WINDOWING AND CLIPPING
(Chapter 6 in *Computer Graphics*)

- Windowing Concepts
- Clipping Algorithms
- Window-to-viewport Transformation

windowing

- specify an area of the world coordinate system to be displayed

viewporting

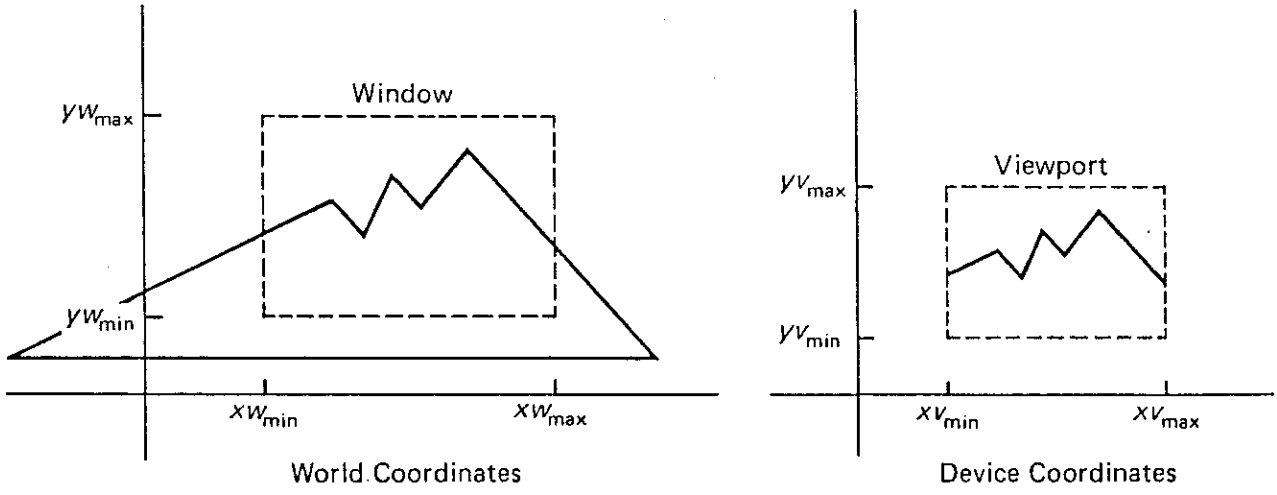
- specifying where on the display device it is to be displayed
  - a single area
  - multiple separate areas
  - one area inside another

clipping

- removing picture parts outside an area

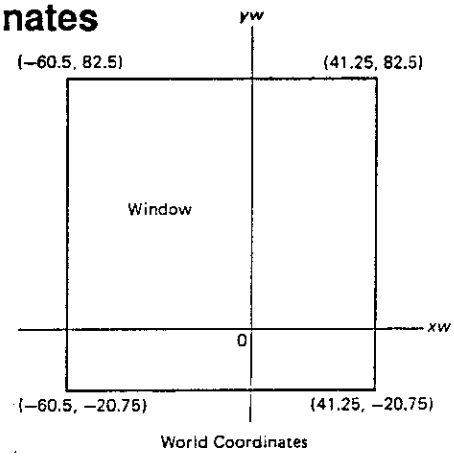
Windowing Concepts

- window
  - a rectangular area specified in world coordinates
- viewport
  - a rectangular area on the display device to which the window is mapped
- mapping
  - viewing transformation or windowing transformation or normalizing transformation

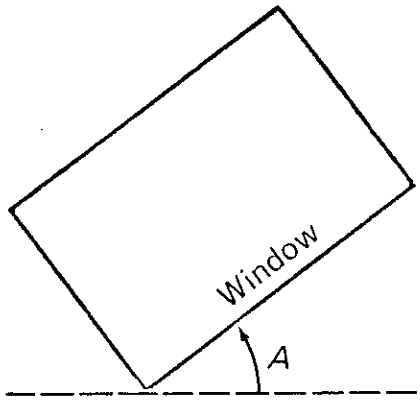


establishing the window and the viewport

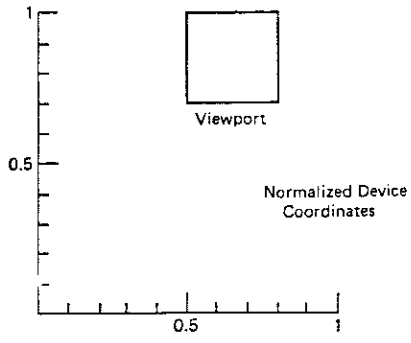
- set_window (xw_min, yw_min, xw_max, yw_max)
  - world coordinates



  - an angle of rotation or an up-vector can be specified

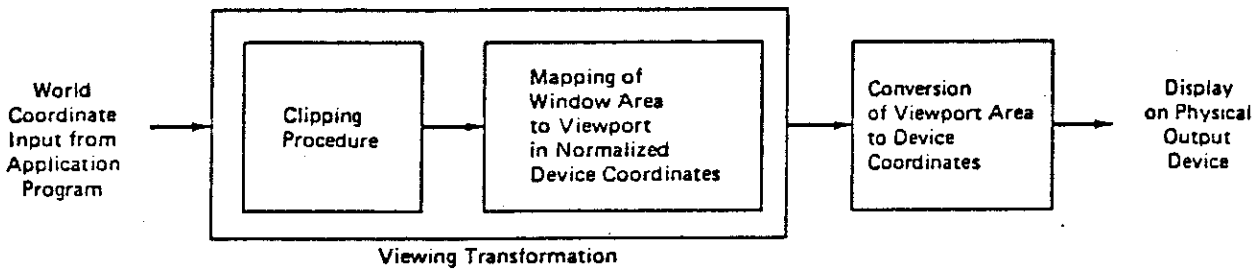


- set_viewport (xv_min, yv_min, xv_max, yv_max)
  - usually normalized device coordinates
  - each viewport can be assigned a relative priority



changing positions and sizes

- viewport
  - a changed position permits display at a different position on the output device
  - a changed size permits an enlargement or a reduction
  - changing the aspect ratio changes proportions directly
- window
  - a changed position selects a different portion of the world
  - moving the window left is like panning right
  - a changed size selects larger or smaller portions of the world
  - shrinking the window is like zooming in
  - changing the aspect ratio changes proportions inversely


World Coordinate Input from Application Program
Clipping Procedure
Mapping of Window Area to Viewport in Normalized Device Coordinates
Viewing Transformation
Conversion of Viewport Area to Device Coordinates
Display on Physical Output Device

implementations of the windowing transformation

- one approach
  - clip against the window
  - map the window interior into the viewport
- another approach
  - map world coordinates into normalized device coordinates
  - clip against the viewport

Clipping Algorithms

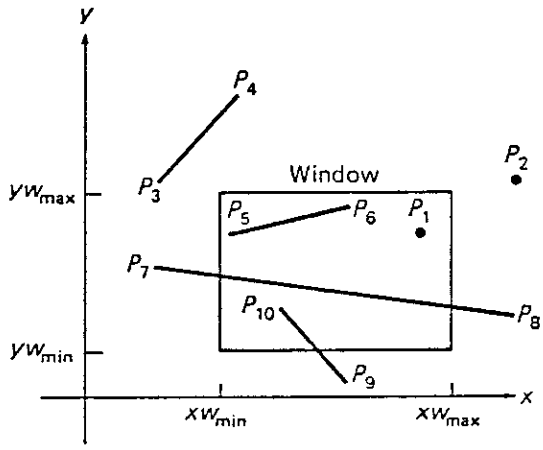
- point clipping
- line clipping
  - Cohen and Sutherland
  - Cohen and Sutherland with midpoint subdivision
  - Liang and Barsky
  - Nicholl, Lee and Nicholl
- area clipping
  - Sutherland and Hodgman
  - Weiler and Atherton
- text clipping
- curve clipping
- blanking

point clipping (in world coordinates)

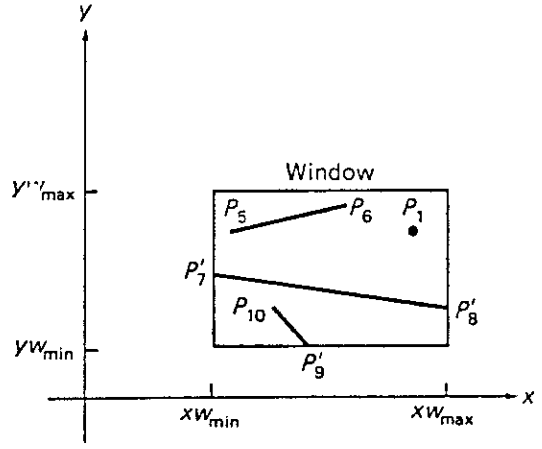
- a point is saved if
  - $xw_{min} \leq x \leq xw_{max}$ and
  - $yw_{min} \leq y \leq yw_{max}$

point and line clipping (in world coordinates)

- determine points and line segments
  - wholly within the window
  - wholly outside
  - partially outside



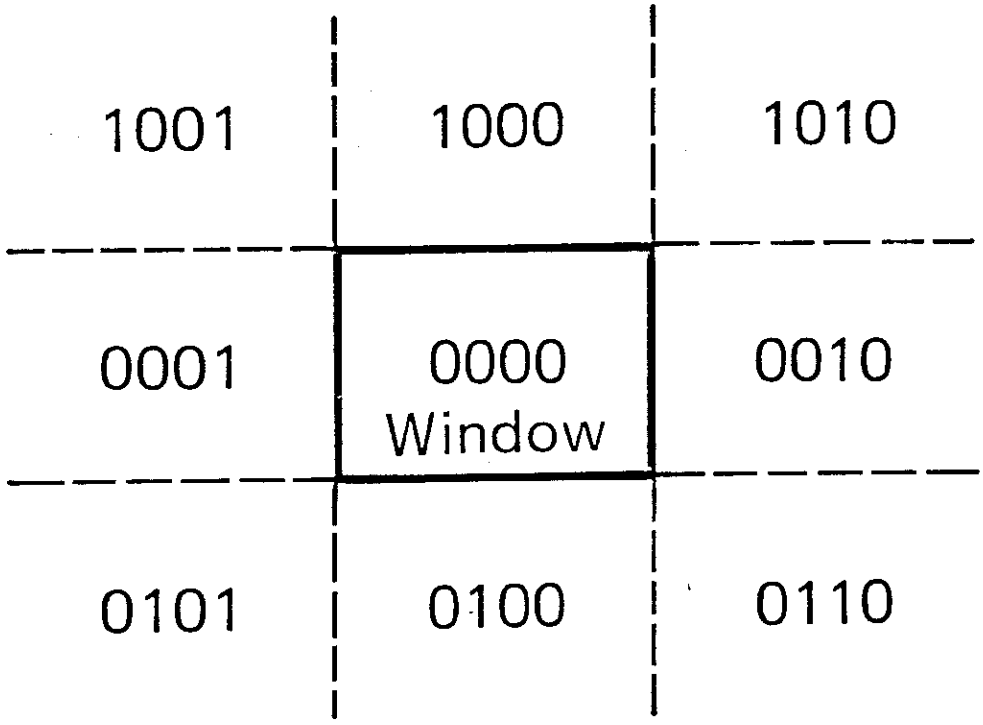
Before Clipping
(a)



After Clipping
(b)

Cohen and Sutherland line clipping (to the window)

- assign a region code to each endpoint
  - bit 1 - left
  - bit 2 - right
  - bit 3 - below
  - bit 4 - above

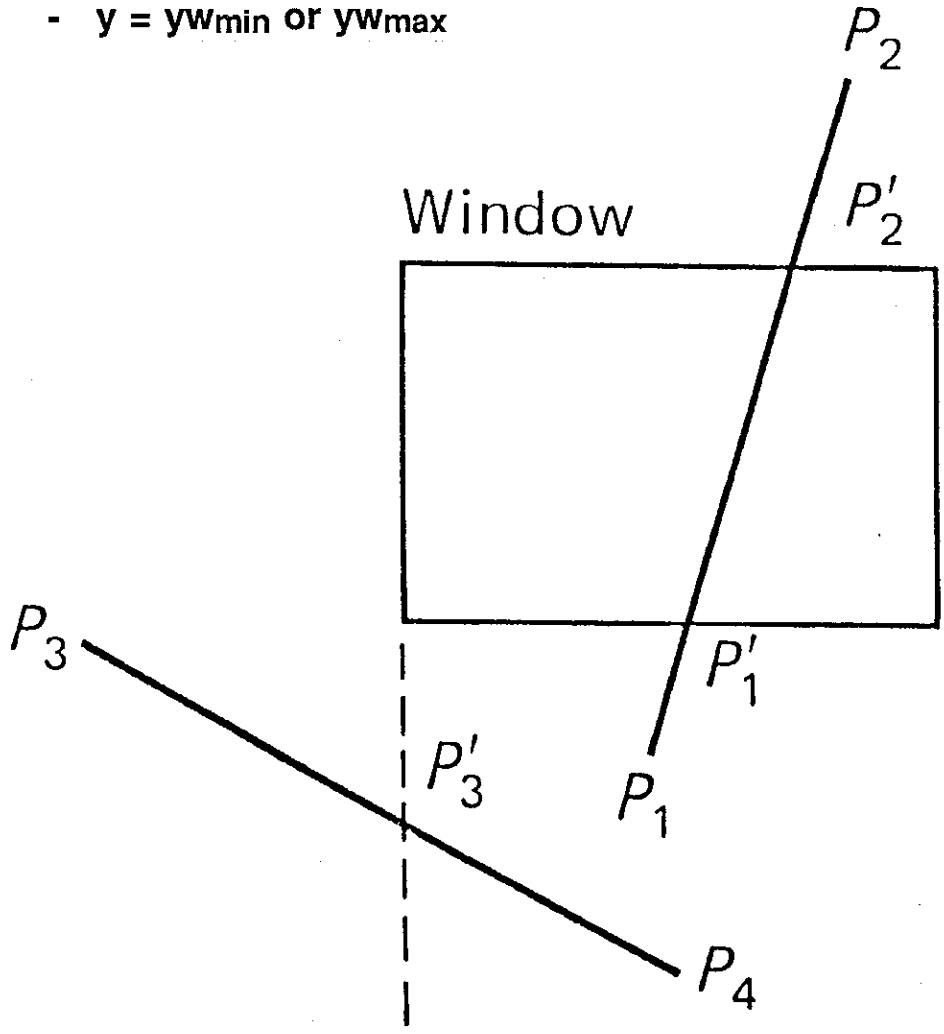


Cohen and Sutherland, continued

- line segments wholly within the window have region codes of 0000 for both endpoints
- line segments that have 1s in the same bit position in the region codes (nonzero logical AND) are wholly outside
- identify boundary crossings by bits which change, and subdivide

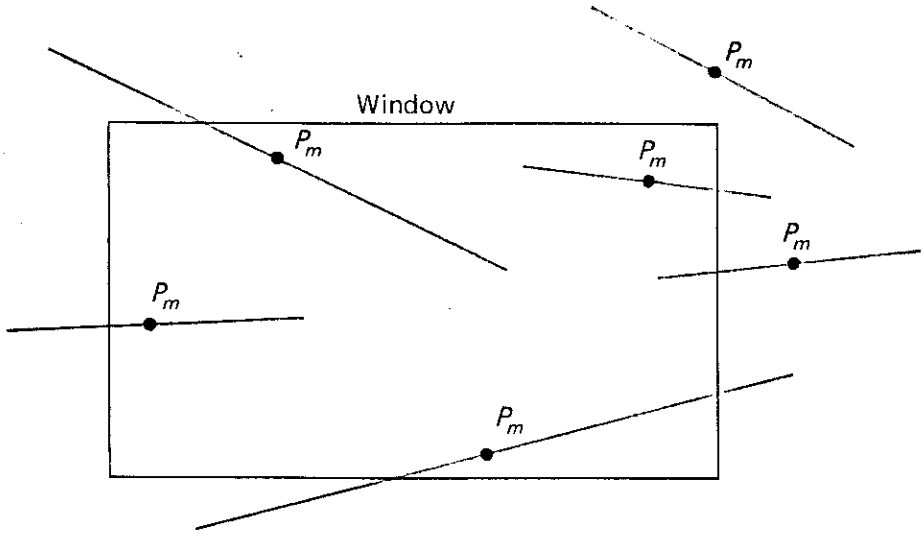
Cohen and Sutherland, continued

- intersection with a vertical boundary
  - $x = xw_{min}$ or xw_{max}
  - $y = y_1 + m(x - x_1)$
- intersection with a horizontal boundary
  - $x = x_1 + (y - y_1)/m$
  - $y = yw_{min}$ or yw_{max}



Cohen and Sutherland with midpoint subdivision

- subdivide line segments which cannot be wholly accepted or wholly clipped
- midpoint calculation
  - $x_m = (x_1 + x_2)/2$
  - $y_m = (y_1 + y_2)/2$



- process each half of the line segment
- recur until an intersection point is found

Liang and Barsky

- uses a parametric form of the line equation
  - $x = x_1 + (x_2 - x_1)u$
  - $y = y_1 + (y_2 - y_1)u$
  - $0 \leq u \leq 1$
- for points inside the window
  - $xw_{min} \leq x_1 + (x_2 - x_1)u \leq xw_{max}$
  - $yw_{min} \leq y_1 + (y_2 - y_1)u \leq yw_{max}$
  - or
  - $p_k u \leq q_k$, $k = 1, 2, 3, 4$
  - where

    $p_1 = -(x_2 - x_1)$
  - $q_1 = x_1 - xw_{min}$

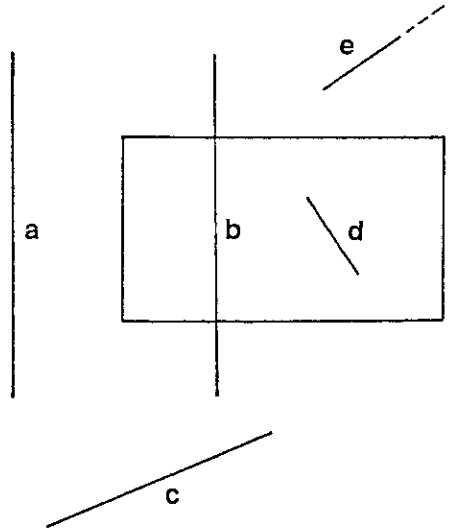
    $p_2 = (x_2 - x_1)$
  - $q_2 = xw_{max} - x_1$

    $p_3 = -(y_2 - y_1)$
  - $q_3 = y_1 - yw_{min}$

    $p_4 = (y_2 - y_1)$
  - $q_4 = yw_{max} - y$

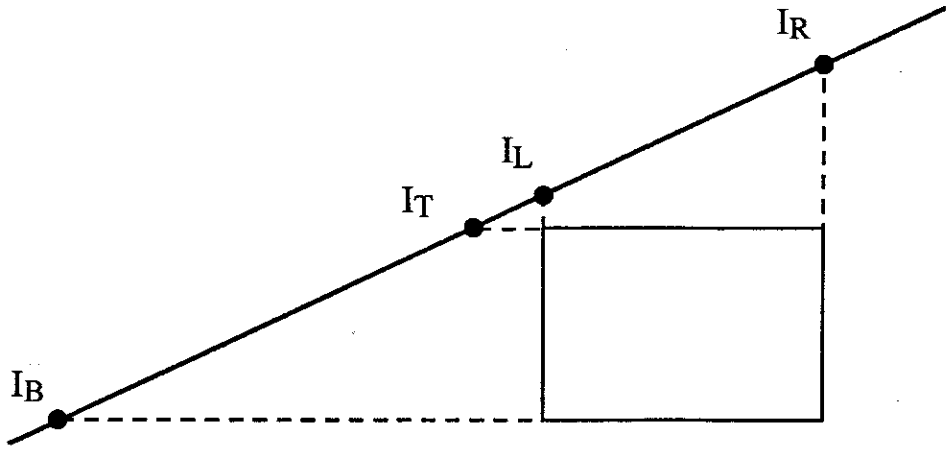
Liang and Barsky, continued

- line segment parallel to the window boundary
  - $p_k = 0$
  - line segment completely outside the window boundary
    - $q_k < 0$
  - line segment completely inside the window boundary
    - $q_k \geq 0$
- infinite extension of the line segment proceeds from outside to inside the infinite extension of the window boundary
  - $p_k < 0$
- infinite extension of the line segment proceeds from inside to outside the infinite extension of the window boundary
  - $p_k > 0$
- point of intersection of infinitely extended line segment with infinitely extended window boundary for nonzero p_k
  - $u = q_k/p_k$



Nicholl, Lee and Nicholl

- avoids computing intersection points which are not endpoints of the resultant line segment



- performance is superior to Cohen and Sutherland and to Liang and Barsky

Nicholl, Lee and Nicholl, continued

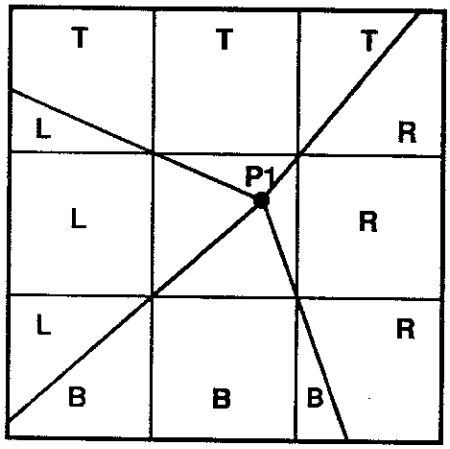
given the endpoints of a line segment, P1 and P2, characterize the location of P1 among the nine regions

top left corner	top edge	top right corner
left edge	window	right edge
bottom left corner	bottom edge	bottom right corner

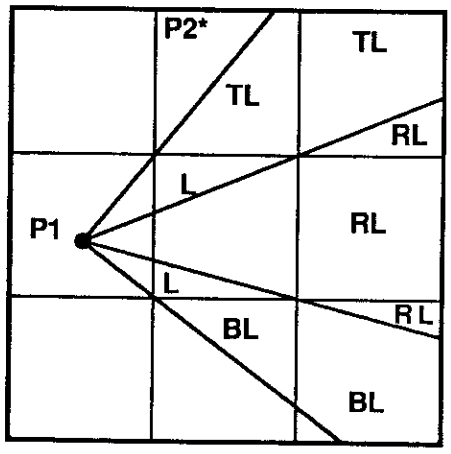
Nicholl, Lee and Nicholl, continued

- then characterize the location of P2 in the appropriate subdivision

- P1 in the window



- P1 in the edge region

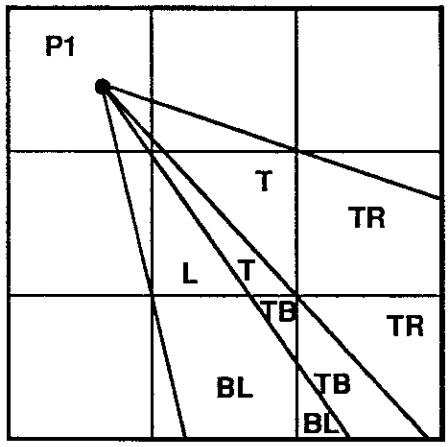


- P2 is above the line through the top left corner of the window iff

$$\frac{ytop-y1}{xleft-x1} < \frac{y2-y1}{x2-x1}$$

Nicholl, Lee and Nicholl, continued

- P1 in a corner region



  - the number of characters in the subdivision indicates the number of intersections

- now calculate the endpoints of the resultant line segment

Nicholl, Lee and Nicholl, continued

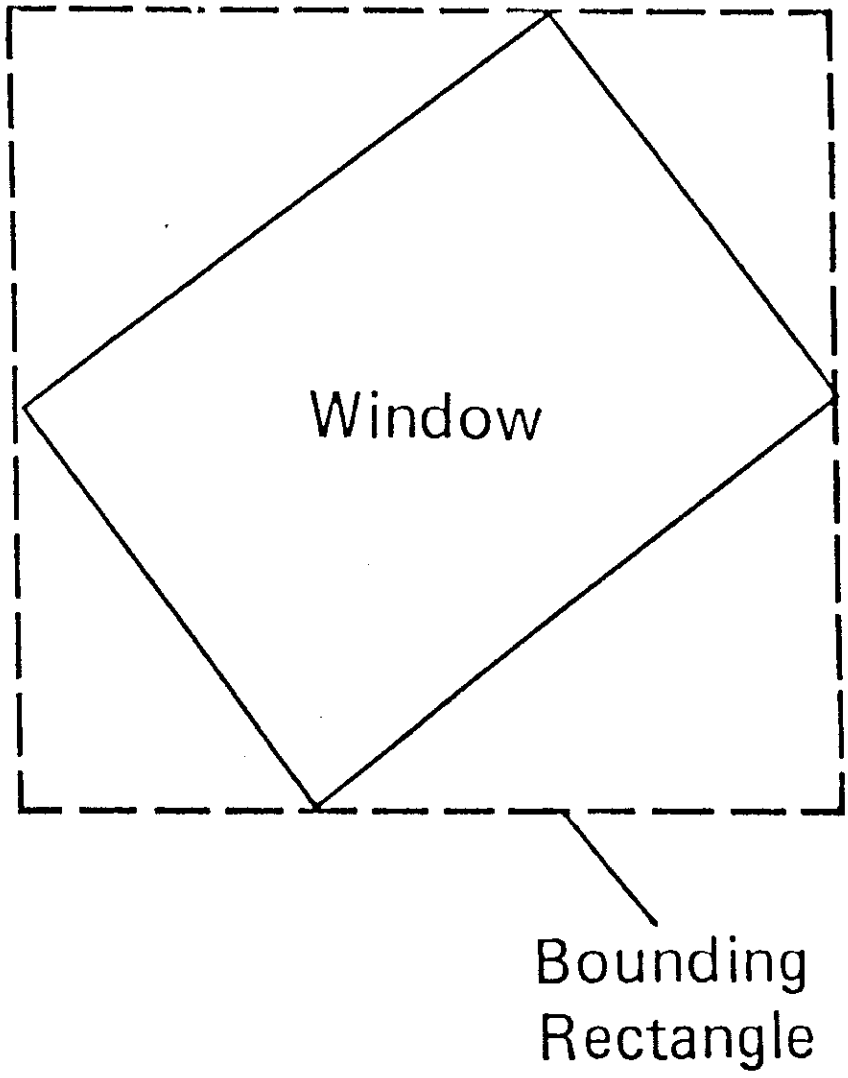
- characterizing endpoints
  - P1 is straightforward
  - P2

    if it lies on an extended window boundary, the subdivision is colinear with the window boundary

    oblique subdivision boundaries pass through window corners
- exploiting symmetry
  - transform all four corner region cases to a single corner region case by computationally simple rotations of 0°, 90°, 180°, or 270°
  - transform all four region cases to a single edge region case by computationally simple reflections about the x-axis or the line x=-y
  - implement by complementing and/or interchanging variables

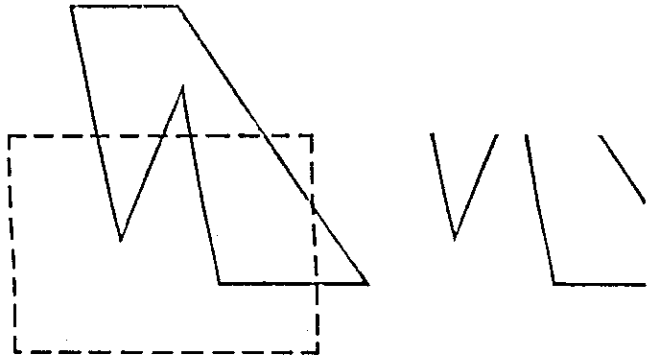
rotated windows

- wholly clip relative to a bounding rectangle

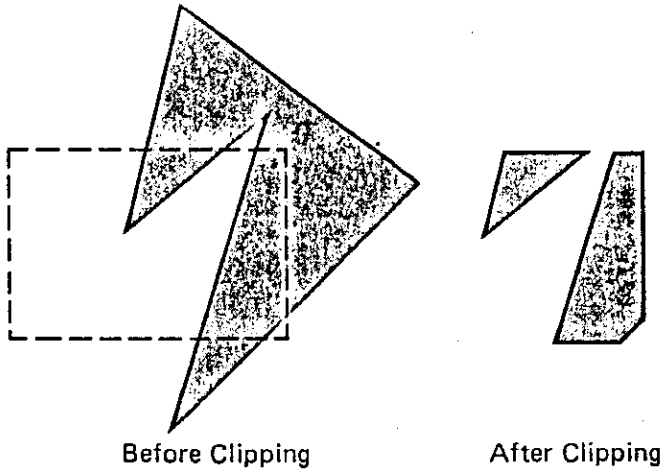


area clipping

- point-plotted polygon boundaries
  - clip each point
- line-drawn polygon boundaries
  - clip each line segment

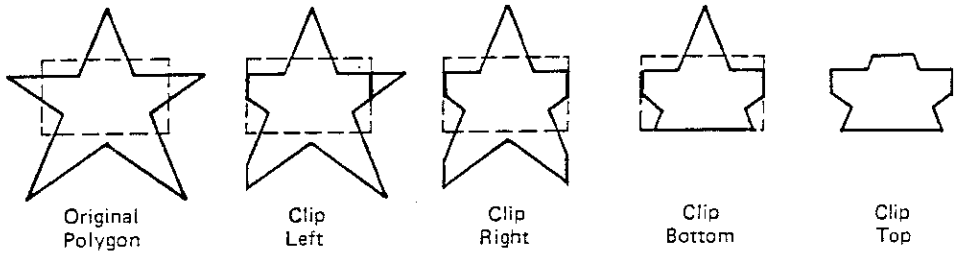


- filled polygons
  - result must be one or more closed polygons



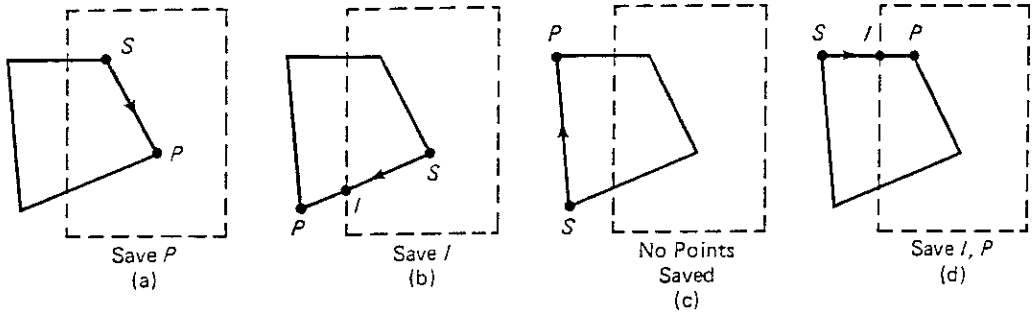
Sutherland and Hodgman clipping (to the window)

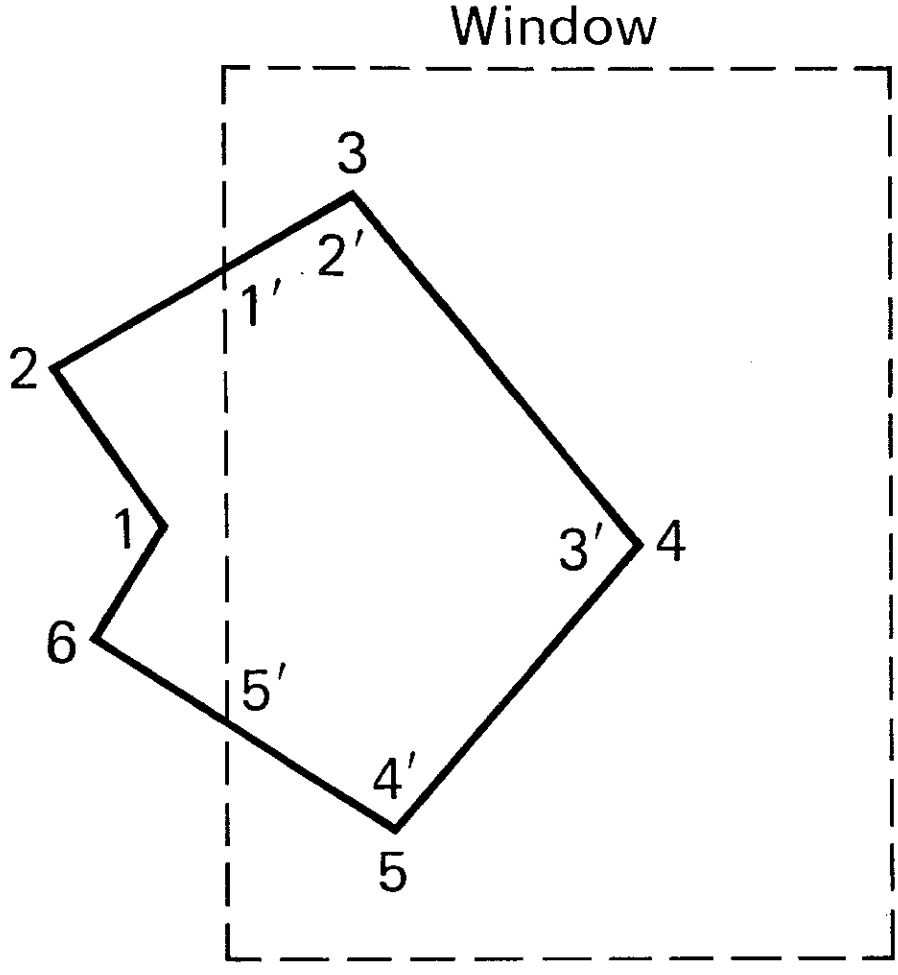
- specify an ordered sequence of vertices
- compare a polygon against each window boundary in turn



Sutherland and Hodgman, continued

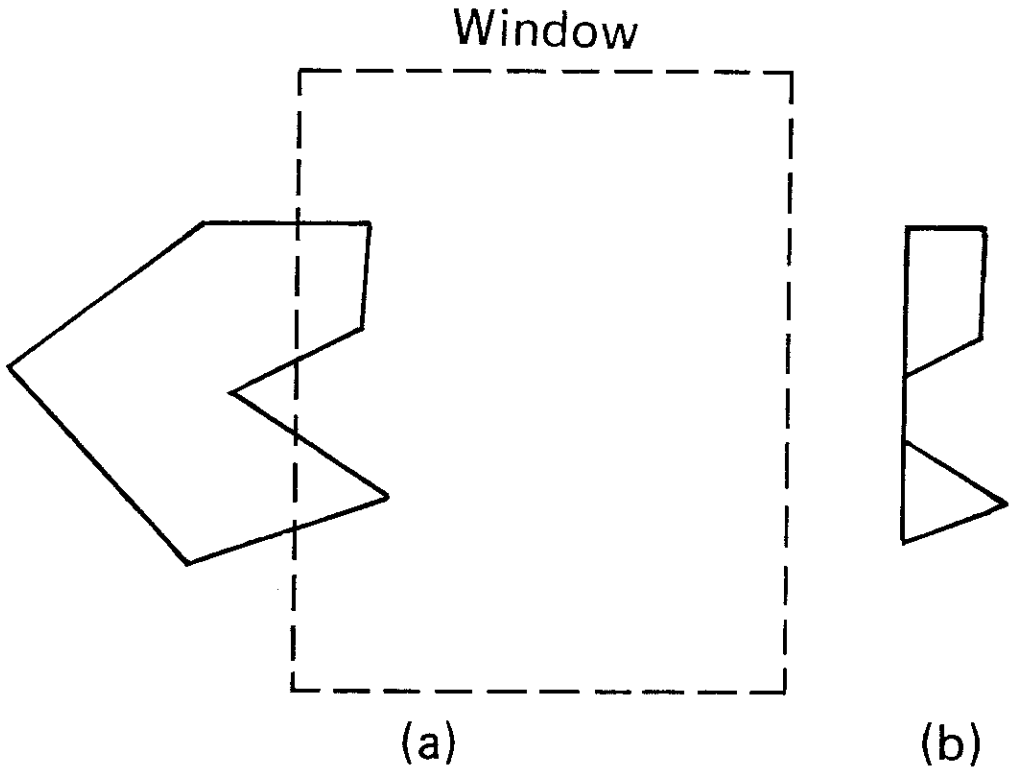
- save vertices inside the edge
- save the intersection when proceeding across a boundary




Window
3
2′
1′
2
1
3′
4
6
5′
4′
5

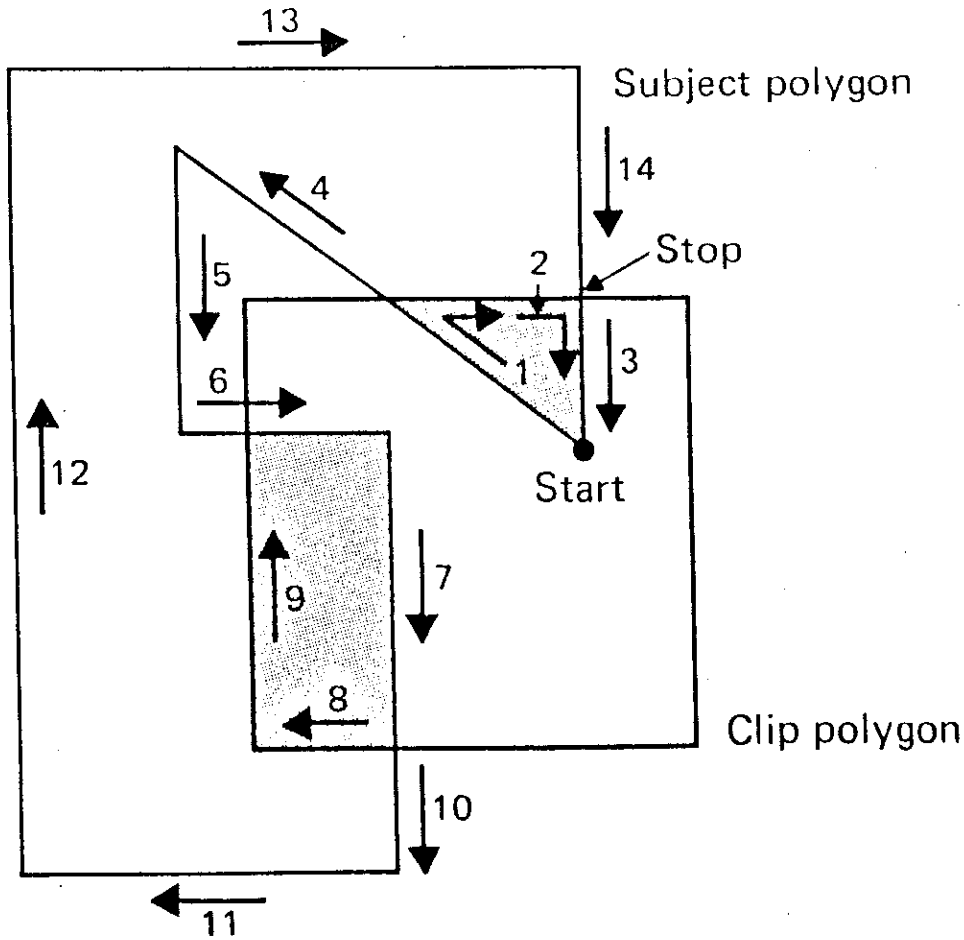
Sutherland and Hodgman, continued

- unwanted lines



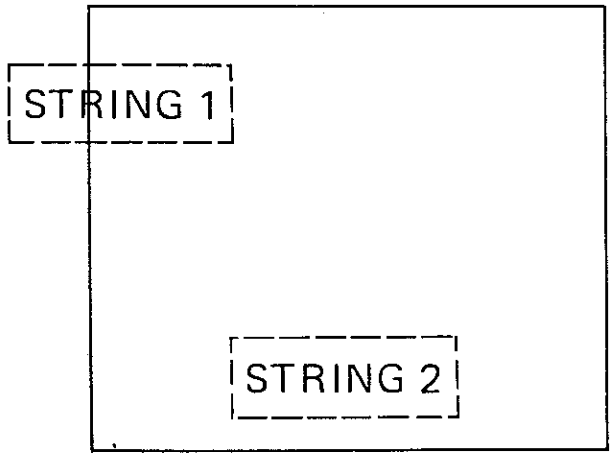
Weiler and Atherton clipping (to the window)

- trace around the border of the subject polygon clockwise until an intersection with the clip polygon is encountered
  - if the edge enters the clip polygon, proceed along the subject polygon edge
  - if the edge leaves the clip polygon, make a right turn and follow the clip polygon; if the subject polygon is encountered, make a right turn and follow the subject polygon
- remember intersections so that all paths are traced exactly once

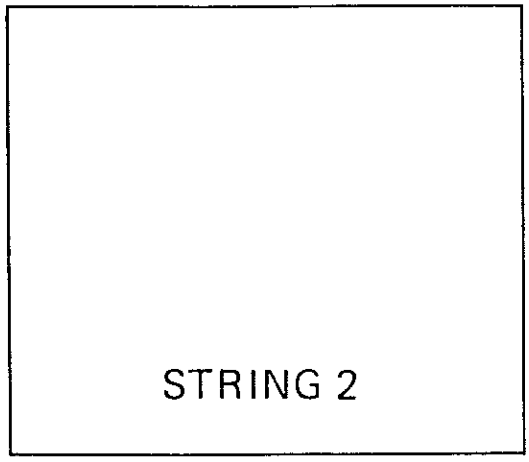


text clipping

- "all-or-nothing" text clipping



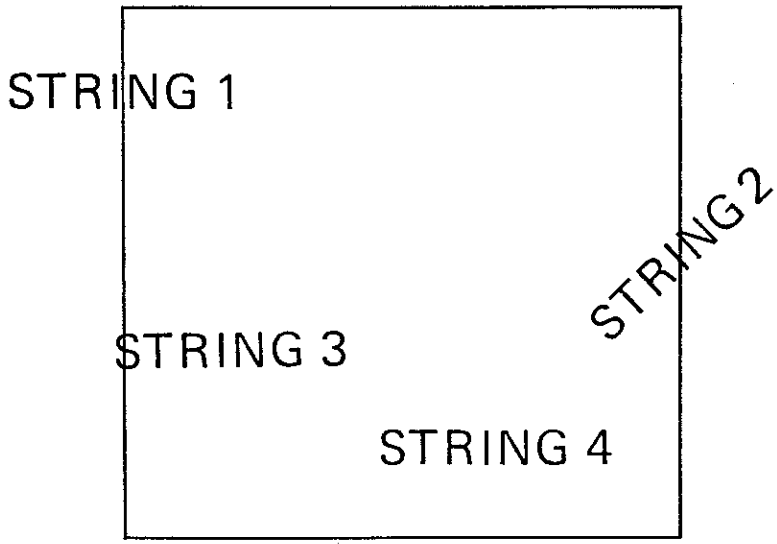
Before Clipping



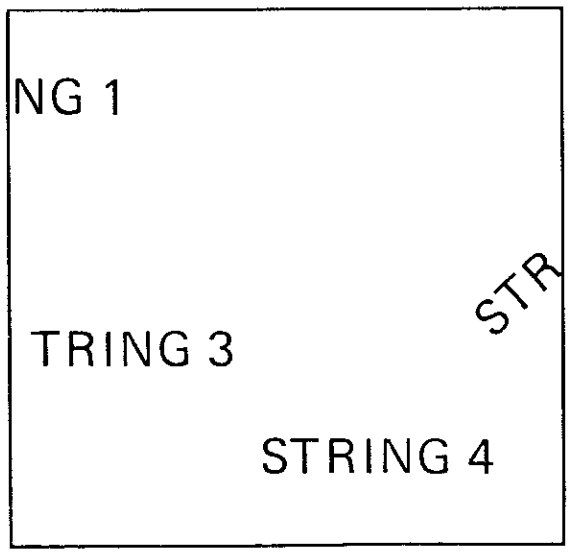
After Clipping

text clipping, continued

- "all-or-nothing" character clipping



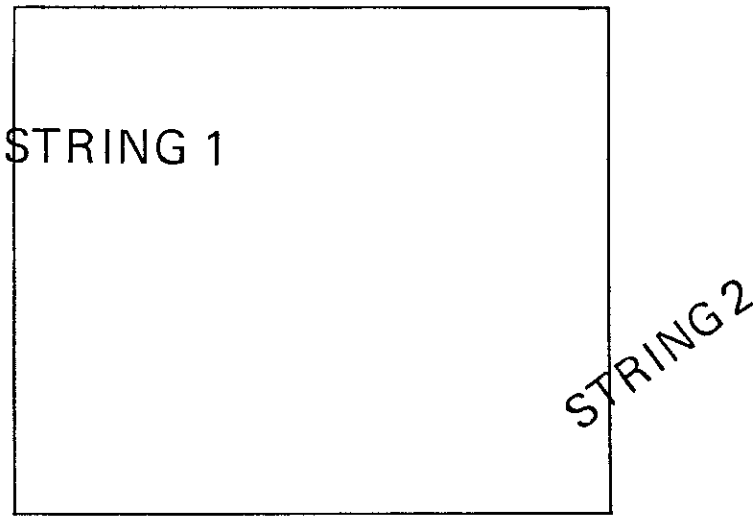
Before Clipping



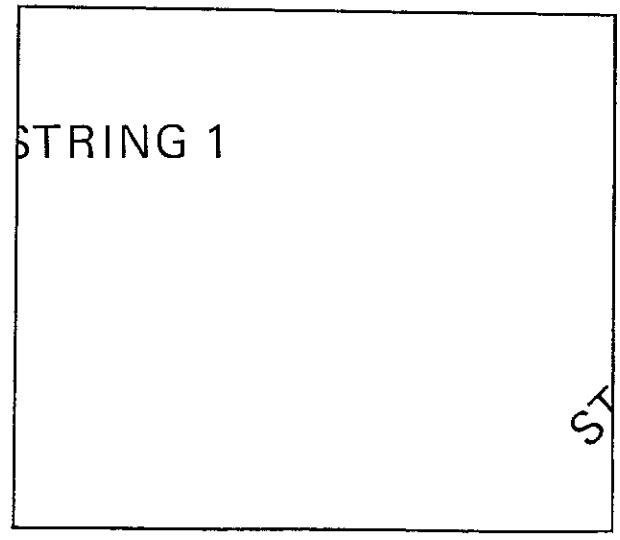
After Clipping

text clipping, continued

- individual character clipping
  - line-drawn characters
    - clip individual line segments
  - bit-map characters
    - clip individual pixels



Before Clipping



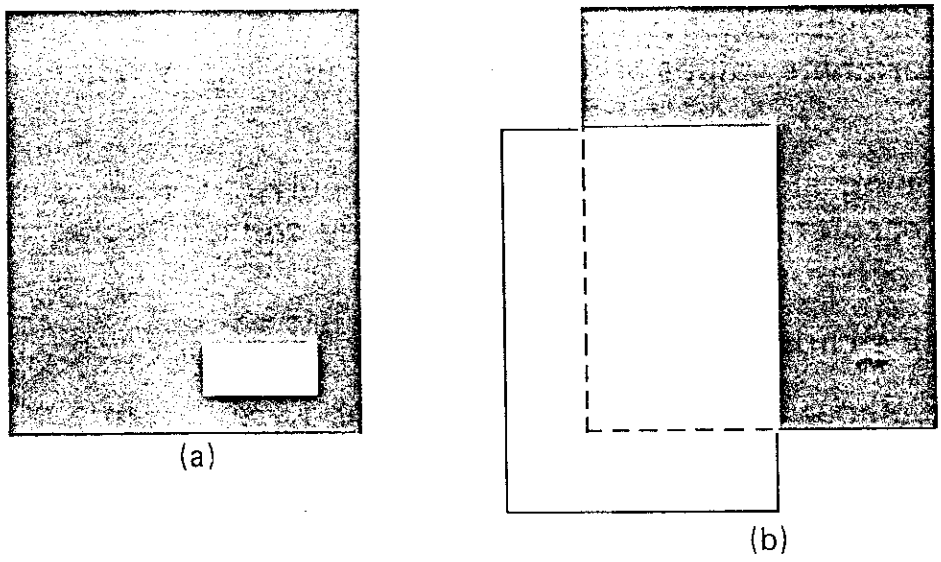
After Clipping

Curve clipping

- point-plotted curves
  - clip individual points
- curves approximated by polylines
  - clip individual line segments
- parametric curves
  - convert to point-plotted curves or polyline approximations and clip

blanking

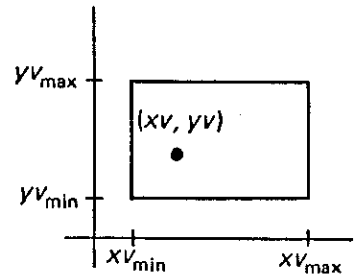
- erase (blank) anything within the window area
- used for overlaying displays



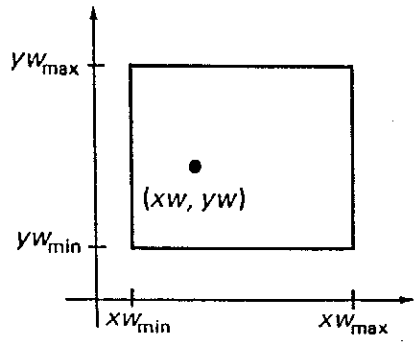
Window-to-viewport Transformation

- mapping points from the window onto the viewport

$$\frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}} = \frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}}$$



$$\frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}} = \frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}}$$



Therefore

$$xv = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}(xw - xw_{\min}) + xv_{\min}$$

$$yv = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}(yw - yw_{\min}) + yv_{\min}$$

which can be represented in matrix form as a scaling and a translation.

$$Sx = \frac{XV\ max - XV\ min}{XW\ max - XW\ min} \qquad Tx = -Sx \bullet XW\ min + XV\ min$$

$$Sy = \frac{YV\ max - YV\ min}{YW\ max - YW\ min} \qquad Ty = -Sy \bullet YW\ min + YV\ min$$

WINDOWING AND CLIPPING

- Windowing Concepts
- Clipping Algorithms
- Window-to-viewport Transformation