**Введение**

Была поставлена задачаразработать игровое приложение «Сказки».

Цель данного проекта заключается в создании компьютерной игры.С помощью данной игры пользователь сможет провести время за досугом.

Создаваемое компьютерной игры рассчитано на любого рода пользователей.

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел носит название «Анализ задачи». В нем можно ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, инструменты разработки и МЖЦ ПО. Также в этом разделе можно узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе «Инструменты разработки» будет рассмотрена среда, в которой создается данный проект.

В разделе «Проектирование задачи» будут рассмотрены основные аспекты разработки компьютерной игры. Здесь можно узнать об организации данных в контексте среды разработки. В данном разделе будет описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации.

«Реализация задачи» – это третий раздел отчета, в котором описываются все элементы и объекты, которые будут использованы при реализации данной игры. В этом разделе описаны функции игрока и их структура. Описано руководство программиста.

Четвертый раздел – «Тестирование». В нем будет описано функциональное и нефункциональное тестирование данной программы, т.е. будет протестирован каждый пункт меню, операция, которая выполняется игрой. В разделе «Руководство пользователя» описана подробная инструкция использования игры для будущих пользователей.

«Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств.

В разделе «Список используемых источников» будет приведен список используемых при разработке приложения.

В приложениях к пояснительной записке будут приведены диаграммы.

**1 Анализ задачи**

**1.1 Постановка задачи**

**1.1.1 Организационно-экономическая сущность задачи**

Темой данного проекта является разработка компьютерной, десктопной, онлайн игры «Сказки».

Цель данного проекта заключается в создание продукта, который будет является итоговой работой по ТРПО. данный программный продукт будет является развлекательным контентом .

Рассмотрим игру Hearthstone. В данной игре есть такие функции как перетаскивание карт и события с ними, создание профиля,его кастомизация, личный кабинет. Все эти основные функции будут присутствовать в новом ПП.

**1.1.2 Функциональные требования - описание перечня функций и задач, которые должен выполнять будущий ПП.**

Игрок:

* авторизация
* Регистрация
* редактирование профиля
* выход из профиля
* поиск игровой партии
* Участие в игровой партии
* Взаимодействие с чатом в течении игровой партии
* Настройка звука и яркости
* Обучение

**1.1.3 Описание входной, выходной и условно постоянной информации**

Вся информация, которой оперирует пользователь в процессе решения задачи подразделяется на:

– входную информацию;

– выходную информацию;

– условно-постоянную информацию.

Входной информацией выступают: Поля регистрации, карты

Выходной информацией выступают: Карты, личный кабинет, профиль,

Условно-постоянной информацией выступают: Кол-во игроков в пратии, уровень звука и яркости

Описание игрового процесса и его элементов

Цель игры: Получить максимальное кол-во победоносных очков

Элементы игрового процесса:

1. Игровые карты
2. Сюжетный карты «А» и «Б»
3. Жетоны золотых монет
4. Жетоны мечей
5. Жетоны доспехов
6. Жетоны жезлов
7. Победные очки(ПО)

Игровой процесс

1.Подготовка к партии:

* + 1. Создание колоды игровых карты
    2. Каждому игроку выдается 3 случайные карты
    3. Создаются две стопки сюжетных карт («А» и «Б»), а затем вынимается по 1 случайной карте из каждой стопки. Обе взятые карты раскрываются и кладутся на стол. Карта «А» определит, какие предметы можно будет использовать во время игры и какими эффектами они будут обладать, а карта «Б» — сколько победных очков принесут предметы в конце игры
    4. Случайным образом определяется первый игрок

1. Ход игровой партии
   * 1. Игрок берет 1 карту из колоды карт. Если в колоде закончились карты, карту вы игрокне берёт.
     2. Разыгрывает один из эффектов сюжетной карты «А»(по желанию). В этой фазе игрок можете разыграть один предмет игрок вовращает его в запас и применяются эффект(ы), указанный(е) на сюжетной карте «А».
     3. Игрок разыгрывает карту с руки. В этой фазе игрок обязан выбрать одну карту на руке и разыграть её. Текст и символы карты определят, сколько победных очков она принесёт, куда её нужно будет поместить
2. Конец партии

После того как заканчивается колода карт, игроки уже не берут новые карты. Игра продолжается , пока один из игроков не начнёт ход без карт на руке. Игра немедленно заканчивается (этот игрок не делает ход). После этого вступают в силу эффекты, указанные на сюжетной карте «Б». Затем подсчитывают суммарный результат Победных очков — игрок, набравший больше очков, побеждает. При равенстве побеждает претендент с наибольшим суммарным числом предметов и карт на руке. Если ничья не разрешается, в игре несколько победителей

Игровые карты

Таблица 1-Игровые карты.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя карты | Кол-во карт | ПО за исполь-зование карты | Тип карты | Место, куда помещается  карта | Свойство карты |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | Цыганка | 4 | 1+ | женщина | В общую зону | Все игроки показывают карты на руках. Игрок скинувший карту, получает 1 ПО + 1 ПО за каждое раскрытое место. Игрок берет одну цыганку за каждую раскрытую цыганку |
|  | Старый мудрец | 2 | ? | Мужчина | Перед собой | Если сейчас день, игрок получает 2 ПО в конце своего хода. |
|  | Дитя ночи | 2 | ? | Злодей | Перед собой | Если сейчас ночь, игрок получает 2 ПО в конце своего хода. |
|  | Огр | 2 | 2 | Злодей | Перед собой | Игрок берет на руку 1 разыгранное место и 1 разыгранную женщину |
|  | Темный владыка | 2 | ? | Злодей | В сброс | Игрок получает 2 ПО за каждого разыгранного злодея. Игрок сбрасывает 1 разыгранное место |
|  | Проклятый замок | 2 | 2 | Место | В общую зону | Игрок берет из запаса 1 доспех или 1 золотую монету. |
|  | Купец | 3 | ? | Мужчина | В общую зону | Игрок берет 2 карты из колоды. Игрок получает 2 ПО за каждого разыгранного купца |
|  | Тайная помощница | 2 | 2 | Женщина | В сброс | Игрок смотрит 2 верхние карты колоды. Игрок берет одну из них на руку, а вторую удаляет из игры. Игрок может разыграть 1 дополнительную карту в этом ходу |
|  | Чары | 2 | 1 | Событие | В сброс | Игрок берет из запаса 1 предмет: доспех, волшебную палку, золотую монету или меч. |
|  | Зачарованный лес | 2 | 2 | Место | В общую зону | Игрок берет 1 карту из колоды .Если сейчас день, все игроки берут по 1 карте из колоды. Если сейчас ночь, все игроки сбрасывают по 1 карте |
|  | Потаённая пещера | 2 | 2 | Место | В общую зону | Игрок берет из запаса 1 волшебную палку или 1 золотую монету |
|  | Фея | 2 | 2 | Женщина | В общую зону | Игрок берет из запаса 1 волшебную палку или 1 меч |
|  | Принцесса | 2 | 2 | Женщина | В общую зону | Игрок берет на руку 1 разыгранного мужчину |
|  | Мудрость короля | 2 | ? | Событие | В сброс | Игрок берет из колоды 1 карту.  если сейчас день получает 2 ПО за каждое разыгранное место.  Если сейчас ночь , получает 1 по за каждое разыгранное место |
|  | Сумасбродство короля | 2 | ? | Событие | В сброс | Игрок берет из колоды 1 карту.  если сейчас ночь получает 2 ПО за каждое разыгранное место.  Если сейчас день , получает 1 по за каждое разыгранное место |
|  | Молодой герой | 2 | 2 | Мужчина | В общую зону | Игрок берет из запаса 1 доспех или 1 меч |
|  | Ночь | 4 | 2 | Событие | В общую зону | Сейчас ночь. Сбрасываются все разыгранные карты «День» |
|  | День | 3 | 3 | Событие | В общую зону | Сейчас день . Сбрасываются все разыгранные карты «Ночь» |
|  | Дракон | 2 | ? | Злодей | В сброс | Игрок сбрасывает 1 разыгранное место и 1 разыгранного мужчину. Игрок получает 2 ПО за каждую сброшенную карту |
|  | Ведьма | 2 | 0 | Женщина | Перед собой | Игрок не может больше брать карты из колоды, но может играть одной дополнительной картой в свой ход |
|  | Волшебник | 2 | 2 | Мужчина | В общую зону | Игрок берет верхнюю карту из колоды и немедленно её разыгрывает |

Таблица 2-Сюжетные карты А и Б

|  |  |
| --- | --- |
| Номер карты | Эффекты карты |
| 1 | 2 |
| А1  1 | 1)Игрок сбрасывает 1 доспех, чтоб взять верхнюю карту стопки сброса.  2) Игрок сбрасывает меч и 1 любого разыгранного злодея, чтобы получить 3 ПО.  2 |
|  | 3) Игрок сбрасывает 1 жезл, чтобы взять 1 карту из колоды и 1 золотую монету из запаса, а также получить 1 ПО. |
| А2 | 1) Игрок сбрасывает 1 жезл, чтобы разыграть 1 дополнительную карту в этом же ходу.  2) Игрок сбрасывает 1 меч и 1 любого разыгранного злодея ,чтобы взять из запаса 1 золотую монету и получить 1 ПО.  3)Если перед Игроком на столе есть разыгранный злодей, Игрок сбрасывает 1 доспех, чтобы заставить каждого игрока сбросить 1 карту с руки, либо потерять 1 ПО. Игрок получает потерянные другими игроками ПО. |
| А3 | 1) Игрок сбрасывает 1 доспех и 1 меч, чтобы взять 2 карты из колоды и получите 3 ПО.  2) Игрок сбрасывает 1 жезл и любого разыгранного злодея, чтобы взять из запаса 1 золотую монету и получите 1 ПО. |
| Б1 | 1)Игрок получает 12 ПО за каждый набор из 3 золотых монет и 1 жезл. |
| Б2 | 1) Игрок получает 3 ПО за каждую пару меч и злодей. И еще 3 ПО за каждую пару меч и мужчина  2) Между игроками распределяются призовые места согласно суммарному числу их золотых монет. В зависимости от места игрок получает определенное кол-во ПО |
| Б3 | 1)Каждая пара доспех и меч считается за 3 золотые монеты.  2)Игроки получают дополнительные ПО за кол-во их золотых монет |

**1.1.4 Нефункциональные требования**

**Требования к применению:**Позволяет играть в карточную игру «Сказки».

**Требования к реализации:** Для реализации используется платформа Unity с языком программирования C#.

**Требования к надежности:** система может быть недоступна не более чем 24 часа в год.

**Требования к интерфейсу:** при разработке приложения должны быть использованы преимущественного нейтральные оттенки. Грамотный пользовательский интерфей

**1.2 Диаграмма вариантов использования**

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

Актером или действующим лицом является любая сущность, взаимодействующая с системой извне (рисунок 1). Это может быть как живое существо, так и любая друга система, которая может служить источником воздействия на моделируемую систему так, как определяет сам разработчик.

Актер



Рисунок 1 – Графическое обозначение актера

Вариант использования применяется для спецификаций общих особенностей системы и любой другой сущности. Отдельные варианты использования обозначаются на диаграмме эллипсом, в котором содержится его краткое название (рисунок 2).

Взятие карты

Рисунок 2 – Графическое обозначение вариантов использования

Отношение ассоциации и используется при построении всех графических моделей. Оно служит для обозначения роли актера и отдельном варианте использования. На диаграмме отношение ассоциации обозначается сплошной линией между актером и вариантом использования (рисунок 3).





Взятие карты

Рисунок 3 – Графическое обозначение отношения ассоциации

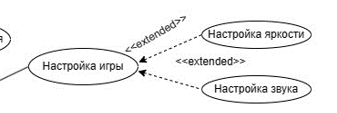
Для отображения взаимосвязи экземпляра отдельного варианта использования с более общим вариантом, используется отношение расширения, обозначаемое направленной пунктирной линией со стрелкой от исходного варианта. Данная линия помечается ключевым словом <<extend>> (рисунок 4).

Рисунок 4 – Графическое обозначение отношения расширения

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данная линия помечается ключевым словом <<include>> (рисунок 5).

Рисунок 5 – Графическое обозначение отношения включения

Определяя для выбранного актера варианты использования и устанавливая отношения между вариантами использования, получим полную диаграмму вариантов использования, её можно увидеть в Приложении А (рисунок А.1).

**1.3 Модель данных**



Рисунок 6 – Связь между двумя сущностями



Рисунок 7 – Сущность

**Таблица**  - Сущности

|  |  |  |
| --- | --- | --- |
| № | Сущность | Описание |
| 1 | Игрок | Содержит информацию о игроках |
| 2 | Партия | Содержит информацию о партии |
| 3 | Карта | Содержит информацию о карте |
| 4 | Тип карты | Содержит информацию о типе карты |
| 5 | Жетон | Содержит информацию о жетоне |
| 6 | Тип жетона | Содержит информацию о типе жетона |

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных.

Модели данных служат для проектирования структуры постоянных хранилищ данных, используемых системой.

Модели данных могут создаваться на уровне предприятия, отдела или отдельного приложения. Модели данных на уровне предприятия или отдела могут использоваться для предоставления стандартных определений для ключевых бизнес-сущностей (таких как клиент и сотрудник), которые будут применяться всеми приложениями всего бизнес-процесса или его части. С помощью этих типов Моделей данных можно также определить, какая система предприятия будет "владельцем" данных для конкретной бизнес-сущности и какие другие системы будут пользователями (подписчиками) данных.

С моделью данных созданную для игрового приложения «Сказки» можно ознакомиться в Приложение А (рисунок А.3).

**2 Проектирование**

**2.1 Выбор стратегии разработки и модели жизненного цикла**

Для разработки мобильного приложения следует выбрать стратегию разработки и модель жизненного цикла. Выбираем посредством составления таблиц:

**Таблица 2 –** Выбор модели жизненного цикла на основе характеристик требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории требований | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Являются ли требования к проекту легко определимыми и реализуемыми? | Да | Да | Да | Нет | Нет | Нет |
| 2. | Могут ли требования быть сформулированы в начале ЖЦ? | Да | Да | Да | Да | Нет | Нет |
| 3. | Часто ли будут изменяться требования на протяжении ЖЦ? | Нет | Нет | Нет | Нет | Да | Да |
| 4. | Нужно ли демонстрировать требования с целью их определения? | Нет | Нет | Да | Нет | Да | Да |
| 5. | Требуется ли проверка концепции программного средства или системы? | Нет | Нет | Да | Нет | Да | Да |
| 6. | Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Нужно ли реализовать основные требования на ранних этапах разработки? | Нет | Нет | Да | Да | Да | Да |

Вычисления: 4 за каскадную, 4 за V- образную, 5 за RAD, 5 за инкрементную, 3 за быстрого прототипирования и 3 за эволюционную.

Итог: на основе результатов заполнения таблицы 2 подходящей является RAD модель и инкрементная модель.

**Таблица 3** – Выбор модели жизненного цикла на основе характеристик команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории команды разработчиков  проекта | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Являются ли проблемы предметной области проекта новыми для большинства разработчиков? | Нет | Нет | Нет | Нет | Да | Да |
| 2. | Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков? | Да | Да | Нет | Нет | Нет | Да |
| 3. | Изменяются ли роли участников проекта на протяжении ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Является ли структура процесса разработки более значимой для разработчиков, чем гибкость? | Да | Да | Нет | Да | Нет | Нет |
| 5. | Важна ли легкость распределения человеческих ресурсов проекта? | Да | Да | Да | Да | Нет | Нет |
| 6. | Приемлет ли команда разработчиков оценки, проверки, стадии разработки? | Да | Да | Нет | Да | Да | Да |

Вычисления: 5 за каскадную, 5 за V-образную, 4 за RAD, 5 за инкрементную, 2 за быстрого прототипирования и 1 за эволюционную.

Итог: на основе результатов заполнения таблицы 3 подходящими являются каскадная, V-образная и инкрементная модели.

**Таблица 4 –** Выбор модели жизненного цикла на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории коллектива пользователей | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Будет ли присутствие пользователей ограничено в ЖЦ разработки? | Да | Да | Нет | Да | Нет | Да |
| 2. | Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки? | Нет | Нет | Нет | Да | Да | Да |

Продолжение таблицы 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3. | Будут ли пользователи вовлечены во все фазы ЖЦ разработки? | Нет | Нет | Да | Нет | Да | Нет |
| 4. | Будет ли заказчик отслеживать ход выполнения проекта? | Нет | Нет | Нет | Нет | Да | Да |

Вычисления: 0 за каскадную, 0 за V-образную, 3 за RAD, 0 за инкрементную, 4 за быстрого прототипирования и 1 за эволюционную.

Итог: на основе результатов заполнения таблицы 4 подходящей является модель быстрого проектирования.

**Таблица 5 –** Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории типов проекта и рисков | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Разрабатывается ли в проекте продукт нового для организации направления? | Нет | Нет | Нет | Да | Да | Да |
| 2. | Будет ли проект являться расширением существующей системы? | Да | Да | Да | Да | Нет | Нет |
| 3. | Будет ли проект крупно- или среднемасштабным? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Ожидается ли длительная эксплуатация продукта? | Да | Да | Нет | Да | Нет | Да |
| 5. | Необходим ли высокий уровень надежности продукта проекта? | Нет | Да | Нет | Да | Нет | Да |
| 6. | Предполагается ли эволюция продукта проекта в течение ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Велика ли вероятность изменения системы (продукта) на этапе сопровождения? | Нет | Нет | Нет | Да | Да | Да |
| 8. | Является ли график сжатым? | Нет | Нет | Да | Да | Да | Да |
| 9. | Предполагается ли повторное использование компонентов? | Нет | Нет | Да | Да | Да | Да |
| 10. | Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)? | Нет | Нет | Нет | Нет | Да | Да |

Вычисления: 5 за каскадную, 6 за V-образную, 6 за RAD, 8 за инкрементную, 4 за быстрого прототипирования и 6 за эволюционную.

Итог: на основе результатов заполнения таблицы 5 подходящей является инкрементная модель.

Общий итог: в итоге заполнения таблиц – наиболее подходящей является инкрементная модель.

**2.2 Инструменты разработки**

Для разработки UML-диаграмм использовалось приложение «Draw.io», так как данное приложение имеет удобный и понятный интерфейс для составления диаграмм.

Для UX/UI дизайна использовалась «Figma», так как в данном приложении удобно работать с компонентами — заранее подготовленными элементами дизайна (кнопки, формы и т. д.), изменения которых распространяются на все его копии.

Для составления документации использовался «Microsoft Word», так как эта программа наиболее подходит для выполнения данной задачи.

Для разработки данного проекта выбран конструктор сайтов Glide. Так как Glide является одним из самых удобных конструкторов для создания мобильных приложений.

Unity - это мощный игровой движок, который позволяет создавать игры для разных платформ, таких как ПК, мобильные устройства, консоли и т.д. Он имеет широкий набор инструментов для разработки игр, включая удобный редактор сцен и объектов, систему компонентов, поддержку скриптов на C# и других языках программирования, а также возможности для создания 3D-моделей и анимации.

Unity был выбран для разработки игры из-за своей многофункциональности и удобства использования. Он позволяет создавать игры с высоким качеством графики и звука, а также обладает хорошей производительностью и оптимизацией для разных платформ. Кроме того, Unity имеет большое сообщество разработчиков и множество готовых решений, которые можно использовать в своих проектах.

Также Unity поддерживает различные форматы файлов и интегрируется с другими программами, что делает процесс разработки более эффективным и удобным. В целом, Unity является одним из наиболее популярных и универсальных инструментов для создания игр, что делает его отличным выбором для разработки проекта.

Разработка проекта будет происходить на компьютере со следующими

параметрами:

– процессор: 13th Gen Intel(R) Core (TM) i3-1315U 1.20 GHz

– объем оперативной памяти: 8.00 GB;

– объем места на жестком диске: 500 GB;

– ОС: Windows 10

**2.3 Разработка UML-диаграмм**

В первом разделе «1.1 Постановка задачи» были описаны функциональные требования к проекту. На их основе построена диаграмма последовательности.

Диаграмма последовательности – это диаграмма, предназначенная для моделирования взаимодействия объектов системы во времени, а также обмена сообщениями между ними.

Графические диаграммы последовательности имеют два представления. Одно – слева направо – в виде вертикальных линий, которые соответствуют линии жизни отдельного участка взаимодействия. Второе – вертикальная временная ось, направленная сверху вниз.

На диаграмме отображен процесс добавления товара в корзину.

Диаграмма последовательности представлена в Приложении А (рисунок А.7).

**2.4 Разработка пользовательского интерфейса**

Важным элементом проектирования данного программного продукта является описание внешнего интерфейса. Для разработки визуального дизайна использовались сдержанные, мягкие цвета для удобства использования программного продукта. В ходе разработки был спроектирован дизайн главного меню игрового приложения «Сказки». Разработанное главное меню расположено в Приложении А (рисунок А.2).

Для организации эффективной работы пользователя нужно создать целостный программный продукт данной предметной области, в котором все компоненты будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя.

Прототип – это наглядная модель пользовательского интерфейса. В сущности, это «черновик», созданный на основе представления разработчика о потребностях пользователя. Итоговые страницы приложения могут отличаться от прототипа. С прототипом главного меню можно ознакомиться в Приложении А (рисунок А.2).

Диаграмма деятельности – это графическое представление процессов и задач, выполняемых в рамках определенного проекта или деятельности. Диаграмма представлена в Приложении А (рисунок А.8).

Диаграмма объектов описывает конкретные экземпляры объектов и напрямую соотносится с диаграммой классов, которая даёт общее представление о конфигурации системы. Она используется для документирования структур данных и создания статических снимков состояний объектов принимая во внимание реальные экземпляры или прототипы. Диаграмма объектов и диаграмма классов также расположены в приложении А (рисунки А.4 и А.5).

Функциональная модель – это модель, которая показывает, какие функции у проектируемой модели и как они взаимодействуют между собой. Для построения функциональной модели предназначена методология функционального моделирования DFD. Функциональная модель представлена в приложении А на рисунке А.6.

UX/UI макеты представлены в Приложении Б.

**2.5 Тесты на использование**

В ходе разработки мобильного приложения были составлены тесты, которые необходимо выполнить в дальнейшем. Тесты составлены таким образом, чтобы предусмотреть максимальное количество возможных действий. Тесты описаны в разделе тестирование.

**3 Реализация**

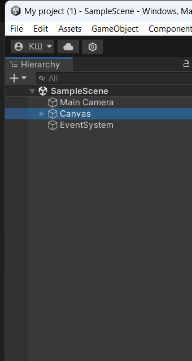
**3.1 Руководство программиста**

Данный программный продукт был разработан с помощью платформы Unity. Данная платформа предлагает большой выбор в способах создания игр.

**3.1.1 Создание игрового приложения**

Для того, чтобы создать свое игровое приложение в Unity необходимо сначала зарегистрироваться. После авторизации нужно нажать кнопку «New project». Далее нужно выбрать приложение какого вида пользователь хочет создать. После создания представлен широкий выбор различных компонентов. данных

**3.1.2 Настройка функциональности и дизайна**

В Unity представлен большой выбор различных компонентов.. Компоненты можно редактировать: менять их дизайн, размер. В Unity есть уже готовые поле свойств(рисунок 8), свойства объекта (рисунок 9)

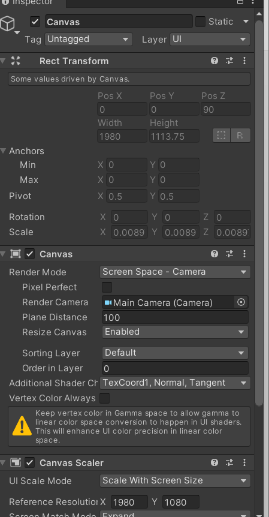


Рисунок 8 – Формы для заполнения

Рисунок 9 – свойства объекта

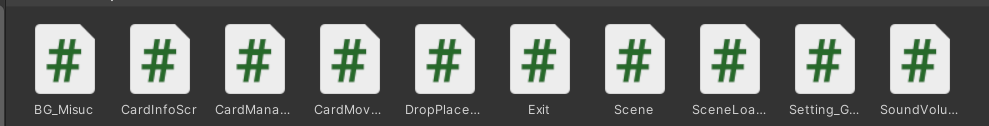
К различным компонентам (кнопкам) можно создавать доп функции функции рис 10.

Рисунок 10 – Работа c функциями

**4 Тестирование**

Тестирование программного обеспечения было осуществлено в соответствии с тестами на использование, описанными в разделе «2.5 Тесты на использование».

**Таблица 6** – Тестирование функций

|  |  |
| --- | --- |
| № | Статус |
| Т1 | Выполнено успешно |
| Т2 | Выполнено успешно |
| Т3 | Выполнено успешно |
| Т4 | Выполнено успешно |

**Таблица 7** – Тесты на использование

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название теста | Действие | Исходная информация | Ожидаемый результат | Фактический результат | Статус |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 Просмотр сцен приложения | Нажатие на все пункты меню в приложении |  | Открытие выбранного пункта меню | Открытие выбранного пункта меню | Выполнено успешно |
| 2 перетаскивание карт с руки на поле | 1 начать партию  2 перетащить карту |  | Карта зафиксировалась на общем поле | Карта зафиксировалась на общем поле | Выполнено успешно |
| 3 перетаскивание карт внутри руки | 1начать партию  2 перетащить карту |  | Карта встала между двумя другими | Карта встала между двумя другими | Выполнено успешно |
| 4 Изменение громкити музыки | сдвинуть ползунок музыки |  | Изменение громкости | Изменение громкости | Выполнено успешно |

В результате проведения тестирования программного продукта выяснилось, что все раннее оговоренные функции были разработаны, а также протестированы. Тесты показали, что все функции работают правильно.

**5 Руководство пользователя**

**5. 1 Общие сведения**

Наименованием программного продукта является игровое приложение «Сказыки». Назначение - для людей любого возраста, которые хотят поиграть в игру. Периодичность использования ПП - по мере необходимости.

**5.2 Выполнение программы**

На главном меню игроку предоставлена возможность пройти обучение, начать партию с другим игроком, редактировать профиль,пройти обучение и посмотреть карты, настроки громкости игры и выход(рисунок 11) .



Рисунок 11 – главное меню

Окно редактирования профиля(рисунок 12 ).



Рисунок 12 – Редактирование профиля

НА экране настроек, можно изменять громкость звуков, музыки и будут ли показываться подсказки(рисунок 13).



Рисунок 13 – Настройки игры

При нажатии на обучение, игрок переходит на сцену описания карт, а после посмотреть пример партии(рисунок 14)

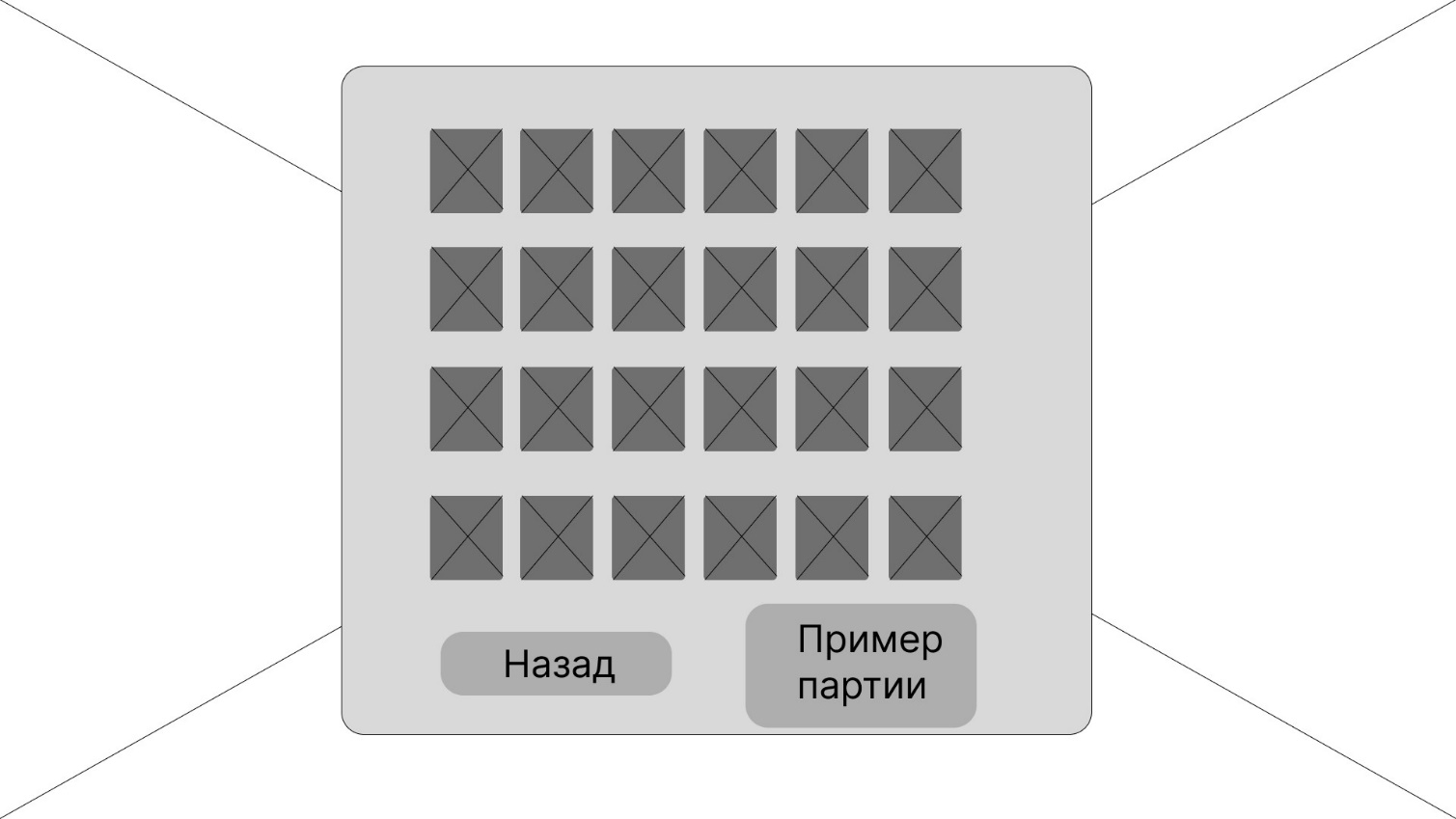


Рисунок 14 – Свойства карт

При нажатии на кнопку играть, игрок поппадает в партию с другим игроком и партия начинается(рисунок 15).

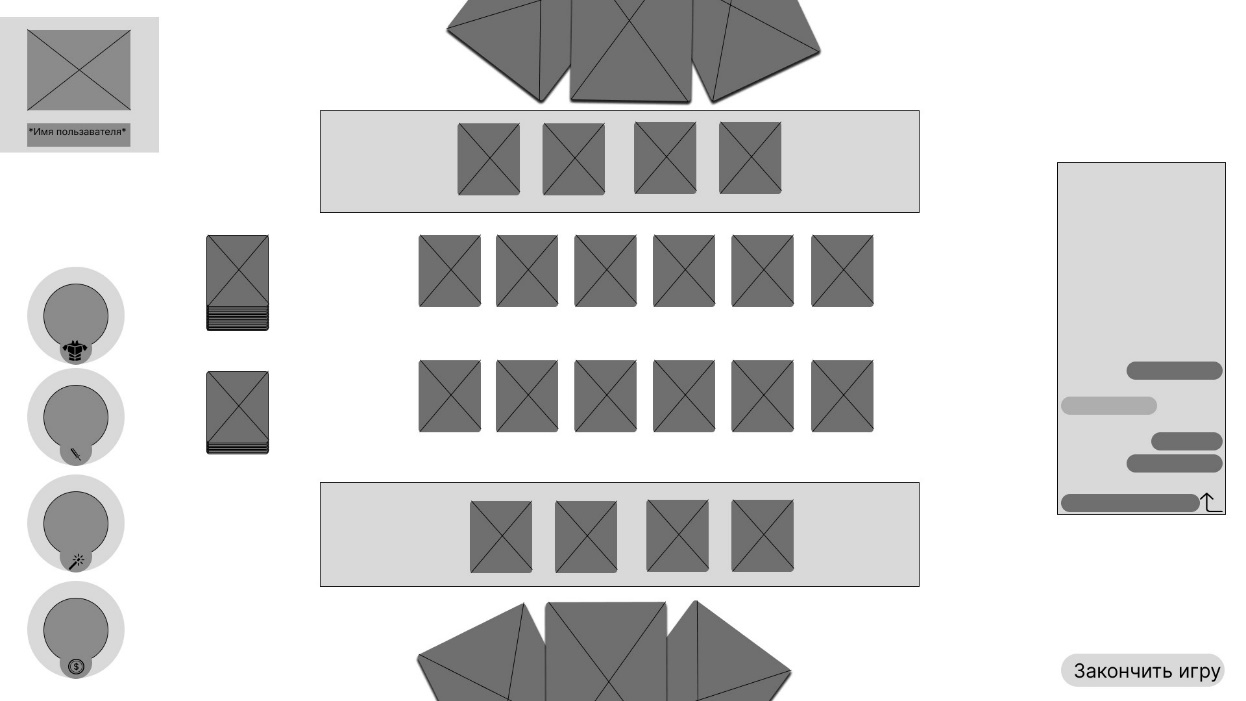


Рисунок 15 – Партия

**Заключение**

Целью данного учебного проекта являлась разработка игрового приложения «Сказки».

В ходе реализации поставленной задачи были закреплены знания по использованию Unity, а также курс дисциплины ТРПО.

Следует также учесть, что в разработанном ПП был реализован простой интерфейс, который позволяет использовать приложения любому роду пользователей.

После тщательного тестирования приложения были выявлены некоторые недоработки, которые были полностью исправлены на стадии тестирования. В целом при реализации программы были выполнены все условия, перечисленные в предыдущих разделах пояснительной записки. Таким образом можно сказать, что программа была реализована успешно.

**Список использованных источников**

1 Курс видео по обученью созданию [карточных игр](https://www.youtube.com/watch?v=fouT5jMRI2g&list=PL0isw1lWHi7v1Nz91eGEnqDIAI8y9dgoN) на платформе YouTube

2 Справочник по [unity](https://docs.unity3d.com/Manual/)

Приложение А

UML-диаграммы

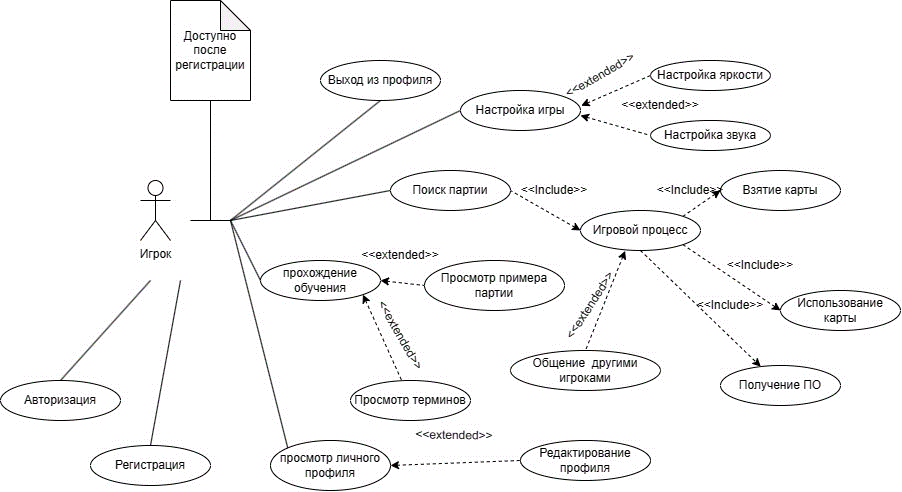


Рисунок А.1 — Диаграмма вариантов использования

Рисунок А.2 — Структура главного меню

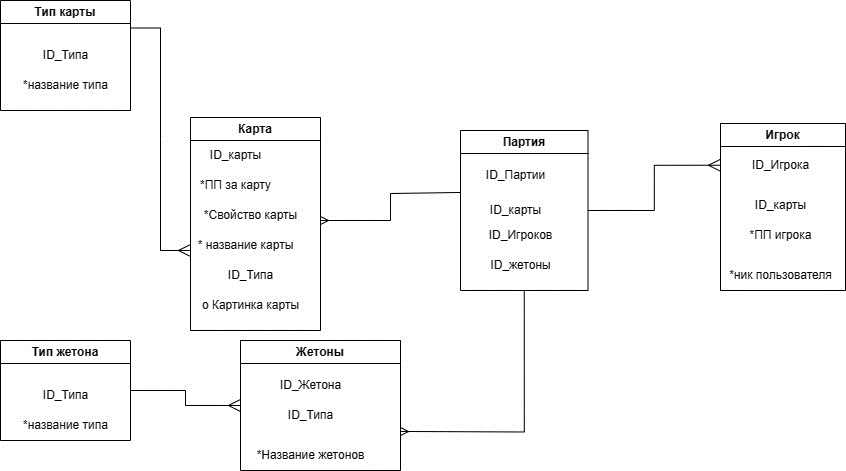


Рисунок А.3 — Модель данных

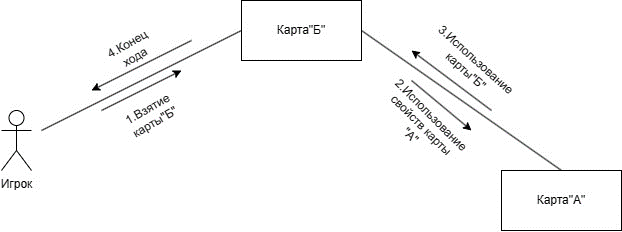


Рисунок А.4 — Диаграмма объектов

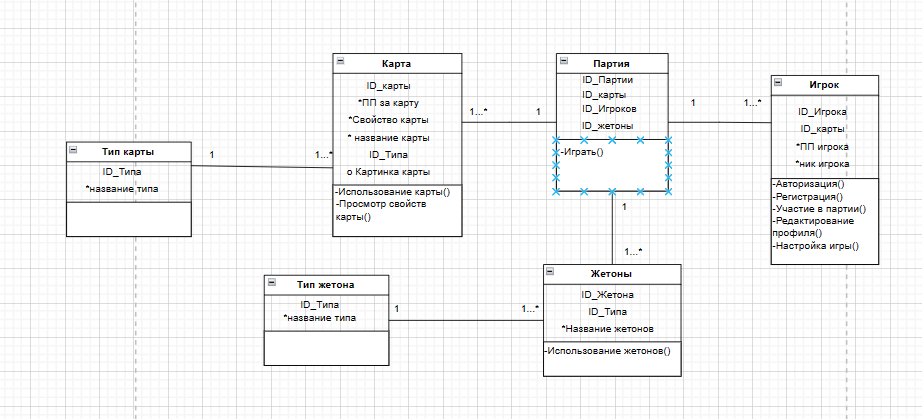


Рисунок А.5 — Диаграмма классов

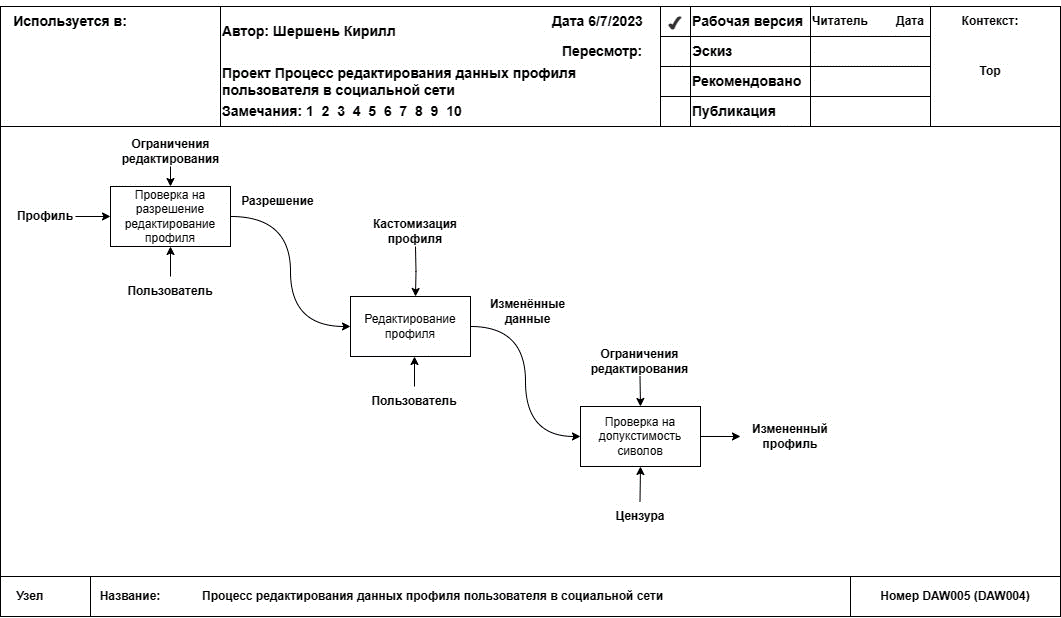
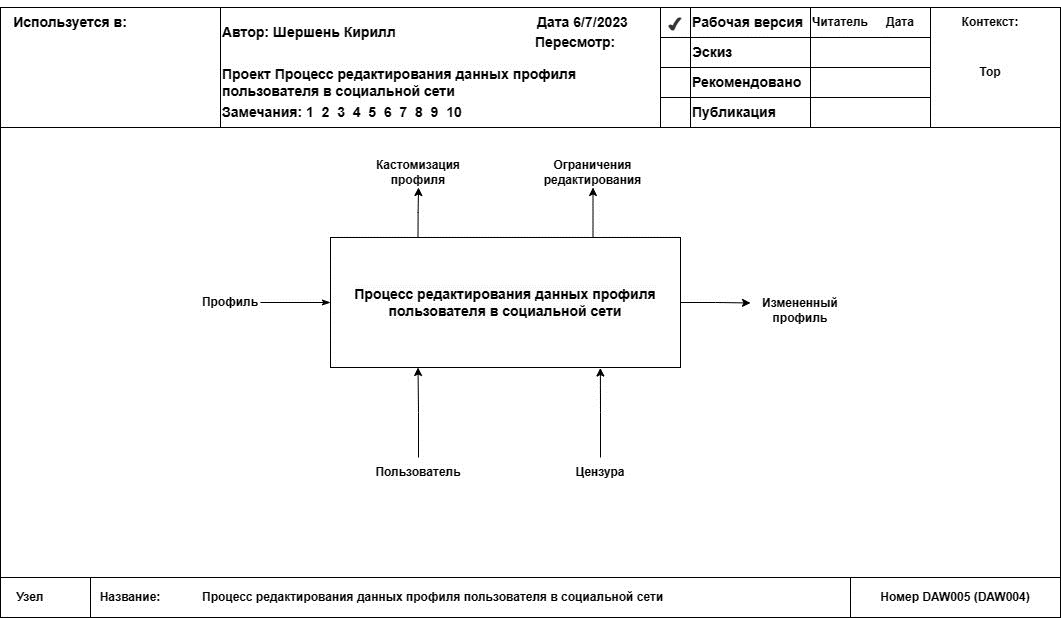


Рисунок А.6 — Функциональная модель

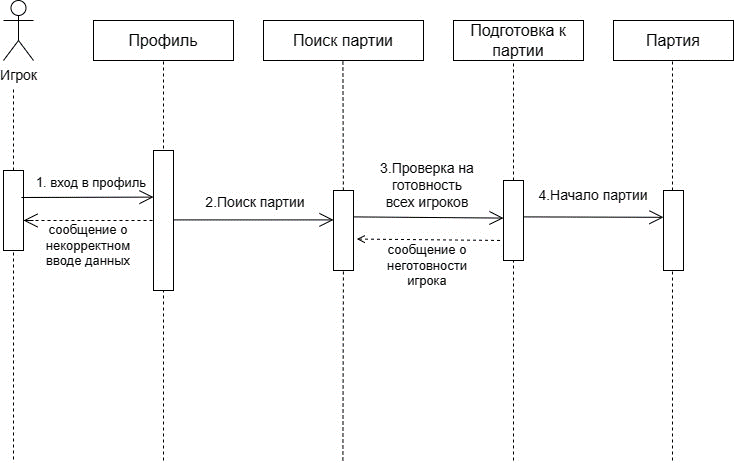


Рисунок А.7 — Диаграмма последовательности

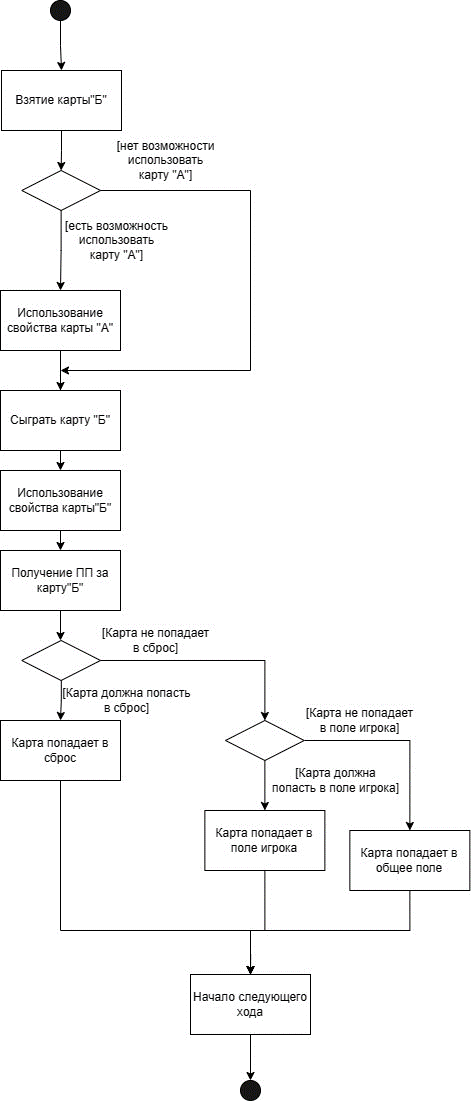


Рисунок А.8 — Диаграмма деятельности

Приложение Б

UX/UI-интерфейсы



Рисунок Б.1 — UX-интерфейс меню регистрации



Рисунок Б.2 — UX-интерфейс меню авторизации

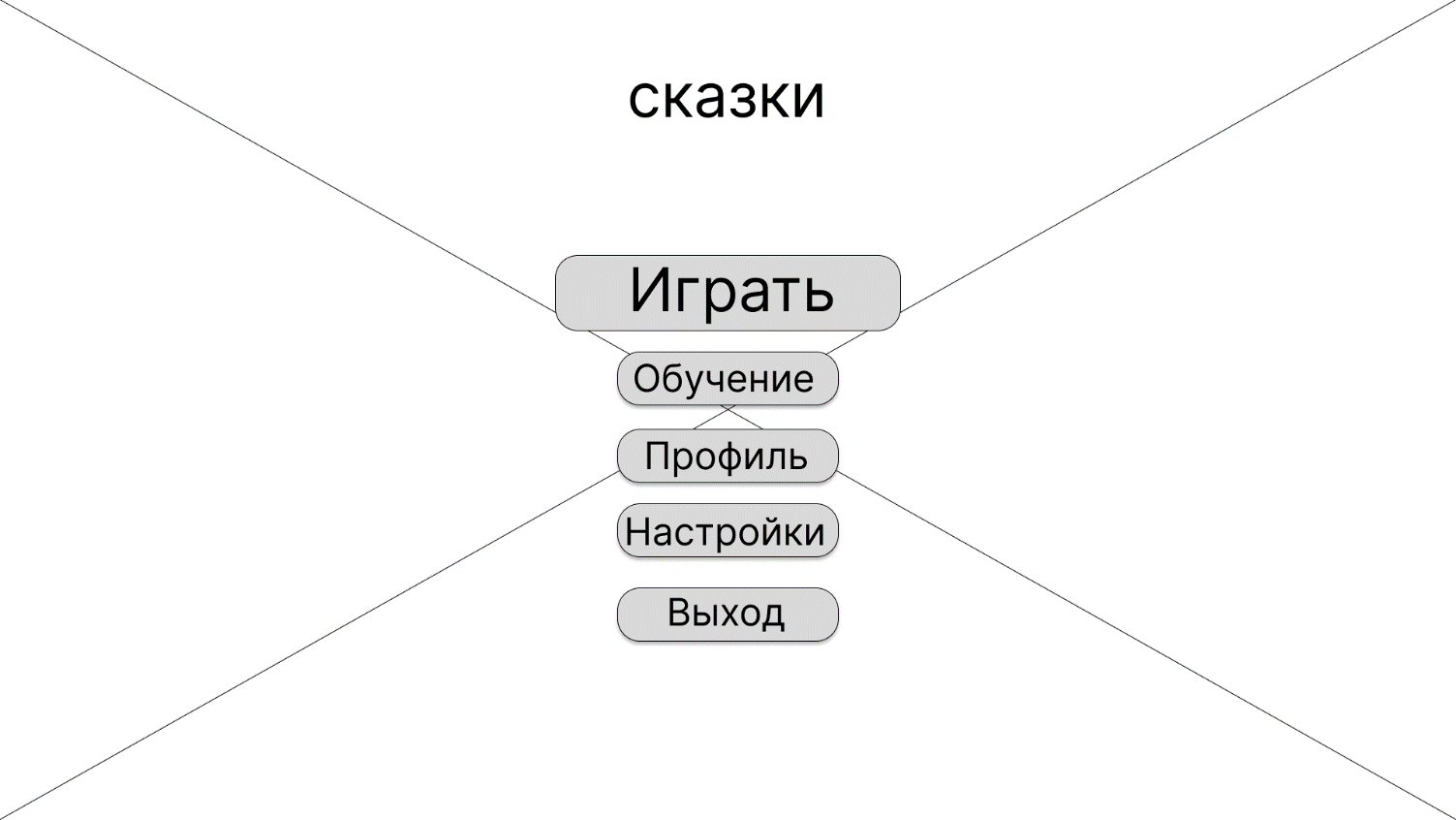


Рисунок Б.3 — UX-интерфейс главной страницы

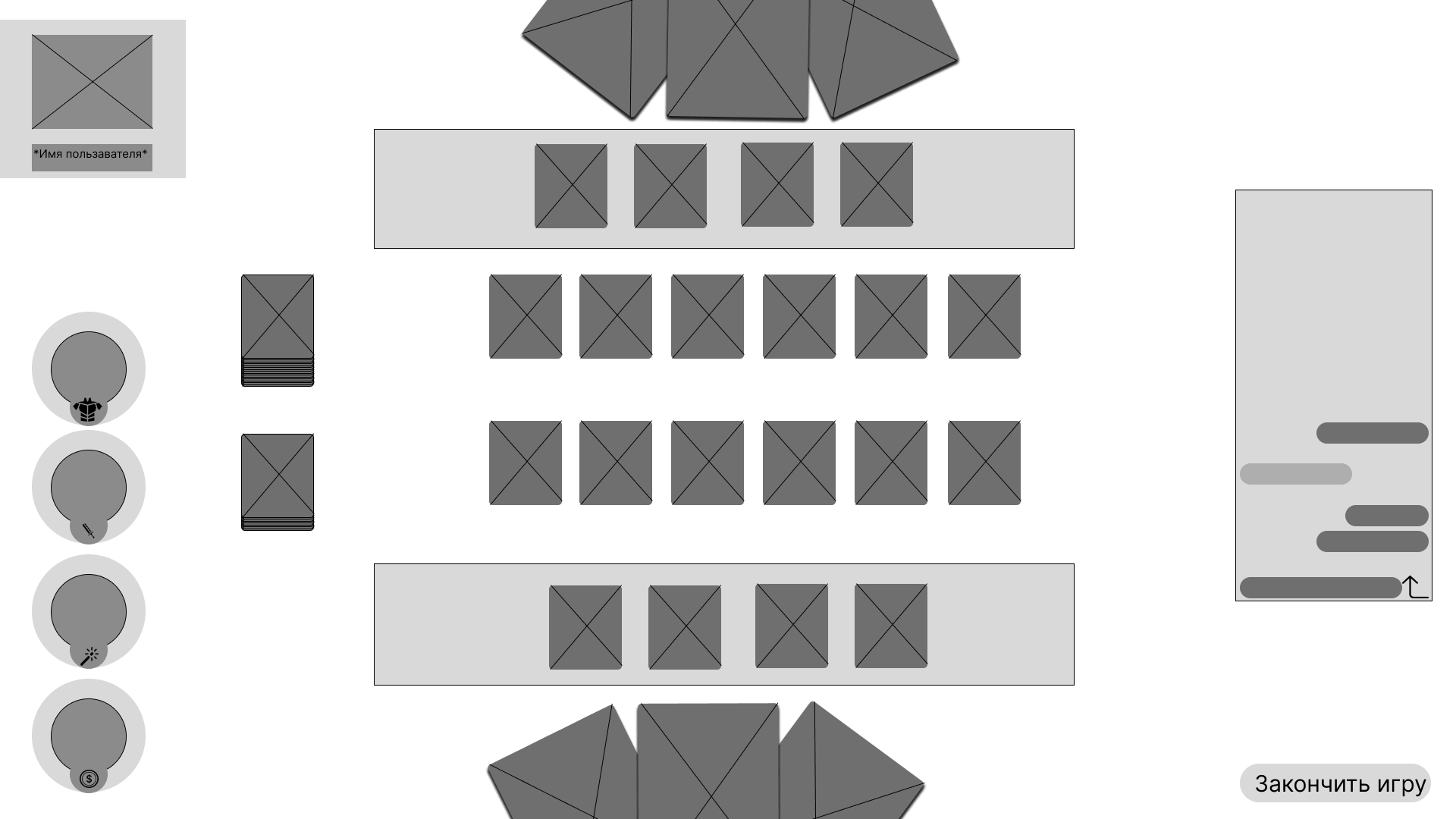


Рисунок Б.4 — UX-интерфейс партии игры

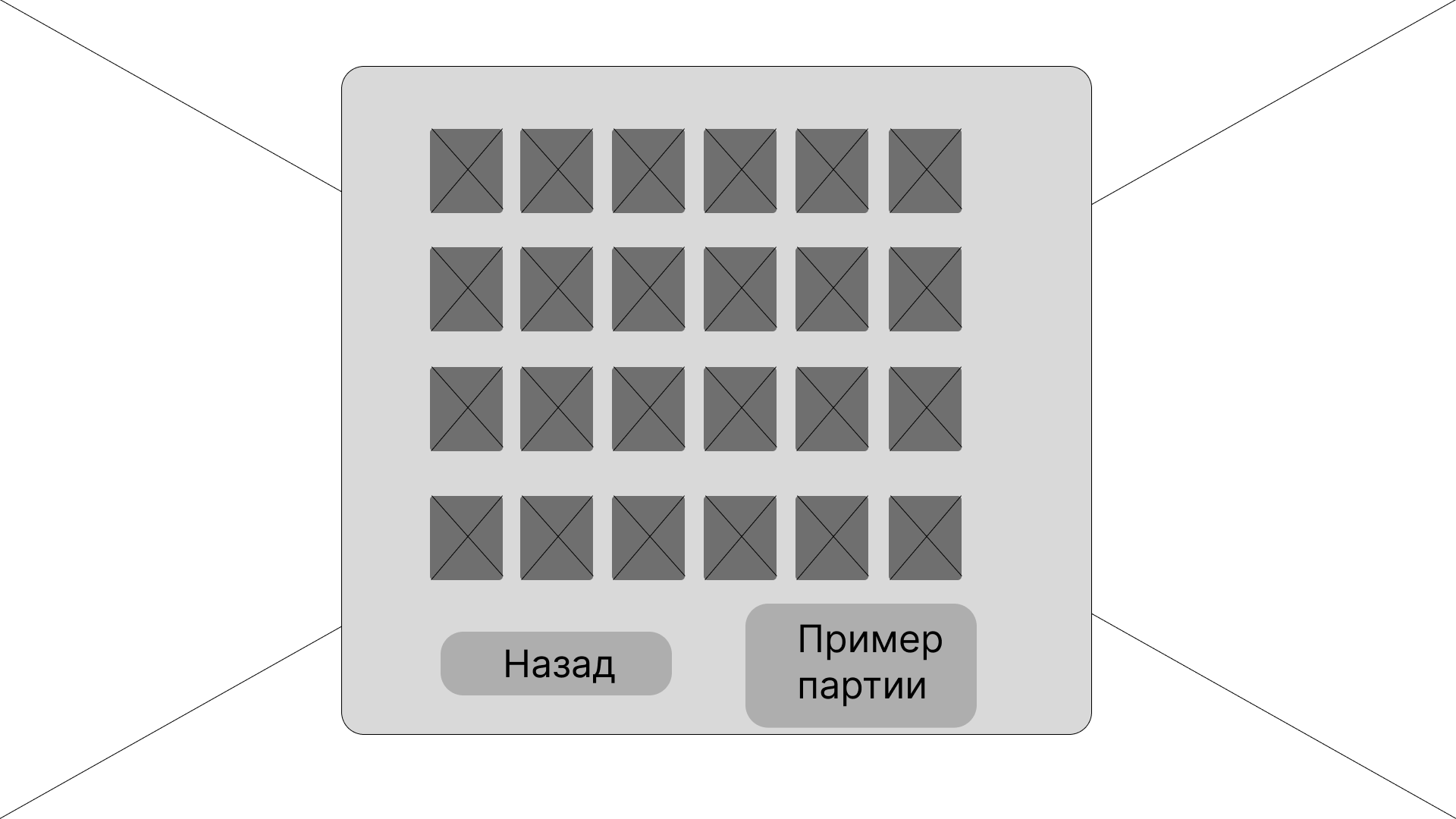


Рисунок Б.5 — UX-интерфейс страницы карт

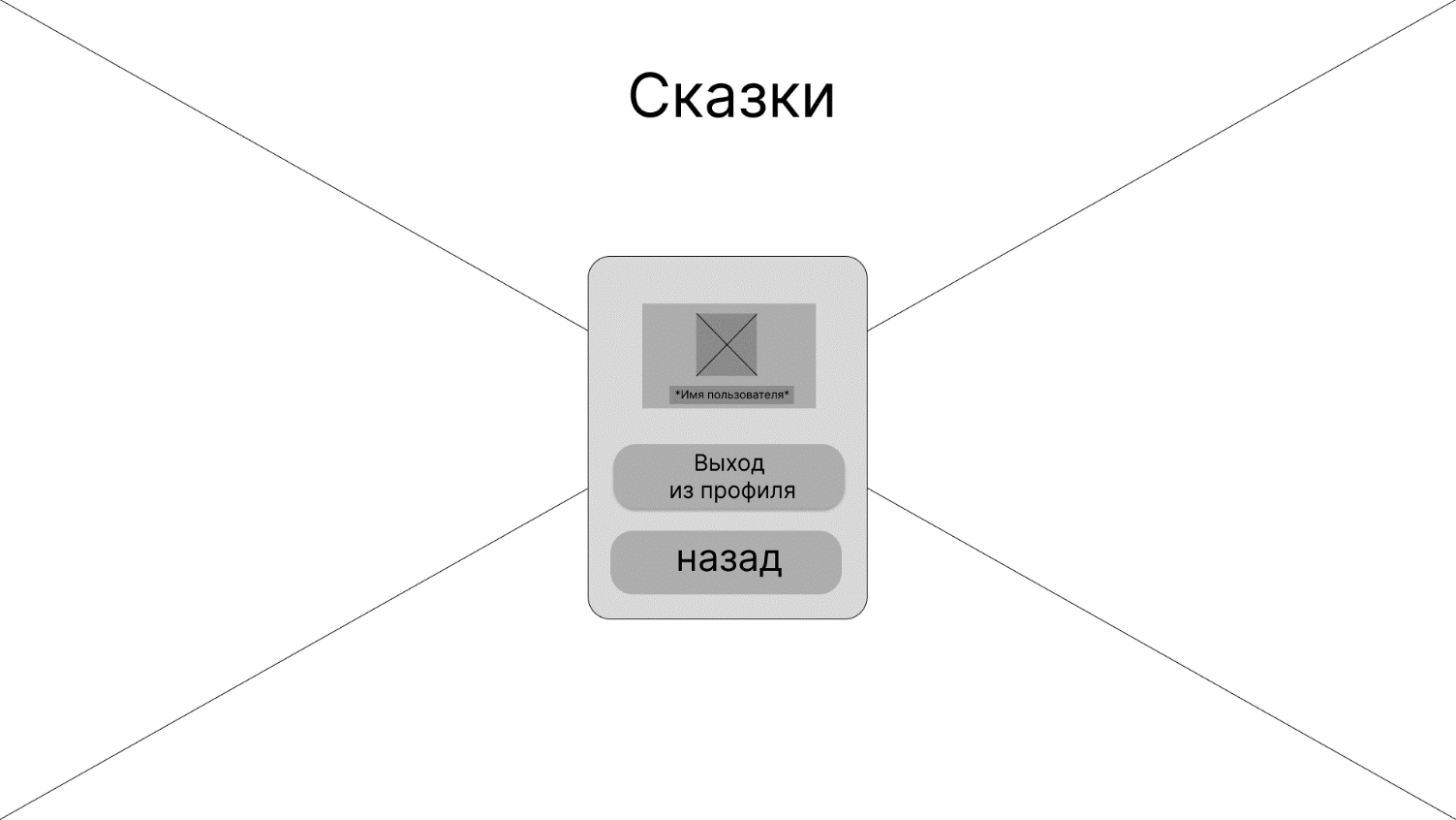


Рисунок Б.6 — UX-интерфейс кастомизации профиля



Рисунок Б.7 — UX-интерфейс настройки звука



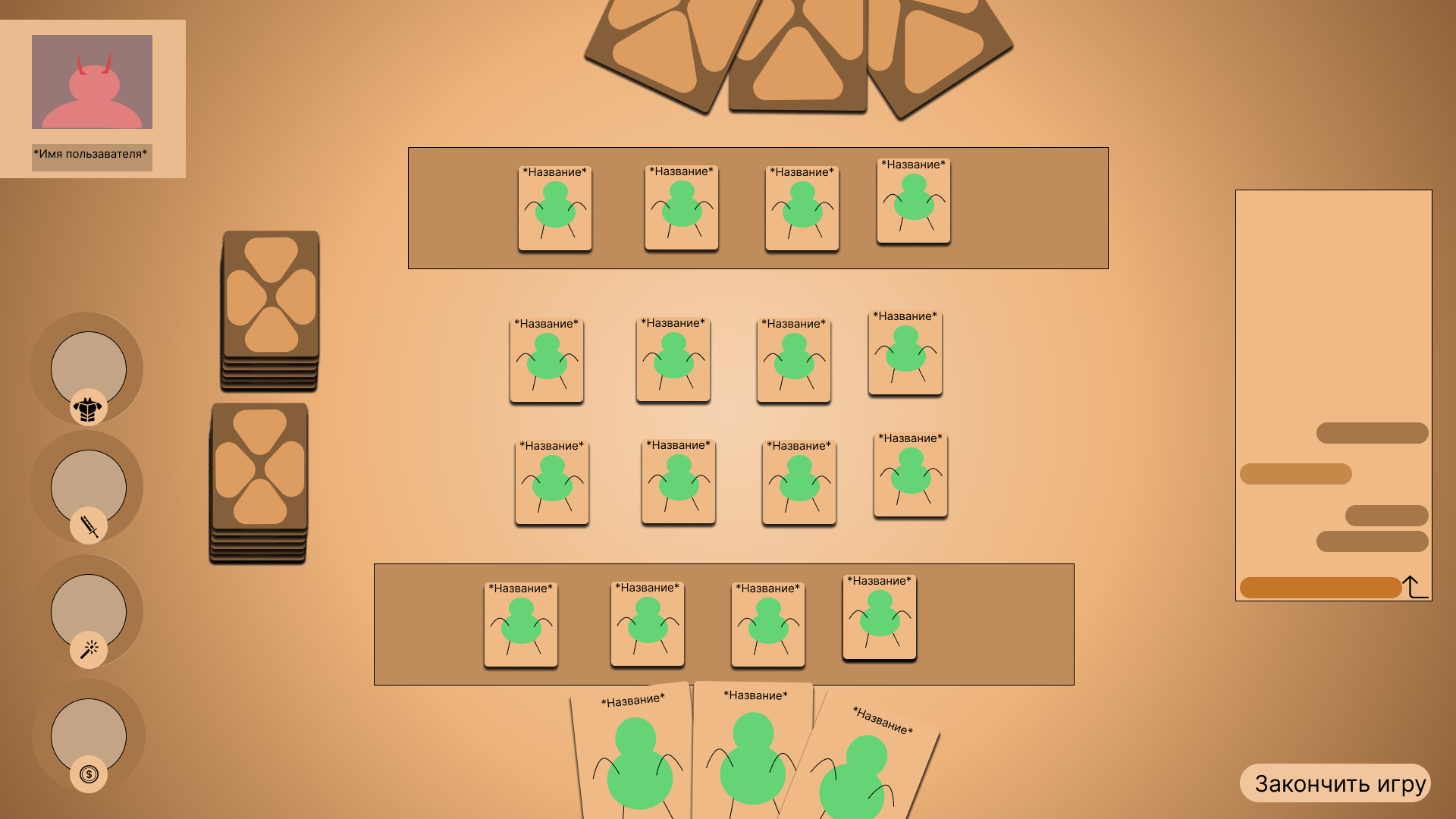
Рисунок Б.8 — UI-интерфейс главной страницы

Рисунок Б.9 — UI-интерфейс партии игры

Приложение В

Листинг

using UnityEngine;

public class BG\_MUsic : MonoBehaviour

{

[Header("Tags")]

[SerializeField] private string createdTag;

private void Awake()

{

GameObject obj= GameObject.FindWithTag(this.createdTag);

if (obj != null)

{

Destroy(this.gameObject);

}

else

{

this.gameObject.tag = this.createdTag;

DontDestroyOnLoad(this.gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using TMPro;

public class CardInfoScr : MonoBehaviour

{

public Card SelfCard;

public Image Logo;

public TextMeshProUGUI Name;

public void ShowCardInfo(Card card)

{

SelfCard = card;

Logo.sprite = card.Logo;

Logo.preserveAspect = true;

Name.text= card.Name;

}

public void Awake()

{

CardManager.AllCards.Add(new Card("Dog", "Card\_Sprite/Dog", 2, 2, 2, 2, 2));

CardManager.AllCards.Add(new Card("Wizard", "Card\_Sprite/Wizard", 2, 2, 2, 2, 2));

CardManager.AllCards.Add(new Card("Defender", "Card\_Sprite/Defender", 2, 2, 2, 2, 2));

CardManager.AllCards.Add(new Card("Woman", "", 2, 2, 2, 2, 2));

CardManager.AllCards.Add(new Card("Man", "Card\_Sprite/Man", 2, 2, 2, 2, 2));

CardManager.AllCards.Add(new Card("Dog", "Card\_Sprite/Dog", 2, 2, 2, 2, 2));

}

public void Start()

{

ShowCardInfo(CardManager.AllCards[transform.GetSiblingIndex()]);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public struct Card

{

public string Name;

public Sprite Logo;

public int Point;

public int Coin, Wand, Sword, Armor;

public Card(string name, string logopath,int points, int coin, int wand, int sword,int armor)

{

Name = name;

Logo= Resources.Load<Sprite>(logopath);

Point= points;

Coin = coin;

Wand = wand;

Sword = sword;

Armor = armor;

}

}

public static class CardManager

{

public static List<Card> AllCards = new List<Card>();

}

public class CardManagerScr : MonoBehaviour

{

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Exit : MonoBehaviour

{

public void ExitGame()

{

Application.Quit();

}

}

using UnityEngine;

public class BG\_MUsic : MonoBehaviour

{

[Header("Tags")]

[SerializeField] private string createdTag;

private void Awake()

{

GameObject obj= GameObject.FindWithTag(this.createdTag);

if (obj != null)

{

Destroy(this.gameObject);

}

else

{

this.gameObject.tag = this.createdTag;

DontDestroyOnLoad(this.gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Setting\_Game : MonoBehaviour

{

[SerializeField] private GameObject setting;

private void Start()

{

setting.SetActive(false);

}

public void OpenSettings(bool scene)

{

setting.SetActive(scene) ;

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Scene : MonoBehaviour

{

public void OpenScene(int scene)

{

SceneManager.LoadScene(scene);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class SoundVolueControl : MonoBehaviour

{

[Header("Components")]

[SerializeField] AudioSource audio;

[SerializeField] Slider slider;

[SerializeField] Text text;

[Header("Key")]

[SerializeField] private string saveVolumeKey;

[Header("tags")]

[SerializeField] private string sliderTag;

[SerializeField] private string textVolumeTag;

[Header("parameters")]

[SerializeField] private float volume;

private void Awake()

{

if (PlayerPrefs.HasKey(this.saveVolumeKey))

{

this.volume = PlayerPrefs.GetFloat(this.saveVolumeKey);

this.audio.volume = this.volume;

GameObject sliderObj = GameObject.FindWithTag(this.sliderTag);

if (sliderObj != null)

{

this.slider = sliderObj.GetComponent<Slider>();

this.slider.value = this.volume;

}

}

else

{

this.volume = 0.5f;

PlayerPrefs.SetFloat(this.saveVolumeKey, this.volume);

this.audio.volume= this.volume;

}

}

private void LateUpdate()

{

GameObject sliderObj=GameObject.FindWithTag(this.sliderTag);

if (sliderObj != null)

{

this.slider = sliderObj.GetComponent<Slider>();

this.volume = slider.value;

if (this.audio.volume != this.volume)

{

PlayerPrefs.SetFloat(this.saveVolumeKey, this.volume);

}

GameObject textObj=GameObject.FindWithTag (this.textVolumeTag);

if (textObj != null)

{

this.text =textObj.GetComponent<Text>();

this.text.text = Mathf.Round(f: this.volume \* 100) + "%";

}

}

this.audio.volume = this.volume;

}

}

using UnityEngine;

using System.Collections;

using UnityEngine.UI;

public class SceneLoader : MonoBehaviour

{

private bool loadScene = false;

[SerializeField]

private int scene;

[SerializeField]

private Text loadingText;

void Update()

{

if (Input.GetKeyUp(KeyCode.Space) && loadScene == false)

{

loadScene = true;

loadingText.text = "Загрузка...";

StartCoroutine(LoadNewScene());

}

if (loadScene == true)

{

loadingText.color = new Color(loadingText.color.r, loadingText.color.g, loadingText.color.b, Mathf.PingPong(Time.time, 1));

}

}

IEnumerator LoadNewScene()

{

yield return new WaitForSeconds(3);

AsyncOperation async = Application.LoadLevelAsync(scene);

while (!async.isDone)

{

yield return null;

}

}

}