Computer Graphics 05101301

Prof. Hina Chokshi, Sr. Assistant Professor
Parul Institute of Computer Application - BCA

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-4

Two Dimensional Viewing

Coordinate System

World Coordinate System (Object Space)

- It is space in which the application model is defined.
- Space in which the object **geometry** is defined. Here we describe the coordinates of image to be displayed

Screen Coordinate System (Image Space)

- Space in which the image is displayed, **eg. 800x600 pixels.**
- Usually measured in pixels.
- Space in which the object's **raster image** is defined.

Interface Window (Image Subspace)

- Visual representation of the screen coordinate system for windowed displays (coordinate system moves with the interface window)

World Window (Object Subspace)

- Rectangle defining the part of the world we wish to display.

Viewing and Clipping

- The process of mapping of a part of world coordinate scene to device coordinates is known as **viewing transformation.**
- A **world coordinate area** selected for display is called as a **window.**
- A **area of a display device** to which a window is mapped as called a **view port**.

Viewing and Clipping

Window - **Windowing** is a process of selecting and enlarging the portions of a drawing or an object.

Clipping: The technique for not showing that part of the drawing in which one is not interested. This is done by means of transformation.

Majorly clipping is applied to three different entities:

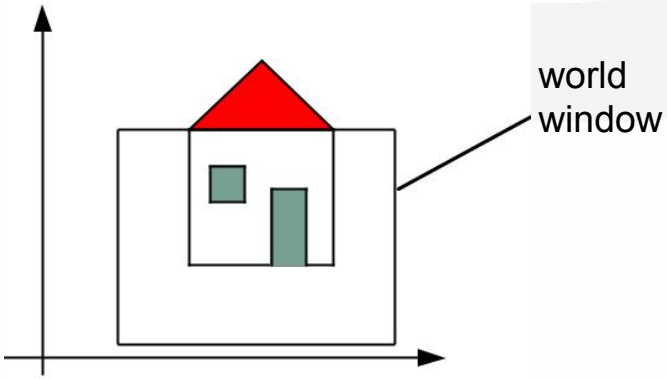
1) lines

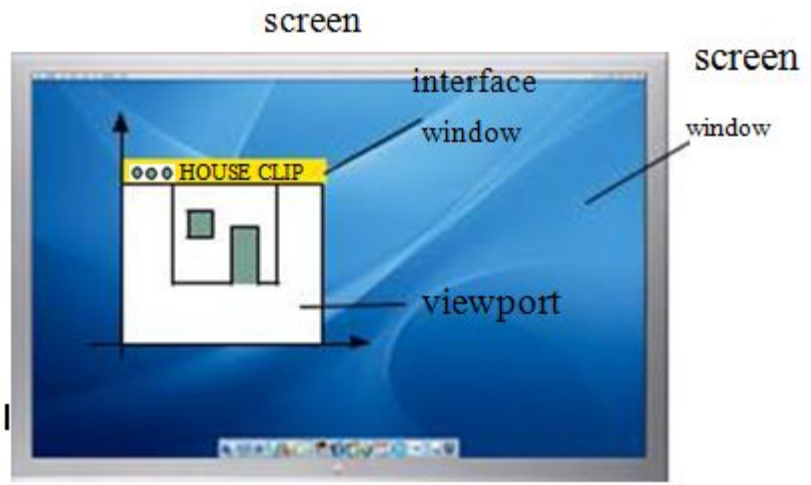
2) polygons

3) text

Window and Clipping

A window defines “WHAT” is to be viewed and the viewport defines “WHERE” it is to be displayed, “CLIPPING” means what to omit.





Window-Viewport Mapping

- **Object is selected from a big image and then the viewport window is set to its origin (transform the object to viewport).**
- Then the object has to be scaled (Size of object has to be increased as per the viewport size, called Scaling).
- Then viewport window is kept at its original position on the screen.

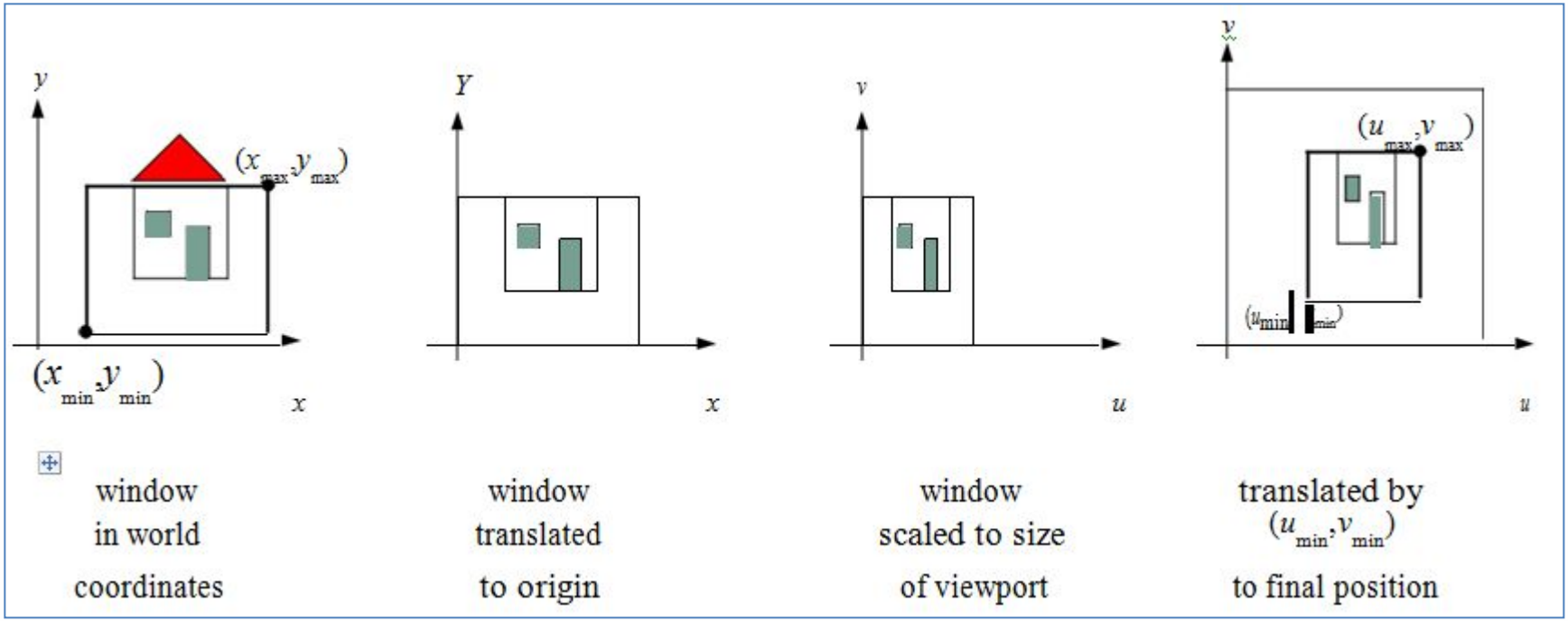
ie WCS ----------▯ NDCS ---------▯ PDCS
Infinite region) (Finite region)

- Any window is specified with 4 coordinate system namely: W**xmin,** W**xmax,** Wy**min,** Wy**max.**
- Similarly a viewport can be represented by 4 normalised coordinates namely : V**xmin,** V**xmax, Vymin,** Vy**max.**

Parul® University

DIGITAL LEARNING CONTENT

Window-Viewport Mapping



window in world coordinates	window translated to origin	window scaled to size of viewport	translated by (u_{min}, v_{min}) to final position

Types of Clipping

Clipping in is used to remove objects, lines, or line segments that are outside the viewing pane. Different types of clipping are…

- Point Clipping
- Line Clipping
- Character Clipping
- Polygon Clipping

Point Clipping

- Point clipping helps to know whether the given point (X, Y) is within the given window or not.
- Also to decides whether we will use the minimum and maximum coordinates of the window.
- The X-coordinate of the given point is inside the window, if X lies in between $Wx1 \leq X \leq Wx2$.
- Same way, Y coordinate of the given point is inside the window, if Y lies in between $Wy1 \leq Y \leq Wy2$.

Line Clipping

- In line clipping, the portion of line which is outside of window is cut and only the portion that is inside the window is kept.
- **Cohen-Sutherland Line Clippings**

  Minimum coordinate for the clipping region is **(XWmin,YWmin)(XWmin,YWmin)**
- Maximum coordinate for the clipping region is **(XWmax,YWmax)(XWmax,YWmax)**

Cohen-Sutherland Line Clippings

The Cohen–Sutherland algorithm is a computer-graphics algorithm used for **line clipping**

The algorithm divides a two-dimensional space into **9 regions**

It efficiently determines the lines and portions of lines that are visible in the central region of interest (the viewport)

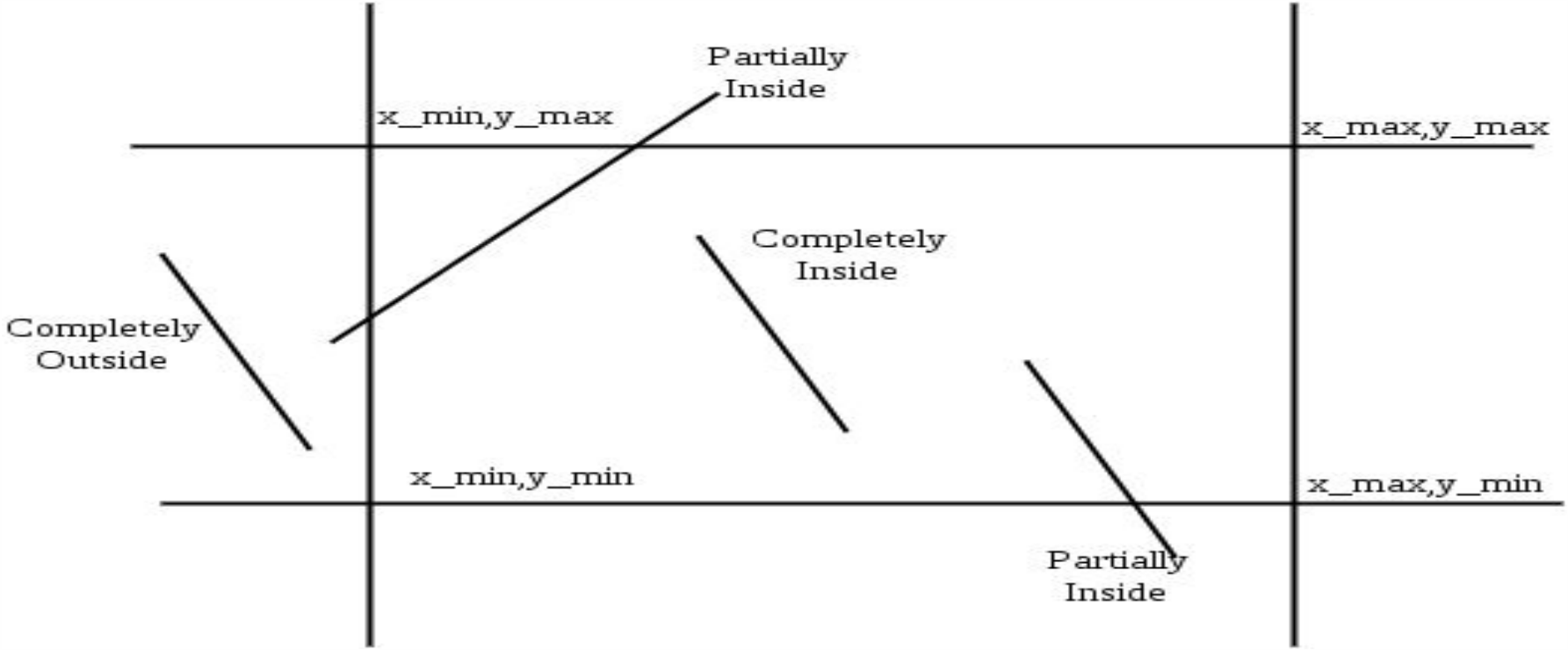
The Cohen-Sutherland algorithm uses a **divide-and-conquer** strategy.

Cohen-Sutherland Line Clippings

There are 3 possibilities for the line −

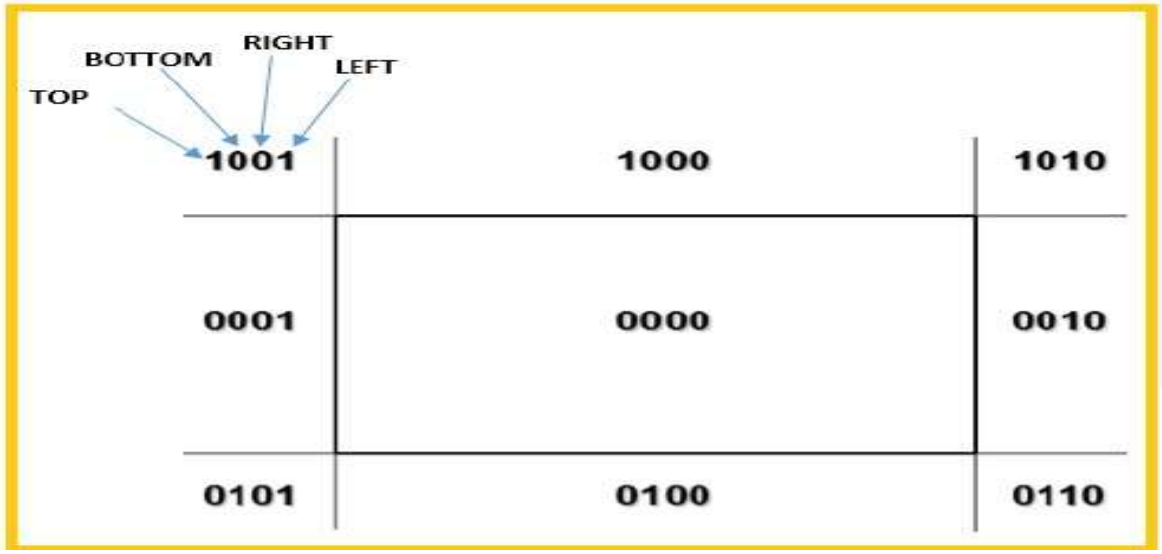
- Line can be completely inside the window (This line should be accepted).
- Line can be completely outside of the window (This line will be completely removed from the region).
- Line can be partially inside the window (We will find intersection point and draw only that portion of line that is inside region).

Cohen-Sutherland Line Clippings



Cohen-Sutherland Line Clipping Algorithm

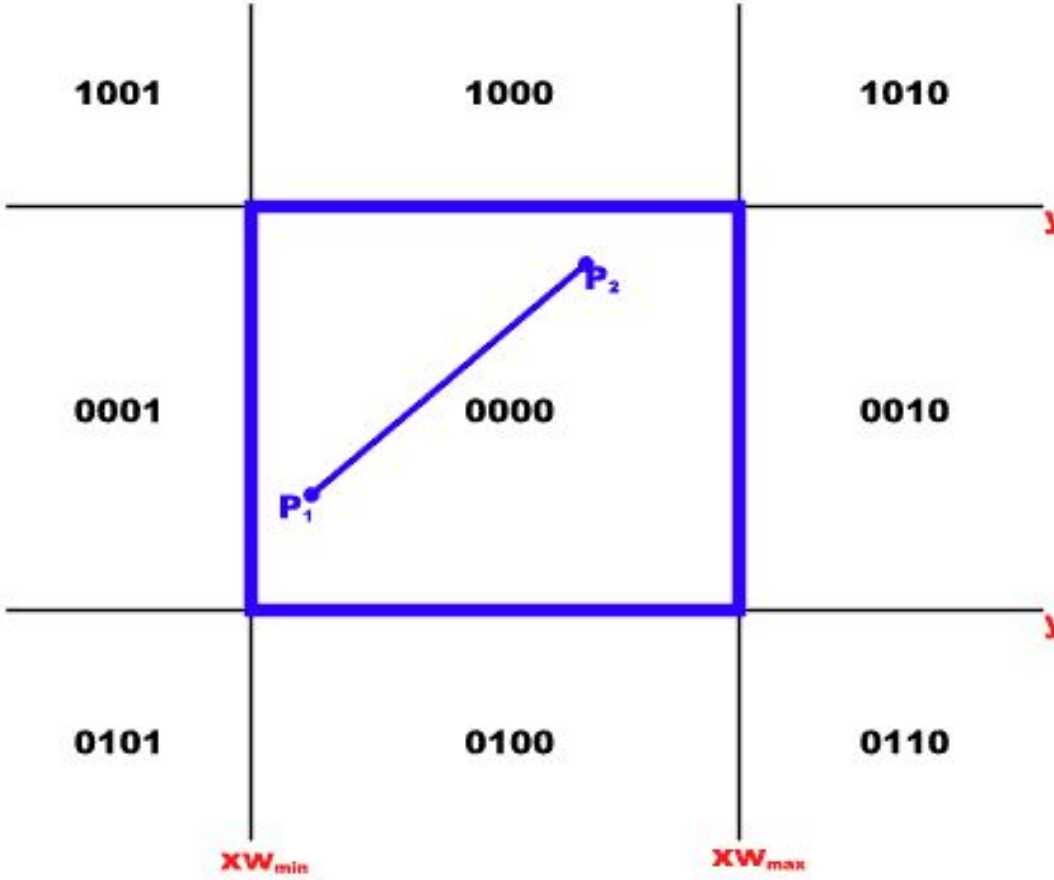
4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the **TOP** and **LEFT** bit is set to 1 because it is the **TOP-LEFT** corner.



Cohen-Sutherland Line Clipping 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 (base on region code).
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.

Cohen-Sutherland Line Clipping Algorithm



4 bit clip code:

A B R L

A = y > yw_{max}

B = y < yw_{min}

R = x > xw_{max}

L = x < xw_{min}

Calculate clip code

$P_1 = 0\ 0\ 0\ 0$

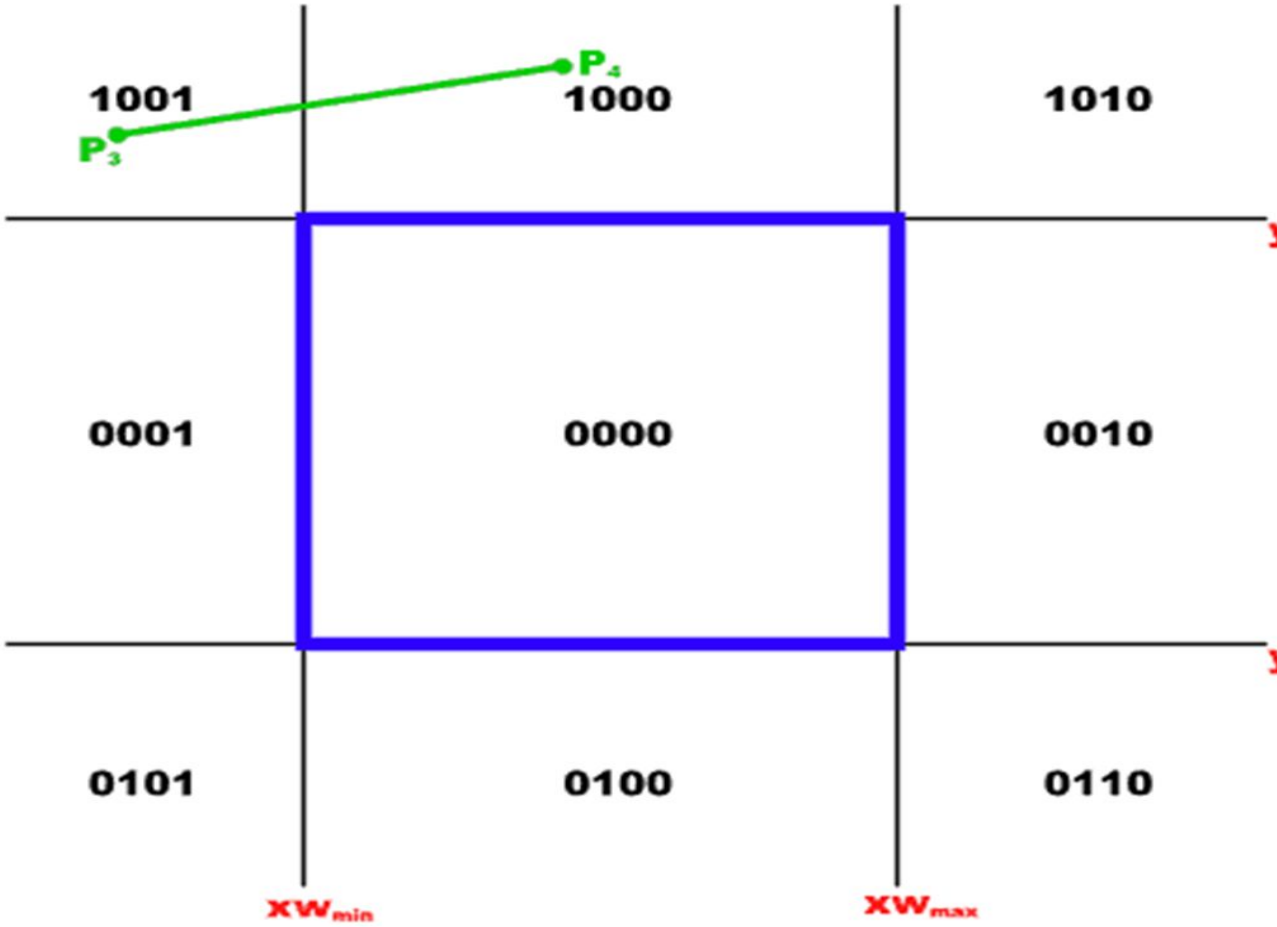
$P_2 = 0\ 0\ 0\ 0$

--- 0 0 0 0

=> accept segment $P_1\ P_2$

(trivial accept since both end points are in the window

Cohen-Sutherland Line Clipping Algorithm



4 bit clip code:
A B R L
A = y > yw_{max}
B = y < yw_{min}
R = x > xw_{max}
L = x < xw_{min}

Calculate clip code

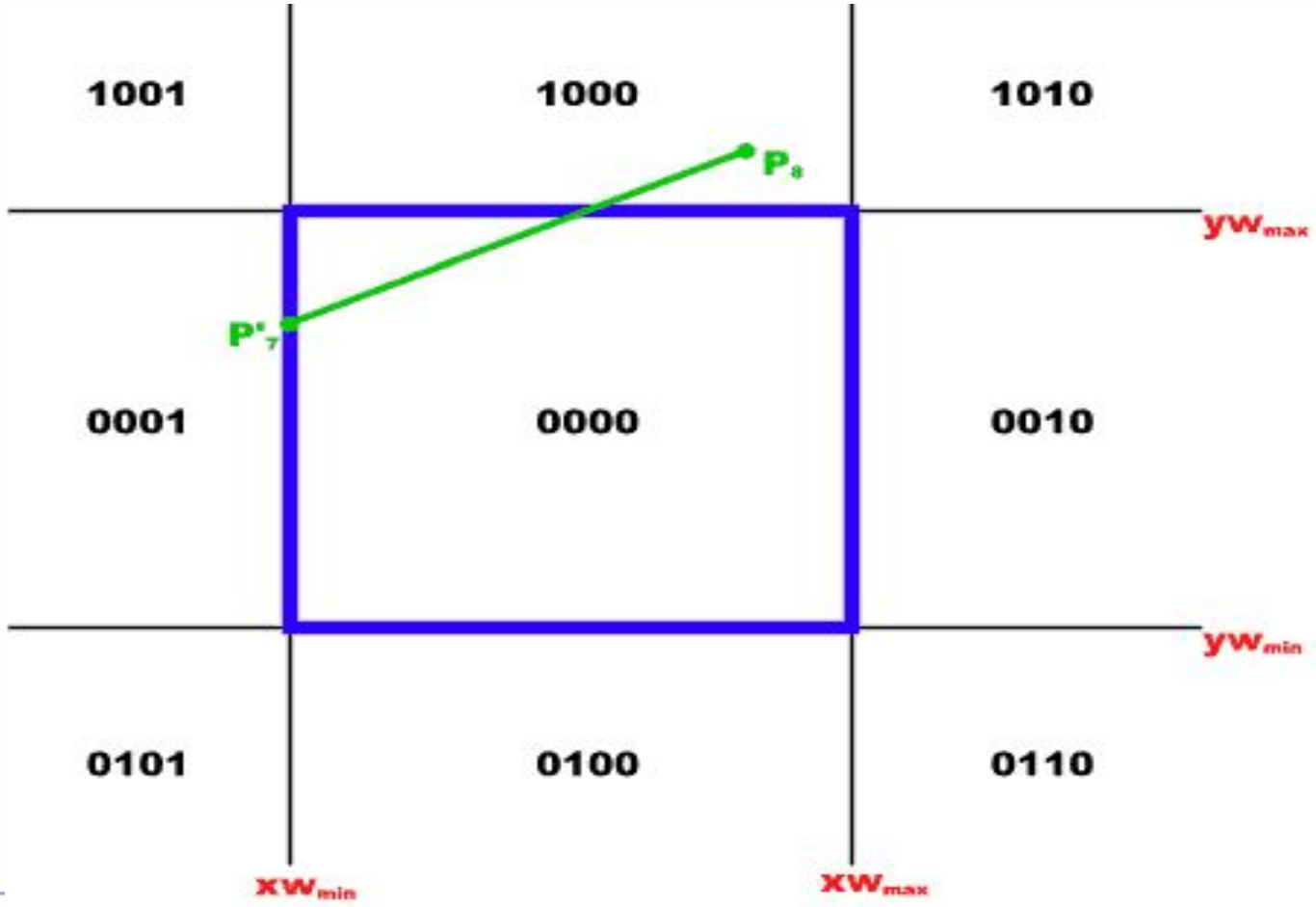
$P_3 = 1\ 0\ 0\ 1$

$P_4 = 1\ 0\ 0\ 0$

--- 1 0 0 0

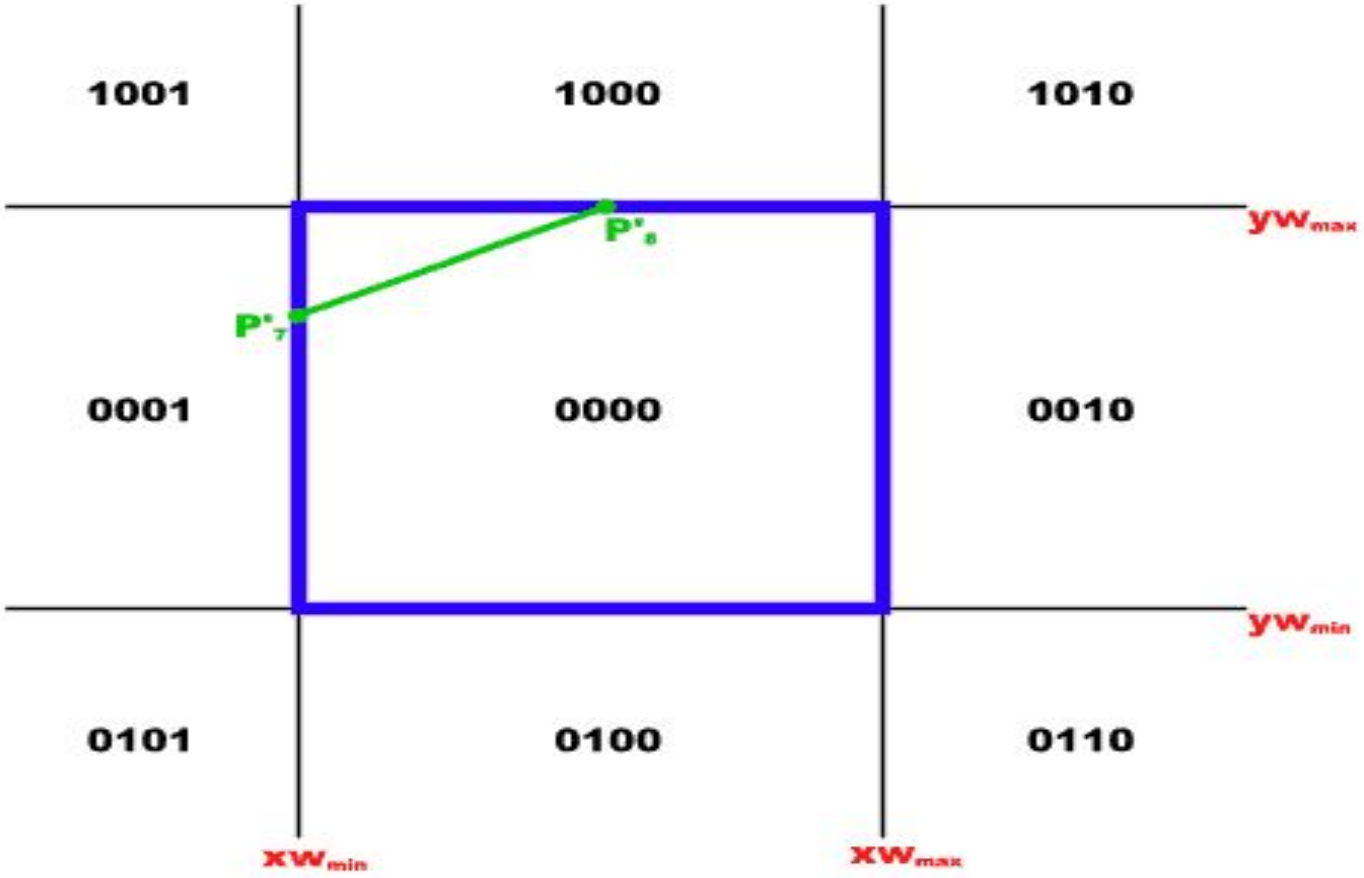
=> reject segment P_3 P_4

Cohen-Sutherland Line Clipping Algorithm



4 bit clip code:
A B R L
A = y > yw_{max}
B = y < yw_{min}
R = x > xw_{max}
L = x < xw_{min}

Cohen-Sutherland Line Clipping Algorithm



4 bit clip code:
A B R L
A = y > yw_{max}
B = y < yw_{min}
R = x > xw_{max}
L = x < xw_{min}

Cohen-Sutherland Line Clipping Algorithm

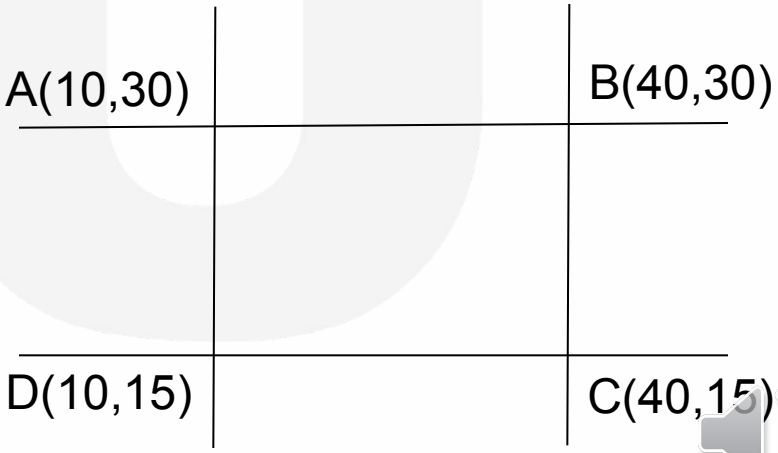
Example 1:

A window is defined as A(10,30) , B(40,30) , C (40,15) and D (10,15)

Line PQ cordinates are P(20,20) and Q(15,30)

Xmin = 10 and Xmax = 40

Ymin = 15 and Ymax = 30



Cohen-Sutherland Line Clipping Algorithm

Step 1. Find ABRL CODE for both point P(20,20) and Q(15,30)

Xmin = 10 and Xmax = 40

Ymin = 15 and Ymax = 30

ABRL CODE

Set 1 if Condition Satisfies, Else 0

A = Y > Ymax

B = Y < Ymin

R = X > Xmax

L = X < Xmin

Cohen-Sutherland Line Clipping Algorithm

P(20,20) = 0000

A = 20 > 30 **FALSE** = 0

B = 20 < 15 **FALSE** = 0

R = 20 > 40 **FALSE** = 0

L = 20 < 10 **FALSE** = 0

Q(15,30) = 0000

A = 30 > 30 **FALSE** = 0

B = 30 < 15 **FALSE** = 0

R = 15 > 40 **FALSE** = 0

L = 15 < 10 **FALSE** = 0

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

Cohen-Sutherland Line Clipping Algorithm

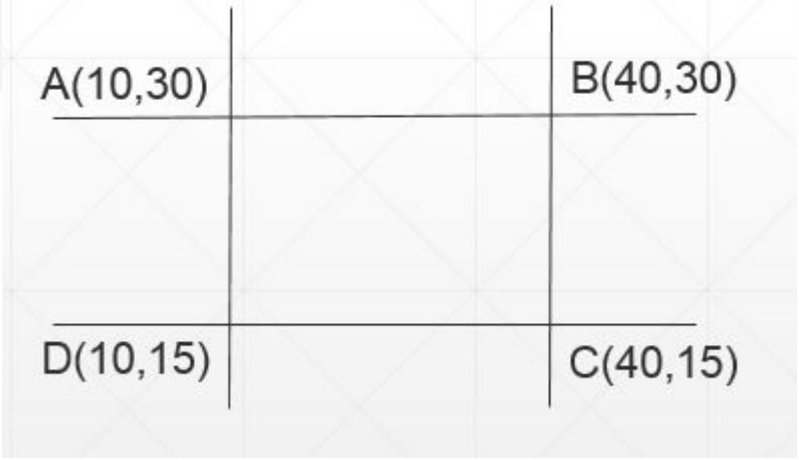
Example 2:

A window is defined as A(10,30) , B(40,30) , C (40,15) and D (10,15)

Line RS cordinates are R(15,10) and S(45,5)

Xmin = 10 and Xmax = 40

Ymin = 15 and Ymax = 30



Cohen-Sutherland Line Clipping Algorithm

Step 1. Find ABRL CODE for both point R(15,10) and S(45,5)

Xmin = 10 and Xmax = 40

Ymin = 15 and Ymax = 30

ABRL CODE

Set 1 if Condition Satisfies, Else 0

A = Y > Ymax

B = Y < Ymin

R = X > Xmax

L = X < Xmin

Cohen-Sutherland Line Clipping Algorithm

P(15,10) = 0100

A = 10 > 30 FALSE = 0

B = 10 < 15 TRUE = 1

R = 15 > 40 FALSE = 0

L = 15 < 10 FALSE = 0

Q(45,5) = 0110

A = 5 > 30 FALSE = 0

B = 5 < 15 TRUE = 1

R = 45 > 40 TRUE = 1

L = 45 < 10 FALSE = 0

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

Cohen-Sutherland Line Clipping Algorithm

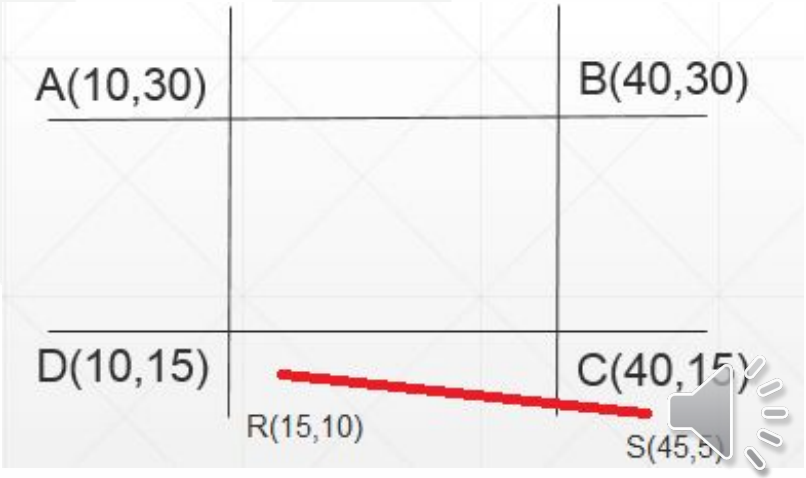
Step 3 – Else, perform the logical **AND** operation for both region codes.

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

Perform AND Operation on Region Code

R[0100] **AND** S[0110] = **0100**

Result is **NOT 0000**. Therefore Reject the Line RS



Cohen-Sutherland Line Clipping Algorithm

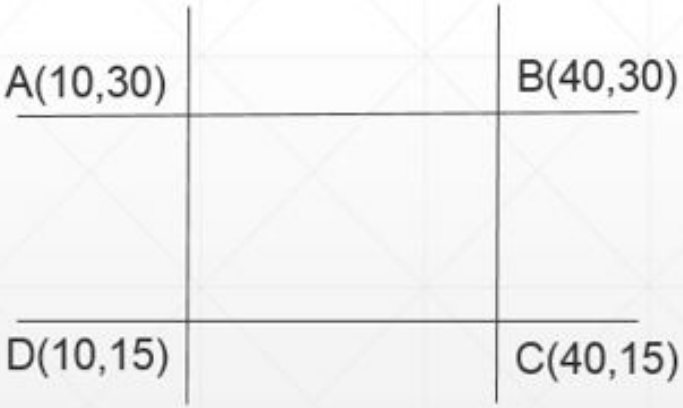
Example 3:

A window is defined as A(10,30) , B(40,30) , C (40,15) and D (10,15)

Line RS cordinates are E(20,20) and F(30,40)

Xmin = 10 and Xmax = 40

Ymin = 15 and Ymax = 30



Cohen-Sutherland Line Clipping Algorithm

Step 1. Find ABRL CODE for both point E(20,20) and F(30,40)
Xmin = 10 and Xmax = 40
Ymin = 15 and Ymax = 30

ABRL CODE

Set 1 if Condition Satisfies, Else 0

A = Y > Ymax

B = Y < Ymin

R = X > Xmax

L = X < Xmin

Cohen-Sutherland Line Clipping Algorithm

E(20,20) = 0000

A = 20 > 30 **FALSE** = 0

B = 20 < 15 **FALSE** = 0

R = 20 > 40 **FALSE** = 0

L = 20 < 10 **FALSE** = 0

F(30,40) = 1000

A = 40 > 30 **TRUE** = 1

B = 40 < 15 **FALSE** = 0

R = 30 > 40 **FALSE** = 0

L = 30 < 10 **FALSE** = 0

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

Cohen-Sutherland Line Clipping Algorithm

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.

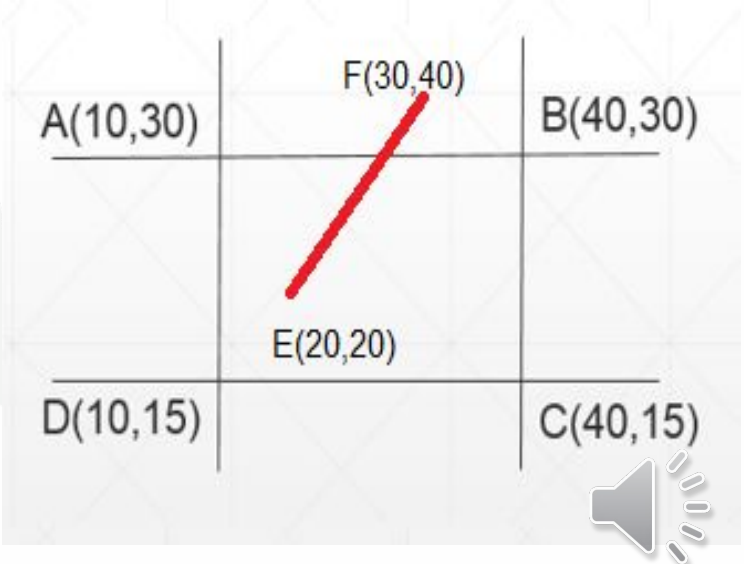
Perform AND Operation on Region Code

E[0000] **AND** F[1000] = **0000**

Step 4 – For Clipping we need to find Line equation

Line Equation – y=mx+b

Where:m is the slope, and b is the y-intercept



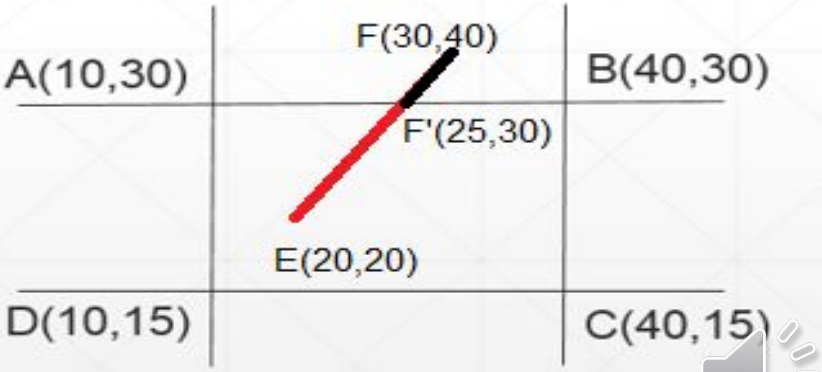
Cohen-Sutherland Line Clipping Algorithm

Line Equation of (20,20) and (30,40)
m= (y2-y1)/(x1-x1)
m=2
Therefore now - y=2x+b
You can use either (x,y) point you want. The answer will be the same:
(20,20)
y=mx+b or 20=2 × 20+b, or b=20-(2)(20)
b=-20.
(30,40)
y=mx+b or 40=2 × 30+b, or b=40-(2)(30)
b=-20.
Line Equation is y=2x-20

Cohen-Sutherland Line Clipping Algorithm

If Line cuts TOP : y=ymax
If Line Cuts BOTTOM : y=ymin
If Line Cuts LEFT : x=xmin
If Line Cuts Right : x=xmax

Line Equation is y=2x-20 – our line cuts TOP i.e y=ymax
y=2x-20
30=2x-20
50=2x
x=50/2
x=25
Clipped point is F'(25,30)
Clipped Line is EF' – E(20,20) and F(25,30)

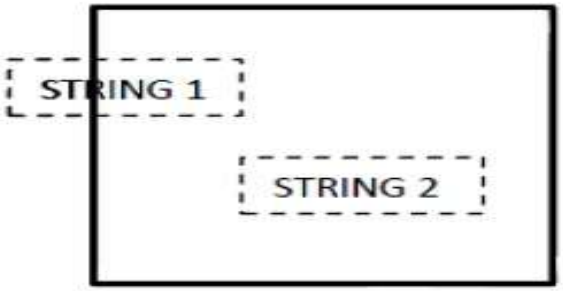


Text Clipping

Various techniques are used to provide text clipping. Three methods for text clipping are listed below –

- All or none string clipping
- All or none character clipping
- Text clipping

All or none String Clipping



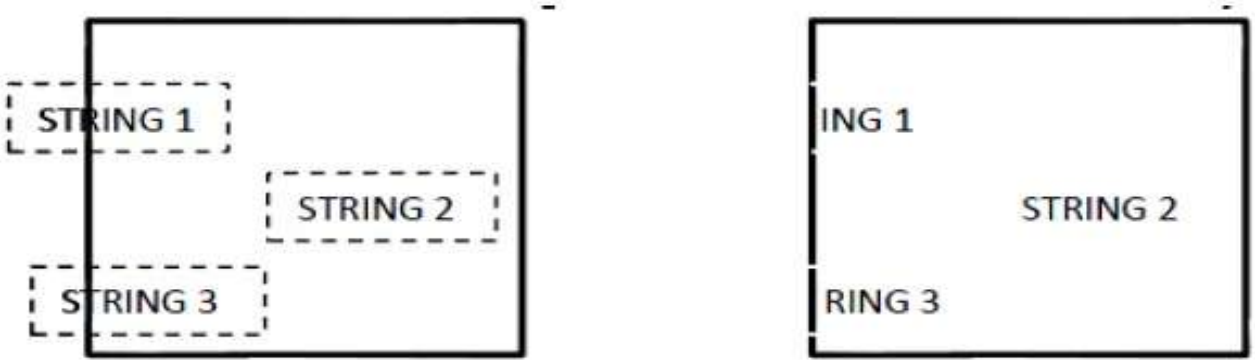
Before Clipping

After Clipping

- Either we keep the entire string or we reject entire string based on the clipping window.
- STRING2 is entirely inside the clipping window so we keep it and STRING1 being only partially inside the window, we reject.

All or none String Clipping

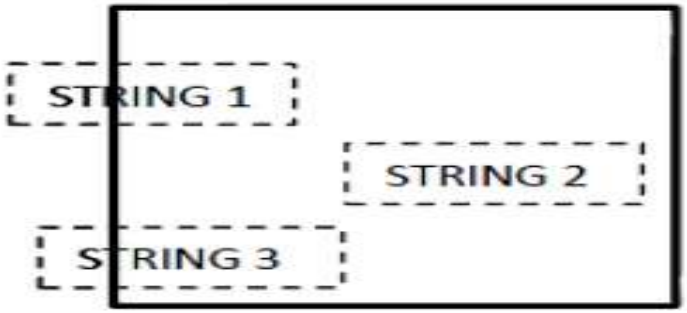
- If the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then –
- You reject only the portion of the string being outside
- If the character is on the boundary of the clipping window, then we discard that entire char.acter and keep the rest string.



Before Clipping

After Clipping

Text Clipping



Before Clipping

After Clipping

- If the string is entirely inside the clipping window:- Then keep it.
- If string is partially outside the window: - Reject only the portion of string being outside.
- If the character is on the boundary of the clipping window: Discard only that portion of character that is outside of the clipping window.

Polygon Clipping (Sutherland Hodgman Algorithm)

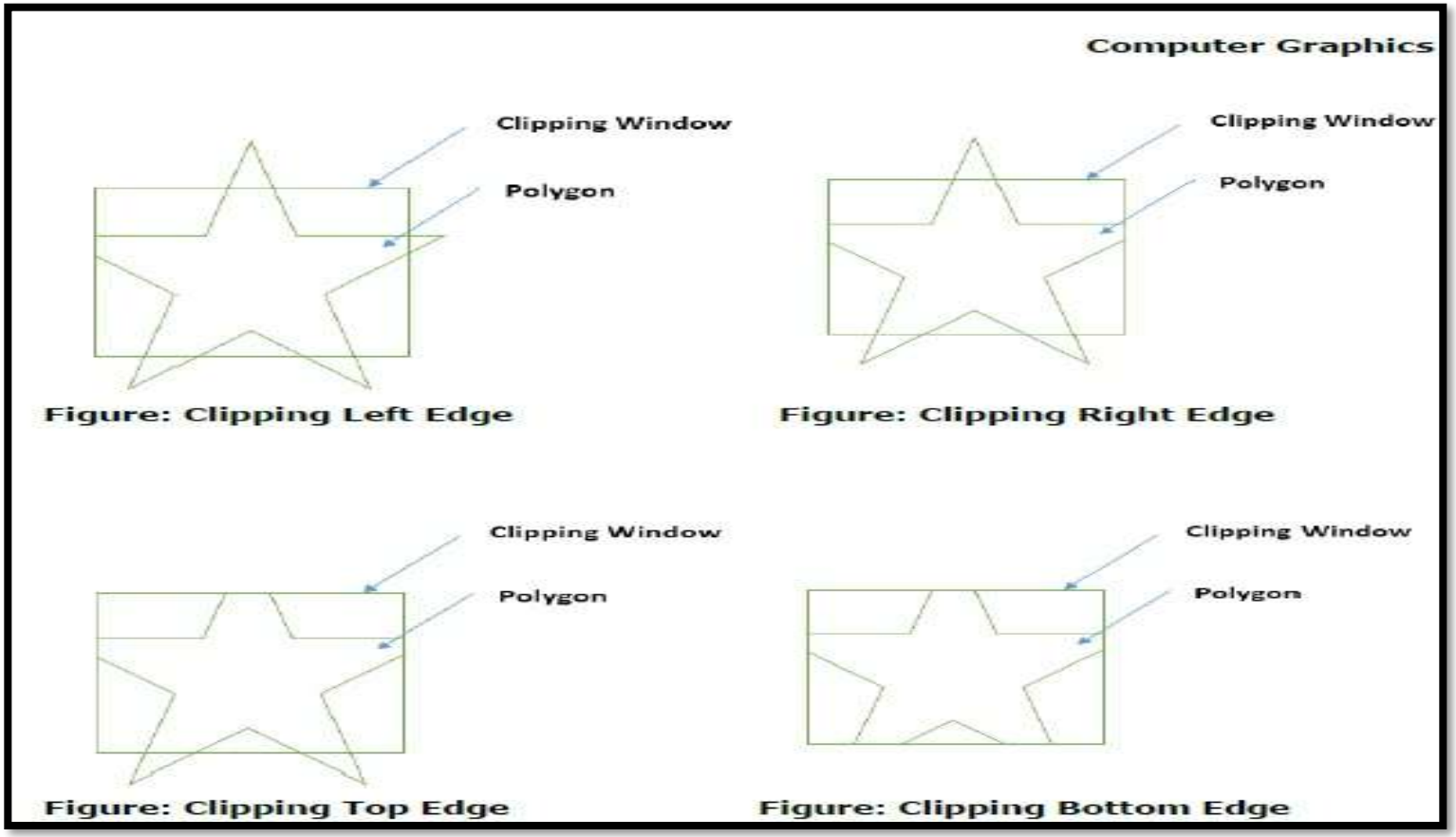
A polygon can also be clipped by specifying the clipping window.

- Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping.
- Here all the vertices of the polygon are clipped against each edge of the clipping window.

Polygon Clipping (Sutherland Hodgman Algorithm)

- Here 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.

Polygon Clipping

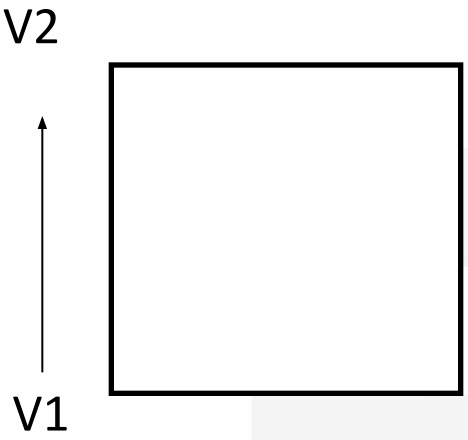


Pipelined Polygon Clipping

- Here all the vertices of the polygon are clipped against each edge of the clipping window.
- Clipping against one edge is *independent* of all others, it is possible to arrange the clipping stages in a **pipeline**.
- This way four polygons can be at different *stages* of the clipping process simultaneously.
- 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.

Polygon Clipping: 4 Rules to be followed

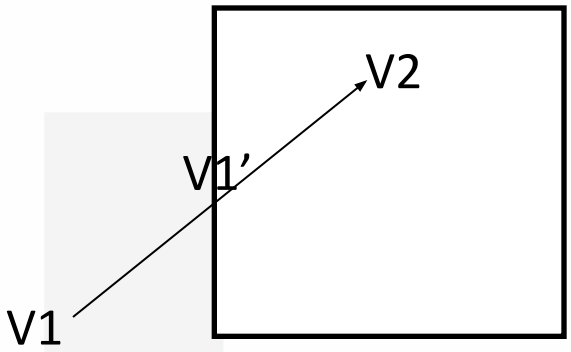
1)



V1 - OUTSIDE
V2 –OUTSIDE

SAVE NOTHING

2)

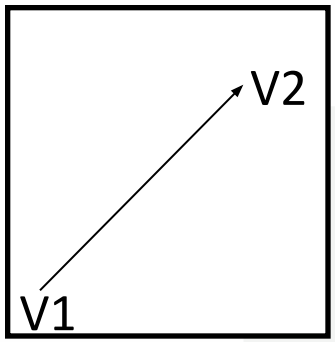


V1 - OUTSIDE
V2 - INSIDE

SAVE V1’
SAVE V2

Definition: 4 Rules to be followed

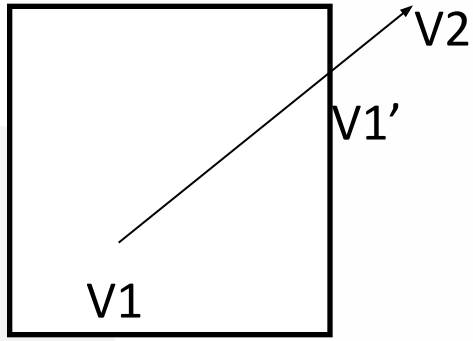
3)



V1 - INSIDE
V2 - INSIDE

SAVE V2

4)



V1 - INSIDE
V2 - OUTSIDE

SAVE V1'

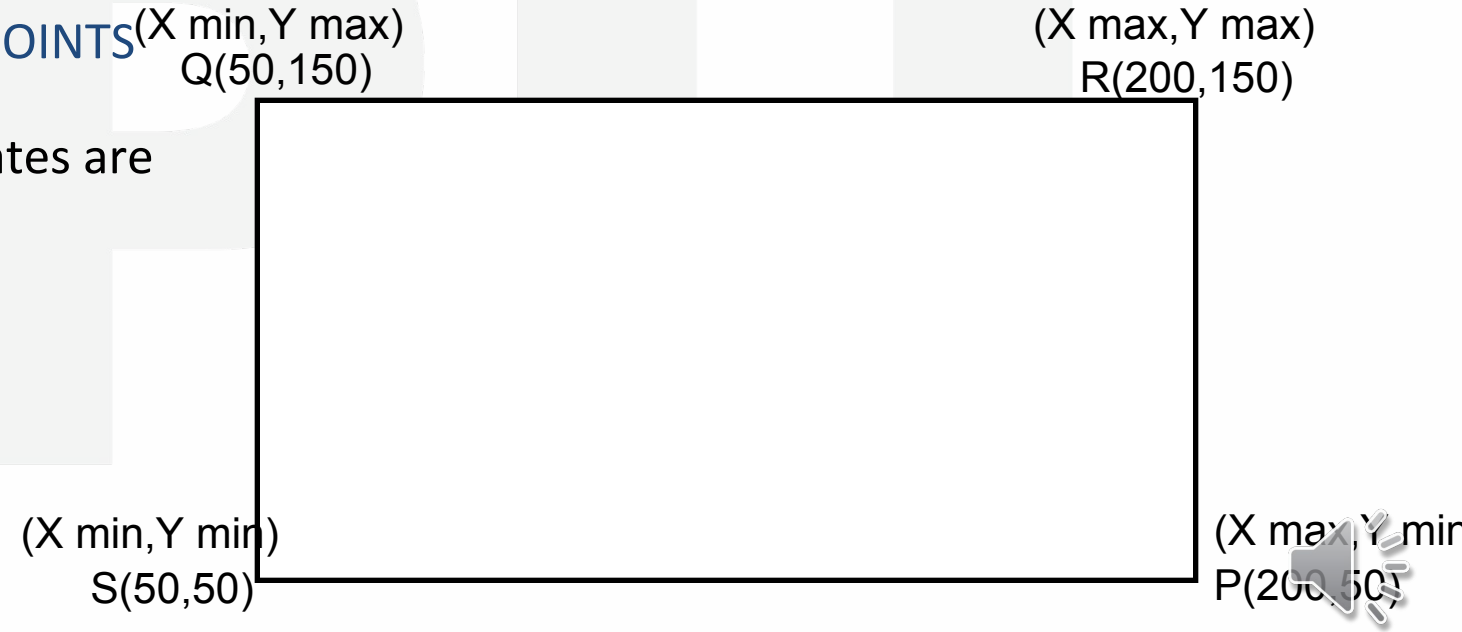
Numeric Example

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- ordination of the window are P(200,50), Q(50,150), R(200,150), S(50,50).

Answer:-

STEP 1: PLOT THE POINTS

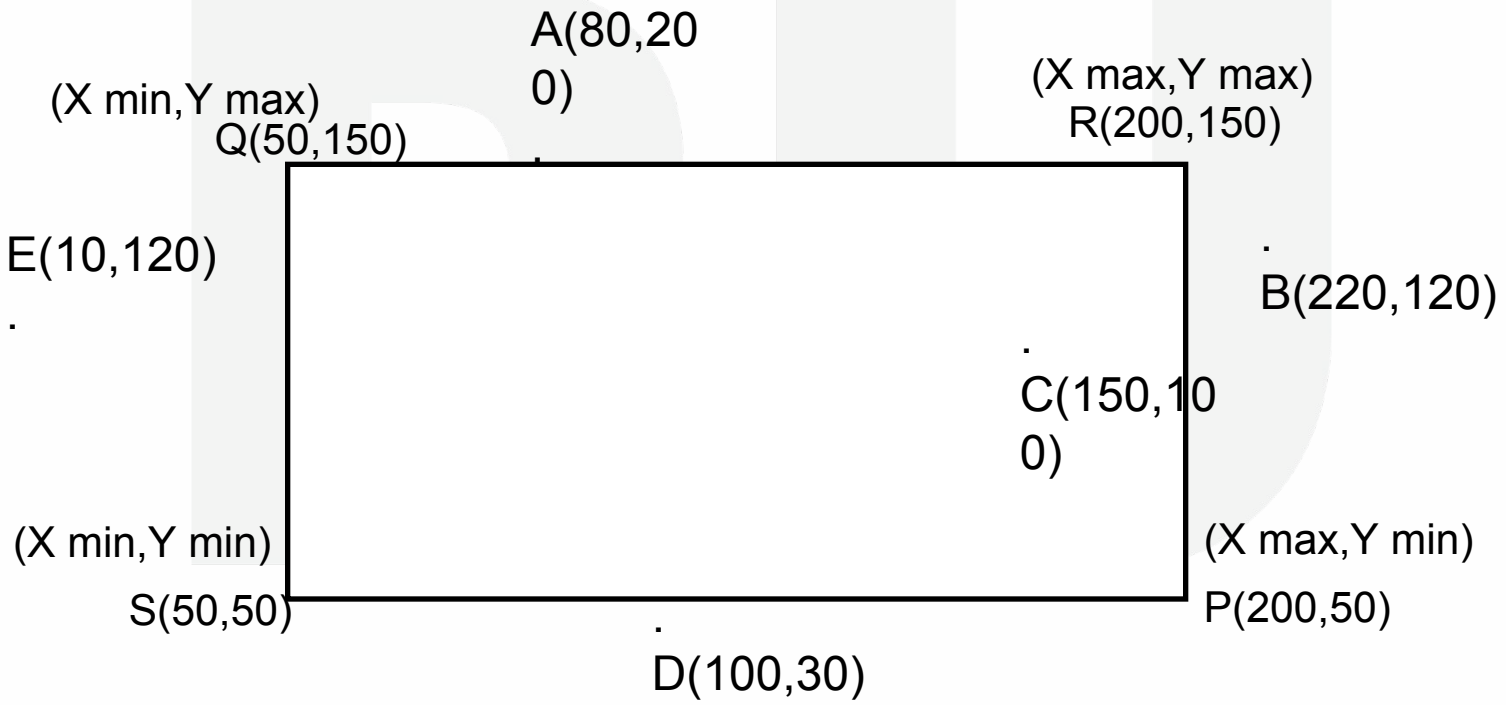
Window Co-ordinates are
P(200,50)
Q(50,150)
R(200,150)
S(50,50)



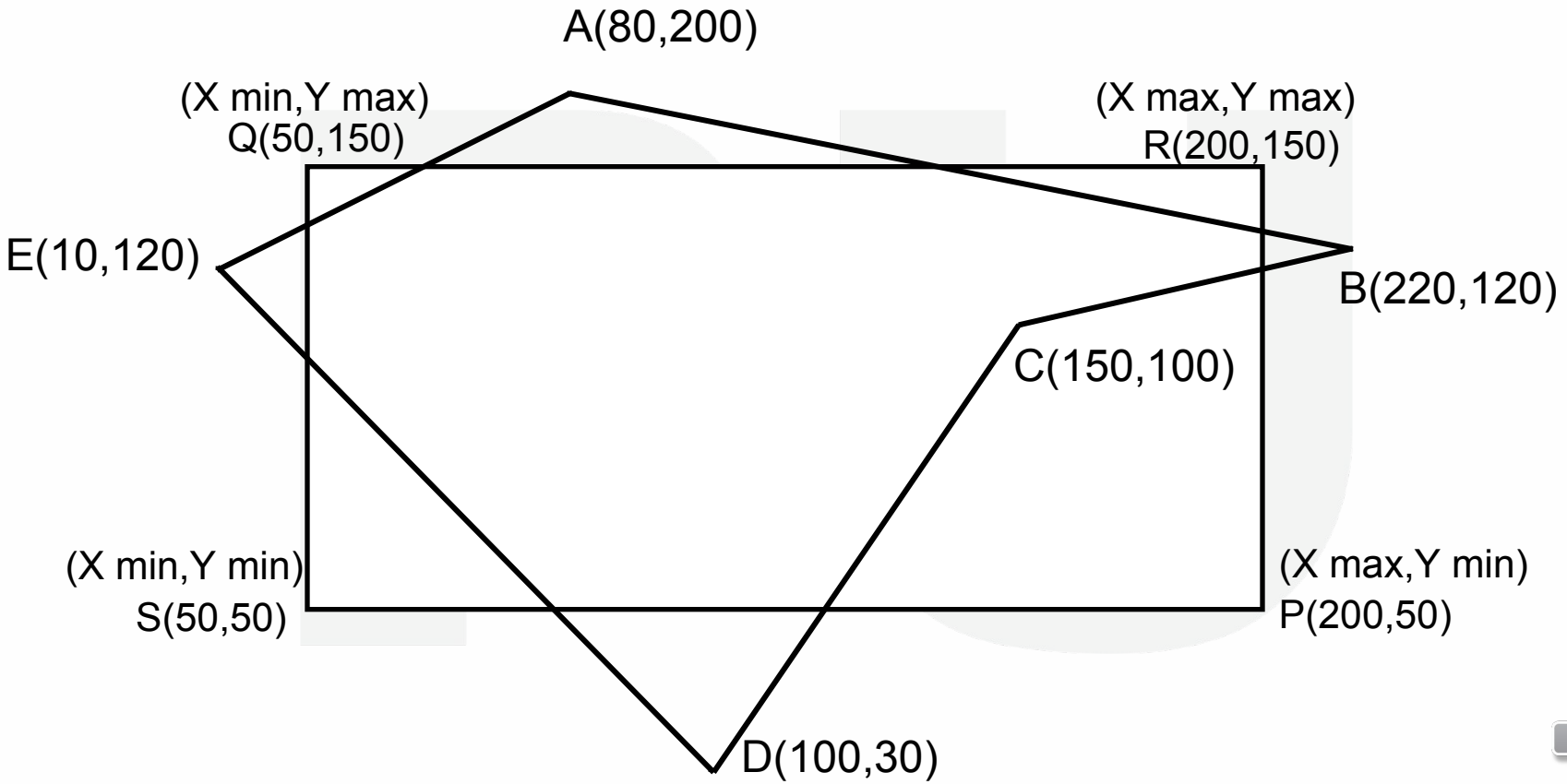
Numeric Example

POLYGON CO-ORDINATES ARE:-

A(80,200), B (220,120),C (150,100), D (100,30) , E (10,120).

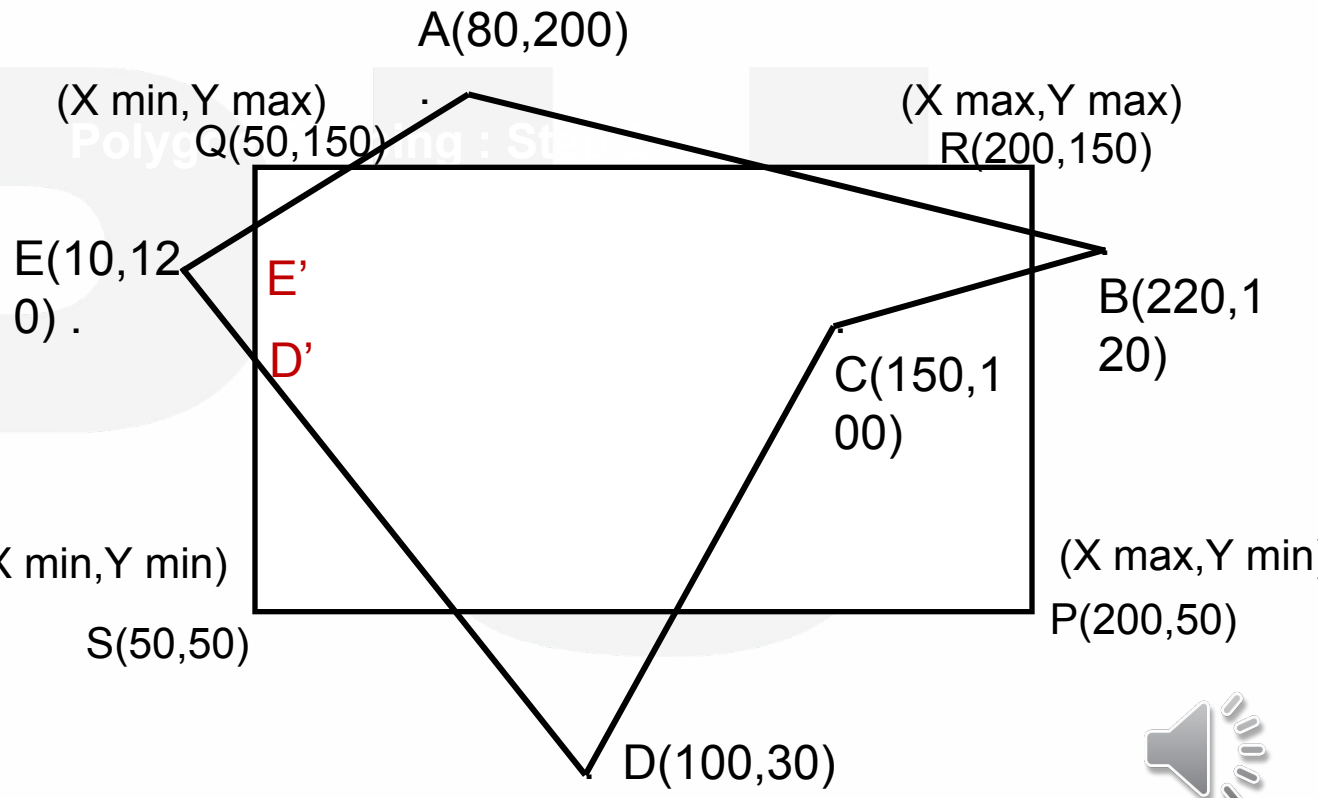


Polygon Clipping: Join the points of Polygon



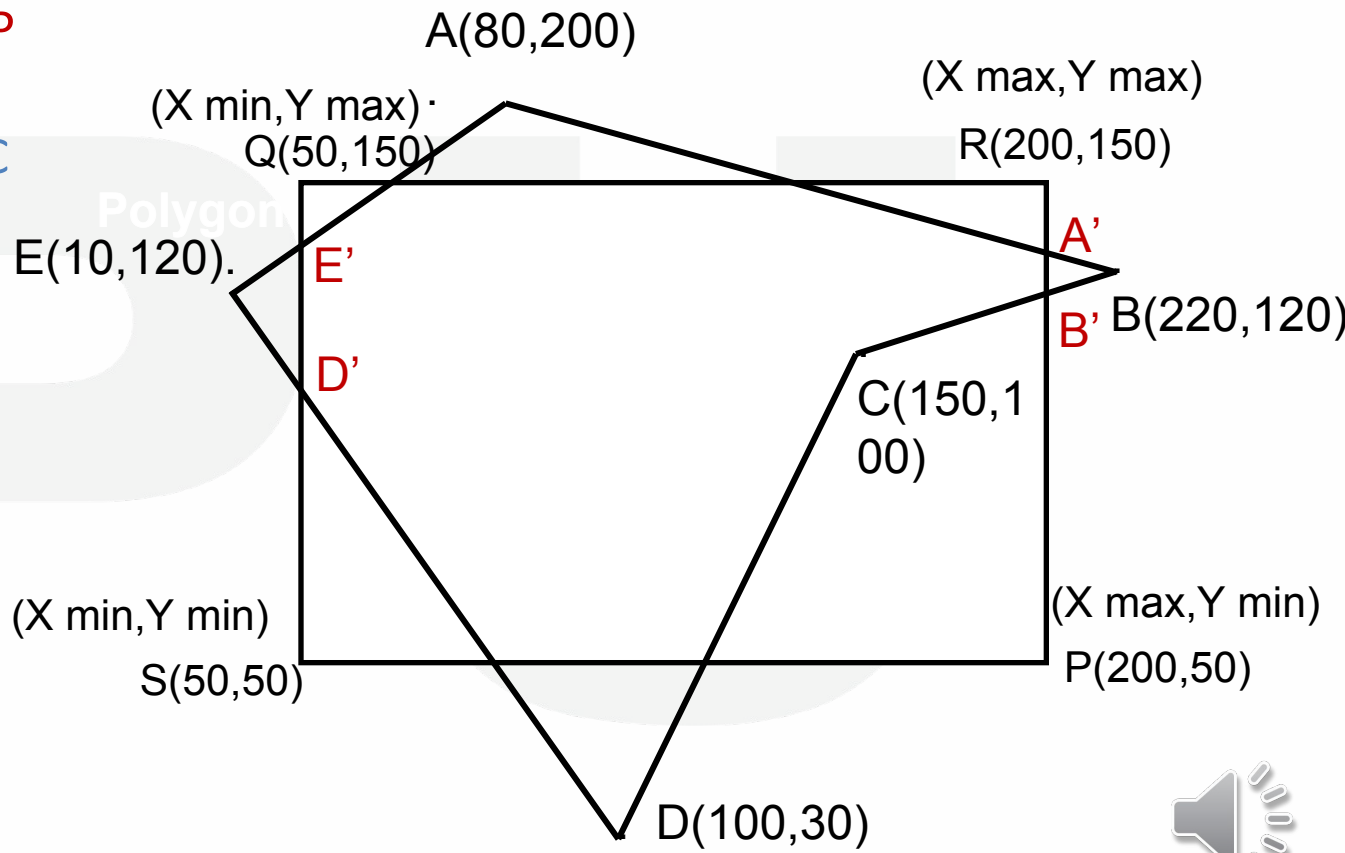
Polygon Clipping : Step 2(LEFT Clipping)

VERTEX	CASE	O/P
AB	IN>IN	B
	IN>IN	C
	IN>IN	D
	IN>OUT	D’
	OUT>IN	E’A



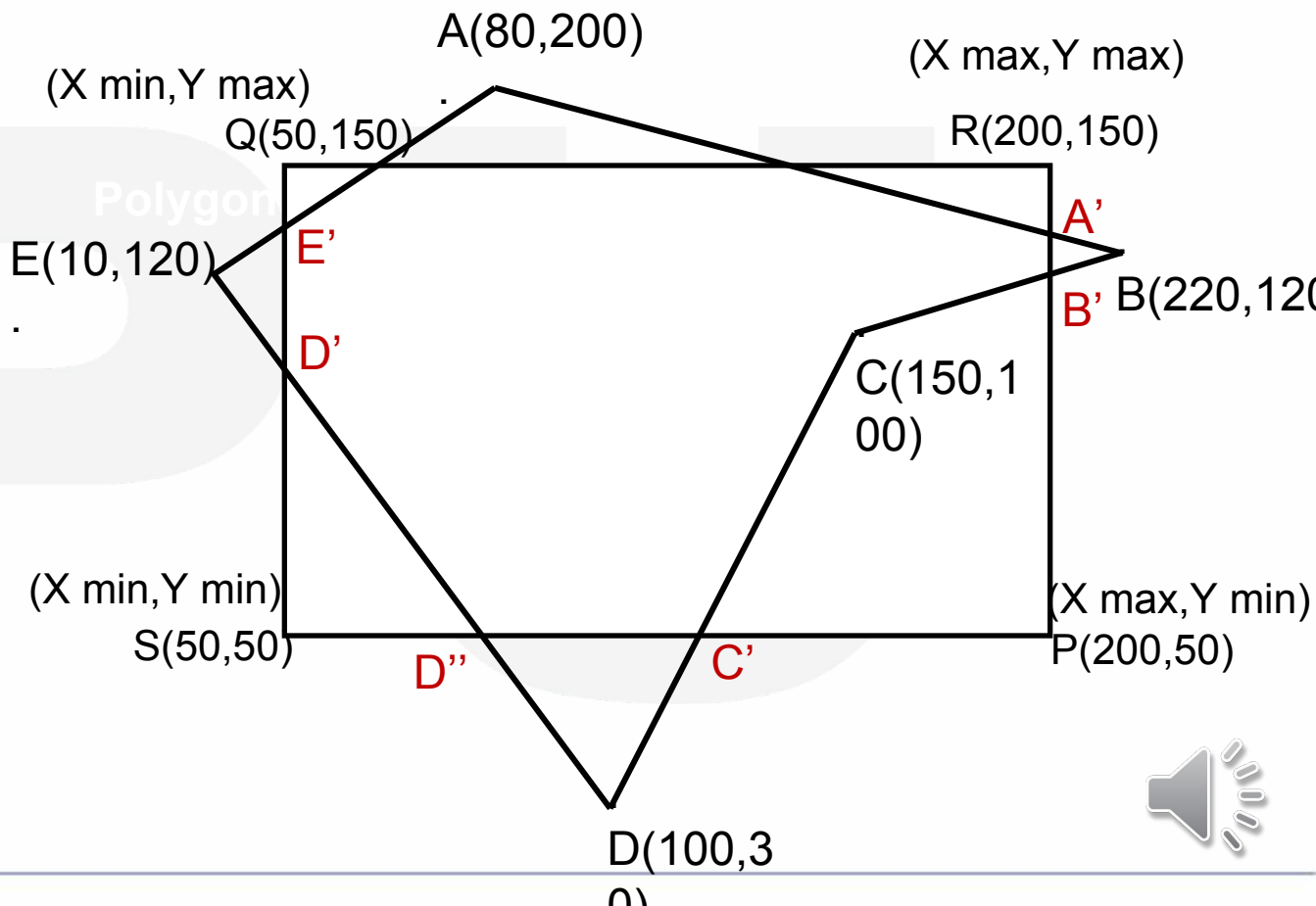
Polygon Clipping : Step 3 (RIGHT Clipping)

VERTEX	CASE	O/P
AB	IN>OUT	A’
BC	OUT>IN	B’C
CD	IN>IN	D
DD’	IN>IN	D’
D’E	IN>IN	E’
E’A	IN>IN	A



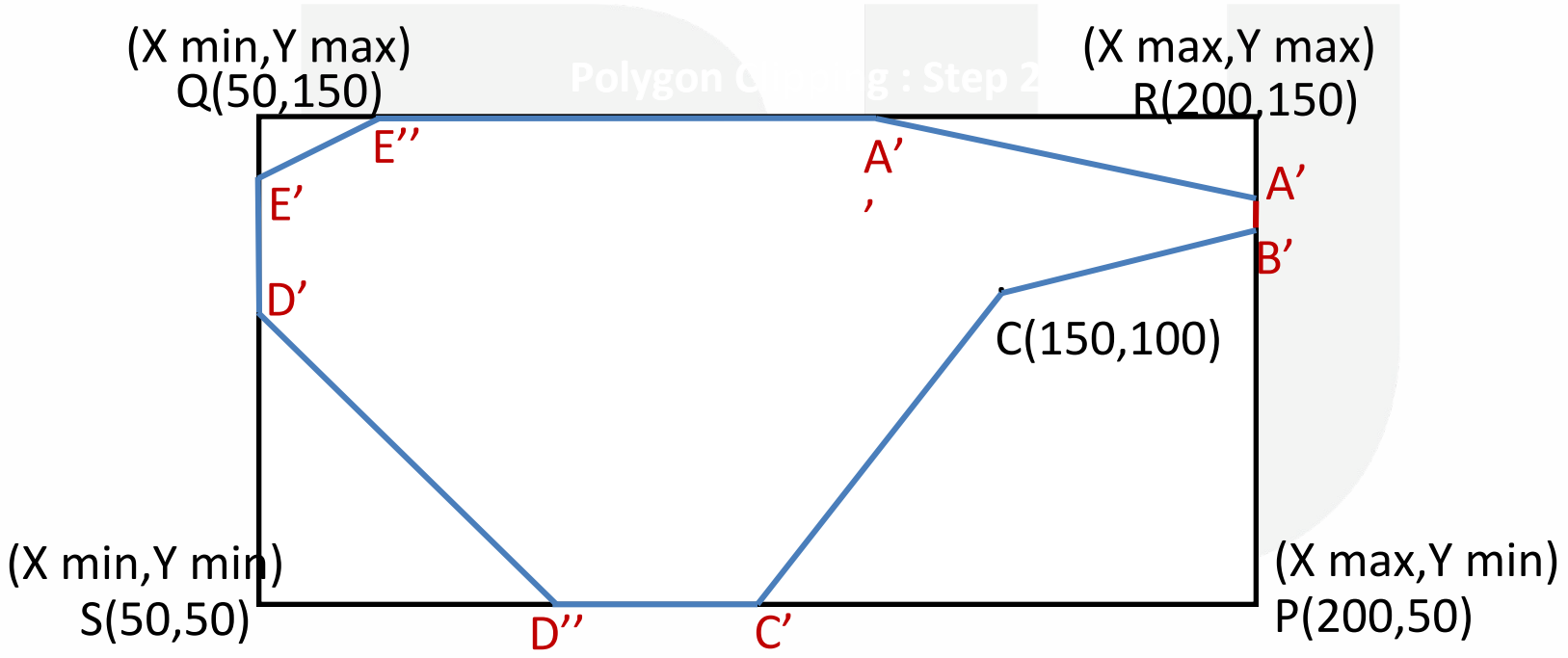
Polygon Clipping : Step 4 : (BOTTOM Clipping)

VERTEX	CASE	O/P
AA’	IN>IN	A’
A’B’	IN>IN	B’
B’C	IN>IN	C
CD	IN>OUT	C’
DD’	OUT>IN	D’’D’
D’E’	IN>IN	E’
E’A	IN>IN	A



Polygon Clipping : Step 2

Clip all the 4 side & join new points.



Liang–Barsky Algorithm

- In computer graphics, the Liang–Barsky algorithm is a line clipping algorithm.
- The Liang–Barsky algorithm uses the parametric equation of a line and inequalities describing the range of the clipping window to determine the intersections between the line and the clip window.
- With these intersections it knows which portion of the line should be drawn.
- This algorithm is significantly more efficient than Cohen–Sutherland

Steps of Liang–Barsky Algorithm

1. Umin = 0 and Umax = 1
2. Uk = qk/pk (‘K’ is Constant)
3. Umin <= Uk <= Umax i.e 0 <= Uk <= 1
4. Use Following Inqualities to Calculate qk and pk
   - **p0 = -dx q0 = x0 – xmin**
   - **p1 = dx q1 = xmax – x0**
   - **p2 = -dy q2 = y0 – ymin**
   - **p3 = dy q3 = ymax – y0**
5. To Calculate Intersections Points
   - X = x0 + Udx, Y = y0 + Udy

Liang–Barsky Algorithm

Let ABCD be the rectangular window with A(0,0) , B(10,0) , C (10,10) , D(0,10)

Clip Line PQ with P(-5,3) and Q(15,9)

Step 1. Find dx and dy

dx = x1 - x0 = 15 – (-5) = **20**

dy = y1 - y0 = 9 – 3 = **6**

Step 2. Find Uk=qk/pk where k = 0,1,2,3.

u0 = q0/p0 = (x0 - xmin)/(-dx) = (-5-0)/(-20) = ¼ = **0.25**

u1 = q1/p1 = (xmax – x0)/(dx) = (10+5)/(20) = ¾ = **0.75**

u3 = q2/q2 = (y0-ymin)/(-dy) = (3-0)/(-6) = -3/6 = **-0.5**

u4 = q3/p3 = (ymax – y0)/(dy) = (10-3)/6 = 7/6 = **1.16**

Liang–Barsky Algorithm

Consider Values of Uk if it Satisfies Umin <= Uk <= Umax i.e 0 <= Uk <= 1

We consider U0 = 0.25 and U1= 0.75

Step 4. Calculate Intersection Points

1. **When U = 0.25**

   X = x0 + Udx

   X = -5 + 0.25(20) = **0**

   Y = y0 + Udy

   Y = 3 + 0.25(6) = **4.5** **Therefore (X,Y) = (0,4.5)**

2. **When U = 0.75**

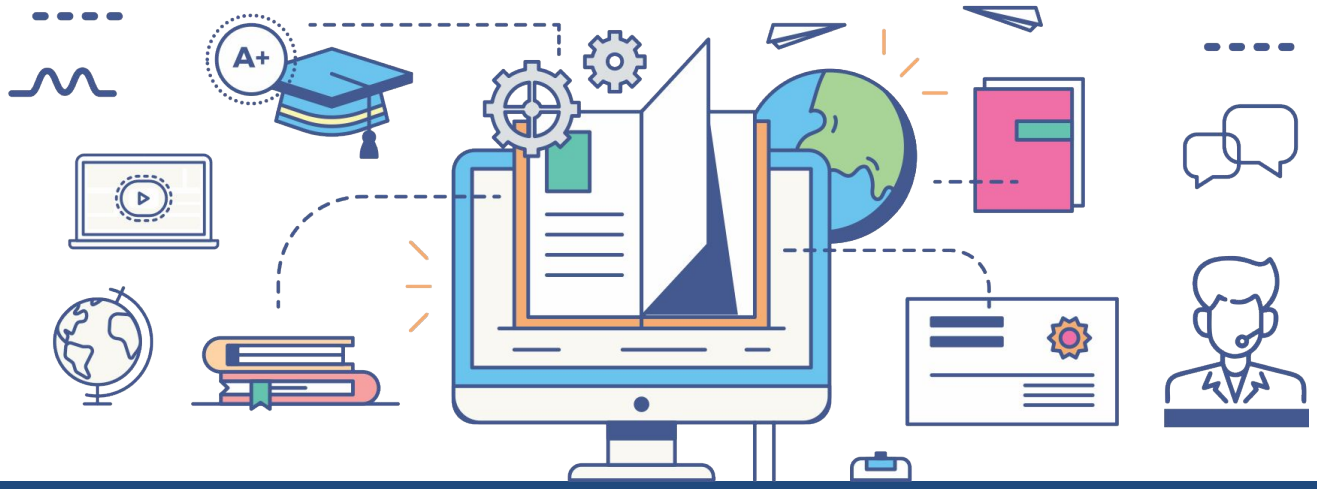
   X = x0 + Udx

   X = -5 + 0.75(20) = **10**

   Y = y0 + Udy

   Y = 3 + 0.75(6) = **7.5** **Therefore (X,Y) = (10,7.5)**

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in