МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка приложения генератор фонтов для матричного экрана»

Обучающийся С.А. Морозов _____
(Инициалы Фамилия) (личная подпись)

Руководитель С.В. Митин
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант кандидат педагогических наук, С.А. Гудкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия

Тольятти 2024

Аннотация

Тема выпускной квалификационной работы – «Разработка приложения генератор фонтов для матричного экрана».

Актуальность темы заключается в том, что генератор фонта с разными размерами существует, но позволяет формировать только латиницу.

Объектом исследования является генерация фонтов для матричных экранов.

Целью работы является разработка приложения для генерирования матричных фонтов, которые будут использоваться для вывода текста на матричных экранах.

В ходе выполнения работы были проанализированы различные уже существующие решения для генерации матричных фонтов, разработаны схемы «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ» для лучшего понимания проблем существующих приложений и достоинствах разрабатываемого. Далее была произведена проектировка приложения, во время которой был определён язык программирования для разработки и его библиотеки, а также разработаны диаграммы вариантов использования и последовательности. В последней главе была проведена разработка приложения и проведено тестирование.

Результаты выпускной квалификационной работы могут быть востребованы для различных производств и предприятий, на которых присутствует оборудование с матричными экранами.

Разработанное приложение повышает удобство использования и избавляет от ручного труда пользователя.

Работа состоит из 41 страниц, включая приложение. Содержит 25 рисунков, 2 таблицы, 1 приложение и список из 24 используемого источника.

Abstract

The title of the graduation work is «Development of the font generator application for the matrix screen» The senior paper consists of an introduction, three chapters, a conclusion, tables, images, list of references including foreign sources.

The key issue of the thesis is the generation of matrix fonts, we touch upon the problem that the user has to manually draw characters, and then create a font file from the drawn characters using software.

The aim of the work is to develop an application that will recognize characters from an image that has all the necessary font characters, after which it will create a matrix font file based on the characters found. Thanks to this, the user does not have to manually draw each character of the font.

The graduation work may be divided into several logically connected parts which are: analysis and comparison of existing applications for matrix font generation, highlighting their advantages and disadvantages, on the basis of which requirements for software development are developed; application development, namely the choice of programming language and necessary libraries, as well as the design of the application structure.

Finally, we develop the application in accordance with all the development requirements, after which we conduct the necessary testing.

In conclusion we’d like to stress this work is relevant for any company that uses matrix screens, as well as due to the fact that the program can recognize characters not only in Russian, it can be modified to create fonts in other languages.

Оглавление

Введение

Несмотря на то, что сегодня матричные экраны не так широко используются, как более современные технологии для вывода картинки пользователю, они всё равно остаются очень востребованы во многих сферах деятельности, и имеют в них преимущества, благодаря специфическим характеристикам.

Матричные экраны более долговечны и надёжны благодаря чему их можно легко применять в условиях повышенной вибрации, ударов или повышенных температур. Поэтому их часто используют в разных ситуациях, например в промышленности, где информация различных станков и систем управления отображается на небольших приборных панелях, которые показывают состояние процесса в реальном времени. Так же очень важны системы навигации, в которых необходима стабильная работа.

Целью дипломной работы будет разработка приложения для генерирования матричных фонтов (фонт — это файл, содержащий набор описаний текстовых символов и используемый программой (или графической оболочкой операционной системы) при отображении текста. [11]) для того, чтобы обеспечить корректное отображение текста. Для этого нужно проанализировать уже существующие решения, сравнить их, а после представить собственное решение этой задачи.

Структурно работа будет выглядеть так: в первой главе проанализируем и сравним существующие аналоги решений для генерирования матричного фонта, выделим преимущества и недостатки, на основе чего будут поставлены задачи для дальнейшей работы, во второй главе будет проектировка приложения, в третьей же главе описывается разработка и проводятся необходимые тестирования.

Глава 1 Анализ предметной области и постановка задачи

1.1 Выбор CASE-средств для описания процессов

Для моделирования необходимых схем в работе необходимо изучить существующие средства, выделить их основные преимущества и выбрать самый оптимальный вариант. Будут рассмотрены следующие: Enterprise Architect, Visual Paradigm, draw.io.

Enterprise Architect (EA) – CASE-инструмент для проектирования и конструирования программного обеспечения. EA поддерживает спецификацию UML2.0+, описывающую визуальный язык, которым могут быть определены модели проекта. [13]. В программе присутствуют широкие возможности для моделирования, с помощью нее можно создавать детальные и сложные модели.

Visual Paradigm — это комплексное и высокоэффективное решение UML, которое поможет вам нарисовать диаграммы вариантов использования для моделирования функций системы. Это приложение может поддерживать и предоставлять все необходимое для организации и поиска системных требований с помощью анализа вариантов использования. Используя этот мощный инструмент, вы также сможете управлять сценариями использования и писать новые с помощью редактора потока событий. [24]. Visual Paradigm имеет интуитивно понятный интерфейс что облегчает работу с ней новичку что является однозначным плюсом.

Draw.io — это бесплатный онлайн-сервис, который помогает создавать блок-схемы, прототипы, инфографику и диаграммы любого вида. Чаще всего его используют именно для построения диаграмм, поэтому недавно сервис переименовали в Diagrams.net. Но старое название по-прежнему в ходу. [10]. Draw.io так же имеет интуитивно понятный интерфейс к тому же он включает в себя все необходимые инструменты для рисования диаграмм и

других задач. Распространяется бесплатно с помощью онлайн-сервиса что позволяет сразу же приступить к работе, не нуждаясь в установке.

Для сравнения выделим следующие критерии:

- функциональность: этот критерий выделит количество поддерживаемых диаграмм и инструментов для редактирования;
- интерфейс: здесь будет учитываться эргономичность, простота использования для новичка и внешний вид;
- совместимость: возможность импорта и экспорта диаграмм, а также совместимость с разными форматами файлов;
- распространение: оценим, насколько популярна программа среди пользователей;
- цена и доступность: способы использования пользователем сервиса, стоимость, наличие демо версии и прочее.

Ниже представлена таблица 1, необходимая для сравнения выбранных программ моделирования в нотации UML.

Таблица 1 - Таблица сравнительной оценки

Критерии	Enterprise Architect	Visual Paradigm	draw.io
Функциональность	Высокая	Высокая	Средняя
Интерфейс	Сложный	Интуитивный	Интуитивный
Совместимость	Хорошая	Хорошая	Средняя
Распространение	Популярный	Популярный	Популярный
Цена и доступность	Платный	Платный	Бесплатный

Проведя сравнительный анализ можно сделать выбор в пользу draw.io благодаря следующим качествам.

Draw.io является самым простым в доступе среди сравниваемых программ, потому что начать работу можно за самое минимальное количество времени, так как нет необходимости в установке, нужно только перейти на онлайн-страницу сервиса, так же он не требует оплаты для использования.

Так же сервис имеет самый простой, интуитивно понятный интерфейс среди прочих, в нём нет нужды долго разбираться, он позволяет быстро освоить все необходимые инструменты и приступить к созданию диаграмм.

Высокая интеграция с различными сервисами так же является неоспоримым плюсом, такими как как Google Диск, Dropbox, OneDrive и другими. Благодаря этому можно сохранять созданные диаграммы в этих сервисах для дальнейшего использования и обмена с другими пользователями. А экспорт во многие используемые форматы файлов такие как PNG, JPEG, PDF, SVG, позволят более гибко использовать созданные диаграммы.

1.2 Сравнение и анализ приложений для генерации фонтов представленных на данный момент

На сегодняшний день есть уже несколько решений для данной задачи, которые по своему функционалу и исполнению несколько отличаются друг от друга. Возьмём для рассмотрения некоторые их них такие как веб-сервис «oleddisplay.squix», приложение «PixelPixie», веб-сервис «riyas.org».

«oleddisplay.squix» это веб приложение для работы с которым нужно лишь перейти по его ссылке, оно имеет интуитивно понятный интерфейс в котором можно выбрать необходимый фонт и его размер, но при этом отсутствует поддержка русского языка, сайт выполнен на английском языке. Хоть на сайте и можно выбирать многие фонты, при этом ни один из них не поддерживает кириллицу, только латинский алфавит, из-за чего он не может быть применим нами так как нам необходима поддержка кириллицы.

Приложение «PixelPixie» представляет из себя консольное приложение, которое не имеет интерфейса, а только принимает некоторый набор параметров перед началом работы, для того чтобы получить необходимый фонт необходимо самостоятельно нарисовать каждый символ в редакторе изображений по заранее заданному шаблону.

Веб сервис «riyas.org» имеет достаточно сложный для понимания новичку пользовательский интерфейс, который при это на английском языке. Работа с сервисом предполагает некоторый набор инструментов для того, чтобы рисовать символы по каждому пикселю (Пиксель — это одна из множества точек, составляющих изображение на экране электронного устройства, а также наименьшая единица растровой графики. [9]), что несколько облегчает работу по сравнению с рисованием символов в редакторе изображений.

Далее проведём сравнительный анализ этих решений по следующим критериям:

– функциональность – этот параметр определяет нужно ли пользователю самому рисовать символы и тратить на это много времени и сил;

– наличие русского интерфейса – есть ли поддержка русского языка;

– поддержка кириллицы – самая главная характеристика, так как без неё невозможно написать текст на русском языке;

– удобство использования – оценка того насколько просто работать с данным решением.

После проведения сравнительного анализа составим таблицу 2.

Таблица 2 - Сравнительная оценка генераторов для матричных фонтов

Критерии	oleddisplay.squix	PixelPixie	riyas.org
Функциональность	Выбор нужного фонта	Необходимо вручную рисовать фонт	Необходимо вручную рисовать фонт
Наличие русского интерфейса	Нет	Нет	Нет
Поддержка кириллицы	Нет	Да	Да
Удобство использования	Высокое	Низкое	Низкое

По таблице 2 видно что все они имеют разные преимущества и недостатки, например oleddisplay.squix хоть и имеет самый удобный и простой интерфейс имеет самый значительный недостаток, а именно отсутствие поддержки кириллицы в генерируемом фонте. PixelPixie и riyas.org тоже имеют значительный недостаток, в них фонт приходится рисовать самостоятельно, что занимает много времени, но из-за этого есть возможность написать кириллический фонт.

В совокупности критериев нет подходящих для нас решений, которые бы полностью нас устраивали.

1.3 Разработка и анализ текущей модели «КАК ЕСТЬ»

Разработка модель «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ» будет производиться с помощью нотации BPMN. Нотация BPMN (Business Process Management Notation, нотация моделирования бизнес-процессов) — нотация класса WorkFlow, один из наиболее распространенных методов описания бизнес-процессов на сегодня. В нотации BPMN используется базовый набор интуитивно понятных элементов, которые позволяют определять сложные семантические конструкции. [7]

Для понимания того, как устроена работа приложений для генерации фонтов на данный момент разработаем модель «КАК ЕСТЬ». Данная модель позволяет систематизировать протекающие в данный момент процессы, а также используемые информационные объекты. На основе этого выявляются узкие места в организации и взаимодействии бизнес-процессов, определяется необходимость тех или иных изменения в существующей структуре. [15] Далее на рисунке 1 показана диаграмма в нотации BPMN.

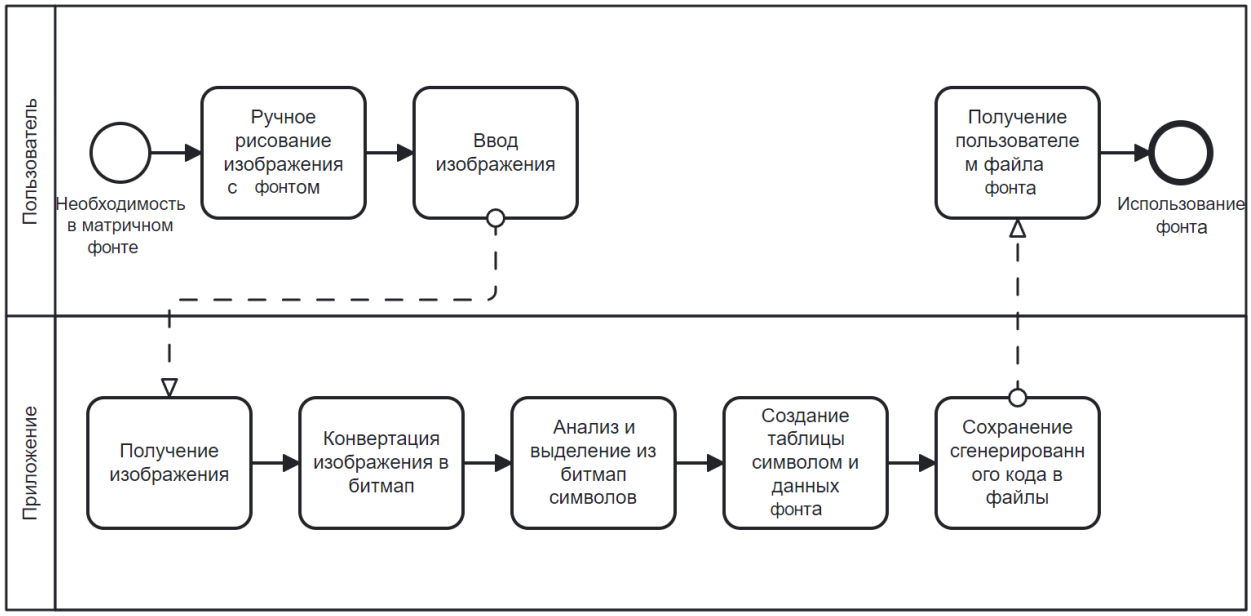

Рисунок 1 - Диаграмма, представляющая работу приложения для генерации фонтов «КАК ЕСТЬ»

В настоящее время приложения для генерации матричных фонтов предполагает самостоятельное рисование фонта с помощью редактора изображений, пользователю приходится рисовать фонт по пикселю для получения желаемого результата. Помимо этого, нет такого приложения, на котором интерфейс был представлен на русском языке, что мешает работе пользователям, которые не знают английского языка. Сама работа в таких приложениях происходит следующим образом: пользователь рисует каждый символ фонта в определенном формате что занимает большое количество сил и времени, после чего загружает полученное изображение в программу, которая уже на основе изображения генерирует фонт.

1.4 Разработка модели «КАК ДОЛЖНО БЫТЬ»

Далее необходимо разработать модель того, как мы хотим видеть приложение. Как правило, данная модель создается на основе AS IS, с устранением недостатков в существующей организации бизнес-процессов, а также с их совершенствованием и оптимизацией. Это достигается за счет

устранения выявленных на базе анализа AS IS узких мест. Подразумевается, что это позволяет существенно снизить риск проявления автоматизации как исключительно источника затрат из-за автоматизации несовершенных процессов. [23] Ниже на рисунке 2 представлена диаграмма в нотации BPMN.

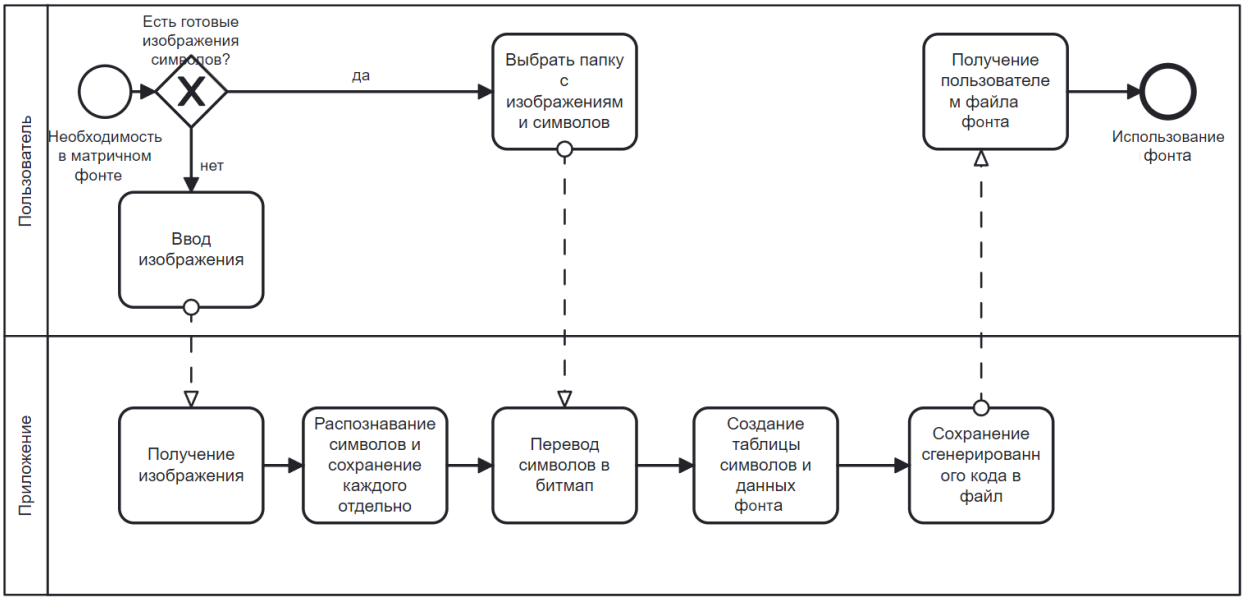

Рисунок 2 - Диаграмма, представляющая работу приложения для генерации фонтов «КАК ДОЛЖНО БЫТЬ»

По сравнению с моделью «КАК ЕСТЬ» здесь нет процесса рисования фонта вручную, что может занять много времени и сил у пользователя, а в случае, если необходимы не минимально возможные фонты с малым размером, например 8x8, а большие, к примеру 12x18 это займёт ещё больше времени, а также уже может стать невозможным для людей, которые имеют мало опыта в рисовании. Мы добиваемся этого тем, что добавляем в программу модуль использующий искусственный интеллект, для того чтобы любого изображения, на котором есть все необходимые символы для генерации фонта, сформировать изображение, которое готово для последующей его конвертации в битмап(Bitmap (битмап) — это формат изображения в компьютерной графике, который представляет собой матрицу

пикселей, каждый из которых содержит информацию о цвете и яркости.[12]). Таким образом работа пользователя состоит только в том, чтобы найти понравившийся ему фонт и загрузить его изображение в программу.

1.5 Разработка требований к разрабатываемому приложению

Взяв за основу данные, которые мы получили выше можно разработать список необходимых требований, которые должны учитываться в разрабатываемом приложении.

Во-первых, самое главное, приложение должно поддерживать генерацию фонтов с символами кириллицы.

Вторым же станет удобство использования, мы должны сделать такое интуитивно понятное приложение, при этом, используя которое, нет необходимости тратить время на рисование фонта вручную.

Так же необходимо разработать интерфейс, выполненный на русском языке, потому как это очень облегчает работу пользователя с приложением. Пользовательский интерфейс (ПИ) играет важнейшую роль в функционировании человеко-машинной системы, поскольку от него зависит продуктивность взаимодействия оператора и информационного комплекса. [4]

В ходе выполнения первой главы были рассмотрены разные CASE-средства для описания процессов, было проведено сравнение, в ходе которого был выбран сервис Draw.io. Далее рассмотрены существующие приложения для генерирования матричных фонтов. Были разработаны модели «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ» для лучшего понимания недостатков и узких мест существующего процесса. Благодаря проведенной ранее работе были созданы требования к разрабатываемому приложению, которым необходимо следовать при реализации программного обеспечения.

Глава 2. Проектирование приложения

2.1 Выбор языка программирования

Java - широко используемый язык программирования для написания приложений. Язык Java широко использовался на протяжении более двух десятилетий. Миллионы приложений Java используются и сегодня. Java – это многоплатформенный, объектно-ориентированный и сетецентрический язык, который сам по себе может использоваться как платформа. Это быстрый, безопасный и надежный язык программирования для всего: от мобильных приложений и корпоративного программного обеспечения до приложений для работы с большими данными и серверных технологий [18]. Так же в стандартной библиотеке (Применительно к программированию, под библиотекой понимается набор готовых объектов, функций, классов и других вспомогательных инструментов, которые необходимые для решения различных задач при создании ПО. [1]) есть большое количество классов и методов, что позволяет создавать приложения с его помощью намного проще и удобнее.

«Язык программирования C++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день C++ является одним из самых популярных и распространенных языков.» [3] Он широко используется при разработке различного программного обеспечения, а также довольно популярен при написании компьютерных игр. Плюсами этого языка являются производительность, универсальность, поддержка.

И наконец рассмотрим язык программирования Python. Он достаточно гибкий в сравнении с другими, на нём можно реализовать практически любую задумку. Python поддерживает несколько стилей программирования. Он не принуждает разработчика придерживаться определенной парадигмы.

Python поддерживает объектно-ориентированное и процедурное программирование. Существует и ограниченная поддержка функционального программирования. Язык обладает чётким и последовательным синтаксисом, продуманной модульностью и масштабируемостью, благодаря чему исходный код написанных на Python программ легко читаем. [2] Другим же плюсом является то, что это очень популярный язык программирования, благодаря чему для него существует очень много документации для реализации нужных решений. Так же есть большое количество библиотек, которые помогут при решении различных задач.

2.2 Выбор необходимых библиотек языка Python

Далее нам нужно будет выбрать необходимые библиотеки для реализации приложения для генерации матричных фонтов. Рассмотрим и опишем каждую из выбранных библиотек.

Tkinter – это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя (graphical user interface – GUI). Эта библиотека написана на языке программирования Tcl.

Под графическим интерфейсом пользователя (GUI) подразумеваются все те окна, кнопки, текстовые поля для ввода, скроллеры, списки, радиокнопки, флажки и другие элементы, которые вы видите на экране, открывая то или иное приложение. Через них вы взаимодействуете с программой и управляете ею. [14]

Эта библиотека понадобится в проекте для создания графического интерфейса приложения. С её помощью можно будет несложно создать необходимые элементы пользовательского интерфейса такие как окно приложения, кнопки и текстовые поля.

Библиотека изображений Python (расширение PIL) — это де-факто пакет обработки изображений для языка Python. Он включает в себя легкие

инструменты обработки изображений, которые помогают редактировать, создавать и сохранять изображения. Pillow поддерживает большое количество форматов файлов изображений, включая BMP, PNG, JPEG и TIFF. [21] Эта библиотека позволяет обрабатывать изображения, а если точнее, то с её помощью можно сохранять и преобразовывать изображения в разных форматах, так же есть возможность изменять размеры, обрезать и поворачивать. Мы будем использовать эту библиотеку для преобразования входного изображения в ч/б формат, создания битмапов и сохранения изображений.

OpenCV (Библиотека компьютерного зрения с открытым исходным кодом) — это мощная и широко используемая библиотека для обработки изображений и задач компьютерного зрения. Используя Python, популярный и простой в освоении язык программирования, разработчики могут создавать эффективные приложения для таких задач, как обработка изображений, обнаружение объектов. [8] Будет использоваться в приложении для обработки изображений, а точнее преобразование в градации серого, а также для получения координат символов на изображении.

NumPy — это фундаментальный пакет для научных вычислений на Python. Это библиотека Python, которая предоставляет объект многомерного массива, различные производные объекты (например, маскированные массивы и матрицы), а также набор подпрограмм для быстрых операций с массивами, включая математические, логические манипуляции с фигурами, сортировку, выбор, ввод-вывод, дискретные преобразования Фурье, основы линейной алгебры, основные статистические операции, случайное моделирование и многое другое. [19]

Pytesseract — это полезная библиотека Python, предоставляющая интерфейс для механизма OCR Tesseract. Сначала он предварительно обрабатывает входное изображение, чтобы улучшить его качество. После этого он проверяет расположение/ориентацию страницы для определения текстовых блоков, абзацев и символов. Сопоставляя шаблоны в

сегментированных областях, Tesseract распознает отдельные символы с помощью комбинации машинного обучения и традиционных подходов к обработке изображений. Чтобы повысить точность и обрабатывать множество языков, он использует языковые модели. После идентификации для улучшения результатов используются операции постобработки, такие как проверка орфографии и исправление ошибок. [16] В нашем приложении будем использовать эту библиотеку для распознавания и вырезки символов из изображения для дальнейшего преобразования в файл фонта.

PyInstaller объединяет приложение Python и все его зависимости в один пакет. Пользователь может запустить упакованное приложение без установки интерпретатора Python или каких-либо модулей. PyInstaller поддерживает Python 3.8 и новее и правильно объединяет многие основные пакеты Python, такие как numpy, matplotlib, PyQt, wxPython и другие. [20] Будет использоваться для того, чтобы создать .exe файл приложения, чтобы можно было запустить его на любом компьютере без необходимости установки python и библиотек.

2.3 Проектирование архитектуры приложения для генерации матричного фонта

Разделим структуру приложения на 3 модуля: Обработка изображений и извлечение символов, сохранение изображений символов, генерация файла фонта.

Обработка изображений и извлечение символов: с помощью этого модуля будем обрабатывать изображение, которое указал пользователь, используя комбинацию библиотек OpenCV и pytesseract.

Сохранение изображений символов – с помощью этого модуля будем сохранять изображения символов в указанную папку для последующего использования программой.

Генерация файла фонта – этот модуль будет завершающим и производить генерацию файла фонта из изображений символов, созданными ранее.

2.4 Разработка диаграммы последовательности

Диаграмма последовательности — это диаграмма унифицированного языка моделирования (UML), которая иллюстрирует последовательность сообщений между объектами. Диаграмма последовательности состоит из группы объектов. Которые представлены линиями жизни и сообщениями, которыми они обмениваются во время взаимодействия.

Диаграмма последовательности показывает последовательность сообщений, передаваемых между объектами. Диаграммы последовательности также могут отображать структуры управления между объектами. Например, линии жизни на диаграмме последовательности банковского сценария могут представлять клиента, банковского кассира или менеджера банка. Общение между клиентом, кассиром и менеджером представлено сообщениями, передаваемыми между ними. Диаграмма последовательности показывает объекты и сообщения между объектами. [22]

Разработаем диаграмму последовательностей для приложения генерации матричных фонтов, дальше на рисунке 3 будет представлена диаграмма последовательности.

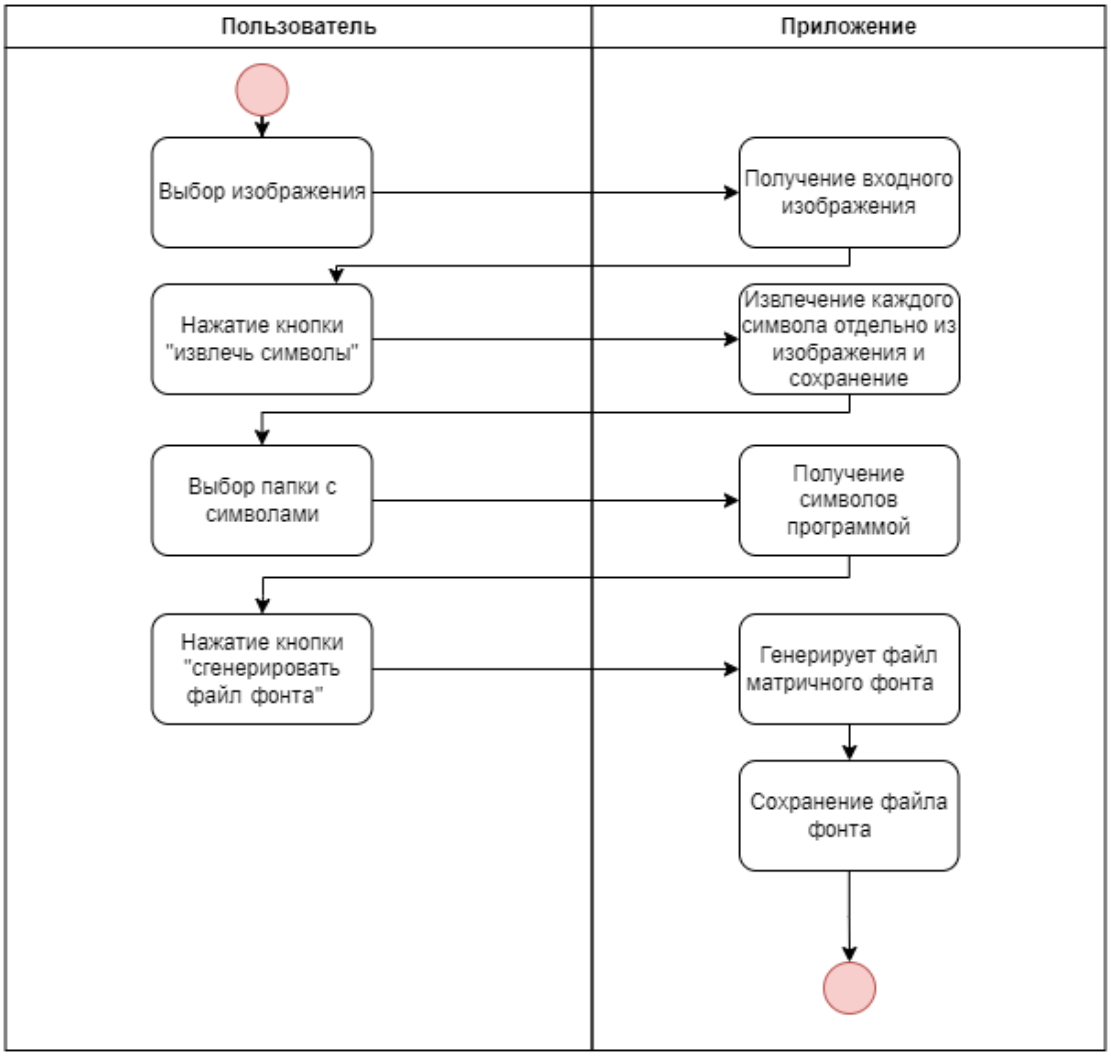

Рисунок 3 – Диаграмма последовательности

На диаграмме представлен процесс использования программы. Сначала пользователь выбирает нужное изображение и загружает его в программу, после чего приложение получает изображение, далее пользователь нажимает кнопку «извлечь символы» приложение обрабатывает изображение, распознаёт, выделяет и сохраняет отдельно каждый символ. Пользователь выбирает директорию, в которой находятся изображения с символами будущего фонта благодаря чему программа получает эти изображения. И в завершение пользователь нажимает кнопку «сгенерировать файл фонта», посылая команду приложению, вследствие происходит чего генерирование

файла фонта на основе символов из выбранной папки и сохраняет файл фонта

2.5 Разработка диаграммы вариантов использования

Диаграмма вариантов использования или диаграмма прецедентов (англ. use case diagram) — это графический инструмент универсального языка моделирования (UML), который используется для описания функциональных требований к системе, ее возможных сценариев использования и взаимодействия системы с внешними сущностями (акторами). Суть диаграммы вариантов использования заключается в том, что она помогает лучше понять требования к системе и определить ее функциональные возможности. Она используется для определения сценариев использования системы и для выявления потенциальных проблем взаимодействия между пользователями и системой. Диаграмма вариантов использования является основным инструментом для описания поведения системы на ранней стадии ее проектирования. [5] Для лучшего понимания как пользователь сможет использовать приложение разработаем диаграмму вариантов использования. Диаграмма представлена на рисунке 4.

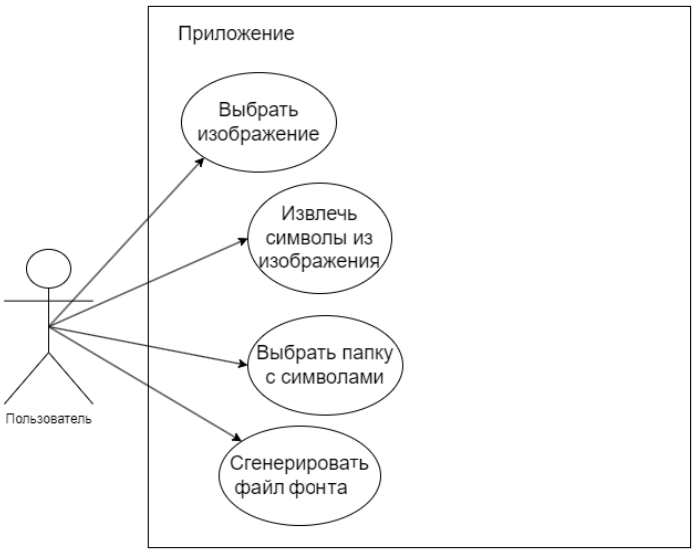

Рисунок 4 – Диаграмма вариантов использования

С помощью диаграммы, представленной на рисунке 5 можно понять как пользователь может взаимодействовать с приложением. А именно может выбрать изображение, извлечь символы из изображения, выбрать папку с символами и сгенерировать файл фонта.

В этой главе необходимо было спроектировать приложение, для этого сначала были проанализированы некоторые существующие языки программирования, в ходе чего для разработки был выбран язык Python поскольку он больше подходит для работы с изображениями и имеет большое количество документации и удобных библиотек. Далее были описаны выбранные библиотеки для Python, которые помогут при разработке приложения. Так же при проектировании архитектуры приложения было решено разделить его на 3 основных модуля: обработка изображений и извлечение символов, сохранение изображений символов, генерация файла фонта. После этого были разработаны диаграмма последовательности и диаграмма вариантов использования для того, чтобы лучше понимать как пользователь должен взаимодействовать с приложением.

Глава 3 Разработка приложения для генерации матричных фонтов

3.1 Реализация основных модулей приложения для генерирования матричных фонтов

Модуль 1 - Обработка изображений и извлечение символов. Этот модуль отвечает за обработку изображения, распознавания и извлечения символов в отдельные изображения. Для этого используем OpenCV и pytesseract, благодаря чему получаем и вырезаем символы. Подробно разберём каждую функцию модуля. Далее на рисунке 5 представлена функция для предварительной обработки изображения.

```
8   # Функция для предварительной обработки изображения
9   def preprocess_image(image_path):
10      image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
11      _, thresh = cv2.threshold(image, thresh: 128, maxval: 255, cv2.THRESH_BINARY_INV)
12      return thresh
13
```

Рисунок 5 – Функция для предварительной обработки изображения

С помощью это функции мы предварительно обрабатываем изображение, а именно переводим его в градации серого и применяем пороговую обработку для того, чтобы изображение было более контрастным и лучше распознавались символы. В итоге мы получаем изображение в ч/б формате с более высокой контрастностью.

Далее на рисунке 6 рассмотрим функцию для извлечения символов.

```python
# Функция для извлечения символов с использованием pytesseract
1 usage
def extract_characters(image_path, padding=0):
    image = Image.open(image_path)
    custom_config = r'--oem 3 --psm 6'
    data = pytesseract.image_to_boxes(image, config=custom_config, lang='rus')
    img = cv2.imread(image_path)
    h, w, _ = img.shape

    character_images = []
    for d in data.splitlines():
        d = d.split()
        char = d[0]
        x, y, x2, y2 = int(d[1]), int(d[2]), int(d[3]), int(d[4])
        y = h - y
        y2 = h - y2

        x = max(0, x - padding)
        y = min(h, y + padding)
        x2 = min(w, x2 + padding)
        y2 = max(0, y2 - padding)

        char_image = img[y2:y, x:x2]
        char_image = cv2.cvtColor(char_image, cv2.COLOR_BGR2GRAY)
        _, char_image = cv2.threshold(char_image, thresh: 128, maxval: 255, cv2.THRESH_BINARY_INV)
        character_images.append((char, char_image))

    return character_images
```

Рисунок 6 – Функция для извлечения символов

Эта функция нужна для того, чтобы распознавать и извлекать символы из входного изображения с помощью библиотеки pytesseract, далее она определяет координаты символов, после чего вырезает их в виде отдельных изображений.

С помощью параметров «custom_config = r'--oem 3 --psm 6'» и «data = pytesseract.image_to_boxes(image, config=custom_config, lang='rus')» выбираем режим работы и настраиваем на распознавание символов русского алфавита. Далее вырезается область изображения, в которой находится распознанный символ, преобразовывается в градации серого и добавляется в список «character_images». На этом работа модуля заканчивается и передаётся следующему модулю.

Следующий модуль «сохранение изображений символов» представлен на рисунке 7, он нужен для сохранения полученных изображений символов.

```
# Функция для сохранения изображений символов
1 usage
def save_characters(character_images, output_dir='character'):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    max_width = 0
    max_height = 0

    for char, char_image in character_images:
        max_width = max(max_width, char_image.shape[1])
        max_height = max(max_height, char_image.shape[0])

    for i, (char, char_image) in enumerate(character_images):
        char_image_inv = cv2.bitwise_not(char_image)
        ascii_code = ord(char)
        safe_char = char if char.isalnum() else f'char_{i}'
        Image.fromarray(char_image_inv.astype(np.uint8)).save(f'{output_dir}/{ascii_code}_{i}_{safe_char}.png')

    blank_image = np.zeros( shape: (max_height, max_width), dtype=np.uint8) + 255
    Image.fromarray(blank_image).save(f'{output_dir}/32_.png')
```

Рисунок 7 – Модуль сохранения изображений символов

Здесь проверяется существование папки для сохранения в неё изображений, если такой директории нет, создаёт её. Дальше мы проходим по каждому символу и получаем их максимальное значение, это нужно для того, чтобы задать символу пробела максимальный размер символа, при этом сохраняем символы с указанием ASCII (ASCII — это таблица кодировки символов, в которой каждой букве, числу или знаку соответствует определенное число. В них входят латинские буквы, цифры, знаки препинания и управляющие символы. [17]) кода в названии изображения. В целом этот модуль нужен для сохранения полученных с помощью предыдущего модуля символов.

Следующий модуль «генерация файла фонта» представлен на рисунке 8. Для начала рассмотрим функцию преобразования изображений в битмап. Такой формат является основным в большинстве графических редакторов и

используется для отображения реалистичных и детализированных изображений.

Особенностью Bitmap является то, что каждый пиксель в изображении имеет конкретные координаты, которые определяют его положение на экране. Каждый пиксель может быть настроен на определенное значение цвета, которое определяет его оттенок и насыщенность. Это позволяет создавать изображения с высокой точностью и детализацией. [16] С помощью битмапа можно отображать различные изображения, каждый бит представляет собой точку, которая может быть любым цветом, но в нашем случае, когда есть только либо сигнал 1 либо 0, мы можем работать только в ч/б формате.

```
69  # Функция для преобразования изображения в битмап
    1 usage
70  def convert_image_to_bitmap(image_path):
71      img = Image.open(image_path).convert('L')
72      width, height = img.size
73      pixels = list(img.getdata())
74      bitmap = []
75
76      for y in range(height):
77          row = ''
78          for x in range(width):
79              pixel = pixels[y * width + x]
80              row += '1' if pixel < 128 else '0'
81          bitmap.append(row)
82
83      return bitmap, width, height
```

Рисунок 8 – Функция для преобразования изображения в битмап

Эта функция принимает изображение и конвертирует его в битовую строку с порогом 128. Далее, с помощью функции, представленной на рисунке 9, нам нужно преобразовать битмап в бинарные данные.

```python
# Функция для преобразования битмапа в бинарные данные
1 usage
def bitmap_to_binary_data(bitmap, width, height):
    binary_data = []
    for row in bitmap:
        row_data = []
        for x in range(0, width, 8):
            byte = row[x:x+8]
            byte = byte.ljust(8, '0')
            row_data.append(f'{int(byte, 2):02X}')
        binary_data.extend(row_data)
    return binary_data
```

Рисунок 9 – Функция для преобразования битмап в бинарные данные

Функция получает на вход битмап, ширину и высоту символа, после чего создаёт массив байтов в шестнадцатеричном формате. Далее идёт функция, представленная на рисунках 10-11, она необходима для генерации файла фонта.

```python
# Функция для генерации файла шрифта
1 usage
def generate_font_file(image_folder, output_folder, font_file):
    characters = sorted([f for f in os.listdir(image_folder) if f.endswith('.png')],
                        key=lambda x: int(x.split('_')[0]))

    jump_table = []
    font_data = []

    max_width = 0
    max_height = 0
    offset = 0

    for char in characters:
        image_path = os.path.join(image_folder, char)
        bitmap, width, height = convert_image_to_bitmap(image_path)
        binary_data = bitmap_to_binary_data(bitmap, width, height)
        x_advance = width

        max_width = max(max_width, width)
        max_height = max(max_height, height)

        jump_table.append((offset & 0xFF, (offset >> 8) & 0xFF, width, height, x_advance))
        font_data.append((char, binary_data))
        offset += len(binary_data)
```

Рисунок 10 – Функция для генерации файла фонта часть 1

Функция сначала получает список всех изображения символов, далее сортируется по ASCII коду, который извлекается из имени изображения. После чего каждый символ преобразуется в битмап, определяются максимальные ширина и высота, которые вместе со смещением которые будут записаны в таблицу переходов.

```
    with open(font_file, 'w') as f:
        f.write('// Generated font file\n')
        f.write('const char Russian_font_16[] PROGMEM = {\n')
        f.write(f'\t0x{max_width:X}, // Width: {max_width}\n')
        f.write(f'\t0x{max_height:X}, // Height: {max_height}\n')
        f.write(f'\t0x20, // First Char: 32\n')
        f.write(f'\t0x{len(characters):X}, // Number of Chars: {len(characters)}\n\n')

        f.write('\t// Jump Table:\n')
        for i, entry in enumerate(jump_table):
            char_code = int(characters[i].split('_')[0])
            f.write(f'\t0x{entry[0]:02X}, 0x{entry[1]:02X}, 0x{entry[2]:02X}, 0x{entry[3]:02X}, // {chr(char_code)}\n')

        f.write('\n\t// Font Data:\n')
        for char, binary_data in font_data:
            char_code = int(char.split('_')[0])
            f.write('\t')
            for i, byte in enumerate(binary_data):
                f.write(f'0x{byte}, ')
                if (i + 1) % 8 == 0:
                    f.write('\n\t')
            f.write(f'// {chr(char_code)}\n')

        f.write('};\n')

    messagebox.showinfo( title: "Успех",  message: f"Файл шрифта {font_file} успешно сгенерирован!")
```

Рисунок 11 - Функция для генерации файла фонта часть 2

В этой части функции открывается файл для записи, сначала в файл записывается главная информация о фонте такая как ширина, высота, количество символов и первый символ. После чего идет создание таблицы переходов, которая содержит в себе информацию о каждом символе. Далее идёт запись бинарных данных каждого символа.

Далее необходимо реализовать сам интерфейс приложения. Для этого будем использовать ранее выбранную библиотеку Tkinter. На рисунке 12 показан код для создания главного экрана.

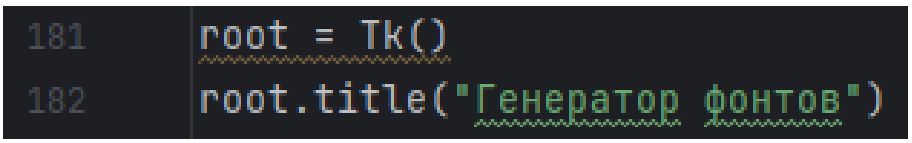

Рисунок 12 – Создание главного окна приложения

На рисунке показано как мы создаём главное окно приложения и устанавливаем заголовок приложения «Генератор фонтов». Так же нужно создать надпись, которая будет отображать выбранный путь к изображению, код представлен на рисунке 13.

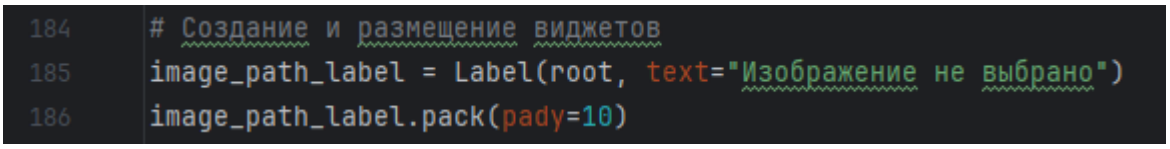

Рисунок 13 – Создание надписи для отображения пути к изображению

В случае если пользователь ещё не выбрал изображение будет написано «Изображение не выбрано» Далее нужно сделать саму кнопку для выбора изображения, код представлен на рисунке 14.

Рисунок 14 – Добавление кнопки «Выбрать изображение»

Когда пользователь нажмет на эту кнопку будет открыто диалоговое окно для выбора изображения. Дальше создадим кнопку для извлечения символов, код представлен на рисунке 15.

```
191  extract_button = Button(root, text="Извлечь символы", state="disabled", command=extract_and_save)
192  extract_button.pack(pady=10)
```

Рисунок 15 – Создание кнопки «Извлечь символы»

Изначально эта кнопка в выключенном состоянии, пока изображение не будет выбрано, её нельзя будет нажать. После нажатия на кнопку символы будут извлечены из изображения. Далее по аналогии создаём надпись, кнопку для выбора папки с символами, кнопку для генерации файла фонта, код представлен на рисунке 16.

```
194  image_folder_label = Label(root, text="Папка с символами не выбрана")
195  image_folder_label.pack(pady=10)
196
197  select_folder_button = Button(root, text="Выбрать папку с символами", command=select_image_folder)
198  select_folder_button.pack(pady=10)
199
200  generate_button = Button(root, text="Сгенерировать файл фонта", state="disabled", command=generate_font)
201  generate_button.pack(pady=10)
202
```

Рисунок 16 – Создание кнопок для генерации файла фонта

Здесь надпись так же изначально говорит о том, что изображение не выбрано, после выбора отображает путь к изображению. Кнопка «Выбрать папку с символами» открывает диалоговое окно для выбора папки с изображениями символов. Кнопка «Сгенерировать файл фонта» тоже изначально будет выключена и станет доступна после выбора папки.

Полный код приложения указан в Приложении А

3.2 Тестирование приложения для генерации матричных фонтов

Теперь, когда работа над самим приложением закончена, необходимо протестировать его. Потому как тестирование программного обеспечения является одной из ключевых стадий процесса разработки качественных и конкурентоспособный продуктов. [6] Сначала рассмотрим саму работу приложения.

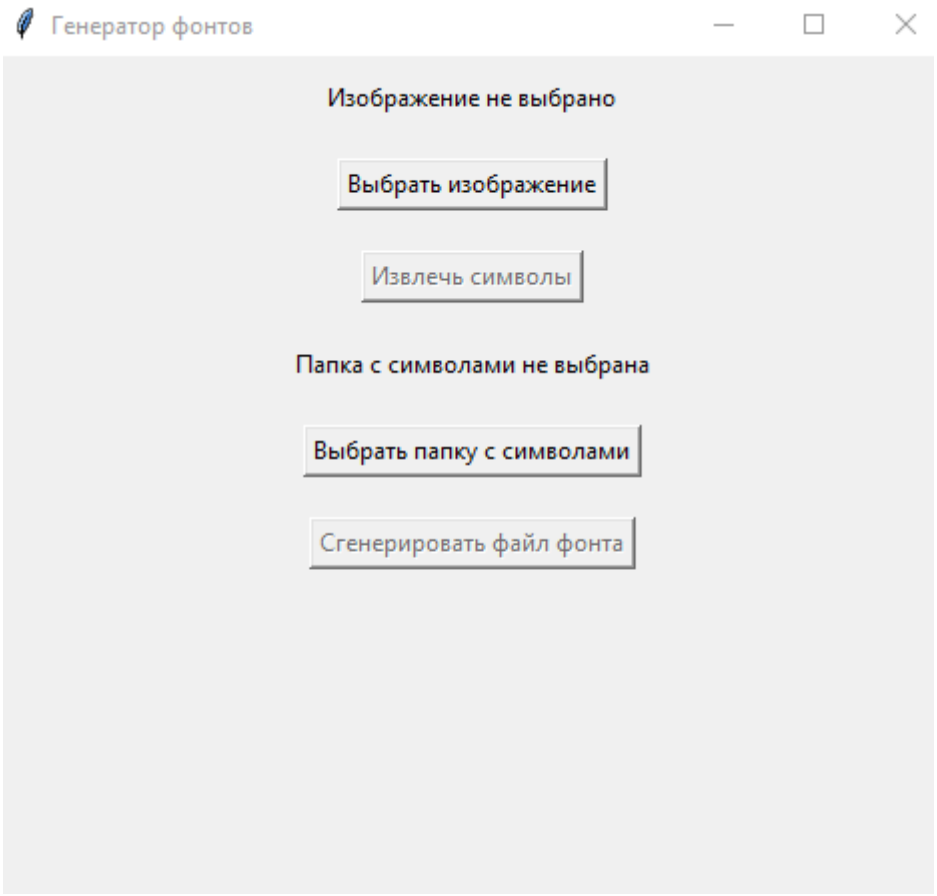

Рисунок 17 - Пользовательский интерфейс программы

На рисунке 17 представленном выше виден сам пользовательский интерфейс, на нём есть все необходимые кнопки для работы с приложением, такие как выбрать изображение, извлечь символы, выбрать папку с символами и сгенерировать файл фонта. Так же приложение обладает простым и интуитивно понятным пользовательским интерфейсом, в котором

будет несложно разобраться человеку, который использует его в первый раз. Далее, на рисунке 18 показано диалоговое окно, которое открывается после нажатия кнопки «Выбрать изображение»

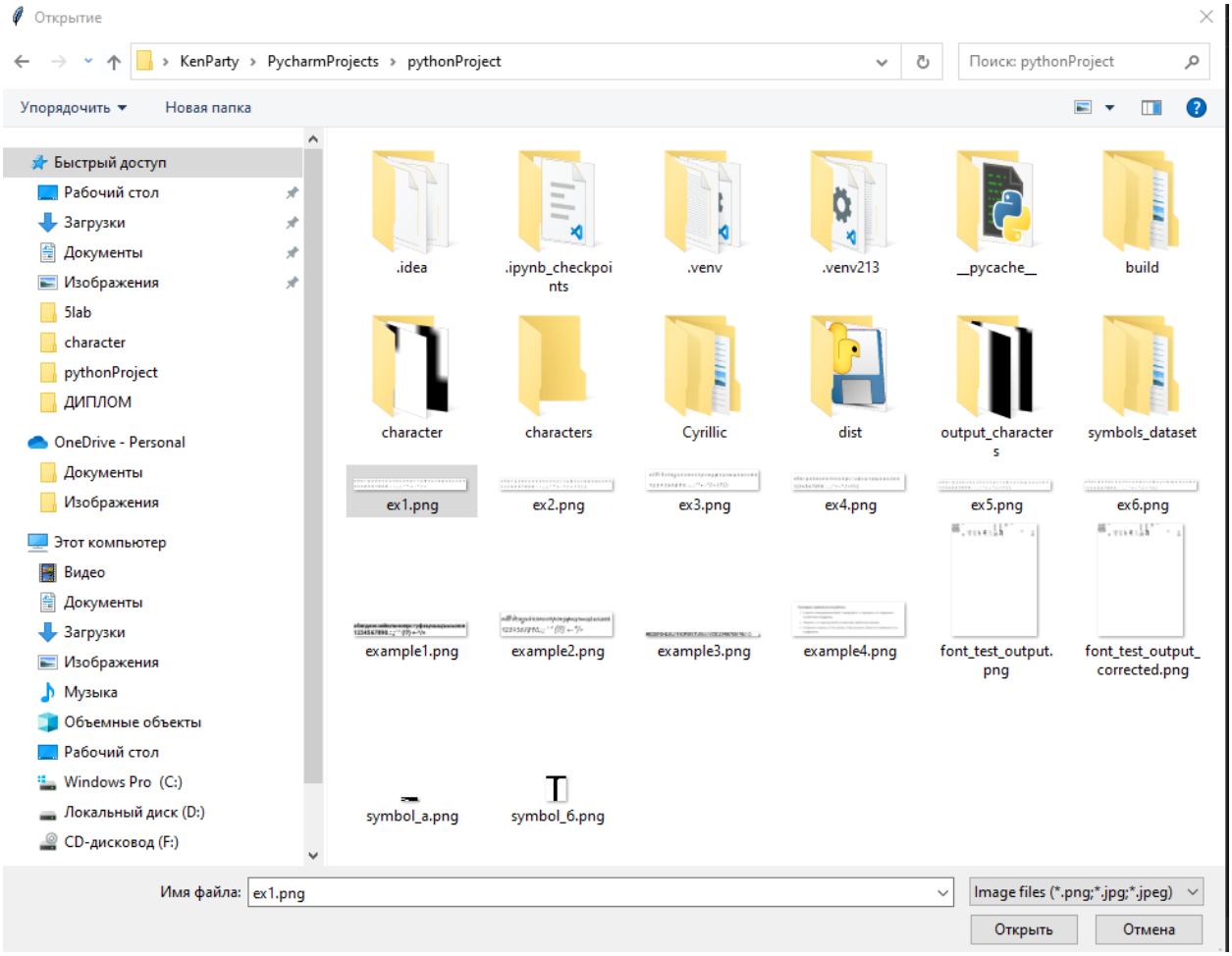

Рисунок 18 – Выбор изображения

На рисунке выше видно, что после нажатия этой кнопки открывается проводник, с помощью которого пользователь может выбрать нужное ему изображение.

На рисунке 19 показано оповещение, которое появляется при нажатии кнопки «извлечь символы»

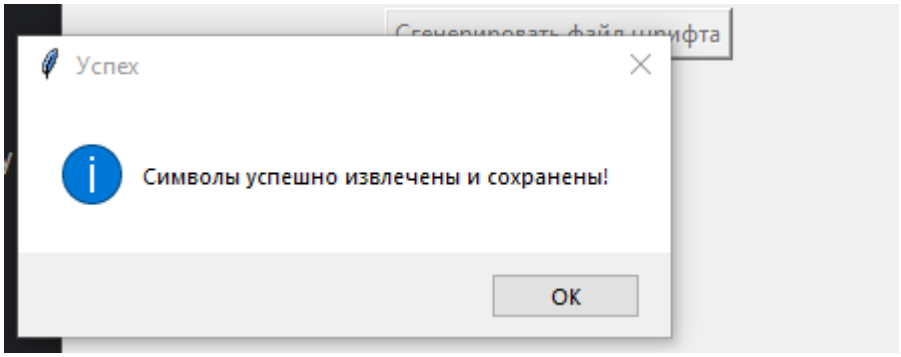

Рисунок 19 – Оповещение об успешном извлечении символов

Оповещение говорит о том, что символы были успешно извлечены из изображения и сохранены.

Дальше выбираем папку с символами для последующего генерирования из них файла фонта, работа кнопки «Выбрать папку с символами» аналогично кнопке «Выбрать изображение».

На рисунке 20 показано оповещение, которое появляется при нажатии на кнопку «Сгенерировать файл фонта»

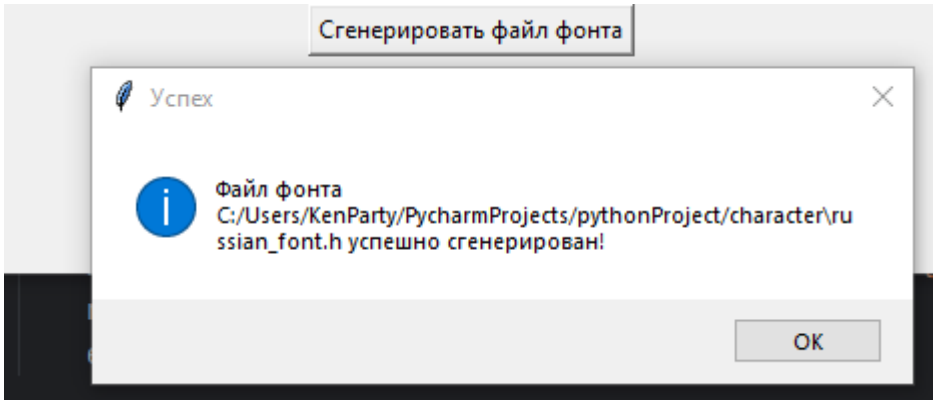

Рисунок 20 – Оповещение об успешном создании файла фонта

Оповещение говорит о том, что файл успешно сгенерирован.

Теперь нужно проверить корректность создания матричного фонта, для этого напишем функцию и посмотрим правильно ли программа создаёт фонт. Первая проверка будет на наборе символов «Arial».

На вход даём изображение представленное на рисунке 21, которое содержит все необходимые для нас символы Arial.

а б в г д е ё ж з и к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я

1 2 3 4 5 6 7 8 9 0 . : , ; ’ ” + - * / =

Рисунок 21 – Изображение, которое содержит символы Arial

На изображении присутствуют все нужные нам символы. Далее на рисунке 22 показана часть файла матричного фонта.

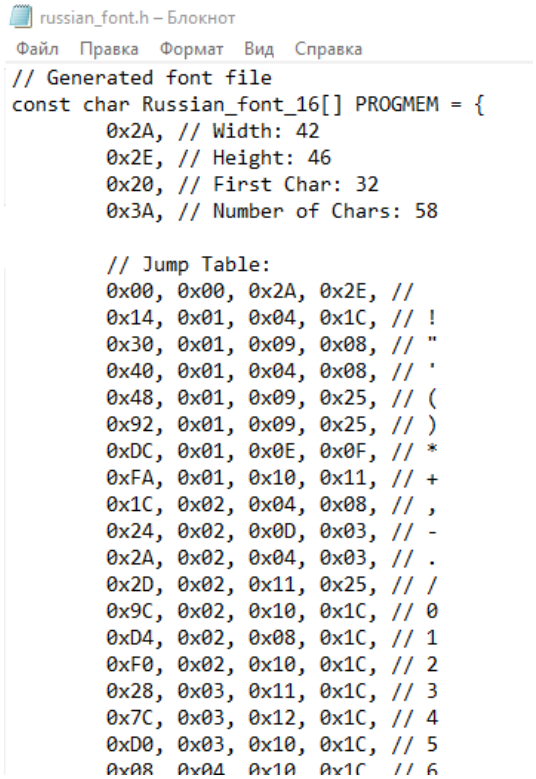

Рисунок 22 - Часть файла матричного фонта

На рисунке видно, что приложение действительно генерирует матричный фонт, но для человека сложно определить, что символ записывается действительно правильно, для этого будем использовать функцию, написанную специально для отображения символов матричного фонта. На рисунке 23 показано как отображается символ «ю», полученный из матричного фонта.

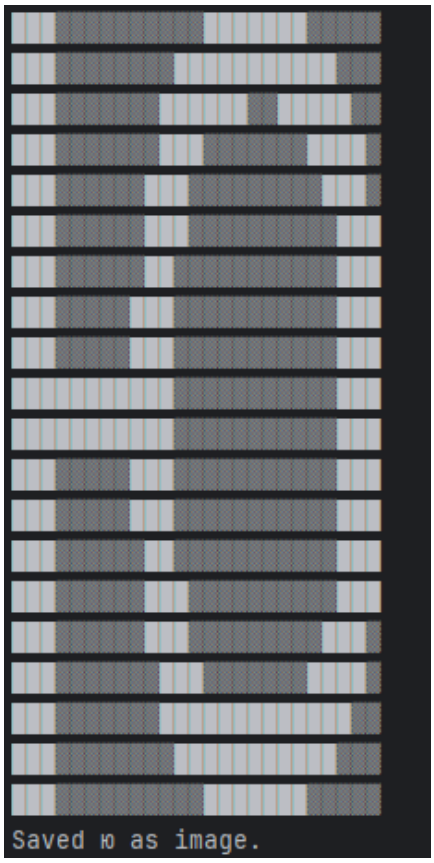

Рисунок 23 – Отображение символа «ю» полученное из файла фонта, созданного на основе символов «Arial»

На рисунке 23 можно увидеть, что выводится действительно символ «ю».

Далее специально выберем сложный необычный фонт, чтобы убедиться, что генерация происходит успешно при разных входных изображениях. создадим матричный фонт из набора символов «Coveat». Входное изображение представлено на рисунке 24

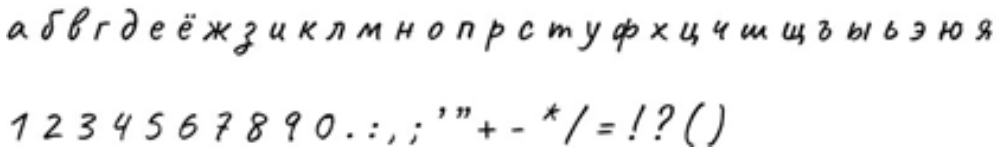

Рисунок 24 – Изображение, на котором представлены необходимые символы «Coveat»

На этом изображении так же представлены все необходимые символы для создания матричного фонта. Такой набор символов был выбран чтобы точно были заметны различия в генерируемых фонтах. Далее на рисунке 25 показано как отображается символ «ю», полученный из матричного фонта.

Рисунок 25 – Отображение символа «ю» полученное из файла фонта, созданного на основе символов «Coveat»

На рисунке 25 видно, что фонт действительно генерируется правильно и символ «ю» корректно отображается.

В ходе выполнения третьей главы было разработано приложение для генерирования матричных фонтов, подробно описаны все ключевые модули:

- модуль 1 для обработки изображений;
- модуль 2 для сохранения изображений символов;
- модуль 3 для генерирования и сохранения файла фонта.

Также были описаны функции, которые необходимы для работы программы.

Далее было проведено необходимое тестирование приложения. Тестирование показало, что приложение хорошо работает, соответствует поставленным требованиям и успешно генерирует матричный фонт.

Заключение

В процессе работы над выпускной квалификационной работой был решён ряд следующих задач.

Были найдены и проанализированы существующие решения для генерации матричных фонтов. Внимание уделялось функциональности, поддержке генерирования кириллических фонтов, удобству использования и наличию интерфейса на русском языке.

Далее была разработана модель «КАК ЕСТЬ» которая показывает процессы, которые используются в аналогах на данный момент. После чего была разработана модель «КАК ДОЛЖНО БЫТЬ». Выполнено сравнение этих двух моделей, указаны недостатки первой модели и преимущества последней для последующей разработки требований к разработке приложения.

Исходя из данных которые были получены во время сравнения моделей «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ», определены требования для дальнейшей разработки приложения для генерирования матричных фонтов. Основными требованиями стали:

- поддержка кириллицы в сгенерированных фонтах;
- удобство использования;
- интерфейс должен быть выполнен на русском языке.

Далее для реализации приложения было необходимо выбрать язык программирования. Для этого были выполнены анализ и сравнение существующих языков программирования. Язык Python был выбран поскольку для него есть множество библиотек, облегчающих работу с изображениями и наличию большого количества документации для решения различных задач.

После выбора языка программирования необходимо было выбрать библиотеки, которые помогут при разработке приложения. Были выбраны Tkinter, OpenCV, NumPy, Pytesseract, PIL и PyInstaller.

После чего нужно было спроектировать архитектуру приложения, было решено использовать 3 модуля. Первый модуль для обработки изображений и выделения символов. Второй модуль для сохранения изображений символов. Третий модуль для генерирования файла матричного фонта.

Разработана диаграмма последовательности работы пользователя с приложением для лучшего представления использования. Так же была разработана диаграмма вариантов использования.

Дальше были реализованы необходимые для работы приложения модули, подробно описана работа функций каждого модуля.

Последним шагом стало тестирование приложения, в ходе которого было установлено что приложение соответствует всем поставленным требованиям и выполняет свою главную функцию.

Список используемых источников

1. Библиотеки в программировании [Электронный ресурс]. URL: https://www.sravni.ru/kursy/info/biblioteki-v-programmirovanii/ (дата обращения 04.06.2024)
2. Бухаров Т. А., Нафикова А. Р., Мигранова Е. А. обзор языка программирования Python и его библиотек // Colloquium-journal. 2019. №3-1 (27). С. 1. URL: https://cyberleninka.ru/article/n/obzor-yazyka-programmirovaniya-python-i-ego-bibliotek (дата обращения: 03.06.2024)
3. Введение в C++ Язык программирования C++ [Электронный ресурс]. URL: https://metanit.com/cpp/tutorial/1.1.php (дата обращения 14.05.2024)
4. Величко Ю.И. Метод адаптации пользовательского интерфейс // Вестник Херсонского национального технического университета. 2013. №1 (46). С 1. URL: https://cyberleninka.ru/article/n/metod-adaptatsii-polzovatelskogo-interfeys (дата обращения: 04.06.2024).
5. Диаграмма вариантов использования (Use Case Diagram) [Электронный ресурс] URL: https://itonboard.ru/analysis/629-diagramma_variantov_ispolzovanija_use_case_diagram/ (дата обращения 03.06.2024)
6. Колмогоров Константин Алексеевич Документооборот в тестировании // КИО. 2006. №4. С 1. URL: https://cyberleninka.ru/article/n/dokumentooborot-v-testirovanii (дата обращения: 04.06.2024).
7. Нотации моделирования бизнес-процессов, Электронный ресурс]. URL: https://leanvector.ru/blog-eksperta/notatsii-modelirovaniya-biznes-protsessov/ (дата обращения: 27.05.2024)
8. Освоение OpenCV с помощью Python: Полное руководство по обработке изображений и компьютерному зрению [Электронный ресурс]. URL: https://vc.ru/u/1389654-machine-learning/661520-osvoenie-opencv-s-

pomoshchyu-python-polnoe-rukovodstvo-po-obrabotke-izobrazheniy-i-kompyuternomu-zreniyu (дата обращения 03.06.2024)

9. Пиксель [Электронный ресурс]. URL: https://media.contented.ru/glossary/piksel/ (дата обращения 04.06.2024)

10. Сервис Draw.io [Электронный ресурс]. URL: https://practicum.yandex.ru/blog/vozmozhnosti-servisa-drawio/ (дата обращения 02.02.2024)

11. Фонт [Электронный ресурс]. URL: https://dic.academic.ru/dic.nsf/ruwiki/278369 (дата обращения 04.06.2024)

12. Что такое битмап? [Электронный ресурс]. URL: https://uchet-jkh.ru/i/cto-takoe-bitmap/ (дата обращения 03.06.2024)

13. Что такое Enterprise Architect [Электронный ресурс]. URL: https://www.uml2.ru/faq/faq-ea/1/ (дата обращения 02.06.2024)

14. Что такое Tkinter [Электронный ресурс]. URL: https://younglinux.info/tkinter/tkinter (дата обращения 03.06.2024)

15. AS-IS модель [Электронный ресурс]. URL: https://piter-soft.ru/knowledge/glossary/process/as-is-model.html (дата обращения 03.06.2024)

16. A Guide to Python Tesseract [Электронный ресурс] URL: https://builtin.com/articles/python-tesseract#:~:text=Pytesseract%20is%20a%20Python%20library,documents%20or%20other%20visual%20media. (дата обращения 03.06.2024)

17. ASCII [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/ascii/ (дата обращения 04.06.2024)

18. Baesens, B. Beginning Java Programming: The Object-Oriented Approach / B. Baesens, A. Backiel, S. Vanden Broucke. – 1st edition, Wrox, 2015. 620 с.

19. NumPy documentation [Электронный ресурс]. URL: https://numpy.org/doc/stable/#numpy-documentation (дата обращения 03.06.2024)

20. PyInstaller Manual [Электронный ресурс] URL: https://pyinstaller.org/en/stable/ (дата обращения 03.06.2024)

21. Python: Pillow (a fork of PIL) [Электронный ресурс]. URL: https://www.geeksforgeeks.org/python-pillow-a-fork-of-pil/?ref=header_search (дата обращения 03.06.2024)

22. Sequence diagrams [Электронный ресурс] URL: https://www.ibm.com/docs/en/rsm/7.5.0?topic=uml-sequence-diagrams (дата обращения 03.06.2024)

23. TO-BE модель [Электронный ресурс] URL: https://piter-soft.ru/knowledge/glossary/process/to-be-model.html (дата обращения 03.06.2024)

24. Visual Paradigm Enterprise [Электронный ресурс]. URL: https://www.syssoft.ru/visual-paradigm/visual-paradigm-enterprise/ (дата обращения 02.04.2024)

Приложение А

Код приложения для генерирования матричных фонтов

```
import os
import cv2
import numpy as np
from tkinter import Tk, Label, Button, filedialog, messagebox
from PIL import Image, ImageDraw
import pytesseract

# Функция для предварительной обработки изображения
def preprocess_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    _, thresh = cv2.threshold(image, 128, 255, cv2.THRESH_BINARY_INV)
    return thresh

# Функция для извлечения символов с использованием pytesseract
def extract_characters(image_path, padding=0):
    image = Image.open(image_path)
    custom_config = r'--oem 3 --psm 6'
    data = pytesseract.image_to_boxes(image, config=custom_config, lang='rus')
    img = cv2.imread(image_path)
    h, w, _ = img.shape
    character_images = []
    for d in data.splitlines():
        d = d.split()
        char = d[0]
          x, y, x2, y2 = int(d[1]), int(d[2]), int(d[3]), int(d[4])
        y = h – y
```

```
        y2 = h - y2

        x = max(0, x - padding)
        y = min(h, y + padding)
        x2 = min(w, x2 + padding)
        y2 = max(0, y2 - padding)

        char_image = img[y2:y, x:x2]
        char_image = cv2.cvtColor(char_image, cv2.COLOR_BGR2GRAY)
        _, char_image = cv2.threshold(char_image, 128, 255, cv2.THRESH_BINARY_INV)
        character_images.append((char, char_image))

    return character_images
# Функция для сохранения изображений символов
def save_characters(character_images, output_dir='character'):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    max_width = 0
    max_height = 0

    for char, char_image in character_images:
        max_width = max(max_width, char_image.shape[1])
        max_height = max(max_height, char_image.shape[0])

    for i, (char, char_image) in enumerate(character_images):
        char_image_inv = cv2.bitwise_not(char_image)
```

```
        ascii_code = ord(char)
        safe_char = char if char.isalnum() else f'char_{i}'

Image.fromarray(char_image_inv.astype(np.uint8)).save(f'{output_dir}/{ascii_cod
e}_{i}_{safe_char}.png')

    blank_image = np.zeros((max_height, max_width), dtype=np.uint8) + 255
    Image.fromarray(blank_image).save(f'{output_dir}/32_.png')

# Главная функция для извлечения символов из изображения
def extract_and_save_characters(image_path):
    character_images = extract_characters(image_path)
    save_characters(character_images)
    messagebox.showinfo("Успех", "Символы успешно извлечены и
сохранены!")

# Функция для преобразования изображения в битмап
def convert_image_to_bitmap(image_path):
    img = Image.open(image_path).convert('L')
    width, height = img.size
    pixels = list(img.getdata())
    bitmap = []

    for y in range(height):
        row = ''
        for x in range(width):
            pixel = pixels[y * width + x]
            row += '1' if pixel < 128 else '0'
```

```
            bitmap.append(row)

    return bitmap, width, height

# Функция для преобразования битмапа в бинарные данные
def bitmap_to_binary_data(bitmap, width, height):
    binary_data = []
    for row in bitmap:
        row_data = []
        for x in range(0, width, 8):
            byte = row[x:x+8]
            byte = byte.ljust(8, '0')
            row_data.append(f'{int(byte, 2):02X}')
        binary_data.extend(row_data)
    return binary_data

# Функция для генерации файла фонта
def generate_font_file(image_folder, output_folder, font_file):
    characters = sorted([f for f in os.listdir(image_folder) if f.endswith('.png')],
                        key=lambda x: int(x.split('_')[0]))

    jump_table = []
    font_data = []

    max_width = 0
    max_height = 0
    offset = 0
```

```
    for char in characters:
        image_path = os.path.join(image_folder, char)
        bitmap, width, height = convert_image_to_bitmap(image_path)
        binary_data = bitmap_to_binary_data(bitmap, width, height)
        x_advance = width

        max_width = max(max_width, width)
        max_height = max(max_height, height)

        jump_table.append((offset & 0xFF, (offset >> 8) & 0xFF, width, height, x_advance))
        font_data.append((char, binary_data))
        offset += len(binary_data)

    with open(font_file, 'w') as f:
        f.write('// Generated font file\n')
        f.write('const char Russian_font_16[] PROGMEM = {\n')
        f.write(f'\t0x{max_width:X}, // Width: {max_width}\n')
        f.write(f'\t0x{max_height:X}, // Height: {max_height}\n')
        f.write(f'\t0x20, // First Char: 32\n')
        f.write(f'\t0x{len(characters):X}, // Number of Chars: {len(characters)}\n\n')
        f.write('\t// Jump Table:\n')
        for i, entry in enumerate(jump_table):
            char_code = int(characters[i].split('_')[0])
            f.write(f'\t0x{entry[0]:02X}, 0x{entry[1]:02X}, 0x{entry[2]:02X}, 0x{entry[3]:02X}, // {chr(char_code)}\n')

        f.write('\n\t// Font Data:\n')
```

```
        for char, binary_data in font_data:
            char_code = int(char.split('_')[0])
            f.write('\t')
            for i, byte in enumerate(binary_data):
                f.write(f'0x{byte}, ')
                if (i + 1) % 8 == 0:
                    f.write('\n\t')
            f.write(f'// {chr(char_code)}\n')

        f.write('};\n')

    messagebox.showinfo("Успех", f"Файл фонта {font_file} успешно сгенерирован!")

# Функция для выбора изображения
def select_image():
    file_selected = filedialog.askopenfilename(filetypes=[("Image files", "*.png;*.jpg;*.jpeg")])
    if file_selected:
        image_path_label.config(text=f"Выбрано изображение: {file_selected}")
        root.image_path = file_selected
        extract_button.config(state="normal")

# Функция для выбора папки с изображениями символов
def select_image_folder():
    folder_selected = filedialog.askdirectory()
    if folder_selected:
        image_folder_label.config(text=f"Выбрана папка: {folder_selected}")
```

```
        root.image_folder = folder_selected
        generate_button.config(state="normal")
# Функция для извлечения символов и их сохранения
def extract_and_save():
    image_path = root.image_path
    extract_and_save_characters(image_path)
    image_folder_label.config(text="Символы сохранены в папку 'character'")

# Функция для генерации файла фонта
def generate_font():
    image_folder = 'character'
    output_folder = filedialog.askdirectory()
    if not output_folder:
        return
    font_file = os.path.join(output_folder, "russian_font.h")
    generate_font_file(image_folder, output_folder, font_file)

# Создание основного окна
root = Tk()
root.title("Генератор фонтов")

# Создание и размещение виджетов
image_path_label = Label(root, text="Изображение не выбрано")
image_path_label.pack(pady=10)

select_image_button = Button(root, text="Выбрать изображение",
command=select_image)
select_image_button.pack(pady=10)
```

```
extract_button = Button(root, text="Извлечь символы", state="disabled", command=extract_and_save)
extract_button.pack(pady=10)

image_folder_label = Label(root, text="Папка с символами не выбрана")
image_folder_label.pack(pady=10)

select_folder_button = Button(root, text="Выбрать папку с символами", command=select_image_folder)
select_folder_button.pack(pady=10)

generate_button = Button(root, text="Сгенерировать файл фонта", state="disabled", command=generate_font)
generate_button.pack(pady=10)

# Запуск основного цикла Tkinter
root.mainloop()
```