**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»  
ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите** Заместитель директора  по УМР \_\_\_\_\_\_\_\_\_\_\_\_ Е.Г. Конакина\_  (Подпись) (ФИО)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2020 г. |
|  |

**курсовой проект**Тема «Игра с графическим интерфейсом ТЕМНЫЙ ЛЕС»

специальность 09.02.03 группа 32928/1

Студент (ка) Дмитриева В.В.  
 (подпись) (ФИО)

Студент (ка) Петрова А.В.  
 (подпись) (ФИО)  
Преподаватель Девятко Н.С.

(подпись) (ФИО)

Санкт-Петербург  
2020

Содержание

[ВВЕДЕНИЕ 3](#_Toc24557942)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ 4](#_Toc24557943)

[1.1. Описание предметной области 4](#_Toc24557944)

[1.2. Анализ методов решения 6](#_Toc24557945)

[1.3. Обзор средств программирования 6](#_Toc24557946)

[1.4. Описание языка C# 7](#_Toc24557947)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc24557948)

[2.1. Постановка задачи 11](#_Toc24557949)

[2.1.1. Основания для разработки 11](#_Toc24557950)

[2.1.2. Назначение программы 11](#_Toc24557951)

[2.2. Проектирование приложения 11](#_Toc24557952)

[2.2.1. Диаграмма прецедентов 11](#_Toc24557953)

[2.2.2. Диаграмма последовательностей 12](#_Toc24557954)

[2.2.3. Диаграмма классов 14](#_Toc24557955)

[2.3. Текст программы 14](#_Toc24557956)

[2.4. Описание программы 15](#_Toc24557957)

[2.4.1. Общие сведения 15](#_Toc24557958)

[2.4.2. Функциональное назначение 15](#_Toc24557959)

[2.4.3. Описание логической структуры системы 15](#_Toc24557960)

[2.4.4. Используемые технические и программные средства 17](#_Toc24557961)

[2.4.5. Вызов и загрузка 17](#_Toc24557962)

[2.5. Руководство оператора 18](#_Toc24557963)

[2.5.1. Назначение программы 18](#_Toc24557964)

[2.5.2. Выполнение программы и сообщения оператору 18](#_Toc24557965)

[2.6. Программа и методика испытаний 19](#_Toc24557966)

[2.6.1. Объект испытаний 19](#_Toc24557967)

[2.6.2. Цель испытаний 19](#_Toc24557968)

[2.6.3. Требования к программе 19](#_Toc24557969)

[2.6.4. Требования к программной документации 20](#_Toc24557970)

[2.6.5. Средства и порядок испытаний 20](#_Toc24557971)

[2.6.6. Методы испытаний 21](#_Toc24557972)

[2.7. Протокол испытаний 24](#_Toc24557973)

[ЗАКЛЮЧЕНИЕ 26](#_Toc24557974)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc24557975)

[ПРИЛОЖЕНИЕ А 28](#_Toc24557976)

[ПРИЛОЖЕНИЕ Б 34](#_Toc24557977)

[ПРИЛОЖЕНИЕ В 42](#_Toc24557979)

# ВВЕДЕНИЕ

Жизнь современного человека не представляется возможной без использования персонального компьютера. Данное устройство используется повсеместно: для образовательных, развлекательных, научно – технических, преподавательских и медицинских целей. Компьютер способен хранить огромное количество информации, он облегчает жизнь пользователю, позволяет быстро передавать и принимать различные файлы, работать с базами данных. Другими словами, человек использует ПК в качестве средства решения различных задач. Также не обошлась без внимания и игровая индустрия. Игры интересны многим пользователям, они разрабатываются для разных возрастных групп и подразделяются на огромное количество различных жанров. Помимо проведения досуга игры позволяют повысить скорость реакции, развить логическое и абстрактное мышление, находить новые подходы для решения той или иной задачи. Кроме того, игровая индустрия позволяет игроку примерить на себя роли разных персонажей, получить новый опыт и эмоции.

Разработанная в данном курсовом проекте программа может быть использована в развлекательных целях.

# 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ

## 1.1. Описание предметной области

Компьютерная игра - техническая игра, в которой игровое поле находится под управлением ЭВМ или воспроизводится на экране дисплея. Компьютерная игра - одно из основных и массовых применений микропроцессорной вычислительной техники, относящейся к досугу, воспитанию и образованию[1].

Жанровое различение игр начало активно развиваться с середины 90-х годов. Основой для формирования классического свода жанров послужили серии игр. Эти серии были образованы в результате того, что после выхода в свет популярной «оригинальной игры», как официальные, так и сторонние разработчики продолжали эксплуатировать черты первой игры, создавая «клонов».

Игры подразделяются на несколько основных жанров:

* *Действие* — жанр компьютерных игр, в котором делается упор на эксплуатацию физических возможностей игрока, в том числе координации глаз и рук и скорости реакции. Жанр представлен во множестве разновидностей: от [файтингов](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D1%82%D0%B8%D0%BD%D0%B3" \o "Файтинг), [шутеров](https://ru.wikipedia.org/wiki/%D0%A8%D1%83%D1%82%D0%B5%D1%80) и [платформеров](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D1%80" \o "Платформер), которые считаются наиболее важными для жанра до MOBA и некоторых стратегий в реальном времени.

Данный жанр включает в себя подвиды:

* + «шутеры» (от англ. *shoot — стрелять*) (пример: Half-Life, серии игр Call of Duty и World of Tanks);
  + «хорроры» (*horror - ужас*) (пример: Don’t knock Twice, OutLast);
  + «стелс» (от англ. *stealth — скрытный*) (пример: серии игр Max Payne, Assasins Creed).
* *Симулятор* — имитатор (обычно механический или компьютерный), задача которого состоит в имитации управления каким-либо процессом, аппаратом или транспортным средством. С помощью компьютерно-механических симуляторов, абсолютно точно воспроизводящих интерьер кабины аппарата, тренируются пилоты, космонавты, машинисты высокоскоростных поездов.
* *Стратегия* — игры данного жанра характеризуются тем, что игроку для достижения цели необходимо применять стратегическое мышление, и оно противопоставляется быстрым действиям и реакцией, которые, как правило, не обязательны для успеха в таких играх. Стратегические игры бывают абстрактные, настольные (шахматы, шашки, «Монополия»), с симуляцией менеджмента («Моя Кофейня») и другие.
* *Ролевая игра* (RPG — англ. *Role Playing Game*) —  [жанр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80) [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), основанный на элементах игрового процесса традиционных [настольных ролевых игр](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%82%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%80%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0). В ролевой игре игрок управляет одним или несколькими персонажами, каждый из которых описан набором численных характеристик, списком способностей и умений; примерами таких характеристик могут быть [очки здоровья](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%BA%D0%B8_%D0%B7%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%D1%8C%D1%8F) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *hit points, HP*), показатели силы, ловкости, интеллекта, защиты, уклонения, уровень развития того или иного навыка и т. п.
* *Квест*— один из основных [жанров](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80), представляющий собой интерактивную историю с главным героем, управляемым игроком. Важнейшими элементами игры в жанре квеста являются собственно повествование и исследование мира, а ключевую роль в игровом процессе играет решение головоломок и задач, требующих от игрока умственных усилий. Такие характерные для других жанров компьютерных игр элементы, как бои, экономическое планирование и задачи, требующие от игрока скорости реакции и быстрых ответных действий, в квестах сведены к минимуму или вовсе отсутствуют.
* *Платформер* (англ. platformer, platform game) — жанр компьютерных игр, в которых основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня[2].

## 1.2. Анализ методов решения

Жанр нашей игры можно определить как платформер. Компьютерная игра «Темный Лес» может быть реализована несколькими способами:

1. С помощью тайтлов: в нём позиционирование персонажа ограничено сеткой тайтлов, таким образом, он никогда не сможет встать между двумя тайлами. Для создания иллюзии плавного передвижения могут быть использованы различные анимации, однако, согласно игровой логике, персонаж всегда находится прямо на конкретном тайтле.
2. Используя уклоны: уклоны могут быть использованы как препятствия и в то же время позволяют персонажу заходить на их тайл. Также они вызывают изменение Y координаты при простом перемещении вдоль оси Х.
3. С помощью односторонних платформ: здесь присутствуют обычные платформы, на которые можно встать, но при этом можно запрыгнуть сквозь них снизу. Другими словами, они считаются препятствием если вы стоите на них, и не считаются, если вы прыгаете снизу.
4. С помощью двусторонних платформ: данные платформы не позволяют игроку запрыгнуть на них снизу[3].

В данном курсовом проекте мы будем использовать двусторонние платформы, поскольку это оптимальный и наиболее удобный способ построения сцен игры.

## 1.3. Обзор средств программирования

Требования к курсовому проекту включали в себя удобный пользовательский интерфейс, поэтому при написании данного курсового проекта наша задача состояла в написании программы используя игровой движок Unity. Unity поддерживает несколько языков: среди них: JavaScript и C#. Поскольку JavaScript подходит больше для написания страниц сайтов, мы акцентировали внимание на C#, так как данный язык программирования удовлетворяет нашим условиям написания программы.

Таким образом, средством программирования данного курсового проекта был выбран язык С#, среда разработки Visual Studio 2019.

## 1.4. Описание языка C#

## C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML[4].

## В языке C# существует две разновидности типов данных: значимые типы (value types) и ссылочные типы (reference types). К значимым типам относятся простые типы (char, int, float), перечисляемые типы (enum) и структуры (struct), то есть типы, напрямую содержащие данные. Ссылочными типами являются классы, интерфейсы, массивы (поддерживаются одно- и многомерные массивы) и делегаты: типы, хранящие ссылки на объекты. Для задания новых типов данных разработчики могут использовать перечисления и структуры, а также классы, интерфейсы и делегатов. [5]

**Ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | extern | null | struct |
| as | false | object | switch |
| base | finally | operator | this |
| bool | fixed | out | throw |
| break | float | override | true |
| byte | for | params | try |
| case | foreach | private | typeof |
| catch | goto | protected | uint |
| const | if | public | ulong |
| continue | implicit | readonly | unchecked |
| decimal | in | ref | unsafe |
| default | int | return | ushort |
| delegate | interface | sbyte | using |
| do | internal | sealed | virtual |
| double | is | short | void |
| else | lock | sizeof | while |
| enum | long | stackalloc |  |
| event | namespace | static |  |
| explicit | new | string |  |

**Контекстные ключевые слова C#**

|  |  |  |  |
| --- | --- | --- | --- |
| add | equals | join | set |
| ascending | from | let | value |
| async | get | on | var |
| await | global | orderby | where |
| by | group | partial | yield |
| descending | in | remove |  |
| dynamic | into | select |  |

**Основные алгоритмические конструкции C#**

|  |  |
| --- | --- |
| Оператор присваивания | string hello = "hello " + "world"; //результат равен "hello world"  int x1 = 2 + 4; // результат равен 6  int x2 = 10 - 6; //результат равен 4  int x3 = 10 \* 6; //результат равен 60  double x4 = 10.0 / 4.0; //результат равен 2.5  double x5 = 10.0 % 4.0; //результат равен 2  int y1 = 5;  int z1 = ++y1; // z1=6; y1=6  int y2 = 5;  int z2 = y2++; // z2=5; y2=6  int y3 = 5;  int z3 = --y3; // z3=4; y3=4  int y4 = 5;  int z4 = y4--; // z4=5; y4=4 |
| Условный оператор | **if**  (*условие)* **{**(*действие)* } **else** {(*альтернатива)}* ;  int num1 = 8;  int num2 = 6;  if(num1 > num2)  {      Console.WriteLine("Число {0} больше числа {1}", num1, num2);  }  else  {      Console.WriteLine("Число {0} меньше числа {1}", num1, num2);  } |
|  | *Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:*  Console.WriteLine("Нажмите Y или N");  string selection = Console.ReadLine();  switch (selection)  {      case "Y":          Console.WriteLine("Вы нажали букву Y");          break;      case "N":          Console.WriteLine("Вы нажали букву N");          break;      default:          Console.WriteLine("Вы нажали неизвестную букву");          break;  }  *После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора сase. И если совпадение будет найдено, то будет выполняться определенный блок сase.*  *В конце блока сase ставится оператор break, чтобы избежать выполнения других блоков.*  *Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default, как в примере выше.* |
| Арифметический цикл  **(**применяется, когда известно количество повторений цикла) | *for ([инициализация счетчика]; [условие]; [изменение счетчика])*  for (int i = 0; i < 9; i++)  {      Console.WriteLine("Квадрат числа {0} равен {1}", i, i \* i);  } |
| Цикл с предусловием  **(**применяется, когда неизвестно количество повторений цикла) | Этот цикл будет выполняться до тех пор, пока истинно *условие* (логическое выражение, возвращающее значение типа **Boolean**). При этом если это выражение сразу равно **false**, *тело цикла* не будет выполнено ни разу. Нужно очень внимательно следить за написанием *условия* и контролем завершения цикла, так как в результате ошибки цикл **while** будет повторяться бесконечное количество раз, что приведёт к "зацикливанию" и "зависанию" программы. |
| Цикл с постусловием  **(**применяется, когда неизвестно количество повторений цикла) | *do { тело цикла } while условие;*  Повторения сначала выполняет *тело цикла*, а затем уже проверяет выполнение *условия*: Таким образом, этот вариант цикла гарантирует, что *тело цикла* будет выполнен по крайней мере один раз. И будет выполняться до тех пор, пока *условие* не станет истинным (т.е. **true**). |

C# поддерживает объектно – ориентированное программирование. ООП - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Постановка задачи

### 2.1.1. Основания для разработки

Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.02 «Прикладное программирование» и утверждена Институтом среднего профессионального образования.

### 2.1.2. Назначение программы

Игровая программа с графическим интерфейсом «Темный лес»: человек играет за сказочного персонажа; цель персонажа – освободить лес от демонов и восстановить баланс в природе; для выполнения миссии главный герой сражается с монстрами и собирает кристаллы жизни..

Полный текст технического задания приведен в приложении А.

## 2.2. Проектирование приложения

На этапе проектирования были разработаны диаграмма прецедентов, диаграмма последовательностей и диаграмма классов.

### 2.2.1. Диаграмма прецедентов

Диаграммы прецедентов частично описывает use case – прецедент использования проектируемой системы, давая частичное описание частичного применения системы с точки зрения условного внешнего обозревателя (за которым – в идеале - стоит согласованная точка зрения участников работ). При этом описание фокусируется на том, что должна делать система по отношению к своему внешнему окружению (периферии), а не то на том, как она эта делает, то есть диаграмма прецедентов есть частичная спецификация (рисунок 1).



Рисунок 1 Диаграмма прецедентов

### 2.2.2. Диаграмма последовательностей

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.



Рисунок 2 Диаграмма последовательностей

### 2.2.3. Диаграмма классов

На рисунке 3 показана диаграмма классов проекта. Программа содержит 2 класса: HeroController.cs - класс управления игроком, класс Attack.cs содержит методы и функции для совершения атаки и получения урона.



Рисунок 3 Диаграмма классов

## 2.3. Текст программы

Текст программы в соответствии с ГОСТ 19.101-77 (СТ СЭВ 1626-79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении В.

## 2.4. Описание программы

### 2.4.1. Общие сведения

Игровая программа с графическим интерфейсом «Темный Лес»: человек играет за сказочного персонажа; цель персонажа – освободить лес от демонов и восстановить баланс в природе; для выполнения миссии главный герой сражается с монстрами и собирает кристаллы жизни.

### 2.4.2. Функциональное назначение

Основное назначение программного продукта заключается в организации игры с графическим интерфейсом «Темный Лес»: человек играет за сказочного персонажа; цель персонажа – освободить лес от демонов и восстановить баланс в природе; для выполнения миссии главный герой сражается с монстрами и собирает кристаллы жизни.

### 2.4.3. Описание логической структуры системы

1 версия программы содержит 3 модуля и 8 классов:

Level1 – модуль первого уровня игры.

Level2 – модуль второго уровня игры.

Level3 – модуль третьего уровня игры.

*HeroController.cs* – класс для управления свойствами персонажа: бегом, прыжками, пополнением и упадком здоровья.

*Attack.cs* – класс для совершения атаки как со стороны героя, так и со стороны противника, используя функции анимации.

*Crystall.cs* – класс для отображения количества собранной валюты в игре.

*CrystallCollection.cs* – класс для непосредственного сбора валюты, использующий триггеры.

*DieSpace.cs* – класс для рестарта прохождения уровня в случае, если игрок погибает при падении в пропасть.

*Enemy.cs* – класс для неигровых персонажей, не находящихся под управлением пользователя.

*Patrol.cs* – класс для отображения территории передвижения неигровых персонажей.

*NextLevel.cs* – модуль для перехождения на последующие уровни при условии сбора всех кристаллов на сцене.

2 версия программы содержит 3 модуля и 8 классов аналогичных первой версии, а также 4 дополнительных модуля и 5 классов:

*ChooseLevel* – модуль выбора уровня игры.

*Load* – модуль для загрузки.

*Menu* – модуль для отображения главного меню и меню настроек.

*Titres* – модуль для отображения конечной заставки игры с титрами.

*LevelLoader*.cs – класс для загрузки сцены с уровнем и отображения знака загрузки.

*LevelManger1*.cs – класс для просмотра пройденных уровней и выбора уровня прохождения, а также сохранения прогресса.

*MainMenu*.cs – класс для выбора пункта из главного меню.

*MessageBox*.cs – класс для отображения монологов и повествования сюжета игры.

*PauseMenu*.cs – класс для приостановления прохождения уровня.

*SettingsMenu*.cs – класс для изменения настроек интерфейса игры.

Программа использует функции следующих библиотек среды С# и среды Unity:

System, System.Collections.Generic, System.Collections, UnityEngine, UnityEngine.SceneManagement, UnityEngine.UI, UnityEngine.Audio.

Исполняемый файл программы создан средствами среды C# и графического движка Unity, имеет имя The\_Dark\_Forest.exe и размер 625 кбайт для 1 и 2 версии.

#### 2.4.3.1. Описание методов класса HeroController.cs

Класс HeroController.cs предназначен для управления игровым персонажем. Связность модуля информационная (СС=9), так как выходные данные из класса Crystall.cs используются как входные данные.

Модуль содержит 6 методов:

private void Start() метод для инициализации переменных;

void Update() метод для отслеживания нажатия кнопок с клавиатуры;

private void FixedUpdate() метод для изменения состояния персонажа(с бега на прыжок, пополнение и упадок здоровья);

void OnTriggerEnter2D(Collider2D collision) метод, позволяющий пополнять здоровье, лишаться его и собирать игровую валюту;

public void Run() метод, задающий анимацию для бега;

public void Jump() метод, задающий анимацию для прыжка;

#### 2.4.3.2. Описание методов класса Attack.cs

Класс формы Attack.cs предназначен для совершения атаки как со стороны героя, так и со стороны противника, используя функции анимации. Связность модуля процедурная (СС=5), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 4 метода:

private void Start() метод для инициализации переменных;

void Update() метод для отслеживания нажатия кнопок с клавиатуры;

public void OnAttack() метод для совершения атаки врагу;

void OnDrawGizmosSelected() метод отображения радиуса наноса урона;

#### 2.4.3.3. Описание методов класса Crystall.cs

Класс формы Crystall.cs предназначен отображения количества собранной валюты в игре. Связность модуля процедурная (СС=5), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 2 метода:

private void Start() метод для инициализации переменных;

void Update() метод для подсчета собранной валюты в левом экране игры;

#### 2.4.3.4. Описание методов класса Enemy.cs

Класс формы Enemy.cs предназначен для для неигровых персонажей, не находящихся под управлением пользователя. Связность модуля информационная(СС=9), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 4 метода:

private void Start() метод для инициализации переменных;

private void ТakeDamage(int damage) метод для получения урона врагу;

public void OnEnemyAttack() метод для получения урона главного героя;

private void OnTriggerStay2D(Collider2D other) метод для анимации атаки врага;

#### 2.4.3.5. Описание методов класса Patrol.cs

Класс формы Patrol.cs предназначен для отображения территории передвижения неигровых персонажей. Связность модуля информационная(СС=9), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 5 методов:

private void Start() метод для инициализации переменных;

void Update() метод для выявления состояния неигрового персонажа;

private void Chill() метод для патрулирования неигрового персонажа;

public void GoBack() метод для возращения неигрового персонажа на точку патрулирование

private void Angry() метод для агрессивного состояния неигрового персонажа, в случае которого враг атакует;

#### 2.4.3.6. Описание методов класса NextLevel.cs

Класс формы NextLevel.cs предназначен для перехода на последующие уровни при условии сбора всех кристаллов на сцене. Связность модуля информационная(СС=9), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 3 метода:

private void OnTriggerEnter2D(Collider2D other) метод для осуществления перехода персонажа на новый уровень;

public void LoadLevel(int SceneIndex) метод для загрузки сцены с уровнем;

IEnumerator LoadAsynchronously(int SceneIndex) метод для отображения знака загрузки;

#### 2.4.3.7. Описание методов класса MainMenu.cs

Класс MainMenu.cs предназначен для выбора пункта из главного меню. Связность модуля процедурная (СС=5), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 3 метода:

public void Menu() – метод нажатия кнопки для возвращения в главное меню;

public void PlayGame() – метод нажатия кнопки «Играть»;

public void ExitGame() - метод нажатия кнопки «Выход».

#### 2.4.3.8. Описание методов класса MessageBox.cs

Класс MessageBox.cs предназначен для отображения монологов персонажа и текста повествования сюжета игры. Связность модуля процедурная (СС=5), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 7 методов:

public void PlaySound() – метод проигрывания звука голоса персонажа;

void Awake()– метод для нахождения компонентов MessageBox на сцене;

void Start() - метод для переключения MessageSign в позицию «неактивный»;

void OnTriggerEnter2D(Collider2D col) - метод для отображения MessageBox при достижении игроком определенной позиции сцены;

public void ShowMessageBox() - метод для отображения текста в MessageBox;

public void Accept() – метод для отслеживания нажатия кнопки «ОК» на поле MessageBox;

public void hideMessageBox() – метод для скрытия поля монолога при достижении конца текста;

#### 2.4.3.9. Описание методов класса PauseMenu.cs

Класс PauseMenu.cs предназначен для отображения монологов персонажа и текста повествования сюжета игры. Связность модуля процедурная (СС=5), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 4 метода:

void Update() – метод отслеживания нажатия кнопки «Escape», показывающий меню паузы;

public void Resume() - метод отслеживания нажатия кнопки «Escape», возобновляющий прохождение уровня;

public void LoadMenu() - метод отслеживания нажатия кнопки «Главное меню» в меню паузы;

public void QuitGame() - метод отслеживания нажатия кнопки «Выход» в меню паузы;

#### 2.4.3.10. Описание методов класса SettingsMenu.cs

Класс SettingsMenu.cs предназначен для изменения настроек интерфейса и звука игры. Связность модуля процедурная (СС=5), так как части модуля реализуют определенный сценарий поведения.

Модуль содержит 5 методов:

void Start() – метод для настройки начального разрешения экрана;

public void SetVolume(float musicSound) - метод изменения громкости звука: озвучки и фоновой музыки;

public void SetQuality(int QualityIndex) - метод изменения качества графики;

public void SetFullScreen(bool isFullscreen) - метод для установки полноэкранного режима;

public void SetResolution(int resolutionIndex) - метод для выбора разрешения экрана;

### 2.4.4. Используемые технические и программные средства

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~3Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

### 2.4.5. Вызов и загрузка

Программа может быть загружена как с диска, так и с жесткого диска. В последнем случае требуется предварительно переписать программу с диска на жесткий диск.

Исполняемым файлом программы является файл The\_Dark\_Forest.exe (1, 2 версии). Необходимые файлы для корректной работы приложения: UnityCrashHandler32.exe, UnityPlayer.dll. Для его запуска необходимо дважды щелкнуть по исполняемому файлу левой кнопкой мышки.

## 2.5. Руководство оператора

### 2.5.1. Назначение программы

Основное назначение программного продукта заключается в организации игры с графическим интерфейсом «Темный Лес»: человек играет за сказочного персонажа; цель персонажа – освободить лес от демонов и восстановить баланс в природе; для выполнения миссии главный герой сражается с монстрами и собирает кристаллы жизни.

### 2.5.2. Выполнение программы и сообщения оператору

Для запуска программы дважды щелкните левой кнопкой мыши по исполняемому файлу The\_Dark\_Forest.exe.

Процесс игры, этапы работы приложения показаны в виде иллюстраций в приложении Б.

При запуске работы приложения загружается главное меню, на которой отображены три кнопки: «Играть», «Настройки» и «Выход». Чтобы начать игру, нужно нажать «Играть», после загрузки, пользователю представляется выбор уровней. По умолчанию отрыт только первый уровень, а два последующих откроются после прохождения предыдущего. Чтобы пройти уровни надо собрать все кристаллы, присутствующие на сцене, и победить врагов. После успешного прохождения всех уровней появляется конечная заставка с титрами, свидетельствующая пользователю о конце игры.

Также на панели главного меню присутствует кнопка «Настройки». При нажатии на нее отображается меню для изменения графического интерфейса и звука игры. Пользователю предоставляется возможность выбрать подходящее для него разрешение экрана, поставить полноэкранный режим, поменять качество графики, а также убавить или прибавить громкость звука.

Во время прохождения уровня, нажав на кнопку «Esc», игра переходит в режим паузы: действие игры приостанавливается, на экране появляются кнопки: «Продолжить», «Главное меню», «Выход», выполняющие возобновление прохождения уровня, открытие панели главного меню и закрытие приложения соответственно.

## 2.6. Программа и методика испытаний

### 2.6.1. Объект испытаний

Объектом испытаний является игровая программа The\_Dark\_Forest.exe. Человек играет за сказочного персонажа; цель персонажа – освободить лес от демонов и восстановить баланс в природе; для выполнения миссии главный герой сражается с монстрами и собирает кристаллы жизни.

### 2.6.2. Цель испытаний

Целью испытаний является проверка соответствия программы требованиям Технического Задания.

### 2.6.3. Требования к программе

В процессе испытаний подлежат проверке следующие требования к программе:

#### 2.6.3.1. Требования к функциональным характеристикам

*1 версия*:

- программа должна предоставлять возможность передвижения игрока по сцене с применением анимации;

- программа должна осуществлять сбор игровых кристаллов: количество собранных кристаллов отображается в верхнем углу экрана (для прохождения игры нужно собрать определенное количество кристаллов);

- программа обязана гарантировать персонажам возможность атаковать: при нанесении удара противнику, уровень здоровья врага уменьшается;

- программа должна предоставлять право пополнения здоровья героя путем лечения лекарственными ягодами в случае атаки со стороны противника;

- программа обязана показывать шкалу здоровья при ее пополнении и уроне;

- в случае гибели персонажа (полного опустошения шкалы здоровья) прохождение уровня возобновляется сначала;

- программа должна обеспечивать переход игрока на новый уровень.

*2 версия*:

- программа должна открывать главное меню при запуске игры;

- программа обязана предоставить пользователю возможность настроить характеристики игры: звук, качество графики, разрешение экрана;

- перед началом каждого уровня игры программа должна отображать знак загрузки;

- во время прохождения игры программа должна проигрывать фоновую мелодию;

- программа должна отображать монологи главного героя (также озвучку) и повествование игры внизу экрана;

- программа должна обеспечивать возможность пользователю поставить прохождение уровня в режим паузы;

- программа должна сохранять и предоставлять возможность просмотра текущего прогресса для пользователей;

- игра завершается в случае нажатия кнопки “Выход”;

- при полном прохождении игры должна загружаться конечная заставка с титрами.

#### 2.6.3.2. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows 7 или совместимой. Язык интерфейса – русский.

#### 2.6.3.3. Требования к маркировке и упаковке

Программа должна поставляться на диске в виде исполняемого (еxе) файла, документации и проекта. На диске должна быть наклейка с надписью "Игра с графическим интерфейсом «Темный Лес»". Диск должен быть упакован в пластиковую коробку.

### 2.6.4. Требования к программной документации

На испытания должны быть представлены следующие программные документы:

* техническое задание
* текст программы
* описание программы
* руководство оператора
* описание языка

### 2.6.5. Средства и порядок испытаний

Для проведения испытаний необходимы следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~500 Кб;
* объем необходимой памяти на жестком диске ~3Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь».

Для проведения испытаний необходимы следующие программные средства:

Операционная система Windows 7.

Испытания проводятся в следующем порядке:

1) проверяется наличие и комплектность программной документации (п.2.6.4)

2) проверяется соответствие требованиям к маркировке и упаковке (п.2.6.3.3)

3) проверяется соответствие требованиям к функциональным характеристикам (п.2.6.3.1)

4) проверяется соответствие требованиям к информационной и программной совместимости (п.2.6.3.2)

### 2.6.6. Методы испытаний

#### 2.6.6.1. Для проверки способности программы предоставлять возможность передвижения игрока по сцене с применением анимации, нужно:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* нажать клавишу «А», «Space» или «D» на клавиатуре
* убедиться, что персонаж передвигается по сцене с применением анимации

#### 2.6.6.2. Для проверки способности программы осуществлять сбор игровых кристаллов: количество собранных кристаллов отображается в верхнем углу экрана (для прохождения игры нужно собрать определенное количество кристаллов), необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* собрать кристаллы, передвигаясь по сцене
* убедиться, что число собранных кристаллов отображается в левом верхнем углу экрана.

#### 2.6.6.3. Для проверки способности программы гарантировать персонажам возможность атаковать ( при нанесении удара противнику, уровень здоровья врага уменьшается), необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* нажать на клавишу «Q», приблизившись к врагу
* убедиться, что при атаке со стороны персонажа, противник теряет свое здоровье и погибает.

#### 2.6.6.4. Для проверки способности программы предоставлять право пополнения здоровья героя путем лечения лекарственными ягодами в случае атаки со стороны противника, необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* передвигаясь по сцене, приблизиться к врагу, получить урон от удара противника
* собрать лекарственные ягоды, передвигаясь по сцене
* убедиться, что шкала здоровья в левом верхнем углу экрана пополняется.

#### 2.6.6.5. Для проверки способности программы возобновлять прохождение уровня сначала в случае гибели персонажа (полного опустошения шкалы здоровья), необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* передвигаясь по сцене, приблизиться к врагу, получить урон от удара противника
* дождаться полного опустошения шкалы здоровья персонажа
* убедиться, что прохождение уровня возобновилось сначала.

#### 2.6.6.6. Для проверки способности программы обеспечивать переход игрока на новый уровень, необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* полностью пройти уровень, дойти до конца сцены
* перейти за пределы уровня
* убедиться, что в случае, если собраны все кристаллы на сцене, осуществляется переход на новый уровень.

2.6.6.7. Для проверки способности программы открывать главное меню при запуске игры, необходимо:

* запустить программу
* убедиться, в том, что главное меню открылось.

2.6.6.8. Для проверки способности программы осуществлять предоставить пользователю возможность настроить характеристики игры: звук, качество графики, разрешение экрана, необходимо:

* запустить программу
* нажать кнопку «Настройки»
* выбрать нужное разрешение экрана
* изменить качество графики игры
* поставить/убрать приложение в полноэкранный режим
* убавить/прибавить звук.
* убедиться, что все выбранные настройки применяются.

2.6.6.9. Для проверки способности программы перед началом каждого уровня игры отображать знак загрузки, нужно:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* убедиться, что знак загрузки отображается.

2.6.6.10. Для проверки способности программы во время прохождения игры проигрывать фоновую мелодию, необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* убедиться, что во время прохождения уровня, на фоне играет музыка.

2.6.6.11. Для проверки способности программы отображать монологи главного героя (также озвучку) и повествование игры внизу экрана, необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* убедиться, что во время прохождения уровня, на некоторых позициях появляется диалоговое окно, в котором отображается монологи персонажа, сюжет и подсказки
* убедиться, что во время монолога проигрывается озвучка персонажа.

2.6.6.12. Для проверки способности программы обеспечивать возможность пользователю поставить прохождение уровня в режим паузы, необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* убедиться, что во время прохождения уровня, нажав на «ESC», игру можно поставить в режим паузы, из которого можно продолжить игру, перейти в главное меню или же выйти из игры.

2.6.6.13. Для проверки способности программы сохранять и предоставлять возможность просмотра текущего прогресса для пользователей, необходимо:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* пройти уровни
* выйти в загрузку уровней
* убедиться, что прогресс пройденных уровней сохраняется
* убедиться, что при нажатии клавиши «Начать сначала» уровни, начиная со второго, будут закрыты, что даст возможность пройти игру другим пользователям.

2.6.6.14. Для проверки способности программы завершать игру в случае нажатия кнопки «Выход», необходимо:

* запустить программу
* выбрать в главном меню или же в режиме паузы кнопку «Выход»
* убедиться, что игра завершается, приложение прекращает свою работу

2.6.6.15. Для проверки способности программы загружать конечную заставку с титрами при полном прохождении игры, нужно:

* запустить программу
* нажать кнопку «Играть»
* нажать кнопку «Загрузить»
* выбрать один из трех уровней, нажать на иконку выбранного уровня
* пройти все уровни
* убедиться, что при полном прохождении игры отобразиться конечная заставка с титрами.

## 2.7. Протокол испытаний

Результаты испытаний программы представлены в таблице 1, рисунки приведены в приложении Б.

**Результаты испытаний программы**

*Таблица 1*

|  |  |  |
| --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Результаты** |
| Способность программы осуществлять предоставлять возможность передвижения игрока по сцене с применением анимации | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * нажать клавишу «А», «Space» или «D» на клавиатуре | Открывается сцена уровня, персонаж передвигается по сцене с применением анимации –  смотри Рис. Б1 |
| Способность программы осуществлять сбор игровых кристаллов: количество собранных кристаллов отображается в верхнем углу экрана собрать определенное | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * собрать кристаллы, передвигаясь по сцене | Число собранных кристаллов отображается в левом верхнем углу экрана –  смотри Рис. Б2 |
| Способность программы гарантировать персонажам возможность атаковать (при нанесении удара противнику, уровень здоровья врага уменьшается) | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * нажать на клавишу «Q», приблизившись к врагу | При атаке со стороны персонажа, противник теряет свое здоровье и погибает –  смотри Рис. Б3 |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Результаты** |
| Способность программы предоставлять право пополнения здоровья героя путем лечения лекарственными ягодами в случае атаки со стороны противника | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * передвигаясь по сцене, приблизиться к врагу, получить урон от удара противника * собрать лекарственные ягоды, передвигаясь по сцене | Шкала здоровья в левом верхнем углу экрана пополняется –  смотри Рис. Б4 |
| Способность программы возобновлять прохождение уровня сначала в случае гибели персонажа (полного опустошения шкалы здоровья) | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * передвигаясь по сцене, приблизиться к врагу, получить урон от удара противника * дождаться полного опустошения шкалы здоровья персонажа | Прохождение уровня возобновилось сначала, пользователь может продолжить работу – смотри Рис. Б5 |
| Способность программы обеспечивать переход игрока на новый уровень | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня | В случае, если собраны все кристаллы на сцене, осуществляется переход на новый уровень – смотри Рис. Б6, Б6.1 |
| **Проверяемые требования** | **Сообщения программы**  **и вводимые значения** | **Результаты** |
|  | * полностью пройти уровень, дойти до конца сцены * перейти за пределы уровня |  |
| Способность программы открывать главное меню при запуске игры | * запустить программу | При запуске приложения открывается главное меню - смотри Рис. Б7 |
| Способность программы осуществлять предоставить пользователю возможность настроить характеристики игры: звук, качество графики, разрешение экрана | * запустить программу * нажать кнопку «Настройки» * выбрать нужное разрешение экрана * изменить качество графики игры * поставить/убрать приложение в полноэкранный режим * убавить/прибавить звук | Все выбранные настройки применяются к интерфейсу и звуку игры – смотри Рис. Б8 |
| Способность программы перед началом каждого уровня игры отображать знак загрузки | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня | Знак загрузки отображается перед началом уровня – смотри Рис. Б9 |
| Способность программы перед во время прохождения уровня проигрывать фоновую мелодию | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня | Во время прохождения уровня, на фоне играет музыка |
| **Проверяемые требования** | **Сообщения программы**  **и вводимые значения** | **Результаты** |
| Способность программы отображать монологи главного героя, повествование и озвучку героя | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * убедиться, что во время прохождения уровня, на некоторых позициях сцены появляется диалоговое окно, в котором отображается монологи персонажа, текст сюжета и подсказки | Во время монолога проигрывается озвучка персонажа, при нажатии на кнопку «ОК» на панели диалогового окна, появляется следующий текст повествования – смотри Рис.Б10 |
| Способность программы обеспечивать возможность пользователю поставить прохождение уровня в режим паузы | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня | Во время прохождения уровня, нажав на «ESC», игру можно поставить в режим паузы, из которого можно продолжить игру, перейти в главное меню или же выйти из игры – смотри Рис.Б11 |
| Способность сохранять и предоставлять возможность просмотра текущего прогресса для пользователей | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * пройти уровни * выйти в меню загрузки уровней | Текущий прогресс пользователя сохраняется, при нажатии клавиши «Начать сначала» уровни, начиная со второго, будут закрыты, что даст возможность пройти игру другим пользователям – смотри Рис.Б12 |

|  |  |  |
| --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Результаты** |
| Способность программы завершать игру в случае нажатия кнопки «Выход» | * запустить программу * выбрать в главном меню или же в режиме паузы кнопку «Выход» | Приложение закрывается, программа завершает свою работу. |
| Способность программы загружать конечную заставку с титрами при полном прохождении игры | * запустить программу * нажать кнопку «Играть» * нажать кнопку «Загрузить» * выбрать один из трех уровней, нажать на иконку выбранного уровня * пройти все уровни | При полном прохождении всех уровней игры открывается сцена с титрами, приложение не закрывается, пользователь может выбрать пункты: «Главное меню» или «Выход» – смотри Рис. Б13 |

# ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения курсового проекта программа удовлетворяет всем требованиям технического задания, что подтверждается протоколом испытаний.

Разработанная программа может быть использована в развлекательных целях, для отдыха и релаксации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Термин «компьютерная игра»: https://studbooks.net/2263023/informatika/ponyatie\_kompyuternoy\_igry\_klassifikatsii
2. Жанры: <https://ru.wikipedia.org/wiki/Классификация_компьютерных_игр>; https://habr.com/ru/post/265911/
3. Описание языка C#: https://ru.wikipedia.org/wiki/C\_Sharp; https://metanit.com/sharp/tutorial/2.16.php
4. Материалы для игры: https://vk.com/away.php?utf=1&to=https%3A%2F%2Fassetstore.unity.com%2Fpackages%2F2d%2FFenvironment%2Fcrystal-world-platformer-150016; https://assetstore.unity.com/packages/2d/characters/monsters-creatures-fantasy-167949; https://free4kwallpapers.com/uploads/wallpaper/purple-forest-wallpaper-1024x768-wallpaper.jpg; https://vk.com/away.php?utf=1&to=https%3A%2F%2Fassetstore.unity.com%2Fpackages%2F2d%2FFcharacter%2Fmonsters-creatures-fantasy-167949; https://img4.goodfon.ru/original/1920x1080/8/af/vecher-firewatch-campo-santo-kholmy-vid-les-igra-peizazh-g-1.jpg; https://assetstore.unity.com/packages/2d/environments/pixel-forest-136825; https://assetstore.unity.com/packages/2d/characters/2d-pixel-art-pack-rousette-167698
5. Стандарты Единой Системы Программной Документации:

ГОСТ 19.105-78 Общие требования к программным документам

ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом

ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению

ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

ПРИЛОЖЕНИЕ А

Министерство науки и ВЫСШЕГО образования Российской Федерации  
Федеральное автономное образовательное учреждение высшего профессионального образования  
«Санкт-Петербургский государственный политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ)  
**Институт среднего профессионального образования**

УТВЕРЖДАЮ  
Председатель ПЦК   
\_\_\_\_\_\_\_\_\_\_\_ Костин Ю.Н.  
\_\_.\_\_. 2020

Игра с графическим интерфейсом «Темный лес»  
**Техническое задание**  
Листов 7

ПРОВЕРИЛ  
Преподаватель   
\_\_\_\_\_\_\_\_\_\_\_ Девятко Н.С.  
\_\_.\_\_. 2020

ВЫПОЛНИЛИ  
Студенты группы 32928/1   
\_\_\_\_\_\_\_\_\_\_\_ Дмитриева В.В. \_\_\_\_\_\_\_\_\_\_\_ Петрова А.В.  
\_\_.\_\_. 2020

2020

# ВВЕДЕНИЕ

* 1. Полное наименование программной разработки: «ИГРА С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ «Темный лес».
  2. Игровая программа с графическим интерфейсом «Темный лес»: человек играет за сказочного персонажа; цель персонажа – освободить лес от демонов и восстановить баланс в природе; для выполнения миссии главный герой сражается с монстрами и собирает кристаллы жизни.
  3. Программа предназначена для развлекательных целей, для отдыха и проведения досуга.

# ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.02 «Прикладное программирование» и утверждена Институтом среднего профессионального образования.

# НАЗНАЧЕНИЕ РАЗРАБОТКИ

* 1. Основное назначение программного продукта: компьютерная игра в жанре «Платформер».
  2. Эксплуатационное назначение программного продукта: программа предназначена для широкого круга пользователей, без ограничения по возрасту, не требует внесения денежных средств или использования платёжных систем для игры, предназначена для развлекательных целей.

4. ТРЕБОВАНИЯ К РАЗРАБОТКЕ

4.1. Требования к функциональным характеристикам:

**1 версия:**

* программа должна предоставлять возможность передвижения игрока по сцене с применением анимации;
* программа должна осуществлять сбор игровых кристаллов: количество собранных кристаллов отображается в верхнем углу экрана (для прохождения игры нужно собрать определенное количество кристаллов);
* программа обязана гарантировать персонажам возможность атаковать: при нанесении удара противнику, уровень здоровья врага уменьшается;
* программа должна предоставлять право пополнения здоровья героя путем лечения лекарственными ягодами в случае атаки со стороны противника;
* программа обязана показывать шкалу здоровья при ее пополнении и уроне;
* в случае гибели персонажа (полного опустошения шкалы здоровья) прохождение уровня возобновляется сначала;
* программа должна обеспечивать переход игрока на новый уровень.

**2 версия:**

* программа должна открывать главное меню при запуске игры;
* программа обязана предоставить пользователю возможность настроить характеристики игры: звук, качество графики, разрешение экрана;
* перед началом каждого уровня игры программа должна отображать знак загрузки;
* во время прохождения игры программа должна проигрывать фоновую мелодию;
* программа должна отображать монологи главного героя (также озвучку) и повествование игры внизу экрана;
* программа должна обеспечивать возможность пользователю поставить прохождение уровня в режим паузы;
* программа должна сохранять и предоставлять возможность просмотра текущего прогресса для пользователей;
* игра завершается в случае нажатия кнопки “Выход”;
* при полном прохождении игры должна загружаться конечная заставка с титрами.

4.2. Требования к надежности:

* использование лицензированного программного обеспечения;
* проверка программы на наличие вирусов;
* организация бесперебойного питания.

4.3. Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~20Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

4.4. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows XP или совместимой. Язык интерфейса – русский.

4.5. Требования к маркировке и упаковке

Программа должна поставляться на диске в виде исполняемого (еxе) файла, документации и проекта. На диске должна быть наклейка с надписью: "Темный лес". Диск должна быть упакован в пластиковую коробку.

4.6. Требования к транспортировке и хранению

Диск с программой должен храниться вдали от электромагнитных полей и не подвергаться механической деформации. Место и условия хранения должны соответствовать санитарным требованиям отрасли. Сроки хранения устанавливаются в соответствии с гарантийными сроками поставщика магнитных носителей.

Основные требования к транспортировке – создание условий, исключающих механические повреждения магнитного носителя.

4.7. Специальные требования

Теоретическая часть включает подробное описание работы с приложением, показывающими схему работы системы. Практическая часть включает разработку и реализацию с использованием среды программирования C# программных модулей программного продукта.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Предварительный состав программной документации:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

5.2. Перечень материалов пояснительной записки

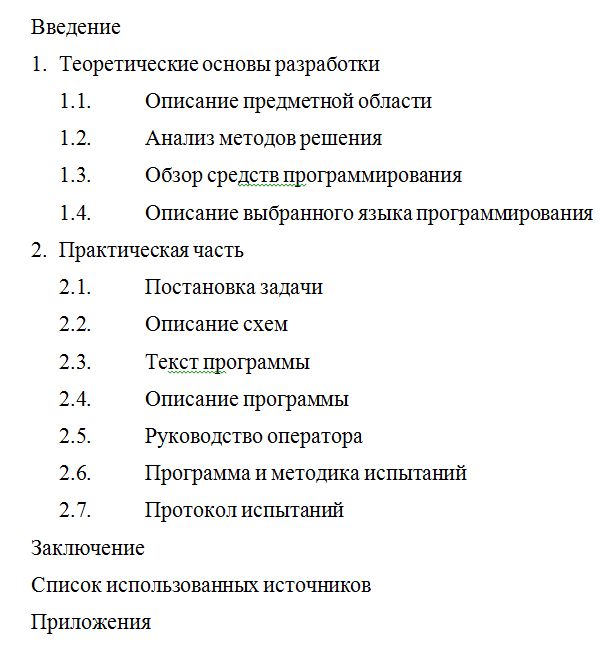


Рисунок 1 – Перечень материалов пояснительной записки

6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели не рассчитываются.

7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание стадии** | **Содержание этапа** | **Срок 2020 г.** | | **Форма  отчетности** |
| **начало** | **конец** |
| Техническое задание | Составление технического задания | 15.10 | 21.10 | Техническое задание |
| Эскизный проект | Разработка спецификаций | 22.10 | 28.10 | Спецификации программного обеспечения |
| Рабочий проект | Проектирование программы | 29.10 | 11.11 | Схема работы системы и спецификации компонентов |
| Составление программы | 12.11 | 25.11 | Программная документация |
| Приёмо-сдаточные испытания | 26.11 | 09.12 | Протокол испытаний (п. 2.7 пояснительной записки) |
| Приёмка | Защита курсового проекта | 10.12 | 15.12 | Оценка за курсовой проект |

8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1. Порядок контроля

Контроль выполнения должен осуществляться руководителем курсового проекта (преподавателем) в соответствие с п.7.

8.2. Порядок приемки

Приемка должна осуществляться с участием руководителя после проведения приемо-сдаточных испытаний. В результате защиты курсового проекта должна быть выставлена оценка за курсовой проект.

ПРИЛОЖЕНИЕ Б



Рисунок Б1 Передвижение персонажа

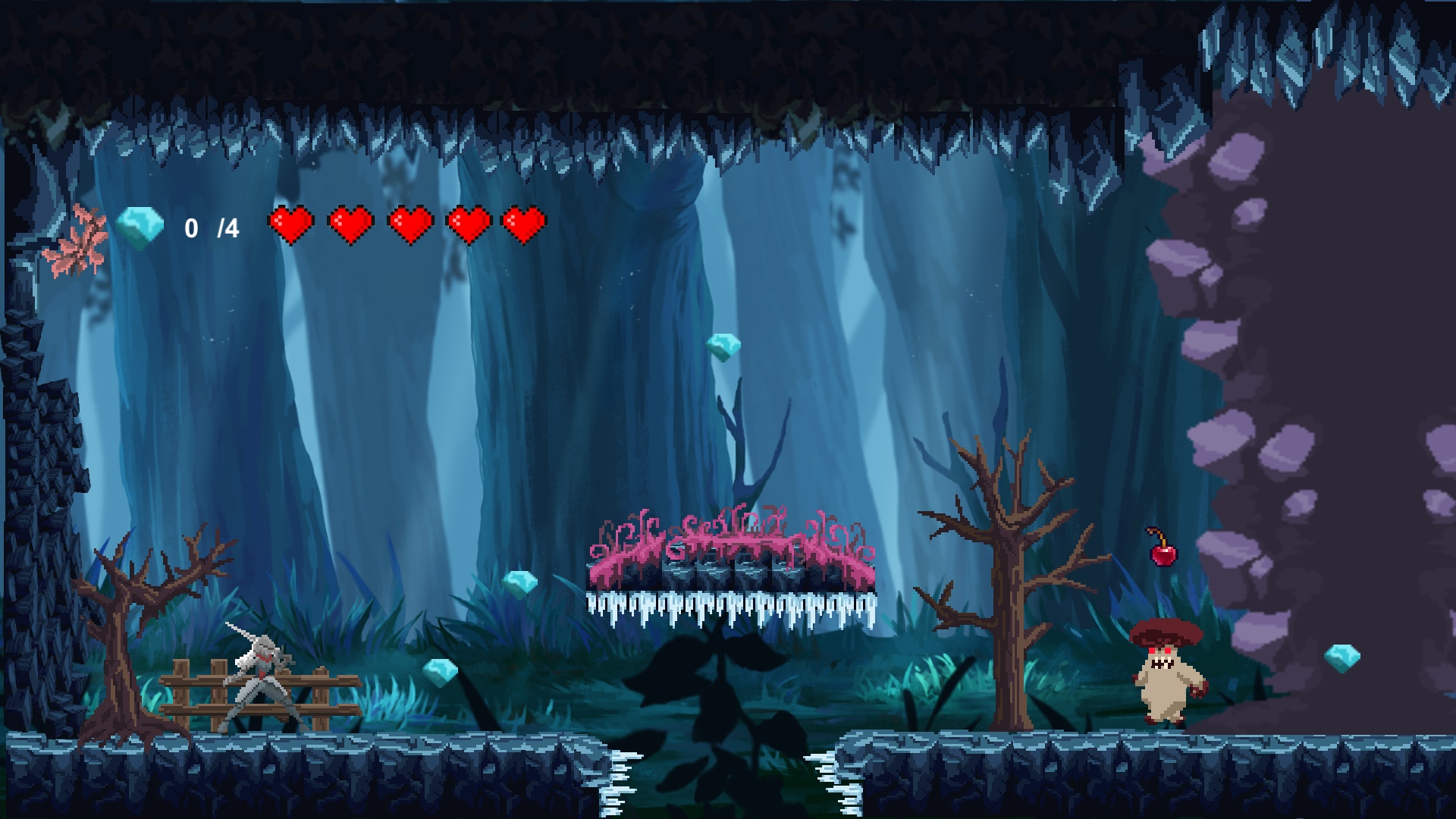


Рисунок Б2 Отображение количества собранных кристаллов



Рисунок Б3 Атака

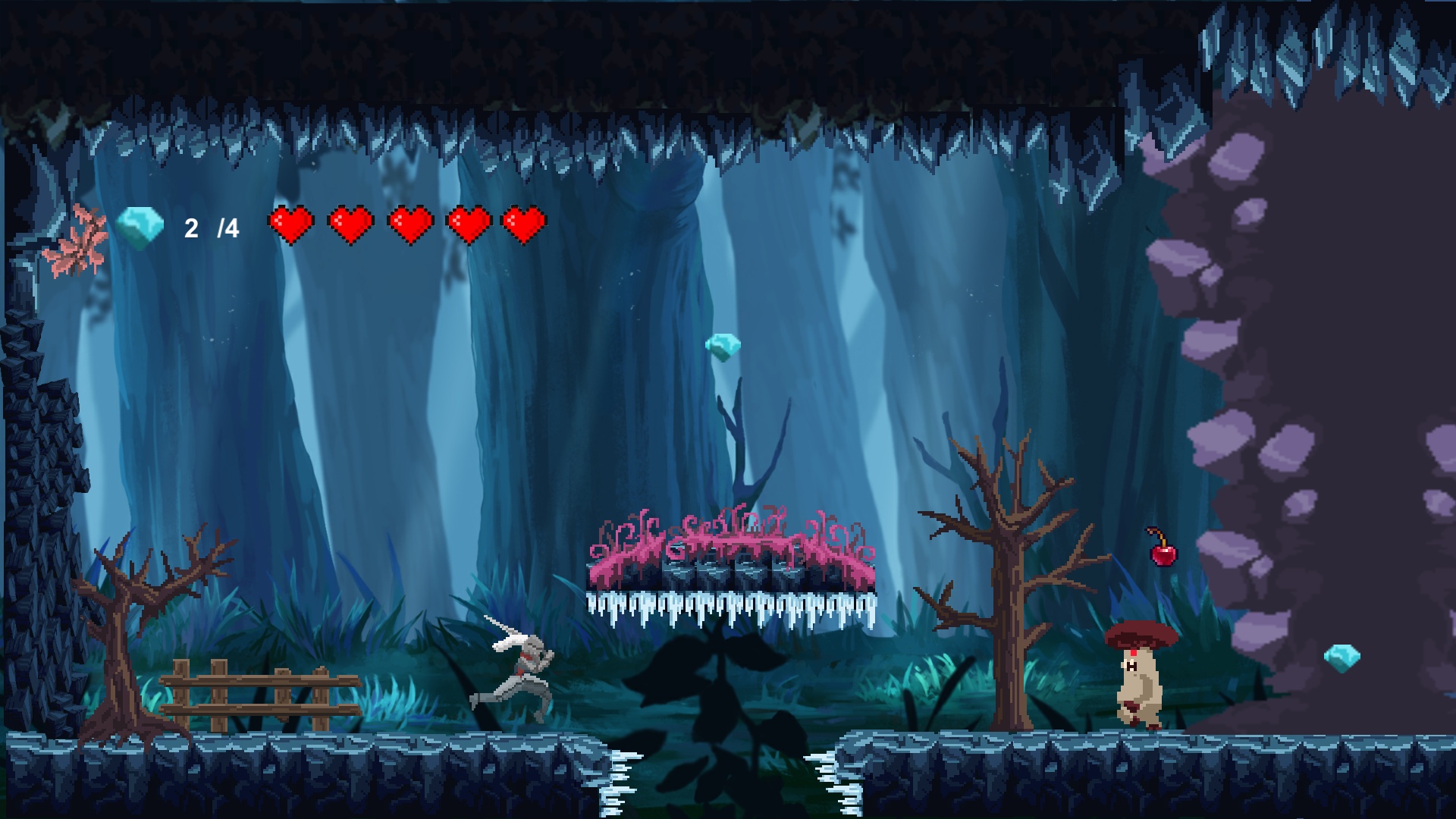


Рисунок Б4 Шкала здоровья

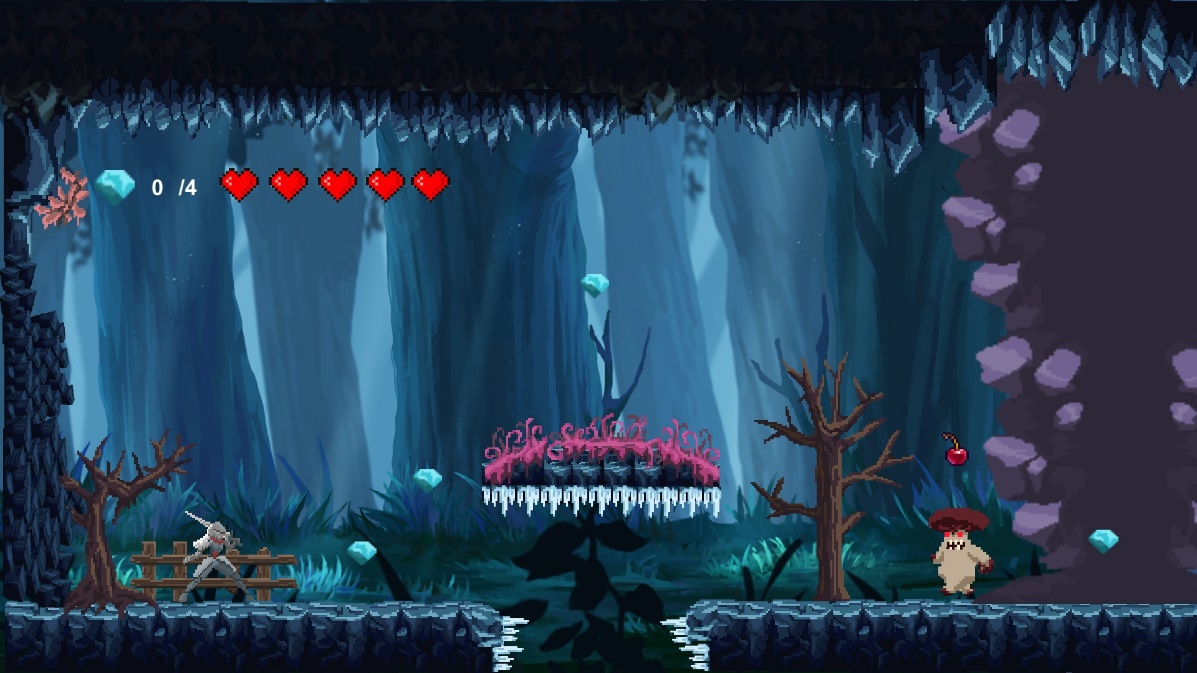


Рисунок Б5 Возобновление прохождения

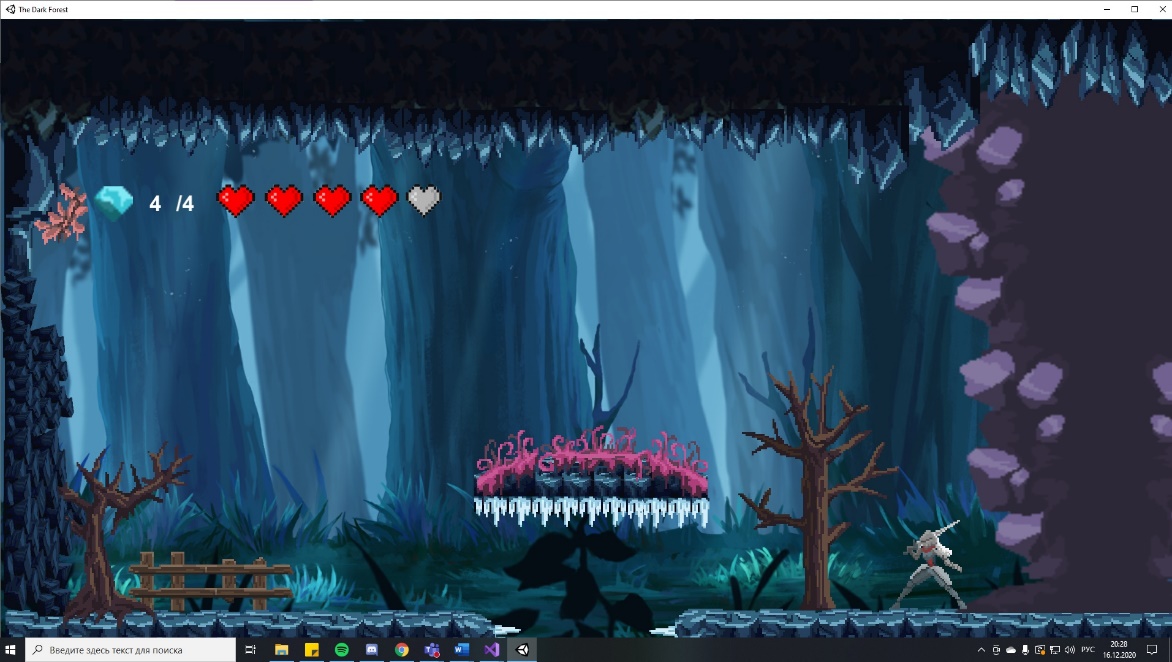


Рисунок Б6 Осуществление сбора всех кристаллов

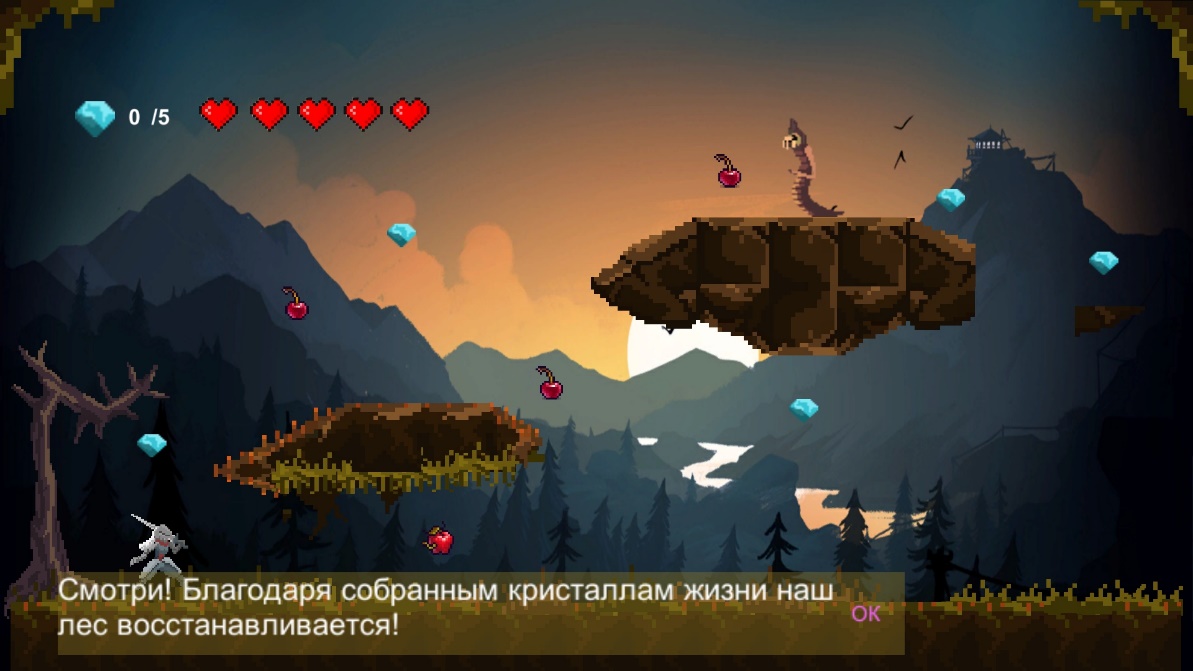


Рисунок Б6.1 Переход на следующий уровень



Рисунок Б7 Главное меню

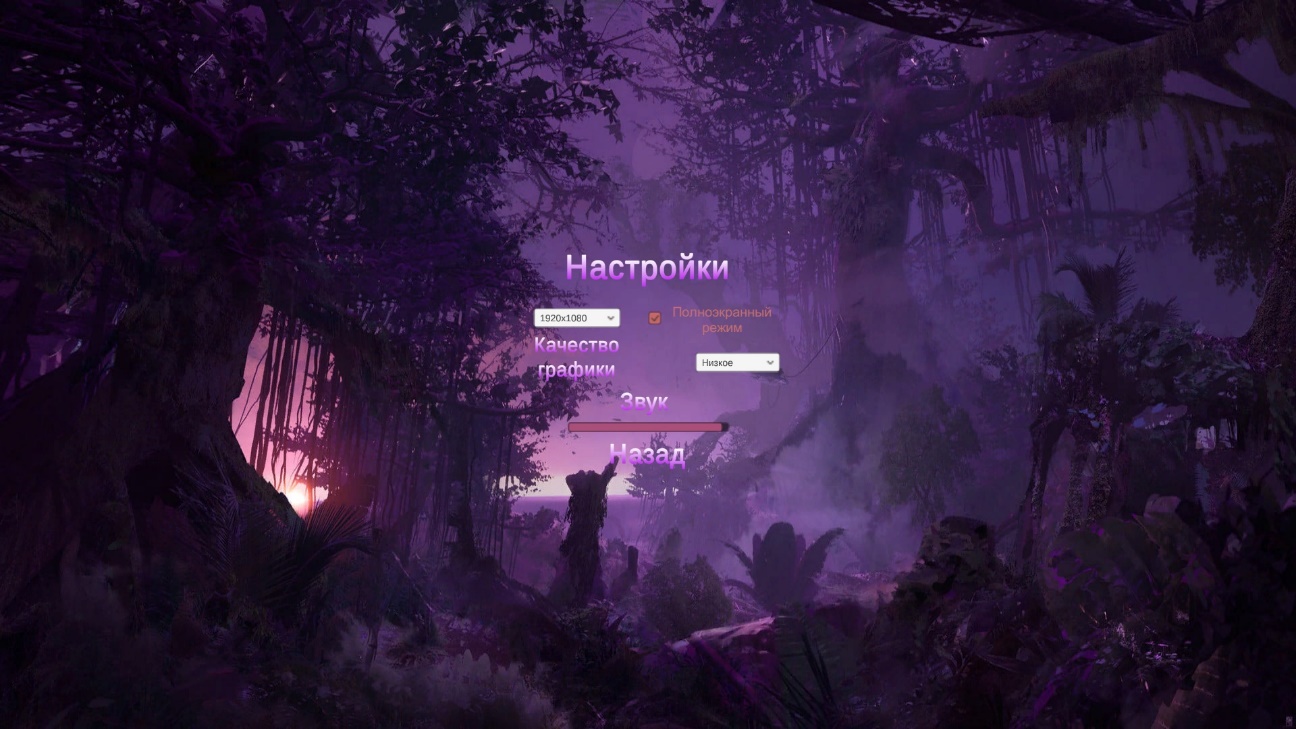


Рисунок Б8 Меню настроек

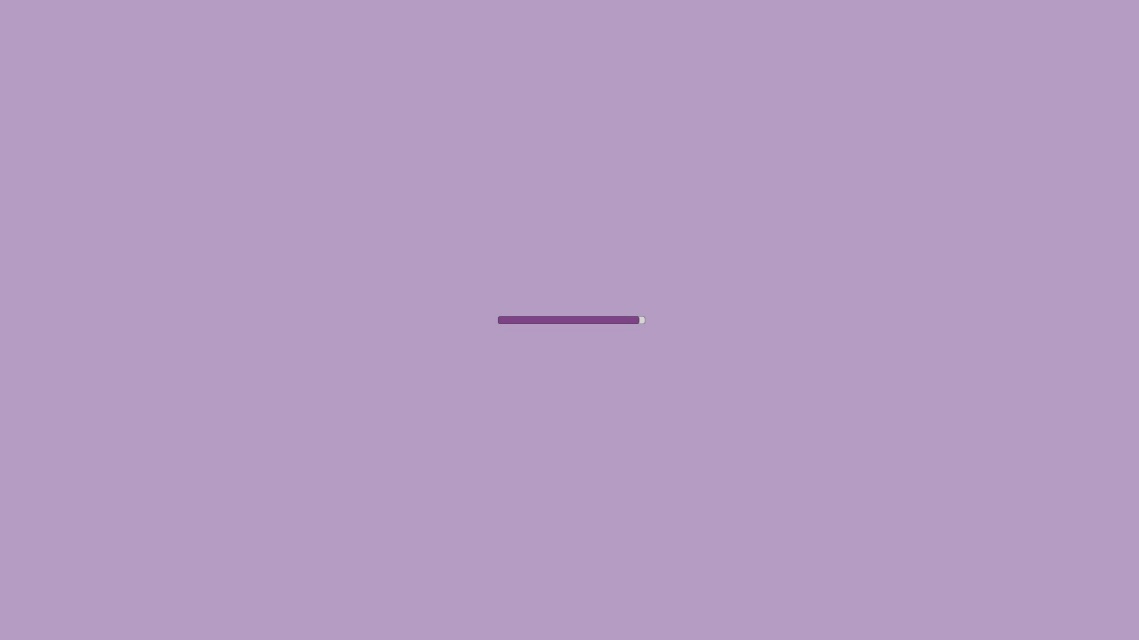


Рисунок Б9 Знак загрузки



Рисунок Б10 Монологи

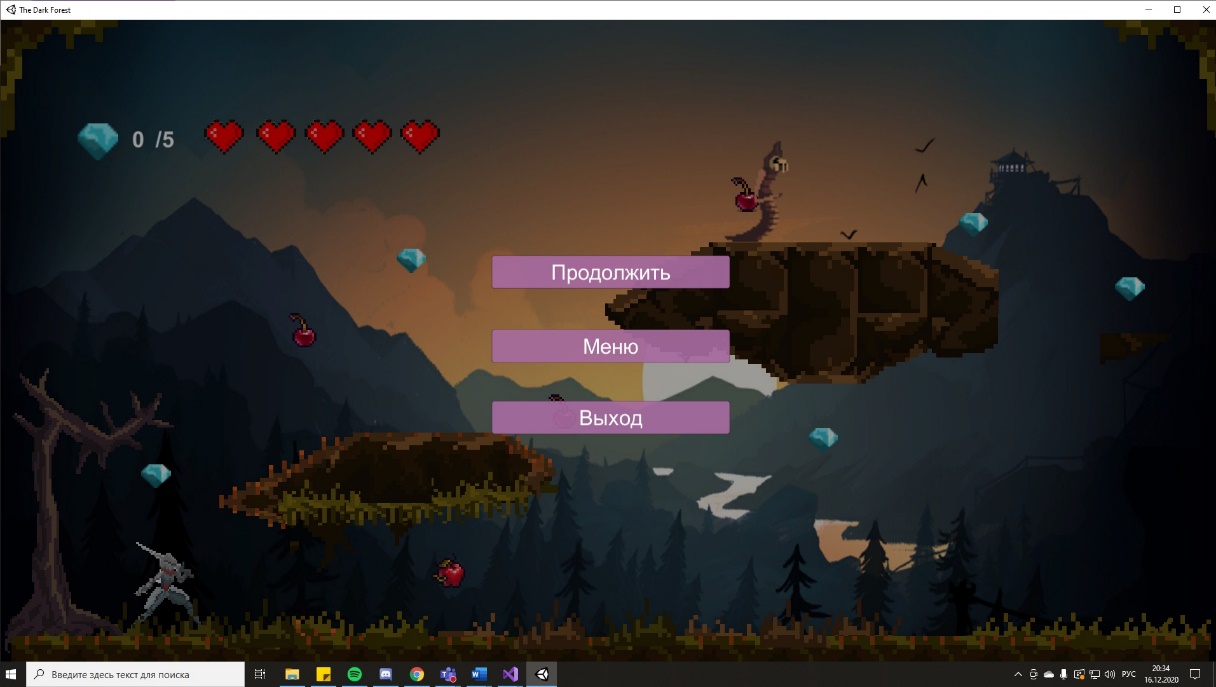


Рисунок Б11 Меню паузы

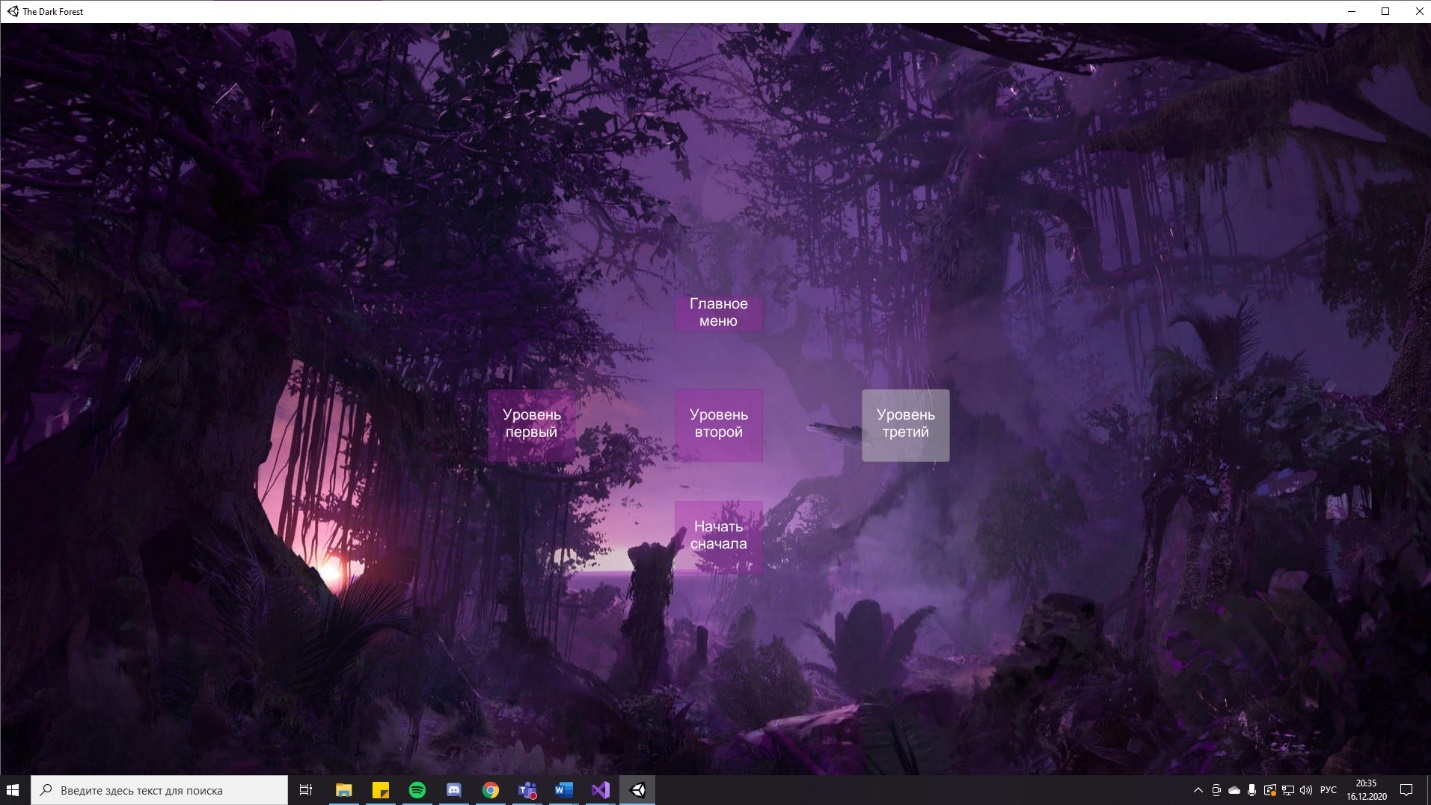


Рисунок Б12 Сохранение прогресса



Рисунок Б13 Титры

ПРИЛОЖЕНИЕ В

**Текст программы**

**Модуль HeroController.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Audio;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class HeroController : MonoBehaviour {

public HeroController hero;

private Animator animator;

public static Rigidbody2D rb;

SpriteRenderer sprite;

private SpriteRenderer sr;

public float jumpForce;

private bool isGrounded=false;

public Transform groundcheck;

private float groundRadius=0.2f;

public LayerMask whatIsGround; //jump

readonly Vector2 force = new Vector2(50, 50);

[SerializeField] int speed=0; //run

public float health;

public int numOfHEarts;

public Image[] hearts;

public Sprite fullHeart;

public Sprite emptyHeart;

// Use this for initialization

private void Start () {

animator = GetComponent<Animator> ();

rb = GetComponent<Rigidbody2D> ();

sprite = GetComponent<SpriteRenderer> ();

sr = GetComponent<SpriteRenderer>();

// attackpose.SetActive (false);

}

// Update is called once per frame

void Update () {

if (Input.GetKey (KeyCode.D) || (Input.GetKey (KeyCode.A))) {

{

animator.SetBool("bool", true);

}

} else {

animator.SetBool ("bool", false);

}

if (isGrounded && Input.GetKeyDown (KeyCode.Space)) {

animator.SetBool ("ground", false);

rb.AddForce (new Vector2 (0, 4300));

}

}

private void FixedUpdate()

{

Run ();

Jump ();

if (health > numOfHEarts)

{

health = numOfHEarts;

}

for (int i = 0; i < hearts.Length; i++)

{

if (i < Mathf.RoundToInt(health))

{

hearts[i].sprite = fullHeart;

}

else

{

hearts[i].sprite = emptyHeart;

}

if (i < numOfHEarts)

{

hearts[i].enabled = true;

}

else

{

hearts[i].enabled = false;

}

}

}

void OnTriggerEnter2D(Collider2D collision)

{

if (collision.gameObject.tag == "life") {

health++;

Destroy (collision.gameObject);

}

if (collision.gameObject.tag == "BadFood")

{

health--;

Destroy(collision.gameObject);

if (health <= 0)

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

}

if (collision.tag.Equals ("crystall")) {

Crystall.crystallCount += 1;

Destroy (collision.gameObject);

}

}

public void Run() {

if (Input.GetKey (KeyCode.D)) { //если нажимаем на кнопку D, то герой идет направо

{

if (Input.GetAxis ("Horizontal") > 0) {

rb.velocity = new Vector2 (speed, rb.velocity.y);

sprite.flipX = false;

}

}

} else if (Input.GetKey (KeyCode.A)) { //если нажимаем на кнопку A, то герой идет налево

{

if (Input.GetAxis ("Horizontal") < 0) {

rb.velocity = new Vector2 (-speed, rb.velocity.y);

sprite.flipX = true;

}

}

}

}

public void Jump ()

{

isGrounded = Physics2D.OverlapCircle (groundcheck.position, groundRadius, whatIsGround);

animator.SetBool ("ground", isGrounded);

animator.SetFloat ("vspeed", rb.velocity.y);

if (!isGrounded)

return;

}

}

**Модуль Attack.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Attack : MonoBehaviour {

private float timeBtwAttack;

public float startTimeBtwAttack;

public Transform attackPose;

public float attackRange;

public LayerMask whatIsEnemy;

public int damage;

private Animator animator;

public AudioClip din;

AudioSource audio;

public void PlayAudio(AudioClip clip)

{

GetComponent<AudioSource>().PlayOneShot(clip);

}

void Start () {

animator = GetComponent<Animator> ();

}

void Update () {

if (timeBtwAttack <= 0) {

if (Input.GetKey(KeyCode.Q))

{

animator.SetTrigger("attack1");

PlayAudio(din);

}

timeBtwAttack = startTimeBtwAttack;

} else {

timeBtwAttack -= Time.deltaTime;

}

}

public void OnAttack(){

Collider2D[] enemiesToDamage = Physics2D.OverlapCircleAll (attackPose.position, attackRange,whatIsEnemy);

for (int i = 0; i < enemiesToDamage.Length; i++) {

enemiesToDamage [i].GetComponent<Enemy> ().TakeDamage (damage);

}

}

void OnDrawGizmosSelected()

{

Gizmos.color = Color.red;

Gizmos.DrawWireSphere (attackPose.position,attackRange);

}

}

**Модуль Crystall.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Crystall : MonoBehaviour

{

public static int crystallCount;

public Text crystallCounter;

// Use this for initialization

void Start()

{

crystallCounter = GetComponent<Text>();

crystallCount = 0;

}

// Update is called once per frame

void Update()

{

crystallCounter.text = " " + crystallCount;

}

}

**Модуль CrystallCollection.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CrystallCollection : MonoBehaviour

{

void OnTriggerEnter2D(Collider2D collision)

{

if (collision.CompareTag("Player"))

{

Crystall.crystallCount ++;

PlayerPrefs.SetInt("crystall", Crystall.crystallCount);

Destroy(gameObject);

}

}

}

**Модуль DieSpace:**

using System.Collections.Generic;

using UnityEngine.SceneManagement;

using UnityEngine;

public class DieSpace : MonoBehaviour

{

void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Player")

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

}

}

**Модуль Enemy.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Enemy : MonoBehaviour

{

private Animator anim;

public int health;

public int damage;

private HeroController hero;

private SpriteRenderer spriteRend;

private void Start()

{

spriteRend = GetComponent<SpriteRenderer>();

anim = GetComponent<Animator>();

hero = FindObjectOfType<HeroController>();

}

public void TakeDamage(int damage)

{

health--;

}

void Update()

{

if (health <= 0)

{

Destroy(gameObject);

}

}

public void OnEnemyAttack()

{

hero.health -= damage;

if (hero.health <= 0)

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

}

private void OnTriggerStay2D(Collider2D other)

{

if (other.CompareTag("Player"))

{

anim.SetTrigger("AttackEnemy");

}

}

}

**Модуль LevelLoader.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class LevelLoader : MonoBehaviour

{

public GameObject LoadingScreen;

public Slider Slider;

public void LoadLevel(int SceneIndex)

{

StartCoroutine(LoadAsynchronously(SceneIndex));

}

IEnumerator LoadAsynchronously(int SceneIndex)

{

AsyncOperation operation = SceneManager.LoadSceneAsync(SceneIndex);

LoadingScreen.SetActive(true);

while(!operation.isDone)

{

float progress = Mathf.Clamp01(operation.progress / .9f);

Slider.value = progress;

yield return null;

}

}

}

**Модуль LevelManager1.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class LevelManager1 : MonoBehaviour

{

int levelUnLock;

public Button[] buttons;

void Start()

{

levelUnLock = PlayerPrefs.GetInt("levels", 3);//Дефолтный левел - 3, порядок сцены

for (int i = 0; i < buttons.Length; i++)

{

if (i + 3 > levelUnLock)

buttons[i].interactable = false;

}

}

public void loadLevel(int levelIndex)

{

SceneManager.LoadScene(levelIndex);

}

public void Reset()

{

PlayerPrefs.DeleteAll();

for (int i = 1; i < buttons.Length; i++)

{

buttons[i].interactable = false;

}

}

}

**Модуль MainMenu.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour

{

public void Menu()

{

SceneManager.LoadScene(0);

}

public void PlayGame()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

public void ExitGame()

{

Debug.Log("Quit");

Application.Quit();

}

}

**Модуль NextLevel.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class NextLevel : MonoBehaviour

{

public GameObject LoadingScreen;

public Slider Slider;

void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Player")

{

LoadLevel(SceneManager.GetActiveScene().buildIndex + 1);

//SceneManager.LoadScene(3);

}

}

public void LoadLevel(int SceneIndex)

{

StartCoroutine(LoadAsynchronously(SceneIndex));

}

IEnumerator LoadAsynchronously(int SceneIndex)

{

AsyncOperation operation = SceneManager.LoadSceneAsync(SceneIndex);

LoadingScreen.SetActive(true);

while (!operation.isDone)

{

float progress = Mathf.Clamp01(operation.progress / .9f);

Slider.value = progress;

yield return null;

}

}

}

**Модуль MessageBox.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class messagebox : MonoBehaviour

{

private GameObject MessageSign;

private Text MessageText;

private BoxCollider2D bc;

private Rigidbody2D rb;

public string[] texts;

public int messagesShown = 0;

bool messageShown = false;

void Awake()

{

MessageSign = GameObject.Find("MessageBox");

MessageText = GameObject.Find("MessageText").GetComponent<Text>();

}

void Start()

{

MessageSign.SetActive(false);

}

void OnTriggerEnter2D(Collider2D col)

{

if (col.tag == "Player")

ShowMessageBox();

}

public void ShowMessageBox()

{

MessageSign.SetActive(true);

MessageText.text = texts[messagesShown];

MessageText.enabled = true;

messageShown = true;

}

public void Accept()

{

if (messageShown)

{

hideMessageBox();

messagesShown++;

if (messagesShown < texts.Length)

{

ShowMessageBox();

}

else

Destroy(gameObject);

}

}

public void hideMessageBox()

{

MessageSign.SetActive(false);

MessageText.enabled = false;

messageShown = false;

}

}

**Модуль Patrol.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Patrol : MonoBehaviour

{

private Animator anim;

public int positionOfPatrul;

public Transform point;

bool moveingRight;

SpriteRenderer sprite;

Transform hero;

public float speed;

public float stoppingDistance;

bool chill = false;

bool angry = false;

bool goBack = false;

// Use this for initialization

void Start()

{

anim = GetComponent<Animator>();

sprite = GetComponent<SpriteRenderer>();

hero = GameObject.FindGameObjectWithTag("Player").transform;

}

// Update is called once per frame

void Update()

{

if (Vector2.Distance(transform.position, point.position) < positionOfPatrul && angry == false)

{

chill = true;

}

if (Vector2.Distance(transform.position, hero.position) < stoppingDistance)

{

angry = true;

chill = false;

goBack = false;

}

if (Vector2.Distance(transform.position, hero.position) > stoppingDistance)

{

goBack = true;

angry = false;

}

if (chill == true)

{

Chill();

}

else if (angry == true)

{

Angry();

}

else if (goBack == true)

{

GoBack();

}

}

void Chill()

{

if (transform.position.x > point.position.x + positionOfPatrul)

{

moveingRight = false;

sprite.flipX = false;

}

else if (transform.position.x < point.position.x - positionOfPatrul)

{

moveingRight = true;

sprite.flipX = true;

}

if (moveingRight)

{

sprite.flipX = true;

transform.position = new Vector2(transform.position.x + speed \* Time.deltaTime, transform.position.y);

}

else

{

transform.position = new Vector2(transform.position.x - speed \* Time.deltaTime, transform.position.y);

}

}

void Angry()

{

sprite.flipX = false;

transform.position = Vector2.MoveTowards(transform.position, hero.position, speed \* Time.deltaTime);

}

void GoBack()

{

sprite.flipX = true;

transform.position = Vector2.MoveTowards(transform.position, point.position, speed \* Time.deltaTime);

}

}

**Модуль SettingMenu.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Audio;

using UnityEngine.UI;

using TMPro;

public class SettingsMenu : MonoBehaviour

{

public AudioMixer audioMixer;

public float musicSound = 10f;

public TMP\_Dropdown resolutionDropdown;

Resolution[] resolutions;

void Start()

{

//audioMixer = GetComponent<AudioMixer>();

resolutions = Screen.resolutions;

resolutionDropdown.ClearOptions();

List<string> options = new List<string>();

int currentResolutionIndex = 0;

for (int i = 0; i < resolutions.Length; i++)

{

string option = resolutions[i].width + "x" + resolutions[i].height;

options.Add(option);

if (resolutions[i].width == Screen.currentResolution.width && resolutions[i].height == Screen.currentResolution.height)

{

currentResolutionIndex = i;

}

}

resolutionDropdown.AddOptions(options);

resolutionDropdown.value = currentResolutionIndex;

resolutionDropdown.RefreshShownValue();

}

public void SetVolume(float musicSound)

{

audioMixer.SetFloat("volume", musicSound);

}

public void SetQuality(int QualityIndex)

{

QualitySettings.SetQualityLevel(QualityIndex);

}

public void SetFullScreen(bool isFullscreen)

{

Screen.fullScreen = isFullscreen;

}

public void SetResolution(int resolutionIndex)

{

Resolution resolution = resolutions[resolutionIndex];

Screen.SetResolution(resolution.width, resolution.height, Screen.fullScreen);

}

}

**Модуль PauseMenu.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour

{

public static bool GameIsPaused = false;

public GameObject pauseMenuUI;

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

if (GameIsPaused)

{

Resume();

}

else

{

Pause();

}

}

}

public void Resume()

{

pauseMenuUI.SetActive(false);

Time.timeScale = 1f;

GameIsPaused = false;

}

void Pause()

{

pauseMenuUI.SetActive(true);

Time.timeScale = 0f;

GameIsPaused = true;

}

public void LoadMenu()

{

Debug.Log("Load");

Time.timeScale = 1f;

SceneManager.LoadScene(0);

}

public void QuitGame()

{

//Debug.Log("Quit");

Application.Quit();//Закрытие приложения

}

}