

Viewing & Clipping In 2D

BY:AKANKSHA YADAV

FOET,LU

SUBJECT:COMPUTER GRAPHICS

BRANCH/YEAR:MCA/2ND.

Windowing Concepts

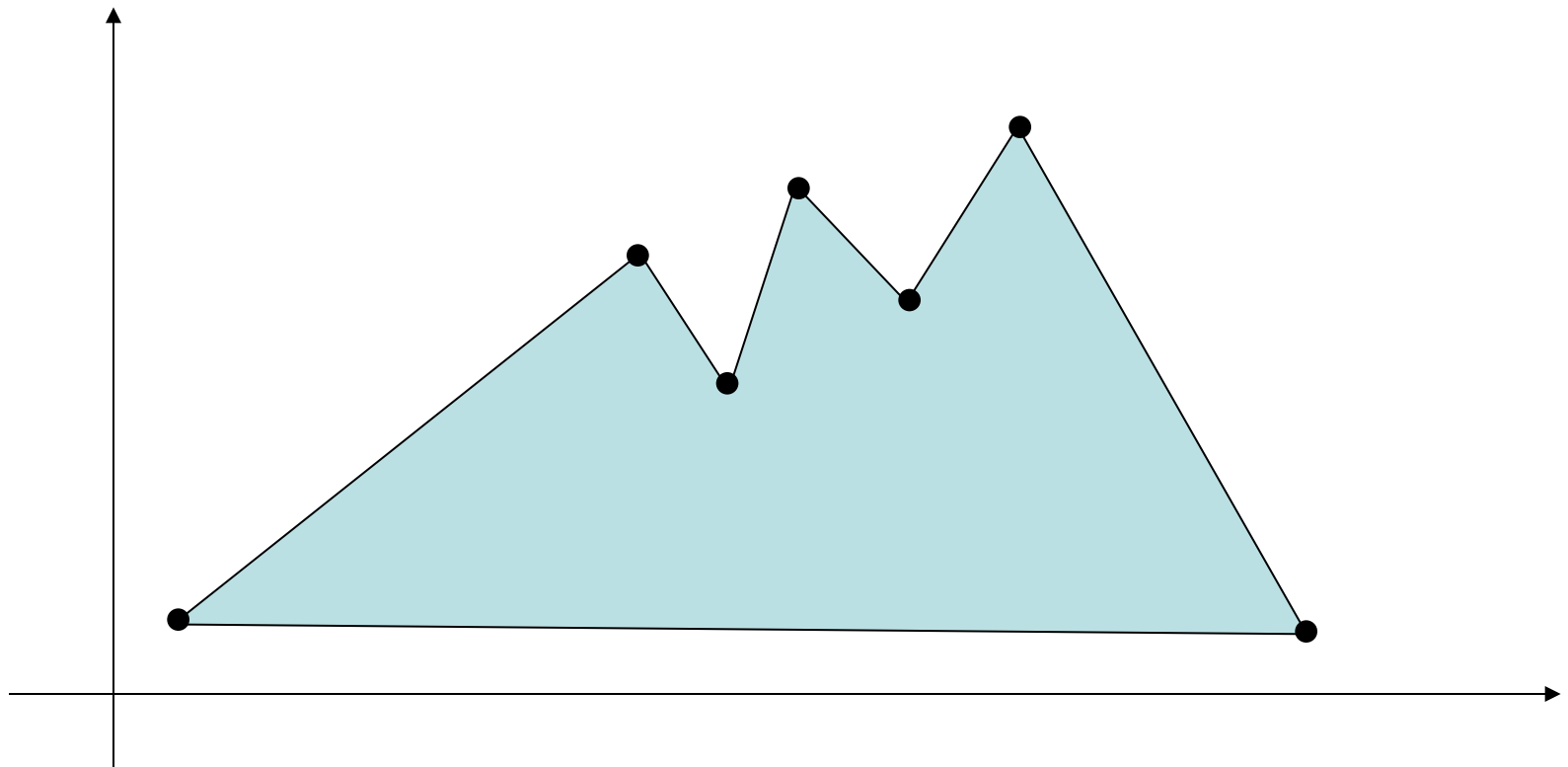
Clipping

- Introduction
- Brute Force
- Cohen-Sutherland Clipping Algorithm

Area Clipping

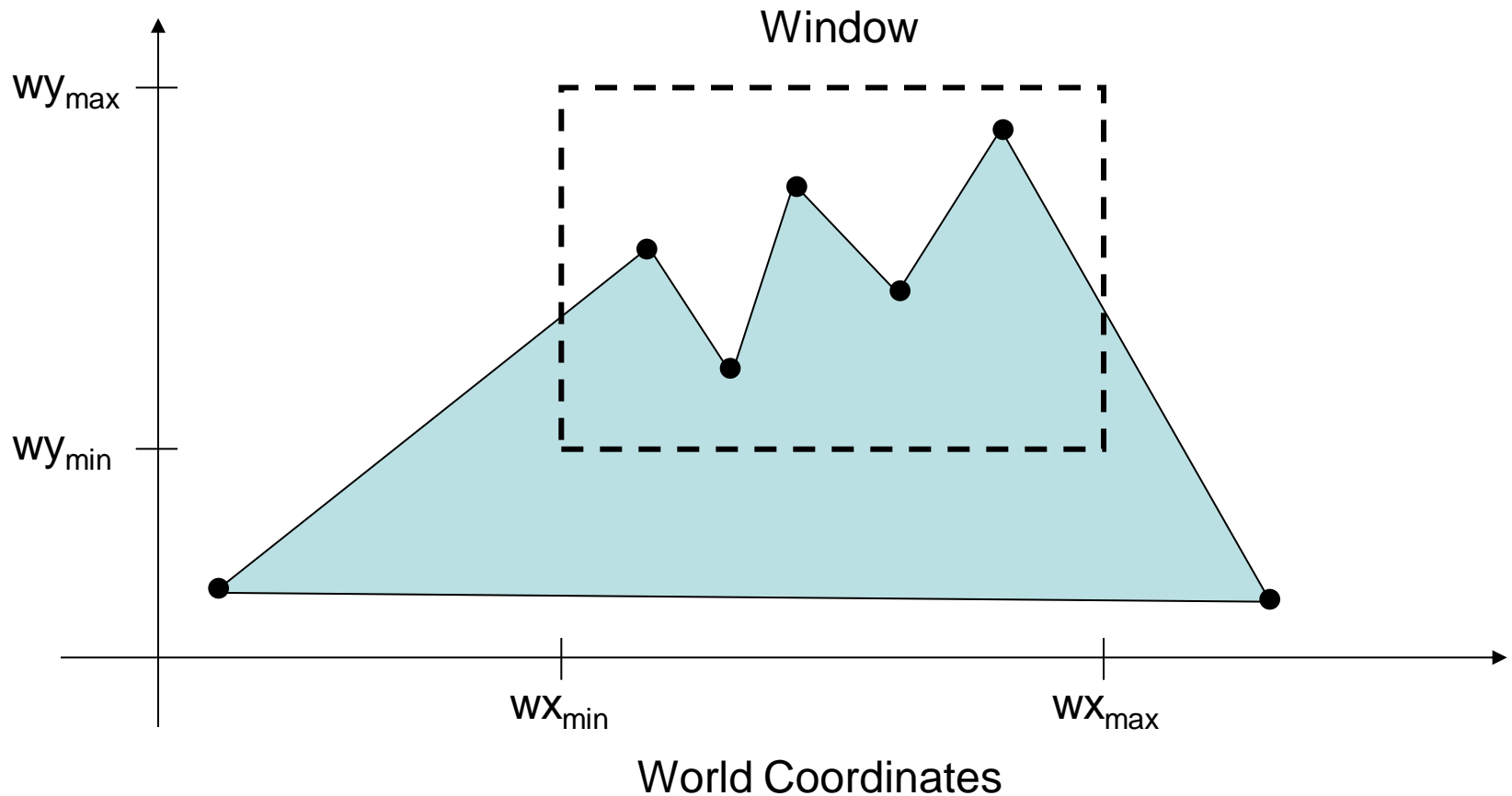
- Sutherland-Hodgman Area Clipping Algorithm

A scene is made up of a collection of objects specified in world coordinates

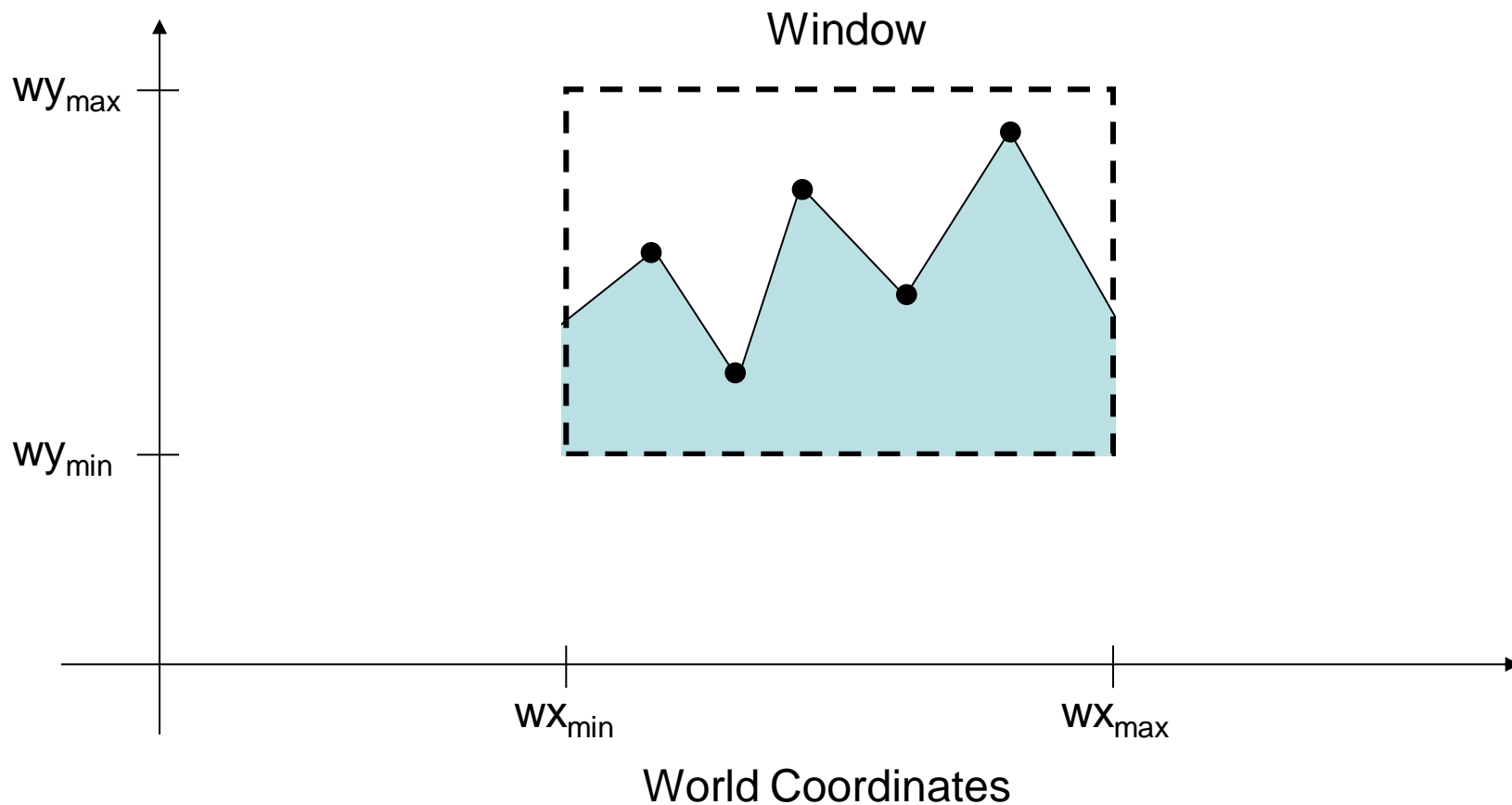


World Coordinates

When we display a scene only those objects within a particular window are displayed



Because drawing things to a display takes time we *clip* everything outside the window



Explain the terms “Window” and “Viewport”. Write the general form of window to viewport coordinate transformation equation.

A world-coordinate area selected for display is called a window.

An area on a display device to which a window is mapped is called a viewport.

The window defines what is to be viewed, the viewport defines where it is to be displayed.

In computer graphics terminology, the term window originally referred to an area of a picture that is selected for viewing.

WINDOW-TO-VIEWPORT COORDINATE TRANSFORMATION

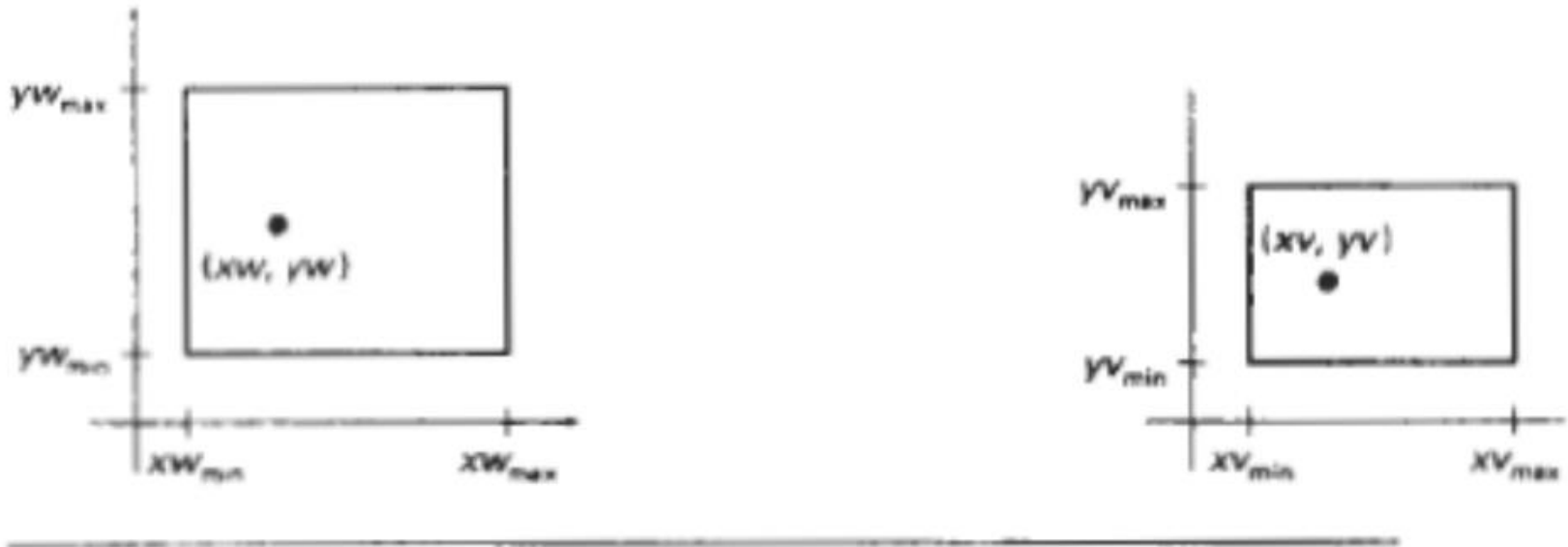


Figure illustrates the window-to-viewport mapping.

A point at position (x_w, y_w) in the window is mapped into position (x_v, y_v) in the associated viewport. To maintain the same relative placement in the viewport as in the window, we require that

$$\frac{x^U - x^U_{\min}}{x^U_{\max} - x^U_{\min}} = \frac{x^W - x^W_{\min}}{x^W_{\max} - x^W_{\min}}$$

$$\frac{y^U - y^U_{\min}}{y^U_{\max} - y^U_{\min}} = \frac{y^W - y^W_{\min}}{y^W_{\max} - y^W_{\min}}$$

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

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

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

where the scaling factors are

$$sx = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$$

$$sy = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$$

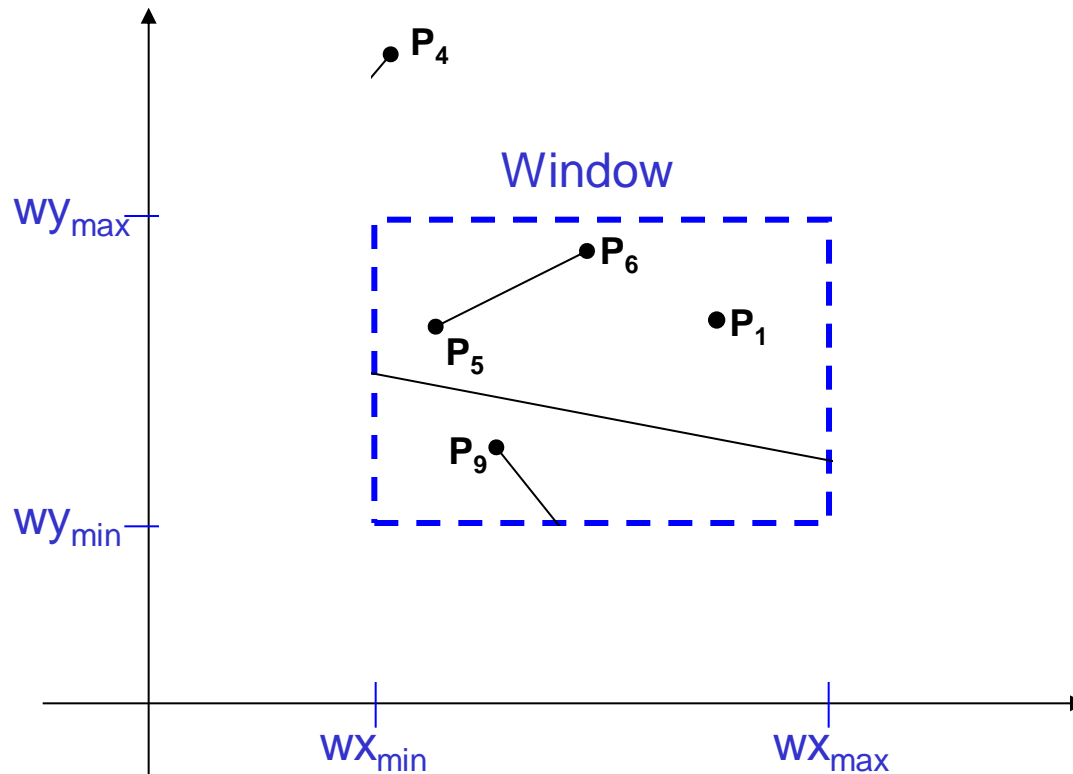
The conversion from window area to viewport area is performed with the following sequence of transformation.

1. Perform a scaling transformation using a fixed point position of (xw_{\min}, yw_{\min}) that scales the window area to the size of the viewport.
2. Translate the scaled window area to the position of the viewport.

Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of a space is referred to as a clipping algorithm or simply clipping.

The region against which an object is to be clipped is called a clip window.

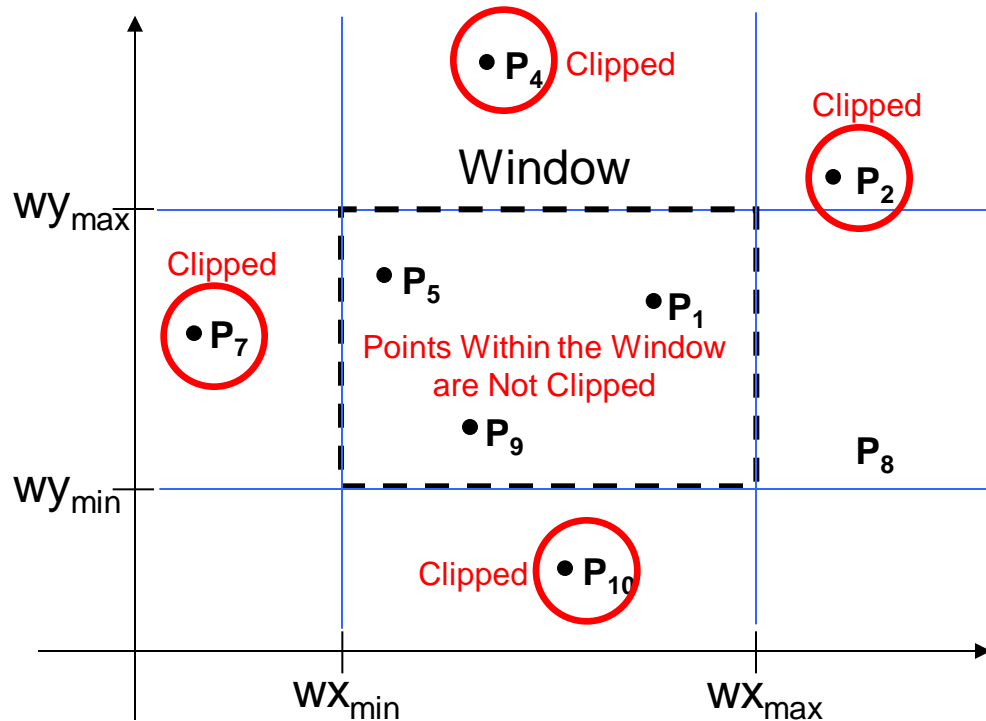
For the image below consider which lines and points should be kept and which ones should be clipped



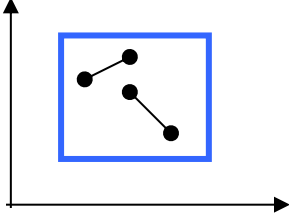
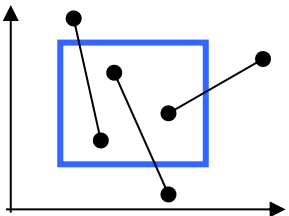
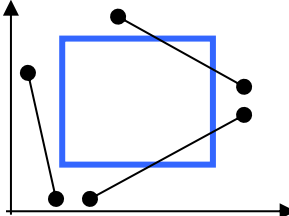
Easy - a point (x,y) is not clipped if:

$$wx_{min} \leq x \leq wx_{max} \text{ AND } wy_{min} \leq y \leq wy_{max}$$

otherwise it is clipped



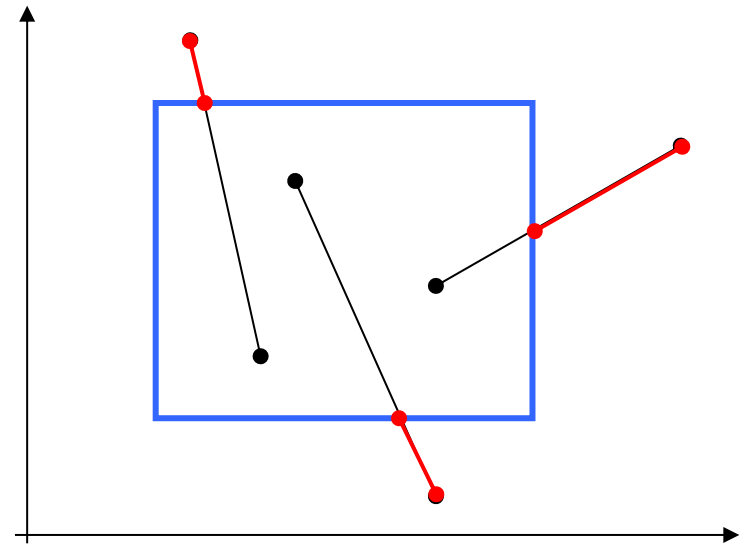
Harder - examine the end-points of each line to see if they are in the window or not

Situation	Solution	Example
Both end-points inside the window	Don't clip	 A 2D coordinate system with x and y axes. A blue square window is drawn. A line segment with two black dots at its endpoints is entirely contained within the square window.
One end-point inside the window, one outside	Must clip	 A 2D coordinate system with x and y axes. A blue square window is drawn. A line segment with two black dots at its endpoints is shown. One dot is inside the square window, and the other dot is outside the window to the right.
Both end-points outside the window	Don't know!	 A 2D coordinate system with x and y axes. A blue square window is drawn. A line segment with two black dots at its endpoints is shown. Both dots are outside the square window, one to the left and one to the right.

Brute Force Line Clipping

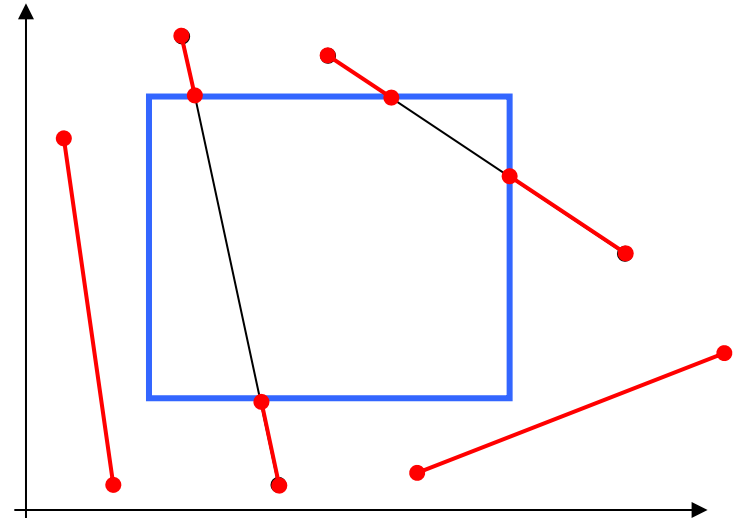
Brute force line clipping can be performed as follows:

- Don't clip lines with both end-points within the window
- For lines with one end-point inside the window and one end-point outside, calculate the intersection point (using the equation of the line) and clip from this point out



Brute Force Line Clipping (cont...)

- For lines with both endpoints outside the window test the line for intersection with all of the window boundaries, and clip appropriately



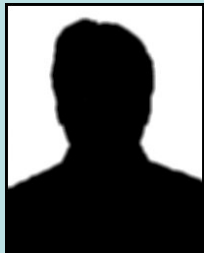
However, calculating line intersections is computationally expensive

Because a scene can contain so many lines, the brute force approach to clipping is much too slow

Cohen-Sutherland Clipping Algorithm

An efficient line clipping algorithm

The key advantage of the algorithm is that it vastly reduces the number of line intersections that must be calculated



Cohen is something of a mystery – can anybody find out who he was?



Dr. Ivan E. Sutherland co-developed the Cohen-Sutherland clipping algorithm. Sutherland is a graphics giant and includes amongst his achievements the invention of the head mounted display.

Cohen-Sutherland Clipping Algorithm

- One of the earliest algorithms with many variations in use.
- Processing time reduced by performing more test before proceeding to the intersection calculation.
- Initially, every line endpoint is assigned a four digit binary value called a region code, and each bit is used to indicate whether the point is inside or outside one of the clipping-window boundaries.

Cohen-Sutherland Clipping Algorithm

- We can reference the window edges in any order, and here is one possibility.

4	3	2	1
Top	Bottom	Right	Left

- For this ordering, (bit 1) references the left boundary, and (bit 4) references the top one.
- A value of 1 (true) in any bit position indicate that the endpoint is outside of that border.
- A value of 0 (false) indicates that the endpoint is inside or on that border.

Cohen-Sutherland: World Division

- The four window borders create nine regions
- The Figure below lists the value for the binary code in each of these regions.

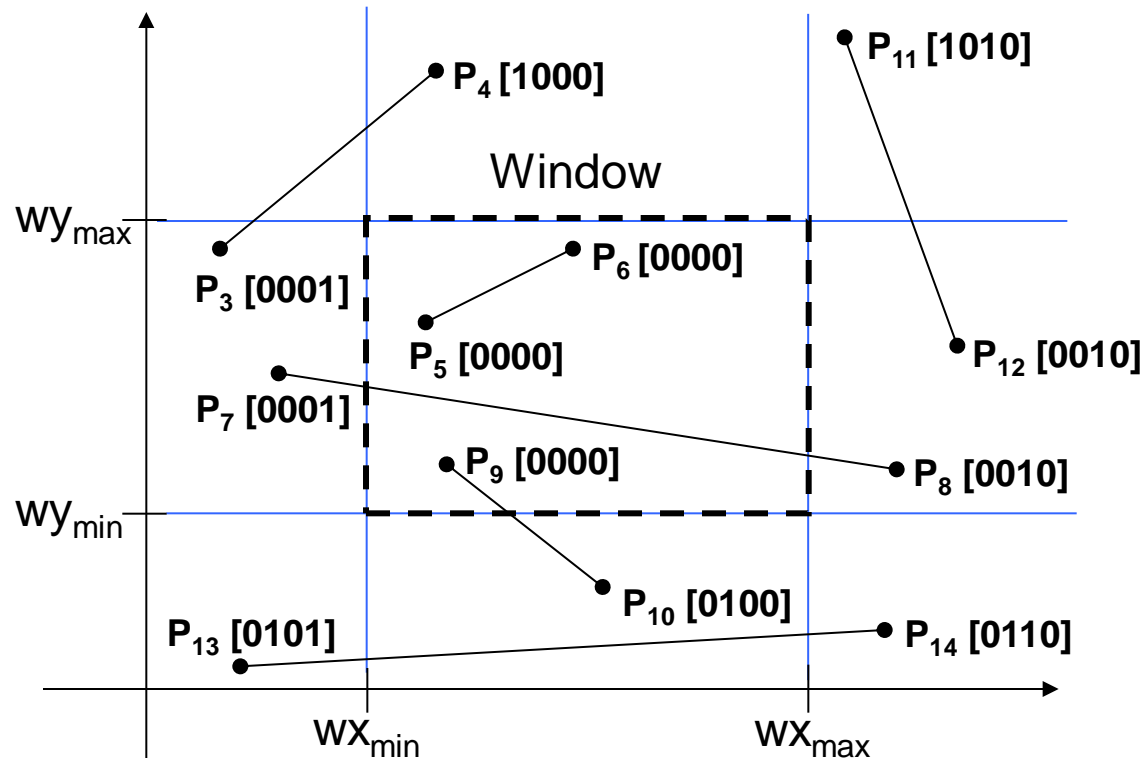
Thus, an endpoint that is below and to the left of the clipping window is assigned the region (0101).

The region code for any endpoint inside the clipping window is (0000).

1001	1000	1010
0001	0000 Window	0010
0101	0100	0110

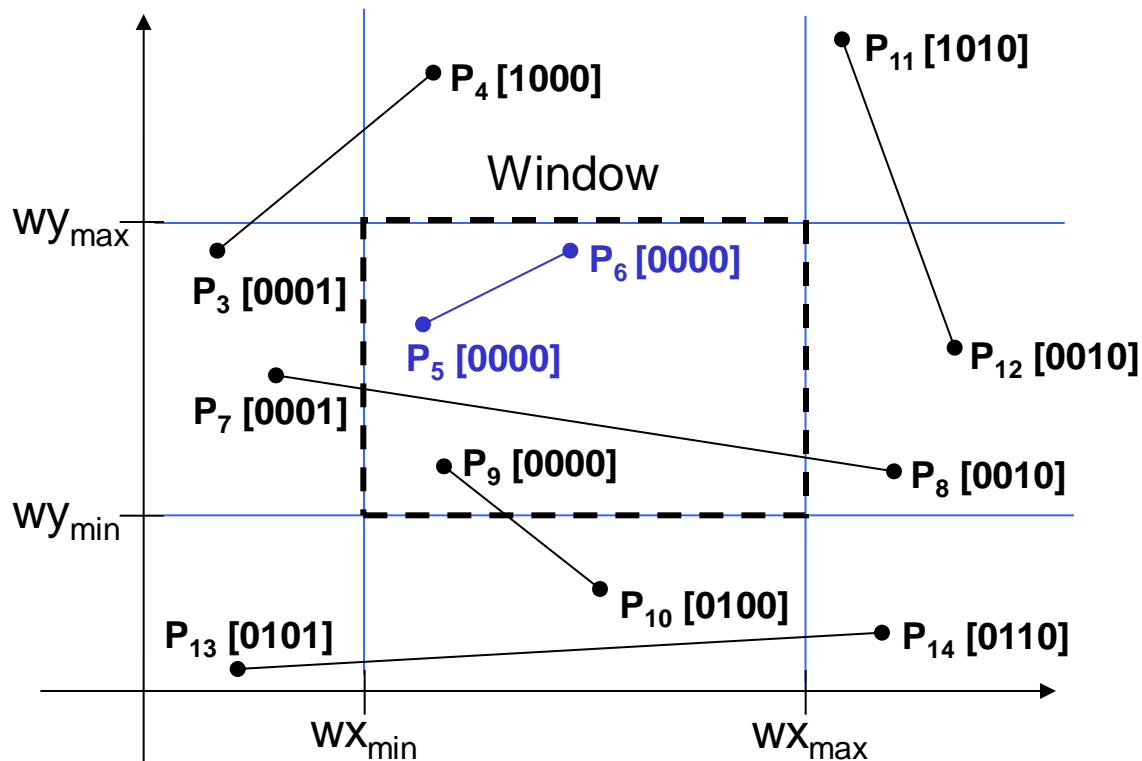
Cohen-Sutherland: Labelling

Every end-point is labelled with the appropriate region code



Cohen-Sutherland: Lines In The Window

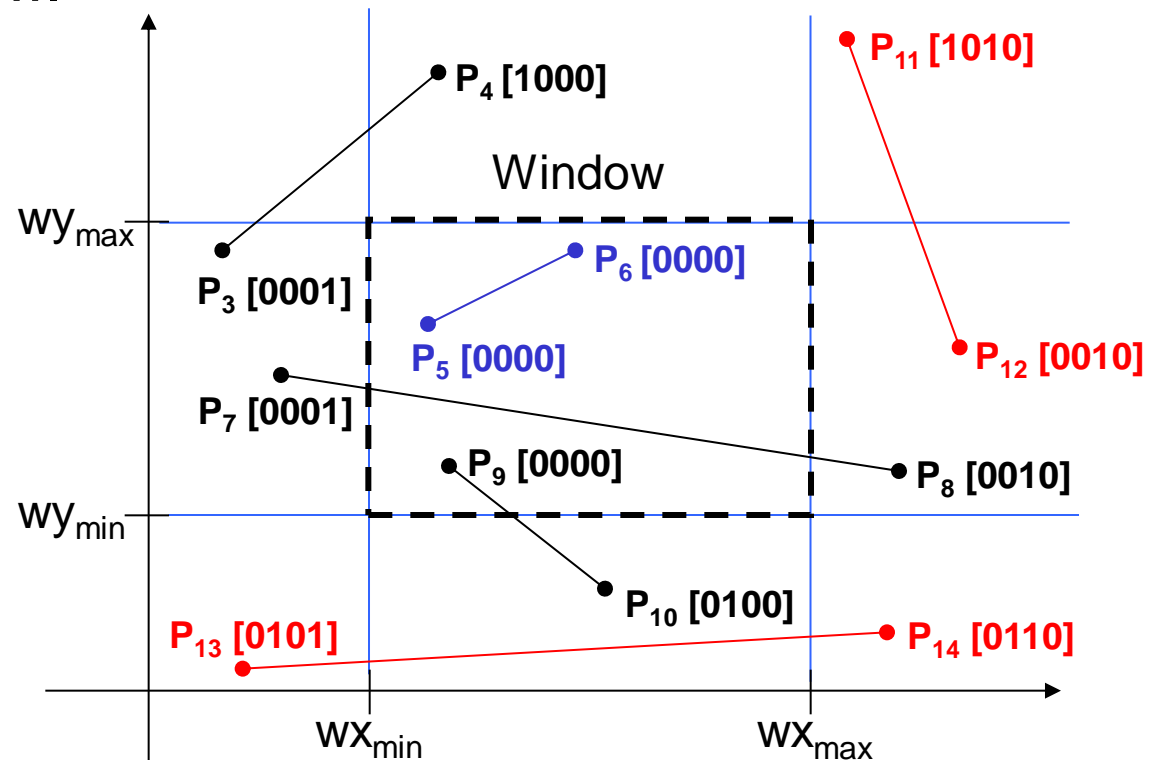
Lines completely contained within the window boundaries have region code [0000] for both end-points so are not clipped



Cohen-Sutherland: Lines Outside The Window

Any lines with 1 in the same bit position for both end-points is completely outside and must be clipped.

For example a line with 1010 code for one endpoint and 0010 for the other (line P11, P12) is completely to the right of the clipping window.



Cohen-Sutherland: Inside/Outside Lines

- We can perform inside/outside test for lines using logical operators.
- When the **or** operation between two endpoint codes is false (0000), the line is inside the clipping window, and we save it.
- When the **and** operation between two endpoint codes is true (not 0000), the line is completely outside the clipping window, and we can eliminate it.

Cohen-Sutherland: Other Lines

Lines that cannot be identified as completely inside or outside the window may or may not cross the window interior

These lines are processed as follows:

- Compare an end-point outside the window to a boundary (choose any order in which to consider boundaries e.g. left, right, bottom, top) and determine how much can be discarded
- If the remainder of the line is entirely inside or outside the window, retain it or clip it respectively

Cohen-Sutherland: Other Lines (cont...)

- Otherwise, compare the remainder of the line against the other window boundaries
- Continue until the line is either discarded or a segment inside the window is found

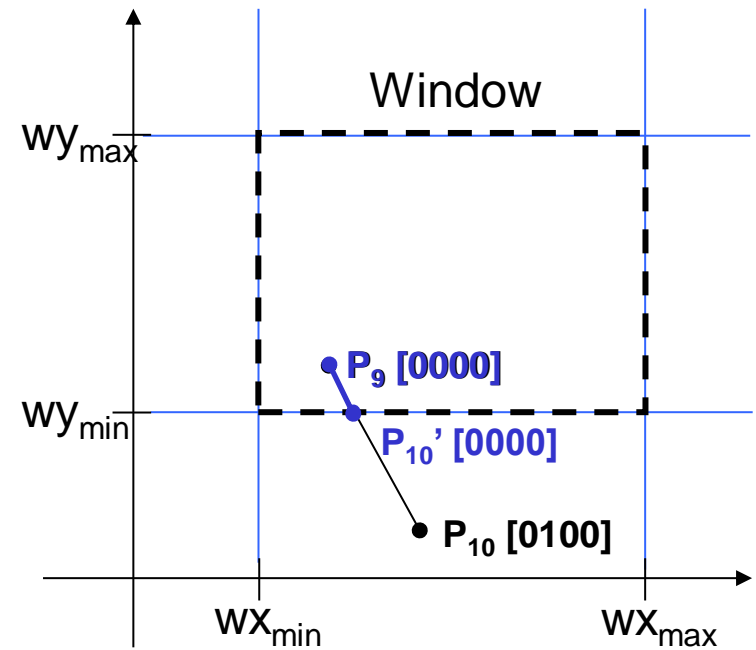
We can use the region codes to determine which window boundaries should be considered for intersection

- To check if a line crosses a particular boundary we compare the appropriate bits in the region codes of its end-points
- If one of these is a 1 and the other is a 0 then the line crosses the boundary

Cohen-Sutherland Examples

Consider the line P_9 to P_{10} below

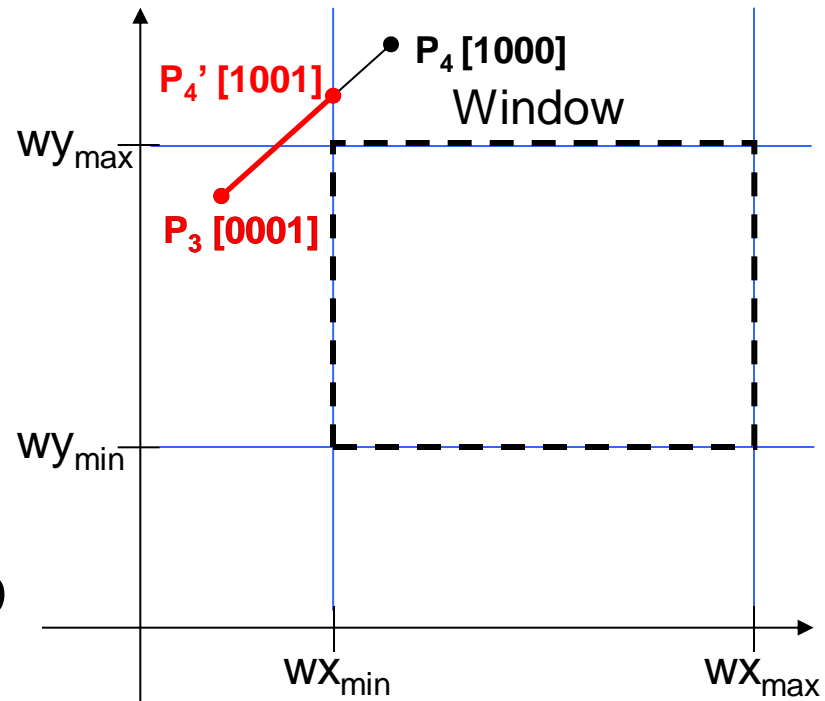
- Start at P_{10}
- From the region codes of the two end-points we know the line doesn't cross the left or right boundary
- Calculate the intersection of the line with the bottom boundary to generate point P_{10}'
- The line P_9 to P_{10}' is completely inside the window so is retained



Cohen-Sutherland Examples (cont...)

Consider the line P_3 to P_4 below

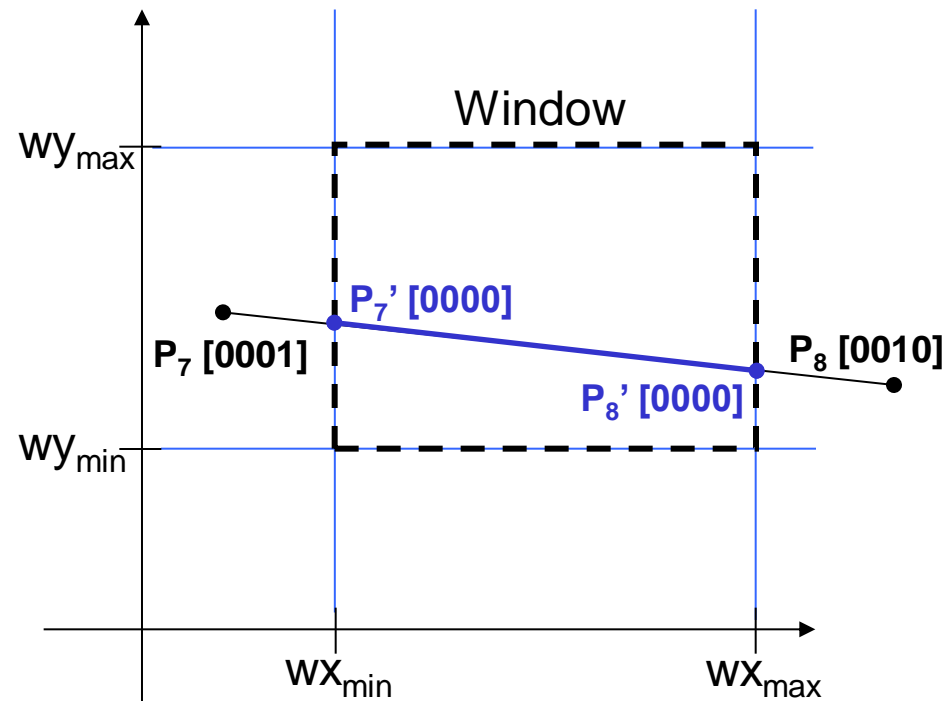
- Start at P_4
- From the region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_4'
- The line P_3 to P_4' is completely outside the window so is clipped



Cohen-Sutherland Examples (cont...)

Consider the line P_7 to P_8 below

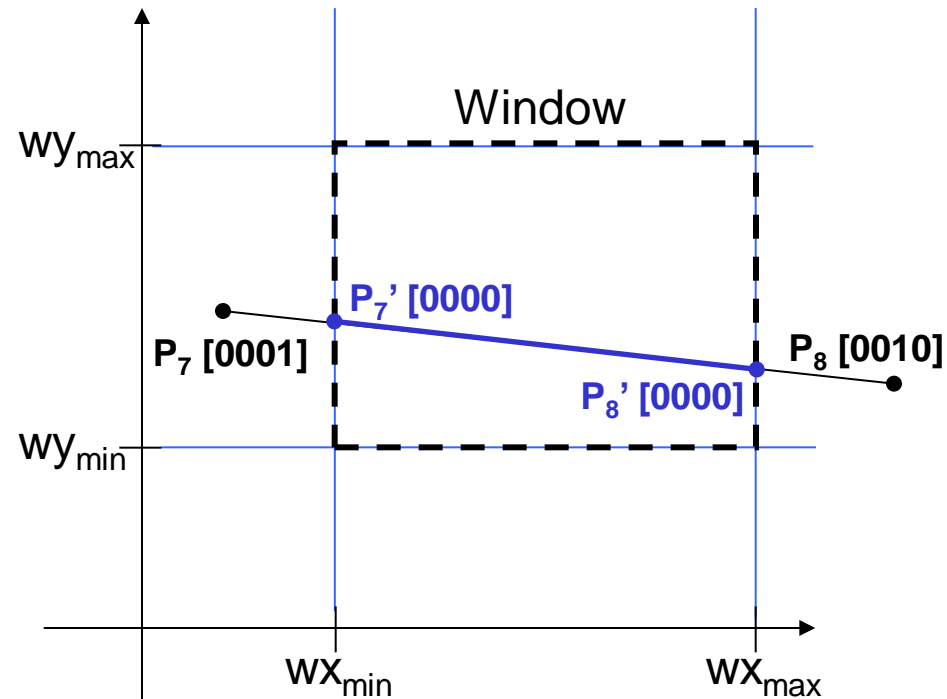
- Start at P_7
- From the two region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_7'



Cohen-Sutherland Examples (cont...)

Consider the line P_7' to P_8

- Start at P_8
- Calculate the intersection with the right boundary to generate P_8'
- P_7' to P_8' is inside the window so is retained

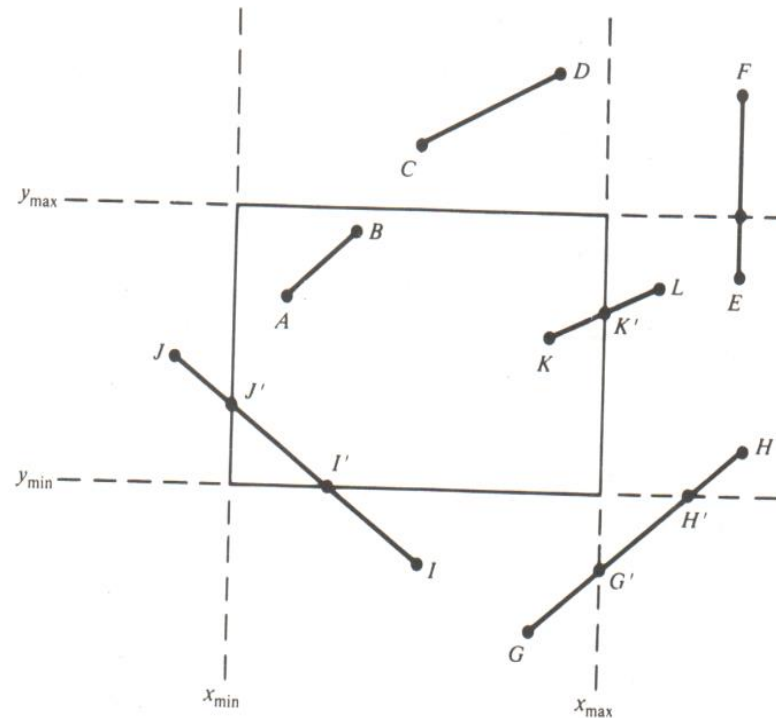


Cohen-Sutherland Examples (cont...)

Q.1: Explain Sutherland-Cohen algorithm giving all possible cases of line clipping with the help of diagram?

Cohen-Sutherland Examples (cont...)

Q.2: Explain Cohen-Sutherland algorithm and list the clipping categories for the line segments given in the picture.



Cohen-Sutherland Examples (cont...)

Q.3: Let R be the rectangular window whose lower left-hand corner is at L(-3, 1) and upper right-hand corner is at R(2, 6). Find the endpoint codes for the following lines using Sutherland-Cohen algorithm and write the clipping category for line AB, CD, EF, GH, IJ:

A(-4, 3) B(-1, 9)

C(-2, 5) D(4, 8)

E(-2, 3) F(1, 2)

G(-1, -2) H(3, 3)

I(-4, 7) J(-2, 10)

Cohen-Sutherland Examples (cont...)

The endpoint codes for point (x, y) are set according to the scheme

$$\text{Bit 1} = \text{sign}(y - y_{\max}) = \text{sign}(y - 6)$$

$$\text{Bit 3} = \text{sign}(x - x_{\max}) = \text{sign}(x - 2)$$

$$\text{Bit 2} = \text{sign}(y_{\min} - y) = \text{sign}(1 - y)$$

$$\text{Bit 4} = \text{sign}(x_{\min} - x) = \text{sign}(-3 - x)$$

Here

$$\text{Sign } a = \begin{cases} 1 & \text{if } a \text{ is positive or zero} \\ 0 & \text{otherwise} \end{cases}$$

So

$$A(-4, 2) \rightarrow 0001$$

$$F(1, 2) \rightarrow 0000$$

$$B(-1, 7) \rightarrow 1000$$

$$G(1, -2) \rightarrow 0100$$

$$C(-1, 5) \rightarrow 0000$$

$$H(3, 3) \rightarrow 0010$$

$$D(3, 8) \rightarrow 1010$$

$$I(-4, 7) \rightarrow 1001$$

$$E(-2, 3) \rightarrow 0000$$

$$J(-2, 10) \rightarrow 1000$$

Cohen-Sutherland Examples (cont...)

Category 1 (visible): EF since both endpoint codes are 0000

Category 2 (not visible): IJ since $(1001) \text{ AND } (1000) = 1000$ (which is not 0000)

Category 3 (candidates for clipping): AB, CD and GH (since logic AND of line end points equal 0000)

Calculating Line Intersections

Intersection points with the window boundaries are calculated using the line-equation parameters

- Consider a line with the end-points (x_1, y_1) and (x_2, y_2)
- The y-coordinate of an intersection with a vertical window boundary can be calculated using:

$$y = y_1 + m (x_{boundary} - x_1)$$

where $x_{boundary}$ can be set to either wx_{min} or wx_{max}

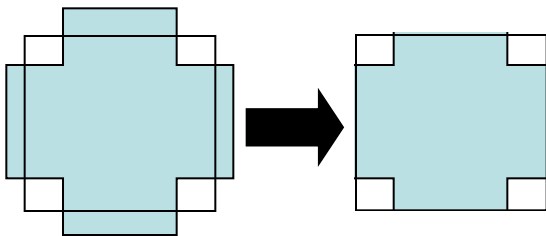
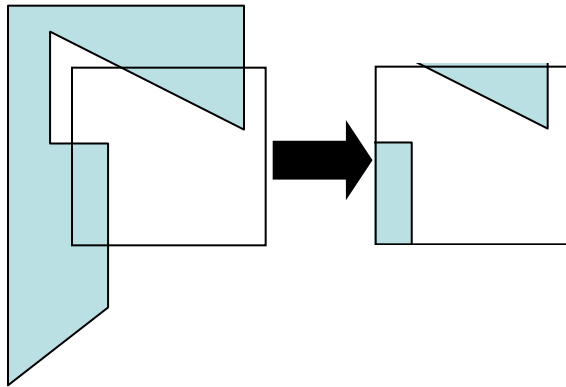
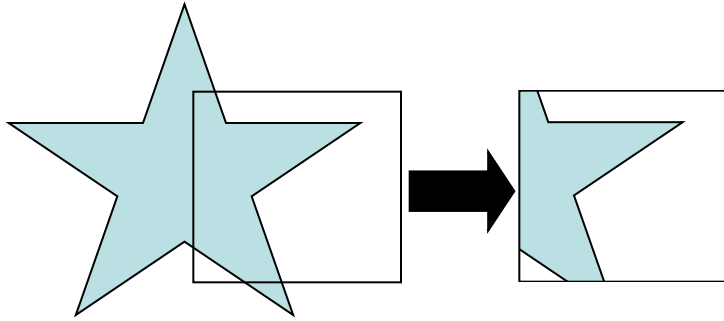
Calculating Line Intersections (cont...)

- The x-coordinate of an intersection with a horizontal window boundary can be calculated using:

$$x = x_1 + (y_{\text{boundary}} - y_1) / m$$

where y_{boundary} can be set to either wy_{min} or wy_{max}

- m is the slope of the line in question and can be calculated as $m = (y_2 - y_1) / (x_2 - x_1)$

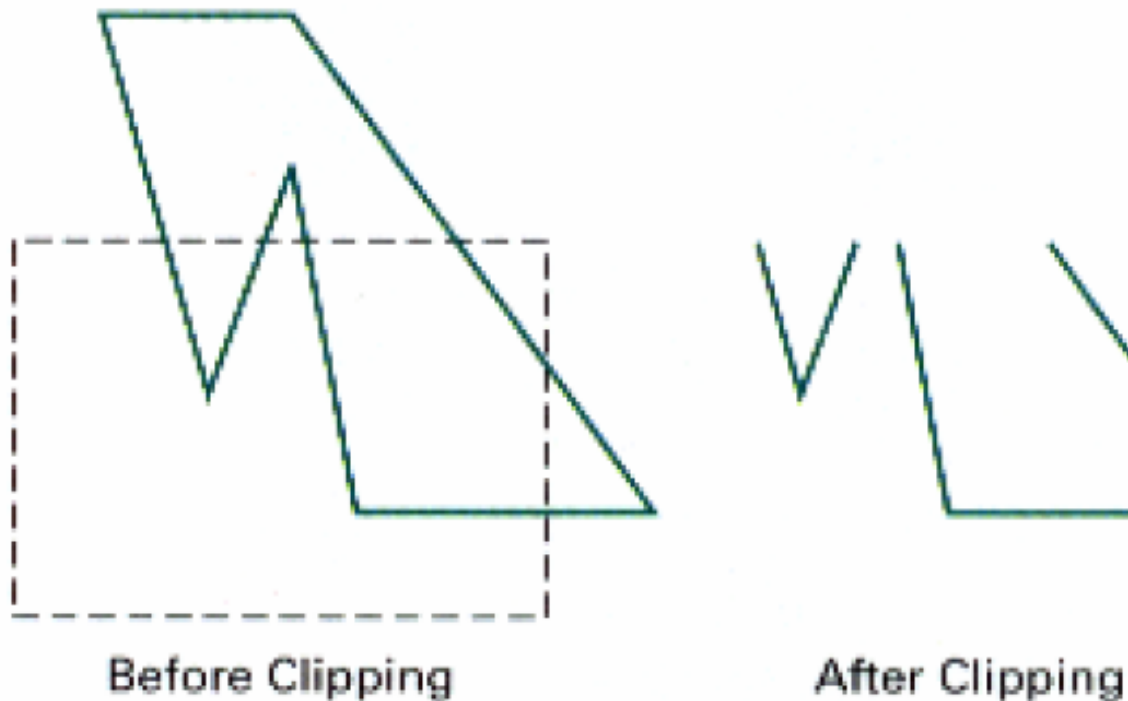


Similarly to lines, areas must be clipped to a window boundary

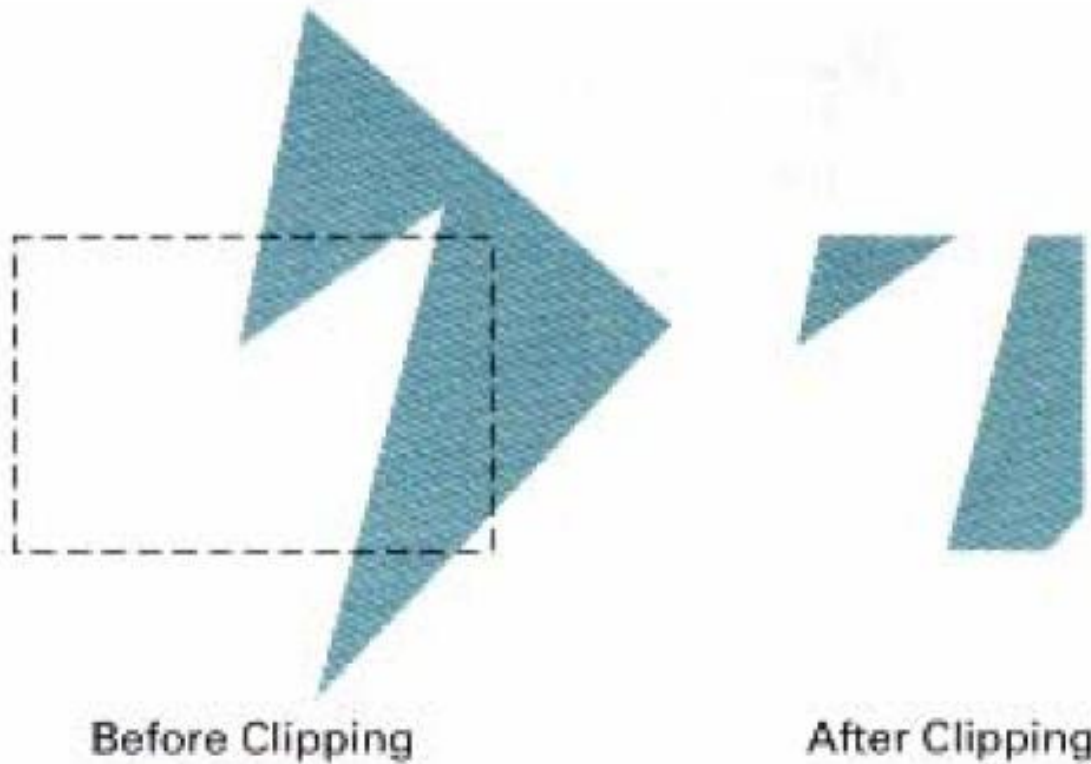
Consideration must be taken as to which portions of the area must be clipped

To clip a polygon:

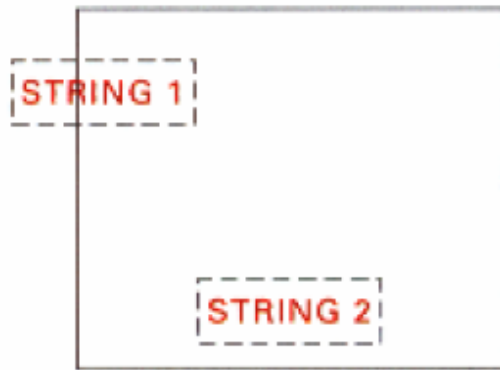
- Modify the line-clipping procedures.
- Polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments as in the following figure:



What really required is a bounded area after clipping as in the following figure:



All-or-none String-clipping



Before Clipping

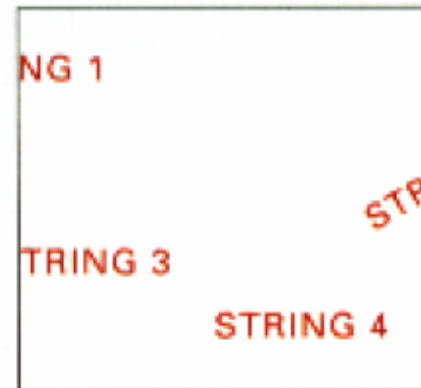


After Clipping

All-or-none Character-clipping

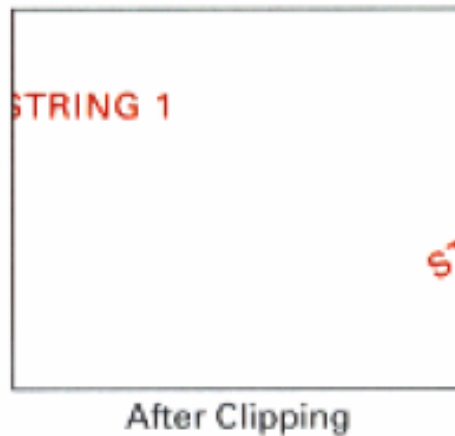
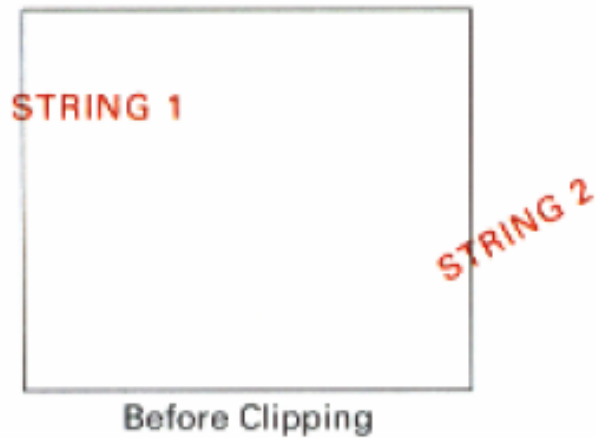


Before Clipping



After Clipping

Clipping on components of individual Characters



Areas with curved boundaries can be clipped with methods similar to those discussed in the previous sections. Curve-clipping procedures will involve nonlinear equations, however, and this requires more processing than for objects with linear boundaries.

Example

