СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 3](#_Toc198062212)

[**1.1. Наименование программы** 3](#_Toc198062213)

[**1.2. Основания для разработки** 3](#_Toc198062214)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc198062215)

[**2.1. Функциональное назначение** 4](#_Toc198062216)

[**2.2. Эксплуатационное назначение** 4](#_Toc198062217)

[**2.3 Краткая характеристика области применения** 4](#_Toc198062218)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5](#_Toc198062219)

[**3.1. Постановка задачи на разработку программы** 5](#_Toc198062220)

[**3.1.1. Требования к игровым механикам главного героя.** 5](#_Toc198062221)

[**3.1.2. Требования к функционалу боссов.** 6](#_Toc198062222)

[**3.1.3. Требования к функционалу достижений.** 6](#_Toc198062223)

[**3.1.4. Требования к снаряжению главного героя.** 7](#_Toc198062224)

[**3.2. Описание архитектуры программы** 7](#_Toc198062225)

[**3.3. Обоснование выбора архитектуры программы** 8](#_Toc198062226)

[**3.4 Описание и обоснование алгоритма программы** 9](#_Toc198062227)

[**3.4.1 Игровой процесс** 9](#_Toc198062228)

[**3.4.2 Передвижение главного героя** 9](#_Toc198062229)

[**3.4.2.1 Состояния главного героя** 9](#_Toc198062230)

[**3.4.2.2 Движение** 10](#_Toc198062231)

[**3.4.2.3 Рывок** 11](#_Toc198062232)

[**3.4.2.4 Коллизии главного героя** 11](#_Toc198062233)

[**3.4.2.5 Анимация главного героя** 11](#_Toc198062234)

[**3.4.3 Управление здоровьем** 11](#_Toc198062235)

[**3.4.3.1 Получение урона** 11](#_Toc198062236)

[**3.4.3.2 Неуязвимость** 11](#_Toc198062237)

[**3.4.3.3 Восстановление здоровья** 12](#_Toc198062238)

[**3.4.4 Поведение босса** 12](#_Toc198062239)

[**3.4.5 Снаряжение** 12](#_Toc198062240)

[**3.4.6 Отслеживание и получение достижений** 13](#_Toc198062241)

[**3.4.7 Игровой процесс со стороны врагов** 13](#_Toc198062242)

[**3.4.8 Поведение врагов-юнитов** 14](#_Toc198062243)

[**3.4.9 Поведение объектов-атак с анимацией** 14](#_Toc198062244)

[**3.4.10 Поведение объектов окружения** 14](#_Toc198062245)

[**3.5 Описание и обоснование выбора метода организации входных и выходных данных** 15](#_Toc198062246)

[**3.5.1 Входные данные** 15](#_Toc198062247)

[**3.5.2 Выходные данные** 15](#_Toc198062248)

[4. Ожидаемые технико-экономические показатели 17](#_Toc198062249)

[**4.1. Ориентировочная экономическая эффективность.** 17](#_Toc198062250)

[**4.2. Предполагаемая потребность.** 17](#_Toc198062251)

[**4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами** 17](#_Toc198062252)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18](#_Toc198062253)

[ПРИЛОЖЕНИЕ 1 20](#_Toc198062254)

[ПРИЛОЖЕНИЕ 2 22](#_Toc198062255)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 24](#_Toc198062256)

# **1. ВВЕДЕНИЕ**

## **1.1. Наименование программы**

Наименование программы – 2D игра «Ночные сны»

Наименование программы на английском языке – 2D game «Night dreams»

Краткое наименование программы – «Ночные сны»

## **1.2. Основания для разработки**

Программа выполняется в рамках темы курсового проекта в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия» Национального исследовательского университета «Высшая школа экономики», факультет компьютерных наук, департамент программной инженерии.

Основанием для разработки служит учебный план подготовки бакалавров по направлению 09.03.04 «Программная инженерия» и утвержденная академическим руководителем тема курсового проекта.

# **2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**

## **2.1. Функциональное назначение**

«Ночные сны» представляет собой однопользовательскую 2D игру. Gameplay состоит из последовательного сражения с боссами в фиксированных различных локациях, а также исследования окружающего игрового сеттинга с целью поиска новых противников, снаряжений и навыков по мере продвижения по сюжету.

## **2.2. Эксплуатационное назначение**

Игра предназначена для досуга и развлечения. «Ночные сны» нацелена на любителей хардкорных игр и поклонников жанра «run and gun». Игра охватывает аудиторию широкого возрастного диапазона (от 6 до 60 лет), заинтересованную в прохождении платформеров.

## **2.3 Краткая характеристика области применения**

«Ночные сны» представляет собой 2D игру в жанре "run and gun", выполненную на движке Unity[10]. Игра погружает игрока в атмосферу ночного мира с мистической тематикой и интенсивными боями, где основная задача — выжить, уничтожая врагов и продвигаясь через уровни, наполненные опасными боссами. Игровой процесс сочетает динамичное движение, сражения с врагами, коллекционирование усилений и использование разнообразного оружия.

# **3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**

## **3.1. Постановка задачи на разработку программы**

Программный продукт должен предоставлять следующий функционал:

## **3.1.1. Требования к игровым механикам главного героя.**

Перечисленные ниже механики должны быть разработаны для главного героя, но в момент игры персонаж может пользоваться ограниченным набором способностей, по причине их недоступности в определенный момент игры либо из-за ограничения их количества.

В каждый момент времени персонаж может находиться в одном из трех состояний: стоять на платформе, быть в приседе на платформе, не находится на платформе в случае прыжка/падения.

Все игровые механики персонажа осуществляется по средствам нажатия установленных в настройках кнопок.

Механики передвижения:

1. Персонаж передвигается по горизонтальной оси, если находится на платформе и может передвигаться по двум осям в случае прыжка/падения.
2. Прыжок – передвижение персонажа по вертикальной оси, который можно совершить, находясь на платформе.
3. Присед – уменьшение персонажа по вертикальной оси.
4. Рывок – ускоренное передвижение по горизонтальной оси на фиксированное расстояние.
5. Спуск с платформы – возможность прохода сквозь платформу, на которой находится персонаж.
6. Подъем на платформу – возможность прохода сквозь платформу, при условии, что персонаж находится под ней.

Механики нанесения урона:

1. Удар ближнего боя – нанесение урона на ограниченное расстояние
2. Удар дальнего боя – нанесение урона оружием, позволяющим бить дальше, чем ближний бой. Необходимо реализовать несколько разновидностей удара дальнего боя, которые будут отличаться уроном, дальностью, скорострельностью, траекторией снаряда и областью поражения, а также будут различаться визуально.

Механика восполнения здоровья:

1. Фиксированное количество HP в начале уровня.
2. Специальные предметы, восстанавливающие здоровье.

## **3.1.2. Требования к функционалу боссов.**

Основными противниками игры являются уникальные боссы.

При победе над некоторыми боссами главный герой получает новые предметы/способности, которые будут отображаться в HUDе.

* Сражение с каждым боссом происходит на уникальной фиксированной игровой локации.
* У каждого босса есть определенное количество очков здоровья
* У каждого босса есть определенное количество фаз (от 1 до 3), которые характеризуются различными видами атак, их скоростью, количеством очков здоровья, изменением визуализации как самого босса, так и игровой локации. Так же в зависимости от фазы может меняться траектория передвижения босса.
* Требуется реализовать по меньшей мере 3 уникальных боссов, различающихся как дизайном, так и игровыми механиками, включая фазы, виды атак, стиль передвижения.

## **3.1.3. Требования к функционалу достижений.**

* Возможность получать достижения по мере прохождения игры при выполнении заранее прописанных условий.
* Отображение полученных достижений в окне «Достижения».

## **3.1.4. Требования к снаряжению главного героя.**

* Требуется реализовать пять различных видов оружия, которые отличаются: дальностью боя, наносимым уроном, траекторией, областью поражения, скоростью атаки и уникальными свойствами.

## 

## **3.2. Описание архитектуры программы**

Архитектура построена по принципу модульного монолита. То есть полученный программный продукт разворачивается одним целым, а внутри приложение разделено на модули посредством языка C#. Модули состоят из MonoBehavior скриптов, которые представляют из себя функциональные классы, с определенной зоной ответственности. Каждый модуль инкапсулирует свою логику, а взаимодействие между ними происходит за счет:

* Событий (UnityEvent)
* Обращений через ссылки на компоненты (FindObjectOfType<TargetClass>)
* Коммуникаций используя вызовы API

Таким образом в рамках игры “Ночные сны” можно выделить следующие модули архитектуры и их зоны ответственности:

1. Модуль главного героя

Отвечает за управление движением и действиями игрока, включая передвижение, прыжок, рывок, стрельбу и перемещение через платформы. Взаимодействие с анимацией, физикой и коллизиями. Отслеживает получение урона, нахождения на земле, управление направлением и позицией стрельбы. Взаимодействует с системой ввода.

2. Модуль босса

Управляет логикой поведения боссов, разделённой на фазы, с последовательным запуском различных атак. Реализует обработку анимации, состояния здоровья и смерть боссов. Взаимодействие с физикой и коллизиями боссов.

3. Модуль оружия

Взаимодействие со снарядами, определяя их скорость, направление и жизненный цикл. Управляет условиями стрельбы и ограничениями по частоте. Отслеживание попадания снарядов во врагов и нанесения урона.

4. Модуль достижений

Отслеживает выполнение достижений.

5. Модуль врагов-юнитов

Реализует поведение обычных врагов. Взаимодействие с физикой, анимацией, коллизиями. Отвечает за удаление врагов после смерти и, при необходимости, за повторное появление.

6. Модуль объектов-атак с анимацией

Управляет окружением, которое наносит урон по определённой анимации — шипы, лазеры и другие ловушки.

7. Модуль окружения

Взаимодействия с объектами окружения: платформы, пол, стены уровня, и другие объекты, от которых главный герой не получает урон.

## **3.3. Обоснование выбора архитектуры программы**

Четкое разделение ответственности по модулям, упрощает разработку. Причем модули находятся в одном пространстве, что обеспечивает удобную коммуникацию между ними. Также с такой архитектурой легко расширять программный продукт добавляя необходимые элементы в нужный модуль. Более сложные архитектуры были бы избыточными для создания такой 2D игры и привели бы к потере времени и излишней сложности.

## **3.4 Описание и обоснование алгоритма программы**

## **3.4.1 Игровой процесс**

На каждом уровне пользователь управляет главным героем, задача которого – убить босса. Игрок имеет при себе оружие, использование которого наносит урон боссу, при попадании по нему. Босс имеет несколько фаз и определенное количество атак, которые наносят урон по игроку. Также в уровне присутствуют объекты окружения, которые являются платформами. Кроме босса, на уровне появляются враги-юниты и анимированные объекты (например, шипы), которые также могут наносить урон главному герою. У босса и игрока есть фиксированное количество HP, в случае если HP героя оказалось меньше или равно 0, то пользователь проиграл в этом уровне. Если же главный герой имеет количество HP большее 0 и у босса количество HP меньше или равно 0, то пользователь победил в этом уровне. Таким образом уровень заканчивается при достижении HP равное 0 у босса или игрока.

## **3.4.2 Передвижение главного героя**

Поскольку главный герой — это многослойный объект, то описание алгоритма передвижения, стоит разобрать также по условным группам:

## **3.4.2.1 Состояния главного героя**

Перемещения главного героя реализовано, с использованием состояний. Герою назначен менеджер состояний, который отслеживает его положение на уровне. Абстрактный класс состояния (от которого наследуются все возможные состояния) контролирует 4 изменения поведения: вход в это состояние, выход из состояния, update состояния, нажатия кнопки прыжка.

Состояние может быть одно из трех:

1) **Состояние приседа.** Достигаетсяпри нажатии соответствующей кнопки (по умолчанию S). При входе/выходе в это состояние мы меняем bool переменную, для установки соответствующей анимации. Update состояния заключается в проверке того, встал главный герой или нет посредством отслеживания его координаты по оси y. При прыжке мы меняем состояние на состояние нахождения на земле (здесь же происходит проверка на onewayplatform – платформа с которой можно спуститься при нажатии клавиш пробел + S по умолчанию. Такие платформы имеют тэг OneWayPlatfrom, при успешной проверке этого тэга, герой падает сквозь эту платформу).

**2) Состояние нахождения на земле.** Герой находится в этом состоянии если он не в воздухе и не в приседе. Аналогично при входе/выходе в это состояние меняется bool переменная для аниматора. Update состояния меняет состояние на присед при значении координаты по оси y < 0, а также задает скорость с которой двигается персонаж, используя RigitBody2D. При прыжке состояние меняется на состояние прыжка.

**3) Состояние прыжка.** Герой находиться в этом состояние, если он не стоит на платформе. При входе мы меняем скорость движения в воздухе, взаимодействуя с rigitBody. Также при входе и выходе мы меняем значение bool переменной для аниматора. Update меняет состояния на состояние нахождения на земле, при True флаге из менеджера состояний.

Менеджер состояний при инициализации главного героя создает объекты этих 3 состояний и изменяет значение текущего состояния на одно из них. В целом полностью за передвижение отвечает менеджер состояний.

## **3.4.2.2 Движение**

Игрок передвигается посредством двух клавиш (по умолчанию A – движение влево и D – движение вправо), как на земле, так и в воздухе. Передвижение происходит за счет изменения координат при нажатии клавиш в менеджере состояний. Также при передвижении активируется анимация бега.

## **3.4.2.3 Рывок**

Игрок может совершить рывок при нажатии клавиши (по умолчанию Left Shift). С помощью rigitBody мы отключаем гравитацию, и значительно увеличиваем скорость. В этот же момент устанавливаем тригер для воспроизведения анимации.

## **3.4.2.4 Коллизии главного героя**

На игроке находятся два 2d Collaider. Один является hit box и

его размер совпадает с размером модели, он считывает получения

урона персонажем и является isTrigger, то есть проходит сквозь другие объекты. Второй коллайдер не является тригерным и нужен для взаимодействия с платформами, благодаря ему персонаж стоит на платформах. При приседе hit box коллайдер уменьшается в анимации. Менеджер состояний проверяет коллизии второго коллайдера и если у объекта слой Ground, то персонаж может стоять на нем.

## **3.4.2.5 Анимация главного героя**

За счет переключения между состояниями, в которых мы явно обращаемся к тригерам, в аниматоре изменяется анимация главного героя и размеры коллайдера для получения урона.

## **3.4.3 Управление здоровьем**

## **3.4.3.1 Получение урона**

Изначально персонаж имеет фиксированное количество здоровья. В начале каждого уровня и в начале каждой попытки прохождения здоровье полностью восстанавливается. На объектах, соприкосновения с которыми могут нанести урон, находиться пустой скрипт Enemy.cs, выполняющий роль флага врага. Если коллайдер главного героя сталкивается с объектом с этим скриптом, то герой получает урон.

## **3.4.3.2 Неуязвимость**

После получения урона, персонаж фиксированное время не может получить урон, на аниматоре это отображается тем, что спрайт персонажа мигает белым цветом. Это достигается за счет игнорирование коллайдером объектов со скриптом Enemy

## **3.4.3.3 Восстановление здоровья**

При убийстве некоторых врагов-юнитов, персонаж может пополнить запас здоровья. На таких врагах висят скрипты EnemyHealth, и при смерти от оружия, а не от уничтожения объекта системой, вызывается метод Heal, который увеличивает здоровье пользователя

## **3.4.4 Поведение босса**

Перемещение босса и его атаки управляются одним контроллером.

В котором в качестве полей указываются расположения атаки(с помощью Transform). При вызове советующего метода атаки, срабатывают тригеры анимации для босса и тригеры анимации для объекта атаки. То есть с помощью Instatiate мы спавним префаб атаки, которую совершает босс (например, лазер или сгустки слизи). Проигрывается анимация атаки, в которой изменяется коллайдер нанесения урона и после чего мы уничтожаем объект атаки. Все передвижения боссов происходят за счет анимации и вызова событий во время анимации. С помощью событий мы вызываем необходимые методы из контроллера босса.

## **3.4.5 Снаряжение**

В игре предусмотрено 5 видов оружий. Стрельба происходит за счет нажатия одной клавиши (стрелка вправо по умолчанию). Дочерний компонент главного героя – Transform положение, откуда ведется выстрел. При стрельбе/ударе из оружия в этой точке спавнятся объекты из префаба нужного типа. Объект снаряда летит по заданной траектории, причем на нем находится коллайдер, который проверяет столкновения. При столкновении с объектом, в случае если это enemy, то босс теряет HP и снаряд уничтожается. Для оружий ближнего боя при ударе снаряд остается на месте и не летит по траектории, вместо этого он имеет достаточно большой коллайдер – по сути зона поражения оружия, который проверяет есть ли в этой зоне враг.

Во время передвижения и изменения состояния главного героя изменяется положение точки выстрела. Так при нажатии вверх (клавиша W по умолчанию) и при нажатии клавиш вверх и вправо (W + D по умолчанию) главный герой будет стрелять вверх и по диагонали соответственно.

Оружия в игре:

1) Лунный револьвер. Дальнобойное оружие, которое стреляет по прямой траектории нанося 2 HP за попадание одного снаряда. Стреляет на любое по дальности расстояние.

2) Веер. Дальнобойное оружие, которое стреляет по сектору окружности сразу 4 снарядами. Урон от одного снаряда 0.8 HP.

3) Когти. Оружие ближнего боя. Наносит 6 урона за один удар. Игрок начинает уровень с +2 HP.

4)Меч. Оружие ближнего боя. Наносит 4 урона за один удар. Зона поражения в 2 раза больше, чем у когтей.

5) Призрачные когти. Дальнобойное оружие. Наносит 1.5 урона за попадание, игнорирует препятствия.

## **3.4.6 Отслеживание и получение достижений**

Работа с достижениями реализуется с помощью одного централизованного менеджера, который слушает события. В случае определенных действий вызываются соответствующие события, на который подписан менеджер. При завершении уровней или выхода в меню менеджер обновляет статусы заработанных достижений.

## **3.4.7 Игровой процесс со стороны врагов**

Игровым процессом управляет AttackManager, он запускает блоки атак возможные для текущей фазы. Блоки атак сконструированы из атаки босса, атаки врагов-юнитов, а также атаки отдельных объектов с анимацией. Между блоками атак происходит задержка в виде WaitForSeconds. При запуске каждого блока, происходит проверка количества HP босса, и при достижении нужного происходит смена фазы в виде установки флага нужной фазы. При смерти босса, или смерти главного героя, посылается сигнал об остановки атак и окончании игры.

## **3.4.8 Поведение врагов-юнитов**

Враги-юниты представляют из себя объекты, которые перемещаются по заданной траектории. Каждый из таких объектов имеет скрипт Enemy.cs, для нанесения урона персонажу. Путь перемещения представляет из себя набор точек Transform, которые передаются в AttackManager. Враги имеют тригерный коллайдер, для прохождения сквозь игрока при столкновении.

## **3.4.9 Поведение объектов-атак с анимацией**

Объекты-атаки – GameObjects, анимация которых запускается из AttackManager, в целом они состоят из последовательного изменения анимации, с изменением коллайдера. В пассивном состоянии коллайдер отключен, и игрок не получает урона при прикосновении, когда происходит запуск аниматора коллайдер включается и изменяется в соответствии с размерами объекта. Таки объекты имеют скрипт enemy для нанесения урона.

## **3.4.10 Поведение объектов окружения**

Объекты окружения представляют из себя платформы, на которых может стоять пользователь, причем он может прыгать через них снизу вверх и спускать с них нажимая пробел + S. Спуск с платформы обрабатывается в состояние приседа и работает при тэге платформы OneWayPlatform. Возможность стоять на таких платформах из-за слоя Ground, который проверяется в PlayerStateManager. Возможность прыгать сквозь такие платформы снизу вверх настроена с Platfrom Effector 2D.

Сюда же относится пол, который имеет прямоугольную форму и слой ground для стояния на нем. На каждом уровне есть 2 невидимых стены (спрайт рендер отключен), которые блокируют выход пользователя за экран.

## **3.5 Описание и обоснование выбора метода организации входных и выходных данных**

## **3.5.1 Входные данные**

В качестве входных данных используется события нажатия клавиш клавиатуры и клика мыши. Управление в игре по умолчанию:

Перемещение(A/D)

Прыжок (Space)

Рывок (Left Shift)

Стрельба (стрелка вправо)

Прицел вверх(W)

Приседания(S)

Система Input System в Unity позволяет гибко обрабатывать пользовательские команды и настраивать управление. Также такие входные данные позволяют реализовывать сложные игровые действия, причем такой подход привычен для большинства игроков.

Кроме этого, в роли входных данных выступают файлы сохранения прогресса прохождения пользователя в предыдущих играх. Это обосновывается необходимостью долговременного хранения пользовательских данных и возможности возвращения к предыдущим состояниям игры.

## **3.5.2 Выходные данные**

В качестве выходных данных выступают изменения визуального состояния на экране, звуковые эффекты и музыка, состояния игрового мира, сохранения прогресса. Такая организация выходных данных является наиболее популярной и привычной для большинства игроков.

**3.6 Описание и обоснование выбора состава технических и программных средств**

Игра написана на движке Unity[10]. Выбор этой программной платформы обусловлен удобной возможностью создания 2D игр, при поддержке работы с анимацией, коллайдерами, сценами, объектами и возможностью быстрого масштабирования проекта. Скрипты написаны на языке C#[11], в силу поддержки Unity языка C# “из коробки”. В качестве редактора спрайтов был использован Aseprite[12], из-за простоты работы с покадровой анимацией и экспорта в формат, поддерживаемый Unity.

Для успешного функционирования программы требуется иметь в распоряжении стандартные устройства ввода – компьютерную мышь и клавиатуру, а также персональный компьютер, оснащенный нижеописанными техническими характеристиками:

• **CPU**: Intel Core 2 Duo E8400 или AMD Athlon 64 X2 6000+

• **RAM**: 2 ГБ

• **Видеокарта**: NVIDIA GeForce 9600 GT или AMD HD 2400

• **DirectX**: Версия 9.0 и выше

• **SSD**: 4 ГБ свободного пространства

Был выбран именно такой набор технических средств на основе анализа работы программы и набора технических средств, используемых при разработке.

# **4. Ожидаемые технико-экономические показатели**

## **4.1. Ориентировочная экономическая эффективность.**

В рамках данной работы расчет экономической эффективности не предусмотрен.

## **4.2. Предполагаемая потребность.**

«Ночные сны» разрабатывается для удовлетворения следующих предполагаемых потребностей пользователей:

* Игроки получают удовольствие от преодоления испытаний и оттачивания мастерства, что позволяет им ощущать личный прогресс и достижение целей.
* Высокий уровень сложности делает игру привлекательной для тех, кто ищет настоящие испытания, сравнимые с такими проектами, как Cuphead, Contra или Hollow Knight.
* Игроки могут соревноваться между собой, кто быстрее и лучше проходит уровни или побеждает боссов.
* Стриминг и speedrun: Сложность игры привлекает сообщество стримеров и любителей соревновательных прохождений. Установление рекордов — еще один источник мотивации для игроков.

## **4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами**

«Ночные сны» нацелена на любителей хардкорных игр и поклонников жанра «run and gun». В игровой сфере 2D платформеров с высоким уровнем сложности существует небольшое количество зарекомендовавших себя игр, которые стали известны многим игрокам, примером таких игр являются «Cuphead», «Hollow knight», «Don’t starve». В жанре «run and gun» выборка игр сложного уровня уменьшается многократно, так что на рынке пользуется спросом лишь одна игра подобного жанра – «Cuphead». Поэтому в условиях низкого предложения было решено создать достойную альтернативу – игру «Ночные сны», которая даст свежий игровой опыт и новую игровую атмосферу любителям данного игрового жанра.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. Unity, [Электронный ресурс] / Unity. Режим доступа: <https://docs.unity.com/>, свободный. (дата обращения: 10.01.2025)
11. С#, [Электронный ресурс] / C#. Режим доступа:

https://dotnet.microsoft.com/en-us/languages/csharp/,

свободный. (дата обращения 10.01.2025)

1. Aseprite, [Электронный ресурс] / Aseprite. Режим доступа:

https://www.aseprite.org, свободный. (дата обращения: 10.01.2025)

1. GitHub, Inc. [Электронный ресурс] / GitHub. Режим доступа: <https://github.com/>, свободный. (дата обращения: 10.01.2025).
2. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# **ПРИЛОЖЕНИЕ 1**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

**Пространство имен Player**

|  |  |
| --- | --- |
| Класс | Назначение класса |
| State | Абстрактный класс состояния |
| PlayerJumpState | Состояние прыжка |
| PlayerCrouchState | Состояние приседа |
| PlayerGroundState | Состояние нахождение на платформе, не в приседе |
| PlayerStateManager | Менеджер управления состояниями |
| PlayerWeapon | Менеджер управления атакой персонажа |
| Health | Класс отвечающий за получение урона пользователем |

**Пространство имен Weapon**

|  |  |
| --- | --- |
| Класс | Назначение класса |
| Bullet | Управление префабом снаряда револьвера |
| Fan | Управление префабом снаряда веера |
| Sword | Управление префабом снаряда меча |
| Claw | Управление префабом снаряда когтей |
| ShadowClaw | Управление префабом снаряда призрачных когтей |
| PlayerWeapon | Менеджер управления атакой персонажа |
| DamageDealer | Класс отвечающий за нанесения урона пользователем |

**Пространство имен Achievement**

|  |  |
| --- | --- |
| Класс | Назначение класса |
| Achievement | Базовый класс представляющий достижение |
| AchievementManager | Контроллер для отслеживания, проверки и активации достижений |
| ProgressTracker | Класс для отслеживания выполнения условий достижений |

**Пространство имен Boss**

|  |  |
| --- | --- |
| Класс | Назначение класса |
| BossController\_{NAME} | Контроллер для управления боссом NAME |
| Enemy | Флаг для нанесения урона |
| EnemyHealth\_{NAME} | Здоровье босса NAME |
| AttackManager | Менеджер управления переключения фазами босса и блоками атак |

**Пространство имен Enemies**

|  |  |
| --- | --- |
| Класс | Назначение класса |
| Enemy\_{NAME} | Управления логикой передвижения и нанесения урона врагом-юнитом NAME |
| Enemy | Флаг для нанесения урона |
| EnemyHealth | Здоровье enemy NAME |

# **ПРИЛОЖЕНИЕ 2**

|  |  |
| --- | --- |
| Термин | Значение |
| Босс | Особенно сильный и сложный противник, который обычно появляется в ключевых моментах игрового процесса, таких как завершение уровня. |
| OS | Операционная система |
| Стриминг | Технология передачи данных в реальном времени, позволяющая воспроизводить аудио, видео или другой контент на устройстве пользователя без необходимости полной загрузки. |
| Speedrun | Прохождение видеоигры или её части с максимально возможной скоростью, обычно с целью установить рекорд времени. |
| Стример | Человек, который ведёт прямые трансляции (стримы) через интернет, демонстрируя игровую сессию. |
| СPU | Центральный процессор |
| RAM | Оперативная память |
| SSD | Твердотельное запоминающее устройство, которое использует интегральные схемы для постоянного хранения данных |
| Ячейка сохранения | Это отдельная область данных, в которую сохраняется текущее состояние игры. Каждая ячейка позволяет игроку начать, продолжить или перезаписать своё прохождение, не затрагивая другие сохранения. |
| HP | Игровая переменная, отражающая количество «жизней» или выносливости персонажа |
| HUD | Пользовательский интерфейс, отображающий на экране важную информацию, такую как HP, очки, таймер и т. д. Не является частью игрового мира. |
| MonoBehaviour | Базовый класс Unity, от которого наследуются все пользовательские скрипты |
| RigitBody2D | Компонент, отвечающий за физику 2D-объекта |
| Collider2D | Компонент, определяющий форму и зону столкновений объекта. |
| Префаб | Сохранённый шаблон игрового объекта, который можно многократно создавать в сцене. |
| Спавниться | Термин, обозначающий создание нового экземпляра объекта в сцене во время игры |
| WaitForSecond | Метод Unity, позволяющий приостановить выполнение корутины на определённое время |
| Transform | Компонент, содержащий информацию о положении, повороте и масштабе объекта в сцене. |
| GameObject | Базовый элемент Unity, представляющий любой объект в сцене |
| PlatformEffector2D | Компонент, позволяющий создавать платформы, через которые можно прыгать снизу вверх, но стоять сверху |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

# **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**