Содержание

Введение

Цель данной выпускной квалификационной работы состоит в разработке игрового проекта на Unity под названием «Bullet Hell». Для достижения поставленной цели необходимо, изучить игровой рынок на наличие игр в похожем жанре, и во-вторых, изучить и проанализировать возможности визуального редактора. В выполнении этих задач может помочь использование ресурсов сети Интернет, а также обширные познания в области игровой индустрии для более детального анализа различных игровых проектов.

Также необходимо оценить актуальность выбранной темы и обосновать выбор версии движка Unity для разработки проекта. Перед разработкой необходимо проанализировать жанр разрабатываемого проекта, создать концепт, на базе которого будет вестись разработка проекта, определиться с механиками, которые будут включены в игру, со стилистикой проекта и выявить целевую аудиторию.

После разработки провести ряд тестов для выявления возможных ошибок и проанализировать расходы на разработку.

1. Постановка задачи и обзор аналогов

1.2 Характеристика организации

ООО «Д-ЛинкТрейд» - компания-издатель, занимается разработкой видеоигр, а также их издательством и сопровождением. Основана в 2003 году и существует по сей день.

По основным показателям отчетности за 2023 год (Рисунок 1) сумма доходов составила 139 миллионов рублей, сумма расходов – 135 миллионов рублей, а прибыль – 3.6 миллиона рублей

Рисунок 1 – Основные показатели отчетности за 2023 год

По финансовому анализу отчетности с 2016 года по 2023 год (Рисунок 2) видны серьезные темпы роста компании

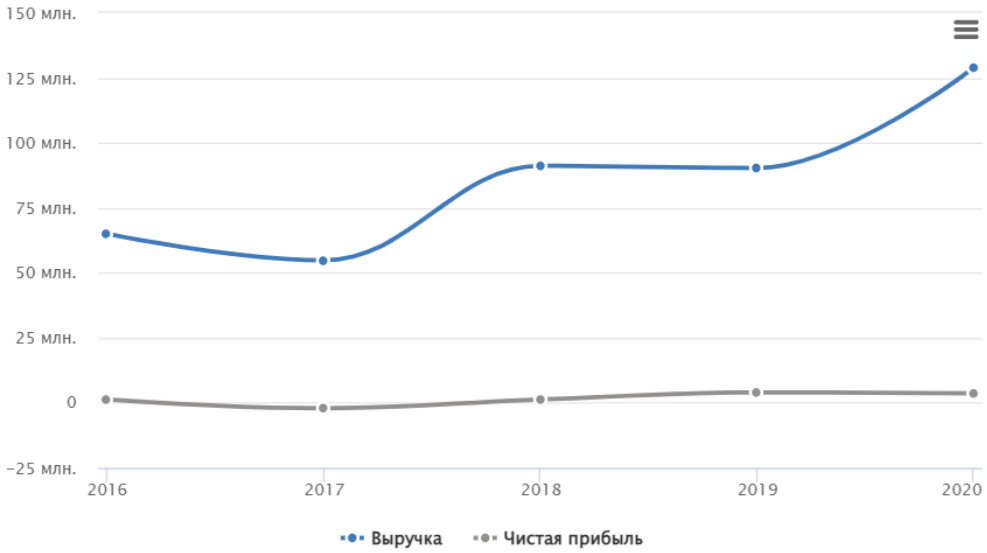

Рисунок 2 – Финансовый анализ отчетности

1.1.2 Структура организация

ООО «Д-ЛинкТрейд» является самостоятельной структурой. Функциональная структура ООО «Д-ЛинкТрейд» разветвленная, во главе стоит руководитель, которому подчиняются все отделы (Рисунок 3).

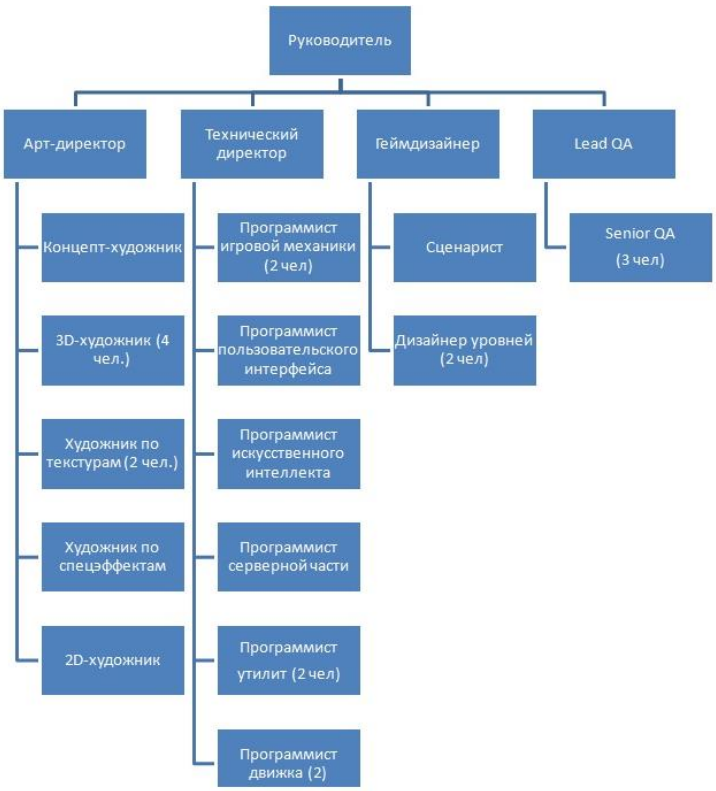

Рисунок 3 – Функциональная структура

1.2 Постановка задачи

1.2.1 Назначение разработки

Игровой проект предназначен для размещения в цифровых магазинах игр и досуговых целях пользователей. Требуется разработать приложение, а релизом, аналитикой, обновлениями и AB тестами будут заниматься другие сотрудники компании.

1.3 Актуальность выбранной темы

На сегодняшний день разработка игр является одной из самых больших индустрий развлечений в мире, обгоняя по скорости развития даже индустрию кино. Сама же киноиндустрия начала заимствовать инструменты, созданные для разработки игр для создания дорогих сцен с компьютерной графикой. Более миллиарда человек играют на своих устройствах. Но нельзя ограничивать игровую индустрию только разработкой игр, в нее также входят следующие элементы:

1) игровые платформы;

2) программное обеспечения для создания игр;

3) издание и распространение.

Все это говорит о том, что это очень быстро развивающаяся индустрия с огромным потенциалом.

1.4 Анализ рынка видеоигр

1.4.1 Общий анализ

Большую долю рынка видеоигр занимают видеоигры для мобильных устройств. На момент 2022 года – это половина от всего рынка разработки видеоигр.

Теперь сравним сколько денег вкладывают игроки в игры на разных платформах (Рисунок 4). Пользователи в мобильных играх готовы вкладывать гораздо больше, нежели пользователи на других платформах.

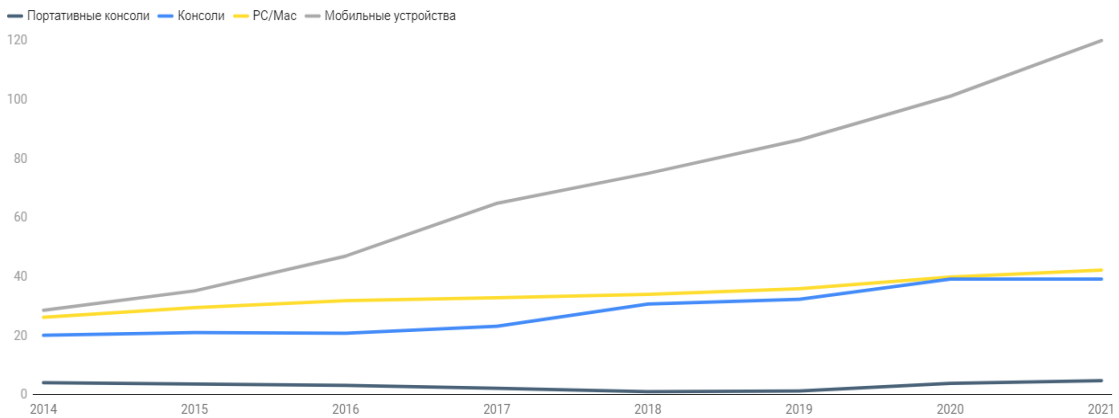

Рисунок 4 - Траты на игры в зависимости от платформы, $ млрд.

Судя по выше приведенным данным, рынок видеоигр для мобильных устройств хоть и является самым успешным в отрасли но игры на ПК в перспективе приносят большие доходы далее будем рассматривать игры на ПК.

1.4.2 Анализ рынка видеоигр для ПК

Для разработки игры требуется проанализировать не только в каких сегментах большее количество игроков и больше прибыль, но и определиться с платежеспособной аудиторией, популярными жанрами.

На рынке игр для ПК самой платежеспособной аудиторией считаются Япония и Южная Корея, после которых идут США, Австралия, Китай и т.д. (Рисунок 5)

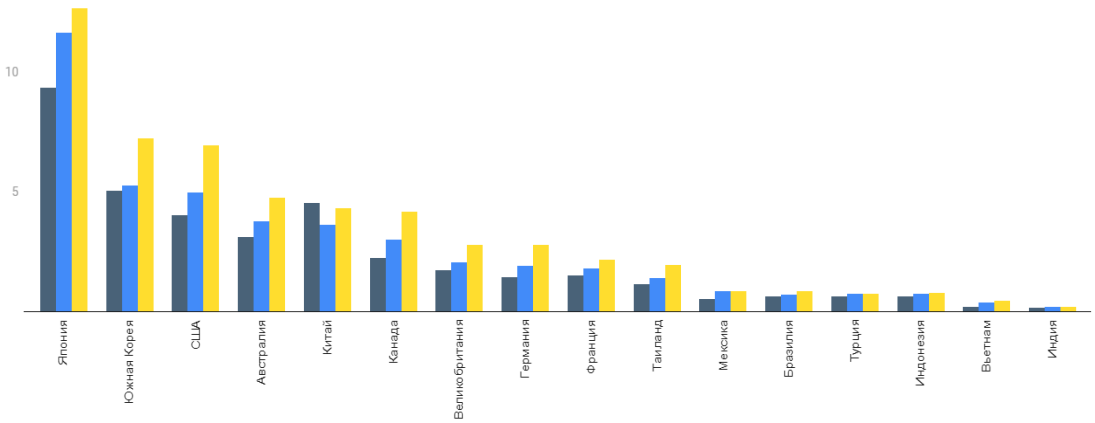

Рисунок 5 - Среднемесячные потребительские расходы на мобильные игры на одном устройстве на отдельных рынках, $

Что касаемо жанров на ПК, то самым популярным является action игры, и аналитики предсказывают дальнейший рост популярности. Этот жанр представляет собой игры в которых делается упор на эксплуатацию физических возможностей игрока, в том числе координации глаз и рук и скорости реакции. Но самыми прибыльными играми считаются в жанре RPG, так как в них больше возможностей для монетизации контента, но при этом требуются большие ресурсы для разработки и продвижения таких игр.

1.5 Создание концепта видеоигры

1.5.1 Основная механика

Вокруг основной механики строиться весь игровой процесс, и остальные аспекты игры должны дополнять и развивать. Игровая механика должна быть простой и понятной для того, чтобы она подошла для выбранного жанра action игр, а также соответствовать базовой схеме игрового цикла (Рисунок 6).

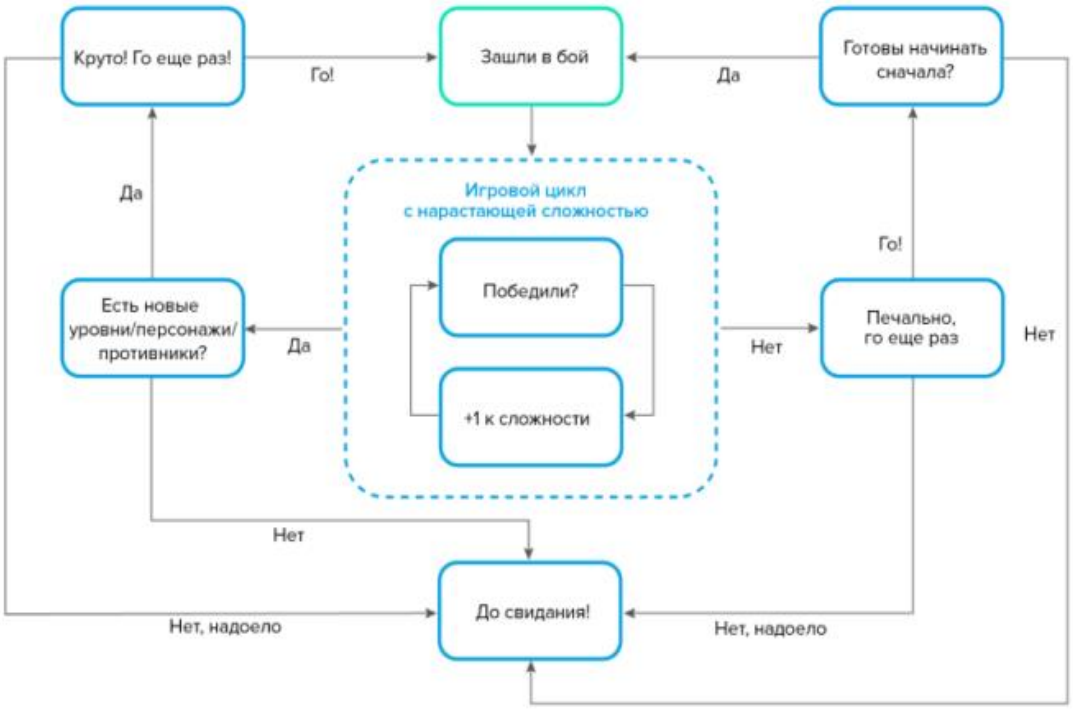

Рисунок 6 – Базовая схема игрового цикла

Было решено создать action игру в жанре Bullet Hell, основная механика которой будет заключаться в уклонении от снарядов противника игрока по полю.

Основная механика была придумана, далее следует ее развивать, чтобы игроку не наскучило. Развитие геймплея будет происходить благодаря добавлению противников с определенными типами атак, а также время от

времени появляющийся элитный противник с большим количеством жизней, и уронам.

1.5.2 Тестирование прототипа на реальной аудитории

Компанией были проведены тесты на небольшой аудитории разного возраста и пола (Рисунок 7), после создания еще нескольких тестовых уровней. Большинство отзывов было положительными, что дало зеленый свет для дальнейшей разработки на основе данной идеи.

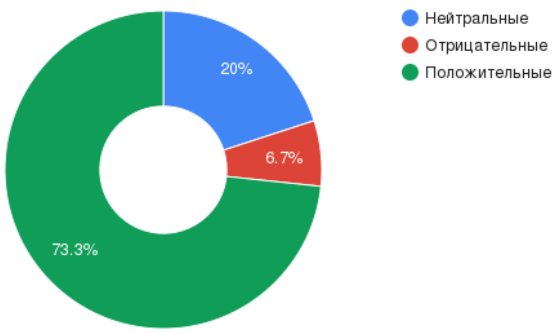

Рисунок 7 – Тестирование на аудитории

1.6 Анализ схожих видеоигр

Игры, в которых основная механика была бы в точности схожа с нашей, не были найдены. Поэтому было решено выполнить поиск успешных игр в жанре Action, Bullet Hell.

1.6.1.1 Игра «Enter the Gungen»

Игра Bullet Hell «ETG» в которой основным геймплеем является продвижение персонажа по локации до попадания к боссу уровня, и последующему переходу на следующий уровень (Рисунок 8).

Рисунок 8 – Игра «ETG»

По показателям игра является относительно успешной на площадке Steem: более 3 миллионов скачиваний и 253 тысячи отзывов со средней оценкой 4,7 из 5.

1.6.1.2 Игра «Cuphead»

Данная игра является Bullet Hell игрой, в которой главная задача является пройти всех боссов (Рисунок 9).

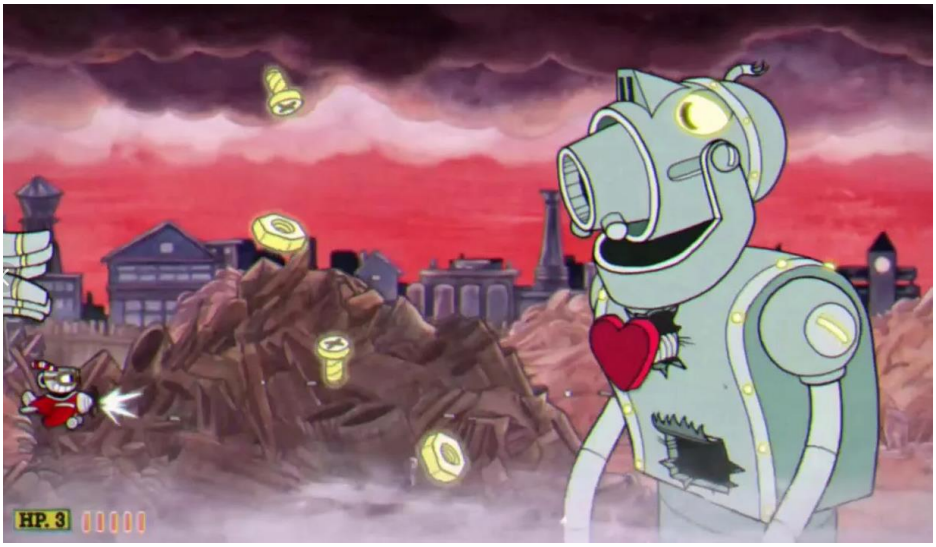

Рисунок 9 – Игра «Cuphead»

У игры неплохие показатели в Steem: 4 миллион скачиваний и оценка 4,6 из 5 на основе 138 тысяч отзывов.

1.8 Написание дизайн-документа

1.8.1 Жанр

Жанр видеоигры – это совокупность схожих критериев, которая объединяет определенную группу игр. Жанров большое множество, и они постоянно изменяются и дополняются.

Вывод по приведённой аналитике рынка такой, что видеоигра будет разрабатываться в Action жанре с поджанром Bullet Hell.

Жанр Action игр подразумевает собой игры в которых делается упор на эксплуатацию физических возможностей игрока, в том числе координации глаз и рук и скорости реакции.

1.8.2 Стилистика

Стилистика – это стиль, в котором будут выполнены элементы дизайна нашего проекта.

Взяв в пример найденные схожие видеоигры, придем к выводу, что можно использовать изометрический вид камеры.

Для всех элементов будут использоваться базовые формы. Это даст упрощение в стиле, которое увеличит концентрацию на деталях и уменьшит требования к ресурсам разработки.

Элементы UI должны быть выполнены в стиле минимализма, соответствовать цветовой гамме уровня и быть настолько интуитивными, чтобы пользователь мог, взглянув на элемент, понять его функцию без какого-либо текста.

1.8.3 Целевая аудитория

Целевая аудитория может делиться по разным признакам, но наша цель охватить максимально большое количество потенциальных игроков. Основываясь на выбранном жанре и анализе рынка, можно понять, что Action игры обязаны свой популярностью драйвовым и захватывающим геймплеем, ведь чем больше драйва, тем большей аудитории это понравиться.

Также пришла идея сделать максимально интуитивный интерфейс и геймплей для того, чтобы убрать текст из игры. Такой шаг даст возможность не тратить ресурсы на локализацию и дает возможность охватить еще большее количество игроков.

1.8.4 Механики

Механика – это набор каких-то условий и действий в совокупности, создающий часть интерактивного взаимодействия игрока с игрой.

Наша основная механика из концепта останется неизменной и будем устраивать новые уже вокруг нее.

Часто используется пилообразное наращивание сложности в играх (Рисунок 10). Но так как у нас головоломка, то тут подход будет немного отличаться, и сложность будет наращиваться без спадов.

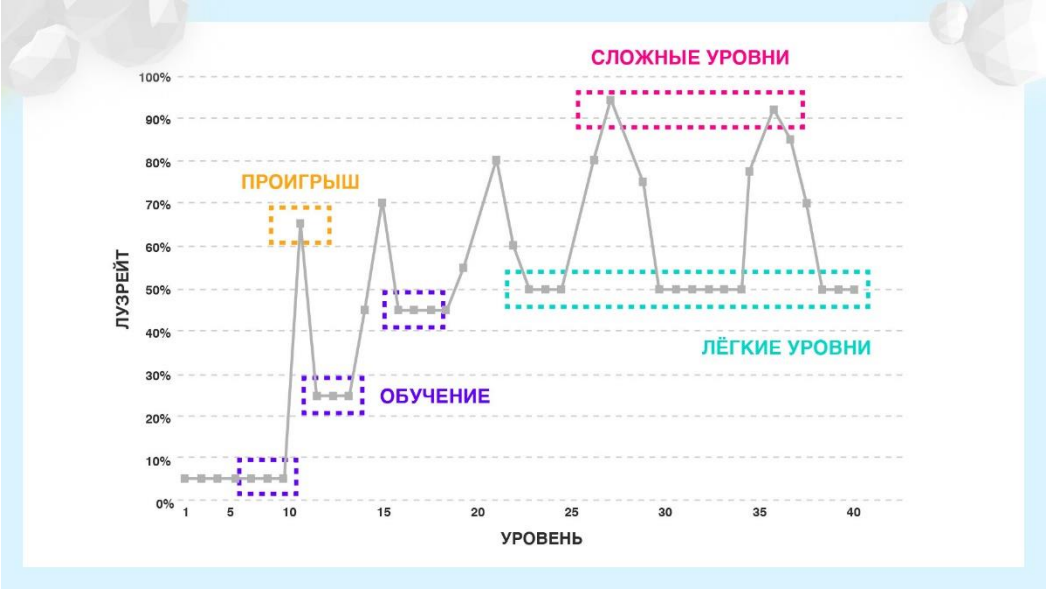

Рисунок 10 – Кривая сложности

1.8 Выбор среды разработки

Unity и Unreal Engine - это два наиболее популярных движка для создания игр. Оба инструмента обладают достаточно мощными и удобными инструментами для воплощения творческих идей в игровых проектах. Unity/Unreal Engine позволяет создавать игры для множества платформ,

включая ПК, мобильные устройства, консоли и виртуальную реальность. Каждый из этих двух движков имеет свои уникальные особенности и настройки, которые можно настроить под конкретный проект. Однако, перед выбором движка для создания игры, необходимо учитывать не только функциональность и удобство инструментов, но и навыки разработчиков, бюджет проекта и другие факторы.

Опираясь на тему “Разработка игрового приложения на движке «Unity/Unreal Engine», сравним эти два инструмента для создания игры, чтобы понять какой лучше подойдет именно для нашего проекта.

1.8.1 Unreal Engine

Unreal Engine - это один из самых мощных и популярных игровых движков на рынке, который по праву заслужил звание одного из лучших. Это многопоточный и многоплатформенный инструментарий создания игр, который позволяет разработчикам создавать проекты любой сложности и масштаба, от простых 2D-игр до сложнейших 3D-приключений.

Одной из особенностей Unreal Engine является его мощный и автоматический ландшафтный редактор. Он позволяет создавать уникальные и реалистичные пейзажи, природные объекты и другие элементы окружения с помощью встроенных инструментов, которые значительно облегчают процесс разработки. Кроме того, движок имеет большую библиотеку готовых материалов и текстур, которые можно применять к любым моделям.

Среди прочих функциональных особенностей Unreal Engine можно выделить мощную систему физики, которая отвечает за реалистичное поведение объектов в игре. Она позволяет создавать уникальные игровые механики и взаимодействия на основе физических законов, что может значительно улучшить игровой процесс.

Разработчики также могут использовать встроенный Blueprint Visual Scripting для создания игровых логик и сценариев. Это интуитивно понятная и мощная система визуального программирования, тесно интегрированная в Unreal Engine, что упрощает и ускоряет процесс разработки.

Unreal Engine имеет поддержку виртуальной реальности (VR) и аугментированной реальности (AR), что делает его идеальным инструментом для создания игр для VR- и AR- устройств. Более того, движок является открытым и доступен для настройки и модификации, что дает разработчикам свободу в создании уникальных проектов для широкой аудитории игроков.

Наконец, Unreal Engine интегрируется с многими современными платформами, такими как PlayStation, Xbox, Nintendo Switch, iOS, Android и PC, что позволяет разработчикам создавать игры для разных устройств и платформ с помощью одного инструментария. В сочетании со всеми вышеперечисленными функциями, это делает Unreal Engine лучшим выбором для создания игр на любой платформе.

Движок пользуется популярностью у больших разработчиков и применялся в таких играх как:

1) Valorant;
2) Mortal Kombat X;
3) Dishonored;
4) Серия игр Mass Effect;
5) It takes two.

Достаточно большой список успешных игр, большинство из которых являются проектами уровней AAA и AA, что говорит о удобстве использования в большой разработке и возможностях в создании передовой графики.

Является бесплатным с 2015 года, но требует уплаты роялти в размере 5 процентов с общемирового дохода с дополнительными условиями при доходе с игры от 1 миллиона долларов.

1.8.2 Unity

Unity - это многофункциональный игровой движок, который позволяет создавать игры для широкой аудитории, от мобильных устройств до настольных компьютеров и консолей. Он отличается от других игровых движков своей универсальностью, гибкостью и доступностью для разработчиков с любым опытом.

Особенностью Unity является его возможность создавать игры на различных платформах с единым кодом, что упрощает жизнь разработчикам и позволяет сократить время создания игры. Это особенно полезно при создании игр для мобильных устройств, где требуется быстрая разработка и высокое качество графики.

Unity также имеет интуитивно понятный интерфейс и богатый набор инструментов для создания игровых механик и оформления оформления. Среди них - встроенные обработчики физики, анимационные функции и различные визуальные эффекты. Это позволяет разработчикам сосредоточиться на создании интересного геймплея, а не на написании сложного кода.

В Unity также есть большое сообщество разработчиков, которое помогает новичкам и опытным разработчикам в любых вопросах. Сообщество также создает множество бесплатных ассетов, которые могут быть использованы в играх, что экономит время и средства на разработку.

Кроме того, Unity имеет возможность интеграции со сторонними средствами разработки, такими как Visual Studio, Blender и другими, что упрощает процесс разработки и повышает качество готового продукта.

В целом, Unity является мощным и универсальным инструментом для создания игр, который может быть использован для разработки широкого спектра игр, от 2D игр до сложных 3D проектов.

Примеры успешных видеоигр на Unity:

1) Genshin Impact;
2) Hollow Knight;
3) Cuphead;
4) Call of Duty: Mobile;
5) Standoff 2.

Судя по примеру выше, в основном все успешные инди хиты и популярные мобильные игры разрабатывались на данном игровом движке.

Лицензия на использование движка Unity существует в четырех вариантах (Таблица 1), но также есть бесплатный вариант, как у конкурента, при определенном доходе.

Таблица 1 – лицензии Unity

Тип лицензии	Доход компании в год, доллар	Доступ к исходному коду	Цена, доллар
Personal	До 100 тысяч	нет	Бесплатно
Plus	До 200 тысяч	нет	399 в год или 40 ежемесячно
Pro	Неограничен	нет	1800 в год или 150 ежемесячно
Enterprise	Неограничен	нет	200 в месяц

1.8.3 Сравнение Unity и Unreal Engine

Перед тем, как сравнить посмотрим на те факторы, которые нам будут нужны при разработке опираясь на наш дизайн документ:

Поддержка разработки для мобильных платформ Android и IOS

Полная поддержка 2д разработки

Высокоуровневый язык программирования

А теперь взглянем на присутствие данных факторов в движках (Таблица 2).

Таблица 2 – сравнение Unity и Unreal Engine

Фактор	Unreal	Unity
Поддержка мобильной разработки Android и IOS	Поддерживает	Поддерживает
Полная поддержка 2д	Неполная поддержка 2D. Больше заточен под разработку в 3D, а 2D является бонусом	Полная поддержка 2D и большое количество инструментов для этого
Язык программирования	C++ либо скриптинг.	C#, JavaScript и скриптинг.

По результатам сравнения для нашей видеоигры больше подойдет Unity. У данного движка присутствует высокоуровневый язык программирования C#, что ускорит разработку, а также большее количество 2D инструментов, что облегчит работу.

2. Создание игры

2.1 Дизайн уровня

Перед созданием игры необходимо создать схематичное построение уровня. Для этого было решено использовать встроенную в операционную систему Windows программу Paint. Концепт уровня можно увидеть на рисунке 10.

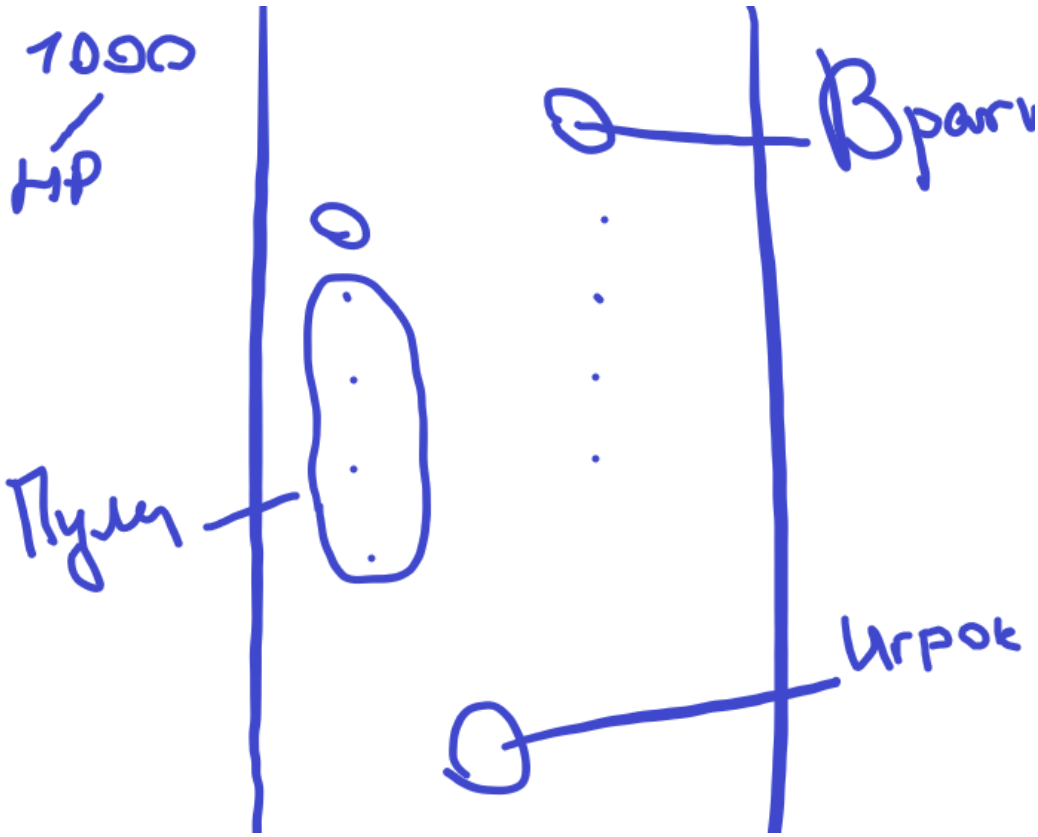

Рисунок 10 – Схематичное построение уровня

На схеме, представленной выше, изображен уровень, состоящий из туннеля по которому игрок может свободно передвигаться. В туннеле время от времени появляться враги которые стреляют в игрока, цель игрока победить врагов потеряв как можно меньше здоровья.

2.2 Создание уровня

Создание уровня осуществлялось при помощи встроенного в движок Unity редактора двухмерных сцен. С помощью базовых элементов, вроде квадрата, была создана сцена, максимально приближенная к изначально нарисованному схематичному варианту. Пока что сцена не содержит в себе декоративных элементов или элементов, с которыми игроку непосредственно предстоит взаимодействовать, однако геометрия, на базе которой можно будет расставлять все остальные элементы, полностью готова. Как уровень выглядит со стороны в редакторе двухмерных сцен, можно увидеть на рисунке 11.

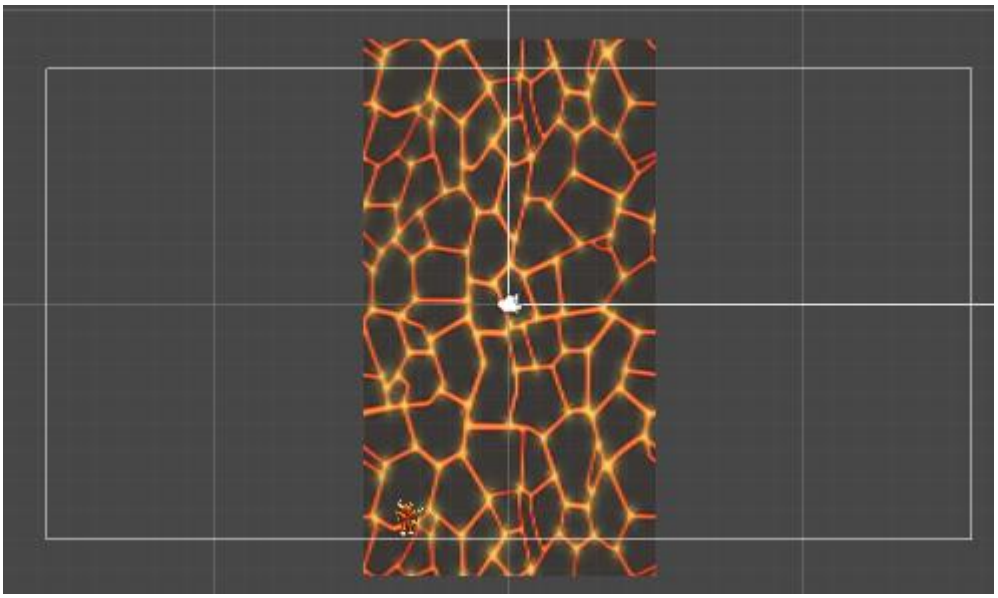

Рисунок 11 – Внешнее представление игрового уровня

2.3 Элементы уровня

2.3.1 Игрок

2.3.1.1 Передвижение

Передвижение игрока осуществляется при помощи клавиатуры и мыши. Клавишами на клавиатуре игрок перемещает персонажа по уровню в двух осях координат. Также игрой предусмотрена возможность совершать атаку, все это делается по нажатию соответствующих клавиш. Для каждого из этих действий необходимо создать соответствующие скрипты, считывающие нажатия клавиш и конвертирующие это в действия персонажа на экране.

Для передвижения персонажа по уровню было решено использовать стандартное для игр от первого лица сочетание клавиш – WASD, отвечающее за передвижение вперед, влево, назад и вправо соответственно. Эта комбинация прописывается в скрипте относившихся к игроку, а также подключаеться к элементу за который будет происходить геймплей что отражено на рисунке 11.

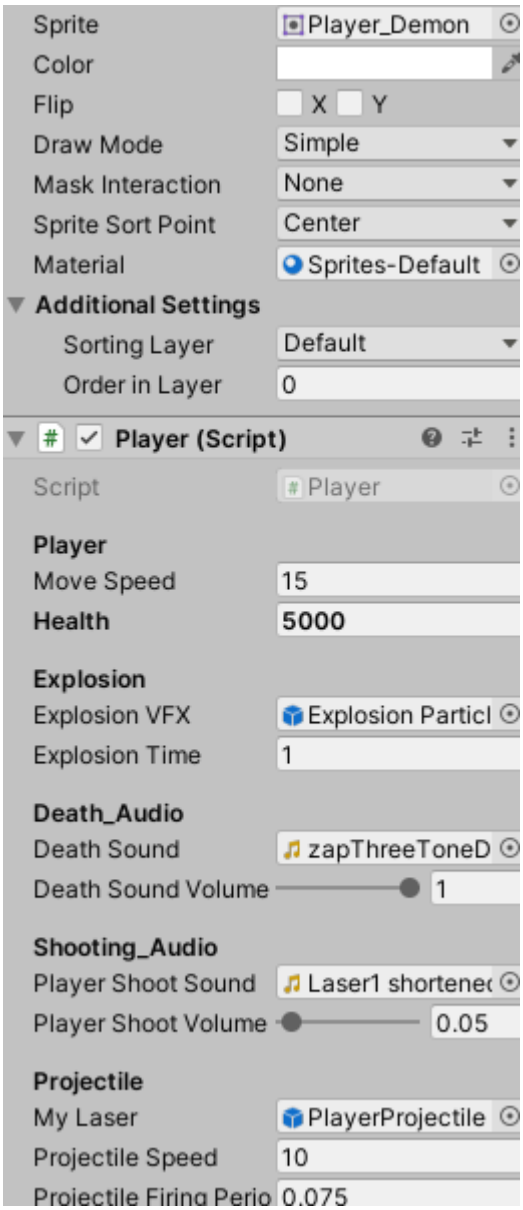

Рисунок 11 – Привязка элемента к скрипту

Для считывания команд ввода передвижения используется два события, которые были назначены в скрипте Player: для считывания команд

ввода передвижения используется два события. Для считывания команд ввода передвижения используется два события.

Полную реализацию движения персонажа можно увидеть на рисунке 12.

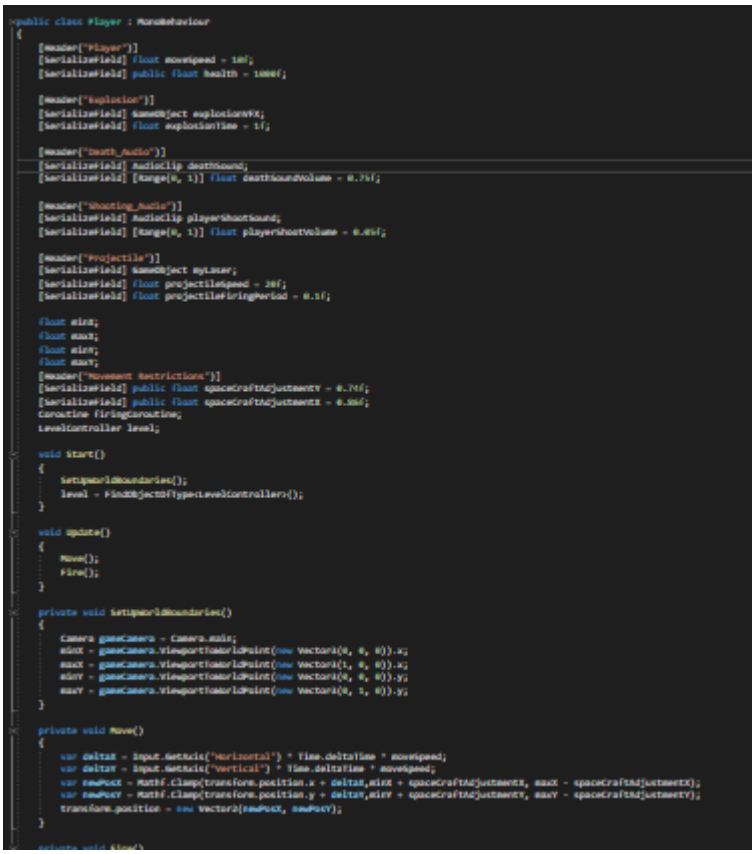

Рисунок 12 – Реализация передвижения персонажа игры

2.3.1.2 Выстрелы

Так как основная суть игрового проекта, это стрельба, игроку необходимо дать инструмент для стрельбы в противников, в данном случае это магия. Для создания такого функционала необходимо, во-первых, назначить клавишу для стрельбы и, во-вторых, создать логику, которая будет считывать нажатие клавиши.

Примерно так можно описать алгоритм создания стрельбы в Unity:

1. Создание снарядов. Создайте объект снаряда и назначьте ему меш и материал, который будет применятся к шару снаряда. Добавьте компонент Rigidbody, чтобы скрипт контроля движения мог использовать его свойства, и Collider, чтобы обеспечить корректное взаимодействие с твердыми объектами,
2. Определение событий Input для выстрела. Добавьте код, который будет считывать нажатие клавиши или касание экрана для выполнения выстрела что демонстрирует рисунок 13.

```
private void Fire()
{
    if(Input.GetButtonDown("Fire1"))
    {
        firingCoroutine = StartCoroutine(FireContinously());
    }
    if(Input.GetButtonUp("Fire1"))
    {
        StopCoroutine(firingCoroutine);
    }
}
```

Рисунок 13 – Логика компонента Fire

3. Создание и настройка скрипта обработчика выстрела. Создайте скрипт на объекте пушки или персонажа, который будет контролировать выстрелы. В этом скрипте нужно определить, как выстрелы вызываются, и настроить параметры, такие как скорость снаряда, соответствующую точку появления и т.д. на рисунке 14.

```
IEnumerator FireContinously()
{
    while (true)
    {
        GameObject Laser = Instantiate(myLaser, transform.position, Quaternion.identity) as GameObject;
        Laser.GetComponent<Rigidbody2D>().velocity = new Vector2(0, projectileSpeed);
        AudioSource.PlayClipAtPoint(playerShootSound, Camera.main.transform.position, playerShootVolume);
        yield return new WaitForSeconds(projectileFiringPeriod);
    }
}
```

Рисунок 14 – Логика выстрела

3. Расчет и обработка коллизий снарядов с объектами в игре. Добавьте код, который будет обрабатывать столкновения снарядов с твердыми объектами, другими игровыми объектами или второстепенными объектами,

например, защитными щитами. Можно использовать обработчики OnTriggerEnter или OnCollisionEnter в скрипте снаряда

```
void OnTriggerEnter2D(Collider2D other)
{
    // если столкнулись с объектом врага, то наносим ему урон и уничтожаем снаряд
    if (other.CompareTag("Enemy"))
    {
        Enemy enemy = other.GetComponent<Enemy>();
        if (enemy != null)
        {
            enemy.TakeDamage(damage);
        }
        Destroy(gameObject);
    }
    // если столкнулись с объектом препятствия, то уничтожаем снаряд
    if (other.CompareTag("Obstacle"))
    {
        Destroy(gameObject);
    }
}
```

Рисунок 15 – Пример кода

После того, как вся логика была создана, выстрелы в игре выглядит так, как показано на рисунке 16.

Рисунок 16 – Выстрелы в игре

2.3.1.3 Ссколькжение и разгон при движении

Скольжение и разгон являются важными аспектами механики передвижения в 2D игре Unity.

Скольжение — это способность персонажа сохранять свой импульс движения, даже когда он больше не нажимает на клавиши управления. Это часто используется в платформере, чтобы персонаж продолжал двигаться после того, как он прыгнул или начал бег.

Для этого вам нужно добавить Rigidbody 2D компонент к персонажу и настроить его свойства:

- Gravity Scale — это значение контролирует, как быстро персонаж падает вниз. Меньшие значения замедляют падение.

- Linear Drag — это значение определяет, как быстро персонаж останавливается при движении. Меньшие значения продолжают движение на большее расстояние.

Разгон — это способность персонажа начать движение быстро и плавно по мере увеличения скорости. Для лучшего результата необходимо использовать анимации и код для ускорения и замедления персонажа.

Для создания плавного разгона вам нужно настроить скрипт управления персонажем, чтобы он инкрементировал скорость до максимального значения за определенный промежуток времени. Пример кода:

```
[SerializeField] private float accelerationSpeed = 5f;
[SerializeField] private float maxSpeed = 10f;
private float currentSpeed;

void Update() {
  if (Input.GetKeyDown(KeyCode.RightArrow)) {
    currentSpeed = Mathf.MoveTowards(currentSpeed, maxSpeed, accelerationSpeed * Time.deltaTime);

    // Move character
    transform.position += new Vector3(currentSpeed * Time.deltaTime, 0f, 0f);
  }
}
```

Рисунок 17 – Пример кода скольжения

В этом примере мы используем функцию Mathf.MoveTowards для плавного ускорения, а затем передвигаем персонажа по оси X в зависимости от текущей скорости. Это может быть изменено в соответствии с вашими потребностями в разгоне и скольжении.

2.3.1.4 Здоровье персонажа

Здоровье персонажа должно отображаться непосредственно на экране все время, пока игрок выживает, в виде цифры. В начале игры эта цифра будет установлена на 5000, однако, когда игроку будет наноситься урон от

противника, то цифра будет постепенно уменьшаться с каждым ударом противника. Когда значение шкалы упадет до нуля, игра будет окончена.

Для отображения шкалы здоровья необходимо Создание логики здоровья в виде цифры в Unity 2D может быть выполнено несколькими способами, но я опишу один из наиболее простых и популярных.

1. Создайте UI Text объект в графе иерархии: выберите Canvas, и в контекстном меню выберите UI -> Text.

2. Расположите его на сцене, выберите шрифт, цвет и размер текста.

4. Добавьте скрипт, который будет отображать текущее значение здоровья на текстовом поле. Создайте новый C# скрипт в папке Scripts в Assets.

5. Добавьте скрипт HealthDisplay к UI Text объекту.

6. Присвойте переменные healthImage и healthText в редакторе Unity, и затем переместите объект в игровой объект Player.

7. Измените значение здоровья в скрипте, чтобы текст был обновлен, например, при контакте персонажа с враждебными объектами (рисунок 18)

Рисунок 18 – Отображение показателя здоровья на экране игры

2.3.1.5 Очки персонажа

Для фиксации на сколько далеко продвинулся игрок, было принято решение создания счетчика очков, для этого был создан скрипт почти полностью повторяющий счётчик ХП с одним исключением, что он считал очки за каждого побежденного врага

2.3.1.6 Регулировка громкости

Целью кнопки регулировки громкости в игровом проекте является управление громкостью звуковых эффектов и музыки в игре. Кнопка может

быть настроена на изменение громкости звуков отдельно или в целом, то есть на изменение громкости звуковых эффектов и музыки одновременно. Это позволяет игрокам настраивать звуковое сопровождение игры под свои личные предпочтения и потребности. Кроме того, кнопка регулировки громкости может использоваться в качестве средства контроля уровня шума, что особенно важно при игре в общественных местах или ночью, в данном проекте для экономии места и выдерживания стиля старой аркадной игры было решено сделать регулировку громкости, не ползунком, а кнопкой с несколькими режимами громкости, сто процентов, пятьдесят процентов, двадцать пять процентов и ноль процентов, для упрощения понимания громкости было решено добавить визуальный индикатор уровня громкости в виде динамика покрытого огнем, чем громче уровень звука тем больше пламени вокруг динамика рисунок 19.

Рисунок 19 – Уровень громкости

2.3.2 Игровое поле (уровень)

2.3.2.1 Задний фон

Задний фон было решено сделать с анимацией для иллюзии продвижения по уровню

В Unity можно создать анимированный задний фон с помощью компонента Render Texture и материала с анимированной текстурой.

1. Создайте новый объект (GameObject) и добавьте ему компонент Camera.
2. Задайте параметры камеры (Field of View, Clipping Planes и т. д.) так, чтобы объекты на сцене отображались по вашим требованиям.

3. Создайте новый материал (Material) и добавьте ему текстуру (Texture). Материал должен иметь Shader с поддержкой анимации текстуры, например, Unlit/Texture.

4. Добавьте компонент Render Texture к камере и настройте его параметры (Resolution, Depth, Anti-aliasing и т. д.).

5. Настройте материал так, чтобы он использовал Render Texture в качестве источника текстуры.

6. Добавьте компонент Animation к материалу и создайте анимацию текстуры.

7. Присвойте материал объекту Render Quad, чтобы отобразить анимированный задний фон.

2.3.2.2 Противники

Для создания противника нам нужно 3 скрипта первый будет отвечать за путь противника, как он будет двигаться по полю, и второй, который будет отвечать за хп противника, количество очков за его убийство и интервал атак, и третий который будет отвечать за количество урона наносимый противником, в данном игровом проекте представлены следующие игровые противники на рисунках 20-26. Характеристики соответственно приведены в таблицах 3-9.

Рисунок 20 – Виверна

Таблица 3 – Характеристики Виверны

HP	Урон	Очки за убийство
300	50	50

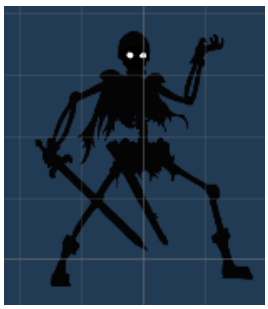

Рисунок 21 – Егор

Таблица 4 – Характеристики Егора

HP	Урон	Очки за убийство
7000	10000	1000

Рисунок 22 – Бафомет

Таблица 5 – Характеристики Бафомет

HP	Урон	Очки за убийство
300	50	50

Рисунок 23 – Химера

Таблица 6 – Характеристики Химера

HP	Урон	Очки за убийство
500	100	200

Рисунок 24 – Краб

Таблица 7 – Характеристики Краб

HP	Урон	Очки за убийство
800	200	300

Рисунок 25 – Демон

Таблица 8 – Характеристики Демон

HP	Урон	Очки за убийство
500	80	150

Рисунок 26 – Ионита

Таблица 9 – Характеристики Ионита

HP	Урон	Очки за убийство
10000	10000	2500

2.4 Меню

2.4.1 Главное меню

Главное меню, как и во многих других играх, будет содержать в себе две основных клавиши: начать игру и выйти из игры. Кнопка начала будет запускать созданный уровень, а кнопка выйти – закрывать игру.

Чтоб это сделать надо:

1. Создайте новую сцену

2. Создайте пустой объект (GameObject) на сцене и назовите его "MainMenu" (или любое другое удобное имя).

3. Добавьте компонент Canvas к объекту MainMenu. Этот компонент говорит Unity, что этот объект является контейнером для UI-элементов.

4. Настройте параметры Canvas

5. Добавьте UI-элементы на Canvas с помощью кнопок, текста, изображений и т. д. Кнопки должны быть связаны с соответствующими функциями, которые будут вызываться при нажатии на них.

6. При помощи компонента EventSystem добавьте Event System на сцену – это обязательно для выполнения управления пользовательским вводом, включая навигацию с помощью клавиш (переход от кнопки к кнопке и т.п.) (рисунок 27)

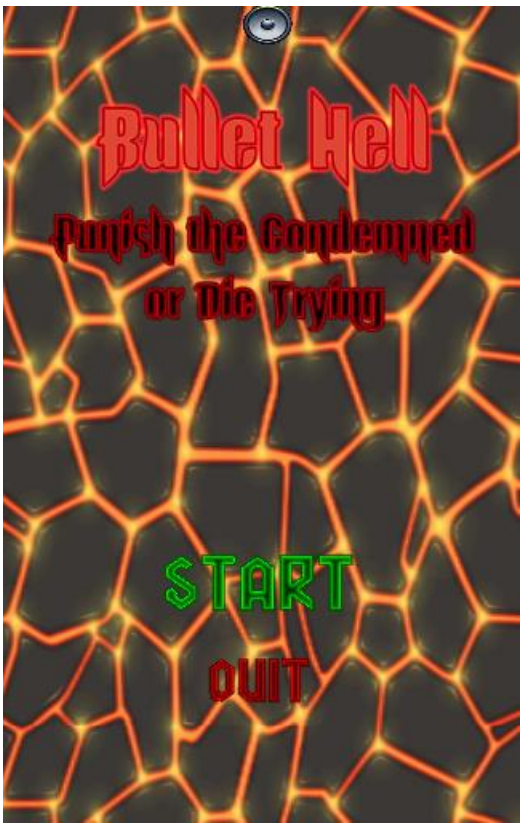

Рисунок 27 – Меню старта

2.4.2 Меню конца игры

Меню конца игры будет появляться в тот момент, когда противник нанесет достаточный урон игроку, и показатель здоровья последнего упадет до нуля. Само меню будет содержать в себе две кнопки: Restart и Main Menu, отвечающие за перезапуск уровня и выход в главное меню соответственно.

Как и в случае с созданием главного меню игры, для создания меню конца игры мы делаем тоже самое что и в прошлом случае, а также прописываем скрипт который выводит это меню при потере всего здоровья (Рисунок 28)

Рисунок 28 – Меню конца игры

3. Экономическая часть

3.1 Расчет затрат на создание

В соответствии с ГОСТ 34.601-90 «Автоматизированные системы.

Стадии создания» сформирован план работ.

Разработка системы разделено на отдельные процессы:

– Предпроектное обследование. В данном процессе происходит исследование и анализ предметной области, проблематики, а также разработка ТЗ.

– Эскизное проектирование. В данном процессе происходит теоретическое проектирование структуры системы, её интерфейс и разрабатывается проектная документация.

– Технорабочий проект. В данном процессе происходит реализация системы в соответствии с разработанной структурой в предыдущем шаге, также происходит разработка технической документации.

– Ввод в эксплуатацию. В данном процессе происходит тестирование готовой системы и ее выпуск.

Отчет о затратах ресурсов на создание игрового персонажа представлен на рисунке 29.

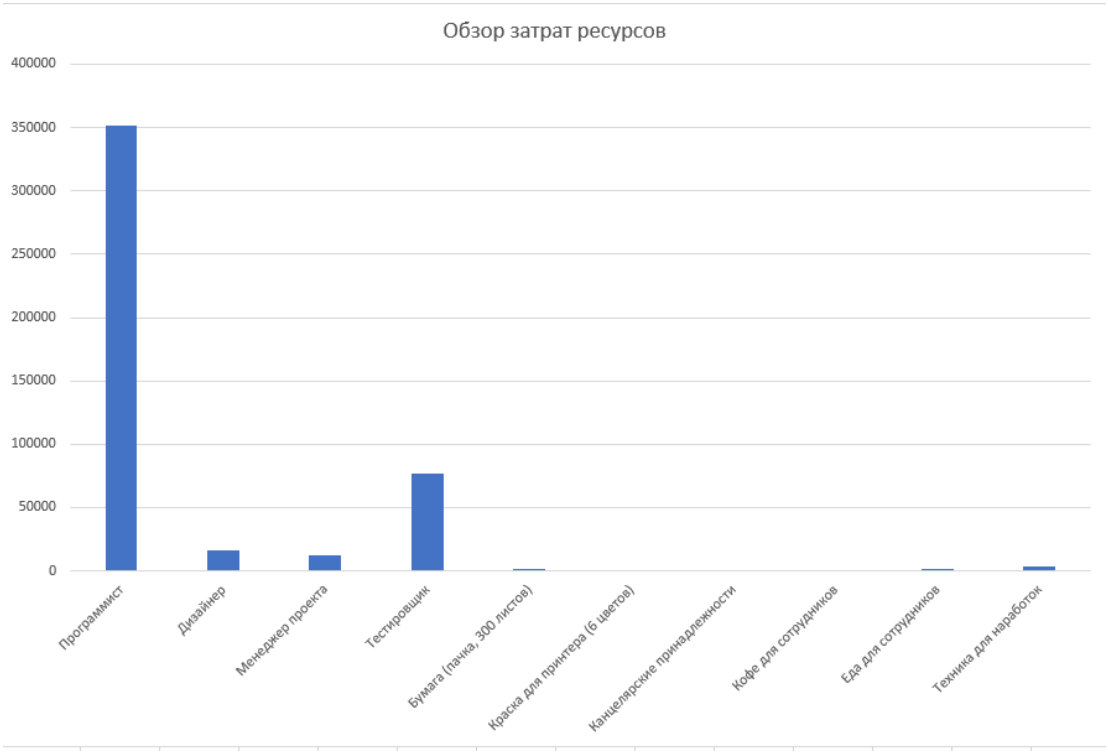

Рисунок 29 – График затрат ресурсов

3.2 Затраты на создание

3.2.1 Материальные затраты

Материальные ресурсы, необходимые для разработки технического задания с учетом черновых работ, представлены в таблице 10.

Таблица 10 – Затраты на разработку ТЗ

№ п/п	Наименование материального ресурса	Количество, шт.	Цена, руб.	Сумма, руб.
1	Бумага (пачка, 300 листов)	1	1500,00	1500,00
2	Краска для принтера (6 цветов)	1	1000,00	1000,00
3	Канцелярские принадлежности	2	500,00	1000,00
4	Кофе для сотрудников	2	340	680
5	Еда для сотрудников	24	50	1200

6	Техника для наработок	3	1080,00	3240,00
Итого:				8 620,00

Итого, общие затраты – 8 620,00 рублей

3.2.2 Заработная плата

Основная заработная плата при выполнении работ по разработке системы включает зарплату всех сотрудников, принимающих непосредственное участие в разработке. Основная заработная плата $З_{осн}$ рассчитывается по следующей формуле:

$$З_{осн} = \sum_{j=1}^{n} З_{средj},$$

где $З_{сред}$ – зарплата j-го сотрудника, руб., а n – количество сотрудников, принимающих непосредственное участие в разработке игрового проекта.

В таблице 11 приведены результаты вычисления ЗП сотрудников.

Таблица 11 – Заработная плата работников

№ п/п	Наименование трудового ресурса	Почасовая ставка, руб./ч	Трудозатраты, ч	Суммарная заработная плата, руб.
1	Программист	1000,00	352 ч	352 000,00
2	Дизайнер	400,00	40 ч	16 000,00
3	Менеджер проекта	300,00	40 ч	12 000,00
4	Тестировщик	300,00	256 ч	76 800,00
	Итого:			456 800,00

Общая затрата на выплату составляет 456 800,00 руб.

3.2.3 Затраты на программное обеспечение

Разработка персонажа производится при помощи бесплатного и платного ПО. Используемое ПО и их стоимость на 05.05.2023 приведены в таблице 4 в соответствие с курсом (1 $ = 77,82 руб., 1 € = 85,73 руб.).

Таблица 12 - Используемое программное обеспечение и его стоимость

№ п/п	Наименование ПО	Цена одной копии	Количество копий	Суммарная стоимость, руб.
1	Windows 10 Professional	24000,00	1	24000,00
2	Microsoft Office 2019	10000,00	1	10000,00
3	Unreal Engine 4	0,00	1	0,00
Итого:				34 000,00

Таким образом, затраты на ПО составляет 34 000,00 руб.

3.2.4 Затраты на электроэнергию

Затраты на электроэнергию вычисляются по следующей формуле:

$З_{эн} = P_{пот} * T_{вр} * C_{квт.ч}$

$P_{пот}$ – потребляемая мощность. Так как работа проводилась на удаленной основе, этот показатель равен 0 кВт. $T_{вр}$ – фонд времени за период амортизации – 0 д. * 8 ч/д. = 0 ч. $C_{квт.ч}$ – стоимость одного киловатта энергии – 4,20 руб.

Таким образом, Зэн = 0 кВт * 0 ч * 4.20 руб. = 0,00 руб.

3.2.5 Накладные расходы

Накладные расходы составляют 10% от прямых затрат. В прямые затраты входят затраты по статьям 1-6. Таким образом, накладные расходы составляют:

(8620 + 456800 + 34000 + 0) * 0,1 = 499420 * 0,1 = 4994,2

Общая себестоимость разрабатываемого программного средства

составляет: 8620 + 456800 + 34000 + 0 + 4994,2 = 504414,2 руб.

Заключение

По итогам выпускной квалификационной работы был создан игровой проект в жанре Bullet Hell на базе Unity, который предлагает игроку бесконечно идти по уровню уничтожая врагов зарабатывая как можно больше очков. После анализа движка Unity, анализа рынка видеоигр в целом и видеоигр в жанре Bullet Hell в частности, был выбран вектор создания проекта.

При создании был использован визуальный редактор и стандартные ассеты движка Unity, которые помогли создать анимации уровня, хит боксы противников, игрока, и аттак классы с простой, но обширной логикой, которая позволяет игроку передвигаться по уровню с помощью клавиатуры, а также атаковать.

После анализа расходов на разработку, был сделан вывод, что изначальные задачи были выполнены, а цели достигнуты.

Список использованных источников

1. Нажми Reset. Как игровая индустрия рушит карьеры и дает второй шанс / Джейсон Шрейер – Бомбора, 2021. – 368 с.

2. Кровь, пот и пиксели: Обратная сторона индустрии видеоигр / Джейсон Шрейер. – Бомбора, 2022. – 368 с.

3. Resident Evil. Обитель зла игровой индустрии / Алекс Аниэл. – Бомбора, 2022. – 320 с.

4. Silent Hill. Навстречу ужасу. Игры и теория страха / Бернар Перрон. – Бомбора, 2022. – 288 с.

5. Повелители DOOM. Как два парня создали культовый шутер и раскачали индустрию видеоигр / Дэвид Кушнер. – Бомбора, 2022. – 496 с.

6. Кодзима – гений. История разработчика, перевернувшего индустрию видеоигр / Терри Вульф. – Бомбора, 2021. – 600 с.

7. Dark Souls: за гранью смерти. История создания Demon’s Souls, Dark Souls, Dark Souls II / Дамьен Мешери, Сильвен Ромье. – Бомбора, 2023. – 496 с.

8. Power Up! Как Япония вдохнула в игровую индустрию новую жизнь / Крис Колер. – Бомбора, 2023. – 560 с.

9. Оптимизация игр в Unity 5. Крис Дикинсон, 2017. -211 с.

10. Виртуальная реальность в Unity. Джонатан Линовес, 2016. -146 с.