Министерство образования Республики Беларусь  
Учреждение образования  
«Брестский государственный технический университет»  
Кафедра ИИТ

Лабораторная работа №3  
По дисциплине: «ОсиСП»

Выполнил:  
Студент 3 курса  
Группы ПО-4(1)  
Грибовский Д. С.  
Проверил:  
Монтик Н.С.

Брест 2021

**Цель работы:**

научиться разрабатывать и использовать динамические библиотеки (DLL) с использованием Qt

**Задание:**

Доработать программу, разработанную в лабораторной работе No1-2, внеся следующие изменения:

1) Выбрать вспомогательные функции и вынести их описание и реализацию в динамическую библиотеку helper.dll. В основном приложении осуществить загрузку реализованных функций.

2) Выбрать вспомогательный класс и вынести его описание и реализацию в динамическую библиотеку helper\_class.dll. В основном приложении осуществить загрузку реализованного класса.

3) Реализовать окно «О программе» в виде объекта динамической библиотеки about.dll с указанием автора программы, группы, курса и краткого описания разработанного приложения.

Осуществить импорт указанной библиотеки и отображение соответствующего окна при выборе пункта меню «О программе».

4) Реализовать расширения для приложения, позволяющие изменять оформление пунктов меню (шрифт, размер, начертание и т.д.). Соответствующие изменения должны происходить при выборе специального пункта меню. Создать как минимум три расширения такого типа.

**Helper.h**

#ifndef HELPER\_H

#define HELPER\_H

#include "helper\_global.h"

#include <QString>

extern "C" HELPER\_EXPORT QString getWindowTitle();

#endif // HELPER\_H

**Helper.cpp**

#include "helper.h"

QString getWindowTitle() {

return QString("Tetrix");

}

**Helper\_class.h**

#include "helper\_class\_global.h"

#ifndef TETRIXPIECE\_H

#define TETRIXPIECE\_H

enum TetrixShape { NoShape, ZShape, SShape, LineShape, TShape, SquareShape,

LShape, MirroredLShape };

class HELPER\_CLASS\_EXPORT TetrixPiece

{

public:

TetrixPiece() { setShape(NoShape); }

void setRandomShape();

void setShape(TetrixShape shape);

TetrixShape shape() const { return pieceShape; }

int x(int index) const { return coords[index][0]; }

int y(int index) const { return coords[index][1]; }

int minX() const;

int maxX() const;

int minY() const;

int maxY() const;

TetrixPiece rotatedLeft() const;

TetrixPiece rotatedRight() const;

private:

void setX(int index, int x) { coords[index][0] = x; }

void setY(int index, int y) { coords[index][1] = y; }

TetrixShape pieceShape;

int coords[4][2];

};

#endif // HELPER\_CLASS\_H

**Helper\_class.cpp**

#include "helper\_class.h"

#include <QtCore>

void TetrixPiece::setRandomShape()

{

setShape(TetrixShape(QRandomGenerator::global()->bounded(7) + 1));

}

void TetrixPiece::setShape(TetrixShape shape)

{

static constexpr int coordsTable[8][4][2] = {

{ { 0, 0 }, { 0, 0 }, { 0, 0 }, { 0, 0 } },

{ { 0, -1 }, { 0, 0 }, { -1, 0 }, { -1, 1 } },

{ { 0, -1 }, { 0, 0 }, { 1, 0 }, { 1, 1 } },

{ { 0, -1 }, { 0, 0 }, { 0, 1 }, { 0, 2 } },

{ { -1, 0 }, { 0, 0 }, { 1, 0 }, { 0, 1 } },

{ { 0, 0 }, { 1, 0 }, { 0, 1 }, { 1, 1 } },

{ { -1, -1 }, { 0, -1 }, { 0, 0 }, { 0, 1 } },

{ { 1, -1 }, { 0, -1 }, { 0, 0 }, { 0, 1 } }

};

for (int i = 0; i < 4 ; i++) {

for (int j = 0; j < 2; ++j)

coords[i][j] = coordsTable[shape][i][j];

}

pieceShape = shape;

}

int TetrixPiece::minX() const

{

int min = coords[0][0];

for (int i = 1; i < 4; ++i)

min = qMin(min, coords[i][0]);

return min;

}

int TetrixPiece::maxX() const

{

int max = coords[0][0];

for (int i = 1; i < 4; ++i)

max = qMax(max, coords[i][0]);

return max;

}

int TetrixPiece::minY() const

{

int min = coords[0][1];

for (int i = 1; i < 4; ++i)

min = qMin(min, coords[i][1]);

return min;

}

int TetrixPiece::maxY() const

{

int max = coords[0][1];

for (int i = 1; i < 4; ++i)

max = qMax(max, coords[i][1]);

return max;

}

TetrixPiece TetrixPiece::rotatedLeft() const

{

if (pieceShape == SquareShape)

return \*this;

TetrixPiece result;

result.pieceShape = pieceShape;

for (int i = 0; i < 4; ++i) {

result.setX(i, y(i));

result.setY(i, -x(i));

}

return result;

}

TetrixPiece TetrixPiece::rotatedRight() const

{

if (pieceShape == SquareShape)

return \*this;

TetrixPiece result;

result.pieceShape = pieceShape;

for (int i = 0; i < 4; ++i) {

result.setX(i, -y(i));

result.setY(i, x(i));

}

return result;

}

**About.h**

#ifndef ABOUT\_H

#define ABOUT\_H

#include "about\_global.h"

extern "C" ABOUT\_EXPORT void about();

#endif // ABOUT\_H

**About.cpp**

#include "about.h"

#include <QMessageBox>

void about()

{

QMessageBox msgBox;

msgBox.setWindowTitle("About programm");

msgBox.setText("The programm was developed by Gribovskij Dima\ncourse : 3\ngroup: PO-4\n2021");

msgBox.exec();

}

**Подключение библиотек:**

#include <QuiLib/include/helper\_class.h>

#include <QuiLib/include/about.h>

#include <QuiLib/include/helper.h>

**Функция смены шрифта:**

void TetrixBoard::changeFont() {

bool Changed;

QFont newFont = QFontDialog::getFont(&Changed);

if (Changed) {

QApplication::setFont(newFont);

QSettings settings(this);

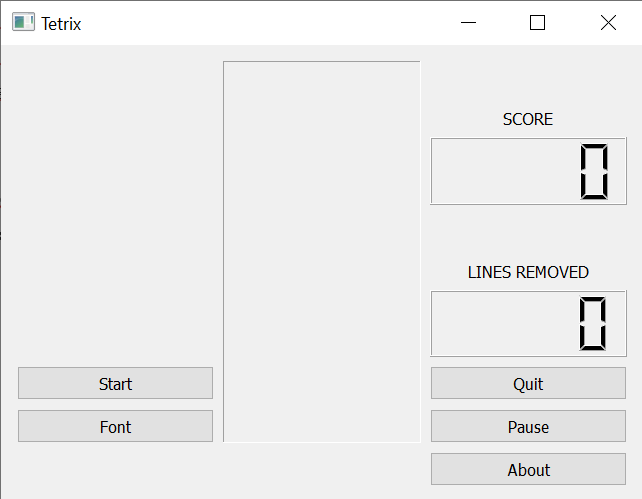
settings.setValue("VIEWF", newFont);

}

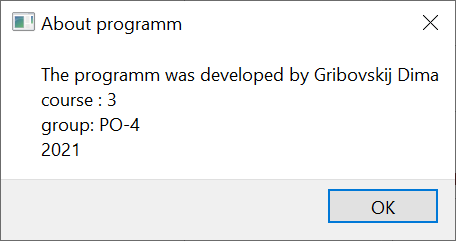
}

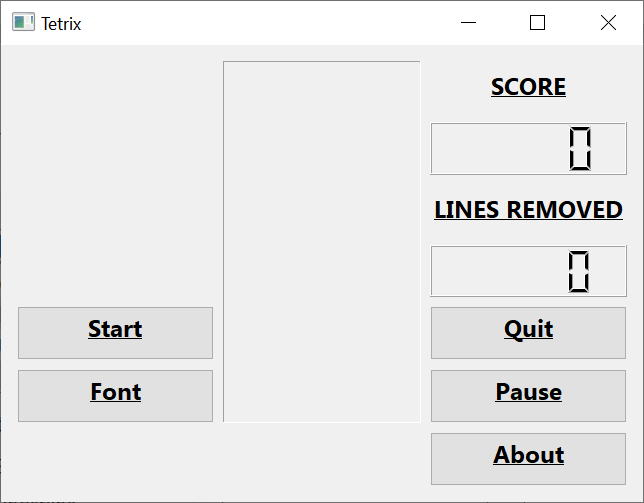
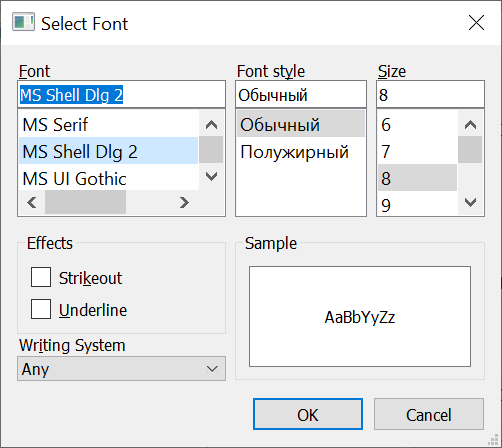
connect(fontButton, &QPushButton::clicked, board, &TetrixBoard::changeFont);

**Результат:**

****

Добавлены кнопки для изменения шрифта, о программе.





Вывод: Изучил создание динамических библиотек в Qt.