Lab Report

Week 6

Jitendra Kumar , 1401CS19

27/02/2017

Title

- Implement the Cohen-Sutherland algorithm for line clipping.
  1). OpenGL
  2). MatLab

Procedure

OpenGL

1). Choose end points of a line and define a clipping window and then apply the algorithm described below :

- Create a C file and name it as *cohenSutherland.c*.
- Following is the final code for cohenSutherland algorithm for line clipping :

```
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>
#include <ctype.h>
#include <GL/glut.h>

double x_min=10 , x_max=50, y_min=20, y_max=50;
double xf1=0, yf1=0, xf2=60, yf2=50;

int get_code_for_this_point(float x,float y)
{
        int c=0;
        if(y>y_max) c=8;
        if(y<y_min) c=4;
        if(x>x_max) c=c|2;
        if(x<x_min) c=c|1;
        return c;
}

void clipping_Line(float x1,float y1,float x2,float y2)
{
        int c1=get_code_for_this_point(x1,y1);
        int c2=get_code_for_this_point(x2,y2);

        while((c1|c2) != 0)
        {
                if((c1&c2) > 0)
                {
                    printf("\nLine outside the window\n");
                    break;
                }

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

                float xi = x1 ;
                float yi = y1 ;
                int code = c1 ;
```

```
                if(code==0)
                {
                    code = c2 ;
                    xi = x2 ;
                    yi = y2 ;
                }

                float x = 0 ; float y = 0;

                if((code & 8)>0)
                    {
                        y = y_max;
                        x = xi+ 1.0/slope*(y_max-yi);
                }
                else if((code & 4)>0)
                {
                    y = y_min;
                    x = xi+1.0/slope*(y_min-yi);
                }
                else if((code & 2)>0)
                {
                    x = x_max;
                    y = yi+slope*(x_max-xi);
                }
                else if((code & 1)>0)
                {
                    x = x_min;
                    y = yi+slope*(x_min-xi);
                }

                if(code == c1)
                {
                    xf1 = x ;
                    yf1 = y ;
                    c1 = get_code_for_this_point(xf1,yf1);
                }
                if(code == c2)
                {
                    xf2 = x ;
                    yf2 = y ;
                    c2 = get_code_for_this_point(xf2,yf2);
                }
        }
}


void display_function_after()
{
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glLineWidth(3);
        glBegin(GL_LINES);
                glColor3f(0.0f, 0.0f, 0.0f);
                glVertex2f(0.0f,400.0f);
                glVertex2f(0.0f,-400.0f);
                glVertex2f(400.0f,0.0f);
                glVertex2f(-400.0f,0.0f);
        glEnd();

        glLineWidth(3);
        glBegin(GL_LINE_LOOP);
                glColor3f(1.0f, 0.0f, 0.0f);
                glVertex2f(x_min/100.0f,y_min/100.0f);
                glVertex2f(x_max/100.0f,y_min/100.0f);
                glVertex2f(x_max/100.0f,y_max/100.0f);
                glVertex2f(x_min/100.0f,y_max/100.0f);
        glEnd();


        clipping_Line(xf1,yf1,xf2,yf2);

        glLineWidth(3);
        glBegin(GL_LINES);
                glColor3f(0.0f, 0.0f, 1.0f);
                glVertex2f(xf1/100.0f,yf1/100.0f);
                glVertex2f(xf2/100.0f,yf2/100.0f);
        glEnd();
        glFlush();
        glutSwapBuffers();
```

```
}
void display_function_before()
{
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glLineWidth(3);
        glBegin(GL_LINES);
                glColor3f(0.0f, 0.0f, 0.0f);
                glVertex2f(0.0f,400.0f);
                glVertex2f(0.0f,-400.0f);
                glVertex2f(400.0f,0.0f);
                glVertex2f(-400.0f,0.0f);
        glEnd();

        glLineWidth(3);
        glBegin(GL_LINE_LOOP);
                glColor3f(1.0f, 0.0f, 0.0f);
                glVertex2f(x_min/100.0f,y_min/100.0f);
                glVertex2f(x_max/100.0f,y_min/100.0f);
                glVertex2f(x_max/100.0f,y_max/100.0f);
                glVertex2f(x_min/100.0f,y_max/100.0f);
        glEnd();


        glLineWidth(3);
        glBegin(GL_LINES);
                glColor3f(0.0f, 1.0f, 0.0f);
                glVertex2f(xf1/100.0f,yf1/100.0f);
                glVertex2f(xf2/100.0f,yf2/100.0f);
        glEnd();

        glFlush();
    glutSwapBuffers();
}

int main(int argc, char const *argv[])
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGB);
        glutInitWindowSize(500, 500);
        gluOrtho2D(-400, 400, -400, 400);

        glutCreateWindow("Cohen-sutherland Line Clipping: Before");
        glutDisplayFunc(display_function_before);

        glutCreateWindow("Cohen-sutherland Line Clipping: After");
        glutDisplayFunc(display_function_after);

        glutMainLoop();
        return 0;
}
```

- Compile and run the executable file in terminal by typing in the following commands :
  (a) gcc cohenSutherland.c -lGL -lGLU -lglut -ll
  (b) ./a.out

MatLab

1). Choose end points of a line and define a clipping window and then apply the algorithm described below :

- Open a new matlab script and define a function cohenSutherland().
- Following is the final code for cohenSutherland algorithm for line clipping.

```
function z = cohenSutherland()

        xmin=20; xmax=40; ymin=20 ; ymax=40 ;
        xf1=10 ; yf1=10 ; xf2=41  ; yf2=47 ;
        plot([xf1 xf2], [yf1 yf2], 'r--','LineWidth', 3);
        hold on;
        code1 = code(xf1, yf1, xmin, xmax, ymin, ymax);
        code2 = code(xf2, yf2, xmin, xmax, ymin, ymax);

                if bitand(code1,code2) > 0
                        disp("Line  outside the clipping window");
                        exit(1);
```

```
                end

        while bitor(code1,code2) > 0

                slope  = (yf2-yf1)/(xf2-xf1);
                codex = code1 ;
                xi = xf1 ;
                yi = yf1 ;

                if(codex==0)
                        codex = code2;
                        xi = xf2 ;
                        yi = yf2 ;
                end

                if bitand(codex,8)>0
                        y = ymax;
                        x = xi + 1/slope*(ymax-yi);
                elseif bitand(codex,4)>0
                        y = ymin ;
                        x = xi + 1/slope*(ymin-yi);
                elseif bitand(codex,2)>0
                        x = xmax;
                        y = yi + slope*(xmax-xi);
                elseif bitand(codex,1)>0
                        x = xmin;
                        y = yi + slope*(xmin-xi);
                end

                if codex == code1
                        xf1 = x;
                        yf1 = y;
                        code1 = code(xf1, yf1, xmin, xmax, ymin, ymax);
                elseif codex == code2
                        xf2 = x ;
                        yf2 = y ;
                        code2 = code(xf2, yf2, xmin, xmax, ymin, ymax);
                end
        end

        boundries_x = [xmin,xmax,xmax,xmin,xmin];
        boundries_y = [ymin,ymin,ymax,ymax,ymin];
        plot(boundries_x, boundries_y, 'b-', 'LineWidth', 3);
        hold on;
        line_x = [xf1,xf2];
        line_y = [yf1,yf2];
        plot(line_x, line_y, 'g-', 'LineWidth', 5);

end


function c = code(xf2, yf2, xmin, xmax, ymin, ymax)
        c = 0;
        if (yf2>ymax)
                c = 8;
        end
        if (yf2<ymin)
                c = 4;
        end
        if (xf2>xmax)
                c = bitor(c,2);
        end
        if (xf2<xmin)
                c = bitor(c,1);
        end
end
```

Output

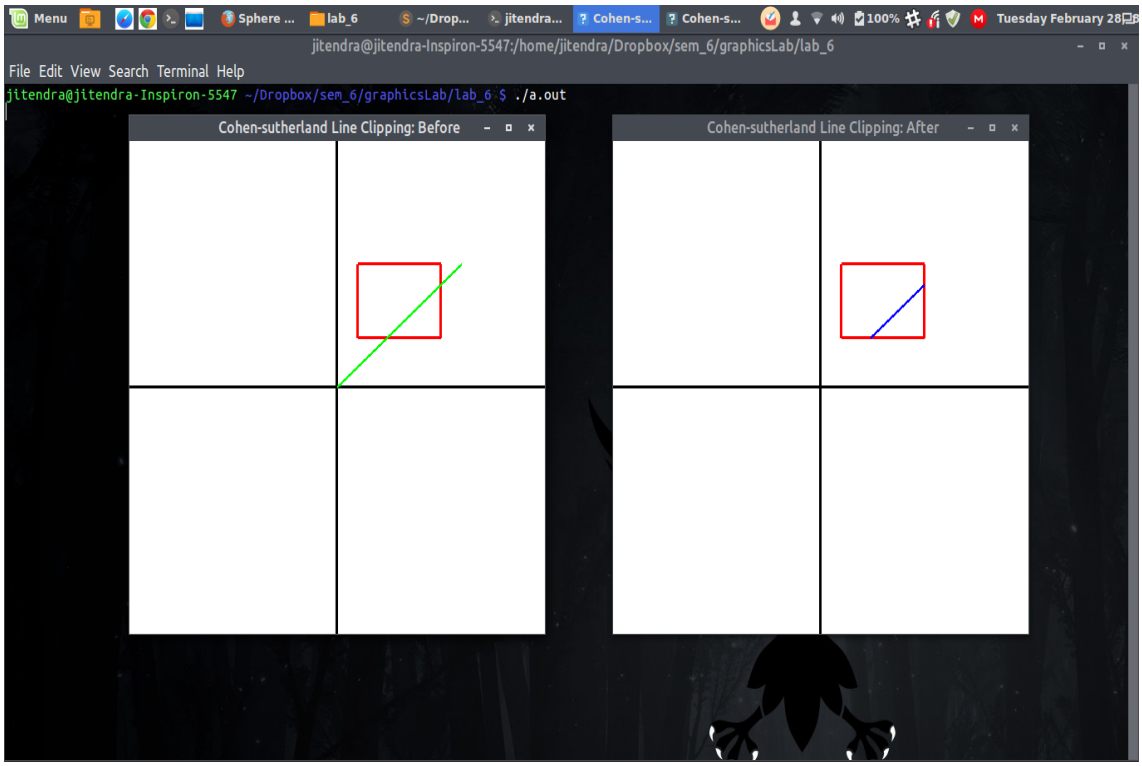

FIGURE 1 – cohenSutherland clipping in OpenGL

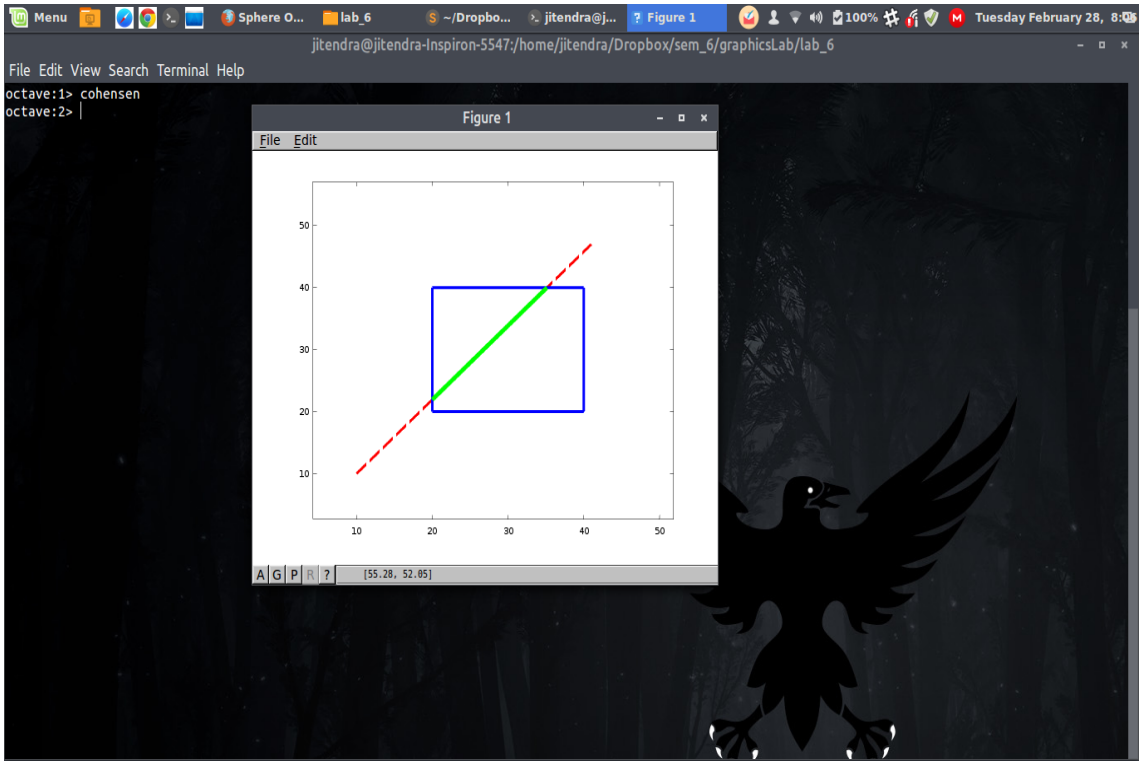

FIGURE 2 – cohenSutherland clipping in Matlab