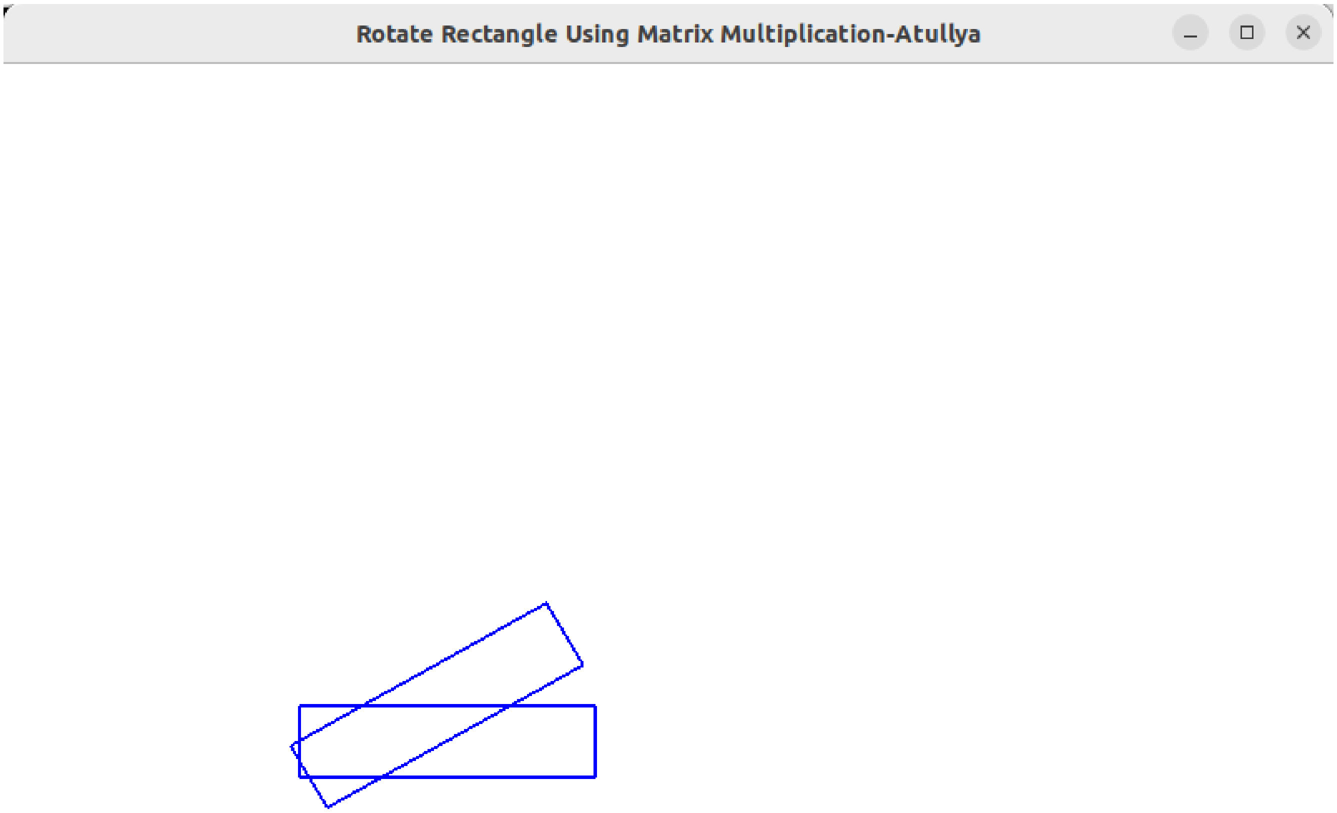
18. Do the same thing 17 without using opengl transformation methods and see if the results are same.

Source Code

```
#include <GL/glut.h>
#include <cmath>

float x, y;
float width, height;
int win_width = 800;
int win_height = 600;

void drawRectangle(float x, float y, float width, float height) {
   glColor3f(0.0f, 0.0f, 1.0f);
   glLineWidth(2.0);
   glBegin(GL_LINE_LOOP);
   glVertex2f(x, y);
   glVertex2f(x + width, y);
   glVertex2f(x + width, y + height);
   glVertex2f(x, y + height);
   glEnd();
}

void matrixMultiply(float matrix[3][3], float& x, float& y) {
   float tempX = matrix[0][0] * x + matrix[0][1] * y + matrix[0][2];
   float tempY = matrix[1][0] * x + matrix[1][1] * y + matrix[1][2];
   x = tempX;
   y = tempY;
}

void display() {
   glClear(GL_COLOR_BUFFER_BIT);

   // Draw the original rectangle
   drawRectangle(x, y, width, height);
```

```
// Calculate the center of rotation
float centerX = 150.0f;
float centerY = 125.0f;

// Create rotation matrix
float angle = 30.0 * M_PI / 180.0;
float cosA = cos(angle);
float sinA = sin(angle);

float rotationMatrix[3][3] = {
    {cosA, -sinA, centerX * (1 - cosA) + centerY * sinA},
    {sinA, cosA, centerY * (1 - cosA) - centerX * sinA},
    {0, 0, 1}
};

// Rotate the vertices of the rectangle using matrix multiplication
float x1 = x;
float y1 = y;
float x2 = x + width;
float y2 = y;
float x3 = x + width;
float y3 = y + height;
float x4 = x;
float y4 = y + height;

matrixMultiply(rotationMatrix, x1, y1);
matrixMultiply(rotationMatrix, x2, y2);
matrixMultiply(rotationMatrix, x3, y3);
matrixMultiply(rotationMatrix, x4, y4);

// Draw the rotated rectangle
glBegin(GL_LINE_LOOP);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
```

```
    glVertex2f(x4, y4);
    glEnd();

    glFlush();
}

void reshape(int w, int h) {
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-100, win_width, -100, win_height);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

int main(int argc, char *argv[]) {
    x = 100;
    y = 100;
    width = 200;
    height = 50;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(win_width, win_height);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Rotate Rectangle Using Matrix Multiplication-
Atullya");

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);

    glutMainLoop();

    return 0;
}
```



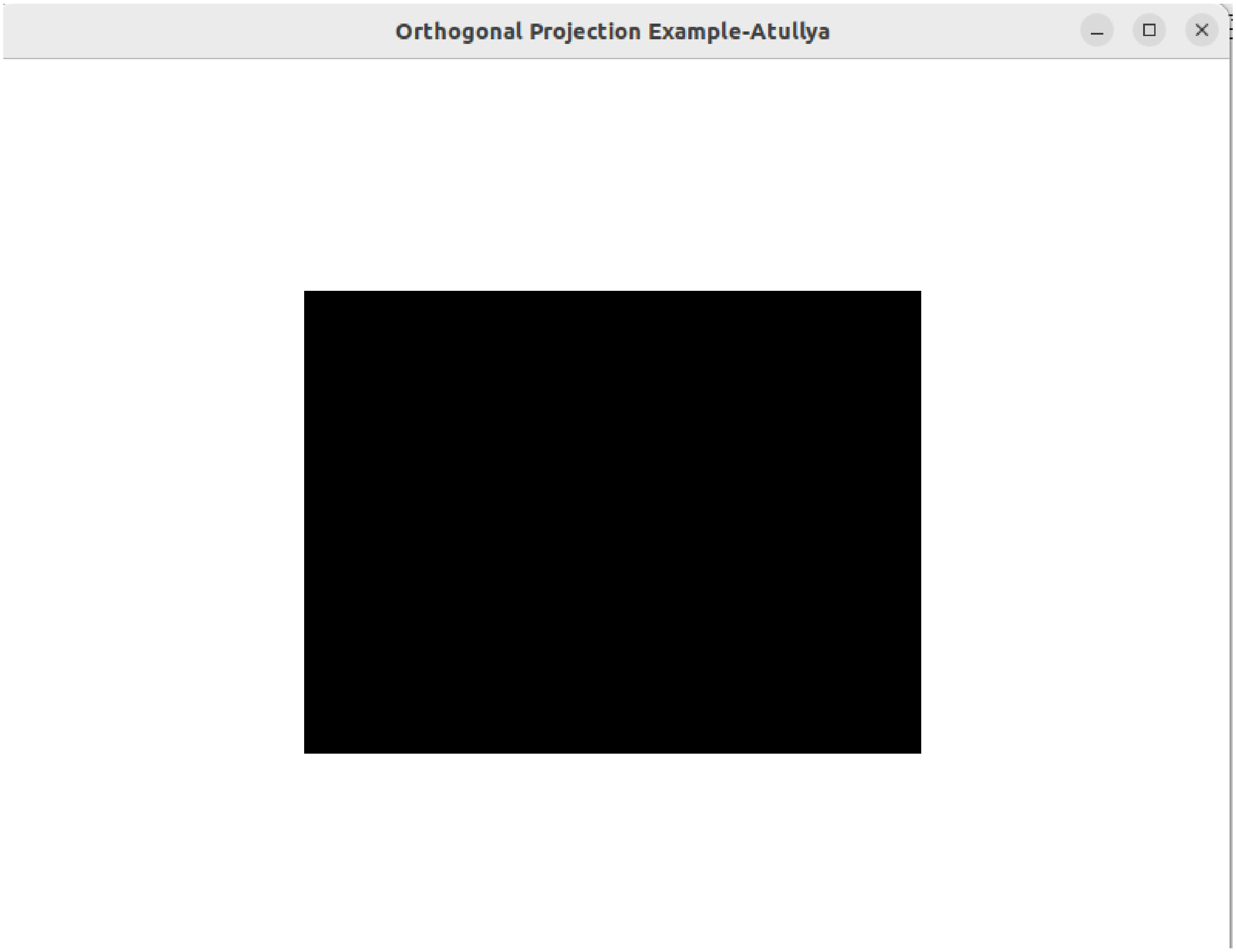
Rotate Rectangle Using Matrix Multiplication-Atullya

19. Write an OPENGL app to illustrate orthogonal projection

Source Code

```
#include <GL/glut.h>
#include <iostream>


int windowWidth = 800;
int windowHeight = 600;

void drawScene() {
   glClear(GL_COLOR_BUFFER_BIT);

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();


   glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();

   // Set the background color to white
   glClearColor(1.0, 1.0, 1.0, 1.0);
   // Clear the color buffer
   glClear(GL_COLOR_BUFFER_BIT);

   // Set the square color to black
   glColor3f(0.0, 0.0, 0.0);

   // Draw a square
   glBegin(GL_QUADS);
   glVertex2f(-0.5, -0.5);
   glVertex2f(0.5, -0.5);
   glVertex2f(0.5, 0.5);
```

```
    glVertex2f(-0.5, 0.5);
    glEnd();

    glutSwapBuffers();
}

void reshape(int width, int height) {
    glViewport(0, 0, width, height);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

    glutInitWindowSize(windowWidth, windowHeight);
    glutCreateWindow("Orthogonal Projection Example-Atullya");

    glutDisplayFunc(drawScene);
    glutReshapeFunc(reshape);
    glClearColor(1.0, 1.0, 1.0, 1.0);

    glutMainLoop();
    return 0;
}
```



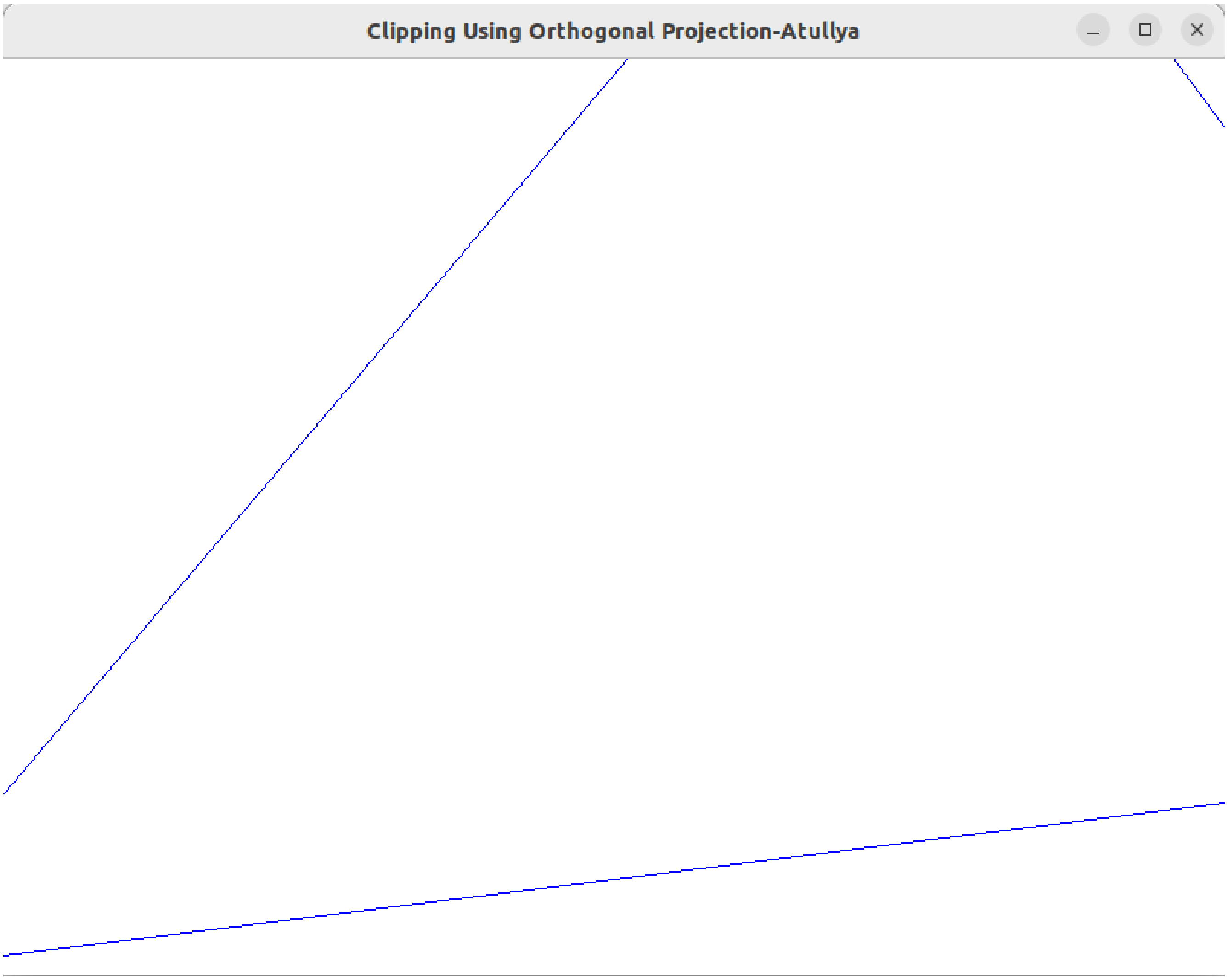
Orthogonal Projection Example-Atullya

20. Write an OPENGL app to show clipping using orthogonal projection.

Source Code

```
#include <GL/glut.h>

int win_width = 800;
int win_height = 600;

void display() {
   glClear(GL_COLOR_BUFFER_BIT);

   // Set up the clipping region using glOrtho
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-0.4, 0.4, -0.4, 0.4, -1.0, 1.0);

   // Draw the original non-filled triangle
   glColor3f(0.0f, 0.0f, 1.0f);
   glBegin(GL_LINE_LOOP);
   glVertex2f(-0.5f, -0.4f);
   glVertex2f(0.7f, -0.2f);
   glVertex2f(0.2f, 0.7f);
   glEnd();

   glutSwapBuffers();
}

void reshape(int w, int h) {
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}
```

```
int main(int argc, char *argv[]) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(win_width, win_height);
   glutInitWindowPosition(100, 100);
   glutCreateWindow("Clipping Using Orthogonal Projection-Atullya");

   glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

   glutDisplayFunc(display);
   glutReshapeFunc(reshape);

   glutMainLoop();

   return 0;
}
```



Clipping Using Orthogonal Projection-Atullya

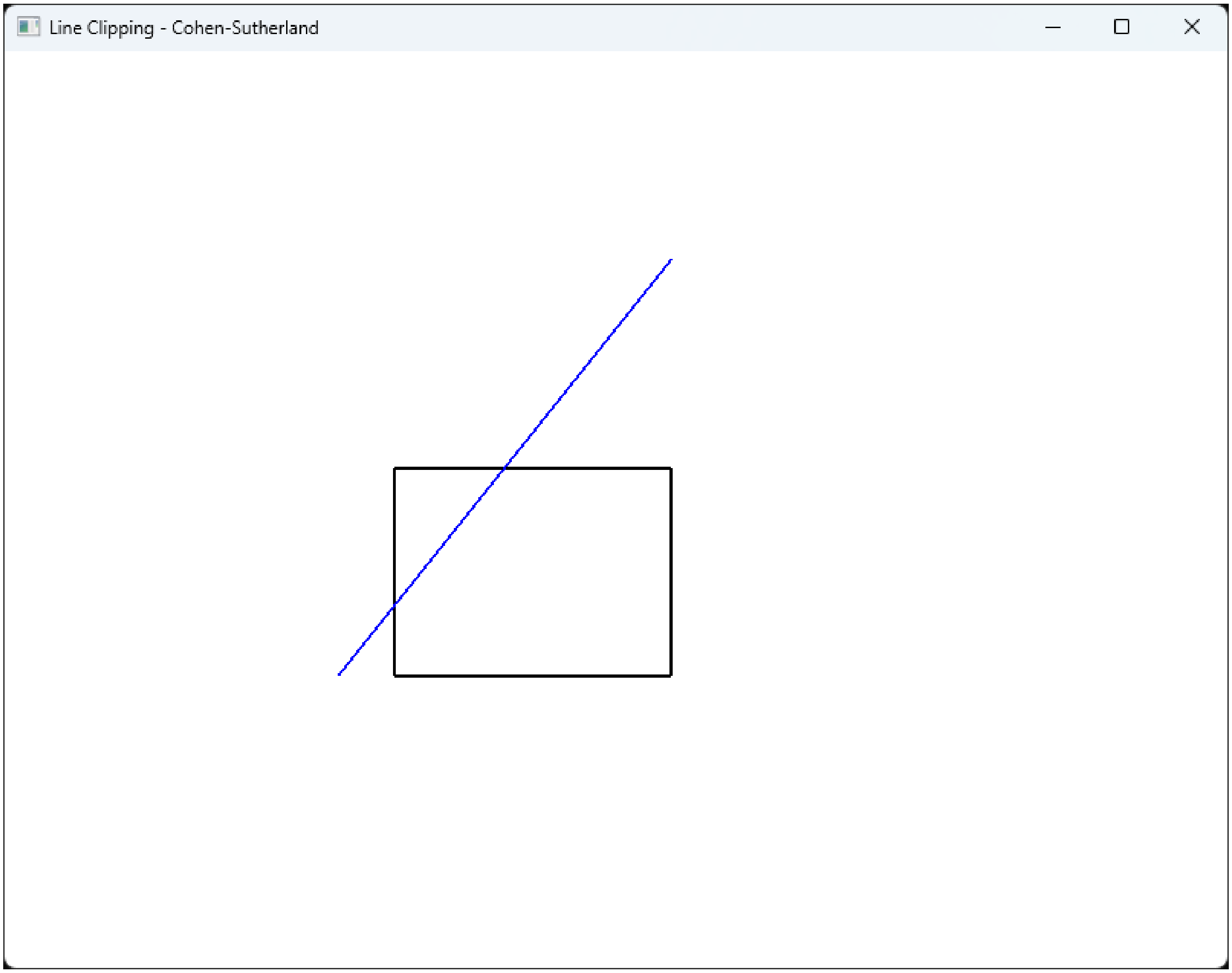
22. Implement sutherland cohen clipping algorithm to clip the given line.

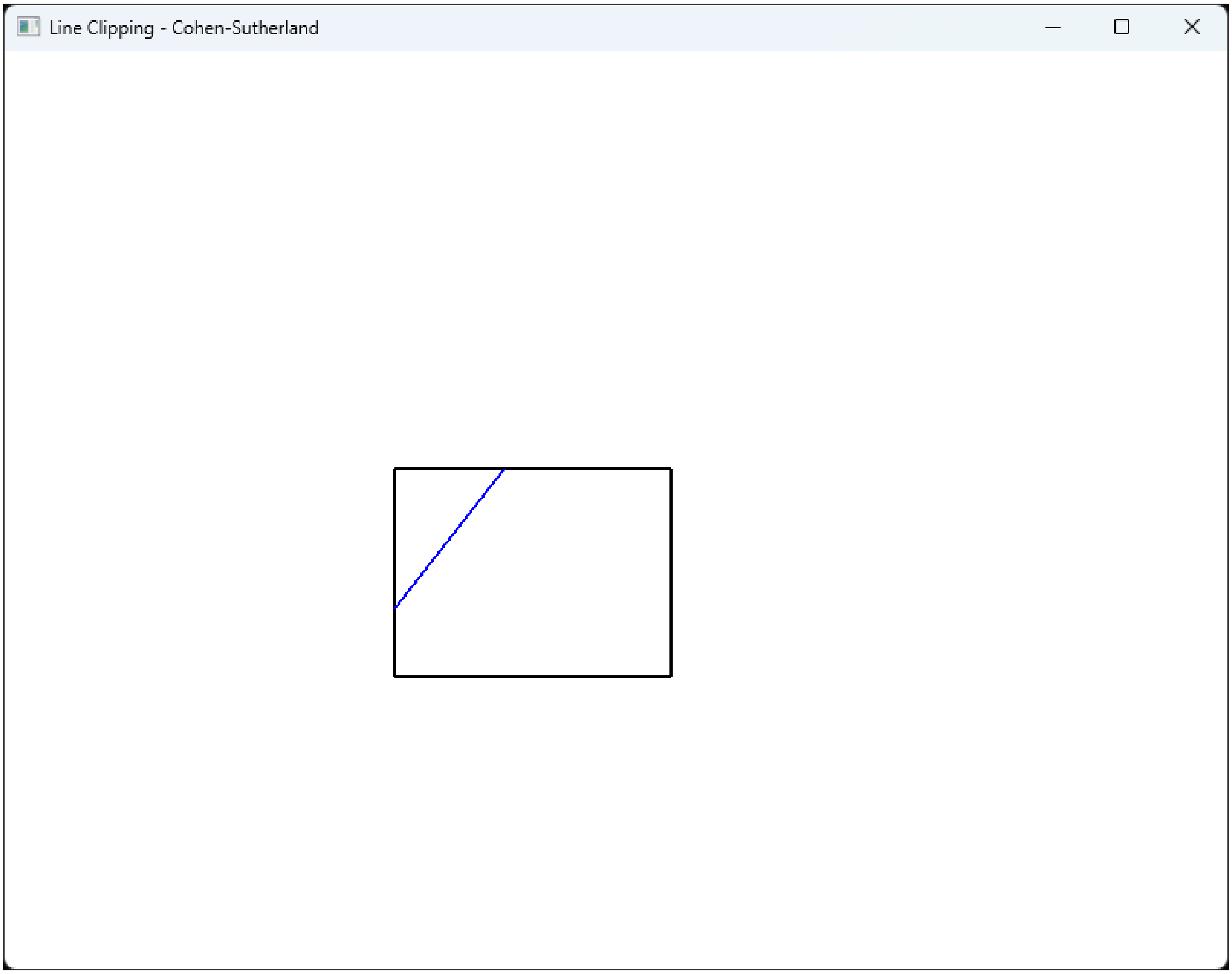
Source Code

```
#include <GL/glut.h>

int wx_max = 100, wy_max = 100, wx_min = 50, wy_min = 50;
int x_1 = 40, x_2 = 100, y_1 = 50, y_2 = 150;

int getCode(int x, int y)
{
   int code = 0000;

   if (x < wx_min)
      code |= 1;
   else if (x > wx_max)
      code |= 2;
   if (y < wy_min)
      code |= 4;
   else if (y > wy_max)
      code |= 8;

   return code;
}
void drawline(int x_1, int y_1, int x_2, int y2)
{
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0.0f, 0.0f, 0.0f); // black
   glLineWidth(2.0);
   // Draw the window
   glBegin(GL_LINE_LOOP);
   glVertex2i(wx_min, wy_min);
   glVertex2i(wx_max, wy_min);
   glVertex2i(wx_max, wy_max);
   glVertex2i(wx_min, wy_max);
   glEnd();
```

```
    // Draw the line
    glColor3f(0.0f, 0.0f, 1.0f); // blue
    glBegin(GL_LINES);
    glVertex2i(x_1, y_1);
    glVertex2i(x_2, y2);
    glEnd();
    glFlush();
}
void cohen_sutherland()
{

    int code1 = getCode(x_1, y_1);
    int code2 = getCode(x_2, y_2);
    bool accept = false;

    do
    {
        // Check if the line is completely inside or outside the window
        if ((code1 == 0) && (code2 == 0))
        {
            accept = true;
            break;
        }
        else if (code1 & code2)
        {
            break;
        }
        else
        {
            glColor3f(1.0f, 0.0f, 0.0f); // red

            // Clipping variables
            int code;
            float m = (float)(y_2 - y_1) / (float)(x_2 - x_1);
            float c = y_1 - m * x_1;
            float x, y;

            // Find the intersection points
            if (code1 != 0)
                code = code1;
            else
                code = code2;

            if (code & 1) // Left boundary
            {
                y = m * wx_min + c;
                x = wx_min;
            }
            else if (code & 2) // Right boundary
            {
                y = m * wx_max + c;
                x = wx_max;
            }
            else if (code & 4) // Bottom boundary
            {
                x = (wy_min - c) / m;
                y = wy_min;
            }
            else if (code & 8) // Top boundary
            {
                x = (wy_max - c) / m;
                y = wy_max;
            }
            if (code == code1)
            {
                x_1 = x;
                y_1 = y;
                code1 = getCode(x_1, y_1);
            }
            else
            {
```

```
                x_2 = x;
                y_2 = y;
                code2 = getCode(x_2, y_2);
            }
        }
    } while (accept == false);
}
void display()
{
    drawline(x_1, y_1, x_2, y_2);
    Sleep(1000);
    cohen_sutherland();
    drawline(x_1, y_1, x_2, y_2);
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Line Clipping - Cohen-Sutherland");
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-20, 200, -20, 200);
    glMatrixMode(GL_MODELVIEW);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```


Line Clipping - Cohen-Sutherland


Line Clipping - Cohen-Sutherland