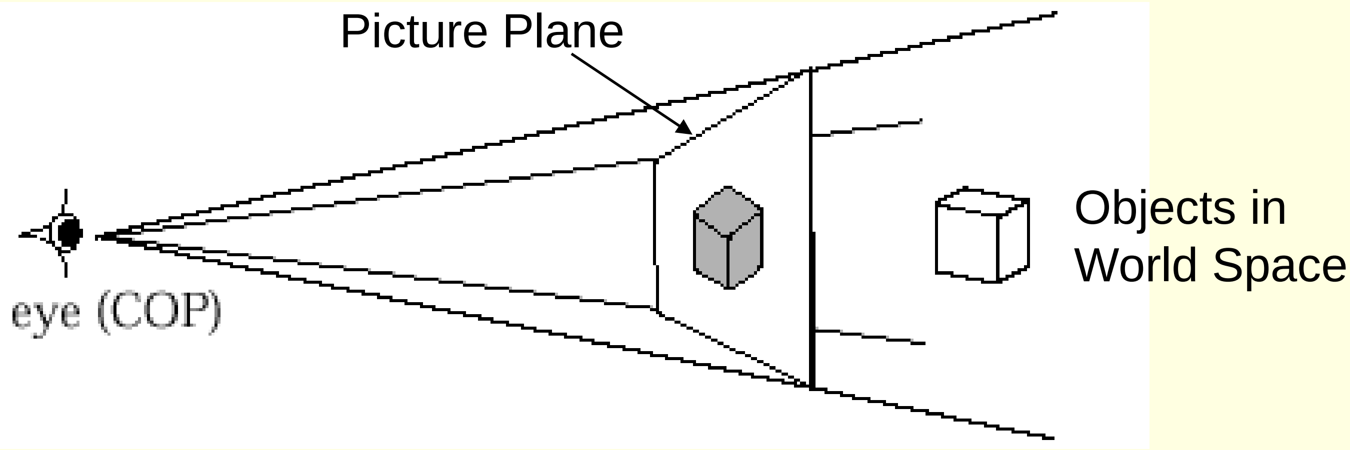
Computer Graphics Viewing

What Are Projections?

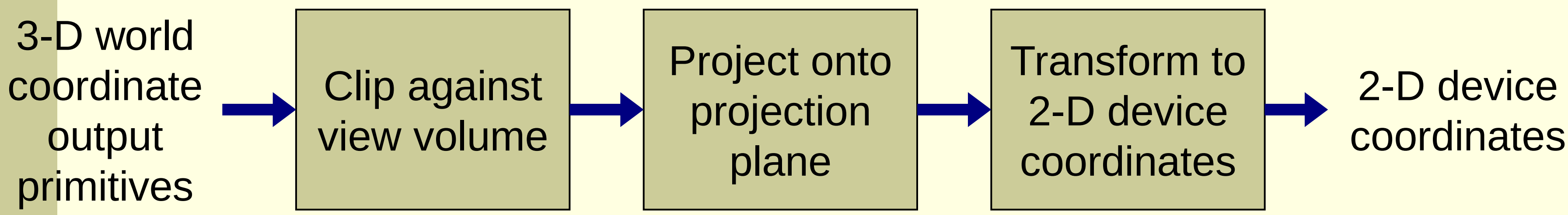
- Our 3-D scenes are all specified in 3-D world coordinates
- To display these we need to generate a 2-D image - *project* objects onto a *picture plane*



- So how do we figure out these projections?

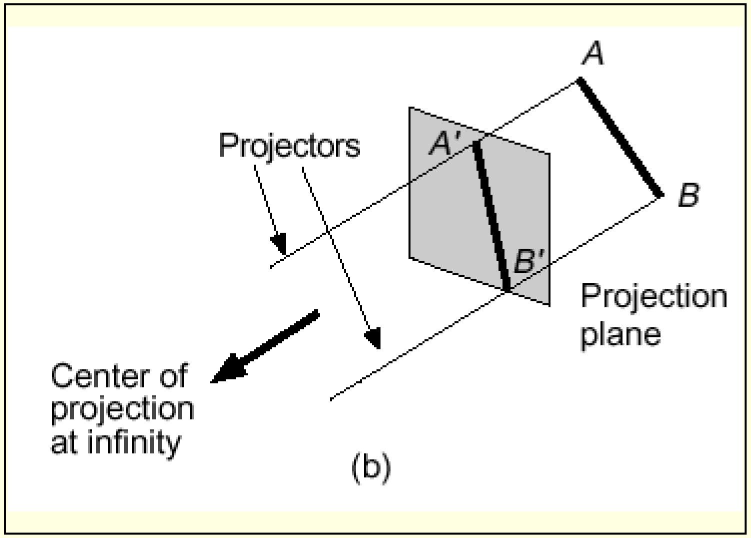
Converting From 3-D To 2-D

- Projection is just one part of the process of converting from 3-D world coordinates to a 2-D image

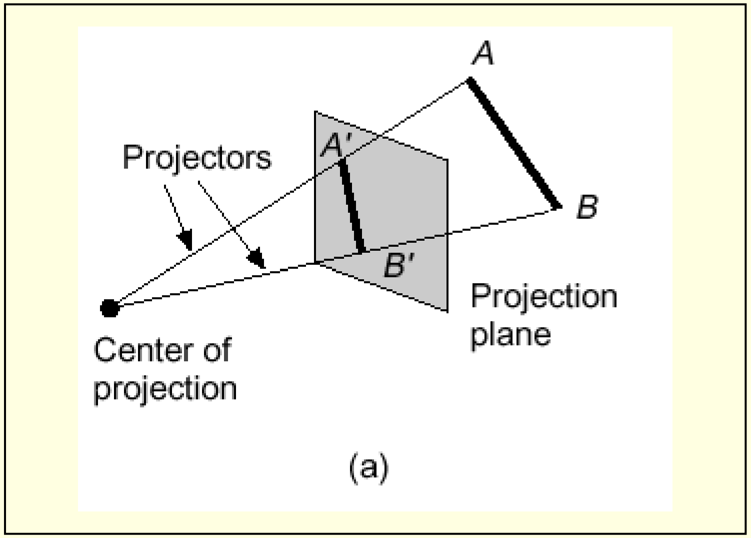


Types Of Projections

- There are two broad classes of projection:
  - Parallel: Typically used for architectural and engineering drawings
  - Perspective: Realistic looking and used in computer graphics



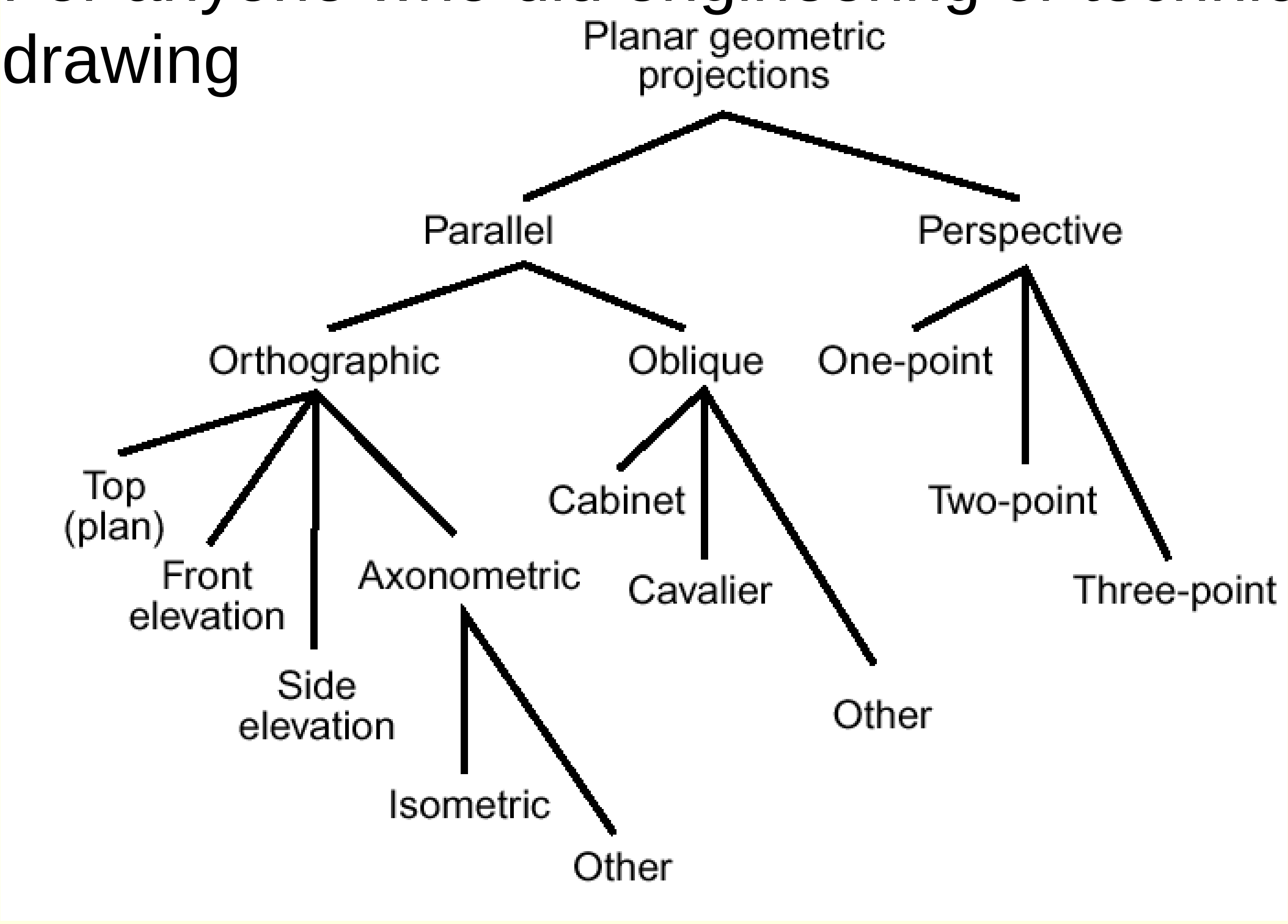
Parallel Projection



Perspective Projection

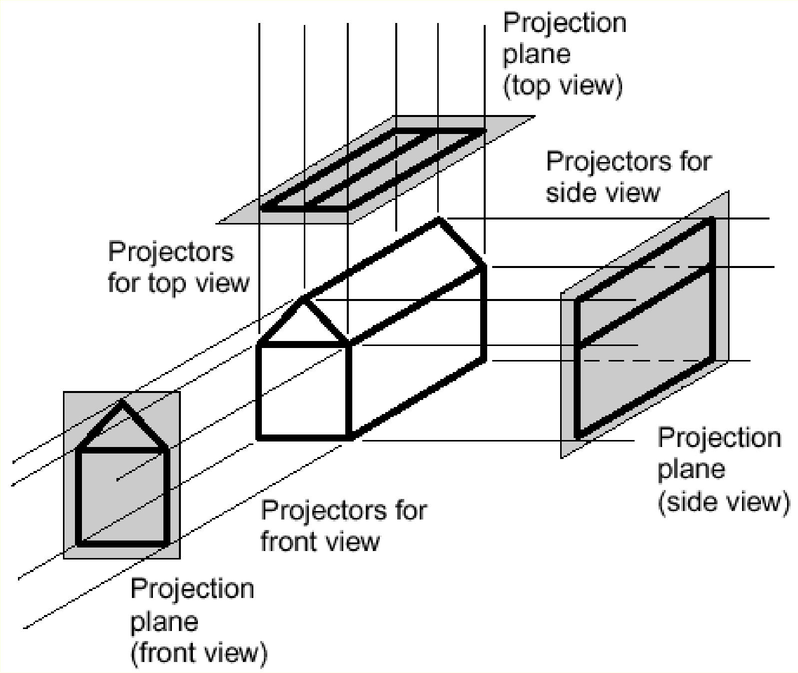
Types Of Projections (cont…)

- For anyone who did engineering or technical drawing

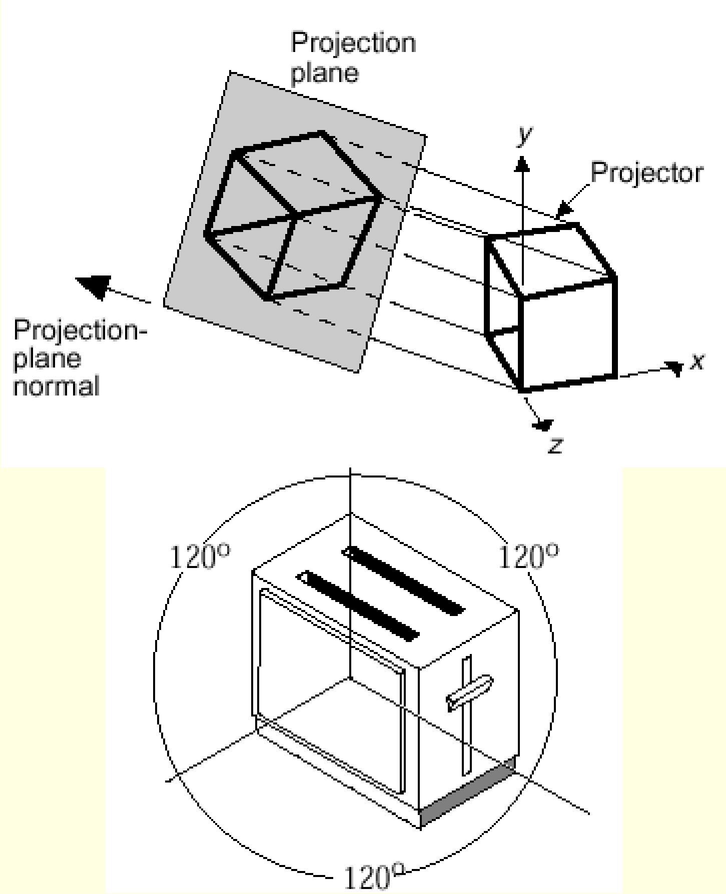


Parallel Projections

- Some examples of parallel projections



Orthographic Projection



Isometric Projection

Isometric Projections

- Isometric projections have been used in computer games from the very early days of the industry up to today

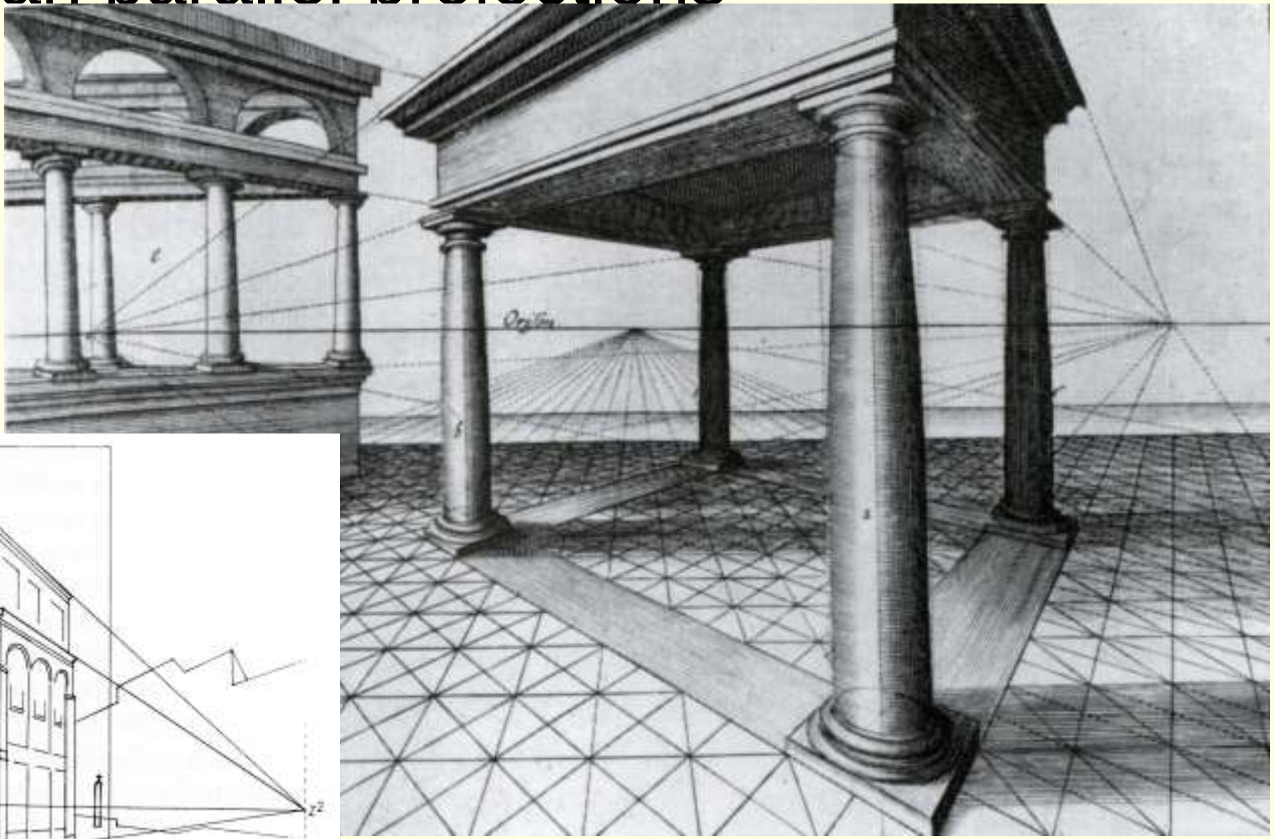
Q*Bert

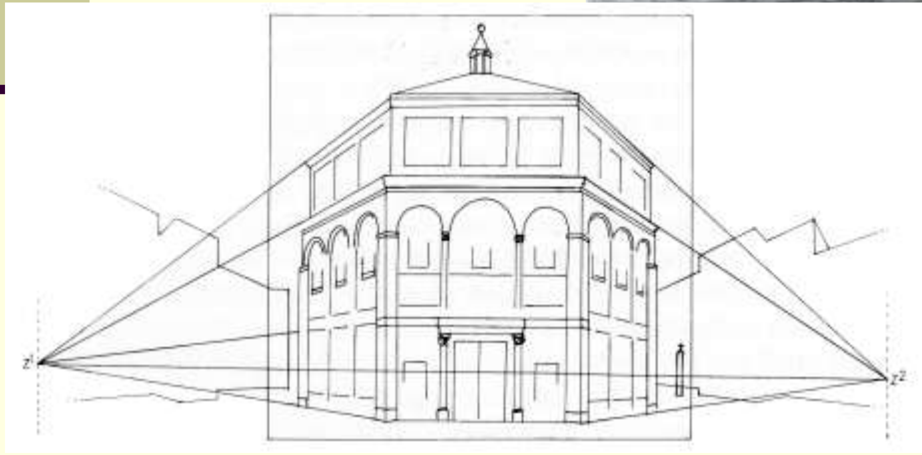
Sim City

Virtual Magic Kingdom

Perspective Projections

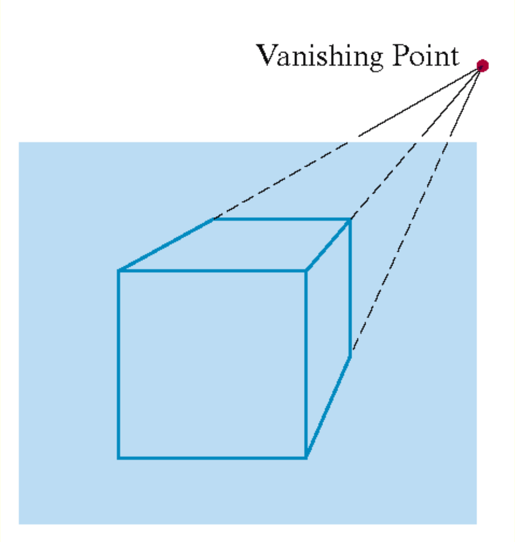
- Perspective projections are much more realistic than parallel projections



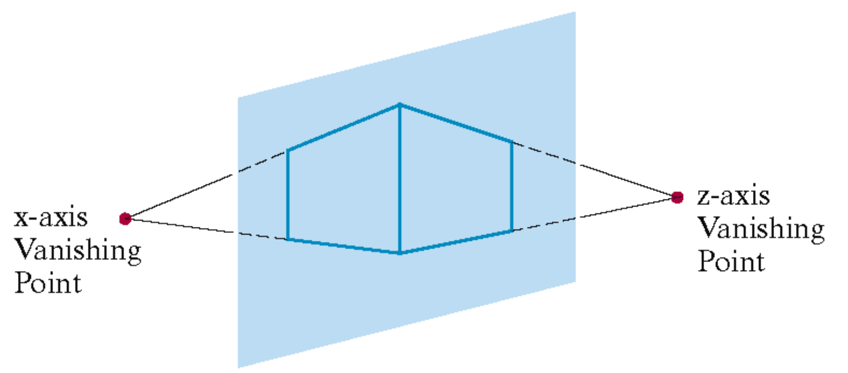


Perspective Projections

- There are a number of different kinds of perspective views
- The most common are one-point and two point perspectives

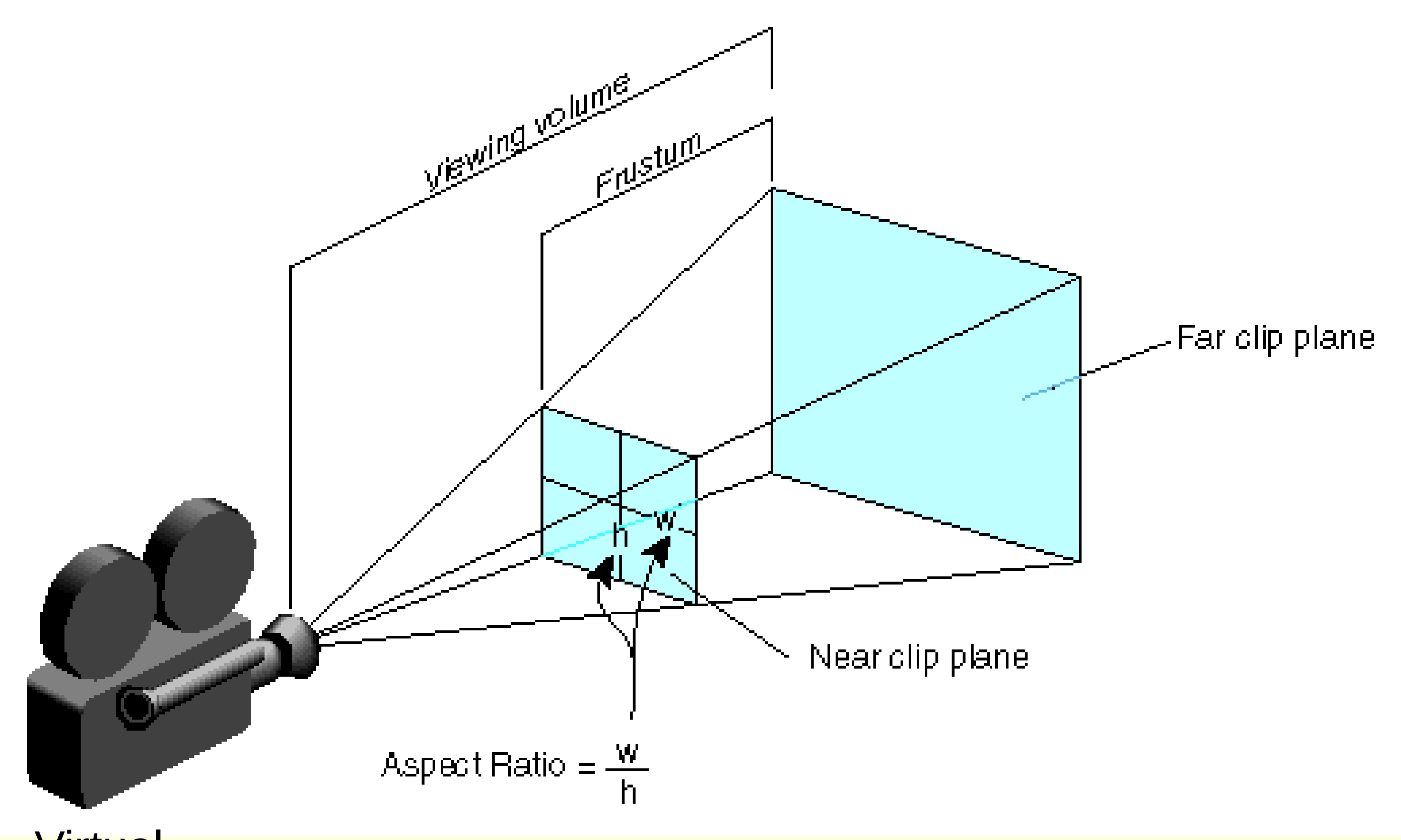


One Point Perspective Projection



Two-Point Perspective Projection

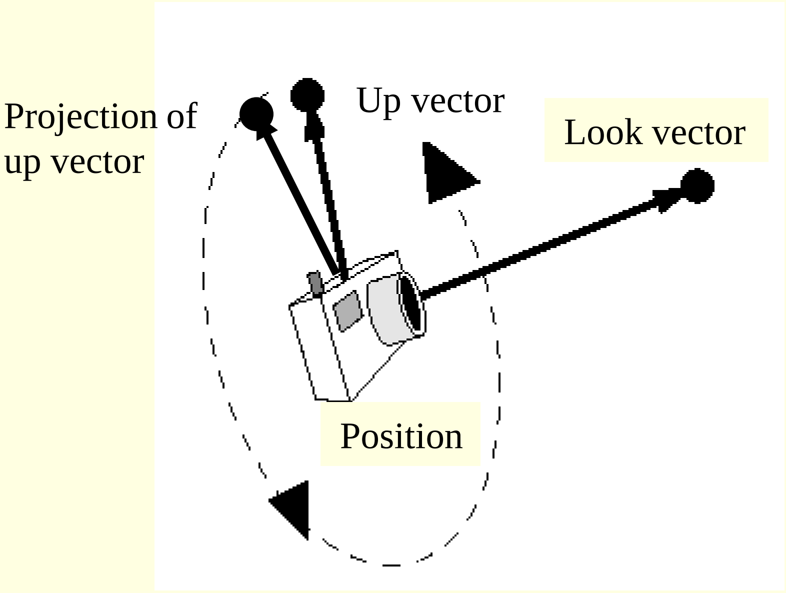
Elements Of A Perspective Projection



Virtual
Camera

The Up And Look Vectors

- The *look vector* indicates the direction in which the camera is pointing
- The *up vector* determines how the camera is rotated
- For example, is the camera held vertically or horizontally

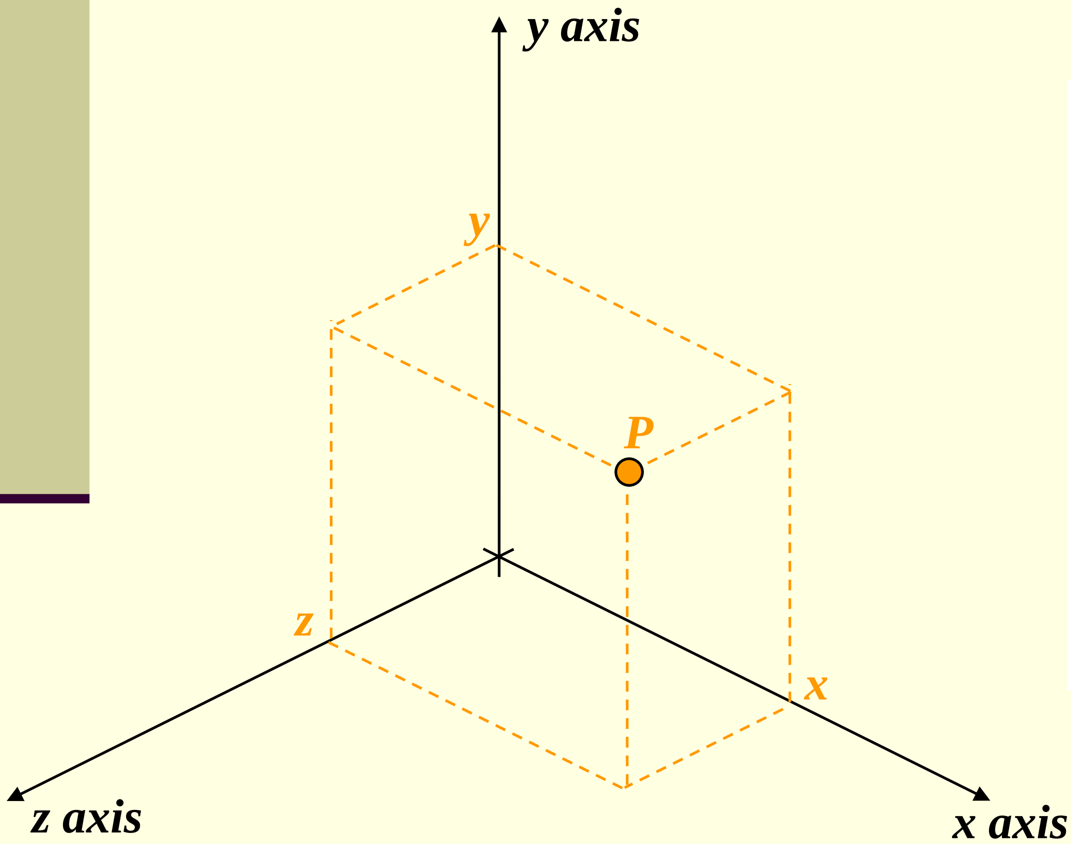


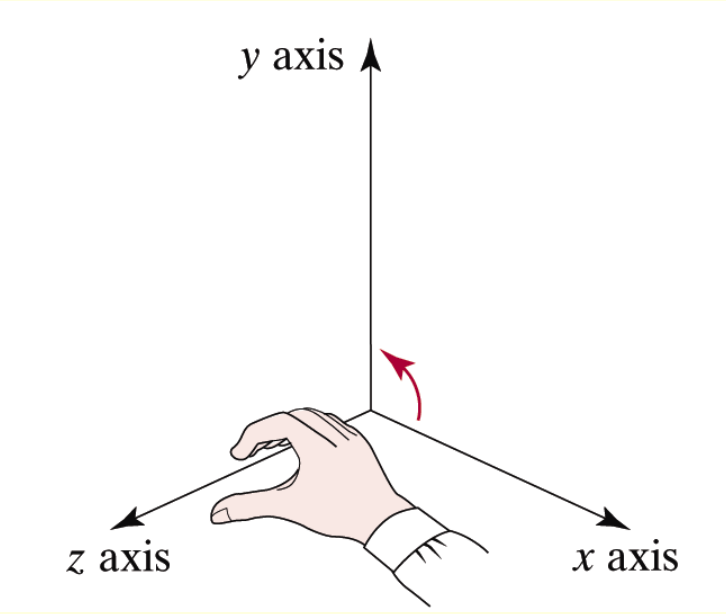
Contents

- In today's lecture we are going to have a look at:
  - Transformations in 3-D
    - How do transformations in 3-D work?
    - 3-D homogeneous coordinates and matrix based transformations
  - Projections
    - History
    - Geometrical Constructions
    - Types of Projection
    - Projection in Computer Graphics

3-D Coordinate Spaces

- Remember what we mean by a 3-D coordinate space

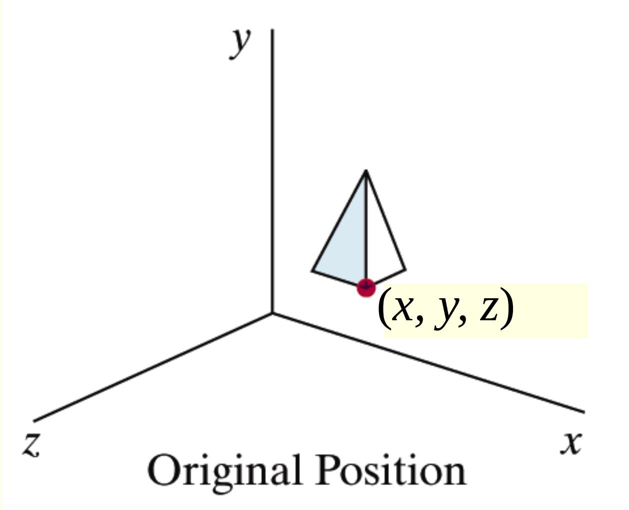


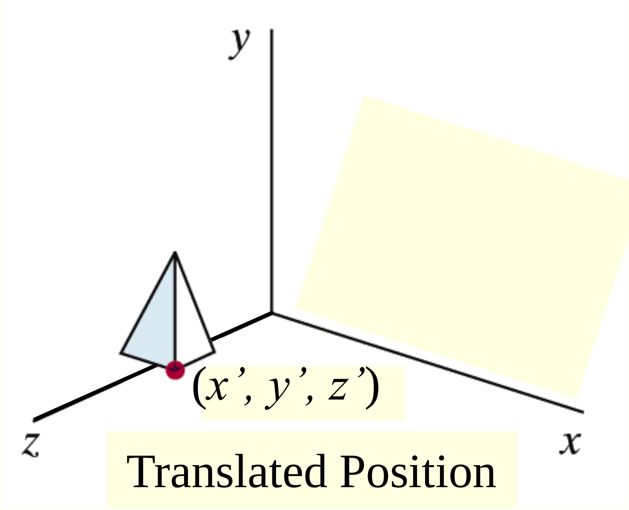


Right-Hand
Reference System

Translations In 3-D

- To translate a point in three dimensions by *dx*, *dy* and *dz* simply calculate the new points as follows:
  - $x' = x + dx \qquad y' = y + dy \qquad z' = z + dz$

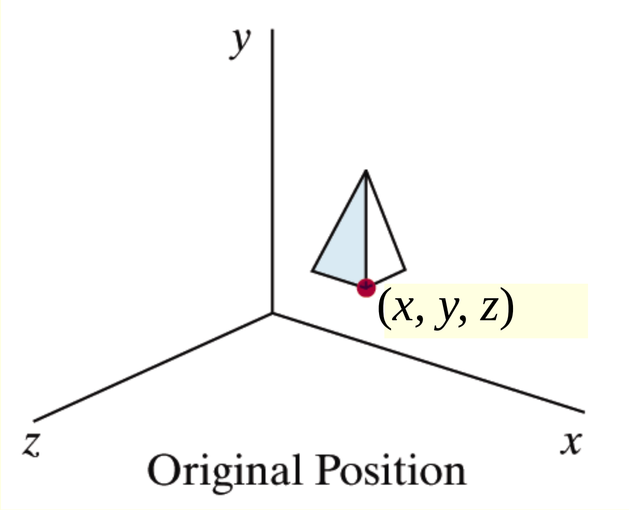




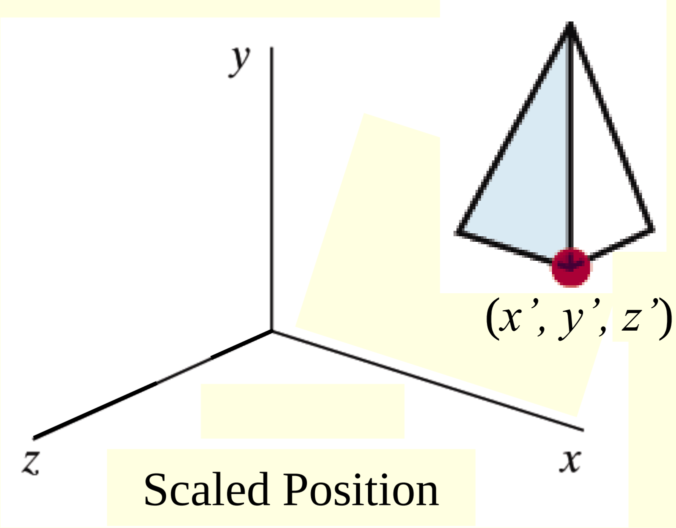
Scaling In 3-D

- To sale a point in three dimensions by *sx*, *sy* and *sz* simply calculate the new points as follows:

➢ $x' = sx*x \qquad y' = sy*y \qquad z' = sz*z$



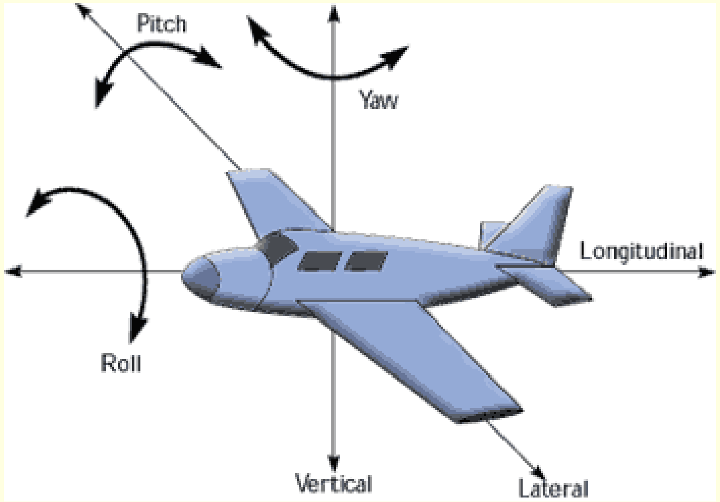
Original Position



Scaled Position

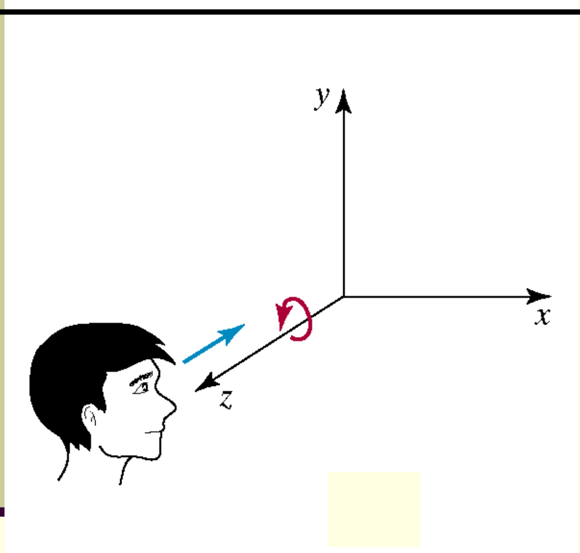
Rotations In 3-D

- When we performed rotations in two dimensions we only had the choice of rotating about the z axis
- In the case of three dimensions we have more options
  - Rotate about x – pitch
  - Rotate about y – yaw
  - Rotate about z - roll

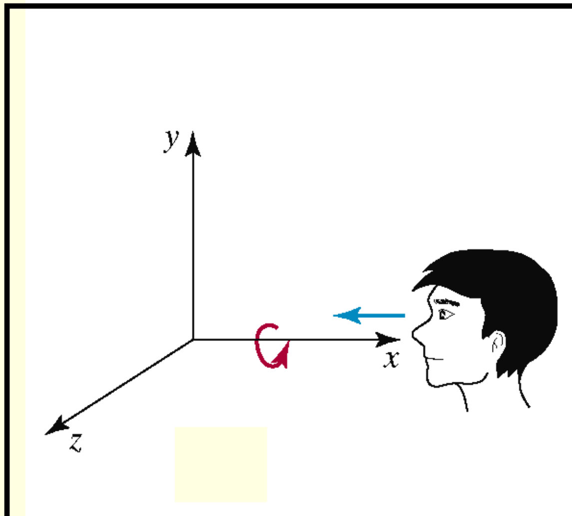


Rotations In 3-D (cont…)

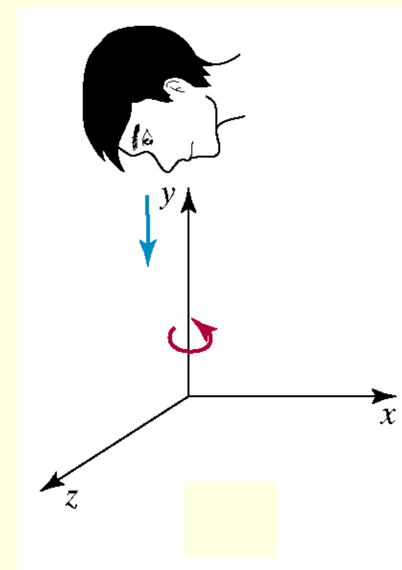
- The equations for the three kinds of rotations in 3-D are as follows:



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



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

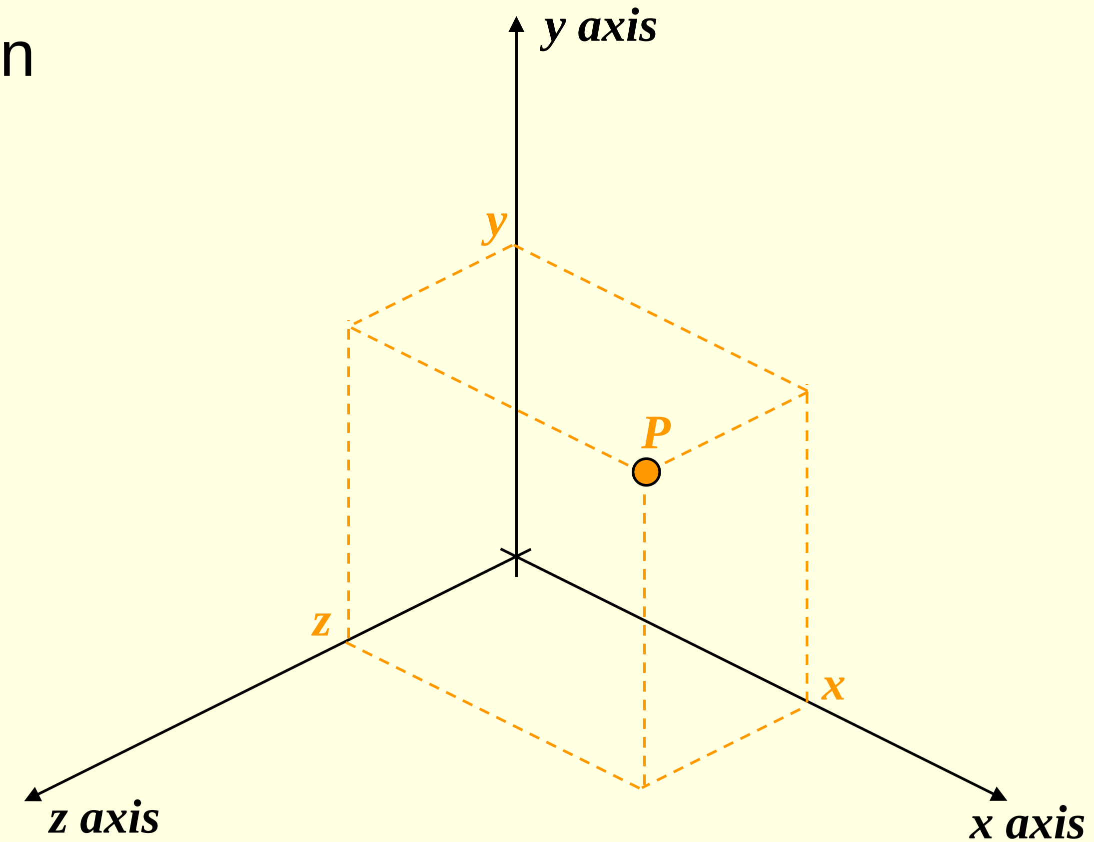


- $x' = z \cdot \sin\theta + x \cdot \cos\theta$
- $y' = y$
- $z' = z \cdot \cos\theta - x \cdot \sin\theta$

Homogeneous Coordinates In 3-D

- Similar to the 2-D situation we can use homogeneous coordinates for 3-D transformations - 4 coordinate column vector
- All transformations can then be represented as matrices

$$P(x, y, z) = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



3D Transformation Matrices

Translation by dx, dy, dz

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling by sx, sy, sz

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

Rotate About X-Axis

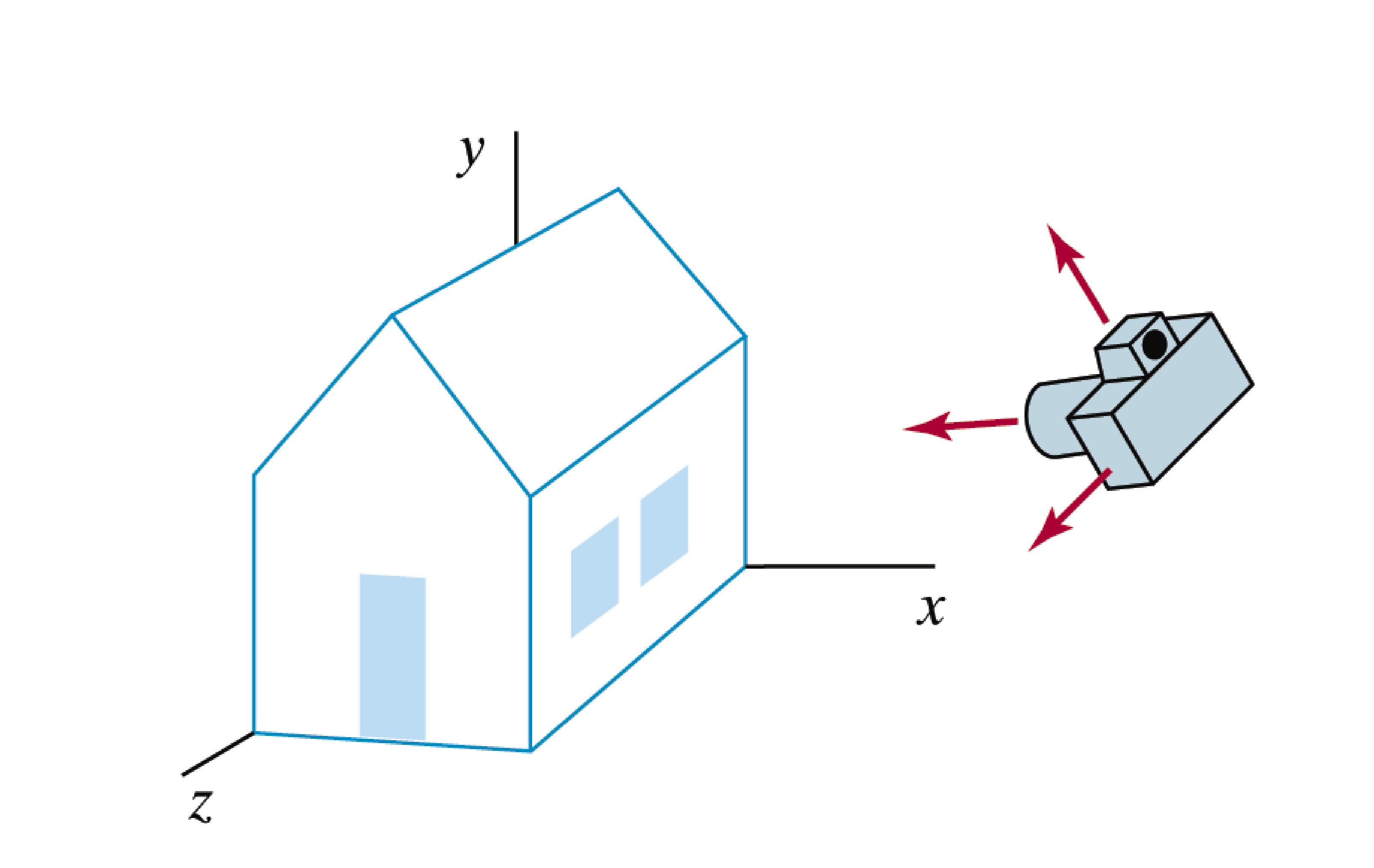
$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate About Y-Axis

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

Rotate About Z-Axis

Remember The Big Idea



Summary

- In today’s lecture we looked at:
  - Transformations in 3-D
    - Very similar to those in 2-D
  - Projections
    - 3-D scenes must be projected onto a 2-D image plane
    - Lots of ways to do this
      - Parallel projections
      - Perspective projections
    - The virtual camera

Who’s Choosing Graphics?

- A couple of quick questions for you:
  - Who is choosing graphics as an option?
  - Are there any problems with option time-tabling?
  - What do you think of the course so far?
    - Is it too fast/slow?
    - Is it too easy/hard?
    - Is there anything in particular you want to cover?

3D Transformations

- Same idea as 2D transformations
  - Homogeneous coordinates: (x,y,z,w)
  - 4x4 transformation matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation

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

Identity

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

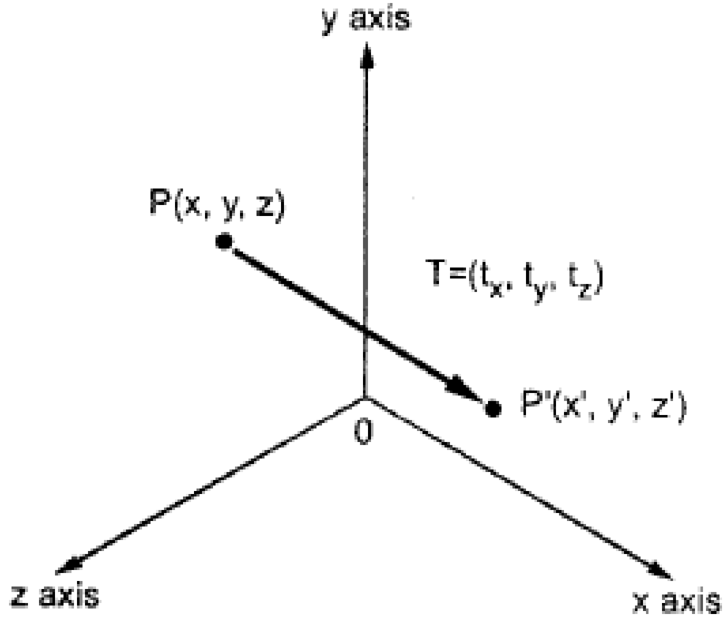
W=1 Translation

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

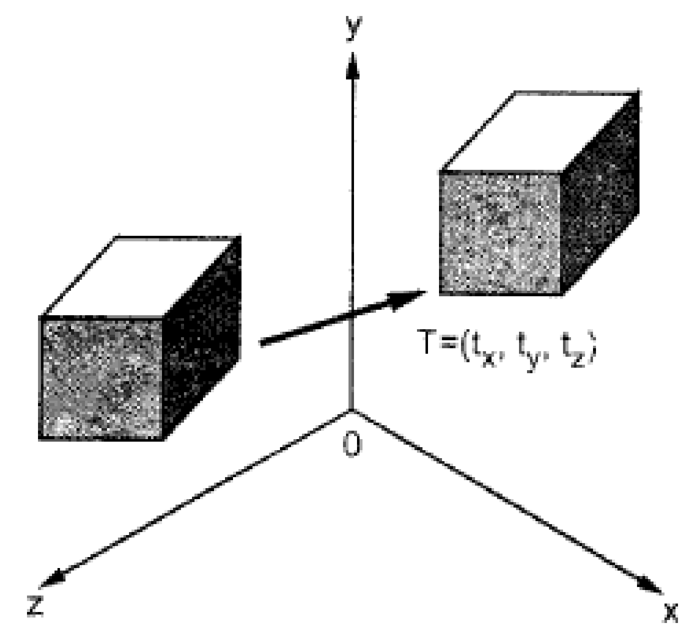
$$\therefore \quad P' = P \cdot T$$

$$\therefore [x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [x + t_x \quad y + t_y \quad z + t_z \quad 1]$$



(a) Translating point



(b) Translating object

Scaling

Column Vector Representation

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

Identity

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

Scale

Row Vector Representation

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

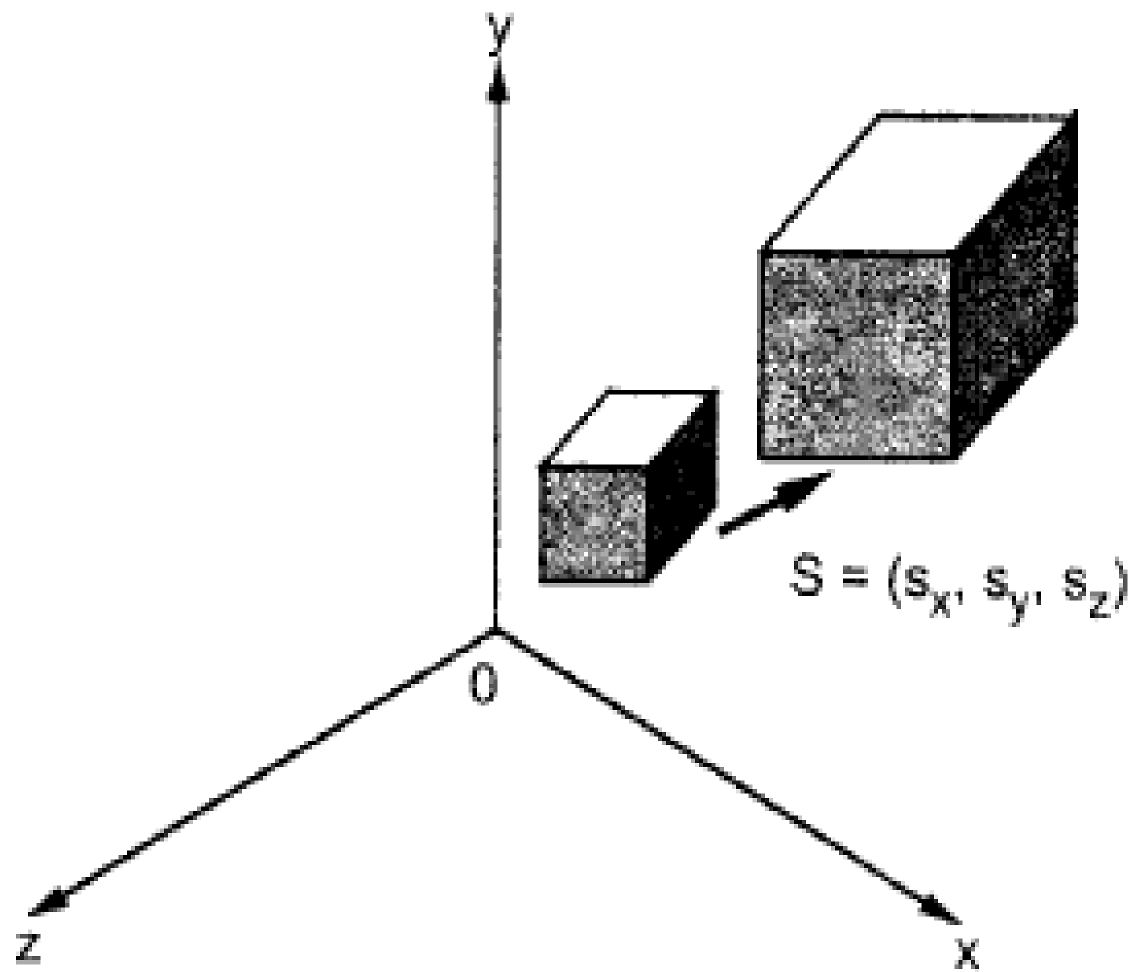
It specifies three coordinates with their own scaling factor.

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \qquad P' = P \cdot S$$

$$[x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[X \cdot S_x \quad y \cdot S_y \quad z \cdot S_z \quad 1 \right]$$


y
S = (sx, sy, sz)
0
z
x

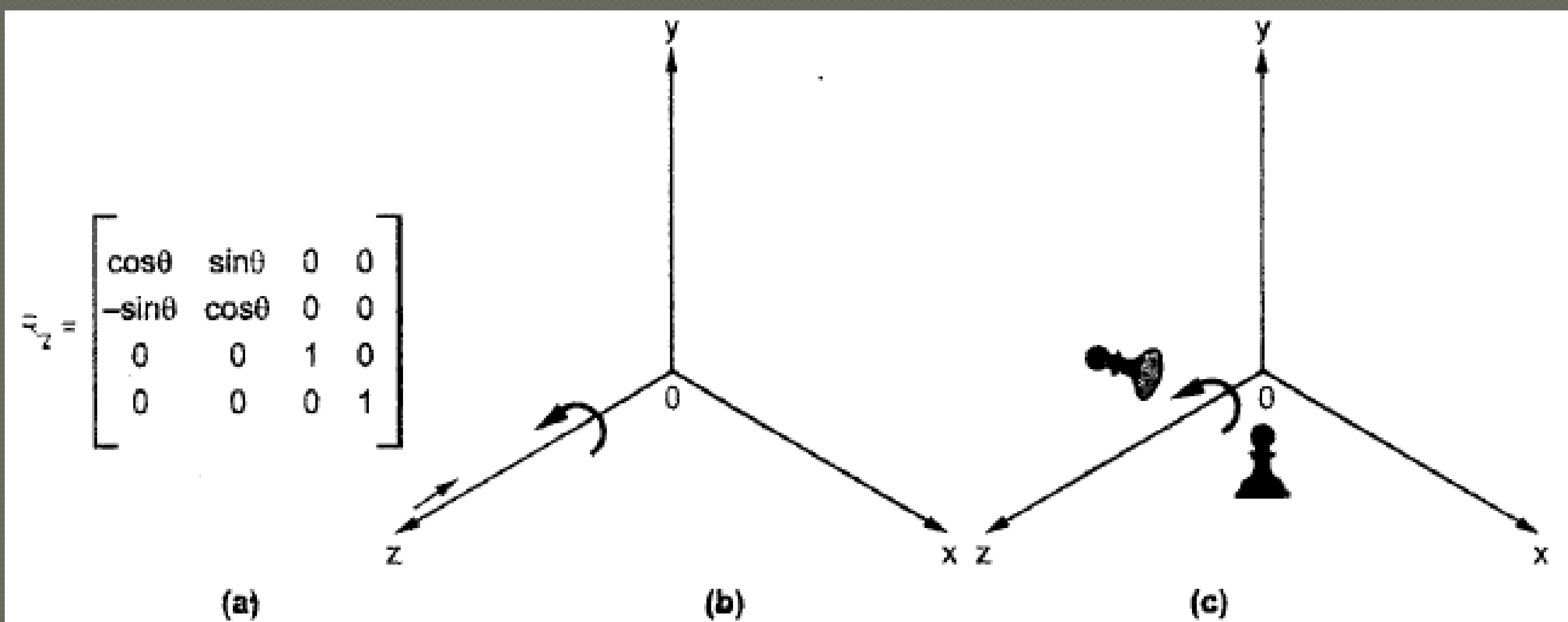
Rotation

Column Vector Representation

Rotate around Z axis:

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

Row Vector Representation



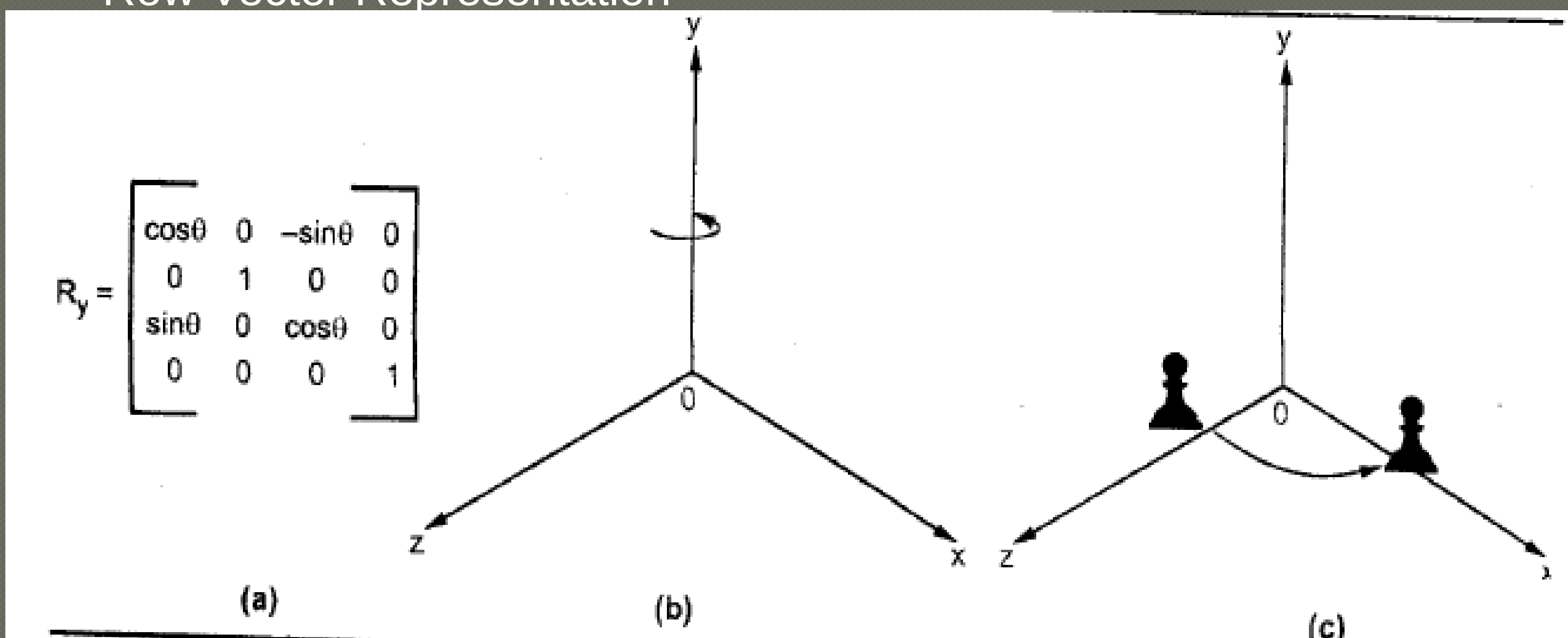
Rotation

Column Vector Representation

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

Rotate around Y axis:

Row Vector Representation



Rotation

Column Vector Representation

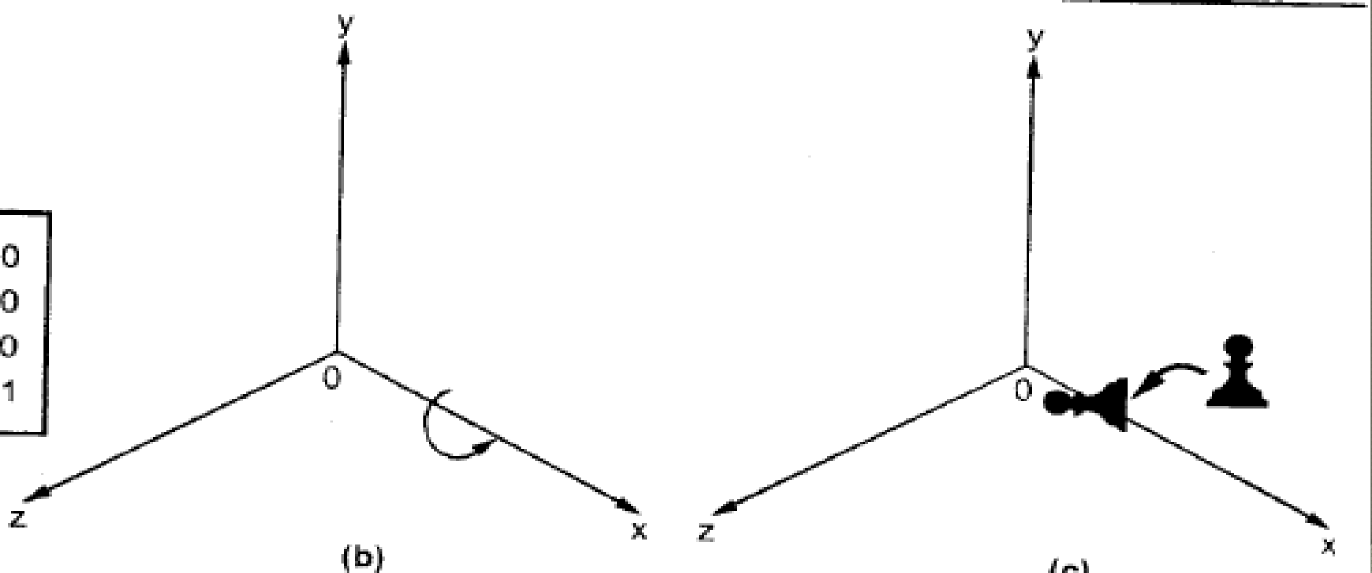
Rotate around X axis:

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

Row Vector Representation

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a)

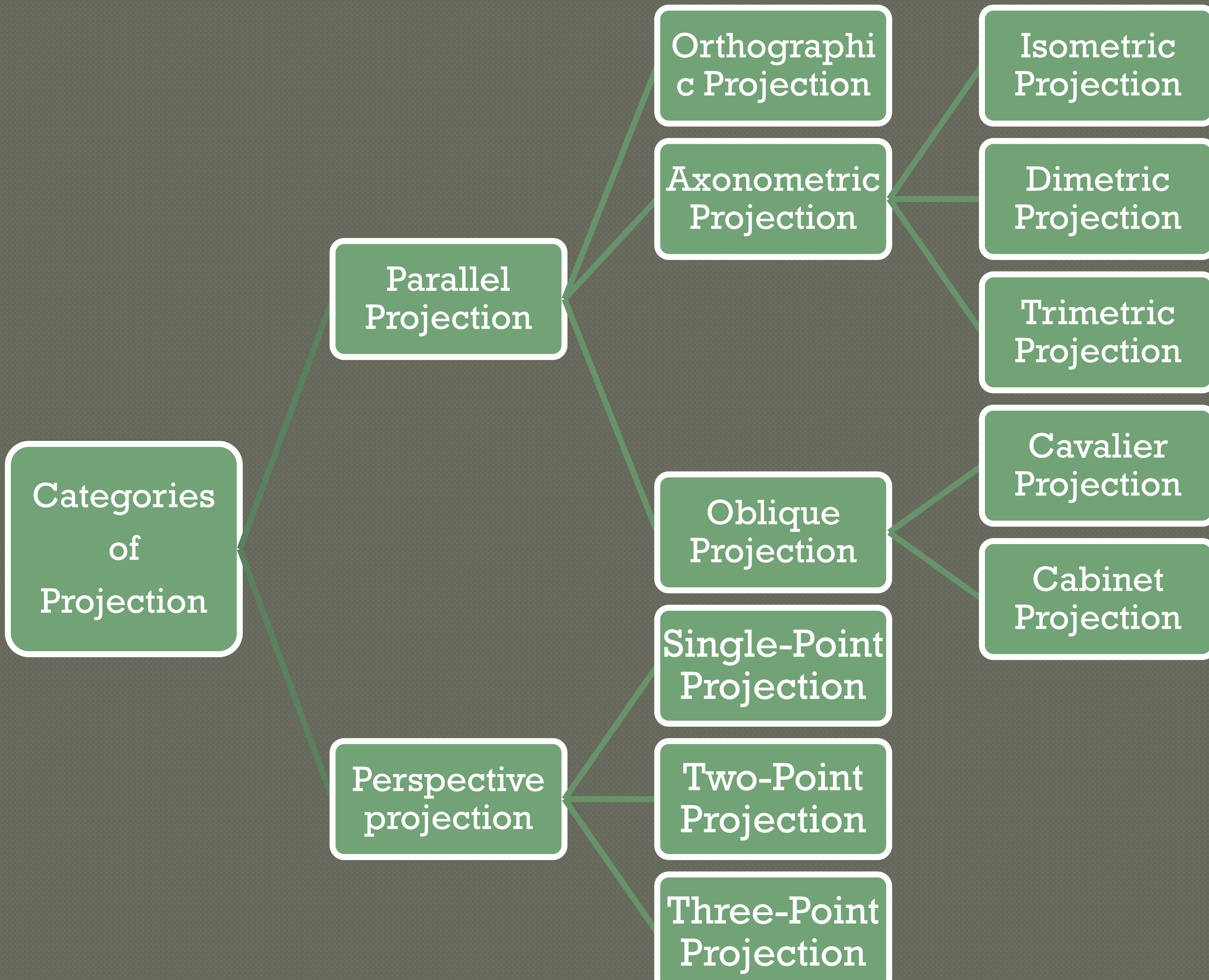


3D PROJECTIONS

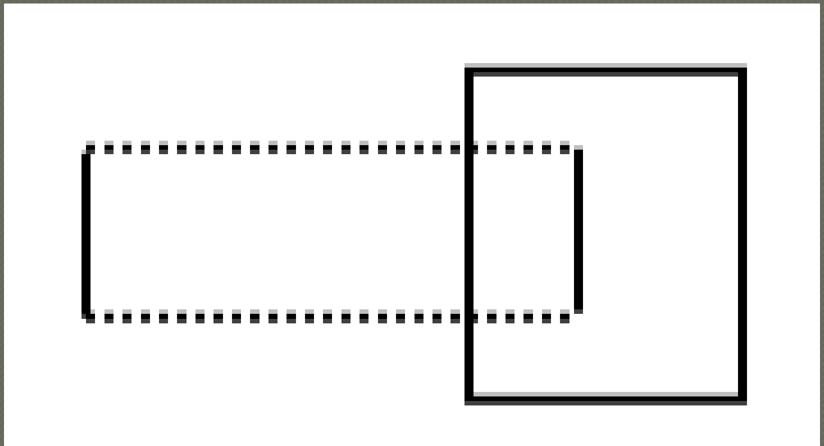
Projection

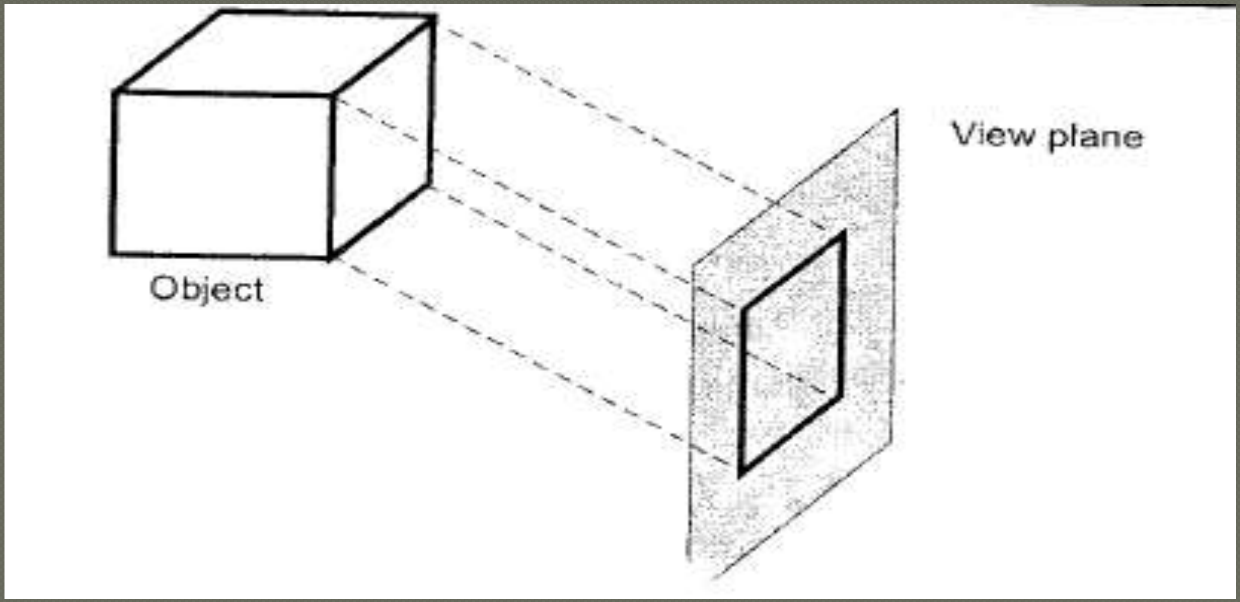
Representing a three-dimensional object or scene in 2-dimensional objects onto the 2-dimensional view plane.
There are 2 types of projections.

- Parallel Projection
- Perspective Projection


Categories of Projection
Parallel Projection
Perspective projection
Orthographic Projection
Axonometric Projection
Oblique Projection
Single-Point Projection
Two-Point Projection
Three-Point Projection
Isometric Projection
Dimetric Projection
Trimetric Projection
Cavalier Projection
Cabinet Projection

Parallel Projection

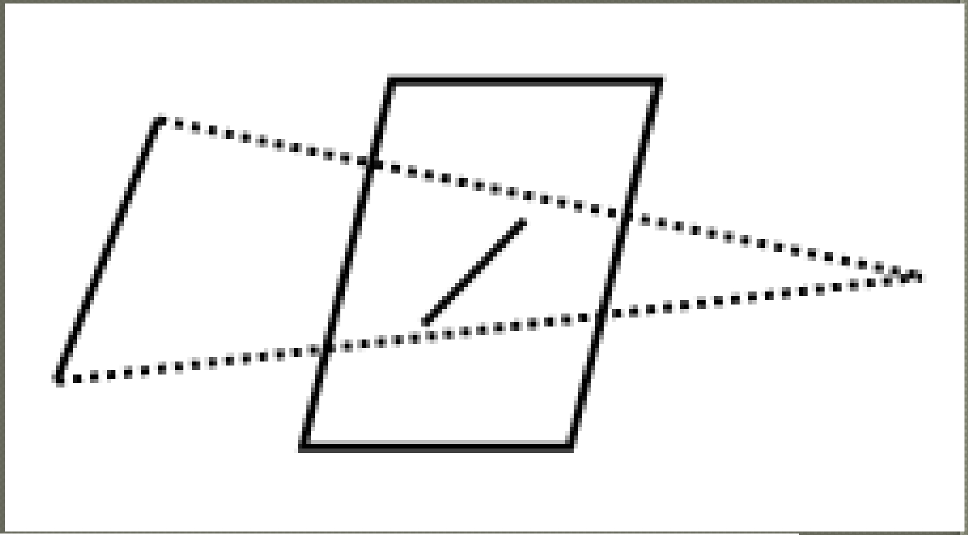


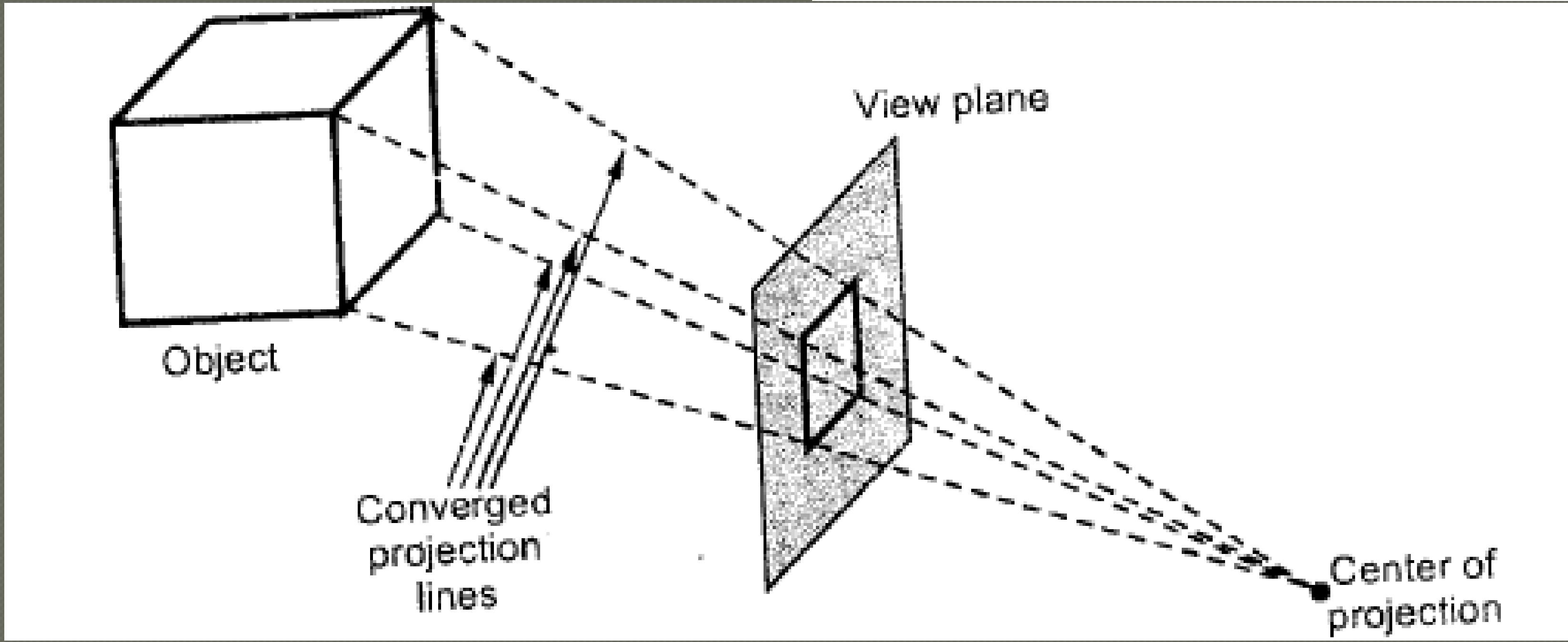


Parallel projection of an object to the view plane

Parallel Projection preserves relative proportions of objects but does not produce the realistic views

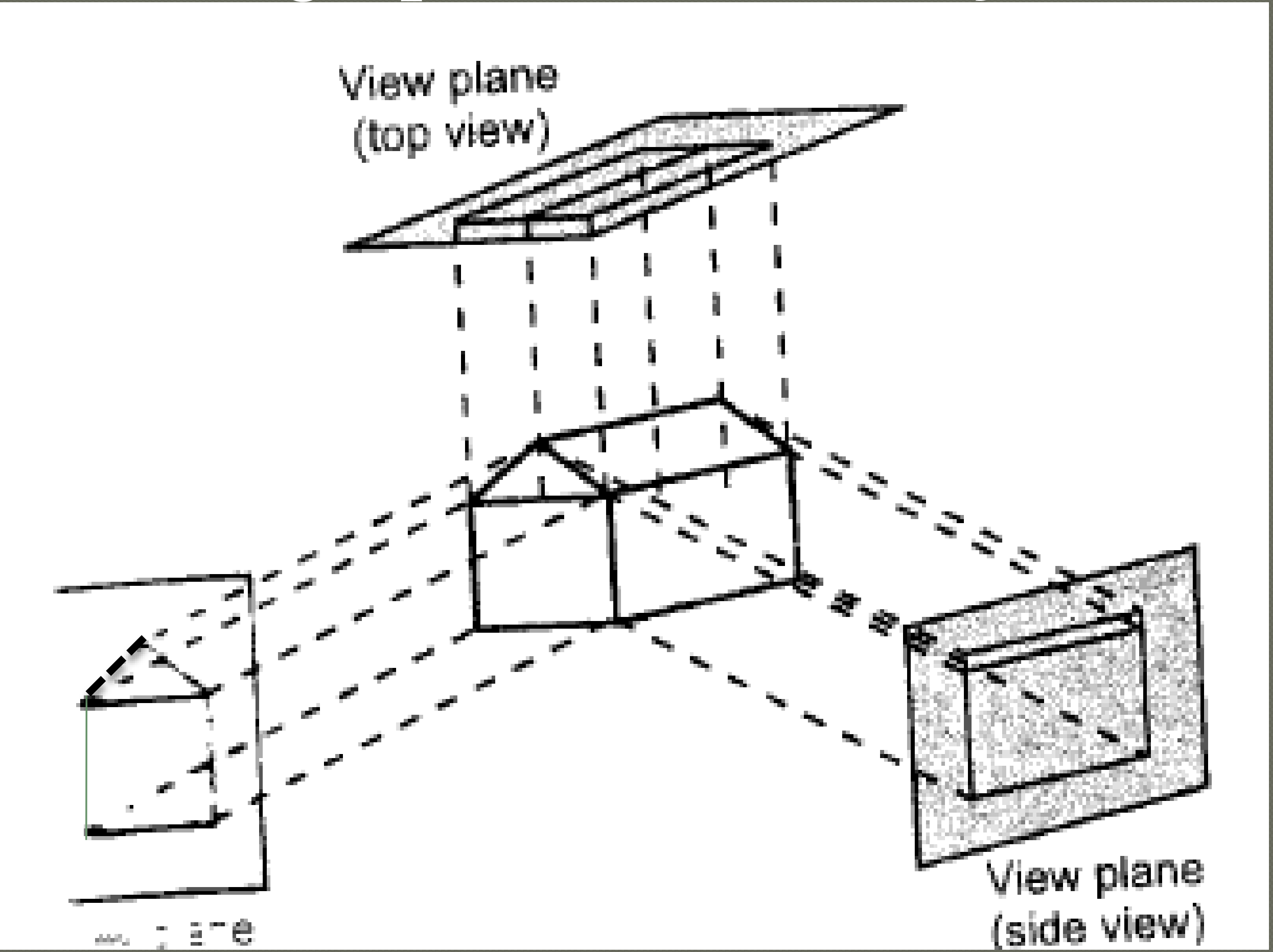
Perspective Projection





Perspective Projection produce the realistic views but does not preserves relative proportions of objects

Orthographic Parallel Projection

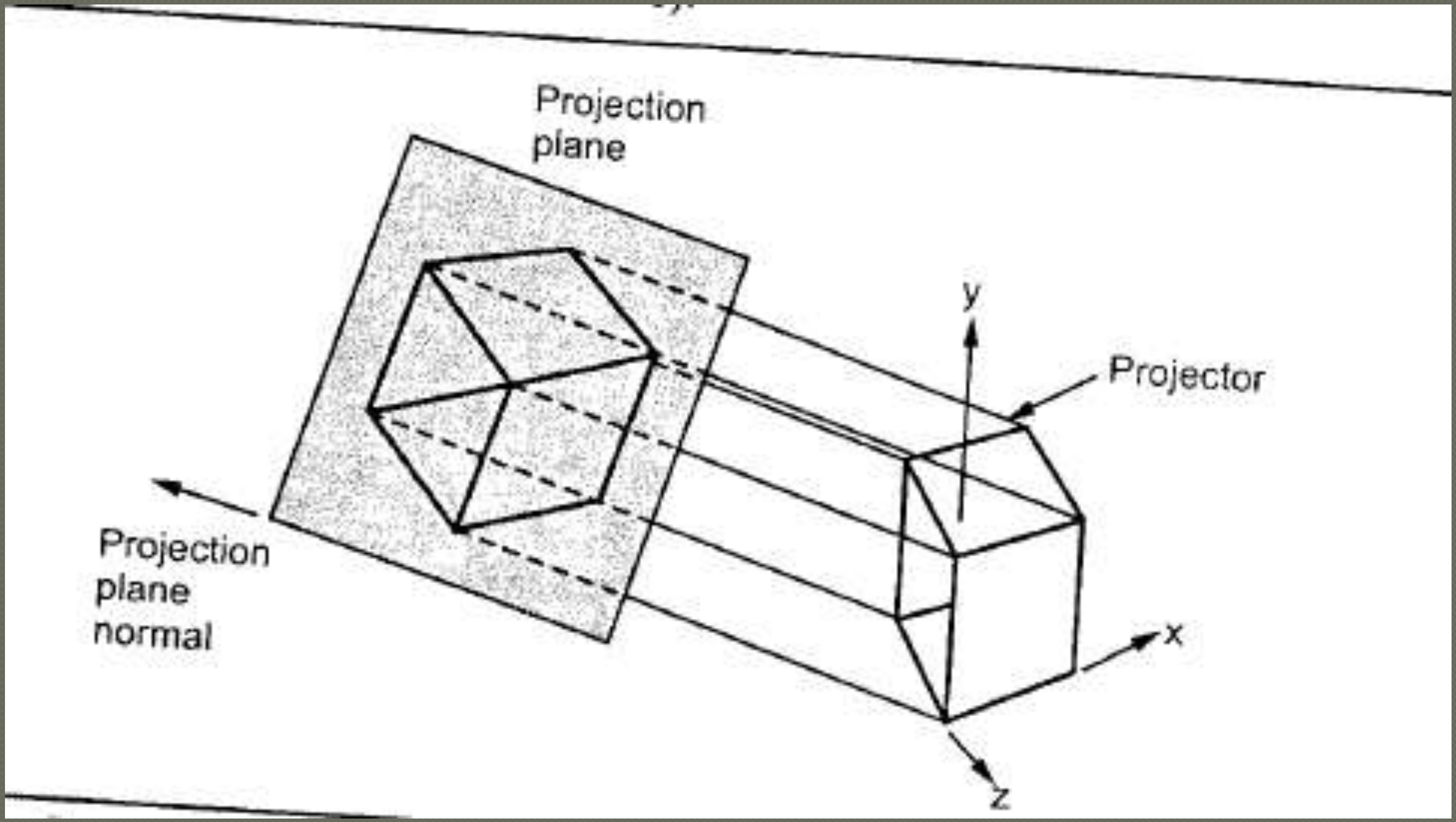


View Plane
Front View

Top,
Side,
rear
(far) :
Elevati
ons

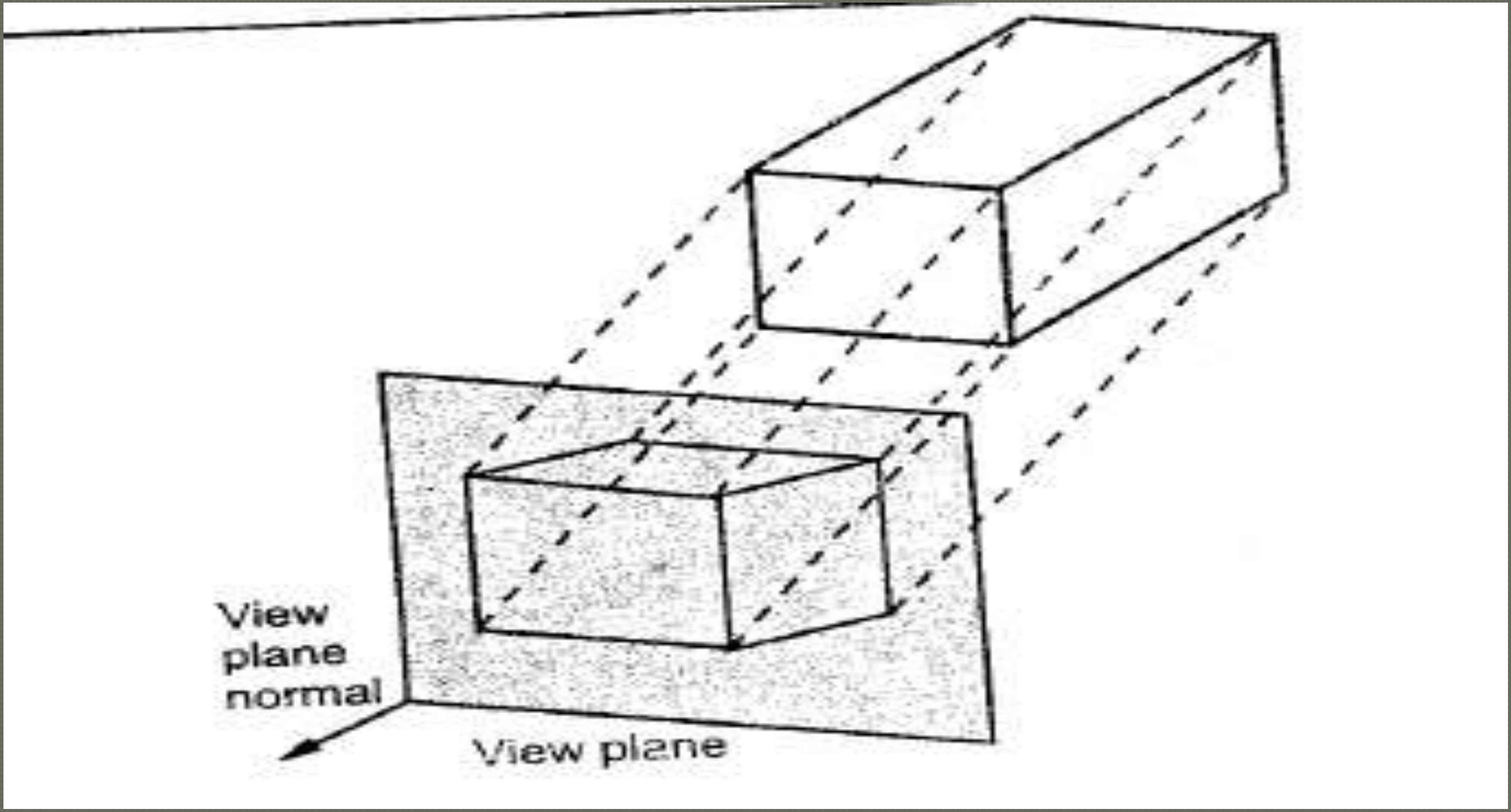
&
Top
is
called
Plan
View

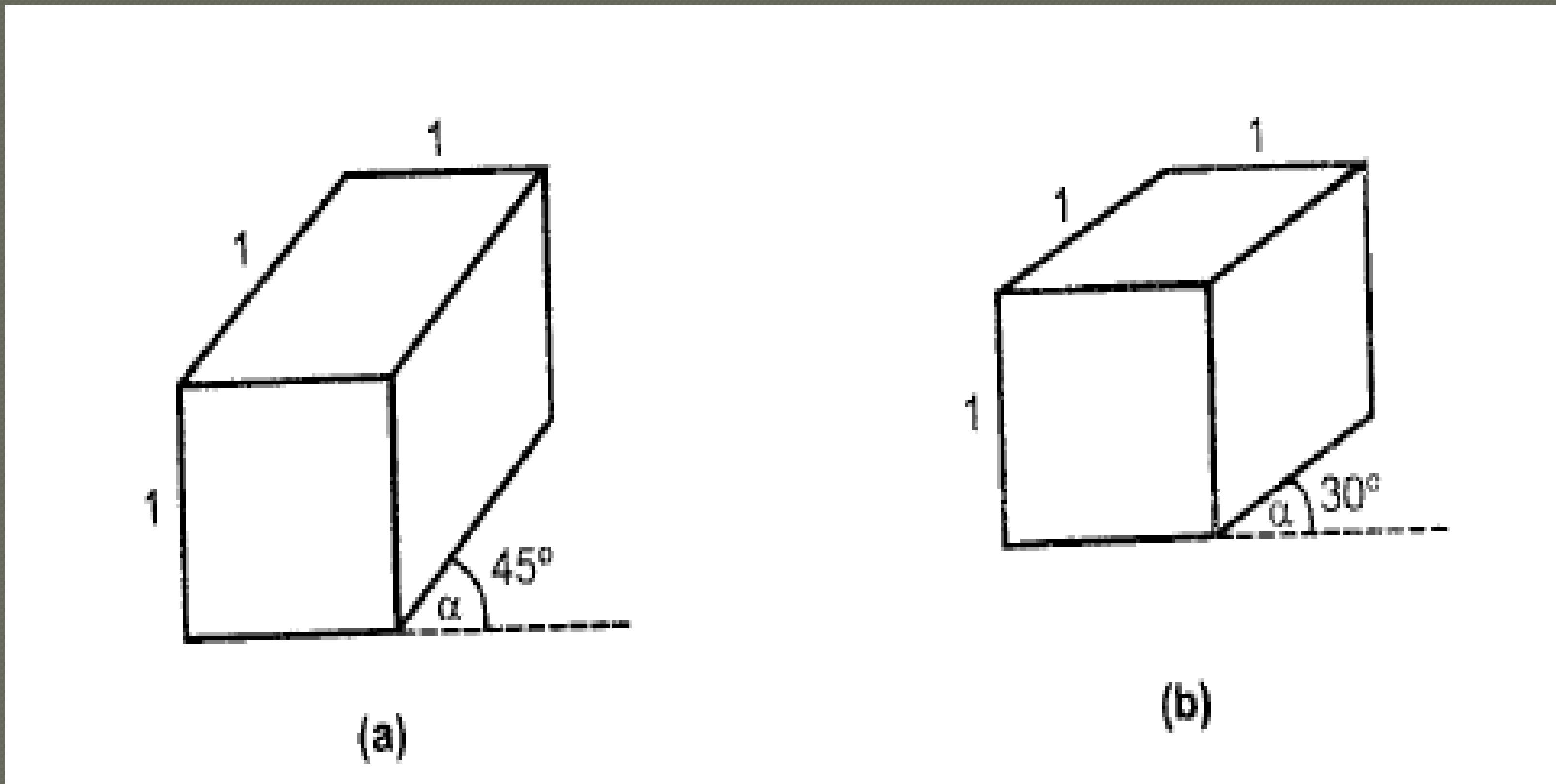
Axonometric Ortho…



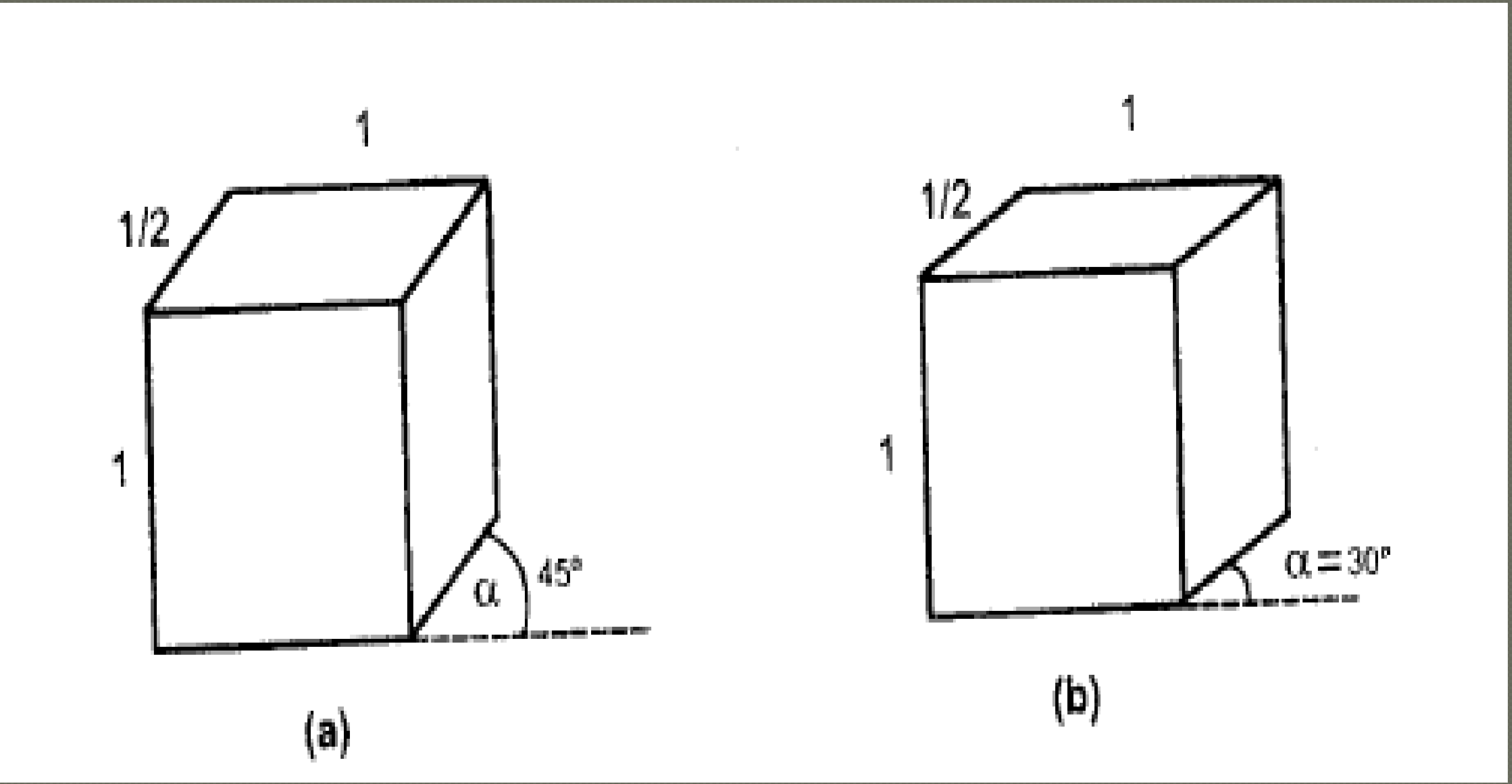
Isometric Projection of an object onto a viewing plane

Oblique Parallel Projection



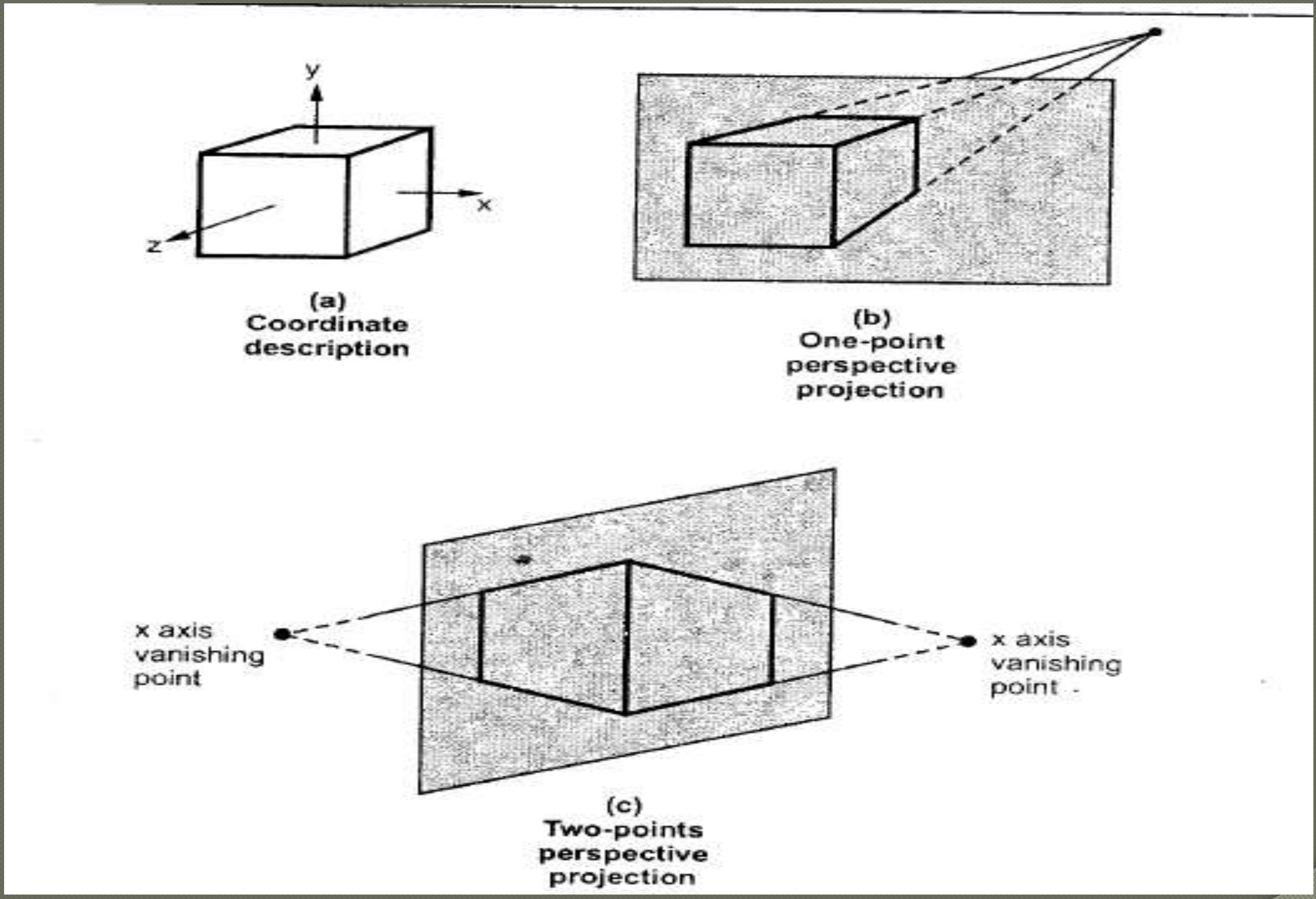


Cavalier Projections of the unit cube

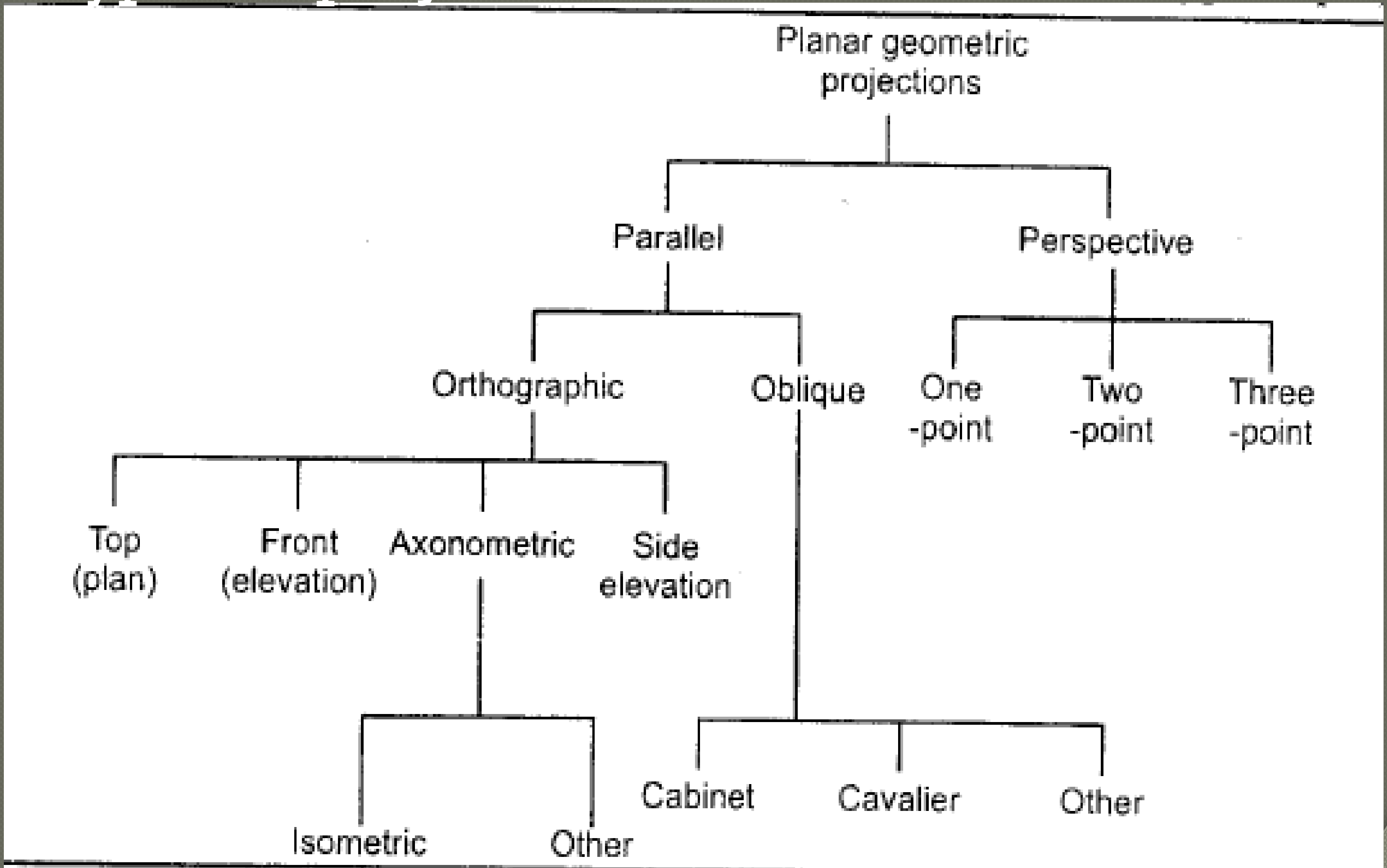


Cabinet Projections of the Unit Cube

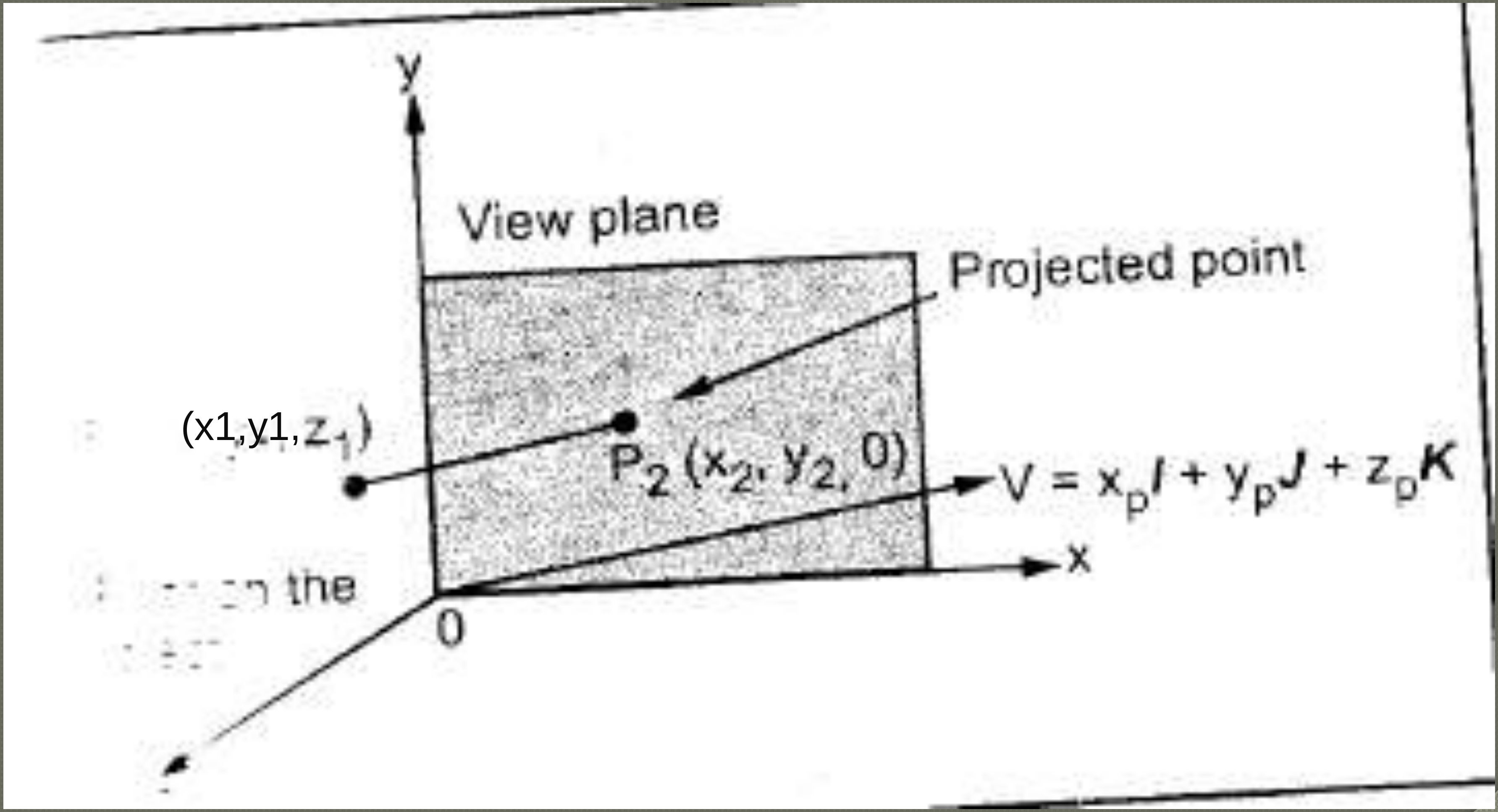
Types of Perspective Projections



Logical Relationship among the various types of projections



Transformation Matrix for general Parallel Projection (on XY plane)



$$x_2 = x_1 + x_p u$$

$$y_2 = y_1 + y_p u$$

$$z_2 = z_1 + z_p u$$

For projected point z_2 is 0, therefore, the third equation can be written as,

$$0 = z_1 + z_p u$$

$$\therefore \quad u = \frac{-z_1}{z_p}$$

Substituting the value of u in first two equations we get,

$$x_2 = x_1 + x_p(-z_1/z_p) \quad \text{and}$$

$$y_2 = y_1 + y_p(-z_1/z_p)$$

... form as given below :

Matrix Representation

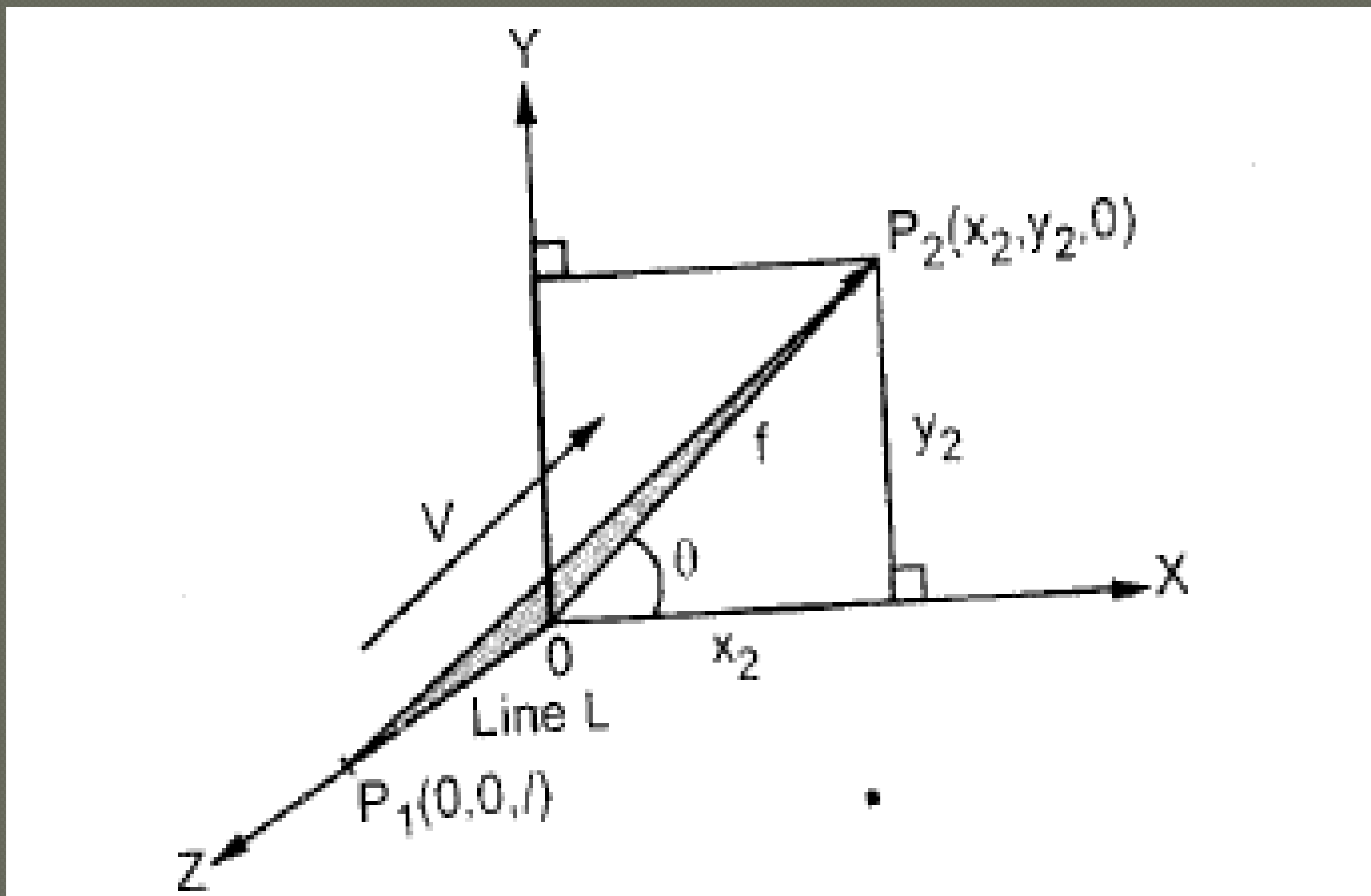
$$[x_2\ y_2] = [x_1\ y_1\ z_1]\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -x_p/z_p & -y_p/z_p \end{bmatrix}$$

or in homogeneous coordinates we have,

$$[x_2\ y_2\ z_2\ 1] = [x_1\ y_1\ z_1\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -x_p/z_p & -y_p/z_p & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

i.e. $\quad P_2 = P_1 \cdot Par_v$

Transformation Matrix for general Oblique Projection (on XY plane)



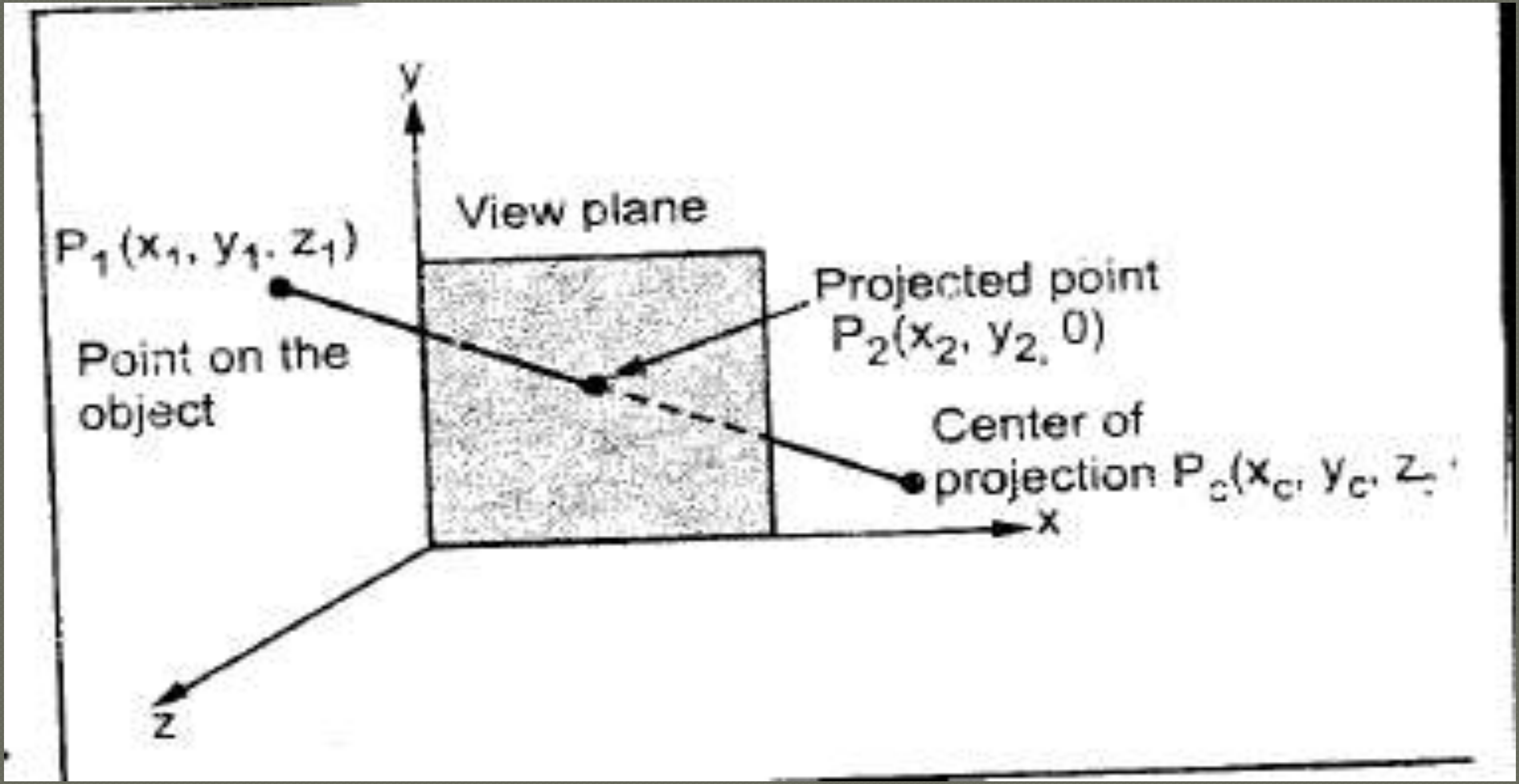
$$V = \overline{P_1 P_2} = x_2 I + y_2 J - l K$$

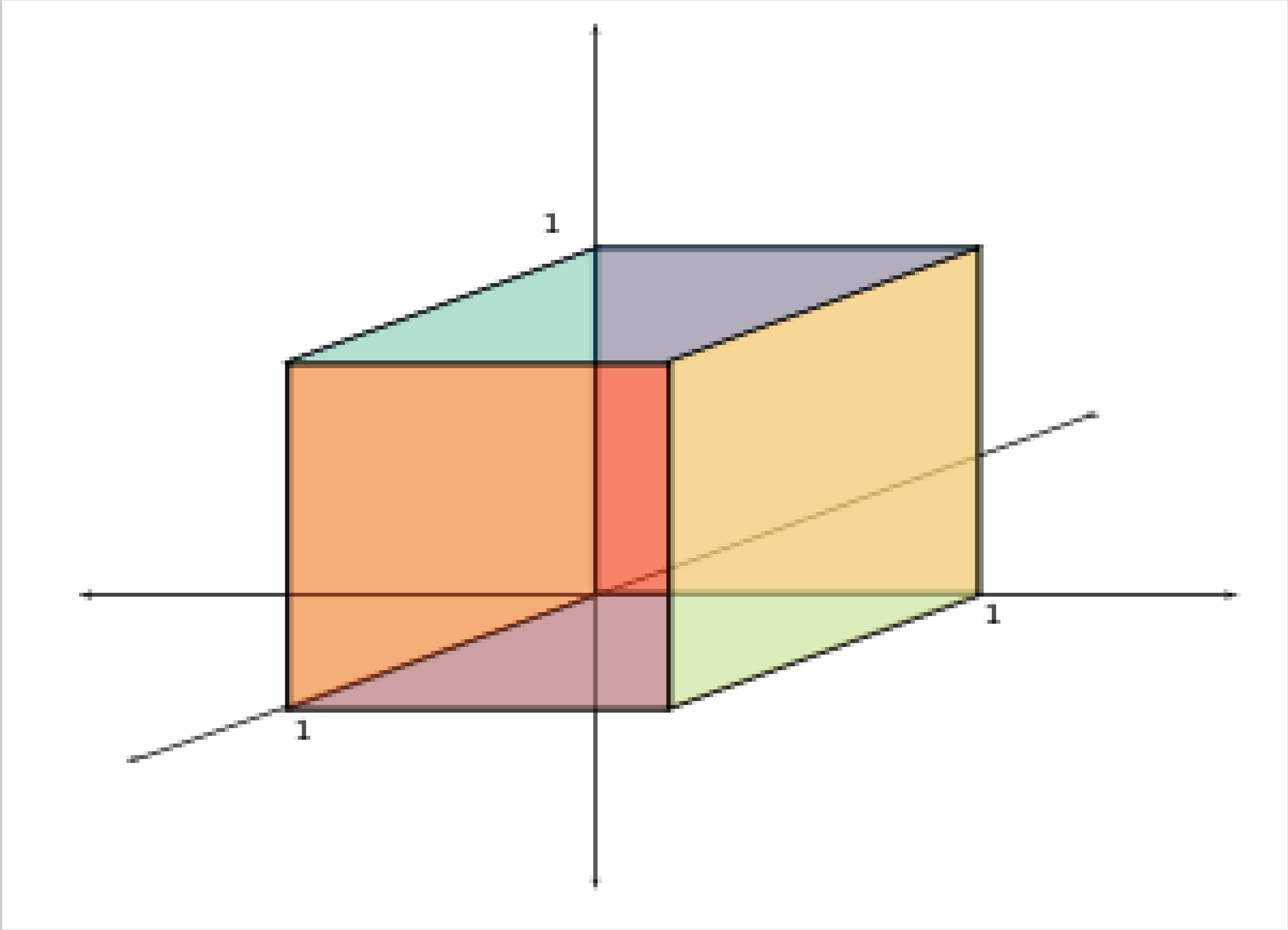
$$V = x_p I + y_p J + z_p K$$

$$x_p = x_2 = f \cos\theta$$
$$y_p = y_2 = f \sin\theta$$
$$z_p = -l$$

$$\text{Par}_v = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \dfrac{f\cos\theta}{l} & \dfrac{f\sin\theta}{l} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

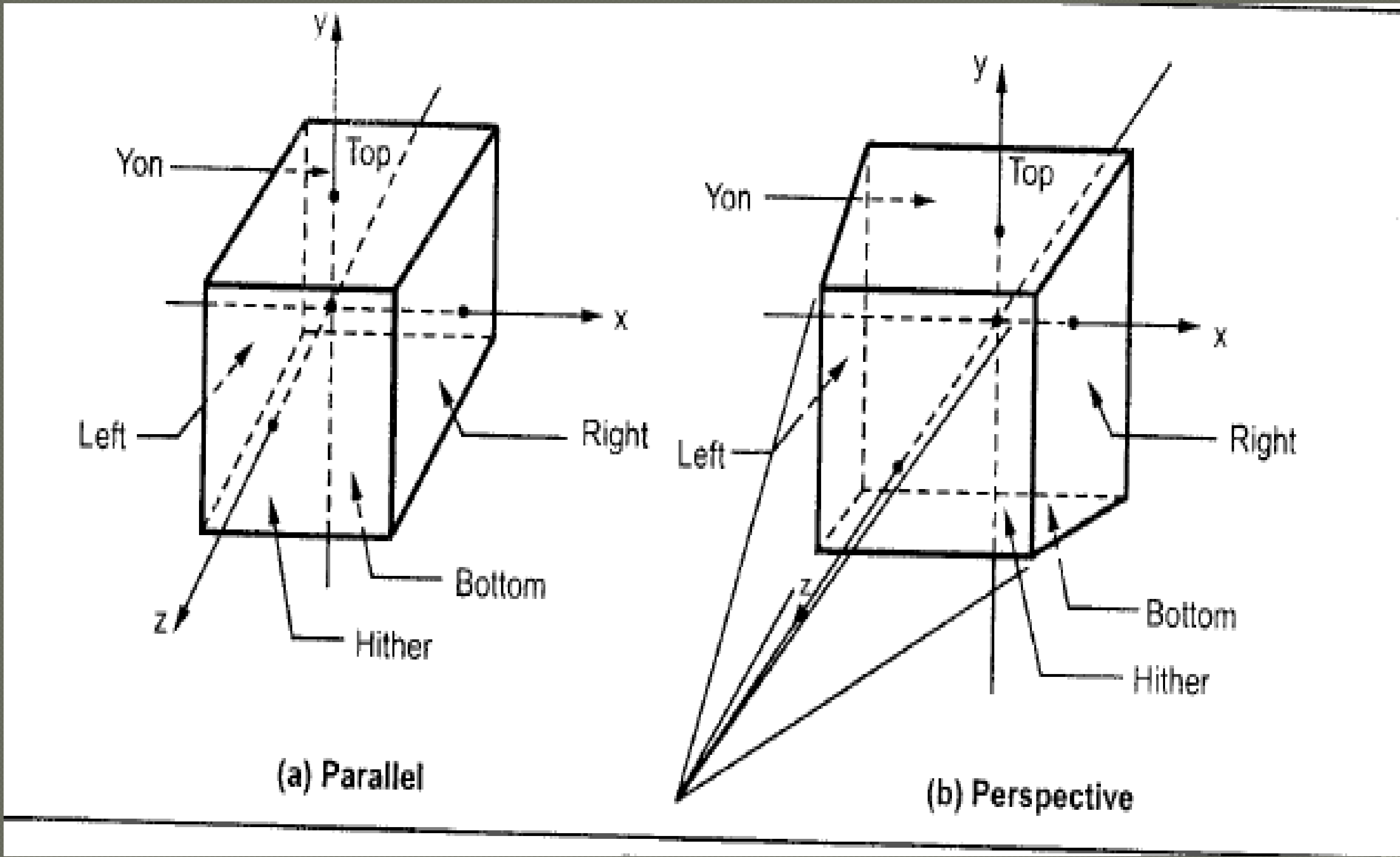
Transformation Matrix for Perspective Projection (on XY plane)





Unit cube

3D Clipping



y
Yon
Top
x
Left
Right
z
Bottom
Hither
(a) Parallel
y
Yon
Top
x
Left
Right
z
Bottom
Hither
(b) Perspective

The two-dimensional concept of region codes can be extended to three dimensions by considering six sides and 6-bit code instead of four sides and 4-bit code. Like two-dimension, we assign the bit positions in the region code from right to left as

Bit 1 = 1, if the end point is to the left of the volume

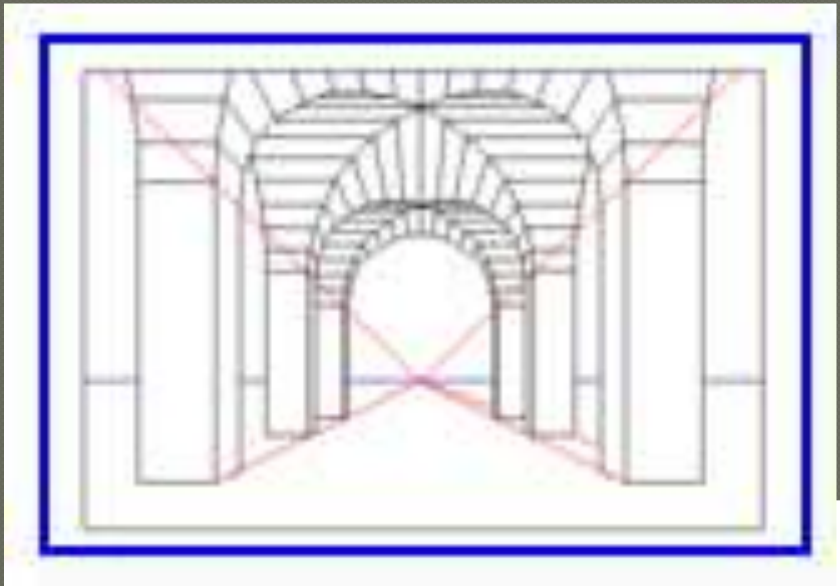
Bit 2 = 1, if the end point is to the right of the volume

Bit 3 = 1, if the end point is the below the volume

Bit 4 = 1, if the end point is above the volume

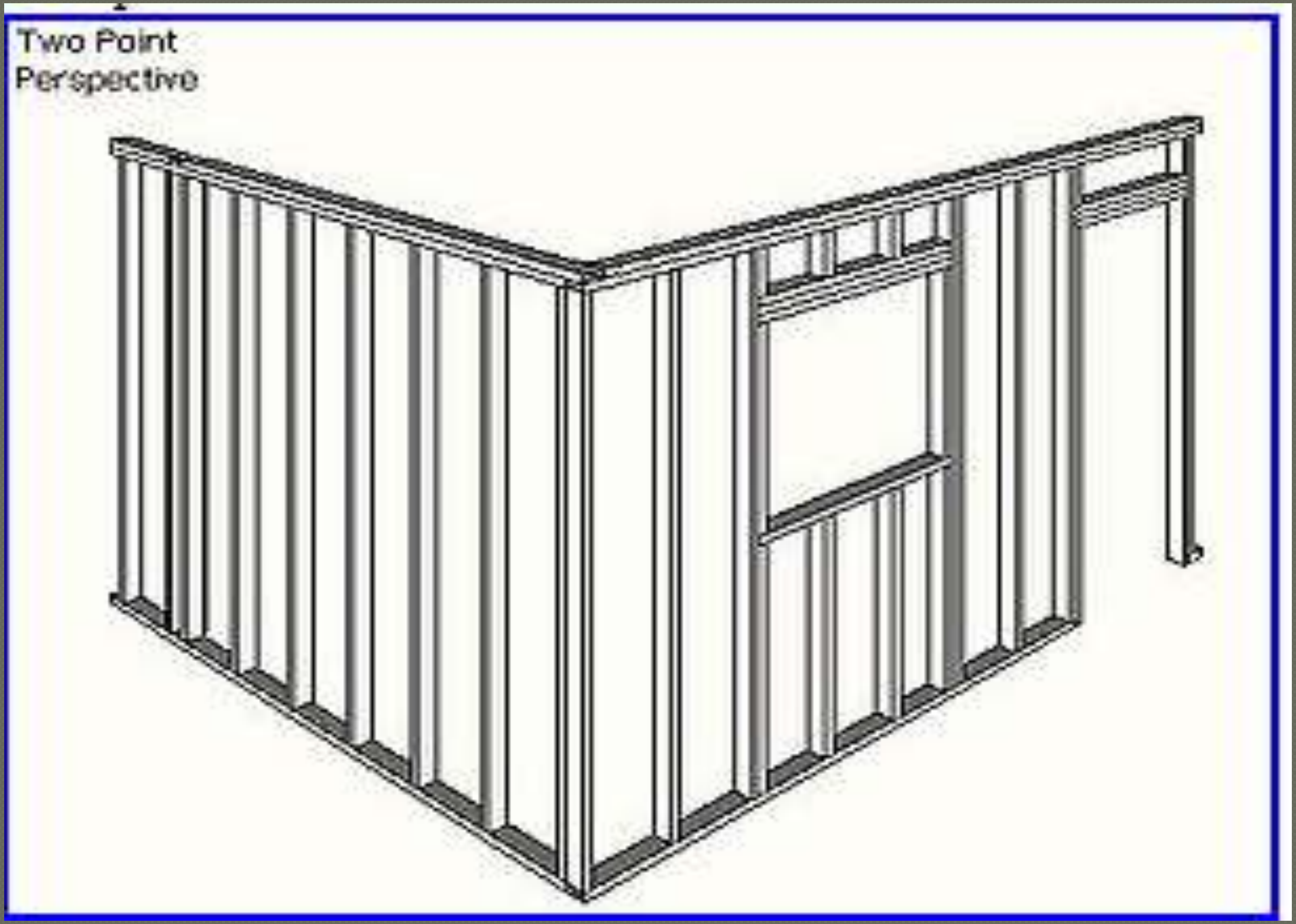
Bit 5 = 1, if the end point is in front of the volume

Bit 6 = 1, if the end point is behind the volume

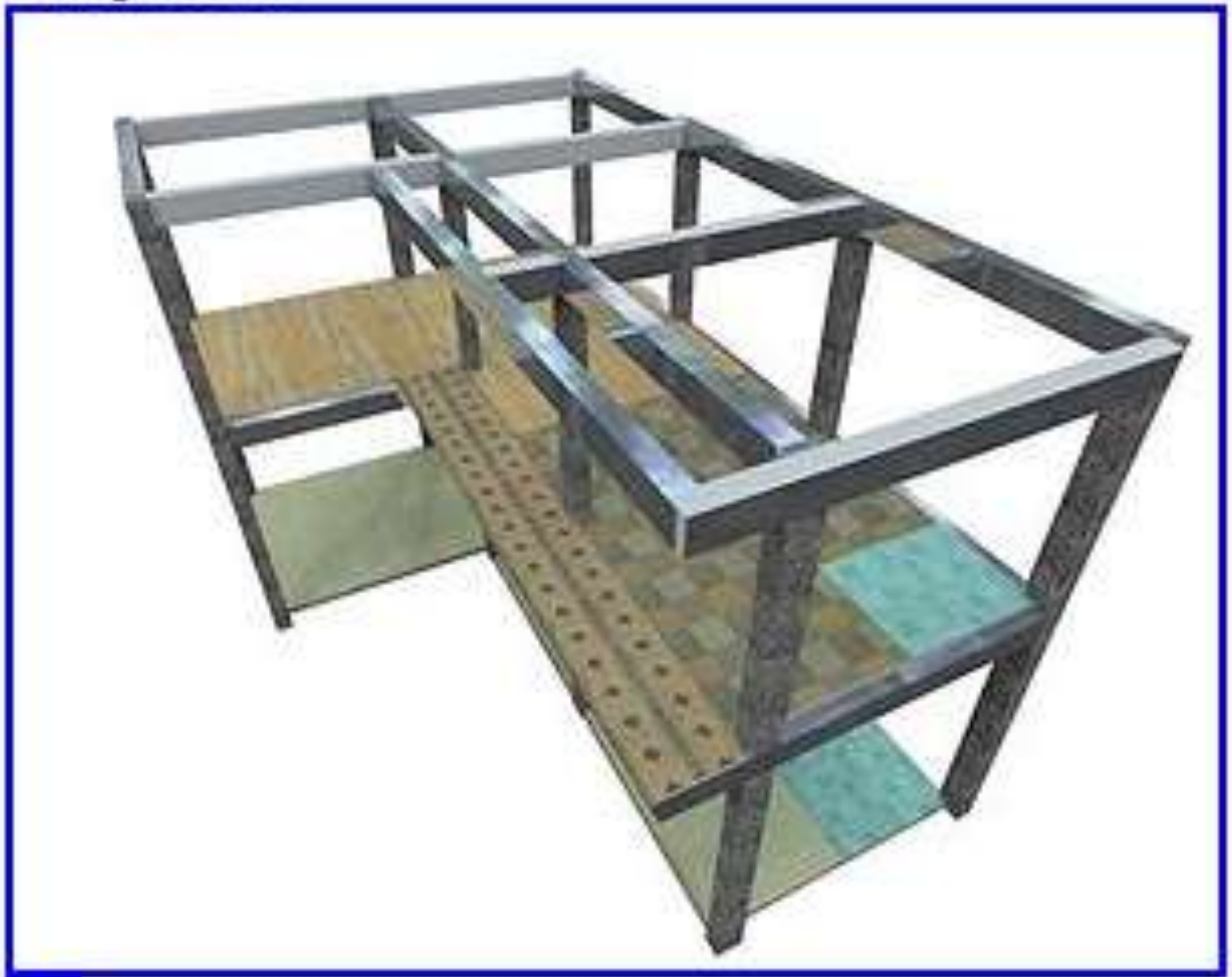


One-point perspective projection.



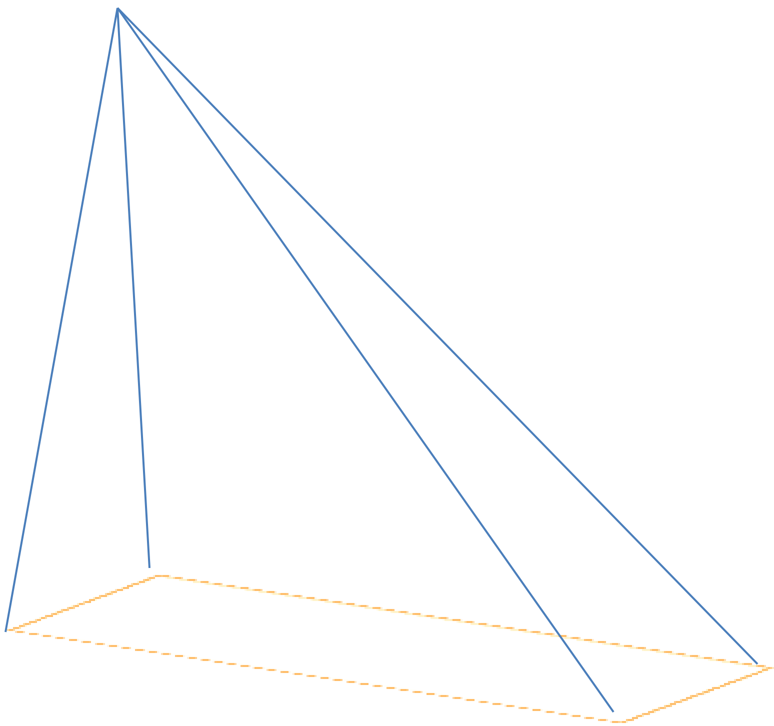

Two Point
Perspective

Three-Point Perspective

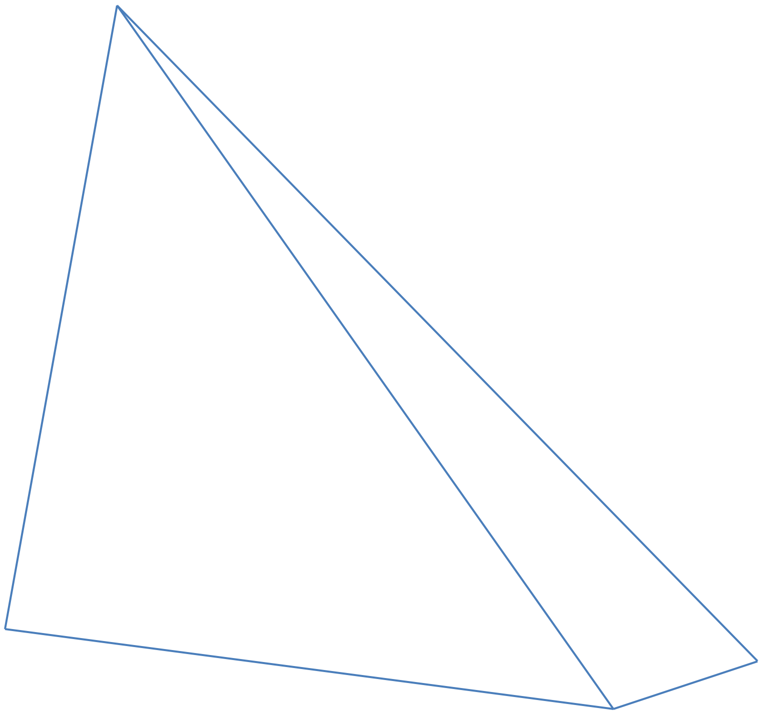


HiDdEn SuRfAcE rEmOvAl

Example



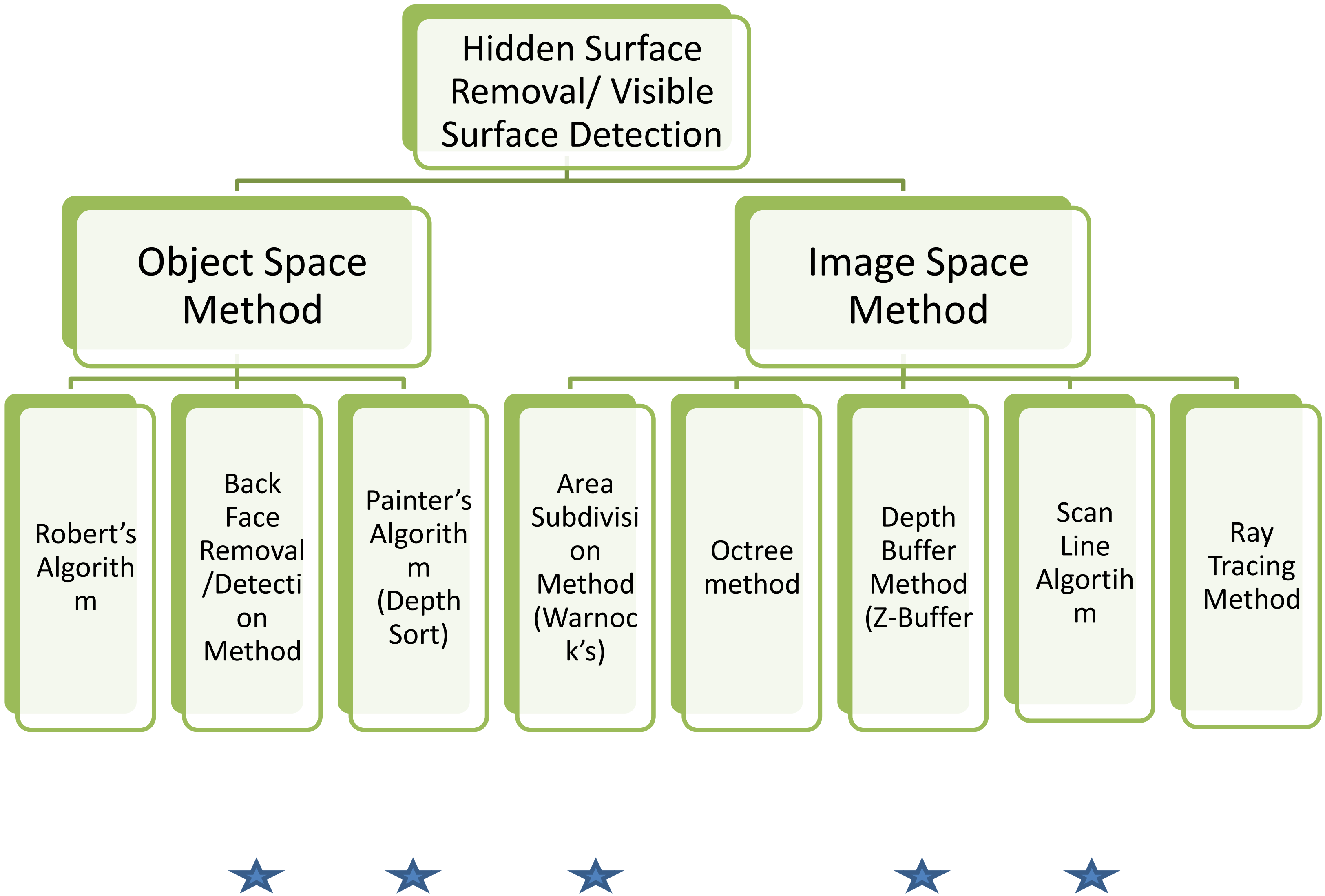
Original Pyramid



After Hidden Surface Removal

cAtEgOrIeS oF hIDdEn SuRfAcE rEmOvAl

- oBJECT sPACE mETHOD
- iMAGE sPACE mETHOD


Hidden Surface Removal/ Visible Surface Detection
Object Space Method
Image Space Method
Robert's Algorith m
Back Face Removal /Detecti on Method
Painter's Algorith m (Depth Sort)
Area Subdivisi on Method (Warnoc k's)
Octree method
Depth Buffer Method (Z-Buffer
Scan Line Algortih m
Ray Tracing Method

oThErS

- fLOATING hORIZON aLGORITHM
- bINARY sPACE pARTITIONING

bAcK fAcE rEmOvAl mEtHoD

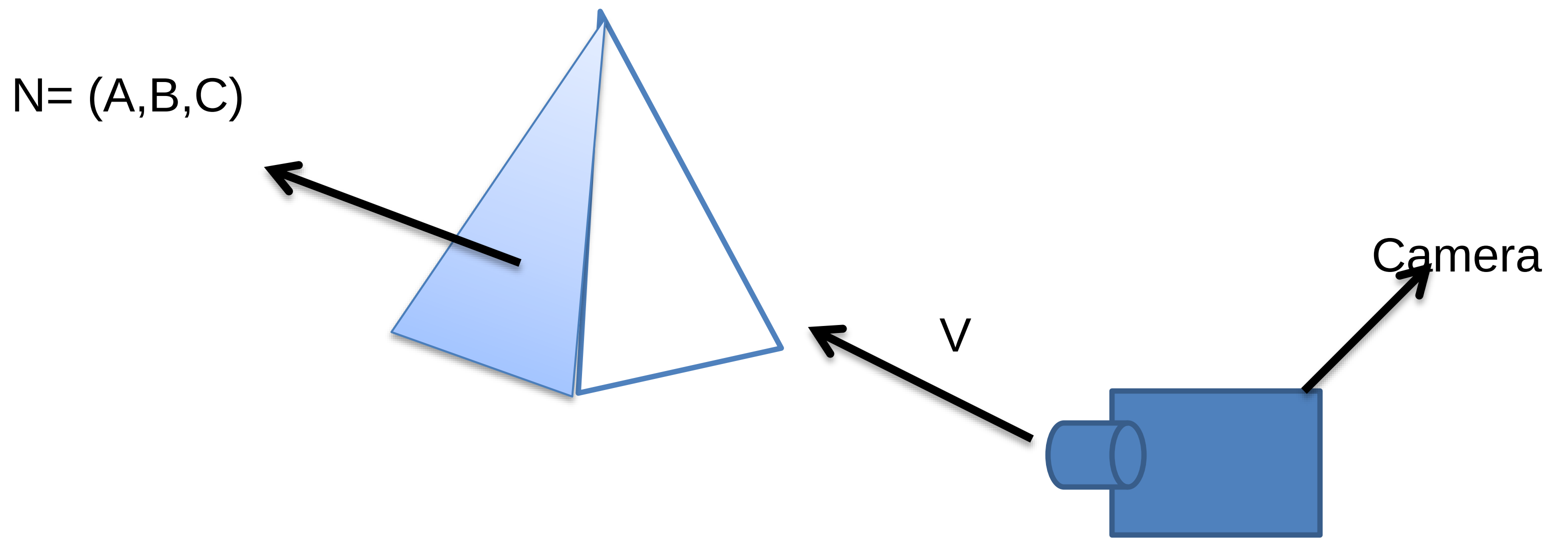
- Back face means the surface of the polygon which in not visible in projection. So we have to remove this surface from projection.
- It is used for identifying back faces of a polyhedron is base on the inside-outside test.
- Back face removal algorithm will be applied on plane polygons.

 A point (x,y,z) is inside a polygon surface with plane parameters A, B, C and D if

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

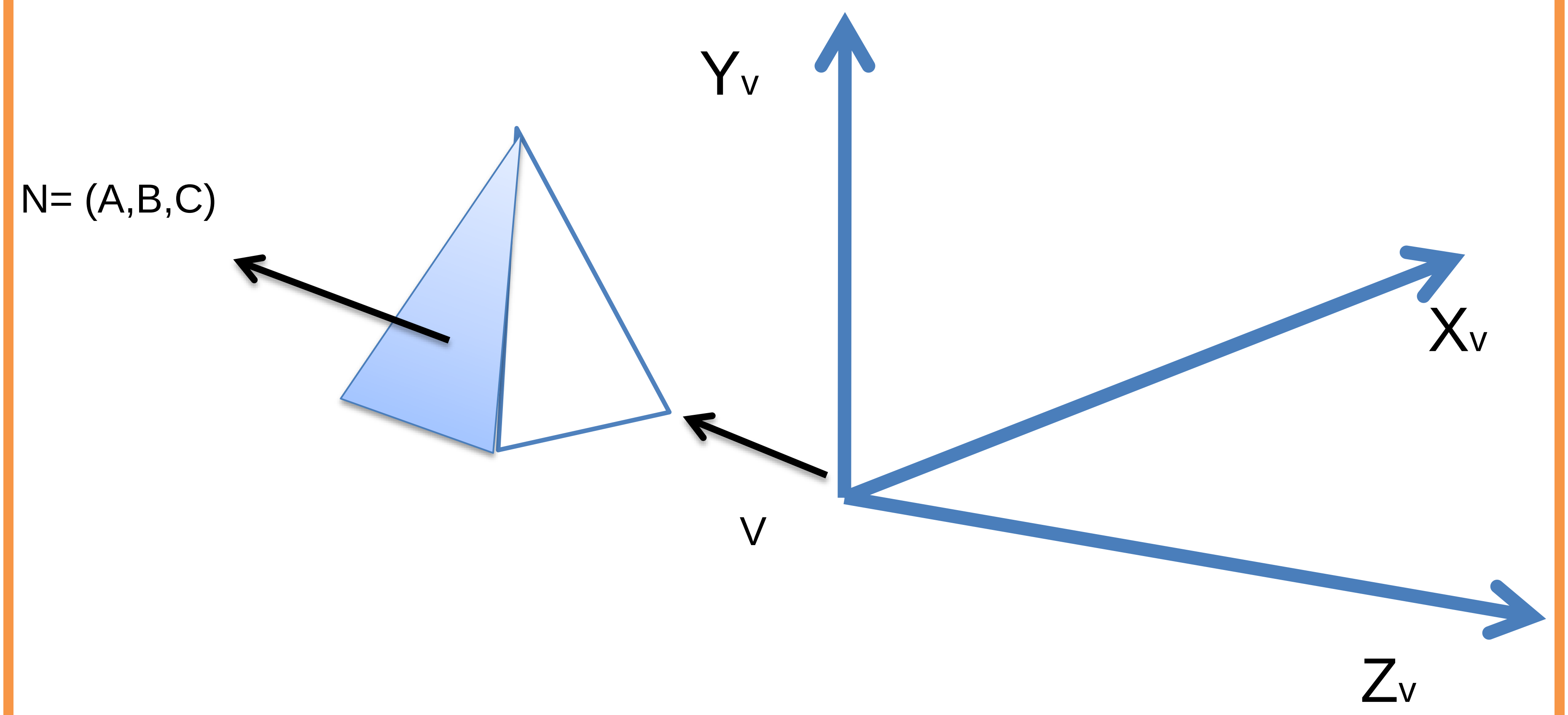
- The normal vector N to a polygon surface, which has Cartesian Components (A,B,C).
- If V is a vector in the viewing direction from the eye or camera position, then this position is a back face if

$$V.N > 0$$



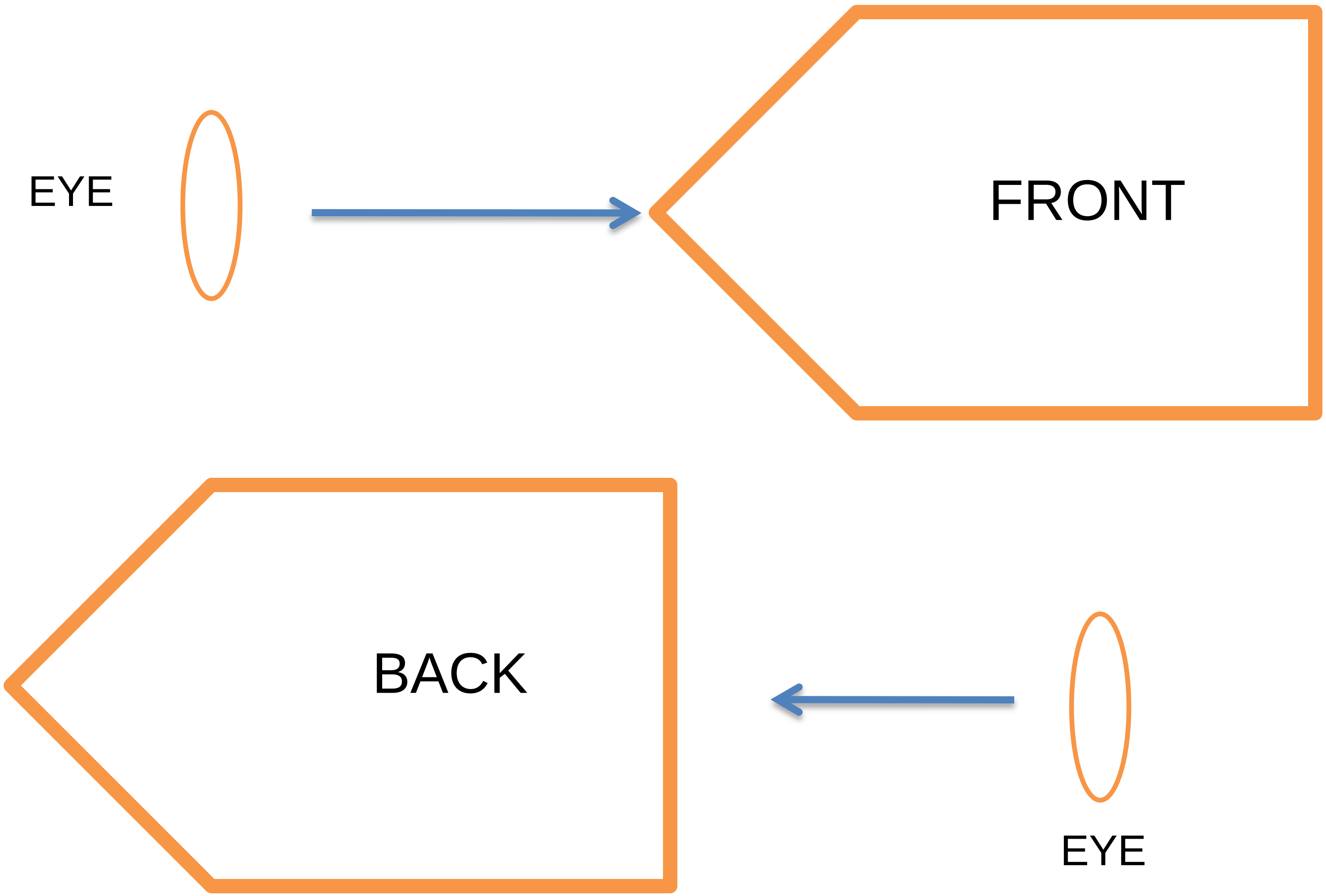
- Figure : Vector V in the viewing direction and a back-face normal vector N of a polyhedron

- If the dot product is positive, we can say that the polygon faces towards the viewer, otherwise it faces away and should be removed.
- In case, if object description has been converted to projection coordinates and our viewing direction is parallel to the viewing Z_v axis, then $V = (0,0,Z_v)$ and $V.N = Z_v C$
- To consider the sign of C , the z component of the normal vector N. Now if the Z component is positive, then the polygon faces towards the viewer, if negative it faces away .

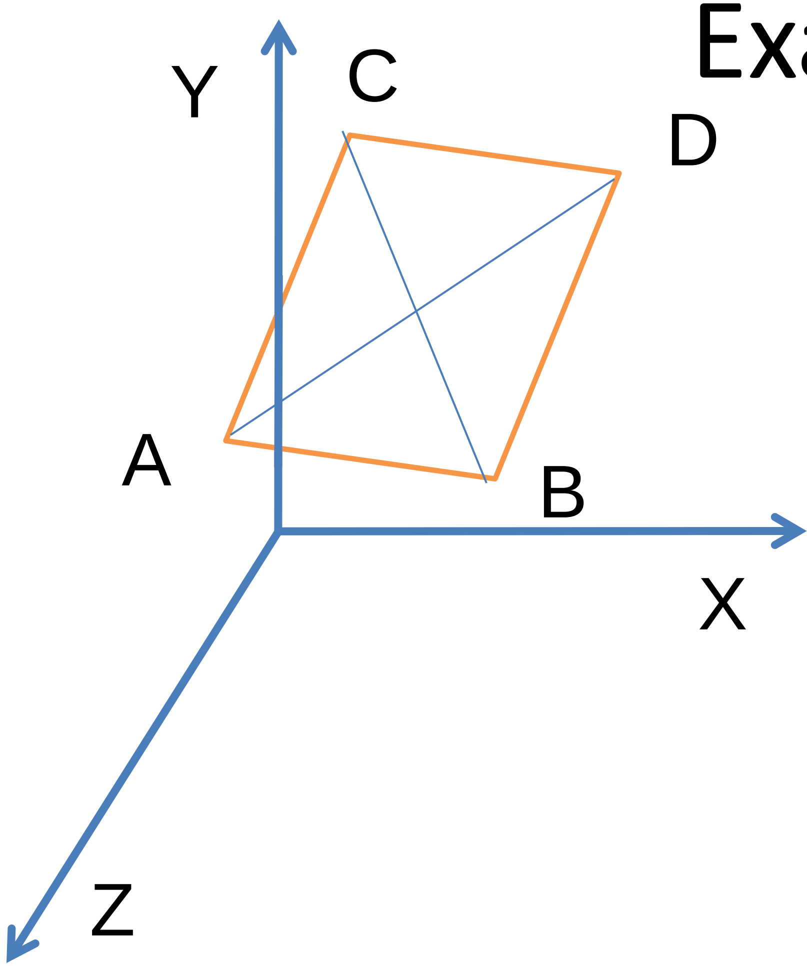


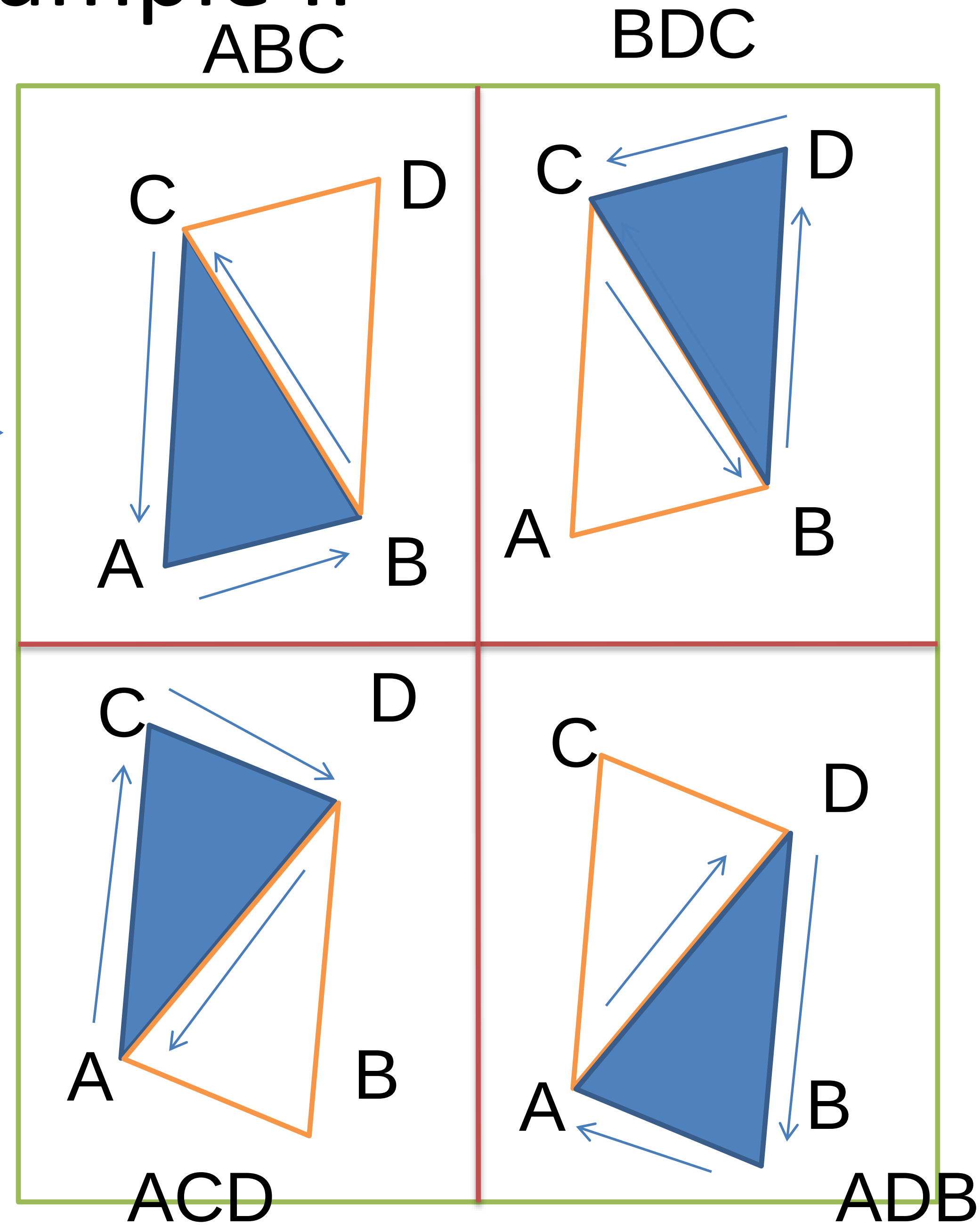
Y_v
N= (A,B,C)
X_v
V
Z_v

Example I



Example II



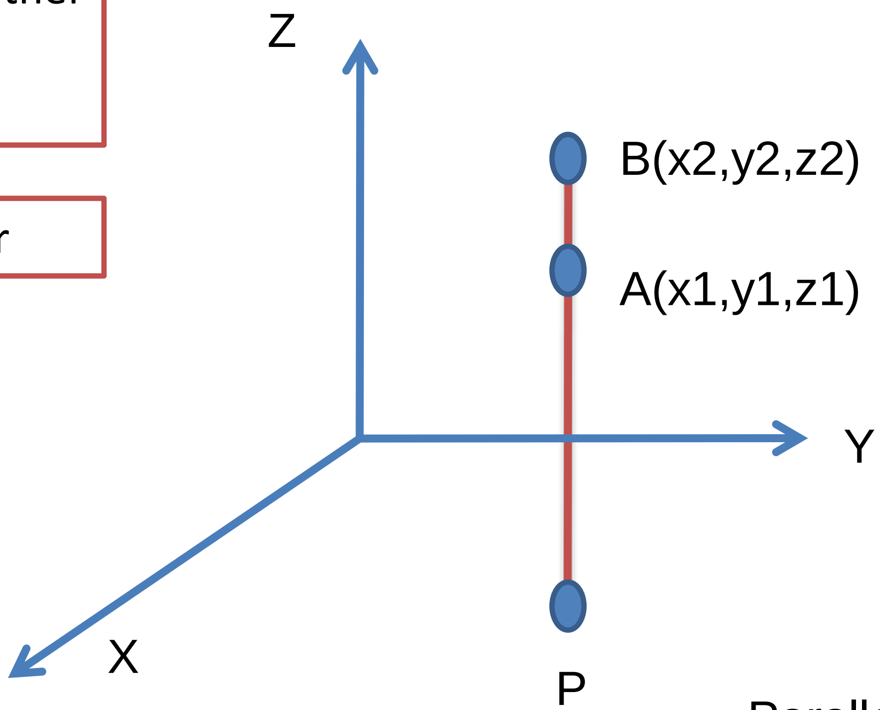


Back-face Detection Method Direction for Faces

DePtH cOmPaRiSiOn

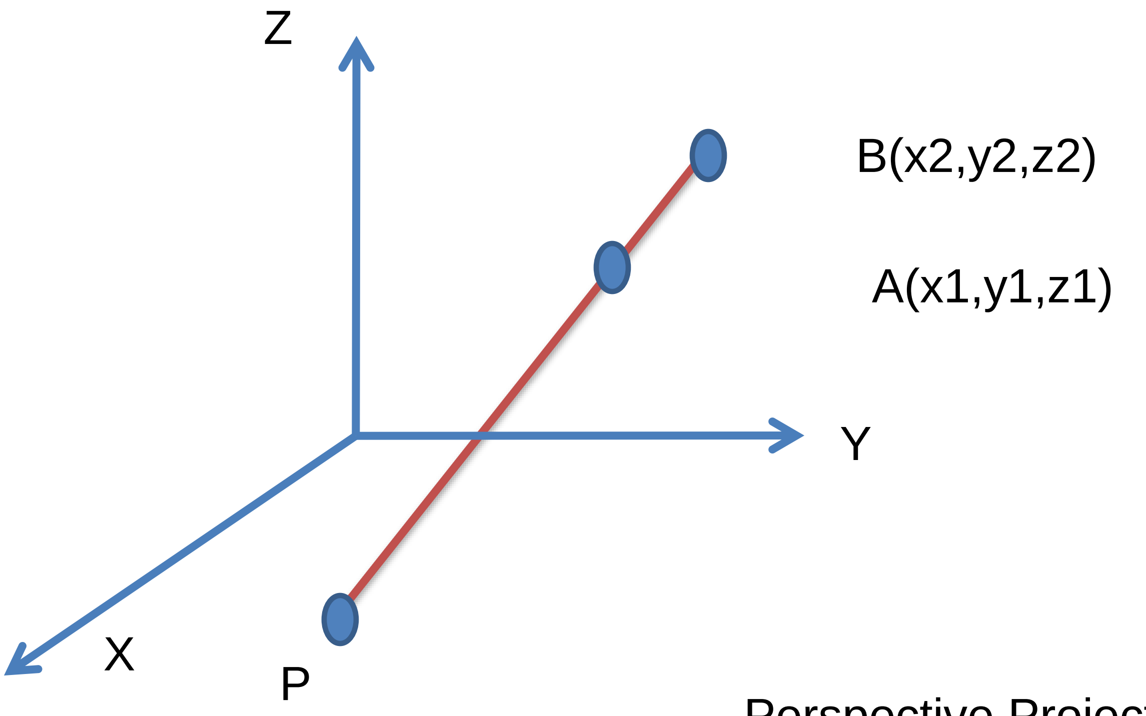
If A and B are not on the same projection line then no point hide the other point.

Frame Buffer



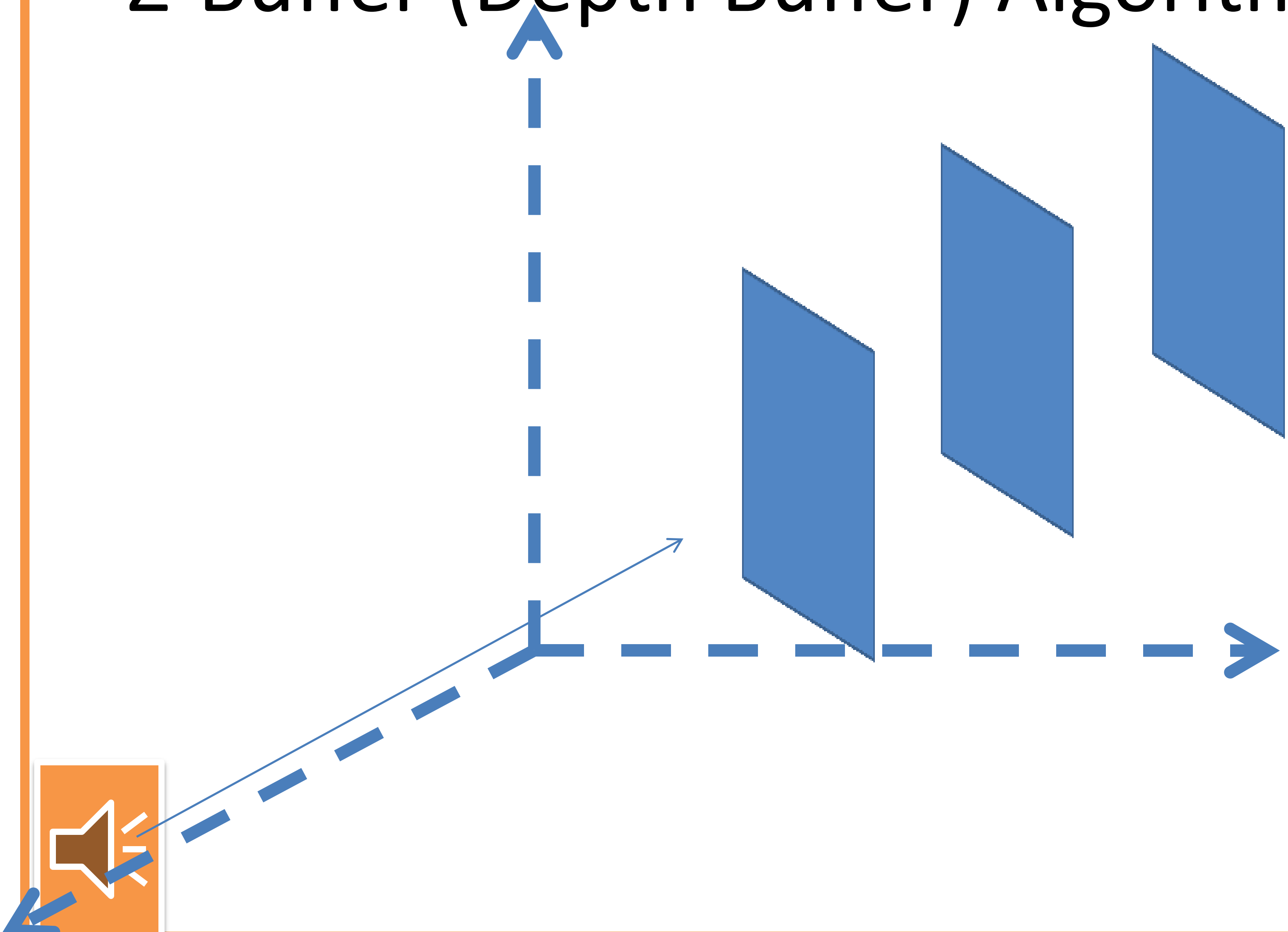
Parallel Projection

If A and B are on the same projection line then in case of parallel projection on xy plane if x1=x2 and y1 = y2 then A and B are on same plane. If Z1<Z2 then A point hide B.

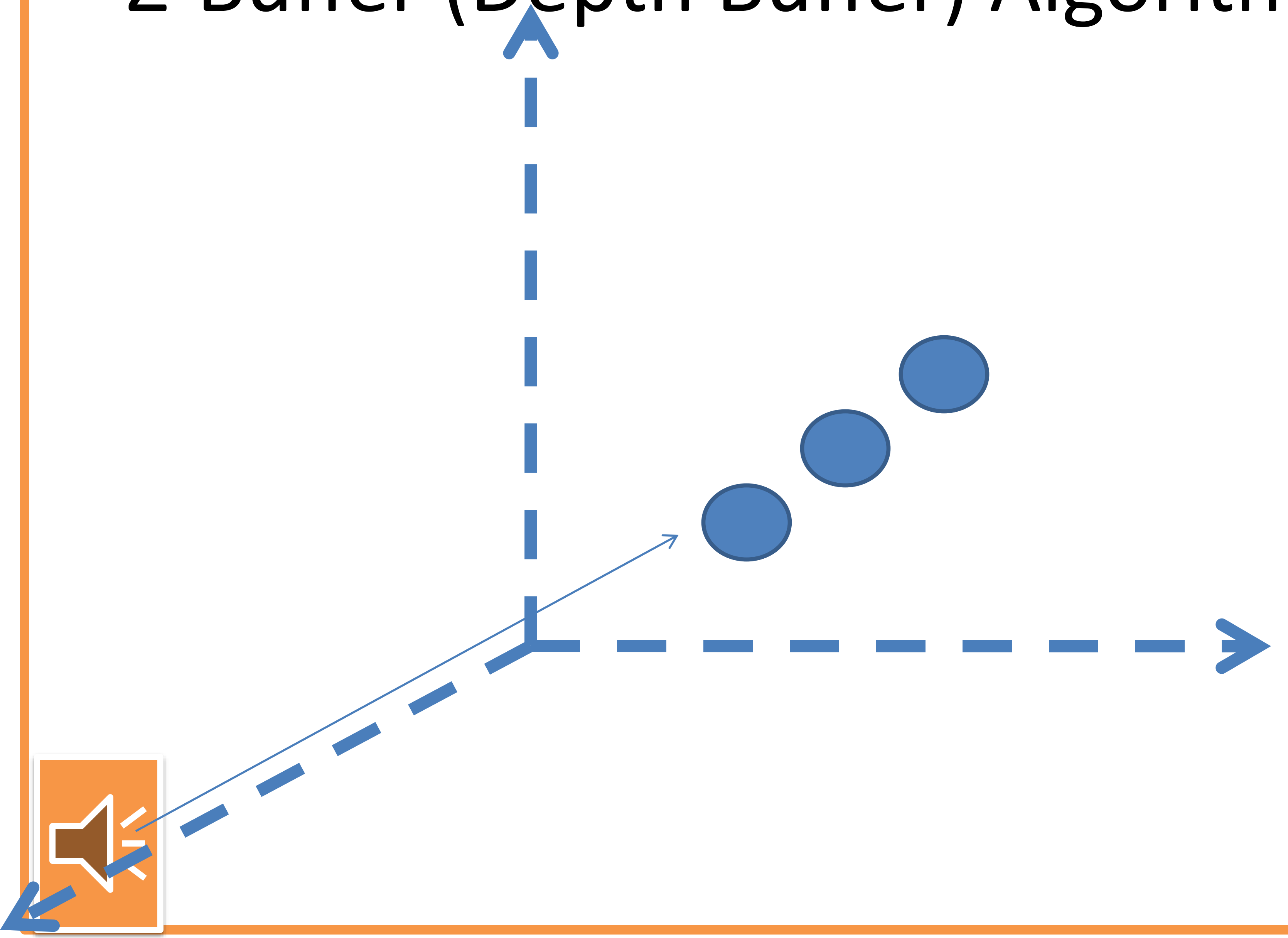


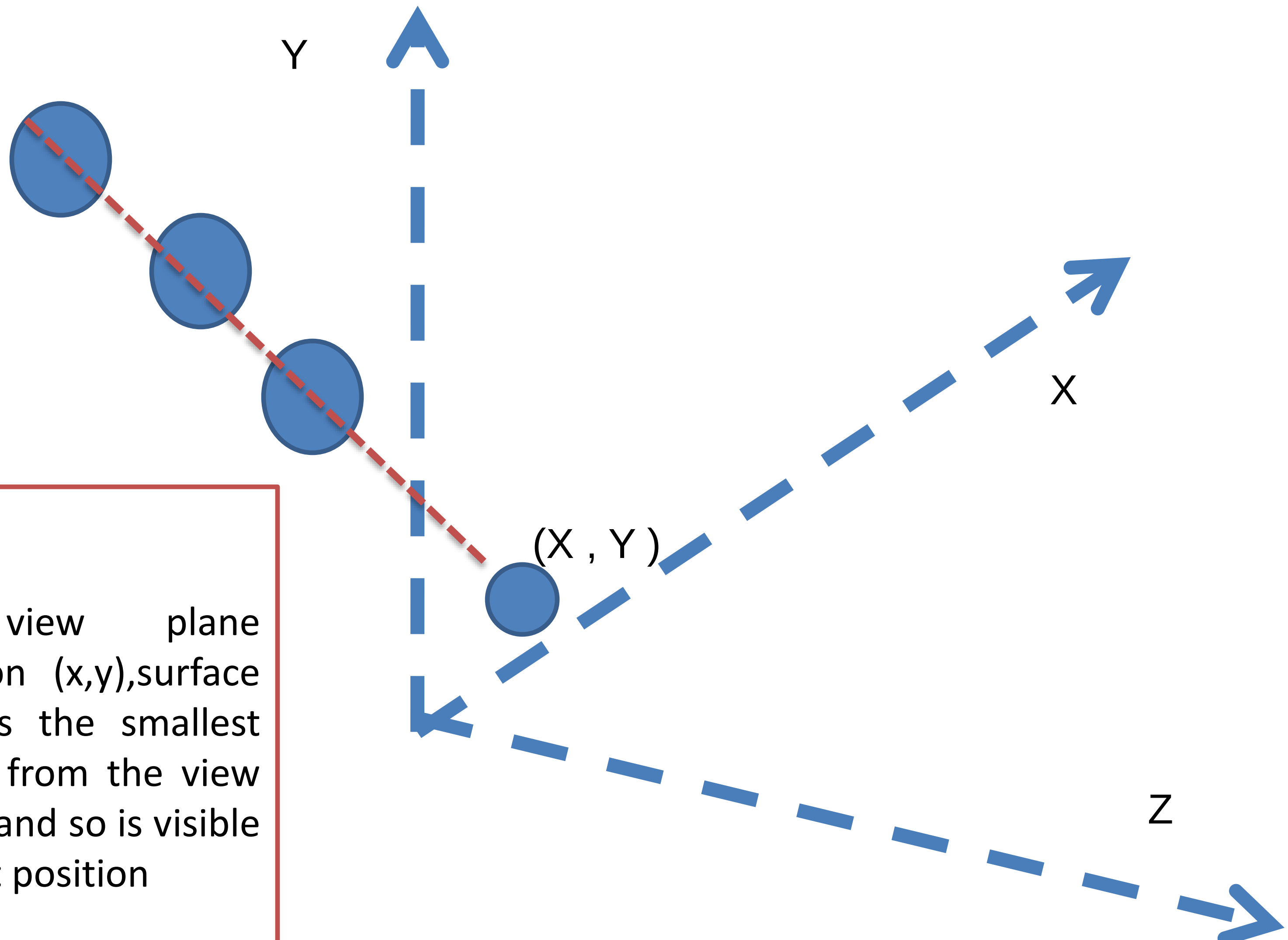
Perspective Projection

Z-Buffer (Depth Buffer) Algorithm



Z-Buffer (Depth Buffer) Algorithm





Result

At view plane position (x,y),surface S1 has the smallest depth from the view plane and so is visible at that position

Z-Buffer Algorithm

- Initialize frame buffer to background colour.
- Initialize z-buffer to minimum z value.
- Scan convert each polygon in arbitrary order.
- For each(x,y) pixel, calculate depth ‘z’ at that pixel(z(x,y)).
- Compare calculated new depth z(x,y) with value previously stored in z-buffer at that location z(x,y).
- If z(x,y)>z(x,y), then write the new depth value to z-buffer and update frame buffer.
- Otherwise, no action is taken.

- The plane polygon define a surface or plane whose equation can be written as

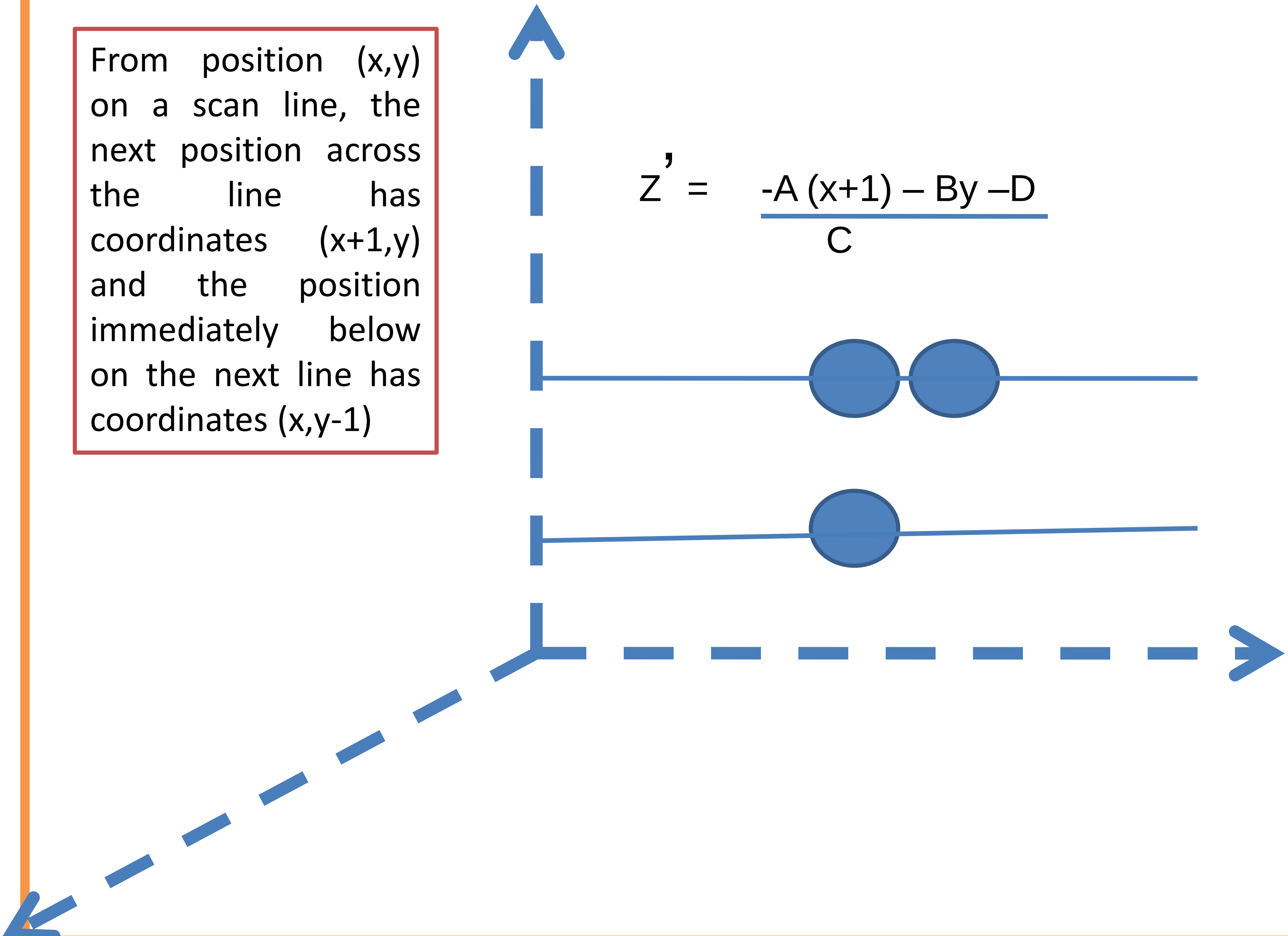
$$Ax + By + Cz + D = 0$$

Depth value for a surface position (x.y) are calculated from the plane equation for each surface

$$z = \frac{-Ax - By - D}{C}$$

From position (x,y) on a scan line, the next position across the line has coordinates (x+1,y) and the position immediately below on the next line has coordinates (x,y-1)

$$Z' = \frac{-A(x+1) - By - D}{C}$$



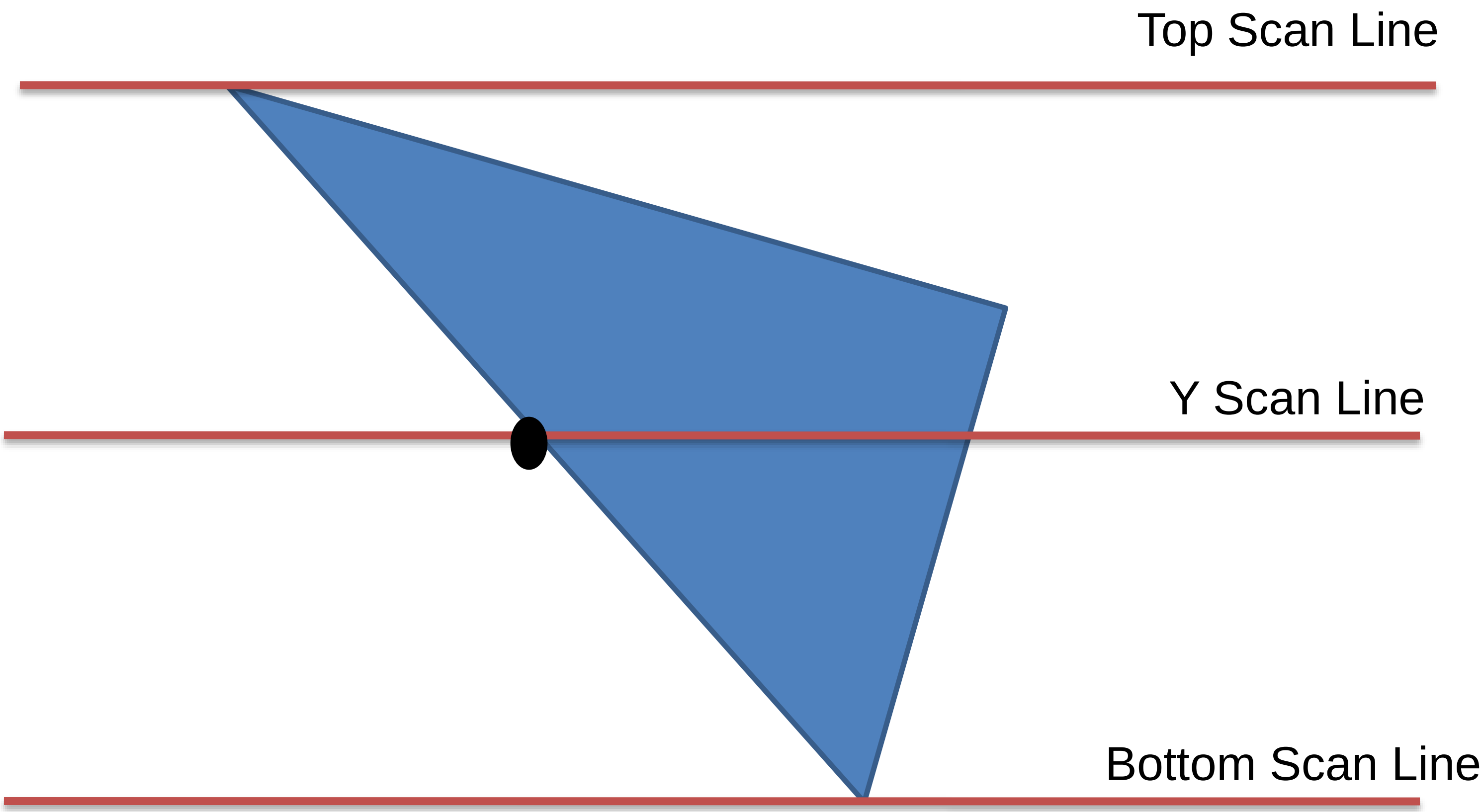


Fig: Scan Lines intersecting a polygon surface

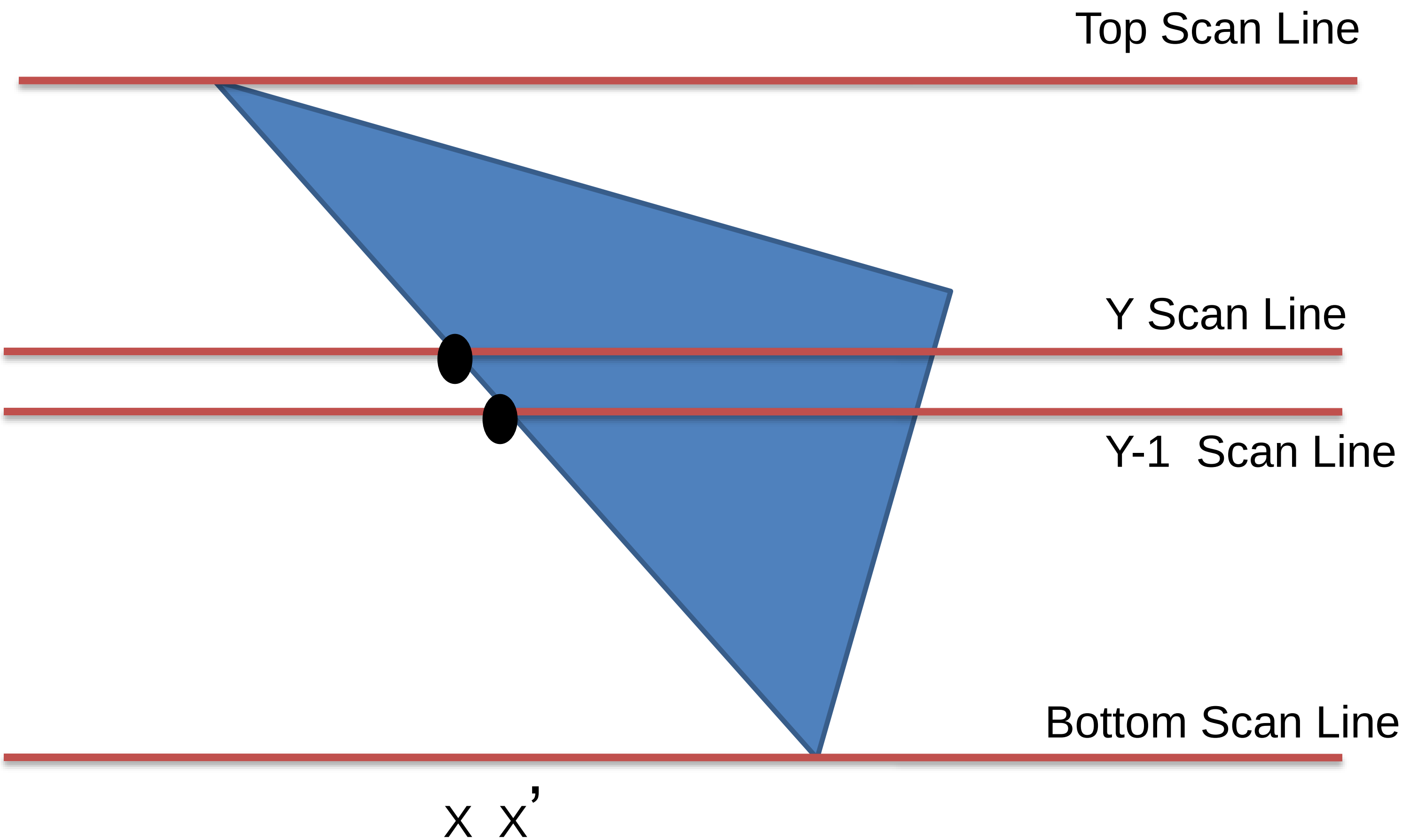
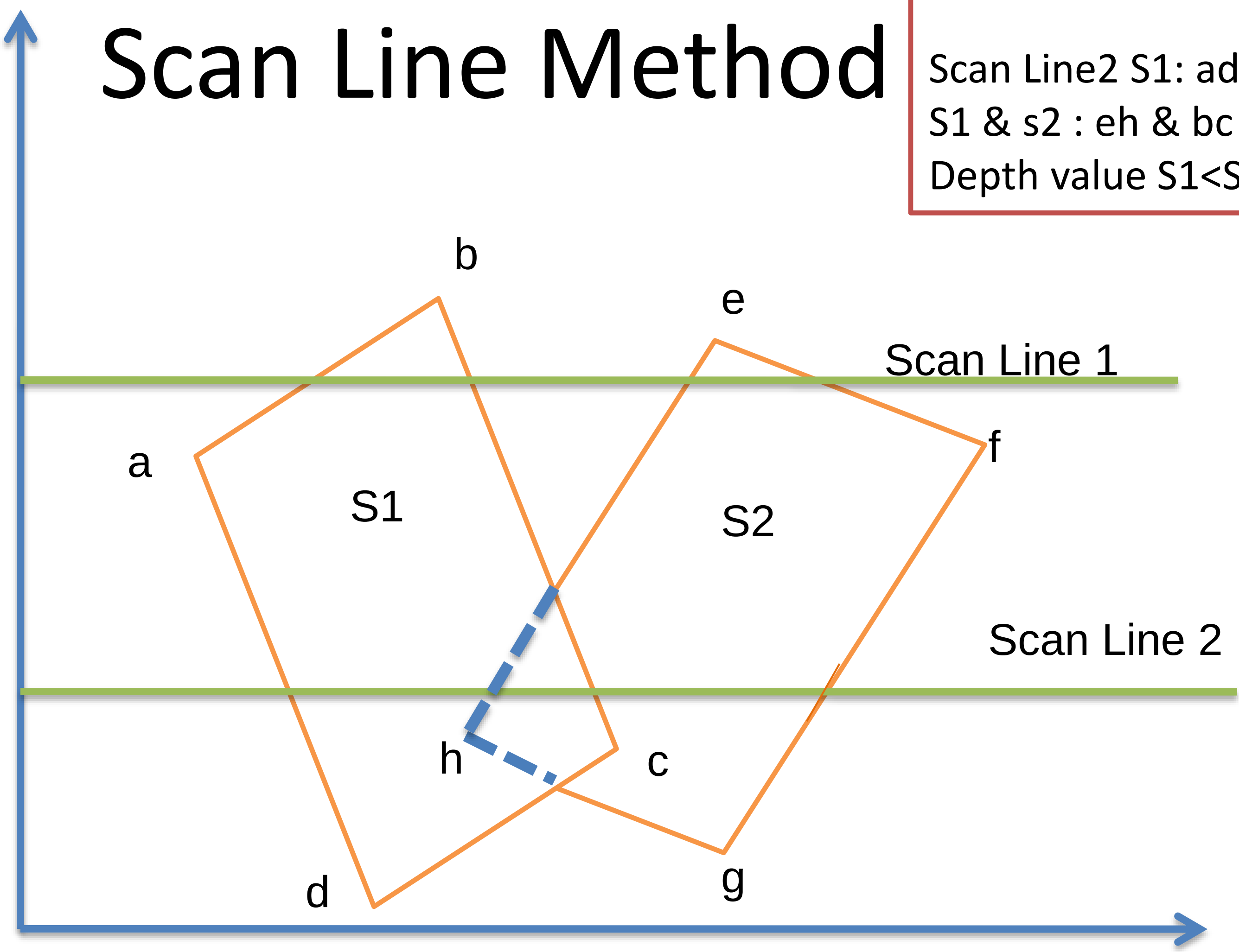


Fig: Intersecting positions on successive scan line along a Left Polygon edge

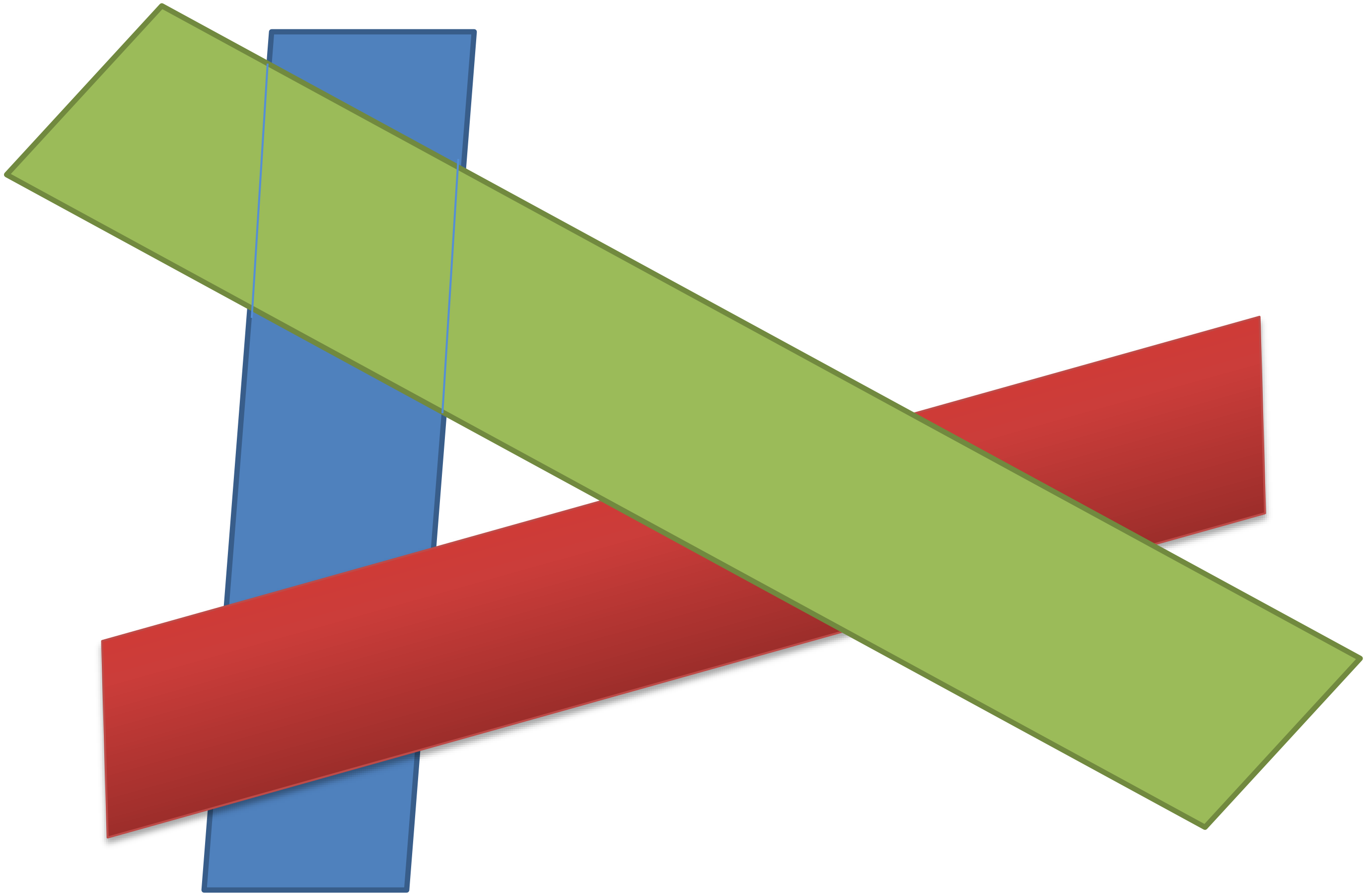
Scan Line Method

Scan Line1 S1: edge ab & bc , s2: eh &ef

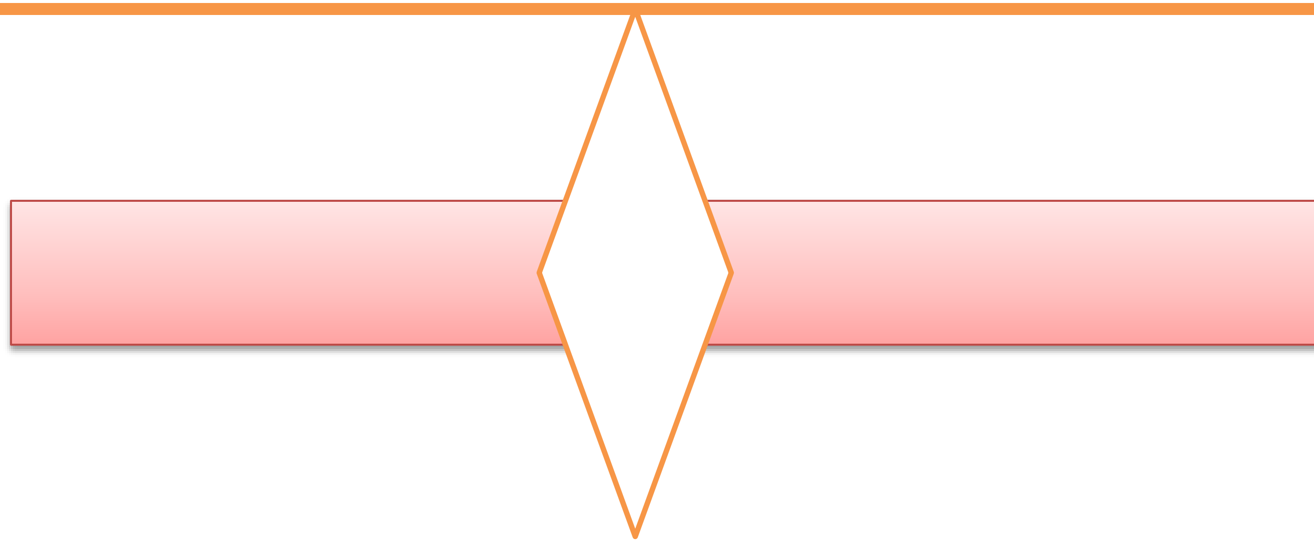
Scan Line2 S1: ad & eh , s2 :bc &fg
S1 & s2 : eh & bc
Depth value S1<S2 then s1 is visible



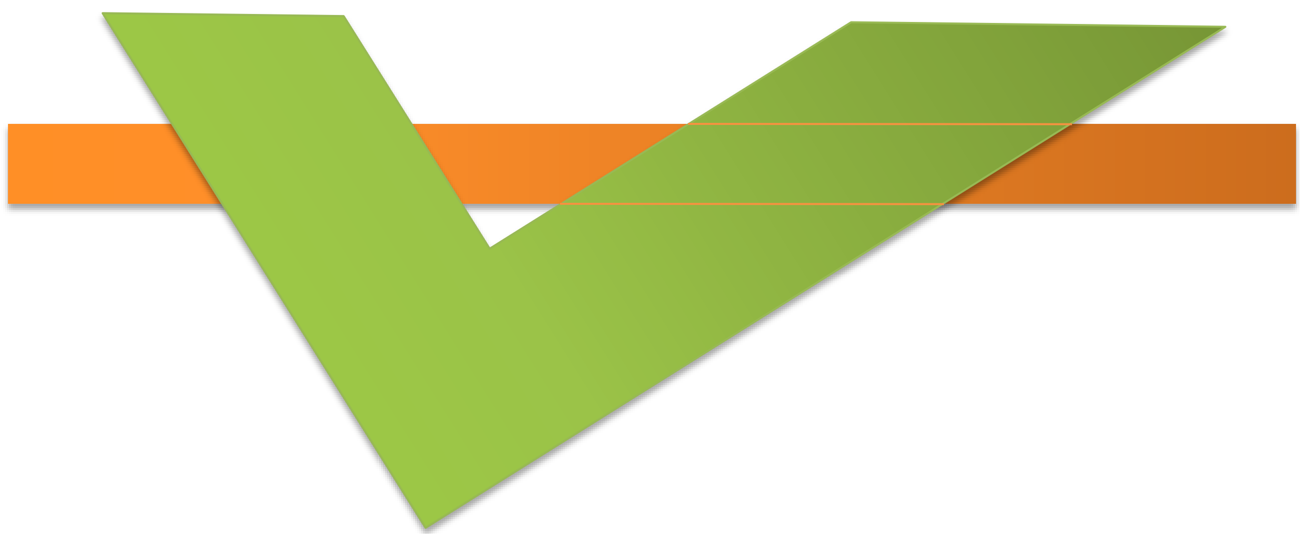
Scan Lines crossing the projection of two surface, S1 and S2 in the view plane. Dashed lines indicates the boundaries of hidden surfaces.



Intersecting and cyclically overlapping surfaces
that alternately obscure(Unclear) one anther



Intersecting and cyclically overlapping surfaces
that alternately obscure(Unclear) one anther

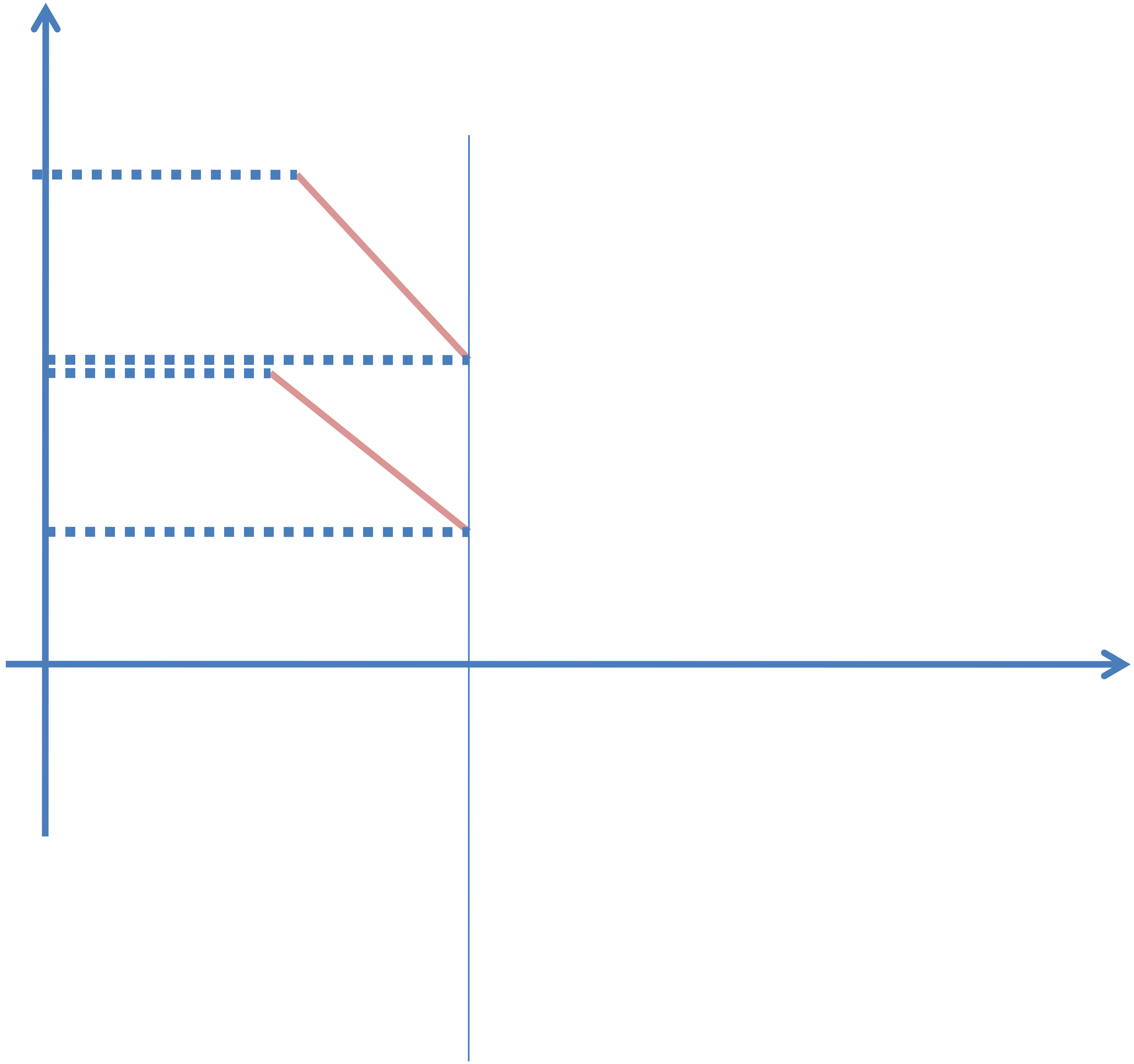


Painter Algorithm (Depth Method)

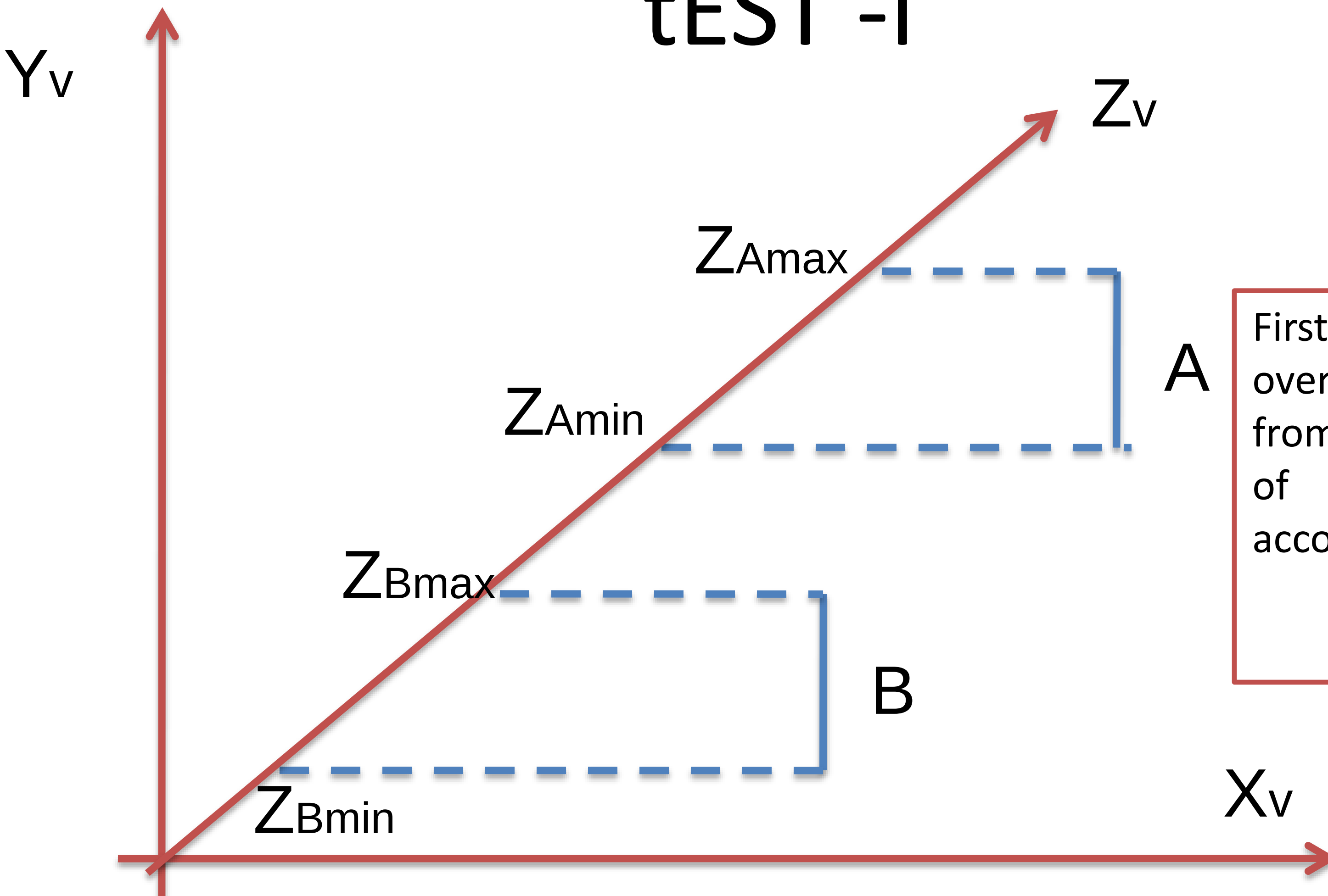
- Using both image space and object space operations, the depth-sorting method performs the following basic functions:
  - Surfaces are sorted in order of decreasing depth.
  - Surfaces are scan converted in order, starting with the surface of greatest depth.

  Used : Oil Painting ,an artist first paints the

  background color



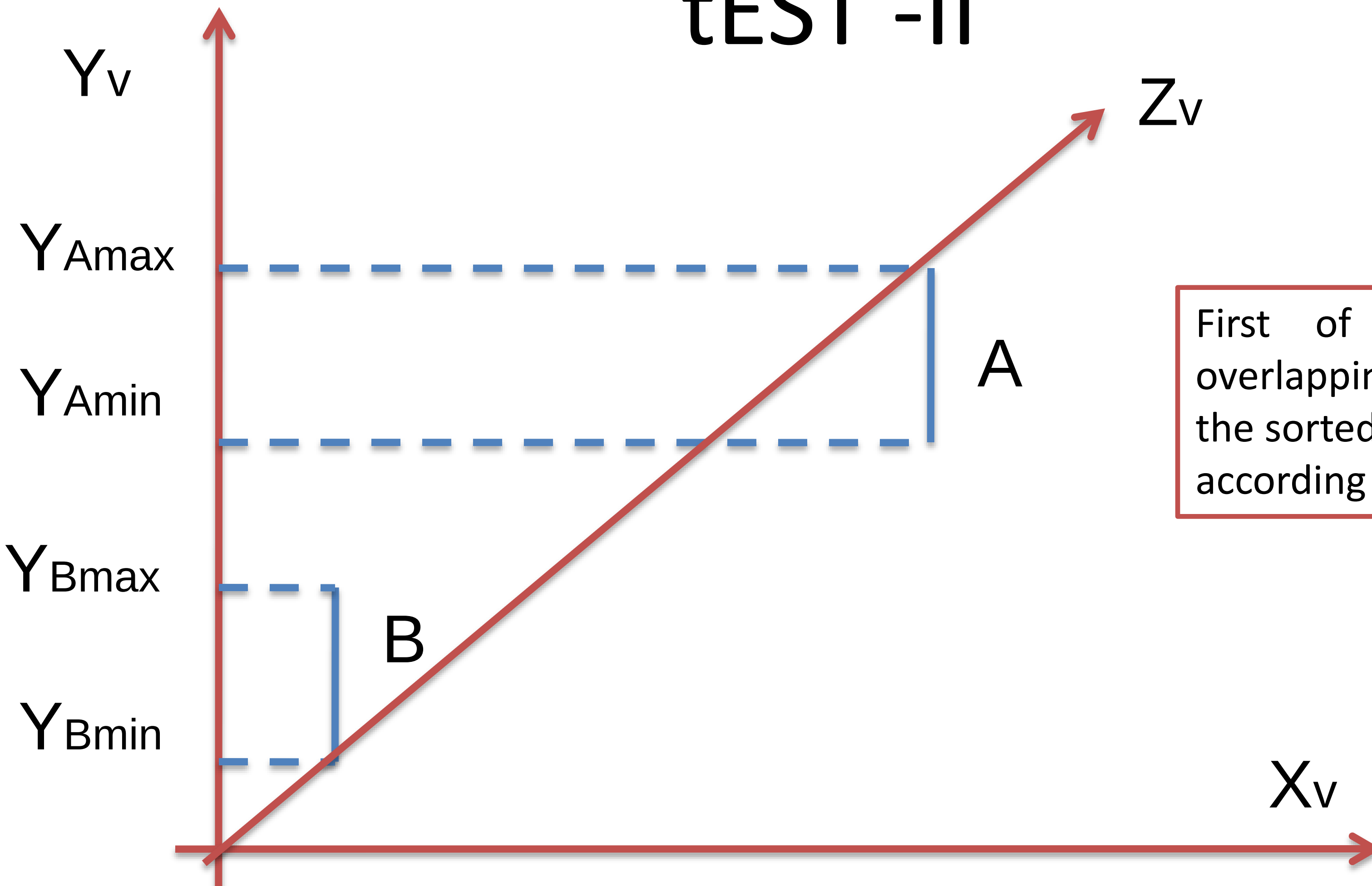
tEST -I



First of all find the overlapping in z axis from the sorted list of polygons according to z value

Surface A and B with no overlapping in Z direction

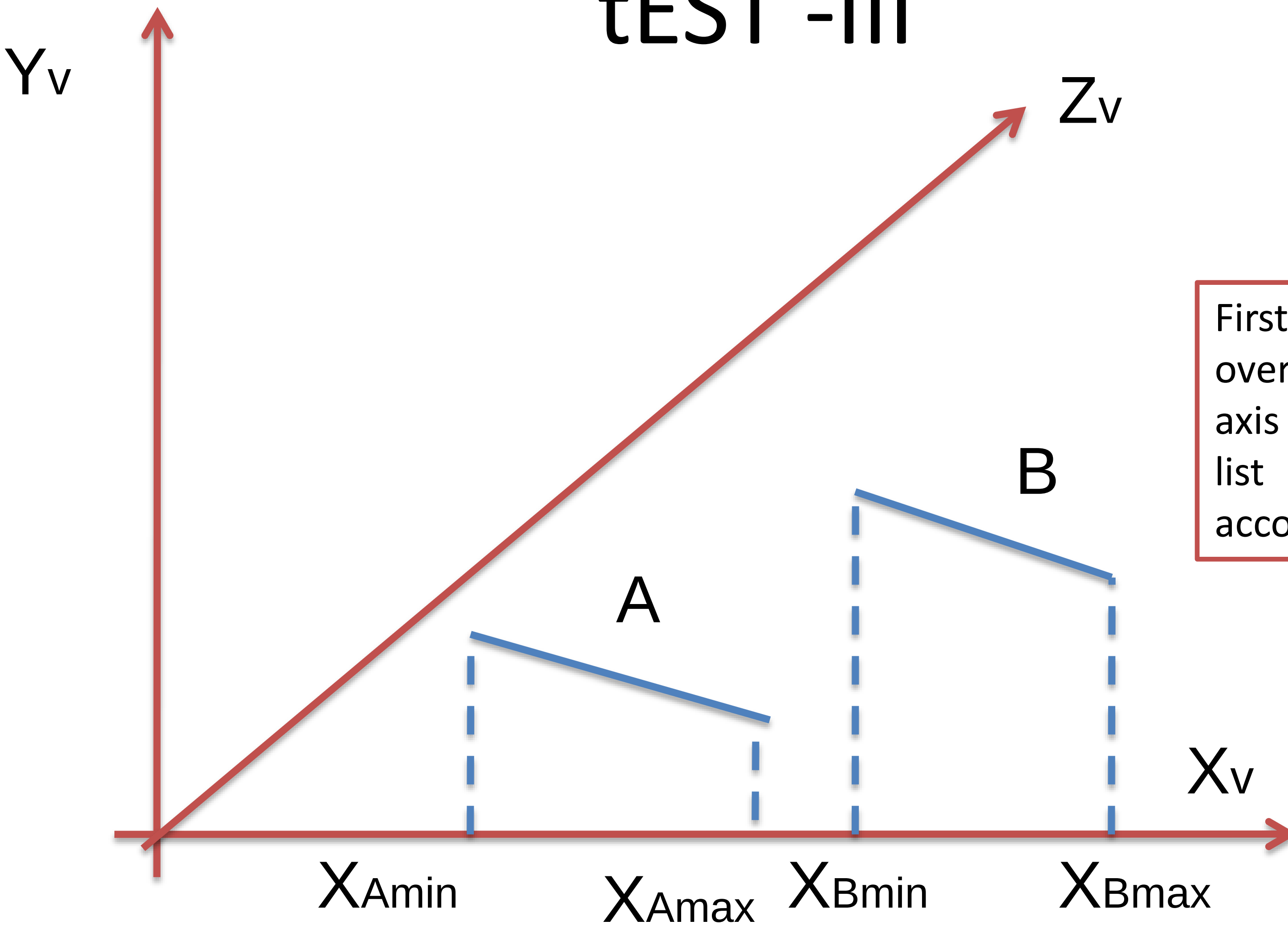
tEST -II



First of all find the overlapping in y axis from the sorted list of polygons according to y value

Surface A and B with no overlapping in Y direction

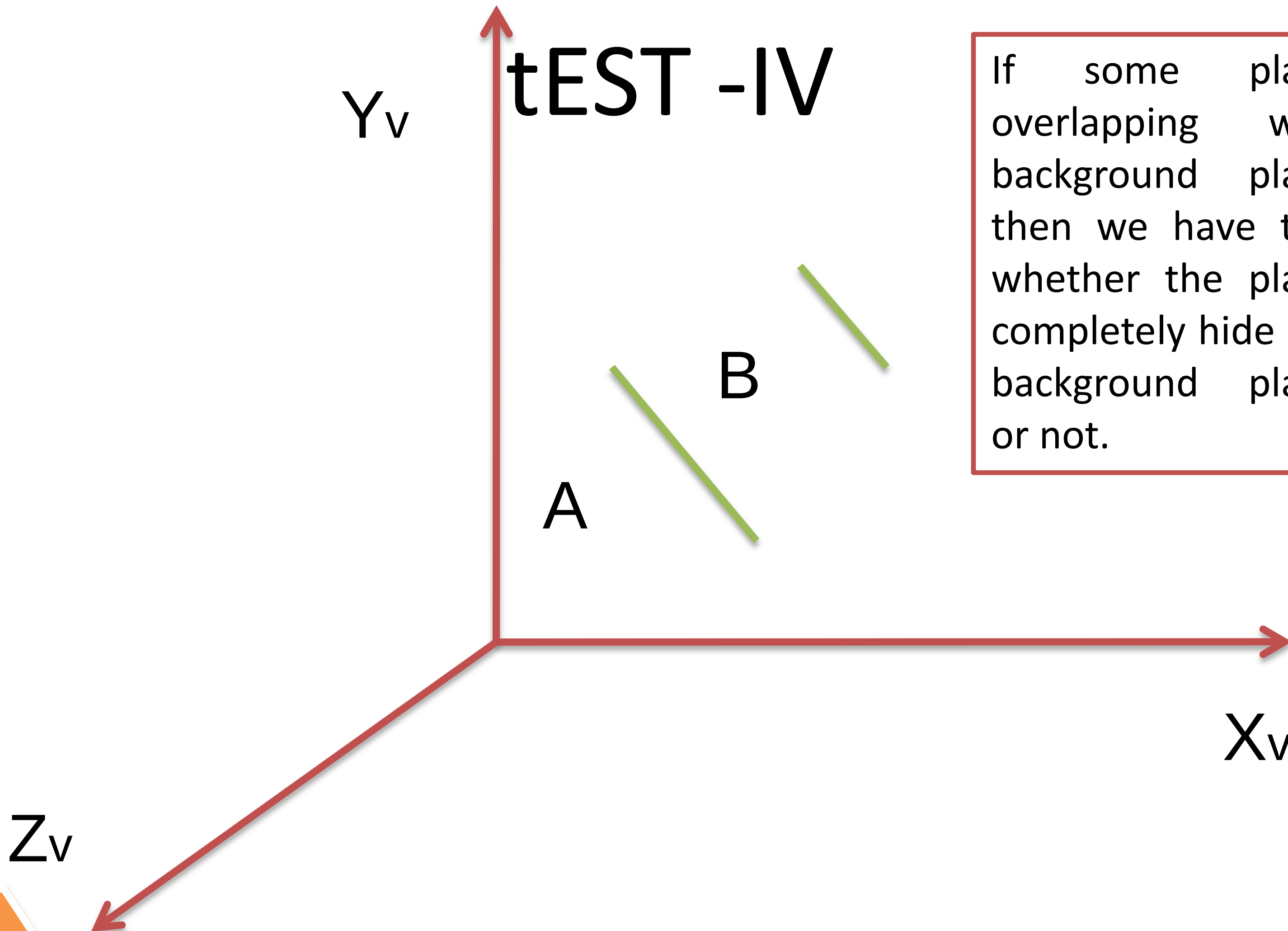
tEST -III



First of all find the overlapping in x axis from the sorted list of polygons according to x value

Surface A and B that do not overlap in x direction

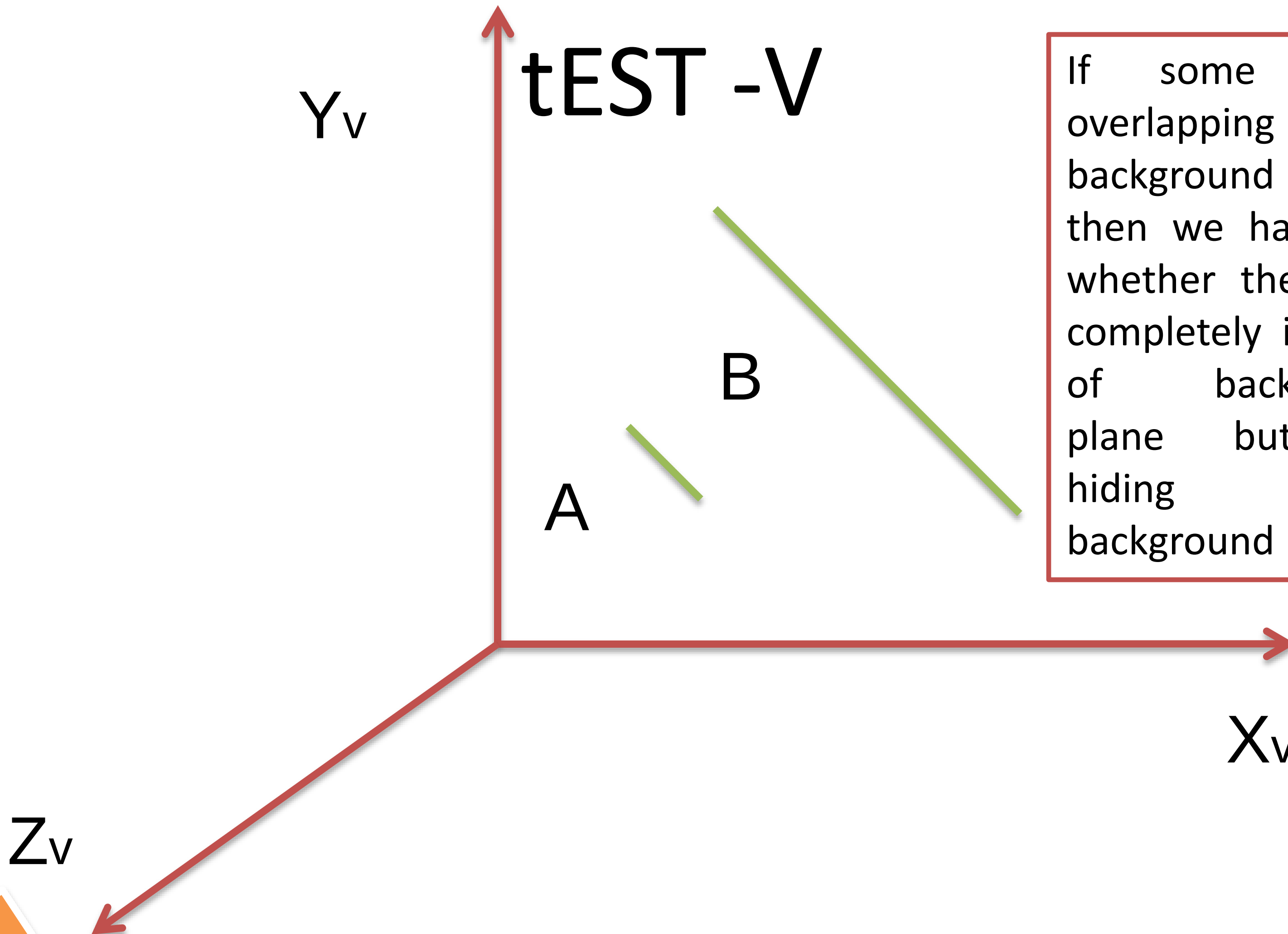
tEST -IV



If some plane overlapping with background plane then we have test whether the plane completely hide the background plane or not.

Surface A completely overlaps the background plane B

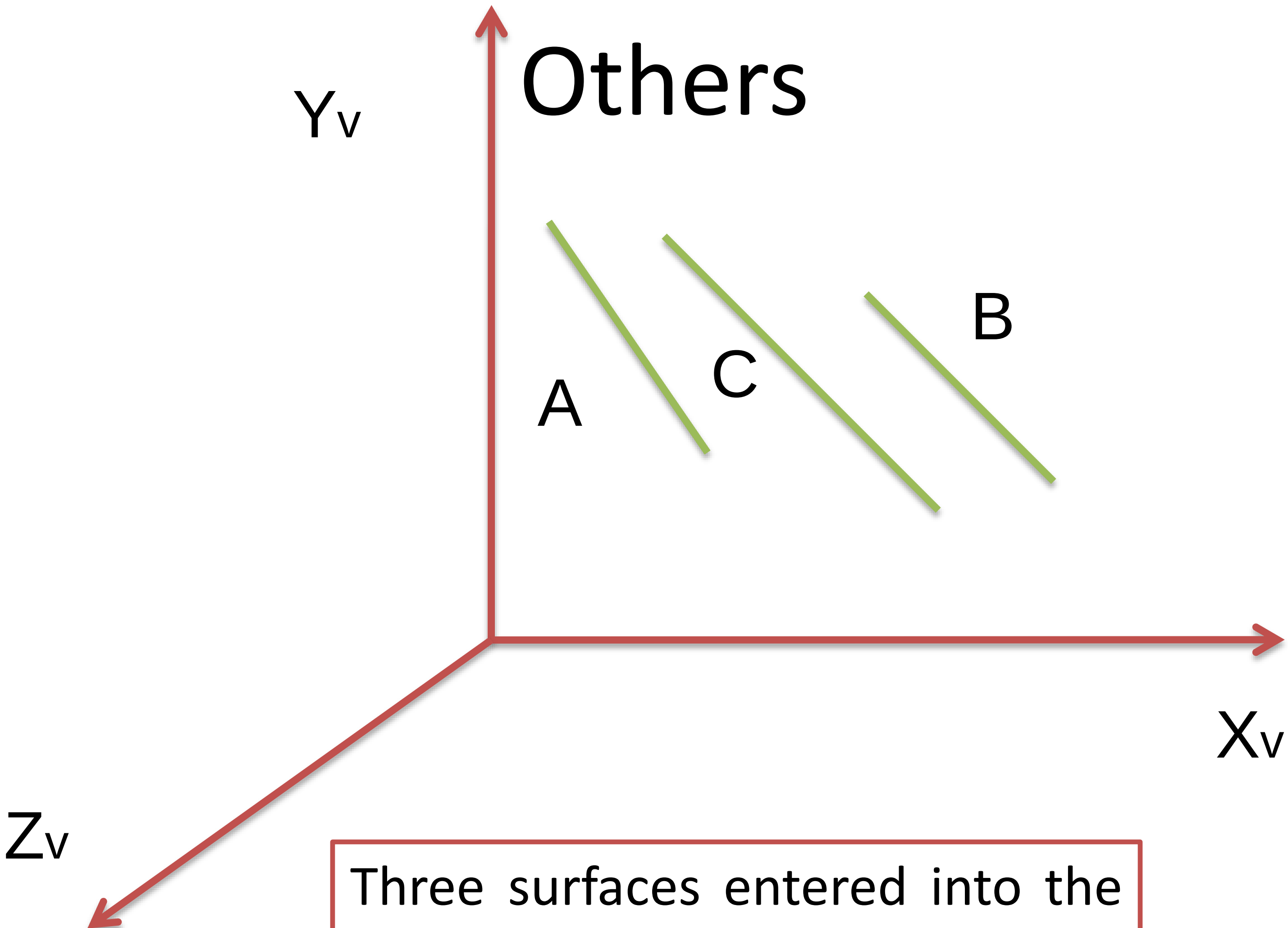
tEST -V



If some plane overlapping with background plane then we have test whether the plane completely in front of background plane but not hiding the background plane.

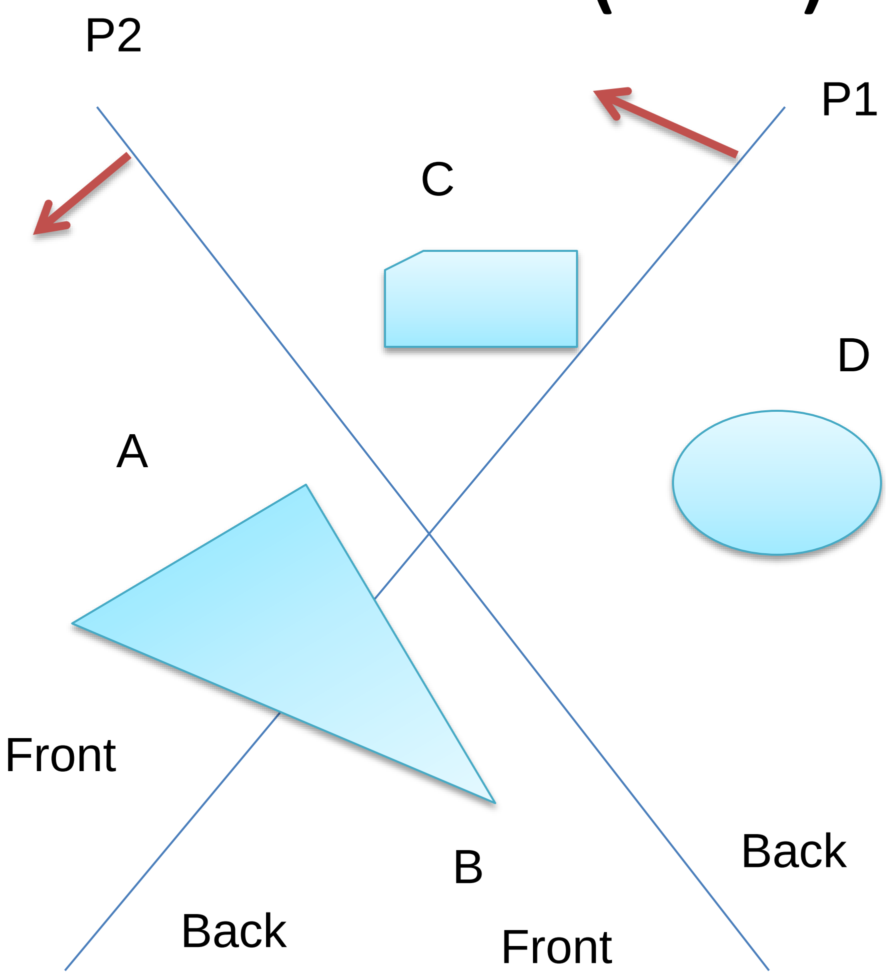
Surface A does not hide B

Others



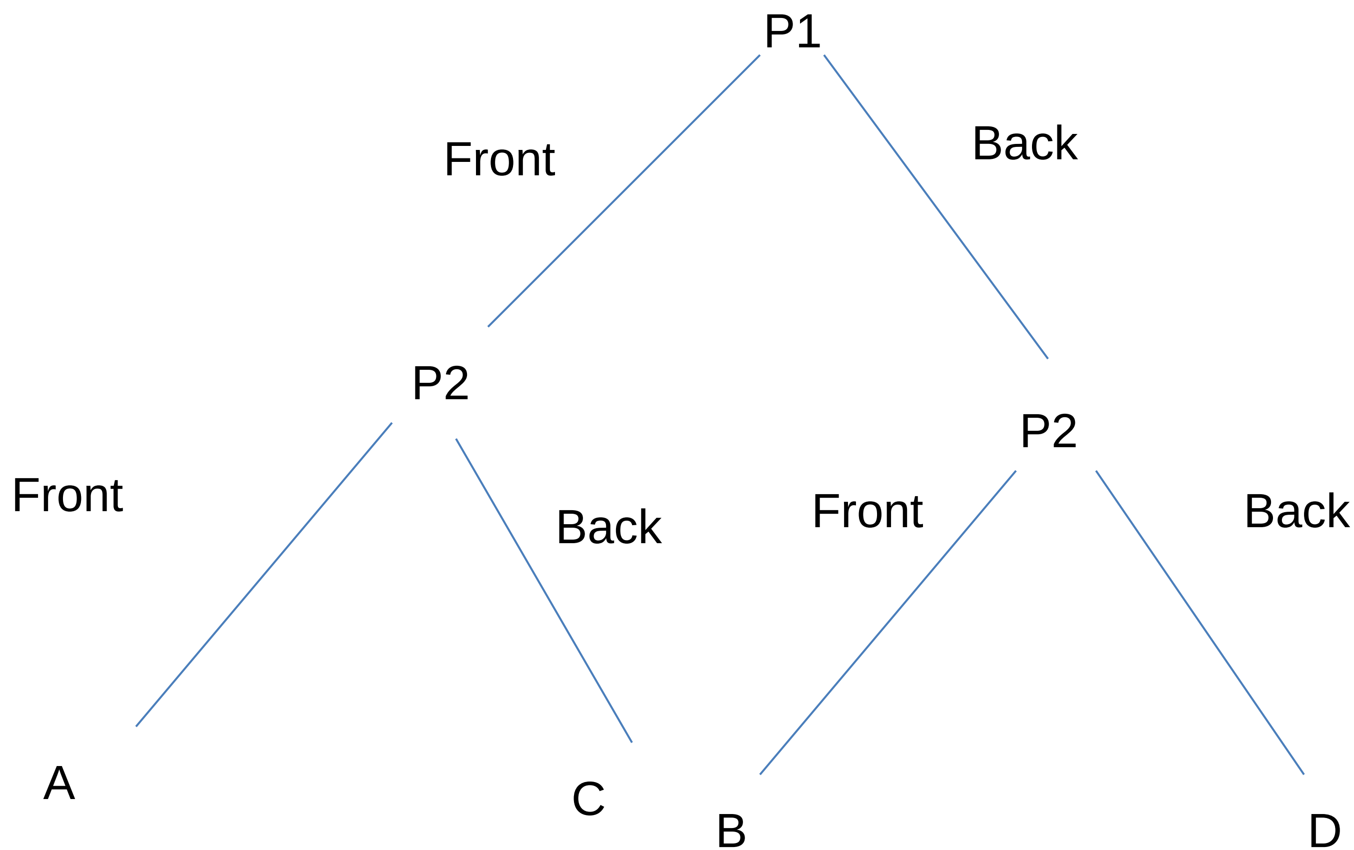
Three surfaces entered into the sorted surface list in the order A,B,C should be reordered A, C, B.

Binary Space Portitioning Tree (BSP) Method



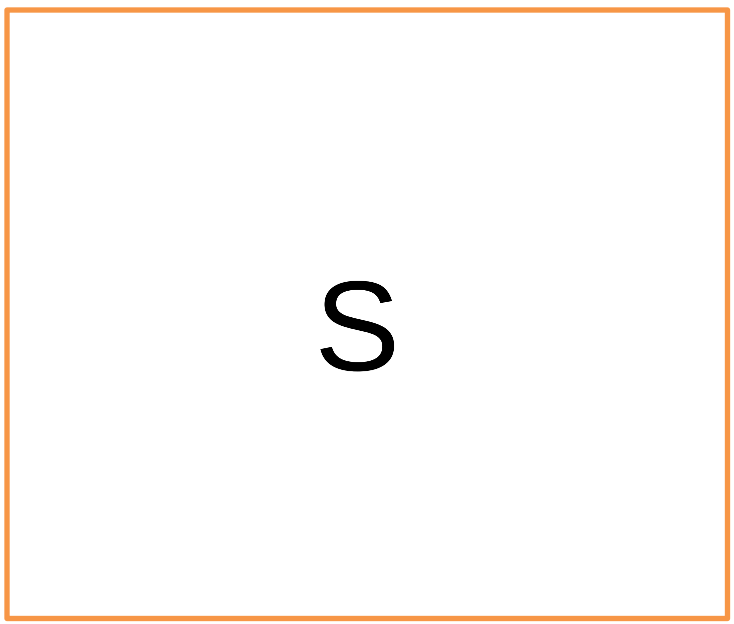
View	P1	
FRONT	P2	A , B
BACK	P2	C, D

A region of space is partitioned with two planes P1 and P2.

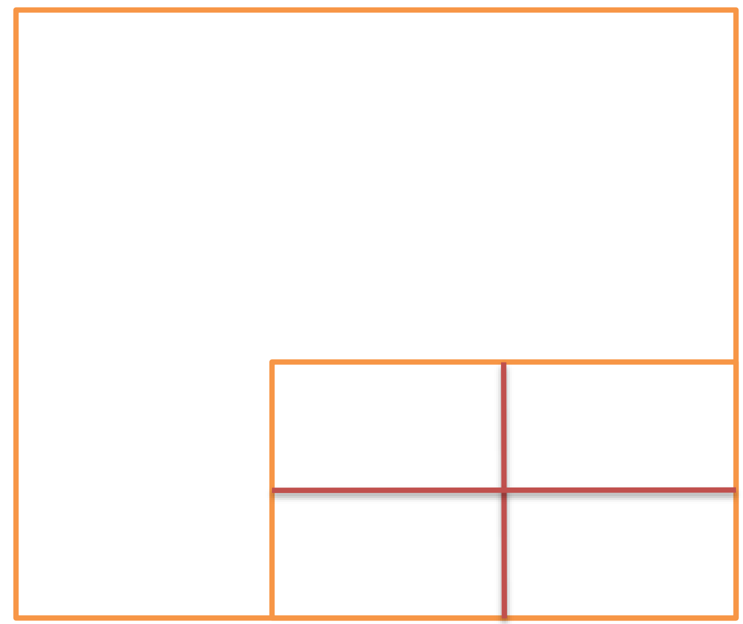


BSP Tree Representation

aReA-sUbDiViSiOn MeThOd (WoRnOcK's AlGoRiThM)



Initial Viewing Area

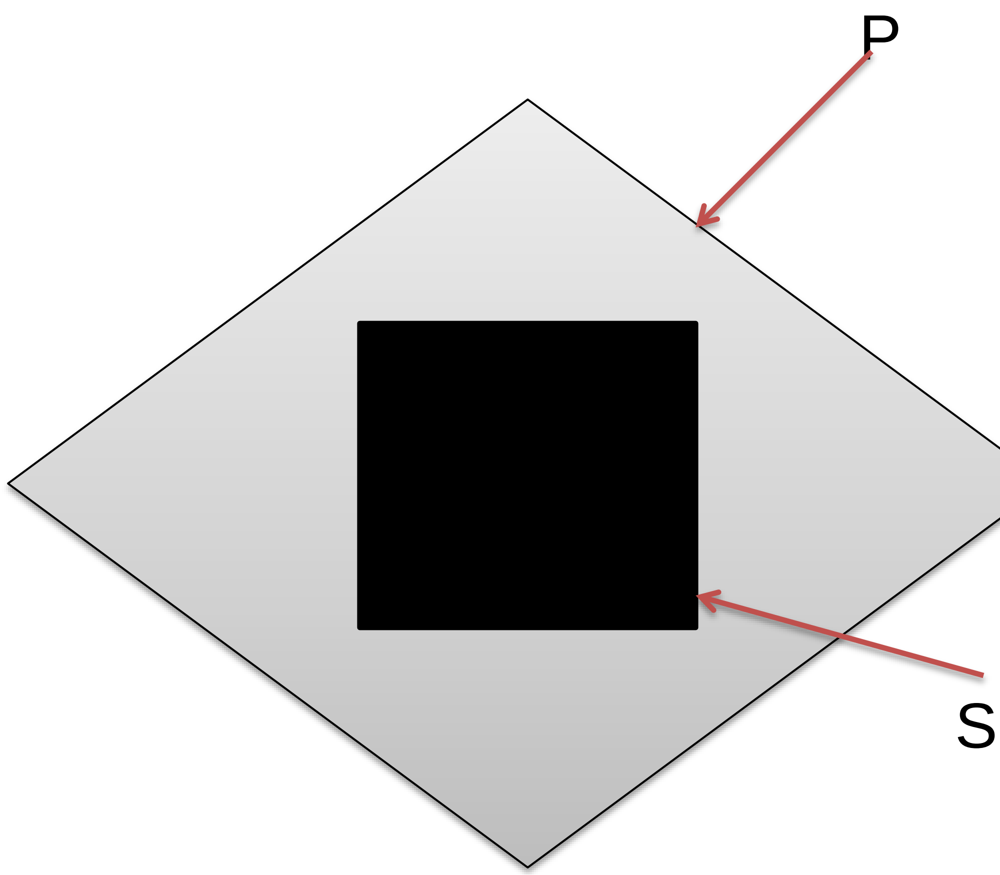


Subdivision of Viewing Area

- The relationship between projection each polygon and the area of interest is checked for four possible relationships :

  - Surrounding Surface
  - Overlapping OR Intersecting Surface
  - Inside OR Contained Surface
  - Outside OR Disjoint Surface

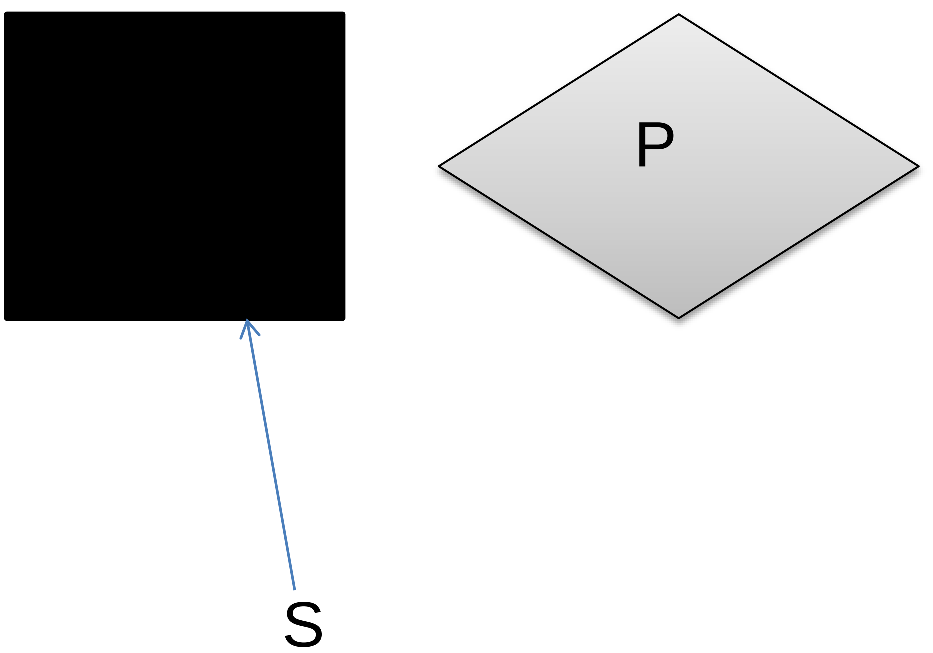
Case I – Surrounding Surface



Polygon that completely surrounds the screen area is called surrounding surface .

If polygon is surrounding the screen area , color the all pixels of screen area as color of screen.

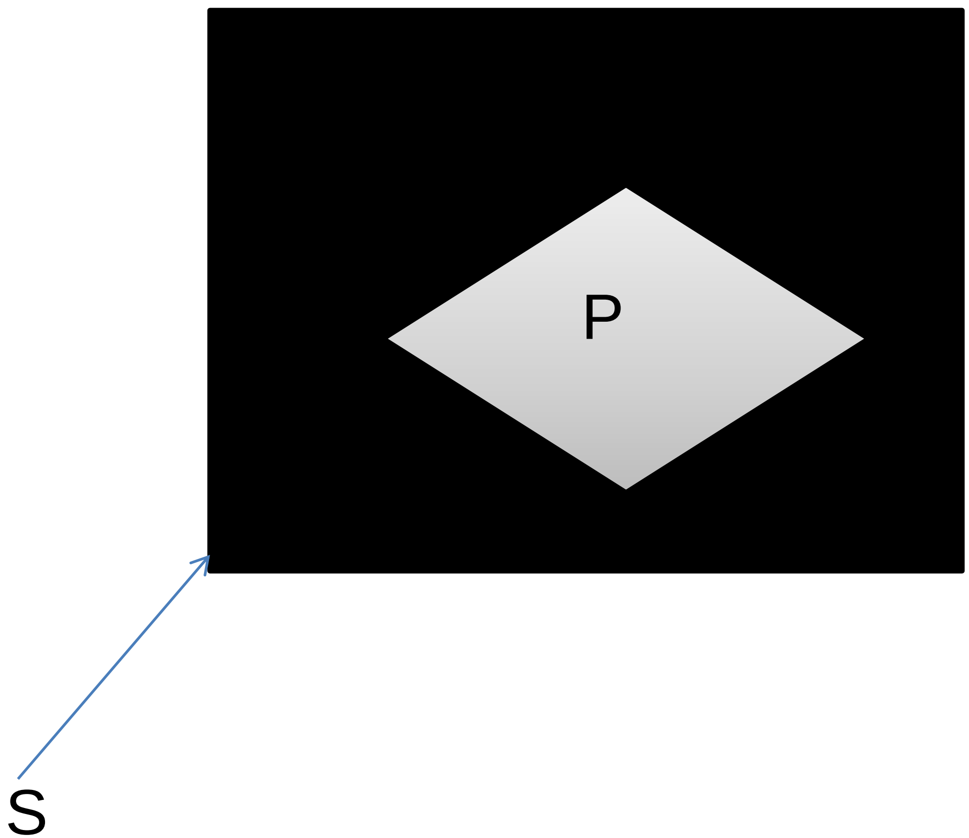
Case II – Outside or Disjoint Surface



Polygon that is completely outside the screen area.

If all polygon comes under the category of outside polygon, color all pixel of viewing screen as background color.

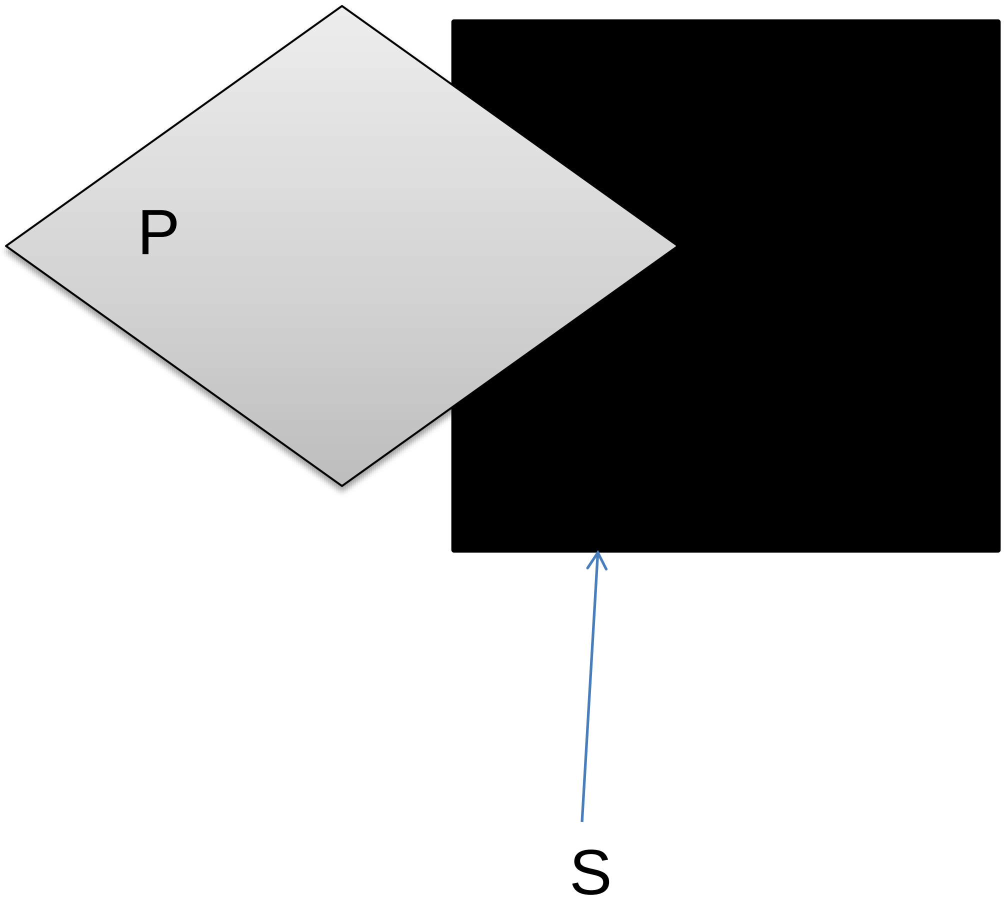
Case III – Contained or Inside Surface



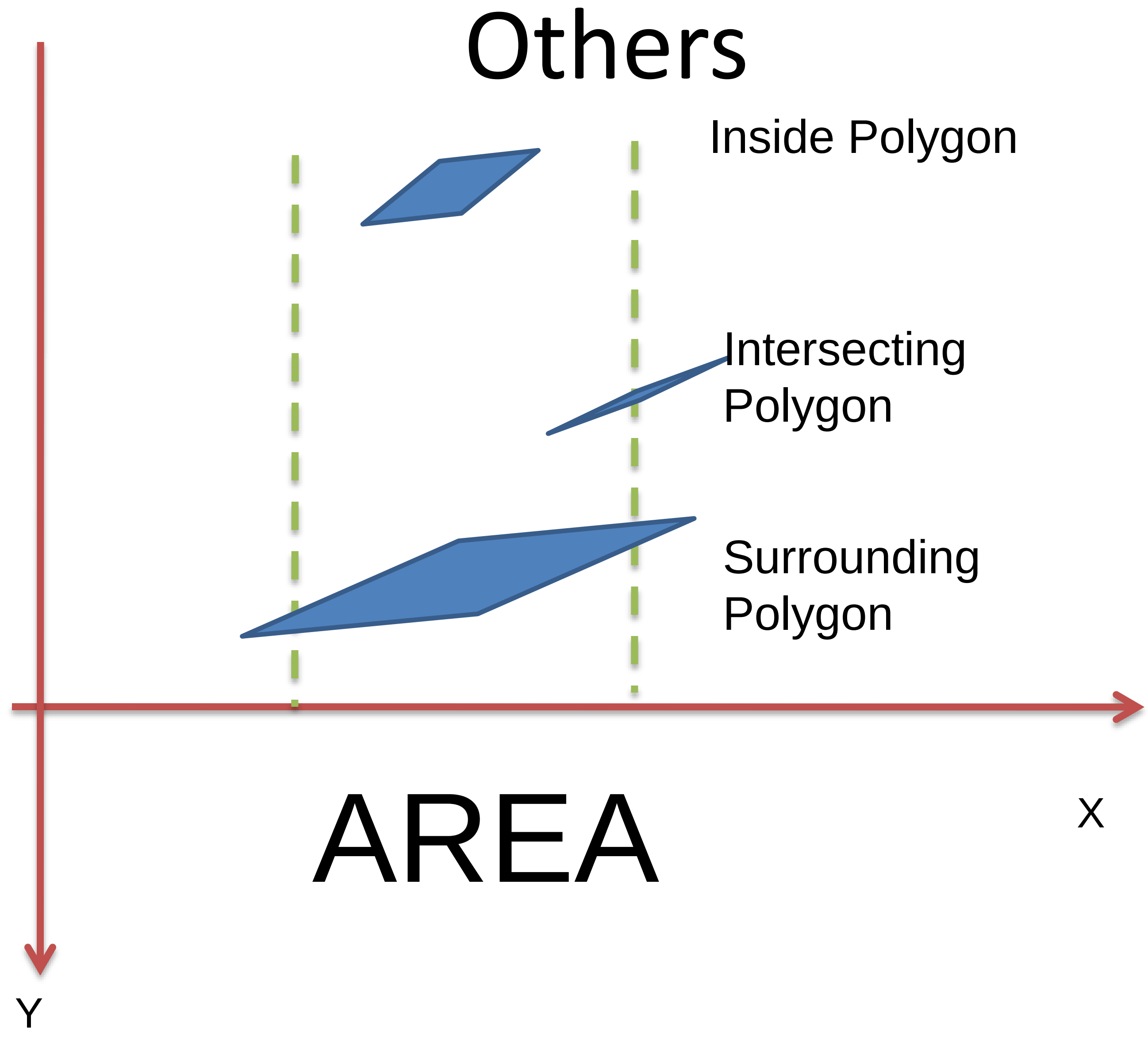
Polygon that is completely inside the screen area.

If polygon in inside the screen area, we scan convert that area and the remaining area of screen will colored with background color.

Case IV – Intersecting or Overlapping Surface



Polygon that intersect the screen area S.


Others
Inside Polygon
Intersecting
Polygon
Surrounding
Polygon
AREA
X
Y

Others