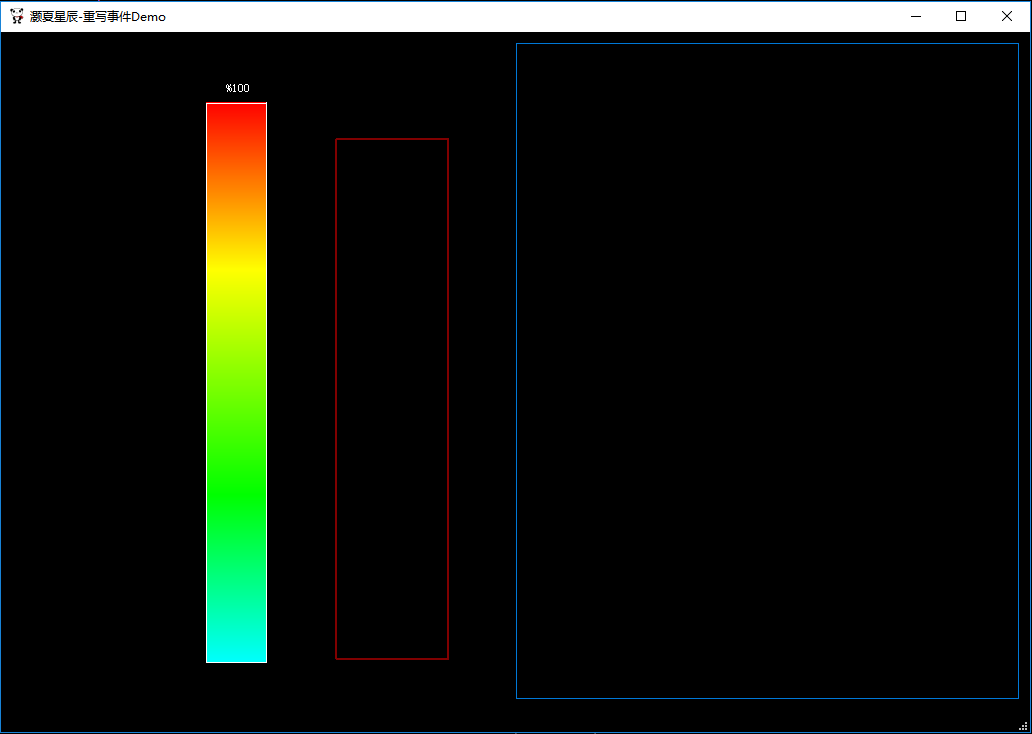
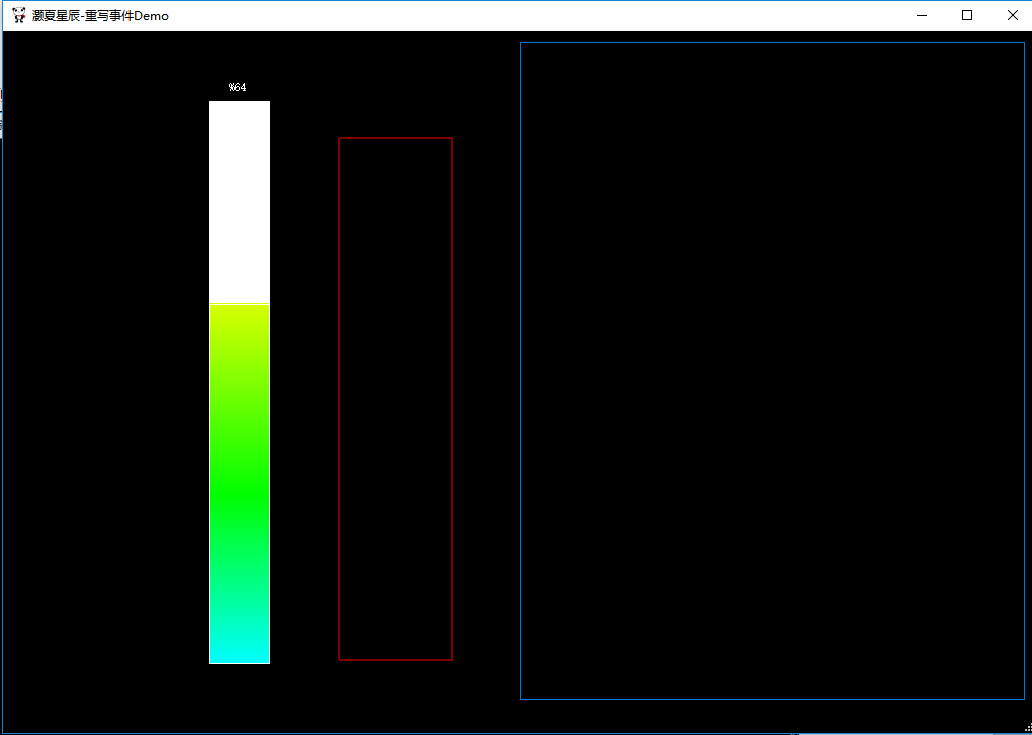
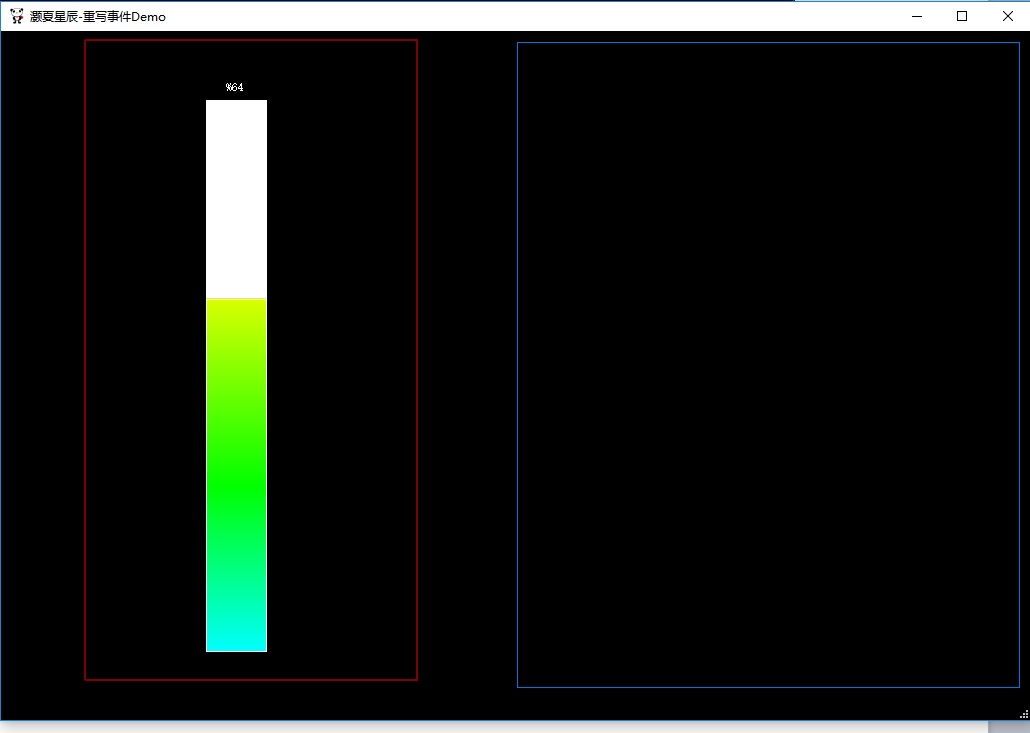
重写事件例子具体演示效果



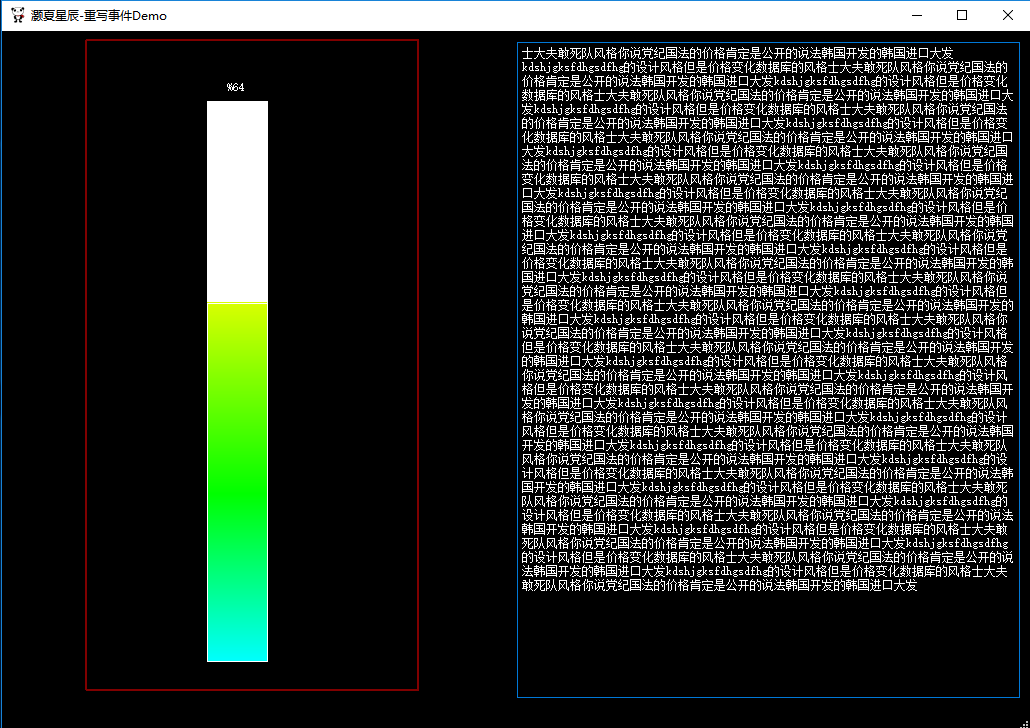
滑动滚轮可以使矩形区域改变，使用的是滚轮滚动事件和画图事件如下图



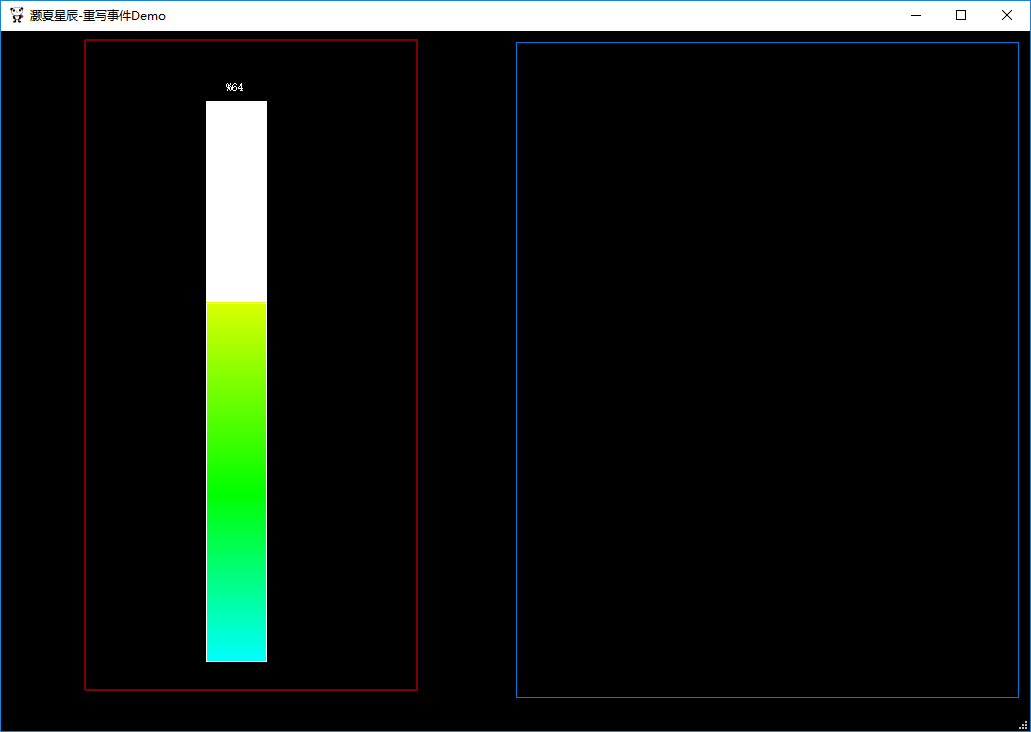
使用鼠标的点击滑动和释放三个响应事件可以画出任意大小的矩形域



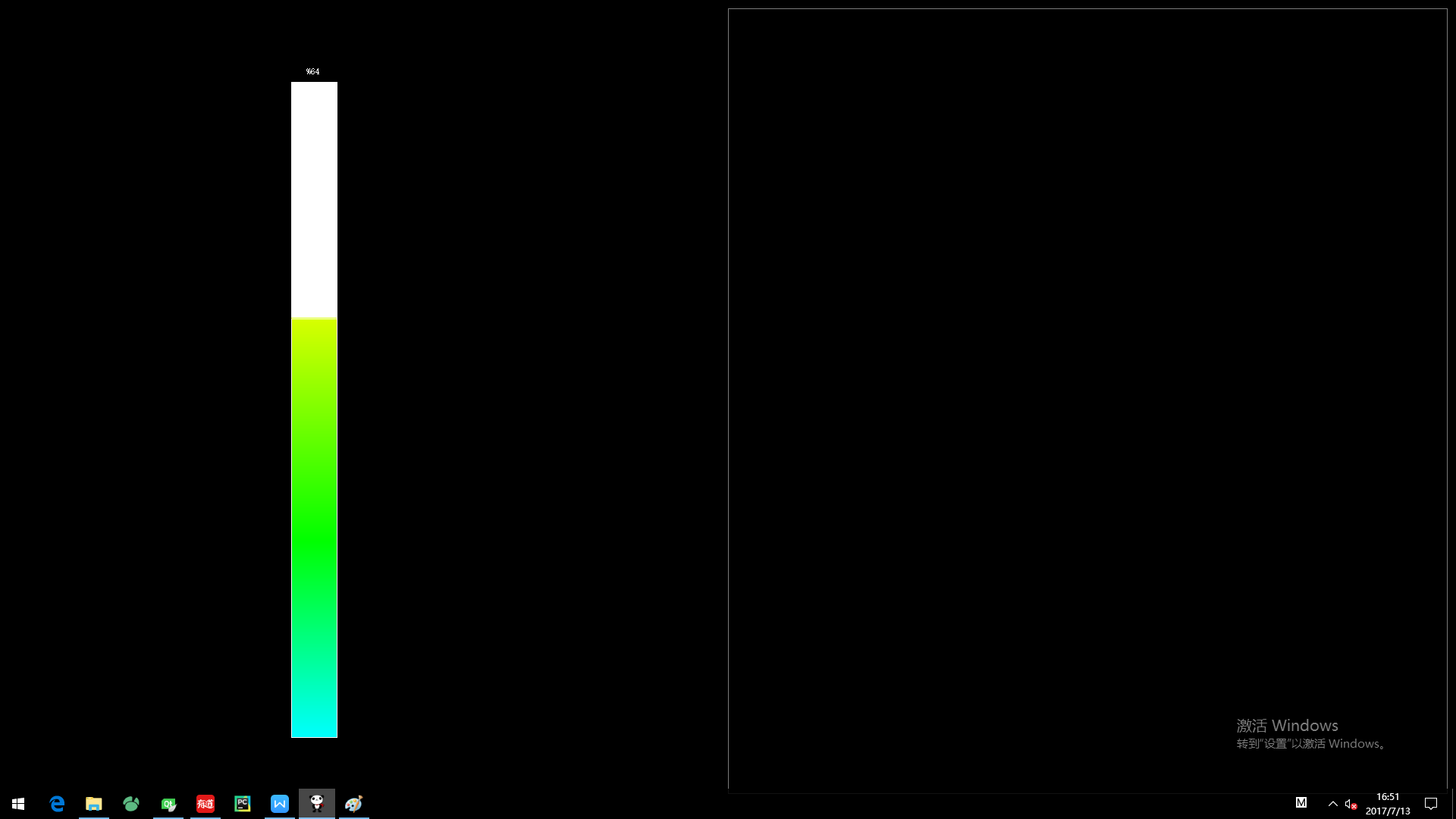
使用拖入事件和释放事件，可将文本内容读取到文本框



使用键盘事件，点击F5键，可将文本内容清空

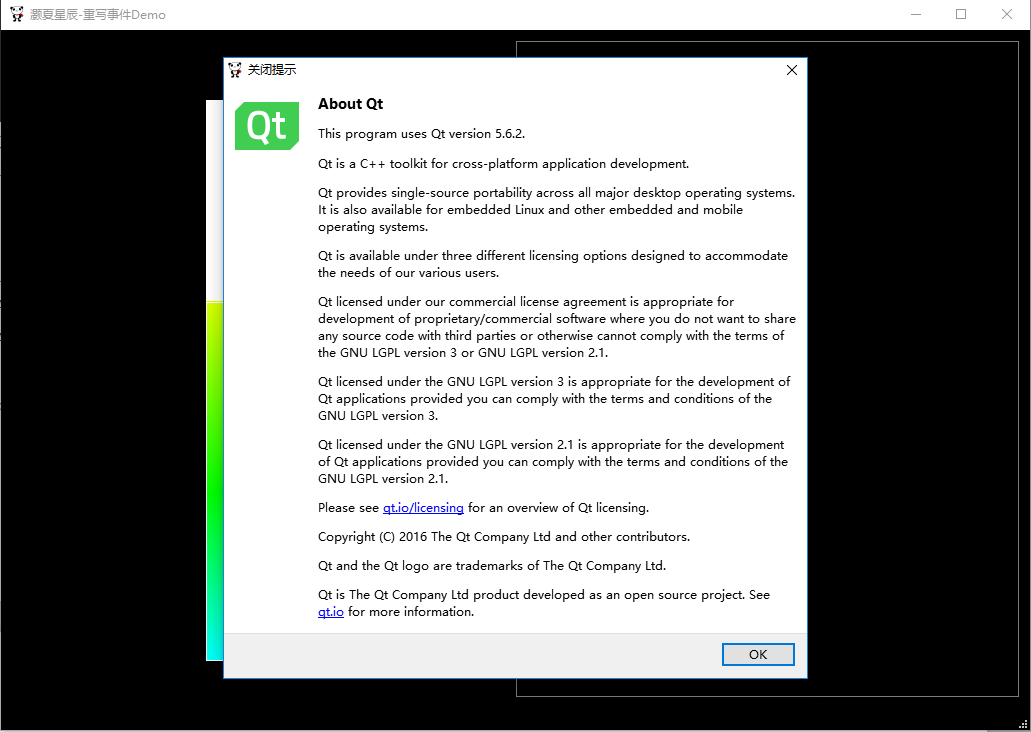


使用鼠标双击事件可将程序满屏，双击满屏，再双击退出



使用鼠标进入事件和离开事件改变鼠标样式进入是手型鼠标，离开当前页面是箭头鼠标

使用关闭事件，当点击关闭按钮，弹出关于Qt的对话框如下图



下面是源代码：

下面是MainWindow.h文件内容

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QEvent>

#include <QPaintEvent>

#include <QWheelEvent>

#include <QMouseEvent>

#include <QDragEnterEvent>

const static QPoint POINT = QPoint(8, 31);

namespace Ui {

class MainWindow;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能:事件响应例子

创建人:YYC

创建时间:2017-7-13

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private:

Ui::MainWindow \*ui;

void *closeEvent*(QCloseEvent\* event); //关闭事件

void *mousePressEvent*(QMouseEvent\*event); //鼠标点击

void *mouseReleaseEvent*(QMouseEvent \*event); //鼠标释放

void *mouseMoveEvent*(QMouseEvent \*event); //鼠标移动

void *mouseDoubleClickEvent*(QMouseEvent \*event); //鼠标双击事件

void *wheelEvent*(QWheelEvent\*event); //滚轮滚动事件

void *paintEvent*(QPaintEvent \* event); //画图事件

void *enterEvent*(QEvent\*event); //鼠标进入事件

void *leaveEvent*(QEvent\*event); //鼠标离开事件

void *keyPressEvent*(QKeyEvent\*event); //键盘响应事件

void *dragEnterEvent*(QDragEnterEvent \* event); //拖入事件

void *dropEvent*(QDropEvent\*event); //释放事件

private:

float sliderValue; //进度条的值

QPoint startRectPoint; //点坐标

QPoint endRectPoint; //点坐标

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

#include <QPainter>

#include <QMimeData>

#include <QRect>

#include <QDebug>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 构造函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent), sliderValue(0),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

//设置背景色为黑色

this->setPalette(QPalette(Qt::*black*));

this->setAutoFillBackground(true);

this->setWindowTitle("灏夏星辰-重写事件Demo");

ui->textEdit->setAcceptDrops(false);

this->setAcceptDrops(true);

startRectPoint = QPoint(0, 0);

endRectPoint = QPoint(0, 0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 析构函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

MainWindow::~*MainWindow*()

{

delete ui;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 退出程序事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*closeEvent*(QCloseEvent \*event)

{

QMessageBox::aboutQt(this, tr("关闭提示"));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 鼠标点击事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*mousePressEvent*(QMouseEvent \*event)

{

if (event->button() == Qt::*LeftButton*)

{

startRectPoint = event->globalPos() - pos() - POINT;

endRectPoint = event->globalPos() - pos() - POINT;

update();

}

else if(event->button() == Qt::*RightButton*)

{

startRectPoint = QPoint(0, 0);

endRectPoint = QPoint(0, 0);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 鼠标释放事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*mouseReleaseEvent*(QMouseEvent \*event)

{

if (event->button() == Qt::*LeftButton*)

{

endRectPoint = event->globalPos() - pos() - POINT;

update();

}

else if(event->button() == Qt::*RightButton*)

{

startRectPoint = QPoint(0, 0);

endRectPoint = QPoint(0, 0);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 鼠标移动事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*mouseMoveEvent*(QMouseEvent \*event)

{

if (event->buttons() & Qt::*LeftButton*)

{

endRectPoint = event->globalPos() - pos() - POINT;

update();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 鼠标双击事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*mouseDoubleClickEvent*(QMouseEvent \*event)

{

static int tempCount = 0;

if(tempCount % 2 == 0)

{

this->showFullScreen();

}

else

{

this->showNormal();

}

tempCount ++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 滚轮滚动事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*wheelEvent*(QWheelEvent \*event)

{

//鼠标向上滚动

if (event->delta() > 0)

{

if (sliderValue > 0.99) return;

sliderValue += 0.02;

}

//鼠标向下滚动

else

{

if (sliderValue <= 0.01) return;

sliderValue -= 0.02;

}

update();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 画图事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*paintEvent*(QPaintEvent \*event)

{

QPainter painter(this);

float sliderMaxValue = height() \* 0.9;

float sliderHigh = -height() \* 0.8; //进度条高度

const int sliderWidth = 60; //进度条宽度

//将坐标系的该点变成原点坐标

painter.translate(width() \* 0.2, sliderMaxValue);

//设置画笔为红色

QPen pen(Qt::*white*, 1, Qt::*SolidLine*, Qt::*RoundCap*, Qt::*RoundJoin*);

painter.setPen(pen);

//画下面直线

painter.drawLine(0, 0, sliderWidth, 0);

//画左边直线

painter.drawLine(0, 0, 0, sliderHigh);

//画右边直线

painter.drawLine(sliderWidth, 0, sliderWidth, sliderHigh);

//画上面直线线

painter.drawLine(0, sliderHigh, sliderWidth, sliderHigh);

//画进度值

if(sliderValue < 0.01)

{

painter.drawText(sliderWidth - 40, sliderHigh - 10,tr("%0"));

}

else

{

painter.drawText(sliderWidth - 40, sliderHigh - 10,tr("%%1").arg(sliderValue \* 100));

}

//将矩形填充为白色

QPolygon polyGon;

polyGon<<QPoint(sliderWidth, 0);

polyGon<<QPoint(sliderWidth, sliderHigh);

polyGon<<QPoint(0, sliderHigh);

polyGon<<QPoint(0, 0);

painter.setBrush(Qt::*white*);

painter.drawConvexPolygon(polyGon);

//设置渐变

QLinearGradient linerGradient = QLinearGradient(0, 0, 0, sliderHigh);

linerGradient.setColorAt(0.0, Qt::*cyan*);

linerGradient.setColorAt(0.3, Qt::*green*);

linerGradient.setColorAt(0.7, Qt::*yellow*);

linerGradient.setColorAt(1.0, Qt::*red*);

QBrush brush(linerGradient);

painter.setBrush(brush);

painter.drawRect(0, 0, sliderWidth, sliderHigh \* sliderValue);

int width = endRectPoint.x() - startRectPoint.x(); //画的矩形宽度

int height = endRectPoint.y() - startRectPoint.y(); //画的矩形高度

QPainter painterRect(this);

painterRect.setPen(QPen(Qt::*darkRed*, 2));

QRect rect(startRectPoint.x(), startRectPoint.y(), width, height);

painterRect.drawRect(rect);

painterRect.end();

painter.end();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 进入事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*enterEvent*(QEvent \*event)

{

this->setCursor(Qt::*PointingHandCursor*);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 离开事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*leaveEvent*(QEvent \*event)

{

this->setCursor(Qt::*ArrowCursor*);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 键盘事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*keyPressEvent*(QKeyEvent \*event)

{

if(event->key() == Qt::*Key\_F5*)

{

ui->textEdit->clear();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 拖入事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*dragEnterEvent*(QDragEnterEvent \*event)

{

event->acceptProposedAction();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 释放事件 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::*dropEvent*(QDropEvent \*event)

{

QList<QUrl>urls = event->mimeData()->urls();

if (urls.isEmpty()) return;

QFile file(urls.first().toLocalFile());

if(!file.*open*(QFile::*ReadOnly* | QFile::*Text*)) return;

ui->textEdit->setText(QString::fromLocal8Bit(file.readAll()));

}

使用事件过滤器

重写eventFilter(Qobject\*watched,Qevent\*event)实现在控件上面实现以上事件

ui->widgetPaint->installEventFilter(this);

bool MainWindow::*eventFilter*(QObject \*watched, QEvent \*event)

{

if(watched==ui->widget && event->type()==QEvent::*Paint*)

{

paintImage();

}

if(watched==ui->widget && event->type()==QEvent::*MouseMove*)

{

mouseMove();

}

if(watched==ui->widget && event->type()==QEvent::*MouseButtonPress*)

{

mousePress();

}

if(watched==ui->widget && event->type()==QEvent::*MouseButtonRelease*)

{

mouseRelease();

}

if(watched==ui->widget && event->type()==QEvent::*Paint*)

{

paintImage();

}

if(watched==ui->widget && event->type()==QEvent::*MouseButtonDblClick*)

{

mouseDoubleClick();

}

if(watched==widget && event->type()==QEvent::*Wheel*)

{

QWheelEvent\* e=static\_cast<QWheelEvent\*>(event);

if(e->delta()>0){}else{}

}

return QWidget::*eventFilter*(watched,event);

}

要现在构造函数中对你所要画图的控件注册监视对象

ui->widget->installEventFilter(this);//注册监视对象