

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет  
вычислительной техники

Кафедра  
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»  
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА  
на тему  
**БИЗНЕС-ПРОЕКТ «СИМУЛЯТОР  
СЕЛЬСКОГО ХОЗЯЙСТВА С ЭЛЕМЕНТАМИ  
РОЛЕВОЙ ИГРЫ»**

Студент

П. 4.06.2025 Гришин Сергей Вадимович  
(подпись, дата) (ФИО полностью)

Руководитель

И. 4.06.2025 Акифьев И.В.  
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

А. 5.06.2025 Бычков А.С.  
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой

М.А. Митрохин  
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

6 июня 2025 г. № 5)

Секретарь ГЭК

Е.И. Гурин  
(подпись) (фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет  
вычислительной техники

Кафедра  
«Вычислительная техника»

«Утверждаю»  
Заведующий кафедрой  
М.А. Митрохин  
« 6 » мая 2025 г.

ЗАДАНИЕ  
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Гришин Сергей Вадимович гр. 21BBB1 факультета ФВТ

2. Тема работы Бизнес-проект "Симулятор сельского хозяйства с элементами ролевой игры"

Тема утверждена приказом ПГУ № 601/о от " 6 " мая 2025 г.

3. Руководитель бакалаврской работы Акифьев И.В. 

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать симулятор сельского хозяйства с элементами ролевой игры. Язык программирования C#, среда разработки Visual Studio Code, игровой движок Unity. Игра должна обладать понятным интерфейсом и использовать следующие технологии:

1. Mirror

2. DoTween

3. URP2D

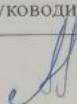
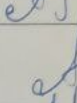
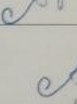
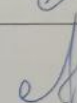
4. \_\_\_\_\_

5. \_\_\_\_\_

6. \_\_\_\_\_

5. Перечень подлежащих разработке вопросов

6. Календарный график выполнения работы

| № п/п | Наименование этапов работы                                    | Объем работы | Срок выполнения         | Подпись руководителя                                                                |
|-------|---------------------------------------------------------------|--------------|-------------------------|-------------------------------------------------------------------------------------|
| 1     | Изучение предметной области                                   | 32           | 26.04.2025 – 28.05.2025 |  |
| 2     | Анализ требований                                             | 32           | 29.04.2025 – 30.05.2025 |  |
| 3     | Создание макета графического интерфейса                       | 30           | 31.04.2025 – 3.05.2025  |  |
| 4     | Выбор технологий, применяемых при проектировании и разработке | 40           | 4.05.2025 – 8.05.2025   |  |
| 5     | Реализация графического интерфейса                            | 66           | 9.05.2025 – 15.05.2025  |  |
| 6     | Разработка основной части программы                           | 70           | 16.05.2025 – 23.05.2025 |  |
| 7     | Оформление пояснительной записки                              | 54           | 24.05.2025 – 31.05.2025 |  |

Дата выдачи задания " 6" мая 2025 г.

Руководитель бакалаврской работы  6.05.25 Акифьев И.В.  
(подпись, дата)

Задание к исполнению принял студент  6.05.25 Гришин С.В.  
(подпись, дата)

Работу к защите допустить  
Декан факультета  06.06.2025 Фионова Л.Р.  
(подпись, дата)

## Реферат

Бакалаврская работа содержит 141 листов, 17 рисунков, 1 таблицу, 6 источников и 6 приложений.

ИГРА, UNITY, MIRROR, DOTWEEN, TOPDOWNRPG

Цель работы – разработать симулятор сельского хозяйства с элементами ролевой игры с PixelART графикой. Игра должна обеспечить высокую реиграбельность и возможность кооперативного прохождения.

Объектами разработки являются:

- игровое приложение с реализацией основных механик, характерных для данного жанра;
- мультиплеерная составляющая;
- визуальная составляющая (пользовательский интерфейс).

В результате дипломного проектирования было разработано игровое приложение в жанре TopDownRPG, имеющее возможность игры в кооперативе до 4 человек, архитектуру, заточенную под долгосрочную поддержку и обновления приложения, а также определена система монетизации данного проекта.

|           |      |               |         |         |                                                                                                        |                            |      |        |  |
|-----------|------|---------------|---------|---------|--------------------------------------------------------------------------------------------------------|----------------------------|------|--------|--|
|           |      |               |         |         | ПГУ 1.090301.05 001 ПЗ                                                                                 |                            |      |        |  |
| Изм       | Лист | № докум.      | Подпись | Дата    | Бизнес-проект «Симулятор<br>сельского хозяйства с<br>элементами ролевой игры»<br>Пояснительная записка | Лит.                       | Лист | Листов |  |
| Разработ. |      | Гришин С.В.   |         | 4.06.25 |                                                                                                        |                            |      |        |  |
| Проверил  |      | Акифьев И.В.  |         | 4.06.25 |                                                                                                        |                            | 2    | 116    |  |
| Н. контр. |      | Бычков А.С.   |         | 5.06.25 |                                                                                                        |                            |      |        |  |
|           |      |               |         |         |                                                                                                        |                            |      |        |  |
| Утвердил  |      | Митрохин М.А. |         | 5.06.25 |                                                                                                        |                            |      |        |  |
|           |      |               |         |         |                                                                                                        | ФВТ, каф. ВТ<br>гр. 21BVB1 |      |        |  |

## Содержание

|                                                                            |    |
|----------------------------------------------------------------------------|----|
| Перечень сокращений и обозначений.....                                     | 5  |
| 1 Стартап-проект симулятор сельского хозяйства с элементами ролевой игры.. | 6  |
| 1.1 Анализ предметной области .....                                        | 6  |
| 1.1.1 Актуальность разработки.....                                         | 6  |
| 1.1.2 Анализ существующих аналогов.....                                    | 8  |
| 1.2 Постановка задачи.....                                                 | 14 |
| 1.2.1 Цель проекта.....                                                    | 14 |
| 1.2.2 Функционал игры .....                                                | 14 |
| 1.2.3 Требования к аппаратному обеспечению .....                           | 15 |
| 1.2.4 Инструменты разработки .....                                         | 16 |
| 1.3 Выбор технологий и инструментов разработки .....                       | 16 |
| 1.3.1 Игровой движок.....                                                  | 16 |
| 1.3.2 Язык программирования .....                                          | 18 |
| 1.3.3 Среда разработки.....                                                | 18 |
| 1.3.4 Графический редактор.....                                            | 18 |
| 1.3.5 Выбор дополнительных ассетов .....                                   | 19 |
| 2 Проектирование и разработка .....                                        | 21 |
| 2.1 Описание архитектуры игры .....                                        | 21 |
| 2.2 Подготовка рабочей среды.....                                          | 23 |
| 2.3 Подготовка сцен .....                                                  | 24 |
| 2.3.1 Главное меню.....                                                    | 24 |
| 2.3.2 Главная сцена.....                                                   | 25 |
| 2.4 Создание игрока .....                                                  | 29 |
| 2.4.1 Внешний вид.....                                                     | 29 |
| 2.4.2 Взаимодействие с миром.....                                          | 30 |
| 2.4.3 Система магических навыков .....                                     | 32 |
| 2.4.4 Управление.....                                                      | 33 |
| 2.5 Графика .....                                                          | 34 |

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 3    |

|                                                                |     |
|----------------------------------------------------------------|-----|
| 2.6 Пользовательский интерфейс .....                           | 35  |
| 2.7 Управление звуками .....                                   | 36  |
| 2.8 Инвентарь .....                                            | 40  |
| 2.9 Игровая локация .....                                      | 43  |
| 2.10 Мультиплеер .....                                         | 45  |
| 3 Тестирование и отладка .....                                 | 47  |
| 4 Безопасность жизнедеятельности .....                         | 48  |
| 5 Стартап проект .....                                         | 50  |
| 5.1 Целевая аудитория .....                                    | 50  |
| 5.2 Источники дохода и аудитории .....                         | 50  |
| 5.3 Риски .....                                                | 51  |
| Заключение .....                                               | 53  |
| Список использованных источников .....                         | 54  |
| Приложение А – UML – диаграммы .....                           | 55  |
| Приложение А.1 – UML – диаграмма вариантов использования ..... | 56  |
| Приложение А.2 – UML – диаграмма классов .....                 | 57  |
| Приложение А.3 – UML – диаграмма состояний .....               | 58  |
| Приложение А.4 – UML – диаграмма взаимодействия .....          | 59  |
| Приложение Б – Листинг .....                                   | 60  |
| Приложение В – Презентация .....                               | 133 |

## Перечень сокращений и обозначений

В дипломной работе применяются следующие сокращения и обозначения  
FPS (Frames Per Second) – количество кадров в секунду, характеристика производительности

PixelART – стиль графики, основанный на пиксельном изображении.

UI (User Interface) – пользовательский интерфейс

Unity – игровой движок

PixelART – стиль графики, основанный на пиксельном изображении

UML (Unified Modeling Language) – унифицированный язык моделирования

URP (Universal Render Pipeline) – система обработки графики в Unity

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 5    |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

# 1 Стартап-проект симулятор сельского хозяйства с элементами ролевой игры

## 1.1 Анализ предметной области

### 1.1.1 Актуальность разработки

Проведенный анализ текущего состояния российской индустрии разработки видеоигр выявил ряд проблем, обусловленных акцентом на создание низкокачественных проектов, ориентированных на быструю прибыль. Данная тенденция препятствует развитию отрасли и ее конкурентоспособности на международном уровне.

По данным исследований компании HackerRank, Россия располагает значительным потенциалом в области программирования, включая игровую разработку. Однако, в отличие от других стран, таких как Китай, США и Япония, где ежегодно выпускаются сотни инновационных и качественных игровых проектов (330 проектов за 2024 год), в России за аналогичный период отмечается значительное отставание (около 12 проектов за 2024 год, уступающих по качеству зарубежным аналогам).

Примерами неудачных проектов, вызвавших негативную реакцию игрового сообщества, являются “Смута” и “The day before”. Первый проект, несмотря на большие ожидания, получил низкие оценки. “Смута” набрала 1,2 балла из 5 по версии журнала StopGame из-за проблем с сюжетом, оптимизацией и графической составляющей. Второй проект продемонстрировал некачественную реализацию и чрезмерное использование готовых ресурсов, что привело к закрытию проекта и негативному отношению к разработчикам.

Анализ инди-сегмента показывает, что независимые разработчики, как и крупные студии, сосредотачиваются на создании проектов с целью получения быстрой прибыли. В результате, большинство игр не обладают художественной ценностью и часто демонстрируют низкое качество исполнения. Примером являются многочисленные игры на платформе “Яндекс игры” и подобные ей.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 6    |

В настоящее время видеоигры представляют собой не только развлечение, но и форму искусства, сочетающую элементы литературы, кинематографа, музыки и других видов творчества. Разработчики создают уникальные миры и глубокие сюжеты, сопровождая их качественным визуальным и музыкальным оформлением. Международное признание качества игровых проектов подтверждается наличием престижных наград, таких как Game Awards. Однако, российская индустрия в основном представлена проектами, не отвечающими современным стандартам качества, что может негативно сказаться на культурном имидже страны.

Ситуация усугубляется введением санкций, что снижает конкуренцию для российских разработчиков. Ограничение доступа к зарубежным играм на таких платформах, как Steam, может привести к дальнейшему снижению качества отечественных игровых продуктов.

На основании вышеизложенного можно заключить, что существует острая необходимость в изменении отношения российских разработчиков и игроков к качеству игровых продуктов. Данный проект является первым шагом в этом направлении, нацеленным на создание игры, превосходящей среднестатистические отечественные аналоги.

Данный проект ориентирован на широкую аудиторию, в первую очередь на возрастную группу 16-25 лет, но также может привлечь и более старших игроков. Основными потребностями этой аудитории являются:

- развлечение и отдых;
- самореализация и прогресс;
- социальное взаимодействие (для многопользовательского режима);
- исследование и открытие нового;
- творчество и самовыражение.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 7    |

### 1.1.2 Анализ существующих аналогов

Аналоги можно разделить на прямые и косвенные. Прямые аналоги представляют собой игры, которые имеют схожий жанр и основные механики с разрабатываемым проектом.

Первый аналог это Stardew valley.

Жанр: симулятор жизни с элементами RPG.

Достоинства:

- большое разнообразие занятий;
- приятная графика.

Недостатки:

- обрывистый сюжет;
- полноценные обновления больше не выходят;
- большое количество механик, но все имеют простую реализацию;
- плохой баланс.

Является самым популярным представителем жанра. Разрабатывался одним человеком на протяжении 4 лет с 2012 года. Изредка получает обновления, направленные на улучшения графики и исправление багов.

Игровой процесс включает в себя множество занятий, таких как фермерство, рыбалка, улучшения фермы, социальные взаимодействия, но все они реализованы максимально просто. Прогресс игрока привязан к улучшению клуба, для которого нужны определённые ресурсы, получаемые при определённых условиях, например, ко времени года. Из-за этого, если игрок не хочет сильно отстать в прогрессе, он вынужден торопиться и продумывать каждый свой шаг, что превращает размеренный и медитативный геймплей в быстрый и напряжённый. Так же всё это усугубляется из-за плохого баланса внутриигровых предметов.

На рисунке 1 представлен кадр из Stardew valley.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 8    |



Рисунок 1 – Кадр из игры «Stardew valley»

Второй аналог это Harvest moon.

Жанр: симулятор жизни с акцентом на выращивание культур.

Достоинства:

- спокойный геймплей;
- приятная графика.

Недостатки:

- упрощённая система развития;
- малое разнообразие занятий.

Является одним из основателей жанра симулятора жизни, однако на сегодняшний день потерял свою популярность. Игровой процесс похож на Stardew valley, но сильно упрощённый. Графика игра достаточно приятная, но уже считается устаревшей.

На рисунке 2 представлен кадр из Harvest moon.



Рисунок 2 – Кадр из игры «Harvest moon»

Третий аналог это My time at Sandrock.

Жанр: симулятор жизни с элементами RPG.

Достоинства:

- большой открытый мир;
- интересная система улучшения города.

Недостатки:

- плохой баланс;
- затянутый игровой процесс.

В отличие от выше представленных игр, игровой процесс основан не на фермерстве, а на постройке своей мастерской и улучшении города. Мир игры интересен, его хочется исследовать, но со второй половины игры, локации начинают повторяться, а процесс получения определённых предметов искусственно усложнён. В результате чего, начиная со второй половины игры, весь игровой процесс кажется слишком затянутым и быстро наскучивает.

На рисунке 3 представлен кадр из My time at Sandrock.



Рисунок 3 – Кадр из игры «My time at Sandrock»

Косвенные аналоги представляют собой игры, которые имеют общие элементы с разрабатываемым проектом, но относятся к другим жанрам.

Четвёртым аналогом является Animal crossing.

Жанр: симулятор жизни с акцентом на социальное взаимодействие.

Достоинства:

- расслабляющий геймплей;
- приятная графика.

Недостатки:

- малое разнообразие занятий и простая реализация имеющихся;
- привязка к интернету.

Спокойная и медитативная игры. Основное упор делается на социальные взаимодействия и дополнительные активности. Но главной проблемой является слишком простая реализация всех механик игры. Они могут зацепить игрока, но не могут удержать его надолго. В результате, разработчикам нужно выпускать регулярные обновления и события, чтобы удерживать внимание игроков.

На рисунке 4 представлен кадр из Animal crossing.



Рисунок 4 – Кадр из игры «Animal crossing»

Косвенным аналогом является Terraria.

Жанр: 2D-экшен-платформер с элементами RPG.

Достоинства:

- большой и разнообразный мир;
- хороший баланс механик.

Недостатки:

- сложная система крафта;
- высокий порог вхождения.

Представляет из себя 2D RPG с открытым миром, что является основной фишкой игры. В ней огромное число занятий, которые могут увлечь игрока, но игра не даёт никакой инструкции к самой себе. Новичок, который впервые зашёл в игру не сможет пройти её без посторонней помощи.

На рисунке 5 представлен кадр из Terraria.



Рисунок 5 – Кадр из игры «Terraria»

Таблица 1 - Анализ аналогов

| Название            | Особенности                                     | недостатки                                 |
|---------------------|-------------------------------------------------|--------------------------------------------|
| Stardew valley      | Приятная атмосфера, много разнообразных занятий | Жёсткая привязка ко времени, плохой баланс |
| Harvest moon        | Спокойный игровой процесс                       | Малое разнообразие занятий                 |
| My time at Sandrock | Большой открытый мир                            | Затянутый игровой процесс                  |
| Animal crossing     | Спокойный игровой процесс                       | Простая реализация механик                 |
| Terraria            | разнообразный мир, хорошая механика боя         | высокий порог вхождения                    |

Анализ аналогов показывает, что существует спрос на игры, сочетающие элементы симулятора жизни и RPG. Успешные проекты в этом жанре предлагают игрокам глубокий и разнообразный геймплей, интересные

истории и персонажей, а также возможность для творчества и самовыражения.

Так же можно заметить, что в сегменте российских инди-игр нет известных проектов в этом жанре.

Разрабатываемый проект должен стремиться к улучшению недостатков, выявленных в аналогах, таких как однообразный геймплей, ограниченная кастомизация и упрощенная боевая система, плохой баланс и слишком простая реализация доступных занятий. Важно также учитывать особенности целевой аудитории и адаптировать геймплей под ее потребности.

## 1.2 Постановка задачи

### 1.2.1 Цель проекта

В рамках дипломного проекта разрабатывается «Симулятора сельского хозяйства с элементами ролевой игры» с рабочим названием «The little story».

### 1.2.2 Функционал игры

Игра представляет собой симулятор жизни с элементами RPG, в котором игрок управляет персонажем, попавшим в неизведанный мир. В релизной версии должны быть реализованы следующие основные функциональные возможности:

- фермерство (базовая реализация) – выращивание одного-двух видов культур;
- добыча ресурсов (упрощенная реализация) – сбор руды в одной шахте с ограниченным количеством уровней;
- боевая система (базовая реализация) – сражения с одним-двумя видами монстров с использованием одного типа оружия. Прокачка боевых навыков отсутствует или сильно упрощена;
- прокачка персонажа (базовая) – повышение уровня игрока, получение нескольких базовых навыков;
- социальное взаимодействие (базовое) – взаимодействие с несколькими NPC;

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 14   |

- инвентарь – система управления предметами;
- магические способности – использование базовых заклинаний;
- мультиплеер – возможность игры по локальной сети.

На рисунке 6 изображена диаграмма использования приложения игроком.

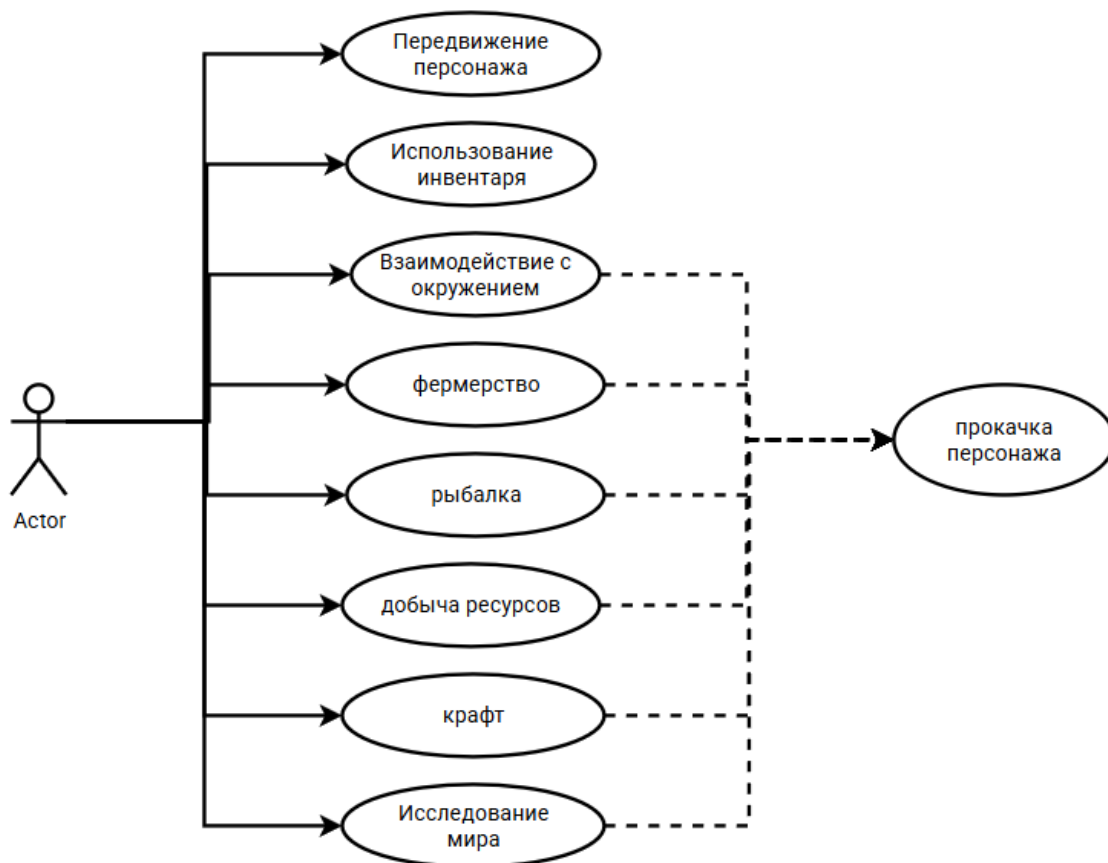


Рисунок 6 – Диаграмма вариантов использования приложения

### 1.2.3 Требования к аппаратному обеспечению

Минимальные требования к аппаратному обеспечению:

- операционная система: Windows 10;
- процессор: Intel Core i3 3-го поколения;
- оперативная память: 2 ГБ;
- видеокарта: intel HD Graphics 5500;
- место на диске: 1 ГБ.

Рекомендуемые требования к аппаратному обеспечению:

- операционная система: Windows 10;
- процессор: Intel Core i5 3-го поколения;
- оперативная память: 4 ГБ;
- видеокарта: NVIDIA GeForce GT 1030;
- место на диске: 1 ГБ.

#### 1.2.4 Инструменты разработки

Инструменты, использованные при разработке:

- игровой движок: Unity;
- язык программирования: C#;
- графический редактор: Aseprite;
- редактор звука: Reaper;
- система контроля версий: Git.

### 1.3 Выбор технологий и инструментов разработки

В Этом подразделе будут описаны технологии и средства разработки, выбранные для разработки.

Список выбранных технологий и средств разработки:

- Unity;
- C#;
- Visual studio 22;
- Aseprite;
- Mirror;
- DoTween.

#### 1.3.1 Игровой движок

На данный момент для создания игр используются 3 самых популярных игровых движка:

- Unity;
- Unreal Engine 5;

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 16   |

– Godot.

Unreal Engine 5 – это мощный игровой движок, разработанный Epic Games, который широко используется для создания высококачественных 3D-игр, виртуальной реальности и кинематографических проектов. Он отличается передовыми технологиями, такими как Lumen для динамического освещения и Nanite для детализированной геометрии. Unreal Engine 5 поддерживает визуальное программирование через Blueprint, что делает его доступным для разработчиков разного уровня. Однако он требует значительных ресурсов и может быть сложным для новичков. Так же, из-за ориентированности на 3D проекты, инструментарий для создания 2D-игр у него очень ограничен.

Godot – это лёгкий и открытый игровой движок, который идеально подходит для простых 2D-игр и небольших 3D-проектов. Он отличается простым интерфейсом и встроенным языком GDScript, который похож на Python. Godot не требует мощного оборудования и поддерживает C# и C++, но у него меньше возможностей для создания сложных игр по сравнению с Unreal Engine 5 и Unity.

Unity – это универсальный игровой движок, который подходит для разработки 2D и 3D-игр, мобильных приложений, а также VR/AR-проектов. Unity известен своей простотой и гибкостью, что делает его популярным среди инди-разработчиков и крупных студий. Движок использует язык C# и предлагает множество готовых ассетов и инструментов, но его графика уступает Unreal Engine 5, а производительность в сложных 3D-проектах может быть нестабильной. Несмотря на это, движок имеет очень удобный инструментарий для создания больших 2D-игр.

Получается, что Unreal Engine 5 не подходит для создания 2D-игр, Godot не приспособлен для создания сложных проектов, а unity имеет большую библиотеку готовых ассетов, а также удобный встроенный функционал для создания больших 2D проектов, поэтому именно он был выбран для разработки.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 17   |

### 1.3.2 Язык программирования

Unity позволят писать на C# и JavaScript, но больший функционал доступен именно на C#. Помимо этого, это очень удобный объектно-ориентированный язык программирования, который позволяет быстро писать хорошо читаемый код, что улучшит долгосрочную поддержку проекта. Поэтому именно C# был выбран для разработки.

### 1.3.3 Среда разработки

Так как язык программирования и движок уже выбраны, а именно C# и Unity. В данной ситуации в качестве среды разработки подходит один вариант и это Visual Studio. Была выбрана версия 2022 года, так как лично для меня это наиболее удобная версия, но в целом для разработки данного проекта можно взять любую версию старше Visual Studio 2019.

### 1.3.4 Графический редактор

Так как данный проект представляет из себя 2D-игру в пиксельном стиле, то и редактор должен поддерживать работу с растровыми изображениями с возможностью экспорта в png.

На данный момент существует множество подобных редакторов, например Adobe photoshop, krita, SPT и так далее. Но у большинства подобных редакторов есть один минус, они заточены под создание изображения с большим разрешением и создание пиксельных спрайтов в них может быть трудным. Поэтому мы сразу их исключаем и рассматриваем только варианты, которые заточены именно под создание пиксель артов.

Pixelorama – программа с открытым исходным кодом, бесплатная и с полным доступом ко всем обновлениям. Удобно, что у неё есть веб-версия, которую можно запустить прямо в браузере без скачивания на ПК. В ней можно рисовать прямо во время проигрывания анимации, создавать идеальные линии в специальном режиме, пользоваться предустановленными цветовыми палитрами и различными эффектами обработки.

Ruxel Edit – в первую очередь это хороший редактор тайлов, вместе с этим способный на создание анимаций. Ruxel Edit позволяет редактировать тайлы

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 18   |

сразу в готовой картинке и смотреть, как они работают вместе. А после — экспортировать тайлсет и грузить его напрямую в проект, что очень полезно при разработке игры с повторяющимися паттернами в окружении.

Aseprite – редактор хорошо подходит как для профессионалов, так и для начинающих: он довольно понятный и вдобавок популярный, по нему есть немало уроков в свободном доступе. Ещё Aseprite можно получить бесплатно, если скомпилировать программу из открытого исходного кода. По отдельным взятым особенностям не превосходит Pixelorama и Puxel Edit, но имеет сразу все эти особенности и не уступает в них конкурентам.

По итогу, Aseprite имеет наиболее обширный функционал, множество видеоуроков и возможность получить программу бесплатно, поэтому в качестве графического редактора был выбран именно он.

### 1.3.5 Выбор дополнительных ассетов

Как говорилось выше, unity очень гибкий движок, который, в сочетании с C#, позволяет написать 2D-игру любой сложности, но, для ускорения процесса разработки, были подключены два набора готовых ассетов, а именно Mirror и DoTween.

Mirror – это высокоуровневая сетевая библиотека, созданная для упрощения разработки мультиплеерных игр в Unity. Она является форком старой системы UNET, но с улучшенной производительностью, стабильностью и удобством.

Ключевые особенности:

- поддержка клиент-серверной архитектуры;
- встроенные компоненты для синхронизации игровых объектов (например, NetworkTransform, NetworkManager);
- гибкость: можно использовать как для простых, так и для сложных сетевых проектов.

Преимущества Mirror:

- простота в освоении (по сравнению с чистой UNET);

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 19   |

- открытый исходный код и активное сообщество;
- подходит для инди-разработчиков и небольших студий;
- итоговый код не связан с серверами какой-либо компании, что даёт проекту независимости от сторонних компаний.

DoTween – это библиотека для анимации и твининга (плавного изменения значений) в Unity. Она позволяет легко создавать сложные анимации с минимальным кодом.

Ключевые особенности DoTween:

- простое управление анимациями через метод DoTween.To();
- поддержка анимации UI, 3D-объектов, материалов и даже самописных параметров;
- оптимизирована для производительности;
- цепочки анимаций (chaining) и кастомизация через параметры (ease, delay, loops).

Преимущества DoTween:

- интуитивно понятный API;
- высокая скорость работы (по сравнению с стандартным аниматором и LeanTween);
- широкие возможности для создания плавных и сложных анимаций.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 20   |

## 2 Проектирование и разработка

### 2.1 Описание архитектуры игры

Обычно для разработки небольших проектов используется архитектура с общим контроллером, который полностью контролирует игру. Однако, в TLS нужно реализовать симуляцию живого мира, а указанная выше архитектура для этого не подходит, вместо неё будет использоваться децентрализованная система контроля игры. Её суть в том, что в игре нет единого контроллера для всех процессов, вместо этого, каждый отдельный элемент игры имеет свой собственный контроллер, который не связан с остальными. С помощью такой архитектуры можно добиться того, что все элементы игры становятся независимыми друг от друга, но при этом могут друг на друга влиять. Для дальнейшего удобства, эта архитектура будет именоваться “Atmosphere”.

Atmosphere включает в себя несколько основных отдельных элементов:

- контроллер неба. Он следит за погодой и сменой дня и ночи;
- контроллер природы. Следит за сменой времени года и созданием природных объектов (деревья, камни и так далее);
- контроллер событий. Создаёт случайные события в мире.

Помимо основных контроллеров мира. Так же будут реализованы контроллеры отдельных объектов (например жителей города, генератора пещер и так далее).

Так же для корректной работы игры нужны контроллеры, не подвязанные к системе atmosphere. В основном это контроллеры, отвечающие за управление игроком и код, отвечающий за визуальную часть игры.

И ещё стоит обратить внимание на правильное разделение скриптов. Как правило, почти все объекты имеют внутреннюю часть и визуальную. Для упрощения работы с проектом, эти части должны быть разделены на разные скрипты, при этом внутренняя часть не должна зависеть от внешней.

Исходя из вышесказанного, архитектура игры должна придерживаться следующих принципов:

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 21   |

- все объекты на карте имеют собственный контроллер и набор интерфейсов для связи объектов друг с другом;
- объекты, не связанные с игровой картой, имеют свой контроллер, не связанный с Atmosphere;
- должно быть реализовано разделение скриптов на внутреннюю и визуальную часть.

Диаграмма основных классов представлена на рисунке 7.

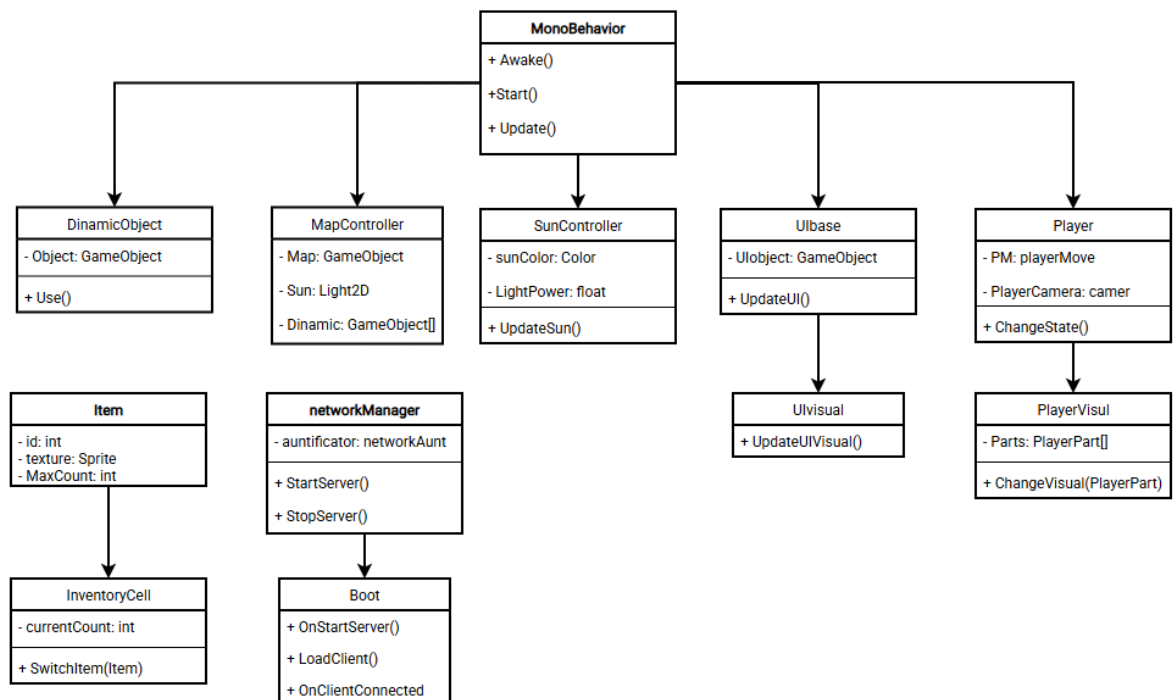


Рисунок 7 – UML диаграмма классов

При правильной настройке систем и архитектуры, игровой процесс должен иметь следующий вид (рисунок 8).

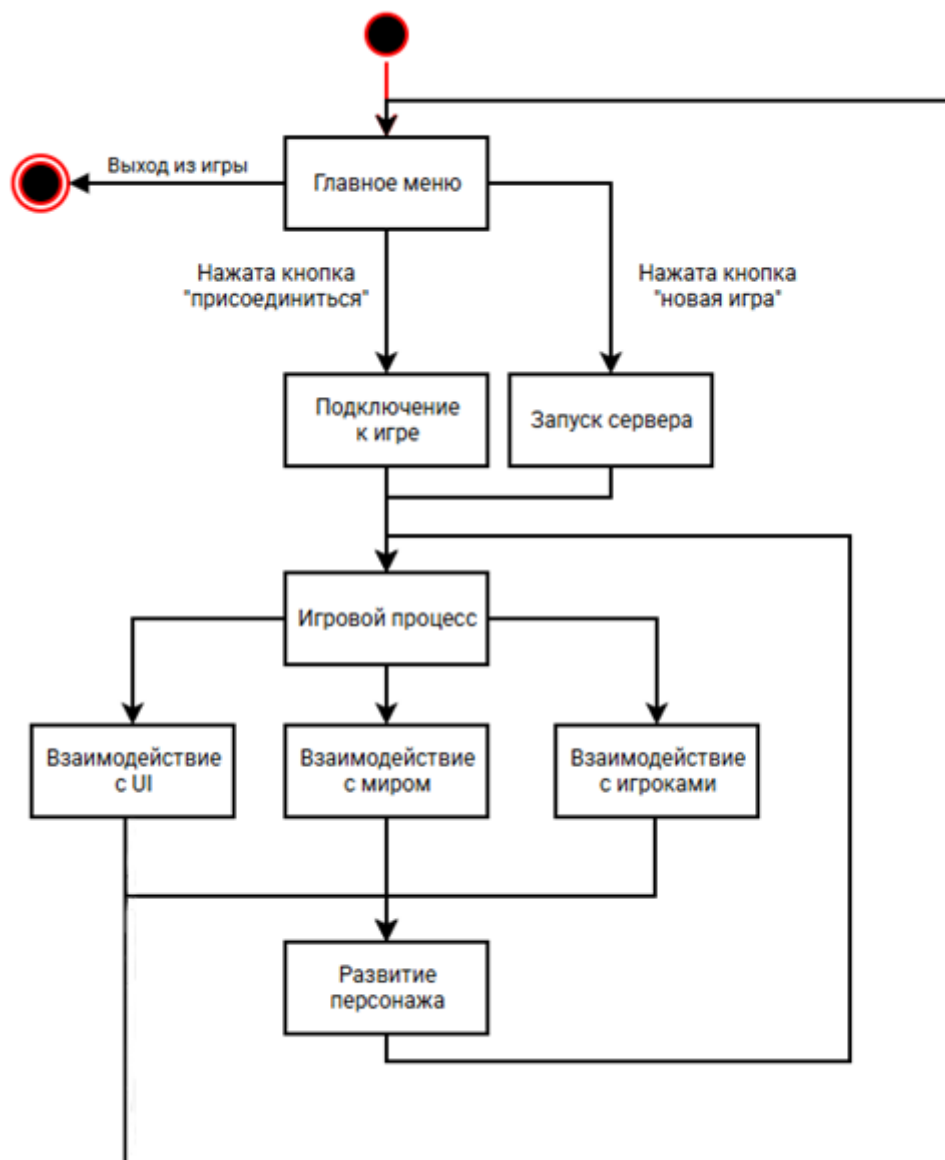


Рисунок 8 – UML диаграмма состояний

## 2.2 Подготовка рабочей среды

Перед началом разработки нужно создать проект в Unity и загрузить в него все нужные материалы и библиотеки.

В качестве шаблона был выбран 2D URP. Этот шаблон позволяет реализовывать 2D проекты с динамическим освещением. Но в этом шаблоне нужно правильно реализовать сортировку слоёв для TopDown проекта. Для этого открываем настройки проекта и во вкладке “Graphics” меняем в поле “Transparency Sort Axis” значение Y с 1 на 0, а Z с 0 на 1.

Следующий шаг это Mirror и DoTween. Для этого открываем “Package Manager” и находим в нём эти библиотеки, затем нажимаем кнопку “import”.

После выполнения вышеописанных действий, Unity настроен и готов к разработке TLS.

## 2.3 Подготовка сцен

Так как The little story это мультиплеерный проект, то нам понадобится две сцена (главное меню и основная игровая).

### 2.3.1 Главное меню

Для реализации главного, на сцене были размещены несколько кнопок, которые отвечают за создание игры, подключение к игре и выход из игры (рисунок 9). Так же был создан отдельный объект с компонентом “NetworkManager”, который будет обрабатывать все сетевые взаимодействия уже на главной сцене.



Рисунок 9 – Внешний вид главного меню

Также был написан скрипт для каждой из кнопок.

```
private NetworkManager NM;  
[SerializeField] private TMP_InputField ip;  
void Awake()  
{  
    NM = GetComponent<NetworkManager>();  
}
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 24   |

```

public void CreateOnline()
{
    GameSettings.gameType = GameType.NewGame;
    NM.maxConnections = 4;
    NM.StartHost();
    GameSettings.GameNetType = NetType.Host;
}

public void CreateSingle()
{
    GameSettings.gameType = GameType.NewGame;
    NM.maxConnections = 1;
    NM.StartHost();
    GameSettings.GameNetType = NetType.Singleplayer;
}

}

public void JoinServer()
{
    MainButtons.SetActive(false);
    JoinButtons.SetActive(true);
}

```

Все скрипты, связанные с обработкой UI будут написаны по похожему принципу, то есть пишется набор публичных методов, каждый из которых привязывается к своей кнопке.

### 2.3.2 Главная сцена

Проблема главной сцены заключается в порядке загрузки. Обычно для решения этой проблемы достаточно стандартных методов Unity (Start и Awake), но так как этот проект мультиплеерный, то нужно задавать чёткий порядок создания и инициализации объектов. Поэтому изначально главная сцена всегда пустая, а все игровые объекты создаются из префабов.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 25   |

Для реализации правильного порядка загрузки был создан класс Boot, которые наследуется от NetworkManager. Это позволяет переопределять стандартные методы NetworkManager, такие как OnStartServer и OnClientSceneChanged, и именно внутри этих методов мы реализуем порядок создания и инициализации объектов для клиента и сервера.

Для корректной загрузки TheLittleStory был написан следующий скрипт.

```
using Mirror;
using UnityEngine;
public class Boot : NetworkManager
{
    public static System.Action LoadMainUI;
    public static System.Action LoadWorldObjects;

    /// <summary>
    /// выполняется при подключении клиента
    /// </summary>
    public override void OnClientConnect()
    {
        base.OnClientConnect(); // Важно вызывать базовый
метод!

        // Выполняем нужный код (например, активация UI,
загрузка данных и т. д.)
    }
    /// <summary>
    /// выполняется на клиенте после смены сцены на игровую
    /// </summary>
    public override void OnClientSceneChanged()
    {
        base.OnClientSceneChanged();
        LoadClient();
    }
    public override void OnStartServer()
    {
        Debug.Log("Начало загрузки сервера");
    }
}
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 26   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        base.OnStartServer();
        Debug.Log("Загрузка карты");
        GameObject Map =
Instantiate(Resources.Load<GameObject>("Prefabs/map"));
        NetworkServer.Spawn(Map);
        Debug.Log("Загрузка динамических объектов");
        GameObject DynamicsObjects =
Resources.Load<GameObject>("Prefabs/DynamicsObjects");
        GameObject DynamicsObjectsContainer = new
GameObject("DynamicsObjectsContainer");
        for (int i = 0; i <
DynamicsObjects.transform.childCount; i++)
        {
            GameObject NewObject =
Instantiate(DynamicsObjects.transform.GetChild(i).gameObject.GetCo
mponent<TestDynamMap>().Prefab,
DynamicsObjectsContainer.transform);
            NewObject.transform.position =
DynamicsObjects.transform.GetChild(i).gameObject.transform.positio
n;
            NetworkServer.Spawn(NewObject);
        }
        Debug.Log("Загрузка геймконтроллера");
        GameObject GameController =
Instantiate(Resources.Load<GameObject>("Prefabs/GAME_CONTROLLER"))
;
        NetworkServer.Spawn(GameController);
        Debug.Log("Сервер загружен");

        LoadClient();

    }

    public override void
OnServerAddPlayer(NetworkConnectionToClient conn)
    {

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 27   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        base.OnServerAddPlayer(conn); // Создаёт игрока по
умолчанию
    }
    public void LoadClient()
    {
        Debug.Log("Загрузка игрока");
        base.OnClientSceneChanged();
        Debug.Log("Создание камеры");

        Instantiate(Resources.Load<GameObject>("Prefabs/MainCamera"));
        Debug.Log("Создание модели игрока");
        NetworkClient.connection.Send(new
AddPlayerMessage());
        //Загрузка главного UI
        Debug.Log("Создание интерфейса");
        GameObject MainUI =
Instantiate(Resources.Load<GameObject>("Prefabs/UI/PlayerInterface
")); //.GetComponent<MenuController>().Load();
        //DontDestroyOnLoad(MainUI);
        MainUI.GetComponent<MenuController>().Load();
        LoadMainUI?.Invoke();
        Debug.Log("Открывается редактор персонажа");

        Instantiate(Resources.Load<GameObject>("Prefabs/UI/CreateMenu"),
MenuController.singleton.mainUICanvas.transform);
        Debug.Log("Игрок загружен");
        LoadWorldObjects?.Invoke();
    }
}

```

Он переопределяет основные методы, которые вызываются при создании главной сцены и подключении игрока, при этом для инициализации сцены на клиенте и на сервере используются разные методы, что позволяет реализовать сетевой код загрузки сервера.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 28   |

## 2.4 Создание игрока

### 2.4.1 Внешний вид

Так как The little story имеет систему кастомизации персонажа, то модель персонажа разбита на несколько отдельных частей:

- тело;
- глаза;
- причёска верх;
- причёска низ;
- хвост.

При этом, некоторые отдельные части также разбиты ещё на две отдельные части для возможности более детальной кастомизации цвета. По этой же причине модель выполнена в белом цвете (рисунок 10)

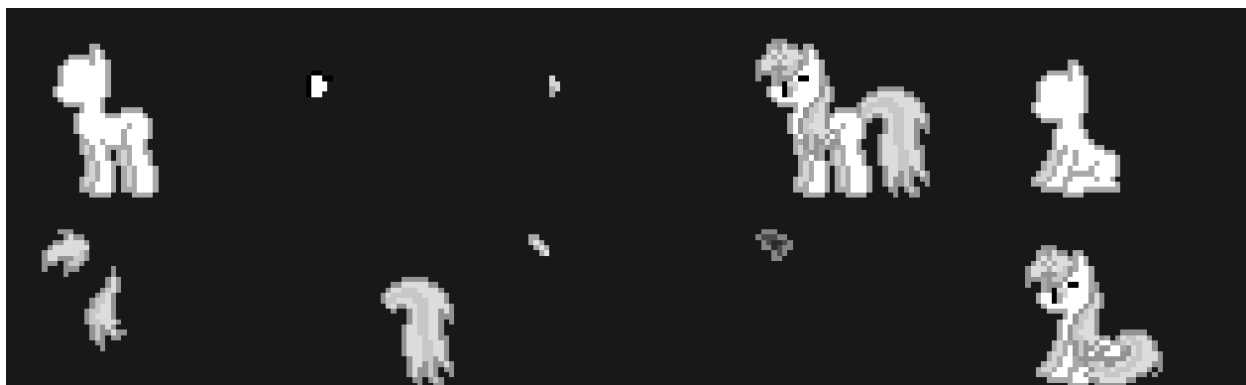


Рисунок 10 – Пример реализации модели персонажа

Для хранения каждой отдельной части персонажа был написан следующий класс.

```
[CreateAssetMenu(fileName = "PlayerSkin", menuName =  
"PlayerPart/Player Skin")]  
public class PlayerPartSkin : ScriptableObject  
{  
    [Tooltip("Название скина"),SerializeField] private string  
    _skinName;  
    public string SkinName { get { return _skinName; } }  
    [Tooltip("Основной спрайт скина"),SerializeField] private  
    Sprite _mainSprite;
```

```

        public Sprite MainSprite { get { return _mainSprite; } }
        [Tooltip("дополнительный спрайт скин"),SerializeField]
private Sprite _secondSprite;
        public Sprite SecondSprite { get { if (_secondSprite !=
null) return _secondSprite; else return null; } }
        [SerializeField] private AnimatorOverrideController
_skin;
        public AnimatorOverrideController Skin { get { return
_skin; } }
    }

```

Он наследуется от ScriptableObject, что позволяет хранить элементы персонажа в виде конфигов.

Так же был создан редактор персонажа, который открывается в самом начале игры (рисунок 11).



Рисунок 11 – Внешний вид редактора персонажа

#### 2.4.2 Взаимодействие с миром

Так как игрок должен взаимодействовать с миром, была разработана система взаимодействия игрока и мира. Состоит она из трёх частей:

- интерфейс для объектов в мире. Через него игрок может взаимодействовать с объектами в мире;
- интерфейс игрока. Через него объекты в мире могут взаимодействовать с игроком;
- скрипт взаимодействия. Это компонент, который задаётся объектам, которые могут взаимодействовать с игроком или другими объектами в мире.

Скрипт взаимодействия игрока имеет следующий код:

```
public class PlayerUse : NetworkBehaviour
{
    private Transform PlayerTransform;
    [SerializeField] private float MaxDistanse;
    void Awake()
    {
        PlayerTransform = gameObject.transform;
    }

    void Update()
    {
        if (!isLocalPlayer) return;
        if (Input.GetKeyDown(KeyCode.Mouse0) &&
            PlayerController.CanMove) Use();
    }

    public void Use()
    {
        Vector2 MousePosition =
            Camera.main.ScreenToWorldPoint(Input.mousePosition);
        if (MaxDistanse - Vector2.Distance(MousePosition,
            PlayerTransform.position) <= 0) return;
        Collider2D Object =
            Physics2D.OverlapPoint(MousePosition);
        UsableObject UO;
        if (Object != null &&
            Object.TryGetComponent<UsableObject>(out UO)) UO.Use();
    }
}
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 31   |

}

Его главным отличием от скриптов остальных объектов является привязка к NetworkManager для корректного сетевого взаимодействия.

### 2.4.3 Система магических навыков

Магические навыки являются одной из особенностей игры, так что их качественная реализация напрямую влияет на заинтересованность игрока. Для того чтобы создать хороший баланс этой системы были сделаны два основных ограничивающих фактора.

“Мана”. У игрока есть некоторый запас “маны” и использование магических способностей постепенно опустошает этот запас. Это позволит ограничить время использования магических способностей.

Максимальный заряд магии. Игрок может использовать несколько навыков одновременно и, чтобы игрок не мог использовать слишком много навыков за раз, у него есть некоторый заряд, который тратится при активации навыка и восстанавливается при деактивации.

Ниже представлена основная часть кода системы навыков.

```
[SerializeField] SpriteRenderer Magic;
[SerializeField] [SyncVar(hook =
nameof(SwtichEffect))]private int ActiveCount;

private void Start()
{
    if (!isLocalPlayer) return;
    gameObject.GetComponent<LightSkill>().SkillActive +=
AddMagic;
    gameObject.GetComponent<LightSkill>().SkillDisable +=
DeletMagic;
}

[Command]
public void AddMagic()
{
    ActiveCount++;
}
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 32   |

```

    }

    [Command]
    public void DeletMagic()
    {
        int NewValue = ActiveCount - 1;
        if (NewValue < 0) NewValue = 0;
        ActiveCount = NewValue;
    }

    public void SwtichEffect(int Old, int New)
    {
        if (ActiveCount > 0)
        {
            Magic.color = new Color(Magic.color.r,
Magic.color.g, Magic.color.b, 1);
        }
        else Magic.color = new Color(Magic.color.r,
Magic.color.g, Magic.color.b, 0);
    }

```

Она отвечает за активацию и деактивацию навыков, обработку сетевых взаимодействий и визуальные эффекты.

#### 2.4.4 Управление

На протяжении всей игры игрок будет переключаться между несколькими состояниями и для удобства реализации текущих состояний и добавления новых, система управления была реализована в виде машины состояний, в которой условием перехода из одного состояния в другое является нажатие кнопок на клавиатуре (рисунок 12).

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 33   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

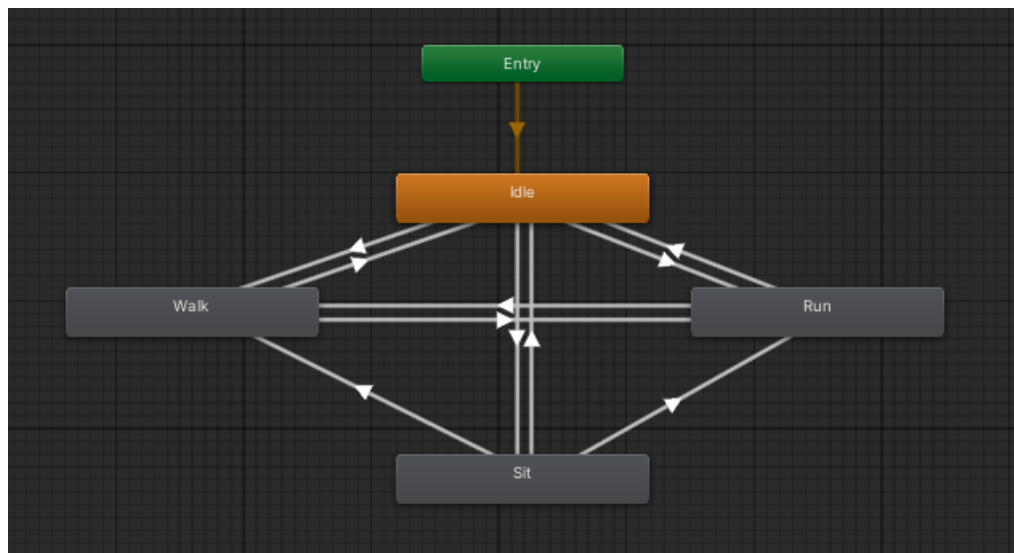


Рисунок 12 – Графическое представление состояний игрока

## 2.5 Графика

Unity по умолчанию имеет хороший URP для обработки пиксельной 2D графики, но для корректной работы света, все объектам, которые имеют графическое отображение в мире, нужно задать материал, в котором будут описаны свойства, необходимые для обработки света. Для этого создаём Material и задаём ему свойства, указанные на рисунке 13. После чего прикрепляем материал к спрайту объекта.

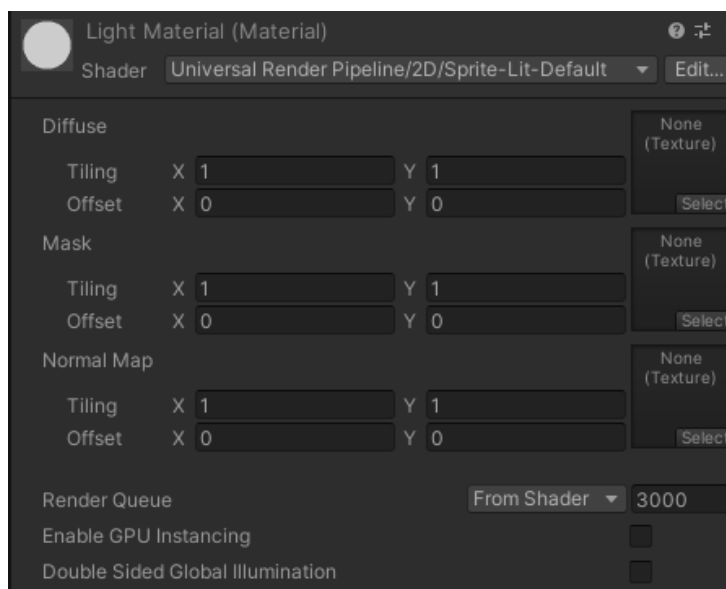


рисунок 13 – Настройки материала спрайтов

Этих настроек достаточно, чтобы сделать приятную графику (рисунок 14).



Рисунок 14 – Кадр из игры

## 2.6 Пользовательский интерфейс

Любой существующий объект на сцене, независим от других объектов и может иметь свой интерфейс и уникальный скрипт обработки этого интерфейса, но у многих из них есть похожие функции (например, открыть, закрыть и обновить), а также все их нужно контролировать, чтобы игрок не мог открыть слишком много интерфейсов.

Для решения этой проблемы был реализован контроллер интерфейсов, который состоит из самого контроллера и базового класса, которого наследуются все UI элементы на сцене.

Ниже представлен код базового класса элементов интерфейса.

```
public class StandartUI : UIBase
{
    [SerializeField] private GameObject UIobject;
    public GameObject pUIobject { get { return UIobject; } }

    private void Start()
```

```

        {
            if(UIObject == null) UIObject = gameObject;
        }

        public void OpenUI(bool EscListen = true, bool IsDestroy
= false)
        {
            MenuController.Regist(this, EscListen, IsDestroy);
        }
        public override bool Open()
        {
            UIObject.SetActive(true);
            return true;
        }

        public void CloseUI()
        {
            MenuController.Close(this);
        }

        public override void Close()
        {
            UIObject.SetActive(false);
        }
    }

```

Он реализует все необходимые методы для стабильно работы интерфейса.

## 2.7 Управление звуками

Unity позволяет создавать источники звука, но для грамотного саунд-дизайна игры нужно сделать нормальную систему воспроизведения звуков. Такая система должна решать две проблемы.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 36   |

Во-первых, возможность воспроизведения одиночных звуков, чтобы не приходилось крепить источник звука к каждому объекту (особенно это касается интерфейсов).

Во-вторых, должна быть возможность контролировать громкость разных типов источников звука (например, звуки интерфейса, погоды и так далее).

Для решения первой проблемы был написан класс GlobalAudio, который приведён ниже.

```
public class GlobalAudio : MonoBehaviour
{
    public static AudioSource Source;
    [SerializeField] private bool General;

    private void Start()
    {
        if (General) Source = GetComponent();
    }

    public static void Play(AudioClip Clip)
    {
        if (Source == null) return;
        Source.pitch = 1;
        Source.PlayOneShot(Clip);
    }

    public static void PlayRand(AudioClip Clip)
    {
        if (Source == null) return;
        Source.pitch = Random.Range(0.85f, 1.15f);
        Source.PlayOneShot(Clip);
    }
}
```

Его суть заключается в том, что можно выбрать один главный источник звука и воспроизводить аудиозаписи через него в любом другом скрипте и

объекте, даже если на самом объекте нет источника звука. Так же в нём есть функция PlayRand(), которая позволяет воспроизводить звуки со случайным питчем, что разнообразит звуковое сопровождение.

Для решения второй проблемы был создан скрипт LinkToVolume, который приведён ниже:

```
public class LinkToVolume : MonoBehaviour
{
    [SerializeField] private bool Global, Music, Radio,
Effects, Nature, Interface;
    [SerializeField] private float[] AddFloats;
    [SerializeField] private Slider[] AddSlider;
    [SerializeField] private AudioSource Source;

    void Start()
    {
        if(Source ==
null)gameObject.TryGetComponent<AudioSource>(out Source);
        if(Source != null)
        {
            if (Global) VolumeSettings.IsChangeGlobalVolume
+= ChangeVolume;
            if (Music) VolumeSettings.IsChangeMusicVolume +=
ChangeVolume;
            if (Radio) VolumeSettings.IsChangeRadioVolume +=
ChangeVolume;
            if (Effects) VolumeSettings.IsChangeEffectsVolume
+= ChangeVolume;
            if (Nature) VolumeSettings.IsChangeNatureVolume
+= ChangeVolume;
            if (Interface)
VolumeSettings.IsChangeInterfaceVolume += ChangeVolume;
            ChangeVolume();
        }
    }
}
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 38   |

```

private void OnDestroy()
{
    if (Source != null)
    {
        if (Global) VolumeSettings.IsChangeGlobalVolume -
= ChangeVolume;
        if (Music) VolumeSettings.IsChangeMusicVolume -=
ChangeVolume;
        if (Radio) VolumeSettings.IsChangeRadioVolume -=
ChangeVolume;
        if (Effects) VolumeSettings.IsChangeEffectsVolume
-= ChangeVolume;
        if (Nature) VolumeSettings.IsChangeNatureVolume -
= ChangeVolume;
        if (Interface)
VolumeSettings.IsChangeInterfaceVolume -= ChangeVolume;
    }
}

public void ChangeVolume()
{
    float NewVolume = 1;
    if (Global) NewVolume *=
VolumeSettings.GetGlobalVolume();
    if (Music) NewVolume *=
VolumeSettings.GetMusicVolume();
    if (Radio) NewVolume *=
VolumeSettings.GetRadioVolume();
    if (Effects) NewVolume *=
VolumeSettings.GetEffectsVolume();
    if (Nature) NewVolume *=
VolumeSettings.GetNatureVolume();
    if (Interface) NewVolume *=
VolumeSettings.GetInterfaceVolume();
    if(AddFloats.Length != 0) foreach (float New in
AddFloats) NewVolume *= New;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 39   |

```

        if (AddSlider.Length != 0) foreach (Slider New in
AddSlider) NewVolume *= New.value;
        Source.volume = NewVolume;
    }
}

```

По сути, этот компонент привязывает источник звука к переменным в классе VolumeSettings, которые можно менять в других скриптах, например в скрипте настроек звука.

## 2.8 Инвентарь

Есть два основных подхода к созданию инвентаря.

В первом подходе, все ячейки инвентаря хранятся в виде массива в рамках одного класса. Он подходит для игр, в которых инвентарей мало, и они статичны.

Второй подход состоит в привязки скрипта одной ячейки к объекту на сцене, который является этой самой ячейкой. Он сложнее в реализации, но при этом позволяет легко создавать динамические инвентари в любом количестве.

Так как в TLS планируется добавить большое количество объектов, многие из которых могут содержать свой инвентарь, то был выбран второй вариант. Для его реализации был написан базовый класс игрового предмета:

```

[System.Serializable]
public class Item
{
    [SerializeField] private int _id;
    public int id
    { get { return _id; } }
    [SerializeField] private string _itemName;
    public string itemName
    { get { return _itemName; } }

    [SerializeField] private string _description;
    public string description
    { get { return _description; } }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 40   |

```

[SerializeField] private Sprite _itemIcon;
public Sprite itemIcon
{ get { return _itemIcon; } }

[SerializeField] private ItemType _itemType;
public ItemType itemType
{ get { return _itemType; } }

[SerializeField] private int _coast;
public int coast
{ get { return _coast; } }

[SerializeField] private Quality _quality;
public Quality quality
{ get { return _quality; } set { _quality = value; } }

[SerializeField] private int _maxStackCount;
public int maxStackCount
{ get { return _maxStackCount; }}

//Конструктор \ / \ / \ /

public Item()
{
    _id = -1;
    _itemName = "ItemName";
    _description = "ItemDescription";
    _itemType = ItemType.None;
    _coast = 1;
    _quality = Quality.None;
    _maxStackCount = 1;
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 41   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

Он содержит все необходимые поля, которые могут быть использованы в ходе игрового процесса, а также несколько полей (такие как `_quality`), которые можно будет задействовать в следующих обновлениях.

Так же был написан скрип ячейки инвентаря.

```
public void SyncCellID(int Old, int New)
{
    _item.CopyItem(New);
    _itemID = New;
    CellUpdated?.Invoke();
}

public void SyncCellQuality(Quality Old, Quality New)
{
    _quality = New;
    CellUpdated?.Invoke();
}

[Command(requiresAuthority = false)]
public void addItemCMD(int ID, int Count, Quality
ItemQuality)
{
    addItem(ID, Count, ItemQuality);
}

public bool addItem(int ID, int Count, Quality ItemQuality)
{
    if ((_item.id != -1 && _quality != ItemQuality) ||
(_item.id != -1 && _item.id != ID)) return false;

    if (_itemCount == 0) _item.CopyItem(ID);

    if (_itemCount + 1 > _item.maxStackCount) return false;
    _itemCount += Count;
}
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 42   |

```

        _quality = ItemQuality;
        CellUpdated?.Invoke();
        _itemID = ID;
        return true;
    }

    [Command(requiresAuthority = true)]
    public void SwitchCellCMD(InventoryCell NewCell, int Count)
    {
        SwitchCellRPC(NewCell, Count);
        if (_itemType != ItemType.None && NewCell.item.itemType
        != _itemType) return;
        if (NewCell.item != _item)
        {
            Item itemBuf = _item;
            int itemCountBuff = _itemCount;
            _item = NewCell._item;
            _itemCount = NewCell._itemCount;
            NewCell._item = itemBuf;
            NewCell._itemCount = itemCountBuff;
        }
    }

    CellUpdated?.Invoke();
}

```

В нём реализованы все базовые методы для взаимодействия с ними.

## 2.9 Игровая локация

Для ускорения процесса сборки мира, был выбран плиточный метод сборки. Его суть заключается в том, что весь мир собирается из определённого набора плиток.

Для реализации данного метода в unity есть встроенный инструментарий. Чтобы его использовать, достаточно создать 2D объект с компонентом Grid и собрать палитру плиток из заранее загруженных спрайтов.

Для текущей версии игры была собрана небольшая карта, включающая в себя две локации, а именно основной мир (рисунок 15) и пещера (рисунок 16)

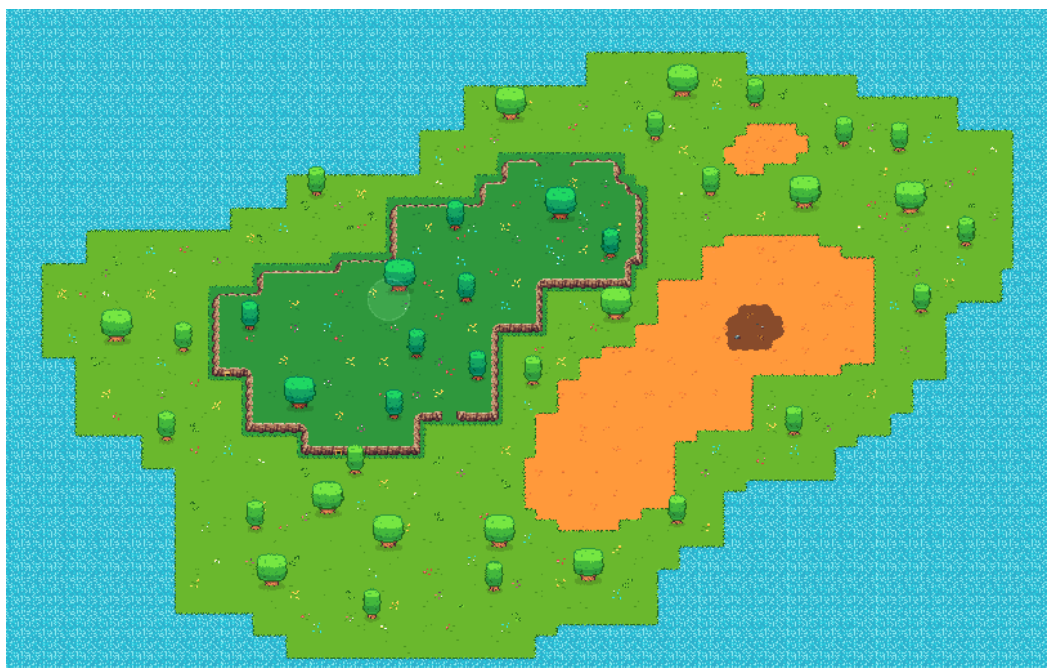


Рисунок 15 – Первая игровая локация



Рисунок 16 – Вторая игровая локация

## 2.10 Мультиплеер

Для реализации клиент-серверных взаимодействий, в проект была импортирована библиотека Mirror. Она добавляет набор атрибутов для методов. В основном на понадобятся 3 атрибута:

- syncvar. Этот атрибут применяется к переменной. Если она изменяется на сервере, то автоматически изменяется и на клиентах;
- command. Применяется к методам. Если метод с этим атрибутом вызвать на клиенте, то выполнится он на сервере;
- clientRPC. Применяется к методам. При вызове на сервера, произойдёт вызов этого метода на всех клиента.

Все эти методы могут передавать данные (например параметры функция у command и ClientRPC). Но не все типы данных можно передавать. В перечень запретов входят самописные классы, что усложняет передачу данных между сервером и клиентом. Частично это проблему можно решить с помощью структур. Все эти атрибуты спокойно могут передавать самописные структуры, но внутри них не должно быть типов, которые не поддерживают передачу.

Command и ClientRPC позволяют синхронизировать данные по ходу игры, при подключении клиента, ему нужно синхронизироваться с сервером. Частично в этом помогает Syncvar, так как все поля с этим атрибутом автоматически синхронизируются при подключении, но одного Syncvar недостаточно. Для правильной синхронизации нужно правильно задать последовательность загрузки игровых объектов на клиенте и их последующую синхронизацию на сервере. Для этого используется скрипт Boot, описанный в пункте 2.3.2

Общий порядок взаимодействия клиента и сервера указан на рисунке 17.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 45   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

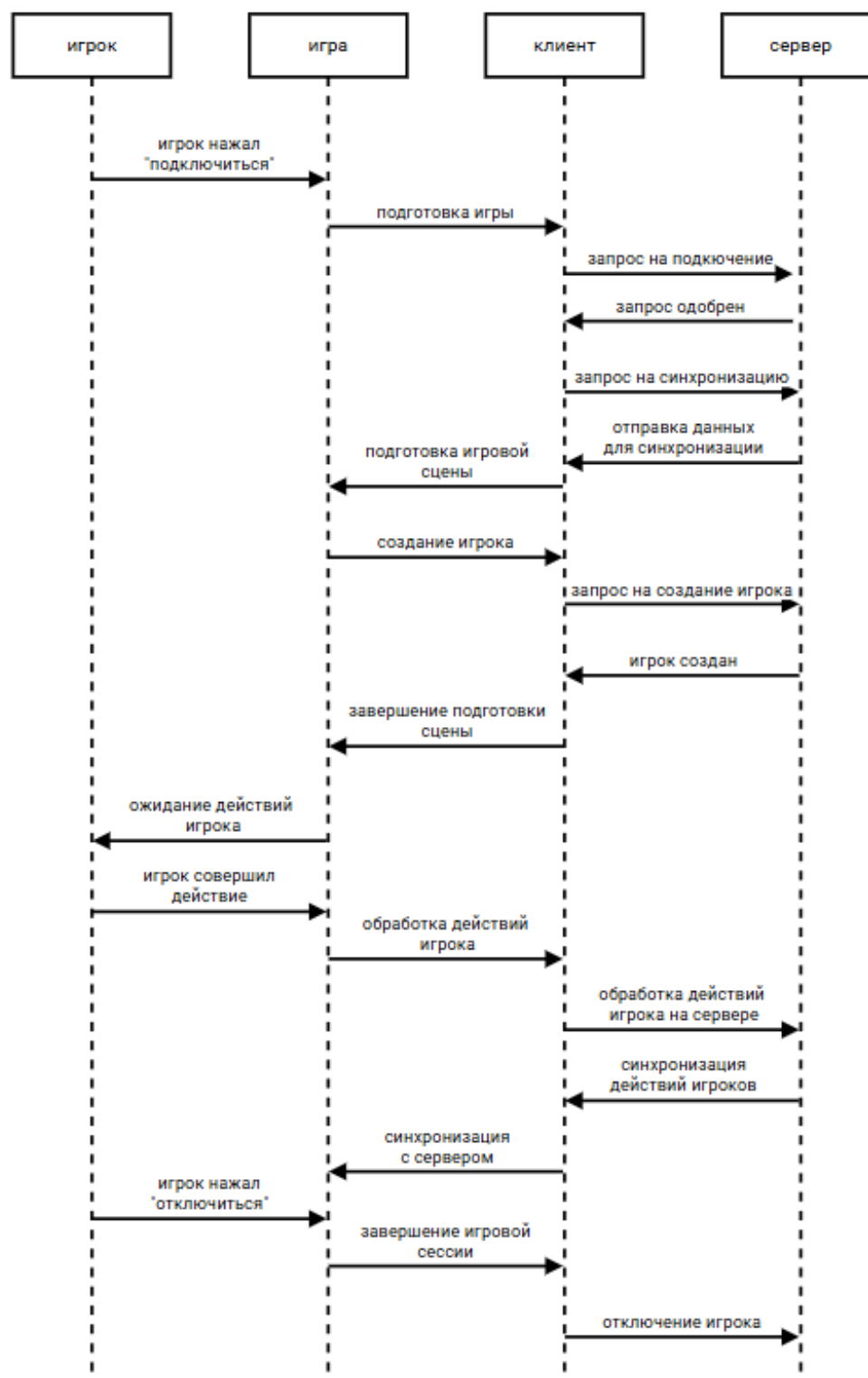


Рисунок 17 – UML диаграмма взаимодействия

### 3 Тестирование и отладка

Для тестирования игры я запустил сборку игры на двух компьютеров, соединив их с помощью эмулятора локальной сети RadminVPN. После запустил на первом компьютере сервер и подключился к нему со второго компьютера.

Сборка была проверена на наличие следующих ошибок:

- ошибки при синхронизации объектов;
- визуальные баги;
- ошибки работы отдельных систем связанных с Atmosphere;
- ошибки в работе UI.

В результате были найдены небольшие ошибки синхронизации и визуальные баги.

Визуальный баг заключался в появлении белых, мерцающих полос между тайлами. Эта ошибка была исправлена с помощью объединения спрайтов тайлов в один общий текстурный атлас.

Ошибка синхронизации заключалась в блокировке вызова функций с атрибутом [command] на объектах, которые не являлись игроками. Для решения этой проблемы, данным объектам был выдан соответствующий приоритет, который позволил проводить вызовы таких функций без проблем.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 47   |

## 4 Безопасность жизнедеятельности

При разработке игрового приложения на Unity чрезвычайно важно учитывать правила безопасности в работе с персональным компьютером. Ниже привожу основные риски и рекомендации по их устранению, которые основаны на требованиях ГОСТ 12.0.003-2015.

Риск ухудшения зрения и утомления глаз:

- длительная работа перед монитором без перерыва очень быстро может привести к усталости глаз, а со временем к ухудшению зрения;
- неправильное освещение рабочего места и различные блики на экране, также увеличивают напряжение на глаза.

Рекомендации для избежания ухудшения зрения:

- нужно делать перерывы каждые 60 минут и использовать правило 20-20-20 (каждые 20 минут смотреть на другой объект на расстоянии 6 метров на протяжении 20 секунд);
- требуется настроить яркость и контраст монитора, чтобы при данном освещении в помещении на экране не было бликов;
- монитор должен находиться на расстоянии 50-70 см от глаз, а верхний край монитора необходимо расположить на уровне и немного ниже уровня глаз.

Риски травм и заболеваний опорно-двигательной системы:

- при работе за компьютером в неправильной позе могут развиваться боли в спине, шее и запястьях;
- если ваше рабочее кресло не эргономичное, это создает дополнительную нагрузку на позвоночник и шею.

Рекомендации для избежания травм и заболеваний опорно-двигательной системы:

- необходимо использовать кресло, у которого есть функция поддержки поясницы и регулирования высоты. Ваши стопы должны полностью стоять на полу, а колени согнуты под углом 90 градусов;

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 48   |

– при работе с клавиатурой и мышкой рекомендуется использовать подставки для запястий.

Риск пожара и электрического удара. Чрезмерная нагрузка на розетки, использование кабелей, у которых нарушена изоляционная оплетка или любого другого неисправного оборудования может привести к возгоранию рабочего помещения или удару электрическим током.

Чтобы снизить риск данной проблемы, перед подключением оборудования к электрической сети, необходимо убедиться в исправности розетки, проверить кабель на повреждение оплетки и только после этого приступать к подключению.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 49   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## 5 Стартap проект

### 5.1 Целевая аудитория

Поскольку данный проект не имеет ограничений по возрасту, сложного или специфического игрового процесса, то и охватить можно не малую часть рынка видео игр. Ближайший аналог, а именно Stardew valley, согласно официально статистике магазина Steam, смог привлечь внимание игроков в возрасте от 8 до 57 лет, а средний возраст пользователя составил 23 года. Если смотреть в сторону интересов игроков, то большую популярность Stardew valley обрела среди казуальных игроков (более 70% пользователей). Игроки, предпочитающие более требовательные к навыкам проекты реже обращали внимание на этот проект, но, согласно опросам того же Steam, лишь около 15% всех игроков являются такими.

Основываясь на приведённой статистике, для TLS был выбран следующий курс. Изначально игра ориентирована на казуальных игроков в возрасте от 14 до 30 лет, при этом, будет добавляться контент, который сможет заинтересовать любителей сложных игр. Уже через некоторое время, основываясь на статистике уже данного проекта, фокус будет смещаться на ту аудиторию, которую больше всего привлёк проект.

### 5.2 Источники дохода и аудитории

В связи с проблемами выхода на зарубежные площадки, не получится использовать стандартную схему монетизация с единоразовой покупкой продукта, поэтому была выбрана следующая схема. Старые версии проекта будут распространяться бесплатно, но самую новую версию можно будет получить, купив подписку. Такая схема даст игрокам возможность ознакомиться с игрой без лишних проблем, а в связке с частыми обновлениями, такая схема позволит не только продать проект, но и создать ажиотаж вокруг проекта. Со временем размер обновлений и промежутки между ними будут увеличиваться,

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 50   |

что придаст людям мотивации покупать подписку, чтобы получить последнюю версию раньше.

Подобная схема использовалась в игре SCP 3008 и VOTV. Оба проекта не выкладывались на популярные торговые площадки и не закупали много рекламы, но благодаря описанной схеме смогли получить массовую известность и преданную аудиторию.

### 5.3 Риски

Для проекта такого типа есть два основных риска.

Во-первых, из-за отсутствия нормальной рекламы, есть вероятность не собрать стартовую аудиторию, которая будет основным источником продвижения игры. Это может произойти из-за неверного выбора площадок для тизера и анонсов проекта. Чтобы снизить риск такого исхода, нужно внимательно изучить аудиторию площадки прежде, чем начинать работу с ней. К тому же, плохо подобранная площадка, с незаинтересованной аудиторией, может не только не дать старта распространению проекта, но и вызвать негативную реакцию на него.

Во-вторых, выбор формата подписки. Будет ли это ежемесячная подписка или ежегодная, сколько будет стоить и будет ли давать ещё какие-либо преимущества помимо новейшей версии игры. Универсального ответа на эти вопросы нет, так как всё также зависит от стартовой аудитории, которая зависит от выбранной площадки. Как показала практика подобного проекта с подобной схемой Kenshi, слишком высокая цена отпугнула большую часть аудитории из-за чего, получить распространение проект смог только после выхода на более крупные площадки. К тому же более молодую аудиторию может отпугнуть отсутствие дополнительного контента.

Помимо всего этого есть проблема пиратства, которая может сильно снизить продажи. Поскольку создание нормальной антипиратской защиты стоит огромных денег, то для борьбы с пиратством будет использоваться всё та же

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 51   |

схема бесплатного распространения старых версий, но полностью исключить пиратство задача невыполнимая.

В общем, для снижения основных рисков делаем следующие шаги:

- правильно подбираем площадку для распространения с заинтересованной аудиторией;
- на основе выбранной площадки и её аудитории составляем систему подписки, чтобы получить максимальную прибыль и не отпугнуть аудиторию;
- для снижения вероятности пиратства распространяем старые версии игры бесплатно.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 52   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## Заключение

В рамках выпускной квалификационной работы было разработан симулятор сельского хозяйства с элементами ролевой игры при использовании игрового движка Unity и языка программирования C#.

Разработанная игра «The little story» реализована в соответствии с поставленными в начале техническими и функциональными требованиями. Получилось достигнуть высокий уровень производительности, разнообразие геймплея и возможность кооперативного прохождения.

Использование URP позволило реализовать красивую и не требовательную графику.

Использование Mirror позволило реализовать возможности прохождения игры в кооперативе до 4 человек, не исключая возможности одиночной игры.

В дальнейшем можно выложить игру на платформы и продолжить ее поддержку.

Постоянно добавлять новый контент:

- расширение игровой карты;
- добавление новых персонажей с уникальными сюжетными линиями;
- добавить большего разнообразия для развития персонажа;
- добавление новых механик и улучшение имеющихся.

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 53   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## Список использованных источников

1. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер. – М.: Вильямс, 2020. – 544с.
2. Unity Learn. [электронный ресурс]: <https://learn.unity.com> (дата обращения 1.04.2025).
3. Mirror GitBook. [электронный ресурс]: <https://mirror-networking.gitbook.io/docs> (дата обращения 7.04.2025)
4. Unity Technologies. Unity Official Documentation [электронный ресурс]: <https://docs.unity3d.com/Manual/URP.html> (дата обращения 12.03.2025).
5. Официальная документация Unity. [электронный ресурс]: <https://docs.unity.com/en-us> (дата обращения 17.04.2025).
6. Документация по C# от Microsoft. [электронный ресурс]: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения 15.03.2025).

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 54   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## Приложение А

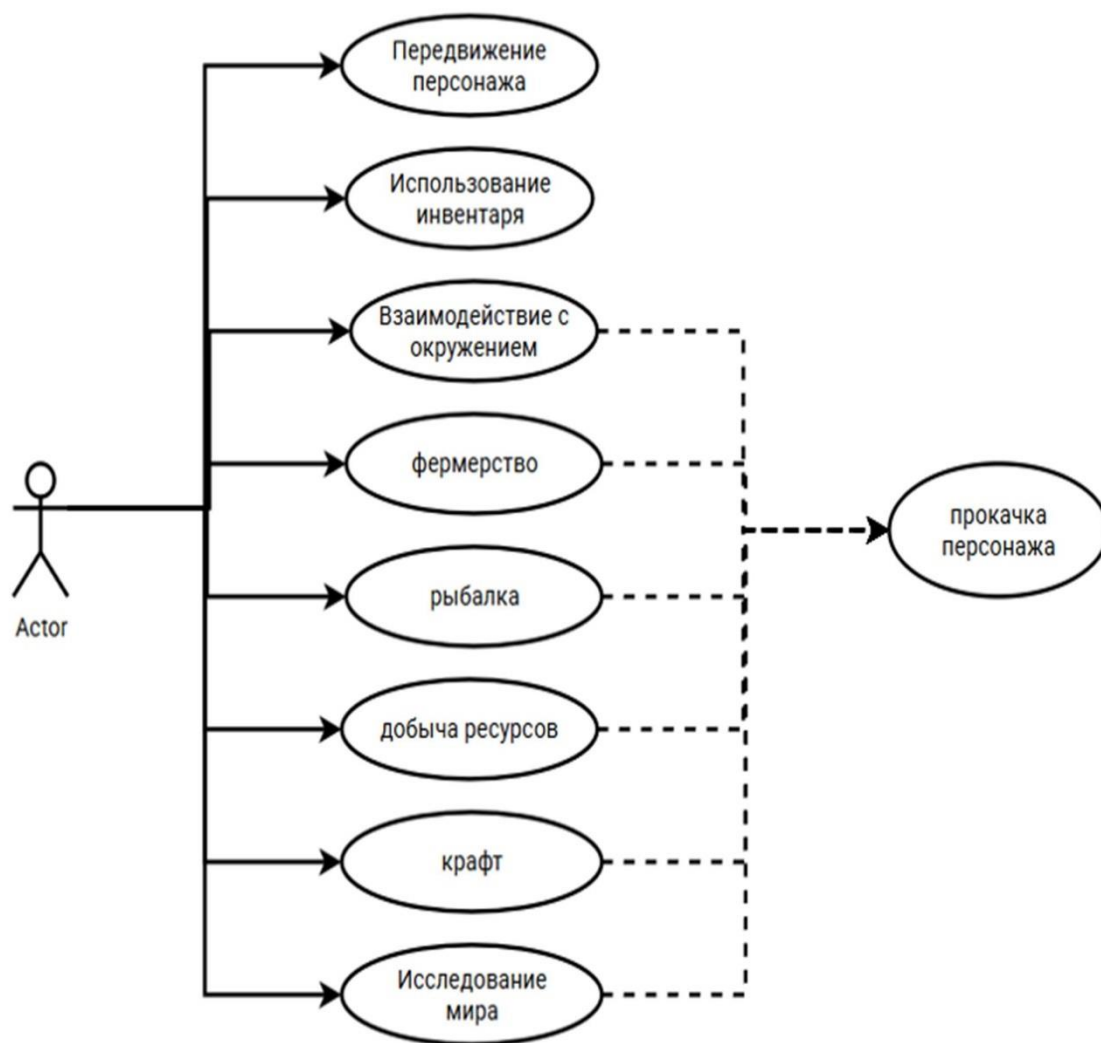
(Обязательное)

Бизнес-проект “Симулятор сельского хозяйства с элементами ролевой игры”

UML диаграммы

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 55   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## Приложение А.1 – UML – диаграмма вариантов использования



|         |                |          |         |      |                         |         |         |
|---------|----------------|----------|---------|------|-------------------------|---------|---------|
|         |                |          |         |      | ПГУ 1.090301.05 002 D13 |         |         |
| Изм.    | Лист           | № докум. | Подпись | Дата | Продолжение таблицы     |         |         |
| Разраб. | Григорьев С.В. |          |         |      | Лист                    | Масштаб | Масштаб |
| Провер. | Андреев И.В.   |          |         |      |                         |         |         |
| Утверд. |                |          |         |      | Лист                    | Листов  |         |
| Исполн. | Богачев А.С.   |          |         |      | ФОРМ. зр. 2.10.001      |         |         |
| Исполн. | Михайлов М.А.  |          |         |      |                         |         |         |

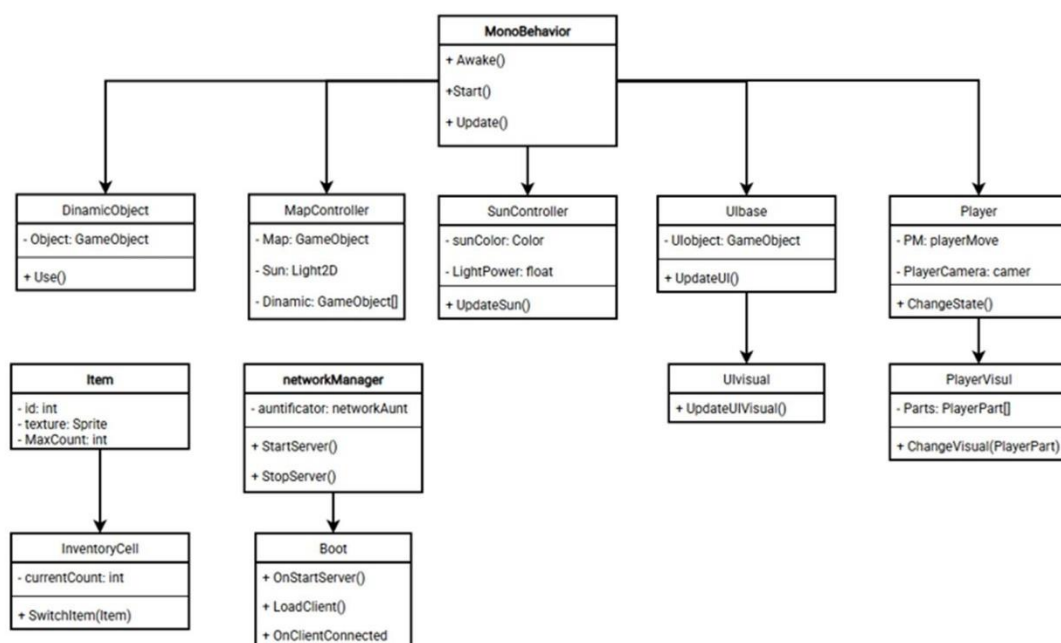
|     |      |          |         |      |
|-----|------|----------|---------|------|
|     |      |          |         |      |
| Изм | Лист | № докум. | Подпись | Дата |

ПГУ 1.090301.05 001 ПЗ

лист

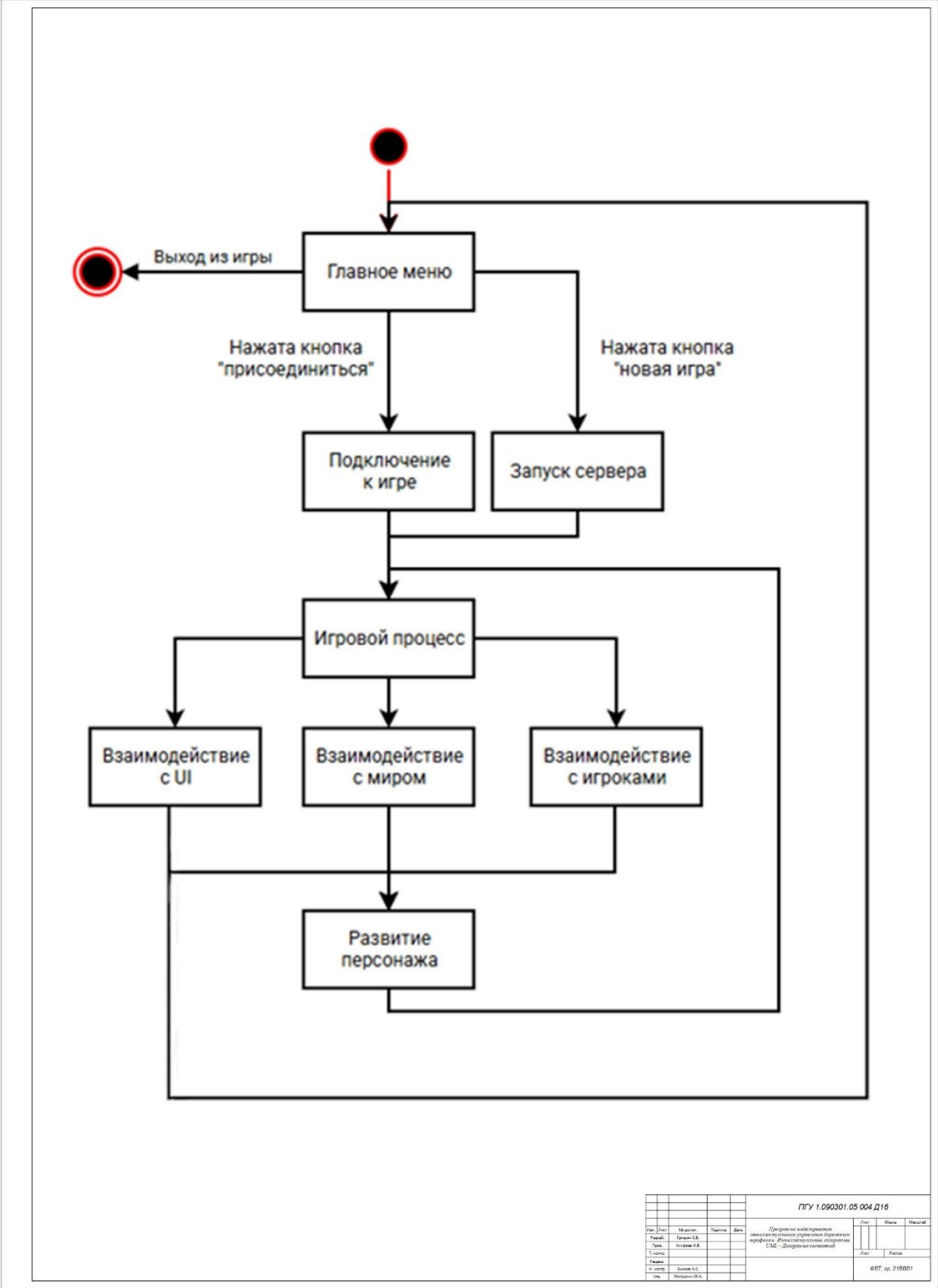
56

## Приложение А.2 – UML – диаграмма классов



|                         |                |     |     |      |      |         |         |
|-------------------------|----------------|-----|-----|------|------|---------|---------|
| ПГУ 1.090301.05 002 Д11 |                |     |     |      | Лист |         | Масштаб |
| Имя                     | Пол            | Имя | Пол | Дата | Лист | Масштаб |         |
| Фамилия                 | Григорьев С.В. | Пол | Муж | Дата | Лист | Масштаб |         |
| Телефон                 | Александр В.В. | Пол | Муж | Дата | Лист | Масштаб |         |
| Почта                   |                | Пол | Муж | Дата | Лист | Масштаб |         |
| Имя                     | Иванов А.С.    | Пол | Муж | Дата | Лист | Масштаб |         |
| Фамилия                 | Михайлов М.А.  | Пол | Муж | Дата | Лист | Масштаб |         |

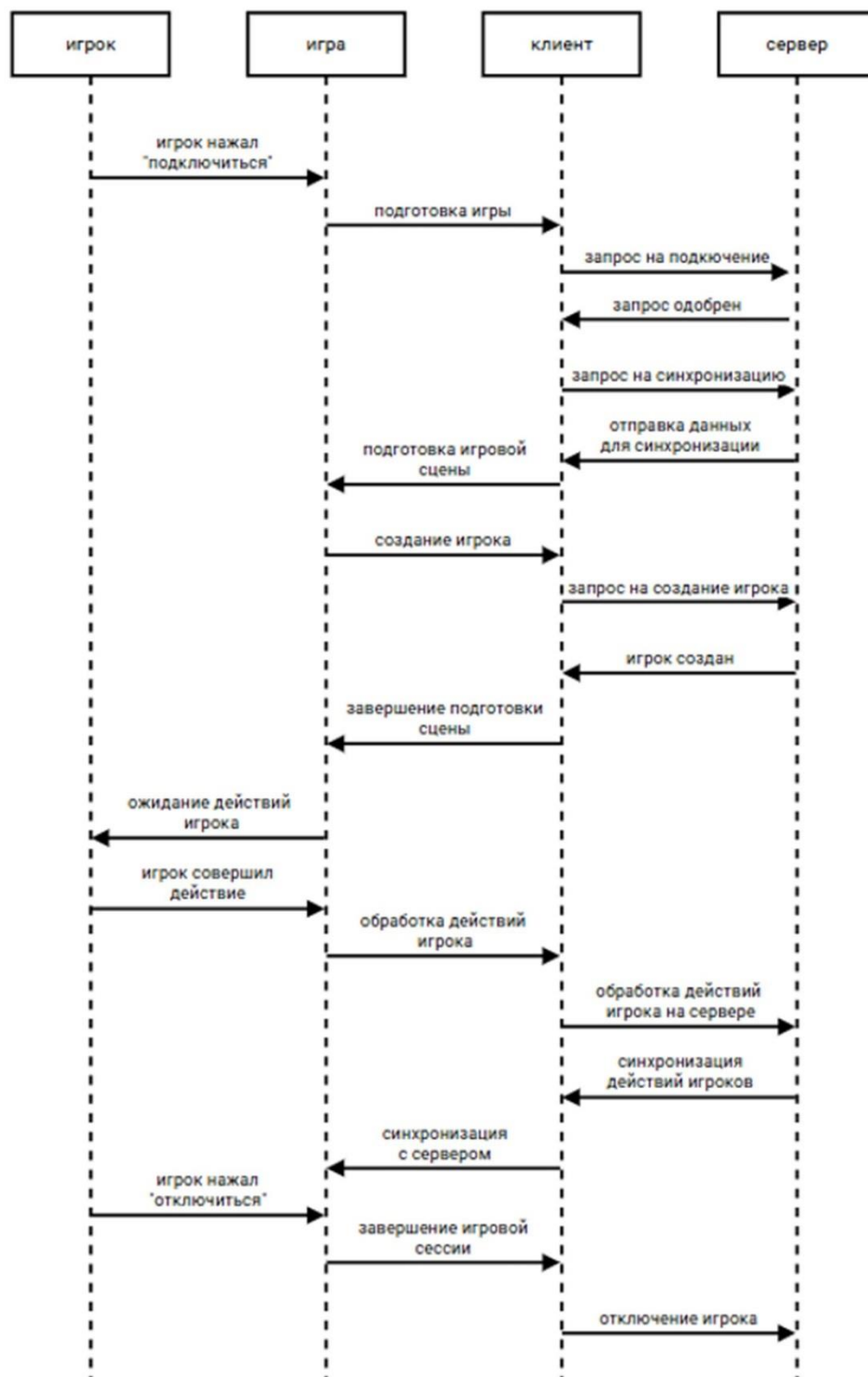
Приложение А.3 – UML – диаграмма состояний



|      |      |      |      |      |                                                                                                                  |  |  |  |  |
|------|------|------|------|------|------------------------------------------------------------------------------------------------------------------|--|--|--|--|
|      |      |      |      |      | ПГУ 1.090301.05 004 Д16                                                                                          |  |  |  |  |
| Имя  | Лист | Имя  | Имя  | Имя  | Проектное подразделение<br>информационных технологий<br>профилю «Информационные технологии<br>СД» – «Дисциплины» |  |  |  |  |
| Роль | Роль | Роль | Роль | Роль |                                                                                                                  |  |  |  |  |
| Дата | Дата | Дата | Дата | Дата |                                                                                                                  |  |  |  |  |
| Имя  | Имя  | Имя  | Имя  | Имя  |                                                                                                                  |  |  |  |  |
| Имя  | Имя  | Имя  | Имя  | Имя  | ФБТ, стр. 216001                                                                                                 |  |  |  |  |
| Имя  | Имя  | Имя  | Имя  | Имя  |                                                                                                                  |  |  |  |  |

|     |      |          |         |      |
|-----|------|----------|---------|------|
|     |      |          |         |      |
| Изм | Лист | № докум. | Подпись | Дата |

## Приложение А.4 – UML – диаграмма взаимодействия



|     |         |     |         |           |                                                                                                                  |  |  |
|-----|---------|-----|---------|-----------|------------------------------------------------------------------------------------------------------------------|--|--|
|     |         |     |         |           | ПУ 1.090301.05 005 Д17                                                                                           |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |
| Имя | Фамилия | Имя | Фамилия | Должность | Программа подготовки специалистов среднего звена по специальности 19.03.01 – Информационные системы и технологии |  |  |

007, стр. 210001

|     |      |          |         |      |
|-----|------|----------|---------|------|
| Изм | Лист | № докум. | Подпись | Дата |
|     |      |          |         |      |

ПУ 1.090301.05 001 ПЗ

лист

59

## Приложение Б

(Обязательное)

Бизнес-проект “Симулятор сельского хозяйства с элементами ролевой игры”

Листинг программы

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 60   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## Файл "Boot.cs"

```
using System.Collections;
using Mirror;
using UnityEngine;

public class Boot : NetworkManager
{
    public static System.Action LoadMainUI;
    public static System.Action LoadWorldObjects;

    /// <summary>
    /// выполняется при подключении клиента
    /// </summary>
    public override void OnClientConnect()
    {
        base.OnClientConnect(); // Важно вызывать базовый
метод!

        // Выполняем нужный код (например, активация UI,
загрузка данных и т. д.)
    }

    /// <summary>
    /// выполняется на клиенте после смены сцены на игровую
    /// </summary>
    public override void OnClientSceneChanged()
    {
        base.OnClientSceneChanged();
        LoadClient();
    }

    /// <summary>
    /// Выполняется при запуске сервера
    /// </summary>
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 61   |

```

public override void OnStartServer()
{
    Debug.Log("Начало загрузки сервера");
    base.OnStartServer();
    Debug.Log("Загрузка карты");
    GameObject Map =
Instantiate(Resources.Load<GameObject>("Prefabs/map"));
    NetworkServer.Spawn(Map);
    Debug.Log("Загрузка динамических объектов");
    GameObject DynamicsObjects =
Resources.Load<GameObject>("Prefabs/DynamicsObjects");
    GameObject DynamicsObjectsContainer = new
GameObject("DynamicsObjectsContainer");
    for (int i = 0; i <
DynamicsObjects.transform.childCount; i++)
    {
        GameObject NewObject =
Instantiate(DynamicsObjects.transform.GetChild(i).gameObject.GetCo
mponent<TestDynamMap>().Prefab,
DynamicsObjectsContainer.transform);
        NewObject.transform.position =
DynamicsObjects.transform.GetChild(i).gameObject.transform.positio
n;

        //if (!spawnPrefabs.Contains(NewObject))
        //{
        //    spawnPrefabs.Add(NewObject);
        //}
        NetworkServer.Spawn(NewObject);
    }
    Debug.Log("Загрузка геймконтроллера");
    GameObject GameController =
Instantiate(Resources.Load<GameObject>("Prefabs/GAME_CONTROLLER"))
;

    NetworkServer.Spawn(GameController);
    Debug.Log("Сервер загружен");
}

```

```

        LoadClient();

    }

    /// <summary>
    /// Выполняется на сервере после спауна игрока
    /// </summary>
    public override void
OnServerAddPlayer(NetworkConnectionToClient conn)
    {
        base.OnServerAddPlayer(conn); // Создаёт игрока по
умолчанию
    }

    /// <summary>
    /// Метод для загрузки игрока
    /// </summary>
    public void LoadClient()
    {
        Debug.Log("Загрузка игрока");
        base.OnClientSceneChanged();
        Debug.Log("Создание камеры");

Instantiate(Resources.Load<GameObject>("Prefabs/MainCamera"));
        Debug.Log("Создание модели игрока");
        NetworkClient.connection.Send(new
AddPlayerMessage());
        //Загрузка главного UI
        Debug.Log("Создание интерфейса");
        GameObject MainUI =
Instantiate(Resources.Load<GameObject>("Prefabs/UI/PlayerInterface
")); //.GetComponent<MenuController>().Load();
        //DontDestroyOnLoad(MainUI);
        MainUI.GetComponent<MenuController>().Load();
        LoadMainUI?.Invoke();
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 63   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        //GameObject Map =
Instantiate(Resources.Load<GameObject>("Prefabs/map"));

//Instantiate(Resources.Load<GameObject>("Prefabs/GAME_CONTROLLER"
)).GetComponent<DayController>().SetSun(Map.gameObject.GetComponent
t<MapController>().sun.gameObject.GetComponent<Light2D>());

        //LoadWorldObjects?.Invoke();
        Debug.Log("Открывается редактор персонажа");

Instantiate(Resources.Load<GameObject>("Prefabs/UI/CreateMenu"),
MenuController.singleton.mainUICanvas.transform);

        Debug.Log("Игрок загружен");
        LoadWorldObjects?.Invoke();

    }

}

```

#### Файл "StandartUI.cs"

```

using UnityEngine;

/// <summary>
/// стандартный класс для управления элементов UI
/// </summary>
public class StandartUI : UIBase
{
    [SerializeField] private GameObject UIobject;
    public GameObject pUIobject { get { return UIobject; } }

    private void Start()
    {
        if(UIobject == null) UIobject = gameObject;
    }
}

```

```

        public void OpenUI(bool EscListen = true, bool IsDestroy
= false)
        {
            MenuController.Regist(this, EscListen, IsDestroy);
        }
        public override bool Open()
        {
            UIObject.SetActive(true);
            return true;
        }

        public void CloseUI()
        {
            MenuController.Close(this);
        }

        public override void Close()
        {
            UIObject.SetActive(false);
        }
    }

```

#### Файл "EscMenuScript.cs"

```

using System.Collections;
using UnityEngine;
using DG.Tweening;

/// <summary>
/// класс для управления Esc меню. Крепится к главному
канвасу так как долже всегда выполняться
/// </summary>
public class EscMenuScript : UIbase
{
    /// <summary> ссылка на объект с Esc меню </summary>
    [SerializeField] private GameObject UI;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 65   |

```

        /// <summary> ссылка на объект с главными кнопками для
реализации анимации </summary>
        [SerializeField] private GameObject MainButtons;
        /// <summary> ссылка на объект с кнопками настроек для
реализации анимации </summary>
        [SerializeField] private GameObject SettingsButtons;
        /// <summary> ссылка на объект с настройками звука для
реализации анимации </summary>
        [SerializeField] private GameObject VolumeSettingPanel;

        /// <summary> разрешение меню в пикселях для реализации
анимации </summary>
        [SerializeField] private float Xsize, Ysize;
        /// <summary> главный трансформ Esc меню </summary>
        private RectTransform UITransform;
        /// <summary> трансформ панели с главными кнопками
</summary>
        private RectTransform MainButtonsTransform;
        /// <summary> трансформ панели с кнопками
настроек</summary>
        private RectTransform SettingsButtonsTransform;
        /// <summary> трансформ панели настроек звука </summary>
        private RectTransform VolumePanelTransform;
        /// <summary> флаг активной анимации </summary>
        private bool IsChanging;
        void Start()
        {
            UITransform = UI.GetComponent<Transform>() as
RectTransform;
            MainButtonsTransform =
MainButtons.GetComponent<Transform>() as RectTransform;
            SettingsButtonsTransform =
SettingsButtons.GetComponent<Transform>() as RectTransform;
            VolumePanelTransform =
VolumeSettingPanel.GetComponent<Transform>() as RectTransform;
            IsOpen = false;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 66   |

```

        IsChanging = false;
    }

    void LateUpdate()
    {

        if (Input.GetKeyDown(KeyCode.Escape) && !IsChanging
&& MenuController.singleton.OpenMenuCount == 0 &&
!MenuController.EscMenuFlag && !IsOpen)
        {
            MenuController.Regist(this, true, false);
        }
        if (MenuController.EscMenuFlag)
        {
            MenuController.EscMenuFlag = false;
        }
    }

    public override bool Open()
    {
        StartCoroutine(OpenMainMenu());
        return true;
    }

    public IEnumerator OpenMainMenu()
    {
        EscListen = false;
        IsChanging = true;
        MainButtonsTransform.anchoredPosition = new
Vector2(0, 0);
        VolumeSettingPanel.SetActive(false);
        SettingsButtons.SetActive(false);
        MainButtons.SetActive(true);
        UI.SetActive(true);
        UITransform.anchoredPosition = new Vector2(-Xsize,
0);

        UITransform.DOAnchorPosX(0, 0.25f);
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 67   |

```

        yield return new WaitForSeconds(0.5f);
        IsOpen = true;
        IsChanging=false;
        EscListen = true;
    }

    public void CloseEscMenuButton()
    {
        MenuController.Close(this);
    }
    public override void Close()
    {
        StartCoroutine(CloseMainMenu());
    }
    public IEnumerator CloseMainMenu()
    {
        IsChanging = true;
        UITransform.DOAnchorPosY(-Ysize, 0.25f);
        yield return new WaitForSeconds(0.5f);
        UI.SetActive(false);
        IsOpen = false;
        IsChanging = false;
    }

    public void SettingButton()
    {
        StartCoroutine(OpenVolumeSettings());
    }
    public IEnumerator OpenVolumeSettings()
    {
        VolumePanelTransform.anchoredPosition = new
Vector2(Xsize, 0);
        SettingsButtonsTransform.anchoredPosition = new
Vector2(-Xsize, 0);
        VolumeSettingPanel.SetActive(true);
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 68   |

```

        SettingsButtons.SetActive(true);
        VolumePanelTransform.DOAnchorPosX(0, 0.4f);
        MainButtonsTransform.DOAnchorPosX(-Xsize, 0.25f);
        yield return new WaitForSeconds(0.25f);
        SettingsButtonsTransform.DOAnchorPosX(0, 0.25f);
        MainButtons.SetActive(false);
    }

    public void BackButton()
    {
        StartCoroutine(Back());
    }

    public IEnumerator Back()
    {
        VolumePanelTransform.DOAnchorPosX(Xsize, 0.4f);
        SettingsButtonsTransform.DOAnchorPosX(-Xsize, 0.25f);
        MainButtonsTransform.anchoredPosition = new Vector2(-
Xsize, 0);

        MainButtons.SetActive(true);
        yield return new WaitForSeconds(0.25f);
        MainButtonsTransform.DOAnchorPosX(0, 0.25f);
        VolumeSettingPanel.SetActive(false);
        SettingsButtons.SetActive(false);
    }
}

```

#### Файл "DayController.cs"

```

using System.Collections;
using UnityEngine;
using DG.Tweening;

/// <summary>
/// класс для управления Esc меню. Крепится к главному
канвасу так как долже всегда выполняться
/// </summary>
public class EscMenuScript : UIBase
{

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 69   |

```

    /// <summary> ссылка на объект с Esc меню </summary>
    [SerializeField] private GameObject UI;
    /// <summary> ссылка на объект с главными кнопками для
реализации анимации </summary>
    [SerializeField] private GameObject MainButtons;
    /// <summary> ссылка на объект с кнопками настроек для
реализации анимации </summary>
    [SerializeField] private GameObject SettingsButtons;
    /// <summary> ссылка на объект с настройками звука для
реализации анимации </summary>
    [SerializeField] private GameObject VolumeSettingPanel;

    /// <summary> разрешение меню в пикселях для реализации
анимации </summary>
    [SerializeField] private float Xsize, Ysize;
    /// <summary> главный трансформ Esc меню </summary>
    private RectTransform UITransform;
    /// <summary> трансформ панели с главными кнопками
</summary>
    private RectTransform MainButtonsTransform;
    /// <summary> трансформ панели с кнопками
настроек</summary>
    private RectTransform SettingsButtonsTransform;
    /// <summary> трансформ панели настроек звука </summary>
    private RectTransform VolumePanelTransform;
    /// <summary> флаг активной анимации </summary>
    private bool IsChanging;
    void Start()
    {
        UITransform = UI.GetComponent<Transform>() as
RectTransform;
        MainButtonsTransform =
MainButtons.GetComponent<Transform>() as RectTransform;
        SettingsButtonsTransform =
SettingsButtons.GetComponent<Transform>() as RectTransform;

```

```

        VolumePanelTransform =
VolumeSettingPanel.GetComponent<Transform>() as RectTransform;
        IsOpen = false;
        IsChanging = false;
    }

    void LateUpdate()
    {

        if (Input.GetKeyDown(KeyCode.Escape) && !IsChanging
&& MenuController.singleton.OpenMenuCount == 0 &&
!MenuController.EscMenuFlag && !IsOpen)
        {
            MenuController.Regist(this, true, false);
        }
        if (MenuController.EscMenuFlag)
        {
            MenuController.EscMenuFlag = false;
        }
    }

    public override bool Open()
    {
        StartCoroutine(OpenMainMenu());
        return true;
    }

    public IEnumerator OpenMainMenu()
    {
        EscListen = false;
        IsChanging = true;
        MainButtonsTransform.anchoredPosition = new
Vector2(0, 0);
        VolumeSettingPanel.SetActive(false);
        SettingsButtons.SetActive(false);
        MainButtons.SetActive(true);
        UI.SetActive(true);
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 71   |

```

        UITransform.anchoredPosition = new Vector2(-Xsize,
0);

        UITransform.DOAnchorPosX(0, 0.25f);
        yield return new WaitForSeconds(0.5f);
        IsOpen = true;
        IsChanging=false;
        EscListen = true;
    }

    public void CloseEscMenuButton()
    {
        MenuController.Close(this);
    }
    public override void Close()
    {
        StartCoroutine(CloseMainMenu());
    }
    public IEnumerator CloseMainMenu()
    {
        IsChanging = true;
        UITransform.DOAnchorPosY(-Ysize, 0.25f);
        yield return new WaitForSeconds(0.5f);
        UI.SetActive(false);
        IsOpen = false;
        IsChanging = false;
    }

    public void SettingButton()
    {
        StartCoroutine(OpenVolumeSettings());
    }
    public IEnumerator OpenVolumeSettings()
    {
        VolumePanelTransform.anchoredPosition = new
Vector2(Xsize, 0);

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 72   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        SettingsButtonsTransform.anchoredPosition = new
Vector2(-Xsize, 0);
        VolumeSettingPanel.SetActive(true);
        SettingsButtons.SetActive(true);
        VolumePanelTransform.DOAnchorPosX(0, 0.4f);
        MainButtonsTransform.DOAnchorPosX(-Xsize, 0.25f);
        yield return new WaitForSeconds(0.25f);
        SettingsButtonsTransform.DOAnchorPosX(0, 0.25f);
        MainButtons.SetActive(false);
    }

    public void BackButton()
    {
        StartCoroutine(Back());
    }

    public IEnumerator Back()
    {
        VolumePanelTransform.DOAnchorPosX(Xsize, 0.4f);
        SettingsButtonsTransform.DOAnchorPosX(-Xsize, 0.25f);
        MainButtonsTransform.anchoredPosition = new Vector2(-
Xsize, 0);

        MainButtons.SetActive(true);
        yield return new WaitForSeconds(0.25f);
        MainButtonsTransform.DOAnchorPosX(0, 0.25f);
        VolumeSettingPanel.SetActive(false);
        SettingsButtons.SetActive(false);
    }
}

```

#### Файл "LinkToVolume.cs"

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class LinkToVolume : MonoBehaviour
{

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 73   |

```

        [SerializeField] private bool Global, Music, Radio,
Effects, Nature, Interface;

        [SerializeField] private float[] AddFloats;
        [SerializeField] private Slider[] AddSlider;
        [SerializeField] private AudioSource Source;

    void Start()
    {
        if (Source ==
null) gameObject.TryGetComponent<AudioSource>(out Source);
        if (Source != null)
        {
            if (Global) VolumeSettings.IsChangeGlobalVolume
+= ChangeVolume;
            if (Music) VolumeSettings.IsChangeMusicVolume +=
ChangeVolume;
            if (Radio) VolumeSettings.IsChangeRadioVolume +=
ChangeVolume;
            if (Effects) VolumeSettings.IsChangeEffectsVolume
+= ChangeVolume;
            if (Nature) VolumeSettings.IsChangeNatureVolume
+= ChangeVolume;
            if (Interface)
VolumeSettings.IsChangeInterfaceVolume += ChangeVolume;
            ChangeVolume();
        }
    }

    private void OnDestroy()
    {
        if (Source != null)
        {
            if (Global) VolumeSettings.IsChangeGlobalVolume -
= ChangeVolume;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 74   |

```

        if (Music) VolumeSettings.IsChangeMusicVolume -=
ChangeVolume;

        if (Radio) VolumeSettings.IsChangeRadioVolume -=
ChangeVolume;

        if (Effects) VolumeSettings.IsChangeEffectsVolume
-= ChangeVolume;

        if (Nature) VolumeSettings.IsChangeNatureVolume -
= ChangeVolume;

        if (Interface)
VolumeSettings.IsChangeInterfaceVolume -= ChangeVolume;
    }

}

public void ChangeVolume()
{
    float NewVolume = 1;
    if (Global) NewVolume *=
VolumeSettings.GetGlobalVolume();
    if (Music) NewVolume *=
VolumeSettings.GetMusicVolume();
    if (Radio) NewVolume *=
VolumeSettings.GetRadioVolume();
    if (Effects) NewVolume *=
VolumeSettings.GetEffectsVolume();
    if (Nature) NewVolume *=
VolumeSettings.GetNatureVolume();
    if (Interface) NewVolume *=
VolumeSettings.GetInterfaceVolume();
    if(AddFloats.Length != 0) foreach (float New in
AddFloats) NewVolume *= New;
    if(AddSlider.Length != 0) foreach (Slider New in
AddSlider) NewVolume *= New.value;
    Source.volume = NewVolume;
}
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 75   |

Файл "PlayerAnimationController.cs"

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerAnimationController : MonoBehaviour
{
    private PlayerMoveScript Move;

    [SerializeField] private Animator BodyAnimator;      //1
    [SerializeField] private Animator HairAnimator;      //2
    [SerializeField] private Animator ManeAnimator;      //3
    [SerializeField] private Animator TailAnimator;      //4
    [SerializeField] private Animator HornAnimator;      //5
    [SerializeField] private Animator MagicAnimator;     //6
    [SerializeField] private Animator EyeAnimator;       //7

    void Start()
    {
        Move = GetComponent<PlayerMoveScript>();
        gameObject.GetComponent<PlayerMoveScript>().IsStay +=
Stay;

        gameObject.GetComponent<PlayerMoveScript>().IsWalk +=
Walk;

        gameObject.GetComponent<PlayerMoveScript>().IsRun +=
Run;

        gameObject.GetComponent<PlayerMoveScript>().IsSit +=
Sit;
    }

    private void Stay()
    {
        ChangeAllStates(1);
    }

    private void Walk()
    {
        ChangeAllStates(2);
    }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 76   |

```

    }

    private void Run()
    {
        ChangeAllStates(3);
    }

    private void Sit()
    {
        ChangeAllStates(4);
    }

    private void ChangeAllStates(int NewState)
    {
        switch(NewState)
        {
            case 2:
            {
                ChangeSpeed(Move.GetWalkSpeed());
                break;
            }
            case 3:
            {
                ChangeSpeed(Move.GetRunSpeed() / 2);
                break;
            }
            default:
            {
                ChangeSpeed(1);
                break;
            }
        }

        BodyAnimator.SetInteger("State", NewState);
        TailAnimator.SetInteger("State", NewState);
        ManeAnimator.SetInteger("State", NewState);
        EyeAnimator.SetInteger("State", NewState);
        HornAnimator.SetInteger("State", NewState);
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 77   |

```

        MagicAnimator.SetInteger("State", NewState);
        HairAnimator.SetInteger("State", NewState);
    }

    private void ChangeSpeed(float speed)
    {
        BodyAnimator.speed = speed;
        TailAnimator.speed = speed;
        ManeAnimator.speed = speed;
        HairAnimator.speed = speed;
        MagicAnimator.speed = speed;
        EyeAnimator.speed = speed;
        HornAnimator.speed = speed;
    }
}

```

#### Файл "MagicController.cs"

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mirror;
using DG.Tweening;
using Mirror.Examples.Common;

public class MagicController : NetworkBehaviour
{
    [SerializeField] SpriteRenderer Magic;
    [SerializeField] [SyncVar(hook =
nameof(SwtichEffect))]private int ActiveCount;

    private void Start()
    {
        if (!isLocalPlayer) return;
        gameObject.GetComponent<LightSkill>().SkillActive +=
AddMagic;
    }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 78   |

```

        gameObject.GetComponent<LightSkill>().SkillDisable +=
DeletMagic;
    }

    [Command]
    public void AddMagic()
    {
        ActiveCount++;
    }

    [Command]
    public void DeletMagic()
    {
        int NewValue = ActiveCount - 1;
        if (NewValue < 0) NewValue = 0;
        ActiveCount = NewValue;
    }

    public void SwtichEffect(int Old, int New)
    {
        //if (ActiveCount > 0)
        //{
        //    Magic.DOColor(new Color(Magic.color.r,
Magic.color.g, Magic.color.b, 1), 0.4f);
        //}
        //else Magic.DOColor(new Color(Magic.color.r,
Magic.color.g, Magic.color.b, 0), 0.4f);

        if (ActiveCount > 0)
        {
            Magic.color = new Color(Magic.color.r,
Magic.color.g, Magic.color.b, 1);
        }
        else Magic.color = new Color(Magic.color.r,
Magic.color.g, Magic.color.b, 0);
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 79   |

```
}
```

### Файл "PlayerMoveScript.cs"

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mirror;
using Unity.VisualScripting;
public enum PlayerState
{
    Stay,
    Walk,
    Run,
    Sit
}

public class PlayerMoveScript : NetworkBehaviour
{

    public System.Action IsStay;
    public System.Action IsWalk;
    public System.Action IsRun;
    public System.Action IsSit;

    private PlayerState CurrentState; //1-stay 2-Walk 3-Run
4-Sit

    private Transform PlayerTransform;
    [SerializeField] private float WalkSpeed;
    [SerializeField] private float RunSpeed;
    private float CurrentSpeed;
    private Vector2 MoveVector;
    private bool CapsLock;
```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 80   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

private void Awake()
{
    PlayerTransform = gameObject.transform;
}
private void Start()
{
    if (!isLocalPlayer) return;
    CurrentState = PlayerState.Stay;
    CapsLock = false;
    CurrentSpeed = WalkSpeed;

    Camera.main.GetComponent<CameraControllerScript>().SetTarger(Playe
rTransform);
}
void Update()
{
    if (!isLocalPlayer) return;

    if (Input.GetKeyDown(KeyCode.KeypadPlus))
ChangeRunSpeed(1);
    if (Input.GetKeyDown(KeyCode.KeypadMinus))
ChangeRunSpeed(-1);

    MoveVector.x = 0;
    MoveVector.y = 0;

    if (PlayerController.CanMove)
    {
        MoveVector.x = Input.GetAxisRaw("Horizontal");
        MoveVector.y = Input.GetAxisRaw("Vertical");
    }

    if (Input.GetKeyDown(KeyCode.CapsLock))

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 81   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        {
            if (CurrentSpeed == WalkSpeed) CurrentSpeed =
RunSpeed;

            else CurrentSpeed = WalkSpeed;
            CapsLock = !CapsLock;
        }

        if (Input.GetKeyDown(KeyCode.LeftShift))
        {
            if (!CapsLock) CurrentSpeed = RunSpeed;
            else CurrentSpeed = WalkSpeed;
        }
        if (Input.GetKeyUp(KeyCode.LeftShift))
        {
            if (!CapsLock) CurrentSpeed = WalkSpeed;
            else CurrentSpeed = RunSpeed;
        }

        if (CurrentState == PlayerState.Stay)
        {
            if (Input.GetKeyDown(KeyCode.LeftControl) &&
PlayerController.CanMove)
            {
                IsSit?.Invoke();
                CurrentState = PlayerState.Sit;
                return;
            }

            if (MoveVector.x != 0 || MoveVector.y != 0)
            {
                if (CurrentSpeed == WalkSpeed)
                {
                    IsWalk?.Invoke();
                    CurrentState = PlayerState.Walk;
                    return;
                }
            }
        }
    }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 82   |

```

        }
        if (CurrentSpeed == RunSpeed)
        {
            IsRun?.Invoke();
            CurrentState = PlayerState.Run;
            return;
        }
    }

    if (CurrentState == PlayerState.Walk)
    {
        if (MoveVector.x == 0 && MoveVector.y == 0)
        {
            IsStay?.Invoke();
            CurrentState = PlayerState.Stay;
            return;
        }
        if (CurrentSpeed == RunSpeed)
        {
            IsRun?.Invoke();
            CurrentState = PlayerState.Run;
            return;
        }
    }

    if (CurrentState == PlayerState.Run)
    {
        if (MoveVector.x == 0 && MoveVector.y == 0)
        {
            IsStay?.Invoke();
            CurrentState = PlayerState.Stay;
            return;
        }
        if (CurrentSpeed == WalkSpeed)
        {

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 83   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        IsWalk?.Invoke();
        CurrentState = PlayerState.Walk;
        return;
    }
}

if (CurrentState == PlayerState.Sit)
{
    if (Input.GetKeyDown(KeyCode.LeftControl) &&
        PlayerController.CanMove)
    {
        IsStay?.Invoke();
        CurrentState = PlayerState.Stay;
        return;
    }

    if (MoveVector.x != 0 || MoveVector.y != 0)
    {
        if (CurrentSpeed == WalkSpeed)
        {
            IsWalk?.Invoke();
            CurrentState = PlayerState.Walk;
            return;
        }
        if (CurrentSpeed == RunSpeed)
        {
            IsRun?.Invoke();
            CurrentState = PlayerState.Run;
            return;
        }
    }
}

if (MoveVector.x != 0 || MoveVector.y != 0)
{

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 84   |

```

        MoveVector.Normalize();

        PlayerTransform.position = new
Vector3(PlayerTransform.position.x + MoveVector.x * CurrentSpeed *
Time.deltaTime, PlayerTransform.position.y + MoveVector.y *
CurrentSpeed * Time.deltaTime, PlayerTransform.position.z);
        if (MoveVector.x > 0) PlayerTransform.localScale
= new Vector3(-1, 1, 1);
        if (MoveVector.x < 0) PlayerTransform.localScale
= new Vector3(1, 1, 1);
    }
}

private void FixedUpdate()
{

}

public void Exit()
{
    if(isClient)
    {
        NetworkClient.Disconnect();
    }
    if (isServer)
    {
        NetworkManager.singleton.StopServer();
    }
}

public float GetWalkSpeed()
{
    return WalkSpeed;
}

public float GetRunSpeed()
{
    return RunSpeed;
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 85   |

```

        public void ChangeRunSpeed(int Mod)
        {
            RunSpeed += Mod;
            if(RunSpeed < 1) RunSpeed = 1;
        }
    }

```

#### Файл "PlayerPart.cs"

```

using Mirror;
using UnityEngine;

/// <summary>
/// Хранит в себе текущий скин и меняет его
/// </summary>
public class PlayerPart: NetworkBehaviour
{
    [SerializeField][SyncVar(hook = nameof(SwitchPart))]
private int State;
    [SerializeField][SyncVar(hook = nameof(SwitchColor))]
private Color color;
    [SyncVar(hook = nameof(SwitchSecondColor))] private Color
SecondColor;

    /// <summary>
    /// Сетевая оболочка для SwitchSecondColor
    /// </summary>
    /// <param name="NewColor"></param>
    [Command]
    public void SwitchSecondColorCMD(Color NewColor)
    {
        SecondColor = NewColor;
    }

    /// <summary>
    /// Меняют дополнительный цвет
    /// </summary>

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 86   |

```

    /// <param name="Old"></param>
    /// <param name="New"></param>
    private void SwitchSecondColor(Color Old, Color New)
    {
        SpriteRenderer SecondSprite = null;
        if (transform.childCount > 0)
        {
            SecondSprite =
transform.GetChild(0).GetComponent<SpriteRenderer>();
        }
        else return;
        if (SecondSprite != null) SecondSprite.color = new
Color(New.r, New.g, New.b, SecondSprite.color.a);
    }

    /// <summary>
    /// Сетевая оболочка для SwitchColor
    /// </summary>
    /// <param name="NewColor"></param>
    [Command]
    public void SwitchColorCMD(Color NewColor)
    {
        color = NewColor;
    }

    /// <summary>
    /// Меняет основной цвет
    /// </summary>
    /// <param name="Old"></param>
    /// <param name="New"></param>
    private void SwitchColor(Color Old, Color New)
    {
        SpriteRenderer Buffer =
gameObject.GetComponent<SpriteRenderer>();
        Buffer.color = new
Color(New.r, New.g, New.b, Buffer.color.a);
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 87   |

```

    /// <summary>
    /// Сетевая оболочка для SwitchPart
    /// </summary>
    /// <param name="Index"></param>
    [Command]
    public void SwitchPartCMD(int Index)
    {
        State = Index;
    }

    /// <summary>
    /// Меняет текущую часть на новую по ID
    /// </summary>
    /// <param name="Old"></param>
    /// <param name="New"></param>
    private void SwitchPart(int Old, int New)
    {
        PlayerPartSkin Skin =
Resources.Load<PlayerPartSkin>($"PlayerParts/{gameObject.name}/{New}");

        if(Skin == null)
        {
            Debug.LogError("Не удалось загрузить скин");
            return;
        }
        AnimatorOverrideController NewAnimator = Skin.Skin;
        if (NewAnimator == null)
        {
            Debug.LogError("Не удалось получить аниматор
скина");

            return;
        }
        Animator Animator;
        Animator = GetComponent<Animator>();
        if (Animator != null)

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 88   |

```

        {
            Animator.runtimeAnimatorController = NewAnimator;
        }
        else Debug.LogError("Ошибка при изменении
аниматора");
    }
}

```

#### Файл "PlayerSound.cs"

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mirror;

public class PlayerSoundControl : NetworkBehaviour
{
    [SerializeField] private AudioClip RunSound;
    [SerializeField] private AudioClip WalkSound;
    private AudioSource audioSource;
    private bool IsPlaying;
    private int CurrentState;

    private void Awake()
    {
        if (!isLocalPlayer) return;
        //PlayerNetworkManager.IsConnected += SyncSound;
    }
    void Start()
    {
        audioSource = GetComponent<AudioSource>();
        IsPlaying = false;
        CurrentState = 0;
        if (!isLocalPlayer) return;
        gameObject.GetComponent<PlayerMoveScript>().IsRun +=
Run;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 89   |

```

        gameObject.GetComponent<PlayerMoveScript>().IsWalk +=
Walk;
        gameObject.GetComponent<PlayerMoveScript>().IsStay +=
Stop;
        gameObject.GetComponent<PlayerMoveScript>().IsSit +=
Stop;
    }

    public void Run()
    {
        if (isLocalPlayer) RunCMD();
    }
    [Command]
    public void RunCMD()
    {
        RunRPC();
    }
    [ClientRpc]
    public void RunRPC()
    {
        audioSource.clip = RunSound;
        audioSource.Play();
        IsPlaying = true;
        CurrentState = 2;
    }

    public void Walk()
    {
        if (isLocalPlayer) WalkCMD();
    }
    [Command]
    public void WalkCMD()
    {
        WalkRPC();
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 90   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

[ClientRpc]
public void WalkRPC()
{
    audioSource.clip = WalkSound;
    audioSource.Play();
    IsPlaying = true;
    CurrentState = 1;
}

public void Stop()
{
    StopCMD(IsPlaying);
}

[Command]
public void StopCMD(bool Active)
{
    StopRPC(Active);
}

[ClientRpc]
public void StopRPC(bool Active)
{
    if (!IsPlaying) return;
    audioSource.Stop();
    IsPlaying = false;
    CurrentState = 0;
}

public void SyncSound()
{
    if (isLocalPlayer) SyncSoundCMD(IsPlaying,
CurrentState);
}

[Command]
public void SyncSoundCMD(bool Active, int State)
{
    SyncSoundRPC(Active, State);
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 91   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

    }

    [ClientRpc]
    public void SyncSoundRPC(bool Active, int State)
    {
        if (Active)
        {
            if (State == 1) audioSource.clip = WalkSound;
            if (State == 2) audioSource.clip = RunSound;
            audioSource.Play();
            IsPlaying=true;
        }
        else
        {
            IsPlaying=false;
            CurrentState=0;
        }
    }
}

```

#### Файл "PlayerUse.cs"

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mirror;

public class PlayerUse : NetworkBehaviour
{
    private Transform PlayerTransform;
    [SerializeField] private float MaxDistanse;
    // Start is called before the first frame update
    void Awake()
    {
        PlayerTransform = gameObject.transform;
    }

    // Update is called once per frame
    void Update()

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 92   |

```

        {
            if (!isLocalPlayer) return;
            if (Input.GetKeyDown(KeyCode.Mouse0) &&
PlayerController.CanMove) Use();
        }

        public void Use()
        {
            Vector2 MousePosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
            if (MaxDistanse - Vector2.Distance(MousePosition,
PlayerTransform.position) <= 0) return;
            Collider2D Object =
Physics2D.OverlapPoint(MousePosition);
            UsableObject UO;
            if (Object != null &&
Object.TryGetComponent<UsableObject>(out UO)) UO.Use();
        }
    }

```

#### Файл "CreateMenuScript.cs"

```

using DG.Tweening;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

/// <summary>
/// главный скрипт для управления редактором персонажа
/// </summary>
public class CreateMenuScript : UIBase
{
    /// <summary>ссылка на объект с редактором</summary>
    [SerializeField] private GameObject CreateMenu;

```

```
/// <summary> ссылка на панель для блокировки меню  
редактирования </summary>
```

```
[SerializeField] private GameObject BlockPanelRef;  
public static GameObject BlockPanel;
```

```
/// <summary> звук выбора цвета </summary>
```

```
[SerializeField] private AudioClip PickSoundRef;  
public static AudioClip PickSound;
```

```
/// <summary> список страниц редактора </summary>
```

```
[SerializeField] private List<GameObject> Pages;
```

```
/// <summary> индекс текущей страницы </summary>
```

```
private int Index;
```

```
/// <summary> флаг активной анимации </summary>
```

```
private bool IsChanging;
```

```
[SerializeField] private PartSettingScript Body;
```

```
[SerializeField] private PartSettingScript Hair;
```

```
[SerializeField] private PartSettingScript HairSec;
```

```
[SerializeField] private PartSettingScript Mane;
```

```
[SerializeField] private PartSettingScript ManeSec;
```

```
[SerializeField] private PartSettingScript Tail;
```

```
[SerializeField] private PartSettingScript TailSec;
```

```
[SerializeField] private PartSettingScript Eye;
```

```
[SerializeField] private PartSettingScript Magic;
```

```
[SerializeField] private SkinSettingScript BodySkin;
```

```
[SerializeField] private SkinSettingScript HairSkin;
```

```
[SerializeField] private SkinSettingScript ManeSkin;
```

```
[SerializeField] private SkinSettingScript TailSkin;
```

```
[SerializeField] private SkinSettingScript EyeSkin;
```

```
private void Start()
```

```
{
```

```
    MenuController.Regist(this, false, true);
```

```

        BlockPanel = BlockPanelRef;
        PickSound = PickSoundRef;
        Index = 0;
        IsChanging = false;

Pages[0].GetComponent<RectTransform>().anchoredPosition = new
Vector2(33f, 0.5f);
        Pages[0].SetActive(true);
        for(int i = 1; i < Pages.Count; i++)
        {
            Pages[i].SetActive(false);
        }
    }

    public void NextPageButton()
    {
        if (!IsChanging) StartCoroutine(NextPage());
    }

    private IEnumerator NextPage()
    {
        IsChanging = true;
        int NextIndex = Index + 1;
        if (NextIndex > Pages.Count - 1) NextIndex = 0;

Pages[NextIndex].GetComponent<RectTransform>().anchoredPosition =
new Vector2(33f, 67.5f);
        Pages[NextIndex].SetActive(true);

Pages[Index].GetComponent<RectTransform>().DOAnchorPosY(-
66.5f, 0.3f);

Pages[NextIndex].GetComponent<RectTransform>().DOAnchorPosY(0.5f,
0.3f);

        yield return new WaitForSeconds(0.3f);
        Pages[Index].SetActive(false);
        Index = NextIndex;
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 95   |

```

        IsChanging = false;
    }

    public void PrevPageButton()
    {
        if (!IsChanging) StartCoroutine(PrevPage());
    }

    private IEnumerator PrevPage()
    {
        IsChanging = true;
        int NextIndex = Index - 1;
        if (NextIndex < 0) NextIndex = Pages.Count-1;

        Pages[NextIndex].GetComponent<RectTransform>().anchoredPosition =
        new Vector2(33f, -66.5f);

        Pages[NextIndex].SetActive(true);

        Pages[Index].GetComponent<RectTransform>().DOAnchorPosY(67.5f,
        0.3f);

        Pages[NextIndex].GetComponent<RectTransform>().DOAnchorPosY(0.5f,
        0.3f);

        yield return new WaitForSeconds(0.3f);
        Pages[Index].SetActive(false);
        Index = NextIndex;
        IsChanging = false;
    }

    public void ApplySkin()
    {
        PlayerModelController PMC =
        PlayerController.PlayerObject.GetComponent<PlayerModelController>(
        );
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 96   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

//Body
PMC.GetBody().SwitchPartCMD(BodySkin.GetIndex());
PMC.GetBody().SwitchColorCMD(Body.GetColor());

//Hair
PMC.GetHair().SwitchPartCMD(HairSkin.GetIndex());
PMC.GetHair().SwitchColorCMD(Hair.GetColor());

PMC.GetHair().SwitchSecondColorCMD(HairSec.GetColor());

//Tail
PMC.GetTail().SwitchPartCMD(TailSkin.GetIndex());
PMC.GetTail().SwitchColorCMD(Tail.GetColor());

PMC.GetTail().SwitchSecondColorCMD(TailSec.GetColor());

//Mane
PMC.GetMane().SwitchPartCMD(ManeSkin.GetIndex());
PMC.GetMane().SwitchColorCMD(Mane.GetColor());

PMC.GetMane().SwitchSecondColorCMD(ManeSec.GetColor());

//Eye
PMC.GetEye().SwitchPartCMD(EyeSkin.GetIndex());
PMC.GetEye().SwitchSecondColorCMD(Eye.GetColor());

//Horn
PMC.GetHorn().SwitchPartCMD(1);
PMC.GetHorn().SwitchColorCMD(Body.GetColor());

//Magic
PMC.GetMagic().SwitchPartCMD(1);
PMC.GetMagic().SwitchColorCMD(Magic.GetColor());

StartCoroutine(CloseMenu());
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 97   |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        public IEnumerator CloseMenu()
        {

CreateMenu.GetComponent<RectTransform>().DOAnchorPosY(-90, 0.25f);
            yield return new WaitForSeconds(0.25f);
            MenuController.Close(this);
        }
    }

```

#### Файл "PartSettingScript.cs"

```

using UnityEngine;
using UnityEngine.UI;

/// <summary>
/// скрипт для управления цветом модели в редакторе
/// </summary>
public class PartSettingScript : MonoBehaviour
{
    [SerializeField] private Image[] PartPreview;

    [SerializeField] private Slider H;
    [SerializeField] private Slider V;
    [SerializeField] private Slider S;

    private bool IsPickColor;

    private void Start()
    {
        IsPickColor = false;
    }

    public void ChangeColor()

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 98   |

```

        {
            foreach (var part in PartPreview)
            {
                part.color = Color.HSVToRGB(H.value, S.value,
V.value);
            }
        }

        private void Update()
        {
            if ((IsPickColor) &&
Input.GetKeyDown(KeyCode.Mouse0))
            {
                PickColor();
            }
        }

        public void PickColorButton()
        {
            IsPickColor = true;
            CreateMenuScript.BlockPanel.SetActive(true);
        }

        public void PickColor()
        {
            Color Inp = new Color(0, 0, 0);
            Vector2 mousePos = Input.mousePosition;
            int targetWidth = Screen.width;
            int targetHeight = Screen.height;
            RenderTexture renderTexture = new
RenderTexture(targetWidth, targetHeight, 32,
RenderTextureFormat.ARGB32);
            Camera.main.targetTexture = renderTexture;

            Camera.main.Render();

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 99   |

```

Texture2D capturedScreenshot = new
Texture2D(targetWidth, targetHeight, TextureFormat.ARGB32, false);
RenderTexture.active = renderTexture;
capturedScreenshot.ReadPixels(new Rect(0, 0,
targetWidth, targetHeight), 0, 0);
capturedScreenshot.Apply();
RenderTexture.active = null;

Camera.main.targetTexture = null;
Destroy(renderTexture);
if (capturedScreenshot != null)
{
    int x = (int)mousePos.x;
    int y = (int)mousePos.y;
    if (x >= 0 && x < Screen.width && y >= 0 && y <
Screen.height)
    {
        Inp = capturedScreenshot.GetPixel(x, y);
    }
}

float Hc, Vc, Sc;
Color.RGBToHSV(Inp, out Hc, out Sc, out Vc);
if (Vc < V.minValue) Vc = V.minValue;
H.value = Hc;
V.value = Vc;
S.value = Sc;

GlobalAudio.Play(CreateMenuScript.PickSound);
ChangeColor();
IsPickColor = false;
CreateMenuScript.BlockPanel.SetActive(false);
}

public Color GetColor() { return PartPreview[0].color; }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 100  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## Файл "SkinSetting.cs"

```

using UnityEngine;
using UnityEngine.UI;

/// <summary>
/// скрипт для управления одной страницей редактора
/// </summary>
public class SkinSettingScript : MonoBehaviour
{
    private int Index;
    private int MaxIndex;
    [SerializeField] private string PartName;
    [SerializeField] private Image PartPreview;
    [SerializeField] private Image PartPreviewSec;

    private void Awake()
    {
        Index = 1;
        PlayerPartSkin[] Buf =
Resources.LoadAll<PlayerPartSkin>($"PlayerParts/{PartName}");
        MaxIndex = Buf.Length;
    }

    public void NextSkin()
    {
        Index++;
        if (Index > MaxIndex) Index = 1;
        PlayerPartSkin buf =
Resources.Load<PlayerPartSkin>($"PlayerParts/{PartName}/{Index}");
        PartPreview.sprite = buf.MainSprite;
        if(PartPreviewSec != null) PartPreviewSec.sprite =
buf.SecondSprite;
    }

    public void PrevSkin()

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 101  |

```

        {
            Index--;
            if (Index < 1) Index = MaxIndex;
            PlayerPartSkin buf =
Resources.Load<PlayerPartSkin>($"PlayerParts/{PartName}/{Index}");
            PartPreview.sprite = buf.MainSprite;
            if (PartPreviewSec != null) PartPreviewSec.sprite =
buf.SecondSprite;
        }

        public int GetIndex() { return Index; }
    }

```

#### Файл "DebugMenuScript.cs"

```

using TMPro;
using UnityEngine;

/// <summary>
/// класс для управления дебаг меню
/// </summary>
public class DebugMenu : MonoBehaviour
{
    private static DebugMenu _main;
    public static DebugMenu main { get { return _main; } }

    /// <summary> ссылка на текст с ФПС </summary>
    [SerializeField] private TextMeshProUGUI _FPS_text;
    /// <summary> счётчик ФПС </summary>
    private int _CurrentFPS;
    /// <summary> таймер для вывода фпс </summary>
    private float _CurrentTimeFPS;

    /// <summary> ссылка на текст для вывода времени и даты
    </summary>
    [SerializeField] private TextMeshProUGUI _TimeData_text;
    /// <summary> публичная ссылка для вывода времени и даты
    </summary>

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 102  |

```

        public TextMeshProUGUI TimeData_text { get { return
_TimeData_text; } }
        void Start()
        {
            if (_main == null) _main = this;
            else Debug.LogWarning("попытка создать второй
синглтон (DebugMenu)");
            _CurrentFPS = 0;
            _CurrentTimeFPS = 1f;
            gameObject.SetActive(false);
        }

        void Update()
        {
            CheckFPS();
        }

        /// <summary> метод для расчёта и вывода фпс. вызывается
в update </summary>
        private void CheckFPS()
        {
            _CurrentTimeFPS -= Time.deltaTime;
            _CurrentFPS++;
            if( _CurrentTimeFPS < 0 )
            {
                _FPS_text.text = "FPS: " + _CurrentFPS;
                _CurrentFPS = 0;
                _CurrentTimeFPS = 1f;
            }
        }
    }

```

#### Файл "MenuController.cs"

```

using System.Collections.Generic;
using System.Linq;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 103  |

```

using UnityEngine;

/// <summary>
/// Базовый класс для элементов интерфейса для их контроля
/// </summary>
public abstract class UIbase:MonoBehaviour
{
    /// <summary> открыт ли интерфейс </summary>
    public bool IsOpen;

    /// <summary> используется ли Esc для закрытия интерфейса
</summary>
    public bool EscListen;

    /// <summary> уничтожается ли интерфейс при закрытии
</summary>
    public bool IsDestoy;

    /// <summary> метод для открытия интерфейса </summary>
    public virtual bool Open()
    { return true; }

    /// <summary> метод для закрытия интерфейса </summary>
    public virtual void Close()
    { }
}

/// <summary>
/// главный класс, контролирующий элементы интерфейса
/// </summary>
public class MenuController : MonoBehaviour
{
    /// <summary> флаг для корректного открытия Esc меню
</summary>
    public static bool EscMenuFlag;

```

```

        /// <summary> список всех дополнительных открытых
элементов интерфейса </summary>
        public List<UIbase> OpenUI = new List<UIbase>();

        /// <summary> количество дополнительных открытых
элементов интерфейса </summary>
        public int OpenMenuCount { get { return OpenUI.Count; }
    }

        /// <summary> ссылка на объект с интерфейсом дебаг меню
</summary>
        [SerializeField] private GameObject DebugMenuObjectRef;

        /// <summary> ссылка на объект с интерфейсом меню помощи
</summary>
        [SerializeField] private GameObject HelpMenu;

        /// <summary> доступ к дебаг меню для других классов
</summary>
        public static DebugMenu DebugMenuObj;

        /// <summary> ссылка на главный канвас для UI элементов
</summary>
        [SerializeField] private GameObject _mainUICanvas;
        /// <summary> публичная ссылка на главный канвас для UI
элементов </summary>
        public GameObject mainUICanvas { get { return
        _mainUICanvas; } }

        /// <summary> ссылка на канвас для UI элементов игрока
</summary>
        [SerializeField] private GameObject _playerUICanvas;
        /// <summary> ссылка на канвас для UI элементов игрока
</summary>
        public GameObject playerUICanvas { get { return
        _playerUICanvas; } }

```

```

    /// <summary> синглтон </summary>
    private static MenuController _singleton;
    /// <summary> синглтон </summary>
    public static MenuController singleton { get { return
_singleton; } }

    public void Load()
    {
        if (_singleton == null) _singleton = this;
        else Debug.LogWarning("попытка присвоить второй
синглтон (MenuController)");
        gameObject.GetComponent<Canvas>().worldCamera =
Camera.main;
        DebugMenuObj =
DebugMenuObjectRef.GetComponent<DebugMenu>();
        EscMenuFlag = false;
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.F1))
        {
            if (DebugMenuObjectRef.activeSelf)
DebugMenuObjectRef.SetActive(false);
            HelpMenu.SetActive(!HelpMenu.activeSelf);
        }
        if (Input.GetKeyDown(KeyCode.F2))
        {
            if (HelpMenu.activeSelf)
HelpMenu.SetActive(false);

DebugMenuObjectRef.SetActive(!DebugMenuObjectRef.activeSelf);
        }

        if (Input.GetKeyDown(KeyCode.Escape))
        {

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 106  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        if (OpenMenuCount != 0 &&
OpenUI.Last().EscListen)
        {
            UIBase UIB = OpenUI.Last();
            UIB.Close();
            OpenUI.Remove(UIB);
            if (OpenMenuCount == 0)
            {
                EscMenuFlag = true;
            }
        }
        if (OpenMenuCount == 0)
        {
            PlayerController.CanMove = true;
        }
        if (singleton.OpenMenuCount == 0)
        {
            singleton.playerUICanvas.SetActive(true);
        }
        else
        {
            singleton.playerUICanvas.SetActive(false);
        }
    }
}

/// <summary> добавляет интерфейс в общий список и
открывает его </summary>
public static void Regist(UIBase UI, bool EscActive =
true, bool IsDestroy = false)
{
    if (!UI.Open()) return;
    PlayerController.CanMove = false;
    UI.EscListen = EscActive;
    UI.IsDestoy = IsDestroy;
    singleton.OpenUI.Add(UI);
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 107  |

```

        if (singleton.OpenMenuCount == 0)
        {
            singleton.playerUICanvas.SetActive(true);
        }
        else
        {
            singleton.playerUICanvas.SetActive(false);
        }
    }

    /// <summary> удаляет интерфейс из общего списка и
закрывает его </summary>
    public static void Close(UIbase UI)
    {
        if (!singleton.OpenUI.Contains(UI)) return;
        UI.Close();
        singleton.OpenUI.Remove(UI);
        if (UI.IsDestoy) Destroy(UI.gameObject);
        if (singleton.OpenMenuCount == 0)
        {
            EscMenuFlag = true;
            PlayerController.CanMove = true;
        }
        if (singleton.OpenMenuCount == 0)
        {
            singleton.playerUICanvas.SetActive(true);
        }
        else
        {
            singleton.playerUICanvas.SetActive(false);
        }
    }
}

```

Файл "VolumeSetting.cs"

```

using System.Collections;
using System.Collections.Generic;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 108  |

```

using UnityEngine;

public static class VolumeSettings
{
    public static System.Action IsChangeGlobalVolume;
    public static System.Action IsChangeMusicVolume;
    public static System.Action IsChangeRadioVolume;
    public static System.Action IsChangeEffectsVolume;
    public static System.Action IsChangeNatureVolume;
    public static System.Action IsChangeInterfaceVolume;

    private static float GlobalVolume = 1;
    private static float MusicVolume = 1;
    private static float RadioVolume = 1;
    private static float EffectsVolume = 1;
    private static float NatureVolume = 1;
    private static float InterfaceVolume = 1;

    public static void ChangeGlobalVolume(float NewVolume)
    {
        GlobalVolume = NewVolume;
        if(GlobalVolume < 0) GlobalVolume = 0;
        if(GlobalVolume > 1) GlobalVolume = 1;
        IsChangeGlobalVolume?.Invoke();
    }

    public static float GetGlobalVolume() { return
GlobalVolume; }

    public static void ChangeMusicVolume(float NewVolume)
    {
        MusicVolume = NewVolume;
        if (MusicVolume < 0) MusicVolume = 0;
        if (MusicVolume > 1) MusicVolume = 1;
        IsChangeMusicVolume?.Invoke();
    }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 109  |

```

        public static float GetMusicVolume() { return
MusicVolume; }

        public static void ChangeRadioVolume(float NewVolume)
        {
            RadioVolume = NewVolume;
            if (RadioVolume < 0) RadioVolume = 0;
            if (RadioVolume > 1) RadioVolume = 1;
            IsChangeRadioVolume?.Invoke();
        }

        public static float GetRadioVolume() { return
RadioVolume; }

        public static void ChangeEffectsVolume(float NewVolume)
        {
            EffectsVolume = NewVolume;
            if (EffectsVolume < 0) EffectsVolume = 0;
            if (EffectsVolume > 1) EffectsVolume = 1;
            IsChangeEffectsVolume?.Invoke();
        }

        public static float GetEffectsVolume() { return
EffectsVolume; }

        public static void ChangeNatureVolume(float NewVolume)
        {
            NatureVolume = NewVolume;
            if (NatureVolume < 0) NatureVolume = 0;
            if (NatureVolume > 1) NatureVolume = 1;
            IsChangeNatureVolume?.Invoke();
        }

        public static float GetNatureVolume() { return
NatureVolume; }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 110  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        public static void ChangeInterfaceVolume(float NewVolume)
        {
            InterfaceVolume = NewVolume;
            if (InterfaceVolume < 0) InterfaceVolume = 0;
            if (InterfaceVolume > 1) InterfaceVolume = 1;
            IsChangeInterfaceVolume?.Invoke();
        }

        public static float GetInterfaceVolume() { return
InterfaceVolume; }

    }

```

Файл "GameSettings.cs"

```

public enum NetType
{
    Singleplayer,
    Host,
    Connected,
}

public enum GameType
{
    NewGame,
    Continue,
    Connected,
}

```

```

/// <summary>
/// хранит конфигурацию текущей игры
/// </summary>
public static class GameSettings
{
    /// <summary> сетевой тип игры </summary>

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 111  |

```

private static NetType _gameNetType;
/// <summary> публичный сетевой тип игры </summary>
public static NetType GameNetType
{
    get { return _gameNetType; }
    set { _gameNetType = value; }
}

/// <summary> тип игры </summary>
private static GameType _gameType;
/// <summary> публичный тип игры </summary>
public static GameType gameType
{
    get { return _gameType; }
    set { _gameType = value; }
}
}

```

#### Файл "InventoruCell.cs"

```

using Mirror;
using UnityEngine;

public class InventoryCell : MonoBehaviour
{
    public System.Action CellUpdated;

    /// <summary> Текущий предмет в ячейке </summary>
    private Item _item;
    /// <summary> Текущий предмет в ячейке </summary>
    public Item item { get { return _item; } }

    /// <summary> ID предмета </summary>
    //[SyncVar(hook = nameof(SyncCellID))]
    private int _itemID;
    /// <summary> ID предмета </summary>
    public int itemID { get { return _itemID; } }
}

```

```

/// <summary> Количество предметов в ячейке </summary>
//[SyncVar(hook = nameof(SyncCellCount))]
private int _itemCount;
/// <summary> Количество предметов в ячейке </summary>
public int itemCount { get { return _itemCount; } }

/// <summary> доступна ли ячейка </summary>
private bool _canTake;
/// <summary> доступна ли ячейка </summary>
public bool canTake
{ get { return _canTake; }}

//[SyncVar(hook = nameof(SyncCellQuality))]
private Quality _quality;

/// <summary> Тип предмета в ячейке </summary>
private ItemType _itemType;
/// <summary> Тип предмета в ячейке </summary>
public ItemType itemType { get { return _itemType; }}

private void Awake()
{
    _item = new Item();
    _itemID = -1;
    _itemCount = 0;
    _quality = Quality.None;
}

public void SyncCellCount(int Old, int New)
{
    _itemCount = New;
    CellUpdated?.Invoke();
}

public void SyncCellID(int Old, int New)
{

```

```

        _item.CopyItem(New);
        _itemID = New;
        CellUpdated?.Invoke();
    }

    public void SyncCellQuality(Quality Old, Quality New)
    {
        _quality = New;
        CellUpdated?.Invoke();
    }

    //[Command(requiresAuthority = false)]
    public void addItemCMD(int ID, int Count, Quality
ItemQuality)
    {
        addItem(ID, Count, ItemQuality);
    }

    public bool addItem(int ID, int Count, Quality
ItemQuality)
    {
        if ((_item.id != -1 && _quality != ItemQuality) ||
(_item.id != -1 && _item.id != ID)) return false;

        if (_itemCount == 0) _item.CopyItem(ID);

        if (_itemCount + 1 > _item.maxStackCount) return
false;

        _itemCount += Count;
        _quality = ItemQuality;
        CellUpdated?.Invoke();
        _itemID = ID;
        return true;
    }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 114  |

```

public void DeleteItem()
{
    _item = new Item();
    _itemID = -1;
    _itemCount = 0;
    _quality = Quality.None;
    CellUpdated?.Invoke();
}

public void DeleteItem(int count)
{
    _itemCount = _itemCount - count;
    if (_itemCount <= 0)
    {
        _item = new Item();
        _itemID = -1;
        _itemCount = 0;
        _quality = Quality.None;
    }
    CellUpdated?.Invoke();
}

//[Command(requiresAuthority = true)]
public void SwitchCellCMD(InventoryCell NewCell, int
Count)
{
    SwitchCellRPC(NewCell, Count);
    if (_itemType != ItemType.None &&
NewCell.item.itemType != _itemType) return;
    if (NewCell.item != _item)
    {
        Item itemBuf = _item;
        int itemCountBuff = _itemCount;
        _item = NewCell._item;
        _itemCount = NewCell._itemCount;
        NewCell._item = itemBuf;
    }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 115  |

```

        NewCell._itemCount = itemCountBuff;
    }
}

//[ClientRpc]
public void SwitchCellRPC(InventoryCell NewCell, int
Count)
{
    if (_itemType != ItemType.None &&
NewCell.item.itemType != _itemType) return;
    if (NewCell.item != _item)
    {
        Item itemBuf = _item;
        int itemCountBuff = _itemCount;
        _item = NewCell._item;
        _itemCount = NewCell._itemCount;
        NewCell._item = itemBuf;
        NewCell._itemCount = itemCountBuff;
    }

    CellUpdated?.Invoke();
}
}

```

#### Файл "InventoruCellui.cs"

```

using System.Collections;
using System.Collections.Generic;
using TMPro;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(InventoryCell))]
public class InventoryCellUi : MonoBehaviour
{
    [SerializeField] private InventoryCell _connectedCell;

```

```

[SerializeField] private Image itemImage;
[SerializeField] private TextMeshProUGUI itemCountText;

private void Start()
{
    _connectedCell = GetComponent<InventoryCell>();
    itemCountText.gameObject.SetActive(false);
    itemImage.gameObject.SetActive(false);
    _connectedCell.CellUpdated += UpdateCellVisual;
    UpdateCellVisual();
}

private void OnDestroy()
{
    _connectedCell.CellUpdated -= UpdateCellVisual;
}

public void UpdateCellVisual()
{
    if (_connectedCell.itemCount > 1)
    {
        itemCountText.gameObject.SetActive(true);
        itemCountText.text = "" +
_connectedCell.itemCount;
    }
    else
    {
        itemCountText.gameObject.SetActive(false);
    }

    if (_connectedCell.item.itemIcon != null)
    {
        itemImage.gameObject.SetActive(true);
        itemImage.sprite = _connectedCell.item.itemIcon;
    }
    else

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 117  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        {
            itemImage.gameObject.SetActive(false);
        }

    }
}

```

#### Файл "item.cs"

```

using UnityEngine;

public enum ItemType
{
    None,
    Craftable,
}

public enum Quality
{
    None,
    common
}

/// <summary>
/// Базовый класс предмета
/// </summary>
[System.Serializable]
public class Item
{
    /// <summary> id предмета </summary>
    [SerializeField] private int _id;
    /// <summary> id предмета </summary>
    public int id
    { get { return _id; } }

    /// <summary> название предмета </summary>
    [SerializeField] private string _itemName;
    /// <summary> название предмета </summary>

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 118  |

```

public string itemName
{ get { return _itemName; } }

/// <summary> //описание предмета </summary>
[SerializeField] private string _description;
/// <summary> //описание предмета </summary>
public string description
{ get { return _description; } }

/// <summary> //иконка предмета </summary>
[SerializeField] private Sprite _itemIcon;
/// <summary> //иконка предмета </summary>
public Sprite itemIcon
{ get { return _itemIcon; } }

/// <summary> Тип предмета </summary>
[SerializeField] private ItemType _itemType;
/// <summary> Тип предмета </summary>
public ItemType itemType
{ get { return _itemType; } }

/// <summary> Базовая цена предмета </summary>
[SerializeField] private int _coast;
/// <summary> Базовая цена предмета </summary>
public int coast
{ get { return _coast; } }

/// <summary> Качество предмета </summary>
[SerializeField] private Quality _quality;
/// <summary> Качество предмета </summary>
public Quality quality
{ get { return _quality; } set { _quality = value; } }

/// <summary> Максимальное количество в стаче </summary>
[SerializeField] private int _maxStackCount;
/// <summary> Максимальное количество в стаче </summary>

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 119  |

```

public int maxStackCount
{ get { return _maxStackCount; }}

//Конструктор \ / \ / \ /

public Item()
{
    _id = -1;
    _itemName = "ItemName";
    _description = "ItemDescription";
    _itemType = ItemType.None;
    _coast = 1;
    _quality = Quality.None;
    _maxStackCount = 1;
}

    public Item(int id, string name, string description,
Sprite itemIcon, ItemType itemType, int coast, Quality quality,
int maxStackCount)
    {
        _id = id;
        _itemName = name;
        _description = description;
        _itemIcon = itemIcon;
        _itemType = itemType;
        _coast = coast;
        _quality = quality;
        _maxStackCount = maxStackCount;
    }

public void CopyItem(int id)
{
    if (ItemBase.singleton.FullItemList.ContainsKey(id))
    {
        _id = ItemBase.singleton.FullItemList[id].id;
    }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 120  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

        _itemName =
ItemBase.singleton.FullItemList[id].itemName;
        _description =
ItemBase.singleton.FullItemList[id].description;
        _itemIcon =
ItemBase.singleton.FullItemList[id].itemIcon;
        _coast =
ItemBase.singleton.FullItemList[id].coast;
        _quality =
ItemBase.singleton.FullItemList[id].quality;
        _maxStackCount =
ItemBase.singleton.FullItemList[id].maxStackCount;
    }
    else
    {
        _id = id;
        _itemName = "ItemName";
        _description = "ItemDescription";
        _itemType = ItemType.None;
        _coast = 1;
        _quality = Quality.None;
        _maxStackCount = 1;
    }
}

public static bool operator ==(Item a, Item b)
{
    //if (a == null || b == null) return false;
    if (a._id == b._id)
    {
        if (a._quality == b._quality)
        {
            return true;
        }
    }
    return false;
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 121  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

    }

    public static bool operator !=(Item a, Item b)
    {
        if (a._id != b._id || a._quality != b._quality)
return true;
        else return false;
    }

    public override bool Equals(object obj)
    {
        return base.Equals(obj);
    }

    public override int GetHashCode()
    {
        return base.GetHashCode();
    }

    public virtual void Use()
    {
    }
}

```

#### Файл "itemPrefab.cs"

```

using UnityEngine;

[CreateAssetMenu(fileName = "BaseItem", menuName =
"Game/Items/BaseItem")]
public class BaseItemPrefab : ScriptableObject
{
    /// <summary> id предмета </summary>
    [SerializeField] private int _id;
    /// <summary> id предмета </summary>
    public int id
    { get { return _id; } }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 122  |

```

/// <summary> название предмета </summary>
[SerializeField] private string _itemName;
/// <summary> название предмета </summary>
public string itemName
{ get { return _itemName; } }

/// <summary> //описание предмета </summary>
[SerializeField] private string _description;
/// <summary> //описание предмета </summary>
public string description
{ get { return _description; } }

/// <summary> //иконка предмета </summary>
[SerializeField] private Sprite _itemIcon;
/// <summary> //иконка предмета </summary>
public Sprite itemIcon
{ get { return _itemIcon; } }

/// <summary> Тип предмета </summary>
[SerializeField] private ItemType _itemType;
/// <summary> Тип предмета </summary>
public ItemType itemType
{ get { return _itemType; } }

/// <summary> Базовая цена предмета </summary>
[SerializeField] private int _coast;
/// <summary> Базовая цена предмета </summary>
public int coast
{ get { return _coast; } }

/// <summary> Качество предмета </summary>
[SerializeField] private Quality _quality;
/// <summary> Качество предмета </summary>
public Quality quality
{ get { return _quality; } set { _quality = value; } }

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 123  |

```

    /// <summary> Максимальное количество в стаче </summary>
    [SerializeField] private int _maxStackCount;
    /// <summary> Максимальное количество в стаче </summary>
    public int maxStackCount
    { get { return _maxStackCount; } }
}
using UnityEngine;

[CreateAssetMenu(fileName = "BaseItem", menuName =
"Game/Items/BaseItem")]
public class BaseItemPrefab : ScriptableObject
{
    /// <summary> id предмета </summary>
    [SerializeField] private int _id;
    /// <summary> id предмета </summary>
    public int id
    { get { return _id; } }

    /// <summary> название предмета </summary>
    [SerializeField] private string _itemName;
    /// <summary> название предмета </summary>
    public string itemName
    { get { return _itemName; } }

    /// <summary> //описание предмета </summary>
    [SerializeField] private string _description;
    /// <summary> //описание предмета </summary>
    public string description
    { get { return _description; } }

    /// <summary> //иконка предмета </summary>
    [SerializeField] private Sprite _itemIcon;
    /// <summary> //иконка предмета </summary>
    public Sprite itemIcon
    { get { return _itemIcon; } }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 124  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

    /// <summary> Тип предмета </summary>
    [SerializeField] private ItemType _itemType;
    /// <summary> Тип предмета </summary>
    public ItemType itemType
    { get { return _itemType; } }

    /// <summary> Базовая цена предмета </summary>
    [SerializeField] private int _coast;
    /// <summary> Базовая цена предмета </summary>
    public int coast
    { get { return _coast; } }

    /// <summary> Качество предмета </summary>
    [SerializeField] private Quality _quality;
    /// <summary> Качество предмета </summary>
    public Quality quality
    { get { return _quality; } set { _quality = value; } }

    /// <summary> Максимальное количество в стаче </summary>
    [SerializeField] private int _maxStackCount;
    /// <summary> Максимальное количество в стаче </summary>
    public int maxStackCount
    { get { return _maxStackCount; } }
}

```

#### Файл "MouseCell.cs"

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MouseCell : MonoBehaviour
{
    private InventoryCell _pickedItem;
    private int _count;
    private Transform _gameObjectTransform;
}

```

```

private void Start()
{
    _pickedItem = GetComponent<InventoryCell>();
    _gameObjectTransform = transform;
    _count = 0;
}

public void Pick(InventoryCell PickedCell)
{
}

private void Update()
{
    if (_pickedItem.item.id != -1)
    {
        Vector3 mousePosition = Input.mousePosition;
        Vector3 worldPosition =
Camera.main.ScreenToWorldPoint(mousePosition);
        //worldPosition.y = worldPosition.y - 0.3f;
        //worldPosition.x = worldPosition.x + 0.3f;
        worldPosition.z = -5;
        _gameObjectTransform.position = worldPosition;
    }

    if (Input.GetMouseButtonDown(0))
    {
        Vector2 mousePosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
        RaycastHit2D hit =
Physics2D.Raycast(mousePosition, Vector2.zero);
        if (hit.collider != null)
        {
            // Получаем компонент SpriteRenderer
            InventoryCell IC;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 126  |

```

        if
(hit.collider.TryGetComponent<InventoryCell>(out IC))
        {

            if (_pickedItem.item.id == -1 &&
IC.item.id != -1)
            {
                _pickedItem.addItemCMD(IC.item.id,
IC.itemCount, IC.item.quality);
                IC.DeleteItem();
            }
            else
            {
                if (IC.item.id == -1)
                {

IC.addItemCMD(_pickedItem.item.id, _pickedItem.itemCount,
_pickedItem.item.quality);

                _pickedItem.DeleteItem();
                }
                if (_pickedItem.item.id ==
IC.item.id)
                {
                    if (_pickedItem.itemCount +
IC.itemCount <= IC.item.maxStackCount)
                    {

IC.addItemCMD(_pickedItem.item.id, _pickedItem.itemCount,
_pickedItem.item.quality);

                    _pickedItem.DeleteItem();
                    }
                    else
                    {
                        int delItems =
IC.item.maxStackCount - IC.itemCount;

```

```

IC.addItemCMD(_pickedItem.item.id, delItems,
_pickedItem.item.quality);

_pickedItem.DeleteItem(delItems);

    }

    }

    }

    }

    }

    }
}

```

#### Файл "RadioScript.cs"

```

using Mirror;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.UI;

public class RadioScript : NetworkBehaviour, UsableObject
{
    [SerializeField] private AudioSource Source;
    [SerializeField] private List<AudioClip> Clips;
    [SerializeField] private GameObject UI;
    [SerializeField] private Slider slider;
    [SerializeField] private StandartUI standartUI;

    [SyncVar(hook =
nameof(SyncVolume))] [SerializeField] [Range(0f, 1f)] private float
Volume;

    [SyncVar(hook = nameof(SyncChange))] [SerializeField]
private int CurrentSong;

    [SyncVar(hook = nameof(Switch))] [SerializeField] private
bool IsPlaying;

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 128  |

```

private void Awake()
{
    Boot.LoadWorldObjects += Load;
    CameraControllerScript.CameraLoaded += SetCamera;
    UI.SetActive(false);
}

private void OnDestroy()
{
    Boot.LoadWorldObjects -= Load;
    CameraControllerScript.CameraLoaded -= SetCamera;
}

public void Load()
{
    if(UI == null)
    {

Debug.LogWarning(gameObject.GetComponent<StandartUI>().pUIobject
== null);

        UI =
gameObject.GetComponent<StandartUI>().pUIobject;
    }
    SetCamera();
    Source.clip = Clips[CurrentSong];
    //NetworkServer.Spawn(gameObject);
}

public void SetCamera()
{
    if(Camera.main != null)
UI.gameObject.GetComponent<Canvas>().worldCamera = Camera.main;
}

//Включение/выключение

[Command(requiresAuthority = false)]

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 129  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

public void PlayStopCMD()
{
    IsPlaying = !IsPlaying;
    if (IsPlaying) Source.Play();
    else Source.Stop();
}

public void Switch(bool old, bool New)
{
    if (New)
    {
        Source.Play();
        SyncSourceCMD();
    }

    Camera.main.GetComponent<MusicTemp>().StopMusic();
    }
    else
    {
        Source.Stop();
    }

    Camera.main.GetComponent<MusicTemp>().StartMusic();
    }

}

[Command(requiresAuthority = false)]
public void SyncSourceCMD()
{
    SyncSourceRPC(Source.time);
}

[ClientRpc]
public void SyncSourceRPC(float Time)
{
    Source.time = Time;
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 130  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

```

    }

    //Смена музыки

    [Command(requiresAuthority = false)]
    public void ChangeButton(int New)
    {
        int NewMusic = CurrentSong + New;
        if (NewMusic < 0) NewMusic = Clips.Count-1;
        if (NewMusic > Clips.Count - 1) NewMusic = 0;
        CurrentSong = NewMusic;
        Source.clip = Clips[CurrentSong];
        if(IsPlaying) Source.Play();
    }

    public void SyncChange(int Old, int New)
    {
        Source.clip = Clips[CurrentSong];
        if (IsPlaying) Source.Play();
    }

    //Изменение громкости

    public void ChangeVolume()
    {
        ChangeVolumeCMD(slider.value);
    }

    [Command(requiresAuthority = false)]
    public void ChangeVolumeCMD(float NewVolume)
    {
        Volume = NewVolume;
    }

    public void SyncVolume(float Old, float New)
    {

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
| Изм | Лист | № докум. | Подпись | Дата |                        | 131  |

```

        slider.value = Volume;

gameObject.GetComponent<LinkToVolume>().ChangeVolume();
    }

    //Интерфейс
    public void Use()
    {
        standartUI.OpenUI();
    }

    public void CloseUI()
    {
        standartUI.CloseUI();
    }
}

```

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 132  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

## Приложение В

(Обязательное)

Бизнес-проект “Симулятор сельского хозяйства с элементами ролевой игры”

Презентация

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 133  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |

# «The Little story»

## Симулятор сельского хозяйства с элементами ролевой игры

Автор работы:  
студент гр. 21BBV1 Гришин Сергей Вадимович

Научный руководитель:  
кандидат э.н., доцент, Акифьев Илья Владимирович

Пенза, 2025



## Цель работы и объекты

Цель работы – разработка игры в жанре TopDown RPG в PixelART стиле на движке Unity.

Объекты разработки:

- игровое приложение с реализацией основных механик, характерных для данного жанра;
- Мультиплеерная составляющая;
- визуальное составляющая (пользовательский интерфейс).

2

## Актуальность

Последние 3 года наблюдается рост спроса на игры с простым и медитативным игровым процессом. Так же растёт спрос на игры с открытым миром.

При этом игр, которые сочетают в себе две эти особенности выходит всё меньше. Согласно статистике Steam за последний год вышло всего 2 таких проекта.

3

## Анализ аналогов

Самые популярные проекты вышли более 5 лет назад, из чего выходят следующие проблемы:

- больше не выходят обновления
- сломанный баланс
- баги при запуске на современных устройствах
- больше нет возможности приобрести некоторые из них



4

|     |      |          |         |      |
|-----|------|----------|---------|------|
|     |      |          |         |      |
| Изм | Лист | № докум. | Подпись | Дата |

## Сравнение аналогов

| название                   | плюсы                                                                                                                    | минусы                                                                                                                                        |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Stardew Valley</b>      | <ul style="list-style-type: none"> <li>- Иногда выходят обновления.</li> <li>- Большое разнообразие контента.</li> </ul> | <ul style="list-style-type: none"> <li>- Игра на скорость.</li> <li>- Сломанный баланс.</li> </ul>                                            |
| <b>My time at Sandrock</b> | <ul style="list-style-type: none"> <li>- большой открытый мир.</li> <li>- система социальных взаимодействий.</li> </ul>  | <ul style="list-style-type: none"> <li>- затянутый игровой процесс.</li> <li>- много повторяющихся локаций во второй половине игры</li> </ul> |
| <b>Animal Crossing</b>     | <ul style="list-style-type: none"> <li>- хороший баланс механик.</li> <li>- Бесконечная генерация мира.</li> </ul>       | <ul style="list-style-type: none"> <li>- малое разнообразие механик.</li> <li>- привязка к интернету.</li> </ul>                              |

5

## Описание продукта

**Название: The little story**

**Жанр: TopDown RPG**

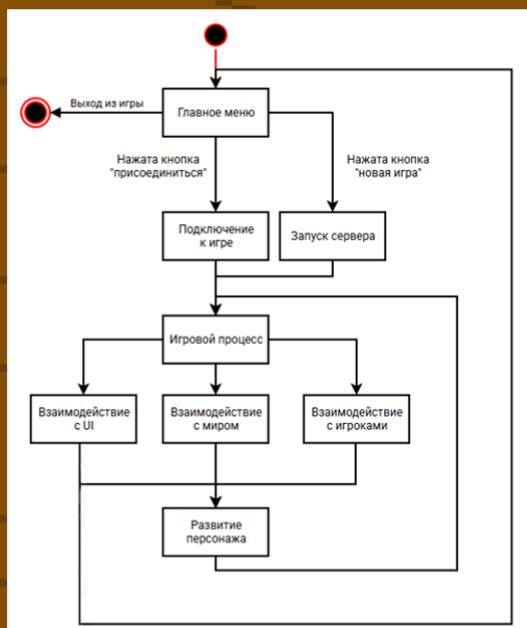
**Платформа: Windows**

**Движок: Unity**

**Модель монетизации: подписка**

**Целевая аудитория: казуальные игроки в возрасте от 14 до 30**

6



## Игровой процесс Основные принципы:

- Цикличность процесса
- Улучшение после каждого цикла
- кооператив

7

## Используемые технологии

**Mirror для реализации мультиплеера без привязки к стороннему серверу**



**DoTween для оптимизации передвижения UI элементов и анимаций**



8

|     |      |          |         |      |
|-----|------|----------|---------|------|
|     |      |          |         |      |
| Изм | Лист | № докум. | Подпись | Дата |

ПГУ 1.090301.05 001 ПЗ

лист

137

## Mirror

**Имеет множество заготовок для работы клиент серверного приложения**

**Наличие атрибутов SyncVar, Command и TargetRCP позволят удобно реализовывать сетевой код, не ломая архитектуру игры.**

```
[SyncVar(hook = nameof(Switch))] private bool IsPlaying;

//Включение/выключение

[Command(requiresAuthority = false)]
Ссылка: 0
public void PlayStopCMD()
{
    IsPlaying = !IsPlaying;
    if (IsPlaying) Source.Play();
    else Source.Stop();
}

[ClientRpc]
Ссылка: 0
public void SwitchRpc()
{
    Switch();
}
```

9

## DoTween

**Позволяет создавать плавные анимации и перемещения объектов через код, не используя аниматор Unity.**

**Это позволяет улучшить визуальную часть, производительность и скорость разработки игры.**

```
Ссылка: 1
public IEnumerator SlideShow()
{
    int ImageIndex = 0;
    while (true)
    {
        BackgroundImage.sprite = Backgrounds[ImageIndex];

        ImageIndex++;
        if (ImageIndex >= Backgrounds.Length) ImageIndex = 0;

        yield return new WaitForSeconds(DurationTime);

        BackgroundBuffer.sprite = BackgroundImage.sprite;
        BackgroundBuffer.color = new Color(1f, 1f, 1f, 1f);
        BackgroundImage.sprite = Backgrounds[ImageIndex];
        float Alpha = 1f;
        while(BackgroundBuffer.color.a > 0)
        {
            Alpha = Alpha - ChangeStep;
            BackgroundBuffer.color = new Color(1f, 1f, 1f, Alpha);
            yield return new WaitForSeconds(ChangeTime * ChangeStep);
        }
    }
}
```

```
VolumePanelTransform.DOAnchorPosX(Xsize, 0.4f);
SettingsButtonsTransform.DOAnchorPosX(-Xsize, 0.25f);
```

10

|     |      |          |         |      |
|-----|------|----------|---------|------|
|     |      |          |         |      |
| Изм | Лист | № докум. | Подпись | Дата |

ПГУ 1.090301.05 001 ПЗ

лист

138

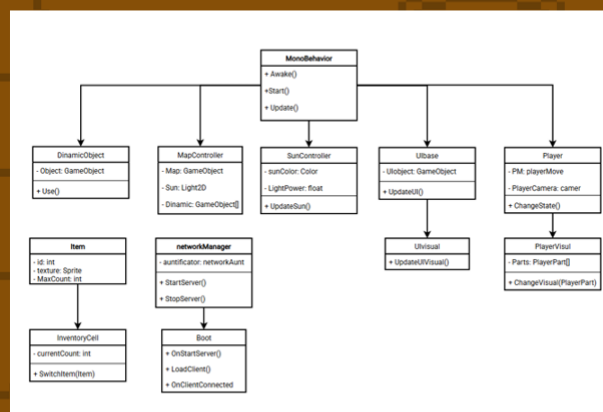
# Архитектура

Симуляция живого мира:

- Минимум общих контроллеров
- Возможность управления игровым миром из любой части кода
- Общий интерфейс доступа ко всем объектам

Масштабируемость:

- Разделение внутренней и визуальной части объекта
- Отдельные объекты для хранения настроек игры, мира и персонажа
- передача классов через string
- Инициализация главных контроллеров через boot



11

# Внешний вид игры



12

|     |      |          |         |      |
|-----|------|----------|---------|------|
|     |      |          |         |      |
| Изм | Лист | № докум. | Подпись | Дата |

ПГУ 1.090301.05 001 ПЗ

лист

139

## Экономическая эффективность

### Расходы:

- отрисовка графики: 30 000
  - написание музыки: 50 000
  - реклама в тематических группах: 20 000
  - создание сайта проекта: 20 000
  - ежемесячная оплата хостинга: 1000
- Итого 120 000 рублей**

При стартовой аудитории в 1000 человек и стартовой цене подписки 50 выходит 50 000 в месяц.

С ростом аудитории начинается пропорциональный рост дохода.

**13**

# Спасибо за внимание

|     |      |          |         |      |
|-----|------|----------|---------|------|
|     |      |          |         |      |
| Изм | Лист | № докум. | Подпись | Дата |

ПГУ 1.090301.05 001 ПЗ

лист

140

|     |      |          |         |      |                        |      |
|-----|------|----------|---------|------|------------------------|------|
|     |      |          |         |      | ПГУ 1.090301.05 001 ПЗ | лист |
|     |      |          |         |      |                        | 141  |
| Изм | Лист | № докум. | Подпись | Дата |                        |      |