Assignment 5

(By Shashank Kapadnis 23310)

Aim:

Implement Cohen Sutherland polygon clipping method to clip the polygon with respect the viewport and window. Use mouse click, keyboard interface

Theory:

Sutherland-Hodgeman Polygon Clipping is performed by processing the boundary of polygon against each window corner or edge. First of all entire polygon is clipped against one edge, then resulting polygon is considered, then the polygon is considered against the second edge, so on for all four edges.

Four possible situations while processing

1. If the first vertex is an outside the window, the second vertex is inside the window. Then second vertex is added to the output list. The point of intersection of window boundary and polygon side (edge) is also added to the output line.
2. If both vertexes are inside window boundary. Then only second vertex is added to the output list.
3. If the first vertex is inside the window and second is an outside window. The edge which intersects with window is added to output list.
4. If both vertices are the outside window, then nothing is added to output list.

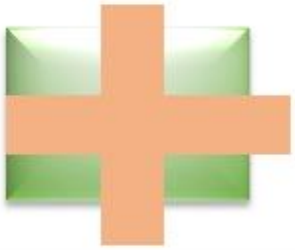
Following figures shows original polygon and clipping of polygon against four windows.

Algorithm:

Original polygon

Clipping against left edge of window

Clipping against bottom corner of window



Clipping against Top of window

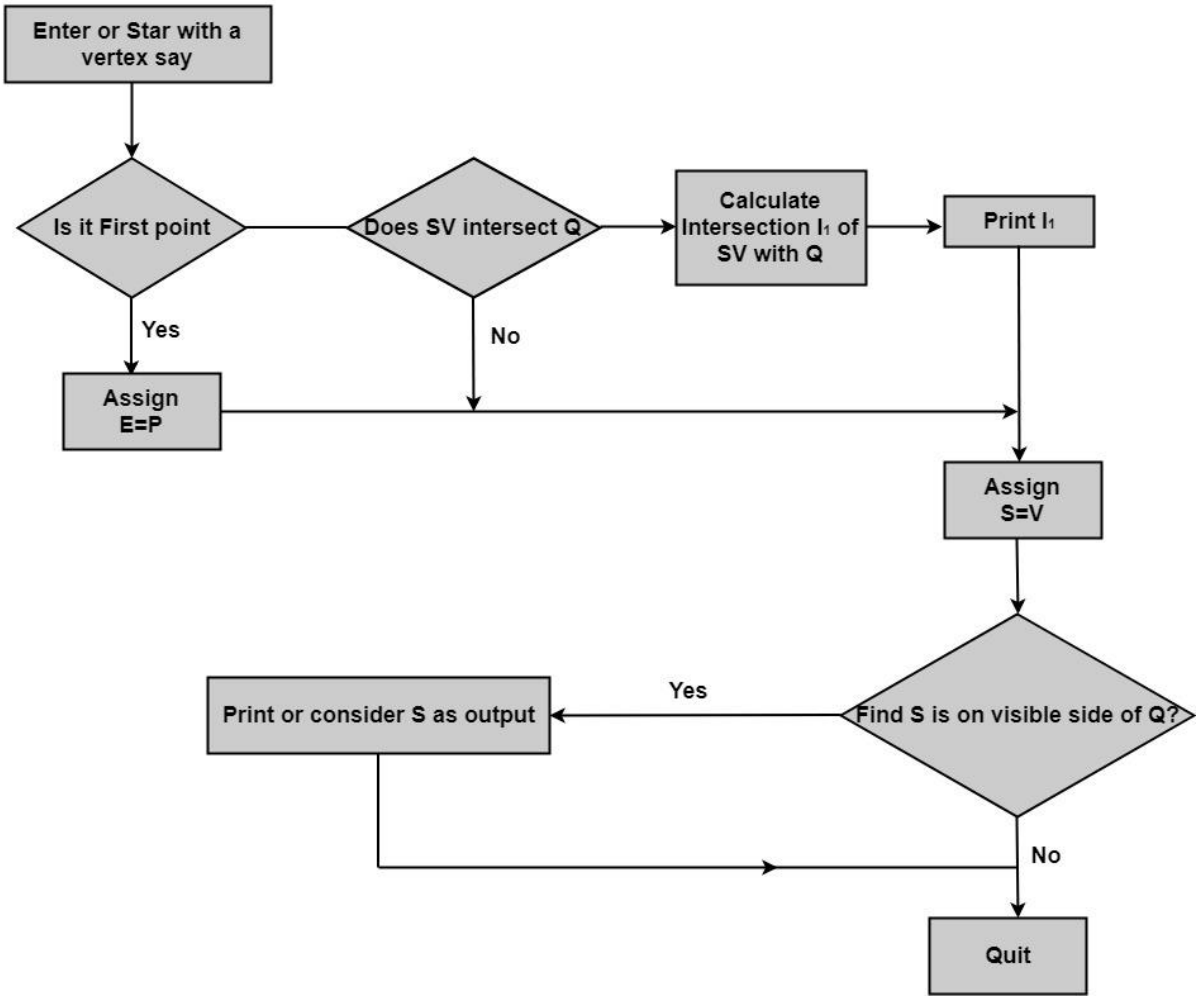
Clipping against right edge of window

Final polygon after Clipping

Disadvantage of Cohen Hodgmen Algorithm:

This method requires a considerable amount of memory. The first of all polygons are stored in original form. Then clipping against left edge done and output is stored. Then clipping against right edge done, then top edge. Finally, the bottom edge is clipped. Results of all these operations are stored in memory. So wastage of memory for storing intermediate polygons.

Sutherland Hodgemen Algorithm:



Code :

```
#include<stdio.h>        //initial inclusions
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<math.h>
float xd1,yd1,xd2,yd2; //storing values for end points of line
int ymax=100; //initializing window coordinates
int ymin=-100;
int xmax=100;
int xmin=-100;
static int p=0;
void disp(); //declaring display function
```

```
float round_value(float v) //function to round value to next greater float
{
	return (v+0.5);
}
void plotpoint(float a,float b)
{
	glBegin(GL_POINTS);
	glVertex2f(a,b);
	glEnd();
}
void dda(float X1,float Y1,float X2,float Y2) //dda algorithm
{
	/*
	 * Input : Initial and final co-ordinates of line points.
	 * Utility : plot line using Digital Differential Analyzer
	 * Output : Line on initialized window.
	 */
	float dx,dy,x,y,xinc,yinc;//initializations
	int k,steps;
	dx=X2-X1;				//difference of x coordinates
	dy=Y2-Y1;				//difference of y coordinates
	steps=abs(dx)>abs(dy)?abs(dx):abs(dy); //calculation of number of steps
	xinc=dx/(float)steps; //value for incrementing x
	yinc=dy/(float)steps; //value for incrementing y
	x=X1,y=Y1;
	plotpoint(x,y);		//function to plot point on window
	for(k=0;k<steps;k++) //loop to plot points
	{
		x+=xinc;			//incrementing x by xinc
		y+=yinc;			//incrementing y by yinc
```

```
                plotpoint(round_value(x),round_value(y)); //plotting point
        }
        glFlush();
}
int code(int x,int y)
{
        /*
         * Input : x and y coordinates of the point.
         * Utility : Determine outcode  for given point.
         * Output : Out code.
         */
        int c=0;
        if(y>ymax) c=8;          //if greater than ymax set code to 8
        if(y<ymin) c=4;  //if less than ymin set code to 4
        if(x>xmax) c=c|2;        //if greater than xmax set code to 2
        if(x<xmin) c=c|1;        //if less than ymin set code to 1
        return c;
}
void cohen(float x1,float y1,float x2,float y2)   //implementing cohen-sutherland algorithm
{
        int c1=code(x1,y1);                  //checking for outcode of point 1
        int c2=code(x2,y2);                          //checking for outcode of point 2
        float m=(y2-y1)/(x2-x1);             //checking slope of line
        while((c1|c2)>0)                             //iterating loop till c1|c2>0
        {
                if((c1 & c2)>0)              //if both lie completely outside the window
                {
                        disp();
                        return;
                }
```

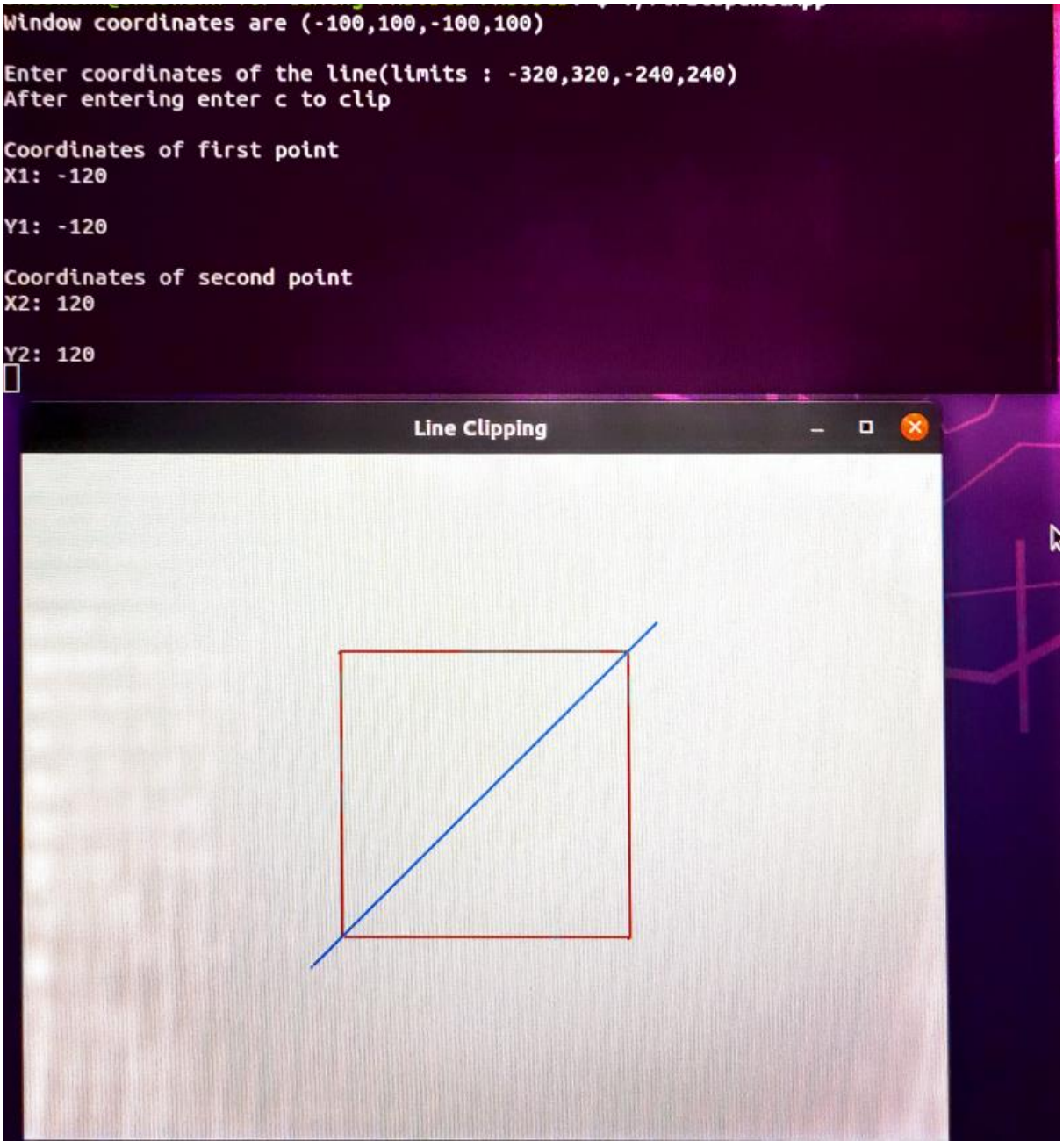
```
int c;
float xi=x1;
float yi=y1;
c=c1;
float x,y;
if(c==0)                                    //checking if outcode is equal to 0
{
        c=c2;                               //assigning outcode of c2
        xi=x2;                              //assigning x coordinate of
c2
        yi=y2;                              //assigning y coordinate of
c2
}
if((c & 8)>0)                               //checking if  c&8 >0 ( greater than
ymax)
{
        y=ymax;                             //assigning      new
values to x and y
        x=xi+1.0/(m*(ymax-yi));
}
 if((c & 4)>0)                              //checking  if  c>  4  >0  (less  than
ymin)
{
        y=ymin;                             //assigning      new
values to x and y
        x=xi+1.0/(m*(ymin-yi));
}
 if((c & 2)>0)                              //checking if  c&2 >0 ( greater than
xmax)
{
        x=xmax;
```

```
                y=yi+m*(xmax-xi);
        }
        if((c & 1)>0)                          //checking if  c&1 >0 (less than
xmin)
        {
                x=xmin;
                y=yi+m*(xmin-xi);
        }
        if(c==c1)                              //checking code and
assigning new values
        {
                xd1=x;
                yd1=y;
                c1=code(xd1,yd1);
        }
        if(c==c2)                              //checking code and
assigning new values
        {
                xd2=x;
                yd2=y;
                c2=code(xd2,yd2);
        }
    }
    p++;
    disp();                                    //calling display function
again to display new line
}
void mykey(unsigned char ch,int x,int y)
{
    if(ch=='c')
    {
```

```
                cohen(xd1,yd1,xd2,yd2);                    //if character c is pressed calling algorithm
                glFlush();

        }
}
void disp()
{
        glClear(GL_COLOR_BUFFER_BIT);//clearing buffer
        glColor3f(1.0,0.0,0.0);                    //assigning color
        dda(xmin,ymin,xmax,ymin);                  //creating window using dda algorithm to draw lines
        dda(xmax,ymin,xmax,ymax);
        dda(xmax,ymax,xmin,ymax);
        dda(xmin,ymax,xmin,ymin);

        glColor3f(0.0,0.0,1.0);                    //assigning color for line
        dda(xd1,yd1,xd2,yd2);                      //drawing line
        glFlush();

}
void init()
{
        glClearColor(1.0,1.0,1.0,0);          //clearing background color to new color
        glClear(GL_COLOR_BUFFER_BIT);              //clearing buffer
        glPointSize(2);                                //assigning point size
        gluOrtho2D(-320,320,-240,240);

        glFlush();

}
```

```
int main(int argc,char **argv)
{
        printf("Window coordinates are (-100,100,-100,100)\n");
        printf("\nEnter coordinates of the line(limits : -320,320,-240,240) \nAfter entering enter
c to clip\n");
        printf("\nCoordinates of first point");
        printf("\nX1: ");
        scanf("%f",&xd1);                                   //accepting value of x1
        printf("\nY1: ");                                   //accepting value of y1
        scanf("%f",&yd1);
        printf("\nCoordinates of second point");
        printf("\nX2: ");
        scanf("%f",&xd2);                                   //accepting value of x2
        printf("\nY2: ");                                   //accepting value of y2
        scanf("%f",&yd2);

        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowPosition(100,100);
        glutInitWindowSize(640,480);
        glutCreateWindow("Line Clipping");
        init();
        glutDisplayFunc(disp);
        glutKeyboardFunc(mykey);
        glutMainLoop();
        return 0;
}
```

Output :



Conclusion:

23310

Successfully implemented Cohen Sutherland polygon clipping algorithm.