Глава 9

Цвета: проект COLORS

Проект COLORS знакомит с классом, предназначенным для работы с цветом (QColor), и с компонен­том, обеспечивающим прокрутку данных (QSlider). Кроме того, описывается способ доступа к ком­понентам через клавиши-ускорители связанных с ними меток, и рассматриваются варианты привязки компонентов к границам формы (anchoring).

9.1. Определение цвета как комбинации четырех цветовых составляющих. Ползунки

После создания проекта COLORS разместите в форме Form компонент типа **Form Layout** (formLayout) и разместите в нем виджет типа **Horizontal Slider** (horizontalSlider) во втором столбце и настройте свойства формы и компонента horizontalSlider (листинг 9.1; указанный в листинге ком­понент label\_6 будет размещен в форме позднее).

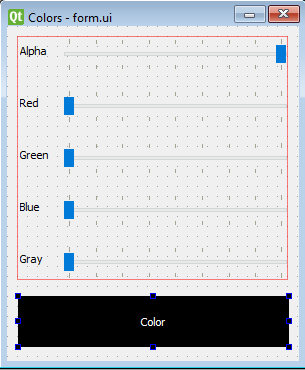
Не снимая выделения с компонента horizontalSlider, нажмите клавиатурную комбинацию <Ctrl>+<C>, а затем четыре раза – комбинацию <Ctrl>+<V>. В результате в форму будут добавлены еще четыре компонента QSlider с именами horizontalSlider\_2, ..., horizontalSlider\_5, причем для всех этих компонентов значения свойств будут совпадать со значениями соответствующих свойств компонента horizontalSlider (заметим, что подобное копирование компонентов можно выполнять и с помощью контекстного меню формы). Расположите все компоненты сверху вниз во втором столбце formLayout (рис. 9.1).

Для компонента horizontalSlider дополнительно установите свойство value равным **255** (у осталь­ных компонентов QSlider это свойство должно остаться равным **0**).

После этого разместите в первом столбце formLayout пять меток (label, ..., label\_5) и установите их свойства text равными соответственно **Alpha**, **Red**, **Green**, **Blue**, **Gray**.

Наконец, разместите в нижней части формы еще одну метку label\_6. Настройте свойства метки label\_6 (листинг 9.1).

Определите обработчик сигнала valueChanged(int) для компонента horizontalSlider (листинг 9.2), после чего свяжите в конструкторе формы сигнал valueChanged(int) компонентов horizontalSlider\_2 – horizontalSlider\_4 (обработчик сигнала valueChanged(int) для horizontalSlider\_5 будет добавлен позднее, в *разд*. 9.3).



**Рис. 9.1.** Вид формы Form для проекта COLORS

**Листинг 9.1. Настройка свойств формы Form и ее компонентов**

Form: windowTitle = **Colors**

slider: maximum = **255**, tickPosition = **TicksBothSides**, tickInterval = **32**

label\_6: text = **Color**, alignment.Horizontal **= AlignHCenter,**

styleSheet = **background-color: black; color: white;**

**Листинг 9.2. Слот on\_horizontalSlider\_valueChanged**

void Form::on\_horizontalSlider\_valueChanged()

{

**ui->label\_6->setStyleSheet(QString("background-color: rgba(%1,%2,%3,%4)")**

**.arg(ui->horizontalSlider\_2->value()).arg(ui->horizontalSlider\_3->value())**

**.arg(ui->horizontalSlider\_4->value()).arg(ui->horizontalSlider->value() / 255.0));**

}

**Результат.** Цвет фона метки label\_6 определяется как комбинация четырех цветовых составляющих, а именно прозрачности (Alpha) и интенсивности трех базовых цветов: красного (Red), зеленого (Green) и синего (Blue). Каждая из первых трех цветовых составляющих может меняться в пределах от 0 до 255; прозрачность меняется в пределах от 0 до 1 (значение 1 для составляющей Alpha соответствует полной непрозрачности). В нашей программе значения цветовых составляющих задаются положением соответствующего компонента QSlider (нижний компонент horizontalSlider\_5 пока не использу­ется).

Комментарий

Компонент QSlider, называемый ползунком, удобно использовать в ситуациях, когда требуется задать параметр, принимающий значения целого типа из некоторого (не слишком большого) диапазона. При настройке компонента horizontalSlider в нашем проекте мы установили значения следующих его свойств: singleStep (шаг ползунка при нажатии клавиш со стрелками, по умолчанию равен **1**), maximum (максимальное допустимое значение), tickInterval (частота следования штрихов), tickPosition (стиль расположения штрихов), а также value (текущее значение ползунка). Другие свойства изменять не потребовалось, так как нас устроили их значения по умолчанию. Перечислим некоторые из таких свойств: pageStep (шаг при нажатии клавиш <PgUp> и <PgDn>), minimum (мини­мальное допустимое значение, равное по умолчанию **0**) и orientation (определяет ориентацию пол­зунка; по умолчанию ориентация вертикальная: Vertical, однако при выборе **Horizontal Slider** меня­ется автоматически). Заметим, что изменение ползунка на величину pageStep выполняется не только при нажатии клавиш <PgUp> или <PgDn>, но и при щелчке мышью слева.

9.2. Инвертирование цветов и вывод цветовых констант

Добавьте новые операторы в слот on\_horizontalSlider\_valueChanged (листинг 9.3).

**Листинг 9.3. Новый вариант слота on\_horizontalSlider\_valueChanged**

void Form::on\_horizontalSlider\_valueChanged()

{

**QColor c = QColor(ui->horizontalSlider\_2->value(), ui->horizontalSlider\_3->value(),**

**ui->horizontalSlider\_4->value(), ui->horizontalSlider->value());**

**ui->label\_6->setStyleSheet(QString("background-color: rgba(%1,%2,%3,%4); color:**

**rgb(%5,%6,%7)")**

**.arg(c.red()).arg(c.green()).arg(c.blue()).arg(c.alpha() / 255.0)**

**.arg(255 ^ c.red()).arg(255 ^ c.green()).arg(255 ^ c.blue()));**

**ui->label\_6->setText(QString("%1%2")**

**.arg(QString::number(c.alpha(), 16).toUpper())**

**.arg(c.name().remove(0, 1).toUpper()));**

}

Результат. Числовое значение текущего цвета в формате ARGB (Alpha, Red, Green, Blue) отображается на панели в виде шестнадцатеричного числа. При этом для каждой цветовой составляющей отводится по два знака, а буквы A—F (соответствующие шестнадцатеричным цифрам от 10 до 15) изображаются в верхнем регистре. Например, значение цвета Maroon (непрозрачный темно-красный цвет интенсив­ности 128) имеет вид **FF800000**. Цвет текста — непрозрачный и *инверсный* по отношению к цвету фона панели. См. также комментарии 1 и 2.

**Недочет 1.** При запуске программы метка label\_6 содержит текст **Color**, а не числовое значение не­прозрачного черного цвета.

**Исправление.** В конструктор класса Form добавьте оператор:

on\_horizontalSlider\_valueChanged();

**Результат.** Теперь метод on\_horizontalSlider\_valueChanged() вызывается в момент создания формы, что обеспечивает правильную настройку внешнего вида метки label\_6.

***ПРИМЕЧАНИЕ***

Отмеченный недочет можно было бы исправить, просто положив значение свойства text метки label\_6 равным **FF000000** (числовое значение непрозрачного черного цвета). Однако использованный вариант исправления явля­ется более гибким, поскольку позволяет правильно отображать начальный вид метки при любых изменениях метода on\_horizontalSlider\_valueChanged(), которые могут быть сделаны впоследствии *(см. разд. 9.4)*.

**Недочет 2.** Если прозрачность имеет значение, меньшее 16, то в метке label\_6 отображается менее 8 цифр (в частности, при полностью прозрачном черном цвете метка будет содержать единственную цифру 0). Подобный способ представления цвета является не вполне естественным; логичнее *всегда* отображать шестнадцатеричное число с 8 знаками (по два знака на каждую цветовую составляющую).

**Исправление.** В методе on\_horizontalSlider\_valueChanged() замените последний оператор на сле­дующий:

**ui->label\_6->setText(QString("%1%2")**

**.arg(QString::number(c.alpha(), 16).toUpper(), 2, '0')**

**.arg(c.name().remove(0, 1).toUpper()));**

**Результат.** Теперь цвет выводится в шестнадцатеричном формате с заглавными латинскими буквами и обязательными 8 знаками.

Комментарии

1. Свойства alpha, red, green, blue структуры QColor позволяют получить числовое значение соот­ветствующей цветовой составляющей. Для инвертирования каждого из базовых цветов использо­вана побитовая операция ^ (исключающее ИЛИ).
2. Метод toUpper класса QString преобразует все буквенные символы строки к верхнему регистру и возвращает измененную строку (метод, преобразующий буквенные символы к нижнему регистру, имеет имя toLower). Символы, не являющиеся буквами, не изменяются. Заметим, что методы toUpper и toLower правильно обрабатывают не только латинские, но и русские буквы.

9.3. Отображение оттенков серого цвета

Определите слот on\_horizontalSlider\_5\_valueChanged() (листинг 9.4).

**Листинг 9.4. Слот on\_horizontalSlider\_5\_valueChanged**

void Form::on\_horizontalSlider\_5\_valueChanged()

{

**int value = ui->horizontalSlider\_5->value();**

**ui->horizontalSlider\_2->setValue(value);**

**ui->horizontalSlider\_3->setValue(value);**

**ui->horizontalSlider\_4->setValue(value);**

}

**Результат.** Перемещение ползунка horizontalSlider\_5 обеспечивает синхронное изменение всех трех базовых цветов, давая в итоге различные оттенки серого цвета (значение прозрачности при этом не изменяется).

9.4. Вывод цветовых имен

Добавьте в конструктор формы следующие операторы:

**QFont font = ui->label\_6->font();**

**font.setCapitalization(QFont::Capitalize);**

**ui->label\_6->setFont(font);**

Дополните слот on\_horizontalSlider\_valueChanged() (листинг 9.5).

**Листинг 9.5. Добавление к слоту on\_horizontalSlider\_5\_valueChanged**

**switch (c.alpha())**

**{**

**case 0:**

**ui->label\_6->setText(ui->label\_6->text() + " Transparent");**

**break;**

**case 255:**

**foreach (QString color, QColor::colorNames())**

**if (QColor(color).rgb() == c.rgb())**

**{**

**ui->label\_6->setText(ui->label\_6->text() + " " + color);**

**break;**

**}**

**break;**

**}**

**Результат.** В том случае, когда с текущим цветом связано определенное имя (например, **Black** или **Maroon**), на панели отображается не только числовое значение текущего цвета в шестнадцатеричном формате, но и его имя (первая буква будет заглавной). Если прозрачность имеет значение, равное 0, то рядом с числовым значением цвета выводится текст **Transparent**.

Комментарии

1. Все цвета, имеющие имена (именованные цвета), содержатся в списке строк colorNames класса QColor. К сожалению, поиск по значению RGB в данном списке нет, поэтому приходится прохо­дить для каждого цвета весь список и сравнивать значение интересующего нас цвета со значением именованного цвета.
2. Свойство capitalization класса QFont отвечает за рендеринг текста, к которому применяется дан­ный шрифт, значения данного свойства находятся в перечислении QFont::Capitalization. По умолчанию это свойство имеет значение QFont::MixedCase (рендеринг не производится, сохраня­ется исходное форматирование). Для того, чтобы первая буква слов всегда была заглавной, исполь­зуется свойство QFont::Capitalize. Стоит отметить также значения QFont::AllUpperCase (все символы в верхнем регистре) и QFont::AllLowerCase (все символы в нижнем регистре).

9.5. Привязка компонентов

В свободном от компонентов месте формы нажмите правой кнопкой мыши и выберите пункт **Компо­новка**, далее **Скомпоновать по сетке**.

**Результат.** Если при выполнении программы изменить размер формы Form, то размер компонентов будет скорректирован в соответствии с новым размером формы (ползунки horizontalSlider—horizontalSlider\_5 изменят ширину, а метка label\_6 — ширину и высоту).

**Недочет.** При уменьшении размера формы может возникнуть ситуация, когда цветовая метка переста­нет быть видна, а ползунки станут слишком узкими.

**Исправление.** Добавьте в конструктор класса Form оператор

**setMinimumSize(size());**

**Результат.** Теперь при выполнении программы размер формы можно только увеличить (по сравнению с исходным), поскольку минимально допустимый размер формы, определяемый свойством minimumSize, совпадает с исходным размером формы, хранящимся в свойстве size (см. комментарий).

Комментарий

Помимо свойства minimumSize, использованного при исправлении недочета, имеется аналогичное свойство maximumSize, которое позволяет ограничить максимальный размер формы. Заметим, что данные свойства имеются у всех визуальных компонентов; по умолчанию minumumSize равно **0; 0**, а maximumSize 16777215; 16777215, т. е. ограничение на размер отсутствует.