### Qt环境下图像的打开和涂色

1. 设计目标

能够在 Qt QtCreator 环境下打开常用图像格式文件，诸如 bmp、jpg、png 图像等，然后将他们转化为 Qt 中的 QImage 类，并进行矩形范围内的涂色。

1. 需要提前掌握的知识
2. Qt 图像类

Qt 中用于图像操作的类有 QImage 和 QPixmap，其中 QImage 主要负责图像编辑和修改，而 QPixmap 则负责图像的显示。

1. Qt 如何修改图像

QImage 中有一个 setPixelColor() 方法可以对image 的像素点进行操作，但是当需要操作的图像区域面积较大时，该方法的效率较低。这时最有效率的方法就是直接操作 image 在内存中的数据，可以通过 bits() 方法来获取 image 数据的首地址。

1. Qt 如何显示图像

Qt 中图像的显示主要是通过使用 QLabel 类来实现的。步骤：先打开一个图像；将图像加载进 QImage 中；在用 QPixmap 对象获得图像；最后用 QLabel 使用 setPixmap() 方法显示。

1. 程序需求
2. 一个包含打开图像，涂色功能的菜单的主窗口；
3. 一个可以获取需要修改的矩形范围、颜色，并返回给主窗口的编辑对话框；
4. 实现大致过程
5. 首先实现主窗口和编辑对话框的布局；
6. 分别在两个窗口中加入需要的信号和槽；
7. 完成窗口内部控件以及两窗口之间的通信；
8. 详细步骤
9. 主窗口
10. 头文件

在 MainWindow 的头文件中引用头文件（Qt中使用一个类，就要引用这个类的头文件）：

#include <Qlabel>

#include <QPixmap>

#include <QImage>

#include <QMenuBar>

#include <QMenu>

#include <QAction>

#include <QFileDialog>

#include <QString>

#include <QPoint>

#include <QDebug>

1. 布局

在 MainWindow 的定义中加入

*private*:

QLabel \*label;

QImage \*image; *//* *全局image*

QMenuBar \*menubar;

QMenu \*fileMenu;

QMenu \*editMenu;

QAction \*openAction;

QAction \*rectAction;

然后在MainWindow的构造函数中加入

label = *new* QLabel(*this*);

menubar = *this*->menuBar();

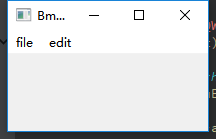
fileMenu = menubar->addMenu("file");

editMenu = menubar->addMenu("edit");

openAction = fileMenu->addAction("openfile");

rectAction = editMenu->addAction("rect");

这样，主窗口的空间布局就完成了。效果如下：



1. 信号

在MianWindow的定义中加入 freshSignal 信号，该信号用于通知窗口需要更新label上的图片内容：

*signals*:

void **freshSignal**();

1. 槽函数

在MainWIndow的定义中加入下列槽函数：

*private* *slots*:

void **openFileSlot**(); // 打开文件

void **freshSlot**(); // 刷新label上的图片

void **editDialogSlot**(); // 打开编辑对话框

void **rectChangeSlot**(QPoint,QPoint,QColor); // 修改制定rect中的像素

它们的实现如下：

void MainWindow::**openFileSlot**()

{

QString path = QFileDialog::getOpenFileName(

*this*,

"文件对话框",

"../",*//上一级路径*

"Image(\*.bmp \*.jpg \*.png)"

);

image = *new* QImage(path);

*emit* freshSignal();

}

void MainWindow::**freshSlot**()

{

label->setPixmap(QPixmap::fromImage(\*image));

label->resize(image->size());

}

void MainWindow::**editDialogSlot**()

{

editDialog->show();

}

void MainWindow::**rectChangeSlot**(QPoint startPoint,QPoint endPoint,QColor color)

{

*//* *Qt* *打开的* *bmp、jpg* *图像格式为* *Format\_RGB32，在内存中的顺序为* *B* *G* *R* *0*

*//* *打开的* *png* *图像的格式为* *Format\_ARGB32，在内存中的顺序为* *B* *G* *R* *A*

unsigned char \*scrdata = image->bits();

int width = image->width();

int height = image->height();

int bytesPerLine = image->bytesPerLine();*//图像每行字节对齐*

unsigned char \*dstdata = *new* unsigned char[bytesPerLine\*height];*//存储处理后的数据*

int r = color.red();

int g = color.green();

int b = color.blue();

*for*(int i=0;i<height;i++)

*for*(int j=0;j<width;j++)

{

*if*((i>=startPoint.x())&&i<endPoint.x()&&

(j>=startPoint.y())&&j<endPoint.y())

{

dstdata[i\*bytesPerLine+j\*4] = b;

dstdata[i\*bytesPerLine+j\*4+1]= g;

dstdata[i\*bytesPerLine+j\*4+2]= r;

}

*else*{

dstdata[i\*bytesPerLine+j\*4] = scrdata[0];

dstdata[i\*bytesPerLine+j\*4+1]= scrdata[1];

dstdata[i\*bytesPerLine+j\*4+2]= scrdata[2];

}

scrdata+=4;

}

image= *new* QImage(dstdata,width,height,bytesPerLine,QImage::*Format\_RGB32*);

*emit* freshSignal();

}

1. 连接信号与槽

在MainWindow的实现函数中进行连接：

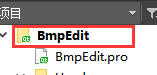
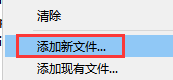
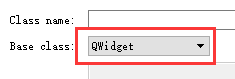
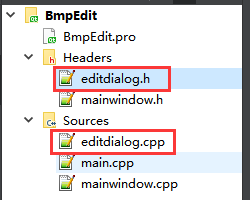
connect(openAction,*SIGNAL*(triggered()),*this*,*SLOT*(openFileSlot()));

connect(*this*,*SIGNAL*(freshSignal()),*this*,*SLOT*(freshSlot()));

connect(rectAction,*SIGNAL*(triggered()),*this*,*SLOT*(editDialogSlot()));

这样，MainWindow 的实现就基本完成

1. 编辑对话框
2. 创建编辑对话框类

编辑对话框类需要自己创建，步骤：右键项目文件夹->添加新文件->选择C++类->选择基类为QWidget->命名为EditDialog->完成。这样工程就会自动添加两个新的文件，编辑对话框创建完成。

1. 头文件

在 MainWindow 的 cpp 文件中加入头文件（Qt中使用一个类，就要引用这个类的头文件）：

#include <QLabel>

#include <QGridLayout>

#include <QPushButton>

#include <QLineEdit>

#include <QPoint>

1. 布局

在 EditDialog 的定义中加入

*public*:

QGridLayout \*layout;

QLabel \*startLabel;

QLabel \*endLabel;

QLineEdit \*x1Text;

QLineEdit \*y1Text;

QLineEdit \*x2Text;

QLineEdit \*y2Text;

QLabel \*rLabel;

QLabel \*gLabel;

QLabel \*bLabel;

QLineEdit \*rText;

QLineEdit \*gText;

QLineEdit \*bText;

QPushButton \*okButton;

然后在 EditDialog 的构造函数中加入

layout = *new* QGridLayout(*this*);

startLabel = *new* QLabel("start",*this*);

endLabel = *new* QLabel("end",*this*);

x1Text = *new* QLineEdit("0",*this*);

y1Text = *new* QLineEdit("0",*this*);

x2Text = *new* QLineEdit("100",*this*);

y2Text = *new* QLineEdit("100",*this*);

rLabel = *new* QLabel("R:",*this*);

gLabel = *new* QLabel("G:",*this*);

bLabel = *new* QLabel("B:",*this*);

rText = *new* QLineEdit("0",*this*);

gText = *new* QLineEdit("0",*this*);

bText = *new* QLineEdit("0",*this*);

okButton = *new* QPushButton("OK",*this*);

layout->addWidget(startLabel,0,0,1,1); // 添加布局

layout->addWidget(x1Text,0,1,1,1);

layout->addWidget(y1Text,0,2,1,1);

layout->addWidget(endLabel,1,0,1,1);

layout->addWidget(x2Text,1,1,1,1);

layout->addWidget(y2Text,1,2,1,1);

layout->addWidget(rLabel,2,0,1,1);

layout->addWidget(gLabel,3,0,1,1);

layout->addWidget(bLabel,4,0,1,1);

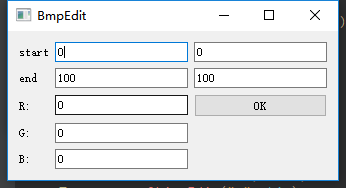
layout->addWidget(rText,2,1,1,1);

layout->addWidget(gText,3,1,1,1);

layout->addWidget(bText,4,1,1,1);

layout->addWidget(okButton,2,2,1,1);

这样 EditDialog 的布局完成，效果如下



1. 信号

在 EditDialog 的定义中加入如下信号，该信号用于向 MainWindow 返回需要涂色的矩形起点和终点，并告诉 MainWIndow 执行涂色操作。

*signals*:

void **resultSignal**(QPoint,QPoint,QColor);

1. 槽函数

在 EditDialog 的定义中加入如下函数

*public* *slots*:

void **okButtonSlot**(); // 处理案件操作

槽函数的实现：

void EditDialog::**okButtonSlot**()

{

int x1 = x1Text->text().toInt();

int y1 = y1Text->text().toInt();

int x2 = x2Text->text().toInt();

int y2 = y2Text->text().toInt();

int r = rText->text().toInt();

int g = gText->text().toInt();

int b = bText->text().toInt();

QPoint startPoint = QPoint(x1,y1);

QPoint endPoint = QPoint(x2,y2);

QColor color = QColor(r,g,b);

*emit* resultSignal(startPoint,endPoint,color);

*this*->hide();

}

1. 连接信号与槽

在 EditDialog 的构造函数中加入

connect(okButton,*SIGNAL*(clicked()),*this*,*SLOT*(okButtonSlot()));

1. 主窗口与编辑对话框的连接
2. 在 MainWindow 的头文件中引用头文件

#include "editdialog.h"

1. 在 MainWindow 的构造函数中加入

editDialog = *new* EditDialog();

1. 在 MainWIndow 的构造函数中加入

connect(editDialog,*SIGNAL*(resultSignal(QPoint,QPoint,QColor)),*this*,*SLOT*(rectChangeSlot(QPoint,QPoint,QColor)));

至此，整个程序就编写完成，效果图如下：

