HIDDEN-ELEMENT REMOVAL
(Chapter 13 in *Computer Graphics*)

- Classification of Algorithms
- Back-face Removal
- Depth-buffer Method
- Scan-line Method
- Depth-sorting Method
- Area-subdivision Method
- Octree Methods
- Comparison of Hidden-surface Methods
- Hidden-line Elimination
- Curved Surfaces
- Hidden-line and Hidden-surface Command

Classification of Algorithms

- object-space methods
  - deal directly with object definitions
  - compare objects and parts of objects
  - common for hidden-line algorithms
- image-space methods
  - deal with projected images
  - visibility is determined at each pixel position
  - common for hidden-surface algorithms

achieving efficiency

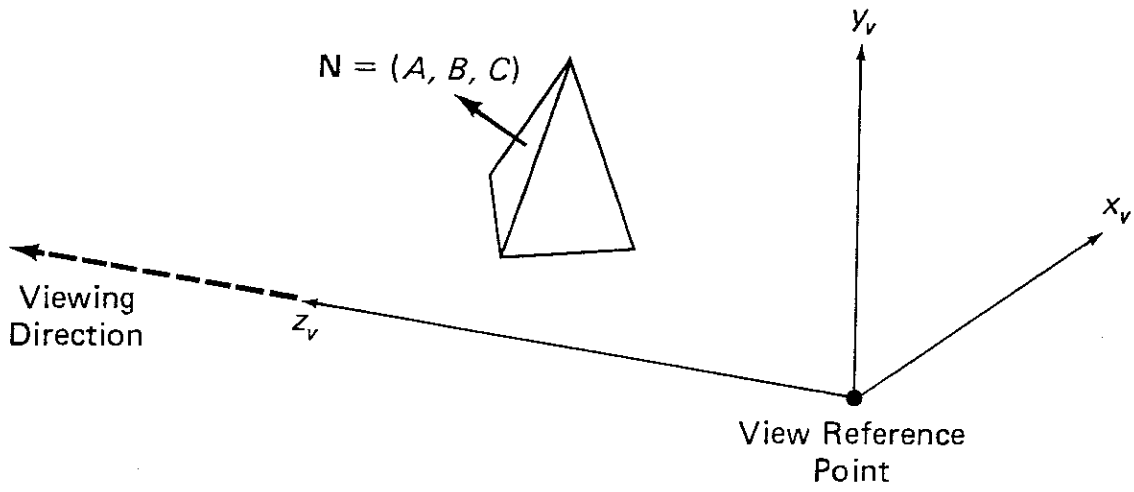
- use sorting
  - order elements according to their distances from the view plane
  - simplifies depth comparisons
- use coherence
  - scan lines contain runs of constant intensity
  - patterns are similar from scan line to scan line
  - scenes are similar from frame to frame
  - relationships between elements often are constant

Back-face Removal

- let (x, y, z) be the viewing position
- let $Ax + By + Cz + D = 0$ be a polygonal face
  - A, B, C and D are determined using vertices chosen in a counterclockwise manner
- if $Ax + By + Cz + D < 0$, then the polygonal face must be a back face and can be eliminated when (x,y,z) is the viewpoint

Back-face Removal, continued

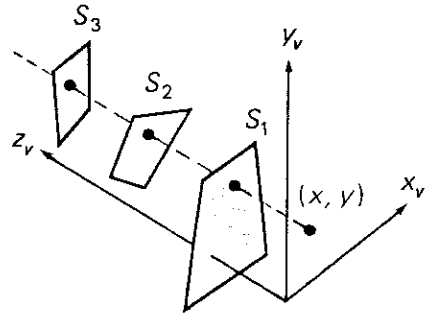
- alternatively, consider the normal vector to the polygonal face, (A, B, C)
  - the viewing direction in left-handed viewing coordinates is along the positive z_v axis



  - this test alone is sufficient for a single convex polyhedron
  - for more complex scenes, the back-face test eliminates about half the polygonal faces
- if C > 0, the normal points away from the viewing position and the polygonal face must be a back face

Depth-buffer (z-buffer) Method

- an image space method
- surfaces are tested one at a time
- at each pixel position on the view plane, the surface with the smallest z coordinate is visible



- requires two buffers
  - a depth buffer stores z values for each (x, y) position
  - a refresh buffer stores intensities
- mapping each surface in the three-dimensional viewport onto the view plane is an orthographic projection

depth-buffer algorithm

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

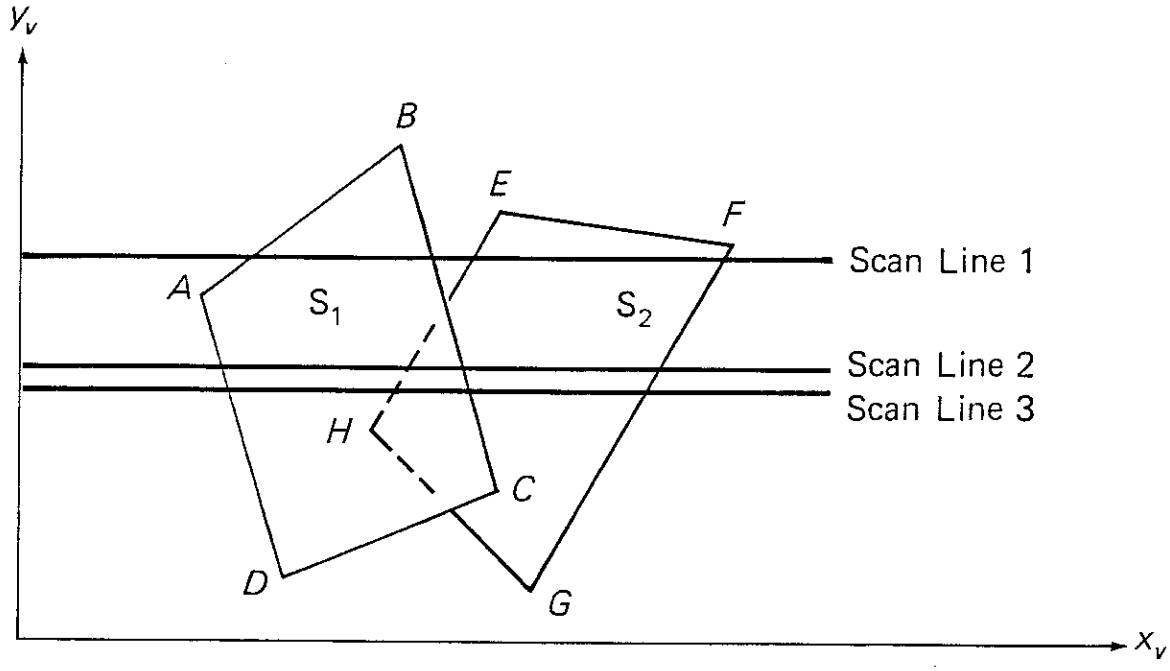
2. For each position on each surface, compare depth values to previously stored values in the depth buffer to determine visibility.
   a. Calculate the z value for each (x, y) position on the surface.
   b. If $z < depth(x, y)$, then set $depth(x, y) = z$ and $refresh(x, y) = i$, where i is the value of the intensity on the surface at position (x, y).

depth buffer calculations

- depth of a planar surface at (x, y)
  $z = (-Ax - By - D)/C$
- depth of a planar surface at (x + 1, y)
  $z' = (-A(x + 1) - By - D)/C$
  $= z - A/C$
- depth of a planar surface at (x, y - 1)
  $z'' = (-Ax - B(y - 1) - D)/C$
  $= z + B/C$
- sorting of surfaces is not required
- substantial storage is required unless the contents of the three-dimensional viewport are processed a section at a time

Scan-line Method

- for each scan line, all polygon surfaces intersecting the scan line are examined to determine which is visible at each position along the scan line

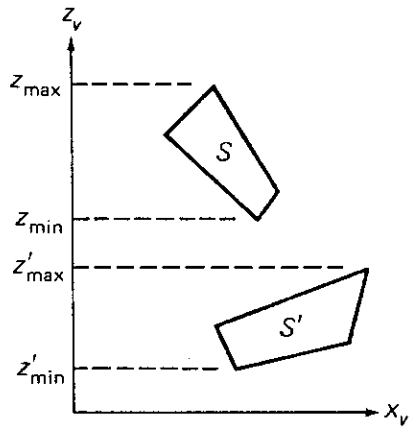


Scan-line Method, continued

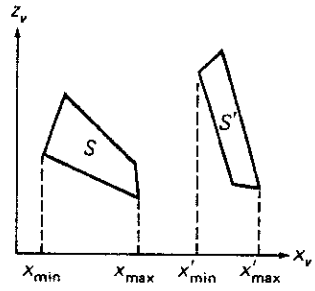
- implementation
  - initialize the refresh buffer
  - maintain a list of active edges, identifying each as a left ("on") edge or a right ("off") edge
  - if only one surface is "on," that surface is visible
  - if multiple surfaces are "on," depth calculations determine which is closest
  - record intensity information in the refresh buffer
- achieving efficiency
  - determine spans of pixels with common intensities along each scan line
  - capitalize on similarities between adjacent scan lines

Depth-sorting Method (painter's algorithm)

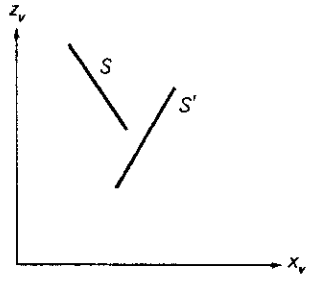
- sort surfaces in order of decreasing depth (in object space)
- scan-convert surfaces in order, starting with the surface of greatest depth (in object space)
- implementation
  - sort surfaces according to the largest z value of each surface
  - if the surface of greatest depth does not overlap any other surface, scan-convert it and proceed to the surface of next greatest depth



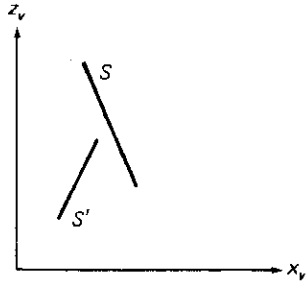
- implementation, continued
  - if depth overlap is detected
    - test the two surfaces for overlap in x and y



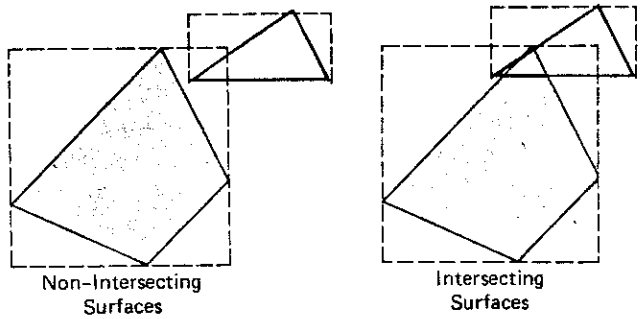
  - if x and y overlap is detected
    - test if all vertices of the surface of greatest depth are further away than the plane of the surface of next greatest depth



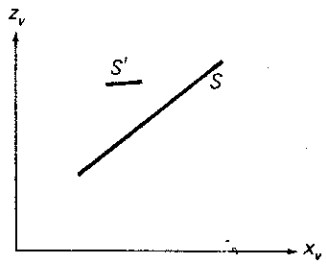
  - if the previous test fails
    - test if all the vertices of the surface of next greatest depth are closer than the plane of the surface of greatest depth



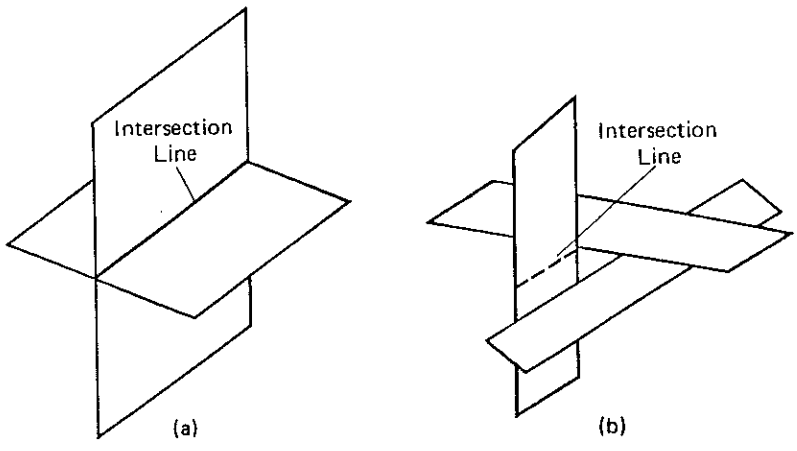
- implementation, continued
  - if the previous test fails
    - test if the projections of the two surfaces overlap



  - if all tests fail
    - interchange the two surfaces and begin again

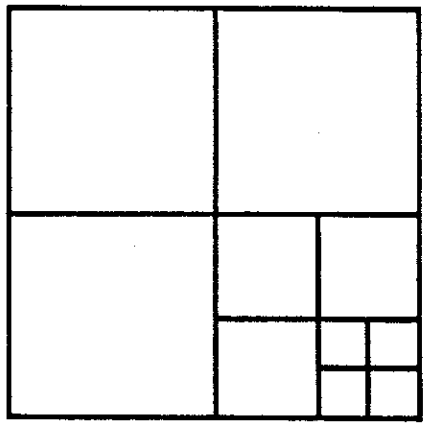


  - if all tests fail again
    - divide along the intersection line and begin again



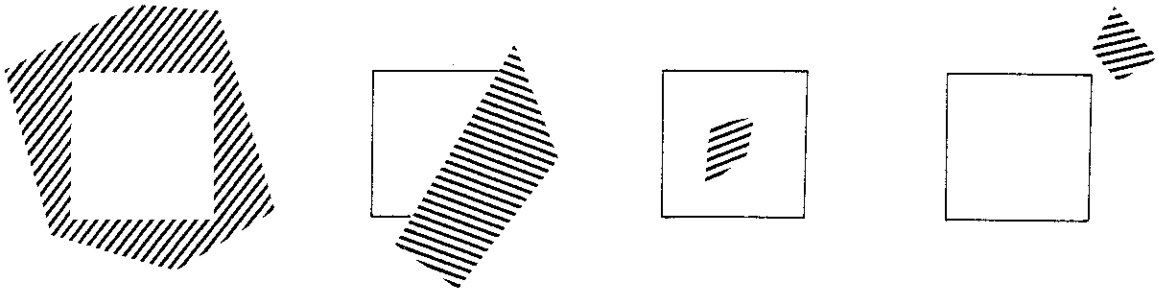
Area-subdivision Method

- algorithm
  - test the entire scene
    - if it is too complex to scan-convert and larger than pixel size, recursively subdivide and test each subdivision
    - otherwise, scan convert

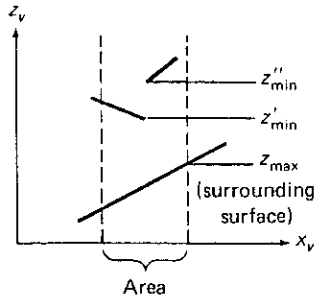


Area-subdivision Method, continued

- relationships between a surface and an area



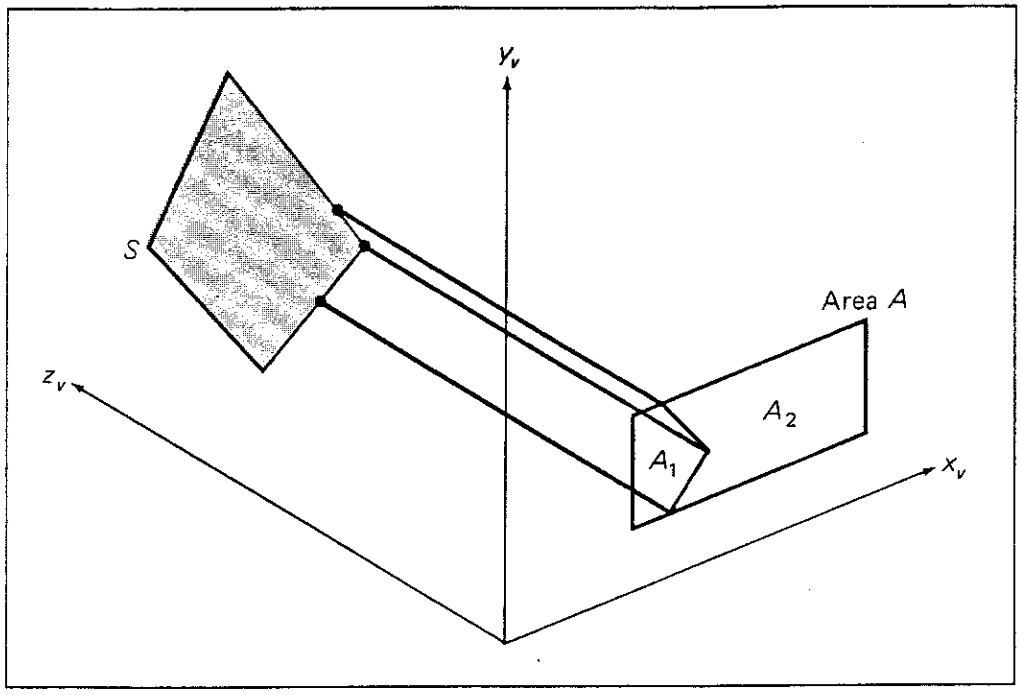
- terminate subdivision if
  - all surfaces are outside the area
    - use bounding rectangles
  - only one surface intersects the area
    - use bounding rectangles
  - a surrounding surface is closer than all other surfaces which intersect the area



  - pixel size is reached

Area-subdivision Method, continued

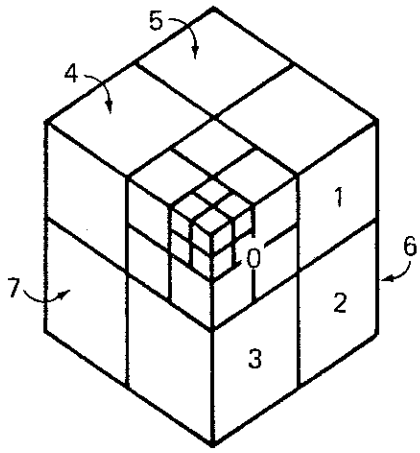
- variation
  - subdivide according to the boundaries of the closest surface



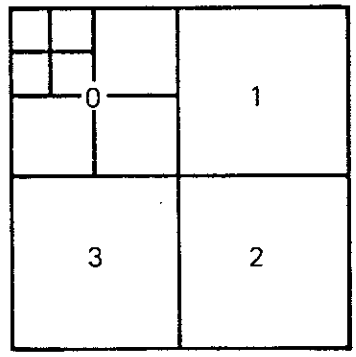
- complexity
  - smaller subdivisions have fewer intersecting surfaces

Octree Methods

- project octree nodes onto the viewing surface in order,front to back, ignoring completely obscured nodes
- write a pixel value into the frame buffer only if no value has yet been written



Octants In Space



Quadrants For
the View Plane

Comparison of Hidden Surface Methods

- if surfaces are distributed in z, depth sorting may be best
- if surfaces are well separated in y, the scan-line or area-subdivision approach may be best
- if there are only a few surfaces, depth sorting or the scan-line approach may be best
- for scenes with at least a few thousand surfaces, the depth-buffer method or the octree approach may be best
- the octree method is useful for obtaining cross-sectional slices

Hidden-line Elimination

- a direct approach
  - compare each line to each surface in the scene to determine if the line
    - is in front
    - is behind
    - intersects
- adaptation of hidden-surface methods
  - only front-face lines are candidates
  - use the depth-sorting method, painting interiors with background color
  - use the area-subdivision method or the scan-line method, displaying boundaries of visible surfaces
- hidden lines can be presented in a different color or with a different texture

Curved Surfaces

- use the octree method
  or
- approximate the curved surface with planar polygons
  or
- use the equation of the surface and numerical approximation techniques to locate intersections with scan lines

Hidden-line and Hidden-surface Command

- select the method
  set_hlhs_method_index (i)

HIDDEN-ELEMENT REMOVAL

- Classification of Algorithms
- Back-face Removal
- Depth-buffer Method
- Scan-line Method
- Depth-sorting Method
- Area-subdivision Method
- Octree Methods
- Comparison of Hidden-surface Methods
- Hidden-line Elimination
- Curved Surfaces
- Hidden-line and Hidden-surface Command