2D Viewing & Clipping

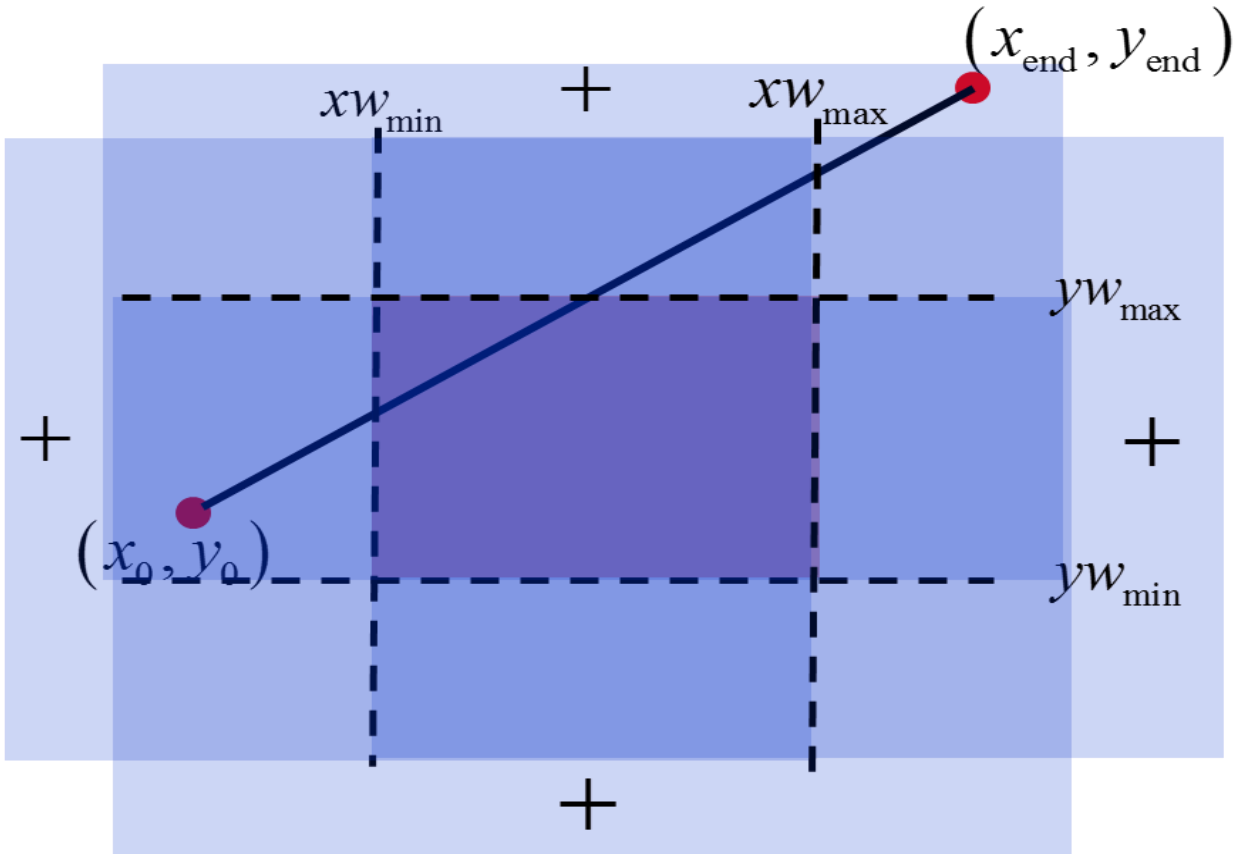
By
Poonam Saini
Dept. of Computer Science & Engineering
Sir Padampat Singhania University
Udaipur

2D Viewing & Clipping

Liang-Barsky Line Clipping Algorithm

Treat undecided lines in Cohen-Sutherland more efficiently.

Define clipping window by intersections of four half-planes.



Liang-Barsky Line Clipping Algorithm

- Consider the parametric definition of a line:

$x = x_1 + u\Delta x$

$y = y_1 + u\Delta y$

Where $0 \leq u \leq 1$

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

Liang-Barsky Line Clipping Algorithm

- Clipping condition in parametric form

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

$$y_{min} \le y_1 + v\Delta y \le y_{max}$$

- Rearranging, we get
  - $-u\Delta x \le (x_1 - x_{min})$
  - $u\Delta x \le (x_{max} - x_1)$
  - $-v\Delta y \le (y_1 - y_{min})$
  - $v\Delta y \le (y_{max} - y_1)$
- In general: $u * p_k \le q_k$

Liang-Barsky Line Clipping Algorithm

- In general: $u * p_k \leq q_k$, k=1,2,3,4

$p_1 = -\Delta X$ $\quad q_1 = X_1 - Xwmin$ (Left)

$p_2 = \Delta X$ $\quad q_2 = Xwmax - X_1$ (Right)

$p_3 = -\Delta Y$ $\quad q_3 = Y_1 - Ywmin$ (Bottom)

$p_4 = \Delta Y$ $\quad q_4 = Ywmax - Y_1$ (Top)

Liang-Barsky Line Clipping Algorithm

- **Cases:**
  1. **If $p_k = 0$, the line is parallel to boundaries and**
     - **If for the same k, $q_k < 0$, the line is completely outside the boundary and can be eliminated**
     - **If for the same k, $q_k >= 0$, the line is inside the boundary and needs further condsideration**
  2. **If $p_k < 0$, the extended line proceeds from outside to inside of the corresponding boundary line.**

     $r_k = q_k / p_k$ and $u_1 = \max(0, r_k)$

Liang-Barsky Line Clipping Algorithm

- **Cases:**
  3. **If $p_k > 0$, the extended line proceeds from inside to outside of the corresponding boundary line.**
     $r_k = q_k / p_k$ and $u_2 = \min(1, r_k)$
  4. **If $u_1 > u_2$, the line is completely outside**

Example 1: Liang-Barsky Line Clipping

Problem :Let P1 (-1, -2), P2 (2, 4) be the line and XWmin= 0, XWmax = 1, YWmin = 0, YWmax = 1 be the clipping window boundaries. Find the visible portion of the line after clipping

Solution: (1) Δx = 2 - (-1) = 3 Δ y = 4 - (-2) = 6

(2)

- **P1 = - Δ x = -3 q1 = x1 - XWmin = -1 - 0 = -1 q1 / P1 = 1/3**
- **P2 = Δ x = 3 q2 = XWmax - x1 = 1 - (-1) = 2 q2 / P2 = 2/3**
- **P3 = - Δ y = -6 q3 = y1 - YWmin = -2 q3 / P3 = 1/3**
- **P4 = Δ y = 6 q4 = YWmax - y1 = 3 q4 / P4 = 1/2**

(3) for (Pi < 0) u1="MAX" (1 / 3, 1 / 3, 0)=1/ 3

(4) for (Pi > 0) u2 = MIN (2 / 3, 1 / 2, 1) = 1 / 2

(5) Since u1 < u2 there is a visible section

x=x1+u1 Δx=-1+(1/3*3)=0

y=y1+u1 Δy=-2+(1/3*6)=0

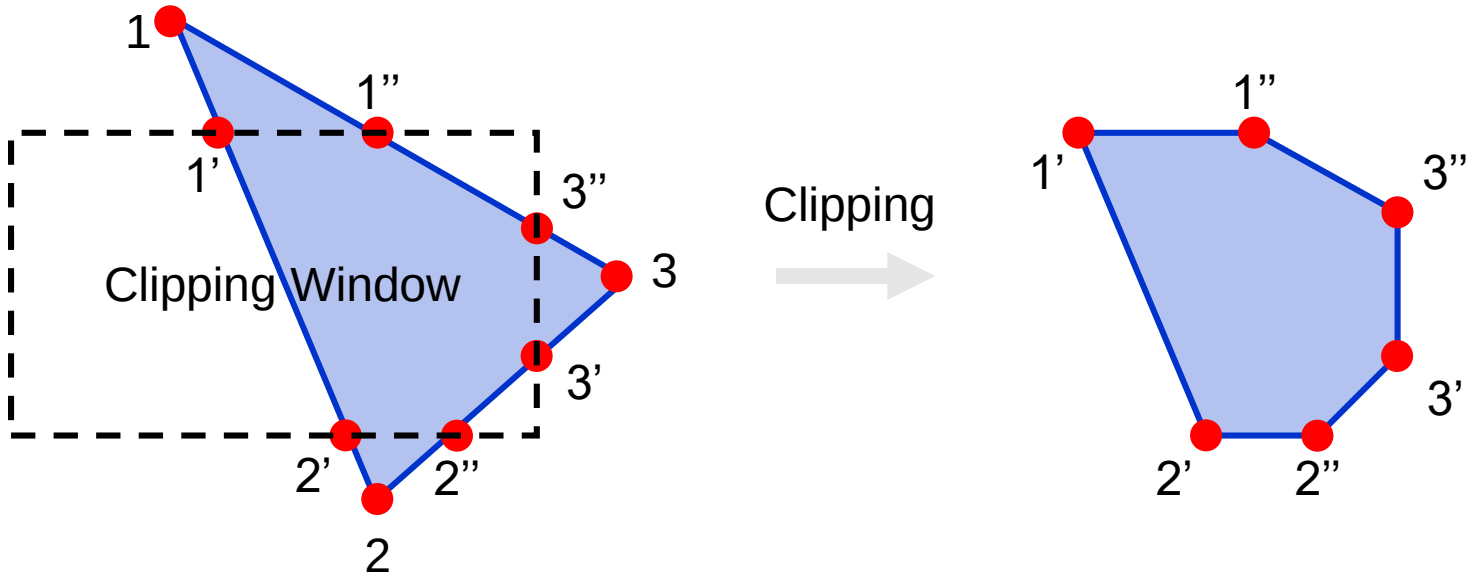
x'=x1+u2 Δx=-1+(1/2*3)=1/2=0.5

y'=y1+u2 Δy=-2+(1/2*6)=1

Differences

Liang-Barsky Algorithm	Cohen-Sutherland Algorithm
Less intersection calculations are required	Repeated intersection calculations are required
Each update of u1 and u2 requires only one division	Each intersection calculation requires both division and multiplication

Polygon Clipping



A polygon boundary processed by a line clipping clipper may be displayed as a series of unconnected line segments depending on the orientation of the polygon to the clipping window. But we want to display a bounded region after clipping as shown in figure above.

The output of the polygon clipping algorithm should be a sequence of vertices that defines the clipped polygon boundaries.

Sutherland-Hodgeman Polygon Clipping

- **A polygon can be clipped correctly by processing the polygon boundary as a whole against each window edge.**
- **Process all polygon vertices against each clip rectangle boundary in turn.**
- **Steps:**

➢ **Begin with initial set of polygon vertices, first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.**

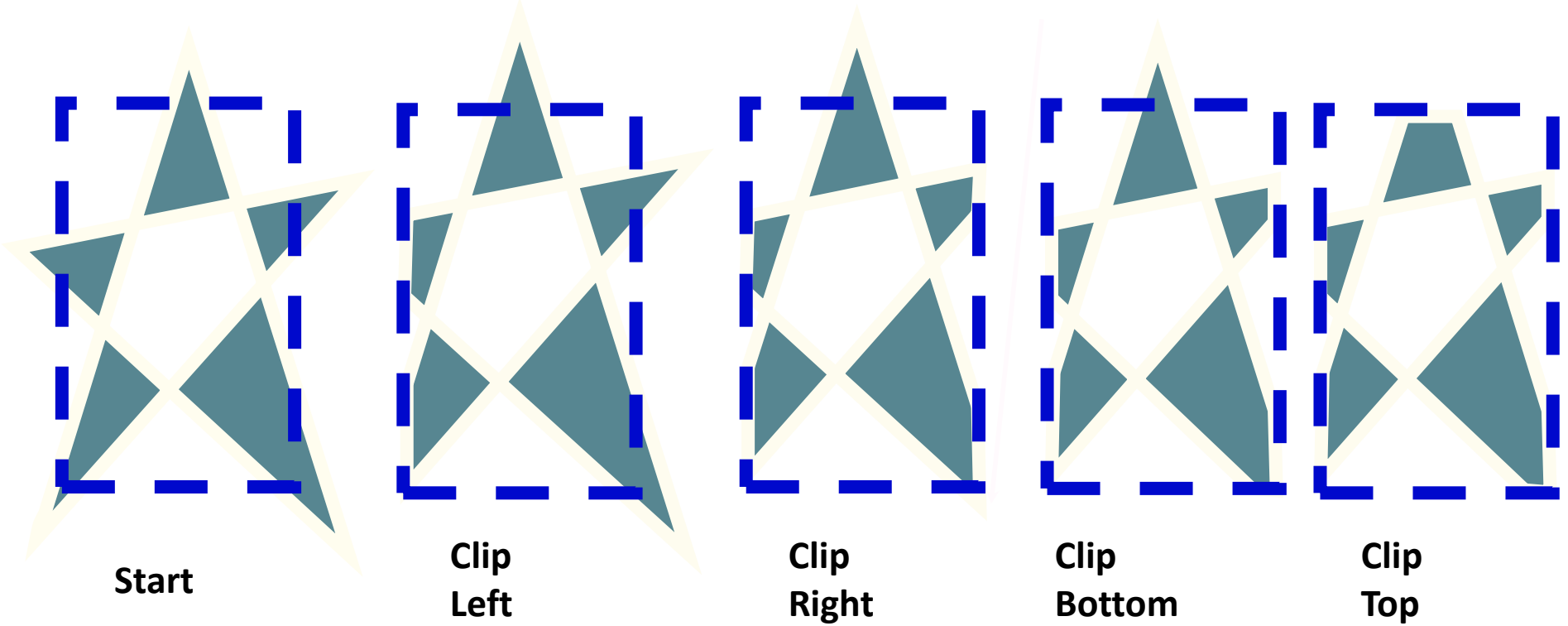
➢ **The new set of vertices could then be successively passed to a right boundary clipper, a bottom boundary clipper and to a top boundary clipper.**

Sutherland-Hodgeman Polygon Clipping

- **Steps:**
  - **Begin with initial set of polygon vertices, first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.**
  - **The new set of vertices could then be successively passed to a right boundary clipper, a bottom boundary clipper and to a top boundary clipper.**
  - **At each step a new sequence of output vertices is generated and passed to the next window boundary clipper.**

Sutherland-Hodgeman Polygon Clipping

Example : Clipping a polygon against successive window boundaries



Sutherland-Hodgeman Polygon Clipping

There are four possible cases when vertices are processed in sequence around the perimeter of the polygon:

1. **If the first vertex is outside the window boundary and the second vertex is inside the window boundary, both the intersection of the polygon edge with the window boundary and the second vertex is added to the output vertex list.**
2. **If both input vertices are insides the window boundary, only the second vertex is added to the output vertex list.**

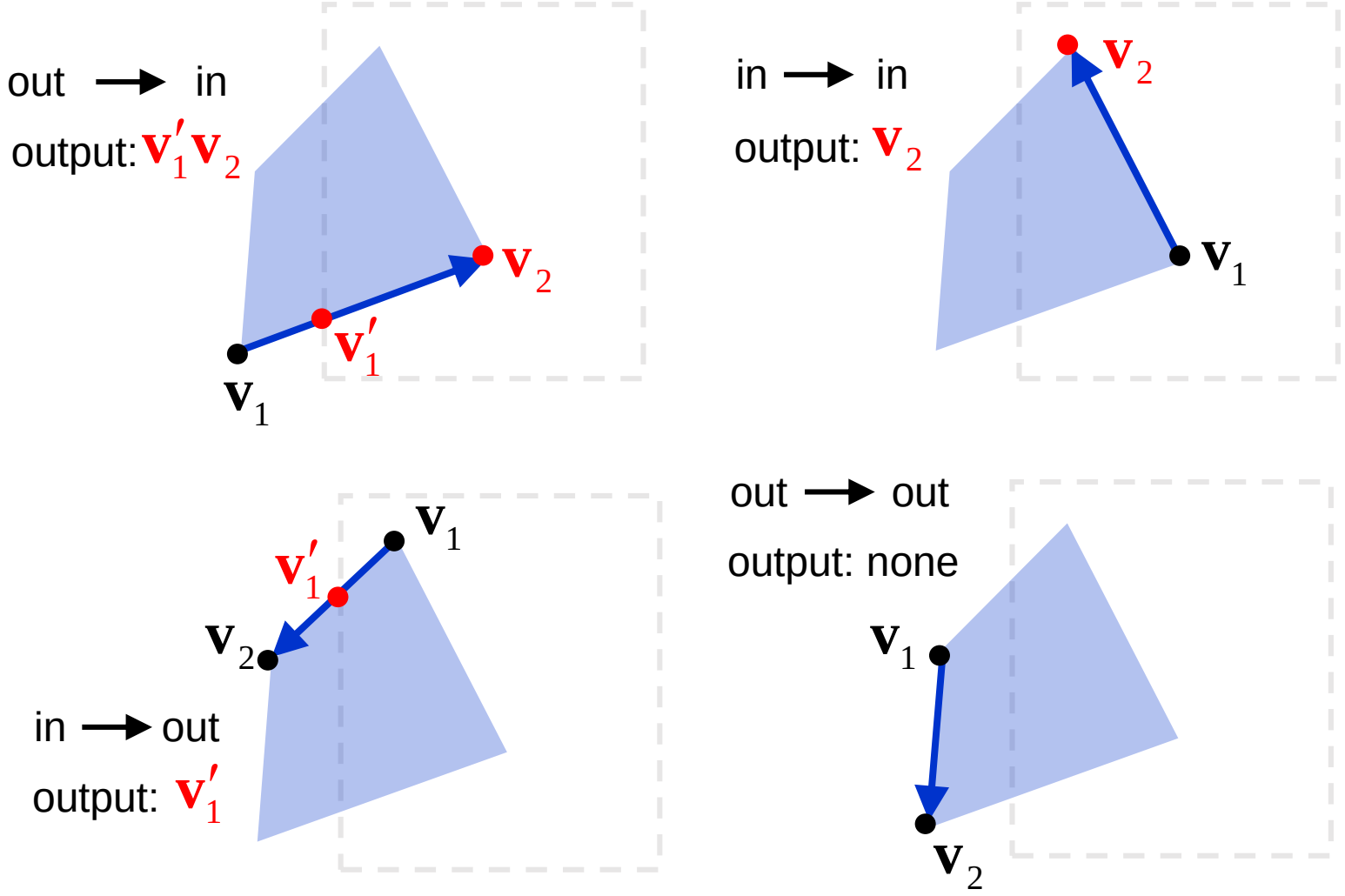
Sutherland-Hodgeman Polygon Clipping

(contd.)

3. **If the first vertex is inside the window boundary and second vertex is outside, only the edge intersection with the boundary is added to the output vertex list.**
4. **If both input vertices are outside the window boundary, nothing is added to the vertex output list.**

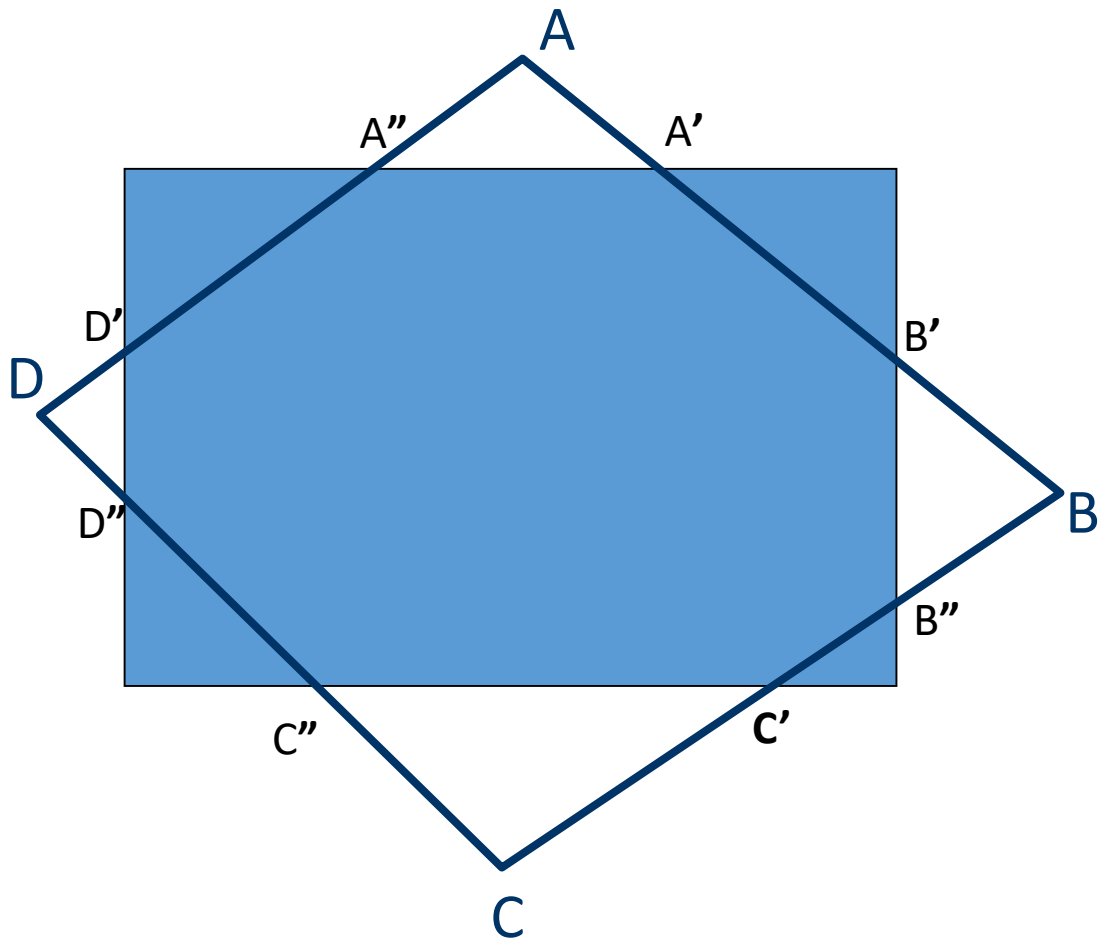
Note: Once all the vertices have been processed for one clip window boundary the output list of vertices is clipped against the next window boundary.

Example: Successive processing of pairs of polygon vertices against the left window boundary



Sutherland-Hodgeman Polygon Clipping

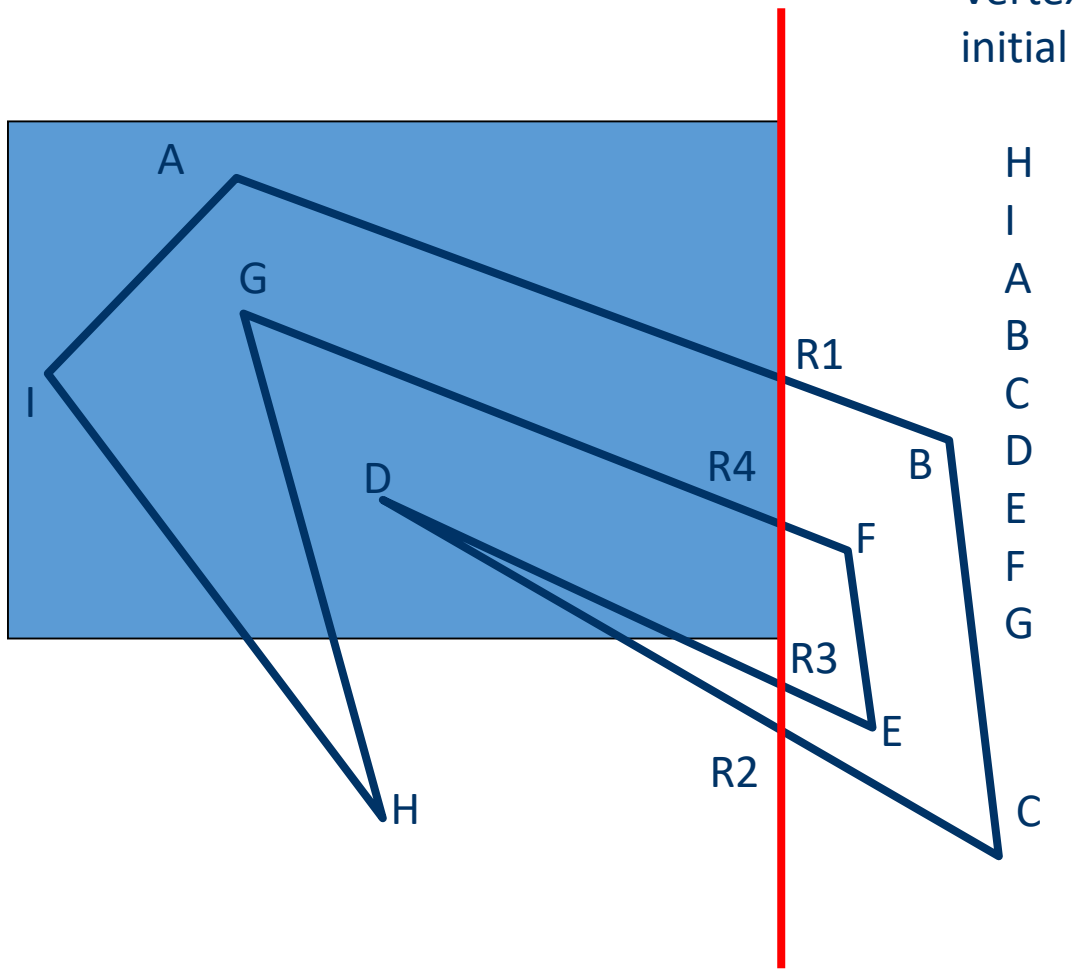
(Excercise)



Vertex List

initial	after left	after Right	after top	after bottom
A	D’	A	A’	A’
B	A	B’	B’	B’
C	B	B”	B”	B”
D	C	C	C	C’
	D”	D”	D”	C”
		D’	D’	D”
			A”	D’
				A”

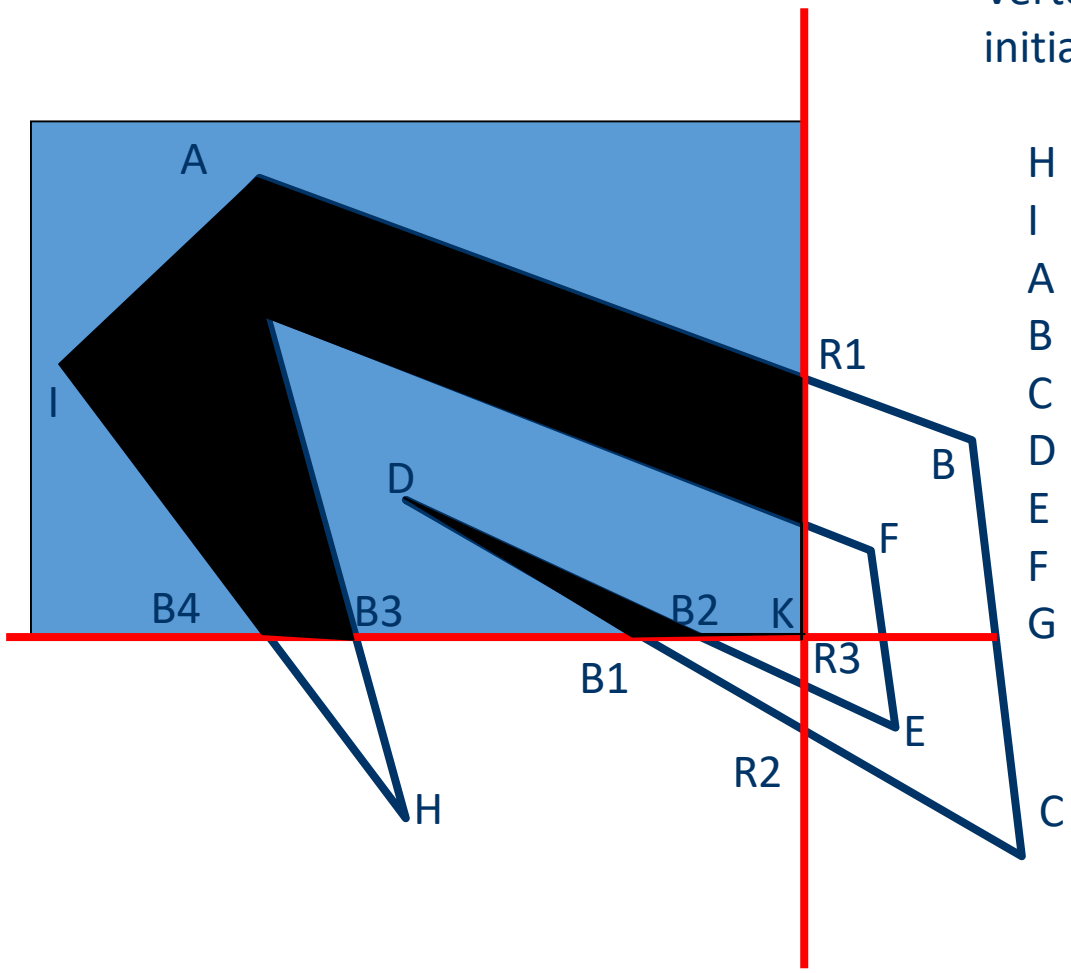
Sutherland-Hodgeman Exercise 2



Vertex List

initial	after left	after top	after right	after bottom
H				
I	I			
A	A	A		
B	B	B		
C	C	C		
D	D	D		
E	E	E		
F	F	F		
G	G	G		
	H	H		
		I		

Sutherland-Hodgeman Exercise 2



Vertex List

initial	after left	after top	after right	after bottom
H				
I	I			
A	A	A	R1	
B	B	B	R2	B1
C	C	C	D	D
D	D	D	R3	B2
E	E	E	R4	K
F	F	F	G	R4
G	G	G	H	G
	H	H	I	B3
		I	A	B4
				I
				A
				R1

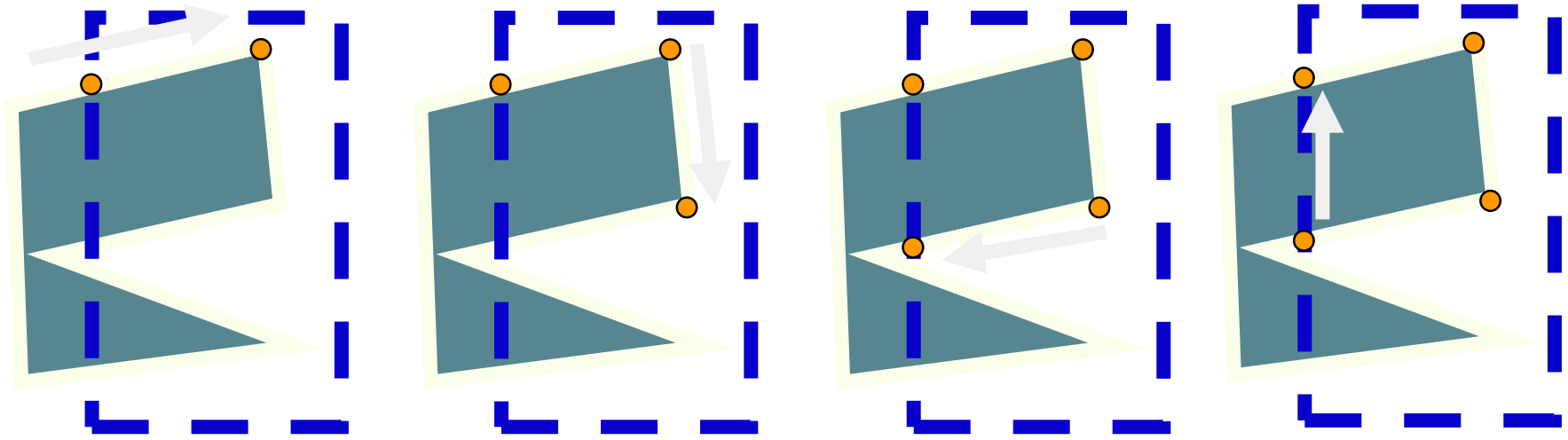
Weiler-Atherton Polygon Clipping

- **The vertex processing procedures for window boundaries are modified so that concave polygons are displayed correctly.**
- **We can follow the clockwise or anticlockwise path for polygon processing whether the pair of polygon vertices currently being processed represent outside-to-inside pair or an inside-to-outside pair.**
- **For clockwise processing of polygon vertices, use the following:**

  (a) For an outside-to-inside pair of vertices follow the polygon boundary.

  (b) For an inside-to-outside pair of vertices, follow the window boundary in clockwise direction.

Weiler-Atherton Polygon Clipping Example



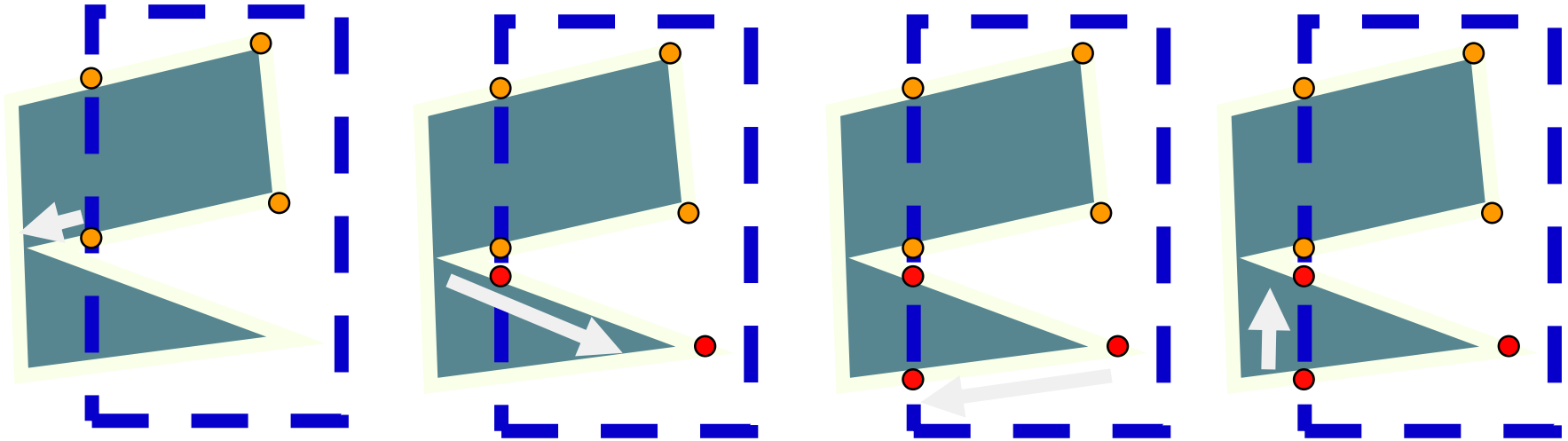
Out -> In
Add clip vertex
Add end vertex

In -> In
Add end vertex

In -> Out
Add clip vertex
Cache old direction

Follow clip edge until
(a) new crossing found
(b) reach vertex already added

Weiler-Atherton Polygon Clipping Example(contd..)



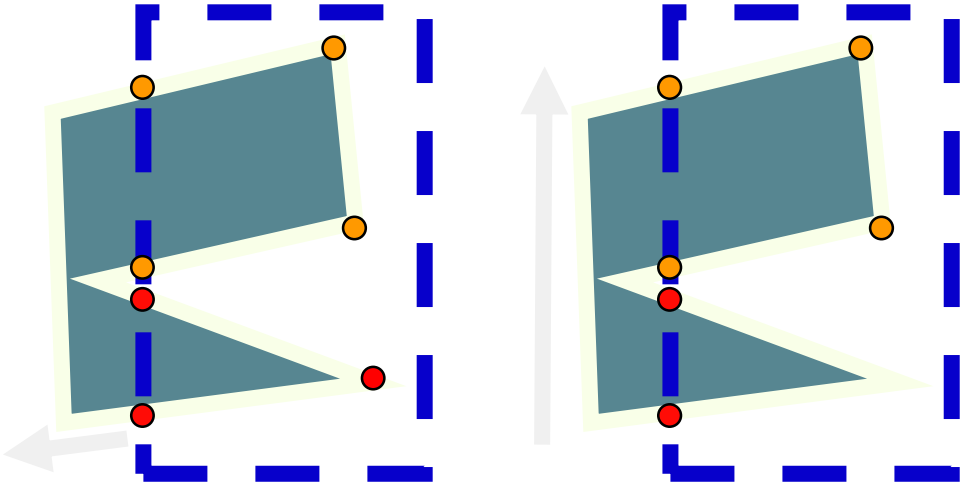
Continue from cached vertex and direction

Out -> In
Add clip vertex
Add end vertex

In -> Out
Add clip vertex
Cache old direction

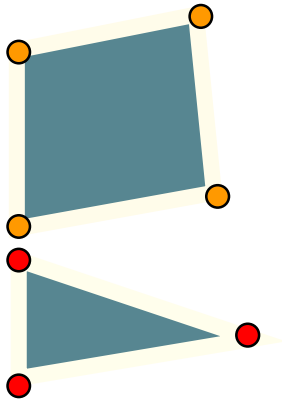
Follow clip edge until
(a) new crossing found
(b) reach vertex already added

Weiler-Atherton Polygon Clipping Example(contd..)



Continue from cached vertex and direction

Nothing added
Finished



Final Result:
2 unconnected polygons