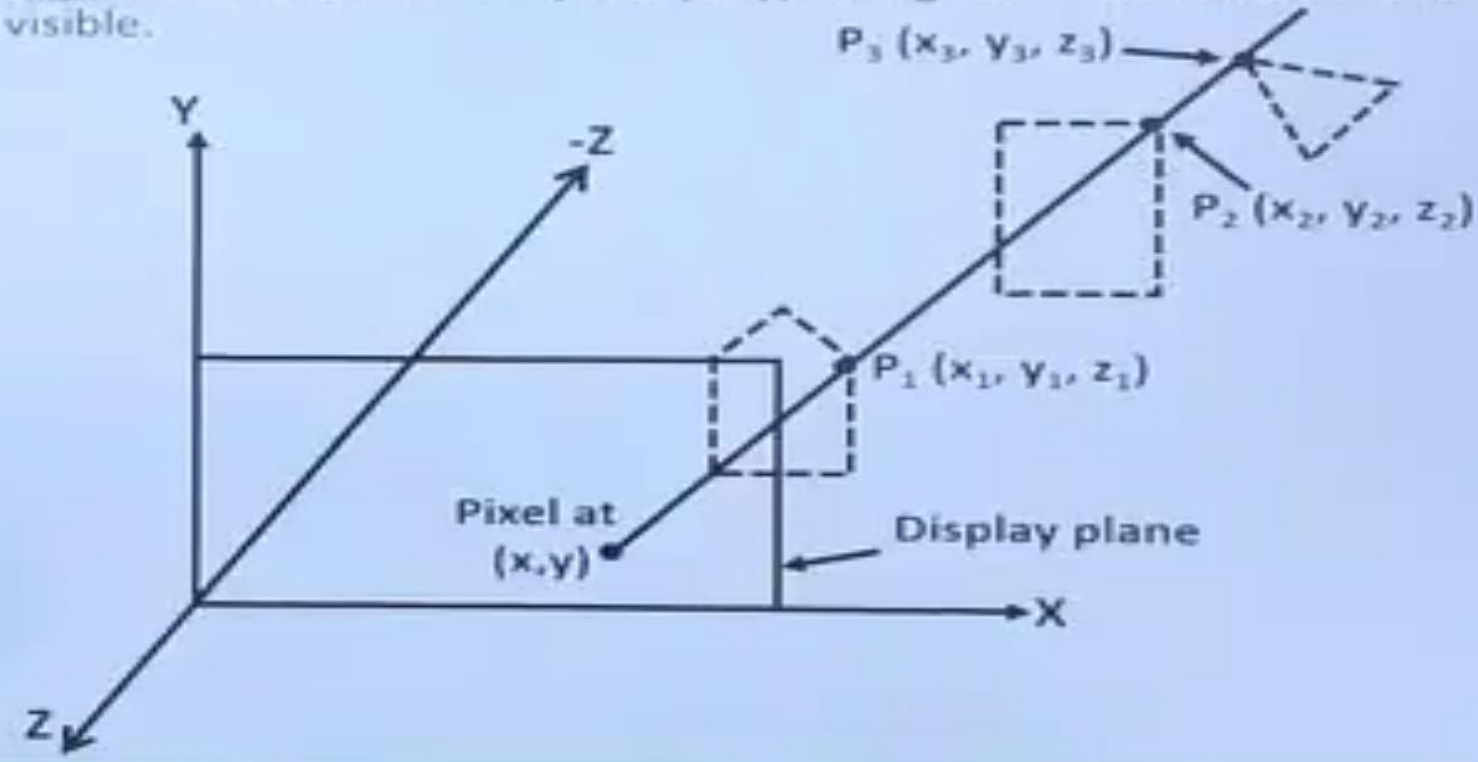
Hidden Lines and Surfaces

- This problem of hidden surface removal or equivalently visible surface determination is solved by applying a host of standard algorithms. These algorithms are broadly classified into two groups, namely –
  - ✓ **Image space algorithms** – that which determines object point closest to the viewer for generating each pixel in an image: for example Z-buffer algorithm, Floating Horizon method, Scan Line algorithm.
  - ✓ **Object space algorithms** – that which determines faces of the object visible (i.e., unobstructed by other faces or objects) w.r.t. the viewport for each object in the scene; for example Back Face Removal algorithm, BSP-Tree method.
- Theoretically object space algorithms require less computational work than image space algorithms, as the number of objects in a scene is far less than number of pixels in an image. But in practice the image space algorithms are found more efficient (taking advantage of coherence in rasterization).

Z-Buffer Algorithm

In Perhaps this is the most simple and popular of all algorithms. It can be very effectively implemented specially for objects whose faces can be described individually as planar surfaces.

Each point (x, y, z) on a planar surface corresponds to the orthographic projection point (x, y) object depths can be compared by comparing z values of all corresponding object points. Maximum z value means (x, y, z_{max}) lies closest to the pixel (x, y) along the Z direction. So it should be visible.

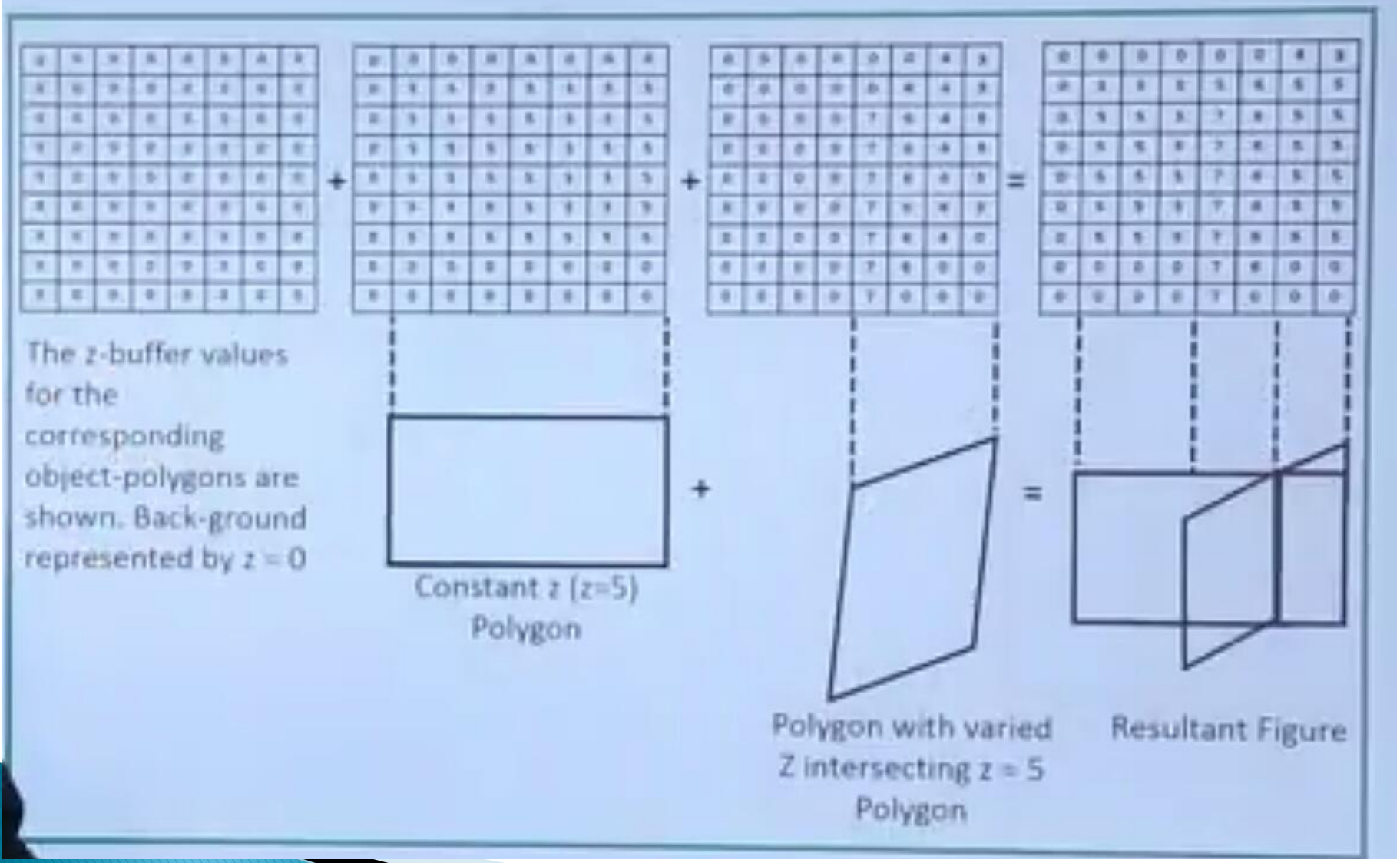


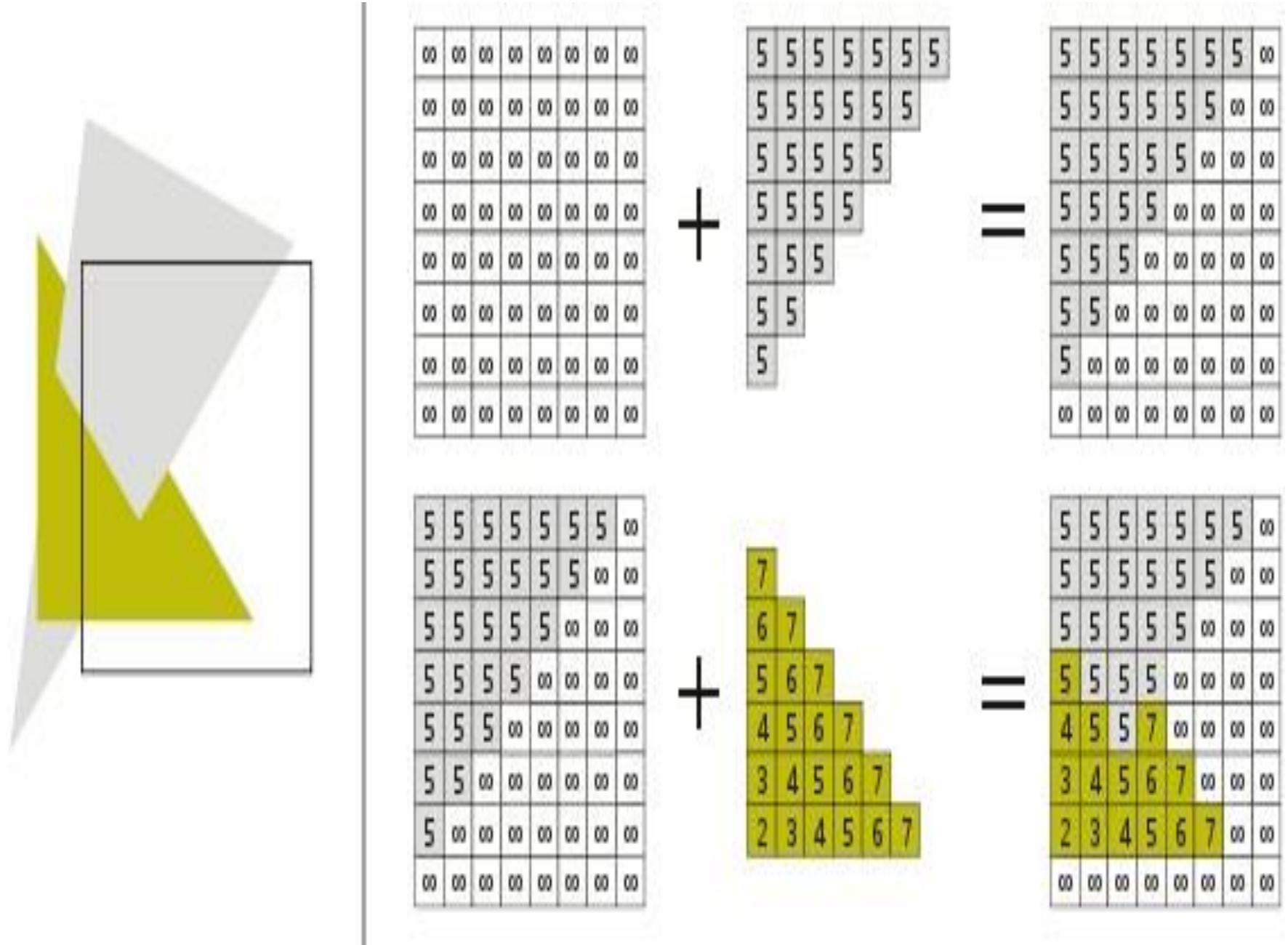
Z-Buffer Algorithm

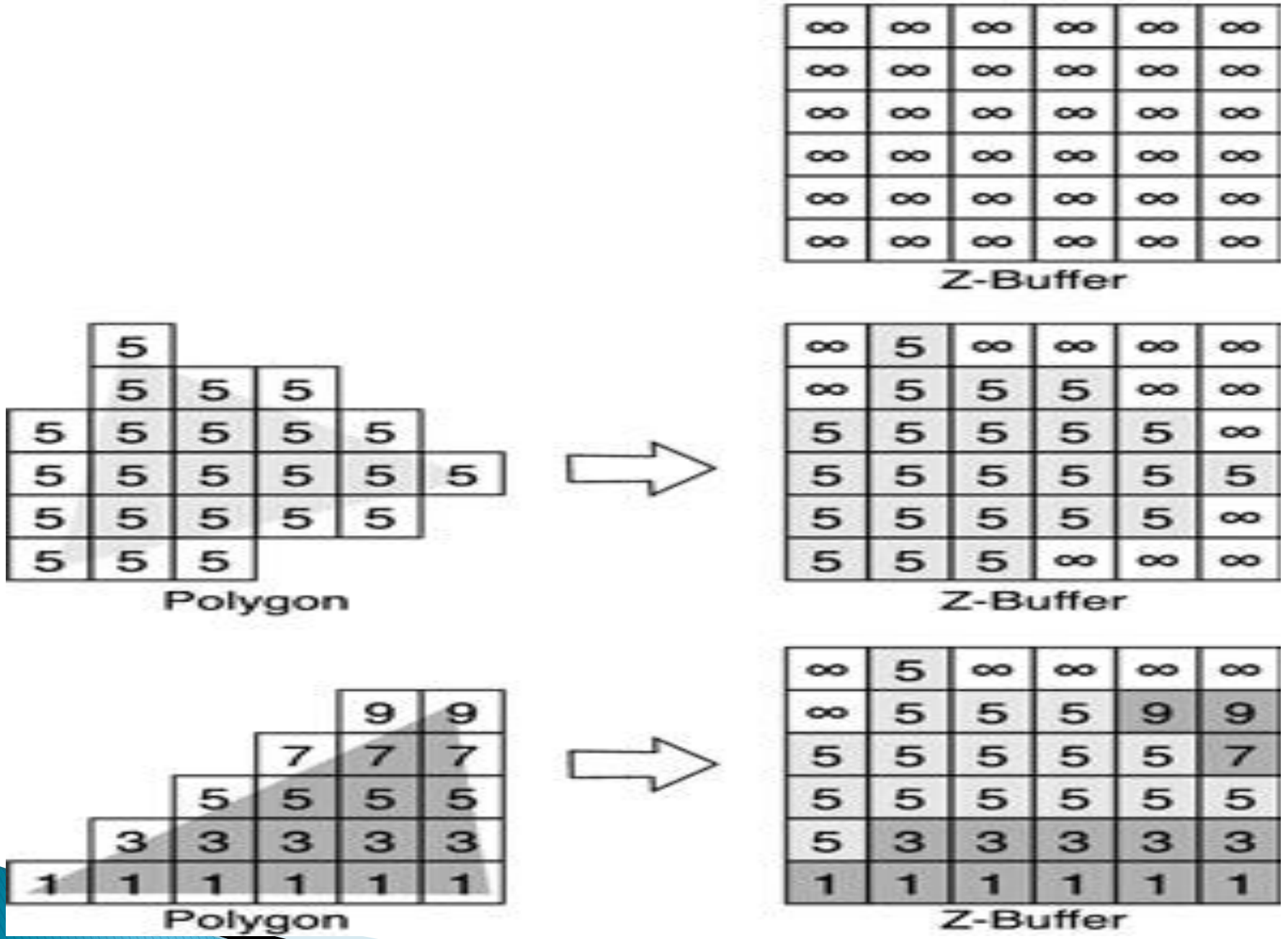
The algorithm works by sequentially executing the following steps :

1. For each pixel position (x, y)
   a) Set the frame buffer locations to the background intensity
   b) Set the z-buffer locations to $z_{max} = z_{min}$, the minimum of available depths
2. For each polygon (object surface) in the scene
   a) For each pixel (x, y) in a polygon's projection, calculate the depth z(x, y)
   b) Compare the depth z(x, y) with the value z_{max} stored in the z-buffer at (x, y) location.
      i. If $z(x, y) > z_{max}(x, y)$ then write the actual attributes (intensity) of point (x, y, z) of that polygon to the frame buffer at equivalent position of pixel location (x, y); also set $z_{max}(x, y) = z(x, y)$ i.e., replace the previous z-buffer value with current z.
      ii. If $z(x, y) \leq z_{max}(x, y)$ then nothing is done.

Z-Buffer Algorithm




∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
+
5 5 5 5 5 5 5
5 5 5 5 5 5
5 5 5 5 5
5 5 5 5
5 5 5
5 5
5
=
5 5 5 5 5 5 5 ∞
5 5 5 5 5 5 ∞ ∞
5 5 5 5 5 ∞ ∞ ∞
5 5 5 5 ∞ ∞ ∞ ∞
5 5 5 ∞ ∞ ∞ ∞ ∞
5 5 ∞ ∞ ∞ ∞ ∞ ∞
5 ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
5 5 5 5 5 5 5 ∞
5 5 5 5 5 5 ∞ ∞
5 5 5 5 5 ∞ ∞ ∞
5 5 5 5 ∞ ∞ ∞ ∞
5 5 5 ∞ ∞ ∞ ∞ ∞
5 5 ∞ ∞ ∞ ∞ ∞ ∞
5 ∞ ∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞
+
7
6 7
5 6 7
4 5 6 7
3 4 5 6 7
2 3 4 5 6 7
=
5 5 5 5 5 5 5 ∞
5 5 5 5 5 5 ∞ ∞
5 5 5 5 5 ∞ ∞ ∞
5 5 5 5 ∞ ∞ ∞ ∞
4 5 5 7 ∞ ∞ ∞ ∞
3 4 5 6 7 ∞ ∞ ∞
2 3 4 5 6 7 ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

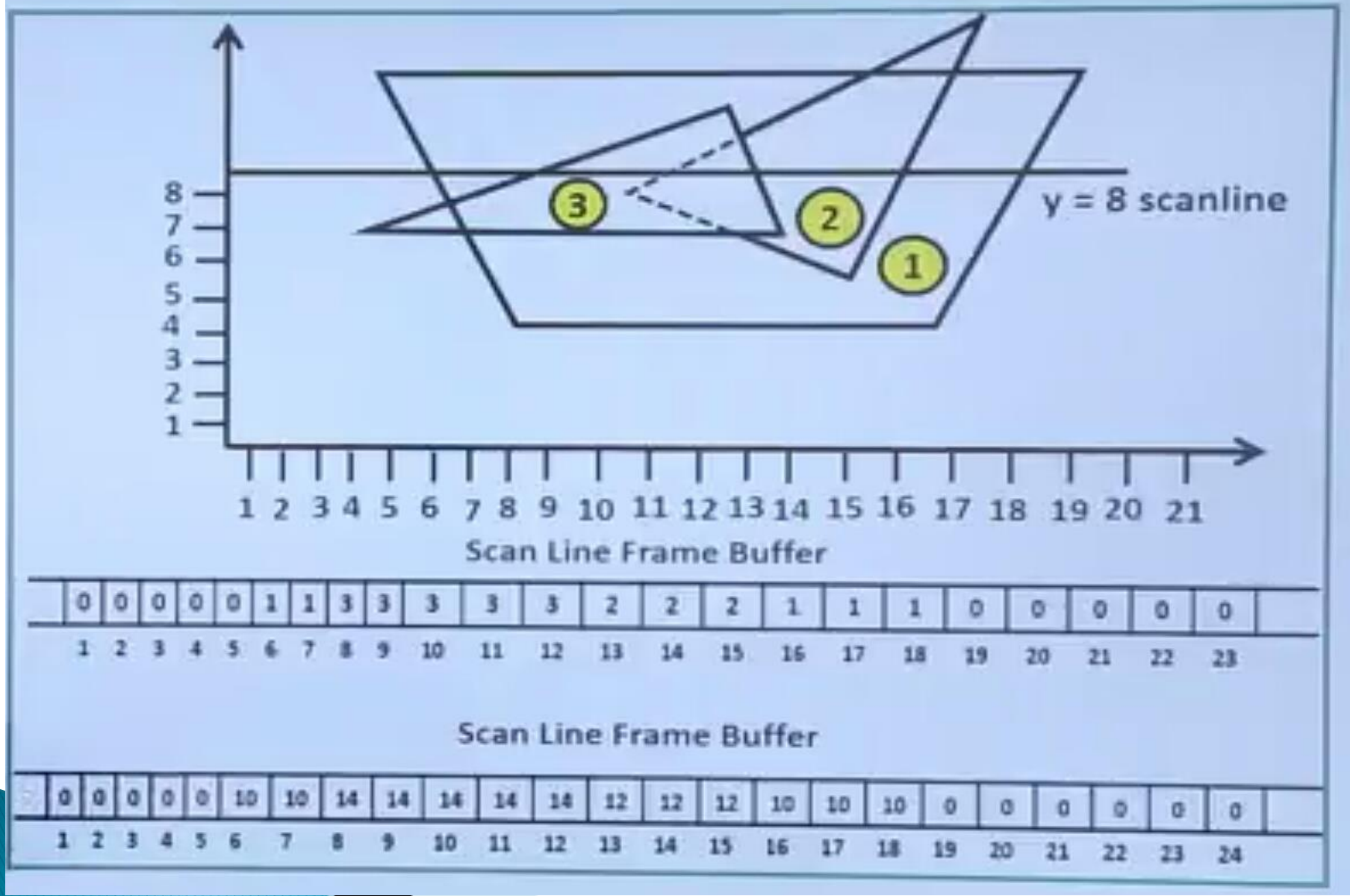

∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞ ∞
Z-Buffer
5
5 5 5
5 5 5 5 5
5 5 5 5 5 5
5 5 5 5 5
5 5 5
Polygon
∞ 5 ∞ ∞ ∞ ∞
∞ 5 5 5 ∞ ∞
5 5 5 5 5 ∞
5 5 5 5 5 5
5 5 5 5 5 ∞
5 5 5 ∞ ∞ ∞
Z-Buffer
9 9
7 7 7
5 5 5 5
3 3 3 3 3
1 1 1 1 1 1
Polygon
∞ 5 ∞ ∞ ∞ ∞
∞ 5 5 5 9 9
5 5 5 5 5 7
5 5 5 5 5 5
5 3 3 3 3 3
1 1 1 1 1 1
Z-Buffer

Scan Line Z-Buffer Algorithm

This algorithm is an extension of the method of scan converting a solid polygon and can be treated as a special case of Z-buffer algorithm. The working principle of the algorithm can be summarized through following steps.

1. For each scan line (y = constant) of the pixel grid
   a) Set the frame buffer to the background intensity
   b) Set the z-buffer to z_{min}
2. For each polygon in the scene (in arbitrary order)
   a) Find intersection, if any, of the polygon's 2D projection with the scan line.
   b) For each pixel (in left to right order) on the scan line between the intersection pairs compare its depth (z value) to the depth recorded in the z-buffer at that location.
      i. If the pixel depth is greater than that in the z-buffer then this line segment from the left intersection to current pixel is currently visible line segment. Hence the polygon attribute for this line segment is written to the frame buffer and the z-buffer is updated.

Scan Line Z-Buffer Algorithm



Z buffer Algorithm

```
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);
```

Advantages of zbuffer

- Easy to implement, •Scan-line algorithm
- Fast with hardware support .Fast depth buffer memory
- On most hardware
- No sorting of objects

Disadvantages

- Extra memory required for z-buffer:

Thank You