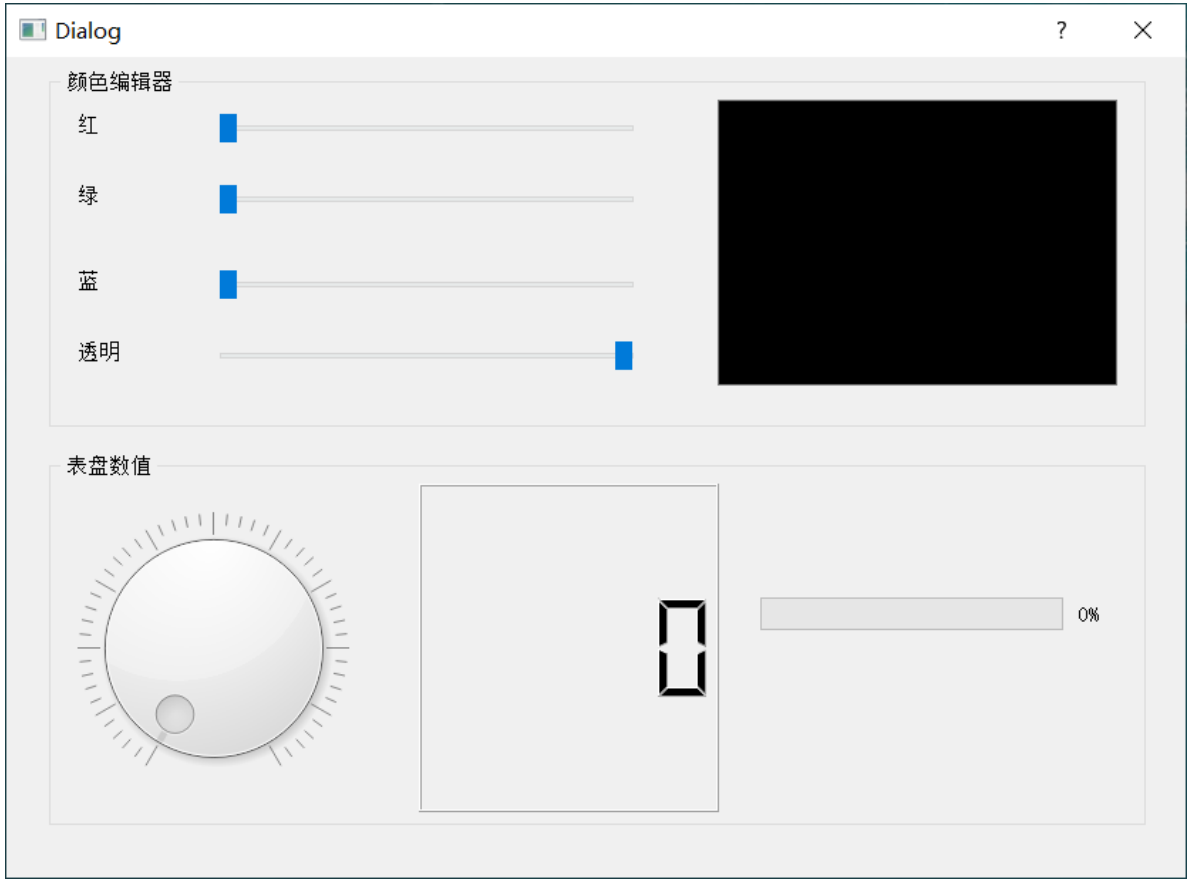
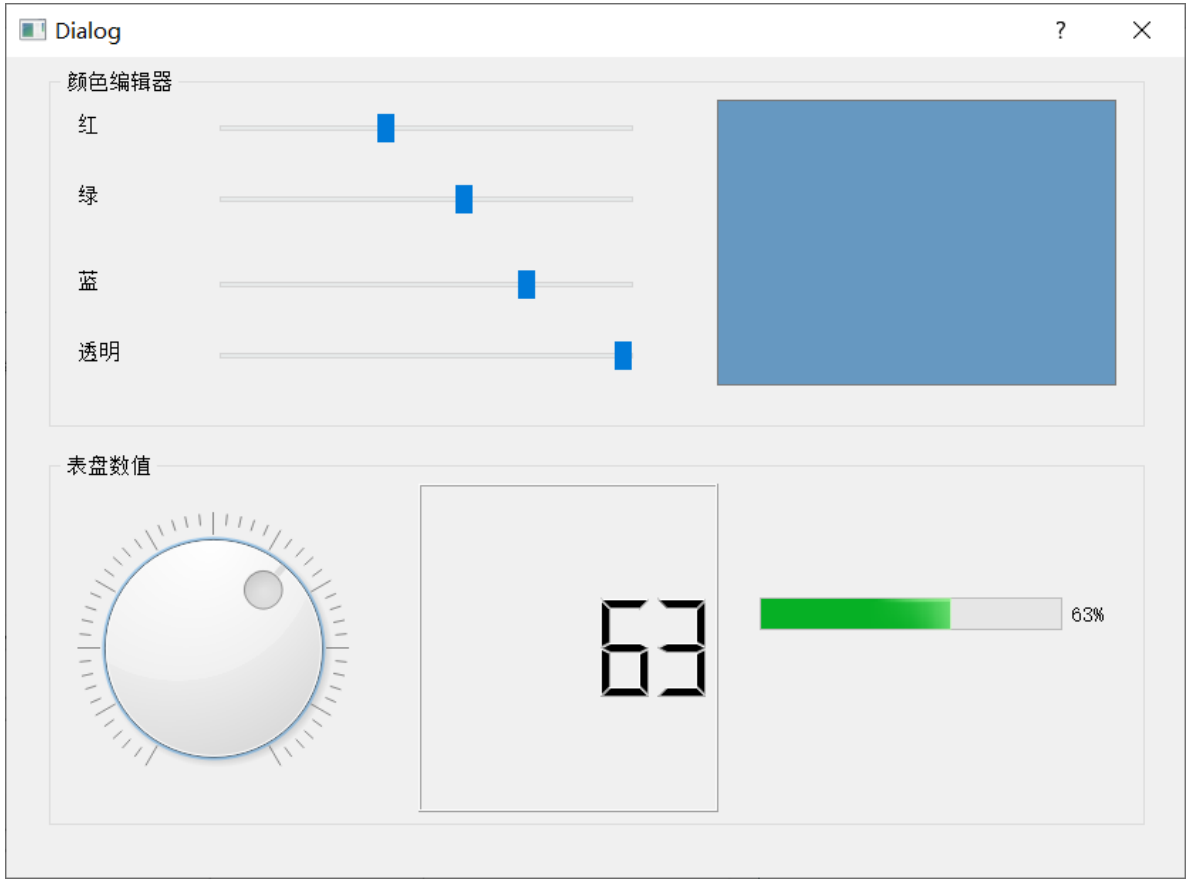
qt实战之颜色控制器

颜色控制器





案例知识点

- 1. Qslider滑动条的使用
- 2. QProgressBar进度条的使用
- 3. Qdia表盘式数值输入组件
- 4. QLCDNumber：　模仿LCD数字的显示组件

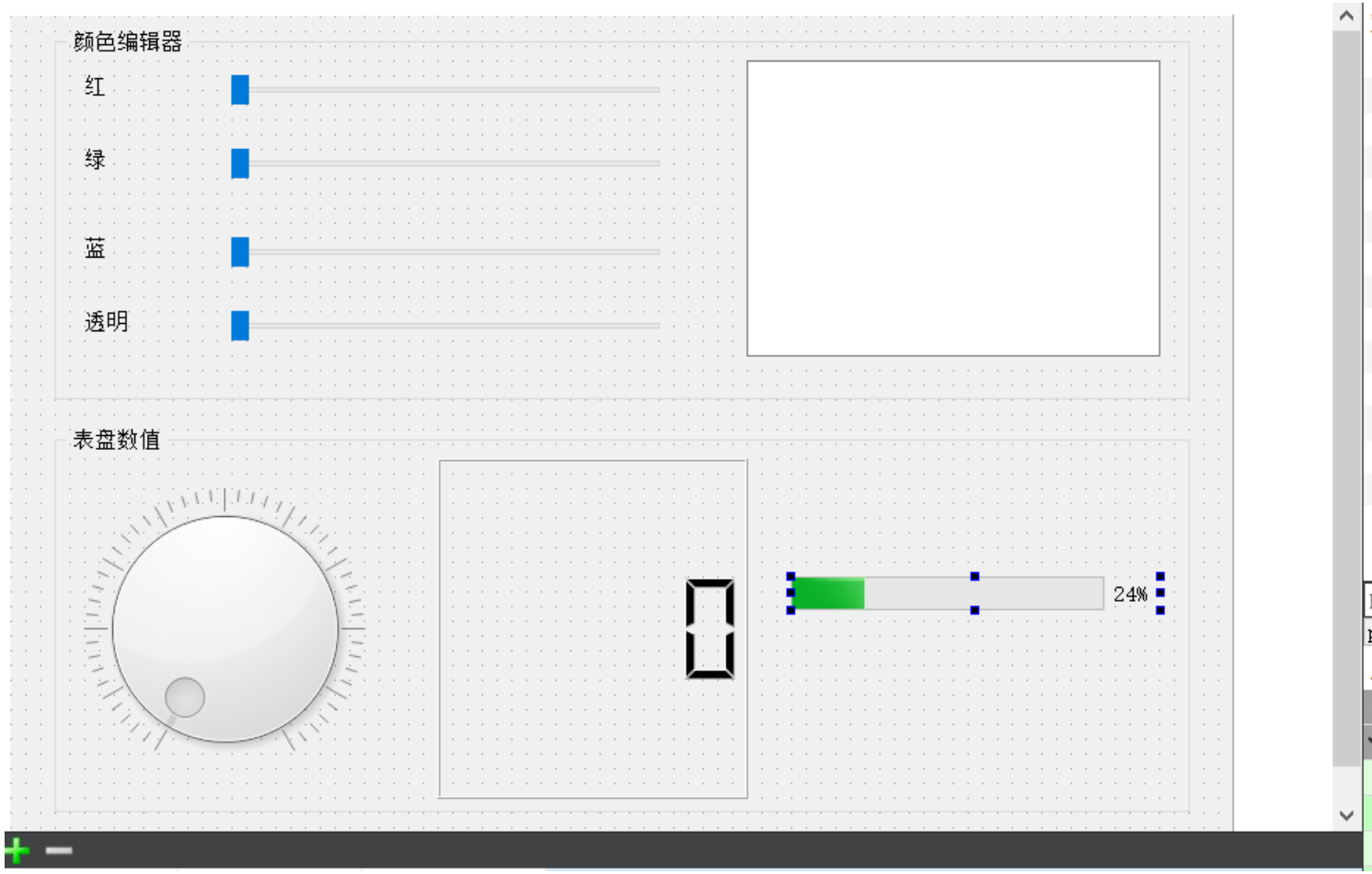
组件主要的功能和属性

QSlider、QScrollBar 和 Qdial 3 个组件都从 QAbstractSlider 继承而来，有一些共有的属性。QSlider 是滑动的标尺型组件，滑动标尺上的一个滑块可以改变值。

基类 QAbstractSlider 的主要属性包括以下几种。

- minimum、maximum：设置输入范围的最小值和最大值，例如，用红、绿、蓝配色时，每种基色的大小范围是 0~255，所以设置 minimum 为 0，maximum 为 255。
- singleStep：单步长，拖动标尺上的滑块，或按下左/右光标键时的最小变化数值。
- pageStep：在 Slider 上输入焦点，按 PgUp 或 PgDn 键时变化的数值。
- value：组件的当前值，拖动滑块时自动改变此值，并限定在 minimum 和 maximum 定义的范围之内。
- sliderPosition：滑块的位置，若 tracking 属性设置为 true，sliderPosition 就等于 value。
- tracking：sliderPosition 是否等同于 value，如果 tracking=true，改变 value 时也同时改变 sliderPosition。
- orientation：Slider 的方向，可以设置为水平或垂直。方向参数是 Qt 的枚举类型 enum Qt::Orientation，取值包括以下两种。
  - Qt::Horizontal 水平方向
  - Qt::Vertical 垂直方向
- invertedAppearance：显示方式是否反向，invertedAppearance=false 时，水平的 Slider 由左向右数值增大，否则反过来。
- invertedControls：反向按键控制，若 invertedControls=true，则按下 PgUp 或 PgDn 按键时调整数值的反向相反。

UI设计



对象	类
Dialog	QDialog
groupBox	QGroupBox
horizontalSlider_aplan	QSlider
horizontalSlider_blue	QSlider
horizontalSlider_green	QSlider
horizontalSlider_red	QSlider
label	QLabel
label_2	QLabel
label_3	QLabel
label_4	QLabel
textEdit_color	QTextEdit
groupBox_2	QGroupBox
dialvalue	QDial
lcdNumber	QLCDNumber
progressBar	QProgressBar

Filter

progressBar : QProgressBar

属性	值
QWidget	
QProgressBar	
minimum	0
maximum	99
value	24

颜色编辑的实现

dialog.h — Dialog

```h
#ifndef DIALOG_H
#define DIALOG_H

#include <QDialog>

namespace Ui {
class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(QWidget *parent = nullptr);
    ~Dialog();

private:
    Ui::Dialog *ui;

private slots:
    void on_slider_valueChanges(int value);
    void on_dialvalue_valueChanged(int value);
};

#endif // DIALOG_H
```

dialog.cpp — Dialog::~Dialog() -> void

```cpp
#include "dialog.h"
#include "ui_dialog.h"

Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);

    QObject::connect(ui->horizontalSlider_red, SIGNAL(valueChanged(int)), this, SLOT(on_slider_valueChanges(int)));
    QObject::connect(ui->horizontalSlider_green, SIGNAL(valueChanged(int)), this, SLOT(on_slider_valueChanges(int)));
    QObject::connect(ui->horizontalSlider_blue, SIGNAL(valueChanged(int)), this, SLOT(on_slider_valueChanges(int)));
    QObject::connect(ui->horizontalSlider_aplan, SIGNAL(valueChanged(int)), this, SLOT(on_slider_valueChanges(int)));

    ui->horizontalSlider_aplan->setValue(255);
    ui->progressBar->setValue(0);
    on_slider_valueChanges(0);
}

Dialog::~Dialog()
{
    delete ui;
}
```

事件关联

颜色编辑的实现

```
24
25  void Dialog::on_slider_valueChanges(int value) {
26      Q_UNUSED(value);
27      QColor color;
28      int red = ui->horizontalSlider_red->value();
29      int green = ui->horizontalSlider_green->value();
30      int blue = ui->horizontalSlider_blue->value();
31      int aplan = ui->horizontalSlider_aplan->value();
32      color.setRgb(red,green,blue,aplan);
33
34      QPalette palette = ui->textEdit_color->palette();
35      palette.setColor(QPalette::Base, color);
36      ui->textEdit_color->setPalette(palette);
37  }
38
39  void Dialog::on_dialvalue_valueChanged(int value)
40  {
41      ui->lcdNumber->display(value);
42      ui->progressBar->setValue(value);
43  }
44
```

将获得的数值设置为颜色对象

控件联动