c++论文

23级工科试验班邵康蕾

1.头文件引入：

#include "widget.h"：引入Widget类的头文件。

#include "ui\_widget.h"：这是由Qt Designer生成的UI类的头文件，通常包含了窗口的布局和控件的定义。

其他头文件包含了Qt库中的其他类和功能。

构造函数定义：

Widget::Widget(QWidget \*parent) : QWidget(parent), ui(new Ui::Widget) {...}：这是Widget类的构造函数，它接受一个指向QWidget的指针作为父对象（这允许Widget对象被嵌入到其他的Qt窗口中）。ui是一个指向Ui::Widget的指针，用于操作UI元素。

初始化UI：

ui->setupUi(this);：这是由Qt Designer生成的代码，用于初始化窗口的布局和控件。

对话框设置：

dlg.setWindowFlags(Qt::FramelessWindowHint | windowFlags());：设置dlg（一个对话框或其他窗口类型的对象，虽然在代码中未明确给出其类型）为无边框窗口，并保留其原有的窗口标志。

dlg.setAttribute(Qt::WA\_TranslucentBackground);：设置dlg的背景为透明。

hLayout->addWidget(l);：向水平布局hLayout中添加一个名为l的控件（l在代码段中没有定义，可能在类的其他部分或头文件中定义）。

dlg.setLayout(hLayout);：设置dlg的布局为hLayout。

定时器设置：

timerDig = new QTimer(this);：创建一个新的定时器对象timerDig，并将其父对象设置为当前Widget。

timerDig->setInterval(5000);：设置定时器的间隔为5000毫秒（即5秒）。

connect(timerDig,&QTimer::timeout,this,&Widget::slot\_timerout);：连接定时器的timeout信号到当前类的slot\_timerout槽函数。当定时器超时（即5秒后）时，会调用slot\_timerout函数。

其他初始化：

initUI();：调用一个名为initUI的成员函数，用于进行其他可能的UI初始化操作（尽管这个函数在代码段中没有给出）。

2.这段代码是Widget类的析构函数和paintEvent成员函数的实现。

析构函数：

Widget::~Widget() {

delete ui;

}

这是Widget类的析构函数。在对象被销毁之前，析构函数会被调用。在这个析构函数中，delete ui;语句用于释放由ui指针所指向的内存。这个ui指针是在构造函数中通过new Ui::Widget创建的，用于设置UI界面。

paintEvent成员函数：

void Widget::paintEvent(QPaintEvent \*){

QPainter p(this);

p.drawPixmap(0, 0, width(), height(), QPixmap(":/ico/attachments/background.png"));

}

这是Widget类的paintEvent成员函数。在Qt中，当窗口或控件需要重绘时，paintEvent函数会被调用。这个函数通常用于自定义窗口或控件的外观。

在这个paintEvent函数中，首先创建了一个QPainter对象p，它用于在窗口上进行绘制。

然后，使用p.drawPixmap方法绘制一个图片作为窗口的背景。这个图片位于Qt的资源系统中，路径是:/ico/attachments/background.png。drawPixmap函数的参数指定了图片在窗口上的绘制位置（左上角为(0, 0)）和大小（窗口的宽度和高度）。

3.你的initUI函数是用于初始化Widget类的一些用户界面元素的，特别是系统托盘（system tray）的相关设置。以下是对你代码的详细解释：

设置窗口标志:

setWindowFlags(Qt::FramelessWindowHint|Qt::WindowStaysOnTopHint|Qt::Tool);

这里设置了窗口的一些属性。

Qt::FramelessWindowHint使窗口没有边框，Qt::WindowStaysOnTopHint使窗口始终保持在其他窗口的上面，Qt::Tool则表示这是一个工具窗口。

设置动作的信号与槽:

使用connect函数将动作（QAction）的triggered信号连接到了一些lambda函数或成员函数上。例如，当点击“显示窗口”时，窗口会显示并移动到特定的位置。

当点击“退出”时，程序会退出。

创建系统托盘图标:

创建了一个QSystemTrayIcon对象（trayIcon），并为其设置了上下文菜单（托盘菜单）、图标、工具提示等。

也连接了activated信号到一个槽函数（slot\_trayIconActived）上，该函数在托盘图标被激活（例如，被点击）时会被调用。

设置窗口位置:

获取了托盘图标的几何位置，并根据这个位置来设置窗口的位置，使其位于托盘图标的某个位置附近。

4.在这段代码中，Widget::slot\_trayIconActived 是一个槽函数（slot function），它响应 QSystemTrayIcon 对象的激活事件。QSystemTrayIcon 是 Qt 框架中用于在系统托盘（也称为通知区域或任务栏）中显示图标的类。

这个函数接受一个 QSystemTrayIcon::ActivationReason 枚举类型的参数 reason，该参数指示了托盘图标被激活的原因（如左键点击、中键点击、双击等）。

下面是代码段的逐行解释：

void Widget::slot\_trayIconActived(QSystemTrayIcon::ActivationReason reason): 声明了一个返回类型为 void 的槽函数，它属于 Widget 类，并接受一个 QSystemTrayIcon::ActivationReason 类型的参数 reason。

switch(reason): 使用 switch 语句根据 reason 参数的值来执行不同的代码块。

case QSystemTrayIcon::Trigger:: 如果 reason 等于 QSystemTrayIcon::Trigger（这通常表示左键点击或双击托盘图标），则执行以下代码。

Widget::on\_btn\_band40\_clicked();: 调用 Widget 类中的 on\_btn\_band40\_clicked 函数（可能是一个槽函数，由 Qt 的信号和槽机制自动生成，用于响应某个按钮的点击事件）。

break;: 结束 case 语句的执行，防止进入下一个 case。

case QSystemTrayIcon::MiddleClick:: 如果 reason 等于 QSystemTrayIcon::MiddleClick（表示中键点击托盘图标），则执行以下代码。

注意这里和 QSystemTrayIcon::Trigger 的处理是一样的，都调用了 Widget::on\_btn\_band40\_clicked(); 函数。

break;: 同样结束 case 语句的执行。

default:: 如果 reason 不匹配上述任何 case，则执行此处的代码

}: 结束 switch 语句。

}: 结束 slot\_trayIconActived 函数的定义。

5.这段代码是C++中Qt框架的一部分，它定义了一个名为Widget的类中的一个槽函数slot\_trayIconActived。这个函数用于处理系统托盘图标（QSystemTrayIcon）的不同激活原因。下面是对这段代码的详细解释：

函数定义：

void Widget::slot\_trayIconActived(QSystemTrayIcon::ActivationReason reason): 这是一个没有返回值（void）的成员函数，属于Widget类。它接受一个枚举值作为参数，这个枚举值表示系统托盘图标的激活原因（QSystemTrayIcon::ActivationReason）。

switch语句：

这里使用switch语句来根据不同的激活原因执行不同的操作。

case QSystemTrayIcon::Trigger:

当系统托盘图标被点击（左键点击通常是这种情况）时，会执行这个case下的代码。

Widget::on\_btn\_band40\_clicked();: 这行代码调用了Widget类中的另一个槽函数on\_btn\_band40\_clicked。从函数名可以推测，这可能是某个按钮（可能是名为“band40”的按钮）的点击事件处理函数。但值得注意的是，直接在槽函数中调用另一个槽函数可能不是最佳实践，除非这是特定应用场景下的需求。

case QSystemTrayIcon::MiddleClick:

当系统托盘图标被中键点击时，也会执行Widget::on\_btn\_band40\_clicked();这行代码。

这意味着无论是左键点击还是中键点击托盘图标，都会触发相同的操作。

default:

如果没有匹配到任何case，则执行default下的代码。但在这个例子中，default下没有任何代码，所以什么都不会发生。

1. on\_btn\_hide\_clicked()

这个函数是响应名为“btn\_hide”的按钮点击事件的。当这个按钮被点击时，会执行this->hide();这行代码。这行代码的作用是隐藏当前Widget对象，即使得该窗口或控件在屏幕上不可见。this指针指向的是调用该函数的Widget对象实例。

on\_btn\_band3\_clicked()

这个函数是响应名为“btn\_band3”的按钮点击事件的。当这个按钮被点击时，会执行cmd.start("calc");这行代码。但是，这里有一个潜在的问题或不明确之处，因为cmd这个对象并没有在给出的代码片段中定义。我假设cmd是一个成员变量，可能是某种执行系统命令或启动外部程序的对象。

如果cmd是一个QProcess对象（这是Qt中用于执行外部程序或命令的常用类），那么cmd.start("calc");会尝试启动Windows的计算器程序（因为"calc"是Windows上启动计算器的命令）。

总的来说，这两个函数都是槽函数，用于响应按钮点击事件。第一个函数隐藏当前窗口，而第二个函数（基于假设）尝试启动Windows的计算器程序。

7.这段代码是Widget类中的一个槽函数on\_btn\_band40\_clicked，该函数在名为“btn\_band40”的按钮被点击时执行。下面是对这段代码的详细解释：

创建动画对象：m = new QMovie(QString(":/ico/attachments/gif/%1.gif").arg(QRandomGenerator::global()->bounded(0,8)));

这行代码创建了一个新的QMovie对象m，该对象用于播放GIF动画。GIF动画的路径由资源路径（:/ico/attachments/gif/）和一个随机的文件名组成。文件名通过QRandomGenerator::global()->bounded(0,8)生成，这意味着文件名将是0.gif到8.gif之间的一个随机文件。

获取当前帧的QPixmap：

QPixmap p = m->currentPixmap();

这行代码试图获取QMovie对象m当前帧的QPixmap对象

设置对话框大小和标签的动画：

dlg.setFixedSize(p.width(), p.height());

l->setMovie(m);

m->start();

l->setScaledContents(true);

这几行代码做了以下事情：

dlg.setFixedSize(p.width(), p.height());：设置某个对话框（可能是成员变量dlg）的固定大小为当前帧的宽度和高度（尽管前面提到p可能没有被直接使用，但这里假设它代表了动画的某个合适大小）。

l->setMovie(m);：将QMovie对象m设置为某个标签（可能是成员变量l）的动画源。

m->start();：开始播放动画。

l->setScaledContents(true);：设置标签的内容（即GIF动画）可以缩放以适应标签的大小。

显示对话框并启动定时器：

dlg.show();

timerDig->start();

dlg.show();：显示前面设置过大小和内容的对话框。

timerDig->start();：启动一个定时器（可能是成员变量timerDig）。这个定时器可能用于更新对话框的内容、动画或其他与这个操作相关的定时任务。

注意：

在这段代码中，如果timerDig没有被定义为一个QTimer对象并连接了适当的信号和槽，那么timerDig->start();将不会有任何效果。

1. 这段代码是Widget类中的一个槽函数slot\_timerout，它通常与某个定时器（可能是timerDig）的timeout()信号连接，当定时器超时时会被调用。以下是该函数的详细解释：

隐藏对话框：

dlg.hide();

这行代码隐藏了前面显示过的对话框dlg。如果dlg是一个QDialog或其派生类的实例，并且之前通过dlg.show();显示过，那么这行代码会将其从屏幕上移除，但并不会删除它。

停止动画并删除QMovie对象：

m->stop();

delete m;

m = nullptr;

首先，m->stop();停止了之前通过m->start();启动的GIF动画。然后，delete m;释放了QMovie对象m占用的内存。最后，将指针m设置为nullptr，以确保之后不会错误地访问已经被删除的对象。

停止定时器：

timerDig->stop();

这行代码停止了与slot\_timerout槽函数关联的定时器timerDig。如果timerDig是一个QTimer对象，并且之前通过timerDig->start();启动过，那么这行代码会停止其计时，并且当定时器再次触发时，不会再调用slot\_timerout槽函数。

注意：

将指针设置为nullptr是一个好的做法，但它并不自动删除对象。删除对象是通过delete操作符完成的。将指针设置为nullptr只是确保之后不会错误地访问该指针所指向的内存位置。