**Лабораторная работа №5**

**«ИССЛЕДОВАНИЕ СПОСОБОВ ИНТЕГРАЦИИ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ НА ЯЗЫКЕ QML И**·**ФУНКЦИОНАЛЬНОСТИ НА ЯЗЫКЕ C++»**

**Цель работы**

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

**Постановка задачи**

1. Изучить способы организации взаимодействия QML и серверных класов на C++ (выполняется в ходе самостоятельной подготовки к лабораторной работе).

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

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

4. Добавить класс в контекст Qt Quick приложения.

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

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

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

**Ход работы**

1. Был создан класс FileSaveReader, позволяющий сохранять в файл и загружать из него состояние игрового поля. Также на разметку были добавлены 2 кнопки по нажатию на первую текущее состояние игрового поля сохраняется в файл, по нажатию на вторую – загружается из файла.

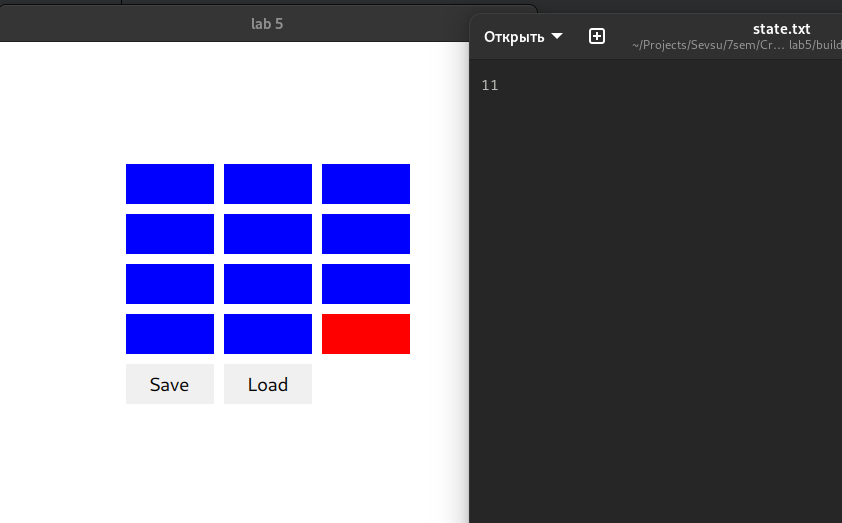


Рисунок 1 – Графический интерфейс приложения

**Выводы**

В ходе выполнения лабораторной работы были исследованы способы взаимодействия языка C++ и языка разметки QML. Приобретены навыки разработки приложений на основе QML-интерфейса.

Рассматриваемый подход с использованием возможностей QTQuick и QML затрачивает больше времени при разработке, но является более гибким, поскольку можно разделить логику отображений от логики самой программы, что позволит в дальнейшем легче масштабировать приложение.

**ПРИЛОЖЕНИЕ А**

**«ИСХОДНЫЙ КОД ПРОГРАММЫ»**

Листинг А1 – Файл main.qml

import QtQuick 2.15

import QtQuick.Window 2.15

Window {

width: 640

height: 480

visible: true

title: qsTr("lab 5")

function buttonClicked(button) {

var buttons = [but1, but2, but3, but4, but5, but6, but7, but8, but9, but10, but11, but12];

if(button.color == "#ff0000")

{

button.color = "#0000ff";

var rand = Math.floor(Math.random() \* buttons.length);

var rand\_button = buttons[rand];

rand\_button.color = "#ff0000";

file.set\_state(rand);

}

}

function loadStates(id)

{

var buttons = [but1, but2, but3, but4, but5, but6, but7, but8, but9, but10, but11, but12];

for (let i = 0; i < buttons.length; i++)

{

if (i === id)

{

buttons[i].color="#ff0000";

} else

buttons[i].color = "#0000ff";

}

}

Grid {

columns: 3

rows: 5

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

columnSpacing: 10

rowSpacing: 10

Button {

id: but1

color: "#ff0000"

onButtonClick:

buttonClicked(but1)

}

Button {

id: but2

onButtonClick:

buttonClicked(but2)

}

Button {

id: but3

onButtonClick:

buttonClicked(but3)

}

Button {

id: but4

onButtonClick:

buttonClicked(but4)

}

Button {

id: but5

onButtonClick:

buttonClicked(but5)

}

Button {

id: but6

onButtonClick:

buttonClicked(but6)

}

Button {

id: but7

onButtonClick:

buttonClicked(but7)

}

Button {

id: but8

onButtonClick:

buttonClicked(but8)

}

Button {

id: but9

onButtonClick:

buttonClicked(but9)

}

Button {

id: but10

onButtonClick:

buttonClicked(but10)

}

Button {

id: but11

onButtonClick:

buttonClicked(but11)

}

Button {

id: but12

onButtonClick:

buttonClicked(but12)

}

Button {

label: "Save"

color: "#f0f0f0"

onButtonClick:

file.save\_state(file.get\_state());

}

Button {

label: "Load"

color: "#f0f0f0"

onButtonClick:

loadStates(file.load\_state());

}

}

}

}

Листинг А2 – Файл main.cpp

#include <QGuiApplication>

#include <QQmlApplicationEngine>

#include <QQmlContext>

#include "filesavereader.h"

int main(int argc, char \*argv[])

{

#if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)

QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);

#endif

QGuiApplication app(argc, argv);

FileSaveReader file;

QQmlApplicationEngine engine;

const QUrl url(QStringLiteral("qrc:/main.qml"));

QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,

&app, [url](QObject \*obj, const QUrl &objUrl) {

if (!obj && url == objUrl)

QCoreApplication::exit(-1);

}, Qt::QueuedConnection);

engine.load(url);

QQmlContext \*rootContext = engine.rootContext();

rootContext->setContextProperty("file",&file);

file.set\_state(0);

return app.exec();

}

}

Листинг А3 – Файл filesavereader.h

#ifndef FILESAVEREADER\_H

#define FILESAVEREADER\_H

#include <QtGui/QGuiApplication>

#include <QFile>

#include <QTextStream>

class FileSaveReader : public QObject

{

Q\_OBJECT

public:

FileSaveReader(QObject \*parent = 0);

int cur\_state;

Q\_PROPERTY(int cur\_state READ get\_state WRITE set\_state)

Q\_INVOKABLE int get\_state();

Q\_INVOKABLE int set\_state(int state);

Q\_INVOKABLE void save\_state(int state);

Q\_INVOKABLE int load\_state();

~FileSaveReader();

private:

QString filename;

};

#endif // FILESAVEREADER\_H

}

Листинг А4 – Файл filesavereader.cpp

#include "filesavereader.h"

FileSaveReader::FileSaveReader(QObject \*parent):

QObject(parent)

{

this->filename = QString("state.txt");

}

FileSaveReader::~FileSaveReader()

{

}

int FileSaveReader::get\_state()

{

return cur\_state;

}

int FileSaveReader::set\_state(int state)

{

cur\_state = state;

return cur\_state;

}

int FileSaveReader::load\_state()

{

QFile file(filename);

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

{

QTextStream in\_stream(&file);

cur\_state = in\_stream.readAll().toInt();

}

file.close();

return cur\_state;

}

void FileSaveReader::save\_state(int state)

{

QFile file(filename);

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

{

QTextStream in\_stream(&file);

QTextStream out\_stream(&file);

out\_stream << cur\_state;

}

file.close();

}

}