Experiment 8

Student Name: Gaganjot Singh **UID: 22BCS14843**
Branch: B.E. CSE III Yr **Section:** 22BCS-IOT-612-B
Semester: 6th
Subject Name: Computer Graphics with Lab **Subject Code:** 22CSH-352

1. **Aim:** Apply the Cohen-Sutherland Line Clipping algorithm to clip a line intersecting at:
   - **i.** one point with a given window.
   - **ii.** two or more points with a given window.

2. **Objective:** To clip a line intersecting at a single point and two or more points with a window using the Cohen-Sutherland Line Clipping algorithm.

3. **Code:**

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <graphics.h>

void main()

{
    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    outtextxy(150, 90, "Name: Gaganjot Singh | Roll No: 22BCS14843");

    int i, xmax, ymax, xmin, ymin, x1, y1, x2, y2, m;
    int start[4], end[4], code[4];

    initgraph(&gd, &gm, "");

    // viewport coordinates
    xmin = 100; // down left x
    ymin = 100; // down left y
    xmax = 300; // top right x
    ymax = 300; // top right y

    printf("Viewport coordinates:\n");
    printf("Bottom-left: (%d, %d)\n", xmin, ymin);
    printf("Top-right: (%d, %d)\n", xmax, ymax);

    // line coordinates from user
    printf("\nEnter the coordinates for starting point of line: ");
    scanf("%d %d", &x1, &y1);
    printf("\nEnter the coordinates for ending point of line: ");
```

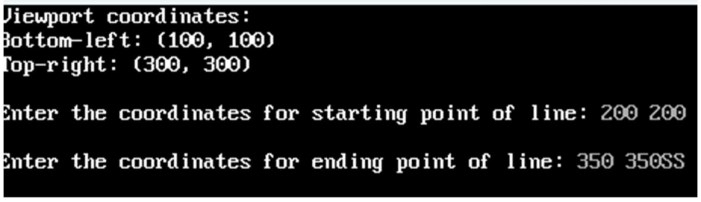
```
    scanf("%d %d", &x2, &y2);

    // init start and end codes
    for (i = 0; i < 4; i++)
    {
        start[i] = 0;
        end[i] = 0;}

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

    // TBRL codes
    if (x1 < xmin)
        start[0] = 1; // left
    if (x1 > xmax)
        start[1] = 1; // right
    if (y1 > ymax)
        start[2] = 1; // top
    if (y1 < ymin)
        start[3] = 1; // down

    if (x2 < xmin)
        end[0] = 1; // left
    if (x2 > xmax)
        end[1] = 1; // right
    if (y2 > ymax)
        end[2] = 1; // top
    if (y2 < ymin)
        end[3] = 1; // down

    for (i = 0; i < 4; i++)
    {
        code[i] = start[i] && end[i];}

    // Check visibility
    if ((code[0]==0) && (code[1]==0) && (code[2]==0) && (code[3]==0))
    {
        if ((start[0]==0) && (start[1]==0) && (start[2]==0) && (start[3]==0)
&
            (end[0]==0) && (end[1]==0) && (end[2]==0) && (end[3]==0)){

            cleardevice();
            printf("\n\t\tThe line is totally visible\n\t\tand not a clipping
candidate");
            rectangle(xmin, ymin, xmax, ymax);
            line(x1, y1, x2, y2);
            getch();}
        else
        {
            cleardevice();
            printf("\n\t\tLine is partially visible");
            rectangle(xmin, ymin, xmax, ymax);
            line(x1, y1, x2, y2);
            getch();

            // clipping logic
```

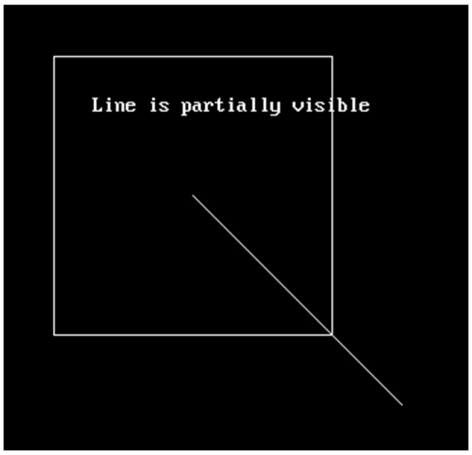
```
if ((start[2] == 0) && (start[3] == 1))
{
    x1 = x1 + (ymin - y1) / m;
    y1 = ymin;
}
if ((end[2] == 0) && (end[3] == 1))
{
    x2 = x2 + (ymin - y2) / m;
    y2 = ymin;}
if ((start[2] == 1) && (start[3] == 0))
{
    x1 = x1 + (ymax - y1) / m;
    y1 = ymax;}
if ((end[2] == 1) && (end[3] == 0))
{
    x2 = x2 + (ymax - y2) / m;
    y2 = ymax;}
if ((start[1] == 0) && (start[0] == 1))
{
    y1 = y1 + m * (xmin - x1);
    x1 = xmin;}
if ((end[1] == 0) && (end[0] == 1))
{
    y2 = y2 + m * (xmin - x2);
    x2 = xmin;}
if ((start[1] == 1) && (start[0] == 0))
{
    y1 = y1 + m * (xmax - x1);
    x1 = xmax;}
if ((end[1] == 1) && (end[0] == 0))
{
    y2 = y2 + m * (xmax - x2);
    x2 = xmax;}
```

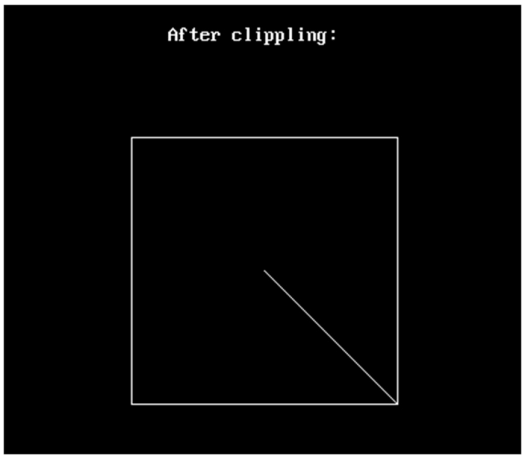
```
        clrscr();
        cleardevice();
        printf("\n\t\tAfter clippling:");
        rectangle(xmin, ymin, xmax, ymax);
        line(x1, y1, x2, y2);
        getch();}}
else
{
    clrscr();
    cleardevice();
    printf("\nLine is invisible");
    rectangle(xmin, ymin, xmax, ymax);}
getch();
closegraph();}
```

4. **Output: Viewport (100,100,300,300)**

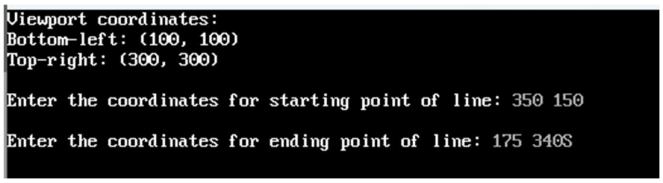
a) Single Point Intersection (200 200 350 350)

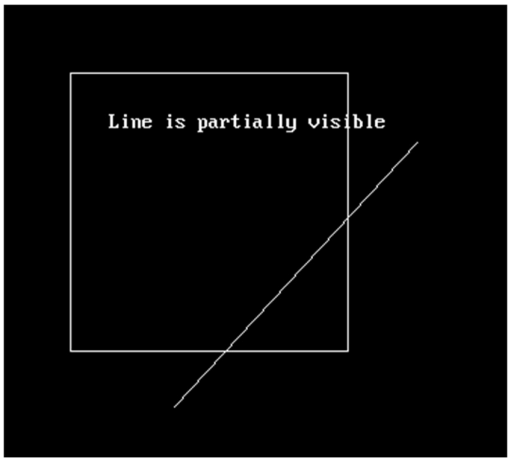


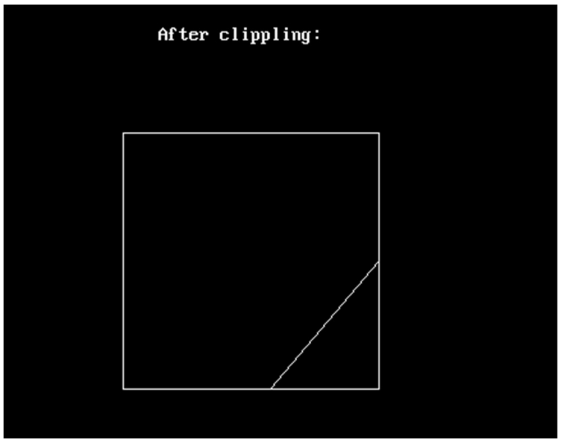




b) Multi-Point Intersection (350 150 175 340)







5. Learning Outcome:

i. By implementing the Cohen-Sutherland line clipping algorithm, I learned how to determine the visibility of a line segment relative to a defined rectangular viewport, allowing for efficient rendering in computer graphics.

ii. The code provided insight into how to calculate and utilize region codes to classify the endpoints of a line segment, which is essential for determining whether the line is completely visible, completely outside, or partially within the clipping window.

iii. I gained an understanding of how to handle edge cases in line clipping, such as when a line intersects the clipping window at multiple points, and how to adjust the endpoints accordingly to ensure accurate rendering.

iv. The implementation highlighted the importance of using integer arithmetic for slope calculations and coordinate adjustments, which can help avoid issues with floating-point precision in graphical applications.