УДК 004

Использование модуля кроссплатформенного фреймворка Qt - QtQuick и описательного языка QML для разработки программного обеспечения

Ленкин Алексей Викторович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Цель исследования рассмотреть использование модуля QtQuick для быстрого создания кроссплатформенных приложений, а также, используемый в нём. описательный язык QML. Методами исследования являются анализ теоретической информации, а также синтез полученной информации для создания простого проекта. Итогом статьи является демонстрация полезности использования QtQuick для создания проектов любого масштаба.
Ключевые слова: фреймворк, Qt, C++, QtQuick, QML

Using the Qt - QtQuick Cross Platform Framework Module and QML Descriptive Language for Software Development

Lenkin Aleksei Viktorovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of the study is to consider using the QtQuick module to quickly create cross-platform applications, also available in it. QML descriptive language. Research methods are the analysis of theoretical information, as well as the synthesis of the information obtained to create a simple project. The result of this article is a demonstration of the usefulness of using QtQuick to create projects of any scale.
Keywords: framework, Qt, C++, QtQuick, QML

Чаще всего, приоритетной задачей для крупных производителей программного обеспечения помимо выпуска качественного продукта является также и его быстрая разработка, но в большинстве случаев создание приложения проходит медленнее запланированных сроков. Как правило, это бывает связанно с тем, что дизайнеры и программисты с трудом понимают друг друга, так как мыслят совершенно по-разному и работают в разной сфере, поэтому стараются избегать общения.

Обычно в таком случае для оказания помощи во взаимодействии своих разработчиков компания нанимает дорогостоящего специалиста, который в совокупности разбирается в обеих сферах. Но также существует и

программное средство для связи дизайнеров с программистами - это использование языка QML и специального модуля фреймворка Qt QtQuick.

Цель исследования рассмотреть использование модуля QtQuick для быстрого создания кроссплатформенных приложений, а также, используемый в нём. описательный язык QML

Исследованиями в данной теме занимались следующие авторы. Семенов А.А. и Афанасьев Г.И. описали технологию созданию графического интерфейса с помощью QML [1]. Продемонстрировали разработку подсистем графического интерфейса системы Talgat с использованием QtQuick Квасников А.А., Куксенко С.П., Лежнин Е.В. [2]. Пчелов А.К., Ильина Л.А. показали создание приложений для Android, используя QML [3]. Авербух В.Л., Бахтерев М.О., Васёв П.А., Манаков Д.В., Стародубцев И.С. выделили QtQuick в своей статье «Развитие подходов к разработке специализированных систем компьютерной визуализации» как один из интрументов визуализации [4]. Трусов Е.В. описал в своей работе как использовать библиотеку «QT Quick Controls», наборы готовых моделей для создания интерфейсов [5].

Qt Quick - это свободно распространяемая прикладная среда, разработанная и поддерживаемая Qt Project в рамках Qt. Она предоставляет способ создания пользовательского высокодинамичного графического интерфейса пользователя с плавными переходами и эффектами, которые становятся все более распространенными, особенно в мобильных устройствах. Qt Quick включает в себя описательный язык сценариев, называемый QML. Qt Declarative – это интерпретатор, который читает описание пользовательского интерфейса на Qt, данные QML и отображает пользовательский интерфейс, который он описывает. Синтаксис QML позволяет использовать JavaScript для обеспечения логики, и он часто используется для этой цели. Однако это не единственный способ: логика может быть написана и с нативным кодом [6].

QML - описательный язык программирования, основанный на JavaScript, предназначенный для дизайна приложений, делающих основной упор на пользовательский интерфейс. QML также позволяет встраивать куски кода JavaScript.

Приведём пример работы со связкой QtQuick и фреймворка Qt, на примере создания простой формы:

1. Установим с официального сайта https://www.qt.io/ [6] бесплатное приложение для разработки Qt Creator.
2. Создадим новый QML проект «File – New File or Project – QT Quick Application Empty (рис. 1).

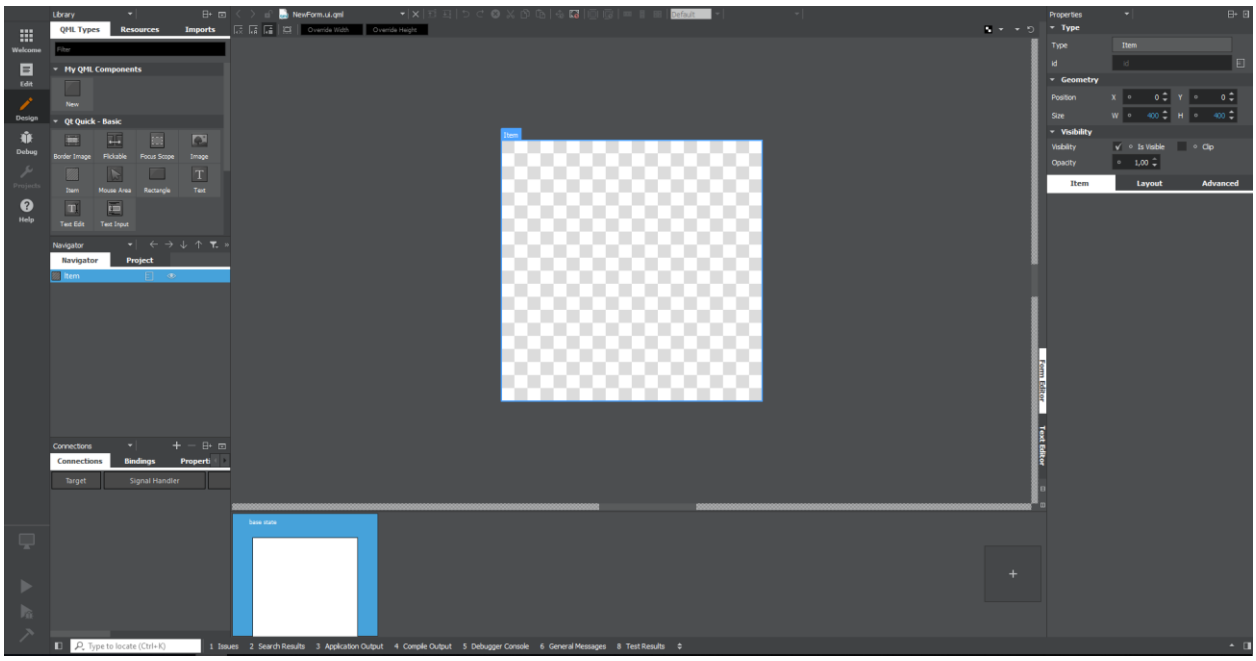

Рисунок 1. Интерфейс среды Qt Creator для работы с модулем QtQuick

3. Создадим простую прозрачную форму, кнопку с округлением и крупный текст «Hello World», а также поля для ввода текста и кнопку. Для этого во вкладке Design расположим три объекта вида Rectangle, после чего перейдём во вкладку Edit и используем следующий листинг:

```
import QtQuick 2.9
import QtQuick.Window 2.2
import QtQuick.Controls 2.4

Window {
    visible: true
    width: 400
    height: 400
    title: qsTr("Hello World")
    color: "red"
    Text {
        objectName: "textlabel"
            x: 171
            y: 65
            anchors.horizontalCenter:
parent.horizontalCenter
            anchors.verticalCenter:
parent.verticalCenter
            text: "Hello World"
            font.pointSize: 24
            color: "black"
    }

    TextField {
```

```
        id: textField
        x: 86
        y: 244
        width: 73
        height: 40
        text: qsTr("")

        Text {
            id: element
            x: 75
            y: 8
            text: qsTr("+")
            font.pixelSize: 20
        }
    }

    TextField {
        id: textField1
        x: 179
        y: 244
        width: 73
        height: 40
        text: qsTr("")
    }

    TextField {
            id: textField2
            x: 280
            y: 244
            width: 73
            height: 40
            text: qsTr("")
        }

        Text {
            id: element1
            x: 259
            y: 252
            text: qsTr("=")
            font.pixelSize: 20
        }

        Button {
            id: button
            x: 119
```

```
            y: 301
            width: 162
            height: 21
            text: qsTr("Посчитать")
            font.pointSize: 12
            focusPolicy: Qt.WheelFocus
        }
}
```

4. Получив в результате готовую форму (рис.2), покажем как проведёт разработку дальше программист для интеграции данного макета в приложение. Ведь на данный момент эта форма не связаны никакими функциями и может быть легко отделена от проекта.

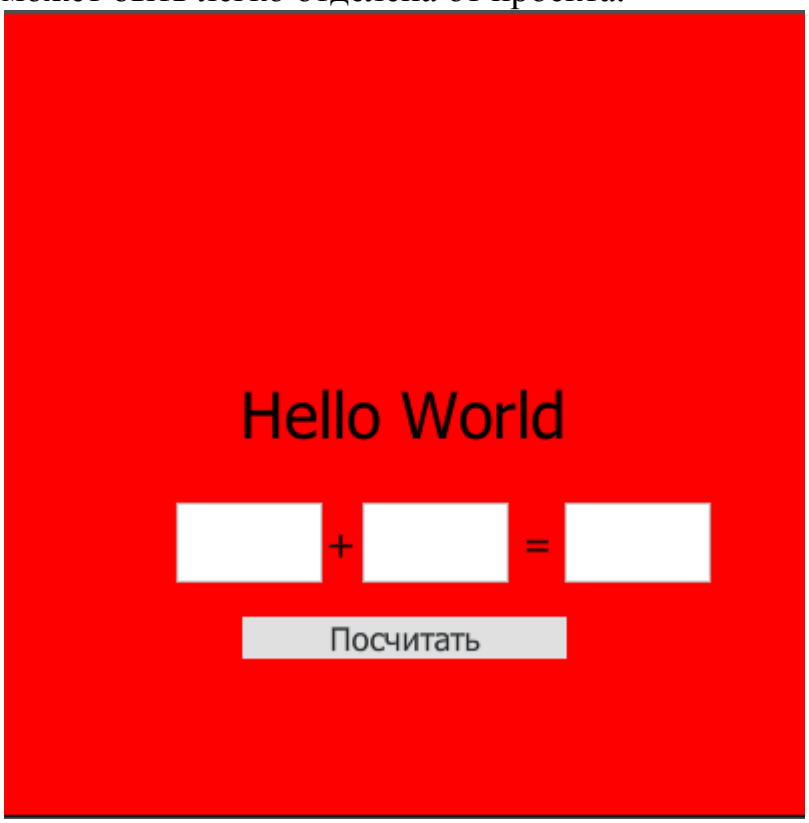

Рисунок 2. Готовая форма QML

5. Для дальнейшего взаимодействия с данной формой необходимо создать заголовочный файл с перечнем используемых функций, а также исходный файл с описанным кодом этих функций. Назовём их соответственно «forqmlform.h» и «forqmlform.cpp». В функции мы будем складывать полученные из формы числа. Листинг представлен ниже.

Листинг «forqmlform.h»:

```
#ifndef FORQMLFORM_H
#define FORQMLFORM_H
extern int plvl;
#include <QObject>
#include <QVariant>
```

```
class ForQMLForm : public QObject
{
    Q_OBJECT
public:
    explicit ForQMLForm(QObject *parent = 0);

signals:


public slots:
    QString plus(QString a, QString b);
};

#endif // FORQMLFORM_H
```

Листинг «forqmlform.cpp»:

```
#include "forqmlform.h"
#include <QTextStream>

ForQMLForm::ForQMLForm(QObject      *parent)      :
QObject(parent)
{

}

QString ForQMLForm::plus(QString a, QString b) {
    return QString::number(a.toInt()+b.toInt());
}
```

6. Данный код будет никак не связан с формой пока их не объединить. Это делается добавлением в главный исходный файл («main.cpp») кода:

```
ForQMLForm appCore;
    QQmlContext *context=engine.rootContext();
    context->setContextProperty("appCore",
&appCore);
```

7. Чтобы отправить сигнал с данными с формы в область кнопки дописывается следующий код, который при нажатии отправит данные с формы на выполнение функции:

```
onClicked: pressplus();
            function pressplus() {

textField2.text=appCore.plus(textField.text,
textField1.text);
            }
```

Таким образом, внешний вид формы, не обладающий никаким функционалом, был связан с программным кодом (результат выполнения на рис.3) и стал полноценной программой.

Рисунок 3. Связанная с кодом форма

Опишем преимущества использования такого подхода при разработки пользовательских приложений:

1. Программист и дизайнер могут работать отдельно, не вмешиваясь в проекты друг друга.
2. Ускорение разработки из-за полностью разделенной разработки внешнего вида и программной части.

Библиографический список

1. Семенов А.А., Афанасьев Г.И. Технология создания графического интерфейса с помощью QML // Теория и практика современной науки. 2017. № 4 (22). С. 1052-1059.
2. Квасников А.А., Куксенко С.П., Лежнин Е.В. Разработка подсистем графического интерфейса системы Talgat // Электронные средства и системы управления. 2017. № 1-2. С. 15-18.
3. Пчелов А.К., Ильина Л.А. Создание приложений для Android // В сборнике: Информатика и вычислительная техника Сборник научных трудов. Посвящается 50-летию Чувашского государственного университета имени И.Н. Ульянова . . Чебоксары, 2017. С. 149-150.

4. Авербух В.Л., Бахтерев М.О., Васёв П.А., Манаков Д.В., Стародубцев И.С. Развитие подходов к разработке специализированных систем компьютерной визуализации // В сборнике: ГРАФИКОН'2015 Труды Юбилейной 25-й Международной научной конференции. 2015. С. 17-21.
5. Трусов Е.В. QT Quick Controls // В сборнике: Новые информационные технологии и системы сборник научных статей XIV Международной научно-технической конференции, посвященной 70-летию кафедры «Вычислительная техника» и 30-летию кафедры «Системы автоматизированного проектирования». 2017. С. 66-67.
6. Qt | Cross-platform software development for embedded & desktop URL: https://www.qt.io/ (дата обращения: 30.08.2019).