**Содержание**

[**ВВЕДЕНИЕ 3**](#_Toc98122389)

[**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4**](#_Toc98122390)

[**1.1 Предметная Область 4**](#_Toc98122391)

[**1.1.1 Описание предметной области 4**](#_Toc98122392)

[**1.2 Общие сведения 4**](#_Toc98122393)

[**1.3 Определение автоматизируемых задач 5**](#_Toc98122394)

[**1.3.1 Диаграмма прецедентов 5**](#_Toc98122395)

[**1.4 Постановка цели и задач 6**](#_Toc98122396)

[**2 РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ 7**](#_Toc98122397)

[**2.1 Техническое обоснование проекта 7**](#_Toc98122398)

[**2.2 Формирование технического задания 7**](#_Toc98122399)

[**3 РАБОЧЕЕ ПРОЕКТИРОВАНИЕ 8**](#_Toc98122400)

[**3.1 Файловая структура игры 8**](#_Toc98122401)

[**3.2 Создание пользовательского интерфейса 10**](#_Toc98122402)

[**3.3 Реализация программного продукта 13**](#_Toc98122403)

[**3.3.1 Реализация БД 17**](#_Toc98122404)

[**3.3.2 Реализация Противника 18**](#_Toc98122405)

[**3.3.3 Реализация Персонажа 19**](#_Toc98122406)

[**4 ТЕСТИРОВАНИЕ 20**](#_Toc98122407)

[**4.1 Функциональное тестирование 20**](#_Toc98122408)

[**4.2 Юзабилити тестирование. 25**](#_Toc98122409)

[**ЗАКЛЮЧЕНИЕ 26**](#_Toc98122410)

[**СПИСОК ИСПЛЬЗОВАНННОЙ ЛИТЕРАТУРЫ 27**](#_Toc98122411)

[**ПРИЛОЖЕНИЕ 28**](#_Toc98122412)

[**Программный код 28**](#_Toc98122413)

# **ВВЕДЕНИЕ**

Разработка компьютерных игр - процесс создания компьютерных игр. Разработкой видеоигр занимается разработчик, который может быть представлен как одним человеком, так и фирмой. Обычно крупномасштабные коммерческие игры разрабатываются командами разработчиков в пределах компании, специализирующейся на играх для персонального компьютера или консолей. Как правило, разработку финансирует другая, более крупная компания-издатель, которая по окончанию разработки занимается изданием игры и связанными с ним тратами.

Что бы не трать время на разработку основы для игры, таких как игровой движок, разработчики используют готовые решения в виде игрового движка Unity.

Платформа Unity обладает широким спектром возможностей, удобным и интуитивно понятны интерфейсом. Большим преимуществом Unity является кроссплатформенная разработка, которая делает легким и быстрым портирование игр под такие платформы как Windows, iOS, Android, Windows Phone 8, а также разрабатывать игры для Xbox, PS, Nintendo, веб-броузеров и даже VR

С точки зрения организации, данные игры с простым содержанием и увлекательным игровым процессом помогают организациям в удержании аудитории на сайте или приложении на любой платформе.

Такие игры могут быть внедрены в любой существующий проект что делает данную тему актуальной.

Все это в совокупности и обусловило определение цели, задач, объекта и предмета данной работы.

Объектом исследования является компьютерные игры.

Предметом исследования является влияние игрового процесса на интерес к игре.

Целью курсового проекта является создание игры на Unity.

# **1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Предметная Область**

Предметной областью в выпускной квалификационной работе является: Разработка программно-информационного продукта «Компьютерная игра «Fix It»

### **1.1.1 Описание предметной области**

Разработка компьютерных игр - процесс создания компьютерных игр. Разработкой видеоигр занимается разработчик, который может быть представлен как одним человеком, так и фирмой. Обычно крупномасштабные коммерческие игры разрабатываются командами разработчиков в пределах компании, специализирующейся на играх для персонального компьютера или консолей. Как правило, разработку финансирует другая, более крупная компания-издатель, которая по окончанию разработки занимается изданием игры и связанными с ним тратами.

## **1.2 Общие сведения**

Игровое приложение рассчитано на использование одним игроком. Игра начинается с игрового поля, в центре которого находится игрок. Игрок управляет персонажем, которым может ходить в разные стороны. У персонажа присутствует оружие, которое вращается за указателем мыши и может стрелять в выбранном направлении.

В данной игре отсутствуют победные условия. Цель игрока набрать наибольшее количество очков, за уничтожение вражеских объектов, движущихся на встречу игроку. Игра будет окончена если у персонажа останется 0 здоровья. Здоровье персонажа падает на 1 за каждое столкновение с противником.

В процессе игры будут появляться усиления, которые упрощают процесс выживания в игре.

Камера в игре следует за игроком в любом направлении.

## **1.3 Определение автоматизируемых задач**

Основные требования к Игре

Игра должна состоять из отдельных подсистем, способных работать, как единый программный продукт с использованием базы данных, так и самостоятельно.

Игра должна:

1. Быть оптимизированной;
2. Заносить информацию в БД;
3. Иметь удобное управление;

В игре должны быть реализованы функции, авторизации, игрового процесса, формирование отчетов для улучшения игры.

### **1.3.1 Диаграмма прецедентов**

Диаграмма прецедентов использования предназначена для описания функционального назначения игры. Основные элементы диаграммы – варианты ее использования и актеры, потребности которых они удовлетворяют (См. Рис. 1)

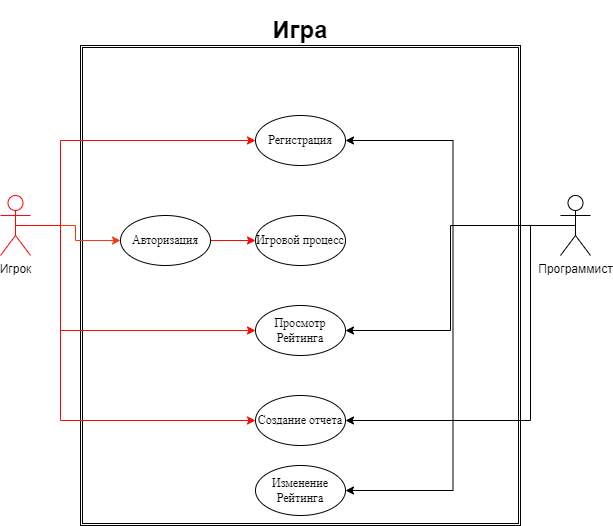


Рисунок 1 Диаграмма прецедентов

## **1.4 Постановка цели и задач**

Задача: Создание программного продукта «Компьютерная игра на движке Unity»

Цель проекта:

-исследовать предметную область;

-построить информационно-логическую модель с использованием СУБД;

-создать пользовательский интерфейс;

-разработать формы для внесения информации;

-разработать визуальный стиль игры

-разработать методы управления игровым персонажем

-протестировать игру

-разработать справочники, документы, отчеты и другие объекты конфигурации для данного проекта;

# **2 РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ**

## **2.1 Техническое обоснование проекта**

Игра предназначена для зарегистрированных пользователей любой организации, которые хотят нацелено поиграть или отдохнуть. Данная игра предоставляет несколько интересных механик и увлекательный игровой процесс, а также возможность просматривать свой счет и счет других игроков.

## **2.2 Формирование технического задания**

Система запускается под управлением ОС семейства Windows.

Системные требования, предъявляемые при установке Open Server:

Процессор: ядро с частотой в 1000 МГц и более.

Оперативная память: от 500 МБ оперативной памяти и более;

Жесткий диск: 4 ГБ свободного дискового пространства

Операционные системы Windows 7 или более поздние версий.

# **3 РАБОЧЕЕ ПРОЕКТИРОВАНИЕ**

Для разработки игры была выбрана платформа Unity. Данная платформа удобна в использовании и имеет бесплатную версию для персонального использования. Так же имеется большое количество обучающего материала от создателей платформы.

## **3.1 Файловая структура игры**

Разработанный проект содержит каталоги, в которых хранятся:

1) шаблоны объектов;

2) игровые сцены;

3) скрипты;

4) изображения персонажей, фонов;

5) анимация персонажа.

Файловая структура представлена ниже (См Рис. 2). В директории Prefabs находятся готовые шаблоны игровых объектов, при помощи которых можно составить игровой уровень после его проектирования. Директория Scenes содержит все сцены игры, в которые может перейти пользователь. В папке Scripts находятся скрипты с описанием всех классов и взаимодействий.

В каталоге Sprites содержатся три подкаталога:

- Buffs содержит в себе усиления.

- Player содержит в себе главного персонажа и то, что с ним связано.

- Stuff содержит в себе то, что используется для создания окружения

В каталоге Prefab содержатся три подкаталога:

- Buffs содержит в себе шаблон усиления.

- Player содержит в себе шаблон персонажа.

- Stuff содержит в себе шаблоны окружения.

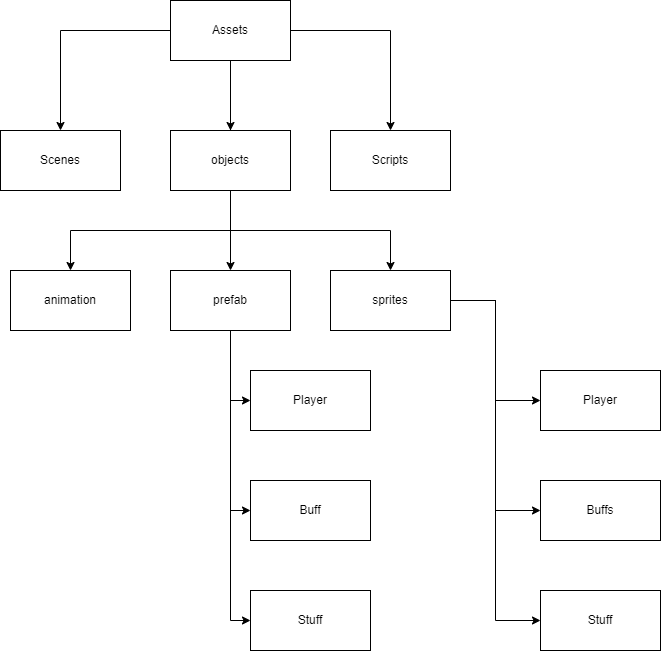


Рисунок 2 Файловая система

## **3.2 Создание пользовательского интерфейса**

При открытии игры игрок будет видеть окно авторизации, в которое нужно ввести логин или пароль (См. Рис 3). Если игрок не зарегистрирован, то снизу указана кнопка, которая переведет на сайт для регистрации аккаунта (См. Рис. 4).

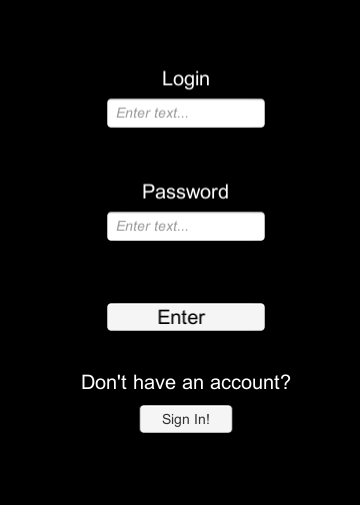


Рисунок 3 Окно авторизации

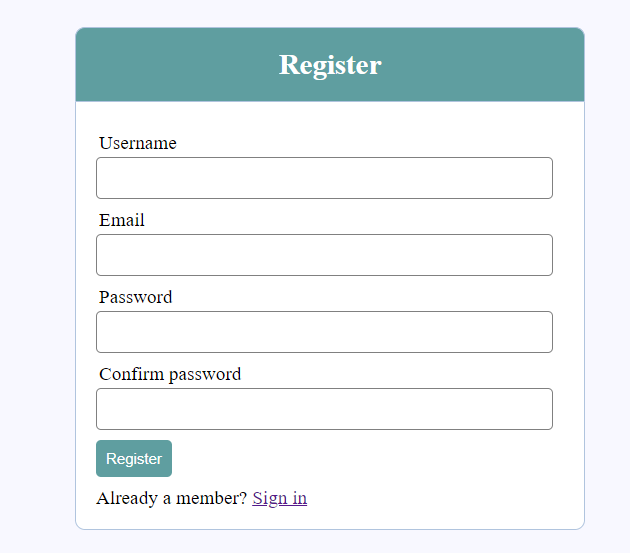


Рисунок 4 Регистрация на сайте

При введении данных от аккаунта, игрок будет перемещен на главную страницу игры (См. Рис. 5)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 Главное меню

При нажатии на кнопку Play игрок сможет начать играть в саму игру (См. Рис. 6).

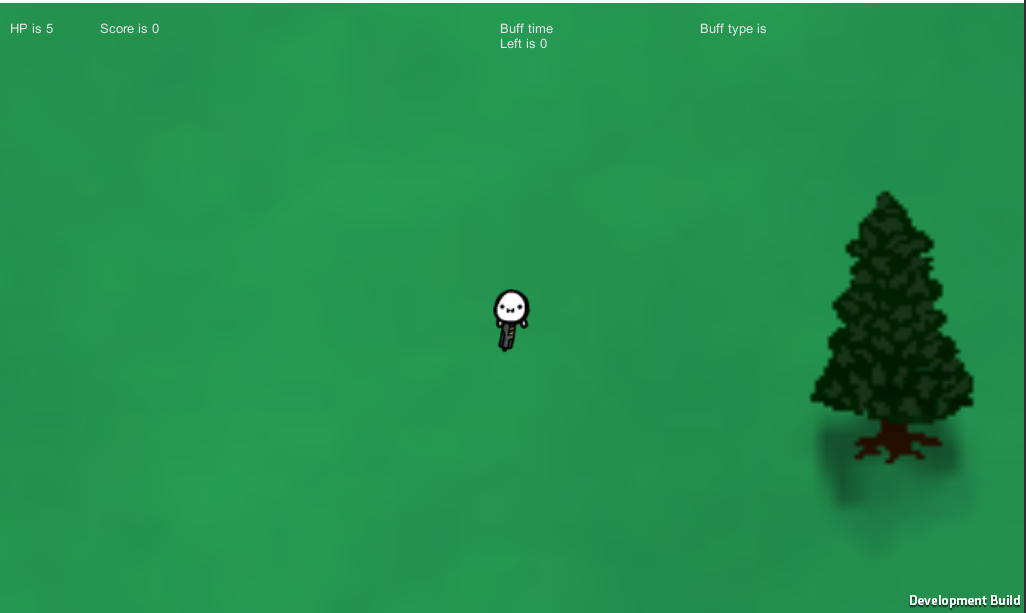


Рисунок 6 Игровой процесс

При нажатии на кнопку Settings игрок будет перемещен на страницу настроек игры (См. Рис. 7).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 Настройки игры

При получении смертельного урона игрок будет перенесен в меню проигрыша, где можно будет начать заново (См. Рис 7).



Рисунок 8 Меню проигрыша

## **3.3 Реализация программного продукта**

Основные компоненты проекта в Unity, использованные в программе:

Sprite Renderer – Один из важнейший компонентов, который используется в каждом объекте игры. Он нужен для того, что бы обрисовывать спрайты на объектах (См. Рис. 8).

Transform – Так же обязательный компонент в каждом объекте, благодаря ему можно определять положение объекта в игре (См. Рис. 9).

Rigidbody2d – Требуется для придании объекту физических качеств, таких как масса и сила тяжести (См. Рис. 10).

Collider – Требуется для отслеживания столкновений в игре, на нем строятся почти все механики (См. Рис. 11).

Script – Файл, в котором пишется код, который после прикрепляется к объекту. (См. Рис. 12).

Animator – позволяет привязывать к объекту анимацию (См. Рис 13).

Все эти компоненты изображены ниже.

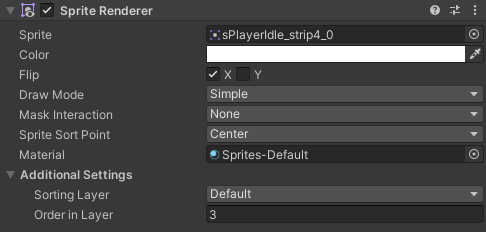


Рисунок 9 Компонент Sprite Renderer

Изображение выглядит как текст, устройство, счетчик, табло

Автоматически созданное описание

Рисунок 10 Компонент Transform

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 Компонент Rigidbody2d

Изображение выглядит как текст, дорога, электроника, снимок экрана

Автоматически созданное описание

Рисунок 12 Компонент Collider

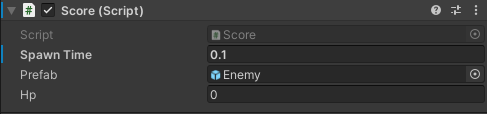


Рисунок 13 Компонент Script

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14 Компонент Animator

### **3.3.1 Реализация БД**

Для создания базы данных была выбрана программа PhpMyAdmin, которая позволяет управлять, редактировать, соединять различные таблицы в БД. Так же внутри есть удобный интерфейс и редактор SQL запросов. База данных игры находится на сервере что обеспечивает достойный уровень безопасности (См. Рис 14).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15 База данных

Игру можно запускать на нескольких компьютерах одновременно, и все они будут подключены к БД (См. Рис. 16)

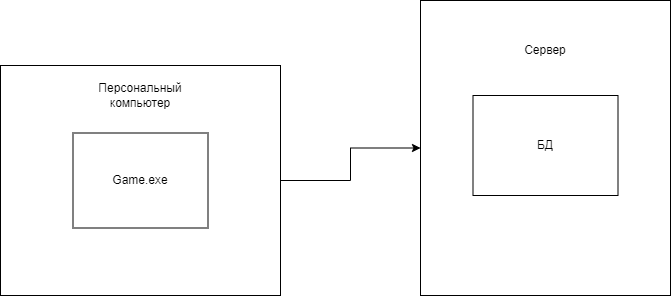


Рисунок 16 Диаграмма развертывания

Для работы с БД со стороны клиента используется метод WWW и WWWForms (См. Рис. 15) благодаря которому происходит подключение к PHP файлу на сервере (См. рис. 16)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 Отправка запроса и получение ответа

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 18 Обработка запроса со стороны сервера

### **3.3.2 Реализация Противника**

При появлении противник получает координаты игрока, после чего он будет изменять свое положение для сближения с игроком. Для того что бы найти игрока используется методы FindGameObjectWithTag() и GetComponent<>. Первый метод нужен для нахождения объекта с тегом Player, а второй для того, чтобы получить координаты объекта.

Для того что бы противник начал двигаться, используется метод Vector2.MoveTowards который двигает противника к цели. (См. Рис 17).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 19 Реализация противника

### **3.3.3 Реализация Персонажа**

При появлении персонажа на поле им можно управлять стрелками на клавиатуре или кнопками W, A, S, D. Для того что бы двигать персонажа был использован Rigidbody2d и Vector2. Для того что бы понять какую кнопку нажимает игрок используется метод Input.GetAxisRaw(). Так как игрок — это физический объект, для его движение используется rb.MovePosition() в котором происходят вычисления. (См. Рис 18)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20 Реализация движения

# **4 ТЕСТИРОВАНИЕ**

Для проверки работоспособности игрового приложения было проведено функциональное тестирование и юзабилити-тестирование.

## **4.1 Функциональное тестирование**

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователям.

Тесть № 1 Авторизация в игре.

Входные данные: Игрок вводит логин и пароль для авторизации.

Ожидаемый результат: Неудачная попытка входа, Успешная попытка входа.

Полученный результат: совпадает с ожидаемым (См. Рис. 19)

Тест успешно пройден.



Рисунок 21 Тест Авторизации

Тест № 2 Перемещение и стрельба

Входные данные: игрок управляет персонажем перемещает персонажа и стреляет в выбранном направлении.

Ожидаемый результат: Персонаж перемещается и стреляет в выбранном направлении.

Полученный результат: совпадает с ожидаемым (См. Рис. 20)

Тест успешно пройден.



Рисунок 22 Тест Перемещение и стрельба

Тест №3 Получение урона от противника

Входные данные: игрок сталкивается с одним или несколькими противниками.

Ожидаемый результат: Уровень здоровья персонажа снизился на количество задетых противников.

Полученный результат: совпадает с ожидаемым (См. Рис 21)



Рисунок 23 Тест Получение урона

Тест № 4 Попадание пулей по врагу.

Входные данные: Игрок стреляет в противника.

Ожидаемый результат: Выпущенный снаряд сталкивается с противном, убивая его, на счет игрока поступает 10 очков.

Полученный результат: совпадает с ожидаемым (См. Рис 22).



Рисунок 24 Тест Убийство Противника

Тест № 5 Получение усиления

Входные данные: Игрок подходит к усилению, тем самым подбирая его.

Ожидаемый результат: Игрок получил усиление в виде здоровья, скорости, щита.

Полученный результат: совпадает с ожидаемым (См. Рис. 23,24)

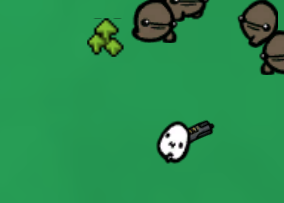


Рисунок 25 Тест Игрок нашел усиление

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 26 Тест игрок получил усиление

Тест № 6 Изменение количества противников

Входные данные: Игрок пытается изменить количество противников в настройках

Ожидаемый результат: Игрок вписывает значение и изменяет глобальную переменную

Полученный результат: совпадает с ожидаемым (См. Рис. 25)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 27 Тест Количество противников

## **4.2 Юзабилити тестирование.**

Юзабилити-тестирование (проверка эргономичности) – метод оценки удобства продукта в использовании, основанный на привлечении пользователей в качестве испытателей и суммировании полученных от них выводов. При прохождении данного теста были привлечены восемь пользователей. Тестировщикам было предложено решить следующие задачи:

-авторизироваться;

-начать игру;

-убить несколько противников;

-получить усиление;

-проверить свой результат после игры;

Все задачи были решены всеми тестировщиками без затруднений. Пользователи отметили интуитивность управления и правил игры. Юзабилити-тестирование прошло успешно.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта была разработана игра на движке Unity. Поставленные задачи выполнены и прошли тестирование. Была создана и настроена база данных, используемая в проекте.

# **СПИСОК ИСПЛЬЗОВАНННОЙ ЛИТЕРАТУРЫ**

1. Unity Manual 2020.3 URL: <https://docs.unity3d.com/Manual/index.html>
2. Original MySQL API URL: <https://www.php.net/manual/en/book.mysql.php>
3. MySQL Improved Extension URL: <https://www.php.net/manual/en/book.mysqli.php>
4. C# Documentation URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>

# **ПРИЛОЖЕНИЕ**

# **Программный код**

1. Листинг aiscript.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class aiscrip : MonoBehaviour

{

public float speed = 2f;

private Transform target;

void Start()

{

target = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>();

}

void Update()

{

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

}

}

2. Листинг Buffs.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Buffs : MonoBehaviour

{

buffSpeed \_buffSpeed;

movement \_movement;

Healthbar hp;

public GameObject buffPick;

GameObject ShieldBuff;

private float buffid = 0;

public static float buffDur;

private float speed = 0;

private string buffName;

public GameObject shield;

private void OnCollisionEnter2D(Collision2D collision)

{

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

{

\_movement = gameObject.GetComponent<movement>();

hp = gameObject.GetComponent<Healthbar>();

\_buffSpeed = collision.gameObject.GetComponent<buffSpeed>();

buffDur = \_buffSpeed.buffDuration;

buffid = \_buffSpeed.buffID;

speed = \_movement.movementSpeed;

OnBuff();

Destroy(collision.gameObject);

}

}

private void OnGUI()

{

GUI.Label(new Rect(500, 15, 75, 75), "Buff time Left is " + buffDur.ToString());

GUI.Label(new Rect(700, 15, 75, 75), "Buff type is " + buffName);

}

private void Update()

{

if(buffDur > 0)

{

buffDur -= Time.deltaTime;

Change(buffDur);

}

}

void OnBuff()

{

if (buffid == 0)

{

\_movement.movementSpeed += 2;

buffName = "Speed";

}

else if (buffid == 1)

{

hp.hp += 2;

buffName = "HEALTH";

}

else if (buffid == 2)

{

ShieldBuff = Instantiate(shield, gameObject.transform.position,Quaternion.identity);

buffName = "SUPER SHIELD";

}

}

void Change(float time)

{

if(time <= 0)

{

\_movement.movementSpeed = 4;

Destroy(ShieldBuff);

buffDur = 0;

}

}

}

3. Листинг BuffSpawn.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BuffSpawn : MonoBehaviour

{

public float respawnCd = 5f;

public GameObject prefab;

// Start is called before the first frame update

void Start()

{

InvokeRepeating("buffSpawn", respawnCd, respawnCd);

}

// Update is called once per frame

void Update()

{

}

void buffSpawn()

{

int posX = Random.Range(-20, 20);

int posY = Random.Range(-20, 20);

Vector3 spawnPos = new Vector3(transform.position.x + posX, transform.position.y + posY, 0);

Instantiate(prefab, spawnPos, Quaternion.identity);

}

}

4 Листинг buffSpeed.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class buffSpeed : MonoBehaviour

{

public float buffDuration = 10f;

public int buffID = 0;

public int buffChangingTime = 1;

private SpriteRenderer spriteRenderer;

public Sprite[] newSprite;

// Start is called before the first frame update

private void Awake()

{

}

void Start()

{

spriteRenderer = gameObject.GetComponent<SpriteRenderer>();

InvokeRepeating("ChangeSprite",buffChangingTime, buffChangingTime);

}

// Update is called once per frame

void Update()

{

}

void ChangeSprite()

{

buffID = Random.Range(0, 3);

spriteRenderer.size += new Vector2(5, 5);

spriteRenderer.sprite = newSprite[buffID];

}

}

5 Листинг bullet.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class bullet : MonoBehaviour

{

public GameObject prefab;

private void OnCollisionEnter2D(Collision2D collision)

{

if (collision.transform.gameObject.tag == "Enemy")

{

// GameObject effect = Instantiate(hitEffect, collision.transform.position, collision.transform.rotation);

Destroy(collision.gameObject);

//Destroy(effect, 2f);

Score.score += 10;

Score.enemyCount--;

Instantiate(prefab, collision.transform.position,Quaternion.identity);

}

Destroy(gameObject);

}

}

6 Листинг CameraFollow.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraFollow : MonoBehaviour

{

public Transform player;

private Vector3 offset;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

transform.position = new Vector3(player.position.x + offset.x, player.position.y + offset.y,-40);

}

}

7 Листинг dealingDamage.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class dealingDamage : MonoBehaviour

{

public GameObject prefab;

Healthbar hp;

private void OnCollisionEnter2D(Collision2D collision)

{

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

{

hp = GameObject.FindGameObjectWithTag("Player").GetComponent<Healthbar>();

hp.hp--;

Instantiate(prefab, transform.position, Quaternion.identity);

Destroy(gameObject);

Score.enemyCount--;

}

}

}

8 Листинг ShieldBuff.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ShieldBuff : MonoBehaviour

{

public GameObject prefab;

private void Update()

{

if (Buffs.buffDur == 0)

{

Destroy(gameObject);

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if (collision.transform.gameObject.tag == "Enemy")

{

// GameObject effect = Instantiate(hitEffect, collision.transform.position, collision.transform.rotation);

Destroy(collision.gameObject);

//Destroy(effect, 2f);

Score.score += 10;

Score.enemyCount--;

GameObject corpse = Instantiate(prefab, collision.transform.position, Quaternion.identity);

Destroy(corpse, 10f);

}

}

}

9 Листинг shooting.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class shooting : MonoBehaviour

{

public Transform firePoint;

public GameObject prefab;

public float bulletForce = 20f;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

if (Input.GetButtonDown("Fire1"))

{

Shoot();

}

}

void Shoot()

{

GameObject bullet = Instantiate(prefab, firePoint.position, firePoint.rotation);

Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();

rb.AddForce(firePoint.up \* bulletForce,ForceMode2D.Impulse);

}

}

10 Листинг movement.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class movement : MonoBehaviour

{

public Rigidbody2D rb;

public Camera cam;

public Animator animator;

public int movementSpeed = 4;

Vector2 mvmnt;

public float animspeed;

public bool facingRight = true;

private SpriteRenderer sprite;

private void Start()

{

sprite = gameObject.GetComponent<SpriteRenderer>();

}

void Update()

{

mvmnt.x = Input.GetAxisRaw("Horizontal");

mvmnt.y = Input.GetAxisRaw("Vertical");

animspeed = Mathf.Abs(mvmnt.x) + Mathf.Abs(mvmnt.y);

animator.SetFloat("speed", Mathf.Abs(animspeed\*movementSpeed));

if (mvmnt.x > 0)

{

sprite.flipX = false;

}

else if (mvmnt.x < 0 )

{

sprite.flipX = true;

}

}

private void FixedUpdate()

{

rb.MovePosition(rb.position + mvmnt \* movementSpeed \* Time.fixedDeltaTime);

}

}

11 Листинг ShieldConnect.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ShieldConnect : MonoBehaviour

{

private Transform target;

// Start is called before the first frame update

void Start()

{

target = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>();

}

// Update is called once per frame

void Update()

{

transform.position = target.position;

}

}

12 Листинг Web.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.Networking;

using UnityEngine.SceneManagement;

using System;

public class Web : MonoBehaviour

{

public static string login, password;

private string warning;

public static int DBscore;

public void PasswordText(string pass)

{

password = pass;

}

public void LoginText(string log)

{

login = log;

}

public IEnumerator Test()

{ //Создаем Запрос

WWWForm form = new WWWForm();

form.AddField("Login", login);

form.AddField("Password", password);

//Отправляем запрос

WWW req = new WWW("siteTest.loc", form);

//Поулчаем Ответ

yield return req;

Debug.Log(req.text);

string[] mas = req.text.Split(' ');

//Score DB

try

{

DBscore = int.Parse(mas[1]);

}

catch(Exception )

{

}

if (mas[0] == "True")

{

SceneManager.LoadScene(1);

}

else

{

warning = "Неправильный логин или пароль!";

}

}

public void ButtonPress()

{

StartCoroutine(Test());

}

private void OnGUI()

{

GUI.Label(new Rect(100, 15, 200, 200), warning);

}

public void OpenURL()

{

Application.OpenURL("http://site.loc/login.php");

}

}

13 Листинг Death.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Death : MonoBehaviour

{

public GameObject particle;

void Start()

{

GameObject particleInstance = Instantiate(particle, transform.position, Quaternion.identity);

Destroy(particleInstance, 1f);

Destroy(gameObject, 10f);

}

}

14 Листинг rotation.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class rotation : MonoBehaviour

{

//Weapon Rotation Script

public float speed = 50f;

public bool facingRight = false;

public Transform player;

void Start()

{

}

// Update is called once per frame

void Update()

{

Vector2 dir = Camera.main.ScreenToWorldPoint(Input.mousePosition) - transform.position;

float angle = Mathf.Atan2(dir.y, dir.x) \* Mathf.Rad2Deg;

Quaternion rotation = Quaternion.AngleAxis(angle, Vector3.forward);

transform.rotation = Quaternion.Slerp(transform.rotation, rotation, speed \* Time.deltaTime);

transform.position = player.position;

}

private void FixedUpdate()

{

}

}

15 Листинг Healthbar.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Healthbar : MonoBehaviour

{

public int hp = 5;

private void Update()

{

if (hp <= 0)

{

SceneManager.LoadScene(3);

}

}

private void OnGUI()

{

GUI.Label(new Rect(10, 15, 75, 75), "HP is " + hp.ToString());

}

}

16 Листинг Score.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Score : MonoBehaviour

{

public static int score = 0;

public static int enemyCount = 0;

public float spawnTime = 1000f;

public GameObject prefab;

public int hp;

GameObject[] posSpawn = new GameObject[6];

void Start()

{

InvokeRepeating("SpawnEnemy", spawnTime, spawnTime);

posSpawn = GameObject.FindGameObjectsWithTag("Point");

}

void Update()

{

}

private void OnGUI()

{

GUI.Label(new Rect(100, 15, 75, 75), "Score is " + score.ToString());

}

void SpawnEnemy()

{

if (MaxEnemies.enemyMax > enemyCount)

{

Instantiate(prefab, posSpawn[Random.Range(0, 6)].GetComponent<Transform>());

enemyCount++;

}

}

}

17 Листинг detatching.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class detatching : MonoBehaviour

{

// Update is called once per frame

void Update()

{

transform.DetachChildren();

}

}

18 Листинг Link.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Link : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

}

public void UrlOpen()

{

Application.OpenURL("http://site.loc/login.php");

}

}

19 Листинг MainMenu.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour

{

// Start is called before the first frame update

public void Play(int index)

{

SceneManager.LoadScene(index);

}

}

20 Листинг following.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class following : MonoBehaviour

{

public Transform player;

private Vector3 offset;

// Update is called once per frame

void Update()

{

transform.position = new Vector3(player.position.x + offset.x, player.position.y + offset.y, -40);

}

}

21 Листинг ParticleScript.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ParticleScript : MonoBehaviour

{

void Update()

{

Destroy(gameObject, 3f);

}

}

22 Листинг ScoreGameOver.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class ScoreGameOver : MonoBehaviour

{

Text endScore;

void Start()

{

endScore = GetComponent<Text>();

endScore.text +=" "+ Score.score.ToString(); ;

StartCoroutine(ScoreSend());

}

public IEnumerator ScoreSend()

{

if(Web.DBscore >= Score.score)

{

}

else

{

WWWForm form = new WWWForm();

form.AddField("Score", Score.score);

form.AddField("Login", Web.login);

WWW req = new WWW("siteTest.loc", form);

yield return req;

}

Score.score = 0;

}

}

23 Листинг Textt.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Textt : MonoBehaviour

{

Text text;

// Start is called before the first frame update

void Start()

{

text = GetComponent<Text>();

text.text += ", "+Web.login;

}

}

24 MaxEnemiews.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class MaxEnemies : MonoBehaviour

{

public static int enemyMax = 100;

public Text text;

public Text inputText;

private void Start()

{

inputText.text = enemyMax.ToString();

}

private void Update()

{

text.text = "Максимальное количество противников = "+ enemyMax.ToString();

}

public void EnemyChange(string enemies)

{

try

{

enemyMax = System.Int32.Parse(enemies);

}

catch (System.Exception)

{

}

}

public void ButtonPressed(int x)

{

SceneManager.LoadScene(x);

}

}