[**QT-窗口组件(QWidget),QT坐标系统,初探消息处理(信号与槽)**](https://www.cnblogs.com/lifexy/p/8876016.html)

**本章主要内容如下:**

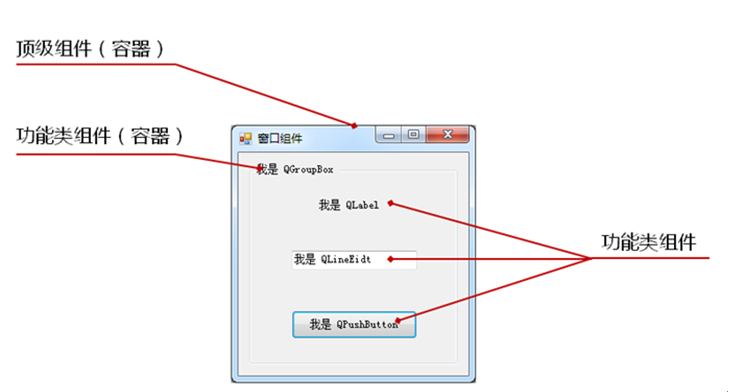
* **1)**窗口组件(QWidget)
* **2)**QT坐标系统
* **3)**消息处理(信号与槽)

**窗口组件(QWidget)**

**介绍**

* Qt以**组件对象的方式**构建图形用户界面
* Qt中**没有父组件**的顶级组件,则被**叫做窗口**
* 组件的类型分为:
* **容器类(父组件)** :  用于包含功能的界面组件
* **功能类(子组件)** :  用于实现特定的交互功能

如下图所示:

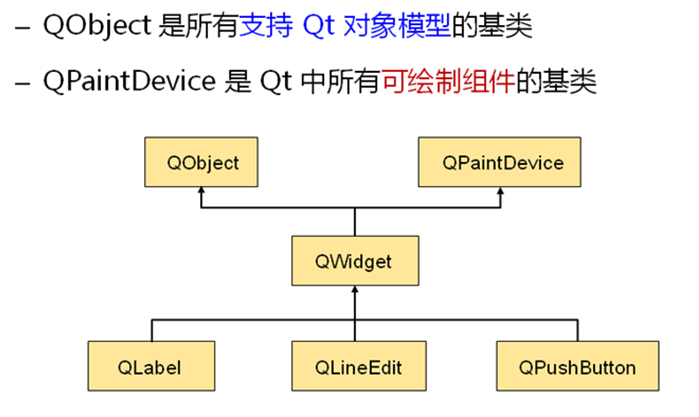


比如上面的**QgroupBox** ,即属于顶级组件的**功能类(子组件)**,又是3个功能类组件的**父组件(容器)**

**组件继承**

Qt中**所有窗口组件**都**继承于QWidget类**,而**QWidget类**又**继承于QObject类**和**QPaintDevice类.**

如下图所示(只举例了3个窗口组件类):

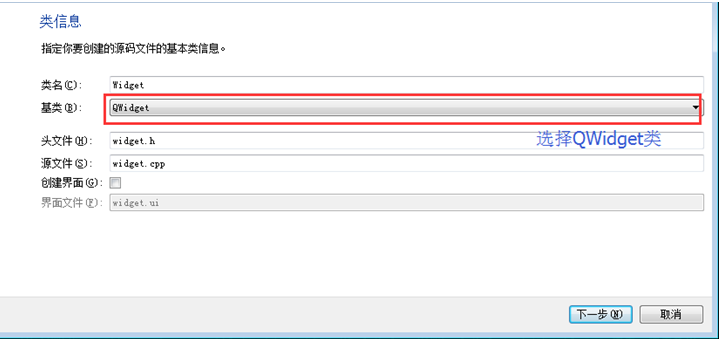


**QWidget组件介绍**

* QWidget能够绘制自己(因为继承了QPaintDevice类),也能够处理用户的输入,比如点击按钮
* QWidget是Qt窗口组件类的**父类**
* Qt中每个窗口组件都可以当做一个**QWidget**(因为子类可以初始化父类)
* QWidget类对象常作为**父组件**或**顶级组件**使用

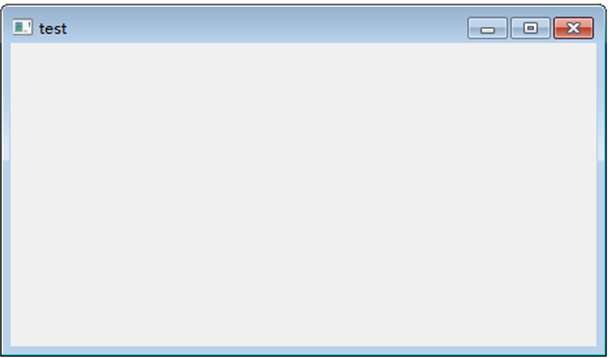
**初探QWidget**

**1)新建工程,选择Qt Gui应用,设置类信息:**



**2)生成QWidget模板**

运行模板:



可以看到生成了一个窗口,然后我们来看看**模板代码**,是如何生成的.

**3)模板代码如下所示**

[复制代码](javascript:void(0);)

#include <QtGui/QApplication>

#include "widget.h"

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

{

QApplication a(argc, argv);

QWidget w; //创建QWidget类对象

w.show(); //显示QWidget类对象

return a.exec();

}

[复制代码](javascript:void(0);)

根据之前讲的内容可以发现,由于上面的**QWidget w**对象没有父组件,所以**QWidget w**便成为了没有父组件的顶级组件,**从而生成了窗口**.

**Qt坐标系统**

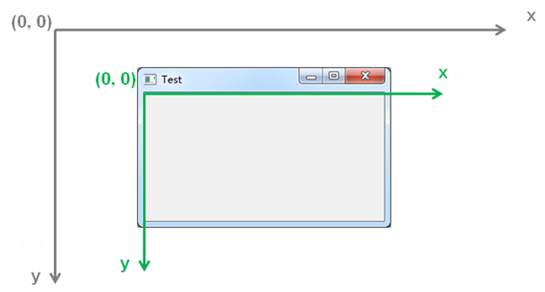
**介绍**

* Qt使用**统一的坐标系统**定位窗口部件的位置和大小
* **QWidget类**为组件类**提供了**窗口部件所需的**坐标系统成员函数**

**在Qt里,坐标类型分为**

* **顶级窗口部件**的定位
* **窗口内部件**的定位
* **窗口部件**的大小设置

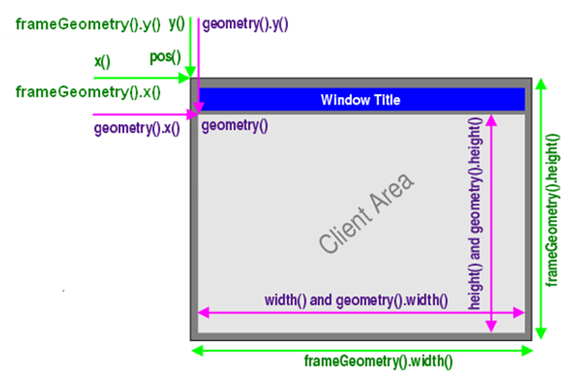
**坐标位置示意图,如下所示**



**QWidget类提供的常用坐标系统成员函数有:**

* **resize() :**   设置窗口内部的宽高( width()和height()值)
* **move() :**   设置整个窗口的x,y坐标( x()和y()值)
* **setGeometry() :**设置窗口内部的x,y,w,h(不包括标题和窗口边框)
* **size()  :**  获取窗口部件的大小
* **pos() :** 获取窗口部件的位置
* **x()  :**    获取整个窗口x坐标
* **y()  :**  获取整个窗口y坐标
* **width()  :**获取窗口内部的宽度(不包括外边框的宽度)
* **height()  :**   获取窗口内部的高度(不包括窗口标题栏的高度)
* **const QRect&  geometry ()  :**   获取窗口内部的x,y,w,h(不包括标题和窗口边框)
* **const QRect&  framgeometry ()  :** 获取整个窗口的x,y,w,h

**可以参考下图所示**



**注意:**在代码里,执行show()后, 再获取 x,y,w,h坐标 才有效

**接下来我们通过3组不同的获取坐标函数,来打印(x,y,w,h)坐标信息**

代码如下所示:

[复制代码](javascript:void(0);)

#include <QtGui>

#include "widget.h"

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

{

QApplication a(argc, argv);

Widget w;

QPushButton b("button",&w); //生成 QPushButton对象， 其父组件为 QWidget

/\*设置窗口大小位置\*/

w.resize(200,300);

w.move(300,300);

/\*设置按钮大小位置\*/

b.resize(100,50);

b.move(50,100);

w.show();

qDebug()<<"QWidget:";

qDebug()<<"x()="<<w.x();

qDebug()<<"y()="<<w.y();

qDebug()<<"width()="<<w.width();

qDebug()<<"height()="<<w.height();

qDebug()<<"QWidget::geometry()";

qDebug()<<"x="<<w.geometry().x();

qDebug()<<"y="<<w.geometry().y();

qDebug()<<"w="<<w.geometry().width();

qDebug()<<"h="<<w.geometry().height();

qDebug()<<"QWidget::frameGeometry()";

qDebug()<<"x="<<w.frameGeometry().x();

qDebug()<<"y="<<w.frameGeometry().y();

qDebug()<<"w="<<w.frameGeometry().width();

qDebug()<<"h="<<w.frameGeometry().height();

return a.exec();

}

[复制代码](javascript:void(0);)

运行打印:

[复制代码](javascript:void(0);)

QWidget:

x()= 300

y()= 300

width()= 200

height()= 300

QWidget::geometry()

x= 308

y= 330

w= 200  
h= 300

QWidget::frameGeometry()

x= 300

y= 300

w= 216

h= 338

[复制代码](javascript:void(0);)

 可以看到,获取的**窗内坐标(x,y)**永远比窗外坐标大,**窗外大小(w,h)**永远比窗内大小大

**初探消息处理(信号与槽)**

QT封装了具体操作系统的消息机制,如下图所示:



**Qt中定义了与系统信息相关的概念**

**信号(signal)**

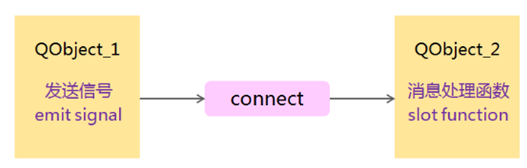
* 由操作系统产生的消息,**比如按键消息**

**槽(slot)**

* 程序中的消息处理函数,用来处理信号,**比如处理按键点击信号**

**连接(Connect)**

* 将系统信息绑定到信息处理函数**(信号到槽的连接)**,通过**connect()函数**实现,且必须发生在两个Qt类对象之间,如下图所示:



**connect()函数原型**

[复制代码](javascript:void(0);)

bool QObject::connect (

　　　　　　　　const QObject \* sender, 　　 //发送对象

　　　　　　　　const char \* signal, //消息名(信息)

　　　　　　　　const QObject \* receiver, //接收对象

　　　　　　　　const char \* method, //接收对象的成员函数(槽)

　　　　　　　　Qt::ConnectionType type = Qt::AutoConnection ) ; //正常情况不需要设置

//当出现sender对象的signal信号,则会自动调用receiver对象的method  
//连接成功，则返回true;否则返回false。

[复制代码](javascript:void(0);)

**在信号与槽里,Qt引进了几个新的关键字:**

* **SIGNAL** :指定消息名(信号),用于connect()函数里
* **SLOT**   : 指定消息处理函数名(槽),用于connect()函数里
* **Q\_OBJECT** : 指定该类拥有槽(消息处理),在类声明的内部开始处加上Q\_OBJECT即可
* **slots** : 用于在类中声明消息处理函数,比如:

private slots:

void buttonCliked();

**初探信号与槽**

通过点击按钮,使程序自动退出,代码如下所示:

[复制代码](javascript:void(0);)

#include <QtGui>

#include <QApplication>

#include <QPushButton>

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

{

　　QApplication app(argc,argv);

　　QPushButton \*quitButton = new QPushButton("Quit");

　　QObject::connect(quitButton, SIGNAL(clicked()), &app, SLOT(quit()));

　　//\*quitButton(发送对象), &app(接收对象)

　　//quit()作用是退出程序, QApplication的成员函数

　　//clicked()作用是鼠标点击, 很多常用组件的成员函数

quitButton->show();

return app.exec();

}

[复制代码](javascript:void(0);)

**其中上面的quit() 和clicked()都是系统预定义好的,接下来我们自定义槽**

**首先需要注意**

* + 类中声明槽(处理信号的成员函数)时,需要**slots声明**
  + 槽和信号的函数参数必须一致,**比如clicked()和quit()都是无参数的**
  + SIGNAL和SLOT指定的函数(信号和槽)只能包含参数类型,不能包含参数名

**开始试验,通过不同按钮点击,来打印不同的信息**

**写QButtonDebug.h:**

[复制代码](javascript:void(0);)

#ifndef QBUTTONDEBUG\_H

#define QBUTTONDEBUG\_H  
  
#include <QWidget>

#include <QPushButton>

class QButtonDebug : public QWidget

{

Q\_OBJECT //指定该类拥有slots(槽)

private:

QPushButton \*mbton1;

QPushButton \*mbton2;

private slots: //通过slots 声明 槽

void buttonCliked();

public:

explicit QButtonDebug(QWidget \*parent=0,Qt::WindowFlags f=0);

};

#endif

[复制代码](javascript:void(0);)

**写QButtonDebug.cpp:**

[复制代码](javascript:void(0);)

#include "QButtonDebug.h"

#include <QDebug>

QButtonDebug:: QButtonDebug(QWidget \*parent,Qt::WindowFlags f) :

QWidget(parent,f) //显示初始化父类

{

mbton1 = new QPushButton("button1",this);

mbton2 = new QPushButton("button2",this);

/\*设置按钮坐标\*/

mbton1->resize(100,50);

mbton1->move(50,50);

mbton2->resize(100,50);

mbton2->move(50,100);

/\*设置连接\*/

QObject::connect(mbton1,SIGNAL(clicked()),this,SLOT(buttonCliked()));

QObject::connect(mbton2,SIGNAL(clicked()),this,SLOT(buttonCliked()));

QWidget::show();

}

void QButtonDebug:: buttonCliked() 　　　　　　　　　　//消息处理函数

{

QPushButton\* p\_buton =dynamic\_cast<QPushButton\*>(sender()); //获取发送信号的对象 使用

qDebug()<< p\_buton->text(); //更据不同的按钮 打印不同信息

}

[复制代码](javascript:void(0);)

**写main.cpp**

[复制代码](javascript:void(0);)

#include <QtGui>

#include <QApplication>

#include "QButtonDebug.h"

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

{

　　QApplication a(argc, argv);

　　QButtonDebug b(NULL,Qt::WindowCloseButtonHint);

// Qt::WindowCloseButtonHint:去掉标题按钮提示

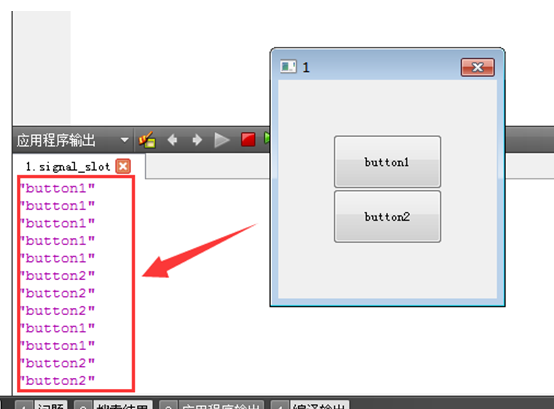
　　return a.exec();

}

[复制代码](javascript:void(0);)

**运行测试**

如下图所示,可以看到通过点击不同的按钮,便能打印不同的信息出来



**深入信号槽-自定义信号**

**介绍**

* 只有Qt类才能定义信号,且该类必须在头文件中声明
* **信号函数**只能通过**signals关键字**进行声明,**不能定义**,且返回值必须是**void类型**
* 信号函数的属性会被自动设置为protected类型
* 发送信号时,只需要通过**emit关键字**调用信号函数即可
* 如果信号函数的参数多于槽函数时,**多于的参数将被忽略**
* 槽函数的返回值**必须是void类型**,且可以被其它**普通成员函数**调用

**自定义信号示例:**

[复制代码](javascript:void(0);)

class MySignal : public QObject

{

　　Q\_OBJECT

signals: 　　　　//自定义信号函数

　　void SendSignal(int i);

public:

　　void send(int i)

　　{

　　　　emit SendSignal(i); //调用信号函数,发送信号

　　}

};

[复制代码](javascript:void(0);)

**自定义槽函数示例:**

[复制代码](javascript:void(0);)

class MySlot : public QObject

{

　　Q\_OBJECT  
protected slots:

　　void RecvSlot(int i)

　　{

　　　　qDebug()<<"Send:"<<sender()->objectName(); //打印发送对象名

　　　　qDebug()<<"Recv:"<<i;

　　　　qDebug()<<endl;

　　}

};

[复制代码](javascript:void(0);)

**信号与槽的组合**

* 信号函数可以连接多个槽函数
* 多个信号函数可以连接一个槽函数
* 一个信号就可以连接到另一个信号
* 通过connect函数进行连接,也可以通过disconnect函数取消连接

**示例1-多个信号连接一个槽:**

[复制代码](javascript:void(0);)

　　　　MySignal s1;

　　　　MySignal s2;

　　　　MySlot t;

　　　　s1.setObjectName("Signal1");

　　　　s2.setObjectName("Signal2");   
　　　　QObject::connect(&s1,SIGNAL(SendSignal(int)),&t,SLOT(RecvSlot(int)));

　　　　QObject::connect(&s2,SIGNAL(SendSignal(int)),&t,SLOT(RecvSlot(int)));

　　　　s1.send(10);

　　　　s2.send(12);

[复制代码](javascript:void(0);)

打印:

Send: "Signal1"

Recv: 10

Send: "Signal2"

Recv: 12

**示例2-信号1连接信号2,信号2连接槽:**

[复制代码](javascript:void(0);)

　　　　MySignal s1;

　　　　MySignal s2;

　　　　MySlot t;

　　　　s1.setObjectName("Signal1");

　　　　s2.setObjectName("Signal2");

　　　　QObject::connect(&s1,SIGNAL(SendSignal(int)),&s2,SIGNAL(SendSignal(int)));

　　　　QObject::connect(&s2,SIGNAL(SendSignal(int)),&t,SLOT(RecvSlot(int)));

　　　　s1.send(10);

　　　　s2.send(12);

[复制代码](javascript:void(0);)

打印:

　　　　Send: "Signal2"

　　　　Recv: 10

　　　　Send: "Signal2"

　　　　Recv: 12