UNIT-3-VISUAL REALISM

VISUALIZATION

- Visualization can be defined as a technique for creating images , diagrams or animations to communicate ideas.
- Projection and shading are common methods for visualizing geometric models.
- CAD uses isometric and perspective projection in addition to orthographic projection for generating rich visual images with complete design information.

- To project 3D to 2D objects we need to remove the ambiguities of the different views, which can be got by the elimination of hidden lines , surfaces , solid removal approaches.
- **Shading,**
- **Lighting**
- **Transparency**
- **Coloring** approaches provide more visual realism

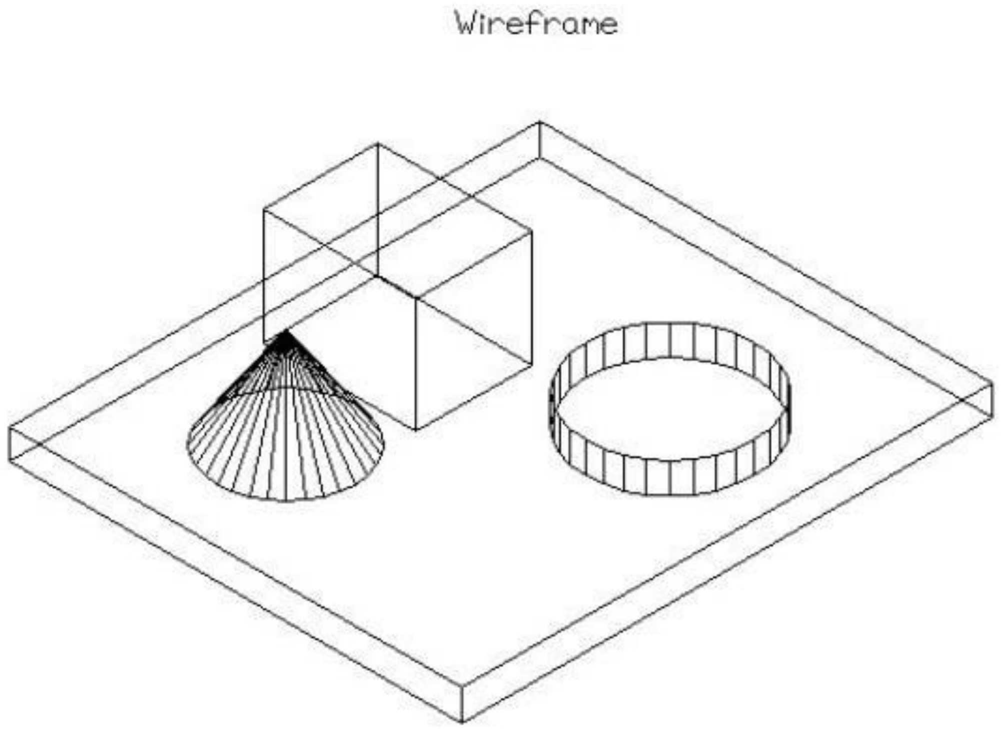
Model clean up process

- Generate orthographic views
- Eliminate hidden lines
- Changing necessary hidden lines as dashed line or adding dimension and text to the different views.

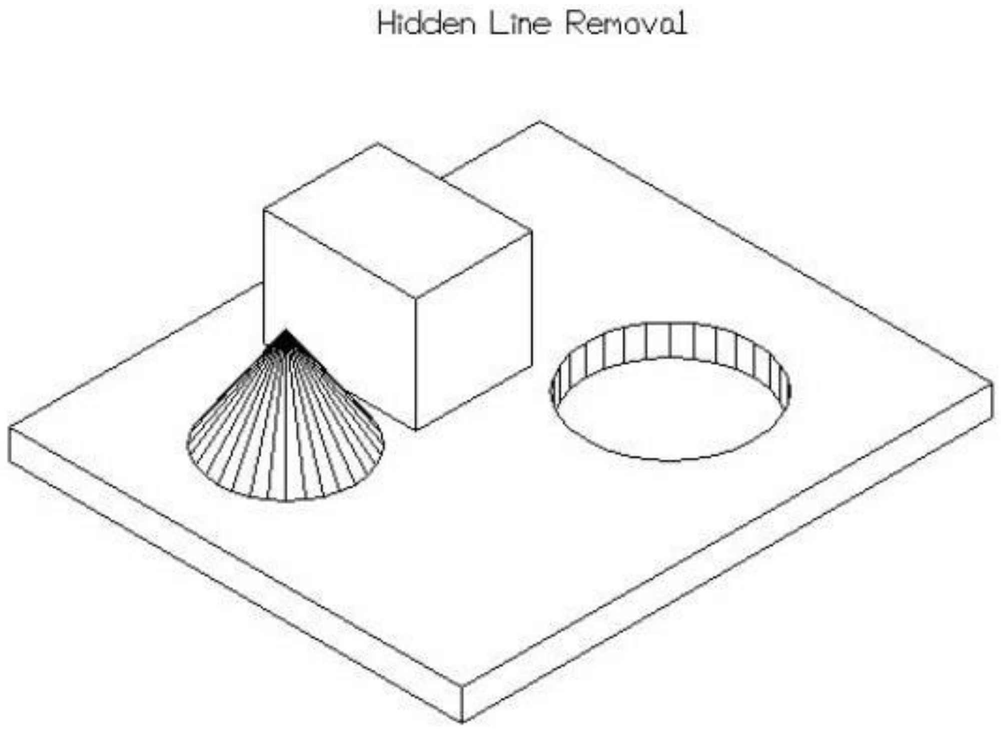
Application of realism

- **Robot Simulations :** Visualization of movement of their links and joints and end effector movement etc.
- CNC programs verification of tool movement along the path prescribed and estimation of cup height and surface finish etc.
- **Discrete Even Simulation :** Most of DES packages provide the user to create shop floor environment on the screen to visualize layout of facilities, movement of material handling systems, performance of machines and tools.
- **Scientific Computing :** Visualization of results of FEM analysis like iso-stress and iso-strain regions, deformed shapes and stress contours. Temperature and heat flux in heat-transfer analysis. Display and animation of mode shape in vibration analysis.
- **Flight Simulation :** Cockpit training for pilots is first being provided with flight simulators, which virtually simulates the surrounding that an actual flight will pass through.

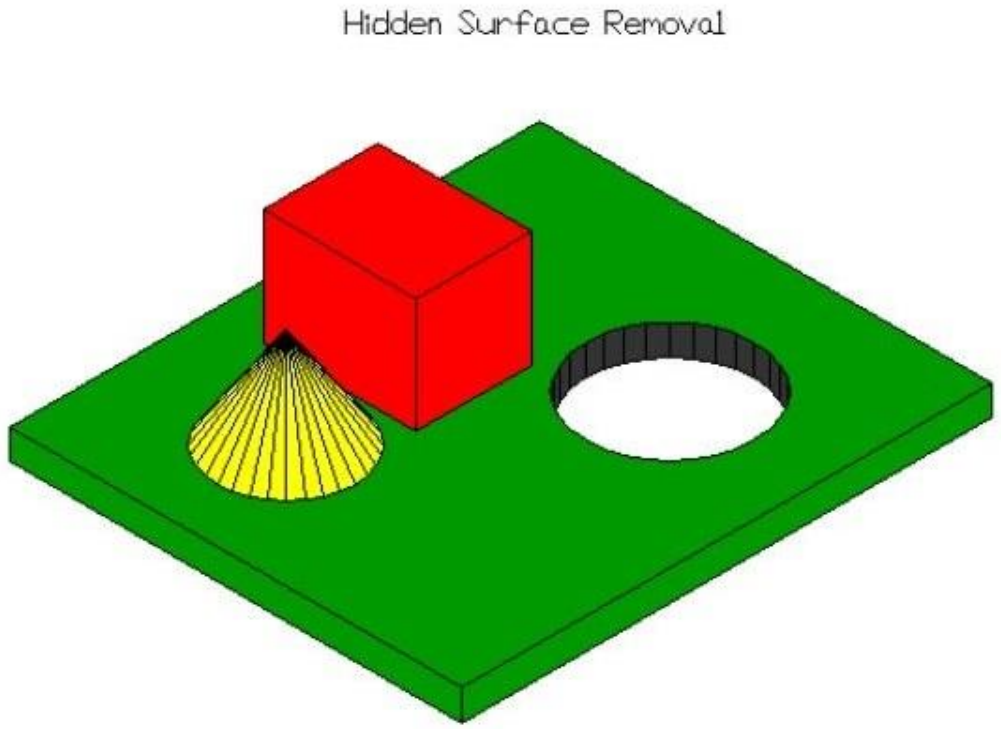
No Lines Removed



Hidden Lines Removed



Hidden Surfaces Removed



HIDDEN LINE REMOVAL

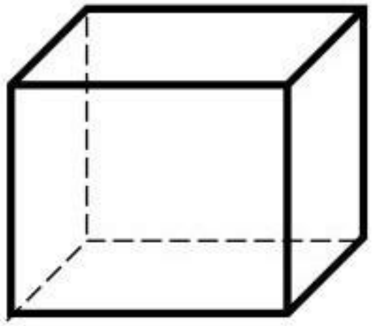
- *"For a given three dimensional scene, a given viewing point and a given direction eliminate from an appropriate two dimensional projection of the edges and faces which the observer cannot see"*

- **Object space method**
- **Image space method**

Two main types of algorithms:

- **Object space**: Determine which part of the object are visible. Also ca as World Coordinates. Object is described in physical coordinate syste
- It compares the object and parts to each other within the scene definiuon to determine which surface is visible.

- **Image space**: Determine per pixel which point of an object is visible. Also called as Screen Coordinates. Visibility is decided point by point at each pixel position on view plane.
- Zooming does not degrade the quality.


Object space

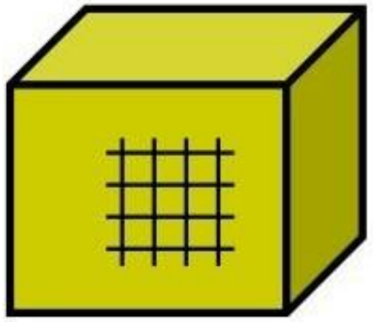


Image space

Two main Hidden Surface Removal Algorithm Techniques:

Object space: Hidden Surface Removal for all objects

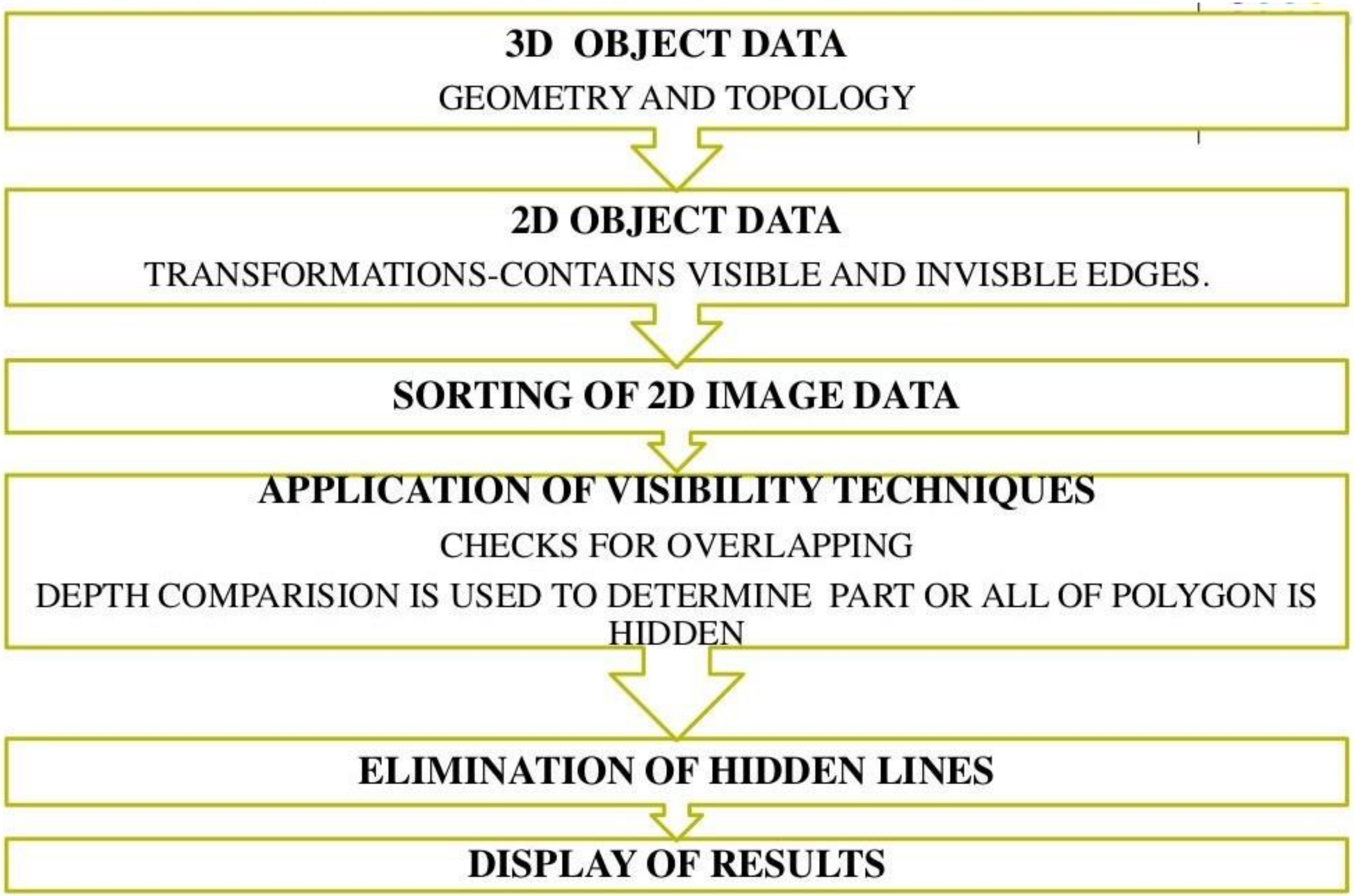
Image space:

Objects: 3 D Clipping

Transformed to Screen Coordinates

Hidden Surface Removal

HIDDEN LINE ELIMINATION PROCESS



3D OBJECT DATA

GEOMETRY AND TOPOLOGY

2D OBJECT DATA

TRANSFORMATIONS-CONTAINS VISIBLE AND INVISBLE EDGES.

SORTING OF 2D IMAGE DATA

APPLICATION OF VISIBILITY TECHNIQUES

CHECKS FOR OVERLAPPING

DEPTH COMPARISION IS USED TO DETERMINE PART OR ALL OF POLYGON IS HIDDEN

ELIMINATION OF HIDDEN LINES

DISPLAY OF RESULTS

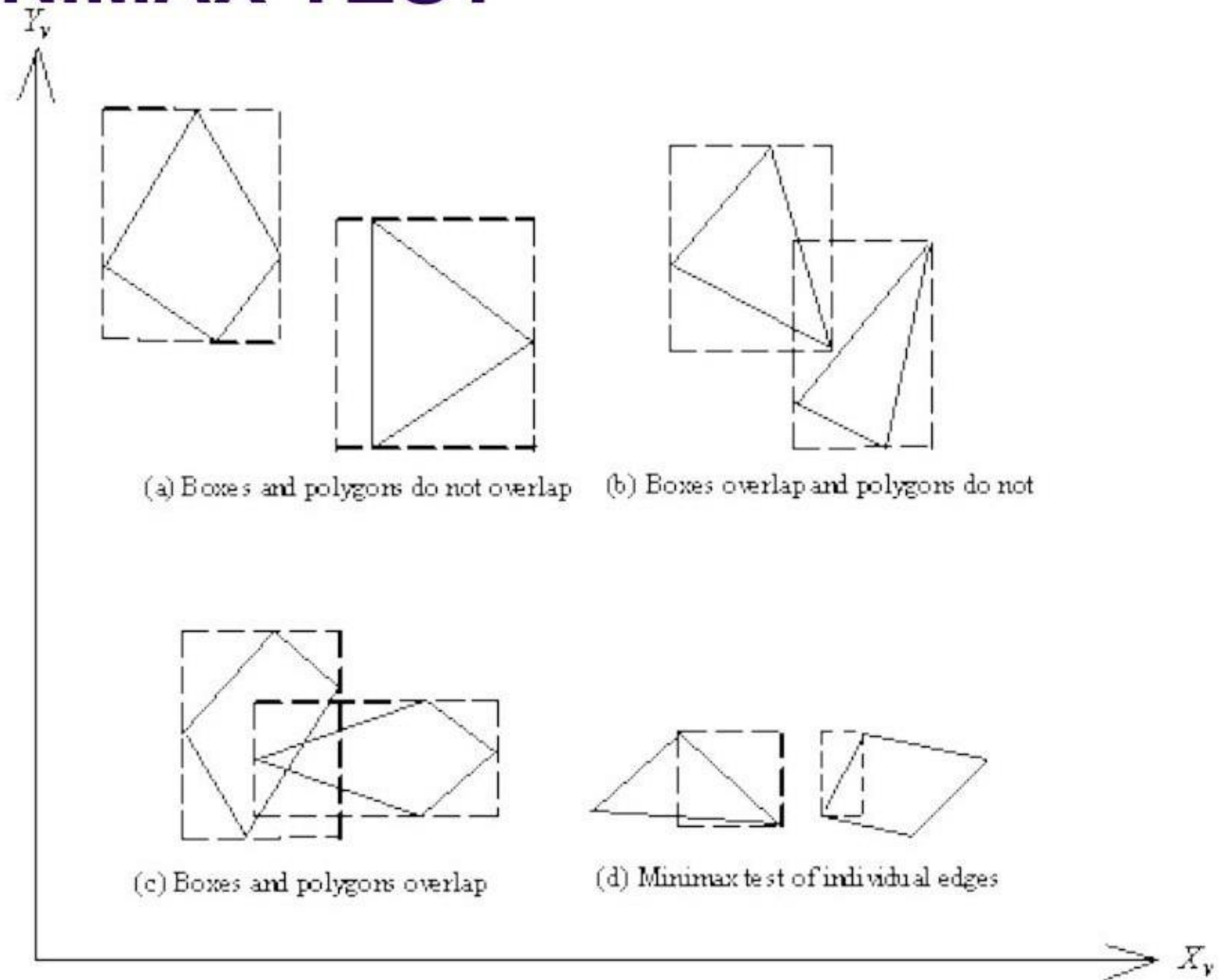
VISIBILITY TECHNIQUES

- MINIMAX TEST
- CONTAINMENT TEST
- SURFACE TEST
- COMPUTING SILHOUETTES
- EDGE INTERSECTION
- SEGMENT COMPARISONS
- HOMOGENITY TEST

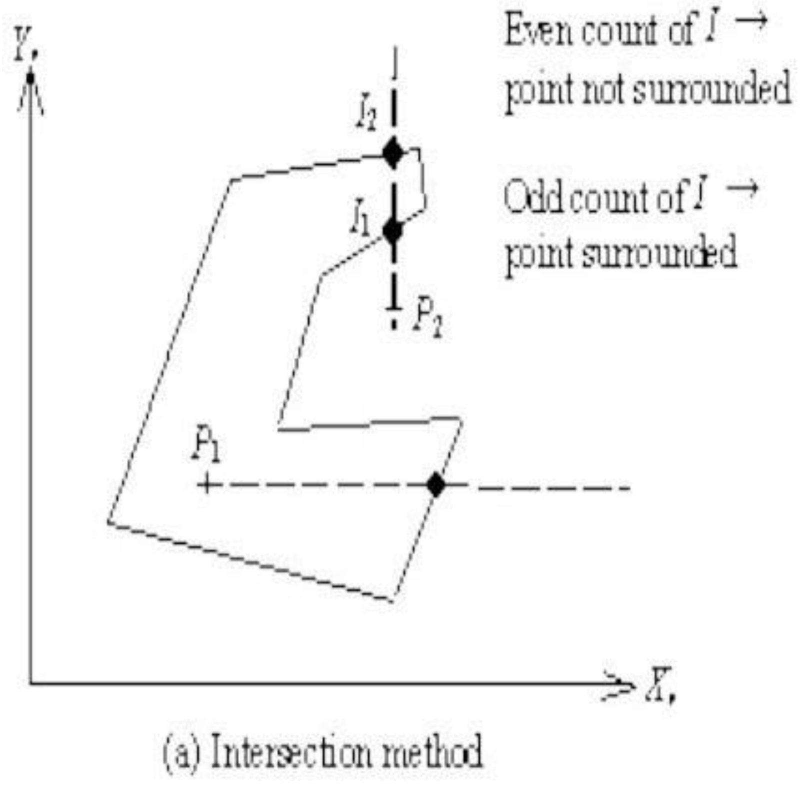
MINIMAX TEST

- Minimax test (also called the overlap or bounding box test) check polygons overlap. **The test provides a quick method to determine if two polygons do not overlap.**
- It surrounds each polygon with a box by finding its extents (minimum and maximum x and y coordinates) and then checks for the intersection for any two boxes in both the X and Y directions.
- If two boxed do not intersect, their corresponding polygons do not overlap (see Figure 1). In such a case, no further testing of the edges of the polygons is required.
- If the minimax test fails (two boxes intersect), the two polygons may or may not overlap, as shown in Figure 1. Each edge of one polygon is compared against all the edges of the other polygon to detect intersections. The minimax test can be applied first to any two edges to speed up this process

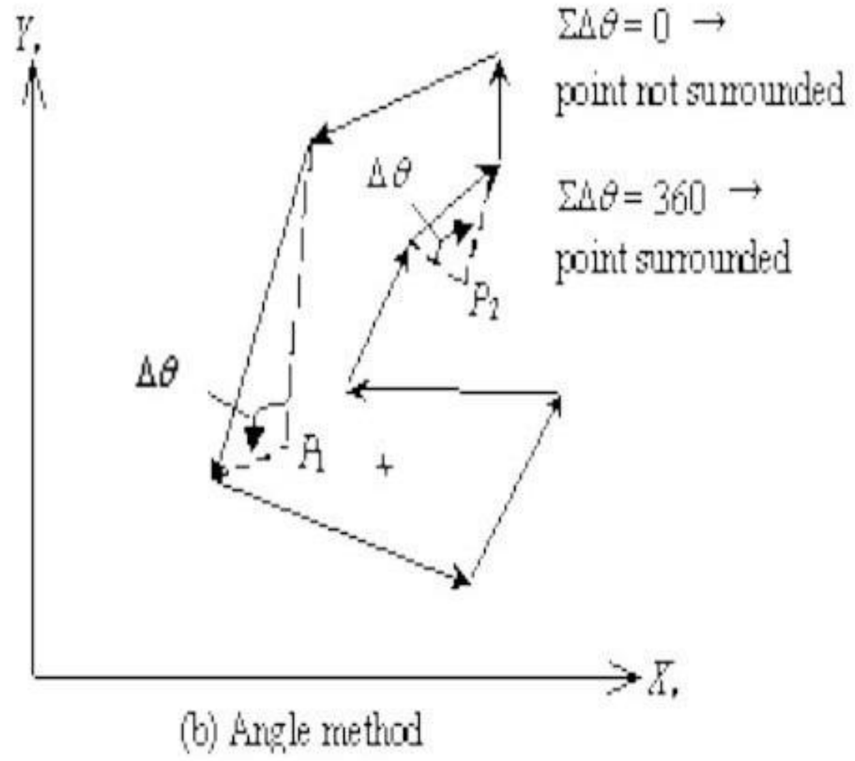
MINIMAX TEST


Y_v
(a) Boxes and polygons do not overlap
(b) Boxes overlap and polygons do not
(c) Boxes and polygons overlap
(d) Minimax test of individual edges
X_v

CONTAINMENT TEST



(a) Intersection method



(b) Angle method

- **The containment test checks whether a given point lies inside a given polygon or polyhedron**. There are three methods to compute containment or surroundness.

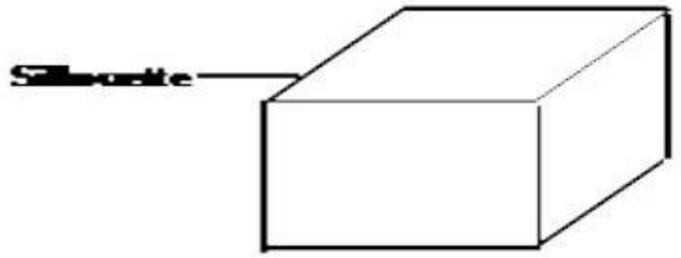
- For a convex polygon, one can substitute the X and Y coordinates of the point into the line equation of each edge. If **all substitutions result in the same sign, the point is on the same side of each edge and is therefore surrounded.**

- For non-convex polygons, two other methods can be used. In the first method, we draw a line from the point under testing to infinity as shown in Figure 2a. The **semi-infinite line** is **intersected with the polygon edges**. If the **intersection count is even**, the point is **outside the polygon** (in Figure 2a). If it is **odd, the point is inside**.
- If one of the polygon edges lies on the semi-infinite line, a singular case arises which needs special treatment to guarantee the consistency of the results.

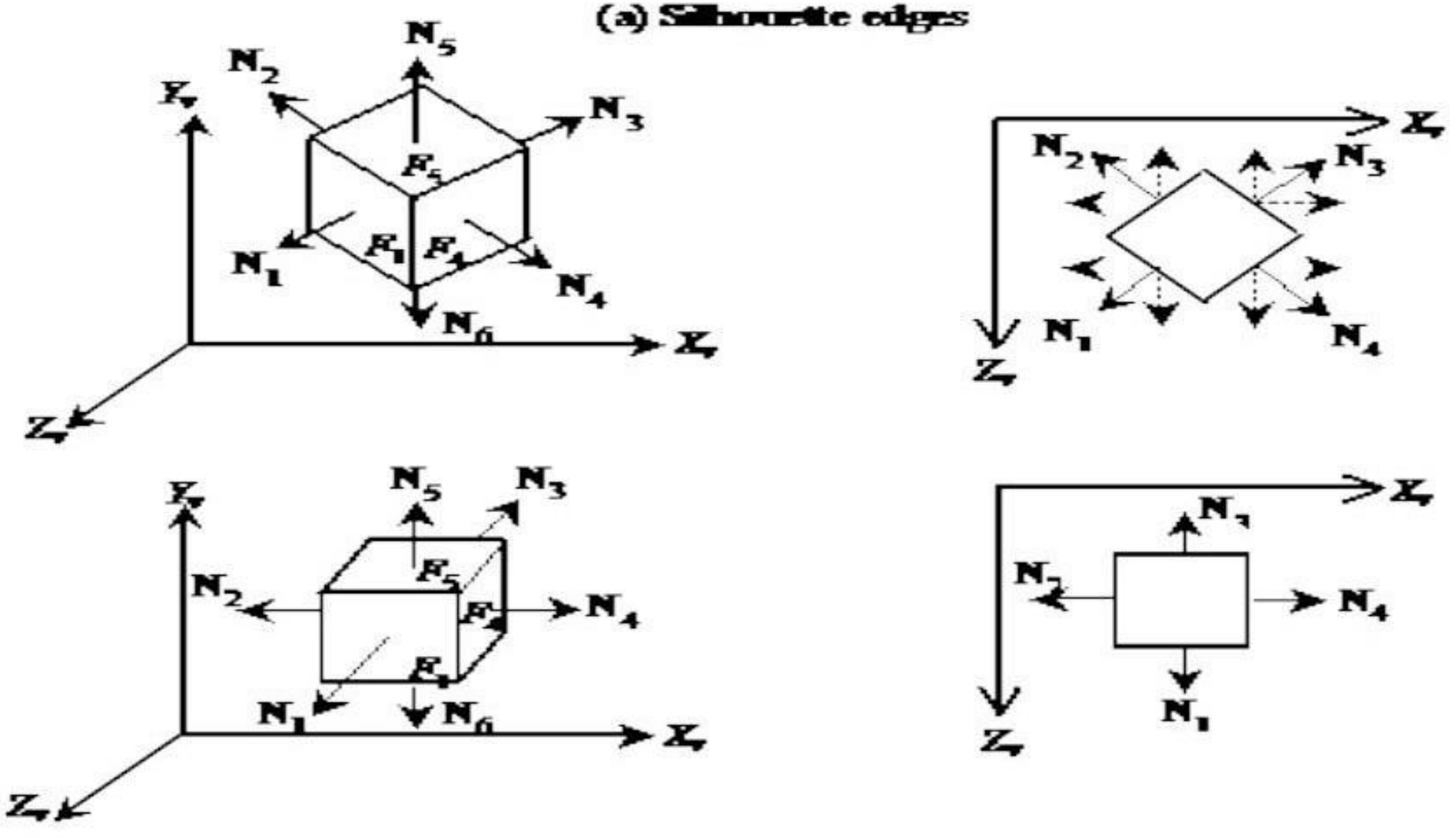
- The second method for non-convex polygons (Figure 2b) **computes the sum of the angles subtended by each of the oriented edges** as seen from the test point. If the **sum is zero**, the point is **outside the polygon**. If the sum is -360 or +360 the point is inside. The minus sign reflects whether the vertices of the polygon are ordered in a clockwise direction instead of counter clockwise.

Computing silhouettes

- A set of edges that separates visible faces from invisible faces of an object with respect to a given viewing direction is called silhouette edges (or silhouettes).
- The signs of the components of normal vectors of the object faces can be utilized to determine the silhouette.
- An edge that is part of the silhouette is characterized as the intersection of one visible face and one invisible face.
- An edge that is the intersection of two visible faces is visible, but does not contribute to the silhouette.
- The intersection of two invisible faces produces an invisible edge.



(a) Silhouette edges

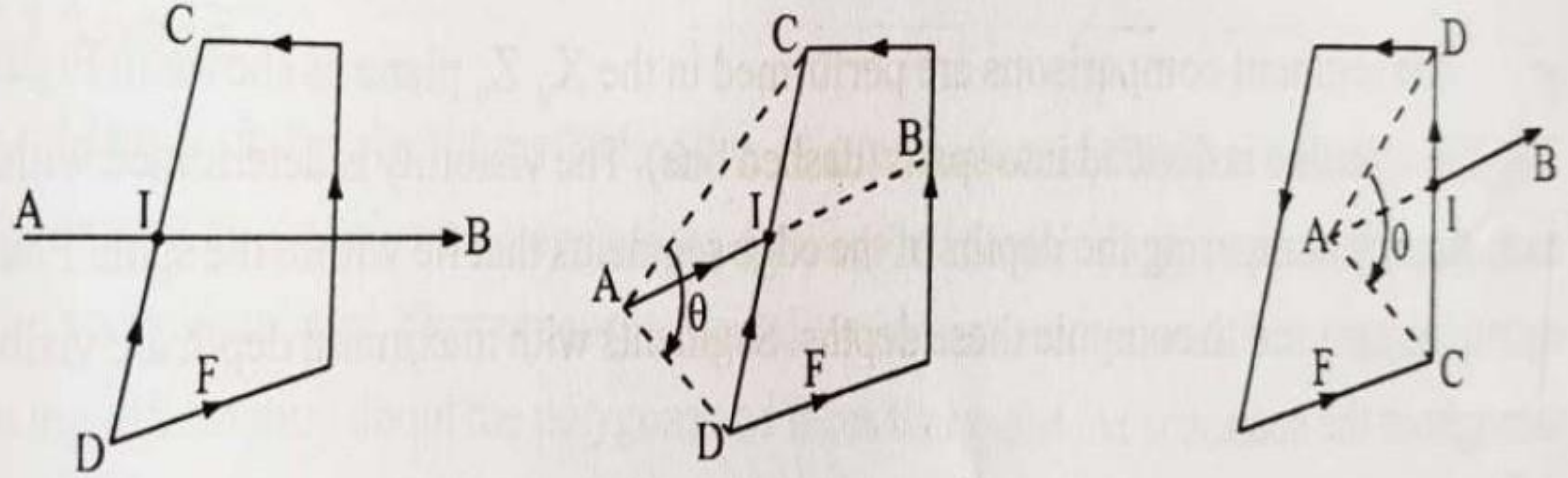


(b) Determining silhouette edges

Edge intersection

- The hidden algorithm initially calculates the edges intersections in 2D.
- To find out partially visible lines.
- The two edges intersect at a point where y2-y1=0.

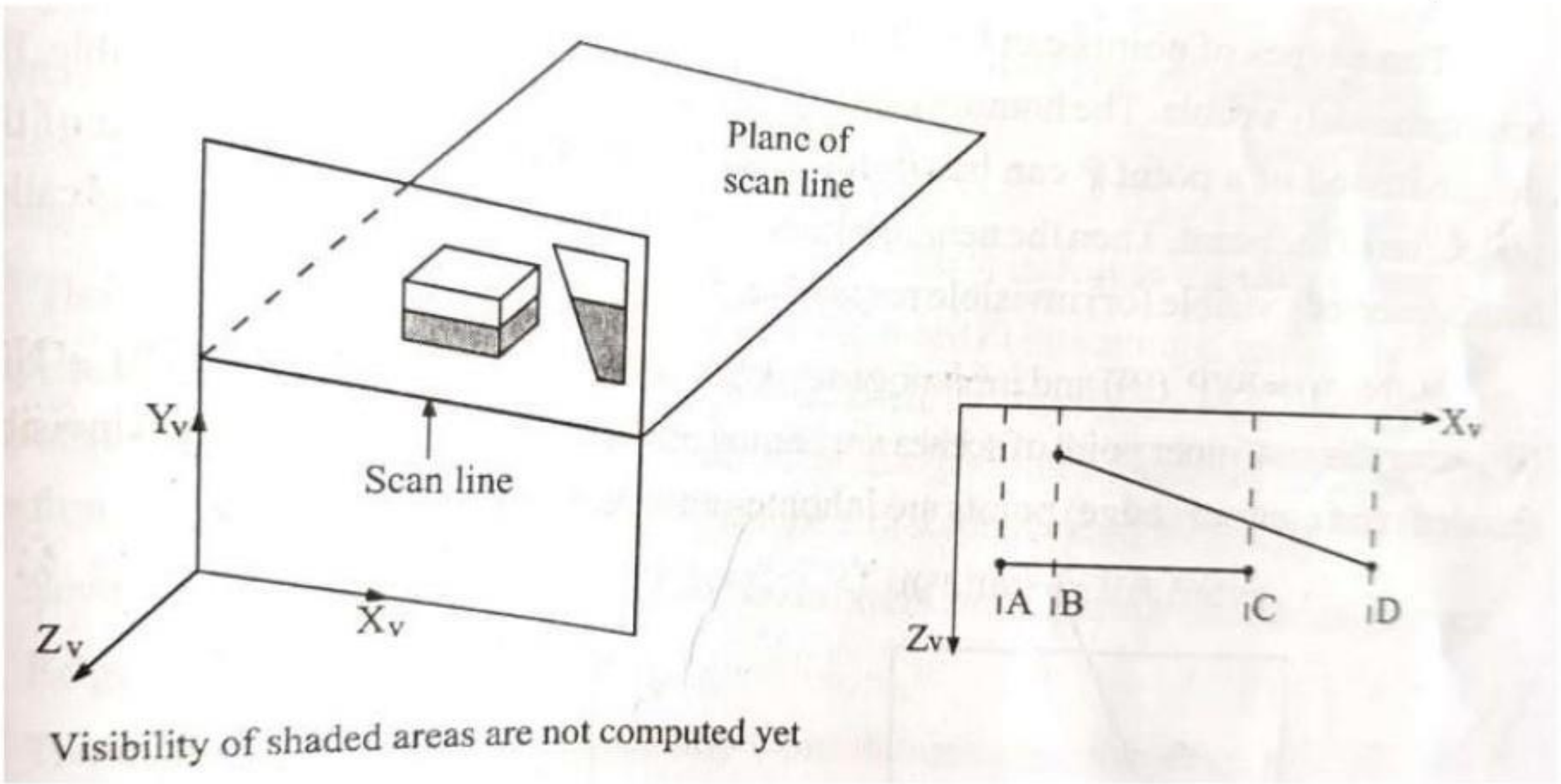
- Then segment comparisons are used to further determine visibility



(a) Fully visible edge AB (b) CD marks the disappearance of partially hidden edge AB (c) CD marks the appearance of partially hidden edge AB

Segment comparison

- The image is computed scan line by line that is in segments and displayed in the same order.
- The scan line is divided into spans(dashed lines).
- The visibility is determined within each span by comparing the depths of the edge segments that lie in the span.
- Segments with maximum depth are visible throughout the span.


Plane of scan line
Yv
Scan line
Zv
Xv
Xv
A
B
C
D
Zv
Visibility of shaded areas are not computed yet

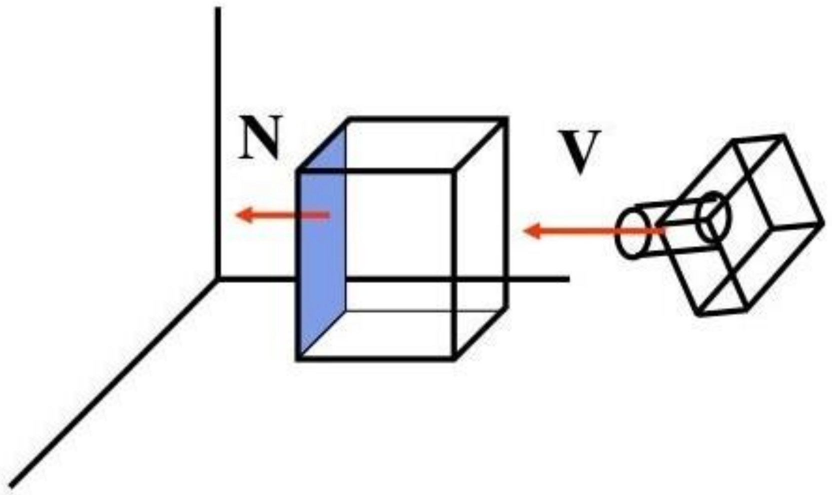
Homogeneity test

- Points are compared for visibility.
- Homogeneously visible
  - Neighborhood of point P can be projected objectively onto neighborhood of the projection of a point.
- Homogeneously invisible
  - Neighborhood of point P cannot be projected objectively onto neighborhood of the projection of a point.
- In-homogeneously visible
  - $P_r(N(P))=N(P_r(P))$
  - $P_r(N(P))\neq N(P_r(P))$ inhomogeneously invisible

Surface Back-face elimination

We cannot see the back-face of solid objects:

Hence, these can be ignored

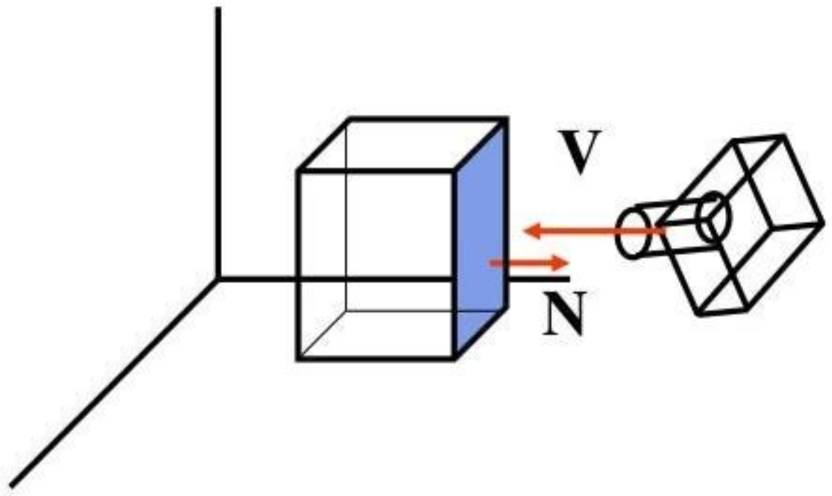


$$\mathbf{V} \cdot \mathbf{N} > 0 : \text{back face}$$

Back-face elimination

We cannot see the back-face of solid objects:

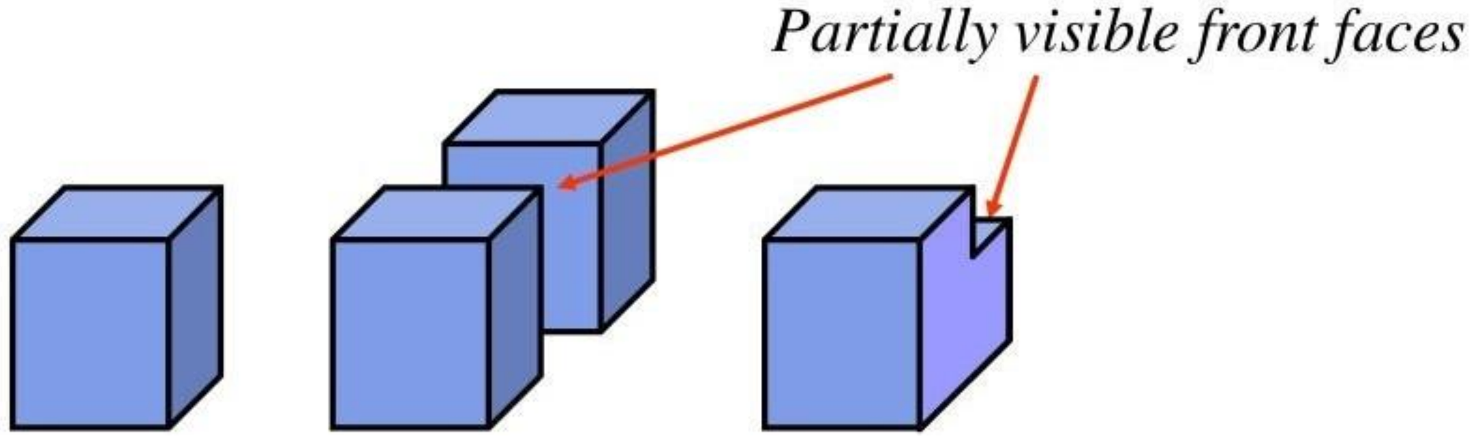
Hence, these can be ignored



$$\mathbf{V} \cdot \mathbf{N} < 0 : \text{front face}$$

Back-face elimination

- Object-space method
- Works fine for convex polyhedra: ±50% removed
- Concave or overlapping polyhedra: require additional processing
- Interior of objects can not be viewed



- **Hidden line removal algorithm**
  - **Depth Algorithm or Z algorithm or Priori algorithm**
  - **Area oriented algorithms**
  - Overlay algorithm-Curved surface
  - Roberts algorithm
- **Hidden surface removal algorithm**
  - **Depth buffer algorithm or z-buffer algorithm**
  - **Area coherence algorithm or Warnock's algorithm**
  - **Scan-line algorithm or Watkin's algorithm**
- **Hidden solid removal algorithm**
  - **Ray tracing algorithm**

Depth or priority algorithm

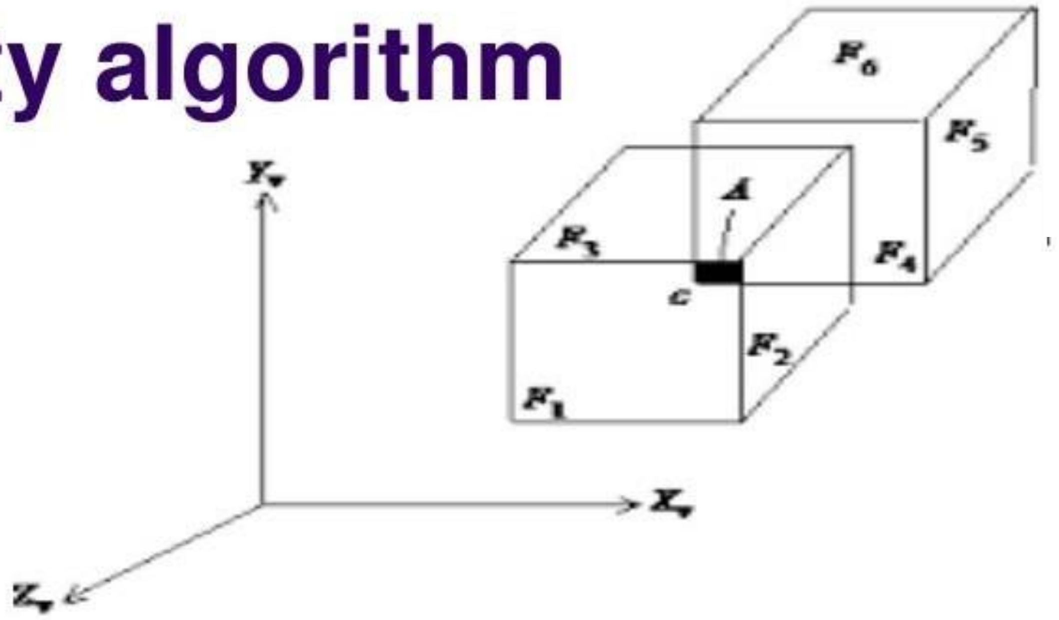
- This algorithm is also known as the depth or z algorithm. The algorithm is based on sorting all the faces (polygons) in the scene according to the largest z coordinate value of each.

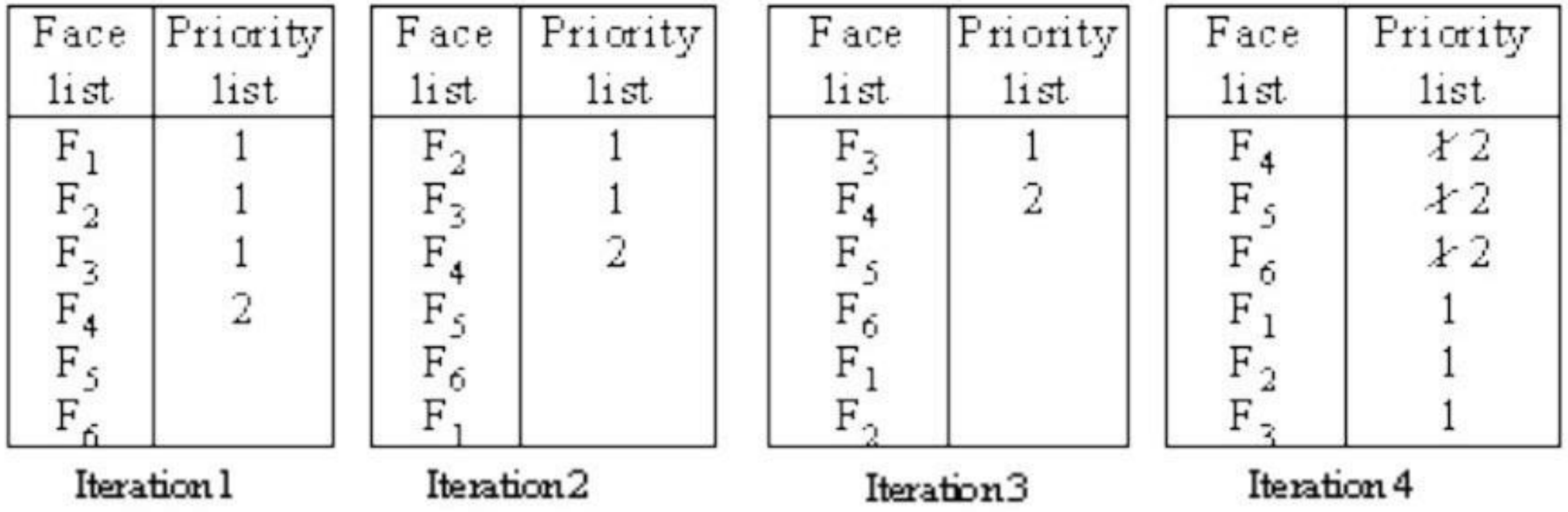
- This step is sometimes known as assignment of priorities. If a face intersects more than one face, other visibility tests besides the z depth are needed to resolve any ambiguities.

- Its basis is on the view according to the biggest Z co-ordinate value.

- If face intersects more than one face, other visibility test beside z-depth is required to solve any issue.

The priority algorithm





Face list	Priority list
F_1	1
F_2	1
F_3	1
F_4	2
F_5	
F_6	

Iteration 1

Face list	Priority list
F_2	1
F_3	1
F_4	2
F_5	
F_6	
F_1	

Iteration 2

Face list	Priority list
F_3	1
F_4	2
F_5	
F_6	
F_1	
F_2	

Iteration 3

Face list	Priority list
F_4	~~1~~ 2
F_5	~~1~~ 2
F_6	~~1~~ 2
F_1	1
F_2	1
F_3	1

Iteration 4

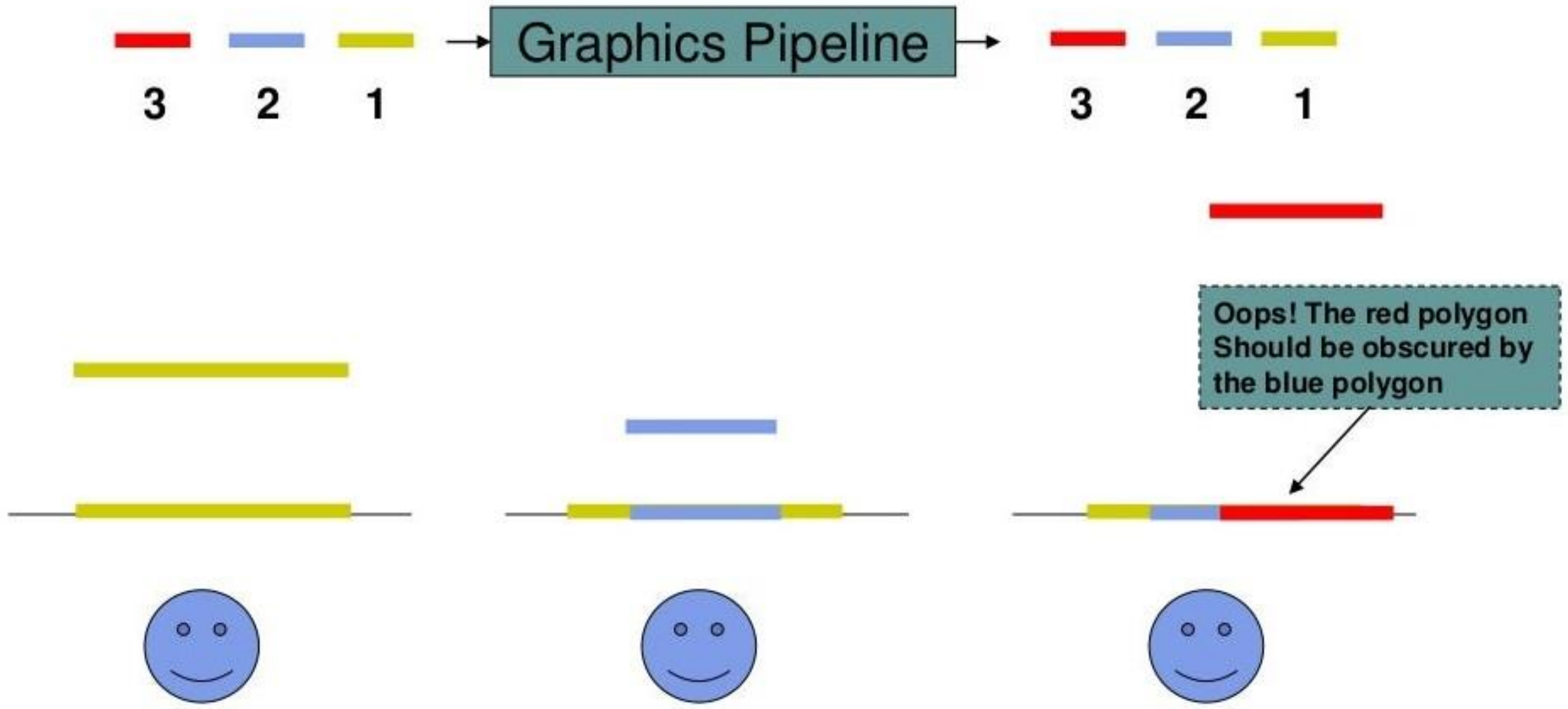
(b) Assignment of priorities

Depth or priority algorithm

- Painter's algorithm
  - As we utilize the procedure the painter's way of creating the background first and then the overlaying layer and then the outermost layer with reducing depth.
- When we view from z and x axis there is no overlapping view.

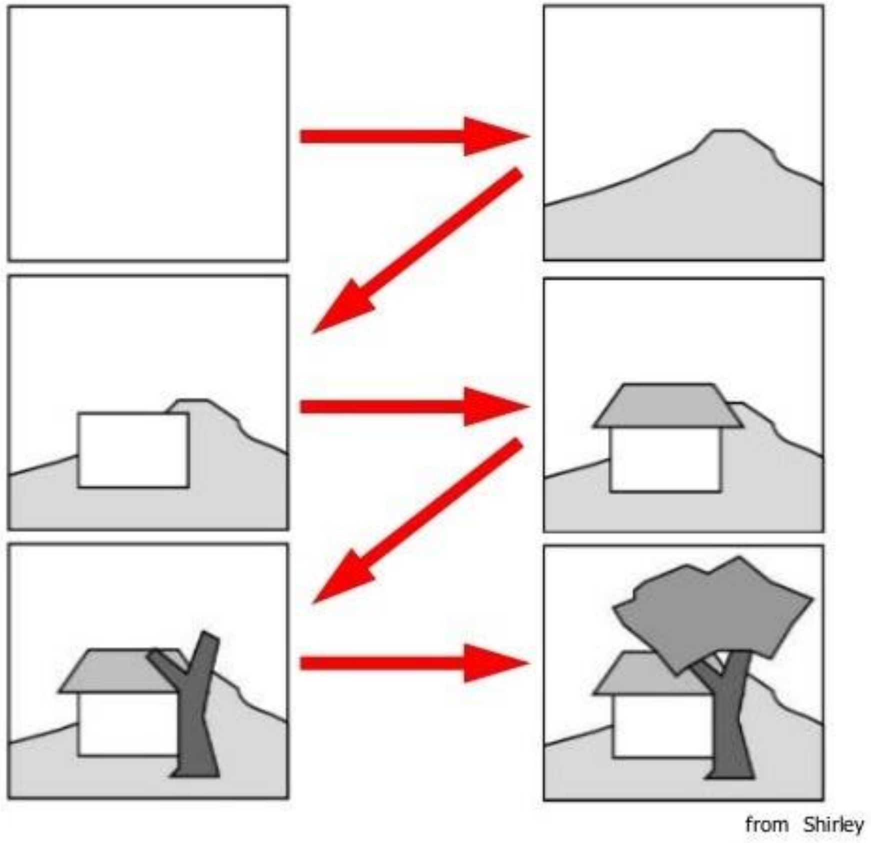
Painter's Algorithm

- Assumption: Later projected polygons overwrite earlier projected polygons


3
2
1
Graphics Pipeline
3
2
1
Oops! The red polygon Should be obscured by the blue polygon

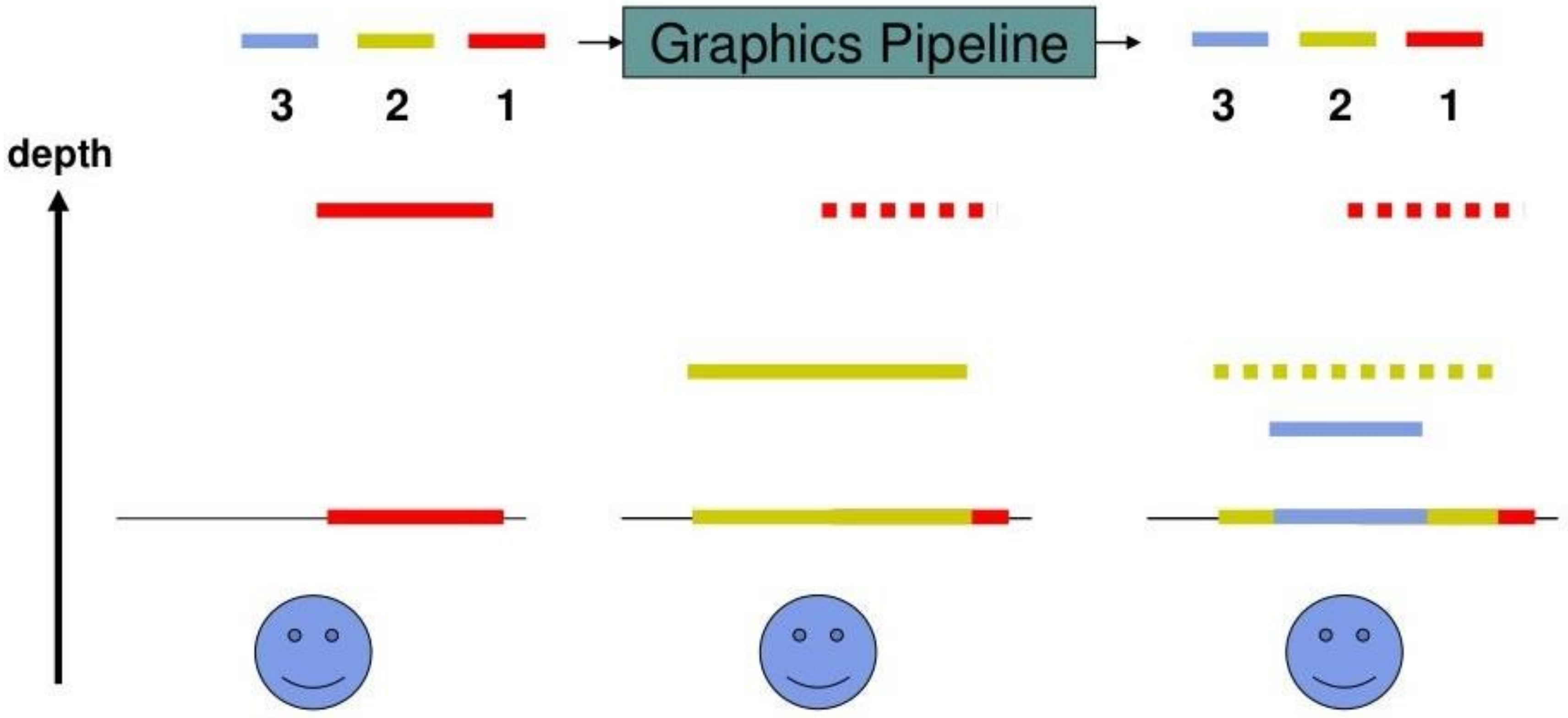
Painter's Algorithm

- Main Idea
  - A painter creates a picture by drawing background scene elemens before foreground ones
- Requirements
  - Draw polygons in back-to-front order
  - Need to **sort** the polygons by depth order to get a correct image



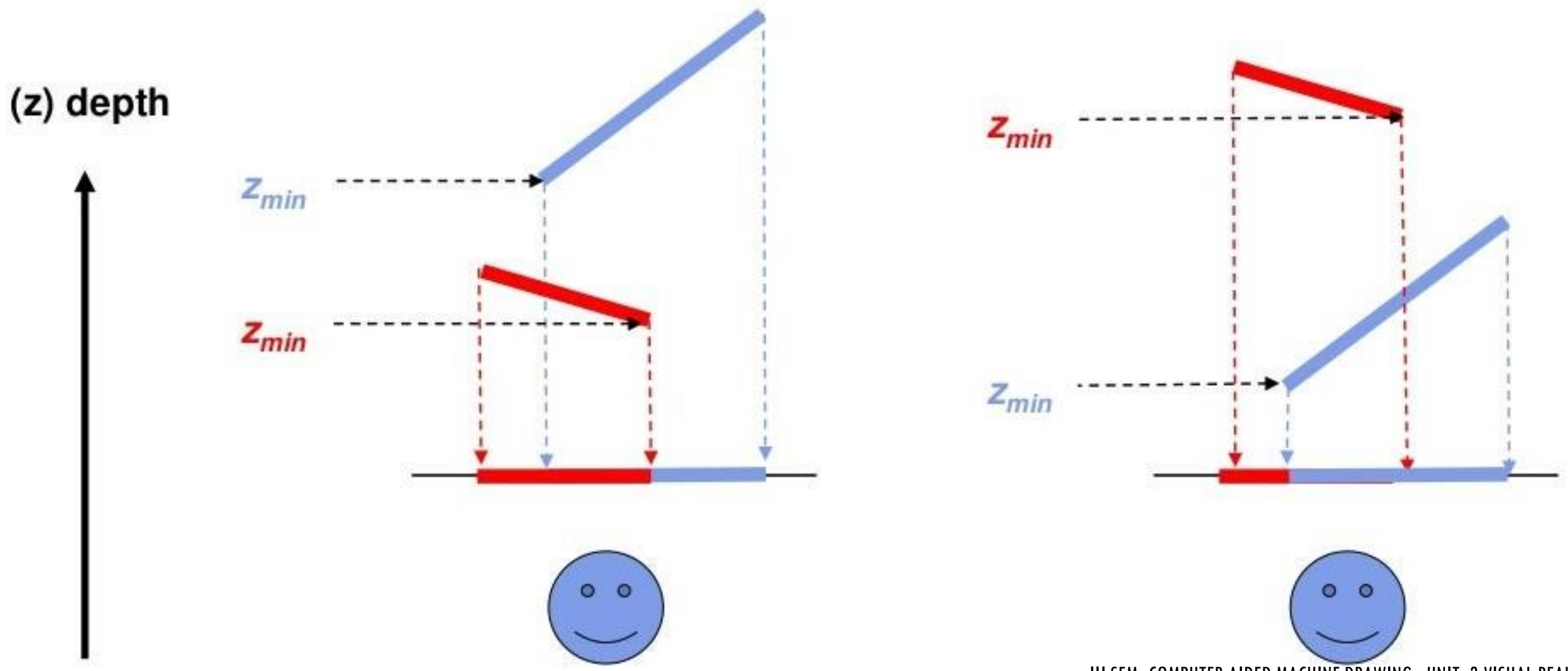
Painter’s Algorithm

- Sort by the depth of each polygon



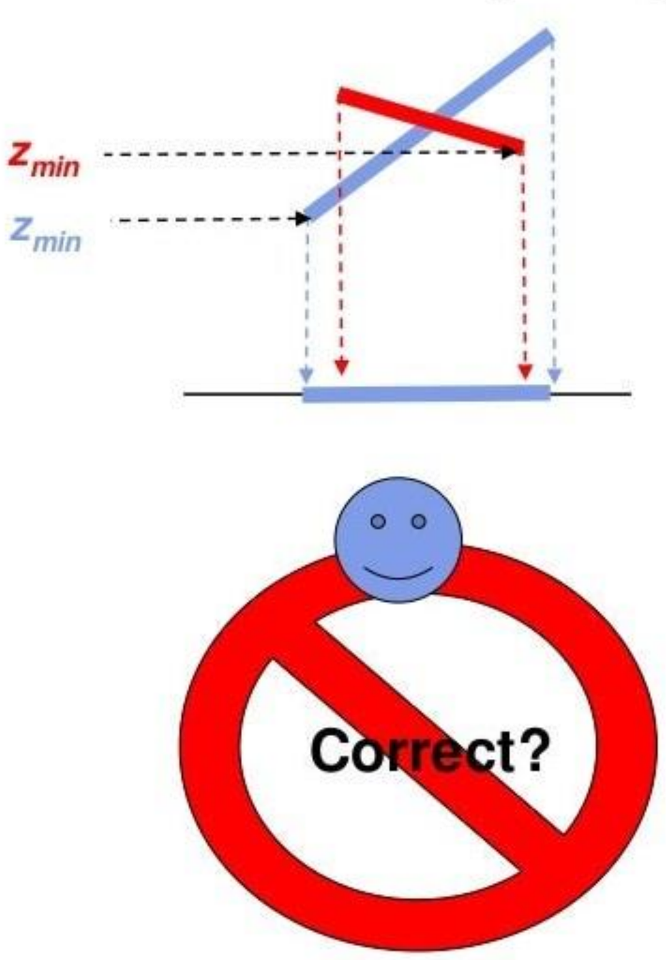
Painter's Algorithm

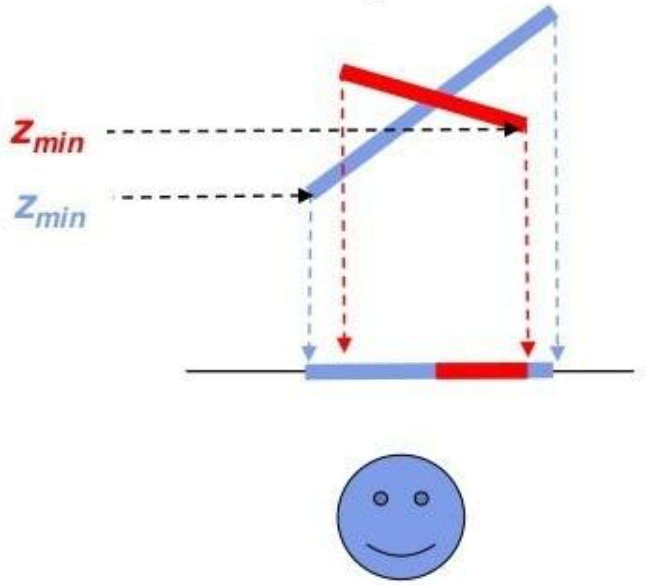
- Compute z_{min} ranges for each polygon
- Project polygons with furthest z_{min} first



Painter's Algorithm

- Problem: Can you get a total sorting?





Area oriented algorithm

It is based on subdivision of given data set in a stepwise fashion until all visible areas are determined and displayed.

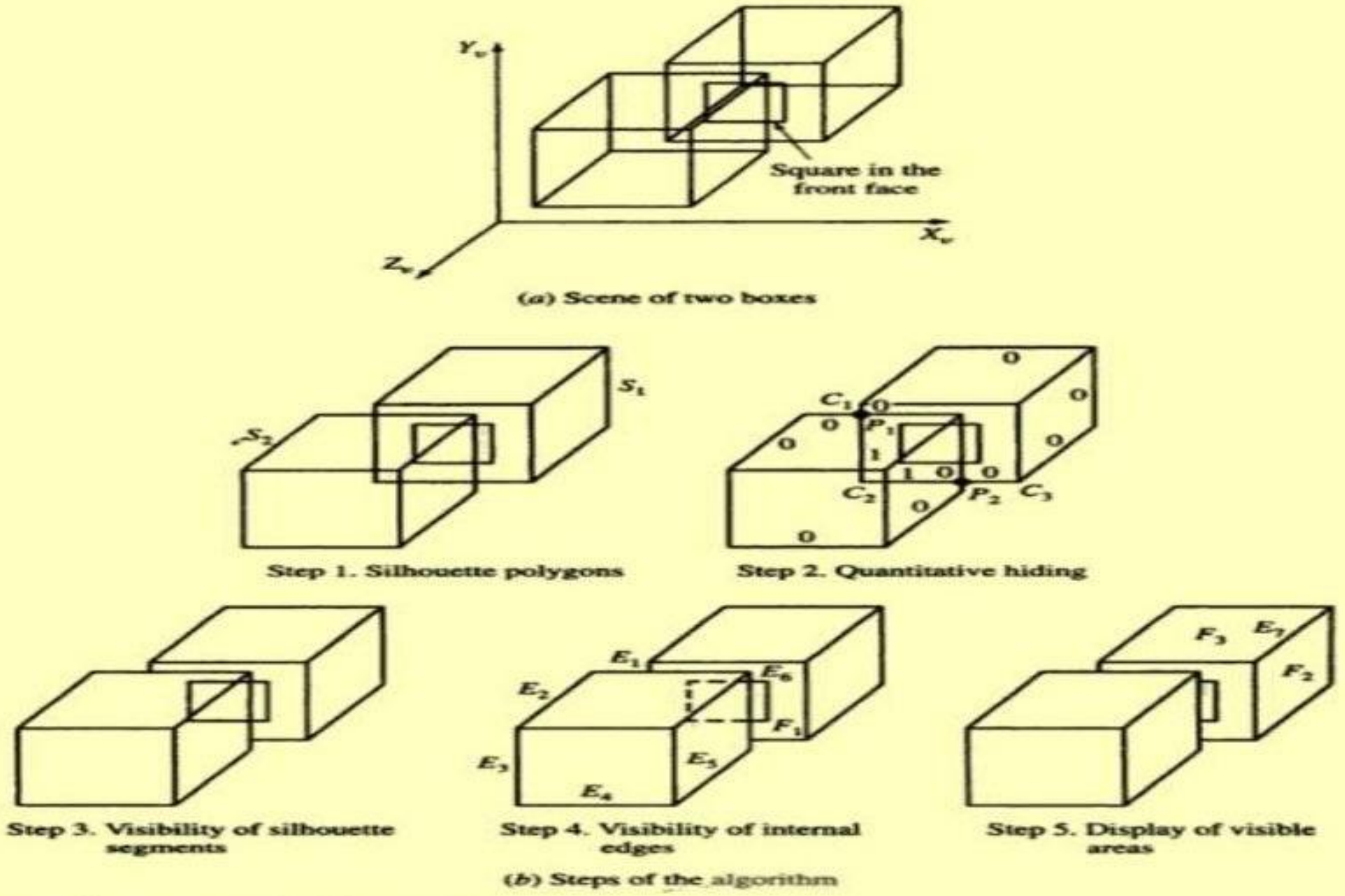

Y_v
X_v
Z_v
Square in the front face
(a) Scene of two boxes
S_1
S_2
Step 1. Silhouette polygons
C_1
C_2
C_3
P_1
P_2
Step 2. Quantitative hiding
Step 3. Visibility of silhouette segments
E_1
E_2
E_3
E_4
E_5
E_6
F_1
Step 4. Visibility of internal edges
F_3
E_7
F_2
Step 5. Display of visible areas
(b) Steps of the algorithm

Figure 13.14 Area-oriented algorithm.

- **Identify silhouette polygons** – silhouette edges are recognized and connection of silhouette edges to form closed polygons by sorting all edges for equal end points
- **Assign quantitative hiding(QH) values to silhouette polygon.**
  - This is achieved by intersecting the polygons. The intersection points define points where QH may change. Find QH value using depth test. 0 is visible and 1 is invisible.
- **Determine the visible silhouette segments.**
  - If closed silhouette polygon is completely invisible it need not be considered any further. If it is visible the segments with least QH values are considered.

- **Intersect the visible silhouette segments with partially visible faces.**
  - To find out the partially visible and fully visible faces.
- **Display the interior of the visible or partially visible polygons.**
  - By using stack and simply enumerates all faces lying inside a silhouette polygon.
  - The stack is started with a visible face and a loop begins popping of face F2

Overlay algorithm

- The curved surfaces are approximated as planar surfaces.
- The u-v grid is used to create grid surface which consists of regions having straight edges.
- The curves in each region are approximated as line segment.
- The 1st step is to use surface equation and grid linear edges are created.

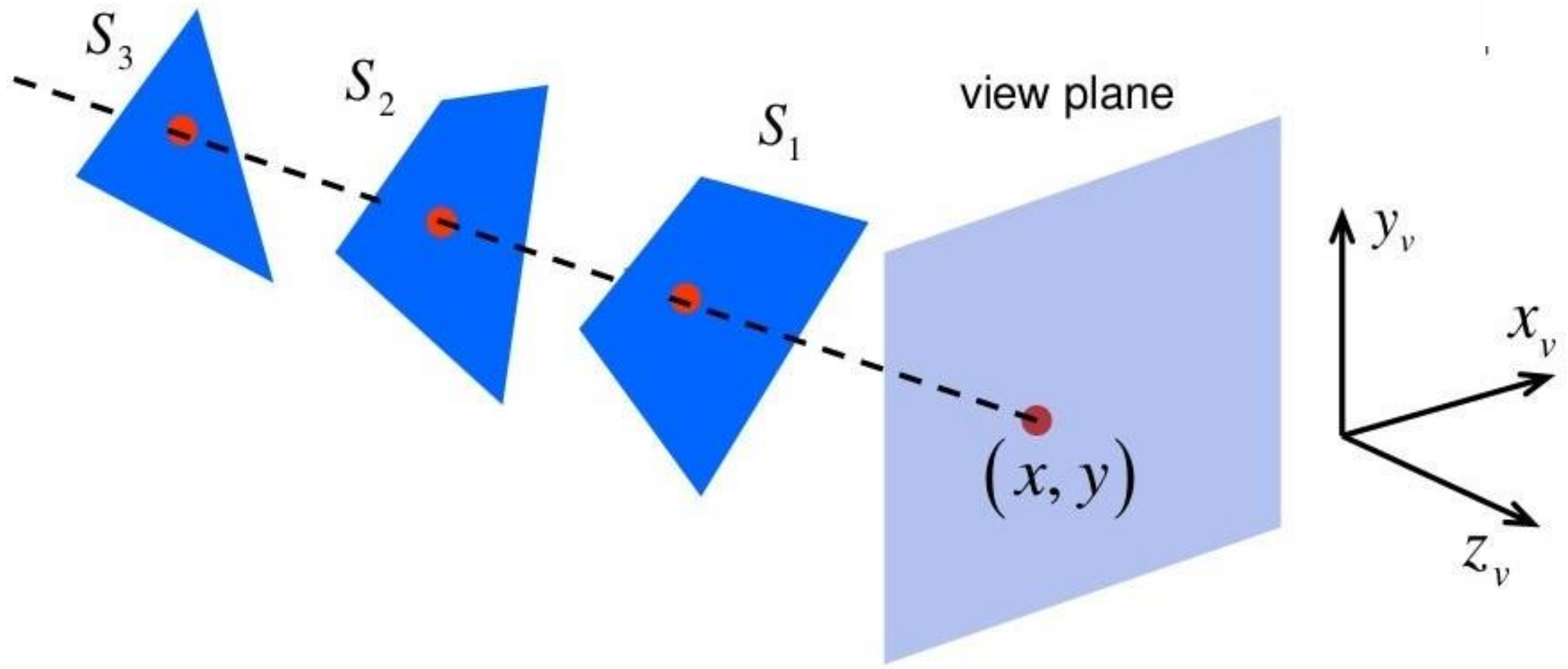
Hidden line removal for curved surface

- To compute the exact visibility we introdu notion of **visibility curves** obtained by projection of silhouette and boundary curves and decomposing the surface into nonoverlapping regions.
- The nonoverlapping and visible portions of the surface are represented as trimmed surfaces and we present a representation based on polygon trapezoidation algorithms.
- The curved surface is converted in polygon mesh and calculated for visiblity.

Hidden surface removal algorithm

- The elimination of parts of a solid objects that are covered by others is called hidden surface removal.
- *Depth buffer or Z-buffer Algorithm*
- *Area coherence or Warnock's algorithm*
- *Scan-line algorithm or Watkin's algorithm*

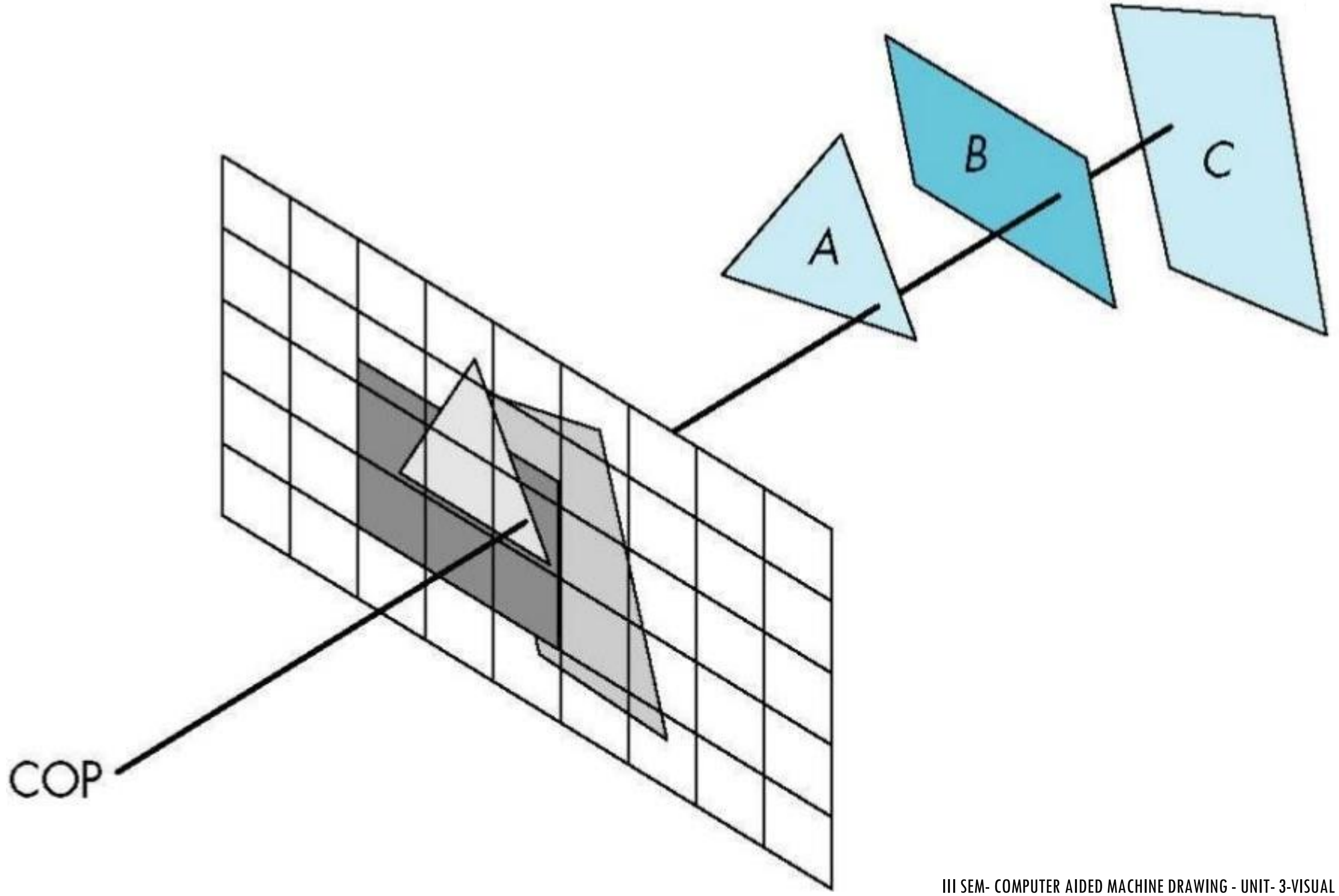
Depth-Buffer Methods



Three surfaces overlapping pixel position *(x,y)* on the view plane.

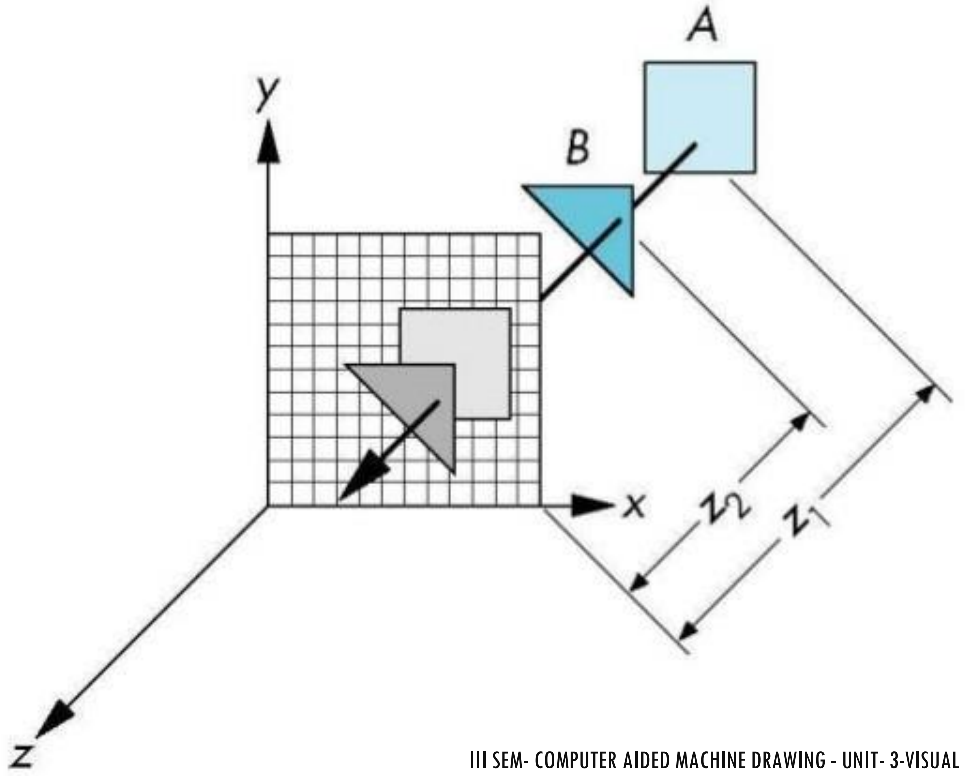
The visible surface, S_1, has the smallest depth value.

Depth buffer or z-buffer algorithm



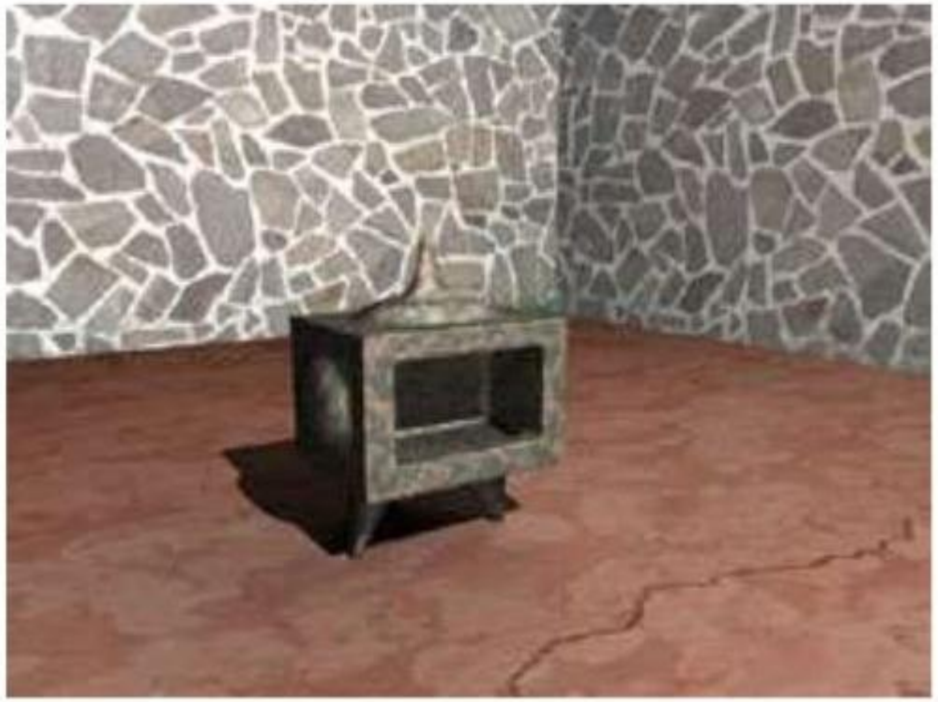
Z-Buffer Algorithm

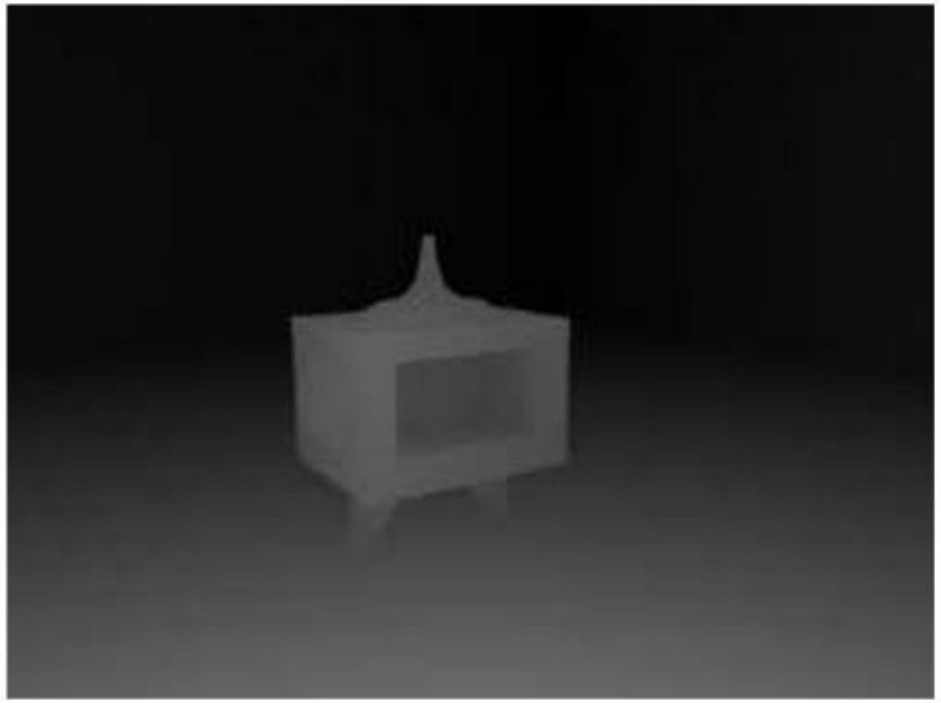
- As we render each polygon, compare the depth of each pixel to depth in z buffer
- If less, place shade of pixel in color buffer and update z buffer



45

Z-buffer: A Secondary Buffer


Color buffer


Depth buffer

- Two buffer areas are required
  - Depth buffer
    - Store depth values for each (x, y) position
    - All positions are initialized to minimum depth
      - Usually 0 – most distant depth from the viewplane
  - Refresh buffer
    - Stores the intensity values for each position
    - All positions are initialized to the background intensity

Z-BUFFER ALGORITHM:

- Its an extension of Frame Buffer
- Display is always stored on Frame Buffer
- Frame Buffer stores information of each and every pixel on the screen
- Bits (0, 1) decide that the pixel will be ON or OFF

- Z- Buffer apart from Frame buffer stores the depth of pixel
- After analyzing the data of the overlapping polygons, pixel closer to the eye will be updated
- Resolution of X,Y => Array[X,Y]

Given set of polygon in image space
Z-Buffer Algorithm:

1. Set the frame buffer & Z-Buffer to a background value

   (Z-BUFFER=Z_{min}) where, Z_{min} is value

To display polygon decide=>color, intensity and depth

2. Scan convert each polygon

i.e, for each pixel, find depth at that point

If Z(X,Y)>Z-BUFFER(X,Y)

Update Z-BUFFER(X,Y)=Z(X,Y)

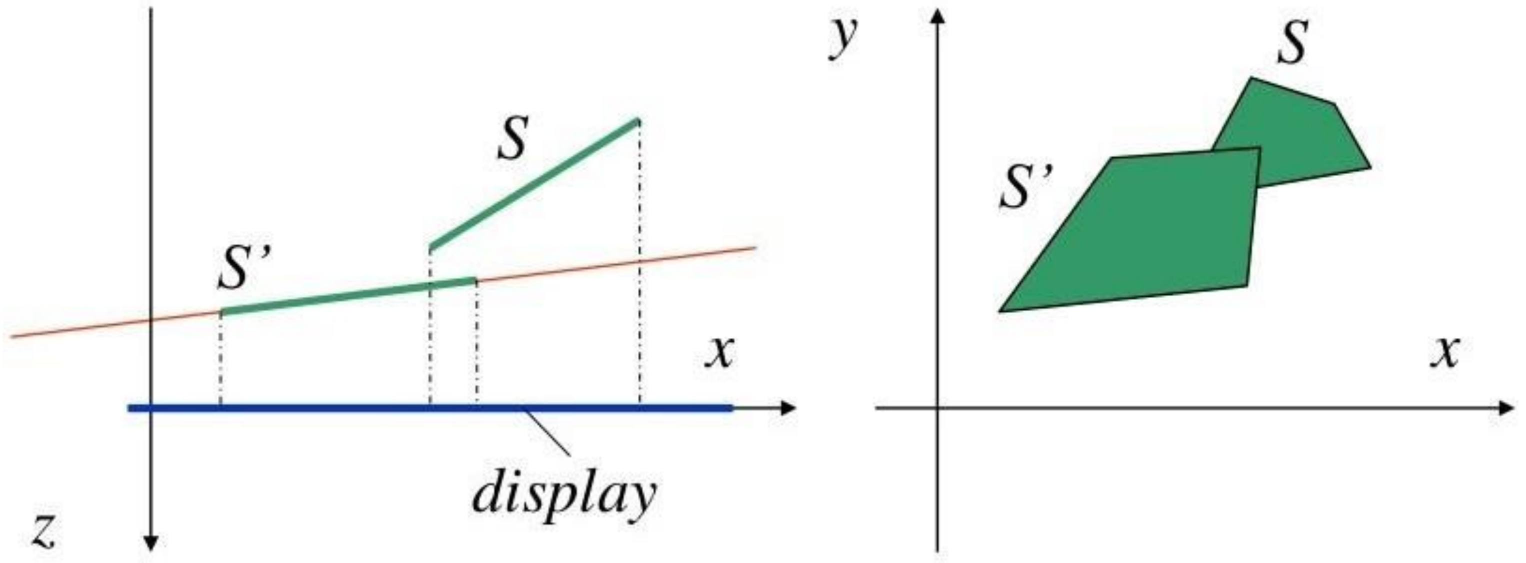
& FRAME BUFFER

This process will repeat for each pixel

- By this way we can remove hidden lines and disp those polygons which are closer to eye

- X*Y space required to maintain Z-Buffer=> X*Y times will be scanned

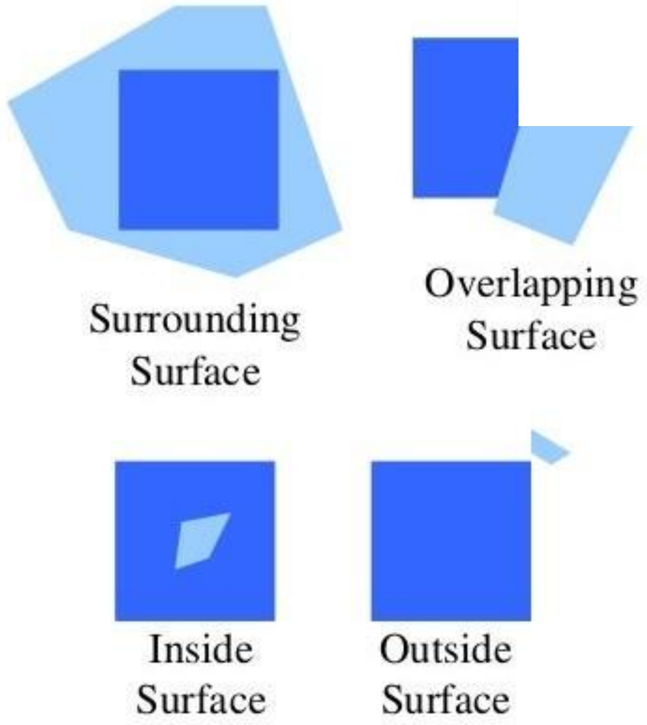
- Expensive in terms of time and space as space is very large



Area coherence or warnock's algorithm
Area Subdivision

- Exploits *area coherence*: Small areas of an image are likely to be covered by only one polygon
- Three easy cases for determining what's in front in a given region:
  1. a polygon is completely in front of everything else in that region
  2. no surfaces project to the region
  3. only one surface is completely inside the region, overlaps the region, or surrounds the region

Identifying Tests



- Four possible relationships
  - Surrounding surface
    - Completely enclose the area
  - Overlapping surface
    - Partly inside and partly outside the area
  - Inside surface
  - Outside surface
- No further subdivisions are needed if one of the following conditions is true
  - All surface are outside surfaces with respect to the area
  - Only one inside, overlapping, or surrounding surface is in the area
  - A surrounding surface obscures all other surfaces within the area boundaries → from depth sorting, plane equation

Warnock’s Area Subdivision
(Image Precision)

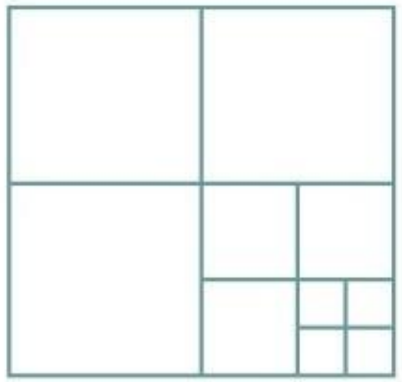
- Start with whole image
- If one of the easy cases is satisfied (previous slide), draw what’s in front
- Otherwise, subdivide the region and recurse
- If region is single pixel, choose surface with smallest depth
- Advantages:
  - No over-rendering
  - Anti-aliases well - just recurse deeper to get sub-pixel information
- Disadvantage:
  - Tests are quite complex and slow

Characteristics

- Takes advantage of area coherence
  - Locating view areas that represent part of a single surface
  - Successively dividing the total viewing area into smaller rectangles
    - Until each small area is the projection of part of a single visible surface or no surface
  - Require tests
    - Identify the area as part of a single surface
    - Tell us that the area is too complex to analyze easily
- Similar to constructing a *quadtree*

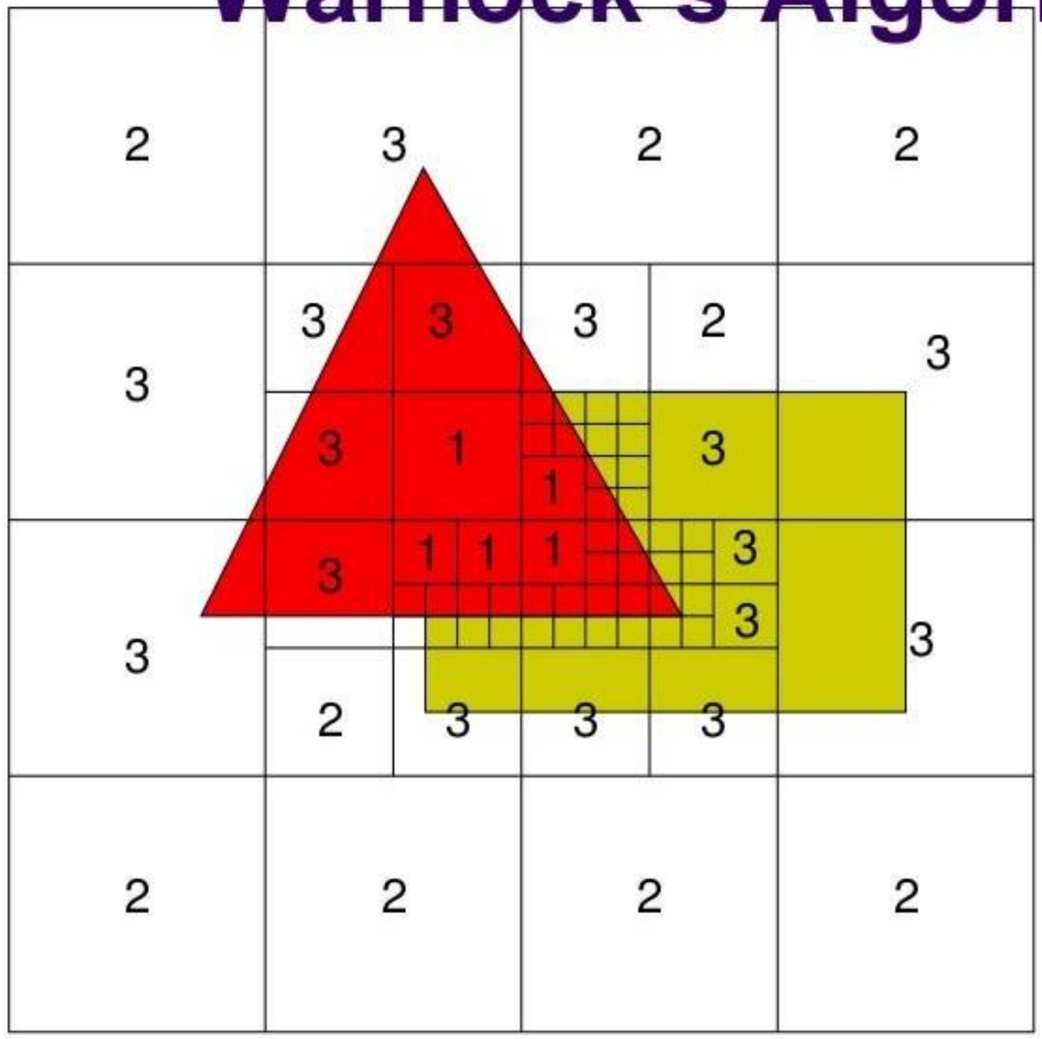
Process

- Staring with the total view
  - Apply the identifying tests
  - If the tests indicate that the view is sufficiently complex
    - Subdivide
  - Apply the tests to each of the smaller areas
    - Until belonging to a single surface
    - Until the size of a single pixel
- Example
  - With a resolution 1024 × 1024
    - 10 times before reduced to a point



55

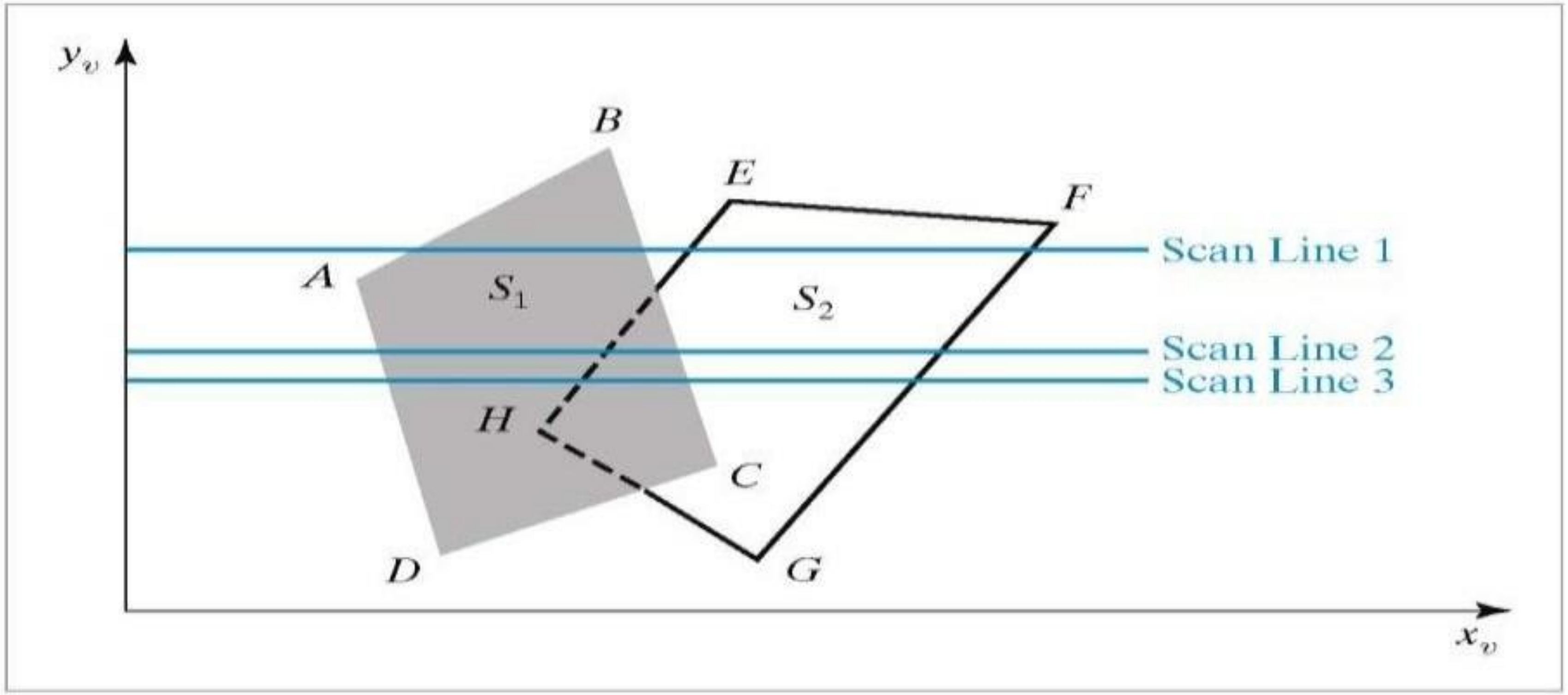
Warnock's Algorithm



- Regions labeled with case used to classify them:
  1) One polygon in front
  2) Empty
  3) One polygon inside, surrounding or intersecting
- Small regions not labeled

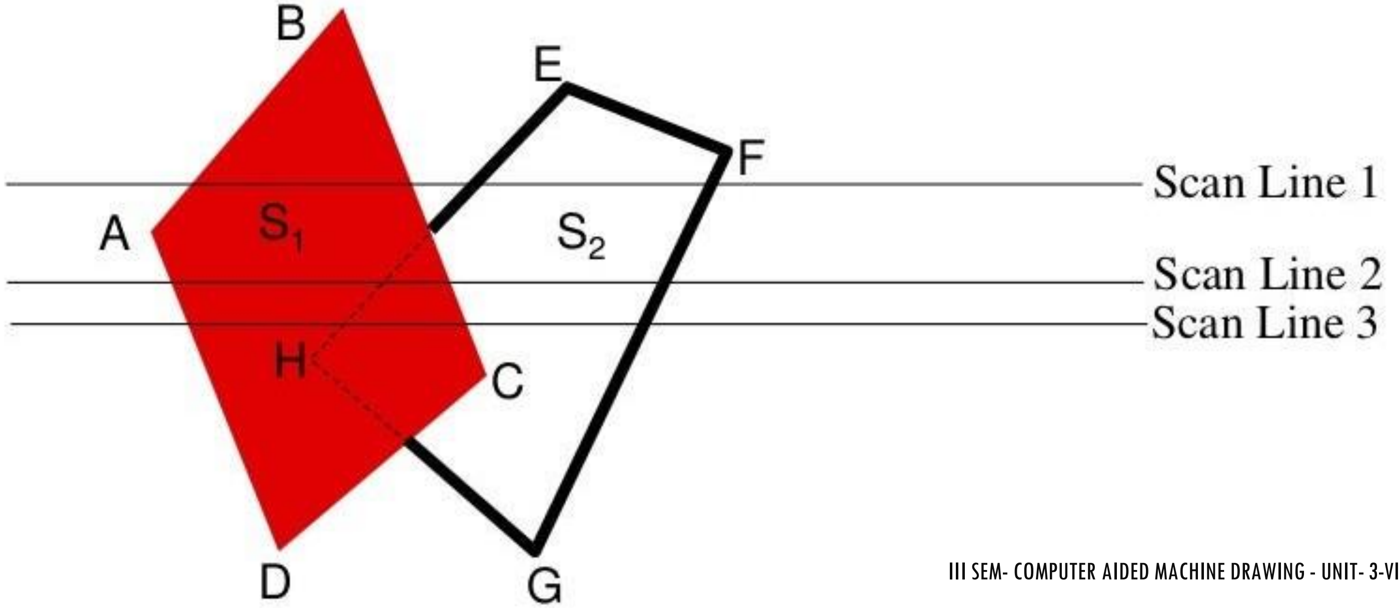
SCAN LINE Z-BUFFER ALGORITHM:

- An image space method for identifying v
  surfaces
- Computes and compares depth values along the various scan-lines for a scene



Scan-Line Method Basic Example

- Scan Line 1:
  - (A,B) to (B,C) only inside S_1, so color from S_1
  - (E,H) to (F,G) only inside S_2, so color from S_2
- Scan Line 2:
  - (A,D) to (E,H) only inside S_1, so color from S_1
  - (E,H) to (B,C) inside S_1 and S_2 , so compute & test depth In this example we color from S_1
  - (B,C) to (F,G) only inside S_2, so color from S_2



- Scanning takes place row by row
- To facilitate the search for surfaces crossing a g scan-line an active list of edges is formed for 'each scan-line as it is processed
- The active list stores only those edges that cross the scan-line in order of increasing x
- Pixel positions across each scan-line are processed from left to right
- We only need to perform depth calculations
- In Scan Line, Z-Buffer(X) whereas earlier it was X*Y

Use Z-Buffer for only 1 scan line / 1 row of pixel

- During scan conversion in the Active List(AEL)=> Calculate Z(X,Y)

i.e, -Pixel info between 1 & 2 active edges will only be stored
-Next pixel will be stored as $z=z_1+\Delta z$
-Δz will be constant but can change anywhere in case of slope

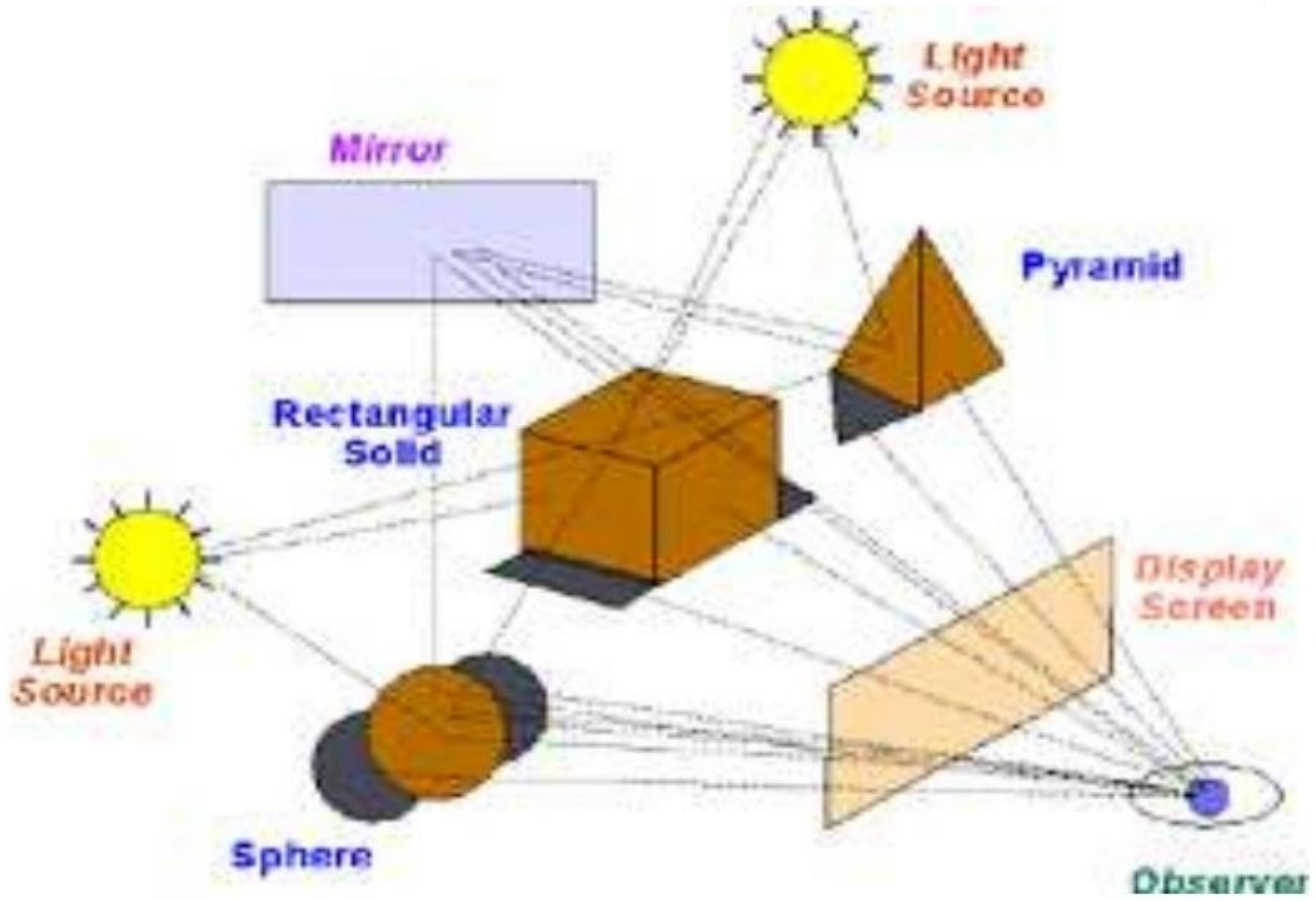
- If Z(X,Y)>Z BUFFER(X) => Update
- If 2 polygons are present-along with Active Edge List, Active Polygon List will also be included
- Active Polygon List=> List of polygons intersecting a scan line

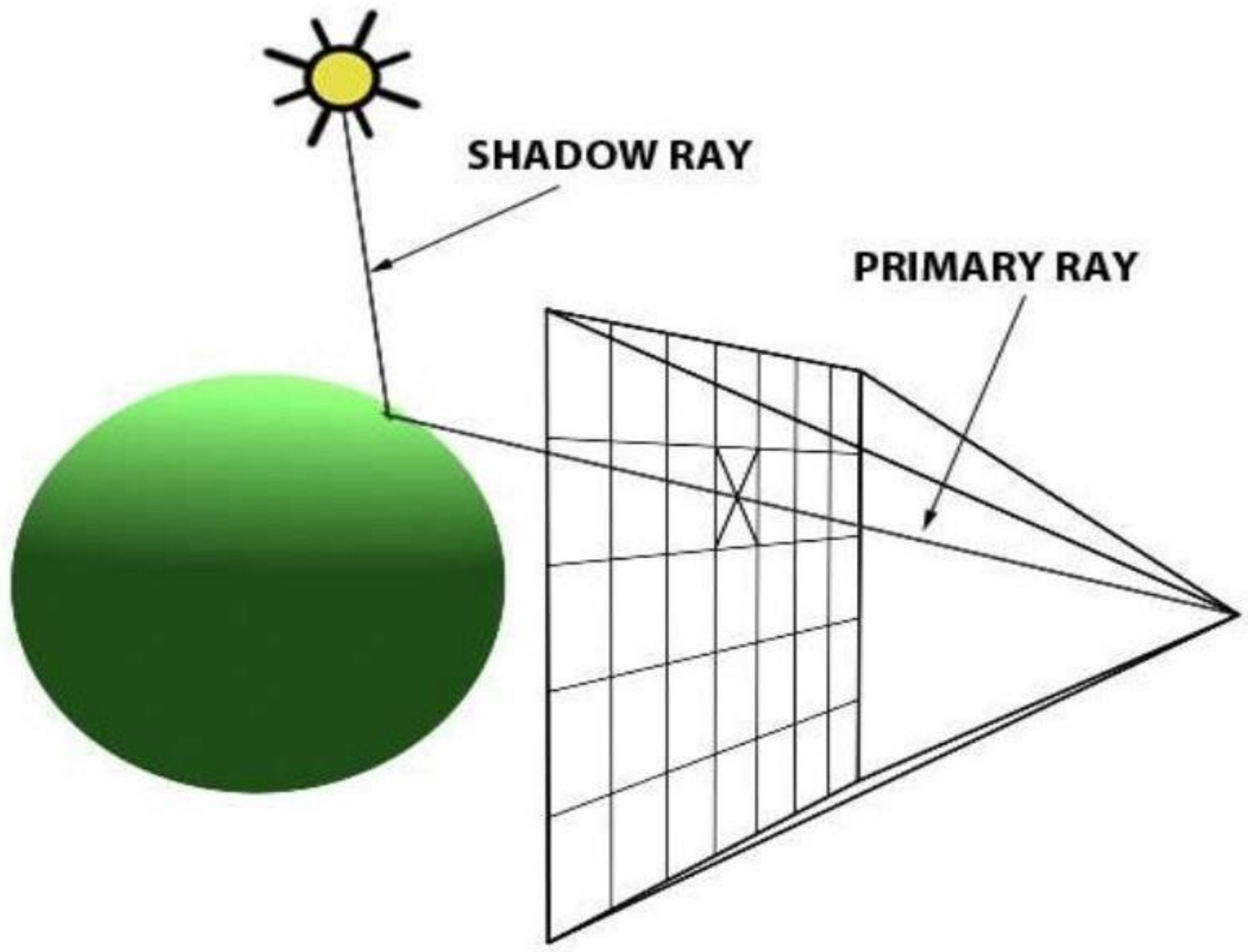
Hidden solid removal

- The hidden solid removal of B-rep model are algorithms such as z-buffer.
- Convert CSG to B-rep.
- Render it with standard hidden surface removal techniques.
- **RAY TRACING**
- The complex 3D solid/solid intersection problem is converted into a 1D ray/solid intersection calculation

Ray tracing

- If we shoot a ray from the viewpoint through pixel, the first object which hits is the one that is visible at the pixel.
- It can be used for both flat and curved surface.
- Shoot the ray from the eyepiece one per pixel
- Find the closest object blocking the path of the ray.
- Since it has infinite rays, the light rays are traced backwards, a ray from the viewpoint is traced through a pixel until it reaches a surface.


Light Source
Mirror
Pyramid
Rectangular Solid
Light Source
Display Screen
Sphere
Observer


SHADOW RAY
PRIMARY RAY

Ray casting

- If resolution is x,y then there are xy pixels so xy light rays are traced.
- Each ray is tested for intersections with each object in the picture including the non clipping plane.
- The intersection closest to the viewpoint is determined since rays intersect many objects.

- The main advantage is that it can create extremely realistic rendering of pictures by incorporating laws of optics for reflection and transmitting light rays
- The major disadvantage is the performance since it starts the process a new and treat each eye ray separately.