Today

- Clipping
  - Clipping to view volume
  - Clipping arbitrary polygons
- Hidden Surface Removal
  - Z-Buffer
  - BSP Trees
  - Others
Clipping

- Stuff outside view volume should not be drawn
  - Too close: obscures view

- Too far:
  - Complexity
  - Z-buffer problems

- Too high/low/right/left:
  - Memory errors
  - Broken algorithms
  - Complexity
Clipping Line to Line/Plane

Line segment to be clipped
\[ \mathbf{x}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a}) \]

Line/plane that clips it
\[ \hat{\mathbf{n}} \cdot \mathbf{x} - \hat{\mathbf{n}} \cdot \mathbf{r} = 0 \]
Clipping Line to Line/Plane

Line segment to be clipped
\[ \mathbf{x}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a}) \]

Line/plane that clips it
\[ \mathbf{n} \cdot \mathbf{x} - f = 0 \]
\[ \mathbf{n} \cdot (\mathbf{a} + t(\mathbf{b} - \mathbf{a})) - f = 0 \]
\[ \mathbf{n} \cdot \mathbf{a} + t(\mathbf{n} \cdot (\mathbf{b} - \mathbf{a})) - f = 0 \]

**Clipping Line to Line/Plane**

- Segment may be on one side
  \[ t \notin [0 \ldots 1] \]
- Lines may be parallel
  \[ \mathbf{n} \cdot \mathbf{d} = 0 \]
Clipping Line to Line/Plane

- Segment may be on one side
  \[ t \notin [0 \ldots 1] \]
- Lines may be parallel
  \[ \hat{n} \cdot d = 0 \]
  \[ |\hat{n} \cdot d| \leq \varepsilon \quad \text{(Recall comments about numerical issues)} \]

Triangle Clip/Split
Triangle Clip/Split

Double vertices if you want separation...

Polygon Clip to Convex Domain

• Convex domain defined by collection of planes (or lines or hyper-planes)
• Planes have outward pointing normals
• Clip against each plane in turn
• Check for early/trivial rejection
Polygon Clip to Convex Domain

Note double edges.
<table>
<thead>
<tr>
<th>Inside</th>
<th>Outside</th>
<th>Inside</th>
<th>Outside</th>
<th>Inside</th>
<th>Outside</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- Sutherland-Hodgman algorithm
  - Basically edge walking
- Clipping done often... should be efficient
  - Liang-Barsky parametric space algorithm
  - See text for clipping in 4D homogenized coordinates
General Polygon Clipping

- Weiler Algorithm
- Double edges
Hidden Surface Removal

- True 3D to 2D projection would put every thing overlapping into the view plane.
- We need to determine what's in front and display only that.

Z-Buffers

- Add extra depth channel to image
- Write Z values when writing pixels
- Test Z values before writing
Z-Buffers

• Benefits
  • Easy to implement
  • Works for most any geometric primitive
  • Parallel operation in hardware

• Limitations
  • Quantization and aliasing artifacts
  • Overfill
  • Transparency does not work well

Z-Buffers

• Transparency requires partial sorting:

  Partially transparent 3rd
  Opaque 2nd
  Opaque 1st

  Front

  Partially transparent 1st
  Opaque 3rd
  Opaque 2nd

  Good
  Not Good
### Z-Buffers

Recall depth-value distortions.

It's a feature...

- More resolution near viewer
- Best use of limited precision

### A-Buffers

- Store sorted list of “fragments” at each pixel
- Draw all opaque stuff first then transparent
- Stuff behind full opacity gets ignored
- Nice for antialiasing...
Scan-line Algorithm

- Assume polygons don’t intersect
- Each time an edge is crossed determine who’s on top

Painter’s Algorithm

- Sort Polygons Front-to-Back
  - Draw in order
  - Back-to-Front works also, but wasteful
- How to sort quickly?
- Intersecting polygons?
- Cycles?
BSP-Trees

- Binary Space Partition Trees
  - Split space along planes
  - Allows fast queries of some spatial relations
- Draw Front-to-Back
  - Draw same-side polygons first
  - Draw root node polygon (if any)
  - Draw other-side polygons last