ICT 4203

Computer Graphics and Animation

Lecture 12

Liang-Barsky Line-Clipping
Sutherland-Hodgeman Polygon Clipping

Md. Mahmudur Rahman
Lecturer
Institute of Information Technology

Liang-Barsky Algorithm

- The Liang-Barsky algorithm is a line clipping algorithm. This algorithm is more efficient than Cohen–Sutherland line clipping algorithm and can be extended to 3-Dimensional clipping. This algorithm is considered to be the faster parametric line-clipping algorithm.
- The following parametric equations represent a line from **(x_1,y_1)** to **(x_2,y_2)** along with its infinite extension:

$$x = x_1 + \Delta x.u$$

$$y = y_1 + \Delta y.u$$

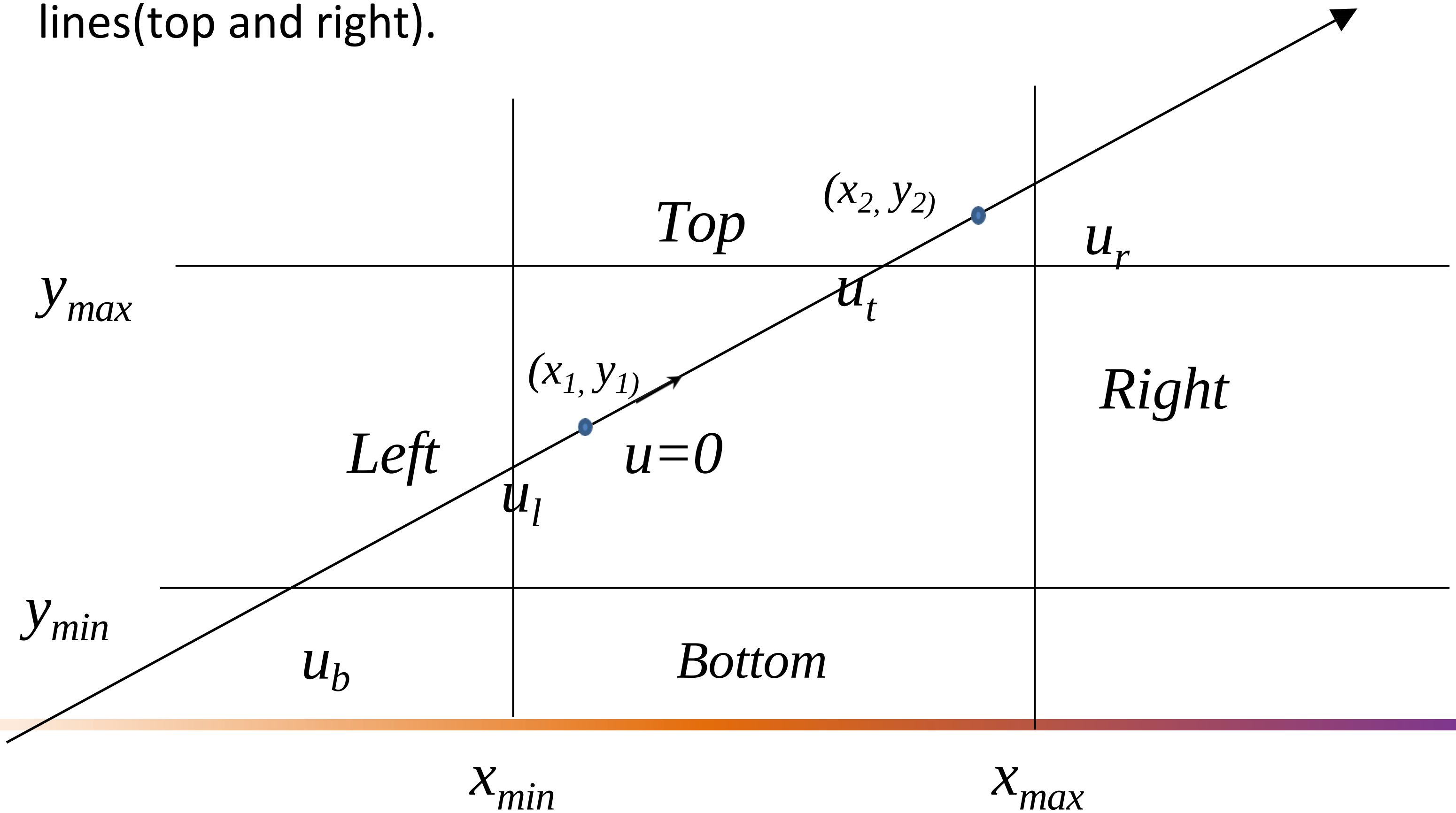
Where,

$$\Delta x = x_2 - x_1$$

$$\Delta y = y_2 - y_1$$

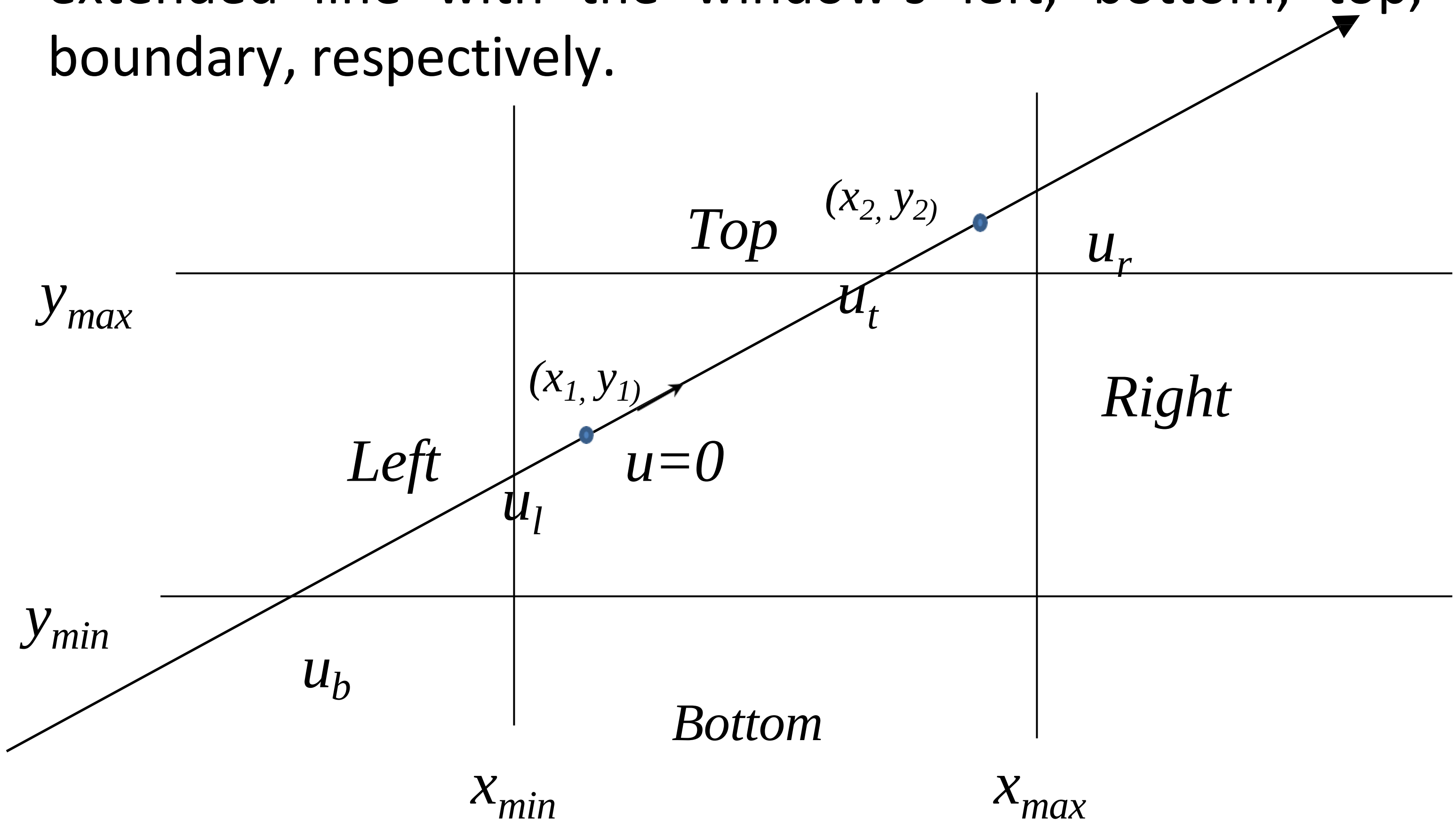
Liang-Barsky Algorithm

- The line itself corresponds to 0<=u<=1.
- U increasing from - ∞ to ∞.
- First move from the outside to the inside of the clipping window's two boundary lines (bottom and left).
- Then move from the inside to the outside of the other two boundary lines(top and right).



Liang-Barsky Algorithm

- u_1=maximum(0,u_l,u_b) and u_2=minimum(1,u_t,u_r)
- u_l, u_b, u_t, u_r correspond to the intersection point of the extended line with the window's left, bottom, top, right boundary, respectively.



Liang-Barsky Algorithm

- For point (x,y) inside the clipping window, we have:

$$x_{\min} \le x_1 + u\Delta x \le x_{\max}$$
$$y_{\min} \le y_1 + u\Delta y \le y_{\max}$$

- Rewrite the four inequalities as:

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

- Where

$p_1 = -\Delta x,$	$q_1 = x_1 - x_{\min}$	Left
$p_2 = \Delta x,$	$q_2 = x_{\max} - x_1$	Right
$p_3 = -\Delta y,$	$q_3 = y_1 - y_{\min}$	Buttom
$p_4 = \Delta y$	$q_4 = y_{\max} - y_1$	Top

Observation

- If $p_k = 0$, the line is parallel to the corresponding boundary and

  $q_k < 0,$ the line is completely outside the boundary and can be eliminated;

  $q_k \geq 0,$ the line is inside the boundary and needs further consideration;

- If $p_k < 0,$ the extended line proceeds from the outside to the inside of the corresponding boundary line.
- If $p_k > 0,$ the extended line proceeds from the inside to the outside of the corresponding boundary line.
- When $p_k \neq 0,$ the value of u that corresponds to the intersection point is q_k / p_k

Liang-Barsky - Algorithm

- If $p_k=0$ and $q_k<0$ for any k, eliminate the line and stop. Otherwise proceed to the next step.
- For all k such that $p_k<0$, calculate $r_k= q_k/p_k$. Let u_1 be the maximum of the set containing 0 and the calculated r values.
- For all k such that $p_k>0$, calculate $r_k= q_k/p_k$. Let u_2 be the minimum of the set containing 1 and the calculated r values.
- If $u_1 > u_2$, eliminate the line since it is completely outside the clipping window. Otherwise, use u_1 and u_2 to calculate the end points of the clipped line.

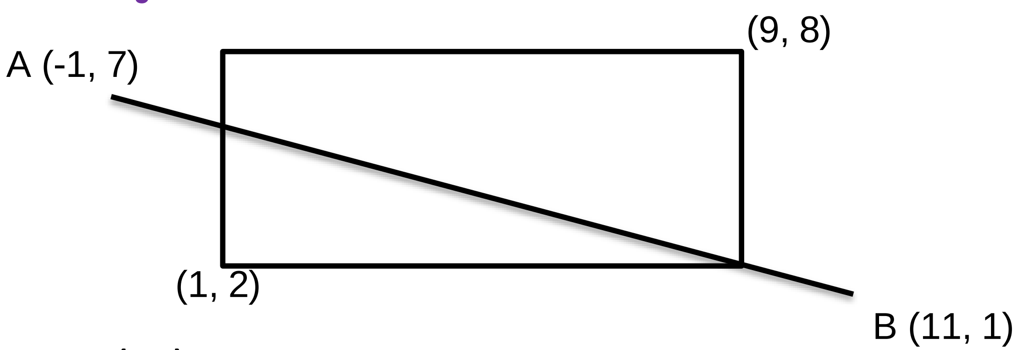
Line Clipping – Liang-Barsky

- If u1 > u2, the line lies completely outside of the clipping area.
- Otherwise the segment from u1 to u2 lies inside the clipping window.

Summary

- Calculate:
  - $p_1 = -\Delta X$ $\quad q_1 = X_1 - X_{min}$
  - $p_2 = \Delta X$ $\quad q_2 = X_{max} - X_1$
  - $p_3 = -\Delta Y$ $\quad q_3 = Y_1 - Y_{min}$
  - $p_4 = \Delta Y$ $\quad q_4 = Y_{max} - Y_1$
- If $p_k = 0$: line is parallel to the window.
  - If $q_k < 0$, **line is completely outside**.
  - Otherwise, need clipping.
- If $p_k < 0$:
  - $u_1 = \text{Max}(0, q_k / p_k)$.
- If $p_k > 0$:
  - $u_2 = \text{Min}(1, q_k / p_k)$.
- If $u_1 > u_2$: **line is completely outside**
- Otherwise: Clip accordingly-
  - $X = X_1 + u*\Delta X$
  - $Y = Y_1 + u*\Delta Y$

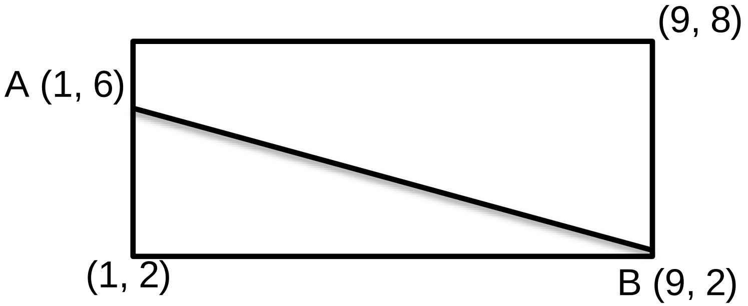
Example



- $\Delta X = 11 - (-1) = 12$; $\Delta Y = 1 - 7 = -6$
  - $p_1 = -12$ $\quad$ $q_1 = -2$
  - $p_2 = 12$ $\quad$ $q_2 = 10$
  - $p_3 = 6$ $\quad$ $q_3 = 5$
  - $p_4 = -6$ $\quad$ $q_4 = 1$
- Here, none of $p_k = 0$: line is not parallel to the window.
- $p_k < 0$ for k = 1 & 4:
  - $u_1 = \text{Max}(0, q_k / p_k) = \text{Max}(0, (-2/-12), (1/-6)) = 1/6$
- $p_k > 0$ for k = 2 & 3:
  - $u_2 = \text{Min}(1, q_k / p_k) = \text{Min}(1, (10/12), (5/6)) = 5/6$
- Here, $u_1 < u_2$: need clipping.
  - $X = X_1 + u * \Delta X$
  - $Y = Y_1 + u * \Delta Y$

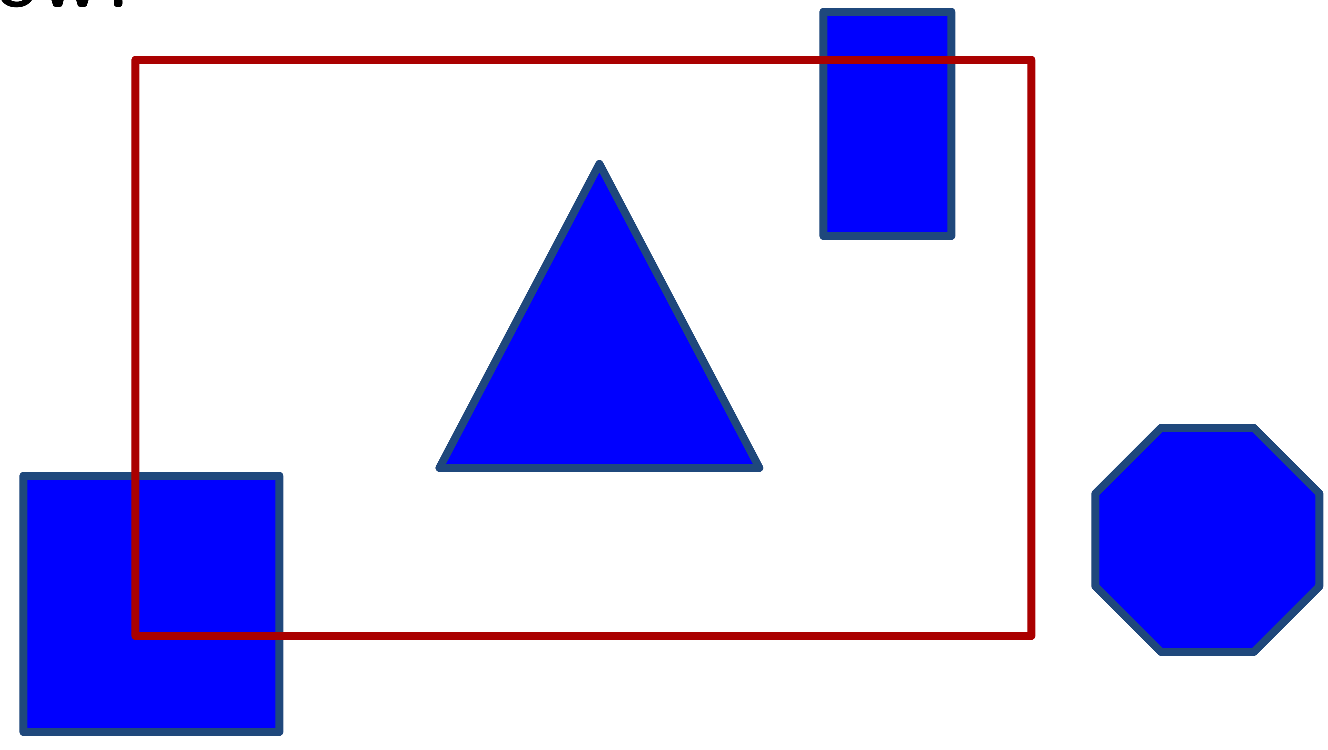
Continue…

- A' (X, Y) = (1, 6)
  - $X = X_1 + u_1 * \Delta X$
  - $Y = Y_1 + u_1 * \Delta Y$
- B' (X, Y) = (9, 2)
  - $X = X_1 + u_2 * \Delta X$
  - $Y = Y_1 + u_2 * \Delta Y$



Polygon Clipping

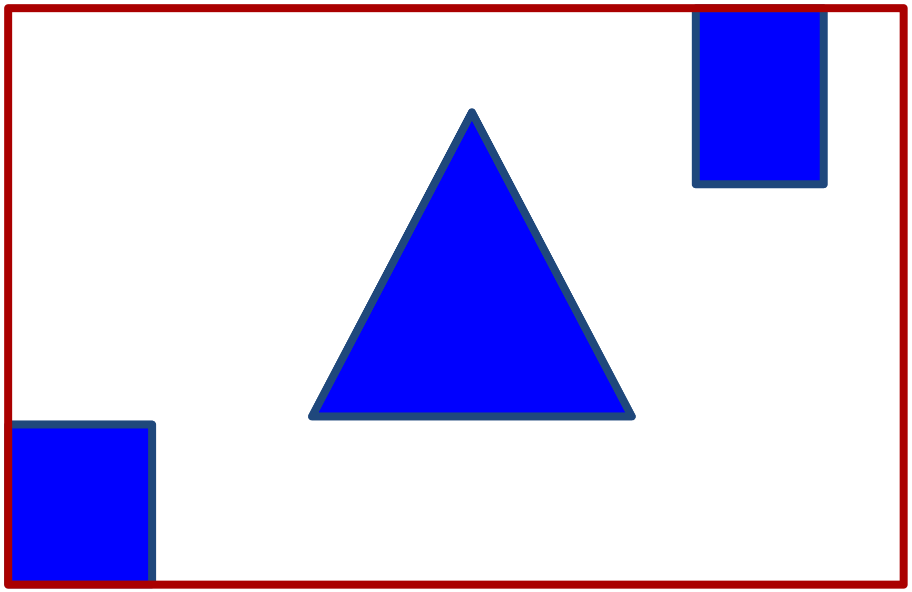
- Find the Part of a Polygon Inside the Clip Window?



Before Clipping

Polygon Clipping

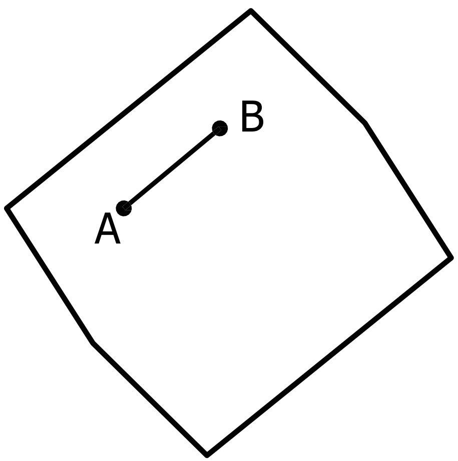
- Find the Part of a Polygon Inside the Clip Window?



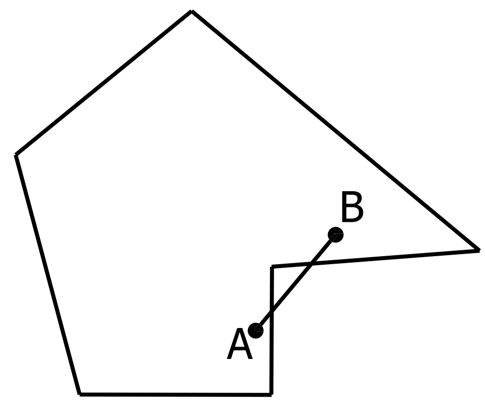
After Clipping

Polygon Clipping

- Convex Polygonal Clipping Windows:
  - A polygonal is called convex if the line joining any two interior points of the polygon lies completely inside the polygon.
  - A non-convex polygon is said to be concave.



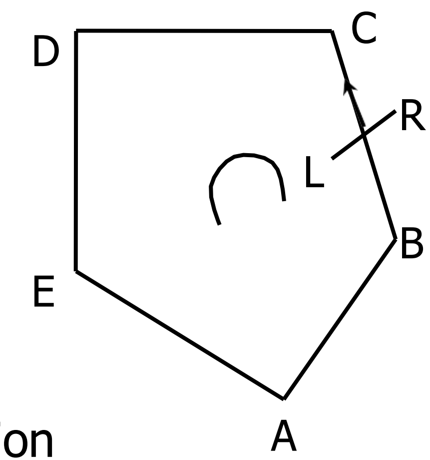
Convex Polygon



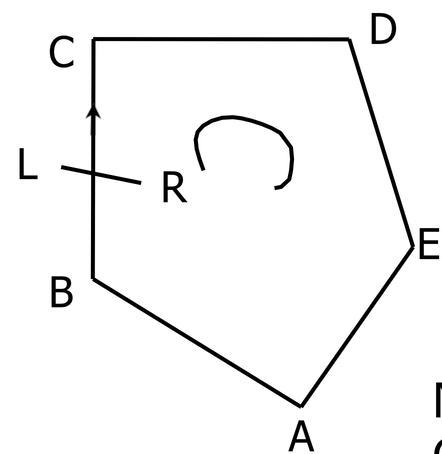
Concave Polygon

Polygon Clipping

- A Polygon with vertices P_1 .....P_N (and edges P_iP_{i-1} and P_1P_N) is said to be positively oriented if a tour of the vertices in the given order produces acounterclockwise circuit.
- The left hand of a person standing along any directed edge $\overline{P_iP_{i-1}}$ or $\overline{P_1P_N}$ would be pointing inside the polygon.



Positive Orientation



Negative Orientation

Polygon Clipping

- $A(x_1,y_1)$ and $B(x_2,y_2)$ be the end points of a directed line segment.
- A point $p(x,y)$ will be to the left of the line segment if the expression $C=(x_2-x_1)(y-y_1)-(y_2-y_1)(x-x_1)$ is positive.
- The point is to the right of the line segment if this quantity is negative.
- If a point p is to the right of any one edge of a positively oriented, convex polygon, it is outside the polygon.
- If it is to the left of every edge of the polygon, it is inside the polygon.

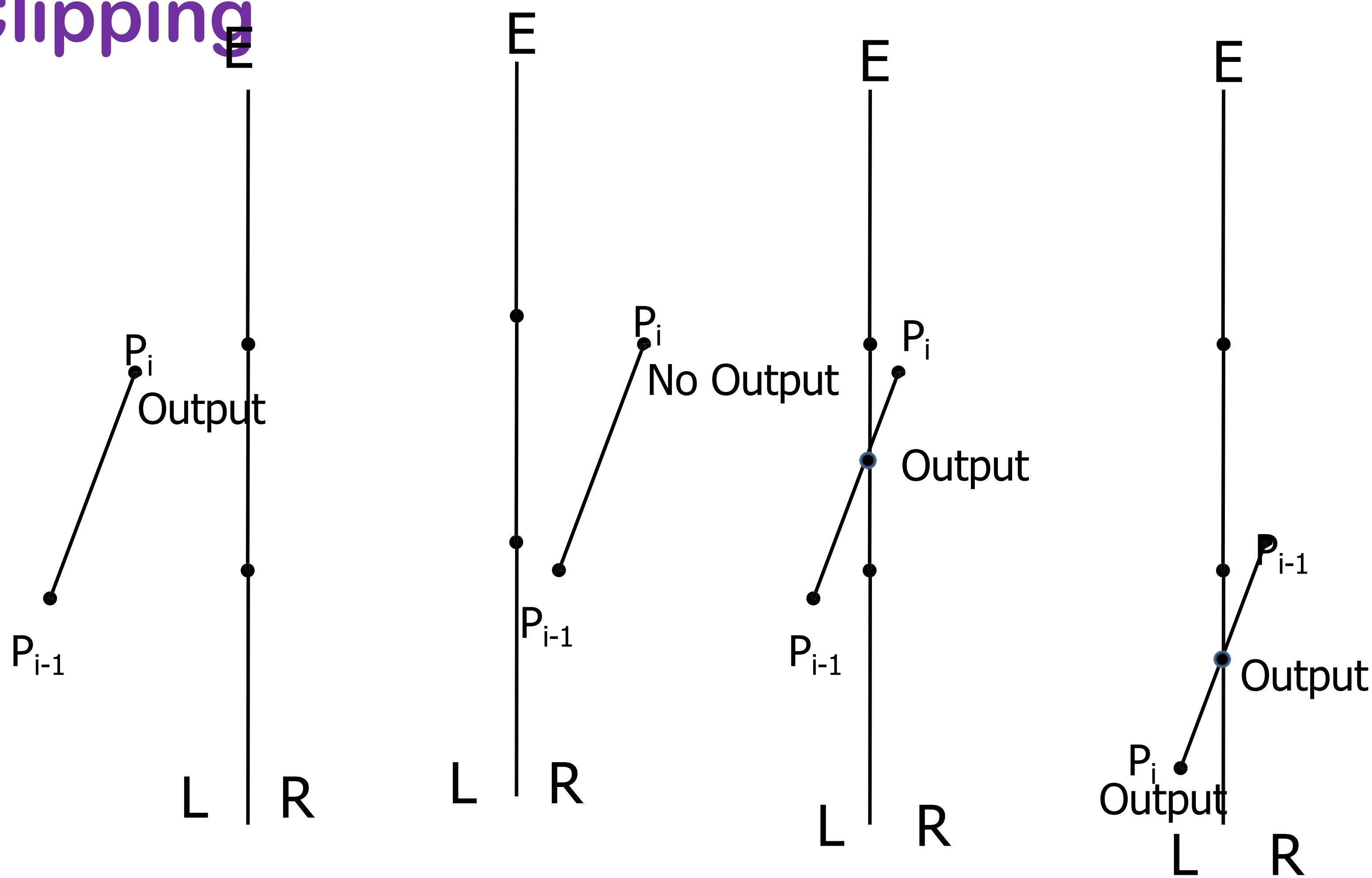
Sutherland-Hodgeman Polygon Clipping

- Let P_1 .....P_N be the vertex list of the polygon to be clipped. Let edge E, determined by endpoints A and B, be any edge of the positively oriented, convex clipping polygon.
- Clip each edge of the polygon in turn against the edge E of the clipping polygon, forming a new polygon whose vertices are determined as follows:

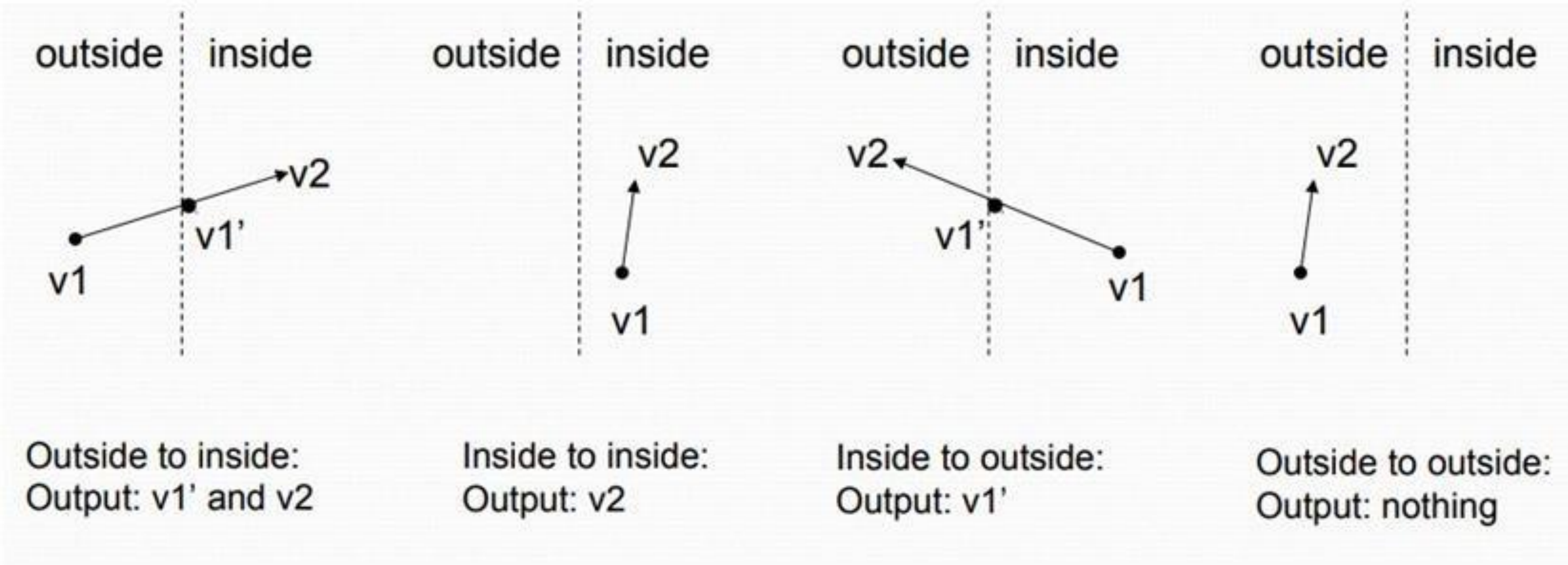
Sutherland-Hodgeman Polygon Clipping

- Consider the edge $\overline{P_{i-1}P_i}$
- If both P_{i-1} and P_i are to the left of the edge, vertex P_i is placed on the vertex output list of the clipped polygon
- If both P_{i-1} and P_i are to the right of the edge, nothing is placed on the vertex output list of the clipped polygon
- If both P_{i-1} to the left and P_i is to the right of the edge E, the intersection point *I* of the line segment $\overline{P_{i-1}P_i}$ with the extended edge E is calculated and placed on the vertex output list.
- If both P_{i-1} to the right and P_i is to the left of the edge E, the intersection point *I* of the line segment $\overline{P_{i-1}P_i}$ with the extended edge E is calculated. Both *I* and P_i are placed on the vertex output list.

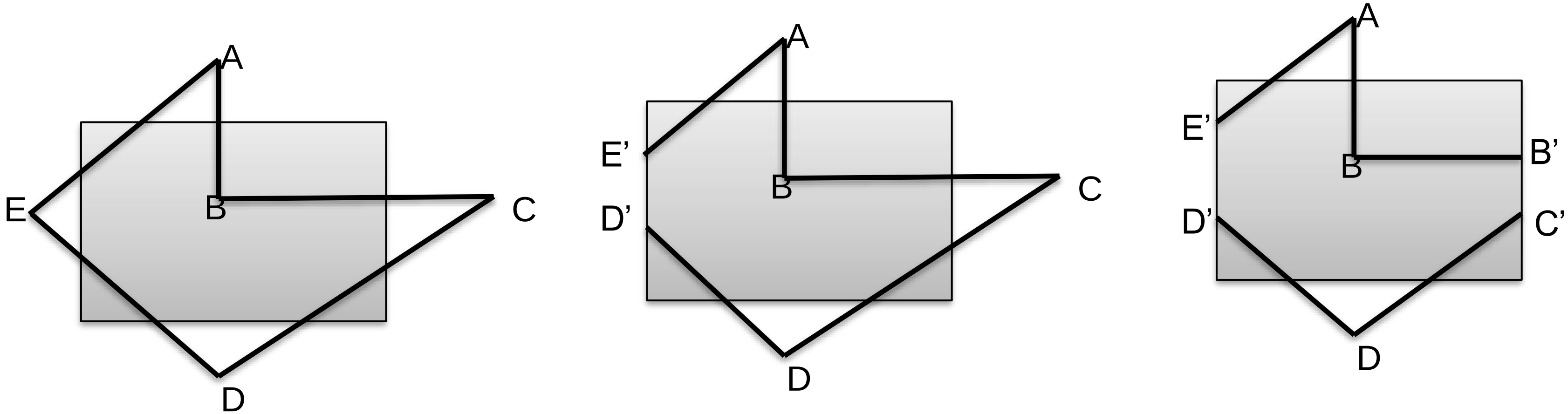
Sutherland-Hodgeman Polygon Clipping



Sutherland-Hodgeman Polygon Clipping



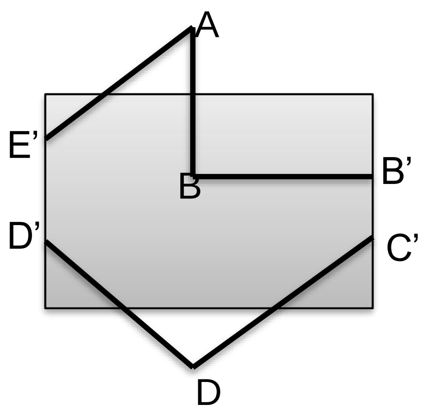
Sutherland-Hodgeman Polygon Clipping

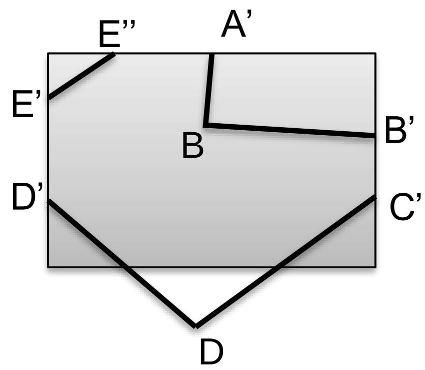


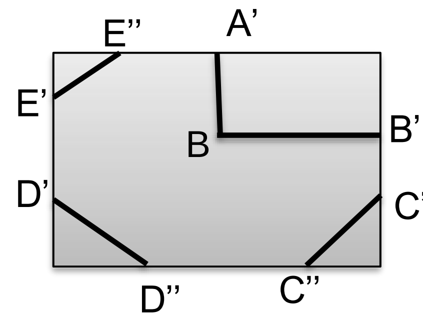
Left Clip		
Edge	**Case**	**Output**
AB	in-in	B
BC	in-in	C
CD	in-in	D
DE	in-out	D’
EA	out-in	E’A

Right Clip		
Edge	**Case**	**Output**
AB	in-in	B
BC	in-out	B’
CD	out-in	C’D
DD’	in-in	D’
D’E’	in-in	E’
E’A	in-in	A

Sutherland-Hodgeman Polygon Clipping





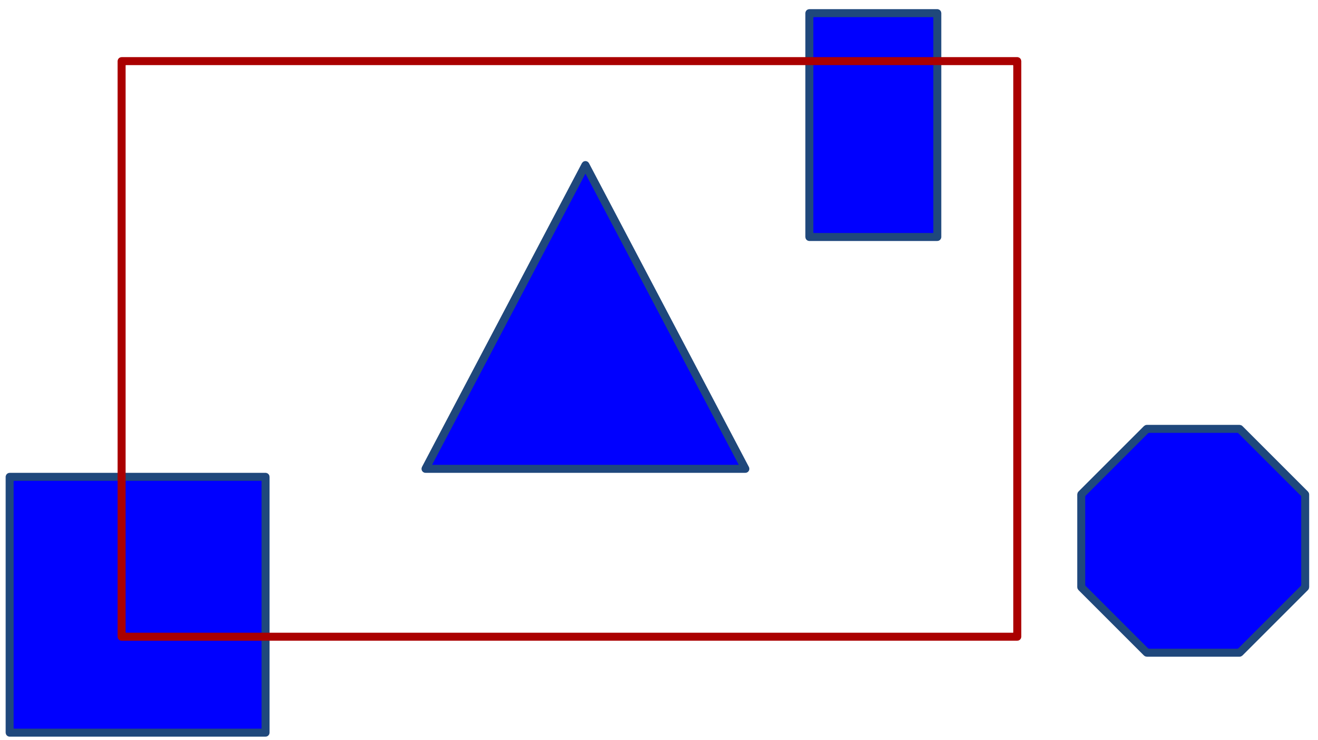


Top Clip		
Edge	**Case**	**Output**
AB	out-in	A’B
BB’	in-in	B’
B’C’	in-in	C’
C’D	in-in	D
DD’	in-in	D’
D’E’	in-in	E’
E’A	in-out	E’’

Bottom Clip		
Edge	**Case**	**Output**
A’B	in-in	B
BB’	in-in	B’
B’C’	in-in	C’
C’D	in-out	C’’
DD’	out-in	D’’D’
D’E’	in-in	E’
E’E’’	in-in	E’’
E’’A’	in-in	A’

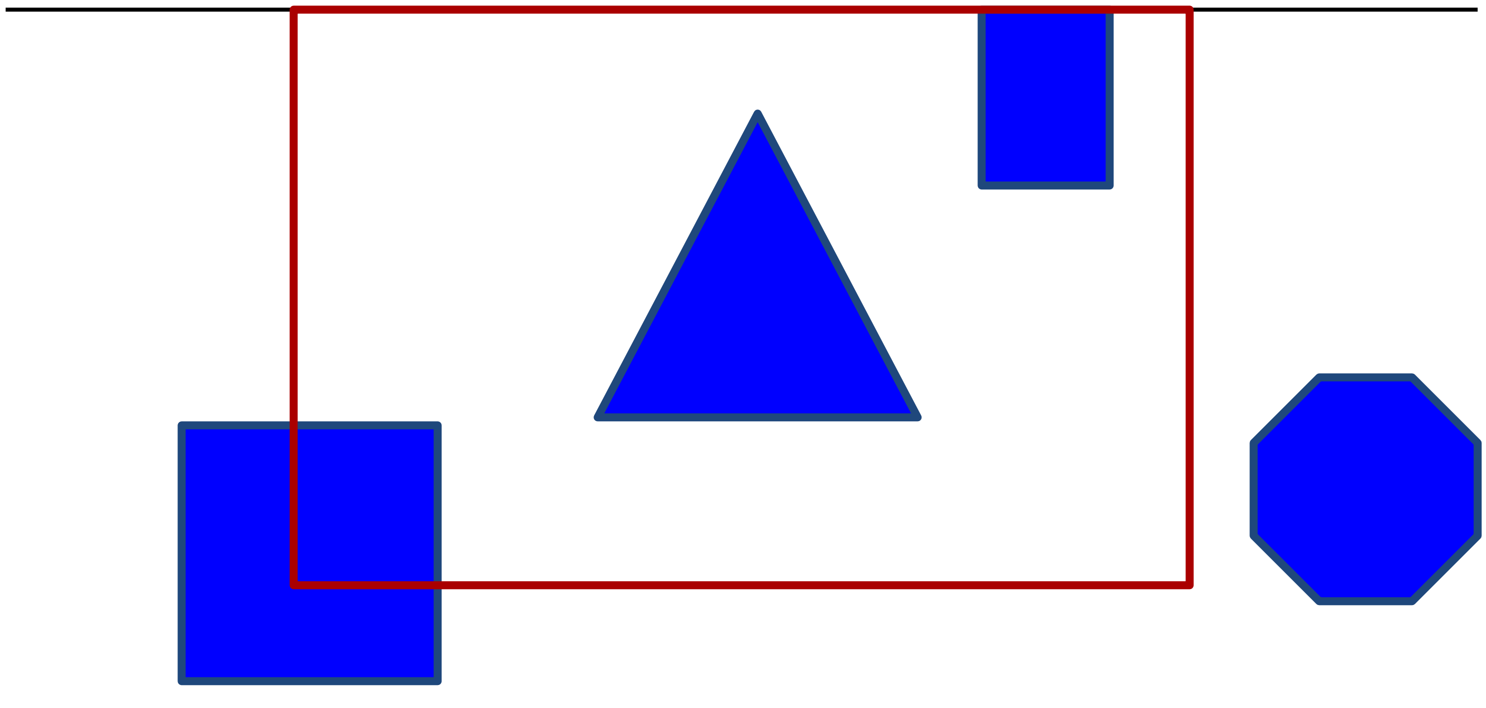
Sutherland-Hodgeman Polygon Clipping

- Clip to Each Window Boundary One at a Time



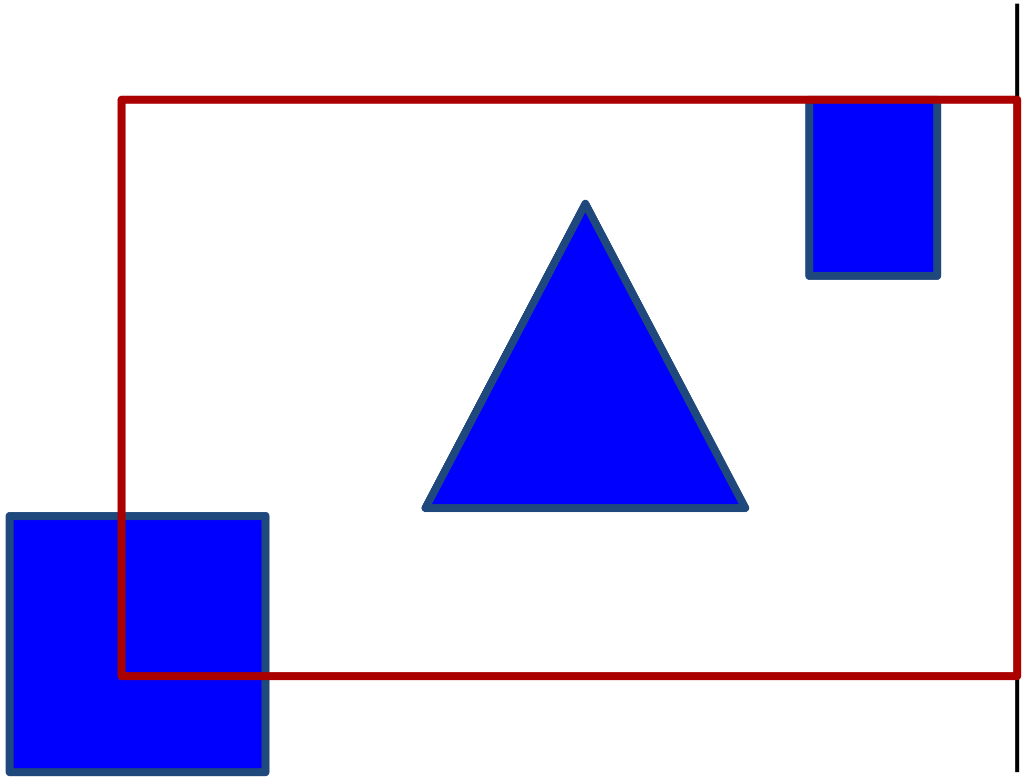
Sutherland-Hodgeman Polygon Clipping

- Clip to Each Window Boundary One at a Time



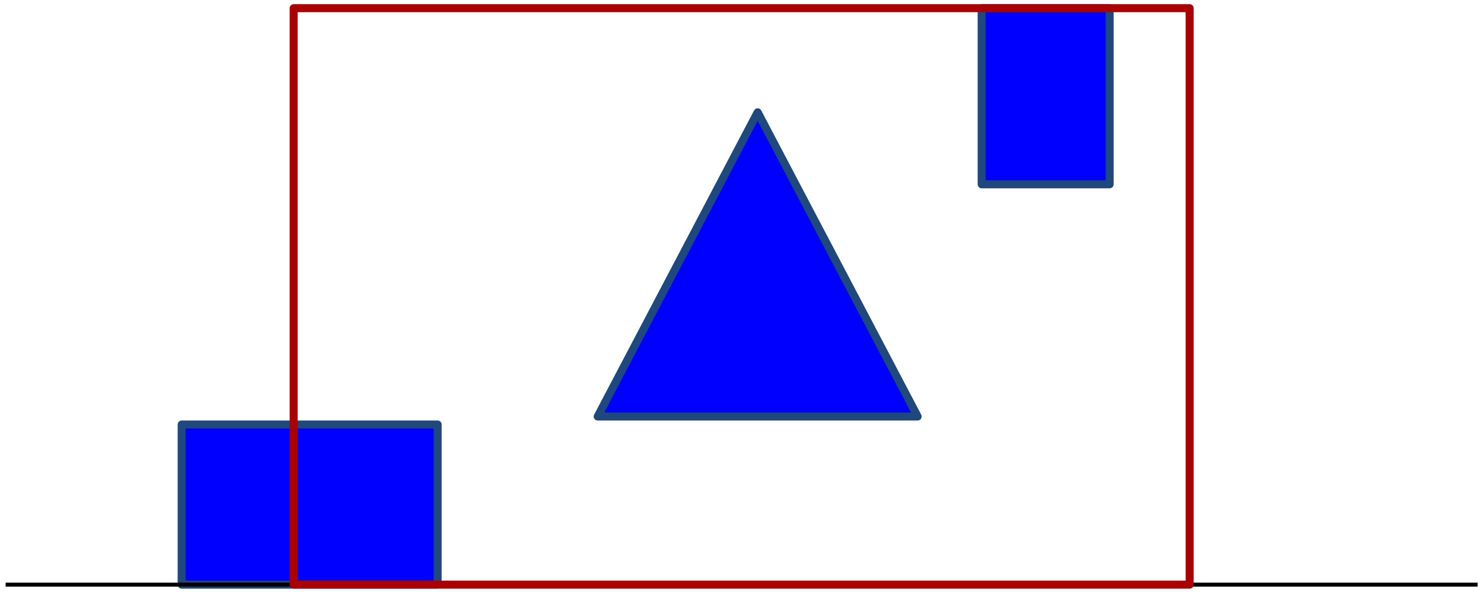
Sutherland-Hodgeman Polygon Clipping

- Clip to Each Window Boundary One at a Time



Sutherland-Hodgeman Polygon Clipping

- Clip to Each Window Boundary One at a Time



Sutherland-Hodgeman Polygon Clipping

- Clip to Each Window Boundary One at a Time