МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-08

Дипломний проєкт

здобувача освіти денної форми навчання
КГ.08.09.000.ДП

ЗАЛЕСЬКОГО
КИРИЛА ВЯЧЕСЛАВОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Комп'ютерна графіка і Web-дизайн»**

Група: **4КГ-08**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity

Проєктний матеріал складається з пояснювальної записки на 84 сторінках та графічного (презентаційного) матеріалу на 16 аркушах (слайдах)

Дипломник _____ (Залеський К.В.)

Керівник _____ (Джабраілов Д.В.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «21» червня 2025 р. Протокол ЕК № 2

Оцінка ЕК 5 (відмінно) / 90 S.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна графіка і Web-дизайн»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Залеському Кирилу Вячеславовичу*

(прізвище, ім'я, по батькові)

1. Тема проєкту *Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity*

затверджена наказом по коледжу від "14" листопада 2024р. № 246

2. Термін здачі закінченого проєкту

3. Вихідні дані до проєкту *Використання програмного рушія Unity; Використання мови програмування C# та основних бібліотек рушія Unity; Використання фізичного ядра рушія Unity; Реалізація основних ігрових елементів рівнів; Реалізація основних механік управління; Реалізація фізики поверхонь та гравця; Реалізація інтерфейсу; Реалізація ігрового меню*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз існуючих ігрових рішень та обрання ігрового жанру; Вибір програмного забезпечення для розробки гри; Проектування основних ігрових елементів; Проектування роботи фізики рушія у ігровому процесі; Реалізація основних ігрових елементів; Реалізація використання фізичного ядра програмного рушія Unity; Реалізація ігрового меню; Тестування гри

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Аналіз існуючих ігрових рішень та обрання жанру; Обрані засоби розробки; Структура основних ігрових елементів; Реалізація основних ігрових елементів; Використання фізики рушія в ігровому процесі; Реалізація використання фізики рушія; Реалізація ігрового інтерфейсу та меню гри; Тестування гри та скріншоти проекту

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Джабраілов Д.В.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 02.06.2025

Керівник *Джабраілов Д.В.* _____
(підпис)

Завдання прийняв до виконання *Залеський К.В.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	*14.05.2025*	виконав
2	*Аналіз існуючих ігрових рішень та обрання жанру*	*16.05.2025*	виконав
3	*Вибір програмних засобів розробки*	*17.05.2025*	виконав
4	*Проектування основних ігрових механік*	*20.05.2025*	виконав
5	*Проектування використання фізики рушія*	*22.05.2025*	виконав
6	*Реалізація основних механік гри*	*28.05.2025*	виконав
7	*Реалізація взаємодії з фізикою рушія*	*01.06.2025*	виконав
8	*Реалізація інтерфейсу гри та ігрового меню*	*03.06.2025*	виконав
9	*Відлагодження елементів гри*	*06.06.2025*	виконав
10	*Тестування працездатності елементів гри*	*10.06.2025*	виконав
11	*Виправлення виявлених помилок*	*11.06.2025*	виконав
12	*Аналіз результатів, підготовка слайдів презентації*	*12.06.2025*	виконав
13	*Економічні розрахунки та питання з охорони праці*	*13.06.2025*	виконав
14	*Підготовка графічної частини проекту*	*14.06.2025*	виконав
15	*Підготовка проекту до захисту та тестування гри*	*16.06.2025*	виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 08. 09 000. 00 ДП	Дипломний проєкт		
А4			КГ 08. 09 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КГ 08. 09 000. 00 ДП		
Розробив		Залеський К.В.		16.02	Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity	Літ.	Аркуш 4 / Аркушів 84
Перевірив		Джабраілов Д.В.		16.02		Н Д П	
Н. контр.		Петрашова В.І.		16.06.25		ВСП "ОТФК ОНТУ" гр.4КГ-08	
Затверд.		Кривченко Ю.В.		16.06.25			

ЗМІСТ

ВСТУП

Створення комп'ютерних ігор стало важливою частиною IT-ринку, який продовжує свій бурхливий розвиток. Кількість розробляємих проектів збільшується створюючи конкуренцію на ринку ігрових проектів. Залучаючи все більш складні та високотехнологічні засоби розробки, ігрова індустрія сама собою рухає вперед ці технології утворюючи запит на них.

Разом з тим ігрова індустрія все більш чітко поділяється на види ігрових проектів за кількістю залучених фінансів, технологій та спеціалістів. Від багатомільйонних проектів з великою кількістю розробників, до розробників-ентузіастів, які самотужки створюють хіти ігрової індустрії.

Для того щоби мати актуальні знання та навички у сфері розробки ігор та бути конкурентоспроможнім, потрібно практикувати та використовувати сучасні засоби розробки, використовуючи основні етапи розробки, залучати нові прийоми вирішення тих чи інших проблем у розробці.

Темою мого дипломного проекту є «Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity», яка полягаю у створені гри з досить простим, але при тому цікавим ігровим процесом, виконанні основних етапів розробки ігрового проекту, від створення концепції ігрового процесу, до реалізації ігрових механік. Ігрові правила основані на механіках ігор платформерів. Окрім того до ігрового процесу будуть залучені механізми фізики ігрового програмного рушія Unity, що дають змогу більш тісно познайомитись із цим інструментом, та розвинути власні нщавички у роботі з ним. На меті реалізувати ігровий процес вільного переміщення по рівню, використання фізичних поверхонь, використання інтегрованого у ігровий процес ігрового інтерфейсу.

Ігровий програмний рушій Unity вимагає використовувати сучасні засоби програмування на мові C#, використовувати графічні редактори, та інші технології вміння використовувати які є важливою складовою професіональних навичок будь-якого працівника в IT-сфері, а реалізація власного невеликого проекту є цінним досвідом.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз існуючих ігрових рішень та обрання ігрового жанру

1.1.1 Огляд існуючих ігрових рішень

Розробка будь-якого ігрового проекту починається з обрання ігрового жанру. Цей вибір безпосередньо впливає на ігрові механіки, способи ігрового контролю, графічні особливості, особливості огляду рівнів та переміщання по ним, тактики гравця та реакції гри на них. Ігрові жанри можна порівняти з жанрами у кінематографі, або книгах, де вони визначають те, як буде розвиватись сюжет, та які сцени можна використовувати у творі, а які будуть зайвими.

Вказаний вище зв’язок є зворотно-пропорційним, оскільки саме те, як розробник бачить основний ігровий процес, так і потрібно визначати жанр ігрового проекту. Раніше в іграх жанр визначався односкладово, оскільки технології тих часів не дозволяли реалізувати велику кількість ігрових механік. В наш час ігрові жанри можуть поєднуватись та створювати більш складні та цікаві поєднання ігрових механік.

Оскільки основною ідеєю розробляємої гри є переміщення шарика та проходження рівнів з використанням фізичних можливостей ігрового рушія Unity, слід зосередитись на таких проектах, які мають схожий ігровий процес та в результаті такого підбору інформації виконати аналіз з якого отримати вихідні дані для розробки власного проекту.

Одним із таких проектів є гра Balance 2004 року від компанії Cyparade, а видавником вийшла компанія Atari. Її основою ігрового процесу було проходження рівнів-головоломок, складність яких підвищувалась з рівнем. Грати необхідно було шаром, який рухався по рівню та гравець мав балансувати на гойдалках, мостах та вузьких тропках для того щоби попасти на фінальну частину рівня. Ще одною особливістю гри була можливість змінювати матеріал шару, яким керував гравець, через що змінювались ігровий процес завдяки взаємодії фізики шару та оточення. На рисунку 1.1 можна побачити скріншот гри Ballance.

Рисунок 1.1. Скріншот ігрового процесу комп’ютерної гри Ballance

Ця гра реалізована у жанрі платформер із дещо зміненими принципами ігрових механік. В грі немає звичним ворогів, замість них використовуються механічні та фізичні головоломки.

Другий розглянутий проект це гра Vertigo 2008 року від компанії Nintendo яка й виступила видавником. Початково розроблялась для ігрової приставки Wii. Ця гра ігровий процес схожий на попередній, але важливою частиною було те, що необхідно було проходити рівні чим швидше, тим краще. Рівні представляли собою «траси», що потрібно було «прокачуватись» на швидкості. На відміну від попередньої гри, змінювати матеріал шару неможна було, але можна було використовувати зароблені медалі для розвитку властивостей шару, таких як маневреності, швидкості або можливості тормозіння у повітрі. Ці властивості в деяких рівнях відігравали важливу роль, тому їх налаштування було важливою частиною ігрового процесу. Сама гра використовувала особливості ігрових

контролерів для приставки Wii, тому мала більшу популярність саме на ринку Nintendo. На рисунку 1.2 можна побачити скріншот гри Vertigo.

Рисунок 1.2. Скріншот ігрового процесу комп’ютерної гри Vertigo

Ця гра також реалізована в жанрі платформеру, але в ній дещо особливим чином розміщена камера, для більш чуткого контролю за шаром і можливості більш корректно виконувати управління ним.

Також було розглянуто серію проектів Rock of Ages від розробника Ace Team та видавника AtlusUSA. Ця гра перш за все виділяється своїм зовнішнім стилем та основною сюжетною ідеєю. По суті гра є сюжетною, але ігровий процес рівнів складається з того, що потрібно котитись на камінні від старту до фінішу при цьому каміння має параметр міцності і через контакт з оточуючими предметами, в залежності від їх матеріалу, каміння втрачає свою міцність. В кінці кожного рівня потрібно вибити ворота замку та перемогти свого опонента-комп’ютера. На шляху гравця встають різні припони від стилізованих солдат, до

захисних споруд, стін, та елементів природи. В цілому гра має мінімалістичний інтерфейс, але із однієї частини в іншу, ігрові механіки змінювались та додавались нові ідеї такі як, боси, зміна типів каміння, створення власних рівнів та багато інших нововведень, які урізноманітнювали ігровий процес. Рисунок 1.3 зображує скріншот ігрового процесу гри Rock of Ages.

Рисунок 1.3. Скріншот ігрового процесу комп'ютерної гри Rock of Ages III

Ця гра також виконана у жанрі платформер, і на відміну від попередніх проектів як головну характерну рису своєї гри розробники обрали стилістику гри, виконавши її у кумедному стилі історичних гравюр та картин.

1.1.2 Вибір ігрового жанру

Виходячи з розглянутих ігрових проектів, основним ігровим елементом яких був шар, або сфера, якою керував гравець, можна зробити висновок, що домінуючим ігровим жанром для цього типу ігор є платформер.

Платформер обирається через свою направленість у головному на переміщення по рівням, пошук виходів, вирішення задач та головоломок які основані на рівнях, або ігрових механіках. В основному, в таких іграх потрібно грати якимось персонажем, що має більш гуманоїдну будову тіла. Не рідно такі проекти реалізовуються із ворогами, або декількома особливими ігровими

механіками. Для платформерів також характерні проекти, в яких гравець не має ворогів на рівнях, але повинен проходити гру з використанням ігрових механік, чи уникаючи складних ситуацій на рівнях – шипів, прірв або інших небезпечних явищ. Кожний ігровий жанр має свої визначні риси, тому, для реалізації проекту потрібно їх виділити:

- Управління персонажем: гравцю дають змогу керувати персонажем, який може переміщуватись по рівню не тільки в горизонтальній, але й вертикальній площині. Окрім того, як елементи для переміщення можуть виступати різноманітні елементи оточення, як драбини, мотузки, листя та інше.
- Платформи: в таких іграх рівні будують за допомогою «платформ» вони частіше за все стилізовані під оточення, та мають різні форми та розміри. Саме через взаємодію із платформами жанр отримав свою назву.
- Перешкоди і вороги: на шляху гравця часто зустрічаються різні перешкоди, від природніх статичних та динамічних, до ворогів. Такі перешкоди та ворогів частіше можна уникати або знищувати.
- Збір предметів: не рідко в іграх цього жанру розробники реалізують механізми збору предметів, які можуть відкривати частини рівнів, або бути обов’язковими для подальшого переміщення по грі.

Також, як будь-який інший ігровий жанр, платформери мають свої технічні аспекти, які наведені нижче:

- Фізика: Важливою частиною платформерів є реалістичне чи стилізоване моделювання фізики, особливо гравітації та інерції, що впливає на рух та стрибки персонажа.
- Графіка і арт-стиль: Жанр може варіюватися від простої піксельної графіки до складної тривимірної візуалізації, і часто має яскравий арт-стиль, що запам’ятовується.

Отже, виходячи з аналогічних проектів, а також особливостей жанру, для розробки 3D-гри 3DBallz, обрано ігровий жанр платформер, як такий, що відповідає цілям, які поставлені перед проектом.

1.2 Вибір програмного забезпечення для розробки гри

1.2.1 Ігровий програмний рушій

Темою дипломного проекту зазначено використання фізичного ядра програмного рушія Unity, то вибір останнього зумовлений темою розробки. Ігровий програмний рушій Unity є сучасним інструментом розробки ігрових проектів, який дає змогу використовувати всі сучасні ігрові технології. Гнучкі інструменти розробки дають змогу реалізовувати різноманітні варіанти ігор, від простих 2D-ігор, до складних 3D-ігор з використанням сучасної графіки, анімації та ігрових механік.

Система монетизації ігрового програмного рушія Unity є умовно безкоштовною. Будь-який користувач може завантажити та вільно використовувати ігровий програмний рушій для створення своїх проектів та не платити за використання ПЗ, але якщо річний дохід зареєстрованої компанії буде досягати визначеної умовами використання суми, потрібно буде обов’язково оформляти підписку на Pro версію ПЗ.

1.2.2 Середа розробки коду

Створення гри складається з багатьох складових, в тому числі використання мов програмування для написання коду гри. Мова програмування обрана виходячи з ігрового програмного рушія Unity, а саме мова C#, на якій і функціонує рушій. Для створення скриптів гри можна використовувати будь-які середи розробки, які підтримують мову програмування C# та її форматування.

Для розробки свого проекту було обрано середу розробки Microsoft Visual Studio, яка є досить потужною та зручною середою розробки. Завдяки вмонтованому стилізатору коду, інтелектуальній допомозі та зручним підказкам, можна досить легко створювати ефективний код.

Microsoft Visual Studio є безкоштовною середою розробки та вільно розповсюджується на офіційному сайті цього пакету розробки. Є платна версія, втім, для розробки проекту для мого дипломного проекту достатньо безкоштовної навчальної ліцензії.

1.2.3 Графічний редактор

У контексті розробки поточного проекту було вирішено використовувати Adobe Photoshop. Вибір зумовлений декількома факторами, головним з яких є досвід його використання. Також, Adobe Photoshop є потужним графічним редактором, для якого досить легко знайти відео-інструкції по роботі з ним, якщо виникнуть складнощі. Окрім того для розробки даного проекту не потрібно буде створювати складні рисунки або графіку, тому достатньо буде пробної версії графічного редактора для створення всіх необхідних спрайтів, текстур гри та роботи з пропорціями та розмірами зображень.

1.3 Проектування основних ігрових елементів

1.3.1 Загальна концепція проекту

Перш за все потрібно визначитись із тим, як має виглядати ігровий проект в кінці поточної ітерації розробки. Це важливо зробити з точки зору коректного проектування модулів гри, схем взаємодії та побудови зв’язків між ігровими об’єктами, ігровими механіками, скриптами та компонентами.

Отже, вже визначено, що гра має бути реалізованою в жанрі платформер, що дає стійке розуміння того, як будуть побудовані рівні. Вони будуть складатись з платформ різних форм та розмірів, або кутів нахилу. Окрім того, мають бути поверхні по яким гравець не може переміщуватись, так звані «прірви». Потрібно також реалізувати роботи із фізикою ігрового рушія Unity. Ця робота буде базуватись на властивостях об’єкту гравця, а також поверхонь по яким гравець зможе переміщуватись. Для більш глибокої реалізації використання 3D напревленості проекту та гнучкого ігрового процесу, гравець повинен мати можливість не тільки рухатись у горизонтальній площині, але й у вертикальній за допомогою стрибків.

Для більшого розуміння того, на що буде походити ігровий процес, можна звернутись до класики ігор такого типу – Marble Madness, яка існувала ще у часи ігрових приставок Sega MegaDrive. На рисунку 1.4 можна побачити скріншот ігрового процесу гри Marble Madness. Так само як і у розробляємому проекті, там

можна було переміщуватись по рівню за допомогою сфери, рівень був сформований з платформ та ландшафту різного нахилу. Гравець мав переміщуватись по ігровому рівню та проходити до кінця, збираючи бонуси та знищуючи ворогів. Проходження рівня повинно було вкладатись у визначений час.

Рисунок 1.4. Скріншот ігрового процесу комп’ютерної гри Rock of Ages III

Для мого проекту будуть декілька інші вимоги. Гравець не буде обмежений у часі, а також у кількості сфер для проходження рівня. Натомість, при втраті сфери, гравець буде повертатись на початок рівня, йому не буде доступна карта, а можливість стрибків обмежена кількістю енергії у сфери та силою стрибка, що залежить від рівня складності.

Управління ігровим процесом буде реалізовуватись за допомогою клавіатури та кнопок управління рухом сфери вліво-вправо та вперед-назад. Окремо клавіша, яка відповідає за стрибки.

Інтерфейс гри буде інтегрованим у ігровий процес. Оскільки концепцією передбачено лише один ігровий ресурс, енергія, то його відображення буде реалізовано зміною кольору сфери гравця, чим сфера буде чорнішою, тим меншим є запас енергії і навпаки, чим красніше сфера, тим більше енергії у розпорядженні гравця.

1.3.2 Проектування основних ігрових елементів

Для подальшої роботи, потрібно спроектувати основні елементи гри. Виходячи з теми дипломного проекту, а також створеної концепції, можна схематично зобразити структуру проекту, як зображено на рисунку 1.5. Розглянемо цю схему більше детально.

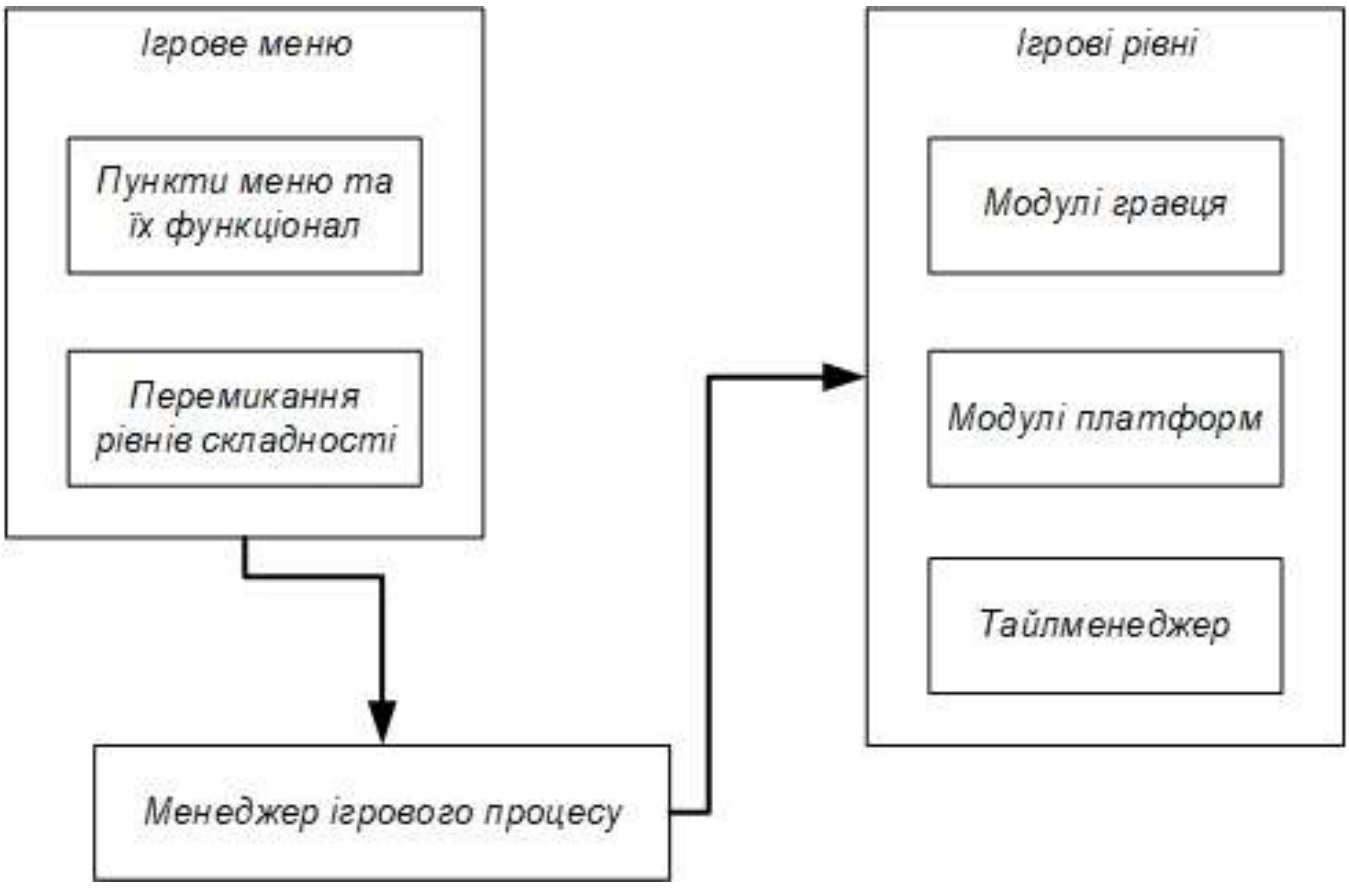


Рисунок 1.5. Схематичне зображення структури гри 3DBallz

Гра складається з трьох основних блоків. Два з цих блоків є послідовними, а третій існує поза черговості роботи цих блоків. Ігрове меню є початковим елементом гри, яке в першу чергу бачить гравець. Переглянути схему роботи ігрового меню можна на рисунку 1.5. Розглянемо його детальніше. Меню складається з чотирьох пунктів, кожний з яких виконує свою окрему функцію.

Пункт «Нова гра» повинен розпочинати ігровий процес. Особливістю роботи цього пункту буде полягати у тому, що він завантажуватиме перший рівень гри. При цьому завантаження параметрів рівня складності виконується при створенні ігрової сцени першого рівня. За замовченням гра буде розпочинатись із нормальним рівнем складності.

Пункт «Налаштування» буде давати змогу обрати рівень складності для гри, при натисканні на цю кнопку, вона ховає основне меню та відображає меню обрання складності. У меню доступним є три кнопки рівнів складності – легкий, нормальний та важкий. Кожний з цих пунктів впливає на те які максимальні параметри будуть доступні для гравця. Вибір рівня складності при натисканні на кнопку, змінюють значення параметрів у Менеджері ігрового процесу.

Пункт «Правила гри» повинен виконувати функціонал ознайомлення гравців із правилами. При натисканні на цю кнопку, основне меню буде ховатись, та почне відображатись меню з описанням ігрового процесу та основних ігрових механік. З цього меню можна повернутись у головне меню.

Пункт «Вихід з гри» має виконувати простий функціонал завершення роботи додатку та повернення гравця у операційну систему.

Також важливо спроектувати структуру модулів рівнів майбутньої гри. Це необхідно зробити для того, щоби створити основу для працюючого механізму ігрового процесу. Також такий підхід дасть продумати будову модулів рівнів таким чином, щоби у майбутньому було легше вносити якісь зміни, або покращення ігрового процесу. Структуру модулів ігрових рівнів можна побачити на рисунку 1.7. Важливо виділити, що рівні у спрощеному вигляді, побудовані з модулів, які забезпечують роботу об’єктів пов’язаних з гравцем, модулів які забезпечують роботу платформ а також кодів для тайлменеджера.

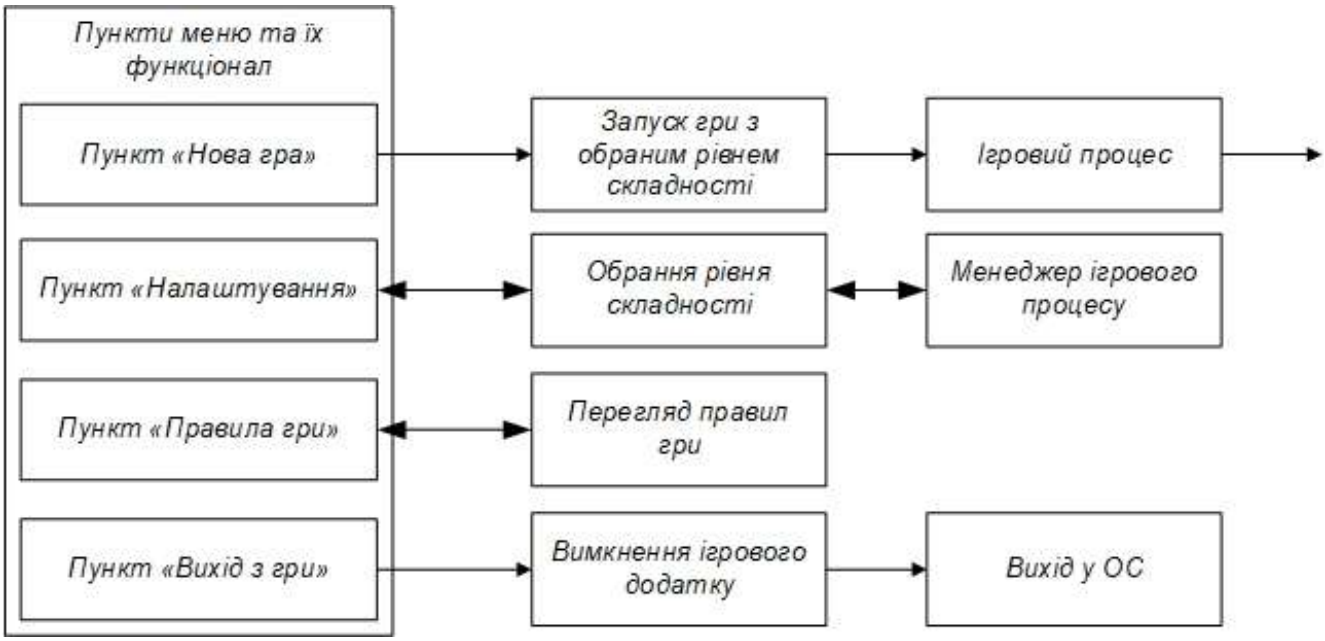


Рисунок 1.6. Схема роботи ігрового меню гри 3DBallz

Модуль гравця складається з Коду параметрів гравця, який зберігатиме параметри сфери гравця, його швидкість, силу стрибка, а також допоміжні методи, які дадуть змогу виконувати дії пов'язані з енергією гравця та інше. Також цей модуль вміщує в собі Код руху гравця, які забезпечує переміщення гравця, стрибків а також всі команди вводу з клавіатури. Нарешті останньою складовою цього модуля є Код камери, який керує роботою камери, її кутом а також разворотом відносно гравця.

Модуль платформ вмішуватиме в собі коди, які будуть реалізовувати механізми роботи із активними платформами гри. Такими які забезпечують вихід з рівня, які збільшують, або зменшують кількість енергії у гравця.

Тайлменеджер складається з коду тайлів води, які мають знищувати сферу гравця, коли та потрапляє у воду.

Залишилось розглянути Менеджер ігрового процесу, який зображений на рисунку 1.5. Цей елемент грає зв'язуючу роль між головним меню та ігровими рівнями. Через особливості роботи ігрового рушія Unity потрібно передбачити механізм збереження та передачі даних від однієї сцени до іншої.

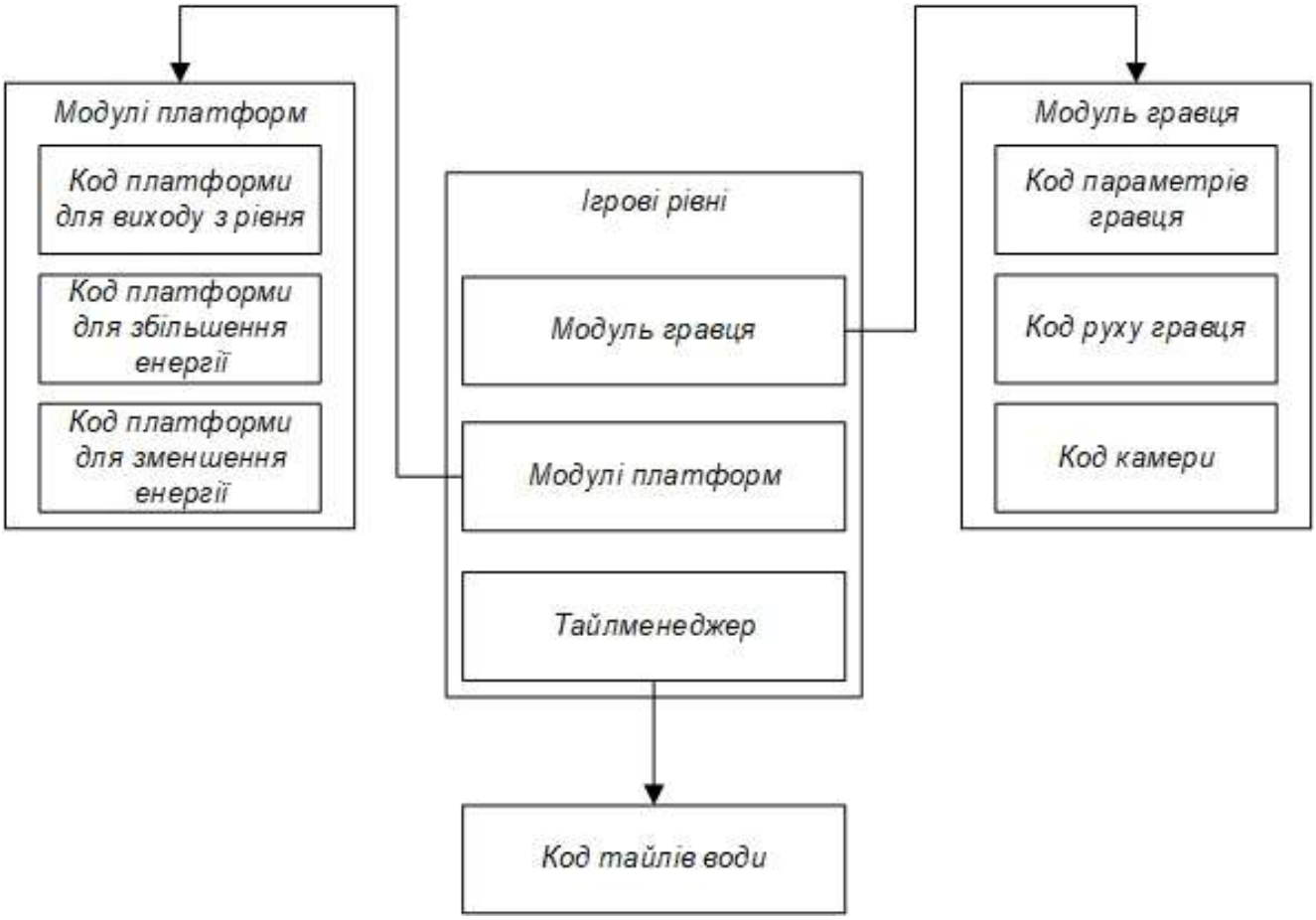


Рисунок 1.7. Структура модулів ігрових рівнів гри 3DBallz

Саме це є головною задачею цього менеджеру. Потрібно реалізувати системи збереження інформації між рівнями. Серед такої інформації може бути дані про рівень складності, або напряму нові параметри гравця, що встановлюються рівнем складності.

1.4 Проектування роботи фізики рушія у ігровому процесі

1.4.1 Загальний аналіз питання

Фізична складова ігрового рушія Unity, відома як Unity Physics, є ключовим елементом для створення реалістичної інтерактивності в іграх. Вона базується на фізичному движку PhysX від компанії NVIDIA, який забезпечує симуляцію різноманітних фізичних явищ, таких як гравітація, зіткнення, тертя, та еластичність. Всі розрахунки Unity виконує автоматично, без необхідності розробника втручатись у цей процес. Фізика у Unity існує лише для ігрових об’єктів з компонентом RigidBody. Цей компонент використовує параметри сцени, а також властивості ігрового об’єкту до якого доданий для симуляції

фізики. Якщо від ігрового об’єкта потребують додаткових функцій, окрім впливу гравітації, до нього додають додаткові компоненти та описують ці функції за допомогою коду.

Також у симуляції фізики ігрових об’єктів використовується компонент Collider. Цей компонент відіграє роль фізичного кордону у оболонці ігрових об’єктів. Якщо до об’єкту додано тверде тіло, але відсутній колайдер, то симуляція фізики буде відбуватись, але ігрові об’єкти почнуть проходити один крізь другий. Тому використання колайдерів обов’язково для більш коректної реалізації фізики у ігровому процесі.

Крім того, для симуляції деяких властивостей поверхонь потрібно використовувати фізичні матеріали. Це ассети, які можна створити у редакторі Unity та додати до проекту. Фізичні матеріали надають змогу налаштовувати те, яким є матеріал об’єкту до якого він доданий. Наприклад, можна зробити поверхню зовсім не стрибучою, або навпаки посилити стрибучість, симулюючи поверхню батуту. Фізичні матеріали можна використовувати лише додавши їх до колайдерів, тому використання останніх обов’язкове.

Ще одною складовою є Physics Engine Settings – налаштування, які дозволяють контролювати загальну поведінку фізичної системи, включаючи гравітацію, час симуляції, та інші параметри.

Unity також надає інструменти для оптимізації фізичних розрахунків, такі як Layer Collision Matrix, яка дозволяє визначити, які шари (layers) можуть взаємодіяти між собою, та Physics Raycaster, який використовується для виявлення об’єктів у сцені за допомогою променів.

У розробляємій грі потрібно використовувати основний функціонал фізики ігрового рушія Unity, а саме використання гравітації, тертя о поверхні, параметри стрибучості, використання фізики поверхонь.

1.4.2 Проектування роботи фізики у грі

Перед створенням проекту, потрібно продумати використання ігрової фізики ядра Unity. На рисунку 1.8 можна побачити основні фізичні елементи, які будуть задіяні у проекті.

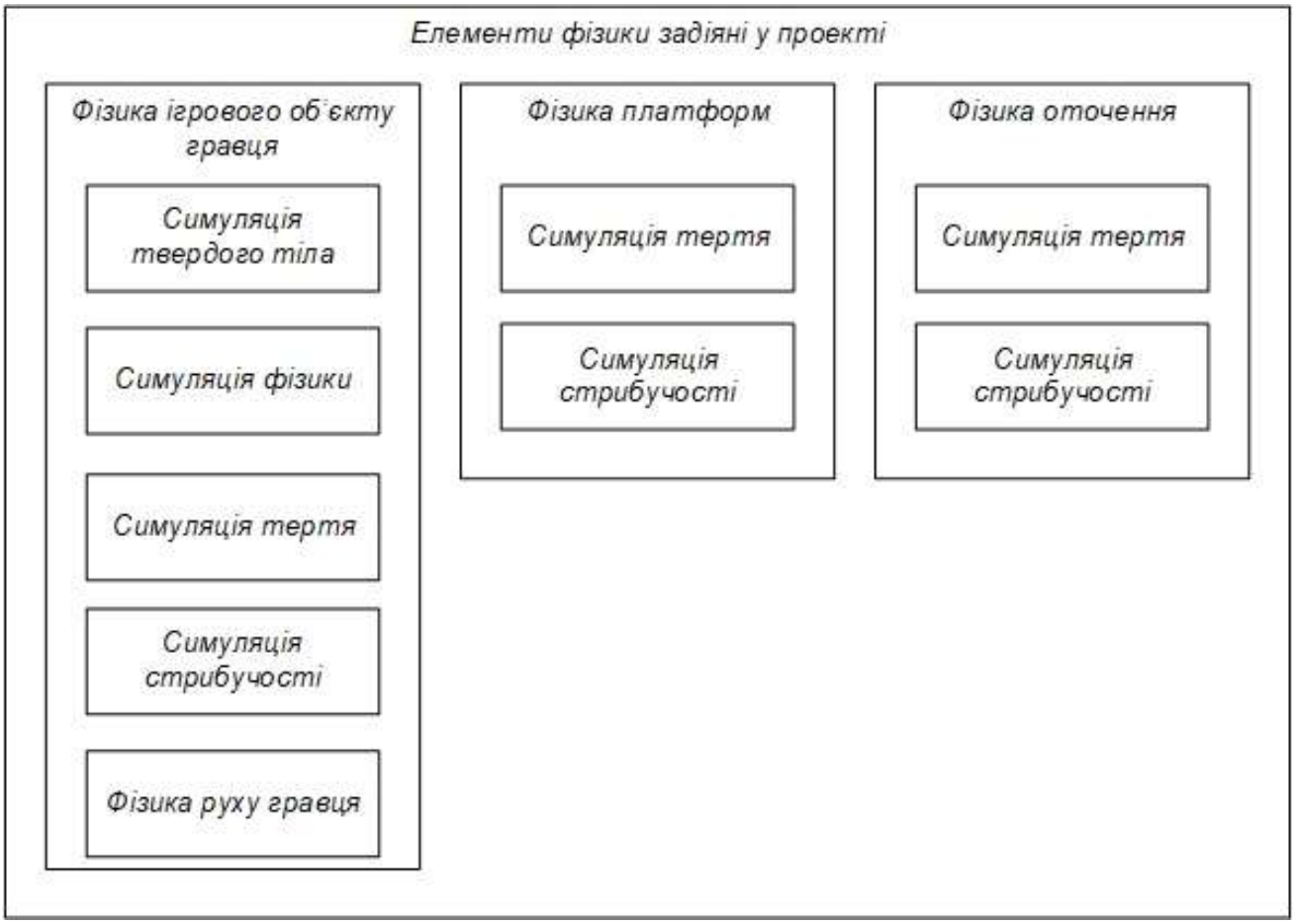


Рисунок 1.8. Елементи фізики задіяні у грі 3DBallz

Як можна побачити з рисунку, ігрові об'єкті для яких передбачено використання фізики поділені на три види. Основним оператором фізичного ядра Unity буде ігровий об'єкт гравця. Симуляція твердого тіла необхідна для коректного руху сфери по рівню, а також використання оточення. Також потрібна симуляція фізики – гравітації та взаємодії при зіштовхуванні з іншими ігровими об'єктами, що мають свої колайдери. Симуляція тертя та стрибучості потрібні для створення більш різних ігрових ситуацій. Окремо можна виділити фізику руху гравця. Не дивлячись на те, що вона забезпечується симуляцією твердого тіла, але фізика руху буде прораховуватись окремо через код та введення від користувача.

Фізика платформ та оточення схожі. Вони мають симулювати властивості тертя поверхонь та стрибучості для створення ситуацій, коли гравцю буде складніше діставатись до точки виходу, або навпаки створювати ситуації, коли гравець буде направлений у потрібному руслі за допомогою левелдизайну.

Всі ігрові об'єкти задіяні у симуляції фізики повинні мати у своєму складі

компонент колайдеру та відповідний матеріал. Додавання ж твердого тіла є обов'язковим лише для ігрового об'єкту гравця, оскільки тоді потрібно буде редагувати кожну платформу та вимикати симуляцію гравітації.

1.5 Реалізація основних ігрових елементів

1.5.1 Створення проекту гри та його налаштування

Для початку розробки ігрового проекту необхідно створити проект. Для цього у ігрового програмного рушія Unity є ПЗ, яке дає змогу створювати проекти централізовано, виконувати їх менеджмент, налаштування, інсталювати різні версії рушія та інше. Ця програма називається Unity Hub і саме в ній буде створено проект. Процес створення проекту можна побачити на рисунку 1.9.

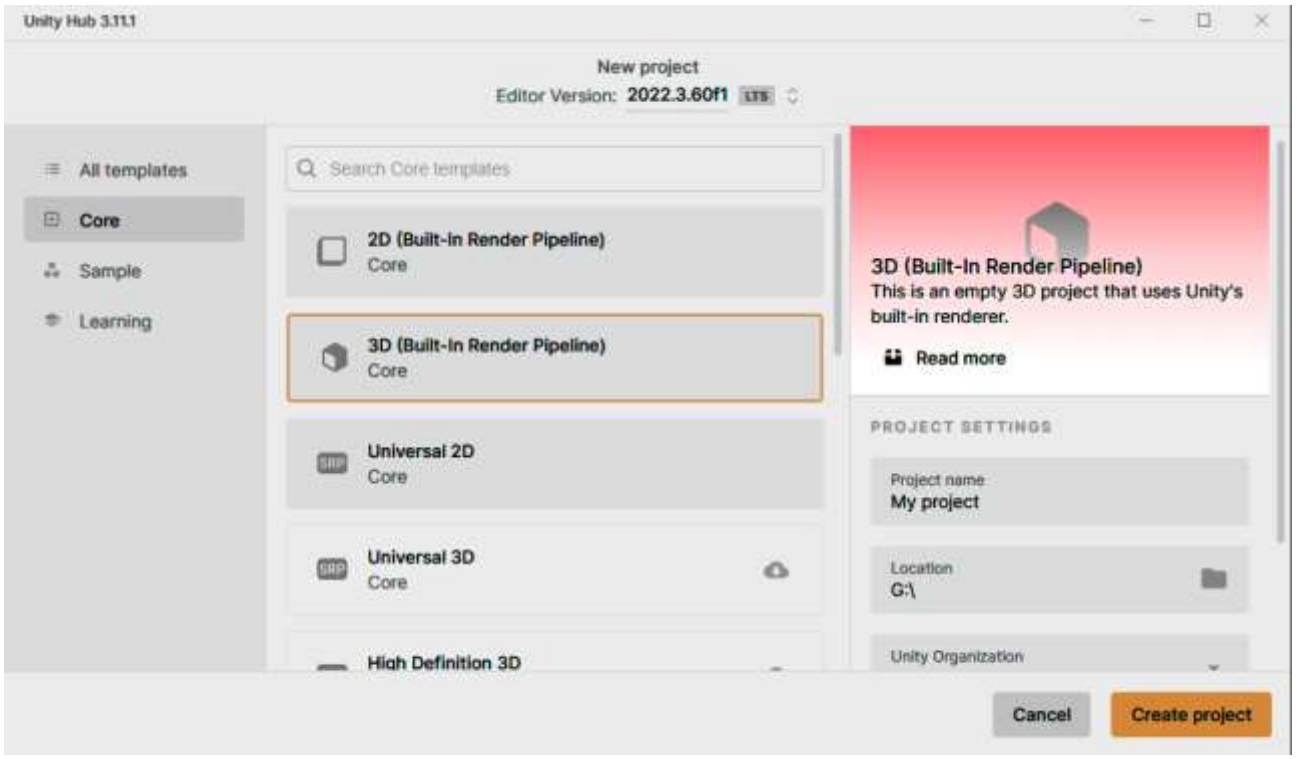


Рисунок 1.9. Вікно створення проекту у ігровому програмному рушії Unity

При створені проектів потрібно визначити із тим, який шаблон проекту використовувати. Ігровий програмний рушій дає деякий набір початкових шаблонів. Ці шаблони мають вже завантажені моделі, ігрові об'єкти, а також налаштування, які більш підходять для тієї чи іншої гри. Використання шаблонів є корисним, оскільки дає змогу використати вже створений базис та не витрачати час на його створення.

Після введення даних проекту та натискання кнопки Create Project, Unity Hub власноруч виконає побудову проекту, створить всі необхідні папки та пакети, а також завантажить всі ассети, які були обрані початковим шаблоном. Середа розробки Unity у які буде виконуватись робота має вигляд, як зображено на рисунку 1.10.

Рисунок 1.10. Загальний вигляд середи розробки Unity

Налаштування проекту полягають у обрані параметрів управління гравця, середі розробки для скриптів, створенню каталогів для ассетів гри.

1.5.2 Створення матеріалів та спрайтів для гри

Наступним важливим кроком у розробці гри є створення матеріалів та спрайтів. Спрайти потрібні для використання їх, як текстур для примітивів. Оскільки розробка гри знаходиться на стартовому етапі та створення повноцінних 3D-об’єктів не є основною метою, для реалізації першого білду гри будуть використовуватись примітиви. Для їх зображення будуть створені спрайти. Всього необхідно створити спрайти для поверхонь платформ, а також для води. В поточному проекті, вода не буде використовувати шейдери. В цілому весь проект буде стилізований під гру Minecraft. Всі спрайти створюються у графічному редакторі Photoshop. На рисунку 1.11 можна побачити створені спрайти для гри.

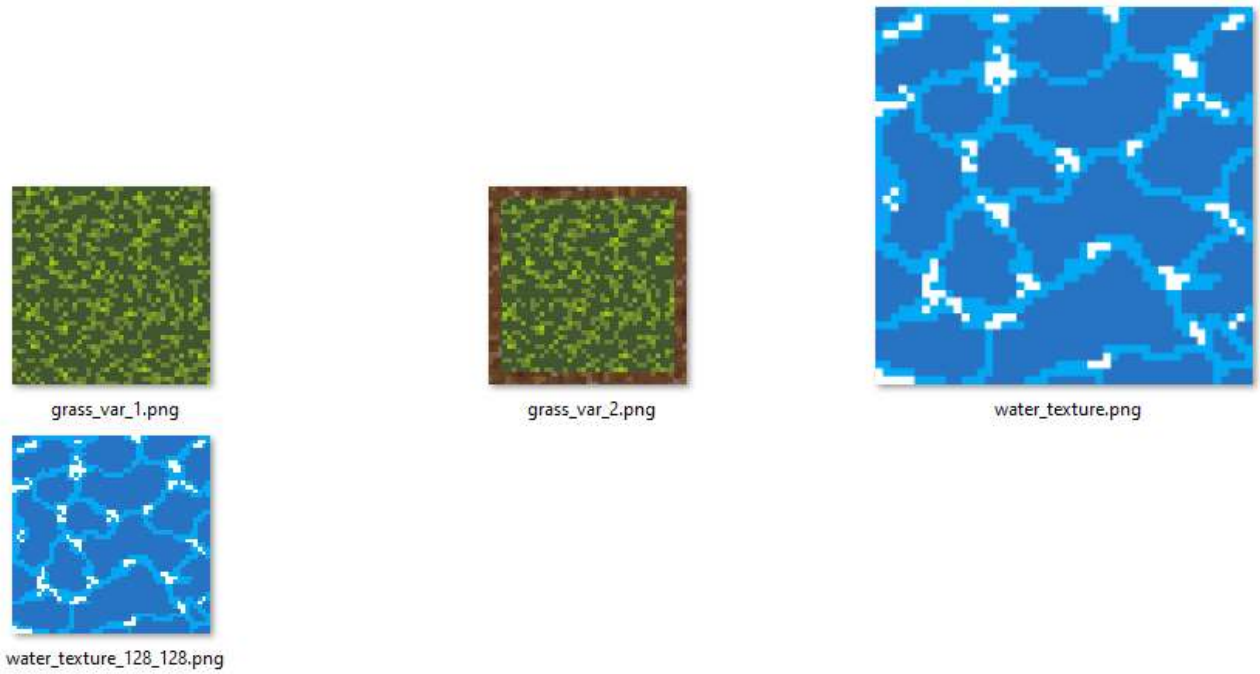


Рисунок 1.11. Створені спрайти для гри 3DBallz

Після створення спрайтів, потрібно реалізувати матеріали. Матеріали – це прості шейдери, які додаються до компонентів рендеру ігрових об’єктів, які мають у своєму складі Mesh-сітку. Матеріали є однорідними та більше нагадують використання інструменту заливка на об’єкти. У випадках, коли потрібно ігровий об’єкт покрити одним кольором, використання матеріалів є досить простим та швидким виходом. Окрім того, матеріали мають свої параметри кольору та альбедо, які можна редагувати безпосередньо з коду, що дає змогу змінювати зовнішній вигляд ігрових об’єктів на льоту. Для даного проекту було створено матеріали для сфери гравця, та платформ різного призначення.

Доцільність використання монокольору для сфери гравця викликана потребою реалізувати через колір індикації запасу енергії у розпорядженні для виконання стрибків.

Платформи різного призначення мають різні монокольори для того, щоби гравець міг запам’ятати, яка платформа що робить у ігровому процесі. Результат створення матеріалів можна побачити на рисунку 1.12.

Рисунок 1.12. Створені матеріали для об’єктів гравця та різних платформ

1.5.3 Створення ігрового об’єкту гравця

Для гравця було використовувано 3D-примітив – сферу. Вона створюється у панелі ієрархії ігрових об’єктів. На рисунку 1.13 зображено створений примітив сфери гравця.

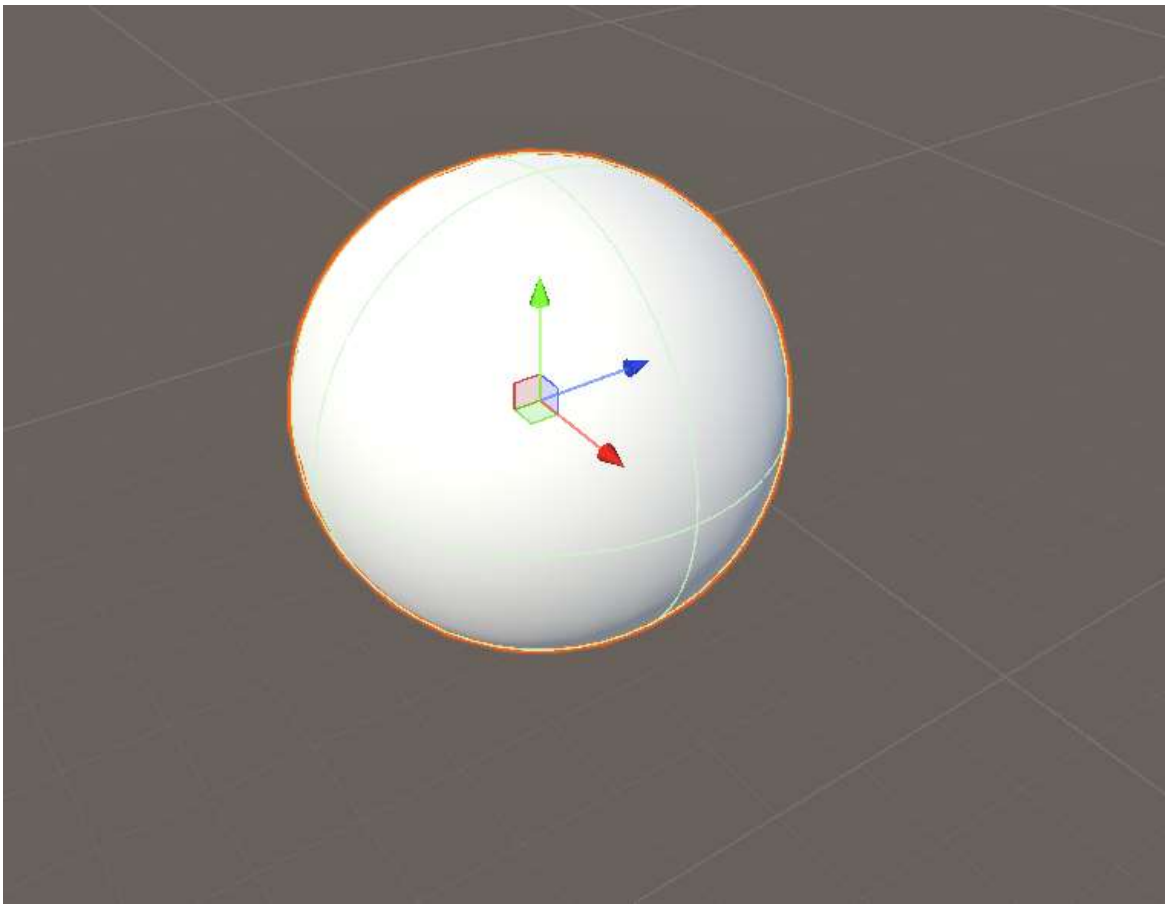


Рисунок 1.13. Примітив сфери гравця

На початку, як будь-який інший примітив, сфера має свої компоненти. Компоненти – це властивості, які описують ігровий об’єкт, його властивості та можливості. У будь-якого ігрового об’єкту є лише один компонент, який

загальний для всіх – Transform. Цей компонент відповідає за розміщення ігрового об’єкту у просторі, його кутів нахилу, та масштабування. Параметри цього компоненту виставлені у значення 0, для розміщення у середині сцени, виключення поворотів. Властивість Scale залишаю зі значеннями 1, оскільки це вказує на те, що сфера буде відтворюватись у сцені у масштабі 1 до 1. Переглянути компоненти сфери можна на рисунку 1.14.

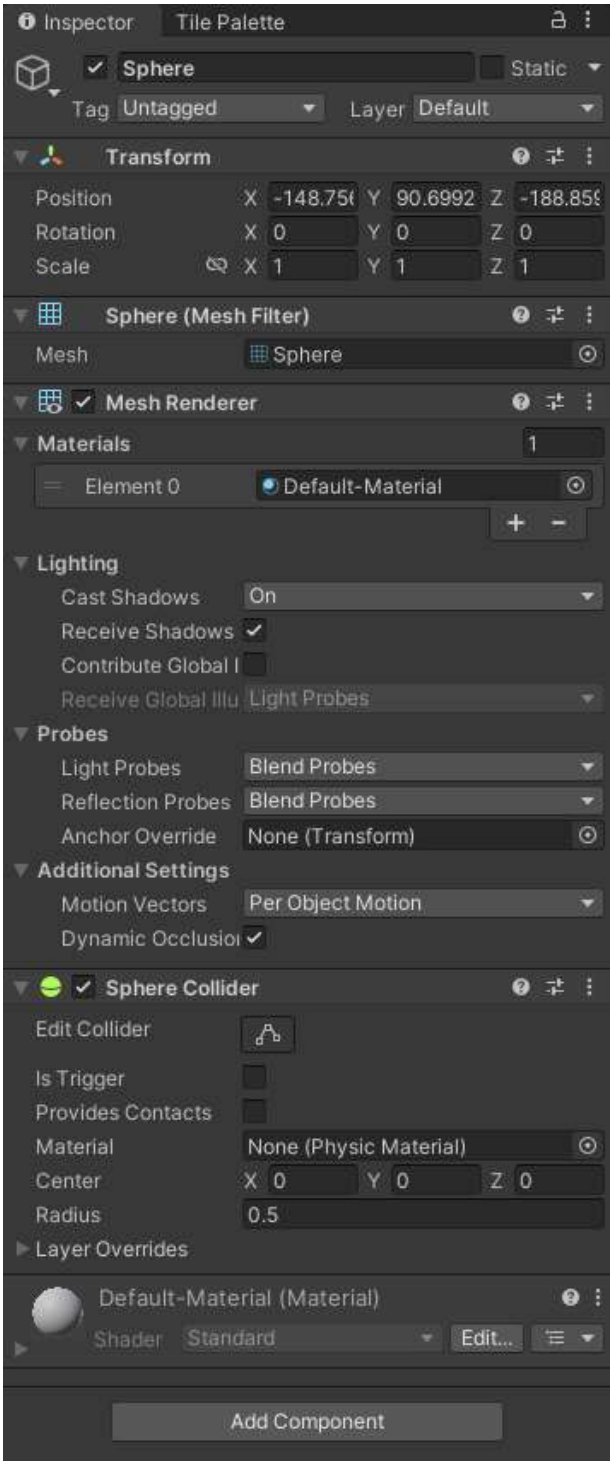


Рисунок 1.14. Компоненти примітиву Sphere

Компоненти можна вільно редагувати, змінювати та додавати нові. Для реалізації ігрового об’єкту гравця, потрібно додати до нього матеріал гравця у Mesh Renderer. Тепер, звертаючись до цього компоненту, можна отримати доступ до матеріалу сфери гравця, та при потребі змінювати його.

Також, згідно проекту використання фізики, потрібно додати до сфери ще один компонент Collider який буде відповідати за тригерні події, тому йому виставлено відповідний параметр у властивостях компоненту.

Для симуляції фізики та реалізації руху, також додаю до сфери компонент RigidBody. У цього компонента обов’язково потрібно увімкнути використання гравітації. Також цей об’єкт потрібно налаштувати таким чином, щоби заморозити його оберти навколо осі Y, оскільки зміна в цій осі буде змінювати орієнтацію руху сфери гравця впливаючи на швидкість та коректність його руху. Також, важливо переконатись, що властивість IsKinematic вимкнена, у противному випадку, сфера буде статично знаходитись на сцені без симуляцій.

1.5.4 Створення рівня за допомогою тайлів

Для створення локацій у ігровому програмному рушії існують декілька підходів. Але найбільш продуктивним с них для поточного проекту є використання системи тайлів. Tile – це невелике зображення, яке може безшовно з’єднуватись з іншим тайлом, не порушуючи їх зовнішній вигляд. Частіше за все у 2D-іграх використовують саме цю систему, оскільки за допомогою інструменту Tile Palette можна створювати палітру тайлів та просто малювати ними на рівні.

Для даного проекту використовувати стандартній інструмент малювання тайлів не вийде, оскільки мені потрібно використовувати 3D-об’єкти, які не можуть бути отримані із класичних тайлів. Для вирішення цієї проблеми, потрібно у Package Manager підключити додатковій інструмент який має назву 2D Tilemap Extras. Цей пакет дає змогу працювати у Tile Palette не тільки зі спрайтами, але і ігровими об’єктами, або їх префабами. На рисунку 1.15 зображено процес роботи з Package Manager.

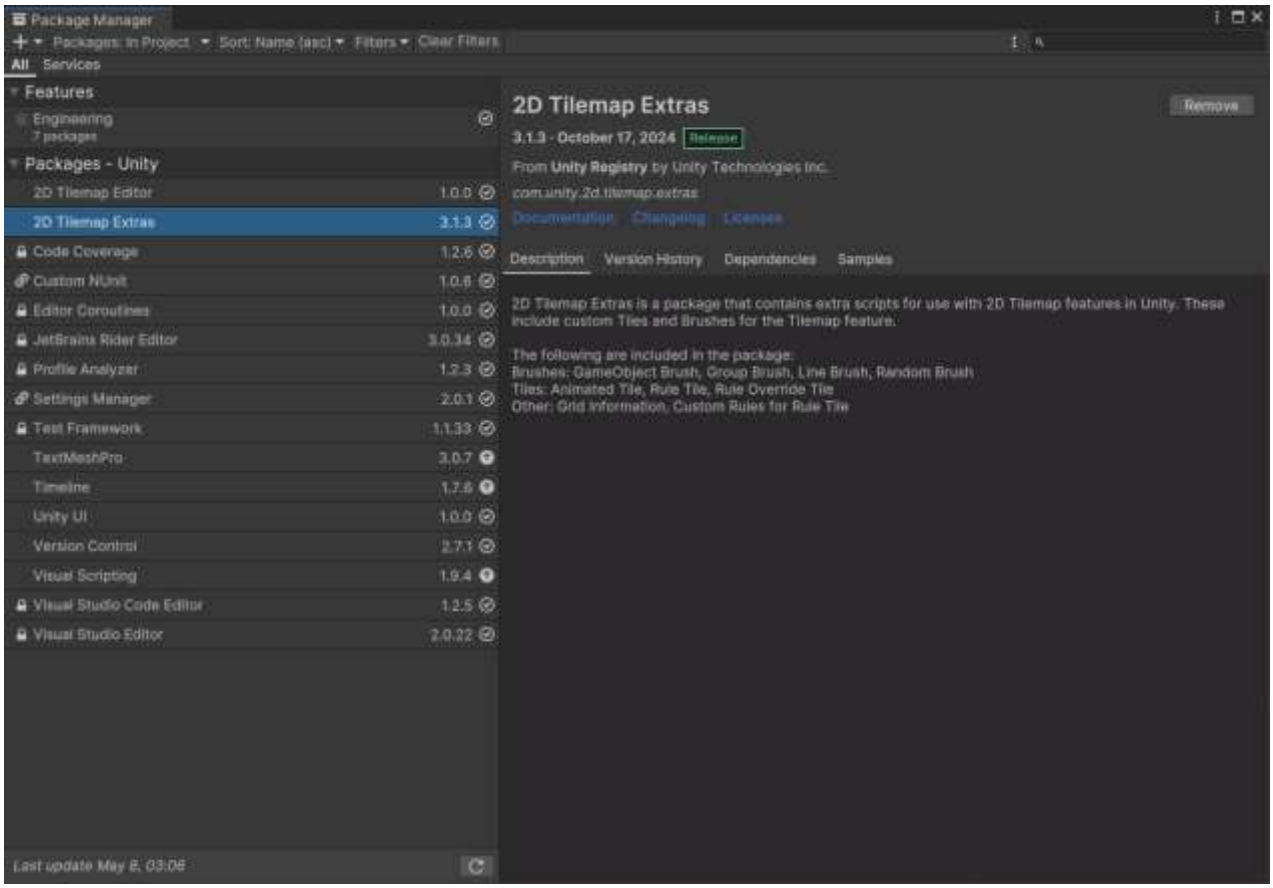


Рисунок 1.15. Процес роботи з Package Manager

Після додавання цього пакету, можна створити тайл-префаб. Префаби – це ігрові об’єкти, які збережені, як ассети проекту та можуть бути використані багаторазово. Це свого роду шаблон, який розробник може використовувати потім у проекті, змінюючи екземпляри цього шаблони. Ці зміни не будуть впливати на інші екземпляри та оригінал. У той же час додавання змін у оригінал приведе до внесення цих змін у всіх екземплярах. Ця особливість дуже зручна коли потрібно швидко змінити властивості, або компоненти у клонах префабу, що дає змогу значно зменшити кількість ітерацій виконання монотонної роботи з ігровими об’єктами.

Для створення префабу тайлу землі та води було використовано примітиви кубів, до яких додам відповідні спрайти. Для більш гнучкого підходу до створення рівнів, тайли землі та води було створено з чотирьох примітивів кубів, які будуть поєднані у один комбінований ігровий об’єкт. До кожного з кубів додаю колайдер для взаємодії з ігровим об’єктом гравця. Отриманий префаб має вигляд, як на рисунку 1.16:

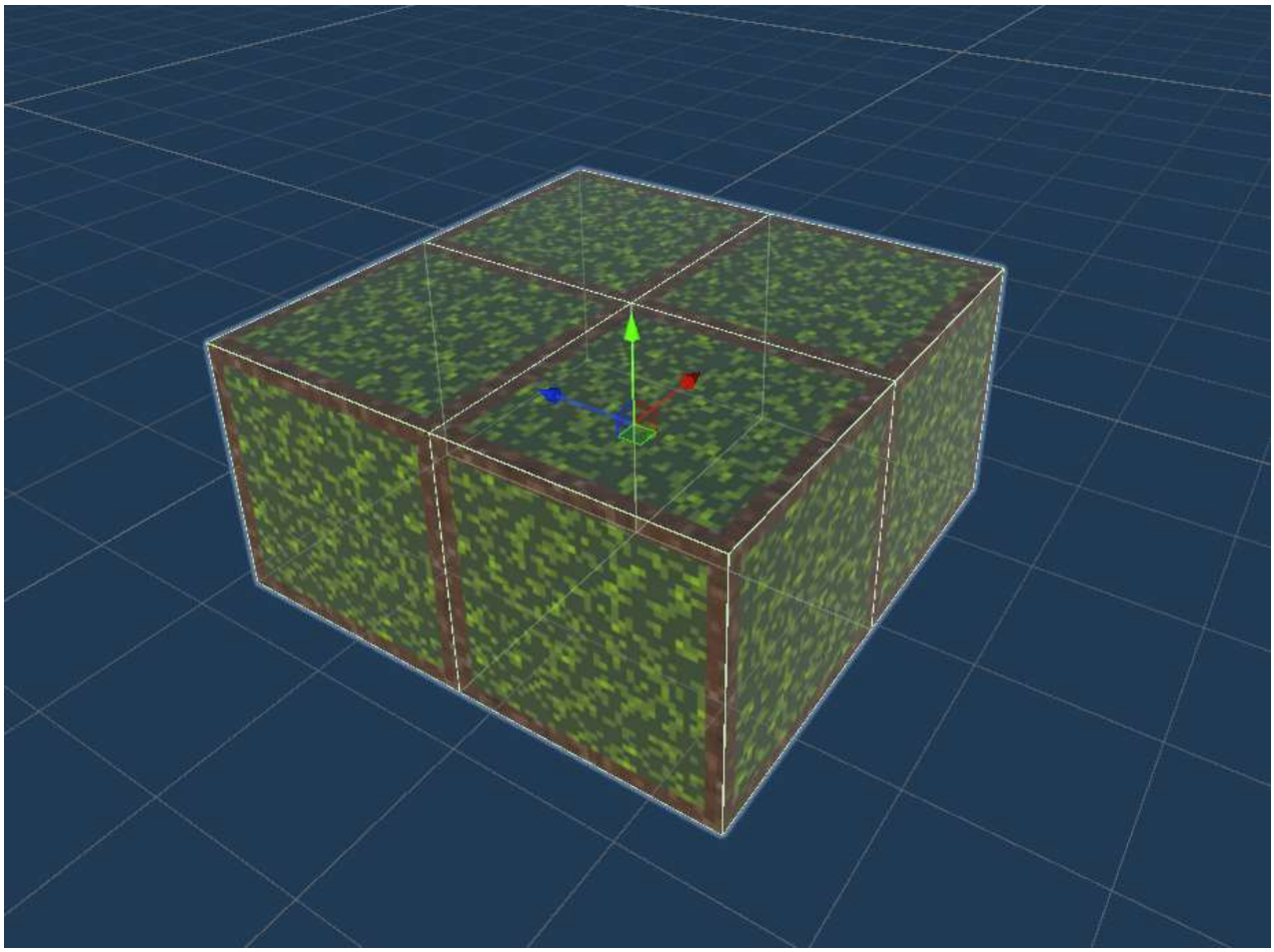


Рисунок 1.16. Префаб тайлу землі

Тепер, коли префаби готові, їх можна було б розміщувати просто створюючи їх копії на сцені, але це контрпродуктивно. Для полегшення роботи, обрано режим роботи пензеля у Tile Palette у режим GameObjects, після цього префаби потрібно додати до палітри. Це дає змогу використовувати ігрові об’єкти, як спрайти та швидко відтворювати рівень таким чином, як нам потрібно. Префаби з палітри будуть розміщуватись у груповому ігровому об’єкті Grid, який буде центрувати блоки рівня по клітинкам. В разі, якщо потрібен елемент іншої форми, або зовнішнього вигляду, достатньо створити префаб та також додати його до палітри. Таким самим чином створюю префаб для води, але в його параметрах висоти виставляємо її трохи нижче, ніж висота землі. Це створить ілюзію водної поверхні та берегової лінії.

Вказані вище прийоми дають змогу створити рівень але він буде плоским. Для додавання вертикальності локаціям, потрібно створювати ще один груповий ігровий об’єкт Grid, виставляти висоту на один рівень вищу та розміщувати

префаби землі. За допомогою такого прийому можна побудувати великі рівні досить швидко та ефективно, перемикаючись між грідами, утворюючи не тільки нерівності місцевості, але й більш складні структури, як каньйони, рампи чи мости. Для цих цілей потрібно створювати окремі префаби. Проблемним місцем використання такого підходу є продуктивність системи, якій потрібно буде відтворювати велику кількість невеличких примітивів та їх спрайтів. При побудові рівнів значно більшого розміру потрібно використовувати більші префаби за розміром, імітуючи таким чином розмір пензля.

У подальшому на рівнях таким самим чином можна розміщувати об’єкти на поверхнях наприклад такі як дерева, кущі, каміння. Для цього також потрібно створити префаб таких об’єктів, задати їм відповідні параметри, за потребою виставити потрібні теги, після чого додати до палітри. Для об’єктів оточуючої середи потрібно буде створити окремий грід, який буде розміщувати об’єкти відразу над рівнем поверхні. Сам грід потрібно буде відрегулювати таким чином, щоби розміри комірок гріда були досить малими. Це потрібно зробити для того, щоби отримати більш гнучкі можливості щодо позиціонування об’єктів на рівні.

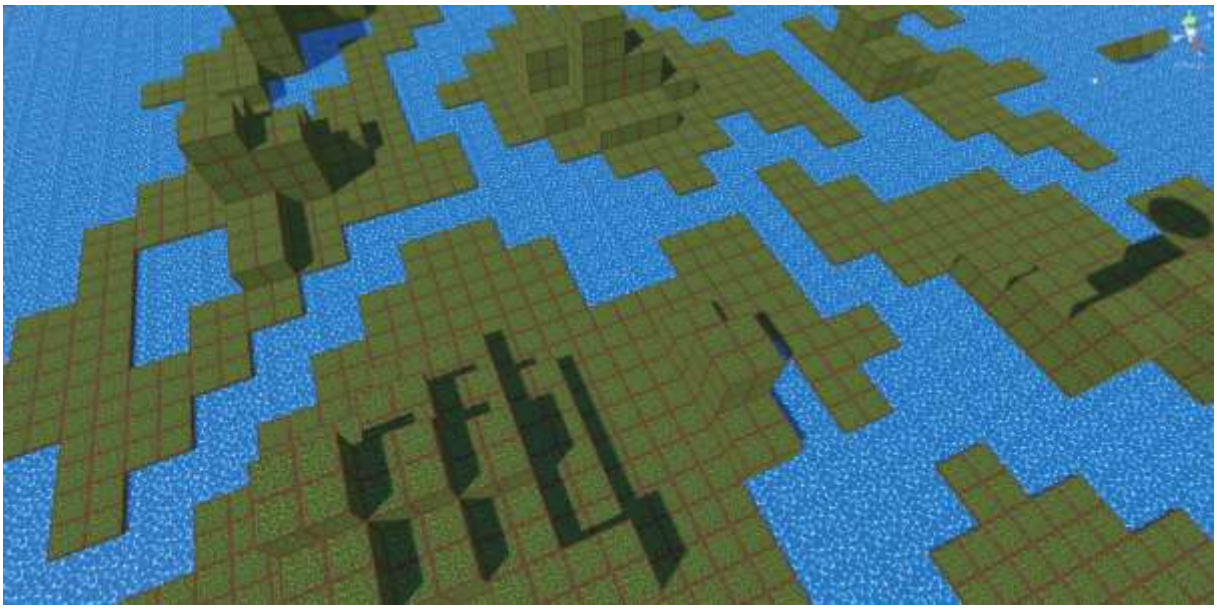


Рисунок 1.17. Результат створення рівня за допомогою GameObject у Tile Palette для гри 3DBallz

Важливим є той факт, що кожний створений таким чином префаб можна відредагувати, додавши до нього якусь унікальну властивість. У деяких місцях рівня, мною був змінений кут нахилу префабів, у інших префабах змінені фізичні

властивості. Ця особливість дозволяє більш детально редагувати рівень, за потреби створюючи унікальні ігрові ситуації. Кінцевий результат створення рівня за допомогою GameObject у Tile Palette можна на рисунку 1.17:

Рівень створювався з ціллю використання вертикального геймплею, щоби гравець використовував стрибки, а також шукав вихід з рівня деякий час. Падіння з платформ буде призводити до проходження рівня знову збільшуючи час проведений у грі.

1.5.5 Написання логіки ігрових об’єктів

Після створення рівня а також ігрового об’єкту гравця. Потрібно реалізувати логіку гравця – його параметри – та платформи з різними факторами дії на гравця. Параметри гравця є важливою частиною гри, оскільки вони залежать від рівня складності та реалізують механізм динамічної зміни ігрового процесу. Також виділення параметрів гравця в окремий модуль дає змогу у подальшому додавати нові параметри, або змінювати існуючі параметри виходячи з ігрового процесу, чи оновлень ігрового процесу. Те ж саме стосується і логіки платформ. Для створення більш гнучкої структури гри а також реалізації принципів ООП, є обов’язковим створення окремого модуля логіки для ігрових платформ. Їх структура дасть змогу модифікувати існуючі платформи, або з часом створювати нові платформи з цікавими ігровими механіками. Структурно, параметри та методи гравця можна побачити на рисунку 1.18.

Всього для роботи логіки гравця потрібно чотири змінні, а саме:

- Енергія гравця – це поточне значення енергії гравця. Енергія єдиний ресурс доступний для гравця. Вона витрачається на стрибки та з часом регенерується.
- Максимум енергії – це максимальне значення до якого може регенерувати енергія гравця. Це значення змінюється в залежності від рівня складності або зовнішніх факторів.
- Сила стрибка гравця – це сила з якою сфера гравця буде виконувати стрибок. Так само як і максимум енергії, це значення залежить від рівня складності або зовнішніх факторів.

Рисунок 1.18. Параметри гравця та його методи

- Енергія на стрибок – значення енергії яка буде витрачена для стрибка. Це значення не є сталим и виходить з сили стрибка.

Властивості, або змінні, реалізуються створенням звичайних змінних типу значення з плаваючою точкою, а також публічним доступом. Єдиною змінною, яка не має публічного доступу є змінна MeshRenderer. Вона відповідає за внутрішню логіку коду параметрів гравця. Також для гравця реалізовані такі методи у коді скрипта player_parameters:

Метод Ініціалізації. Його головна мета лежить у тому, щоби виконати першочергову ініціалізацію параметрів гравця під час завантаження рівня. Параметри гравця в ідеалі мають бути завантажені з ігрового менеджеру, але у

разі, якщо з якоїсь причини ігровий менеджер не доступний, завантажуються параметри за замовченням, що можна побачити з частини коду нижче.

```
void Start(){
   if (GameManager.Instance != null)
   {
      maximum_player_energy = GameManager.Instance.ini_maximum_
player_energy;
      player_jump_force = GameManager.Instance.ini_player_jump_force;
   }
   else
   {
      maximum_player_energy = 100f;
      player_jump_force = 200f;
   }
   player_mr = GetComponent<MeshRenderer>();
   player_energy = maximum_player_energy;
   energy_to_jump = player_jump_force / 10f;}
```

Також можна побачити, що значення енергії для стрибка розраховується за формулою:

$$E_j = \frac{J_f}{10} \tag{1.1}$$

де:

- E_j – енергія на стрибок;
- J_f – сила стрибка гравця.

Методи Встановлення максимуму енергії та Метод встановлення сили стрибка представляють собою методи з публічним доступом для зміни значень змінних параметрів гравця. Це потрібно для того, щоби у зовнішніх модулях гри можна було отримати доступ до зміни параметрів гравця.

Метод зміна енергії гравця виконує роль збільшення, або зменшення енергії у залежності від отриманого коефіцієнта у разовому виконанні. Перед виконанням

свого коду, цей метод виконує перевірку, чи більше значення енергії за нуль, після чого виконує складання значення енергії з коефіцієнтом. Якщо значення енергії після цього буде меншою за нуль, то її значення встановлюється у нуль. Після виконання всіх операцій, цей метод встановлює новий колір сфери гравця в залежності від значення енергії. Код цього методу можна переглянути нижче:

```
public void change_player_energy(float amount){
    if(player_energy > 0){
        player_energy += amount;
        if (player_energy < 0)
            player_energy = 0;
        player_mr.material.color = new Color(player_energy / 100f,
player_mr.material.color.g, player_mr.material.color.b);}}
```

Метод регенерація енергії гравця виконує завдання з постійного збільшення енергії гравця – її регенерації – або із підсилення цього процесу. Структурно цей метод дещо відрізняється від попереднього, тим що перевіряє чи менше значення поточної енергії гравця від максимального значення енергії. Якщо воно менше, то виконує додавання до поточної енергії коефіцієнту збільшення. Так само як і попередній метод виконує зміну кольору сфери в залежності від поточного значення енергії гравця. Важливою частиною цього методу є обмеження кількості енергії – виконується перевірка, щоби енергії не було більше за її максимальний показник, якщо перевірка не виконується, то значення енергії встановлюється у максимальний показник. Код цього методу можна побачити нижче:

```
public void regen_player_energy(float rate_amount){
    if (player_energy < maximum_player_energy)
    {
        player_energy += rate_amount;
        player_mr.material.color = new Color(player_energy / 100f,
player_mr.material.color.g, player_mr.material.color.b);
        if(player_energy > maximum_player_energy)
            player_energy = maximum_player_energy;}}
```

В обох випадках використання коефіцієнту, параметрів amount та rate_amount, дає можливість передавати в методи і від’ємні значення, що досить корисно, оскільки зменшує кількість коду, потрібну для написання.

Наступними ігровими об’єктами, які потрібно запрограмувати є активні платформи, що взаємодіють з енергією гравця. За оригінальним задумом одні платформи повинні віднімати енергію, а інші її відновлювати. Для цього було написано досить простий код, який базується на роботі тригерів. На рисунку 1.19 схематично зображено роботу цього коду.

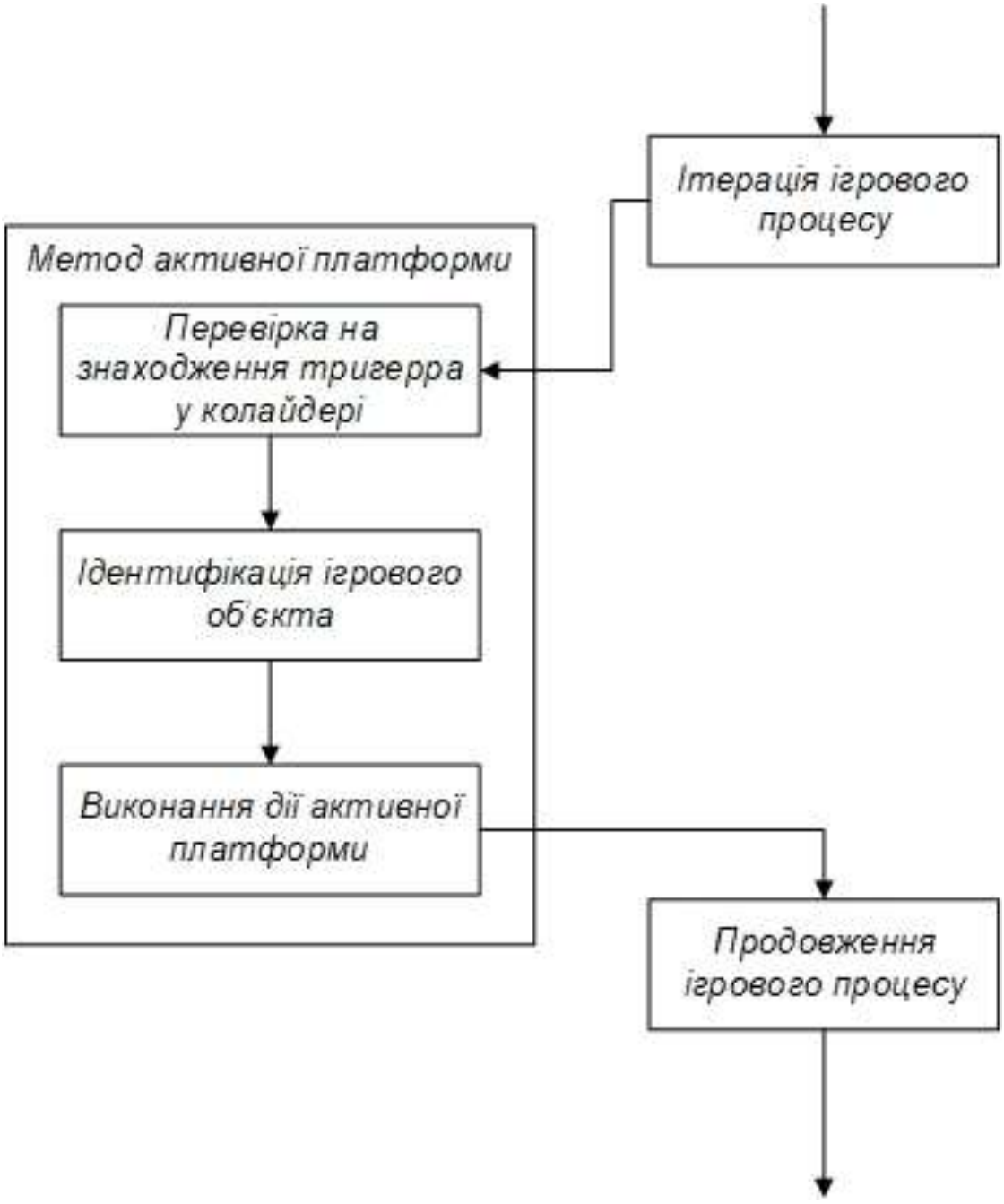


Рисунок 1.19. Схема роботи коду активних платформ

Як видно з рисунку, кожну ітерацію ігрового процесу, гра перевіряє тригерні методи у активних платформах на предмет знаходження в колайдерах клатформ інших колайдерів. Якщо такі знайдені, платформа повинна ідентифікувати ігровий об’єкт якому належить взаємодіючий колайдер. Для цього у коді активних платформ вказується посилання на код параметрів гравця, і при виконанні тригера, платформа намагається отримати код параметрів гравця. Якщо це вийшло зробити, то з активною платформою взаємодіє сфера гравця, тоді потрібно виконати дію платформи. Якщо отримати параметри гравця не вдалось, тоді тригер ігнорується.

Всього було створено два окремих коди для двох видів платформ: “regen_platform_logic.cs” та “reduce_platform_logic.cs”. Тих що генерують енергію для сфери гравця та тих, які навпаки енергію зменшують. Вони відрізняються тим, що у випадку ідентифікації сфери гравця викликають у коді параметрів гравця Метод регенерації, або Метод зміни енергії та передають значення коефіцієнту. Для створення більш керованих ігрових ситуацій, величини коефіцієнтів створені як змінні-поля, тому їх значення можна редагувати безпосередньо для кожної платформи окремо у редакторі сцени ігрового рушія Unity. Це також є інструментом для впливу на ігрову ситуацію під час створення рівнів, створюючи просторові головоломки для гравця. На рисунку 1.20 зображено деякі реалізовані активні платформи на рівні гри.

Ще одна активна платформа, яка потребує опису – платформа для виходу з рівня “exit_platform_logic.cs”. Її призначення у виводі гравця з рівня та запуск наступного. У Unity існує поняття сцен. Вони представляють собою простір, в якому знаходяться ігрові об’єкти та виконується який-небудь код. Сцени можна використовувати багатьма способами, але головне призначення сцен – реалізація рівнів у іграх. Тому кожний рівень у моїй грі це окрема сцена. Сцени можна викликати з коду за допомогою спеціальної команди, яка запускає з менеджеру сцен саме потрібну, знаходячи її за іменем. Для автоматизації роботи цього скрипта при розробці наступних рівнів, ім’я сцени, яка повинна завантажуватись при активації платформи вводиться у поле у редакторі Unity розробником.

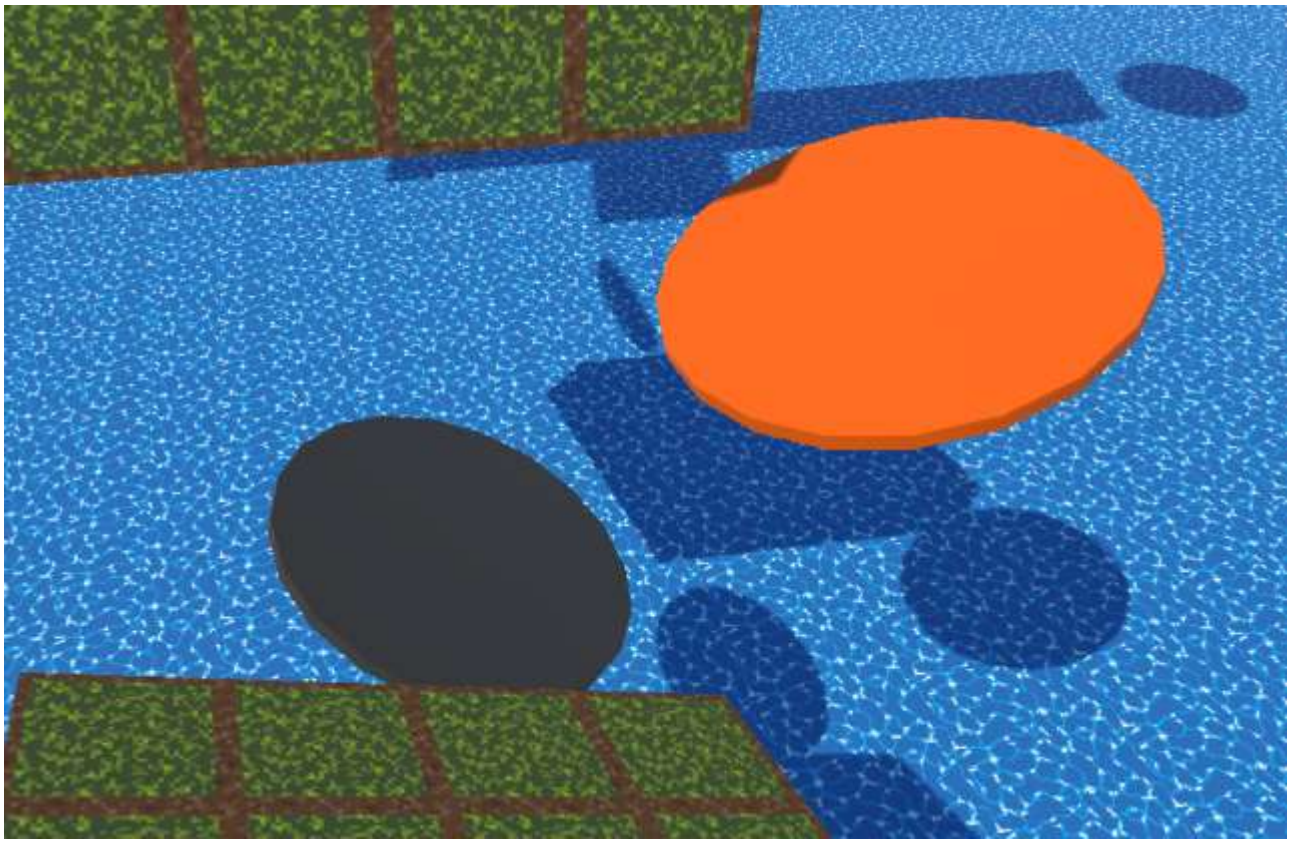


Рисунок 1.20. Скріншот активних платформ розміщених на сцені

Тому не потрібно створювати окремий код для кожного рівня. В цілому, функціонально, платформа працює так само, як і активні платформи розглянуті вище, для виконання виклику наступного рівня, необхідно, щоби тригер колайдер сфери гравця залишився в колайдері платформи. Нижче представлений код для цієї платформи:

```
[SerializeField] string scene_name;
Player_parameters _player_parameters;
private void OnTriggerStay(Collider other){
   _player_parameters = other.GetComponent<Player_parameters>();
   if (_player_parameters != null){
      SceneManager.LoadScene(scene_name);}
}
```

Останній код, який потрібно реалізувати – це код для неігрових поверхонь, або таких поверхонь, які мають знищувати сферу гравця. В моєму випадку, роль таких поверхонь грають тайли води та код “water_logic.cs”. Принцип роботи цього коду такий самий, як і код, що завантажує наступний рівень, з тою відмінністю,

що тайли води переміщують гравця не на наступний рівень, а перезавантажують поточний рівень. Тому рядок коду, який завантажує рівень виглядає наступним чином:

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

Як можна побачити, тут менеджер сцени звертається до самого себе та отримує ім’я активної сцени, тобто тієї в якій зараз знаходиться гравець. З точки зору доцільно перезавантаження рівня, то це можна розглядати, як перезавантаження рівня у іграх серії Dark Souls.

Описані вище кроки створили ігрові об’єкти та ігрову логіку для основних елементів гри, без врахування фізичних розрахунків. Створені скрипти, префаби та ігрові об’єкти дають базис для створення великої кількості рівнів різного наповнення та ігрового досвіду.

1.6 Реалізація використання фізичного ядра програмного рушія Unity

1.6.1 Реалізація переміщення гравця по локації. Робота з камерою

Для виконання основного ігрового задуму потрібно дати змогу гравцю переміщуватись по рівню. Для цього потрібно реалізувати відповідний код. Разом з тим важливим є реалізація механізму слідкування камери за сферою гравця. Оскільки гра буде від третього лиця, дуже важливо, щоби камера чітко слідкувала за рухами сфери. Водночас, потрібно враховувати, що через додавання вертикальності на рівнях, необхідно реалізувати можливість змінювати кут камери. Це в свою чергу тягне за собою проблему з переміщенням гравця по рівню. Проблема криється у методі переміщення сфери.

У Unity є досить широкий спектр можливих варіантів, як реалізувати переміщення об’єктів у просторі. Самий простий з них – це переміщення об’єкту задаючи йому нові координати. Такий спосіб не є фізично коректним, оскільки він не враховує фізичне тіло ігрового об’єкту, при цьому переміщення по координатам виконується до фізичних розрахунків, тому об’єкти будуть заходити у інші ігрові об’єкти, після чого фізика ігрового рушія буде намагатись їх

виштовхувати. У зв'язку з цим, потрібно використовувати метод фізично коректного руху гравця у просторі, а саме методи класу RigidBody, який відповідає за симуляцію твердого тіла.

Клас RigidBody також має декілька способів реалізації переміщення. Це рух по вектору напрямку Velocity(), рух від точки А до точки Б MoveTo(), а також додавання сили до об'єкту AddForce(). У моєму випадку важливо використовувати саме додавання сили до об'єкту, оскільки важливо, щоби сфера мала інерцію руху та мала можливість саме перекачуватись по поверхням, а не ковзати.

Використання методу AddForce() полягає у тому, щоби завдавати силу у визначених векторах напрямку. Після чого розпочинається розрахунок сили, з якою ігровий об'єкт що має тверде тіло повинен почати інерційний рух у напрямку вектору з визначеною силою, яка буде гаснути відповідно до сили тертя оточення навколо об'єкту, що рухається. Складність полягає у тому, що управління сферою виконується у напрямках вперед-назад, вліво-вправо. Якщо змінити кут огляду камери, тоді напрямки руху з точки зору гравця зміняться, але для сфери гравця, вектори сторін ігрового світу залишаться незмінними, через що зміниться перспектива руху. Наприклад якщо почати дивитись на сферу з правої її сторони, тоді кнопка вправо буде рухати м'яч у бік камери, що для нас є рухом назад. При реалізації коду переміщення гравця та камери потрібно буде врахувати ці особливості. На рисунку 1.21 зображено схему взаємодії реалізуємих модулів.

Код слідкування камерою "player_folower.cs" функціонально досить простий. В першу чергу, у блоці Ініціалізації змінних, створюється посилання на ігровий об'єкт гравця тобто на сферу з метою отримання її координат в просторі. Також створюється змінна режиму роботи камери, яка має тип short, оскільки буде зберігати в собі лише чотири значення, від нуля до трьох. Кожна з цифр відповідає тому, з якого боку в поточний момент часу камера дивиться на сферу гравця. Наприклад за замовченням ця змінна має значення нуль, тобто дивиться «у спину» сфері. Ця змінна потрібна для того, щоби коректно реалізовувати код переміщення гравця у разі зміни режиму камери.

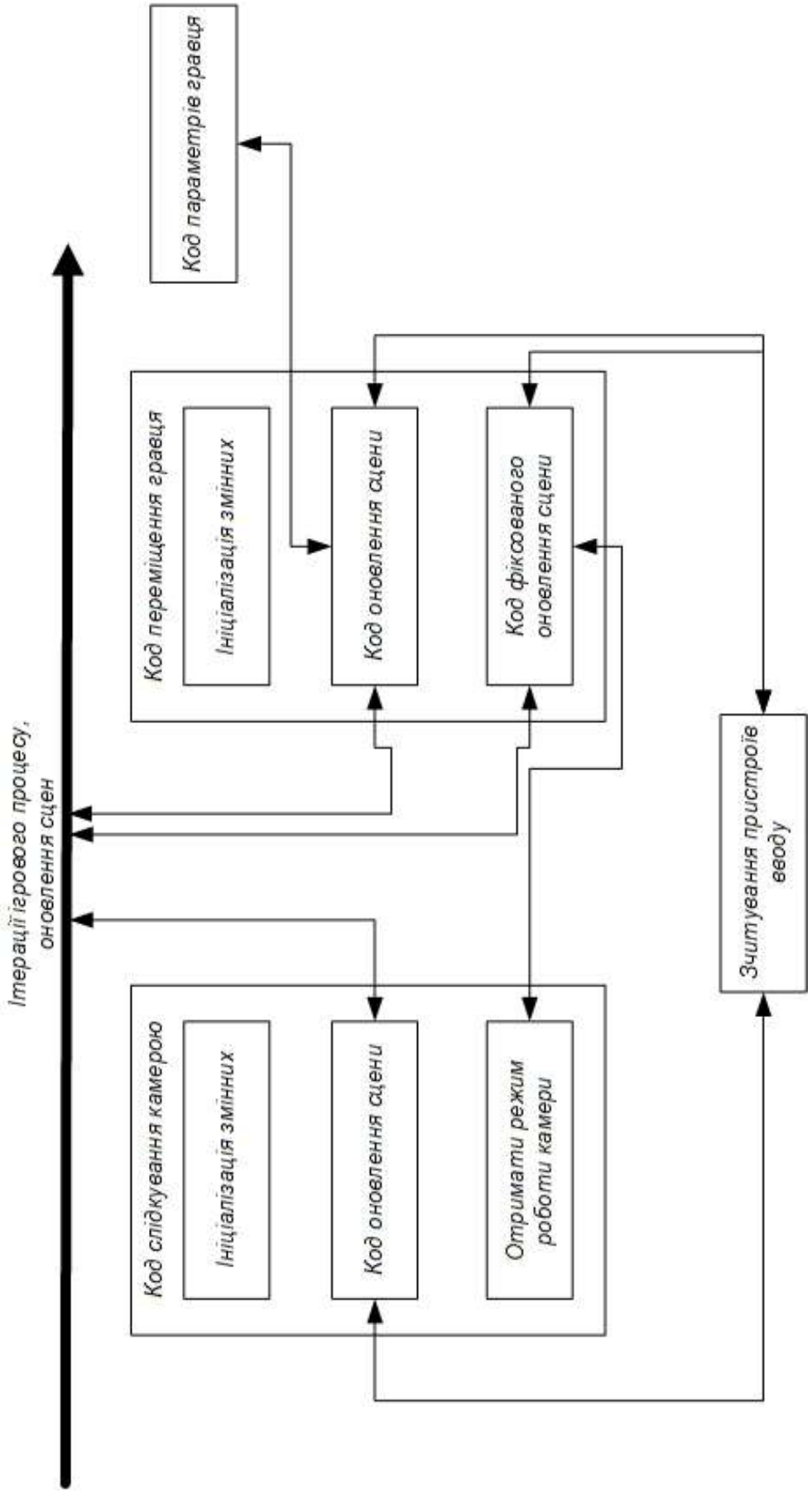


Рисунок 1.21. Схема роботи коду Коду слідкування камерою та Коду переміщення гравця для гри 3DBallz

Наступний блок це Код оновлення сцени. Він виконується кожний кадр гри, тому має велику частоту виконання. Саме в цьому блоці камера звертається до блоку Зчитування пристроїв вводу, перевіряючи чи не була натиснуті кнопки Q або E які відповідають за зміну куту перегляду камери. У разі натискання однієї з цих кнопок значення змінної режиму роботи камери змінюється, з виконанням перевірок на перемикання зі значення три на нуль, утворюючи кільцевий процес зміні куту огляду гравця камерою. Нижче представлено код, який це реалізовує:

```
if (Input.GetKeyDown(KeyCode.E)){
    if (camera_mode < 3)
        camera_mode++;
    else if (camera_mode == 3)
        camera_mode = 0;}
else if (Input.GetKeyDown(KeyCode.Q)){
    if (camera_mode > 0)
        camera_mode--;
    else if (camera_mode == 0)
        camera_mode = 3;}
```

Наступним кодом який виконується у блоці оновлення є перемикач, який дивиться, на поточний режим роботи камери та виставляє відповідні координати для камери та її кут огляду гравця. Зміна цих значень виконується у компоненті Transform в його полях Position та Rotation. Оскільки камера не має фізичного тіла, можна напряму вказувати потрібні значення. Слід зауважити, що тут виконується звернення до ігрового об’єкту гравця для отримання його координат. Це потрібно робити кожний раз для точного переміщення камери у просторі відразу за гравцем. Нижче представлена частина коду, яка реалізує позиціонування та кут огляду гравця для одного з режимів:

```
case 2:{
        transform.position = new Vector3(player_game_object.transform.
position.x, player_game_object.transform.position.y + 7, player_game_object.
transform.position.z + 5);
```

transform.rotation = Quaternion.Euler(60, -180, 0);

break;}

Останній блок це метод Отримання режиму роботи камери – все що робить цей код, повертає значення режиму роботи камери. Цей метод викликається лише кодом переміщення гравця та реалізує безпечну передачу значення змінної.

Код переміщення гравця “player_movement.cs” також має три складові. У своїй початковій частині, під час створення сцени, виконується блок Ініціалізація змінних. Тут створюються посилання на ігровий об’єкт камери гравця, на код параметрів гравця, а також отримується посилання на компонент RigidBody сфери гравця.

Блок Код оновлення сцени виконується кожний кадр, який відтворює гра. Його головна задача відслідковувати натискання гравця на кнопки, що відповідають за той, чи інший функціонал, але не переміщення. Переміщення повинне виконуватись однакову кількість разів у секунду, а метод Update(), який лежить в основі розглядаємого блоку, виконується таку кількість разів, яку може відтворити комп’ютер гравця. Це може бути і 30 кадрів, а може бути і 1226 кадрів, тому у різних випадках переміщення буде виконуватись або 30 разів, або 1226 разів, що не допустимо. У той же час зчитування натискання на кнопку стрибку, або виходу з гри, потрібно виконувати максимальну кількість разів за секунду, щоби не виникало проблем з імпактом вводу.

Окремо потрібно зупинитись на реалізації фізично вірних стрибків у цьому блоці коду. Кожний кадр, гра перевіряє чи кількість енергії гравця – яка отримується через посилання на параметри гравця – більша за потрібну кількість енергії для стрибка. Якщо це вірно, тоді перевіряється натискання на клавішу Space. У разі виконання цієї умови, то до твердого тіла сфери гравця надається сила у напрямку глобальної осі Y сцени. Після виконання команди на стрибок, код звертається до параметрів гравця та зменшує кількість енергії на значення енергії потрібної для стрибка. Нижче приведений код, який реалізує фізично коректні стрибки:

if (_player_parameters.player_energy > _player_parameters.energy_to_jump){

```
        if (Input.GetKeyDown(KeyCode.Space)){
            player_rb.AddForce(Vector3.up                    *
_player_parameters.player_jump_force);
            _player_parameters.change_player_energy(-
(_player_parameters.energy_to _jump));}}
```

Наступний блок коду, який реалізовується Код фіксованого оновлення сцени. Саме тут реалізується переміщення гравця коректну кількість разів, фізично коректними способами. Цей блок відразу розпочинається зі звернення до Коду слідкування камерою та отримує режим роботи камери. Режим впливає на те, як саме буде реагувати сфера гравця на натискання кнопок управління, тому запускається відповідна гілка перемикача. Після чого код звертається до пристроїв вводу. Якщо була натиснута кнопка, яка відповідає за рухи в сторону, то буде виконуватись процес надавання сили до сфери гравця у відповідному напрямку. Нижче приведений код двох режимів перемикача руху гравця:

```
    case 0:{
        if (Input.GetAxis("Horizontal") > 0f)
            player_rb.AddForce(Vector3.right * 5f);
        if (Input.GetAxis("Horizontal") < 0f)
            player_rb.AddForce(Vector3.left * 5f);
        if (Input.GetAxis("Vertical") > 0f)
            player_rb.AddForce(Vector3.forward * 5f);
        if (Input.GetAxis("Vertical") < 0f)
            player_rb.AddForce(Vector3.back * 5f);
        break;}
    case 1:{
        if (Input.GetAxis("Horizontal") > 0f)
            player_rb.AddForce(Vector3.forward * 5f);
        if (Input.GetAxis("Horizontal") < 0f)
            player_rb.AddForce(Vector3.back * 5f);
        if (Input.GetAxis("Vertical") > 0f)
```

```
    player_rb.AddForce(Vector3.left * 5f);
if (Input.GetAxis("Vertical") < 0f)
    player_rb.AddForce(Vector3.right * 5f);
break;}
```

Як можна бачити з представленого коду, у різних режимах роботи камери, при однаковому натисканні клавіш управління, до сфери гравця застосовуються сила у різних напрямках, що відповідає тому, як гравець бачить сферу у поточний час ігрового процесу. Останнім що робить цей блок коду – викликає у параметрах гравця метод регенерації енергії гравця. Саме тут, фіксовану кількість разів в кадр, енергія гравця виконує своє відновлення.

1.6.2 Робота з фізичними матеріалами

Важливою частиною роботи з фізикою у моїй грі є робота з фізичними матеріалами. Вище було створено звичайні матеріали, які можна застосувати до мешів ігрових об’єктів, але фізичні матеріали це зовсім інше. Це окремий вид матеріалів, які передають до фізичного рушія Unity додаткову інформацію про те, яким чином він має симулювати взаємодію об’єктів. Такі матеріали створюються таким же чином, як і матеріали для візуально зміни мешей моделей. На рисунку 1.22 зображено типовий неналаштований фізичний матеріал.

Рисунок 1.22. Властивості фізичного матеріалу за замовченням

Як можна бачити з рисунку вище, у фізичних матеріалів є параметри динамічного тертя, статичного тертя, стрибучості, а також режими примноження

сил тертя та стрибучості. Всі ці параметри можна вільно налаштовувати.

Для реалізацій симуляції поверхонь потрібно створити фізичні матеріали для різних випадків. При роботі з фізичними матеріалами важливо розуміти, що вони додаються до ігрових об’єктів, як звичайні матеріали, але тільки до компонентів колайдерів. Якщо ігрового об’єкту немає колайдеру, то фізичний матеріал використати до нього неможливо. Тому це дає змогу створювати різні фізичні параметри поверхонь для невеличких проміжків ігрової локації. Для своєї гри створено три матеріали, а саме фізичний матеріал для стрибучих поверхонь, який забезпечую надвисоку стрибучість, фізичний матеріал для поверхонь з великою силою тертя, для ковзання, а також окремий фізичний матеріал для гравця, щоби сфера відчувалась не кругли камінням, а створювала враження кульки-м’яча. На рисунку 1.23 зображено створені фізичні матеріали.

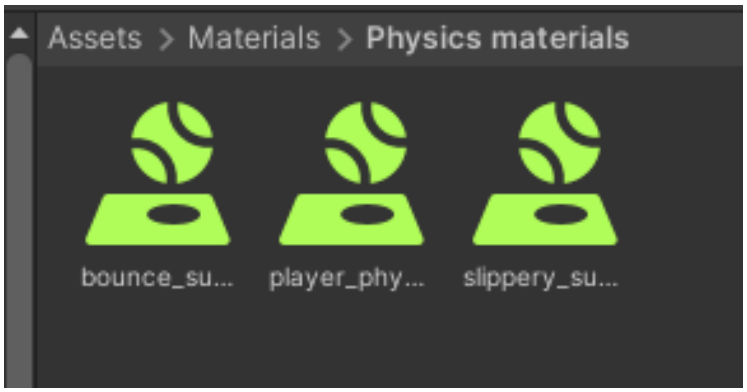


Рисунок 1.23. Фізичні матеріали для різних поверхонь гри 3DBallz

У подальшому розвитку проекту, можна створювати додаткові матеріали та комбінувати їх, додаючи колайдери. В цілому різні властивості поверхонь дають змогу створювати складні ситуації у ігровому процесі, що спонукає гравця знаходитись у напруженні.

1.6.3 Налаштування платформ на локації

Маючи код, який реалізовує ігрову логіку, а також ігрові механіки для гравця, можна розпочати редагування створеного рівня та додавати до його елементів фізичних параметрів та відповідного коду.

В цілому поточний етап роботи цілком є геймдизайнерским елементом, коли потрібно не тільки розмістити платформи та префаби на рівні, але зробити

це таким чином, щоби утворювати єдиний геймплейний задум. Основною метою гри є створення виклику для гравця у його подоланні складних ситуацій на рівні. Через це головною метою є створення складного, але в той же час збалансованого та реального для проходження рівня.

Можна виділити й паттерни у налаштуванні платформ на ігровому рівні. Наприклад, поверхні платформ біля води, або на краю платформ, що знаходяться у повітрі, можна роботи більш слизькими, щоби гравець мав тримати контроль за своєю сферою. Поверхні платформ, які регенерують енергію потрібно зробити більш стрибучими, щоби гравець не міг так просто отримувати енергію. Платформи, що навпаки забирають енергію потрібно зробити максимально не стрибучими, щоби вони створювали враження чорних дірок з яких немає шляху на втечу.

З технічної сторони питання, налаштування платформ на локації складається з редагування компонентів на панелі Inspector кожної з платформ, які розміщені на рівні. Окремо, при бажанні, можна редагувати кожний окремий куб у префабах поверхні для того, щоби надавати їм унікальні властивості. На рисунку 1.24 зображено відредаговані компоненти панелі Inspector для однієї з активних платформ.

На зображені можна виділити використання матеріалу відповідної платформи на меші ігрового об’єкту, що розмальовує платформу у рожевий колір. У колайдері платформи для регенерації в поле фізичного матеріалу застосовано поверхню з підвищеною стрибучістю. Слід відмітити, що колайдери платформ не мають власних тригерних колайдерів. Система тригерів виконується завдяки колайдеру-тригеру у сфери гравця. Цей колайдер на декілька пікселів більший за фізичний колайдер сфери і саме це забезпечує механізм активації платформ при контакті. Також потрібно відмітити, що у кожній активній панелі є такий компонент-скрипт, який описує її роботу. Цей компонент має лише одне поле – коефіцієнт регенерації, або зменшення енергії гравця. Оскільки всі платформи є похідними від префаба, то можна виставляти різні значення цих коефіцієнтів для кожної з платформ.

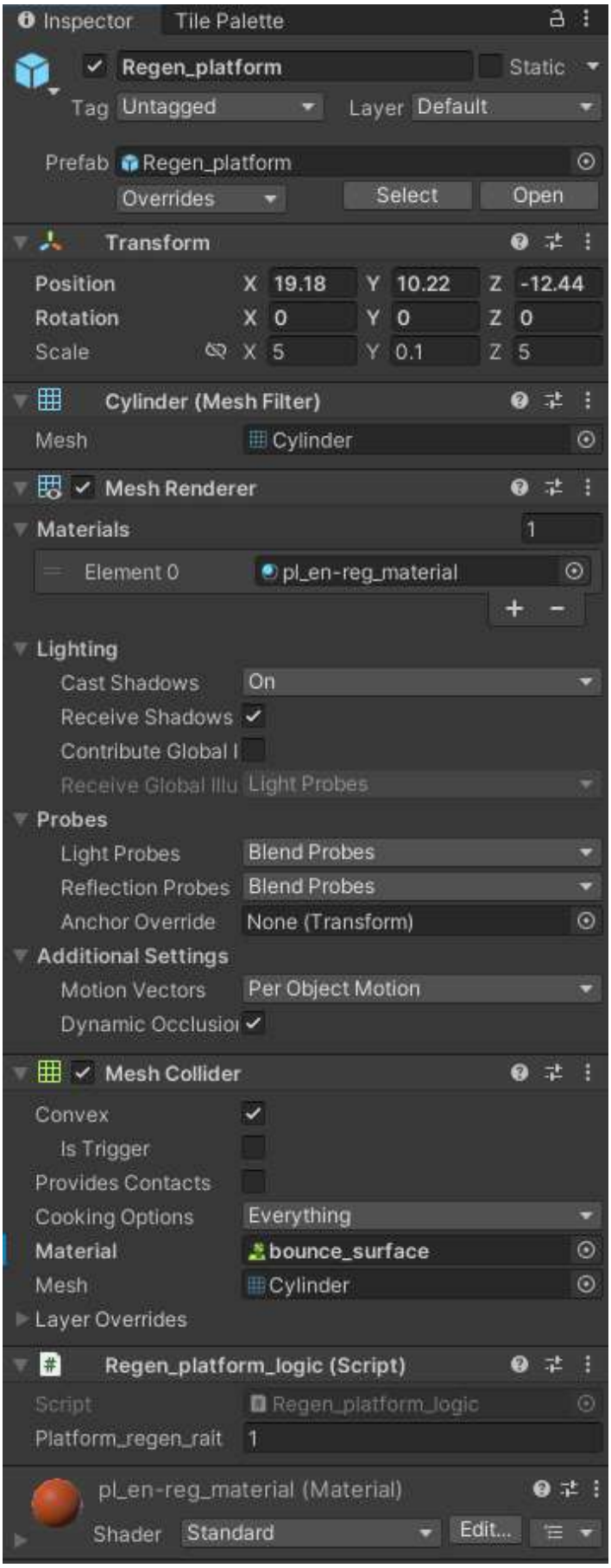


Рисунок 1.24. Відредаговані компоненти платформи для регенерації енергії гравця

1.7 Реалізація ігрового меню

1.7.1 Створення сцени ігрового меню та редагування Canvas

Для повноцінної реалізації гри потрібно реалізувати ігрове меню, яке буде зустрічати гравця. Для цього потрібно створити окрему сцену для головного меню у відповідному каталозі проекту. Між сценами можна вільно перемикатись під час роботи прямо у вікні редактора ігрового рушія Unity.

Для реалізації інтерфейсу гри у Unity є окремий ігровий об’єкт який має назву Canvas. Цей об’єкт створює площину, яка розміщується поверх усіх ігрових об’єктів на сцені гри, а також відображається на екрані. Всі елементи інтерфейсу, з якими можна взаємодіяти розміщуються саме у цьому ігровому об’єкті. У Unity достатньо велика кількість UI-елементів, кожний з котрих виконує ту чи іншу роль інтерфейсу.

Графічний інтерфейс користувача (UI) є важливою складовою взаємодії гравця з ігровим середовищем. У Unity для його створення використовується Canvas, який забезпечує належне розташування та рендеринг елементів інтерфейсу.

Основні концепції побудови UI в Unity:

Canvas як базова структура – забезпечує площину для розміщення всіх UI-елементів, що відображаються поверх ігрової сцени.

Render Mode – визначає спосіб рендерингу UI (екранний, світловий або у вигляді камери).

UI-елементи – набір інструментів для взаємодії користувача (текст, кнопки, поля введення, слайдери тощо).

Реалізувати ігрове меню не тривіальна задача, оскільки вона лежить у площині розмежування елементів один від одного та створення ієрархії та вкладених ігрових об’єктів. На етапі проектування було створено схему роботи ігрового меню (рисунок 1.6). Згідно цієї схеми потрібно створити щонайменше три окремі панелі зі своїми інтерфейсами, а також рисунок на задній фон та текст заголовку гри. На рисунку 1.25 зображено структуру інтерфейсу ігрового меню.

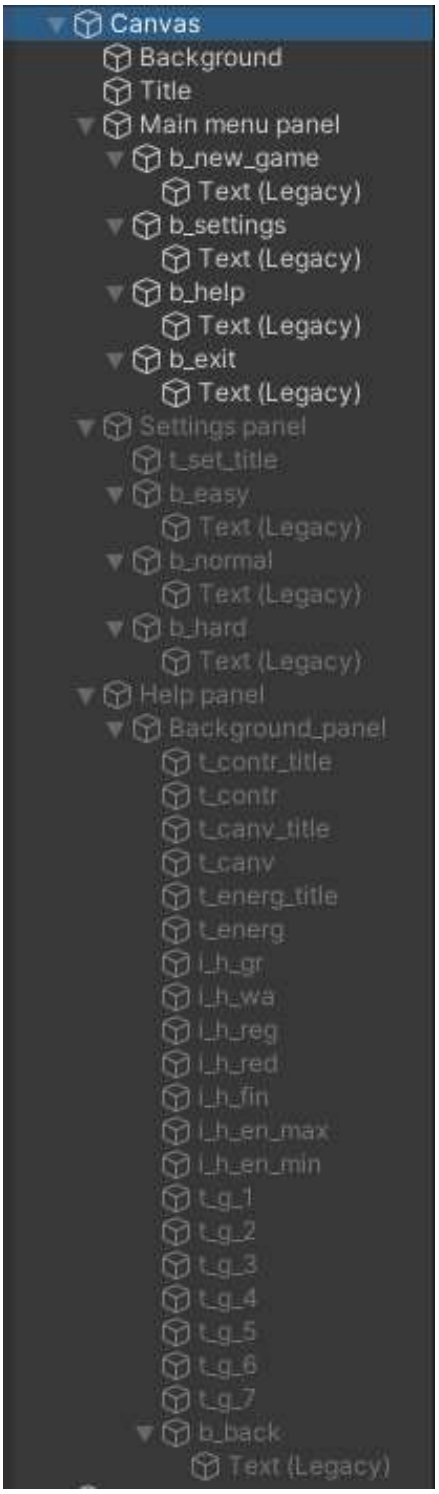


Рисунок 1.25. Структура інтерфейсу головного меню гри у ієрархії ігрових об’єктів сцени Main_menu

Як видно з рисунку, деякі панелі початково скриті, в той час як панель самого головного меню доступна відразу при завантаженні сцени. У подальшому, для реалізації роботи інтерфейсу потрібно буде написати код, який описуватиме його роботу. На рисунку 1.26 можна побачити створену панель головного меню гри.

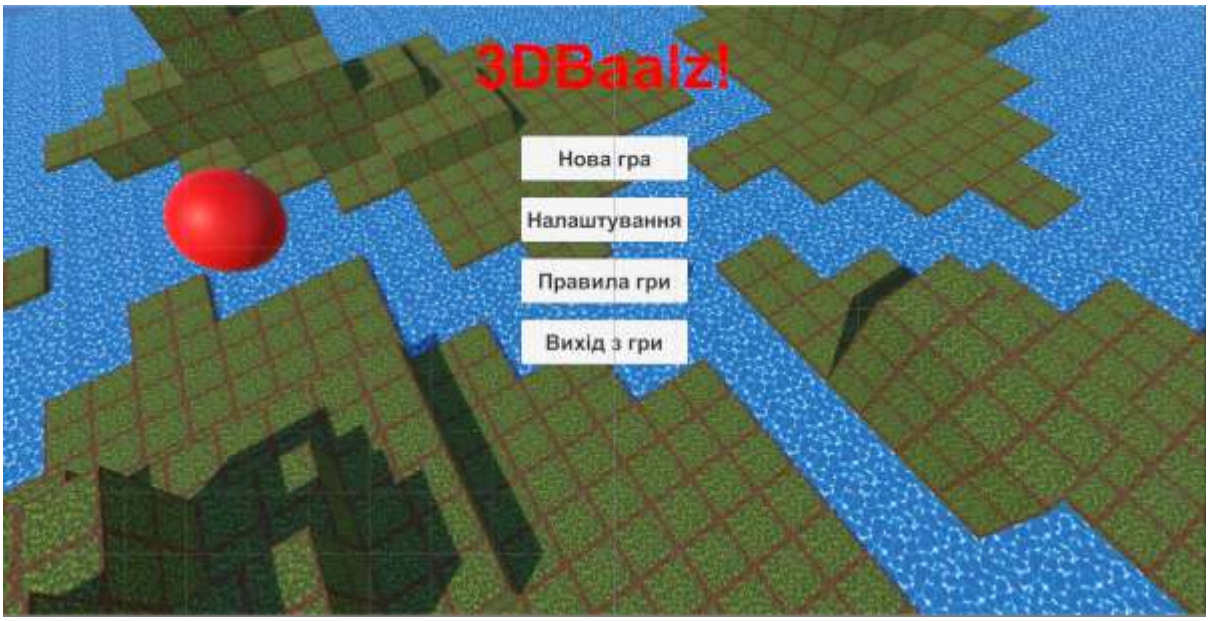


Рисунок 1.26. Панель головного меню гри 3DBallz

Ця панель є функційною відразу з початку гри. На ній розміщено чотири кнопки, кожна з яких веде гравця до відповідної дії. Кнопка Нова гра буде завантажувати перший рівень гри за його ім'ям. Кнопка Налаштування викликає панель обрання складності гри, зображену на рисунку 1.27.

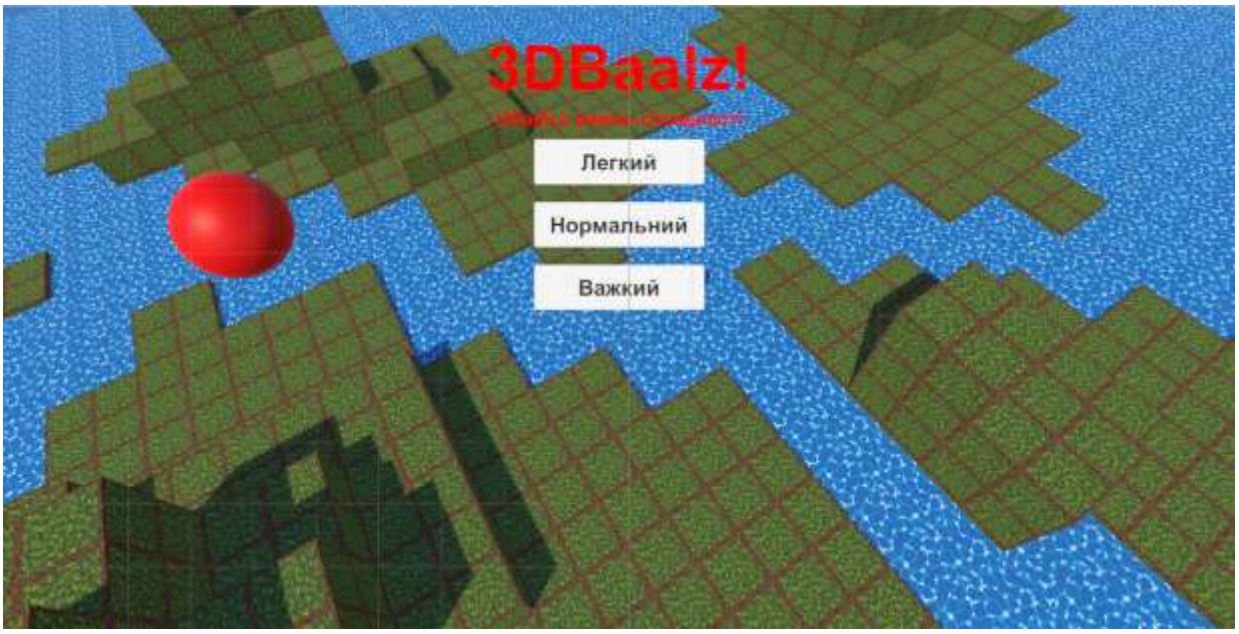


Рисунок 1.27. Панель обрання складності гри 3DBallz

Ця панель залишається активною до тих пір, поки гравець не натисне одну з кнопок складності. Кожна з цих кнопок буде виставляти параметри рівня складності у коді менеджеру рівнів. При натисканні на кнопку складності виконується завантаження панелі головного меню, наступною кнопкою в якому є

Правила гри. При натисканні завантажується панель з правилами гри, як зображено на рисунку 1.28.

Рисунок 1.28. Панель Правил гри 3DBallz

На цій панелі розміщено у текстовому форматі з зображеннями правила гри, управління сферою, мета гри, приклади поверхонь для взаємодії та інше. Єдиною функційною кнопкою тут є кнопка Назад, яка знову відображає головне меню гри. Остання кнопка в цьому меню вимикає додаток.

1.7.2 Програмування елементів меню та створення менеджеру рівнів

Для коректної роботи ігрового меню недостатньо розмістити елементи інтерфейсу гри. Потрібно прописати логіку роботи кожної кнопка, а також черговість виклику панелей, які відображаються на екрані гравця. Для цього створено код, який розміщено компонентом для Canvas сцени головного меню.

Безпосередньо принцип роботи цього коду дуже простий. Він отримує посилання на кожну з панелей ігрового меню. Після чого, всередині коду створити методи для реакції на натискання кожної кнопки у головному меню. З точки зору коду, який виконує цей скрипт, то він досить простий – одні панелі ховаються, а інші панелі знову стають доступні для взаємодії з гравцем.

Однією із головних складностей при імплементації цього коду є необхідність власноруч вказувати кожній кнопці, який метод потрібно викликати

у коді для головного меню.

Останній код, який потрібно реалізувати – це менеджер ігрового процесу. Цей скрипт обов'язковий до виконання, оскільки саме він відповідає за передачу даних між рівнями, за завантаження у параметри гравця даних по рівню складності.

Через особливість роботи сцен у Unity при написанні коду потрібно звернутись до принципів статичності у мові програмування C#. Цей принцип полягає у тому, що статичні члени класів існують лише в одному екземплярі для кожного екземпляру цього класу. Тобто, такий член не буде змінюватись на протязі всього циклу життя програми, а у моєму випадку гри. Таким статичним членом у моєму випадку буде змінна, що посилається на сам клас. Сутність роботи цього коду досить проста. Під час запуску головного меню, створюється ігровий об'єкт який зберігає в собі цей код. Сам код посилаючись на свій ігровий об'єкт повідомляє ігровому рушію Unity, що цей ігровий об'єкт не може бути знищений при зміні сцен, що показано у частині коду нижче:

```
public static GameManager Instance;
private void Awake(){
   if(Instance != null){
      Destroy(gameObject);
      return;}
   Instance = this;
   DontDestroyOnLoad(gameObject);}
```

Також можна побачити, що цей код перевіряє, чи вже існує цей ігровий об'єкт. Це робиться для того, щоби при завантажені рівнів не створювалась ще одна копія цього ігрового об'єкту. Далі у коді будь-якого коду в проекті, можна звернутись до цього класу та отримати з нього дані.

Дані, які зберігає цей клас – це дані рівня складності, які встановлюються шляхом натискання на кнопку складності у панелі обрання рівня складності. Ця кнопка викликає відповідний метод у класі ігрового менеджеру. Приклад такого

коду нижче:

```
public void set_medium_mode(){
    ini_maximum_player_energy = 200f;
    ini_player_jump_force = 300f;}
```

В результаті розробки було створено гру, яка дає можливість керуючи сферою переміщуватись по рівням. Реалізовано різні рівні складності, симуляцію фізичних поверхонь, створено різні активні платформи, які вносять різноманітність у ігровий процес.

Використання фізичного ядра програмного ігрового рушія Unity, дає змогу реалізовувати різні цікаві фізичні ефекти, як стрибучі, слизькі, чи навпаки поверхні від яких гравець може відірватись лише застосувавши силу. Окрім того фізичне ядро дало змогу симулювати коректне переміщення сфери на рівні, що в кращу сторону сприяє ігровому процесу. Кінцева структура реалізованого проекту зображена на рисунку 1.29:

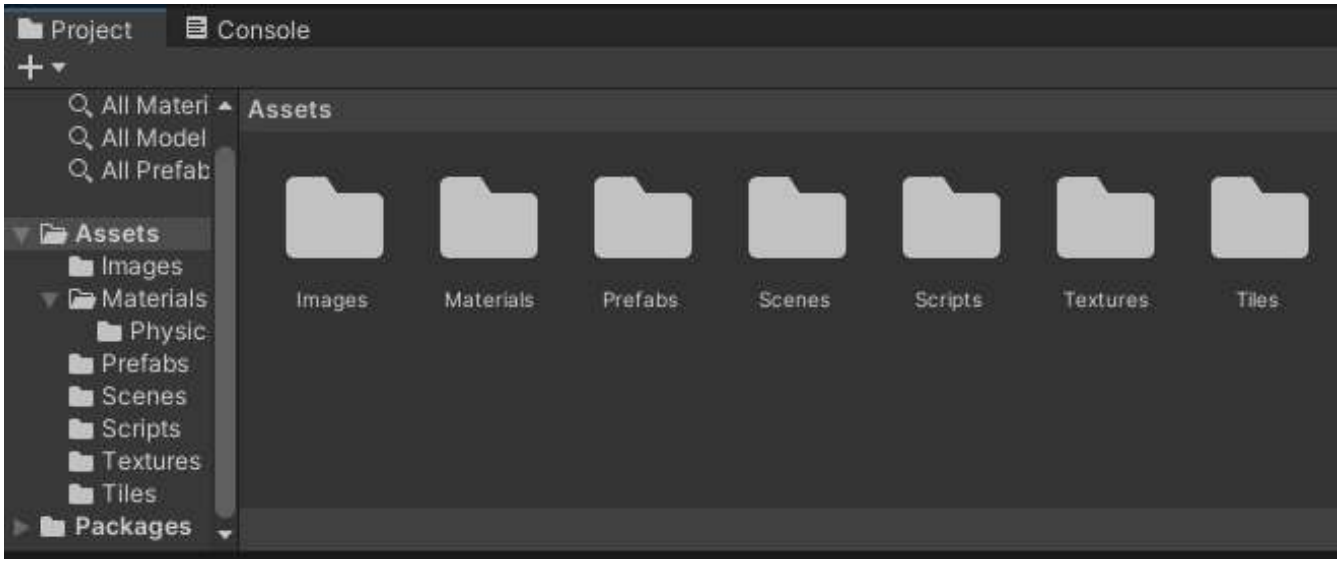


Рисунок 1.29. Кінцева структура каталогів гри 3DBallz

Створений проект має великий потенціал для подальшого розвитку як окремими доповненнями ігрових механік, так і повноцінними оновленнями з рівнями. Окремим напрямком для роботи може стати випадкова генерація рівнів, яка була б основана на вихідних даних, що задавав би гравець, чи рівні які б генерувались навколо гравця, а основною метою режиму був би збір очок до тих пір, поки гравець не впаде у воду, або інші клітки, на які неможна потрапляти.

1.8 Тестування гри

Для якісної реалізації будь-якої гри, її необхідно тестувати та відлагоджувати не тільки під час розробки, а ще й після закінчення кожного етапу розробки. Багато ігрових продуктів продовжують тестувати навіть тоді, коли гра у фінальній версії вже відправилась до печаті, або у магазини. Це робиться для того, щоби якомога швидше виправляти помилки, які не були знайдені під час розробки.

У випадку з реалізацією ігор на ігровому програмному рушії Unity, цей процес виконується безпосередньо в самому редакторі. Для тестування та відладки гри під час розробки, потрібно встановлювати ігрові об’єкти та сцени у такий стан, який потрібно протестувати та натиснути кнопку «Play». Таким же чином виконується тестування і відладка після закінчення розробки, достатньо виставити гру у її початковий стан и натиснути ту ж саму кнопку. На рисунках 1.30, 1.31 та 1.32 можна побачити процес тестування гри.

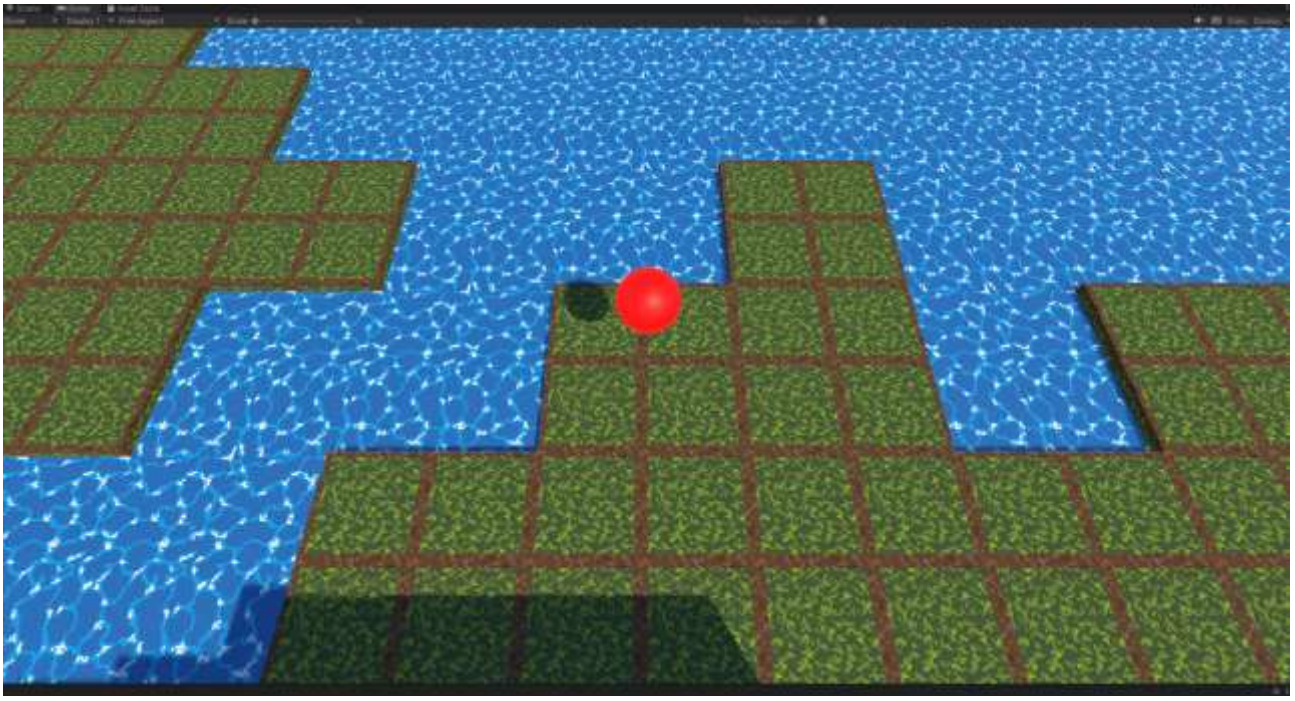


Рисунок 1.30. Процес тестування гри 3DBallz

Тестування гри проходило під час всього процесу розробки гри. Кожний етап написання коду закінчувався процесом відлагодження та перевірки коду на працездатність.

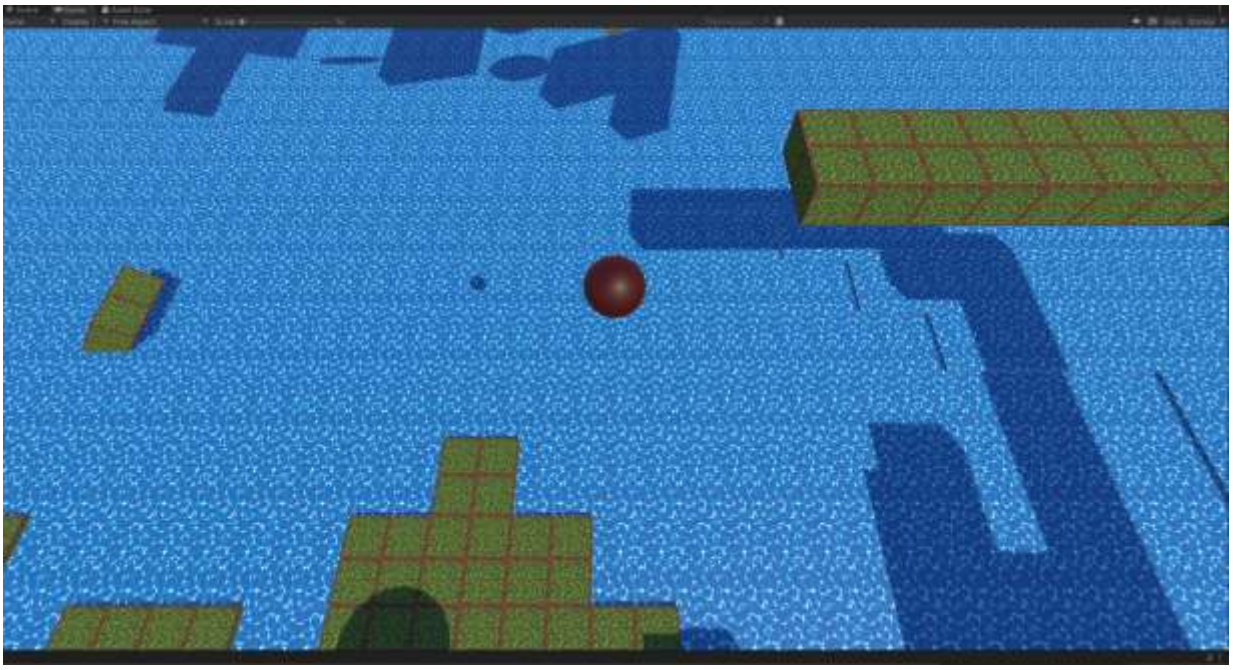


Рисунок 1.31. Процес тестування гри 3DBallz

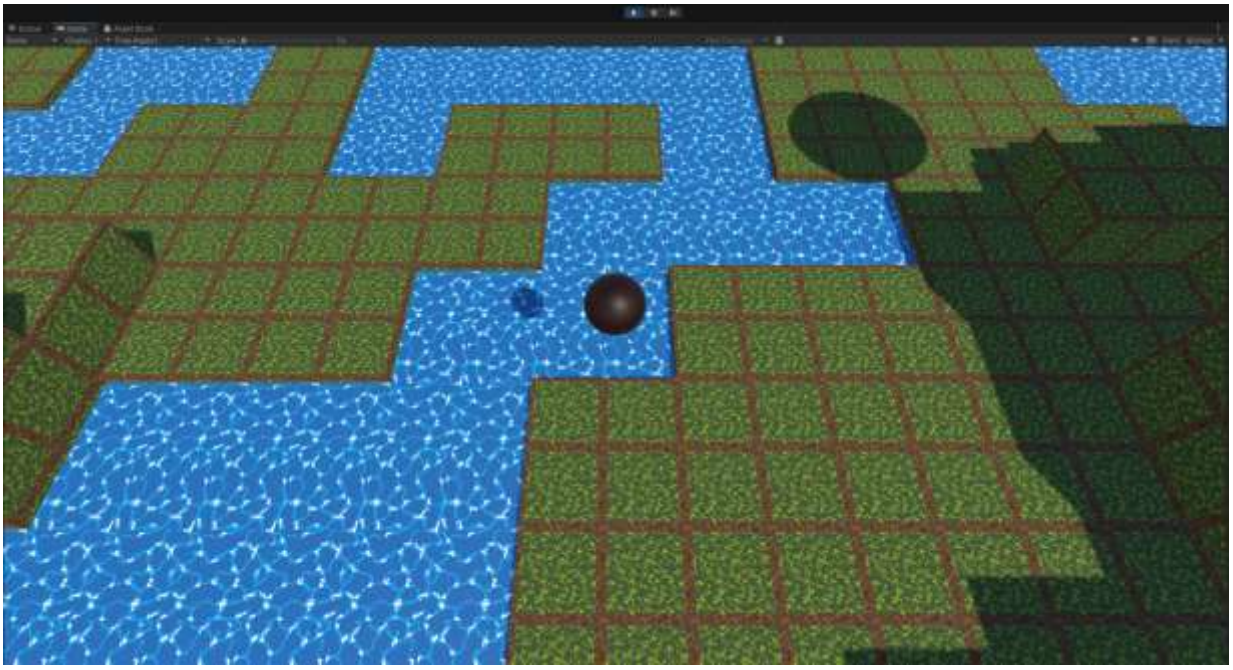


Рисунок 1.32. Процес тестування гри 3DBallz

Основною метою тестування є перевірка працездатності ігрових механік та управління сферою гравця. Також, цей процес відіграє важливу роль у балансуванні ігрового процесу та побудові карти рівня. Оскільки дає змогу швидко та оперативно тестувати ті, чи інші ігрові ситуації.

Серед знайдених помилок була неправильне отримання параметрів гравця при переході з рівня на рівень, що відбувалось через помилку у процесі отримування посилання на менеджер рівнів. Також неправильна реакція

поверхонь на сферу гравця, коли вона потрапляла до платформ, що було спричинено занадто невеликим колайдером тригера. Поверхня води досить часто чіпляла сферу гравця, через що можна було потрапити на початок рівня, навіть не впавши у воду, що довелося виправляти зменшенням колайдеру води. Найскладнішою помилкою була важко відслідковуєма помилка управління при перемиканні режимів роботи камери. Початково режими перемикались операторами if, але виявилось, що для мого випадку більш правильно працював оператор switch, через що довелося вносити виправлення у код слідкування камери за гравцем та код переміщення гравця.

Неодноразово проводились тестування рівня для знаходження більш цікавих ігрових ситуацій з точки зору розміщення платформ а також налаштування їх фізичних властивостей. В ході цих тестувань було концептуалізовано ще декілька видів платформ, які мали би схожі властивості з тими, які зараз є у проекті, але впливали би на фізичні параметри сфери гравця. Так, наприклад, можна створити брудні платформи, які будуть на деякий час зменшувати параметри швидкості переміщення та сили стрибку. Іншим варіантом, який можна розглянути – це платформи із динамічною стрибучістю, або такі, що будуть відбивати сферу гравця, підбрасуючи її у сторони.

Як результат реалізований проект був протестований та підтверджено його працездатність. Основні ігрові механіки працюють без помилок да дають гравцю можливість отримати новий для себе ігровий досвід.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Темою мого дипломного проекту була «Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity». За мету було поставлено реалізацію 3D-гри з використанням фізичних можливостей ігрового програмного рушія Unity. Ігровий процес відповідає жанру платформер та дає змогу проходити створений рівень, вирішуючи фізичні головоломки.

У цьому розділі проводиться розрахунок вартості розробленого програмного забезпечення. Рівень ефективності програмного забезпечення визначається як його якісними характеристиками, так і успішністю процесу розробки. Оцінка якості ПЗ здійснюється за кількома аспектами: з урахуванням думки користувачів, ефективності використання ресурсів та відповідності встановленим вимогам. Із погляду користувача, якість програмного продукту включає оцінювання витрат на його створення, зокрема трудових затрат і загальної вартості.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість створення програмного продукту визначається низкою чинників, серед яких – обсяг робіт, рівень трудомісткості, кваліфікація розробників, а також строки, що задаються ринковими умовами. Для оцінки обсягу програмного забезпечення використовується метод структурної аналогії, згідно з яким на основі спеціалізованих каталогів аналогів визначається кількість умовних машинних команд для подібного програмного продукту (у тисячах команд).

Таблиця 2.1. Каталог аналогів

Найменування ПП	*Обсяг функції ПП – V_o, усл. машинних командах.*
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП імітаційного моделювання	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткість визначаємо на основі табл.2.2.

Таблиця.2.2. Таблиця трудомісткості

Обсяг ПЗ, тис.умов.машинних команд	*Норма часу, люд/год*
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначаємо укрупнену норму часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПЗ, тобто в умовах комп'ютера, Кк=0,7÷0,8), для нашого варіанта виділено сірим кольором:

Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначаємо по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.).

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	*Ступінь новизни*		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	*Ступінь новизни*	*Значення $K_н$*
А	Принципово нове ПЗ	1,75 – 1,2
Б	ПЗ – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПЗ маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПЗ типовими програмами, %	*Значення $K_т$*
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором. Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$T_{тз}=Ta*L_1*K_н=183,2 *0,12*0,7= 15,38 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=Ta*L_2*K_н=183,2 *0,11*0,7 = 14,11 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$$T_{рп}=Ta*L_3*K_н*K_т=183,2 *0,61*0,7*0,6= 46,94 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=2 (стр), розробка ТП $N_{тп}$=28 (стр), розробка

робочого проекту $N_{рп}$=7 (стр), пояснювальна записка відповідно $N_{пз}$ 20 (стр). Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	*Розрахунок, годин.*		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}=0,15*2=0,3$
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}=0,7*28=19,6$	$T_{нк}=0,15*N_{тп}=0,15*28=4,2$
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}=0,7*N_{рп}=0,7*7=4,9$	$T_{нк}=0,15*N_{рп}=0,15*7=1,05$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$ = 1,5*20=30	$T_{кк}=0,7*N_{тз}=0,7*20=14$	$T_{нк}=0,15*N_{пз}=0,15*20$ =3
Усього, в т.ч.:	151,64		
- на розробку	$\Sigma T_р$=106,43		
- контроль керівника		$\Sigma T_{кк}$=39,9	
-нормоконтроль			$\Sigma T_{нк}$=5,31

2.3 Розрахунок ціни програмного продукту

Для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПЗ. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2025» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2025 року – 8000 гривень; мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	*Трудомісткість робіт, години*	*Погодинна тарифна ставка, грн.*	*Розрахунок, грн.*
1.Розробка ПП	106,43	48,00	5108,64
2.Контроль керівника	39,9	110,00	4389,00
3.Нормоконтроль	5,31	120,00	637,2
Усього	-	-	ΣЗо= 10134,84

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8:

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	*Тип, модель*	*Кількість*	*Ціна одиниці, грн.*	*Вартість, грн.*
Папір	Лист А4	57	5.0	285,00
Разом	-	-	-	$В_{м\,і}$ =285,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1 =$ 0,1*285=28,5
Усього				$В_м = В_{мі} + В_{тр\ з=}$ 313,5

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	313,5	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	10134,84	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1013,48	$Зд = 0{,}1 \times Зо$ = 10134,84*0,1
4.Відрахування до єдиного фонду соціального внеску	2452,63	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (10134,84+1013,48)
5. Накладні витрати	4053,93	$Внак. = 0{,}4 \times Зо$ =0,4 * 10134,84
6. Повна собівартість	17968,38	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 313,5+10134,84+1013,48+2452,63 +4053,93

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_{пов}*Р)/100=(17968,38*15)/100=2695,26 \text{ грн.} \quad (2.4)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о = С_{пов}+П = 17968,38+2695,26 = 20663,64 \text{ грн.} \quad (2.5)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного забезпечення становитиме:

$$Ц_р=Ц_о+ПДВ = 20663,64 + 20663,64 *0.2 = 24796,37 \text{ грн.} \quad (2.6)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Забезпечення безпеки життя та здоров'я громадян під час виконання ними трудових обов'язків, а також створення умов, що не шкодять здоров'ю, є одним із пріоритетних завдань держави. Кожне робоче місце повинно бути оснащене таким чином, щоб гарантувати зручність і безпеку співробітників. Наприклад, виробниче обладнання, обслуговування якого вимагає переміщення персоналу, слід обладнати надійними і зручними проходами, майданчиками, сходами та поручнями. Водночас експлуатація устаткування не повинна перевищувати встановлені норми викидів шкідливих речовин і створювати загрози пожеж або вибухів.

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста при розробці даного програмного комплексу

При розробці програмного комплексу проводиться ретельний аналіз можливого впливу виробничих чинників, які можуть зашкодити здоров'ю програміста. Згідно зі стандартом ГОСТ 12.1.003-74, до небезпечних факторів належать ті, що можуть спричинити раптове погіршення здоров'я або навіть летальний результат під час роботи.

Аналіз також враховує різні якісні характеристики робочого середовища — фізичні параметри приміщень, такі як температура, вологість, електричний опір підлоги, а також дані щодо концентрації іонів і забруднювачів у повітрі.

3.2 Гігієнічні вимоги до виробничого середовища

Сучасне виробництво вимагає створення оптимальних санітарних умов, які забезпечують плідну роботу співробітників без надмірного навантаження. Це досягається організацією комфортного робочого місця з чистим повітрям, правильною освітленістю, а також заходами щодо захисту від шумових та вібраційних впливів.

3.2.1 Мікроклімат

Недотримання норм мікроклімату негативно позначається на здоров'ї людини, що може призвести до зниження працездатності або її повної втрати.

Показники мікроклімату мають відповідати нормам, визначеним у ДСН 3.3.6.042-99. Згідно з чинними нормативними документами (ДСанПіН 3.3.2-007-98), у холодні періоди температура повітря повинна перебувати від –22 до +24°С, швидкість руху повітря – близько 0,1 м/с, а відносна вологість – у межах 40–60%.

В теплий сезон допустимі значення температури складають 23–25°С при збереженні вологості та швидкості повітря (0,1–0,2 м/с) на тих же рівнях. Підвищення кількості позитивних іонів у робочій зоні також може негативно впливати на здоров'я, тому оптимальний рівень аероіонізації вважається в межах від 150 до 5000 легких аерофонів на 1 см³.

Вплив на покращення складу робочого повітря здійснюється примусовою вентиляцією, застосуванням захисних екранів із заземленням або іонізаторів, а також можливістю регулювання основних параметрів мікроклімату.

3.2.2 Освітлення

Правильно організоване освітлення позитивно впливає на центральну нервову систему, сприяє зниженню енергетичних витрат організму під час виконання завдань і покращує продуктивність праці. Надмірне або недостатнє освітлення може призводити до перенапруження зору, втоми, зниження швидкості робочих процесів та, з часом, до розвитку захворювань очей, таких як короткозорість.

Тому освітлення робочих приміщень має відповідати нормам СниП II.4-79. Забезпечення рівномірного світлового потоку досягається за допомогою відбитого або розсіяного світла, яке слід поєднувати з природним освітленням, дотримуючись нормативного рівня в 300–500 лк. При цьому необхідно уникати відблисків від клавіатури, екрана та інших пристроїв, спрямованих у бік очей користувача.

3.2.3 Шум

Деякі пристрої, що працюють з ВДТ, можуть стати джерелами різних звукових коливань – як у чутному, так і в ультразвуковому діапазоні. Постійний або тривалий вплив такого шуму спричиняє зниження працездатності,

погіршення концентрації, збільшення кількості помилок, зорову втому, зміну сприйняття кольорів та появу головного болю.

Нормативним показником для робочого місця є рівень шуму до 50 дБ, а для його зниження слід застосовувати заходи, як-от усунення причин шуму на етапі проектування, використання звукопоглинаючих матеріалів та оптимізація планування виробничих приміщень.

3.2.4 Вимоги до організації робочого місця працівника

Під час виконання паяльних робіт потрібно суворо дотримуватися норм організації робочого місця. Кожен елемент робочої зони має бути розташований так, щоб забезпечити максимальний комфорт і безпеку, усуваючи зайві предмети, які можуть створювати перешкоди.

Паяльне обладнання, робочі інструменти і деталі, а також засоби індивідуального захисту повинні перебувати у справному стані та відповідати стандартам охорони праці. Паяльні роботи проводяться з використанням електропаяльника, який живиться від мережі 220 В і має споживання не більше 100 Вт. Використання кислот або рідин на основі кислотних розчинів суворо заборонено.

Під час ремонтних робіт обладнання повинно бути повністю відключене від електроживлення (штк. вилка вилучається з розетки), а всі доступні елементи – ізольовані від мережі. Через застосування різних припоїв і флюсів, що містять шкідливі компоненти (свинець, цинк, літій, калій, натрій, кадмій тощо), робочі місця паяльників повинні бути обладнані додатковими локальними витяжними системами.

3.2.5 Електробезпека

Для запобігання ураженню електричним струмом необхідно чітко дотримуватися правил безпечного виконання робіт і експлуатації техніки. Оператор повинен бути захищений від доступу до частин обладнання, що працюють під високою напругою, а також до неізольованих елементів, які не

підключені до захисного заземлення. Електроживлення комп'ютерної техніки має підключатися виключно через спеціальні штекери із заземленням.

3.3 Пожежна безпека

До систем гасіння пожеж належать як внутрішні пожежні водопроводи (крани-ПК), так і різні типи вогнегасників, зокрема вуглекислотні, порошкові, а також сухий пісок. У будівлях пожежні крани розташовують у коридорах та на сходових майданчиках; кожен кран комплектується пожежним рукавом і встановлюється у спеціальних ящиках, розташованих на певній висоті від підлоги.

На початкових стадіях пожеж застосовують вогнегасники, найбільш ефективними серед яких є вуглекислотні пристрої, що дозволяють не лише гасити загоряння, але й зберігати електрообладнання. Такі засоби мають бути розміщені у легкодоступних місцях, на відповідній висоті від підлоги. Крім того, виробничі приміщення повинні мати запасні виходи, де двері мають бути позначені освітленим написом «Запасний вихід», а схема евакуації – розміщена біля основного виходу.

До засобів гасіння пожежі відносяться внутрішні пожежні водопроводи (крани - ПК), вогнегасники (вуглекислотні та порошкові), сухий пісок тощо.

В будівлях пожежні крани встановлюють в коридорах, на майданчиках сходових кліток. Кожний пожежний кран укомплектований пожежним рукавом і розміщений у відповідних ящиках, які знаходяться на висоті 1,35 м від полу.

Пожежна безпека є комплексною системою заходів, спрямованих на запобігання займання, своєчасне виявлення пожежі та ефективне ліквідування загоряння. До основних засобів цього захисту відносяться як внутрішні пожежні водопроводи (крани-ПК), так і різноманітні типи вогнегасників: вуглекислотні, порошкові, а також засоби за основою сухого піску. У будівлях пожежні крани зазвичай розташовують у коридорах та на сходових майданчиках; кожен кран комплектується пожежним рукавом і встановлюється в спеціально обладнаних ящиках, оптимально розміщених на висоті приблизно 1,35 м від підлоги. Таке

розташування забезпечує легкий доступ до засобів гасіння у разі надзвичайної ситуації.

На початкових стадіях загоряння застосовують вогнегасники, найбільш ефективними з яких є вуглекислотні пристрої. Вони дозволяють не лише швидко знищити загоряння, але й мінімізувати можливі пошкодження електрообладнання, що особливо важливо у виробничих приміщеннях. Водночас, стандартними засобами гасіння пожежі можуть виступати і порошкові вогнегасники, а також сухий пісок, які використовуються для локалізації загоряння до прибуття аварійних служб. Застосування цих пристроїв обов'язково повинно відповідати нормам безпеки та бути розміщено у відкритих, легкодоступних місцях.

Окрім технічних засобів, надзвичайне значення має організаційний аспект пожежної безпеки. Виробничі та громадські приміщення повинні бути забезпечені запасними виходами. Двері цих виходів повинні бути позначені яскравим, освітленим знаком «Запасний вихід», а поблизу – розміщеним планом евакуації, який чітко окреслює маршрути безпечного виходу з приміщення. Регулярне проведення тренувань з евакуації серед персоналу дозволяє зменшити ризик панічної поведінки та забезпечує оперативність реагування у випадку загоряння.

Сучасні технології протипожежного захисту передбачають встановлення систем автоматичного пожежного оповіщення та сигналізації. Детектори диму, теплові сенсори, а іноді й газоаналізатори, дозволяють оперативно виявити загоряння та розпочати автоматичну активацію систем гасіння (наприклад, систем розпилення води або пінних розчинів). Такі системи значно скорочують час реагування і сприяють оперативному інформуванню всіх осіб, що знаходяться у приміщенні, що є критично важливим для збереження життя та мінімізації матеріальних збитків.

ВИСНОВКИ

Темою мого дипломного проекту була «Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity». За мету було поставлено реалізацію 3D-гри з використанням фізичних можливостей ігрового програмного рушія Unity. Під час виконання дипломного проектування, для виконання коректного процесу розробки, було виконано аналіз існуючих ігрових рішень з якого було отримано оптимальний ігровий жанр для розробляємої гри.

Після обрання програмного забезпечення для розробки – Visual Studio та Adobe Photoshop – було виконано проектування основних ігрових елементів, а саме схеми роботи ігрових сцен, ігрових об’єктів, спроектовано зв’язки взаємодії між модулями гри. Окремо було виконано проектування використання фізичних властивостей ігрового рушія Unity для реалізації теми дипломного проектування.

Процес реалізації почався зі створення проекту гри, його початкового налаштування, створення матеріалів та спрайтів для наповнення графічної частини гри. Також було виконано створення ігрового об’єкту гравця, який міг би виконувати всі ігрові механіки, які були продиктовані ігровим концептом. Однією з важливих частин розробки було налаштування інструментарію для створення ігрових рівнів. Для цього використовувались Tile Palette та додатковий модуль, який розширював його функціонал. Написано код для ігрових об’єктів.

Окремо виділено реалізацію фізичної складової гри, а саме написання коду для переміщення гравця та робота з камерою. Створені фізичні матеріали для симуляції різних поверхонь, а також налаштовано префаби для активних платформ. Всі створені ігрові об’єкти та префаби були розміщені на рівні та налаштовані. Створено ігрове меню та менеджер ігрових рівнів для передачі даних між сценами гри.

В результаті було розроблено та реалізовано 3D-гру на ігровому програмному рушії Unity з використанням можливостей його фізичного ядра. Ігровий процес відповідає жанру платформер та дає змогу проходити створений рівень, вирішуючи фізичні головоломки. Реалізовано головне меню, а також декілька рівнів, перехід між ними без втрати параметрів гравця.

У подальшому розроблену гру можна покращувати. Першим, що можна покращити є графічна складова гри, яка в поточному варіанті реалізована з примітивів та з використанням спрайтів, а не повноцінних текстур. Також ігровий процес можна дещо покращити додаванням нових активних платформ, що дасть змогу створювати більш цікаві ігрові ситуації.

Створений модульний базис гри дає широкі можливості для подальшої підтримки проекту, від додавання нового графічного матеріалу, до створення нових ігрових механік, об’єктів для переміщення по рівню, або зовсім переглянути концепцію гри, перемістивши її у безповітряне середовище.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Ляшенко О.М., Електронний навчальний посібник «Розробка комп'ютерних ігор за допомогою Unity 3D», Херсон: видавництво ФОП Вишемирський В.С., 2018.;
2. Джозеф Хокінг Unity в дії / Джозеф Хокінг., 2018. – 335 с.;
3. Дейбук В.Г. Віртуальний електронний практикум: Навчальний посібник – Чернівці: Чернівецький нац. ун-т, 2021. – 188 с.;
4. Трофименко О.Г. С++. Алгоритмізація та програмування: підручник / О.Г. Трофименко, Ю.В. Прокоп, Н.І. Логінова, О.В. Задерейко. 2-ге вид. перероб. і доповн. – Одеса : Фенікс, 2019. – 477 с.;
5. Джон Менінг Unity для розробників. Мобільні мультиплатформені ігри / Джон Менінг, Періс Батфилд-Еддісон., 2018. – 352 с.;
6. Мосіюк О.О., Федорчук А.Л. Операційні системи та системне програмування: навчально-методичний посібник. Житомир: Вид-во ЖДУ ім. Івана Франка, 2022. – 76 с.;
7. Stroustrup B. A Tour of C++ (Second Edition). – Addison-Wesley, 2018. – 240 p. – ISBN 978-0-13-499783-4;
8. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms – Boston., McGraw-Hill, 2001. – 1082p.;
9. Robert Sedgewick, Kevin Wayne. Algorithms - Addison-Wesley, 2020. – 956 p. – ISBN 978-0321573513;
10. Procedural Generation in Game Design / Тарн Адамс, 2017. – 336 с.;
11. Бібліотека MSDN [Електронний ресурс]. – Режим доступу: URL http://msdn.microsoft.com/ru-ru/library/default.aspx (дата звернення 20.05.2025);
12. Бібліотека Unity [Електронний ресурс]. – Режим доступу: URL https://docs.unity.com/ (дата звернення 20.05.2025).

ДОДАТОК А. Лістинг основних модулів гри мовою C#

Скрипт Player_parameters.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Player_parameters : MonoBehaviour
{
    MeshRenderer player_mr;
    public float player_energy = 100f;
    public float maximum_player_energy = 100f;
    public float player_jump_force = 200f;
    public float energy_to_jump = 0f;
    void Start()
    {
        if (GameManager.Instance != null)
        {
            maximum_player_energy = GameManager.Instance.ini_maximum_player_energy;
            player_jump_force = GameManager.Instance.ini_player_jump_force;
        }
        else
        {
            maximum_player_energy = 100f;
            player_jump_force = 200f;
        }
        player_mr = GetComponent<MeshRenderer>();
        player_energy = maximum_player_energy;
        energy_to_jump = player_jump_force / 10f;
    }
    public void set_maximum_player_energy(float new_max_player_energy)
    {
        maximum_player_energy = new_max_player_energy;
    }
    public void set_player_jump_force(float new_jump_force)
    {
        player_jump_force = new_jump_force;
    }
    public void change_player_energy(float amount)
    {
        if(player_energy > 0)
        {
            player_energy += amount;
            if (player_energy < 0)
                player_energy = 0;
            player_mr.material.color = new Color(player_energy / 100f, player_mr.
material.color.g, player_mr.material.color.b);
        }
    }
    public void regen_player_energy(float rate_amount)
    {
        if (player_energy < maximum_player_energy)
        {
            player_energy += rate_amount;
            player_mr.material.color = new Color(player_energy / 100f, player_mr.
material.color.g, player_mr.material.color.b);
            if(player_energy > maximum_player_energy)
                player_energy = maximum_player_energy;
        }
```

```
    }
}
```

Скрипт Player_movement.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Player_movement : MonoBehaviour
{
    [SerializeField] Player_folower _player_folower;
    [SerializeField] Player_parameters _player_parameters;
    Rigidbody player_rb;
    void Start()
    {
        player_rb = GetComponent<Rigidbody>();
    }
    void Update()
    {
        if (_player_parameters.player_energy > _player_parameters.energy_to_jump)
        {
            if (Input.GetKeyDown(KeyCode.Space))
            {
                player_rb.AddForce(Vector3.up   *   _player_parameters.player_jump_
force);
                _player_parameters.change_player_energy(-(_player_parameters.
energy_to_jump));
            }
        }
        if(Input.GetKeyDown(KeyCode.Escape))
        {
            Application.Quit();
        }
    }
    private void FixedUpdate()
    {
        switch (_player_folower.get_camera_mode())
        {
            case 0:
                {
                    if (Input.GetAxis("Horizontal") > 0f)
                    {
                        player_rb.AddForce(Vector3.right * 5f);
                    }
                    if (Input.GetAxis("Horizontal") < 0f)
                    {
                        player_rb.AddForce(Vector3.left * 5f);
                    }
                    if (Input.GetAxis("Vertical") > 0f)
                    {
                        player_rb.AddForce(Vector3.forward * 5f);
                    }
                    if (Input.GetAxis("Vertical") < 0f)
                    {
                        player_rb.AddForce(Vector3.back * 5f);
                    }
                    break;
                }
            case 1:
```

```
    {
        if (Input.GetAxis("Horizontal") > 0f)
        {
            player_rb.AddForce(Vector3.forward * 5f);
        }
        if (Input.GetAxis("Horizontal") < 0f)
        {
            player_rb.AddForce(Vector3.back * 5f);
        }
        if (Input.GetAxis("Vertical") > 0f)
        {
            player_rb.AddForce(Vector3.left * 5f);
        }
        if (Input.GetAxis("Vertical") < 0f)
        {
            player_rb.AddForce(Vector3.right * 5f);
        }
        break;
    }
case 2:
    {
        if (Input.GetAxis("Horizontal") > 0f)
        {
            player_rb.AddForce(Vector3.left * 5f);
        }
        if (Input.GetAxis("Horizontal") < 0f)
        {
            player_rb.AddForce(Vector3.right * 5f);
        }
        if (Input.GetAxis("Vertical") > 0f)
        {
            player_rb.AddForce(Vector3.back * 5f);
        }
        if (Input.GetAxis("Vertical") < 0f)
        {
            player_rb.AddForce(Vector3.forward * 5f);
        }
        break;
    }
case 3:
    {
        if (Input.GetAxis("Horizontal") > 0f)
        {
            player_rb.AddForce(Vector3.back * 5f);
        }
        if (Input.GetAxis("Horizontal") < 0f)
        {
            player_rb.AddForce(Vector3.forward * 5f);
        }
        if (Input.GetAxis("Vertical") > 0f)
        {
            player_rb.AddForce(Vector3.right * 5f);
        }
        if (Input.GetAxis("Vertical") < 0f)
        {
            player_rb.AddForce(Vector3.left * 5f);
        }
        break;
    }
```

```
            }
            _player_parameters.regen_player_energy(0.1f);
        }
    }
```

Скрипт Player_folower.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Player_folower : MonoBehaviour
{
    [SerializeField] GameObject player_game_object;
    short camera_mode = 0;
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            if (camera_mode < 3)
                camera_mode++;
            else if (camera_mode == 3)
                camera_mode = 0;
        }
        else if (Input.GetKeyDown(KeyCode.Q))
        {
            if (camera_mode > 0)
                camera_mode--;
            else if (camera_mode == 0)
                camera_mode = 3;
        }
        switch (camera_mode)
        {
            case 0://view on +z
                {
                    transform.position    =    new    Vector3(player_game_object.
transform.position.x, player_game_object.transform.position.y + 7,
                    player_game_object.transform.position.z - 5);
                    transform.rotation = Quaternion.Euler(60, 0, 0);
                    break;
                }
            case 1://view on -x
                {
                    transform.position                    =                    new
Vector3(player_game_object.transform.position.x+5,         player_game_object.transform.
position.y + 7,
                    player_game_object.transform.position.z);
                    transform.rotation = Quaternion.Euler(60, -90, 0);
                    break;
                }
            case 2://view on -z
                {
                    transform.position = new Vector3(player_game_object.transform.
position.x, player_game_object.transform.position.y + 7,
                    player_game_object.transform.position.z + 5);
                    transform.rotation = Quaternion.Euler(60, -180, 0);
                    break;
                }
            case 3://view on +x
                {
```

```
                        transform.position = new Vector3(player_game_object.transform.
position.x - 5, player_game_object.transform.position.y + 7,
                        player_game_object.transform.position.z);
                        transform.rotation = Quaternion.Euler(60, -270, 0);
                        break;
                    }
            }
        }
        public short get_camera_mode()
        {
            return camera_mode;
        }
    }
```

Скрипт Reduce_platform_logic.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Reduce_platform_logic : MonoBehaviour
{
    [SerializeField] float platform_reduce_rait = -1f;
    Player_parameters _player_parameters;
    private void OnTriggerStay(Collider other)
    {
        _player_parameters = other.GetComponent<Player_parameters>();
        if (_player_parameters != null){
            _player_parameters.change_player_energy(platform_reduce_rait);
        }
    }
}
```

Скрипт Regen_platform_logic.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Regen_platform_logic : MonoBehaviour
{
    [SerializeField] float platform_regen_rait = 1f;
    Player_parameters _player_parameters;
    private void OnTriggerStay(Collider other)
    {
        _player_parameters = other.GetComponent<Player_parameters>();
        if(_player_parameters != null )
        {
            _player_parameters.regen_player_energy(platform_regen_rait);
        }
    }
}
```

Скрипт Exit_platform_logic.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class Exit_platform_logic : MonoBehaviour
{
    Player_parameters _player_parameters;
```

```
    private void OnTriggerStay(Collider other)
    {
        _player_parameters = other.GetComponent<Player_parameters>();
        if (_player_parameters != null)
        {
            SceneManager.LoadScene("Second_level");
        }
    }
}
```

Скрипт Water_logic.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class Water_logic : MonoBehaviour
{
    Player_parameters _player_parameters;
    private void OnTriggerEnter(Collider other)
    {
        _player_parameters = other.GetComponent<Player_parameters>();
        if (_player_parameters != null)
        {
            SceneManager.LoadScene("First_level");
        }
    }
}
```

Скрипт GameManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class GameManager : MonoBehaviour
{
    public static GameManager Instance;
    public float ini_maximum_player_energy = 0f;
    public float ini_player_jump_force = 0f;
    private void Awake()
    {
        if(Instance != null)
        {
            Destroy(gameObject);
            return;
        }
        Instance = this;
        DontDestroyOnLoad(gameObject);
    }
    public void set_easy_mode()
    {
        ini_maximum_player_energy = 300f;
        ini_player_jump_force = 400f;
    }
    public void set_medium_mode()
    {
        ini_maximum_player_energy = 200f;
        ini_player_jump_force = 300f;
    }
    public void set_hard_mode()
```

```
    {
        ini_maximum_player_energy = 100f;
        ini_player_jump_force = 200f;
    }
}
```

Скрипт Main_menu_button_logic.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class Main_menu_button_logic : MonoBehaviour
{
    [SerializeField] GameObject Main_menu_panel;
    [SerializeField] GameObject Setting_panel;
    [SerializeField] GameObject Help_panel;
    public void start_first_level()
    {
        SceneManager.LoadScene("First_level");
    }
    public void exit_game(){
        Application.Quit();
    }
    public void show_main_menu(){
        Main_menu_panel.SetActive(true);
    }
    public void hide_main_menu(){
        Main_menu_panel.SetActive(false);
    }
    public void show_settings(){
        Setting_panel.SetActive(true);
    }
    public void hide_settings(){
        Setting_panel.SetActive(false);
    }
    public void show_help()
    {
        Help_panel.SetActive(true);
    }
    public void hide_help()
    {
        Help_panel.SetActive(false);
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity

ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА ГРУПИ 4КГ-08: ЗАЛЕСЬКОГО КИРИЛА ВЯЧЕСЛАВОВИЧА

КЕРІВНИК: ДЖАБРАІЛОВ Д.В.

Аналіз існуючих ігрових рішень та обрання жанру

Скріншот ігрового процесу комп'ютерної гри Ballance

Скріншот ігрового процесу комп'ютерної гри Vertigo

Скріншот ігрового процесу комп'ютерної гри Rock of Ages III

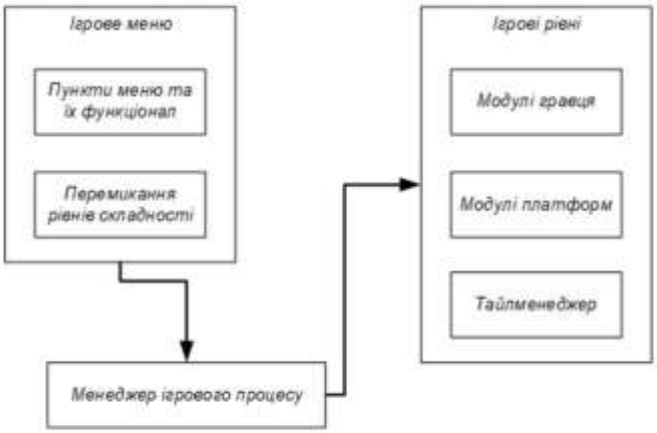
Обрані засоби розробки

Unity

Visual Studio

Ps

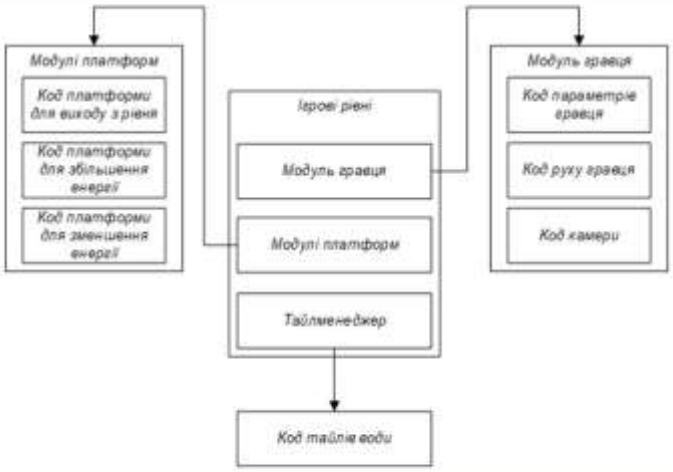
Структура основних ігрових елементів



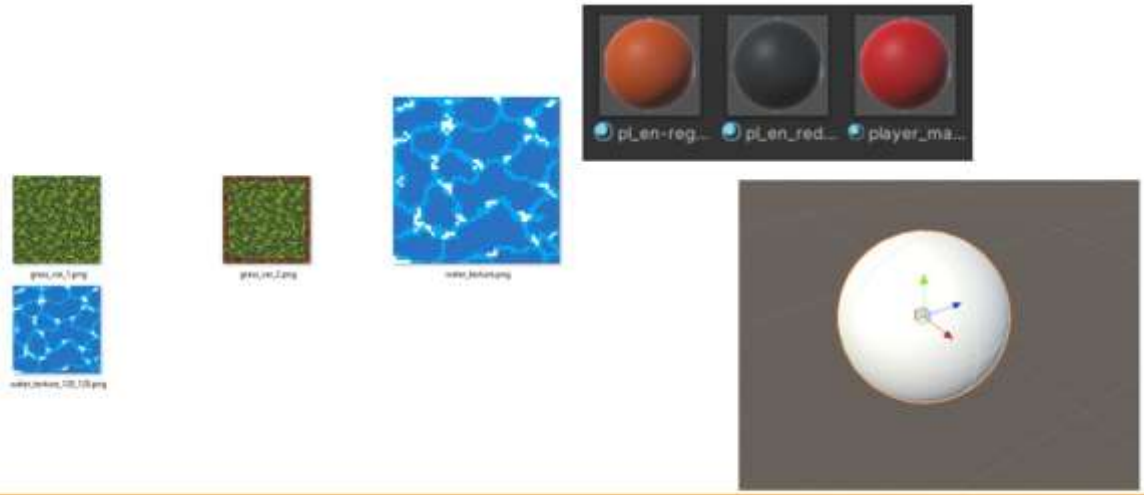
Схематичне зображення структури гри 3DBallz

Схема роботи ігрового меню гри 3DBallz

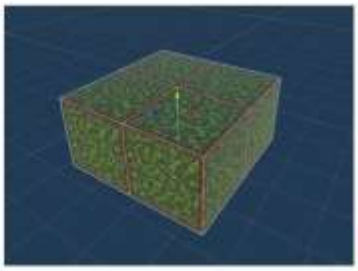
Структура модулів ігрових рівнів гри 3DBallz



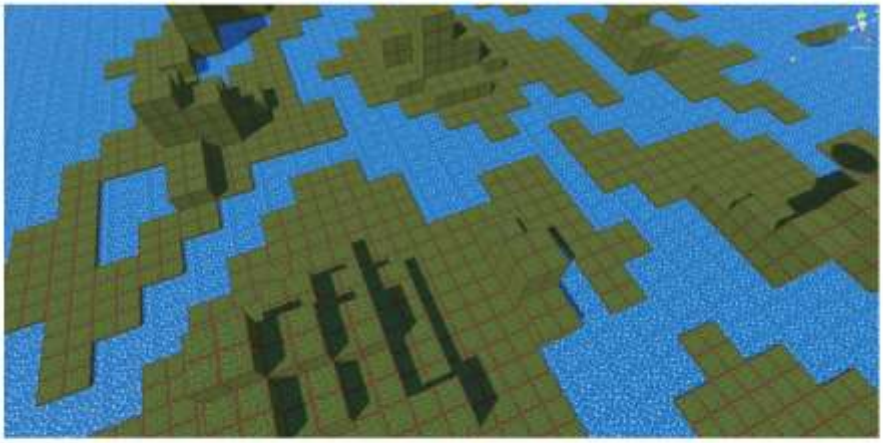
Створені спрайти для гри 3DBallz



Результат створення рівня за допомогою GameObject у Tile Palette для гри 3DBallz



Префаб тайлу землі



Реалізація основних ігрових елементів

Параметри гравця та його методи

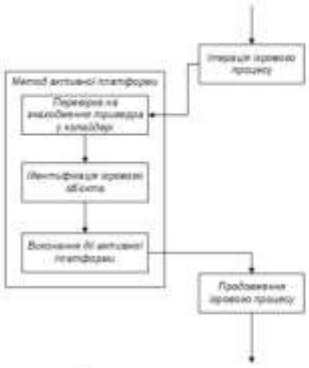
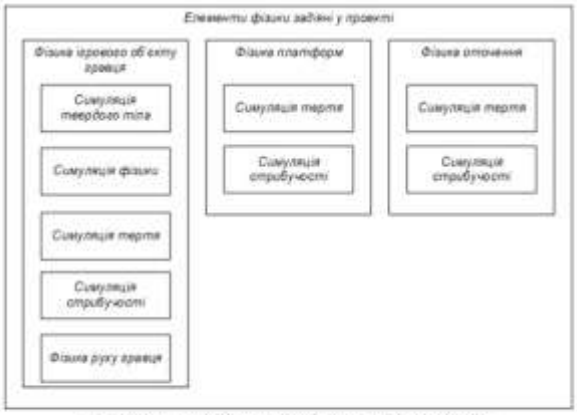


Схема роботи коду активних платформ

Використання фізики рушія в ігровому процесі



Елементи фізики задіяні у грі 3DBallz

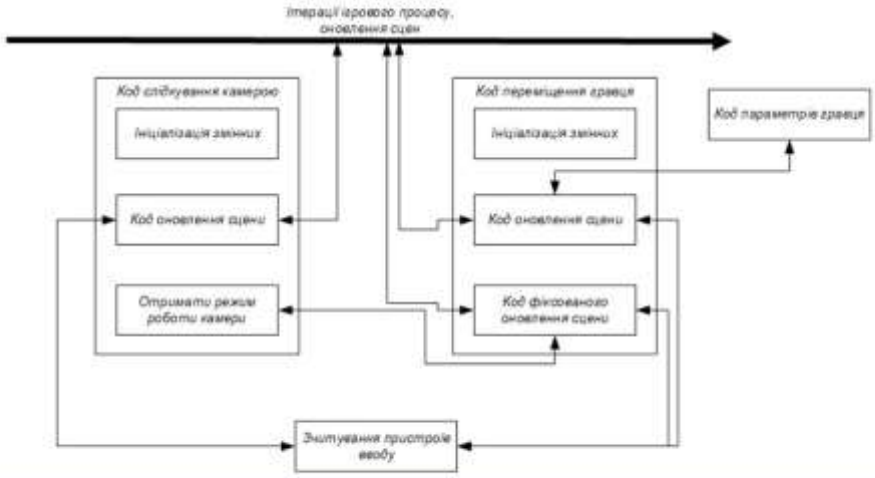
Робота з фізичними матеріалами та компонентами

Властивості фізичного матеріалу за замовченням

Фізичні матеріали для різних поверхонь гри 3DBallz

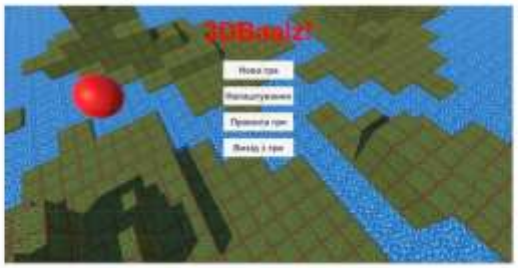
Відредаговані компоненти платформи для регенерації енергії гравця

Схема роботи коду Коду слідкування камерою та Коду переміщення гравця для гри 3DBallz



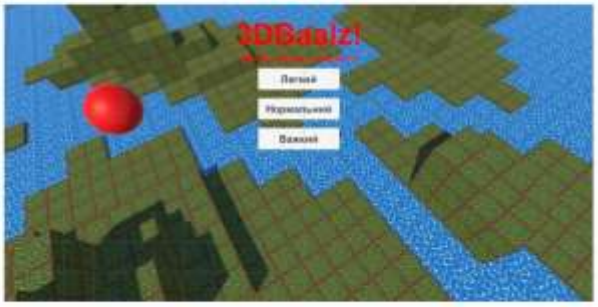
Реалізація ігрового інтерфейсу та меню гри

Структура інтерфейсу головного меню гри у ієрархії ігрових об'єктів сцени Main_menu



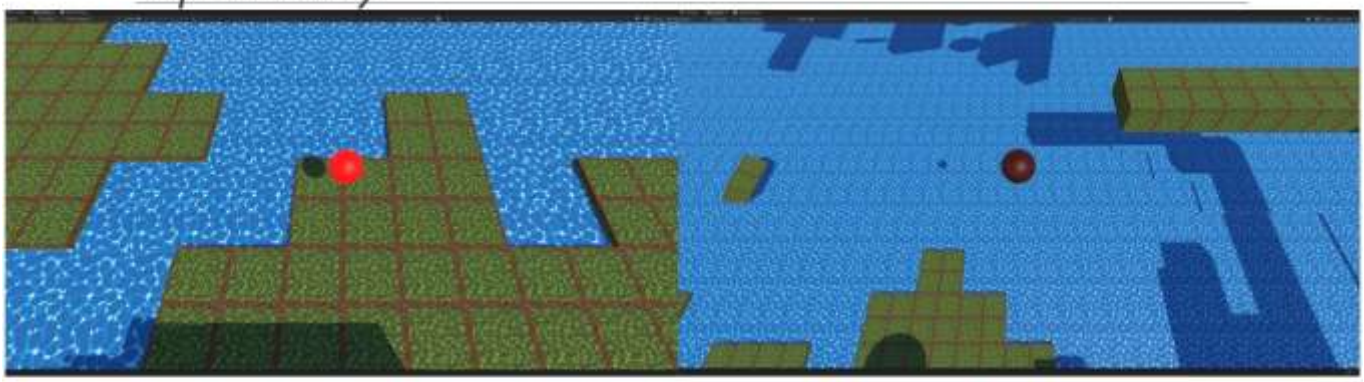
Панель головного меню гри 3DBallz

Панель обрання складності гри 3DBallz

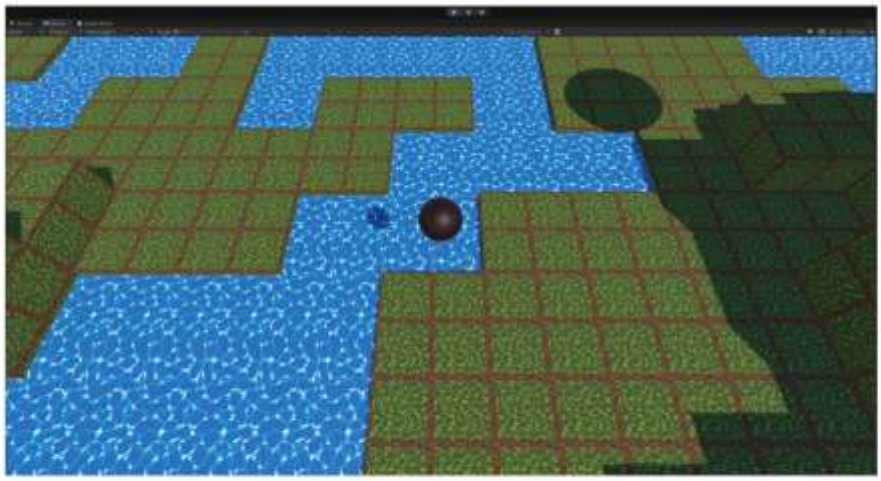


Панель Правил гри 3DBallz

Тестування гри та скріншоти проекту



Тестування гри та скріншоти проекту



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Залеський Кирило В'ячеславович

(прізвище, ім`я та по батькові)

Спеціальність *123 " Комп'ютерна інженерія "*

Освітня програма *«Комп'ютерна графіка і Web-дизайн»*

Керівник дипломного проекту (роботи) *Джабраілов Дмитро Володимирович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity*

Обсяг розрахунково-пояснювальної записки *84* сторінок

Обсяг графічної (презентаційної) частини *16* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню *Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням створення ігрових продуктів в цілому та розробці ігор на ігровому програмному рушії Unity зокрема.*

б) характеристика виконання кожного розділу дипломного проекту (роботи) *Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У основному розділі розглянуті питання проблематики розробки гри на ігровому програмному рушії Unity, сформовано концепцію гри згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів розробляємої гри. За допомогою відповідного програмного забезпечення реалізовані всі намічені роботи з ігровим процесом.*

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) *Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено*

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально описано процес виконання розробки гри;

2. Виконано проектування елементів гри із поясненнями на схемах та за допомогою коду;

3. Використано незвичайний підхід до ігрового інтерфейсу.

д) основні недоліки дипломного проекту (роботи)

Використання 4×4-клітинних груп без віджету Tilemap нав'язує ручне викладання шаром кубів, а не модульні 3D-блоки; Швидкість=5 у всіх напрямах, сила стрибка й енерговитрати фіксовані, без залежності від маси чи тертя; не показано профайлинг рушія; Базова графіка/арти: лише примітивні куби й спрайти; не вистачає власного стилю

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

20 червня

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проєкт здобувача (здобувачки) освіти

відділення комп'ютерних систем

Залеський Кирило Вячеславович

(прізвище, ім`я та по батькові)

Спеціальність: *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма: *«Комп'ютерна графіка і Web-дизайн»*

Тема дипломного проєкту: *Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ

а) обсяг і якість виконання проєкту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проєкт виконано відповідно технічному завданню. Пояснювальна записка містить 84 сторінки. У пояснювальній записці виконано опис етапів розробки комп'ютерної гри для ОС Windows, від розробки ігрової концепції, проектування гри, до реалізації елементів гри. Графічна частина складається з 16 слайдів мультимедійної презентації, які також містять графічні матеріали, передбачені технічним завданням. Якість виконання пояснювальної записки відмінна, графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проєктом: *Протягом всього строку дипломного проєктування та переддипломної практики здобувач освіти Залеський К.В. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Залеський К.В. під час роботи над дипломним проєктом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проєкту

г) вміння розв'язувати виробничі та конструкторські питання _____

Під час дипломного проєктування здобувач освіти Залеський К.В. мав змогу самостійно приймати рішення з реалізації елементів і модулів гри, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проєкту _____

Джабраілов Дмитро Володимирович

Місце роботи і посада керівника дипломного проєкту _____

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії

Підпис _____

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Залеський К.В.
здобувач освіти гр. 4КГ-08, та

Джабраілов Д.В.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity» (автор роботи – Залеський К.В., керівник роботи – Джабраілов Д.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Залеський К.В. /

Керівник _____ / Джабраілов Д.В. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КГ-08

Залеського Кирила Вячеславовича

на тему *Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 13.06.2025 р. значення коефіцієнту подібності в роботі становить 9,66%, коефіцієнт цитування – 1,13%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Залеського К.В.* (П.І.Б.)

проведена « *16* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/13/2025
Дата редагування 6/13/2025

Документ прийнятий

Звіт подібності

метадані

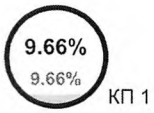
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка 3D-гри 3DBaalz з використанням фізичного ядра програмного рушія Unity

Автор
Залеський Кирило Вячеславович
Науковий керівник / Експерт
Джабраілов Дмитро Володимирович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КЦ

25
Довжина фрази для коефіцієнта подібності 2

13937
Кількість слів

106730
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	16
Інтервали	A⟷	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	58

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	146 1.05 %
2	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	51 0.37 %
3	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	51 0.37 %
4	https://card-file.ontu.edu.ua/bitstreams/63ee88cb-a3d0-4005-9cf2-0cff89f28c0d/download	45 0.32 %
5	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	41 0.29 %

6	https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download	31 0.22 %
7	https://card-file.ontu.edu.ua/bitstreams/9908b7a9-6b3e-46f5-a46e-84d83787cfd4/download	31 0.22 %
8	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	31 0.22 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	29 0.21 %
10	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	28 0.20 %

з домашньої бази даних (0.33 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	25 (3) 0.18 %
2	Розробка 3D-гри у жанрі survival-horror з налаштуваннями рівнів складності [redacted] 6/12/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	21 (2) 0.15 %

з програми обміну базами даних (0.17 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка VR гри в середовищі Unity Engine [redacted] 6/9/2023 National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	14 (2) 0.10 %
2	Методичні рекомендації.docx [redacted] 1/12/2023 Zhytomyr Ivan Franko State University (ZIFSU)	10 (1) 0.07 %

з Інтернету (9.16 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**199 (5) 1.43 %**
2	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**194 (9) 1.39 %**
3	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**120 (5) 0.86 %**
4	https://ir.nmu.org.ua/bitstream/handle/123456789/161848/122-18-2_%D0%92%D0%BE%D0%BB%D0%BE%D1%88%D0%B8%D0%BD%D0%B0%20%D0%92.%D0%92.pdf?sequence=1	115 (12) 0.83 %
5	**https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download**	**111 (8) 0.80 %**
6	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**70 (4) 0.50 %**
7	**https://card-file.ontu.edu.ua/bitstreams/9908b7a9-6b3e-46f5-a46e-84d83787cfd4/download**	**48 (2) 0.34 %**
8	**https://card-file.ontu.edu.ua/bitstreams/63ee88cb-a3d0-4005-9cf2-0cff89f28c0d/download**	**45 (1) 0.32 %**

9	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	36 (2) 0.26 %
10	https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download	31 (1) 0.22 %
11	https://essuir.sumdu.edu.ua/bitstream-download/123456789/92986/1/Sholoh_bac_rob.pdf	30 (3) 0.22 %
12	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	30 (2) 0.22 %
13	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	27 (3) 0.19 %
14	http://www.matmodel.puet.edu.ua/files/lic2020/rp-ossp-24.pdf	25 (1) 0.18 %
15	https://card-file.ontu.edu.ua/bitstreams/158e44b0-583e-4b2d-b758-6b86979e33bb/download	23 (1) 0.17 %
16	https://ep3.nuwm.edu.ua/18432/1/06-08-16.pdf	22 (2) 0.16 %
17	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	22 (2) 0.16 %
18	https://core.ac.uk/download/pdf/286084798.pdf	19 (2) 0.14 %
19	https://essuir.sumdu.edu.ua/bitstream/123456789/91990/1/Hruzdo_bak_rob.pdf	18 (1) 0.13 %
20	https://knute.edu.ua/file/NjI4Nw==/354eadce03a1a6cfc898e26acceeea0d.pdf	18 (1) 0.13 %
21	https://card-file.ontu.edu.ua/server/api/core/bitstreams/f5042058-9544-4ac7-bd33-42bd733c8e6b/content	16 (1) 0.11 %
22	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	12 (1) 0.09 %
23	https://card-file.ontu.edu.ua/bitstreams/aed610a6-43ef-47e0-9066-e85c89456f3e/download	11 (1) 0.08 %
24	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	10 (1) 0.07 %
25	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	9 (1) 0.06 %
26	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	8 (1) 0.06 %
27	https://card-file.ontu.edu.ua/server/api/core/bitstreams/6e367988-fd72-47a3-ac3d-c5748c863588/content	8 (1) 0.06 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн» **Група:** 4КГ- 08

Дипломний **проєкт здобувача освіти денної форми навчання КГ.** 08.09.000.ДП

ЗАЛЕСЬКОГО
КИРИЛА ВЯЧЕСЛАВОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн»
Група: 4 КГ- 08