**Assignment No.: 05**

Name: Aditya Kangune

Roll No.: 23365

Class: SE11

Batch: G11

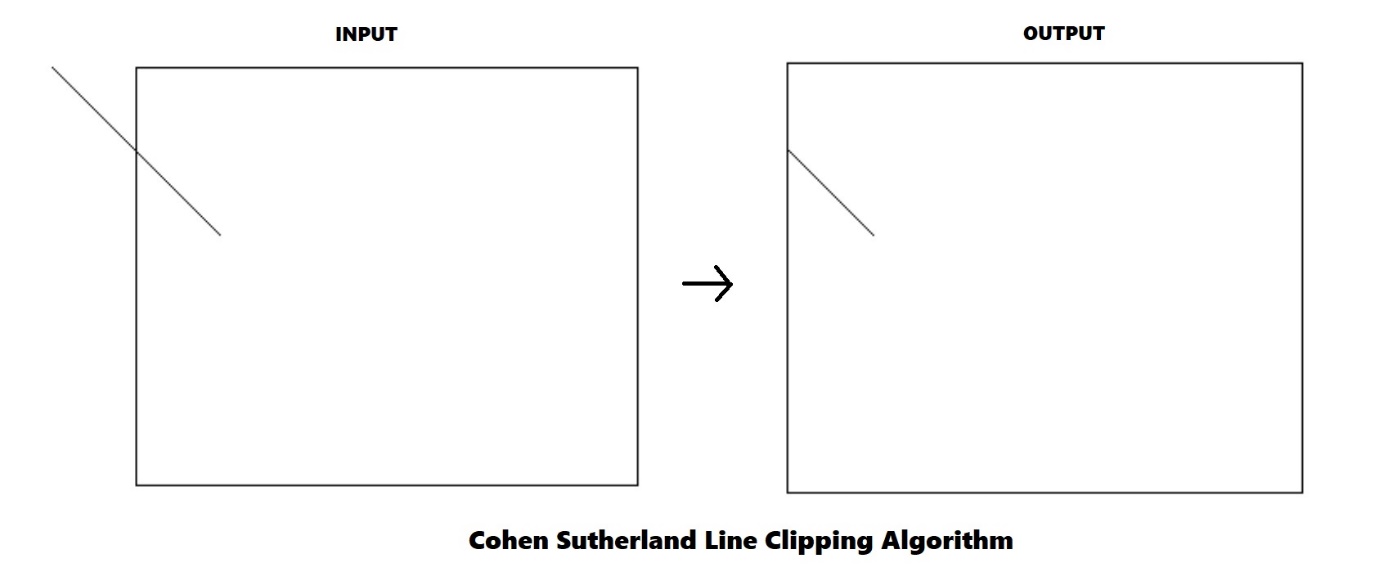
Subject: Computer Graphics Laboratory

**Title:** Line Clipping using Cohen-Sutherland Algorithm.

**Problem Statement:** Implement Cohen-Sutherland line clipping method to clip the polygon with respect the viewport and window. Use mouse click, keyboard interface.

**Theory:**

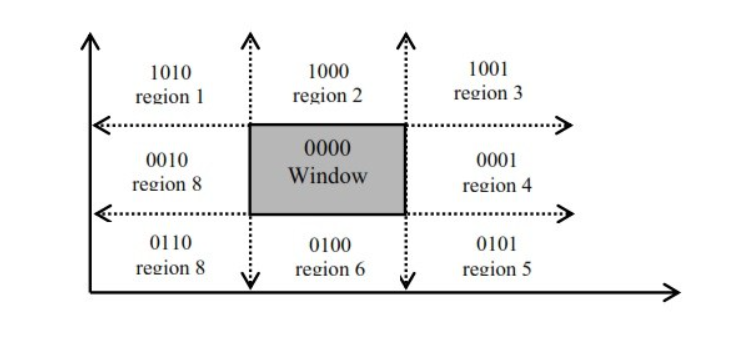
* Cohen Sutherland Algorithm is a line clipping algorithm that cuts lines to portions which are within a rectangular area.
* It eliminates the lines from a given set of lines and rectangle area of interest (view port) which belongs outside the area of interest and clip those lines which are partially inside the area of interest.
* Cohen-Sutherland is a 2D line clipping algorithm. The main advantage of the algorithm is that it vastly reduces the number of line intersections that must be calculated in scan conversion approach.The following figure shows the working of the algorithm.



* The algorithm operates in two phases given as follows:

1. **Region code generation:**

* The algorithm divides a two-dimensional space into 9 regions (eight outside regions and one inside region) and then efficiently determines the lines and portions of lines that are visible in the central region of interest (the viewport).
* Following image illustrates how the space is divided into 9 regios:



* Four Bit code is calculated by comparing extreme end point of given line (x,y) by four co-ordinates x\_min, x\_max, y\_max, y\_min which are the coordinates of the area of interest (0000)
* For any element(x,y) of a line, the outcode is set according to following conditions:

1. Set first bit (most significant bit) if a point lies above window

i.e. y> Ymax

1. Set second bit if a point lies below window

i.e. y<Ymin

1. Set third bit if a point lies to the right of window

i.e. x> Xmax

1. Set fourth bit (least significant bit) if a point lies to left of window

i.e. x < Xmin

* If the point is inside the clipping window, none of the above conditions would be true. So, outcode of endpoint inside the clipping region would be 0000.

1. **Clipping:**

* The algorithm quickly detects two trivial cases.
* If both endpoints of a line lie inside the window, the entire line is visible. It is trivially accepted and needs no clipping.
* On the other hand, if both endpoints ofa line lie completely on one side of the window, the line lies completely outside the window. It is completely rejected.
* The line segment's endpoints are tested to see if the line can be trivally accepted or rejected. If the line cannot be trivally accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted.



* The processing can be divided into three cases as follows:

1. Case 1:

* The line is completely inside clipping window if logical OR operation of outcodes yields to 0000 and the line is trivially accepted.
* In short, if outcodes of both endpoints are 0000, the line is fully visible.

1. Case 2:

* If logical OR of the outcodes is not 0000, there are two possibilities, the line may be partially visible or it may be completely outside.
* The line is completely outside the clipping window if logical AND operation of outcomes does not yield to 0000.
* Above both tests are performed in the same order, i.e. Case 2 is applicable only if Case 1 is not true.

1. Case 3:

* If the line is not trivially accepted or rejected, then compare endpoints with window boundary to determinehow much line segment can be discarded. The intersection point of line with clipping region edge can be computed as follows:
  + Consider the line with endpoints(x1, y1) and (x2,y2). The slope of a line is given as,

m=(y2-y1)/(x2-x1)

* + Using explicit representation of the line, y = mx+c, we can compute Y-intercept of the line as, c =y-mx, where (x, y) is any point on the line. Both endpoints of the line are known, so we can put anyone in this equation to compute the value of c,

let's put (x1,y1), so c =y1 - mx1

* + The Y-coordinate of an intersection with a vertical window boundary can be calculated as y = mx + c,where x is either Xmin or Xmax,depends on for which vertical line we are calculating Y.

Y = mXmin+c (for intersection with left edge)

Or

Y = mXmax+c (for intersection with right edge)

* + Similarly, X-coordinate of an intersection with a horizontal window boundary is computed as,

X = (y-c)/m

Where y is either Ymin or Ymax, depends on for which horizontal

line we are caleulating X.

X = (Ymin-c)/m (for intersection with bottom edge)

Or

X = (Ymax+c)/m (for intersection with top edge)

* This process is repeated until we find the entire line segment within the window.
* **Advantages Cohen-Sutherland algorithm:**
* It is easy to understand.
* Simple to implement.
* Best suitable for the lines fully inside or outside.
* It can easily be extended for 3D line clipping.
* **Limitations Cohen-Sutherland algorithm:**
* Repeated clipping is expensive.
* Only applicable to rectangular clipping window. It cannot handle any other shape.
* It can be improved using more regions (e.g., Nichol Lee Nichol approach).

**Algorithm:**

* Cohen-Sutherland algorithm for line clipping:

1. **Step1:** Read (Xmin ,Ymin) and (Xmax ,Ymax)-Lower-left and top-right corner of clipping window.
2. **Step 2:** Read A(x. y1) and B(x2. Y2) end points of line.
3. **Step 3:** Compute outcode of A and B

if y1 > Ymax then Outcode\_A(1) =1 else 0

if y2 > Ymax then Outcode\_B(1) = 1 else 0

if y1 < Ymin then Outcode\_A(2) = 1 else 0

if y2 < Ymin then Outcode\_B(2) = 1 else 0

if x1 > Xmax then Outcode\_A(3) = 1 else0

if x2 > Xmax then Outcode\_B(3) = 1 else 0

if x1 < Xmin then Outcode\_A(4) = 1 else0

if x2 < Xmin then Outcode\_B(4) = 1 else 0

1. **Step 4:** if Outcode\_A **OR** Outcode\_B == 0000 then

Display entire line and goto step 5

else if Outcode\_A **AND** Outcode\_B != 0000 then

Reject the entire line

else

Compute the intersection point with window boundaries

Repeat Step 4.

1. **Step 5:** Stop

**Input:**

**Output:**

**Conclusion:**

1. The Cohen-Sutherland algorithm for line clipping is implemented successfully.
2. It was implemented using OpenGL library by mouse click and keyboard interface.