Clipping

The primary use of clipping in CG is to remove objects, lines or line segments that are outside the viewing pane (screen). The viewing transformation is insensitive to the position of points relative to the viewing volume - especially those points relative to ~~be to the~~ behind the viewer, and it is necessary to remove these points before generating the view.

For deciding the visible and invisible portion, a particular process called clipping is used. Clipping differentiate each element into visible and invisible portion. Visible portion is selected and invisible portion is discarded.

Clipping can be done through hardware or software

Types of Clipping

1.) Point Clipping
2) Line Clipping
3) Area (Polygon) Clipping
4) Curve Clipping
5) Text Clipping
6) Exterior Clipping

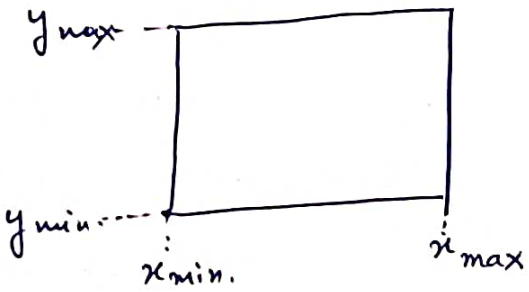
Point Clipping

It is used to determine whether a point lies inside the window or not.

Terminologies

1.) Window → not a portion where objects are displayed.
2.) viewport → which tells us where the objects should be displayed.
3.) viewing pane → screen on which object is displayed.

So, before starting the clipping the first requirement is of defining a window. and the next is of finding a point which lies outside / inside the window. All the points lying outside the window are discarded.
Every window has some co-ordinates which defines its boundaries.
Let's take one window :-



Now for a point (x, y) to be lying inside this window, its x coordinate should lye between $x_{min} \leq x \leq x_{max}$ and its y coordinate should lye between $y_{min} \leq y \leq y_{max}$
So, we have in all four conditions to be satisfied for a point to be lying inside a window.

1.) $x_{min} \leq x$
2.) $x \leq x_{max}$
3) $y_{min} \leq y$
4) $y \leq y_{max}$

Algorithm :-

Step 1 :- Get the minimum and maximum coordinates of both viewing panes.

Step 2 :- Get the coordinates of point.

Step 3 :- Check whether given input lies between minimum and maximum coordinate of viewing pane.

Step 3.1 If yes display the point which lies inside the region otherwise discard it.

Line Clipping

It is a process in which we can cut the part of a line which lies outside the view pane. There are various algorithms to perform line Clipping. like :-

1.) Cohen - Sutherland
2) Liang - Barsky
3) Midpoint - Subdivision

Let's take the Cohen-sutherland algo for line Clipping.

Cohen Sutherland Line Clipping

Named after "Danny Cohen" and "Ivan Sutherland"

1.) In - this algorithm, we will divide the view pane into nine equal segments that only serve the viewport.

Now, we will represent the top, bottom, left and right corner of the window with 4 bits. These 4 bits can be described with the following points that

2.1) If an object lies within any particular corner position, that corner value will be 1, else it will be 0.

2.2) The allocation of the bits depends on TBRL (Top, Bottom, Right, Left) Rule.

Suppose if a point of a line appears in the top-left corner, then according to TBRL rule, the value will be 1001.

In this way, we will check TBRL for each segment and allot the bits accordingly.

1001	1000	1010
0001	0000	0010
0101	0100	0110

In this algo, we will divide the lines into following 2 sections :-

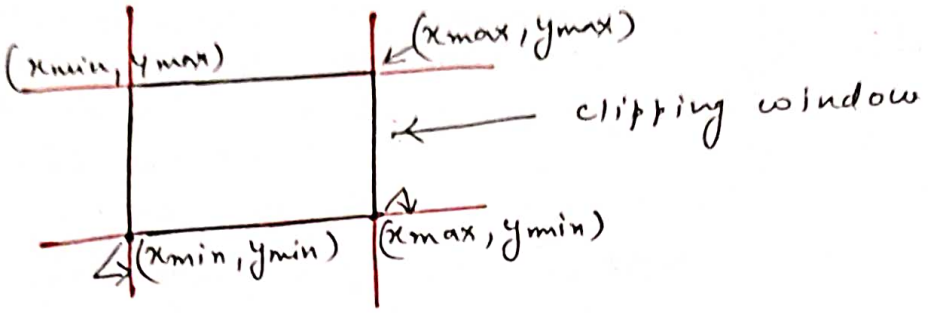
i) Visible lines : when both end points of the lines are entirely situated inside the window.

ii) Invisible lines :- when both points of the line are situated outside the window.

iii) Clipped lines :- If one point of line lies inside the window and other is outside

then the line is called clipped line.

For this also, we need to define the window by (x_{min}, x_{max}) and (y_{min}, y_{max}).



Algo for Cohen - Sutherland Line Clipping

Step 1:- Assign the bit code for both end points of the line.

Step 2:- Now implement "OR" operation on both endpoints of the line.

Step 3:- If the OR = 0000,

then

"The line is visible & acceptable"

else

"Implement "AND" operation on endpoints."

then

if AND ≠ 0000

then.

"The line is not acceptable."

else

AND = 0000

"The line needs to be clipped".

Step 4:- If a line needs to be clipped, first find an intersection point of all boundaries with the following formula:-

$$m = (y_1 - y_0)\ (x_1 - x_0)$$

Step 4.1 When the line intersects the left boundary of window port.

$$y_0 = y_1 + m(x - x_1)$$

here $x = x_{min}$.

here $x = xw_{max}$ (max value of coordinate),

Step 4.2 when the line intersects the right side boundary of window port.

$$y_0 = y_1 + m(x - x_1)$$

here $x = x_{max}$

Step 4.3 · when the line intersects top side boundary of the window port.

$$x_0 = x_1 + (y - y_1)/m$$

here $y = y_{max}$ (max value of y coordinate)

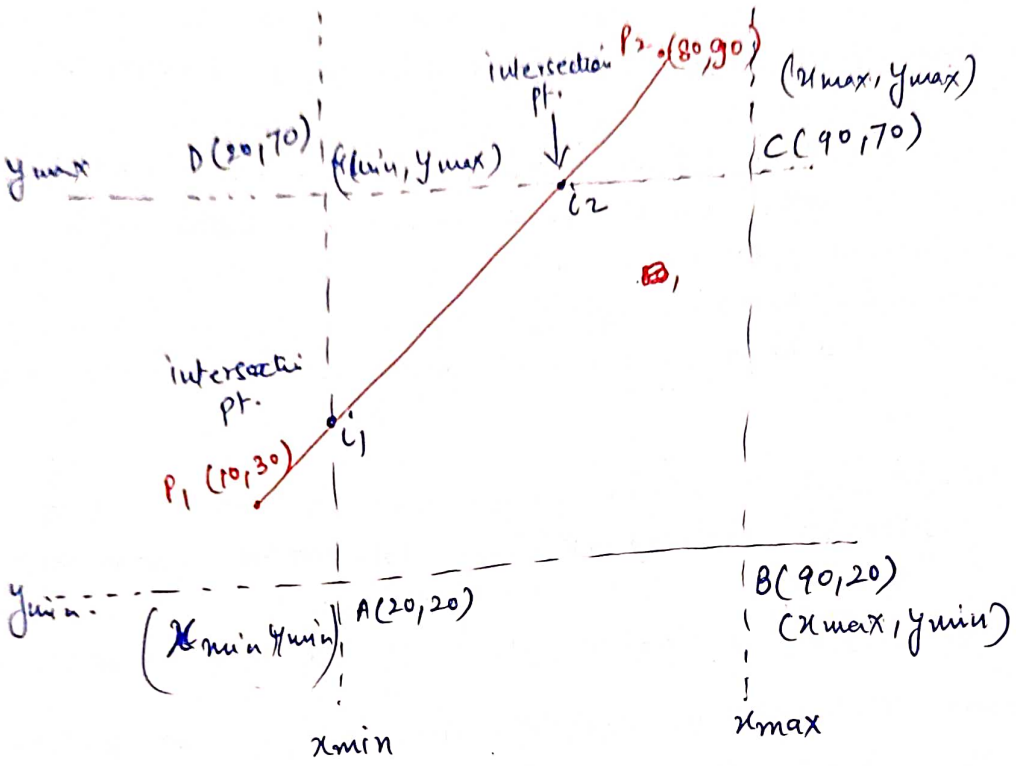
Step 4.4 when the line intersects the bottom side boundary of the window port.

$$x_0 = x_1 + (y - y_1)/m$$

here $y = y_{min}$.

Let us understand this with an example.

Let ABCD be the rectangular window with A(20,20) B(90,20), C(90,70) and D(20,70). Find region codes for the endpoints and use Cohen Sutherland Algo to clip the line P_1P_2 with $P_1(10,30)$ and $P_2(80,90)$.



(i) finding region code of the line segment. (TBRL)

So here, $P_1 = 0001$, $P_2 = 1000$

(ii) Now perform "AND" operation on both the endpoints

$$\begin{array}{r} 0001 \\ 1000 \\ \hline \text{AND} = 0000 \end{array}$$

Since it is 0000, there could be a portion which will be visible on the screen. If it would not have been "0000", we would have discarded the whole line.

(iii) Next, we need to find out the slope of the line with the formula,

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{90-30}{80-10} = \frac{60}{70} = 0.857$$

* Now let's find the value of Ist intersection point, i.e. i_1. As it is lying on the left boundary of the window, we will use the formula —

$$y_0 = y_1 + m(x_{min} - x_1)$$

here we already know, $(x_1, y_1) = (10, 30)$ and $x_{min} = 20$ we need to only find the value of y_0 here, so solving the equation for y_0,

$$y_0 = 30 + 0.857(20 - 10)$$
$$= 30 + 8.57$$
$$= 38.57$$

So now value of $i_1 = (20, 38.57) \simeq (20, 39)$

* Now we need to find the value for IInd intersection point, ~~where~~ as it is intersecting at top boundary, we have the formula,

$$x_0 = x_1 + (y_{max} - y_1)/m$$
$$x_0 = 10 + (70 - 30)/0.857$$
$$= 10 + \frac{40}{0.857}$$
$$x_0 = 56.67$$

So $i_2 = (56.67, 70) \simeq (57, 70)$ // pixels can't be represented in points (fraction).

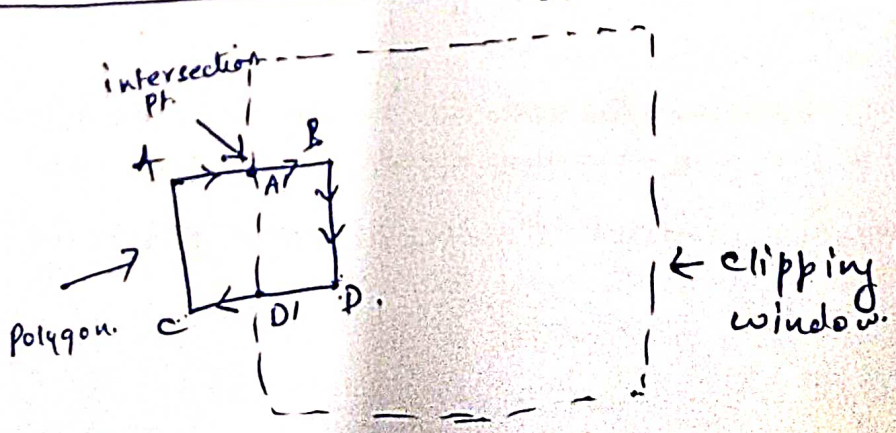
So, now we know both the intersection points which are $i_1 = (20, 39)$ and $i_2 = (57, 70)$. These are the points which show that which part of the line segment will be visible in the window, and which parts will get clipped.

Polygon Clipping

A polygon is an enclosed collection or group of line having a property of closed circuit.

Now polygon clipping is a process in which we consider the part of polygon which is inside the window/ view pane. It has minimum 3 edges.

Sutherland Hodgeman Polygon Clipping Algo.



List → vertex

↓ inside intersec

4 rules.

① out → inside.
A', B. → save in list

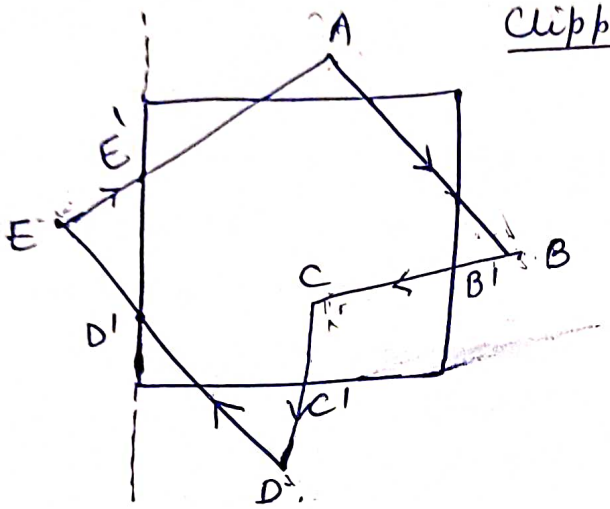
② Lines which are totally inside (both point), then pick its last point and save D

③ in → out
save intersection pt. only. Here it is 'D'.

④ both pts. of lines are outside them, neglect.

To find the new set of vertices we have four rules which we have discussed above.

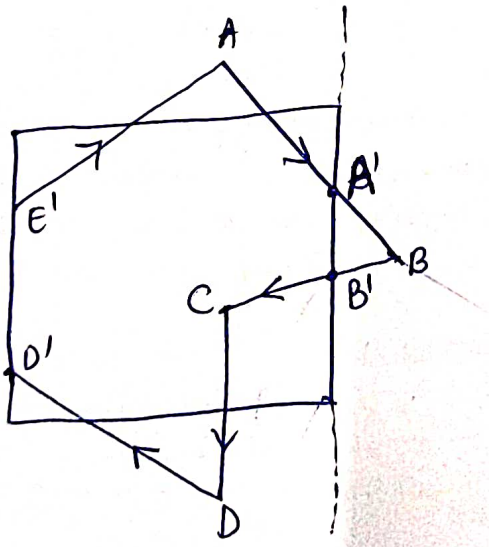
Clipping against left boundary



vertex	case	o/p.
A, B	in → in.	B.
B. C	in → in	C.
C D	in → in	D.
D E	in → out	D'
E A	out → in	E', A.

⇓ resultant

Clipping against Right boundary

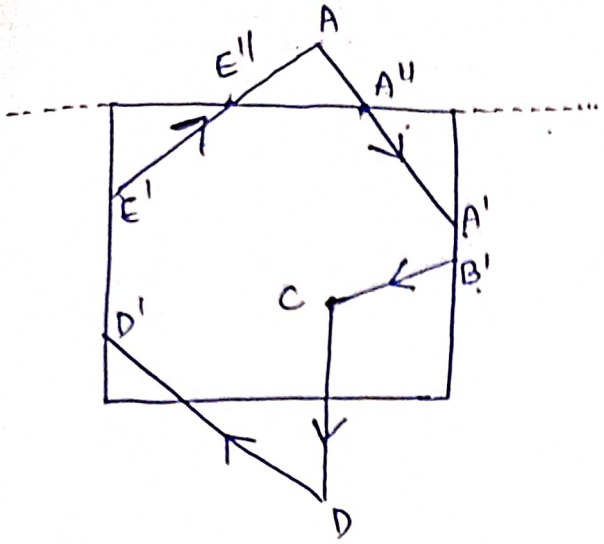


vertex	case	output
A B	in → out	A'
B C	out → in	B', C.
C D	in → in	D
C D'	in → in	D'
E' A	in → in	A

⇓ resultant

continues....

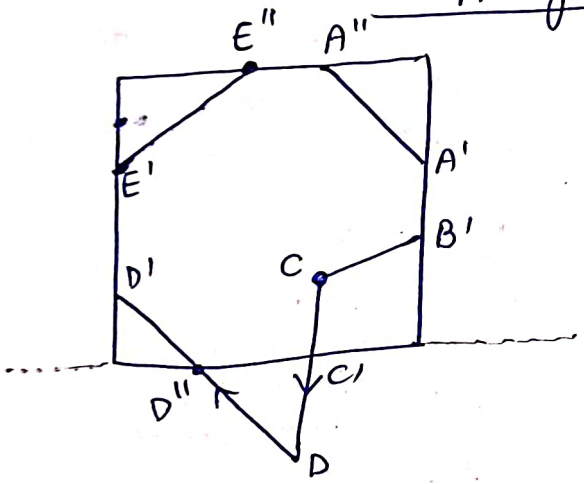
Clipping against Top boundary..



vertex	rule	output
A A'	out → in	A'', A'
B' C	in → in	C
C D	in → in	D
D D'	in → in	D'
E' A	in → out	E''

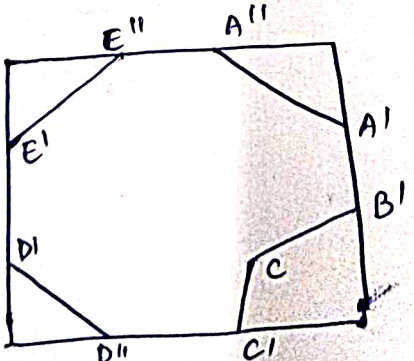
resultant.

Clipping against bottom boundary



vertex	rule	output
D D'	out → in	D'', D'
C D	int → out	C'

resultant



Clipped polygon