

Rohan Nyati

500075940

R177219148

Batch-5 ( Ai & MI )

## **EXPERIMENT-6**

### **CODE :-**

```
#include<iostream>
#include<windows.h>
#include<GL/glut.h>
using namespace std;

const int MAX_POINTS = 20;
GLint count = 0;

void init(void)
{
    glClearColor(1.0,1.0,1.0,0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-1000,1000,-1000,1000);
}

void plotline(float a,float b,float c,float d)
{
    glBegin(GL_LINES);
    glVertex2i(a,b);
```

```
glVertex2i(c,d);
```

```
glEnd();
```

```
}
```

```
// Returns x-value of point of intersection of two lines
```

```
int x_intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4)
```

```
{
```

```
    int num = (x1*y2 - y1*x2) * (x3-x4) - (x1-x2) * (x3*y4 - y3*x4);
```

```
    int den = (x1-x2) * (y3-y4) - (y1-y2) * (x3-x4);
```

```
    return num/den;
```

```
}
```

```
// Returns y-value of point of intersection of two lines
```

```
int y_intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4)
```

```
{
```

```
    int num = (x1*y2 - y1*x2) * (y3-y4) - (y1-y2) * (x3*y4 - y3*x4);
```

```
    int den = (x1-x2) * (y3-y4) - (y1-y2) * (x3-x4);
```

```
    return num/den;
```

```
}
```

```
// This functions clips all the edges w.r.t one clip edge of clipping area
```

```
void clip(int poly_points[][2], int &poly_size, int x1, int y1, int x2, int y2)
```

```
{
```

```
    int new_points[MAX_POINTS][2], new_poly_size = 0;
```

```
    // (ix,iy),(kx,ky) are the co-ordinate values of the points
```

```
    for (int i = 0; i < poly_size; i++)
```

```
    {
```

```
        // i and k form a line in polygon
```

```
int k = (i+1) % poly_size;
int ix = poly_points[i][0], iy = poly_points[i][1];
int kx = poly_points[k][0], ky = poly_points[k][1];
```

```
// Calculating position of first point
```

```
// w.r.t. clipper line
```

```
int i_pos = (x2-x1) * (iy-y1) - (y2-y1) * (ix-x1);
```

```
// Calculating position of second point
```

```
// w.r.t. clipper line
```

```
int k_pos = (x2-x1) * (ky-y1) - (y2-y1) * (kx-x1);
```

```
// Case 1 : When both points are inside
```

```
if (i_pos < 0 && k_pos < 0)
```

```
{
```

```
    //Only second point is added
```

```
    new_points[new_poly_size][0] = kx;
```

```
    new_points[new_poly_size][1] = ky;
```

```
    new_poly_size++;
```

```
}
```

```
// Case 2: When only first point is outside
```

```
else if (i_pos >= 0 && k_pos < 0)
```

```
{
```

```
    // Point of intersection with edge
```

```
    // and the second point is added
```

```
    new_points[new_poly_size][0] = x_intersect(x1,
```

```
        y1, x2, y2, ix, iy, kx, ky);
```

```
    new_points[new_poly_size][1] = y_intersect(x1,
```

```

        y1, x2, y2, ix, iy, kx, ky);
    new_poly_size++;

    new_points[new_poly_size][0] = kx;
    new_points[new_poly_size][1] = ky;
    new_poly_size++;
}

// Case 3: When only second point is outside
else if (i_pos < 0 && k_pos >= 0)
{
    //Only point of intersection with edge is added
    new_points[new_poly_size][0] = x_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
    new_points[new_poly_size][1] = y_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
    new_poly_size++;
}

// Case 4: When both points are outside
else
{
    //No points are added
}
}

// Copying new points into original array and changing the no. of vertices
poly_size = new_poly_size;
for (int i = 0; i < poly_size; i++)
{
    poly_points[i][0] = new_points[i][0];

```

```

        poly_points[i][1] = new_points[i][1];
    }
}

```

// Implements Sutherland-Hodgman algorithm

```

void suthHodgClip(int poly_points[][2], int poly_size, int clipper_points[][2], int clipper_size)

```

```

{
    //i and k are two consecutive indexes
    for (int i=0; i<clipper_size; i++)
    {
        int k = (i+1) % clipper_size;

        // We pass the current array of vertices, it's size
        // and the end points of the selected clipper line
        clip(poly_points, poly_size, clipper_points[i][0],
            clipper_points[i][1], clipper_points[k][0],
            clipper_points[k][1]);
    }
}

```

// Printing vertices of clipped polygon

```

for (int i=0; i < poly_size; i++)
{
    glColor3f(0.0,0.0,0.0);
    if(i!=(poly_size-1))
    {
        glBegin(GL_LINES);
        glVertex2i(poly_points[i][0],poly_points[i][1]);
        glVertex2i(poly_points[i+1][0],poly_points[i+1][1]);
        glEnd();
    }
}

```

```

    }
    else
    {
        glBegin(GL_LINES);
        glVertex2i(poly_points[i][0],poly_points[i][1]);
        glVertex2i(poly_points[0][0],poly_points[0][1]);
        glEnd();
    }
}
}

```

```

void mousePtPlot(GLint button, GLint action, GLint xMouse, GLint yMouse)
{
    if(button == GLUT_LEFT_BUTTON && action == GLUT_UP)
    {
        if(!count)
        {
            int poly_size = 8;

            int poly_points[20][2] = {{-450,0},{-450,800},{0,800},{0,500},{-350,700},{-350,200},{-
200,200},{-200,0}};

            // Defining clipper polygon vertices in clockwise order
            // 1st Example with square clipper
            int clipper_size = 4;
            int clipper_points[][2] = {{-300,100},{-300,600},{200,600},{200,100}};

            //Calling the clipping function
            suthHodgClip(poly_points, poly_size, clipper_points, clipper_size);

            count++;

            cout<<"Polygon clipped\n";

```

```
    glFlush();  
}  
}  
if(button == GLUT_RIGHT_BUTTON && action == GLUT_UP)  
{  
    exit(0);  
}  
}
```

```
void display()
```

```
{  
  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(0.0,1.0,0.0);  
  
    glBegin(GL_LINE_LOOP);  
        glVertex2i(-300,100);  
        glVertex2i(200,100);  
        glVertex2i(200,600);  
        glVertex2i(-300,600);  
    glEnd();  
  
    glColor3f(1.0,0.0,0.0);  
    glBegin(GL_LINE_LOOP);  
        glVertex2i(-450,0);  
        glVertex2i(-200,0);  
        glVertex2i(-200,200);  
        glVertex2i(-350,200);  
        glVertex2i(-350,700);  
    glEnd();  
}
```

```

        glVertex2i(0,500);
        glVertex2i(0,800);
        glVertex2i(-450,800);
    glEnd();

    glFlush();

}

int main(int argc,char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Sutherland – Hodgeman ");
    glutMouseFunc(mousePtPlot);
    glutDisplayFunc(display);

    init();

    glutMainLoop();

    return 0;
}

```

**OUTPUT :-**



