Министерство образования Калининградской области

государственное автономное учреждение

Калининградской области

профессиональная образовательная организация

«Колледж предпринимательства»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ (ДИПЛОМНАЯ) РАБОТА**

**Тема: Разработка игры. Взаимодействие с игровой средой.**

|  |  |
| --- | --- |
| Выпускная квалификационная  (дипломная) работа  допущена к защите  Заместитель директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_ Бурыкина Ю.И.  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | Выполнил:  обучающийся группы ИСП-18.2  Специальность 09.02.07 Информационные системы и программирование  \_\_\_\_\_\_\_\_\_\_\_ Лунина А.В.  Руководитель:  \_\_\_\_\_\_\_\_\_\_\_ Григорьева К.М. |

Калининград

2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc14362)

[1. ОБЩИЕ СВЕДЕНИЯ 5](#_Toc31013)

[1.1. Классификации компьютерных игр. 5](#_Toc24579)

[1.2. Общие сведения о языке C# 8](#_Toc21658)

[1.3. Инструментальные средства 12](#_Toc17844)

[1.4. Среда разработки Unity 14](#_Toc26229)

[1.5. Обоснование актуальности темы игры 17](#_Toc17909)

[2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 19](#_Toc18521)

[2.1. Концептуальная модель 19](#_Toc27632)

[2.2. Отрисовка дизайна 21](#_Toc6292)

[2.3. Создание макета в Unity 25](#_Toc17758)

[2.4. Написание скриптов 29](#_Toc31765)

[2.4.1. Реализация игрового персонажа Player 29](#_Toc24864)

[2.4.2. Реализация фона и декораций 29](#_Toc5406)

[2.4.3. Реализация генерации и уничтожения объектов 29](#_Toc5044)

[2.5. Создание пользовательского интерфейса 29](#_Toc15477)

[3. ТЕСТИРОВАНИЕ 30](#_Toc14363)

[3.1. Функциональное тестирование 30](#_Toc31004)

[3.2. Юзабилити тестирование 30](#_Toc6448)

[3.3. Исправление ошибок 30](#_Toc21214)

[3.4. Экономические расчеты 30](#_Toc20100)

[ЗАКЛЮЧЕНИЕ 30](#_Toc15923)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 30](#_Toc28978)

[ПРИЛОЖЕНИЯ 32](#_Toc5188)

# **Введение**

На сегодняшний день, очень мало людей не знают, что такое вычислительные устройства. И сейчас компьютерные средства можно встретить в любой сфере жизни населения. Изначально такие машины были созданы для вычисления конкретных задач, но сегодня они могут выполнять практически любые желания. К одним из таких относится - игровая индустрия.

Для разработчика - программиста компьютерные игры имеют множество задач, с которыми ему предстоит справится. В то время, как для пользователя, компьютерная игра должна удовлетворять собственным потребностям. Человек, захотевший отдохнуть остановит свой выбор на незамысловатой игре, от которой нужен лишь отдых. Есть и другие люди, которые хотят поиграть, и чему-то научиться, проверить себя и свою стойкость. А также есть дети, их игры должны быть развивающими.

В данной дипломной работе была поставлена цель – разработать компьютерную игру.

Для достижения данной цели необходимо выполнить следующие задачи:

* Изучить и проанализировать существующие жанры игр и выбрать подходящий.
* Выбрать необходимый язык программирования и инструментальные средства.
* Нарисовать и оформить визуальную часть игры.
* Написать и протестировать код разработки игры.

Компьютерные игры могут быть доступны на различных платформах, и созданы для различных операционных систем. Так же есть и кроссплатформенные проекты, они способны работать на нескольких платформах. Данная игра должна быть создана для компьютера, на операционной системе Windows 10.

# **ОБЩИЕ СВЕДЕНИЯ**

* 1. **Классификации компьютерных игр.**

**Авторские классификации компьютерных игр**.

Классификация компьютерных игр является междисциплинарной категорией, которая изучается в рамках культурологии, философии, эстетики и на их пересечении. Активно применяется подход, основанный на системе правил, по аналогии с классификацией фильмов.

Так, в диссертационной работе Тимура Кутлалиева «Жанровая типология компьютерных игр: проблема систематизации художественных средств» за 2014 год был выделен ряд источников, которые в достаточной степени хорошо показывают развитие подходов к классификации компьютерных игр начиная с 1980-х годов. Но также эти классификации нельзя назвать окончательными, ведь индустрия развивается и каждый год появляется все больше подвидов и жанров игр.

**Категории игр по Кроуфорду(1984)**

Наиболее ранняя работа по классификации «электронных игр» была проведена геймдизайнером Крисом Кроуфордом в 1984 году. Результатом проведенной работы стало деление игр в два этапа. На первом этапе игры делились на две категории:

Таблица 1

Классификации электронных игр Криса Кроуфорда

|  |  |  |
| --- | --- | --- |
| **Категория** | **Синонимы** | **Описание** |
| **1** | **2** | **3** |
| Skill-and-action game (Игры на ловкость и реакцию) | S&A games, Экшен. | Акцентирует внимание на точности и скорости движений. |
| Strategy (Стратегия) | Cognitive games, стратегии. | Акцентирует внимание на когнитивной активности. |

Далее игры из обеих категорий классифицировались на 6 классов, по принципу схожести игровой механики и наличия «исторической связи».

Таблица 2

Классы игр Strategy по Кроуфорду

|  |  |  |
| --- | --- | --- |
| **Категория** | **Класс** | **Описание** |
| **1** | **2** | **3** |
| Strategy (Стратегия) | Adventure (Приключение) | Акцент на прохождении сложных заданий, накоплении инструментов и достижении цели. |
| D&D games (Игры про подземелье и драконов) | Фантазийная ролевая игра |
| Educational and children’s games (Обучающие игры и игры для детей ) | Фокус на обучающей составляющей |
| Game of Chance (Азартные игры ) | Игры с вероятностью, азартные игры |
| Inretpersonal games (Межличностные игры) | Фокус на отношениях между несколькими игроками |
| War games (Военные игры ) | Фокус на активном противостоянии с противником |

Стратегическая игра — жанр компьютерных игр, в которых игроку для победы необходимо применять стратегическое мышление. В популярных играх такого жанра игроку предлагается играть не за конкретного персонажа, а за их условные массы, к примеру, руководить строительством города или командовать целыми армиями в военных кампаниях.

Смысл таких игр заключается в управлении некоторым ресурсом, который необходимо преобразовать в преимущество над противником при помощи оперативного плана, разрабатываемого с учётом меняющейся обстановки. Обычными ресурсами в военных стратегиях являются войска (отдельные персонажи, подразделения или армии) и позиция, которые следует развивать и использовать для достижения преимущества и победы. В экономических стратегиях акцент ставится на развитие экономической инфраструктуры подконтрольной игроку стороны. Современные стратегические игры, как правило, соединяют в себе как военные, так и экономические признаки.

Таблица 3

Классы игр Skill-and-action game по Кроуфорду

|  |  |  |
| --- | --- | --- |
| **Категория** | **Класс** | **Описание** |
| **1** | **2** | **3** |
| Skill-and-action game (Игры на ловкость и реакцию) | Combat games (Боевые игры) | Игры с прямой конфронтацией на нанесение и избегание урона |
| Maze games (Игры в лабиринте) | Игры-лабиринты на выживание |
| Miscellaneous game (Смешанные игры) | Игры не вошедшие в предыдущие пять классов |
| Paddle games (Игры с веслом) | Игры на удержание мяча на игровом поле |
| Race games (Игры гонки) | Гонки на различных болидах |
| Sports games (Спортивные игры) | Моделирование классических видов спортивных игр |

В данный момент такая категория игр называется Экшн, что переводится как действия, и является жанром компьютерных игр, в котором делается упор на эксплуатацию физических возможностей игрока, в том числе координации глаз и рук и скорости реакции. Жанр представлен во множестве разновидностей от файтингов, шутеров и платформеров, которые считаются наиболее важными для жанра, до MOBA и некоторых стратегий в реальном времени, которые возможно отнести к жанру экшен.

К сожалению, данная классификация актуальна только для игр, созданных в 70—80-е годы и не вполне применима к современному игровому разнообразию.

Далее предложены наиболее актуальные авторские классификации игр.

**Традиционная классификация по жанрам игр**

Жанровое различие игр начало активно развиваться с середины 90-х годов. Основой для формирования классического свода жанров послужили серии игр. Эти серии были образованы в результате того, что после выхода в свет популярной «оригинальной игры» как официальные, так и сторонние разработчики продолжали эксплуатировать черты первой игры, создавая «клонов».

Таблица 4

Традиционно выделяемые жанры по Кутлалиеву

|  |  |  |
| --- | --- | --- |
| **Жанр** | **Синонимы** | **Описание** |
| **1** | **2** | **3** |
| Действие | Action, Эншн, Экшен | Требующие хорошей моторики, глазомера |
| Симулятор | Simulation, Симуляция | Имитирующее определённые направления деятельности |
| Стратегия | Strategy | С упором на управление ресурсами |
| Ролевая игра | Role-playing Games | С развитой системой изменения персонажей |
| Приключения | Adventure | Предполагающие преодоление различных препятствии на пути персонажа |
| Головоломка | Puzzle | Заключающиеся в решение логических задач, построенных на общем наборе правил |

Количество компьютерных игр просто огромно, и часто они классифицируются по характеристикам или задачам, а не по типу геймплея. Поэтому категории игр, или жанры, могут подразделяться на поджанры, а одна игра вполне может принадлежать к нескольким жанрам.

Выбор видеоигр поистине безграничен, и существует не только множество самых разнообразных жанров, но и огромное количество успешных поджанров в рамках каждого из них.

После анализа большинства существующих жанров для разработки игры был выбран жанр RPG (Компьютерная ролевая игра), а именно Dungeon crawler(Зачистка подземелий ).

* 1. **Общие сведения о языке C#**

C# (произносится как "си шарп") — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C# - это объектно- и компонентно-ориентированный язык программирования. Он предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык программирования обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

Язык C#, разработанный компанией Майкрософт, один из самых популярных современных языков программирования. Он востребован на рынке разработки в различных странах, C# применяют при работе с программами для ПК, создании сложных веб-сервисов или мобильных приложений. Появившийся как язык для собственных нужд платформы Microsoft .NET, постепенно этот язык стал очень популярным. Разработка языка началась в 1998 году, а первая версия увидела свет в 2001. Группой разработчиков руководил известный в профессиональных кругах специалист Андерс Хейлсберг. Новые версии C# выходят сравнительно часто, а текущие доработки, исправление ошибок и расширение библиотек ведётся практически на постоянной основе. Долгое время платформа .NET поставлялась с закрытым ядром, что создавало определённые сложности в разработке и снижало популярность C# в профессиональной среде. Но в ноябре 2014 Майкрософт радикально изменила подход и стала выдавать бесплатные лицензии для Visual Studio уже с открытым исходным кодом для всех наборов инструментов.

Далее предоставлены несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения.

* Сборка мусора - автоматически освобождает память, занятую недоступными неиспользуемыми объектами;
* Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты;
* Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них;
* Лямбда-выражения поддерживают приемы функционального программирования;
* Синтаксис LINQ создает общий шаблон для работы с данными из любого источника;
* Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем;
* В C# действует единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object;
* Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом;
* C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений;
* C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке;
* C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность;
* C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

Для работы приложений на C# необходимо установить и настроить платформу NET Framework. Она поставляется полностью бесплатно, применяется крайне широко, а потому проблем с пользовательскими устройствами обычно не возникает. Платформа встроена в установочный пакет Windows, при необходимости ее также можно скачать и установить отдельно. Существуют версии для Линукс и MAC.

В рамках платформы к обработке исполняемого кода подключается среда CLR – единый объединенный набор библиотек и классов, который был разработан Майкрософт и является реализацией мирового стандарта Common Language Infrastructure (CLI).

После работы компилятора текст программы переводится в промежуточный язык IL, который разбирает CLI. Работает это так. IL и все необходимые ресурсы, включая строки и рисунки формата BMP, сохраняются на жесткий диск в виде исполняемого файла dll или exe. Из таких файлов с промежуточным кодом формируется сборка приложения, которая включает в себя описание с полной информацией обо всех важных параметрах работы.

Непосредственно при выполнении программы CLR обращается к сборке и производит действия в зависимости от полученных сведений. Если код написан правильно и проходит проверку безопасности системы, производится компиляция из IL в инструкции в машинные команды. В результате код C# считается управляемым, т.е. он компилируется в двоичный вид на пользовательском устройстве с учетом особенностей установленной системы.

**Классы и объекты в C#.**

Классы являются основным типом в языке C#. Класс представляет собой структуру данных, которая объединяет в себе значения (поля) и действия (методы и другие функции-члены). Класс предоставляет определение для экземпляров класса, которые также именуются объектами. Классы поддерживают механизмы наследования и полиморфизма, которые позволяют создавать производные классы, расширяющие и уточняющие определения базовых классов.

Новые классы создаются с помощью объявлений классов. Объявление класса начинается с заголовка. Заголовок указывает:

* Атрибуты и модификаторы класса;
* Имя класса;
* Базовый класс (при наследовании от базового класса);
* Интерфейсы, реализуемые классом.

За заголовком между разделителями {} следует тело класса, в котором последовательно объявляются все члены класса.

Экземпляры классов создаются с помощью оператора new, который выделяет память для нового экземпляра, вызывает конструктор для инициализации этого экземпляра и возвращает ссылку на экземпляр.

Занимаемая объектом память автоматически освобождается, когда объект становится недоступен. В C# нет ни необходимости, ни возможности освобождать память объектов явным образом.

* 1. **Инструментальные средства**

**IDE Visual Studio.**

Интегрированная среда разработки (IDE) — это многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения. Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Среда Visual Studio доступна для Windows и Mac. Функции Visual Studio для Mac во многом аналогичны возможностям Visual Studio для Windows и оптимизированы для разработки кроссплатформенных и мобильных приложений.

Visual Studio признана лучшей IDE для C#. Дело в том, что оба продукта принадлежат корпорации Microsoft. Поэтому они идеально подходят для работы друг с другом.

В число преимуществ Visual Studio входит следующее:

* Среда содержит множество инструментов, которые очень хорошо работают на C#;
* Наличие бесплатной версии — Community Edition;
* Community содержит все что нужно для независимого разработчика;
* Самое эффективное ПО для разработки на любой платформе, включая .Net и C#;
* Возможность хранения данных в облаке.

У продукта есть и некоторые недостатки:

* Требовательность к ресурсам;
* После перехода на платную версии, могут слететь настройки и корпоративный сервер;
* Сложность при самостоятельном освоении.

**Популярные средства повышения производительности.**

Вот несколько популярных возможностей Visual Studio, которые повышают производительность при разработке программного обеспечения:

* Волнистые линии и быстрые действия;

Волнистые линии обозначают ошибки или потенциальные проблемы кода прямо во время ввода. Эти визуальные подсказки помогают немедленно устранить проблемы, не дожидаясь появления ошибок во время сборки или выполнения. Если навести указатель мыши на волнистую линию, на экран будут выведены дополнительные сведения об ошибке.

* Очистка кода;

Возможность одним нажатием кнопки отформатировать код и применить к нему исправления, предложенные параметрами стиля кода, соглашениями в файле. editorconfig и (или) анализаторами Roslyn. Очистка кода, которая сейчас доступна только для кода C#, помогает устранять проблемы в коде перед переходом к его проверке.

* Рефакторинг;

Рефакторинг включает в себя такие операции, как интеллектуальное переименование переменных, извлечение одной или нескольких строк кода в новый метод и изменение порядка параметров методов.

* IntelliSense;

IntelliSense — это набор возможностей, отображающих сведения о коде непосредственно в редакторе и в некоторых случаях, автоматически создающих небольшие отрывки кода. По сути, это встроенная в редактор базовая документация, которая избавляет от необходимости искать информацию в других источниках.

* CodeLens.

CodeLens помогает находить ссылки на код, изменения кода, связанные с кодом ошибки, рабочие элементы, проверки кода и модульные тесты — не выходя из редактора.

* 1. **Среда разработки Unity**

Unity - межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие.

Первая версия Unity появилась в 2005 году, когда игровой движок был анонсирован на Worldwide Developers Conference. Изначально Unity предназначался исключительно для компьютеров Mac, а в августе вышло обновление, позволяющее работать под Windows. В следующих версиях постепенно добавлялись новые платформы и развёртывания: межплатформенный веб-плеер в 2006-м, iPhone в 2008-м, Android в 2010-м, и далее на игровых консолях Xbox и Playstation.

Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (Unity Web Player), а также с помощью реализации технологии WebGL. Ранее была экспериментальная поддержка реализации проектов в рамках модуля Adobe Flash Player, но позже команда разработчиков Unity приняла сложное решение по отказу от этого.

В декабре 2009 года Gamasutra назвал Unity одним из самых значительных участников на рынке игровых компаний.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями. Благодаря удобству использования, Unity делает создание игр максимально простым и комфортным, а мультиплатформенность движка позволяет игроделам охватить как можно большее количество игровых платформ и операционных систем.

По данным сайта Game Developer, в 2021 году 49,48% всех платных игр, вышедших в Steam, разработаны на Unity. А в сфере мобильных игр этот показатель уже давно превысил 50%.

Unity стал наиболее популярным игровым движком из-за целого ряда преимуществ:

* простота, благодаря которой движком могут овладеть даже новички;
* универсальность — Unity показывает себя одинаково хорошо при работе над самыми разными играми;
* гибкость — движок можно глубоко настраивать под каждый новый проект;
* огромная библиотека ассетов, с помощью которых можно ускорить и упростить разработку;
* наличие системы визуального скриптинга;
* множество источников для обучения и обмена опытом.

Unity повсеместно используется в игровой индустрии — этот движок применяют как для крупных консольных игр, так и для небольших мобильных проектов. Кроме того, на нём можно сделать игру и в одиночку. Благодаря этому у разработчика на Unity есть широкие возможности при выборе места работы — он одинаково легко может пойти и в инди-геймдев, и в сферу мобильных игр.

**Игровой процесс в 2D.**

Несмотря на то, что Unity славится своими 3D-возможностями, его также можно использовать для создания 2D-игр. Знакомые функции редактора по-прежнему доступны, но с полезными дополнениями для упрощения 2D-разработки.

Графические объекты в 2D известны как спрайты. Спрайты - это, по сути, просто стандартные текстуры, но существуют специальные методы для объединения текстур спрайтов и управления ими для повышения эффективности и удобства во время разработки. Unity предоставляет встроенный редактор спрайтов, чтобы вы могли извлекать спрайтовую графику из увеличенного изображения. Это позволяет редактировать несколько изображений в одной текстуре в редакторе спрайтов. Это позволяет использовать руки, ноги и тело персонажа как отдельные элементы в одном изображении. Спрайты визуализируются с помощью средства визуализации спрайтов, а не в средстве визуализации сетки, используемом с 3D-объектами. Unity имеет отдельный физический движок для обработки 2D-физики, чтобы использовать оптимизации, доступные только в 2D. Компоненты Большинство компонентов 2D-физики являются просто “уплощенными” версиями 3D-эквивалентов (например, Box Collider 2D представляет собой квадрат, в то время как Box Collider представляет собой куб), но есть несколько исключений.

* 1. **Обоснование актуальности темы игры**

Геймдев — обширная тема, с большими видами жанров, и поджанров. Нельзя обесценивать работу разработчиков, ведь это очень кропотливый труд. Для гейм - дизайнера важно удержать игрока. Нужно придумать сюжет, правила, красивое оформление дизайна, и приятное звуковое сопровождение. При этом сюжетная линия должна быть интересной, правила не должны быть очень трудными, а дизайн не должен отпугивать игрока.

Причиной выбора жанра Dungeon crawler является то, что жанр ставит во главу угла развитие персонажа, превращая повышение характеристик и приобретение новой, более ценной экипировки в самоцель. Сюжеты и декорации таких игр очень просты и служат не более чем предлогом для истребления множества противников и поиска сокровищ. Так же этот жанр представляет собой лабиринтообразное подземелье (пещера, древний храм, катакомбы и т. п.), по которому странствует герой, уничтожая населяющих подземелье монстров и забирая себе разного рода сокровища.

В игре нет слишком сложного сюжета, у нее очень простые правила, приятная глазу рисовка, и не навязчивое звуковое сопровождение. Игра не требует усидчивости, и крайне понятна. Также в ней имеются элементы шутера поэтому игроку не придется скучать. Игрок всегда может начать игру сначала, чтобы увеличить свои очки.

Актуальность данной игры в том, что она поможет человеку расслабиться, насладится красивыми визуальными качествами, и приятным звуковым сопровождением. Бросить вызов самому себе и почувствовать себя победителем.

# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

## **Концептуальная модель**

Игра«Афина»представляет собой шутер с элементами рпг, в котором игроку придётся стрелять во врагов, улучшать своего персонажа, и зарабатывать очки, которые помогут ему продвинуться среди таблицы лидеров.

Ниже представлен приоритет функциональных особенностей игры.

**Высокий приоритет**

* Персонаж передвигается по клавишам – стрелкам. По горизонтали и вертикали;
* Персонаж наносит урон врагам из огнестрельного оружия. Атака производится при помощи мыши. Направление оружия следует за курсором мыши;
* Персонаж имеет ограниченные «очки жизни», при потере которых игра закончится;
* При окончании игры появляется таблица лидеров. Она отображает время прохождения и «очки за убитых врагов» (позже «монеты»);
* Атаки персонажа по врагам наносят урон, который увеличивается в зависимости от уровня персонажа;
* При смерти врага, персонажу прибавляются «очки за убитых врагов». Разные типы врагов приносят разное количество «очков»;
* Враги имеют разную атаку, скорость передвижения, и «очки здоровья».
* После смерти врага может появится «зелье лечения», которое восстанавливает 20% от максимальных «очков здоровья». Если «очки здоровья» полные, то «зелье» добавит дополнительные «очки здоровья»;

**Низкий приоритет**

1. Персонаж имеет возможность усиливать свои характеристики. При достижении определённого количества «очков за убитых врагов» уровень персонажа повышается, что позволят ему усилить свои характеристики;

* «Сила» - Увеличивает очки здоровья персонажа, и урон от атаки с оружия;
* «Ловкость» - Увеличивает скорость атаки персонажа, и скорость передвижения;
* «Защита» - Уменьшает входящий урон от врагов, и зелье лечения восстанавливает больше очков здоровья.

1. При убийстве врага могут появится редкие бонусы.

* «Бонус неуязвимости» - персонаж не может получать никакой урон в течении пяти секунд;
* «Бонус смерти» - все враги в поле зрения персонажа умирают. Действует одну секунду;
* «Бонус скорости» - персонаж увеличивает свою скорость передвижения и атаки до максимума в течении пяти секунд.

## **Отрисовка дизайна**

В данном проекте, по отношению к прорисовке дизайна, необходимо реализовать :

1. Пользовательский интерфейс;
2. Спрайты персонажей;
3. Тайлы окружения.

**Пользовательский интерфейс**

Пользовательский интерфейс необходимо реализовать понятным для игрока. На сцене Меню пользователь может увидеть:

* Поле для ввода имени;
* Кнопку сохранения имени;
* Кнопку начала игры;
* Кнопку продолжения игры;
* Кнопку выбора уровня;
* Кнопку выхода из игры.

Кнопка сохранить имя является неактивной пока пользователь не введёт имя., а кнопка начать игру неактивна пока пользователь не нажмёт кнопку сохранения имени. Кнопки выйти, выбор уроня являются активными, кнопка продолжить активна, если игрок уже вводил свое имя Основная сцена меню предоставлена на рисунке 1.

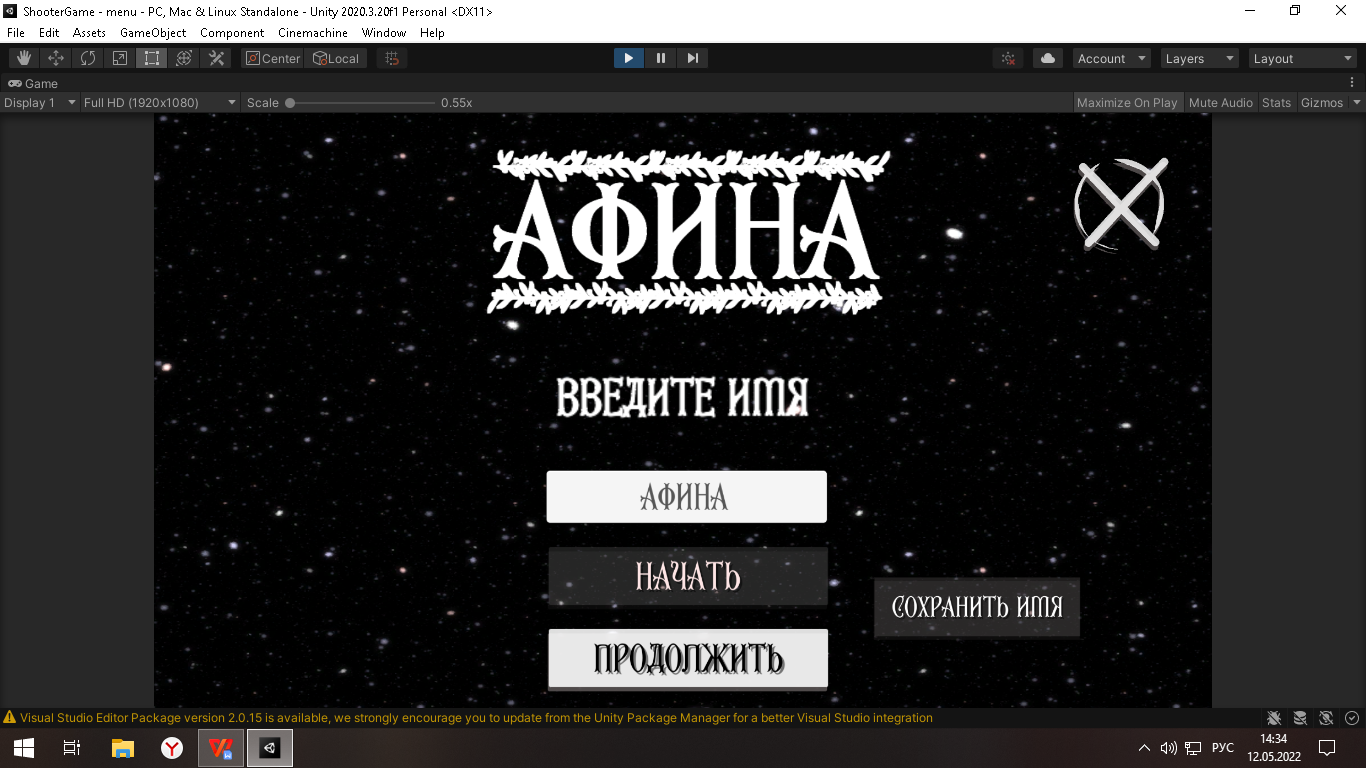


Рисунок 1. Сцена меню

После первого начала игры, пользователя необходимо ознакомить с правилами игры. Все правила игры предоставлены на рисунках 2, 3, 4, 5, 6.

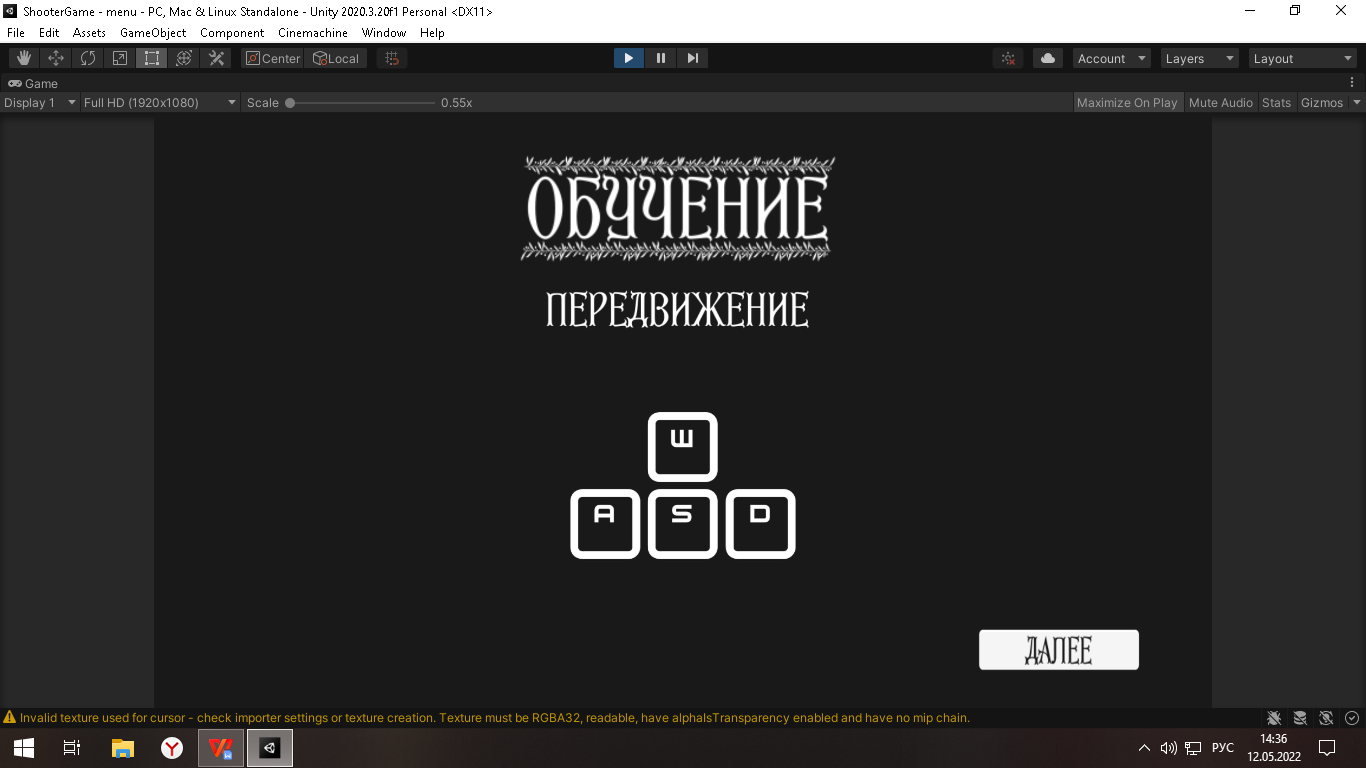


Рисунок 2. Обучение передвижению

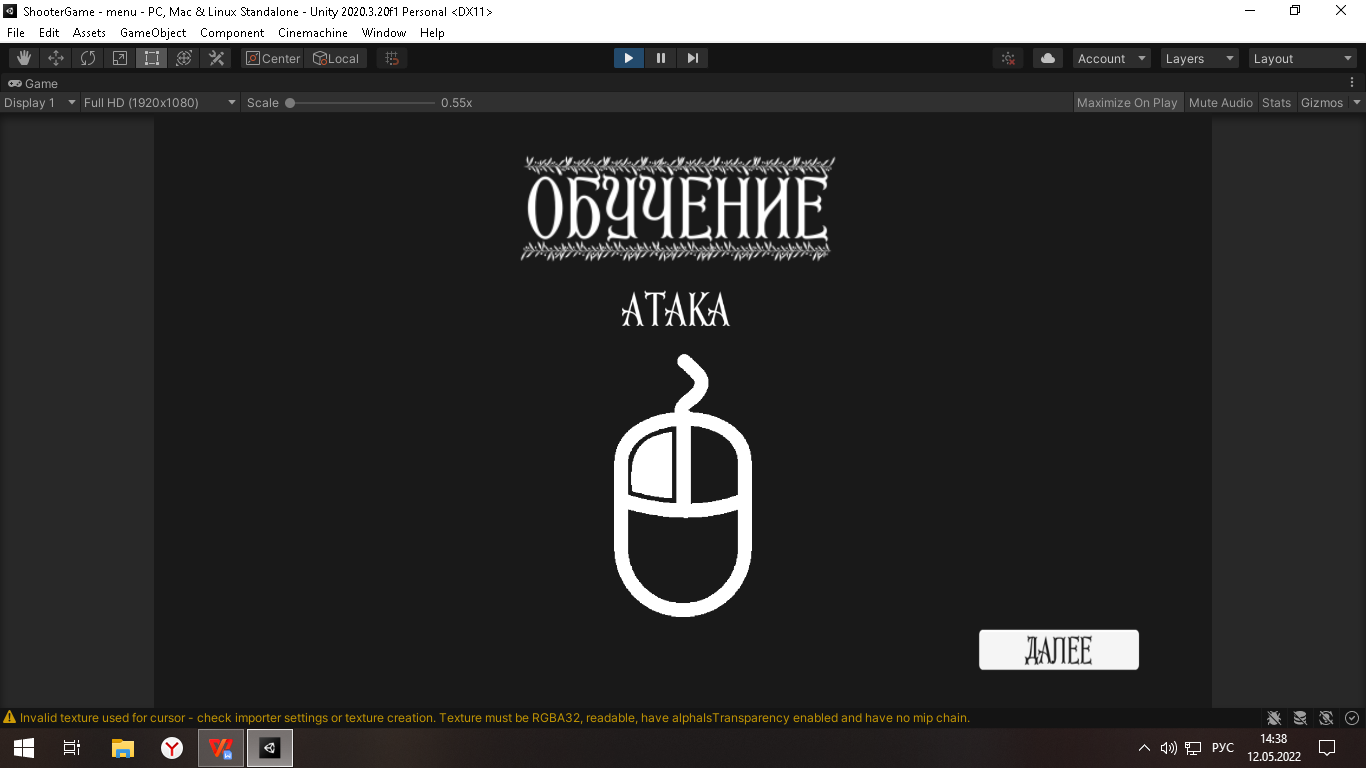


Рисунок 3. Обучение атаки

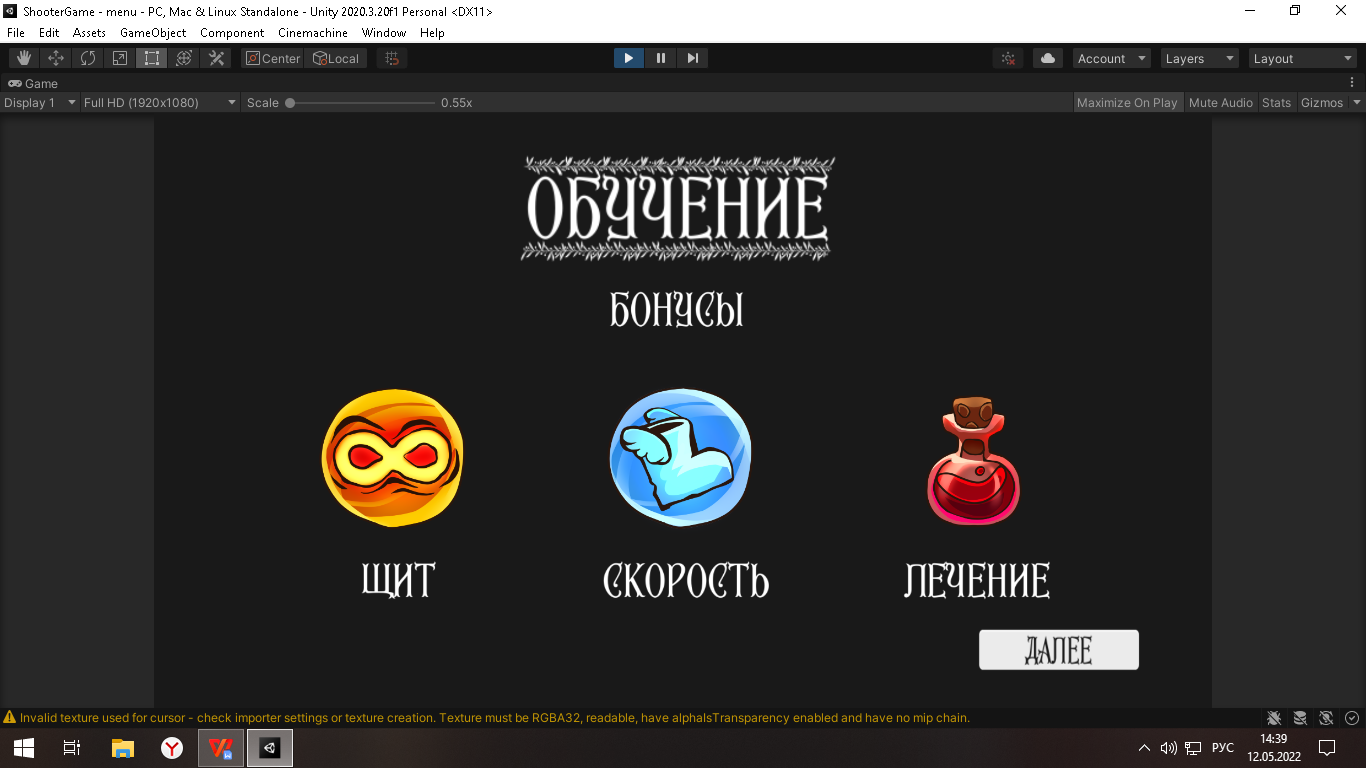


Рисунок 4. Обучение бонусам

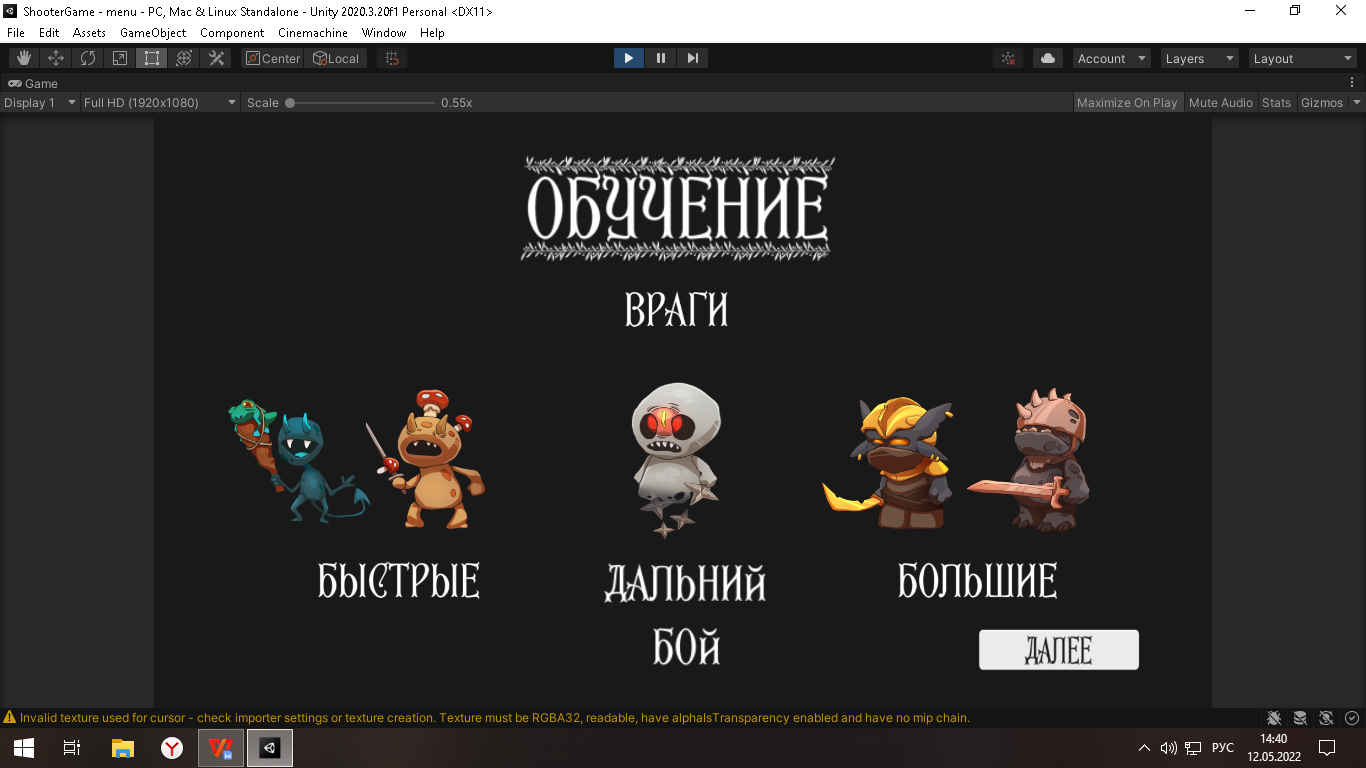


Рисунок 5. Обучение врагам

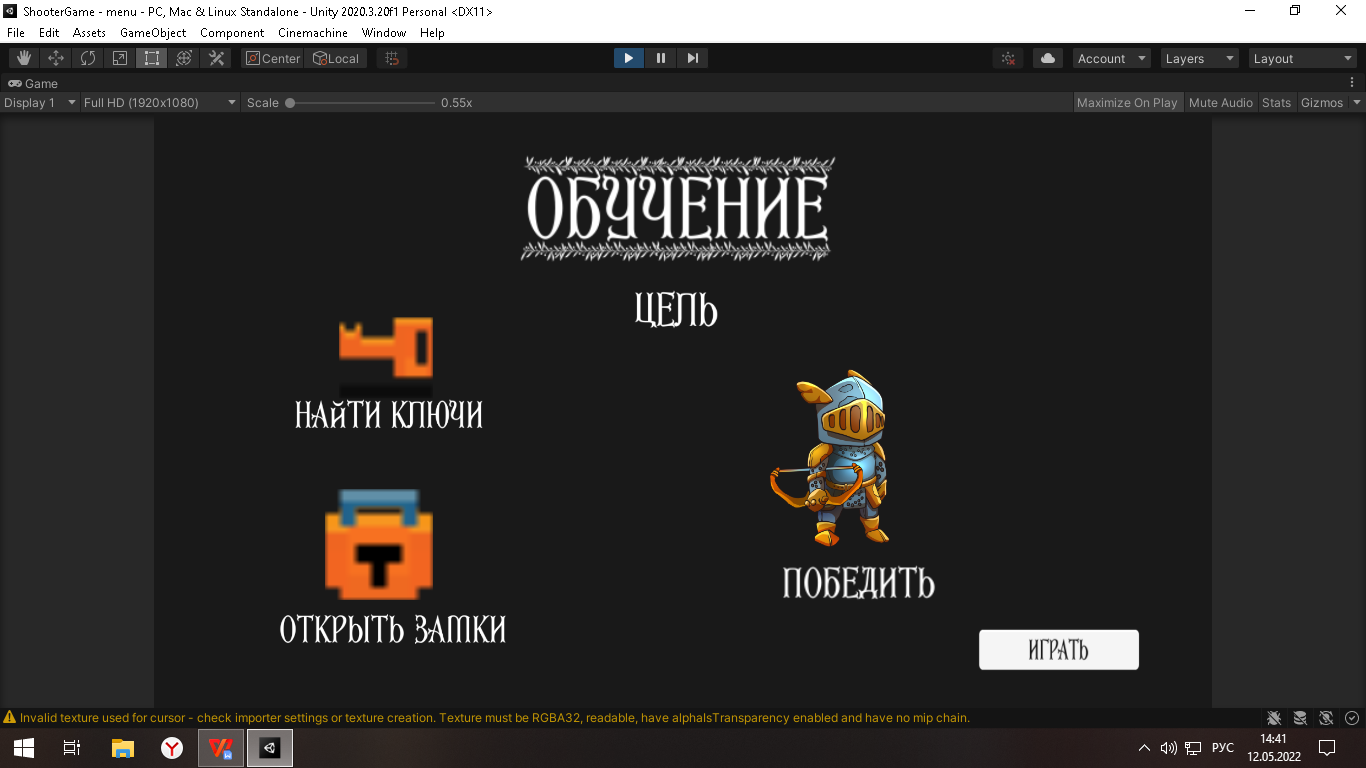


Рисунок 6. Обучение цели

Экран загрузки появляется при начале игры и предоставлен на рисунке 7.

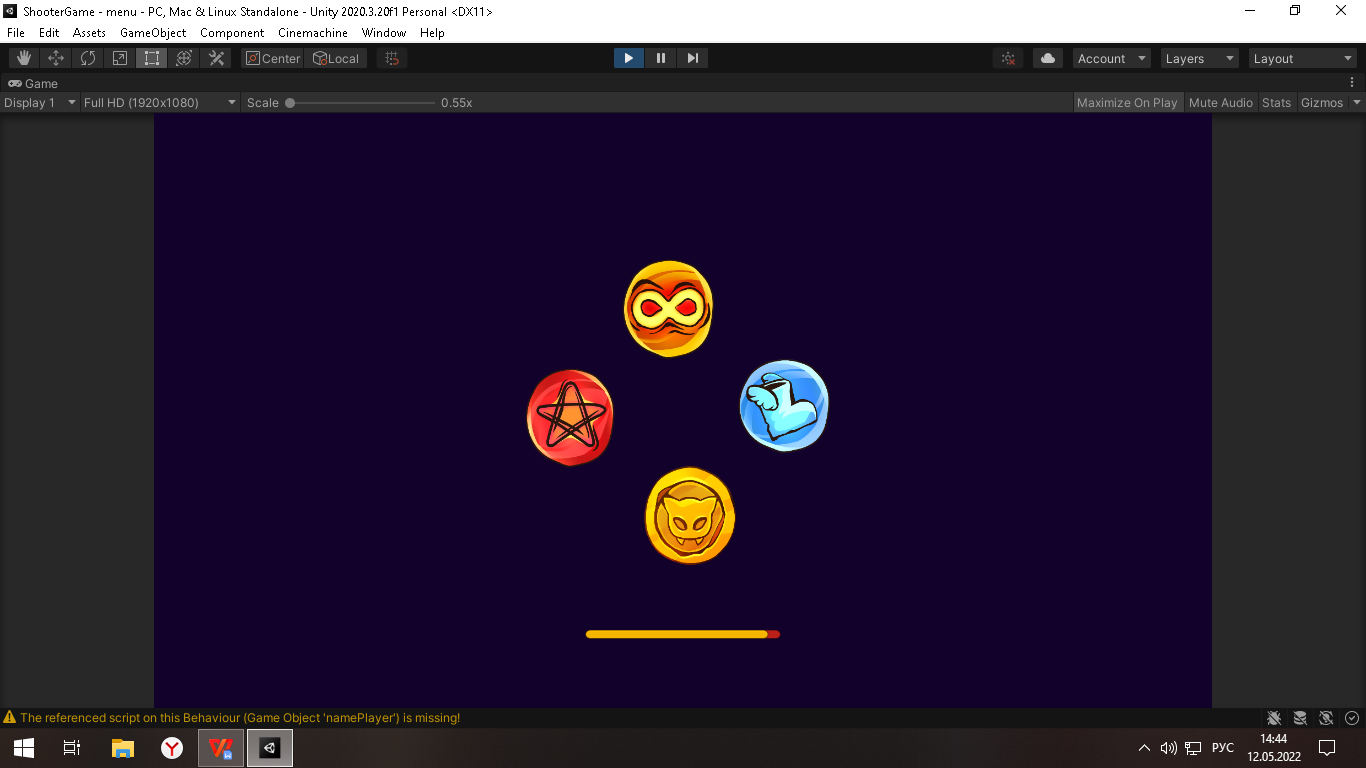


Рисунок 7. Экран загрузки

Так же необходимо реализовать пользовательский интерфейс для основной игровой сцены. На нем должны присутствовать: имя игрока, кнопка паузы, таймер, количество очков за поверженных врагов, и количество очков здоровья. При подборе бонуса необходим визуальный отсчёт до окончания действия бонуса.Скриншот основной игровой сцены предоставлен на рисунке 8.

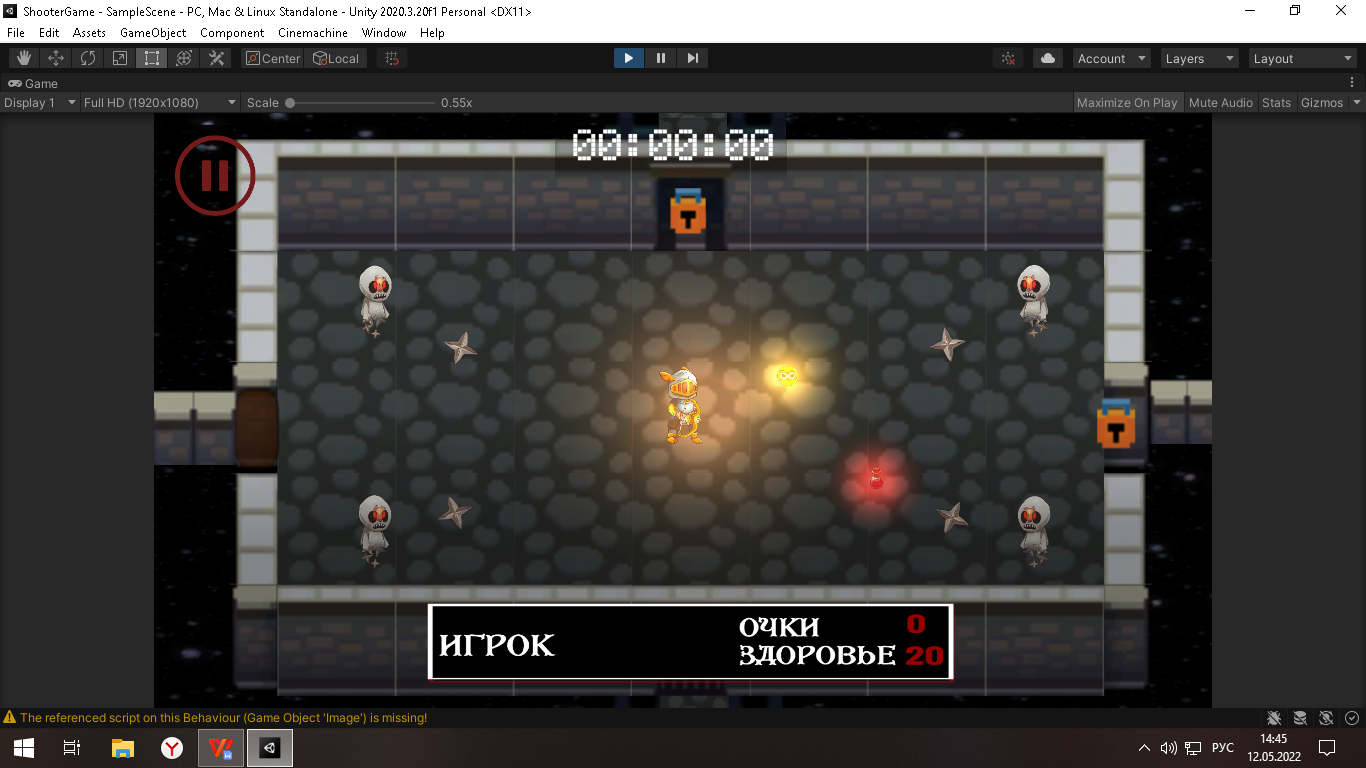


Рисунок 8. Основная игровая сцена

Локация первого уровня, содержащая в себе 8 комнат предоставлена на рисунке 9.

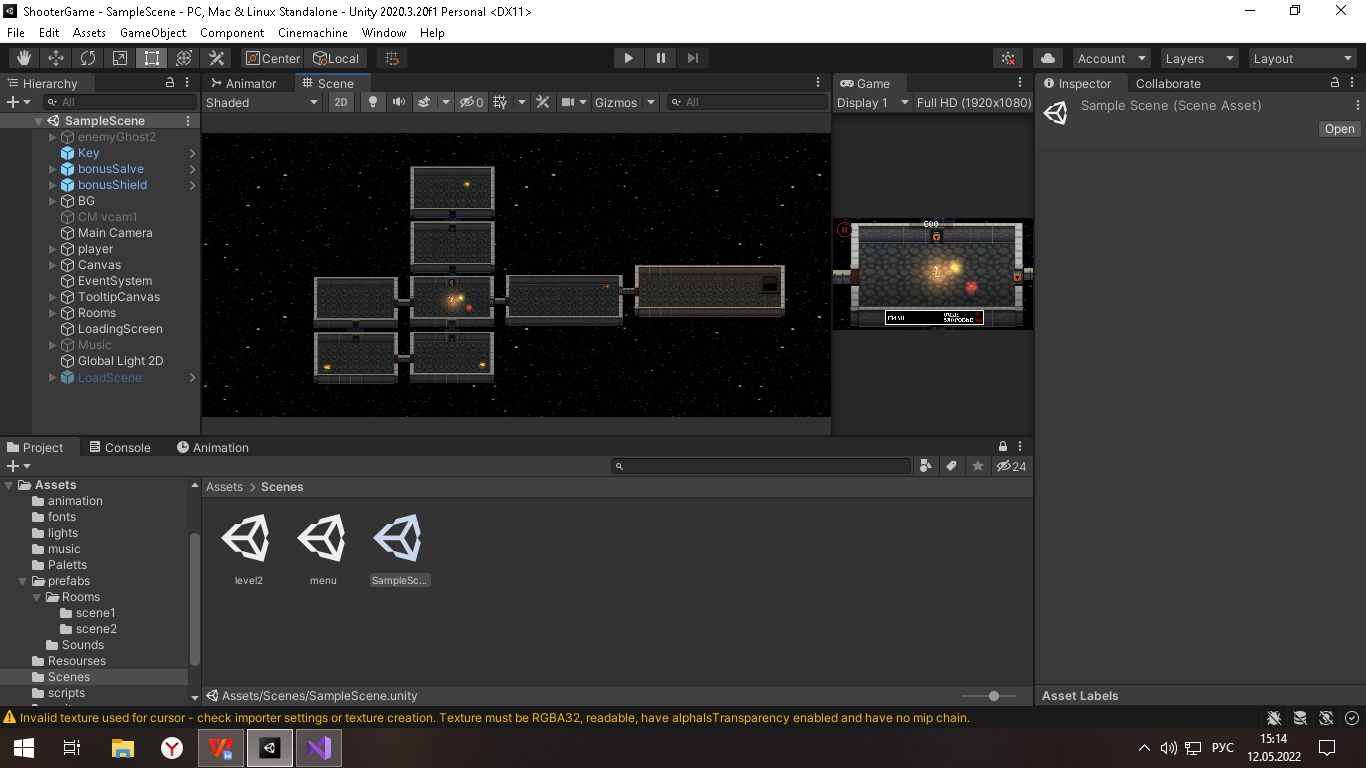


Рисунок 9. Карта первого уровня

Пользовательский интерфейс также необходим для меню паузы, меню проигрыша и меню выигрыша

В меню паузы находятся кнопки продолжить игру и выйти из игры в меню проигрыша. Скриншот меню паузы предоставлен на рисунке 10.

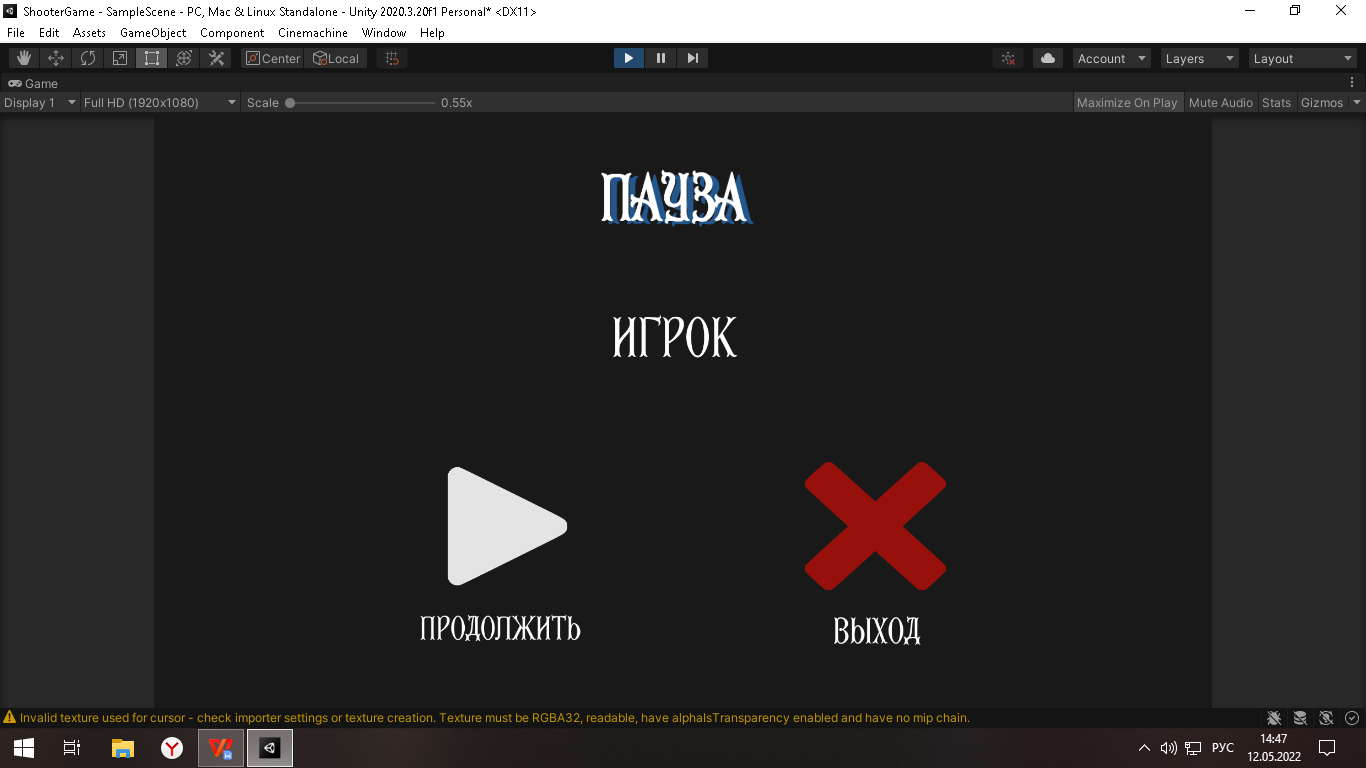


Рисунок 10. Меню паузы

В меню проигрыша находятся кнопки выйти из игры, начать игру заново, удалить все имена. Так же, в виде списка, выведены все игроки, которые играли раньше, и их рекордные очки за поверженных врагов. Скриншот меню проигрыша предоставлен на рисунке 11.

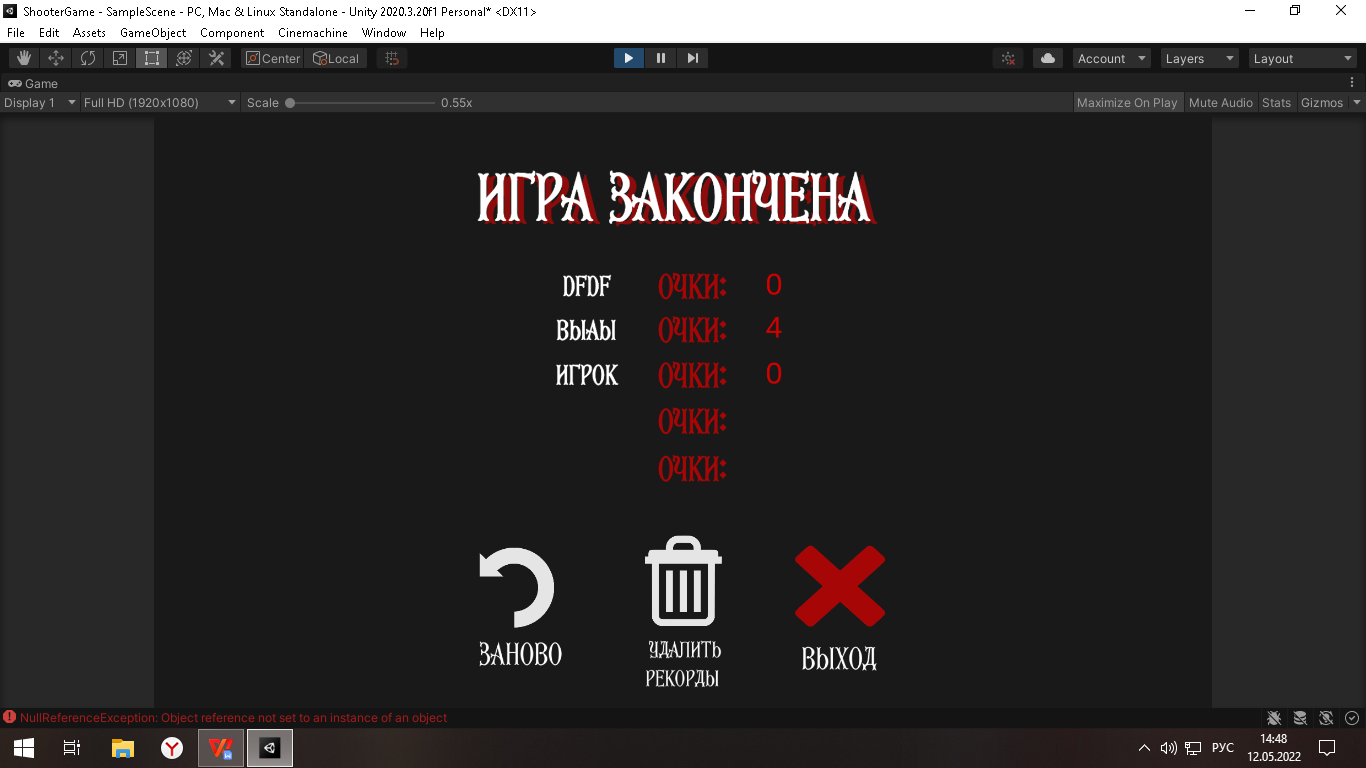


Рисунок 11. Меню проигрыша

В меню выигрыша находятся кнопки выйти из игры, начать игру заново, выбор уровня, удалить все имена. Так же, в виде списка, выведены все игроки, которые играли раньше, и их рекордные очки за поверженных врагов. Скриншот меню выигрыша предоставлен на рисунке 12.

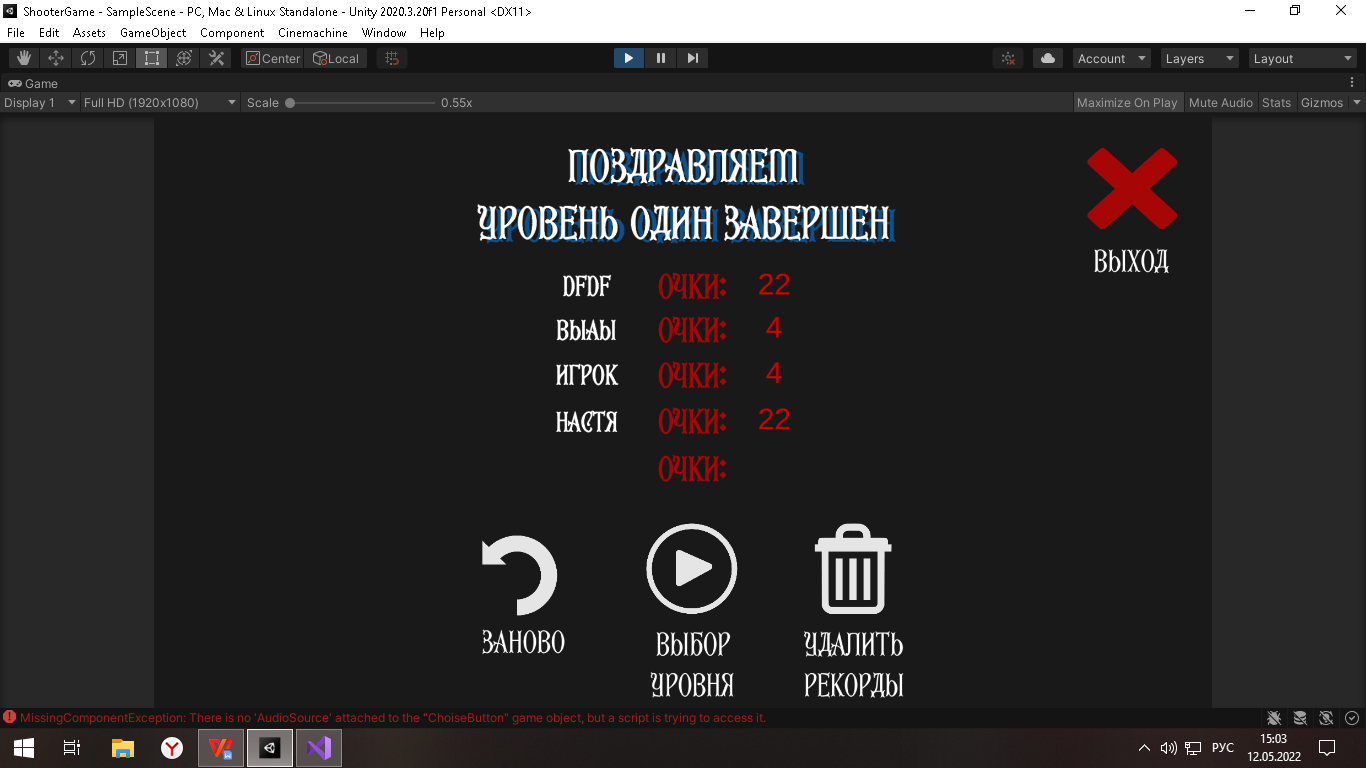


Рисунок 12. Меню выигрыша

В меню выбора уровня находятся кнопки выхода, выбора первого уровня, выбора второго уровня. Меню выбора уровня предоставлен на рисунке 13.

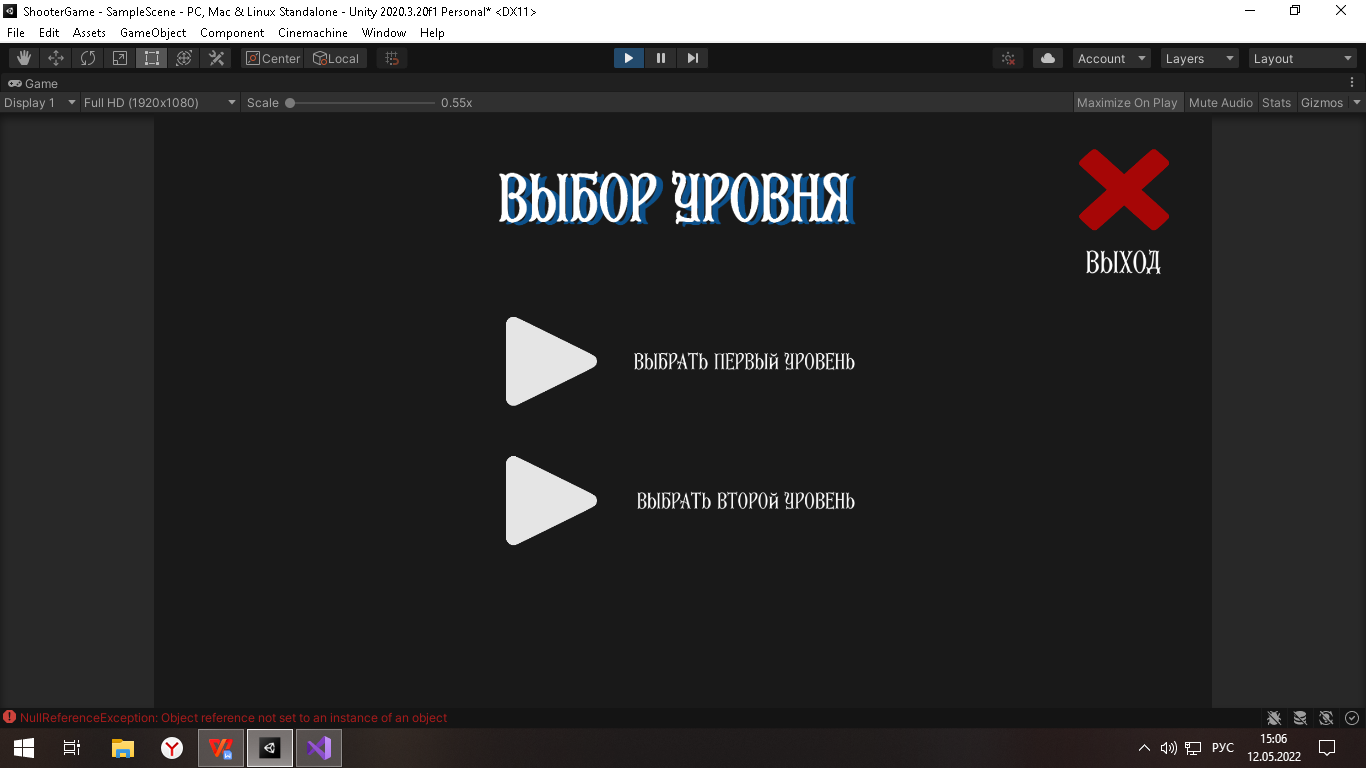


Рисунок 13. Меню выбора уровня

После прохождения первого уровня, игрок перемещается на вторую сцену со вторым уровнем, содержащую в себе 5 комнат Персонаж остаётся тем же, но добавляются 3 вида врагов. Камера увеличивается и уменьшается при входе в определённую комнату.Карта локации предоставлена на рисунке 14.

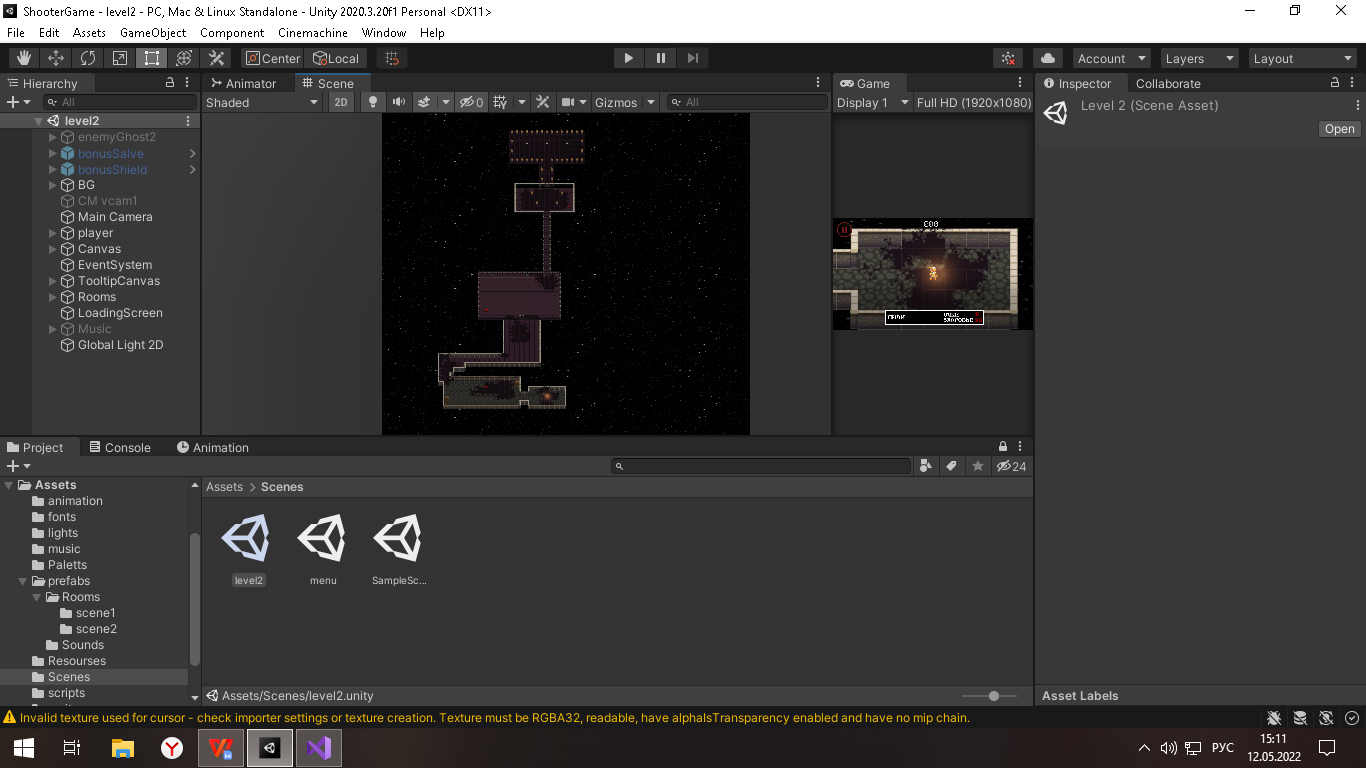


Рисунок 14.Карта локации второго уровня

**Спрайты персонажей**

Для создания спрайтов персонажей была использована программа Adobe Photoshop. Тематика игры - подземелье с рыцарем и монстрами из подземелья. Рыцарь и монстры нарисованы в одной стилистике, но отличаются. Рыцарь имеет при себе лук, а разные монстры разные виды оружия. Всего было нарисовано пять моделей для врагов, и одна модель для рыцаря. Спрайт врагов предоставлен на рисунке 15.

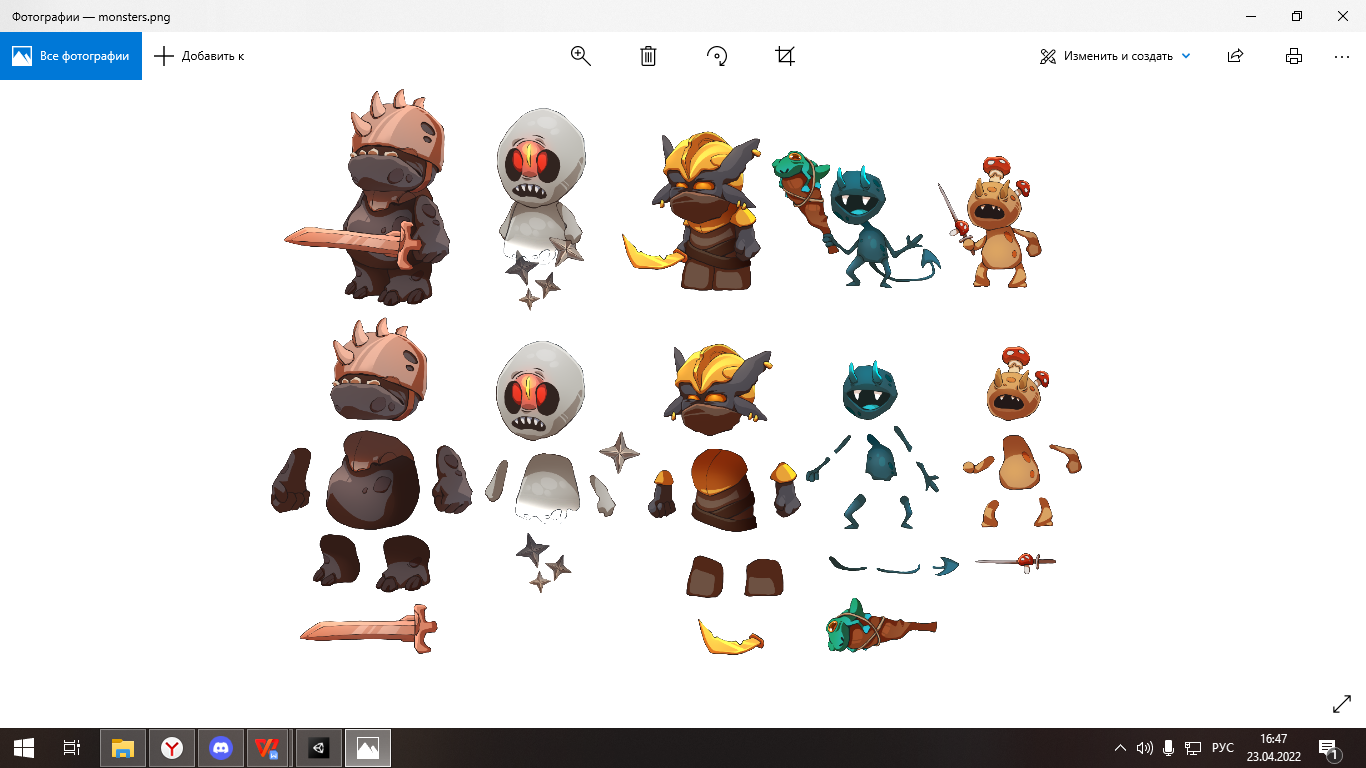


Рисунок 15. Спрайт врагов

На рисунке 16 предоставлен спрайт рыцаря.

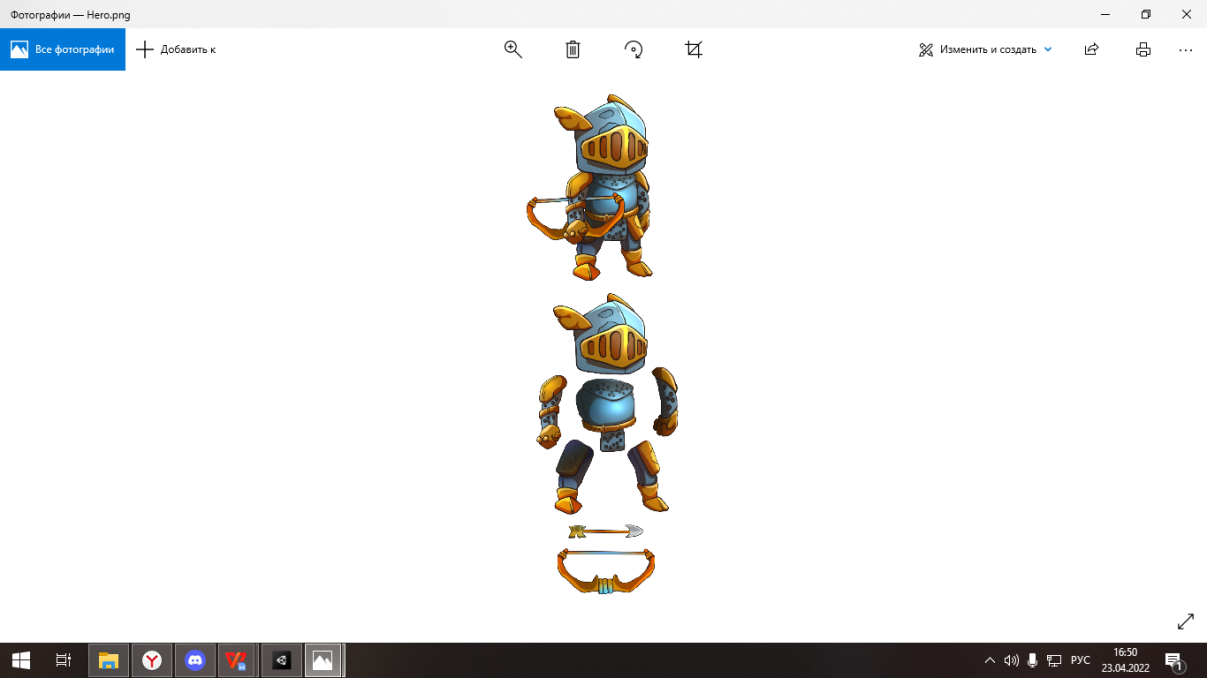


Рисунок 16. Спрайт рыцаря

**Тайлы окружения**

Тайлы - небольшие изображения одинаковых размеров, служащие фрагментами большой картины. В подземельях чаще всего поверхностью выступает земля, в данном проекте были созданы тайлы - булыжники. А для стен были созданы тайлы кирпичной стены. Размер одного тайла 32 на 32 пикселя. Так же были прорисованы проходы между локациями, и крайние тайлы. Реализовать тайлы на Unity необходимо при помощи Tailmap. Был создан спрайт из тайлов и загружен в палитру Tailmap. На рисунке 17, предоставлена палитра из тайлов для создания игрового окружения.

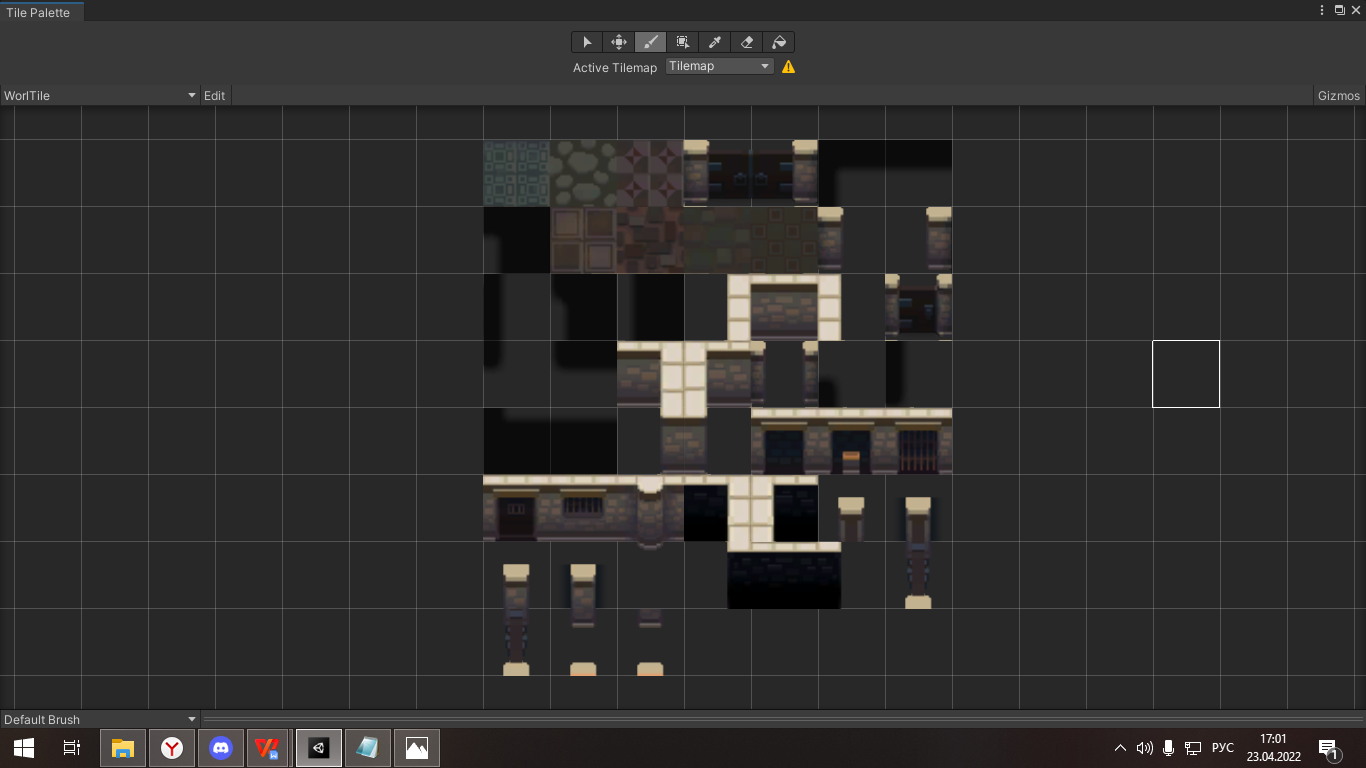


Рисунок 17. Палитра из тайлов окружения

Иконки бонусов так же являются частью окружения, и были прорисованы в Adobe Photoshop. На рисунке 18 предоставлен спрайт бонусов.

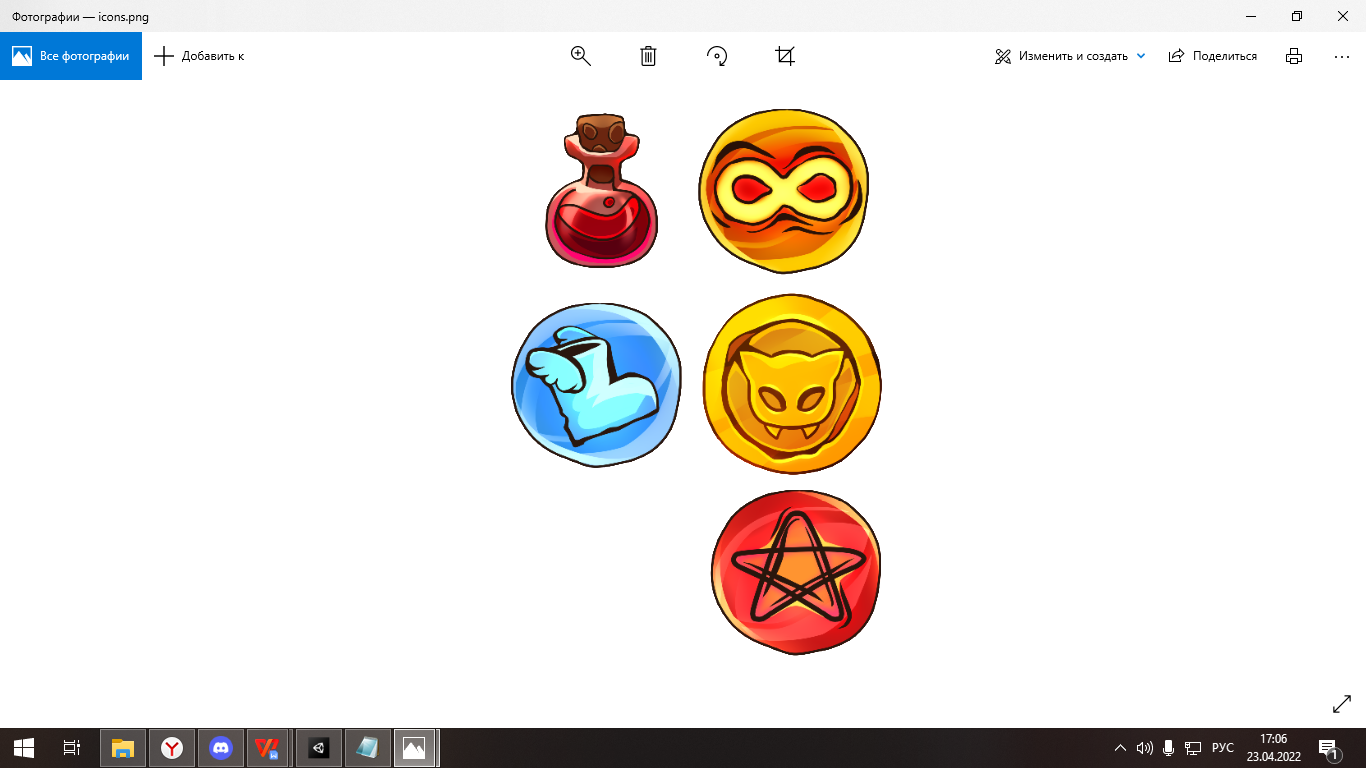


Рисунок 18. Спрайт бонусов

## **Создание макета в Unity**

После запуска Unity создается сцена на которой располагаются объекты будущей игры. Объектами разработки являются:

* Player, который хранит в себе информацию о игроке;
* EnemyGhost- враг - призрак;
* EnemyGrib- враг - монстр с грибами;
* Canvas, который хранит в себе пользовательский интерфейс(UI).
* Main Camera - главная камера игры.
* Rooms, хранит в себе объекты окружения.
* Global Light 2D, хранит в себе настройки освещения

Так же есть объекты и подобъекты, но основные предоставлены выше.

На рисунке 19 предоставлен скриншот с данными объектами, которые располагаются на сцене.

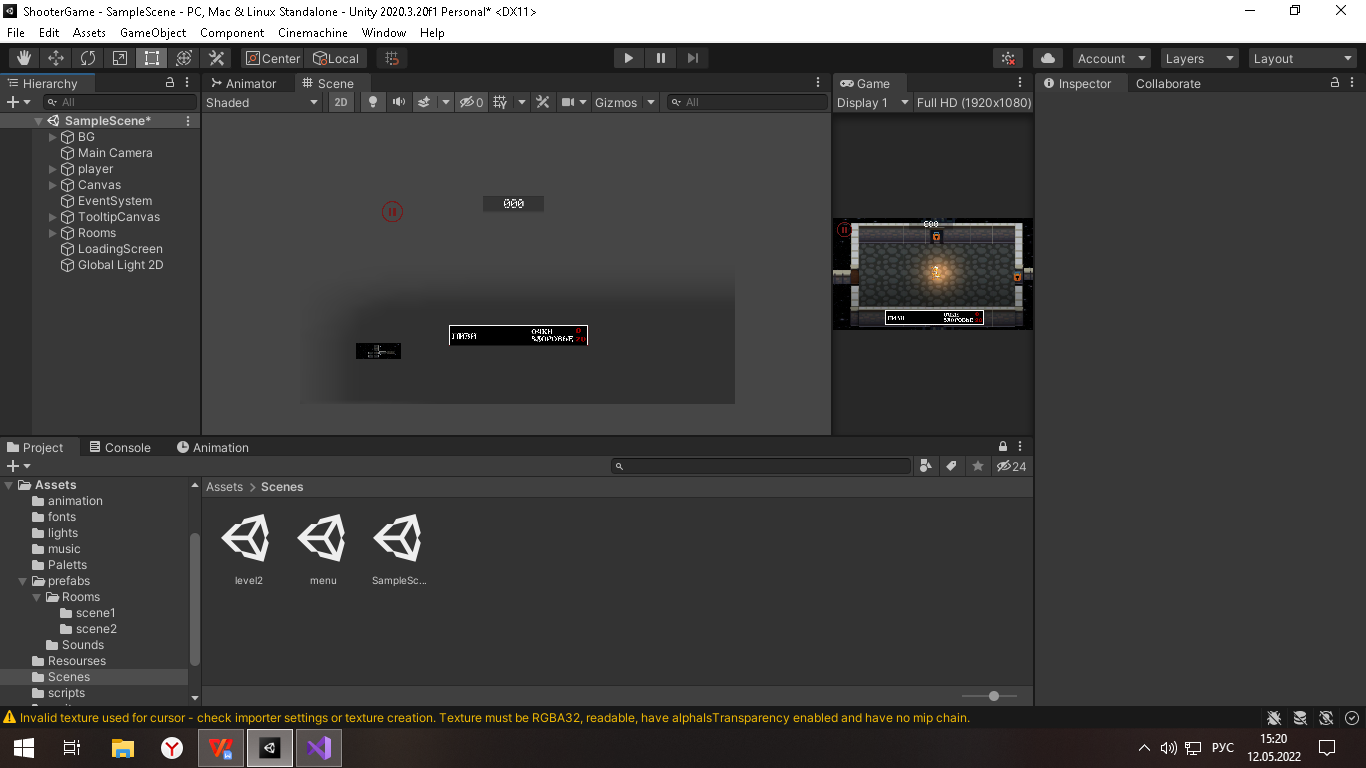


Рисунок 19. Объекты сцены

Далее объектам необходимо задать физические характеристики, чтобы Unity мог с ними взаимодействовать. Физические характеристики Player и виды врагов одинаковы, за исключением параметров в них. На рисунке 20 предоставлены физические характеристики Player.

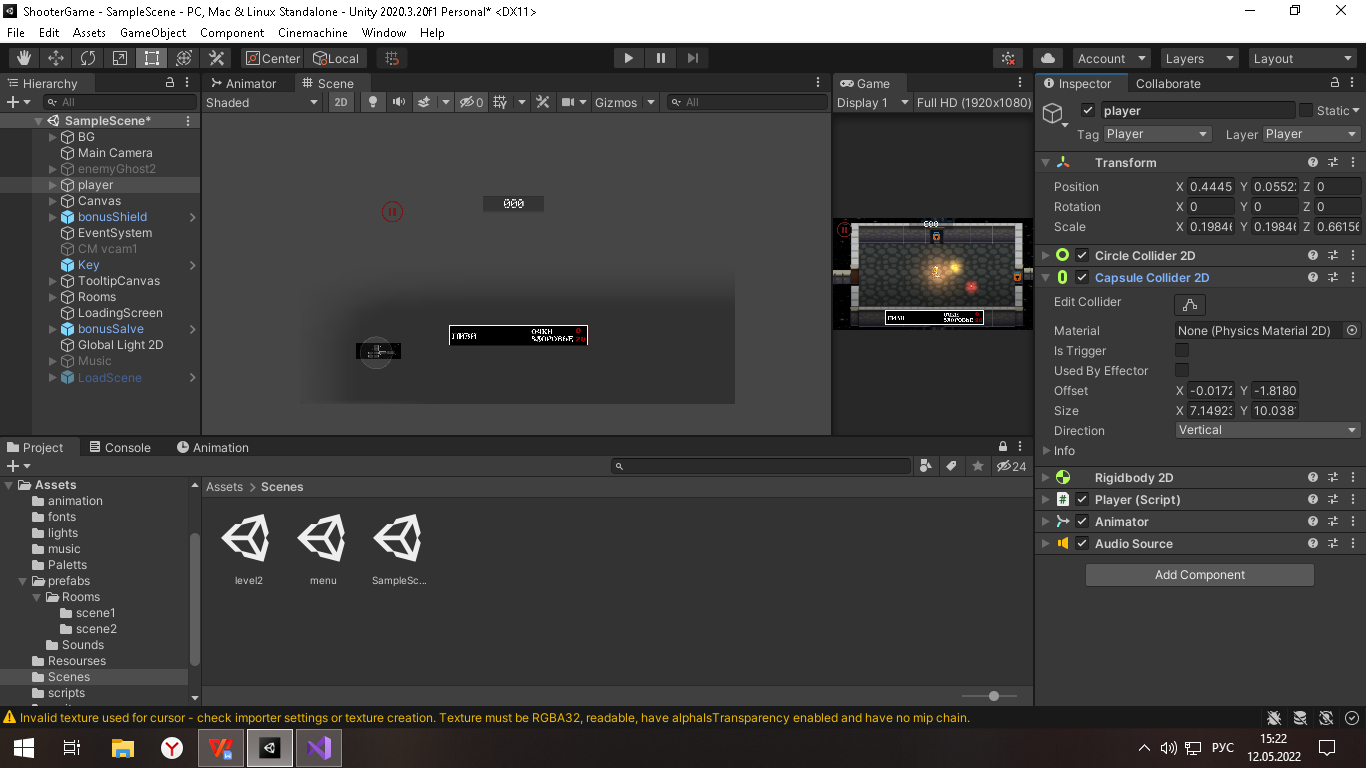


Рисунок 20. Физические характеристики объекта Player

Для бонусов тоже необходимо задать физические характеристики, чтобы Player мог с ними взаимодействовать. bonusSalve является зельем лечения. И предоставлен в виде префаба. Префаб - особый тип ассетов, позволяющий хранить весь GameObject со всеми компонентами и значениями свойств. На рисунке 21 предоставлены физические характеристики bonusSalve.

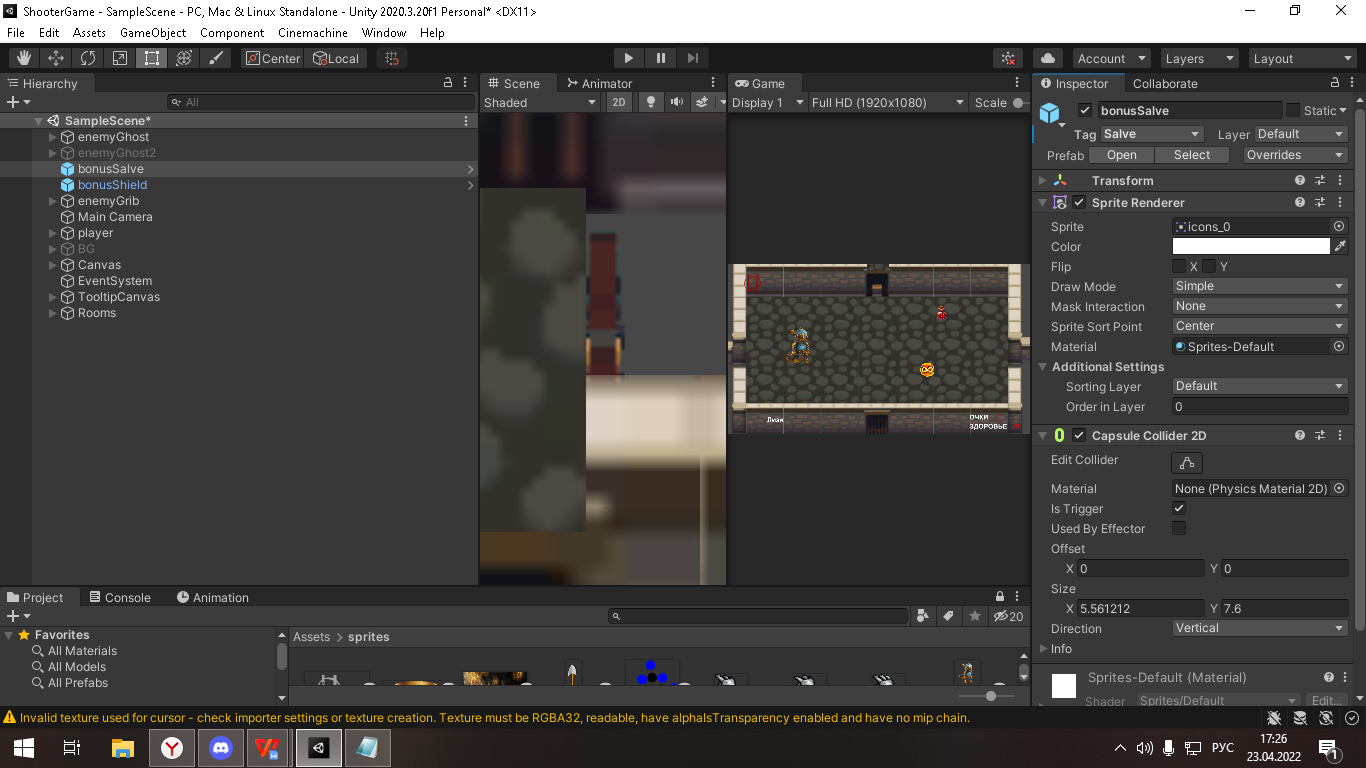


Рисунок 21. Физические характеристики bonusSalve

Главная камера тоже должна содержать физические характеристики, которые предоставлены на рисунке 22.

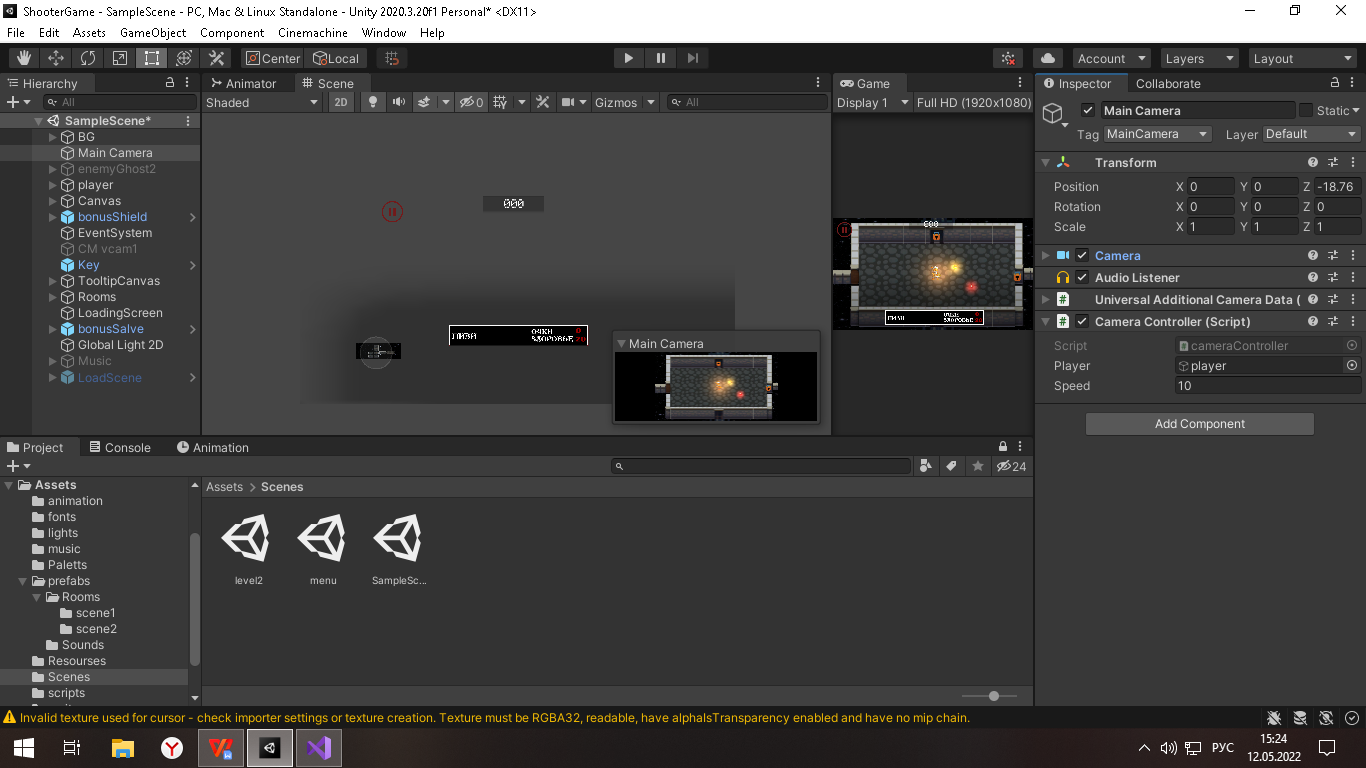


Рисунок 22. Физические характеристики Main Camera

В создании макета так же необходимо учитывать пользовательский интерфейс, который предоставлен в виде полотна - Canvas. На нем отображаются:

* Кнопка паузы;
* Очки за поверженных врагов;
* Очки здоровья;
* Тень сцены;
* Имя игрока;
* Таймер и его тень;
* Бонусный эффект щита;
* Панель проигрыша игрока;
* Панель выигрыша игрока;
* Панель выбора уровня;
* Панель паузы.

На рисунке 23 предоставлен скриншот объекта Canvas на сцене.

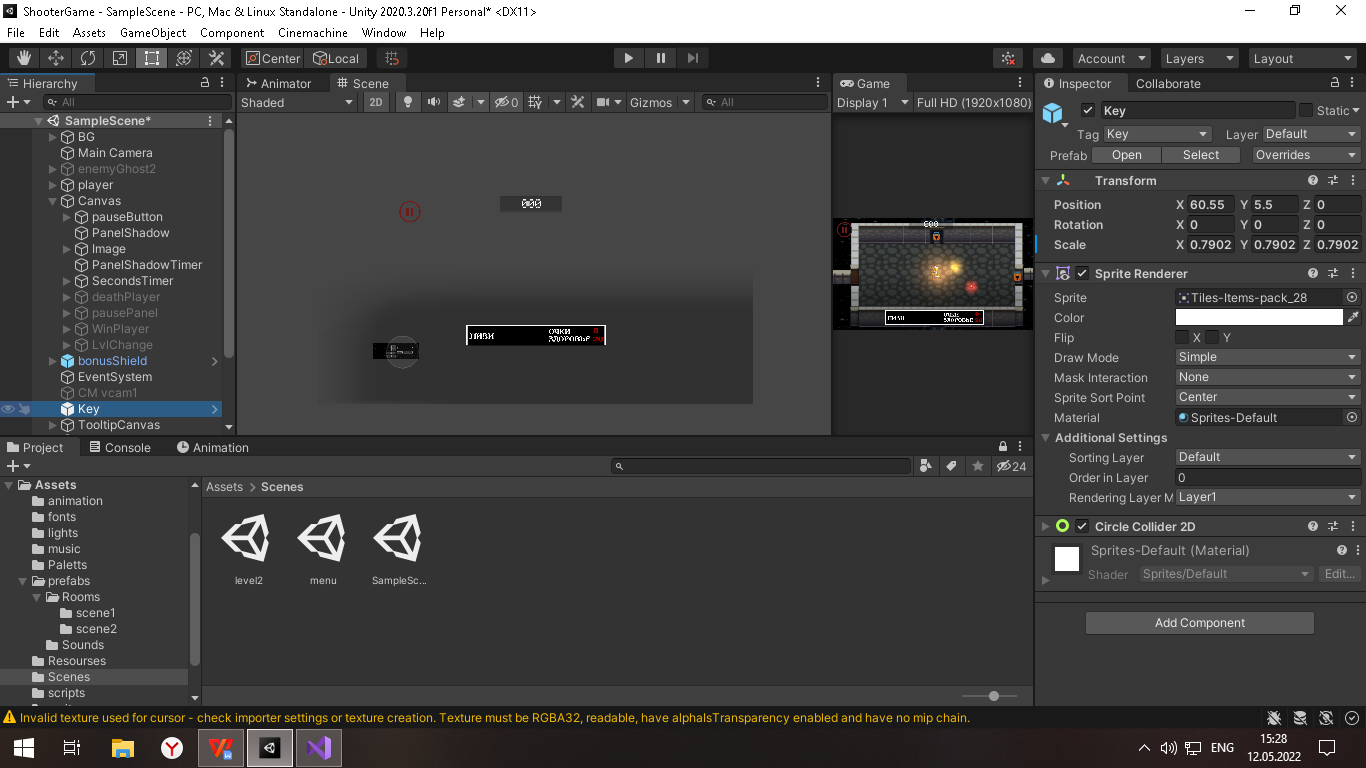


Рисунок 23.Объекты Canvas

## **Написание скриптов**

Для правильного функционирования игрового приложения необходимо написать сценарии работы для игровых механик.

## **Реализация игрового персонажа Player**

Игроку необходимо передвигаться на клавши WASD, стрелять из оружия, получать очки за победу над врагом, подбирать бонусы, воспроизводить анимацию ходьбы при передвижении, получать урон от противников,умирать если здоровье упадёт ниже нуля, подбирать ключи и открывать двери. Ниже предоставлен скрипт игрового объекта Player.

public class Player : MonoBehaviour

{

private Rigidbody2D rb;

private bool facingRight = true;

private Animator anim;

private Vector2 moveInput;

private Vector2 moveVelocity;

private int score;

private AudioSource audioSource;

[Header("effects")]

public GameObject salveEffect;

public GameObject shieldEffect;

public GameObject shield;

public GameObject aura;

[Header("music")]

public GameObject soundSalve;

public GameObject soundShield;

public GameObject soundKey;

public AudioClip[] soundStep;

public GameObject soundDoor;

[Header("controls")]

public float speed;

public int health;

public GameObject PanelDeath;

public GameObject PanelWin;

public Bonus shieldTimer;

public Slider slider;

public int levelToLoad;

[Header("Text")]

public Text textScore;

public Text healthDisplay;

public Text DisplayName;

[Header("Key")]

public GameObject keyIcon;

public GameObject keyEffect;

Стартовая функция содержит в себе информацию о звуке, теле игрока, анимации, отображении имени игрока, отображения здоровья игрока, и записывает имя игрока в библиотеку PlayerPrefs.

private void Start()

{

audioSource = GetComponent<AudioSource>();

rb = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

healthDisplay.text = "" + health;

DisplayName.text = PlayerPrefs.GetString("playerName");

}

Функция Update изменят здоровье чтобы оно не превышало 20, так же функция позволяет взять значения клавиш. Записывает набранные игроком очки в библиотеку.

private void Update()

{

if (health >= 20)

{

health = 20;

healthDisplay.text = "" + health;

}

moveInput = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));

moveVelocity = moveInput.normalized \* speed;

PlayerPrefs.SetInt("Score", score);

}

Функция FixedUpdate позволяет игроку передвигаться с определённой скоростью. И при помощи функции Flip разворачивает игрока либо влево либо вправо. Так же воспроизводится анимация ходьбы, если игрок идёт. Если здоровье игрока падает ниже нуля, то появляется панель смерти и игра оканчивается.

private void FixedUpdate()

{

rb.MovePosition(rb.position + moveVelocity \* Time.fixedDeltaTime);

if (facingRight == false && moveInput.x > 0)

{

Flip();

}

else if (facingRight == true && moveInput.x < 0)

{

Flip();

}

if (moveInput.x == 0 && moveInput.y == 0)

{

anim.SetBool("isRunning", false);

}

else

{

anim.SetBool("isRunning", true);

}

if (health <= 0)

{

PanelDeath.SetActive(true);

Destroy(gameObject);

}

if (Input.GetKey(KeyCode.LeftShift))

{

speed = 19;

aura.SetActive(true);

}

else

{

speed = 8;

aura.SetActive(false);

}

}

void Flip()

{

facingRight = !facingRight;

Vector3 Scaler = transform.localScale;

Scaler.x \*= -1;

transform.localScale = Scaler;

}

Функция StepSoundPlay воспроизводит звук шагов игрока. Его реализация происходит через окно анимации.

private void StepSoundPlay()

{

audioSource.PlayOneShot(soundStep[Random.Range(0, soundStep.Length)]);

}

Функция OnTriggerEnter2D отвечает за то когда игрок попадает в зону данного предмет. При подборе зелья лечения, появляется эффект зелья и его звук, зелье лечит на 5 очков здоровья. После подбора объекта он уничтожается. Тот же эффект происходит с бонусом щита. Щит блокирует входящий урон по игроку, но с каждым полученым уроном, время его действия уменьшается. Подбор ключа происходит таким же образом как и бонусы, но исчезает при открытие двери.

private void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Salve"))

{

ChangeHealth(5);

Instantiate(salveEffect, other.transform.position, Quaternion.identity);

Destroy(other.gameObject);

Instantiate(soundSalve, transform.position, Quaternion.identity);

}

else if (other.CompareTag("Shield"))

{

if (!shield.activeInHierarchy)

{

shield.SetActive(true);

shieldTimer.gameObject.SetActive(true);

shieldTimer.isCooldown = true;

Instantiate(soundShield, transform.position, Quaternion.identity);

Instantiate(shieldEffect, other.transform.position, Quaternion.identity);

Destroy(other.gameObject);

}

else

{

shieldTimer.ResetTimer();

Destroy(other.gameObject);

}

}

else if (other.CompareTag("Key"))

{

if (keyIcon.activeInHierarchy)

{

}

else if (!keyIcon.activeInHierarchy)

{

Instantiate(soundKey, transform.position, Quaternion.identity);

keyIcon.SetActive(true);

Destroy(other.gameObject);

}

}

}

Функция OnTriggerStay2D работает почти также, как предыдущая функция, за некоторыми исключениями. Если игрок вошел в зону действия двери, имея при себе ключ, то дверь открывается с эфектом открытия двери и своим звуком. Если игрок вошел в зону действия выхода, то загружается панель выигрыша.

private void OnTriggerStay2D(Collider2D other)

{

if (other.CompareTag("Door") && keyIcon.activeInHierarchy)

{

Instantiate(keyEffect, other.transform.position, Quaternion.identity);

keyIcon.SetActive(false);

other.gameObject.SetActive(false);

Instantiate(soundDoor, transform.position, Quaternion.identity);

}

if (other.CompareTag("Exit") || Input.GetKey(KeyCode.Escape))

{

//SceneManager.LoadScene(2);

PanelWin.SetActive(true);

}

}

Функция ChangeHealth работает по всему проекту и изменят значение здоровья игрока, так же на эту функцию влияет щит.

public void ChangeHealth(int healthValue)

{

if (!shield.activeInHierarchy || shield.activeInHierarchy&& healthValue > 0)

{

health += healthValue;

healthDisplay.text = "" + health;

}

else if (shield.activeInHierarchy && healthValue < 0)

{

shieldTimer.ReduseTime(healthValue);

}

}

Функция Kill позволяет игроку зарабатывать очки, и вызывается она через скрипт Enemy.

public void Kill()

{

score++;

textScore.text = PlayerPrefs.GetInt("Score").ToString();

}

}

Далее необходимо привязать скрипт к Unity. В приложение А предоставлено подключение скрипта Player к игровым объектам.

## **Реализация локации и пользовательского интерфейса**

Всего в игре три сцены:

* Меню;
* Уровень один;
* Уровень два.

Скрипт отвечающий за меню предоставлен ниже.

public class Menu : MonoBehaviour

{

[Header("buttons")]

public InputField inGame;

public GameObject playButton;

public GameObject SaveNameButton;

public GameObject ContinueButton;

[Header("scenes")]

public Text PlayerName;

public Slider slider;

public GameObject loadingScene;

public GameObject[] rules;

public int levelToLoad;

[Header("music")]

public AudioClip mainSound;

private AudioSource audioSource;

int saveNameGame = 0;

Функция Start берет данные для воспроизведения звуков.

private void Start()

{

audioSource = GetComponent<AudioSource>();

audioSource.clip = mainSound;

audioSource.Play();

inGame.GetComponent<Text>();

}

Функция Update отвечает за то, чтобы кнопка начать игру была доступна только после ввода имени. Так же функция сохраняет имя в библиотеку, для дальнейшего использования.

private void Update()

{

if (PlayerPrefs.HasKey("playerName"))

{

ContinueButton.GetComponent<Button>().interactable = true;

}

if (inGame.text == "")

{

SaveNameButton.GetComponent<Button>().interactable = false;

}

else

{

SaveNameButton.GetComponent<Button>().interactable = true;

PlayerPrefs.SetString("playerName", inGame.text.ToString());

if (saveNameGame == 1)

{

playButton.GetComponent<Button>().interactable = true;

}

}

}

Функция SaveName сохраняет имя в библиотеку, для дальнейшего использования, и является функцией для кнопки сохранить имя.

public void SaveName()

{

PlayerName.text = PlayerPrefs.GetString("playerName");

Debug.Log(inGame.text.ToString());

saveNameGame = 1;

}

Функция StartGame привязана к кнопке продолжить, и к последней кнопке правил при нажатии на кнопку запускает экран загрузки, после загрузки загружается сцена первого уровня.

public void StartGame()

{

Time.timeScale = 1f;

StartCoroutine(LoadingScreenOnFade());

}

IEnumerator LoadingScreenOnFade()

{

AsyncOperation operation = SceneManager.LoadSceneAsync(levelToLoad);

loadingScene.SetActive(true);

while (!operation.isDone)

{

float progress = Mathf.Clamp01(operation.progress / 0.9f);

slider.value = progress;

yield return null;

}

}

Функции NextRules отвечают за показ правил, и появляются при нажатии на кнопку начать игру.

public void NextRules1()

{

rules[0].SetActive(true);

}

public void NextRules2()

{

rules[0].SetActive(false);

rules[1].SetActive(true);

}

public void NextRules3()

{

rules[1].SetActive(false);

rules[2].SetActive(true);

}

public void NextRules4()

{

rules[2].SetActive(false);

rules[3].SetActive(true);

}

public void NextRules5()

{

rules[3].SetActive(false);

rules[4].SetActive(true);

}

Функция Exit привязана к кнопке выхода, и закрывает игру.

public void Exit()

{

Debug.Log("quit");

Application.Quit();

}

}

## **Реализация генерации и уничтожения объектов**

Тут общие сведения

## **Реализация врагов и оружия**

Тут общие сведения

## **Внедрение музыкального сопровождения**

Тут общие сведения

# **ТЕСТИРОВАНИЕ**

## **Функциональное тестирование**

## **Юзабилити тестирование**

## **Исправление ошибок**

## **Экономические расчеты**

# **ЗАКЛЮЧЕНИЕ**

Тут общие сведения

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. [https://www.wikiwand.com/ru/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F\_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85\_%D0%B8%D0%B3%D1%80#/%D0%9F%D0%BE\_%D0%AE%D0%B3%D0%B0%D0%B9\_(2008)](https://www.wikiwand.com/ru/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80" \l "/%D0%9F%D0%BE_%D0%AE%D0%B3%D0%B0%D0%B9_(2008))
2. [https://ru.wikipedia.org/wiki/Компьютерная\_стратегическая\_игра#:~:text=Стратегическая%20игра%20—%20жанр%20компьютерных,(M.U.L.E.%2C%20Spaceward%20Ho!)%20и%20другие](https://ru.wikipedia.org/wiki/Компьютерная_стратегическая_игра" \l ":~:text=Стратегическая%20игра%20—%20жанр%20компьютерных,(M.U.L.E.%2C%20Spaceward%20Ho!)%20и%20другие)
3. <https://dic.academic.ru/dic.nsf/ruwiki/977044>
4. <https://ru.wikipedia.org/wiki/Action>
5. <https://docs.microsoft.com/ru-ru/dotnet/csharp/>
6. <https://docs.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2019>
7. <https://ru.wikipedia.org/wiki/Unity_(игровой_движок)>
8. <https://docs.unity3d.com/Manual/Overview2D.html>

# **ПРИЛОЖЕНИЯ**

**ПРИЛОЖЕНИЕ А**

Подключение скрипта Player к игровым объектам

