EX-07 - Cohen Sutherland Line clipping in C++ using OpenGL

28/09/2021

Venkataraman Nagarajan, CSE - C
18500192

AIM

To implement Cohen Sutherland line clipping algorithm in C++.

SPECIFICATION

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

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.

PROGRAM - 01

Cohen Sutherland - "init.h header"

```
#include<bits/stdc++.h>
#include<GL/glut.h>

using namespace std;
using ld = long double;
using ll = long long;

#define X       first
#define Y       second

typedef pair<ld,ld> pld;

const int WINDOW_WIDTH = 900;
const int WINDOW_HEIGHT = 900;

const int X_MIN = -300;
const int X_MAX = 300;
const int Y_MIN = -300;
const int Y_MAX = 300;

enum pos {LEFT, RIGHT, BOTTOM, TOP};

struct Window {
    ld X_MIN, X_MAX, Y_MIN, Y_MAX;

    Window(): X_MAX(::X_MAX), X_MIN(::X_MIN), Y_MAX(::Y_MAX), Y_MIN(::Y_MIN) {}
    Window(ld X_MIN, ld X_MAX, ld Y_MIN, ld Y_MAX): X_MAX(X_MAX), X_MIN(X_MIN), Y_MAX(Y_MAX), Y_MIN(Y_MIN) {}

    void displayFull(ld r = 1.0, ld g = 0.0, ld b = 0.0) {
        glColor3f(r,g,b);
        glBegin(GL_LINES);
            glVertex2d(::X_MIN, Y_MIN);
            glVertex2d(::X_MAX, Y_MIN);

            glVertex2d(::X_MIN, Y_MAX);
            glVertex2d(::X_MAX, Y_MAX);

            glVertex2d(X_MIN, ::Y_MIN);
            glVertex2d(X_MIN, ::Y_MAX);

            glVertex2d(X_MAX, ::Y_MIN);
```

```
42              glVertex2d(X_MAX, ::Y_MAX);
43          glEnd();
44      }
45
46      void displayWindow(ld r = 1.0, ld g = 0.0, ld b = 0.0) {
47          glColor3f(r,g,b);
48          glBegin(GL_LINE_LOOP);
49              glVertex2d(X_MIN, Y_MIN);
50              glVertex2d(X_MIN, Y_MAX);
51              glVertex2d(X_MAX, Y_MAX);
52              glVertex2d(X_MAX, Y_MIN);
53          glEnd();
54      }
55
56  };
57
58  struct CohenVector {
59      ll vector;
60      ll TOP, BOTTOM, RIGHT, LEFT;
61
62      CohenVector(): TOP(8), BOTTOM(4), RIGHT(2), LEFT(1), vector(0) {};
63
64      ll calcCohenValue(pld point, Window window) {
65          vector = 0;
66
67          vector += (point.Y > window.Y_MAX)*TOP;
68          vector += (point.Y < window.Y_MIN)*BOTTOM;
69          vector += (point.X > window.X_MAX)*RIGHT;
70          vector += (point.X < window.X_MIN)*LEFT;
71
72          return vector;
73      }
74
75      bool trivialAccept(CohenVector next) {
76          return (vector | next.vector) == 0;
77      }
78
79      bool trivialReject(CohenVector next) {
80          return (vector & next.vector);
81      }
82  };
83
84  struct Line {
85      pld A,B;
86      ld slope;
87      Line(pld A, pld B): A(A), B(B), slope((B.Y - A.Y)/(B.X - A.X)) {}
88
```

```
89      void display(ld r = 1.0, ld g = 0.0, ld b = 0.0) {
90          glColor3f(r,g,b);
91          glBegin(GL_LINES);
92              glVertex2d(A.X, A.Y);
93              glVertex2d(B.X, B.Y);
94          glEnd();
95      }
96  };
97
98  pld findNewPosition(pld A, Line l, Window window, pos i) {
99      pld c;
100     if(i == TOP) {
101         c.Y = window.Y_MAX;
102         c.X = A.X + 1/l.slope * (c.Y - A.Y);
103     } else if(i == BOTTOM) {
104         c.Y = window.Y_MIN;
105         c.X = A.X + 1/l.slope * (c.Y - A.Y);
106     } else if(i == LEFT) {
107         c.X = window.X_MIN;
108         c.Y = A.Y + l.slope * (c.X - A.X);
109     } else {
110         c.X = window.X_MAX;
111         c.Y = A.Y + l.slope * (c.X - A.X);
112     }
113
114     return c;
115 }
116
117 Line findIntersection(Line line, Window window) {
118     CohenVector c = CohenVector();
119     c.calcCohenValue(line.A, window);
120
121     for(ll i=0;i<4;i++) {
122         if((c.vector & (1<<i))) {
123             line.A = findNewPosition(line.A, line, window, (pos)i);
124             return line;
125         }
126     }
127
128     c.calcCohenValue(line.B, window);
129
130     for(ll i=0;i<4;i++) {
131         if((c.vector & (1<<i))) {
132             line.B = findNewPosition(line.B, line, window, (pos)i);
133             return line;
134         }
135     }
```

```
136
137     return line;
138 }
```

Cohen Sutherland- implementation

```
1  // Apply Cohen Sutherland line clipping on a line (x1,y1) (x2,y2) with ↩
       respect to a clipping window
2  // (XWmin,YWmin) (XWmax,YWmax).
3
4  // After clipping with respect to an edge, display the line segment with ↩
       the calculated intermediate
5  // intersection points and the vertex list.
6
7  // Input: The clipping window co-ordinates and the line endpoints
8
9  // Note: The output should show the clipping window and the line to be ↩
       clipped in different colors.
10
11 // You can show the intermediate steps using time delay.
12
13 #include "__init__.h"
14 #include <unistd.h>
15
16 const ld PADDING = 0;
17 const ld STEP = 10;
18 const ld SCALE = 1;
19 const ld PI = 3.14159265358979323846264338327950288419716939937510582;
20 const ll SCREEN_FPS = 1;
21
22 void myInit();
23 void myDisplay();
24
25 void LineCuttingAlgorithm();
26 void LinePrinting(ll val);
27 Line getLineInput();
28 Window getWindowInput();
29
30 vector<Line> lines;
31 Window window;
32 ll cou = 0;
33 bool isLineAccepted;
34
35 void runMainLoop(int val);
36
37 int main(int argc,char* argv[]) {
```

```
38      LineCuttingAlgorithm();
39      glutInit(&argc,argv);
40      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
41      glutInitWindowSize(WINDOW_WIDTH,WINDOW_HEIGHT);
42      glutCreateWindow("Cohen Sutherland");
43      glutDisplayFunc(myDisplay);
44      glutTimerFunc(1000/ SCREEN_FPS, runMainLoop, 0);
45      myInit();
46      glutMainLoop();
47      return 1;
48  }
49
50  void myInit() {
51      glClearColor(1.0,1.0,1.0,0.0);
52      glColor3f(0.0f,0.0f,0.0f);
53      glPointSize(5.0);
54      glMatrixMode(GL_PROJECTION);
55      glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
56      glEnable( GL_BLEND );
57      glLoadIdentity();
58      gluOrtho2D(X_MIN,X_MAX,Y_MIN,Y_MAX);
59  }
60
61  void runMainLoop(int val) {
62      myDisplay();
63
64      glutTimerFunc(1000/ SCREEN_FPS, runMainLoop, 0);
65  }
66
67  void myDisplay() {
68      glClear(GL_COLOR_BUFFER_BIT);
69
70      LinePrinting(cou);
71      cou = (cou + 1)%(lines.size()+1);
72
73      glFlush();
74  }
75
76  Window getWindowInput() {
77      ld X_MAX, X_MIN, Y_MAX, Y_MIN;
78
79      cout << "Enter Window Limits: \n";
80      cout << "\t X_MIN : "; cin >> X_MIN;
81      cout << "\t X_MAX : "; cin >> X_MAX;
82      cout << "\t Y_MIN : "; cin >> Y_MIN;
83      cout << "\t Y_MAX : "; cin >> Y_MAX;
84
```

```
85      return Window(X_MIN, X_MAX, Y_MIN, Y_MAX);
86  }
87
88  Line getLineInput() {
89      pld a,b;
90
91      cout << "\nEnter End-points of the line: \n";
92      cout << "\t A(x,y) : "; cin >> a.X >> a.Y;
93      cout << "\t B(x,y) : "; cin >> b.X >> b.Y;
94
95      return Line(a, b);
96  }
97
98  void LineCuttingAlgorithm() {
99      cout << "\t\t Cohen Sutherland Line Cutting \n\n";
100
101     window = getWindowInput();
102     Line line = getLineInput();
103
104     // Handle corner cases of infinite looping at trivial reject
105     ll local_counter = 0;
106
107     do {
108         lines.push_back(line);
109
110         CohenVector c = CohenVector();
111         CohenVector d = CohenVector();
112
113         ll vec1 = c.calcCohenValue(line.A, window);
114         ll vec2 = d.calcCohenValue(line.B, window);
115
116         if(c.trivialAccept(d)) {
117             isLineAccepted = true;
118             break;
119         }
120
121
122         if(c.trivialReject(d)) {
123             isLineAccepted = false;
124             break;
125         }
126
127         line = findIntersection(line, window);
128
129
130         local_counter ++;
131     } while(local_counter <= 5);
```

```
132
133 }
134
135 void LinePrinting(ll i) {
136     if(i != lines.size()) {
137         window.displayFull(0,1,0);
138     } else {
139         window.displayWindow(0,0,0);
140     }
141
142     if(i < lines.size() -1 || isLineAccepted)
143         lines[min((ll)lines.size()-1,i)].display(1,0,0);
144 }
```

SAMPLE I/0

```
LAB/Multimedia and graphics Lab(main*) » g++ "/media/venky/New Volume/SSN/SEMESTER 7/LAB/Multimedia and graphics Lab/EX07 - Cohen Sutherland/02-CohenSutherland-Dynamic.cpp" -lGL -lGLU -lglut && ./a.out
                Cohen Sutherland Line Cutting

Enter Window Limits:
        X_MIN : -100
        X_MAX : 100
        Y_MIN : -100
        Y_MAX : 100

Enter End-points of the line:
        A(x,y) : -115 -225
        B(x,y) : 115 225
```

Figure 1: Input for Line clipping

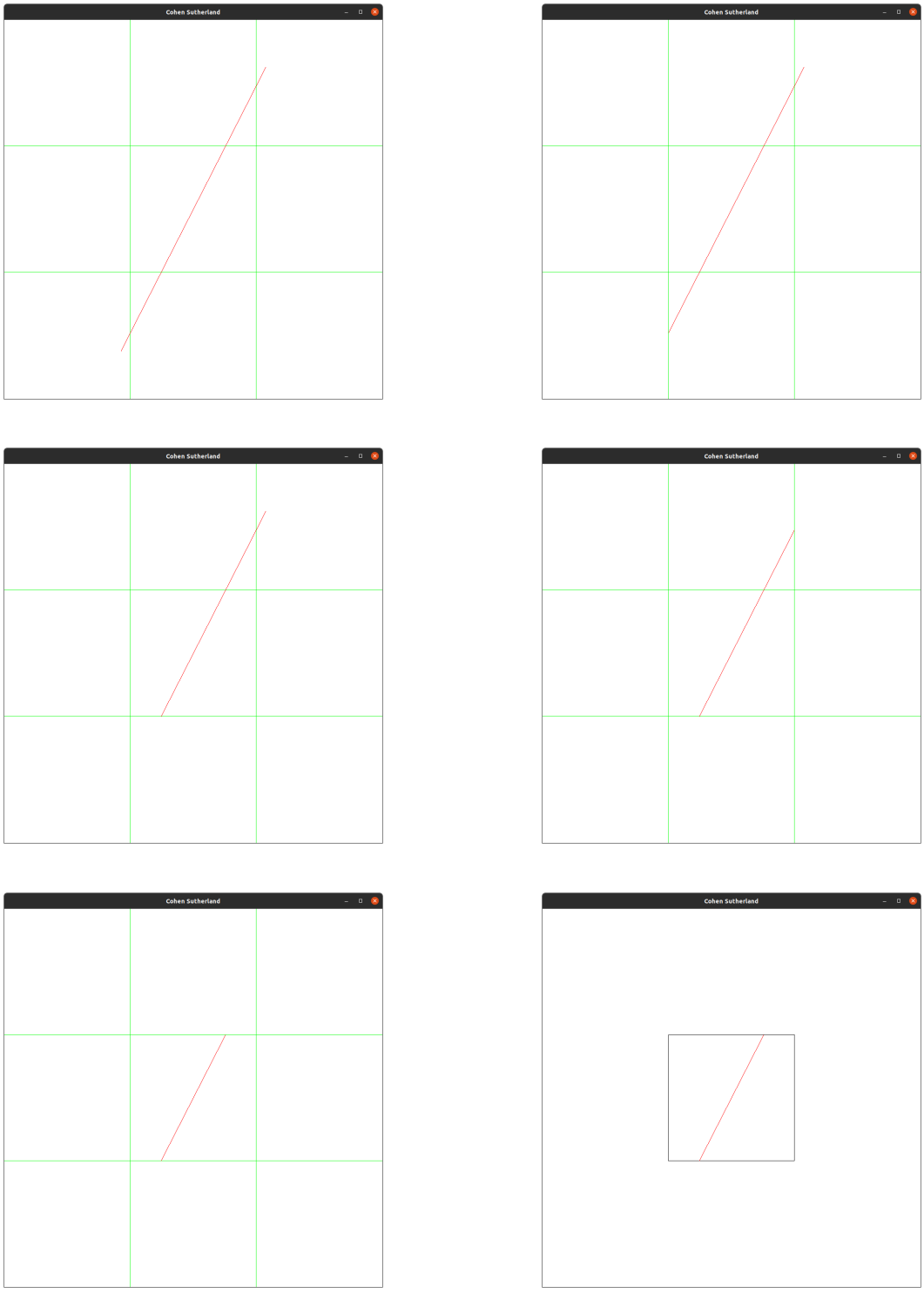

Figure 2: Moving images of Cohen Sutherland Line clipping algorithm

RESULT

The code for Cohen Sutherland Line clipping algorithm is written and output is verified.
