Window and Viewport

The **window** is a rectangular area in the 2D world coordinate system. It defines what the user can see. Note that the meaning of the term **window** as used here is different from that used in the window-based operating system.

The **viewport** is a rectangular region on computer screen. It is where objects enclosed in the window appear on screen. Objects falling inside the window are mapped into the viewport.

Let the left-lower corner of the window be (wx_l, wy_b), and the right-upper corner be (wx_r, wy_t). Let the left-lower corner of the viewport be (vx_l, vy_b), and the right-upper corner be (vx_r, vy_t). See the figure below. Then the window-to-viewport transformation $(x, y) \to (x', y')$ is given by

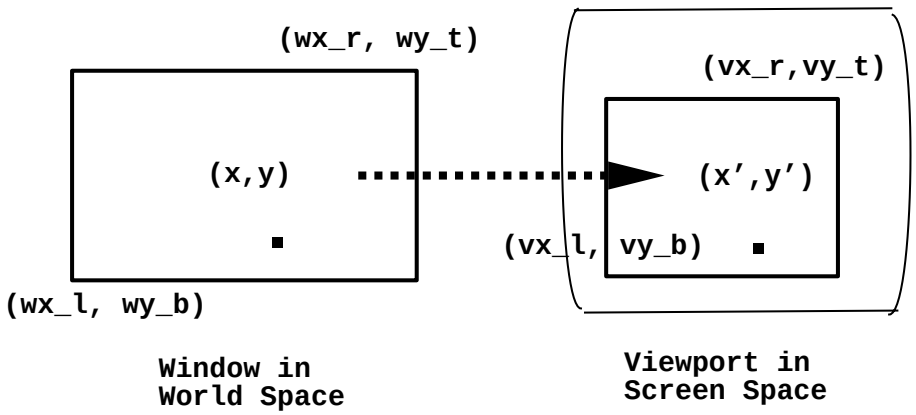
$$\frac{x' - vx_l}{vx_r - vx_l} = \frac{x - wx_l}{wx_r - wx_l}.$$

Or,

$$x' = \frac{x - wx_l}{wx_r - wx_l}(vx_r - vx_l) + vx_l.$$

Similarly

$$y' = \frac{y - wy_l}{wy_r - wy_l}(vy_r - vy_l) + vy_l.$$



Line Clipping Against Window – Cohen-Sutherland Algorithm

The whole plane is divided into 9 regions by extending the four boundaries of the window. See Figure 4. Each region is associated with a 4-bit code, called *outcode*. A bit of the code is set 1 if the corresponding region is outside the window with respect to a designated side of the window.

The two end points X_1 and X_2 of an input line segment are assigned codes C_1 and C_2, respectively, of the regions which they fall in. For instance, a point inside the window is given 0000, and a point right above the window is given 1000. Now Cohen-Sutherland algorithm proceeds as follows.

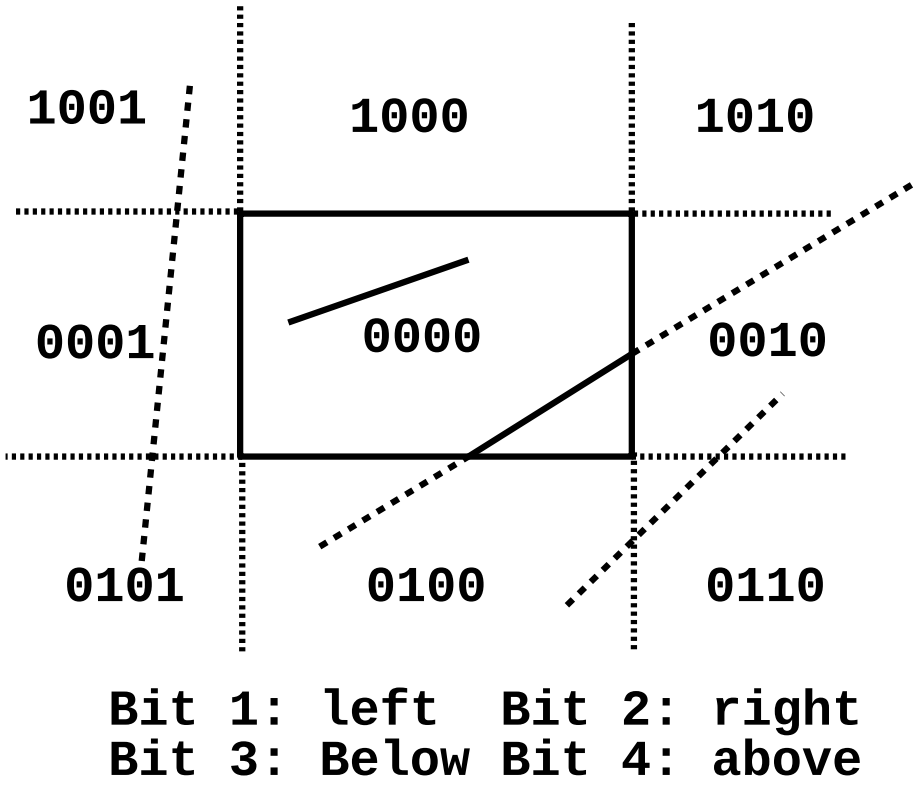


Figure 4.

1) If $C_1||C_2 = 0000$, the line segment is inside the window; **display**.

2) If $C_1\&\&C_2 \neq 0000$, the line segment is outside the window; **discard**.

3) If neither 1) nor 2), the line segment is subdivided into two shorter segments by one (extended) boundary of the window which crosses the line, and for each of the resulting segments, repeat 1) and 2). A boundary crossed by

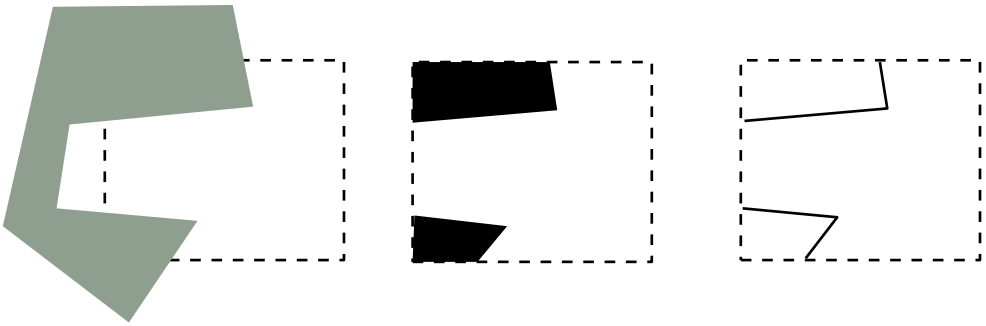
the line is identified by the non-zero bit in $C_1||C_2$.

Polygon clipping

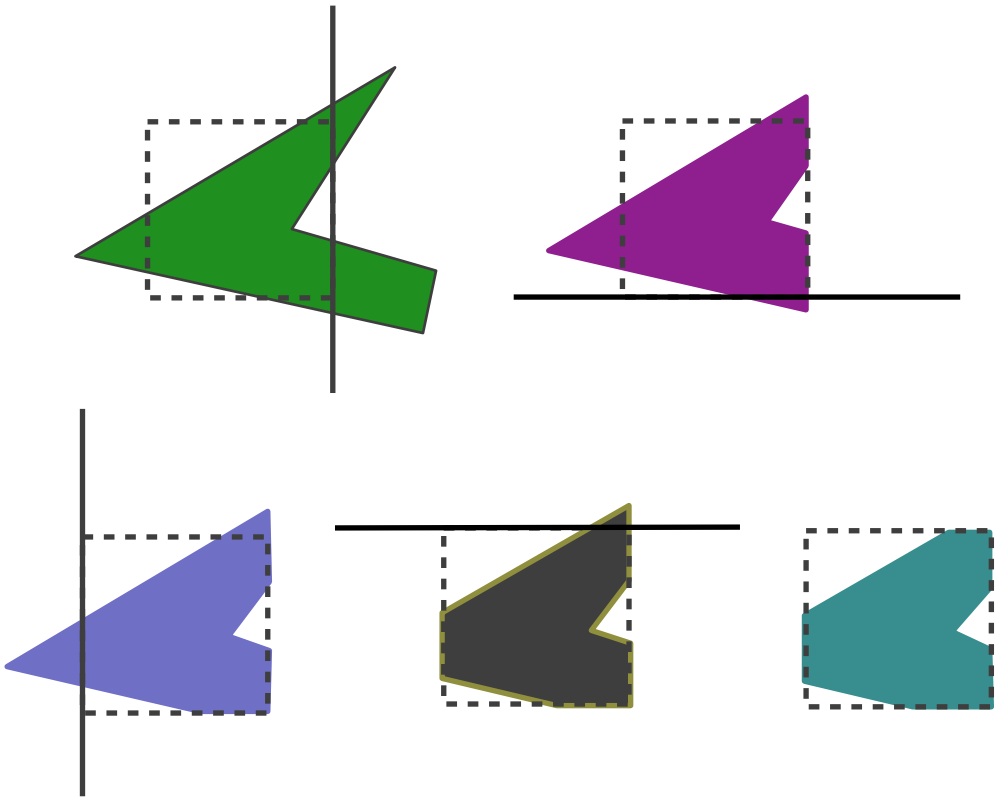
For filled polygons, line clipping algorithms do not produce useful results as the inside/outside property of a polygon is not taken into consideration.

A polygon clipping algorithm may be simply described as follows. The sequence of vertices of a polygon is input and clipped against, in turn, the four boundaries of a rectangular clipping region, and an output sequence of vertices is maintained. The four cases of a vertex relative to a clipping boundary, and how output vertices are determined in each case, are illustrated in the following figures.

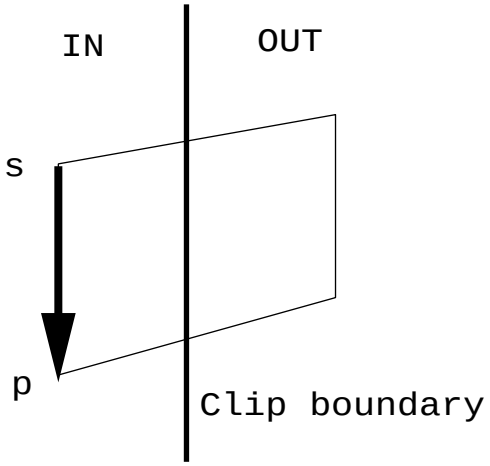
Further processing is needed to connect all output vertices to form meaningful polygons, by considering edges on the clipping boundaries.



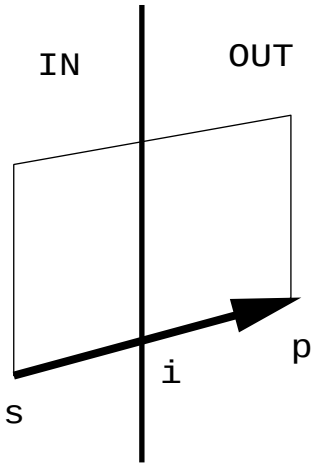
Polygon clipping vs line clipping.



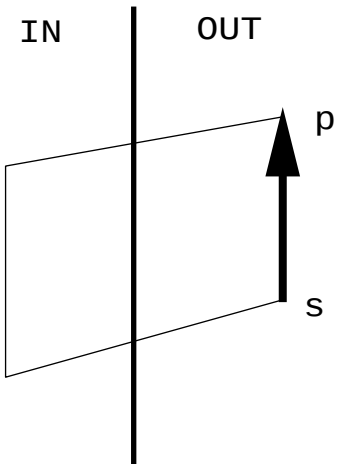
Clipping against four successive boundaries.



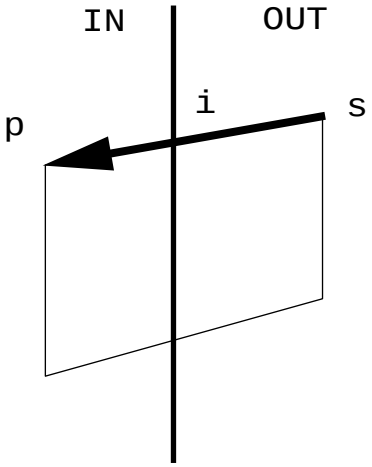
(a) Output p



(b) Output i



(c) No output



(d) Output i and p

Four cases of an edge relative to a clip boundary.