Chapter 8
Two-Dimensional Viewing

Myung-Soo Kim
Seoul National University
http://cse.snu.ac.kr/mskim
http://3map.snu.ac.kr

2D Viewing

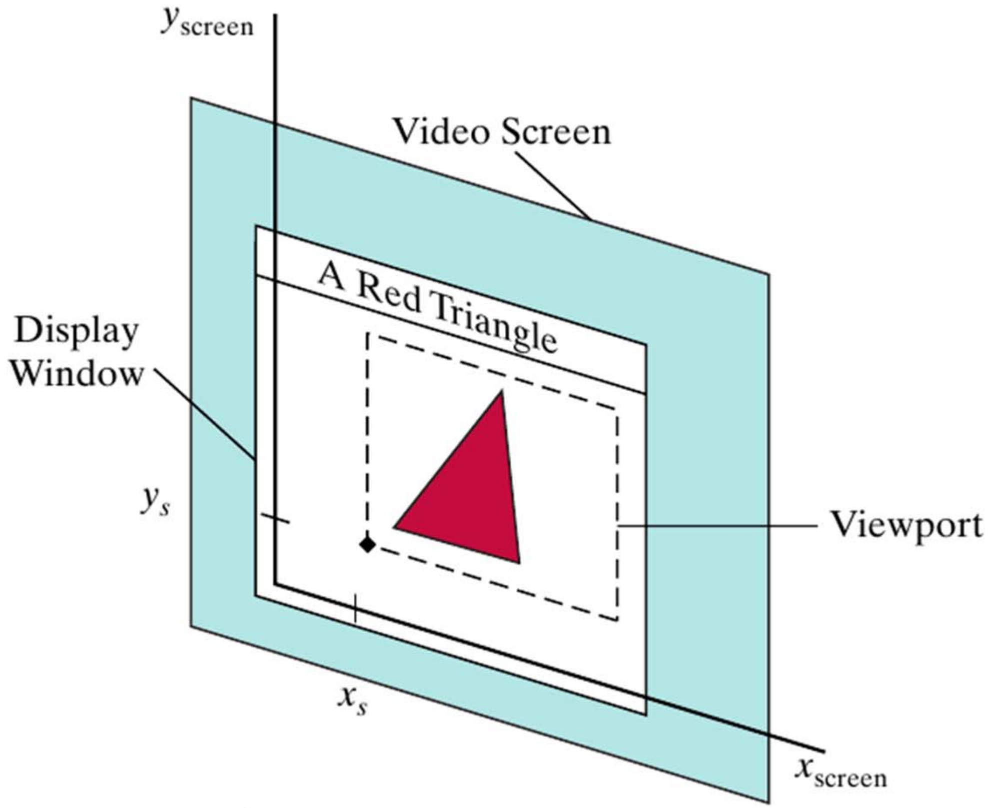


FIGURE 6-9 A viewport at coordinate position (x_s, y_s) within a display window.

2D Viewing Transformation

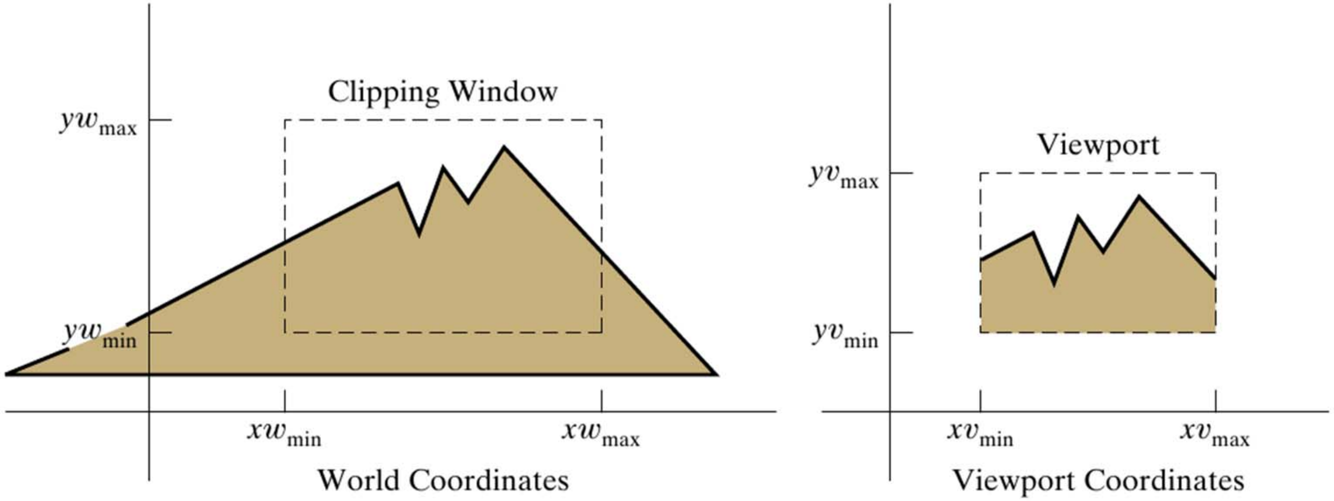


FIGURE 6-2 A clipping window and associated viewport, specified as rectangles aligned with the coordinate axes.

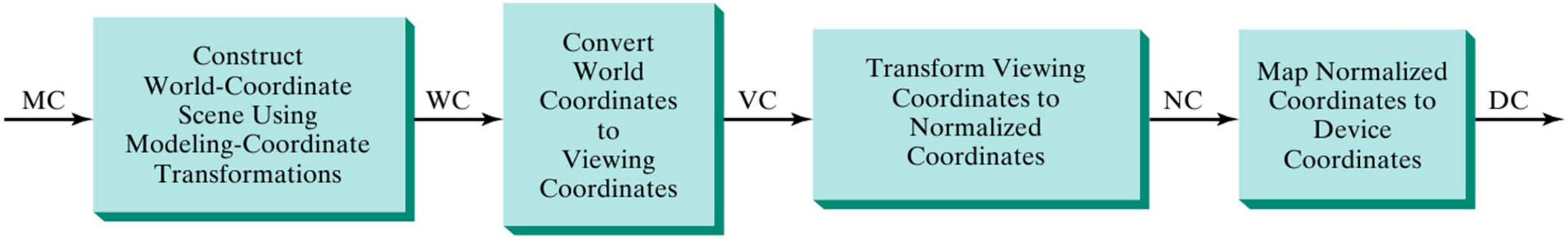


FIGURE 6-3 Two-dimensional viewing-transformation pipeline.

Clipping Window

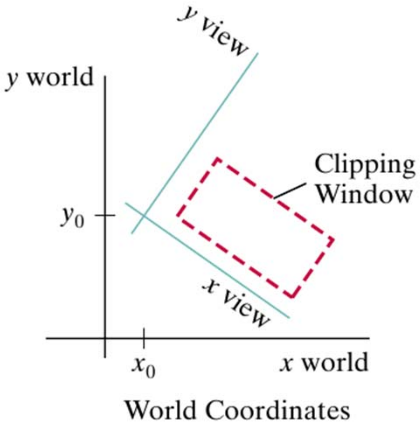


FIGURE 6-4 A rotated clipping window defined in viewing coordinates.

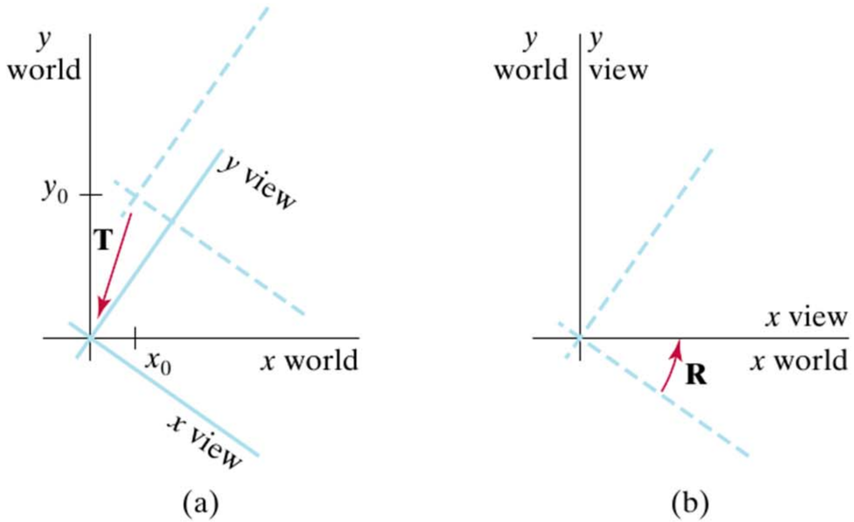


FIGURE 6-5 A viewing-coordinate frame is moved into coincidence with the world frame by (a) applying a translation matrix **T** to move the viewing origin to the world origin, then (b) applying a rotation matrix **R** to align the axes of the two systems.

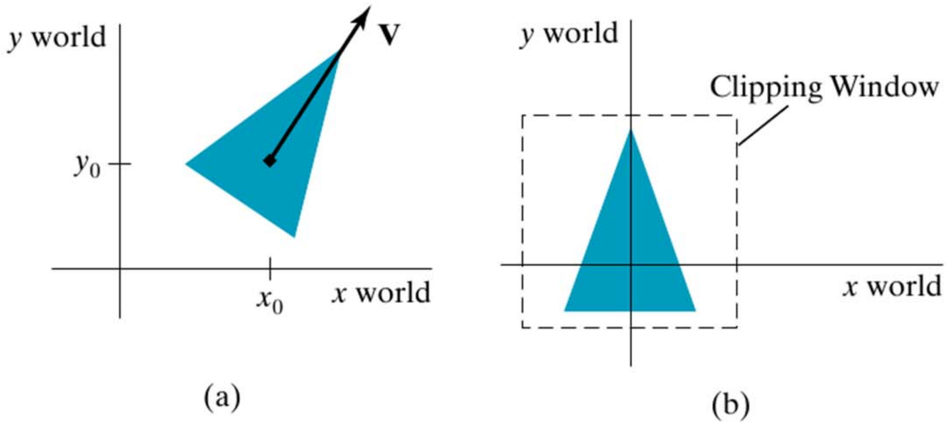


FIGURE 6-6 A triangle (a), with a selected reference point and orientation vector, is translated and rotated to position (b) within a clipping window.

Normalized Viewport/Square

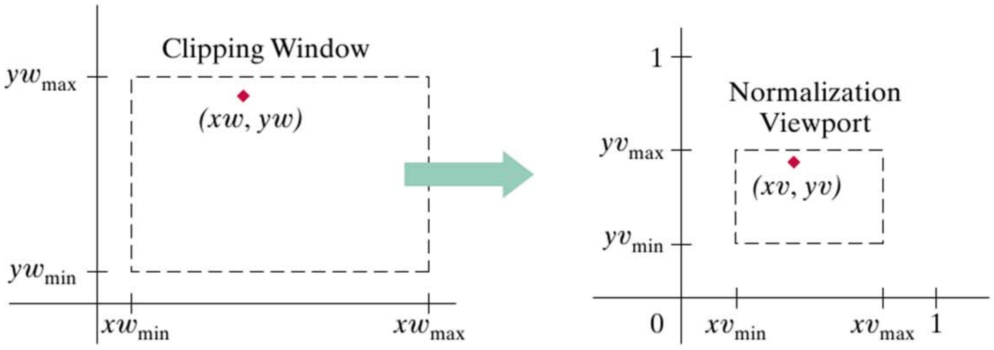


FIGURE 6–7 A point (xw, yw) in a world-coordinate clipping window is mapped to viewport coordinates (xv, yv), within a unit square, so that the relative positions of the two points in their respective rectangles are the same.

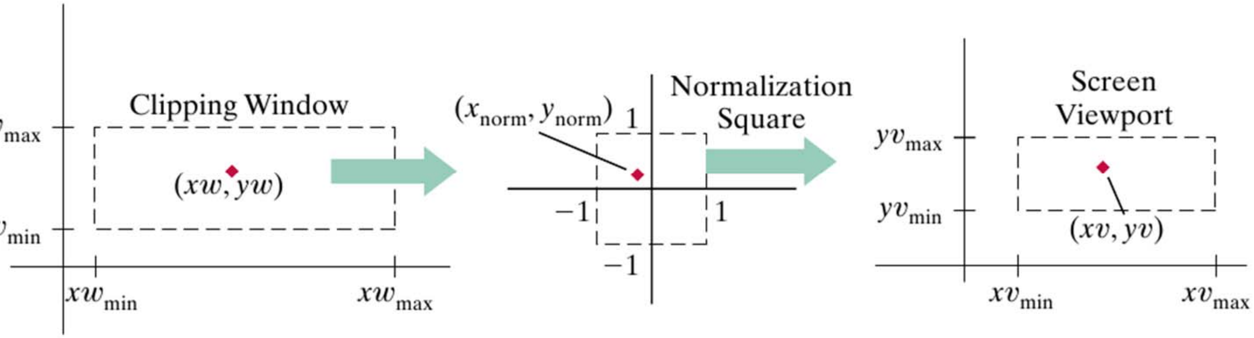


FIGURE 6–8 A point (xw, yw) in a clipping window is mapped to a normalized coordinate position (x_{norm}, y_{norm}), then to a screen-coordinate position (xv, yv) in a viewport. Objects are clipped against the normalization square before the transformation to viewport coordinates.

Line Clipping

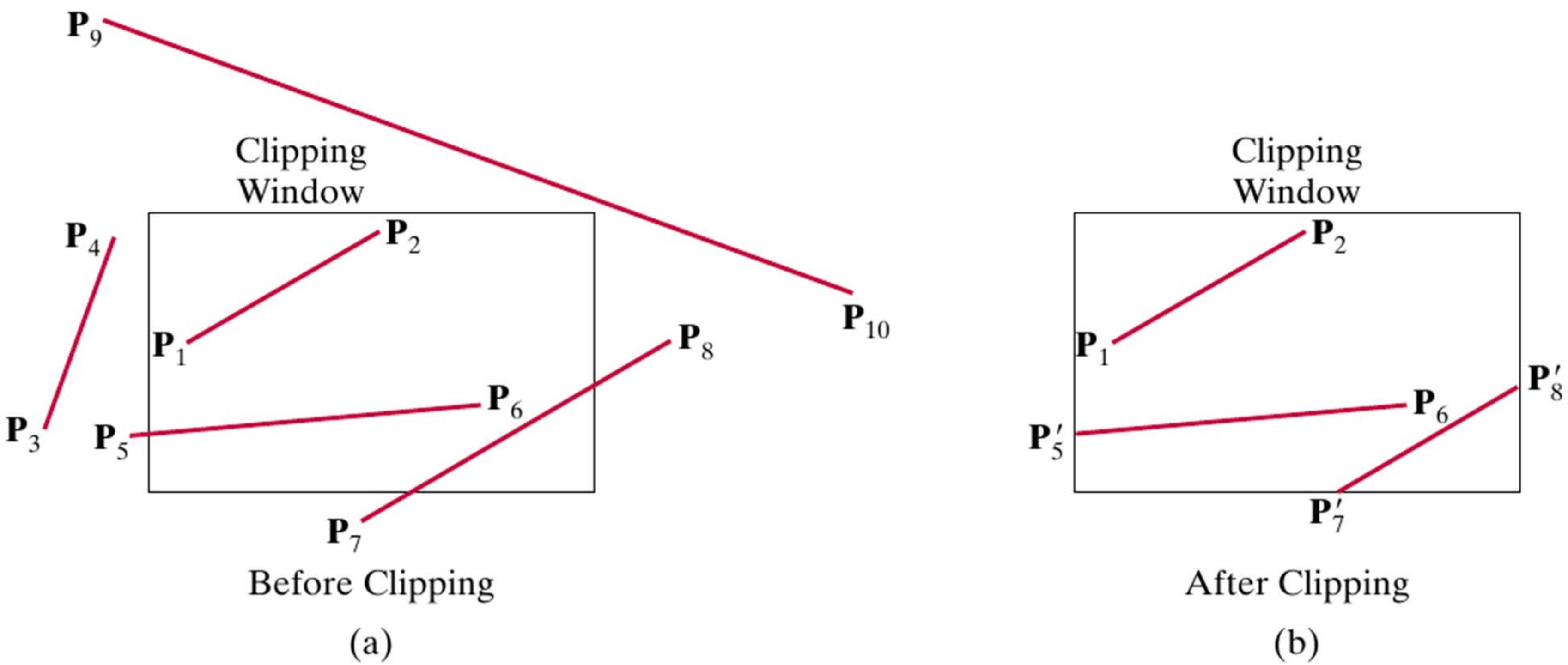
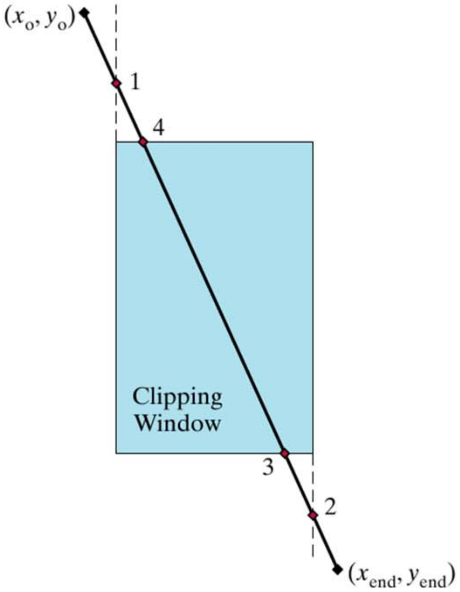
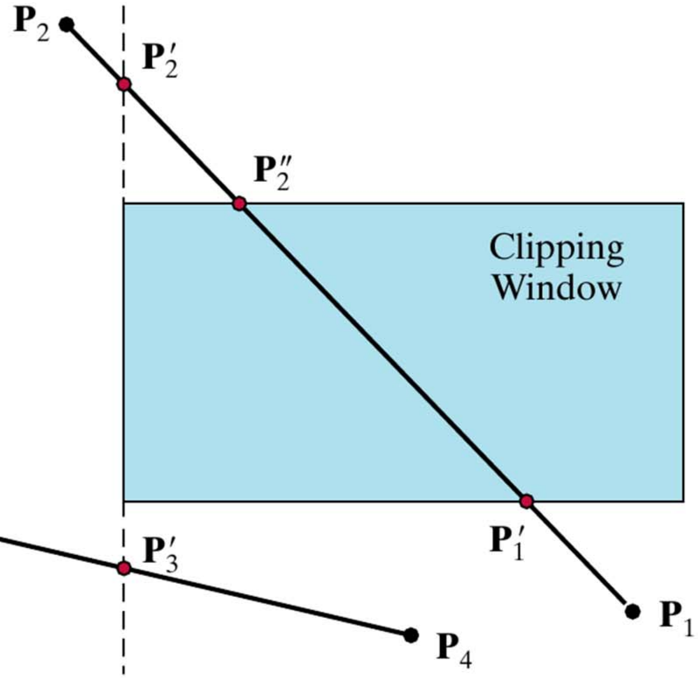


FIGURE 6-11 Clipping straight-line segments using a standard rectangular clipping window.

Cohen-Sutherland Algorithm





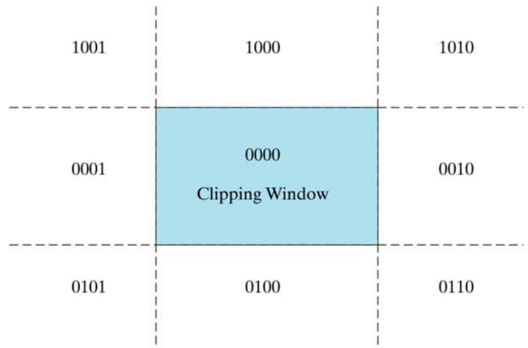


FIGURE 6-13 The nine binary region codes for identifying the position of a line endpoint, relative to the clipping-window boundaries.

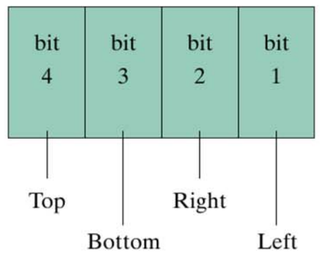


FIGURE 6-12 A possible ordering for the clipping-window boundaries corresponding to the bit positions in the Cohen-Sutherland endpoint region code.

Liang–Barsky Algorithm

endpoints

$(x_0, y_0), (x_{\text{end}}, y_{\text{end}})$

$\Delta x = x_{\text{end}} - x_0$

$\Delta y = y_{\text{end}} - y_0$

$$x = x_0 + u\Delta x$$

$$y = y_0 + u\Delta y$$

$$0 \le u \le 1$$

$$xw_{\min} \le x_0 + u\Delta x \le xw_{\max}$$

$$yw_{\min} \le y_0 + u\Delta y \le yw_{\max}$$

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

$$p_1 = -\Delta x, \qquad q_1 = x_0 - xw_{\min}$$

$$p_2 = \Delta x, \qquad q_2 = xw_{\max} - x_0$$

$$p_3 = -\Delta y, \qquad q_3 = y_0 - yw_{\min}$$

$$p_4 = \Delta y, \qquad q_4 = yw_{\max} - y_0$$

Liang–Barsky Algorithm

$$xw_{\min} \le x_0 + u\Delta x \le xw_{\max} \qquad p_1 = -\Delta x, \qquad q_1 = x_0 - xw_{\min}$$
$$yw_{\min} \le y_0 + u\Delta y \le yw_{\max} \qquad p_2 = \Delta x, \qquad q_2 = xw_{\max} - x_0$$
$$p_3 = -\Delta y, \qquad q_3 = y_0 - yw_{\min}$$
$$u\, p_k \le q_k, \qquad k = 1, 2, 3, 4 \qquad p_4 = \Delta y, \qquad q_4 = yw_{\max} - y_0$$

Any line that is parallel to one of the clipping-window edges has $p_k = 0$ for the value of k corresponding to that boundary, where $k = 1, 2, 3$, and 4 correspond to the left, right, bottom, and top boundaries, respectively. If, for that value of k, we also find $q_k < 0$, then the line is completely outside the boundary and can be eliminated from further consideration. If $q_k \ge 0$, the line is inside the parallel clipping border.

When $p_k < 0$, the infinite extension of the line proceeds from the outside to the inside of the infinite extension of this particular clipping-window edge. If $p_k > 0$, the line proceeds from the inside to the outside. For a nonzero value of p_k, we can calculate the value of u that corresponds to the point where the infinitely extended line intersects the extension of window edge k as

$$u = \frac{q_k}{p_k} \qquad \text{(6-20)}$$

Liang-Barsky Algorithm

$$xw_{\min} \le x_0 + u\Delta x \le xw_{\max}$$
$$yw_{\min} \le y_0 + u\Delta y \le yw_{\max}$$

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

$$p_1 = -\Delta x, \qquad q_1 = x_0 - xw_{\min}$$
$$p_2 = \Delta x, \qquad q_2 = xw_{\max} - x_0$$
$$p_3 = -\Delta y, \qquad q_3 = y_0 - yw_{\min}$$
$$p_4 = \Delta y, \qquad q_4 = yw_{\max} - y_0$$

$$u = \frac{q_k}{p_k} \tag{6-20}$$

For each line, we can calculate values for parameters u_1 and u_2 that define that part of the line that lies within the clip rectangle. The value of u_1 is determined by looking at the rectangle edges for which the line proceeds from the outside to the inside ($p < 0$). For these edges, we calculate $r_k = q_k/p_k$. The value of u_1 is taken as the largest of the set consisting of 0 and the various values of r. Conversely, the value of u_2 is determined by examining the boundaries for which the line proceeds from inside to outside ($p > 0$). A value of r_k is calculated for each of these boundaries, and the value of u_2 is the minimum of the set consisting of 1 and the calculated r values. If $u_1 > u_2$, the line is completely outside the clip window and it can be rejected. Otherwise, the endpoints of the clipped line are calculated from the two values of parameter u.

Nicholl-Lee-Nicholl Algorithm

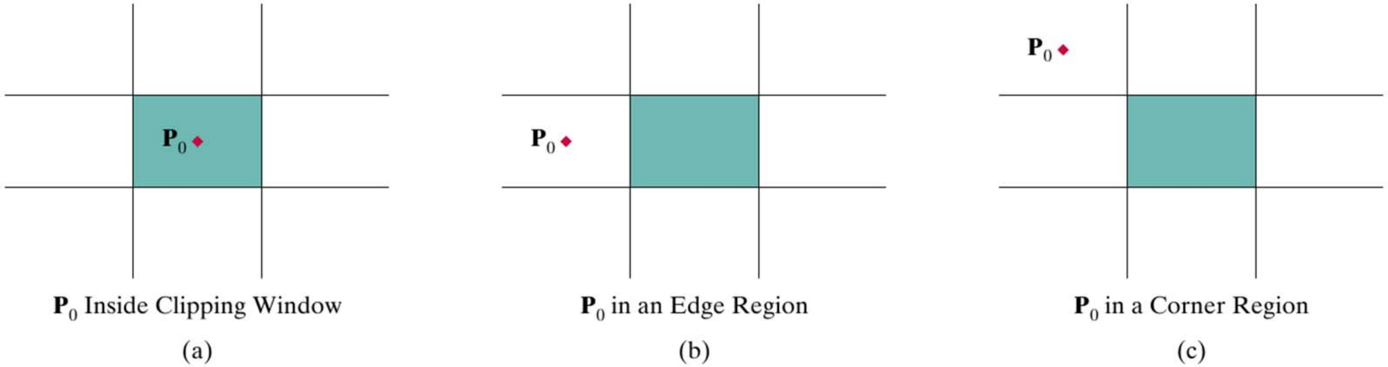


FIGURE 6-16 Three possible positions for a line endpoint $\mathbf{P}_0$ in the NLN line-clipping algorithm.

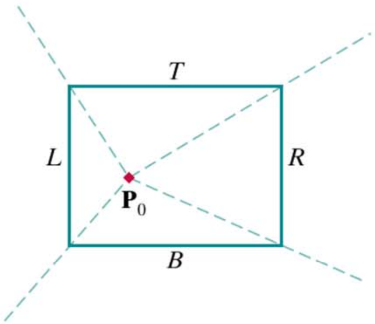


FIGURE 6-17 The four regions used in the NLN algorithm when $\mathbf{P}_0$ is inside the clipping window and $\mathbf{P}_{end}$ is outside.

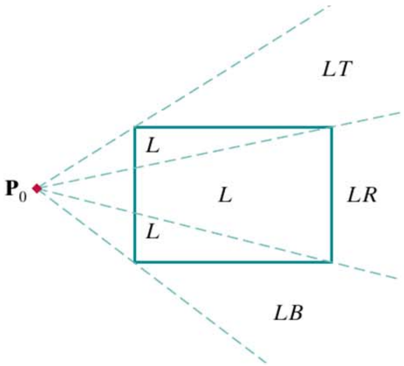


FIGURE 6-18 The four clipping regions used in the NLN algorithm when $\mathbf{P}_0$ is directly to the left of the clip window.

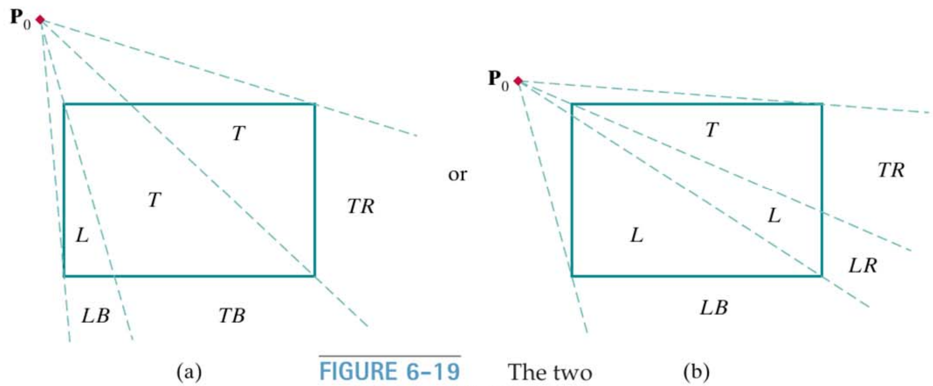


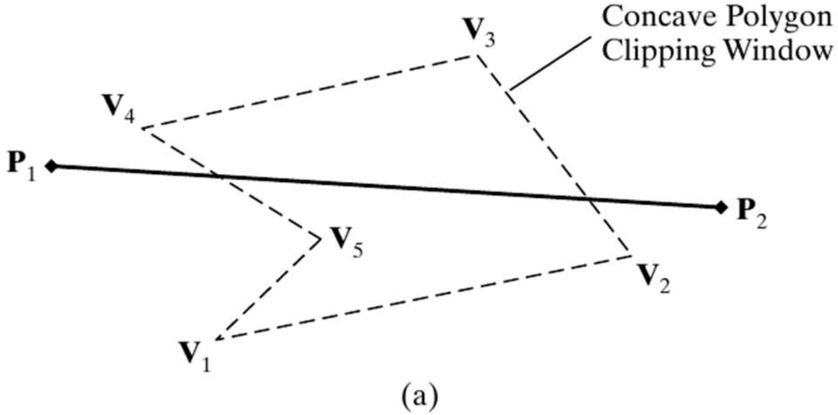
FIGURE 6-19 The two possible sets of clipping regions used in the NLN algorithm when $\mathbf{P}_0$ is above and to the left of the clipping window.

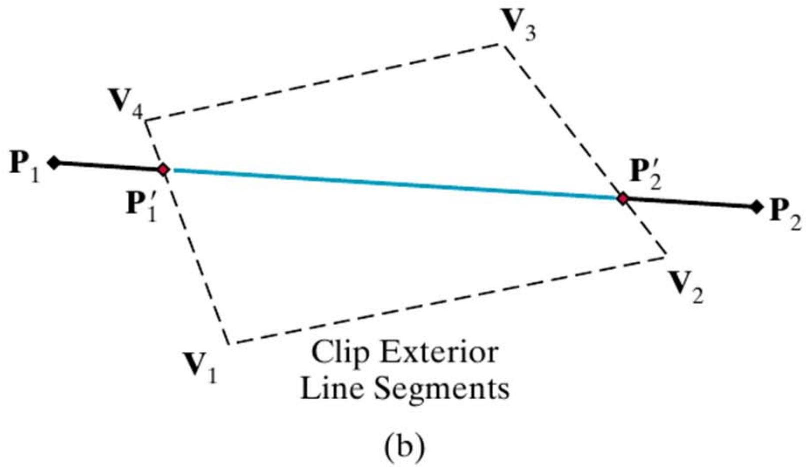
Comparisons

In general, the Liang-Barsky algorithm is more efficient than the Cohen-Sutherland line-clipping algorithm. Each update of parameters u_1 and u_2 requires only one division; and window intersections of the line are computed only once, when the final values of u_1 and u_2 have been computed. In contrast, the Cohen and Sutherland algorithm can repeatedly calculate intersections along a line path, even though the line may be completely outside the clip window. And, each Cohen-Sutherland intersection calculation requires both a division and a multiplication. The two-dimensional Liang-Barsky algorithm can be extended to clip three-dimensional lines (Chapter 7). The extension of the Cohen-Sutherland line-clipping algorithm to three dimensions is straightforward.

Compared to both the Cohen-Sutherland and the Liang-Barsky algorithms, the Nicholl-Lee-Nicholl algorithm performs fewer comparisons and divisions. The trade-off is that the NLN algorithm can be applied only to two-dimensional clipping, whereas both the Liang-Barsky and the Cohen-Sutherland methods are easily extended to three-dimensional scenes.

Nonrectangular Clip Windows





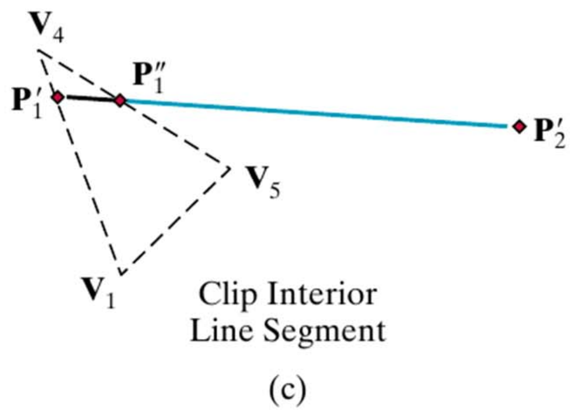


FIGURE 6-20 A concave-polygon clipping window (a), with vertex list ($\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$, $\mathbf{V}_4$, $\mathbf{V}_5$), is modified to the convex polygon ($\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$, $\mathbf{V}_4$) in (b). The external segments of line $\overline{\mathbf{P}_1\mathbf{P}_2}$ are then snipped off using this convex clipping window. The resulting line segment, $\overline{\mathbf{P}'_1\mathbf{P}'_2}$, is next processed against the triangle ($\mathbf{V}_1$, $\mathbf{V}_5$, $\mathbf{V}_4$) (c) to clip off the internal line segment $\overline{\mathbf{P}'_1\mathbf{P}''_1}$ to produce the final clipped line $\overline{\mathbf{P}''_1\mathbf{P}'_2}$.

Polygon Fill-Area Clipping

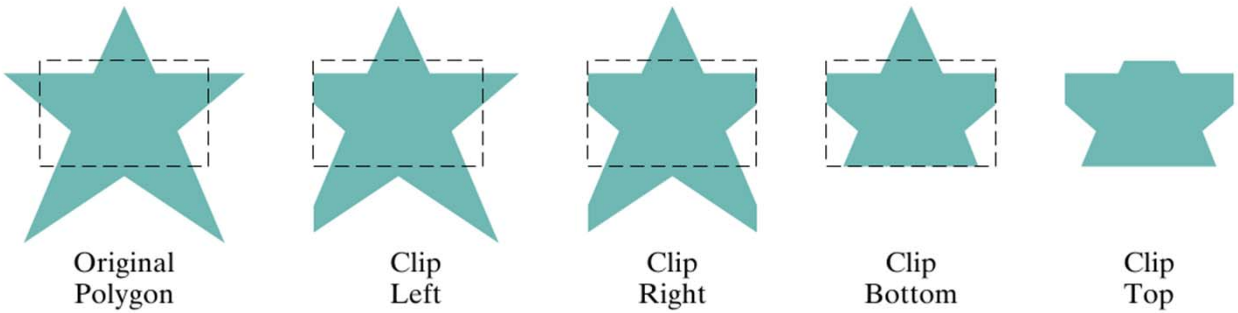


FIGURE 6-23 Processing a polygon fill area against successive clipping-window boundaries.

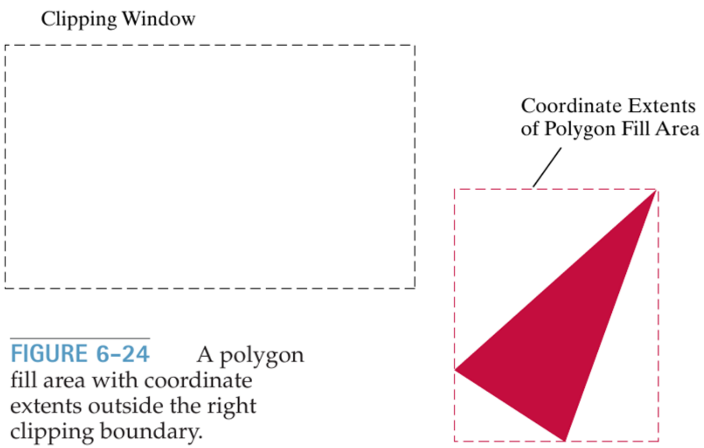


FIGURE 6-24 A polygon fill area with coordinate extents outside the right clipping boundary.

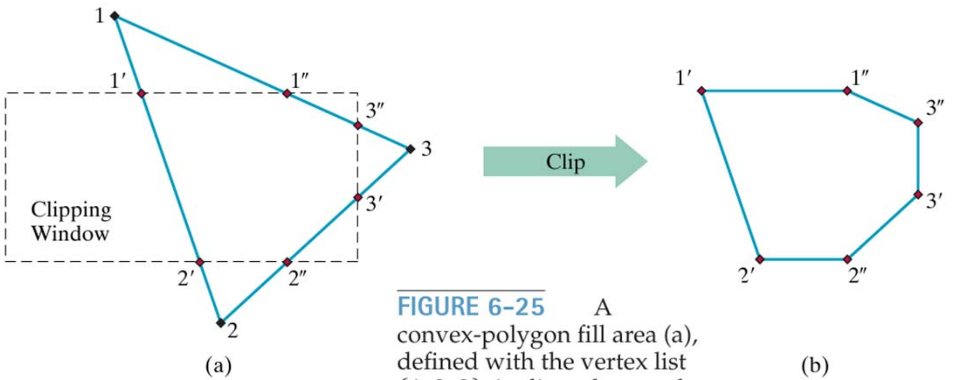


FIGURE 6-25 A convex-polygon fill area (a), defined with the vertex list {1, 2, 3}, is clipped to produce the fill-area shape shown in (b), which is defined with the output vertex list {1′, 2′, 2″, 3′, 3″, 1″}.

Sutherland-Hodgman Algorithm

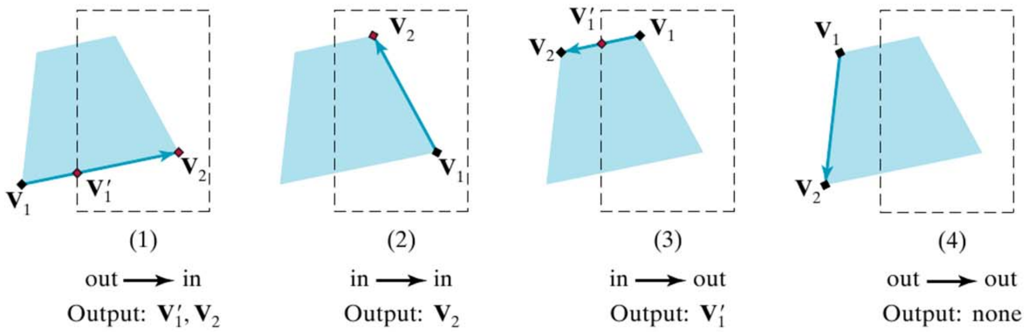


FIGURE 6-26 The four possible outputs generated by the left clipper, depending on the position of a pair of endpoints relative to the left boundary of the clipping window.

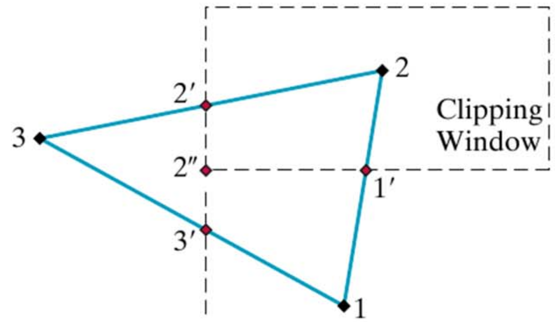
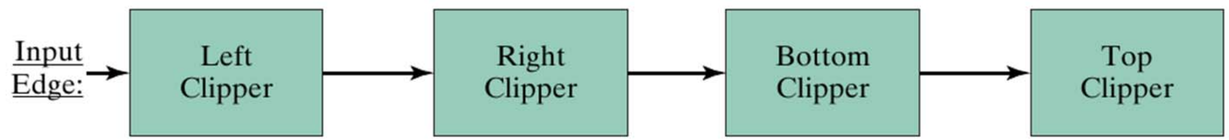


FIGURE 6-27 Processing a set of polygon vertices, {1, 2, 3}, through the boundary clippers using the Sutherland-Hodgman algorithm. The final set of clipped vertices is {1′, 2, 2′, 2″}.



Left Clipper	Right Clipper	Bottom Clipper	Top Clipper
{1, 2}: (in – in) → {2}			
{2, 3}: (in – out) → {2′}	{2, 2′}: (in – in) → {2′}		
{3, 1}: (out – in) → {3′, 1}	{2′, 3′}: (in – in) → {3′}	{2′, 3′}: (in – out) → {2″}	
	{3′, 1}: (in – in) → {1}	{3′, 1}: (out – out) → { }	
	{1, 2}: (in – in) → {2}	{1, 2}: (out – in) → {1′, 2}	{2″, 1′}: (in – in) → {1′}
		{2, 2′}: (in – in) → {2′}	{1′, 2}: (in – in) → {2}
			{2, 2′}: (in – in) → {2′}
			{2′, 2″}: (in – in) → {2″}

Weiler-Atherton Algorithm

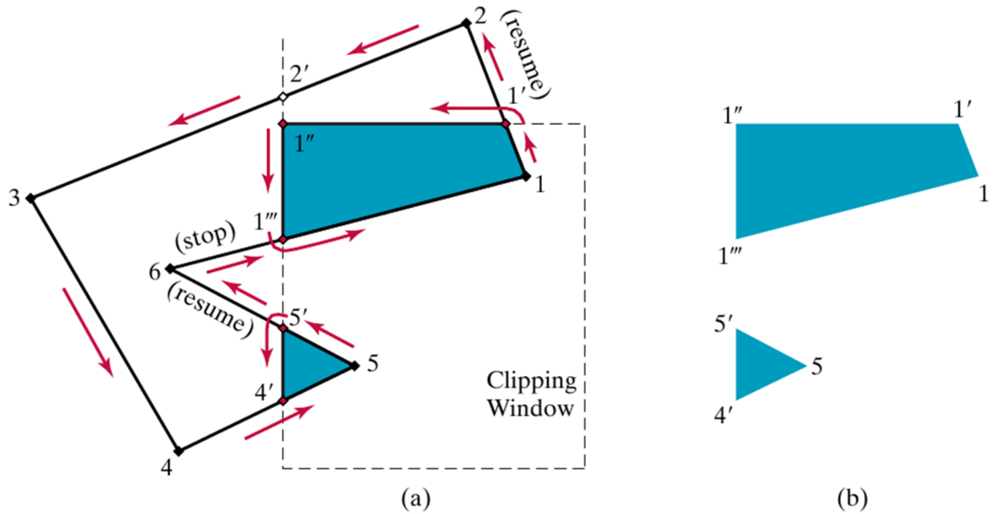


FIGURE 6-29 A concave polygon (a), defined with the vertex list {1, 2, 3, 4, 5, 6}, is clipped using the Weiler-Atherton algorithm to generate the two lists {1, 1′, 1″, 1‴} and {4′, 5, 5′}, which represent the separate polygon fill areas shown in (b).

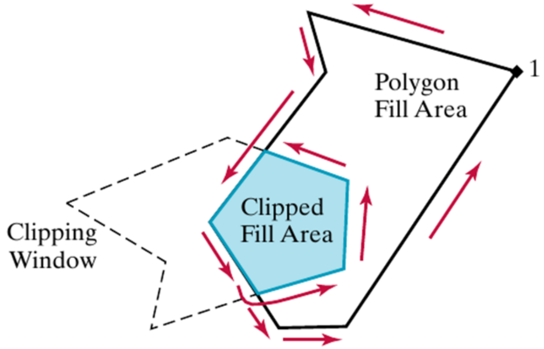


FIGURE 6-30 Clipping a polygon fill area against a concave-polygon clipping window using the Weiler-Atherton algorithm.