

Department of CSE

SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VII

29 September 2021

UCS 1712 - Graphics And Multimedia Lab

Exercise 7: Cohen-Sutherland Line Clipping in C++ Using OpenGL

Aim:

Apply Cohen-Sutherland line clipping on a line $(x_1, y_1)(x_2, y_2)$ with respect to a clipping window $(X_{Wmin}, Y_{Wmin})(X_{Wmax}, Y_{Wmax})$.

After clipping with respect to an edge, display the line segment with the calculated intermediate intersection points and the vertex list.

Input: The clipping window co-ordinates and the line endpoints.

Note: The output should show the clipping window and the line to be clipped in different colors.

You can show the intermediate steps using time delay.

Code: Cohen-Sutherland Algorithm:

```
1  /*
2  To perform the Cohen-Sutherland Line Clipping Algorithm on a given line,
3  based upon a rectangular clipping window
4  */
5
6  #include <stdio.h>
7  #include <math.h>
8  #include <GL/glut.h>
9  #include <iostream>
10 #include <cstring>
11
12 using namespace std;
13
14 const int LeftBit = 0x1;
15 const int RightBit = 0x2;
16 const int BottomBit = 0x4;
17 const int TopBit = 0x8;
18
19 const int WINDOW_WIDTH = 800;
20 const int WINDOW_HEIGHT = 800;
21 const int FPS = 60;
22
23 class Point
24 {
25 private:
26     GLfloat x, y;
27
28 public:
29     Point()
30     {
31         x = y = 0;
32     }
33
34     Point(GLfloat X, GLfloat Y)
35     {
36         x = X;
37         y = Y;
38     }
39
40     GLfloat getX()
41     {
42         return x;
43     }
44
45     GLfloat getY()
46     {
47         return y;
```

```

48     }
49
50     void setX(GLfloat X)
51     {
52         x = X;
53     }
54
55     void setY(GLfloat Y)
56     {
57         y = Y;
58     }
59
60     int encode(Point windowMin, Point windowMax)
61     {
62         int RC = 0x00;
63
64         if (x < windowMin.getX())
65         {
66             RC = RC | LeftBit;
67         }
68         if (x > windowMax.getX())
69         {
70             RC = RC | RightBit;
71         }
72         if (y < windowMin.getY())
73         {
74             RC = RC | BottomBit;
75         }
76         if (y > windowMax.getY())
77         {
78             RC = RC | TopBit;
79         }
80
81         return RC;
82     }
83 };
84
85 class Line
86 {
87 private:
88     Point p, q;
89
90 public:
91     Line()
92     {
93         p.setX(0); p.setY(0);
94         q.setX(0); q.setY(0);
95     }
96
97     Line(float x1, float y1, float x2, float y2)
98     {

```

```

99         p.setX(x1); p.setY(y1);
100        q.setX(x2); q.setY(y2);
101    }
102
103    Point getP()
104    {
105        return p;
106    }
107
108    Point getQ()
109    {
110        return q;
111    }
112
113    void setP(float x, float y)
114    {
115        p.setX(x); p.setY(y);
116    }
117
118    void setQ(float x, float y)
119    {
120        q.setX(x); q.setY(y);
121    }
122
123    int isInside(int RC)
124    {
125        return !RC;
126    }
127
128    int trivialReject(int RC1, int RC2)
129    {
130        return (RC1 & RC2);
131    }
132
133    int trivialAccept(int RC1, int RC2)
134    {
135        return (!(RC1 | RC2));
136    }
137
138    void swapPoints()
139    {
140        Point x;
141        x.setX(p.getX()); x.setY(p.getY());
142        p.setX(q.getX()); p.setY(q.getY());
143        q.setX(x.getX()); q.setY(x.getY());
144    }
145
146    bool lineClipCohenSutherland(Point windowMin, Point windowMax)
147    {
148        int RC1, RC2;
149        int plotLine = false, done = false;

```

```

150     GLfloat m;
151
152     while (!done)
153     {
154         RC1 = p.encode(windowMin, windowMax);
155         RC2 = q.encode(windowMin, windowMax);
156
157         //cout << "RC1: " << RC1 << "RC2: " << RC2 << endl;
158
159         if (trivialAccept(RC1, RC2))
160         {
161             //Line coordinates are inside boundary
162             done = true;
163             continue;
164         }
165
166         if (trivialReject(RC1, RC2))
167         {
168             //Line coordinates are outside boundary
169             done = true;
170             continue;
171         }
172
173         plotLine = true;    //Clipped Line needs to be highlighted
174
175         if (isInside(RC1))
176         {
177             //If P is inside, then swap P with Q.
178             swapPoints();
179             RC1 = p.encode(windowMin, windowMax);
180             RC2 = q.encode(windowMin, windowMax);
181         }
182
183         if (q.getX() != p.getX())
184         {
185             //Avoid dx = 0 case
186             m = (q.getY() - p.getY()) / (q.getX() - p.getX());
187         }
188
189         if (RC1 & LeftBit)
190         {
191             p.setY(p.getY() + (windowMin.getX() - p.getX()) * m);
192             p.setX(windowMin.getX());
193         }
194
195         else if (RC1 & RightBit)
196         {
197             p.setY(p.getY() + (windowMax.getX() - p.getX()) * m);
198             p.setX(windowMax.getX());
199         }
200

```



```

251     //Render the display using the timer function
252     getParams();
253 }
254
255 void dummyFunction()
256 {
257     //Placeholder function
258 }
259
260 void initializeDisplay()
261 {
262     //Initialize the display parameters
263
264     glClearColor(1, 1, 1, 0);
265     glMatrixMode(GL_PROJECTION);
266     gluOrtho2D(0, WINDOW_WIDTH, 0, WINDOW_HEIGHT);
267     glClear(GL_COLOR_BUFFER_BIT);
268 }
269
270 void getParams()
271 {
272     //To get user parameters for the line clipping process
273
274     Point windowMin = Point(100, 100), windowMax = Point(500, 500);
275     Line lineSegment = Line(300, 200, 400, 500);
276
277     int option = 0;
278
279     drawClippingWindow(windowMin, windowMax);
280     drawLine(lineSegment.getP(), lineSegment.getQ());
281     glFlush();
282
283
284     while (true)
285     {
286         //Await user input
287         cout << "\n\t1. Set Line Coordinates" << endl;
288         cout << "\t2. Set Clipping Window" << endl;
289         cout << "\t0. Exit" << endl;
290         cout << "\tYour Option -> ";
291         cin >> option;
292
293         if (!option)
294         {
295             cout << "\n\t\t-----[COHEN SUTHERLAND LINE CLIPPING ALGORITHM]-----\n";
296             exit(0);
297         }
298
299         if (option == 1)
300         {

```

```

301         float x, y;
302
303         cout << "\n\n\tEnter Point P:" << endl;
304         cout << "\t\tEnter X: ";
305         cin >> x;
306         cout << "\t\tEnter Y: ";
307         cin >> y;
308         lineSegment.setP(x, y);
309
310         cout << "\n\n\tEnter Point Q:" << endl;
311         cout << "\t\tEnter X: ";
312         cin >> x;
313         cout << "\t\tEnter Y: ";
314         cin >> y;
315         lineSegment.setQ(x, y);
316     }
317
318     if (option == 2)
319     {
320         float x, y;
321         cout << "\n\n\tEnter Window Minimums:" << endl;
322         cout << "\t\tEnter X: ";
323         cin >> x;
324         cout << "\t\tEnter Y: ";
325         cin >> y;
326         windowMin.setX(x);
327         windowMin.setY(y);
328
329         cout << "\n\n\tEnter Window Maximums:" << endl;
330         cout << "\t\tEnter X: ";
331         cin >> x;
332         cout << "\t\tEnter Y: ";
333         cin >> y;
334         windowMax.setX(x);
335         windowMax.setY(y);
336
337         glClear(GL_COLOR_BUFFER_BIT);    //Clear the display window
338     }
339
340     drawClippingWindow(windowMin, windowMax);
341     drawLine(lineSegment.getP(), lineSegment.getQ());
342     glFlush();
343
344     bool plotLine = lineSegment.lineClipCohenSutherland(windowMin,
windowMax);
345
346     if (plotLine)
347     {
348         drawLine(lineSegment.getP(), lineSegment.getQ(), true);
349     }
350

```



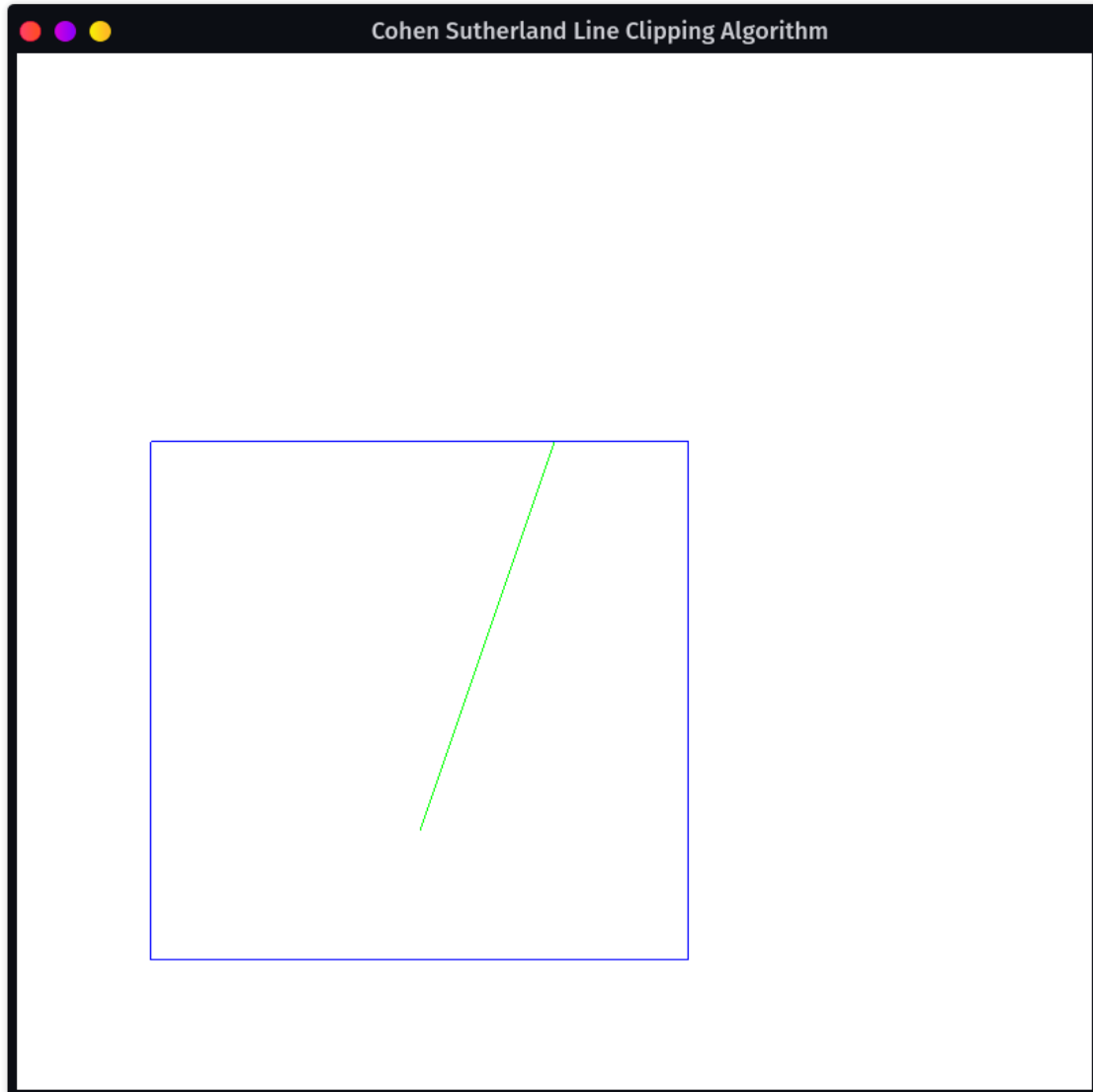
```

351         glFlush();
352     }
353 }
354
355 void drawLine(Point p, Point q, bool clip)
356 {
357     glBegin(GL_LINES);
358
359     if (clip)
360     {
361         glColor3d(1, 0, 0);
362     }
363     else
364     {
365         glColor3d(0, 1, 0);
366     }
367
368     glVertex2f(p.getX(), p.getY());
369     glVertex2f(q.getX(), q.getY());
370
371     glEnd();
372     glFlush();
373 }
374
375 void drawClippingWindow(Point windowMin, Point windowMax)
376 {
377     glBegin(GL_LINE_LOOP);
378     glColor3d(0, 0, 1);
379
380     glVertex2f(windowMin.getX(), windowMin.getY());
381     glVertex2f(windowMax.getX(), windowMin.getY());
382     glVertex2f(windowMax.getX(), windowMax.getY());
383     glVertex2f(windowMin.getX(), windowMax.getY());
384
385     glEnd();
386     glFlush();
387 }

```

Output: Default Line & Clipping Window

Figure 1: Default Line & Clipping Window




Output: Console, Window[(100, 100), (500, 500)]

Figure 2: Output: Console, Window[(100, 100), (500, 500)]

```
.%;888:8898898:
x;XxB889b8:b8%b88:
      .8Xd          8X:.
    .8Xx;          8x:.
      .t8x          .d       x88;
    .@8x8;         .db:     xx@;
      ,tSXX^        .bbbbbbbbbbbbbbB8x@;
    .SXxx          bBBBBBBBBBBBBBBBbSBX8;
      ,88S         pd!
    8X88/         q
    GBB.
    x88      d88@8@x@x@x88x@8x@x@x@8@x.
    dxXd     db8b8b8b8b8b8b8b8b8b8b8b8b8x.
    dx8@     .@@;.
    dx88     .t@x.
    d:SS8ba89aa67a853Sxxad.
    .d88999889889899dd.
```

OS:	Garuda Linux
Host:	81FV Lenovo Legion Y530-15ICH
Kernel:	5.13.13-zen1-1-zen
Uptime:	31 mins
Packages:	1415 (pacman)
Shell:	bash
DE:	Plasma 5.22.4
WM:	KWin
Terminal:	konsole
CPU:	Intel i7-8750H (12) @ 4.1GHz
GPU:	Intel CoffeeLake-H GT2 [UHD Graphics 630]
GPU:	NVIDIA GeForce GTX 1050 Mobile
Memory:	2.7GiB / 7.64GiB



```
vishakan@Legion in repo: GraphicsLab/Ex07 - Cohen Sutherland Line Clipping on ✎ main [x?]
λ ./CohenSutherland
```

-----[COHEN SUTHERLAND LINE CLIPPING ALGORITHM]-----

1. Set Line Coordinates
2. Set Clipping Window
0. Exit

Your Option → _

Output: Window[(100, 100), (600, 600)]

Figure 3: Output: Window[(100, 100), (600, 600)].



Output: Console, Window[(100, 100), (600, 600)]

Figure 4: Output: Console, Window[(100, 100), (600, 600)].



```
Ex07 - Cohen Sutherland Line Clipping : CohenSutherland - Konsole

d:5548ba89aa67a853Sxxad.
.d9889998898898899dd.

vishakan@Legion in repo: GraphicsLab/Ex07 - Cohen Sutherland Line Clipping on P main [x?]
A ./CohenSutherland

-----[COHEN SUTHERLAND LINE CLIPPING ALGORITHM]-----

1. Set Line Coordinates
2. Set Clipping Window
0. Exit
Your Option → 2

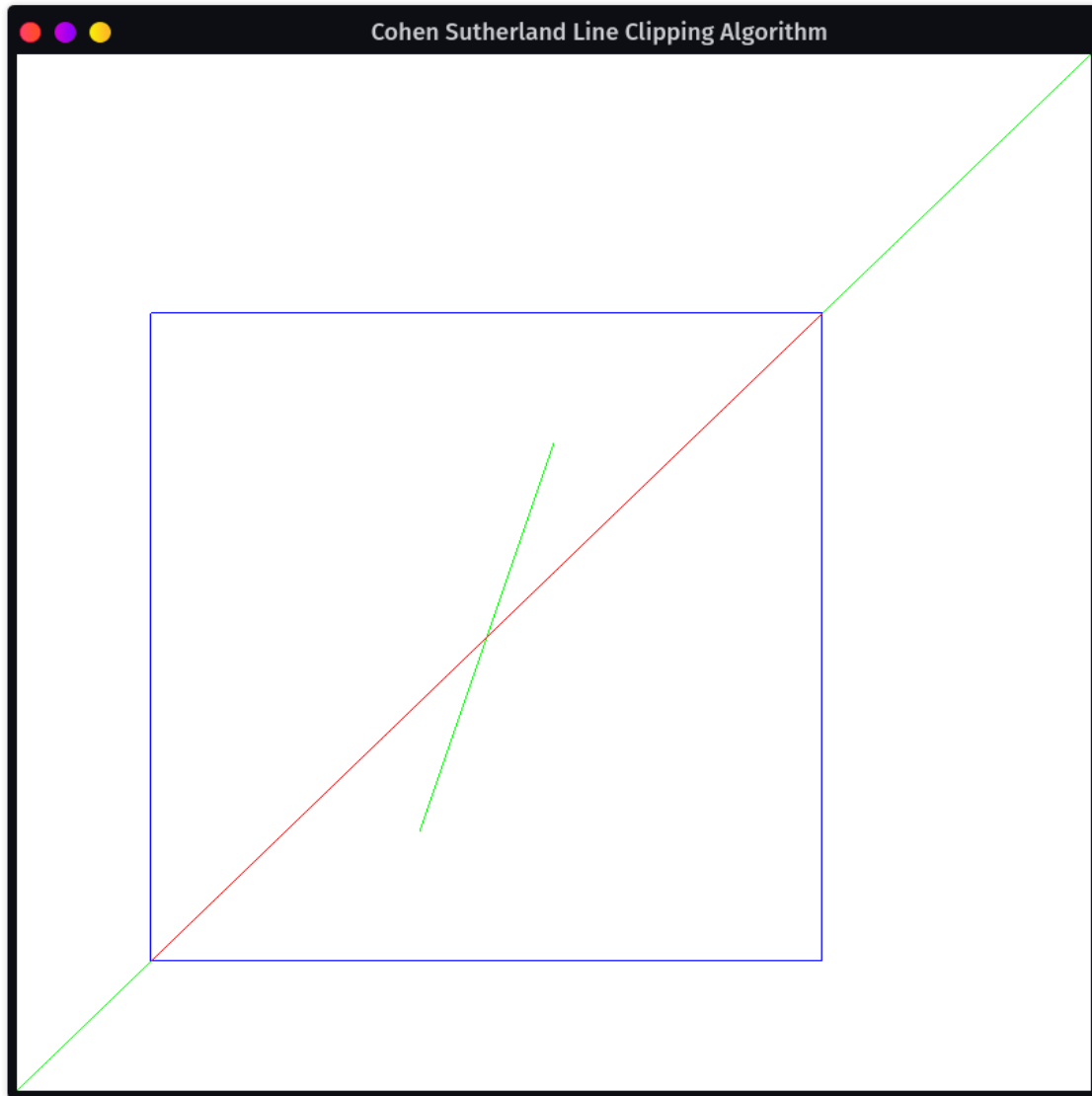
Enter Window Minimums:
  Enter X: 100
  Enter Y: 100

Enter Window Maximums:
  Enter X: 600
  Enter Y: 600

1. Set Line Coordinates
2. Set Clipping Window
0. Exit
Your Option →
```

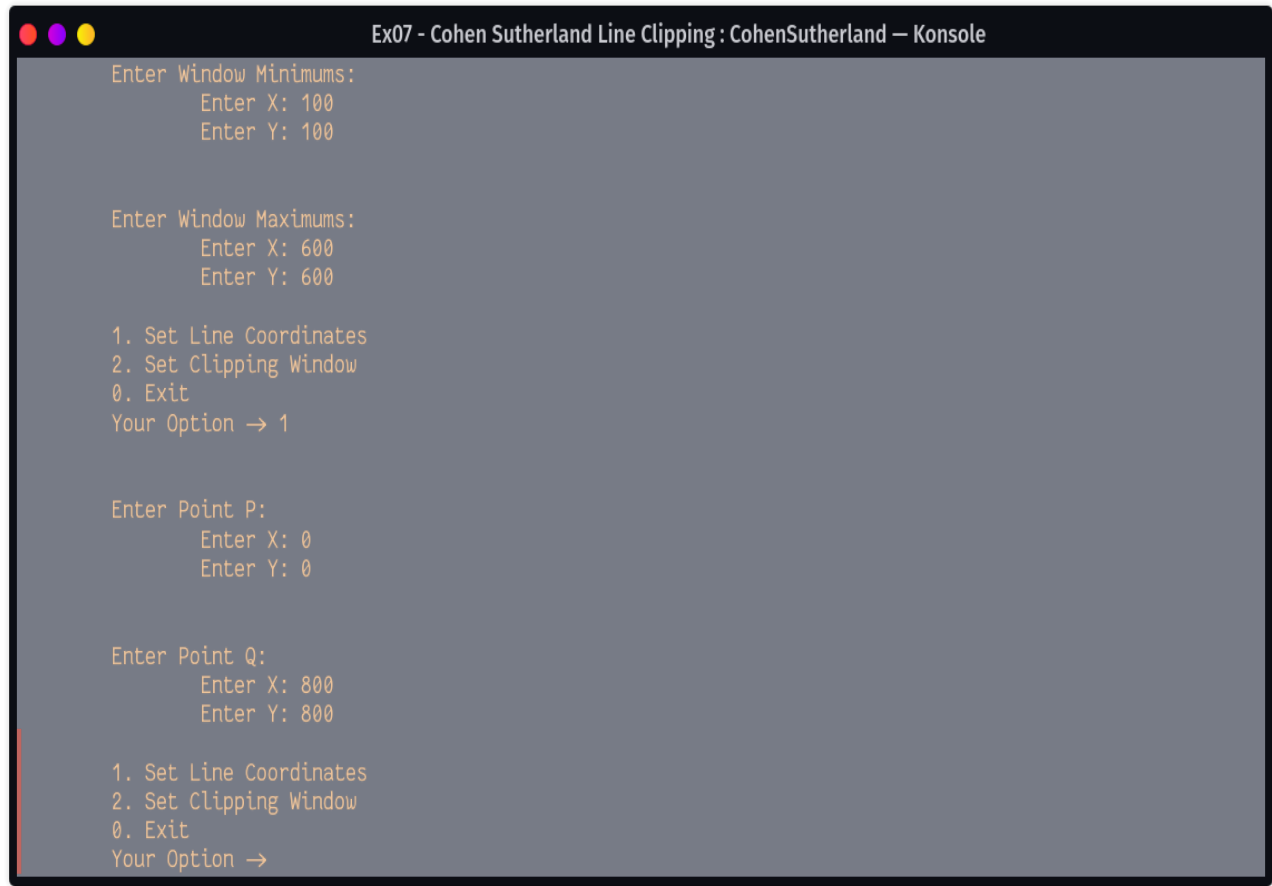
Output: Line[(0, 0), (800, 800)] & Clipping

Figure 5: Output: Line[(0, 0), (800, 800)] & Clipping.



Output: Console, Line[(0, 0), (800, 800)] & Clipping

Figure 6: Output: Console, Line[(0, 0), (800, 800)] & Clipping.



```
Ex07 - Cohen Sutherland Line Clipping : CohenSutherland — Konsole

Enter Window Minimums:
    Enter X: 100
    Enter Y: 100

Enter Window Maximums:
    Enter X: 600
    Enter Y: 600

1. Set Line Coordinates
2. Set Clipping Window
0. Exit
Your Option → 1

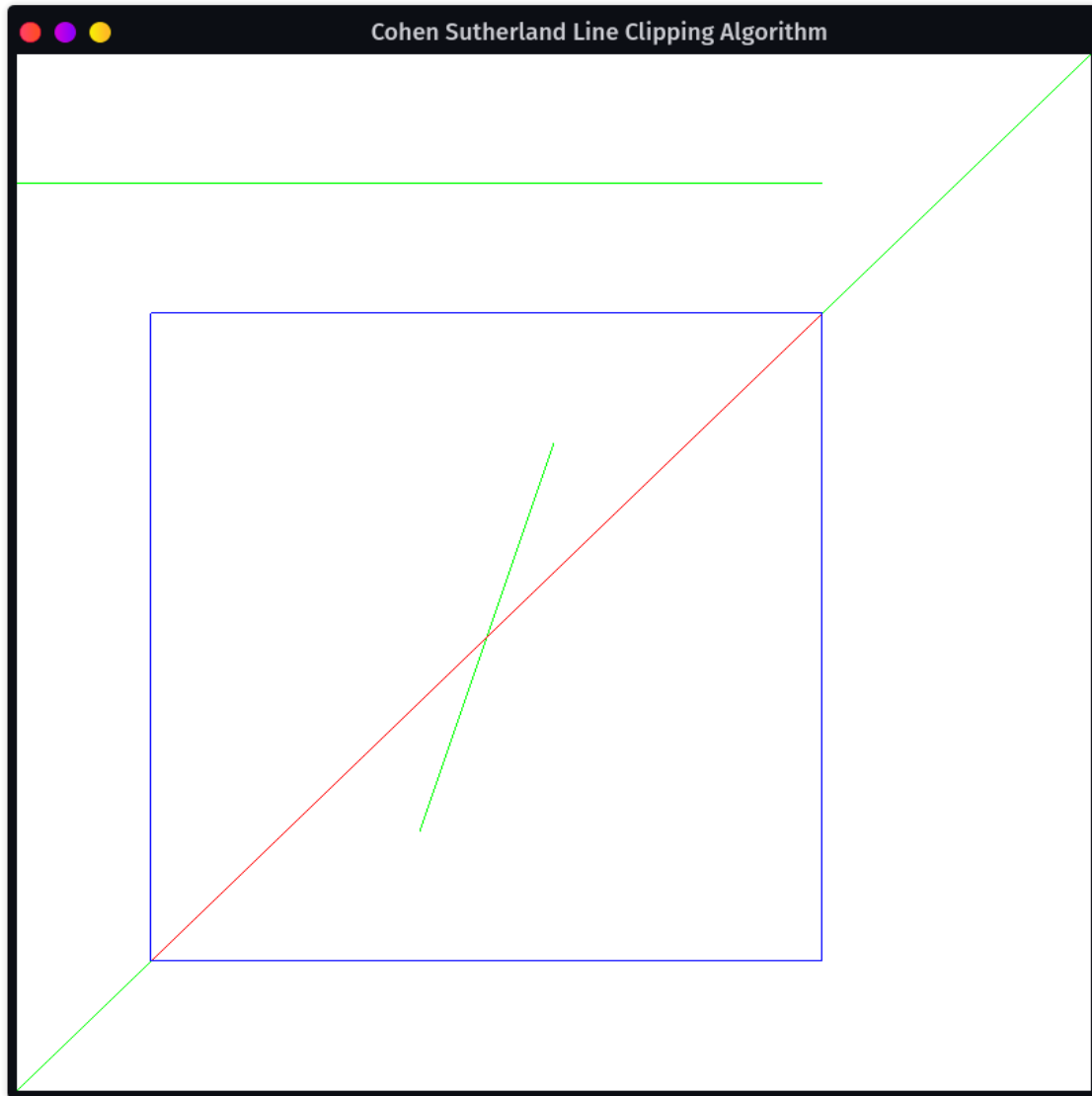
Enter Point P:
    Enter X: 0
    Enter Y: 0

Enter Point Q:
    Enter X: 800
    Enter Y: 800

1. Set Line Coordinates
2. Set Clipping Window
0. Exit
Your Option →
```

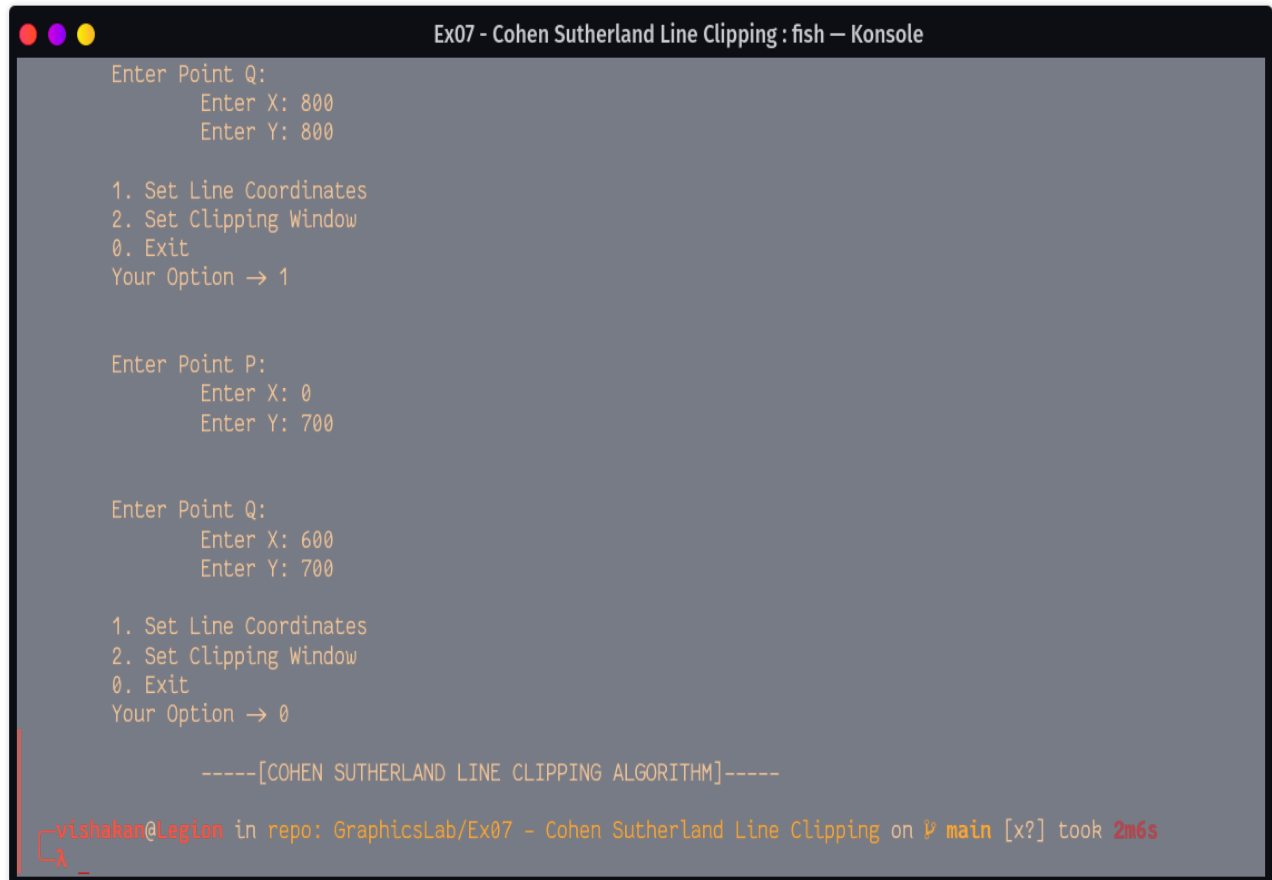
Output: Line[(0, 700), (600, 700)] & Clipping

Figure 7: Output: Line[(0, 700), (600, 700)] & Clipping.



Output: Console, Line[(0, 700), (600, 700)] & Clipping

Figure 8: Output: Console, Line[(0, 700), (600, 700)] & Clipping.



```
Ex07 - Cohen Sutherland Line Clipping : fish - Konsole

Enter Point Q:
  Enter X: 800
  Enter Y: 800

1. Set Line Coordinates
2. Set Clipping Window
0. Exit
Your Option → 1

Enter Point P:
  Enter X: 0
  Enter Y: 700

Enter Point Q:
  Enter X: 600
  Enter Y: 700

1. Set Line Coordinates
2. Set Clipping Window
0. Exit
Your Option → 0

-----[COHEN SUTHERLAND LINE CLIPPING ALGORITHM]-----

vishakan@Legion in repo: GraphicsLab/Ex07 - Cohen Sutherland Line Clipping on P main [x?] took 2m6s
└─
```

Learning Outcome:

- I learnt about **Cohen-Sutherland Line Clipping Algorithm**'s theoretical basis.
- I learnt how to compute and program **Region Codes** for a given point and clipping window.
- I understood about the advantages and pitfalls of the Cohen-Sutherland Algorithm.
- I understood how to implement **trivial accept & reject** conditions in the algorithm.
- I was able to implement the algorithm for any given line and a rectangular clipping window region.
- I demonstrated the clipped line, the original line and the clipping window using different colors.
- I handled corner cases where $dx = 0$ and thus avoiding divide by zero errors in slope calculation.
- I understood how to swap the points and continue clipping till region codes of both points become 0 for trivial acceptance.