**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**"ВЫСШАЯ ШКОЛА ЭКОНОМИКИ"**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СОГЛАСОВАНО  Научный сотрудник МЛ ИССА факультета компьютерных наук, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.В. Максименкова  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  проф, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. |  | | **КОМПЬЮТЕРНАЯ ИГРА ВДОХНОВЛЁННАЯ LEGENDS OF ZELDA  Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 81 01-1-ЛУ**  Исполнитель  Студент группы БПИ191  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Д.П.Архаров /  "\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | | | | |
|  | |  | | |

**УТВЕРЖДЕН**

**RU.17701729.04.01-01 81 01-1-ЛУ**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**КОМПЬЮТЕРНАЯ ИГРА ВДОХНОВЛЁННАЯ LEGENDS OF ZELDA  
  
Пояснительная записка  
  
RU.17701729.04.01-01 81 01-1-ЛУ**

**Листов 40**

СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 3](#_Toc40603952)

[1.1. Наименование программы 3](#_Toc40603953)

[1.2. Документы, на основании которых ведется разработка 3](#_Toc40603954)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc40603955)

[2.1. Назначение программы 4](#_Toc40603956)

[2.1.1. Функциональное назначение 4](#_Toc40603957)

[2.1.2. Эксплуатационное назначение 4](#_Toc40603958)

[2.2. Краткая характеристика и области применения 4](#_Toc40603959)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5](#_Toc40603960)

[3.1. Постановка задачи на разработку программы 5](#_Toc40603961)

[3.2. Описание алгоритма и функцинирования программы 5](#_Toc40603962)

[3.2.1. Описание общего игрового процесса 5](#_Toc40603963)

[3.2.2. Описание функционирования персонажа игрока 6](#_Toc40603964)

[3.2.3. Описание функционирования UI (User Interface) 12](#_Toc40603965)

[3.2.4. Описание функционирования противников 14](#_Toc40603966)

[3.2.5. Описание функционирования объектов доступных для взаимодействия 16](#_Toc40603967)

[3.2.6. Описание функционирования переходов между уровнями. 17](#_Toc40603968)

[3.3. Описание и обоснование выбора метода организации входных и выходных данных. 18](#_Toc40603969)

[3.4. Описание и обоснование выбора состава технических и программных средств 18](#_Toc40603970)

[3.4.1. Состав технических и программных средств 18](#_Toc40603971)

[3.4.2. Обоснование выбора технических и программных средств. 18](#_Toc40603972)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 20](#_Toc40603973)

[4.1. Предполагаемая потребность 20](#_Toc40603974)

[4.2. Ориентировочная экономическая эффективность 20](#_Toc40603975)

[4.3. Преимущества разработки по сравнению с отечественными или зарубежными аналогами 20](#_Toc40603976)

[ПРИЛОЖЕНИЕ 1 21](#_Toc40603977)

[ПРИЛОЖЕНИЕ 2 22](#_Toc40603978)

[ПРИЛОЖЕНИЕ 3 23](#_Toc40603979)

[ПРИЛОЖЕНИЕ 4 26](#_Toc40603980)

1. ВВЕДЕНИЕ
   1. Наименование программы

Наименование программы «Компьютерная игра вдохновлённая Legends of Zelda».

Наименование программы на английском языке – «PC Game Based on Legends of Zelda».

Наименование программы для пользователя – «Firefly».

* 1. Документы, на основании которых ведется разработка

Приказ декана факультета компьютерных наук И.В. Аржанцева "Об утверждении тем, руководителей курсовых работ студентов образовательной программы «Программная инженерия» факультета компьютерных наук" № 2.3-02/1112-04 от 11.12.2019

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. Назначение программы
      1. Функциональное назначение

Данная программа представляет собой систему, обеспечивающая доступ к геймплею игры. Пользователю предоставлен доступ к различным компонентам системы, представляющими собой элементы управления. Используя предоставленные элементы управления, пользователь может играть в игру. Игра в первую очередь вдохновленна легендарной серией видеоигр Legend of Zelda, поэтому направлена быть узнаваемой и схожей с ней, при этом содержа внутри себя интересный геймплей утоляющий потребность в развлечениях и исследовании внутриигрового мира у пользователя.

* + 1. Эксплуатационное назначение

Программа может эксплуатироваться при наличии PC на базе OC Windows. Никакой специализированной квалификации и подготовки от пользователя не требуется. Программа будет предоставлять уровни которые пользователь будет иметь возможность исследовать и проходить.

* 1. Краткая характеристика и области применения

“Firefly” предназначена для широкой области применения, среди пользователей испытывающих потребность в развлечениях, интересном досуге. Конечной целевой аудиторией предполагаются люди любящие видеоигры, в частности фанаты серии Legend of Zelda.

1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ
   1. Постановка задачи на разработку программы

Программа должна будет реализовывать следующий перечень задач :

- Соответствие функциональным характеристикам, подробнее описанным в Техническом задании («Компьютерная игра вдохновлённая Legend of Zelda». Техническое задание);

- Соответствие требованиям к интерфейсу, подробнее описанным в Техническом задании («Компьютерная игра вдохновлённая Legend of Zelda». Техническое задание);

- Соответствие требованиям к надёжности, подробнее описанным в Техническом задании («Компьютерная игра вдохновлённая Legend of Zelda». Техническое задание).

* 1. Описание алгоритма и функцинирования программы
     1. Описание общего игрового процесса

Перед пользователем предстаёт мир, разбитый на прямоугольные комнаты, по анналогии с устройством мира в классических играх серии Legend of Zelda. Игрок управляет персонажем, имея возможности для перемещения, атаки противника, набора очков, а так же контактирования с доступными для взаимодействия объектами.

В игре представлено 3 игровых уровня. Пользователю предлагается исследовать их, побеждая там противников и отыскивая монеты. Монеты, как эквивалент внутриигровых набираемых очков, будут сохранятся при переходе между уровнями. Сам переход может осуществляться игроком самостоятельно в любое время.

В ходе игры игроку предстоит столкнутся с тремя типами противников, которые будут атаковать его. Также он сможет собирать монеты – эквивалент набора очков, который показывает степень исследованности мира игроком. Кроме того предусмотрены разрушаемые объекты, которые будут разбиваться после атаки игрока по ним.

Для того, чтобы пройти игру пользователю нужно отыскать ключ от подземелья, пройдя по пути из монет, а после победить всех противников в комнате подземелья.

* + 1. Описание функционирования персонажа игрока

Игроку предоставлен для управления персонаж, в список возможностей которого входит:

1) Перемещение по миру игры. Осуществляется с помощью либо комбинации клавиш “WASD”, либо клавиш стрелок (рис. 1).



*Рисунок 1 Перемещение по миру.*

2) Атака противника. Осуществляется с помощью клавишы space (рис. 2).



*Рисунок 2 Атака противника.*

3) Разрушение объекта, доступного для такого действия. Осуществляется с помощью клавишы space (рис. 3).



*Рисунок 3 Разрушение объекта.*

4) Взаимодействие с доступным для этого объектом. Перечень таких: Табличка с текстом,Сундук, Дверь, Переключатель. Осуществляется при помощи клавишы space (рис. 4)

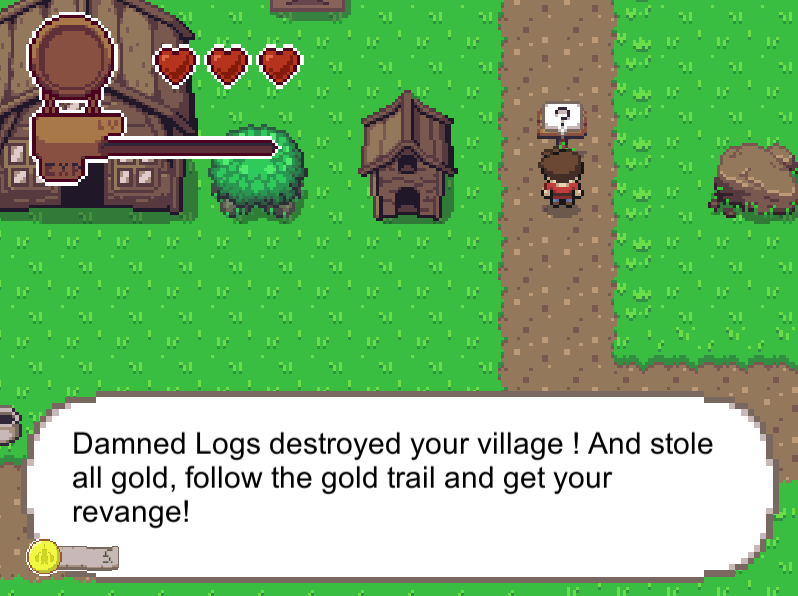


Рисунок 4 Взаимодействие с объектом

Разберём действия подробнее с внутриигровой точки зрения. Начнём с рассмотрения самого персонажа игрока. Он состоит из эллемента типа Transform, предоставляемого Unity, который поддерживает внутри положение объекта, эллемента типа Sprite Renderer, который осуществляет поддержание визуального представления объекта, двух эллементов типов Box Collider 2D, которые отвечают за коллизии происходящие между персонажем и миром(один для регистрации попаданий по персонажу, другой отвечает за коллизии с различными объектами, например внешними стенами комнат). А также эллемента типа Rigidbody 2D, представляющего собой физическое тело персонажа, отвечающего за всю физику происходящую с ним, и, наконец, эллемента типа Animator, предоставляющего доступ к смене анимаций во время осуществления разных действий. Для того, чтобы корректно выбирать анимации для проигрывания, легко менять управление персонажем, а так же упрощения алгоритмов, была разработана машина положений игрока, представляющая собой enum текущее состояние которого показывает состояние персонажа игрока. Кроме того у игрока есть инвентарь, способный содержать монеты и ключи, содержимое которого поддерживается между разными уровнями игры. Теперь, с пониманием игровых внутренностей нашего персонажа перейдём к описанию алгоритмов связанных с выше обозначенными действиями, доступными игроку.

**Алгоритм перемещения по миру** может быть легко описан так: Начало, проверка возможности для перемещения(игрок не может передвигаться когда находится в состоянии атаки, либо в состоянии отбрасывания), определение позиции, в которую хочет переместится игрок, вычисление того, по какой оси нужно будет осуществить перемещение, придание Rigidbody 2D перемещения, в соответствии с текущими параметрами скорости игрока, выбор соответствующей анимации для проигрывания, Конец.

**Алгоритм атаки противника** в свою очередь специально был создан так, чтобы соответствовать идейному вдохновителю Firefly – Legend of Zelda. Выражено это в форме того, что игрок не может одновременно перемещаться и атаковать, соответственно алгоритм атаки будет выглядеть так: Начало, проверка того, что атака возможна, инициализация соответствующего направлению атаки HitBox, представленного Polygon Colider 2D, отвечающего за коллизии вызывающие нанесение удара по цели, проигрывание соответствующих анимаций, Конец. Атака может повлечь за собой 3 результата : Нанесение урона или Отбрасывание, в случае атаки противника, либо Разрушение доступного для этого предмета. Для проигрывания анимации атаки, был использован Coroutine[8] т.е. параллельно запускаемая функция) для обеспечения симуляции нахождения внутри момента атаки, и последующего выхода из него. У персонажа игрока есть 6 очков здоровья выраженные в 3 сердцах, располагающихся на экране, по истечению которых он будет уничтожен, кроме того его можно отталкивать.

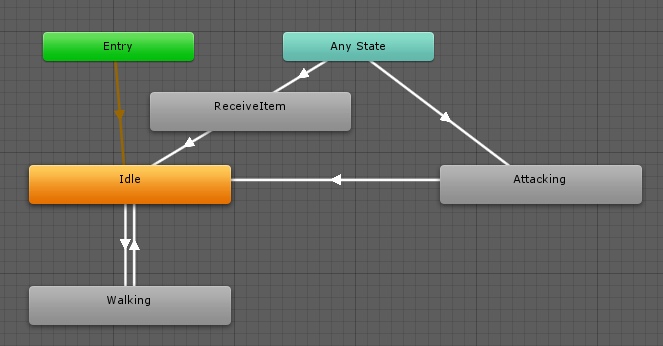
Наконец, **Алгоритм контакта с доступными для взаимодействия объектами**. Все такие объекты имеют Box Colider 2D, которые проверяют то, что пользователь находится в определённом радиусе рядом с объектом. Когда пользователь находится в радиусe, над его головой появляется индикатор того, что он может взаимодействовать с предметом (рис. 5).



*Рисунок 5 Индикатор доступности взаимодействия*

Находясь в радиусе, пользователь может провзаимодействовать с объектом, что приводит к разному эффекту для 3 разных типов объектов доступных для взаимодействия, что будет подробнее описано позже.

Проигрывание а также взаимодействие анимаций между разными действиями всегда осуществляются с помощью Animator, который обеспечивает плавность и отсутствие конфликтов, работая вместе с машиной положений персонажа (рис. 6)



*Рисунок 6 Аниматор персонажа игрока*

* + 1. Описание функционирования UI (User Interface)

Основная идея, стоящая за осуществлением работы UI стоит в использовании Observer паттерна для настройки взаимодействия между внутриигровыми объектами. Говоря в общем – можно сказать, что большинство внутриигровых взаимодействий были выполнены с использованием подобного паттерна, поскольку в отличии от Singltone взаимодействий, формирующих жёсткие связи между объектами, Observer предоставляет нам гораздо более широкую свободу, позволяя избегать излишних изменений кода при изменении различных эллементов игры, за счёт отсутствия жёстких связей между объектами. Это сильно облегчает работу, задействуя максимум гибкости и возможностей Unity, значительно увеличивая скорость создания новых уровней, поскольку становится достаточно просто перетащить соответствующий объект на сцену в Unity и настроить соответствующие эллементы не меняя кода.

Выражается этот Observer паттерн в наличии двух эллементов : Signal и Signal Listener. Signal это конструкция осуществляющая передачу в список подписанных на приём этого сигнала наблюдателей информации о том, что произошло конкретное событие. Signal Listener это конструкция состоящяя из UnityEvent, который будет отвечать за действия принимающего сигнал эллемента, и соответствующего ему Signal, который будет отвечать за посылку соответствующего сигнала для обрабатывающего наблюдателя. Соответственно наблюдателю не нужно точно знать то, с кем конкретно он работает, ему достаточно иметь способность к приниманию соответствующщего сигнала, что повышает гибкость и позволяет изменять разные эллементы не разрушая общей цепочки взаиомдействия.

Помогают для осуществления этого паттерна а так же Scriptable Objects, эллементы кода, которые находятся вне сцен в Unity, наследуются не от MonoBehavior, а от Scriptable Objects, и соответственно существуют отдельно от сцен, в таких объектах будем хранить информацию, необходимую наблюдателю для обработки и реакции на поступивший ему Signal, например колличество очков здоровья у персонажа игрока.[9]

Суммируя всё выше-сказанное и переходя к конкретному описанию функционирования, наш UI содержит два постоянно показываемых эллемента : показатель очков здоровья персонажа, а также показатель количества монет, собранных игроком (рис. 7)



*Рисунок 7 Постоянно активные показатели здоровья и монет*

А также один не постоянно показываемый – окно диалога, через которое пользователю представляется различного рода информация (рис. 8).



*Рисунок 8 Появляющееся на экране окно диалога*

* + 1. Описание функционирования противников

В игре представлено 2 типа противников, нападающих на игрока. Первый тип самый простой – это ожившее демоническое бревно. Его алгоритм действия предельно прост, он имеет радиус обнаружения, при попадании игрока в радиус обнаружения бревна оно пробуждается (когда игрока нет в радиусе, мы можем наблюдать, что бревно спит), и начинает двигаться к игроку, при соприкосновении с игроком бревно наносит урон и вызывает небольшую встряску камеры, которую мы подробнее рассмотрим в пункте о работе камеры. Если же игрок разрывает дистанцию до противника, то он возвращается ко сну. Все анимации осуществляются с помощью аниматора, высчитывание столкновений с помощью Box Colider 2D. Противник имеет 2 очка здоровья, по опустошению которых будет уничтожен, а так же его можно отталкивать.

Второй тип уже сложнее, Турель представляет собой модификацию первого, это бревно, которое умеет выстреливать в игрока камень. Камень это объект, который создаётся при условии, что игрок находится в радиусе атаки Противником-Турелью, камень не может нанести вреда самим противникам, но при этом он запускается в ту позицию, в которой находился игрок в момент запуска, и летит туда с заданной скоростью. При столкновении с игроком камень нанесёт урон а так же исчезнет. Противник-Турель имеет настраиваемую частоту выстрелов, которая определяет как часто он будет создавать камни, а также имеет 2 очка здоровья, по опустошению которых будет уничтожен, а так же его можно отталкивать.

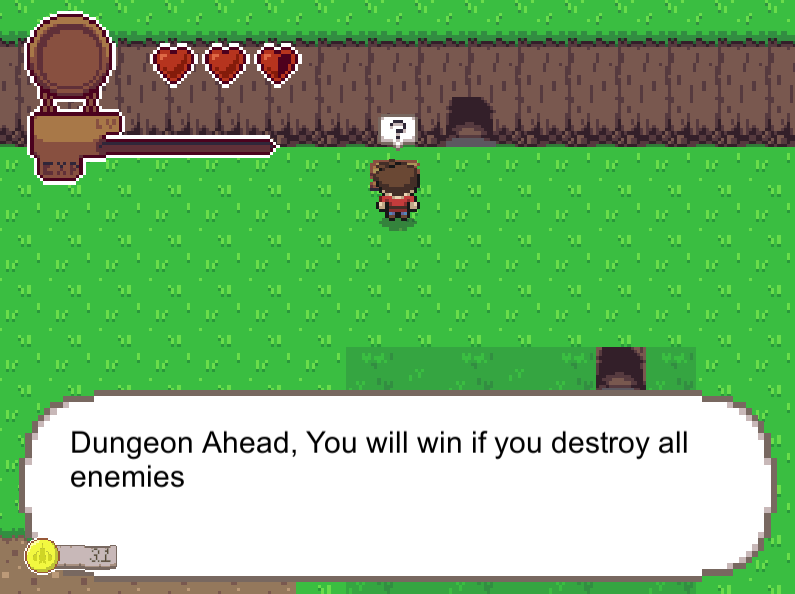
Все противники имеют анимацию смерти (рис. 9), проигрываемую при их уничтожении, а так же специальный объект – Loot Table. Этот объект отвечает за создание предметов, которые игрок потом может подобрать. В игре это могут быть либо восстанавливающие здоровье сердца, либо монеты, которые может подобрать игрок.



*Рисунок 9 Анимация смерти противника*

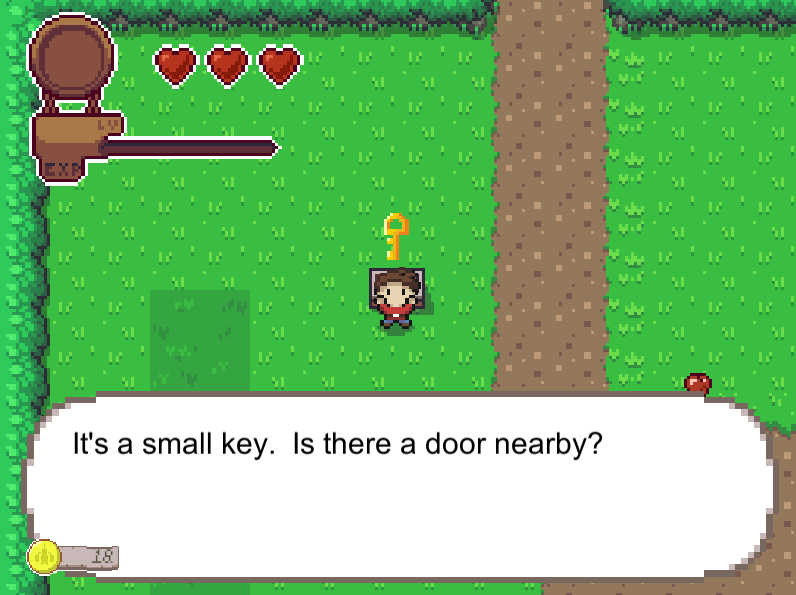
* + 1. Описание функционирования объектов доступных для взаимодействия

В игре пользователю могут встретится 3 вида объектов, доступных для взаимодействия, это табличка, сундук и дверь. Когда игрок будет в зоне доступа к объекту, над ним появится соответствующий индикатор оповещающий о том, что игрок может провзаимодействовать с предметом нажав клавишу space. Объект определяет, что игрок рядом с ним через Box Colider 2D. После взаимодействия с табличкой, откроется окно диалога, в котором игрок увидит надпись, находящуюся на табличке. (рис. 10)



*Рисунок 10 Надпись на табличке*

После взаимодействия с сундуком, игрок получит содержимое этого сундука в свой инвентарь, кроме того будет проиграна анимация получения предмета (рис. 11).



*Рисунок 11 Анимация получения предмета*

Конкретно на данных уровнях – игрок получит ключ. Ключ нужен чтобы провзаимодействовать с последним из доступных объектов – дверью, она откроется только если в момент взаимодействия у игрока будет ключ. Однако помимо двери, открываемой ключом, в игре есть ещё 2 типа дверей, один из них открывается при соприкосновении игрока с кнопкой на полу комнаты, а другой откроется только когда игрок уничтожит всех противников в комнате.

* + 1. Описание функционирования переходов между уровнями.

Во время перехода между 3 доступными в игре уровнями (дом игрока, наружный мир, подземелье) происходит загрузка новой сцены для Юнити. Реализуется это с помощью Coroutine[8] внутри которого видимый пользователем экран заменяется на белую заглушку для загрузки новой сцены. Во время пока на экране заглушка происходит использование AsyncOperation, то есть запуск Coroutine, который подгружает новую сцену, а пока он не закончит мы возвращаем нашим исходным корутином null, то есть пропускаем кадры и держим нашу белую заглушку перед пользователем.

Между разными уровнями за счёт ScriptableObject сохраняется количество монет, собранных пользователем, а вот его уровень здоровья восстанавливается.

* 1. Описание и обоснование выбора метода организации входных и выходных данных.

Входные данные представлены действиями пользователя внутри игры, нажатие им клавиш.

Выходных данных в игре не предусмотрено.

* 1. Описание и обоснование выбора состава технических и программных средств
     1. Состав технических и программных средств

Для корректной работы игры потребуется :

1) Операционная система : Windows 7(последний пакет обновлений).

2) Microsoft .NET Framework 4.7 / более поздняя версия

Список технических средств для корректной работы программы:

1) Клавиатура

2) 1,5 GB свободного места на жёстком диске или больше.

3) Оперативная память: 96 MB или больше.

* + 1. Обоснование выбора технических и программных средств.

Клавиатура потребуется для ввода входных данных и управления программой.

При процессоре ниже указанного процесс игры скорее всего будет затруднён.

Количества оперативной памяти ниже указанного скорее всего будет не хватать для бесперебойной работы игры.

Меньше чем 1,5 Гб места на жёстком диске могут восприпятствовать корректной работе Операционной системы, Microsoft .NET Framework и Firefly.

1. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ
   1. Предполагаемая потребность

“Firefly” – Игра предоставляющая широкие возможности для исследования внутреннего мира а так же для развлечения. Предполагаемая потребность также обусловлена популярностью и известностью серии игр Legend of Zelda, идейного вдохновителя Firefly, однако нынешние игры Legend of Zelda принципиально отличаются от старых, поэтому предполагаемый пользователь может захотеть ознакомится с игрой в силу желания снов ощутить именно тот игровой процесс.

* 1. Ориентировочная экономическая эффективность

В рамках данной курсовой работы не предусмотренно рассчёта экономической эффективности.

* 1. Преимущества разработки по сравнению с отечественными или зарубежными аналогами

Основным преимуществом данной программы безусловно будет являться бесплатное распространение, однако помимо этого есть ещё ряд преимуществ. В сравнении со старыми играми серии Legend of Zelda, данная разработка куда доступнее, в силу поддержки PC. Кроме того, посредством использования Observer паттерна в разработке и средств самой Unity была достигнута сильная модульность системы, позволяющая создавать новые уровни или противников в кратчайшие сроки.

ПРИЛОЖЕНИЕ 1

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – ИПК Издательство стандартов, 2001.
2. ГОСТ 19.301-79 Программа и методика испытаний. //Единая система программной документации. – ИПК Издательство стандартов, 2001.
3. ГОСТ 19.401-78 Текст программы. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.505-79 Руководство оператора. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.404-79 Пояснительная записка. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. Документация Unity [Электронный pесурс]// URL: https://docs.unity3d.com/ru/2018.4/Manual/Coroutines.html (Дата обращения: 10.03.2020, режим доступа: свободный).
9. Unite Austin 2017 - Game Architecture with Scriptable Objects. [Электронный pесурс]// URL: https://www.youtube.com/watch?v=raQ3iHhE\_Kk (Дата обращения: 05.03.2020, режим доступа: свободный).

ПРИЛОЖЕНИЕ 2

**ТЕРМИНОЛОГИЯ**

Ниже приведен список используемых в тексте терминов, рекомендованных для ознакомления.

**User Interface –** интерфейс который пользователь видит перед собой во время игрового процесса.

**PC** – компьютер.

**Observer паттерн –** паттерн проектирования взаимодействия внутри программы.

**WASD –** набор клавиш W, A, S, D на клавиатуре.

**Скорость –** показатель быстроты движений объекта.

**Противник –** не управляемый пользователем объект, совершающий нападение на персонажа игрока.

**Unity –** также Unity 3D, программа для создания игр.

**Box Colider 2D –** один из эллементов доступных в Unity.

**Rigidbody 2D –** эллемент Unity.

**Подземелье -** финальный уровень игры.

**Дом игрока –** уровень в который можно попасть из первой комнаты.

**Наружний мир –** изначальный уровень, на котором появляется игрок.

**Сцена –** текущий уровень игры, поддерживаемый Unity в памяти и иерархии.

**Комната –** конкретный участок на уровне, на котором находится игрок, и который отделён от остальных внешними стенами.

**Инвертарь –** место где хранятся предметы которыми владеет персонаж игрока

**Коллайдер –** эллемент Unity обрабатывающий соприкосновения объектов.

ПРИЛОЖЕНИЕ 3

Таблица 1 Описание и функциональное назначение классов

|  |  |
| --- | --- |
| Классы | Назначение |
| BoolValue | Класс для ScriptableObject вида bool |
| CameraMovement | Класс, регулирующий движение камеры |
| Coin | Класс для работы и создания монет внутри игры. |
| CoinTextManager | Класс, выводящий кол-во монет у игрока в инвертаре на экран |
| ContextClue | Класс, отвечающий за появление над игроком индикатора объекта доступного для взаимодействия |
| Door | Внутриигровой объект доступный для взаимодействия - дверь, класс определяющий работу с ним |
| DoorType | Enum содержащий возможные типы дверей |
| DungeonEnemyRoom | Класс, обеспечивающий работу комнаты, в которой нужно будет убить всех противников, чтобы разблокировать дверь |
| DungeonRoom | Класс, определяющий работу комнаты в подземелье |
| MenuManager | Класс меню для запуска новой игры |
| Enemy | Общий класс для всех противников, от которого будут наследоваться остальные |
| EnemyState | Enum, в котором перечислены все возможные состояния противников |
| FloatValue | Класс для ScriptableObject вида float |
| Heart | Класс отвечающий за объект в мире игры, подняв который игрок восстановит себе очки здоровья |
| HeartManager | Класс обеспечивающий корректную работу UI с здоровьем игрка, в том числе контролирующий визуальное представление на экране |
| Interactable | Общий класс для всех объектов с которыми можно взаимодействовать. |
| Inventory | Класс, ScriptableObject, отвечающий за мэнеджмент инвертаря игрока. |
| Item | Общий класс, представляющий собой определённую вещь, которая может быть помещена в инвертарь. |
| Knockback | Класс занимающийся обработкой всех отталкиваний и получений урона, которые происходят в игре |
| log | Класс представляющий собой обычного противника – бревно. |
| Loot | Класс представляющий собой любой объект, который появляется из противника при смерти |
| LootTable | Класс управляющий появлением предметов из умирающих противников, в том числе вероятностями |
| OverworldRoom | Класс представляющий собой одну из комнат наружного мира |
| PlayerHit | Класс представляющий собой удар игрока, ломающий разрушаемые объекты |
| PlayerMovement | Класс управляющий текущим состоянием игрока и контролирующий то, что с ним происходит |
| PlayerState | Enum содержащий в себе все возможные состояния игрока |
| pot | Класс описывающий разрушаемый объект |
| Powerup | Общий класс представляющий собой подбираемые предметы – монеты или сердца. |
| Projectile | Общий класс от которого наследуются все проджектайлы, т.е. объекты создаваемые персонажами для нанесения урона. |
| RockProjectile | Класс представляющий собой конкретный проджектайл камня |
| Room | Общий класс для всех комнат, любого типа. |
| RoomMove | Класс контролирующий перемещение между комнатами |
| RoomTransfer | Класс поддерживающий перемещение между комнатами и меняющий настройки камеры в новой комнате. |
| SceneTransition | Класс реализующий пермеещение между разными сценами Unity, т.е. между разными уровнями игры |
| Sign | Класс представляющий доступный для взаимодействия эллемент - табличку |
| Signal | Класс, ScriptableObject, реализующий сигнал, который будет посылаться слушающим этот сигнал. |
| SignalListener | Класс, реализующий наблюдателя, позволяющий объекту принимать и обрабатывать входящий сигнал. |
| Switch | Класс представляющий собой переключатель, открывающий связанную с ним дверь |
| TreasureChest | Класс представляющий собой сундук, в котором содержится ключ |
| TurretEnemy | Класс, реализующий тип противника – турель. |
| VectorValue | Класс для ScriptableObject вида Vector2 |

ПРИЛОЖЕНИЕ 4

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ МЕТОДОВ И СВОЙСТВ**

Таблица 2 Описание методов и полей класса BoolValue.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| initialValue | public | bool | Значение изначально заданное для ScriptableObject | |
| RuntimeValue | public | bool | Значение поддерживаемое во время симуляции сцены для ScriptableObject | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| OnAfterDeserialize | public | void | - | Приравнивает текущие значение к изначально заданному |

Таблица 3 Описание методов и полей класса CameraMovement.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| smoothing | public | float | Скорость с которой камера движется за объектом | |
| target | public | Transform | Объект, за которым следит камера | |
| maxPosition | public | Vector2 | Максимальная позиция за которую не может двигаться камера | |
| minPosition | public | Vector2 | Минимальная позиция за которую не может двигаться камера | |
| animi | public | Animator | Аниматор для камеры | |
| camMin | public | VectorValue | Сохраняемый между сценами минимум камеры передающийся через Scriptable Object. | |
| camMax | public | VectorValue | Сохраняемый между сценами максимум камеры передающийся через Scriptable Object. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | void | - | Устанавливает начальные координаты для камеры, а также подгружает Аниматор |
| LateUpdate | private | void | - | Осуществляет плавное движение камеры за целью |
| BeginKick | public | void | - | Осуществляет подрагивание камеры при получении урона игроком |
| KickCo | public | IEnumerator | - | Оканчивает подрагивание камеры после исполнения анимации |

Таблица 4 Описание методов и полей класса Coin.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| playerInventory | public | Inventory | Инвертарь игрока | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | void | - | Отправляет сигнал о монете |
| OnTriggerEnter2D | public | void | Collider2D other | Определяет что пользователь подобрал монету и отправляет сигнал о подборе. |

Таблица 5 Описание полей и методов класса CoinTextManager.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| playerInventory | public | Inventory | Обрабатываемый инвентарь игрока | |
| coinDisplay | public | TextMeshProUGUI | Показатель монет игрока в UI | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | void | - | Инциализация счётчика монет игрока |
| UpdateCoinCount | public | void | - | Обнавляет счётчик монет |

Таблица 6 Описание полей и методов класса ContextClue.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| contextClue | public | GameObject | Индикатор появляющийся над игроком | |
| contextActive | public | bool | Активен ли индикатор | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| ChangeContext | public | void | - | Меняет состояние индикатора |

Таблица 7 Описание полей и методов класса Door.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| thisDoorType | public | DoorType | Тип конкретной двери | |
| open | public | bool | Открыта ли дверь | |
| playerInventory | public | Inventory | Инвентарь игрока | |
| doorSprite | public | SpriteRenderer | Текущий спрайт двери | |
| physicsCollider | public | BoxCollider2D | Коллайдер соприкосновения | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Update | private | void | - | Осуществляет взаимодействие с дверью, проверяет наличие ключа, если дверь такого типа |
| Open | public | void | - | Открывает дверь |
| Close | public | void | - | Закрывает дверь |

Таблица 8 Описание методов и полей класса DungeonEnemyRoom.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| doors | public | Door[] | Массив дверей, ведущих в комнату | |
| dialogBox | public | GameObject | Окно для вывода диалога | |
| dialogText | public | Text | Содержимое окна для вывода диалога | |
| dialog | public | string | Выводимое сообщение | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | void | - | Открывает двери |
| EnemiesActive | public | int | - | Подсчитывает кол-во активных противников |
| CheckEnemies | public | void | - | Проверяет убиты ли все противники, если да, то запускает сообщение о победе в игре |
| OnTriggerEnter2D | public | void | Collider2D other | Осуществляет колизию с игроком, когда он заходит в комнату, закрывает все двери |
| OnTriggerExit2D | public | void | Collider2D other | Деактивирует противников и камеру по выходу пользователя из комнаты |
| OpenDoors | public | void | - | Открывает все двери |
| CloseDoors | public | void | - | Закрывает все двери |
| Co\_ForEnd | private | IEnumerator | float delay | Выключает приложение по прошествию задержки |
| Co\_OnEnter | private | IEnumerator | float delay | Сообщает игроку что он попал в особую комнату |

Таблица 9 Описание полей и методов класса Enemy.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  Доступа | Тип | Назначение | |
| currentState | public | EnemyState | Текущие состояние противника | |
| maxHealth | public | FloatValue | Заранее заданное кол-во очков здоровья для противника | |
| health | public | float | Текущее кол-во здоровья противника | |
| enemyName | public | string | Имя противника | |
| baseAttack | public | int | Величина урона наносимого атакой противника | |
| moveSpeed | public | float | Скорость противника | |
| homePosition | public | Vector2 | Изначальная позиция противника | |
| deathEffect | public | GameObject | Эффект смерти противника | |
| deathEffectDelay | public | float | Задержка в течении которой будет показываться эффект смерти | |
| thisLoot | public | LootTable | Объекты, которые появятся после смерти противника | |
| roomSignal | public | Signal | Сигнал для комнаты, позволяющий проверять, активен ли противник | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Awake | private | void | - | Выставляет текущие здоровье равным изначально заданному |
| OnEnable | private | void | - | Выставляет позицию на изначальную, а также состояние и очки здоровья |
| TakeDamage | private | void | float damage | Наносит урон противнику, а так же просчитывает мёртв ли он, если мёртв создаёт предметы, посылает сигнал и проигрывает анимацию смерти |
| MakeLoor | private | void | - | Создаёт предметы, которые должны выпасть после смерти противника |
| DeathEffect | private | void | - | Проигрывает анимацию смерти противника |
| Knock | public | void | Rigidbody2D myRigidbody, float knockTime, float damage | Осуществляет нанесение урона и отталкивание противника |
| KnockCo | private | IEnumerator | Rigidbody2D myRigidbody, float knockTime | Осуществляет отбрасывание противника |

Таблица 10 Описание полей и методов класса FloatValue.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| initialValue | public | float | Изначально задавемое значение для ScriptableObject | |
| RuntimeValue | public | float | Значения во время симуляции | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| OnAfterDeserialize | public | void | - | Приравнивает значение во время симуляции к изначальному |

Таблица 11 Описание методов и полей класса Heart.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| playerHealth | public | FloatValue | Здоровье персонажа | |
| healthContainers | public | FloatValue | Текущее максимальное кол-во здоровья персонажа | |
| amountToIncrease | public | float | Кол-во восстанавливаемого здоровья | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| OnTriggerEnter2D | public | void | Collider2D other | Осуществляет восстановление здоровья |

Таблица 12 Описание методов и полей класса HeartManager.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| hearts | public | Image[] | Массив изображений сердец из UI | |
| fullHeart | public | Sprite | Изображение полного сердца | |
| halfFullHeart | public | Sprite | Изображение половинчатого сердца | |
| emptyHeart | public | Sprite | Изображение пустого сердца | |
| heartContainers | public | FloatValue | Кол-во контейнеров сердец у игрока | |
| playerCurrentHealth | public | FloatValue | Текущее здоровье игрока | |
| dialogBox | public | GameObject | Окно для вывода сообщения ироку | |
| dialogText | public | Text | Текст окна для вывода сообщения | |
| dialog | public | string | Текст который нужно вывести игроку | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | void | - | Инициализирует UI здоровья игрока |
| InitHearts | public | void | - | Заполняет UI сердца для изображения очков здоровья игрока |
| UpdateHearts | public | void | - | Поддерживает актуальную информацию о кол-ве очков здоровья игрока на нашем UI, если игрок умер высвечивает соответствующее сообщение |
| Co\_ForEnd | private | IEnumerator | Float delay | После заданной задержки закрывает приложение |

Таблица 13 Описание полей и методов класса Interactable.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| context | public | Signal | Сигнал сообщающий о необходимости индикатора над головой игрока | |
| playerInRange | public | bool | Находится ли игрок в радиусе | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| OnTriggerEnter2D | private | void | Collider2D other | Отправляет сигнал о том, что игрок появился в радиусе |
| OnTriggerExit2D | private | void | Collider2D other | Отправляет сигнал о том, что игрок покинул радиус |

Таблица 14 Описание полей и методов класса Inventory.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| currentItem | public | Item | Текущий выбранный предмет | |
| items | public | List<Item> | Содержимое инвентаря | |
| numberOfKeys | public | int | Кол-во ключей в инвентаре | |
| coins | public | int | Кол-во монет в инвентаре | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| AddItem | public | void | Item itemToAdd | Добавляет переданный предмет в инвентарь |

Таблица 15 Описание методов и полей класса Item.cs

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор  доступа | Тип | Назначение |
| itemSprite | public | Sprite | Изображение предмета |
| itemDescription | public | string | Описание предмета |
| isKey | public | bool | Является ли предмет ключём |

Таблица 17 Описание полей и методов класса Knockback.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| thrust | public | float | Сила отбрасывания | |
| knockTime | public | float | Время на которое отбросит | |
| damage | public | float | Урон | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| OnTriggerEnter2D | private | void | Collider2D other | Если то, по чему наносят удар разрушаемое – разрушает, иначенаносит урон и отбрасывает |

Таблица 18 Описание методов и полей класса log.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| myRigidbody | public | Rigidbody2D | Физическое тело противника | |
| target | public | Transform | Текущая позиция цели для атак | |
| chaseRadius | public | float | Радиус в котором противник преследует цель | |
| attackRadius | public | float | Радиус от цели, на котором останавливается противник | |
| anim | public | Animator | Аниматор | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| CheckDistance | public | void | - | Метод получающий расстояние до цели и решающий, преследовать ли её |
| Start | private | void | - | Устанавливает изначальные значения |
| FixedUpdate | private | void | - | Проверяет дистанцию |
| SetAnimFloat | public | void | Vector2 setVector | Высставляет анимации передвижения в разные стороны |
| changeAnim | public | void | Vector2 direction | Выясняет какую из анимаций нужно проиграть при передвижении |
| ChangeState | public | void | EnemyState newState | Меняет состояние противника |

Таблица 19 Описание полей и методов класса Loot.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| thisLoot | public | Powerup | Принадлежность предмета к сердцам/монетам | |
| lootChance | public | int | Шанс на создание | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | IEnumerator | - | Выводит на экран начальное сообщение обучения |
| TypeText | public | TypeText | String text | При вызове и передаче параметра-строки выводит ее на экран с анимацией |

Таблица 20 Описание полей и методов класса LootTable.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| loots | public | Loot[] | Все объекты которые дожлны появится при смерти противника | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| LootPowerup | public | Powerup | - | Создаёт объект которые должен появится при смерти противника |

Таблица 21 Описание методов и полей класса OverworldRoom.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| starter | public | BoolValue | Нужно ли показывать игроку приветсвенное сообщение | |
| dialogBox | public | GameObject | Окно диалога | |
| dialogText | public | Text | Текст внутри окна диалога | |
| dialog | public | string | Диалог которный нужно вывести в окно | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | void | - | Выводит сообщение о начале игры, если игра только началась |
| Co\_OnEnter | private | IEnumerator | Float delay | Выключает сообщение о начале игры после задержки |

Таблица 21 Описание методов и полей класса PlayerMovement.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор  доступа | Тип | Назначение | |
| currentState | public | PlayerState | Текущее состояние игрока | |
| speed | public | float | Скорость игрока | |
| myRigidbody | private | Rigidbody2D | Физическое тело игрока | |
| change | private | Vector3 | Изменение позиции игрока | |
| animator | private | Animator | Аниматор | |
| currentHealth | public | FloatValue | Текущие здоровье игрока | |
| playerHealthSignal | public | Signal | Сигнал для передачи информации о здоровье игрока | |
| startingPosition | public | VectorValue | Стартовая позиция игрока | |
| playerInventory | public | Inventory | Инвертарь игрока | |
| receivedItemSprite | public | SpriteRenderer | Спрайт полученного предмета | |
| playerHit | public | Signal | Сигнал для передачи информации о атаке игрока | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Start | private | void | - | Инициализирует стартовые значения |
| Update | private | void | - | Просчитывает текущую позицию игрока, при подходящей позиции инициализирует атаку и перемещение |
| AttackCo | private | IEnumerator | - | Отвечает за анимацию атаки, и смену позиций |
| RaiseItem | public | void | - | Отвечает за приём предмета, а так же проигрывание соответствующей анимации |
| UpdateAnimationAndMove | private | void | - | Совершает выбор правильной анимации для передвижения |
| MoveCharacter | private | void | - | Придаёт перемещение физическому телу игрока |
| Knock | public | void | float knockTime, float damage | Управляет отбрасыванием игрока, а также менеджментов получаемого урона |
| KnockCo | private | IEnumerator | float knockTime | Отбрасывает игрока, держа его в отброшенной позиции заданное время |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подп. | Дата |
| Измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |