BCA
Fifth Semester
Computer Graphics and Animation
Unit: #3

Prepared by :Narayan Dhamala

Two Dimensional Viewing

- A world coordinate area selected for display is called 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 while the viewport defines where it is to be displayed.
- The windows and viewports are rectangles in standard position, with the rectangle edges parallel to the coordinate axes.
- The mapping of a part of a world-coordinate scene to device coordinate is referred to as a viewing transformation.
- The two dimensional viewing transformation is sometimes referred as windows to viewport transformations or the windowing transformation.

Windows to viewport coordinate Transformation

- The windows to viewport coordinate transformation defines the mechanism for displaying views of a picture on an output device.

Figure 6-5
A point at position (xw, yw) in a designated window is mapped to viewport coordinates (xv, yv) so that relative positions in the two areas are the same.

Windows to viewport coordinate Transformation Contd……

- An object description is transferred to device coordinate. This can be done by using a transformation that maintains the same relative placement of an object.
- If a coordinate position is at the center of the window, it will be displayed at the center of the viewport.
- A point at (xw, yw) in the window is mapped into position (xv, yv) in the associated viewport.
- To place the same relative placement in the viewport as in the window we require that

$$\frac{xv - xvmin}{xvmax - xvmin} = \frac{xw - xwmin}{xwmax - xwmin}$$

or, $$xv = xvmin + \frac{xvmax - xvmin}{xwmax - xwmin}(xw - xwmin)$$

Windows to viewport coordinate Transformation Contd……

or, $xv = xvmin + Sx(xw - xwmin)$

Where

$$\text{Scaling factor } (Sx) = \frac{xvmax - xvmin}{xwmax - xwmin}$$

Similarly,

$$yv = yvmin + Sy(yw - ywmin)$$

Where

$$\text{Scaling factor } (Sy) = \frac{yvmax - yvmin}{ywmax - ywmin}$$

Windows to viewport coordinate Transformation Contd……

Q.N # A window having lower left hand corner (0,0) and upper right hand corner (4,4) is mapped to the viewport having lower left hand corner (0,0) and upper right hand corner (2,2). If we place the window at position (2,4) then at what position, we have to place the viewport to maintain the same relative placement as in the window?

Clipping

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

OR

- The process which divides the given picture into two parts : visible and Invisible and allows to discard the invisible part is known as clipping.
- For clipping we need reference window called as clipping window.
- The region against which an object is to clip is called a clip window.
- Clip window can be general polygon or it can be curved boundary.

Fig. Window

Application of Clipping

- It can be used for displaying particular part of the picture on display screen.
- Identifying visible surface in 3D views.
- Antialiasing line segments or object boundaries
- Creating objects using solid-modeling procedures.
- Displaying multiple windows on same screen.
- Drawing and painting.

Point Clipping

- In point clipping we eliminate those points which are outside the clipping window and draw points which are inside the clipping window.
- Here we consider clipping window is rectangular boundary with edge ($x_{wmin}, x_{wmax}, y_{wmin}, y_{wmax}$).
- So for finding wether given point is inside or outside the clipping window we use following inequality:

  $$x_{wmin} \leq x \leq x_{wamx}$$

  $$y_{wmin} \leq y \leq y_{wamx}$$
- If above both inequality is satisfied then the point is inside otherwise the point is outside the clipping window.

Line Clipping

A line clipping procedure involves several parts:

- First, we can test a given line segment to determine whether it lies completely inside the clipping window.
- If it does not, we try to determine whether it lies completely outside the window.
- Finally, if we cannot identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries. We process lines through the "inside-outside" tests by checking the line endpoints.

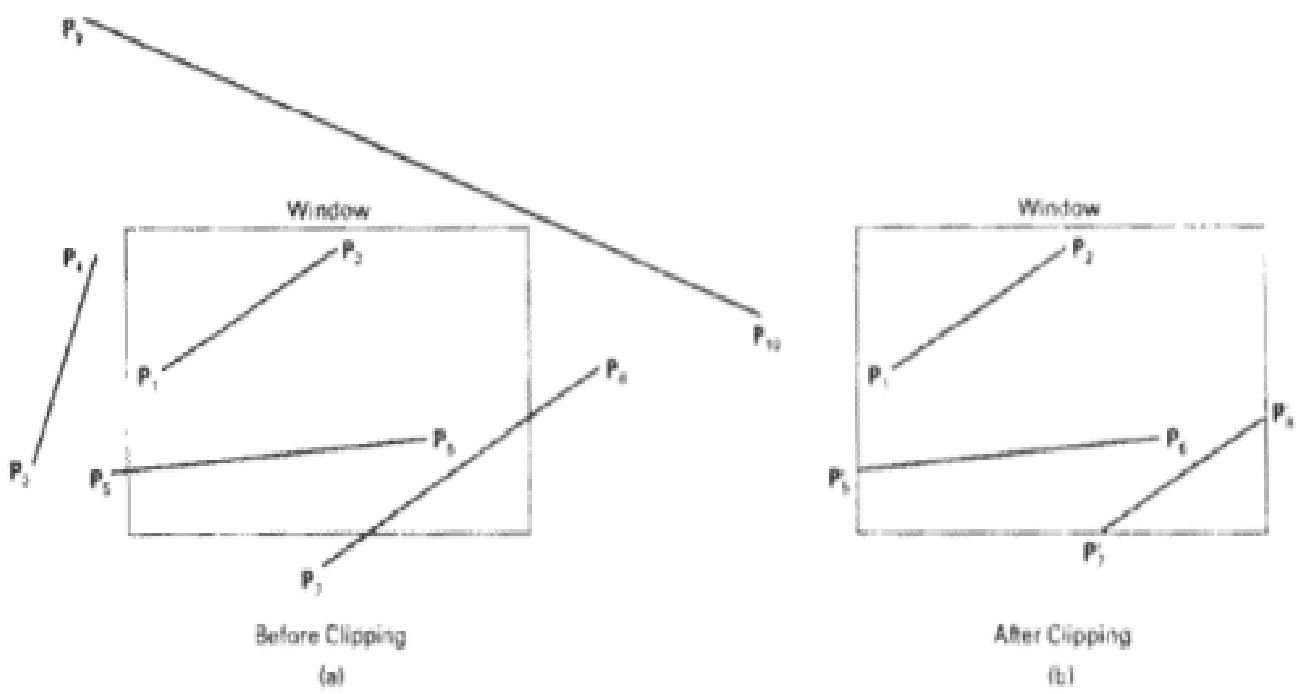


Fig: Line-clipping against a rectangular window

For a line segment with endpoints (x_1, y_1) and (x_2, y_2) and one or both endpoints outside the clipping rectangle, the parametric representation

$$x = x_1 + u(x_2 - x_1)$$
$$y = y_1 + u(y_2 - y_1), \quad 0 \leq u \leq 1$$

can be used to determine values of parameter **u** for intersections with the clipping boundary coordinates. If the value of **u** for an intersection with a rectangle boundary edge is outside the range **0** to **1**, the line does not enter the interior of the window at the boundary. If the value of ***u*** is within the range from **0** to **1**, the line segment does indeed cross into the clipping area. This method can be applied to each clipping boundary edge in turn to determine whether any part of the line segment is to be displayed. Line segments that are parallel to window edges can be handled as special cases.

Line Clipping

Discard the part of lines which lie outside the boundary of the window.

We require:

1. To identify the point of intersection of the line and window.
2. The portion in which it is to be clipped.
   The lines are divided into three categories.

a) Invisible
b) Visible
c) Partially Visible [Clipping Candidates]

To clip we give 4- bit code representation defined by

Bit 1	Bit 2	Bit 3	Bit 4
Ymax	Ymin	Xmax	Xmin

Fig. 7.5

Where, Bits take the volume either 0 or 1 and
Here, we divide the area containing the window as follows. Where, the coding is like this, Bit value = 1 if point lies outside the boundary
OR

= 0 if point lies inside the boundary.
(Xmin ≤ X ≤ Xmax and Ymin ≤ Y ≤Ymax)

Line Clipping Contd....

Bit 1 tells you the position of the point related to Y=Ymax

Bit 2 tells you the position of the point related to Y=Ymin

Bit 3 tells you the position of the point related to X=Xmax

Bit 4 tells you the position of the point related to X=Xmin

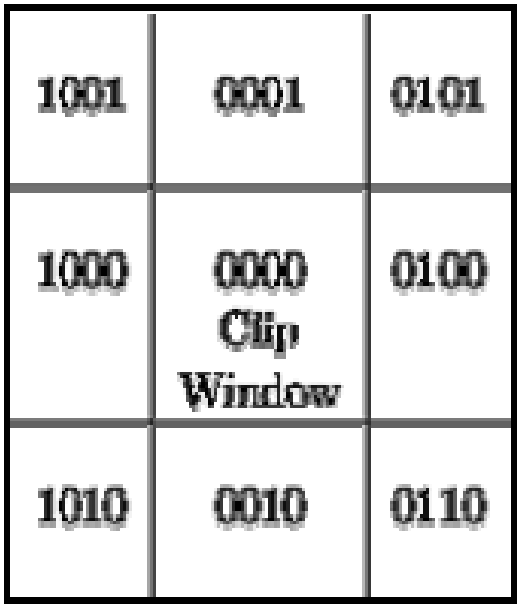


Fig. 7.6 Bit Code Representation

Rules for the visibility of the line:

1. If both the end points have bit code 0000 the line is visible.
2. If atleast one of the end point in non zero and
   a) The logical "AND"ing is 0000 then the line is Partially Visible
   b) If the logical "AND"ing isnon-zero then line is Not Visible.

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)

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

$$\text{Bit } 1 = \text{sign}(y - y_{max}) = \text{sign}(y - 6) \qquad \text{Bit } 3 = \text{sign}(x - x_{max}) = \text{sign}(x - 2)$$

$$\text{Bit } 2 = \text{sign}(y_{min} - y) = \text{sign}(1 - y) \qquad \text{Bit } 4 = \text{sign}(x_{min} - x) = \text{sign}(-3 - x)$$

Here

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

So

$A(-4, 2) \rightarrow 0001$ $\qquad$ $F(1, 2) \rightarrow 0000$

$B(-1, 7) \rightarrow 1000$ $\qquad$ $G(1, -2) \rightarrow 0100$

$C(-1, 5) \rightarrow 0000$ $\qquad$ $H(3, 3) \rightarrow 0010$

$D(3, 8) \rightarrow 1010$ $\qquad$ $I(-4, 7) \rightarrow 1001$

$E(-2, 3) \rightarrow 0000$ $\qquad$ $J(-2, 10) \rightarrow 1000$

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

Category 2 (not visible): IJ since (1001) AND (1000)=1000 (which is not 0000)

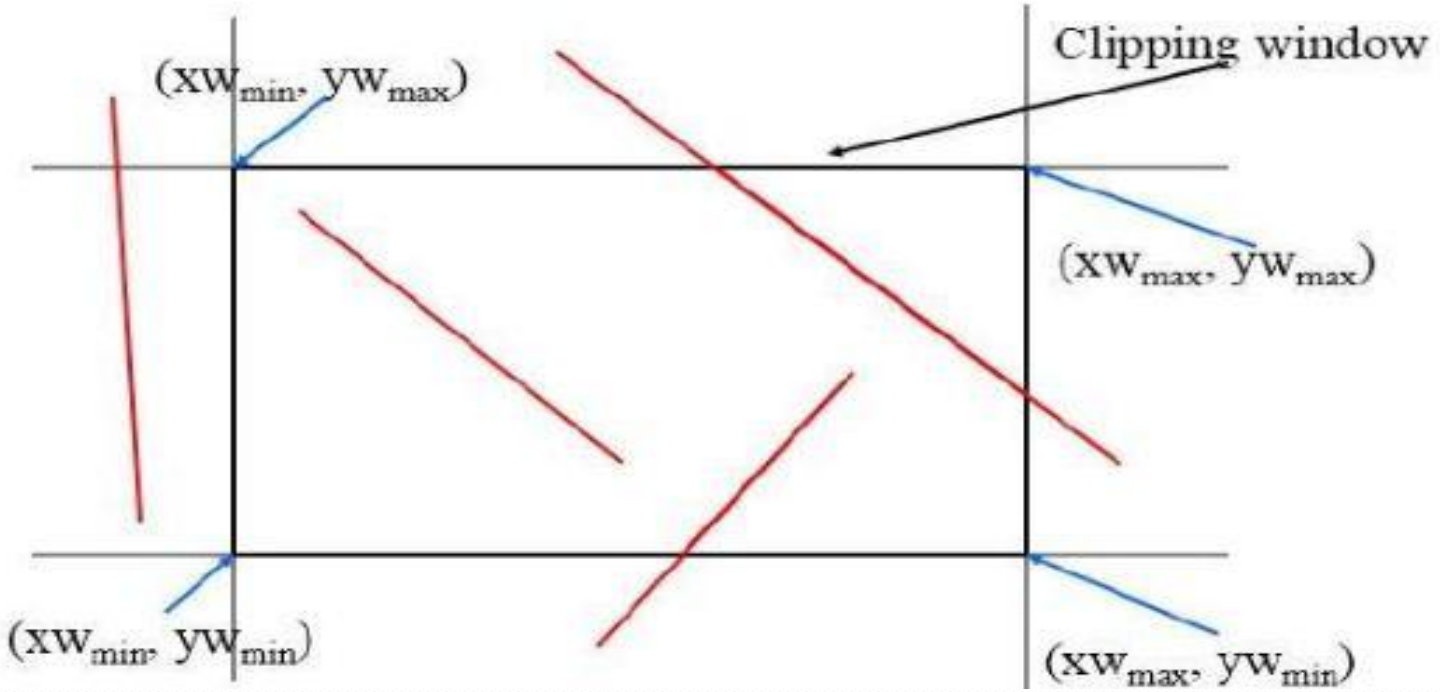
Category 3 (candidates for clipping): AB, CD and GH (since logic AND of line end points equal 0000)

Line Clipping

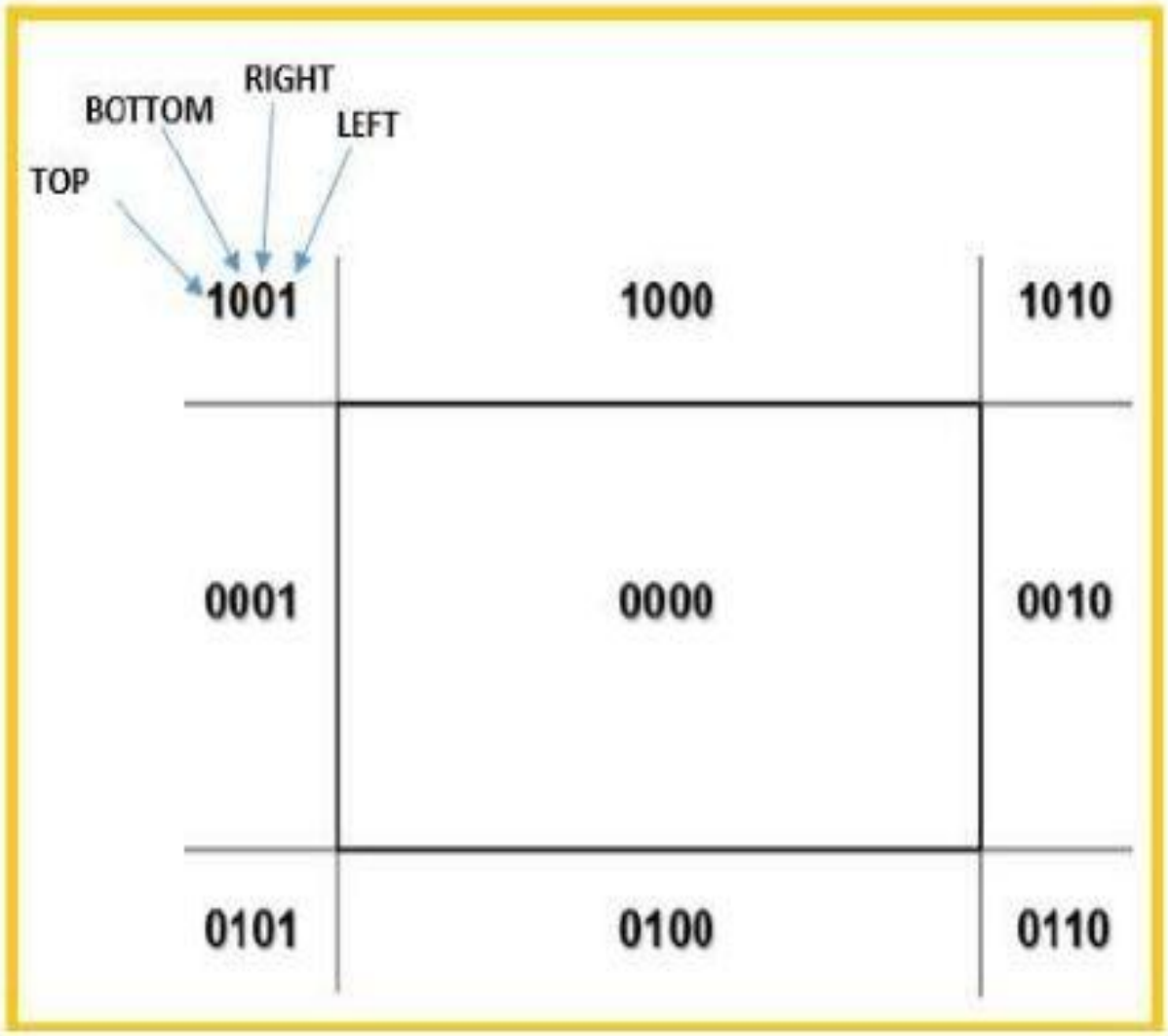
The concept of line clipping is same as point clipping. In line clipping, we will cut the portion of line which is outside of window and keep only the portion that is inside the window.

Cohen-Sutherland Line Clippings

This algorithm uses the clipping window as shown in the following figure. The minimum coordinate for the clipping region is (XW_{min}, YW_{min}) and the maximum coordinate for the clipping region is (XW_{max}, YW_{max}).



We will use 4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the **TOP** and **LEFT** bit is set to 1 because it is the **TOP-LEFT** corner.



There are 3 possibilities for the line –

- Line can be completely inside the window $Thislineshouldbeaccepted$.
- Line can be completely outside of the windov $Thislinewillbecompletelyremovedfromtheregion$.
- Line can be partially inside the windov $Wewillfindintersectionpointanddrawonlythatportionoflinethatisinsideregion$.

Algorithm

Step 1 – Assign a region code for each endpoints.

Step 2 – If both endpoints have a region code **0000** then accept this line.

Step 3 – Else, perform the logical **AND**operation for both region codes.

Step 3.1 – If the result is not **0000,** then reject the line.

Step 3.2 – Else you need clipping.

Step 3.2.1 – Choose an endpoint of the line that is outside the window.

Step 3.2.2 – Find the intersection point at the window boundary $baseonregioncode$.

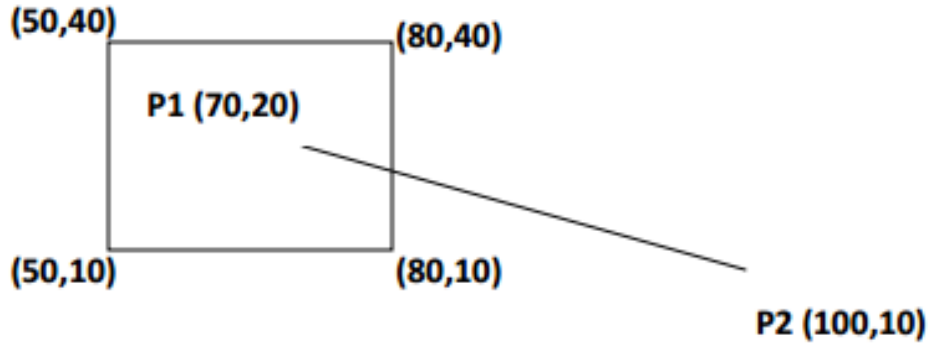
Step 3.2.3 – Replace endpoint with the intersection point and update the region code.

Step 3.2.4 – Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

Step 4 – Repeat step 1 for other lines.

Q53. Use the Cohen Sutherland algorithm to clip line P1 (70,20) and p2(100,10) against a window lower left hand corner (50,10) and upper right hand corner (80,40).

Ans:



Given , P1(70,20) and p2(100,10)
Window lower left corner = (50,10)
Window upper right corner = (80,40)

Now, we assign 4 bit binary outcode.
Point P1 is inside the window so the outcode of P1 = 0000 and the outcode for P2 = 0010.
Logical AND operation will give , 0000

$$\underline{0010}$$
$$0000$$

Slope of the line P1P2 is $m = \frac{y2 - y1}{X2 - x1} = \frac{10 - 20}{100 - 70} = \frac{-10}{30} = \frac{-1}{3}$

We, have to find intersection of line P1 P2 with right edge of window i.e P2 (x,y).
Here x=80 , we have to find the value of y.
We use the point P2(x2,y2) = P2(100,10)\

$$M = \frac{y - y2}{x - x2}$$

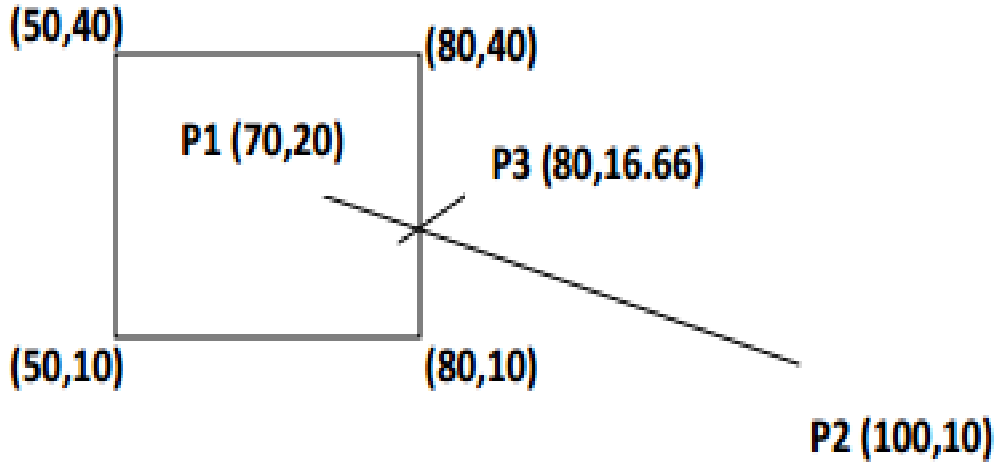
$$-1/3 = \frac{y - 10}{80 - 100}$$

$y-10 = 20 / 3$

$y=16.66$

thus, the intersection point P3 = (80, 16.66)

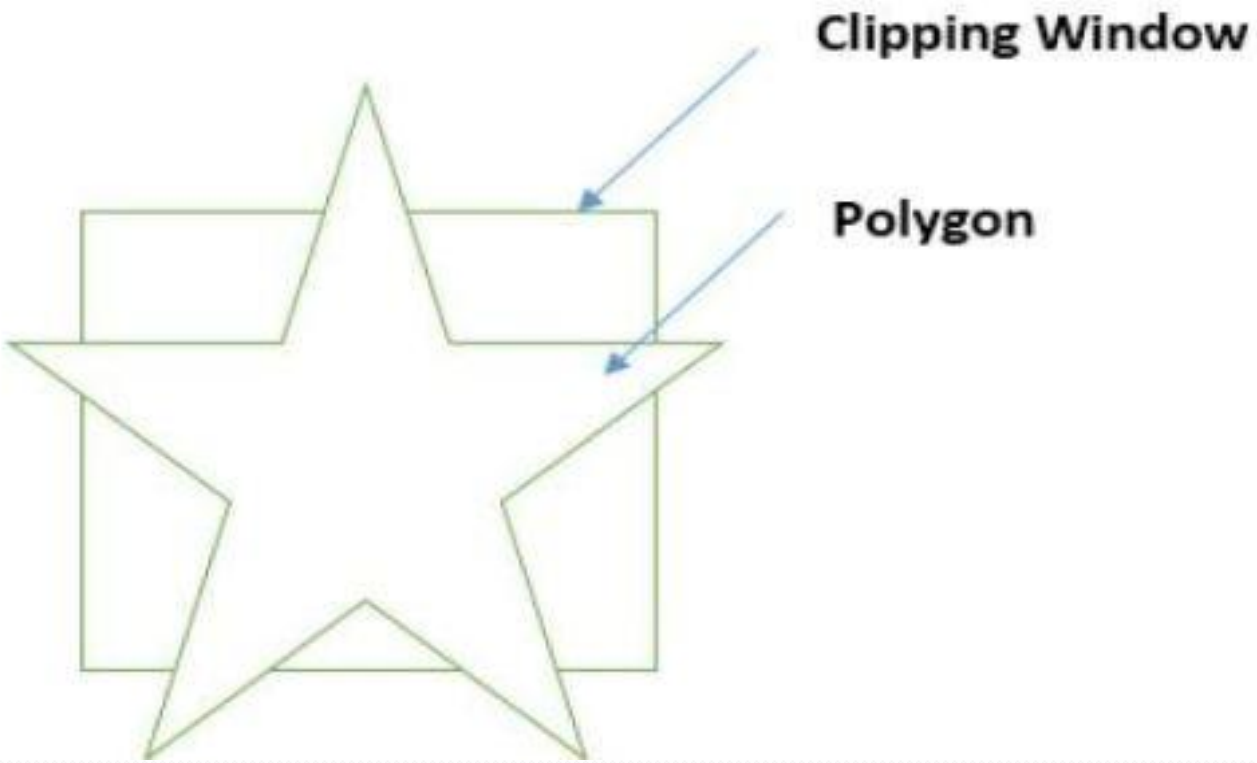
So, after clipping line P1P2 against the window, new line P1P3 with co ordinates P1(70, 20) and P3 (80, 16.66)



Polygon Clipping *SutherlandHodgmanAlgorithm*

A polygon can also be clipped by specifying the clipping window. Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping. In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.

First the polygon is clipped against the left edge of the polygon window to get new vertices of the polygon. These new vertices are used to clip the polygon against right edge, top edge, bottom edge, of the clipping window as shown in the following figure.



While processing an edge of a polygon with clipping window, an intersection point is found if edge is not completely inside clipping window and the a partial edge from the intersection point to the outside edge is clipped. The following figures show left, right, top and bottom edge clippings –

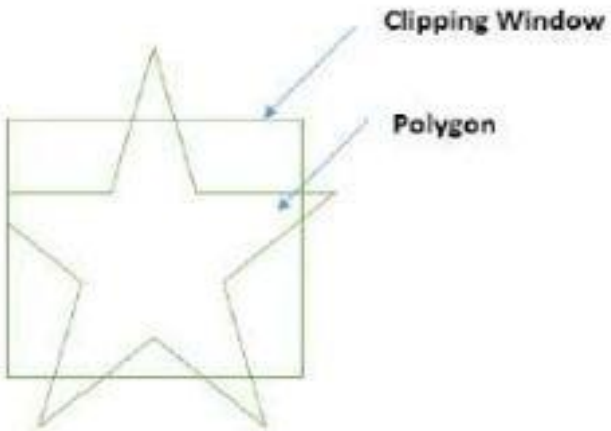


Figure: Clipping Left Edge

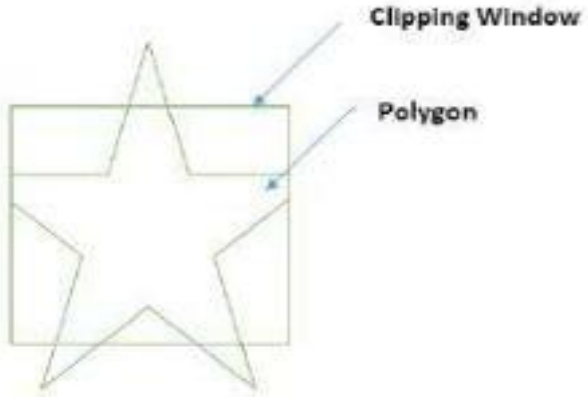


Figure: Clipping Right Edge

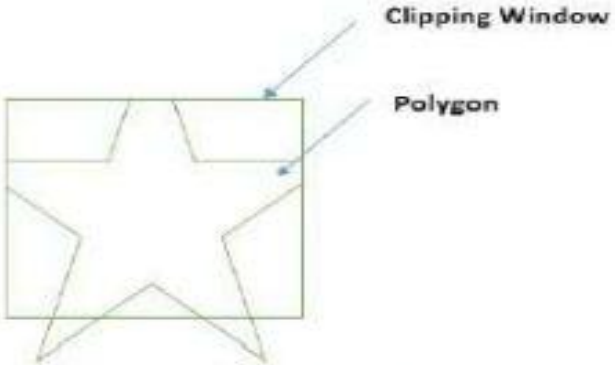


Figure: Clipping Top Edge

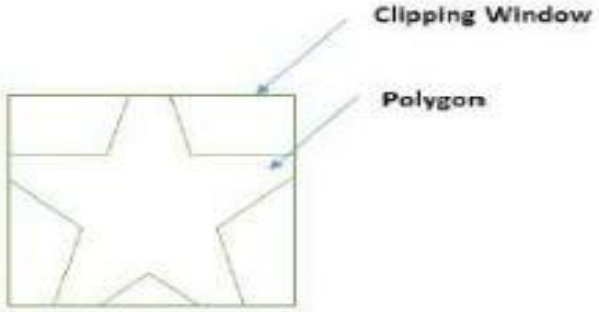


Figure: Clipping Bottom Edge

•Sutherland–Hodgeman algorithm

The Sutherland–Hodgeman algorithm is used for clipping polygons. It works by extending each line of the convex clip polygon in turn and selecting only vertices from the subject polygon that are on the visible side

Input each edge (vertex pair) successively.
Output is a new list of vertices.
Each edge goes through 4 clippers.
The rule for each edge for each clipper is:

- If first input vertex is outside, and second is inside, output the intersection and the second vertex
- If first both input vertices are inside, then just output second vertex
- If first input vertex is inside, and second is outside, output is the intersection
- If both vertices are outside, output is nothing

How To Calculate Intersections

Assume that we're clipping a polgon's edge with vertices at (x1,y1) and (x2,y2) against a clip window with vertices at (xmin, ymin) and (xmax,ymax).

The location (IX, IY) of the intersection of the edge with the left side of the window is:

i. IX = xmin
ii. IY = slope*(xmin-x1) + y1, where the slope = (y2-y1)/(x2-x1)

The location of the intersection of the edge with the right side of the window is:

i. IX = xmax
ii. IY = slope*(xmax-x1) + y1, where the slope = (y2-y1)/(x2-x1)

The intersection of the polygon's edge with the top side of the window is:

i. IX = x1 + (ymax - y1) / slope
ii. IY = ymax

Finally, the intersection of the edge with the bottom side of the window is:

i. IX = x1 + (ymin - y1) / slope
ii. IY = ymin

Some Problems With This Algorithm

1. This algorithm does not work if the clip window is not convex.
2. If the polygon is not also convex, there may be some dangling edges.

Class Assignment

CLIP POLYGON ABCDE AGAINST WINDOW PQRS. THE CO-ORDINATES OF THE POLYGON ARE A(80,200); B(220,120); C(150,100); D(100,30); E(10,120). CO-ORDINATES OF THE WINDOW ARE P(200,50); Q(50,150); R(200,150); S(50,50).

Curve Clipping

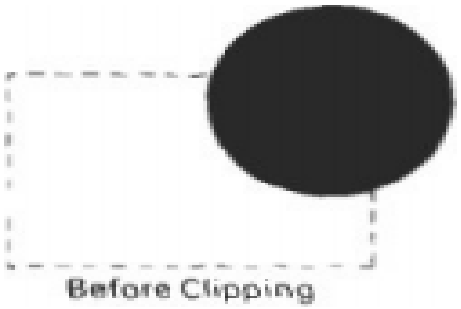
Curve-clipping procedures will involve nonlinear equations, and this requires more processing than for objects with linear boundaries. The bounding rectangle for a circle or other curved object can be used first to test for overlap with a rectangular clip window.

If the bounding rectangle for the object is completely inside the window, we save the object. If the rectangle is determined to be completely outside the window, we discard the object. In either case, there is no further computation necessary.

But if the bounding rectangle test fails, we can look for other computation-saving approaches. For a circle, we can use the coordinate extents of individual quadrants and then octants for preliminary testing before calculating curve-window intersections.

The below figure illustrates circle clipping against a rectangular window. On the first pass, we can clip the bounding rectangle of the object against the bounding rectangle of the clip region. If the two regions overlap, we will need to solve the simultaneous line-curve equations to obtain the clipping intersection points.

Clipping a filled circle



END