Assignment 4

Class - SE IV Roll NO - 21430
Batch - F4
D.O.P - 10/9/2020. D.O.S - 18/9/2020

Title -:
Concave polygon filling using scanfill algorithm

Problem statement -: Write c++ program to draw a concave polygon and fill it with desired colour using scan fill algorithm. Apply concept of inheritance

Pre-requisite -:
1. Basic programming skills of c++.
2. 64 bit open source Linux.
3. open source c++ programming tool like G++ / GCC

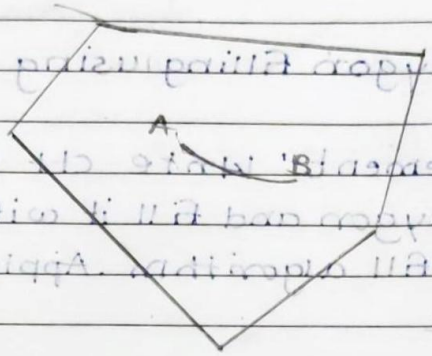
Learning objective -:
To understand and implement scanline polygon fill algorithm.

Theory -:
A Polygon -:
A polygon is a closed path composed of a finite number of sequential line segments. A polygon is a two-dimensional shaped formed with more than three straight lines. When startingpoint and terminal point is same then it is called polygon.

* Types of Polygon

1) Convex Polygon :- A convex polygon is a simple polygon whose interior is convex set. In a convex polygon, all interior angles are less than 180 degrees.

Eg.

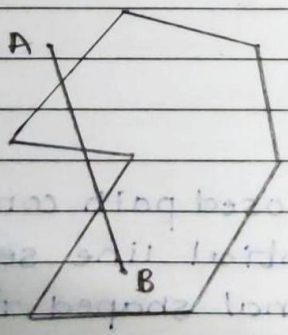


2) Concave polygon :-

A concave polygon has always interior angle greater than 180 degrees. It is possible to cut concave polygon into a set of convex polygons.

You can draw at least one straight line through a concave polygon that crosses more than two sides

Eg.

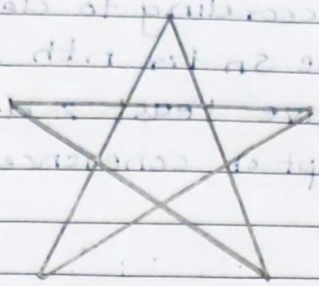


3) Complex polygon :-

Complex polygon is a polygon whose sides cross

each other one or more times.

E.g.:



* Algorithm-:

Step 1 : Start

Step 2 : Initialize the desired data structure.

1. Create polygon table having colour, edge pointers, coefficients.
2. Establish edge table contains information regarding the endpoint of edges, pointer to polygon, inverseslope
3. Creative active edge list, This will be sorted in increasing order of x.
4. Creating a flag f. It will have two values either on or off.

Step 3: Perform the following steps for all scan lines.

1. Enter values in Active edge list in sorted order using y as value
2. Scan until the flag i.e. f is on using a background colour.
3. When one polygon flag is on, and this is for surface S1 enter colour intensity as I1 into refresh buffer.

4. When two or image surface flag are on, sort the surfaces according to depth and use intensity value s_n for nth surface. This surface will have least z depth value
5. Use the concept of coherence for remaining planes.

step4 : stop algorithm.

* Algorithm for DDA line

Step 1 : Get co-ordinate of both end points (x_1, y_1) and (x_2, y_2) from user

Step 2 : Calculate difference between two end points

$dx = x_2 - x_1$ & $dy = y_2 - y_1$.

Step 3 : Based on calculated difference in step 2 you need to identify number of steps to put pixel. If $dx > dy$ then you need more steps in x co-ordinate, otherwise in y co-ordinate

```
if ( (absolute(dx)) > absolute(dy) )
   steps = absolute (dx)
else
   steps = absolute(dy)
```

Step 5 : plot pixels by successfully incrementing x and y co-ordinate accordingly and complete the drawing of line.

```
for ( int i=0 ; i<steps; i++)
{
    x1 = x1 + xincrement ;
    y1 = y1 + y increment ;
    putpixel ( x1 , y1 , colour name);
}
```

* Psuedocode

i) Mousepress Event.

```
start
define Mouse_press_event():
    if ( left mouse click)
        take x co-ordinate and store in p
        take y co-ordinate and stor in q.
        Store all vertices in Array
            a[ver] = p
            b[ver] = q
            increment ver.
    else
        a[ver] = a[0]
        b[ver] = b[0]
        draw polygon
END.
```

ii) For DDA

i. start

def DDA(int x_1, int y_1, int x_2, int y_2)

1. Read x_1, y_1, x_2, y_2
2. set $dx = x_2 - x_1$

   set $dy = y_2 - y_1$
3. if ($dx > dy$)

   3.1. set step = abs(dx)

   else

   3.2. set step = abs(dy)
4. set x_{inc} = dx / steps

   set y_{inc} = dy / steps
5. Initialize $x = x_1$

   Initialize $y = y_1$
6. set i = 0
7. for (i = 0; i < steps; i++)

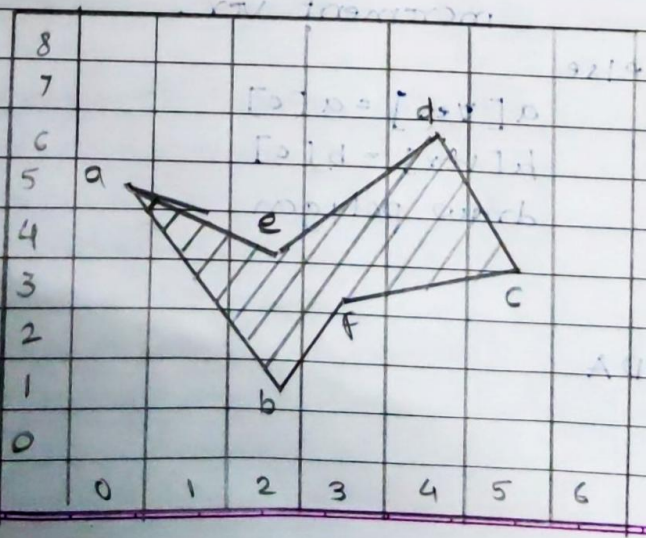
   setpixel (x, y, colour);

   $x = x + x_{inc}$

   $y = y + y_{inc}$

END.

* Dry Run.



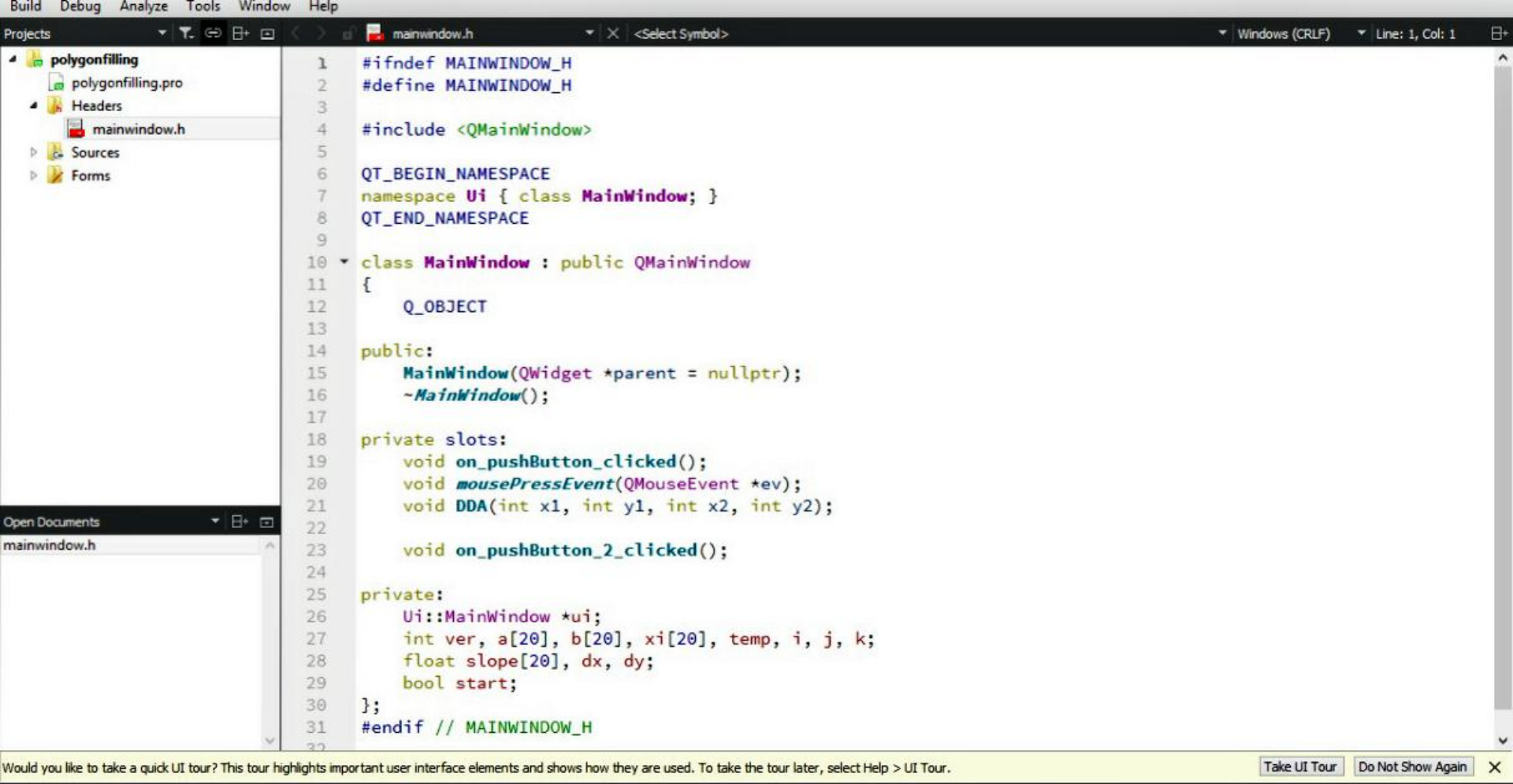
*

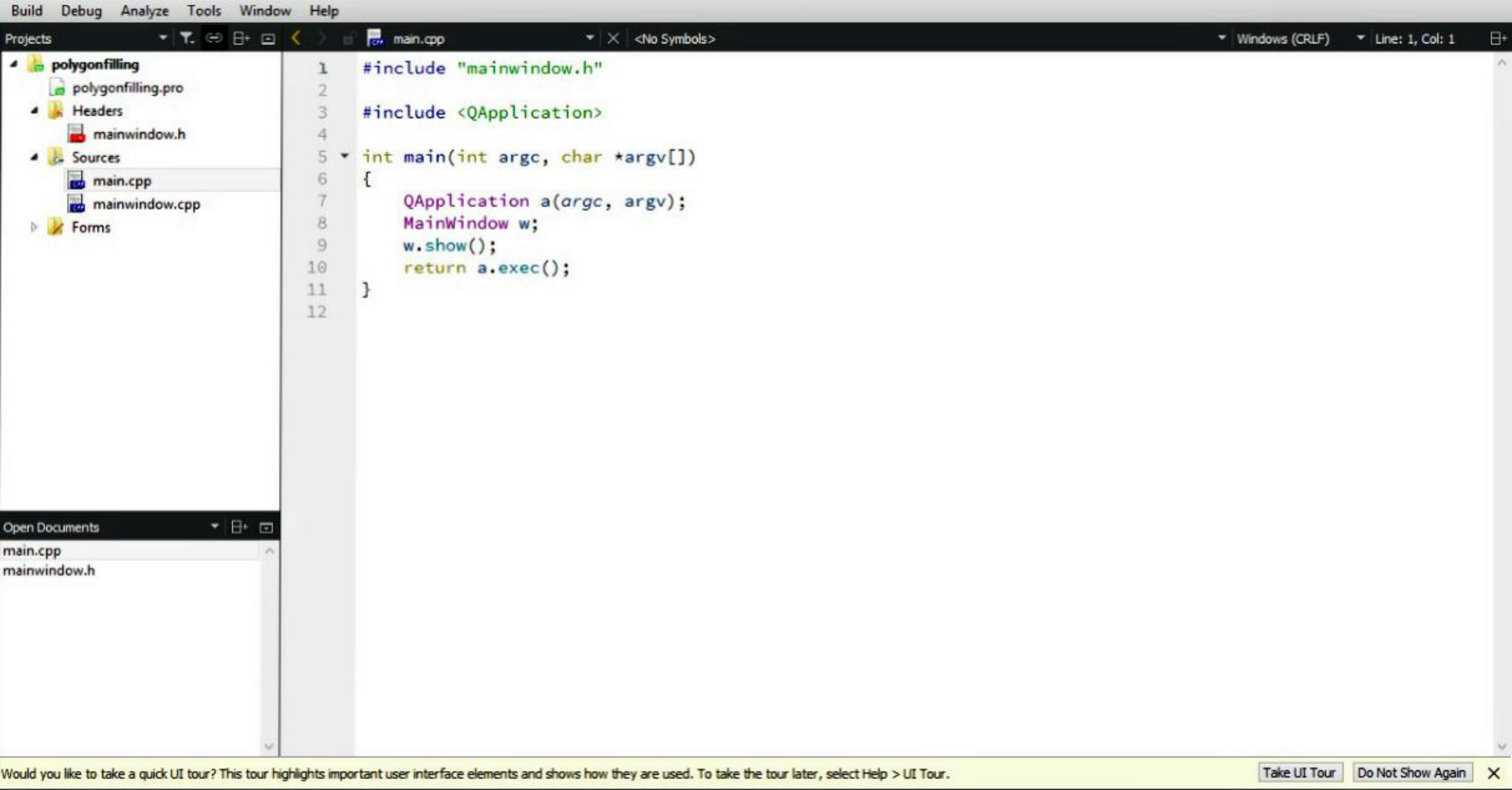
NO. of points	x_1	y_1	x_2	y_1	step
1. (a,b)	(0),5	(5)	2	1	3
2. (b,f)	2	1	3	3	2
3. (f,c)	3	3	5	4	2
4. (c,d)	5	4	4	6	2
5. (d,e)	4	6	2	4	2
6 (e,a)	2	4	0	5	2

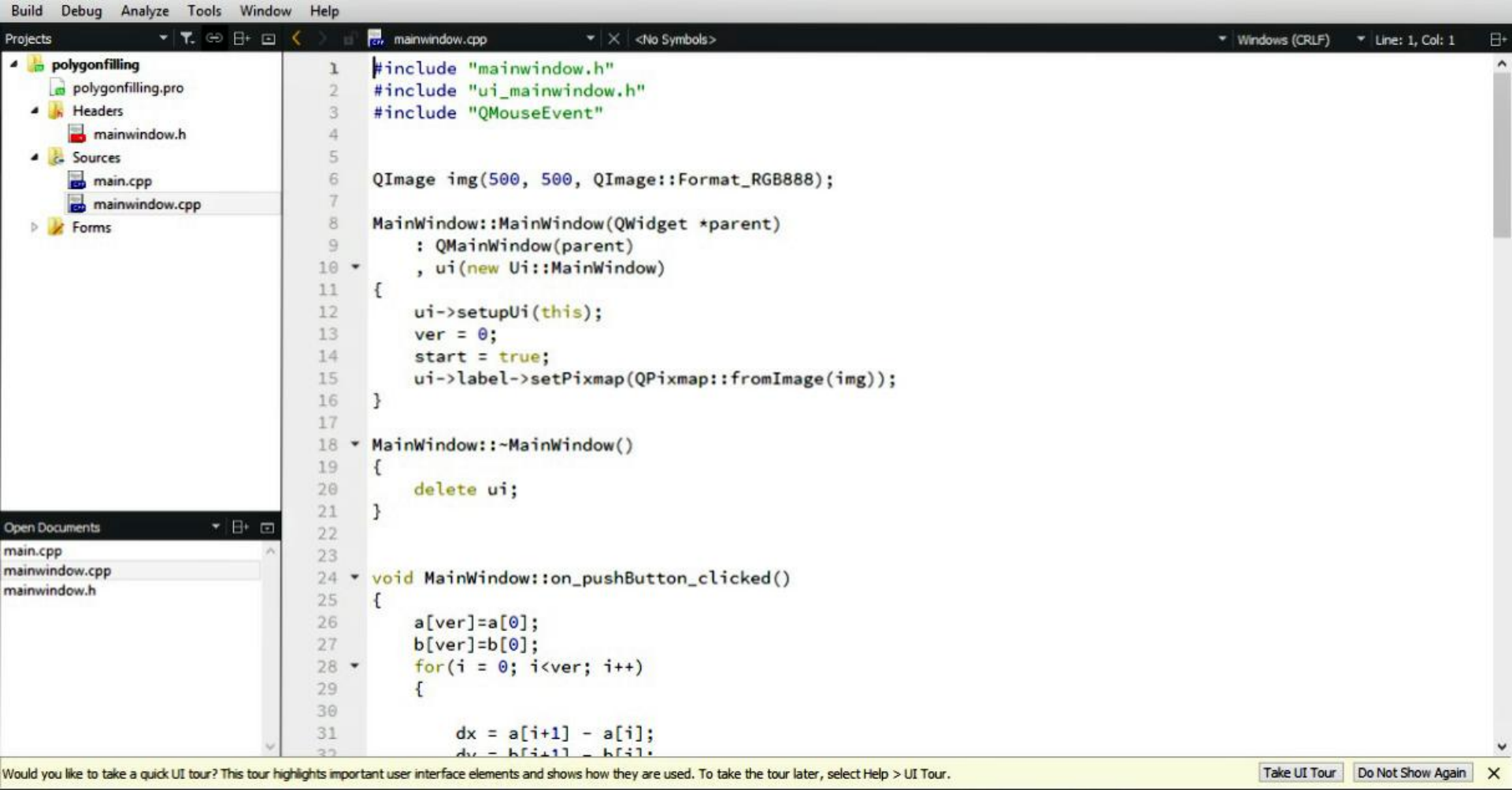
Conclusion :-

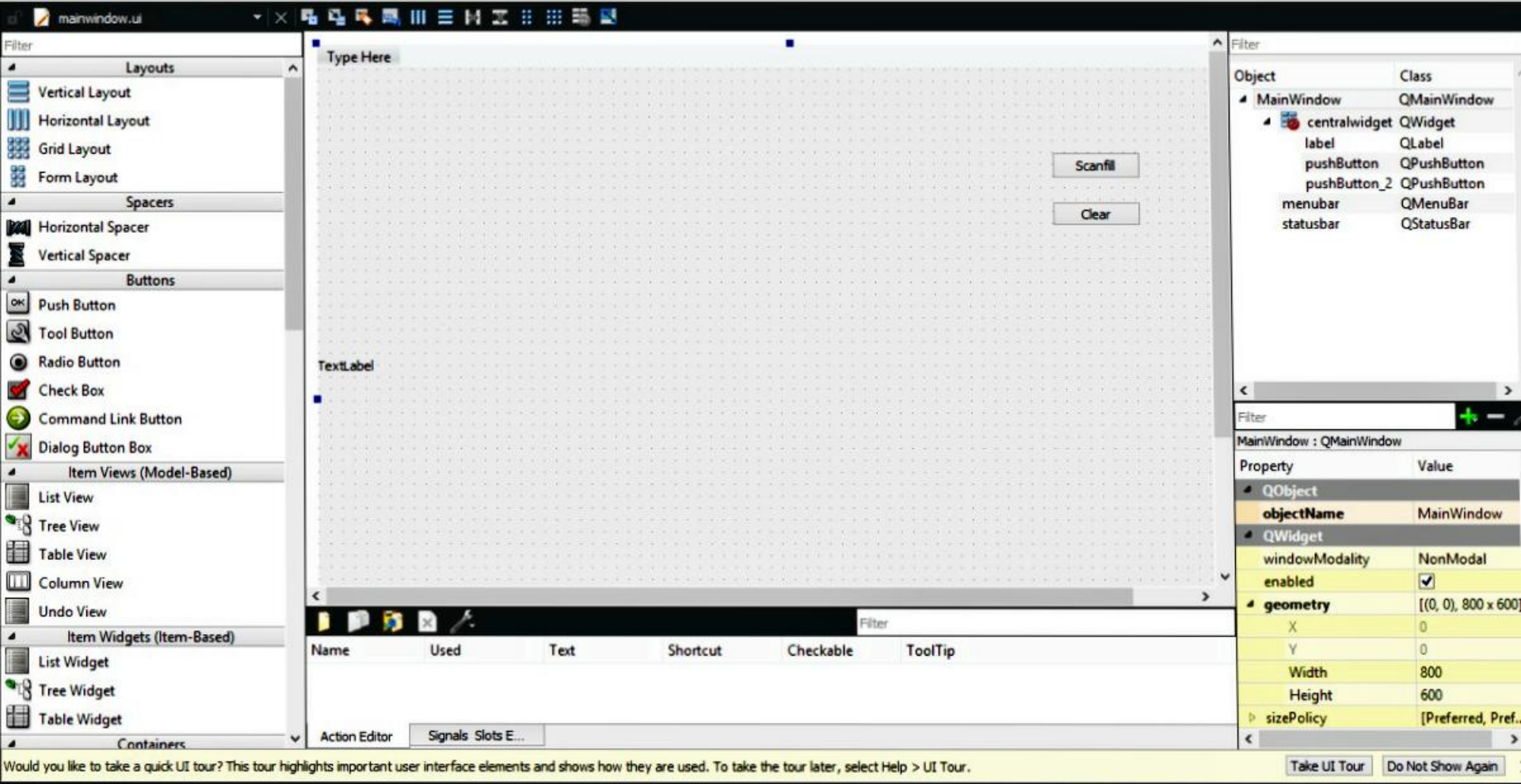
i. The concept of scanline fill algorithm using DDA Breshenam line algorithm was learned.

2 Using concept of mouse instance and scanfill algorithm, the require concave or convex polygon was drawn and filled succesfully on UI.


Build Debug Analyze Tools Window Help
Projects
polygonfilling
polygonfilling.pro
Headers
mainwindow.h
Sources
Forms
Open Documents
mainwindow.h
mainwindow.h
<Select Symbol>
Windows (CRLF)
Line: 1, Col: 1
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
class MainWindow : public QMainWindow
{
Q_OBJECT
public:
MainWindow(QWidget *parent = nullptr);
~MainWindow();
private slots:
void on_pushButton_clicked();
void mousePressEvent(QMouseEvent *ev);
void DDA(int x1, int y1, int x2, int y2);
void on_pushButton_2_clicked();
private:
Ui::MainWindow *ui;
int ver, a[20], b[20], xi[20], temp, i, j, k;
float slope[20], dx, dy;
bool start;
};
#endif // MAINWINDOW_H
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again


Build Debug Analyze Tools Window Help
Projects
polygonfilling
polygonfilling.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
Open Documents
main.cpp
mainwindow.h
main.cpp
<No Symbols>
Windows (CRLF)
Line: 1, Col: 1
1 #include "mainwindow.h"
2
3 #include <QApplication>
4
5 int main(int argc, char *argv[])
6 {
7 QApplication a(argc, argv);
8 MainWindow w;
9 w.show();
10 return a.exec();
11 }
12
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again


Build Debug Analyze Tools Window Help
Projects
polygonfilling
polygonfilling.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
Open Documents
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.cpp
<No Symbols>
Windows (CRLF)
Line: 1, Col: 1
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "QMouseEvent"
QImage img(500, 500, QImage::Format_RGB888);
MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow)
{
ui->setupUi(this);
ver = 0;
start = true;
ui->label->setPixmap(QPixmap::fromImage(img));
}
MainWindow::~MainWindow()
{
delete ui;
}
void MainWindow::on_pushButton_clicked()
{
a[ver]=a[0];
b[ver]=b[0];
for(i = 0; i<ver; i++)
{
dx = a[i+1] - a[i];
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again


mainwindow.ui
Filter
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Undo View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Type Here
Scanfill
Clear
TextLabel
Filter
Name
Used
Text
Shortcut
Checkable
ToolTip
Action Editor
Signals Slots E...
Filter
Object
Class
MainWindow
QMainWindow
centralwidget
QWidget
label
QLabel
pushButton
QPushButton
pushButton_2
QPushButton
menubar
QMenuBar
statusbar
QStatusBar
Filter
MainWindow : QMainWindow
Property
Value
QObject
objectName
MainWindow
QWidget
windowModality
NonModal
enabled
geometry
[(0, 0), 800 x 600]
X
0
Y
0
Width
800
Height
600
sizePolicy
[Preferred, Pref...
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again

Build Debug Analyze Tools Window Help

Projects

- polygonfilling
  - polygonfilling.pro
  - Headers
    - mainwindow.h
  - Sources
    - main.cpp
    - mainwindow.cpp
  - Forms
    - mainwindow.ui

Open Documents

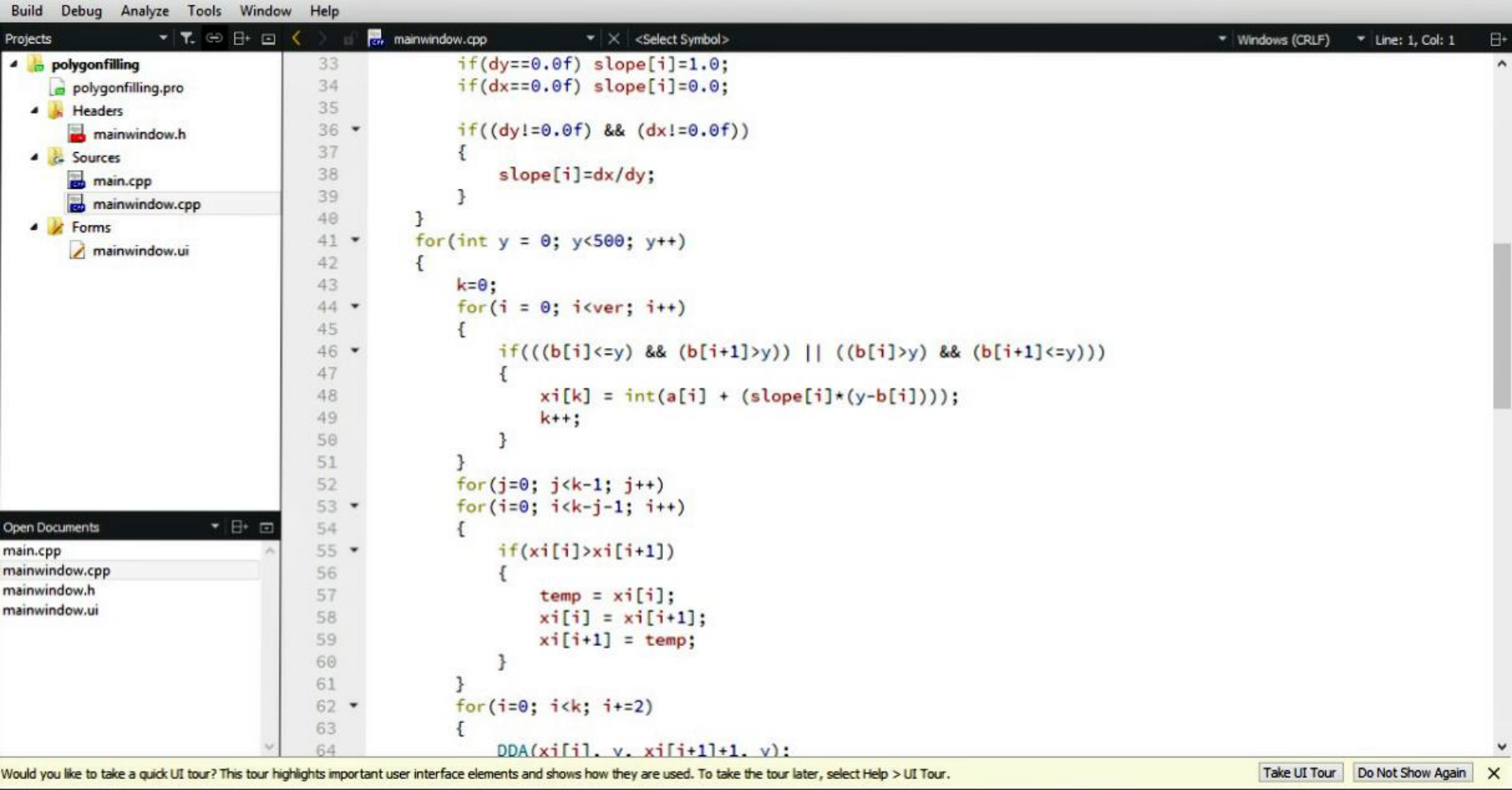
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui

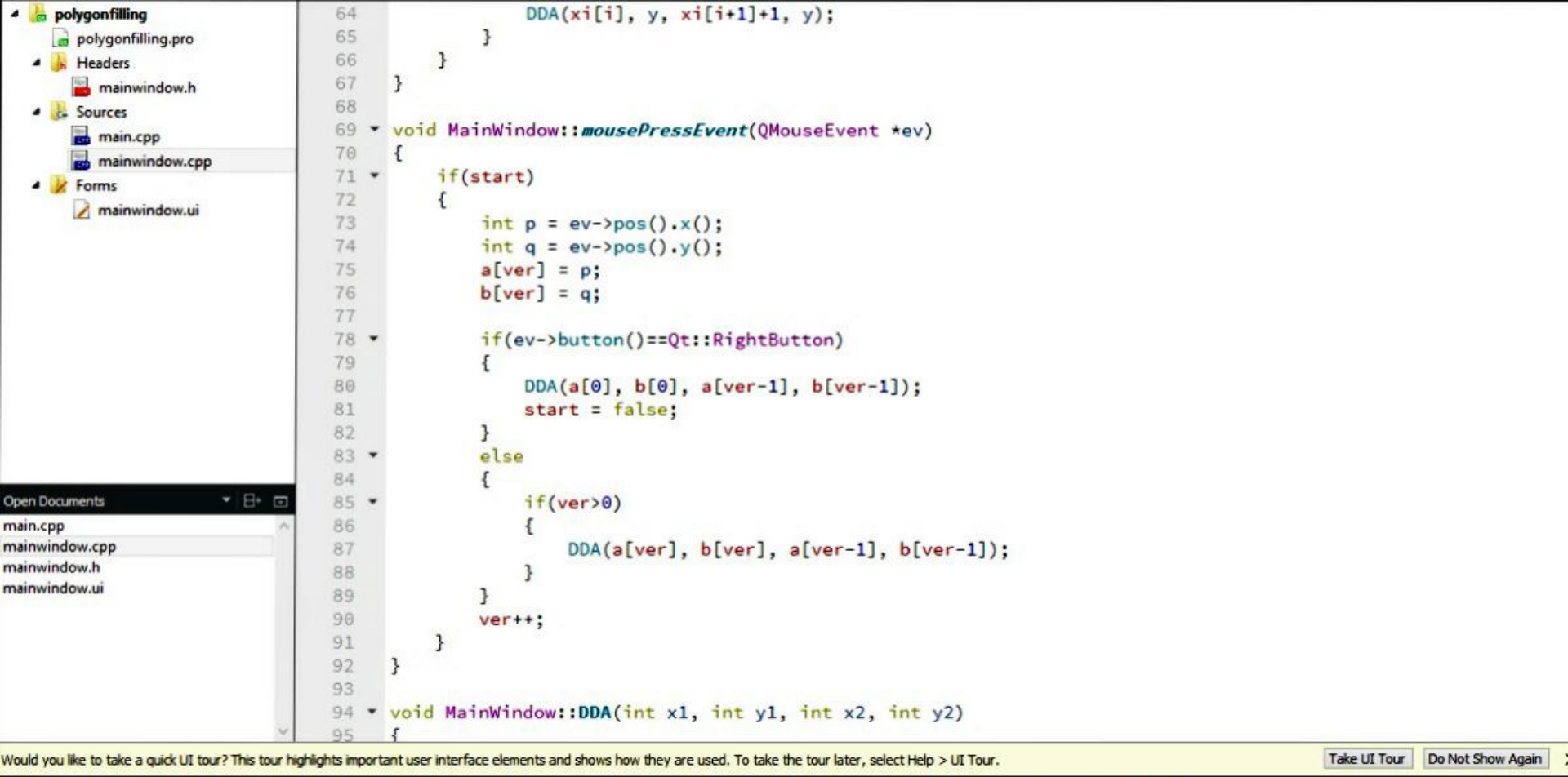
mainwindow.cpp | <Select Symbol> | Windows (CRLF) | Line: 1, Col: 1

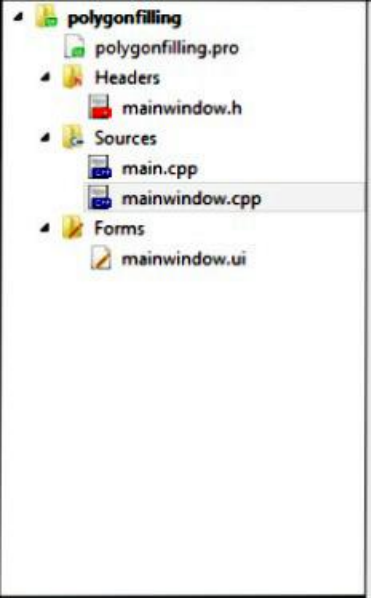
```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "QMouseEvent"


QImage img(500, 500, QImage::Format_RGB888);

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ver = 0;
    start = true;
    ui->label->setPixmap(QPixmap::fromImage(img));
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_pushButton_clicked()
{
    a[ver]=a[0];
    b[ver]=b[0];
    for(i = 0; i<ver; i++)
    {

        dx = a[i+1] - a[i];
```

Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.

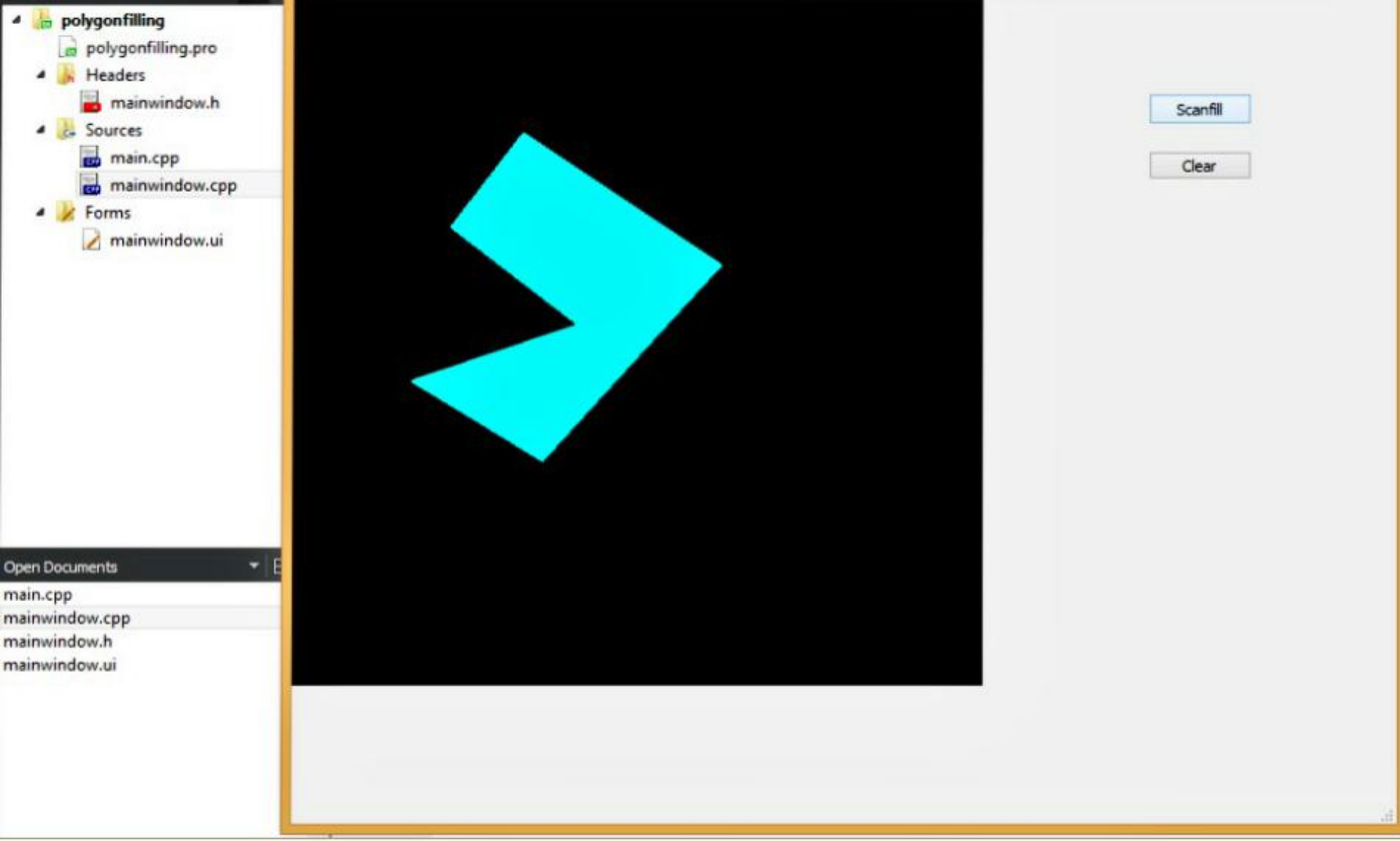
Take UI Tour | Do Not Show Again

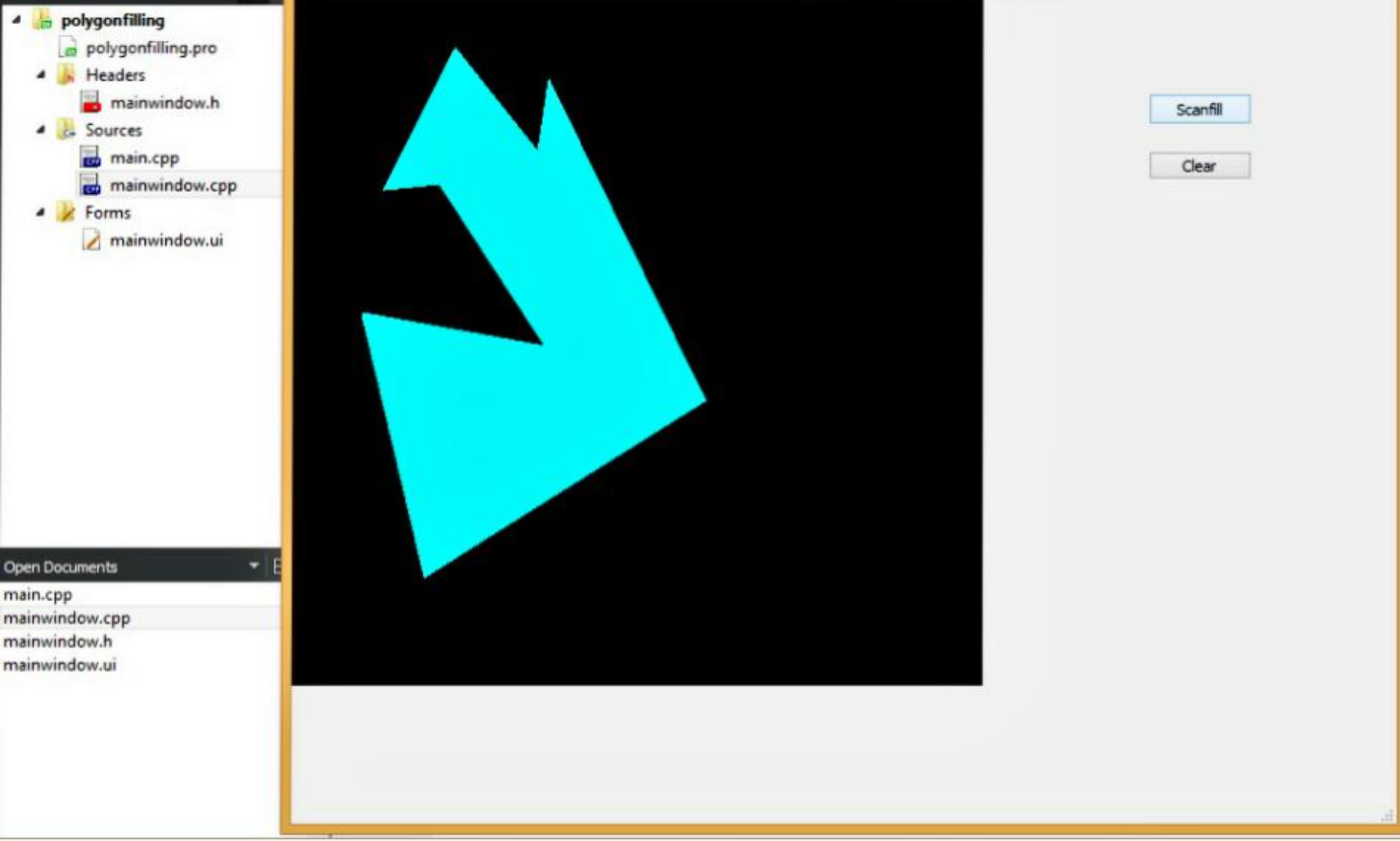

Build Debug Analyze Tools Window Help
Projects
polygonfilling
polygonfilling.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Open Documents
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
mainwindow.cpp
<Select Symbol>
Windows (CRLF)
Line: 1, Col: 1
if(dy==0.0f) slope[i]=1.0;
if(dx==0.0f) slope[i]=0.0;
if((dy!=0.0f) && (dx!=0.0f))
{
slope[i]=dx/dy;
}
}
for(int y = 0; y<500; y++)
{
k=0;
for(i = 0; i<ver; i++)
{
if(((b[i]<=y) && (b[i+1]>y)) || ((b[i]>y) && (b[i+1]<=y)))
{
xi[k] = int(a[i] + (slope[i]*(y-b[i])));
k++;
}
}
for(j=0; j<k-1; j++)
for(i=0; i<k-j-1; i++)
{
if(xi[i]>xi[i+1])
{
temp = xi[i];
xi[i] = xi[i+1];
xi[i+1] = temp;
}
}
for(i=0; i<k; i+=2)
{
DDA(xi[i], y, xi[i+1]+1, y);
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again


polygonfilling
polygonfilling.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Open Documents
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
DDA(xi[i], y, xi[i+1]+1, y);
}
}
}
void MainWindow::mousePressEvent(QMouseEvent *ev)
{
if(start)
{
int p = ev->pos().x();
int q = ev->pos().y();
a[ver] = p;
b[ver] = q;
if(ev->button()==Qt::RightButton)
{
DDA(a[0], b[0], a[ver-1], b[ver-1]);
start = false;
}
else
{
if(ver>0)
{
DDA(a[ver], b[ver], a[ver-1], b[ver-1]);
}
}
ver++;
}
}
void MainWindow::DDA(int x1, int y1, int x2, int y2)
{
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again




```
94  void MainWindow::DDA(int x1, int y1, int x2, int y2)
95  {
96      float dx=x2-x1;
97      float dy= y2-y1;
98
99      float steps = abs(dx) >= abs(dy) ?  abs(dx) : abs(dy);
100
101     float xinc= dx/steps;
102     float yinc=dy/steps;
103
104     float x=x1;
105     float y=y1;
106
107     for(int i=0; i<steps; i++)
108     {
109         img.setPixel(x,y,qRgb(0,255,255));
110         x+=xinc;
111         y+=yinc;
112     }
113     ui->label->setPixmap(QPixmap::fromImage(img));
114 }
115
116
117 void MainWindow::on_pushButton_2_clicked()
118 {
119     start = true;
120     ver = 0;
121     img.fill(0);
122     ui->label->setPixmap(QPixmap::fromImage(img));
123 }
124
```


polygonfilling
polygonfilling.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Open Documents
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
Scanfill
Clear
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.


polygonfilling
polygonfilling.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Open Documents
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
Scanfill
Clear
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.