AA Example: Visibility

- The problem: Given a geometric model (list of polygons) and a view specification, generate the image which represents that scene viewed in that way
- Many ways to approach the problem
  - Ivan Sutherland, A Characterization of Ten Hidden Surface Algorithms, 1974
  - More approaches in the decades since
Painter’s Algorithm

- **Approach**
  - Sort polygons
  - Draw polygons in order: furthest to closest

Given
- List of Polygons \{P1, P2, \ldots, Pn\}
- Array of Intensity \([x, y]\)

Begin
- Sort polygon list on minimum z (largest z first)
- For each polygon P in selected list do
  - For each pixel \((x,y)\) that intersects P do
    - Intensity \([x,y]\) = intensity of P at \((x,y)\)
  - Display Intensity array
Z-buffer

- **Approach**
  - Draw polygons in arbitrary order: store depth
  - At each pixel, overwrite if new pgon closer
  
  - *Pgon: (1,1,5), (7,7,5), (1,7,5)*

---

Z-buffer Example (1)

- *Pgon: (1,1,5), (7,7,5), (1,7,5)*
Zbuffer Example (2)

- **Pgon**: (3,5,9), (10, 5, 9), (10,9,9), (3,9,9)

Zbuffer Example (3)

- **Pgon**: (2,6,3), (2,3,8), (7,3,3)
Zbuffer Example (4)

Zbuffer Algorithm

- **Given**
  - List of Polygons \{P_1, P_2, \ldots, P_n\}
  - Array of Intensity \([x, y]\)
  - Array z-buffer \([x, y]\), initialized to +infinity

Begin
  For each polygon \(P\) in selected list do
    For each pixel \((x, y)\) that intersects \(P\) do
      Calculate z-depth of \(P\) at \((x, y)\)
      If z-depth < z-buffer\([x, y]\) then
        Intensity \([x, y]\) = intensity of \(P\) at \((x, y)\)
        z-buffer\([x, y]\) = z-depth
      Display Intensity array
Scanline Algorithm

- **Approach**
  - Consider one row of image (scanline) at a time
  - Identify coherent runs in scanline

### Given
- List of Polygons \( \{P_1, P_2, \ldots, P_n\} \)
- Array of Intensity \([x, y]\)

**Begin**
- Sort polygons into sorted surface table (SST) on \( y \)
- Initialize \( y \) and active surface table (AST)
- Repeat until AST and SST empty
  - identify spans for this scanline (sorted on \( x \))
  - for each span
    - determine visible element (based on \( z \))
    - fill pixel intensities with values from element
  - update AST: \( y++ \)
  - remove exhausted edges
  - update \( x \) intercepts
  - resort AST on \( x \)
  - add entering polygons
- **Display Intensity array**
Ray Casting

- Approach
  - Project sight rays from eye point through pixel into scene
  - Draw thing found at first intersection of each pixel

Ray Casting Algorithm (1)

- Given
  - List of Polygons \( \{ P_1, P_2, \ldots, P_n \} \)
  - Array of Intensity \([x, y]\)

Begin

For each pixel \((x,y)\) {
  - form a ray \( R \) in object space through the camera position \( C \) and the pixel \((x,y)\)
  - Intensity \([x,y]\) = trace\((R)\)
}

Display array Intensity
Ray Casting Algorithm (2)

Intensity trace(Ray) {
    for each polygon P in the scene
        Calculate the intersection of P and R
    If ( R hit no pgon)
        return (background intensity)
    Else {
        Find pgon P with closest intersection
        Calculate intensity I at intersection point
        Return (I)
    }
}

Visibility Algorithm Taxonomy

- Basic design choice
  - Object space: organize by pgon
  - Image space: organize by pixel

diagram
Visibility Algorithm Taxonomy

- Also consider continuous output

![Diagram of Visibility Algorithm Taxonomy](image)
Reviewing the Code

Painter’s Algorithm

- **Given**
  - List of Polygons \{P_1, P_2, \ldots, P_n\}
  - Array of Intensity \([x, y]\)

**Begin**

Sort polygon list on minimum z (largest z first)
For each polygon \(P\) in selected list do
  - For each pixel \((x, y)\) that intersects \(P\) do
    - Intensity \([x, y]\) = intensity of \(P\) at \((x, y)\)
  - Display Intensity array
Zbuffer Algorithm

- **Given**
  - List of Polygons \{P_1, P_2, \ldots, P_n\}
  - Array of Intensity \([x, y]\)
  - Array z-buffer \([x, y]\), initialized to +infinity

  **Begin**
  
  For each polygon \(P\) in selected list do
  
  For each pixel \((x, y)\) that intersects \(P\) do
  
  Calculate z-depth of \(P\) at \((x, y)\)
  
  If z-depth < z-buffer\([x, y]\) then
  
  Intensity \([x, y]\) = intensity of \(P\) at \((x, y)\)
  
  z-buffer\([x, y]\) = z-depth

  Display Intensity array

Scanline Algorithm

- **Given**
  - List of Polygons \{P_1, P_2, \ldots, P_n\}
  - Array of Intensity \([x, y]\)

  **Begin**
  
  Sort polygons into sorted surface table (SST) on \(y\)
  
  Initialize \(y\) and active surface table (AST)
  
  Repeat until AST and SST empty
  
  identify spans for this scanline (sorted on \(x\))
  
  for each span
  
  determine visible element (based on \(z\))
  
  fill pixel intensities with values from element
  
  update AST: \(y++\)
  
  remove exhausted edges
  
  update \(x\) intercepts
  
  resort AST on \(x\)
  
  add entering \(pgons\)

  Display Intensity array
Ray Casting Algorithm (1)

- **Given**
  - List of Polygons \{P1, P2, ..., Pn\}
  - Array of Intensity \([x, y]\)

Begin

- For each pixel \((x,y)\) {
  - form a ray \(R\) in object space through the camera position \(C\) and the pixel \((x,y)\)
  - Intensity \([x,y] = \text{trace}(R)\)
}

Display array Intensity

---

Looking at Performance

<table>
<thead>
<tr>
<th>Approach</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Painter’s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z-buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scanline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Raycast</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>