МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Вычислительная техника»

Курсовой проект
по дисциплине «Системное программирование»
на тему «Мобильная игра на Android»

Специальность 09.05.01 «Применение и эксплуатация автоматизированныхсистем специального назначения»

Специализация № 1: Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения

Выполнил: студент гр. 18ВО2
Лучинкин Г.Ю.
Руководитель д.т.н, профессор каф.ВТ
Дубинин В.Н.

Работа защищена с оценкой отл.
Преподаватели

Дата защиты 8.06.2021

2021

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра “Вычислительная техника”

“УТВЕРЖДАЮ”

Зав. кафедрой ВТ

_____ М.А. Митрохин

«___»_____2021 г.

ЗАДАНИЕ

на курсовое проектирование по курсу

_____Системное программирование_____

Студенту Лучинкину Глебу Юрьевичу_____Группа_18ВО2_ _ _ ___

Тема_проекта: ____Мобильная игра на Android____

Исходные данные (технические требования) на проектирование

Разработать и программно реализовать 2D игру для мобильного устройства с операционной системой Android. Сюжет игры связан с преодолением персонажем различных препятствий.

Требования к игре:

-- создать управления персонажем;

-- предусмотреть два уровня и реализовать переход с первого на второй;

-- предусмотреть «мертвые зоны»;

-- разработать управление, ориентированное под мобильное устройство.

Операционная среда: Android

Язык программирования: C#.

Среда разработки: Unity

Объем работы по курсу

1. Расчетная часть

2. Экспериментальная часть

Тестирование и отладка программы на устройстве SmartPhone

Срок выполнения проекта по разделам

1 В соответствии с графиком выполнения курсового проекта

2 _____

3 _____

4 _____

5 _____

6 _____

7 _____

8 _____

Дата выдачи задания “16” февраля 2021 г.

Дата защиты проекта “____”_____

Руководитель_____ Дубинин В.Н.

Задание получил “_____”_____2021 г.

Студент_____/ Лучинкин Г.Ю._____

Реферат

Пояснительная записка состоит из 41 листов, 37 рисунков, 1 таблицы.

Цель работы – создание мобильной игры для устройств с операционной системой Android.

Результатом проделанной работы была игра, запущенная на мобильном устройстве.

Содержание

Введение 6
1 Расчетная часть 7
1.1 Основание для разработки 7
1.2 Требование к программе 7
1.2.1 Требование к функциональным характеристикам 7
1.2.2 Требование к параметрам технических средств 7
1.3 Требование к программной документации 8
1.4 Стадии и этапы разработки 8
1.5 Порядок контроля и приемки 9
2 Разработка игры 9
2.1 Создание окружения 9
2.2 Управление персонажем 12
2.3 Зоны смерти 17
2.4 Зоны Победы 19
2.5 Главное меню 21
3 Запуск игры на мобильном устройстве 23
3.1 Сборка проекта под операционную систему Android 23
3.2 Запуск игры на телефоне 27
Заключение 31
Список используемой литературы 31

Введение

Ни для кого не секрет, что видео игры прочно заняли свою позицию в современной индустрии развлечений. Существуют попытки выделить компьютерные игры как отдельную область искусства, наряду с театром, кино и т.п. Разработка игр может оказаться не только увлекательным, но и прибыльным делом, примеров этому предостаточно в истории. Первые примитивные компьютерные и видео игры были разработаны в 1950-х и 1960-х годах. Они работали на таких платформах, как осциллографы, университетские мейнфреймы и компьютеры EDSAC. Самой первой компьютерной игрой стал симулятор ракеты, созданный в 1942 году Томасом Голдсмитом Младшим (англ. Thomas T. Goldsmith Jr.) и Истл Рей Менном (англ. Estle Ray Mann). Позже, в 1952 году, появилась программа "OXO", имитирующая игру "крестики-нолики", созданная А.С. Дугласом как часть его докторской диссертации в Кембриджском Университете. Игра работала на большом университетском компьютере, известном как EDSAC (Electronic Delay Storage Automatic Calculator). В настоящее время, разработка игры - это многомиллионный процесс, в котором задействована целая команда разработчиков, сложные современные технологии и даже маркетинговые ходы/1/

Сегодня же наиболее востребованными видеоиграми считаются те, которые находятся у нас в кармане, внутри нашего мобильного устройства. В эти игры можно играть где угодно, когда угодно и с кем угодно, ведь у каждого сейчас есть телефон. Наиболее распространенные операционные системы это IOS и Android. Каждый день на эти две платформы выходят тысячи игр/2/.

Целью моей курсовой работы было создание 2D платформера на базе ОС Android, в среде разработки Unity.

1. Расчетная часть

1.1Основание для разработки

Мобильная игра разработана на основании задания на курсовое проектирование по дисциплине «Системное программирование» - утвержденного заведующий кафедры ВТ, Митрохиным М.А. и выданным руководителем курсовой работы Дубининым В.Н.

1.2 Требование к программе

1.2.1 Требование к функциональным характеристикам

Мобильная игра должна содержать следующие параметры:

1) Окружение, по которому будет ходить персонаж;

2) Управление персонажем разработанное под мобильное устройство;

3) Зоны «смерти», сталкиваясь с которыми персонаж будет возвращаться на точку «возрождения»;

4) Зоны «победы», при преодолении которой у игрока появляется возможность перейти на следующий уровень.

5) Создание главного меню

1.2.2 Требование к параметрам технических средств

Мобильная игра должна запускаться на устройствах с версией операционной системы Android 4.4 и выше.

1.3 Требование к программной документации

Программная документация к разрабатываемой игре должна содержать следующие разделы:

-Расчетная часть

-Графическая часть

-Экспериментальная часть

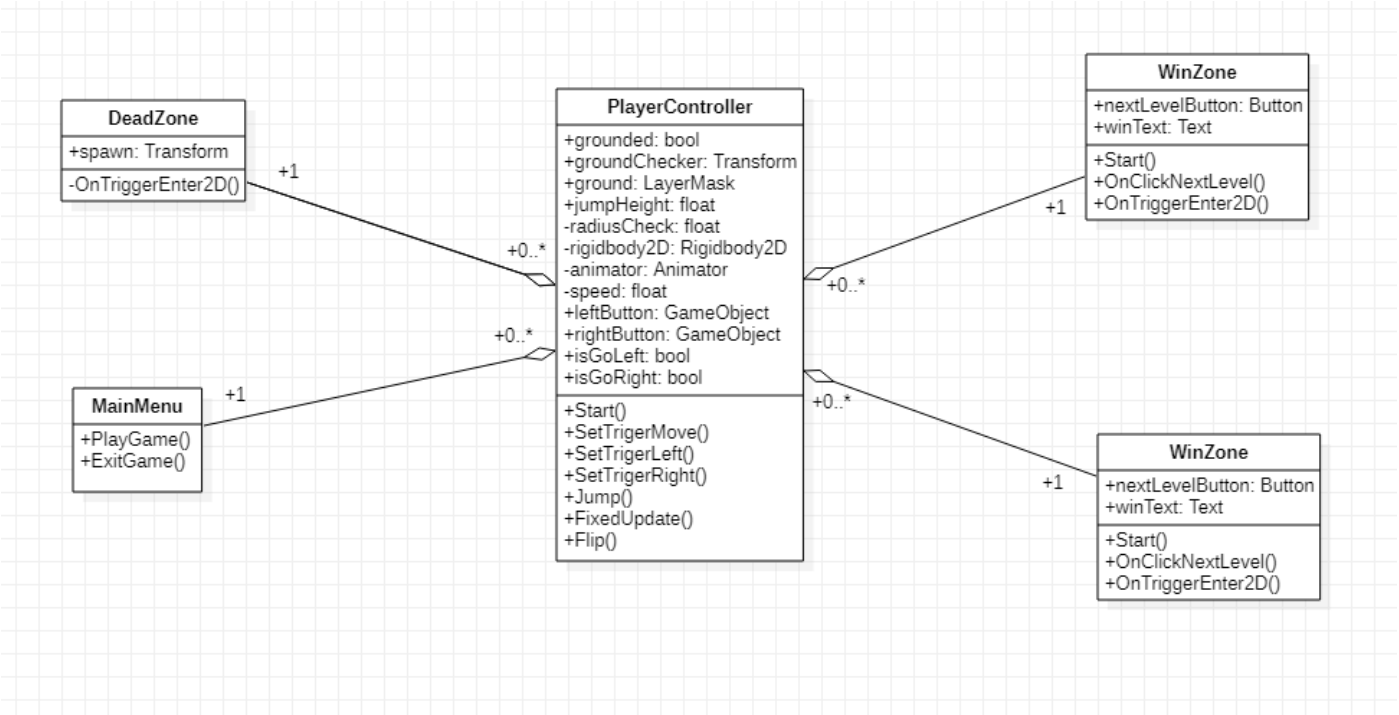
-Заключение

1.4 Стадии и этапы разработки

Стадии и этапы разработки приведены в таблице 1.

Таблица 1 - Стадии и этапы разработки модели

№	Стадии	Этапы разработки	Сроки выполнения	Исполнитель
1	Расчетная часть	Создание окружения	20.03.2021	Лучинкин Г.Ю.
		Создание управления	25.03.2021	Лучинкин Г.Ю.
		Создание зон «смерти»	30.03.2021	Лучинкин Г.Ю.
		Создание зон победы	7.04.2021	Лучинкин Г.Ю.
		Создание меню главного экрана	20.04.2021	Лучинкин Г.Ю.
		Написание пояснительной записки	31.06.2021	Лучинкин Г.Ю.
2	Графическая часть	UML диаграмма	20.05.2021	Лучинкин Г.Ю.
3	Экспериментальная часть	Запуск игры на мобильном устройстве	25.05.2021	Лучинкин Г.Ю.



					ПГУ 1.090501.08		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Лучинкин Г.Ю.		1.06.21	UML ДИАГРАММА КЛАССОВ	Лит.	Лист 9 / Листов 41
Провер.		Дубинин В.Н.					
						Гр. 18ВО2	
Н. Контр.							
Утверд.							

1.5 Порядок контроля и приёмки

Для проверки правильности работы игры необходимо запустить ее на мобильном устройстве, и проверить работы всех функции. После этого необходимо пройти ее до конца, после чего выйти в главное меню.

Если все функции работают, и из можно выйти через главное меню, то это значит, что игра работает правильно.

2 Разработка игры.

2.1 Создание окружения.

Для создания окружения персонажа, по которому он смог бы ходить, необходимо воспользоваться средствами, которые предлагает игровой движок Unity.

В начале нужно выбрать из чего будет состоять окружение. Были использованы текстуры из открытых источников, которые были загружены в проект.

Все текстуры, которые нам понадобятся, находятся в папке Sprites (Рисунок 1)

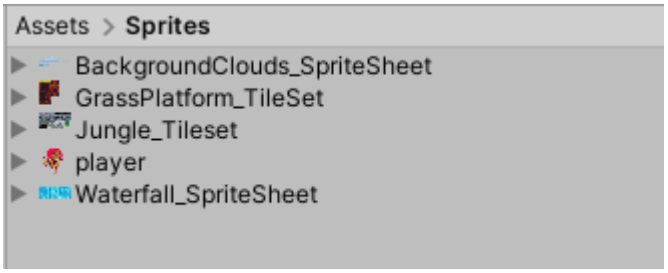


Рисунок 1 - текстуры

Необходимо задать правила для построения текстур.

В Unity можно создать правила, по которым будут строится текстуры. Для этого нам нужно создать функцию TileRule, и вставить в него соответсвующие «спрайты», по которым будет создаваться текстура (Рисунок 2). Было загружено 9 спрайтов, которые составляют из себя одну текстуру. При рисовании мы будем получать полноценную зону, без пропусков, и багов в текстурах (Рисунок 3). В

дальнейшем именно из них и будет создана основная часть карты.

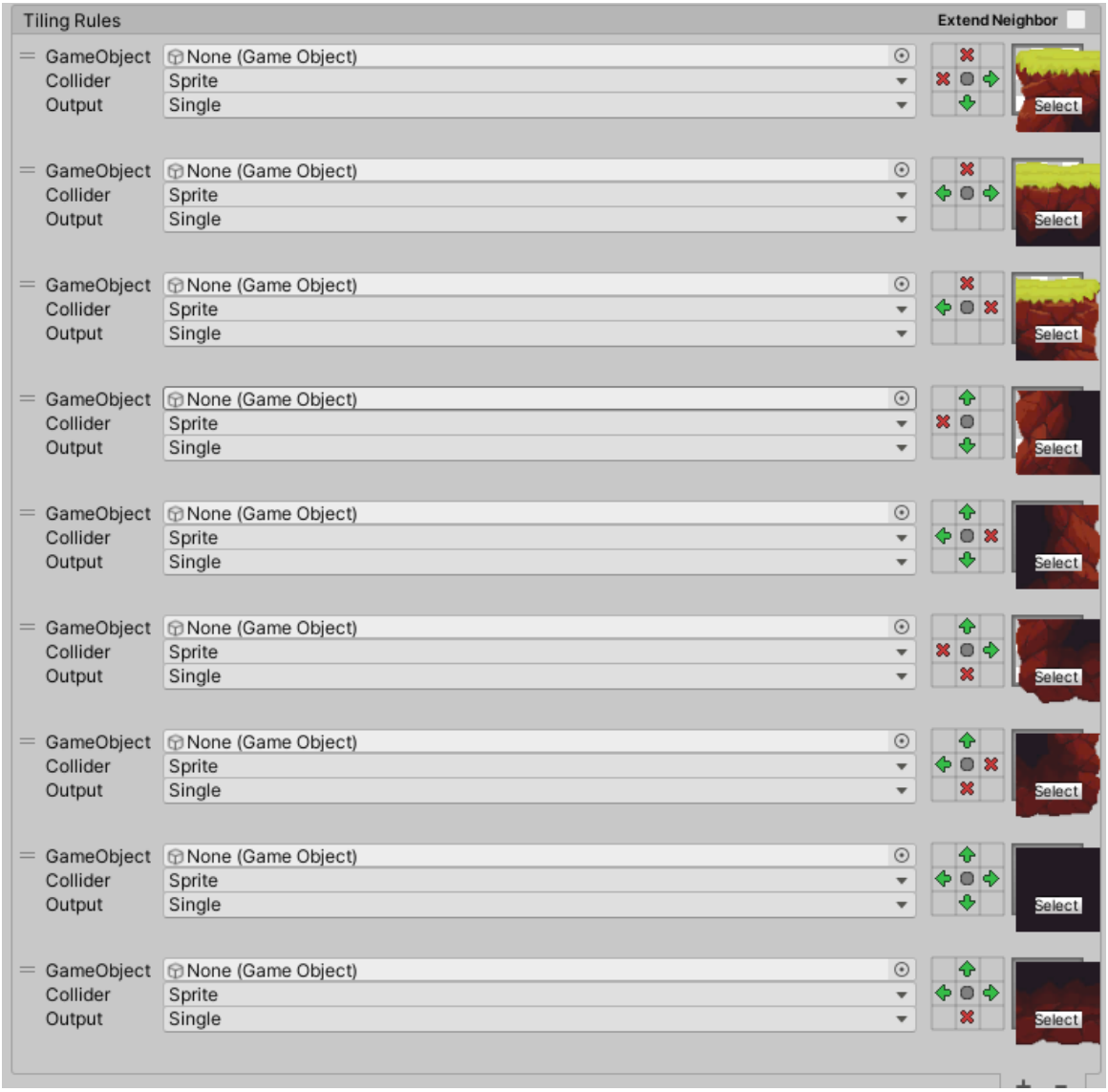


Рисунок 2 - TileRule

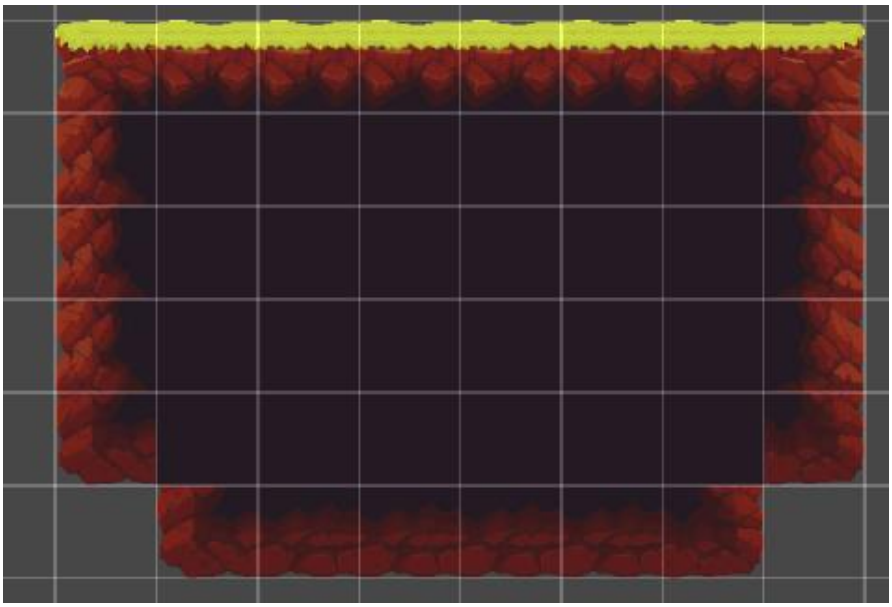


Рисунок 3 - Готовая текстура

Таким же образом можно создавать анимированые текстуры. Необходимо создать новый TileRule, и загрузить в него спрайты водопада (Рисунок 4). Только в этот раз мы выберем только 2 спрайта, начальную и конечную точку водопада.

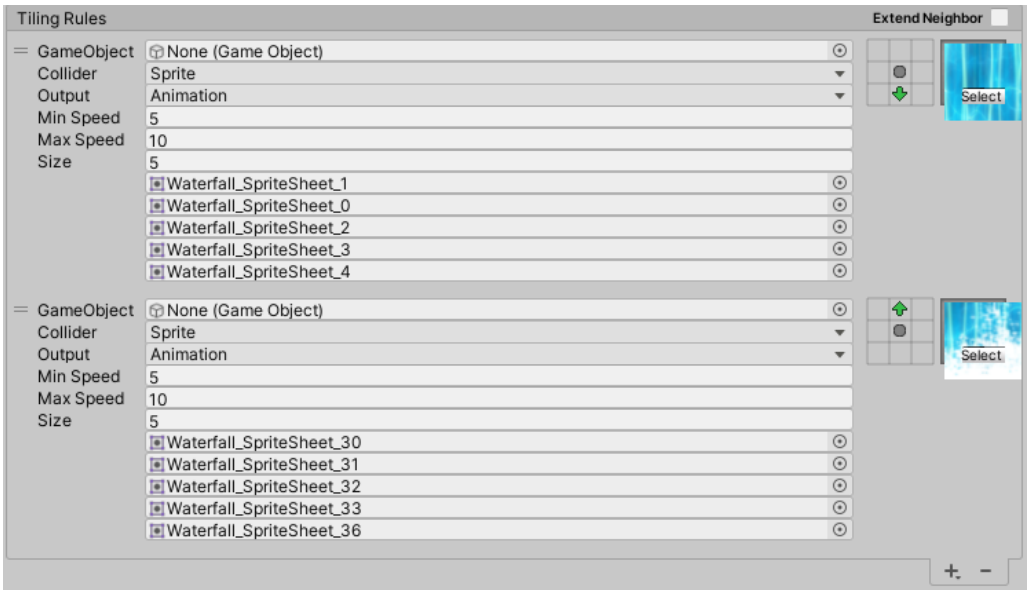


Рисунок 4 - TileRule водопада

На рисунке 4 видно, что были загружены промежуточные спрайты, и было выставлено время, по которому текстуры будут сменять друг друга. В результате создания TileRule’ов мы можем создать такую комбинацию текстур (Рисунок 5)

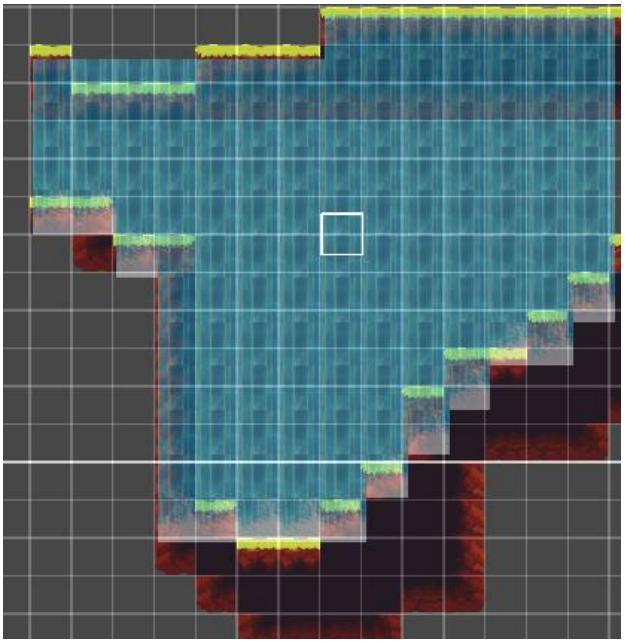


Рисунок 5 - готовая текстура

Таким образом, комбинируя два вида текстур были созданы два полноценных уровня (Рисунки 6-7).

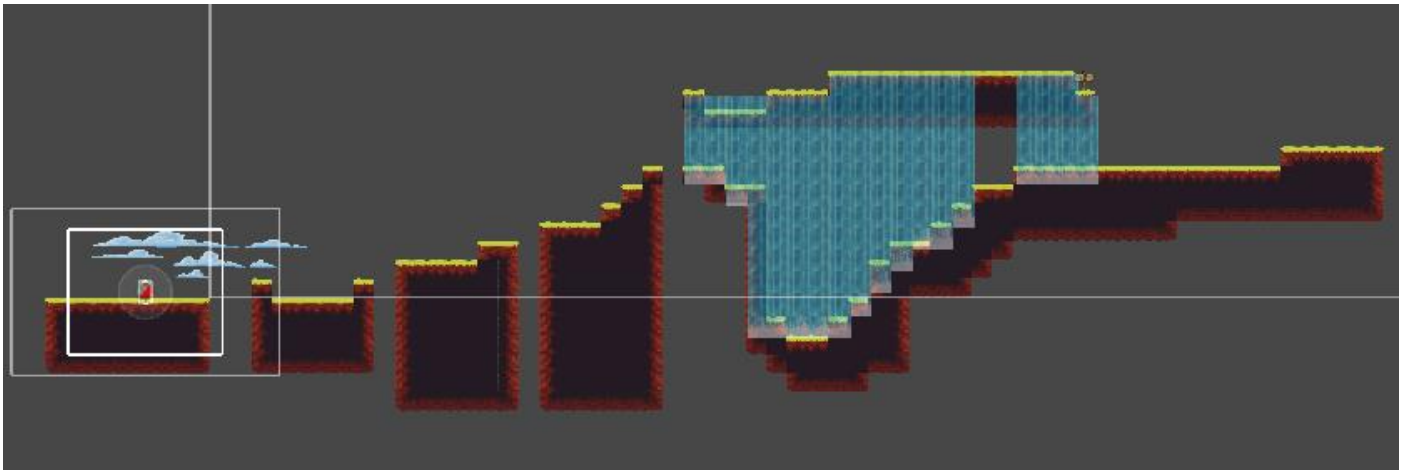


Рисунок 6 - первый уровень

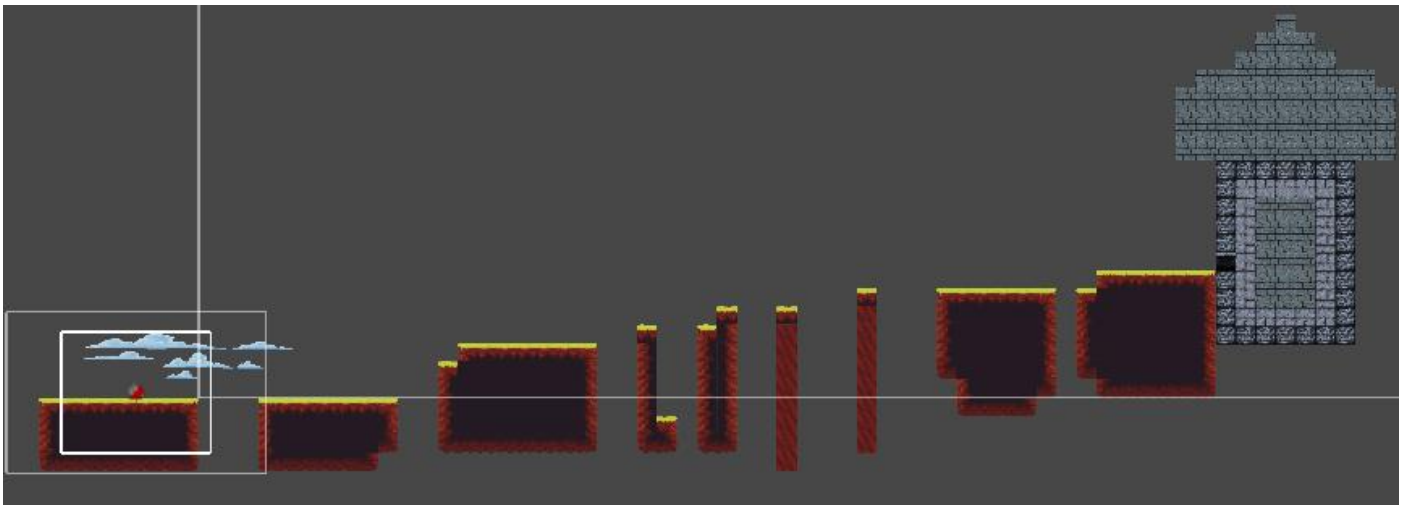


Рисунок 7 - второй уровень

2.2 Управление персонажем

Важной частью любой игры является управление. В этой игре главным является управление персонажа, потому что без него мы не смогли бы дальше продвигаться по уровню.

Для начала нужно выбрать как будет выглядеть наш персонаж. На рисунке 1, в списках текстур есть персонаж, нам нужно поместить его на рабочую область (Рисунок 8).

Рисунок 8 - Персонаж

В данный момент персонаж это просто, картинка, которая не может двигаться, и просто падает с экрана. Для того, чтобы этого не было необходимо добавить свойство RigidBody 2d (Рисунок 9), с помощью него текстура сможет физически взаимодействовать с окружением.

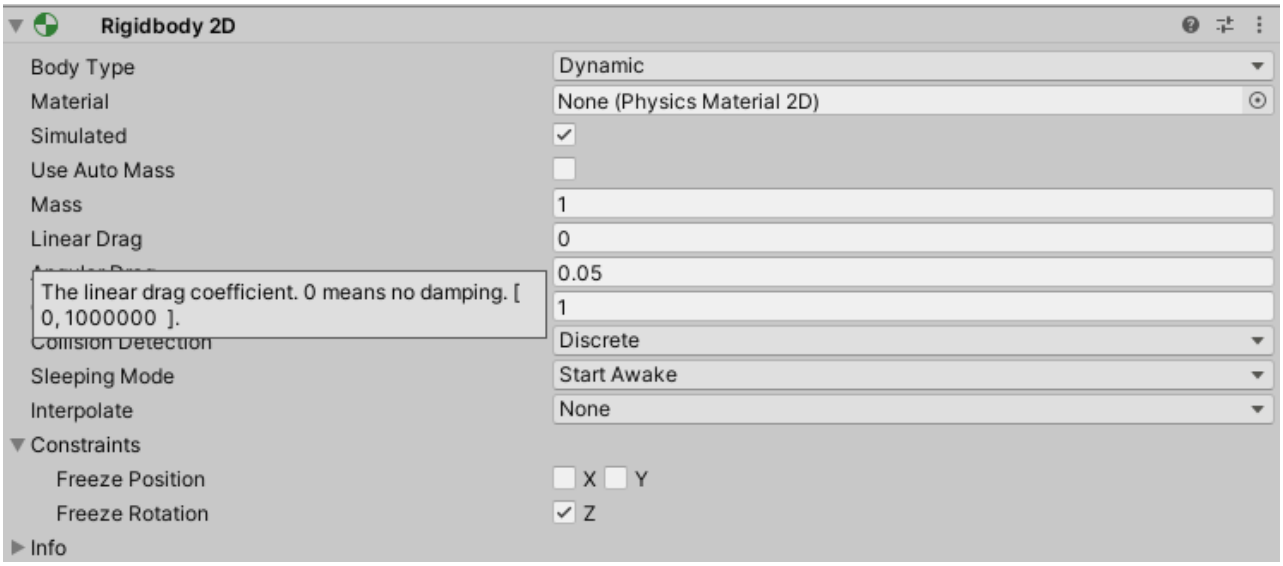


Рисунок 9 - Добавление RigidBody 2d

Далее необходимо создать C# скрипт, и поместить его в свойства персонажа. Был создан C# скрипт под название PlayerController, и был закреплен в свойствах у персонажа (Рисунок 10).

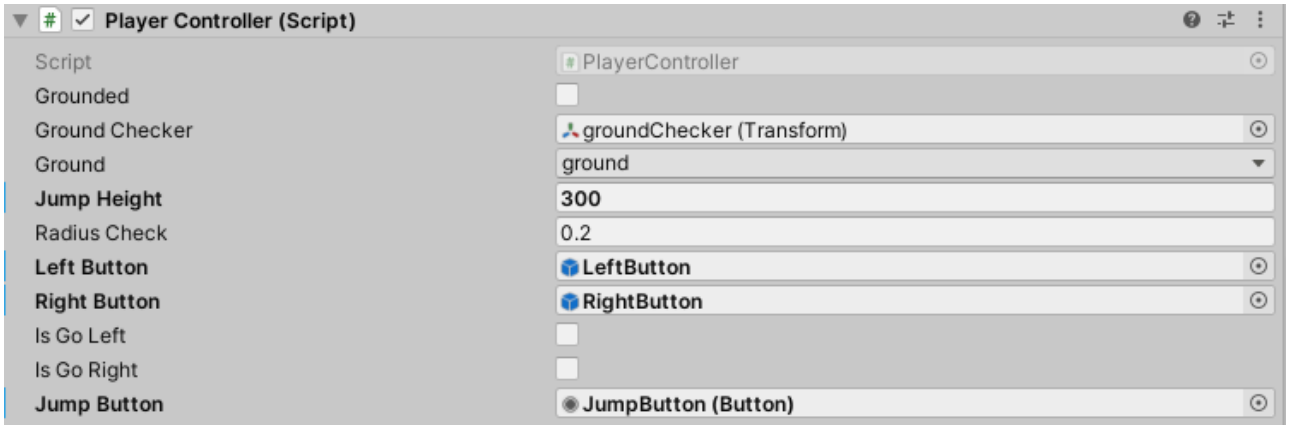


Рисунок 10 - PlayerController

Сначала дадим возможность персонажу поворачиваться в разные стороны. На языке C# был создан метод с названием flip, который определяет, если координаты движения персонажа меняется, то нужно его развернуть (Рисунок 11 - 12). Поворот персонажа:

Рисунок 11

Рисунок 12

Далее необходимо дать персонажу возможность ходить. Для этого были созданы две публичные переменные isGoleft и isGoRight типа bool. Так же была создана переменная move типа float, которая будет определять по какой координате персонаж двигается, и уже тогда менять его положение экране. При движение координата по оси x будет меняться (Рисунок 13-12).

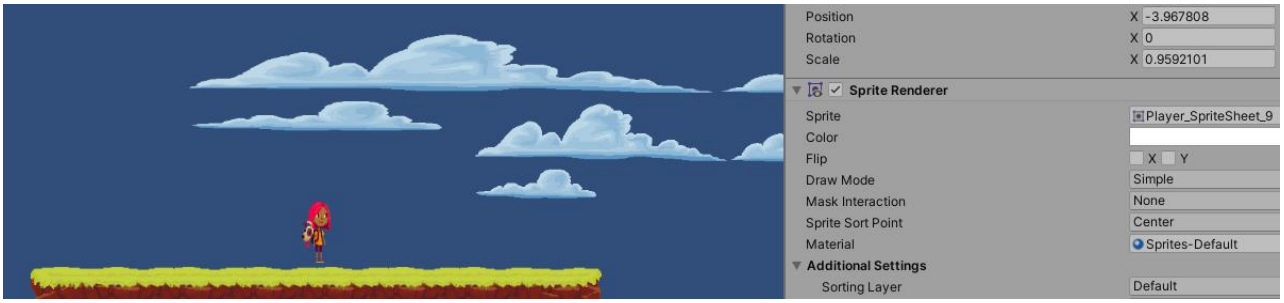


Рисунок 13 - Движение вправо

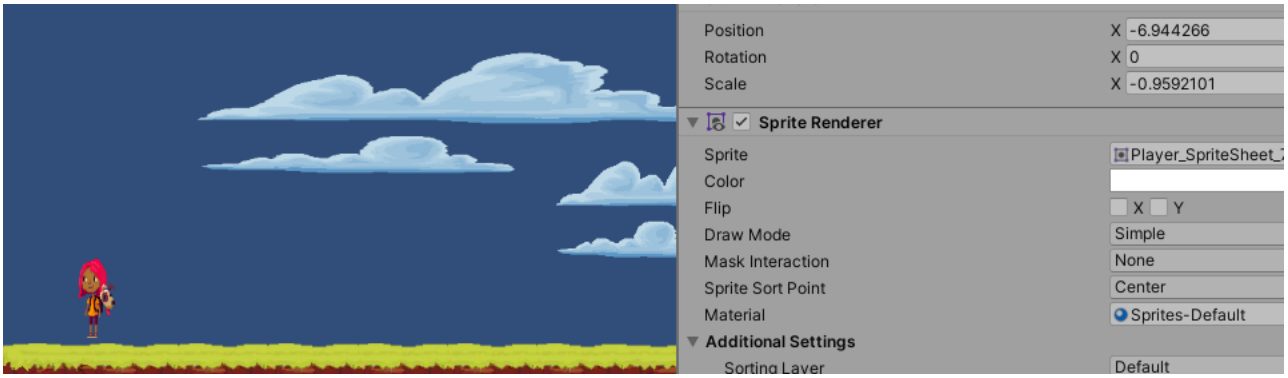


Рисунок 14 - Движение влево

Далее необходимо дать возможность персонажу прыгать. Для этого потребуется создать пустой объект с название groundcheker. Так же нужно поместить его в свойства к персонажу (Рисунок 15).

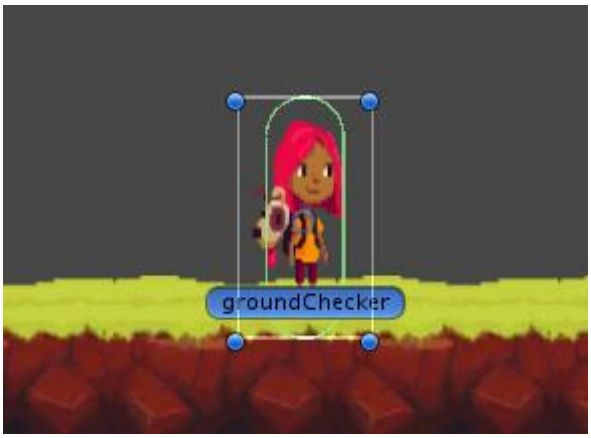


Рисунок 15 - groundcheker

С помощью groundcheker персонаж будет определять, что он находится на земле, и тогда можно будет совершить прыжок. Для этого был создан метод jump на языке C#, Который определяет, если персонаж на земле, то изменить координату по оси Y (Рисун ок 16).

Рисунок 16 - Прыжок

На данный момент было создано управление, которое заточено только под управление с клавиатуры компьютера. Для того, чтобы в игру можно было играть на мобильном устройстве, необходимо создать кнопки, которые будут размещены на экране.

В меню GameObject->UI есть разновидности кнопок, нам нужно поместить на форму 3 кнопки, Jump, left , right (Рисунок 17).

Рисунок 17 - Кнопки для экрана телефона

Так же, в C# скрипте были созданы метода settriggerleft и settriggerright. В них описывается простой алгоритм, если кнопка left или right нажата, то персонажа двигается в указанном направлении.

Весь код можно посмотреть в приложении А.

2.3 Зоны смерти

Для создания зон «смерти» необходимо создать пустой объект, и назвать его DeadZone (Рисунок 18). После создания необходимо добавить свойство BoxCollider. Чтобы пустой объект мог принимать форму прямоугольника (Рисунок 19).

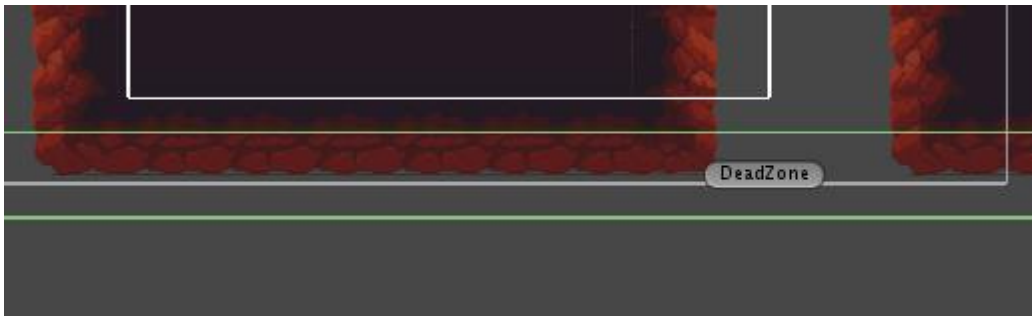


Рисунок 18 - Зона смерти

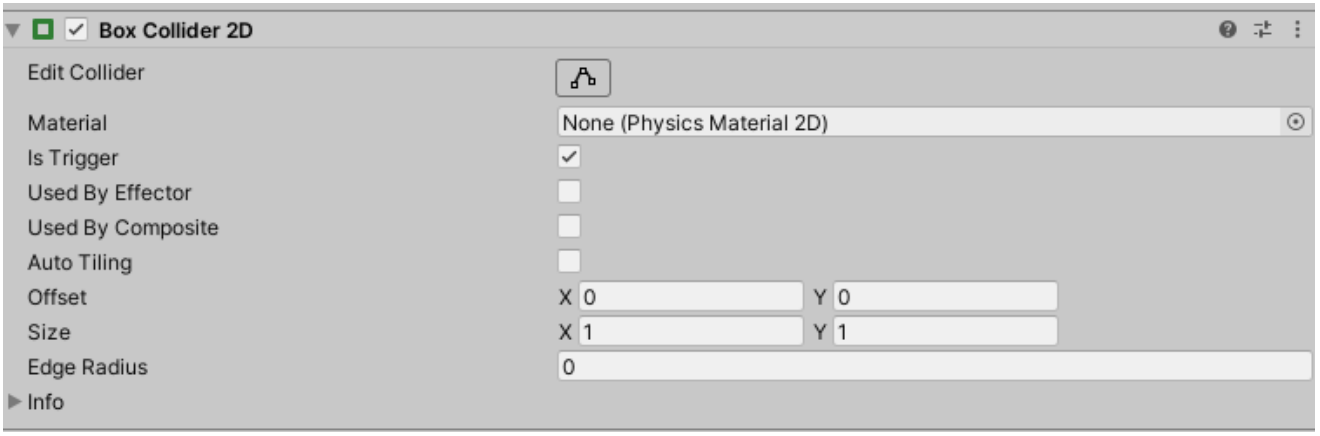


Рисунок 19 - BoxCollider

Затем необходимо создать пустой объект, на который будет возвращаться персонаж, после попадания в зону «смерти». Для этого в меню CreateObject нужно выбрать Create empty, и назвать его spawn (рисунок 20).

Рисунок 20 - Создание точки возрождения

Необходимо создать скрип на языке C#, в котором описано поведение персонажа после соприкосновения с зоной «смерти». Была создана публичная переменная spawn типа transform. Так же был создан метод OnTriggerEnter2d, в котором описывается перемещение персонажа после попадания в зону смерти (Рисунок 21-22)

Рисунок 21 - Попадание в зону смерти

Рисунок 22 - Возрождение

Весь код можно посмотреть в приложении А.

2.4 Зоны победы

Для создания зоны победы необходимо создать пустой объект, при попадании в который на экране появятся кнопка и надпись, предлагающая перейти на следующий уровень. Для этого в меню CreateObject нужно выбрать Create empty, и назвать его WinZone (Рисунок 23). Затем необходимо добавить на WinZone свойство BoxCollider, чтобы этот объект мог принять форму прямоугольника (Рисунок 24).

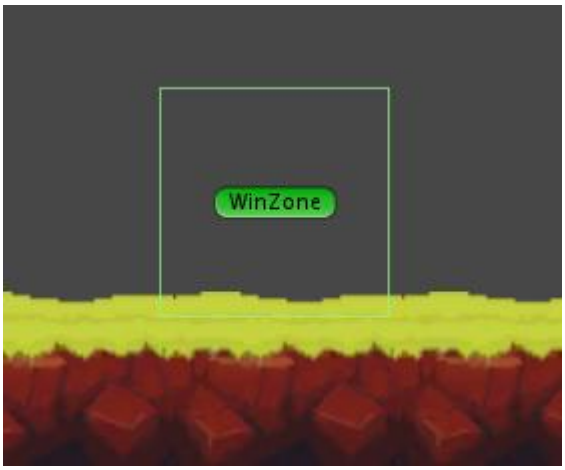


Рисунок 23- Зона победы

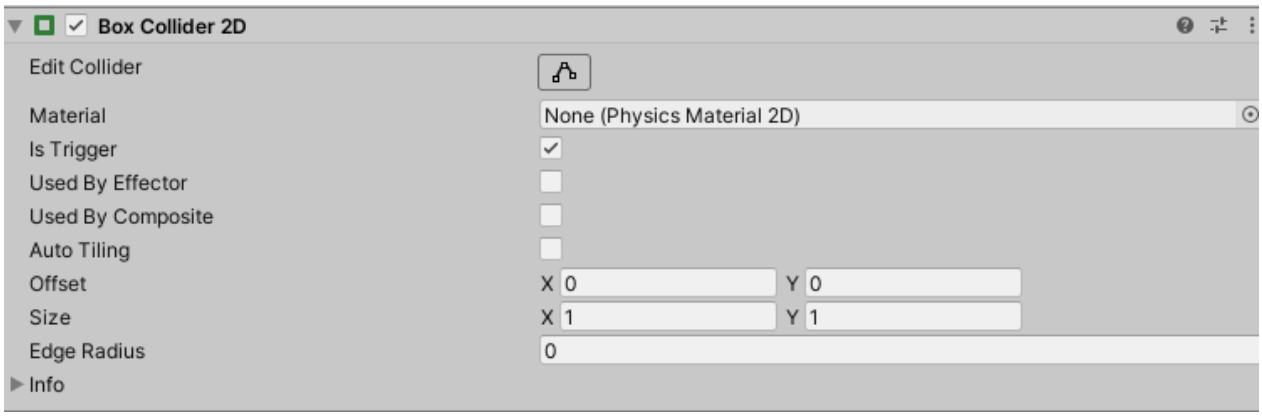


Рисунок 24 - Box Collider

Для появления надписи о окончании уровня нужно добавить объекты типа UI. Была добавлена кнопка NextLevelButton и надпись «Ты победил». В свойстве text у кнопки NextLevelButton было написано «Следующий уровень» (Рисунок 25).

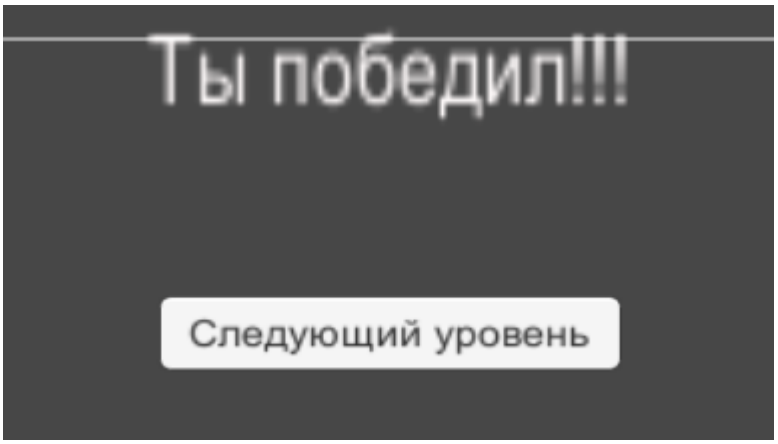


Рисунок 25 - Текст победителя

Необходимо обработать появление этих надписей, после попадания персонажа в зону победы. Был создан скрипт на языке C#, который описывает событие попадания персонажа в зону победы. В методе Start, значение кнопки NextLevelButton и текста WinText указано как false. Это значит, что при старте уровня они не отображаются, но когда персонаж попадает в зону победы, то они появляются (Рисунок 26).

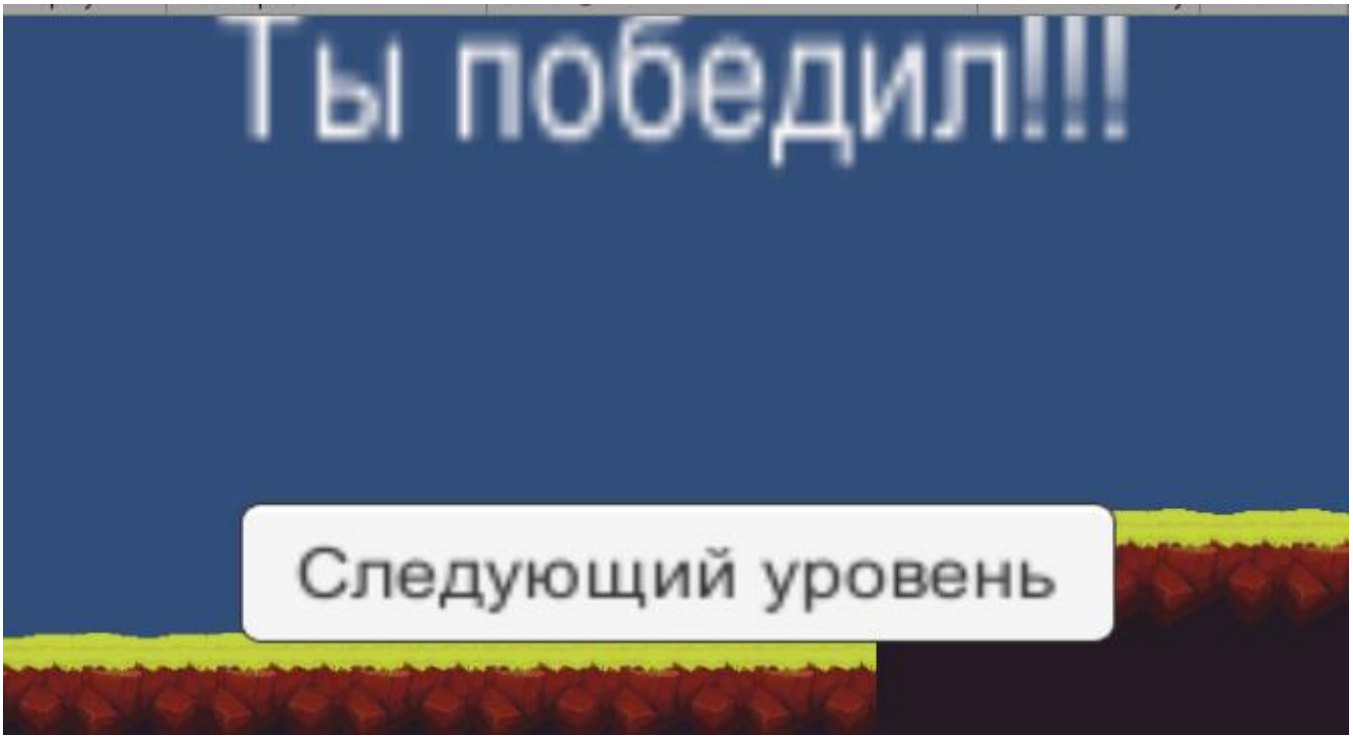


Рисунок 26 - WinZone

После прохождения уровня, игра предлагает перейти на следующий уровень.

Весь код можно посмотреть в приложении А.

2.5 Главное меню.

Для создания главного меню необходимо поместить на экран две кнопки и текст с названием игры (Рисунок 27).

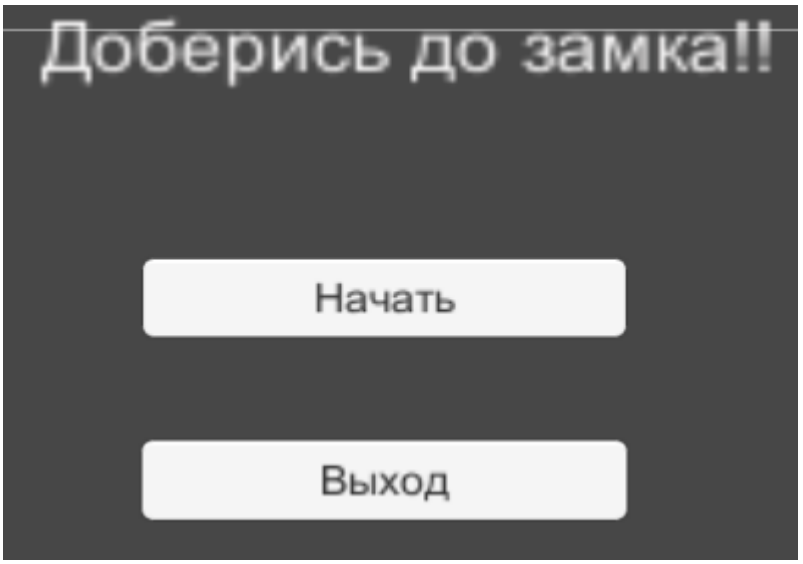


Рисунок 27- Главное меню

Далее необходимо создать скрипт на языке C#, который будет обрабатывать кнопки.

Был создан скрипт с название MainMenu, в котором описано, что при нажатии кнопки Начать, будет загружаться первый уровень, а при нажатии кнопки выйти будет происходить завершение работы игры (Рисунок 28).

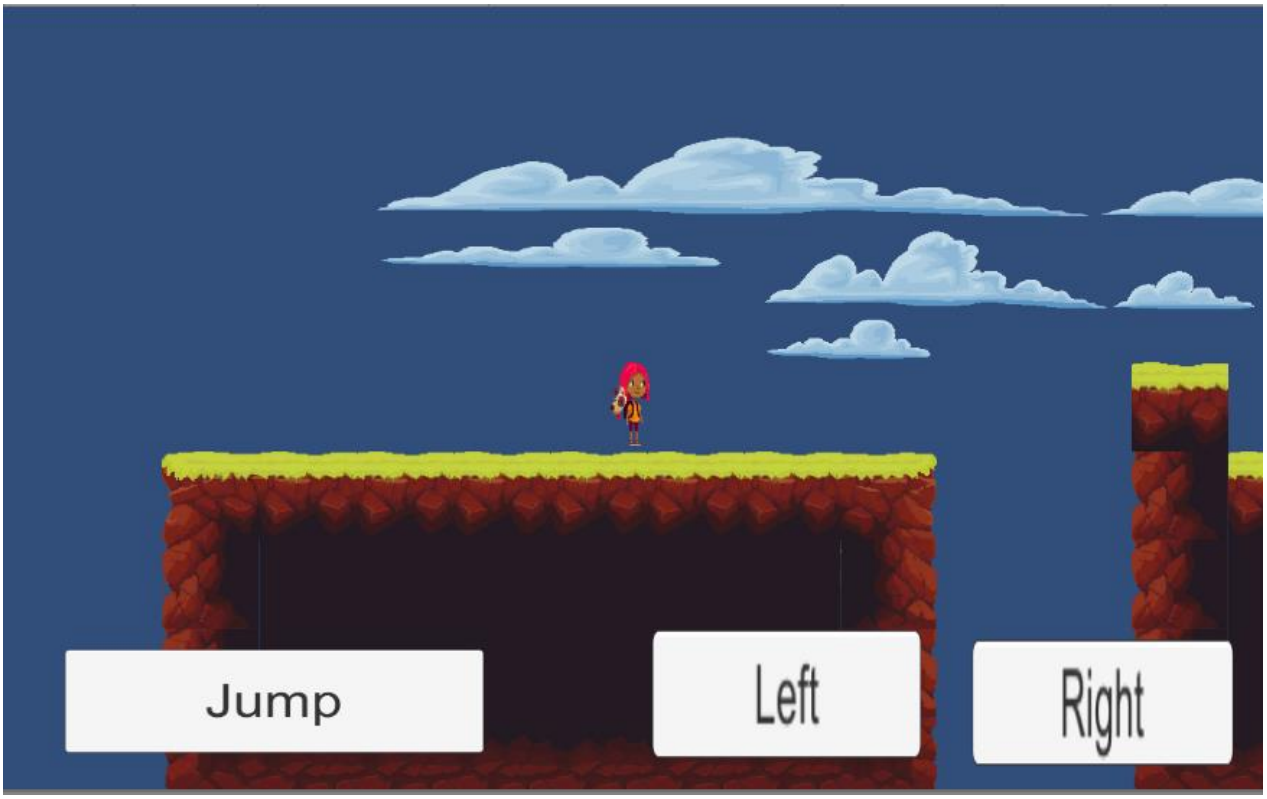


Рисунок 28 - Загрузка первого уровня после нажатия кнопки Начать

3. Запуск игры на мобильном устройстве

3.1 Сборка проекта под операционную систему Android.

Для сборки проекта под Android необходимо открыть панель Build Settings и выбрать там операционную систему Android (Рисунок 29)

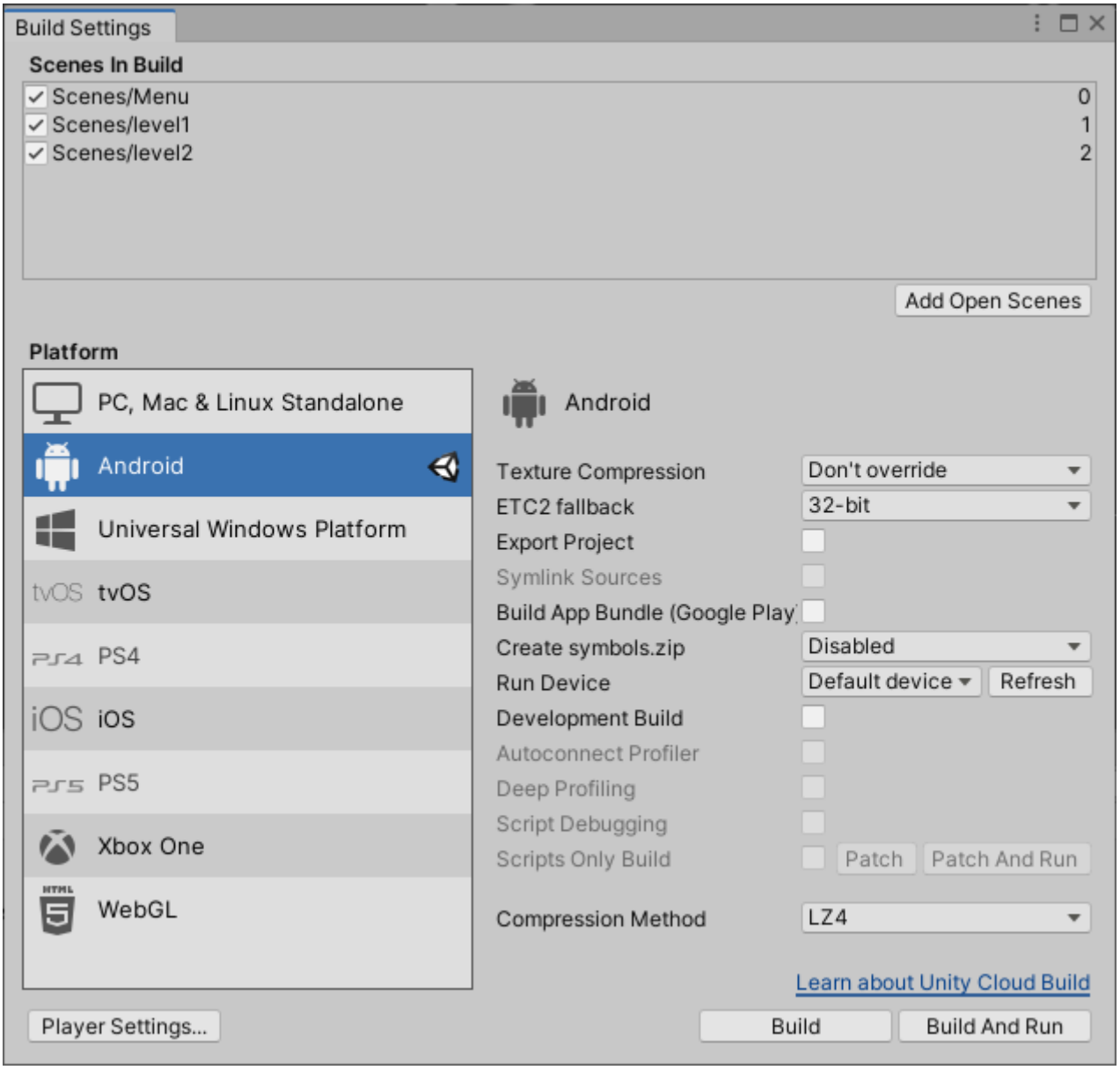


Рисунок 29 - Buid Settings

В этих настройках необходимо выбрать все сцены, которые нужно будет собрать.

Так же нужно нажать на кнопку Player Settings… , после чего откроется новое окно, в котором можно дополнительно настроить игру (Рисунок 30).

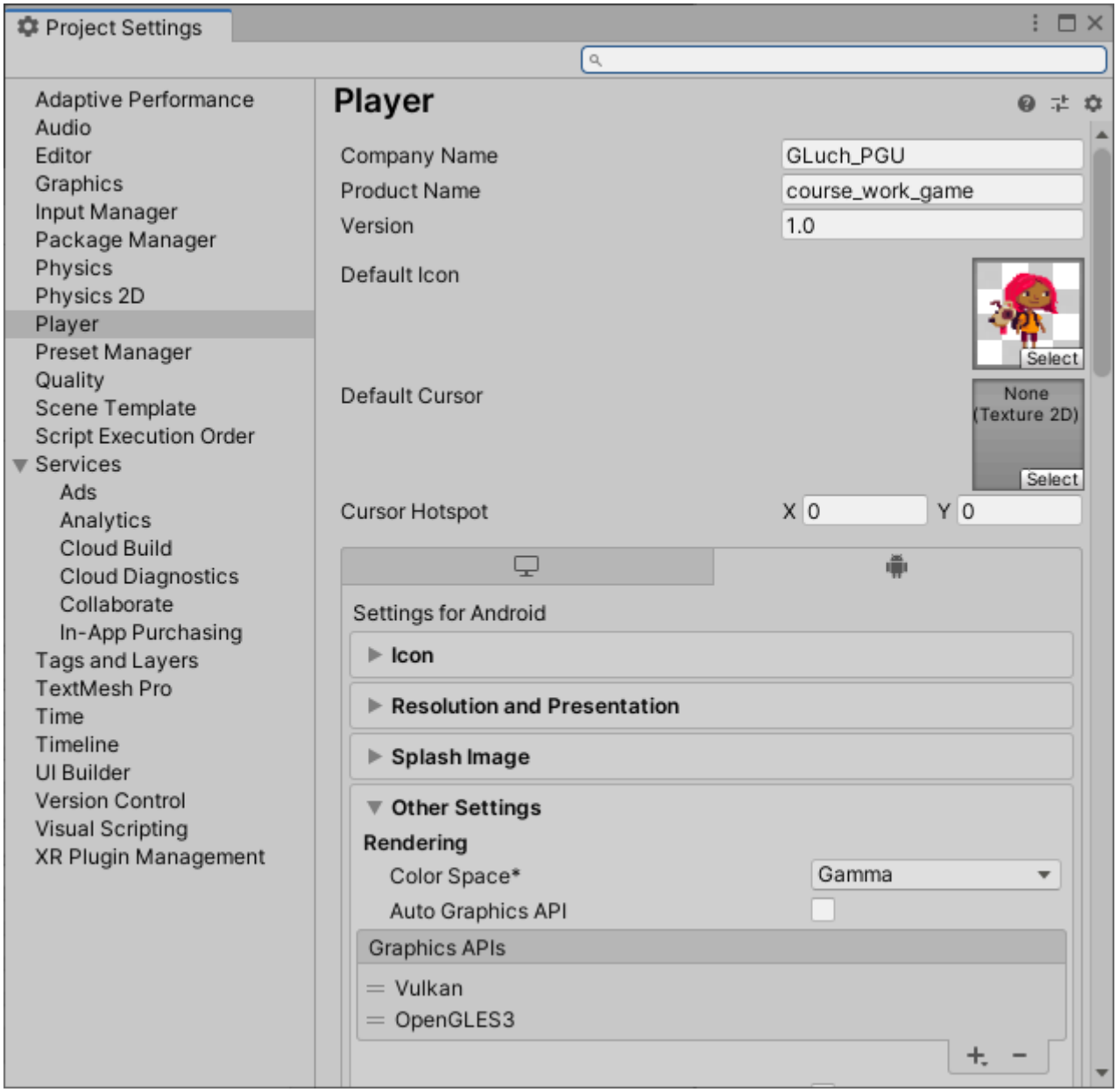


Рисунок 30 - Player Settings…

В Player Settings можно указать имя компании, название продукта и указать его версию. Так же можно установить значок, на который пользователь в дальнейшем будет нажимать, чтобы запустить игру.

После настройки необходимо закрыть окно Player Settings и нажать на кнопку Build. В начале ничего не получится, появится окно, что не удается устройство, под которое нужно собрать проект. Чтобы избежать этой проблемы необходимо подключить телефон к компьютеру через usb.

Но прежде необходимо активировать режим разработчика на своем телефоне. Для этого нужно открыть настройки, и найти пункт «о телефоне», в котором будет пункт «Версия ядра» (Рисунок 31). На этот пункт нужно нажать около 10 раз, после чего откроется режим разработчика (Рисунок 32).

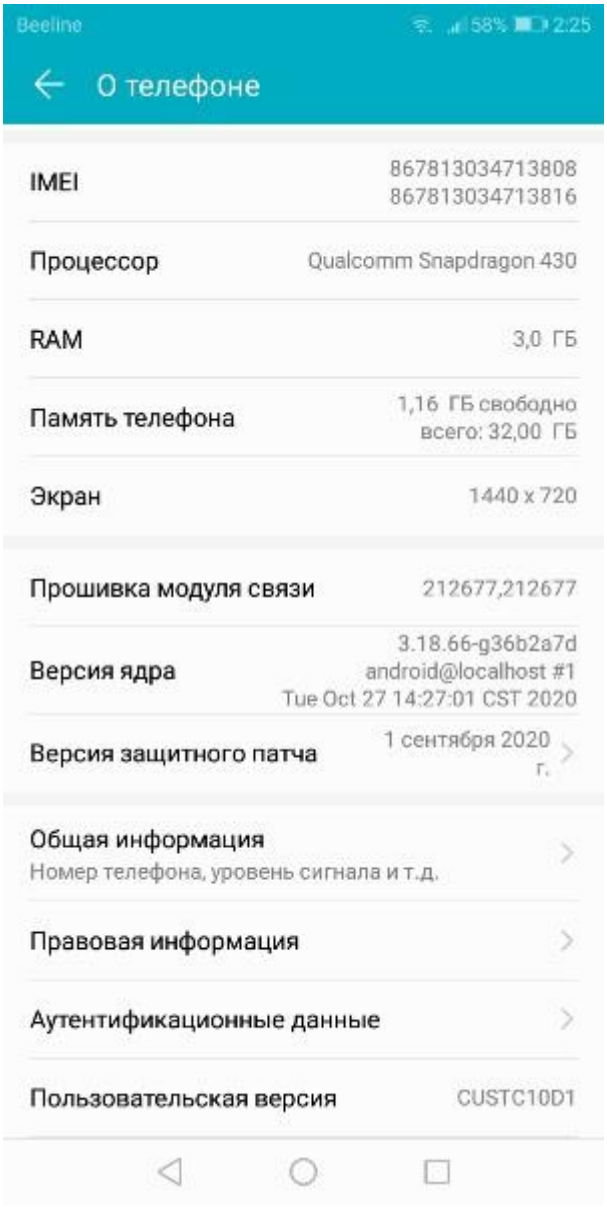


Рисунок 31 - Версия ядра

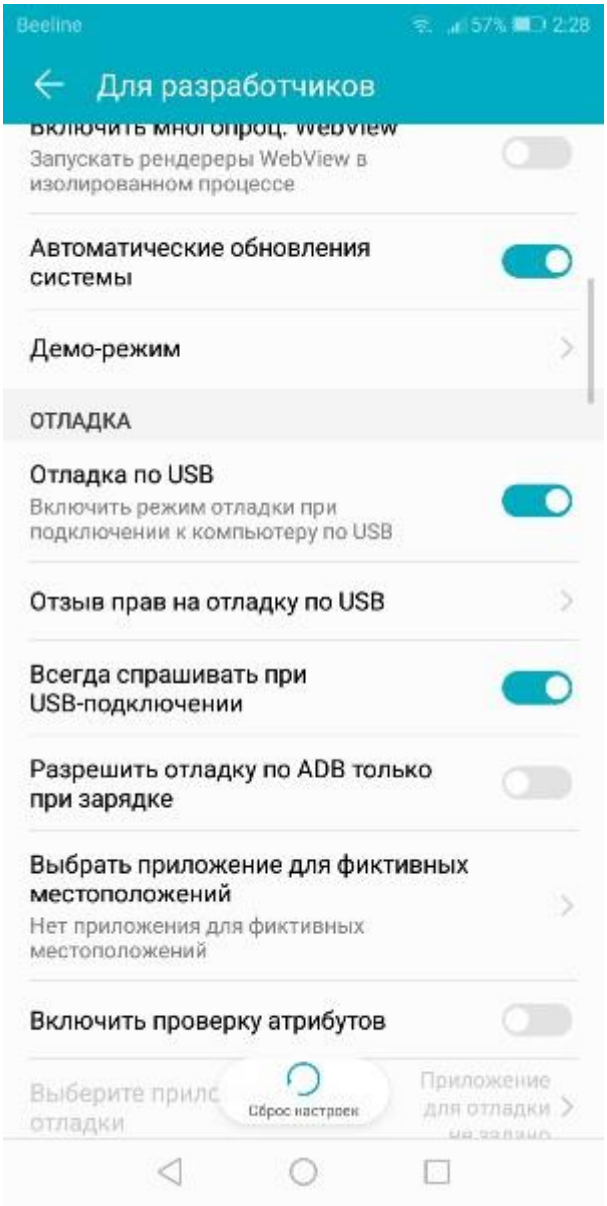


Рисунок 32 - Режим разработчика

В режиме разработчика необходимо активировать пункт «Откладка по USB».

После этих действий, Unity сможет собрать проект под Android. После сборки создается файл с разрешением apk (Рисунок 33)

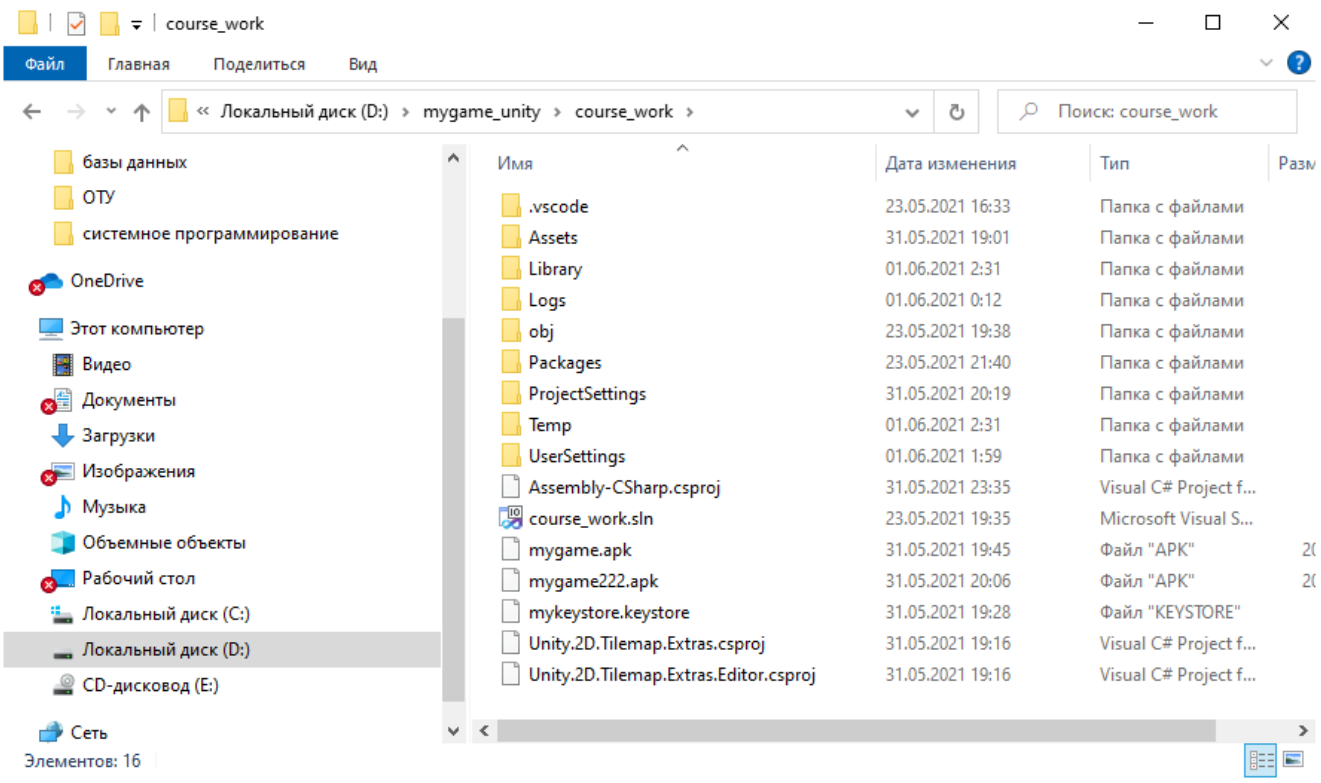


Рисунок 33 - apk файл

Этот файл необходимо перенести на телефон в любую папку.

3.2 Запуск игры на телефоне

Нужно воспользоваться любым файловым менеджером для телефона. Там нужно найти папку с файлом, у которого расширение apk (Рисунок 34)

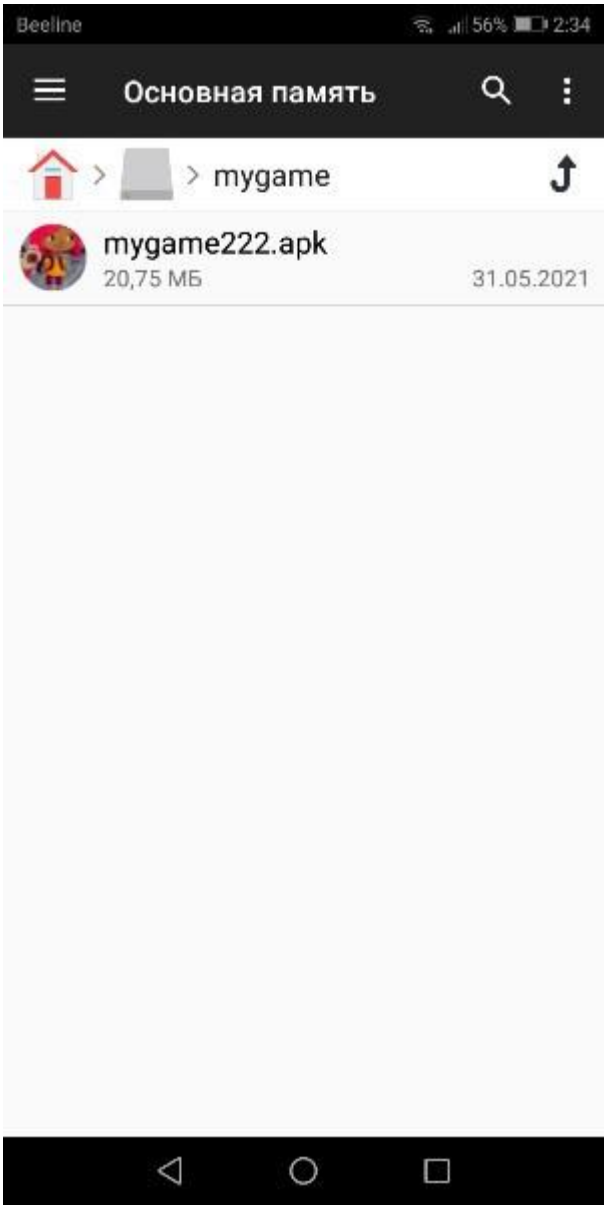


Рисунок 34 - apk файл на телефоне

Затем необходимо нажать на него, и установить, выполнив все условия. После установки появится иконка на экране телефона (Рисунок 35).

Рисунок 35 - Установленная игра

Для запуска игры нужно нажать на иконку, после чего откроется главный экран (Рисунок 36), и после нажатия кнопки Начать загрузится первый уровень (Рисунок 37).

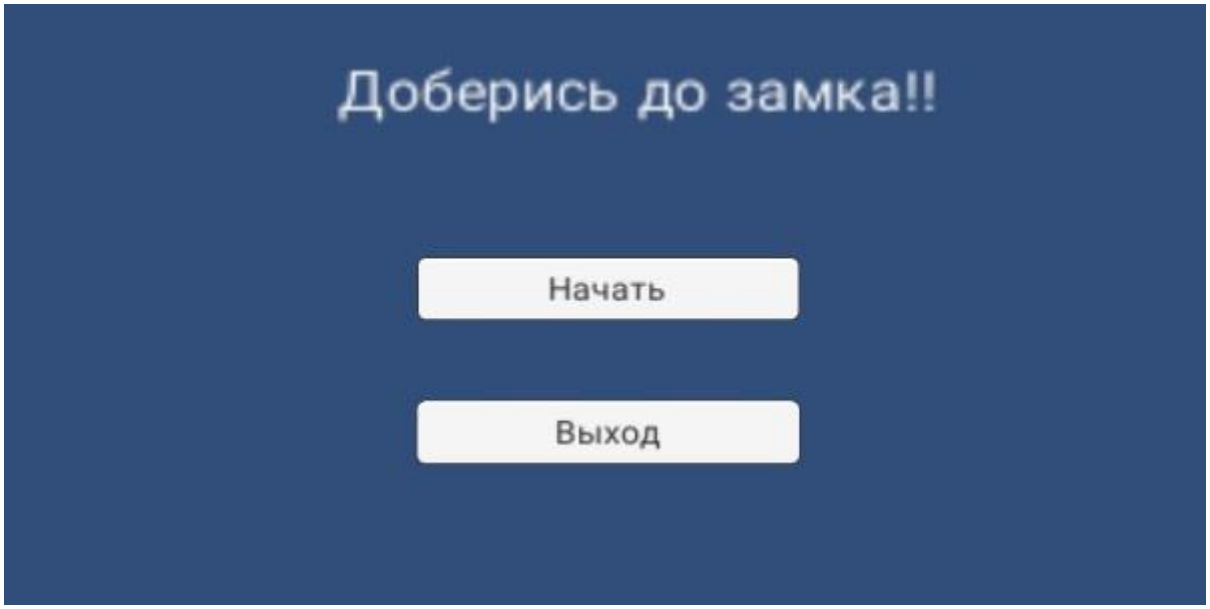


Рисунок 36 - Начальный экран

Рисунок 37 - Первый уровень

Заключение

В ходе выполнения курсовой работы была разработана игра для мобильного устройства с операционной системой Android. Было разработано окружение персонажа, создано движение персонажа, и предусмотрено его управление. Созданы зоны смерти и зоны победы. Создано главное меню, из которого можно запустить игру. В итоге игра была собрана для запуска на Android с помощью средств среды разработки Unity. Проведённые испытания показали, что игра работает без ошибок.

Список литературы.

1. https://www.bestreferat.ru/referat-204405.html

2. Майк Гейг: Разработка игр на Unity 2018 за 24 часа

3. Троелсен, Джепикс: Язык программирования C# 7 и платформы .NET и .NET Core

Приложение А-1

(Текст скрипта Player Controller)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class PlayerController : MonoBehaviour
{

public bool grounded;
public Transform groundChecker;
public LayerMask ground;
public float jumpHeight = 300;
public float radiusCheck = 0.5f;

private Rigidbody2D rigidbody2D;
private Animator animator;
private float speed = 10.0f;
private bool facingRight = true;
public GameObject leftButton;
public GameObject rightButton;

public bool isGoLeft;
public bool isGoRight;

public Button jumpButton;
// Start is called before the first frame update
void Start()
{
rigidbody2D = gameObject.GetComponent<Rigidbody2D>();
animator = gameObject.GetComponent<Animator>();
jumpButton.onClick.AddListener(Jump);
```

```
SetTrigerMove();


}

void SetTrigerMove(){
SetTrigerLeft();
SetTrigerRight();
}

void SetTrigerLeft(){
EventTrigger eventTrigger = leftButton.GetComponent<EventTrigger>();
EventTrigger.Entry eventDowm = new EventTrigger.Entry();
eventDowm.eventID = EventTriggerType.PointerDown;
eventDowm.callback.AddListener((data)=>{isGoLeft = true;});
EventTrigger.Entry eventUp = new EventTrigger.Entry();
eventUp.eventID = EventTriggerType.PointerUp;
eventUp.callback.AddListener((data)=>{isGoLeft = false;});
eventTrigger.triggers.Add(eventUp);
eventTrigger.triggers.Add(eventDowm);
}

void SetTrigerRight(){
EventTrigger eventTrigger = rightButton.GetComponent<EventTrigger>();
EventTrigger.Entry eventDowm = new EventTrigger.Entry();
eventDowm.eventID = EventTriggerType.PointerDown;
eventDowm.callback.AddListener((data)=>{isGoRight = true;});
EventTrigger.Entry eventUp = new EventTrigger.Entry();
eventUp.eventID = EventTriggerType.PointerUp;
eventUp.callback.AddListener((data)=>{isGoRight = false;});
eventTrigger.triggers.Add(eventUp);
eventTrigger.triggers.Add(eventDowm);


}
```

```
// Update is called once per frame
void Update()
{

}

void Jump(){
if(grounded){
rigidbody2D.AddForce(new Vector2(0, jumpHeight));
animator.SetBool("grounded",false);
}
}
void FixedUpdate(){


grounded = Physics2D.OverlapCircle(groundChecker.position, radiusCheck, ground);
animator.SetBool("grounded",grounded);
if (Input.GetKeyDown(KeyCode.Space)){
Jump();

}

float y = rigidbody2D.velocity.y;
float move = Input.GetAxis("Horizontal");
if(move == 0){
if(isGoLeft) move = -1;
if(isGoRight) move = 1;
}

float currentSpeed = move*speed;
rigidbody2D.velocity = new Vector2(currentSpeed,y);
```

```
animator.SetFloat("speed",Mathf.Abs(currentSpeed));

if (move>0 && !facingRight){
flip();
}
else if(move<0 && facingRight){
flip();
}
}

void flip() {
facingRight = !facingRight;
Vector3 vector3 = transform.localScale;
vector3.x = vector3.x * -1;
transform.localScale=vector3;
}
}
```

Приложение А-2

(Текст скрипта WinZone)

```
public class WinZone : MonoBehaviour
{

public Button nextLevelButton;
public Text winText;

// Start is called before the first frame update
void Start()
{
nextLevelButton.gameObject.SetActive(false);
winText.enabled = false;
nextLevelButton.onClick.AddListener(OnClickNextLevel);
}

void OnClickNextLevel(){
SceneManager.LoadScene("level2");
}

// Update is called once per frame
void Update()
{

}

private void OnTriggerEnter2D(Collider2D collision){
if (collision.gameObject.tag == "Player"){
nextLevelButton.gameObject.SetActive(true);
winText.enabled = true;
}
}

}
```

Приложение А-3

(Текст скрипта DeadZone)

```
private void OnTriggerEnter2D(Collider2D collision){

var spawnPosition = spawn.transform.position;
collision.gameObject.transform.position = new Vector3(spawnPosition.x,
spawnPosition.y,0);

}
```