

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»

за спеціальністю 122 «Комп'ютерні науки»

(освітня програма «Комп'ютерна графіка та розробка ігор»)

на тему:

**«3D гра жанру шутер засобами Unity під Windows-
платформу»**

Виконала студентка групи ЗКН-19-1
МАЧУШНИК Олена Олексіївна

Керівник роботи:
ЛЕВКІВСЬКИЙ Віталій Леонідович

Рецензент:
КОРОТУН Ольга Володимирівна

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **МАЧУШНИК Олена Олексіївна**

Керівник роботи: **ЛЕВКІВСЬКИЙ Віталій Леонідович**

Тема роботи: **«3D гра жанру шутер засобами Unity під Windows-платформу»**,
затверджена наказом закладу вищої освіти від **«05» січня 2023 р., №11/с**

Термін здачі закінченої роботи: **«10» червня 2023 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): об'єктом дослідження є процес проєктування та розробки гри жанру шутер засобами ігрового рушія Unity. Предметом дослідження є використання рушія Unity для створення гри жанру шутер.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	3 лютого 2023 – 6 березня 2023	Виконано
2	Проєктування структури системи	7 березня 2023 – 27 березня 2023	Виконано
3	Написання програмного коду	28 березня 2023 – 25 квітня 2023	Виконано
4	Тестування системи	26 квітня 2023 – 15 травня 2023	Виконано
5	Оформлення пояснювальної записки	15 травня 2023 – 2 червня 2023	
6	Попередній захист	2 червня 2023	Виконано

Здобувач вищої освіти **Олена МАЧУШНИК**

Керівник **Віталій ЛЕВКІВСЬКИЙ**

РЕФЕРАТ

Кваліфікаційна робота складається з програмного продукту та пояснювальної записки. Пояснювальна записка містить опис розробленої комп'ютерної гри, а саме 3D шутера.

Текстова частина викладена на 81 сторінці друкованого тексту. Пояснювальна записка містить 25 сторінок додатків. Список використаних джерел містить 43 найменувань та займає чотири сторінки. У роботі наведено 90 рисунків та 6 таблиць. Загальний обсяг роботи – 106 сторінок.

Метою роботи є створення комп'ютерної гри жанру шутер, що має високий рівень графіки та геймплею, розробка гри з використанням сучасних технологій, визначення і розробка унікального сюжету, впровадження різних рівнів складності. Також метою є проведення тестування та оптимізації гри для забезпечення її безперебійної роботи.

У роботі визначено основні завдання на розробку 3D шутера та обґрунтовано вибір технологій розробки, описано структуру програмного продукту та роботу алгоритмів, вказано результати тестування комп'ютерної гри.

Ключові слова: UNITY, ДОДАТОК, ГРА, 3D ШУТЕР, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

					КН.КР.Б – 122 – 23 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	3D гра жанру шутер засобами Unity під Windows- платформу Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		О.О.МАЧУШНИК						
Керівник		В.Л. ЛЕВКІВСЬКИЙ					3	106
Рецензент		О.В. КОРОТУН				Житомирська політехніка, група ЗКН-19-1		
Н. контр.								
Зав. каф.		М.С. ГРАФ						

ABSTRACT

The qualification work consists of a software product and an explanatory note. The explanatory note contains a description of the developed computer game, namely a 3D shooter.

The text part is laid out on 81 pages of printed text. The explanatory note contains 25 pages of annexes. The list of used sources contains 43 names and occupies four pages. The work contains 90 figures and 6 tables. The total volume of work is 106 pages.

The purpose of the work is to create a computer game of the shooter genre, which has a high level of graphics and gameplay, game development using modern technologies, definition and development of a unique plot, introduction of different levels of complexity. Also, the goal is to test and optimize the game to ensure its smooth operation on all computers.

The work defines the main tasks for the development of a 3D shooter and substantiates the choice of development technologies, describes the structure of the software product and the operation of algorithms, and indicates the results of testing the computer game.

Keywords: UNITY, APP, GAME, 3D SHOOTER, SOFTWARE.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	7
1.1. Постановка задачі	7
1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.....	8
1.3. Обґрунтування вибору інструментальних засобів	14
Висновки до першого розділу	17
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ	18
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	18
2.2. Дизайн-документ	22
2.3. Програмна реалізація гри.....	34
Висновки до другого розділу.....	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ	44
3.1. Інтерфейс та порядок роботи з програмним додатком.....	44
3.2. Тестування роботи програмного комплексу.....	68
Висновки до третього розділу	76
ВИСНОВКИ.....	77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	78
ДОДАТКИ.....	82

ВСТУП

Актуальність теми.

У поточному столітті спостерігається значний розквіт комп'ютерних технологій, що використовуються в широкому спектрі галузей суспільства, включаючи економіку, військові дослідження, енергоефективні технології та розробку ігрових програм, що стимулюють далі розвиток людства.

Ігри становлять важливу складову соціального та психічного розвитку суспільства. Ігрові процеси не лише замінюють реальні предмети, але також надають гравцеві можливість виконувати відповідну роль, пов'язану з виконанням покладених на нього обов'язків та встановлювати взаємини з іншими учасниками гри. Також ігри виконують не тільки розважальну роль, але і розвиваючу: у процесі гри формуються навички системного аналізу, і для гравця необхідно швидко та правильно обробляти інформацію, а потім прогнозувати подальший розвиток подій на її основі.

3D шутери є одним з найпопулярніших жанрів в індустрії відеоігор. Гравці по всьому світу зацікавлені у захоплюючих та динамічних віртуальних боях, які пропонують шутери. Тому розробка нових ігор такого жанру є важливою для задоволення попиту геймерів та залучення нових аудиторій. Також Windows є однією з найпопулярніших платформ серед геймерів і надає розробникам доступ до великого ринку споживачів.

Метою кваліфікаційної роботи є дослідження актуальних існуючих алгоритмів та способів реалізації комп'ютерної гри жанру шутер, дослідження побудови сучасних 3D ігор, розробка 3D шутера з високоякісною графікою та реалістичною фізикою.

Об'єктом дослідження є процес проектування та розробки гри жанру шутер засобами ігрового рушія Unity.

Предметом дослідження є використання рушія Unity для створення гри жанру шутер.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Постановка задачі

3D шутери є актуальними і популярними відеоіграми у сучасному світі, адже вони мають високу графічну якість та надають гравцям можливість зануритися в динамічні віртуальні бої.

На виконання кваліфікаційної роботи було поставлено завдання розробки 3D гри жанру шутер засобами Unity під Windows-платформу з використанням мови програмування C#. Гра має використовувати сучасну 3D графіку і бути зрозумілою в ігровому процесі.

3D шутер буде мати два режима гри: проходження рівнів за сюжетом та режим мультиплеєра. Режим проходження рівнів за сюжетом дозволить гравцеві зануритись в одиночну кампанію, де він буде виконувати різноманітні місії на різних планетах. У процесі проходження сюжетних рівнів гравець буде розблоковувати нові види зброї. У режимі мультиплеєра гравці матимуть можливість змагатися з іншими гравцями в онлайн-середовищі.

Режим гри проходження рівнів за сюжетом в 3D шутері буде пропонувати захопливу подорож планетами Сонячної системи, складність та різноманітність якої зростає з кожним наступним рівнем. Гравцю необхідно пройти п'ять рівнів. Початкові два рівні будуть відбуватися на Землі, де гравець зіштовхнеться із першими викликами та буде мати змогу оволодіти основними навичками гри. Третій рівень буде розгортатись на Марсі, де гравець зіткнеться з більш складними завданнями. Проходження четвертого рівня відбувається на Венері, а на останньому п'ятому рівні, розташованому на Урані, гравець зіткнеться з найбільшою кількістю ворогів.

Поєднання режиму проходження за сюжетом і мультиплеєра надасть гравцям широкий спектр ігрових можливостей і варіативність у грі. Гравці зможуть насолоджуватися захоплюючими моментами одиночної гри, а також випробувати свої навички та здібності у битвах з реальними суперниками.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Комп'ютерна гра має забезпечувати:

- зручне меню користувача із налаштуваннями;
- доступ до ігрового процесу із баловою системою оцінювання проходження кожного рівня;
- режим мультиплеєра.

1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи

3D шутер від першої особи, також відомий як FPS - це комп'ютерна гра, в якій гравець контролює персонажа у тривимірному просторі та може вільно керувати його діями. Рівні гри зазвичай представляють собою лабіринт із розташованими ворогами, союзниками і неігровими персонажами. Основна мета геймплею в шутерах полягає у знищенні перешкод та ворогів. Зазвичай рівні у FPS мають лінійну структуру, що означає, що вони розгортаються в одному напрямку, і гравцю надається бойове завдання. Але існують також нелінійні рівні, які можуть бути пройдені по-різному[1].

Battlefield - це серія відеоігор у жанрі шутер від першої особи. Ця серія відеоігор відрізняється особливою увагою до великих карт, командної роботи та загальновійськових конфліктів. Головною метою комп'ютерних ігор серії Battlefield є багатокористувацька гра в мережі. Battlefield ігри зазвичай фокусуються на масштабних онлайн-битвах, де ключовою складовою є гра в загонах. Крім солдатів, у цих битвах можуть брати участь також танки, літаки та інша військова техніка. Починаючи з Battlefield 2, ця серія систематично відстежувала онлайн-статистику для кожного гравця, що дозволило користувачам підвищувати свій ранг[2].

На сьогоднішній день дуже популярною є гра Battlefield 2042, яка була випущена у 2021 році. Ця гра відтворює конфлікт між двома країнами у близькому майбутньому, коли людство перебуває на межі вичерпання всіх ресурсів на планеті. Одним з основних привабливих аспектів гри є надзвичайно велика кількість гравців, які можуть одночасно приймати участь на одному

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

сервері. Версії гри для комп'ютерів з операційною системою Windows та ігрових консолей PlayStation 5 і Xbox Series X/S підтримують до 128 гравців, а версії для PlayStation 4 та Xbox One — до 64 гравців[3].



Рис. 1.1. Приклад ігрового процесу гри «Battlefield 2042»

У грі Battlefield 2042 доступний широкий асортимент різноманітних транспортних засобів та зброї. Натиснувши клавішу «В» на клавіатурі гравці можуть використовувати вбудований планшетний пристрій, для вибору зброї та автомобілів[4].



Рис. 1.2. Приклад ігрового процесу гри «Battlefield 2042»: вибір транспортного засобу



Рис. 1.3. Приклад ігрового процесу гри «Battlefield 2042»: вибір озброєння

Також у грі Battlefield 2042, гравці мають доступ до корисних функцій, таких як інтерактивна мапа та загальний рейтинг. Мапа з'являється на екрані після натиснення клавіші «М» на клавіатурі, а рейтинг – після натиснення клавіші «Tab»[4].



Рис. 1.4. Перегляд карти у грі «Battlefield 2042»



Рис. 1.5. Перегляд рейтингу в грі «Battlefield 2042»

BioShock - це відеогра від першої особи, сюжет якої розгортається у 1960 році і розповідає про пригоди головного героя на ім'я Джек, який відкриває підводне місто під назвою Rapture. Це місто було засноване бізнесменом Ендрю Райаном як ідеальна утопія, але після відкриття речовини «ADAM», що надає людям надздібності, місто зазнало краху. Джек намагається знайти шлях до виживання, борючись зі ворогами, що скористалися «ADAM», та величезними захисниками названими «Big Daddies». Гравець може використовувати різні підходи до боротьби зі своїми ворогами, використовуючи зброю, плазмід, які надають унікальні здібності, та захисні механізми «Rapture». Гра поєднує елементи рольових ігор, що дає гравцеві різноманітні підходи до протистояння ворогам, а також вибір між порятунком чи вбивством персонажів.

BioShock здобув широке визнання та отримав особливо високу оцінку критиків за його сюжет, тематику, візуальне оформлення, атмосферу та геймплей, його розглядають як одну з найкращих відеоігор, створених коли-небудь[5].



Рис. 1.6. Приклад ігрового процесу гри «Bioshock»

У грі «Bioshock», гравці мають можливість вибирати зброю через спеціальне меню, яке надає розманітність варіантів для бойових потреб героя.



Рис. 1.7. Приклад ігрового процесу гри «Bioshock»: вибір зброї

Меню ігрового процесу «Bioshock» складається із необхідних та чітко сформованих пунктів[5].

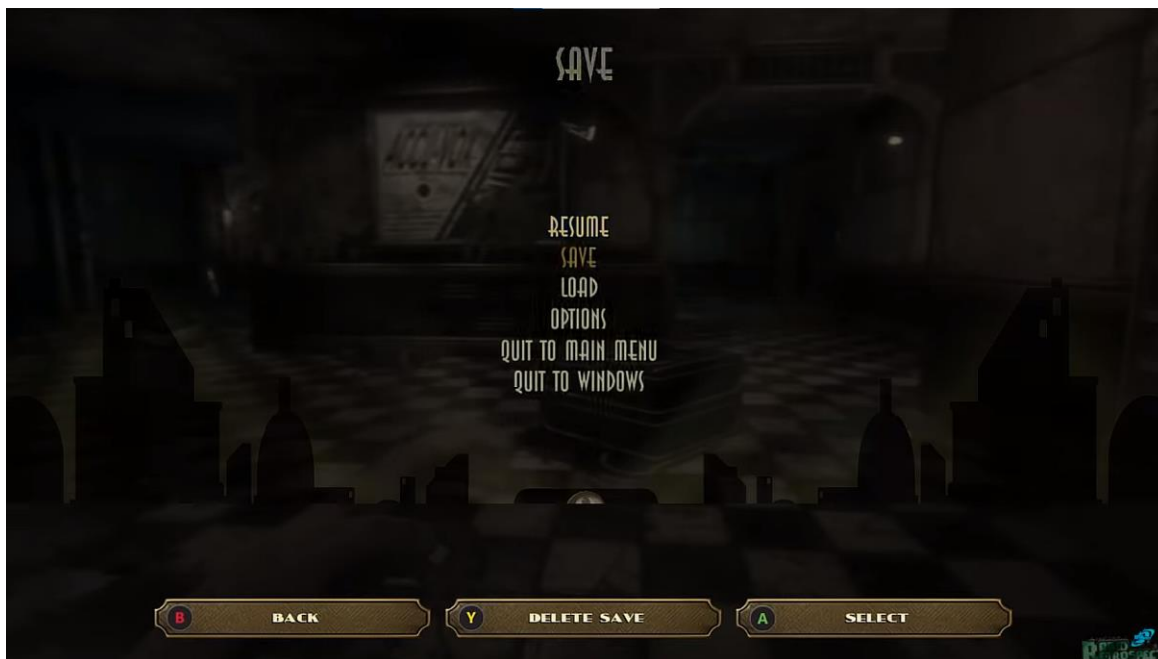


Рис. 1.8. Приклад ігрового процесу гри «Bioshock»: меню гри

Таблиця 1.1

Порівняння ігор «Battlefield 2042» та «Bioshock»

Функціонал	Battlefield 2042	Bioshock
Режим гри	Багатокористувацький	Одиночний
Кількість гравців	До 128 гравців для комп'ютерів з операційною системою Windows та ігрових консолей PlayStation 5 і Xbox Series X/S, до 64 гравців для PlayStation 4 та Xbox One	Один гравець
Сценарій	Військовий	Футуристичний
Різноманітність зброї	Різноманітна	Різноманітна
Елементи гри	Масштабні битви, техніка	Головоломки
Вплив гравця на сюжет	Обмежений	Прийняття рішень, вплив на кінцівку

Отже, було вирішено розробити власну 3D гру у жанрі шутер. У даному проєкті увагу слід приділити наступним важливим критеріям, що забезпечать якісний досвід геймплею:

- простий та зручний інтерфейс гри: необхідно створити інтуїтивно зрозумілий інтерфейс, що дозволить гравцям швидко освоїти управління та навігацію в грі;
- чіткість, послідовність та наповненість ігрового процесу: необхідно створити насичений геймплей, де гравець буде виконувати різноманітні завдання, проходити рівні та взаємодіяти з різними персонажами та об'єктами в грі; слід зберегти логіку та зв'язок між різними рівнями гри, щоб створити послідовний сюжет.

1.3. Обґрунтування вибору інструментальних засобів

Для того, щоб розробити комп'ютерну гру необхідно визначитись із ігровим рушієм — програмною основою, яка включає відповідні бібліотеки та програми підтримки, і зазвичай пропонує набір інструментів і функцій для розробки ігор.

Ігровий рушій надає широкий набір функцій, які допомагають у створенні якісної взаємодії під час ігрового процесу. Основні функціональні можливості, які зазвичай присутні у ігрових рушіях, включають механізм візуалізації для 2D або 3D-графіки, фізичні механізми, взаємодію із зіткненнями, звук, анімацію, штучний інтелект, мережу та інше. Розробники, які використовують ігрові рушії, часто здійснюють економію часу під час процесу написання ігор, що дозволяє зосередитися на вдосконаленні геймплею. Крім того, використання ігрового рушія дозволяє розробникам полегшити процес оптимізації технічної складової, забезпечуючи кращу продуктивність та сумісність з різними платформами[6].

Найпопулярнішими ігровими рушіями є CryEngine, Unity та Unreal Engine, тому слід здійснити аналіз кожного з даних ігрових рушіїв. Основні характеристики ігрових рушіїв CryEngine, Unity та Unreal Engine представлені у таблиці 1.2.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння базових характеристик ігрових рушіїв

Назва ігрового рушія	Мова програмування для написання коду гри	Кросплатформеність	Орієнтованість на 2D/3D	Платформи, які підтримуються
CryEngine	C++, C#	Присутня	3D	Windows, macOS, Linux, PlayStation 3, PlayStation 4, Wii U, Xbox 360, Xbox One, iOS, Android
Unity	C#	Присутня	2D, 2.5D, 3D	Windows, macOS, Linux, Xbox 360, Xbox One, Wii U, New 3DS, Nintendo Switch, PlayStation 4, PlayStation Vita, Windows Phone, iOS, Android, BlackBerry 10, Tizen, Unity Web Player, Windows Store, WebGL, Oculus Rift, Gear VR, Android TV, Samsung Smart TV
Unreal Engine	C++, Blueprints	Присутня	3D	Windows, macOS, Linux PCs; PlayStation 4, PlayStation 5, Xbox One, Xbox Series X, Nintendo Switch; iOS, Android mobile devices.

CryEngine відомий своєю потужною графікою та візуальними ефектами; має підтримку широкого спектру платформ, пропонує потужні засоби для створення великих відкритих світів та реалістичних фізичних ефектів. У той самий час, CryEngine також вимагає високих системних вимог та має більш складний план використання.

Unity широко використовується в ігровій індустрії та підтримує різні платформи, включаючи персональні комп'ютери, консолі, мобільні пристрої та віртуальну реальність. Unity має зручний інтерфейс середовища розробки та велику спільноту розробників; підтримує широкий спектр можливостей, включаючи візуалізацію, фізику, штучний інтелект та мережеву гру; ідеально підходить для початківців і розробки ігор середньої складності.

Unreal Engine забезпечує потужні засоби для реалістичної графіки та візуалізації; підтримує різні платформи та пропонує гнучку розробку ігор. Також Unreal Engine має велику бібліотеку готових рішень та високу продуктивність, але потребує значно більшого технічного досвіду для розробки складних проєктів[7].

Таблиця 1.3

Порівняння характеристик ігрових рушіїв

Категорія	CryEngine	Unity	Unreal Engine
Графіка	Фотореалістична, потужні візуальні ефекти	Фотореалістична, потужні візуальні ефекти	Фотореалістична, потужні візуальні ефекти
Складність	Складна	Для початківців і середньої складності	Складна
Спільнота розробників	Середня	Велика	Велика
Можливості	Великі відкриті світи, реалістична фізика	Візуалізація, фізика, штучний інтелект, мережа	Реалістична графіка, багато готових рішень, висока продуктивність

Враховуючи усі фактори кожного ігрового рушія, для розробки 3D шутера був обраний ігровий рушій Unity. Unity має широку підтримку платформ, зручний інтерфейс розробки та велику спільноту розробників, що сприяє обміну знаннями та розв'язанню потенційних проблем. Він також надає розширений набір функцій, які можуть бути використані для створення захоплюючого 3D шутера.

Висновки до першого розділу

У першому розділі було проведено постановку задачі, були визначені основні функціональні вимоги до програмного продукту, які повинні бути враховані під час розробки, були розглянуті існуючі аналоги програмного забезпечення, що дозволило проаналізувати їх переваги та недоліки, а також визначити напрямки покращення власного продукту. Враховуючи визначені вимоги до програмного продукту та результати дослідження існуючих аналогів, було вирішено використовувати ігровий рушій Unity для розробки 3D шутера.

Отже, можна зробити висновок, що перший розділ кваліфікаційної роботи містить важливу інформацію щодо аналізу наявних на ринку 3D шутерів, що дозволяє зробити правильний вибір напрямків розробки та забезпечити якість та цікавість гри для користувачів.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Діаграма варіантів використання (Use case Diagram) – це графічне зображення можливої взаємодії користувача з системою. Часто діаграма варіантів використання показує різні варіанти використання та різні типи користувачів системи. Варіанти використання представлені колами або еліпсами. Актори часто зображуються як фігурки[8].

На рисунку 2.1 представлена діаграма варіантів використання головного меню гри, а на рисунку 2.2 – діаграма використання меню в режимі «Multiplayer».



Рис. 2.1. Діаграма варіантів використання головного меню гри



Рис. 2.2. Діаграма варіантів використання меню у «Multiplayer»

Для розробки діаграм варіантів використання гри під час ігрового процесу (рисунки 2.3 - 2.4) були визначені два актори: гравець та противник. Ці два актора зустрічаються у режимах «Play by plot» та «Multiplayer».

Гравець перебуває у ігровому процесі, де можна обрати зброю, зробити постріл або отримати шкоду. У режимі «Play by plot» також доступні меню паузи, програшу та успішного проходження рівня.



Рис. 2.3. Діаграма варіантів використання гри у «Multiplayer»



Рис. 2.4. Діаграма варіантів використання гри у «Play by plot»

Діаграма класів є основним будівельним блоком об'єктно-орієнтованого моделювання та описує структуру системи, показуючи класи системи, їхні атрибути, методи і зв'язки між об'єктами[9].

Для 3D шутера було побудовано діаграму класів, яка складається із класів «Player» та «Enemy».

Клас «Player» має властивості здоров'я, набраних балів та обраної зброї, які представлені як цілі числа. Даний клас також має методи стрільби, нанесення удару та отримання пошкодження.

Клас «Enemy» має властивості здоров'я, пошкодження та можливості пострілу, які представлені як цілі числа (int) та логічне число (bool). У класі «Enemy» є методи атаки, пострілу у граця, отримання пошкодження та помирання.

Різні типи ворогів генеруються на основі класу «Enemy» із використанням патерну фабрика. Цей підхід дозволяє легко додавати різноманітних ворогів, додаючи нові префаби.

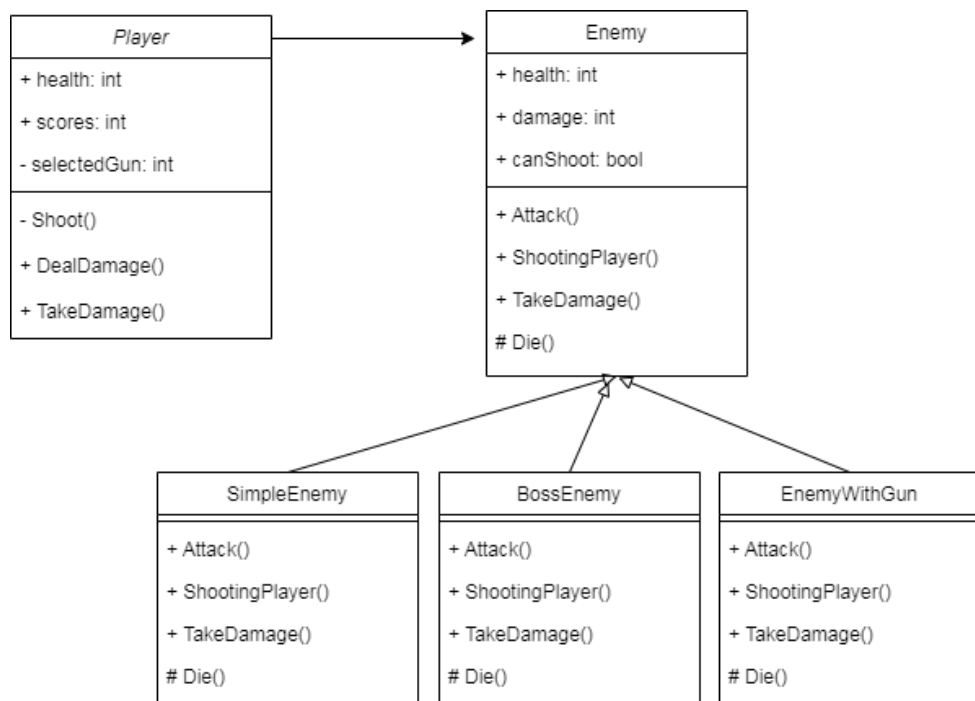


Рис. 2.5. Діаграма класів

У грі реалізовані різноманітні класи, що забезпечують управління рухами користувача, міні-картою, зброєю, відображенням запасу здоров'я та іншим функціоналом. Наприклад, клас міні-карти забезпечує оновлення відображення об'єктів та локацій; клас відображення запасу здоров'я відповідає за оновлення графічного інтерфейсу запасу здоров'я. Налаштування гри реалізовані через відповідний клас, що дозволяє користувачу налаштовувати різні параметри гри, такі як якість графіки та звукові ефекти.

BackgroundMusic Класс → MonoBehaviour	CameraController Класс → MonoBehaviour	Dialogue Класс → MonoBehaviour	EnemyFactory Класс → MonoBehaviour	FloatingEnemy... Класс → MonoBehaviour	GameManagerS... Класс → MonoBehaviour
Gun Класс → MonoBehaviour	MainMenuMan... Класс → MonoBehaviour	MatchManager Класс → MonoBehaviour	MGameManager Класс → MonoBehaviour	Minimap Класс → MonoBehaviour	MLauncher Класс → MonoBehaviourPunC...
MMinimap Класс → MonoBehaviour	MouseLook Класс → MonoBehaviour	MPlayerController Класс → MonoBehaviourPunC...	MPlayerHealthBar Класс → MonoBehaviour	MPlayerSpawner Класс → MonoBehaviour	MRoomButton Класс → MonoBehaviour
MSpawnManager Класс → MonoBehaviour	PlayerHealthBar Класс → MonoBehaviour	PlayerManager... Класс → MonoBehaviour	PlayerMovement Класс → MonoBehaviour	SettingsMenu Класс → MonoBehaviour	WeaponManage... Класс → MonoBehaviour
WeaponManage... Класс → MonoBehaviour	WeaponSwitching Класс → MonoBehaviour				

Рис. 2.6. Перелік всіх класів

2.2. Дизайн-документ

2.2.1. Геймплей

У 3D шутері керування головним героєм здійснюється від першої особи, що надає гравцеві відчуття участі у подіях гри. Гравець має повний контроль над рухом персонажа, включаючи переміщення вперед, назад, вліво та вправо, а також можливість обертати камеру, що дозволяє змінювати напрямок перегляду.

Одним з ключових елементів геймплею є можливість вибору різних видів зброї. Гравець має можливість обирати шість різноманітних гвинтівок, які поступово стають доступними по мірі проходження рівнів. Кожен вид гвинтівки має свої унікальні характеристики та можливості.

У 3D шутері наявні п'ять рівнів гри, події яких відбуваються на різних планетах: Земля, Марс, Венера та Уран. Після запуску рівня повідомляється ціль місії. Для успішного проходження рівня необхідно знищити всіх ворогів протягом всіх раундів. По закінченню гри гравець має можливість переглянути набрані бали та перейти на наступний рівень. Початково гравець має доступ тільки до першого рівня. Можливість переходу на ішний рівень відбувається тільки після успішного проходження попереднього рівня.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

У комп'ютерній 3D грі жанру шутер також існує багатокористувацький режим, який дозволяє гравцям змагатися між собою в реальному часі через підключення до мережі Інтернет через використання Photon Unity Networking.

Photon Unity Networking (PUN) – це пакет Unity для багатокористувацьких ігор, де об'єкти можна синхронізувати через мережу Інтернет. Завдяки PUN з'являється можливість створювати нові кімнати, або приєднуватись до вже існуючих, де гравці можуть зіграти разом[10].

У багатокористувацькому режимі гри максимальна кількість гравців у одній кімнаті обмежена і становить шість учасників, що сприяє збалансованій та динамічній грі.

Після поразки у багатокористувацькій грі, гравець автоматично перепідключається в ігровий процес. Це важливо, оскільки гравцю не потрібно повністю виходити з гри та знову створювати або шукати вільну кімнату.

2.2.2. Локації та сцени

Кожен рівень 3D шутера розгортається на новій локації, які були завантажені із Unity Asset Store. Кількість нових локацій значно додає різноманітності до комп'ютерної гри. Наприклад, мапа багатокористувацької гри розгортається на складській базі із різноманітними ангарами та приміщеннями. Гравці можуть використовувати навколишні споруди як укриття від постірілів ворогів та як безпечне місце для здійснення власних атак[12].

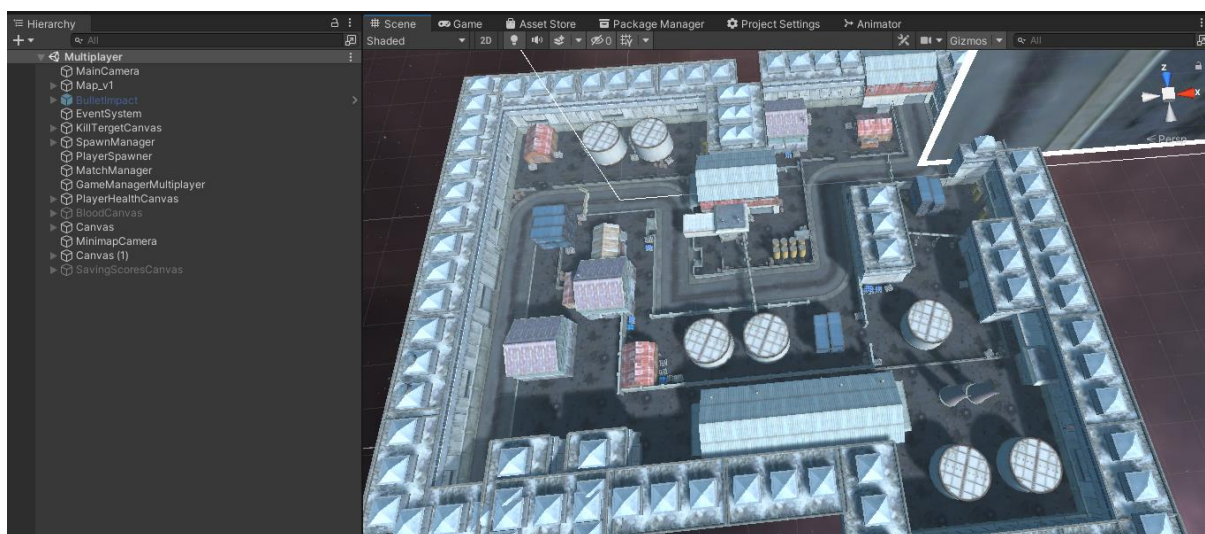


Рис. 2.7. Мапа багатокористувацької гри: вигляд зверху

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

У перших двох рівнях використовується локація міста Нью-Йорк із величезними хмарочосами. Головний герой має місію врятувати рідну планету Земля від ворогів, тому сюжет починається саме із цієї планети[13].

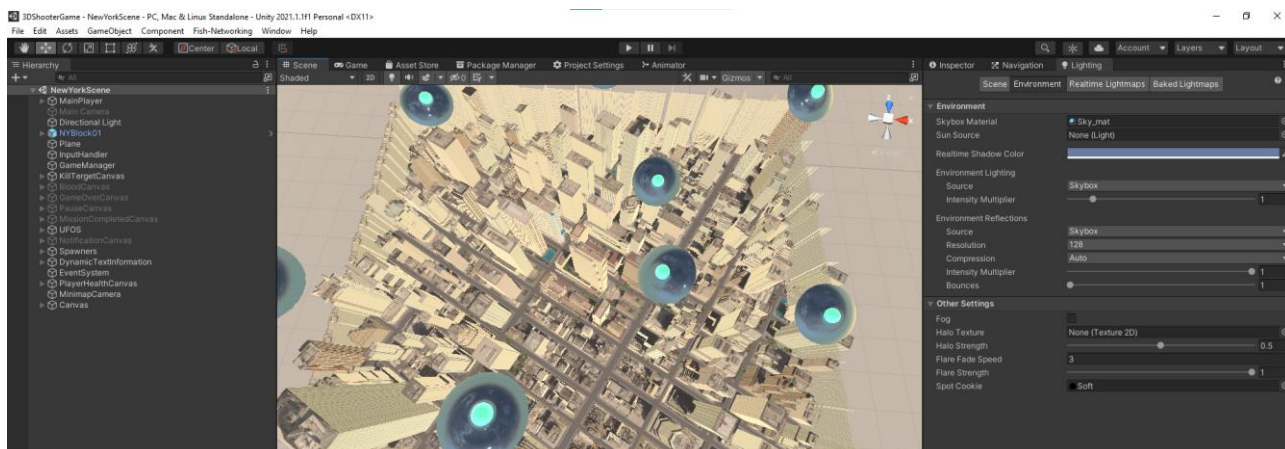


Рис. 2.8. Мапа першого рівня гри: вигляд зверху

Другий рівень, зі зміненою локацією, переносить гравця до парку в Нью-Йорку. Це мальовниче середовище зелених насаджень та доріжок, які пролягають серед природи. Парк створює іншу атмосферу та додає нові можливості для битви. Гравець може скористатися відкритими полями для стратегічного розташування та використовувати ландшафт для своєї переваги[14].

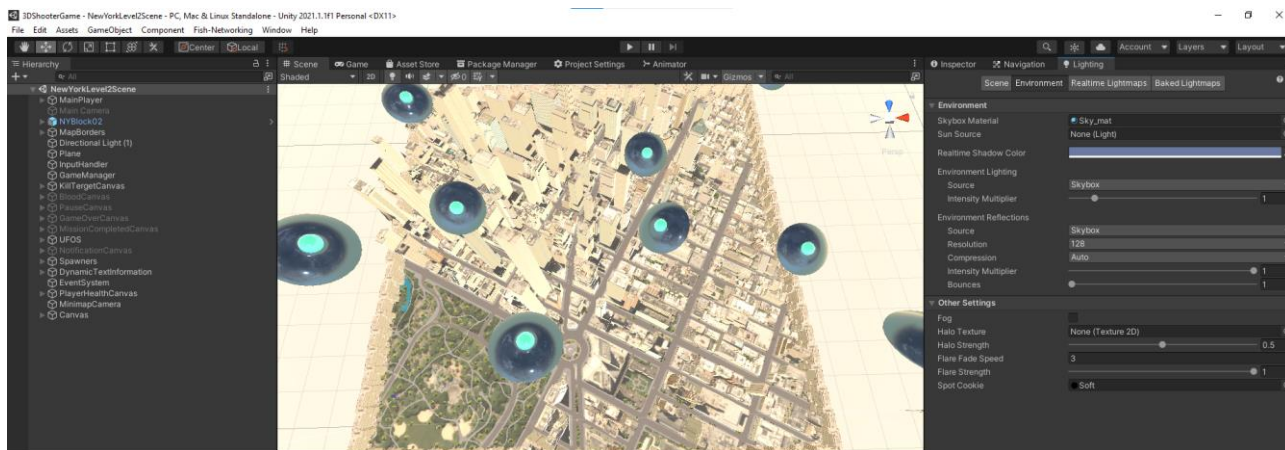


Рис. 2.9. Мапа другого рівня гри: вигляд зверху

Починаючи з третього рівня, гравець занурюється в захоплюючий світ інопланетних пейзажів, де кожна нова локація пропонує унікальну та захопливу неземну атмосферу. Замість продовження пригод у місті Нью-Йорк, головний герой переноситься на інші планети, а саме Марс, Венеру та Уран, де відбуваються події, пов'язані з місією порятунку всієї Сонячної системи[15].

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

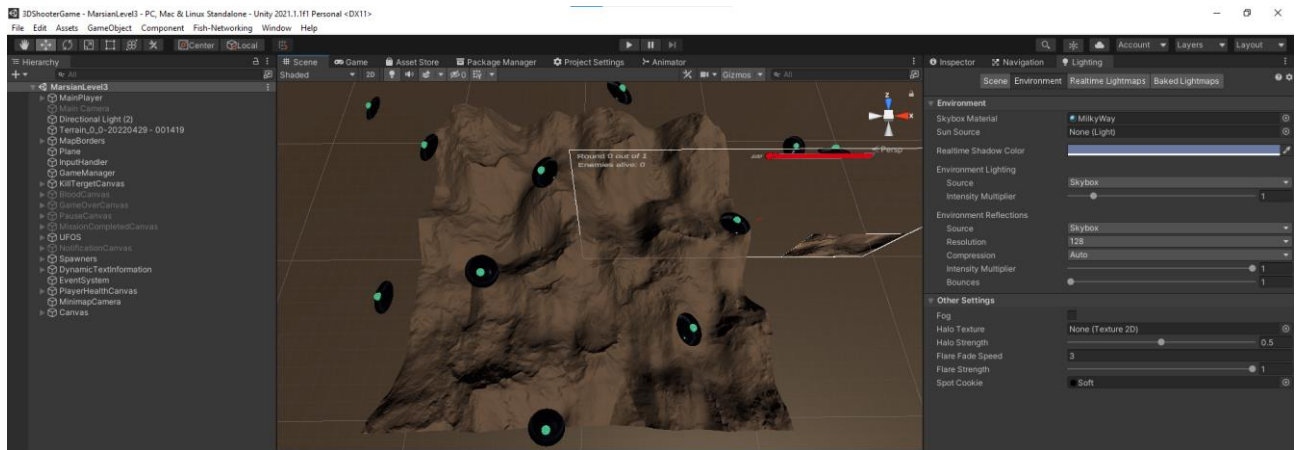


Рис. 2.10. Мапа третього рівня гри: вигляд зверху

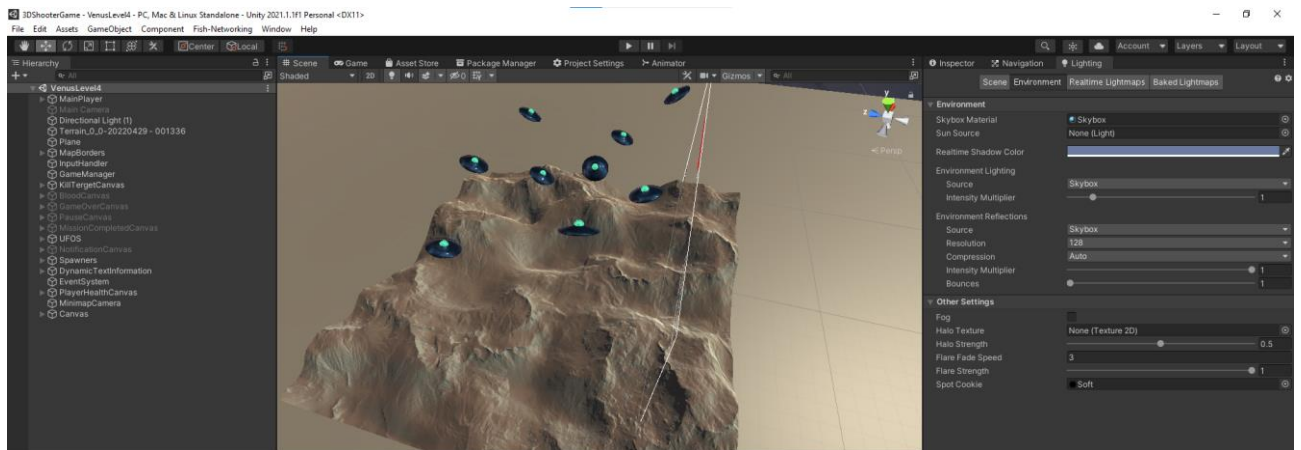


Рис. 2.11. Мапа четвертого рівня гри: вигляд зверху

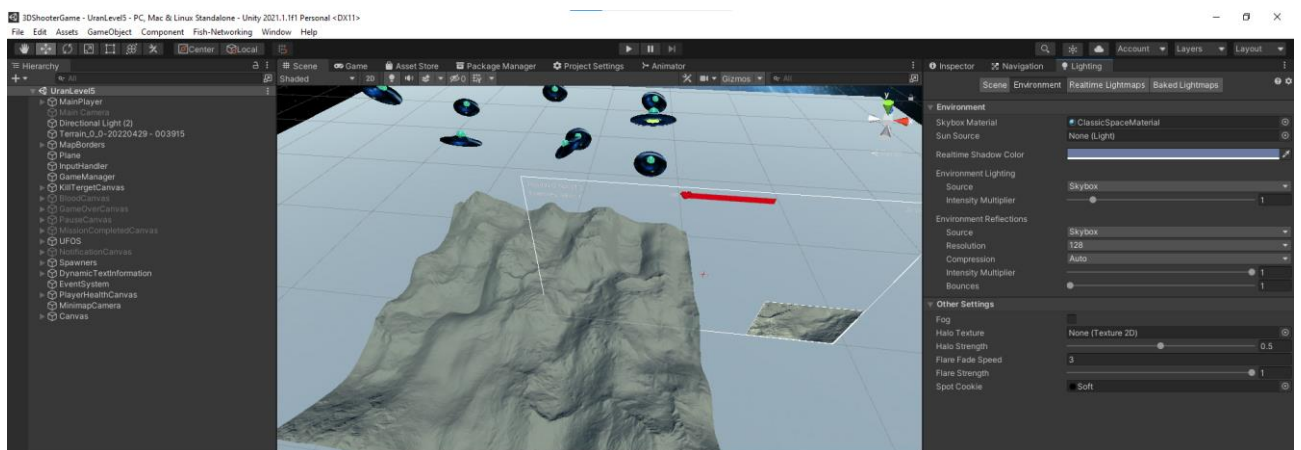


Рис. 2.12. Мапа п'ятого рівня гри: вигляд зверху

2.2.3. Персонажі, гейм-об'єкти та текстури

Для створення гри було використано широкий спектр ресурсів, включаючи гейм-об'єкти та текстури, які були отримані з Unity Asset Store. Персонажі та їх анімації були отримані із Міхато[16].

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

У багатокористувацькій грі всім гравцям надається доступ лише до одного персонажа, який володіє різними видами озброєння, що зберігаються в об'єкті «GunHolder». Персонаж містить «Character Controller», який дозволяє легко виконувати рухи, «Audio Source» для відтворення звуку пострілу, «Animator» для відтворення анімацій, «Photon Transform View» для синхронізації руху персонажів між усіма гравцями у реальному часі та скрипт «MPlayerController» для управління логікою персонажа. Також над персонажем розташований текстовий об'єкт для відображення імені користувача під час гри.

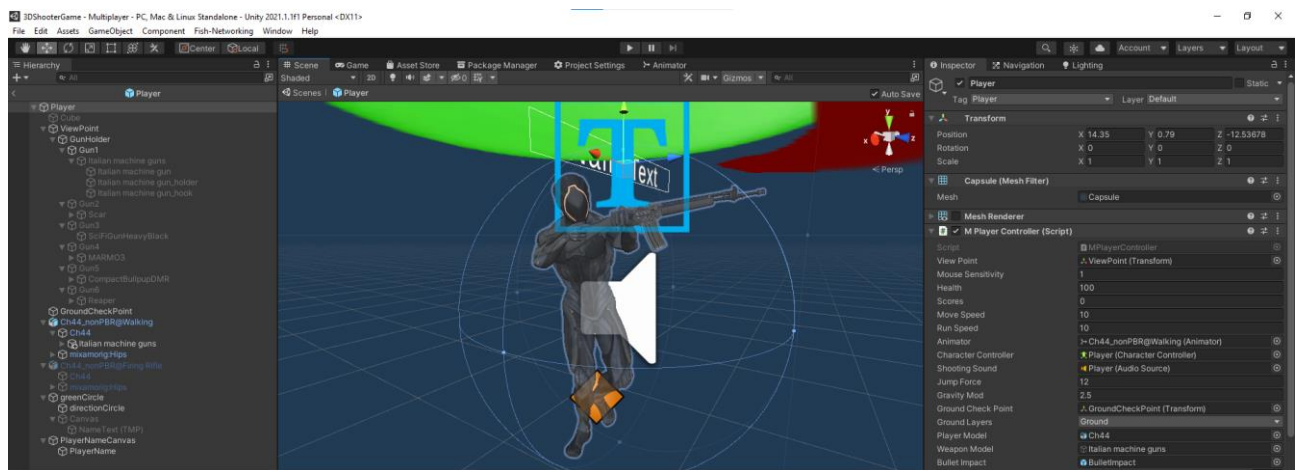


Рис. 2.13. Персонаж у режимі гри «Multiplayer»

У кожному рівні були додані літаючі тарілки для підкреслення сюжетної лінії, пов'язаної із інопланетним вторгненням в Сонячну систему[17].

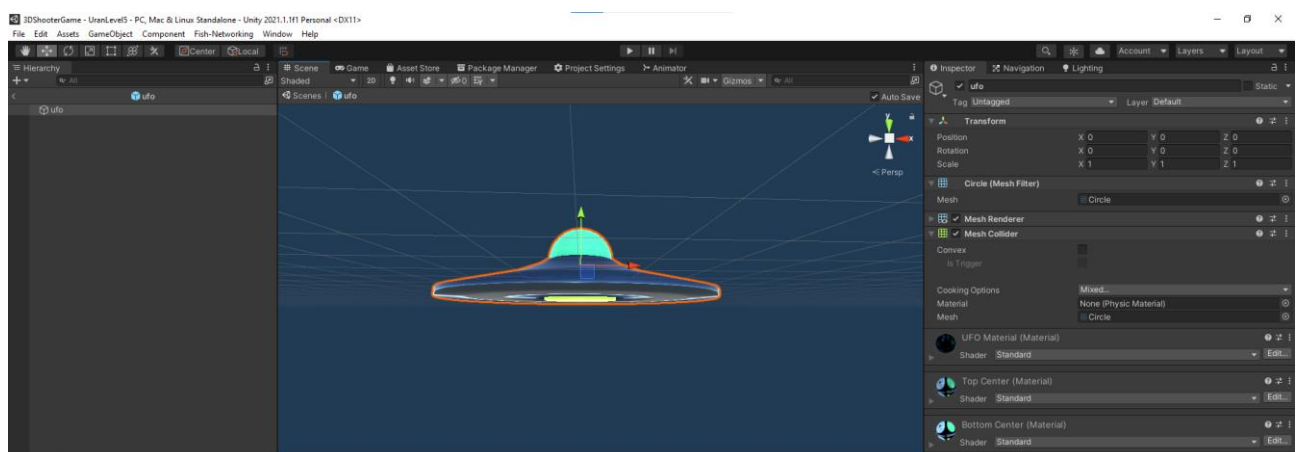


Рис. 2.14. Модель літаючої тарілки

Головний герой режиму гри «Play by plot» містить «Character Controller», «Audio Source», «Animator» та два скрипта «PlayerMovement» і «PlayerManagerByPlot».

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

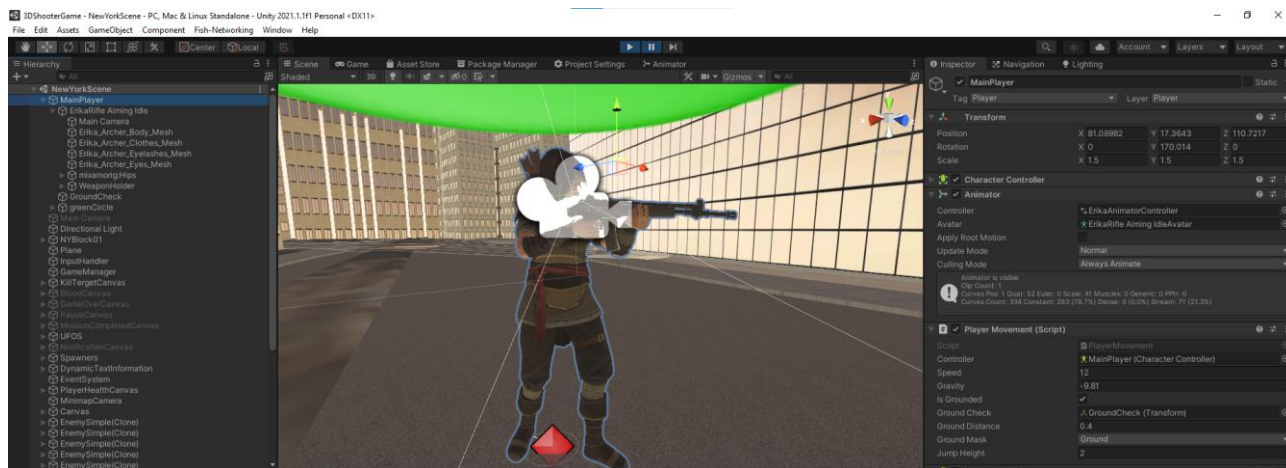


Рис. 2.15. Головний персонаж у режимі гри «Play by plot»

У всіх рівнях гри гравець зустрічає різноманітних ворогів. Три основних типи ворогів, доступних у всіх рівнях, включають простих, сильних і ворогів з рушницями. Кожен з них має власні характеристики та рівень здоров'я, що впливає на їхню стійкість під час битви.

Перший тип ворога – простий ворог, який має запас здоров'я у 50 одиниць. Це найбільш поширений ворог і представляє базову загрозу для гравця. Боротьба з простим ворогом допомагає гравцеві освоїти основи геймплею і набратися досвіду перед зустріччю з більш складними супротивниками.

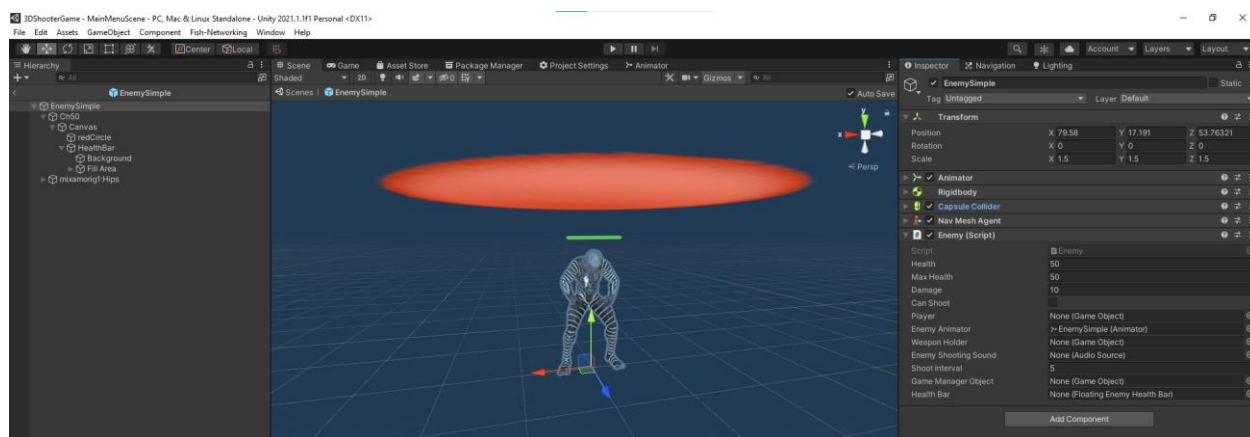


Рис. 2.16. Простий ворог у режимі гри «Play by plot»

Другий тип - сильний ворог - має запас здоров'я у 150 одиниць. Даний ворог володіє більшою фізичною силою та стійкістю, що робить його складнішим для перемоги.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

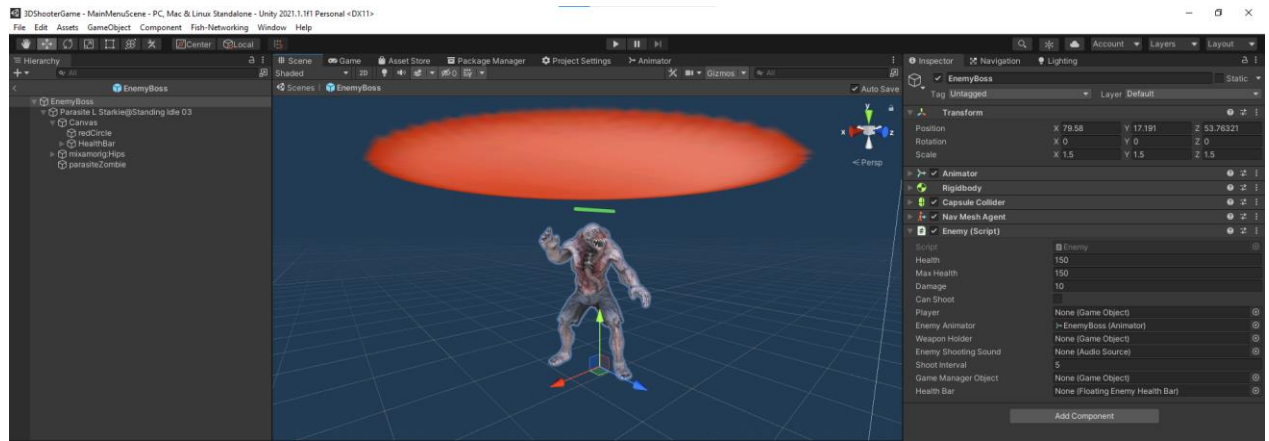


Рис. 2.17. Сильний ворог у режимі гри «Play by plot»

Третій тип - ворог з рушницею - має запас здоров'я у 90 одиниць. Цей ворог відрізняється від інших типів можливістю стріляти на відстані у головного персонажа.

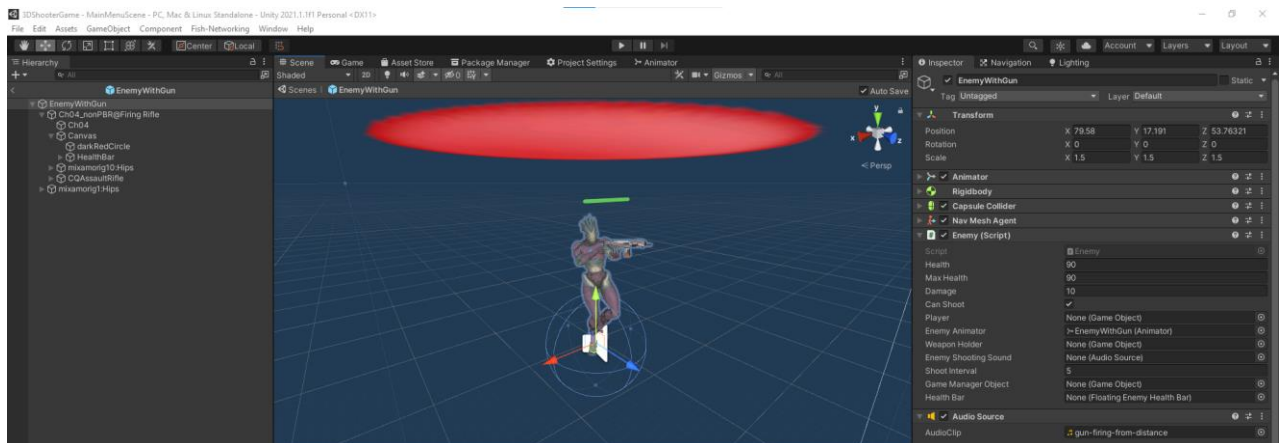


Рис. 2.18. Ворог із рушницею у режимі гри «Play by plot»

У грі доступні шість різних видів рушниць, кожна з них має свою унікальну силу пострілу. Рушниці виступають як важлива складова зброї гравця, дозволяючи йому адаптуватись до різних ситуацій та стратегій[18-23].

У першому рівні гри гравцю доступні лише дві рушниці. Це дозволяє початківцю ознайомитись з основними можливостями рушниць та їхньою силою.

Кожен наступний рівень відкриває нову рушницю, розширюючи арсенал гравця і надаючи більше можливостей для ведення битви. У багатокористувацькій грі всі рушниці доступні відразу для всіх гравців. Це дозволяє зрівняти шанси усіх гравців і надає можливість вибору в залежності від

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

особистих уподобань та геймплейної стратегії. Кожен гравець може обрати рушницю, яка найбільше відповідає його стилю гри або поточним умовам.

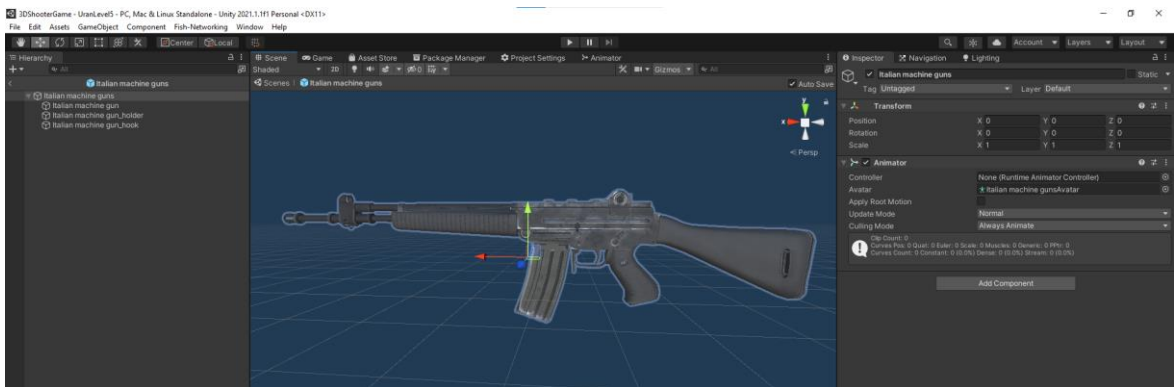


Рис. 2.19. Перша рушниця

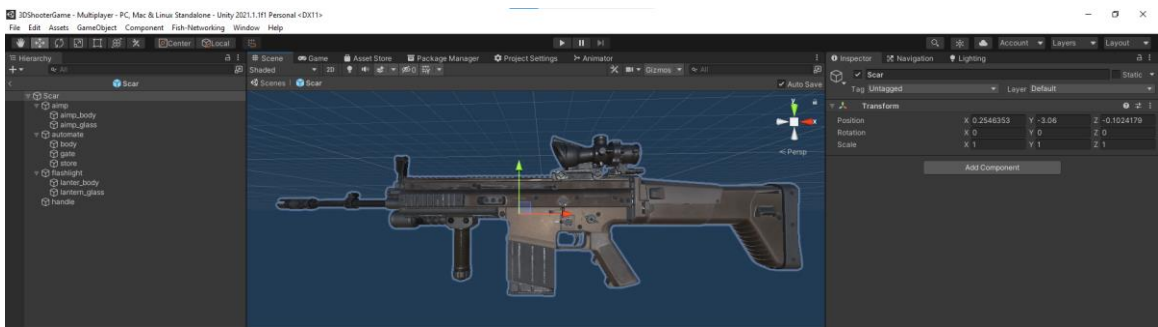


Рис. 2.20. Друга рушниця

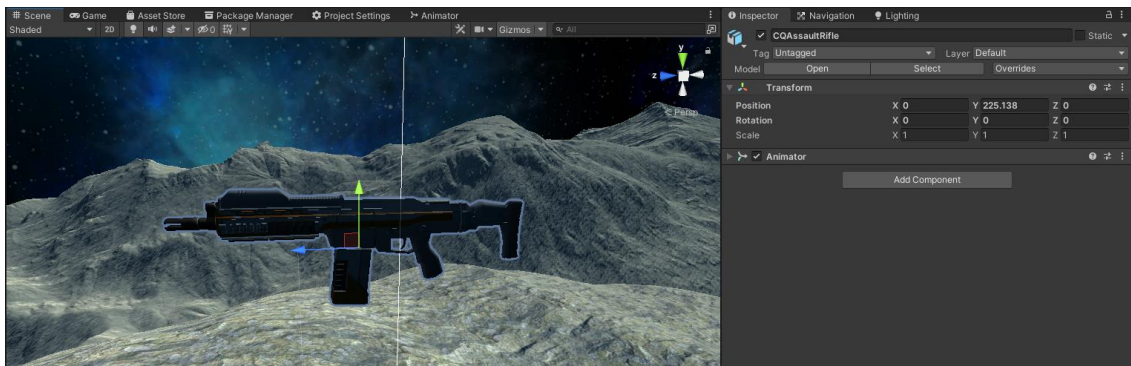


Рис. 2.21. Третя рушниця

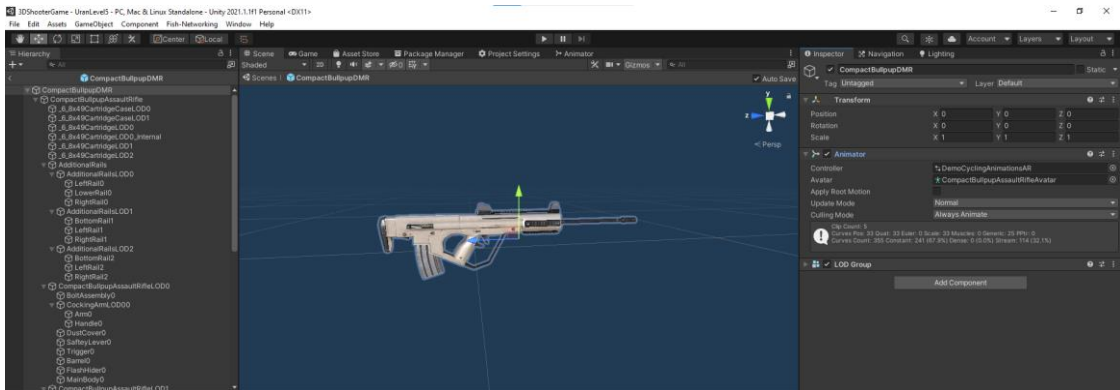


Рис. 2.22. Четверта рушниця

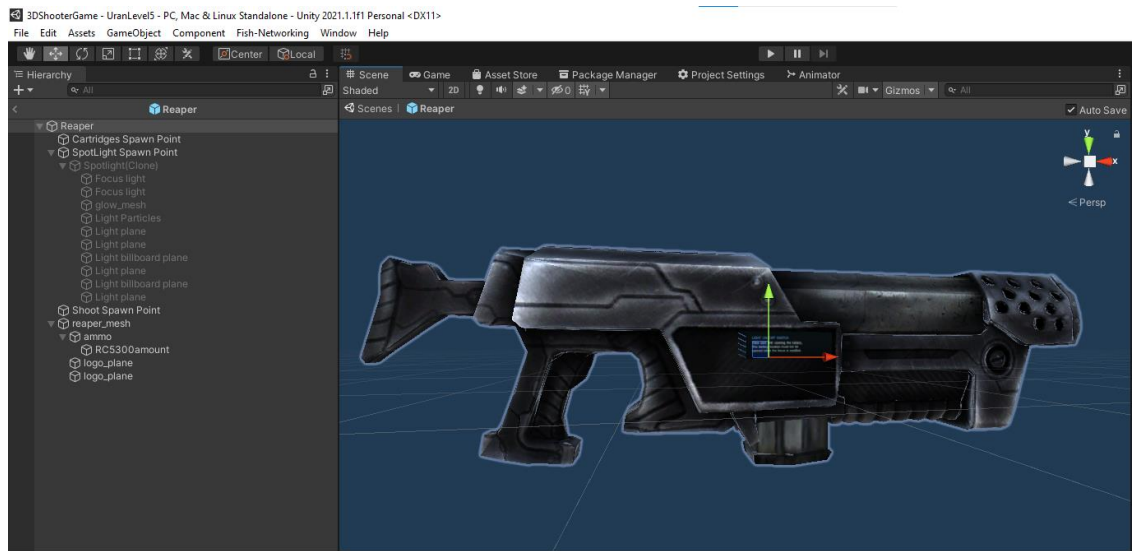


Рис. 2.23. П'ята рушниця

Для досягнення реалістичного візуального представлення Марсу, Венери та Урану в грі були використані спеціальні текстури, що відтворюють унікальні особливості цих планет[15].

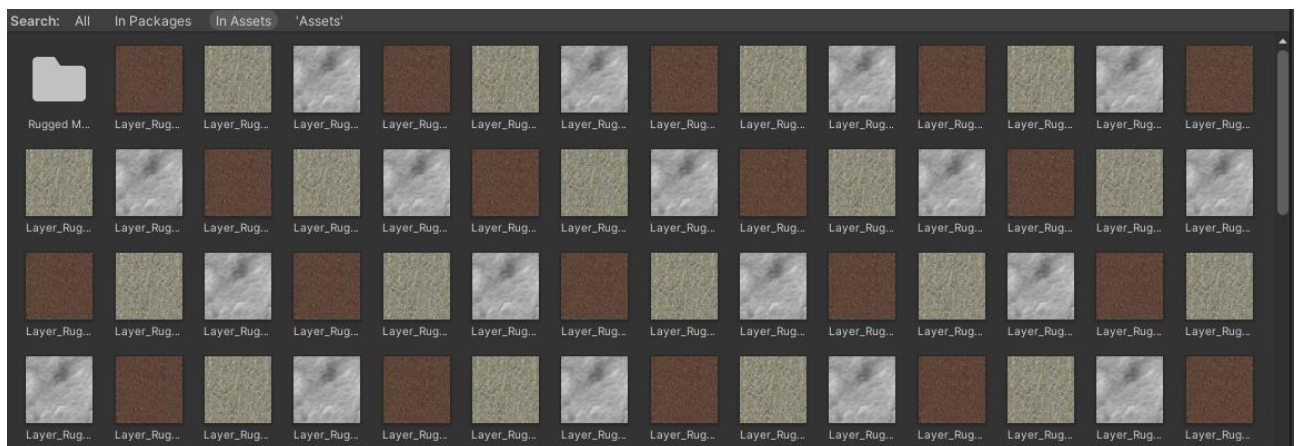


Рис. 2.24. Текстури поверхонь планет

Для кожного рівня гри було додано «Skybox material» для створення особливої атмосфери та візуального настрою. «Skybox material» є текстурою, яка охоплює всі сторони гри і створює враження зовнішнього середовища, такого як небо або космос. Завдяки «Skybox material» кожен рівень має свою унікальну візуальну атмосферу[24-27].

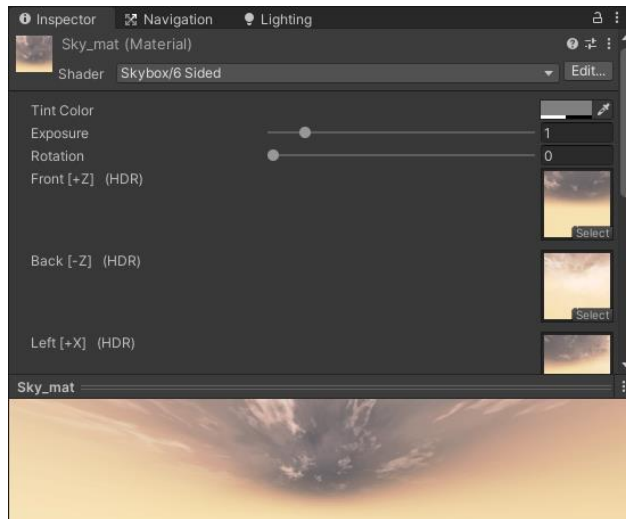


Рис. 2.25. «Skybox material» для рівнів на планеті Земля

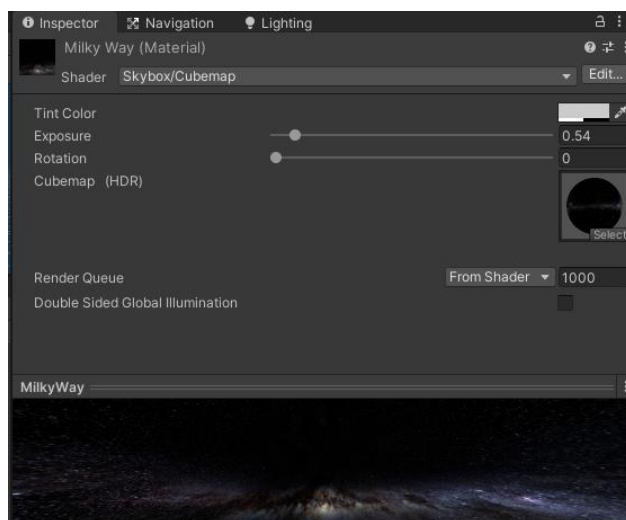


Рис. 2.26. «Skybox material» для рівня на планеті Марс

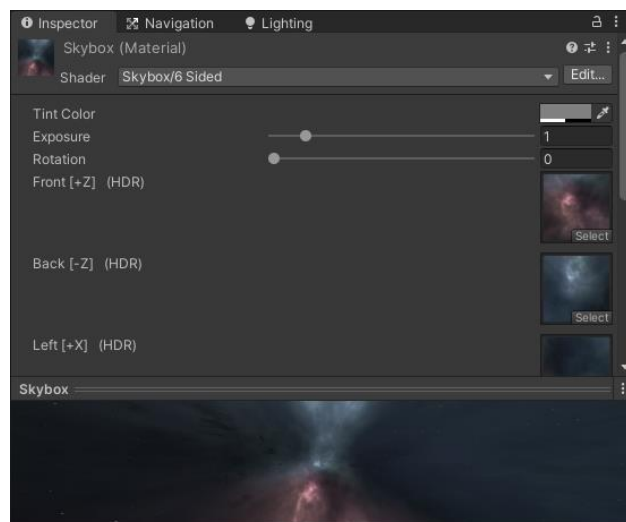


Рис. 2.27. «Skybox material» для рівня на планеті Венера та багатокористувацької гри

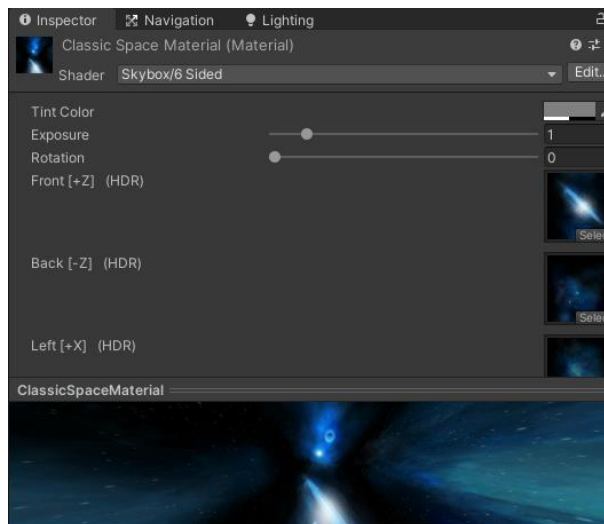


Рис. 2.28. «Skybox material» для рівня на планеті Уран

У рівні на планеті Земля та у багатокористувацькій грі, окрім рельєфів та поверхневих особливостей, також були використані різноманітні будівлі для створення живих та реалістичних гральних сцен.

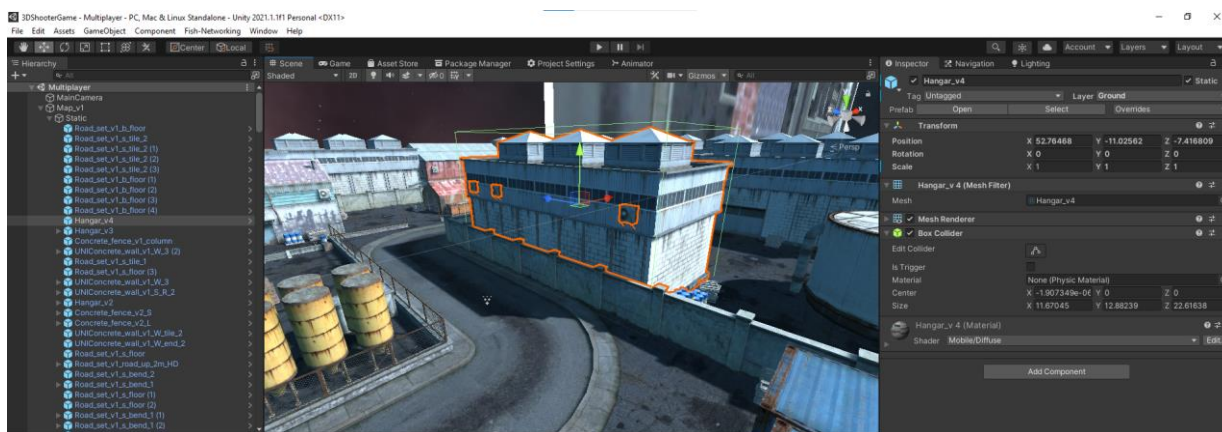


Рис. 2.29. Об'єкти на мапі багатокористувацької гри

У рівнях на планеті Земля гравці можуть зустріти такі елементи як міські будівлі, високі хмарочоси та житлові будинки.

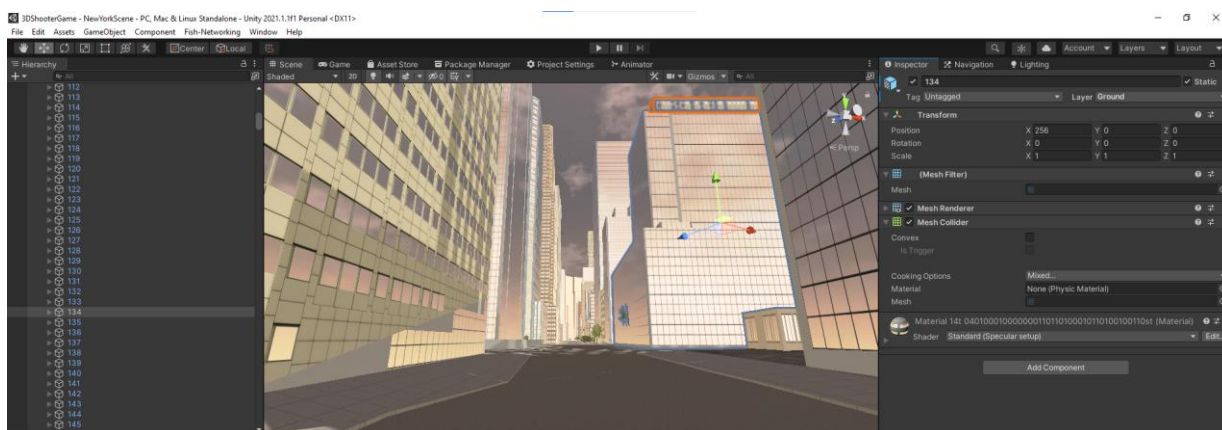


Рис. 2.30. Об'єкти на мапі першого рівня гри

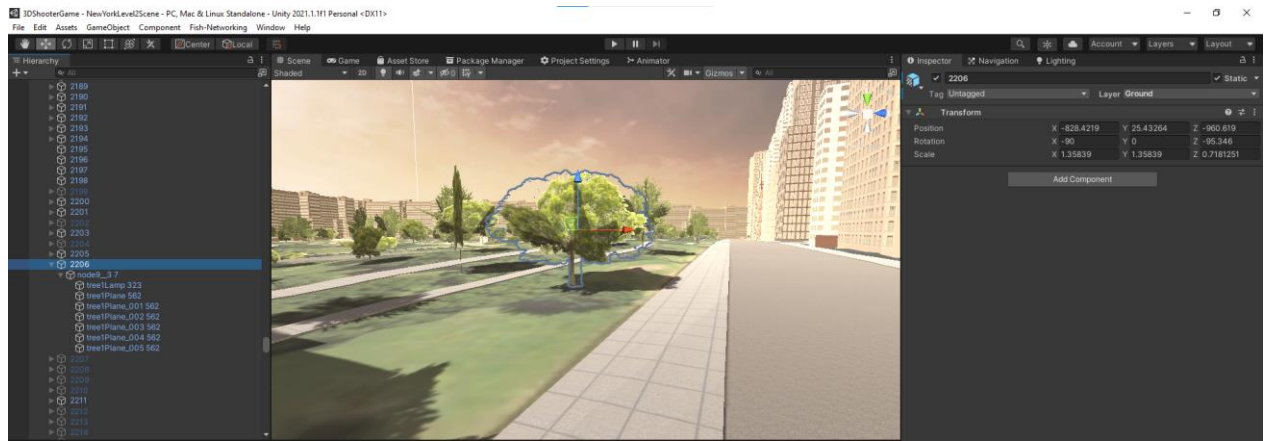


Рис. 2.31. Об'єкти на мапі другого рівня гри

На планетах Марс, Венера та Уран, відповідно до їх природних характеристик, рельєф місцевості виступає як основний елемент. Гравець має можливість досліджувати ландшафти з гірськими вершинами та іншими геологічними формаціями.

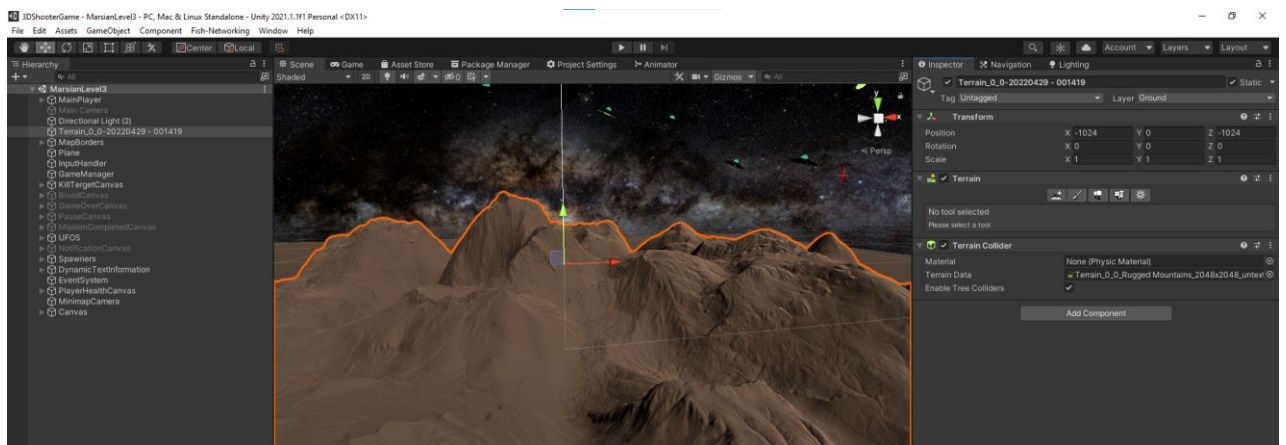


Рис. 2.32. Об'єкти на мапі третього рівня гри

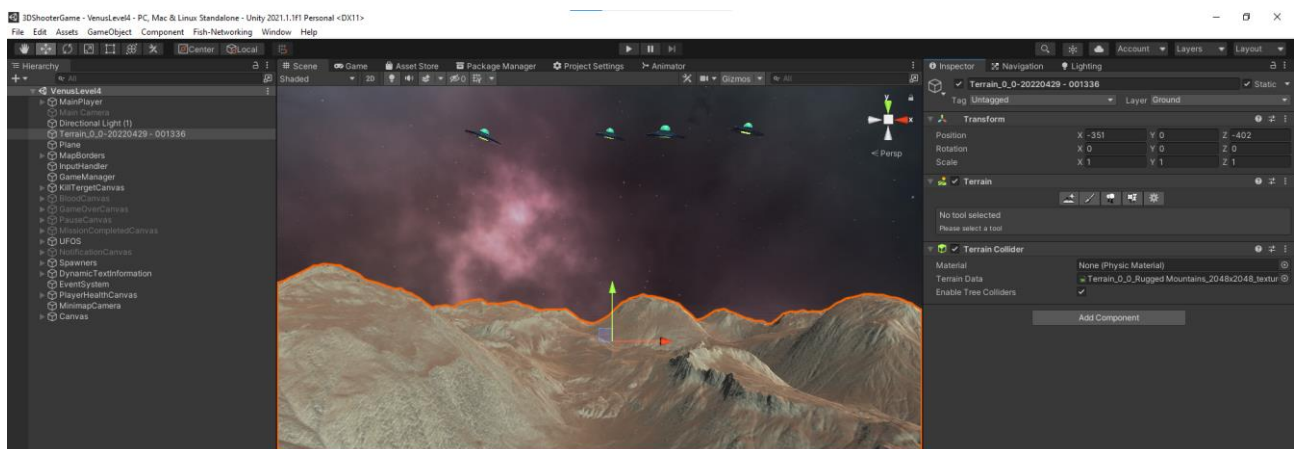


Рис. 2.33. Об'єкти на мапі четвертого рівня гри

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33

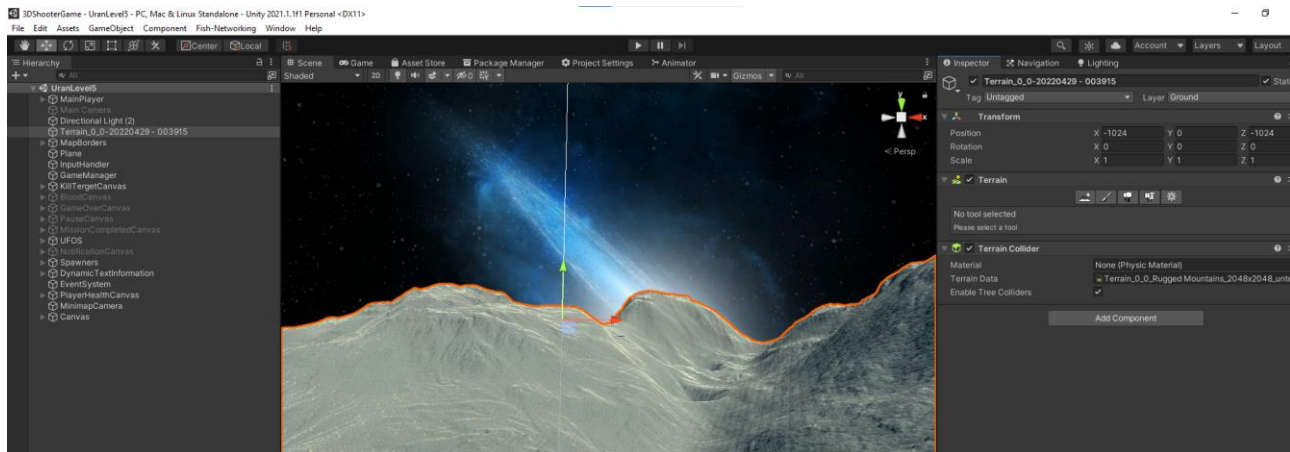


Рис. 2.34. Об'єкти на мапі п'ятого рівня гри

Для створення фонової музики та звуку пострілу було використано «Audio Mixer», за допомогою якого можна контролювати та налаштовувати звук у грі. Аудіофайли звуків для гри були завантажені із веб-сайту Pixabay із безкоштовними аудіо-ефектами[28].

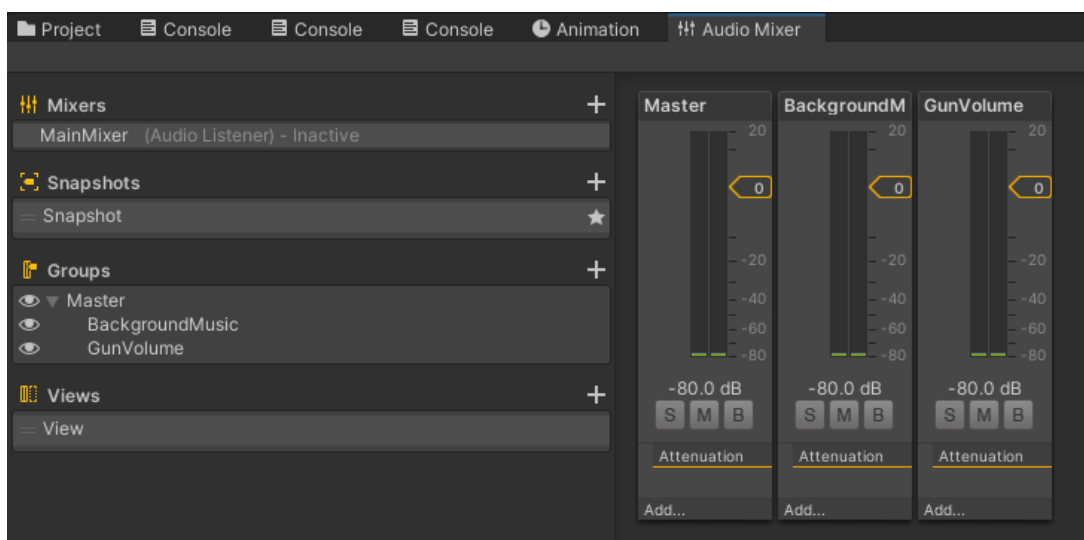


Рис. 2.35. Вміст «Audio Mixer»

2.3. Програмна реалізація гри

Програмна реалізація гри будується на основі алгоритмів роботи гри, які визначають основні механізми, правила і логіку гри. Це включає рух гравця, взаємодію гравця із ворогами та навколишнім середовищем.

Почнемо з головного меню, яке з'являється відразу після запуску гри. У меню постійно оновлюється інформація про доступність кожного рівня користувачу та набрані бали у режимах «Multiplayer» та «Play by plot».

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

```

private void Update()
{
    int scores = PlayerPrefs.GetInt("LevelScoresScene3");
    highestScoreLevel1Text.text = scores.ToString();
    scores = PlayerPrefs.GetInt("LevelScoresScene5");
    highestScoreLevel2Text.text = scores.ToString();
    scores = PlayerPrefs.GetInt("LevelScoresScene7");
    highestScoreLevel3Text.text = scores.ToString();
    scores = PlayerPrefs.GetInt("LevelScoresScene9");
    highestScoreLevel4Text.text = scores.ToString();
    scores = PlayerPrefs.GetInt("LevelScoresScene11");
    highestScoreLevel5Text.text = scores.ToString();
    scores = PlayerPrefs.GetInt("MultiplayerScores");
    highestScoreMultiplayerText.text = scores.ToString();

    if (PlayerPrefs.GetInt("LevelCompletedScene3") != 1)
    {
        level2Button.interactable = false;
    }
    if (PlayerPrefs.GetInt("LevelCompletedScene5") != 1)
    {
        level3Button.interactable = false;
    }
    if (PlayerPrefs.GetInt("LevelCompletedScene7") != 1)
    {
        level4Button.interactable = false;
    }
    if (PlayerPrefs.GetInt("LevelCompletedScene9") != 1)
    {
        level5Button.interactable = false;
    }
}

```

У головному меню наявні такі методи як OpenScores(), CloseScores() та подібні, які відповідають за ефективне управління вікнами.

```

public void OpenScores()
{
    scoresScreen.SetActive(true);
}
public void CloseScores()
{
    scoresScreen.SetActive(false);
}
public void OpenSettings()
{
    settingsScreen.SetActive(true);
}
public void CloseSettings()
{
    settingsScreen.SetActive(false);
}
public void OpenLevels()
{
    levelsScreen.SetActive(true);
}
public void CloseLevels()
{
    levelsScreen.SetActive(false);
}

```

Метод ExitGame() відповідає за логіку виходу із гри.

```

public void ExitGame()
{

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Application.Quit();
    }

```

У режимі «Play by plot» рух гравця постійно перевіряється та опрацьовується у методі Update(). Цей метод відповідає за актуалізацію стану гри та обробку дій, здійснюваних гравцем на основі отриманої від нього вхідної інформації.

```

void Update()
{
    isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance,
    groundMask);
    if(isGrounded && velocity.y < 0)
    {
        velocity.y = -2f;
    }
    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 move = transform.right * (-x) + transform.forward*(-z);
    controller.Move(move * speed * Time.deltaTime);
    velocity.y += gravity * Time.deltaTime;
    controller.Move(velocity * Time.deltaTime);
    if (Input.GetButtonDown("Jump") && isGrounded)
    {
        velocity.y = Mathf.Sqrt(jumpHeight * -2 * gravity);
    }
}

```

Зміна зброї у режимі «Play by plot» відбувається у методі SelectWeapon().

```

void SelectWeapon()
{
    int i = 0;
    foreach(Transform weapon in transform)
    {
        if(i == selectedWeapon)
        {
            weapon.gameObject.SetActive(true);
        }
        else
        {
            weapon.gameObject.SetActive(false);
        }
        i++;
    }
}

```

У режимі «Play by plot» вороги створюються за допомогою патерну фабрика. У класі EnemyFactory є наступні поля:

- simpleEnemyPrefab: посилання на префаб простого ворога;
- bossEnemyPrefab: посилання на префаб сильного ворога;
- withGunEnemyPrefab: посилання на префаб ворога зі зброєю;
- spawnPoints: масив, який містить посилання на точки, де можуть з'являтися вороги.

```

public class EnemyFactory : MonoBehaviour
{
    public GameObject simpleEnemyPrefab;
    public GameObject bossEnemyPrefab;
    public GameObject withGunEnemyPrefab;
    public GameObject[] spawnPoints;
    public void CreateEnemy(EnemyType enemyType)
    {
        GameObject prefab = null;
        switch (enemyType)
        {
            case EnemyType.Simple:
                prefab = simpleEnemyPrefab;
                break;
            case EnemyType.Boss:
                prefab = bossEnemyPrefab;
                break;
            case EnemyType.WithGun:
                prefab = withGunEnemyPrefab;
                break;
            default:
                Debug.LogError("Unknown enemy type: " + enemyType);
                break;
        }
        if (prefab != null)
        {
            GameObject spawnPoint = spawnPoints[Random.Range(0, spawnPoints.Length)];
            Instantiate(prefab, spawnPoint.transform.position, Quaternion.identity);
        }
    }
}

```

Здоров'я головного персонажа у режимі «Play by plot» реалізоване через зміну значення слайдера під назвою «HealthBar».

```

public class PlayerHealthBar : MonoBehaviour
{
    public Slider slider;
    public void SetMaxHealth()
    {
        slider.maxValue = 100;
        slider.value = 100;
    }
    public void SetHealth(float health)
    {
        slider.value = health;
    }
}

```

Відображення здоров'я ворога реалізоване подібним чином як і здоров'я персонажа: спочатку задається початкове значення здоров'я у методі StartHealthBar(), а далі при зміні кількості здоров'я викликається метод UpdateHealthBar(), який змінює значення у слайдері.

```

public class FloatingEnemyHealthBar : MonoBehaviour
{
    [SerializeField] private Slider slider;
    public void UpdateHealthBar(int currentHealth, int maxHealth)
    {
        slider.value = currentHealth;
    }
}

```

```

public void StartHealthBar(int maxHealth)
{
    slider.maxValue = maxHealth;
    slider.minValue = 0;
}
}

```

Міні-карта з відображенням головного героя та ворогів здійснюється за допомогою постійного оновлення позиції головного гравця на карті, щоб забезпечити коректне відображення його місцезнаходження, а також для уникнення виходу гравця за межі видимої області карти.

```

void LateUpdate()
{
    Vector3 newPosition = player.position;
    newPosition.y = transform.position.y;
    transform.position = newPosition;
    transform.rotation = Quaternion.Euler(90f, player.eulerAngles.y, 180f);
}

```

Метод Hit() відповідає за взаємодію головного героя з ворогом, зокрема за удар ворогу по гравцю. Якщо головний герой зіштовхнувся з ворогом, відбувається наступне:

- зменшення запасу здоров'я: відбувається віднімання відповідної кількості очків здоров'я від поточного значення запасу здоров'я героя;
- візуальне відображення пошкодження: з'являється напівпрозоре фото, яке створює ефект пошкодження.

```

public void Hit(int damage)
{
    health -= damage;
    playerHealthText.text = health.ToString();
    playerHealthBar.SetHealth(health);
    if (health <= 0)
    {
        injuredScreen.SetActive(false);
        gameOverScreen.SetActive(true);
        gameManagerObject.GetComponent<GameManagerScript>().SaveLevelScores();
        Cursor.lockState = CursorLockMode.None;
        Time.timeScale = 0f;
    }
    else
    {
        InjuredScreen();
    }
}

```

Постійно оновлюється наявність влучання у ворога через промінь зіткнення. Якщо ворог має зброю та знаходиться у 70 умовних одиниць

дистанції, тоді відбувається пряме влучання у головного персонажа кожні 5 секунд.

```
void Update()
{
    if (Input.GetMouseButtonDown(0) || Input.GetKeyDown(KeyCode.F))
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit))
        {
            if (hit.collider.gameObject == gameObject)
            {
                TakeDamage(damage);
            }
        }
    }
    float distance = Vector3.Distance(player.transform.position,
gameObject.transform.position);
    GetComponent<NavMeshAgent>().destination = player.transform.position;
    if (GetComponent<NavMeshAgent>().velocity.magnitude > 1)
    {
        enemyAnimator.SetBool("isRunning", true);
    }
    else
    {
        enemyAnimator.SetBool("isRunning", false);
    }
    if(canShoot == true)
    {
        shootTimer += Time.deltaTime;
        if (shootTimer >= shootInterval)
        {
            if (canShoot == true && distance <= 70)
            {
                ShootInPlayer(damage);
            }
            shootTimer = 0f;
        }
    }
}

public virtual void ShootInPlayer(int amount)
{
    enemyShootingSound.Play();
    player.GetComponent<PlayerManagerByPlot>().Hit(amount);
}
```

У залежності від застосованої зброї ворогу наноситься різна сила удару. Метод TakeDamage() відповідає за зменшення запасу здоров'я ворога після отримання удару.

```
public virtual void TakeDamage(int amount)
{
    int activeWeaponIndex = GetActiveWeaponIndex();
    if(activeWeaponIndex == 0)
    {
        health -= amount;
    }
    else if(activeWeaponIndex == 1)
    {
        health -= amount+5;
    }
}
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

else if (activeWeaponIndex == 2)
{
    health -= amount + 10;
}
else if (activeWeaponIndex == 3)
{
    health -= amount + 20;
}
else if (activeWeaponIndex == 4)
{
    health -= amount + 30;
}
else if (activeWeaponIndex == 5)
{
    health -= amount + 50;
}
else
{
    health -= amount;
}
healthBar.UpdateHealthBar(health, maxHealth);
if (health <= 0)
{
    gameManagerObject.GetComponent<GameManagerScript>().AddScores(10);
    Die();
}
}

```

Якщо запас здоров'я ворога вичерпується, спрацьовує метод Die(): спочатку викликається метод OneEnemyDied() з об'єкта gameManagerObject, після цього видаляється відповідний об'єкт ворога.

```

protected virtual void Die()
{
    gameManagerObject.GetComponent<GameManagerScript>().OneEnemyDied();
    Destroy(gameObject);
}

```

Режим гри «Multiplayer» починається із підключення до Photon Network, використовуючи налаштування, які були попередньо задані.

```

void Start()
{
    OpenLoadingScreen();
    loadingText.text = "Connecting to network...!";
    Time.timeScale = 1f;
    if (!PhotonNetwork.IsConnected)
    {
        PhotonNetwork.ConnectUsingSettings();
    }
    if (PhotonNetwork.IsConnected)
    {
        CloseLoadingScreen();
    }
}

```

Для того, щоб запустити багатокористувацьку гру необхідно викликати метод StartGame(), який використовує Photon Network для завантаження рівня гри з ідентифікатором levelToPlay. PhotonNetwork.LoadLevel() виконує

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

асинхронне завантаження рівня гри на всіх клієнтах, підключених до Photon Network. Коли завантаження завершується, клієнти автоматично переходять на вказаний рівень гри.

```
public void StartGame()
{
    CloseLoadingScreen();
    PhotonNetwork.LoadLevel(levelToPlay);
}
```

Метод SetPlayerName() відповідає для встановлення імені гравця в багатокористувацьку гру. Якщо поле введення імені не є порожнім, встановлюється ім'я гравця в Photon Network. Значення з поля введення nameInput.text присвоюється властивості NickName у Photon Network, щоб ідентифікувати гравця за його іменем.

```
public void SetPlayerName()
{
    if (!string.IsNullOrEmpty(nameInput.text))
    {
        PhotonNetwork.NickName = nameInput.text;

        PlayerPrefs.SetString("playerName", nameInput.text);

        hasSetPlayerName = true;
        CloseNameInputScreen();
    }
}
```

Метод CreateRoom() відповідає для створення кімнати у багатокористувацькій грі. Створюється новий об'єкт RoomOptions, який представляє параметри кімнати гри, йому встановлюється максимальна кількість гравців MaxPlayers у 6. Створення кімнати відбувається через передачу назви та налаштувань кімнати у PhotonNetwork.CreateRoom().

```
public void CreateRoom()
{
    if (!string.IsNullOrEmpty(roomNameInput.text))
    {
        RoomOptions roomOptions = new RoomOptions();
        roomOptions.MaxPlayers = 6;
        PhotonNetwork.CreateRoom(roomNameInput.text, roomOptions);
        CloseCreateRoomScreen();
        loadingText.text = "Creating room...";
        OpenLoadingScreen();
    }
}
```

Після того як гравець приєднується до створеної кімнати викликається метод OnJoinedRoom(), який виводить інформацію про кімнату: назву кімнати та імена учасників.

```

public override void OnJoinedRoom()
{
    CloseCreateRoomScreen();

    OpenRoomScreen();
    roomNameText.text = PhotonNetwork.CurrentRoom.Name;

    ListAllPlayers();

    if (PhotonNetwork.IsMasterClient)
    {
        startButton.SetActive(true);
    }
    else
    {
        startButton.SetActive(false);
    }
}

```

Відображення міні-карти у «Multiplayer» значно відрізняється від відображення у режимі «Play by plot». Спочатку знаходяться всі об'єкти у кімнаті із тегом «Player», після цього із знайдених об'єктів вибирається поточний гравець від розміщення якого оновлюється положення і обертання міні-карти.

```

void LateUpdate()
{
    players = GameObject.FindGameObjectsWithTag("Player");
    foreach (GameObject player in players)
    {
        int playerActorNumber =
        player.GetComponent<PhotonView>().Owner.ActorNumber;
        if (playerActorNumber == myID)
        {
            myPlayer = player;
        }

        if (myPlayer != null)
        {
            player = myPlayer.transform;
        }

        if (player != null)
        {
            Vector3 newPosition = player.position;
            newPosition.y = transform.position.y;
            transform.position = newPosition;
            transform.rotation = Quaternion.Euler(90f, player.eulerAngles.y, 180f);
        }
    }
}

```

Висновки до другого розділу

У другому розділі кваліфікаційної роботи було проведено аналіз варіантів використання та визначено об'єктно-орієнтовану структуру системи. Крім того, був розроблений дизайн-документ, який включає опис локацій та сцен,

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

персонажів, гейм-об'єктів та текстур, що використовуються в грі. Додатково, була надана детальна інформація щодо реалізації алгоритмів, що використовуються в грі.

У результаті проведених досліджень та розробок, було сформовано чітке бачення функціональних можливостей системи та її структуру, а також зроблено значний прогрес у написанні алгоритмів, які забезпечують роботу гри.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ

3.1. Інтерфейс та порядок роботи з програмним додатком

Після побудови 3D гри жанру шутер засобами Unity під Windows-платформу були отримані файли для запуску програмного додатку.

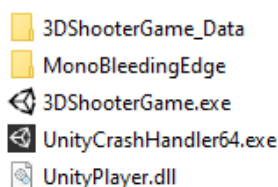


Рис. 3.1. Файли для запуску програмного додатку

Після запуску файлу 3DShooterGame.exe, користувач отримує доступ до головного меню комп'ютерної гри. Це меню включає шість кнопок, кожна з яких виконує певну функцію:

- «Play by Plot» - розпочинає гру за сюжетом, користувач буде проведений через захоплюючу одиночну кампанію та насичені бойовими пригодами подорі на різні планети;
- «Play Multiplayer» - ця кнопка дозволяє користувачу зайти в режим багатокористувацької гри та змагатись проти інших гравців онлайн;
- «About Game» - натиснувши цю кнопку, користувач отримає додаткову інформацію про саму гру: інформацію про розробника, особливості геймплею, різновиди ворогів, зброї та локацій;
- «Scores» - ця опція дозволяє користувачу переглянути найбільш високі набрані бали протягом всіх рівнів та у багатокористувацькій грі;
- «Settings» - натиснувши на цю кнопку, користувач зможе налаштувати різні параметри гри: налаштування графіки та звуку.
- «Exit» - ця кнопка дозволяє користувачу вийти з гри та повернутися до робочого столу.

Загальною метою головного меню є надання користувачу контролю над різними аспектами гри, починаючи від вибору режиму гри і закінчуючи налаштуваннями та виходом з програми.



Рис. 3.2. Головне меню

Після натискання кнопки «About Game», користувач отримує можливість ознайомитись з додатковою інформацією про гру. Тут він зможе дізнатись про розробника гри, а також рік розробки гри. Крім того, користувач зможе дізнатись про головну мету гри, яка описує основну ціль, що стоїть перед гравцем. Також користувач може отримати вичерпну інформацію про те, як управляти головним персонажем гри, включаючи вказівки щодо рухів і пострілів.

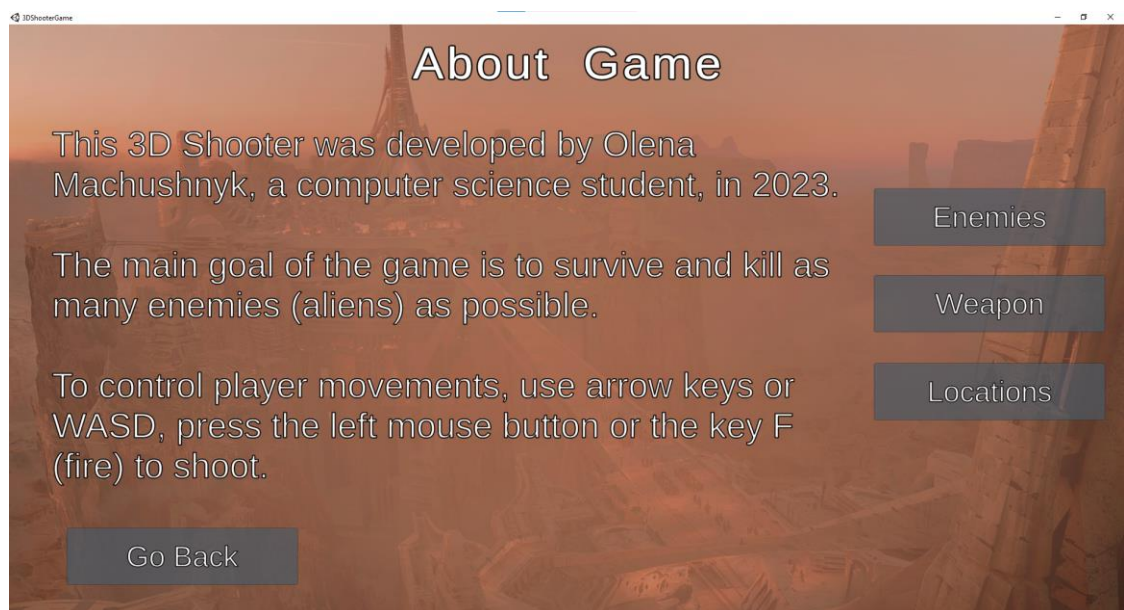


Рис. 3.3. Про гру у вікні «About Game»

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

У вікні «Enemies in Game» користувач може ознайомитись з інформацією про різних ворогів, що зустрічаються в грі, а саме їх опис, характеристики та особливості. Тут також вказані запас здоров'я кожного ворога, що вказує на його стійкість до пошкоджень, а також приблизна кількість пострілів, необхідних для його знищення.

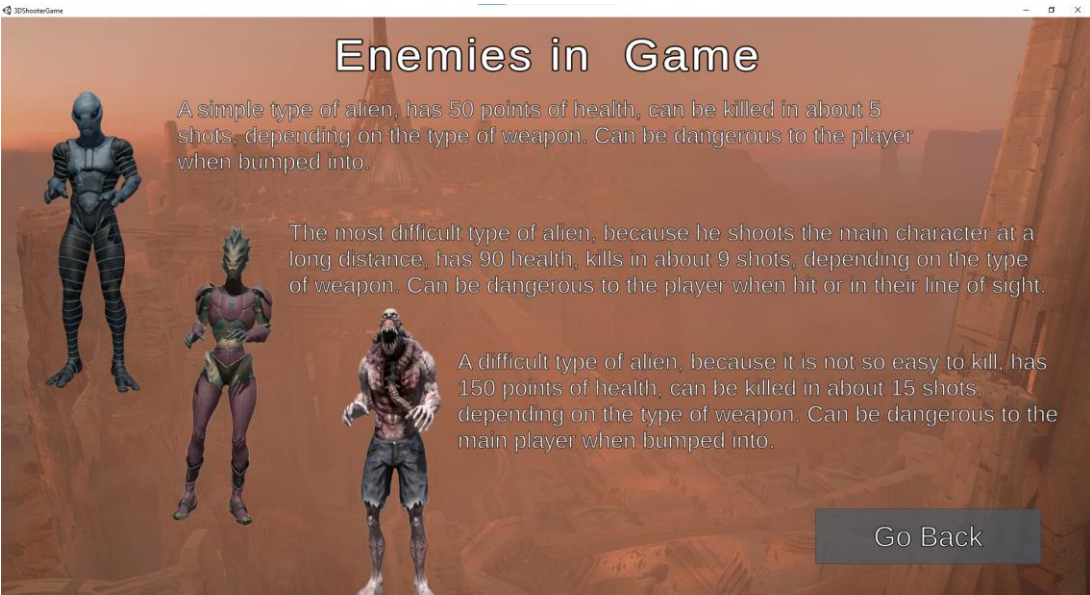


Рис. 3.4. Про гру: опис ворогів

У вікні «Weapon in Game» користувач може ознайомитись з інформацією про зброю: її силу пострілу та з якого рівня доступна та чи інша зброя.



Рис. 3.5. Про гру: опис зброї

У вікні «Locations in Game» користувач може ознайомитись з інформацією про локації, а саме кількість планет, їх назви та фото.



Рис. 3.6. Про гру: опис локацій

У вікні «Settings» користувач має можливість налаштувати рівень гучності музики, що відтворюється у фоновому режимі, а також рівень гучності звуків пострілів і якість графіки. Крім цього, користувач може видалити всі збережені дані на своєму комп'ютері: набрані бали всіх рівнів гри та набрані бали у мультиплеєрі.

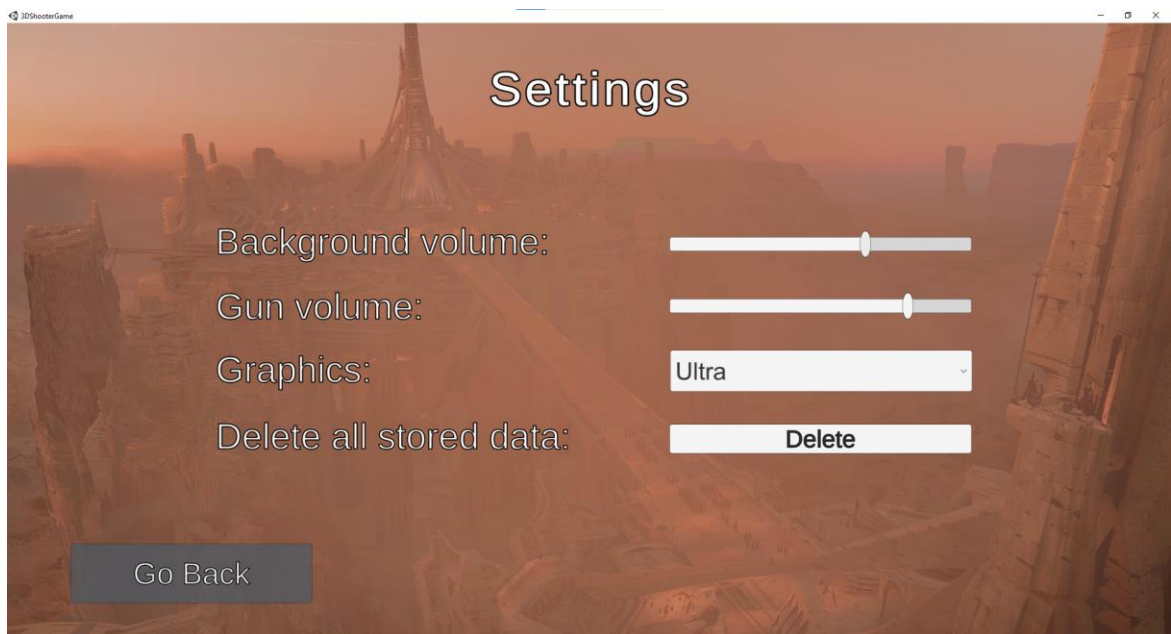


Рис. 3.7. Налаштування гри

Набрані бали користувача можна переглянути у вікні «Scores». Тут відображаються найвищі результати у кожному рівні гри, а також найбільші набрані бали у мультиплеєрі.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 3.8. Набрані бали: до видалення даних користувача

Після натиснення кнопки «Delete» у вікні налаштувань, всі дані користувача будуть повністю видалені. Це означає, що всі наявні дані та інформація, пов'язана з локальним користувачем комп'ютера, будуть остаточно втрачені та недоступні для відновлення.



Рис. 3.9. Набрані бали: після видалення даних користувача

Після натискання кнопки «Play by plot» у головному меню, відбувається відкриття вікна з вибором рівня, де перераховані всі наявні рівні гри. Спочатку доступний лише перший рівень, і гравець може почати гру саме з нього. Проте, по мірі успішного проходження попередніх рівнів, розблоковуються наступні рівні, надаючи гравцеві можливість грати на нових, більш складних рівнях.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		48

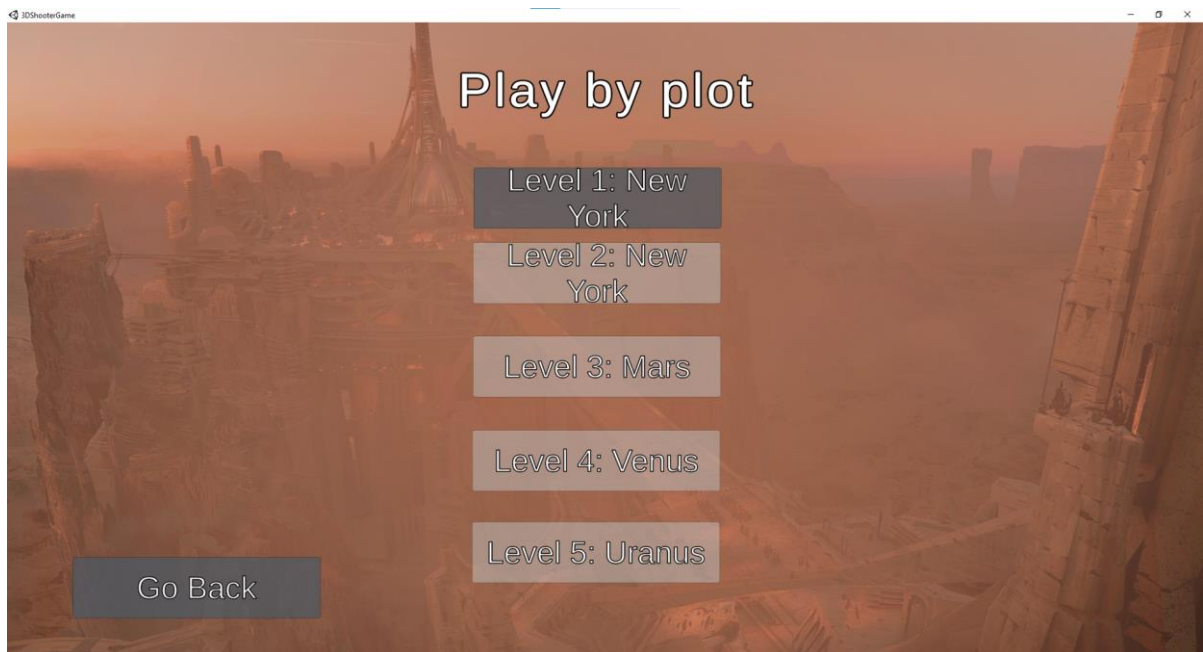


Рис. 3.10. Рівні гри: початковий доступний рівень

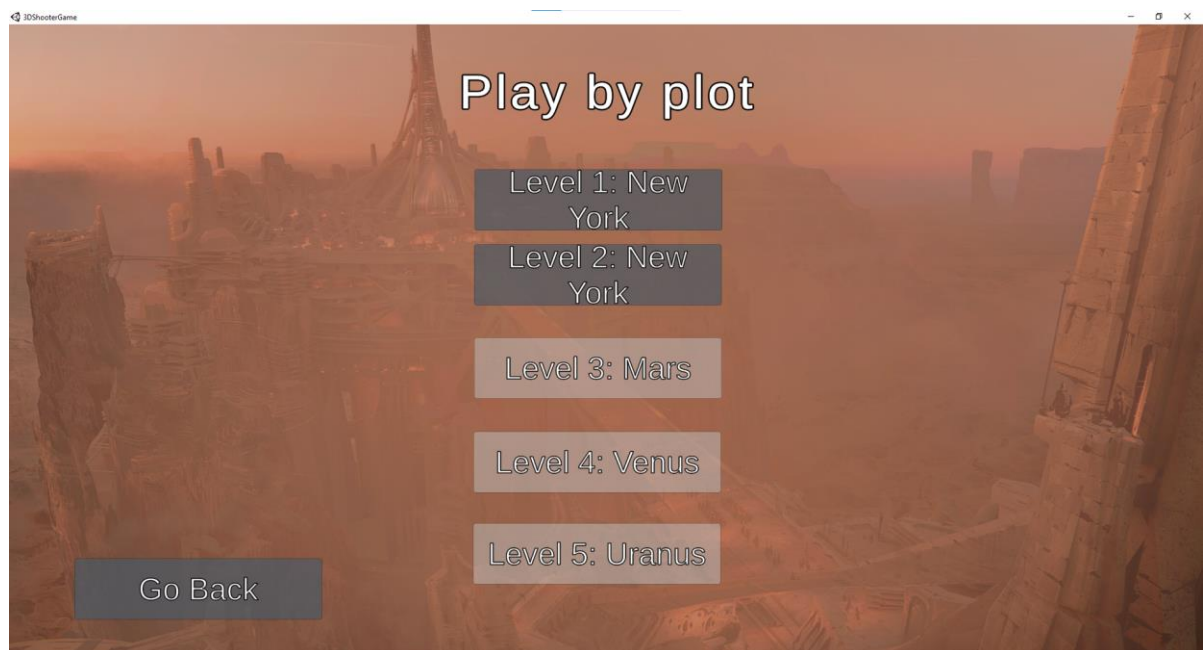


Рис. 3.11. Рівні гри: доступні рівні по мірі проходження гри

Після запуску першого рівня, на екрані починається опис місії та детальний опис обраної планети. Текст з'являється на екрані з ефектом, що нагадує друкарську машинку, додаючи більшого відчуття інтерактивності. У той же час, на задньому фоні з'являється 3D модель планети, яка виконує рух оберту навколо своєї осі. Цей захоплюючий візуальний ефект створює враження, що гравець потрапляє в самий центр подорожі, сприймаючи масштаб планети та її унікальні особливості.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

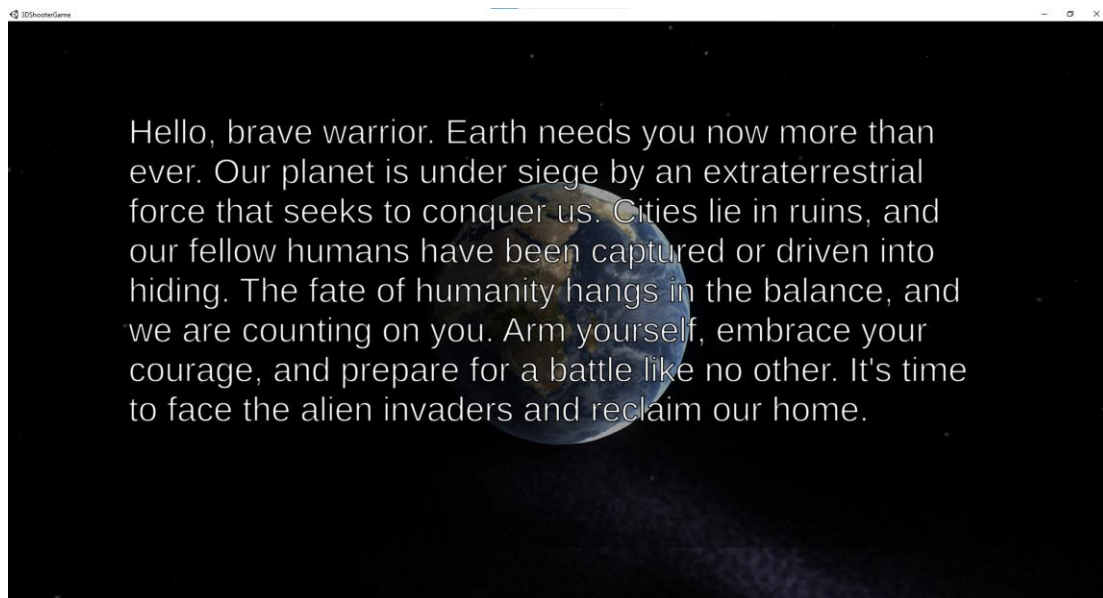


Рис. 3.12. Перший рівень: опис місії

Після того як користувач завершує читання опису місії, він відразу потрапляє у гру. У верхній частині екрана розміщений динамічний текст, який показує поточний раунд гри, кількість живих ворогів та кількість набраних балів. По центру верхньої частини екрана розташований слайдер, який відображає поточний стан здоров'я гравця. У нижній частині екрана справа знаходиться інтерактивна карта рівня, на якій відображається рух гравця та ворогів. Користувач може використовувати цю карту для орієнтації та планування своїх наступних кроків.

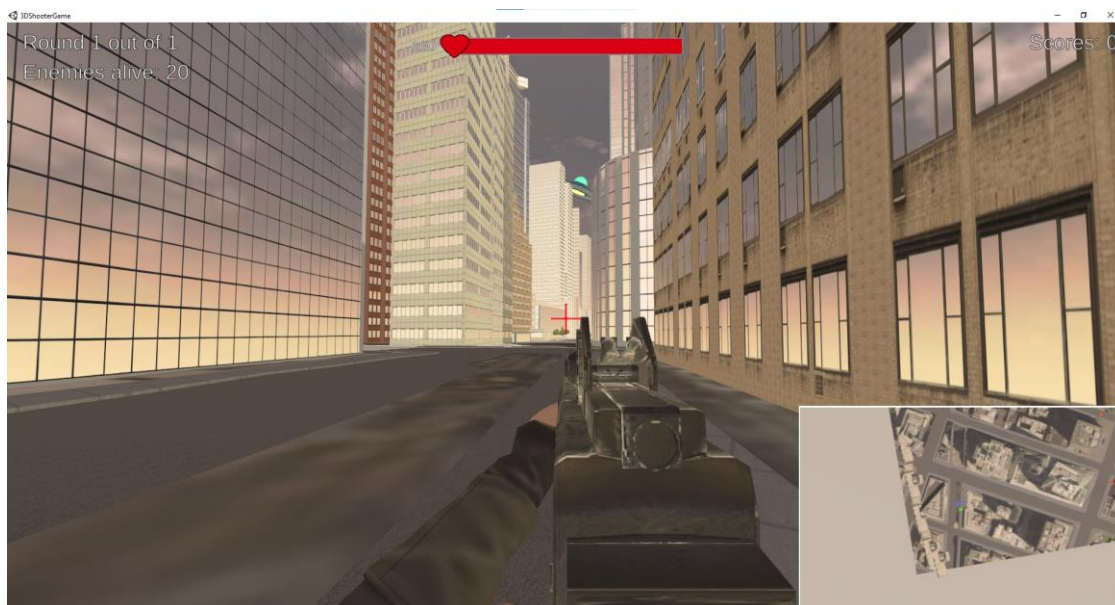


Рис. 3.13. Перший рівень: ігровий процес

Змн.	Арк.	№ докум.	Підпис	Дата

КН.КР.Б – 122 – 23 – ПЗ

Арк.

50

У кожному рівні, гравцеві надається можливість змінювати свою зброю шляхом використання прокрутки комп'ютерної миші. У першому рівні, гравцю доступні дві рушниці, які відрізняються силою пострілу.



Рис. 3.14. Перший рівень: зміна зброї

У грі існує різноманітність ворогів з різними способами атаки, які можуть завдати пошкодження головному персонажу. Деякі вороги становлять загрозу лише при прямому фізичному контакті з ними. Інші вороги, навпаки, здатні стріляти з власної рушниці на відстані, намагаючись вразити гравця.

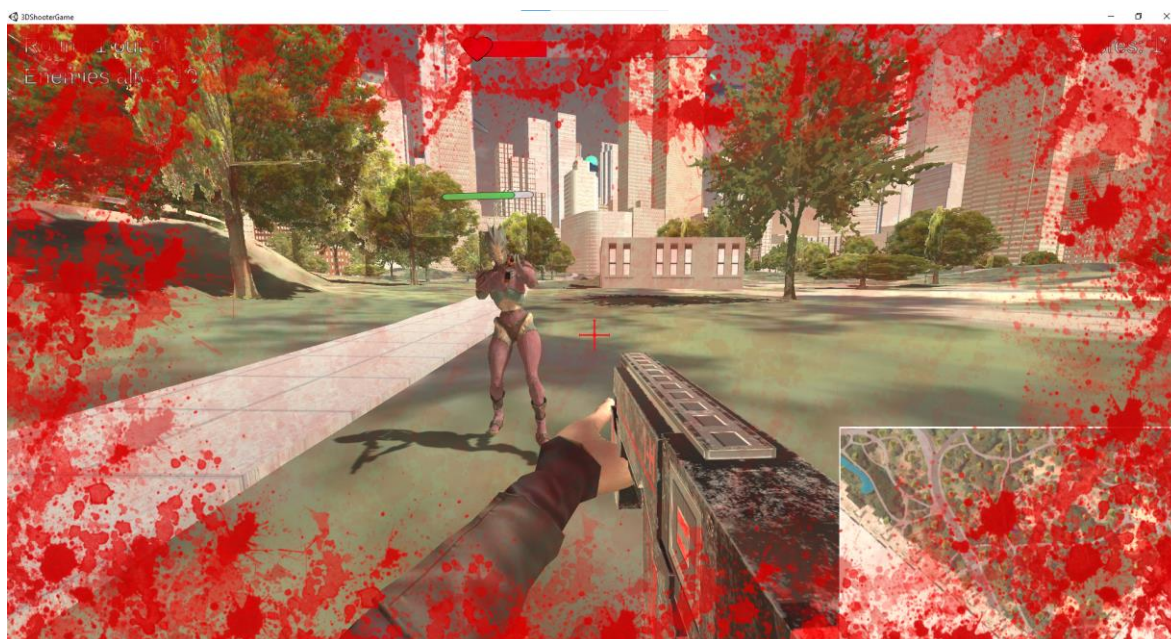


Рис. 3.15. Приклад ворога із власною рушницею

При отриманні поранення, на екрані відображається візуальний ефект: екран забарвлюється у червоний колір, що символізує кров головного героя. Цей ефект додає реалізму у геймплей, нагадуючи гравцеві про його стан здоров'я і важливість уникання подальших пошкоджень.

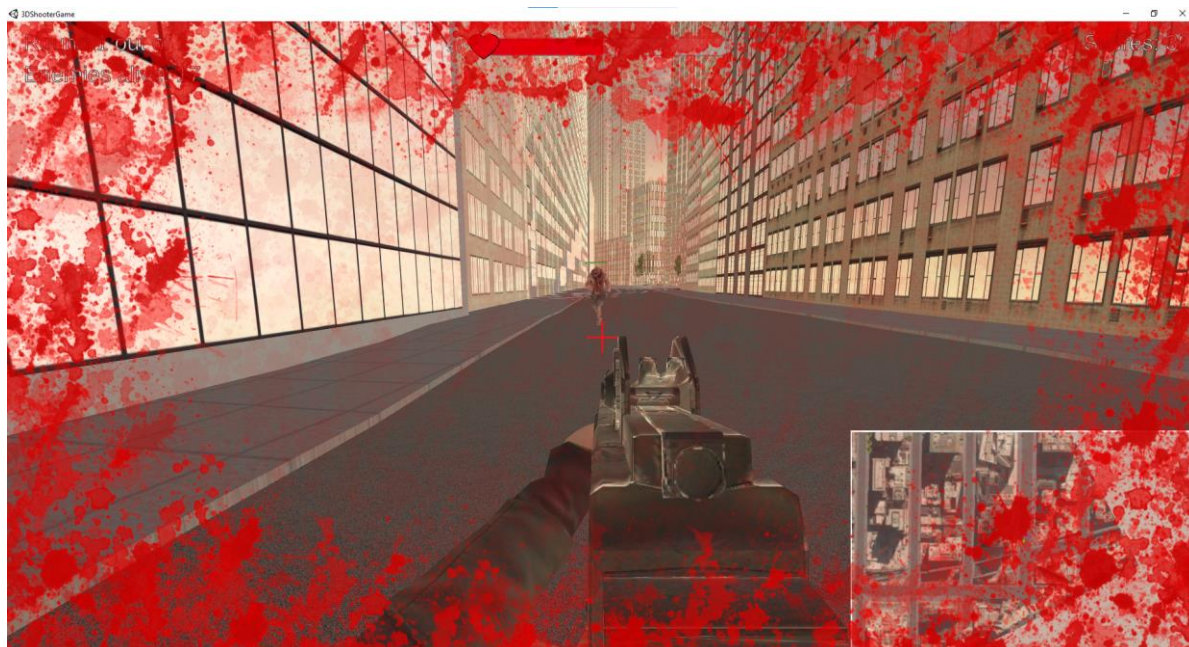


Рис. 3.16. Перший рівень: пошкодження головного персонажу

При нанесенні удару у ворога, слайдер здоров'я, що знаходиться над головою інопланетянина, зменшується з кожним пострілом, який досягає його. Цей слайдер демонструє, наскільки близько інопланетянин до досягнення поразки. Кожен постріл віднімає певну кількість очків здоров'я, яка залежить від сили удару застосованої зброї. Гравцю слід знизити слайдер здоров'я до нуля, щоб перемогти інопланетянина.

Важливо бачити який запас здоров'я залишився у ворога, адже саме інформація про здоров'я ворога допомагає розробити стратегію бою. Якщо ворог має низький рівень здоров'я, гравець може вибрати більш швидкі атаки, щоб швидше його знищити. У разі, коли у ворога залишилось багато здоров'я, гравцю може знадобитись більше обережності та стратегічного підходу, щоб уникнути ризику поразки.

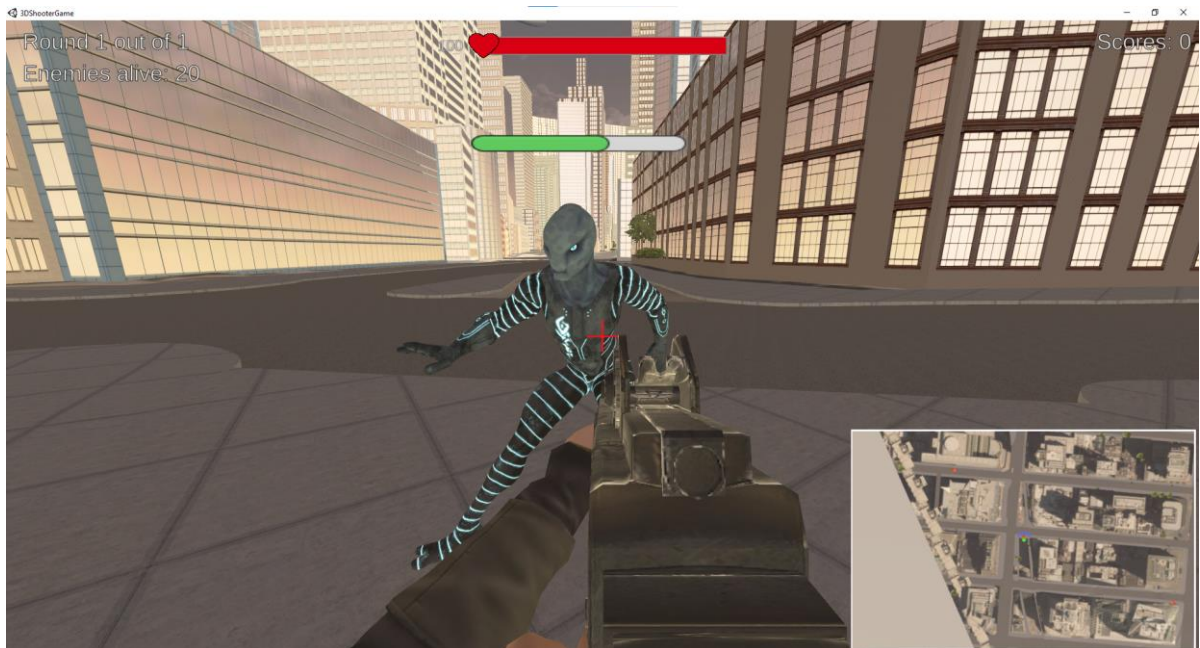


Рис. 3.17. Перший рівень: відображення здоров'я ворога

Якщо гравець натисне клавішу «Esc» на клавіатурі, то на екрані з'явиться вікно паузи, на якому доступні три кнопки:

- «Resume» - продовжує зупинений ігровий процес поточного рівня;
- «Start Again» - перезапускає поточний рівень із збереженням набраних балів;
- «Menu» - повертає гравця до початкового меню із збереженням набраних балів.

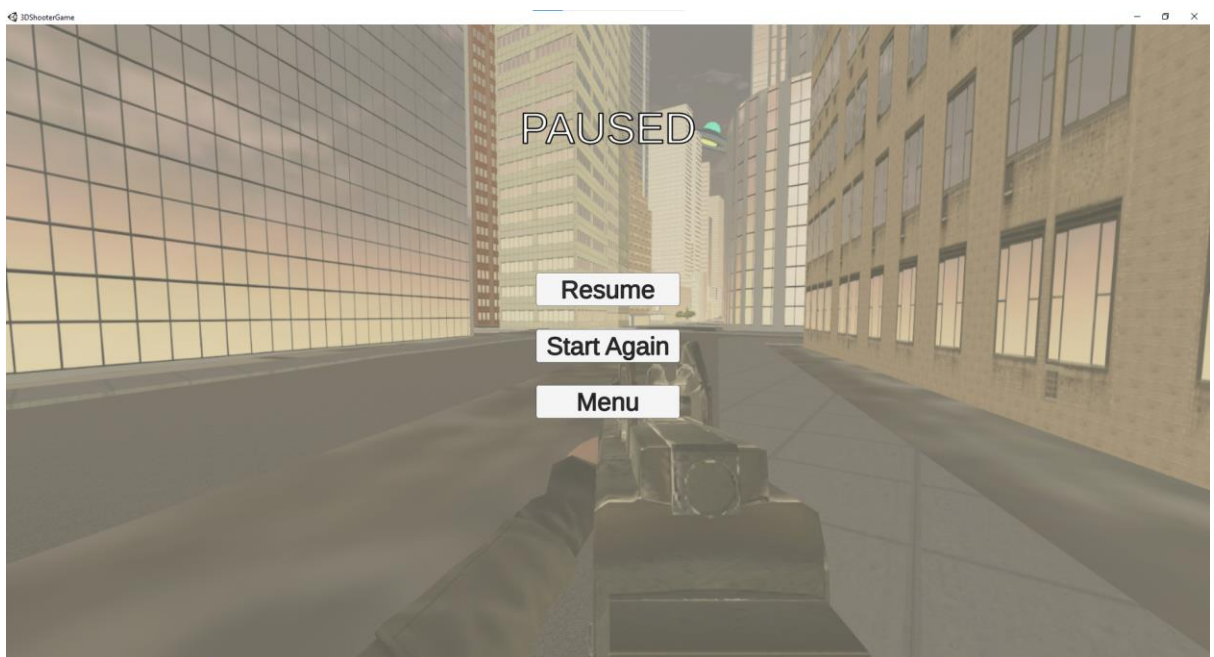


Рис. 3.18. Перший рівень: екран паузи

Якщо гравець використав увесь свій запас здоров'я, то з'являється екран програшу, повідомляючи про його поразку. На даному екрані відображається текст із набраними балами, кількістю виживших супротивників та номером раунда, також на екрані доступні дві кнопки:

- «Play Again» - перезапускає поточний рівень із збереженням набраних балів;
- «Menu» - повертає гравця до початкового меню із збереженням набраних балів.

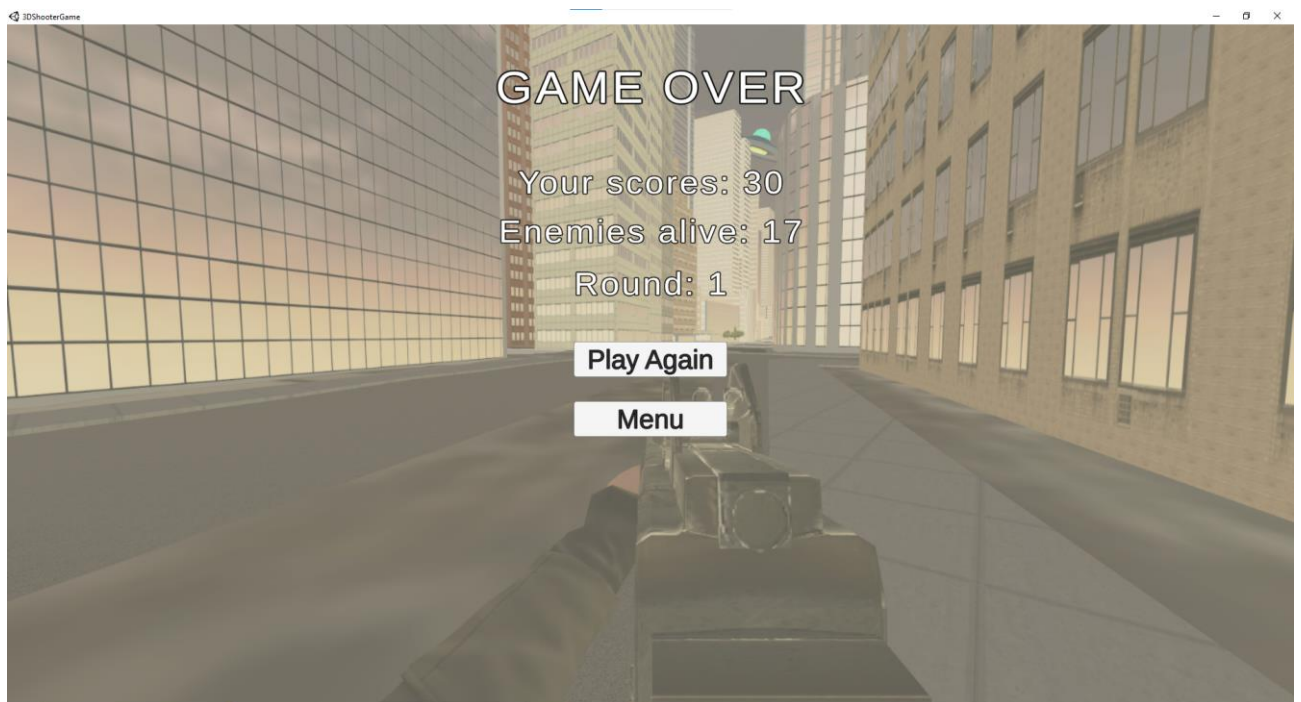


Рис. 3.19. Перший рівень: екран програшу

Якщо гравець успішно виконав місію відповідного рівня, на екрані з'являється екран успішного проходження місії, який свідчить про досягнення гравцем поставленої мети. На даному екрані відображається текст із набраними балами та номером раунда, також на екрані доступні три кнопки:

- «Next level» - запускає наступний рівень із збереженням набраних балів;
- «Replay level» - перезапускає поточний рівень із збереженням набраних балів;
- «Menu» - повертає гравця до початкового меню із збереженням набраних балів.



Рис. 3.20. Перший рівень: екран успішного проходження місії

Після успішного проходження наступних рівнів гри, гравець може перейти на вищий рівень. На початку всіх наступних рівнів, аналогічно як і в першому рівні, на екрані починається опис місії та детальний опис обраної планети.

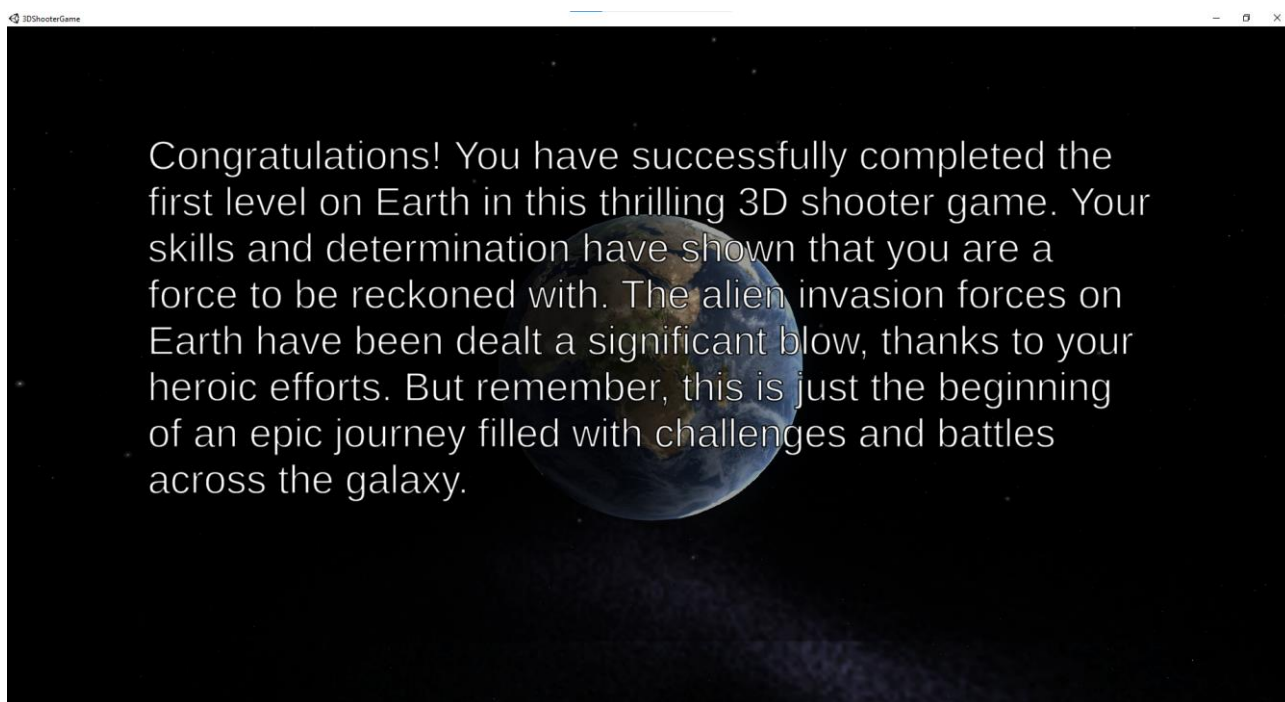


Рис. 3.21. Другий рівень: опис місії

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

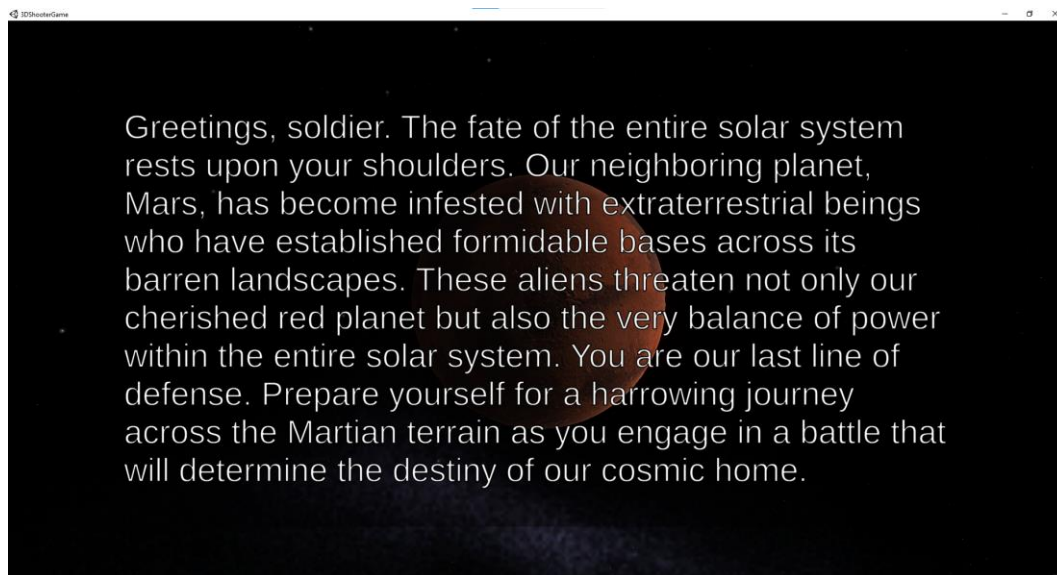


Рис. 3.22. Третій рівень: опис місії

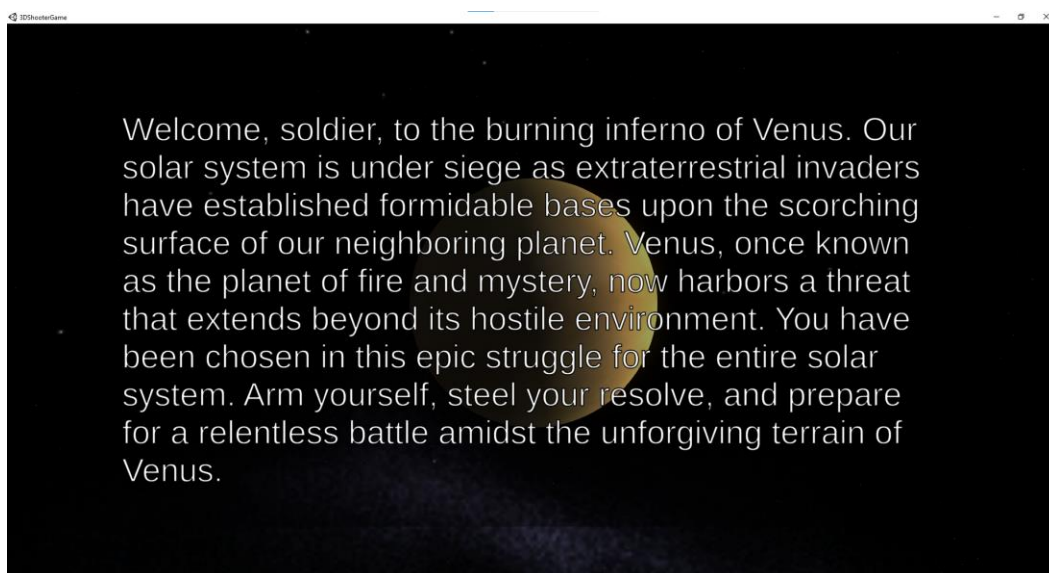


Рис. 3.23. Четвертий рівень: опис місії

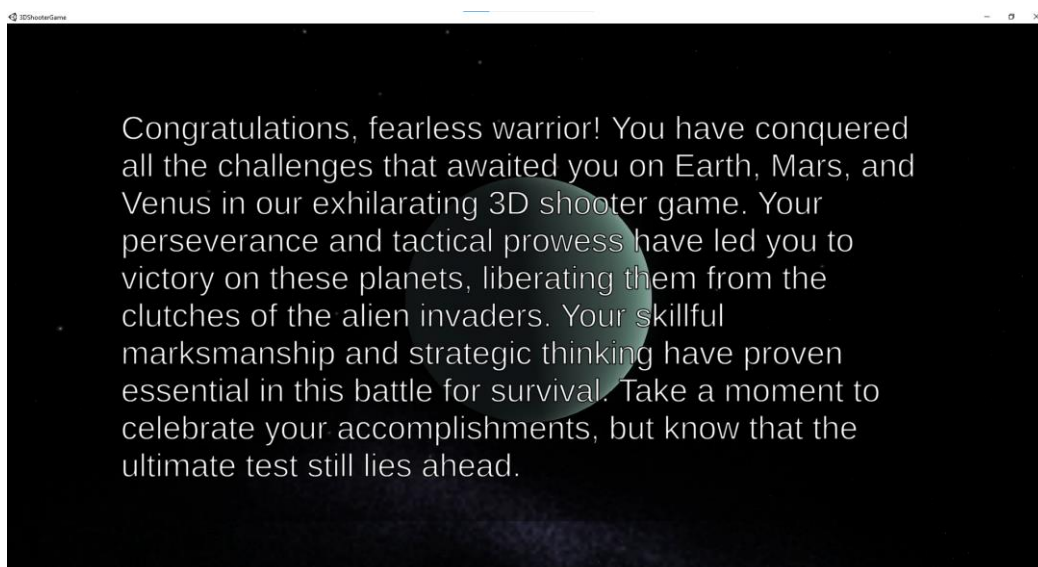


Рис. 3.24. П'ятий рівень: опис місії

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

У кожному наступному рівні гри гравець отримує можливість відкрити та використовувати більш потужну зброю, яка не була доступна раніше. Це створює прогресію в геймплеї та надає гравцю нові можливості для ефективного протистояння ворогам або виконання завдань.

У другому рівні додана рушниця із потужністю у 20 одиниць.



Рис. 3.25. Нова зброя у другому рівні

У третьому рівні додана рушниця із потужністю у 30 одиниць.



Рис. 3.26. Нова зброя у третьому рівні

У четвертому рівні додана рушниця із потужністю у 40 одиниць.



Рис. 3.27. Нова зброя у четвертому рівні

У п'ятому рівні додана рушниця із потужністю у 60 одиниць.

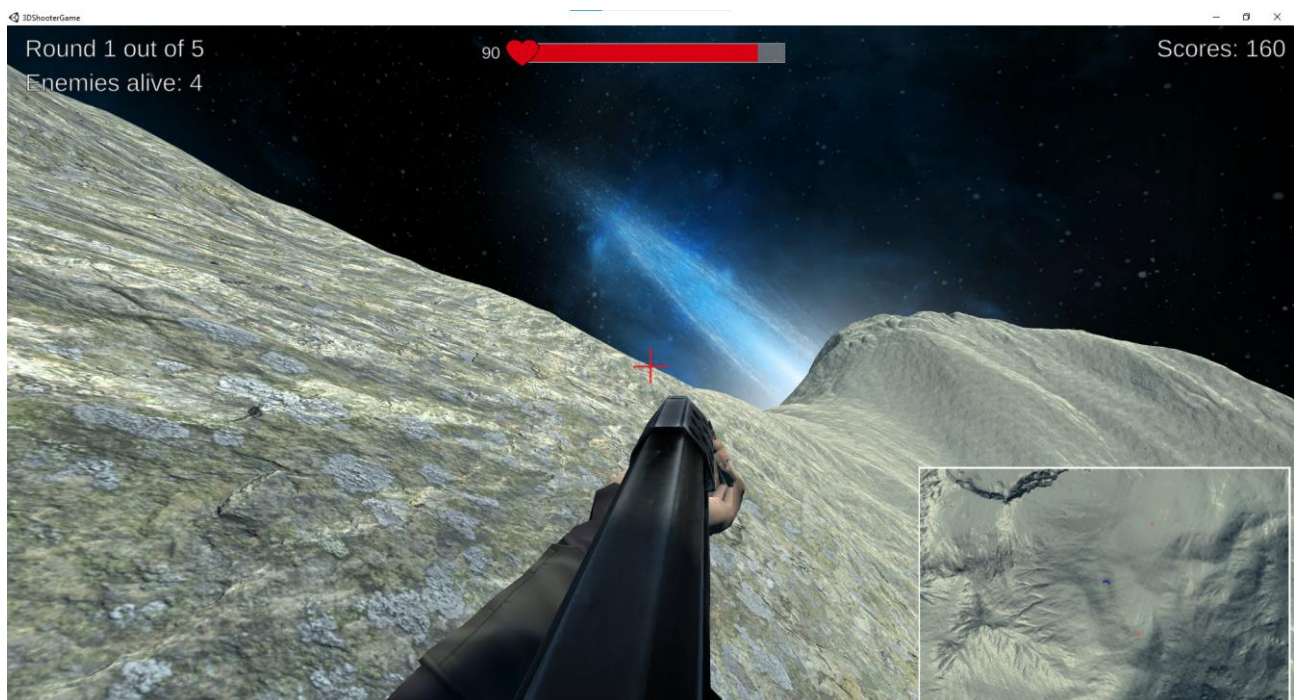


Рис. 3.28. Нова зброя у п'ятому рівні

У залежності від обраного рівня, кількість балів, необхідних для успішного проходження місії, значно варіюється. Це додає різноманітності та викликів до геймплею, а також стимулює гравця до досягнення кращих результатів.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

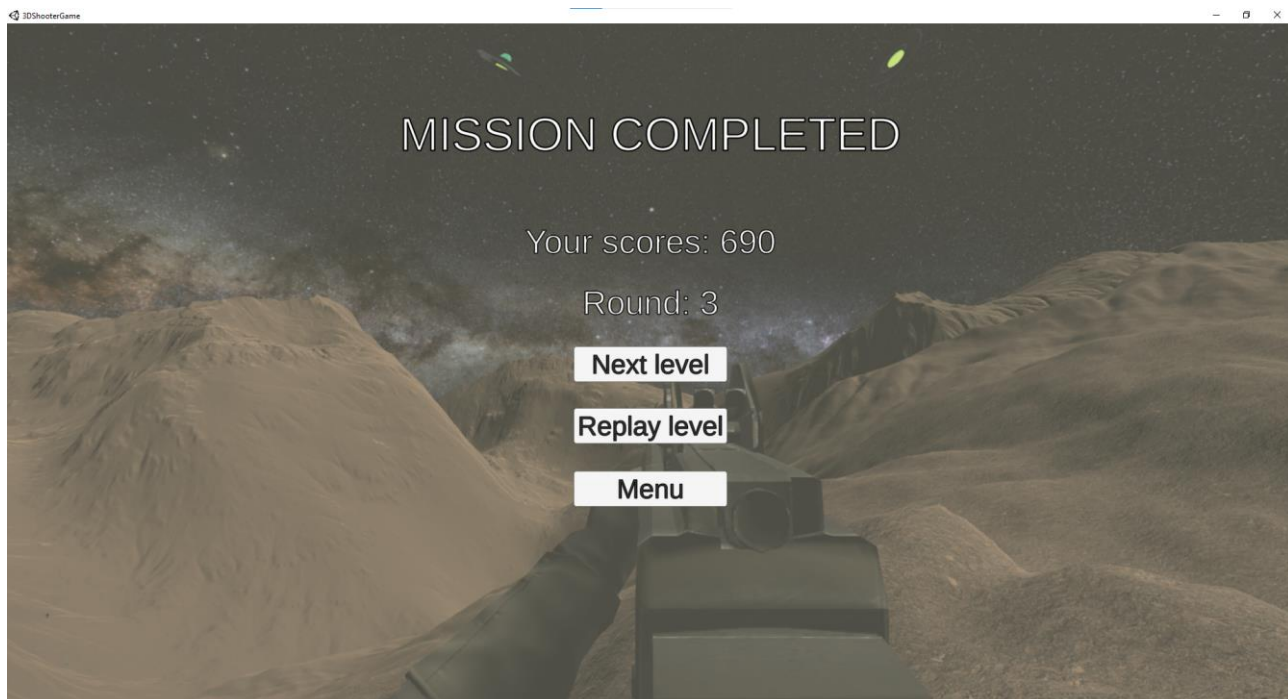


Рис. 3.29. Успішне проходження третього рівня: 690 балів



Рис. 3.30. Успішне проходження четвертого рівня: 980 балів

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

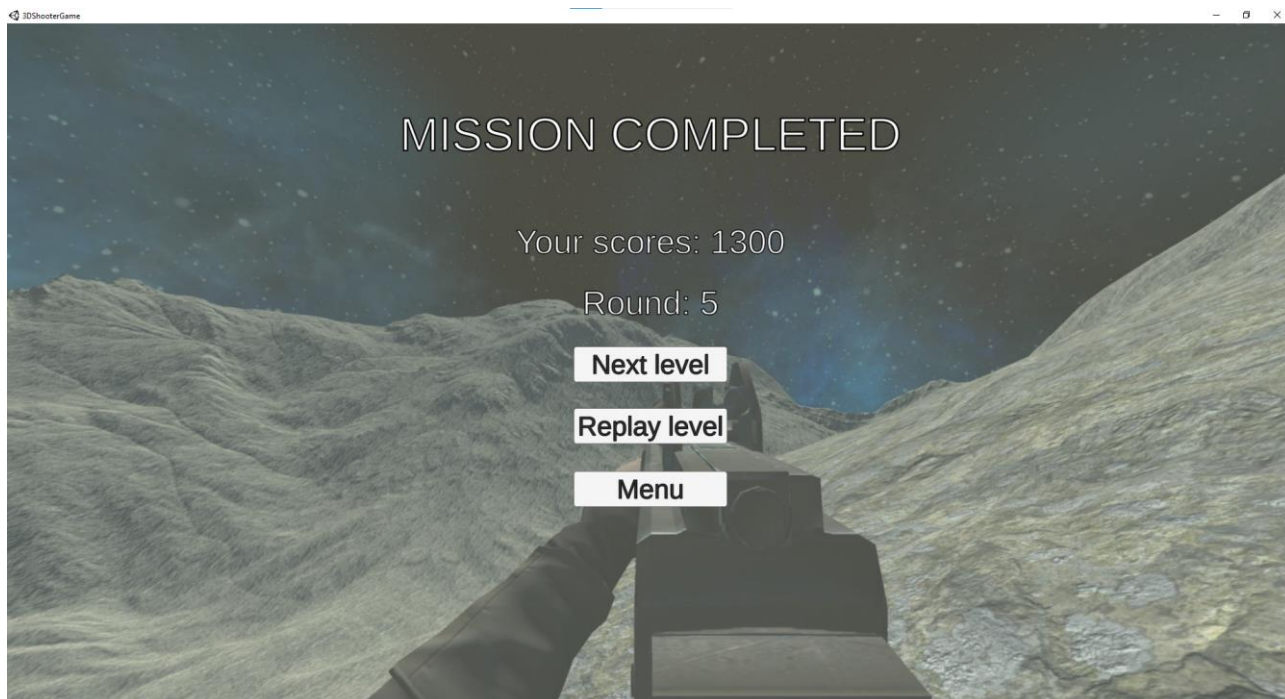


Рис. 3.31. Успішне проходження четвертого рівня: 1300 балів

Після вдалого проходження останнього п'ятого рівня гри гравець може прочитати привітання із успішним завершенням гри на фоні всіх планет. Це створює відчуття досягнення закінчення гри та задоволення від зусиль.

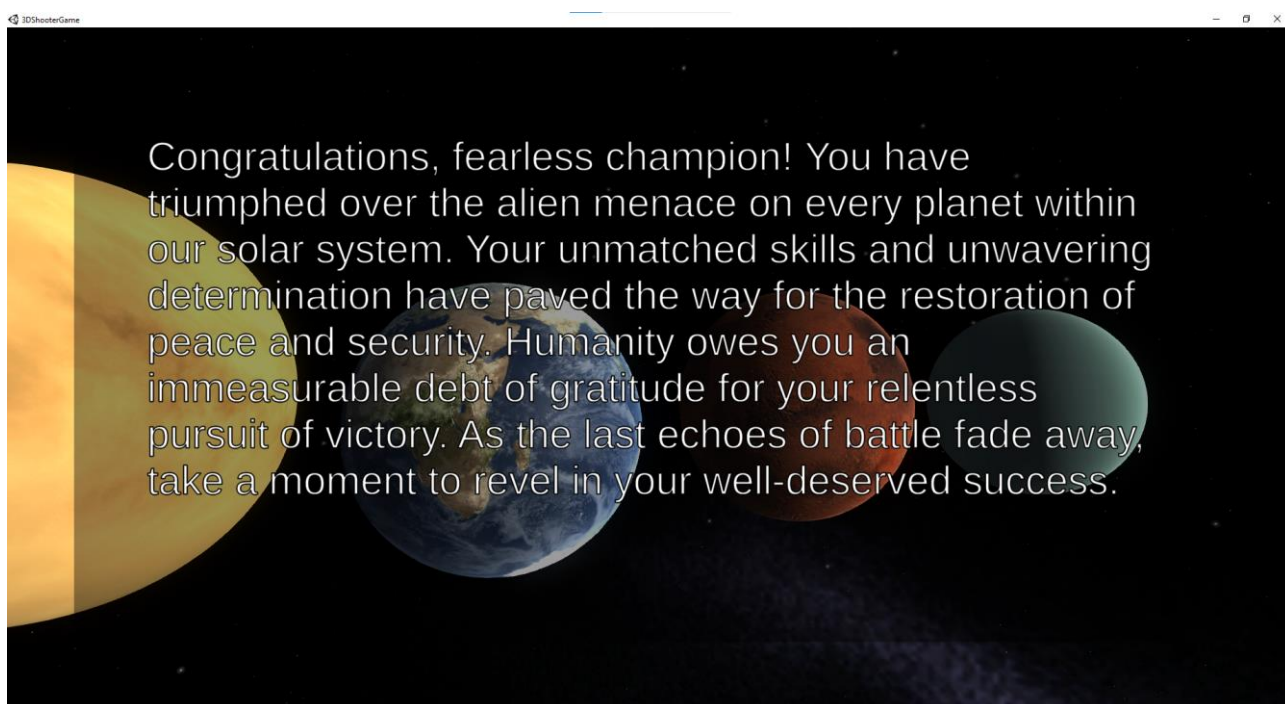


Рис. 3.32. Привітання після успішного проходження всіх рівнів

Після натиснення кнопки «Multiplayer» відкривається вікно введення імені користувача для багатокористувацької гри.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

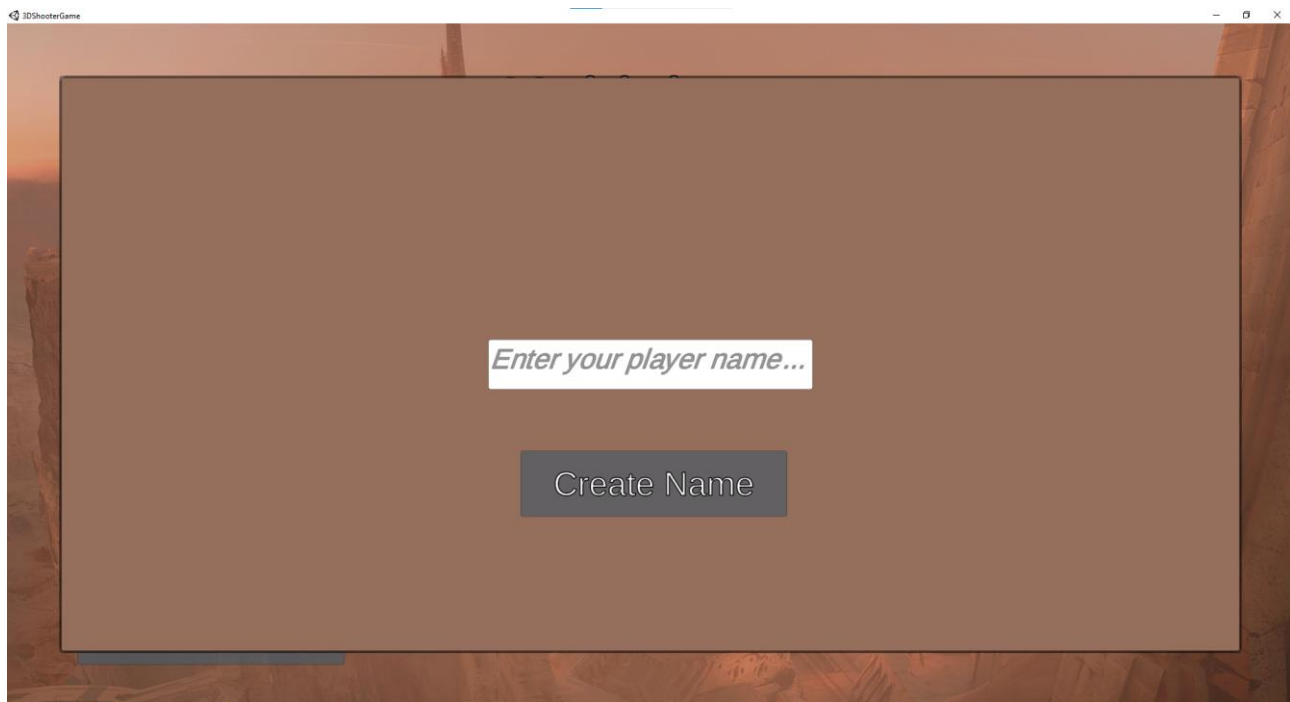


Рис. 3.33. Вікно введення імені користувача

Після введення імені користувача гравцю слід натиснути кнопку «Create Name» для збереження введеної інформації.

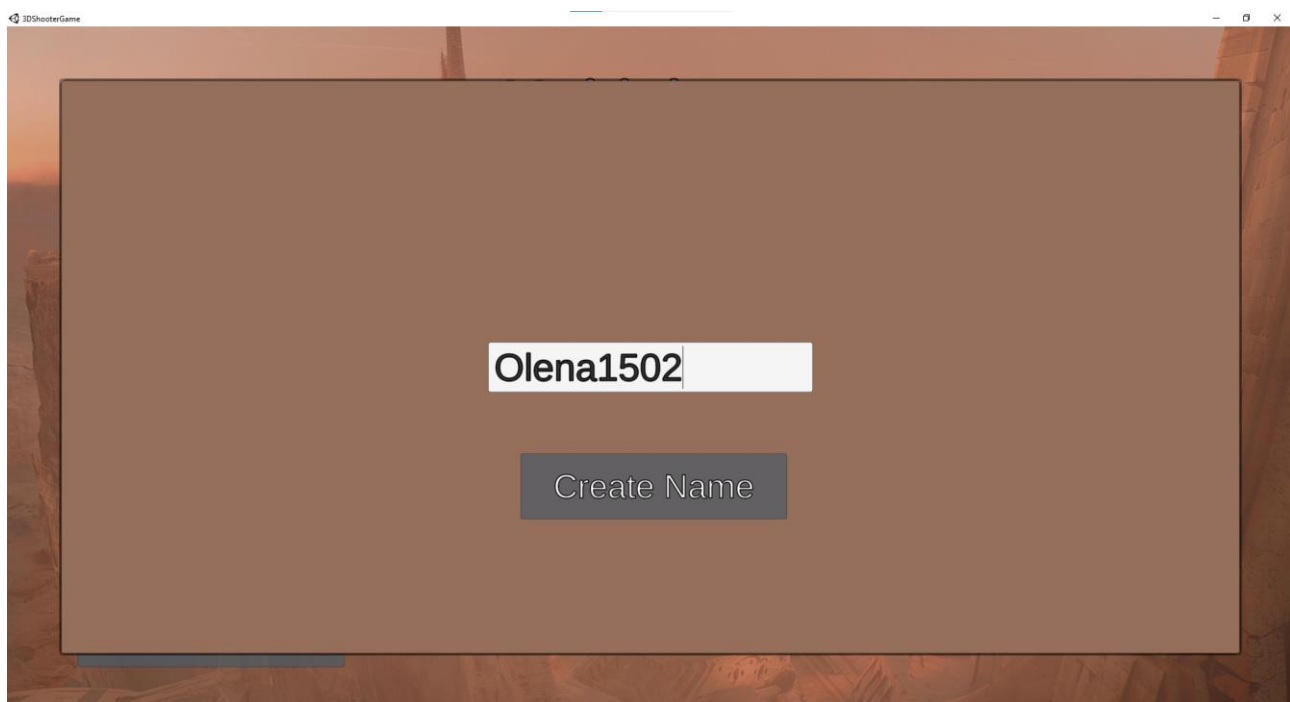


Рис. 3.34. Процес введення імені користувача

Після введення імені користувача, гравець може створити нову кімнату для багатокористувацької гри або знайти наявну кімнату.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

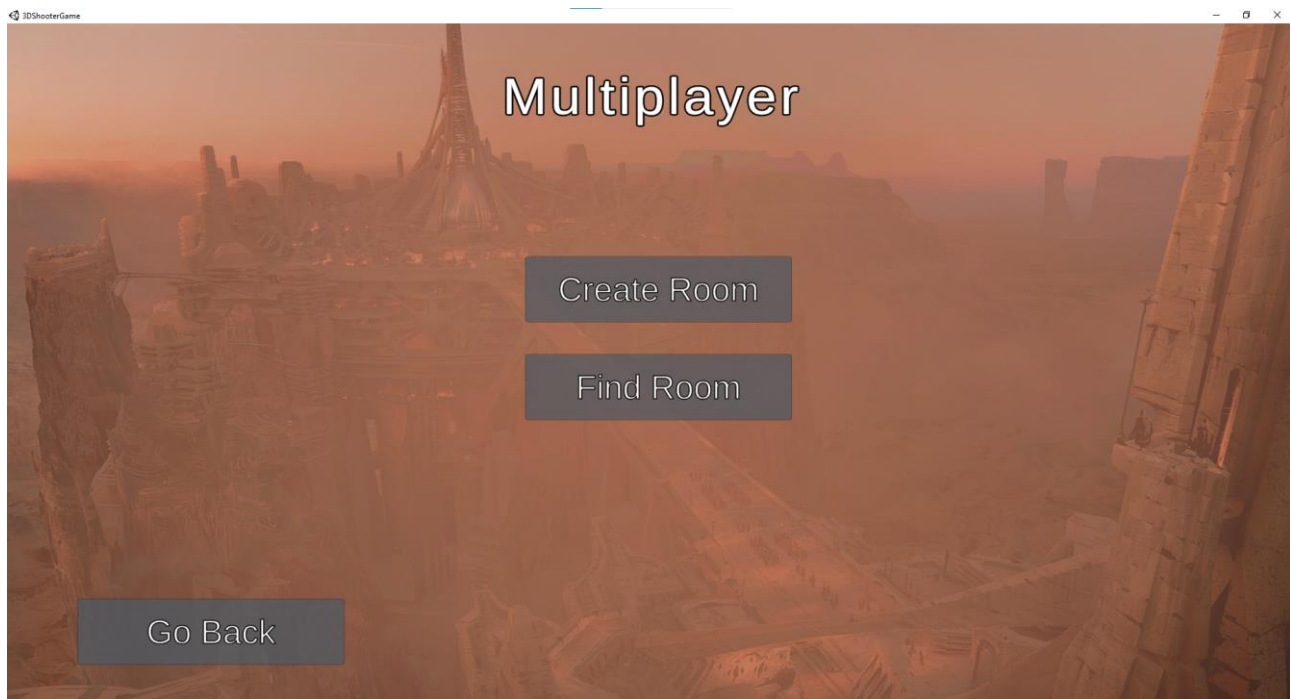


Рис. 3.35. Операції із кімнатами у «Multiplayer»

Після натиснення кнопки «Find Room» гравець отримує можливість переглянути список наявних багатокористувацьких кімнат. Якщо створених кімнат немає, то відображається порожній список.

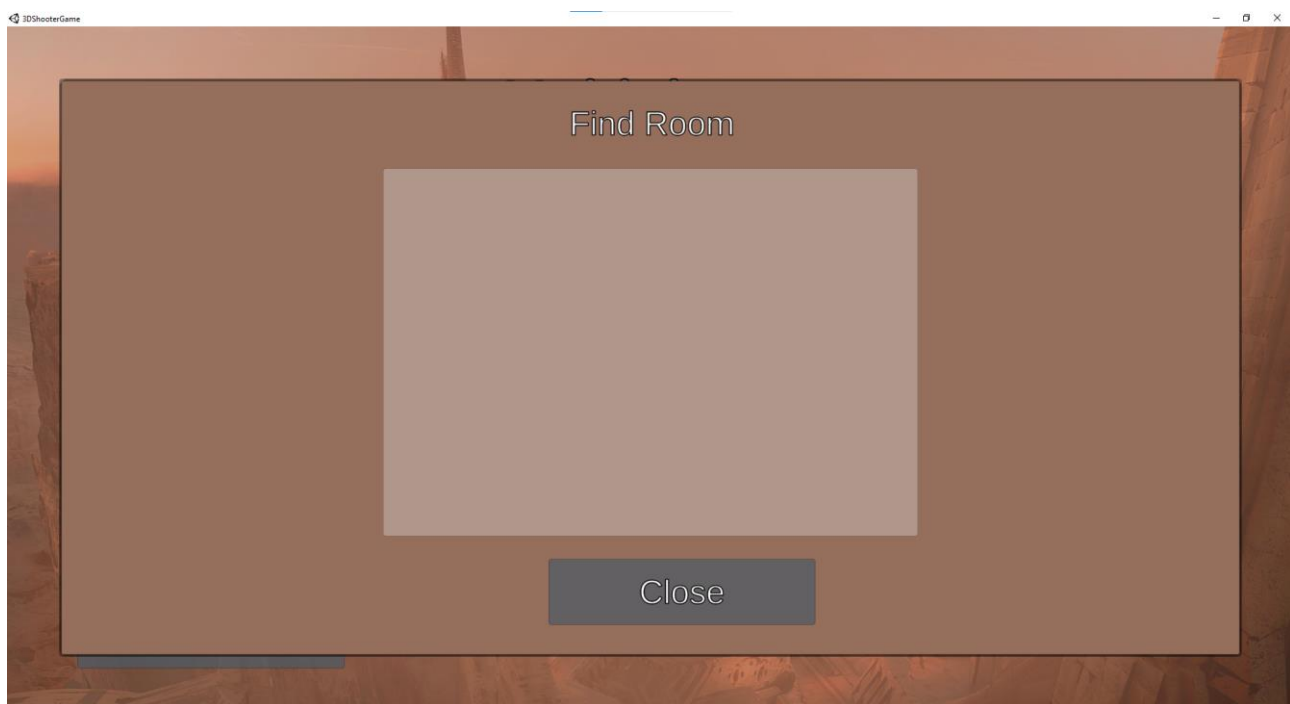


Рис. 3.36. Порожній список кімнат

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 3.37. Список кімнат

Для того, щоб створити нову кімнату потрібно натиснути кнопку «Create Room» та ввести назву кімнати.

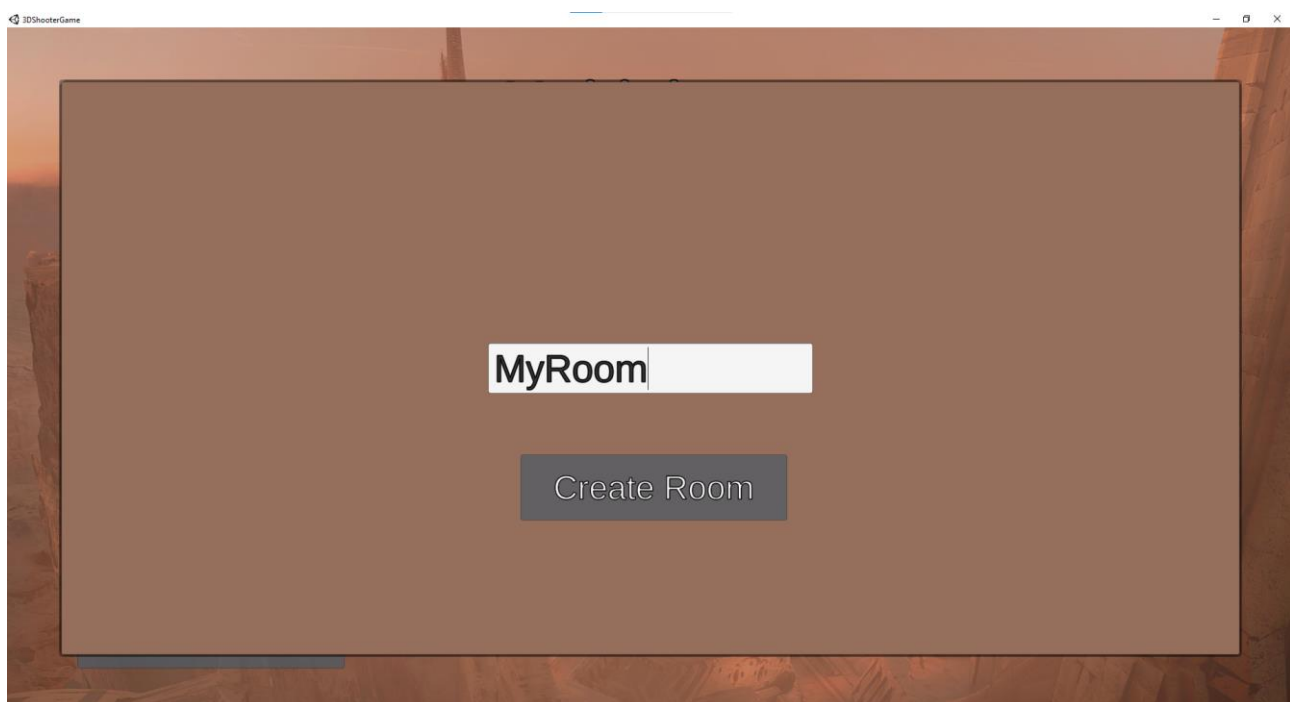


Рис. 3.38. Створення нової кімнати: введення назви

Після введення назви кімнати і натиснення на кнопку «Create Room» з'являється інформаційне вікно, яке повідомляє, що процес створення кімнати розпочався. Під час цього процесу система автоматично створює нову кімнату з введеною назвою і надає їй унікальний ідентифікатор.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

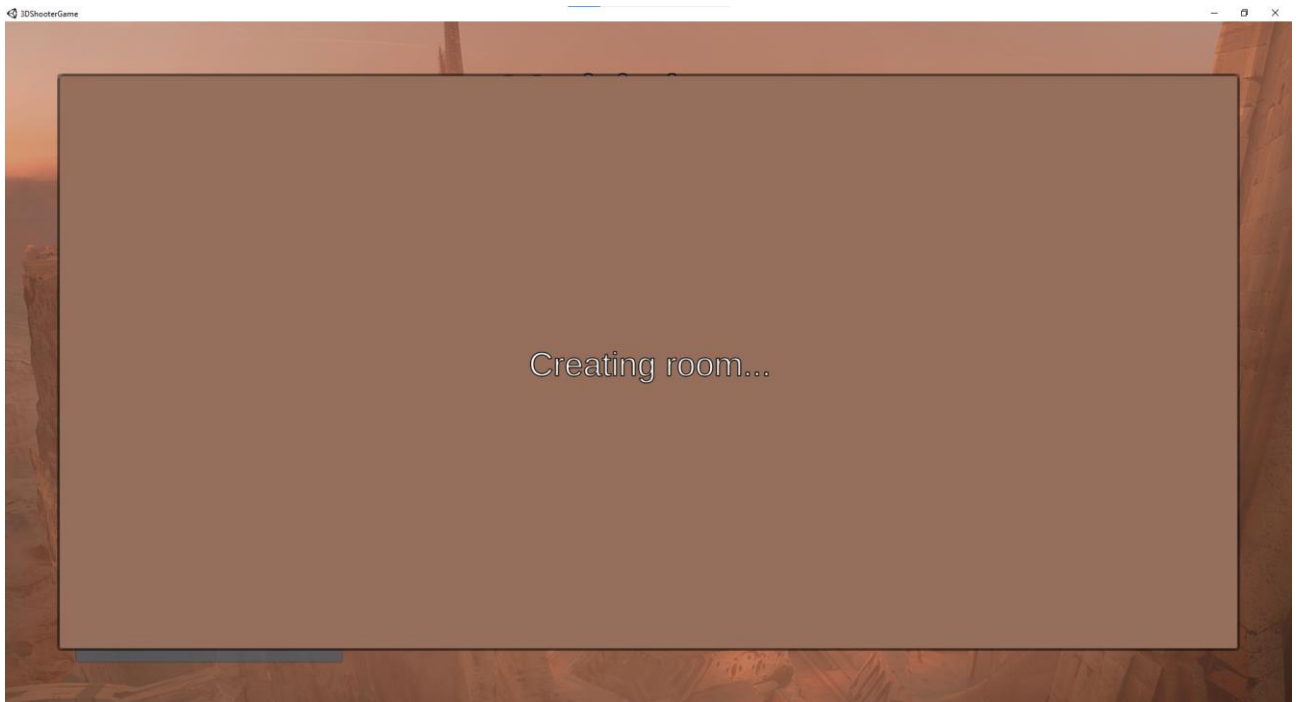


Рис. 3.39. Інформаційне вікно створення нової кімнати

Також наявна можливість перегляду імен користувачів, які перебувають у вибраній кімнаті на даний момент часу.



Рис. 3.40. Перелік користувачів у вибраній кімнаті

Після приєднання до відповідної кімнати гравець опиняється у багатокористувацькому середовищі. У верхній частині екрана розміщений динамічний текст, який показує кількість гравців у кімнаті та кількість набраних балів. По центру верхньої частини екрана розташований слайдер, який

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

відображає поточний стан здоров'я гравця. У нижній частині екрана справа знаходиться інтерактивна карта багатокористувацького середовища, на якій відображається рух всіх гравців у реальному часі.



Рис. 3.41. Ігровий процес у «Multiplayer»

У багатокористувацькій грі кожен гравець має можливість змінювати свою зброю шляхом використання прокрутки комп'ютерної миші. Користувачам відразу доступно 6 рушниць із різною силою влучання в противника.

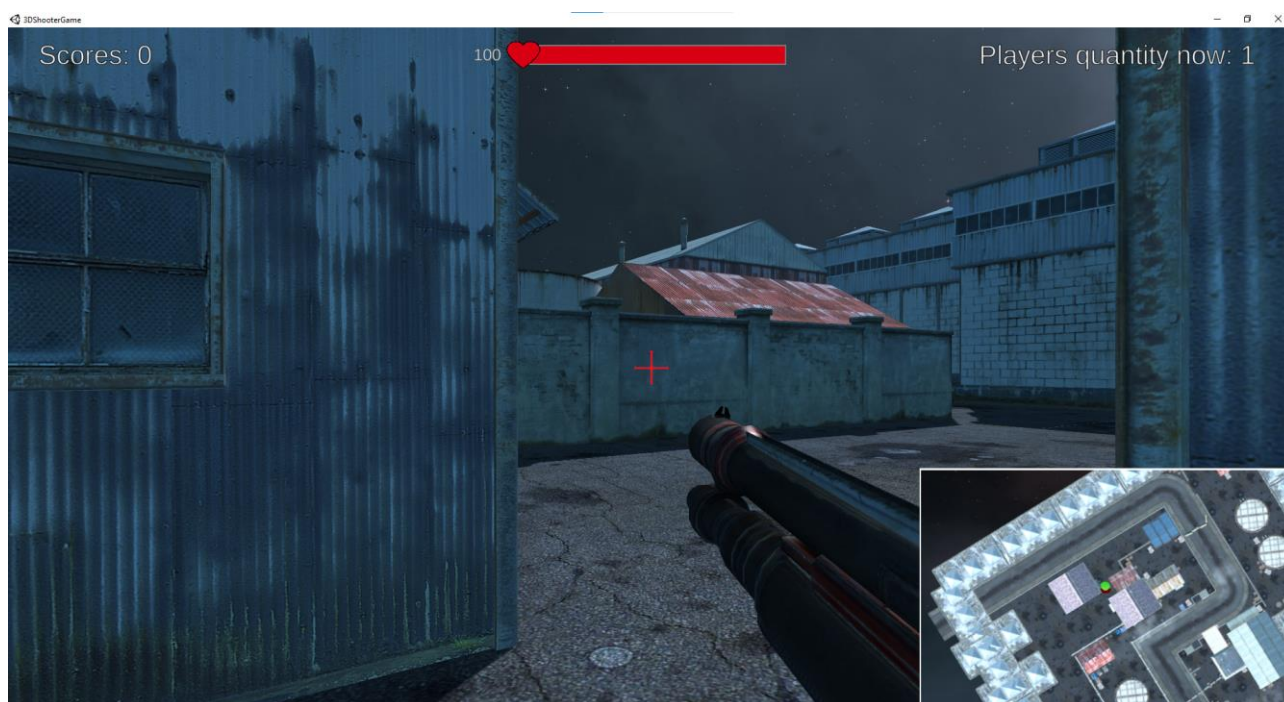


Рис. 3.42. Ігровий процес у «Multiplayer»: зміна зброї

Якщо інший гравець підключається до наявної кімнати, перед ним з'являється інформаційне вікно із інформацією про процес підключення до обраної кімнати.

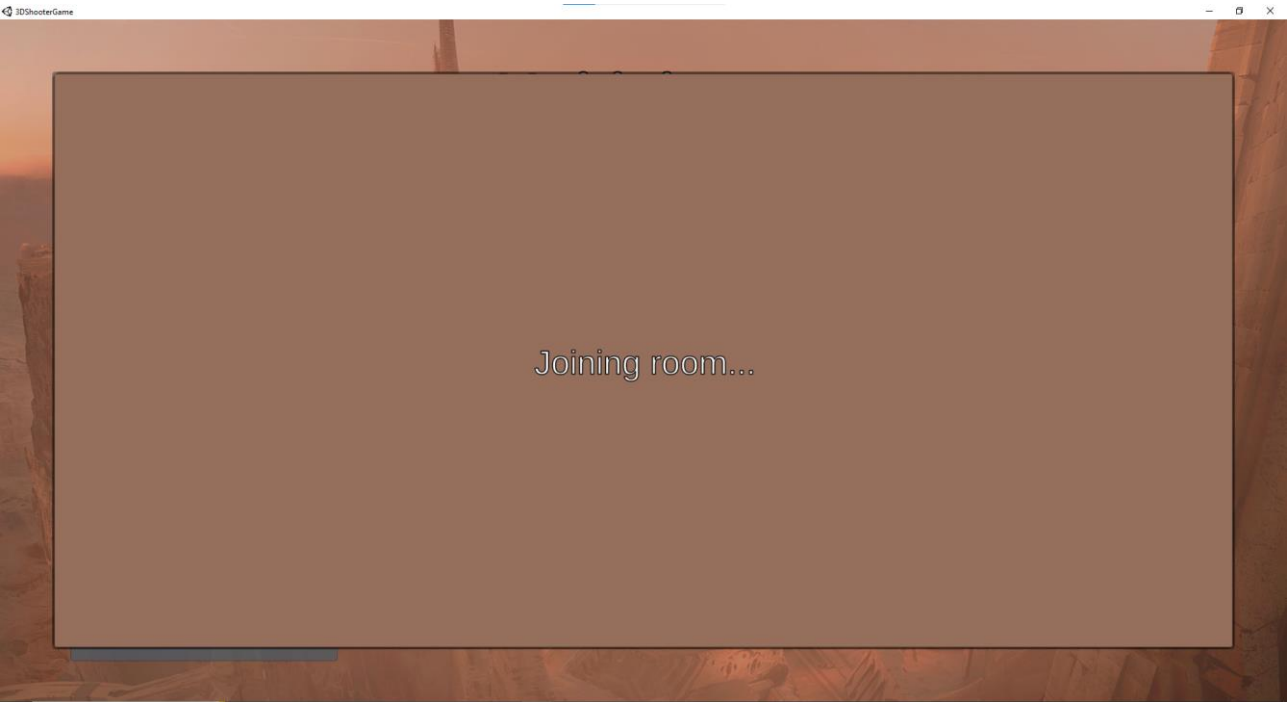


Рис. 3.43. Інформаційне вікно підключення до обраної кімнати

Над кожним гравцем у багатокористувацькій грі пишеться ім'я, це допомагає легко ідентифікувати кожного учасника гри.

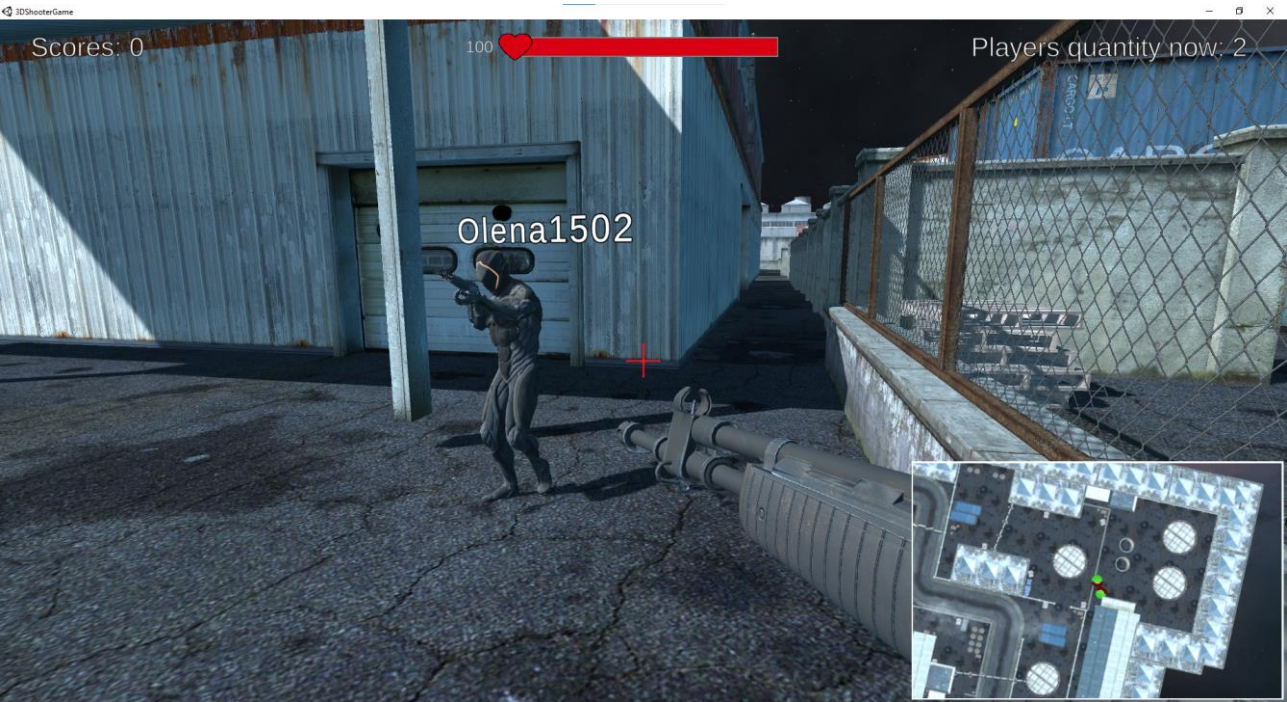


Рис. 3.44. Ігровий процес у «Multiplayer»: відображення гравця

При отриманні поранення, гра використовує візуальний ефект, забарвлюючи екран у червоний колір. Цей ефект створює реалістичне відчуття в багатокористувацькому геймплеї.

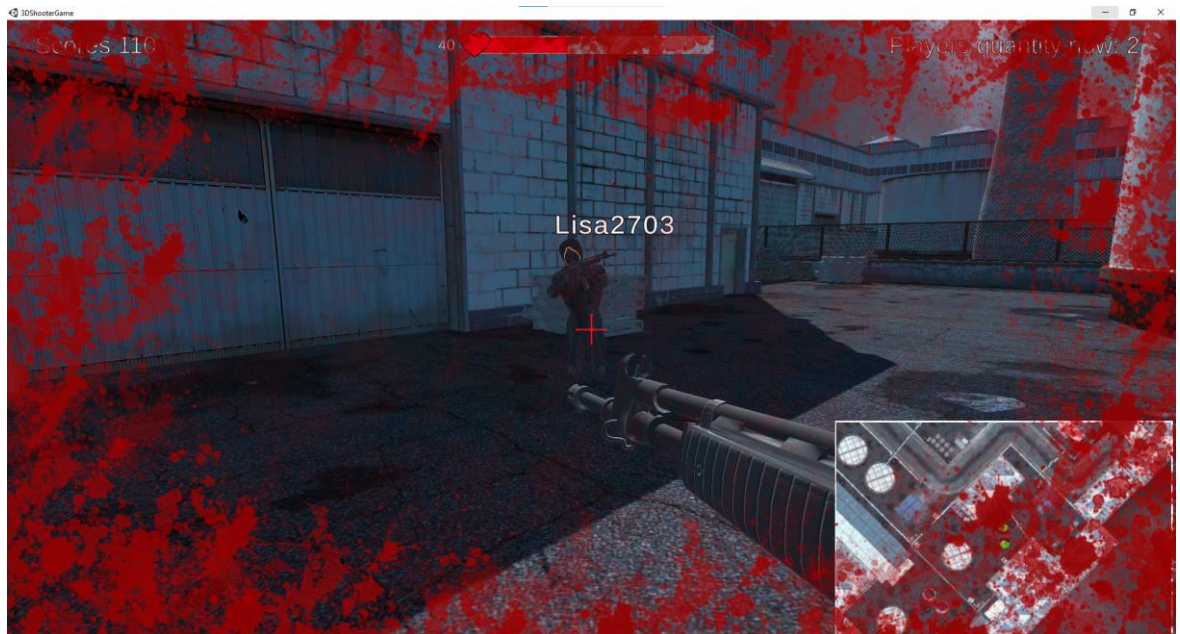


Рис. 3.45. Ігровий процес у «Multiplayer»: отриманні поранення

Якщо гравця вбивають, то на екрані з'являється інформація хто саме вбив користувача. Після цього відбувається процес збереження результату та через декілька секунд гравець знову відроджується у багатокористувацькій грі у випадково згенерованому місці на карті.

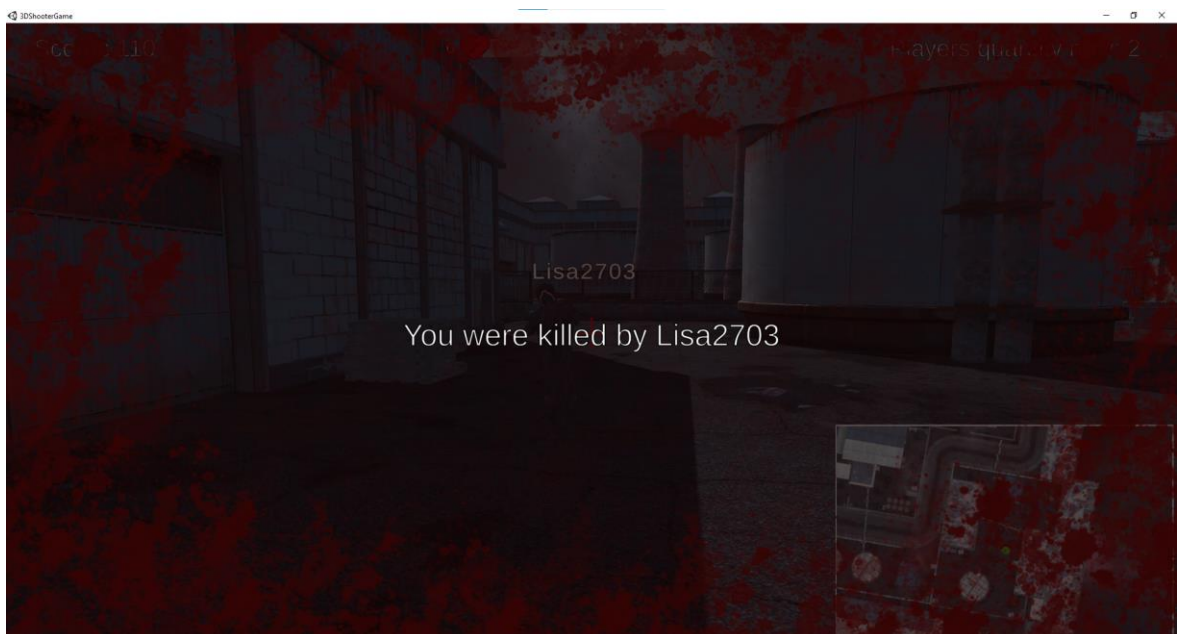


Рис. 3.46. Ігровий процес у «Multiplayer»: відображення інформації після програшу

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		67

Для того, щоб вийти із багатокористувацької гри необхідно натиснути клавішу «Esc» на клавіатурі. Це призведе до збереження поточних результатів гри та відкриття головного меню гри.

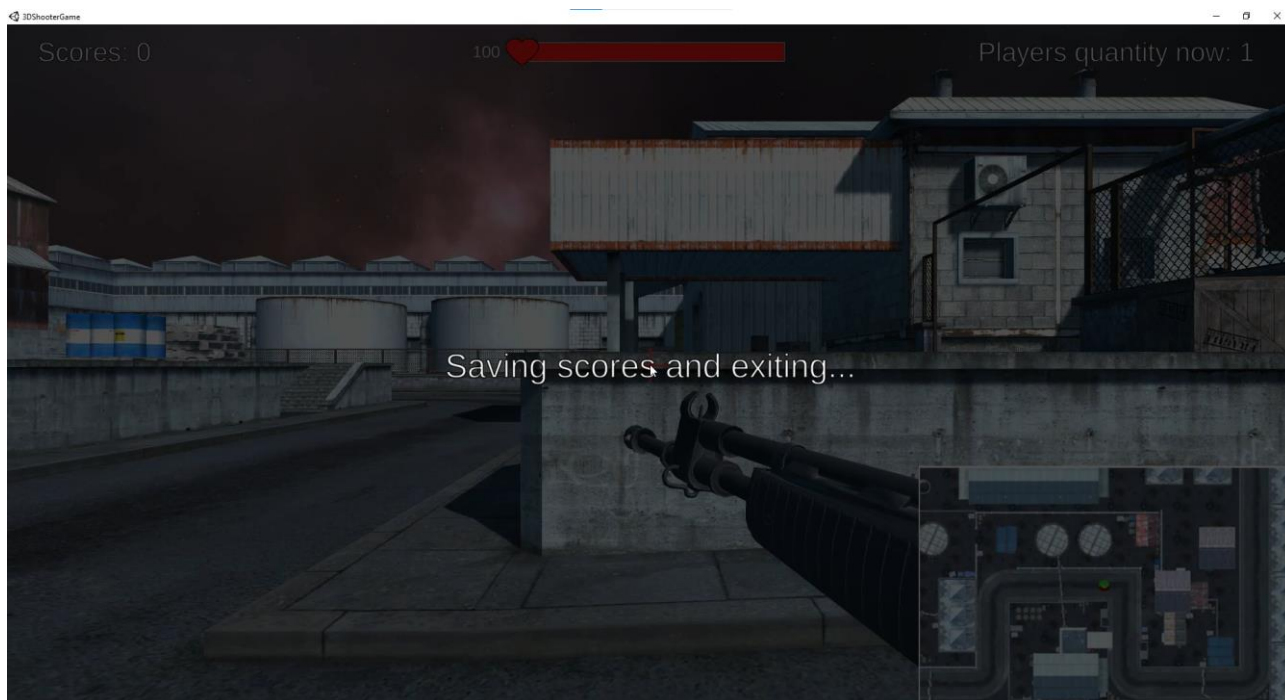


Рис. 3.47. Ігровий процес у «Multiplayer»: збереження результату та вихід із багатокористувацької гри

3.2. Тестування роботи програмного комплексу

Функціональне тестування спрямоване на перевірку функцій, можливостей та взаємодії програмної системи або програми з метою забезпечення відповідності визначеним функціональним вимогам і відповідності передбачуваним потребам бізнесу.

Основна мета функціонального тестування полягає у перевірці, як програмна система взаємодіє з різними компонентами, а також у перевірці її функцій. Цей процес включає тестування вхідних і вихідних даних програмного забезпечення, маніпулювання даними, взаємодію з користувачем і реакцію системи на різні сценарії та умови. Функціональне тестування обмежується перевіркою коректності роботи системи і відсутністю помилок в її функціонуванні[43].

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Спочатку необхідно протестувати головне меню комп'ютерної гри, а саме: працездатність усіх кнопок, слайдерів та полів для введення даних.

Таблиця 3.1

Тестування головного меню комп'ютерної гри

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Перевірка кнопки «Play by plot» головного меню	Натискання кнопки «Play by plot»	Відкриття вікна вибору сюжетних рівнів	Відкриття вікна вибору сюжетних рівнів	Пройдено
2	Перевірка кнопки «Play multiplayer» головного меню	Натискання кнопки «Play multiplayer»	Відкриття вікна налаштування багатокористувацької гри	Відкриття вікна налаштування багатокористувацької гри	Пройдено
3	Перевірка кнопки «About Game» головного меню	Натискання кнопки «About Game»	Відкриття вікна з інформацією про гру	Відкриття вікна з інформацією про гру	Пройдено
4	Перевірка кнопки «Scores» головного меню	Натискання кнопки «Scores»	Відкриття вікна з відображенням всіх найвищих набраних балів у всіх рівнях та у багатокористувацькій грі	Відкриття вікна з відображенням всіх найвищих набраних балів у всіх рівнях та у багатокористувацькій грі	Пройдено
5	Перевірка кнопки «Settings» головного меню	Натискання кнопки «Settings»	Відкриття вікна з налаштуваннями гри	Відкриття вікна з налаштуваннями гри	Пройдено

Продовження таблиці 3.1

6	Перевірка зміни гучності фонової музики у меню «Settings»	Зміна слайдера гучності фонової музики	Зміна гучності фонової музики	Зміна гучності фонової музики	Пройдено
7	Перевірка зміни гучності звуку пострілу у меню «Settings»	Зміна слайдера гучності звуку пострілу	Зміна гучності звуку пострілу	Зміна гучності звуку пострілу	Пройдено
8	Перевірка зміни якості графіки у меню «Settings»	Вибрати із випадального списку якості графіки	Зміна якості графіки у відповідності із вибраною якістю	Зміна якості графіки у відповідності із вибраною якістю	Пройдено
9	Перевірка кнопки видалення всіх даних користувача у меню «Settings»	Натискання кнопки «Delete»	Локально збережені бали пройдених рівнів та багатокористувацької гри видаляються	Локально збережені бали пройдених рівнів та багатокористувацької гри видаляються	Пройдено
10	Перевірка кнопки повного виходу з гри «Exit» головного меню	Натискання кнопки «Exit»	Вихід з гри	Вихід з гри	Пройдено
11	Перевірка кнопки «Level 1: New York» у меню «Play by plot»	Натискання кнопки «Level 1: New York»	Запуск першого рівня гри	Запуск першого рівня гри	Пройдено

Продовження таблиці 3.1

12	Перевірка кнопки «Level 2: New York» у меню «Play by plot»	Натискання кнопки	Запуск другого рівня гри	Запуск другого рівня гри	Пройдено
13	Перевірка кнопки «Level 3: Mars» у меню «Play by plot»	Натискання кнопки	Запуск третього рівня гри	Запуск третього рівня гри	Пройдено
14	Перевірка кнопки «Level 4: Venus» у меню «Play by plot»	Натискання кнопки	Запуск четвертого рівня гри	Запуск четвертого рівня гри	Пройдено
15	Перевірка кнопки «Level 5: Uranus» у меню «Play by plot»	Натискання кнопки	Запуск п'ятого рівня гри	Запуск п'ятого рівня гри	Пройдено
16	Перевірка кнопки «Go back» у меню «Play by plot»	Натискання кнопки «Go back»	Перехід до головного меню	Перехід до головного меню	Пройдено
17	Перевірка поля введення імені у меню «Multiplayer»	Введення імені у поле введення	Введений текст відображаєть ся у полі введення імені користувача	Введений текст відображаєть ся у полі введення імені користувача	Пройдено
18	Перевірка кнопки «Create name» у меню «Multiplayer»	Натискання кнопки «Create name»	Введене ім'я зберігається локально на комп'ютері	Введене ім'я зберігається локально на комп'ютері	Пройдено
19	Перевірка поля введення назви кімнати у меню «Multiplayer»	Введення назви кімнати у поле введення	Введений текст відображаєть ся у полі введення назви кімнати	Введений текст відображаєть ся у полі введення назви кімнати	Пройдено

Продовження таблиці 3.1

20	Перевірка кнопки «Create room» у меню «Multiplayer»	Натискання кнопки «Create room»	Створюється кімната із введеною у поле введення назвою кімнати	Створюється кімната із введеною у поле введення назвою кімнати	Пройдено
21	Перевірка кнопки «Start game» у меню «Multiplayer»	Натискання кнопки «Start game»	Запускається багатокористувацька гра у відповідній кімнаті	Запускається багатокористувацька гра у відповідній кімнаті	Пройдено
22	Перевірка кнопки «Find room» у меню «Multiplayer»	Натискання кнопки «Find room»	Виводиться список всіх наявних кімнат, до яких можна підключитись	Виводиться список всіх наявних кімнат, до яких можна підключитись	Пройдено
23	Перевірка кнопки «Leave room» у меню «Multiplayer»	Натискання кнопки «Leave room»	Гравець покидає кімнату та повертається у меню «Multiplayer»	Гравець покидає кімнату та повертається у меню «Multiplayer»	Пройдено
24	Перевірка кнопки «Go back» у меню «Multiplayer»	Натискання кнопки «Go back»	Перехід до головного меню гри	Перехід до головного меню гри	Пройдено

Далі необхідно провести тестування ігрового процесу в режимі «Play by plot», зосереджуючись на наступних аспектах: перевірка взаємодії з оточенням та ворогами, виведення достовірної інформації про стан здоров'я, набрані бали, кількість ворогів та номер раунду гри.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестування ігрового процесу в режимі «Play by plot»

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Перевірка пострілу влучання у ворога	Натиснення лівої клавіші миші та виставлення прицілу в ворога	Нанесення удару та зменшення здоров'я у ворога	Нанесення удару та зменшення здоров'я у ворога	Пройдено
2	Перевірка паузи	Натиснення клавіші «Esc» на клавіатурі	Відкриття вікна паузи	Відкриття вікна паузи	Пройдено
3	Перевірка зменшення здоров'я у гравця	Отримання пострілу від ворога	Відображення рівня здоров'я на слайдері зменшується	Відображення рівня здоров'я на слайдері зменшується	Пройдено
4	Перевірка ліквідації ворога	Нанесення декількох пострілів ворогу	Ворог зникає із ігрового процесу	Ворог зникає із ігрового процесу	Пройдено
5	Перевірка інтерактивності відображення мапи	Виконати будь-який рух головним персонажем	Позначки руху головного персонажу та ворогів рухаються	Позначки руху головного персонажу та ворогів рухаються	Пройдено
6	Перевірка відображення набраних балів	Ліквідація будь-якого ворога	Збільшення кількості набраних балів	Збільшення кількості набраних балів	Пройдено
7	Перевірка відображення кількості ворогів	Ліквідація будь-якого ворога	Зменшення кількості ворогів	Зменшення кількості ворогів	Пройдено

Продовження таблиці 3.2

8	Перевірка правильності відображення раундів	Ліквідація всіх ворогів поточного раунду	Збільшення поточного раунду на одиницю	Збільшення поточного раунду на одиницю	Пройдено
9	Перевірка перезапуску рівня	Натиснути кнопку «Play again»	Поточний рівень перезапускається з початку	Поточний рівень перезапускається з початку	Пройдено
10	Перевірка переходу на наступний рівень	Натиснути кнопку «Next level»	Запускається наступний рівень гри	Запускається наступний рівень гри	Пройдено

Заключним етапом тестування 3D гри жанру шутер є тестування ігрового процесу в режимі «Multiplayer». Необхідно зосередитись на таких аспектах як: перевірка взаємодії з оточенням та ворогами, виведення достовірної інформації про стан здоров'я, набрані бали та кількість ворогів у відповідній кімнаті.

Таблиця 3.3

Тестування ігрового процесу в режимі «Multiplayer»

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Перевірка пострілу влучання у ворога	Натиснення лівої клавіші миші та виставлення прицілу в ворога	Нанесення удару та зменшення здоров'я у ворога	Нанесення удару та зменшення здоров'я у ворога	Пройдено
2	Перевірка зменшення здоров'я у гравця	Отримання пострілу від ворога	Відображення рівня здоров'я на слайдері зменшується	Відображення рівня здоров'я на слайдері зменшується	Пройдено

Продовження таблиці 3.3

3	Перевірка ліквідації ворога	Нанесення декількох пострілів ворогу	Ворог зникає із ігрового процесу	Ворог зникає із ігрового процесу	Пройдено
4	Перевірка інтерактивності відображення мапи	Виконати будь-який рух головним персонажем	Позначки руху головного персонажу та ворогів рухаються	Позначки руху головного персонажу та ворогів рухаються	Пройдено
5	Перевірка відображення набраних балів	Ліквідація будь-якого ворога	Збільшення кількості набраних балів	Збільшення кількості набраних балів	Пройдено
6	Перевірка відображення кількості ворогів	Ліквідація будь-якого ворога	Зменшення кількості ворогів	Зменшення кількості ворогів	Пройдено
7	Перевірка відродження гравця після програшу	Отримати поразку та почекати декілька секунд	Гравець знову з'явився у багатокористувацькій грі	Гравець знову з'явився у багатокористувацькій грі	Пройдено
8	Перевірка виходу із гри	Натиснути клавішу «Esc» на клавіатурі	Набрані бали зберігаються, відкривається головне меню гри	Набрані бали зберігаються, відкривається головне меню гри	Пройдено

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було проведено аналіз та опис інтерфейсу розробленого програмного додатку, були визначені основні елементи інтерфейсу та їх функціональні можливості. Також був наданий детальний опис порядку роботи з програмним додатком, починаючи від його запуску та закінчуючи проходженням всіх рівнів та проходження багатокористувацької гри.

Також було проведено процес тестування розробленого програмного комплексу з метою перевірки його працездатності та відповідності вимогам, а саме було протестовано роботу головного меню комп'ютерної гри, ігрові процеси в режимах «Play by plot» та «Multiplayer». Під час тестування було перевірено весь функціонал комп'ютерної 3D гри.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено аналіз напрямку, обрано інструментальні засоби, спроектовано та реалізовано 3D гру жанру шутер засобами Unity під Windows-платформу, та вирішено наступні завдання:

- проведений аналіз предметної області дав можливість визначити ключові аспекти комп'ютерної гри;
- для аналізу структури комп'ютерної гри було вивчено функціональні та нефункціональні вимоги, а також була побудована діаграма варіантів використання, було розроблено діаграму класів;
- були проаналізовані алгоритми основних процесів гри, а також взаємодія класів системи під час ігрового процесу;
- було детально розібрано головні аспекти реалізації ігрового процесу 3D шутера;
- був реалізований користувацький інтерфейс комп'ютерної гри;
- було проведено функціональне тестування комп'ютерної гри та було перевірено, що 3D шутер дійсно працює стабільно та виконує поставлені задачі.

Розроблена комп'ютерна гра є готовою до ігрового процесу та придатна для використання гравцями різного рівня складності. У майбутньому даний 3D шутер може бути швидко доповнений новими функціональними можливостями та розширений графічно завдяки правильно обраному інструментарію.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шутер від першої особи [Електронний ресурс] – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Шутер_від_першої_особи
2. Battlefield [Електронний ресурс] – 2023. – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Battlefield_\(video_game_series\)](https://en.wikipedia.org/wiki/Battlefield_(video_game_series))
3. Battlefield 2042 [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Battlefield_2042
4. Battlefield 2042 Accessibility Resources for PC [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://www.ea.com/able/resources/battlefield-2042/pc/mouse-keyboard-edit-controls>
5. BioShock [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/BioShock>
6. Game engine [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Game_engine
7. List of game engines [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/List_of_game_engines
8. Use case diagram [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Use_case_diagram
9. Class diagram [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Class_diagram
10. Photon Unity Networking [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://doc.photonengine.com/pun/current/getting-started/pun-intro>
11. Unity Asset Store [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://assetstore.unity.com>
12. RPG/FPS Game Assets for PC/Mobile (Industrial Set v2.0), Unity Asset Store [Електронний ресурс] – 2021. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/environments/industrial/rpg-fps-game-assets-for-pc-mobile-industrial-set-v2-0-86679>

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Real New York City Vol. 1, Unity Asset Store [Електронний ресурс] – 2022. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/environments/urban/real-new-york-city-vol-1-208247>
14. Real New York City Vol. 2, Unity Asset Store [Електронний ресурс] – 2022. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/environments/urban/real-new-york-city-vol-2-222827>
15. Rugged Mountains – Terrains, Stamps and Brushes, Unity Asset Store [Електронний ресурс] – 2022. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/environments/landscapes/rugged-mountains-terrains-stamps-and-brushes-220806>
16. Mixamo [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://www.mixamo.com>
17. Low Poly 3D Flying Saucer Model, Unity Asset Store [Електронний ресурс] – 2016. – Режим доступу до ресурсу: <https://opengameart.org/content/low-poly-3d-flying-saucer-model>
18. Italian Machine Gun, Unity Asset Store [Електронний ресурс] – 2018. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/props/guns/italian-machine-gun-44430>
19. FPS Gun 4K – Assault Rifle 1, Unity Asset Store [Електронний ресурс] – 2022. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/props/guns/fps-gun-4k-assault-rifle-1-223855>
20. Sci-Fi Gun Heavy, Unity Asset Store [Електронний ресурс] – 2017. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/props/guns/sci-fi-gun-heavy-87878>
21. Shotgun, Unity Asset Store [Електронний ресурс] – 2015. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/props/guns/shotgun-26685>

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

22. Compact Bullpup Assault Rifle, Unity Asset Store [Електронний ресурс] – 2018. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/props/guns/compact-bullpup-assault-rifle-111003>
23. Submachine Gun Reaper, Unity Asset Store [Електронний ресурс] – 2022. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/3d/props/guns/submachine-gun-reaper-7890>
24. WorldSkies Free Skybox Pack, Unity Asset Store [Електронний ресурс] – 2017. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/2d/textures-materials/sky/worldskies-free-skybox-pack-86517>
25. Milky Way Skybox, Unity Asset Store [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/2d/textures-materials/milky-way-skybox-94001>
26. Starfield Skybox, Unity Asset Store [Електронний ресурс] – 2017. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/2d/textures-materials/sky/starfield-skybox-92717>
27. Classic Space Skybox, Unity Asset Store [Електронний ресурс] – 2021. – Режим доступу до ресурсу: <https://assetstore.unity.com/packages/2d/textures-materials/classic-space-skybox-11596>
28. Pixabay [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://pixabay.com/sound-effects/>
29. Unity Game Developer Masterclass 2021 (2D, 3D and FPS) [Електронний ресурс] – 2021. – Режим доступу до ресурсу: <https://ua.udemy.com/course/unity-game-developer-masterclass/learn/lecture/26677302#overview>
30. Learn To Create An Online Multiplayer Game In Unity [Електронний ресурс] – 2021. – Режим доступу до ресурсу: <https://ua.udemy.com/course/unity-online-multiplayer/learn/lecture/25944118#overview>
31. Unity in Action / Joe Hocking. – Manning, 2022. – 416 p.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

32. Unity Game Development Cookbook: essentials for every game / Paris Buttfeld-Addison, Jonathon Manning, Tim Nugent. – O`Reilly Media, 2019. – 408 p.
33. Learning C# by Developing Games With Unity / Harrison Ferrone. – Packt Publishing, 2022. – 458 p.
34. Unity documentation [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/index.html>
35. C# documentation [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/csharp/>
36. Settings Menu in Unity [Електронний ресурс] – 2017. – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=YOaYQrN1oYQ>
37. 5 Minute Dialogue System in Unity Tutorial [Електронний ресурс] – 2021. – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=8oTYabhj248>
38. Easy Enemy Health Bars in Unity [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://www.youtube.com/watch?v=_lREXfAMUcE
39. How to make a Health Bar in Unity [Електронний ресурс] – 2020. – Режим доступу до ресурсу: https://www.youtube.com/watch?v=BLfNP4Sc_iA
40. How to Play or Stop Music Through Scenes in Unity [Електронний ресурс] – 2022. – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=ha6U8jHl9ak>
41. Weapon Switching – Unity Tutorial [Електронний ресурс] – 2017. – Режим доступу до ресурсу: https://www.youtube.com/watch?v=Dn_BUIVdAPg
42. How to make a Minimap in Unity [Електронний ресурс] – 2017. – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=28JTTXqMvOU>
43. Functional Testing: A Detailed Guide [Електронний ресурс] – 2023. – Режим доступу до ресурсу: <https://www.browserstack.com/guide/functional-testing>

ДОДАТКИ

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл MainMenuManager.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class MainMenuManager : MonoBehaviour {
    public GameObject allLevels;
    public GameObject aboutScreen;
    public GameObject scoresScreen;
    public GameObject levelsScreen;
    public GameObject enemiesScreen;
    public GameObject weaponScreen;
    public GameObject locationsScreen;
    public GameObject settingsScreen;
    public GameObject multiplayerScreen;
    public TextMeshProUGUI highestScoreMultiplayerText;
    public TextMeshProUGUI highestScoreLevel1Text;
    public TextMeshProUGUI highestScoreLevel2Text;
    public TextMeshProUGUI highestScoreLevel3Text;
    public TextMeshProUGUI highestScoreLevel4Text;
    public TextMeshProUGUI highestScoreLevel5Text;
    public Button level2Button;
    public Button level3Button;
    public Button level4Button;
    public Button level5Button;
    void Start() {
        Cursor.lockState = CursorLockMode.None;
        int scores = PlayerPrefs.GetInt("LevelScoresScene3");
        highestScoreLevel1Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("LevelScoresScene5");
        highestScoreLevel2Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("LevelScoresScene7");
        highestScoreLevel3Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("LevelScoresScene9");
        highestScoreLevel4Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("LevelScoresScene11");
        highestScoreLevel5Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("MultiplayerScores");
        highestScoreMultiplayerText.text = scores.ToString();
        if (PlayerPrefs.GetInt("LevelCompletedScene3") != 1) {
            level2Button.interactable = false;
        }
        if (PlayerPrefs.GetInt("LevelCompletedScene5") != 1) {
            level3Button.interactable = false;
        }
        if (PlayerPrefs.GetInt("LevelCompletedScene7") != 1) {
            level4Button.interactable = false;
        }
        if (PlayerPrefs.GetInt("LevelCompletedScene9") != 1) {
            level5Button.interactable = false;
        }
    }
    private void Update() {
        int scores = PlayerPrefs.GetInt("LevelScoresScene3");
        highestScoreLevel1Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("LevelScoresScene5");
        highestScoreLevel2Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("LevelScoresScene7");
        highestScoreLevel3Text.text = scores.ToString();
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        scores = PlayerPrefs.GetInt("LevelScoresScene9");
        highestScoreLevel4Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("LevelScoresScene11");
        highestScoreLevel5Text.text = scores.ToString();
        scores = PlayerPrefs.GetInt("MultiplayerScores");
        highestScoreMultiplayerText.text = scores.ToString();
        if (PlayerPrefs.GetInt("LevelCompletedScene3") != 1) {
            level2Button.interactable = false;
        }
        if (PlayerPrefs.GetInt("LevelCompletedScene5") != 1) {
            level3Button.interactable = false;
        }
        if (PlayerPrefs.GetInt("LevelCompletedScene7") != 1) {
            level4Button.interactable = false;
        }
        if (PlayerPrefs.GetInt("LevelCompletedScene9") != 1) {
            level5Button.interactable = false;
        }
    }}

    public void StartStoryLevel1() {
        SceneManager.LoadScene(2);
    }
    public void StartLevel1Game() {
        SceneManager.LoadScene(3);
    }
    public void StartStoryLevel2() {
        SceneManager.LoadScene(4);
    }
    public void StartLevel2Game() {
        SceneManager.LoadScene(5);
    }
    public void StartStoryLevel3() {
        SceneManager.LoadScene(6);
    }
    public void StartLevel3Game() {
        SceneManager.LoadScene(7);
    }
    public void StartStoryLevel4() {
        SceneManager.LoadScene(8);
    }
    public void StartLevel4Game() {
        SceneManager.LoadScene(9);
    }
    public void StartStoryLevel5() {
        SceneManager.LoadScene(10);
    }
    public void StartLevel5Game() {
        SceneManager.LoadScene(11);
    }
    public void StartFinalStoryTelling() {
        SceneManager.LoadScene(12);
    }
    public void ExitGame() {
        Application.Quit();
    }
    public void OpenAllLevels() {
        allLevels.SetActive(true);
    }
    public void CloseAllLevels() {
        allLevels.SetActive(false);
    }
    public void OpenAbout() {
        aboutScreen.SetActive(true);
    }
    public void CloseAbout() {
        aboutScreen.SetActive(false);
    }
}

```

```

public void OpenScores() {
    scoresScreen.SetActive(true);
}
public void CloseScores() {
    scoresScreen.SetActive(false);
}
public void OpenSettings() {
    settingsScreen.SetActive(true);
}
public void CloseSettings() {
    settingsScreen.SetActive(false);
}
public void OpenLevels() {
    levelsScreen.SetActive(true);
}
public void CloseLevels() {
    levelsScreen.SetActive(false);
}
public void OpenEnemiesScreen() {
    enemiesScreen.SetActive(true);
}
public void CloseEnemiesScreen() {
    enemiesScreen.SetActive(false);
}
public void OpenWeaponScreen() {
    weaponScreen.SetActive(true);
}
public void CloseWeaponScreen() {
    weaponScreen.SetActive(false);
}
public void OpenLocationsScreen() {
    locationsScreen.SetActive(true);
}
public void CloseLocationsScreen() {
    locationsScreen.SetActive(false);
}
public void StartMultiplayerGame() {
    SceneManager.LoadScene(1);
}
public void OpenMultiplayerScreen() {
    multiplayerScreen.SetActive(true);
}
public void CloseMultiplayerScreen() {
    multiplayerScreen.SetActive(false);
}
}
}

```

Файл SettingsMenu.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;
using UnityEngine.UI;
public class SettingsMenu : MonoBehaviour
{
    public AudioManager audioMixer;
    public Slider backgroundVolumeSlider;
    public Slider gunVolumeSlider;
    public Dropdown qualityDropdown;
    public void SetVolume (float volume)
    {
        audioMixer.SetFloat("backgroundVolume", volume);
    }
    private void Start()
    {

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        StartSetVolume();
        StartSetGunVolume();
        StartSetQuality();
    }
    public void StartSetVolume()
    {
        float volume;
        audioMixer.GetFloat("backgroundVolume", out volume);
        backgroundVolumeSlider.value = volume;
    }
    public void SetGunVolume(float volume)
    {
        audioMixer.SetFloat("gunVolume", volume);
    }
    public void StartSetGunVolume()
    {
        float volume;
        audioMixer.GetFloat("gunVolume", out volume);
        gunVolumeSlider.value = volume;
    }
    public void SetQuality(int qualityIndex)
    {
        QualitySettings.SetQualityLevel(qualityIndex);
    }
    public void StartSetQuality()
    {
        int currentQualityLevel = QualitySettings.GetQualityLevel();
        qualityDropdown.value = currentQualityLevel;
    }
    public void DeleteAllStoredData()
    {
        PlayerPrefs.DeleteAll();
        PlayerPrefs.Save();
    }
}

```

Файл BackgroundMusic.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class BackgroundMusic : MonoBehaviour
{
    public static BackgroundMusic instance;
    void Awake()
    {
        if (instance != null)
            Destroy(gameObject);
        else
        {
            instance = this;
            DontDestroyOnLoad(this.gameObject);
        }
    }
}

```

Файл Dialogue.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.SceneManagement;
public class Dialogue : MonoBehaviour
{
    public TextMeshProUGUI textComponent;
    public string[] lines;
    public float textSpeed;
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public int sceneNumber;
private int index;
void Start()
{
    textComponent.text = string.Empty;
    StartDialogue();
}
void Update()
{
    if(Input.GetMouseButtonDown(0))
    {
        if(textComponent.text == lines[index])
        {
            NextLine();
        }
        else
        {
            StopAllCoroutines();
            textComponent.text = lines[index];
        }
    }
}
void StartDialogue()
{
    index = 0;
    StartCoroutine(TypeLine());
}
IEnumerator TypeLine()
{
    foreach (char c in lines[index].ToCharArray())
    {
        textComponent.text += c;
        yield return new WaitForSeconds(textSpeed);
    }
}
void NextLine()
{
    if(index < lines.Length - 1)
    {
        index++;
        textComponent.text = string.Empty;
        StartCoroutine(TypeLine());
    }
    else
    {
        gameObject.SetActive(false);
        SceneManager.LoadScene(sceneNumber);
    }
}
}
}

```

Файл MouseLook.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MouseLook : MonoBehaviour
{
    public float mouseSensitivity = 100f;
    public Transform playerBody;
    float xRotation = 0f;
    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
    }
    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;
        xRotation += mouseY;
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);
        transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
        playerBody.Rotate(Vector3.up * mouseX);
    }
}

```

Файл PlayerMovement.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class PlayerMovement : MonoBehaviour
{
    public CharacterController controller;
    public float speed = 12f;
    Vector3 velocity;
    public float gravity = -9.81f;
    public bool isGrounded;
    public Transform groundCheck;
    public float groundDistance = 0.4f;
    public LayerMask groundMask;
    public float jumpHeight = 2f;
    void Update()
    {
        isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance,
        groundMask);
        if(isGrounded && velocity.y < 0)
        {
            velocity.y = -2f;
        }
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");
        Vector3 move = transform.right * (-x) + transform.forward*(-z);
        controller.Move(move * speed * Time.deltaTime);
        velocity.y += gravity * Time.deltaTime;
        controller.Move(velocity * Time.deltaTime);
        if (Input.GetButtonDown("Jump") && isGrounded)
        {
            velocity.y = Mathf.Sqrt(jumpHeight * -2 * gravity);
        }
    }
}

```

Файл PlayerManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using TMPPro;
public class PlayerManagerByPlot : MonoBehaviour
{
    public float health = 100;
    public GameObject gameOverScreen;
    public GameObject injuredScreen;
    public TextMeshProUGUI playerHealthText;
    public PlayerHealthBar playerHealthBar;
    public GameObject gameManagerObject;
    public void Hit(int damage)
    {
        health -= damage;
        playerHealthText.text = health.ToString();
        playerHealthBar.SetHealth(health);
        if (health <= 0)
        {
            injuredScreen.SetActive(false);
            gameOverScreen.SetActive(true);
            gameManagerObject.GetComponent<GameManagerScript>().SaveLevelScores();
            Cursor.lockState = CursorLockMode.None;
            Time.timeScale = 0f;
        }
        else
        {
            InjuredScreen();
        }
    }
    public void RefillHealth()
    {
        health = 100;
    }
    public void InjuredScreen()
    {
        injuredScreen.SetActive(true);
        Invoke("RemoveInjuredScreen", 1f);
    }
    public void RemoveInjuredScreen()
    {
        injuredScreen.SetActive(false);
    }
}
```

Файл PlayerHealthBar.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class PlayerHealthBar : MonoBehaviour
{
    public Slider slider;
    public void SetMaxHealth()
    {
        slider.maxValue = 100;
        slider.value = 100;
    }
    public void SetHealth(float health)
    {

```

```

        slider.value = health;
    }
}

```

Файл GameManager.cs

```

using System.Collections;
using System.Collections.Generic;
using System.Security.Cryptography;
using TMPPro;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.SceneManagement;
public class GameManagerScript : MonoBehaviour
{
    public TextMeshProUGUI roundNumberText;
    public TextMeshProUGUI enemiesAliveText;
    public TextMeshProUGUI scoresText;
    public EnemyFactory enemyFactory;
    public GameObject pauseScreen;
    public GameObject missionCompletedScreen;
    public GameObject gameOverScreen;
    public TextMeshProUGUI gameOverScoresText;
    public TextMeshProUGUI gameOverRoundsText;
    public TextMeshProUGUI gameOverEnemiesAliveText;
    public TextMeshProUGUI missionCompletedScoresText;
    public TextMeshProUGUI missionCompletedRoundText;
    public GameObject bulletImpact;
    public int totalScores = 0;
    public int maxRounds;
    public GameObject player;
    public AudioSource shootingSound;
    private int quantityEnemiesSimplePerRound = 10;
    private int quantityEnemiesBossPerRound = 4;
    private int quantityEnemiesWithGunPerRound = 3;
    public int enemiesAlive = 0;
    public int round = 0;
    public void AddScores(int scoresNumber)
    {
        totalScores += scoresNumber;
    }
    private void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        Cursor.lockState = CursorLockMode.Locked;
        Time.timeScale = 1f;
    }
    public void NextWave(int round, int quantityEnemiesSimplePerRound, int
quantityEnemiesBossPerRound, int quantityEnemiesWithGunPerRound)
    {
        for (int i = 0; i < quantityEnemiesSimplePerRound + round; i++)
        {
            enemyFactory.CreateEnemy(EnemyType.Simple);
            enemiesAlive++;
        }
        for (int i = 0; i < quantityEnemiesBossPerRound + round; i++)
        {
            enemyFactory.CreateEnemy(EnemyType.Boss);
            enemiesAlive++;
        }
        for (int i = 0; i < quantityEnemiesWithGunPerRound + round; i++)
        {
            enemyFactory.CreateEnemy(EnemyType.WithGun);
            enemiesAlive++;
        }
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (round % 2 != 0)
        {
            player.GetComponent<PlayerManagerByPlot>().RefillHealth();
        }
    }
    private void Update()
    {
        scoresText.text = "Scores: " + totalScores.ToString();
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (pauseScreen.activeSelf)
            {
                NotPausedGame();
            }
            else
            {
                PausedGame();
            }
        }
        if(enemiesAlive == 0 && round < maxRounds)
        {
            round++;
            roundNumberText.text = "Round " + round.ToString() + " out of " +
maxRounds.ToString();
            NextWave(round, quantityEnemiesSimplePerRound,
quantityEnemiesBossPerRound, quantityEnemiesWithGunPerRound);
        }
        else if(enemiesAlive == 0 && round >= maxRounds)
        {
            MissionCompleted();
        }
        enemiesAliveText.text = "Enemies alive: " + enemiesAlive.ToString();
        if (!gameOverScreen.activeSelf && !pauseScreen.activeSelf &&
!missionCompletedScreen.activeSelf && (Input.GetMouseButtonDown(0) ||
Input.GetKeyDown(KeyCode.F)))
        {
            shootingSound.Play();
            Camera cam = Camera.main;
            Ray ray = cam.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0f));
            ray.origin = cam.transform.position;
            if (Physics.Raycast(ray, out RaycastHit hit))
            {
                Debug.Log("We hit " + hit.collider.gameObject.name);
                GameObject bulletImpactObject = Instantiate(bulletImpact, hit.point +
(hit.normal * 0.002f), Quaternion.LookRotation(hit.normal, Vector3.up));
                Destroy(bulletImpactObject, 1f);
            }
        }
        if (gameOverScreen.activeSelf)
        {
            gameOverScoresText.text = "Your scores: " + totalScores.ToString();
            gameOverEnemiesAliveText.text = "Enemies alive: " +
enemiesAlive.ToString();
            gameOverRoundsText.text = "Round: " + round.ToString();
        }
    }
    public void OneEnemyDied()
    {
        enemiesAlive--;
        enemiesAliveText.text = "Enemies alive: " + enemiesAlive.ToString();
    }
    public void MissionCompleted()
    {
        Cursor.lockState = CursorLockMode.None;
        missionCompletedScreen.SetActive(true);
        missionCompletedScoresText.text = "Your scores: " + totalScores.ToString();
    }

```

					KH.KP.Б – 122 – 23 – ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        missionCompletedRoundText.text = "Round: " + round.ToString();
        Time.timeScale = 0f;
        int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
        PlayerPrefs.SetInt("LevelCompletedScene"+ currentSceneIndex, 1);
    }
    public void PausedGame()
    {
        pauseScreen.SetActive(true);
        Cursor.lockState = CursorLockMode.None;
        Time.timeScale = 0f;
    }
    public void NotPausedGame()
    {
        pauseScreen.SetActive(false);
        Cursor.lockState = CursorLockMode.Locked;
        Time.timeScale = 1f;
    }
    public void StartAgain()
    {
        int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
        SaveLevelScores();
        Time.timeScale = 1f;
        SceneManager.LoadScene(currentSceneIndex);
    }
    public void PlayAgainLevel()
    {
        int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
        SaveLevelScores();
        Time.timeScale = 1f;
        SceneManager.LoadScene(currentSceneIndex);
    }
    public void PlayNextLevel()
    {
        int sceneIndex = SceneManager.GetActiveScene().buildIndex;
        sceneIndex += 1;
        if (sceneIndex <= 11)
        {
            Time.timeScale = 1f;
            SaveLevelScores();
            SceneManager.LoadScene(sceneIndex);
        }
        else
        {
            Time.timeScale = 1f;
            SaveLevelScores();
            SceneManager.LoadScene(12);
        }
    }
    public void GoToMenu()
    {
        Time.timeScale = 1f;
        SaveLevelScores();
        SceneManager.LoadScene(0);
    }
    public void SaveLevelScores()
    {
        int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
        int previousScores = PlayerPrefs.GetInt("LevelScoresScene"+
currentSceneIndex);
        if (totalScores > previousScores)
        {
            PlayerPrefs.SetInt("LevelScoresScene" + currentSceneIndex, totalScores);
        }
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл WeaponSwitching.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class WeaponSwitching : MonoBehaviour {
    public int selectedWeapon = 0;
    void Start() {
        SelectWeapon();
    }
    void Update() {
        int previousSelectedWeapon = selectedWeapon;
        if(Input.GetAxis("Mouse ScrollWheel") > 0f)
        {
            if(selectedWeapon >= transform.childCount-1)
            {
                selectedWeapon = 0;
            }
            else
            {
                selectedWeapon++;
            }
        }
        if (Input.GetAxis("Mouse ScrollWheel") < 0f)
        {
            if (selectedWeapon <= 0)
            {
                selectedWeapon = transform.childCount - 1;
            }
            else
            {
                selectedWeapon--;
            }
        }
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            selectedWeapon = 0;
        }
        if (Input.GetKeyDown(KeyCode.Alpha2) && transform.childCount >= 2)
        {
            selectedWeapon = 1;
        }
        if (Input.GetKeyDown(KeyCode.Alpha3) && transform.childCount >= 3)
        {
            selectedWeapon = 2;
        }
        if (previousSelectedWeapon != selectedWeapon)
        {
            SelectWeapon();
        }
    }
    void SelectWeapon() {
        int i = 0;
        foreach(Transform weapon in transform) {
            if(i == selectedWeapon) {
                weapon.gameObject.SetActive(true);
            }
            else {
                weapon.gameObject.SetActive(false);
            }
            i++;
        }
    }
}
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						93
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл FloatingEnemyHealthBar.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class FloatingEnemyHealthBar : MonoBehaviour {
    [SerializeField] private Slider slider;
    public void UpdateHealthBar(int currentHealth, int maxHealth) {
        slider.value = currentHealth;
    }
    public void StartHealthBar(int maxHealth) {
        slider.maxValue = maxHealth;
        slider.minValue = 0;
    }
}
```

Файл EnemyFactory.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class EnemyFactory : MonoBehaviour {
    public GameObject simpleEnemyPrefab;
    public GameObject bossEnemyPrefab;
    public GameObject withGunEnemyPrefab;
    public GameObject[] spawnPoints;
    public void CreateEnemy(EnemyType enemyType) {
        GameObject prefab = null;
        switch (enemyType) {
            case EnemyType.Simple:
                prefab = simpleEnemyPrefab;
                break;
            case EnemyType.Boss:
                prefab = bossEnemyPrefab;
                break;
            case EnemyType.WithGun:
                prefab = withGunEnemyPrefab;
                break;
            default:
                Debug.LogError("Unknown enemy type: " + enemyType);
                break;
        }
        if (prefab != null) {
            GameObject spawnPoint = spawnPoints[Random.Range(0, spawnPoints.Length)];
            Instantiate(prefab, spawnPoint.transform.position, Quaternion.identity);
        }
    }
}
```

Файл Enemy.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
using System;
using System.Reflection;
using Unity.VisualScripting;
public enum EnemyType
{
    Simple,
    Boss,
    WithGun
}
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public class Enemy : MonoBehaviour
{
    public int health;
    public int maxHealth;
    public int damage;
    public bool canShoot;
    public GameObject player;
    public Animator enemyAnimator;
    public GameObject weaponHolder;
    public AudioSource enemyShootingSound;
    private float shootTimer = 0f;
    public float shootInterval = 5f;
    public GameObject gameManagerObject;
    [SerializeField] FloatingEnemyHealthBar healthBar;
    public virtual void Attack()
    {
        Debug.Log("Enemy attacks with damage: " + damage);
    }
    public virtual void ShootInPlayer(int amount)
    {
        enemyShootingSound.Play();
        player.GetComponent<PlayerManagerByPlot>().Hit(amount);
    }
    public virtual void TakeDamage(int amount)
    {
        int activeWeaponIndex = GetActiveWeaponIndex();
        if(activeWeaponIndex == 0)
        {
            health -= amount;
        }
        else if(activeWeaponIndex == 1)
        {
            health -= amount+5;
        }
        else if (activeWeaponIndex == 2)
        {
            health -= amount + 10;
        }
        else if (activeWeaponIndex == 3)
        {
            health -= amount + 20;
        }
        else if (activeWeaponIndex == 4)
        {
            health -= amount + 30;
        }
        else if (activeWeaponIndex == 5)
        {
            health -= amount + 50;
        }
        else {
            health -= amount;
        }
        healthBar.UpdateHealthBar(health, maxHealth);
        if (health <= 0) {
            gameManagerObject.GetComponent<GameManagerScript>().AddScores(10);
            Die();
        }
    }
    protected virtual void Die() {
        gameManagerObject.GetComponent<GameManagerScript>().OneEnemyDied();
        Destroy(gameObject);
    }
    void Start() {
        enemyShootingSound = GetComponent<AudioSource>();
        healthBar = GetComponentInChildren<FloatingEnemyHealthBar>();
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

healthBar.StartHealthBar(maxHealth);
healthBar.UpdateHealthBar(health, maxHealth);

player = GameObject.FindGameObjectWithTag("Player");
gameManagerObject = GameObject.FindGameObjectWithTag("GameManager");
weaponHolder = GameObject.FindGameObjectWithTag("WeaponHolder");
}
void Update() {
    if (Input.GetMouseButtonDown(0) || Input.GetKeyDown(KeyCode.F)) {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit)) {
            if (hit.collider.gameObject == gameObject) {
                TakeDamage(damage);
            }
        }
    }
    float distance = Vector3.Distance(player.transform.position,
gameObject.transform.position);
    GetComponent<NavMeshAgent>().destination = player.transform.position;
    if (GetComponent<NavMeshAgent>().velocity.magnitude > 1) {
        enemyAnimator.SetBool("isRunning", true);
    }
    else {
        enemyAnimator.SetBool("isRunning", false);
    }
    if(canShoot == true) {
        shootTimer += Time.deltaTime;
        if (shootTimer >= shootInterval) {
            if (canShoot == true && distance <= 70) {
                ShootInPlayer(damage);
            }
            shootTimer = 0f;
        }
    }
}
private int GetActiveWeaponIndex() {
    Transform holderTransform = weaponHolder.transform;
    int activeIndex = -1;
    for (int i = 0; i < holderTransform.childCount; i++) {
        if (holderTransform.GetChild(i).gameObject.activeSelf) {
            activeIndex = i;
            break;
        }
    }
    return activeIndex;
}
private void OnCollisionEnter(Collision collision) {
    if (collision.gameObject == player) {
        player.GetComponent<PlayerManagerByPlot>().Hit(damage);
    }
}
}

```

Файл SimpleEnemy.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class SimpleEnemy : Enemy {
    public override void Attack() {
        base.Attack();
    }
    public override void TakeDamage(int amount) {
        base.TakeDamage(amount);
    }
    protected override void Die() {

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        base.Die();
    }
    public override void ShootInPlayer(int amount) {
        base.ShootInPlayer(amount);
    }
}

```

Файл BossEnemy.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class BossEnemy : Enemy {
    public override void Attack() {
        base.Attack();
    }
    public override void TakeDamage(int amount) {
        base.TakeDamage(amount);
    }
    protected override void Die() {
        base.Die();
    }
    public override void ShootInPlayer(int amount) {
        base.ShootInPlayer(amount);
    }
}

```

Файл EnemyWithGun.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class EnemyWithGun : Enemy {
    public override void Attack() {
        base.Attack();
    }
    public override void TakeDamage(int amount) {
        base.TakeDamage(amount);
    }
    protected override void Die() {
        base.Die();
    }
    public override void ShootInPlayer(int amount) {
        base.ShootInPlayer(amount);
    }
}

```

Файл Minimap.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Minimap : MonoBehaviour {
    public Transform player;
    void LateUpdate() {
        Vector3 newPosition = player.position;
        newPosition.y = transform.position.y;
        transform.position = newPosition;
        transform.rotation = Quaternion.Euler(90f, player.eulerAngles.y, 180f);
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						97
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл MLauncher.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;
using TMPPro;
public class MLauncher : MonoBehaviourPunCallbacks {
    public static MLauncher instance;
    private void Awake() {
        instance = this;
    }
    public GameObject loadingScreen;
    public TMP_Text loadingText;
    public GameObject createRoomScreen;
    public TMP_InputField roomNameInput;
    public GameObject roomScreen;
    public TMP_Text roomNameText;
    public TMP_Text playerNameLabel;
    private List<TMP_Text> allPlayerNames = new List<TMP_Text>();
    public GameObject errorScreen;
    public TMP_Text errorText;
    public GameObject findRoomScreen;
    public MRoomButton roomButton;
    private List<MRoomButton> allRoomButtons = new List<MRoomButton>();
    public GameObject nameInputScreen;
    public TMP_InputField nameInput;
    private bool hasSetPlayerName;
    public string levelToPlay;
    public GameObject startButton;
    void Start() {
        OpenLoadingScreen();
        loadingText.text = "Connecting to network...!";
        Time.timeScale = 1f;
        if (!PhotonNetwork.IsConnected) {
            PhotonNetwork.ConnectUsingSettings();
        }
        if (PhotonNetwork.IsConnected) {
            CloseLoadingScreen();
        }
    }
    public override void OnConnectedToMaster() {
        PhotonNetwork.JoinLobby();
        PhotonNetwork.AutomaticallySyncScene = true;
        loadingText.text = "Joining lobby...";
    }
    public void StartGame() {
        CloseLoadingScreen();
        PhotonNetwork.LoadLevel(levelToPlay);
    }
    public override void OnMasterClientSwitched(Player newMasterClient) {
        if (PhotonNetwork.IsMasterClient) {
            startButton.SetActive(true);
        }
        else {
            startButton.SetActive(false);
        }
    }
    public override void OnJoinedLobby() {
        CloseLoadingScreen();
        CloseCreateRoomScreen();
        PhotonNetwork.NickName = Random.Range(0, 1000).ToString();
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!hasSetPlayerName) {
            CloseFindRoomScreen();
            OpenNameInputScreen();
            if (PlayerPrefs.HasKey("playerName")) {
                nameInput.text = PlayerPrefs.GetString("playerName");
            }
        }
        else {
            PhotonNetwork.NickName = PlayerPrefs.GetString("playerName");
        }
    }

    public void SetPlayerName() {
        if (!string.IsNullOrEmpty(nameInput.text)) {
            PhotonNetwork.NickName = nameInput.text;
            PlayerPrefs.SetString("playerName", nameInput.text);
            hasSetPlayerName = true;
            CloseNameInputScreen();
        }
    }

    public void OpenLoadingScreen() {
        loadingScreen.SetActive(true);
    }

    public void CloseLoadingScreen() {
        loadingScreen.SetActive(false);
    }

    public void OpenCreateRoomScreen() {
        createRoomScreen.SetActive(true);
    }

    public void CloseCreateRoomScreen() {
        createRoomScreen.SetActive(false);
    }

    public void OpenRoomScreen() {
        roomScreen.SetActive(true);
    }

    public void CloseRoomScreen() {
        CloseLoadingScreen();
        roomScreen.SetActive(false);
    }

    public void OpenErrorScreen() {
        errorScreen.SetActive(true);
    }

    public void CloseErrorScreen() {
        CloseLoadingScreen();
        errorScreen.SetActive(false);
    }

    public void OpenFindRoomScreen() {
        findRoomScreen.SetActive(true);
    }

    public void CloseFindRoomScreen() {
        findRoomScreen.SetActive(false);
    }

    public void OpenNameInputScreen() {
        nameInputScreen.SetActive(true);
    }

    public void CloseNameInputScreen() {
        nameInputScreen.SetActive(false);
    }

    public void CreateRoom() {
        if (!string.IsNullOrEmpty(roomNameInput.text)) {
            RoomOptions roomOptions = new RoomOptions();
            roomOptions.MaxPlayers = 6;
            PhotonNetwork.CreateRoom(roomNameInput.text, roomOptions);
            CloseCreateRoomScreen();
            loadingText.text = "Creating room...";
            OpenLoadingScreen();
        }
    }

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}
public override void OnJoinedRoom() {
    CloseCreateRoomScreen();
    OpenRoomScreen();
    roomNameText.text = PhotonNetwork.CurrentRoom.Name;
    ListAllPlayers();
    if (PhotonNetwork.IsMasterClient)
    {
        startButton.SetActive(true);
    }
    else
    {
        startButton.SetActive(false);
    }
}
private void ListAllPlayers() {
    foreach (TMP_Text player in allPlayerNames) {
        Destroy(player.gameObject);
    }
    allPlayerNames.Clear();
    Player[] players = PhotonNetwork.PlayerList;
    for (int i = 0; i < players.Length; i++) {
        TMP_Text newPlayerLabel = Instantiate(playerNameLabel,
playerNameLabel.transform.parent);
        newPlayerLabel.text = players[i].NickName;
        newPlayerLabel.gameObject.SetActive(true);
        allPlayerNames.Add(newPlayerLabel);
    }
}
public override void OnPlayerEnteredRoom(Player newPlayer) {
    TMP_Text newPlayerLabel = Instantiate(playerNameLabel,
playerNameLabel.transform.parent);
    newPlayerLabel.text = newPlayer.NickName;
    newPlayerLabel.gameObject.SetActive(true);
    allPlayerNames.Add(newPlayerLabel);
}
public override void OnPlayerLeftRoom(Player otherPlayer) {
    ListAllPlayers();
}
public override void OnCreateRoomFailed(short returnCode, string message) {
    errorText.text = "Error in creating a room: " + message;
    OpenErrorScreen();
}
public void LeaveRoom() {
    PhotonNetwork.LeaveRoom();
    CloseCreateRoomScreen();
    CloseRoomScreen();
    loadingText.text = "Leaving a room...";
    OpenLoadingScreen();
}
public override void OnRoomListUpdate(List<RoomInfo> roomList) {
    foreach (MRoomButton rb in allRoomButtons) {
        Destroy(rb.gameObject);
    }
    allRoomButtons.Clear();
    roomButton.gameObject.SetActive(false);
    for(int i = 0; i < roomList.Count; i++) {
        if (roomList[i].PlayerCount != roomList[i].MaxPlayers &&
!roomList[i].RemovedFromList)
        {
            MRoomButton newButton = Instantiate(roomButton,
roomButton.transform.parent);
            newButton.SetButtonDetails(roomList[i]);
            newButton.gameObject.SetActive(true);
            allRoomButtons.Add(newButton);
        }
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}
public void JoinRoom(RoomInfo inputInfo) {
    PhotonNetwork.JoinRoom(inputInfo.Name);
    CloseFindRoomScreen();
    loadingText.text = "Joining room...";
    OpenLoadingScreen();
}
}

```

Файл MatchManager.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using Photon.Pun;
public class MatchManager : MonoBehaviour {
    public static MatchManager instance;
    private void Awake() {
        instance = this;
    }
    void Start() {
        if(!PhotonNetwork.IsConnected) {
            SceneManager.LoadScene(0);
        }
    }
}

```

Файл MRoomButton.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using Photon.Realtime;
public class MRoomButton : MonoBehaviour {
    public TMP_Text buttonText;
    private RoomInfo info;
    public void SetButtonDetails(RoomInfo inputInfo) {
        info = inputInfo;
        buttonText.text = info.Name;
    }
    public void OpenRoom() {
        MLauncher.instance.JoinRoom(info);
    }
}

```

Файл MSpawnManager.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MSpawnManager : MonoBehaviour {
    public Transform[] spawnPoints;
    public static MSpawnManager instance;
    private void Awake() {
        instance = this;
    }
    public Transform GetSpawnPoint() {
        return spawnPoints[Random.Range(0, spawnPoints.Length)];
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						101
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл MPlayerSpawner.cs

```
using FishNet.Component.Spawning;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
public class MPlayerSpawner : MonoBehaviour {
    public static MPlayerSpawner instance;
    private void Awake() {
        instance = this;
    }
    public GameObject playerPrefab;
    private GameObject player;
    public float respawnTime = 5f;
    void Start() {
        if (PhotonNetwork.IsConnected) {
            SpawnPlayer();
        }
    }
    public void SpawnPlayer() {
        Transform spawnPoint = MSpawnManager.instance.GetSpawnPoint();
        player = PhotonNetwork.Instantiate(playerPrefab.name, spawnPoint.position,
spawnPoint.rotation);
    }
    public void Die(string damager) {
        MGameManager.instance.deathText.text = "You were killed by " + damager;
        if (player != null) {
            StartCoroutine(DieCoroutine());
        }
    }
    public IEnumerator DieCoroutine() {
        PhotonNetwork.Destroy(player);
        MGameManager.instance.deathScreen.SetActive(true);
        yield return new WaitForSeconds(respawnTime);
        MGameManager.instance.deathScreen.SetActive(false);
        SpawnPlayer();
    }
}
```

Файл MGameManager.cs

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
public class MGameManager : MonoBehaviour {
    public static MGameManager instance;
    private void Awake() {
        instance = this;
    }
    public GameObject deathScreen;
    public TMP_Text deathText;
    public GameObject bloodScreen;
}
```

Файл MPlayerController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using FishNet.Component.Spawning;
using UnityEngine.SceneManagement;
using TMPro;
public class MPlayerController : MonoBehaviourPunCallbacks {
    public Transform viewPoint;
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public float mouseSensitivity = 1f;
private float verticalRotStore;
private Vector2 mouseInput;
public int health = 100;
public int scores = 0;
public float moveSpeed = 12f;
public float runSpeed = 12f;
public Animator animator;
public CharacterController characterController;
public AudioSource shootingSound;
public float jumpForce = 7f;
public float gravityMod = 2.5f;
public Transform groundCheckPoint;
private bool isGrounded;
public LayerMask groundLayers;
public GameObject playerModel;
public GameObject weaponModel;
public GameObject bulletImpact;
private TMP_Text playerNameText;
public Gun[] allGuns;
private int selectedGun;
private Camera cam;
private Vector3 moveDirection;
private Vector3 movement;
void Start() {
    var photonView = GetComponent<PhotonView>();
    playerNameText = GetComponentInChildren<TMP_Text>();
    playerNameText.text = photonView.Owner.NickName;
    SwitchGun();
    cam = GameObject.FindGameObjectWithTag("MainCamera").GetComponent<Camera>();
    Cursor.lockState = CursorLockMode.Locked;
    if (photonView.IsMine) {
        playerModel.SetActive(false);
        weaponModel.SetActive(true);
        MPlayerHealthBar.instance.SetMaxHealth(health);
    }
}
void Update() {
    if (Input.GetKeyDown(KeyCode.Escape)) {
        Time.timeScale = 1f;
        MPlayerHealthBar.instance.SavingScoresExitScreen();
        MPlayerHealthBar.instance.SaveHighestScores(scores);
        SceneManager.LoadScene(0);
    }
    MPlayerHealthBar.instance.SetPlayersQuantityNowText(PhotonNetwork.CurrentRoom.PlayerCount);
    if (photonView.IsMine) {
        mouseInput = new Vector2(Input.GetAxisRaw("Mouse X"),
Input.GetAxisRaw("Mouse Y")) * mouseSensitivity;
        transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles.x,
transform.rotation.eulerAngles.y + mouseInput.x, transform.rotation.eulerAngles.z);
        verticalRotStore += mouseInput.y;
        verticalRotStore = Mathf.Clamp(verticalRotStore, -60f, 60f);
        viewPoint.rotation = Quaternion.Euler(-verticalRotStore,
viewPoint.rotation.eulerAngles.y, viewPoint.rotation.eulerAngles.z);
        moveDirection = new Vector3(Input.GetAxisRaw("Horizontal"), 0f,
Input.GetAxisRaw("Vertical"));
        float yVel = movement.y;
        movement = ((transform.forward * moveDirection.z) + (transform.right *
moveDirection.x)).normalized * moveSpeed;
        movement.y = yVel;
        if (characterController.isGrounded) {
            movement.y = 0f;
        }
        isGrounded = Physics.Raycast(groundCheckPoint.position, Vector3.down,
.25f, groundLayers);
        if (Input.GetButtonDown("Jump") && isGrounded) {

```

					KH.KP.Б – 122 – 23 – ПЗ	Арк.
						103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        movement.y = jumpForce;
    }
    movement.y += Physics.gravity.y * Time.deltaTime * gravityMod;
    characterController.Move(movement * Time.deltaTime);
    if(Input.GetMouseButtonDown(0) || Input.GetKey(KeyCode.F)) {
        Shoot();
        GameObject currentActiveGun = GameObject.FindGameObjectWithTag("Gun");
        Animator gunAnimator = currentActiveGun.GetComponent<Animator>();
        gunAnimator.SetBool("isShooting", true);
        StartCoroutine(ResetIsShooting(gunAnimator));
    }
    if(Input.GetAxisRaw("Mouse ScrollWheel") > 0f) {
        selectedGun++;
        if(selectedGun >= allGuns.Length) {
            selectedGun = 0;
        }
        SwitchGun();
    }
    else if (Input.GetAxisRaw("Mouse ScrollWheel") < 0f) {
        selectedGun--;
        if (selectedGun < 0) {
            selectedGun = allGuns.Length - 1;
        }
        SwitchGun();
    }
    if(isGrounded) {
        animator.SetBool("isGrounded", true);
    }
    else {
        animator.SetBool("isGrounded", false);
    }
    if (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow) ||
    Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.DownArrow) || Input.GetKey(KeyCode.A)
    || Input.GetKey(KeyCode.LeftArrow) || Input.GetKey(KeyCode.D) ||
    Input.GetKey(KeyCode.RightArrow)) {
        animator.SetBool("isWalking", true);
    }
    else {
        animator.SetBool("isWalking", false);
    }
    }}}
IEnumerator ResetIsShooting(Animator gunAnimator) {
    yield return new WaitForSeconds(1f);
    gunAnimator.SetBool("isShooting", false);
}
private void Shoot() {
    shootingSound.Play();
    Ray ray = cam.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0f));
    ray.origin = cam.transform.position;
    if (Physics.Raycast(ray, out RaycastHit hit)) {
        if (hit.collider.gameObject.tag == "Player") {
            scores += 10;
            MPlayerHealthBar.instance.SetScores(scores);
            hit.collider.gameObject.GetPhotonView().RPC("DealDamage",
RpcTarget.All, photonView.Owner.NickName, allGuns[selectedGun].damage);
        }
        GameObject bulletImpactObject = Instantiate(bulletImpact, hit.point +
(hit.normal * 0.002f), Quaternion.LookRotation(hit.normal, Vector3.up));
        Destroy(bulletImpactObject, 5f);
    }
}
[PunRPC]
public void DealDamage(string damager, int damageAmount) {
    TakeDamage(damager, damageAmount);
}
public void TakeDamage(string damager, int damageAmount) {
    if (photonView.IsMine) {

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						104
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        health -= damageAmount;
        MPlayerHealthBar.instance.ShowInjuredScreen();
        MPlayerHealthBar.instance.SetHealth(health);
        if (health <= 0) {
            health = 0;
            MPlayerHealthBar.instance.SaveHighestScores(scores);
            MPlayerSpawner.instance.Die(damager);
        }
    }
}

private void LateUpdate() {
    if (photonView.IsMine) {
        cam.transform.position = viewPoint.position;
        cam.transform.rotation = viewPoint.rotation;
    }
}

void SwitchGun() {
    foreach(Gun gun in allGuns) {
        gun.gameObject.SetActive(false);
    }
    allGuns[selectedGun].gameObject.SetActive(true);
}
}

```

Файл MPlayerHealthBar.cs

```

using System.Collections;
using System.Collections.Generic;
using TMPPro;
using UnityEngine;
using UnityEngine.SocialPlatforms.Impl;
using UnityEngine.UI;
public class MPlayerHealthBar : MonoBehaviour {
    public static MPlayerHealthBar instance;
    private void Awake() {
        instance = this;
    }
    public Slider slider;
    public TMP_Text healthText;
    public TMP_Text scoresText;
    public GameObject injuredScreen;
    public GameObject savingScoresScreen;
    public TMP_Text playersQuantityNowText;
    public void SetMaxHealth(int maxHealth) {
        healthText.text = maxHealth.ToString();
        slider.maxValue = maxHealth;
        slider.value = maxHealth;
    }
    public void SetHealth(float health) {
        slider.value = health;
        if(health >= 0) {
            healthText.text = health.ToString();
        }
        else {
            healthText.text = "0";
        }
    }
    public void SaveHighestScores(int scores) {
        int prevScores = PlayerPrefs.GetInt("MultiplayerScores");
        if (prevScores < scores) {
            PlayerPrefs.SetInt("MultiplayerScores", scores);
        }
    }
    public void SavingScoresExitScreen() {
        savingScoresScreen.SetActive(true);
    }
}

```

```

    }
    public void SetPlayersQuantityNowText(int playersQuantity) {
        playersQuantityNowText.text = "Players quantity now: " +
playersQuantity.ToString();
    }
    public void SetScores(int scores) {
        scoresText.text = "Scores " + scores.ToString();
    }
    public void ShowInjuredScreen() {
        injuredScreen.SetActive(true);
        Invoke("RemoveInjuredScreen", 1f);
    }
    public void RemoveInjuredScreen() {
        injuredScreen.SetActive(false);
    }
}

```

Файл MMinimap.cs

```

using Photon.Pun;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MMinimap : MonoBehaviour {
    public Transform player;
    int myID = PhotonNetwork.LocalPlayer.ActorNumber;
    GameObject[] players;
    GameObject myPlayer;
    void Start() {
        players = GameObject.FindGameObjectsWithTag("Player");
        foreach (GameObject player in players) {
            int playerActorNumber =
player.GetComponent<PhotonView>().Owner.ActorNumber;
            if (playerActorNumber == myID) {
                myPlayer = player;
            }
        }
        if (myPlayer != null) {
            player = myPlayer.transform;
        }
    }
    void LateUpdate() {
        players = GameObject.FindGameObjectsWithTag("Player");
        foreach (GameObject player in players) {
            int playerActorNumber =
player.GetComponent<PhotonView>().Owner.ActorNumber;
            if (playerActorNumber == myID) {
                myPlayer = player;
            }
        }
        if (myPlayer != null) {
            player = myPlayer.transform;
        }
        if(player != null) {
            Vector3 newPosition = player.position;
            newPosition.y = transform.position.y;
            transform.position = newPosition;
            transform.rotation = Quaternion.Euler(90f, player.eulerAngles.y, 180f);
        }
    }
}

```