Two-Dimensional Viewing

The 2D Viewing Pipeline

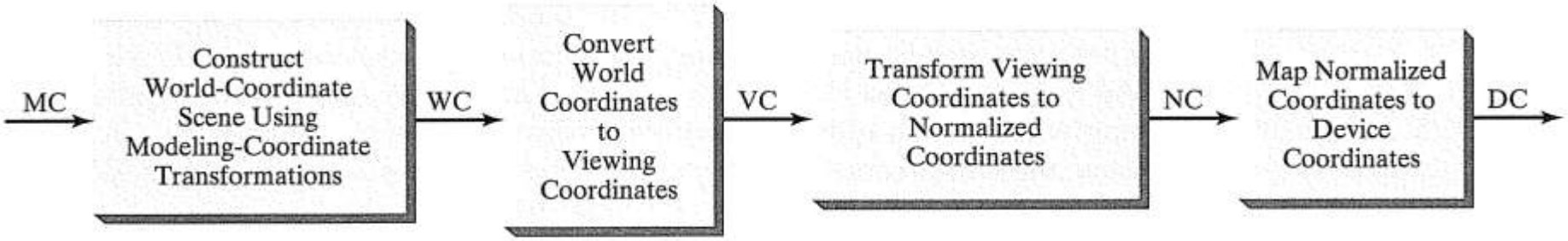
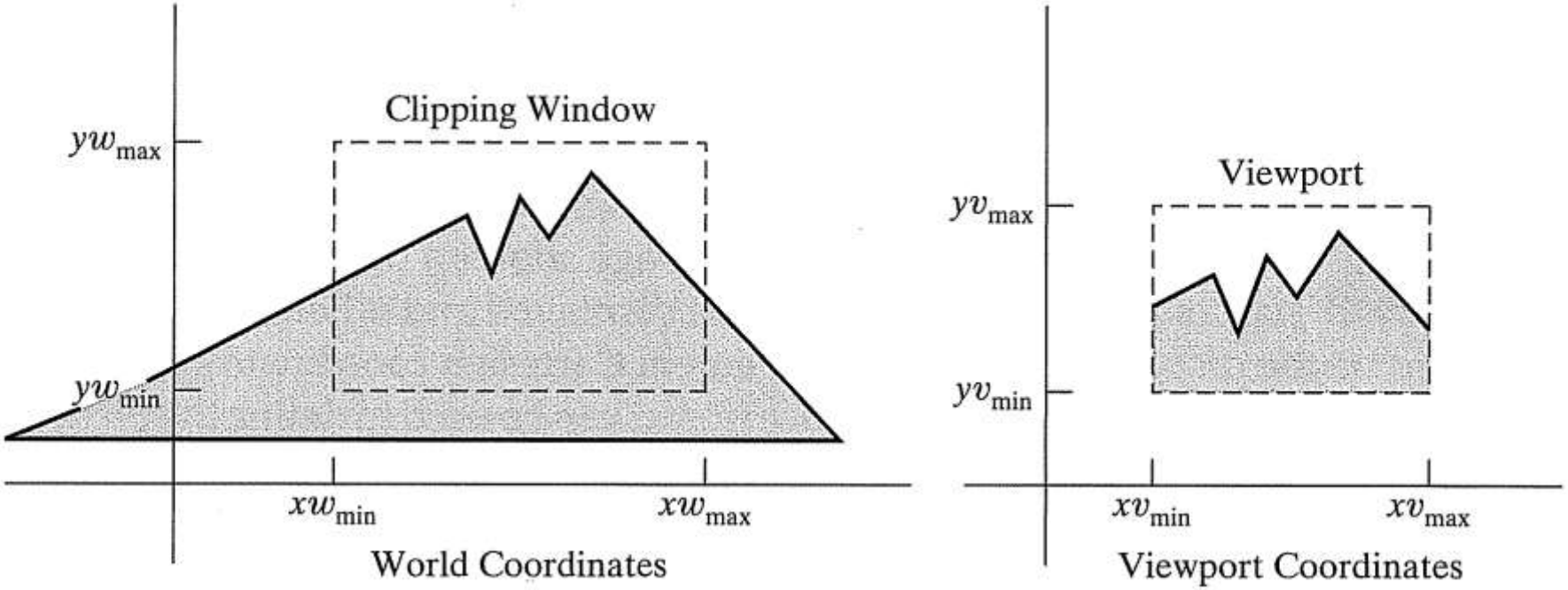


FIGURE 6-3 Two-dimensional viewing-transformation pipeline.

World coordinates clipping window



Window to viewport transformation: is the mapping from world coordinates to device coordinates.

Normalization and view port transformation

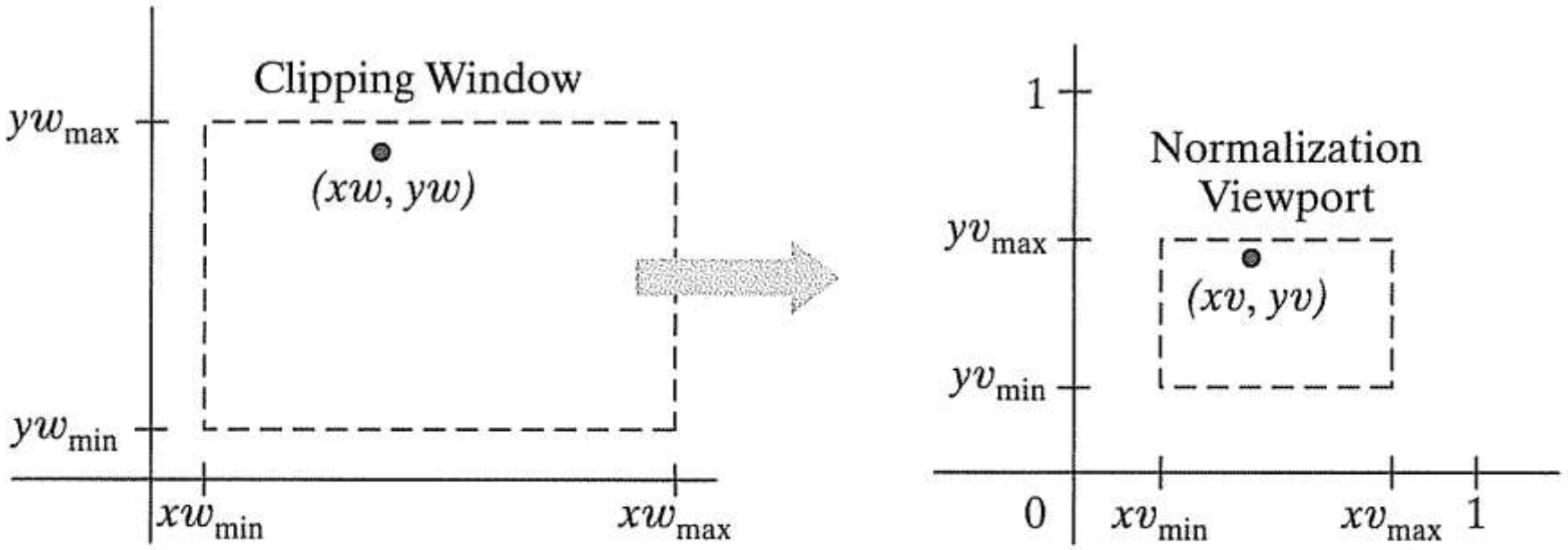


FIGURE 6-7 A point (xw, yw) in a world-coordinate clipping window is mapped to viewport coordinates (xv, yv), within a unit square, so that the relative positions of the two points in their respective rectangles are the same.

To transform the world-coordinate point into the same relative position within the viewport, we require that

$$\frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}}$$
$$\frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}} \tag{6-2}$$

Solving these expressions for the viewport position (xv, yv), we have

$$xv = s_x xw + t_x$$
$$yv = s_y yw + t_y \tag{6-3}$$

where the scaling factors are

$$s_x = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$$
$$s_y = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}} \tag{6-4}$$

and the translation factors are

$$t_x = \frac{xw_{\max} xv_{\min} - xw_{\min} xv_{\max}}{xw_{\max} - xw_{\min}}$$
$$t_y = \frac{yw_{\max} yv_{\min} - yw_{\min} yv_{\max}}{yw_{\max} - yw_{\min}} \tag{6-5}$$

Transformation form world coordinates to view port coordinates

(1) Scale the clipping window to the size of the viewport using a fixed-point position of $(xw_{\min}, yw_{\min})$.

(2) Translate $(xw_{\min}, yw_{\min})$ to $(xv_{\min}, yv_{\min})$.

The scaling transformation in step (1) can be represented with the two-dimensional matrix

$$\mathbf{S} = \begin{bmatrix} s_x & 0 & xw_{\min}(1 - s_x) \\ 0 & s_y & yw_{\min}(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix} \qquad (6\text{-}6)$$

where s_x and s_y are the same as in Eqs. 6-4. The two-dimensional matrix representation for the translation of the lower-left corner of the clipping window to the lower-left viewport corner is

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & xv_{\min} - xw_{\min} \\ 0 & 1 & yv_{\min} - yw_{\min} \\ 0 & 0 & 1 \end{bmatrix} \tag{6-7}$$

And the composite matrix representation for the transformation to the normalized viewport is

$$\mathbf{M}_{\text{window, normviewp}} = \mathbf{T} \cdot \mathbf{S} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6-8}$$

which gives us the same result as in Eqs. 6-3

Clipping Algorithms

Generally, any procedure that eliminates those portions of a picture that are either inside or outside of a specified region of space is referred to as a **clipping algorithm** or simply **clipping.** Usually a clipping region is a rectangle in standard position, although we could use any shape for a clipping application.

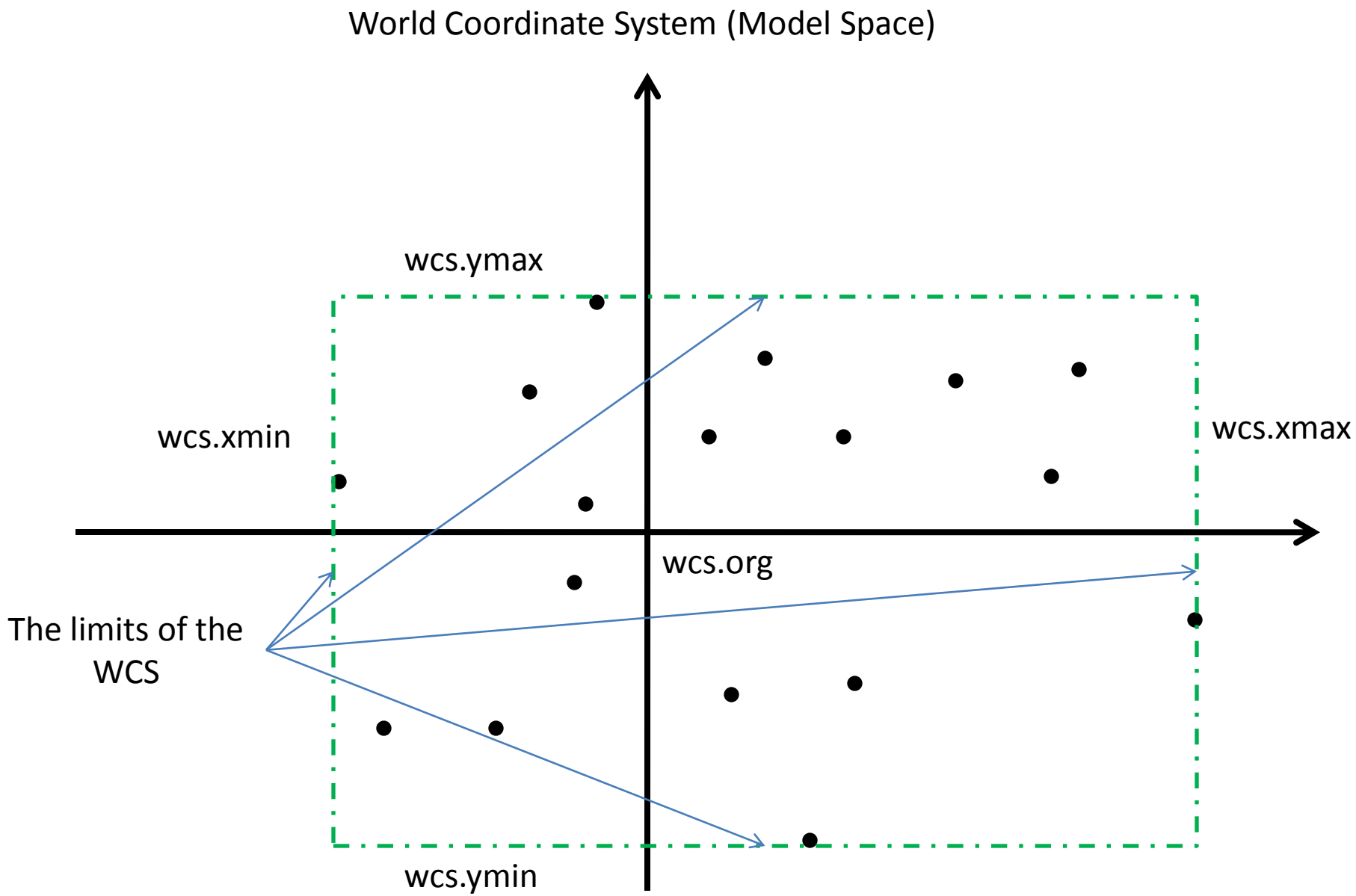
- Point Clipping
- Line Clipping (straight-line segments)
- Fill-Area Clipping (polygons)

Point clipping

6-6 TWO-DIMENSIONAL POINT CLIPPING

For a clipping rectangle in standard position, we save a two-dimensional point $\mathbf{P} = (x, y)$ for display if the following inequalities are satisfied:

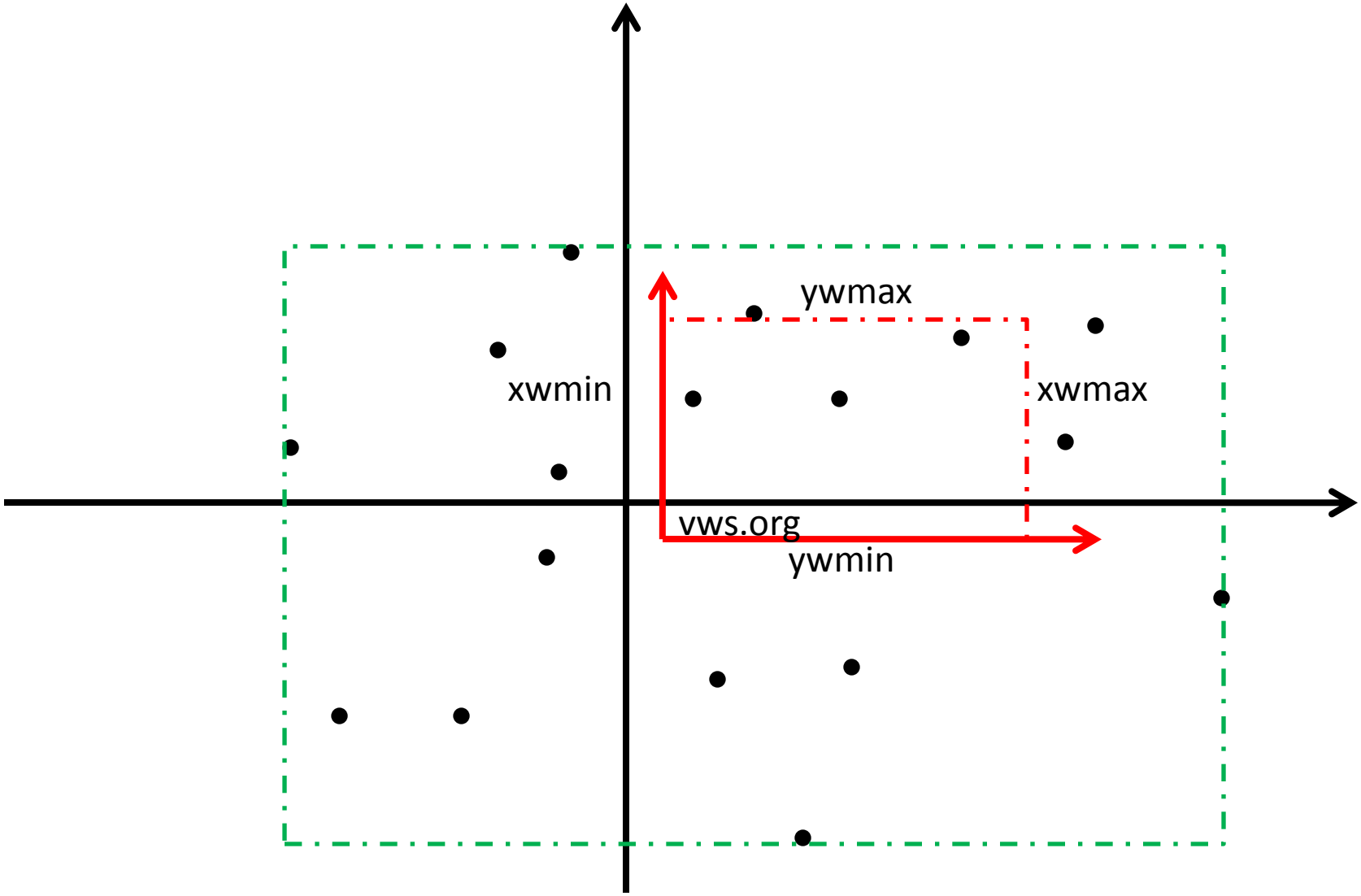
$$\begin{aligned} xw_{\min} &\leq x \leq xw_{\max} \\ yw_{\min} &\leq y \leq yw_{\max} \end{aligned} \qquad (6\text{-}12)$$



The coordinates of the points are given in world co-ordinates

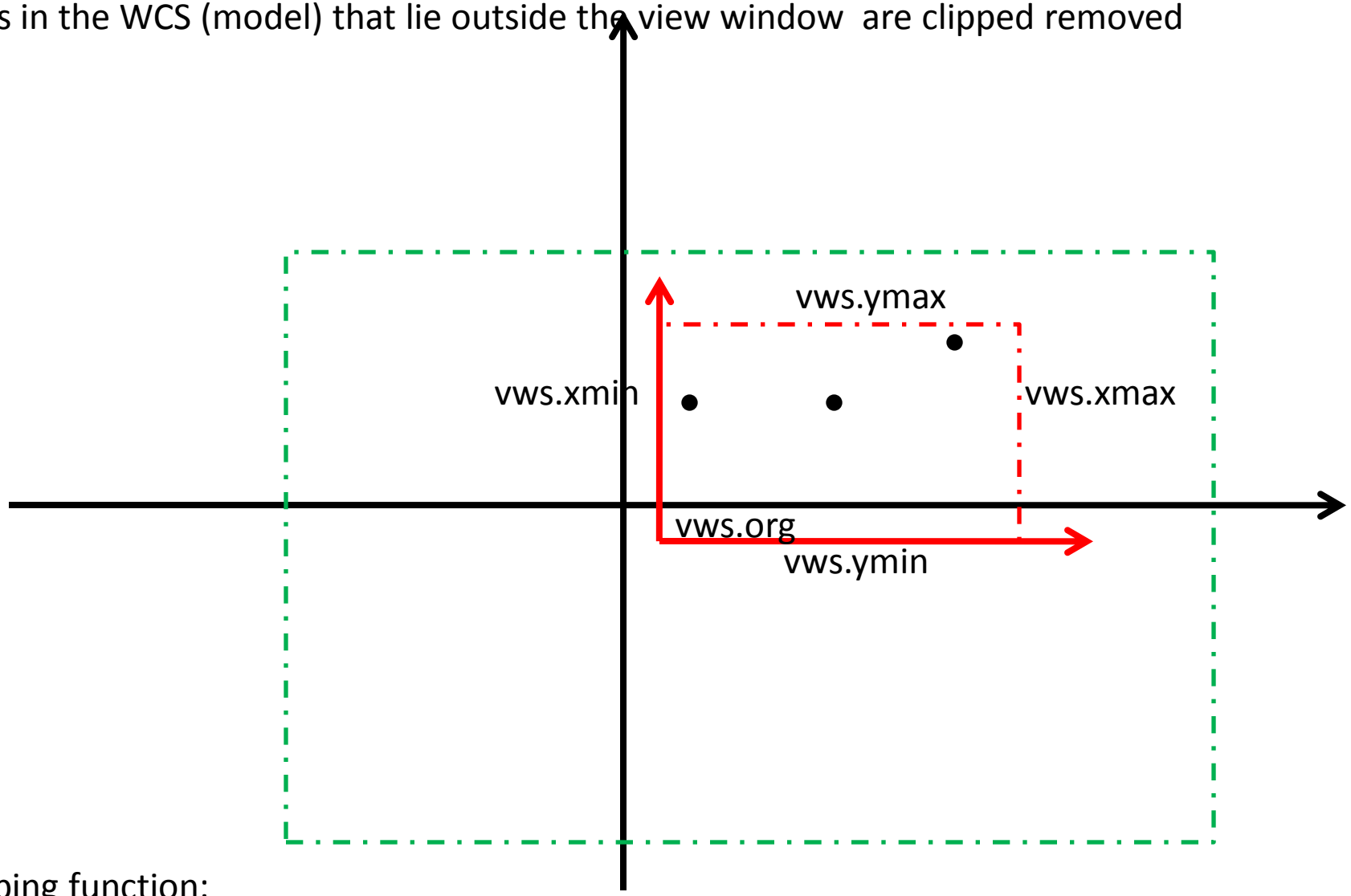
The limits (extents) of the WCS are determined by the points in the model

The View Window System



The limits (extents) of the VWS are determined by the user by panning (sliding the view window) and zooming (enlarging or shrinking the view window)

Points in the WCS (model) that lie outside the view window are clipped removed



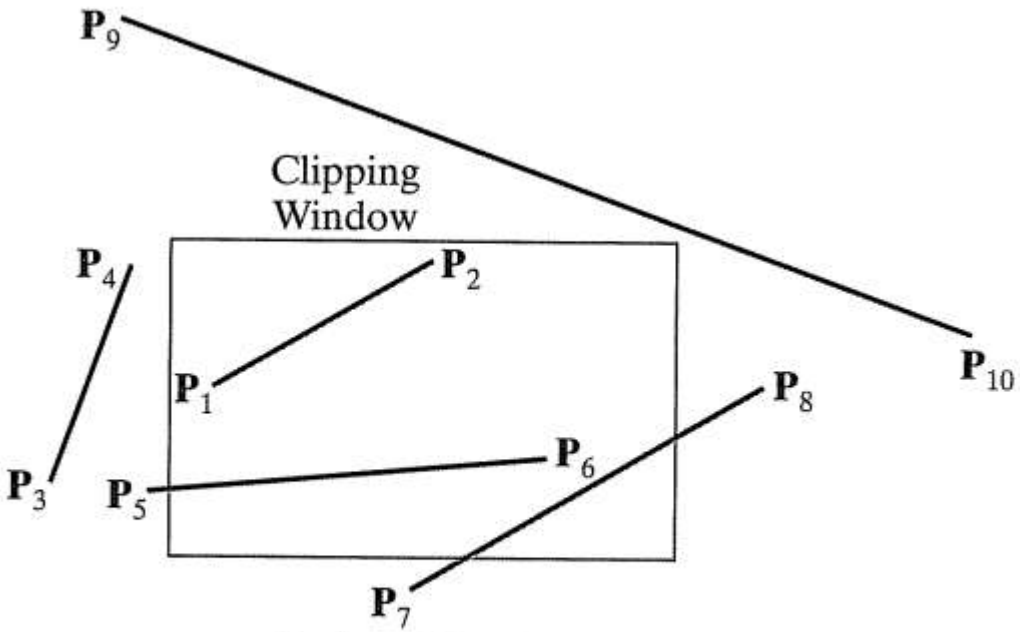
Clipping function:

for each point (p) in the wcs

if ((p.x < vws.xmin) or p.x > (vws.xmax) or (p.y < vws.ymin) or (p.y > vws.ymax))

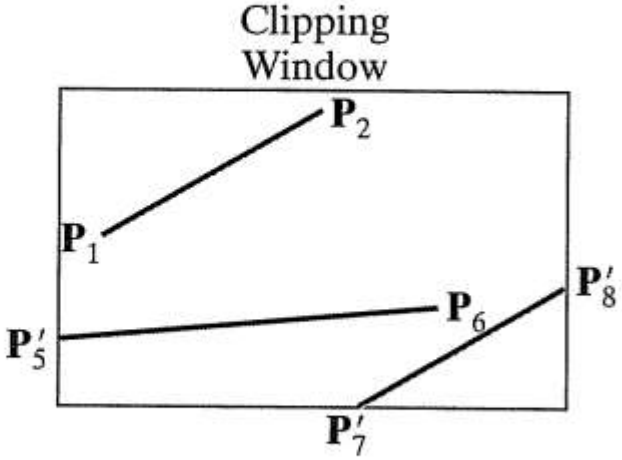
Clip point p

Line clipping



Before Clipping

(a)



After Clipping

(b)

Line clipping Algorithms

We test a line segment to determine if it is completely inside or outside a selected clipping-window edge by applying the point-clipping tests of the previous section. When both endpoints of a line segment are inside all four clipping boundaries, such as the line from $\mathbf{P}_1$ to $\mathbf{P}_2$ in Fig 6-11, the line is completely inside the clipping window and we save it. And when both endpoints of a line segment are outside any one of the four boundaries (line $\overline{\mathbf{P}_3\mathbf{P}_4}$ in Fig. 6-11), that line is completely outside the window and it is eliminated from the scene description. But if both these tests fail, the line segment intersects at least one clipping boundary and it may or may not cross into the interior of the clipping window.

Cohen-Sutherland Line Clipping

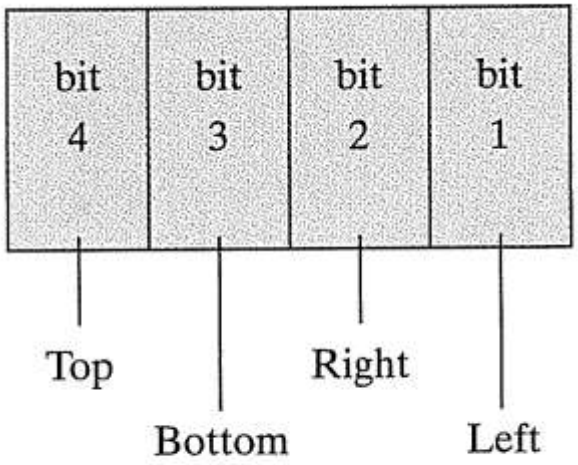
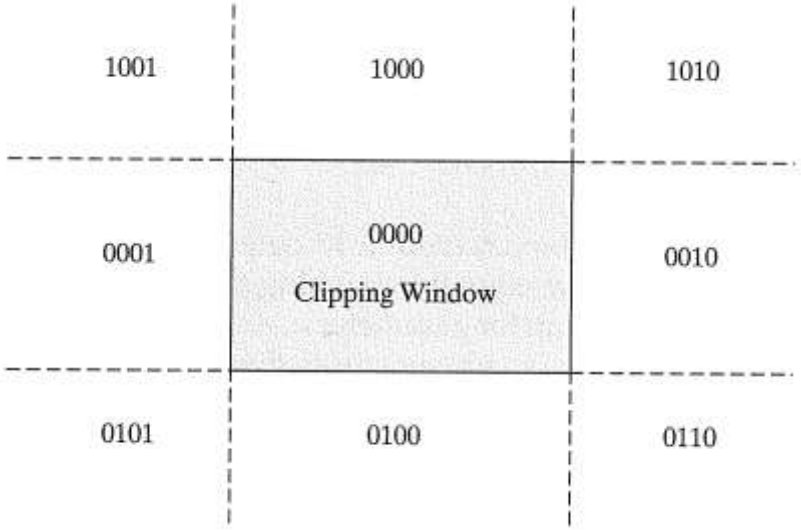


FIGURE 6-12 A possible ordering for the clipping-window boundaries corresponding to the bit positions in the Cohen-Sutherland endpoint region code.

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



Once we have established region codes for all line endpoints, we can quickly determine which lines are completely inside the clip window and which are clearly outside. Any lines that are completely contained within the window edges have a region code of 0000 for both endpoints, and we save these line segments. Any line that has a region-code value of 1 in the same bit position for each endpoint is completely outside the clipping rectangle, and we eliminate that line segment.

we can more efficiently determine the values for a region-code using bit-processing operations and the following two steps: (1) Calculate differences between endpoint coordinates and clipping boundaries. (2) Use the resultant sign bit of each difference calculation to set the corresponding value in the region code. For the ordering scheme shown in Fig. 6-12, bit 1 is the sign bit of $x - xw_{\min}$; bit 2 is the sign bit of $xw_{\max} - x$; bit 3 is the sign bit of $y - yw_{\min}$; and bit 4 is the sign bit of $yw_{\max} - y$.

We can perform the inside-outside tests for line segments using logical operators. When the *or* operation between two endpoint region codes for a line segment is *false* (0000), the line is inside the clipping window. Therefore, we save the line and proceed to test the next line in the scene description. When the *and* operation between the two endpoint region codes for a line is *true* (not 0000), the line is completely outside the clipping window, and we can eliminate it from the scene description.

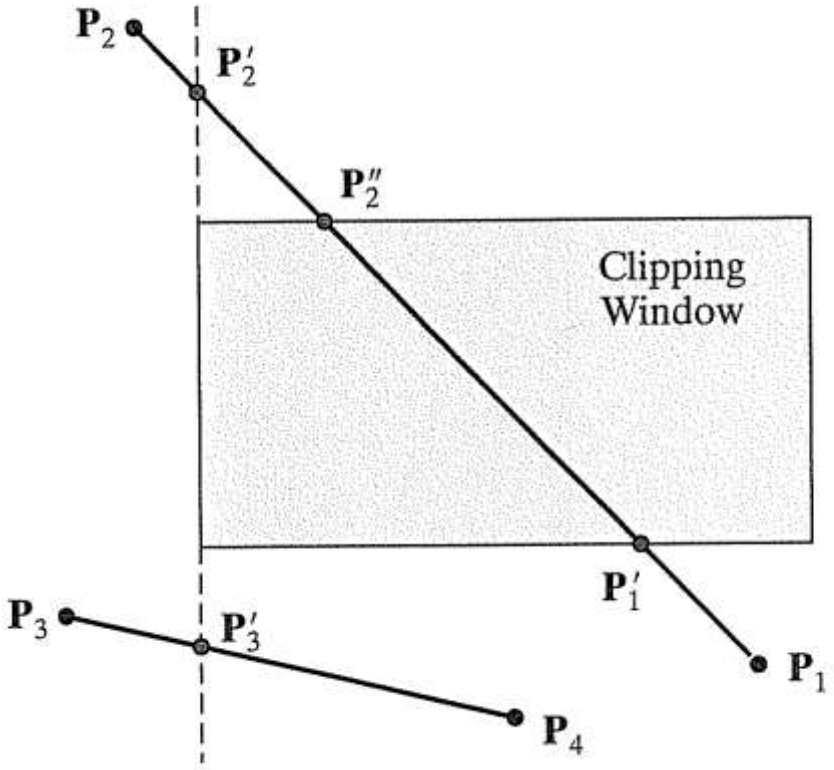


FIGURE 6-14 Lines extending from one clipping-window region to another may cross into the clipping window, or they could intersect one or more clipping boundaries without entering the window interior.

Figure 6-14 illustrates two line segments that cannot be immediately identified as completely inside or completely outside the clipping window. The region codes for the line from $\mathbf{P}_1$ to $\mathbf{P}_2$ are 0100 and 1001. Thus, $\mathbf{P}_1$ is inside the left clipping boundary and $\mathbf{P}_2$ is outside that boundary. We then calculate the intersection position $\mathbf{P}'_2$, and we clip off the line section from $\mathbf{P}_2$ to $\mathbf{P}'_2$. The remaining portion of the line is inside the right border line, and so we next check the bottom border. Endpoint $\mathbf{P}_1$ is below the bottom clipping edge and $\mathbf{P}'_2$ is above it, so we determine the intersection position at this boundary ($\mathbf{P}'_1$). We eliminate the line section from $\mathbf{P}_1$ to $\mathbf{P}'_1$ and proceed to the top window edge. There we determine the intersection position to be $\mathbf{P}''_2$. The final step is to clip off the section above the top boundary and save the interior segment from $\mathbf{P}'_1$ to $\mathbf{P}''_2$. For the second line, we find that point $\mathbf{P}_3$ is outside the left boundary and $\mathbf{P}_4$ is inside. Thus, we calculate the intersection position $\mathbf{P}'_3$ and eliminate the line section from $\mathbf{P}_3$ to $\mathbf{P}'_3$. By checking region codes for the endpoints $\mathbf{P}'_3$ and $\mathbf{P}_4$, we find that the remainder of the line is below the clipping window and can be eliminated also.

END OF PRESENTATION