图形填充规则：

函数QPainterPath：：setFillRule，Qt::WindingFill（射线法，判断焦点的方向，不同方向的焦点相加为0,和>0填充），OddEvenFill(射线法，焦点为技术则填充，偶数不填充).

color.setAlpha（）设置颜色的透明度

画图

color.setAlpha(150 - qSqrt(i)\*50);

painter.setPen(color);

painter.drawPath(path);

填充

布局

QVBoxLayout \*pLayout

pLayout->addwidget(button);

pLayout->addStretch();加弹簧

setLayout(pLayout);

打开保存对话框QFileDialog

第一个参数parent，用于指定父组件。注意，很多Qt组件的构造函数都会有这么一个parent参数，并提供一个默认值0；

第二个参数caption，是对话框的标题；

第三个参数dir，是对话框显示时默认打开的目录，"." 代表程序运行目录，"/" 代表当前盘符的根目录(Windows，Linux下/就是根目录了)，也可以是平台相关的，比如"C:\\"等；

第四个参数filter，是对话框的后缀名过滤器；

多个文件使用空格分隔：比如我们使用"Image Files(\*.jpg \*.png)"就让它只能显示后缀名是jpg或者png的文件。

多个过滤使用两个分号分隔：如果需要使用多个过滤器，使用";;"分割，比如"JPEG Files(\*.jpg);;PNG Files(\*.png)"；

第五个参数selectedFilter，是默认选择的过滤器；

第六个参数options，是对话框的一些参数设定，比如只显示文件夹等等，它的取值是enum QFileDialog::Option，每个选项可以使用 | 运算组合起来。

信号和槽：

信号声明： signals: 发射信号 槽返回值 = emit mysignal();

槽声明：public slots:

链接：connect（sender，SIGNAL(),receive,SLOT（））；

设置树多选

nodetreeWdiget->setSelectionMode(QAbstractItemView::ExtendedSelection);

然后通过QList<QTreeWidgetItem\*> selectedItemList = this->selectedItems();获得所选择的items。

item角色

item->data(Qt::UserRole).toInt()

lineEdit

cdlg.ui.lineEdit->setText(text);设置文字

cdlg.ui.lineEdit->selectAll();选择全部

QVariant 设置自定义类型的指针

pPortraitView->setData( QVariant::fromValue((QObject\*)pNewItem) ) ;

列表排序功能实现

ui.treeWidget->setSortingEnabled(true);

增加启动界面

QSplashScreen类

pspalsh->setPixmap(QPixmap(":/Post/Resources/launch.jpeg"));

pspalsh->show();

Qt::Alignment topright = Qt::AlignRight | Qt::AlignRight;

pspalsh->showMessage(QObject::tr("Loading..."), topright, Qt::white);

Post w;

w.show();

pspalsh->finish(&w);

Qtreewidget + QtreeWidgetItem::setcheckstate()会发射itemChanged(QtreeWidgetItem \*,int)信号

自定义排序方式

listP->setSortingEnabled( false ); //不使能QT的自动排序

listP->header()->setSortIndicatorShown( true ); // 设置三角标识符可见

listP->header()-setSectionsClickable( true ); // 设置标题栏单击响应使能

然后在

connect( listP->header(), SIGNAL( sectionClicked(int) ), this, SLOT( OnClickColumn(int) ) );

的OnClickColumn( int ) 响应中

加上自定义的排序代码就可以了。

对容器的排序

qSort(begin,end,函数指针)

bool compare(const BarAmount &barAmount1, const BarAmount &barAmount2)

编译ui文件

在你的qt安装路径的bin目录中有个uic.exe 的程序 这个是用来生成 ui文件对应的头文件用的。

使用命令 uic.exe a.ui -o a.h 就可以生成代码了。 你可以使用a.h 中生成的 类。那是一个窗口控件类，申明一个实例 然后 .show() 就可以了。

如果你想自动的将 ui 文件编成 h文件 需要安装 qt addin for visual studio

然后在你的项目上右键 选择 转换为 qt－addin 工程 就可以。

编译具有Q\_OBJECT宏的类型：

使用moc工具

命令行 ：moc.exe xxxx.h –o moc\_xxx.cpp

自定义生成工具-常规：

命令行："$(QTDIR)\bin\moc.exe"  "%(FullPath)" -o ".\GeneratedFiles\$(ConfigurationName)\moc\_%(Filename).cpp"

输出：.\GeneratedFiles\$(ConfigurationName)\moc\_%(Filename).cpp

1.手工编译.qrc文件：

用如下的命令可以将ras.qrc文件转化成cpp代码:

rcc.exe -no-compress ras.qrc -o ras.cpp

这样在vs项目中添加ras.cpp文件后，就可以用如下的路径来使用资源。

QIcon icon\_app(":/ras/app.png");

拖拽功能实现

void dragEnterEvent(QDragEnterEvent \*event) Q\_DECL\_OVERRIDE;

void dragMoveEvent(QDragMoveEvent \*event) Q\_DECL\_OVERRIDE;

void dropEvent(QDropEvent \*event) Q\_DECL\_OVERRIDE;

void mousePressEvent(QMouseEvent \*event) Q\_DECL\_OVERRIDE;

QMimeData 使用方式 ，

QByteArray使用

QByteArray aData;

QDataStream stream(&aData, QIODevice::WriteOnly);

stream<< event->pos();

QMimeData \*pmime = new QMimeData();

pmime->setData("Position", aData);

使用QPainter画透明度

QPainter painter;

painter.*begin*(&cpixMap);

painter.fillRect(cpixMap.rect(),QColor(127,127,127,127));

painter.*end*();

获得窗口部件childAt

1. 对搜索于setAlignment()，看函数名就知道是设置对齐方式。比如QLayout中的setAlignment()：

|  |  |
| --- | --- |
| 1 | bool QLayout::setAlignment(QWidget \* w, Qt::Alignment alignment) |

就是对窗口w设置对齐方式，你可以看看帮助文档，Qt::Alignment有左对齐、右对齐、居中对齐等等。

* sizePolicy
* 这个属性保存了该 widget 的默认布局属性，如果它有一个 layout 来布局其子 widgets，那么这个 layout 的 size policy 将被使用；如果该 widget 没有 layout 来布局其子 widgets，那么它的 size policy 将被使用。默认的 policy 是 Preferred/Preferred。QSizePolicy::Policy 枚举值有如下几个：

|  |  |
| --- | --- |
| Constants | Description |
| QSizePolicy::Fixed | widget 的实际尺寸只参考 sizeHint() 的返回值，不能伸展（grow）和收缩（shrink） |
| QSizePolicy::Minimum | 可以伸展和收缩，不过sizeHint() 的返回值规定了 widget 能缩小到的最小尺寸 |
| QSizePolicy::Maximum | 可以伸展和收缩，不过sizeHint() 的返回值规定了 widget 能伸展到的最大尺寸 |
| QSizePolicy::Preferred | 可以伸展和收缩，但没有优势去获取更大的额外空间使自己的尺寸比 sizeHint() 的返回值更大 |
| QSizePolicy::Expanding | 可以伸展和收缩，它会尽可能多地去获取额外的空间，也就是比 Preferred 更具优势 |
| QSizePolicy::MinimumExpanding | 可以伸展和收缩，不过sizeHint() 的返回值规定了 widget 能缩小到的最小尺寸,同时它比 Preferred 更具优势去获取额外空间 |
| QSizePolicy::Ignored | 忽略 sizeHint() 的作用 |

Qt实现动画效果

QPropertyAnimation \*anim2=new QPropertyAnimation(bird\_2, “pos”);  
  
anim2->setDuration(2000);  
  
anim2->setStartValue(QPoint(0, 0));  
  
anim2->setEndValue(QPoint(150, 180));  
  
anim2->setEasingCurve(QEasingCurve::OutBounce);

Anim2->starte()

QVBoxLayout\* verticalLayout\_2 = new QVBoxLayout(widget);

verticalLayout\_2->setSpacing(110);设置widget之间的空格

verticalLayout\_2->setContentsMargins(11, 11, 11, 11);设置四周到边界的空格