**Занятие 3**

Давайте попробуем воспользоваться знаниями, полученными после прохождения занятий 1 и 2, чтобы собрать небольшую программу на одном из самых популярных GUI фреймворков QT.

Стоит отметить, что существует множество нюансов при сборке программ, написанных с использованием QT, которые мы рассматривать не станем, но некоторые из таких нюансов мы посмотрим: QT имеет собственную систему сборки QMake, QT использует ряд специфичных CMake опций в своей работе.

Задание

1. Скачайте и установить QT.
2. Давайте запишем простую программу в файле main.cpp.

#include <QtWidgets>

int main(int argc, char\*\* argw)

{

    QApplication app(argc, argw);

    QWidget wgt;

    QWidget\* pwgt1 = new QWidget(&wgt);

    QPalette pal1;

    pal1.setColor(pwgt1->backgroundRole(), Qt::blue);

    pwgt1->setPalette(pal1);

    pwgt1->resize(100, 100);

    pwgt1->move(25, 25);

    pwgt1->setAutoFillBackground(true);

    QWidget\* pwgt2 = new QWidget(&wgt);

    QPalette pal2;

    pal2.setBrush(pwgt2->backgroundRole(), QBrush(QPixmap(":/stone.jpg")));

    pwgt2->setPalette(pal2);

    pwgt2->resize(100, 100);

    pwgt2->move(75, 75);

    pwgt2->setAutoFillBackground(true);

    wgt.resize(200, 200);

    wgt.show();

    return app.exec();

}

1. В проекте мы используем файл ресурсов, он нужен чтобы мы могли добавить внешние данные в нашу программу, например, в нашем случае мы используем изображение stone.jpg.



Давайте сформируем наш файл ресурсов res.qrc, поместив его в одной директории с CMakeLists.txt.

<!DOCTYPE RCC><RCC version="1.0">

<qresource>

<file alias="stone.jpg">C:/path/to/resources/stone.jpg</file>

</qresource>

</RCC>

Внесите необходимые поправки в этот код указав путь к вашему файлу stone.jpg.

1. Приступим к заполнению CMakeLists.txt файла.

cmake\_minimum\_required(VERSION 3.17)

project(lesson3)

add\_executable(lesson3

main.cpp

res.qrc)

В компиляцию будет включен файл ресурсов res.qrc. Чтобы CMake корректно обработал добавляемые ресурсы нужно указать опцию set(CMAKE\_AUTORCC ON) перед add\_executable(lesson3 main.cpp res.qrc).

По умолчанию QT библиотеки не включены в системные пути. Мы можем указать к ним путь и добавить в наш проект следующим образом(Следите за путями к проекту они могут отличатся):

set (CMAKE\_PREFIX\_PATH "C:\\Qt\\5.12.10\\mingw73\_64\\")

find\_package(Qt5Widgets)

Остается последний этап подключения наших библиотек, добавленных командами выше:

target\_link\_libraries(lesson3 ${Qt5Widgets\_LIBRARIES})

Финальная версия CMakeLists.txt файла выглядит так:

cmake\_minimum\_required(VERSION 3.17)

project(lesson3)

set (CMAKE\_PREFIX\_PATH "C:\\Qt\\5.12.10\\mingw73\_64\\")

set(CMAKE\_AUTORCC ON)

find\_package(Qt5Widgets)

add\_executable(lesson3

                main.cpp res.qrc)

target\_link\_libraries(lesson3 ${Qt5Widgets\_LIBRARIES})

1. Начнем сборку проекта знакомыми нам командами



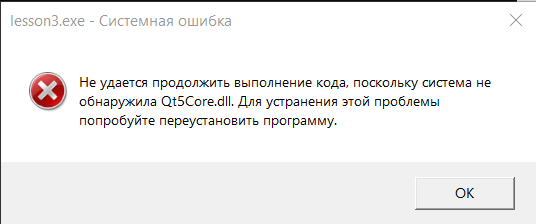
Напоминаю эта команда может отличаться в зависимости от используемого вами компилятора и ОС, в моем случае это MinGW.



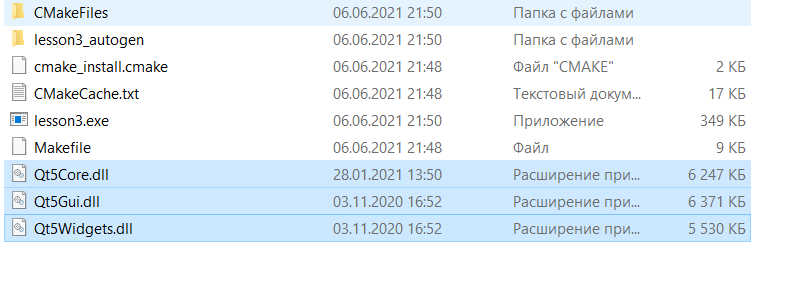
[100%] Built target lesson3 – сборка прошла успешно, можем запустить программу.

1. Запуск программы

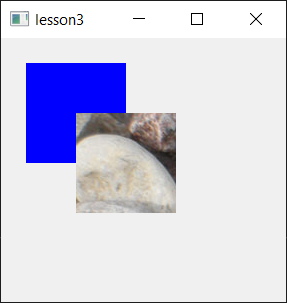
При запуске программы могут возникнуть ошибки, их мы разбирали на занятии 2.



Разыщите необходимые dll в файлах QT как правило они лежат в каталоге bin.



Запустите приложение



P. S. Если при работе с QT вы использует qt-препроцессор, например, Q\_OBJECT, указывайте в CMakeLists.txt следующую опцию.

set(CMAKE\_AUTOMOC ON)

Я рекомендую заглянуть в документацию QT, чтобы ознакомится для чего это нужно.