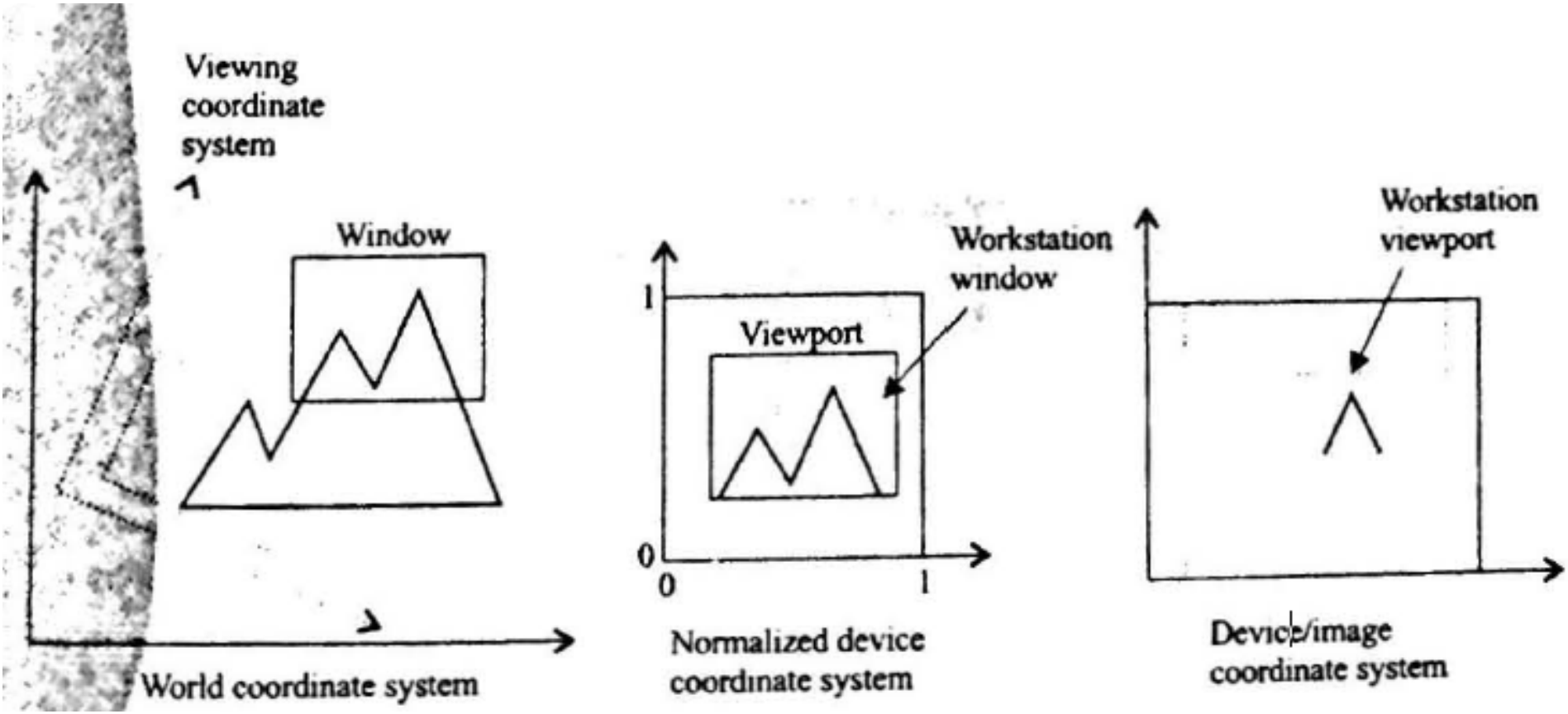
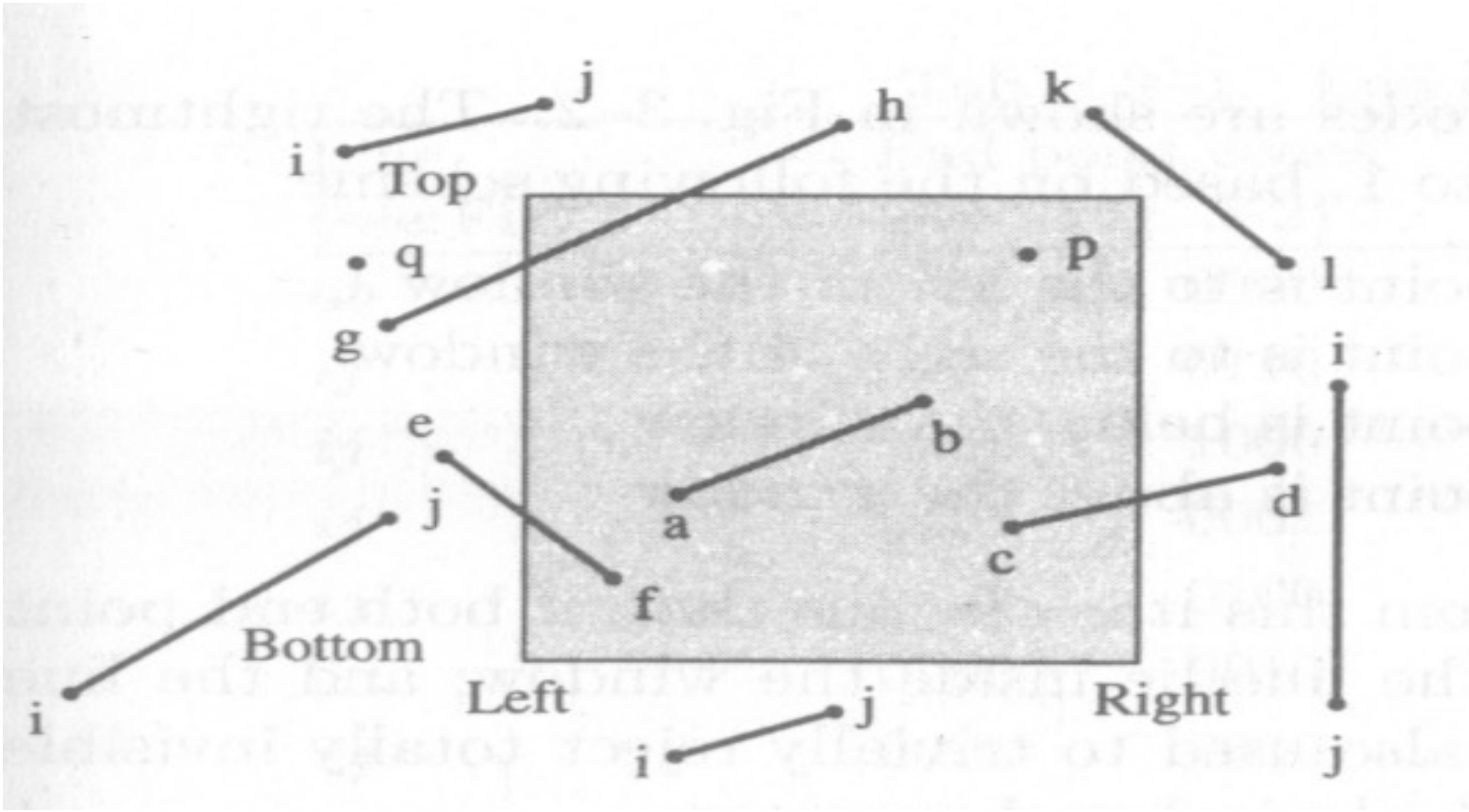
TWO DIMENSIONAL VIEWING & CLIPPING

INTRODUCTION



INTRODUCTION



POINT CLIPPING

- Point clipping is essentially the evaluation of the following inequalities:
- Xmin ≤ *x* ≤ Xmax and Ymin ≤ *Y* ≤ Ymax
- where Xmin, Xmax, Ymin and Ymax, define the clipping window.
- A point *(x, y)* is considered inside the window when the inequalities all evaluate to true.

LINE CLIPPING

- Lines that do not intersect the clipping window are either completely inside the window or completely outside the window.
- On the other hand, a line that intersects the clipping window is divided by the intersection point(s) into segments that are either inside or outside the window.

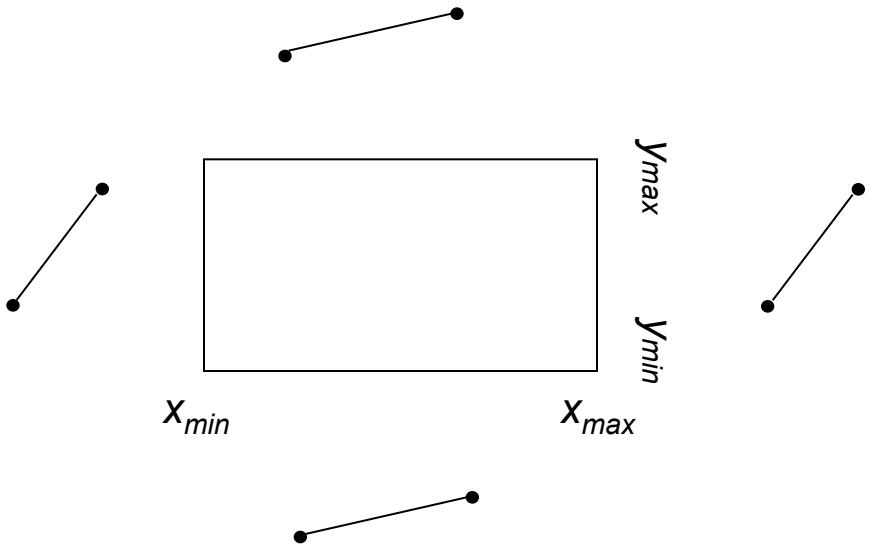
LINE CLIPPING-Visibility

- The visibility study provides efficient way to decide the spatial relationship between an arbitrary line and the clipping window and to find intersection point(s).

SIMPLE VISIBILITY ALGORITHM

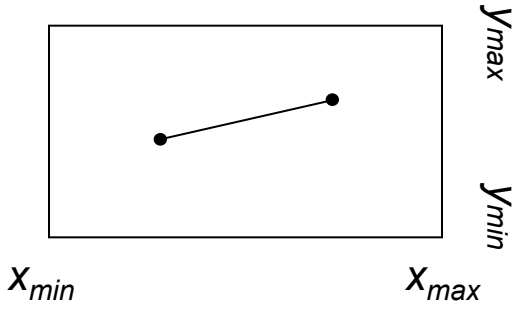
1. If $x_0 < x_{min}$ ***and*** $x_1 < x_{min}$
   or $x_0 > x_{max}$ ***and*** $x_1 > x_{max}$
   or $y_0 < y_{min}$ ***and*** $y_1 < y_{min}$
   or $y_0 > y_{max}$ ***and*** $y_1 > y_{max}$

   Trivial rejection.



2. If $x_{min} \le x \le x_{max}$
   and $y_{min} \le y \le y_{max}$

   Trivially accepted.



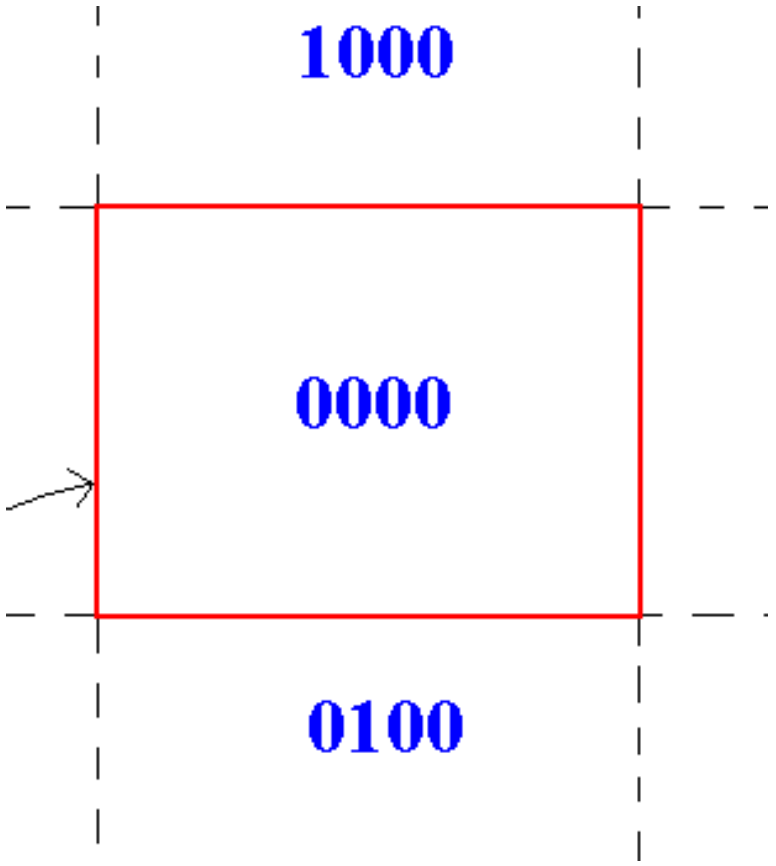
END POINT CODES

- The tests for totally visible lines and the region tests given above for totally invisible lines are formalized using a technique due to Dan Cohen and Ivan Sutherland.
- The technique uses a 4 bit (digit) code to indicate which of nine regions contains point within it.

END POINT CODES

The end point of a line.

- The 4 bit codes are shown in Figure
- The rightmost bit is the first bit.
- The bits are set to 1, based on the following scheme:
  - First-bit set : if the end point is to the left of the window
  - Second-bit set : if the end point is to the right of the window
  - Third-bit set : if the end point is below the window
  - Fourth-bit set : if the end point is above the window
  - Otherwise, the bit is set to zero.



Cohen—Sutherland Line Clipping Algorithm

- A clipping algorithm developed by Dan Cohen and Ivan Sutherland uses the end point codes to trivially accept or reject a line segment.
- In this algorithm we divide the line clipping process into two phases:
  - Identify those lines which intersect the clipping window and so need to be clipped.
  - Perform the clipping.

Cohen—Sutherland Subdivision Line Clipping Algorithm

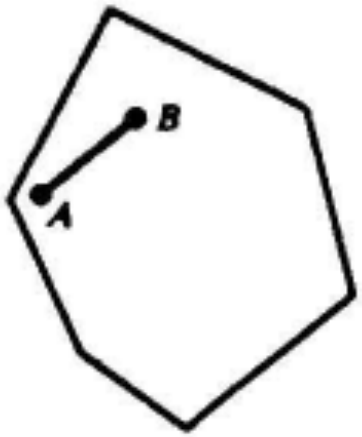
- For each window edge:
  - For the line P determine if the line is totally visible or can be trivially rejected as invisible.
  - If P is outside the window, continue;
  - Otherwise, swap P and Replace P with the intersection of P and the window edge.

Mid Point Algorithm

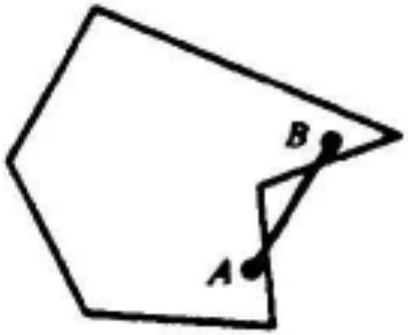
- For each end point:
  - If the end point is visible, then it is the farthest visible point. The process is complete. If not, continue.
  - If the line is trivially invisible, no output is generated. The process is complete. If not, continue.
  - Guess at the farthest visible point by dividing the line P1p2 at its midpoint, Pm. Apply the previous tests to the two segments P1Pm and PmP2 is rejected as trivially invisible, the midpoint is an overestimation of the farthest visible point. Continue with P1Pm.
  - Otherwise, the midpoint is an underestimation of the farthest visible point. Continue with P2Pm.
  - If the segment becomes so short that the midpoint corresponds to the accuracy of the machine or, as specified, to the end points, evaluate the visibility of the point and the process is complete.

POLYGON CLIPPING

- A polygon is called *convex* if the line joining any two interior points of the polygon lies completely inside the polygon.
- A non-convex polygon is said to be *concave.*



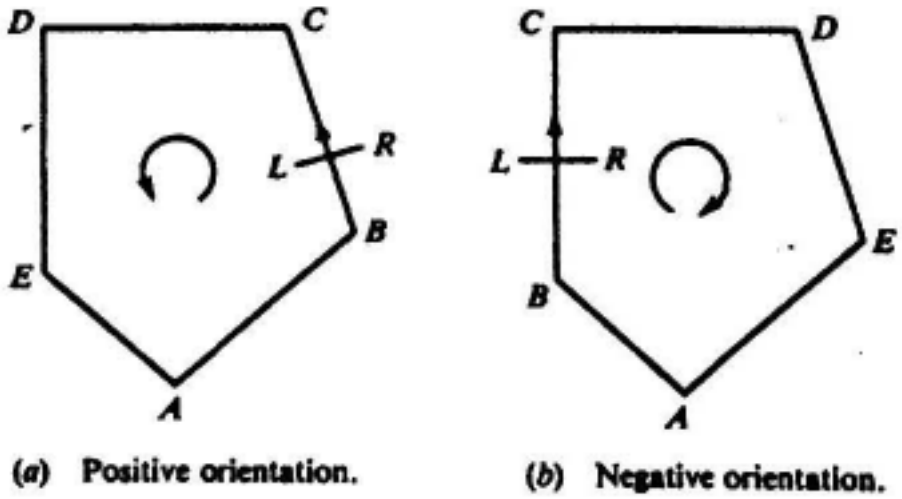
Convex polygon



Concave polygon

POLYGON CLIPPING

- By convention, a polygon with vertices P_1' ... ' P_n' (and edges P_{i-1} P_i and $P_N P_1$) is said to be *positively oriented* if a tour of the vertices in the given order produces a counterclockwise circuit.
- Equivalently, the left hand of a person standing along any directed edge $P_{i-1}P_i$, or P_nP_1 would be pointing inside the polygon.



(a) Positive orientation. (b) Negative orientation.

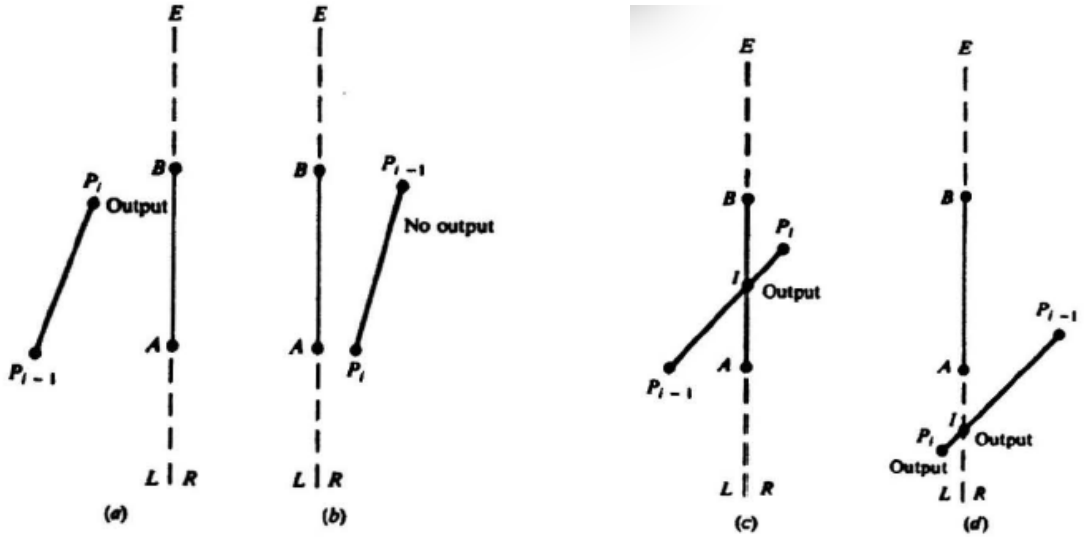
Sutherland-Hodgman polygon clipping

- Let $P_1 \ldots P_n$ be the vertex list of the polygon to be clipped.
- Let edge E, determined by endpoints A and B, be any edge of the positively oriented, convex clipping polygon.
- We clip each edge of the polygon in turn against the edge E of the clipping polygon, forming a new polygon whose vertices are determined as follows.

Sutherland-Hodgman polygon clipping

Consider the edge $\overline{P_{i-1}P_i}$:

1. If both P_{i-1} and P_i are to the left of the edge, vertex P_i is placed on the *vertex output list* of the clipped polygon [Fig. 5-9(*a*)].
2. If both P_{i-1} and P_i are to the right of the edge, nothing is placed on the vertex output list [Fig. 5-9(*b*)].
3. If P_{i-1} is to the left and P_i is to the right of the edge E, the intersection point I of line segment $\overline{P_{i-1}P_i}$ with the extended edge E is calculated and placed on the vertex output list [Fig. 5-9(*c*)].
4. If P_{i-1} is to the right and P_i is to the left of edge E, the intersection point I of the line segment $\overline{P_{i-1}P_i}$ with the extended edge E is calculated. Both I and P_i are placed on the vertex output list [Fig. 5-9(*d*)].



Weller-Atherton Polygon clipping

- We start with an arbitrary vertex of the subject polygon and trace around its border in the clock wise direction until an intersection with the clip polygon is encountered:
  - If the edge enters the clip polygon, record the intersection point and continue to trace the subject polygon.
  - If the edge leaves the clip polygon, record the intersection point and make a right turn to follow the clip polygon in the same manner.