Visible Surface Detection Methods

- the following algorithms are examples for different classes of methods
  - back-face detection
  - depth buffer method
  - scan-line method
  - depth-sorting method
  - area-subdivision method
  - octree methods
  - ray-casting method

Visible-Surface Detection

- identifying visible parts of a scene (also hidden-surface elimination)
- type of algorithm depends on:
  - complexity of scene
  - type of objects
  - available equipment
  - static or animated displays
- object-space methods
  - objects compared to each other
- image space methods
  - point by point at each pixel location
  - often sorting and coherence used

Back-Face Detection (1)

- surfaces (polygons) with a surface normal pointing away from the eye cannot be visible (back faces)
- eliminate them before visibility algorithm!

Back-Face Detection (2)

- eliminating back faces of closed polyhedra
- view point \((x,y,z)\) “inside” a polygon surface if \(Ax + By + Cz + D < 0\)
- or polygon with normal \(N=(A, B, C)\) is a back face if \(V_{\text{view}} \cdot N > 0\)

Back-Face Detection (3)

- object description in viewing coordinates \(V_{\text{view}}=(0,0,V_z)\)
  \(V_{\text{view}} \cdot N = V_z C\)
- sufficient condition: if \(C \leq 0\) then back face
**Back-Face Detection (4)**

- complete visibility test for non-overlapping convex polyhedra

  ![Face partially hidden by other faces](image)

- preprocessing step for other objects: about 50% of surfaces eliminated

**Depth-Buffer Method (1)**

- z-buffer method
- image-space method
- hardware implementation
- no sorting!

**Depth-Buffer Method (2)**

- two buffers
  - depth buffer (distance information)
  - refresh buffer (intensity information)
- size corresponds to screen resolution (for every pixel: r, g, b, z)

**Depth-Buffer Algorithm Example**

<table>
<thead>
<tr>
<th>Image</th>
<th>Depth-Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 3</td>
<td>3 3</td>
</tr>
<tr>
<td>6 7</td>
<td>8 7</td>
</tr>
<tr>
<td>6 5 4</td>
<td>6 5 4</td>
</tr>
<tr>
<td>6 5 4</td>
<td>6 5 4</td>
</tr>
</tbody>
</table>

**Depth-Buffer Algorithm**

for all \((x,y)\)

- \(\text{depthBuff}(x,y) = -\infty\)
- \(\text{frameBuff}(x,y) = \text{backgndColor}\)

for each polygon \(P\)

for each position \((x,y)\) on polygon \(P\)

<table>
<thead>
<tr>
<th>calculate depth (z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (z &gt; \text{depthBuff}(x,y)) then</td>
</tr>
<tr>
<td>(\text{depthBuff}(x,y) = z)</td>
</tr>
<tr>
<td>(\text{frameBuff}(x,y) = \text{surfColor}(x,y))</td>
</tr>
</tbody>
</table>

**Depth-Buffer: Incremental z-Values**

- depth at \((x,y)\):
  \[ z = \frac{-Ax-By-D}{C} \]
- depth at \((x+1,y)\):
  \[ z' = \frac{-A(x+1)-By-D}{C} = z - \frac{A}{C} \]
- depth at \((x,y-1)\):
  \[ z'' = \frac{-Ax-B(y-1)-D}{C} = z + \frac{B}{C} \]
Determine y-coordinate extents of polygon P.

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

\[
z' = \frac{-A(x - 1/m) - B(y - 1) - D}{C}
\]

\[
z = z + \frac{A/m + B}{C}
\]

**Depth-Buffer: Values down an Edge**

- Scan-Line Method
  - Image-space method
  - Extension of scan-line algorithm for polygon filling

- Scan-Line M.: Edge & Polygon Tables
  - Edge table (all edges, y-sorted)
    - Coordinate endpoints
    - Inverse slope
    - Pointers into polygon table
  - Polygon table (all polygons)
    - Coefficients of plane equation
    - Intensity information
    - Pointers into edge table

- Scan-Line Method: Active Edge List
  - Active edge list (all edges crossing current scanline, x-sorted, flag)

- Scan-Line Method Example
Scan-Line Method Details
- coherence between adjacent scan lines
  - incremental calculations
  - active edge list very similar (easy sorting, avoid depth calculations)
- intersecting or cyclically overlapping surfaces!

Depth-Sorting Method: Overview
- surfaces sorted in order of decreasing depth (viewing in −z-direction)
  - "approximate"-sorting using smallest z-value (greatest depth)
  - fine-tuning to get correct depth order
- surfaces scan converted in order
- sorting both in image and object space
- scan conversion in image space
- also called "painter’s algorithm"

Depth-Sorting Method: Sorting (1)
- surface S with greatest depth is compared to all other surfaces S’
  - no depth overlap → ordering correct
  - depth overlap → do further tests in increasing order of complexity

Depth-Sorting Method: Sorting (2)
- ordering correct if
  - bounding rectangles in xy-plane do not overlap
  - check x-, y-direction separately

Depth-Sorting Method: Sorting (3)
- ordering correct if
  - S completely behind S’
  - substitute vertices of S into equation of S’

Depth-Sorting Method: Sorting (4)
- ordering correct if
  - S’ completely in front of S
  - substitute vertices of S’ into equation of S
  - overlapping S’ is completely in front ("outside") of S, but S is not completely behind S’
  - S is not completely behind ("inside") the overlapping S’
Depth-Sorting Method: Sorting (5)

- ordering correct if
  - projections of S, S’ in xy-plane don’t overlap

Depth-Sorting Method: Sorting (6)

- all five tests fail ⇒
  - ordering probably wrong
  - interchange surfaces S, S’
  - repeat process for reordered surfaces

Depth-Sorting: Special Cases

- avoiding infinite loops due to cyclic overlap
  - reordered surfaces S’ are flagged
  - if S’ would have to be reordered again ⇒ divide S’ into two parts

Area-Subdivision Method (1)

- image-space method
- area coherence exploited
- viewing area subdivided until visibility decision very easy

Area-Subdivision Method (2)

- relationship polygon ⇒ rectangular view area
  - surrounding, overlapping, inside, outside surface
- only these four possibilities

Area-Subdivision Method (3)

- three easy visibility decisions
  - all surfaces are outside of viewing area
    - checking bounding rectangles
  - only one inside, overlapping, or surrounding surface is in the area
    - bounding rectangles for initial check
  - one surrounding surface obscures all other surfaces within the viewing area
    - minimum depth ordering
a surrounding obscuring surface
- surfaces ordered according to minimum depth
- maximum depth of surrounding surface closest to view plane?
- test is conservative

Area-Subdivision Method (4)

if all three tests fail ⇒ do subdivision
- subdivide area into four equal subareas
- outside and surrounding surfaces will remain in this status for all subareas
- some inside and overlapping surfaces will be eliminated
- no further subdivision possible (pixel resolution reached)
- sort surfaces and take intensity of nearest surface

Area-Subdivision Method (5)

Area-Subdivision Method Example

Octree Methods
- recursive traversal of octree
  - traversal order depends on processing direction
- front-to-back:
  - pixel(x,y) written once
  - completely obscured nodes are not traversed
- back-to-front:
  - painter’s algorithm

Ray-Casting Method (1)
- line-of-sight of each pixel is intersected with all surfaces
- take closest intersected surface
Ray-Casting Method (2)

- based on geometric optics, tracing paths of light rays
- backward tracing of light rays
- suitable for complex, curved surfaces
- special case of ray-tracing algorithms
- efficient ray-surface intersection techniques necessary
  - intersection point
  - normal vector