

# Clipping: LINES and POLYGONS

Dr. V Masilamani

[masila@iiitdm.ac.in](mailto:masila@iiitdm.ac.in)

Department of Computer Science and Engineering  
IIITDM Kancheepuram  
Chennai-127



What is Clipping

Need for Clipping

Cropping Vs Clipping

Clippings of Objects of Different Dimensions

Point Clipping

Line Clipping

Cohen-Sutherland Line Clipping Algorithm

Clipping with Polygon Window

Sutherland-Hodgman Algorithm to clip a polygon with polygon window

*Weiler\_Atherton algorithm to clip a polygon with polygon window*

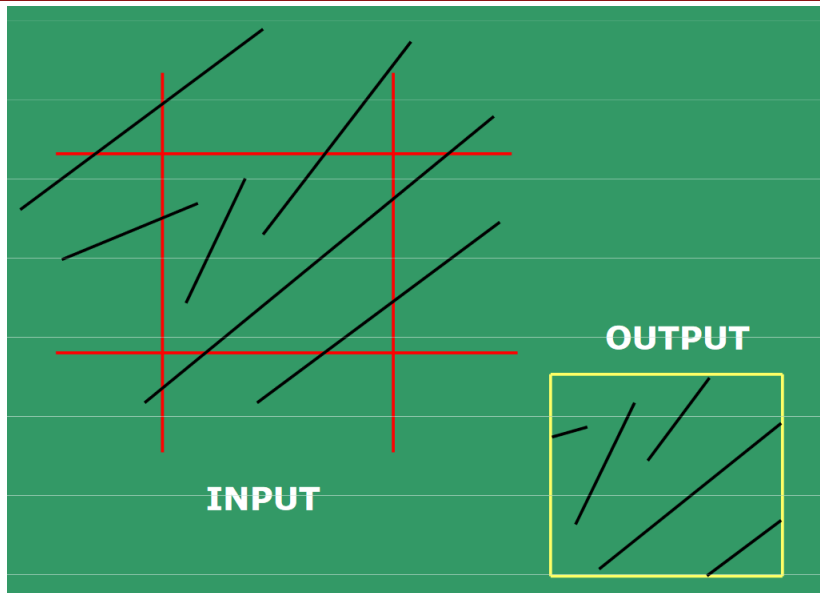
2D Graphics Pipeline

Acknowledgements

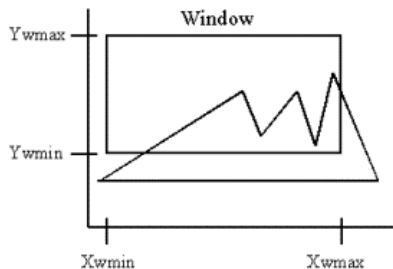


- ▶ A **Scene** is modeled as a set of objects
- ▶ **Viewing Window**: Rectangle is called as viewing window if a scene is viewed only inside the rectangle
- ▶ **Representation of Viewing Window**:  $(x_{min}, y_{min})$  and  $(x_{max}, y_{max})$
- ▶ **2D clipping**: Given a set of 2D objects, and a viewing window, display the portion of the objects inside the window
- ▶ **3D clipping**: Given a set of 3D objects, and a 3D viewing window(Cuboid), display the portion of the objects inside the window

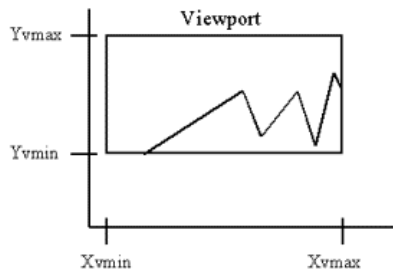
# What is Clipping (cont.)



To display a portion of an object on screen



World Coordinates



Device Coordinates



- ▶ The representation of image in memory is 2D array of numbers
- ▶ The representation of object in memory is the values of its parameters
- ▶ **Input for cropping sub image:** the image(2D array) and representation of viewing rectangle
- ▶ **Input for clipping object:** The representation of objects and a viewing rectangle
- ▶ **Output for cropping sub image:** the portion of image(2D array) within the viewing rectangle
- ▶ **Output for clipping object:** The portion of objects within the viewing rectangle



- ▶ Advantage of representation of object with parameters:
  - Less storage requirement
  - No interpolation is required when transformation is applied on the object, and hence no reduction in quality of displayed image

# Clippings of Object of Different Dimensions



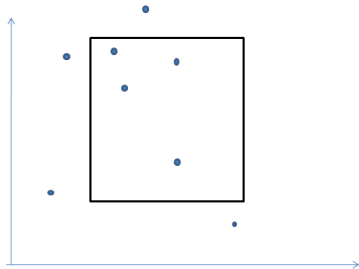
Point Clipping

Line Clipping

Polygon Clipping



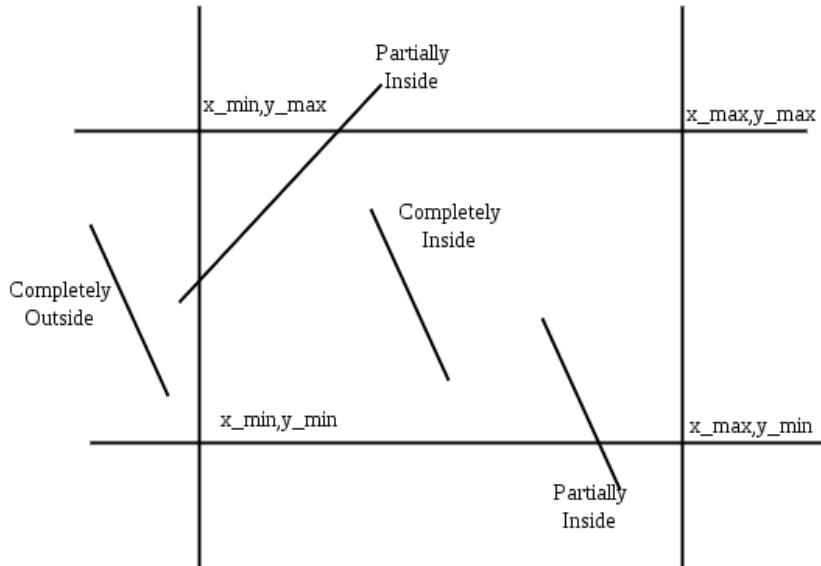
Given the set of points, display only the points inside the rectangular window



- Given a rectangle, display all the points within the rectangle

- ▶ Given a rectangle, display the portions of all line segments within the rectangle
- ▶ Input: Set of pairs of 2D points  $(p_{i-1}, p_i)$ , and  $(x_{min}, y_{min})$  and  $(x_{max}, y_{max})$  representing viewing window
  - Each pair of pint  $(p_{i-1}, p_i)$  represents line segment  $L_i$ , where  $p_i = (x_i, y_i)$
- ▶ Output: Set of pairs of 2D points  $(p'_{i-1}, p'_i)$  representing line segments  $L'_i$  that are portions of the  $L_i$  , which are completely inside the the viewing window

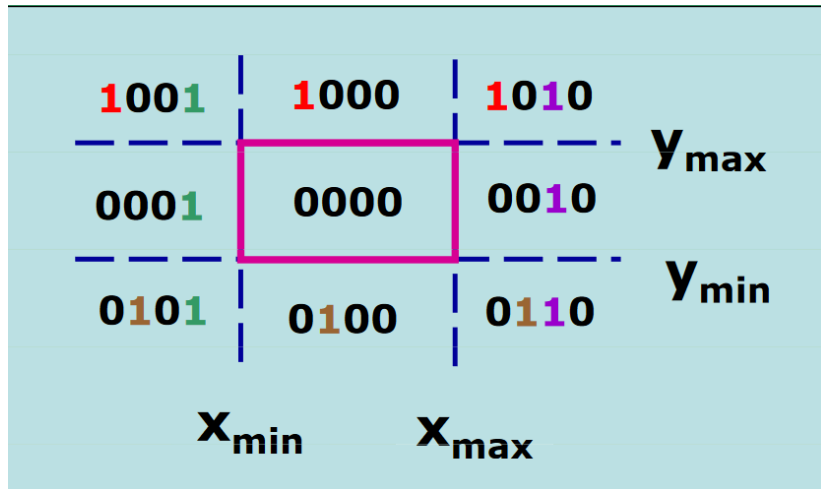
# Line Clipping (cont.)



# Cohen-Sutherland Line Clipping



Region Codes: Define a 4 bit code for each of the nine regions as given below



# Cohen-Sutherland Line Clipping (cont.)



Bit Number	1	0
First (MSB)	Above Top Edge $Y > Y_{max}$	Below Top Edge $Y < Y_{max}$
Second	Below Bottom Edge $Y < Y_{min}$	Above Bottom Edge $Y > Y_{min}$
Third	Right of Right Edge $X > X_{max}$	Left of Right Edge $X < X_{max}$
Fourth(LSB)	Left of Left Edge $X < X_{min}$	Right of Left Edge $X > X_{min}$



► To determine the region code for any point  $(X, Y)$ , use:

- $b_1 = \text{sign}(Y - Y_{\max});$   
( ie.  $b_1 = 1$  iff  $(X, Y)$  is above the line  $y = y_{\max}$  )
- $b_2 = \text{sign}(Y_{\min} - Y);$   
( ie.  $b_2 = 1$  iff  $(X, Y)$  is below the line  $y = y_{\min}$  )
- $b_3 = \text{sign}(X - X_{\max});$   
( ie.  $b_3 = 1$  iff  $(X, Y)$  is on right of  $x = x_{\max}$  )
- $b_4 = \text{sign}(X_{\min} - X);$   
( ie.  $b_4 = 1$  iff  $(X, Y)$  is on left of  $x = x_{\min}$  )
- Where  $\text{sign}(a) = 1$  if  $a \geq 0$ ; 0 otherwise;
- $\text{sign}(-4) = 0$ ;  $\text{sign}(3) = 1$



- ▶ To check if the line segment is completely inside or completely outside or partially inside use region codes of endpoints of the line segment as given below
  - **Case 1: Trivial Acceptance:** If both endpoint codes are [0000], the line lies **completely inside** the box, no need to clip. This is the simplest case (e.g. L1).
  - **Case 2: Trivial Rejection:** If any line has 1 in the same bit positions of both the endpoints, then it is guaranteed to **lie outside the box completely** (e.g. L2 and L3).
  - **Case 3:** Line that does not belong to case 1 and case 2 (eg: L4, L5, L6)

# Cohen-Sutherland Line Clipping (cont.)



1001	1000	1010
0001	0000	0010
0101	0100	0110

$x_{\min}$   $x_{\max}$   $y_{\min}$   $y_{\max}$

- Neither completely reject nor inside the box:  
Lines:  $L_4$  and  $L_5$  - needs more processing.
- What about Line  $L_6$  ?





- ▶ Clipping is required for line segment that is neither Completely IN nor completely OUT; e.g. Lines: L4, L5 and L6.
- ▶ Basic idea to clip line segment: Clip the part of the line segment which lies outside the viewing window

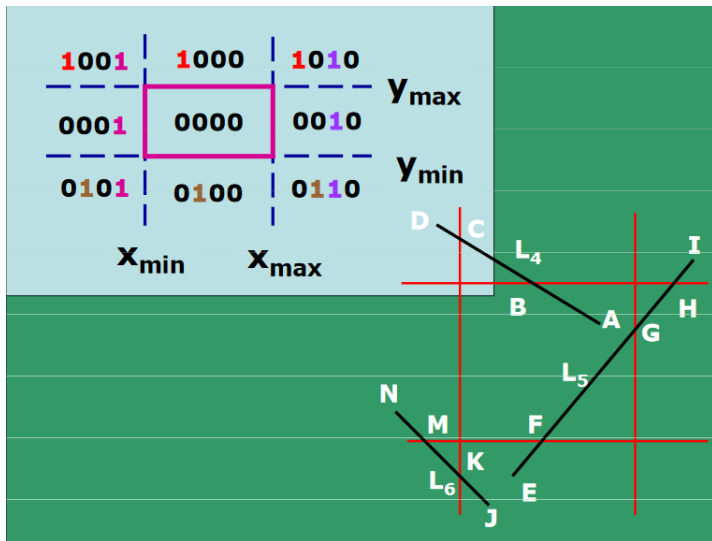
## Algorithm Steps:

- ▶ **S1)** For end points  $(p_{i-1}, p_i)$  of each line segment  $L_i$ , Compute region codes, say  $c_{i-1}$  and  $c_i$  respectively.
- ▶ **S2)** if  $((c_{i-1} == c_i) == 0000)$  accept(display) the segment  $L_i$   
else if  $((c_{i-1} \& c_i) \neq 0000)$  reject( do not display)  $L_i$   
else
  - Obtain an endpoint p that lies outside the box (at least one will ?)
  - Using the region code of that end point p, obtain the edge(the infinitely extended side of the viewing window) as follows.  
In the region code of p
    - ▶ if  $b_1 = 1$ ,  $y = y_{max}$  is to be intersected with line  $L_i$



- ▶ if  $b_2 = 1$ ,  $y = y_{min}$  is to be intersected with line  $L_i$
- ▶ if  $b_3 = 1$ ,  $x = x_{max}$  is to be intersected with line  $L_i$
- ▶ if  $b_4 = 1$ ,  $x = x_{min}$  is to be intersected with line  $L_i$
- **Question:** If more than one bit are 1s, which one is to be used?  
-Ans: use any bit
- Let the intersection point be  $I$ , replace the end point  $p$  with  $I$ , and compute the region code for  $I$   
(ie. the line segment  $(p, q)$  has become  $(I, q)$ , and this process is called as clipping)
- Repeat S2 if Codes for  $I$  and  $q$  are not 0000

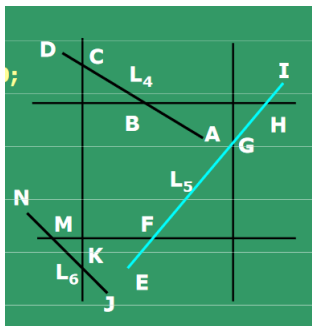
# Cohen-Sutherland Line Clipping (cont.)



# Cohen-Sutherland Line Clipping (cont.)



e.g. Take Line  $L_5$  (endpoints - E and I): E has region code 0100 (to be clipped w.r.t. bottom edge);



- ▶ So EI is clipped to FI;
- ▶ Region code of F is 0000; But Region code of I is 1010; Clip(w.r.t.top edge) to get FH.
- ▶ Region code of H is 0010; Clip(w.r.t.right edge) to get FG.



- ▶ Since region code of G and also R are 0000, display the segment FG.
- ▶ Alternative process of clipping to get FG
  - Intersect line EI with  $x = x_{max}$  to get G
  - Intersect line EG with  $y = y_{min}$  to get F
  - Display the segment FG as codes of both F and G are 0000



## How to find intersection of line passing through the end points of a segment with an edge of the viewing window

- ▶ When the end points of the segment are  $(x_1, y_1)$  and  $(x_2, y_2)$ , and edge is  $y = y_{max}$  (Top Edge)
- ▶ slope of the line:  $m = (y_2 - y_1)/(x_2 - x_1)$
- ▶ Line equation:  $y - y_1 = m(x - x_1)$
- ▶ sub  $y = y_{max}$  in line eqn. we get,  $y_{max} - y_1 = m(x - x_1)$
- ▶ Therefore
$$x = x_1 + (1/m)(y_{max} - y_1) = x_1 + (x_2 - x_1)/(y_2 - y_1) * (y_{max} - y_1)$$
- ▶ The Intersection point is  $(x, y)$ ,  
where  $x = x_1 + (x_2 - x_1)(y_{max} - y_1)/(y_2 - y_1)$ ,  
 $y = y_{max}$



Similarly the intersection with other edges, namely Bottom, Right and left can be derived

The Intersection points  $(x, y)$  in each of the types is given below

► Bottom Edge:

$$x = x_1 + (x_2 - x_1) * \frac{y_{min} - y_1}{y_2 - y_1}$$

$$y = y_{min}$$

► Right Edge:

$$x = x_{max}$$

$$y = y_1 + (y_2 - y_1) * \frac{x_{max} - x_1}{x_2 - x_1}$$

► Left edge:

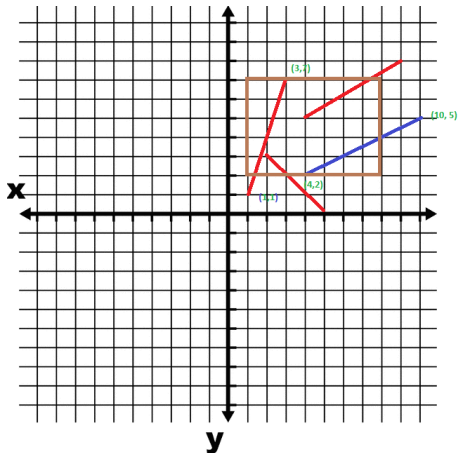
$$x = x_{min}$$

$$y = y_1 + (y_2 - y_1) * \frac{x_{min} - x_1}{x_2 - x_1}$$

# Cohen-Sutherland Line Clipping (cont.)

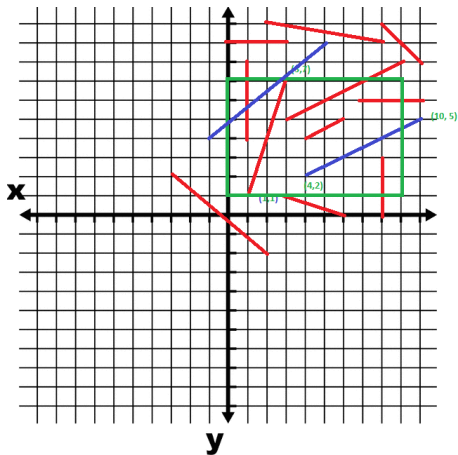


HW: Trace Cohen-Sutherland algorithm on the following inputs





# Cohen-Sutherland Line Clipping (cont.)



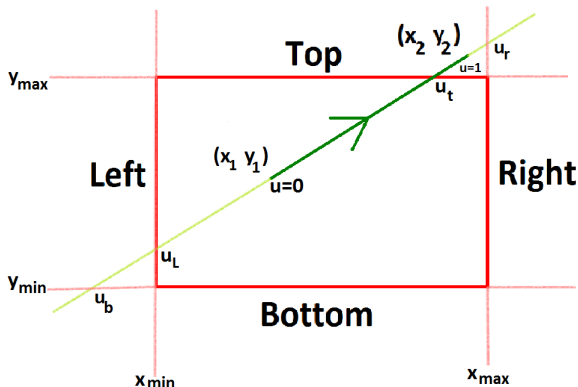
**Note:**

Viewing window is in brown in the first, and green in the second. A few endpoints are explicitly given other end points can be calculated from the graph

# Liang-Barsky Line Clipping



**Basic Idea:** 1) Extend the line segment and the window boundaries; 2) Find the visible portion using intersection points of the line with each of the window boundaries





- ▶ Parametric equation of the **line segment** with end points  $(x_1, y_1)$  and  $(x_2, y_2)$ :  
 $x = x_1 + u\Delta x$ ;  
 $y = y_1 + u\Delta y, 0 \leq u \leq 1$   
Where,  $\Delta x = x_2 - x_1$ ;  $\Delta y = y_2 - y_1$
- ▶ Parametric equation of the **line** passing through points  $(x_1, y_1)$  and  $(x_2, y_2)$ :  
 $x = x_1 + u\Delta x$ ;  
 $y = y_1 + u\Delta y, -\infty \leq u \leq \infty$   
Where,  $\Delta x = x_2 - x_1$ ;  $\Delta y = y_2 - y_1$
- ▶ The point  $(x, y)$  is considered to be within a rectangle with parameters  $(x_{min}, y_{min})$  and  $(x_{max}, y_{max})$  iff
  - $x_{min} \leq x_1 + u\Delta x \leq x_{max}$
  - $y_{min} \leq y_1 + u\Delta y \leq y_{max}$
- ▶ Equivalently



- $u(-\Delta x) \leq x_1 - x_{min}$
- $u\Delta x \leq x_{max} - x_1$
- $u(-\Delta y) \leq y_1 - y_{min}$
- $u\Delta y \leq y_{max} - y_1$

Each of these four inequalities, can be expressed as:

$$u.p_k \leq q_k; k = 1, 2, 3, 4$$

Where the parameters are defined as:

- ▶  $p_1 = -\Delta x, q_1 = x_1 - x_{min}$
- ▶  $p_2 = \Delta x, q_2 = x_{max} - x_1$
- ▶  $p_3 = -\Delta y, q_1 = y_1 - y_{min}$
- ▶  $p_4 = \Delta y, q_1 = y_{max} - y_1$



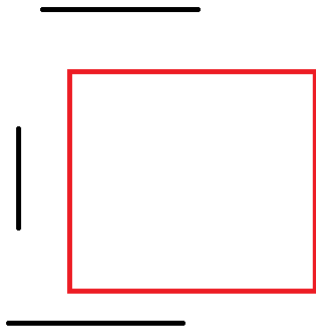
Based on these four inequalities, we can find the following conditions for line clipping:

- ▶ If  $p_k = 0$ , the line is parallel to  $X$  or  $Y$  axes:
  - $k = 1 \implies \Delta x = 0 \implies x = x_1$  which is a line parallel to  $Y$  axis
  - $k = 2 \implies \Delta X = 0 \implies x = x_1$  which is a line parallel to  $Y$  axis
  - $k = 3 \implies \Delta y = 0 \implies y = y_1$  which is a line parallel to  $X$  axis
  - $k = 4 \implies \Delta y = 0 \implies y = y_1$  which is a line parallel to  $X$  axis

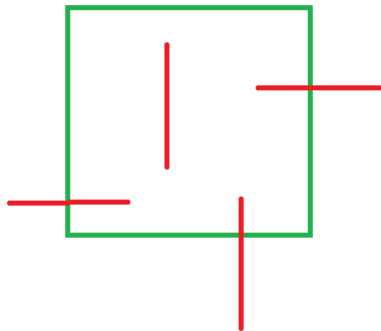
## Observations:

- ▶ **Case 1:** for any  $k$ , for which  $p_k = 0$ :
  - if  $q_k < 0$ , the line is completely outside the boundary  
(When  $k = 1$ ,  $q_1 < 0 \implies x_1 < x_{\min}$ , and  $p_1 = 0 \implies x = x_1$ .  
Similarly for other  $K$ , the observation can be proved)
  - $q_k \geq 0$ , the line is inside the boundary, and hence requires clipping

# Liang-Barsky Line Clipping (cont.)

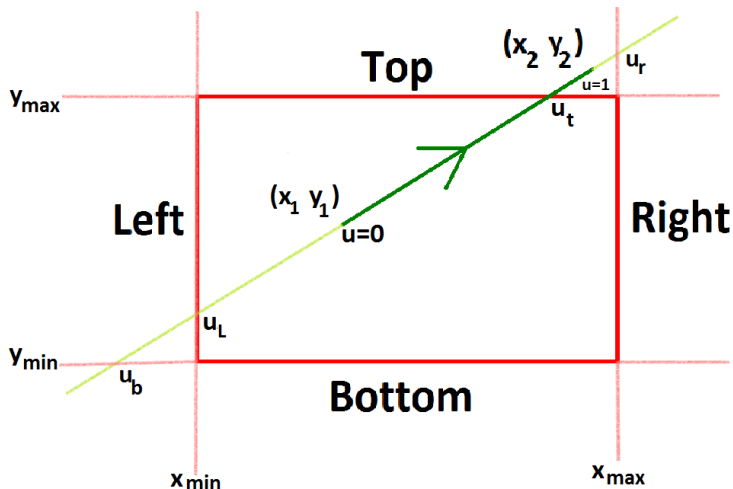


When  $p_k=0$  and  $q_k<0$



When  $p_k=0$  and  $q_k>0$

# Liang-Barsky Line Clipping (cont.)



Line when  $\Delta x > 0$  and  $\Delta y > 0$  ;





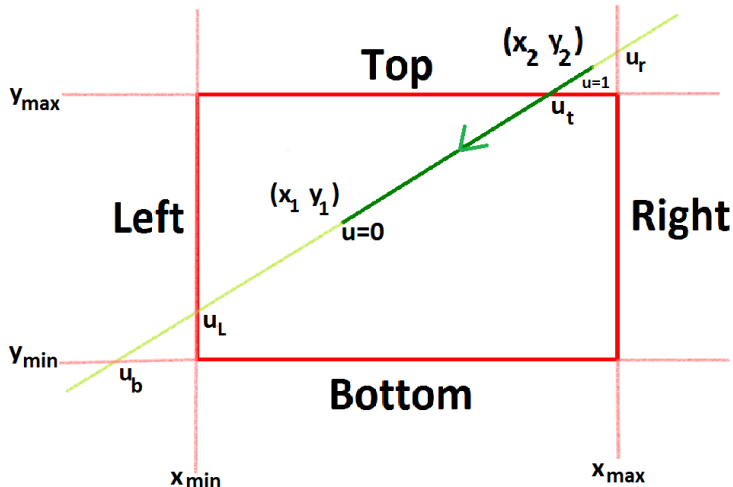
**When  $\Delta x > 0$  and  $\Delta y > 0$**

- $(x, y)$  will increase in both  $x$  and  $y$  direction as  $u$  increases as  
 $x = x_1 + u\Delta x$ ;  $y = y_1 + u\Delta y$
- $p_1 = -\Delta x < 0$ ;  $p_3 = -\Delta y < 0$
- But  $p_2 = \Delta x > 0$ ;  $p_4 = \Delta y > 0$
- Notice that the line moves from out side to inside of left edge(due to  $p_1 < 0$ ) and also the line moves from outside to inside of the bottom edge(due to  $p_3 < 0$ )
- The line moves from inside to outside of top edge(due to  $p_4 > 0$ ) and also the line moves from inside to outside of the right edge(due to  $p_2 > 0$ )
- Hence the clipping boundary corresponding to  $k$ :
  - ▶ When  $k=1$ , Left edge
  - ▶ When  $k=2$ , Right edge
  - ▶ When  $k=3$ , Bottom edge



- ▶ When  $k=4$ , Top edge
- **Conclusion when  $\Delta x > 0$  and  $\Delta y > 0$ :**
  - ▶ If  $p_k < 0$ , the line proceeds from the outside to the inside of the corresponding clipping boundary line (consider infinite extensions in both line and window boundary).
  - ▶ If  $p_k > 0$ , the line proceeds from the inside to the outside of the corresponding clipping boundary line (consider infinite extensions in both line and window boundary).

# Liang-Barsky Line Clipping (cont.)



Line when  $\Delta x < 0$  and  $\Delta y < 0$  ;

When  $\Delta x < 0$  and  $\Delta y < 0$

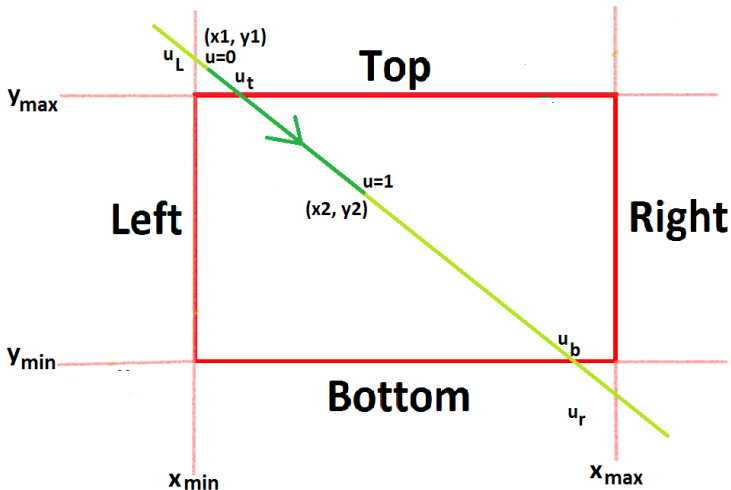


- $(x, y)$  will decrease in both  $x$  and  $y$  direction as  $u$  increases as  
 $x = x_1 + u\Delta x$ ;  $y = y_1 + u\Delta y$
- $p_1 = -\Delta x > 0$ ;  $p_3 = -\Delta y > 0$
- But  $p_2 = \Delta x < 0$ ;  $p_4 = \Delta y < 0$
- Notice that the line moves from out side to inside of right edge(due to  $p_2 < 0$ ) and also the line moves from outside to inside of the top edge(due to  $p_4 < 0$ ) ( Justification when  $k = 1$ :  
 $p_1 > 0 \implies \Delta x < 0 \implies x = x_1 + u\Delta x$  will decreases as  $u$  increases.  
As  $u$  moves from  $-\infty$  to  $\infty$ , the  $x$  coordinate of the line ( $x = x_1 + u\Delta x$ ) moves inside to outside of left clipping boundary)
- Notice that the line moves from inside to outside of bottom edge(due to  $p_3 > 0$ ) and also the line moves from inside to outside of the left edge(due to  $p_1 > 0$ ) ( Justification when  $k = 1$ :  
 $p_1 > 0 \implies \Delta x < 0 \implies x = x_1 + u\Delta x$  will decreases as  $u$  increases.  
As  $u$  moves from  $-\infty$  to  $\infty$ , the  $x$  coordinate of the line ( $x = x_1 + u\Delta x$ ) moves inside to outside of left clipping boundary.  
Similarly, for other  $k$ , boundary crossing can be proved)



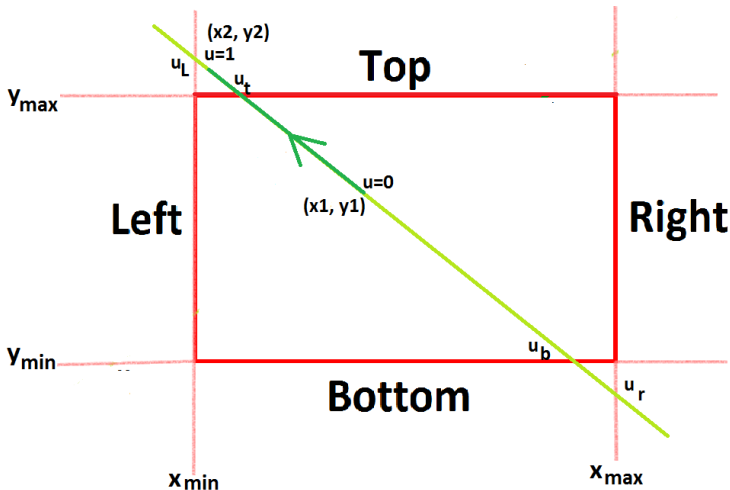
- Hence the clipping boundary corresponding to  $k$ :
  - ▶ When  $k=1$ , Left edge
  - ▶ When  $k=2$ , Right edge
  - ▶ When  $k=3$ , Bottom edge
  - ▶ When  $k=4$ , Top edge
- **Conclusion when  $\Delta x < 0$  and  $\Delta y < 0$ :**
  - ▶ If  $p_k < 0$ , the line proceeds from the outside to the inside of the corresponding clipping boundary line (consider infinite extensions in both line and window boundary).
  - ▶ If  $p_k > 0$ , the line proceeds from the inside to the outside of the corresponding clipping boundary line (consider infinite extensions in both line and window boundary).

# Liang-Barsky Line Clipping (cont.)



Line when  $\Delta x > 0$  and  $\Delta y < 0$  ; Hw: prove the same conclusion

# Liang-Barsky Line Clipping (cont.)



Line when  $\Delta x < 0$  and  $\Delta y > 0$  ; HW: Prove the same conclusion



- ▶ Conclusion for arbitrary line:
  - If  $p_k < 0$ , the line proceeds from the outside to the inside of the corresponding clipping boundary line  
**called as case 2**
  - If  $p_k > 0$ , the line proceeds from the inside to the outside of the particular clipping boundary  
**called as case 3**
- ▶ In both case 2 and 3, the intersection parameter is calculated as follows:
  - Consider  $q_1/p_1 = (x_1 - x_{min})/(-\Delta x) = (x_{min} - x_1)/\Delta x = u$  ( WKT  $x = x_1 + u\Delta x$  for any  $x$ . put  $x = x_{min}$  )
  - Similarly, we can prove  $u = \frac{q_k}{p_k}$  for other  $k$
  - Hence  $u = \frac{q_k}{p_k}$  is the parameter for intersecting point of line and corresponding window boundary (in case of  $k = 1$ , the boundary is  $x = x_{min}$ )





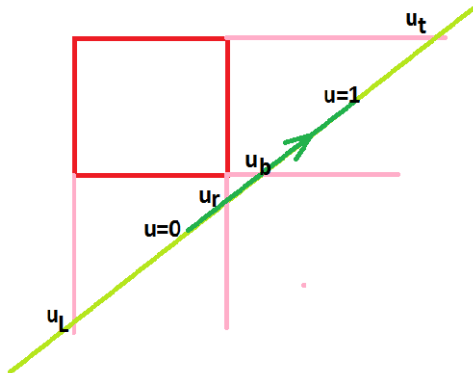
- Recall that the parametric equation of line passing through end points  $(x_1, y_1)$  and  $(x_2, y_2)$  of the corresponding line segment is  $x = x_1 + u\Delta x$ ;  $y = y_1 + u\Delta y$   $-\infty < u < \infty$ .
- When  $u = 0$ , the point on the line is  $(x_1, y_1)$ ; and
  - When  $u = 1$ , the point on the line is  $(x_2, y_2)$



## ► Clipping:

- Let  $u_1$  and  $u_2$  be the values of  $u$  such that  $u_1$  and  $u_2$  correspond to the beginning and ending of portion of the line segment inside the window
- Then  $u_1 = \max(\{q_k/p_k \mid p_k < 0\} \cup \{0\})$ ,
- Let  $u_2 = \min(\{q_k/p_k \mid p_k > 0\} \cup \{1\})$
- if  $(u_1 > u_2)$  then reject the line as the line lies completely outside the window  
else  
Obtain  $(x, y)$  and  $(x', y')$  by substituting  $u_1$  and  $u_2$  for  $u$  respectively in the parametric line equation

# Liang-Barsky Line Clipping (cont.)



Let  $u_l, u_b, u_r, u_t$  be the parameter values corresponding to the intersection of line and left, bottom, right and top boundaries of the window respectively  
 $u_1 = \max(0, u_l, u_b) = u_b$ ;  $u_2 = \min(1, u_r, u_t) = u_r$ ;  $u_1 > u_2$



**Show that if( $u_1 > u_2$ ) then the line lies completely outside the window**

**else**

**the line segment with endpoints  $(x, y)$  and  $(x', y')$  obtained when  $u_1$  and  $u_2$  are substituted for  $u$  in the parametric line equation lies completely inside the window**

**The proof can be given through geometry by considering each of the possible cases on  $\Delta x$  and  $\Delta y$  for given end points of a line segment**

There are four cases:

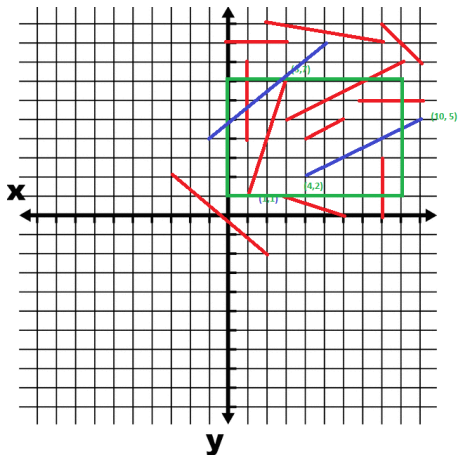
1.  $\Delta x > 0$  and  $\Delta y > 0$  (already done)
2.  $\Delta x > 0$  and  $\Delta y < 0$
3.  $\Delta x < 0$  and  $\Delta y > 0$
4.  $\Delta x < 0$  and  $\Delta y < 0$



## The Algorithm

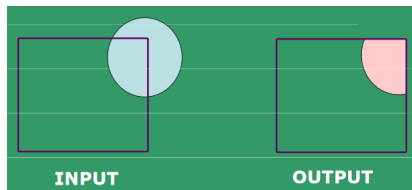
- ▶ if ( $p_k = 0$  and  $q_k < 0$ ) for any  $k$  , eliminate the line and stop  
else  
    proceed to the next step
- ▶ Find  $u_1 = \max( \{q_k/p_k \mid p_k < 0\} \cup \{0\} )$
- ▶ Find  $u_2 = \min( \{q_k/p_k \mid p_k > 0\} \cup \{1\} )$
- ▶ if( $u_1 > u_2$ ) eliminate the line as it completely lies outside the window  
else
  - Find  $x'_1 = x_1 + u_1 \Delta x$
  - Find  $y'_1 = y_1 + u_1 \Delta y$
  - Find  $x'_2 = x_1 + u_2 \Delta x$
  - Find  $y'_2 = y_1 + u_2 \Delta y$
  - Output  $(x'_1, y'_1)$  and  $(x'_2, y'_2)$

# Liang-Barsky Line Clipping (cont.)



- Show the trace of Liang-Barsky algorithm for the above input

What about Circle/Ellipse clipping or for curves ??

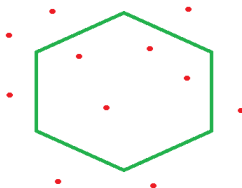


Home Work:

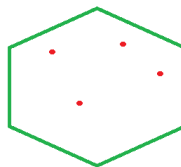
- Show that if  $(u_1 > u_2)$  then the line lies completely outside the window; else the line segment with endpoints  $(x, y)$  and  $(x', y')$  obtained when  $u_1$  and  $u_2$  are substituted for  $u$  in the parametric line equation lies completely inside the window when
  - $\Delta x > 0$  and  $\Delta y < 0$
  - $\Delta x < 0$  and  $\Delta y > 0$
  - $\Delta x < 0$  and  $\Delta y < 0$

## Keep viewing window as polygon

- ▶ Clip Points
- ▶ Clip Lines
- ▶ Clip Polygons



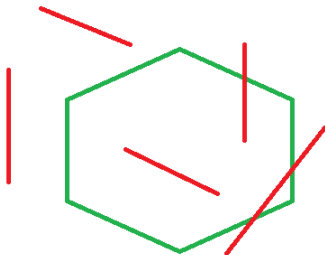
**Input for point clipping with a polygon window**



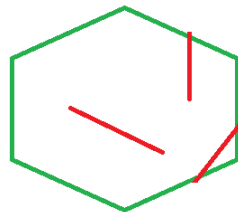
**Output of point clipping with the polygon window**



## Clipping Lines using Polygon Window

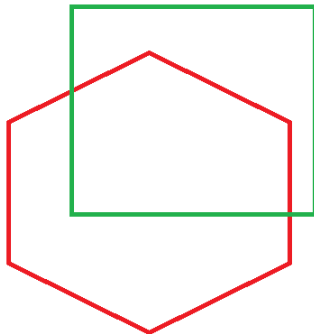


**Input for line clipping  
using polygon window**



**Output of line clipping  
using polygon window**

## Clipping Polygon using Polygon Window



**Input for polygon clipping  
using polygon window**



**Output of polygon clipping  
using polygon window**

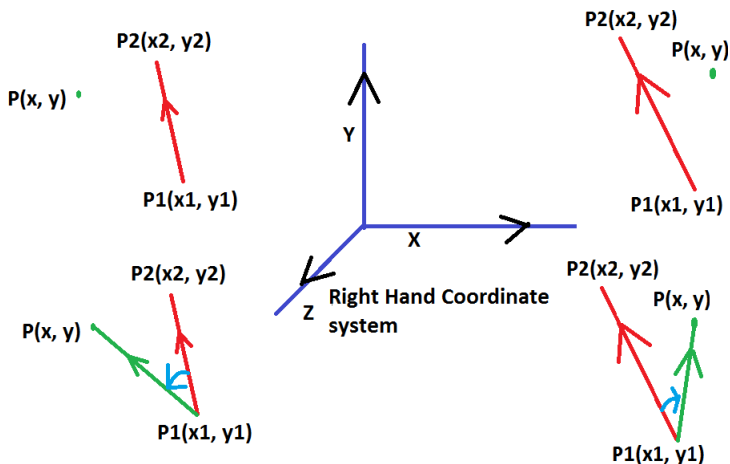


- ▶ **Basic Problem:** Given a point  $p$  and a directed line  $l$ , check if  $p$  lies on left or right of the directed line
- ▶ **Soln:**
  - Consider the vectors  $P_1\vec{P}_2$  and  $P_1\vec{P}$
  - Find the cross product  $P_1\vec{P}_2 \times P_1\vec{P}$
  - if sign  $(P_1\vec{P}_2 \times P_1\vec{P})$  is positive then  $p$  lies on left; else on right
  - How to find  $(P_1\vec{P}_2 \times P_1\vec{P})$ 
    - ▶  $P_1\vec{P}_2 = (x_2 - x_1, y_2 - y_1)$
    - ▶  $P_1\vec{P} = (x - x_1, y - y_1)$
    - ▶  $P_1\vec{P}_2 \times P_1\vec{P} = ((x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1))\vec{k}$
    - ▶ if  $((x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)) > 0$  then  $p$  lies on left; else on right

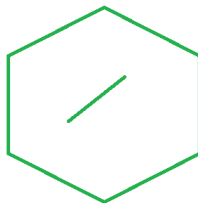
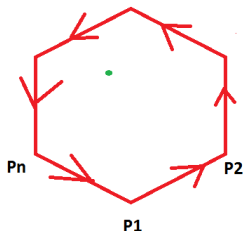
# Clipping with Polygon Window (cont.)



Left-Right test:

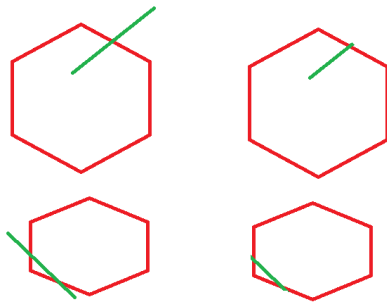


**Inside-Outside test: Given a point  $P$  and a polygon window  $W$ , check if  $P$  lies inside  $W$**

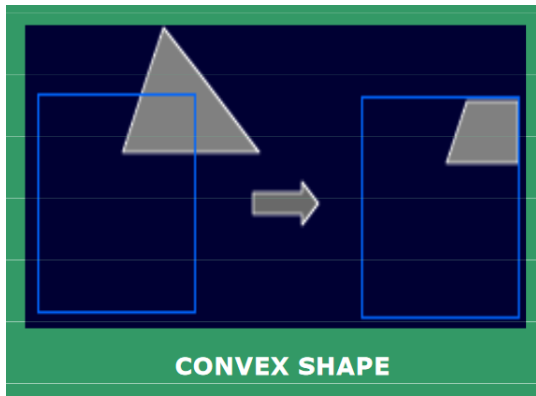


**How to use**

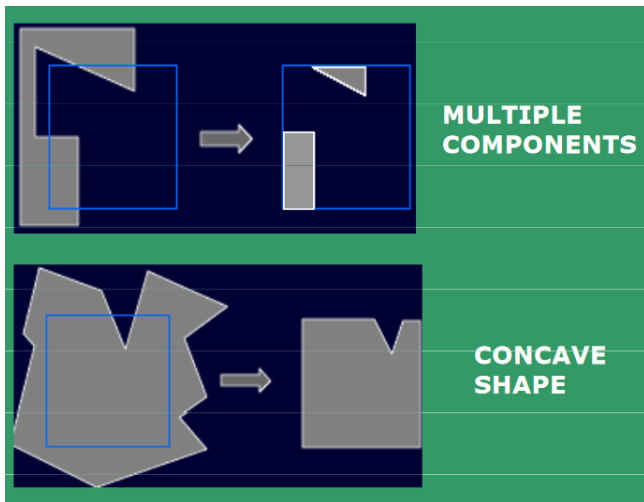
**inside-outside test to clip line with polygon window**



## Examples of Clipping Polygon using Polygon Window



# Clipping with Polygon Window (cont.)





# Sutherland-Hodgman Algorithm to clip a polygon with polygon window



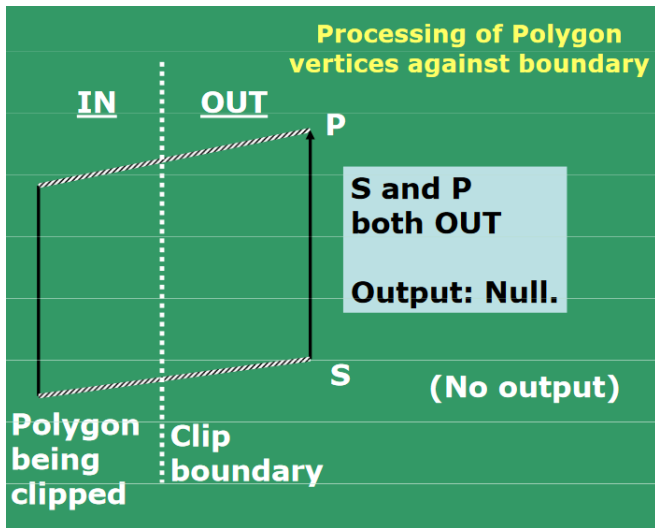
- ▶ Input: Vertex sequence  $P_1, \dots, P_N$  representing polygon to be clipped, and another vertex sequence  $Q_1, \dots, Q_M$  representing convex polygon window (clipping window)
- ▶ Output: The sequence of vertices of clipped polygon
- ▶ For each edge  $E = (A, B)$  in clipping polygon window  
for each edge  $P_{i-1} \vec{P_i}$  in the polygon to be clipped
  - If both  $P_{i-1}$  and  $P_i$  lie on the left of the edge  $E$  then output the vertex  $P_i$
  - If both  $P_{i-1}$  and  $P_i$  lie on the right of the edge  $E$  then output nothing
  - If  $P_{i-1}$  is on the left and  $P_i$  is on the right of the edge  $E$ , then find the intersection point  $I$  between the extended edge  $E$  and the line segment  $P_{i-1} \vec{P_i}$ , and output  $I$

# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)

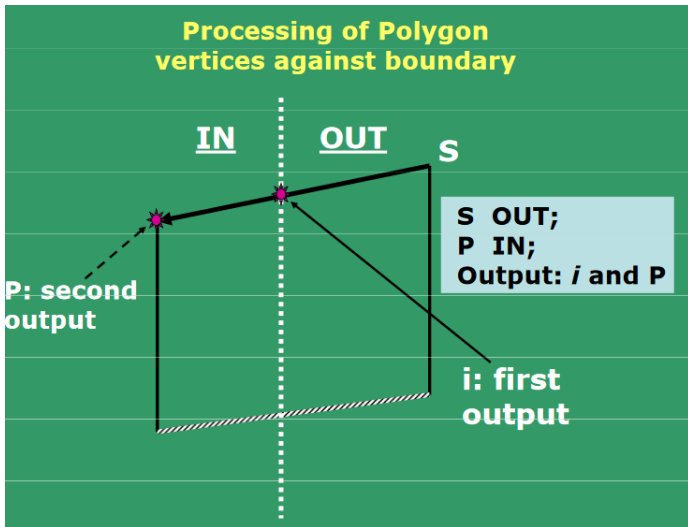


- If  $P_{i-1}$  is on the right and  $P_i$  is on the left of the edge  $E$ , then find the intersection point  $I$  between the extended edge  $E$  and the line segment  $\overrightarrow{P_{i-1}P_i}$ , and output  $I$  and  $P_i$

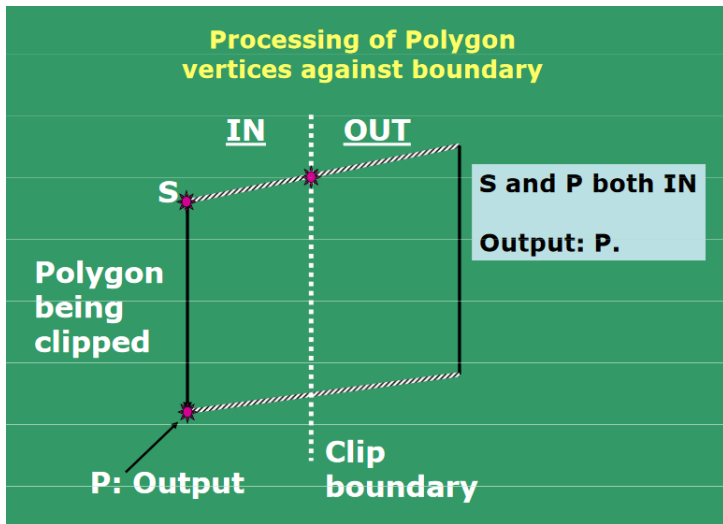
# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)



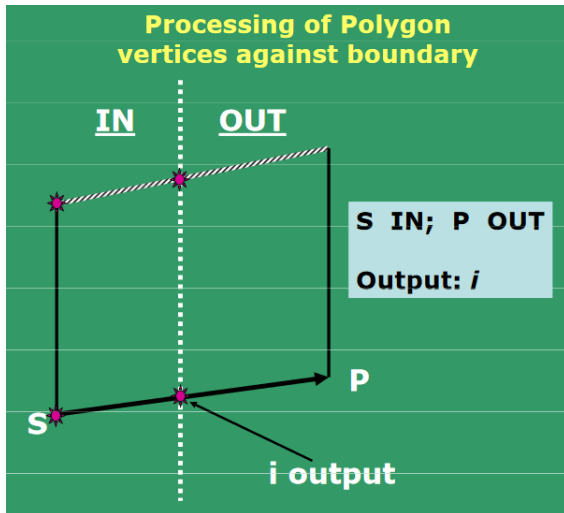
# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)



# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)



# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)

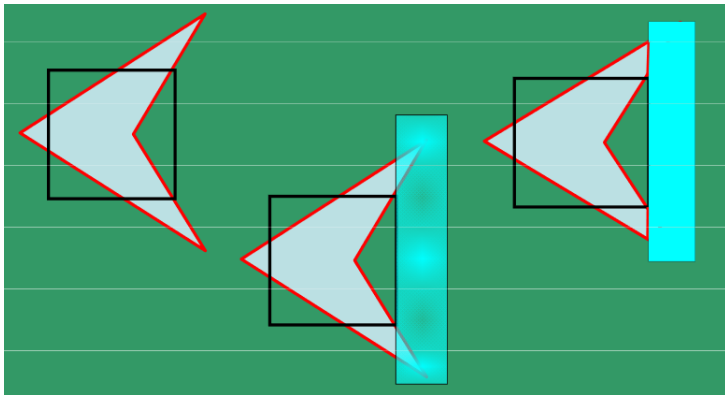


# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)

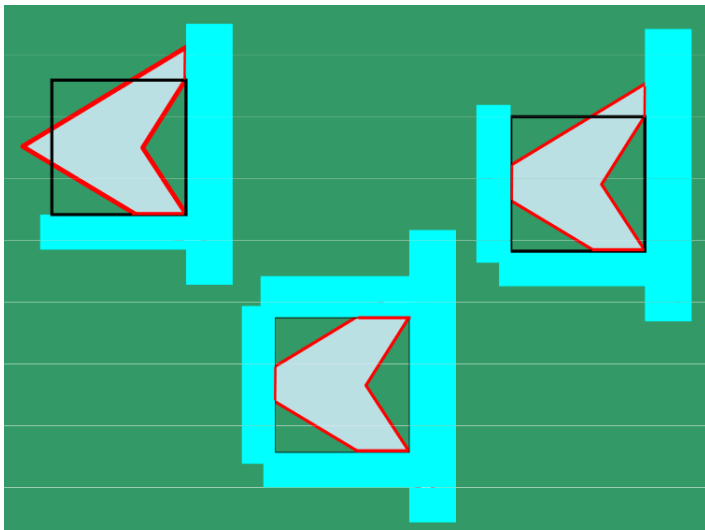


## Trace of the algorithm

May have to add new vertices to the list.



# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)

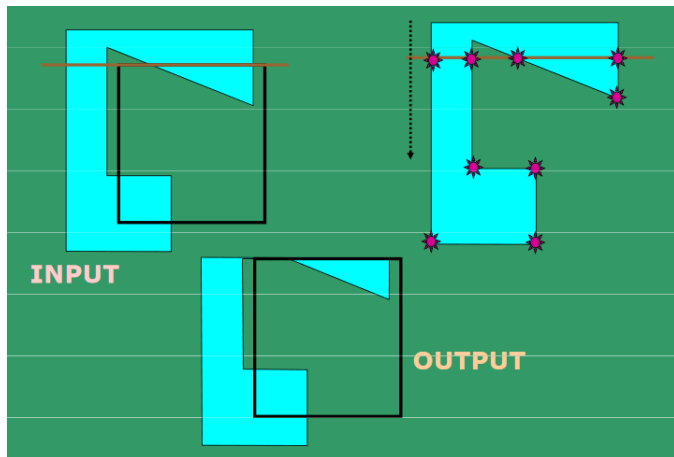




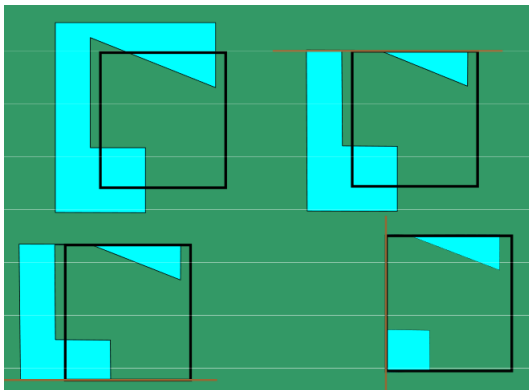
# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)



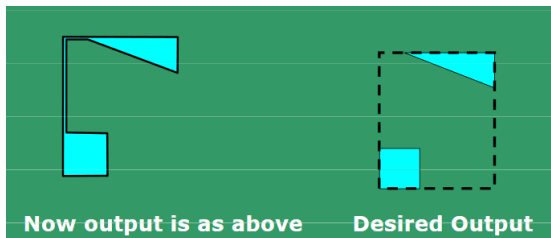
## Problems with multiple components



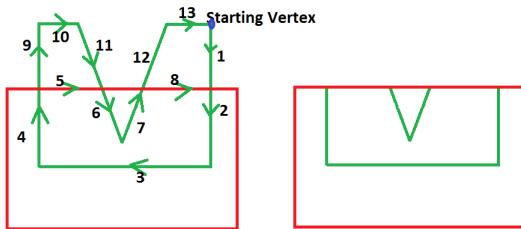
# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)



# Sutherland-Hodgman Algorithm to clip a polygon with polygon window (cont.)

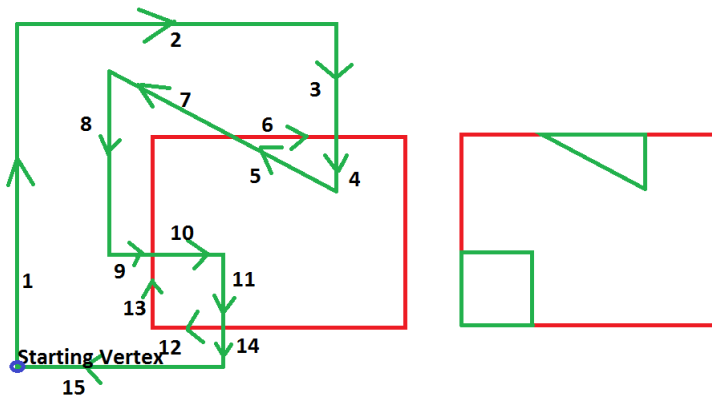


- ▶ Any Idea??
- ▶ Weiler-Atherton algorithm

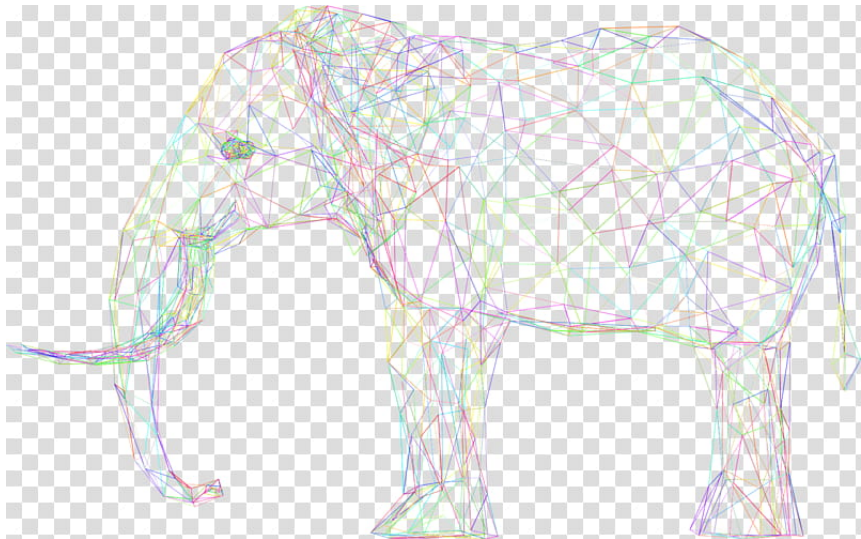


- For say, clockwise processing of polygons, follow:
  - For OUT – > IN pair, follow the polygon boundary
  - For IN – > OUT pair, follow Window boundary

- For say, clockwise processing of polygons, follow:
  - For OUT  $\rightarrow$  IN pair, follow the polygon boundary
  - For IN  $\rightarrow$  OUT pair, follow Window boundary



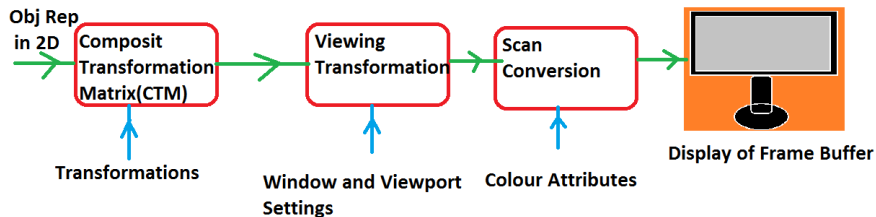
# 2D Graphics Pipeline



# 2D Graphics Pipeline (cont.)



# 2D Graphics Pipeline (cont.)







- ▶ Some of the slides have been adopted from NPTEL and different internet sources. The due credits are acknowledged.



Thank You! :)