Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

Polygon filling program using 4 connected algorithm

Code:

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

void init(){
   glClearColor(1.0,1.0,1.0,0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0,500,0,500);
}

void bound_it(int x, int y, float* fillColor, float* bc) {
   float color[3];
   glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,color);
   if((color[0]!=bc[0]||color[1]!=bc[1]||color[2]!=bc[2])&&(color[0]!=fillColor[0] ||
color[1]!=fillColor[1] || color[2]!=fillColor[2])){
      glColor3f(fillColor[0],fillColor[1],fillColor[2]);
      glBegin(GL_POINTS);
      glVertex2i(x,y);
      glEnd();
      glFlush();
      bound_it(x+1,y,fillColor,bc);
      bound_it(x-1,y,fillColor,bc);
      bound_it(x,y+1,fillColor,bc);
      bound_it(x,y-1,fillColor,bc);
   }
}

void world(){
   int N,i,a,b;
   glLineWidth(2);
   glPointSize(1);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1,0,0);
   glBegin(GL_LINE_LOOP);
   printf("Enter number of vertices: ");
   scanf("%d",&N);
   printf("Enter vertices in anticlockwise manner:\n");
   for (i=0;i<N;i++)
   {
      printf("Enter X-Coordinate for %d vertex: ",(i+1));
      scanf("%d",&a);
      printf("Enter Y-Coordinate for %d vertex: ",(i+1));
      scanf("%d",&b);
      glVertex2i(a,b);
```

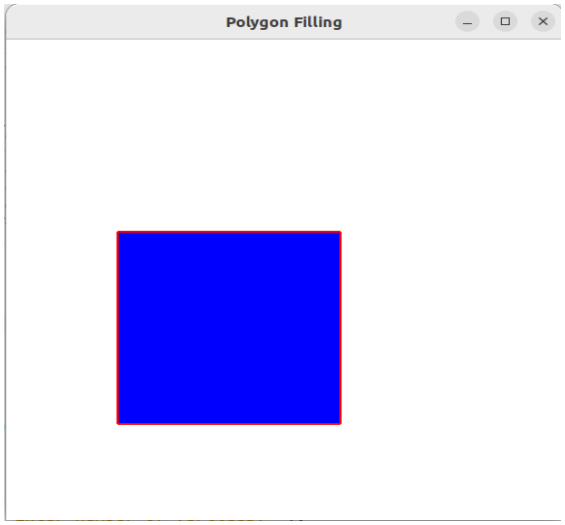
Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

```
    }
    glEnd();
    glFlush();
    float bCol[] = {1,0,0};
    float color[] = {0,0,1};
    bound_it(a+2,b-2,color,bCol);
}

int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(200,200);
    glutCreateWindow("Polygon Filling");
    glutDisplayFunc(world);
    init();
    glutMainLoop();
    return 0;
}
```

Output:

```
himesh@Ubuntu22:~/Downloads$ gcc polygon.c -o polygon -lGL -lGLU -lglut
^[[Ahimesh@Ubuntu22:~/Downloa./polygon
Enter number of vertices: 4
Enter vertices in anticlockwise manner:
Enter X-Coordinate for 1 vertex: 100
Enter Y-Coordinate for 1 vertex: 100
Enter X-Coordinate for 2 vertex: 300
Enter Y-Coordinate for 2 vertex: 100
Enter X-Coordinate for 3 vertex: 300
Enter Y-Coordinate for 3 vertex: 300
Enter X-Coordinate for 4 vertex: 100
Enter Y-Coordinate for 4 vertex: 300
```



Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

Polygon filling program using 8 connected algorithm

Code:

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

void init(){
   glClearColor(1.0,1.0,1.0,0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0,500,0,500);
}

void bound_it(int x, int y, float* fillColor, float* bc) {
   float color[3];
   glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,color);
   if((color[0]!=bc[0]||color[1]!=bc[1]||color[2]!=bc[2])&&(color[0]!=fillColor[0] ||
color[1]!=fillColor[1] || color[2]!=fillColor[2])){
      glColor3f(fillColor[0],fillColor[1],fillColor[2]);
      glBegin(GL_POINTS);
      glVertex2i(x,y);
      glEnd();
      glFlush();
      bound_it(x+1,y,fillColor,bc);
      bound_it(x-1,y,fillColor,bc);
      bound_it(x,y+1,fillColor,bc);
      bound_it(x,y-1,fillColor,bc);
      bound_it(x+1,y+1,fillColor,bc);
      bound_it(x-1,y-1,fillColor,bc);
      bound_it(x+1,y-1,fillColor,bc);
      bound_it(x-1,y+1,fillColor,bc);
   }
}

void world(){
   int N,i,a,b;
   glLineWidth(2);
   glPointSize(1);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1,0,0);
   glBegin(GL_LINE_LOOP);
   printf("Enter number of vertices: ");
   scanf("%d",&N);
   printf("Enter vertices in anticlockwise manner:\n");
   for (i=0;i<N;i++)
   {
         printf("Enter X-Coordinate for %d vertex: ",(i+1));
```

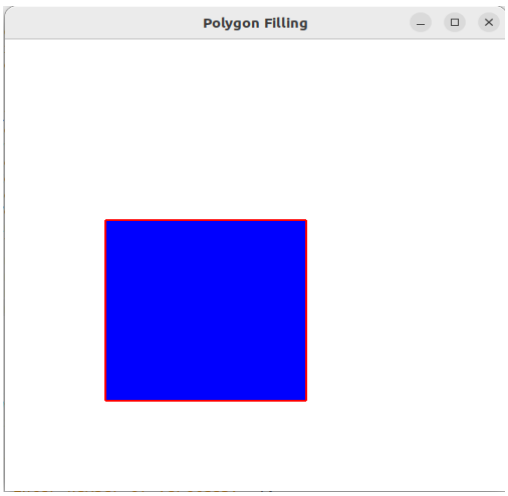
Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

```
        scanf("%d",&a);
        printf("Enter Y-Coordinate for %d vertex: ",(i+1));
        scanf("%d",&b);
        glVertex2i(a,b);
    }
    glEnd();
    glFlush();
    float bCol[] = {1,0,0};
    float color[] = {0,0,1};
    bound_it(a+2,b-2,color,bCol);
}

int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(200,200);
    glutCreateWindow("Polygon Filling");
    glutDisplayFunc(world);
    init();
    glutMainLoop();
    return 0;
}
```

Output:

```
himesh@Ubuntu22:~/Downloads$ gcc polygon2.c -o polygon2 -lGL -lGLU -lglut
himesh@Ubuntu22:~/Downloads$ ./polygon2
Enter number of vertices: 4
Enter vertices in anticlockwise manner:
Enter X-Coordinate for 1 vertex: 100
Enter Y-Coordinate for 1 vertex: 100
Enter X-Coordinate for 2 vertex: 300
Enter Y-Coordinate for 2 vertex: 100
Enter X-Coordinate for 3 vertex: 300
Enter Y-Coordinate for 3 vertex: 300
Enter X-Coordinate for 4 vertex: 100
Enter Y-Coordinate for 4 vertex: 300
```



Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

Polygon filling program using scan line algorithm

Code:

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

using namespace std;

int n;
struct Point
{
   int x, y;
};

Point polygon[100];

bool cmp(Point a, Point b)
{
   return (a.y < b.y || (a.y == b.y && a.x < b.x));
}
void init()
{
   glClearColor(1.0,1.0,1.0,0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0,500,0,500);
   glColor3f(0.0, 0.0, 0.0);
   glPointSize(1.0);
   glLoadIdentity();
}

void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0, 0.0, 0.0);
   glBegin(GL_POLYGON);
   for (int i = 0; i < n; i++)
      glVertex2i(polygon[i].x, polygon[i].y);
   glEnd();
   glFlush();
}

int findYMin()
{
   int ymin = polygon[0].y, min = 0;
   for (int i = 1; i < n; i++)
   {
      int y = polygon[i].y;
```

Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

```
        if ((y < ymin) || (ymin == y && polygon[i].x < polygon[min].x))
            ymin = polygon[i].y, min = i;
    }
    return min;
}

void fillScanLine(int yMin, int x1, int x2, int y1, int y2)
{
    if (y1 > y2)
        swap(y1, y2), swap(x1, x2);
    for (int i = y1; i < y2; i++)
    {
        int xIntersect = x1 + (i - y1) * (x2 - x1) / (y2 - y1);
        glBegin(GL_POINTS);
        glVertex2i(xIntersect, i);
        glEnd();
        glFlush();
    }
}

void scanLineFill(void)
{
    sort(polygon, polygon + n, cmp);

    int yMin = polygon[0].y;
    int yMax = polygon[n - 1].y;

    int yStart, yEnd, xStart, xEnd;
    int yCur = yMin;

    while (yCur <= yMax)
    {
        for (int i = 0; i < n; i++)
        {
            int j = (i + 1) % n;
            int sl = (polygon[i].y < polygon[j].y) ? i : j;
            int el = (sl == i) ? j : i;

            if (yCur >= polygon[sl].y && yCur < polygon[el].y)
            {
                xStart = polygon[sl].x + (yCur - polygon[sl].y) * (polygon[el].x - polygon[sl].x) /
(polygon[el].y - polygon[sl].y);

                if (xStart > polygon[el].x)
                    swap(xStart, polygon[el].x);

                fillScanLine(yMin, xStart, polygon[el].x, yCur, polygon[el].y);
            }
```

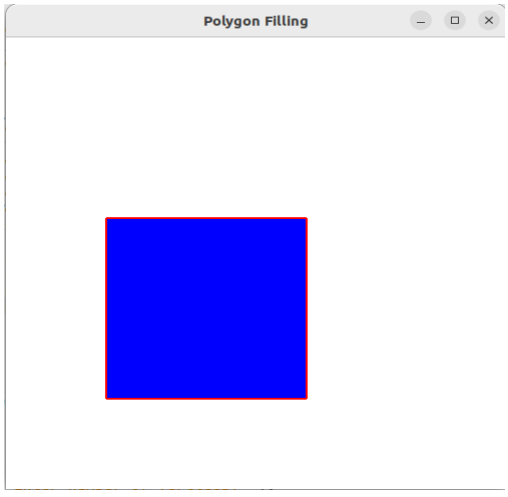
Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

```
        }
        yCur++;
    }
}

int main(int argc, char **argv)
{
    cout << "Enter the number of vertices of polygon: ";
    cin >> n;
    cout << "Enter the vertices of polygon in counter-clockwise order:\n";
    for (int i = 0; i < n; i++)
        cin >> polygon[i].x >> polygon[i].y;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(200,200);
    glutCreateWindow("Polygon Filling");
    glutDisplayFunc(scanLineFill);
    init();
    glutMainLoop();
    return 0;
}
```

Output:



Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

Line clipping program using Cohen-Sutherland algorithm

Code:

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

using namespace std;

const int INSIDE = 0;
const int LEFT = 1;
const int RIGHT = 2;
const int BOTTOM = 4;
const int TOP = 8;

int x_min, y_min;
int x_max, y_max;

int computeCode(double x, double y)
{
   int code = INSIDE;
   if (x < x_min)
      code |= LEFT;
   else if (x > x_max)
      code |= RIGHT;

   if (y < y_min)
      code |= BOTTOM;
   else if (y > y_max)
      code |= TOP;
   return code;
}
void cohenSutherlandClip(double x1, double y1, double x2, double y2)
{
   int code1 = computeCode(x1, y1);
   int code2 = computeCode(x2, y2);
   while (true)

   {
      if (!(code1 | code2))

      {
         glColor3f(0, 0, 1);
         glBegin(GL_LINES);
         glVertex2f(x1, y1);
         glVertex2f(x2, y2);
         glEnd();
         glFlush();
         break;
```

Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

```
    }
    else if (code1 & code2)

    {
       break;
    }
    else
    {
       double x, y;
       int code = code1 ? code1 : code2;
       if (code & TOP)

       {
          x = x1 + (x2 - x1) * (y_max - y1) / (y2 - y1);
          y = y_max;
       }
       else if (code & BOTTOM)

       {
          x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1);
          y = y_min;
       }
       else if (code & RIGHT)

       {

          y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1);
          x = x_max;
       }
       else if (code & LEFT)

       {
          y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1);
          x = x_min;
       }
       if (code == code1)

       {
          x1 = x;
          y1 = y;
          code1 = computeCode(x1, y1);
       }
       else
       {
          x2 = x;
          y2 = y;
          code2 = computeCode(x2, y2);
       }
    }
```

Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

```
    }
}
void display()
{
    cout<<"Enter Minimum window co-ordinates: ";
    cin>>x_min>>y_min;
    cout<<"Enter Maximum window co-ordinates: ";
    cin>>x_max>>y_max;
    double x1, y1;
    double x2, y2;
    cout<<"Enter co-ordinates of first point of line: ";
    cin>>x1>>y1;
    cout<<"Enter co-ordinates of second point of line: ";
    cin>>x2>>y2;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(x_min, y_min);
    glVertex2f(x_max, y_min);
    glVertex2f(x_max, y_max);
    glVertex2f(x_min, y_max);
    glEnd();
    glColor3f(0, 1, 0);
    glBegin(GL_LINES);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glEnd();
    glFlush();
    cohenSutherlandClip(x1, y1, x2, y2);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(680, 500);
    glutInitWindowPosition(200, 200);
    glutCreateWindow("Cohen Sutherland Line Clipping Algorithm");
    glutDisplayFunc(display);
    gluOrtho2D(-500, 500, -500, 500);
    glutMainLoop();
    return 0;
}
```

Lab Assignment-2
Submitted By:
Name: Himesh Maniyar
ID: 2020UCP1776

Output:

```
┌──(himesh㉿root)-[~/Desktop/CG Lab2]
└─$ g++ clip2.cpp -o clip2 -lGL -lGLU -lglut

┌──(himesh㉿root)-[~/Desktop/CG Lab2]
└─$ ./clip2
Enter Minimum window co-ordinates: -230 -139
Enter Maximum window co-ordinates: 433 221
Enter co-ordinates of first point of line: -340 -116
Enter co-ordinates of second point of line: 464 267
```