МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ПО УЧЕБНОЙ ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

Студент
гр. БИС-22-1 _____ А.Е. Лазакович

Руководитель
Канд. хим. наук, доцент
старший преподаватель каф.
ИТС _____ Б.К. Васильев

Владивосток 2024

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАДИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на учебную ознакомительную практику

Студент гр. БИС-22-1 Лазакович Алексей Евгеньевич

1 Тема работы: Разработка приложения

2 Срок сдачи работы: 11.07.2024.

3 Содержание задания

3.1 Решение практических задач

Задача 1.

Изучить библиотеки (eSpeak, Festival, и т.п.) для преобразования текста в речь, выбрать наиболее подходящий для создания программы. Параметры уравнения кривой или поверхности должны вводиться в специально отведённые ячейки экранной формы программы.

Задача 2.

С использованием языка программирования C++ и среды разработки FLUID, создать кроссплатформенное приложение с графическим интерфейсом для преобразования текста в речь (TTS). Приложение должно предоставлять возможность выбора и настройки тембра голоса, высоты тона и управления акцентом (расстановка ударений). Обязательно обеспечить версию для Linux.

3.2 Оформление и защита отчета

Отчет по учебной исследовательской практике должен содержать: титульный лист; индивидуальное задание; содержание; цель и задачи; описание выполненных заданий (в соответствии с методическим руководством и требованиями преподавателя); выводы и предложения; список использованных источников; графический материал (схемы, графики, технологические карты).

Календарный план-график работ прилагается к отчету отдельным листом.

Требования к оформлению отчетов.

Отчёт предоставляется в печатном виде с выполнением требований нормоконтроля и состоит из следующих разделов:

Введение. Во введении обосновывается цель и задачи прохождения практики.

В разделе 1 выполняется краткий обзор литературы по теме индивидуального задания, подбирается необходимый математический инструментарий (аппарат), производится его анализ и разрабатывается алгоритм решения задачи.

В разделе 2 осуществляется обоснование выбора среды реализации разработанного алгоритма.

В разделе 3 описывается процесс кодирования в выбранной среде и языке программирования и основные элементы управления создаваемым приложением.

В разделе 4 описывается процесс тестированию и отладки программного кода.

Раздел 5 (рекомендуемый) должен содержать краткое иллюстрированное руководство пользователя.

Заключение. В заключении обобщается изложенный в отчёте материал, делаются выводы.

Объем отчёта составляет 20-25 страниц.

Отчёт по практике оформляется в соответствии с «Требованиями к оформлению текстовой части выпускных квалификационных работ, курсовых работ (проектов), рефератов, контрольных работ, отчётов по практикам, лабораторным работам (СК-СТО-ТР04-1.005-2020)». Для оформления графического материала блок-схем, алгоритмов использовать ГОСТ 19.701-90 который распространяется на условные обозначения (символы) в схемах алгоритмов, программ, данных и систем и устанавливает правила выполнения схем, применяемых для отображения различных видов задач обработки данных и средств их решения.

Руководитель
Канд. хим. наук, доцент
старший преподаватель каф. ИТС _____Б.К. Васильев

Задание получил: _____А.Е. Лазакович

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАДИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

КАЛЕНДАРНЫЙ ПЛАН-ГРАФИК (ДНЕВНИК)

прохождения учебной ознакомительной практики студента ВВГУ

Студент Лазакович Алексей Евгеньевич направляется для прохождения учебной ознакомительной практики «Учебная ознакомительной практика» в ВВГУ, кафедра ИТС, г. Владивосток.

С __10.06.2024__ по __13.07.2024__ г.

№ п/п	Содержание выполняемых работ по программе	Сроки выполнения		Заключение и оценка руководителя	Подпись руководителя
		Начало	Окончание		
1	**Задание 1.** Анализ поставленной задачи	10.06.2024	17.06.2024		
2	**Задание 2.** Сбор и анализ информации	18.06.2024	19.06.2024		
3	**Задание 3.** Разработка решения поставленных задач	20.06.2024	10.07.2024		
4	**Задание 4.** Оформить отчет и документы практики в печатном и электронном виде и представить на защиту	11.07.2024	13.07.2024		

Согласовано:

Студент-практикант_____ Лазакович А.Е.

подпись

Руководитель от кафедры _____ Васильев Б.К.

подпись

дата

Содержание

Введение 3
1. Приложение для перевода текста в звук 4
1.1 История 4
1.2 Понятие синтеза речи 5
1.3 Алгоритм создания приложения 6
2. Выбор среды разработки и языка программирования 8
2.1 Язык программирования 8
2.2 Выбор библиотеки синтеза речи 10
2.3 Среда разработки 14
3. Процесс написания программного кода 17
4. Тестирование программного кода 22
Заключение 24
Список использованных источников 25

Введение

В рамках данной учебной практики необходимо разработать программу для преобразования текста в звук с использованием языка программирования C++ и инструментов, доступных в операционной системе Linux. Основной целью задания является создание интерактивного графического интерфейса пользователя (GUI), который позволяет загружать текстовые файлы, настраивать параметры озвучивания и сохранять результат в различных аудиоформатах.

Задача превращения текста в речь имеет широкое применение в современных технологиях, таких как создание аудиокниг, голосовых ассистентов и систем, требующих озвучивания текстовой информации. В нашей практике особое внимание уделяется удобству использования интерфейса, гибкости настройки параметров синтеза речи и надежности работы системы.

Первая задача заключается в создании программы, способной считывать текстовые файлы и отображать их содержимое в текстовом дисплее. Для реализации данной функции используется библиотека FLTK (Fast Light Toolkit), которая обеспечивает создание легковесного и быстрого графического интерфейса. Важным аспектом является обеспечение интерактивности интерфейса, чтобы пользователь мог легко выбирать файлы, управлять параметрами синтеза речи и видеть изменения в реальном времени. Акцент делается на интуитивно понятном интерфейсе, поддерживающем выбор языка, настройку голоса и других параметров озвучивания.

Вторая задача состоит в интеграции системы синтеза речи espeak-ng и инструмента конвертации аудио ffmpeg. Espeak-ng выбирается благодаря своей открытой лицензии, широкой поддержке языков и высокой производительности, что позволяет работать без необходимости интернет-соединения. Использование ffmpeg позволяет обеспечивать сохранение результата синтеза речи в качественных аудиоформатах WAV и MP3. Таким образом, пользователи программы могут не только прослушивать озвученный текст, но и сохранять его для дальнейшего использования.

Проект включает в себя глубокое изучение процессов компиляции и сборки программных модулей в операционной системе Linux, что также способствует приобретению навыков работы с системными вызовами и управлением памятью. Выбор языка программирования C++ обуславливается его высокой производительностью и возможностями низкоуровневого управления ресурсами, что критически важно для задач, связанных с синтезом речи и обработкой аудиофайлов.

1. Приложение для перевода текста в звук

1.1 История

В конце XVIII века датский учёный Христиан Кратценштейн, действительный член Российской академии наук, создал модель речевого тракта человека, способную произносить пять долгих гласных звуков (а, э, и, о, у). Модель представляла собой систему акустических резонаторов различной формы, издававших гласные звуки при помощи вибрирующих язычков, возбуждаемых воздушным потоком. В 1778 году австрийский учёный Вольфганг фон Кампелен дополнил модель Кратценштейна моделями языка и губ и представил акустическо-механическую говорящую машину, способную воспроизводить определённые звуки и их комбинации. Шипящие и свистящие выдувались с помощью специального меха с ручным управлением. В 1837 году учёный Чарльз Уитстоун (Charles Wheatstone) представил улучшенный вариант машины, способный воспроизводить гласные и большинство согласных звуков. А в 1846 году Джезеф Фабер (Joseph Faber) продемонстрировал свой говорящий орга́н Euphonia, в котором была реализована попытка синтезирования не только речи, но и пения.

В конце XIX века знаменитый учёный Александр Белл создал собственную «говорящую» механическую модель, очень схожую по конструкции с машиной Уитстоуна. С наступлением XX века началась эра электрических машин, и учёные получили возможность использовать генераторы звуковых волн и на их базе строить алгоритмические модели.

В 1930-х годах работник Bell Labs Хомер Дадли (Homer Dudley), работая над проблемой поиска путей для снижения пропускной способности, необходимой в телефонии, чтобы увеличить её передающую способность, разрабатывает VOCODER (сокращенно от англ. voice – голос, англ. coder – кодировщик) – управляемый с помощью клавиатуры электронный анализатор и синтезатор речи. Идея Дадли заключалась в том, чтобы проанализировать голосовой сигнал, разобрать его на части и пересинтезировать в менее требовательный к пропускной способности линии. Усовершенствованный вариант вокодера Дадли, VODER, был представлен на Нью-Йоркской Всемирной выставке 1939 года.

Первые синтезаторы речи звучали довольно неестественно и часто едва можно было разобрать воспроизводившиеся ими фразы. Однако качество синтезированной речи постоянно улучшалось, и речь, генерируемую современными системами синтеза речи, порой не отличить от реальной человеческой речи. Но, несмотря на успехи электронных синтезаторов речи, исследования в области создания механических синтезаторов речи по-прежнему ведутся, например, для использования в роботах-гуманоидах.

Первые системы синтеза речи на базе вычислительной техники стали появляться в конце 1950-х годов, а первый синтезатор «текст в речь» был создан в 1968 году.

В 2005 году Курцвейл предсказал, что, поскольку соотношение цены и качества приведет к тому, что синтезаторы речи станут дешевле и доступнее, больше людей выиграют от использования программ преобразования текста в речь [1].

Следовательно, разработка синтезаторов речи прошла долгий путь от первых механических моделей до современных электронных систем. В XVIII веке Христиан Кратценштейн создал первую модель, способную произносить гласные звуки. Позже Вольфганг фон Кампелен улучшил её, добавив модели языка и губ. В 1837 году Чарльз Уитстоун представил новую версию машины, способную воспроизводить большинство звуков. В XX веке Хомер Дадли разработал вокодер, позволивший анализировать и синтезировать речевые сигналы. Качество речи, генерируемой синтезаторами, постоянно улучшалось, и сегодня синтезированную речь порой невозможно отличить от настоящей. Первые компьютерные синтезаторы текста в речь появились в конце 1950-х годов, а в 1968 году был создан первый настоящий синтезатор «текст в речь».

1.2 Понятие синтеза речи

Синтез речи – это технология, которая помогает озвучивать текст. Иногда вместо термина «синтез речи» используют аббревиатуру TTS, которая расшифровывается как Text-to-Speech. Есть множество методов синтеза речи, но наибольшее значение имеют две группы технологий: конкатенация и синтез по параметрам с использованием глубоких нейронных сетей. При конкатенации используются предварительно записанные звуки, слова и фразы. Это огромный массив записей, ведь звучание каждого фрагмента зависит от фонетического окружения. Синтезированная речь получается довольно качественной, хотя и с некоторыми недостатками: монотонностью и артефактами на стыке фрагментов. Синтез на основе глубокого обучения стал активно развиваться примерно с 2016 года. Сегодня созданная этим методом речь практически не отличается от натуральной.

Чтобы подготовить TTS-систему, нужен датасет для обучения нейросети. Сначала создаются аудиозаписи с разными голосами и темами, дальше они обрабатываются разметчиками и превращаются в датасет, на котором сеть и обучается. Печатный текст состоит из букв, слов, а аудиозапись можно разобрать на мелкие фрагменты, каждый из которых имеет свой «рисунок» на дорожке аудиозаписи – спектрограмме. Задача нейросети в том, чтобы сначала научиться устанавливать соответствие «рисунка» аудиозаписи тексту, а потом научиться воспроизводить текст по этим примерам самостоятельно.

Чтобы синтезированная речь не была роботоподобной, к обучению подключаются дополнительные модели: например, модель, которая обучается предсказанию, где по тексту

нужно сделать смысловую паузу, или модель, которая работает над интонацией (создаёт в нужных местах повышение-понижение тона и громкости).

Для более человекоподобной речи также важно, чтобы алгоритм синтеза включал в себя понимание контекста. В этом случае конечная интонация исходит не только из знаков препинания и пауз, а ещё и из смысла текста [2].

Таким образом, синтез речи (Text-to-Speech, TTS) – это технология, позволяющая озвучивать текст. Существуют две основных группы методов синтеза речи: конкатенация и синтез на основе глубоких нейронных сетей. При конкатенации используются предварительно записанные звуки, что приводит к качественной, но часто монотонной речи. С 2016 года активно развивается синтез на основе глубокого обучения, создающий почти натуральную речь. Для создания TTS-системы необходим датасет аудиозаписей, соответствующий тексту. Нейросеть учится сопоставлять аудиофрагменты и текст, а также воспроизводить их. Дополнительные модели предсказывают паузы и интонации, что делает синтезированную речь более человекоподобной и учитывающей контекст текста.

1.3 Алгоритм создания приложения

Задача 1: Определение целей и требований приложения

1. Определение целей: разработать приложение для создания аудиофайлов на основе текста.

2. Исследование и анализ рынка: существует множество различных TTS программ со своими плюсами и минусами, изучив которые мы смогли понять концепцию нашего будущего приложения. Самыми популярными из этих программ являются: Acapela, Google Text-to-Speech, Ivona, Loquendo, Nefonit, RHVoice, SamsungTTS, Vocalizer Expressive 2 от Nuance, Yandex SpeechKit, Читатель [3].

3. Определение функционала приложения: программа должна уметь загружать текст из файлов, предоставлять выбор языка (русский или английский) и голоса для озвучивания, настраивать параметры речи (громкость, пауза между словами, высота и скорость), а также позволять сохранять озвученный текст в форматах WAV и MP3

Задача 2: Проектирование архитектуры приложения

1. Создание архитектурной схемы: описать основные модули и их взаимодействие: интерфейс пользователя, работа с файлами, настройки озвучки, вызов внешних программ (например, espeak-ng и ffmpeg), отображение и редактирование текста.

2. Выбор библиотек и инструментов: определить, какие библиотеки будут использоваться: FLTK для GUI, espeak-ng для синтеза речи, ffmpeg для конвертации аудиофайлов [4].

3. Определение структуры проекта: подготовить иерархию файлов и каталогов для удобного расположения кода, библиотек, ресурсов.

Задача 3: Настройка среды разработки

1. Настройка окружения: установить и настроить необходимое ПО: компиляторы, библиотеки, системы контроля версий.

2. Создание проекта: инициализировать проект, создать базовую структуру каталогов и файлов.

Задача 4: Реализация базового функционала

1. Создание базового окна приложения: с использованием FLTK создать главное окно с кнопками, текстовыми полями и другими виджетами.

2. Функции загрузки и отображения текста: реализовать функции для загрузки текстовых файлов в буфер и отображение текста на экране.

3. Реализация настройки параметров синтеза речи: добавить слайдеры и кнопки для изменения громкости, высоты, скорости, выбора голоса и языка.

4. Функции синтеза речи и сохранения аудио: реализовать вызов библиотеки для озвучивания текста и сохранения аудиофайлов.

Задача 5: Тестирование и отладка

1. Отладка и исправление ошибок: запускать приложение, тестировать различные функции и исправлять найденные баги.

В результате выполнения данных шагов будет создано приложение, позволяющее преобразовывать текст в звук с возможностью настройки параметров и сохранения результата в различных аудиоформатах. Пользовательский интерфейс обеспечит удобство использования приложения и его гибкость.

2. Выбор среды разработки и языка программирования

2.1 Язык программирования

Выбор языка программирования практически сразу пал в пользу C++, ведь он является компилируемым языком, и его программы обычно выполняются быстрее, чем программы, написанные на интерпретируемых языках, таких как Python или JavaScript.

В критичных к производительности задачах, таких как обработка аудио и синтез речи, скорость выполнения кода играет важную роль.

C++ предоставляет тонкий контроль над памятью и позволяет разработчикам управлять системными ресурсами напрямую, что важно для управления аудиофайлами, буферами данных и другими аспектами низкоуровневого программирования.

Возможность использования указателей и ручного управления памятью позволяет оптимизировать использование ресурсов и улучшить производительность.

C++ поддерживает множество библиотек и фреймворков, необходимых для нашего проекта, таких как FLTK для создания графического интерфейса пользователя (GUI), “espeak-ng”, “festival”, “RHVoice” для синтеза речи и `ffmpeg` для работы с аудиофайлами.

Наличие статических и динамических библиотек позволяет интегрировать и использовать существующие решения для синтеза и обработки аудио. Язык программирования C++ легко взаимодействует с операционной системой Linux и поддерживает использование системных вызовов и API, что облегчает реализацию функций работы с файлами, управления процессами и межпроцессного взаимодействия.

Прямая работа с системными интерфейсами позволяет более эффективно использовать возможности операционной системы.C++ обладает богатым наследием и активным сообществом разработчиков, что облегчает поиск решений для возникающих проблем и доступ к большому количеству документации, учебных материалов и примеров кода [5].

Рассмотрим преимущества C++ перед другими языками:

1. C++ и. Python

Производительность: C++ значительно быстрее работает за счет компиляции в машинный код, тогда как Python интерпретируется, что замедляет выполнение программ.

Контроль над ресурсами: В C++ разработчик контролирует управление памятью, что позволяет создавать более оптимизированные и эффективные программы. Python использует сборщик мусора, что может приводить к непредсказуемому использованию памяти [6].

Поддержка библиотек: хотя Python имеет богатую экосистему библиотек, многие из них написаны на C++ для повышения производительности, что делает C++ более непосредственным выбором для критичных к скорости приложений.

2. C++ и Java

Низкоуровневое программирование: C++ предоставляет возможности работы на низком уровне, такие как манипуляции с указателями и работа с аппаратурой, что недоступно в Java.

Производительность: несмотря на JIT-компиляцию в Java, C++ все равно обеспечивает больший контроль и зачастую лучшую производительность благодаря прямой компиляции в машинный код.

Системное программирование: C++ лучше подходит для системного программирования и интеграции с операционной системой, что важно для нашего проекта, связанного с аудио и системными ресурсами.

3. C++ и. JavaScript

Применение: JavaScript в основном предназначен для веб-разработки и имеет ограничения при работе с низкоуровневыми системными ресурсами, тогда как C++ предназначен для таких задач.

Производительность: скриптовые языки, такие как JavaScript, обычно медленнее по сравнению с компилируемыми языками, такими как C++.

Функциональность и библиотеки: C++ предлагает обширный набор библиотек для системного программирования и работы с аудио, что отсутствует в JavaScript.

Ниже, в таблице 1, предоставлено наглядное сравнение некоторых языков программирования.

Таблица 1 – Сравнительный анализ языков программирования

Критерий	C++	Python	Java	JavaScript
Производительность	Высокая, компилируемый язык, близок к машинному коду	Средняя, интерпретируемый язык, работает через VM	Высокая, JIT-компиляция, но выше накладные расходы	Низкая, интерпретируемый язык, исполняется в браузере
Контроль над ресурсами	Полный контроль над памятью (ручное управление)	Ограниченный, автоматическое управление памятью	Автоматическое управление памятью	Ограниченный, автоматическое управление памятью

Продолжение таблицы 1

Критерий	C++	Python	Java	JavaScript
Низкоуровневое программирование	Да, доступ к указателям, прямое управление памятью	Нет, высокоуровневый язык	Нет, высокоуровневый язык	Нет, высокоуровневый язык
Совместимость с ОС	Высокая, поддержка системных вызовов и API	Ограниченная, требует дополнительных библиотек	Ограниченная, работает через JVM	Ограниченная, работает в браузере или через Node.js
Поддержка библиотек	Обширная, множество библиотек для системного программирования	Обширная, особенно для научных вычислений и веб-разработки	Обширная, особенно для веб-разработки и бизнес-приложений	Обширная, особенно для веб-разработки
Простота использования	Средняя, требует знаний и опыта	Высокая, простота синтаксиса	Средняя, требует знания JVM и объектов	Средняя, специфична для веб-разработки
Мульти платформенность	Высокая, код можно компилировать под различные ОС	Высокая, требует установки интерпретатора	Высокая, код может выполняться на любой JV/M	Высокая, через браузеры и Node.js

Как видно из таблицы, C++ является оптимальным выбором для нашего проекта благодаря своей высокой производительности, тонкому контролю над системными ресурсами и возможности низкоуровневого программирования. Эти преимущества очень важны для задач синтеза речи и управления аудио, где скорость и эффективность использования ресурсов играют ключевую роль. Это делает C++ лучшим выбором для реализации нашего проекта.

2.2 Выбор библиотеки синтеза речи

Мы провели анализ трех библиотек синтеза речи: espeak-ng, Festival и Google Text-to-Speech (TTS), и рассмотрели их достоинства и недостатки в контексте создания нашего приложения для Linux.

1. Espeak-ng:

Описание: espeak-ng (eSpeak New Generation) является продолжением проекта eSpeak, предлагая улучшенный синтез речи, поддержку большего количества языков и лучшие диалекты.

Лицензия: Open Source (GPLv3)

Поддержка языков: более 100 языков и диалектов

Платформы: Linux, Windows, MacOS

2. Festival:

Описание: Festival – это универсальная система синтеза речи, разработанная в Центре речевых технологий Университета Эдинбурга. Она поддерживает создание пользовательских голосов и текста на основе скриптов.

Лицензия: Open Source (MIT/BSD)

Поддержка языков: ограниченная поддержка (основные языки)

Платформы: Linux, Windows, MacOS

3. Google TTS:

Описание: Google Text-to-Speech (TTS) – это облачный сервис синтеза речи, предоставляемый Google, который предлагает высококачественные голоса на основе искусственного интеллекта.

Лицензия: коммерческая, но есть бесплатный уровень использования

Поддержка языков: более 30 языков и множество диалектов

Платформы: облачный сервис, доступ через API

Мы проанализировали достоинства и недостатки данных библиотек. К достоинствам Espeak-ng можно отнести: лицензия Open Source, что означает бесплатное использование и возможность модификации, обширная поддержка более 100 языков и диалектов, высокая производительность, так как работает локально без необходимости интернет-соединения, возможность настройки множества параметров для изменения скорости, высоты и других аспектов речи, а также легкая интеграция в существующие проекты с использованием командной строки или API. Однако, espeak-ng имеет свои недостатки, такие как менее натуральное звучание голосов по сравнению с коммерческими решениями, и необходимость более сложной настройки через командную строку или конфигурационные файлы.

Рассмотрим достоинства Festival: лицензия Open Source, которая позволяет бесплатное использование и модификацию, хорошее качество синтеза речи и возможность создания пользовательских голосов, высокая производительность за счет локальной работы, а также обширные возможности для настройки и создания пользовательских голосов через скрипты. Недостатки Festival включают ограниченную поддержку языков по сравнению с espeak-ng и Google TTS, более сложную интеграцию, требующую знания системы и создания скриптов, а также

тот факт, что, несмотря на хорошее качество синтеза, он требует значительных настроек для достижения оптимальных результатов.

Достоинства Google TTS заключаются в высоком качестве и естественности звучания благодаря технологиям глубокого обучения, удобном REST API для интеграции с приложениями и широком выборе языков и диалектов. Однако, Google TTS имеет свои недостатки, такие как коммерческая лицензия, требующая оплаты при высоком объеме использования, зависимость от интернет-соединения, что может быть недостатком для оффлайн приложений, и производительность, которая может зависеть от качества интернет-соединения и скоростей передачи данных [7].

Мы составили таблицу со сравнением характеристик данных библиотек (Таблица 2).

Таблица 2 –Сравнительный анализ данных библиотек

Критерий	Espeak-ng	Festival	Google TTS
Лицензия	GPLv3 (Open Source)	MIT/BSD (Open Source)	Коммерческая (бесплатный уровень)
Поддержка языков	Более 100 языков и диалектов	Ограниченная поддержка	Более 30 языков и множество диалектов
Качество синтеза	Среднее, но разборчивое	Хорошее, но требует настройки	Высокое, натуральное звучание
Интерфейс	Командная строка, API библиотеки	Командная строка, API библиотеки	REST API
Производительность	Высокая, локальное выполнение	Высокая, локальное выполнение	Зависит от интернет-соединения и пропускной способности

Продолжение таблицы 2

Критерий	Espeak-ng	Festival	Google TTS
Гибкость настройки	Высокая, множество параметров	Высокая, возможность создания пользовательских голосов	Ограниченная, зависит от API
Интеграция	Легкая, открытый код	Средняя, открытый код	Средняя, через REST API
Стоимость	Бесплатно	Бесплатно	Зависит от использования
Зависимость от интернета	Нет	Нет	Да

Анализ таблицы показывает, что espeak-ng является наилучшим выбором для нашего проекта благодаря своей высокой производительности, широкому спектру поддерживаемых языков и открытой лицензии GPLv3.

Таким образом мы выбрали espeak-ng для нашего проекта по следующим причинам:

1. Лицензия и стоимость: Open Source (GPLv3) лицензия обеспечивает бесплатное использование и возможность модификации.

2. Поддержка языков: Espeak-ng поддерживает более 100 языков и диалектов, что критически важно для многоязыковых приложений.

3. Производительность: Высокая производительность локального исполнения без необходимости подключения к интернету.

4. Гибкость настройки: Возможность детальной настройки параметров синтеза речи для достижения нужного качества.

5. Интеграция: Легкость интеграции в существующие проекты с использованием командной строки или API.

Хотя Google TTS обеспечивает лучшее качество синтеза речи, зависимость от интернета и коммерческая лицензия делают его менее привлекательным для оффлайн-приложений и проектов с ограниченным бюджетом. Festival, в свою очередь, предлагает хорошее качество синтеза, но ограниченная поддержка языков и сложная интеграция делают его менее оптимальным выбором по сравнению с espeak-ng.

2.3 Среда разработки

Мы провели анализ и сравнение инструментов разработки GUI, таких как FLUID, GTK и Qt.

1. FLUID (Fast Light User Interface Designer):

Описание: FLUID – это инструмент для разработки GUI, связанный с библиотекой FLTK (Fast Light Toolkit). Он предлагает простое и легкое в использовании средство для проектирования графических интерфейсов [8].

Лицензия: Open Source (GNU Lesser General Public License)

Поддержка языков: C++

Платформы: Linux, Windows, MacOS

2. GTK:

Описание: GTK – это библиотека для создания графических интерфейсов, изначально разработанная для среды GNOME. Она широко используется в различных настольных приложениях [9].

Лицензия: Open Source (LGPL)

Поддержка языков: C (основной), также доступно множество языковых привязок (Python, JavaScript и др.)

Платформы: Linux, Windows, MacOS, BSD

3. Qt:

Описание: Qt – это мощный и многоплатформенный фреймворк для разработки GUI приложений, который поддерживает много языков и имеет собственный визуальный редактор Qt Designer [10].

Лицензия: Open Source (GPL/LGPL) и Коммерческая лицензия

Поддержка языков: C++ (основной), также доступно множество языковых привязок (Python через PyQt, QML и др.)

Платформы: Linux, Windows, MacOS, Android, iOS, другие

Перейдём к достоинствам и недостаткам средств разработки. Достоинства FLUID: легкость в освоении и использовании, что идеально подходит для простых и средних по сложности проектов, высокая производительность благодаря легковесной библиотеке FLTK, Open Source лицензия (LGPL), что позволяет свободное использование и модификацию, и плотная интеграция с FLTK, что упрощает разработку и деплоймент приложений. Однако FLUID имеет и свои недостатки: некоторая ограниченность в настройках и кастомизации графических интерфейсов, скромный набор стандартных виджетов и тем по сравнению с QT и GTK, и менее обширное сообщество по сравнению с GTK и Qt.

Достоинства GTK включают широкую поддержку множества языков программирования через биндинги, хорошую интеграцию с Linux и окружением GNOME, что делает его популярным выбором для приложений на этой платформе, и высокую гибкость и возможности настройки интерфейсов. Недостатки GTK: более сложен в освоении по сравнению с FLUID, особенно для новичков, может быть медленнее по сравнению с FLTK и Qt в определенных задачах, и несмотря на обширную документацию, иногда сложно найти примеры для менее распространенных задач.

Достоинства Qt заключаются в обширном наборе инструментов и виджетов, высокой степени кастомизации, поддержке всех основных настольных и мобильных платформ, высоком качестве и оптимизации для производительности, а также мощном инструменте для визуального проектирования интерфейсов Qt Designer. Однако Qt имеет свои недостатки: коммерческая лицензия для определенных случаев использования может быть дорогостоящей, более сложен в освоении по сравнению с FLUID и требует глубоких знаний C++, и экосистема Qt требует знания и использования множества инструментов, что может быть пугающим для новичков. Для наглядного сравнения и анализа характеристик данных сред разработки рассмотрим таблицу 3.

Таблица 3 – Сравнение характеристик сред разработки

Критерий	FLUID	GTK	Qt
Лицензия	LGPL	LGPL	GPL/LGPL и Коммерческая
Поддержка языков	C++	C (основной), привязки для множества языков	C++ (основной), привязки для множества языков
Платформы	Linux, Windows, MacOS	Linux, Windows, MacOS, BSD	Linux, Windows, MacOS, Android, iOS, другие

Продолжение таблицы 3

Критерий	FLUID	GTK	Qt
Производительность	Высокая	Средняя, но удовлетворительная для большинства задач	Высокая, оптимизированная для производительности
Простота использования	Простая кривая обучения, простой интерфейс	Средняя, требуется знание C и собственных библиотек	Средняя, требует знаний C++ и инструментов Qt
Развитие и поддержка	Широкая поддержка в сообществе FLTK	Активное сообщество, частые обновления	Активное сообщество, коммерческая поддержка
Множество платформ	Да, кроссплатформенная поддержка	Да, ограниченная поддержка мобильных платформ	Да, обширная поддержка множества платформ

Исходя из представленной таблицы, FLUID является оптимальным выбором для нашего проекта благодаря своей простой кривой обучения, высокому уровню производительности и плотной интеграции с библиотекой FLTK.

Таким образом, мы выбрали FLUID для нашего проекта по следующим причинам:

1. Простота использования: Легкость освоения и использования делают FLUID идеальным для быстрого прототипирования и разработки простых и средних по сложности проектов.

2. Производительность: Высокая производительность FLTK позволяет обеспечивать плавную работу интерфейса даже на менее мощных системах.

3. Лицензия и стоимость: Open Source лицензия (LGPL) позволяет свободное использование и модификацию без необходимости покупки коммерческой лицензии.

4. Интеграция с проектом: Плотная интеграция с FLTK упрощает разработку и деплоймент приложения.

Хотя GTK и Qt предлагают более гибкие и мощные решения, их сложность и дополнительные требования к ресурсам не всегда оправданы для задач нашего проекта. В свою очередь, FLUID обеспечивает удобство и производительность, необходимые для нашего проекта.

3. Процесс написания программного кода

Для начала необходимо было создать код для вывода пустого окна с белым фоном. Для реализации этого была использована библиотека FLTK (Fast Light Toolkit), которая предназначена для создания графического интерфейса.

Первый шаг – это включение необходимых заголовочных файлов в начале программы. Для использования функционала FLTK нужно подключить соответствующие заголовочные файлы. Например, мы подключаем основной заголовочный файл FLTK, который включает в себя все основные определения и функции библиотеки. Также подключаем заголовочный файл, который используется для создания окна с двойной буферизацией. Двойная буферизация позволяет избежать мерцания при перерисовке интерфейса, делая отображение более плавным и приятным для пользователя.

После подключения заголовочных файлов можно перешли к созданию основного окна программы. Для этого создается объект, представляющий собой окно приложения. В конструкторе этого объекта указываются размеры окна и его заголовок. Например, для создания окна размером 840 на 660 пикселей с заголовком «Text to Speech Player» используется соответствующий конструктор. Установка цвета фона окна производится с помощью метода, в который передается соответствующий цветовой код. Например, для установки белого фона используется метод, который принимает код белого цвета.

Далее следует основной цикл приложения, который запускается с помощью специальной функции. Эта функция обрабатывает все события, происходящие в окне, такие как нажатия клавиш, перемещение мыши и другие взаимодействия пользователя с интерфейсом. Функция запускает цикл обработки событий, который продолжается до тех пор, пока окно не будет закрыто пользователем.

Таким образом, окно будет оставаться открытым и реагировать на действия пользователя до тех пор, пока он не решит его закрыть. В итоге, реализация базового окна с использованием FLTK включает в себя несколько простых шагов: подключение необходимых заголовочных файлов, создание и настройку объекта окна, а также запуск основного цикла обработки событий. Этот базовый шаблон можно затем расширять, добавляя различные элементы интерфейса и функциональность по мере необходимости.

Для открытия текстовых файлов в приложении, созданном с использованием библиотеки FLTK, была добавлена кнопка openButton. Эта кнопка позволяет пользователю выбрать файл для открытия. При нажатии на кнопку openButton вызывается функция openFileCallback. Эта функция отвечает за открытие диалогового окна выбора файла, где пользователь может выбрать нужный текстовый файл. Выбранный файл загружается в текстовый буфер textBuffer, который представляет собой структуру данных для хранения текстовой информации в памяти.

Создание текстового буфера и отображение текста в приложении, использующем библиотеку FLTK (Fast Light Toolkit), выполняется с помощью классов Fl_Text_Buffer и Fl_Text_Display. Эти классы предоставляют удобные средства для работы с текстом в графическом интерфейсе.

Fl_Text_Buffer представляет собой объект, который содержит текстовые данные. Он позволяет управлять текстом, включая его вставку, удаление, изменение и получение содержимого. Кроме того, Fl_Text_Buffer поддерживает различные операции с выделением текста и механизмы для обработки изменений текста, таких как отслеживание событий.

Fl_Text_Display используется для отображения содержимого текстового буфера на экране. Этот класс управляет отображением текста в окне приложения, обеспечивая возможность прокрутки текста, установку цветового оформления, выделение и форматирование текста.

Для интеграции Fl_Text_Buffer и Fl_Text_Display необходимо сначала создать объекты этих классов. Затем текстовый буфер связывается с отображением текста. Это позволяет автоматически обновлять содержимое Fl_Text_Display при изменениях в Fl_Text_Buffer.

Основные операции, которые можно выполнять с помощью Fl_Text_Buffer и Fl_Text_Display, включают вставку нового текста, удаление существующего текста, получение текста из буфера для обработки или отображения, а также управление форматированием и стилем отображения текста.

В приложении были добавлены радиокнопки и список выбора для определения языка и голоса для синтеза речи. Радиокнопки englishButton и russianButton позволяют пользователю выбрать между английским и русским языками соответственно. Они реализованы как взаимоисключающие кнопки, что позволяет выбрать только один из языков. Например, если пользователь выбирает русский язык (russianButton->value() возвращает true), то соответствующие действия и настройки для синтеза речи будут применены для русского языка.

Список выбора voiceChoice предоставляет пользователю возможность выбрать конкретный голос для синтеза речи. В зависимости от выбранного языка (русский или английский), список может динамически меняться, чтобы отображать различные варианты голосов. Например, для русского языка voiceChoice содержит голоса с разными характеристиками (мужские и женские голоса, различные акценты и т.д.), а для английского языка список может предложить соответствующие варианты англоязычных голосов.

Выбранный голос из списка voiceChoice используется в функциях speakText и speakTextGoogle для синтеза речи. Например, при сохранении текста в WAV или MP3 файлы или при воспроизведении примера звука (playExampleCallback), выбранный голос передается

в качестве параметра функции speakText, чтобы определить, какой именно голос использовать для синтеза речи в конкретном случае.

Такой подход обеспечивает гибкость и настраиваемость приложения для воспроизведения и сохранения текста с использованием различных языков и голосовых характеристик, что повышает его функциональность и пользовательский комфорт.

Для управления параметрами озвучивания текста в приложении были использованы слайдеры. Каждый слайдер представляет собой элемент пользовательского интерфейса, который позволяет мгновенно настраивать определенный аспект звучания текста.

Например, слайдер amplitudeSlider отвечает за громкость звука. Он позволяет пользователю выбирать значение от 0 до 200, где 0 соответствует минимальной громкости, а 200 – максимальной.

Слайдер wordGapSlider настраивает паузу между словами в тексте. Пользователь может регулировать этот параметр от 0 до 50, где 0 означает минимальное время паузы, а 50 – максимальное.

Слайдер pitchSlider регулирует высоту голоса, озвучивающего текст. Значения на слайдере варьируются от 0 до 99, где 0 представляет наивысшую высоту голоса, а 99 – наименьшую.

Слайдер speedSlider управляет скоростью речи. Пользователь может выбирать значение от 80 до 500, где 80 соответствует самой медленной скорости речи, а 500 - самой быстрой.

Каждый слайдер интуитивно понятен и предоставляет пользователю гибкость в настройке параметров озвучивания текста. Значения, выбранные пользователем на этих слайдерах, передаются в функцию speakText, которая использует их для настройки озвучивания текста в соответствии с предпочтениями пользователя. Это делает приложение более функциональным и удобным в использовании, позволяя адаптировать процесс озвучивания под различные потребности и предпочтения пользователей прямо через графический интерфейс.

Функция speakText в приложении выполняет задачу озвучивания текста с использованием программы espeak-ng. Первым шагом функция создает временный текстовый файл tmp_text.tmp, куда записывает переданный ей текст для последующего использования espeak-ng. Это необходимо, так как espeak-ng требует наличия текстового файла для произнесения текста.

Далее функция проверяет переданный текст на наличие содержимого: если текст пуст или равен nullptr, функция выводит сообщение об ошибке и завершает свою работу, не вызывая espeak-ng.

Для настройки озвучивания текста в соответствии с параметрами, установленными через слайдеры в графическом интерфейсе, функция формирует команду для вызова espeak-ng.

Эта команда включает путь к временному файлу tmp_text.tmp, выбранный голос для озвучивания (например, русский мужской или английский), а также значения параметров звука.

После формирования команды espeak-ng, функция запускает её выполнение в фоновом режиме с использованием std::thread. Это позволяет приложению продолжать работу без блокировки пользовательского интерфейса во время озвучивания текста. После завершения озвучивания текста espeak-ng, временный файл tmp_text.tmp удаляется для освобождения ресурсов компьютера.

Функция playExampleCallback является обработчиком события для кнопки exampleButton в графическом интерфейсе приложения, созданного с использованием библиотеки FLTK (Fast Light Toolkit). При нажатии на кнопку exampleButton пользователь инициирует процесс воспроизведения аудио, соответствующего заданному примеру текста.

В начале функция определяет текст для воспроизведения, выбирая его на основе текущего выбранного языка (русский или английский) и голоса. Затем она вызывает функцию speakText, которая использует внешнюю программу espeak-ng для синтеза речи на основе предоставленного текста. Параметры синтеза речи, такие как громкость, пауза между словами, высота и скорость речи, настраиваются значениями, полученными от соответствующих слайдеров (amplitudeSlider, wordGapSlider, pitchSlider, speedSlider).

Для предотвращения блокировки основного потока пользовательского интерфейса, запуск espeak-ng осуществляется в отдельном потоке с помощью std::thread. После завершения воспроизведения речи удаляется временный файл, который использовался для хранения текста для воспроизведения.

Важно отметить, что процесс воспроизведения аудио может включать несколько этапов, таких как подготовка аудиофайла, его загрузка в память, и непосредственно воспроизведение звука. Функция playExampleCallback, вызываемая при нажатии кнопки, обычно инкапсулирует все эти этапы, обеспечивая плавное и корректное воспроизведение аудио.

Для преобразования текста в речь с использованием сервиса Google Text-to-Speech и сохранения в формате WAV или MP3 были добавлены кнопки saveWavButtonGoogle и saveMp3ButtonGoogle. Эти функции позволяют пользователю сгенерировать аудиофайлы на основе введенного текста. Кнопка saveWavButtonGoogle загружает аудиофайл в формате WAV с сервера Google, а кнопка saveMp3ButtonGoogle загружает файл в формате MP3.

Для реализации загрузки аудиофайлов используется утилита wget, которая выполняет HTTP-запросы к серверу Google для получения аудиоданных. Полученные данные затем обрабатываются и объединяются с помощью инструмента ffmpeg, который используется для конвертации и обработки мультимедийных файлов. Это позволяет пользователю сохранить результат в выбранном формате и обеспечивает гибкость при работе с аудиовыходом.

Таким образом, программа создает окно с различными элементами управления, позволяющими пользователю открывать текстовые файлы, выбирать параметры озвучивания и сохранять результат в различных аудиоформатах. Основной функционал включает использование библиотеки FLTK для создания интерфейса, espeak-ng для преобразования текста в речь и ffmpeg для конвертации аудиофайлов. Создание окна происходит через объект класса Fl_Double_Window с указанием его размеров и заголовка. Для взаимодействия с файловой системой и аудиообработкой используются различные функции и библиотеки, интегрированные в программу.

4. Тестирование программного кода

В ходе разработки программы для конвертации текста в аудиоформаты возникли ряд сложностей, требующих внимания и усилий для их решения. Основные вызовы касались как технических аспектов, так и обеспечения удобства использования программы. Изначально программа столкнулась с трудностями установки необходимых библиотек и инструментов. Некоторые из них, необходимые для работы с аудиофайлами и текстом, требовали дополнительных настроек и проверок совместимости, что значительно усложнило начальную настройку и развертывание программы. Это включало не только загрузку и установку библиотек, но и их интеграцию с основным кодом приложения, что было необходимо для корректного функционирования всех функциональных возможностей.

Одним из ключевых аспектов разработки была реализация функционала конвертации текста в аудиоформаты с использованием внешней библиотеки ffmpeg. Этот процесс включал создание кода для взаимодействия с ffmpeg, а также управление процессами конвертации и обработки аудиоданных. Особое внимание уделялось точной настройке параметров конвертации, необходимой для обеспечения высокого качества и эффективности процесса. Это включало выбор оптимальных настроек для кодеков, битрейтов и других параметров, чтобы гарантировать соответствие требованиям качества и размера файлов аудиоформатов.

Для работы с текстовыми данными программа предусматривала функционал загрузки текста из файлов различных форматов. Важной задачей стало обеспечение разделения текста на блоки, учитывая ограничения на длину текста, которые могли быть наложены используемым сервисом синтеза речи. Разработка алгоритма автоматического разбиения и объединения текстовых блоков в один аудиофайл требовала тщательного проектирования и тестирования.

Для улучшения пользовательского опыта был разработан графический интерфейс на основе библиотеки FLTK. Интерфейс включал элементы управления для загрузки файлов с текстом, выбора языка и голоса для синтеза речи, а также настройки параметров аудиовоспроизведения. Важным аспектом было не только создание интерфейсных элементов, но и их интеграция с основным функционалом программы для обеспечения удобства использования и интуитивно понятного взаимодействия с пользователем.

Тестирование программы играло ключевую роль в процессе разработки. Оно включало проверку стабильности и надежности функций конвертации и синтеза речи, а также обеспечение корректной работы интерфейса и всех основных функций программы. В процессе тестирования выявлялись потенциальные проблемы с производительностью, ошибки в логике конвертации и взаимодействия с внешними сервисами.

Одним из значимых этапов оптимизации программы стал анализ и улучшение алгоритмов обработки текста и аудиоданных. Это включало оптимизацию времени выполнения

операций и улучшение алгоритмов интерполяции и сегментации текста для синтеза речи. Результатом стало значительное повышение производительности программы и улучшение качества конвертации.

Таким образом, разработка программы для конвертации текста в аудиоформаты требовала комплексного подхода к решению технических и пользовательских задач. От оптимизации алгоритмов до тщательного тестирования, каждый этап играл важную роль в создании стабильного и эффективного продукта для конечных пользователей.

Заключение

В ходе выполнения практической работы было создано кроссплатформенное приложение с графическим интерфейсом для преобразования текста в речь (TTS) на основе языка программирования C++ и среды разработки FLUID. Программа предоставляет пользователю возможность выбора и настройки различных параметров, таких как громкость, пауза между словами, высота тона и скорость произношения. Также реализован выбор языка и голоса, что позволяет настроить произношение под различные языковые и стилистические требования. Изучены и проанализированы различные движки для преобразования текста в речь, в результате чего был выбран и интегрирован наиболее подходящий – eSpeak-ng, благодаря его гибким настройкам и кроссплатформенной поддержке.

Основные этапы работы включали разработку интерфейса приложения на базе библиотеки FLTK (Fast Light Toolkit), интеграцию элементов управления для настройки параметров и функциональности загрузки текстовых файлов, преобразования текста в речь с возможностью сохранения результатов в форматах WAV и MP3. Для обеспечения оптимальной производительности и многозадачности был использован механизм многопоточности при выполнении задач конвертации текста в аудиофайлы и их сохранении.

Также было реализовано взаимодействие с Google Text-to-Speech API для расширения выбора языков и акцентов, что позволяет пользователям создавать аудиофайлы в MP3 формате на основе текста, заданного пользователем, при наличии интернет-соединения. Программа разработана с учетом кроссплатформенности, обеспечивая возможность использования как на операционных системах Linux, так и на других платформах.

В итоге разработанное приложение представляет собой полнофункциональное решение для преобразования текста в речь с широким набором настроек и возможностей выбора языка и голоса. Проект успешно демонстрирует эффективное использование языка программирования C++ и графической библиотеки FLTK для создания интуитивно понятного пользовательского интерфейса и реализации сложных функциональных возможностей, что делает его значимым вкладом в область разработки программного обеспечения для TTS.

Список использованных источников

1 Приложение для перевода текста в звук: [сайт]. – URL: http://surl.li/ytphwp (дата обращения: 10.07.2024)

2 Понятие синтеза речи: [сайт]. – URL: https://developers.sber.ru/help/salutespeech/creating-audio-from-text (дата обращения: 10.07.2024)

3 TTS программы. [сайт]. – URL: https://4pda.to/forum/index.php?showtopic=200728 (дата обращения: 10.07.2024)

4 Выбор библиотек: [сайт]. – URL: http://linux.tiflocomp.ru/index.php?mod=tags&sec=329&p=6 (дата обращения: 10.07.2024)

5 Преимущества C++: [сайт]. – URL: https://practicum.yandex.ru/blog/yazyk-c-plus-dlya-chego-nuzhen/ (дата обращения: 10.07.2024)

6 Преимущества C++: [сайт]. – URL: http://surl.li/lbatiq (дата обращения: 10.07.2024)

7 Достоинства Google TTS: [сайт]. – URL: http://surl.li/ztyhey (дата обращения: 10.07.2024)

8 FLUID: [сайт]. – URL: http://surl.li/bvsgll (дата обращения: 10.07.2024)

9 GTK: [сайт]. – URL: https://ru.wikipedia.org/wiki/GTK (дата обращения: 10.07.2024)

10 Qt: [сайт]. – URL: https://ru.wikipedia.org/wiki/Qt (дата обращения: 10.07.2024)