Visual Studio 2008 QT编程学习笔记

1 安装

Visual Studio 2008使用的是简体中文版

QT使用的是Qt by Nokia v4.8.2(VS 2008 OpenSource)，安装软件包名字为“qt-win-opensource-4.8.2-vs2008.exe”。需要注意QT安装的目录不能有汉字和空格。

Visual Studio插件使用的是Qt Visual Studio Add-in 1.1.11，安装软件包名字为“qt-vs-addin-1.1.11-opensource.exe”。

2 配置

2.1 环境变量配置

系统变量中新增”QTDIR”项，其值为QT的安装路径，如下



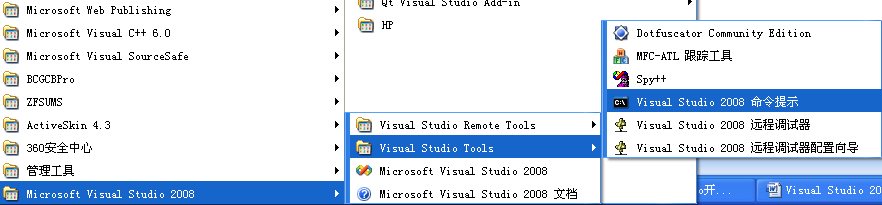
在Path变量中增加bin目录和lib目录，如下



配置完成后，需要重启计算机才能生效。

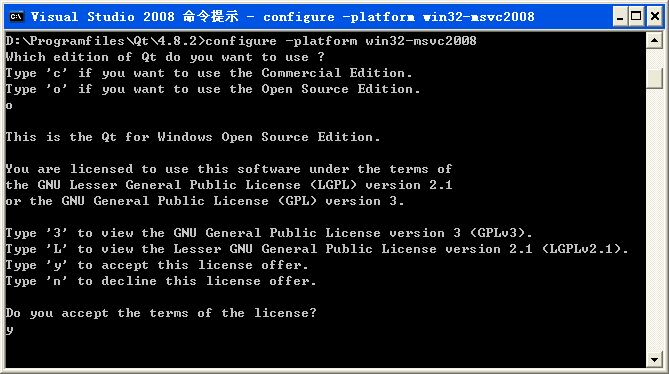
3 编译QT库

进入Visual Studio 2008 Comman Prompt，如下所示



进入到QT安装目录，执行以下命令

Configure –platform win32-msvc2008，如下所示



输入“o”选择开源版本，选择y接受license。接着开始配置，配置需要十几分钟，配置完成后，输入nmake开始编译。

我在执行nmake的时候，总是会编译一二十分钟后就出现错误，没办法，只好使用Visual Studio进行编译。

用Visual Studio 2008打开QT安装目录下的projects.sln，QT的整个工程便被装入VS中，这个过程比较慢，需要等一等，等全部装入完成后，点击VS的Build安装进行编译与连接，此后需要四五个小时的时间进行编译。

4 配置Visual Studio

使用Visual Studio建立一个C++控制台程序Hello，其代码如下所示(参考自《C++ GUI Qt 4编程第二版》)

#include<QApplication>

#include<QLabel>

int main(int argc,char \*argv[])

{

QApplication app(argc,argv);

QLabel \*label = new QLabel("Hello QT!");

label->show();

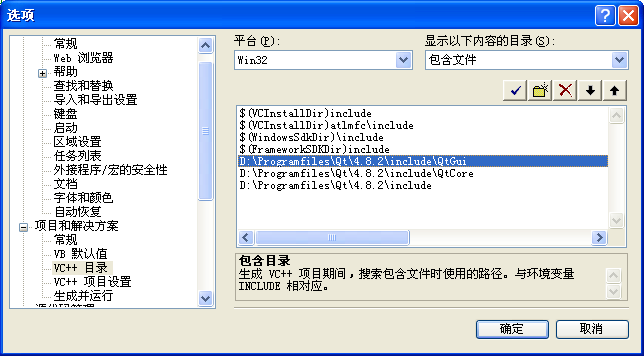
return app.exec();

}

这种情况下是无法通过编译的，需要对VS进行配置，配置过程如下：

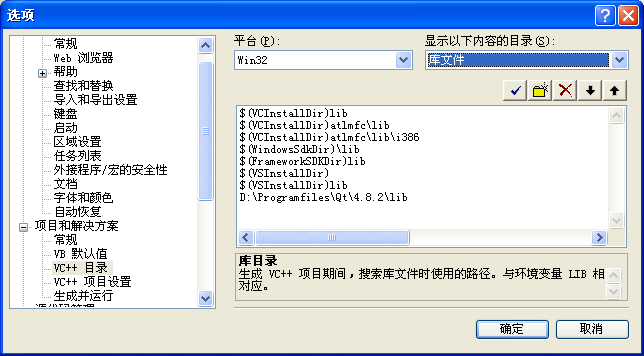
4.1 配置包含文件

选择“菜单”—“选项”—“项目和解决方案”—“VC++目录”—“win32平台”—“包含文件”，添加QT安装目录下的\include\QtGui、\include\QtCore和\include，如下所示



4.2 配置库文件

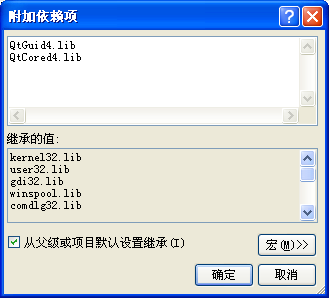
选择“库文件”，添加lib路径，如下所示



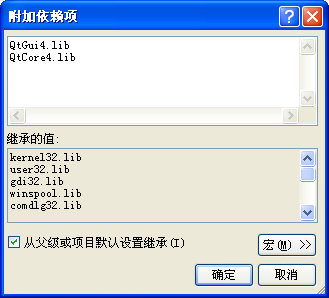
4.3 添加附加依赖项

在“解决方案资源管理器”中的项目名称上点击右键选择属性，在打开的属性页中选择“配置属性”—“链接器”—“输入”—“附加依赖项”，添加附加依赖项，Debug下和Release下的附加依赖项不一样，Debug下需要添加的有QtGuid4.lib和QtCored4.lib，如下所示

添加附加依赖项，Debug下和Release下的附加依赖项不一样，Debug下需要添加的有QtGuid4.lib和QtCored4.lib，如下所示



Release下需要添加是QtGui4.lib和QtCore4.lib，如下所示

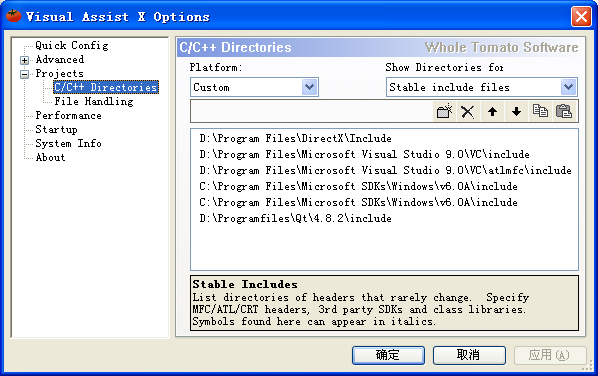


配置完成后，编译刚才的代码并执行，便能看到显示效果。

4.4 配置VAssistX

如果Visual Studio 2008中安装了编程助手VAssistX，则需要进行配置才能使编程助手识别出QT中的类、函数和变量，设置方法如下：

打开Visual Studio 2008，在菜单中找到VAssistX，选择Visual Assist X Options，进入配置页面，找到Projects—C/C++ Directories项，将Platform下的Win32改为Custom，在Stable include files项中添加QT安装目录下的include路径，如下图所示，配置完成后点击确定，重启VS后即可生效。



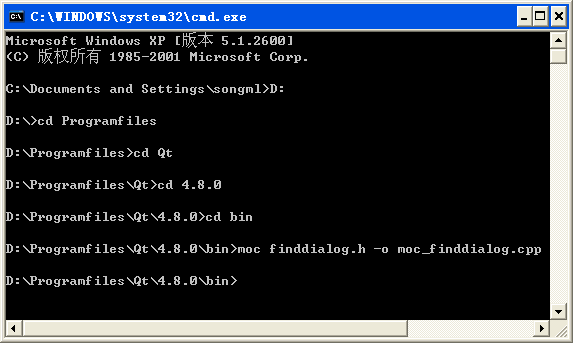
5 moc的使用

《C++ GUI Qt 4编程(第二版)》第12页给出了一个叫做FindDialog的例子，在这个例子中，头文件中声明类时用到了Q\_OBJECT宏，凡是用到Q\_OBJECT宏的类必须要使用moc，使用方法如下：

打开命令提示窗口，将路径定位至Qt安装文件夹的bin目录，同时，将类的头文件即FindDialog.h拷贝至bin目录，在命令提示窗口中运行如下命令

moc finddialog.h -o moc\_finddialog.cpp

如下图所示



-o表示输出

Moc\_finddialog.cpp为moc之后的文件，这个名字可以随意取。

运行完这个命令后，可以看到文件夹下生成了文件moc\_finddialog.cpp，将这个文件拷贝至或剪切至工程文件下，同时，在工程的源文件文件夹下添加该文件，如下图所示，



重新编译后该类便可使用了。

6 uic的使用

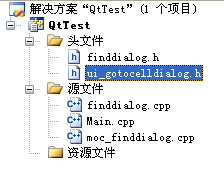
《C++ GUI Qt 4编程(第二版)》第17页使用QT设计师设计了一个界面，界面设计完成后，需要把ui文件转换成C++的头文件，这个转换使用uic工具，使用方法如下：

将界面文件即GoToCellDialog.ui拷贝至QT安装目录下bin文件夹，打开命令提示符，定位至bin文件夹，输入以下命令

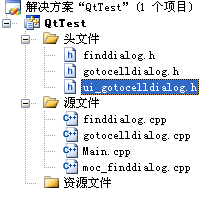
uic gotocelldialog.ui -o ui\_GoToCellDialog.h，如下所示



命令运行完成后，即可生成ui\_GoToCellDialog.h文件，将这个文件拷贝至原工程文件夹，并添加于工程之中，如下所示



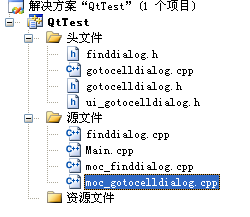
例子继续进行，后来又创建了一个头文件GoToCellDialog.h和实现文件GoToCellDialog.cpp，按照书上的步骤全部添加好后，如下图所示



编译时却出现如下错误



这是因为在GoToCellDialog.h中使用Q\_OBJECT宏，需要使用第5节所述的moc命令对GoToCellDialog.h进行操作，生成一个叫做moc\_GoToCellDialog.cpp的文件，并将其添加进工程中，如下图所示



此时再进行编译就没有问题了。

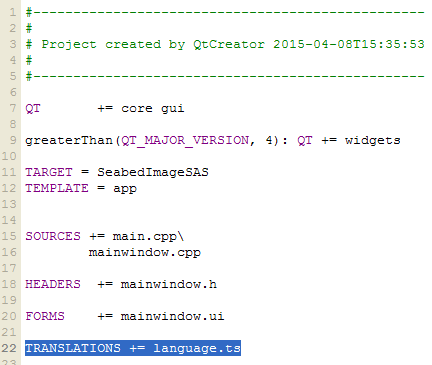
7 国际化

国际化可以用于不同语种之间的文字转换，例如英语转换成英语。以英语汉化成简体中文为例，其基本使用方法如下：

一、首先在pro文件中增加如下一行

TRANSLATIONS += language.ts

如下图所示



其中，language.ts是我们创建的翻译文件，名字可以自由取，但是后缀要求时ts。

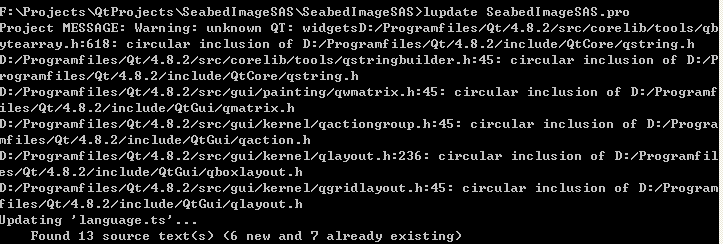
二、打开命令提示符，进入到pro文件所在的文件夹，如下



键入如下命令

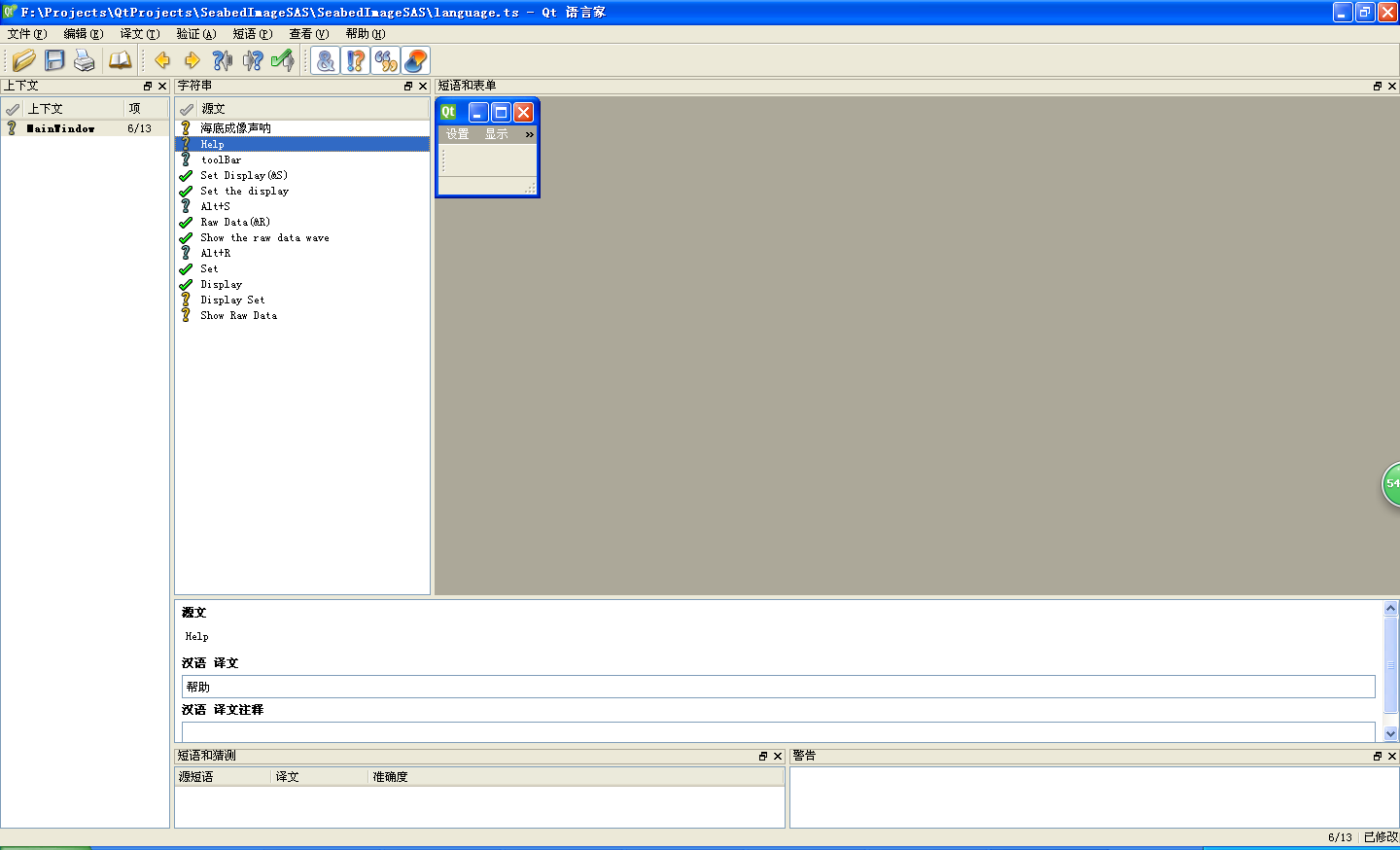
lupdate SeabedImageSAS.pro

其中，SeabedImageSAS.pro是工程名，按下回车键后开始运行，运行结果如下图所示

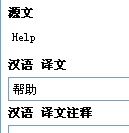


同时，会在目录下生成language.ts文件。

三、用QT的工具linguist打开language.ts，如下所示



所有可能需要翻译的地方都已经在“字符串”列了出来，选中其中的一项，在下面的译文中填入对应的简体中文的汉字即可，如下



填写完成后，需要点击按钮完成本条目，每一条填写完成后都要按这个按钮。然后才能开始下一条的填写。

所有条目都填写完成后，点击文件菜单中的保存，并点击文件菜单中的发布，会在目录中生成文件language.qm，该文件便是翻译得到的文件。

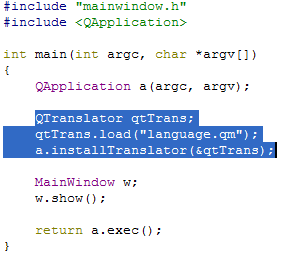
四、要加载language.qm文件，需要在main函数中增加如下代码

QTranslator qtTrans;

qtTrans.load("language.qm");

1. installTranslator(&qtTrans);

如下图所示



五、把language.qm放在build-SeabedImageSAS-unknown-Debug文件夹中，这是Qt Creator链接时使用的存放链接文件的文件夹，重新运行程序，会发现，英语已经被翻译成汉语了。

如果是程序已经生成好了，需要把language.qm与exe文件放置在同一个目录下，为了保险起见，可以在这两个文件夹中各放一个，这样不管是编译时还是最终运行时都不会出问题了。