Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин
Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему
«Видеоплеер»

БГУИР КР 1-40 02 01 310 ПЗ

Студент К. А. Каражан
Руководитель Е. В.Богдан

МИНСК 2023

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ

(подпись)
2023 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Каражан Ксении Алексанровне*

1. Тема проекта *«Видеоплеер»*
2. Срок сдачи студентом законченного проекта *11 декабря 2023 г.*
3. Исходные данные к проекту: картинки jpg для иконок приложения в папке icon
4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)
   1. Введение.
   2. Задание.
   3. Обзор литературы.
   3.1. Обзор методов и алгоритмов решения поставленной задачи.
   4.Функциональное проектирование.
   4.1. Структура входных и выходных данных.
   4.2. Разработка диаграммы классов.
   4.3. Описание классов.
   5.Разработка программных модулей.
   5.1. Разработка схем алгоритмов (два наиболее важных метода).
   5.2. Разработка алгоритмов (описание алгоритмов по шагам для двух методов).
   6.Результаты работы.
   7.Заключение
   8.Литература
   9.Приложения
5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)
   1. *Диаграмма классов.*
   2. *Схема алгоритма loadLastVideoPosition().*
   3. *Схема алгоритма SaveCurrentPlaylist().*
6. Консультант по проекту (с обозначением разделов проекта) Е.В.Богдан

7. Дата выдачи задания 15 *сентября 2023 г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоёмкости отдельных этапов):

1. Выбор задания. Разработка содержания пояснительной записки. Перечень графического материала – 15 %;
разделы 2, 3 – 10 %;
разделы 4 к – 20 %;
разделы 5 к – 35 %;
раздел 6,7,8 – 5 %;
раздел 9 к – 5%;
оформление пояснительной записки и графического материала к 11.12.23 – 10 %
Защита курсового проекта с 21.12 по 28.12.23г.

РУКОВОДИТЕЛЬ Е.В.Богдан

Задание принял к исполнению

К.А Каражан

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Изучение принципов объектно-ориентированного программирования (ООП) предоставляет разработчикам эффективные инструменты для организации кода и разработки масштабируемых приложений. Ознакомление с принципами позволяет разработчикам расширить свои навыки, обеспечивая более интуитивный и гибкий подход к созданию программного обеспечения.

Научное внимание текущего курсового проекта направлено на разработку видеоплеера на языке программирования C++ с использованием фреймворка Qt, с акцентом на основах объектно-ориентированного программирования. Современные технологии разработки на C++ включают использование стандартных библиотек, инструментов разработки и передовых методов программирования, с учётом применения объектно-ориентированного подхода для создания систем, обладающих модульностью и легкочитаемостью.

В современное время, где требования к программному обеспечению непрерывно растут, объектно-ориентированное программирование (ООП) становится неотъемлемой частью подготовки профессиональных разработчиков.

Объектно-ориентированное программирование (ООП), позволяющее создавать программы, основанные на объектах, а не на процедурах. Это позволяет создавать более гибкие и масштабируемые Template Library), которые предоставляют широкий набор готовых компонентов для решения различных задач.

Использование современных инструментов разработки, таких как

среды разработки (IDE), отладчики и компиляторы, которые позволяют ускорить процесс разработки и улучшить качество кода.

Видеоплееры играют важную роль в современном мире, предоставляя пользователям возможность просматривать и воспроизводить контент. Они имеют множество преимуществ и важности, которые делают их неотъемлемой частью нашего повседневного использования. Несколько ключевых важных особенностей:

Поддержка различных форматов видео: Видеоплееры могут поддерживать множество форматов видео, таких как TS, MTS, M2TS, MXF, MP4, M4V, MOV, MPG, MPEG, AVI, FLV, RMVB, WMV, ASF, MKV, SWF, VOB, OGM, WTV, WebM и другие.

Высокое качество видео: Видеоплееры поддерживают воспроизведение видео высокого разрешения, такого как Full HD, 4K, 5K и даже 8K.

Управление воспроизведением: Видеоплееры предоставляют пользователям возможность управлять воспроизведением видео, включая перемотку вперёд и назад, переключение на полноэкранный режим, выбор звуковой дорожки и другие функции.

Совместимость с различными операционными системами:

Видеоплееры обычно совместимы с различными операционными системами, такими как Windows, Mac и другие. Это позволяет пользователям использовать видеоплееры на любом устройстве, независимо от его операционной системы.

Бесплатность или доступная цена: Многие видеоплееры доступны бесплатно или имеют доступную цену, что делает их доступными для широкого круга пользователей.

Расширенные функции: Некоторые видеоплееры, такие как PotPlayer и VLC Player, предлагают расширенные функции, такие как синхронизация субтитров, фильтры видео/аудио, изменение скорости воспроизведения и другие.

Экономичность: Большинство видеоплееров бесплатны или имеют доступную цену, что делает их доступными для широкого круга пользователей. Некоторые из них предлагают дополнительные функции, такие как блокировка рекламы, за которую они взимают плату 5.

Цель настоящего курсового проекта заключается в исследовании особенностей объектно-ориентированного программирования, в том числе анализ современных технологий и их применение при создании видеоплеера с высоким уровнем абстракции и гибкости.

Результатом проекта будет создание функционального видеоплеера, предоставляющего пользователю возможность взаимодействия с мультимедийным контентом и обладающего интуитивным интерфейсом для удобного управления функциональностью.

1. ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня C++, овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня C++[1]. Разработать видеоплеер с использование среды разработки Qt. Понимание основных принципов ООП:

При разработке используется Qt, кроссплатформенный фреймворк для разработки приложений на C++. Он предоставляет разработчикам множество инструментов и библиотек для создания графического интерфейса пользователя, работы с сетью, базами данных, мультимедиа и многого другого.

Создание удобного интерфейса:

- Разработать привлекательный и интуитивно понятный интерфейс для пользователя;
- Обеспечить простоту использования основных функций (воспроизведение, пауза, перемотка и т. д.).
- Поддержка различных форматов видео:
- Обеспечить воспроизведение видео в различных форматах (MP4, AVI, MKV и т. д.).

Масштабируемость и адаптивность:

- Обеспечить масштабируемость для поддержки различных разрешений видео.

Контроль за воспроизведением:

- Добавить функциональность управления скоростью воспроизведения.
- Реализовать возможность выбора конкретного времени в видео.

Плейлисты:

- Возможность создать/сохранить плейлист
- Возможность включить видео непосредственно из плейлиста

2. ОБЗОР ЛИТЕРАТУРЫ

Перед началом разработки следовало ознакомиться с основными концепциями и инструментами Qt C++, такими как сигналы и слоты, система событий, элементы управления интерфейса пользователя (QWidget, QLabel, QPushButton и т. д.), а также основами работы с мультимедийными данными.

Ключевым этапом является изучение классов, ответственных за мультимедийное воспроизведение, таких как QMediaPlayer. Необходимо углублённое понимание основных функций этих классов, возможностей настройки воспроизведения, обработки событий и управления медиаресурсами.

Важным аспектом является анализ документации по работе с видео- и аудиокодеками с целью обеспечения поддержки требуемых форматов. Кроме того, рекомендуется ознакомиться с образцами кода, демонстрирующими создание пользовательского интерфейса видеоплеера с использованием Qt, для усвоения эффективных методов интеграции компонентов и обеспечения удобства использования приложения.

Документация Qt:

Getting Started with Qt: Раздел "Getting Started" в документации предоставляется информация о том, как установить Qt, настроить среду разработки и создать простое приложение.

Qt Multimedia Example: Примеры видео- и аудиоплееров.

Overview: Введение в фреймворк, его основные концепции и принципы. Создание графического интерфейса: Qt Widgets[9]: Информация о виджетах Qt, базовых элементах управления, таких как кнопки, поля ввода и другие.

Примеры кода и Учебные проекты[7]: Qt Examples: Обширный набор примеров кода для различных компонентов Qt.Qt Tutorials: Учебные проекты и туториалы, позволяющие освоить различные аспекты фреймворка.

Форумы и Сообщества: Qt Forum: Онлайн-форумы, где разработчики могут задавать вопросы, делиться опытом и получать поддержку от сообщества Qt.

2.1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

В ходе разработки программы были задействованы разнообразные возможности языка программирования C++, которые будут подробно рассмотрены далее. Для решения поставленной задачи был выбран язык программирования C++ и применена методология объектно-ориентированного программирования [4].

Программа включает в себя механизм обработки исключительных ситуаций. Этот механизм предназначен для описания реакции программы на возможные ошибки, которые могут возникнуть в процессе её выполнения и привести к невозможности или бессмысленности дальнейшей обработки основного алгоритма программы [5].

Касательно интерфейса пользователя, в коде создаются элементы управления, такие как кнопки воспроизведения, паузы, остановки, перемотки и отображения плейлиста. Дополнительно проведена настройка стилей и расположения элементов интерфейса для улучшения пользовательского опыта.

В области мультимедийной обработки использован класс QMediaPlayer для эффективного управления воспроизведением видео. Реализована обработка видео- и аудиокодеков, обеспечивающая поддержку различных форматов мультимедийных файлов. Также предусмотрен механизм обработки исключительных ситуаций для управления ошибками в процессе выполнения программы.

Относительно настройки воспроизведения, реализован слайдер для регулировки скорости воспроизведения, а также механизмы управления громкостью и перемоткой видео вперёд и назад.

В сфере работы с плейлистом создан и настроен плейлист для хранения и управления списком видеофайлов. Реализованы функции загрузки и сохранения плейлиста с использованием диалоговых окон для выбора файла.

Система сохранения и восстановления состояния позволяет сохранять текущее состояние видеоплеера, включая последний открытый файл и его позицию, с последующим восстановлением при повторном запуске программы.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

3.1 Входные данные

Для работы программы и отображения иконок были сделанные специальные jpg картинки, помещённые в папку icon.

3.2 Разработка диаграммы классов

Диаграмма классов данной работы показана в приложении А.

3.3 Описание классов

3.3.1 Класс `MainWindow`

`MainWindow` – Класс предназначен для управления основным окном видеоплеера. Он инкапсулирует в себе логику и элементы управления, наследует функции и характеристики QMainWindow.

Описание полей класса:

`Ui::MainWindow *ui` - указатель на объект Ui. Предназначен для работы с виджетами.

`videoPlayer (QMediaPlayer)` – Указатель на объект класса QMediaPlayer, отвечающий за воспроизведение аудио и видео.

`videoWidget (QVideoWidget)` - Указатель на объект класса QVideoWidget, представляющий виджет для отображения видео.

`videoContainer (QWidget)` - Указатель на объект класса QWidget, используемый в качестве контейнера для видео.

`controlWidget (QWidget)` - Указатель на объект класса QWidget, представляющий виджет для управления воспроизведением.

`slider (QSlider)` - Указатель на объект класса QSlider, представляющий слайдер для управления временем воспроизведения видео.

`volume (QSlider)` - Указатель на объект класса QSlider, представляющий слайдер для управления уровнем громкости.

`speedSlider (QSlider)` – Указатель на объект класса QSlider, представляющий слайдер для управления скоростью воспроизведения.

`Mduration (qint64)` - Переменная типа qint64, содержащая общую длительность текущего видео.

`startTimeLabel (TimeLabel)` - Указатель на объект класса TimeLabel, представляющий виджет для отображения текущего времени

воспроизведения.

`endTimeLabel (TimeLabel)` - Указатель на объект класса TimeLabel, представляющий виджет для отображения общей длительности видео.

`playlistWidget (PlaylistWidget)` - Указатель на объект класса PlaylistWidget, представляющий виджет для отображения и управления плейлистом.

`isPlaylistVisible (bool)` - Флаг типа bool, указывающий, видим ли в данный момент плейлист.

`videoPositions (QMap<QString, qint64>)` - Контейнер QMap, содержащий информацию о позициях воспроизведения для каждого видео в плейлисте.

`lastVideoFilePath (QString)` - Строковая переменная, содержащая путь к последнему выбранному видеофайлу.

`appSettings (QSettings)` - Указатель на объект класса QSettings, представляющий настройки приложения.

Описание методов:

`openFile()` - Открывает диалоговое окно для выбора видеофайла.

`loadVideo(const QString &videoFilePath)` - Загружает выбранный видеофайл для воспроизведения.

`onPlaylistItemClicked(const QString &videoFilePath)` - Обрабатывает событие выбора элемента в плейлисте, начинает воспроизведение выбранного видеофайла.

`togglePlayPause()` - Переключает состояние воспроизведения/паузы видео.

`on_btnStop_clicked()` - Останавливает воспроизведение видео.

`on_btnBackward_clicked()` - Возвращает видео на 5 секунд назад.

`on_btnForward_clicked()` - Перематывает видео на 5 секунд вперёд.

`on_btnShowPlaylist_clicked()` - Скрывает/показывает плейлист.

`loadPlaylist()` - Загружает плейлист с сохранёнными видеофайлами.

`savePlaylist(const QString &playlistPath, const QList<QString> &filePaths)` - Сохраняет текущий плейлист в указанный файл.

`saveCurrentPlaylist()` - Сохраняет текущий плейлист.

`openFolderAndAddToPlaylist()` - Открывает диалоговое окно для выбора папки и добавляет видеофайлы из этой папки в плейлист.

`onSpeedSliderChanged(int value)` - Обрабатывает изменение положения слайдера скорости воспроизведения.

`setPlaybackSpeed025()...setPlaybackSpeed2()` - Устанавливает соответствующую скорость воспроизведения.

`onVolumeSliderChanged(int value)` - Обрабатывает изменение положения слайдера громкости.

`increaseVolume()...offVolume()` - Управляют уровнем громкости.

`updateCurrentTime(qint64 position)` - Обновляет текущее время

воспроизведения видео.

`updateTotalDuration(qint64 duration)` - Обновляет общую длительность видео.

`saveLastVideoPosition()` - Сохраняет последнюю позицию воспроизведения видео.

`loadLastVideoPosition()` - Загружает последнюю позицию воспроизведения видео.

`closeEvent(QCloseEvent *event)` - Обрабатывает событие закрытия главного окна.

`closeApp()` - Завершает выполнение приложения.

`showHelpMessage()` - Отображает справочное сообщение о приложении.

`MainWindow(QWidget parent = nullptr)` - конструктор класса.

3.3.2 Класс `PlaylistWidget`

`PlaylistWidget` - Класс представляет виджет плейлиста, который наследует от QListWidget и предоставляет дополнительную функциональность для управления плейлистом и обработки событий кликов на элементах.

Описание полей класса:

`playlist (QMediaPlaylist)` - Указатель на объект класса QMediaPlaylist, представляющий плейлист для хранения и управления списком аудио- и видеофайлов.

`playlistItemClicked(const QString &videoFilePath)` – Сигнал, отправляемый при клике на элемент плейлиста. Передаёт полный путь к выбранному видеофайлу.

`handleItemClick(QListWidgetItem item)` – Слот, обрабатывающий событие клика на элемент плейлиста. Извлекает полный путь к выбранному видеофайлу из элемента и отправляет соответствующий сигнал.

QMediaPlaylist.

Описание методов:

`PlaylistWidget(QWidget parent = nullptr)` – Конструктор класса.

`addItem(const QString &folderPath, const QString &videoFileName)` - Метод для добавления элемента в плейлист. Принимает путь к папке и имя видеофайла. Создаёт полный путь к видеофайлу и добавляет его в плейлист.

`getPlaylist()` - Метод для получения текущего плейлиста в виде QStringList.

3.3.3 Класс `timelabel`

`timelabel` – Класс наследует функционал и элементы QLabel, который предназначен для отображения информации о времени в виде текста.

Описание полей класса:

`mediaPlayer (QMediaPlayer)` - Указатель на объект класса QMediaPlayer, представляющий плеер, с которым связан виджет.

`videoDuration (qint64)` - Переменная для хранения общей длительности текущего видеофайла в миллисекундах.

Описание методов:

`TimeLabel(QWidget parent = nullptr)` - Конструктор класса.

`setMediaPlayer(QMediaPlayer player)` - Метод для установки плеера, с которым будет связан TimeLabel.

`updateCurrentTime()` - Метод для обновления отображаемого времени текущего воспроизведения. Извлекает текущую позицию видеоплеера и обновляет текст виджета с учётом формата времени.

`updateTotalDuration()` - Метод для обновления отображаемой общей длительности видео. Извлекает общую длительность текущего видеофайла из медиаплеера и обновляет текст виджета с учётом формата времени.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Разработка схем алгоритмов

Метод `loadPlaylist()` загружает позицию последнего просмотренного видео. Схема метода `loadPlaylist()` показана в приложении Б.

Метод `openFolderAndAddToPlaylist()` сохраняет текущий плейлист.

Схема метода `openFolderAndAddToPlaylist()` показана в приложении В.

4.2 Разработка алгоритмов

Метод `saveCurrentPlaylist()` класса `MainWindow`.

Шаг 1. Открытие диалога сохранения файла;

Шаг 2. Выводится диалоговое окно для выбора имени и места сохранения файла плейлиста;

Шаг 3. Используется метод QFileDialog::getSaveFileName;

Шаг 4. Получение пути к файлу плейлиста;

Проверяется, не пуст ли выбранный путь к файлу;

Шаг 5. Если путь не пуст, продолжаем; в противном случае, завершаем выполнение;

Шаг 6. Создаётся список строк (filePaths), в который добавляются полные пути к видеофайлам в плейлисте;

Шаг 7. Проход по каждому элементу виджета плейлиста (playlistWidget);

Шаг 8. Для каждого элемента извлекается закодированный путь к папке (encodedFolderPath) и имя видеофайла;

Шаг 9. Используется QDir для получения полного пути к видеофайлу и добавления его в список filePaths;

Шаг 10. Вызывается функция savePlaylist(playlistPath, filePaths), передавая путь к файлу плейлиста и список путей к видеофайлам;

Шаг 11. Выводится информационное сообщение, уведомляя пользователя о успешном сохранении плейлиста.

Метод `loadLastVideoPosition`() класса `MainWindow`.

Шаг 1: Создание объекта QSettings для доступа к настройкам приложения;

Шаг 2: Получение пути к последнему воспроизведённому видеофайлу из настроек;

Шаг 3: Проверка, что путь не пуст и файл существует;

Шаг 5: Запрос пользователя о загрузке последней позиции;

Шаг 6: Обработка ответа пользователя;

Шаг 7: Обновление переменной lastVideoFilePath перед загрузкой видео;

Шаг 8: Добавление видео в плейлист;

Шаг 9: Установка медиа и позиции;

Шаг 10: Воспроизведение видео;

Шаг 11: Вывод отладочной информации;

Шаг 12: Вывод информации об отказе пользователя от загрузки последней позиции;

Шаг 13: Вывод информации о том, что сохранённой позиции воспроизведения не найдено.

5. РЕЗУЛЬТАТ РАБОТЫ

На рисунке 5.1 изображена начало работы программы. При старте программы пользователь может открыть файл/видео/плейлист.

Рисунок 5.1 – Начало работы программы

На рисунке 5.2 показана работа плейлиста и видеоплеера. Пользователь может переключаться между видео с помощью нажатия на название.

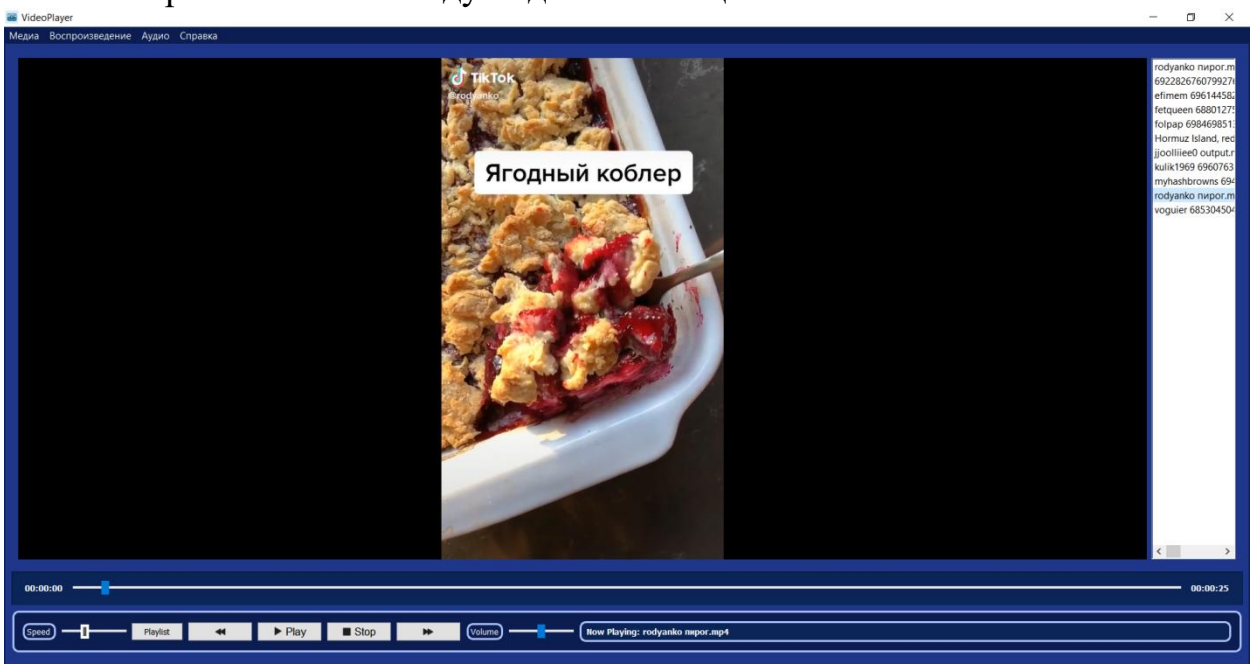

Рисунок 5.2 – Работа видеоплеера и плейлиста

На рисунке 5.3 отражена работа загрузки контрольной точки последнего просмотренного видеоролика. При нажатии на кнопку “Yes” функция срабатывает. При нажатии на кнопку “No” программа запускается как обычно.

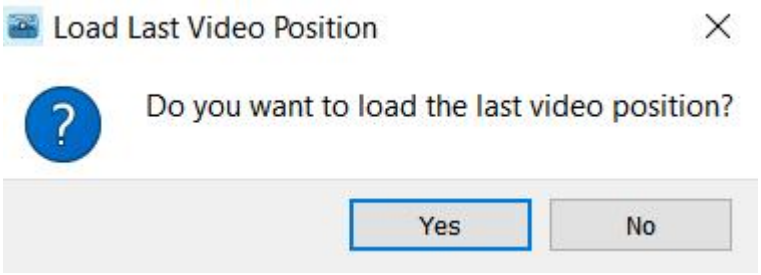

Рисунок 5.3 – Окно загрузки сохранённой точки

На рисунке 5.4 показана демонстрация функции загрузки плейлиста. При удачном выполнении выводится сообщение.

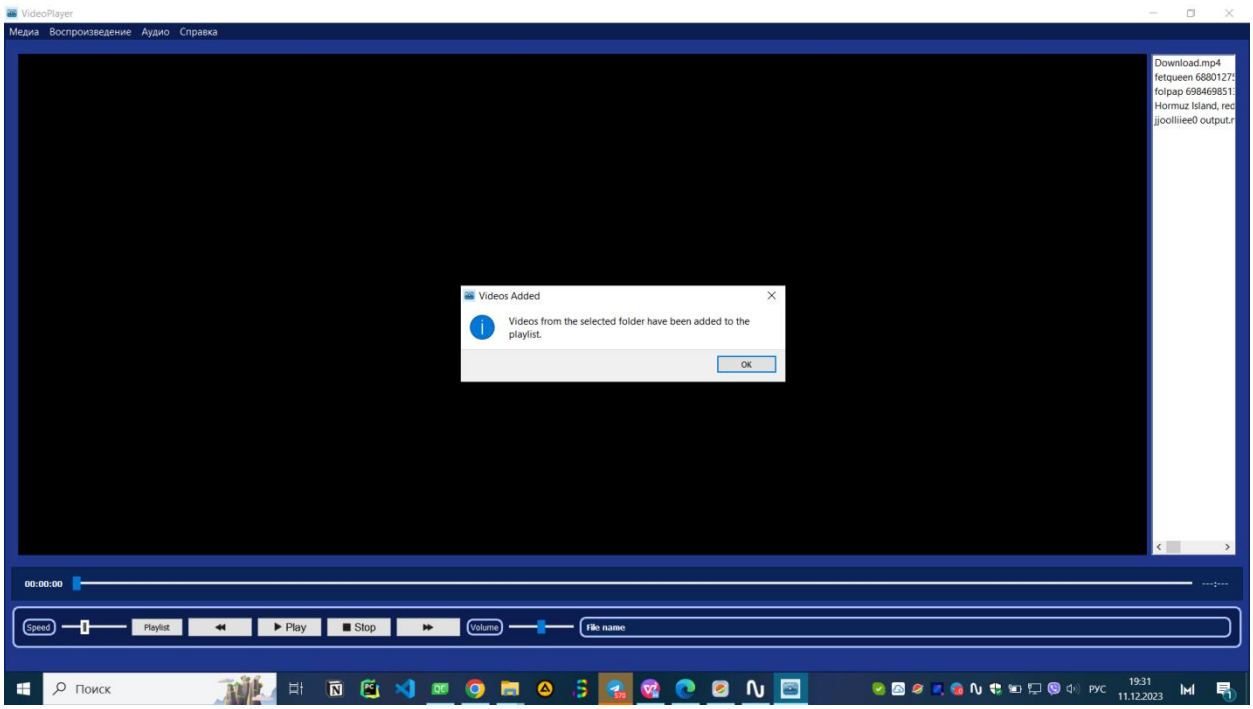

Рисунок 5.4 – Загрузка плейлиста

Рисунок 5.5 показывает возможности плеера: изменение скорости, скачок на 5 секунд вперёд и назад, пауза/плей и стоп.

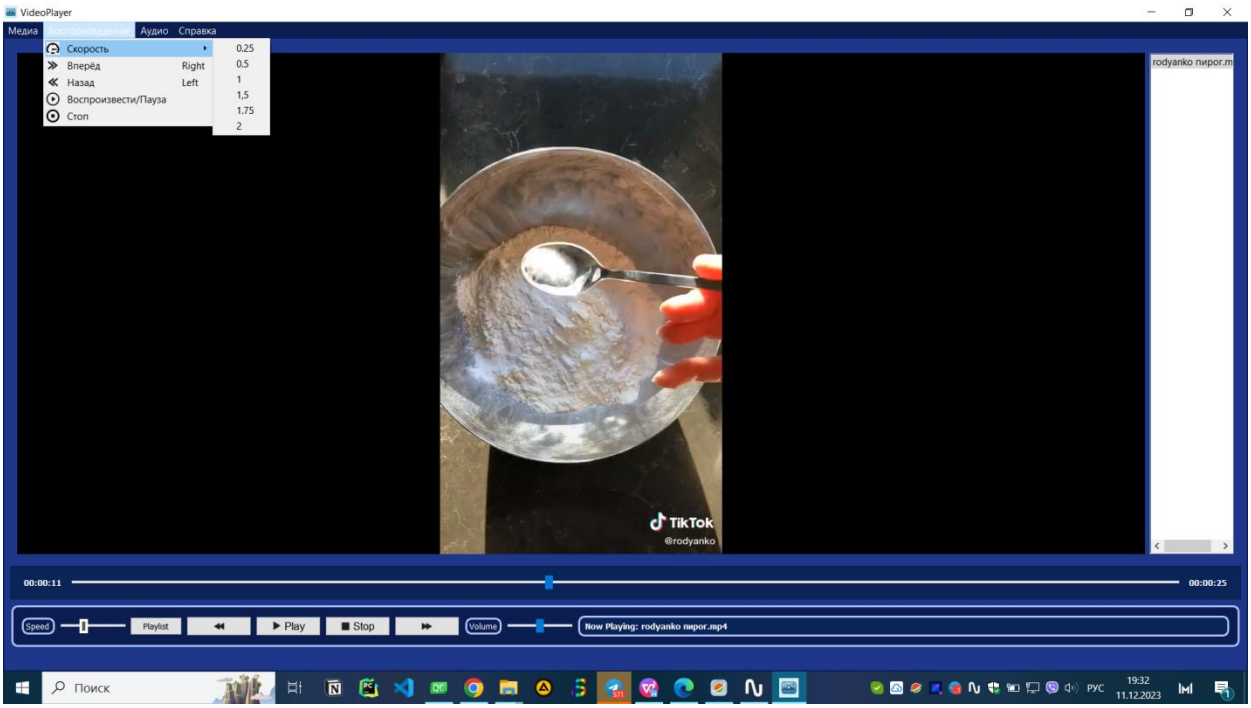

Рисунок 5.5 – Демонстрация функций

Рисунок 5.6 показывает функционал меню «Аудио». Пользователь может увеличивать, уменьшать, отключать звук.

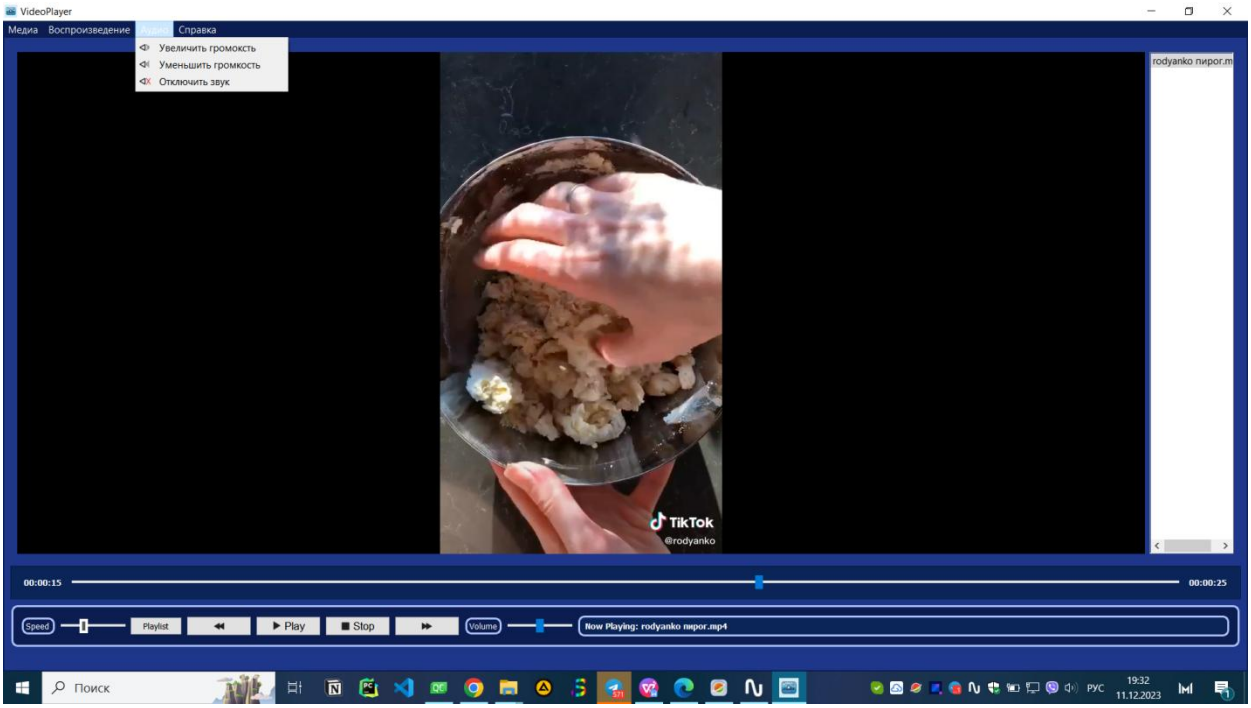

Рисунок 5.6 – Регулирование звука

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы, посвящённой разработке видеоплеера, были успешно достигнуты первоначально поставленные цели. Реализован видеоплеер с базовыми функциями, позволяющие просматривать видео и плейлист.

Помимо этого были приобретены знания в

В дальнейшем можно реализовать более продвинутые функции, например, просмотр субтитров, изменение цвета/контраста изображения.

Итог: Полученные знания в области работы с видеоданными, а также опыт в создании графического интерфейса на основе фреймворка Qt, предоставляют отличные возможности для будущих проектов. Этот опыт может быть успешно применен в разработке различных мультимедийных приложений, таких как видеоредакторы, фильтры видеоизображений и другие приложения, связанные с обработкой видеоданных.

Основанный на данной работе опыт и знания позволяют в дальнейшем изучить область мультимедийных технологий и программирования, открывают перспективы для участия в проектах, направленных на улучшение пользовательского опыта в области мультимедийных приложений.

В ходе выполнения курсовой работы произошло значительное расширение навыков программирования на языке C++. Также были освоены аспекты взаимодействия с графикой и разработки графических интерфейсов с применением фреймворка Qt. Полученные компетенции предоставляют устойчивую базу для последующих исследований и разработок в области компьютерного зрения и графики.

СПИСОК ЛИТЕРАТУРЫ

[1]"Объектно-ориентированное программирование на C++" Бьярн Страуструп
[2] "Язык программирования C++" Герберт Шилдт
[3] "C++ Primer" Липман, Лажойе, Му, Хопкинс
[4] "Алгоритмы. Построение и анализ" Кормен, Лейзерсон, Ривест, Штайн
[5] "Введение в алгоритмы" Кормен, Лейзерсон, Ривест, Штайн
[6] "Алгоритмы на C++" Роберт Седжвик, Кевин Уэйн
[7] " Qt Examples And Tutorials" [Электронный ресурс]. - Режим доступа: https://doc.qt.io/qt-5/qtexamplesandtutorials.html Дата доступа: 22.10.2023
[8] "Структуры данных и алгоритмы в C++" Robert Lafore
[9] "Qt.5.10 Профессиональное программирование на C++" Шлее

ПРИЛОЖЕНИЕ А
(обязательное)
Диаграмма классов

ПРИЛОЖЕНИЕ Б
(обязательное)
Схема метода `loadPlaylist()`

ПРИЛОЖЕНИЕ В
(обязательное)
Схема метода `openFolderAndAddToPlaylist()`

ПРИЛОЖЕНИЕ Г
(обязательное)
Код программы

ПРИЛОЖЕНИЕ Д
(обязательное)
Ведомость документов