AISSMS
COLLEGE OF ENGINEERING
Approved by AICTE, New Delhi, Recognized by
Govt. of Maharashtra, Affiliated to Savitribai Phule Pune University
and recognized 2(f) and 12(B) by UGC (Id.No. PU / PN/ Engg. / 093 (1992)
Accredited by NAAC with 'A+' Grade

Name :- Pratik Pingale *Class :- SE Comp. 1*

Roll no. :- 19CO056

Experiment A-5

Aim :

To write the c++ program to draw the concave polygon and fill it with the desired colour using scan fill algorithm.

Algorithm:

Step 1: Read n, number of vertices of polygon.

Step 2: read x and y coordinates of all vertices in array x[n] and y[n].

Step 3: find Ymin and Ymax.

Step 4: store the initial x value(X1) y values Y1 and Y2 for two endpoints and x increment Δx for scan line to scan line for each edges in array edges. [n] [4]

While doing this check that y1>y2, if not interchange Y1 and Y2 and corresponding x1 and x2 so that for each edge Y1 represents its maximum Y coordinate and Y2 represents its minimum y coordinate.

Step 5: sort the rows of arrays, edges [n] [4] in descending order of y1, descending order of y2 , and ascending order of X1.

Step 6: Set y=ymax

Step 7: find active edges and update active edge list,

If(y>y2 and y>=y1)

{Edge is active}

Else

{Edge is not active }

Step 8: Compute the X intersects for all active edges for current y values [initially x intersects is X1 and x intersects for successive y values can be given as,

Xi+1 xi+ΔX ←

Where Δx = -1/m and m= y2-y1/x2-x1 i.e. slope of a line segment

Step 9: if x intersect vertex i.e. x-intersect=x1 and y=y1 then apply vertex test to check whether they consider the one intersect or two intersects. store all the intersects in x intersect [] array.

Step 10: Sort x intersect [] array in ascending order,

Step 11: Extract pairs of intersects from the sorted from the sorted x- intersect[] array.

Step 12: pass pair of x values to line drawing routine to draw corresponding line segment

Step 13: set y = y-1;

Step14: repeat step 7 through 13 until y>=ymin.

Step 15: STOP

Program :

```
#include <iostream>

#include <graphics.h>
using namespace std;

class point {
    public:
        int x, y;
};

class poly {
    private:
        point p[20];
        int inter[20], x, y;
        int v, xmin, ymin, xmax, ymax;
    public:
        int c;
        void read();
        void calcs();
        void display();
        void ints(float);
        void sort(int);
};

void poly::read() {
    int i;
    cout << "Enter the no of vertices of polygon:" << endl;
    cin >> v;
    if (v > 2) {
        for (i = 0; i < v; i++) {
            cout << "Enter the co-ordinate no." << i + 1 << ": ";
            cout << "x" << (i + 1) << " , y" << (i + 1) << "=";
            cin >> p[i].x >> p[i].y;
        }
        p[i].x = p[0].x;
        p[i].y = p[0].y;
        xmin = xmax = p[0].x;
        ymin = ymax = p[0].y;
    } else
        cout << " Enter valid no. of vertices : " << endl;
}

void poly::calcs() {
    for (int i = 0; i < v; i++) {
        if (xmin > p[i].x)
            xmin = p[i].x;
        if (xmax < p[i].x)
            xmax = p[i].x;
        if (ymin > p[i].y)
            ymin = p[i].y;
        if (ymax < p[i].y)
            ymax = p[i].y;
```

```
    }
}

void poly::display() {
    int ch1;
    char ch = 'y';
    float s, s2;
    s = ymin + 0.01;
    delay(100);
    cleardevice();
    while (s <= ymax) {
        ints(s);
        sort(s);
        s++;
    }
}

void poly::ints(float z) {
    int x1, x2, y1, y2, temp;
    c = 0;
    for (int i = 0; i < v; i++) {
        x1 = p[i].x;
        y1 = p[i].y;
        x2 = p[i + 1].x;
        y2 = p[i + 1].y;
        if (y2 < y1) {
            temp = x1;
            x1 = x2;
            x2 = temp;
            temp = y1;
            y1 = y2;
            y2 = temp;
        }
        if (z <= y2 && z >= y1) {
            if ((y1 - y2) == 0)
                x = x1;
            else {
                x = ((x2 - x1) * (z - y1)) / (y2 - y1);
                x = x + x1;
            }
            if (x <= xmax && x >= xmin)
                inter[c++] = x;
        }
    }
}

void poly::sort(int z) {
    int temp, j, i;
    for (i = 0; i < v; i++) {
        line(p[i].x, p[i].y, p[i + 1].x, p[i + 1].y);
    }
    delay(100);
    for (i = 0; i < c; i += 2) {
        delay(100);
```

```
        line(inter[i], z, inter[i + 1], z);
    }
}

int main() {
    int cl;
    int gd = DETECT, gm;

    initgraph( & gd, & gm, NULL);

    poly x;
    x.read();
    x.calcs();
    cleardevice();

    cout << "Enter the colour u want:(0-15) : ";
    cin >> cl;

    setcolor(cl);
    x.display();
    delay(1000);

    closegraph();
    getch();
    return 0;
}
```

Output :

```
proxima@proxzima:~/Documents/Extras/CG/Practicals

~/Doc/E/C/Practicals
 g++ -g "/home/proxima/Documents/Extras/CG/Practicals/scan_line_fill.cpp" -o "/home/proxima/Documents/Extras/CG/Practicals/scan_line_fill" -lgraph -lGL -lGLU -lglut && "/home/proxima/Documents/Extras/CG/Practicals/scan_line_fill"
Enter the no of vertices of polygon:
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
scan_line_fill: ../../src/xcb_io.c:260: poll_for_event: Assertion `!xcb_xlib_threads_sequence_lost' failed.
5
Enter the co-ordinate no.1: x1 , y1=20 20
Enter the co-ordinate no.2: x2 , y2=20 300
Enter the co-ordinate no.3: x3 , y3=200 125
Enter the co-ordinate no.4: x4 , y4=300 300
Enter the co-ordinate no.5: x5 , y5=300 20
Enter the colour u want:(0-15) : 3

[1]  + 42286 suspended  "/home/proxima/Documents/Extras/CG/Practicals/scan_line_fill"

~/Doc/E/C/Practicals
```