Содержание

[1. ВВЕДЕНИЕ 4](#_Toc162979484)

[1.1. Наименование программы 4](#_Toc162979485)

[1.2. Краткая характеристика и область назначения 4](#_Toc162979486)

[1.3. Документы, на основании которых ведется разработка 4](#_Toc162979487)

[1.4. Наименование темы разработки 4](#_Toc162979488)

[2. НАЗНАЧЕНИЕ РАЗРАБОТКИ 5](#_Toc162979489)

[2.1. Функциональное назначение 5](#_Toc162979490)

[2.2. Эксплуатационное назначение 5](#_Toc162979491)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 6](#_Toc162979492)

[3.1. Постановка задачи на разработку программы 6](#_Toc162979493)

[Описание применяемых математических методов 6](#_Toc162979494)

[3.2. Требования к составу выполняемых функций 6](#_Toc162979495)

[3.3. Описание функционирования разрабатываемого программного продукта 7](#_Toc162979496)

[3.4. Описание противников, присутствующих в игре 7](#_Toc162979497)

[3.5. Описание игровых механик 8](#_Toc162979498)

[3.5.1. Нанесение урона противникам 8](#_Toc162979499)

[3.5.2. Получение урона игроком 8](#_Toc162979500)

[3.5.3. Особая способность игрока 9](#_Toc162979501)

[3.5.4. Обнаружение игрока 9](#_Toc162979502)

[3.5.5. Система инвентаря 9](#_Toc162979503)

[3.5.6. Описание меню 10](#_Toc162979504)

[3.6. Описание программных средств, используемых программой 10](#_Toc162979505)

[3.7. Описание способа организации входных и выходных данных 10](#_Toc162979506)

[3.8. Требования к составу и параметрам технических средств 11](#_Toc162979507)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 11](#_Toc162979508)

[4.1. Ориентировочная экономическая эффективность 11](#_Toc162979509)

[4.2. Предполагаемая потребность 11](#_Toc162979510)

[4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами 11](#_Toc162979511)

[5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 13](#_Toc162979512)

[ПРИЛОЖЕНИЕ 1 14](#_Toc162979513)

[ПРИЛОЖЕНИЕ 2 15](#_Toc162979514)

[ПРИЛОЖЕНИЕ 3 18](#_Toc162979515)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 22](#_Toc162979516)

# ВВЕДЕНИЕ

# Наименование программы

Наименование темы разработки: «Игра-платформер на Unity ‘‘Кактусовая бойня’’»

Наименование темы разработки на английском языке:«Platformer game on Unity “Cactus carnage”»

# Краткая характеристика и область назначения

«Кактусовая бойня» - игра-платформер, в которой игроку предстоит взять управление над персонажем, сражающимся против армии разумных кактусов, перемещаясь по различным игровым уровням.

# Документы, на основании которых ведется разработка

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

# Наименование темы разработки

Наименование темы разработки – «Игра-платформер на Unity “Кактусовая бойня”».

Условное обозначение темы разработки – «Игра-платормер».

Разработка ведется в рамках выполнения курсового проекта направления специальности Программная инженерия.

**Организация утвердившая данный документ:** Национальный исследовательский университет «Высшая школа экономики»

# НАЗНАЧЕНИЕ РАЗРАБОТКИ

# Функциональное назначение

Игра-платформер “Кактусовая бойня” представляет собой однопользовательскую 2D игру. Её игровой процесс состоит из прохождения ряда уровней по мере продвижения по сюжету. Игроку предстоит сражаться против разумных кактусов с помощью арсенала оружия.

# Эксплуатационное назначение

Игра-платформер предназначена для досуга и развлечения.

Основными конечными потребителями разрабатываемой игры являются люди широкого возрастного спектра (от 13 до 70 лет), заинтересованные в прохождении игр жанра “платформер”.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

# Постановка задачи на разработку программы

# Описание применяемых математических методов

Применяемые математические методы включают в себя базовые преобразования векторов в евклидовом пространстве: сложение, вычитание, умножение на число – данные операции являются неотъемлемой частью для компьютерной 2D игры.

Для создания равномерных звуковых и визуальных переходов используется несколько видов интерполяции:

* Линейная интерполяция: используется для вычисления промежуточных значений между векторами и цветами
* Интерполяция с помощью степенной функции: используется для создания равномерных переходов между различными уровнями звука и преобразований значений яркости игровой сцены.

Использование вышеупомянутых математических методов не накладывает дополнительных ограничений в области их применения. Программная реализация данных методов выполнена с использованием встроенных возможностей игрового движка Unity.

# Требования к составу выполняемых функций

Игра-платформер “Кактусовая бойня” должна быть выполнена с использованием системы уровней: игровой процесс должен происходить на отдельных локациях. Должна быть реализована система смены сцен в игре. Такой подход к построению игры позволяет пользователю с удобством ориентироваться между игровыми локациями и расширяет возможности для повторного прохождения игры или определённых её частей.

Игровой процесс пользователя состоит из последовательного прохождения уровней. Игровые уровни должны становиться доступными по мере прохождения игры – очередной уровень становится доступен пользователю только при условии прохождения уровней, предшествующих данному. **(LevelManger.cs скрипт)**

Программа должна определять условия победы / поражения игрока и загружать необходимые сцены при выполнении данных условий **(скрипты LevelController.cs и Scenes.cs).**

В игре должна быть реализована система поднимаемых объектов. **(Скрипт PickUpSystem.cs)**

**Cкрипт InventoryController.cs:**

В игре должна быть реализована система инвентаря, посредством которой игрок должен иметь возможность взаимодействовать с объектами. Система инвентаря должна быть связана с панелями быстрого доступа.

Система взаимодействия с игровыми предметами должна предоставлять пользователю следующий функционал:

* Удаление предмета из инвентаря
* Помещение предмета в слот быстрого доступа
* Использование предметов выделенных категорий
* Экипировка предмета вооружения в отведённую для него ячейку

# Описание функционирования разрабатываемого программного продукта

Описание игрового процесса

Программа должна определять необходимую для загрузки сцену, для того, чтобы у игрока была возможность перехода на нужную сцену без необходимости явно указывать номер сцены **(скрипт LevelManager.cs)**. В основе игрового процесса лежит управление персонажем, который сражается с разумными кактусами.

Цель игрока – достичь конечной точки уровня, путём перемещения по элементам ландшафта уровня и платформам, которые расположены в воздухе. Должен быть создан скрипт **(PlayerMovement.cs),** отвечающий за перемещение персонажа.

По мере прохождения уровня игроку встречаются различные предметы, которые он может подобрать и использовать. Набор предметов, которые игрок может найти во время прохождения, включает в себя лечащие предметы, оружие ближнего и дальнего боя. В программе должны быть реализованы скрипты и ассеты для данных типов предметов: **PickableItem.cs, EdibleItemAsset.cs, MeleeWeaponAsset.cs, RangedWeaponAsset.cs**

Для этих предметов должны быть созданы префабы – они будут упрощать создание однотипных объектов.

# Описание противников, присутствующих в игре

Противники в игре представляют собой разумных кактусов, которые по сюжету игры враждуют с главным персонажем.

В игровых сценах игроку предстоит встретиться как минимум с тремя видами противников:

1. Обычный кактус. Поведение данного кактуса должно определяться скриптами **BasicEnemyMovement.cs** – передвижение кактуса и проверка достижимости игрока, **AttackOnProximity.cs** – атака.
2. Кактус-патрульный. Патрулирует заданную территорию: за патрулирование территории отвечает скрипт перемещения **(PatrolCactusMovement.cs)** – патрулирование территории подразумевает перемещение от одной заранее установленной точки до другой и обратно. За атаку у кактуса-патрульного, как и у обычного кактуса, отвечает скрипт **AttackOnProximity.cs**
3. Катящийся кактус. Небольшой кактус круглой формы. Поведение данного кактуса определяется системой состояний. Система состояний должна быть реализована в виде связки из нескольких компонент: перечисления **(RollingEnemyState.cs)**, отвечающего за состояние кактуса, контроллера **(RollingEnemyController.cs)**, реализующего логику смены состояний, а также скриптов, отвечающих за перемещение **(RollingEnemyMovement.cs)** и атаку **(RollingEnemyMeleeAtack.cs)** катящегося кактуса.

# Описание игровых механик

# Нанесение урона противникам

В игре должны быть реализованы две основные механики для нанесения урона противникам, за работу этих механик должен отвечать отдельных контроллер атаки игрока **(AttackController.cs)**:

* Атака в ближнем бою. Делится на два подвида:
  + Атака врукопашную
  + Атака с помощью оружия ближнего боя
* Дальняя атака.

Результатом совершения успешной дальней атаки должно являться появление нового объекта – снаряда, который существует независимо от оружия и контроллера, которые его создали. Поведение снаряда должно определяться скриптом **Projectile.cs**

Контроллер атаки также должен быть ответственным за обработку всех вспомогательных действий, связанных с атакой, так **AttackController.cs** управляет вспомогательным объектом, используемым для прицеливания.

# Получение урона игроком

Механика получения урона игроком, хотя и представлена в отличающихся видах, фундаментально сводится к взаимодействию игрока с объектами, которые могут наносить урон. То есть, для того чтобы персонаж получил урон, необходимо физическое столкновение с объектом, способным наносить урон. Урон способны наносить несколько видов объектов:

* Противники (о которых говорилось ранее). Наносят урон с использованием скриптов **AttackOnProximity.cs, RollingEnemyMeleeAttack.cs**
* Препятствия. Например, шипы, расположенные в различных участках уровня и не изменяющие своего положения в течение игры. Атака, совершаемая данным видом объектов должна находиться под контролем скрипта **Obstacle.cs**
* Пропасть. Падение персонажа в пропасть должно обрабатываться с помощью скрипта **Void.cs**

# Особая способность игрока

В игре должна быть реализована механика особой способности. Все действия, условия и события, связанные с особой способностью игрока должны обрабатываться скриптом, контролирующим состояние персонажа – **PlayerState.cs**

# Обнаружение игрока

Для обнаружения игрока противники, используют метод трассировки лучей. Применение данного метода заключается в испускании луча в определённую сторону и поиске объектов, которые встретил данный луч на своём пути.

# Система инвентаря

Должен быть реализован контроллер инвентаря **(InventoryController.cs)**. Система инвентаря позволяет сохранять и использовать объекты. Объект, который находится под контролем системы инвентаря должен иметь на себе скрипт **PickableItem.cs**, если он находится непосредственно на игровой локации или скрипт **InventoryItem.cs,** если он уже находится в инвентаре. Инвентарь должен предоставлять описание выбранного предмета, осуществлять контроль за назначением предмета в ячейку быстрого доступа **(с помощью контроллера HotbarController.cs)**, использованием предметов и выбрасыванием предметов.

Инвентарь должен поддерживать систему стаков, то есть несколько предметов могут находиться в одной ячейке инвентаря, если предмет данного типа позволяет это. Так, в одной ячейке инвентаря может находится до 4 исцеляющих зелий, но в одной ячейке инвентаря не может быть больше одного пистолета или мачете, потому что данные предметы не могут образовывать стак. Информация о том, какие предметы могу образовывать стак, о размере стака и о действии, которое будет выполнено при использовании предмета, должен хранить специальный объект, который уникален для каждого типа предметов – ассет (любой наследник абстрактного класса **ItemAsset.cs**).

Система инвентаря должна предоставлять интерфейс для UI, позволяющий обрабатывать закрытие и открытие инвентаря с привязкой к определённым действиям.

# Описание меню

Программа должна предоставлять интерфейс, позволяющий корректно обрабатывать несколько различных панелей меню.

* + - 1. **Меню настроек**

Меню настроек позволяет пользователю изменять ряд параметров, таких как общий уровень громкости звука, уровень громкости музыки и звуковых эффектов, уровень яркости сцены. За изменением настроек следит отдельный скрипт – **SettingsMenuManager.cs**

* + - 1. **Меню выбора уровней**

В данном меню пользователю предоставляется возможность выбора уровня из набора доступных. Набор доступных уровней определяется скриптом, контролирующим изменения состояний уровней – скриптом **LevelManager.cs**. Каждый уровень может находиться в нескольких состояниях, эти состояния определяют доступность уровней для прохождения.

# Описание программных средств, используемых программой

Игра-платформер “Кактусовая бойня” написана на движке Unity. Все скрипты написаны на языке C#. Выбор данного игрового движка обусловлен его широкой поддержкой в среде разработчиков, удобством при разработке 2D игр и обилием обучающих материалов в интернете. Также Unity представляет широкие возможности для кросс-платформенной разработки, что может быть удобно, если возникнет необходимость портировать игру на другую платформу. Выбор языка программирования обусловлен привязкой его к игровому движку.

# Описание способа организации входных и выходных данных

Программа обрабатывает пользовательские входные данные двумя ключевыми способами:

* Непосредственное получение ввода от пользователя через периферийные устройства ввода: клавиатуру и компьютерную мышь
* Получение сохранённых настроек пользователя и данных о прохождении игры. Эти данные программа получает с помощью инструмента, встроенного в сам движок Unity – PlayerPrefs. PlayerPrefs позволяет сохранять простые данные локально на устройстве пользователя. Такой способ сохранения данных удобен своей простотой в использовании, но он не подходит для хранения сложных структур данных и конфиденциальной информации. Разрабатываемой программе достаточно хранить лишь простые данные, поэтому данный способ хранения данных хорошо подходит для нашей задачи.

Выходные данные поступают к пользователю преимущественно в виде аудио-визуальной информации через встроенные в Unity инструменты взаимодействия с пользователем.

# Требования к составу и параметрам технических средств

Для успешного использования программы пользователю необходимо иметь в распоряжении стандартные периферийные устройства ввода – клавиатуру и компьютерную мышь.

Минимальные системные требования:

1. Процессор: Intel i5­-4590 / AMD Ryzen 5 1500X или лучше;
2. Видеокарта: NVIDIA GeForce GTX 470 / Radeon RX 460 или лучше;
3. Оперативная память: 8 GB.

# ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

# Ориентировочная экономическая эффективность

Данный курсовой проект не предусматривает расчет экономической эффективности.

# Предполагаемая потребность

Данное приложение будет востребовано среди людей, заинтересованных в компьютерных играх, в частности в жанре “платформер”.

# Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

В представленной таблице приведена сравнительная характеристика с играми, близкими по тематике к разрабатываемому проекту.

Таблица 1 – Сравнительная характеристика

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Игра  Пункт сравнения | Кактусовая бойня | Super Mario Odyssey | Hollow Knight | Cuphead | Sonic Mania | Dead Cells |
| Поддержка многопользовательского режима игры | **-** | **+** | **-** | **+** | **-** | **-** |
| Поддержка игры на Windows 10 | **+** | **-** | **+** | **+** | **+** | **+** |
| Поддержка игры на macOS | **-** | **-** | **+** | **+** | **-** | **+** |
| Наличие нескольких локаций | **+** | **+** | **+** | **-** | **+** | **-** |
| Наличие системы инвентаря | **+** | **-** | **+** | **-** | **-** | **+** |
| Доступность в России | **+** | **-** | **+** | **+** | **-** | **+** |
| Наличие сюжета | **+** | **+** | **+** | **+** | **+** | **+** |

# ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.103-77 Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.603-78 Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. Официальное руководство по движку Unity [Электронный ресурс] / Unity User Manual. Режим доступа: <https://docs.unity3d.com/Manual/> (дата обращения 15.03.2024)
10. Unity 2D Platformer for Complete Beginners [Электронный ресурс] / Youtube. Режим доступа <https://www.youtube.com/watch?v=TcranVQUQ5U&list=PLgOEwFbvGm5o8hayFB6skAfa8Z-mw4dPV&pp=iAQB> (дата обращения 16.03.2024)
11. Официальная документация по языку C# [Электронный ресурс] / Youtube. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения 27.01.2024)
12. Язык программирования C# и платформа .NET [Электронный ресурс] / Youtube. Режим доступа: <https://metanit.com/sharp/> (дата обращения 5.02.2024)

# ПРИЛОЖЕНИЕ 1

**Терминология**

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| Интерполяция | Метод восстановления (обычно приближённого) функции по значениям самой функции. |
| Игровая механика | Набор правил и способов, реализующий определённым образом некоторую часть интерактивного взаимодействия игрока и игры. |
| Игровой движок | Комплекс программных средств (ПО), обеспечивающих работу графических, звуковых, геймплейных и других элементов видеоигры. |
| Кроссплатформенность | Способность программного обеспечения работать с несколькими аппаратными платформами или операционными системами. |
| Евклидово пространство | Пространство, свойства которого описываются аксиомами евклидовой геометрии. В более общем смысле евклидово пространство – конечномерное действительное векторное пространство Rn со скалярным произведением (x, y), x, y∈Rn |
| Ассет | Цифровой объект, преимущественно состоящий из однотипных данных, неделимая сущность, которая представляет часть игрового контента и обладает некими свойствами. |

# ПРИЛОЖЕНИЕ 2

**Описание и функциональное назначение классов программы.**

|  |  |
| --- | --- |
| Класс | Функциональное назначение и описание |
| Контроллеры | |
| AttackController.cs | Осуществляет контроль за всеми действиями ближней атаки, совершаемыми игроком. Контролирует создание объектов снарядов при дальних атаках. Осуществляет контроль за прицеливанием. |
| HotbarController.cs | Контролирует слот быстрого доступа. |
| InventoryController.cs | Контролирует объекты в инвентаре: их использование, перемещение, выбрасывание. |
| LevelController.cs | Контролирует условие поражения. Контролирует визуальную постобработку уровня. |
| WeaponController.cs | Контролирует активное оружие |
| Скрипты, отвечающие за поведение врагов | |
| BasicEnemyMovement.cs | Контролирует перемещение обычных кактусов. |
| PatrolCactusMovement.cs | Контролирует перемещение патрулирующих кактусов. |
| RollingEnemyController.cs | Контролирует изменение состояний катящегося кактуса. Проверяет возможность обнаружения персонажа кактусом. |
| RollingEnemyMovement.cs | Контролирует перемещение катящегося кактуса. |
| RollingEnemyState.cs | Перечисление возможных состояний катящегося кактуса. |
| AttackOnProximity.cs | Скрипт, отвечающий за атаку игрока на основании физического приближения к нему. |
| EnemyHealth.cs | Скрипт, осуществляющий контроль за здоровьем кактусов. |
| Объекты окружения | |
| Obstacle.cs | Скрипт статичного препятствия, наносящего урон при касании. |
| Void.cs | Скрипт, который контролирует поведение объектов попавших в пустоту под картой. |
| Система подбираемых предметов. | |
| PickableItem.cs | Представляет собой объект, который может быть поднят игроком и помещён в инвентарь. |
| PickUpSystem.cs | Осуществляет контроль за поднимаемыми предметами – добавляет их в инвентарь игрока. |
| Поведение персонажа | |
| PlayerInputActions.cs | Автогенерируемый скрипт новой системы управления Unity. |
| PlayerMovement.cs | Скрипт отвечающий за передвижение персонажа. |
| PlayerState.cs | Скрипт, собирающий в себе информацию о состоянии персонажа – его здоровье, очках особой способности. |
| Контроль за игрой | |
| LevelManger.cs | Осуществляет загрузку уровней. |
| Модели | |
| ItemParameterAsset.cs | Хранит в себе информацию об имени предмета |
| CharacterStatHealthModifier.cs | Класс улучшения, увеличивающего здоровья |
| CharacterStatModifierAsset.cs | Абстрактный класс – предоставляет базис для дочерних модификаторов |
| InventoryItemModel.cs | Модель предмета в инвентаре. |
| Скриптовый объекты (Scriptable objects) | |
| PlayerDetectionAsset.cs | Хранит в себе информацию о способе поиска пути до персонажа. |
| RollingEnemyAsset.cs | Хранит в себе настройки катящегося кактуса. |
| Rotation.cs | Перечисление направлений вращения |
| EdibleItemAsset.cs | Ассет предмета, который можно употребить в пищу |
| IDestroyableItem.cs | Пустой интерфейс, сигнализирующий о том, что данный объект может быть выкинут из инвентаря |
| IItemAction.cs | Интерфейс для действия, которое происходит с объектом инвентаря. |
| InventoryAsset.cs | Ассет инвентаря, хранит в себе общие данные об инвентаре. |
| ItemAsset.cs | Ассет предмета. |
| ItemParameter.cs | Класс, представляющий информацию о параметре предмета. |
| MeleeWeaponAsset.cs | Ассет оружия ближнего боя. |
| RangedWeaponAsset.cs | Ассет оружия дальнего боя. |
| WeaponAsset.cs | Абстрактный класс – базис для специализированных ассетов. |
| Графический интерфейс пользователя | |
| InventoryItem.cs | Визуальное представление предмета в инвентаре |
| InventoryPanel.cs | Скрипт, контролирующий панель инвентаря |
| ItemActionPanel.cs | Скрипт, контролирующий панель действий в инвентаре. |
| ItemDescription.cs | Скрипт, контролирующий информационной панель инвентаря. |
| ItemDragHandler.cs | Скрипт, управляющий вспомогательным предметов инвентаря, использующимся для реализации механики drag and drop |

# ПРИЛОЖЕНИЕ 3

**Описание и функциональное назначение методов контроллеров программ.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс контроллера | Имя метода | Аргументы метода | Возвращаемое значение | Описание и функциональное назначение |
| AttackController | AimBoxFollowMouth | - | void | Изменяет положение вспомогательного объекта для прицеливания |
| HandleRightMouseButtonUp | - | void | Обработка прекращения нажатия правой кнопки мыши |
| CancelWeaponPreparation | - | void | Отмена подготовки оружия |
| HandleRightMouseButtonDown | - | void | Обработка нажатия правой кнопки мыши |
| HandleLeftMouseButtonDown | - | void | Обработка нажатия левой кнопки мыши |
| Attack | - | void | Выбор подходящего способа атаки в зависимости от текущего оружия. Вызов атакующего метода |
| AttackRanged | RangedWeaponAset | void | Проведение дальней атаки. Создается снаряд |
| AttackMelee | MeleeWeaponAsset | void | Проведение ближней атаки |
| BareHandedAttack | - | void | Атака руками (если не выбрано оружие) |
| IsEnemyInAttackBox | float, out GameObject | bool | Проверка на наличие противника в заданной квадратной области |
| PrepareWeapon | float | IEnumarator | Подготовка оружия к атаке |
| AttackCooldown | float | IEnumarator | Запуск ожидания восстановления оружия |
| HotbarController | UpdateCurrentlyDraggedId | int | void | Установить id текущей ячейки, которую перетягивают |
| SubcribeItem | InventoryItem | void | Подписать элемент инвентаря с указанным id на события |
| PerformAction | InventoryItem | void | Выполнить действие на которое ссылается ячейка быстрого доступа с указанным индексом |
| SetTrackedItem | InventoryItem | void | Установить отслеживаемую ячейку для слота быстрого доступа с указанным индексом |
| Inventory controller | SetUpInventory | - | void | Заполнение инвентаря стартовыми предметами |
| UpdateInventory | Dictionary<int, InventoryItemModel> | void | Обновление всех ячеек инвентаря |
| SetUpController | - | void | Инициализация панели инвентаря |
| Subscribe | - | void | Подписка панели инвентаря на события |
| HandleItemActionRequest | int | void | Обработка запроса действия предмета |
| PerformAction | int | void | Выполнение действия присвоенного предмету с указанным индексом |
| DropItem | int, int | void | Выбрасывание указанного количества предмета из инвентаря |
| HandleDragging | int | void | Обработка перетягивания предмета в инвентаре |
| HandleItemSwap | int, int | void | Обмен местами двух указанных ячеек инвентаря |
| HandleDescriptionRequested | Int | void | Обработка запроса на просмотр описания предмета |
| PrepareDesription | InventoryItemModel | string | Подготовка строкового представления описания предмета |
| LevelController | Subscribe | - | void | Подписка на обновление значения яркости и обновления здоровья игрока |
| CheckLoseCondition | int | void | Проверка условия поражения |
| SetUpLevelVisuals | - | void | Загрузка настроек яркости, установка яркости |
| LoadBrightness | - | void | Загрузка настроек яркости |
| WeaponController | SetWeapon | WeaponAsset, List<ItemParameter> | void | Установка активного оружия |
| UpdateActiveWeapon | - | void | Обновление текущего активного оружия |
| ModifyParameters | - | void | Обновление параметров оружия |

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

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