Name:Harshada Gopal Rayate
Roll No:19 SEDA
Subject :CGAVR

Polygon line Clipping

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

#define INSIDE 0 // 0000
#define LEFT 1 // 0001
#define RIGHT 2 // 0010
#define BOTTOM 4 // 0100
#define TOP 8 // 1000

typedef struct {
float x, y;
} Point;

float winMinX = 100, winMinY = 100, winMaxX = 400, winMaxY = 400; // Clipping window
coordinates

// Function to compute the region code for a point
int computeCode(float x, float y) {
int code = INSIDE;
if (x < winMinX) // to the left of the window
code |= LEFT;
else if (x > winMaxX) // to the right of the window
code |= RIGHT;
if (y < winMinY) // below the window
code |= BOTTOM;
else if (y > winMaxY) // above the window
code |= TOP;
return code;
}

// Cohen-Sutherland line clipping algorithm
void cohenSutherlandClip(Point p1, Point p2) {
int code1 = computeCode(p1.x, p1.y);
int code2 = computeCode(p2.x, p2.y);
int accept = 0;

while (1) {
if ((code1 == 0) && (code2 == 0)) {
accept = 1; // Both points are inside
break;
} else if (code1 & code2) {
break; // Both points are outside
} else {
```

```
int codeOut;
float x, y;

// Choose one of the points outside the clipping rectangle
if (code1 != 0)
codeOut = code1;
else
codeOut = code2;

// Find the intersection point using the clip edge
if (codeOut & TOP) { // point is above the clip rectangle
x = p1.x + (p2.x - p1.x) * (winMaxY - p1.y) / (p2.y - p1.y);
y = winMaxY;
} else if (codeOut & BOTTOM) { // point is below the clip rectangle
x = p1.x + (p2.x - p1.x) * (winMinY - p1.y) / (p2.y - p1.y);
y = winMinY;
} else if (codeOut & RIGHT) { // point is to the right of clip rectangle
y = p1.y + (p2.y - p1.y) * (winMaxX - p1.x) / (p2.x - p1.x);
x = winMaxX;
} else if (codeOut & LEFT) { // point is to the left of clip rectangle
y = p1.y + (p2.y - p1.y) * (winMinX - p1.x) / (p2.x - p1.x);
x = winMinX;
}

// Now intersection point x,y is found
if (codeOut == code1) {
p1.x = x;
p1.y = y;
code1 = computeCode(p1.x, p1.y);
} else {
p2.x = x;
p2.y = y;
code2 = computeCode(p2.x, p2.y);
}
}
}

if (accept) {
glColor3f(0.0, 0.0, 1.0); // Color for accepted line
glBegin(GL_LINES);
glVertex2f(p1.x, p1.y);
glVertex2f(p2.x, p2.y);
glEnd();
}
}

// Function to draw the clipping window and the initial line
void display() {
glClear(GL_COLOR_BUFFER_BIT);
```

```
// Draw clipping window
glColor3f(0.0, 0.0, 0.0); // Black color for window border
glBegin(GL_LINE_LOOP);
glVertex2f(winMinX, winMinY);
glVertex2f(winMaxX, winMinY);
glVertex2f(winMaxX, winMaxY);
glVertex2f(winMinX, winMaxY);
glEnd();

// Initial line coordinates
Point p1 = {50, 250}; // Start point outside left of clipping window
Point p2 = {450, 250}; // End point outside right of clipping window

glColor3f(1.0, 0.0, 0.0); // Red color for original line
glBegin(GL_LINES);
glVertex2f(p1.x, p1.y);
glVertex2f(p2.x, p2.y);
glEnd();

cohenSutherlandClip(p1, p2); // Call clipping function

glFlush();
}

// Initialization function
void init() {
glClearColor(1.0, 1.0, 1.0, 1.0); // Set background color to white
gluOrtho2D(0.0, 500.0, 0.0, 500.0); // Set up orthographic projection
}

int main(int argc, char** argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(500, 500);
glutCreateWindow("Cohen-Sutherland Line Clipping Algorithm");

init();
glutDisplayFunc(display); // Register display callback function
glutMainLoop(); // Enter the event-processing loop
return 0;

}
```