Line and Polygon Clipping

- Review of line clipping algorithm
  - Brute-force method
  - Outcode method
    (Cohen-Sutherland algorithm)
  - Parametric method
    (Cyrus-Beck algorithm)
• Review of polygon clipping algorithm
  – Divide-and-Conquer method
    (Sutherland-Hodgman algorithm)
  – Parametric method
    (Liang-Barsky algorithm)
  – Clipping polygon against polygon
    (Weiler algorithm)

**Note:**
We will focus on the introduction of the advanced parametric polygon clipping approach
(i.e., Liang-Barsky algorithm)
1. Line Clipping Methods

- Brute force method:
  Solving the simultaneous equations for each (edge, line) pair. (not efficient)

- Region outcodes method:
  (1) trivial accept and trivial reject
  (2) For the non-trivial lines, the clipping order is: top-bottom-right-left

Outcode: XXXX = top-bottom-right-left
1. Line Clipping Methods (cont’d)

- Parametric method:

  Parametric representation of a line:

  \[ P(t) = P_0 + (P_1 - P_0)t \]

  where \( t \in [0, 1] \)

  \[ t = 0 \text{ at } P_0 \]
  \[ t = 1 \text{ at } P_1. \]

  **Task:** Find a \( t \) at the intersection point of the line and the edge.
- Parametric method (cont’d)

Denote: \( \text{DOT} = N_i \cdot [P(t) - P_{E_i}] \)

Dot product:

\[
N_i \cdot [P(t) - P_{E_i}] = 0
\]

where \( P(t) = P_0 + (P_1 - P_0)t \)

\[
\Rightarrow t = \frac{N_i \cdot [P_0 - P_{E_i}]}{-N_i \cdot D}, \quad \text{where} \quad D = (P_1 - P_0)
\]
Task:
(1) Any value of $t$ outside the interval $[0, 1]$ can be discarded.
(2) Determine whether the intersection lies on the clip boundary.
• Parametric method (cont’d)

Method:
(1) \( PE \): Potential Enter;  
(2) \( PL \): Potential Leave;

(3) Determine the clipped line by \((PE, PL)\) pair and

(4) Condition: \( t_E < t_L \)

• Summary of line clipping:

(1) Outcode method is good at dealing with the trivial acceptance or rejection  
(2) Parametric method is good at dealing with the case where many line segments need to be clipped
2. Polygon Clipping Methods

- Divide-and-conquer polygon clipping:

  Clipping polygon against four edges in succession.

  Clipping against a clip edge
  → output a series of new vertices
  → partially clipped polygon is clipped against the second clip edge.
Case Study:

- **Window inside the polygon**

- **Window outside the polygon**
3. Liang-Barsky Polygon Algorithm
   – An efficient and fast parametric method

   • Regions definition

     • Corner region
     • Window region
     • Edge region

   If we divide the plane into two regions along the clip window edge, the region containing
   the window will be added 1.
3. Liang-Barsky Polygon Algorithm (Cont’d)

- Potential in/out points

\[ P(t) = (1 - t)P_0 + tP_1 \]
Where \( t = 0 \rightarrow P_0 \), \( t = 1 \rightarrow P_1 \)

\[ N_i \cdot \overline{P_0P_1} < 0 \Rightarrow \text{potential “in” } (t_{in}^1, t_{in}^2) \]
\[ N_i \cdot \overline{P_0P_1} > 0 \Rightarrow \text{potential “out” } (t_{out}^1, t_{out}^2) \]

\( t_{in} \): from the region in small number to the region in large number

\( t_{out} \): from the region in large number to the region in small number
3. Liang-Barsky Polygon Algorithm (Cont’d)

• Clipping edge

If \( t_{in}^2 < t_{out}^1 \) ⇒ line segment intersecting the window edge is clipped.
\( (t_{in}^2 \leq 1 \text{ and } t_{out}^1 > 0) \)

If \( t_{in}^2 > t_{out}^1 \) ⇒ line passes by a corner region without intersecting the window clip edge.
3. Liang-Barsky Polygon Algorithm (Cont’d)

• Corner vertex

If the edge line enters the corner region, the corresponding corner vertex must be added as the output vertex, we call the corner vertex “turning vertex”

If last edge ended in region 3, the next edge could only hit one possible side of the window (e.g., top side)

If the last edge ended in region 2 (corner region), the next edge could hit two possible sides (e.g., top or leftside of the window)
3. Liang-Barsky Polygon Algorithm (Cont’d)

- **Adding corner vertex**
  In the case of corner region, the turning vertex must be added as the output.
3. Liang-Barsky Polygon Algorithm (Cont’d)

Condition of adding turning vertex:

\[ 0 < t_{out}^1 \leq 1 \quad \text{or} \quad 0 < t_{out}^2 \leq 1 \] and entering the intermediate corner region

![Diagram showing turning vertex to be added]