Clipping

- Removing primitives (lines, polygons) that are not visible
- Can take place in different stages
  - In image space
    - OpenGL does only this clipping
  - In object space
    - Several methods
Window Coordinates

- After projection we are in normalized device coordinates
- Convert to window coordinates

Line Clipping

- Accept AB
- Reject (Cull) CD
- Clip EF
  - One endpoint outside the window
- Clip GH
  - Both endpoints outside the window
Cohen-Sutherland Clipping

- Divide the window in nine regions marked by binary codes

<table>
<thead>
<tr>
<th>$y = y_{\text{max}}$</th>
<th>$y = y_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>0001</td>
</tr>
<tr>
<td>1000</td>
<td>0000</td>
</tr>
<tr>
<td>1010</td>
<td>0101</td>
</tr>
</tbody>
</table>

- $y > y_{\text{max}}$ $y < y_{\text{min}}$ $x > x_{\max}$ $x < x_{\min}$
- $b_1$ $b_2$ $b_3$ $b_4$

Find binary codes of two endpoints ($C_1$, $C_2$)

- $C_1 = C_2 = 0$
  - Accept the line, both endpoints inside the window
- $C_1 = 0$, $C_2 \neq 0$
  - One endpoint outside the window
  - Nonzero bits give the lines with which to intersect
  - Maximum two intersection to get the clipped line
Cohen-Sutherland Clipping

- Find binary codes of two endpoints \((C_1, C_2)\)
  - \(C_1 \& C_2 \neq 0\)
    - Both endpoints outside the same edge of the window
    - Cull completely
  - \(C_1 \& C_2 = 0\)
    - Both endpoints outside, but different edges
    - Find the first intersection and find its binary code
    - Apply recursively on this culled line

- All these carried out in order
- Boolean operations, intersections if needed

Advantage
- Most lines can be eliminated based on codes
- Can be easily extended to 3D
- Plane-line intersection instead if line-line intersection

Disadvantage
- Has to be applied recursively
Extending it to 3D

- How many bits for codes? How many codes?
- Plane-line intersection
  - \( p(\alpha) = p_1 + \alpha(p_2-p_1) \)
  - \( n.(p(\alpha)-p_0) = 0 \)
  - \( \alpha = \frac{n.(p_1-p_0)}{n.(p_2-p_1)} \)

Liang-Barsky Clipping

- Take the parametric equation of the line
- Find intersection with four lines of windows
- Order the alphas
Liang-Barsky Clipping

- $1 > a_4 > a_3 > a_2 > a_1 > 0$
  - Line meets l before t
  - Line between $a_2$ and $a_3$ is inside the window
- $1 > a_4 > a_2 > a_3 > a_1 > 0$
  - Line meets t before l
  - Whole line is outside
  - Reject completely

Efficiency Improvements

- Compute intersections one by one
  - May need less than four intersections to reject
- Compare without floating point division
  - If $(a_2 < a_3)$ then $(y_{\text{max}} - y_1)(x_2 - x_1) < (x_{\text{min}} - x_1)(y_2 - y_1)$

\[
\begin{align*}
y_{\text{max}} &= (1 - a_3)y_1 + a_3y_2 \\
x_{\text{min}} &= (1 - a_2)x_1 + a_2x_2 \\
a_3 &= \frac{y_{\text{max}} - y_1}{y_2 - y_1} \\
a_2 &= \frac{x_{\text{min}} - x_1}{x_2 - x_1}
\end{align*}
\]
### Polygon Clipping

- Convex polygons clipped to a single polygon
- Concave polygons
  - Clip and join to a single polygon
  - Tessalate and clip triangles

### Clipping Convex Polygons

- Sutherland Hodgeman
- Subproblem
  - Input: vertex list (polygon) and a clipping line
  - Output: vertex list (clipped polygon)
- In a pipeline for 4 clipping lines
Sutherland Hodgeman

- To clip vertex list against the line
  - Test first vertex, Output if inside else skip
  - Then loop through the list, testing transitions
    - In-to-out: Output intersection
    - In-to-in: Output vertex
    - Out-to-in: Output intersection and vertex
    - Out-to-out: Output nothing
- Can form a pipeline
  - Process vertex list concurrently
- Can be extended to 3D easily (line-plane intersection)

Bounding Boxes and Volumes

- Polygon clipping is overkill if entire polygon outside the window
- Maintain a bounding box
  - Axis-aligned
- Can be a big savings
- Can be easily extended to 3D
  - For volumes in object-space
View Frustum Culling

- Preprocessing: Spatial Subdivision
  - Octree subdivision: Hierarchical Structure
  - Each box has a list of polygons inside it
  - An empty box is the leaf node
- If completely inside the view frustum
  - Accept
- If completely outside the view frustum
  - Reject
- If intersects the view frustum
  - Go through the children recursively

View Frustum Culling

- What happens when a triangle spans across a box?
  - Split the triangle
  - Include it in both boxes
    - Screen space clipping takes care of it
- Octree is suboptimal division
  - Other methods control depth of the tree
Hidden Surface Removal

- Object Space Approach
  - Back Face Culling
  - Painter’s Algorithm
- Image Space Approach
  - Z-buffer algorithm

Back Face Culling

- Do not want to render back facing polygons
- If the normal is pointed towards the viewer
  - \(-90 \leq \theta \leq 90\)
  - \(\cos(\theta) \geq 0\)
  - \(n.v \geq 0\)
- Viewing in -z
  - Culled if normal has negative z
Painter’s Algorithm

- Depth Sort
  - For back to front rendering
- Problems when polygons overlap in z
  - Can check overlap in x and y direction using bounding boxes
  - Order can be found if no such overlap
- If also overlaps in x and y
  - Split polygons

Z-buffer

- Store z in screen space during projection
- Interpolate 1/z during rasterization
- Find reciprocal to get correct them
- Check less than
- Then overwrite pixel if true