**Problem Statement:**

Implement Line Clipping Algorithm using Nicholl–Lee–Nicholl algorithm. Share the Code and time complexity achieved with neat documentation. From Scratch implementation with C++ is expected. Share the Input image and output Image.

**SOLUTION:**

**NLN**- is another algorithm in Line clipping which has 3 possible positions for Line endpoint P1 such as

1. in window region

2. in edge region

3. in corner region.

According to where the point lies we need to find out the intersection and then draw a vertex where point lies, so that we can find intersection points.

Name the intersection point for reference and check intersection points lies between the points p1 &p2 also check which region it falls that is [LRBT] Left, Right, Bottom, Top.

Areas are then designated as L, LT, LB, or TR, depending on the location of the initial point.

Then the other end point of the line is checked against these areas.

If a line starts in the L area and finishes in the LT area, then the algorithm concludes that the line should be clipped at xw (max).

Thus the number of clipping points is reduced to one, compared to other algorithms that may require two or more clipping.

Possible cases are there:

CASE1: In inside case, the areas are named T for top, L for left, R for right, and B for bottom.

CASE2: In left case, the areas are named L for left, TL for top and left, LR for left and right, and TB for top and bottom.

CASE3: In top-left case, the areas are named L for left, T for top, TR for top and right, LB for left and bottom, and TB for top and bottom or LR for left and right.

The last step is just locating the end point of the line that wants to be clipped, and clipped it according to the name of that area against the side(s) of the clipping window.

**Source code:**

#include <graphics’>

#include <iostream>

#include <utility>

#include <vector>

using namespace std;

using namespace sf;

int xmin,ymin,xmax,ymax;

int main()

{

int x1,y1,x2,y2;

int gdriver = DETECT, gmode, errorcode;

int findRegionP1(int,int);

void clipline1(int,int,int,int);

void clipline2(int,int,int,int);

void clipline3(int,int,int,int);

int ch;

float m;

printf("\nEnter the xmin:->");

scanf("%d",&xmin);

printf("\nEnter the ymin:->");

scanf("%d",&ymin);

printf("\nEnter the xmax:->");

scanf("%d",&xmax);

printf("\nEnter the ymax:->");

scanf("%d",&ymax);

initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

setcolor(15);

rectangle(xmin,ymin,xmax,ymax);

printf("Enter the x1:->");

scanf("%d",&x1);

printf("Enter the y1:->");

scanf("%d",&y1);

printf("Enter the x2:->");

scanf("%d",&x2);

printf("Enter the y2:->");

scanf("%d",&y2);

setcolor(12);

line(x1,y1,xmin,ymin);

line(x1,y1,xmax,ymin);

line(x1,y1,xmax,ymax);

line(x1,y1,xmin,ymax);

getch();

setcolor(3);

line(x1,y1,x2,y2);

getch();

ch=first\_end\_point\_region(x1,y1);

switch(ch)

{

case 1:

clipline1(x1,y1,x2,y2);

break;

case 2:

clipline2(x1,y1,x2,y2);

break;

case 3:

clipline3(x1,y1,x2,y2);

break;

default:

printf("\nInvalid option: ");

}

getch();

}

int first\_end\_point\_region(int x,int y)

{

/\* u have two equations:- xmin <= x <= xmax; ymin <= y <= ymax; \*/if(x>=xmin && x<=xmax && y>=ymin && y<=ymax)

return 1;

elseif(x<xmin && y>=ymin && y<=ymax)

return 2;

elseif(x<=xmin && y<=ymin)

return 3;

elsereturn 0;

}

/\* point p1 is inside the clip window \*/void clipline1(int x1,int y1,int x2,int y2)

{

int draw=1;

float m,m1,m2,m3,m4;

int nx1,ny1,nx2,ny2;

/\* calculate slopes for all the lines passing thru vertices and including the input line :- \*/

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

m1=((float)(ymin-y1))/(xmin-x1);

m2=((float)(ymin-y1))/(xmax-x1);

m3=((float)(ymax-y1))/(xmax-x1);

m4=((float)(ymax-y1))/(xmin-x1);

nx1=x1;

ny1=y1;

// point p2 is on topif(((abs(m)>=m1 && x2<x1) || (abs(m)>abs(m2) && x2>x1)) && y1>y2)

{

// point p2 is also inside clip windowif(y2>ymin)

{

nx2=x2;

ny2=y2;

}

// point p2 is outside clip windowelse

{

ny2=ymin;

nx2=x1+(ymin-y1)/m;

}

}

// point p2 is on right side of clip windowelseif(m>m2 && m<m3 && x2>=xmax)

{

// point p2 is inside clip windowif(x2<xmax)

{

nx2=x2;

ny2=y2;

}

// point p2 is outside clip windowelse

{

nx2=xmax;

ny2=y1+(xmax-x1)\*m;

}

}

// point p2 is on bottom side of clip windowelseif((abs(m)>=m3 && x2>x1) || (abs(m)>abs(m4) && x2<x1))

{

// point p2 is inside clip windowif(y2<ymax)

{

nx2=x2;

ny2=y2;

}

// point p2 is outside clip windowelse

{

ny2=ymax;

nx2=x1+(ymax-y1)/m;

}

}

// point p2 is on left side of clip windowelseif(m>m4 && m<m1)

{

// point p2 is inside the clip windowif(x2>xmin)

{

nx2=x2;

ny2=y2;

}

// point p2 is outside the clip windowelse

{

nx2=xmin;

ny2=y1+(xmin-x1)\*m;

}

}

else

draw=0;

cleardevice();

setcolor(18);

rectangle(xmin,ymin,xmax,ymax);

if(draw)

{

setcolor(12);

line(x1,y1,xmin,ymin);

line(x1,y1,xmax,ymin);

line(x1,y1,xmax,ymax);

line(x1,y1,xmin,ymax);

setcolor(5);

line(nx1,ny1,nx2,ny2);

}

} return 0; }

Output:

