# 首选在Qt creater下做开发

在使用QwebEngineView的使用 放在QMainWindow中时候应该用setCentralWidget函数

这个函数在设置中央组件的时候就可以先把之前占用中央的组件给remove掉，然后再来设置，而直接用centralWidget()获取中央组件后再给组件作父类会有一些bug

Q3DBars组件

Q3D视图形成后是每次addRow都是只加入前N个，就截止，下行继续从第一个开始载入，并且传入的是指针因此是数据会根据传入的指针的值的变化而变化

Q3Dbars组件的使用

Example:

{

Q3DBars bars;

bars.setFlags(bars.flags()^Qt::*FramelessWindowHint*);

bars.rowAxis()->setRange(0, 4);

bars.columnAxis()->setRange(0, 4);

QBar3DSeries \*series = *new* QBar3DSeries;

QBarDataRow \*data = *new* QBarDataRow;

\*data << 1.0f << 3.0f << 7.5f << 5.0f << 2.2f;

series->dataProxy()->addRow(data);

*//* *QBarDataRow* *\*data1* *=* *new* *QBarDataRow;*

data->clear();

\*data << 1.5f << 6.0f << 5.5f << 5.6f << 9.2f;

series->dataProxy()->addRow(data);

*//该3D视图形成后是每次addRow都是只加入前N个，就截止，下行继续从第一个开始载入，并且传入的是指针因此是数据会适时变化*

bars.addSeries(series);

bars.resize(500,500);

bars.show();

}

下面总结一下自定义信号槽需要注意的事项：

发送者和接收者都需要是QObject的子类（当然，槽函数是全局函数、Lambda 表达式等无需接收者的时候除外）；

使用 signals 标记信号函数，信号是一个函数声明，返回 void，不需要实现函数代码；

槽函数是普通的成员函数，作为成员函数，会受到 public、private、protected 的影响；

使用 emit 在恰当的位置发送信号；

使用QObject::connect()函数连接信号和槽。

Q3Dbars \* m\_graph->scene()->activeCamera()->setCameraPosition(-25.0f, 10.0f, 190.0f); *//设置观察者对应于图的位置，第一个是水平旋转角度，第二个是垂直旋转角度，第三个是放大*

在Qt中，直接用在父类中的resizeEvent()事件处理函数中用QTabWidget.resize()无法让QTabWidget跟随父类窗口大小变化，需要QTabWidget.currentWidget().resize()，这样才有效；

在利用Qt的connect()的时候，对象指针必须先被定义并且不能为空；

在My\_browser项目开发过程之中，VS和Qt creater有些不一样的坑，

通过connect（）连接时，在连续的一段函数代码段中，connect()书写方式应该一致，不能再中间用其他如lambda的连接方式，否则，在Qt creter下编译时会发生一些错误，因为编译器会优化，会全部默认为一种connect（）连接方式，

另外如函数返回值是value传递，则接受值应该用value值的方式接受，因为在Qt中这会认为可能发生错误，因为这会给你报错，如果是明确返回引用就用引用接受即可

在VS和Qt工程互相转换时要谨慎

在Qt中如果重新定义继承的函数时候要保持函数的原型，不能加任何修饰。如resizeevent()重写加inline就是错误