Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»  
(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.03.04 «Программная инженерия»

Профиль «Беспрофильный»

**КУРСОВОЙ ПРОЕКТ**

по дисциплине Конструирование программного обеспечения

(наименование дисциплины)

на тему «Разработка и проектирование игрового приложения. Комплексный проект. Вариант 1»

(тема курсового проекта)

Выполнила студентка   Буниф Мажда   курса 3 группы 18-КБ-ПР2

(фамилия, имя, отчество)

Допущен к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель (нормоконтроллер) проекта ст. преп. А.Г. Волик

(должность подпись, дата)

Защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка

(дата)

Члены комиссии А.Г. Мурлин

К.Е. Тотухов

(должность, подпись, дата, расшифровка подписи)

Краснодар

2020 г.

**ФГБОУ ВО «Кубанский Государственный Технологический Университет»**

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.03.04 "Программная инженерия"

Профиль «Беспрофильный»

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Зав. кафедрой М.В. Янаева \_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

**ЗАДАНИЕ**

на курсовой проект

Студентке Буниф Мажде курса 3 группы 18-КБ-ПР2

Тема проекта «Разработка и проектирование игрового приложения. Комплексный проект. Вариант 1»

(утверждена указанием директора института №\_\_\_ от \_\_\_\_\_ 20\_\_ г.)

План работы:

1. Анализ предметной области

2. Составление технического задания

3. Проектирование и разработка программного продукта

Объем проекта:

а) пояснительная записка 27 с.

б) программа 6 с.

Рекомендуемая литература:

1. Паттерны проектирования. – СПБ.: Питер, 2011. – 656 с.: ил. в

2. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2015. – 368 с.: ил.

Срок выполнения проекта: с " 14 " сентября по " 13 " декабря 2020 г.

Срок защиты: " 20 " декабря 2020 г.

Дата выдачи задания: " 14 " сентября 2020 г.

Дата сдачи проекта на кафедру: " 14 " декабря 2020 г.

Руководитель проекта ст. преподаватель Волик А.Г.

(должность, подпись)

Задание приняла студентка Буниф М.

(подпись)

**Реферат**

Пояснительная записка курсового проекта содержит 27 страниц, 8 рисунков, 5 источников.

КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РАЗРАБОТКА ИГРОВОГО ПРИЛОЖЕНИЯ, ДВУМЕРНАЯ ГРАФИКА, ГРАФИЧЕСКАЯ ПРИКЛЮЧЕНЧЕСКАЯ ИГРА

Объектом исследования является игровое приложение «MVRunner», данный продукт представляет собой приключенческую игру с двумерной графикой.

Цель работы состоит в изучении методов разработки и проектирования игровых приложений, а также в закреплении правил и навыков конструирования программ с использованием языка программирования С#.

К полученным результатам относится игровое приложение в среде разработки Unity.

**Содержание**

[Введение 5](#_Toc60012852)

[1 Нормативные ссылки 6](#_Toc60012853)

[2 Постановка задачи 7](#_Toc60012854)

[3 Описание предметной области 8](#_Toc60012855)

[4 Техническое задание 9](#_Toc60012856)

[4.1 Основание для разработки 9](#_Toc60012857)

[4.2 Назначение разработки 9](#_Toc60012858)

[4.3 Требования к программному продукту 9](#_Toc60012859)

[4.3.1 Требования к функциональным характеристикам 9](#_Toc60012860)

[4.3.2 Требования к надежности и безопасности 10](#_Toc60012861)

[4.3.3 Требования к информационной и программной совместимости 11](#_Toc60012862)

[4.3.4 Требования к составу и параметрам технических средств 11](#_Toc60012863)

[4.3.5 Требования к программной документации 12](#_Toc60012864)

[4.3.6 Технико-экономическое обоснование 12](#_Toc60012865)

[4.3.7 Стадии и этапы разработки 12](#_Toc60012866)

[4.3.8 Порядок контроля и приемки 12](#_Toc60012867)

[5 Проектирование программного продукта 13](#_Toc60012868)

[5.1 Диаграмма классов 13](#_Toc60012869)

[На представленном выше рисунке изображена диаграмма классов. В программе реализовано 7 классов: 13](#_Toc60012870)

[5.2 Диаграмма вариантов использования 14](#_Toc60012871)

[5.3 Дерево поведений 15](#_Toc60012872)

[6 Разработка программного продукта 16](#_Toc60012873)

[6.1 Анимация 16](#_Toc60012874)

[7 Руководство пользователя 17](#_Toc60012875)

[Заключение 20](#_Toc60012876)

[Список используемых источников 21](#_Toc60012877)

[Приложение А – Листинг программы 22](#_Toc60012878)

[Приложение Б – Проверка на антиплагиат 28](#_Toc60012879)

# **Введение**

С развитием цифровых технологий компьютеры стали неотъемлемой частью жизни современного человека. Если раньше ЭВМ использовались только для решения сложных математический задач, то сегодня сфера применения настолько широка, что человечество уже физически не сможет представить свою жизнь без компьютеров и гаджетов.

Следствием данного развития стало то, что одним из наиболее массовых применений электронных вычислительных машин являются игровые приложения.

Развитие игровой индустрии с каждым годом набирает очень серьезные обороты. Над одним проектом работает уже не один человек, а целая команда разработчиков. Так-как данное приложение разрабатывается для курсовой работы, команда разработчиков состоит из двух человек.

Актуальность темы заключается в том, что на сегодняшний день разработка и проектирование игр пользуется хорошим спросом на рынке IT-продуктов и разработчики в данной области ценятся среди специалистов.

В рамках курсовой работы будут рассмотрены все этапы разработки и проектирования игрового приложения «MVRunner».

# **1 Нормативные ссылки**

В данном документе использованы ссылки на следующие стандарты:

ГОСТ 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления

ГОСТ Р 1.12-2004 Стандартизация в Российской Федерации. Термины и определения

ГОСТ 7.12-93 СИБИД. Библиографическая запись. Сокращения слов на русском языке. Общие требования и правила

ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления.

# **2 Постановка задачи**

В данном разделе курсовой работы задачи поставлены следующим образом:

* исследование предметной области;
* анализ полученных данных;
* разработка технического задания;
* проектирование архитектуры приложения;
* создание прототипа приложения;
* доработка функциональности;
* выпуск в эксплуатацию.

# **3 Описание предметной области**

Раннее была рассмотрена актуальность темы разработки и проектирования игрового приложения, которая показала, что проект действительно может-быть перспективен в будущем.

На данном этапе курсовой работы рассмотрим основную идею и сюжет игрового приложения.

Зайдя немного вперед, упоминаю о том, что перед началом разработки было принято решение создать игру с двумерной графикой, то есть все действия происходят в 2D.

Жанр игрового приложения – аркада, характеризуется коротким по времени, но интенсивным игровым процессом.

Идея игры заключается в следующем, пользователь управляет персонажем, задача которого заключается:

- собирать «coins»;

- обходить препятствия;

- не попадаться персонажам вредителям.

Цель игры преодолеть максимально возможное расстояние и собрать максимальное количество «coins».

В ходе игры пользователь может ознакомиться с интерфейсом игрового приложения, изучить элементы управления персонажем, что будет рассмотрено в дальнейших разделах, а также познакомиться с персонажами игрового мира данного приложения.

# **4 Техническое задание**

## **4.1 Основание для разработки**

Основанием для разработки и тестирования программного обеспечения является указание «О закреплении тем и назначении руководителей курсовых работ».

Организация, утвердившая договор: Кубанский государственный технологический университет.

Наименование курсовой работы: разработка и проектирование игрового приложения.

## **4.2 Назначение разработки**

Назначением курсовой работы является разработка и проектирование игрового приложения с двумерной графикой. Предназначенная в развлекательных целях.

## **4.3 Требования к программному продукту**

### 4.3.1 Требования к функциональным характеристикам

Функциональными характеристиками - является набор всех эксплуатационных параметров любого приложения.

На данном этапе сформулированы следующие требования к функциональным характеристикам программного продукта:

* удобный интерфейс пользователя;
* сохранение личного рекорда пользователя;
* сохранение количества собранных «coins».

### 4.3.2 Требования к надежности и безопасности

Надежность – свойство объекта сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных условиях применения, технического обслуживания, хранения и транспортирования.

Надёжность программного обеспечения определяется его безотказностью и восстанавливаемостью.

Безотказность программного обеспечения – это свойство программы сохранять работоспособность при использовании в процессе обработки информации в компьютере.

Восстанавливаемость программного обеспечения – это характеристика, указывающая на затраты времени и труда на устранение отказа из-за появившейся ошибки в программе и ее последствии.

Надежность обеспечена с помощью:

* возможность восстановления после сбоя;
* возможность резервного копирования;
* своевременное выполнение процессов администрирования;
* соблюдение правил эксплуатации;
* соблюдение правил технического обслуживания программно- технических средств.

Безопасность – комплекс мер, направленных на защиту программного обеспечения от несанкционированного приобретения, использования, распространения, модифицирования, изучения и воссоздания аналогов.

Безопасность обеспечена с помощью:

* ограничение несанкционированного доступа к данным;
* исключение несанкционированного копирования.

Разработанный программный продукт имеет возможность сохранять данные во времени, в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях эксплуатации.

### 4.3.3 Требования к информационной и программной совместимости

Совместимость – это способность аппаратных либо программных средств взаимодействовать друг с другом.

Информационная совместимость - способность двух или более систем адекватно воспринимать одинаково представленные данные.

Программная совместимость - способность выполнения одинаковых программ с получением одних и тех же результатов.

Программная совместимость обеспечивается единым набором команд, единой формой представления данных, единой системой адресации. Это позволяет разрабатывать программы независимо от конкретной ЭВМ, применять общие для большинства ЭВМ операционные системы, создавать единый фонд пакетов прикладных программ.

При наличии аппаратной, информационной и программной совместимости устройств без ограничений для конечных пользователей говорят о полной совместимости этих устройств.

Разработка приложения выполнена с помощью языка программирования C# в межплатформенной среде разработки компьютерных игр Unity.

Разработанное игровое приложение «MVRunner» должно функционировать с операционными системами Windows.

### 4.3.4 Требования к составу и параметрам технических средств

Процессор intel core серии 2 duo и выше, 2 ГБ оперативной памяти, жёсткий диск от 120 ГБ.

### 4.3.5 Требования к программной документации

Руководство пользователя.

### 4.3.6 Технико-экономическое обоснование

Экономический эффект от внедрения игрового приложения ожидается за счет востребованности среди пользователей.

### 4.3.7 Стадии и этапы разработки

После утверждения технического задания было принято решение приступить к проектированию и разработке игрового приложения.

Этапы проектирования:

* выявление архитектуры приложения;
* уточнение функций персонажей;
* постановка взаимосвязей объектов;

К этапам разработки относится написание программного кода для функционирования графических элементов программного продукта.

### 4.3.8 Порядок контроля и приемки

Контроль осуществляется в течении семестра и до момента сдачи курсовой работы.

# **5 Проектирование программного продукта**

## **5.1 Диаграмма классов**

Диаграмма классов – является центральным звеном разработки программного обеспечения.

Класс – совокупность общих признаков заданной группы объектов ПО.

Под отношением классов понимают статическую связь между классами.

На представленном ниже рисунке изображена диаграмма классов ПО (см. рисунок 1).

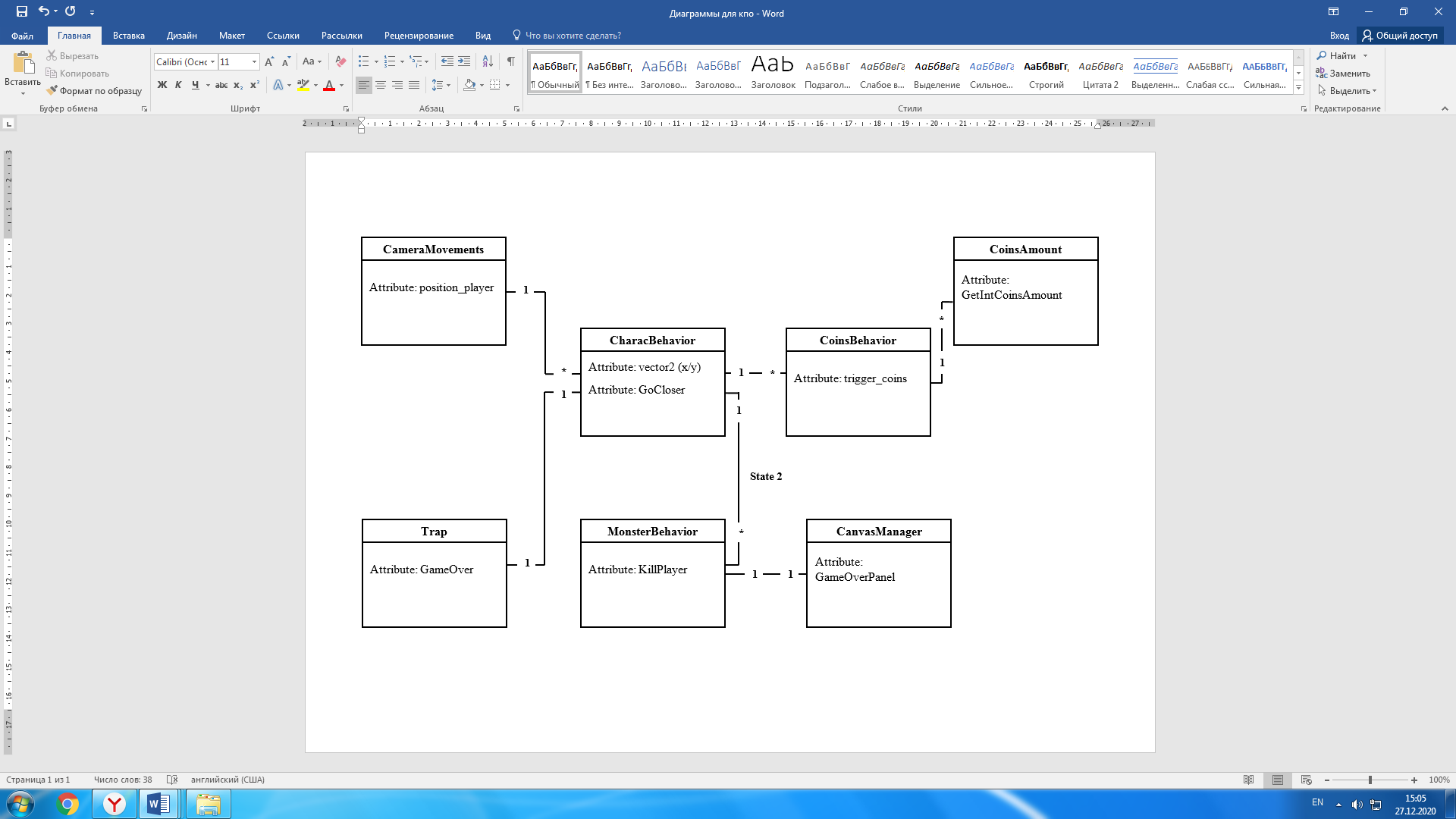


Рисунок 1 – Диаграмма классов

На представленном выше рисунке изображена диаграмма классов. В программе реализовано 7 классов:

* СharacBehavior – класс в котором реализованы методы управления персонажем;
* СameraMovements – класс в котором реализован метод для передвижения камеры, вслед за персонажем;
* Trap – класс в котором реализован метод обнаружения пустого пространства на карте (преграды, которые персонаж должен обойти);
* СoinsBehavior – класс в котором реализованы методы соприкосновения персонажа с coins и подсчет собранных coins;
* СoinsAmount – класс в котором реализован метод сохранения coins;
* MonsterBehavior – класс в котором реализованы методы слежения за главным персонажем, исчезновения с карты и убийства персонажа;
* CanvasManager – класс в котором реализовано окно паузы с кнопками: пауза, рестарт, главное меню. Окно всплывает при проигрыше, либо нажатии кнопки «пауза».

## **5.2 Диаграмма вариантов использования**

На следующем рисунке представлена диаграмма вариантов использования персонажей (см. рисунок 2).

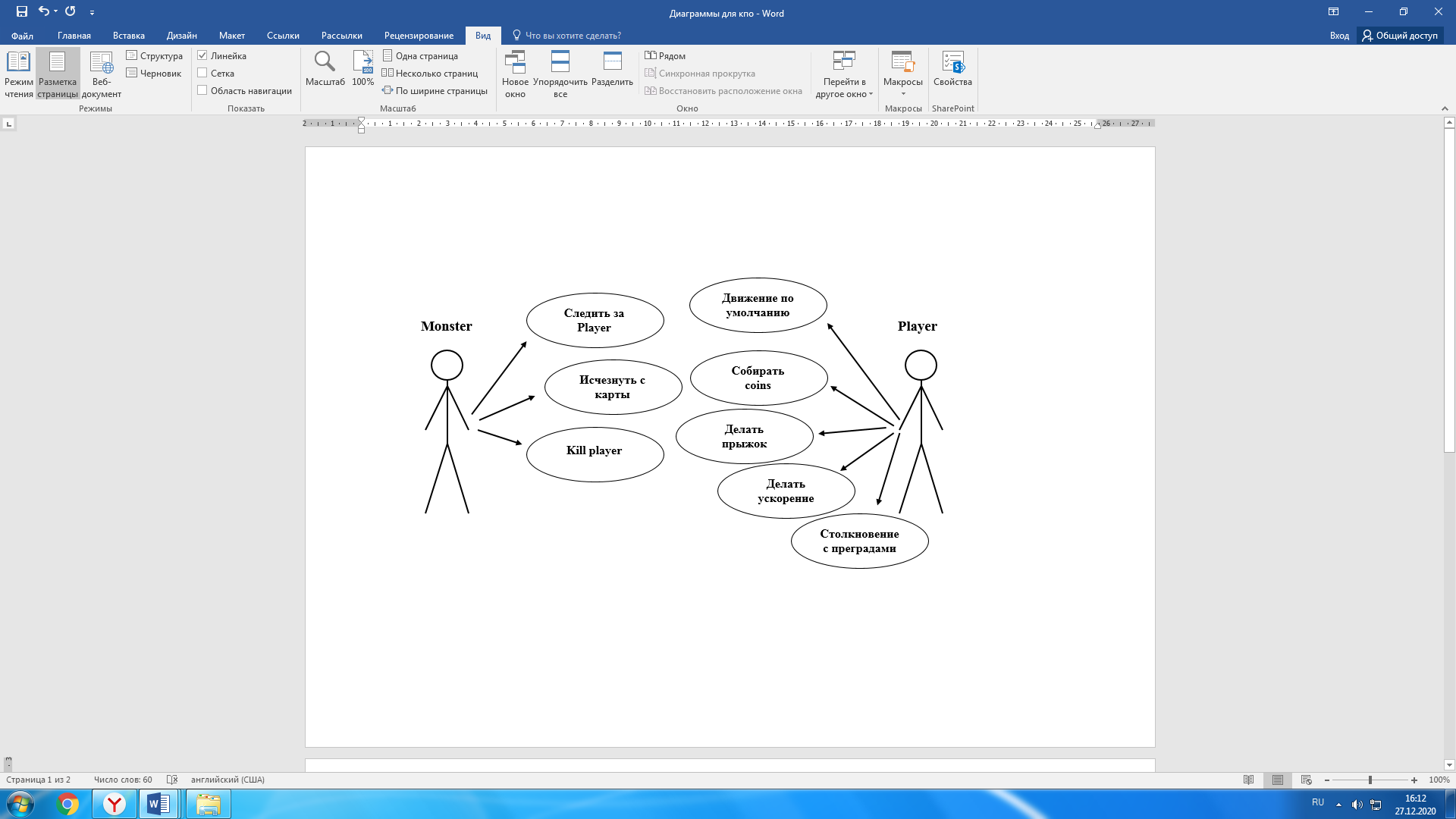


Рисунок 2 – Диаграмма вариантов использования

На рисунке изображены два персонажа с их основными функциями. Пользователь приложения может управлять только персонажем «Player», персонаж «Monster» работает по следующему принципу: если пользователь не обошел преграду 1 раз, то плохой персонаж будет следить за ним 15 секунд, если в течении данного времени пользователь не обойдет еще одну преграду, то игра окончена.

# **5.3 Дерево поведений**

Следующим к рассмотрению представлено дерево поведений в игровом проекте. Дерево поведений представляет собой иерархическую древовидную структуру, в которой в качестве узлов выступают блоки игровой логики (см. рисунок 3).

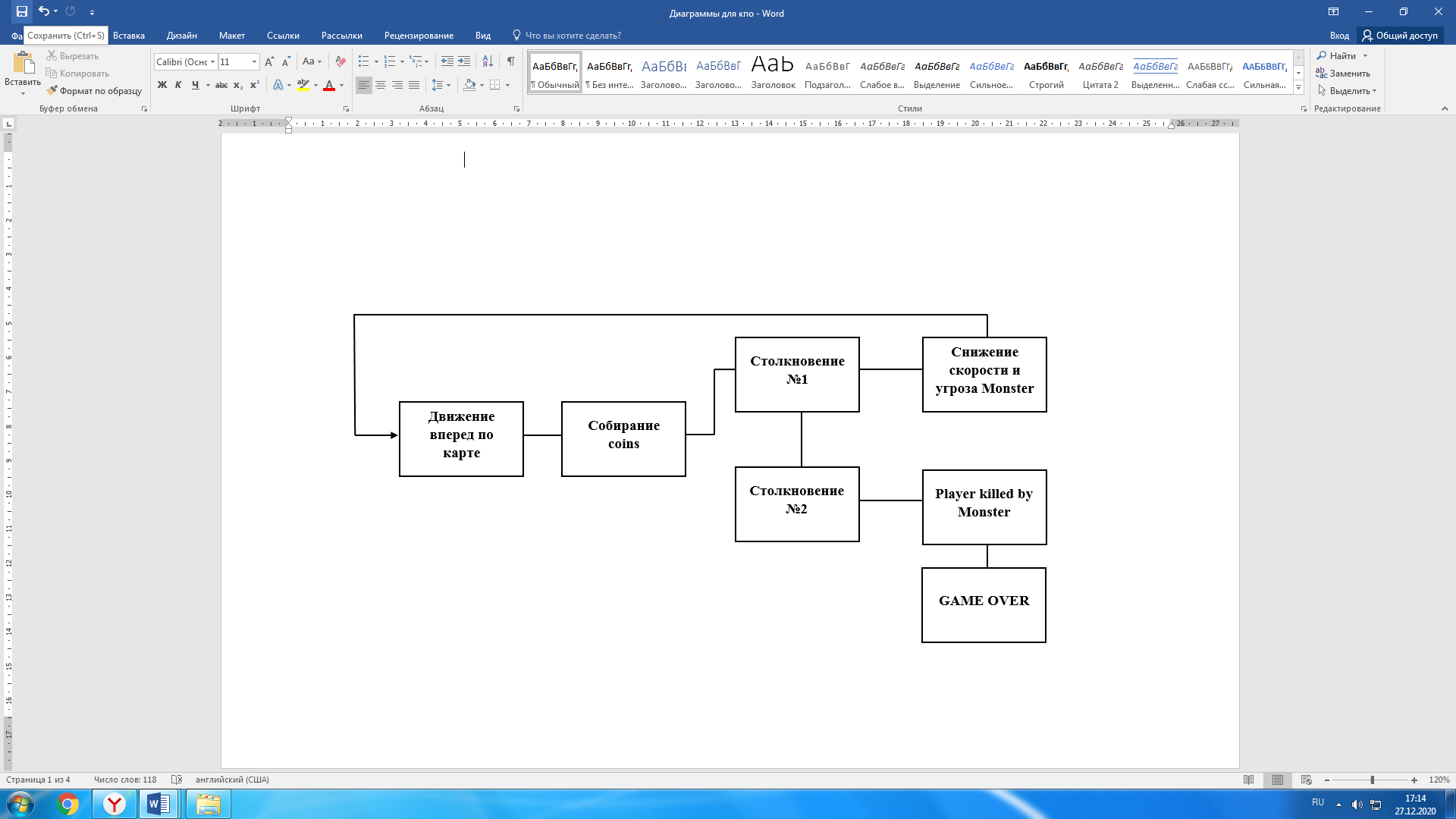


Рисунок 3 – Дерево поведений

Представленное дерево состоит из узлов, которые должны посещаться персонажем, в зависимости от действий игрока выбирается дальнейший путь по представленному дереву.

# **6 Разработка программного продукта**

# **6.1 Анимация**

Работа над игровым проектом «MVRunner» велась в команде, ниже представленные части разработки соответствуют варианту №1.

Анимация позволяет оживить игру даже с самой примитивной графикой. Структура анимации для персонажа «Monster» выглядит следующим образом (см. рисунок 4).

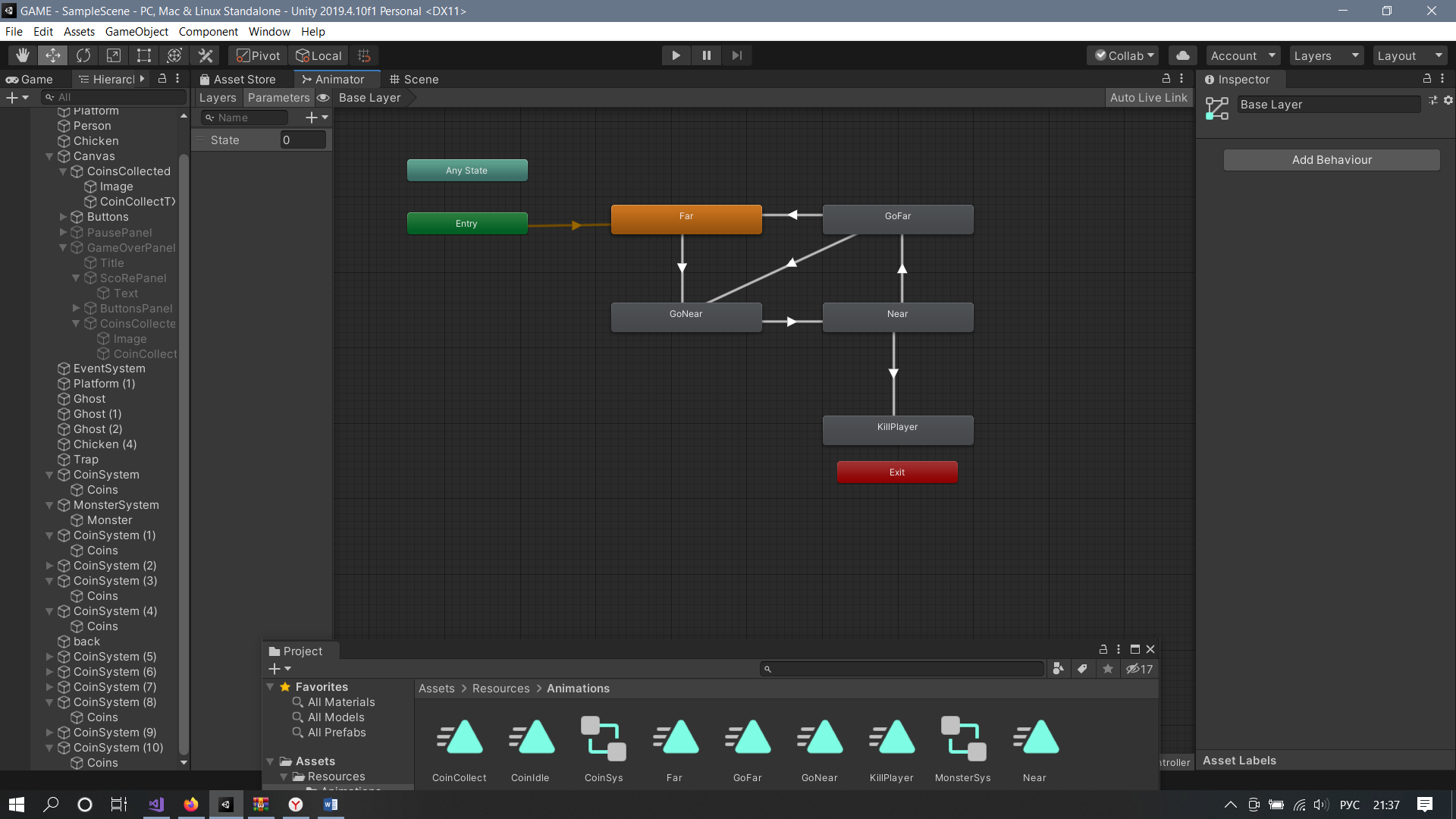


Рисунок 4 – Аниматор в Unity

В платформе Unity устанавливаются связи между блоками анимации. В конечном итоге данная связка позволяет полноценно работать определенному объекту. Предназначение блоков:

* Any State – любая ситуация;
* Entry – начинает работать после запуска программы;
* Far – «Monster» находится далеко не видно в карте;
* GoNear – «Monster» приближается;
* GoFar – «Monster» уходит далеко;
  + Near – «Monster» рядом с «Player»;
  + KillPlayer – «Monster» убивает «Player»;
  + Exit – окончание работы.

# **7 Руководство пользователя**

После запуска игрового приложения главный персонаж начинает движение (см. рисунок 5).



Рисунок 5 – Исходное положение персонажа

При нажатии клавиши «space» персонаж может обходить преграды и собирать coins (см. рисунок 6).

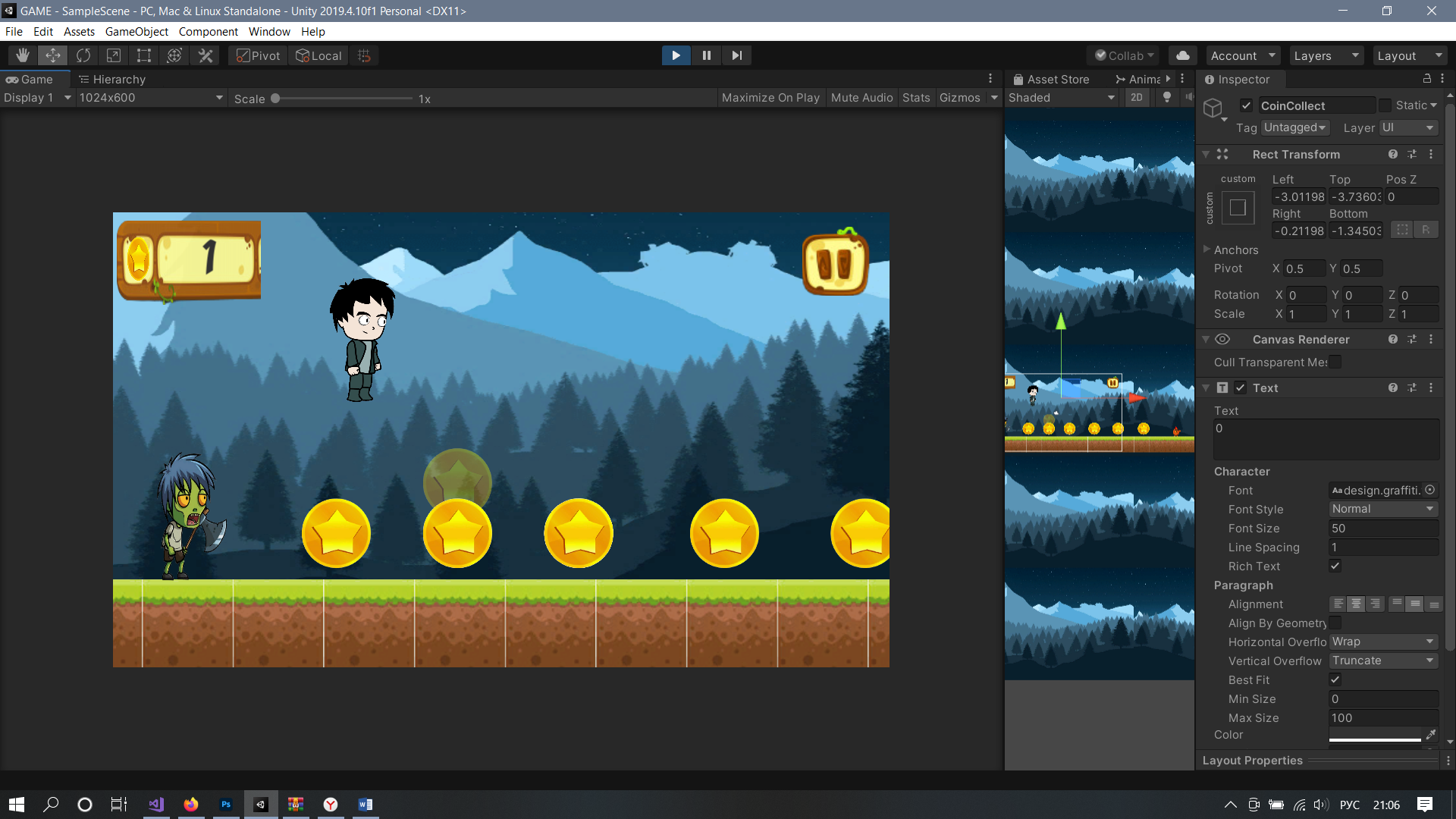


Рисунок 6 – Функция «Прыжок»

Также при нажатии клавиши «Z» скорость персонажа увеличивается, данный момент не зафиксирован в печатной форме.

При столкновении с преградой приближается персонаж «Monster» (см. рисунок 7).

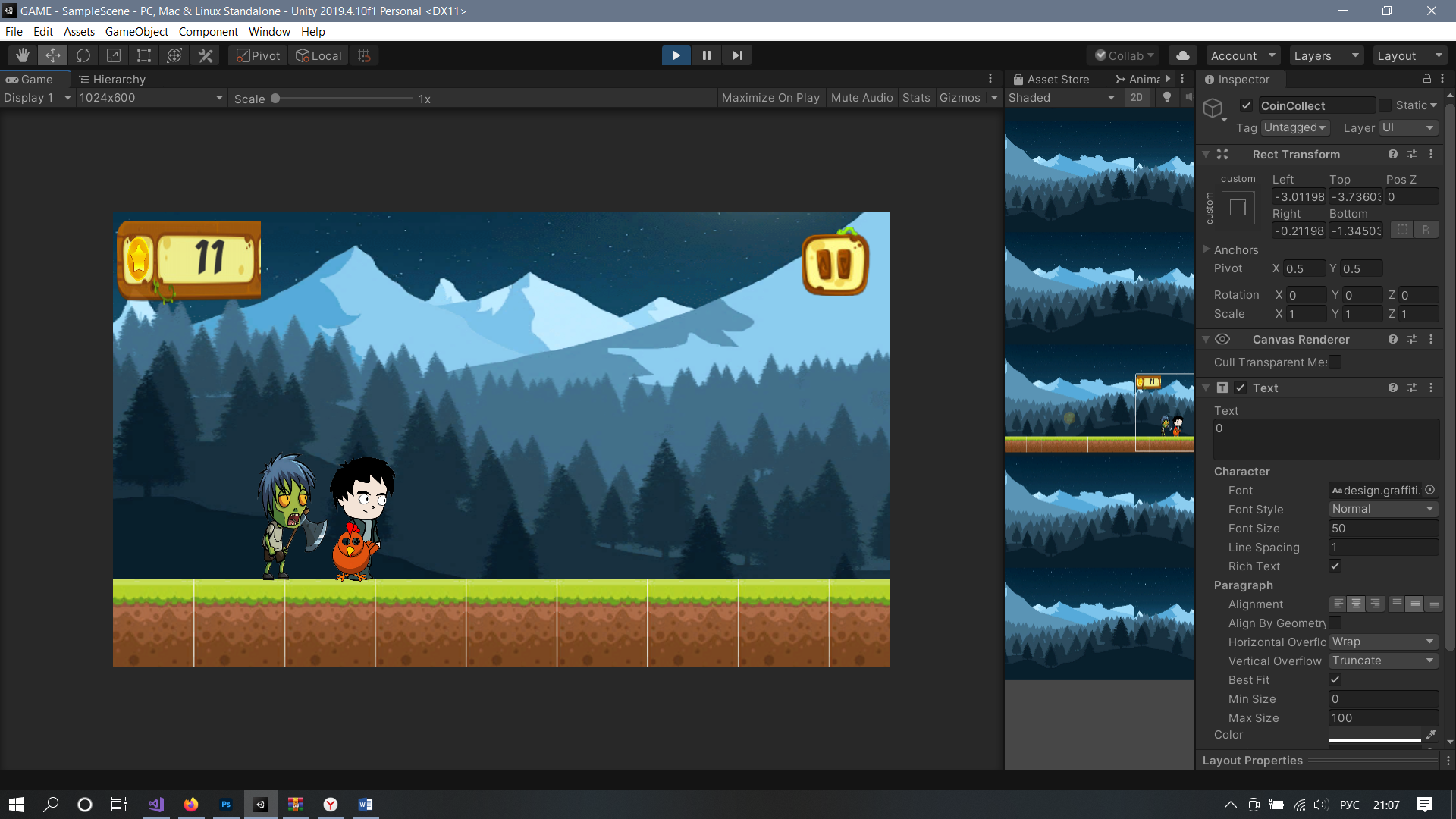


Рисунок 7 – Приближение «Monster»

Если в течении 15 секунд после первого столкновения персонаж будет обходить все преграды, то «Monster» перестанет его преследовать, но в случае того, что персонаж не пройдет какую-либо преграду, то Game over (см. рисунок 8).

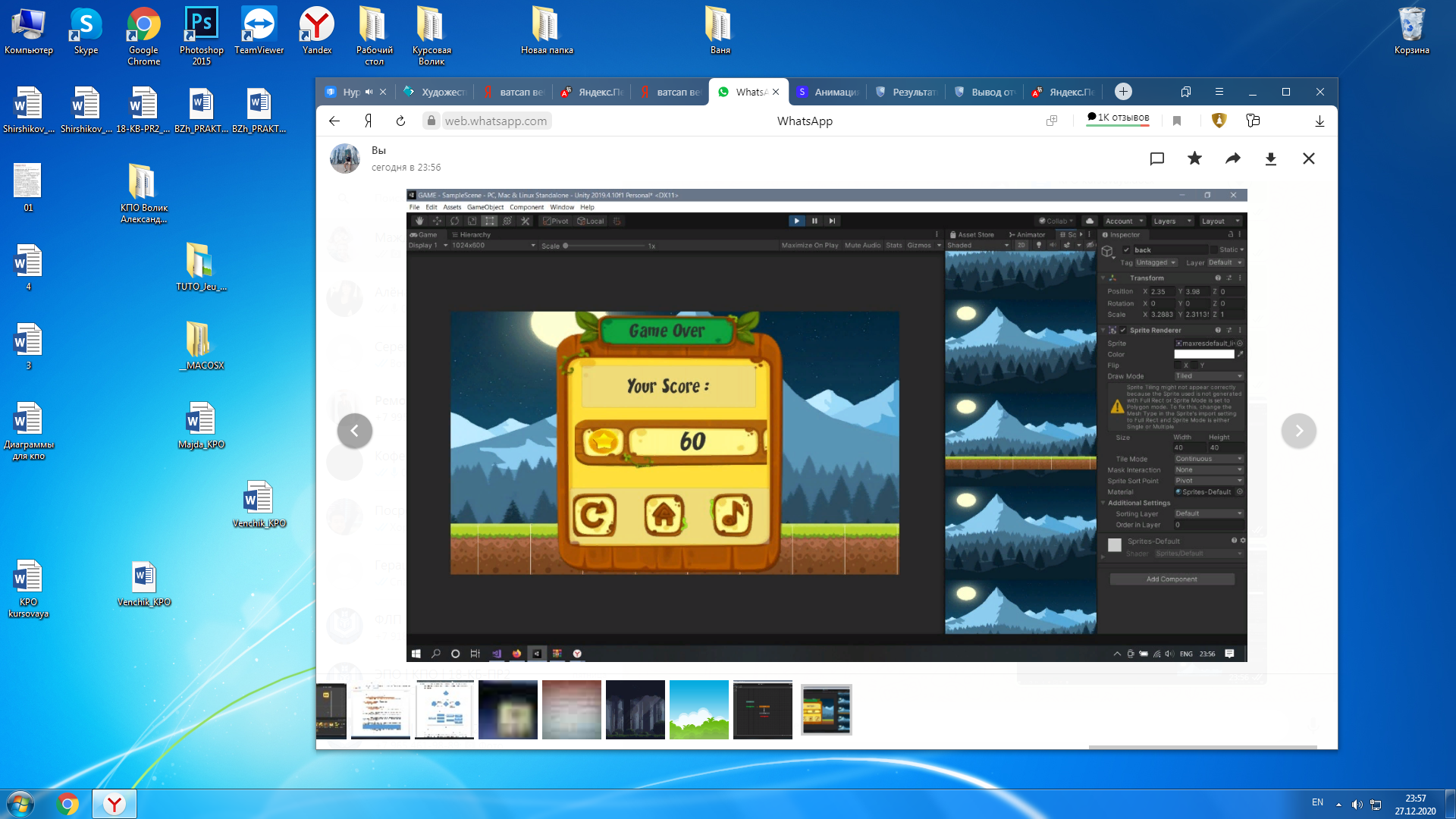


Рисунок 8 – Game over

# **Заключение**

В процессе выполнения курсовой работы были изучены способы разработки игровых приложений в среде разработки компьютерных игр Unity.

При исследовании предметной области была сформулирована идея игрового приложения и выбраны основные задачи персонажа, что способствовало переходу к разработке технического задания программного продукта.

В техническом задании были составлены требования к разработанному программному продукту и техническим средствам, а также сформулированы этапы проектирования и разработки игрового приложения.

На стадии проектирования была построена архитектура игрового проекта и уточнены основные функции персонажа.

При разработке программного продукта был написан программный код для графических элементов.

Все раннее поставленный задачи являются выполненными, следовательно, цель курсовой работы достигнута.

К результатам относится игровой проект «MVRunner».

# **Список используемых источников**

1. Мартин Р. C., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#. : Пер. с англ. – СПб.: Символ-Плюс, 2011. – 768 с.: ил.

2. Фаулер М. UML. Основы, 3-е изд. – М.: Символ-Плюс, 2012. – 192 с.

3. Рудаков А.В, Федорова Г.Н. Технология разработки программных продуктов / Рудаков А.В, Федорова Г.Н. – СПб., 2016. – 220 с.: ил.

4. Художественно-техническое проектирование [Электронный ресурс]

URL:<https://elar.rsvpu.ru/bitstream/123456789/25417/1/RSVPU_2018_397.pdf> (Дата обращения: 19.12.2020)

5. Основные этапы создания видеоигр [Электронный ресурс]

URL: <https://moluch.ru/archive/242/55992/> (Дата обращения: 20.12.2020)

# **Приложение А – Листинг программы**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyBehaviour : MonoBehaviour

{

public GameObject player;

public Animator animator;

private int state = 0;

public GameObject canvas;

public bool isKilling = false;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

transform.position = new Vector3(player.transform.position.x - 4.5f, -1.91f, -1);

if (isKilling)

{

KillPlayer();

}

}

public void GoFurther()

{

state--;

animator.SetInteger("State", state);

}

public void GoCloser()

{

state++;

animator.SetInteger("State", state);

}

void KillPlayer()

{

canvas.GetComponent<CanvasManager>().GameOver();

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CoinsIsAlive : MonoBehaviour

{

public bool IsAlive = true;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

if (!IsAlive)

{

Destroy(gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CoinIsAlive : MonoBehaviour

{

public bool IsAlive = true;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

if (!IsAlive)

{

Destroy(gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class CoinBehaviour : MonoBehaviour

{

public int value = 1;

private GameObject UI;

public Animator animator;

// Start is called before the first frame update

void Start()

{

UI = GameObject.FindGameObjectWithTag("CoinAmount");

}

// Update is called once per frame

void Update()

{

}

void OnTriggerEnter2D (Collider2D col)

{

if (col.gameObject.CompareTag("Player"))

{

int coinUI = int.Parse(UI.GetComponent<Text>().text) + value;

UI.GetComponent<Text>().text = coinUI + "";

animator.SetBool("IsCollect", true);

//Destroy(gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class CoinAmount : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

gameObject.GetComponent<Text>().text = "0";

//gameObject.GetComponent<Text>().text = PlayerPrefs.GetInt("CoinAmount") + "";

}

// Update is called once per frame

void Update()

{

//if (Input.GetKeyDown("a"))

//{

// Debug.Log("Your Coins Are Saved");

// PlayerPrefs.SetInt("CoinAmount", int.Parse(gameObject.GetComponent<Text>().text));

//}

}

public void SaveCoin()

{

int CoinAlreadyCollected = PlayerPrefs.GetInt("CoinAmount");

PlayerPrefs.SetInt("CoinAmount", CoinAlreadyCollected + int.Parse(gameObject.GetComponent<Text>().text));

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MonsterBehaviour : MonoBehaviour

{

public GameObject player;

public GameObject canvas;

public Animator animator;

private int state = 0;

public bool isKilling = false;

//Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

transform.position = new Vector3(player.transform.position.x - 4.5f, -2.6f, -1);

if (isKilling)

{

KillPlayer();

}

}

public void GoFurther()

{

state--;

animator.SetInteger("State", state);

}

public void GoCloser()

{

state++;

animator.SetInteger("State", state);

}

void KillPlayer()

{

canvas.GetComponent<CanvasManager>().GameOver();

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class CoinsBehaviour : MonoBehaviour

{

public int value = 1;

private GameObject UI;

public Animator animator;

// Start is called before the first frame update

void Start()

{

UI = GameObject.FindGameObjectWithTag("CoinAmount");

}

// Update is called once per frame

void Update()

{

}

void OnTriggerEnter2D(Collider2D col)

{

if (col.gameObject.CompareTag("Player"))

{

int coinUI = int.Parse(UI.GetComponent<Text>().text) + value;

UI.GetComponent<Text>().text = coinUI + "";

animator.SetBool("IsCollect", true);

//Destroy(gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraMovements : MonoBehaviour

{

public GameObject character;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

transform.position = new Vector3(character.transform.position.x + 3.5f, 0, -7);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Characbehavior : MonoBehaviour

{

public Rigidbody2D rb;

public float vitesse;

public float maxJump;

private bool isGrounded = false;

public float VulnerabilityTime;

// Start is called before the first frame update

void Start()

{

SetVelocity(vitesse, 0);

//rb.velocity += new Vector2(vitesse, 0);

}

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown("space"))

{

Jump();

}

}

void Jump()

{

rb.velocity += new Vector2(0, maxJump);

}

void OnCollisionEnter2D(Collision2D col)

{

if (col.gameObject.CompareTag("Ground"))

{

isGrounded = true;

}

}

void OnCollisionExit2D(Collision2D col)

{

if (col.gameObject.CompareTag("Ground"))

{

isGrounded = false;

}

}

void SetVelocity(float xVelocity, float yVelocity)

{

rb.velocity = new Vector2(0, 0);

rb.velocity += new Vector2(xVelocity, yVelocity);

}

void OnTriggerEnter2D(Collider2D col)

{

if (col.gameObject.CompareTag("Obstacle"))

{

StartCoroutine(ObstacleFind());

}

}

IEnumerator ObstacleFind()

{

yield return new WaitForSeconds(0.1f);

GameObject.FindWithTag("Monster").GetComponent<EnemyBehaviour>().GoCloser();

SetVelocity(vitesse / 2, 0);

yield return new WaitForSeconds(0.5f);

SetVelocity(vitesse, 0);

yield return new WaitForSeconds(VulnerabilityTime);

GameObject.FindWithTag("Monster").GetComponent<EnemyBehaviour>().GoFurther();

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class CanvasManager : MonoBehaviour

{

public GameObject pauseBtn;

public GameObject pausePanel;

public GameObject gameOverPanel;

public GameObject coinPanelTxt;

public GameObject gameOverCoinTxt;

private bool isPause = false;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

}

public void GameOver()

{

Time.timeScale = 0;

coinPanelTxt.GetComponent<CoinAmount>().SaveCoin();

gameOverCoinTxt.GetComponent<Text>().text = PlayerPrefs.GetInt("CoinAmount") + "";

gameOverPanel.SetActive(true);

//gameOverCoinTxt.GetComponent<Text>().text = PlayerPrefs.GetInt("CoinAmount") + "";

pauseBtn.SetActive(false);

}

public void PausePlay()

{

if (isPause)

{

isPause = false;

Time.timeScale = 1;

pauseBtn.SetActive(true);

pausePanel.SetActive(false);

}

else

{

isPause = true;

Time.timeScale = 0;

pauseBtn.SetActive(false);

pausePanel.SetActive(true);

}

}

public void Restart()

{

Time.timeScale = 1;

SceneManager.LoadScene("SampleScene");

}

}

# **Приложение Б – Проверка на антиплагиат**

