Visual Studio 2022 + Qt 简易教程

本文档简要介绍Visual Studio 2022环境下，如何使用Qt进行GUI开发。整个开发以Visual Studio为主，Qt只提供GUI。

0. Visual Studio 2022的安装

按照官方教程，选择C++的桌面开发。

1. Qt下载

- 在Open Source Development，选择Download the Qt Online Installer，选择对应版本，本文示例为Windows安装程序4.5.2版。

2. Qt安装

2.1 注册/登录Qt账号

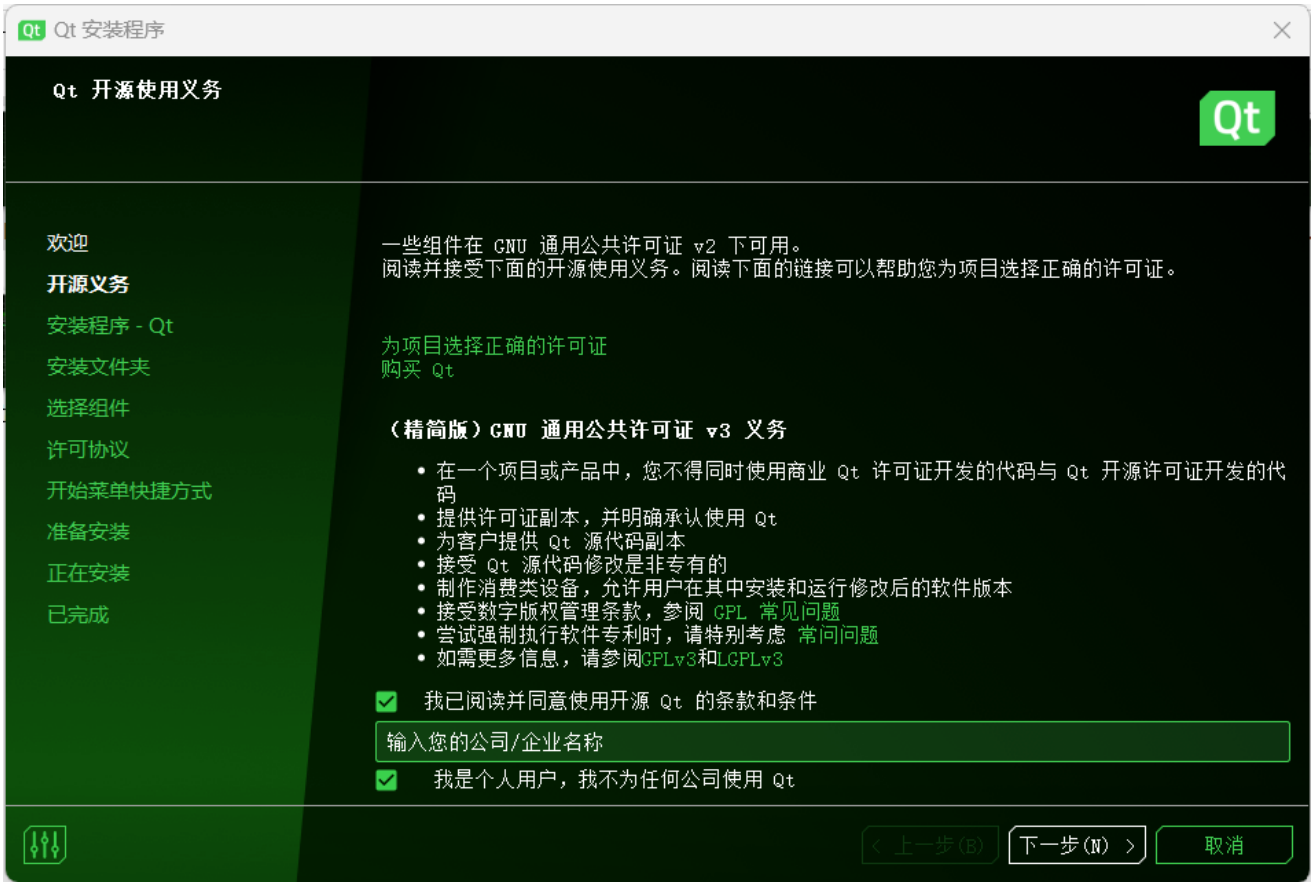
注册：在填写完sign-up之后会给你的邮箱发送一份邮件，需要点开邮件里面的链接进行确认。

登录：如下图

- 

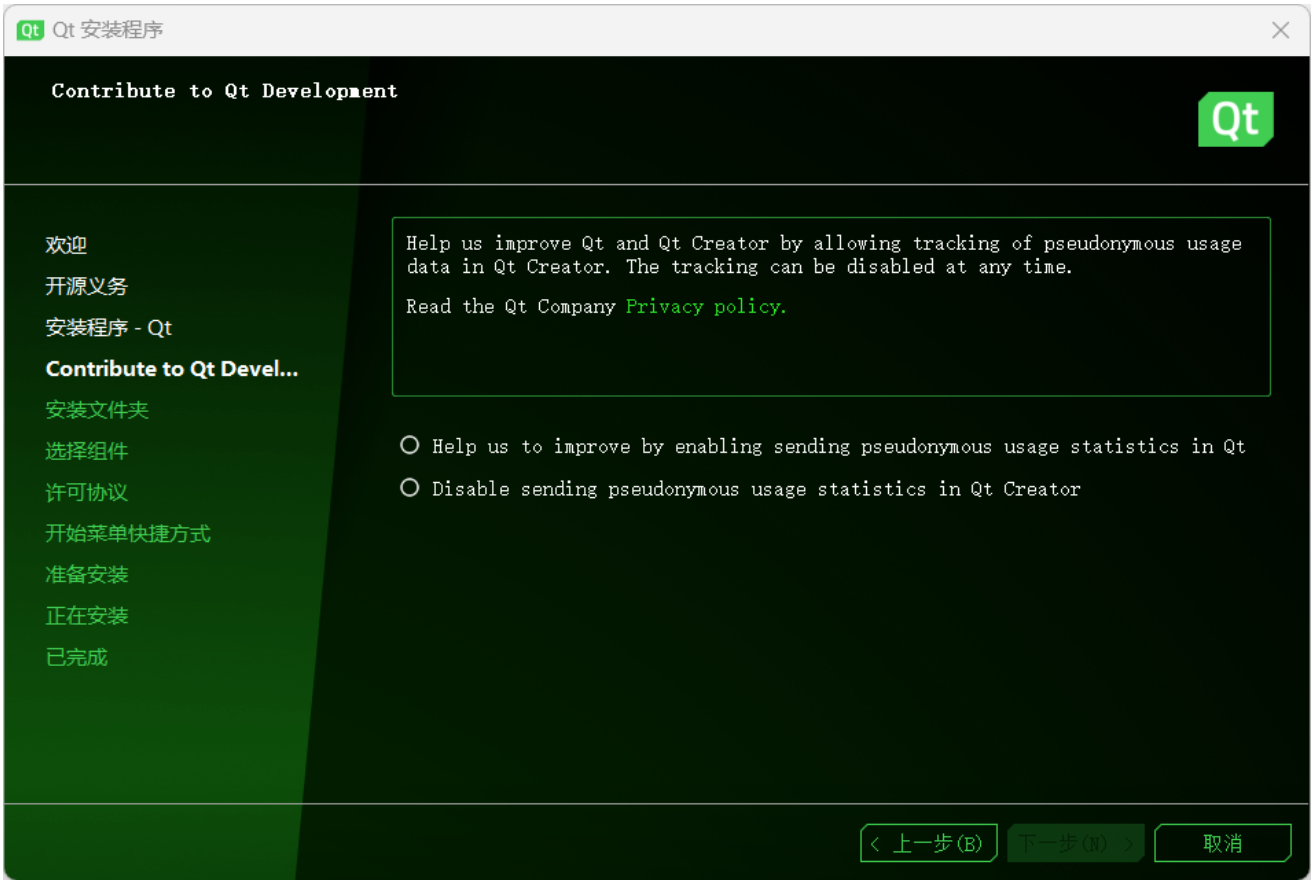
2.2 安装

- 开源使用义务

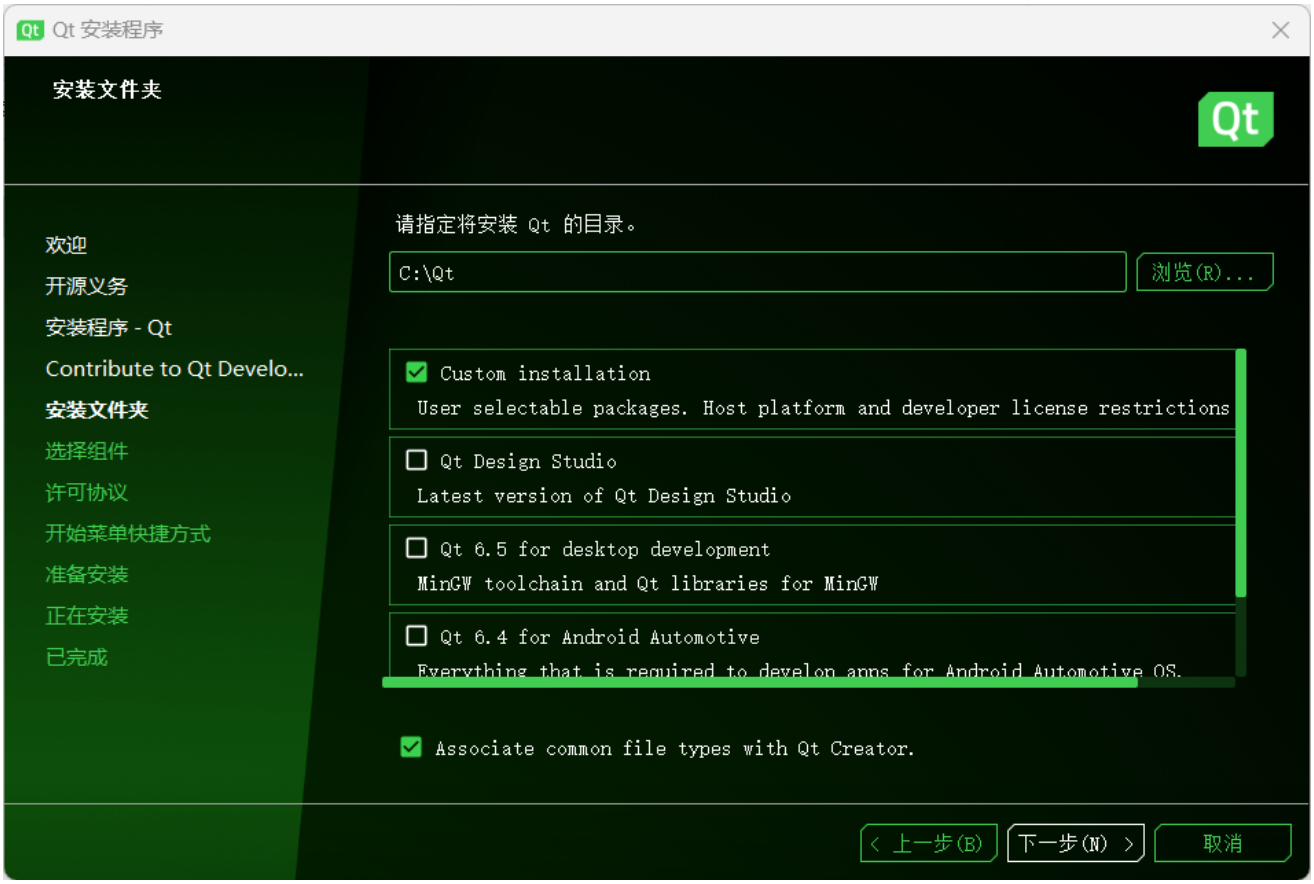


- 安装程序

- Contribute to Qt Development



- 选择安装路径



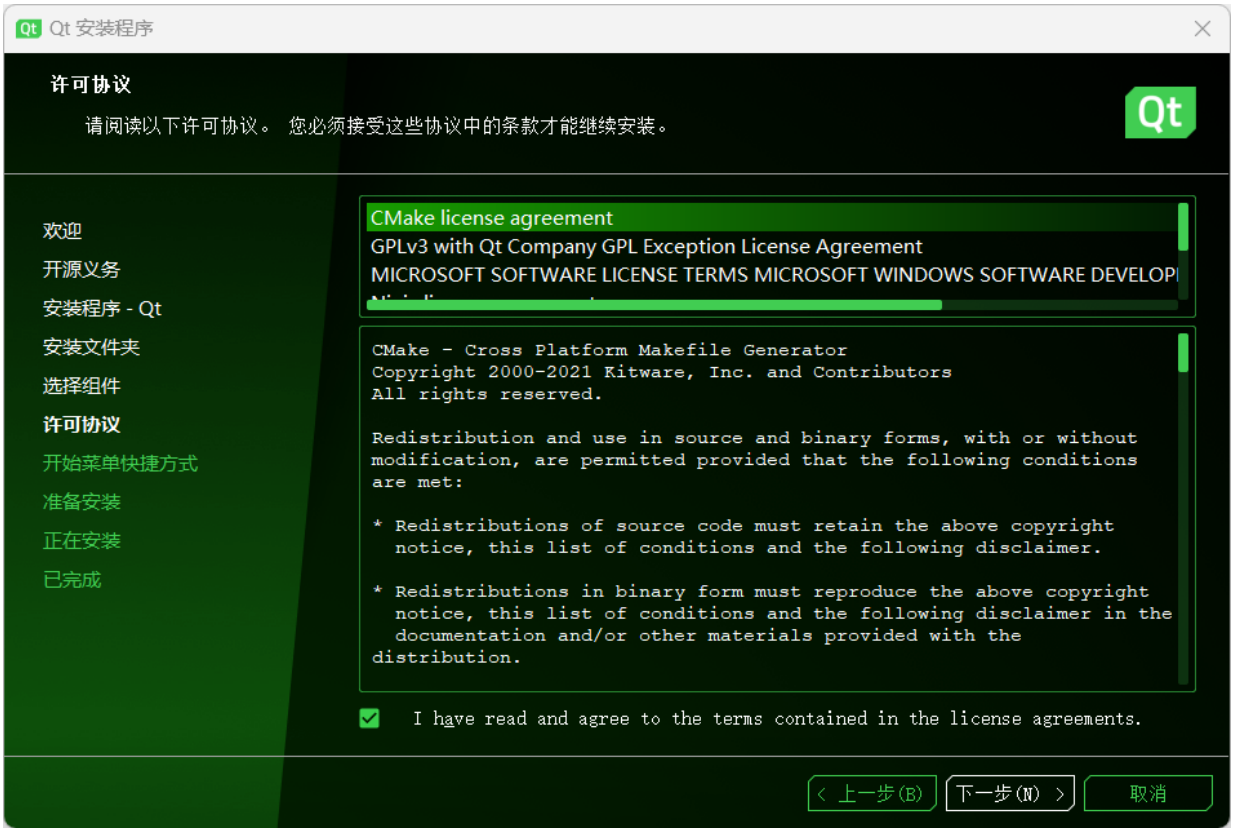
- 选择安装组件

选择合适的版本，此处选择LTS版本，Qt6.2.4，Qt5.15.2【选择一个版本即可】

在Additional Libraries里选择相应的库，如

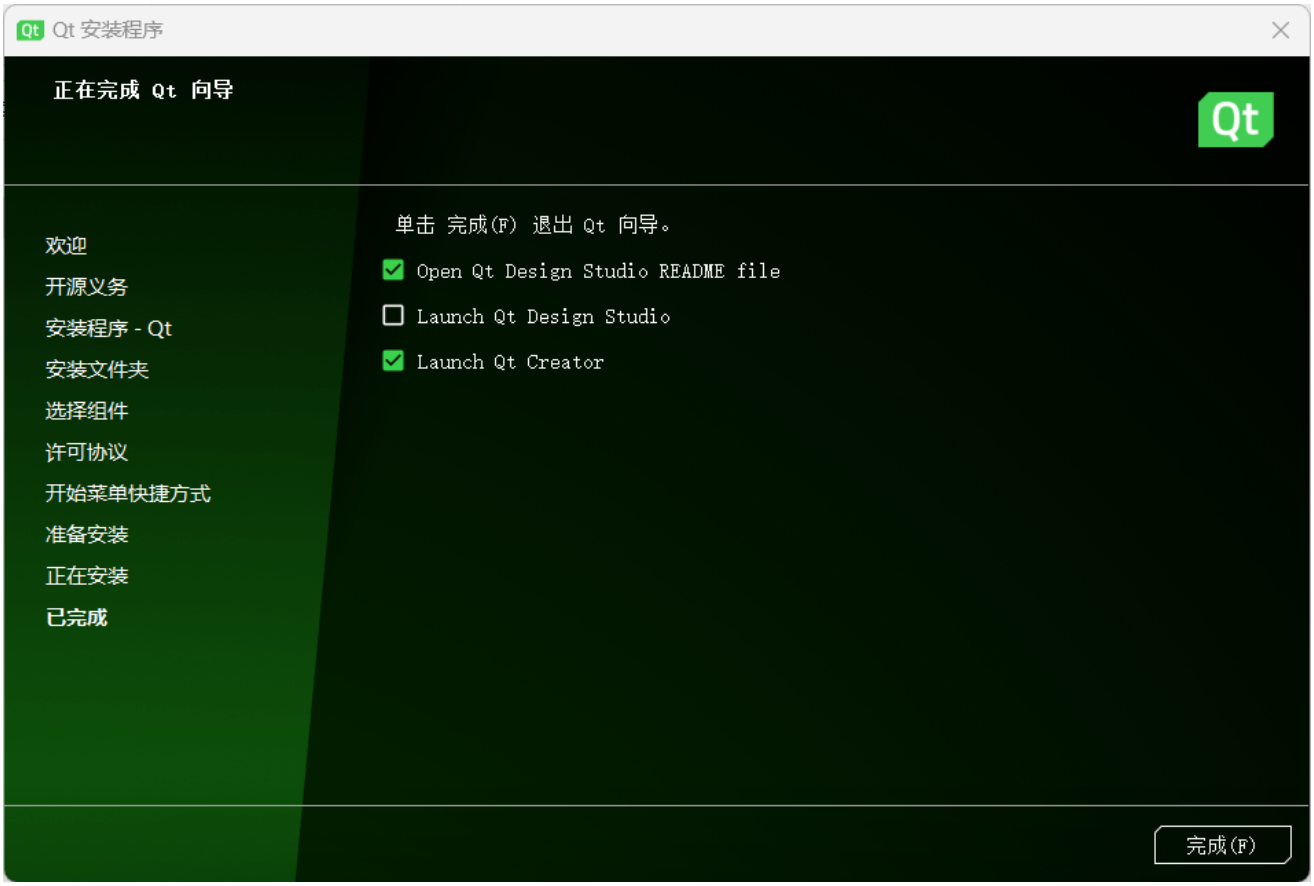
- Qt Multimedia（增加对音频视频等多媒体属性支持）
- Qt Data Visualization（数据可视化）

- 许可协议



- 开始菜单快捷方式-准备安装-正在安装

- 完成



2.3 更新、添加或者移除组件

MaintenanceTool本身可能存在版本更新。

以及通过Additional Libraries添填组件。

2.4 常见问题

由于网络等原因，安装过程可能不能一次完成。需要更换网络连接多次重试。

3. Qt离线版安装

根据Qt的相关协议， Qt 5.15开始，不再提供免费离线安装包。因此，免费安装离线包的最后一个版本是5.14.2，windows版本

4. Visual Studio 2022配置

4.1 Qt Visual Studio Tools插件

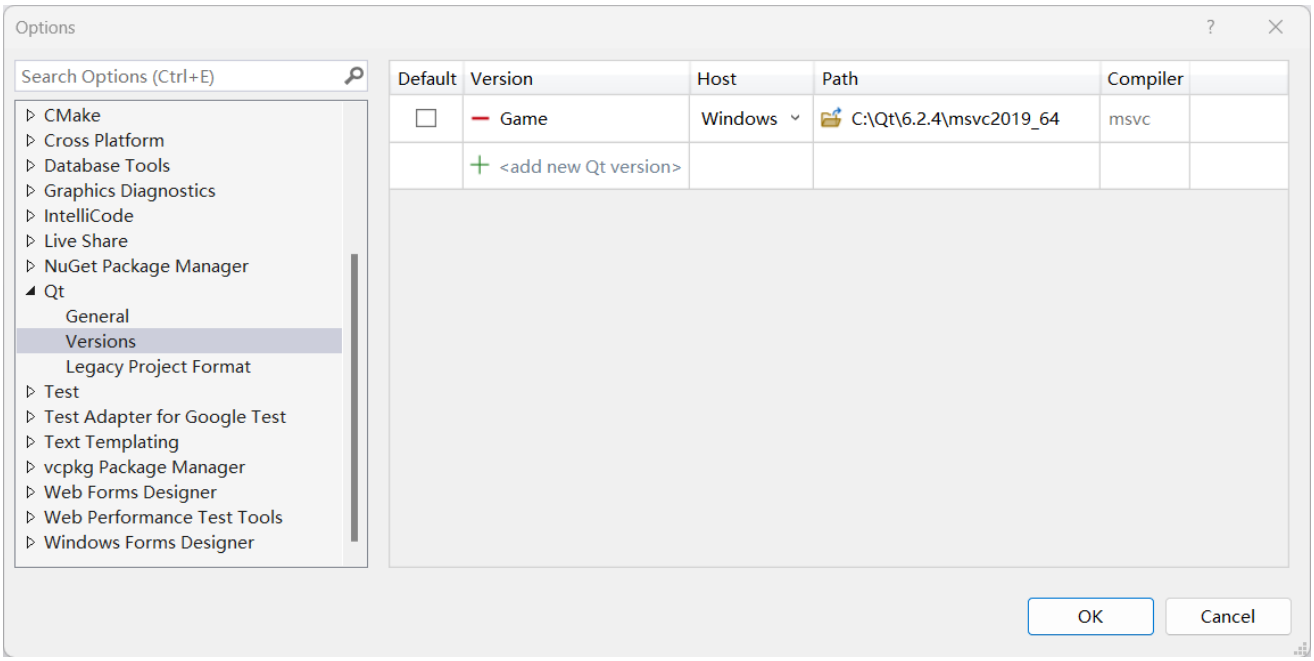
- 菜单操作：Extensions - Manage Extensions
  - Online中Serach: qt 选择Qt Visual Studio Tools下载，关闭Visual Studio，更新插件（退出Visual Studio弹出VSIX Installer对话框，点击Modify）；
- 或者在https://marketplace.visualstudio.com/vs中搜索Qt Visual Studio Tools，下载对应Visual Studio的版本，如，https://marketplace.visualstudio.com/items?itemName=TheQtCompany.QtVisualStudioTools2022，下载vsix文件，直接运行安装即可。

- 如果Qt Visual Studio Tools安装有错误（如界面停止程序死机状态），可以试着安装其它的Extension，然后再Qt Visual Studio Tools。

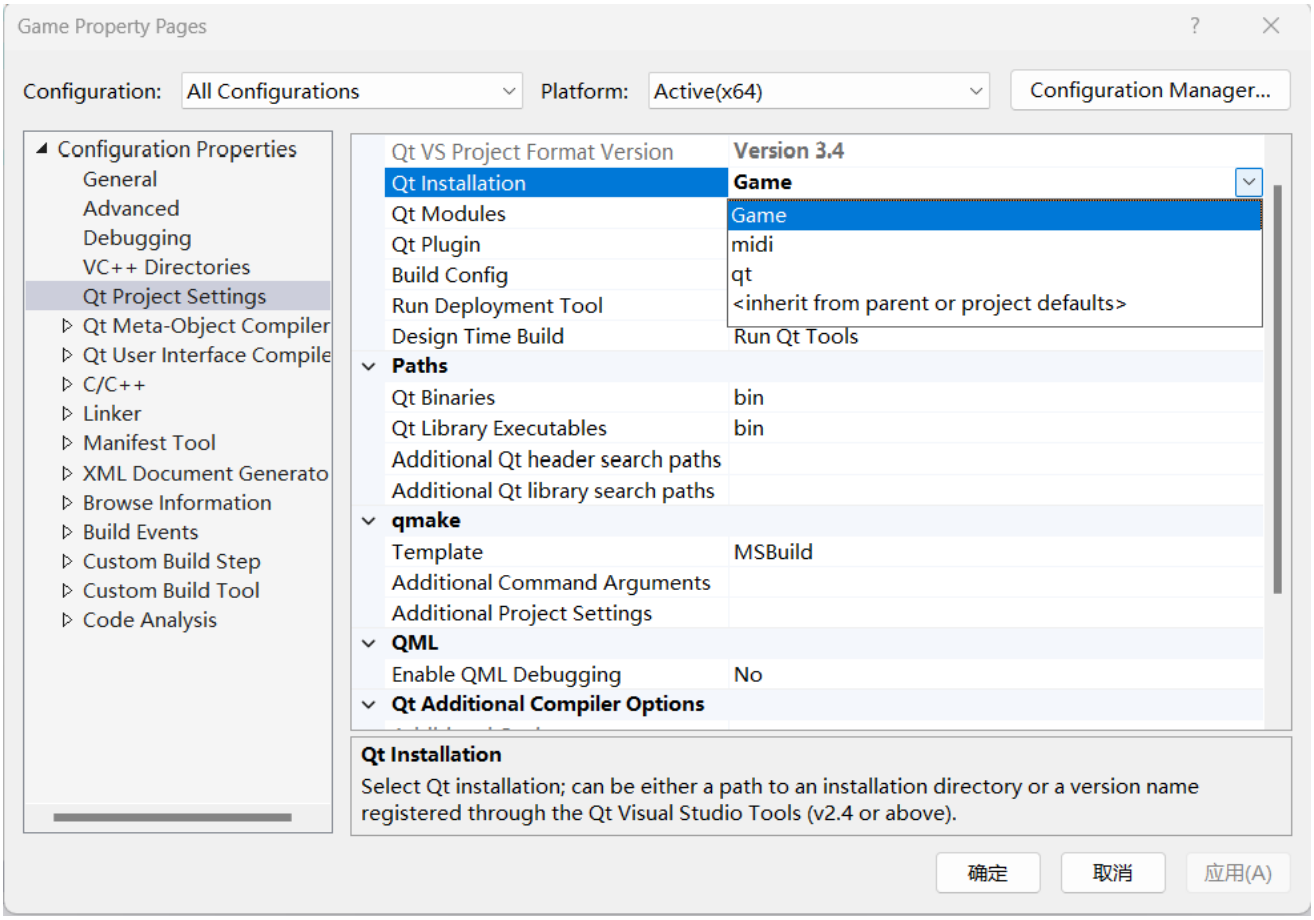
4.2 Qt环境配置（Qt VS Tools）

设置Qt环境，每个环境通过Version标记。具体的Qt项目，通过Qt Installation选择对应Qt环境

- Extensions - Qt VS Tools - Qt Versions
  - 增加Qt配置
    - Path 选择 qmake.exe文件，此处选择C:\Qt\6.2.4\msvc2019_64
    - Version命名，此处命名为Game



如果某个qt工程需要选择qt配置。Project - Properties - Qt Project Settings - Qt Installation中选择相应的qt配置（此处为Game）



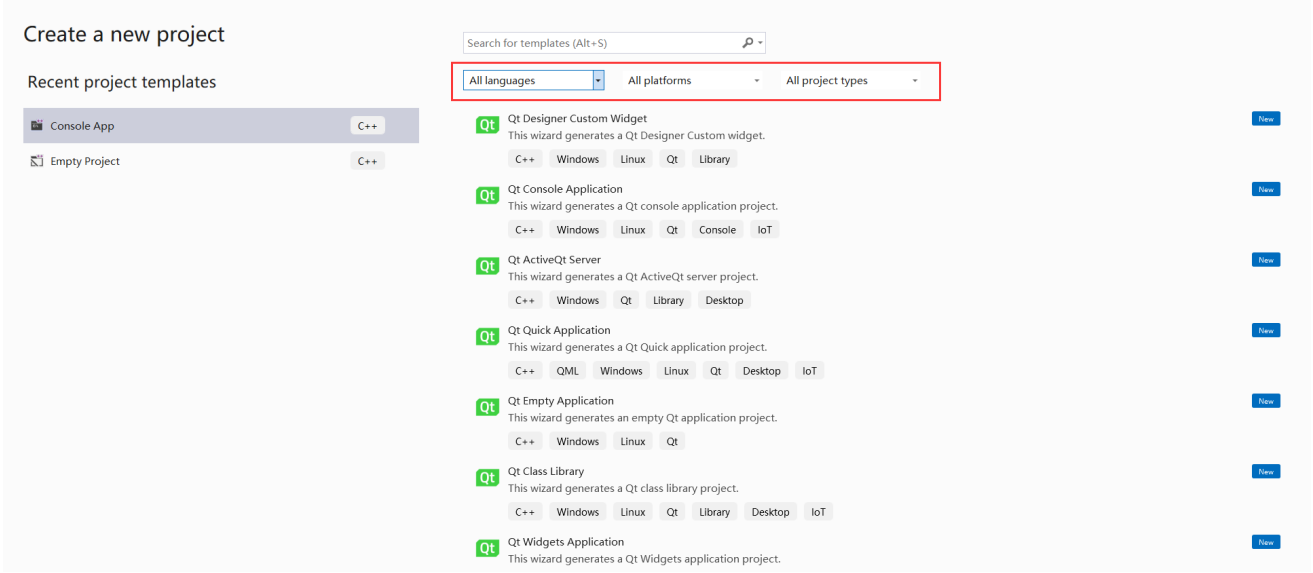
4.3 VS打开qt工程

Extensions - Qt VS Tools - Open Qt Project File (.pro)

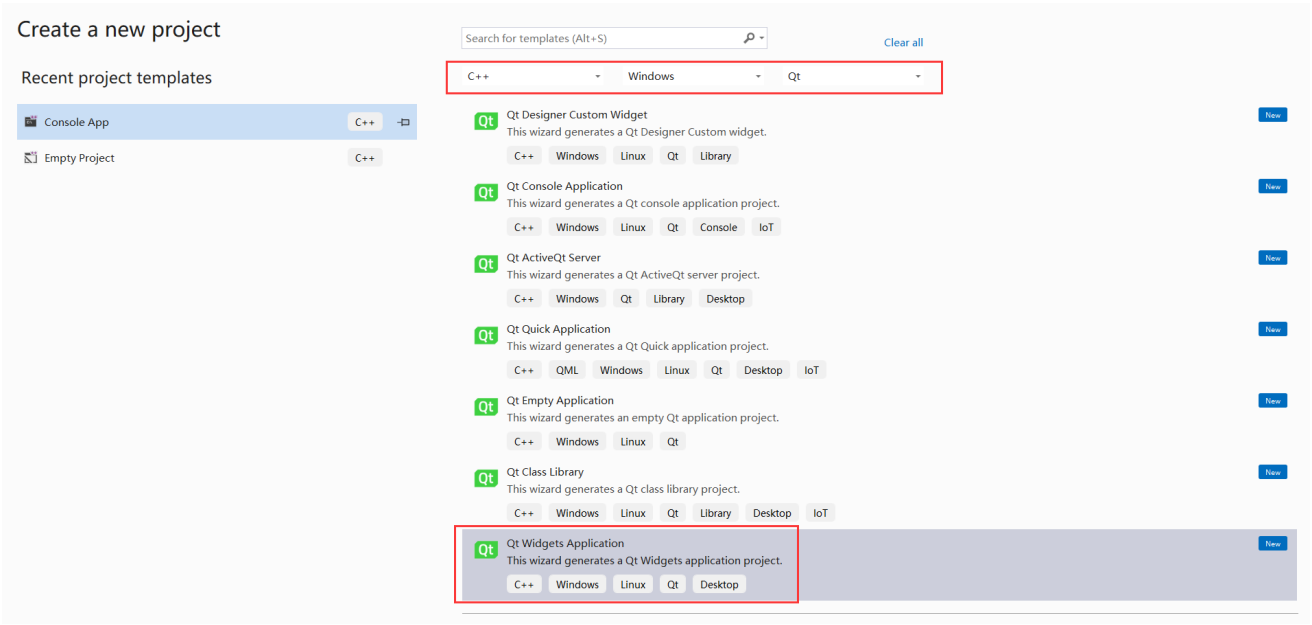
5. 第一个Visual Studio 2022 Qt程序

5.1 创建工程

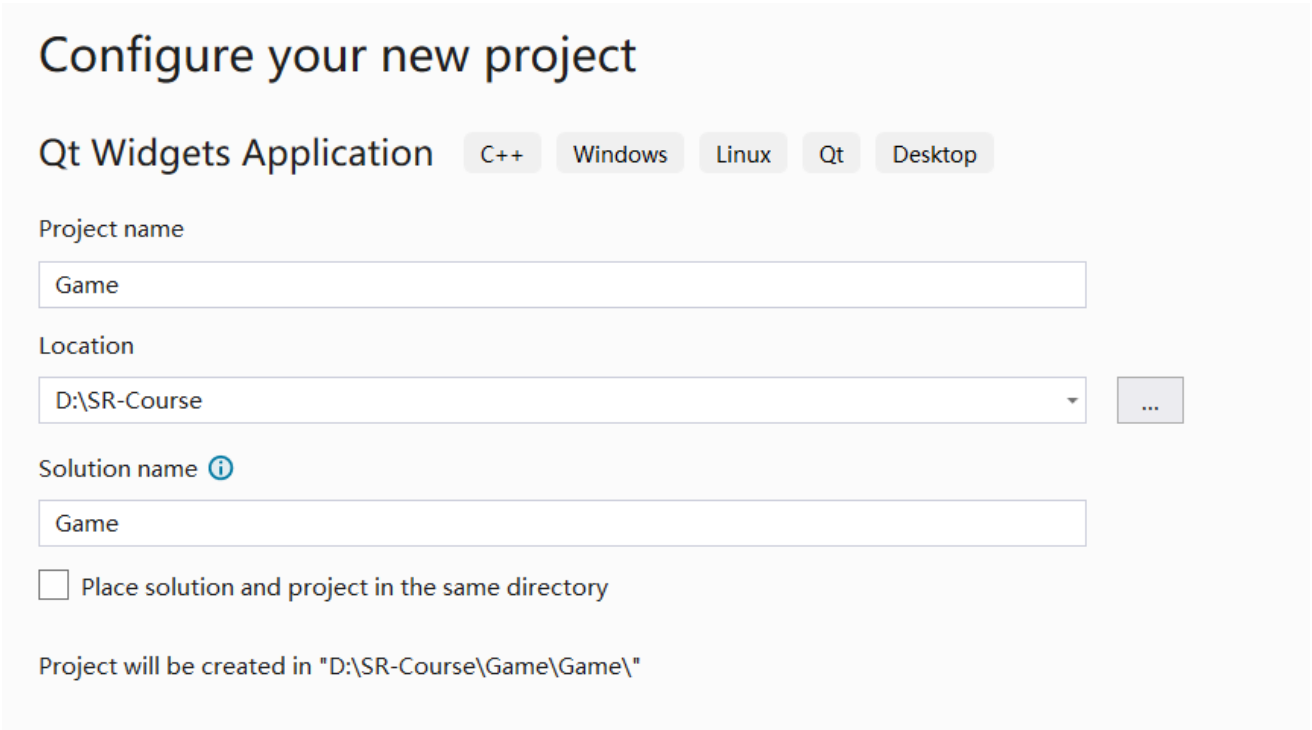
打开Visual Studio 2022，Create a new project，选择

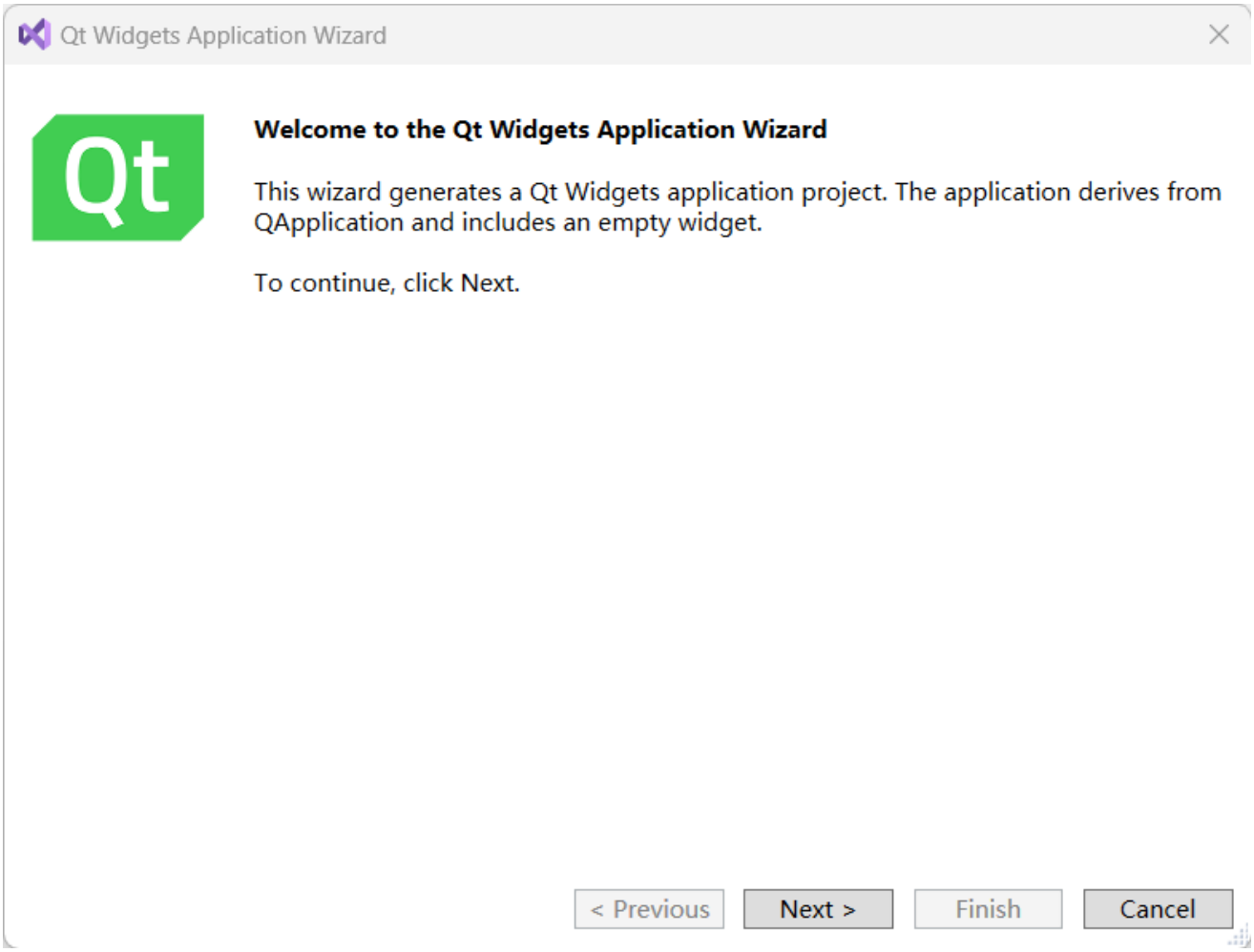


生成Qt Widgets Application

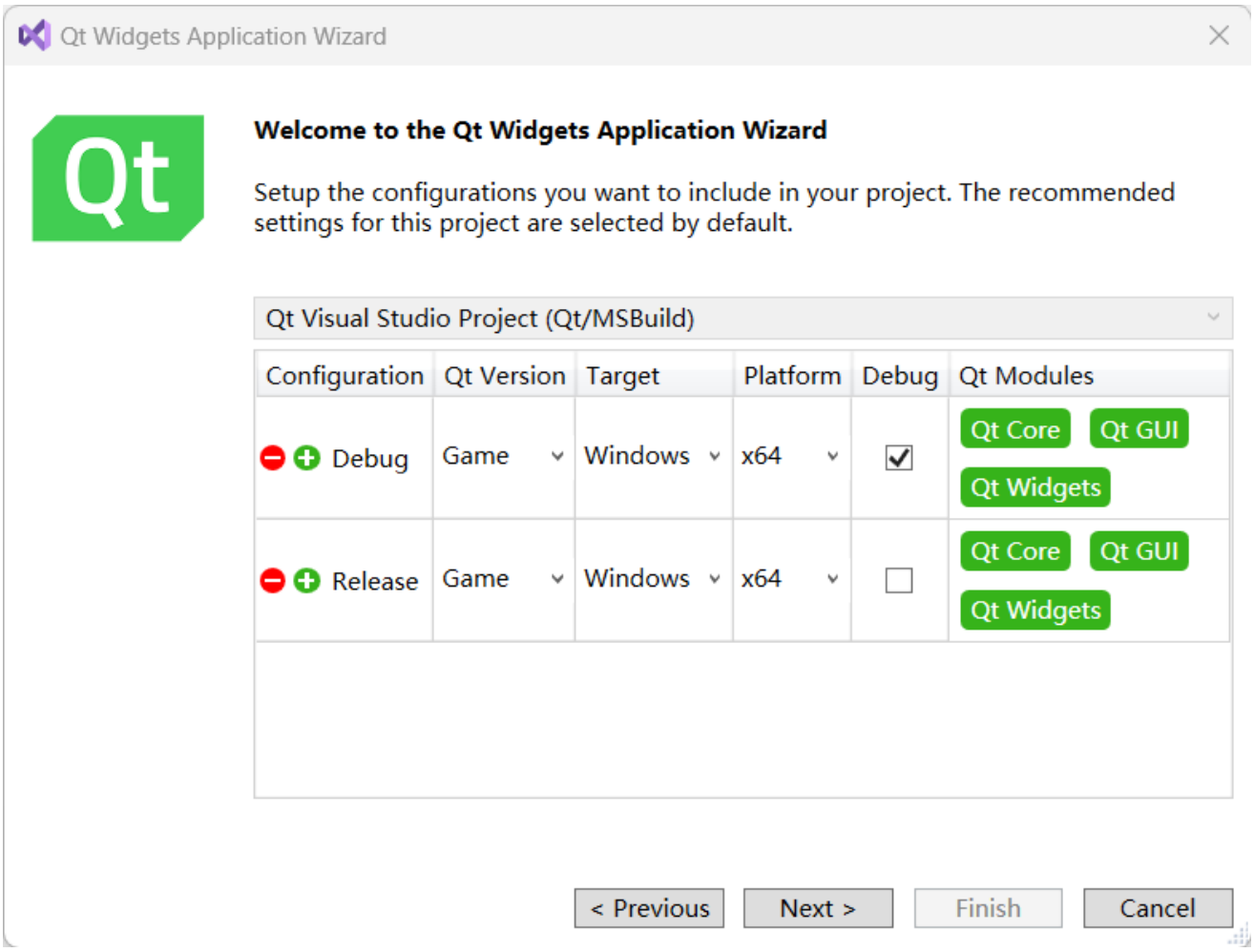


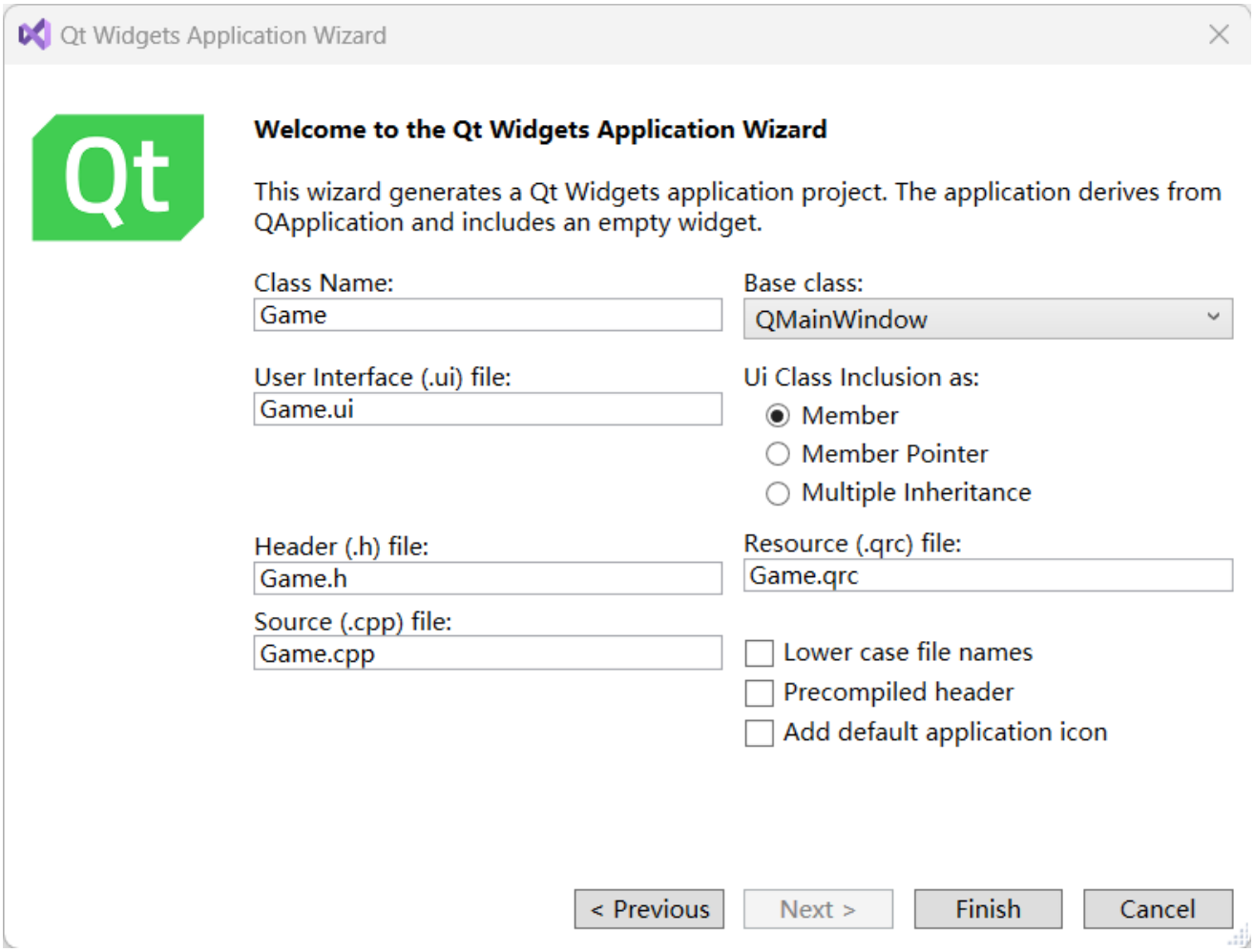
点击Next，配置工程





配置Qt Version，选择Game

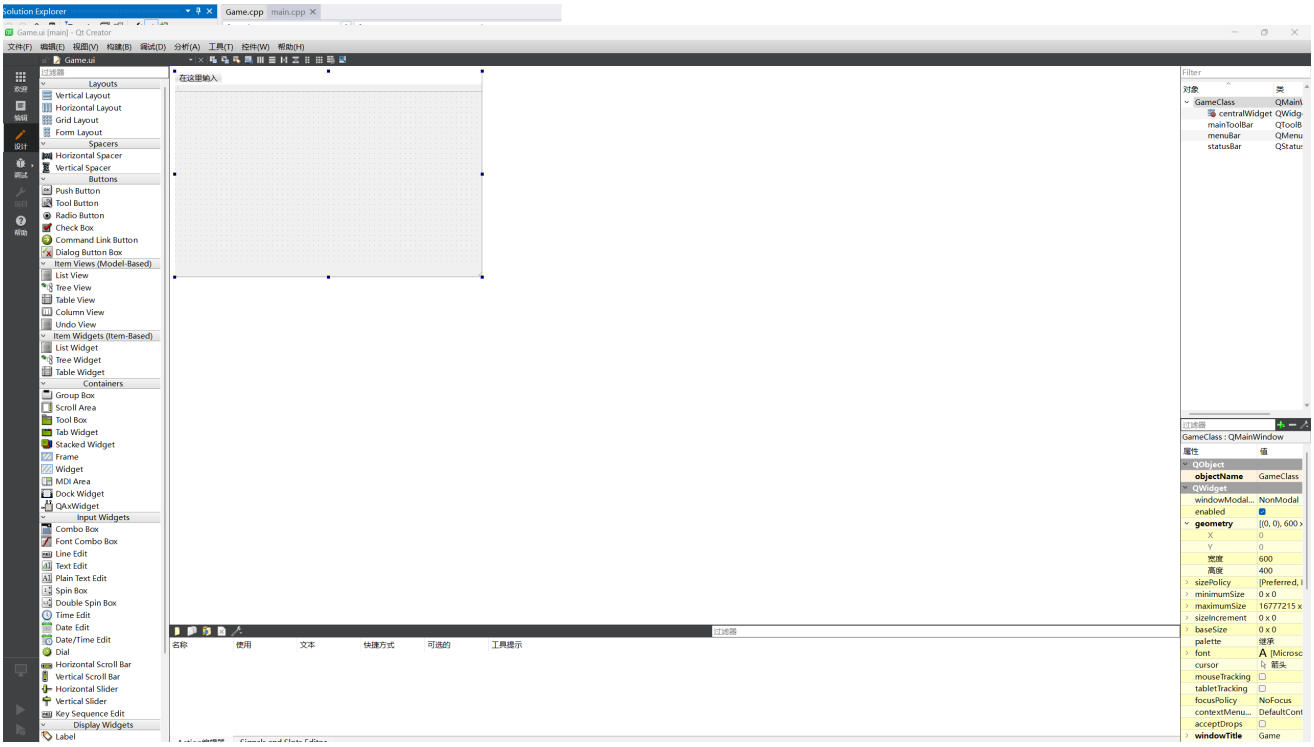




此时，除了正常的VS工程文件，还有Game.qrc, Game.ui。

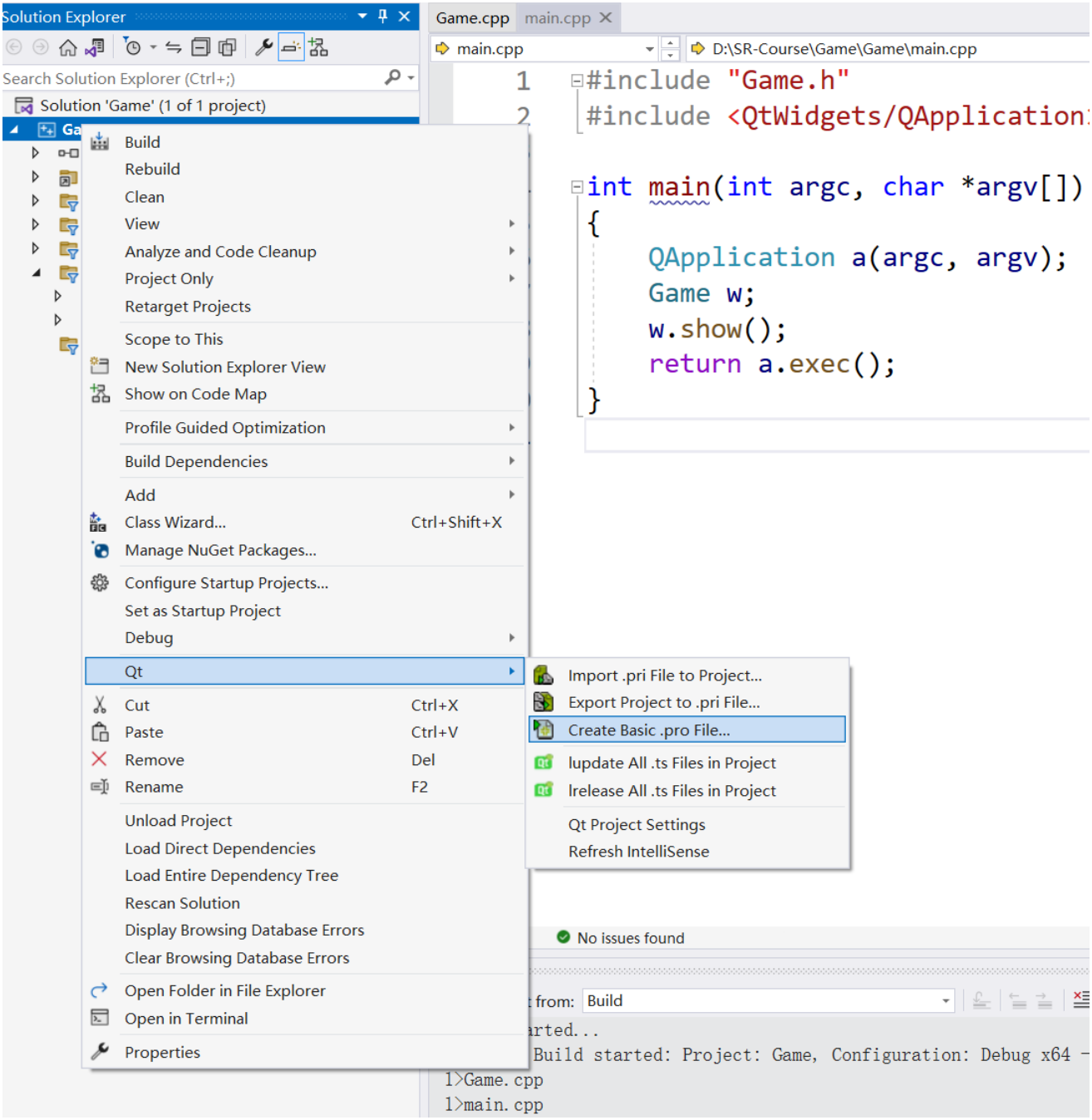
5.2 编辑GUI

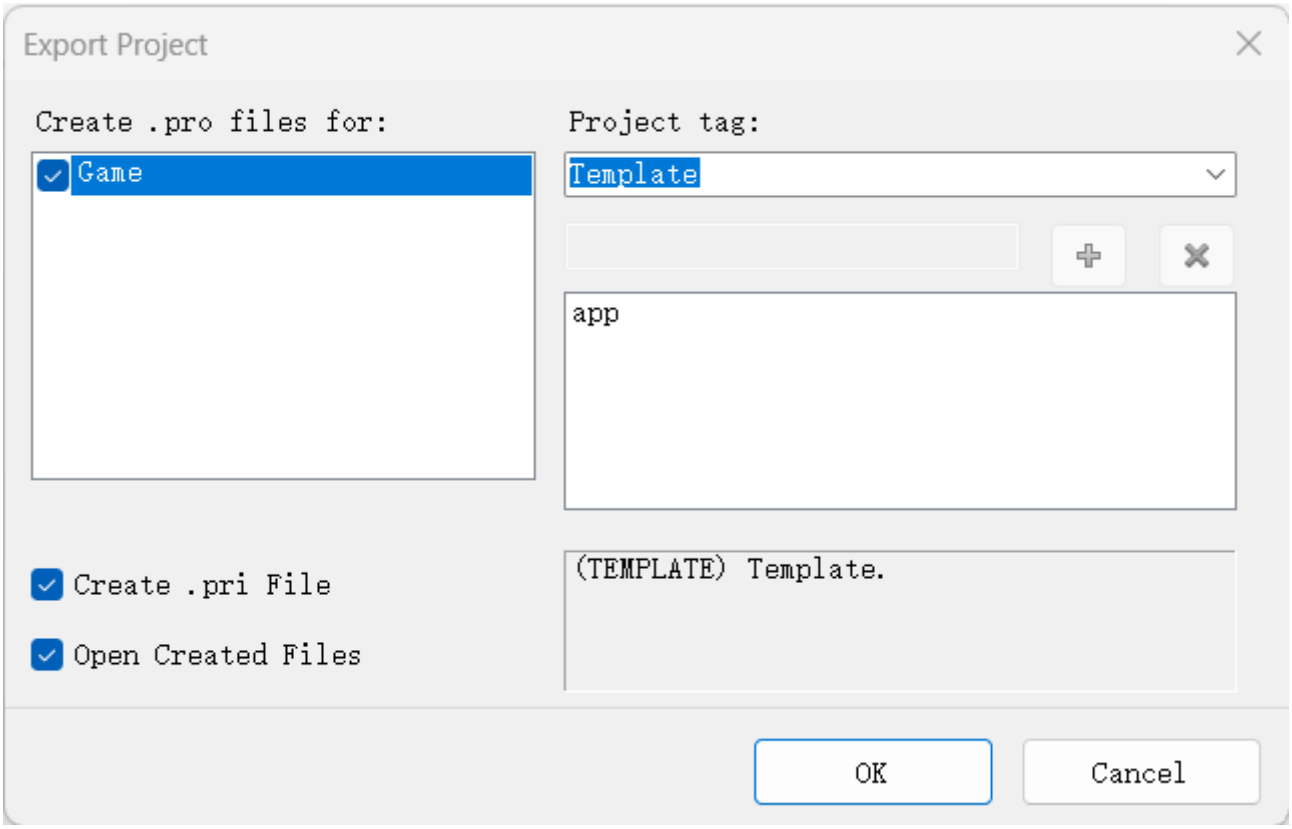
ui文件包含了用户界面的布局和外观，可以用Qt Creator直接打开编辑。



5.3 用Qt Creator打开工程

5.3.1 导出pro文件





导出Game.pro

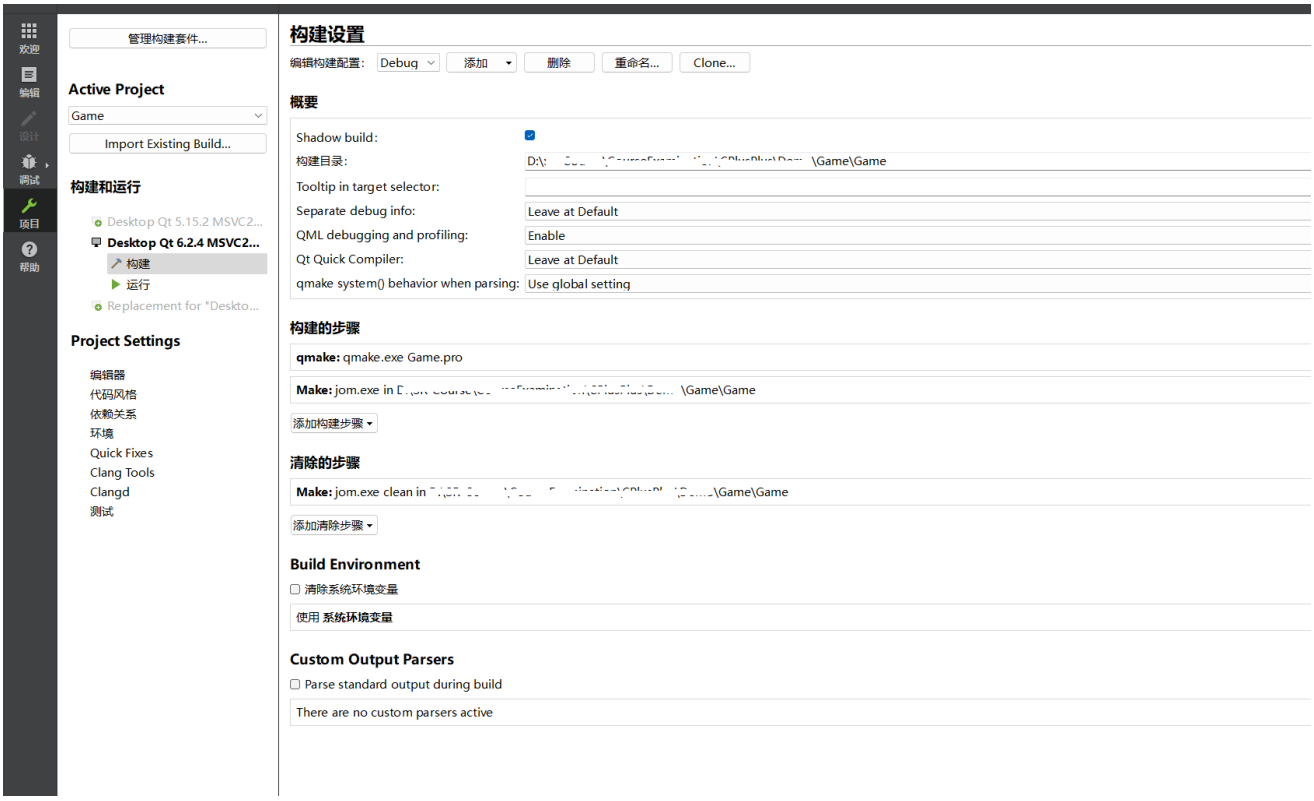
5.3.2 Qt Creator编译运行

用文本编辑器打开Game.pro，添加一行”QT+= widgets“

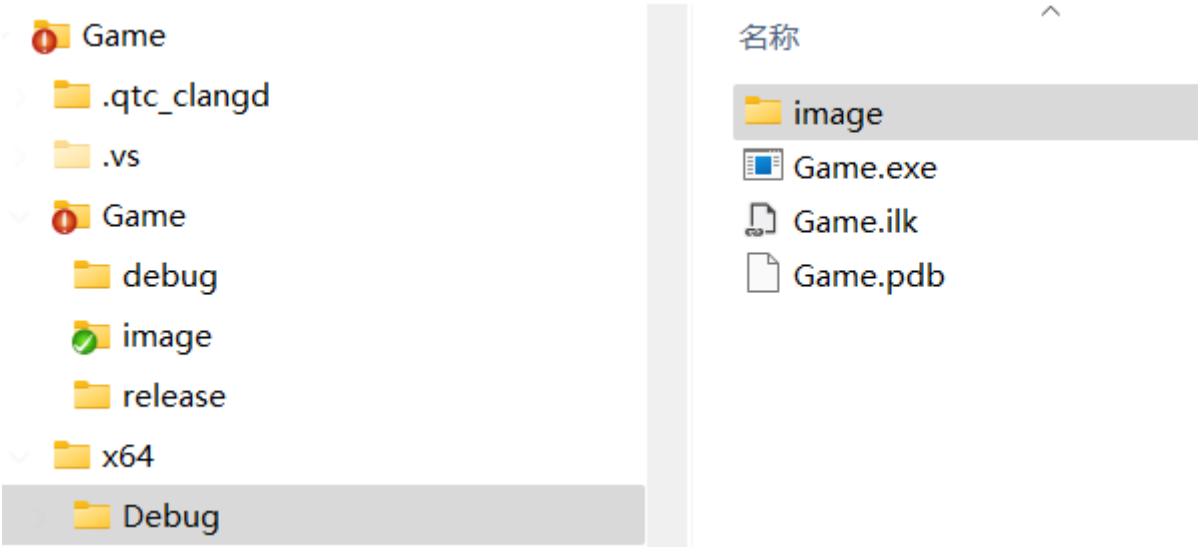
```
# -------------------------------------------------
# This file is generated by the Qt Visual Studio Tools.
# -------------------------------------------------

QT+= widgets

TEMPLATE = app
TARGET = Game
DESTDIR = ../x64/Debug
CONFIG += debug
LIBS += -L"."
DEPENDPATH += .
MOC_DIR += .
OBJECTS_DIR += debug
UI_DIR += .
RCC_DIR += .
include(Game.pri)
```

Qt Creator打开Game.pro，即可编译运行。

构建配置示意



编译出来的exe所在位置如图。注意应将数据文件放exe所在文件夹。



6. QMainWindow类型

main函数如下

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Game w;
    w.show();
    return a.exec();
}
```

MainWindow为QMainWindow派生的子类，GUI通过该子类对象来实现

```
class Game : public QMainWindow
{
    Q_OBJECT

public:
    Game(QWidget *parent = nullptr);
    ~Game();
private:
    void paintEvent(QPaintEvent*);
    void timerEvent(QTimerEvent*);
    void resizeEvent(QResizeEvent* event);
    void mousePressEvent(QMouseEvent* event);
    void keyPressEvent(QKeyEvent* event);

private:
    Ui::GameClass ui;

    QSize   window_size;
};
```

6.1 绘制

绘制在void paintEvent(QPaintEvent*)之中，由专门的QPainter对象来实现

```
void Game::paintEvent(QPaintEvent*)
{
    QPainter painter(this);
    //通过painter来实现绘制
}
```

画圆用QPainter::drawEllipse

```
painter.drawEllipse(pos.x - radius, pos.y - radius, radius * 2, radius * 2);

```

画矩形用QPainter::drawRect

```
painter.drawRect(pos.x - width / 2, pos.y - height / 2, width, height);
```

绘制图像使用Sprite类对象QPainter::drawImage

```
void Sprite::Render(QPainter& painter)
{
    QRectF target(pos.x - width / 2, pos.y - height / 2, width, height); //建立目标矩形
    QRectF source(0.0, 0.0, width, height); //建立源矩形，用来框定源图像文件中要显示的部分
    painter.drawImage(target, image, source); //image是QImage对象
}
```

6.2 事件响应

继承自QWidget Class，

QWidget Class提供了事件响应机制https://doc.qt.io/qt-6/qwidget.html#events，提供了了键盘、鼠标、窗口改变等事件处理函数

```
QWidget::paintEvent() //绘制函数
QWidget::resizeEvent()//窗口大小改变
QWidget::mousePressEvent() //鼠标某个键按下
QWidget::mouseReleaseEvent()//鼠标某个键松开下
QWidget::mouseDoubleClickEvent() //双击
QWidget::mouseMoveEvent() //鼠标移动
QWidget::keyPressEvent()//键盘某个键按下
```

如处理鼠标左键按下，event->pos()得到鼠标当前位置

```
void Game::mousePressEvent(QMouseEvent* event)
{
    QPoint pt = event->pos();
    g_interface.UserWndProc(0, pt.x(), pt.y());
}
```