ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

СОГЛАСОВАНО

Доцент департамента программной инженерии факультета компьютерных наук, кандидат технических наук

_____ О. В. Максименкова

«_7_» _____апреля_____ 2023 г.

УТВЕРЖДАЮ

Академический руководитель образовательной программы «Программная инженерия», профессор департамента программной инженерии, кандидат технических наук

_____ В. В. Шилов

«_7_» _____апреля_____ 2023 г.

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

ПРОГРАММА ВИЗУАЛЬНОЙ ИНТЕРПРЕТАЦИИ ЭМОЦИЙ ПОЛЬЗОВАТЕЛЯ ДЛЯ ПОСТРОЕНИЯ 2D-АВАТАРА

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.05.01-01 81 01-1

Исполнитель студент группы БПИ 204

_____ /М. К. Аленов/

«_03_»____апреля_____ 2023 г.

Москва 2023

УТВЕРЖДЕН
RU. 17701729.05.01-01 81 01-1

ПРОГРАММА ВИЗУАЛЬНОЙ ИНТЕРПРЕТАЦИИ ЭМОЦИЙ ПОЛЬЗОВАТЕЛЯ ДЛЯ ПОСТРОЕНИЯ 2D-АВАТАРА

Пояснительная записка

RU.17701729.05.01-01 81 01-1

Листов 48

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

Москва 2023

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ....4
1.1 Наименование программы....4
1.2 Основания для разработки....4
1.3 Стандарты документации....5
2. НАЗНАЧЕНИЕ ПРОГРАММЫ И ОБЛАСТЬ ЕЕ ПРИМЕНЕНИЯ....6
2.1 Назначение....6
2.2 Область применения....6
3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ....7
3.1 Постановка задачи....7
3.2 Описание алгоритма и функционирования программы....7
3.3 Описание и обоснование способов предоставления входных и выходных данных....9
3.4 Выбор состава технических средств и его обоснование....10
3.5 Выбор программных средств и языков программирования и его обоснование....10
3.6 Выбор способов и алгоритмов решения, их подробное описание и обоснование....11
4 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ....16
4.1 Предположительная потребность....16
4.2 Преимущества разработки в сравнении с аналогами (отечественными и зарубежными)..16
5 ИСТОЧНИКИ....17
ПРИЛОЖЕНИЕ 1 ОПИСАНИЕ И НАЗНАЧЕНИЕ КЛАССОВ....19
ФАЙЛЫ TYPESCRIPT, ИСПОЛЬЗУЕМЫЕ ПРИЛОЖЕНИЕМ....19
ГЛОБАЛЬНЫЕ КОМПОНЕНТЫ....19
МОДАЛЬНЫЕ КОМПОНЕНТЫ....20
КОМПОНЕНТЫ, СВЯЗАННЫЕ С ЭМОЦИЯМИ....21
КОМПОНЕНТЫ, СВЯЗАННЫЕ СО СПРАЙТАМИ....21
КОМПОНЕНТЫ, СВЯЗАННЫЕ С ВИЗУАЛИЗАЦИЕЙ МАСКОТА....21
ИНТЕРФЕЙСЫ И ПЕРЕЧИСЛЕНИЯ....23
ДОПОЛНИТЕЛЬНЫЕ УТИЛИТЫ....23
КОМПОНЕНТЫ, СВЯЗАННЫЕ С НАСТРОЙКАМИ....24
ПРИЛОЖЕНИЕ 2 НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ, СВОЙСТВ, СОБЫТИЙ ФАЙЛОВ ПРОЕКТА....25
Файл EmotionPart.tsx:....25
Файл EmotionsSelection.tsx:....25
Файл IConf.ts:....26
Файл IEmotion.ts:....27
Файл IMascot.ts:....27
Файл IMascotData:....28

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

Файл IPart.ts:....28
Файл MascotCanvas.tsx:....29
Файл MascotPart.tsx:....30
Файл ShadowCanvas.tsx:....32
Файл ShadowPart.tsx:....32
Файл EmotionAdd.tsx:....33
Файл PartAdd.tsx:....34
Файл Proceed.tsx:....35
Файл ProjectAdd.tsx:....35
Файл ProjectAddScrap.tsx:....36
Файл PartPart.tsx:....37
Файл PartsSelection.tsx:....37
Файл BackgroundColorPicker.tsx:....38
Файл MicMinMaxDisplay.tsx:....39
Файл Slider.tsx:....39
Файл TwoSlider.tsx:....40
Файл Commands.ts:....41
Файл DummyMascot.ts:....42
Файл EDescriptor.ts:....42
Файл Save.ts:....43
Файл UUIDGen.ts:....43
Файл App.tsx:....43
Файл Projects.tsx:....44

ПРИЛОЖЕНИЕ 3 ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ....48

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ....49

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 Основания для разработки

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

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].

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

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.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1 Постановка задачи

Разработать приложение для создания двумерных маскотов, повторяющих эмоции пользователя в реальном времени. 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 предложить 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 [10] подразумевает под собой совместную работу Rust и JavaScript. Данная часть проекта использует JavaScript для создания интерфейса, обработки интеракций с пользователем, Tauri Api [11] для работы с файловой системой персонального компьютера, на котором установлено приложение, Rust для создания окна приложения и предоставления необходимых методов для корректной реализации пунктов 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 TypeScript и Tauri Api [11].

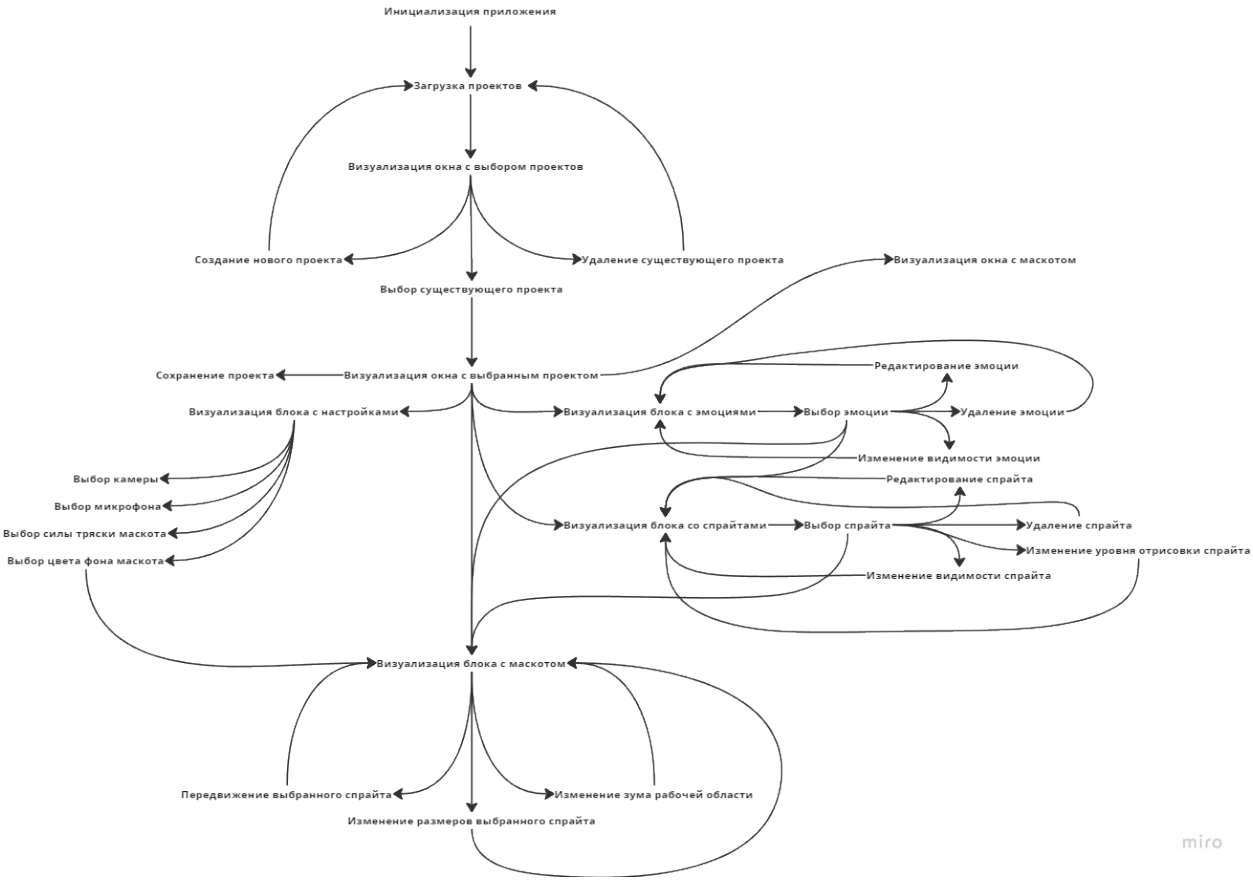
Работу программы можно разделить на следующие этапы:

1. Инициализация приложения
2. Загрузка проектов
3. Создание проекта
4. Выбор проекта
5. Сохранение проекта

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

6. Выбор камеры
7. Выбор микрофона
8. Выбор цвета фона маскота
9. Выбор силы тряски маскота
10. Выбор эмоции
11. Выбор спрайта
12. Добавление эмоции
13. Добавление спрайта
14. Удаление эмоции
15. Удаление спрайта
16. Редактирование эмоции
17. Редактирование спрайта
18. Изменение видимости эмоции
19. Изменение видимости спрайта
20. Изменение уровня отрисовки спрайта
21. Обработка взаимодействий с пользователем в блоке с маскотом
22. Визуализация финальной версии маскота

Вышеописанные этапы могут быть представлены в виде схемы, описывающей их взаимодействие друг с другом:



3.3 Описание и обоснование способов предоставления входных и выходных данных

Входными данными для программы являются ссылки на изображения (спрайты) на персональном компьютере пользователя и ссылки на директории (для сохранения проекта). Все представляют из себя пути в файловой системе, используемые приложением для доступа к их содержимому. Также под входными данными подразумеваются интеракции пользователя с графическим интерфейсом. Во время генерации финальной версии маскота, входными данными приложения являются пакеты данных из Rust, получаемые с использованием Tauri Api [11], состоящие из следующих полей: название эмоции, состояние рта (открыт / закрыт), состояние глаз (открыты / закрыты) и уровень громкости микрофона.

Выходными данными для программы является графический интерфейс приложения, а также следующий перечень файлов: создаваемый при первом запуске конфигурационный файл, сохраняющий все проекты пользователя, директории проектов, состоящих из конфигурационного файла проекта, файлов-копий изображений

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

пользователя, которые тот использует в качестве спрайтов для проекта, а также изображения, использующегося в качестве обложки проекта.

3.4 Выбор состава технических средств и его обоснование.

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

1. Установленная ОС Windows 10 или выше / MacOS 13.2.1 или выше / Ubuntu 20.04 или выше
2. Процессор с тактовой частотой 2.4 ГГЦ или выше, имеющий 8 или более ядер
3. Клавиатура, мышь, монитор с разрешением не менее 1200 на 900 пикселей.
4. 16 ГБ общей оперативной памяти.
5. 8 ГБ свободной памяти на жестком диске

Обоснование:

1. Необходимо для правильной работы всех интегрированных в приложение библиотек
2. Необходимо для корректной работы графического интерфейса и своевременной обработки взаимодействий с пользователем
3. Необходимы для взаимодействия с графическим интерфейсом
4. Необходимо для корректной работы графического интерфейса и своевременной обработки взаимодействий с пользователем
5. 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 [10], 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. Графический интерфейс Tauri [10] использует JavaScript. Вне графического интерфейса приложение использует Rust. В проекте вместо JavaScript 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:

1. mui/icons-material – Иконки, используемые приложением. [14]
2. mui/material – Библиотека с готовыми UI компонентами. [15]
3. tauri-app/api – API Tauri, обязательная для создания приложений на Tauri. [11]

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

4. html2canvas – Сохранение маскота как изображение для превью проекта. [16]
5. react – Библиотека создания графических интерфейсов. [13]
6. react-color – Предоставляет компонент, дающий пользователю возможность выбрать цвет из палитры. [17]
7. react-draggable – Предоставляет компонент, позволяющий реализовать перетягивание выбранного спрайта по области с маскотом. [18]
8. react-image-size – Библиотека, позволяющая получить размеры выбранного изображения в пикселях. [19]
9. react-toastify – Предоставляет компонент в виде всплывающего окна с информацией. [20]
10. set-interval-async – Библиотека, позволяющая устанавливать интервалы в асинхронном режиме. [21]

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

Далее представлено подробное описание каждого пункта из 3.2:

1. Инициализация приложения
   «main.rs» создает окно приложения через Tauri API [10]. Tauri [10] создает WebView, контролируемый «main.tsx», который, в свою очередь создает корневой элемент DOM’а в виде компонента App. При рендере App выводит в интерфейс компонент Projects. Он отвечает за визуализацию доступных проектов, создание новых проектов, удаление и редактирование существующих.
2. Загрузка проектов
   При выводе в DOM компонента Projects происходит обращение к файловой системе персонального компьютера в поиске директории «MASCOTY» в системном аналоге папки «Documents» (Документы). Если папка не найдена или внутри нет файла «conf.json», то приложение создает директорию MASCOTY с пустым файлом «conf.json».
   Внутри файла хранится массив объектов, характеризующих информацию о созданных пользователем проектах. Каждый объект состоит из названия проекта, пути до папки с проектом и пути до превью проекта, которое является изображением в формате «.png». После получения данных из «conf.json» приложение визуализирует пользователю список доступных проектов.
3. Создание проекта
   Приложение поддерживает три варианта создания проектов:
   1. Создание пустого проекта
      Пользователю выводится модальный компонент ProjectAdd, который запрашивает ввести название проекта и путь, где будет располагаться папка с проектом. После ввода необходимых данных, приложение создает папку с названием проекта в указанной директории и инициализирует

конфигурационный файл проекта внутри, содержащий пустой объект типа IMascot. Затем происходит добавление пути конфигурационного файла в описанный в П.3.6.1.2. «conf.json» и повторяется шаг П.3.6.1.2, что обновляет вывод доступных пользователю проектов в приложении.

2. Создание проекта из папки с изображениями
   Пользователю выводится модальный компонент ProjectAddScrap, который запрашивает ввести путь до папки (1), где будет создан проект, а также путь до директории (2), в которой находятся папки с изображениями. После ввода необходимых данных, приложение создает пустой объект IMascot,получает информацию о содержимом указанной директории (2) и проходится по подпапкам, каждая из которых становится эмоцией в IMascot. Если в подпапке присутствуют изображения в форматах «.jpg», «.jpeg», «.png», то они добавляются в эмоцию, соответствующую подпапке, в виде спрайтов. После прохода по всем файлам в директории (2), происходит инициализация директории проекта в папке (1) и создание конфигурационного файла проекта, содержащего ранее созданный объект IMascot. После чего, в директорию проекта сохраняются все изображения, используемые объектом IMascot с изменением формата на «.masset». Затем происходит добавление пути конфигурационного файла в описанный в П.3.6.1.2. «conf.json» и повторяется шаг П.3.6.1.2, что обновляет вывод доступных пользователю проектов в приложении.
3. Добавление существующего проекта
4. Пользователю выводится окно выбора папки с проектом. Если внутри выбранной директории есть файл с конфигурацией проекта, то происходит добавление пути конфигурационного файла в описанный в П.3.6.1.2. «conf.json» и повторяется шаг П.3.6.1.2, что обновляет вывод доступных пользователю проектов в приложении.

4. Выбор проекта
   Выбор проект внутри компонента Projects происходит одинарным нажатием на область, визуализирующую существующий проект. После чего пользователь может удалить проект или перейти к редактированию проекта. При удалении проекта, папка проекта удаляется с компьютера пользователя и происходит сохранение «conf.json» без удаленного проекта, после чего повторяется шаг П.3.6.1.2. При редактировании проекта, система проводит считывание информации из конфигурационного файл выбранного проекта и получает объект типа IMascot, который сохраняется в контексте системы, после чего происходит переход к

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

компоненту App, который выводит окно с редактированием загруженного проекта, что подразумевает под собой компоненты, содержащие: настройки маскота, список эмоций маскота, список спрайтов для выбранной эмоции, область, где визуализируется маскот относительно выбранной эмоции.

5. Сохранение проекта
   При сохранении проекта из контекста приложения достается объект IMascot, хранящий путь до папки с проектом. Приложение сравнивает файлы формата «.masset», находящиеся в папке проекта и «.masset», которые использует IMascot. Если IMascot перестал использовать некоторые файлы из директории проекта, то они удаляются из файловой системы. После удаления, происходит сохранение объекта IMascot в конфигурационный файл проекта и создается изображение в формате «.png», состоящее из видимых спрайтов на экране редактирования маскота, используемое в роли превью проекта в компоненте Projects.
6. Выбор камеры
   При переходе к компоненту App происходит обращение к Rust коду для получения списка камер. После выбора камеры пользователем, индекс выбранной камеры передается обратно в Rust
7. Выбор микрофона
   При переходе к компоненту App происходит обращение к Rust коду для получения списка микрофонов. После выбора микрофона пользователем, индекс выбранного микрофона передается обратно в Rust.
8. Выбор цвета фона маскота
   При выборе цвета фона маскота, расположенного внутри компонента App, происходит визуализация модального компонента SketchPicker - палитры, в которой пользователь может выбрать цвет. После совершенного выбора, цвет записывается в контекст, содержащий объект IMascot. После обновления контекста, цвет заднего фона области с визуализацией маскота меняется на выбранный пользователем.
9. Выбор силы тряски маскота
   При выборе силы тряски маскота, значение, выбранное пользователем, нормируется от 0 до 100 и записывается в соответствующее поле объекта IMascot внутри контекста приложения.
10. Выбор эмоции
    Выбор эмоции происходит внутри области со списком всех эмоций маскота. Выбор производится нажатием на эмоцию и записывает выбранную эмоцию в соответствующее поле объекта IMascot внутри контекста приложения. После обновления контекста область со списком спрайтов меняется в соответствии с выбранной пользователем эмоцией (визуализируются только те спрайты, которые относятся к этой эмоции).

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

11. Выбор спрайта
    Выбор спрайта происходит внутри области со списком всех спрайтов маскота для выбранной эмоции. Выбор производится нажатием на эмоцию и записывает выбранную эмоцию в соответствующее поле объекта IMascot внутри контекста приложения. После обновления контекста в области с визуализацией маскота, визуализация выбранного спрайта становится доступна для взаимодействий с пользователем.
12. Добавление эмоции
    Пользователю выводится модальный компонент EmotionAdd в режиме создания эмоции, требующий ввести имя эмоции и выбрать тип эмоции из списка (злость, грусть, испуг и т.п.). После ввода информации о новой эмоции, происходит добавление новой эмоции в объект IMascot и запись в контекст приложения.
13. Добавление спрайта
    Пользователю выводится модальный компонент PartAdd в режиме создания спрайта, требующий ввести имя спрайта, выбрать тип спрайта из списка (открытый рот, закрытый рот, лицо и т.п.) и указать путь до изображения в файловой системе компьютера. После ввода информации о новом спрайте, происходит копирование файла с изображением выбранного спрайта в директорию с проектом, где к имени копии добавляется уникальный идентификатор и изменяется расширение файла на «.masset». Затем происходит добавление нового спрайта к объекту IMascot к активной эмоции и запись объекта в контекст приложения.
14. Удаление эмоции
    Происходит удаление выбранной пользователем эмоции из объекта IMascot и запись в контекст приложения. При удалении эмоции удаляются все связанные с этой эмоцией спрайты (файлы с изображениями спрайтов при этом остаются в папке проекта до следующего сохранения проекта).
15. Удаление спрайта
    Происходит удаление выбранного пользователем спрайта из объекта IMascot и запись в контекст приложения. При удалении спрайта, файл с изображением относящемся к удаляемому спрайту не удаляется.
16. Редактирование эмоции
    Пользователю выводится модальный компонент EmotionAdd в режиме редактирования эмоции, где пользователю предлагается изменить имя или тип эмоции. После изменения данных эмоции, происходит редактирование эмоции в объекте IMascot и запись в контекст приложения.
17. Редактирование спрайта
    Пользователю выводится модальный компонент PartAdd в режиме редактирования эмоции, где пользователю предлагается изменить имя или тип

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

спрайта. После изменения данных спрайта, происходит редактирование спрайта в объекте IMascot и запись в контекст приложения.

18. Изменение видимости эмоции
    Изменение видимости эмоции влияет на визуализацию финальной версии маскота, где эмоции, отмеченные пользователем как невидимые, не будут использоваться при генерации маскота. При изменении видимости эмоции происходит обновление поля, отвечающего за видимость эмоции в соответствующем объекте внутри IMascot, после чего происходит сохранение объекта IMascot в контекст приложения.
19. Изменение видимости спрайта
    Изменение видимости спрайта влияет на визуализацию маскота в блоке с маскотом. Таким образом в данной области видны лишь те спрайты, которые пользователь оставил видимыми в блоке со списком спрайтов. При изменении видимости спрайта происходит обновление поля, отвечающего за видимость спрайта в соответствующем объекте внутри IMascot, после чего происходит сохранение объекта IMascot в контекст приложения.
20. Изменение уровня отрисовки спрайта
    В области с визуализацией маскота спрайты визуализируются в порядке иерархии, составленной в области со списком спрайтов. Пользователь может переместить спрайт выше или ниже в иерархии, что изменит уровень, на котором спрайт появляется в области с маскотом. При изменении уровня отрисовки, индекс выбранного спрайта в массиве спрайтов для выбранной эмоции меняется, после чего объект IMascot сохраняется в контекст приложения.
21. Обработка взаимодействий с пользователем в блоке с маскотом
    Внутри блока с маскотом пользователь видит наложенные друг на друга изображения спрайтов из блока спрайтов, соответственно их иерархии. Пользователь может единовременно взаимодействовать лишь с одним выбранным спрайтом. Если он хочет изменить положение или размер другого спрайта, то ему сначала нужно его выбрать блоке со списком спрайтов.
    При визуализации каждого спрайта внутри блока MascotCanvas используются компоненты MascotPart, которые могут иметь два разных варианта, зависящих от передаваемых внутрь компонента значений:
    1. Изображение спрайта
    2. Перемещаемое изображение спрайта с возможностью изменять размеры изображения.
       При передвижении и/или изменении размера изображения спрайта происходит запись новых координат и/или размеров в соответствующие поля внутри объекта IMascot в объект выбранного спрайта, находящийся в объекте выбранной эмоции

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

22. Визуализация финальной версии маскота
    Финальная версия маскота визуализируется через компонент ShadowCanvas, который дополняет функционал MascotCanvas тем, что при создании устанавливает таймер с интервалом, обращающийся к Rust коду для получения данных для визуализации маскота, состоящих из позиции (закрыт, открыт) глаз, рта, уровня громкости микрофона и текущей эмоции.

Все компоненты написаны в соответствии с руководством по программированию на языке TypeScript [12] и по работе с библиотекой React [13]

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 3Д.

Преимуществом над второй группой является использование камеры пользователя для получения информации о экспрессии на лице, что позволяет автоматизировать визуализацию разных эмоций на двумерном аватаре.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

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.

9) ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом – М.: ИПК Издательство стандартов, 2001.

10) Документация Tauri [Электронный ресурс]. URL: https://tauri.app/v1/guides/ (даты обращения: 15.11.2022-03.04.2023)

11) Документация Tauri Api для JS/TS [Электронный ресурс]. URL: https://tauri.app/v1/api/js/ (даты обращения: 12.11.2022-05.04.2023)

12) Руководство по языку TypeScript [Электронный ресурс]. URL: https://www.typescriptlang.org/docs/ (даты обращения: 07.07.2022-01.04.2023)

13) Библиотека React: Исходные коды, руководства, примеры [Электронный ресурс]. https://legacy.reactjs.org/docs/getting-started.html (даты обращения: 07.03.2022-5.04.2023)

14) Библиотека Material Icons: Исходные коды, руководства, примеры [Электронный ресурс]. https://mui.com/material-ui/material-icons/ (даты обращения: 07.03.2022-5.04.2023)

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

15) Библиотека Material UI: Исходные коды, руководства, примеры [Электронный ресурс]. https://mui.com/material-ui/getting-started/overview/ (даты обращения: 07.03.2022-5.04.2023)

16) Библиотека Html2Canvas: Документация, примеры [Электронный ресурс]. https://html2canvas.hertzen.com/documentation (даты обращения: 07.03.2022-5.04.2023)

17) Библиотека React Color: Исходные коды, руководства, примеры [Электронный ресурс]. https://casesandberg.github.io/react-color/ (даты обращения: 07.03.2022-5.04.2023)

18) Библиотека React Draggable: Исходные коды, руководства, примеры [Электронный ресурс]. https://github.com/react-grid-layout/react-draggable (даты обращения: 07.03.2022-5.04.2023)

19) Библиотека React Image Size: Исходные коды, руководства, примеры [Электронный ресурс]. https://github.com/andreyk1512/react-image-size#readme (даты обращения: 07.03.2022-5.04.2023)

20) Библиотека React Toastify: Исходные коды, руководства, примеры [Электронный ресурс]. https://github.com/fkhadra/react-toastify#readme (даты обращения: 07.03.2022-5.04.2023)

21) Библиотека set-interval-async: Исходные коды, руководства, примеры [Электронный ресурс]. https://github.com/ealmansi/set-interval-async (даты обращения: 07.03.2022-5.04.2023)

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

ПРИЛОЖЕНИЕ 1 ОПИСАНИЕ И НАЗНАЧЕНИЕ КЛАССОВ

ФАЙЛЫ TYPESCRIPT, ИСПОЛЬЗУЕМЫЕ ПРИЛОЖЕНИЕМ

Файлы проекта можно разделить на несколько смысловых типов:

1. Глобальные компоненты
2. Модальные компоненты
3. Компоненты, связанные с эмоциями
4. Компоненты, связанные со спрайтами
5. Компоненты, связанные с настройками
6. Компоненты, связанные с визуализацией маскота
7. Интерфейсы и перечисления
8. Дополнительные утилиты

ГЛОБАЛЬНЫЕ КОМПОНЕНТЫ

Файл	**Назначение**
main.tsx	Точка входа в приложение, содержит инстанс компонента App.
App.tsx	Основной компонент, в котором происходит взаимодействие пользователя с программой. При запуске приложения возвращает пользователю компонент Projects для выбора проектов, затем визуализирует основное окно для работы с выбранным проектом, включая в себя компоненты MascotCanvas, EmotionsSelection, PartsSelection, BackgroundColorPicker, CamSelection, MicMinMaxSelection, MicSelection, ShakeSettings. Также внутри компонента создается контекст с объектом IMascot, который используется внутри вышеописанных компонентов.
Projects.tsx	Компонент, позволяющий пользователю выбрать/создать/удалить проект. Состоит из списка кнопок, позволяющих манипулировать созданием/выбором/удалением проектов, а также из списка загруженных проектов. Использует модальные компоненты ProjectAdd, ProjectAddScrap и Proceed. При появлении в DOM’е использует/инициализирует список проектов

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

	пользователя из файла, сохраненного в системе, и контролирует его контент.

МОДАЛЬНЫЕ КОМПОНЕНТЫ

Файл	**Назначение**
EmotionAdd.tsx	Позволяет пользователю создать/редактировать эмоцию в выбранном проекте.
PartAdd.tsx	Позволяет пользователю создать/редактировать спрайт в выбранном проекте.
Proceed.tsx	Представляет из себя всплывающее окно с кнопкой и текстом. Нажатие на кнопку является подтверждением того, что пользователь согласен с описанным текстом.
ProjectAdd.tsx	Позволяет пользователю создать пустой проект.
ProjectAddScrap.tsx	Позволяет пользователю создать проект на основе папки с подпапками, хранящими изображения, где подпапки станут эмоциями, а изображения – спрайтами.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

КОМПОНЕНТЫ, СВЯЗАННЫЕ С ЭМОЦИЯМИ

Файл	**Назначение**
EmotionPart.tsx	Является частью списка внутри EmotionsSelection. Хранит в себе информацию о эмоции в выбранном проекте. Имеет функционал изменения видимости используемой эмоции.
EmotionsSelection.tsx	Визуализирует все эмоции, используемые в проекте. Содержит EmotionAdd с возможность использования в режиме редактирования или создания, также функционал выбора и удаления выбранной эмоции из проекта и список всех эмоций, реализуемых через EmotionPart.

КОМПОНЕНТЫ, СВЯЗАННЫЕ СО СПРАЙТАМИ

Файл	**Назначение**
PartPart.tsx	Является частью списка внутри PartsSelection. Хранит в себе информацию о эмоции в выбранном проекте. Имеет функционал изменения видимости используемой эмоции, а также функционал изменение положения в иерархии визуализации в окне.
PartsSelection.tsx	Визуализирует все спрайты, используемые в выбранной в EmotionsSelect эмоции. Содержит PartAdd с возможность использования в режиме редактирования или создания, также функционал выбора и удаления выбранного спрайта из проекта и список всех спрайтов для выбранной эмоции, реализуемых через PartPart.

КОМПОНЕНТЫ, СВЯЗАННЫЕ С ВИЗУАЛИЗАЦИЕЙ МАСКОТА

Файл	**Назначение**
MascotPart.tsx	Визуализирует внутри MascotCanvas изображение, хранимое в переданном при создании спрайте. Если спрайт является активным спрайтов в PartsSelection, то данный компонент предоставляет возможность перемещать себя внутри области MascotCanvas и изменять свои размеры.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

	После интеракции с пользователем, компонент записывает измененное положение / размер спрайта в контекст объект IMascot из App.tsx. Если спрайт не является активным спрайтов в PartsSelection, то данный компонент просто возвращает изображение с указанными в спрайте (объект IPart) характеристиками (размер и позиция внутри MascotCanvas).
ShadowPart.tsx	Является облегченным аналогом компонента MascotPart. Различие заключается в отсутствии функционала перемещения и измене.
MascotCanvas.tsx	При инициализации связывается с контекстом и получает из него объект IMascot. Для выбранной пользователем эмоции визуализирует на отведенной ему области компоненты MascotPart для всех спрайтов эмоции, которые пользователь оставил видимыми в компоненте PartsSelection. Имеет функционал изменения зума. Если пользователь входит в область расположения MascotCanvas, MascotPart, являющийся репрезентацией выбранного пользователем спрайта, становится доступным для редактирования.
ShadowCanvas.tsx	Визуализирует внутри себя объекты ShadowPart. При инициализации получает объект IMascot и открывает интервал, связывающийся через Tauri Api [11] с Rust частью для получения данных о состоянии маскота в формате IMascotData. Полученные данные обрабатываются внутри компонента в визуализацию маскота через изменение копии полученного объекта маскота. Главным отличием от MascotCanvas является единовременный рендер всех спрайтов всех эмоций, добавленных пользователем в маскота. В зависимости от получаемых с Rust’а данных неиспользуемые ShadowPart становятся невидимыми в DOM’е, что позволяет добиться высокой скорости изменения визуализации маскота в

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

	графическом интерфейсе.

ИНТЕРФЕЙСЫ И ПЕРЕЧИСЛЕНИЯ

Файл	**Назначение**
EEmotion.ts	Енумератор типов эмоций (злость, страх и т.п.).
EPart.ts	Енумератор типов спрайтов (лицо, закрытые глаза, открытые глаза и т.п.).
IConf.ts	Интерфейс, используемый для сохранения данных в конфигурационной файле проектов, использующимся в Projects.tsx.
IEmotion.ts	Интерфейс, используемый для описания эмоции. Является частью IMascot.
IMascot.ts	Интерфейс, используемый для описания маскота.
IMascotData.ts	Интерфейс, используемый для получения данных с Rust о состоянии маскота.
IPart.ts	Интерфейс, используемый для описания спрайта. Является частью IEmotion.

ДОПОЛНИТЕЛЬНЫЕ УТИЛИТЫ

Файл	**Назначение**
Colors.ts	Описывает цвета, используемые приложением.
Commands.ts	Имплементации методов, используемых для вызова Rust методов через Tauri Api [11]. Таких как: получение всех камер, выбор камеры, получение информации о состоянии маскота и т.д.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

Config.ts	Структура данных, необходимая для корректного выбора камеры в команде из Commands.ts.
DummyMascot.ts	Имплементация интерфейса IMascot с пустыми полями.
EDescriptor.ts	Используется для получения иконок типов эмоций и спрайтов по передаваемым объектам EEmotion или EPart.
MuiTheme.tsx	Стили для библиотеки mui/material.
Save.ts	Содержит функции для сохранения и очистки директории с проектом.
TextToColor.ts	Содержит функцию, возвращающую случайный цвет в формате hex по переданной строке.
UUIDGen.ts	Содержит функцию генерации случайных уникальных идентификаторов.

КОМПОНЕНТЫ, СВЯЗАННЫЕ С НАСТРОЙКАМИ

Файл	**Назначение**
BackgroundColorPicker.tsx	Компонент, предоставляющий функционал выбора цвета заднего фона маскота.
CamSelection.tsx	Компонент, предоставляющий функционал выбора камеры. Получает список камер и устанавливает выбранную через Commands.
MicMinMaxDisplay.tsx	Компонент, предоставляющий функционал выбора минимальной и максимальной громкости микрофона.
MicSelection.tsx	Компонент, предоставляющий функционал выбора микрофона. Получает список микрофонов и устанавливает выбраннуй через Commands.
ShakeSettings.tsx	Компонент, предоставляющий функционал изменения уровня тряски маскота в окне визуализации маскота реализуемый через Slider.tsx.
Slider.tsx	Компонент, реализующий слайдер. Используется в

	ShakeSettings.tsx.
TwoSlider.tsx	Компонент, реализующий слайдер с двумя хендлами. Используется в MicMinMaxDisplay.tsx.

ПРИЛОЖЕНИЕ 2 НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ, СВОЙСТВ, СОБЫТИЙ ФАЙЛОВ ПРОЕКТА

Все файлы с расширением «.tsx» содержат функции, возвращающие React компонент. Описание компонентов можно найти в Приложении 1, поэтому далее при описании файлов назначение React функции не будет описано.

Файл EmotionPart.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
visible	Поле	Boolean	-	Визуализирует статус видимости спрайта.
setVisible	функция	React.Dispatch <React.SetStateAction <boolean>>	value: boolean	Сеттер для стейт хука.
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.

Файл EmotionsSelection.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
selectIndex	поле	-	-	Отвечает за подсветку выбранной пользователем эмоции.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

setSelectIndex	функция	React.Dispatch <React.SetStateAction <boolean>>	value: boolean	Сеттер для стейт хука.
openAdd	поле	boolean	-	Отвечает за состояние модального компонента EmotionAdd.
setOpenAdd	функция	React.Dispatch <React.SetStateAction <boolean>>	value: boolean	Сеттер для стейт хука.
useRedact	поле	boolean	-	Отвечает за режим модального компонента EmotionAdd (создание эмоции / редактирование).
setUseRedact	функция	React.Dispatch <React.SetStateAction <boolean>>	value: boolean	Сеттер для стейт хука.
handleListItemClick	функция	-	index: number	Обрабатывает нажатие пользователя на элемент в списке эмоций.

Файл IConf.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
path	поле	string	-	Путь до проекта в файловой системе компьютера.
name	поле	string	-	Название проекта.
previewPath	поле	string	-	Путь до изображения, используемого как превью проекта.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

Файл IEmotion.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
name	поле	string	-	Название эмоции.
visibility	поле	boolean	-	Видимость эмоции.
parts	поле	IPart[]	-	Спрайты, привязанные к эмоции.
emotion	поле	EEmotion	-	Тип эмоции.

Файл IMascot.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
emotions	поле	Emotion[]	-	Эмоции маскота.
bgColor	поле	string	-	Цвет фона маскота.
selectedEmotion	поле	number	-	Индекс выбранной пользователем эмоции.
selectedPart	поле	number	-	Индекс выбранного пользователем спрайта.
workingDir	поле	string	-	Путь до директории проекта.
projectName	поле	string	-	Название проекта.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

zoom	поле	number	-	Уровень увеличения маскота в MascotCanvas и ShadowCanvas.
shake	поле	number	-	Уровень тряски маскота в зависимости от громкости микрофона.
maxMic	поле	number	-	Уровень ограничения по максимальной громкости микрофона.
minMic	поле	number	-	Уровень ограничения по минимальной громкости микрофона.

Файл IMascotData:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
emotion	поле	string	-	Тип эмоции. Конвертируется в нормальный енам через EDescriptor.
blink	поле	boolean	-	Состояние глаз (закрытые / открытые).
lips	поле	boolean	-	Состояние губ (закрытые / открытые).
voice	поле	number	-	Громкость микрофона.

Файл IPart.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

name	поле	string	-	Имя спрайта.
visibility	поле	boolean	-	Видимость спрайта.
sourcePath	поле	string	-	Путь до изображения, используемого спрайтом.
positionX	поле	number	-	Позиция по координате X в MascotCanvas и ShadowCanvas.
positionY	поле	number	-	Позиция по координате Y в MascotCanvas и ShadowCanvas.
height	поле	number	-	Высота изображения.
width	поле	number	-	Ширина изображения.
type	поле	EPart	-	Тип спрайта.

Файл MascotCanvas.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
inFocus	поле	boolean	-	Навелся ли пользователь на компонент. Если пользователь навелся, то активируется функционал работы с изображениями в виде перемещения спрайта и изменения его размеров.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

setInFocus	функция	React.Dispatch <React.SetStateAction <boolean >>	value: boolean	Сеттер для стейт хука.
zoomIn	функция	-		Увеличивает зум.
zoomOut	функция	-		Уменьшает зум.

Файл MascotPart.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
currentPos	поле	{x: number, y:number}	-	Позиция компонента на координатной плоскости MascotCanvas.
setCurrentPos	функция	React.Dispatch <React.SetStateAction < {x: number, y:number}>>		Сеттер для стейт хука.
height	поле	number	-	Высота компонента.
setHeight	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
width	поле	number	-	Ширина компонента.
setWidth	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
loading	поле	boolean	-	Установлен в false, пока не будет загружено

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

				изображение в компонент.
setLoading	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
zoom	поле	number	-	Значение увеличения из IMascot.
setZoom	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
resizing	поле	boolean	-	True, когда происходит изменение размеров компонента. False во всех остальных случаях.
setResizing	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
infoPos	поле	{x: number, y:number}	-	Положение информационной панели.
setinfoPos	функция	React.Dispatch <React.SetStateAction < {x: number, y:number}>>	value: {x: number, y:number}	Сеттер для стейт хука.
defaultHW	поле	{x: number, y:number}	-	Стартовые значения высоты и ширины.
setDefaultHW	функция	React.Dispatch <React.SetStateAction < {x: number, y:number}>>	value: {x: number, y:number}	Сеттер для стейт хука.
sizeHandler	функция	-	mouseDownEvent: any, dir:string	Функция изменения размеров компонента

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

handleStop	функция	-	dragElement: {x: number, y:number}>>	Функция, сохраняющая в контекст изменения позиции.

Файл ShadowCanvas.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mc	поле	IMascot	-	Копия маскота из контекста.
setMc	функция	React.Dispatch <React.SetStateAction < >>	value: IMascot	Сеттер для стейт хука.
getData	функция	-	ei: number[]	Получение данных о состоянии маскота со стороны Rust.
applyDataToMascot	функция	-	Dt:IMascotData, emoIndexes: number[]	Применение данных, полученных через getData на mc внутри компонента для перерисовки.

Файл ShadowPart.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
currentPos	поле	{x: number, y: number}	-	Положение компонента относительно ShadowCanvas.
setCurrentPos	функция	React.Dispatch <React.SetStateAction		Сеттер для стейт хука.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

		< >>		
height	поле	number	-	Высота компонента.
setHeight	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
width	поле	number	-	Ширина компонента.
sestWidth	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
loading	поле	boolean	-	Установлен в false, пока не будет загружено изображение в компонент.
setLoading	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.

Файл EmotionAdd.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
handleClose	функция	-		Закрытые компонента.
handleChange	функция	-		Обработка выбора пользователем пути в файловой системе.
designation	поле	EEmotion	-	Тип добавляемой эмоции.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

setDesignation	функция	React.Dispatch <React.SetStateAction < >>		Сеттер для стейт хука.
name	поле	string	-	Название эмоции.
setName	функция	React.Dispatch <React.SetStateAction < >>		Сеттер для стейт хука.

Файл PartAdd.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
path	поле	string	-	Путь до изображения.
setPath	функция	React.Dispatch <React.SetStateAction < string>>	value: string	Сеттер для стейт хука.
designation	поле	IPart	-	Тип спрайта (открытые глаза, закрытые глаза, лицо и т.д.).
setDesignation	функция	React.Dispatch <React.SetStateAction < IPart >>	value: IPart	Сеттер для стейт хука.
name	поле	string	-	Название спрайта.
setName	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

height	поле	number	-	Высота изображения спрайта.
setHeight	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
width	поле	number	-	Ширина изображения спрайта
setWidth	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
handleChange	функция	-	Event: SelectChangeEvent	Обработка выбора пользователем пути в файловой системе.
getSizes	функция	-	Value: string	Функция, инициализирующая height и width.

Файл Proceed.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
handleOpen	функция	-	-	Показывает компонент.
handleClose	функция	-	-	Прячет компонент.

Файл ProjectAdd.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

handleOpen	функция	-	-	Показывает компонент.
handleClose	функция	-	-	Прячет компонент.
name	поле	string	-	Название проекта.
setName	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.
workDir	поле	string	-	Путь до директории, в которой будет создан проект.
setWorkDir	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.

Файл ProjectAddScrap.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
handleOpen	функция	-	-	Показывает компонент.
handleClose	функция	-	-	Прячет компонент.
destination	поле	string	-	Путь до папки, где будет инициализирован проект.
setDestination	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.
workDir	поле	string	-	Путь до папки с подпапками с

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

				изображениями.
setWorkDir	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.

Файл PartPart.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
visible	поле	boolean	-	Визуализирует статус видимости спрайта.
setVisible	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.

Файл PartsSelection.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
selectedIndex	поле	number	-	Указатель на индекс выбранного.
setSelectedIndex	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
openAdd	поле	boolean	-	Контролирует показ компонента PartAdd.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

setOpenAdd	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
useRedact	поле	boolean	-	Контролирует.
setUseRedact	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.

Файл BackgroundColorPicker.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
open	поле	boolean	-	Контроллер открытия.
setOpen	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
handleOpen	функция	-		Функция открытия.
handleClose	функция	-		Функция закрытия.
color	поле	string	-	Выбранный цвет.
setColor	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

Файл MicMinMaxDisplay.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
min	поле	number	-	Уровень минимального звука.
setMin	функция	React.Dispatch <React.SetStateAction <number>>	value: number	Сеттер для стейт хука.
max	поле	number	-	Уровень максимального звука.
setMax	функция	React.Dispatch <React.SetStateAction <number>>	value: number	Сеттер для стейт хука.
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
handleChange	функция	-	change:any	Контроллер изменения состояние компонента.

Файл Slider.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
currentPos	поле	{x:number, y:number}	-	Позиция хендла слайдера.
setCurrentPost	функция	React.Dispatch <React.SetStateAction < >>	value: {x:number, y:number}	Сеттер для стейт хука.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

drag	поле	boolean	-	True, если пользователь использует слайдер, иначе false.
setDrag	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
normToNums	функция	number	value: number	Нормирует значение currentPos в значение от 0 до 100.
numsToNorm	функция	number	value: number	Денормирует значение от 0 до 100 в currentPos.
handleStop	функция	-	MouseEvent: any	Функция, срабатывающая при конце взаимодействия пользователя с хендлом слайдера. Сохраняет текущую позицию.

Файл TwoSlider.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
currentPosMin	поле	number	-	Позиция хендла минимальной громкости слайдера.
setCurrentPostMin	функция	React.Dispatch <React.SetStateAction < number>>	value: number	Сеттер для стейт хука.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

currentPosMax	поле		-	Позиция хендла максимальной громкости слайдера.
setCurrentPostMax	функция	React.Dispatch <React.SetStateAction < >>		Сеттер для стейт хука.
drag	поле	boolean	-	True, если пользователь использует слайдер, иначе false.
setDrag	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
normToNums	функция	number	value: number	Нормирует значение value в значение от 0 до 100.
numsToNorm	функция	number	value: number	Денормирует значение value от 0 до 100 в currentPosMin или currentPosMax.
handleStop	функция	-	MouseEvent: any	Функция, срабатывающая при конце взаимодействия пользователя с хендлом слайдера. Сохраняет текущую позицию.

Файл Commands.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
get_mascot	функция	IMascotData	-	Invoke метода на стороне Rust’а для получения пакета информации IMascotData о состоянии маскота.
get_cams	функция	String[]	-	Invoke метода на стороне Rust’а для получения списка камер, доступных

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

				пользователю.
set_cams	функция	-	cam: number	Invoke метода на стороне Rust’a для установки выбранной камеры по индексу.
get_mics	функция	String[]	-	Invoke метода на стороне Rust’a для получения.
set_mics	функция	-	mic: number	Invoke метода на стороне Rust’a для установки выбранного микрофона по идексу.

Файл DummyMascot.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
DummyMascot	функция	IMascot	-	Реализация интерфейса IMascot с пустыми полями. Используется как заглушка при работе с контекстом приложения.

Файл EDescriptor.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
desctiptPart	функция	string	part: EPart	Возвращает изображение типа спрайта.
descriptEmotion	функция	string	part: emotion	Возвращает изображение типа эмоции.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

descriptRawEmotion	функция	EEmotion	emo: string	Расшифровка эмоции, полученной через get_mascot в нормальный тип перечисления.

Файл Save.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
saveMascot	функция	-	mascot: IMascot	Сохранение проекта.
validateProject	функция	-	mascot: IMascot	Проверка неиспользуемых файлов проекта и удаление ненужных «.masset».
_base64ToArrayBuffer	функция	string	base64: string	Перевод строки в кодировке base64 в массив битов для сохранения изображения превью для проекта.

Файл UUIDGen.ts:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
generateUUID	функция	string	-	Возвращает строку, являющуюся уникальным идентификатором.

Файл App.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**
selecting	поле	boolean	-	Если true, то пользователю показывается компонент Projects. При false - окно с

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

				выбранным проектом.
setSelecting	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
setMascot	функция	React.Dispatch <React.SetStateAction <IMascot >>		Сеттер для стейт хука.
contextVisible	поле	boolean	-	Контролирует показ контекстного меню.
setContextVisible	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
broadcasting	поле	boolean	-	True, когда происходит визуализация финальной версии маскота
setBroadcasting	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
value	функция	-	{mascot, setMascot}	Объект, передаваемый в контекст. Состоит из всех компонентов хука mascot и setMascot.

Файл Projects.tsx:

Название	**Тип члена**	**Тип значения**	**Входные параметры**	**Назначение**

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

mascot	Поле	IMascot	-	Информация о загруженном проекте из контекста.
dataPath	поле	string	-	Путь до основного файла конфигурации проектов приложения.
setDataPath	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.
openDialog	поле	boolean	-	Контролирует показ модального компонента ProjectAdd.
setOpenDialog	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
openDialogAdd	поле	boolean	-	Контролирует показ модального компонента ProjectAddScrap.
setOpenDialogAdd	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
openProceed	поле	boolean	-	Контролирует показ модального компонента Proceed при удалении проекта.
setOpenProceed	функция	React.Dispatch <React.SetStateAction < boolean >>	value: boolean	Сеттер для стейт хука.
projects	поле	IConf[]	-	Выгруженные из «conf.json» конфигурации

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

				проектов пользователя.
setProjects	функция	React.Dispatch <React.SetStateAction < >>		Сеттер для стейт хука.
selected	поле	number	-	Индекс выбранного пользователем проекта.
setSelected	функция	React.Dispatch <React.SetStateAction < number >>	value: number	Сеттер для стейт хука.
loadData	функция	-	-	Функция выгрузки конфигураций проектов из «conf.json».
addProject	функция	-	path: string, dir: string, name: string	Функция добавления нового проекта в «conf.json».
setProject	функция	-	proj: IConf	Чтение файла конфигурации выбранного проекта и сохранение полученного объекта IMascot в контекст приложения.
searchReq	поле	string	-	Наполнение строки поиска для проектов.
setSearchReq	функция	React.Dispatch <React.SetStateAction < string >>	value: string	Сеттер для стейт хука.
deleteProj	поле	number	-	Индекс удаляемого проекта.
setDeleteProj	Функция\	React.Dispatch <React.SetStateAction	value: number	Сеттер для стейт хука.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

		<number >>		
deleteProject	функция	-	-	Функция, удаляющая директорию с выбранным проектом.
validateProject	функция	-	path: string	Функция, проверки файла конфигурации проекта.
createProjectFromFiles	функция	-	path: string, destination: string	Функция, конвертирующая контент внутри папки path в проект, создаваемый в папке destination.

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

ПРИЛОЖЕНИЕ 3 ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ

1. Спрайт – графический объект в компьютерной графике. Как правило изображение
2. Спрайт (в контексте объекта данных в приложении) – объект, характеризующийся изображением, размером и положением относительно маскота
3. Маскот – совокупность спрайтов, создающая аватара пользователя.
4. Контекст (в контексте реализации приложения) – context – понятие из библиотеки React [13], позволяющее передавать объект данных внутри DOM’а
5. DOM (Document Object Model)— это объектная модель документа (Document Object Model). Представляет собой древовидную структуру страницы, состоящую из узлов. Каждый узел –объект, который может иметь определённые свойства и методы.
6. Рендер – рендер или отрисовка является терминов, характеризующим получения изображения на графическом интерфейсе из набора данных
7. Превью - уменьшенная до небольших размеров копия изображения для компактного представления пользователю. В данном проекте представляет изображение маскота из рабочей области проекта в момент сохранения проекта
8. API (Application Programming Interface)– сервисный контракт между двумя приложениями. Контракт определяет, как они взаимодействуют друг с другом
9. Фреймворк – программная среда, упрощающая создание программного обеспечение через использование готовых подходов к разработке
10. Компонент – в React самодостаточные элементы, которые можно использовать на странице любое количество раз.
11. Зум – увеличение или уменьшение объекта

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Номера листов (страниц)	Всего	Номер	Входящ	Подп.	Дата

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения

					листов (страниц в документе)	докумен та	ий номер сопр. докумен та и дата		
Изм.	Изменен ных	Замененн ых	Нов ых	Аннулиров анных					

Номер изменения	Подпись ответственного за внесение изменения	Дата внесения изменения