Algorithm of Liang-Barsky Line Clipping:

Step 1: Set the endpoints of the line **(x_1, y_1)** and **(x_2, y_2).**

Step 2: Calculate the value of **p_1, p_2,p_3, p_4** and **q_1, q_2, q_3,q_4.**

Step 3: Now we calculate the value of **t**

t_1 = 0 (For initial point)

t_2 = 1 (For final point)

Step 4: Now, we have to calculate the value of **p_k** and **q_k**

If

p_k = 0

then

{The line is parallel to the window}

If

q_k < 0

then

{The line is completely outside the window}

Step 5: If we have non zero value of **p_k** –

If

p_k < 0

then

t_1 = max (0, q_k / p_k)

If

p_k > 0

then

t_2 = min (1, q_k / p_k)

Now, if **t_1 < t_2 {**If **t_1** value is changed

Then the first point is outside the window.

If **t_2** value is changed

Then the second point is outside the window**}**

else

t_1 > t_2

then

{Line is completely outside the window}

Step 6: Stop.

Example:

Let a rectangular window size with (5, 9). The points of the line are (4, 12) and (8, 8). Use the Liang- Barsky algorithm to clip the line and find the intersection point.

Solution:

We have,

The initial point of the line (p_1) = (4, 12)

The ending point of the line (p_2) = (8, 8)

$x_1 = 4$, $x_2 = 8$

$y_1 = 12$, $y_2 = 8$

$xw_{min} = 5$, $xw_{max} = 9$

$yw_{min} = 5$, $yw_{max} = 9$

Step 1: We have to calculate the value of **?x** and**?y-**

?x = $x_2 – x_1 = 8-4 = 4$

?y = $y_2 – y_1 = 8-12 = -4$

Step 2: Now, we will calculate-

$p_1 = -4$ $q_1 = 4-5 = -1$
$p_2 = 4$ $q_2 = 9-4 = 5$
$p_3 = 4$ $q_3 = 12-5 = 7$
$p_4 = -4$ $q_4 = 9-12 = -3$

Step 3: Now we will calculate t_1 value–

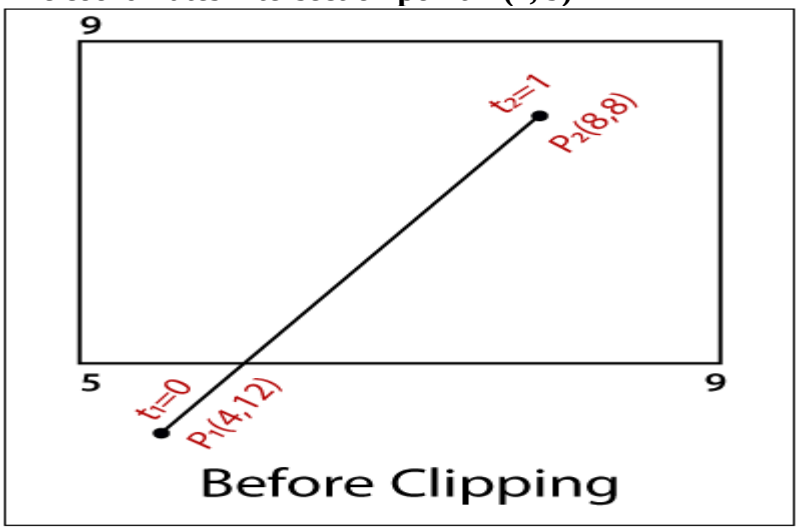
If $p_1, p_4 < 0$
Then $t_1 = \max(0, q_k/p_k)$
$= \max(0, q_1/p_1, q_4/p_4)$
$= \max(0, 1/4, 3/4)$
$t_1 = 3/4$

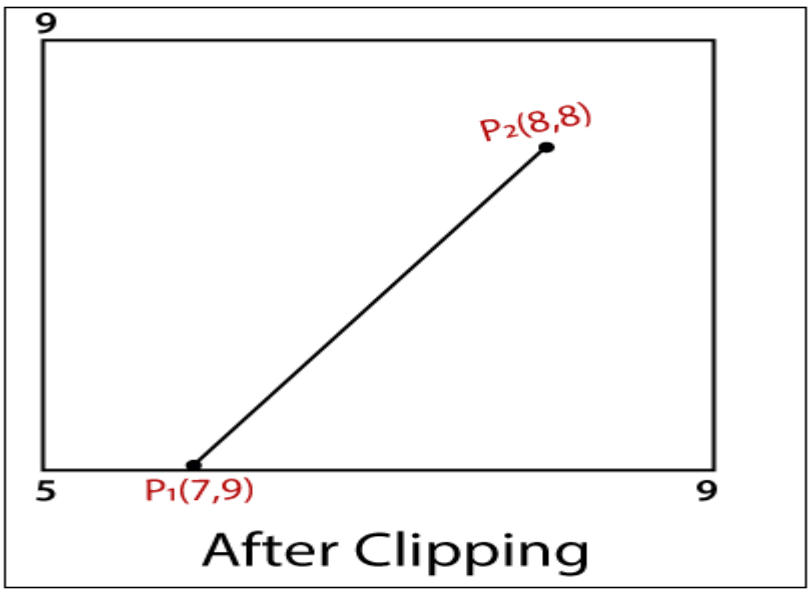
If $p_2, p_3 > 0$
Then $t_2 = \min(1, q_k/p_k)$
$= \min(1, q_2/p_2, q_3/p_3)$
$= \min(1, 5/4, 7/4)$
$t_2 = 1$

Step 4: Now, we have to calculate the intersection point.

$x = x_1 + t_1.$?x= 4+ 3/4 * 4 = 7
$y = y_1 + t_1.$?y= 12+ 3/4 *(-4) = 9

The coordinates intersection point = (7, 9)

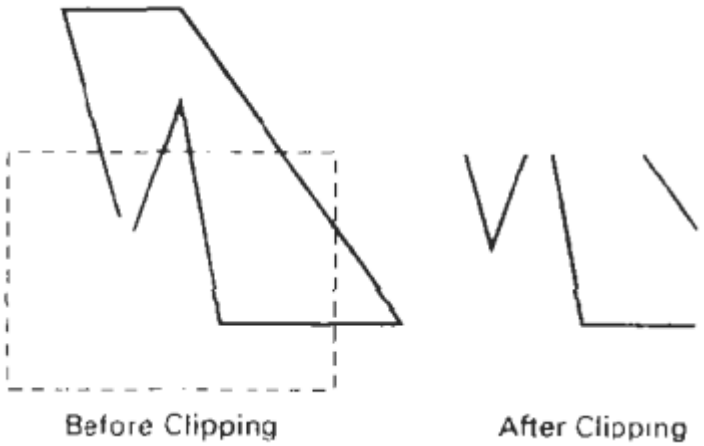




POLYGON CLIPPING

A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments (Fig. 6-17), depending on the orientation of the polygon to the clipping window.

For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill. The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries.



Sutherland-Hodgeman Polygon Clipping

We can correctly clip a polygon by processing the polygon boundary as a whole against each window edge.

Beginning with the initial set of polygon vertices, we could first clip the polygon against the left rectangle boundary to produce a new sequence of vertices. The new set of vertices could then successively passed to a right boundary clipper, a bottom boundary clipper, and a top boundary clipper, as in Fig. 6-19. At each step, a new sequence of output vertices is generated and passed to the next window boundary clipper.

There are four possible cases when processing vertices in sequence around the perimeter of a polygon. As each pair of adjacent polygon vertices is passed to a window boundary clipper, we make the following tests:

(1) If the first vertex is outside the window boundary and the second vertex is inside, both the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list.

(2) If both input vertices are inside the window boundary, only the second vertex is added to the output vertex list.

(3) li the first vertex is inside the window boundary and the second vertex is outside, only the edge intersection with the window boundary is added to the output vertex list.

(4) If both input vertices are outside the window boundary, nothing is added to the output list. These four cases are illustrated in Fig. 6-20 for successive pairs of polygon vertices. Once all vertices have been processed for one clip window boundary, the output 11st of vertices is clipped against the next window boundary.

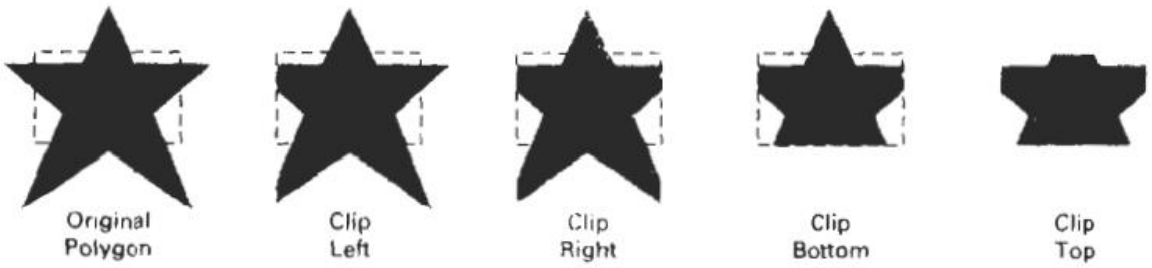


Figure 6-19
Clipping a polygon against successive window boundaries.

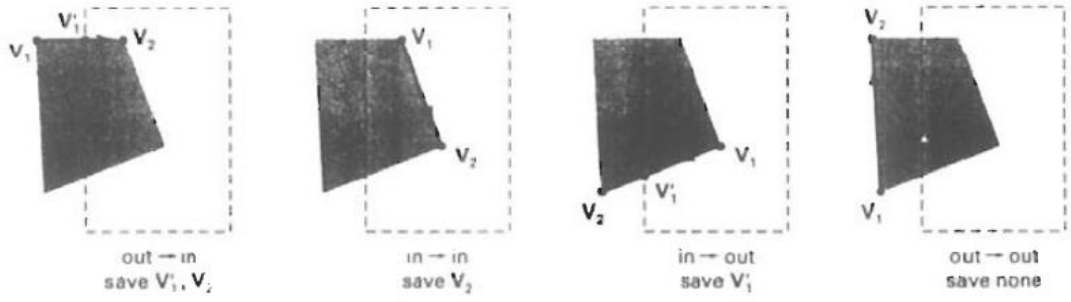


Figure 6-20
Successive processing of pairs of polygon vertices against the left window boundary.

Convex polygons are correctly clipped by the Sutherland-Hodgeman algorithm, but concave polygons may be displayed with extraneous lines, as demonstrated in Fig. 6-24. This occurs when the clipped polygon should have two or more separate sections. But since there is only one output vertex list, the last vertex in the list is always joined to the first vertex. There are several things we could do to correctly display concave polygons. For one, we could split the concave polygon into two or more convex polygons and process each convex polygon

separately. Another possibility is to modify the Sutherland-Hodgeman approach to check the final vertex list for multiple vertex points along any clip window boundary and correctly join pairs of vertices.

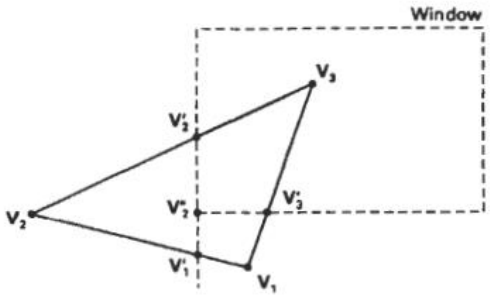


Figure 6-22
A polygon overlapping a rectangular clip window.

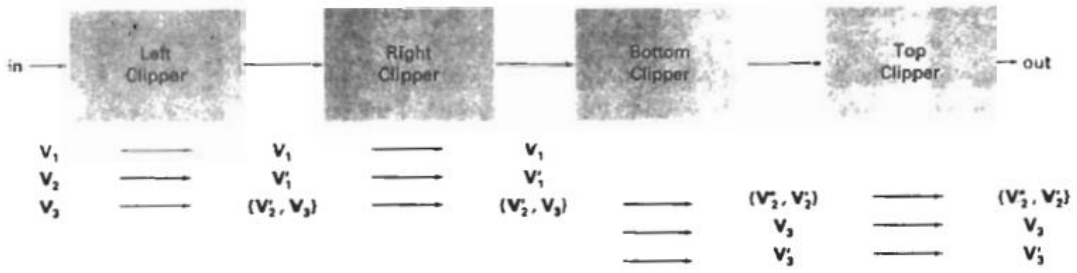


Figure 6-23
Processing the vertices of the polygon in Fig. 6-22 through a boundary-clipping pipeline. After all vertices are processed through the pipeline, the vertex list for the clipped polygon is $\{V_2'', V_2', V_3, V_3'\}$.