Правительство Санкт-Петербурга

Комитет по науке и высшей школе

**Санкт-Петербургское государственное бюджетное**

**профессиональное образовательное учреждение**

**«Политехнический колледж городского хозяйства»**

**ДИПЛОМНЫЙ ПРОЕКТ**

Разработка компьютерной игры The Little Prince

Пояснительная записка

Листов 28

СОДЕРЖАНИЕ

[Введение 3](#_Toc199400092)

[1 Анализ предметной области 4](#_Toc199400093)

[1.1 Описание предметной области 4](#_Toc199400094)

[1.1 Обзор аналогов 5](#_Toc199400095)

[1.2 Требования к разрабатываемой ИС 8](#_Toc199400096)

[1.2.1 Функциональные требования 8](#_Toc199400097)

[1.2.2 Требования к интерфейсу 9](#_Toc199400098)

[1.3 Обоснование выбора стека технологий 9](#_Toc199400099)

[2 Проектирование 11](#_Toc199400100)

[2.1 Проектирование системы 11](#_Toc199400101)

[2.2 Определение группы пользователей 11](#_Toc199400102)

[2.3 Функциональное моделирование 12](#_Toc199400103)

[2.4 Разработка модели базы данных 13](#_Toc199400104)

[2.5 Проектирование интерфейсов 13](#_Toc199400105)

[3 Реализация 15](#_Toc199400106)

[3.1 Реализация основных функций 15](#_Toc199400107)

[3.2 Реализация интерфейсов 17](#_Toc199400108)

[3.3 Тестирование 18](#_Toc199400109)

[4 Руководство пользователя 20](#_Toc199400110)

[5 Мероприятия по информационной безопасности 22](#_Toc199400111)

[Заключение 24](#_Toc199400112)

[Список источников 25](#_Toc199400113)

[Приложение А 27](#_Toc199400114)

[Приложение Б 28](#_Toc199400115)

# **Введение**

Современная игровая индустрия — это не только один из самых быстроразвивающихся секторов IT, но и мощный двигатель развития технологий визуализации, взаимодействия с пользователем и программирования в реальном времени. Игры давно вышли за рамки простого развлечения, став средствами обучения, развития логики, реакции, внимания, а также важным инструментом демонстрации программных навыков. Особенно популярным жанром в игровой индустрии остается 2D-платформер — один из классических форматов, понятный широкой аудитории и позволяющий реализовать множество механик.

В рамках данной дипломной работы была поставлена цель — разработка интерактивного двухмерного платформера, основная задача которого — обеспечить увлекательный геймплей с элементами сражений, прохождением уровней от точки A до точки B, сбором бонусов и использованием временных улучшений (баффов). Проект ориентирован на аудиторию пользователей, заинтересованных в динамичном и понятном игровом процессе. Важно отметить, что подобные игры обладают высоким потенциалом вовлечения, а благодаря грамотному подходу к реализации — могут быть адоптированы на различные платформы, включая мобильные устройства.

Работа выполнена с использованием Unity — одного из самых мощных и популярных игровых движков, позволяющего эффективно реализовывать кроссплатформенные решения. В качестве языка программирования выбран C#, благодаря его поддержке объектно-ориентированного подхода, читаемости и хорошей интеграции с Unity.

Проект решает сразу несколько задач:

* Закрепление теоретических и практических знаний, полученных в процессе обучения;
* Разработка реального программного продукта с полным циклом — от проектирования до тестирования;
* Освоение современных подходов к построению архитектуры игр, оптимизации производительности, реализации визуальных и звуковых эффектов;
* Демонстрация навыков UI/UX-дизайна, работы с физикой объектов и визуализацией взаимодействий.

Кроме того, проект нацелен на повышение уровня компетенций в области информационной безопасности, архитектурной модульности и производственной разработки.

Разработанная игра не просто демонстрирует технические навыки, но и может стать основой для последующего расширения: создания более сложных уровней, введения системы прокачки персонажа, внедрения многопользовательского режима, а также коммерциализации проекта.

# **Анализ предметной области**

# **Описание предметной области**

Жанр 2D-платформеров занимает значительное место в индустрии видеоигр. Он зародился ещё в 80-х годах с таких проектов, как Super Mario Bros, и до сих пор остается актуальным благодаря простоте, доступности и высокой вариативности игрового процесса. Классический платформер основывается на управлении персонажем, которому предстоит пройти сквозь множество препятствий и врагов по горизонтально или вертикально прокручивающемуся уровню. Игроку необходимо точно рассчитывать движения, использовать временные улучшения (баффы), избегать ловушек, сражаться с врагами и достигать финальной точки уровня.

Предметной областью проекта является разработка интерактивной компьютерной игры, сочетающей элементы аркады, боевика и логического мышления, направленной на развитие реакций, внимательности и стратегического планирования действий. Игра реализуется в формате однопользовательского 2D-платформера, что позволяет минимизировать требования к оборудованию при сохранении интересного геймплея.

Проект ориентирован на демонстрацию как творческих, так и технических навыков разработчика. В основу игры положена архитектура, основанная на объектно-ориентированном программировании, а также системный подход к организации игрового цикла, взаимодействию объектов, пользовательскому интерфейсу и визуальному оформлению.

Целью игры является прохождение от начальной точки (A) до конечной (B), при этом игрок сталкивается с различными противниками, преодолевает препятствия и собирает усиления, временно улучшающие характеристики главного героя (например, увеличение скорости, силы, высоты прыжка и т. д.). Подобный подход обеспечивает динамичный, но сбалансированный игровой процесс.

Реализация проекта требует знания базовых и продвинутых аспектов Unity: управление сценами, работа с компонентами, а также оптимизация производительности для корректной работы на широком спектре компьютеров.

# **Обзор аналогов**

#### Перед началом разработки игры «The Little Prince» был проведён анализ существующих игр в жанре 2D-платформеров, а также проектов, сочетающих элементы экшена, головоломок и приключения. Цель анализа — выявить сильные и слабые стороны аналогов, а также определить уникальные особенности, которые можно использовать или адаптировать в собственной игре.

Особое внимание уделялось следующим аспектам:

* визуальной подаче и атмосферности;
* качеству управления персонажем;
* оригинальности игровых механик;
* балансу между боевой системой и платформингом;
* архитектуре уровней и сценарию.

**Rogue Legacy 2**

Платформер с наследственной системой героев, процедурной генерацией и постоянным прогрессом.

* **Достоинства**: высокая пере проходимость, уникальные свойства каждого героя, динамичные бои с магией и ближним боем.
* **Недостатки**: случайность уровня может сбивать с толку, перегрузка эффектами для новичков.



Рисунок 1- Игра Rogue Legacy 2

#### **Celeste**

Платформер с глубоким эмоциональным сюжетом и оригинальной механикой «взлета/рывка». Игра делает упор на точность управления и атмосферу. Несмотря на минимализм графики, проект получил признание благодаря глубине игрового процесса.

* **Достоинства**: уникальная механика, прекрасный звук, адаптивная сложность.
* **Недостатки**: отсутствие боевой системы, больше подходит для "прыжковых" испытаний, чем для экшена.



Рисунок 2 - Игра Celeste

#### **Hollow Knight**

Видеоигра с упором на исследование, сражения и прокачку персонажа. Отличается продуманным дизайном уровней и разнообразием врагов. Несмотря на то, что игра ближе к жанру метроидвания, в ней много механик, применимых в платформерах.

* **Достоинства**: богатый контент, стильный арт, боевая система.
* **Недостатки**: сложная навигация, высокая продолжительность, не подходит для коротких игровых сессий.



Рисунок 3 - Игра Hollow Knight

#### **Limbo / Inside**

Атмосферные платформеры с упором на визуальное повествование и минималистичный дизайн. Меньше боевых элементов, но больше интерактивных объектов и головоломок.

* **Достоинства**: уникальная атмосфера, кинематографичность.
* **Недостатки**: ограниченное взаимодействие, линейность.



Рисунок 4 - Игра Inside

#### **Вывод по результатам анализа аналогов:**

Разрабатываемый проект сочетает **удобство управления и простоту**, как у Celeste, с базовой **боевой системой**, аналогичной Hollow Knight. В отличие от Super Meat Boy, игра будет иметь умеренный уровень сложности и систему временных улучшений (баффов), которые повысят интерес к повторному прохождению. Также реализуется базовая адаптивность под производительность устройств.

Разрабатываемый проект The Little Prince вобрал в себя лучшие черты аналогов:

* атмосферность и бой (Hollow Knight);
* визуальную выразительность (Ori);
* точность управления и платформинг (Celeste);
* нарративные и визуальные решения (Limbo/Inside);
* динамические улучшения и боевые эффекты (Rogue Legacy 2).

При этом игра отличается оригинальностью благодаря:

* интеграции магических механик (эхо, порталы, призрачные платформы);
* системе боевых усилений(баффов);
* уникальной визуальной подаче;
* оригинальному сценарию со спасением принцессы и магическим миром.

# **Требования к разрабатываемой ИС**

Разрабатываемое приложение представляет собой одиночную 2D-игру в жанре платформер, ориентированную на настольные операционные системы Windows. Основное назначение — обеспечение плавного и увлекательного игрового процесса, сочетающего управление персонажем, взаимодействие с объектами, сражения с врагами и сбор временных усилений (баффов).

# **Функциональные требования**

Таблица 1 – Функции игры

|  |  |  |
| --- | --- | --- |
| **№** | **Функция** | **Описание** |
| 1 | Управление персонажем | Поддержка движения влево/вправо, прыжков, атак |
| 2 | Взаимодействие с окружением | Столкновение с врагами, предметами, ловушками |
| 3 | Реализация усилений | Временные усиления (скорость, прыжок, броня, урон) |
| 4 | Боевая система | Атака врагов, система здоровья |
| 5 | Победа/поражение | Уровень считается завершённым при достижении точки B или при поражении игрока |
| 6 | Пользовательский интерфейс | Отображение здоровья, активных усилений, сообщений |
| 7 | Меню управления | Главное меню, перезапуск уровня, выход из игры |

# **Требования к интерфейсу**

Разработка интерфейса предполагает минималистичный, но функциональный подход, ориентированный на удобство игрока.

**Доступность:**

* Игра работает на ПК с разрешением от 1280×720 и выше;
* Поддержка оконного и полноэкранного режимов.

**Пользовательский опыт (UX):**

* Простая навигация в меню;
* Подсказки на экране (например, клавиши управления);
* Мгновенный отклик на действия пользователя.

**Элементы интерфейса:**

* Полоса здоровья игрока;
* Иконки активных усилений(баффов) с таймером действия;
* Сообщения "Победа", "Поражение";
* Кнопки: «Играть», «Выход», «Заново», «Пауза».

# **Обоснование выбора стека технологий**

Для реализации проекта 2D-платформера было выбрано проверенное сочетание инструментов, обеспечивающее высокую производительность, кроссплатформенность и удобство разработки. Ниже приведены основные используемые технологии и обоснование их выбора.

#### **Игровой движок — Unity (версия 2022.3.9f1 LTS)**

Unity — один из самых популярных движков для разработки как 2D-, так и 3D-игр. Он предоставляет готовые инструменты для визуализации, физики, анимации, пользовательского интерфейса и взаимодействия объектов.

**Преимущества:**

* Встроенные инструменты для создания 2D-игр;
* Мощный визуальный редактор сцен;
* Поддержка всех популярных платформ: Windows, macOS, Android, iOS и др.;
* Большое сообщество и обширная документация;
* Интеграция с Asset Store для использования готовых ресурсов (спрайты, скрипты, шрифты и т. д.).

#### **Язык программирования — C#**

C# — основной язык, используемый в Unity, сочетающий производительность, читаемость и мощь объектно-ориентированного программирования. Он позволяет структурировать проект на независимые модули, что делает код гибким и масштабируемым.

**Преимущества:**

* Поддержка ООП и событийной модели;
* Читабельный синтаксис;
* Отличная интеграция с Unity API;
* Большое количество обучающих материалов.

#### **Среда разработки — Visual Studio**

Visual Studio используется как основная IDE для написания C#-кода в Unity.

**Преимущества:**

* Интеграция с Unity через плагин;
* Интеллектуальная подсветка кода (IntelliSense);
* Поддержка отладки, управления пакетами, Git.

#### **Графические редакторы**

* **Aseprite / Photoshop / GIMP** — для создания пиксельных спрайтов, анимаций и иконок.
* **Tilemap Tool (Unity)** — для построения уровней с использованием тайловых карт.

#### **Звук и музыка**

* **Audacity / Bfxr** — для создания простых звуковых эффектов (удар, прыжок, сбор предмета).
* Музыка может быть взята из свободных источников или сгенерирована вручную.

#### **Система управления версиями (опционально)** — Git + GitHub

Позволяет отслеживать изменения в проекте, хранить резервные копии, откатывать к предыдущим версиям и работать в команде (при необходимости).

# **Проектирование**

# **Проектирование системы**

Проектирование системы является ключевым этапом разработки, на котором формируется структура будущего программного продукта. Оно включает в себя определение логики взаимодействия компонентов, ролей пользователей и сценариев использования. Ниже представлены основные проектные решения, применённые в рамках платформера.



# **Определение группы пользователей**

В игре предполагается один тип пользователя — игрок. Он взаимодействует с системой через основные игровые механики:

* управление персонажем;
* сражения с врагами;
* сбор баффов;
* прохождение уровня;
* взаимодействие с UI (меню, кнопки, сообщения).



Рисунок 5 - Диаграмма прецедентов

# **Функциональное моделирование**

Таблица 2 – Модуль игры

|  |  |
| --- | --- |
| Модуль | Назначение |
| PlayerController | Обработка ввода, движение, прыжки, атаки |
| EnemyAI | Поведение врагов: патрулирование, атака |
| BuffManager | Применение временных эффектов |
| UIManager | Отображение здоровья, баффов, сообщений |
| LevelManager | Логика победы/поражения, рестарт уровня |
| AudioManager | Звуковые эффекты, фоновая музыка |



Рисунок 6-Диаграмма деятельности

В проекте реализуются следующие ключевые алгоритмы:

* Обработка столкновений: при пересечении границы персонажа с объектом происходит вызов события (OnTriggerEnter2D), и выполняется логика (например, получение урона или баффа).
* Таймеры баффов: каждый активный бафф действует ограниченное время и автоматически отключается после завершения таймера.
* Проверка окончания уровня: при достижении игроком точки "B" вызывается событие победы.

# **Разработка модели базы данных**

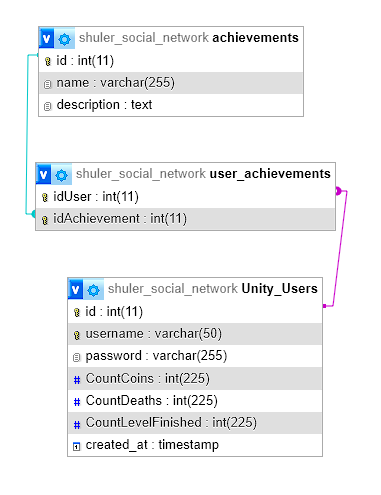


Рисунок 7 - Диаграмма базы данных

# **Проектирование интерфейсов**

Интерфейс пользователя (UI) играет важную роль в восприятии игры и взаимодействии с ней. Он должен быть интуитивно понятным, не перегруженным лишними элементами и функционально отображать необходимую информацию для игрока. В рамках данного проекта интерфейс разрабатывался с учётом принципов простоты, контрастности и адаптивности.

#### **Концепция дизайна интерфейса**

* **Тип интерфейса:** 2D, минималистичный, расположен по краям экрана.
* **Цветовая гамма:** тёплые оттенки для фона, яркие контрастные иконки усилений(баффов), белый/жёлтый текст.
* **Шрифт:** читаемый пиксельный или современный sans-serif (например, через TextMeshPro).

Таблица 3 – Интерфейсы игры

|  |  |  |
| --- | --- | --- |
| **Элемент** | **Назначение** | **Расположение** |
| Полоса здоровья | Отображает текущее здоровье игрока | Верхний левый угол |
| Иконки усилений (баффов) | Показывают активные усиления с таймером | Под полосой здоровья |
| Всплывающие сообщения | Победа, поражение, получение усилений(баффа) и т.д. | Центр экрана |
| Главное меню | Кнопки: Играть, Выход | При запуске и поражении |
| Кнопки управления | Пауза, Перезапуск | В правом верхнем углу |
| Подсказки | WASD — движение, K — атака и т. д. | В начале уровня (по желанию) |

# **Реализация**

# **Реализация основных функций**

На этапе реализации были созданы все основные модули, обеспечивающие функциональность платформера: управление персонажем, поведение врагов, механика временных усилений(баффов), боевая система, система победы/поражения и пользовательский интерфейс.

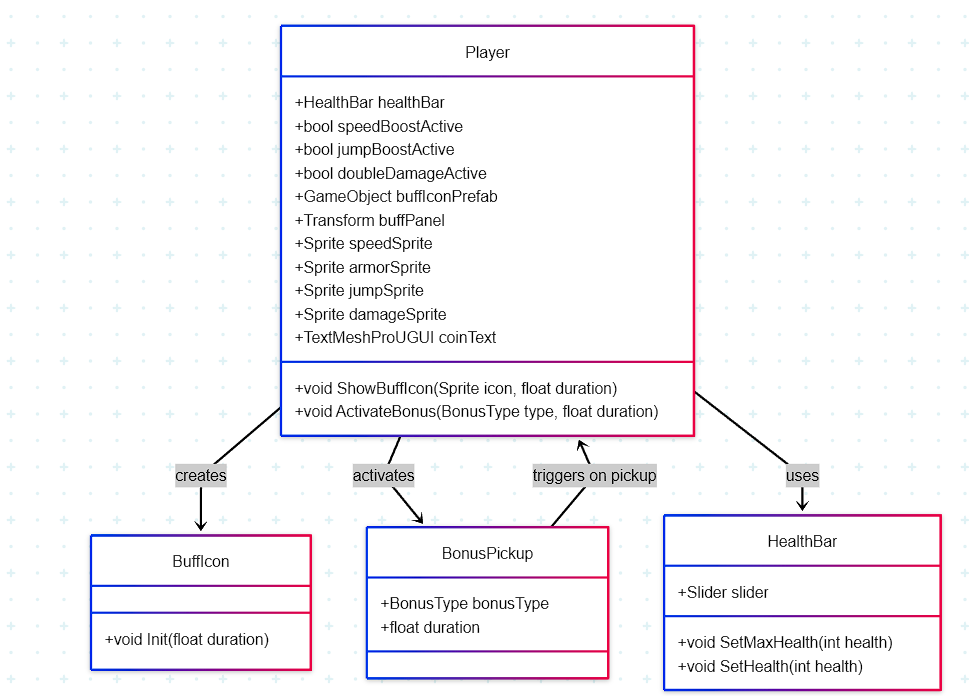


Рисунок 8 - Диаграмма классов (Игрок и система баффов)

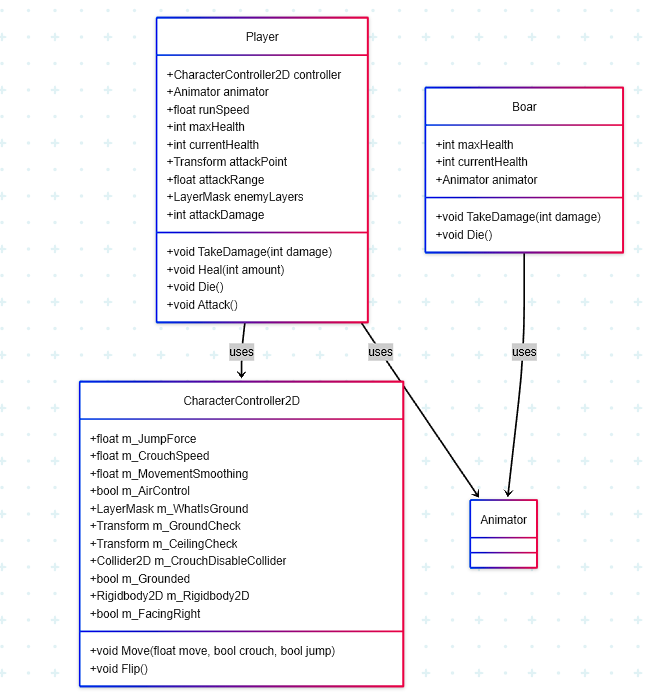


Рисунок 9 - Диаграмма классов (Управление и взаимодействие)

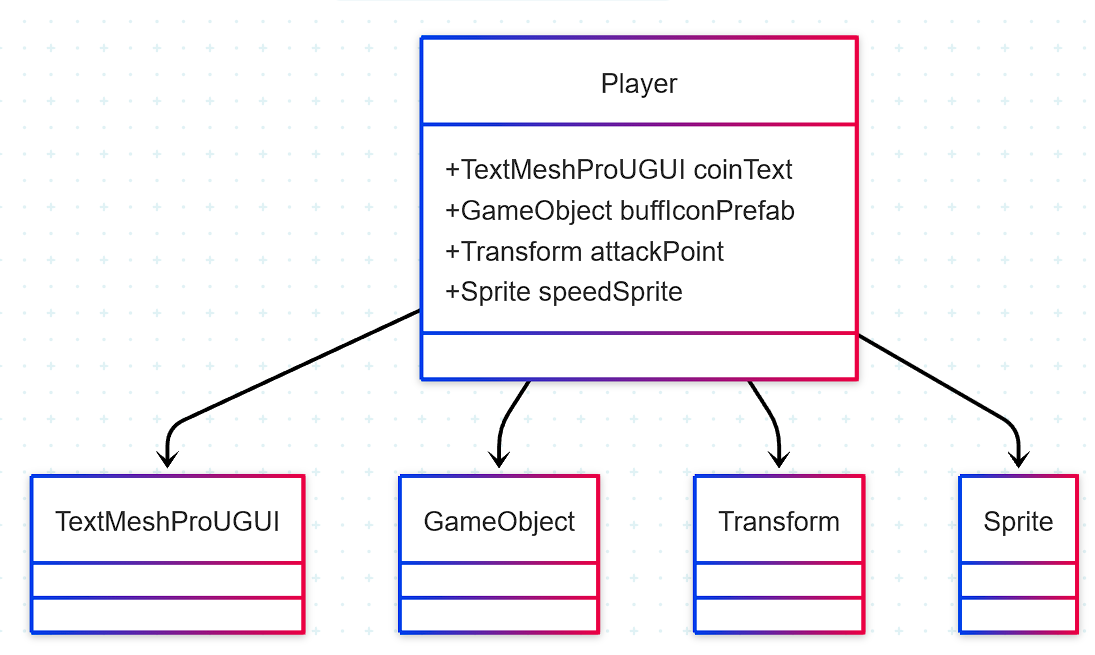


Рисунок 10 - Диаграмма классов (Подключённые типы и зависимости)

**Основные функции**

Таблица 4 – Основных функций игры

|  |  |
| --- | --- |
| Функция | Описание |
| Передвижение игрока | Игрок может бегать, прыгать и разворачиваться в нужном направлении |
| Атака игрока | Игрок может атаковать врагов вблизи и наносить им урон |
| Получение урона | Игрок и враги могут получать урон и умирать при снижении здоровья до нуля |
| Активация бонусов | При касании бонуса игрок получает эффект (ускорение, броня, прыжок, урон) |
| Отображение бонусов | Активные бонусы отображаются в виде иконок с таймерами |
| Отображение здоровья | Полоса здоровья показывает текущее состояние игрока |
| Победа и завершение уровня | При достижении финиша запускается победа и завершение сцены |
| Смерть игрока | При потере всё здоровье игрок умирает, уровень можно перезапустить |
| Взаимодействие с врагами | Враги реагируют на атаки, получают урон и могут быть уничтожены |
| Физика и движение | Реализованы плавные прыжки, падения, касание земли, ускорения |

# **Реализация интерфейсов**

Пользовательский интерфейс (UI) — важный элемент взаимодействия игрока с игрой. От его качества зависит не только удобство управления, но и общее впечатление от проекта. В игре **«The Little Prince»** интерфейс разрабатывался с ориентацией на минимализм, читаемость и соответствие стилистике игры.

Все элементы интерфейса реализованы с использованием встроенной системы **Canvas** в Unity.

Таблица 5 – Компоненты игры

|  |  |
| --- | --- |
| Компонент | Назначение |
| Canvas | Основной контейнер интерфейса |
| Image | Полосы здоровья, иконки, фоны |
| TextMeshProUGUI | Отображение текста (HP, сообщения, таймеры) |
| Button | Кнопки в меню, перезапуск, выход |
| Animator / CanvasGroup | Анимации появления/исчезновения окон |

##### Canvas Scaler:

* Режим: Scale With Screen Size;
* Базовое разрешение: 1920x1080;
* Поддержка адаптивности под разные экраны (16:9, 4:3).

# **Тестирование**

Тестирование является обязательным этапом жизненного цикла программного продукта. Его целью является проверка корректности реализованных функций, устойчивости приложения к ошибкам, а также соответствия заявленным требованиям. В проекте «The Little Prince» тестированию подвергались все основные элементы: механика движения, взаимодействие с врагами и окружением, применение усилений(баффов), работа интерфейса, а также сценарии победы и поражения.

Таблица 6 – Методы тестирования

|  |  |
| --- | --- |
| Метод | Назначение |
| Ручное функциональное тестирование | Проверка всех игровых механик вручную: движение, бой, порталы, баффы |
| Интерактивное тестирование интерфейса | Проверка работы меню, кнопок, отображения информации |
| Тестирование на стабильность | Запуск игры на ПК с различной производительностью |
| Профилирование | Использование Unity Profiler для анализа FPS, потребления памяти и частоты сборок мусора |

**Тест-кейсы**

Таблица 7 – Тест-кейс запуск уровня

|  |  |
| --- | --- |
| Поле | Значение |
| Описание | Проверка запуска уровня и появления игрока |
| Действие | Нажать кнопку «Играть» в главном меню |
| Ожидаемый результат | Уровень загружается, игрок появляется, HUD отображается корректно |
| Фактический результат | Пройден |

Таблица 8 – Тест-кейс получение баффа

|  |  |
| --- | --- |
| Поле | Значение |
| Описание | Проверка применения баффа «Звезда скорости» |
| Действие | Подобрать предмет «Звезда скорости» |
| Ожидаемый результат | Скорость игрока увеличивается, появляется иконка баффа на экране |
| Фактический результат | Пройден |

**UnitTest:**

Start\_InitializesHealthAndMaxHealth

Что должен тестировать:  
Проверяет, что при запуске (например, при инициализации игрока или объекта) поля Health и MaxHealth устанавливаются в правильные начальные значения.

Что должен делать тест:

Вызывает метод Start() или конструктор класса игрока.

Проверяет, что Health равно MaxHealth и эти значения корректны (например, 100 или другое заданное число).

TestPlayerSimplePasses

Что должен тестировать:  
Это скорее шаблонный тест-заготовка, который обычно проверяет, что сам тестовый фреймворк работает и что можно запускать тесты без ошибок.

Что должен делать тест:

Мог бы проверять базовые вещи — например, что игрок создаётся корректно, или что базовые свойства доступны и не вызывают исключений.

Обычно этот тест удаляется или заменяется на реальные проверки.

Die\_SetsIsDeadToTrue

Что должен тестировать:  
Проверяет, что при вызове метода Die() у игрока (или врага) флаг IsDead меняется на true.

Что должен делать тест:

Вызывает метод Die() у объекта игрока или персонажа.

Проверяет, что свойство IsDead стало true.

Можно дополнительно проверить, что дальнейшие действия с персонажем (например, движение или атака) невозможны.

Attack\_AppliesDamageToEnemy

Что должен тестировать:  
Проверяет, что метод Attack() игрока правильно наносит урон врагу, уменьшая его здоровье.

Что должен делать тест:

Создаёт объект врага с определённым здоровьем (например, 100).

Вызывает метод Attack() у игрока, передавая в него врага или атакуя врага.

Проверяет, что здоровье врага уменьшилось на ожидаемую величину урона.

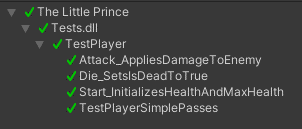


Рисунок 11 - UnitTest

# **Руководство пользователя**

Для запуска игры необходимо:

1. Открыть файл "TheLittlePrince.exe" (или соответствующий файл на macOS).

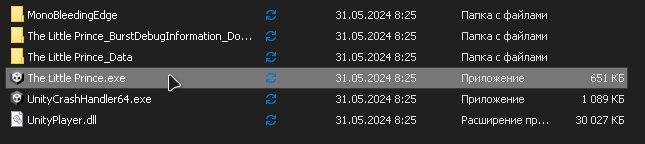


Рисунок 12 - Папка с игрой

1. В главном меню выбрать "Play".



Рисунок 13- Меню игры

1. Управлять персонажем с помощью клавиш WASD или стрелок.



Рисунок 14 - Уровень игры

Сообщение оператору

Программа выводит следующие сообщения:

- "Игра окончена" - при завершении уровня.

В случае сбоя программа автоматически возвращает в главное меню.

### Регистрация и вход в систему

1. После запуска пользователю предлагается пройти **регистрацию**:
   * ввести уникальное имя пользователя;
   * придумать **сложный пароль**;
   * подтвердить регистрацию.
2. При последующих входах:
   * используйте форму авторизации с логином и паролем;
   * при неверном пароле система выдаёт ошибку.

Пароли хранятся в виде зашифрованного хэша, исключая доступ к ним со стороны.

# **Мероприятия по информационной безопасности**

Хотя игра «The Little Prince» ориентирована на одного игрока и работает без подключения к сети, в ней реализована система регистрации пользователей. В процессе разработки были предусмотрены базовые механизмы аутентификации и хранения пользовательских данных. Также учтены ключевые принципы безопасного программирования — для защиты логики игры, предотвращения сбоев, а также подготовки проекта к возможному масштабированию в будущем (например, добавлению онлайн-функций или системы достижений).

Таблица 9 – Возможные угрозы

|  |  |
| --- | --- |
| Угроза | Описание |
| Несанкционированное изменение логики | Вскрытие сборки, модификация механик, получение преимуществ (читерство) |
| Взлом через сохранения | Изменение значений в PlayerPrefs или JSON |
| Сбои при неверной активации объектов | Попытка активации уже уничтоженных или недоступных объектов |
| Утечка ресурсов или ошибок UI | Зависание интерфейса, наложение окон, повторный вызов событий |
| Безопасная идентификация, аутентификация, авторизация | Реализована регистрация; используется хэширование паролей, уникальные ID, разграничение прав доступа (по ролям) |
| Безопасное хранение данных | |  | | --- | |  |  |  | | --- | | Пароли хранятся в виде хэшей (SHA-256/MD5); применена сериализация JSON и защита файлов на уровне ОС или шифрования | |

**Принятые меры**

#### **Защита логики**

* + Все взаимодействия между модулями обернуты в null-проверки и логические условия.
  + Баффы, враги и порталы удаляются с уровня только после завершения всех связанных действий.
  + Используется строгое разделение компонентов и событий.

#### **Минимизация уязвимостей**

* + Исключено сохранение критичных игровых значений в открытых структурах (PlayerPrefs используется только для неважных настроек).
  + Все действия, зависящие от триггеров и коллизий, реализованы через TryGetComponent и проверку состояний.

#### **Защита от нестабильных состояний**

* + Исключена возможность повторного входа в портал;
  + Запрет на многократную активацию кнопок (Button.interactable = false);
  + Обработка всех сцен через LevelManager с централизованной логикой.

#### **Подготовка к масштабированию**

* + При необходимости добавления сохранений будет использоваться сериализация в JSON с валидацией данных;
  + Возможно подключение базовой защиты логики через обфускацию (например, IL2CPP или сторонние инструменты).

#### **Идентификация и аутентификация**

* + Пользователь проходит регистрацию (логин/пароль);
  + Пароль при сохранении хэшируется (без хранения в открытом виде);
  + В системе можно различать профили по уникальному ID или роли (например: игрок, гость);
  + Вход возможен только при совпадении логина и хэша пароля.

#### **Безопасное хранение данных**

* + Данные пользователя (например, прогресс, достижения) сериализуются в зашифрованный JSON;
  + При необходимости используется подпись или контрольная сумма для защиты от подмены;
  + Хранилище размещается в системной директории Application.persistentDataPath, что исключает простой доступ к файлам без прав.

**Рекомендации пользователям по безопасной работе**

* **Используйте надёжные пароли при регистрации.**  
  Не используйте простые пароли вроде 12345, qwerty, admin и т. п. Желательно использовать комбинации из букв верхнего и нижнего регистра, цифр и символов.
* **Не передавайте свои учётные данные третьим лицам.**  
  Учётная запись предназначена только для личного использования. При игре на чужом устройстве используйте временную гостевую сессию (если доступна).
* **Избегайте внесения изменений в игровые файлы вручную.**  
  Не редактируйте конфигурационные или пользовательские файлы игры. Это может привести к потере данных или нарушению логики приложения.
* **Сохраняйте резервные копии прогресса.**  
  В случае сбоя, потери данных или переустановки системы будет возможен откат к сохранённому состоянию.
* **Не запускайте модифицированные версии игры.**  
  Используйте только официальную сборку приложения, полученную из доверенного источника (например, с сайта проекта). Внешние модификации могут содержать вредоносный код.
* **Регулярно обновляйте приложение.**  
  Новые версии могут включать исправления ошибок и улучшения безопасности.

# **Заключение**

В рамках дипломного проекта была разработана 2D-игра в жанре платформера, реализованная с использованием игрового движка Unity и языка программирования C#. Проект охватывает полный цикл разработки программного продукта: от анализа предметной области и проектирования до реализации, тестирования и оценки информационной безопасности.

Целью игры является прохождение уровня от начальной до конечной точки, преодоление препятствий, взаимодействие с усилениями (баффами) и противостояние врагам. Благодаря использованию объектно-ориентированной архитектуры, игровой процесс получился гибким и масштабируемым, что позволяет в будущем легко добавлять новые функции: уровни, врагов, способности, режимы сложности и даже онлайн-составляющую.

В процессе реализации были достигнуты следующие результаты:

* Разработан прототип 2D-платформера с динамичным игровым процессом;
* Реализованы механики движения, атаки, обработки столкновений и временных усилений;
* Создан адаптивный пользовательский интерфейс;
* Проведено тестирование всех ключевых функций, подтверждена стабильность работы;
* Учтены базовые меры по обеспечению информационной безопасности;
* Использован современный стек технологий, включая Unity, C#, TextMeshPro и ScriptableObjects.

Разработанная игра может использоваться как демонстрационный проект, обучающее пособие по основам создания платформеров в Unity, а также как база для создания более сложных коммерческих или образовательных игровых продуктов.

# **С****писок источников**

**Нормативная документация**

* 1. ГОСТ 34.602-2020 Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [Электронный ресурс] - https://protect.gost.ru/document1.aspx?control=31&id=241754
  2. ГОСТ 34.201-2020 Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем [Электронный ресурс] - https://protect.gost.ru/document1.aspx?control=31&id=241756
  3. ГОСТ Р ИСО/МЭК 25051-2017 - Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию [Электронный ресурс] - <https://protect.gost.ru/document.aspx?control=7&id=217667>
  4. ГОСТ 19.106-78 Единая система программной документации. Требования к программным документам, выполненным печатным способом [Электронный ресурс] - https://protect.gost.ru/document.aspx?control=7&id=156370
  5. ГОСТ 19.401-78 Единая система программной документации. Текст программы. Требования к содержанию и оформлению [Электронный ресурс] -https://protect.gost.ru/document.aspx?control=7&id=155463
  6. ГОСТ Р 2.105-2019 Единая система конструкторской документации. Общие требования к текстовым документам [Электронный ресурс] - <https://protect.gost.ru/document1.aspx?control=31&baseC=6&page=0&month=1&year=2025&search=%D0%93%D0%9E%D0%A1%D0%A2%20%D0%A0%202.105-2019&id=237857>

**Интернет – ресурсы**

1. Brackeys. Обучающие материалы по Unity (видеоуроки) [Электронный ресурс]. – Режим доступа: https://www.youtube.com/c/Brackeys, свободный. – (дата обращения: 15.05.2025).
2. GitHub. Официальные репозитории Unity Technologies [Электронный ресурс]. – Режим доступа: https://github.com/Unity-Technologies, свободный. – (дата обращения: 15.05.2025).
3. itch.io. Платформа публикации инди-игр [Электронный ресурс]. – Режим доступа: https://itch.io/, свободный. – (дата обращения: 15.05.2025).
4. JetBrains. Rider – среда разработки с поддержкой Unity [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/rider/features/unity.html, свободный. – (дата обращения: 15.05.2025).
5. Microsoft. Безопасное хранение паролей в приложениях .NET [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/web-api/overview/security/security-hashing-and-salting-passwords, свободный. – (дата обращения: 15.05.2025).
6. Microsoft. Документация по языку C# [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/, свободный. – (дата обращения: 15.05.2025).
7. OpenGameArt. Бесплатные ресурсы для разработки игр [Электронный ресурс]. – Режим доступа: https://opengameart.org/, свободный. – (дата обращения: 15.05.2025).
8. OWASP Foundation. Руководство по безопасной аутентификации [Электронный ресурс]. – Режим доступа: https://owasp.org/www-project-cheat-sheets/cheatsheets/Authentication\_Cheat\_Sheet.html, свободный. – (дата обращения: 15.05.2025).
9. StackExchange. GameDev StackExchange – форум по разработке игр [Электронный ресурс]. – Режим доступа: https://gamedev.stackexchange.com/, свободный. – (дата обращения: 15.05.2025).
10. Unity Asset Store. Магазин ассетов для Unity [Электронный ресурс]. –Режим доступа: https://assetstore.unity.com/, свободный. – (дата обращения: 15.05.2025).
11. Unity Forum. User Login and Save System in Unity – Community Example [Электронный ресурс]. – Режим доступа: https://forum.unity.com/threads/user-login-and-save-system-in-unity.818794/, свободный. – (дата обращения: 15.05.2025).
12. Unity Technologies. Unity Manual [Электронный ресурс]. – Режим доступа: https://docs.unity3d.com/Manual/index.html, свободный. – (дата обращения: 15.05.2025).
13. Unity Technologies. Unity Scripting API [Электронный ресурс]. – Режим доступа: https://docs.unity3d.com/ScriptReference/index.html, свободный. – (дата обращения: 15.05.2025).

# **Приложение А**

**Скрипт для управления персонажам**

void Update()

{

if (currentHealth <= 0)

{

return;

}

else

{

horizontalMove = Input.GetAxisRaw("Horizontal") \* runSpeed;

animator.SetFloat("Speed", Mathf.Abs(horizontalMove));

if (Input.GetButtonDown("Jump"))

{

jump = true;

}

if (Input.GetKeyDown(KeyCode.Mouse0))

{

Attack();

}

}

}

# **Приложение Б**

**Скрипт для активации бонусов**

public void ActivateBonus(BonusType type, float duration)

{

switch (type)

{

case BonusType.SpeedBoost:

if (speedCoroutine != null) StopCoroutine(speedCoroutine);

speedCoroutine = StartCoroutine(SpeedBoost(duration));

ShowBuffIcon(speedSprite, duration);

break;

case BonusType.Invulnerability:

if (invulCoroutine != null) StopCoroutine(invulCoroutine);

invulCoroutine = StartCoroutine(TemporaryInvulnerability(duration));

ShowBuffIcon(armorSprite, duration);

break;

case BonusType.JumpBoost:

if (jumpCoroutine != null) StopCoroutine(jumpCoroutine);

jumpCoroutine = StartCoroutine(JumpBoost(duration));

ShowBuffIcon(jumpSprite, duration);

break;

case BonusType.DoubleDamage:

if (damageCoroutine != null) StopCoroutine(damageCoroutine);

damageCoroutine = StartCoroutine(DoubleDamage(duration));

ShowBuffIcon(damageSprite, duration);

break;

}

}