Visible Surface Detection (V.S.D)

(Chapt. 15 in FVD, Chapt. 13 in Hearn & Baker)
• Given a set of 3D objects and a viewing specifications, determine which lines or surfaces of the objects should be visible.

• A surface might be occluded by other objects or by the same object (self occlusion)

• Two main approaches:
  – Image-precision algorithms: determine what is visible at each pixel.
  – Object-precision algorithms: determine which parts of each object are visible.
Coherence

• Most methods for V.S.D. use coherence features in the surface:
  – Object coherence.
  – Face coherence.
  – Edge coherence.
  – Scan-line coherence.
  – Depth coherence.
  – Frame coherence.
Single Valued Function of two variables

Without Hidden-Line Removal

With Hidden-Line Removal
Floating Horizon Algorithm

- Implicit Function: \( Y = f(X, Z) \).
- Represent as 2D array of \( x \) and \( z \) values, each entry is the corresponding \( y \)-value.
- Surface = many polylines; Each polyline is constant in \( Z \).

**Algorithm:**

Draw polylines of constant \( z \) from front (near \( z \)) to back (far \( z \)).

Draw only parts of polyline that are visible: ie above/below the silhouette (horizon).
Use 2 1D arrays $YMIN$ and $YMAX$ (with 1 entry for each $x$). When drawing a polyline of constant $z$, for each $x$-value, test if above/below $YMAX/YMIN$ (at $x$ location) and update arrays.
• **Floating Horizon Characteristics:**
  – Applied in image space (image precision).
  – Limited to explicit functions only.
  – Exploiting edge coherence.
  – Applicable for free-form surfaces.
Back Face Detection

- **Observation**: In a volumetric object, you never see the “back” faces of the object (self occlusion).

- **Reminder**: 
  - Plane equation: $Ax + By + Cz + D = 0$
  - $N = [A, B, C]^T$ is the plane normal.
  - $N$ points "outside".

- Back facing and front facing faces can be identified using the sign of $V \cdot N$

- In which coordinates $N$ is represented?
• Three possibilities:
  – $V \cdot N > 0$ back face
  – $V \cdot N < 0$ front face
  – $V \cdot N = 0$ on line of view

• For convex objects, back face detection actually solves the visible surfaces problem.

• Back face detection is easily applied to convex polyhedral objects.

• In a general object, a front face can be visible, invisible, or partially visible.
Back Face Polygons:  A, B, D, F
Front Face Polygons:  C, E, G, H
Quantitative Visibility

• First general hidden surface algorithm, by Appel 1967.

• **Definition**: Every edge has a non-negative *quantitative visible value*, $Q_v$, which corresponds to the number of times the edge is obscured. If $Q_v=0$ the edge is visible.
• **Definition:** An *active edge* is a silhouette edge, i.e., it is shared by *back* and *front* faces.

• **Observations:** The visibility of an edge can be changed only where it intersects another *active* edge in the viewing plane.

• If the edge does not intersect any *active* edge, its visibility is homogeneous.
• **Algorithm:**
  - Select a single point on line and test how many polygons obscure it.
  - Increment/decrement $Q_v$ any time the line intersects an active edge, and the intersection is inside the view triangle.
  - Propagate from the end point to a neighboring line.
  - Fill the resulting polygons appropriately.

• **Question:** How do we know if the line “enter” or “leave” an obscuring face?
• **Answer:** Edges of a polygon are described clockwise when viewing the object from “outside” (so the face is to the right).

• We test the configuration of the line direction with the direction of intersecting edge describing the front face.

Leave a face - decrease $Q_v$

Enter a face - increase $Q_v$
Depth-Buffer Method (Z-Buffer)

• In addition to the frame buffer (keeping the pixel values), keep a Z-buffer containing the depth value of each pixel.

• Surfaces are scan-converted in an arbitrary order. For each pixel \((x,y)\), the Z-value is computed as well. The \((x,y)\) pixel is overwritten only if its Z-values is closer to the viewing plane than the one already written at this location.
Example of Compositing surfaces Using Z-Buffer
Algorithm:

- Initialize the z-buffer and the frame-buffer:
  \[ \text{depth}(x,y) = \text{MAX}_Z; \quad I(x,y) = I_{\text{background}} \]

- Calculate the depth Z for each \((x,y)\) position on any surface:
  
  • If \(z < \text{depth}(x,y)\), then set
    
    \[ \text{depth}(x,y) = z; \quad I(x,y) = I_{\text{surf}}(x,y) \]

  • For polygon surfaces, the depth-buffer method is very easy to implement using polygon scan line conversion, and exploiting face coherence and scan-line coherence:
    
    • \( Z = -\frac{(Ax+By+D)}{C} \)
    
    • Along scan lines
      
      \[ Z' = -\frac{(A(x+1)+By+D)}{C} = Z - \frac{A}{C} \]
    
    • Between successive scan lines:
      
      \[ Z' = -\frac{(Ax+B(y+1)+D)}{C} = Z - \frac{B}{C} \]
Depth-Buffer Characteristics

• Implemented in the image space.
• Very common in hardware due its simplicity (SGI's for example).
• 32 bits per pixel for Z is common.
• Advantages:
  – Simple and easy to implement.
• Disadvantages:
  – Requires a lot of memory.
  – Finite depth precision can cause problems.
  – Might spend a lot of time rendering polygons that are not visible.
  – Requires re-calculations when changing the objects scale.
Depth Sort
(Painter Algorithm)

• Sort all of the polygons in the scene by their depth.
• Draw them back to front.
• **Question**: Does a depth ordering always exist?
• **Answer**: Unfortunately, no!
• For polygons with constant $Z$ value, this sorting clearly works.
• For example: VLSI design, or in window systems.
• **Question:** What if polygons are not Z constant?

• **Observation:** Given two polygons P and Q, an order may be determined between them, if at least one of the following holds:

  – 1. Z values of P and Q do not overlap.
  – 2. The bounding rectangle in the x,y plane for P and Q do not overlap.
  – 3. P is totally on one side of Q’s plane.
  – 4. Q is totally on one side of P’s plane.
  – 5. The bounding rectangles of Q and P do not intersect in the projection plane.
If all the above conditions do not hold, P and Q may be split along intersection edge into two smaller polygons.
The BSP Tree

- **BSP** = Binary Space Partitioning.
- Interior nodes correspond to partitioning planes.
- Leaf nodes correspond to convex regions of space.
• Tests 3 and 4 in *Depth Sort* technique can be exploited efficiently:
• Let $L_p$ be the plane $P$ lies in: The 3D space may be divided into the following three groups:
  – Polygons in front of $L_p$.
  – Polygons behind $L_p$.
  – Polygons intersecting $L_p$.
• Polygons in the third class are split, and classified into the first two.
• As a result of the subdivision with respect to $L_p$:
  – The polygons behind $L_p$ cannot obscure $P$, so we can draw them first.
  – $P$ cannot obscure the polygons in front of $L_p$ so we can draw $P$ second.
  – Finally we draw the polygons in front of $P$. 
The BSP-Tree Algorithm

• Construct a BSP tree:
  – Pick a polygon, let its supporting plane be the root of the tree.
  – Create two lists of polygons: these in front, and those behind (splitting polygons as necessary).
  – Recurse on the two lists to create the two sub-trees.

• Display:
  – Traverse the BSP tree back to front, drawing polygons in the order they are encountered in the traversal.
Should be prepared from the beginning!
BSP Properties:

• The BSP tree is *view independent*!
• The BSP tree is constructed using the geometry of the object only.
• The tree can be used for hidden surface removal at an arbitrary direction.
• BSP = Object-precision alg.
Area Subdivision Technique (Warnock 1969)

- Subdivide screen area recursively, until visible surfaces are easy to determine.
- Each polygon has one of four relationships to the area of interest:
  - Surrounding
  - Intersecting
  - Contained
  - disjoint

Surrounding  Intersecting  Contained  disjoint
• If all polygons are disjoint from the area, fill area with background color.
• Only one intersecting or contained polygon: First fill with background color, then scan convert polygon.
• Only one surrounding polygon: Fill area with polygon’s color.
• More than one polygon is surrounding, intersecting, or contained, but one surrounding polygon is in front of the rest: Fill area with polygon’s color.
• If none of the above cases occurs: Subdivide area into four, and recurse.
• Area subdivision = Image precision technique.
When the resolution of the image is reached, polygons are sorted by their Z-values at the center of the pixel, and the color of the closest polygon is used.