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:

\_\_\_\_\_\_\_\_\_\_ / Баранов А. С.

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

Принял:

\_\_\_\_\_\_\_\_\_\_ / 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 LTSC
* Компилятор: MinGW 64-bit
* Среда программирования: Qt 5.12

1. **Ограничение пролистывания изображений**

Во время просмотра изображений пользователь может нажать кнопку пролистать влево при просмотре первого изображения в списке, либо нажать пролистать вправо при просмотре последнего изображения в списке. Есть два пути решения: реализовать пролистывание по кругу, показывать окно предупреждения о крайнем положении в списке, либо просто ограничить действие кнопки перелистывания при положении в начале или конце списка.

Мы воспользуемся третьим вариантом. Для этого нужно отключить **prevAction**, когда пользователь находится в начале списка, и отключить **nextAction**, когда пользователь находится в конце списка, т.е. кнопки не будут ничего выводить.

1. **Добавление значков действий**

Для приятного использования программы следует добавить удобный интерфейс, а именно значки действий.

Для этого добавляем файл ресурсов с расширением .qrc, помещаем изображения в формате .png размером 512x512 для каждого действия, далее через метод **setIcon**и класс **Qicon** связываем иконки с каждым соответственным действием. Результат изображен на рисунке 1.

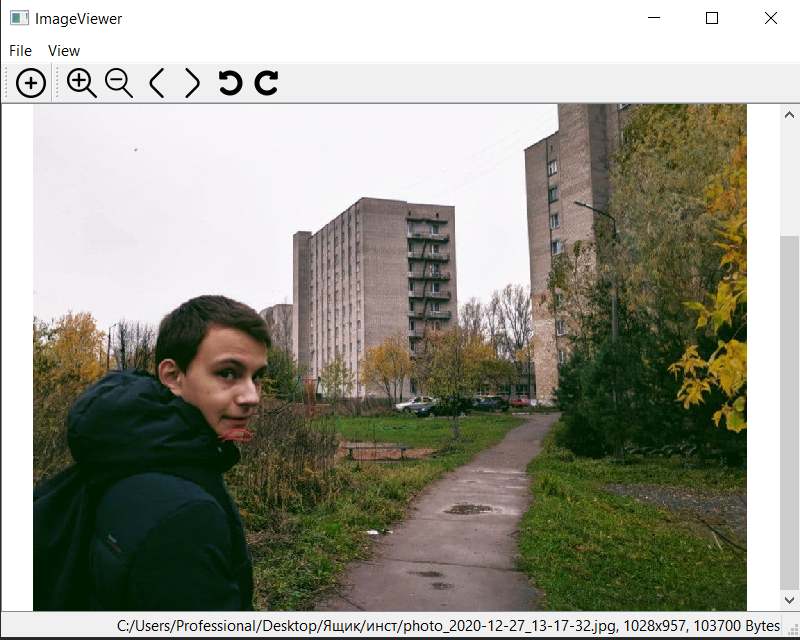


Рисунок 1- Значки

1. **Поворот и зум изображения**

Для зума(увеличение – уменьшения) изображения используется **QGraphicsView.scale** (рис. 2). Для поворота используются функции **rotateImageRight** и **rotateImageLeft** (рис. 3).

Примеры:

void MainWindow::**rotateImageRight**()

{

imageView->rotate(90);

}

void MainWindow::**rotateImageLeft**()

{

imageView->rotate(-90);

}

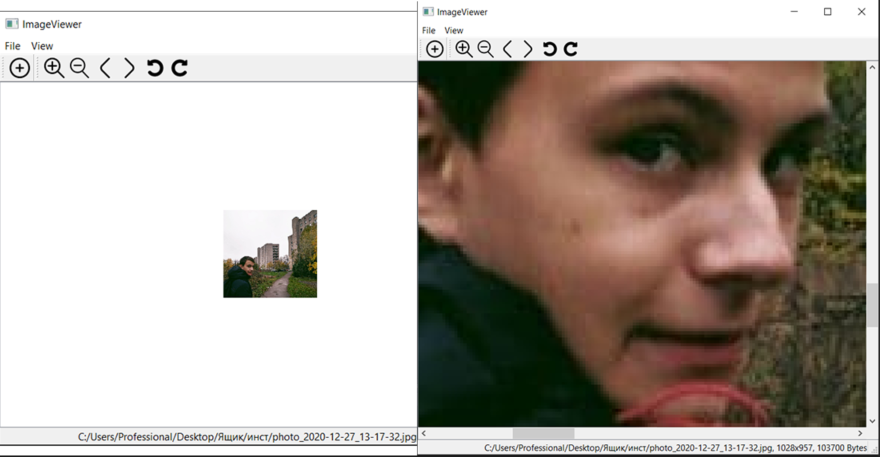


Рисунок 2 – Зум

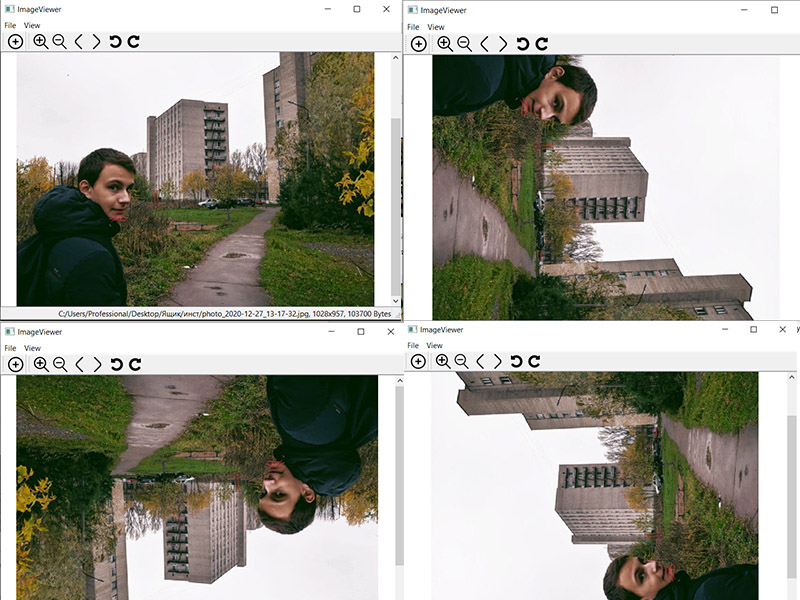


Рисунок 3 – Поворот изображения

1. **Moc, SIGNAL, SLOT**

* Мета-объектный компилятор (***moc***) – программа, которая обрабатывает расширения C++ от Qt. Инструмент moc читает заголовочный файл C++. Если он находит одно или более объявлений классов, которые содержат макрос Q\_OBJECT, то он порождает файл исходного кода C++, содержащий мета-объектный код для этих классов. Файл исходного кода C++, сгенерированный moc, должен компилироваться и компоноваться с помощью реализации класса. Другими словами MOC для каждой библиотеки где есть классы с макросом Q\_OBJECT создаёт файлы moc\_\*.cpp, содержащие таблицу виртуальных методов, чтобы нормально отрабатывались сигналы и слоты.
* В программировании графического интерфейса, когда мы меняем один виджет, мы часто хотим что бы другой виджет получил об этом уведомление. В общем случае, мы хотим что бы объекты любого типа могла общаться с другими. Для этих целей Qt использует механизм ***сигналов*** и ***слотов****.*

Сигнал вырабатывается когда происходит определенное событие. Слот это функция, которая вызывается в ответ на определенный сигнал. Преимущество этого подхода (в отличие от обратного вызова через указатель) в том, что:

1. Типобезопасность, т.е. можно не переживать об ошибках вызванными несовпадением типов;
2. Каждый класс, унаследованный от QObject, может иметь любое количество сигналов и слотов;
3. Сообщения, посылаемые посредствам сигналов могут иметь множество аргументов любого типа;
4. Сигнал можно соединять с различным количеством слотов, при этом высылаемый сигнал поступит ко всем подсоединенным слотам;
5. При уничтожении объекта происходит автоматическое разъединение всех сигнально-слотовых связей; это гарантирует, что сигналы не будут высылаться к несуществующим объектам;
6. Соединение сигналов и слотов можно производить в любой точке приложения.

**Исходный код проекта и дистрибутива**

<https://github.com/lexjesus/computer-zrenie/tree/main/Lab1>

**Вывод**

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