S J P N Trust's

HIRASUGAR INSTITUTE OF TECHNOLOGY, Nidasoshi.

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

Module-4
18CS62

3D Viewing and Visible Surface Detection:

Course Outcome

Apply the concepts of viewing and visible surface detection of 3D objects

Prof. Rahul Palakar

Overview of Three-Dimensional Viewing Concepts

- When we model a three-dimensional scene, each object in the scene is typically defined with a set of surfaces that form a closed boundary around the object interior.
- Viewing functions process the object descriptions through a set of procedures that ultimately project a specified view of the objects onto the surface of a display device.

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.

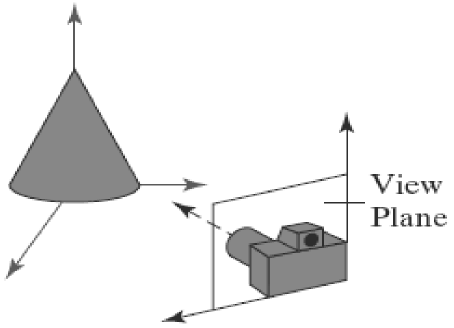
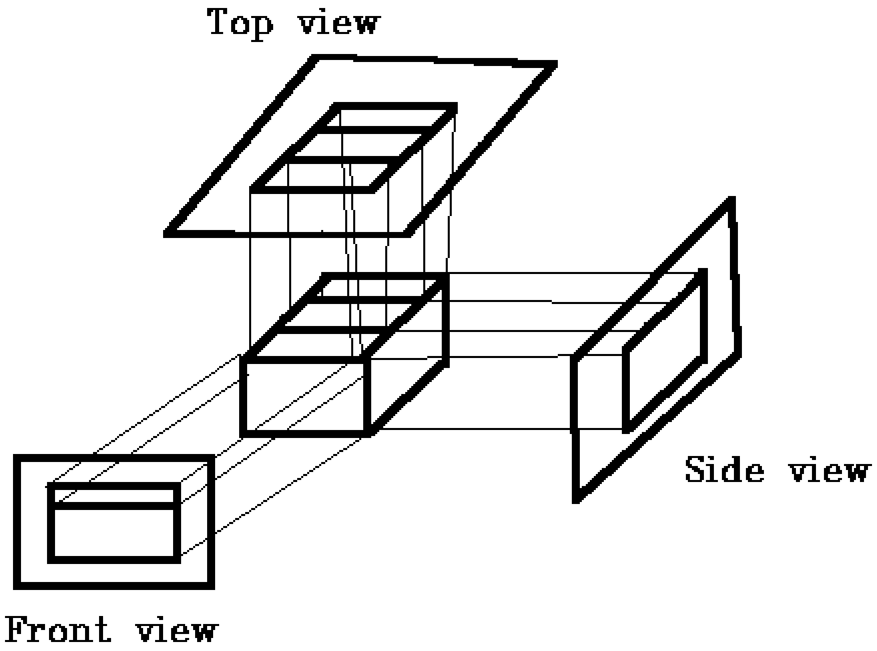


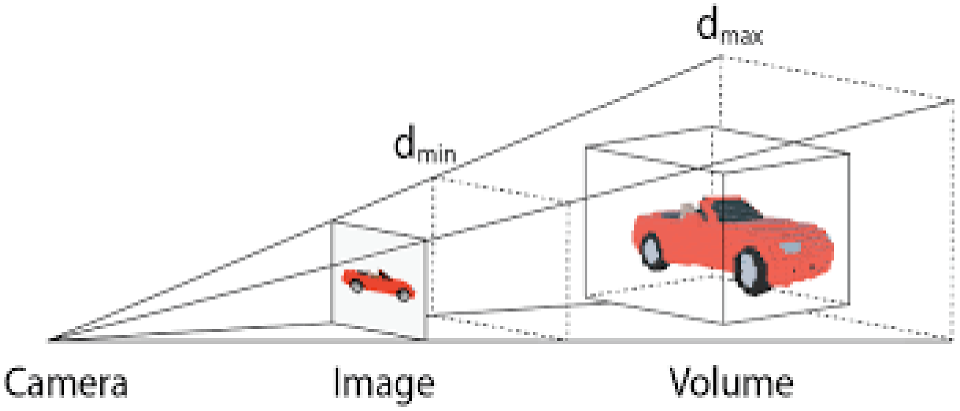
FIGURE 1
Coordinate reference for obtaining a selected view of a three-dimensional scene.

Projections:

- Unlike a camera picture, we can choose different methods for projecting a scene onto the view plane.

1. *parallel projection.* ***2.*** *perspective projection.*

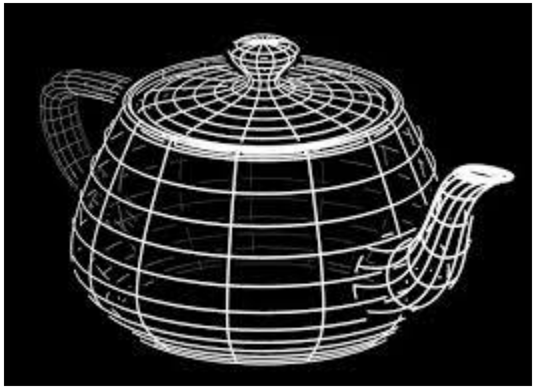


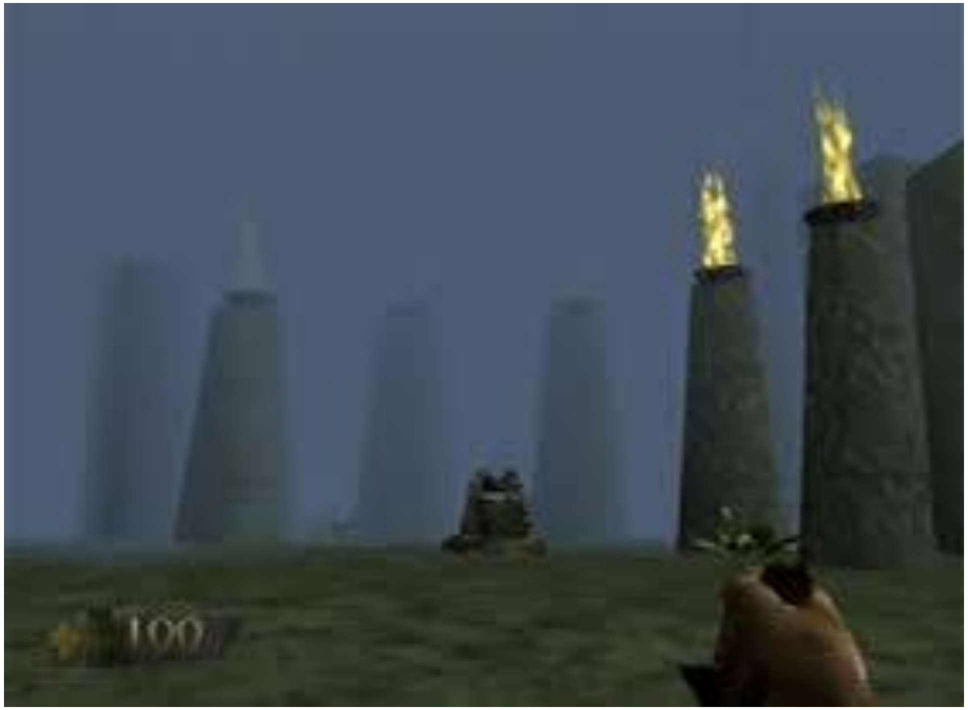


Depth Cueing:

- With few exceptions, depth information is important in a three-dimensional scene so that we can easily identify, for a particular viewing direction, which is the front and which is the back of each displayed object.
- There are several ways in which we can include depth information in the two-dimensional representation of solid objects.
- A simple method for indicating depth with wire-frame displays is to vary the brightness of line segments according to their distances from the viewing position.

- Another application of depth cuing is modeling the effect of the atmosphere on the perceived intensity of objects.

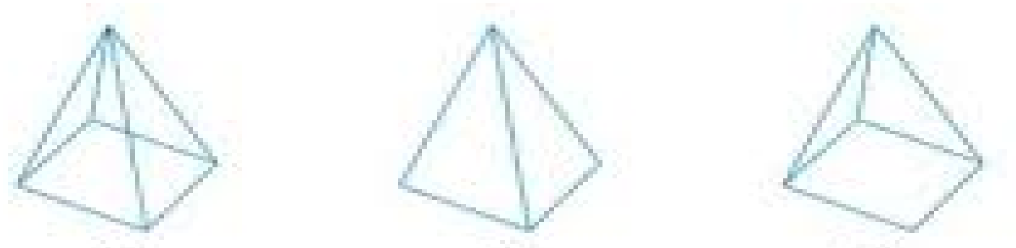




Identifying Visible Lines and Surfaces:

- We can also clarify depth relationships in a wire-frame display using techniques other than depth cueing.
- One approach is simply to highlight the visible lines or to display them in a different color.
- Another technique, commonly used for engineering drawings, is to display the nonvisible lines as dashed lines.
- Or we could remove the nonvisible lines from the display.

- But removing the hidden lines also removes information about the shape of the back surfaces of an object, and wire-frame representations are generally used to get an indication of an object's overall appearance, front and back.



(a) (b) (c)

- When a realistic view of a scene is to be produced, back parts of the objects are completely eliminated so that only the visible surfaces are displayed.

Surface Rendering:

- Added realism is attained in displays by rendering object surfaces using the lighting conditions in the scene and the assigned surface characteristics.
- We set the lighting conditions by specifying the color and location of the light sources, and we can also set background illumination effects.
- Surface properties of objects include whether a surface is transparent or opaque and whether the surface is smooth or rough.

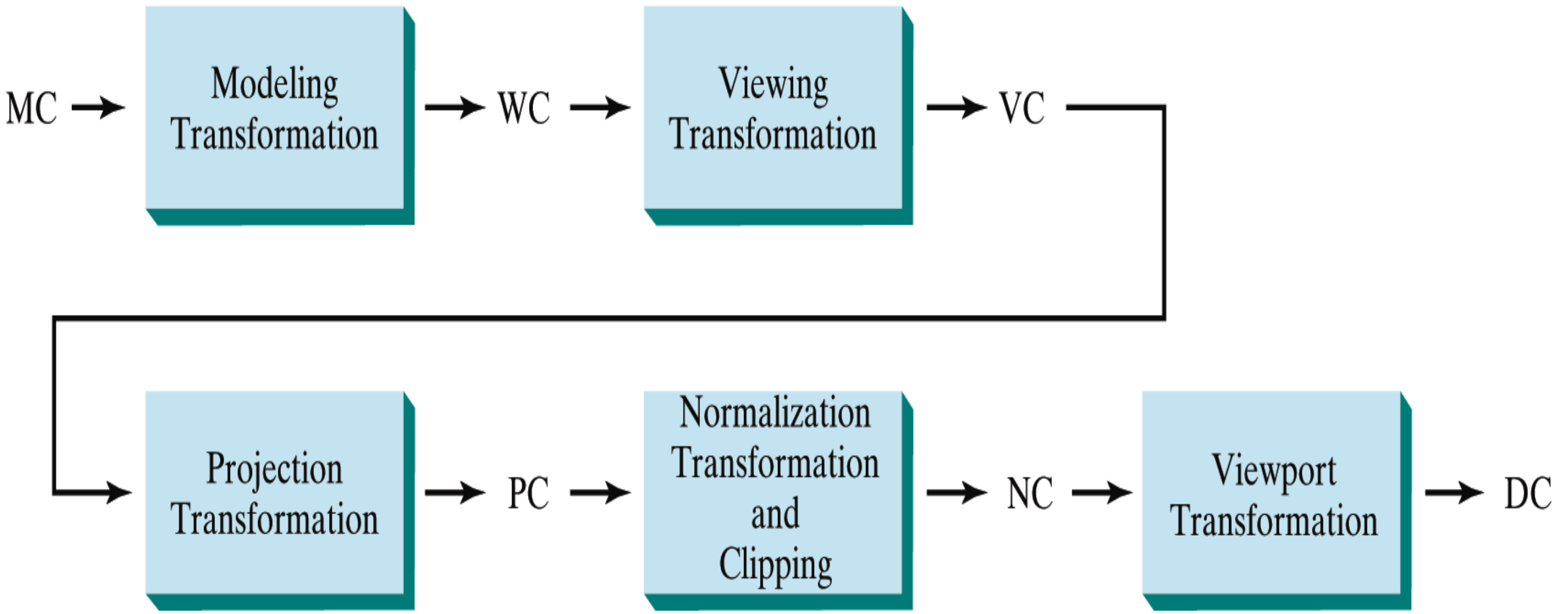
The Three-Dimensional Viewing Pipeline

- First of all, we need to choose a viewing position corresponding to where we would place a camera.
- We could also pick a position in the middle of a group of objects or even inside a single object, such as a building or a molecule.
- Then we must decide on the camera orientation.
- Finally, when we snap the shutter, the scene is cropped to the size of a selected clipping window, and light from the visible surfaces is projected onto the camera film.

- Some of the viewing operations for a three-dimensional scene are the same as, or similar to, those used in the two-dimensional viewing pipeline.
- In three-dimensional viewing, however, the clipping window is positioned on a selected view plane, and scenes are clipped against an enclosing volume of space, which is defined by a set of *clipping planes*.
- The viewing position, view plane, clipping window, and clipping planes are all specified within the viewing-coordinate reference frame.

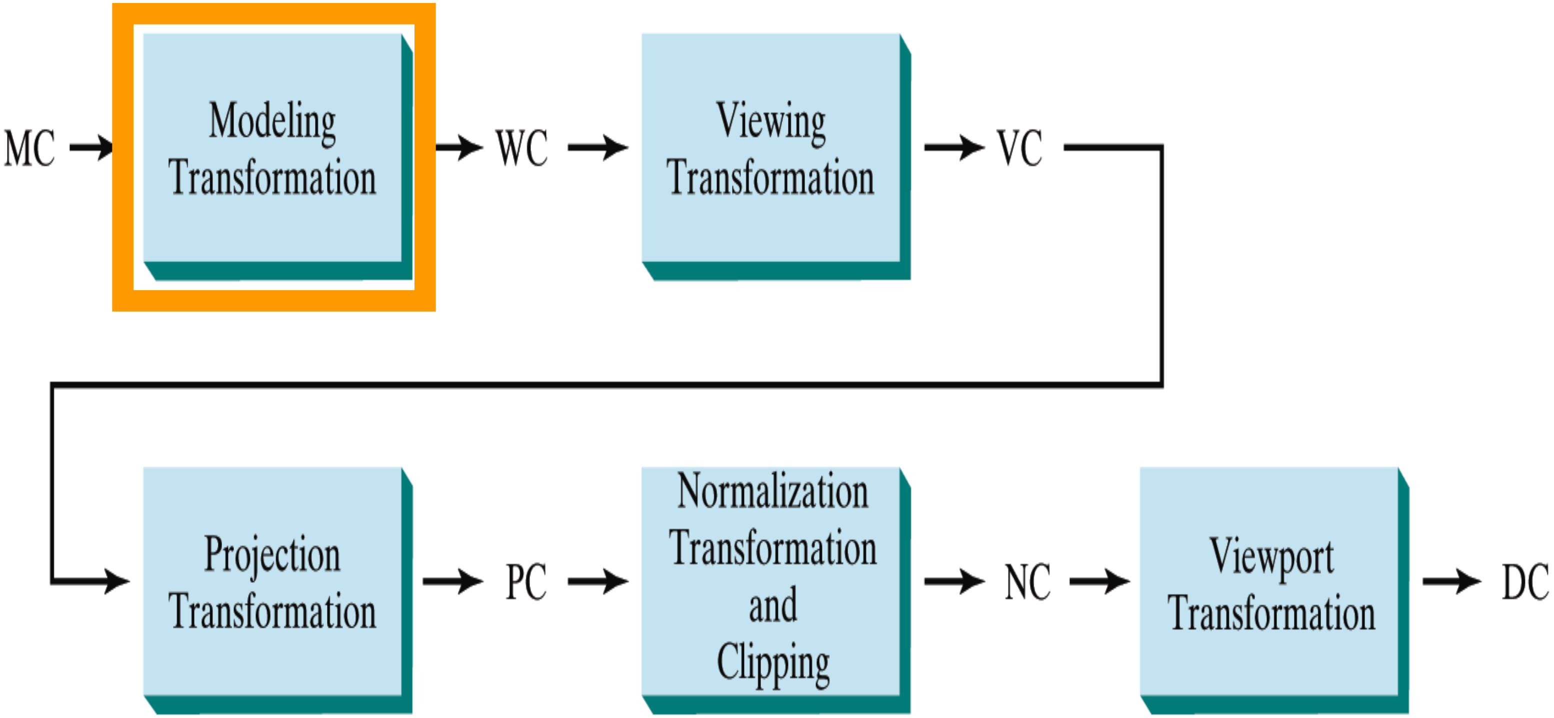
3D Viewing Pipeline

- The general processing steps for modeling and converting a world coordinate description of a scene to device coordinates:



3D Viewing Pipeline

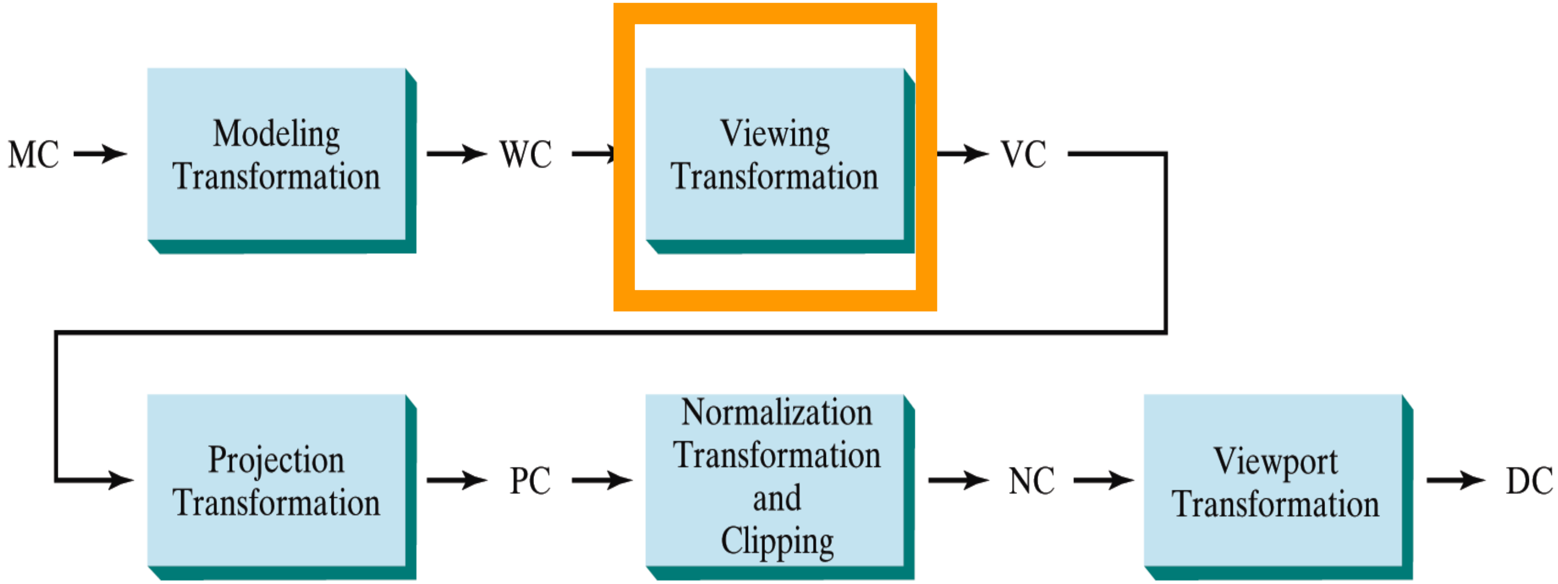
1. Construct the shape of individual objects in a scene within modeling coordinate, and place the objects into appropriate positions within the scene (world coordinate).



3D Viewing Pipeline

2. World coordinate positions are converted to viewing coordinates.

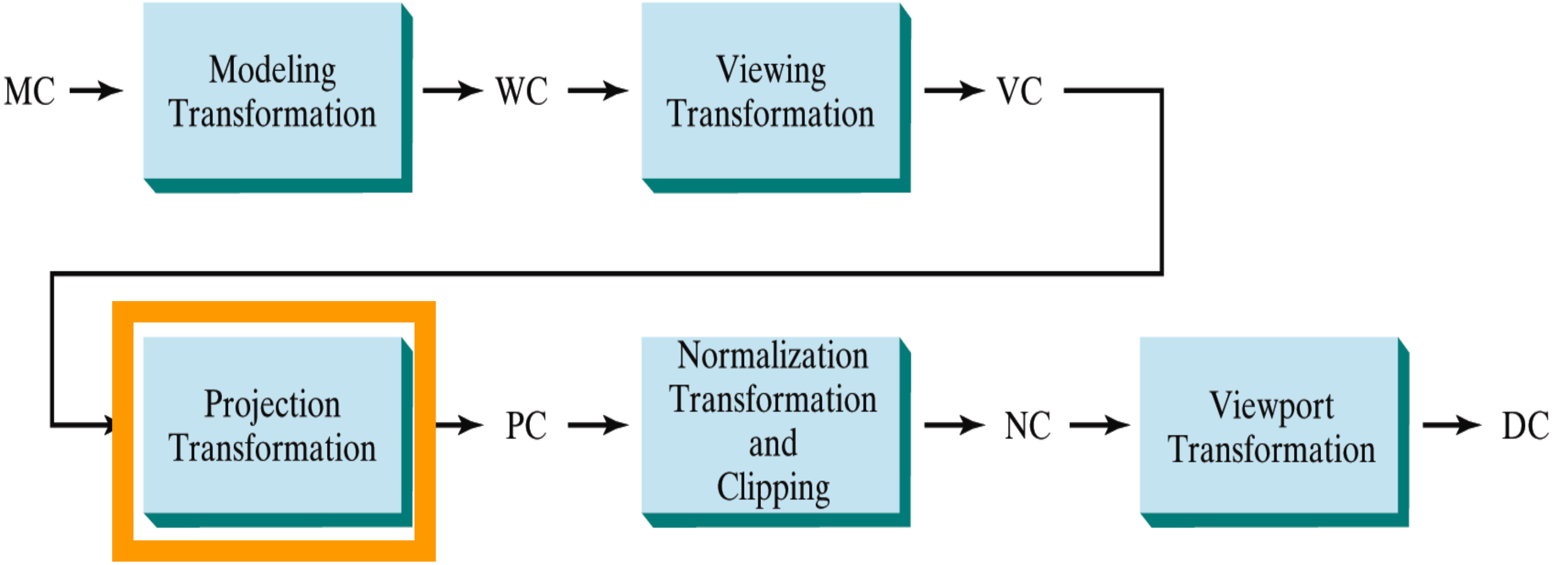
- The viewing coordinate system defines the viewing parameters, including the position and orientation of the projection plane (view plane), which we can think of as the camera film plane.



3D Viewing Pipeline

3. Convert the viewing coordinate description of the scene to coordinate positions on the projection plane.

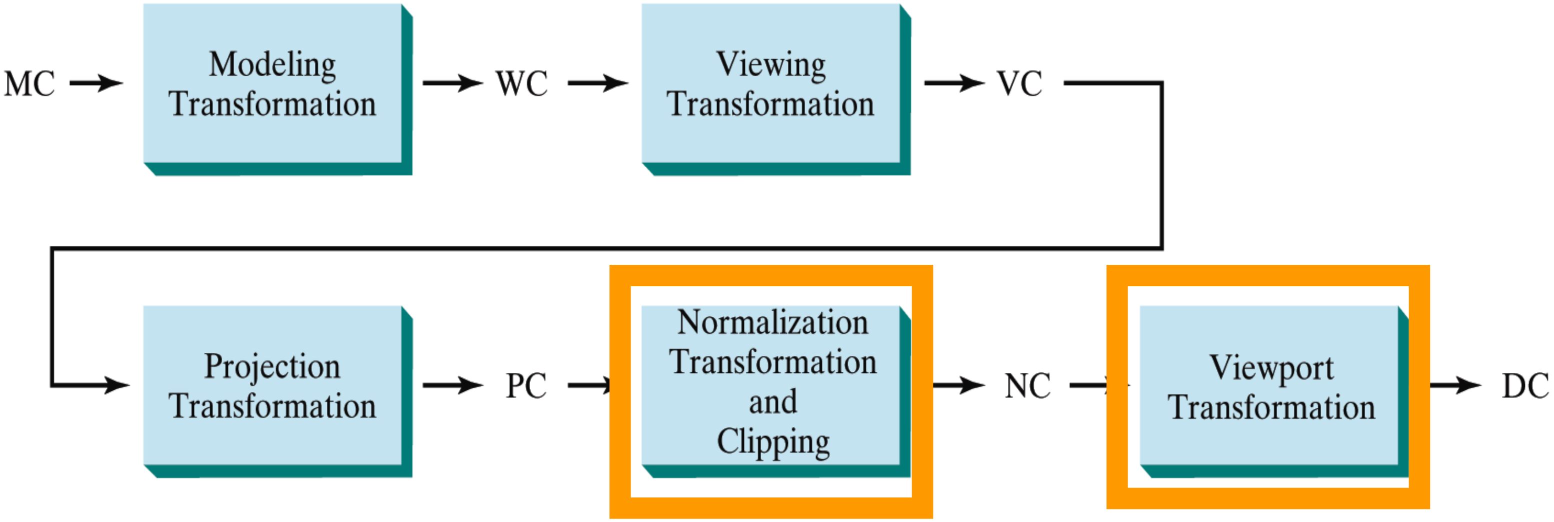
- A two-dimensional clipping window, corresponding to a selected camera lens, is defined on the projection plane, and a three-dimensional clipping region is established. This clipping region is called the **view volume.**



3D Viewing Pipeline

4. Positions on the projection plane, will then mapped to the Normalized coordinate and output device.

- all parts of the scene outside the view volume are clipped off.
- As in two-dimensional viewing, the viewport limits could be given in normalized coordinates or in device coordinates.

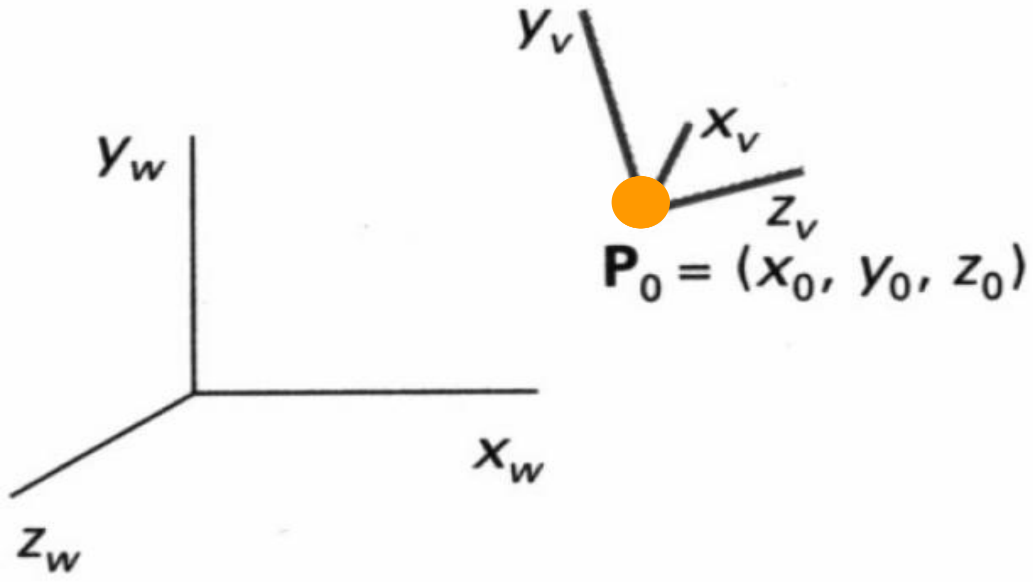


3D Viewing-Coordinate Parameters

- Establishing a three-dimensional viewing reference frame is similar to setting up the two-dimensional viewing reference frame.
- We first select a world-coordinate position $\mathbf{P}_0 = (x_0, y_0, z_0)$ for the viewing origin, which is called the **view point** or **viewing position**. (Sometimes the view point is also referred to as the *eye position* or the *camera position*.)
- And we specify a **view-up vector V,** which defines the y_{view} direction.
- we also need to assign a direction for one of the remaining two coordinate axes. This is typically accomplished with a second vector that defines the z_{view} axis, with the viewing direction along this axis.

Specifying the Viewing Coordinate System (View point/Camera position)

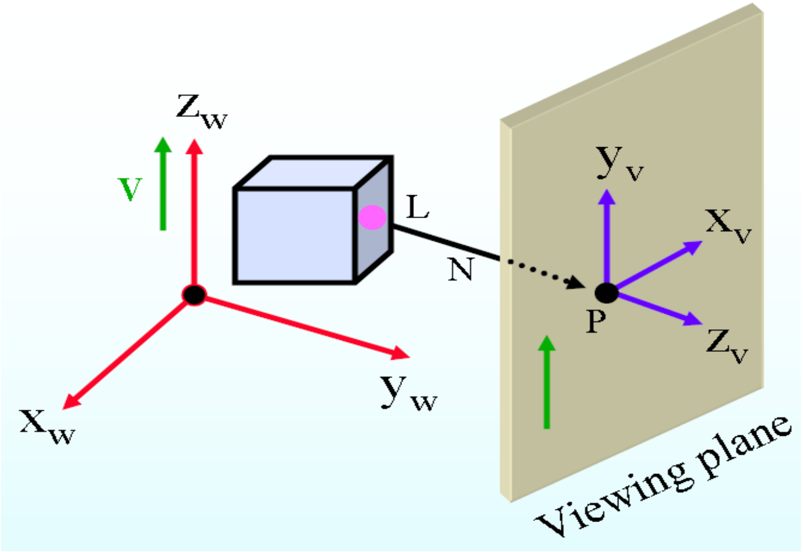
- We first pick a world coordinate position called **view point** (origin of our viewing coordinate system).
- P_0 is a point where a camera is located.



Specifying the Viewing Coordinate System

(View plane Normal vector)

- Because the viewing direction is usually along the z_{view} axis, the **view plane,** also called the **projection plane,** is normally assumed to be perpendicular to this axis.
- Next, we select the positive direction for the viewing $\mathbf{z_v}$ axis, by specifying the **view plane normal vector**, **N**.
- The direction of **N**, is from the **look at point** (L) to the view point.



- An additional scalar parameter is used to set the position of the view plane at some coordinate value z_{vp} along the z_{view} axis.
- The view plane is always parallel to the $x_{\text{view}}y_{\text{view}}$ plane, and the projection of objects to the view plane corresponds to the view of the scene that will be displayed on the output device.

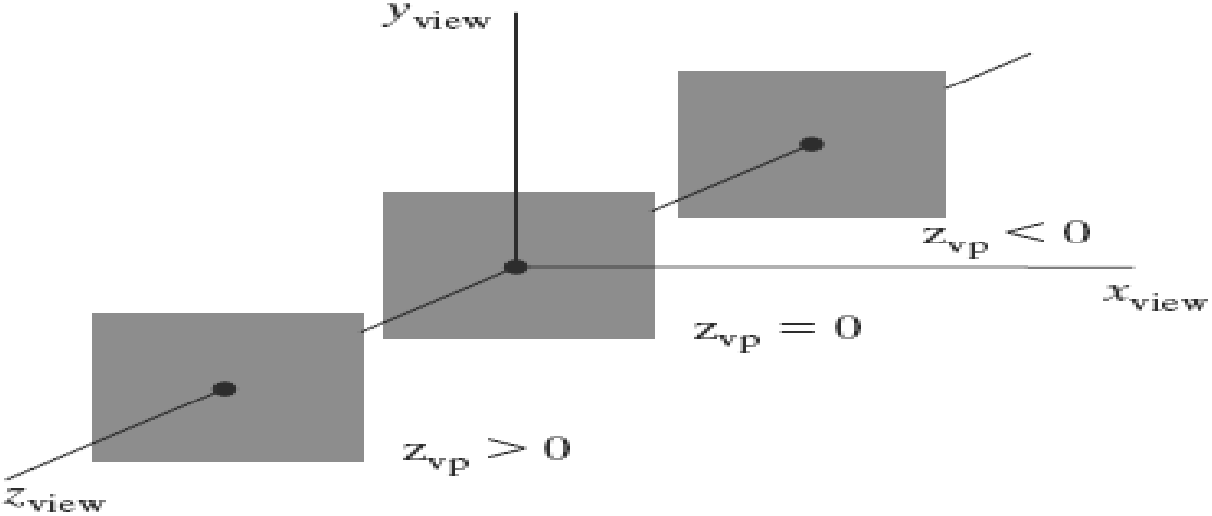
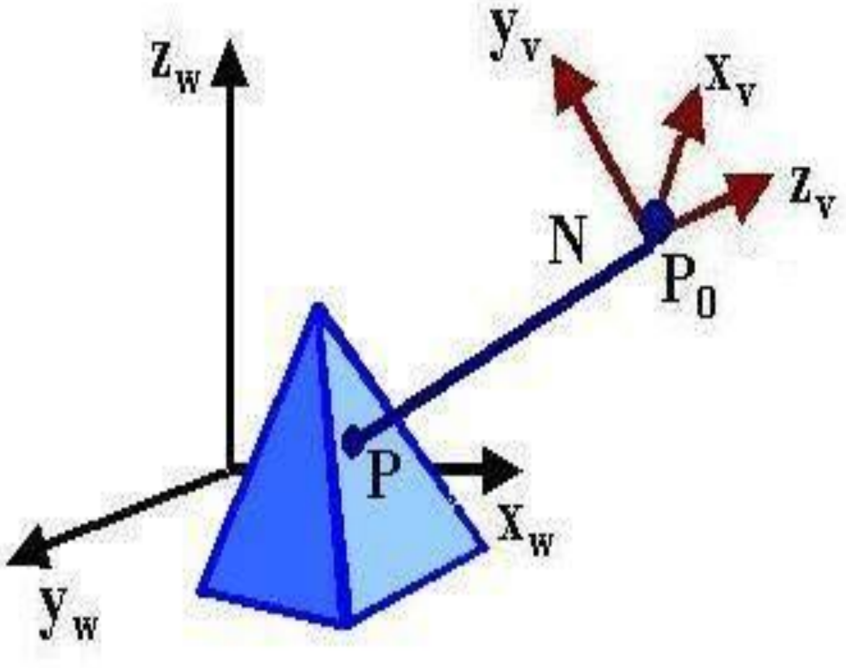


FIGURE 9
Three possible positions for the view plane along the z_{view} axis.

- Vector **N** can be specified in various ways.
- The direction for **N** is defined to be along the line from the world-coordinate origin to a selected point position.
- Other systems take **N** to be in the direction from a reference point $\mathbf{P}_{ref}$ to the viewing origin $\mathbf{P}_0$.
- The reference point is often referred to as a *look-at point* within the scene.



Specifying the Viewing Coordinate System

(view up direction)

- Once we have chosen a view-plane normal vector **N**, we can set the direction for the view-up vector **V**.
- This vector is used to establish the positive direction for the $\mathbf{y_v}$ axis.
- ***V*** is projected into a plane that is perpendicular to the normal vector.
- Usually, **V** is defined by selecting a position relative to the world-coordinate origin, so that the direction for the view-up vector is from the world origin to this selected position.
- In general, it can be difficult to determine a direction for **V** that is precisely perpendicular to **N**.
- We can choose any direction for the view-up vector **V**, so long as it is not parallel to **N**.
- A convenient choice is often in a direction parallel to the world y_w axis; that is, we could set **V** = *(*0, 1, 0*)*.

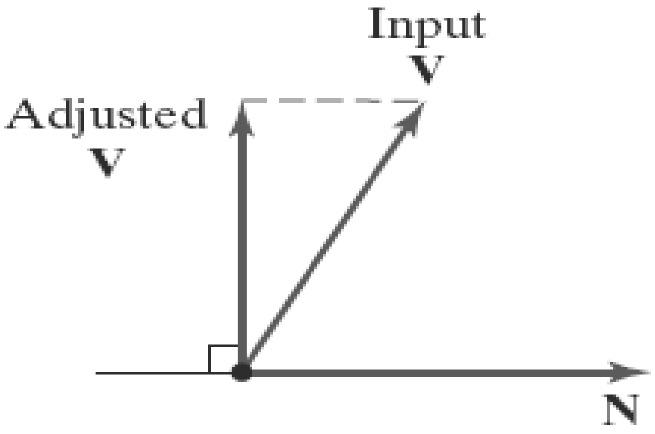
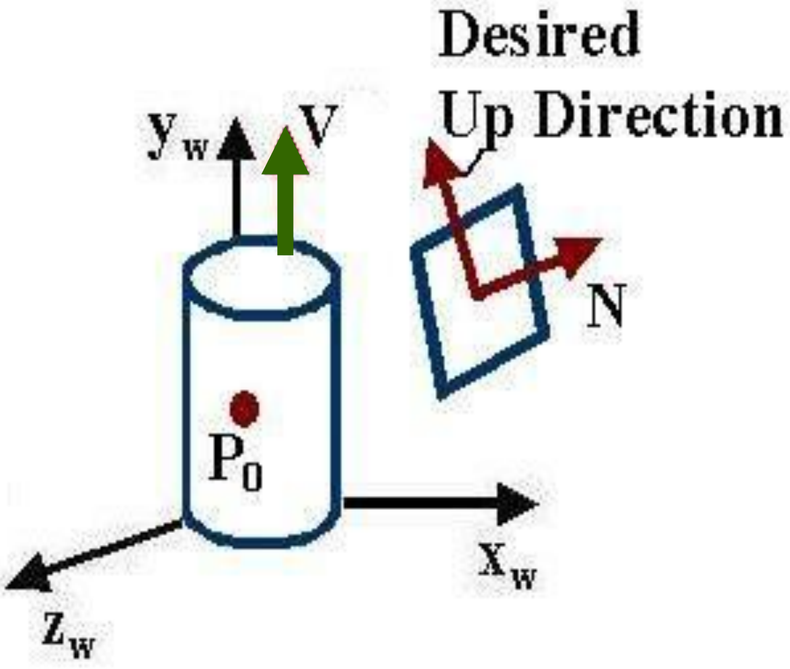


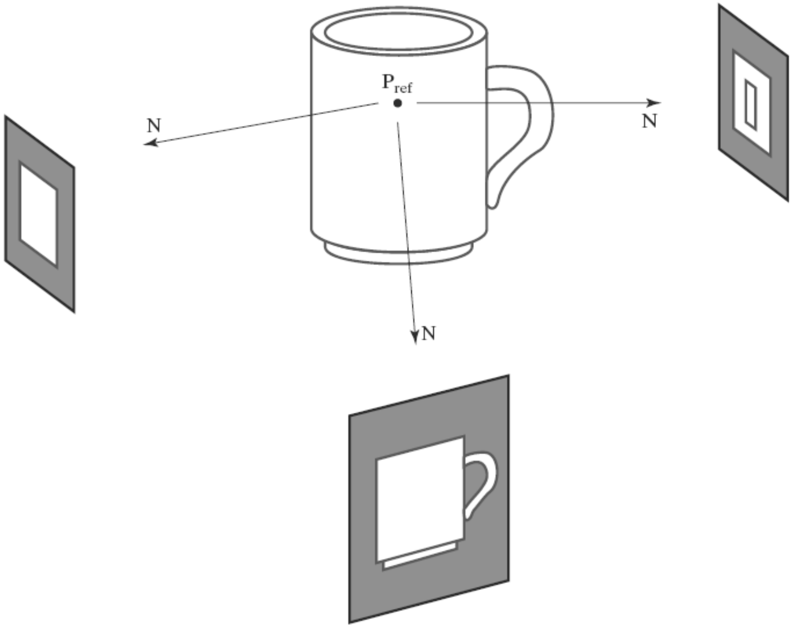
FIGURE 11
Adjusting the input direction of the view-up vector **V** to an orientation perpendicular to the view-plane normal vector **N**.

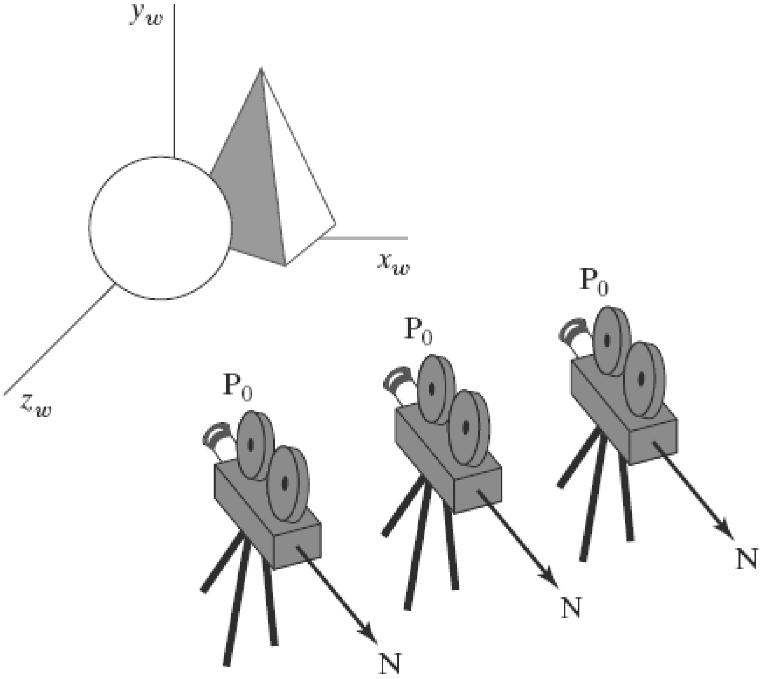


Generating Three-Dimensional Viewing Effects

- By varying the viewing parameters, we can obtain different views of objects in a scene.
- For instance, from a fixed viewing position, we could change the direction of **N** to display objects at positions around the viewing-coordinate origin.
- In interactive applications, the normal vector **N** is the viewing parameter that is most often changed.
- If we want to simulate an animation panning effect, as when a camera moves through a scene or follows an object that is moving through a scene, we can keep the direction for **N** fixed as we move the view point,

Viewing an object from different directions using a fixed reference point.





Panning across a scene by changing the viewing position, with a fixed direction for **N**.

Thank You

S J P N Trust's

HIRASUGAR INSTITUTE OF TECHNOLOGY, Nidasoshi.

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

Module-4

18CS62

3D Viewing and Visible Surface Detection:

Course Outcome

Apply the concepts of viewing and visible surface detection of 3D objects

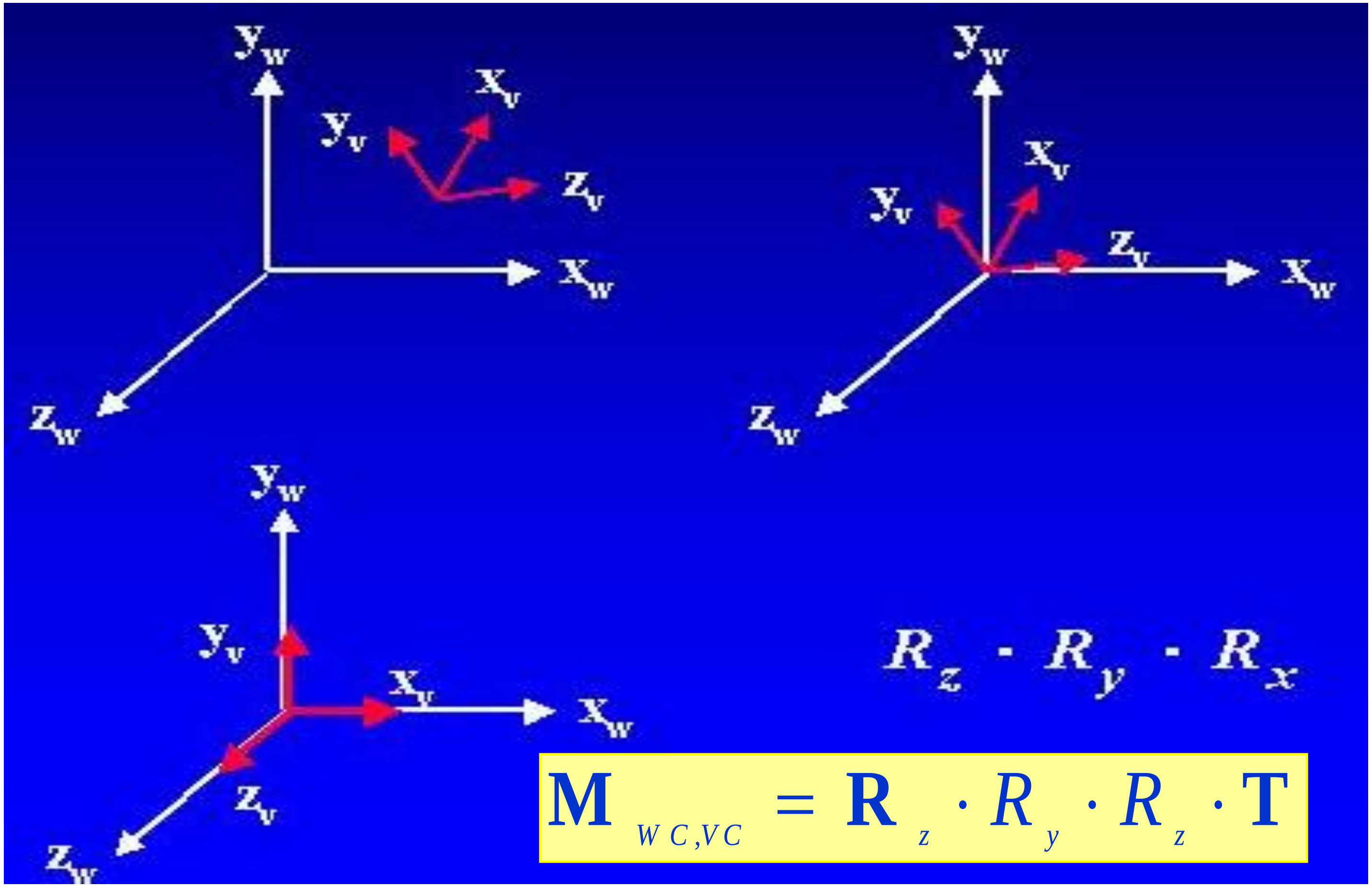
Prof. Rahul Palakar

Transformation from World to Viewing Coordinates

- In the three-dimensional viewing pipeline, the first step after a scene has been constructed is to transfer object descriptions to the viewing-coordinate reference frame.
- This conversion of object descriptions is equivalent to a sequence of transformations that superimposes the viewing reference frame onto the world frame.
- We can accomplish this conversion using the methods for transforming between coordinate system :

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.



$$\mathbf{M}_{WC,VC} = \mathbf{R}_z \cdot R_y \cdot R_z \cdot \mathbf{T}$$

- The viewing-coordinate origin is at world position **P0** = (x_0, y_0, z_0).

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

- 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 fram

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

- 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:

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

- Translation factors in this matrix are calculated as the vector dot product of each of the **u**, **v**, and **n** unit vectors with **P**0, which represents a vector from the world origin to the viewing origin.

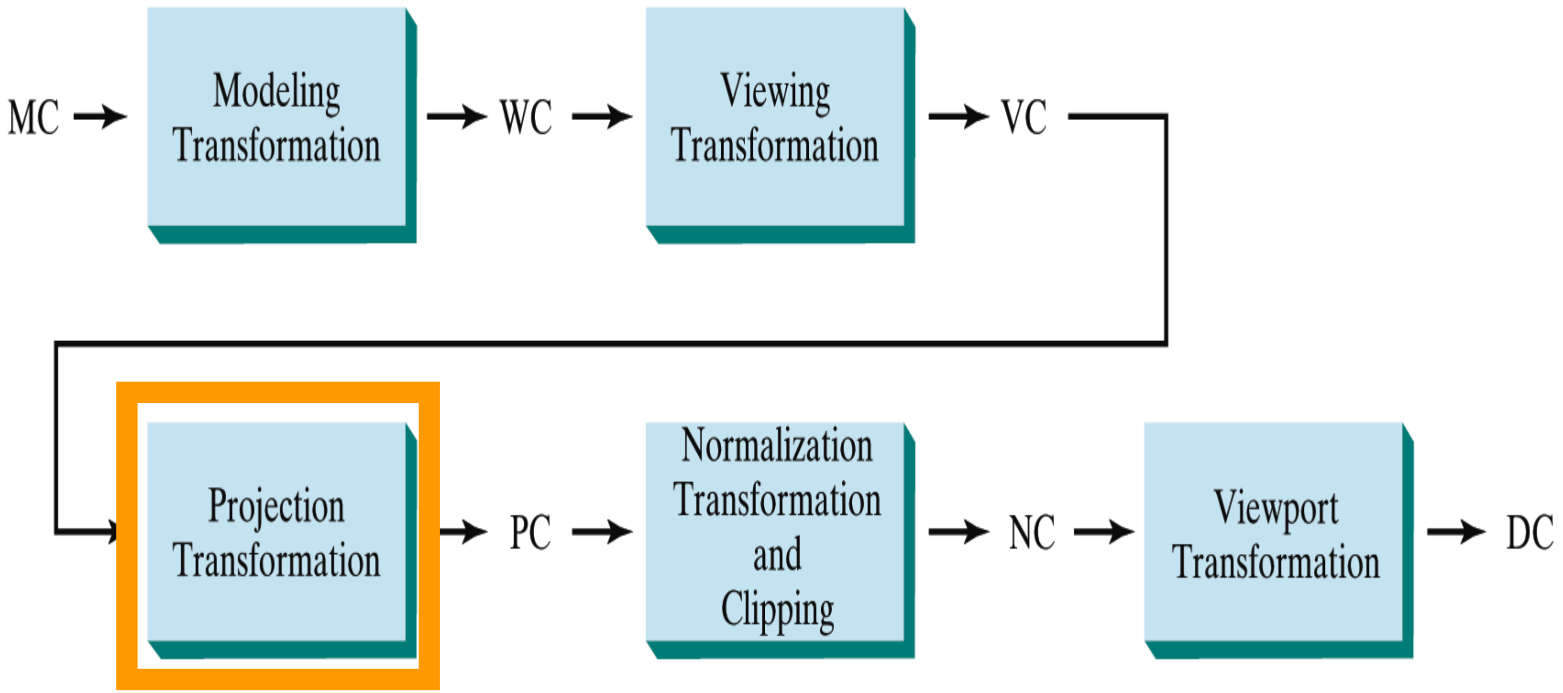
$$-\mathbf{u} \cdot \mathbf{P}_0 = -x_0 u_x - y_0 u_y - z_0 u_z$$

$$-\mathbf{v} \cdot \mathbf{P}_0 = -x_0 v_x - y_0 v_y - z_0 v_z$$

$$-\mathbf{n} \cdot \mathbf{P}_0 = -x_0 n_x - y_0 n_y - z_0 n_z$$

Projection Transformations

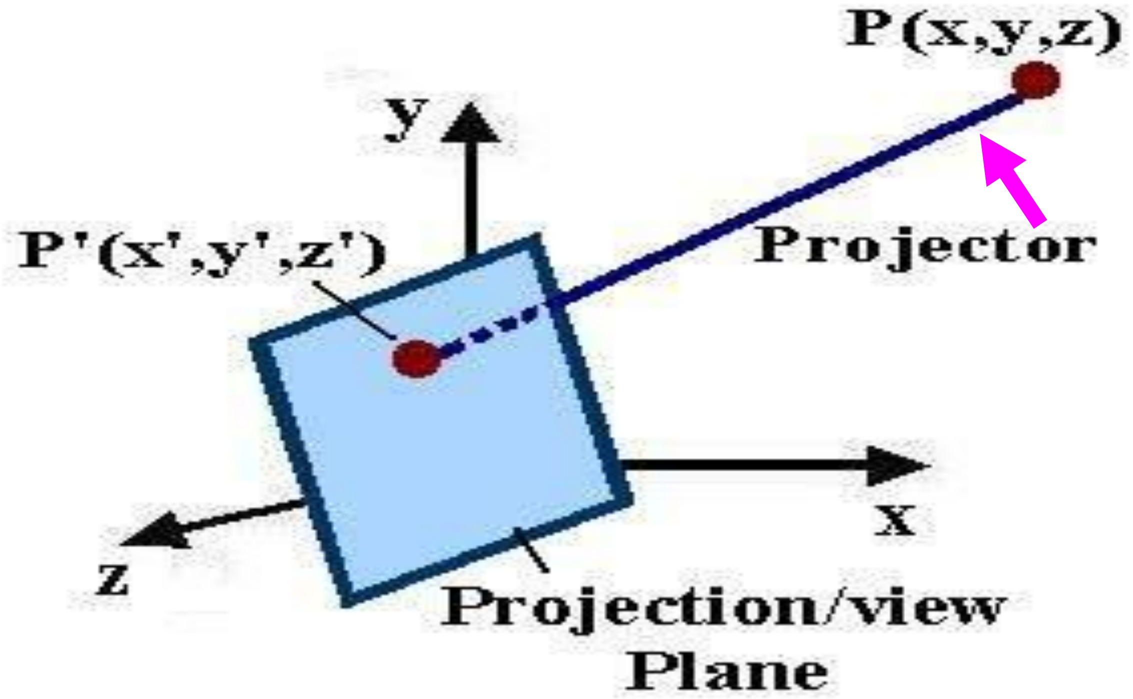
- Convert the viewing coordinate description of the scene to coordinate positions on the projection plane.
- Viewing 3D objects on a 2D display requires a mapping from 3D to 2D.



Projection

- **Projection** can be defined as a mapping of point P(x,y,z) onto its image in the projection plane.

- The mapping is determined by a ***projector*** that passes through P and intersects the view plane

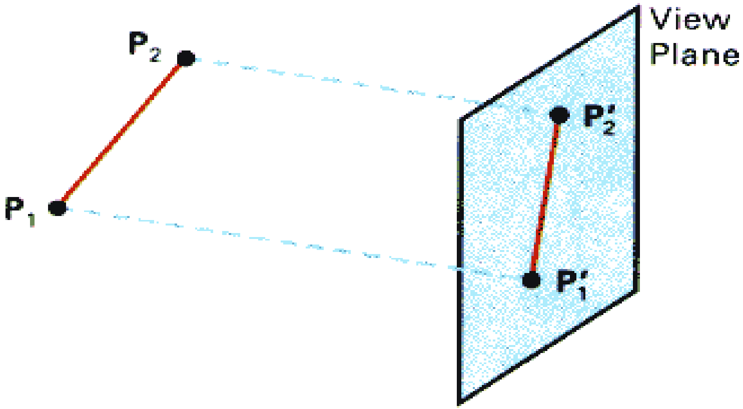


- Graphics packages generally support both:

1. Parallel Projection 2. Perspective Projection

Parallel Projection:

Transforms object positions to the view plane along parallel lines.

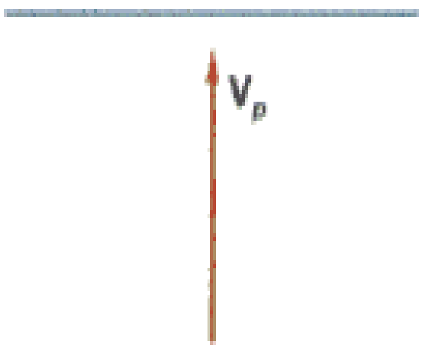


- A parallel projection preserves relative proportions of objects, and this is the method used in computer aided drafting and design to produce scale drawings of three-dimensional objects.

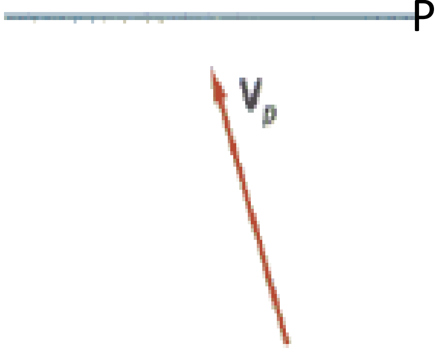
- There are two general methods for obtaining a parallel-projection view of an object:

1. Orthographic   2. Oblique Projection

Projection Plane



Orthographic Projection

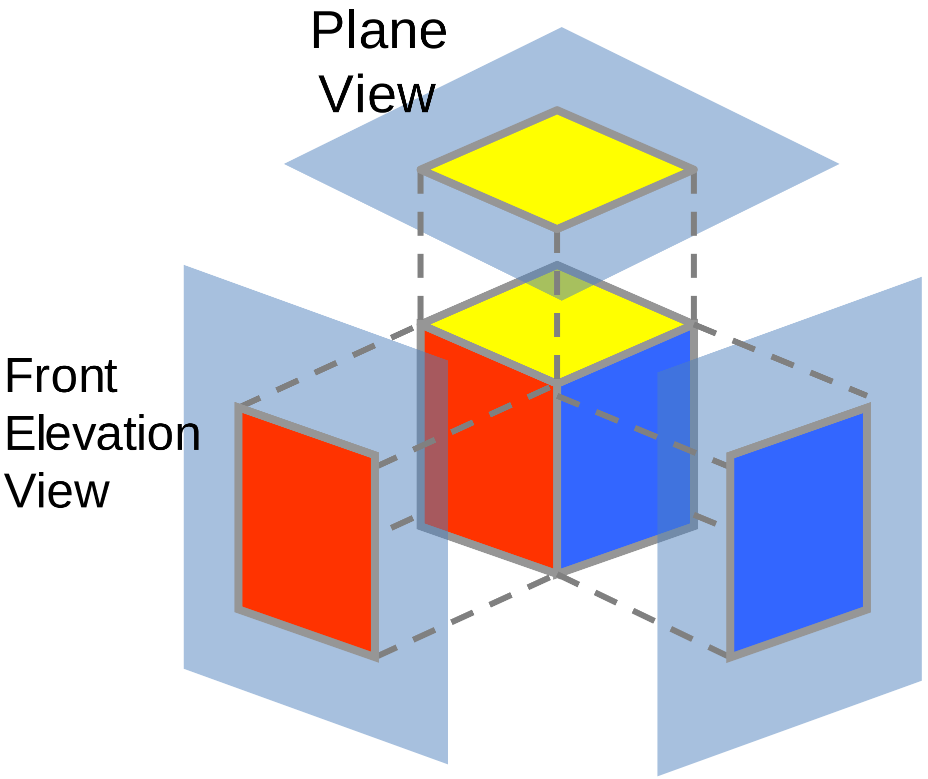


Projection Plane

Oblique Projection

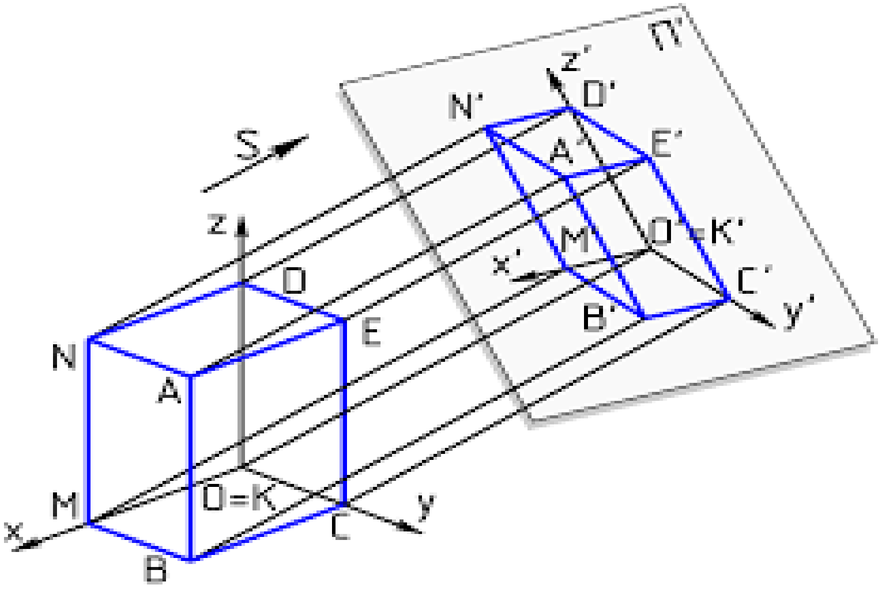
Orthographic parallel projections are done by projecting points along parallel lines that are perpendicular to the projection plane.

Oblique projections are obtained by projecting along parallel lines that are NOT perpendicular to the projection plane.

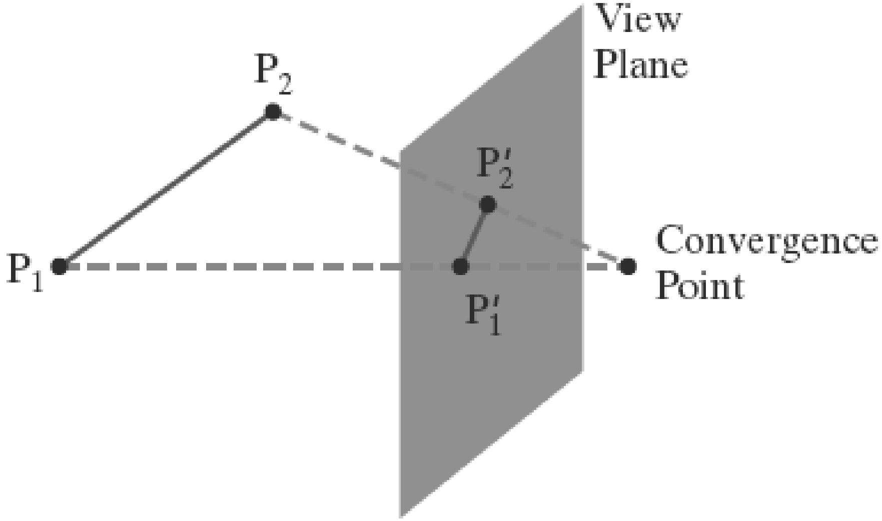


Orthographic Parallel Projections

oblique projections



- For a **perspective projection,** object positions are transformed to projection coordinates along lines that converge to a point behind the view plane.
- An example of a perspective projection for a straight-line segment, defined with endpoint coordinates **P**1 and **P**2.



- But perspective views of a scene are more realistic because distant objects in the projected display are reduced in size.

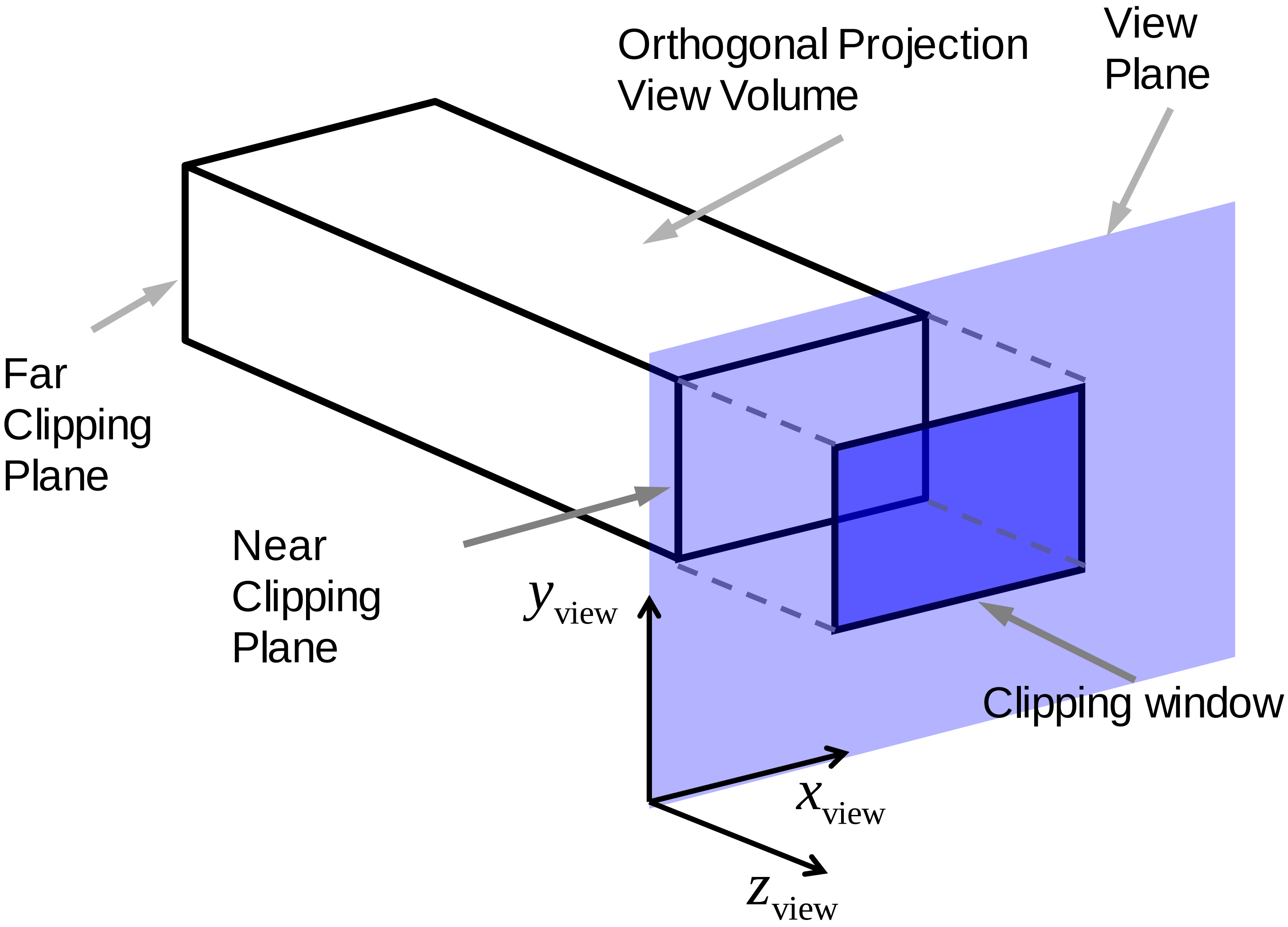
Orthogonal Projections

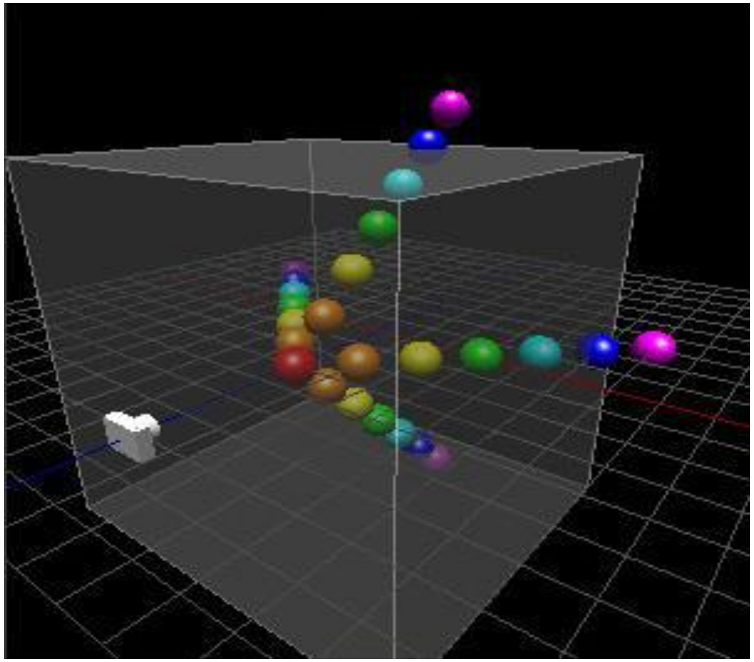
- A transformation of object descriptions to a view plane along lines that are all parallel to the view-plane normal vector **N** is called an **orthogonal projection** (or, equivalently, an **orthographic projection**).
- This produces a parallel-projection transformation in which the projection lines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

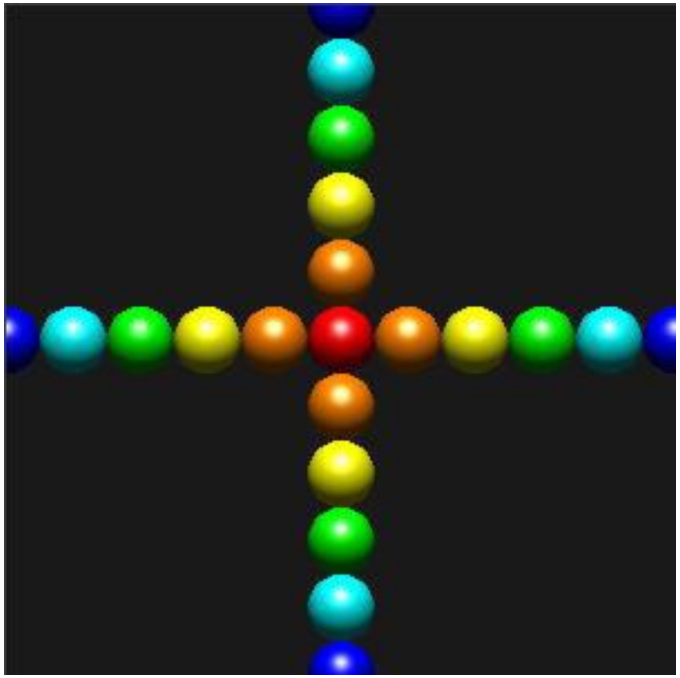
Axonometric and Isometric Orthogonal Projections:

- We can also form orthogonal projections that display more than one face of an object.
- Such views are called **axonometric** orthogonal projections.
- The most commonly used axonometric projection is the **isometric** projection, which is generated by aligning the projection plane (or the object) so that the plane intersects each coordinate axis in which the object is defined, called the *principal axes,* at the same distance from the origin.

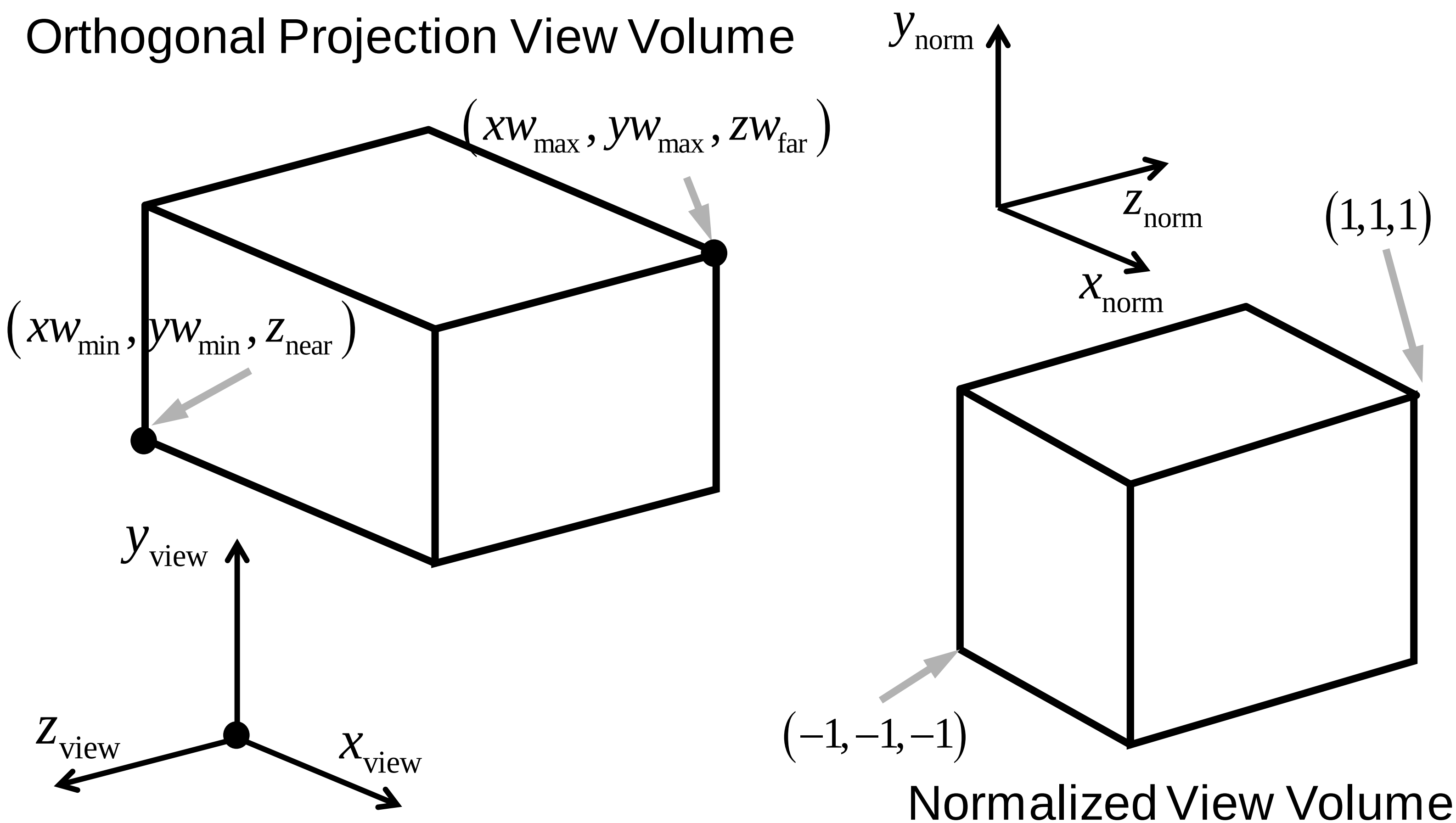
Clipping Window and View Volume







Normalizing Orthogonal Projection



Orthogonal Projection View Volume

Normalized View Volume

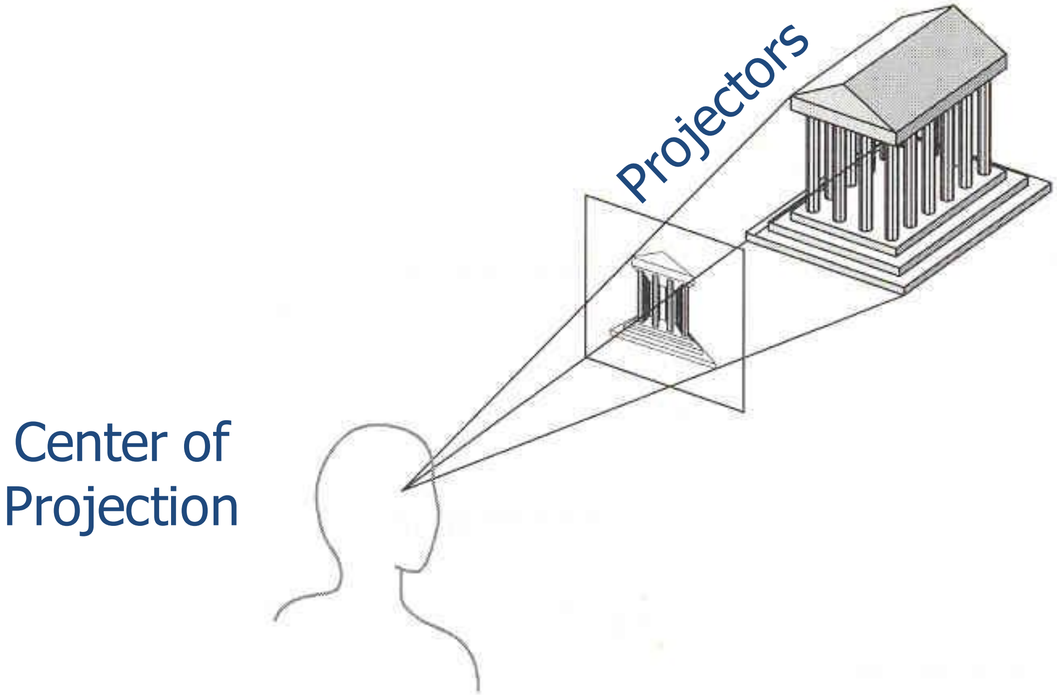
$$\mathbf{M}_{\text{ortho,norm}} =$$

$$\begin{bmatrix} \dfrac{2}{xw_{\max} - xw_{\min}} & 0 & 0 & -\dfrac{xw_{\max} + xw_{\min}}{xw_{\max} - xw_{\min}} \\ 0 & \dfrac{2}{yw_{\max} - yw_{\min}} & 0 & -\dfrac{yw_{\max} + yw_{\min}}{yw_{\max} - yw_{\min}} \\ 0 & 0 & \dfrac{-2}{z_{\text{near}} - z_{\text{far}}} & \dfrac{z_{\text{near}} + z_{\text{far}}}{z_{\text{near}} - z_{\text{far}}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The complete transformation from world coordinated to normalized orthogonal-projection coordinates is obtained by $\mathbf{M}_{\text{ortho,norm}} \cdot \mathbf{M}_{\text{WC,VC}}$.

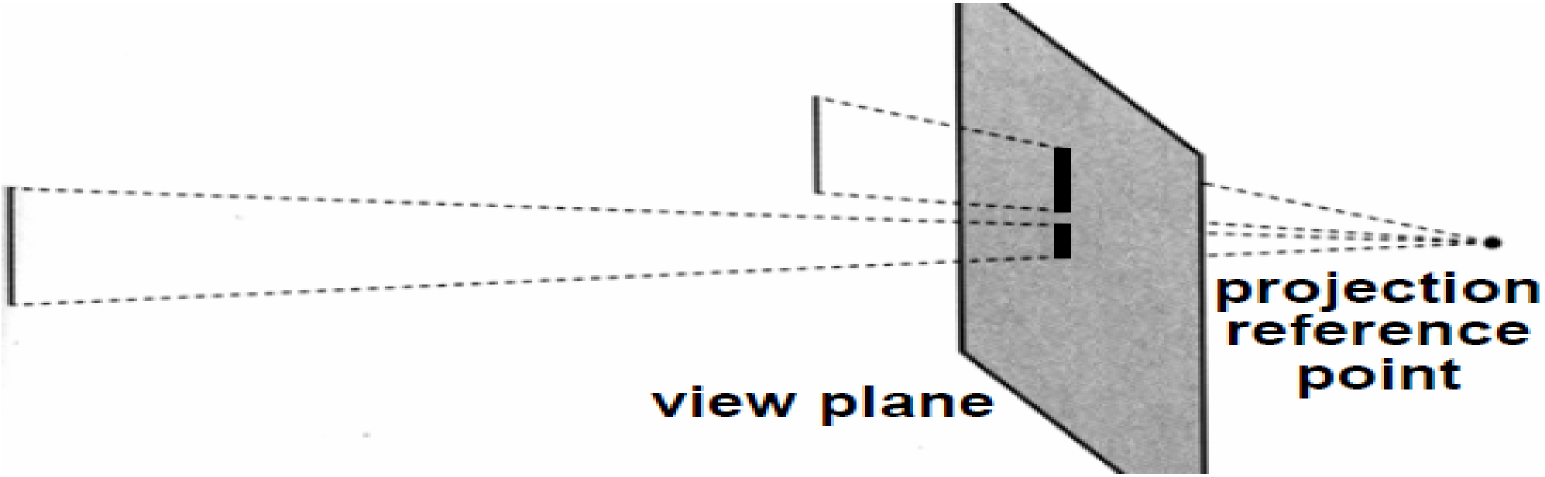
Perspective Projections

- Although a parallel-projection view of a scene is easy to generate and preserves relative proportions of objects, it does not provide a realistic representation.



Perspective Projection

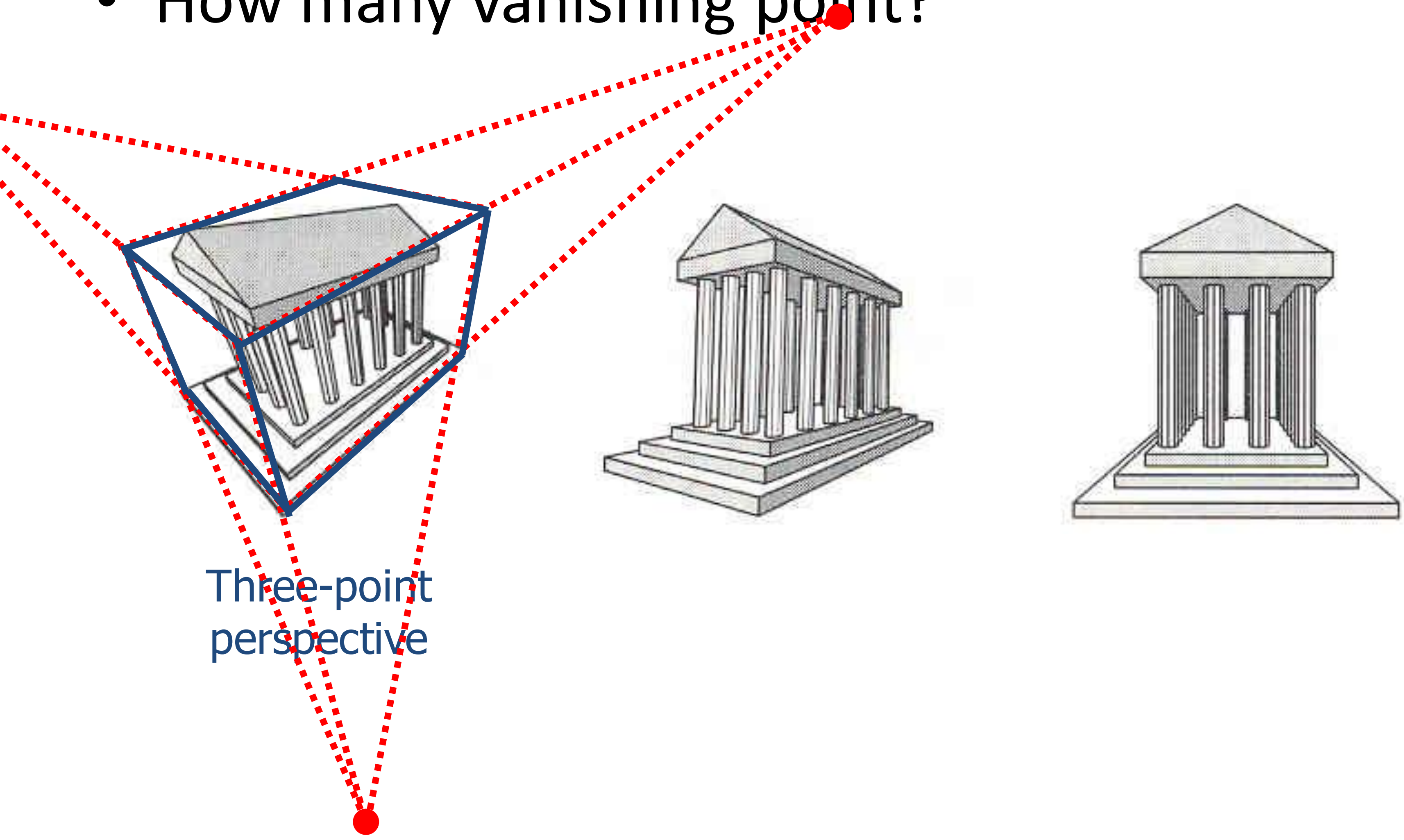
- Map points onto “view plane” along “projectors” emanating from “center of projection”(cop)



perspective projection of equal-sized objects at different distances from the view plane

Perspective Projection

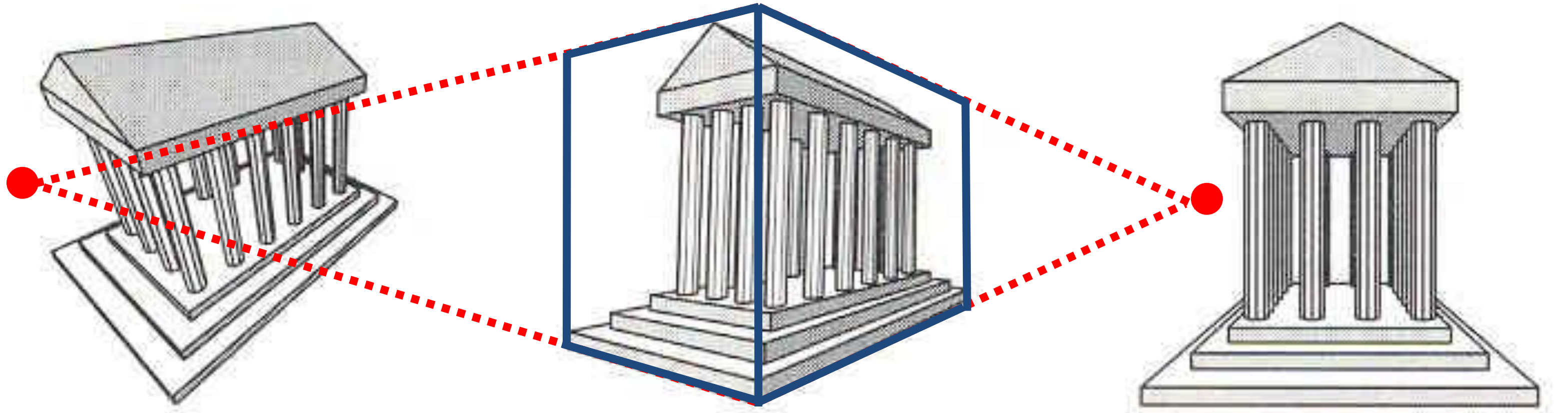
- How many vanishing point?



Three-point perspective

Perspective Projection

- How many vanishing point?

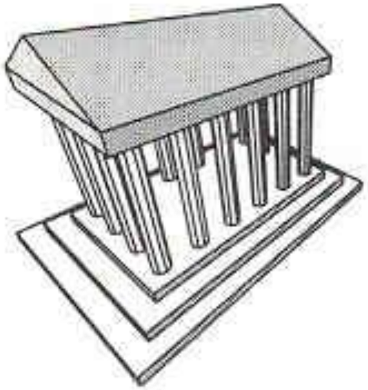


Three-point perspective

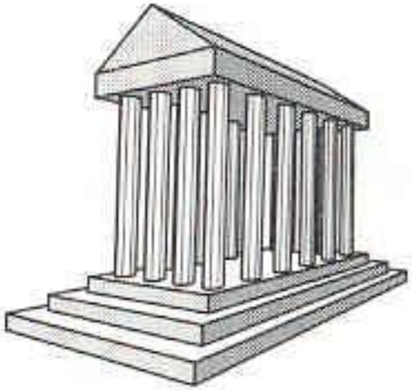
Two-point perspective

Perspective Projection

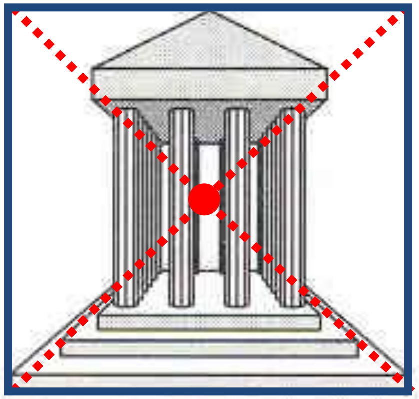
- How many vanishing point?



Three-point perspective



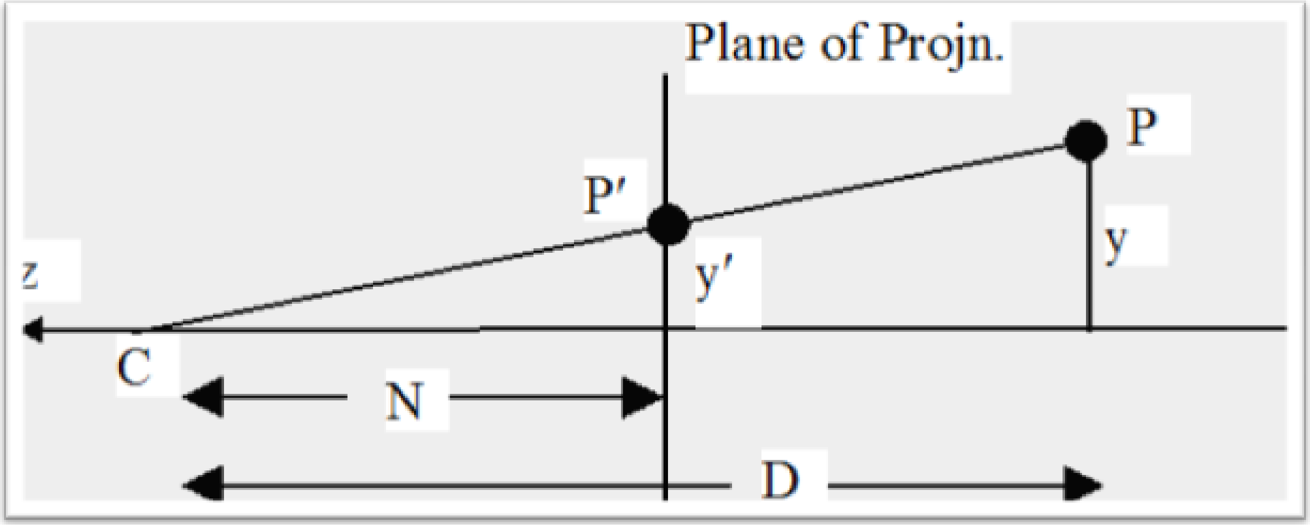
Two-point perspective



One-point perspective

Perspective Projection

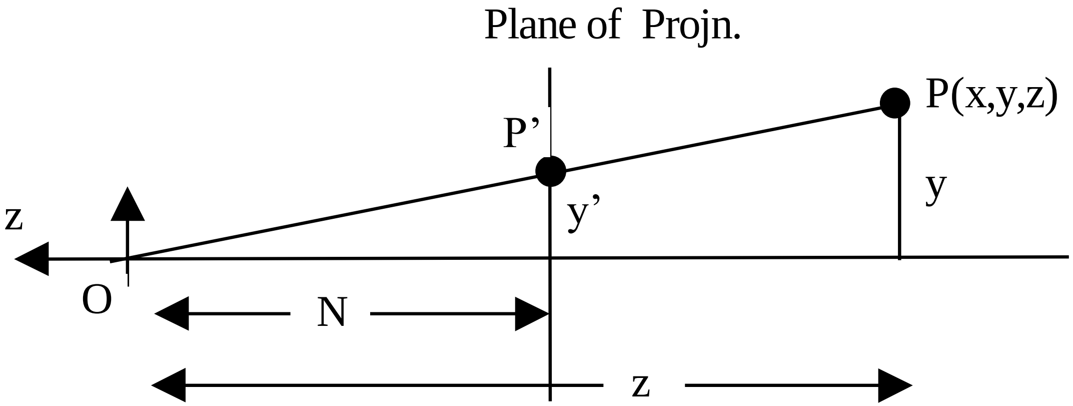
The projectors intersect at a Center of Projection C.



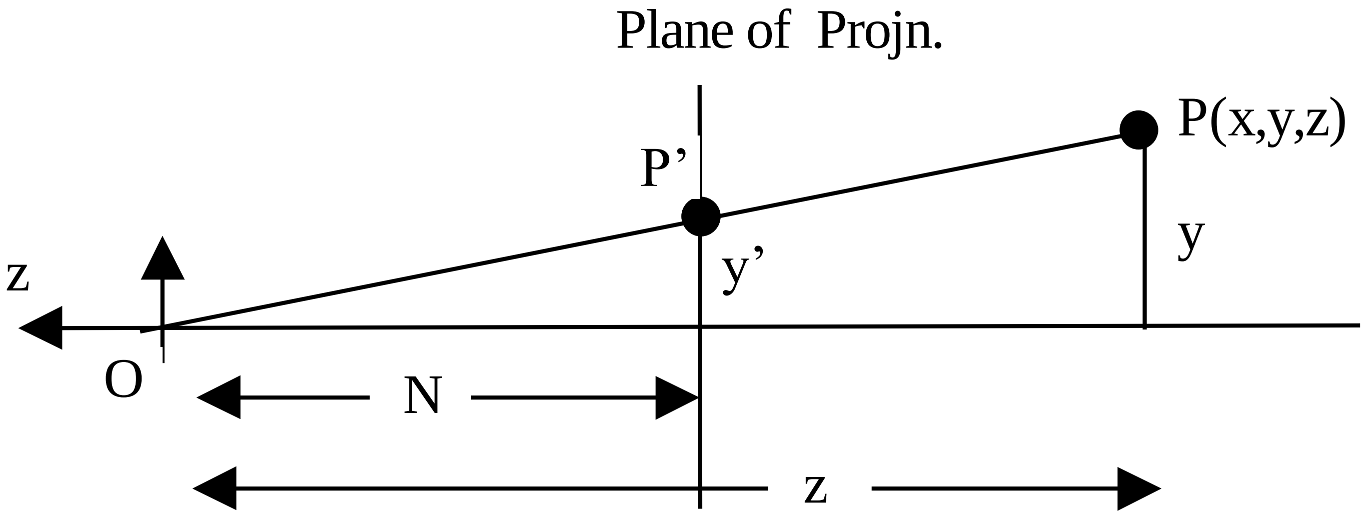
$$\frac{y}{D} = \frac{y'}{N} \qquad \frac{x}{D} = \frac{x'}{N}$$

Perspective Projection

- For convenience of deriving projection equations, we assume that the center of projection is at the origin.
- The object and the plane of projection are now on the negative z side.
- The distance D then refers to the z-coordinate of the object point (towards –z).
- The plane of projection has the equation z = –N.



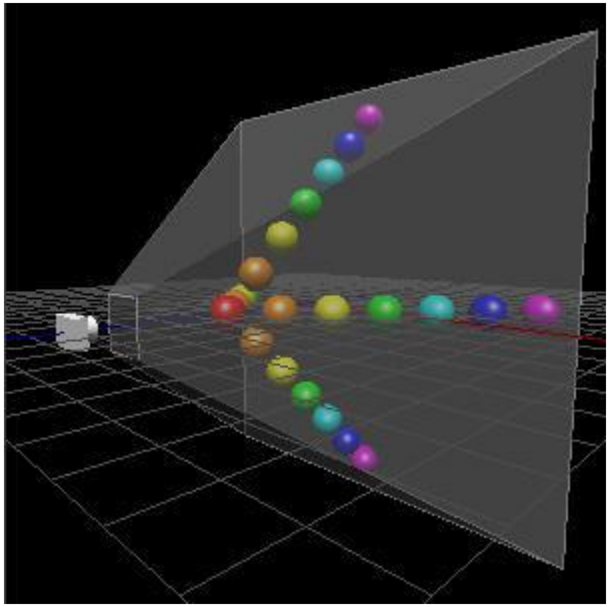
Perspective Projection

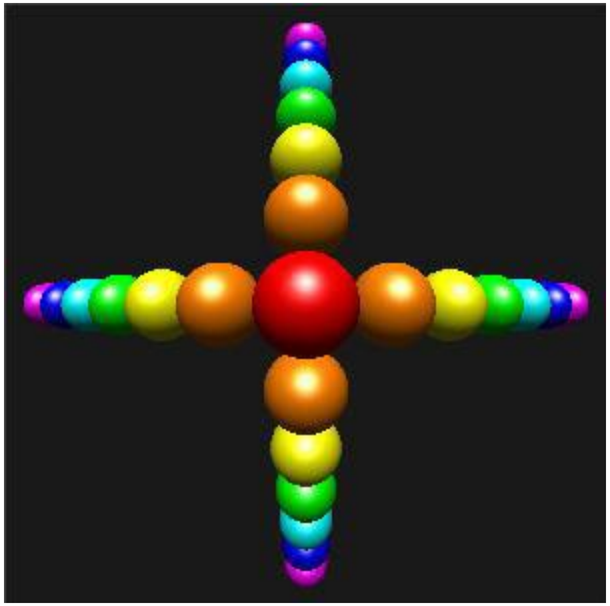


$$(x, y, z) \rightarrow \left(\frac{Nx}{-z}, \frac{Ny}{-z} \right)$$

Perspective Projection Matrix:

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





OpenGL 3D Viewing Functions

- The viewing parameters (camera position, view-up vector, view-plane normal) are specified as part of modeling transformations. A matrix is formed and concatenated with the current modelview matrix. So, to set up camera parameters:

glMatrixMode (GL_MODELVIEW);

gluLookAt (x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);

N = P0 --Pref

Viewing direction is along the –zview axis

- If we do not invoke the gluLookAt function. The default camera parameters are:
  - **P0= (0,0,0)**
  - **Pref= (0,0,-1)**
  - **V = (0,1,0)**

GL_MODELVIEW matrix is a combination of Model and View matrices ($M_{view}.M_{model}$).

- M_{model} is to construct objects from “object space/local space” to “world space”.
- M_{view} is to convert objects from “world space” to “eye space” (camera).

General PerspectiveProjection Function

glFrustum (xwmin, xwmax, ywmin, ywmax, dnear, dfar)

- Projection reference point is the viewing (camera position) and the view plane normal is the zview axis, viewing direction is -zview . If xwmin != -xwmax or ywmin!=-ywmax we can obtain an oblique perspective-projection frustum. Otherwise, we will have a symmetric perspective projection.

S J P N Trust's

HIRASUGAR INSTITUTE OF TECHNOLOGY, Nidasoshi.

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

Module-4
18CS62

3D Viewing and Visible Surface Detection:

Course Outcome

Apply the concepts of viewing and visible surface detection of 3D objects

Prof. Rahul Palakar

Visible Surface Detection Methods

Classification of Visible-Surface Detection Algorithms

Two types of methods used:

- **Object-space methods (Continuous):** Compares parts of objects to each other to determine which surfaces should be labeled to determine as visible (use of bounding boxes, and check limits along each direction).

Order the surfaces being drawn, such that it provides the correct impression of depth variations and positions.

Image Space methods (discrete):

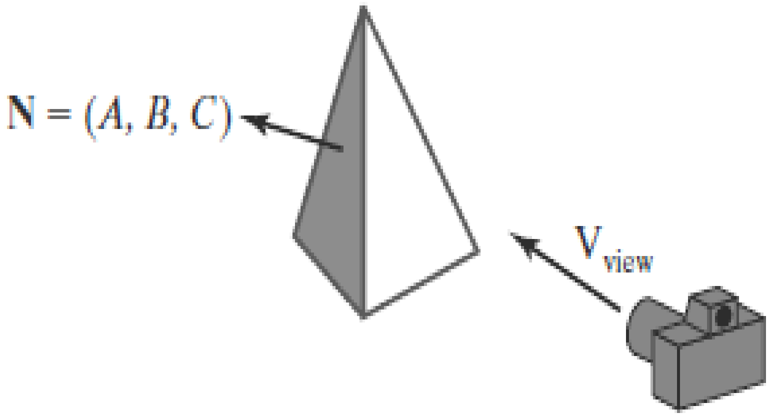
- Visibility is decided point by point at Visibility is decided point by point at each pixel position on the projection plane. Screen resolution can be a limitation.

Back-Face Detection

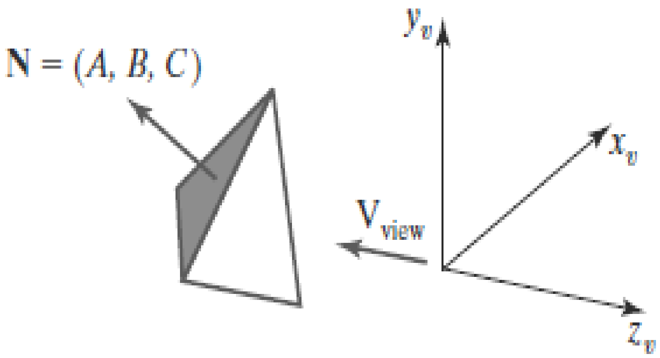
- A fast and simple object-space method for locating the **back faces** of a polyhedron is based on front-back tests. A point (*x*, *y*, *z*) is behind a polygon surface if

$$\boldsymbol{Ax + By + Cz + D < 0}$$

- We can simplify the back-face test by considering the direction of the normal vector **N** for a polygon surface.
- If $\mathbf{V}_{\text{view}}$ is a vector in the viewing direction from our camera position, then a polygon is a back face if $\mathbf{V}_{\text{view}} \cdot \mathbf{N} > 0$



- In a right-handed viewing system with the viewing direction along the negative z_v axis, a polygon is a back face if the *z* component, *C*, of its normal vector **N** satisfies *C* < 0.



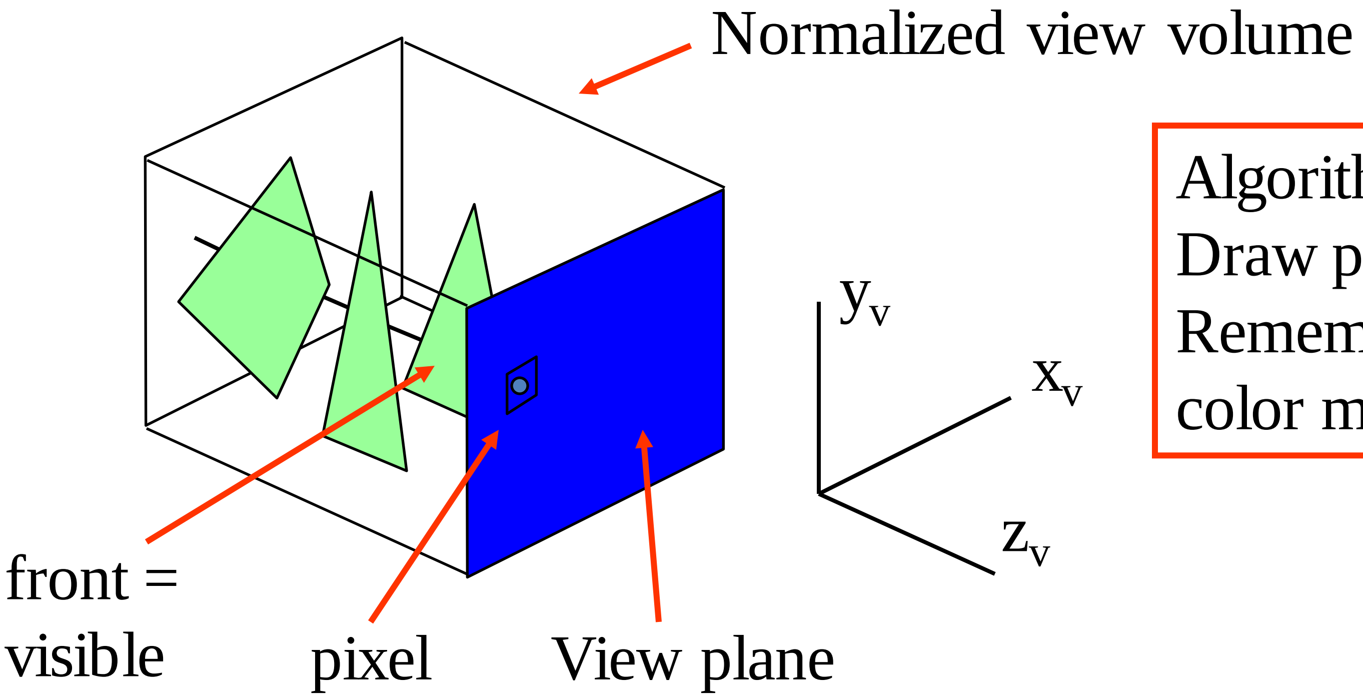
Depth-Buffer (Z-buffer)

- The basic idea is to test the z-depth of each surface to determine the closest (visible) surface

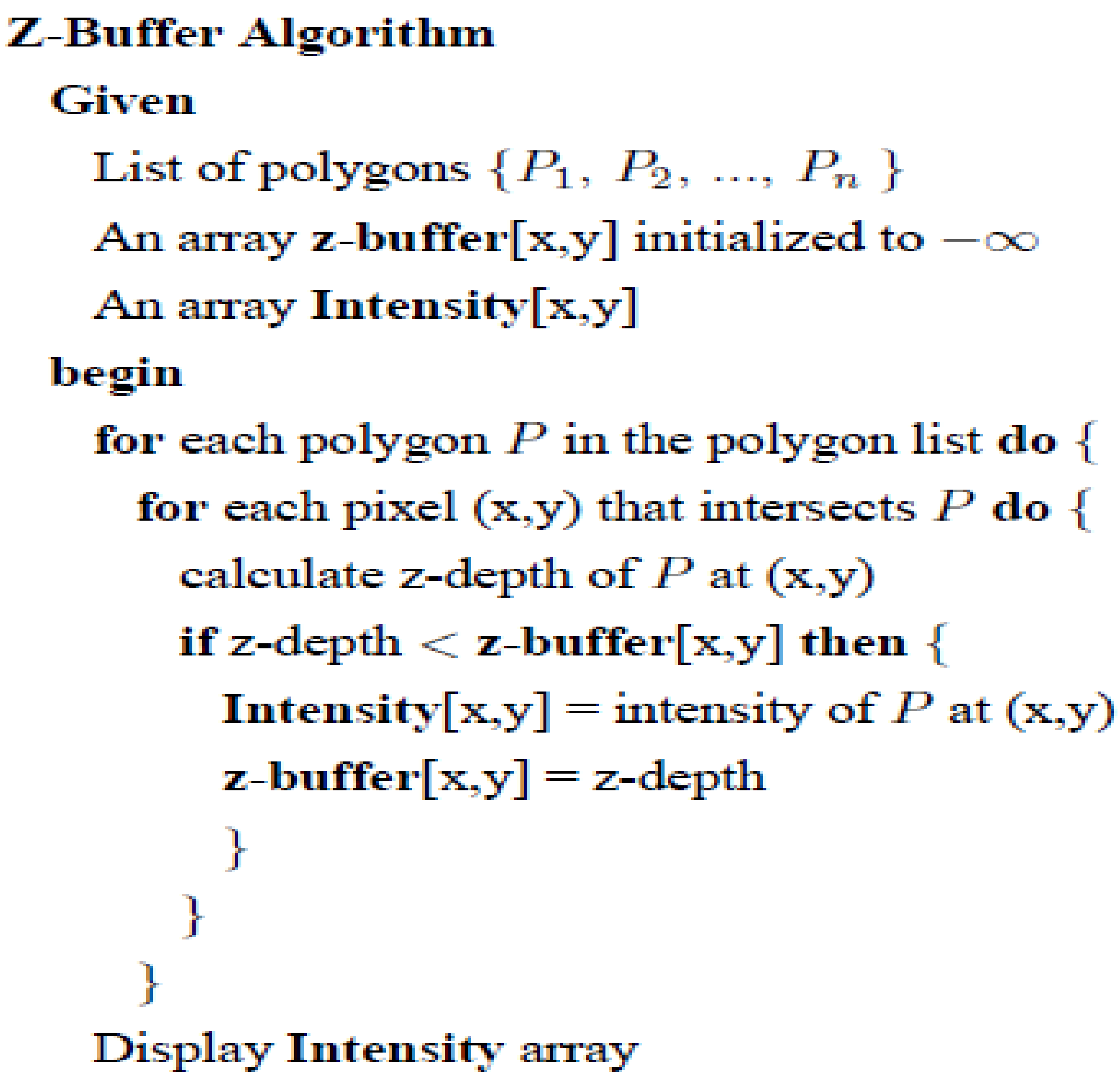
Use two buffers

- frame buffer : image value (intensity)
- depth buffer (z-buffer) : z-value

- Image-space method
- Aka *z-buffer algorithm*



Algorithm:
Draw polygons,
Remember the
color most in front.



```
Z-Buffer Algorithm
  Given
    List of polygons $\{P_1, P_2, ..., P_n\}$
    An array z-buffer[x,y] initialized to $-\infty$
    An array Intensity[x,y]
  begin
    for each polygon $P$ in the polygon list do {
      for each pixel (x,y) that intersects $P$ do {
        calculate z-depth of $P$ at (x,y)
        if z-depth $<$ z-buffer[x,y] then {
          Intensity[x,y] = intensity of $P$ at (x,y)
          z-buffer[x,y] = z-depth
          }
        }
      }
    Display Intensity array
```

Calculating z values of plane: $Ax + By + Cz + D = 0$

[WAY I] $$z = \frac{-Ax - By - D}{C}$$

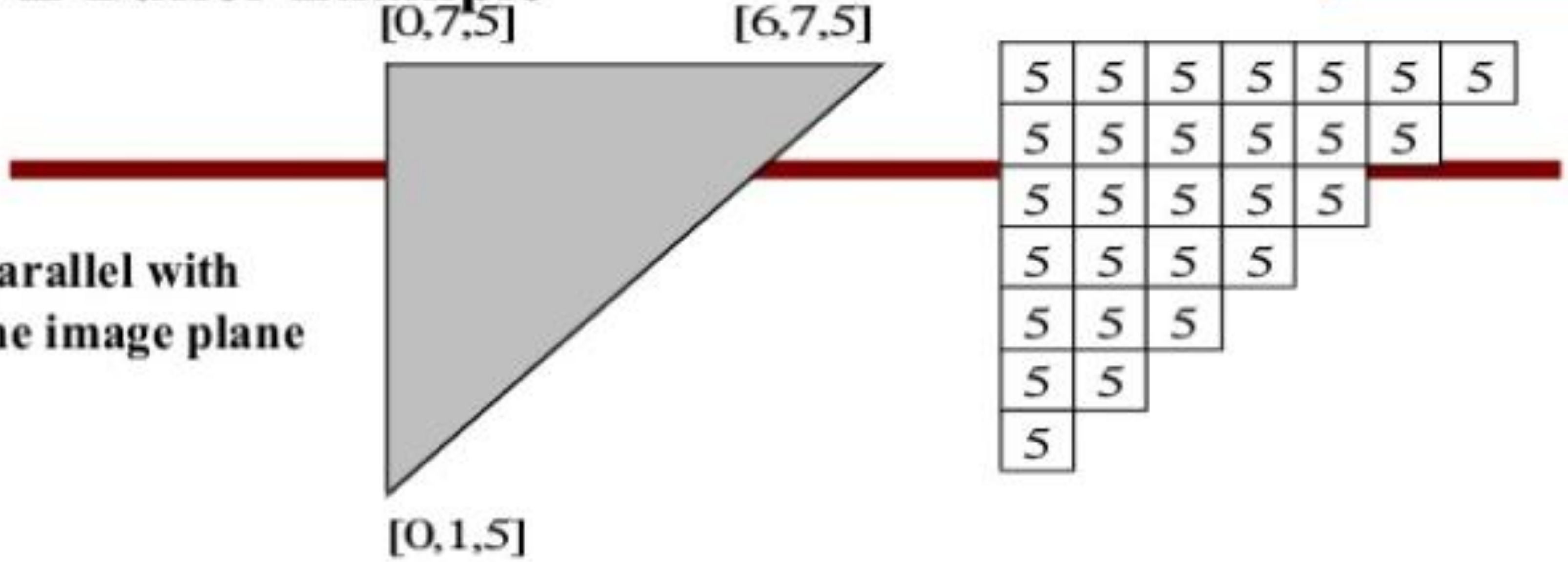
[WAY II] Use incremental calculation

$z_{(x,y)}$: the depth of position (x, y)

$$z_{(x+1,y)} = \frac{-A(x+1) - By - D}{C} = z_{(x,y)} - \frac{A}{C}$$

$$z_{(x,y+1)} = \frac{-Ax - B(y+1) - D}{C} = z_{(x,y)} - \frac{B}{C}$$

Z-Buffer Example



Parallel with the image plane

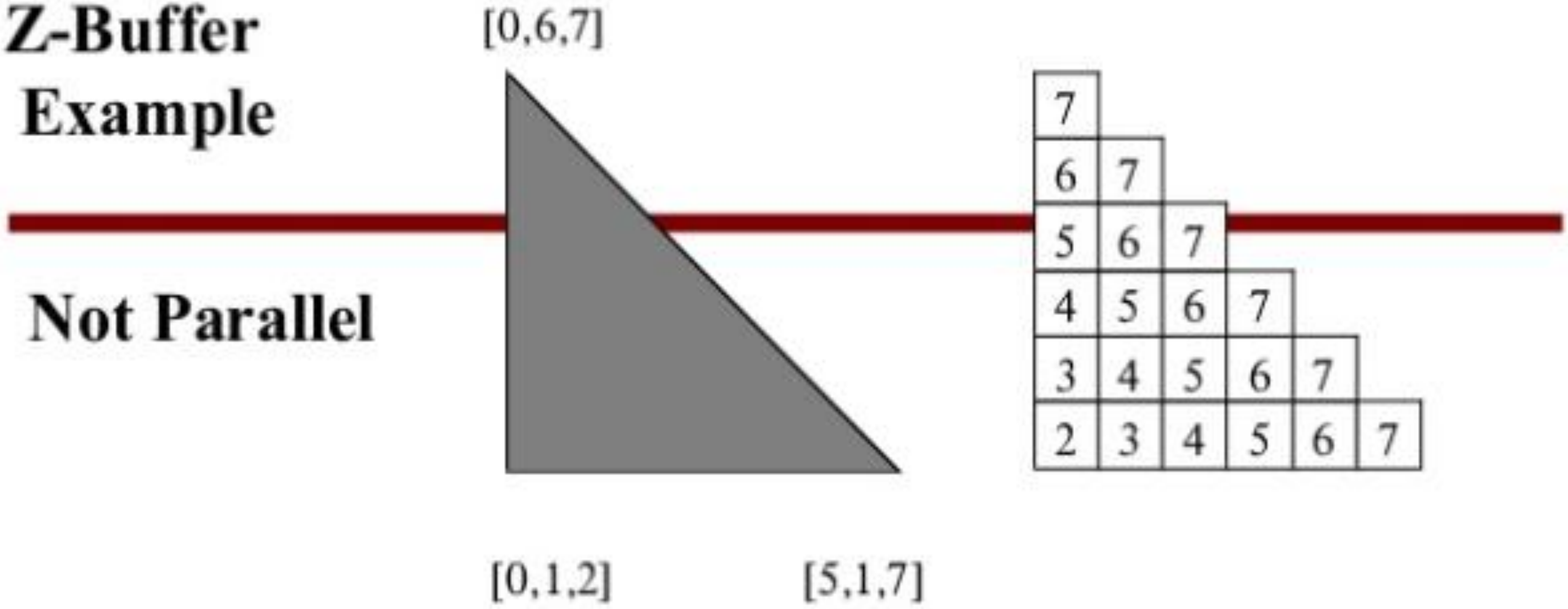
5	5	5	5	5	5	5
5	5	5	5	5	5	
5	5	5	5	5		
5	5	5	5			
5	5	5				
5	5					
5						

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	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

▨	▨	▨	▨	▨	▨	▨	
▨	▨	▨	▨	▨	▨		
▨	▨	▨	▨	▨			
▨	▨	▨	▨				
▨	▨	▨					
▨	▨						
▨							

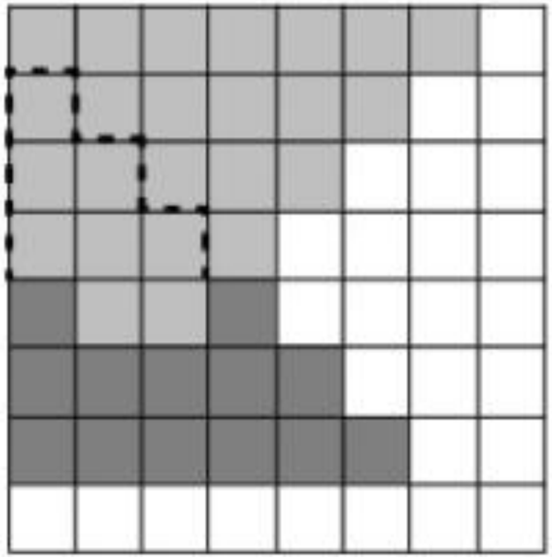
Z-Buffer Example

Not Parallel



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	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞



THANK YOU