Chapter 9

Clipping
The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer
Problem:
Given a 2D line/polygon and a window, clip the line/polygon to their regions that are inside the window.

Objectives

- Efficiency
- Memory access
Analytic Solution

- **Intersection** of convex regions is convex
  - Why?
- $L$ & $D$ are *convex* - intersection is convex
  - single connected segment of $L$
- **Question**: Can boundary of two convex shapes intersect more than twice?
- Clipping - compute intersection of $L$ with four boundary segments of window $D$
Line-Line Intersection

Intersection: $x$ & $y$ values equal in both representations - two linear equations in two unknowns $(r,t)$

$$G_1 = \begin{cases} x^1(t) = x_0^1 + (x_1^1 - x_0^1)t \\ y^1(t) = y_0^1 + (y_1^1 - y_0^1)t \end{cases} \quad t \in [0,1] \quad \begin{cases} x^2(r) = x_0^2 + (x_1^2 - x_0^2)r \\ y^2(r) = y_0^2 + (y_1^2 - y_0^2)r \end{cases} \quad r \in [0,1]$$

$$x_0^1 + (x_1^1 - x_0^1)t = x_0^2 + (x_1^2 - x_0^2)r$$
$$y_0^1 + (y_1^1 - y_0^1)t = y_0^2 + (y_1^2 - y_0^2)r$$
Intersection with vertical/horizontal lines

Intersection: x & y values equal in both representations - two linear equations in two unknowns (r, t)

\[ G_1 = \begin{cases} 
  x^1(t) = x_0^1 + (x_1^1 - x_0^1)t \\ 
  y^1(t) = y_0^1 + (y_1^1 - y_0^1)t 
\end{cases} \quad t \in [0,1] 
\]

\[ G_2 = \begin{cases} 
  x^2(r) = x_0^2 \\ 
  y^2(r) = y_0^2 + (y_1^2 - y_0^2)r 
\end{cases} \quad r \in [0,1] 
\]

\[ x_0^1 + (x_1^1 - x_0^1)t = x_0^2 
\]

\[ t = \frac{x_0^2 - x_0^1}{x_1^1 - x_0^1} 
\]

\[ y_0^1 + (y_1^1 - y_0^1)t = y_0^2 + (y_1^2 - y_0^2)r 
\]
Cohen-Sutherland Algorithm

Purpose:
Fast treatment of line segments that are trivially inside/outside window.

\( P = (x,y) \) - point to be classified against window \( D \)

Idea: Assign to \( P \) a binary code consisting of a bit for each edge of \( D \), using lookup table:

<table>
<thead>
<tr>
<th>bit</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( y &lt; y_{\text{min}} )</td>
<td>( y \geq y_{\text{min}} )</td>
</tr>
<tr>
<td>2</td>
<td>( y &gt; y_{\text{max}} )</td>
<td>( y \leq y_{\text{max}} )</td>
</tr>
<tr>
<td>3</td>
<td>( x &gt; x_{\text{max}} )</td>
<td>( x \leq x_{\text{max}} )</td>
</tr>
<tr>
<td>4</td>
<td>( x &lt; x_{\text{min}} )</td>
<td>( x \geq x_{\text{min}} )</td>
</tr>
</tbody>
</table>
Given $L$ from $(x_0, y_0)$ to $(x_1, y_1)$ & rectangle D.

If bitwise **and** of the codes of $(x_0, y_0)$ and $(x_1, y_1)$ is not zero, or the bitwise **or** is zero, then $L$ can be trivially handled (it is either totally outside or totally inside D).

**Why?**
Cohen-Sutherland Algorithm (cont’d)

\( \text{C-S-Clip} \left( P_0 = (x_0, y_0), P_1 = (x_1, y_1), x_{\min}, x_{\max}, y_{\min}, y_{\max} \right) \)

\( C_0 \leftarrow \text{code} \left( P_0 \right); \quad C_1 \leftarrow \text{code} \left( P_1 \right); \)

if \((\left( C_0 \text{ and } C_1 \right) \neq 0)\) then return;

if \((\left( C_0 \text{ or } C_1 \right) = 0)\) then draw \((P_0, P_1)\);

else if \((\text{OutsideWindow} \left( P_0 \right))\) then
begin

Edge \leftarrow \text{Window boundary of leftmost non-zero bit of } C_0;

\( P_2 \leftarrow P_0, P_1 \cap \text{Edge}; \)

\( \text{C-S-Clip} \left( P_2, P_1, x_{\min}, x_{\max}, y_{\min}, y_{\max} \right); \)

end

else

Edge \leftarrow \text{Window boundary of leftmost non-zero bit of } C_1;

\( P_2 \leftarrow P_0, P_1 \cap \text{Edge}; \)

\( \text{C-S-Clip} \left( P_0, P_2, x_{\min}, x_{\max}, y_{\min}, y_{\max} \right); \)

end

<table>
<thead>
<tr>
<th>bit</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( y &lt; y_{\min} )</td>
<td>( y \geq y_{\min} )</td>
</tr>
<tr>
<td>2</td>
<td>( y &gt; y_{\max} )</td>
<td>( y \leq y_{\max} )</td>
</tr>
<tr>
<td>3</td>
<td>( x &gt; x_{\max} )</td>
<td>( x \leq x_{\max} )</td>
</tr>
<tr>
<td>4</td>
<td>( x &lt; x_{\min} )</td>
<td>( x \geq x_{\min} )</td>
</tr>
</tbody>
</table>
3D clipping

- Determine portion of line inside axis-aligned parallelepiped (viewing frustum in NDC)
- Simple extension to 2D algorithms
- After perspective transform
  - means that clipping volume always the same
    - xmin=ymin= -1, xmax=ymax= 1 in OpenGL
  - boundary lines become boundary planes
    - but bit-codes still work the same way
    - additional front and back clipping plane
      - zmin = -1, zmax = 1 in OpenGL
Triangle Clipping

- How does intersection of rectangle & triangle looks like?
  - How many sides?

- How to expand clipping to triangles?
  - Hint: it is convex
  - Will develop on the board…
Cohen-Sutherland Algorithm for convex polygons

\( \text{C-S-Clip} (\text{poly} = P_0, ..., P_n, x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}) \)

for \( i = 1 \) to \( n \) \( C_i \leftarrow \text{code}(P_i) \);
if \(( (C_0 \text{ and } C_1 \text{ and...and } C_n) \neq 0) \) then return;
if \(( (C_0 \text{ or } C_1 \text{ or...or } C_n) = 0) \) then draw(poly);
else
for \( i = 1 \) to \( n \) if \((\text{OutsideWindow}(P_i)) \) then
begin

\( \text{Edge} \leftarrow \text{Window boundary of leftmost non-zero bit of } C_i; \)
\( P_{i-1,i} \leftarrow P_{i-1} \cdot P_i \cap \text{Edge}; \)
\( P_{i,i+1} \leftarrow P_i \cdot P_{i+1} \cap \text{Edge}; \)
\( \text{C-S-Clip}(P_0, ..., P_{i-1}, P_{i-1,i}, P_{i,i+1}, P_{i+1}, ..., P_n, x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}) ; \)
end

<table>
<thead>
<tr>
<th>( \text{bit} )</th>
<th>( y &lt; y_{\text{min}} )</th>
<th>( y \geq y_{\text{min}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( y &gt; y_{\text{max}} )</td>
<td>( y \leq y_{\text{max}} )</td>
</tr>
<tr>
<td>2</td>
<td>( x &gt; x_{\text{max}} )</td>
<td>( x \leq x_{\text{max}} )</td>
</tr>
<tr>
<td>3</td>
<td>( x &lt; x_{\text{min}} )</td>
<td>( x \geq x_{\text{min}} )</td>
</tr>
</tbody>
</table>