**Зміст**

[Вступ 3](#_Toc532229363)

[Створення ігрового оточення 4](#_Toc532229364)

[Створення гравця 4](#_Toc532229365)

[Створення основного шляху 8](#_Toc532229366)

[Створення платформ 9](#_Toc532229367)

[Створення анімацій платформ 9](#_Toc532229368)

[Налаштування головної камери 12](#_Toc532229369)

[Збереження на рівні 14](#_Toc532229370)

[Лічильник життів на рівні 20](#_Toc532229371)

[Завершення рівня 22](#_Toc532229372)

[Огляд повного рівня 23](#_Toc532229373)

[Головне меню 23](#_Toc532229374)

[Збірка гри 29](#_Toc532229375)

[Висновок 30](#_Toc532229376)

[Список використаних джерел 31](#_Toc532229377)

[Додатки 32](#_Toc532229378)

[Додаток А 32](#_Toc532229379)

[Додаток Б 33](#_Toc532229380)

[Додаток В 33](#_Toc532229381)

[Додаток Г 34](#_Toc532229382)

[Додаток Д 35](#_Toc532229383)

[Додаток Е 36](#_Toc532229384)

[Додаток Є 37](#_Toc532229385)

[Додаток Ж 37](#_Toc532229386)

# **Вступ**

Ігри супроводжують людей із прадавніх часів, коли в такий спосіб пращури здобували нові навички виживання у більш-менш безпечній, а, найголовніше, цікавій формі. На щастя, часи змінилися і потреба у виживанні не така гостра. Проте, бажання вчити щось нове, досягати, перемагати не тільки не зникло, а й перетворилось у новий жанр мистецтва. Вам цей жанр добре знайомий – це відеоігри.

Сучасні відеоігри сміливо можна назвати найбільш потужним жанром мистецтва : суміш кінематографу, музики, літератури, театру, живопису створюють унікальні спогади та досвід. Проте іграм властива одна відмінність, що робить їх унікальними – взаємодія з гравцем, що має повну владу над діями на ігровій сцені. Але кому буде цікаво мати усю владу над грою? Це не створює перепон та не допомагає гравцю розвиватися через персонажа.

З цією думкою було обрано тему курсової роботи – створити гру за допомогою Unity 3D, потужного інструменту для створення дво- та тривимірних ігор. Окрім цього, така робота не тільки цікава, але й потребує часу і змушує постійно знаходити нову інформацію та вчити кілька дисциплін одночасно. Такі умови відмінно підходять для вдосконалення навичок програмування та всебічного розвитку (дизайн, анімація, логіка, математика тощо).

За мету було обрано створення тривимірної гри-платформера. Така гра передбачає переміщення гравця з точки А до точки Б у тривимірному просторі та наявність перепон, що створені ландшафтом рівня.

Гра мала б бути побудована за двома принципами. Перший – “easy to learn – hard to master” (з англійської – “легко вивчити – віжко оволодіти”). Другий – принцип мінімалізму, за яким формами можна знехтувати і залишити лише гру. Загальні вимоги такі : гра має бути цікава у вивченні, спонукала вивчати ігрові механіки, виглядала сучасно та не спонукала вийти з неї після першої сутички зі складністю.

# **Створення ігрового оточення**

## **Створення гравця**

Оскільки головною ідеєю дизайну гри був мінімалізм, найкращим вибором форми гравця стала куля : вона проста, вона виділяється на фоні ландшафту рівня та повністю підходить під модель гри.

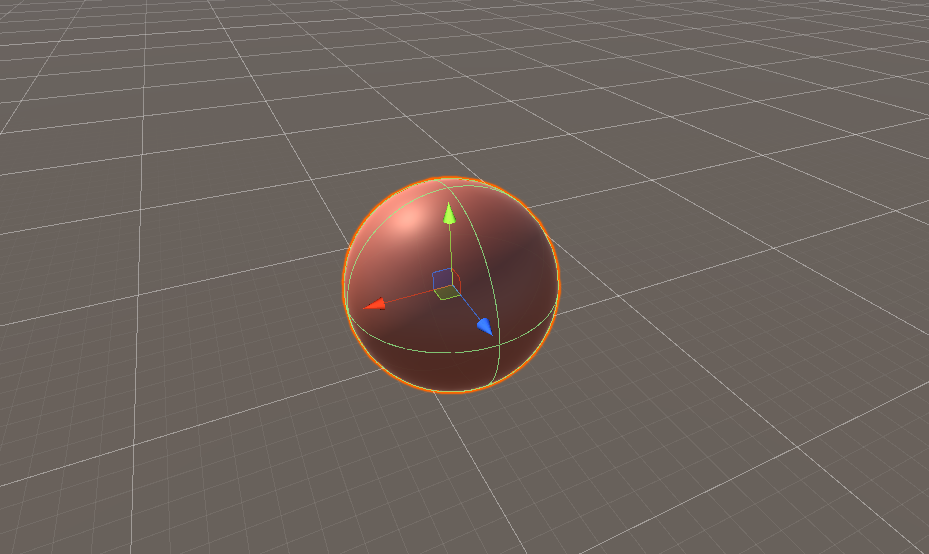


Рис. 1

Початковий колір мав би виділятися як на загальному фоні (фоні оточення), так і на фоні кольорів елементів рівня (платформ, схилів). Тому, спочатку були обрані основні кольори для фону, основного шляху переміщення та колір гравця. Такий порядок дозволяє максимально точно підібрати кольори.

Безпосередньо для гравця був обраний відтінок коричневого кольору. Для цього треба створити новий матеріал та прив’язати його до моделі нашого гравця. Також були додані властивості відображення та відсвічування, що відповідають пунктам Reflections та Specular Highlights відповідно; змінені параметри гладкості для надання ефекту матовості. Усі налаштування відображаються у окремому вікні на панелі Inspector.

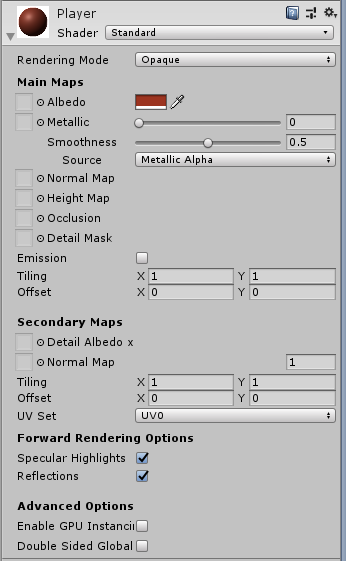


Рис.2

Для того, щоб гравець зміг вільно переміщуватися по рівню, йому була додана фізична модель Rigidbody, що включає у себе гравітацію та тертя. Для активації режиму гравітації потрібно увімкнути Use Gravity на панелі властивостей Rigidbody.

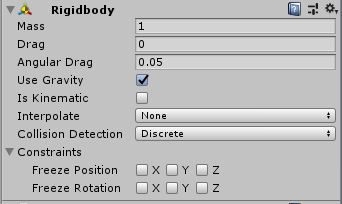


Рис.3

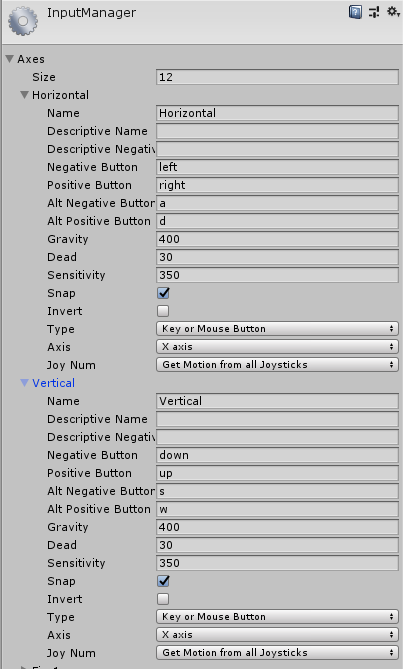
Для управління були обрані клавіші w,a,s,d. Налаштування клавіш виконується за допомогою інструменту Inspector меню Edit. Реалізація виконується за допомогою скрипта мовою C#. Скрипт створюється як окремий компонент (такий, як матеріал або анімація) та прив’язується до об’єкту. В даному випадку – до моделі гравця.

Рис.4



Рис.5

Для цього потрібно створити відкриту змінну типу Rigidbody та прив’язати до неї необхідний об’єкт (гравця). Для того, щоб гравець зрушив з місця, потрібно додавати прискорення за певною віссю тривимірних координат об’єкту гравця при натисканні визначеної клавіші переміщення. Аби гра була плавною, обрахування треба виконувати кожен кадр. Про реалізацію переміщення гравця докладніше у додатку А.

## **Створення основного шляху**

За задумом, рівень мав складатися з простих геометричних форм, тому основний шлях було створено із кубів товщини 1 та різної довжини і куту нахилу, що додало багаторівневості. Колір та фактуру додано тим самим способом, що і гравцю. Якщо обрати шлях як Is Triggered, гравець буде проходити крізь підлогу. Жодних скриптів для шляху не потрібно, як і фізична модель. Поверхня тверда за змовчуванням, тому Rigidbody потрібне тільки для гравця. Приклад відрізку шляху наведений нижче.

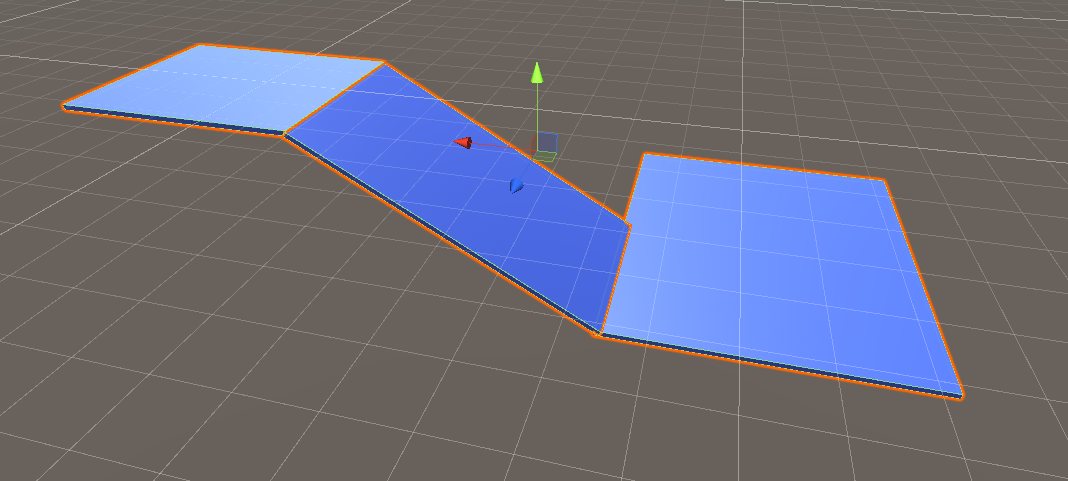


Рис.6

## **Створення платформ**

Для додання більшого різноманіття у переміщенні та багаторівневості були додані платформи. Платформі також не потрібна фізична модель. До платформ були застосовані анімації переміщення у просторі : горизонтально та вертикально.

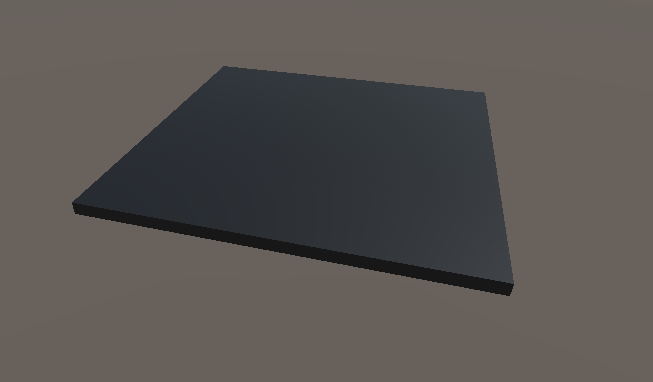


Рис.7

## **Створення анімацій платформ**

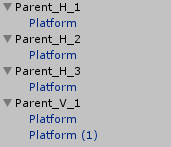


Рис.8

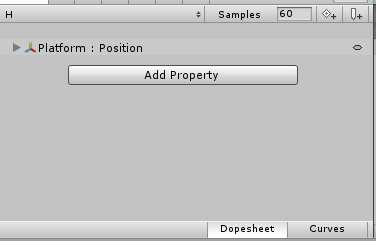


Рис.9

Існує 2 способи застосування анімації до об’єкта : до об’єкта безпосередньо або ж до батьківського об’єкта. Оскільки було вирішено створити декілька платформ, що мали б рухатися урізнобіч, незалежно одна від одної, анімація була прив’язана до пустого батьківського об’єкту типу Game Object.

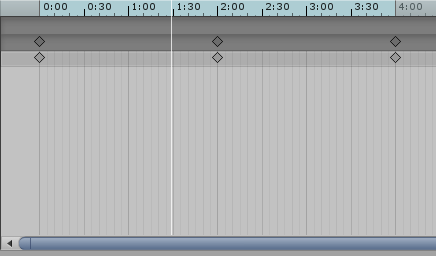


Рис.10

Анімацію було створено за допомогою влаштованого редактору Unity. На рис.8 показано ім’я платформи, прив’язаної до батьківського об’єкта, та можливість вибрати властивість, яка буде змінена при анімуванні. На рис.9 показано шкалу кадрів, протягом яких буде відбуватися анімація. Що більші проміжки між змінами об’єкту, то більш плавною вийде анімація. В даному прикладі показані інтервали при анімуванні горизонтального переміщення платформи.

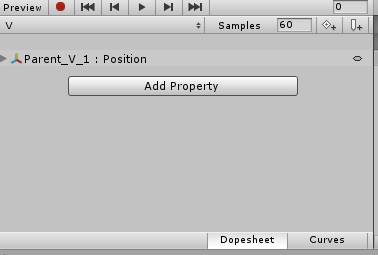


Рис.11

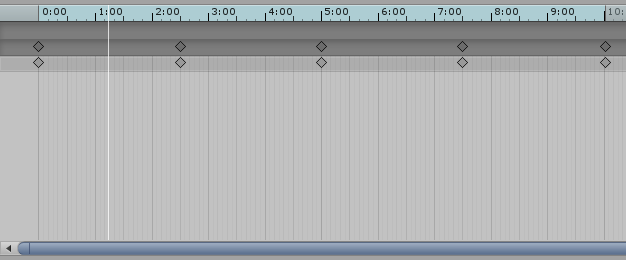


Рис.12

Аналогічно для вертикальної анімації.

## **Налаштування головної камери**

Камера є одним із базових інструментів маніпулювання увагою гравця та інструментом, від якого залежить сприйняття гри. Було вирішено змінити її положення перед гравцем для надання ефекту спостерігача, якому дозволили керувати процесом. До того ж, це не дозволяє побачити увесь рівень одразу, що додає як складності, так і елементу непередбачуваності.

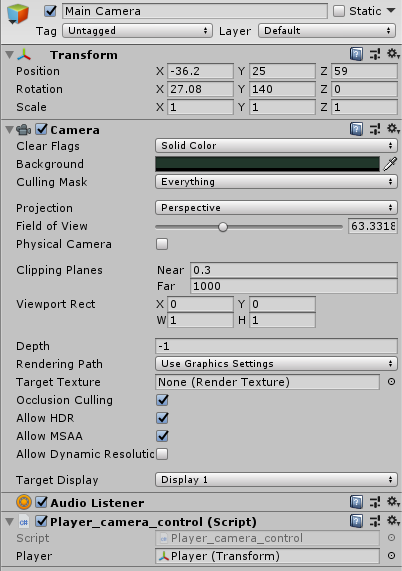


Рис.13

Камера вважається окремим ігровим об’єктом, тому їй також властиві переміщення, зміна розміру та кут нахилу і повороту. Окрім цього, саме у камері налаштовується колір або текстура неба (оточення поза рівнем). Можна вибрати як чистий колір, так і градієнт або власне зображення. Для відповідності кольоровому стилю гри було обрано заливку чистим кольором. За замовчуванням, камера статична. Тому потрібен скрипт для закріплення камери за гравцем та слідування за ним кожен кадр. Докладніше про скрипт – у додатку Б.

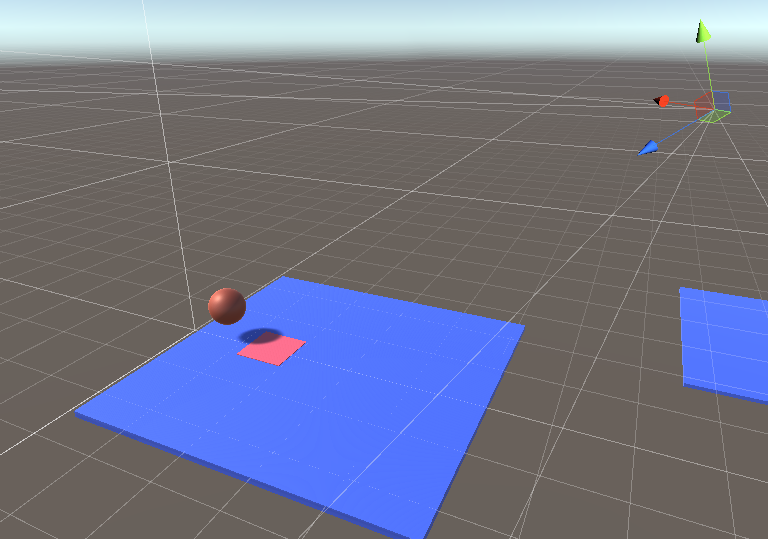


Рис.14

Камера у редакторі.

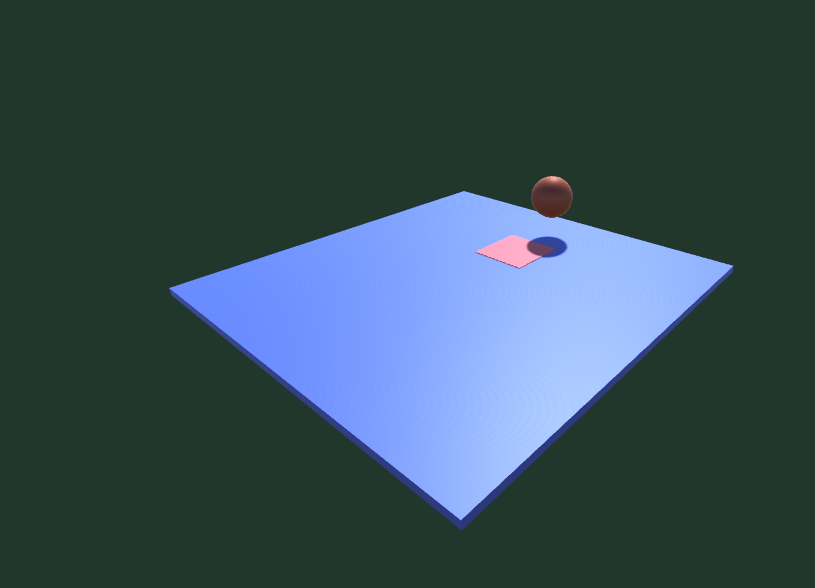


Рис.15

Камера у режимі гри.

## **Збереження на рівні**

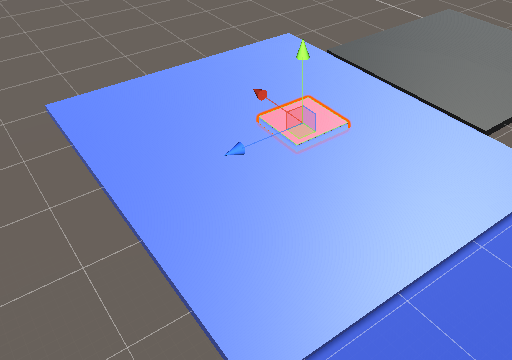


Рис.16

Збереження потрібні, аби розпочати гру з визначеного місця, якщо гравець випав з рівня без можливості самостійно продовжити ігровий процес. У такому випадку кажуть, що гравець «помер».

Система створена таким чином, що коли гравець контактує з «мертвою зоною» (вилітає надто далеко за межі рівня), знищується об’єкт гравця та з визначеного місця завантажується його копія.

Докладніше це виглядає так.

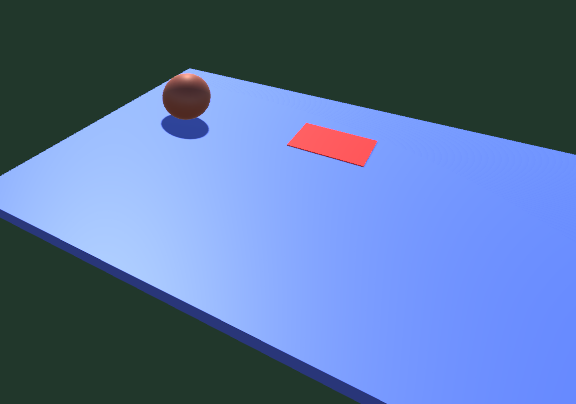


Рис.17

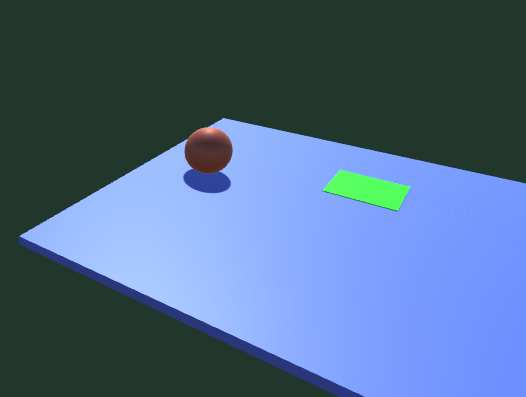


Рис.18

Створюється Game Object, який є точкою збереження гравця. Оскільки він невидимий, його було поміщено (зроблено батьком) звичайного куба, підсвіченого червоним кольором. Це означає, що чекпоінт неактивний у даний момент. Зелений – активний. Скрипт зміни кольору прив’язаний до платформи та детально описаний у додатку Д.

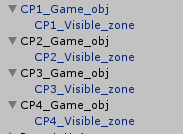


Рис.19

Оскільки на рівні існує 4 чекпоінти, було створено 4 великі бокс-коллайдери, при контакті з якими реєструється смерть гравця (зелені паралелепіпеди). Їх треба розташувати на певній відстані один від одного, аби фізична модель Unity змогла правильно зрозуміти, з яким саме коллайдером зіштовхнувся гравець (рис.20).

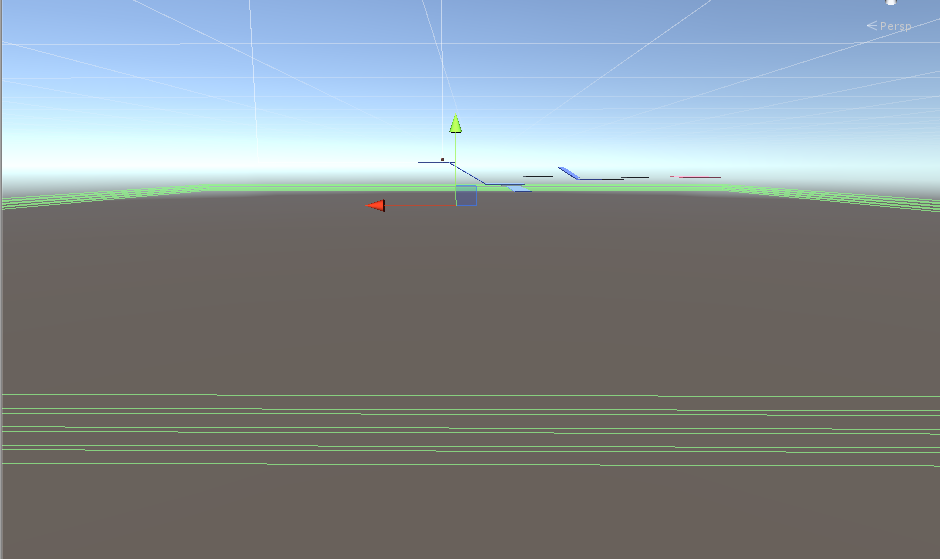


Рис.20

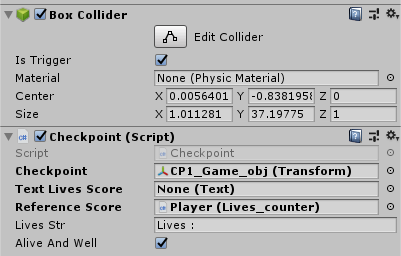


Рис.21

Але якщо ми хочемо мати кілька чекпоінтів, то необхідно знищувати попереднє збереження, якщо активоване нове. Тому біля кожного чекпоінту був створений окремий бокс-коллайдер, при взаємодії гравця з яким попередній чекпоінт стає неактивним, а відповідна велика зона видаляється (рис.22).

Таким чином, завжди можна розпочати з початку рівня, якщо померти, не дійшовши до наступного збереження, та розпочати з точки останнього збереження в усіх інших випадках.

Для коректного розпізнавання гравця використовується пошук об’єкта за тегом. Гравцю був присвоєний тег “Player”, тому при знищенні можна бути впевненим, що видалиться саме гравець.

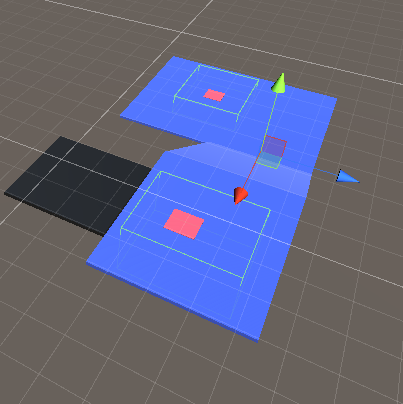


Рис.22

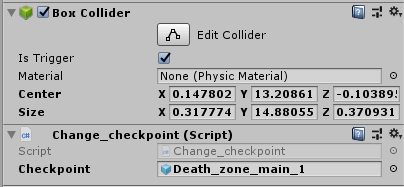


Рис.23

Велика зона приймає пустий Game Object збереження у якості параметра. Деталі – у додатку В. Малі зони приймають великі зони у якості параметра. Деталі – у додатку Г.

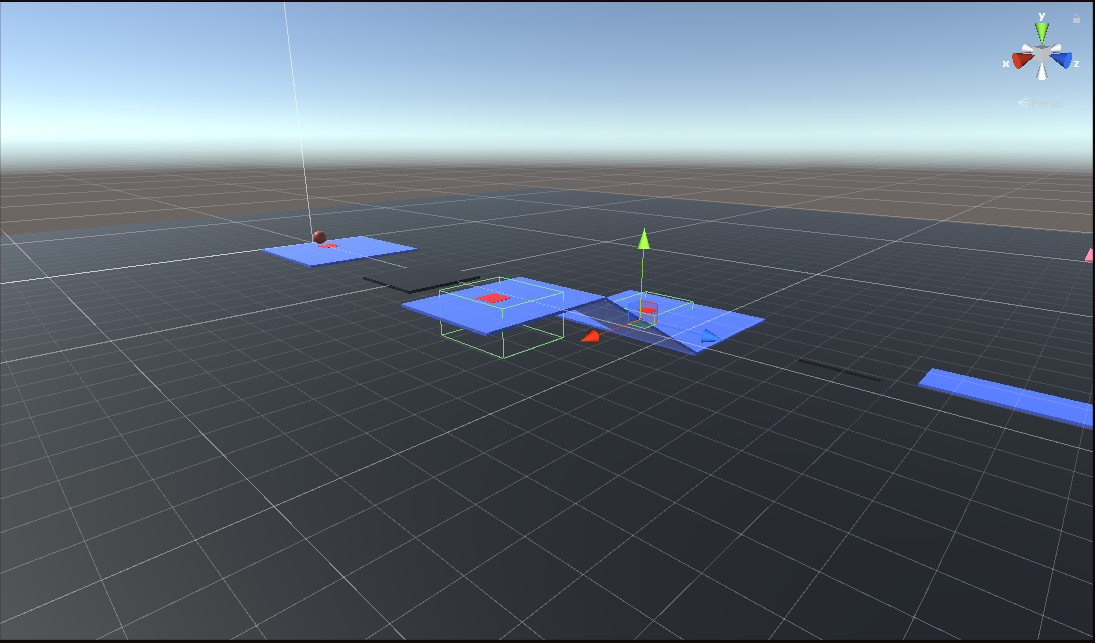


Рис.24

Початок гри.

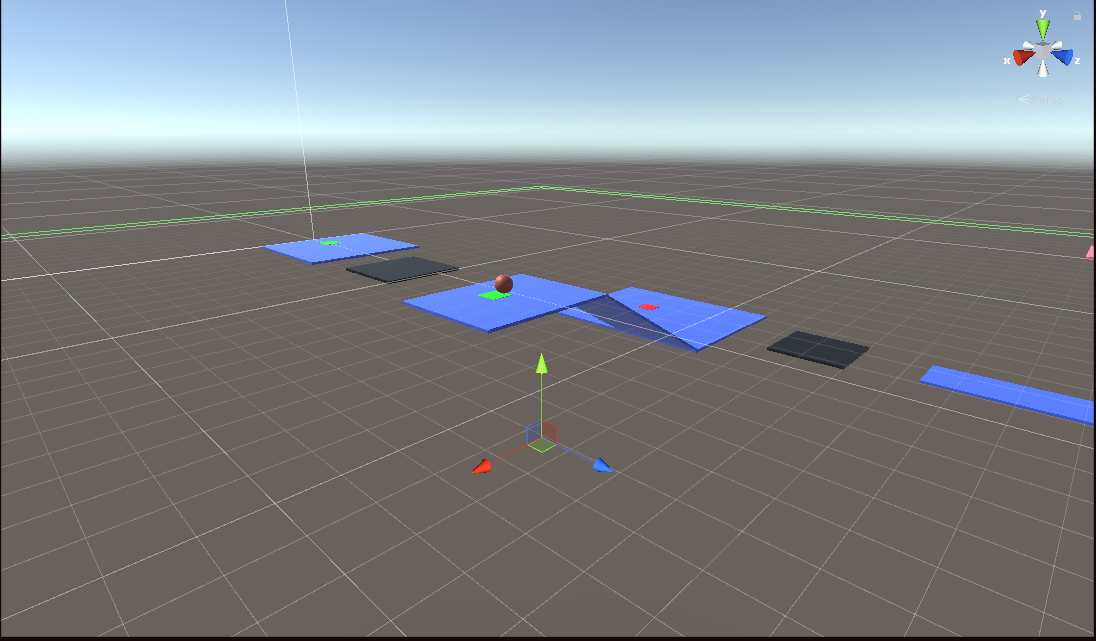


Рис.25

Знищено першу велику зону при активації другого чекпоінту.

## **Лічильник життів на рівні**

При взаємодії із точкою збереження гравцю нараховується одне додаткове життя – спроба розпочати ще один раз із останньої точки збереження.

Лічильник розпочинається з нуля. Для візуального відображення зарахування використовується зміна кольорів чекпоінту (із червоного на зелений, рис.17-18) та текстове відображення у правому верхньому кутку ігрового екрану (рис.24).

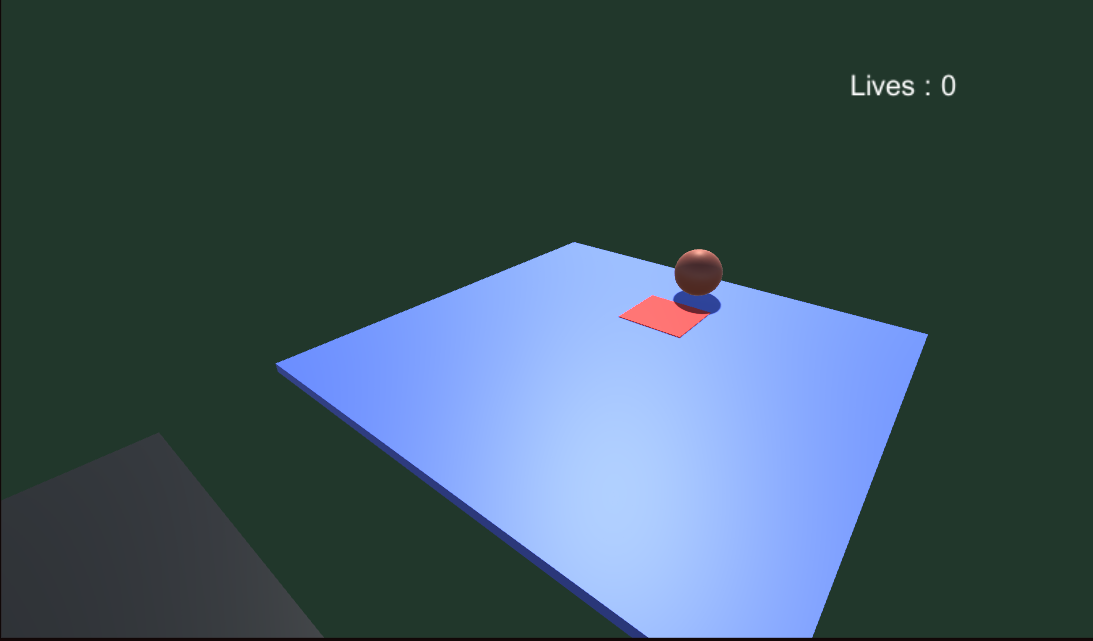


Рис.24

Якщо гравець має на рахунку нуль життів (остання спроба) та помирає – він розпочинає увесь рівень із самого початку.

Текстове відображення створюється за допомогою об’єкту-полотна (canvas), що прив’язується до об’єкту гравця та великої зони збереження за допомогою скрипта (додатки Е та В відповідно).

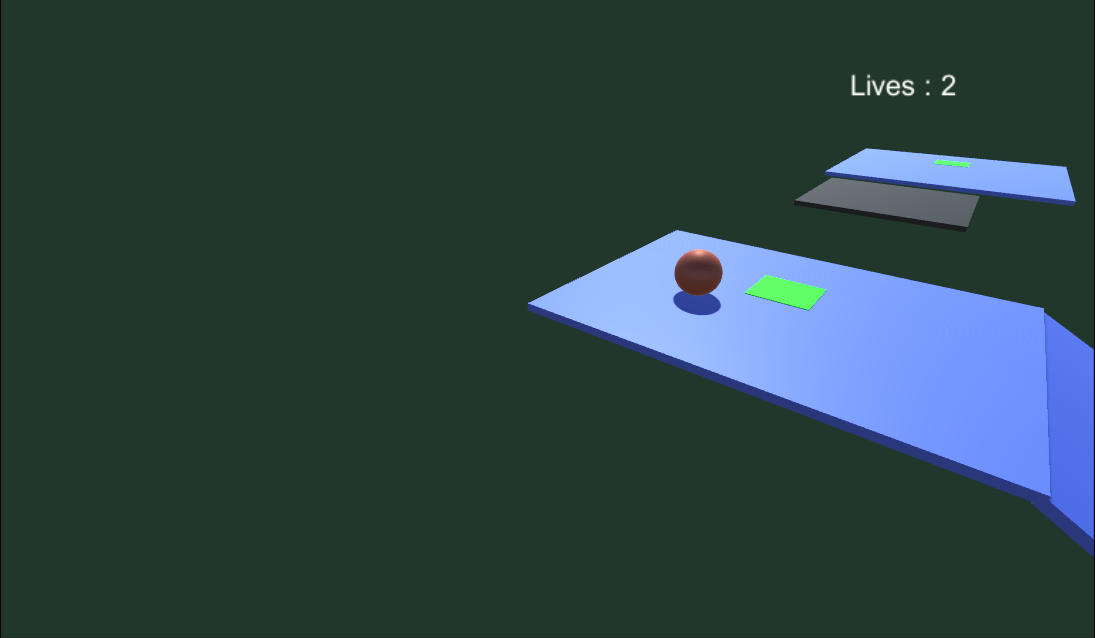


Рис.25

## **Завершення рівня**

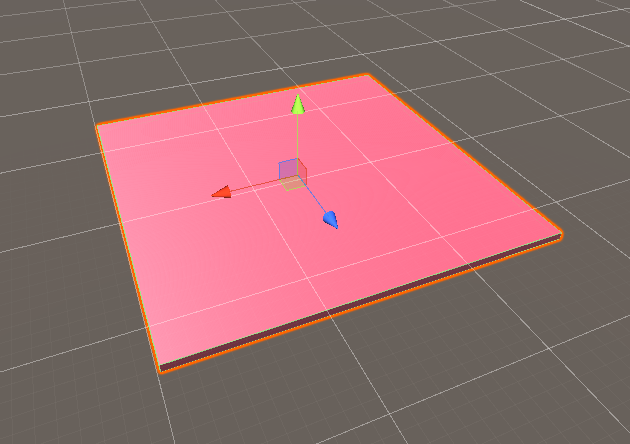


Рис.25

Завершення рівня робиться для переходу на наступний рівень або для виходу у головне меню гри. Алгоритм нескладний : якщо коллайдер гравця (знаходимо його за тегом Player) контактує з коллайдером об’єкту, що завершує рівень, тоді завантажується обрана сцена (рівень, меню тощо).

У розглянутому випадку перехід здійснюється одразу у головне меню.

У Unity кожен рівень або меню – це окрема сцена, що має свій номер та/або унікальне ім’я. Тому у скрипті лічильник зчитує номер поточної сцени та робить необхідну ітерацію. Докладніше – у додатку Є.

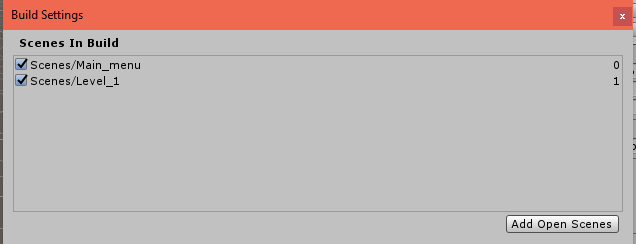


Рис.26

## **Огляд повного рівня**

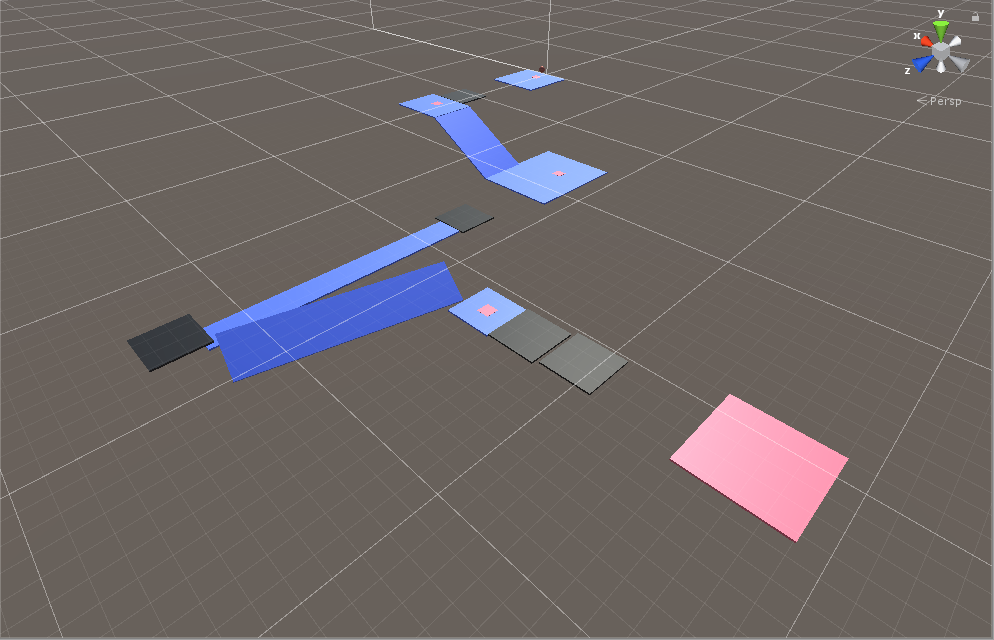


Рис.27

## **Головне меню**

Головне меню, як вже було зазначено раніше, являє собою окрему сцену. У цій сцені було створено текстове поле, у якому були розташовані під-поля та кнопки, накладені на текст.

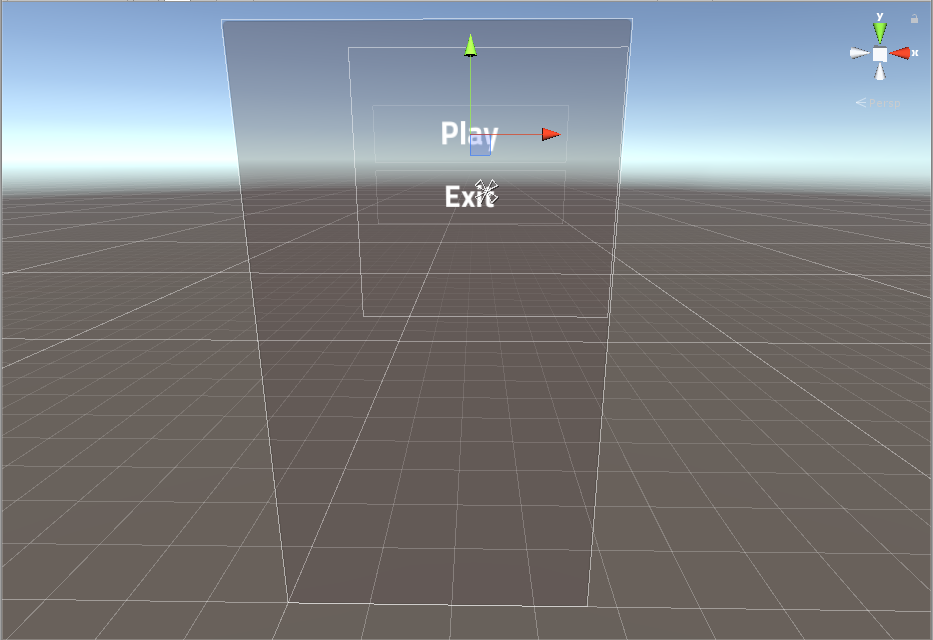


Рис.28

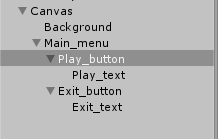


Рис.29

Загальні налаштування для панелі меню виглядають наступним чином.

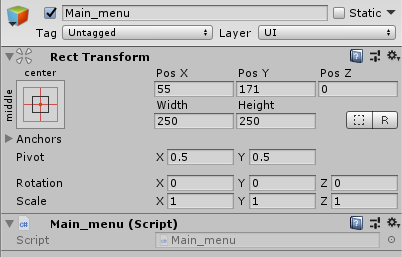


Рис.30

Налаштування для кнопок (загальні для всіх, обрано кнопку Play).

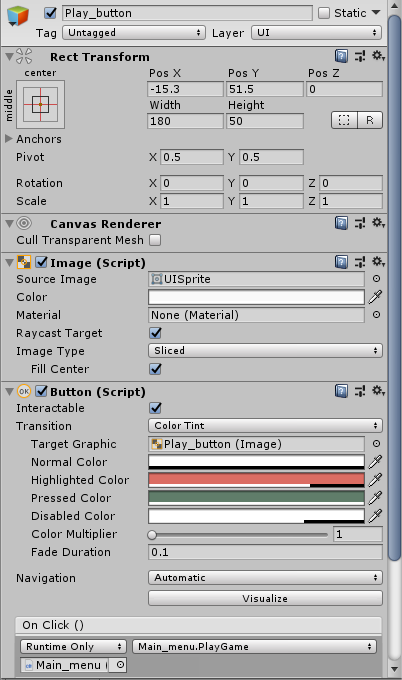


Рис.31

Налаштування для тексту в меню.

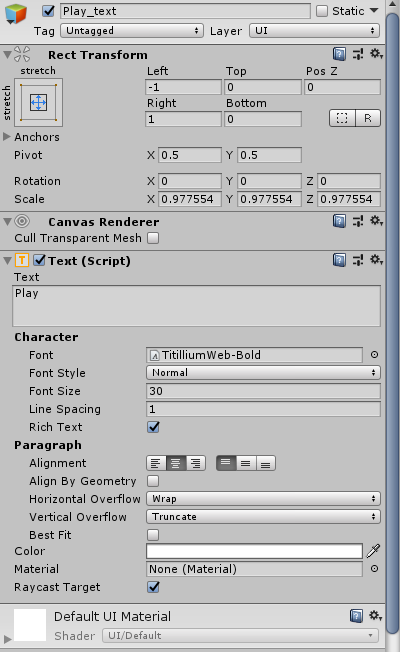


Рис.32

Загальний вигляд меню.



Рис.33

Клавіші підсвічуються червоним при наводженні на них.



Рис.34

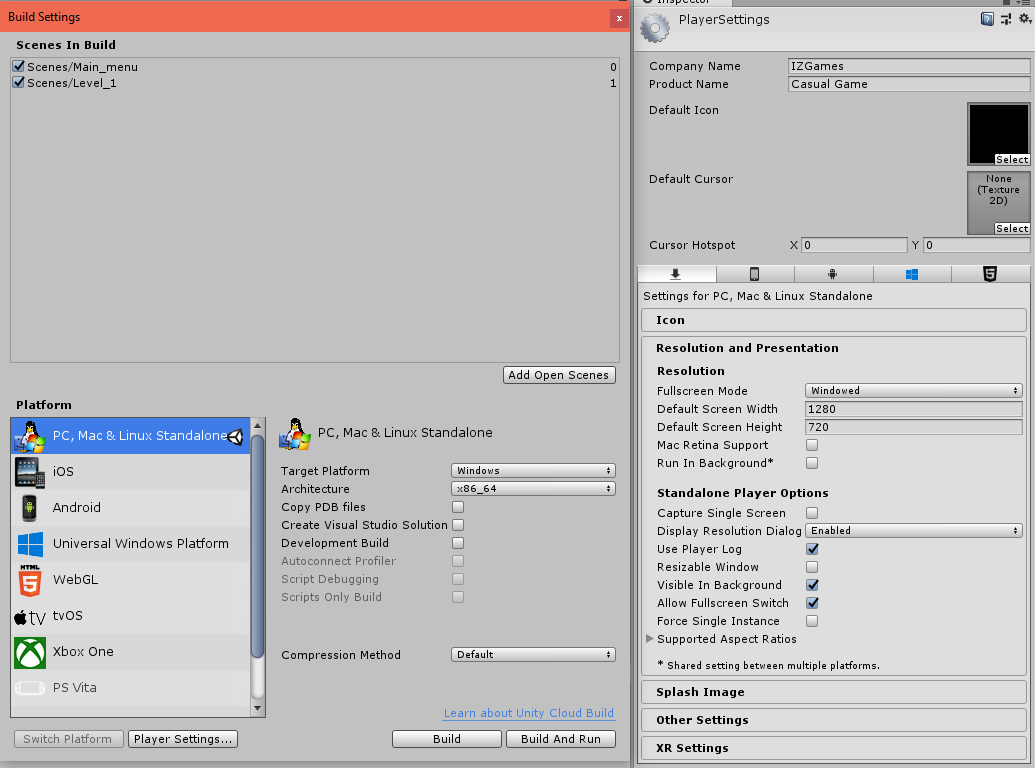
Клавіші підсвічуються зеленим при натисканні.



Рис.35

Реалізація завантаження рівня та вихід з гри детальніше описані у додатку Ж. Також слід відзначити, що у гравця є можливість вийти у головне меню під час гри при натисканні клавіші Esc. Це можливе завдяки тим самим властивостям, що і завантаження рівня, описане у додатку Ж. Реалізація цього доповнення описана у додатку А

## **Збірка гри**



Збірка гри робиться за допомогою меню Build settings панелі File у Unity. У налаштуваннях можна вибрати платформи, на яких буде запускатися проект, налаштування графіки тощо. При натисненні кнопки Build потрібно буде обрати папку для збірки, обрати іконку запуску, натиснути ОК та почекати, доки створиться білд.

# **Висновок**

Під час створення гри були здобуті різноманітні навички : проектування рівнів, анімації, роботи із редактором Unity, але, найголовніше, - навички розробки програмного забезпечення із використанням принципів ООП.

Гра повністю відповідає закладеним у неї принципам. Під час розробки були пройдені усі етапи створення гри : концепт, моделювання, створення ігрової логіки та механіки, тестування, створення початку та завершення, збірка гри.

Хоча у грі є лише один рівень, її концепт дозволяє створювати багато різноманітних та, водночас, складних рівнів. Це ще раз підтверджує той факт, що гра повністю відповідає задуму - “легко вивчити – віжко оволодіти”.

Вивчення середовища розробки відбувалося з нуля, проте чим більше матеріалу було знайдено – тим більше розуміння було отримано. Протягом розробки було зроблено багато помилок новачка, хоча деякі випадки були абсолютно унікальні і не існувало чіткої відповіді на поставлене питання. Це створювало досить екстремальний, та разом із тим, цікавий процес дослідження світу розробки ігор.

Але основна робота була виконана за допомогою скриптів мовою C#, вдосконалення навичок володіння якою і було поставлене за мету створення курсової роботи.

Детальніше про використані ресурси під час розробки – у списку використаних джерел.

# **Список використаних джерел**

1. Cal Schrotenboer, Daniel Solis. Illustrated C# 7. 5-th Edition – Apress, 2018 – 817c.
2. Andrew Troelsen, Philip Japikse. C# 6.0 and the .NET 4.6 Framework, 7-th Edition **–** Apress, 2015 – 1832c.
3. Steven Goodwin. Polished Game Development – Apress, 2016 – 273c.
4. Greg Lukosek. Learning C# by Developing Games with Unity 5.x, 2-nd Edition – PACKT, 2016 – 230c.
5. Micael DaGraca. Practical Game AI Programming – PACKT, 2017 – 342c.
6. Інтернет-ресурс : <https://www.youtube.com/user/Brackeys/featured>
7. Інтернет-ресурс : <https://www.youtube.com/playlist?list=PLPV2KyIb3jR53Jce9hP7G5xC4O9AgnOuL>
8. Інтернет-ресурс : <https://www.youtube.com/watch?v=zc8ac_qUXQY>
9. Інтернет-ресурс : <https://www.youtube.com/watch?v=4eO-6Vgiqyo>
10. Інтернет-ресурс : <https://docs.unity3d.com/2018.3/Documentation/Manual/index.html>
11. Інтернет-ресурс : <https://docs.unity3d.com/2018.3/Documentation/ScriptReference/index.html>
12. Інтернет-ресурс : <https://answers.unity.com/questions/1243589/argumentexception-input-button-right-is-not-setup.html>
13. Інтернет-ресурс : <https://www.youtube.com/watch?v=4OvTLXA1Ark>

# **Додатки**

## **Додаток А**

**Player’s movement**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Player\_movement : MonoBehaviour {

public Rigidbody sandbox\_prop\_1;

public float addSomeForce = 1500f;

void FixedUpdate ()

{

if (Input.GetKey("w"))

{

sandbox\_prop\_1.AddForce(addSomeForce \* Time.deltaTime, 0, -addSomeForce \* Time.deltaTime);

}

else if (Input.GetKey("s"))

{

sandbox\_prop\_1.AddForce(-addSomeForce \* Time.deltaTime, 0, addSomeForce \* Time.deltaTime);

}

else if (Input.GetKey("a"))

{

sandbox\_prop\_1.AddForce(addSomeForce \* Time.deltaTime, 0, addSomeForce \* Time.deltaTime);

}

else if (Input.GetKey("d"))

{

sandbox\_prop\_1.AddForce(-addSomeForce \* Time.deltaTime, 0, -addSomeForce \* Time.deltaTime);

}

else if(Input.GetKey(KeyCode.Escape))

{

SceneManager.LoadScene(0);

}

}

}

## **Додаток Б**

**Player’s camera control**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Player\_camera\_control : MonoBehaviour

{

public Transform player;

private Vector3 offset = new Vector3(-25f, 25f, 60f);

void LateUpdate ()

{

transform.position = player.position + offset;

}

}

## **Додаток В**

**Checkpoints**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class Checkpoint : MonoBehaviour {

public Transform checkpoint;

GameObject player;

public Text textLivesScore;

public Lives\_counter referenceScore;

public string livesStr = "Lives : ";

public bool aliveAndWell = true;

void Start()

{

player = GameObject.FindWithTag("Player");

referenceScore.GetComponent<Lives\_counter>();

}

private void OnTriggerEnter(Collider other)

{

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

{

player.transform.position = checkpoint.position;

player.transform.rotation = checkpoint.rotation;

aliveAndWell = false;

if (aliveAndWell == false)

{

referenceScore.livesScore--;

if (referenceScore.livesScore < 0)

{

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

}

}

}

}

}

## **Додаток Г**

**Change checkpoints**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Change\_checkpoint : MonoBehaviour {

public GameObject checkpoint;

private void OnTriggerEnter(Collider other)

{

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

{

Destroy(checkpoint);

Destroy(gameObject);

}

}

}

## **Додаток Д**

**Change checkpoints’ colors**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Change\_color : MonoBehaviour {

GameObject player;

Renderer rend;

void Start()

{

rend = GetComponent<Renderer>();

player = GameObject.FindWithTag("Player");

if (gameObject.tag == "Player")

{

rend.material.color = Color.green;

}

else

{

rend.material.color = Color.red;

}

}

private void OnTriggerEnter(Collider other)

{

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

{

rend.material.color = Color.green;

}

}

}

## **Додаток Е**

**Lives counter**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Lives\_counter : MonoBehaviour

{

public GameObject player;

GameObject checkpoint;

public Text textLivesScore;

public int livesScore = 0;

public string livesStr = "Lives : ";

private void OnTriggerEnter(Collider other)

{

if(other.gameObject.tag == "Respawn")

{

livesScore++;

other.gameObject.tag = "Checkpoint\_checked";

}

}

public void FixedUpdate()

{

textLivesScore.text = livesStr + livesScore.ToString();

}

}

## **Додаток Є**

**Level finish**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Finish\_level : MonoBehaviour

{

private void OnTriggerEnter(Collider other)

{

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

{

SceneManager.LoadScene(0);

}

}

}

## **Додаток Ж**

**Main menu**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Main\_menu : MonoBehaviour

{

public void PlayGame()

{

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

}

public void QuitGame()

{

Application.Quit();

}

}