Глава 4

Сигналы, слоты и события: проект SIGSLOTS

Проект SIGSLOTS знакомит с базовыми приемами разработки программ, управляемых сиг­налами и слотами (signals and slots), а также событиями (events). Демон­стрируется связыва­ние сигнала с обработчиком (в режиме ди­зайна и на про­граммном уровне), отключение об­работчика от сигнала и его повторное под­ключение. Рассматривается статическая функция currentMSecsSinceEpoch класса QDateTime и свойства визуальных компо­нентов, связанные с их раз­мером и положением на экране. Да­ется обзор структуры графического при­ложения и входящих в него элементов.

4.1. Связывание сигнала с обработчиком

Проект SIGSLOTS является первым графическим приложением, рассматри­ваемым в книге, поэтому действия по его разработке мы опишем более по­дробно.

После создания нового проекта типа **Приложение Qt Widgets** разместите в форме Form компонент-кнопку Push Button, используя *окно компонентов* **Панель виджетов** (она расположена в подразделе **Buttons**). Добавленной кнопке будет автоматически присвоено имя pushButton, замените его на button для уменьшения длины кода.

Настройте свойства формы Form и кнопки button (листинг 4.1). Для этого надо использовать *окно свойств* **Редактор свойств**.

По рис. 4.1 настройте размеры формы и расположение кнопки.

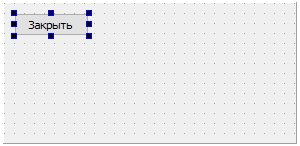
С сигналом clicked() компонента button свяжите стандартный слот формы Form close(). Для этого необходимо перейти во вкладку **Редактор сигналов и слотов**, нажать на кнопку добавления нового соединения (с изображением зеленого плюса) и указать в поле **Отправитель** button, в поле **Сигнал** clicked, в поле **Получатель** Form и в поле **Слот** close.

Перейдите из вкладки **Дизайн** во вкладку **Редактор**. Подключите в файле form.h заголовочный файл QMouseEvent, также добавьте описание события mousePressEvent в класс Form:

**protected:**

**void mousePressEvent(QMouseEvent \*event);**

Создайте в файле form.cpp обработчик события mouseEvent (листинг 4.2).



**Рис. 4.1.** Вид формы Form для проекта SIGSLOTS на начальном этапе разработки

**Листинг 4.1. Настройка свойств**

Form: windowTitle = **Прыгающие кнопки**

button: objectName = **button**, text = **Закрыть**

**Листинг 4.2. Событие** **mousePressEvent**

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

**{**

**ui->button->move(event->x() - ui->button->width() / 2,**

**event->y() - ui->button->height() / 2);**

**}**

**Результат.** После запуска программы на экране появляется окно **Прыгаю­щие кнопки** с кнопкой **Закрыть**. Окно по умолчанию располагается в центре экрана. При щелчке мышью в любом точке окна кнопка Закрыть услужливо прыгает на указанное место. Нажа­тие кнопки **Закрыть** приводит к завершению программы.

Комментарии

1. В режиме дизайнера можно соединять стандартные сигналы компонен­тов со стандартными слотами других компонентов. Благодаря этому не потребовалось добавлять в исходный код обработчик нажатия кнопки button. Для написания более сложных обработчиков нужно в дизайнере нажать правой кнопкой мыши на кнопку и выбрать пункт **Перейти к слоту…**, далее выбрать сигнал clicked(). После этого в файле исход­ных кодов появится заготовка для обработчика сигнала.
2. В методе mousePressEvent изменяется свойство geometry кнопки с по­мощью метода move. Данное свойство содержит координаты и размер виджета, поэтому основной сеттер этого свойства setGeometry содер­жит четыре параметра. Однако размер кнопки не меняется, поэтому до­статочно вызова метода move (его параметры – две координаты типа int). Координаты точки, в которой была нажата кнопка мыши, содержатся в функциях x и y параметра event типа QMouseEvent. Вызовы функций width (ширина) и height (высота) кнопки используются для того, чтобы отцентрировать кнопку относительно курсора мыши.
3. Положение и размеры любого визуального компонента, в том числе и самой формы, можно определить и изменить с помощью набора свойств. За положение отвечает уже упомянутое свойство geometry() типа QRect. Данный тип содержит четыре поля типа int – координаты верхнего левого угла клиентской части формы (в клиентскую часть формы не входит ее заголовок и рамка). За размер компонента отвечает свойство size() c полями width() и height() типа int.
4. Как программа хранит информацию о компонентах, размещенных в форме, и как узнает значения свойств, которые настроены с помощью окна свойств? Вся эта информация сохраняется в файле разметки (в нашем случае этот файл имеет имя form.ui). Хотя обычно необходимости в его «ручной» корректировке нет, полезно ознакомиться с его содержимым, открыв данный файл в текстовом редакторе (для этого достаточно выполнить щелчок правой кнопкой мыши на имени файла и выбрать пункт **Открыть с помощью → Текстовый редактор)**. Файл form.ui содержит основанную на XML разметку, хранящую информацию обо всех виджетах и их свойствах. Во время компиляции разметка с помощью утилиты uic конвертируется в заголовочный файл (имеет такое же имя, что и основной хэдер формы, но с префикcом ui\_) ui\_form.h. В этом файле создаются виджеты и их свойства, указанные в разметке. Данный заголовочный файл автоматически добавляется к основным файлам формы.
5. Где находятся операторы, с которых начинается выполнение программы? Любая программа на языке C++ начинается с выполнения стартового метода main(). При создании шаблона приложения в Qt Creator метод main размещается в файле main.cpp. Этот файл создается автоматически, и, подобно файлу form.ui, как правило, не требует корректировки.

4.2. Отключение слота от сигнала

Добавьте в форму еще одну кнопку (дайте ей имя button\_2) и сделайте её свойство text пустым, используя окно **Редактор свойств** (рис. 4.2).

В файле