ТЕХНОАТОМ

# Лекция 8. Введение в программирование UI при помощи библиотеки Qt 5

МИФИ, 2016

Роман Кузнецов

# Просьба отметиться на портале!

# План лекции

1. Паттерн MVC при программировании UI.
2. Краткий обзор библиотеки Qt 5.
3. Создание оконного приложения на Qt.

# Model View Controller (MVC)

- паттерн проектирования, который предлагает архитектурно разделять данные, бизнес-логику и графический интерфейс программы.

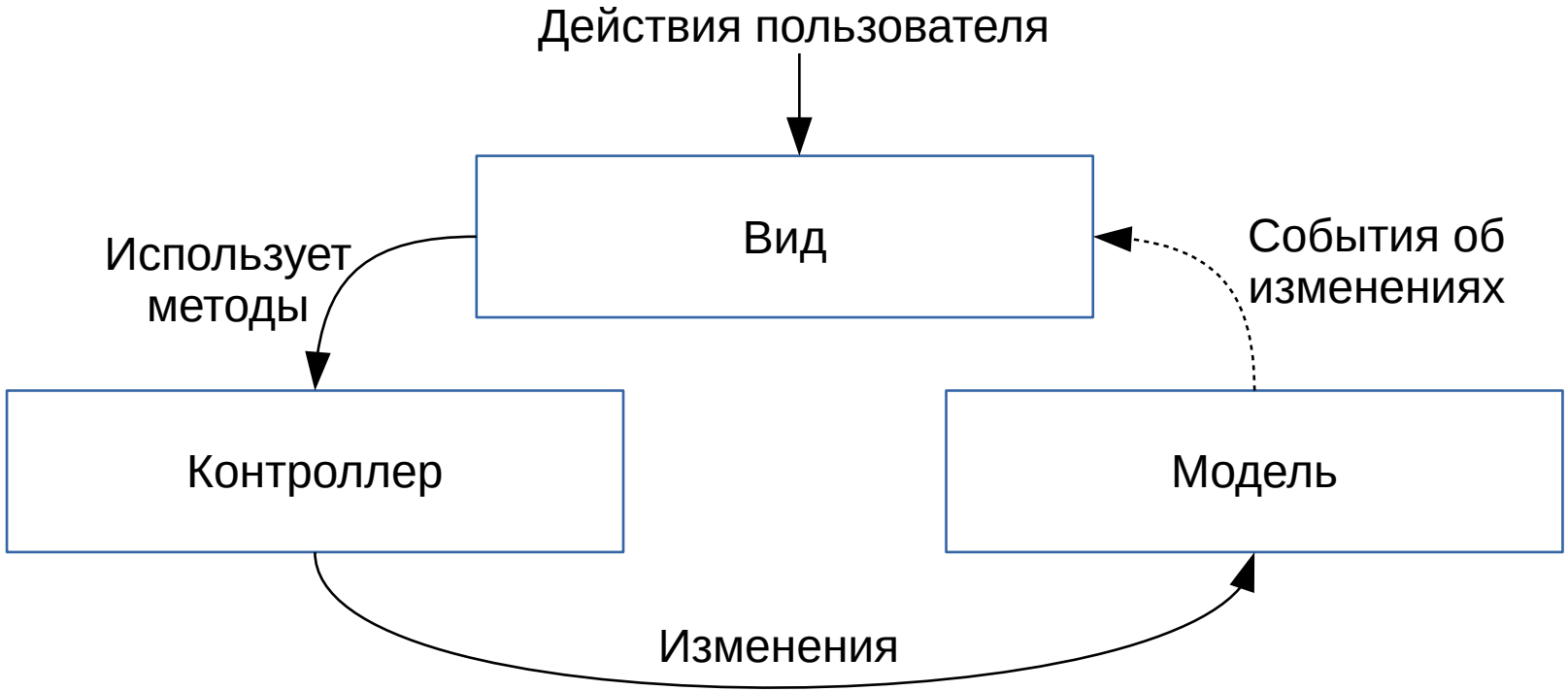
Выделяется 3 компонента:

1) Модель;

2) Контроллер;

3) Представление.

# Model View Controller (MVC)



# Model View Controller (MVC)

Основная идея  - контроллер и представление зависят от модели, но модель никак не зависит от этих двух компонент.

- Контроллер определяет, какое представление должно быть отображено в данный момент.
- События представления могут повлиять только на контроллер.
- Контроллер может повлиять на модель и определить другое представление.
- Возможно несколько представлений для одной модели.

# Qt

— это библиотека для разработки ПО с графическим интерфейсом на C++ (также существует для Ruby — QtRuby, для Python — PyQt, PHP — PHP-Qt и других языков программирования).

Qt полностью объектно-ориентированная и кроссплатформенная (Linux, Windows, Mac OS X и др.). Включает в себя множество классов для работы с сетью, базами данных, классы-контейнеры, для создания графического интерфейса и множество других.

# Meta Object Compiler

Qt использует MOC (Meta Object Compiler) для предварительной компиляции программ. Исходный текст программы обрабатывается MOC, который ищет в классах программы макрос Q_OBJECT и переводит исходный код в мета-объектный код, после чего мета-объектный код компилируется компилятором C++. MOC расширяет функциональность фреймворка, благодаря ему добавляются такие понятия, как слоты и сигналы.

# Виджеты

В Qt имеется огромный набор виджетов (Widget), таких как: кнопки, прогресс бары, переключатели, checkbox'ы, и др. — они обеспечивают стандартную функциональность GUI (графический интерфейс пользователя). Позволяет использовать весь функционал пользовательского интерфейса — меню, контекстные меню, drag&drop и др.

# Состав библиотеки Qt

**QtCore** — классы ядра библиотеки Qt, они используются другими модулями.

**QtGui** — модуль содержит компоненты графического интерфейса.

**QtNetwork** — модуль содержит классы для работы с сетью. В него входят классы для работы с протоколами FTP, HTTP и др.

**QtOpenGL** — модуль содержит классы для работы с OpenGL

# Состав библиотеки Qt

**QtSql** — содержит классы для работы с различными базами данных с использованием языка SQL.

**QtSvg** — содержит классы, позволяющие работать с данными Scalable Vector Graphics (SVG).

**QtXml** — классы для работы с XML.

**QtScript** — классы для работы с Qt Scripts.

# Основа приложения на Qt

Класс **QApplication** представляет само приложение с графическим интерфейсом пользователя, содержит цикл обработки сообщений, поддерживает всю внутреннюю инфраструктуру.

Класс **QMainWindow** определяет главное окно приложения. От данного класса, как правило, наследует пользовательский класс, переопределяющий часть логики главного окна. Пользовательский интерфейс в виде виджетов начинает создаваться отсюда.

# Сигналы и слоты

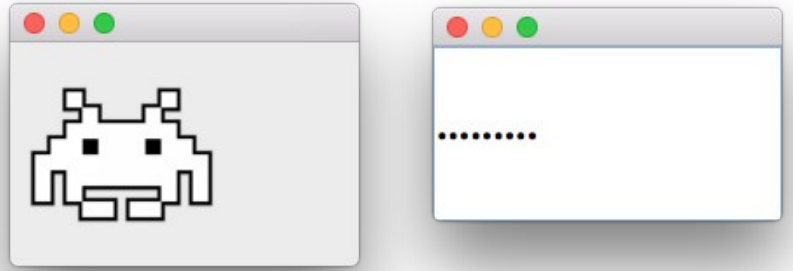
- механизм, реализующйи паттерн Observer в Qt. Благодаря MOC этот механизм встроен в язык.

```
class MyWidget
{
  Q_OBJECT

public:
  MyWidget(){ }

Q_SIGNALS:
  // Декларируем сигналы
  void SmthChanged();

private slots:
  // Декларируем слоты
  void OnButtonPressed();
};
```

# Текстовые виджеты

**QLabel**

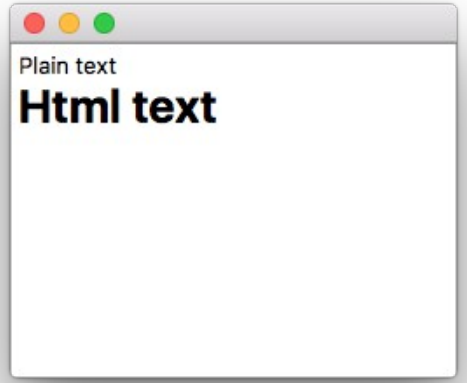
```
QLabel * label = new QLabel("Text", this);
QPixmap pixmap("alien.png");
label->setPixmap(pixmap);
label->adjustSize();
```



**QLineEdit**

```
QLineEdit * line = new QLineEdit(this);
line->setText("Edit me");
line->setEchoMode(QLineEdit::Password);
connect(line, SIGNAL(textChanged(QString const &)), this, SLOT(/* bind to slot */));
connect(line, SIGNAL(editingFinished()), this, SLOT(/* bind to slot */));
```

# Текстовые виджеты




**QTextEdit**

```
QTextEdit * edit = new QTextEdit(this);
edit->setPlainText("Plain text");
edit->append("<h1>Html text</h1>");
edit->adjustSize();
connect(edit, SIGNAL(textChanged(QString const &)), this, SLOT(/* bind to slot */));
```

# Кнопки

**QPushButton**

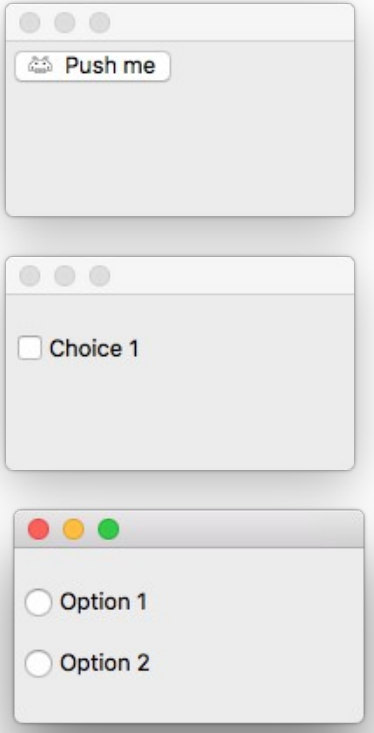
```
QPushButton * button = new QPushButton("Push me", this);
button->setIcon(QIcon("alien.png"));
connect(button, SIGNAL(clicked()), this, SLOT(/* bind to slot */));
```

**QCheckBox**

```
QCheckBox * check = new QCheckBox("Choice 1", this);
check->move(5, 15);
```

**QRadioButton**

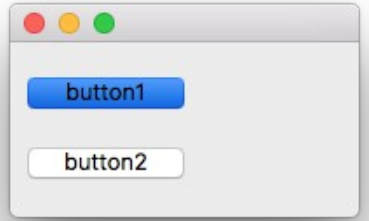
```
QRadioButton * button1 = new QRadioButton("Option 1", this);
button1->move(5, 15);
QRadioButton * button2 = new QRadioButton("Option 2", this);
button2->move(5, 50);
```



# Кнопки

**QButtonGroup**

```
QButtonGroup * group = new QButtonGroup(this);
QPushButton * button1 = new QPushButton("button1", this);
button1->move(5, 15);
button1->setCheckable(true);
group->addButton(button1);
QPushButton * button2 = new QPushButton("button2", this);
button2->move(5, 55);
button2->setCheckable(true);
group->addButton(button2);
group->setExclusive(true);
connect(group, SIGNAL(buttonClicked(QAbstractButton*)), this, SLOT(/* bind to slot */));
```



# Виджеты, работающие со значениями

**QSlider**

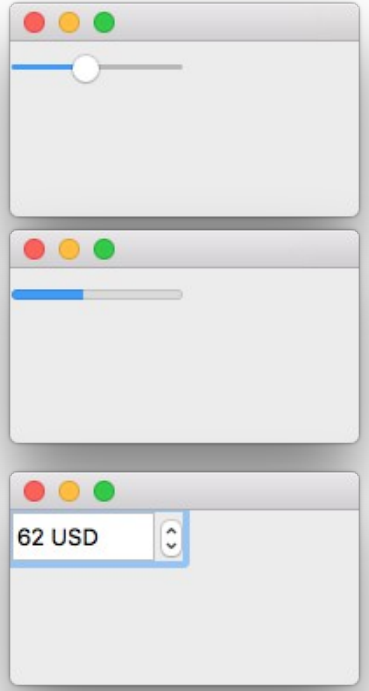
```
QSlider * slider = new QSlider(Qt::Horizontal, this);
slider->setRange(0, 99);
slider->setValue(42);
```

**QProgressBar**

```
QProgressBar * progressBar = new QProgressBar(this);
progressBar->setRange(0, 99);
progressBar->setValue(42);
```

**QSpinBox**

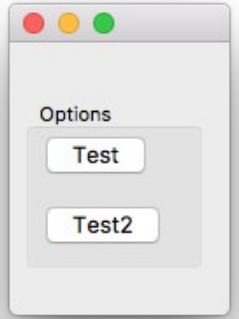
```
QSpinBox * spinBox = new QSpinBox(this);
spinBox->setRange(0, 99);
spinBox->setValue(62);
spinBox->setSuffix(" USD");
connect(spinBox, SIGNAL(valueChanged(int)), this, SLOT(/* bind to slot */));
```



# Группировка

**QGroupBox**

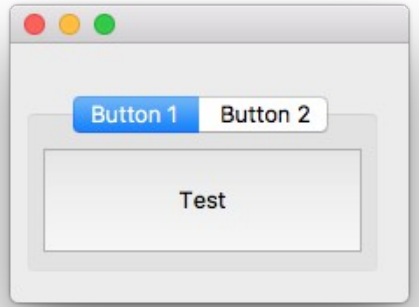
```
QGroupBox * box = new QGroupBox("Options", this);
box->resize(100, 100);
box->move(10, 30);
QPushButton * button1 = new QPushButton("Test", box);
button1->move(5, 20);
QPushButton * button2 = new QPushButton("Test2", box);
button2->move(5, 60);
```



# Вкладки

**QTabWidget**

```
QTabWidget * tab = new QTabWidget(this);
tab->resize(200, 100);
tab->move(10, 30);
QPushButton * button1 = new QPushButton("Test");
tab->addTab(button1, "Button 1");
QPushButton * button2 = new QPushButton("Test2");
tab->addTab(button2, "Button 2");
connect(tab, SIGNAL(currentChanged(int)), this, SLOT(/* bind to slot */));
```

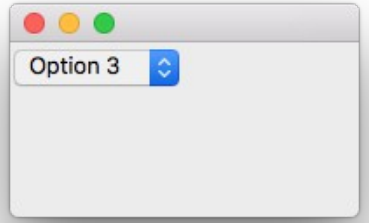



**Ещё методы:**
setCurrentWidget(QWidget)
setTabPosition(TabPosition)
setTabsClosable(bool)

# Виджеты с элементами

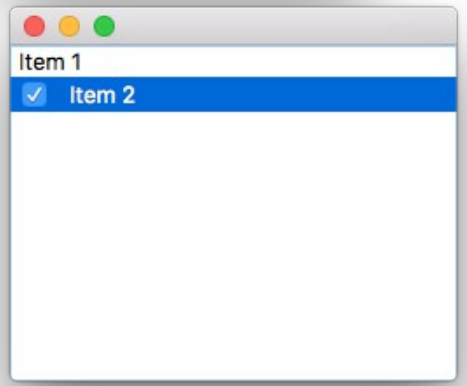
**QComboBox**

```
QComboBox * comboBox = new QComboBox(this);
comboBox->addItem("Option 1", 1);
comboBox->addItem("Option 2", 2);
comboBox->addItem("Option 3", 3);
int const data = comboBox->itemData(1).toInt();
comboBox->setCurrentIndex(2);
connect(comboBox, SIGNAL(activated(int)), this, SLOT(/* bind to slot */));
```




**QListWidget**

```
QListWidget * listWidget = new QListWidget(this);
listWidget->addItem("Item 1");
QListWidgetItem * item = new QListWidgetItem("Item 2", listWidget);
item->setCheckState(Qt::Checked);
```



# Другие виджеты

- QToolBox
- QDateEdit, QTimeEdit, QDateTimeEdit
- QCalendarWidget
- QToolButton
- QSplitter
- QStackedWidget

# Layouts

- классы для организации пространства внутри окна и перестроения интерфейса.

**Основные виды:**

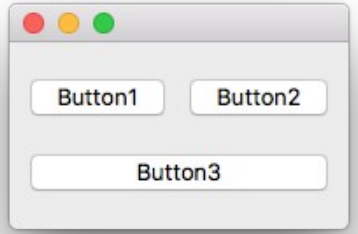
- QHBoxLayout
- QVBoxLayout
- QGridLayout
- QFormLayout
- QstackedLayout — стек виджетов (в один момент времени виден только один)

# Строка (столбец)

```
QWidget * centralWidget = new QWidget(this);
setCentralWidget(centralWidget);
QPushButton * button1 = new QPushButton("Button1");
QPushButton * button2 = new QPushButton("Button2");
QPushButton * button3 = new QPushButton("Button3");
QHBoxLayout * newLayout = new QHBoxLayout(centralWidget);
newLayout->addWidget(button1);
newLayout->addWidget(button2);
newLayout->addWidget(button3);
```

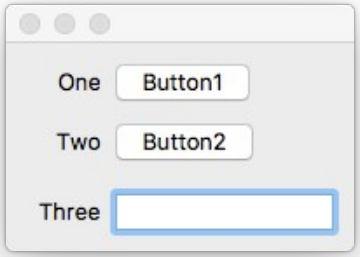
# Таблица

```
QWidget * centralWidget = new QWidget(this);
setCentralWidget(centralWidget);
QPushButton * button1 = new QPushButton("Button1");
QPushButton * button2 = new QPushButton("Button2");
QPushButton * button3 = new QPushButton("Button3");
QGridLayout * layout = new QGridLayout(centralWidget);
layout->addWidget(button1, 0, 0);
layout->addWidget(button2, 0, 1);
layout->addWidget(button3, 1, 0, 1, 2);
```



# Форма

```
QWidget * centralWidget = new QWidget(this);
setCentralWidget(centralWidget);
QPushButton * button1 = new QPushButton("Button1");
QPushButton * button2 = new QPushButton("Button2");
QLineEdit * edit = new QLineEdit();
QFormLayout * layout = new QFormLayout(centralWidget);
layout->addRow("One", button1);
layout->addRow("Two", button2);
layout->addRow("Three", edit);
```



# Интерактив

Смотрим код

https://github.com/rokuz/template/tree/window_demo

# Домашнее задание

1) Перенести главное окно из моего репозитория в ваш проект;
2) Сделать окно настроек для игры (наполнение любое).

**Срок сдачи:** 17.11.2016 23:59:59

# Просьба оставить отзыв о данном занятии на портале!


# Спасибо за внимание!

Роман Кузнецов

r.kuznetsov@mapswithme.com