GIS prototype system based on QGIS

Wang Yiyi

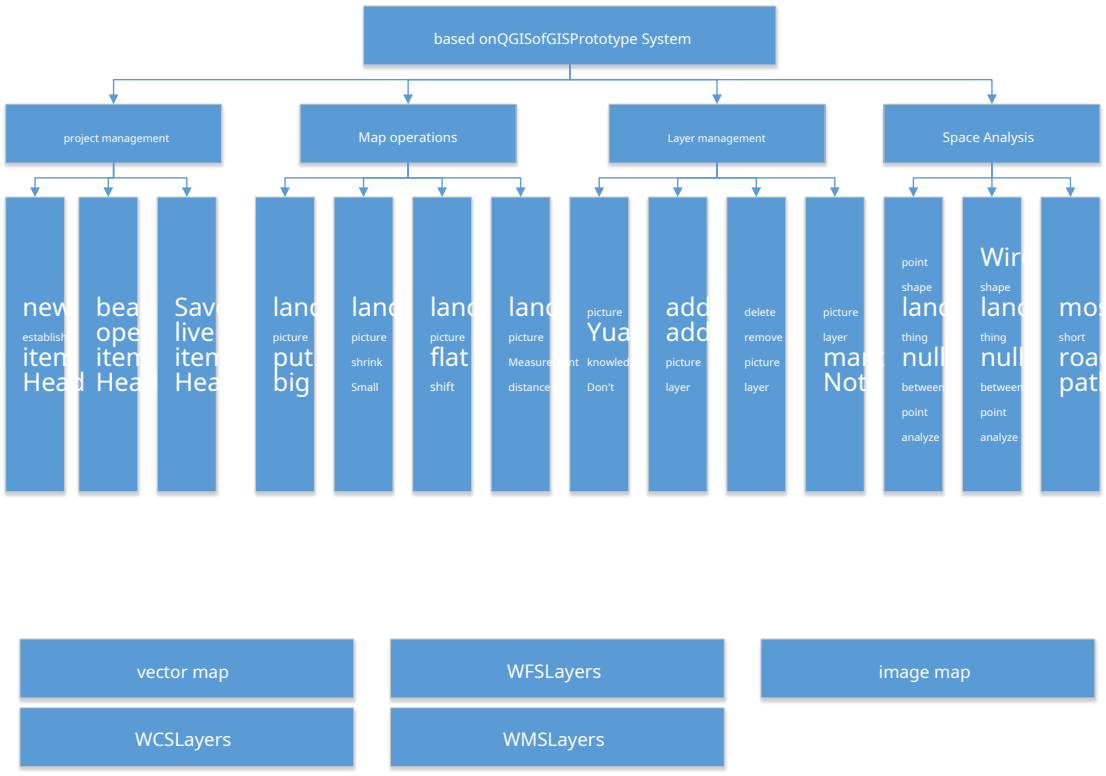
1.Experimental requirements

1.1basic requirements

The main goal is to leverage open source software librariesQGISImplement basicGISsystem. currentQGISin two ways For users to use: For end users, it is available for users to use in the form of applications; for programmers, Provide users with secondary development in the form of dynamic libraries and other methods. This design is mainly carried out in the second way, and Not availableQGISwhich providedQGISModifications to the application's source code must be based onQGISLibrary new construction process to achieve.

1.2Functional requirements

Mainly completed**based onQGISofGISPrototype System**, the system functions mainly include project management, map Operations, layer management and spatial analysis modules. The specific functions of the system are shown in the figure below.



For detailed functional descriptions of project management, map operation, and layer management function modules, seeQGISHelp text files. For the layer management module, vector maps, image maps,WCSlayers,WFSlayers,

WMSlayer this5Management of map layers.

2. Instructions for QGIS secondary development steps (for QGIS compilation, please see the compilation documentation)

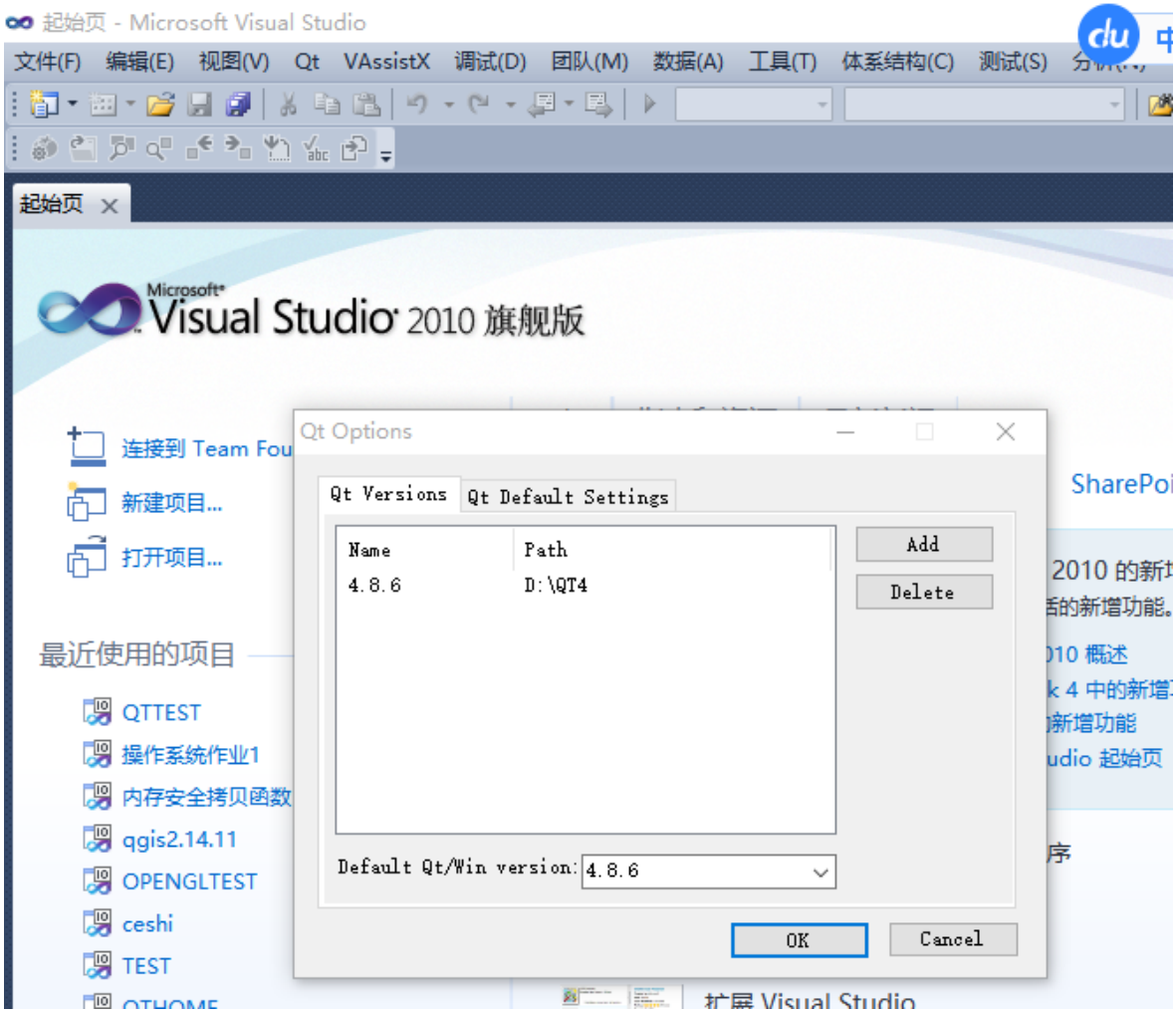
2.1 New construction

Version Notes: QT4.8.6, VS2010

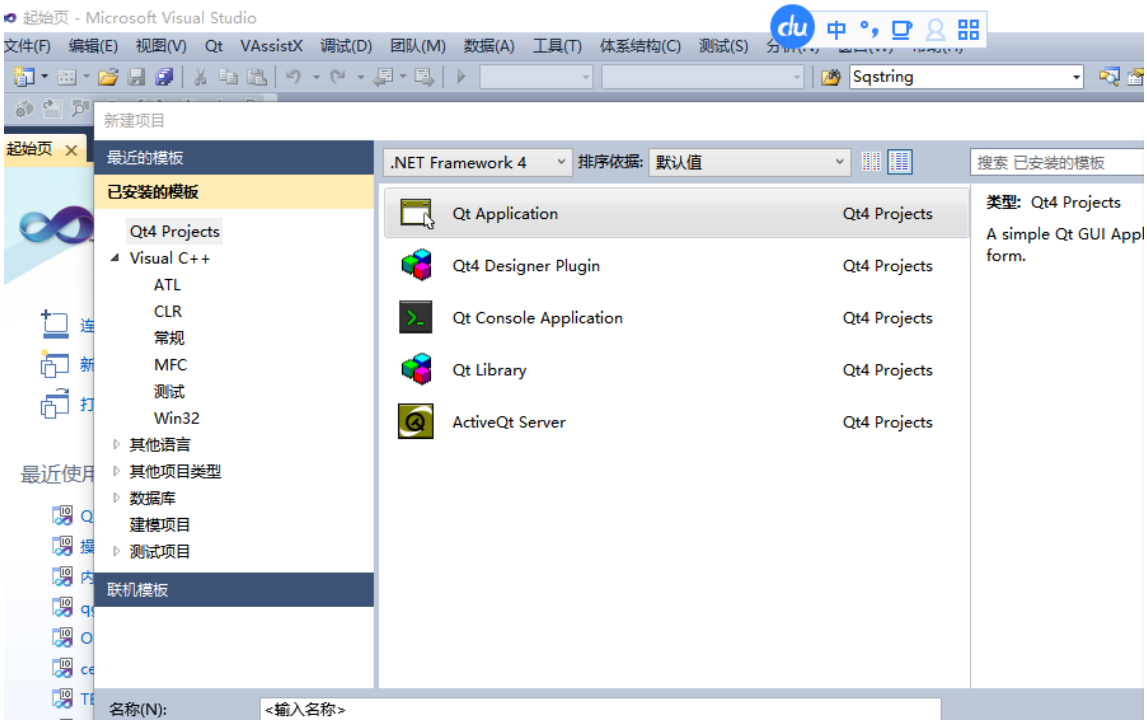
After completing the qgis2.14.11 source code compilation,You can start writing GIS projects.

First, configuring QT4 in VS2010 requires a plug-in qt-vs2010-addin.

So I downloaded one from Baidu and added QT4 to VS2010.



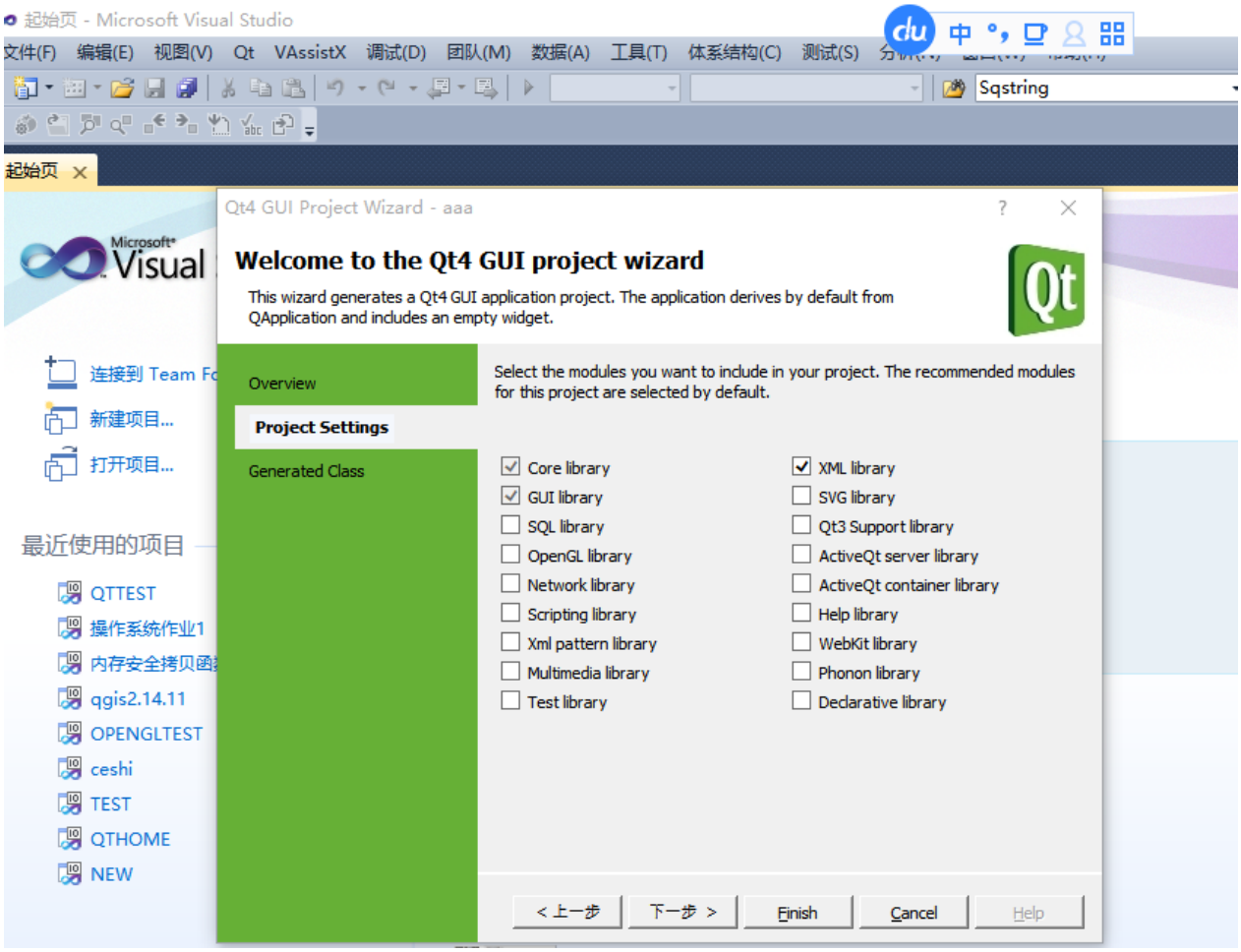
Click New Project, select QT4 projects, and create a QT Application



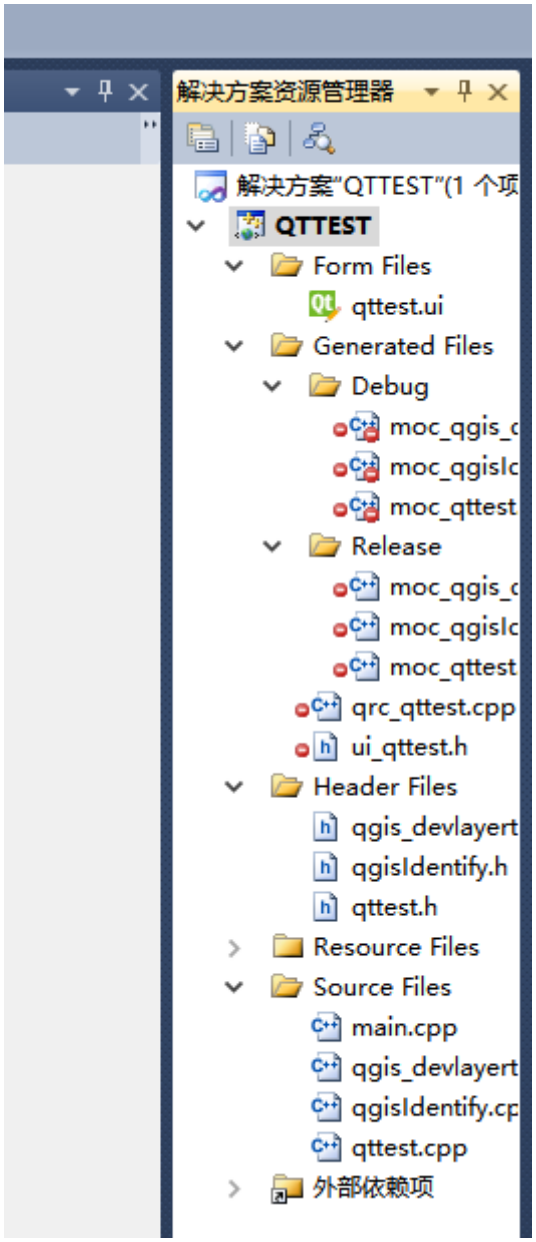
In the new option, be sure to check the XML library, otherwise an error will be reported. Because QgsMapCanvas

You need to use the QDomNode file, and QDomNode is an XML file, included in Qt's XML library

middle.



After creating a new project, you will get a QT project


解决方案资源管理器
解决方案"QTTEST"(1 个项
QTTEST
Form Files
qttest.ui
Generated Files
Debug
moc_qgis_
moc_qgisIc
moc_qttest
Release
moc_qgis_
moc_qgisIc
moc_qttest
qrc_qttest.cpp
ui_qttest.h
Header Files
qgis_devlayert
qgisIdentify.h
qttest.h
Resource Files
Source Files
main.cpp
qgis_devlayert
qgisIdentify.cp
qttest.cpp
外部依赖项

2.2 Project configuration

Since you need to use the compiledqgisCarry out secondary development and therefore carry out engineering configuration. (To selectRelease

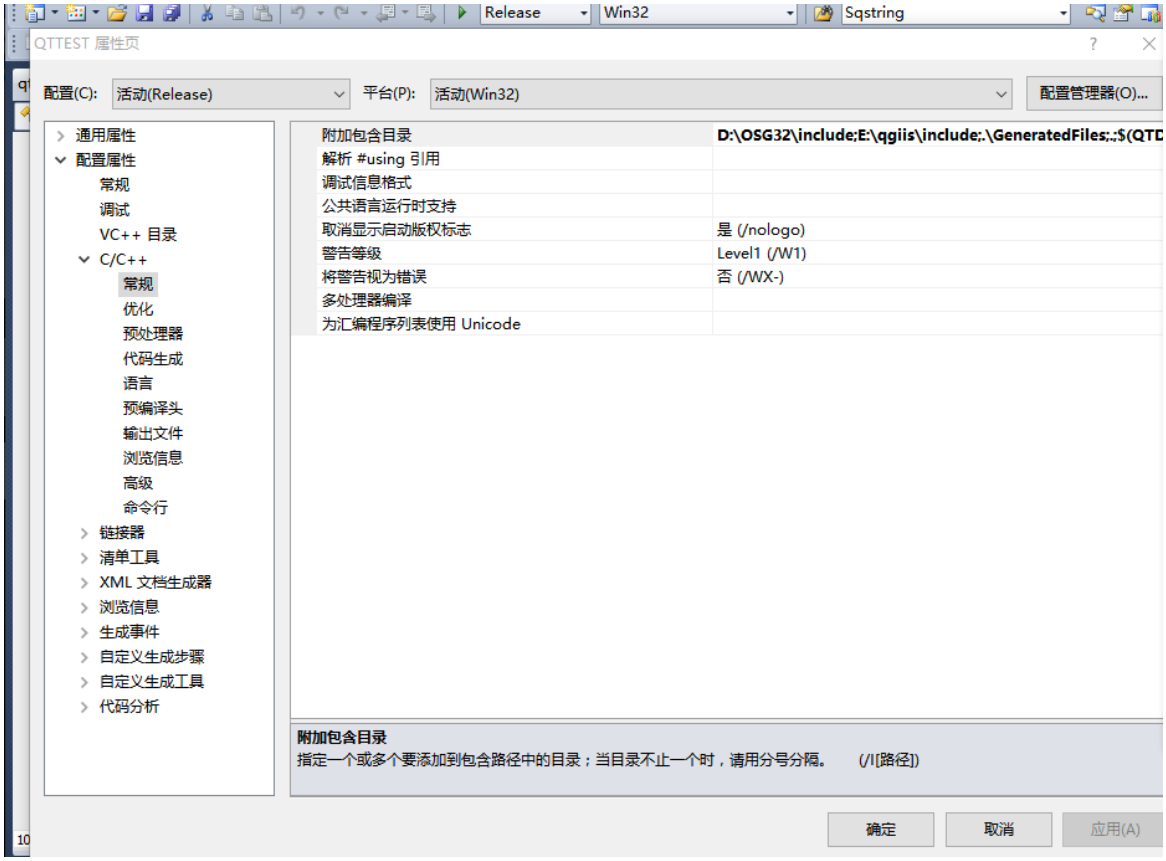
Mode) Right-click on the project project to open the property configuration interface. [Which tab is the following configuration specific?

Please look directly at the picture. 】

2.2.1Configure additional include directories

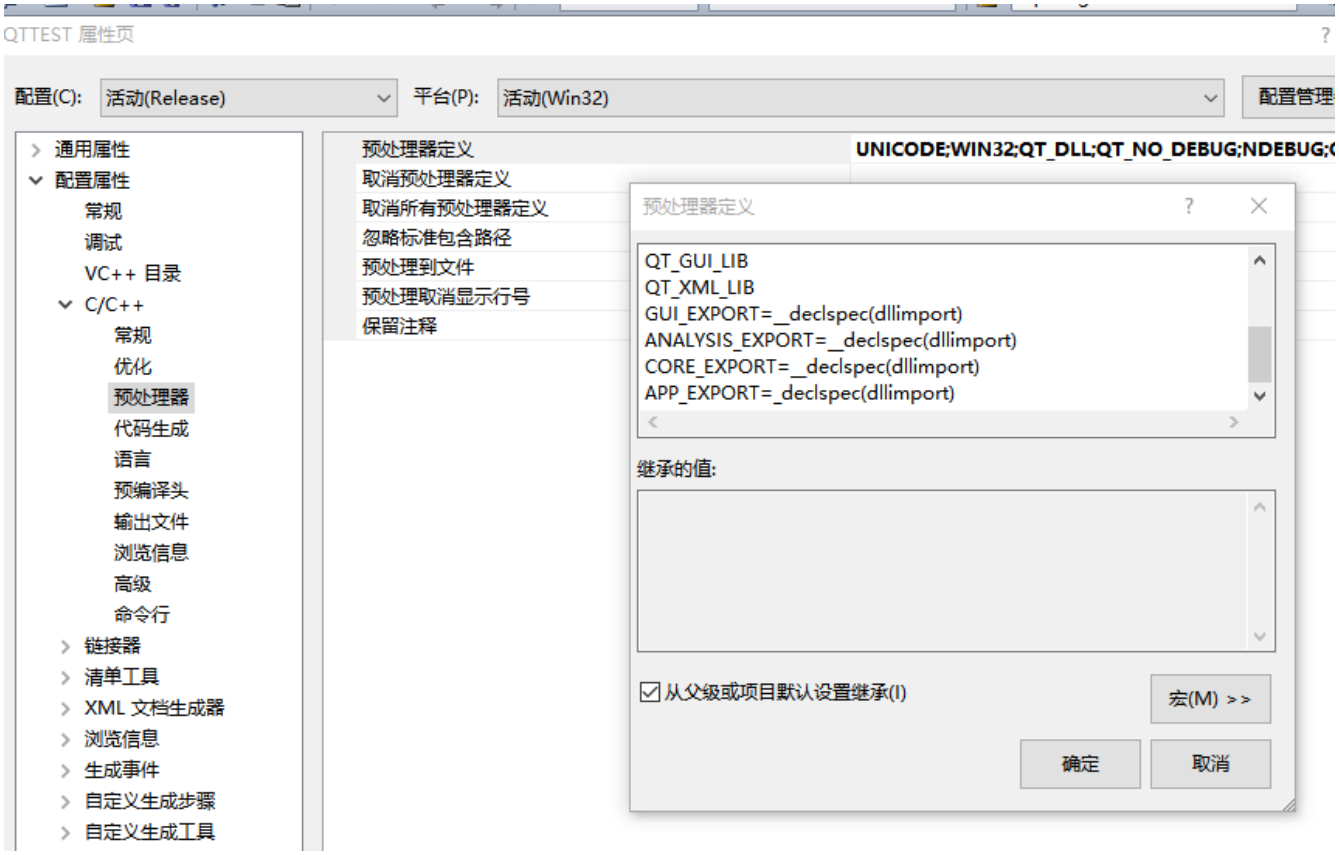
Need to add hereOSGeo4Wofinclude, and by compilingqgisofINSTALLNew file generation

ofqgisDocumentaryinclude.



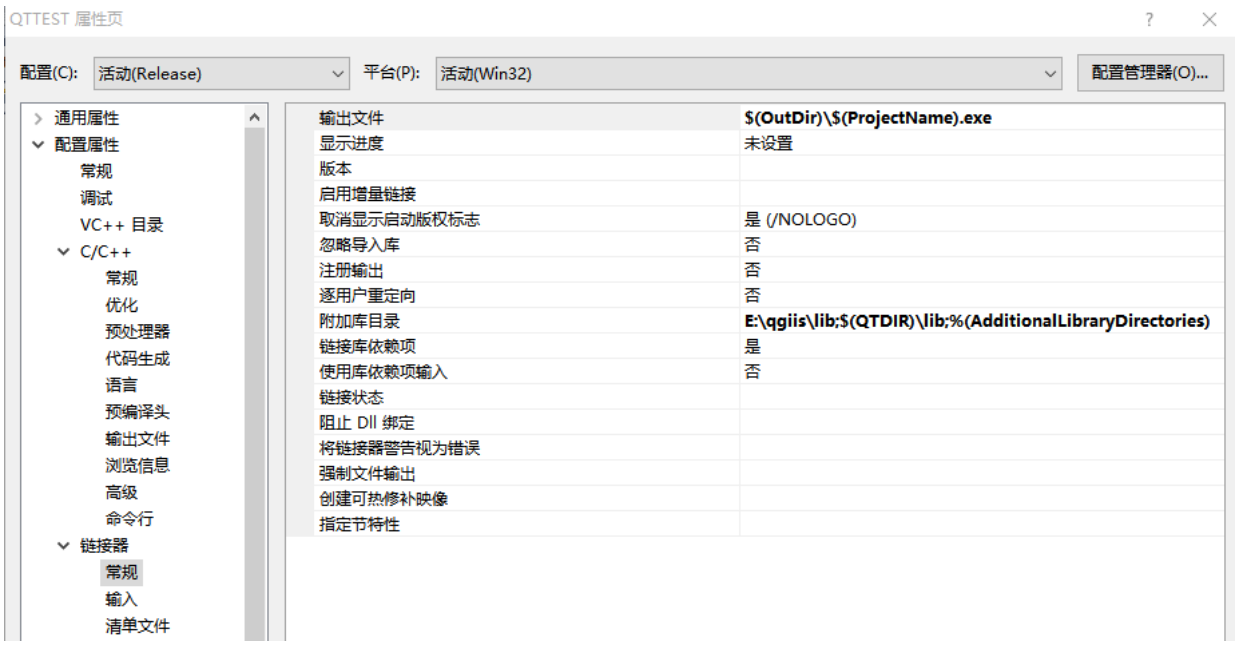
2.2.2Add preprocessing commands

Add the last four lines of preprocessing becausereleaseThe version has no preprocessing definition, and an error will be reported without preprocessing.



2.2.3Add additional library directory

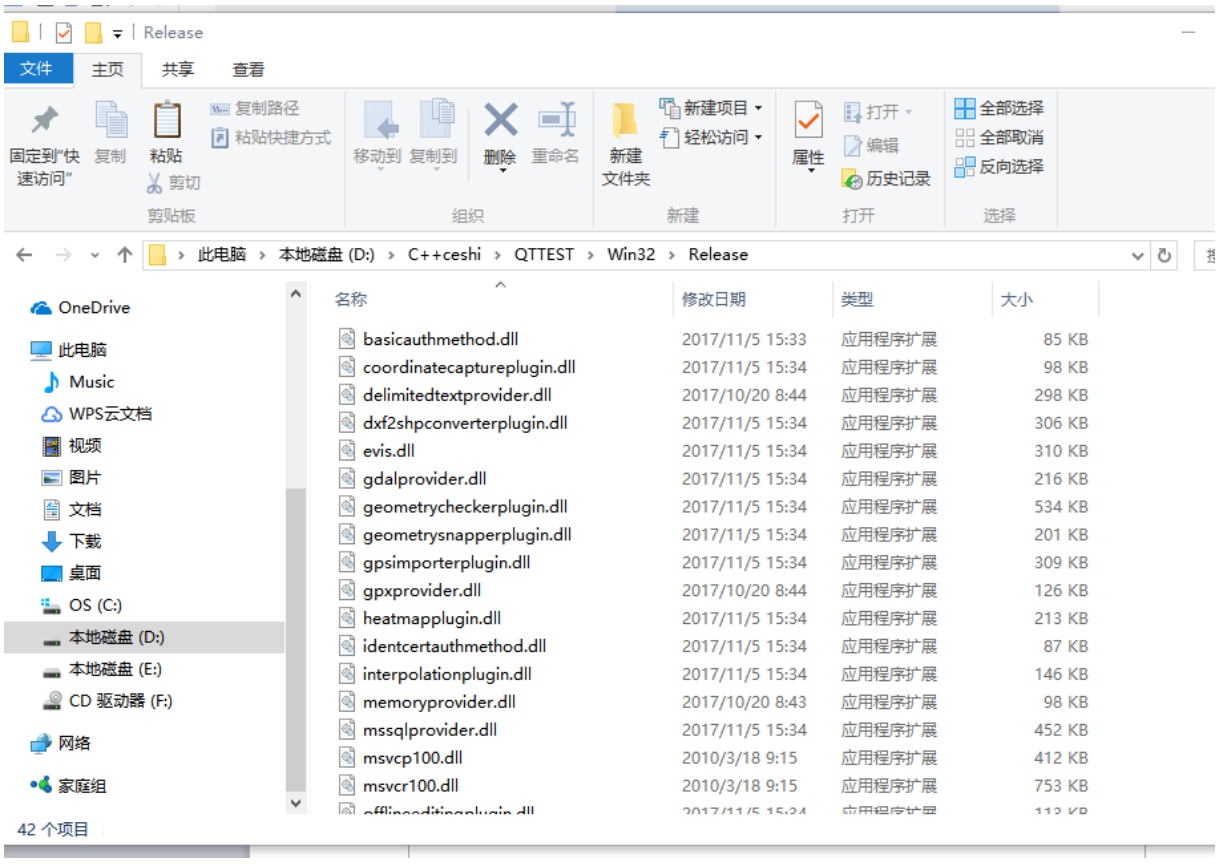
The path isINSTALLGeneratedqgisoflib.

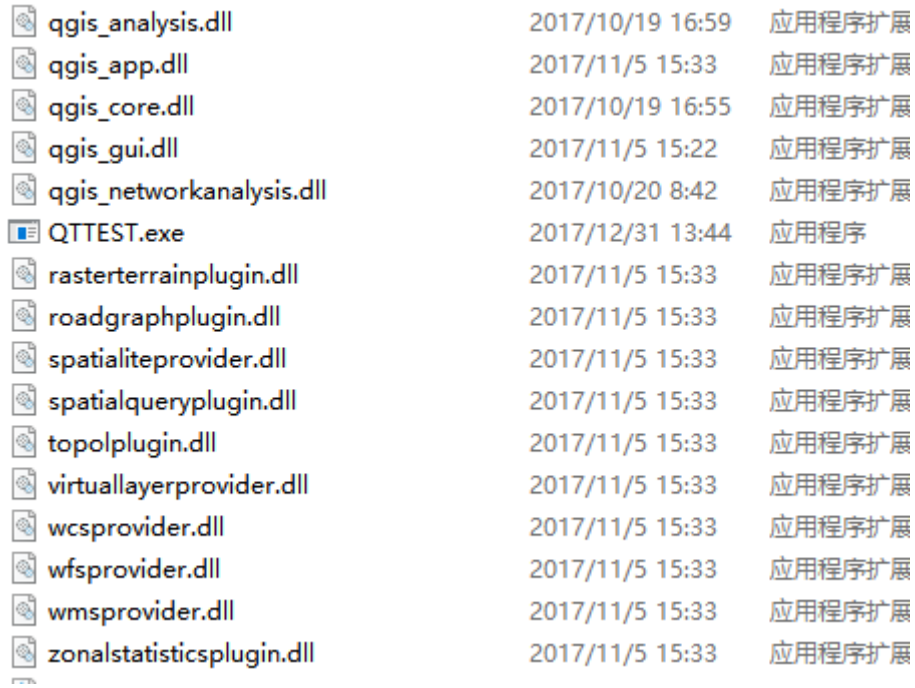


2.2.4Add additional dependencies

Qgis_analysis.lib,Qgis_core.lib,Qgis_gui.lib.

2.2.5Willqgisrelateddllmake a copy and put it in the newly created project directory to run it correctly.





After configuring the above content, you can write code for testing. My first test after configuring the project

The method is to see whether the written layer loading code can load the layer correctly.

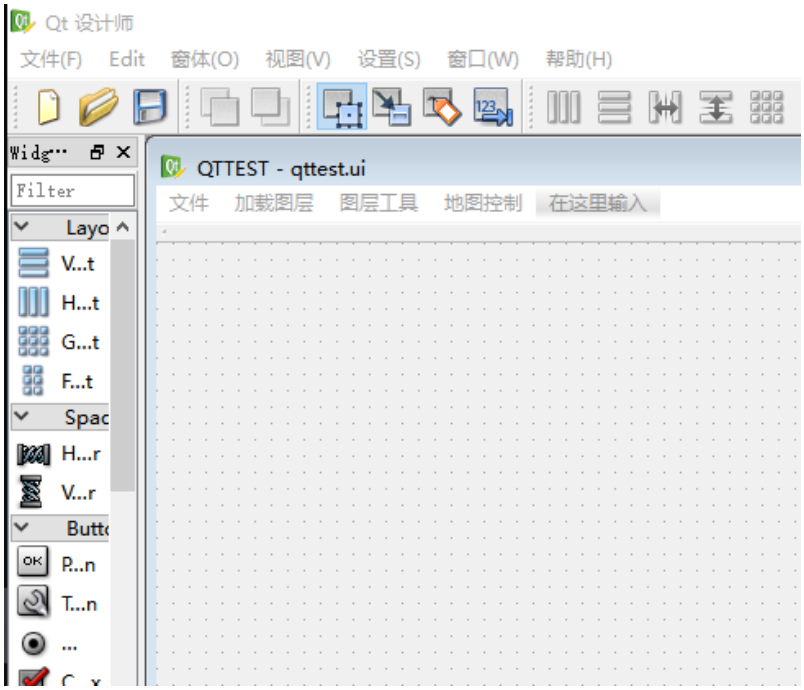
3.based onQGISofGISDevelopment instructions for the prototype system

Off topic, how to find the implementation class corresponding to the function? I did it by usingVSDebug executionqgis.exeThen click

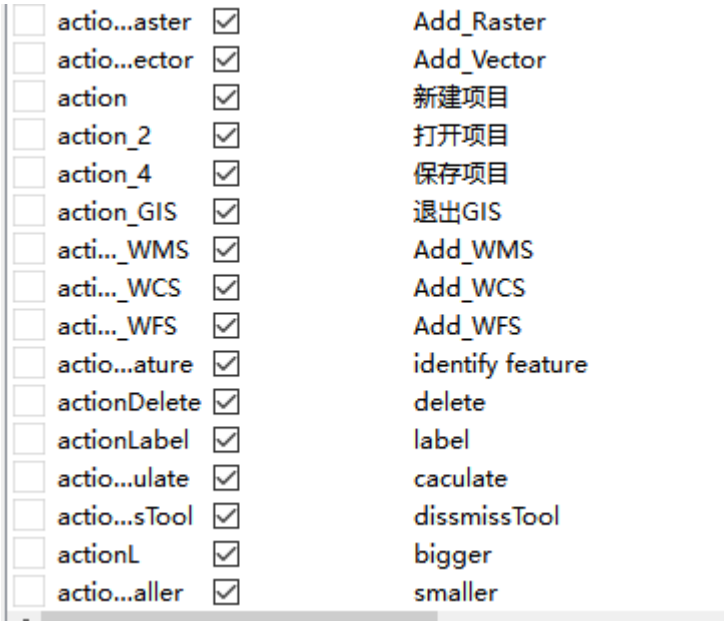
Click the corresponding function button, and then look at the debugging window to display which classes it calls, and you can read the corresponding

QGISofAPIDocument, find the function that implements the function, and finally implement the code.

First, interface design can be done throughQT designeraccomplish:

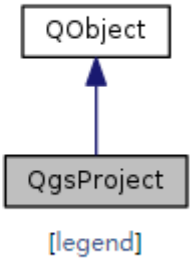


Each time a submenu button is added, a corresponding action object is automatically generated to respond to the click action.

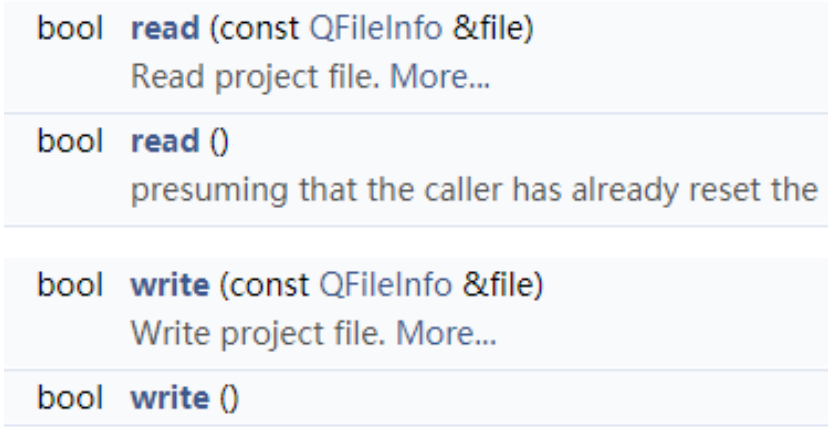


3.1project management

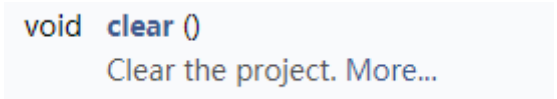
In Implementation of Project Management I readqgisofficialAPImiddleQgsProjectkind.



This class contains read() and write() functions, which can automatically read and write .qgs project files.



The clear() function can clear the qgs project, which also implements the function of creating a new project.



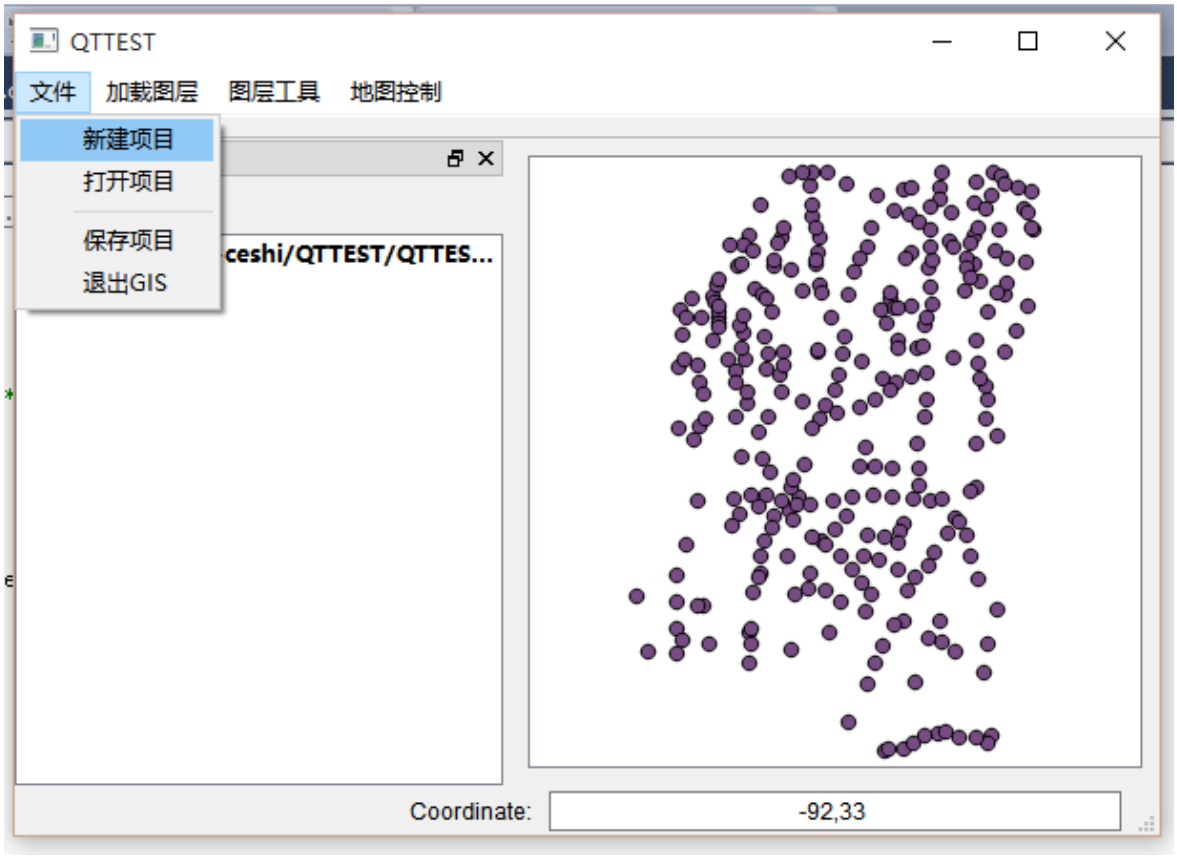
Therefore, the above three functions can be used to create, open, and save qgs projects.

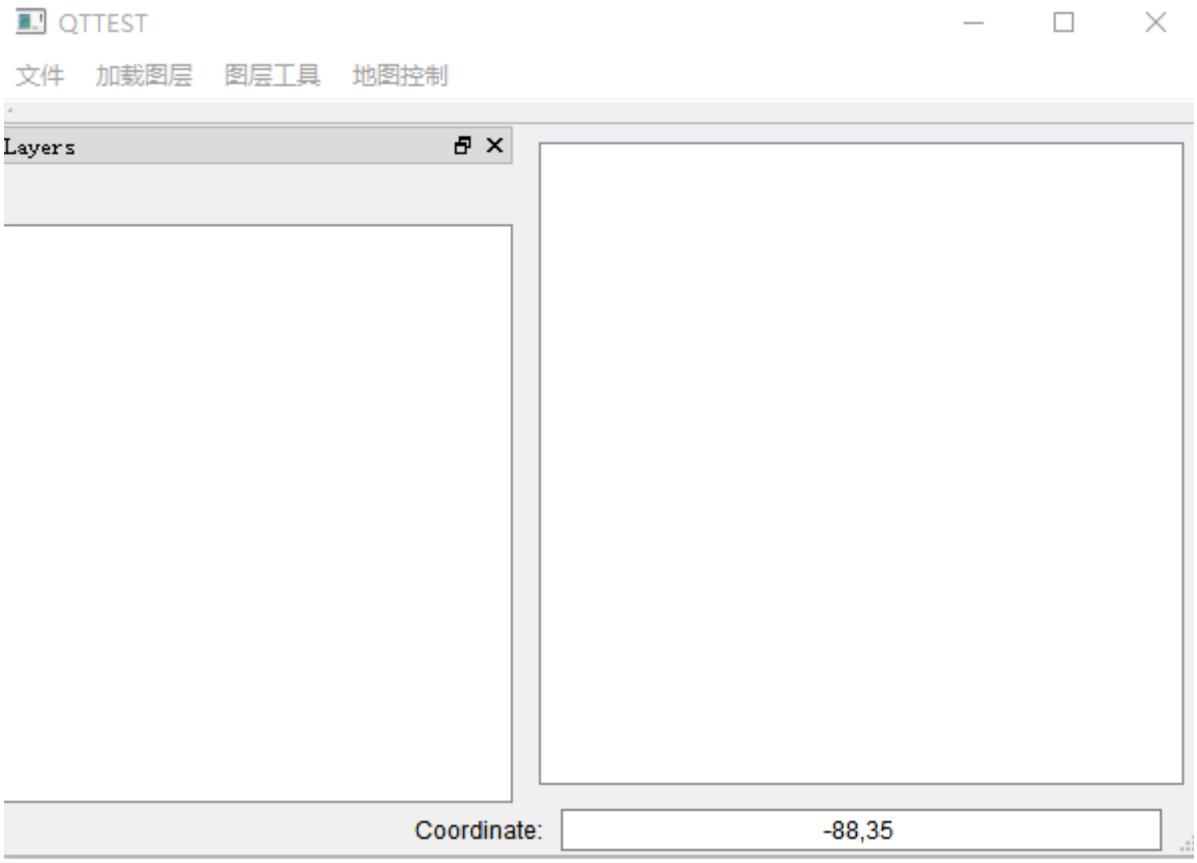
3.1.1New Project

Implementation code:

transferQgsprojectofclear()The method can clear the project and create a new blank project.

```cpp
//*****************************************************************
//项目管理
//新建项目
void QTTEST::newfile()
{

    QgsProject::instance()->clear();//清空项目

}
```





3.1.2Open project

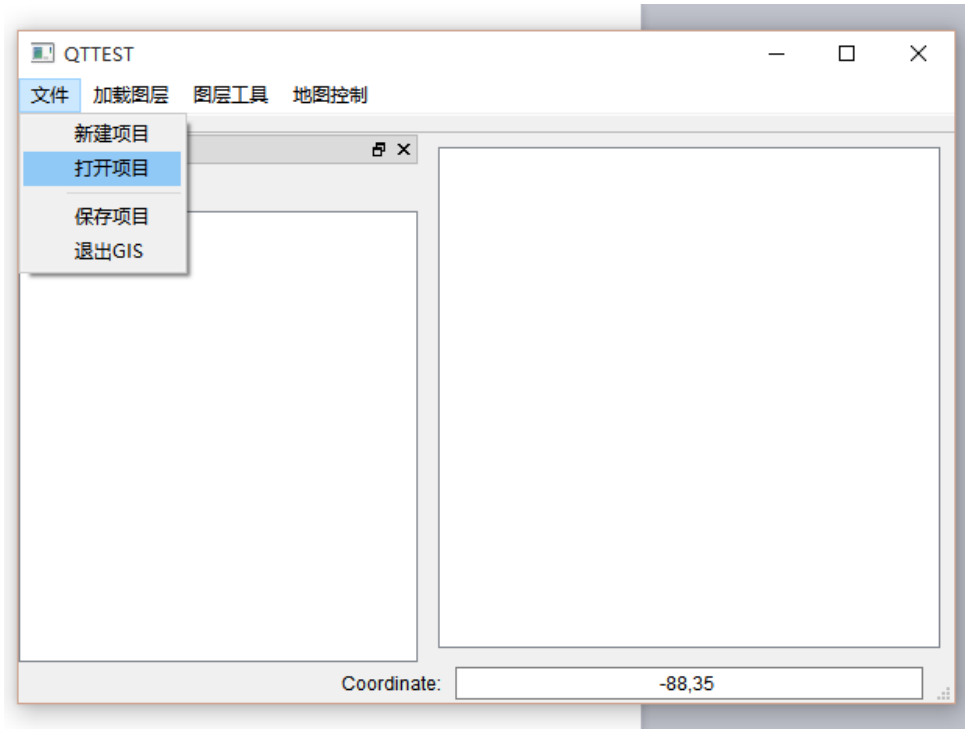
Implementation code:

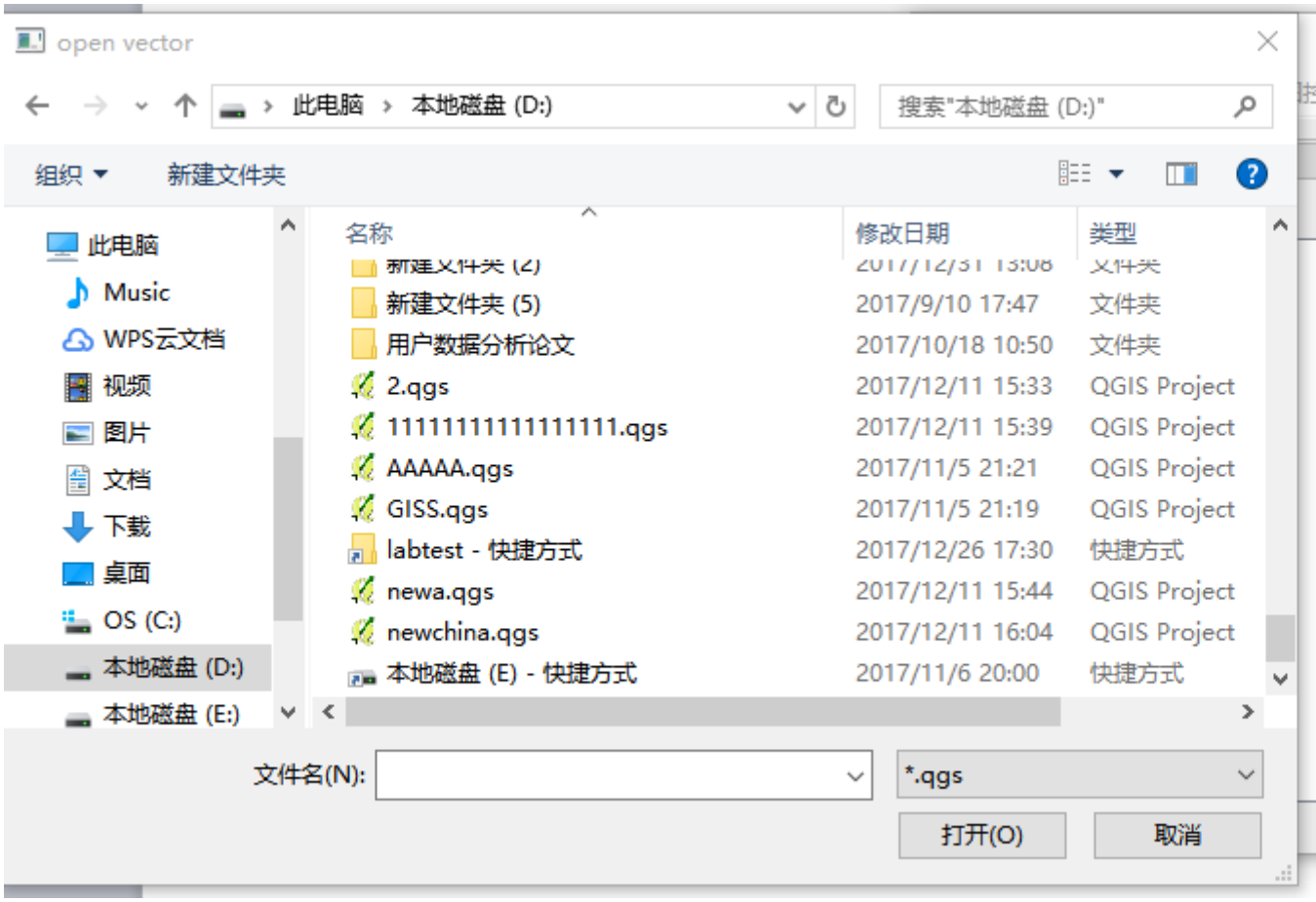
Create a new canvas object and load the layers of the read project file into the new canvas.

filenameUsed to save the path, callQgsProjectofread()method readqgsproject.

```
//打开项目
void QTTEST::openfile()
{


    QgsMapCanvas * canvas = new QgsMapCanvas();
    QString filename = QFileDialog::getOpenFileName( this, tr( "open vector" ), "", "*.qgs" );
    QgsProject::instance()->read(QFileInfo(filename));
    QgsLayerTreeMapCanvasBridge * bridge = new QgsLayerTreeMapCanvasBridge(QgsProject::instance()->layerTreeRoot(), canvas);


}
```

Realization effect:





Here I openednewchina.qgsproject.

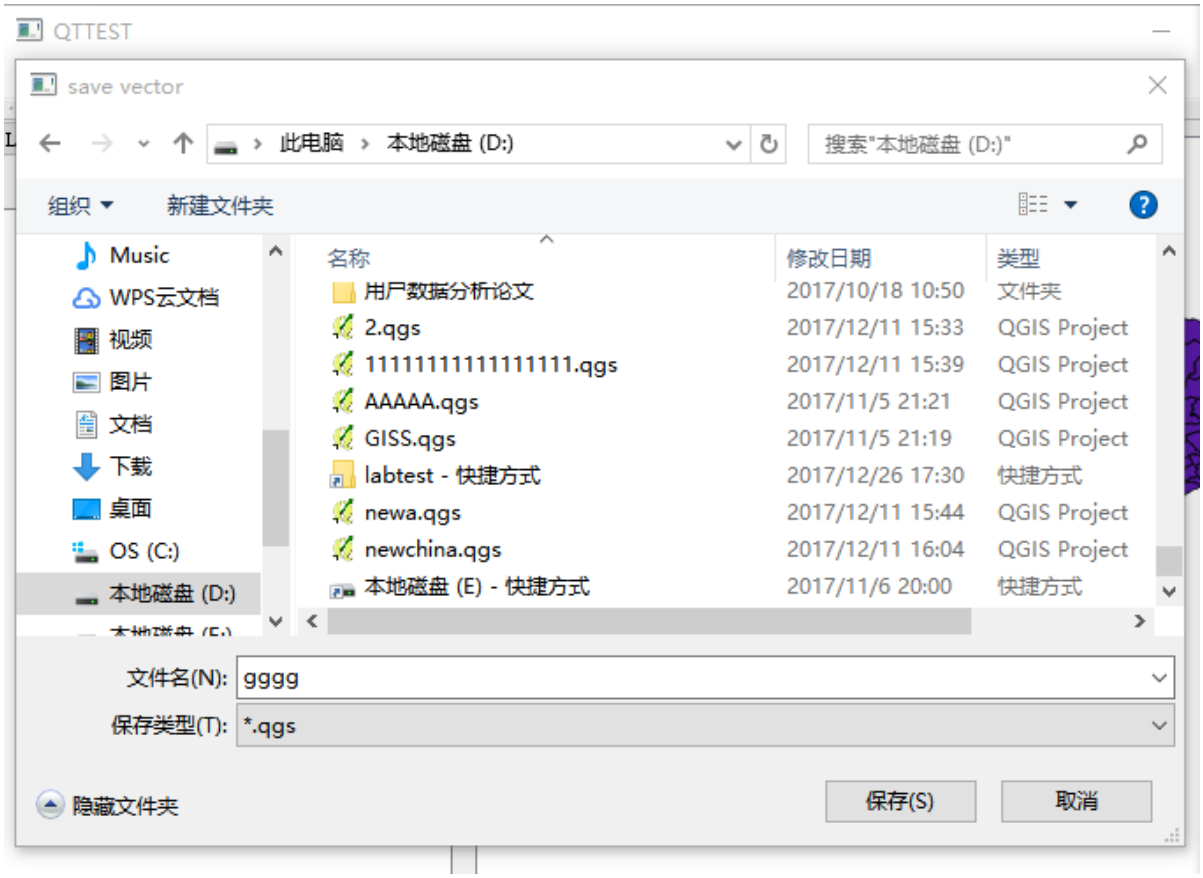
3.1.3Save project

Implementation code:

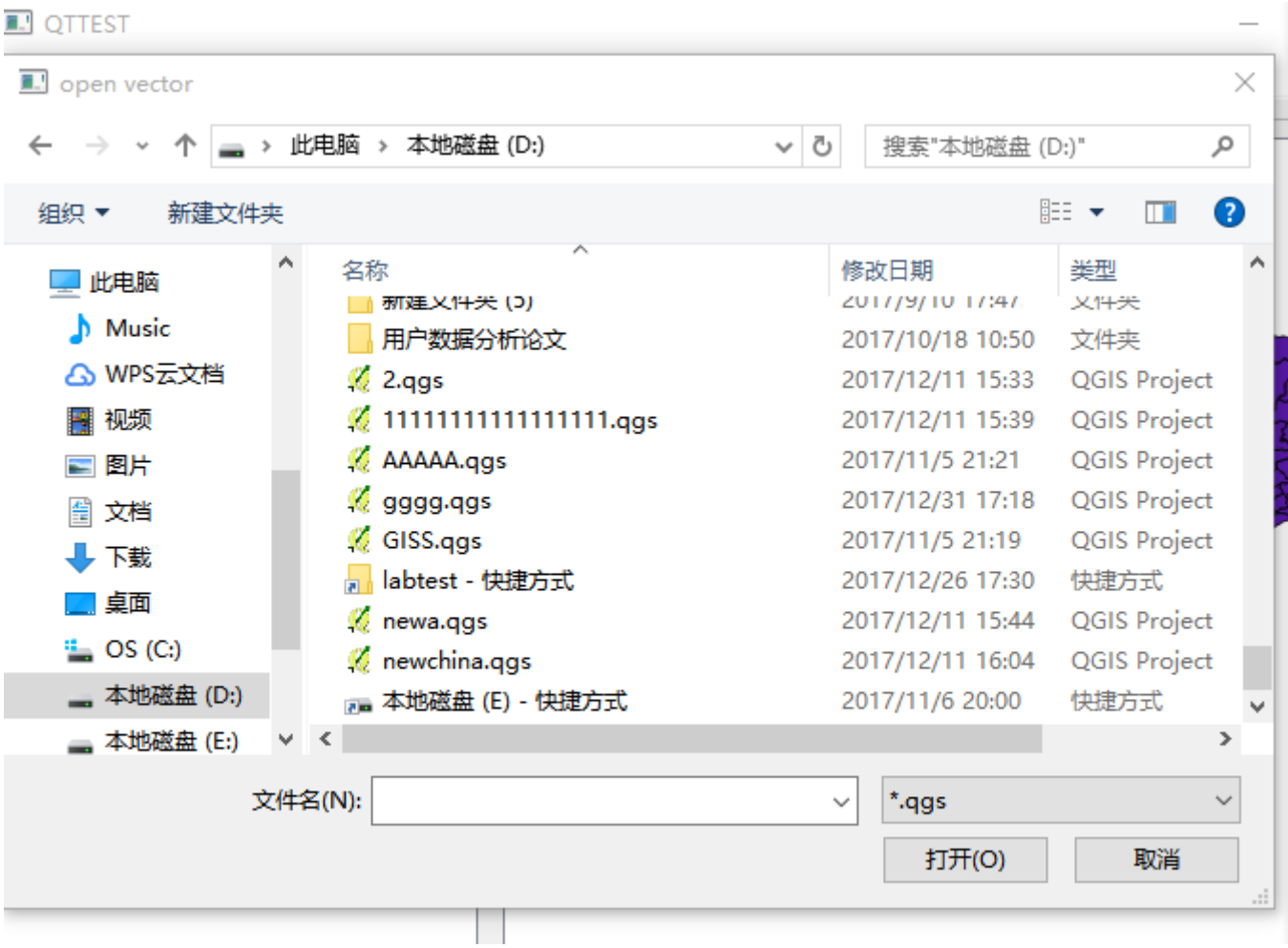
with openqgsProjects similar togetSaveFileName() function gets the file path name and then calls

QgsProjectofwritemethod willqgsSave the project to this path.

```
//保存项目
void QTTEST::savefile()
{


    QString filename = QFileDialog::getSaveFileName( this, tr( "save vector" ), "", "*.qgs" );
    QgsProject::instance()->write(QFileInfo(filename));

}
```


QTTEST
save vector
此电脑 › 本地磁盘 (D:)
搜索"本地磁盘 (D:)"
组织
新建文件夹
Music
WPS云文档
视频
图片
文档
下载
桌面
OS (C:)
本地磁盘 (D:)
名称
修改日期
类型
用尸数据分析论文 2017/10/18 10:50 文件夹
2.qgs 2017/12/11 15:33 QGIS Project
1111111111111111.qgs 2017/12/11 15:39 QGIS Project
AAAAA.qgs 2017/11/5 21:21 QGIS Project
GISS.qgs 2017/11/5 21:19 QGIS Project
labtest - 快捷方式 2017/12/26 17:30 快捷方式
newa.qgs 2017/12/11 15:44 QGIS Project
newchina.qgs 2017/12/11 16:04 QGIS Project
本地磁盘 (E) - 快捷方式 2017/11/6 20:00 快捷方式
文件名(N): gggg
保存类型(T): *.qgs
隐藏文件夹
保存(S)
取消

Again

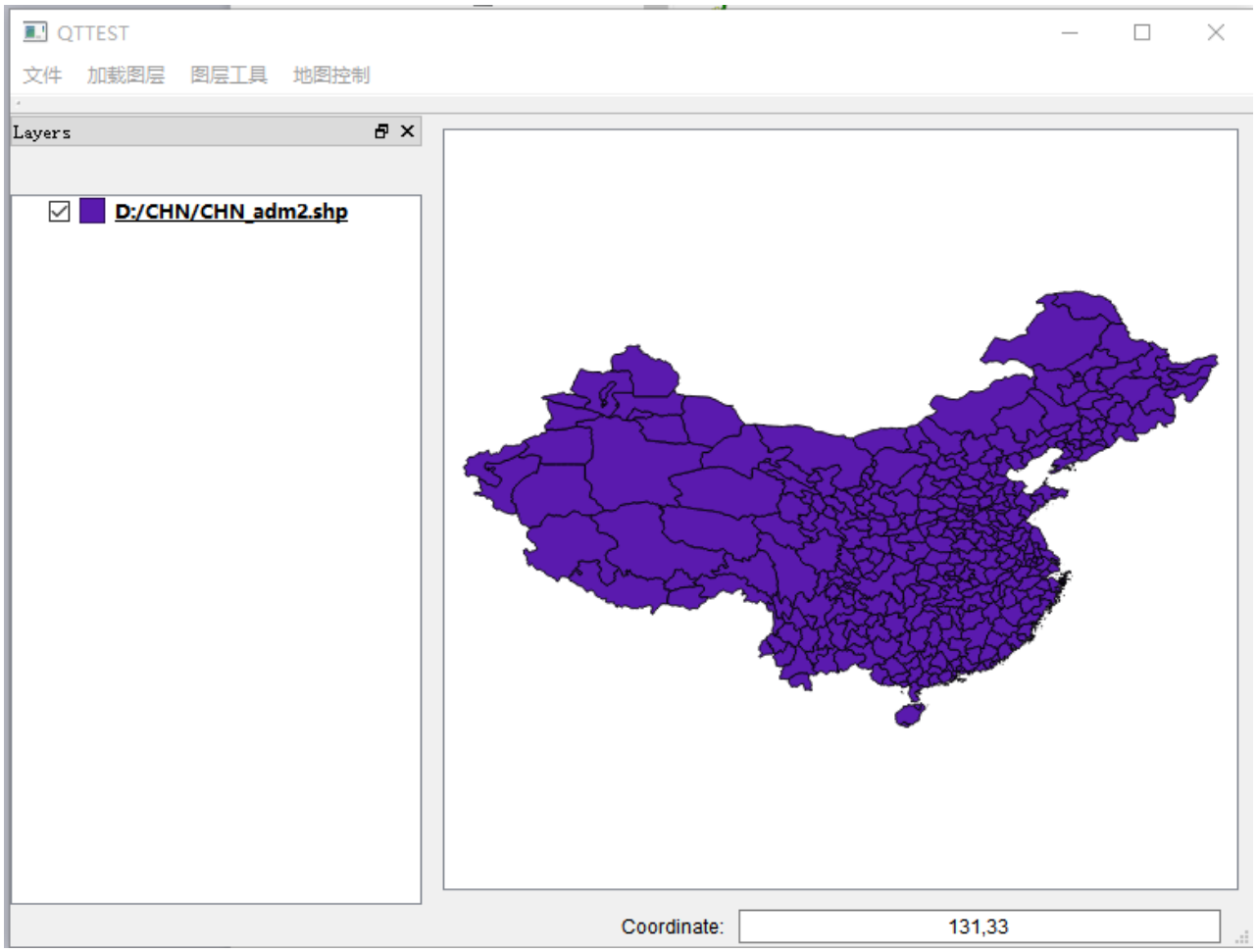


3.1.4Close project (exitGIS)

transferQTofQAapplicationclass, in classquit()Functions can exitQTprogram, and click on the upper right corner

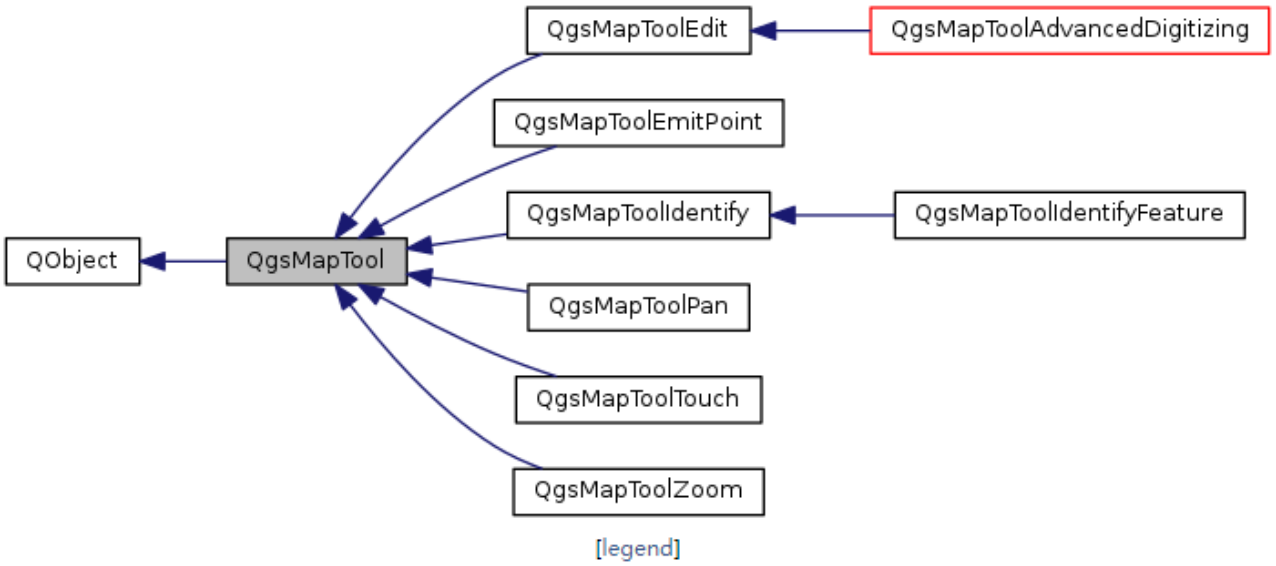
The closing effect is similar.

```
//关闭GIS
void QTTEST::closefile()
{


    QApplication* app;
    app->quit();

}
```


QTTEST
文件
加载图层
图层工具
地图控制
Layers
D:/CHN/CHN_adm2.shp
Coordinate:
131,33

3.2Map operations

QGIS, the operation on the map is inQGISofAPIdocumentQgsMapToolmiddle



inQgsMapToolPanAble to achieve map panning,QgsMapToolZoomAbility to place maps

Big reduction. Three functions can be achieved by calling the corresponding constructor. pass againQgsMapTooldof

setMapToolThe function calls the corresponding layer tool.

3.2.1Map zoom in and out

Implementation code:

L,SforQgsMapTool*type, asQgsMapToolZoomThe constructor parameters offalseto enlarge

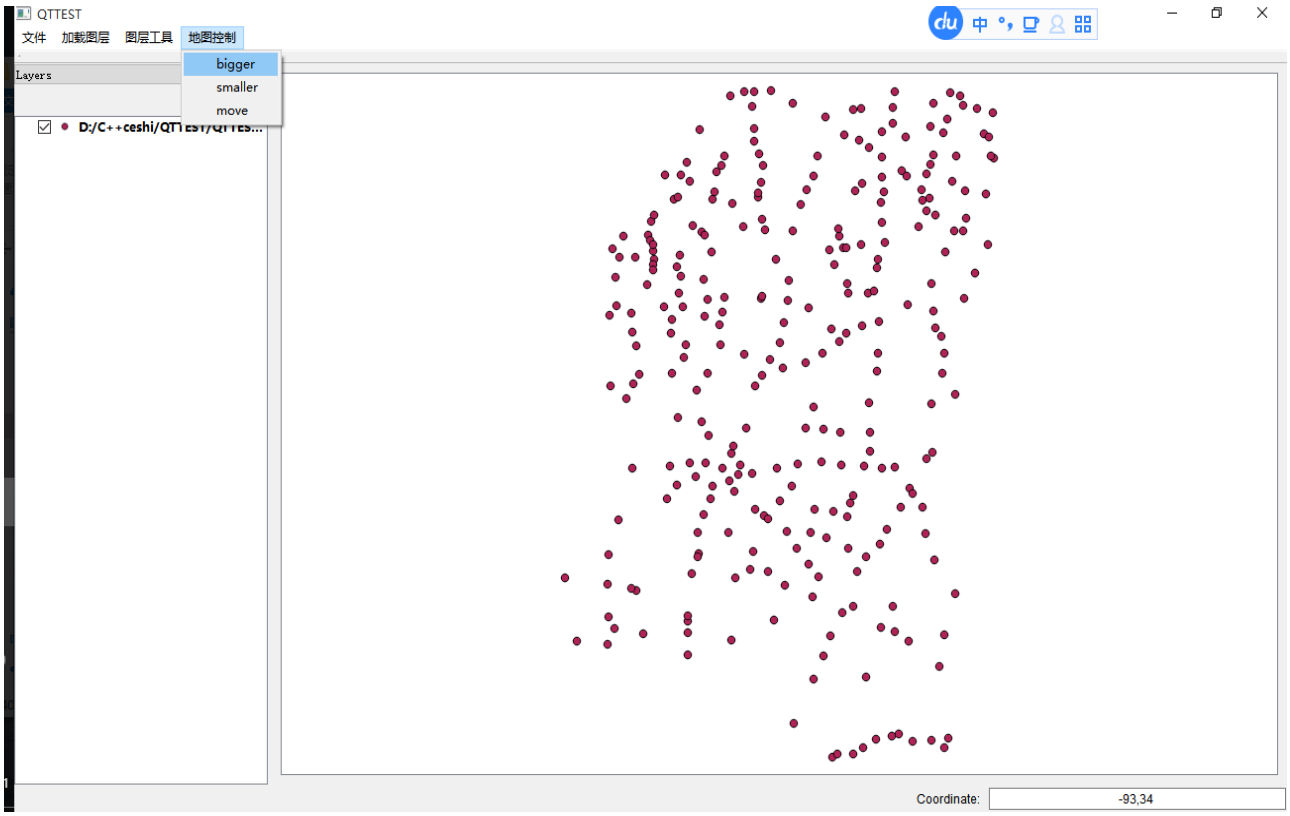
Function,truefor the zoom function.

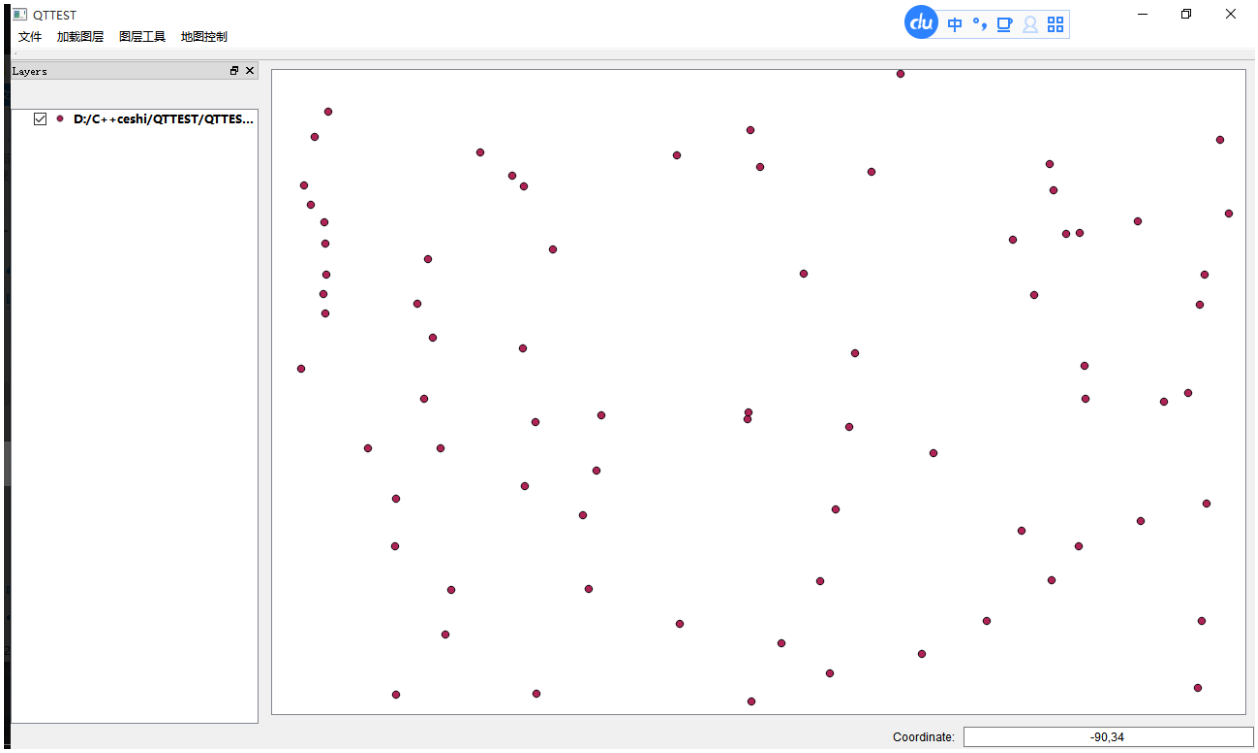
```
//*********************************************************************
//地图放大、缩小、移动
//地图放大
void QTTEST::Large(){
    L=new QgsMapToolZoom(m_mapCanvas,false);
    m_mapCanvas->setMapTool(L);
}
//地图缩小
void QTTEST::Small(){
    S=new QgsMapToolZoom(m_mapCanvas,true);
    m_mapCanvas->setMapTool(S);
}
```

Realization effect:

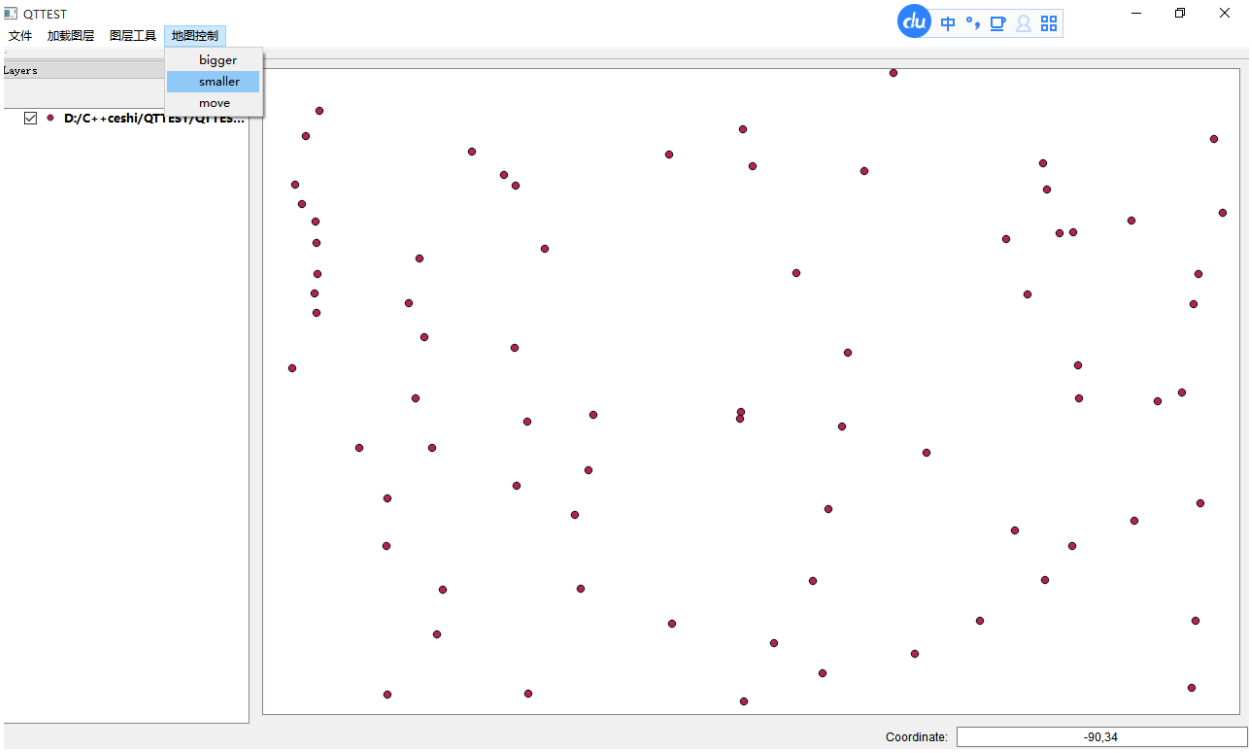
(Click on the corresponding function, the mouse image will turn into a magnifying glass + - oh)

enlarge





zoom out



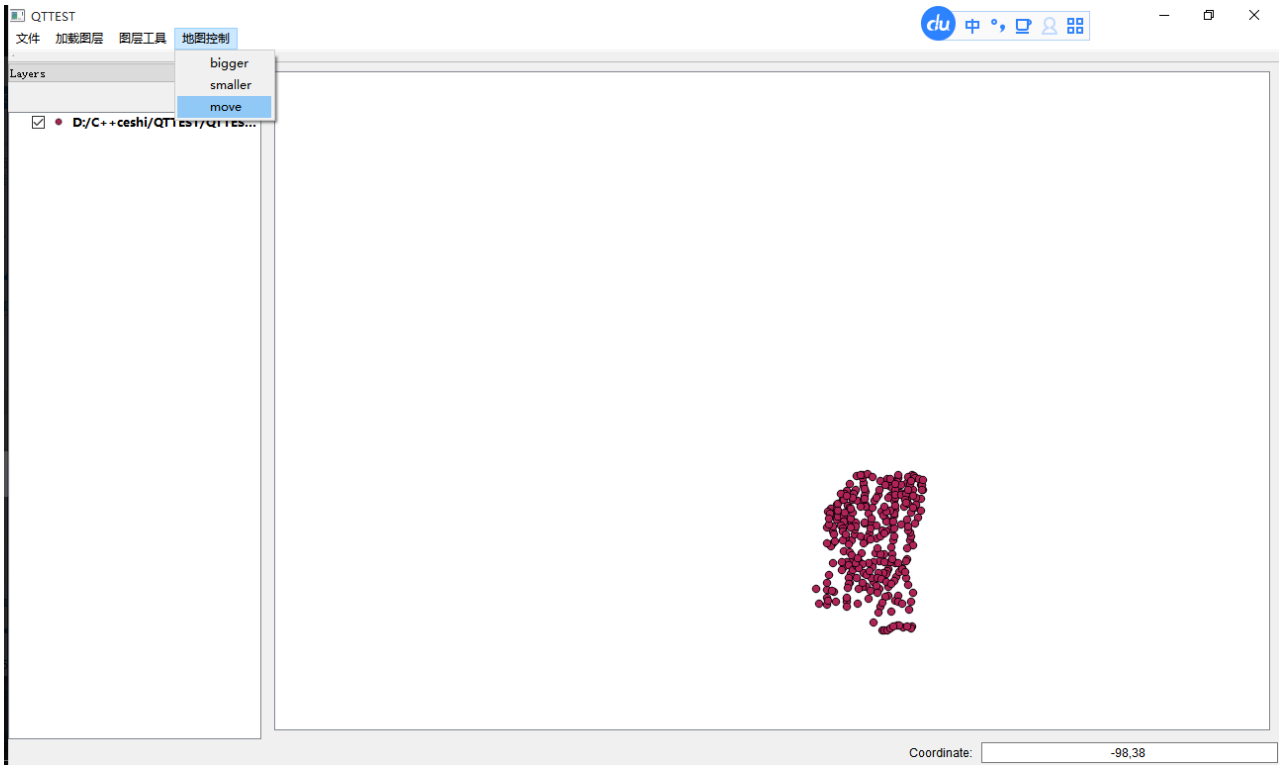
3.2.2Map pan

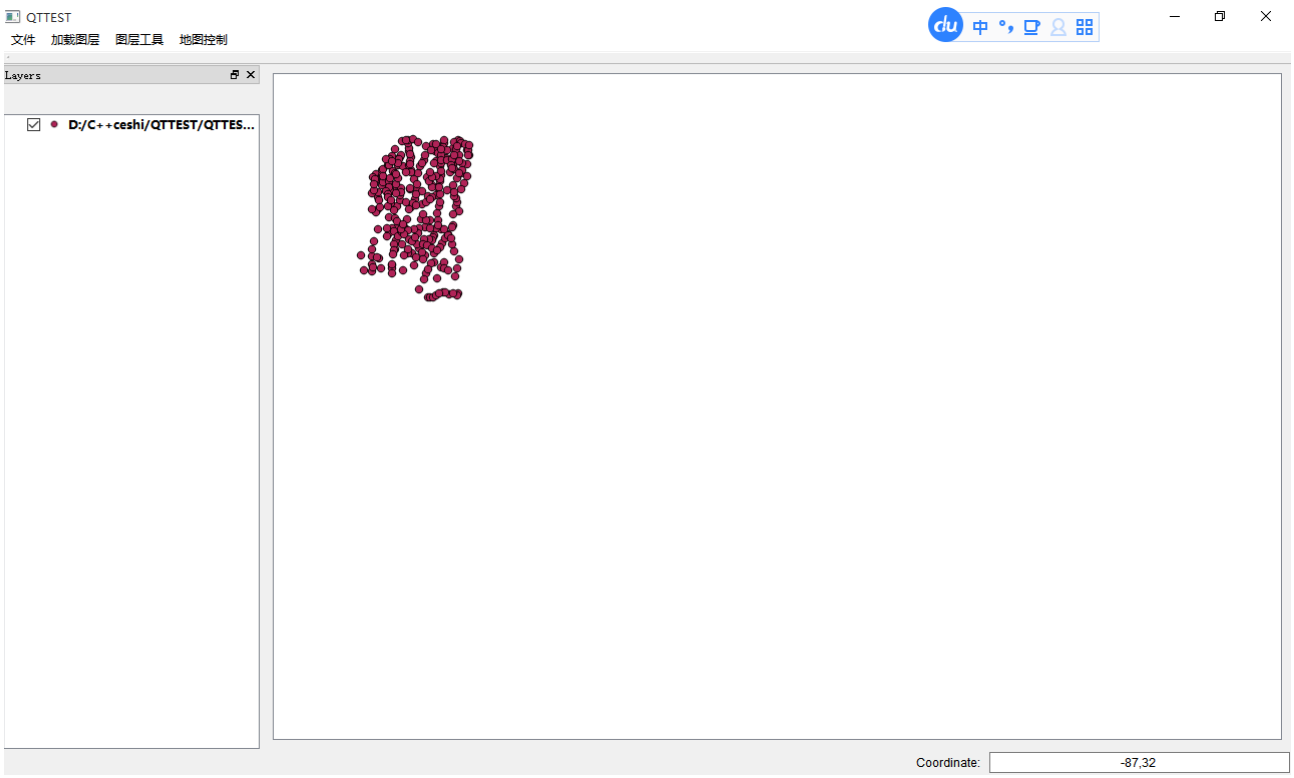
Implementation code:

Mstill oneQgsMapTool*type,QgsMapToolPanThe constructor can implement the translation function.

```
//地图移动
void QTTEST::Move(){
    M=new QgsMapToolPan(m_mapCanvas);
    m_mapCanvas->setMapTool(M);
}
```

Realization effect:





3.2.3Map ranging (replaced by the mouse coordinate display function in the map)

QgisofappIn the directory, there is aqgsmeasuretool.hFunction, similar to zooming in and out, this class inherits

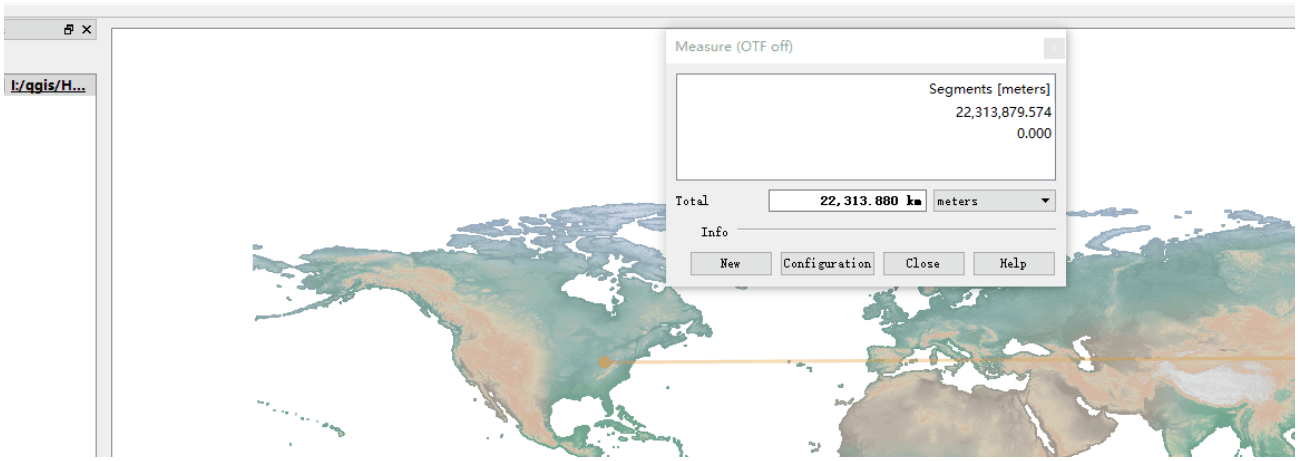
sinceQgsMapTool. Create a new oneQgsMeasureToolClass object, and then call the map ranging tool

Implement map ranging. Unfortunately, due to version issues, my header file cannot be imported correctly on my computer.

Tried many times, still useless, but this code of mine works in other versionsqgisIt can be run in.

```
//*************************************************************
//地图测距
void QTTEST::measure()
{
    //a=new QgsMeasureTool(m_mapCanvas,false);
    m_mapCanvas->setMapTool(a);
}
```

The effect is as shown in the figure:



This method comes fromQgsProjectkind

Implementation code:

Function passed inQgsPointA reference type variable is an object at a certain point in the map.QLineEdit*

m_coordsEditThe value of the coordinate point where the current mouse is located can be displayed and displayed in the bottom status bar.

```
//*************************************************************
//获取当前鼠标所指地图的坐标
void QTTEST::showMouseCoordinate( const QgsPoint &p )
{
    if ( m_mapCanvas->mapUnits() == QGis::Degrees ) // 坐标度分秒的显示方式
    {
        QString format = QgsProject::instance()->readEntry( "PositionPrecision", "/DegreeFormat", "D" );
        if ( format == "DM" ) // 只显示度和分
        {
            m_coordsEdit->setText( p.toDegreesMinutes( m_MousePrecisionDecimalPlaces ) );
        }
        else if ( format == "DMS" ) // 显示度分秒
        {
            m_coordsEdit->setText( p.toDegreesMinutesSeconds( m_MousePrecisionDecimalPlaces ) );
        }
        else
        {
            m_coordsEdit->setText( p.toString( m_MousePrecisionDecimalPlaces ) );
        }
    }
    else
    {
        m_coordsEdit->setText( p.toString( m_MousePrecisionDecimalPlaces ) );
    }

    if ( m_coordsEdit->width() > m_coordsEdit->minimumWidth() )
    {
        m_coordsEdit->setMinimumWidth( m_coordsEdit->width() );
    }
}
```

Use connect in the main function to use the mouse position xyCoordinates (const QgsPoint&) as the touch

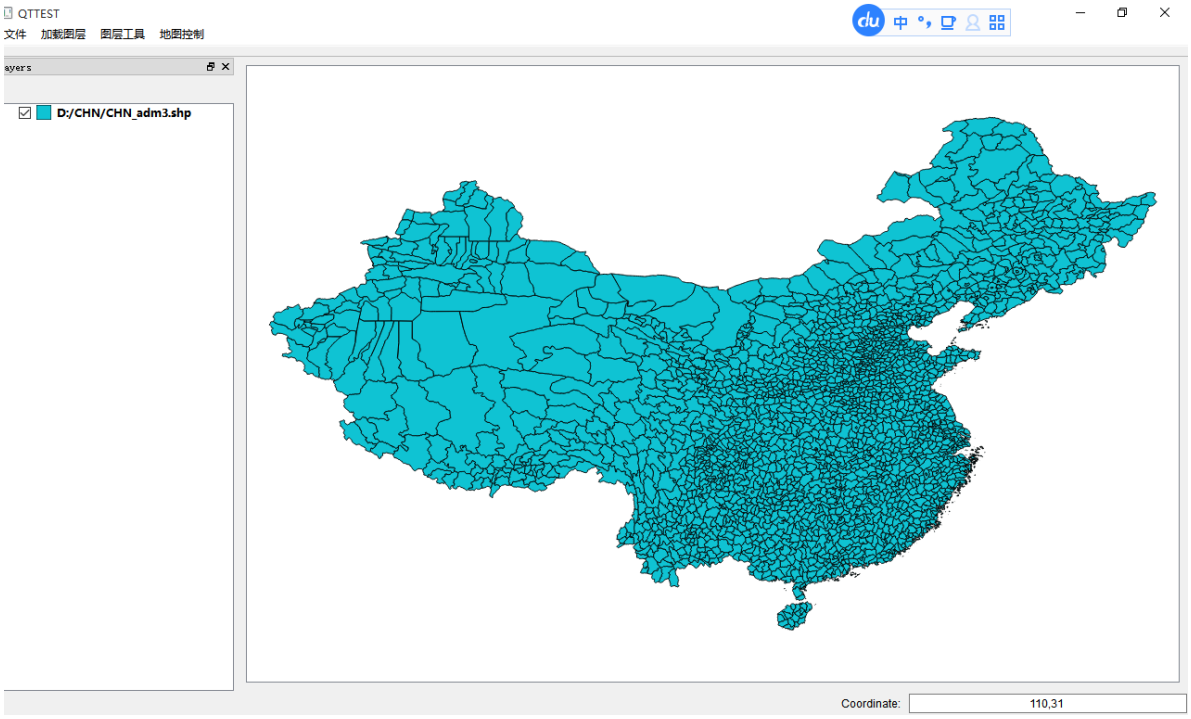
Hair generator. Display coordinate point values in real time.

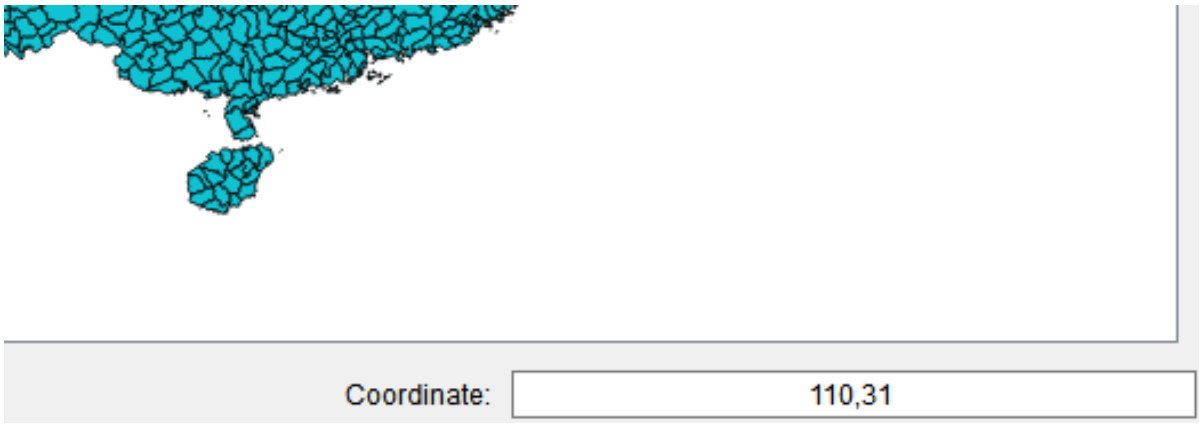
```
    connect( m_mapCanvas, SIGNAL( xyCoordinates( const QgsPoint& ) ), this, SLOT( showMouseCoordinate( const QgsPoint& ) ) );
1
```

Realization effect:

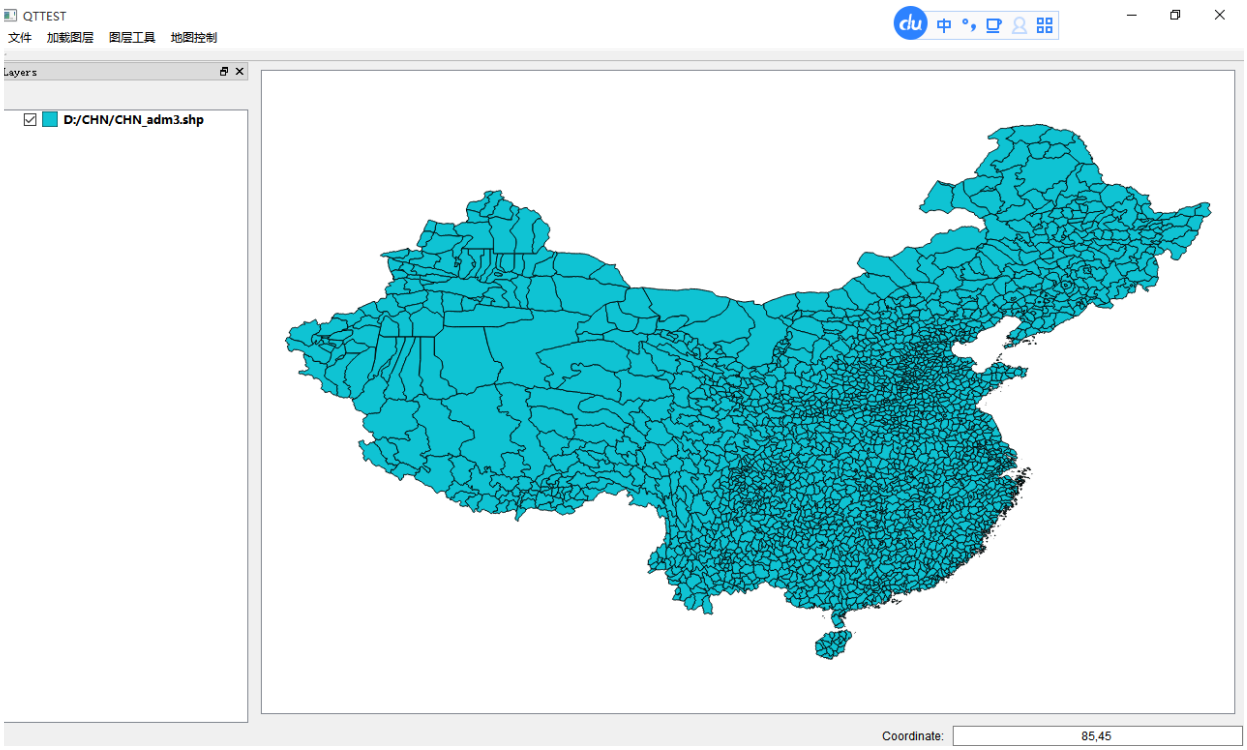
Since the mouse cannot be displayed in the screenshot, let me explain here. My mouse is now pointing to Wuhan on the map.

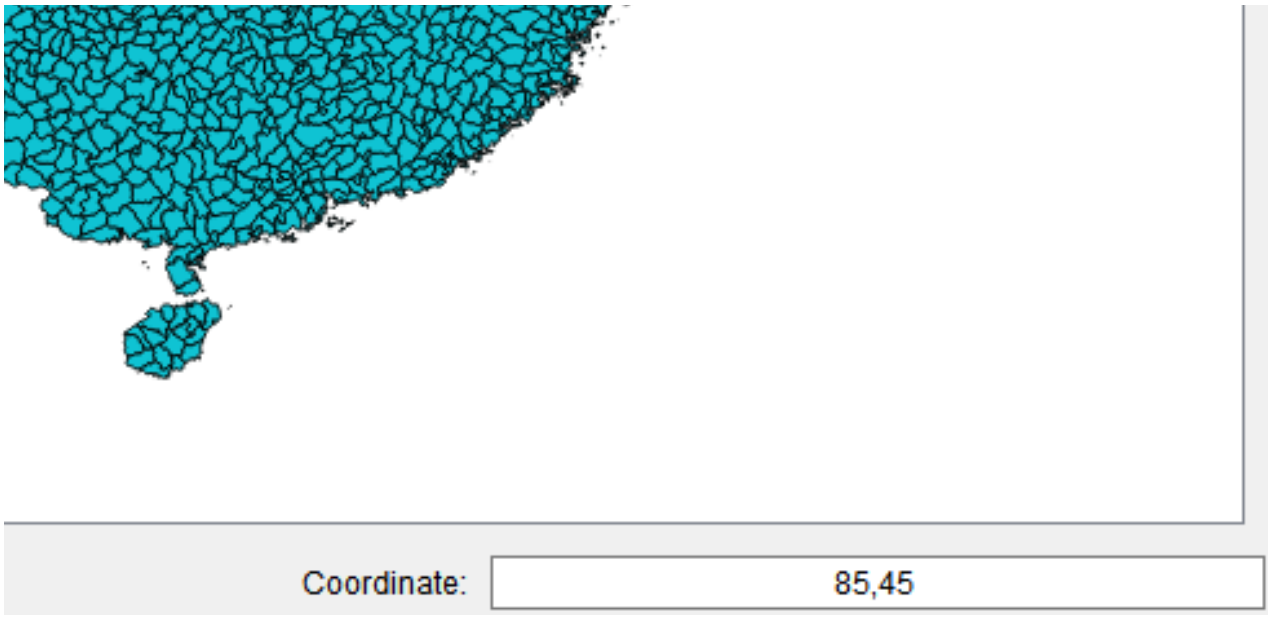
Set.





I pointed the mouse to a certain location in the northwest





After obtaining the two coordinate points, the corresponding distance calculation can be performed, and then after converting the obtained distance,

The actual distance between two points can be measured.

Additional features:**Display mouse coordinates**The status bar at the bottom is implemented, and the effect is as shown above.

Implementation code:

```
void QTTEST::createStatusBar()
{
    statusBar()->setStyleSheet( "QStatusBar::item {border: none;}" );

    QFont myFont( "Arial", 9 );
    statusBar()->setFont( myFont );


    //添加坐标显示标签
    m_coordsLabel = new QLabel( QString(), statusBar() );
    m_coordsLabel->setObjectName( "m_coordsLabel" );
    m_coordsLabel->setFont( myFont );
    m_coordsLabel->setMinimumWidth( 10 );
    m_coordsLabel->setMargin( 3 );
    m_coordsLabel->setAlignment( Qt::AlignCenter );
    m_coordsLabel->setFrameStyle( QFrame::NoFrame );
    m_coordsLabel->setText( tr( "Coordinate:" ) );
    m_coordsLabel->setToolTip( tr( "Current map coordinate" ) );
    statusBar()->addPermanentWidget( m_coordsLabel, 0 );

    //坐标编辑标签
    m_coordsEdit = new QLineEdit( QString(), statusBar() );
    m_coordsEdit->setObjectName( "m_coordsEdit" );
    m_coordsEdit->setFont( myFont );
    m_coordsEdit->setMinimumWidth( 10 );
    m_coordsEdit->setMaximumWidth( 300 );
    m_coordsEdit->setContentsMargins( 0, 0, 0, 0 );
    m_coordsEdit->setAlignment( Qt::AlignCenter );
    statusBar()->addPermanentWidget( m_coordsEdit, 0 );
    m_coordsEdit->setReadOnly( true );//只读
    connect( m_coordsEdit, SIGNAL( returnPressed() ), this, SLOT( userCenter() ) );

    m_dizzyTimer = new QTimer( this );
    connect( m_dizzyTimer, SIGNAL( timeout() ), this, SLOT( dizzy() ) );
```

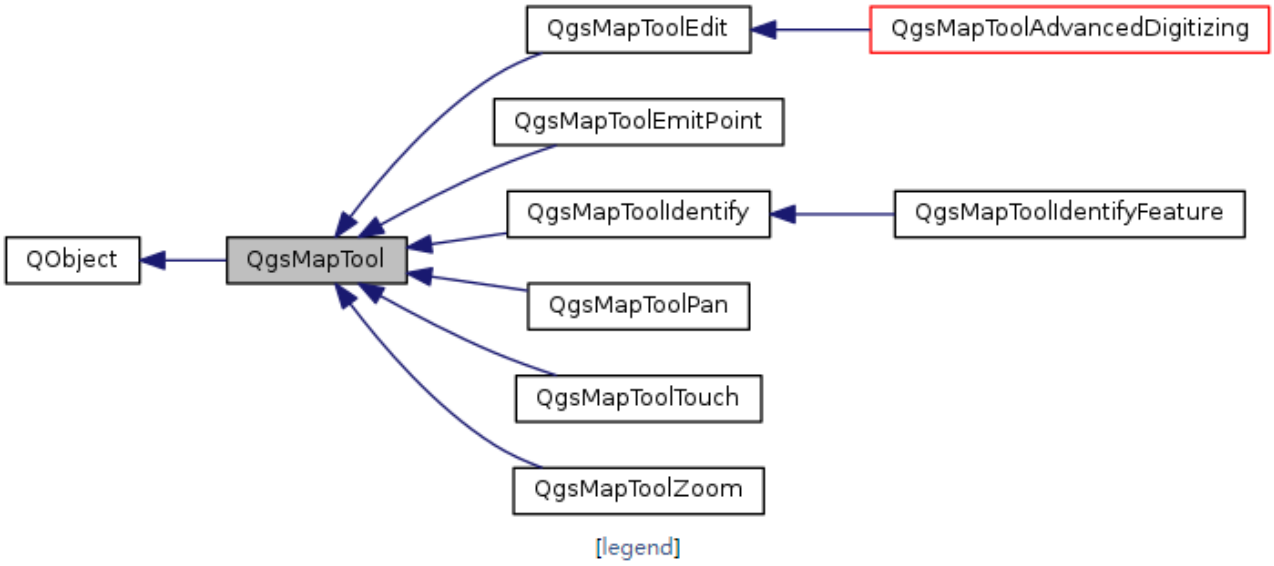
3.3Layer management

3.3.1Graph element recognition

Graph element identification refers to the ability to identify the attribute values of the graph elements in the map selected by the mouse. The implementation of this function depends on

NowQGISofAPIofQgsMapToolclass. inQgsMapToolIdentifyClasses can implement

A class that implements attribute recognition functionality.



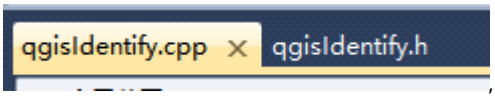
Because in this class, the function that responds to mouse events does not have a corresponding response action.

```
void QgsMapToolIdentify::canvasMoveEvent( QMouseEvent * e )
{
  Q_UNUSED( e );
}

void QgsMapToolIdentify::canvasPressEvent( QMouseEvent * e )
{
  Q_UNUSED( e );
}

void QgsMapToolIdentify::canvasReleaseEvent( QMouseEvent * e )
{
  Q_UNUSED( e );
}
```

Therefore, if this function is called directly, the primitive recognition function will not be implemented. Therefore, you need to define a consciousness yourself

Category and rewrite the mouse release event function to achieve true primitive recognition.

Implementation code:

I created a new CPP and its header file 

, used for

Write a custom primitive recognition class.

Header file code: Set the mouse release function as a virtual function to facilitate overriding the method.**One thing to note here is that I want to implement the function of primitive recognition. This primitive is a layer in QGIS. The event occurs on the layer, not just in the QT panel, so it must respond to the corresponding mouse release. The event type must be of QgsMapMouseEvent type, not QT's QMouseEvent, in order to get the click release event of the correct layer.**

qgisIdentify.cpp | qgisIdentify.h | qttest.h | qttest.cpp | main.cpp | 起始页

(全局范围)

```
#ifndef QGISIDENTIFY
#define QGISIDENTIFY

#include <QObject>
#include <QMouseEvent>

#include <qgsmaptoolidentify.h>

//! 用于实现属性识别工具
class qgisIdentify : public QgsMapToolIdentify
{
    Q_OBJECT

public:
    qgisIdentify( QgsMapCanvas * canvas );
    ~qgisIdentify();

    //! 重写鼠标键释放事件
    virtual void canvasReleaseEvent(/* QMouseEvent * e*/ QgsMapMouseEvent* e);

};

#endif // QGISIDENTIFY
```

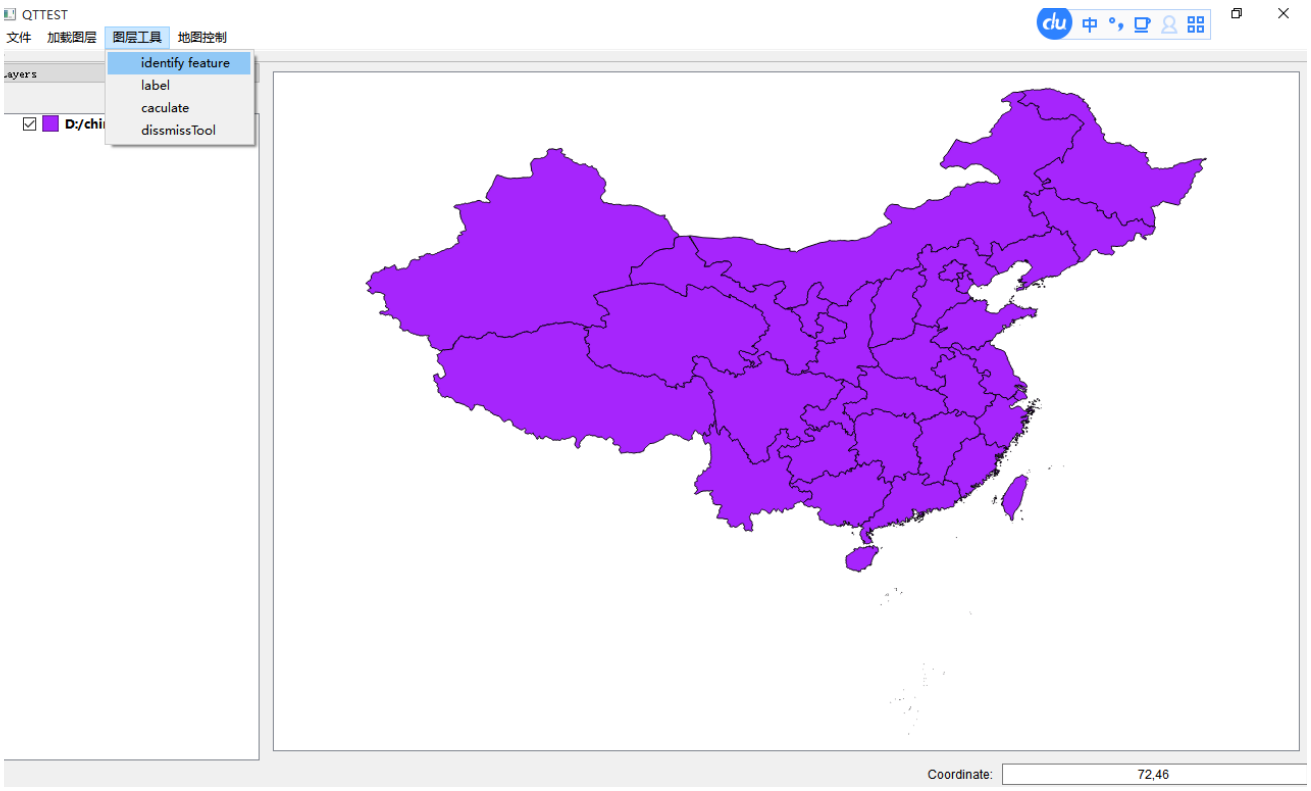
Rewritten mouse release event method:

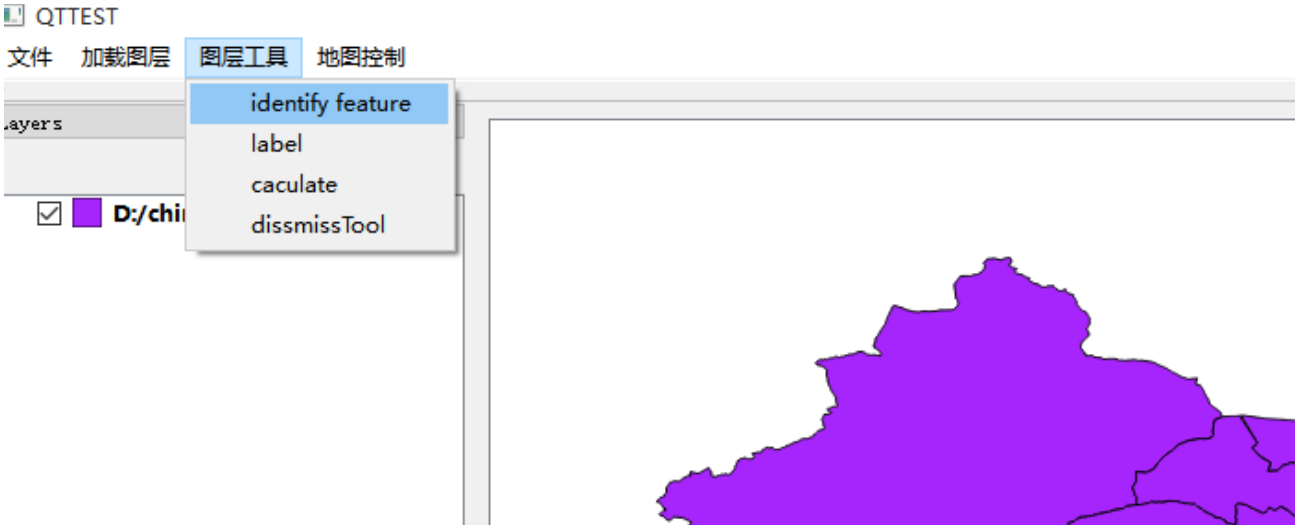
```
void qgisIdentify::canvasReleaseEvent( /*QMouseEvent * e*/QgsMapMouseEvent *e )
{
    IdentifyMode mode = QgsMapToolIdentify::LayerSelection; // 控制识别模式
    QList<IdentifyResult> results = QgsMapToolIdentify::identify( e->x(), e->y(), mode ); // 这句返回识别结果

    if ( results.isEmpty() )
    {
        QTTEST::instance()->statusBar()->showMessage( tr( "No features at this position found." ) );
    }
    else
    {
        // 显示出识别结果，这里仅作示例，结果的展示方式可以自定义
        IdentifyResult feature = results.at( 0 );
        QString title = feature.mLayer->name();
        QString content = feature.mFeature.attribute( 1 ).toString();
        // 显示识别窗口
        QMessageBox::critical( NULL,
                               title,
                               content );
    }
}
```

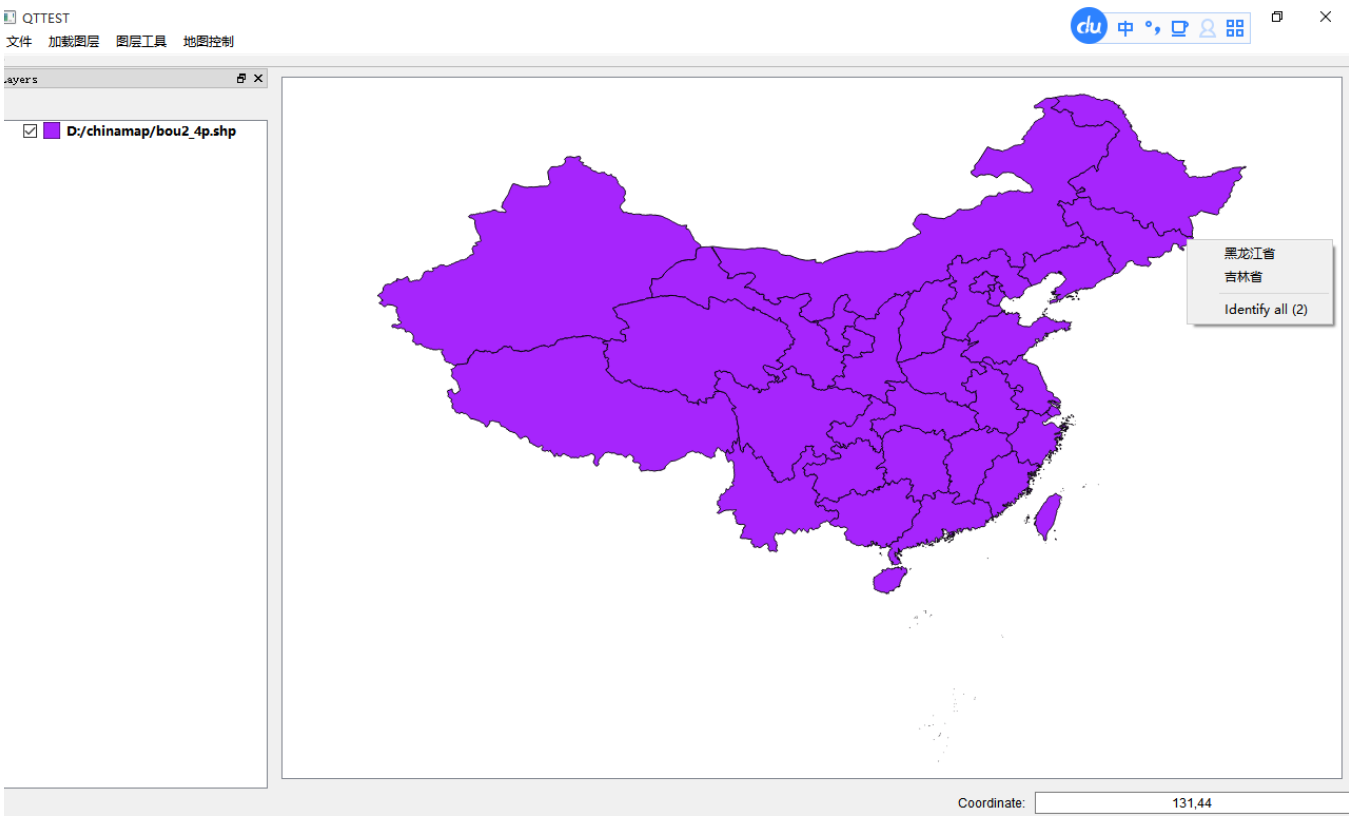
Create a new constructor of this class and activate it when the function is called, and then call the identification tool.

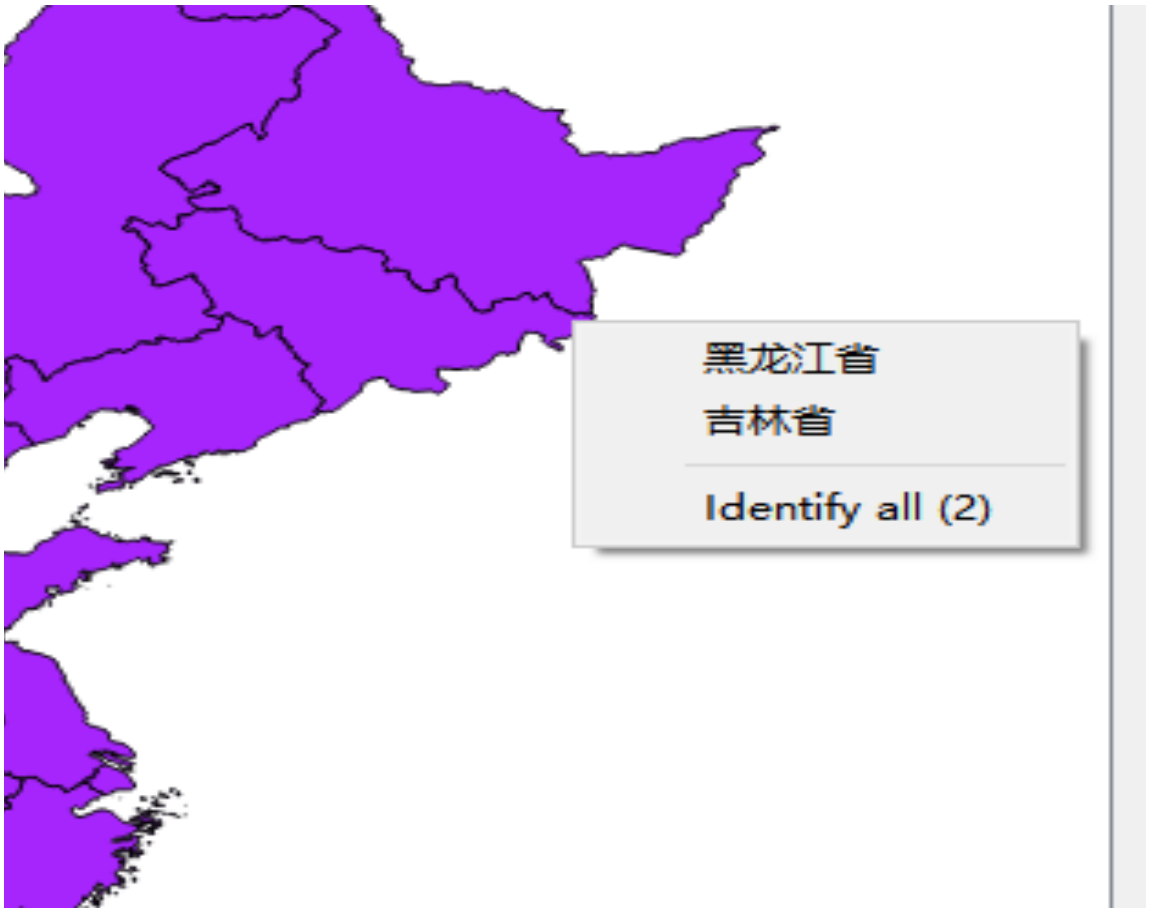
```
//*********************************************************************
//图元识别
void QTTEST::qIdentify()
{
    m_mapIdentify=new qgisIdentify(m_mapCanvas);
    m_mapIdentify->activate();
    m_mapCanvas->setMapTool(m_mapIdentify);

}
```

Realization effect:



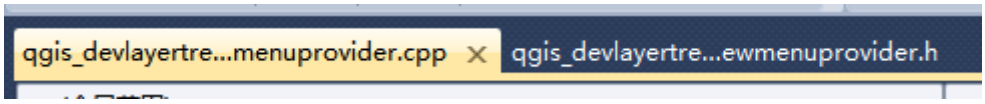



QTTEST
文件 加载图层 图层工具 地图控制
D:/chinamap/bou2_4p.shp
黑龙江省
吉林省
Identify all (2)
Coordinate:
131,44


黑龙江省
吉林省
Identify all (2)

Additional features:

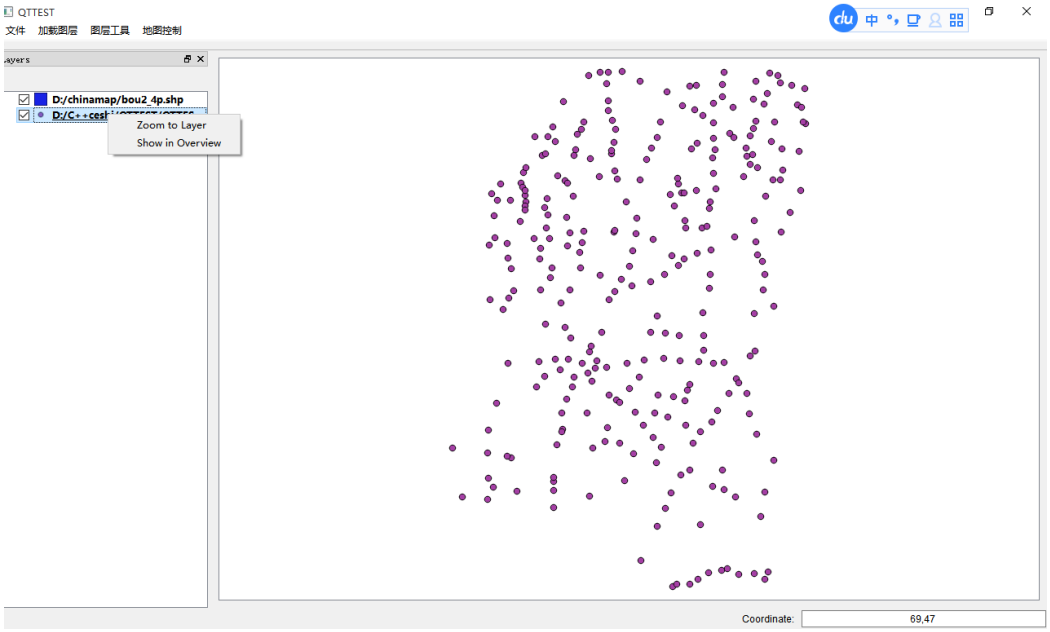
Before adding a layer, you must first implement the function of the layer manager, which is in the following file.



which rewritescreateContextMenu()Method can obtain the right-click menu function.

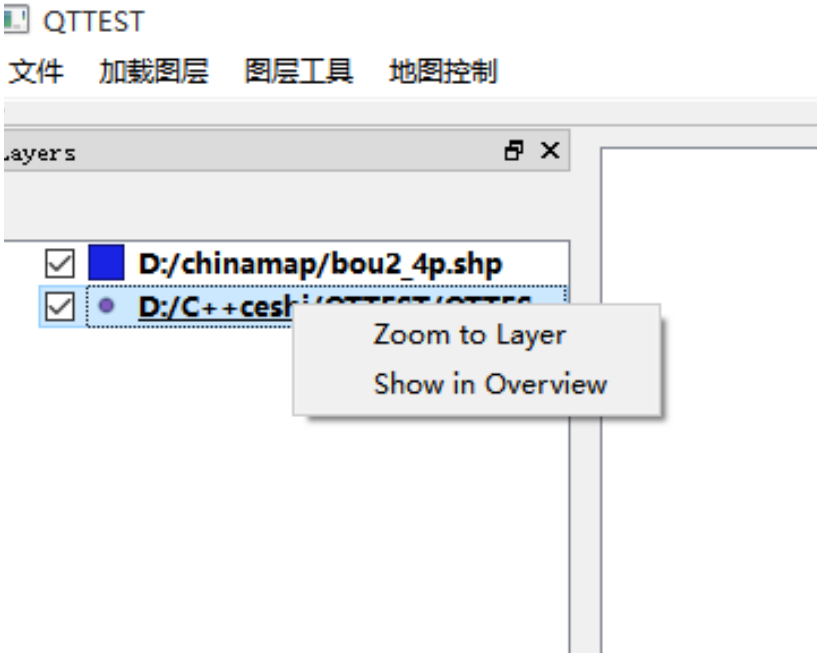
```
QMenu* qgis_devLayerTreeViewMenuProvider::createContextMenu()
{
    // 设置这个路径是为了获取图标文件
    QString iconDir = "../images/themes/default/";

    QMenu* menu = new QMenu;

    QgsLayerTreeViewDefaultActions* actions = mView->defaultActions();

    QModelIndex idx = mView->currentIndex();

    // global menu
    if ( !idx.isValid() )
    {
        menu->addAction( actions->actionAddGroup( menu ) );
        menu->addAction( QIcon( iconDir + "mActionExpandTree.png" ), tr( "&Expand All" ), mView, SLOT( expandAll() ) );
        menu->addAction( QIcon( iconDir + "mActionCollapseTree.png" ), tr( "&Collapse All" ), mView, SLOT( collapseAll() ) );
        menu->addAction( QIcon( iconDir + "mActionRemoveLayer.svg" ), tr( "&Remove" ), QTTEST::instance(), SLOT( removeLayer() ) );
    }
    else if ( QgsLayerTreeNode* node = mView->layerTreeModel()->index2node( idx ) )
    {
        // layer or group selected
        if ( QgsLayerTree::isGroup( node ) )
        {
            menu->addAction( actions->actionZoomToGroup( mCanvas, menu ) );
            menu->addAction( QIcon( iconDir + "mActionRemoveLayer.svg" ), tr( "&Remove" ), QTTEST::instance(), SLOT( removeLayer() ) );
            menu->addAction( actions->actionRenameGroupOrLayer( menu ) );
            if ( mView->selectedNodes( true ).count() >= 2 )
            {
                menu->addAction( actions->actionGroupSelected( menu ) );
            }
            menu->addAction( actions->actionAddGroup( menu ) );
        }
        else if ( QgsLayerTree::isLayer( node ) )
        {
            QgsMapLayer* layer = QgsLayerTree::toLayer( node )->layer();
            menu->addAction( actions->actionZoomToLayer( mCanvas, menu ) );
            menu->addAction( actions->actionShowInOverview( menu ) );

            // 如果选择的是栅格图层
            if ( layer && layer->type() == QgsMapLayer::RasterLayer )
            {
                // TO DO
            }
        }
    }

    return menu;
}
```

Realization effect:



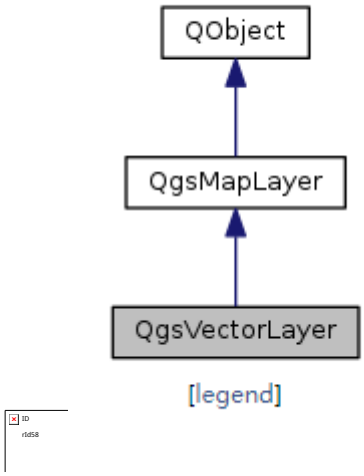
Zoom to Layer implements the function of displaying the selected layer on top, Show in Overview

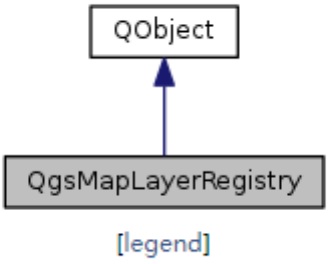
I haven't achieved that yet.



3.3.2 Add layer

The following two classes are needed in the QGIS API.





(1) Add vector map

Implementation code:

Filename, basename, temp are used to save path names and other content.

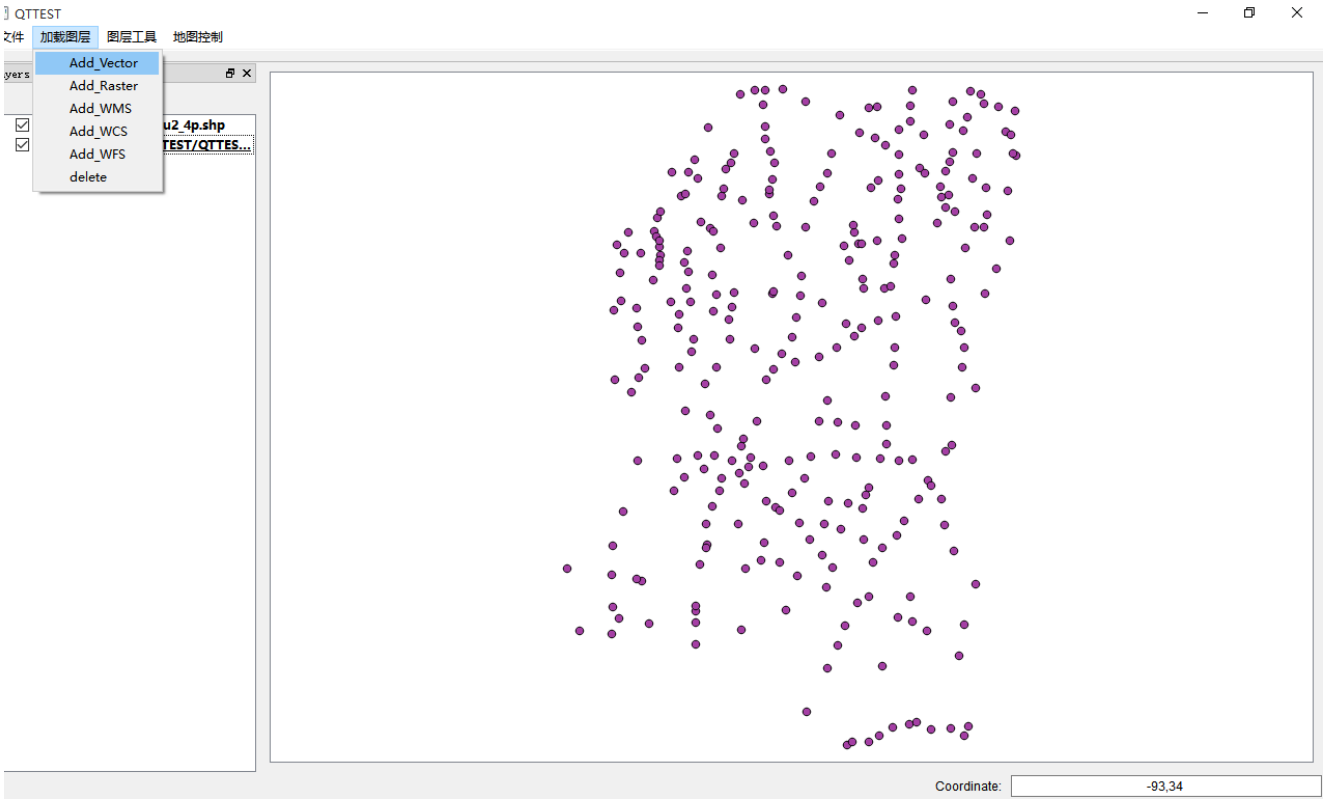
The constructor of QgsVectorLayer passes in the layer path, which can be obtained after matching the supported plug-in name.

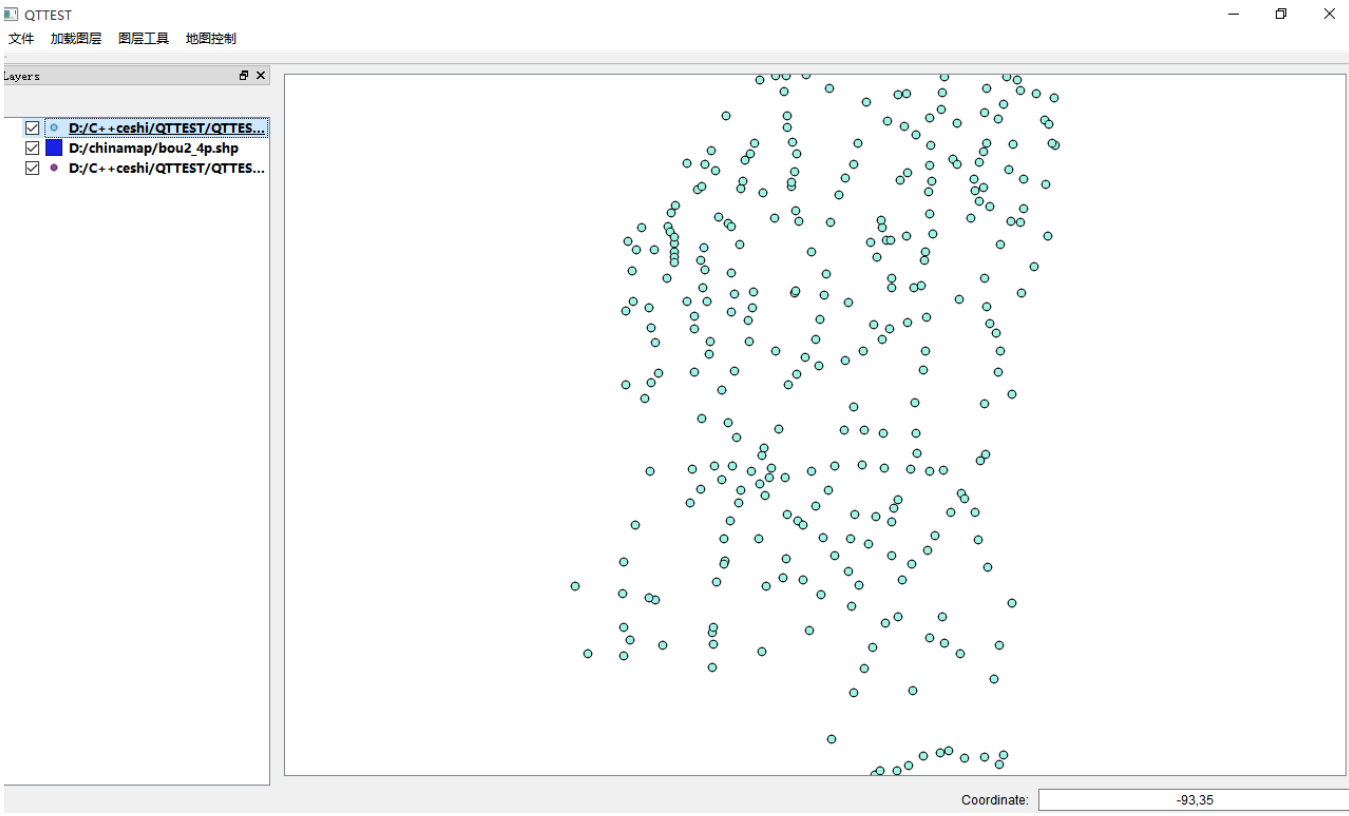
of layers. Then call the QgsMapLayerRegistry class to register the layer, and the m_mapCanvas object calls several

This function expands and displays the layer.

```cpp
//*****************************************************************
//添加各种图层
//Add_VectorLayers
void QTTEST::addVectorLayers()
{

    QString filename = QFileDialog::getOpenFileName( this, tr( "open vector" ), "", "*.shp" );
    QStringList temp = filename.split( QDir::separator() );
    QString basename = temp.at( temp.size() - 1 );
    QgsVectorLayer* vecLayer = new QgsVectorLayer(filename,basename,"ogr",true );
        //QMessageBox::critical(this, "error", filename);
        //QMessageBox::critical(this, "error", basename);
    if(!vecLayer->isValid()){

        QMessageBox::critical(this,"error","layer is invalid");
        return;

    }

    QgsMapLayerRegistry::instance()->addMapLayer( vecLayer);
    mapCanvasLayerSet.append( vecLayer );
    m_mapCanvas->setExtent( vecLayer->extent() );
    m_mapCanvas->setLayerSet( mapCanvasLayerSet );
    m_mapCanvas->setVisible( true );
    m_mapCanvas->freeze( false );
    m_mapCanvas->refresh();

    ////20171227 update
    //vecLayer->enableLabels(true);//显示图层标注
    //m_mapCanvas->refresh();
    ////end update

}
```

Realization effect:



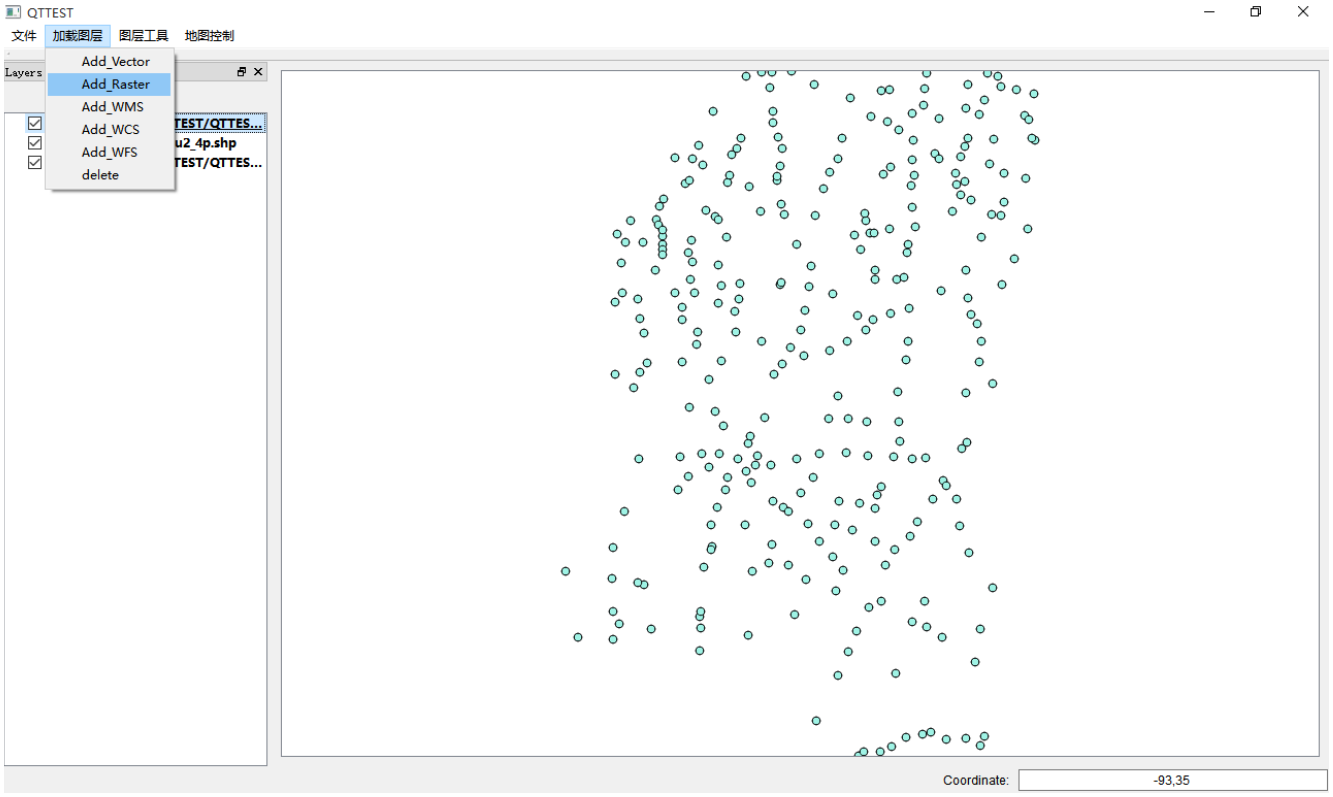


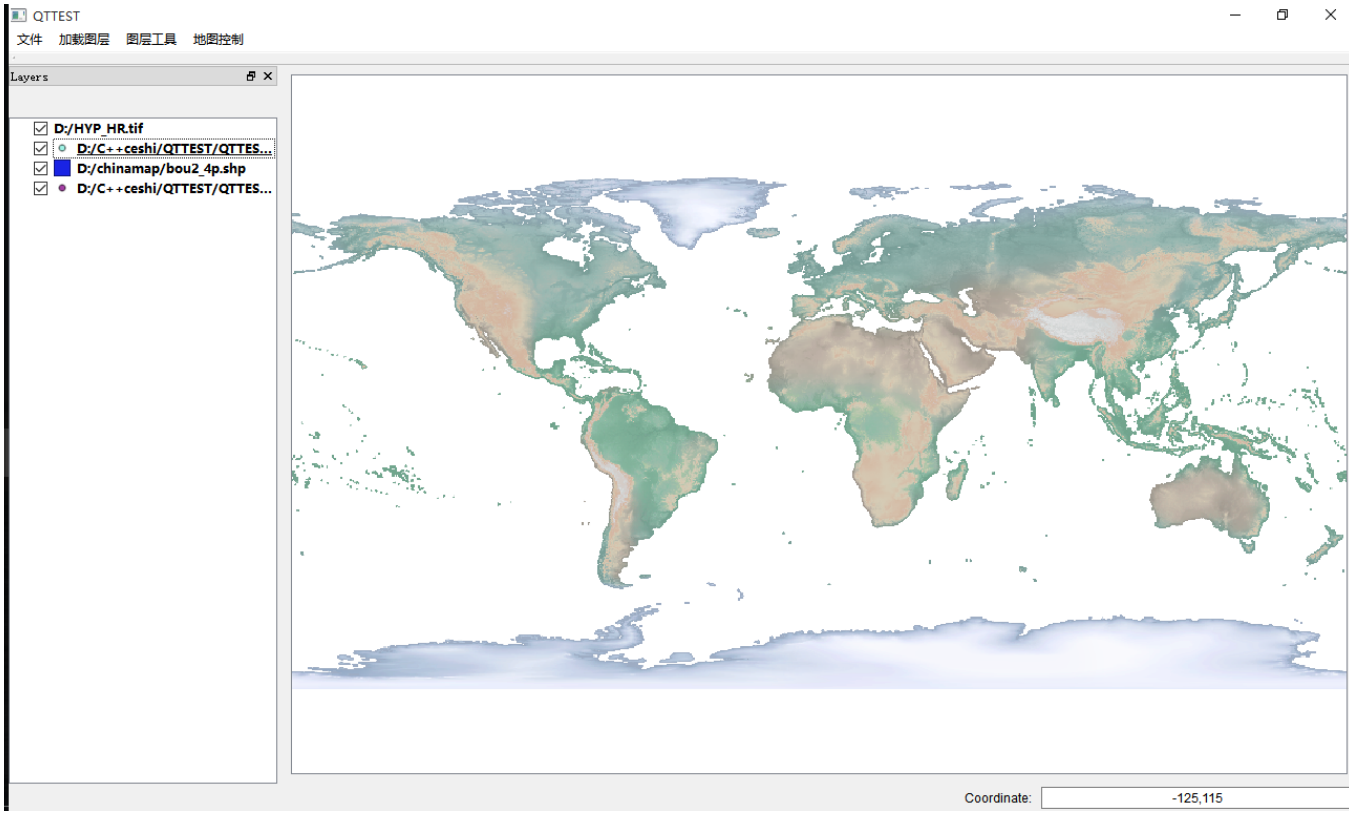
(2) Add image map

Implementation code: Similar to the vector map, you just need to change the plug-in name.

```
r( filename, basename, "gdal", false );
```

Realization effect:





(3) Add WFS, WCS, and WMS layers

WMS, WCS (similar to WFS) implementation code:

```
//ADD WMS
void QTTEST::addWMSLayers()
{

    QDialog *wms = dynamic_cast<QDialog*>(QgsProviderRegistry::instance()->selectWidget( QStr
    if ( !wms )
    {
        statusBar()->showMessage( tr( "cannot add wms layer." ), 10 );
    }

    connect( wms, SIGNAL( addRasterLayer( QString const &, QString const &, QString const & )
             this, SLOT( addOpenSourceRasterLayer( QString const &, QString const &, QString

    wms->exec();

    delete wms;
}
```

What needs to be noted is that the following code must be added to the main function before the connection can be successful.

```
QCoreApplication::setOrganizationName( "oneone" );
QCoreApplication::setOrganizationDomain( "oneone.com" ); // 域名好像是可以不用加的
QCoreApplication::setApplicationName( "QTTEST" );
```

WFS implementation code:

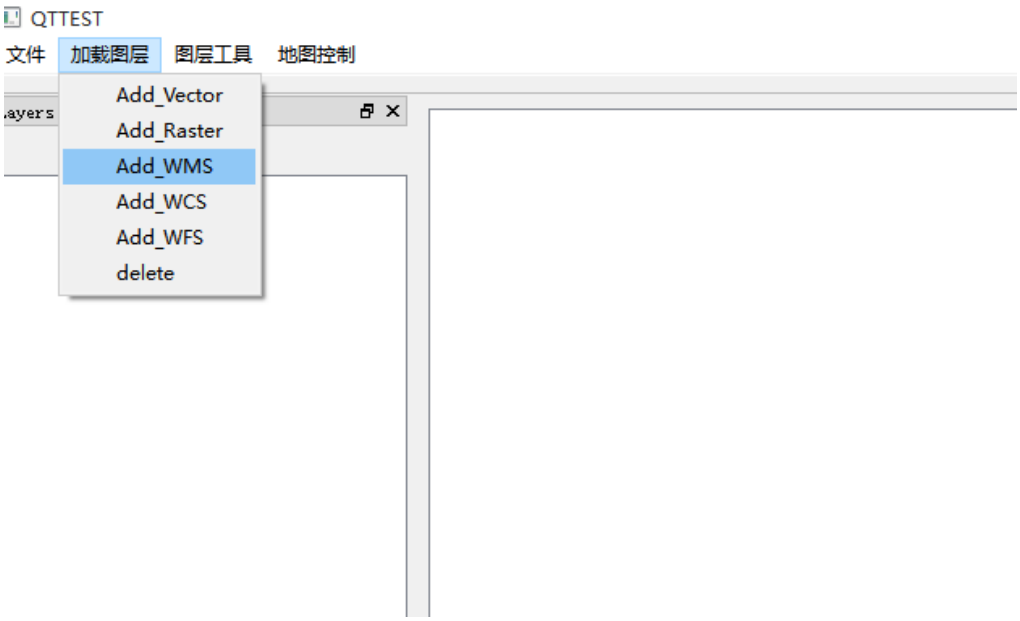
```
//ADD WFS
void QTTEST::addWFSLayers()
{
    if ( !m_mapCanvas ) {return;}

    QDialog *wfs = dynamic_cast<QDialog*>( QgsProviderRegistry::instance()->selectWidget( QString( "WFS" ), t
    if ( !wfs )
    {
        QMessageBox::warning( this, tr( "WFS" ), tr( "Cannot get WFS select dialog from provider." ) );
        return;
    }
    connect( wfs, SIGNAL( addWfsLayer( QString, QString ) ),
             this, SLOT( addWfsLayer( const QString, const QString ) ) );

    //re-enable wfs with extent setting: pass canvas info to source select
    wfs->setProperty( "MapExtent", m_mapCanvas->extent().toString() );
    if ( m_mapCanvas->mapSettings().hasCrsTransformEnabled() )
    {
        //if "on the fly" reprojection is active, pass canvas CRS
        wfs->setProperty( "MapCRS", m_mapCanvas->mapSettings().destinationCrs().authid() );
    }

    bool bkRenderFlag = m_mapCanvas->renderFlag();
    m_mapCanvas->setRenderFlag( false );
    wfs->exec();
    m_mapCanvas->setRenderFlag( bkRenderFlag );
    delete wfs;
}
```

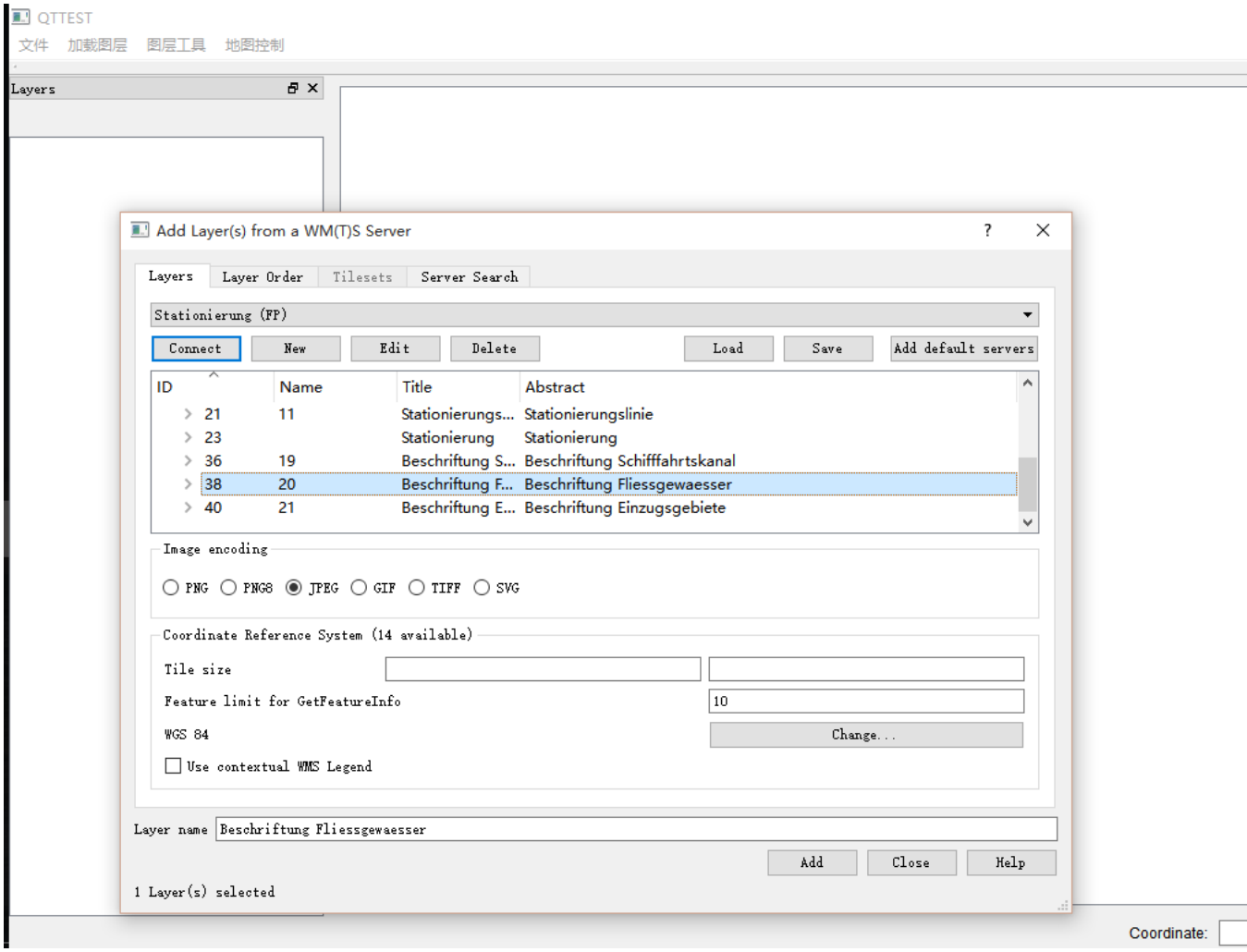
Need to add vector layer:

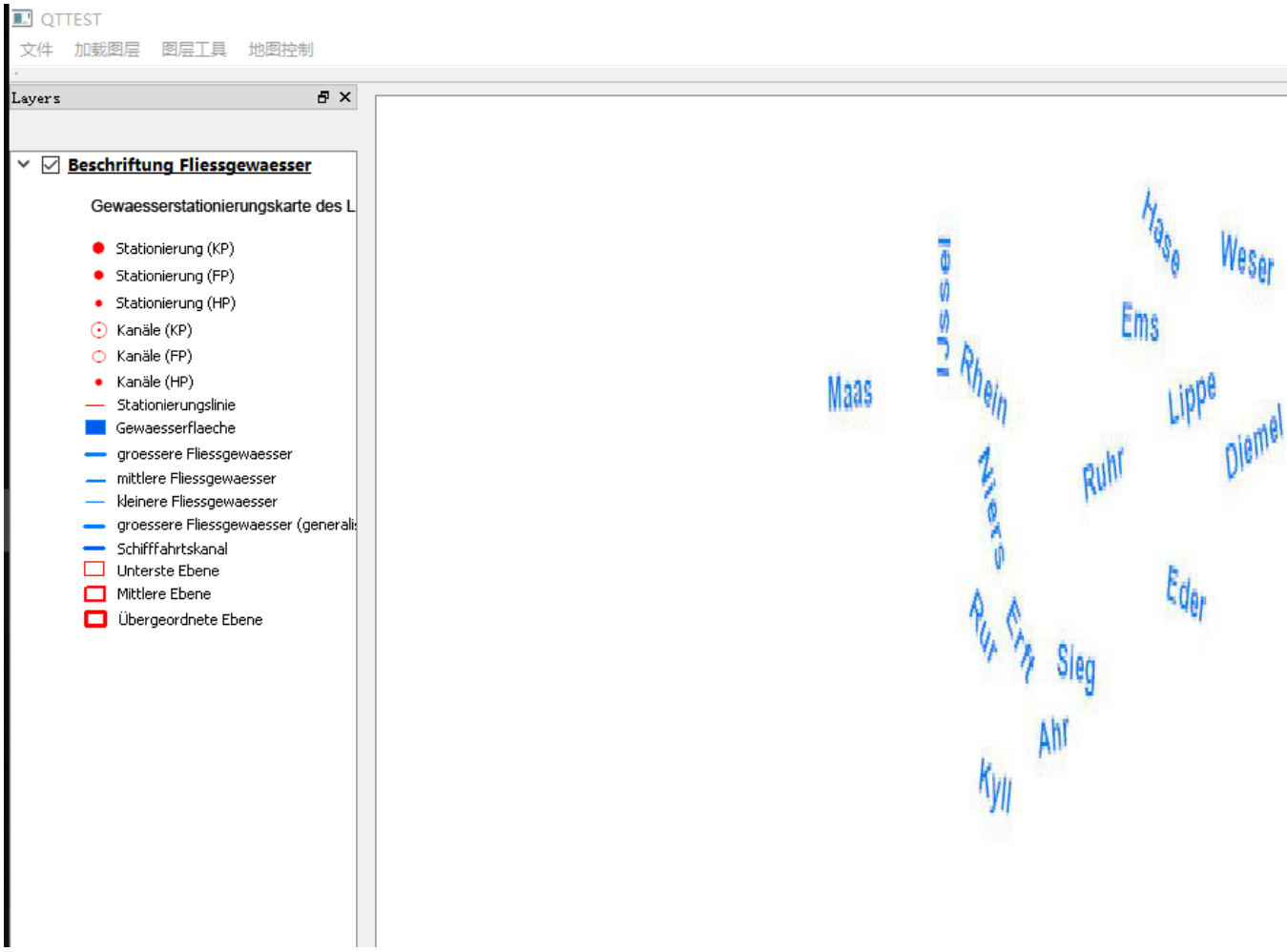
```
// 添加矢量图层
void QTTEST::addWFSLayer( const QString& url, const QString& typeName )
{
    QgsVectorLayer* vecLayer = new QgsVectorLayer( url, typeName, "WFS", false );
    if ( !vecLayer->isValid() )
    {
        QMessageBox::critical( this, "error", "layer is invalid" );
        return;
    }

    QgsMapLayerRegistry::instance()->addMapLayer( vecLayer );
    mapCanvasLayerSet.append( vecLayer );
    m_mapCanvas->setExtent( vecLayer->extent() );
    m_mapCanvas->setLayerSet( mapCanvasLayerSet );
    m_mapCanvas->setVisible( true );
    m_mapCanvas->freeze( false );
    m_mapCanvas->refresh();
}
void QTTEST::addOpenSourceRasterLayer( const QString& url, const QString& basename, const QString& providerKey )
{
    QgsRasterLayer *rasterLayer = 0;

    if ( providerKey.isEmpty() )
    {
        rasterLayer = new QgsRasterLayer( url, basename );
    }
    else
    {
        rasterLayer = new QgsRasterLayer( url, basename, providerKey );
    }

    if ( !rasterLayer->isValid() )
    {
        QMessageBox::critical( this, "error", "layer is invalid" );
        return;
    }

    QgsMapLayerRegistry::instance()->addMapLayer( rasterLayer );
    mapCanvasLayerSet.append( rasterLayer );
    m_mapCanvas->setExtent( rasterLayer->extent() );
    m_mapCanvas->setLayerSet( mapCanvasLayerSet );
    m_mapCanvas->setVisible( true );
    m_mapCanvas->freeze( false );
    m_mapCanvas->refresh();
}
```

Realization effect:

Take loading WMS layers as an example:


QTTEST
文件
加载图层
图层工具
地图控制
Add_Vector
Add_Raster
Add_WMS
Add_WCS
Add_WFS
delete


QTTEST
文件
加载图层
图层工具
地图控制
Layers
Add Layer(s) from a WM(T)S Server
Layers
Layer Order
Tilesets
Server Search
Stationierung (FP)
Connect
New
Edit
Delete
Load
Save
Add default servers
ID
Name
Title
Abstract
21
11
Stationierungs...
Stationierungslinie
23
Stationierung
Stationierung
36
19
Beschriftung S...
Beschriftung Schifffahrtskanal
38
20
Beschriftung F...
Beschriftung Fliessgewaesser
40
21
Beschriftung E...
Beschriftung Einzugsgebiete
Image encoding
PNG
PNG8
JPEG
GIF
TIFF
SVG
Coordinate Reference System (14 available)
Tile size
Feature limit for GetFeatureInfo
10
WGS 84
Change...
Use contextual WMS Legend
Layer name
Beschriftung Fliessgewaesser
Add
Close
Help
1 Layer(s) selected
Coordinate:


QTTEST
文件
加载图层
图层工具
地图控制
Layers
Beschriftung Fliessgewaesser
Gewaesserstationierungskarte des L
Stationierung (KP)
Stationierung (FP)
Stationierung (HP)
Kanäle (KP)
Kanäle (FP)
Kanäle (HP)
Stationierungslinie
Gewaesserflaeche
groessere Fliessgewaesser
mittlere Fliessgewaesser
kleinere Fliessgewaesser
groessere Fliessgewaesser (generali
Schifffahrtskanal
Unterste Ebene
Mittlere Ebene
Übergeordnete Ebene
Hase
Weser
Ems
Rhein
Maas
Lippe
Diemel
Ruhr
Eder
Sieg
Ahr
Kyll

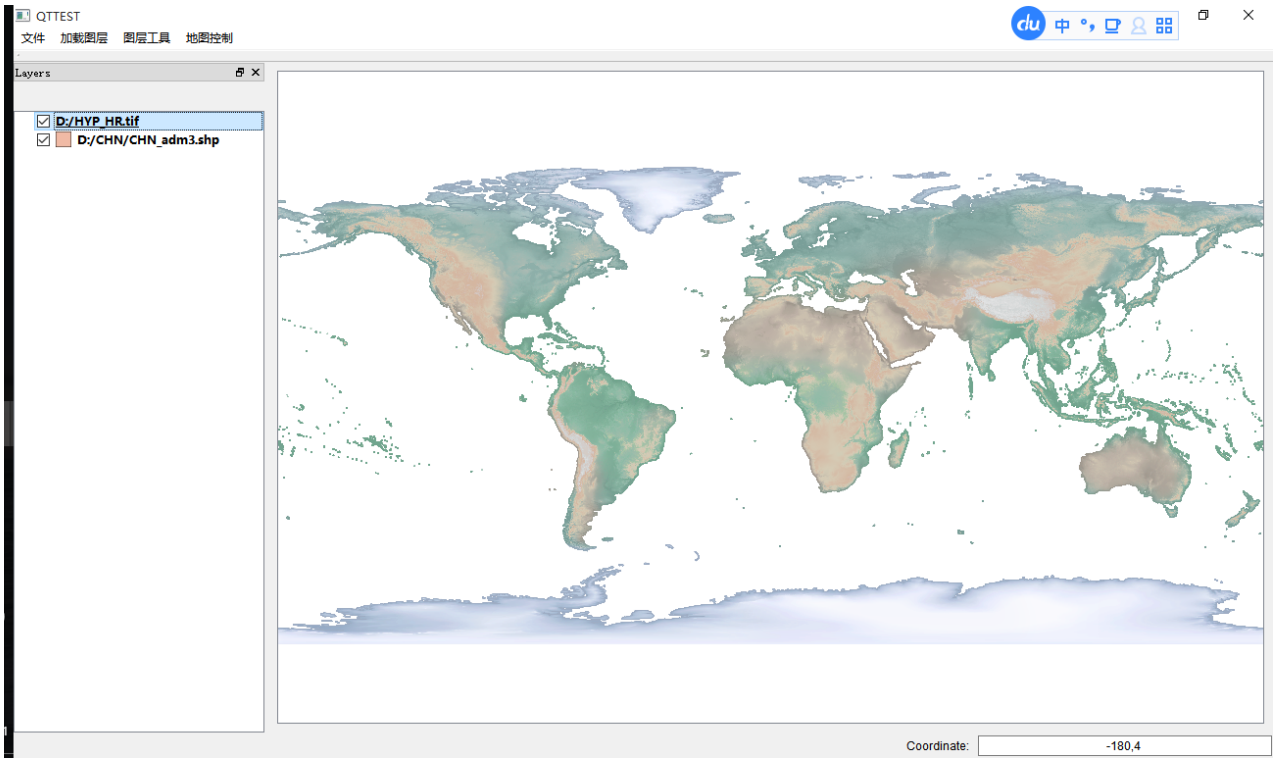
3.3.3 Delete layer

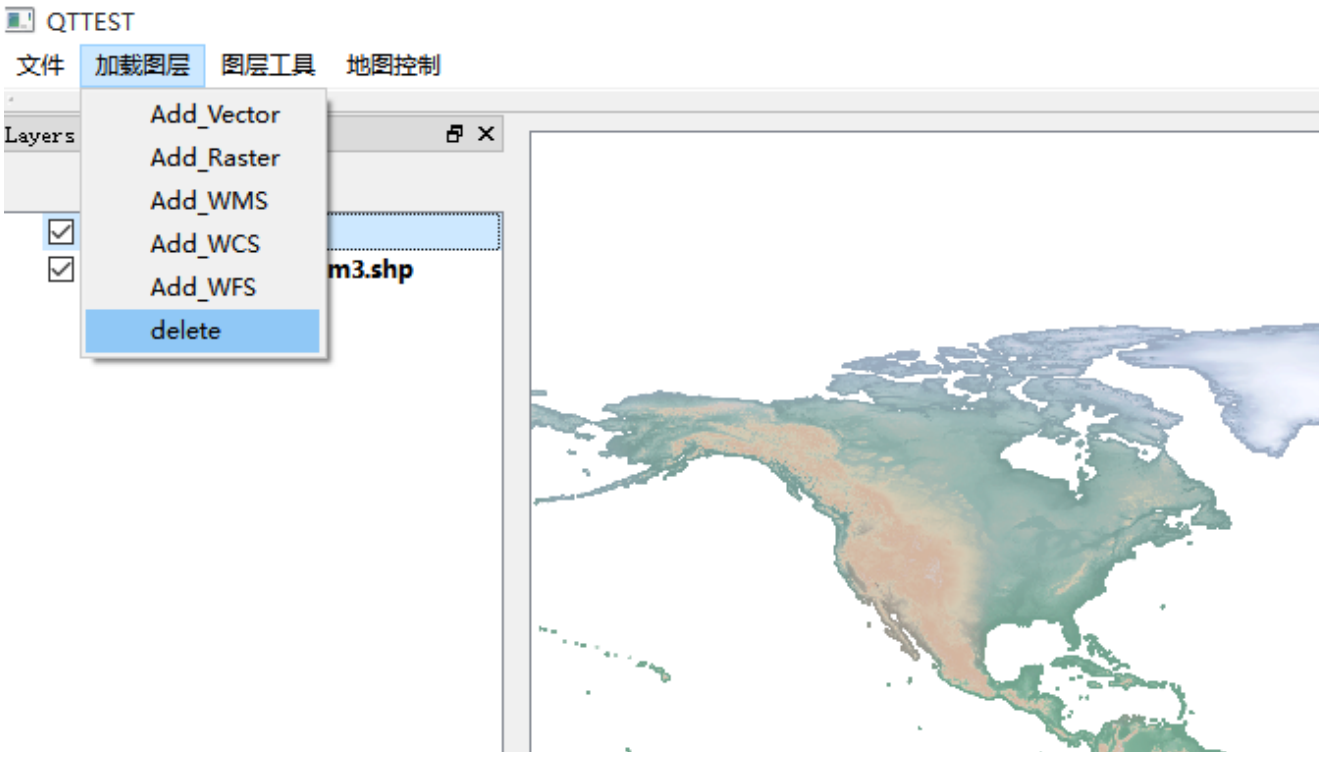
Implementation code:

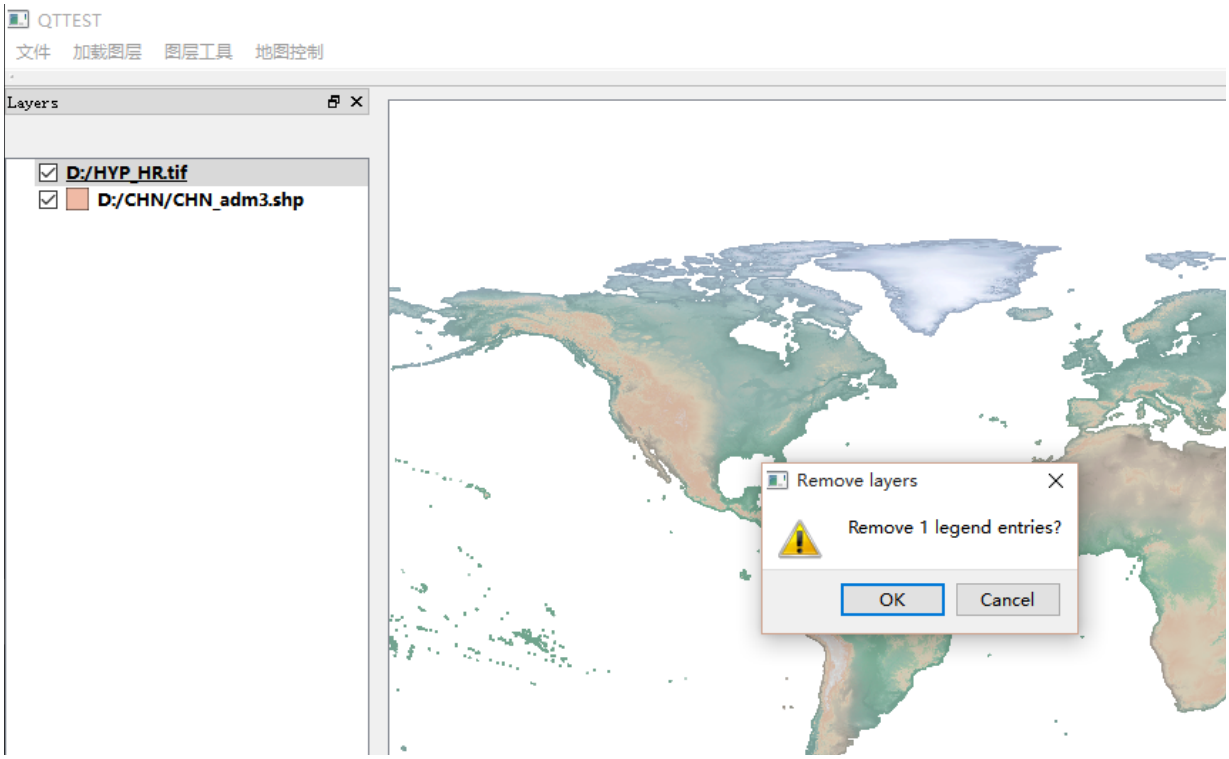
```
//*********************************************************
//删除图层
void QTTEST::removeoneLayer()
{
    QLabel*label=new QLabel("删除!");
    label->show();
    if ( !m_layerTreeView ) {return;}
    foreach( QgsMapLayer* layer, m_layerTreeView->selectedLayers() )
    {
        QgsVectorLayer* veclayer = qobject_cast<QgsVectorLayer*>( layer );
        if ( veclayer && veclayer->isEditable() ) {return;}
    }


    QList<QgsLayerTreeNode*> selectedNodes = m_layerTreeView->selectedNodes( false );

    // validate selection
    if ( selectedNodes.isEmpty() )
    {
        QMessageBox::critical(
            this,
            tr( "Error" ),
            tr( "No selection valid" ) );
        return;
    }
    bool promptConfirmation = QSettings().value( "qgis/askToDeleteLayers", true ).toBool();
    //display a warning
    if ( promptConfirmation && QMessageBox::warning( this, tr( "Remove layers" ), tr( "Remove %n legend entries?", "number of legend i
    {
        return;
    }
    foreach ( QgsLayerTreeNode* node, selectedNodes )
    {
        QgsLayerTreeGroup* parentGroup = qobject_cast<QgsLayerTreeGroup*>( node->parent() );
        if ( parentGroup )
        {
            parentGroup->removeChildNode( node );
        }
    }
    m_mapCanvas->refresh();
}
```

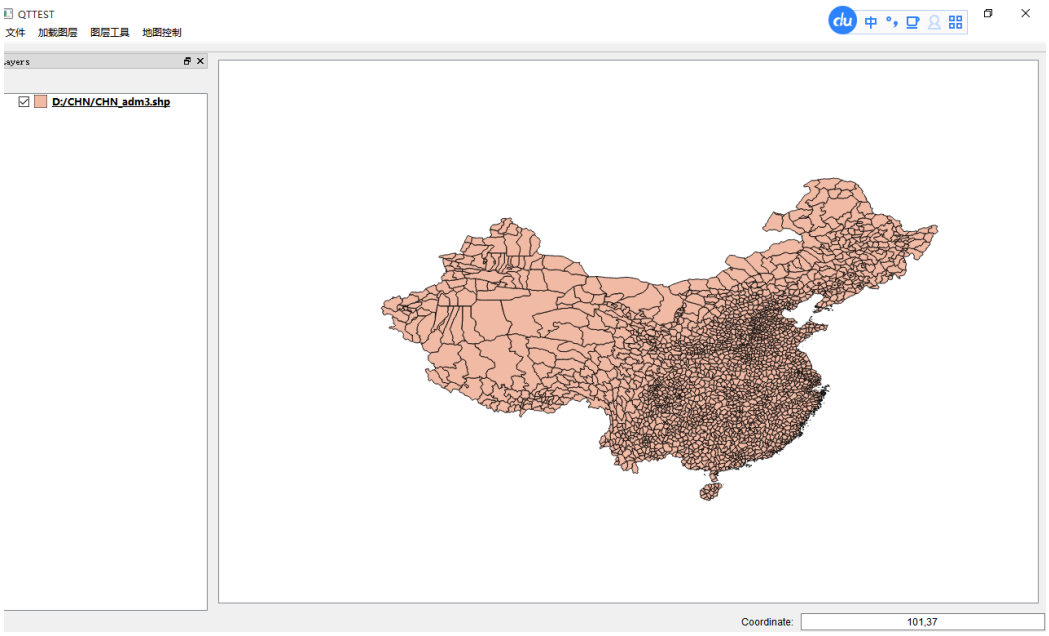
Realization effect:

Select the layers you want to delete:






QTTEST
文件 加载图层 图层工具 地图控制
Layers
D:/HYP_HR.tif
D:/CHN/CHN_adm3.shp
Remove layers
Remove 1 legend entries?
OK
Cancel


QTTEST
文件 加载图层 图层工具 地图控制
D:/CHN/CHN_adm3.shp
Coordinate:
101,37

3.3.4 Layer annotation

Implementation code:

```
//*****************************************************************
//图层标注
void QTTEST::layerSymbolTest()
{
    // 获取当前选中的图层
    QgsVectorLayer* veclayer = qobject_cast<QgsVectorLayer*>( this->activeLayer() );
    if( !veclayer->isValid() ) { return; }

    if ( veclayer->geometryType() == QGis::Point )
    {
        // 创建 svgMarkerSymbolLayer
        QgsSvgMarkerSymbolLayerV2* svgMarker = new QgsSvgMarkerSymbolLayerV2( "money/money_b

        QgsSymbolLayerV2List symList;
        symList.append( svgMarker );

        QgsMarkerSymbolV2* markSym = new QgsMarkerSymbolV2( symList );

        QgsSingleSymbolRendererV2* symRenderer = new QgsSingleSymbolRendererV2( markSym );

        svgMarker->setSize( 10 );
        veclayer->setRendererV2( symRenderer );
    }
```

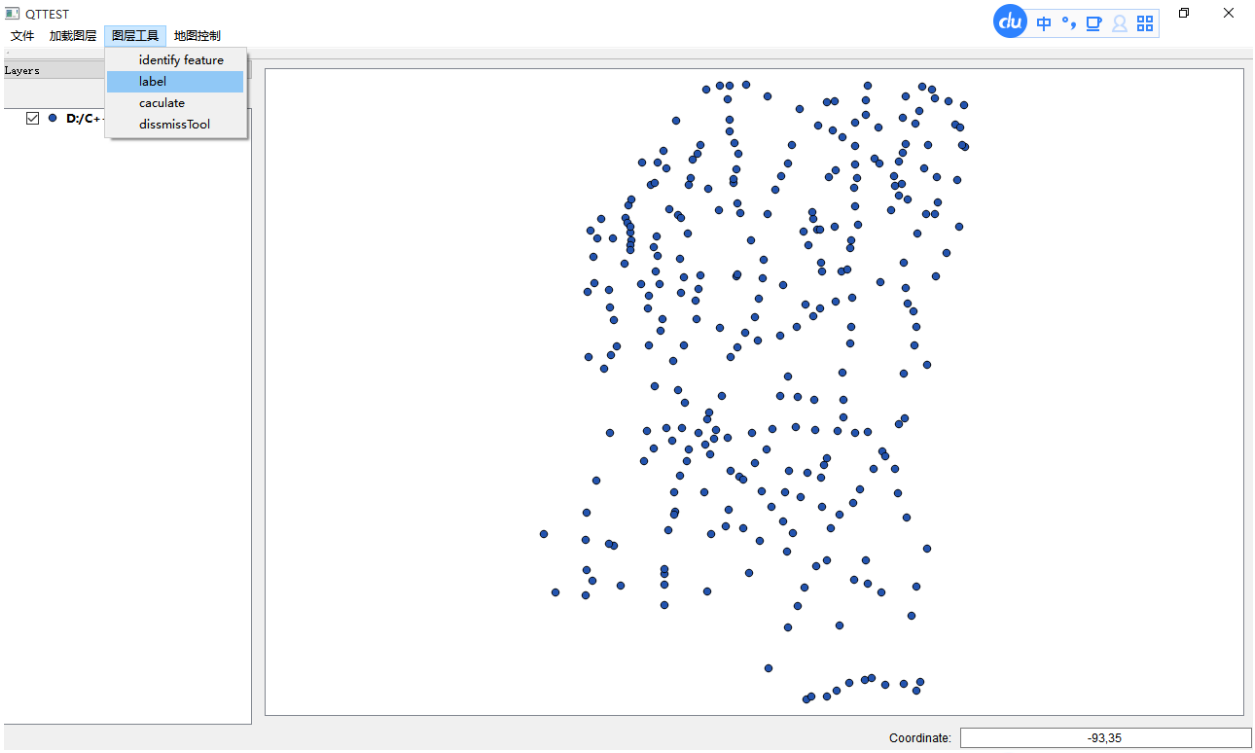
Here you need to get the currently selected layer:

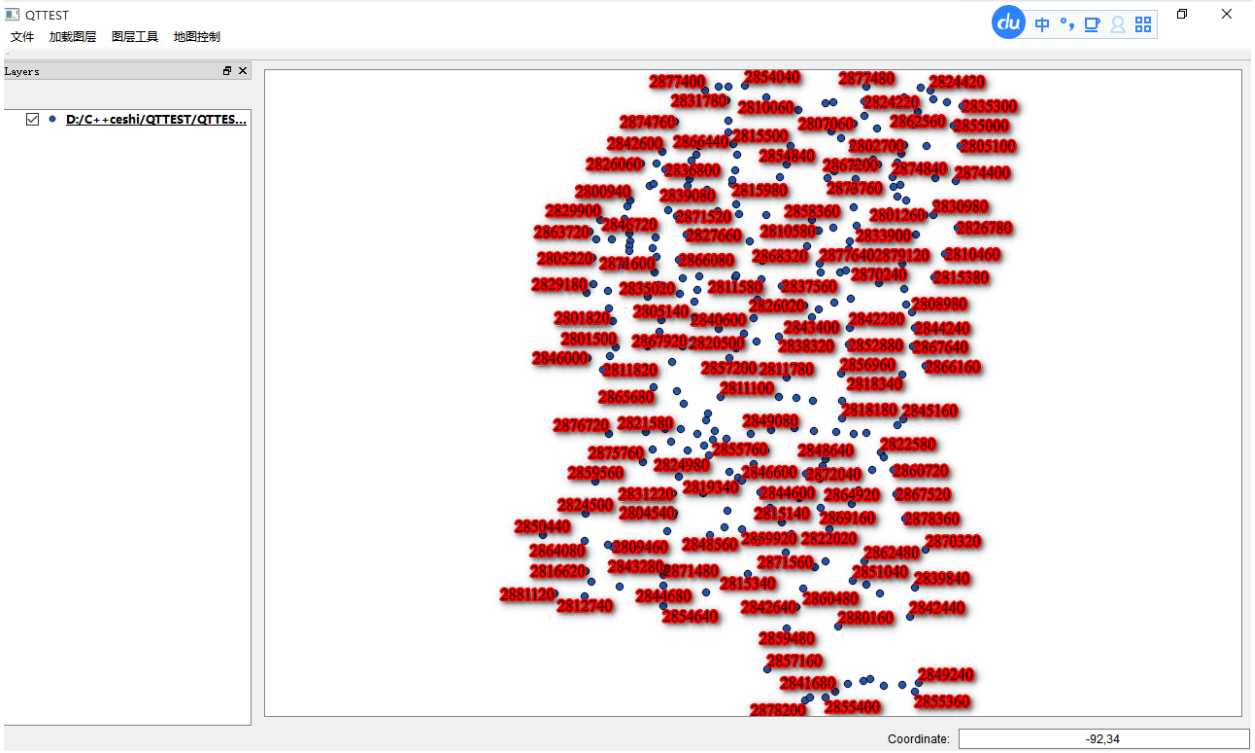
```
//获取当前选中的图层
QgsMapLayer* QTTEST::activeLayer()
{
    return m_layerTreeView ? m_layerTreeView->currentLayer() : 0;
}
```

Configure label properties:

```
//*************************************************************
//图层标注
void QTTEST::layerSymbolTest() { ... }
void QTTEST::testVecLayerLabel()
{
    QgsVectorLayer* layer = ( QgsVectorLayer* )this->activeLayer();
    if ( layer == NULL || layer->isValid() == false ) { return; }

    // 首先是定义一个 QgsPalLayerSettings 变量，并启用他的属性设置
    QgsPalLayerSettings layerSettings;
    layerSettings.enabled = true;

    // 然后就可以开始根据API文档中的属性，进行自定义配置了
    layerSettings.fieldName = layer->pendingFields()[3].name(); // 设置Label图层
    layerSettings.centroidWhole = true; // 设置位置参考的中心点

    // Label 字体设置
    layerSettings.textColor = QColor( 0, 0, 0 ); // 设置字体颜色
    layerSettings.textFont = QFont( "Times", 12 ); // 设置字体和大小

    // Label 轮廓buffer设置
    layerSettings.bufferDraw = true;
    layerSettings.bufferColor = QColor( 255, 0, 0 ); // 轮廓buffer的颜色
    layerSettings.bufferSize = 1; // 轮廓buffer大小
    layerSettings.bufferTransp = 0.5; // 轮廓buffer的透明度

    // Label 阴影绘制
    layerSettings.shadowDraw = true;
    layerSettings.shadowOffsetAngle = 135; // 阴影的角度
    layerSettings.shadowOffsetDist = 1; // 阴影与Label的距离

    layerSettings.fieldName = layer->pendingFields()[3].name(); // 设置Label图层
    layerSettings.setDataDefinedProperty( layerSettings.Size, true, false, NULL, "size" );
    layerSettings.setDataDefinedProperty( layerSettings.Color, true, false, NULL, "color" );
    layerSettings.setDataDefinedProperty( layerSettings.Family, true, false, NULL, "font" );

    layerSettings.writeToLayer( layer ); // 将配置写入图层
    m_mapCanvas->refresh();
}
```

Realization effect:





Additional features:

In the implementation of layer tools, there is a problem: after I select a tool, how to cancel it?

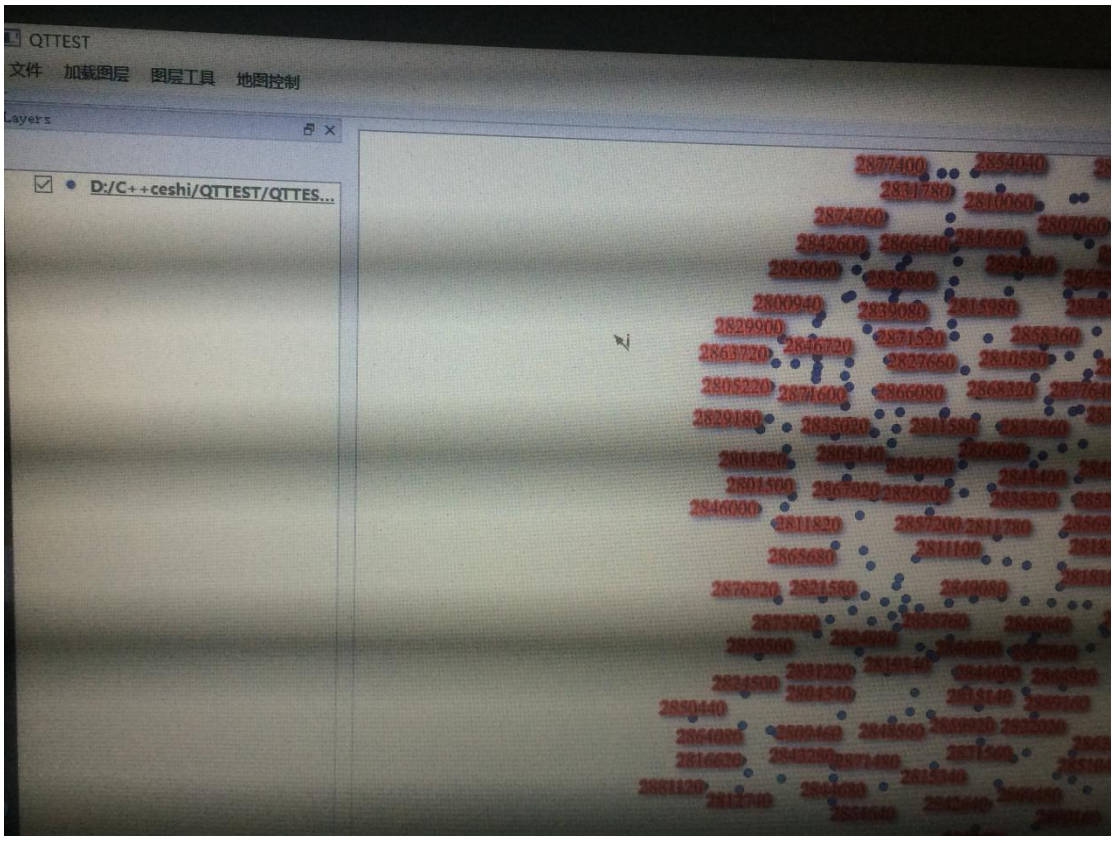
What about the use of tools? Therefore, it is necessary to implement the tool destruction function, that is, to restore the properties of the mouse from tool to normal.

Pass mouse.

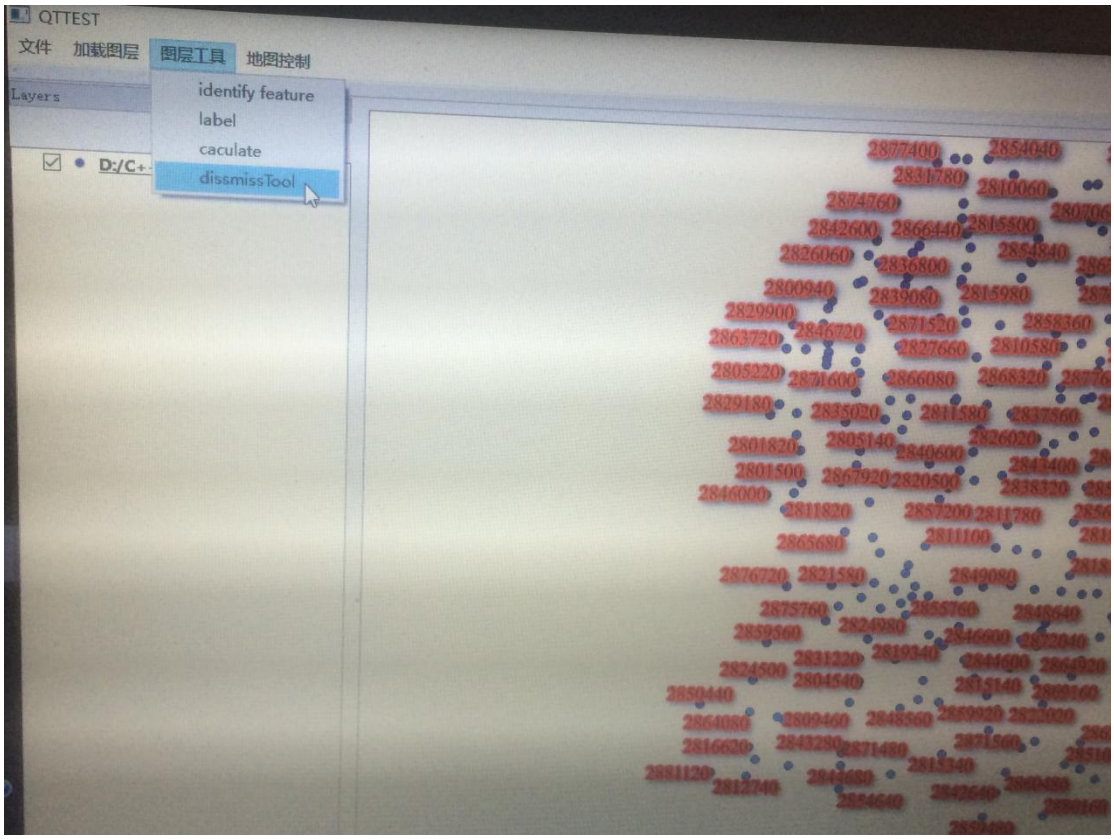
Implementation code:

```
//*****************************************************************
//取消地图工具
void QTTEST::disstool(){
    QgsMapTool *lastMapTool = m_mapCanvas->mapTool();
    m_mapCanvas->unsetMapTool( lastMapTool );
}
```

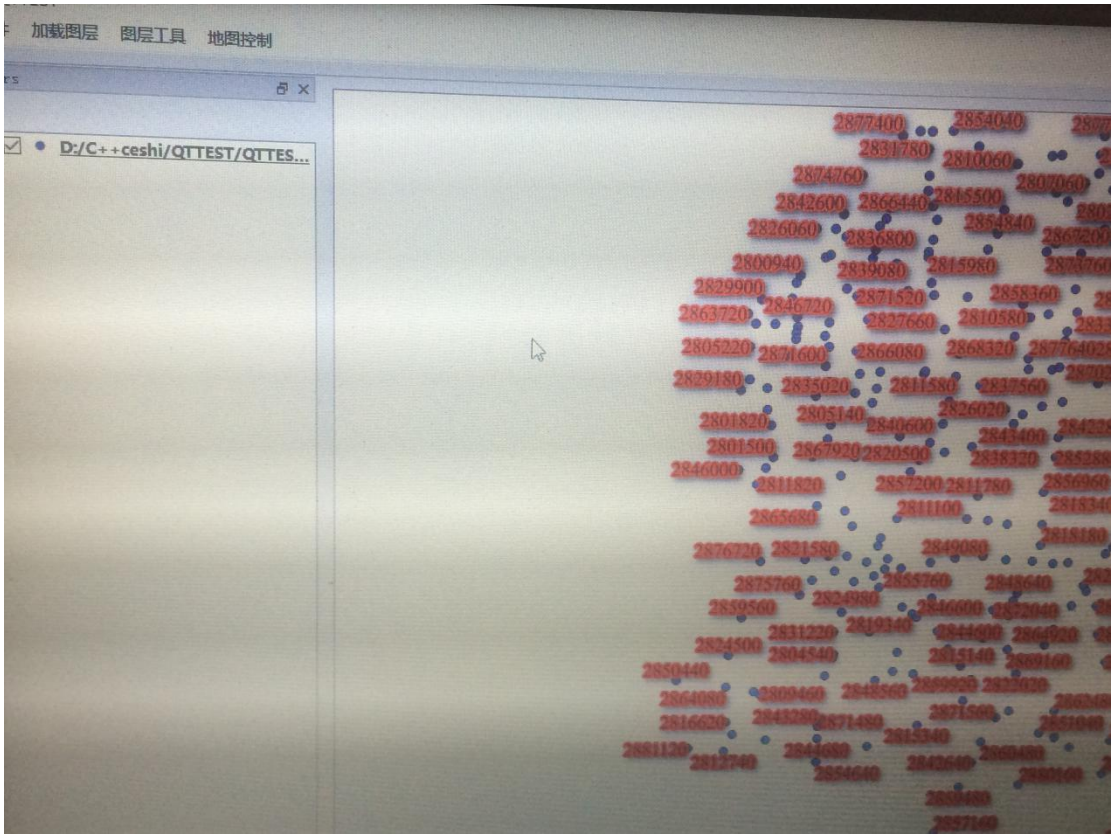
Realization effect:



The current mouse is in the primitive recognition state, click the Cancel tool.



The mouse returns to normal state



! ! ! All the above functions are called through the connect function response button:

```
    // connections响应
    connect( ui.actionAdd_Vector, SIGNAL( triggered() ), this, SLOT( addVectorLayers() ) );
    connect( ui.actionAdd_Raster, SIGNAL( triggered() ), this, SLOT( addRasterLayers() ) );
    connect( ui.action_2, SIGNAL( triggered() ), this, SLOT( openfile() ) );//打开项目
    connect( ui.action_4, SIGNAL( triggered() ), this, SLOT( savefile() ) );//保存项目
    connect( ui.action, SIGNAL( triggered() ), this, SLOT( newfile() ) );//新建项目
    connect( ui.action_GIS, SIGNAL( triggered() ), this, SLOT( closefile() ) );//关闭项目
    connect( ui.actionAdd_WMS,SIGNAL( triggered() ), this, SLOT( addWMSLayers() ) );//加载在线图层WMS
    connect( ui.actionAdd_WFS,SIGNAL( triggered() ), this, SLOT( addWFSLayers() ) );//WFS
    connect( ui.actionAdd_WCS,SIGNAL( triggered() ), this, SLOT( addWCSLayers() ) );//WCS
    connect( ui.actionIdentify_feature,SIGNAL( triggered() ), this, SLOT( qIdentify() ));//图元识别
    connect( ui.actionDelete,SIGNAL( triggered() ), this, SLOT(removeoneLayer()));//删除图层
    connect( ui.actionLabel,SIGNAL( triggered() ), this, SLOT(testVecLayerLabel()));//图层标注
    connect( ui.actionDissmissTool,SIGNAL( triggered() ), this, SLOT(disstool()));//图层标注
    connect( ui.actionL,SIGNAL( triggered() ), this, SLOT(Large()));//放大
    connect( ui.actionSmaller,SIGNAL( triggered() ), this, SLOT(Small()));//缩小
    connect( ui.actionMove,SIGNAL( triggered() ), this, SLOT(Move()));//平移
    connect( m_mapCanvas, SIGNAL( xyCoordinates( const QgsPoint& ) ), this, SLOT( showMouseCoordinate( cc
}
```