Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт «Электронных и информационных систем»

Кафедра «Информационных технология и систем»

Лабораторная работа №1

**«Создание средства просмотра изображений»**

Отчёт по дисциплине:

«Распознавание образов и обработка изображений»

Выполнил студент группы 8091:

\_\_\_\_\_\_\_\_\_\_ / Бочкарев Б. А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Принял преподаватель:

\_\_\_\_\_\_\_\_\_\_ / Назаров А. Г.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Великий Новгород

2022

**Цель лабораторной работы**

Подготовка рабочего программного окружения для выполнения лабораторных работ.

Разработать и реализовать программу просмотра изображений.

**Программное окружение**

Для выполнения данной лабораторной работы было собрано следующее окружение:

* Язык программирования: C++
* Операционная система: MS Windows 10
* Компилятор: MinGW 64-bit
* Среда программирования: QtCreator с версией Qt 5.15.2

**Ответы на вопросы**

Вопрос 1: Мы используем окно сообщения, чтобы сообщить пользователям, что они уже просматривают первое или последнее изображение, в то время как они пытаются просмотреть предыдущее перед первым изображением или следующее после последнего изображения. Но есть и другой способ справиться с этой ситуацией — отключить *prevAction*, когда пользователи просматривают первое изображение, и отключить *nextAction*, когда пользователи просматривают последнее изображение. Как это реализуется?

Ответ: Когда доходим до последнего изображения, кнопка *nextAction* ничего не выводит, аналогично с *prevAction*.

Вопрос 2: Наши пункты меню или кнопки инструментов содержат только текст. Как мы можем добавить к ним изображение значка?

Ответ: Для начала нужно добавить файл ресурсов в проект, он будет с расширением qrc (в него мы будем помещать файлы с иконками), далее, использовав метод *setIcon* и класс *Qicon*, мы сможем добавить иконки к любому действию.

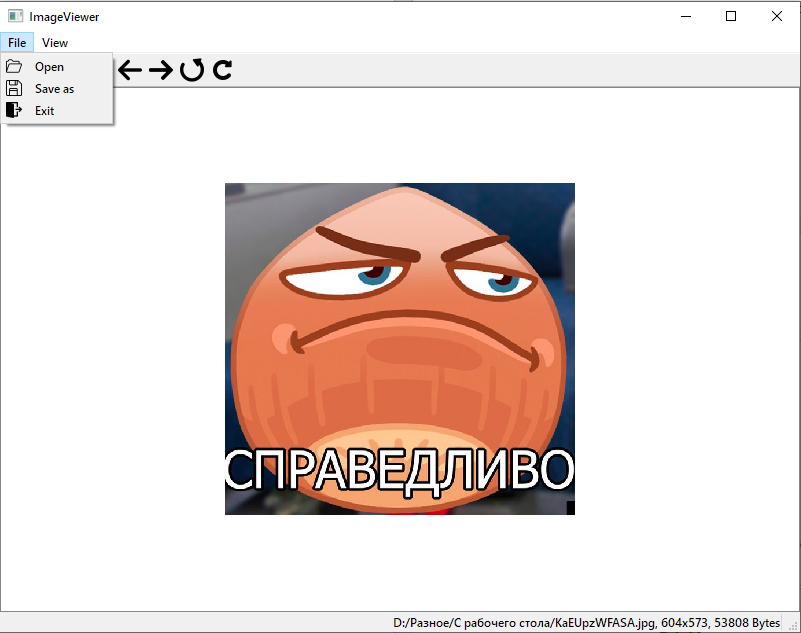


Рисунок 1 Иконки File

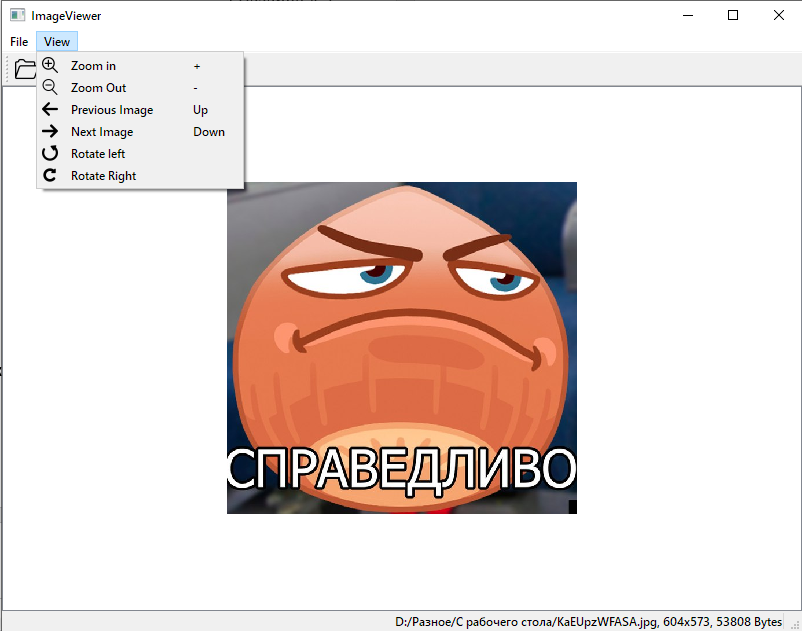


Рисунок 2 Иконки View

Вопрос 3: Мы используем *QGraphicsView.scale* для увеличения или уменьшения изображения, но как повернуть изображение?

Ответ: Для поворота изображения будем использовать 2 функции: *rotateImageRight* и *rotateImageLeft*

Примеры:

void MainWindow::**rotateImageRight**()

{

imageView->rotate(90);

}

void MainWindow::**rotateImageLeft**()

{

imageView->rotate(-90);

}

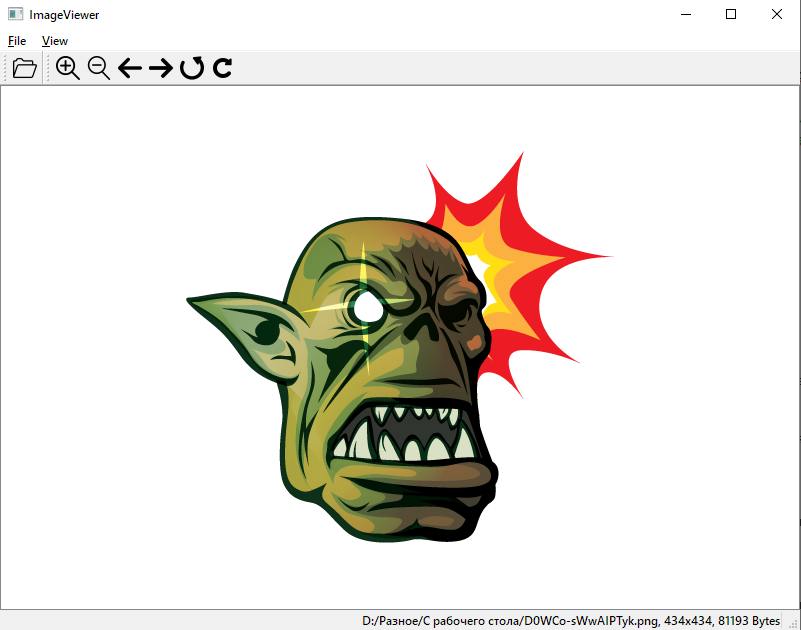


Рисунок 3 Оригинальное положение Рисунок 4 Вращение против часовой стрелки



Рисунок 5 Вращение по часовой стрелке Рисунок 6 Демонстрация вращения

Вопрос 4: Что делает *moc*? Какие действия выполняют макросы *SIGNAL* и *SLOT*?

Ответ: **Метаобъектный компилятор** (сокр. МОК) (от англ.**Meta Object Compiler** или MOC) предназначен для генерации метаобъектного сопутствующего кода на основе библиотеки Qt. МОК считывает одно (или более) определение C++ класса из заголовочного файла языка C++ или исходного файла программы и генерирует один исходный файл на языке С++, содержащий метаобъектную информацию для классов. Получившийся исходный файл, сгенерированный МОК, должен быть скомпилирован и слинкован с реализацией класса (или может быть включен при помощи инструкции #include в исходный файл с описанием класса).

**Сигналы** (signals) - это методы, которые в состоянии осуществлять пересылку сообщений. Сигналы определяются в классе, как обычные методы, но без реализации. Они являются прототипами методов, содержащихся в заголовочном файле определения класса. Всю дальнейшую заботу о реализации кода для этих методов берет на себя препроцессор. Методы сигналов не должны возвращать каких-либо значений, поэтому перед именем метода всегда должно стоять void.

Сигнал не обязательно соединять со слотом. Если соединения не произошло, то он просто не будет обрабатываться.

**Слоты** (slots) — это методы, которые присоединяются к сигналам. По сути, они являются обычными методами. Основное их отличие состоит в возможности принимать сигналы. Как и обычные методы, они определяются в классе как *public*, *private* или *protected*. Соответственно, перед каждой группой слотов должно стоять одно из ключевых слов *private slots:*, *protected slots:* или *public slots:*

**Исходный код всех исходных файлов и файлов проекта**

Исходный код всех исходных файлов и файлов проекта доступен по ссылке: https://github.com/borya178/Pattern\_recognition\_and\_image\_processing

**Дистрибутив программы**

Дистрибутив был создан с помощью **windeployqt.exe.**

Дистрибутив был протестирован на виртуальной машине с ОС MS Windows 7, созданной с помощью программы VirtualBox.

Дистрибутив проекта в архиве доступен по ссылке: https://github.com/borya178/Pattern\_recognition\_and\_image\_processing  
Запустить программу можно, нажав на файл **Lab1.exe**.

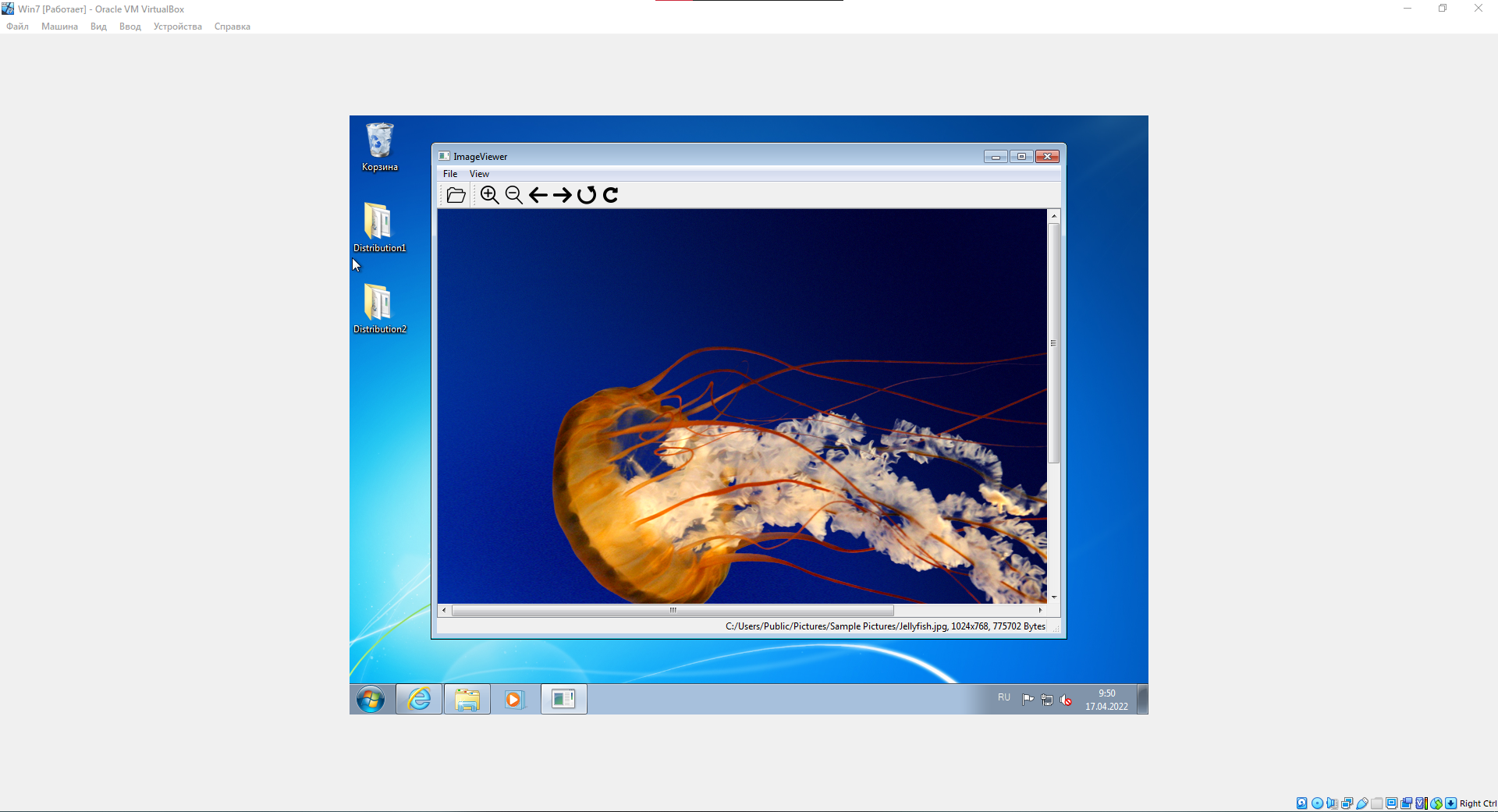


Рисунок 7 Демонстрация работы дистрибутива на другом компьютере.

**Вывод**

В ходе выполнения лабораторной работы я познакомился со средой Qt и разработал, и реализовал программу просмотра изображений.