**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Отчет

По лабораторной работе №5

На тему: “Исследование способов интеграции интерфейса пользователя на языке QML и функциональности на языке C++”

По дисциплине: “Кроссплатформенное программирование”

Выполнил:

ст.гр. ИС/б-17-2

Черняев Н.Г.

Проверил:

Строганов В.А.

Севастополь

2021

1 ЦЕЛЬ РАБОТЫ

Исследование способов взаимодействия языка C++ и языка разметки QML. Приобретение навыков разработки приложений на основе QML-интерфейса.

2 ПОСТАНОВКА ЗАДАЧИ

2.1. Изучить способы организации взаимодействия QML и серверных классов на C++.

2.2. Разработать класс, реализующий функциональность по варианту задания, приведенному в Приложении.

2.3. Определить свойства и методы, необходимые для использования в QML разметке, с помощью соответствующих макросов.

2.4. Добавить класс в контекст Qt Quick приложения.

2.5. Дополнить разметку необходимыми элементами управления с вызовом соответствующих методов.

2.6. Исследовать эффективность работы полученного приложения, имитируя ошибки ввода/вывода.

2.7. Выполнить сравнительный анализ методов построения приложений в данной лабораторной работе и работе №3 по критерию трудоемкости проектирования и программирования.

Вариант 1

Добавить кнопки, позволяющие сохранять и загружать текст из выбранного поля ввода.

3 ТЕКСТ ПРОГРАММЫ

file.h:

#ifndef FILE\_H

#define FILE\_H

#include <QtGui/QGuiApplication>

#include <QFile>

#include <QTextStream>

class **File** : public QObject

{

Q\_OBJECT

public:

**File**(QObject \*parent = 0);

Q\_INVOKABLE QString **load**(QString fileName);

Q\_INVOKABLE void **save**(QString fileName, QString text);

};

#endif // FILE\_H

file.cpp:

#include "file.h"

File::**File**(QObject \*parent):QObject(*parent*)

{

}

QString File::**load**(QString fileName)

{

QFile file(fileName);

QString result = "";

if (file.*open*(QFile::ReadWrite))

{

QTextStream inStream(*&file*);

result = inStream.readAll();

}

file.*close*();

return result;

}

void File::**save**(QString fileName, QString text)

{

QFile file(fileName);

if (file.*open*(QFile::ReadWrite | QIODevice::Truncate))

{

QTextStream outStream(*&file*);

outStream << text;

}

file.*close*();

}

Дополним main.qml следующим кодом:

Button {

id: *button1*

x: 27

y: 22

width: 100

height: 35

labelSize: 11

label: "Загрузить"

onButtonClick:

{

*textEdit*.text = *file*.load("text1.txt")

}

}

Button {

id: *button2*

x: 146

y: 22

width: 100

height: 35

labelSize: 11

label: "Сохранить"

onButtonClick:

{

*file*.save("text1.txt", *textEdit*.text)

}

}

Button {

id: *button3*

x: 27

y: 210

width: 100

height: 35

label: "Загрузить"

labelSize: 11

onButtonClick:

{

*textEdit1*.text = *file*.load("text2.txt")

}

}

Button {

id: *button4*

x: 146

y: 210

width: 100

height: 35

label: "Сохранить"

labelSize: 11

onButtonClick:

{

*file*.save("text2.txt", *textEdit1*.text)

}

}

4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Начальное окно приложения представлено на рисунке 1. Были добавлены 4 кнопки для сохранения и загрузки текста из файла.

В файлах text1.txt и text2.txt хранится введенный в поля текст для первого и второго поля соответственно. Результат выгрузки текста из файла представлен на рисунке 2.

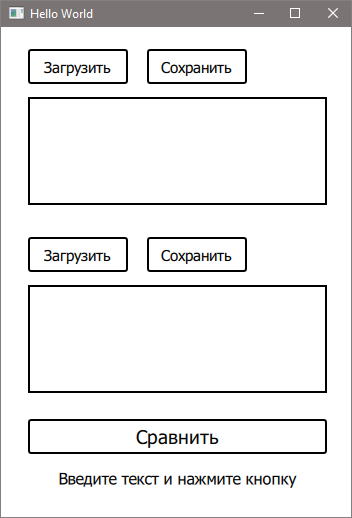


Рисунок 1 – Новые кнопки для загрузки и сохранения текста

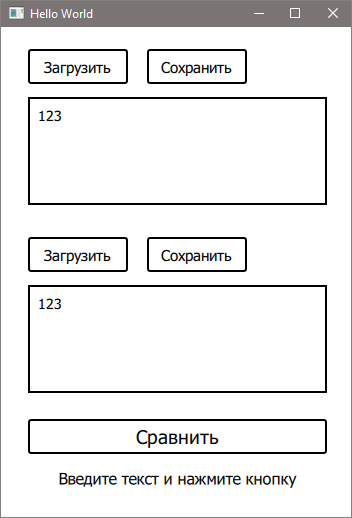


Рисунок 2 – Результат выгрузки текста из файла

ВЫВОДЫ

В ходе выполнения лабораторной работы был изучен способ интеграции C++ кода с QML. Данный метод построения приложений необходимо применять в следующих случаях:

* необходимо получить доступ к функциональности извне окружения QML;
* необходимо реализовывать более производительные функции, где требуется нативный код для повышения эффективности;
* когда пишется большой и сложный код