УТВЕРЖДЕН

RU.17701729.05.01-01 81 01-1-ЛУ

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

ПРОГРАММА ВИЗУАЛЬНОЙ ИНТЕРПРЕТАЦИИ ЭМОЦИЙ ПОЛЬЗОВАТЕЛЯ ДЛЯ ПОСТРОЕНИЯ 2D-АВАТАРА

Пояснительная записка

RU.17701729.05.01-01 81 01-1

Листов 33

Москва 2023

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Наименование программы

Наименование программы – «Программа визуальной интерпретации эмоций пользователя для построения 2D-аватара».

Наименование программы на английском языке – «Program for Visual Interpretation of User's Emotions on a 2D-Avatar».

1.2. Краткая характеристика области применения

Разработка ведется на основании приказа Национального исследовательского университета "Высшая школа экономики" №2.3-02/0812-01 от 08.12.2016.

Наименование темы разработки – «Программа визуальной интерпретации эмоций пользователя для построения 2D-аватара». Выполнение производится в рамках курсовой работы в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия».

1.3. Стандарты документации

Документ был написан в соответствии с ГОСТ и ЕСПД [1-9].

2. НАЗНАЧЕНИЕ ПРОГРАММЫ И ОБЛАСТЬ ЕЕ ПРИМЕНЕНИЯ

2.1. Назначение

Программа предназначена для создания двумерных маскотов, являющихся визуализацией эмоций пользователя в реальном времени, используя спрайты. Программа предоставляет пользователю функционал создания, удаления и редактирования маскотов (проектов), настройку используемых эмоций для выбранного маскота, настройку используемых эмоциями спрайтов и расположения выбранных спрайтов относительно маскота.

2.2. Область применения

Программа разработана для создателей развлекательного видео контента, использующих или желающих использовать в своих видео аватар для репрезентации своего лица, однако не желающих обращаться к решениям, задействующим дорогие и сложные в эксплуатации трехмерные модели. Программа предоставляет пользователю возможность быстро создать аватар (маскот) под свои персональные нужды, используя двумерные спрайты.

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1. Постановка задачи

Разработать приложение для создания двумерных маскотов, повторяющих эмоции пользователя. Программа должна иметь функционал создания, редактирования и удаления маскотов (проектов), а также сохранения и загрузки с персонального компьютера пользователя, на котором установлено приложение. Пользователь должен иметь возможность открыть проект, после чего приложение должно предоставить пользователю возможность выбрать используемую камеру, микрофон, цвет фона окна с маскотом. Также должна присутствовать визуализация созданных пользователем эмоций и используемых ими спрайтов. Пользователь может создать новую эмоцию с указанием имени и типа эмоции, добавить к выбранной эмоции новый спрайт с указанием имени, пути до спрайта на компьютере и назначения спрайта (рот, глаза, лицо и т.п.). При выборе эмоции пользователем, приложение должно визуализировать на выделенной области все спрайты, относящиеся к выбранной эмоции, поверх друг друга. Пользователь должен иметь возможность настраивать отношение между спрайтами, позволяя выводить выбранный спрайт выше или ниже других.

Постановление задачи контролируется приказом Национального исследовательского университета “Высшая школа экономики” №2.3-02/0812-01 от 08.12.2016. Разработка программы происходит в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 “Программная инженерия” в рамках курсовой работы.

3.2. Описание алгоритма и функционирования программы

Фреймворк Tauri [9] подразумевает под собой совместную работу Rust и TypeScript. Данная часть проекта использует Rust для работы с периферией (камерами и микрофонами) ПК пользователя, а также по входным данным строит параметры, использующиеся TypeScript-частью для отрисовки маскота. Tauri Api [9] используется для работы с файловой системой персонального компьютера. Также Rust-часть используется для создания окна приложения и предоставления необходимых методов согласно пункту 4.1 из настоящего Технического Задания, которые, по ходу работы программы, вызываются при взаимодействии пользователя с интерфейсом приложения. В данном документе описывается часть приложения, использующая Rust.

Работу программы можно разделить на следующие этапы:

- Инициализация приложения.
- Инициализация логгера.
- Импорт Конфигураций.
- Подключение к дефолтной (первой по списку) камере.
- Подключение к дефолтному (первому по списку) микрофону.
- Инициализация методов, доступных с фронтенда.
- Инициализация глобального стейта приложения .
- Построение окна приложения.
- Получение списка доступных камер.
- Получение списка доступных микрофонов.
- Выбор микрофона для ввода.
- Выбор камеры для ввода.
- Получение громкости с микрофона.
- Получение изображения с камеры.
- Построение параметров маскота по информации, полученной с микрофона и камеры.
    - По выводу ML с изображения с камеры.
    - По громкости с микрофона.
- Ответы на запросы фронтенда

Процесс получения параметров маскота выглядит следующим образом:

- Делается запрос к камере.
- Полученный буффер декодируем в RGB.
- Сохраняем картинку в формате «.jpg».
- Открываем картинку и обрезаем ее до центрального квадрата.
- Сохраняем картинку в формате «.jpg».
- Открываем картинку в torch и преобразовываем ее в тензор.
- Цветной тензор преобразовываем в черно-белый, посредством усреднения значения трех каналов.
- Увеличиваем размерность тензора.
- Применяем модель.

- Уменьшаем размерность тензора.
- Преобразуем в одномерный массив.
- По данным числам выбираем наибольшее.
- Индекс наибольшего числа декодируем в эмоцию.
- Получаем массив данных с микрофона и преобразовываем посредством сложения абсолютных величин в единственное число - громкость.
- Состояние глаз (открыты / закрыты) определяем случайно.
- Всю полученную информацию передаем фронтенду.

3.3. Описание и обоснование способов предоставления входных и выходных данных

Входными данными являются:

- Информация (массив из чисел с плавающей точкой, который далее преобразовывается в единственное целое число - громкость), полученная с микрофона.
- Информация (изображение), полученное с камеры.
- Информация, переданная в запросе с фронтенда.
    - Индекс нужных камеры/микрофона.
    - Конфигурация маскота в виде строки с ссылками на сорсы (изображения).
    - Конфигурация камеры (структура из 3 полей u32).

Выходными данными являются:

- Ответы на запросы TS-части.
    - Типы ответов подразделяются на.
        - () - пустой ответ.
        - String - ответ в виде строки.
        - Result<(), String>.
            - При успешном выполнении - пусто ответ.
            - При ошибке - её текст.
        - Result<Vec<String>, String>.
            - При успешном выполнении - массив строк.
            - При ошибке - её текст.
        - Result<Mascot, String>.
            - При успешном выполнении - конфигурация маскота.
            - При ошибке - её текст.

- u8.
- Вывод логгера в стандартный поток вывода (в dev-версии).
- Обработанное изображение в каталоге MASCOTY в документах пользователя.

3.4. Выбор состава технических средств и его обоснование

Для установки программы на микроконтроллер требуется персональный компьютер со следующими системными характеристиками:

- Установленная ОС Windows 10 или выше / MacOS 13.2.1 или выше / Ubuntu 20.04 или выше.
- Процессор с тактовой частотой 2.4 ГГЦ или выше, имеющий 8 ядер.
- Клавиатура, мышь, монитор с разрешением не менее 1200 на 900 пикселей.
- 16 ГБ общей оперативной памяти.
- 8 ГБ свободной памяти на жестком диске.

Обоснование:

- Необходимо для правильной работы всех интегрированных в приложение библиотек.
- Необходимо для корректной работы графического интерфейса и своевременной обработки взаимодействий с пользователем.
- Необходимы для взаимодействия с графическим интерфейсом.
- Необходимо для корректной работы графического интерфейса и своевременной обработки взаимодействий с пользователем.
- Необходимо для корректной работы графического интерфейса и своевременной обработки взаимодействий с пользователем.

3.5. Выбор программных средств и языков программирования и его обоснование

Основным фреймворком при создании приложения был выбран Tauri [9], так как он позволяет удобно создавать кроссплатформенные приложения с комплексным пользовательским интерфейсом, при этом не требуя такого же количества ресурсов компьютера, как его основной конкурент Electron, благодаря легковесному Rust на бэкенде. Графический интерфейс Tauri [9] использует TS. Вне графического интерфейса приложение использует Rust. В проекте вместо TS используется TypeScript в следствии строгой типизации, позволяющей разработчику лучше следить за типами объектов, что в долгосрочной перспективе приводит к уменьшению ошибок во время исполнения программы и облегчает разработку.

Были использованы следующие сторонние библиотеки:

- tauri-build [9] - билд-зависимость, помогающая собирать Tauri-приложения [9].
- serde - библиотека сериализации/десереализации.
- serde_json - библиотека сериализации/десереализации JSON.
- serde_yaml - библиотека сериализации/десереализации YAML.
- log - библиотека для логгирования.
- simple_logger - имплементация предыдущей библиотеки.
- rand - библиотека для получения рандомных данных.
- tch [16] - биндинги C++ Torch [17].
- cpal [14] - библиотека для низкоуровнего взаимодействия с аудио.
- tauri [9] - фреймворк для создания кроссплатформенных десктоп-приложений.
- tokio [15] - библиотека асинхронного рагтайма Rust.
- nokhwa [13] - библиотека для взаимодействия с камерами.
- image - библиотека для работы с изображениями.

3.6. Выбор способов и алгоритмов решения, их подробное описание и обоснование

Далее представлено подробное описание каждого пункта из 3.2:

- Инициализация приложения.

«main.rs» создает окно приложения через Tauri API [9]. Tauri [9] создает WebView, контролируемый «main.tsx», который, в свою очередь создает корневой элемент DOM’а в виде компонента App.

- Инициализация логгера.

При старте приложения создаётся SimpleLogger с уровнем фильтрации “Debug” (то есть, выводится будут все сообщения, кроме “Trace”).

- Импорт Конфигураций.

Далее в comile-time импортируется содержимое файлов src-tauri/src/config/config.yaml и src-tauri/src/config/model.pt, после чего преобразовываются в CameraConfig и CModule соответственно.

- Подключение к дефолтной (первой по списку) камере.

Пробное подключение к дефолтной (первой по списку) камере системы с дефолтным конфигом и последующим открытием видеопотока.

- Подключение к дефолтному (первому по списку) микрофону.

Пробное подключение к дефолтному (первому по списку) микрофону системы и последующим открытием аудиопотока.

- Инициализация методов, доступных с фронтенда.

Перед запуском приложения с помощью декларативных и процедурных макросов определяются методы, доступные для JS-части.

- Инициализация глобального стейта приложения.

Перед запуском приложения определяются переменные, хранящие глобальный Стейт приложения.

- Построение окна приложения.

Происходит сразу после всех предыдущих пунктов. После запуска метода run приложение начинает слушать запросы от TS-части.

- Получение списка доступных камер.

С помощью библиотеки Nokhwa [13] определяется список камер, которые далее приводятся к человеку-читаемому виду и отправляются на фронтенд. Если камер не нашлось, фронтону отправляется сообщение об ошибке.

- Получение списка доступных микрофонов.

С помощью библиотеки CPAL [14] определяется список аудио-устройств, которые фильтруются по количеству каналов ввода (должно быть более 0) и приводятся к человеку-читаемому виду, далее отправляются на фронтенд. Если микрофонов не нашлось, фронтону отправляется сообщение об ошибке.

- Выбор микрофона для ввода.

По входящим параметрам фронтенда определяется искомая камера, далее происходит подключение к ней. При возникающих ошибках фронтону возвращается информация о них.

- Выбор камеры для ввода.

По входящим параметрам фронтенда определяется искомый микрофон, далее происходит подключение к нему. При возникающих ошибках фронтону возвращается информация о них.

- Получение громкости с микрофона.

С микрофона приходит массив чисел с плавающей точкой, далее нормализуется по шкале от 1 до 100. Итоговое число отправляется фронтенду.

- Получение изображения с камеры.

Приходящее с камеры изображение декодируется в RGB, после чего обрезается до квадрата и сохраняется в формате “.jpg”.

- Построение параметров маскота по информации, полученной с микрофона и камеры.
    - По выводу ML с изображения с камеры.

      Входящее цветное изображение преобразуется в чёрно-белое, после чего передаётся в модель, на основе вывода которой определяется эмоция.
    - По громкости с микрофона.

      Фронтенду отправляются громкость с типом u8 и состояние рта маскота с типом bool, которое зависит от самой громкости.
- Ответы на запросы фронтенда.

Все виды запросов описаны в файле src-tauri/src/commands.rs.

Существует 3 вида запросов:

- Без ответа: в них гарантировано успешно меняется стейт приложения.
- С пустым ответом или ошибкой: если при изменении стейта происходит ошибка - её текст возвращается.
- С ответом: гарантированно безопасное получение какой-либо информации с бэкенда.
- С ответом или ошибкой: если при получении информации произошла ошибка - её текст возвращается.

4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1. Предположительная потребность

Данная программа ориентируется на людей, не делающих разбираться с сложных аналогах, использующих комплексные трехмерные модели или не готовых создавать и/или покупать трехмерные модели, но при этом желающих получить возможность автоматического изменения аватара в зависимости от экспрессии на лице пользователя.

4.2. Преимущества разработки в сравнении с аналогами (отечественными и зарубежными)

Аналоги можно разделить на 2 группы:

1 Программы, использующие захват лица пользователя через камеру для переноса экспрессий на лицо трехмерного аватара.

2 Программы, использующие двумерных аватаров, состоящих из спрайтов.

Преимуществом над первой группой является использование двумерных аватаров, что позволяет упростить создание проекта, так как при создании двумерного аватара не требуются глубокие знания в разработке моделей внутри специализированных программ для 3D.

5. ИСТОЧНИКИ

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
4. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
5. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001 ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом – М.: ИПК Издательство стандартов, 2001
9. Документация Tauri [Электронный ресурс]. URL: https://tauri.app/v1/guides/ (дата обращения: 02.10.2022)
10. Документация Tauri. Создание хэндлеров [Электронный ресурс]. URL: https://tauri.app/v1/guides/features/command/ (дата обращения: 10.10.2022)
11. Документация Tauri. Continuous Integration [Электронный ресурс]. URL: https://tauri.app/v1/guides/testing/webdriver/ci/ (дата обращения: 22.10.2022)
12. Документация Tauri. Создание хэндлеров [Электронный ресурс]. URL: https://tauri.app/v1/guides/features/command/ (дата обращения: 22.10.2022)
13. Документация Nokhwa [Электронный ресурс]. URL: https://github.com/l1npengtul/nokhwa/blob/senpai/README.md (дата обращения: 24.10.2022)

14. Документация CPAL [Электронный ресурс]. URL: https://github.com/RustAudio/cpal/blob/master/README.md (дата обращения: 26.12.2022)
15. Документация Tokio [Электронный ресурс]. URL: https://tokio.rs/tokio/tutorial (дата обращения: 02.02.2023)
16. Документация tch [Электронный ресурс]. URL: https://github.com/LaurentMazare/tch-rs/blob/main/README.md (дата обращения: 08.02.2023)
17. Документация Torch [Электронный ресурс]. URL: https://pytorch.org/docs/stable/torch.html (дата обращения: 08.02.2023)
18. Документация GitHub Actions [Электронный ресурс]. URL: https://docs.github.com/en/actions (дата обращения: 02.04.2023)

ПРИЛОЖЕНИЕ 1

ОПИСАНИЕ И НАЗНАЧЕНИЕ КЛАССОВ

ФАЙЛЫ .rs (Rust), используемые приложением:

Файлы проекта можно разделить на следующие категории:

- Файл с точкой входа.
    - src-tauri/src/main.rs.
- Файлы с определением локальных и внешних модулей (зависимостей).
    - src-tauri/src/lib.rs.
- Скриптовые файлы.
    - src-tauri/build.rs.
- Файлы с тестами.
    - src-tauri/src/tests.rs.
    - src-tauri/src/config/tests.rs.
- Файлы с API.
    - src-tauri/src/commands.rs.
- Файлы с утилитами.
    - src-tauri/src/utils.rs.
- Файлы с моделями данных и описанием их поведения.
    - src-tauri/src/config/mod.rs.
    - src-tauri/src/devices.rs.
    - src-tauri/src/emotions.rs.
    - src-tauri/src/mascot.rs.

Файлы с точкой входа.

Файл	Назначение

Файл	Назначение
src-tauri/src/main.rs	Файл является точкой входа в приложение, импортирует стандартную конфигурацию камеры, создаёт логгер, а также получает доступ к стандартным камере/микрофону системы и инициализирует приложение

Файлы с определением локальных и внешних модулей (зависимостей).

Файл	Назначение
src-tauri/src/lib.rs	Файл определяет список локальных и внешних модулей (зависимостей)

Скриптовые файлы.

Файл	Назначение
src-tauri/build.rs	Файл содержит скрипт билда приложения

Файлы с тестами.

Файл	Назначение
src-tauri/src/config/tests.rs	Файл содержит тесты, относящиеся к модулю конфига
src-tauri/src/tests.rs	Файл содержит тесты утилит и функций, относящихся к ML

Файлы с API.

Файл	Назначение
src-tauri/src/commands.rs	Файл содержит ручки, доступные фронтенду

Файлы с утилитами.

Файл	Назначение
src-tauri/src/utils.rs	Файл содержит макросы, упрощающие написание кода

Файлы с моделями данных и описанием их поведения.

Файл	Назначение
src-tauri/src/config/mod.rs	Файл содержит сущности, задающие конфигурации проекта такие как конфигурация камеры и использующаяся модель, а также методы и интерфейсы для удобного взаимодействия с ними. Также файл является главным файлом модуля src-tauri/src/config, то есть, в нём перечислены методы и объекты, доступные остальной части проекта

Файл	Назначение
src-tauri/src/devices.rs	Файл содержит сущности, задающие используемые в данный момент девайсы для ввода, такие как камера и микрофон, а также методы и интерфейсы для удобного взаимодействия с ними
src-tauri/src/emotions.rs	Файл содержит сущность, задающую список эмоций, а также методы и интерфейсы для удобного взаимодействия с ней
src-tauri/src/mascot.rs	Файл содержит сущность, задающую параметры маскота, а также методы и интерфейсы для удобного взаимодействия с ней

ПРИЛОЖЕНИЕ 2

НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ, СВОЙСТВ, СОБЫТИЙ ФАЙЛОВ ПРОЕКТА

src-tauri/src/config/mod.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение
camera	Поле	CameraConfig	-	Хранит конфигурацию камеры
model	Поле	CModule	-	Хранит импортированную модель
fmt	Функция	fmt::Result	&Config &mut Formatter<‘_>	Имплементирует функцию для вывода информации о структуре при дебаге
height	Поле	u32	-	Высота разрешения камеры в пикселях
width	Поле	u32	-	Ширина разрешения камеры в пикселях

Название	Тип члена	Тип значения	Входные параметры	Назначение
fps	Поле	u32	-	Количество кадров в секунду
import_config	Функция	Config	-	Возвращает конфигурацию проекта

src-tauri/src/config/tests.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение
check_config_import	Функция	-	-	Тестирование сущностей и функций модуля конфигурации

src-tauri/src/commands.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение
get_raw_mascot	Функция	String	tauri::State<Mutex<String>>	Получение строки с сорсами текущего маскота

Название	Тип члена	Тип значения	Входные параметры	Назначение
set_raw_mascot	Функция	-	tauri::State<Mutex<String>>	Обновление строки с сорсами текущего маскота
get_mascot	Функция	Result<mascot::Mascot, String>	tauri::State<Devices>	Получение параметров маскота или сообщения об ошибке
get_cameras	Функция	Result<Vec<String>, String>	-	Возвращает список названий камер или сообщения об ошибке
select_camera	Функция	Result<(), String>	usize CameraConfig tauri::State<Devices>	Выбор новой активной камеры

Название	Тип члена	Тип значения	Входные параметры	Назначение
is_in_interval	Функция	bool	i32 i32 i32	Проверяет, что одно число находится между двух других. Функция удобна для проверки индексации
set_camera_config	Функция	Result<(), String>	CameraConfig tauri::State<Devices>	Обновляет конфигурацию текущей камеры
get_microphones	Функция	Result<Vec<String>, String>	tauri::State<Host>	Возвращает список названий микрофонов или сообщения об ошибке
select_microphone	Функция	Result<(), String>	usize tauri::State<Host> tauri::State<Devices>	Выбор нового активного микрофона

Название	Тип члена	Тип значения	Входные параметры	Назначение
get_volume	Функция	u8	tauri::State<Devices>	Получает громкость микрофона в границах от 0 до 100

src-tauri/src/devices.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение
receiver	Поле	Arc<Mutex<Vec<f32>>>	-	Содержит данные, вводящиеся с микрофона
_stream	Поле	Stream	-	Callback, получающий данные с микрофона и записывающий в receiver
camera	Поле	Mutex<Camera>	-	Активная камера
config	Поле	Mutex<Config>	-	Конфигурация камеры и модели

Название	Тип члена	Тип значения	Входные параметры	Назначение
microphone	Поле	Mutex<Microphone>	-	Активный микрофон
Sync	Имплементация интерфейса	-	-	Трейт, позволяющий безопасно читать и писать данные по ссылке из нескольких потоков
Send	Имплементация интерфейса	-	-	Трейт, позволяющий безопасно передавать данные между потоками
new	Функция	Devices	Config Camera Microphone	Создаёт структуру девайсов

Название	Тип члена	Тип значения	Входные параметры	Назначение
set_up_camera	Функция	Result<(), String>	&Devices &CameraConfig Camera	Инициализирует новую камеру с данной конфигурацией. Если происходит ошибка - возвращает её текст
set_camera_settings	Функция	Result<(), NokhwaError>	&Devices &CameraConfig	Обновляет конфигурацию камеры. При ошибке возвращает её (ошибку)
get_camera_index	Функция	CameraIndex	&Devices	Возвращает индекс текущей камеры
update_microphone	Функция	Result<(), String>	&Devices Microphone	Меняет активный микрофон
get_volume	Функция	u8	&Devices	Возвращает громкость входных данных в микрофон

Название	Тип члена	Тип значения	Входные параметры	Назначение
get_mikes	Функция	Result<Vec<(usize, Device, SupportedStreamConfig)>, DevicesError>	&Host	Возвращает список микрофонов пользователя или ошибку
set_mike	Функция	Result<Microphone, DevicesError>	usize &Host	Возвращает выбранный микрофон или ошибку
get_cams	Функция	Result<Vec<CameraInfo>, NokhwaError>	-	Возвращает список камер пользователя или ошибку
set_cam	Функция	Result<Camera, NokhwaError>	CameraIndex &CameraConfig	Создаёт объект камеры по индексу и конфигурации

src-tauri/src/emotions.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение

Название	Тип члена	Тип значения	Входные параметры	Назначение
Emotion	Перечисление	Angry Disgust Fear Happy Neutral Sad Surprise	-	Отображает все доступные эмоции
from_num	Функция	Emotion	u8	Преобразует число в эмоцию
to_num	Функция	u8	Emotion	Преобразует эмоцию в число
fmt	Функция	Result	&Emotion &mut Formatter	Имплементирует функцию для вывода информации о структуре при дебаге
fmt	Функция	Result	&Emotion &mut Formatter	Позволяет кастить эмоцию к строке

src-tauri/src/lib.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение

Название	Тип члена	Тип значения	Входные параметры	Назначение
-	-	-	-	-

src-tauri/src/main.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение
main	Функция	-	-	Точка входа в программу

src-tauri/src/mascot.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение
emotion	Поле	Emotion	-	Содержит эмоцию маскота
blink	Поле	bool	-	Содержит состояние глаз маскота
lips	Поле	bool	-	Содержит состояние губ маскота
to_bw	Функция	Tensor	Tensor	Приводит тензор цветного изображения к чёрно-белому

Название	Тип члена	Тип значения	Входные параметры	Назначение
argmax	Функция	u8	&[f64]	Возвращает индекс самого большого элемента в срезе
get_cropped_corners	Функция	(u32, u32, u32, u32)	u32 u32	Возвращает координаты углов обрезанного изображения
crop_image	Функция	-	&str	Обрезает картинку и сохраняет по указанному пути
get_emotion	Функция	Emotion	&Devices &str	Получает эмоцию по фотографии
get_mascot	Функция	Result<Mascot, NokhwaError>	&Devices	Получает конфигурацию маскота по снимку с камеры

src-tauri/src/tests.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение

Название	Тип члена	Тип значения	Входные параметры	Назначение
check_panic_error_ok	Функция	-	-	Тест макроса panic_error с удачным исходом
check_panic_error_err	Функция	-	-	Тест макроса panic_error с неудачным исходом
check_check_error	Функция	-	-	Тест макроса check_error
check_init_dict	Функция	-	-	Тест макроса init_dict
check_to_bw	Функция	-	-	Тест функции to_bw
check_argmax	Функция	-	-	Тест функции argmax

src-tauri/src/utils.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение

Название	Тип члена	Тип значения	Входные параметры	Назначение
check_error	Декларативный макрос	$expr	$expr , $expr $(,)?	Безопасное раскрытие объекта и последующий вывод в дебаг информации о значении или об ошибке
panic_error	Декларативный макрос	$expr	$expr , $expr $(,)?	Опасное раскрытие объекта и последующий вывод в дебаг информации о значении или об ошибке
init_dict	Декларативный макрос	HashMap::<_, _>	$($key:expr => $val:expr) , * $(,)?	Создание словаря по входным данным в виде набора токенов кода

src-tauri/build.rs.

Название	Тип члена	Тип значения	Входные параметры	Назначение
main	Функция	-	-	Точка входа в скрипт билда приложения

ПРИЛОЖЕНИЕ 3

ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ

- TS (TypeScript) - язык программирования, являющийся расширением для JavaScript.
- Rust - мультипарадигменный компилируемый язык программирования низкого уровня, сочетающий парадигмы функционального и процедурного программирования.
- JSON (JavaScript Object Notation) - популярный текстовый формат представления данных.
- YAML (Yet Another Markup Language) - популярный текстовый формат представления данных.
- Фреймворк - крупная библиотека, решающая ряд задач.
- Билд - сборка приложения.
- Спрайт – объект, характеризующийся изображением, размером и положением относительно маскота.
- Маскот – совокупность спрайтов, создающая аватара пользователя.
- Фронтенд - сервис для отображения интерфейса и считывания действий пользователя.
- Бэкенд - сервис для обработки запросов фронтенда, в данном проекте используется для работы с тяжёлыми вычислениями.
- API (Application Programming Interface) - интерефейс для общения сервисов, в данном случае фронтенда и бэкенда.
- Макрос - приём метапрограммирования, при котором на вход подаются токены кода, на выход они же.
- Декларативный макрос - макрос в виде функции.
- Процедурный Маркос - аналог аннотации Java.
- Сериализация - процесс преобразования объекта в некоторый формат представления, чаще всего JSON.
- Десериализация - процесс преобразования текста в некотором формате представления, чаще всего JSON, в объект.
- Биндинг - привязка функции или структуры на одном языке программирования к соответствующей функции или структуре на другом языке программирования.
- RGB (Red Green Blue) – цветовая модель, позволяющая представлять любые цвета с помощью основных (красного,

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. И дата	Взам. Инв. №	Инв. № дубл.	Подп. И дата