Assignment No: 4

Title/ Problem Statement:

Write C++/Java program to fill polygon using scan line algorithm. Use mouse interfacing to draw polygon.

Objectives:

1. To understand Scan Line Polygon filling algorithms used for computer graphics.
2. To understand Scan Line Concept

Aim: To implement the Scan line polygon fill algorithm for coloring a given object.

Theory :

The basic scan-line algorithm is as follows:

- Find the intersections of the scan line with all edges of the polygon
- Sort the intersections by increasing x coordinate
- Fill in all pixels between pairs of intersections that lie interior to the polygon

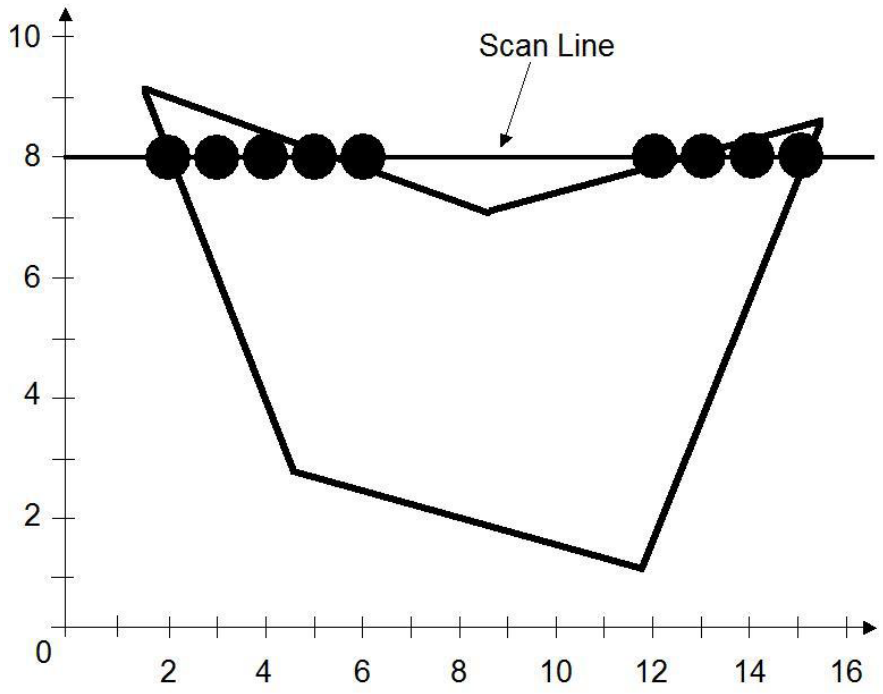
Process involved:

The scan-line polygon-filling algorithm involves

• the horizontal scanning of the polygon from its lowermost to its topmost vertex,

• identifying which edges intersect the scan-line,

• and finally drawing the interior horizontal lines with the specified fill color process.



Algorithm Steps:

1. The horizontal scanning of the polygon from its lowermost to its topmost vertex
2. Identify the edge intersections of scan line with polygon
3. Build the edge table
a. Each entry in the table for a particular scan line contains the maximum y value for that edge, the x-intercept value (at the lower vertex) for the edge, and the inverse slope of the edge.
4. Determine whether any edges need to be spitted or not. If there is need to split, split the edges.
5. Add new edges and build modified edge table.
6. Build Active edge table for each scan line and fill the polygon based on intersection of scan line with polygon edges.

Program:

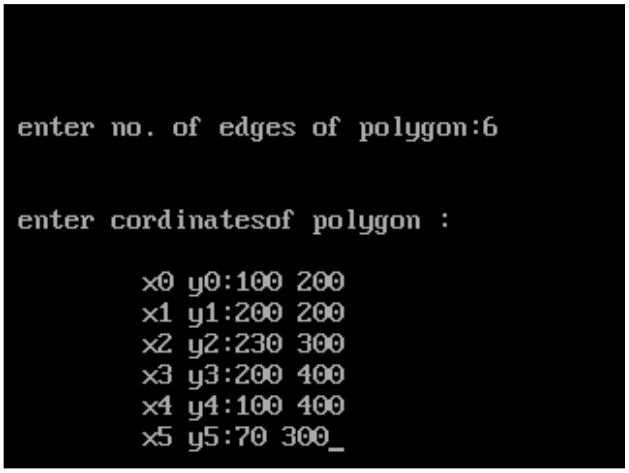
```
// Program to Fill a Polygon Using Scan Line Fill Algorithm in C++.


#include <conio.h>
#include <iostream>
#include <graphics.h>
#include <stdlib.h>
using namespace std;

//Declaration of class point
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 sort();
      void display();
      void calculateX(float);
      void fillPoly(int);
      void drawPoly();
};


void poly::read()
{
   int i;
   cout<<"\n\t SCAN_FILL ALGORITHM";
   cout<<"\n Enter the no of vertices of polygon:";
   cin>>v;
   if(v>2)
   {
      for(i=0;i<v; i++) //ACCEPT THE VERTICES
      {
         cout<<"\nEnter the co-ordinate no.- "<<i+1<<" : ";
         cout<<"\n\tx"<<(i+1)<<"=";
         cin>>p[i].x;
         cout<<"\n\ty"<<(i+1)<<"=";
         cin>>p[i].y;
      }
```

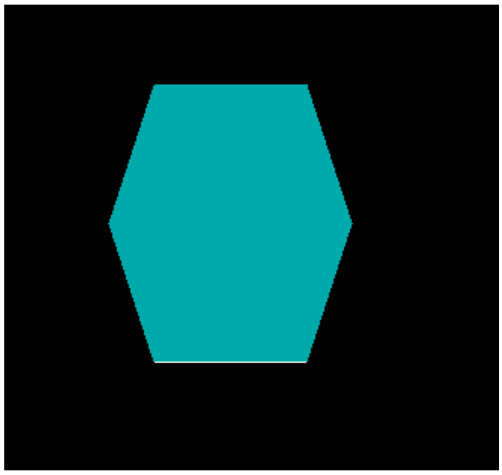
```
        p[i].x=p[0].x;
        p[i].y=p[0].y;
        xmin=xmax=p[0].x;
        ymin=ymax=p[0].y;
    }
    else
        cout<<"\n Enter valid no. of vertices.";
}
//FUNCTION FOR FINDING
void poly::sort()
{ //MAX,MIN
    for(int i=0;i<v;i++)
    {
        if(xmin>p[i].x)
        xmin=p[i].x;
        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;
    }
}
//DISPLAY FUNCTION
void poly::display()
{
    int ch1;
    char ch='y';
    float s,s2;
    do
    {
        cout<<"\n\nMENU:";
        cout<<"\n\n\t1 . Scan line Fill ";
        cout<<"\n\n\t2 . Exit ";
        cout<<"\n\nEnter your choice:";
        cin>>ch1;
        switch(ch1)
        {
            case 1:
                s=ymin+0.01;
                delay(100);
                drawPoly();
                cleardevice();
                while(s<=ymax)
                {
                    calculateX(s);
                    fillPoly(s);
                    s++;
                }
                break;
            case 2:
                exit(0);
```

```
        }

        cout<<"Do you want to continue?: ";
        cin>>ch;
    }while(ch=='y' || ch=='Y');
}

void poly::calculateX(float z) //calculateX FUNCTION INTS
{
    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 // used to make changes in x. so that we can fill our polygon after cerain distance
            {
                x=((x2-x1)*(z-y1))/(y2-y1);
                x=x+x1;
            }
            if(x<=xmax && x>=xmin)
            inter[c++]=x;
        }
    }
}
void poly::drawPoly() //Draw Poly FUNCTION
{
        int i;
        for(i=0;i<v;i++)
        {
            line(p[i].x,p[i].y,p[i+1].x,p[i+1].y); // used to make hollow outlines of a polygon
        }
}

void poly::fillPoly(int z) //fill Poly FUNCTION
{
        int i;
```

```
        for(i=0; i<c;i+=2)
        {
           delay(100);
           line(inter[i],z,inter[i+1],z);  // Used to fill the polygon ....
        }
}

int main() //START OF MAIN
{
   int cl;
   initwindow(500,600);
   cleardevice();
   poly x;
   x.read();
   x.sort();
   cleardevice();
   cout<<"\n\tEnter the colour u want:(0-15)->"; //Selecting colour
   cin>>cl;
   setcolor(cl);
   x.display();
   closegraph(); //CLOSE OF GRAPH
   getch();
   return 0;
}
```

Output:





Conclusion: Thus we have studied Scan Line Polygon Filling Algorithm.