VISIBLE-SURFACE DETECTION METHODS

Visible-Surface Detection Methods

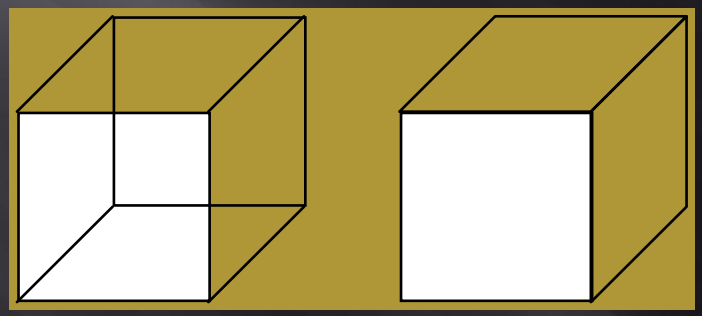
- Determine what is visible within a scene from a chosen viewing position
- Two approaches
  - ***Object-space*** methods: Decide which object, as a whole, is visible
  - ***Image-space*** methods: The visibility is decided point-by-point
- Most visible-surface algorithms use image-space methods
- Sometimes, these methods are referred to as ***hidden-surface elimination***

Approaches

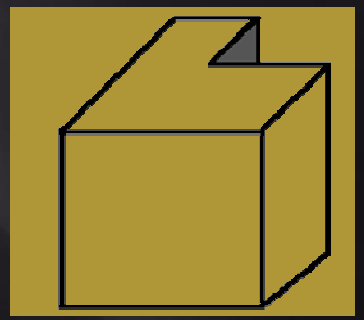
- Back-Face Removal
- Depth Buffer
- A-Buffer
- Scanline
- Depth Sorting
- BSP Tree
- Area Subdivision
- Octree
- Raycasting
- Wireframe Methods

Back-Face Removal (Culling)

- Object Space Method
- Used to remove unseen polygons from convex, closed polyhedron



- Does not completely solve hidden surface problem since one polyhedron may obscure another (concave polygons)

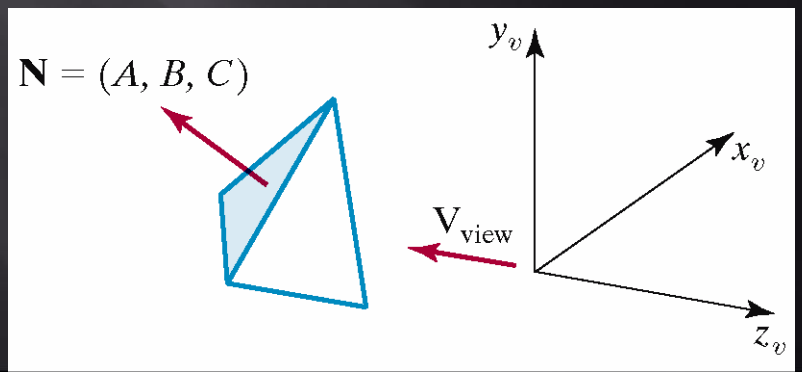


Back-Face Removal (Culling)

- Compute the equation of the plane for each polygon
  - A point (x,y,z) is behind a polygon surface if

$$Ax + By + Cz + D < 0$$

- Determine back-face
  - In projection coordinates, we need to consider only the z component of the normal vector **N**



$$\mathbf{V}_{view} \cdot \mathbf{N} > 0$$

Back-Face Removal (Culling)

- If viewing direction is parallel to z_v axis then,

  V = (0,0,Vz) and

  V.N = Vz.C

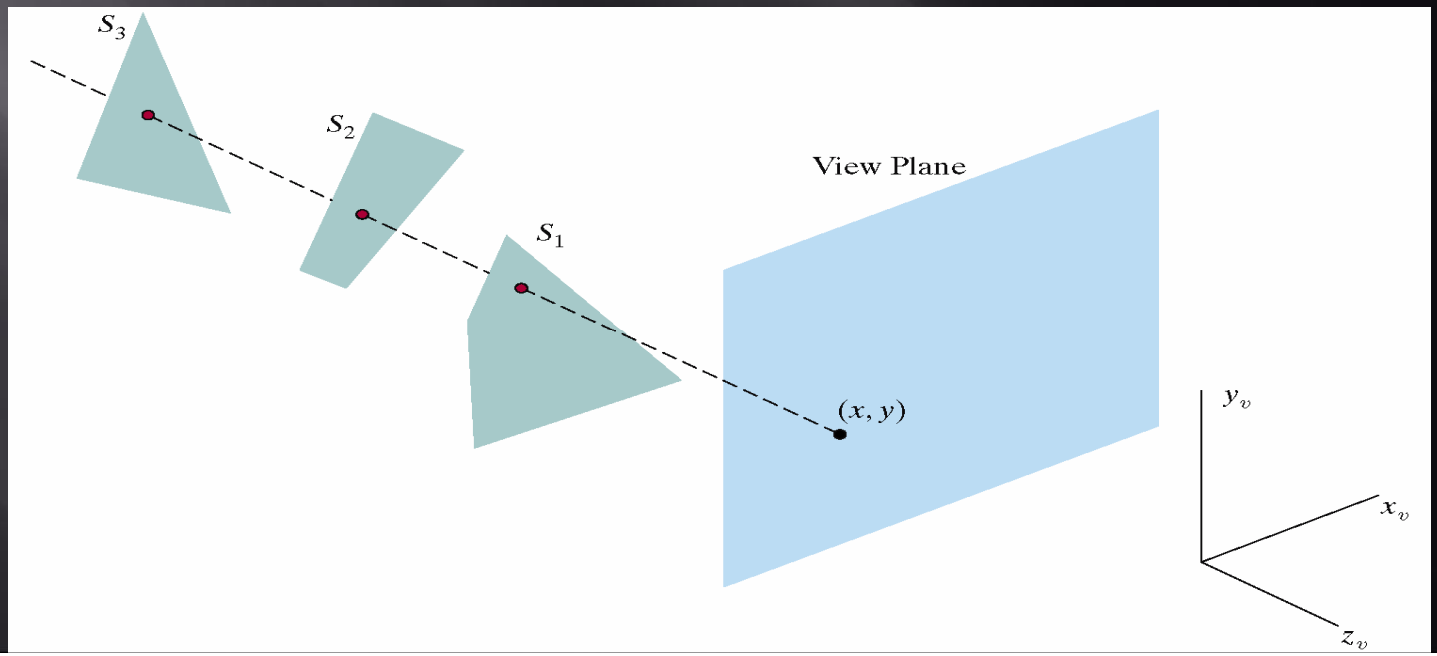
- The sign of C determines the back face.
- Right – handed system, (viewing direction is –Zv axis)

  polygon is back face if C≤0
- Left- handed system, (viewing direction is Zv axis)

  polygon is back face if C≥0

Depth-Buffer (Z-Buffer)

- Image – space method
- Each surface is processed separately, one point at a time across the surface.
- Each point (x,y,z) is projected as (x,y) and z-value is used for depth calculations.



Depth-Buffer (Z-Buffer)

- **Two buffers are needed:**
  - **Depth buffer** – to store the depth of that surface
  - **Refresh buffer –** to store the corresponding intensity values.

1. Initialize the depth buffer and refresh buffer so that for all buffer positions (x, y),

$$\text{depth}(x, y) = 0, \qquad \text{refresh}(x, y) = I_{\text{backgnd}}$$

2. For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.

   - Calculate the depth z for each (x, y) position on the polygon.
   - If $z > \text{depth}(x, y)$, then set

$$\text{depth}(x, y) = z, \qquad \text{refresh}(x, y) = I_{\text{surf}}(x,y)$$

where I_{backgnd} is the value for the background intensity, and $I_{\text{surf}}(x,y)$ is the projected intensity value for the surface at pixel position (x,y). After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces and the refresh buffer contains the corresponding intensity values for those surfaces.

Depth Calculation

- Calculate the z-value on the plane

$$Ax + By + Cz + D = 0 \quad \Rightarrow \quad z = \frac{-Ax - By - D}{C}$$

- Incremental calculation

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

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

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

Depth-Buffer (Z-Buffer)

- Advantages/Disadvantages
  - Very easy to implement
  - No sorting of surfaces is needed
  - Needs lot of storage space
    - Storage can be reduced by processing one section of a scene at a time and reusing the buffers

Accumulation Buffer (A-Buffer)

- An extension of the depth-buffer
- Represents antialiased, area-averaged, accumulation-buffer method
- The depth-buffer method identifies only one visible surface at each pixel position (i.e., deals with only opaque surfaces)
  - Cannot accumulate color values for more than one transparent and translucent surfaces
- Each position in the buffer references a linked list of surfaces

Accumulation Buffer (A-Buffer)

- Each position in the A-buffer has two fields
  - Depth field: Stores a depth value (+ve or –ve real number)
  - Surface data field – surface intensity value or a pointer
    - RGB intensity components
    - Opacity parameter (percent of transparency)
    - Depth
    - Percent of area coverage
    - Surface identifier
    - Pointer to next surface

Accumulation Buffer (A-Buffer)

Background
opaque surface

Foreground
transparent surface

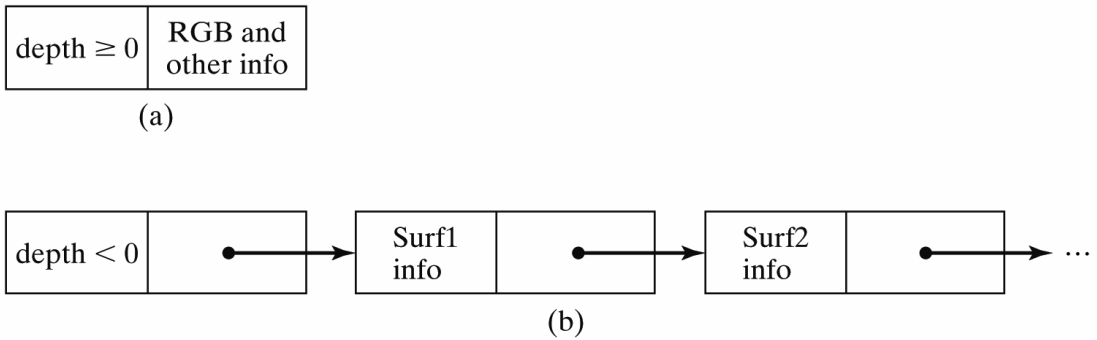


Figure 9-9

Two possible organizations for surface information in an A-buffer representation for a pixel position. When a single surface overlaps the pixel, the surface depth, color, and other information are stored as in (a). When more than one surface overlaps the pixel, a linked list of surface data is stored as in (b).

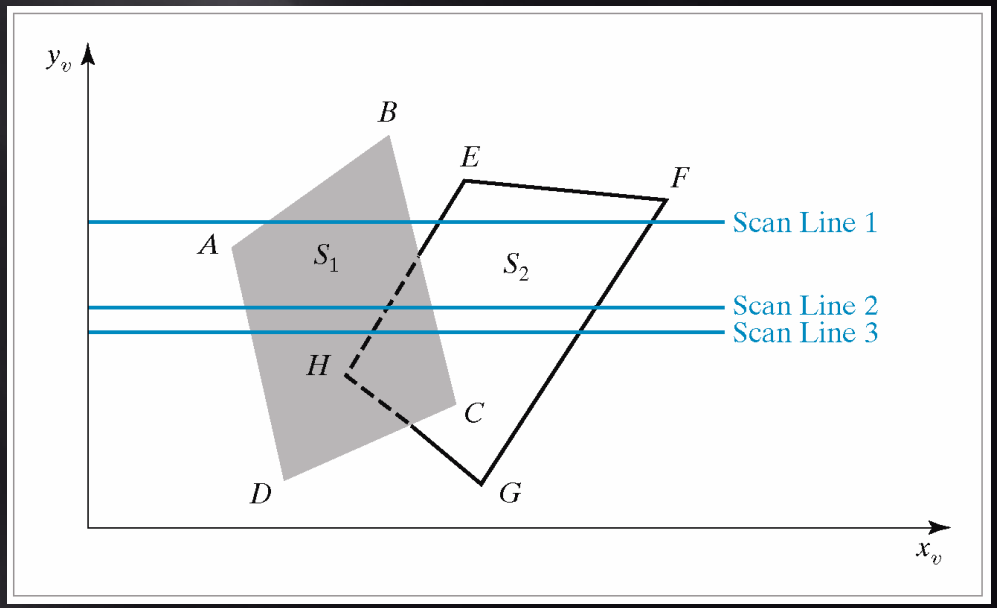
Scan Line Method

- Image-space method
- As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which surfaces are visible.
- Across each scan line, nearest surface is determined using depth calculations
  - Requires a depth buffer equal to only one scan line
  - Requires the entire scene data at the time of scan conversion
- For each scan line maintain
  - an active polygon list
  - active edge list – edges that cross the current scan line
  - Flag for each surface – indicates whether a point along the scan line is inside or outside the surface.

Scan Line Method

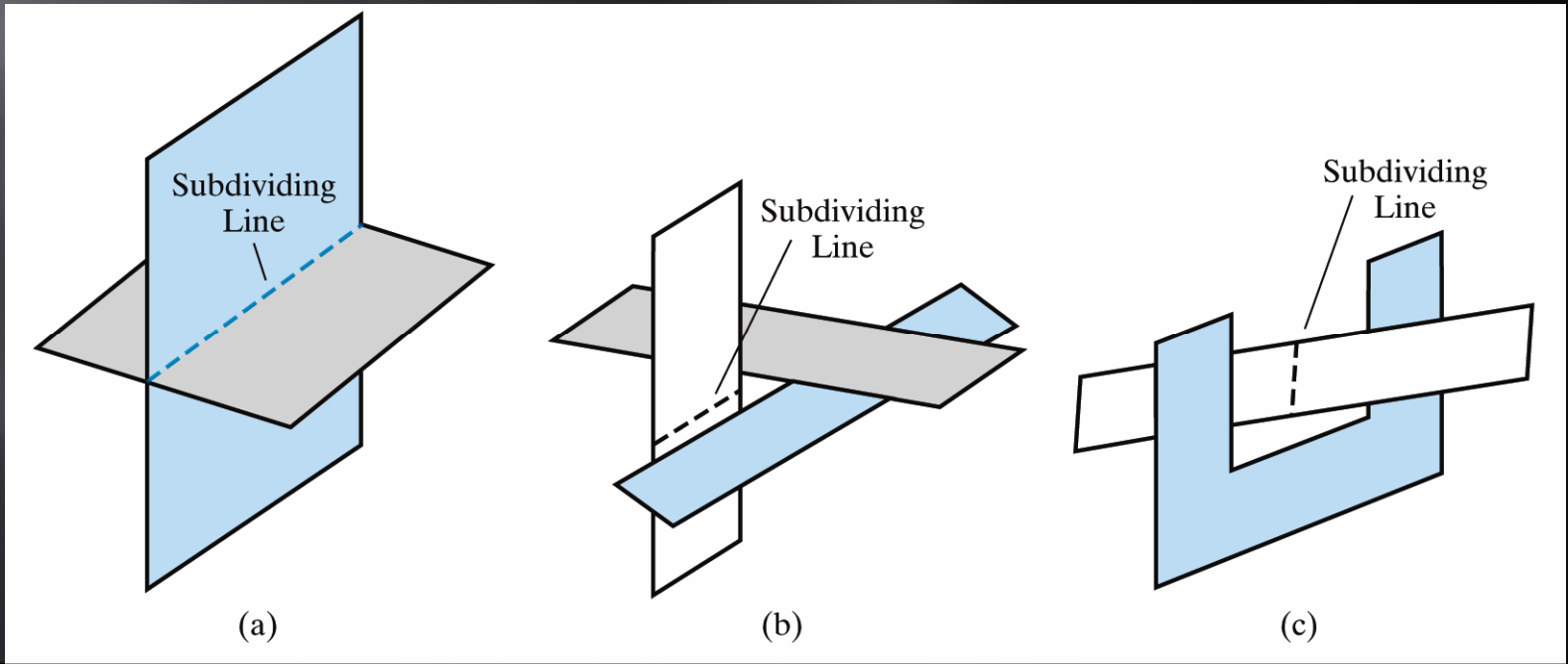
- For Scan line 1,
  - Active edge list – AD,BC,EH and FG
  - Positions between AB and BC – flag for surface S1 is on.
  - Positions between EH and FG – flag for surface S2 is on.
  - All other positions, intensity is set to background intensity values.
- For Scan lines 2
- For Scan line 3 –

Advantage of coherence



Scan Line Method

- Cyclically overlapping surfaces that alternately obscure one another
- We can divide the surfaces to eliminate the cyclic overlaps

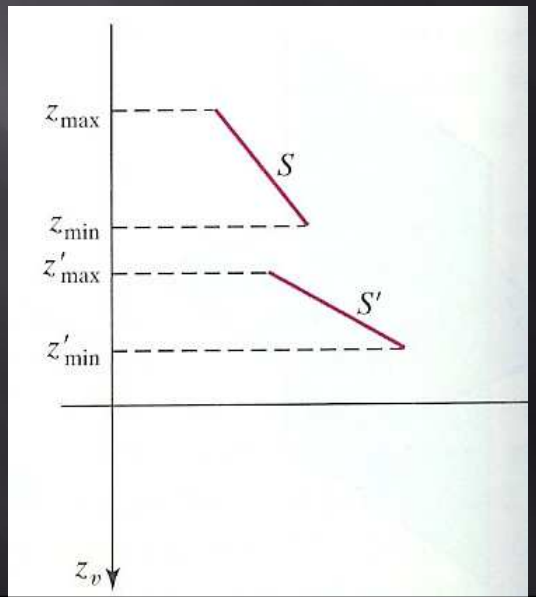


Depth Sorting

- Uses Object-space and image-space methods
- Performs
  1. Surfaces are sorted in order of decreasing depths (image and object space)
  2. Surfaces are scan-converted in order, starting with surface of greatest depth (image space)
- Referred as Painter's algorithm
  - Draw polygons as an oil painter might do
  - background objects are painted first and then foreground objects.

Depth Sorting

- We need a partial ordering (not a total ordering) of polygons
  - The ordering indicates which polygon obscures which polygon
  - Some polygons may not obscure each other
- Simple cases
  - Viewing along – Z axis



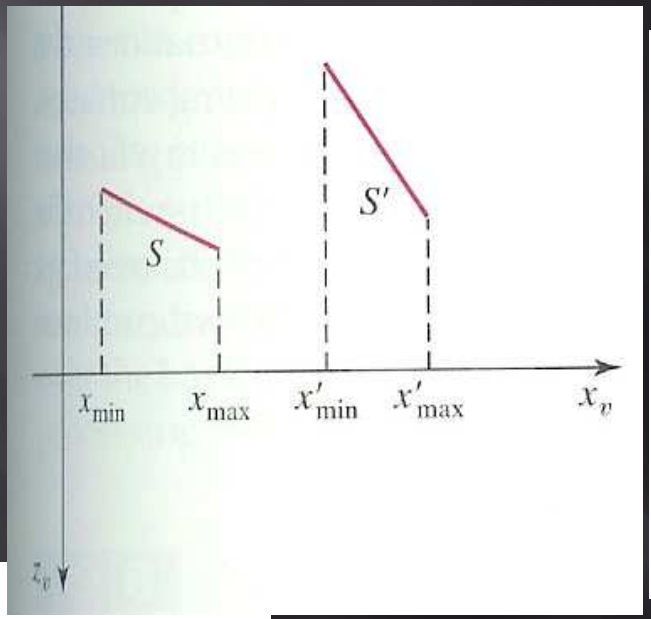
Two surfaces on xy plane with no depth overlap

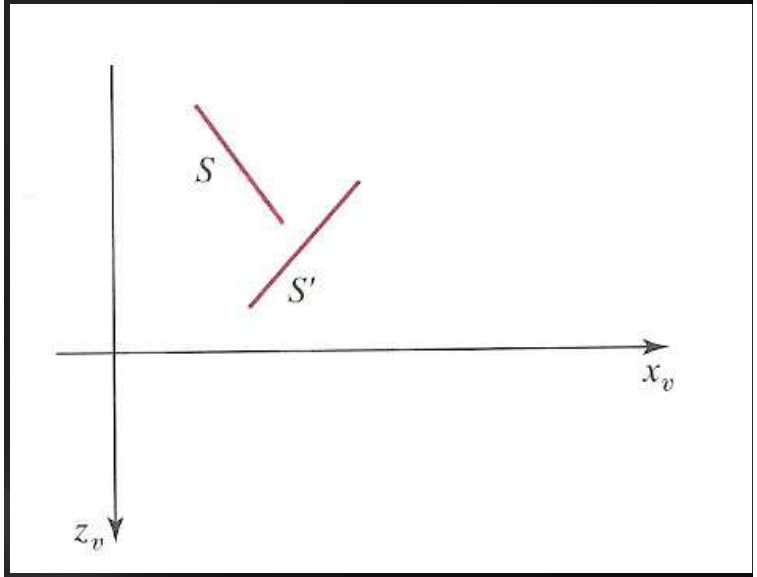
Depth Sorting

- We make the following tests for each polygon that has a depth overlap with S
- If any one of these tests is true, no reordering is necessary for S and the polygon being tested
  - The bounding rectangles in the xy plane for the two surfaces do not overlap.
  - Surface S is completely behind the overlapping surface relative to the viewing position
  - The overlapping surface is completely in front of S relative to the viewing position
  - The projections of the two surfaces onto the view plane do not overlap

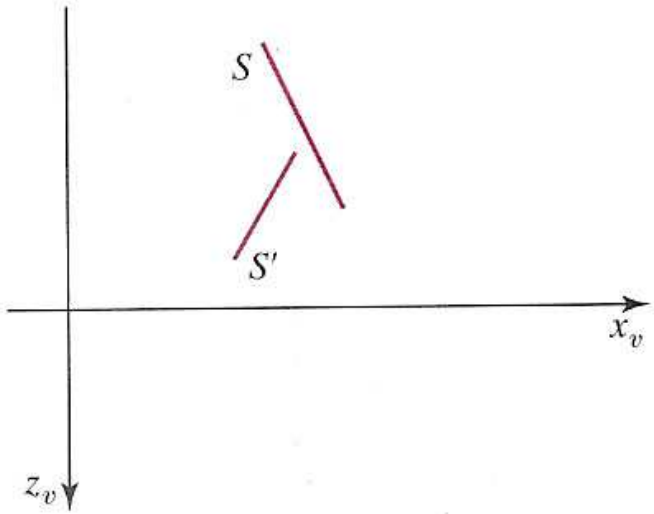
Depth Sorting

Two surfaces with depth overlap but no overlap in the x direction





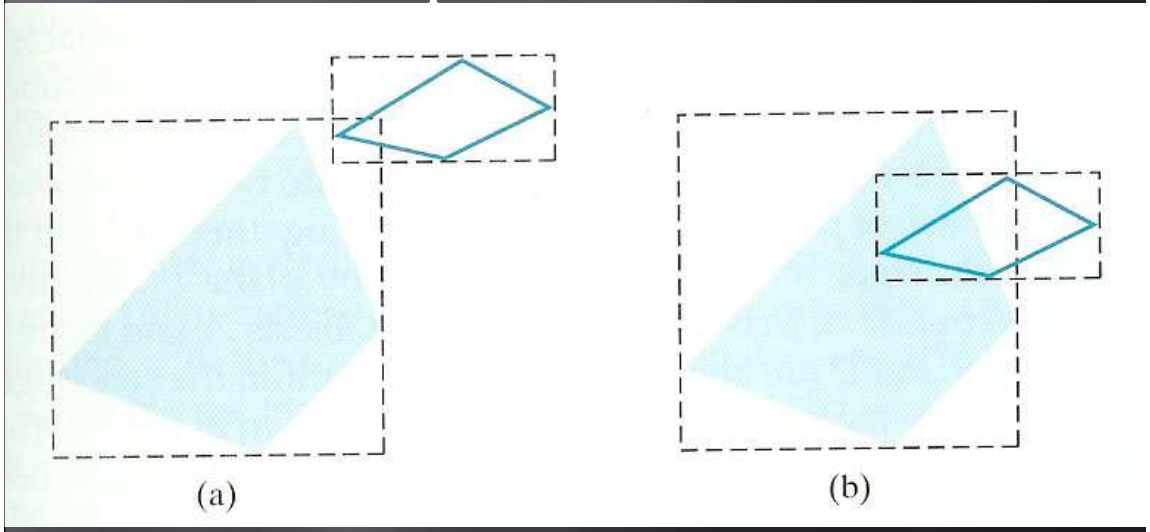
Surface S is completely behind Surface S'



Surface S' is completely in front of Surface S, but S is not completely behind S'

Depth Sorting

- Example



Two surfaces with overlapping bounding rectangles in the xy plane.

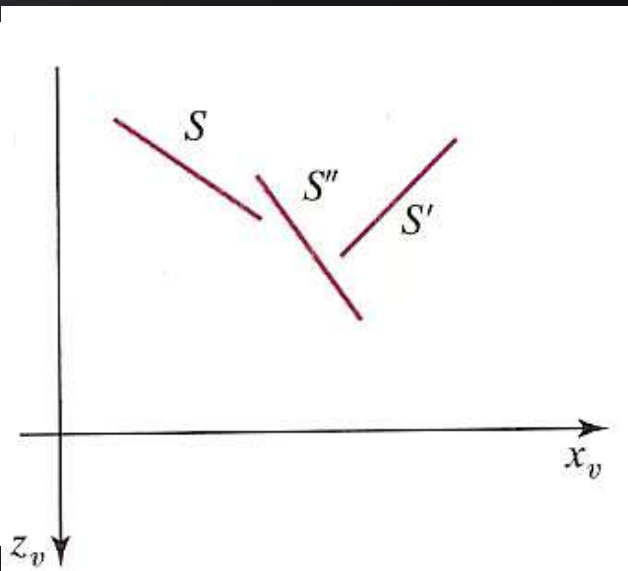


FIGURE 9-18 Three surfaces that have been entered into the sorted surface list in the order S, S', S'' should be reordered as S', S'', S.

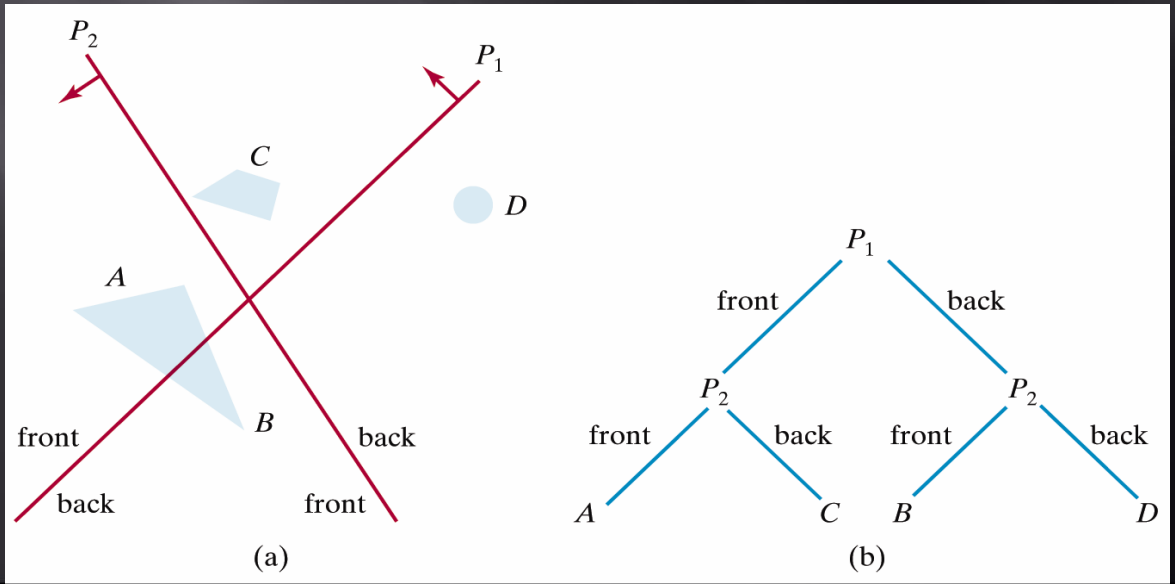
BSP Trees

- Binary space partitioning is an efficient method for determining object visibility
- Paint surfaces into the frame buffer from back to front as in painter's algorithm
- Particularly useful when the view reference point changes, but the objects in a scene are at fixed positions

BSP Tree Construction

Identify surfaces that are inside or outside the partitioning plane at each step of the space subdivision, relative to the viewing direction.

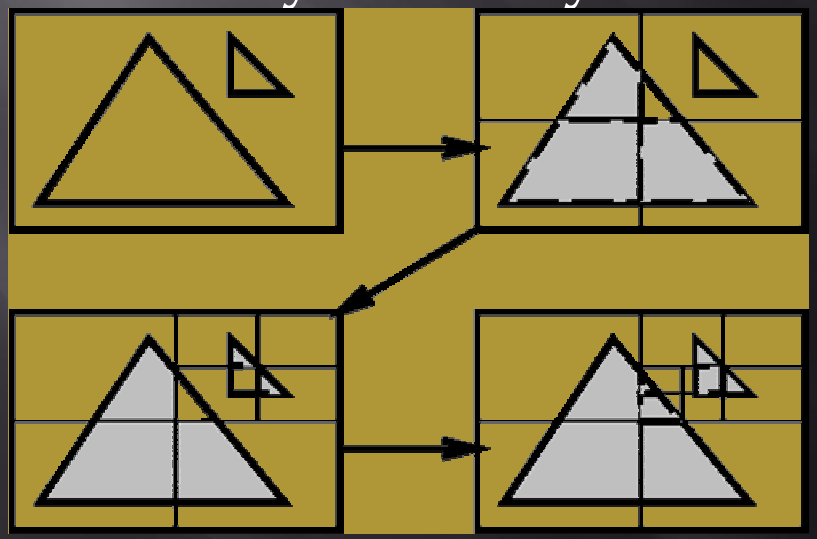
1. Partition the space into two sets of objects using plane P1.
2. One set of objects is in front and the other at the back w.r.t the viewing direction
3. If plane intersects an object, divide it into two
4. Partition the space again using plane P2 and construct the binary tree representation.



Objects are terminal nodes – front objects as left branches and the back objects as right branches

Area Subdivision

- Image-space method taking advantage of area coherence in a scene
- Recursively subdivide a square area into equal-sized quadrants if the area



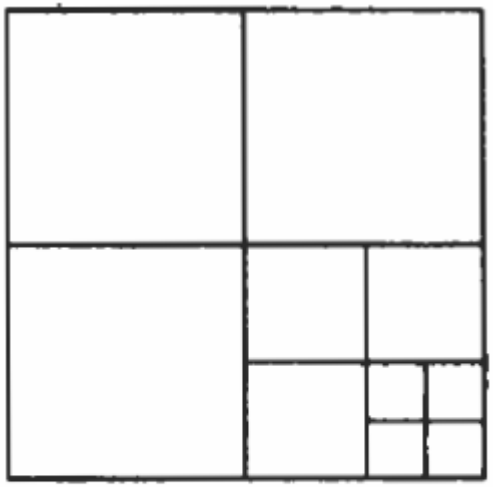
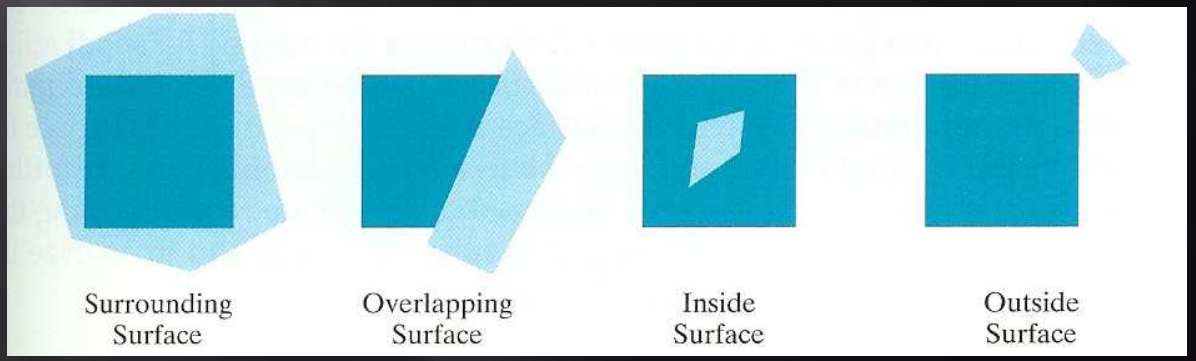


Figure 13-20
Dividing a square area into equal-sized quadrants at each step.

Area Subdivision

- Tests to determine the visibility of a single surface within a specified area are made by comparing surfaces to the boundary of the area.
- Four possible relationships that a surface can have with a specified area boundary.
  - Surrounding Surface – one that completely encloses the area.
  - Overlapping Surface – partly inside and partly outside
  - Inside Surface- completely inside
  - Outside Surface- completely outside



Area Subdivision

- Further subdivision is not necessary if any of the following is true
  1. All surfaces are outside surfaces with respect to the area.
  2. Only one inside, overlapping or surrounding surface is in the area
  3. A surrounding surface obscures all other surfaces within the area boundaries

Area Subdivision

- **Test 1** :
  - checking the bounding rectangles of all surfaces against the area boundaries
- **Test 2:**
  - Bounding rectangle is used as an initial check and then further tests to determine if surface is inside, overlapping or surrounding
- **Test 3:**
  - Order surfaces according to their minimum depth from the view plane.
  - For each surrounding surface, compute the maximum depth within the area under consideration.
  - If maximum depth of one surface is closer to the view plane than the minimum depths of all other surfaces then test 3 is satisfied.

Area Subdivision

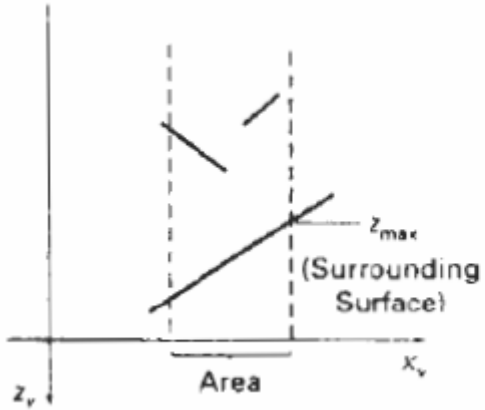


Figure 13-22
Within a specified area, a surrounding surface with a maximum depth of z_{max} obscures all surfaces that have a minimum depth beyond z_{max}.

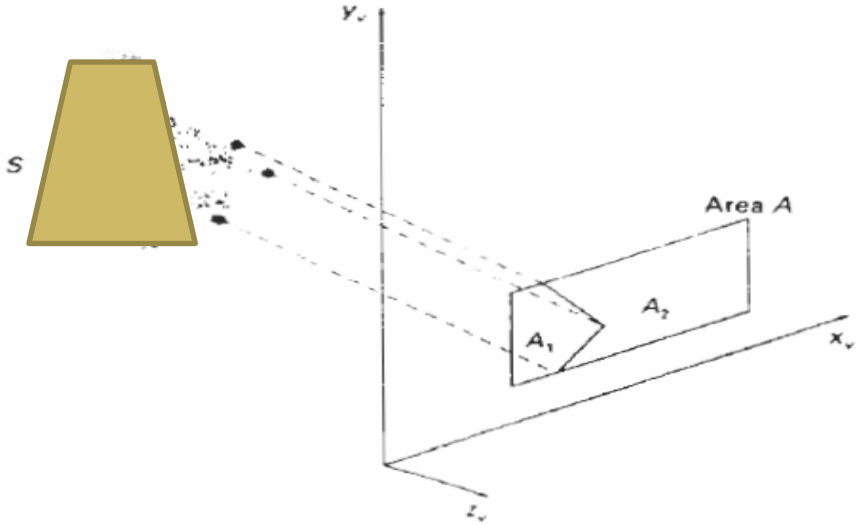


Figure 13-23
Area A is subdivided into A_1 and A_2 using the boundary of surface S on the view plane.

Instead of subdividing into half, sub divide areas using surface boundaries

Octrees

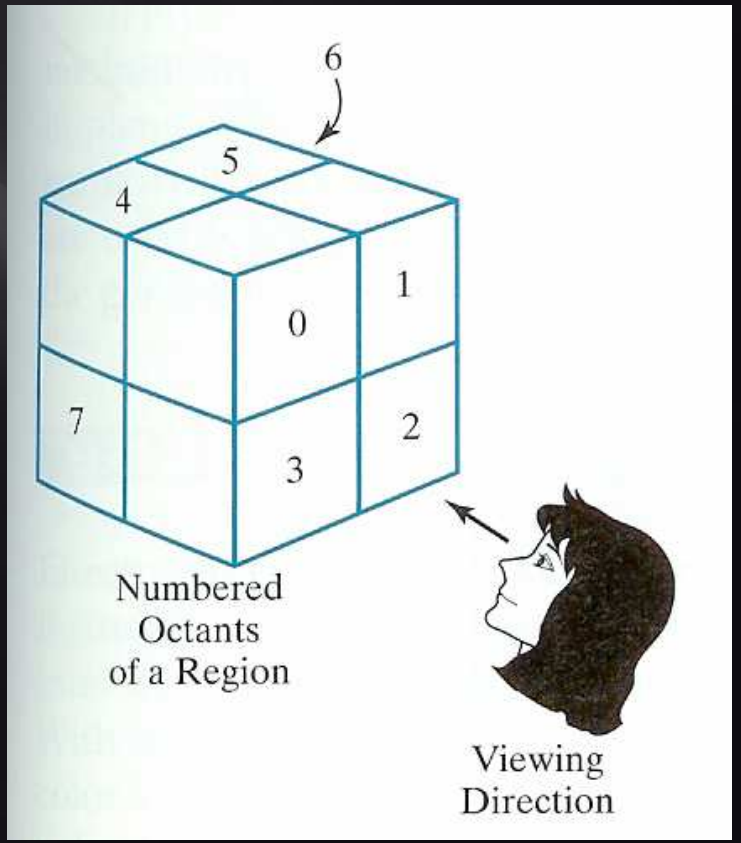
- Octree representation is used for the viewing volume
- Hidden-surface elimination is accomplished by projecting the octree nodes onto the viewing surface in a front-to-back order

Octree nodes are processed in the order 0,1,2,3,4,5,6,7

When a color value is encountered in an octree node, the pixel area in the frame buffer corresponding to this node is assigned that color value only if no values have been previously been stored in this area.

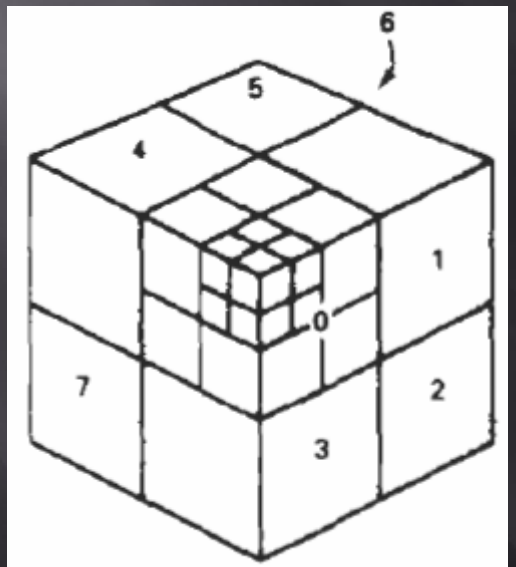
So, only front colors are loaded.

Nothing is loaded if an area is void

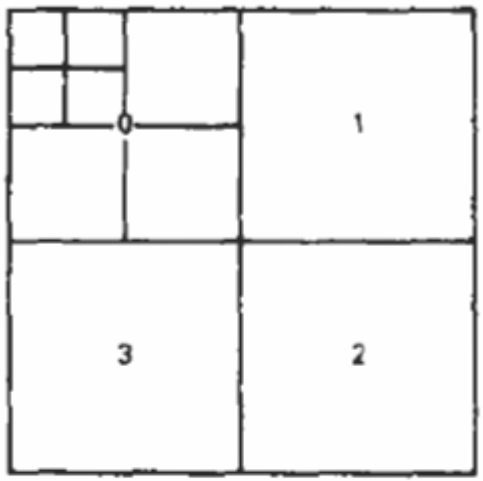


Octrees

- Displaying an octree:
  - Map the octree into a quadtree of visible areas by traversing the octree nodes from front to back in a recursive procedure.
  - Then, this quadtree representation is loaded into the frame buffer.



Octants in Space



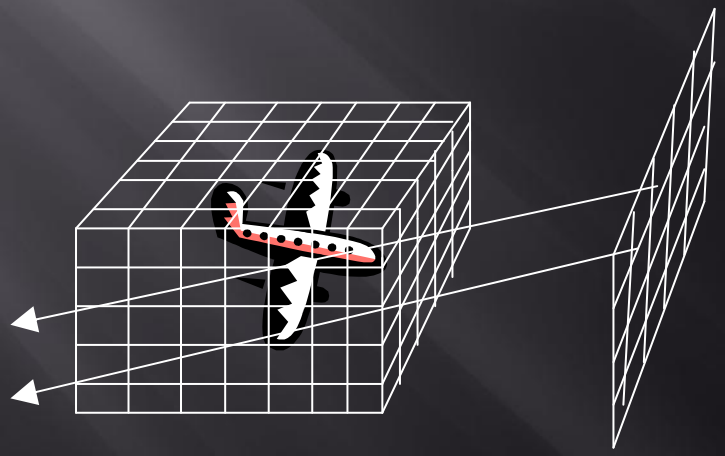
Quadrants for the View Plane

Figure 13-25
Octant divisions for a region of space and the corresponding quadrant plane.

Contributions to quadrant 0 come from octants 0 and 4

Ray Casting

- We consider the line of sight from a pixel position through the scene
- Determine which objects in the scene intersects this line.
- The visible surface is the one whose intersection point is closest to the pi
- Similar to depth-buffer method



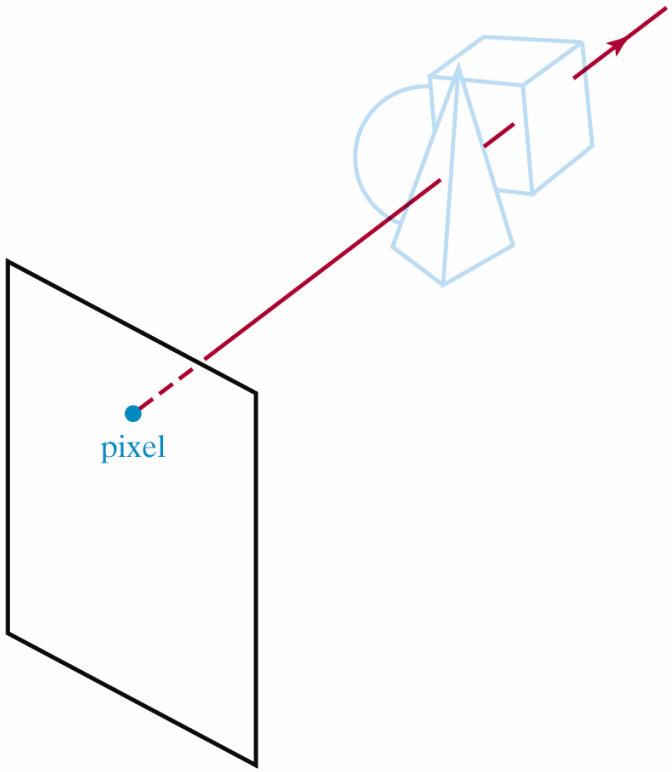
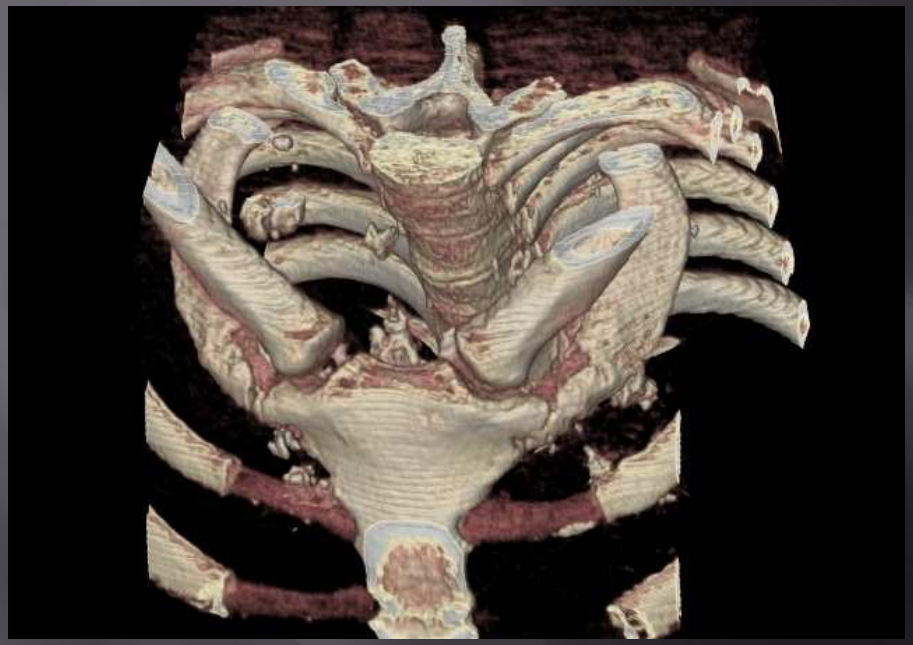


Figure 9-26
A ray along the line of sight from a pixel position through a scene.

Ray Casting Examples



Curved Surfaces

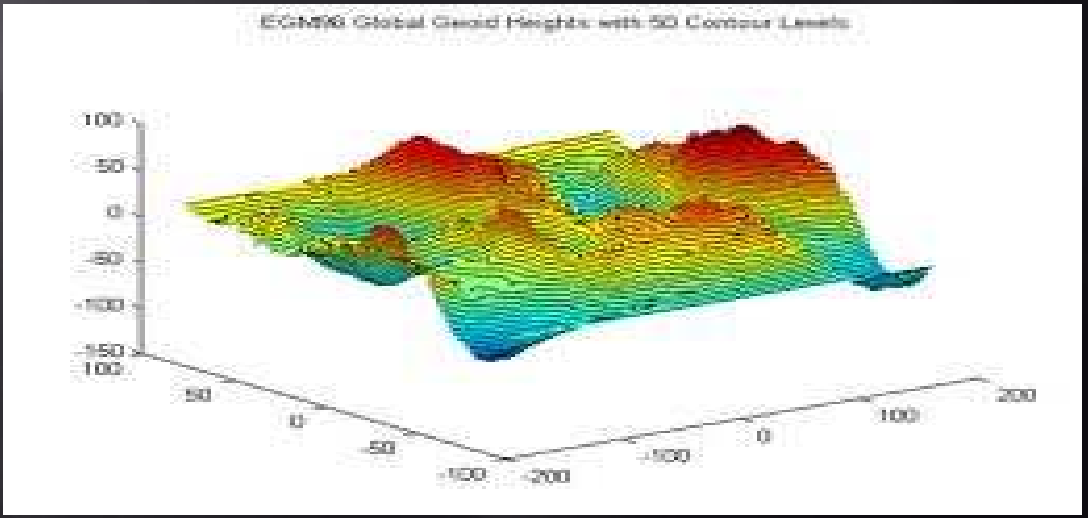
- Approximate a curved surface as a set of plane, polygon surfaces and use any of the previously discussed methods.
- **Surface contour plots**
  - Display a surface function with a set of contour lines that show the surface shape.
  - Plot the visible surface contour lines and eliminate those that are hidden
- A curve in the xy plane can be plotted for values z within some specified Δz.
- Starting with the largest value of z, we plot the curve sections from front to back, eliminating those which are hidden.
  - Unit steps are taken in x and corresponding $y = f(x,z)$

Curved Surfaces

- One way to identify the visible curve sections on the surface is to maintain a list of y_{min} and y_{max} values previously calculated for the pixel coordinates on the screen.
- For next x position check if

$$y_{\min} \leq y \leq y_{\max}$$

- If true, that point on the surface is not visible.
- If y falls outside the range, then, plot the point and reset the 'y' bounds.



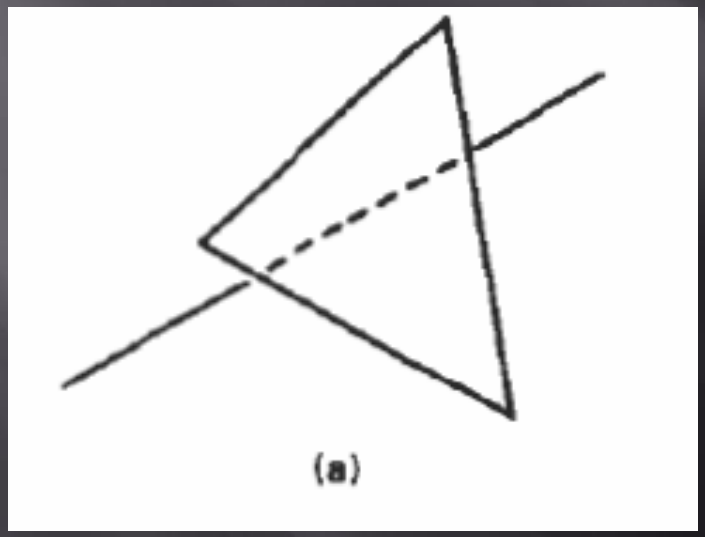
Wireframe methods

- Outline of an object is displayed, so visibility tests are applied to surface edges.
- Hidden sections are either eliminated or displayed differently from the visible edges.
- Identify visible lines by comparing each line to each surface.
- Similar to clipping a line against an arbitrary window shape.

- If both line intersections with the projection of the surface boundary have greater depth than the surface then the line segment between the intersections is hidden.

- When a line has greater depth at one intersection and less depth at the other, then it penetrates the surface, so display only a portion of the line segment that is visible.

Wireframe methods



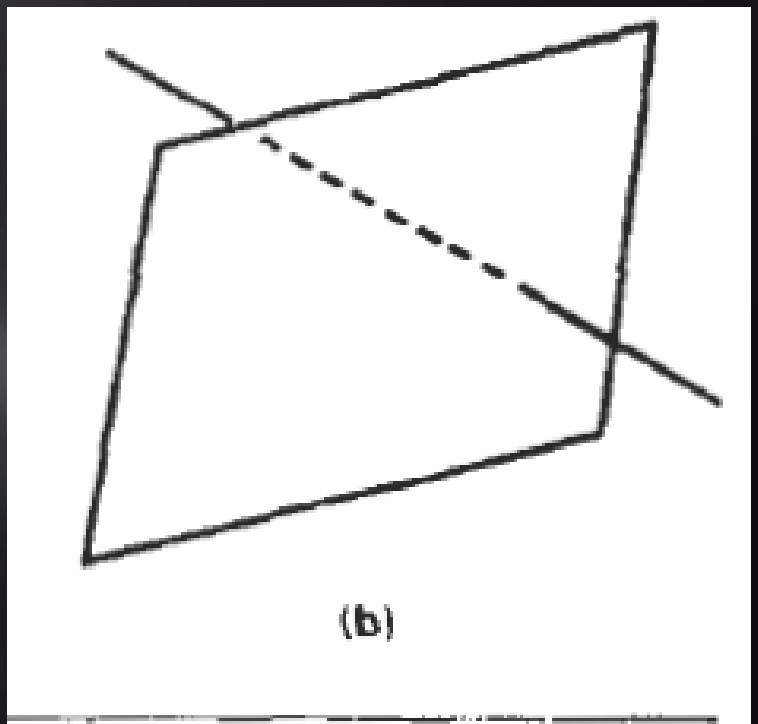


Figure 13-28
Hidden-line sections (dashed) for a line that (a) passes behind a surface and (b) penetrates a surface.