Computer Graphics (22318)

Unit IV : Windowing & Clipping.

Practical Sure (04 comps.)

(weightage - 14 marks)

4.1 Window and clipping Concept: window to viewport Transformation.

Questions

Q// Explain the concepts window, viewport and window-to-view port transformation. (4M)

Q// Explain window to view port transformation. (6M)

Introduction

Windowing : The process of selecting and viewing the picture with different view is called as windowing.

Clipping : The process which divides each elements of picture into its visible and invisible portions, allowing invisible to be discarded is called as clipping.

Window to Viewport Transformation

* It is the process of transforming 2D world co-ordinate object device co-ordinates.
* Object inside the world or clipping window are mapped to viewport which is the area on the screen where world co-ordinates are mapped to be displayed.
* **Window** : A world co-ordinate area selected for display is called window.
* **Viewport** : A area on display device to which window is mapped is called view port.

Unit 6

* Normalization Transformation and workstation transformation leads to window to view port transformation

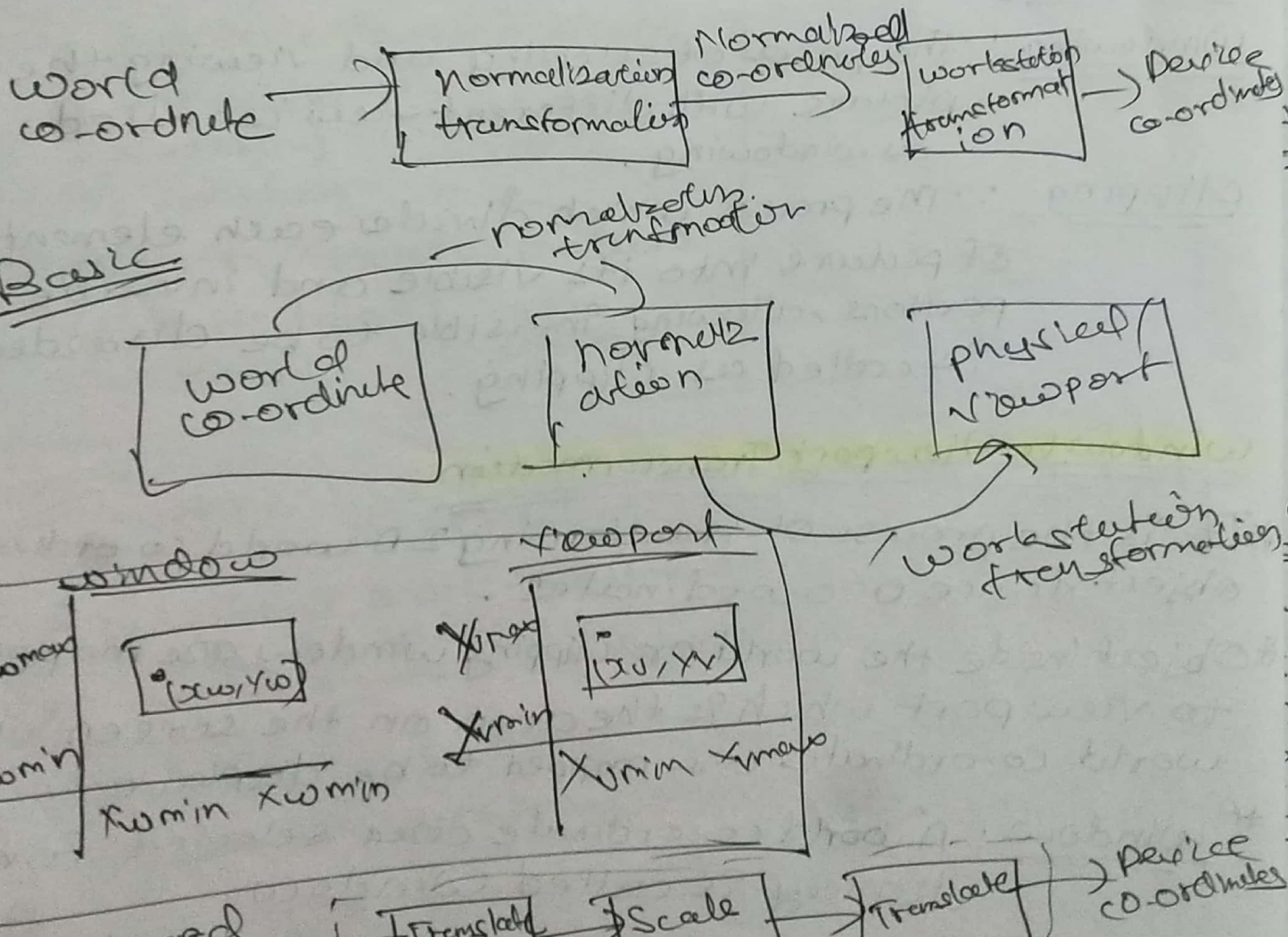
* View transformation consist of two parts as Normalization transformation and workstation transformation.

Normalisation transformation

It maps world co-ordinate system to Normalized Device Co-ordinate System (NDCS).

Workstation transformation

It maps into. Normalized Device Co-ordinate syst to physical Device Co-ordinate System.



Works on principle of Basic Graphic pipeline.

4.4. Text Clipping

Questions

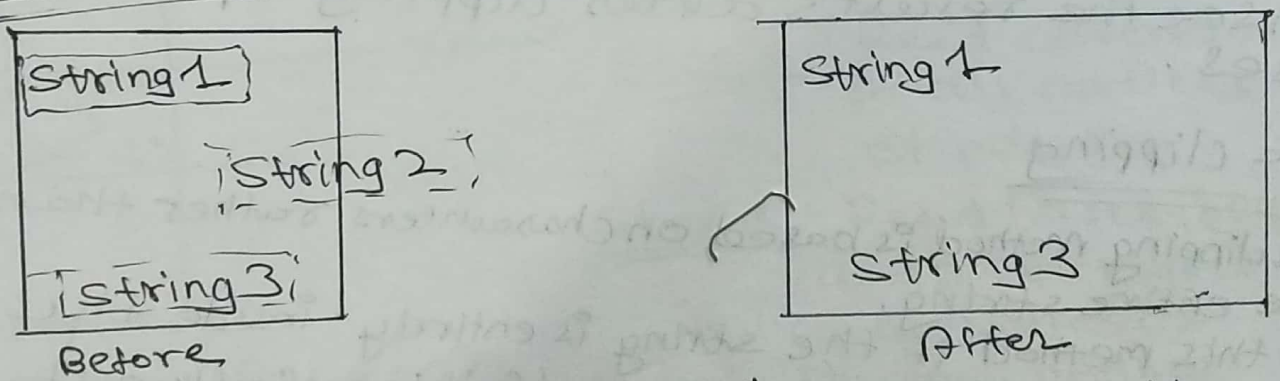
Q) Explain different types of Text clipping in brief. (4M)

Q) Explain Text Clipping ★★★

Q) Enlist different methods of Text clipping

- many techniques are used to provide text clipping in computer graphics.
- The methods use to clip text lie outside window is known as text clipping.
- It depends on method used to generate characters and the requirement of particular application
- there are three method.
  - All or None string clipping.
  - All or None character clipping.
  - Text clipping.

All or None string clipping



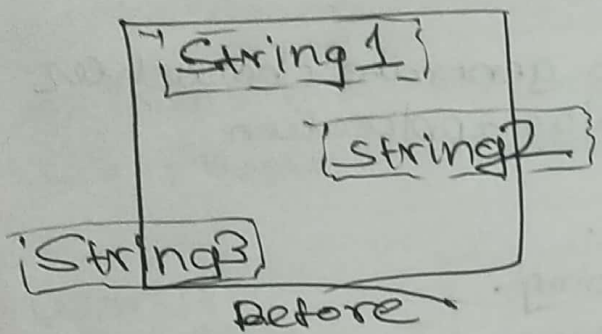
In all or None string clipping, either we keep entire string or we reject entire string based on clipping window.

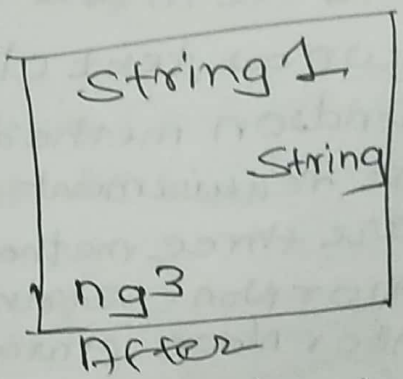
∴ As shown above. Before clipping string 1 & string 3 are totally inside window ~~After~~ & string 2 is partially inside and outside,
So, after clipping String 1 and string 3 will be remaining. String 2 will be clipped.

All or None character clipping

This clipping method is based on character rather than entire string.
In this method if string is entirely inside the clipping window, the we keep it. If it is parcetally outside the window then.

- You reject only the portion of string being outside
- If character is on boundary, then we discard that entire character.



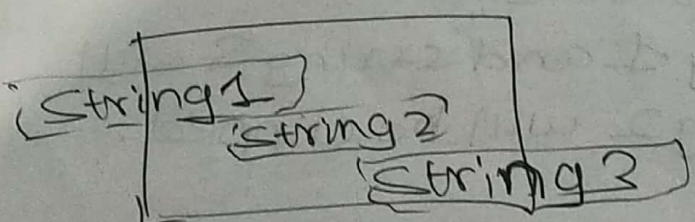


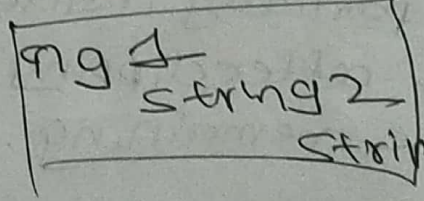
In above example, string 1 is completly inside string 2 and string 3 are parcially outside, then you can see the result's after clipping as per above cases.

Text clipping

This clipping method is based on characters rather than the entire string.
In this method if the string is entirely inside the clipping window, we keep it. (If it is parcically out side the.

- You reject only the portion of string being outside
- If character is on boundary of window, then we discard only that portion of character that is outside clipping window.





As you can see above example.

4.3 Polygon Clipping : Sutherland-Hodgeman

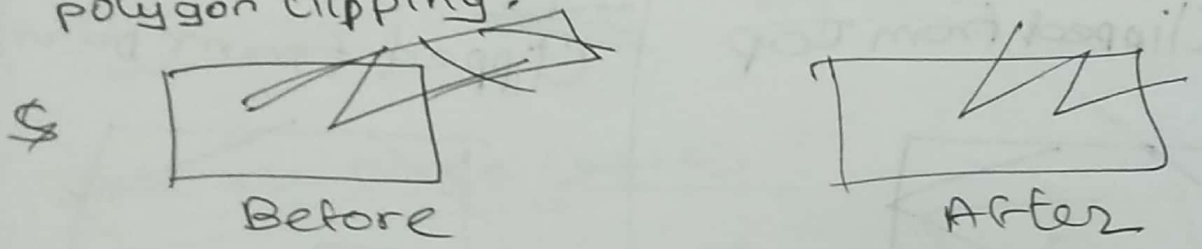
Questions

Q// Explain Sutherland-Hodgeman polygon clipping algorithm. (W-22, S-22)

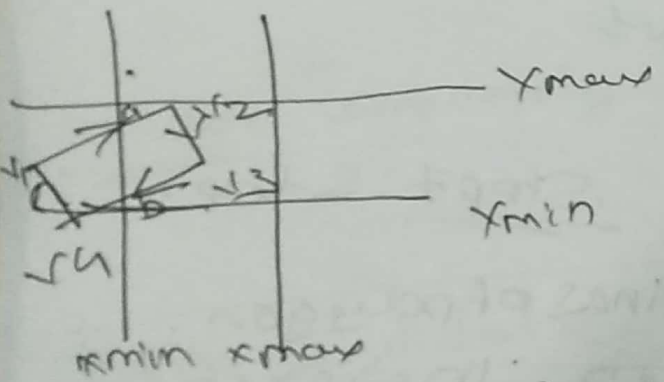
Q// Define Polygon Clipping.

Polygon clipping

The area or shape of polygon which lie outside the polygon is that is deleted is known as polygon clipping.



Sutherland Hodgeman



processing (Polygon)

Step 1 : Start

Step 2 : Read co-ordinate of all vertices of polygon.

Step 3 : Read lower left and upper right co-ordinates of clipping window.

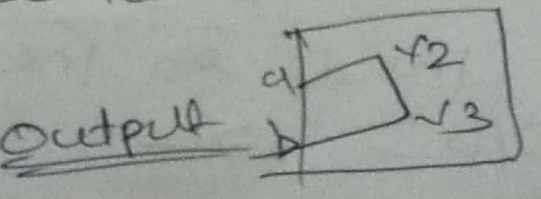
Step 4 : Process vertices against left edge of window to get output vertices.

Step 5: Repeat step 4 for right, top, bottom edges of polygon respectively to get final output vertices set.

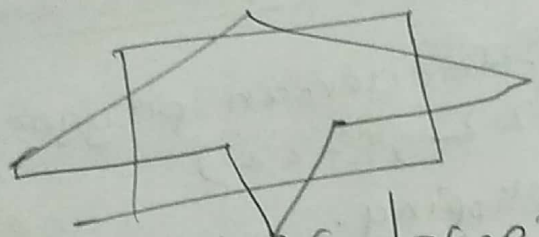
Step 6 : Display polygon connecting all vertices from output vertices list.

Step 7: Stop.

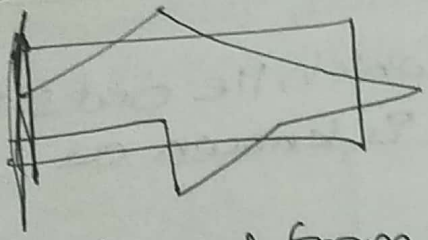
Output



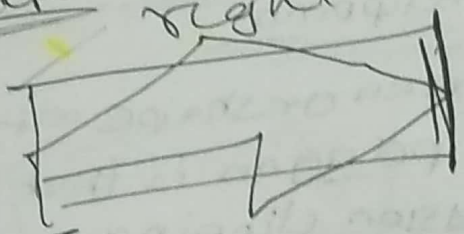
Step 1 : Consider polygon



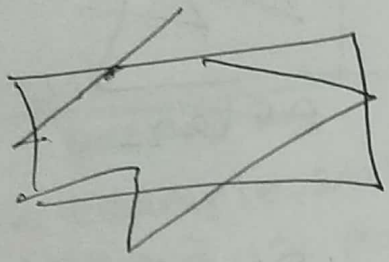
Step 2: Clipped from left



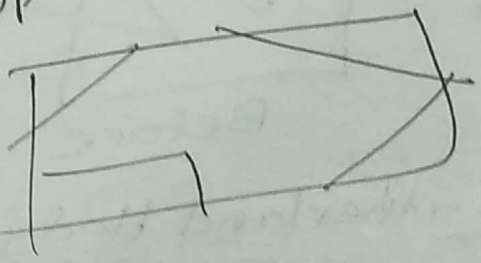
Step 3 Clipped from right



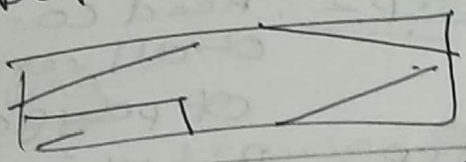
Step 4: Clipped from top



Step 5 Clipped from bottom



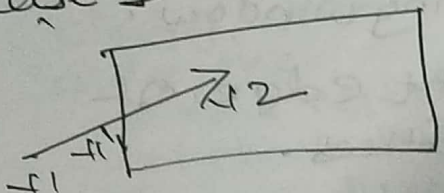
Step 6 : Display final output



Step 7 : Stop.

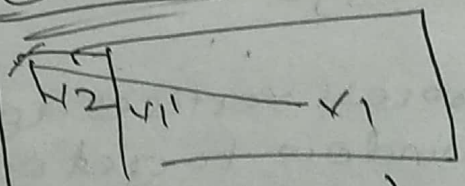
But there are 4 cases of lines of polygon.

Case 1 : Outside → Inside



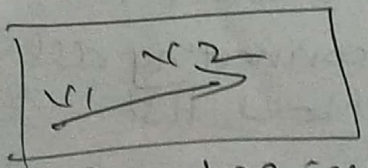
O/P → V1' V2

Case 2 : In → Out



O/P → V1'

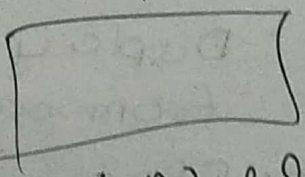
Case 3 :



Inside to Inside

O/P = V2

Case 4

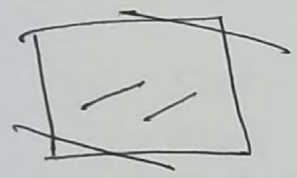


O/P = Null

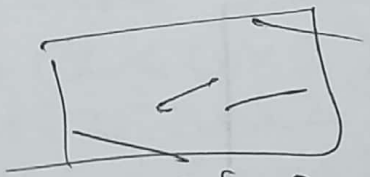
4.2 : Line clipping: Cohen Sutherland clipping algorithm, Cyrusbeck, Liang Barsky, midpoint Subdivision.

Line clipping

The part of lie which lie outside the clipping window that are discarded or deleted is known to be line clipping.



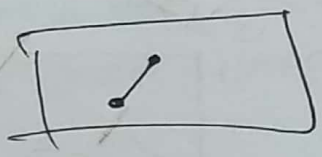
Before



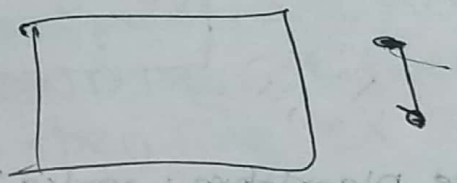
After

There are three cases

Visible: Compeletely inside

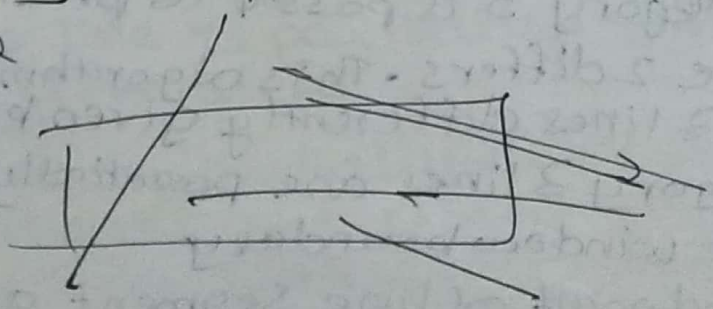


Invisible: Compeletely outside



Clipping candidate

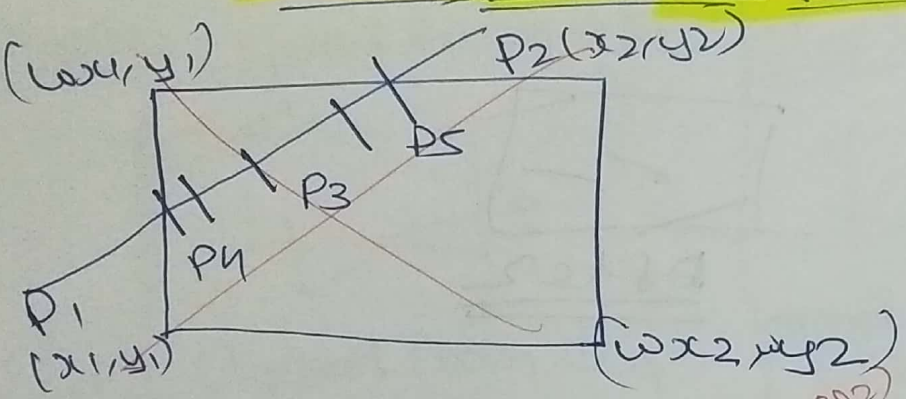
intersecting or partially inside and outside

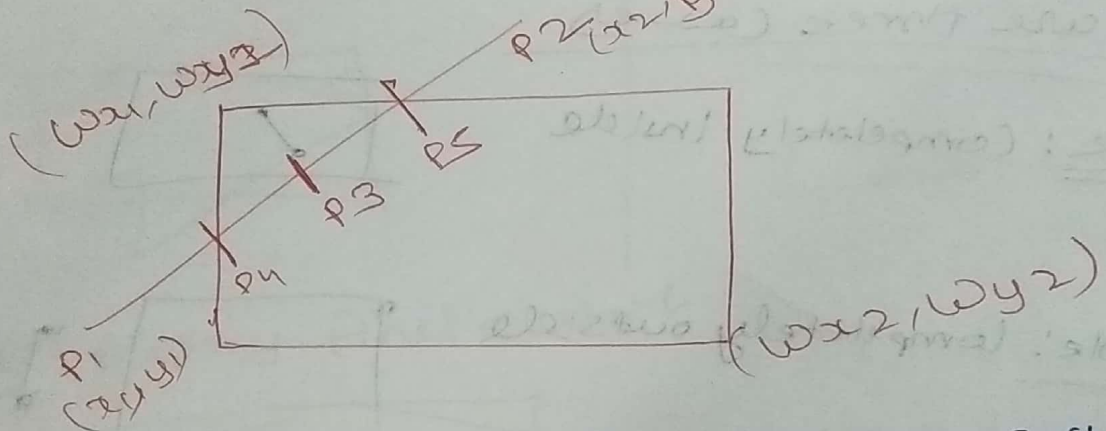


Algorithms for line clipping

- Cohen-Sutherland.
- Cyreius-Beck.
- Midpoint subdivision.
- Liang-Barsky.

Midpoint subdivision Line clipping algorithm





* The Algorithm works in two phases like Cohen Sutherland.

Phase 1: Here phase 1 is exactly same as like Cohen-Sutherland algorithm, where we assign 4 bit region codes to end points of line and find out category of line. Category 3 is passed to phase 2.

Here phase 2 differs. This algorithm handle category 3 lines differently given below.

Phase 2: Category 3 lines are partially visible as they intersect window boundary

Find out mid point of line segment and subdivide into two parts.

Now once again find out category of line to see line segment is visible, invisible or partially visible

i.e If it is category 1, 2 or 3. Repeat this process of subdivision of line and find its category for line segments until we get completely visible or completely invisible

Algorithm

Step1: Read two endpoints of line say $p1(x1, y2)$, & $P_2(x_2, y2)$.

Step2: Read two corners of the window say $(wx1, wy1)$ & $(wx2, wy2)$

Step3: Assign region codes for two endpoints p1 & p2 using steps

Set Bit 1 - if $(x < wx1)$
Set Bit 2 - if $(x > wx2)$
Set Bit 3 - if $(y < wy1)$
Set Bit 4 - if $(y > wx2)$

Step4: Check for visibility of line p1, p2.

Step5: a) If region codes for both the end points p1 & p2 are 0000 then line is completely visible., hence goto step6.

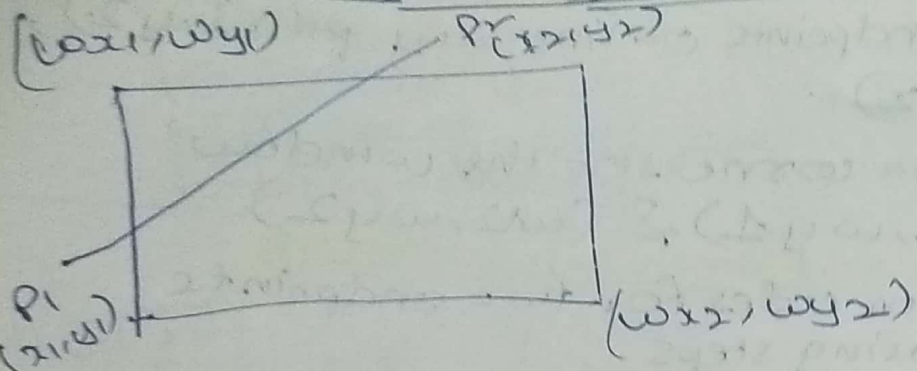
b) If region codes for end points are not 0000 and logical ANDing of them is also not zero then line is compeletly invisible, so reject the line & gotostep6.

c) If region code for two endpoints do not satisfy the conditions in a) & c) b) the line is partially visible. goto step5

Step5: ~~Decide~~ Divide the partially visible segment in equal parts and repeat step3 & step4 & step5. for both subdivided line segment until we get completely visible & invisible line segment.

Step6: Stop.

Cyrus-Beck line clipping Algorithm



There are 3 possibilities for line.

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

Algorithm

Step 1: Read two end points of line say p1(x1, y1) & p2(x2, y2).

Step 2: Read two corners of window say (wx1, wy1) & (wx2, wy2).

Step 3: Calculate D = p2 - p1.

Step 4: Assign boundary point (f) with particular edge.

Step 5: Find inner normal vector for corresponding edge.

Step 6: Calculate $D \cdot n$ and $w = p1 - f$.

Step 7: If $D \cdot n > 0$

$$t_L = \frac{-w \cdot n}{D \cdot n}$$

else

$$t_U = \frac{w \cdot n}{D \cdot n}$$

End if

t is parameter like ($0 <= t <= 1$)

Step8: Repeat step4 to 7 for each edge of clipping window.

Step9: Find maximum lower and upper limit.

Step10: if max upper and lower limit do not satisfy the condition $0 <= t <= 1$ then, ignore the line.

Step11: Calculate the intersection points by substituting values of maximum lower limit and minimum upper limit in parameter eqn of line - p1 p2

$$p(t) = p1 + (p2 - p1)t.$$

Step12: Draw
p(tL) to p(tU)

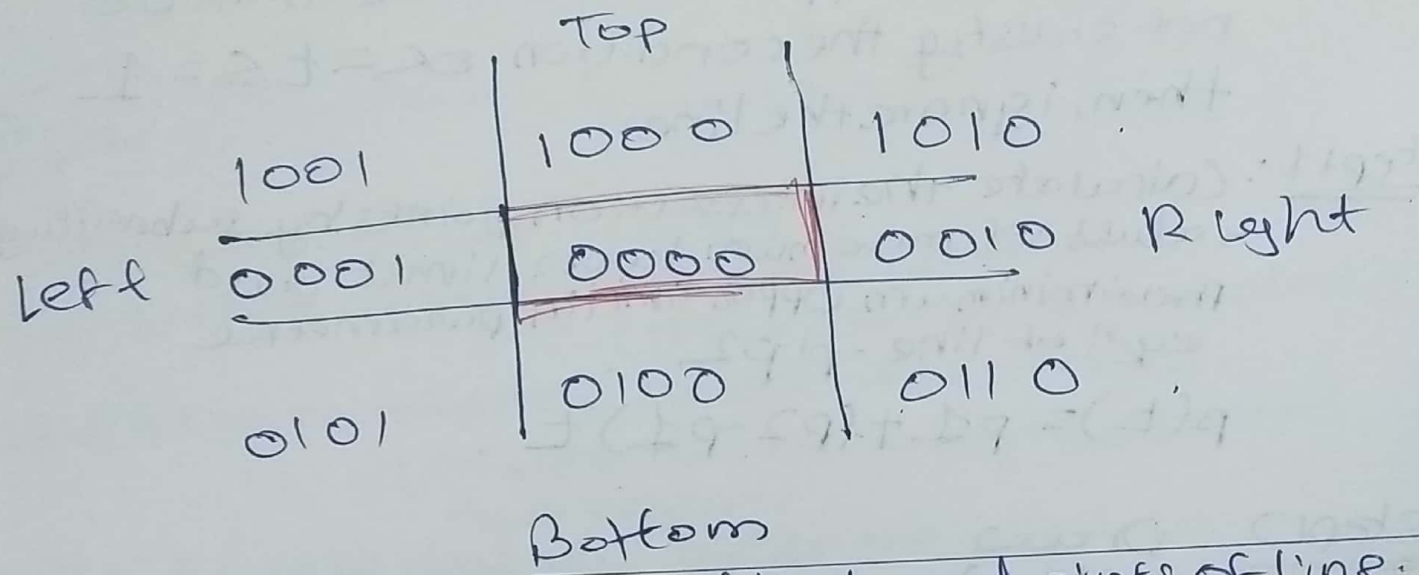
Step13: Stop.

Cohen Sutherland Line Clipping Algorithm:

In this algorithm, we are given 9 regions on the screen.
Out of that 1 is window and rest 8 regions are around it given by 4 bit binary.
The division of regions are based on (x_max, y_max) and (x_min, y_min).
The central part is viewing region or window, all the lines which lie within this region are completely visible.
A region code is always assigned to the endpoints of given line.
To check whether the line is visible or not.

Formula : T B R L

Top Bottom Right left.



Step 1: Calculate positions of both endpoints of line.

Step 2: Perform OR operation on both of these endpoints.

Step 3: If the OR operation gives 0000
then, visible.

Else
Perform AND operation -
If AND ≠ 0000
invisible

If AND = 0000
clipped case.

Step 4: If line is clipped case, find intersection with boundaries of window.

$m = (y_2 - y_1)(x_2 - x_1)$.

Repeat till

Step 5: Stop