Line and Polygon Clipping

Foley & Van Dam, Chapter 3
Topics

- Viewing Transformation Pipeline in 2D
- Line and polygon clipping
- Brute force analytic solution
- Cohen-Sutherland Line Clipping Algorithm
- Cyrus-Beck Line Clipping Algorithm
- Sutherland-Hodgman Polygon Clipping
- Sampling Theorem (Nyquist Frequency)
Viewing Transformation in 2D

World and Viewing Coordinates

Clipping

Device Coordinates

Normalized Device Coordinates
Viewing Transformation in 2D

- Objects are given in *world coordinates*
- The world is viewed through a *window*
- The window is mapped onto a *device viewport*

A scene in World-Coordinates $\rightarrow$ World-Coordinates to Viewing Coordinates $\rightarrow$ Viewing Coordinates to Normalized Device Coordinates $\rightarrow$ Normalized Device Coordinates to Device Coordinates

**Clipping**
Line and Polygon Clipping

The problem:
Given a set of 2D lines or polygons and a window, clip the lines or polygons to their regions that are inside the window.
Motivations

• Efficiency
• Display in portion of a screen
• Occlusions

clip rectangle
Line Clipping

• We will deal only with lines (segments)
• Our window can be described by two extreme points:
  \((x_{min},y_{min})\) and \((x_{max},y_{max})\)
• A point \((x,y)\) is in the window iff:
  \(x_{min} \leq x \leq x_{max} \quad \text{and} \quad y_{min} \leq y \leq y_{max}\)
Brute Force Analytic Solution

- The intersection of convex regions is always convex
- Since both $W$ and $S$ are convex, their intersection is convex, i.e., a single connected segment of $S$

**Question:** Can the boundary of two convex shapes intersect more than twice?

0, 1, or 2 intersections between a line and a window
Pseudo Code for Midpoint Line Drawing

\[ \text{Line}(x_0,y_0,x_1,y_1) \]
\[ \text{begin} \]
\[ \text{int } dx, dy, x, y, d, \Delta_E, \Delta_{NE}; \]
\[ x := x_0; \quad y = y_0; \]
\[ dx := x_1 - x_0; \quad dy := y_1 - y_0; \]
\[ d := 2*dy - dx; \]
\[ \Delta_E := 2*dy; \quad \Delta_{NE} := 2*(dy - dx); \]
\[ \text{PlotPixel}(x,y); \]
\[ \text{while}(x < x_1) \text{ do} \]
\[ \quad \text{if } (d < 0) \text{ then} \]
\[ \quad \quad d := d + \Delta_E; \]
\[ \quad \quad x := x + 1; \]
\[ \quad \text{end;} \]
\[ \quad \text{else} \]
\[ \quad \quad d := d + \Delta_{NE}; \]
\[ \quad \quad x := x + 1; \]
\[ \quad \quad y := y + 1; \]
\[ \quad \text{end;} \]
\[ \quad \text{PlotPixel}(x,y); \]
\[ \text{end;} \]
\[ \text{end;} \]

Assume \( x_1 > x_0 \) and \( 0 < \text{slope} \leq 1 \)
Line Clipping

Midpoint Algorithm: Intersection with a vertical edge

\[ x = x_{\text{min}} \]

\[ (x_{\text{min}}, \text{Round}(mx_{\text{min}} + B)) \]

\[ (x_{\text{min}}, mx_{\text{min}} + B) \]

\[ y = y_{\text{min}} \]
Line Clipping

Midpoint Algorithm: Intersection with a horizontal edge

\[ x = x_{\text{min}} \]

\[ y = y_{\text{min}} \]

\[ y = y_{\text{min}} - 1/2 \]
Cohen-Sutherland for Line Clipping

- Clipping is performed by computing intersections with four boundary segments of the window: $L_i, \ i=1,2,3,4$

- **Purpose**: Fast treatment of lines that are trivially inside/outside the window

- Let $P=(x,y)$ be a point to be classified against window $W$

- **Idea**: Assign $P$ a binary code consisting of a bit for each edge of $W$. The bit is 1 if the pixel is in the half-plane that does not contain $W$. 
Cohen-Sutherland for Line Clipping

<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>

The image shows a table with the conditions for clipping lines based on their positions relative to the clipping areas defined by $y_{\text{min}}$, $y_{\text{max}}$, $x_{\text{min}}$, and $x_{\text{max}}$.
Cohen-Sutherland for Line Clipping

<table>
<thead>
<tr>
<th>Code</th>
<th>Code</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101</td>
<td>0100</td>
<td>0110</td>
</tr>
<tr>
<td>0001</td>
<td>0000</td>
<td>0010</td>
</tr>
<tr>
<td>1001</td>
<td>1000</td>
<td>1010</td>
</tr>
</tbody>
</table>

Given a line segment $S$ from $p_0=(x_0,y_0)$ to $p_1=(x_1,y_1)$ to be clipped against a window $W$.

If $\text{code}(p_0) \ AND \ \text{code}(p_1)$ is not zero, then $S$ is \textit{trivially rejected}.

If $\text{code}(p_0) \ OR \ \text{code}(p_1)$ is zero, then $S$ is \textit{trivially accepted}.
Cohen-Sutherland for Line Clipping

Otherwise: let assume w.l.o.g. that $p_0$ is outside $W$

- Find the intersection of $S$ with the edge corresponding to the MSB in $\text{code}(p_0)$ that is equal to 1. Call the intersection point $p_2$.
- Run the procedure for the new segment $(p_1, p_2)$. 
Inside/Outside Test:

• Assume WLOG that $V=(V_1-V_0)$ is the border vector where "inside" is to its right

• If $V=(V_x,V_y)$, $N$ is the normal to $V$, pointing outside, defined by $N=(-V_y,V_x)$

• Vector $U$ points "outside" if $N \cdot U > 0$

• Otherwise $U$ points "inside"
The parametric line $P(t) = P_0 + (P_1 - P_0)t$
The parametric vector $V(t) = P(t) - Q$

The segment $P_0P_1$ intersects the line $L$ at $t_0$ satisfying $V(t_0)\cdot N = 0$

The intersection point is $P(t_0)$

$\Delta = P_1 - P_0$ points inside if $(P_1 - P_0) \cdot N < 0$. Otherwise it points outside

If $L$ is vertical, intersection can be computed using the explicit equation
Cyrus-Beck Line Clipping

• Denote \( p(t)=p_0+(p_1-p_0)t \) \( t\in[0..1] \)
• Let \( Q_i \) be a point on the edge \( L_i \) with outside pointing normal \( N_i \)
• \( V(t) = p(t)-Q_i \) is a parameterized vector from \( Q_i \) to the segment \( P(t) \)
• \( N_i \cdot V(t) = 0 \) iff \( V(t) \perp N_i \)
• We are looking for \( t \) satisfying \( N_i \cdot V(t) = 0 \)
Cyrus-Beck Line Clipping

\[ 0 = N_i \cdot V(t) = N_i \cdot (p(t) - Q_i) = N_i \cdot (p_0 + (p_1 - p_0)t - Q_i) = N_i \cdot (p_0 - Q_i) + N_i \cdot (p_1 - p_0)t \]

Solving for \( t \) we get:

\[ t = \frac{N_i \cdot (p_0 - Q_i)}{-N_i \cdot (p_1 - p_0)} = \frac{N_i \cdot (p_0 - Q_i)}{-N_i \cdot \Delta} \]

where \( \Delta = (p_1 - p_0) \)

Comment: If \( N_i \cdot \Delta = 0 \), \( t \) has no solution \((V(t) \perp N_i)\)
Cyrus-Beck Line Clipping

Diagram showing the Cyrus-Beck Line Clipping algorithm with points and lines.
Cyrus-Beck Line Clipping

- The intersection of $p(t)$ with all four edges $L_i$ is computed, resulting in up to four $t_i$ values.
- If $t_i < 0$ or $t_i > 1$, $t_i$ can be discarded.
- Based on the sign of $N_i \cdot \Delta$, each intersection point is classified as $PE$ (potentially entering) or $PL$ (potentially leaving).
- $PE$ with the largest $t$ and $PL$ with the smallest $t$ provide the domain of $p(t)$ inside $W$.
- The domain, if inverted, signals that $p(t)$ is totally outside.
Sutherland-Hodgman Polygon-Clipping Algorithm
Sutherland-Hodgman Polygon-Clipping Algorithm

Idea: Clip a polygon by successively clipping against each (infinite) clip edge

After each clipping a new set of vertices is produced.
Sutherland-Hodgman Polygon-Clipping Algorithm

For each clip edge - scan the polygon and consider the relation between successive vertices of the polygon.

Each iteration adds 0, 1 or 2 new vertices.

Assume vertex s has been dealt with, vertex p follows:

- p added to output list
- i added to output list
- no output
- i and p added to output list
Sutherland-Hodgman Polygon-Clipping Algorithm

Left Clipping

Right Clipping

Bottom Clipping

Top Clipping

V_1  V_2  V_3  →  V_1  V_2  V'_2  V'_3  →  V_1  V_2  V'_2  V'_3  →  V'_1  V'_2  V''_2
**Sampling Theorem**

**Question:** How dense should be the pixel grid in order to draw properly a drawn object?

Given a sampling at intervals equal to $d$ then one may recover frequencies of wavelength $> 2d$

**Aliasing:** If the sampling interval is more than $1/2$ the wavelength, erroneous frequencies may be produced
Sampling Theorem

1D Example:

Rule of Thumb: To observe details of size $d$ one must sample at $d/2$ intervals

To observe details at frequency $f$ ($=1/d$) one must sample at frequency $2f$. The Frequency $2f$ is the NYQUIST frequency
Sampling Theorem

2D Example: Moire’ Effect