Computer Graphics

Chapter 8 (II)
Two - Dimensional Viewing(Clipping)

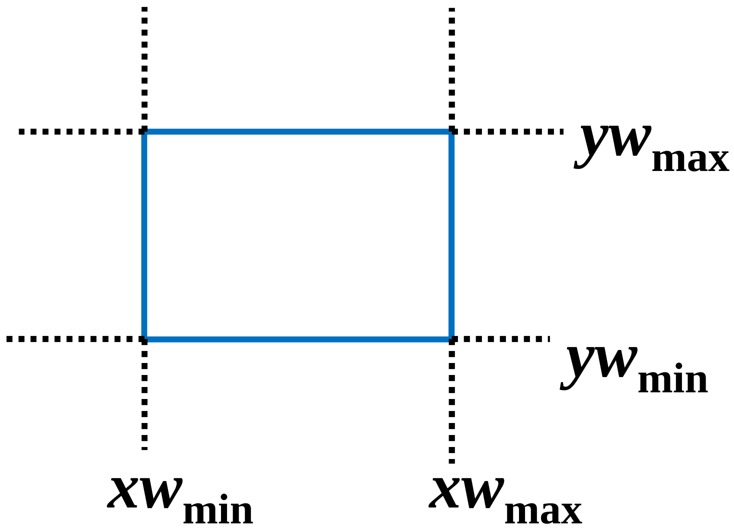
Outline

- 2D Point Clipping
- 2D Line Clipping
- Polygon Fill-Area Clipping
- Text Clipping

2D Point Clipping

- For a clipping rectangle, which is defined by (xw_{min}, yw_{min}) and (xw_{max}, yw_{max})

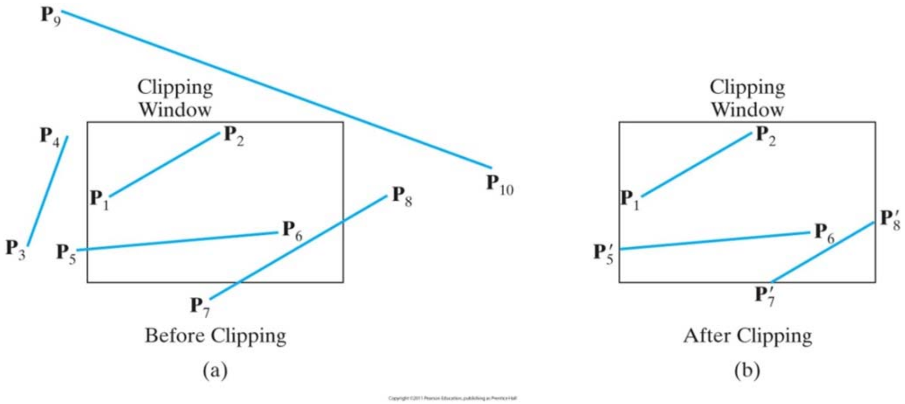
$$xw_{min} \le x \le xw_{max}$$
$$yw_{min} \le y \le yw_{max} \qquad (8\text{-}12)$$



2D Line Clipping

- Basic process
  - Each line goes through the tests and intersection calculations

Expensive part



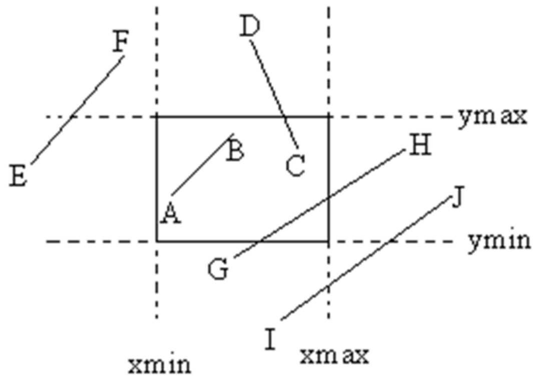
- Optimization
  - Minimize the intersection calculations
    - Special cases: which are completely inside or outside a clipping window.
    - Then, general: determine the intersections

Parametric representation:

$$\begin{cases} x = x_1 + u(x_2 - x_1) \\ y = y_1 + u(y_2 - y_1), \qquad 0 \le u \le 1 \end{cases}$$

2D Line Clipping

- *Cohen - Sutherland* Line Clipping Method



- Relationship between the line segment and the clipping window
  - Completely inside;
  - Completely outside;
  - Other cases

Cohen - Sutherland Line Clipping

- The line endpoint is encoded by a region code.

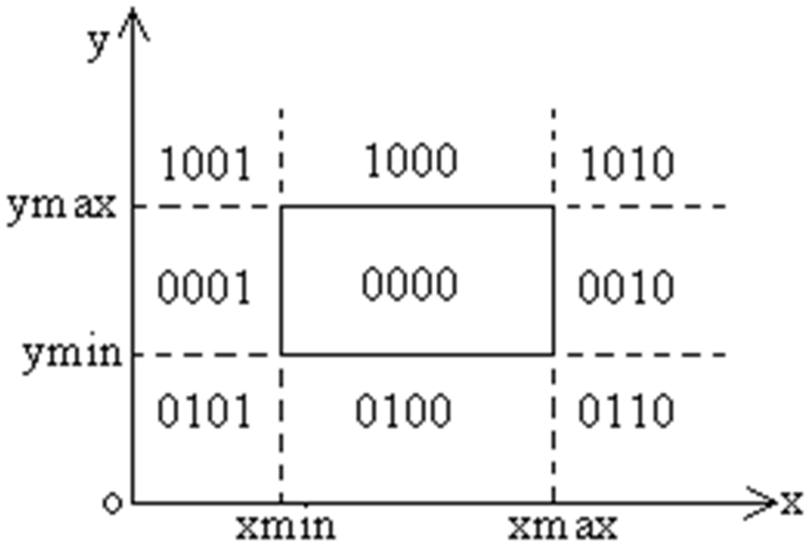
  The four clipping-window edges divide 2D space into 9 regions, each of which corresponds to 4-bit code: $C_t C_b C_r C_l$

$$C_t = \begin{cases} 1 & \text{if } y > ymax \\ 0 & \text{else} \end{cases}$$

$$C_b = \begin{cases} 1 & \text{if } y < ymin \\ 0 & \text{else} \end{cases}$$

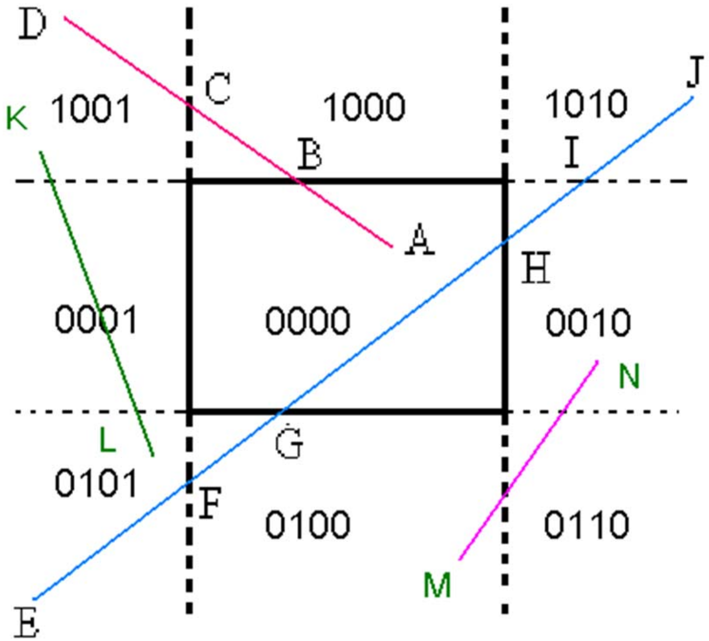
$$C_r = \begin{cases} 1 & \text{if } x > xmax \\ 0 & \text{else} \end{cases}$$

$$C_l = \begin{cases} 1 & \text{if } x < xmin \\ 0 & \text{else} \end{cases}$$



Cohen - Sutherland Line Clipping

- Endpoint encoding – the region code which it belongs to.
- Conclusion:
  - When the logic 'and' operation of two endpoints is nonzero, the line segment is completely outside (obviously invisible);
  - When the region code 0000 for both endpoints, the line segment is completely inside.



A : 0000 E : 0101

D : 1001 J : 1010

K : 1001 M : 0100

L : 0101 N : 0010

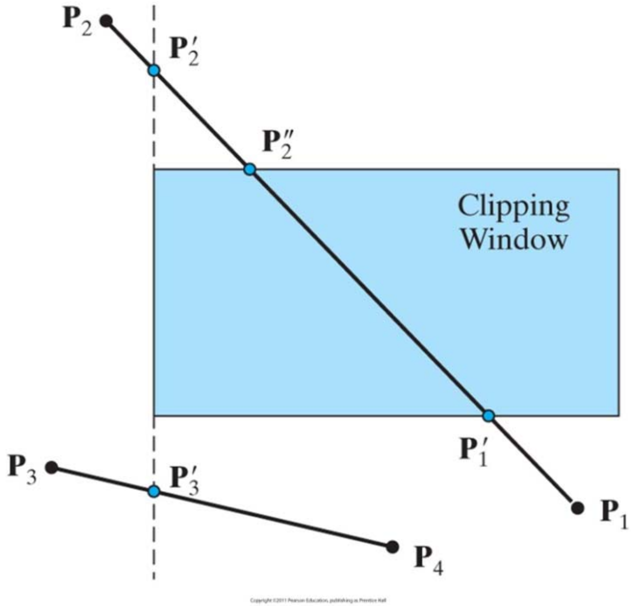
Cohen - Sutherland Line Clipping

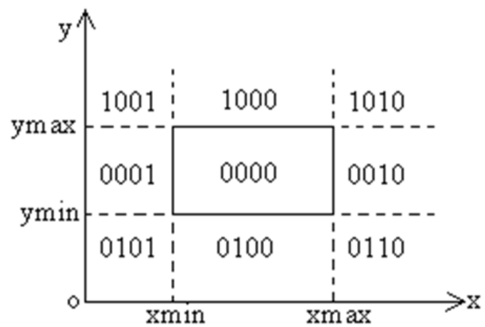
- *Cohen - Sutherland* Line Clipping
  - Produce a **region code** for each endpoint of the line segment, e.g. *c1* and *c2* for both endpoints
  - Check the region codes of the line segment:
    - If both c1 and c2 are '0000's, the line segment is within the clip window; (the inside case)
    - If (c1 & c2) ≠ '0000' the line segment is fully outside the clip window; (the outside case)
    - Otherwise, a next check is made for the intersection of the line with the window boundaries and the intersection are calculated. (HowTO: the next slide)

Cohen - Sutherland Line Clipping

- Lines that cannot be identified as being completely inside or outside the clipping window, to process : L -> R -> B -> T
- P1P2: P1(0100), P2(1001) – **left** boundary
  - Calculate the intersection P2';
  - Clip off P2P2';

    The remain is inside the right border line (the 2nd-bit are same, no need to check "R");
- Next, check the **bottom** border line.
  - P1 is below, P2' is above it; [P1(0100), P2(1001)]
  - Calculate the intersection P1';
  - Clip off P1P1';
- Next, check the **top** border line; [P1(0100), P2(1001)]
  - Get the intersection P2";
  - Clip off P2'P2".



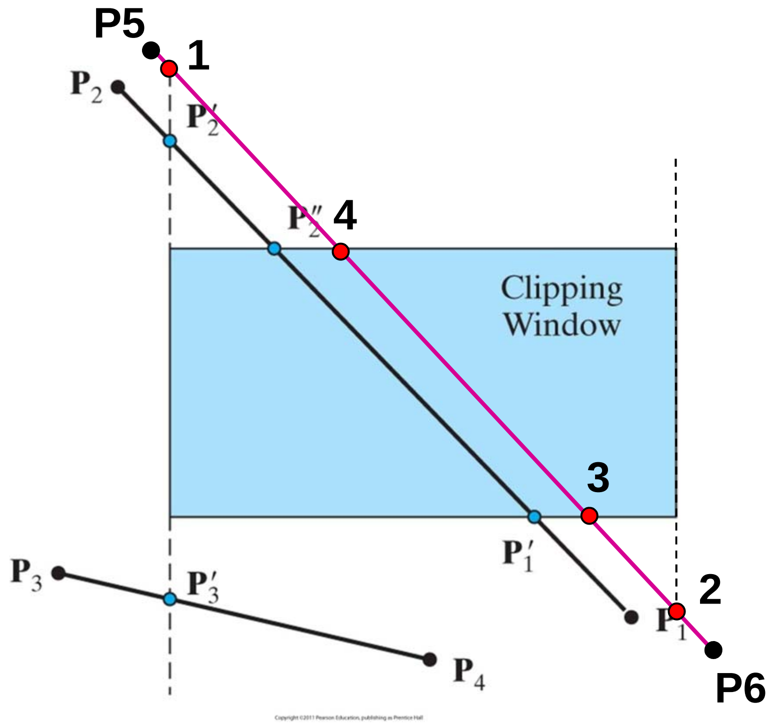


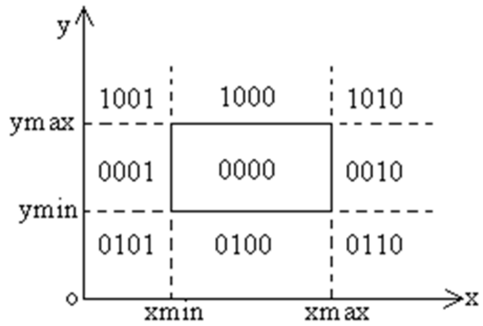
Cohen - Sutherland Line Clipping

- P3P4:P3(0101), P4(0100)
  - Calculate the intersection P3';
  - Clip off P3P3';

  The remain is inside the right border line;
  - Next, check the **bottom** border line;
  - The region codes show that all is below the clipping window.
  - Clip off P3'P4.
- P5P6: P5(1001), P6(0110)
  - Four intersection positions should be calculated in the order of L, R, B, T.

  Variations of this basic approach have been developed





Cohen - Sutherland Line Clipping

- To effectively calculate the intersections
  - Using the **slope equation** of the line
  - For line (x_0, y_0) and (x_{end}, y_{end})
  - y coordinate of the intersection point with a vertical clipping border line:

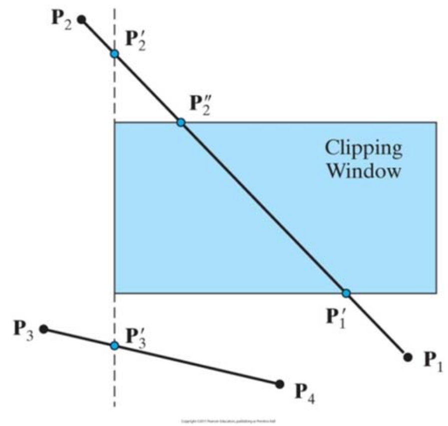
    xw_{min} or xw_{max}

    $y = y_0 + m\,(x - x_0)$,

    where $m=(y_{end}-y_0)/(x_{end}-x_0)$
  - x coordinate of the intersection point with a horizontal border line:

    yw_{min} or yw_{max}

    $x = x_0 + (y - y_0)/m.$

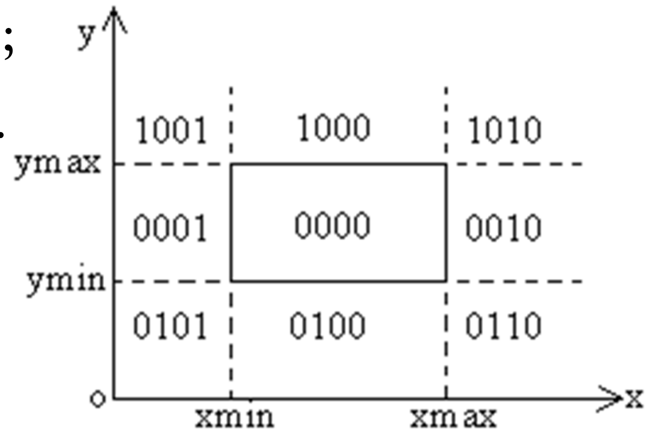


Cohen - Sutherland Line Clipping

- To efficiently determine the region code

  Comparing the coordinate values of an endpoint to the clipping boundaries and using **the sign bit of the result**, instead of using inequality testing.

  - bit 1: left – the sign bit of ($x - xw_{min}$);
  - bit 2: right – the sign bit of ($xw_{max} - x$);
  - bit 3: below – the sign bit of ($y - yw_{min}$);
  - bit 4: above – the sign bit of ($yw_{max} - y$).



/* 2D *Cohen-Sutherland* Program Code */

(See it in the textbook P279)

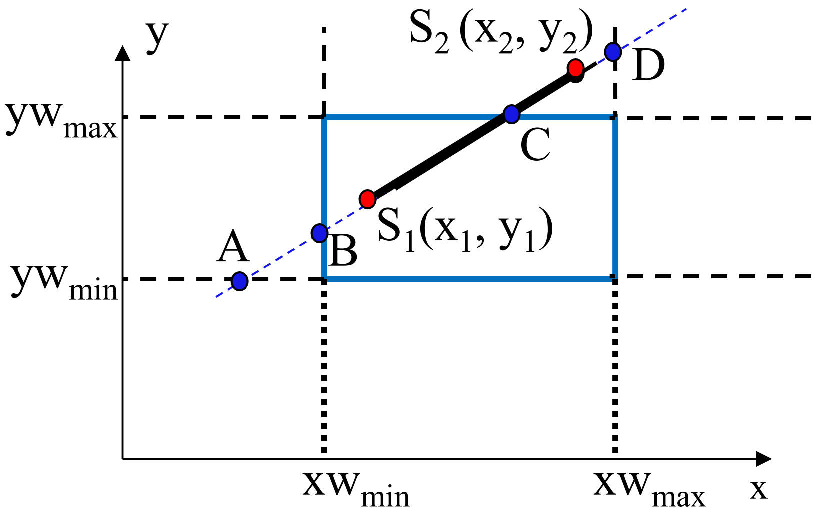
Liang-Barsky Line Clipping

A faster line clipping algorithm: based on the parametric line equations.

Condition: Given a line segment S_1S_2.

A, B, C, D are the intersection points with the clipping window edges.

Key idea: to find the nearest point to S_2 from **A**, **B** and $\mathbf{S_1}$ (S_1 in the Fig); to find the nearest point to S_1 from **C**, **D** and $\mathbf{S_2}$ (C in the Fig). S_1C is the visible part of S_1S_2.



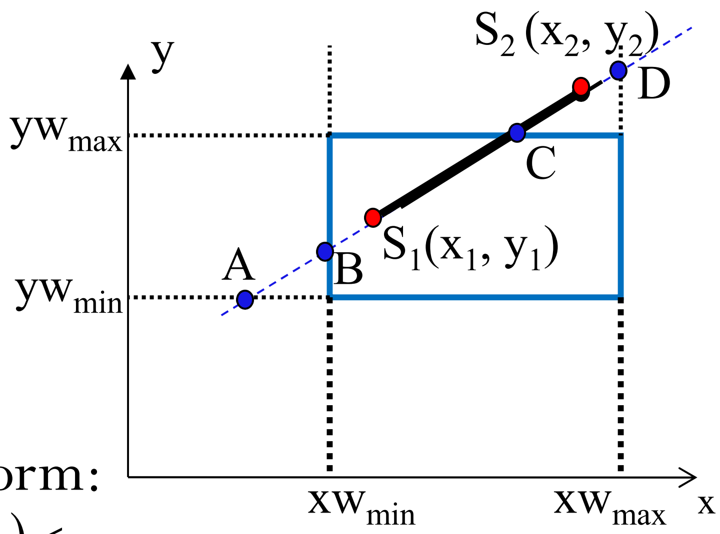
$$\begin{cases} x = x_1 + u\Delta x \\ y = y_1 + u\Delta y \end{cases} \qquad u \in [0,1]$$

$$\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$$

Liang-Barsky Line Clipping

- Line segment in parametric form:

$$\begin{cases} x = x_1 + u\Delta x & \Delta x = x_2 - x_1 \\ y = y_1 + u\Delta y & \Delta y = y_2 - y_1, \quad 0 \le u \le 1 \end{cases} \qquad (8\text{-}16)$$



Point-clipping condition in parametric form:

$$\begin{cases} xw_{min} \le x_1 + u\Delta x \le xw_{max} \\ yw_{min} \le y_1 + u\Delta y \le yw_{max} \end{cases} \Longrightarrow \begin{cases} u(-\Delta x) \le x_1 - xw_{min} \\ u\Delta x \le xw_{max} - x_1 \end{cases} \quad \begin{cases} u(-\Delta y) \le y_1 - yw_{min} \\ u(\Delta y) \le yw_{max} - y_1 \end{cases}$$

in another form

$$up_k \le q_k\,, \qquad k = 1,2,3,4$$

$$\begin{aligned} p_1 &= -\Delta x & q_1 &= x_1 - xw_{min} \\ p_2 &= \Delta x & q_2 &= xw_{max} - x_1 \\ p_3 &= -\Delta y & q_3 &= y_1 - yw_{min} \\ p_4 &= \Delta y & q_4 &= yw_{max} - y_1 \end{aligned}$$

Liang-Barsky Line Clipping

$$\begin{cases} x = x_1 + u\Delta x \\ y = y_1 + u\Delta y \end{cases} \qquad u \in [0,1]$$

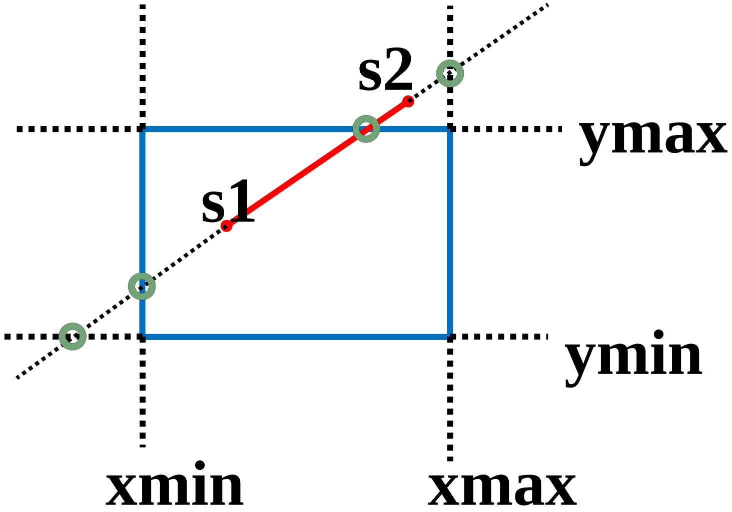
$$\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$$

Decide the direction:

$$u(-\Delta x) \le x_1 - xw_{\min}$$
$$u\Delta x \le xw_{\max} - x_1$$
$$u(-\Delta y) \le y_1 - yw_{\min}$$
$$u(\Delta y) \le yw_{\max} - y_1$$

$$up_k \le q_k \,, \qquad k = 1,2,3,4$$

$$p_1 = -\Delta x \qquad q_1 = x_1 - xw_{\min}$$
$$p_2 = \Delta x \qquad q_2 = xw_{\max} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - yw_{\min}$$
$$p_4 = \Delta y \qquad q_4 = yw_{\max} - y_1$$



Definition	**Starting** edge	**Ending** edge
Δx>=0	xmin	xmax
Δx<0	xmax	xmin
Δy>=0	ymin	ymax
Δy<0	ymax	ymin

- Here, **Δx>0 and Δy>0**

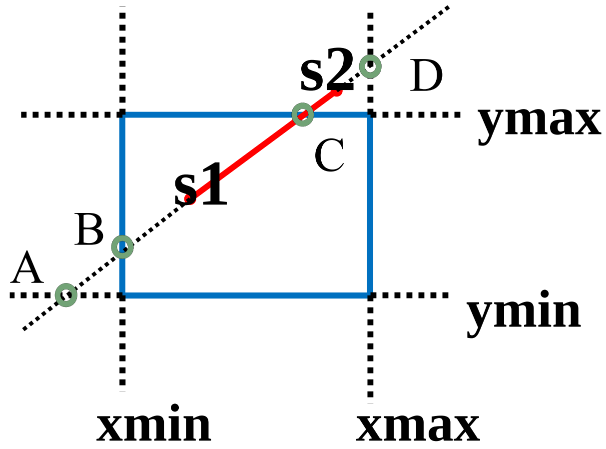
  so, xmin and ymin are the starting edge;

Liang-Barsky Line Clipping

$$\begin{cases} x = x_1 + u\Delta x \\ y = y_1 + u\Delta y \end{cases} \quad u \in [0,1]$$

$$\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$$

Start the test :



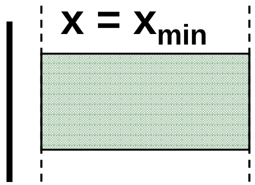
<u>Special cases</u>

- If p_k=0 (k=1,2)， the line segment is **parallel** to the clipping edge, then

  a. If q_k<0 (k=1 or 2), the line is completely outside of the clipping region, so discard it;

  b. If q_k>0 (k=1 or 2), line segment is possible visible, get the intersections with the window edges.

  (Similarly, p_k=0 (k=3,4))



$$up_k \le q_k\,, \qquad k = 1,2,3,4$$

$$p_1 = -\Delta x \qquad q_1 = x_1 - xw_{min}$$

$$p_2 = \Delta x \qquad q_2 = xw_{max} - x_1$$

$$p_3 = -\Delta y \qquad q_3 = y_1 - yw_{min}$$

$$p_4 = \Delta y \qquad q_4 = yw_{max} - y_1$$

Liang-Barsky Line Clipping

Start the test (cont.):

<u>general cases</u>

- The parameters of the intersections of S_1S_2 with the **starting** edge are referred as u1', u1"

  u1 = **max**{u1', u1", 0} – the nearest clipping point to S_2

- The parameters of the intersections of S_1S_2 with two **ending** edges are referred as u2',u2"

  u2 = **min**{u2', u2", 1} – the nearest clipping point to S_1

<u>**Conclusion:**</u>

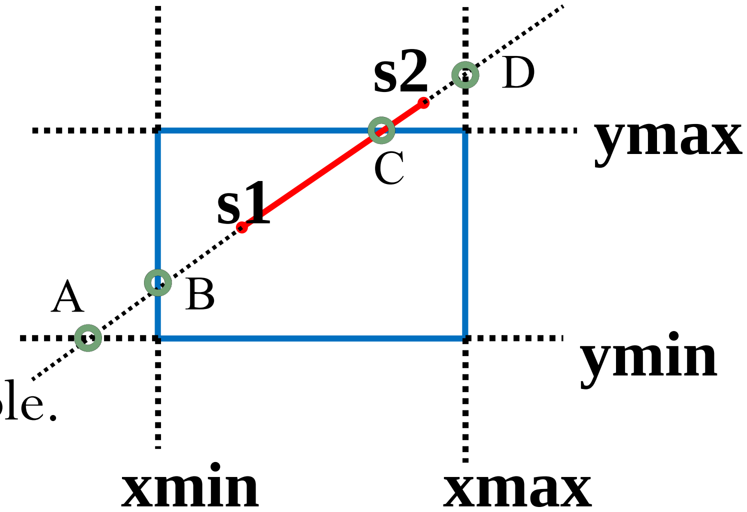
1. If u1<u2 , $x = x1 + \Delta x * u$

   $y = y1 + \Delta y * u$

   (u1 <u <u2)

   they define the visible part.

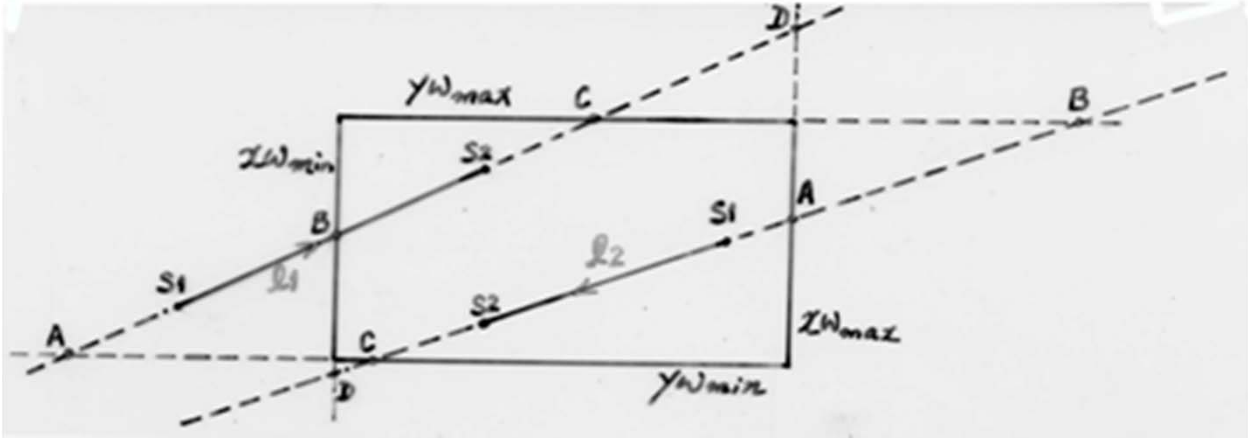
2. If u1>u2 , the line is invisible.



2D Line Clipping

	Starting edge	Ending edge
Δx>=0	xmin	xmax
Δx<0	xmax	xmin
Δy>=0	ymin	ymax
Δy<0	ymax	ymin

- Liang-Barsky Line Clipping example 1



l1: **Δx>0 and Δy>0,** starting edge: xwmin and ywmin (A, B);

l2: **Δx<0 and Δy<0,** starting edge: xwmax and ywmax (A, B);

1. u1’, u1” refer to intersections of S_1S_2 with two ***starting*** edges

   u1=max{u1’, u1”, 0} – the nearest clipping point to S_2

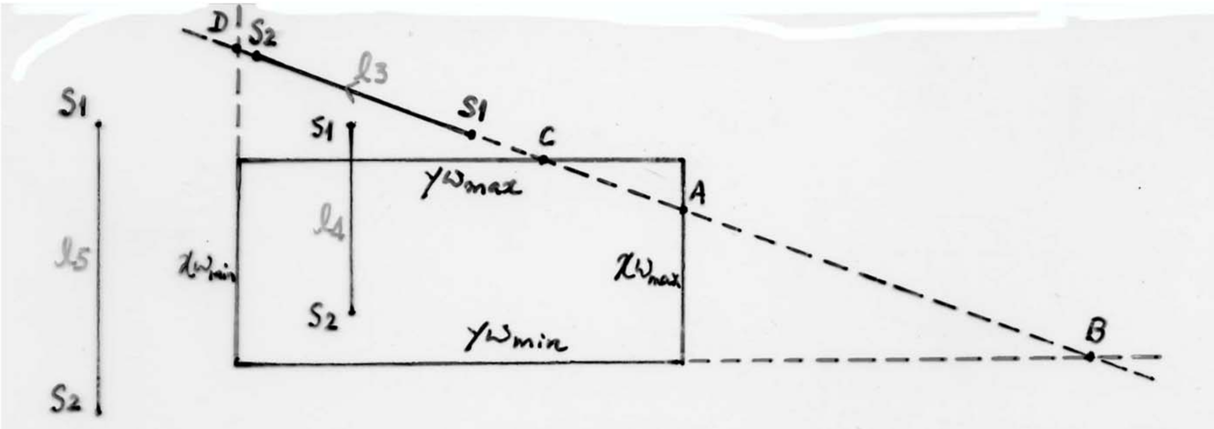
2. u2’,u2” refer to intersections of S_1S_2 with two ***ending*** edges

   u2=min{u2’, u2”, 1} – the nearest clipping point to S_1

2D Line Clipping

- Liang-Barsky Line Clipping example 2

	Starting edge	Ending edge
Δx>=0	xmin	xmax
Δx<0	xmax	xmin
Δy>=0	ymin	ymax
Δy<0	ymax	ymin



$$up_k \le q_k, k = 1,2,3,4$$

$$p_1 = -\Delta x, \quad q_1 = x_1 - xw_{\min}$$
$$p_2 = \Delta x, \quad q_2 = xw_{\max} - x_1$$
$$p_3 = -\Delta y, \quad q_3 = y_1 - yw_{\min}$$
$$p_4 = \Delta y, \quad q_4 = yw_{\max} - y_1$$

I_3: $\Delta x < 0$, $\Delta y > 0$, starting edges are xwmax and ywmin (A, B)

$u_1 = u_{s1} = 0$, $u_2 = u_C < 0$

$\because u_1 > u_2$, $\therefore S_1 S_2$ invisible.

I_4 : $p_1 = p_2 = 0$ but $q_1 > 0$, $q_2 > 0$ $\therefore$ partly visible possibly;

I_5 : $p_1 = p_2 = 0$, $q_1 < 0$, invisible.

Polygon Fill-Area Clipping

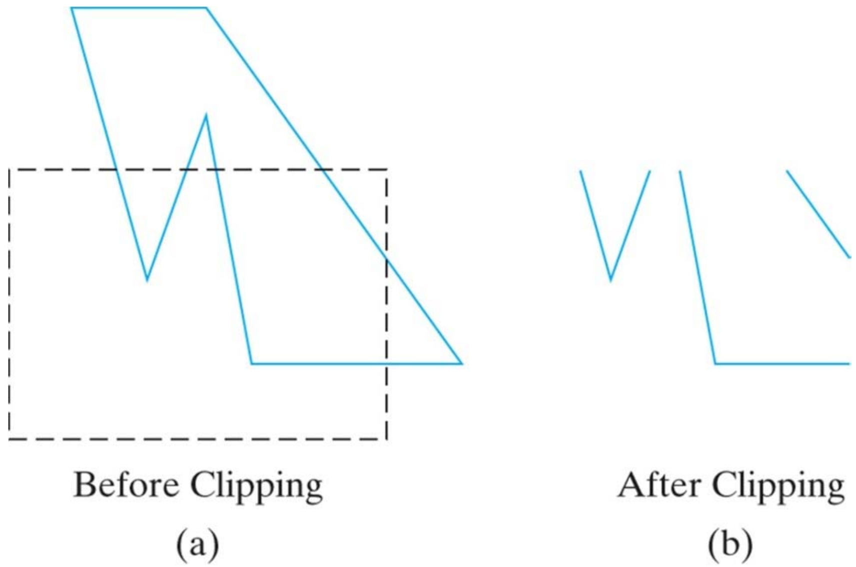


Figure 8-19 A line-clipping algorithm applied to the line segments of the polygon boundary in (a) generates the unconnected set of lines in (b).

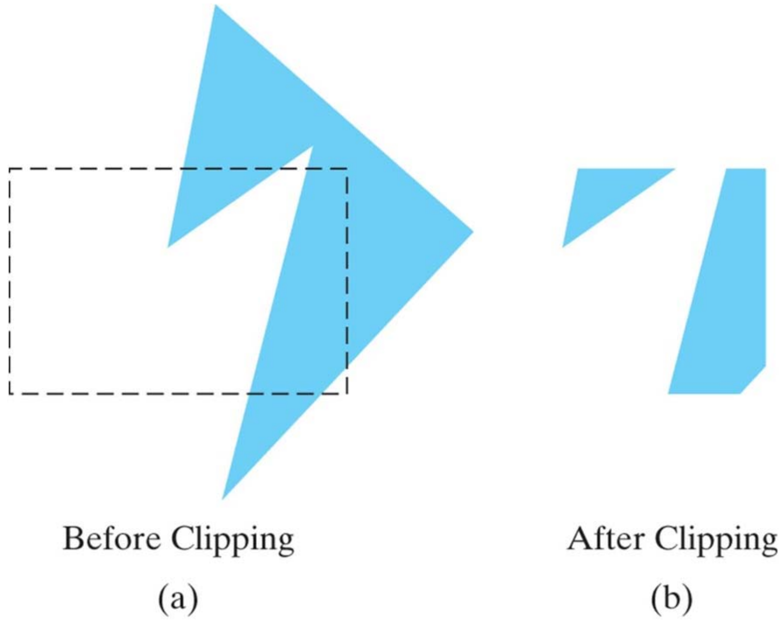


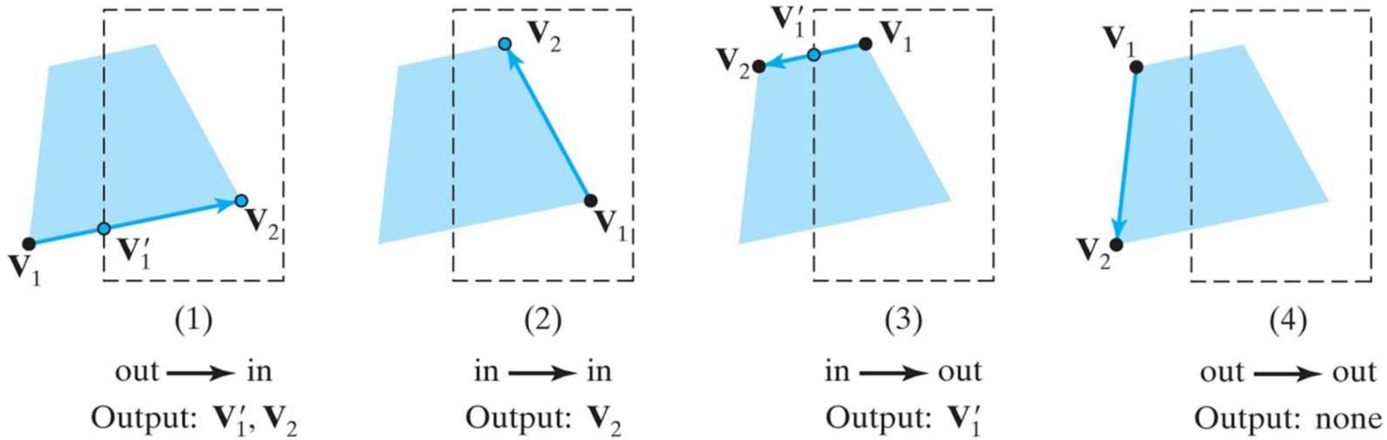
Figure 8-20 Display of a correctly clipped polygon.

Sutherland-Hodgman Polygon Clipping

Intersections of the polygon edge V_1V_2 with the window edge L:

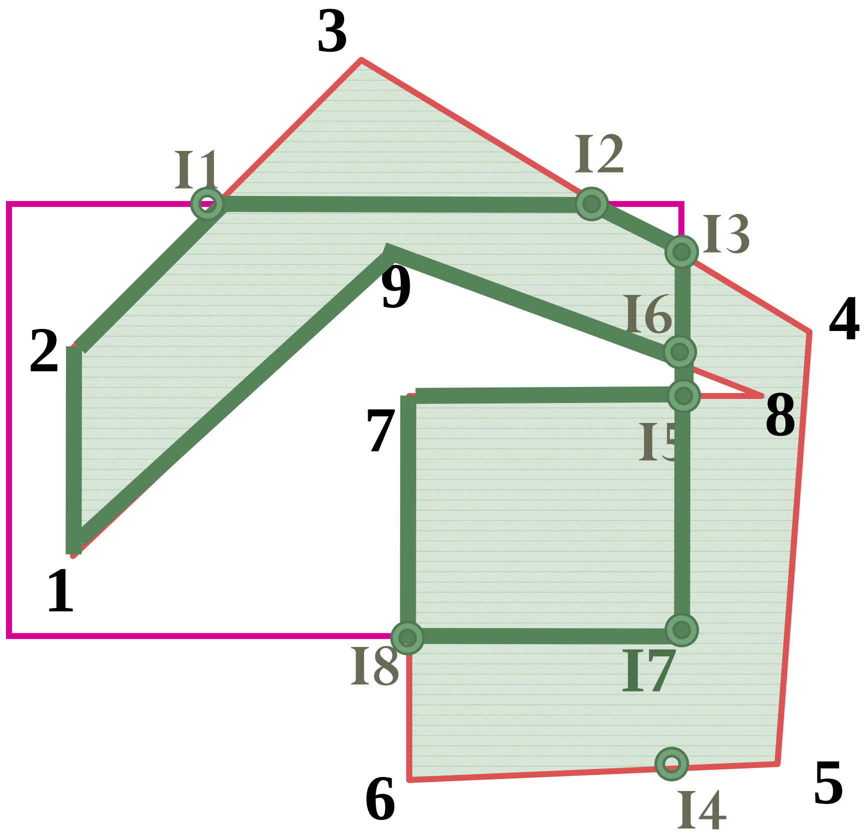
1. If V_1, V_2 lies to the same side of L, their visibility is same and without intersection.

2. If V_1, V_2 lies to the different side of L, their visibility is different and with intersection.



Sutherland-Hodgman Polygon Clipping

Example1: A polygon 1,2,3,4,5,6,7,8,9



After clipping by xmin

1,2,3,4,5,6,7,8,9

After clipping by ymax

1,2,I1,I2,4,5,6,7,8,9

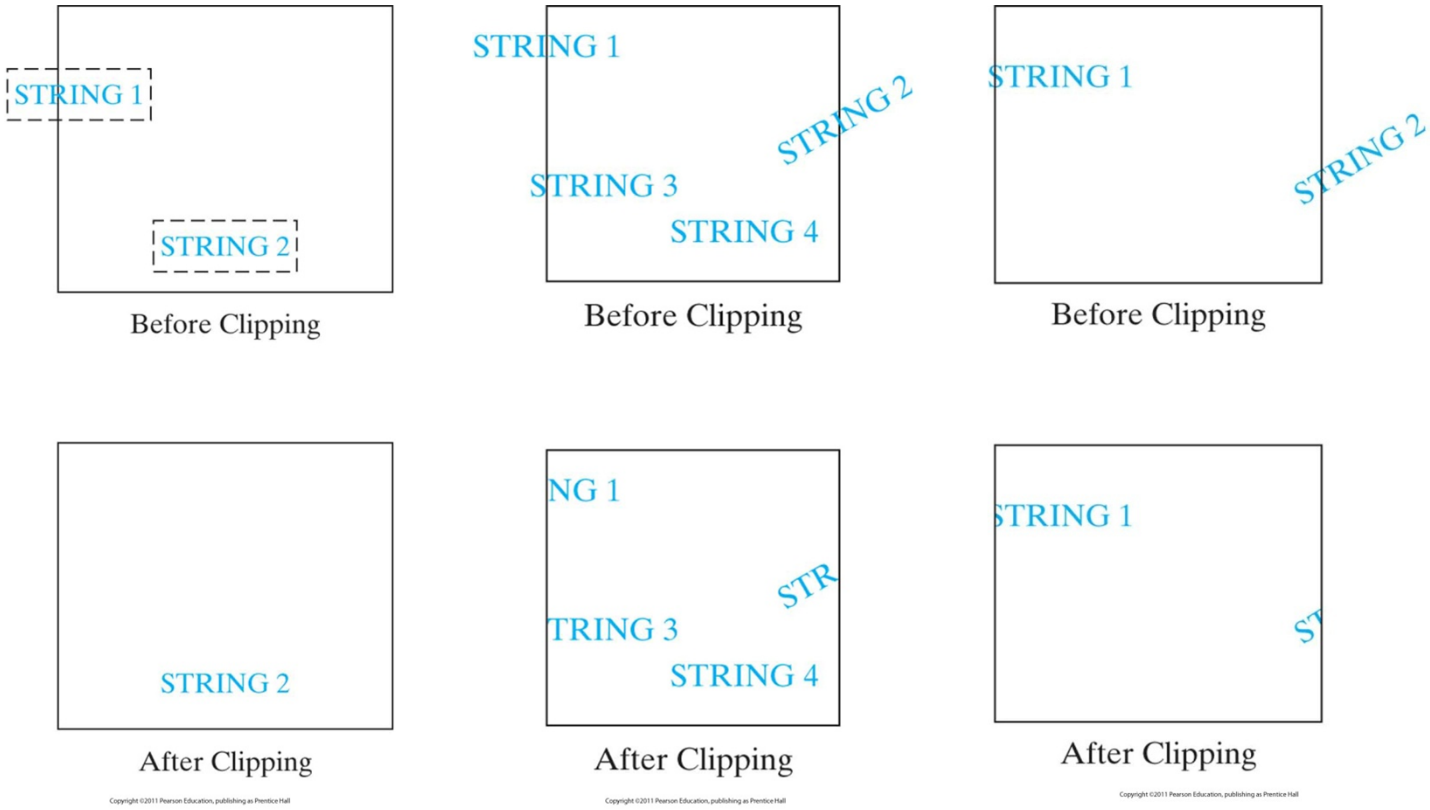
After clipping by xmax

1,2,I1,I2,I3,I4,6,7,I5,I6,9

After clipping by ymin:

1,2, I1,I2,I3,I7,I8,7,I5,I6,9

Text Clipping



Simplest method: all-or-none string clipping

Alternative one: all-or-none character clipping

Most accurate one: clipping on the components of individual characters

Summary

- 2D clipping algorithm
  - 2D point clipping
  - 2D line clipping
    - Cohen-Sutherland
    - Liang-Barsky
  - Polygon clipping
  - Text clipping