**Assignment 9:**

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMouseEvent>

namespace Ui {

class MainWindow;

}

class CEdge

{

int x1, y1, x2, y2;

float mInv;

public:

static int ymin, ymax;

CEdge()

{

x1=y1=x2=y2=0;

mInv=0;

}

void mInput(int x, int y)

{

x1 = x; y1 = y;

if (y1 > ymax)

ymax = y1;

if (y1 < ymin)

ymin = y1;

}

void mOtherInput(int x, int y)

{

x2 = x;

y2 = y;

}

int getX()

{ return x1;

}

int getY()

{ return y1;

}

int getOtherX()

{ return x2;

}

int getOtherY()

{ return y2;

}

int getYMax()

{ return ymax;

}

int getYMin()

{ return ymin;

}

float mCalculateM()

{

if(y2!=y1)

mInv= float(x2-x1)/float(y2-y1);

else

mInv=0.0;

return mInv;

}

void sortOnY()

{

int temp;

if (y1 < y2)

{

temp = x1;

x1 = x2;

x2 = temp;

temp = y1;

y1 = y2;

y2 = temp;

}

}};

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

CEdge edges[10];

int noOfEdges;

int currX, currY;

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

void callDrawPolygon();

public slots:

void *mousePressEvent*(QMouseEvent \*);

void *paintEvent*(QPaintEvent \*);

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

#include <iostream>

#include <QPainter>

#include <QInputDialog>

#include <QMessageBox>

using namespace std;

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

callDrawPolygon();

ui->setupUi(this);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::*mousePressEvent*(QMouseEvent \*event) {

qDebug() << QString::number(event->pos().x());

qDebug() << QString::number(event->pos().y());

}

/\* Defining the structure to store edges

-----------------------------------------\*/

int CEdge::ymax=0;

int CEdge::ymin=480;

void MainWindow::callDrawPolygon()

{

int n, i, j, k;

//struct edge ed[10],temped;

float m[10], x\_int[10],RefX[10];

int tempX,tempY,yMaximum = 0, yMinimum = 480, active[10], temp=0;

/\* Read the number of vertices of the polygon

------------------------------------------------ \*/

n = QInputDialog::getInteger(this, "Polygon",

"Enter the number vertices of the graph(less than 10): ");

noOfEdges = n;

/\* Read the vertices of the polygon and also find Ymax and Ymin

--------------------------------------------------------------- \*/

for(i = 0; i < n; i++)

{

tempX = QInputDialog::getInteger(this, "Polygon",

QString("Enter X coordinate "));

tempY = QInputDialog::getInteger(this, "Polygon",

QString("Enter Y coordinate "));

edges[i].mInput(tempX, tempY);

if (i>0 && i<= (n-1))

edges[i-1].mOtherInput(tempX, tempY);

if (i == (n-1))

edges[i].mOtherInput(edges[0].getX(), edges[0].getY());

}

currX = QInputDialog::getInteger(this, "Inside Outside Test",

"Enter the point's' X coordinate for the given polygon: " );

currY = QInputDialog::getInteger(this, "Inside Outside Test",

"Enter the point's' Y coordinate for the given polygon: " );

yMaximum= edges[0].getYMax();

yMinimum= edges[0].getYMin();

}

void MainWindow::*paintEvent*(QPaintEvent \*)

{

int n, i=0, j=0, k=0;

CEdge ed[10];

float m[10], x\_int[10],RefX[10];

int tempX,tempY,yMaximum = 0, yMinimum = 480, active[10], temp=0;

n=noOfEdges;

// QMessageBox msgBox;

for(i=0; i<10; i++)

{

ed[i] = edges[i];

}

yMaximum= edges[0].getYMax();

yMinimum= edges[0].getYMin();

QPainter painter(this);

/\* Draw the polygon---------------------- \*/

for(i=0;i<n-1;i++)

{

painter.drawLine(ed[i].getX(),ed[i].getY(),ed[i+1].getX(), ed[i+1].getY());

}

painter.drawLine(ed[n-1].getX(),ed[n-1].getY(),ed[0].getX(), ed[0].getY());

for(i = 0; i < n; i++)

{

m[i] = ed[i].mCalculateM();

}

for(i = 0; i < n; i++)

{

ed[i].sortOnY();

}

//set Reference X array as x1 value for each edge

for(i = 0; i < n; i++)

{

RefX[i]=ed[i].getX();

}

painter.setPen(QColor(0,0,255));

painter.drawPoint(currX, currY);

painter.drawPoint(currX+1, currY);

painter.drawPoint(currX, currY+1);

painter.drawText(currX, currY+80, "Above is the point");

if(currY > yMaximum || currY < yMinimum)

{

painter.drawText(currX-2, currY+4, "Given point is outside the polygon");

return;

}

//if currY lies between yMin and yMax

// Marking active egdes for CurrY

//--------------------------

for (i=0; i<n; i++)

{ active[i]=-1; //Initialize active array before marking

x\_int[i]=0;

}

j=0;

for(i=0;i<n;i++)

{

// if currY> y2 and <= y1

if(currY > ed[i].getOtherY() && currY <= ed[i].getY() && ed[i].getY()!=ed[i].getOtherY())

{

active[j++]=i;

}// if ends

}// for ends

// Finding the x intersections

//--------------------------------

for(k=0; k<j; k++) //for each active edge

{

i =active[k];//get active edge number

if (i == -1)

break;

if(currY==ed[i].getY()) //if current scanline == edge[i].y1

{

x\_int[k]=ed[i].getX(); //then x-int= edge[i].x1

}

else

{

x\_int[k] = RefX[i]-m[i]\*(ed[i].getY()-currY);

}

}//k loop ends here

// Sorting the x intersections

//--------------------------------

for(i=0;i<k;i++)

{

for(j=0;j<k-1;j++)

{

if(x\_int[j]>x\_int[j+1])

{

temp =x\_int[j];

x\_int[j] = x\_int[j+1];

x\_int[j+1] = temp;

}

}

}

// Extracting pairs of x values to draw lines

//----------------------------------------------

for(i=0;i<k;i+=2)

{

if (currX >= x\_int[i] && currX <= x\_int[i+1] )

{

painter.drawText(currX, currY+20, "Given point is inside the polygon");

return;

}

}

painter.drawText(currX, currY+20, "Given point is outside the polygon");

}

\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*



