

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Розробка гри у жанрі FPS «Survival»

Виконав студент групи ПЗк-20-1
НАДВОРНИЙ Максим Юрійович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ТОЛСТОЙ Ігор Анатолійович

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **НАДВОРНИЙ Максим Юрійович**

Керівник роботи: **ЧИЖМОТРЕ Олена Геннадіївна**

Тема роботи: **«Розробка гри у жанрі FPS “Survival”»**,

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р. №70с

Вихідні дані для роботи: ефективне використання рушія Unity3D для створення гри в жанрі FPS .

Об'єктом дослідження є процес створення ігрового середовища, механік гри, графічного виконання та програмування інтерактивності.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	16.02.2023р.	16.02.2023р.
2	Чижмотря О.Г.	16.02.2023р.	16.02.2023р.
3	Чижмотря О.Г.	16.02.2023р.	16.02.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	16 лютого 2023 – 18 лютого 2023	виконано
2	Аналіз аналогів програмного продукту	18 лютого 2023 – 23 лютого 2023	виконано
3	Визначення вимог до додатку	23 лютого 2023 – 2 березня 2023	виконано
4	Вибір інструментальних засобів	2 березня 2023 – 4 березня 2023	виконано
5	Визначення варіантів використання	4 березня 2023 – 11 березня 2023	виконано
6	Проектування та реалізація алгоритмів системи	11 березня 2023 – 25 березня 2023	виконано
7	Написання програмного коду	25 березня 2023 – 1 травня 2023	виконано
8	Тестування роботи програмного продукту	1 травня 2023 – 12 травня 2023	виконано
9	Оформлення пояснювальної записки	12 травня 2023 – 30 травня 2023	виконано
10	Попередній захист	5 травня	виконано

РЕФЕРАТ

Випускна робота бакалавра складається з розробки гри у жанрі FPS “Survival” та пояснювальної записки. Пояснювальна записка до випускної роботи містить 85 сторінок, 40 ілюстрацій та 3 таблиці, 14 додатків та 15 літературних джерел.

Метою роботи є розробка за допомогою рушія Unity гри у жанрі FPS.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об’єктної структури системи (діаграма класів). Також наводиться результати тестування системи.

КЛЮЧОВІ СЛОВА: FPS, СИМУЛЯТОР, ГРА, ШУТЕР.

					ІПЗ.КР.Б – 121 – 23 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		М.Ю. Надворний			Пояснювальна записка Розробка гри у жанрі FPS “Survival”	Літ.	Арк.	Аркуші
Перевір.							4	85
Керівник		О.Г. Чижмотря				Житомирська політехніка, Група ІПЗк 20-1		
Рецензент.		І.А. Толстой						
Зав. каф.		А.В. Морозов						

ABSTRACT

The bachelor's thesis consists of the development of a game in the genre of FPS "Survival" and an explanatory note. The explanatory note to the final thesis contains 85 pages, 40 illustrations and 3 tables, 14 appendices and 15 literary sources.

The purpose of the work is to develop a FPS game using the Unity engine.

The paper defines the main tasks for the development of the system, and analyzes the analogues of the developed system. Scenarios of the software complex, the general structure of the software complex, a description of the interaction algorithms of individual system modules, a description of the object structure of the system (class diagram) are given. The results of system testing are also provided.

KEY WORDS: FPS, SIMULATOR, GAME, SHOOTER.

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ГРИ У ЖАНРІ FPS	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів програмного продукту.....	11
1.3. Вибір архітектури	18
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ГРИ FPS.....	27
2.1. Визначення варіантів використання та структури системи.	27
2.2. Проектування та реалізація алгоритмів роботи системи.....	40
2.3. Реалізація функціоналу додатку.....	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ СИМУЛЯЦІЇ.....	47
3.1. Порядок встановлення та налаштування системи.....	47
3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з грою.	47
3.3. Тестування роботи системи	55
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ.....	60

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

FPS – First-person shooter

ЛКМ – Ліва кнопка миші

ПКМ – Права кнопка миші

UE4 – Unreal Engine 4

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми: Зважаючи на те, що гра у жанрі FPS є одним з найбільш популярних видів відеоігор, особливо серед молоді, вивчення даної теми має важливість з наукової, практичної та соціальної точок зору. Подібне дослідження дозволить визначити основні фактори, що впливають на відчуття гравців під час гри, їх емоції та поведінку, а також вивчити історію розвитку жанру FPS та дослідити його вплив на сучасну культуру. Крім того, дослідження даної теми має можливість допомогти в розробці більш якісних ігор, що задовільняють вимоги та потреби гравців, а також сприяти розвитку індустрії відеоігор в цілому. Отже, актуальність теми полягає в її значимості для наукових та практичних галузей, а також для суспільства в цілому.

Мета випускної роботи: Розробка та створення повноцінної гри в жанрі FPS з використанням Unity.

Встановлена мета обумовлює наступні завдання:

- Розробка іммерсивного геймплею: Створення захоплюючого, реалістичного та динамічного геймплею, що характеризується відчуттям присутності гравця у грі;
- Реалізація механік гри: Включення основних механік гри, які є характерними для жанру FPS, наприклад: рух гравця, стрільба, зброя та бойові системи, штучний інтелект ворогів тощо;
- Розробка рівнів та дизайну рівня: Створення різноманітних рівнів гри, у тому числі локацій, об'єктів, лабіринтів та складних розташувань, щоб гравець мав цікавий досвід проходження гри;
- Візуальні ефекти та атмосфера: Впровадження вражаючих візуальних ефектів, освітлення, звукових ефектів та музики, щоб створити відповідну атмосферу гри та підкреслити настрій гравця;

Об'єктом дослідження є процес створення ігрового середовища, механік гри, графічного виконання та програмування інтерактивності.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

Предметом дослідження є ефективне використання рушія Unity3D для створення гри в жанрі FPS.

Методи дослідження в роботі було використано методи об'єктно орієнтованого проектування, веб-програмування та дизайну інтерфейсів.

Розроблена система може бути використана для розваги.

Пояснювальна записка до випускної роботи містить 91 сторінок, 39 ілюстрацій та 3 таблиць, список літературних джерел містить 10 найменувань.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ГРИ У ЖАНРІ FPS

1.1. Постановка задачі

Основною метою розробки гри у жанрі FPS є створення захоплюючого та емоційного ігрового досвіду для гравця. Гра повинна надати можливість гравцеві відчувати себе у ролі головного героя, зануритися в ігровий світ і взаємодіяти з ним.

В випускній роботі необхідно розробити додаток у вигляді гри у жанрі FPS, якомога більше наблизити стрільбу до реалістичного. Для реалізації поставленого завдання, основними етапами є:

1. Розробити дизайн користувацького інтерфейсу.

1.1. Створити загальну концепцію дизайну користувацького інтерфейсу.

1.2. Зверстати та застосувати елементи інтерфейсу за допомогою UnityEditor:

1.2.1. Головне меню

1.2.2. Панель меню паузи

1.2.3. Панель вікна перемоги

2. Проектування та розробка основних ігрових модулів

1.3. Модуль керування персонажем

1.4. Модуль ворогів

1.5. Модуль контрольної точки

3. Моделювання 3D об'єктів за допомогою Blender3D

1.6. Гори

1.7. Космічні предмети

1.8. Ворогів

1.9. Зброю та інші предмети

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

На сьогоднішній день FPS (перша особа) шутери є одними з найпопулярніших жанрів в ігровій індустрії. Вони надають можливість гравцеві зануритись у захоплюючий світ бойових дій і відчувати себе частиною гри. Симулятори FPS шутерів використовуються як для розважальних цілей, так і для тренування навичок гравців, удосконалення тактики та командної гри.

Основними характеристиками симуляторів FPS шутерів є:

1. Фізичний рушій: Якість фізичної моделі гри впливає на реалістичність рухів персонажів, взаємодію з оточуючим середовищем, а також на поведінку різних об'єктів у грі, таких як зброя, вибухи, транспортні засоби тощо;
2. Платформа: Важливою характеристикою є платформа, на якій доступний симулятор. Це може бути персональний комп'ютер, ігрова консоль, або мобільний пристрій;
3. Зручність GUI: Інтерфейс користувача (GUI) повинен бути зручним та легким у використанні, забезпечуючи зручну навігацію, налаштування гри, менеджмент інвентарю та інші функції;
4. Графічне оформлення: Рівень деталізації графіки, реалістичність об'єктів, освітлення та спеціальні ефекти впливають на іммерсивність гри та задоволення від геймплею;
5. Мультиплеєр: Багатокористувацький режим дозволяє гравцям взаємодіяти між собою, утворювати команди та змагатись один з одним. В разі FPS шутерів, мультиплеєр є одним з найважливіших елементів гри, оскільки надає можливість гравцям зіграти проти реальних супротивників з усього світу;
6. Віртуальний гравець: Наявність віртуального гравця у симуляторах FPS шутерів також є важливою характеристикою. Віртуальний гравець - це комп'ютерно контрольовані персонажі, з якими можна грати або які виконують роль супротивників у відсутність реальних гравців. Вони дозволяють гравцеві насолоджуватись грою навіть у

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

випадку, коли доступ до Інтернету відсутній або коли немає можливості грати з реальними гравцями.

Розглянемо детально мультиплеєрний режим та режим одиночної гри з віртуальними гравцями.

Мультиплеєр. Може мати декілька режимів гри, таких як кооперативний режим, командний режим, змагальні режими та інші. Кооперативний режим дозволяє гравцям спільно виконувати завдання або проходити рівні, співпрацюючи для досягнення спільної мети. Командний режим передбачає поділ гравців на команди, які змагаються між собою в різних сценаріях, таких як захоплення прапора, утримання точок або знищення цілей противника.

Мультиплеєр в FPS шутерах також може мати систему рейтингу, де гравці отримують пункти або підвищують свій ранг в залежності від результатів своєї гри. Це стимулює гравців до покращення своїх навичок та дослідження різних стратегій гри.

Деякі FPS шутери також надають можливість гравцям створювати власні сервери або приєднуватись до існуючих серверів, що дозволяє гравцям грати зі своїми друзями або приєднуватись до спільноти гравців зі схожими інтересами.

Мультиплеєр в FPS шутерах може бути реалізований як локально, де гравці можуть підключатись до однієї мережі, або глобально через Інтернет, де гравці можуть зв'язуватись з іншими гравцями з усього світу. Глобальний мультиплеєр через Інтернет надає гравцям можливість зустрічатись і грати з іншими гравцями навіть на великій відстані.

Для забезпечення багатокористувацької взаємодії, гри з мультиплеєром зазвичай мають вбудовану систему мережевого з'єднання, яка дозволяє гравцям підключатись до спеціальних серверів гри або безпосередньо між собою. Це забезпечує можливість обміну даними між гравцями, синхронізацію дій та відображення актуальної інформації на екранах всіх учасників гри.

Зазвичай, мультиплеєр в FPS шутерах вимагає підключення до Інтернету, оскільки гравці мають можливість грати з іншими гравцями з усього світу. Це

		Надворний М.Ю			ПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

може вимагати стабільного Інтернет-з'єднання з достатньою швидкістю, щоб забезпечити плавну гру без затримок або лагів.

Загалом, мультиплеєр в FPS шутерах надає гравцям можливість взаємодіяти, спілкуватись і змагатись з іншими гравцями, роблячи гру більш захопливою і динамічною. Приклади таких ігор: Counter-Strike, Call of Duty, Overwatch: Overwatch.

Одиночна гра з віртуальними гравцями. Основною особливістю віртуальних гравців є те, що вони можуть мати різний рівень інтелекту та навичок, що впливає на їхню поведінку у грі. Вони можуть мати просту програму, яка дозволяє їм рухатися і стріляти, тоді як інші можуть мати складніші алгоритми, що дозволяють їм приймати рішення, використовувати тактику та співпрацювати з іншими віртуальними гравцями.

Наявність ботів у симуляторах FPS шутерів дозволяє гравцям тренуватися, вдосконалювати свої навички та досліджувати різні аспекти гри навіть у відсутність реальних гравців. Вони також можуть створювати відчуття повноцінного мультиплеєра, де гравець може змагатись з розумними противниками. Приклади таких ігор: DOOM, F.E.A.R, Unreal Tournament.

Найбільший інтерес в рамках дипломного проекту являє режим одиночної гри з віртуальними гравцями. Їх порівняльна характеристика наведена у табл. 1.1.

Таблиця 1.1.

Порівняльна характеристика

Назва	Фізичний рушій	Платформа	Зручність GUI	Графічне оформлення	Підтримка 64-розрядних ОС
DOOM	Добре	Windows, PS4, Xbox One	Добре	Добре	Так
F.E.A.R	Добре	Windows, Xbox 360	Нормально	Добре	Так
Unreal Tournament	Добре	Windows, Linux, Mac OS	Добре	Добре	Так

Розглянемо більш детально табл. 1.1.



Рис. 1.1. DOOM

Фізичний рушій: В грі DOOM (2016) фізичний рушій реалістичний та добре виконаний. Рухи персонажа, взаємодія з оточуючим світом та взаємодія зі зброєю допомагають створити швидкий та експресивний геймплей.

Платформа: Гра доступна на платформах Windows, PlayStation 4 та Xbox One, що дозволяє гравцям насолоджуватися її на різних пристроях.

Зручність GUI: Інтерфейс користувача в DOOM (2016) забезпечує зручну навігацію та швидкий доступ до геймплейних опцій. Гравці можуть легко керувати налаштуваннями, обмінюватися зброєю та взаємодіяти з ігровим світом.

Графічне оформлення: Графічне виконання DOOM (2016) вражає високою деталізацією, спеціальними ефектами та відмінним освітленням. Це створює реалістичну та насичену графіку, яка підсилює атмосферу ігрового світу.

Підтримка 64-розрядних ОС: Гра DOOM (2016) підтримує 64-розрядні операційні системи, що дозволяє використовувати більш потужні можливості сучасних комп'ютерів.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14



Рис. 1.2. F.E.A.R.

Фізичний рушій: У F.E.A.R. фізичний рушій добре виконаний, що дозволяє гравцям насолоджуватися реалістичними рухами персонажів та інтенсивними бойовими сценами.

Платформа: Гра доступна на платформах Windows та Xbox 360, що дозволяє гравцям вибрати зручний для них пристрій для гри.

Зручність GUI: Інтерфейс користувача в F.E.A.R. забезпечує зручну навігацію та простоту використання. Гравці можуть легко керувати налаштуваннями, вибирати зброю та виконувати інші геймплейні дії.

Графічне оформлення: Графічне виконання F.E.A.R. вражає деталізацією та реалістичним стилем. Особливу увагу приділено динамічним ефектам, освітленню та атмосферному дизайну, що створює напружену ігрову атмосферу.

Підтримка 64-розрядних ОС: Гра F.E.A.R. підтримує 64-розрядні операційні системи, що дозволяє використовувати більш потужні можливості сучасних комп'ютерів.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15



Рис. 1.3.Unreal Tournament

Фізичний рушій: Unreal Tournament використовує Unreal Engine, який пропонує високоякісний фізичний рух, реалістичну взаємодію з оточуючим середовищем та динамічні ефекти. Геймплей є швидким та енергійним.

Платформа: Гра доступна на різних платформах, включаючи Windows, Linux та macOS, що робить її доступною для широкого кола гравців.

Зручність GUI: Інтерфейс користувача в Unreal Tournament добре пророблений, забезпечуючи зручну навігацію, налаштування гри та інші функції. Гравці можуть легко управляти грою та взаємодіяти з ігровим світом.

Графічне оформлення: Графіка в Unreal Tournament вражає деталізацією, освітленням та спеціальними ефектами. Рівень деталізації об'єктів, текстури та моделі персонажів є високим, що дозволяє створити реалістичні та привабливі візуальні ефекти. Освітлення гри допомагає створити настрій та атмосферу в різних локаціях, а спеціальні ефекти, такі як вибухи, вогонь, частинки, рухомі тіні, підсилюють враження від геймплею.

Підтримка 64-розрядних ОС: Гра Unreal Tournament підтримує 64-розрядні операційні системи, що дозволяє використовувати більш потужні можливості сучасних комп'ютерів.

Розглянемо функціональні можливості даних ігор.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.2.

Функціональні можливості

Назва	Зброя	Мультиплеєр	Анімація	Швидкість гри	ШІ ворогів
DOOM	Різноманітність смертоносної зброї	Різні режими гри	Плавні та реалістичні рухи персонажів	Швидкий темп геймплею та руху персонажа	Розумна поведінка ворожих сутностей
F.E.A.R	Великий вибір зброї та екзоскелетів	Мультиплеєрний режим з командними битвами	Реалістична анімація руху та бою	Реалістична швидкість гри	Розумна поведінка ворожих сутностей
Unreal Tournament	Великий арсенал зброї та спеціальних засобів	Мультиплеєр з великим вибором режимів гри	Реалістична анімація персонажів та зброї	Швидкий темп геймплею та маневреність	Розумна поведінка ворожих сутностей

Розглянемо мінімальні системні вимоги порівнюваних аналогів

Таблиця 1.3.

Мінімальні системні вимоги порівнюваних аналогів

Назва	Процесор	RAM	Відеокарта	Місце на диску
DOOM	Intel Core i5-2400	8 GB	NVIDIA GTX 670 / AMD Radeon HD 7870	55 GB
F.E.A.R	Intel Core 2 Duo 2.4Ghz	1 GB	NVIDIA 6800 GT / ATI X800	17 GB
Unreal Tournament	Intel Core 2 Duo 2.4Ghz	2 GB	NVIDIA GeForce 6	8 GB

Переглянувши та проаналізувавши аналоги FPS можна із впевненістю сказати, що кожен програмний продукт суттєво відрізняється від своїх конкурентів, що створює певну різноманітність та свободу вибору для користувача, хоча містить ряд недоліків.

Таким чином, основними рисами даного продукту мають бути: сучасний фізичний рушій, наявність зручного GUI, можливість вільно пересуватись

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

гравцем ігровим світом, взаємодія із дрібними ігровими елементами, звукові ефекти.

1.3. Вибір архітектури

Для розробки даної системи є можливість використати наступні патерни:

- MVC
- ECS
- MVP
- MVVM
- CBA

Адаптування патернів типу MVC, MVVM чи MVP під розробку ігрового продукту в результаті могло б спричинити плутанину серед об'єктів GameObject та їх батьківських елементів, що є неприйнятним для розробки на ігровому рушії гри такого типу.

Тому мною було використано Component-Based Architecture.

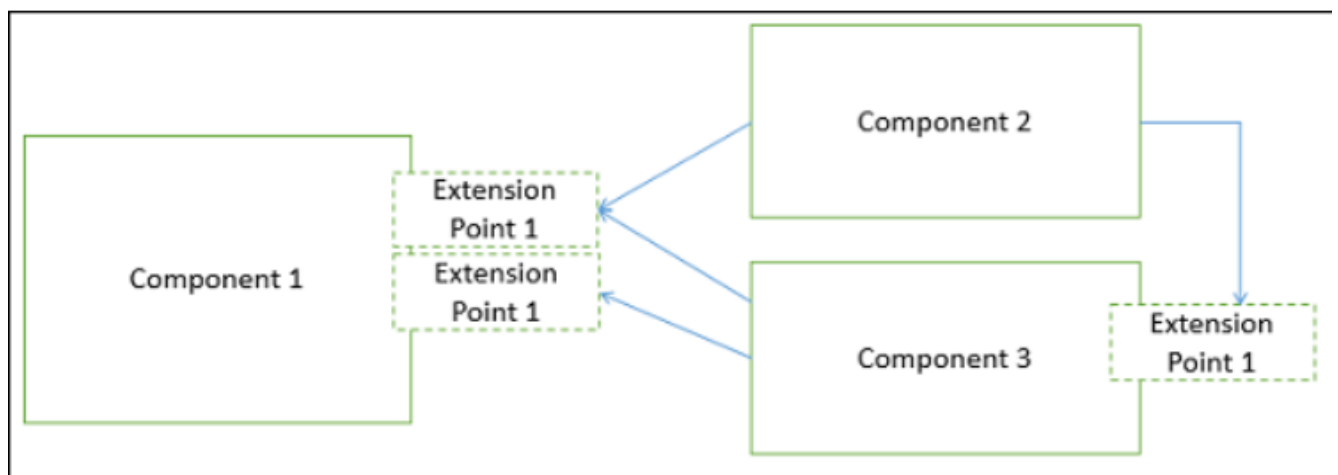


Рис. 1.4. Component-Based Architecture

Component-Based Architecture є підходом до проектування програмного забезпечення, в основі якого лежить концепція компонентів. Компоненти є незалежними модулями, які мають чітку функціональність та можуть бути перевикористані в різних частинах програми. Вони представляють окремі будівельні блоки програми, які можуть включати як код, так і дані [11].

Основні принципи CBA:

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

Перевикористання коду: Через використання компонентів, які вирішують конкретні завдання, можливе значне перевикористання коду в різних частинах програми. Наприклад, якщо у вас є компонент, відповідальний за керування гравцем, ви можете використовувати цей компонент як частину інших сценаріїв, де потрібно керування гравцем. Це економить час і зусилля, оскільки вам не потрібно знову писати весь код для керування гравцем.

Легка інтеграція: Компоненти взаємодіють один з одним через інтерфейси або події, що дозволяє легко інтегрувати нові компоненти до системи. Наприклад, якщо ви хочете додати новий компонент для керування ворогами, ви можете створити новий компонент, який реалізує необхідні інтерфейси або підписується на потрібні події. Це дозволяє додавати новий функціонал до системи без необхідності вносити зміни в існуючі компоненти.

Легка модифікація: Компоненти можуть бути легко модифіковані або замінені без необхідності вносити зміни в інші частини програми [4]. Наприклад, якщо ви хочете змінити рух ворогів, ви можете замінити компонент, відповідальний за рух ворогів, на новий компонент з більш складною логікою руху. Це дає вам гнучкість та швидкість змін у системі.

Спрощена розробка та тестування: Розділення програми на компоненти спрощує процес розробки та тестування. Кожен компонент може бути розроблений та перевірений окремо, що полегшує виявлення та усунення помилок.

Складність управління: З використанням багатьох компонентів може з'явитися потреба у додаткових механізмах для керування цими компонентами та їх взаємодією. Наприклад, можуть знадобитися системи реєстрації, управління подіями або інші механізми для забезпечення координації роботи компонентів.

Загалом, Component-Based Architecture є потужним підходом до проектування програмного забезпечення, який дозволяє розділити систему на незалежні компоненти, що забезпечують перевикористання коду, легку модифікацію та розширення функціоналу. Однак, перед використанням СВА у

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

своєму дипломному проєкті, важливо провести детальний аналіз вимог та обґрунтувати вибір цієї архітектури в контексті вашого проєкту.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.

Найбільш популярними рушіями для розробки ігор сьогодні є Unity3D, UE4, Marmalade і Cocos2D. У кожного з них є свої переваги. Але оскільки необхідно розробити кросплатформенну гру з використанням фізичного рушія, то на першому місці за актуальними дослідженнями знаходиться Unity.

Рушій (двигун) Unity є потужним інструментом для розробки ігор та інтерактивних додатків. Він надає розробникам широкі можливості для створення вражаючої графіки, фізики, анімації, звукових ефектів та ігрового геймплею [9]. Ось кілька переваг Unity порівняно з іншими рушіями:

Хрестоматійність: Unity є одним з найпопулярніших рушіїв для розробки ігор. Він має велику та активну спільноту розробників, яка надає підтримку, документацію, плагіни та ресурси. Це робить Unity легким для вивчення та спільної роботи з іншими розробниками.

Кросплатформеність: Unity підтримує різні платформи, включаючи ПК, консолі, мобільні пристрої (iOS та Android), веб-браузери та віртуальну реальність. Це дозволяє розробникам створювати проєкти, які можуть бути використані на різних пристроях та платформах без необхідності переписування коду з нуля.

Широкий функціонал: Unity має вбудовані інструменти для створення графіки, фізики, анімації, штучного інтелекту, звуку та інших аспектів ігрового процесу. Він надає можливості для швидкої розробки та прототипування ігор, а також дозволяє розширювати його функціонал за допомогою плагінів та скриптів.

Інтегроване середовище розробки (IDE): Unity надає власне інтегроване середовище розробки, яке має зручний інтерфейс користувача та набір інструментів для написання, налагодження та управління проєктом. Це

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

середовище дозволяє розробникам ефективно працювати зі сценами, об'єктами, компонентами та скриптами, забезпечуючи зручну інтерактивну розробку.

Розширюваність: Unity надає можливості розширення за допомогою плагінів та скриптів. Розробники можуть використовувати різноманітні плагіни, створені спільнотою або сторонніми розробниками, щоб розширити функціонал Unity та додати нові можливості до своїх проєктів. Крім того, Unity підтримує скриптинг на мові C#, що дозволяє розробникам створювати власні скрипти та логіку гри з використанням потужних інструментів програмування.

Швидкість розробки: Unity пропонує зручні інструменти для швидкої розробки, такі як перетягування та розміщення об'єктів, візуальний редактор анімації, вбудовані фізичні двигуни та інші. Це дозволяє розробникам прототипувати, тестувати та ітерувати свої ідеї швидше, що є важливим аспектом в ігровій розробці.

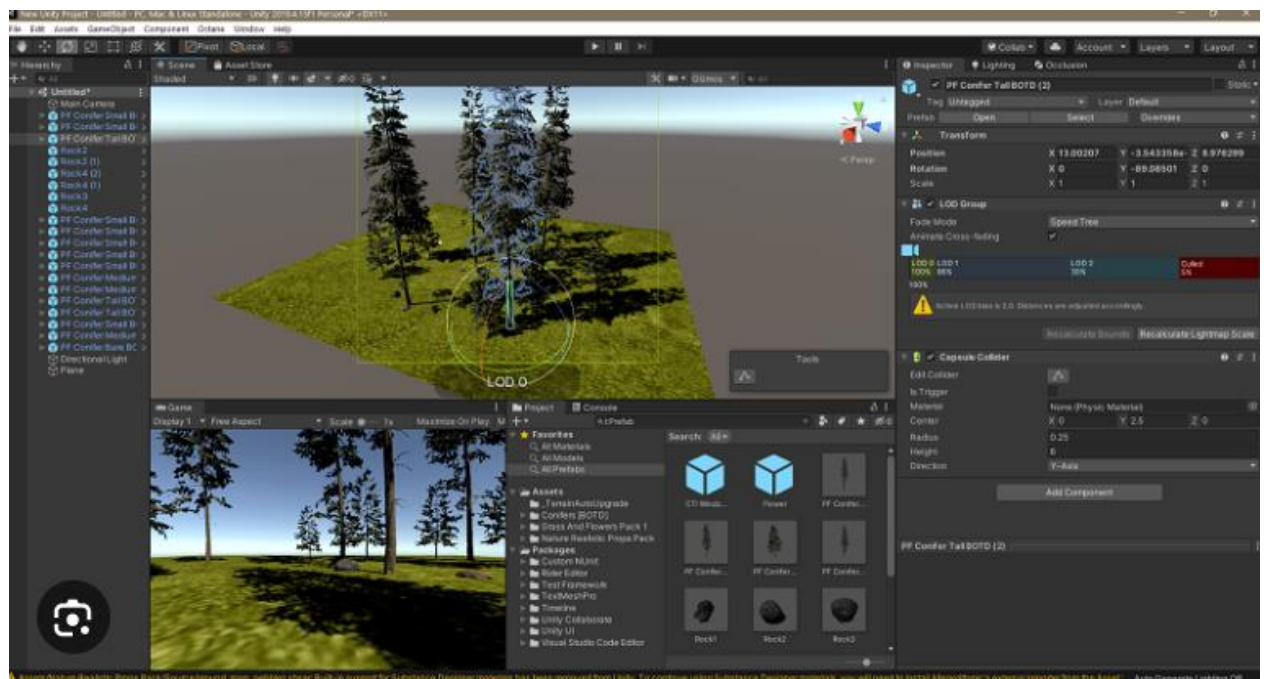


Рис. 1.5. Unity Editor

Інтегрована середовище розробки. Опис переваг Visual Studio Code:

Зручний інтерфейс користувача: VS Code має чистий та інтуїтивно зрозумілий інтерфейс, який дозволяє розробникам зосередитися на написанні коду без зайвих відволікань. Він має просте та ергономічне розташування

		Надворний М.Ю			ПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

панелей, швидко переходи між файлами та багато інших зручних функцій, які полегшують розробку.

Розширення та налаштування: VS Code підтримує велику кількість розширень, які дозволяють розробникам налаштовувати та розширювати функціонал редактора під свої потреби. Розширення доступні для різних мов програмування, фреймворків та інструментів розробки, що дозволяє зручно працювати в різних проектах.

Зручна робота з Git: VS Code має вбудовану підтримку для системи контролю версій Git, що дозволяє розробникам легко керувати та відстежувати зміни в своїх проектах. Він надає інтерфейс для комітів, візуальну зміну коду, гілки та інші корисні функції, які спрощують спільну роботу та співпрацю з командою.

Налагодження: VS Code надає зручні інструменти для налагодження коду, що дозволяє розробникам знайти й виправити помилки та відстежувати виконання програми. Його налагоджувальна панель має можливості точкового зупинення, крокування по коду, перевірки значень змінних та багато інших корисних функцій.

Розширені можливості редактора: VS Code надає розробникам широкий набір функцій, таких як автодоповнення коду, підсвічування синтаксису, перехід до визначення функцій та змінних, швидкий пошук та заміна тексту, інтеграція з системами збірки та багато інших. Ці функції допомагають підвищити продуктивність розробника та зробити роботу з кодом швидшою та зручнішою.

Спільнота та підтримка: VS Code має велику та активну спільноту розробників, яка надає підтримку, документацію та плагіни. Це означає, що можна знайти відповіді на запитання, рішення для проблем та корисні ресурси, щоб поліпшити роботу з редактором.

Інтеграція з іншими інструментами: VS Code може бути легко інтегрований з іншими інструментами розробки, такими як системи керування проектами, лінери, тестувальні фреймворки, середовища віртуальних машин та багато інших. Це дозволяє розробникам працювати в улюбленому середовищі,

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		22

використовуючи різні інструменти, які вони вважають найбільш ефективними для своїх проектів.

3D-редактор. Опис переваг Blender 3.5:

Безкоштовність: Blender є повністю безкоштовним програмним забезпеченням з відкритим вихідним кодом. Це означає, що ви можете використовувати його безкоштовно без обмежень і ліцензійних витрат. Ви також можете змінювати і пристосовувати програму до своїх потреб, оскільки вихідний код є доступним для всіх користувачів.

Кросплатформеність: Blender підтримує різні операційні системи, такі як Windows, macOS і Linux [2]. Це означає, що ви можете використовувати Blender на будь-якому комп'ютері, незалежно від операційної системи, що дозволяє більш широкий охоплення користувачів.

Комплексність та функціональність: Blender має вражаючий набір інструментів та функцій, які охоплюють всі аспекти процесу створення 3D-графіки [13]. Він має інструменти для моделювання, анімації, текстурювання, освітлення, рендерингу, симуляції фізики та багато іншого. Це дозволяє вам виконувати різноманітні завдання без необхідності використання додаткових програм.

Розширюваність: Blender підтримує розширення, які дозволяють розширити його функціональність. Ви можете встановлювати розширення, які додають нові інструменти, ефекти, шаблони та інші ресурси. Це дозволяє вам налаштовувати Blender під свої потреби та розширювати його можливості.

Активна спільнота: Blender має велику та активну спільноту користувачів, розробників і художників. Ви можете продовжити обговорення, знайти підтримку, відповіді на свої питання та навіть знайти навчальні матеріали та ресурси для вдосконалення своїх навичок у використанні Blender.

Розширені можливості моделювання: Blender надає розробникам широкий спектр інструментів для створення 3D-моделей [12]. Він підтримує різні методи моделювання, включаючи полігонове, скульптурне, реберне та поверхневе

		Надворний М.Ю			ПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

моделювання. Це дозволяє створювати складні форми, деталізовані текстури та інші складні елементи для ваших проєктів.

Анімація та рендеринг: Blender є потужним інструментом для створення анімації. Він має вбудовану систему анімації, яка дозволяє створювати рухи об'єктів, персонажів та інших елементів. Крім того, Blender має вбудовану систему рендерингу, яка дозволяє створювати фотореалістичні зображення та анімацію.

Симуляція та фізичне моделювання: Blender надає інструменти для симуляції фізики, такі як симуляція рідини, тканин, диму та інших фізичних ефектів. Це дозволяє створювати реалістичні сцени з фізичною взаємодією між об'єктами.

Візуалізація даних: Blender може бути використаний для візуалізації даних, таких як графіки, діаграми та інші візуальні представлення. Це корисний інструмент для вивчення та аналізу даних у привабливій та зрозумілій формі.



Рис. 1.6. Blender 3.5

Мова розробки. Опис переваг C#:

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Простота вивчення: C# має простий синтаксис та логічну структуру, що робить його досить легким для вивчення, особливо для початківців у програмуванні [5]. Він має багато спільного з іншими мовами програмування, такими як C++ та Java, тому знання C# також може бути корисним при роботі з цими мовами [3].

Об'єктно-орієнтованість: C# підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам структурувати код у вигляді класів та об'єктів. Це допомагає забезпечити більшу модульність, повторне використання коду та полегшує розширення програми у майбутньому [1].

Платформа .NET: C# є однією з основних мов програмування для платформи розробки програмного забезпечення .NET, яка надає розширені можливості для створення різноманітних програмних продуктів. .NET включає багато бібліотек і фреймворків, які спрощують розробку додатків і забезпечують широкий функціонал.

Мультиплатформеність: C# може бути використаний для розробки додатків на різних платформах, включаючи Windows, macOS і Linux. Це означає, що ви можете створювати програми, які працюють на різних операційних системах, з використанням однієї мови програмування.

Висновки до розділу 1

Проведений аналіз предметної області дозволив визначити основні аспекти гри у жанрі FPS. Огляд наявних аналогів та виявлення протиріч між наявними можливостями та потребами показав, що головними функціями майбутньої системи має бути: сучасний фізичний рушій, наявність зручного GUI, можливість вільно пересуватись гравцем ігровим світом, взаємодія із дрібними ігровими елементами, звукові ефекти. Наявність підтримки користувацьких модифікацій, економічна складова та мультиплеєрний режим є також бажаними функціями для таких програмних систем.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

Визначившись з оптимальною архітектурою, обрано необхідний та достатній набір програмних продуктів для реалізації поставлених задач: рушій, 3D-редактор.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ГРИ FPS

2.1. Визначення варіантів використання та структури системи.

Система гри у жанрі FPS створюється з метою максимального поглиблення гравця у відтворення процесу знищення ворогів у віртуальному світі.

Вимоги користувачів.

Користувачі мають доступ до наступних функцій:

1. Старт гри;
2. Пауза гри
3. Знищення ворогів;
4. Підбирання зброї, аптечки та патронів;
5. Зміна зброї;
6. Прицілювання через зброю;
7. Стрибок та подвійний стрибок;
8. Пришвидшення бігу;
9. Перехід на наступний рівень
10. Контрольна точка

Функціональні вимоги:

1. Гемплей: користувач матиме можливість знищувати ворогів від першої особи, підбирати зброю, патрони та аптечку, змінювати зброю, прицілюватися та робити стрибки.
2. Не потрібно перебувати онлайн: користувач може грати без підключення до Інтернету в довільний момент часу.

Нефункціональні вимоги:

1. Сприйняття:
 - Час, потрібний для навчання роботи з програмою для звичайних користувачів — 40 хвилин, для досвідчених — 15 хвилин.
 - Час відповіді системи для звичайних запитів не повинен перевищувати 0.016 секунди, а для більш складних запитів — 0.03 секунди.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

- Інтерфейс симулятора повинен бути інтуїтивно зручним для користувача та не вимагати від нього додаткової підготовки.
- Надійність:
- Доступність - час, потрібний для обслуговування системи не повинен перевищувати 10 % від загального часу роботи.
- Середній час безперервної роботи — 6 годин.
- Максимальна норма помилок та дефектів в роботі програми - 1 помилка на 1000 запитів користувача.

2. Продуктивність. Гра повинна підтримувати безперебійну роботу користувача.

3. Можливість експлуатації:

- Масштабування – після оновлення все повинно працювати без збоїв, не мають виникати жодні помилки при розширенні гри.
- Оновлення версій – оновлення версій повинно здійснюватися за допомогою патчу.

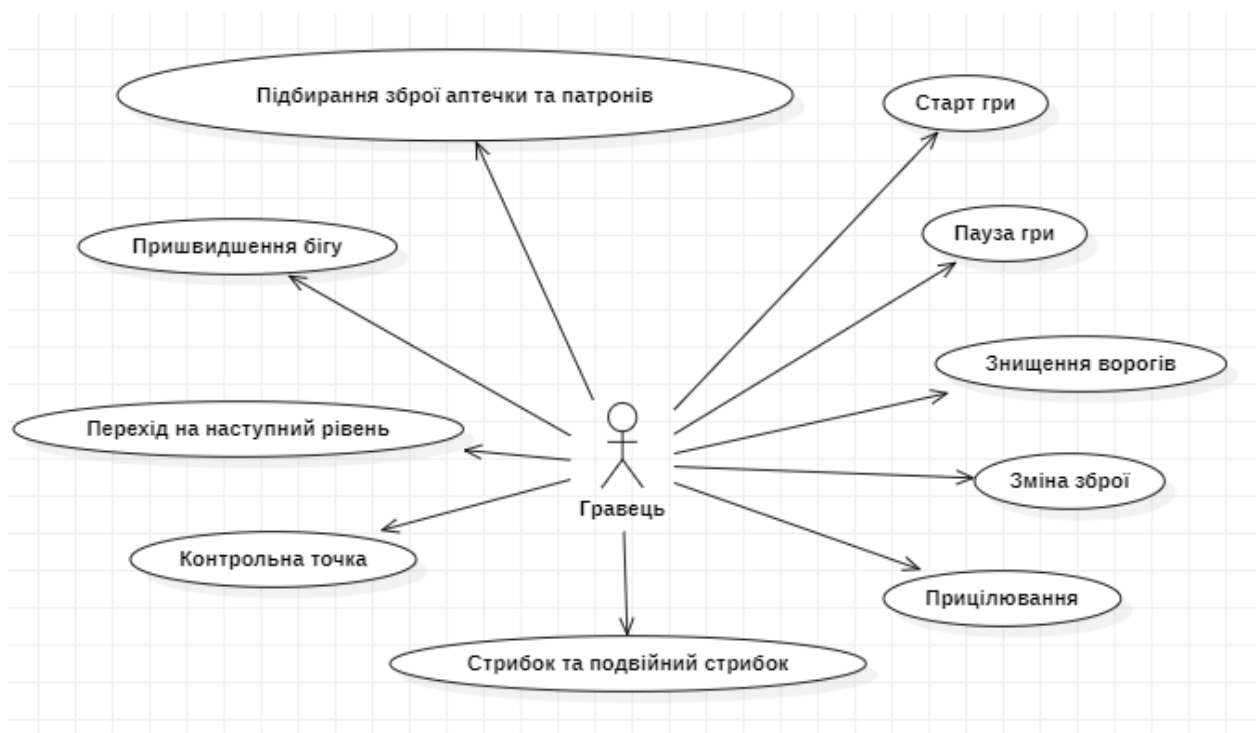


Рис.2.1. Варіанти використання

[illegible]

В даній діаграмі можна виділити класи, які спроектовано, використовуючи шаблон “Одинак”.

Гарантує, що у класу є тільки один екземпляр, і надає до нього глобальну точку доступу [10]. Важливо не тільки те, що можна користуватися саме екземпляром класу, так як при цьому в багатьох випадках стає доступною більш широка функціональність. Наприклад, до описаних компонентів класу можна звертатися через інтерфейс, якщо така можливість підтримується мовою [8].

- Якщо використовується існуюча об'єктно-орієнтована бібліотека;
- Якщо є шанси, що один об'єкт колись перетвориться в кілька;
- Якщо інтерфейс об'єкта (наприклад, ігрового світу) занадто складний, і не засмічувати основний простір імен великою кількістю функцій;
- Якщо, в залежності від яких-небудь умов і налаштувань, створюється декількох об'єктів. Наприклад, в залежності від того, ведеться лог чи ні,

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

створюється або реальний об'єкт, який пише в файл, або «заглушка», нічого не робить [6].

Розглянемо класи та за що вони відповідають:

На рис зображено діаграму класів, що відповідає за дії гравця у грі

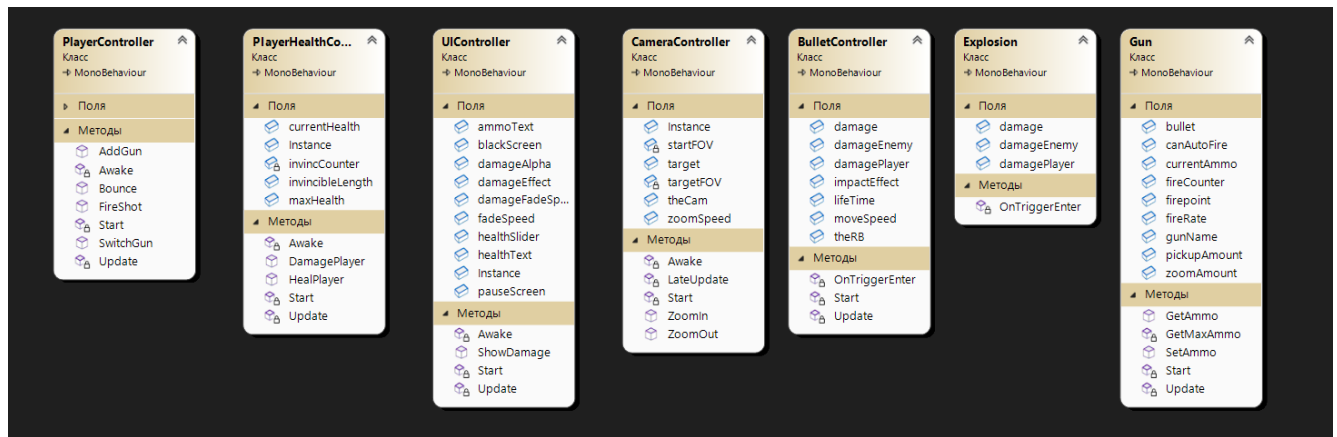


Рис. 2.3. Діаграма класів гравця

Клас PlayerController відповідає за керуванням руху, взаємодією зі зброєю та обробкою введення гравця в ігровому світі :

- AddGun () – додає нову зброю до списку доступної зброї. Перевіряє, чи зброя є доступною для розблокування, додає її до списку allGuns, видаляє зі списку unlockableGuns та оновлює поточну зброю;
- Awake () – викликається під час завантаження сцени. Встановлює статичний екземпляр цього класу (Instance), щоб мати можливість отримати доступ до нього з інших класів;
- Bounce () – викликається для виконання відскоку гравця;
- FireShot () – викликається для виконання пострілу.
- Update () – викликається на кожному кадрі гри. Отримує введення гравця, керує рухом гравця, обертанням камери, вистрілами, переключенням зброї та іншими діями гравця;
- LoadData () – використовується для завантаження даних гравця;
- SaveData () - використовується для збереження даних гравця;

Клас PlayerHealthController відповідає за керування здоров'я гравця:

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		30

- Awake () – викликається при ініціалізації об'єкта. Встановлює посилання на поточний об'єкт, щоб мати можливість звертатися до нього з інших класів;
- DamagePlayer () – викликається для завдання пошкоджень гравцеві;
- HealPlayer () – викликається для відновлення здоров'я гравця;
- Start () – викликається перед першим кадром оновлення. Встановлює початкове значення поточного здоров'я гравця (currentHealth) на максимальне значення (maxHealth).
- Update () – викликається на кожному кадрі гри.
- SetCurrentHealthPoints () - відповідає за встановлення поточного рівня здоров'я гравця.

Клас UIController відповідає за управління графічним інтерфейсом користувача гри :

- DamagePlayer () – додає нову зброю до списку доступних зброїв. Перевіряє, чи зброя є доступною для розблокування, додає її до списку allGuns, видаляє зі списку unlockableGuns та оновлює поточну зброю;
- Awake () – викликається під час завантаження сцени. Встановлює статичний екземпляр цього класу (Instance), щоб мати можливість отримати доступ до нього з інших класів;
- HealPlayer () – викликається для виконання відскоку гравця;
- Update () – викликається для виконання пострілу.

Клас CameraController відповідає за управління камери в грі:

- LateUpdate () – викликається один раз на кожен кадр після оновлення всіх інших об'єктів. Встановлює позицію і обертання камери (transform.position та transform.rotation) рівними позиції та обертанню цільового об'єкта (target). Також виконує згладжування поля зору

камери (theCam.fieldOfView) до цільового значення targetFOV з використанням Mathf.Lerp;

- Awake () – викликається під час завантаження сцени. Встановлює статичний екземпляр цього класу (Instance), щоб мати можливість отримати доступ до нього з інших класів;
- ZoomIn () – викликається для зумування камери. Встановлює цільове значення поля зору камери (targetFOV) на нове значення newZoom, яке вказується при виклику методу;
- ZoomOut () - викликається для віддалення камери. Встановлює цільове значення поля зору камери (targetFOV) на початкове значення startFOV.
- Update () – викликається для виконання пострілу.
- Start () - викликається перед першим кадром оновлення. Встановлює початкове значення поля зору (FOV) камери theCam на значення startFOV, яке встановлюється на початок запуску гри. Також ініціалізує змінну targetFOV значенням startFOV.

Клас BulletController відповідає за поведінку кулі в грі :

- OnTriggerEnter () – викликається, коли об'єкт стикається з коллайдером іншого об'єкта. В даному випадку, метод перевіряє, з якими об'єктами стикається куля;
- Update () – викликається один раз на кожен кадр оновлення. Встановлює швидкість руху кулі (theRB.velocity) в напрямку вперед (transform.forward) зі швидкістю moveSpeed. Зменшує час життя кулі (lifeTime) на величину часу, що пройшов з останнього кадру (Time.deltaTime). Якщо час життя кулі стає меншим або дорівнює нулю, то куля знищується (Destroy(gameObject)).

Клас Explosion відповідає за вибух після використання ракетниці в грі :

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

- OnTriggerEnter () – викликається, коли об'єкт стикається з колайдером іншого об'єкта. В даному випадку, метод перевіряє, з якими об'єктами стикається вибух;

Клас Gun відповідає за поведінку зброї в грі :

- Update () – викликається один раз за кожен кадр і використовується для оновлення стану зброї;
- GetAmmo () – викликається для отримання боєприпасів для зброї;
- GetMaxAmmo () – повертає максимальну кількість боєприпасів для певної зброї;
- SetAmmo () – встановлює поточну кількість боєприпасів для зброї.

На рис зображено діаграму класів, що відповідає за дії ворога у грі:

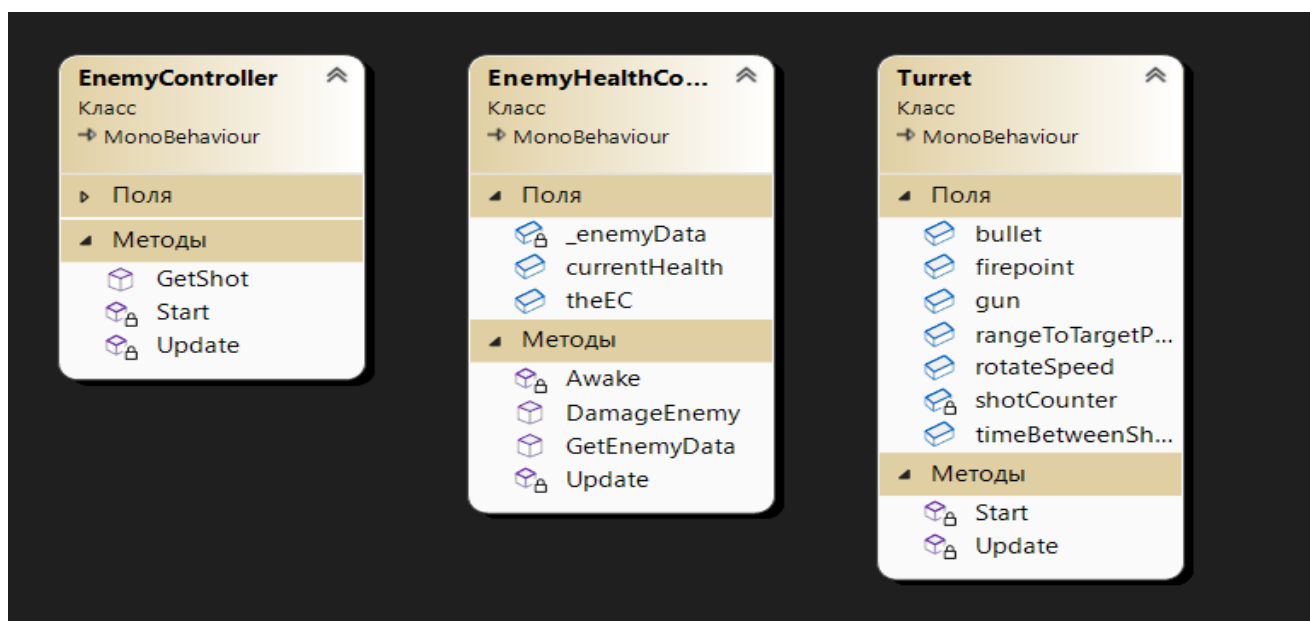


Рис. 2.4. Діаграма класів ворога

Клас EnemyController відповідає за контроль поведінки ворожого персонажа в грі:

- Update () – відповідає за оновлення стану ворожого персонажа;

- Start () – викликається під час старту гри. Встановлює початкові значення змінних, такі як startPoint, shootTimeCounter та shootWaitCounter;
- GetShot () – викликається, коли ворог отримує постріл. Встановлює змінну wasShot в значення true і вмикає режим переслідування (chasing = true);

Клас EnemyHealthController відповідає за контроль здоров'я ворожого персонажа в грі:

- GetEnemyData () – повертає об'єкт типу EnemyData, який містить дані про ворожого персонажа, включаючи інформацію про його стан і статистику;
- DamageEnemy () – викликається під час старту гри. Встановлює початкові значення змінних, такі як startPoint, shootTimeCounter та shootWaitCounter;
- GetShot () – викликається при завданні шкоди ворожому персонажу;

Клас Turret відповідає за контроль башти-ворога в грі:

- Start () – викликається при старті гри або при активації об'єкта. Встановлює початкові значення для змінних та лічильників;
- Update () – перевіряє відстань між баштою-ворогом та гравцем і випускає кулі з певною швидкістю;

На рис зображено діаграму класів, що відповідає за меню гри:

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		34

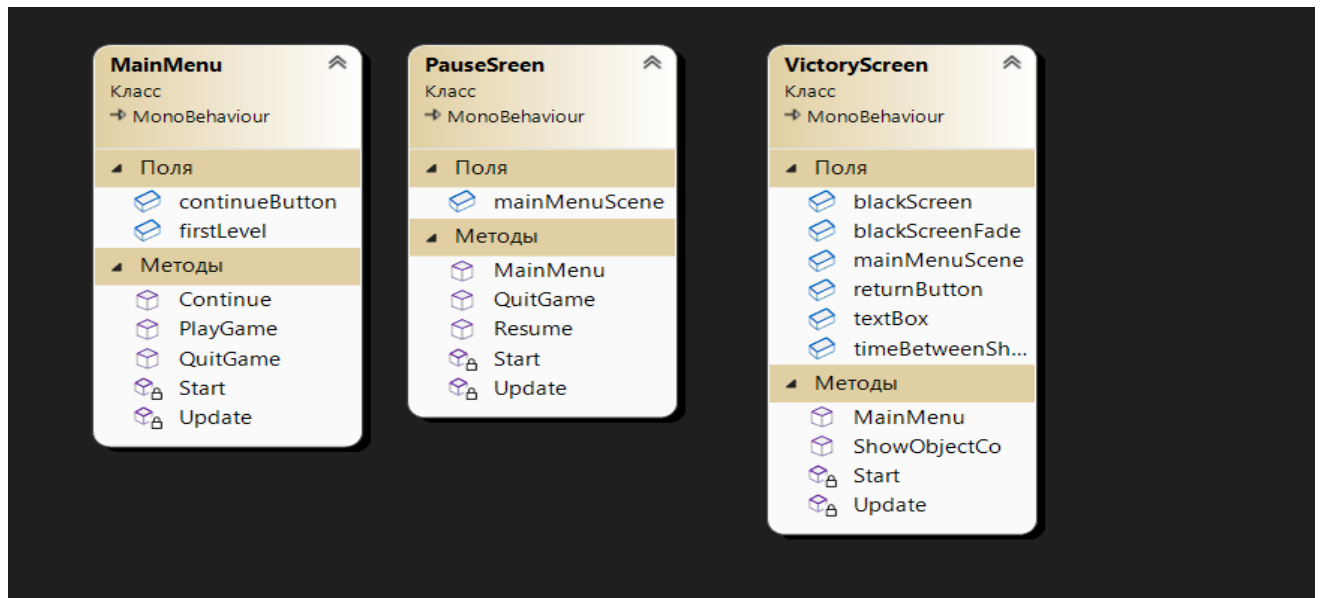


Рис. 2.5. Діаграма класів меню

Клас MainMenu відповідає за контроль головного меню гри:

- Continue () – викликається при натисканні кнопки "Продовжити". Завантажує сцену, яка відповідає збереженому значенню "CurrentLevel" у PlayerPrefs;
- PlayGame () – викликається при натисканні кнопки "Грати". Завантажує початкову сцену гри (firstLevel). Створюється запис у PlayerPrefs для "CurrentLevel" та "назва сцени + _cp" зі значеннями за замовчуванням (порожніми рядками);
- QuitGame () – викликається при натисканні кнопки "Вийти". Завершує виконання програми (гру);
- Start () - викликається при старті гри або при активації об'єкта. Він перевіряє, чи існує значення "CurrentLevel" у PlayerPrefs. Якщо значення порожнє або відсутнє, то вимикається кнопка "Продовжити" (continueButton).

Клас PauseScreen відповідає за контроль екрану паузи гри:

- Resume () – викликається при натисканні кнопки "Продовжити" на екрані паузи. Він викликає метод PauseUnpause() з об'єкта GameManager для продовження гри;

- MainMenu () – викликається при натисканні кнопки "Головне меню" на екрані паузи. Він завантажує сцену головного меню (mainMenuScene) за допомогою SceneManager і встановлює часовий масштаб Time.timeScale на значення 1 для відновлення нормального руху гри;
- QuitGame () – викликається при натисканні кнопки "Вийти" на екрані паузи. Він завершує виконання програми (гру);

Клас VictoryScreen відповідає за управління екраном перемоги у грі:

- Update () – змінює прозорість чорного екрану (blackScreen) на основі заданої швидкості зникнення (blackScreenFade);
- MainMenu () – викликається при натисканні кнопки "Головне меню" на екрані перемоги. Він завантажує сцену головного меню (mainMenuScene) за допомогою SceneManager;
- ShowObjectCo () - показує текстове поле (textBox) та кнопку повернення (returnButton) на екрані перемоги з заданим інтервалом часу (timeBetweenShowing). Використовується yield return new WaitForSeconds () для затримки виконання на певний час перед показом кожного об'єкта.

На рис зображено діаграму класів, що відповідає за звукові ефекти, батут та перехід між рівнями:

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		36

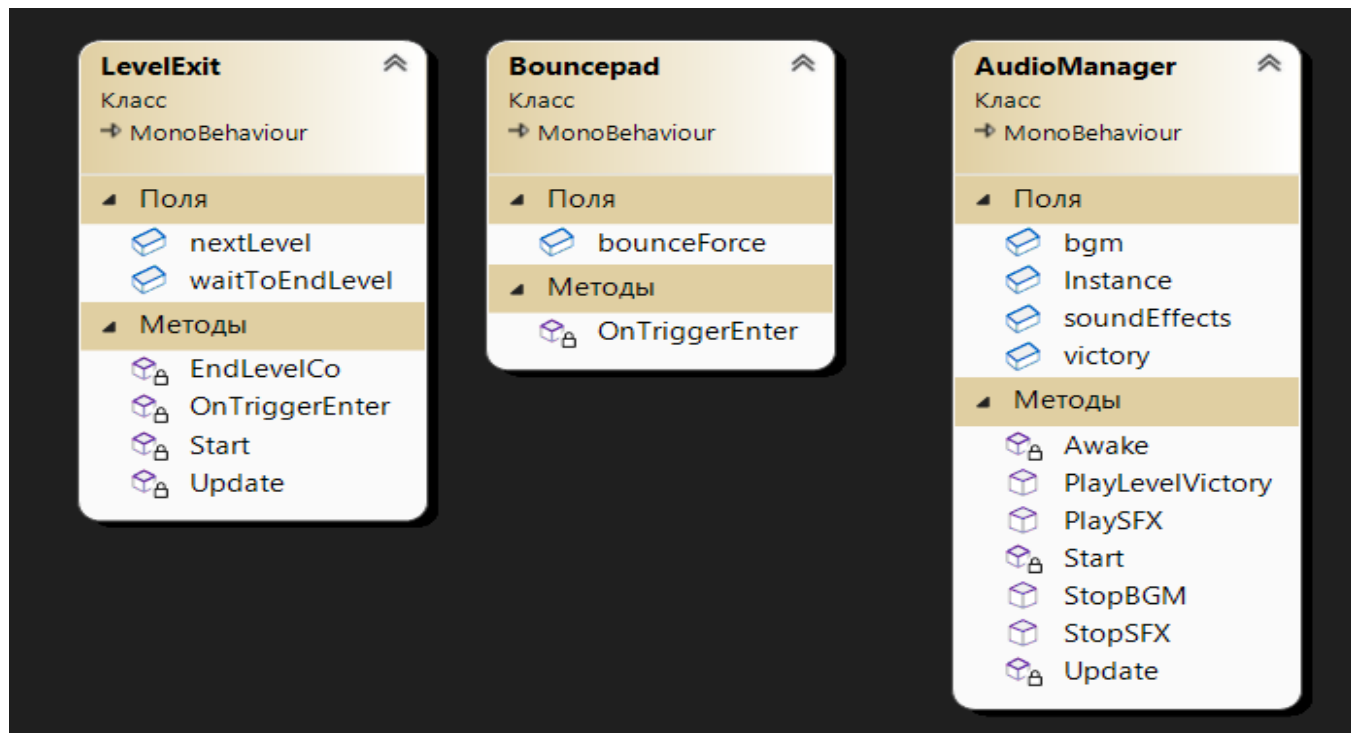


Рис. 2.6. Діаграма класів звукових ефектів, батута та переходу між рівнями

Клас LevelExit відповідає за перехід між рівнями в грі:

- EndLevelCo () – виконує завершення рівня;
- OnTriggerEnter () – викликається, коли об'єкт потрапляє в тригер, до якого прикріплений скрипт;

Клас Bouncepad відповідає за ефект відскоку гравця в грі при його взаємодії з об'єктом:

- OnTriggerEnter () – викликається метод Bounce в PlayerController.Instance для виконання відскоку гравця ;

Клас AudioManager відповідає за керування звуковими ефектами і фоновим музичним супроводом у грі :

- StopBGM () – Зупиняє відтворення фонової музики;
- Awake () – викликається під час завантаження сцени. Встановлює статичний екземпляр цього класу (Instance), щоб мати можливість отримати доступ до нього з інших класів;

- PlayLevelVictory () – Зупиняє відтворення фонової музики та починає відтворення звукового ефекту перемоги;
- PlaySFX () – Зупиняє відтворення звукового ефекту за певним номером sfxNumber і починає його відтворення;
- StopSFX () – Зупиняє відтворення звукового ефекту за певним номером sfxNumber;

На рис зображено діаграму класів, що відповідає за підбирання зброї, патронів та аптечки:

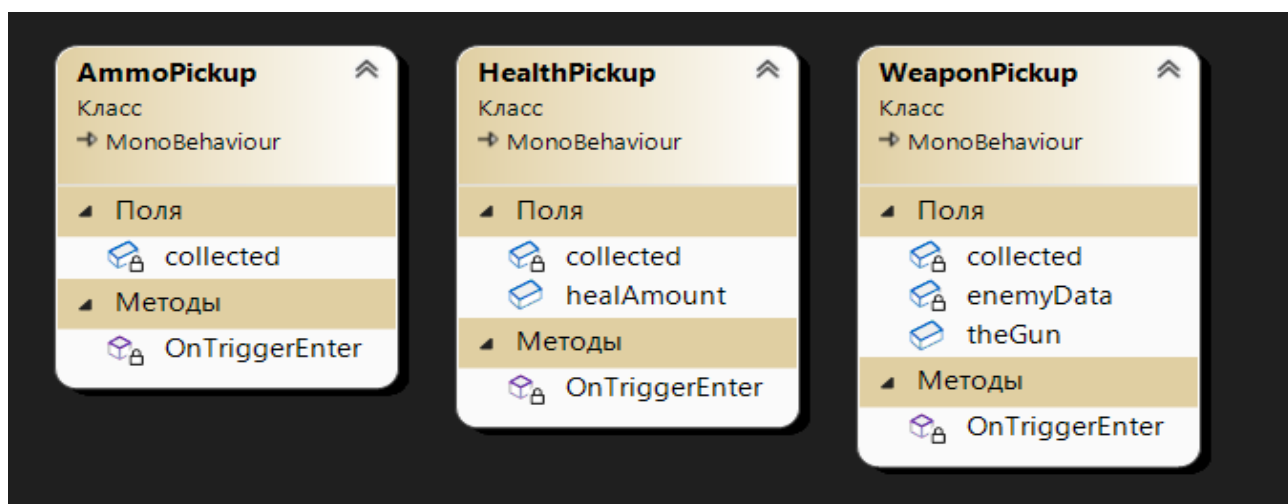


Рис. 2.7. Діаграма класів підбирання зброї, аптечки та патронів

Клас AmmoPickup відповідає за збирання патронів гравцем у грі:

- OnTriggerEnter () – додає патрони до активної зброї гравця, знищує об'єкт патронів, щоб вони не могли бути зібрані знову, і відтворює звуковий ефект для підтвердження збирання;
- OnDisable () - використовується для автоматичного додавання зброї до інвентаря гравця.

Клас HealthPickup відповідає за збирання аптечок гравцем у грі:

- OnTriggerEnter () – відновлює певну кількість здоров'я гравцю, знищує об'єкт аптечки, щоб вона не могла бути зібрана знову, і відтворює звуковий ефект для підтвердження збирання;

Клас WeaponPickup відповідає за збирання зброї гравцем:

- OnTriggerEnter () – надає гравцю нову зброю, знищує об'єкт зброї, щоб його не можна було зібрати знову, відтворює звуковий ефект для підтвердження збирання;

На рис зображено діаграму класів, що відповідає за збереження даних в контрольній точці:

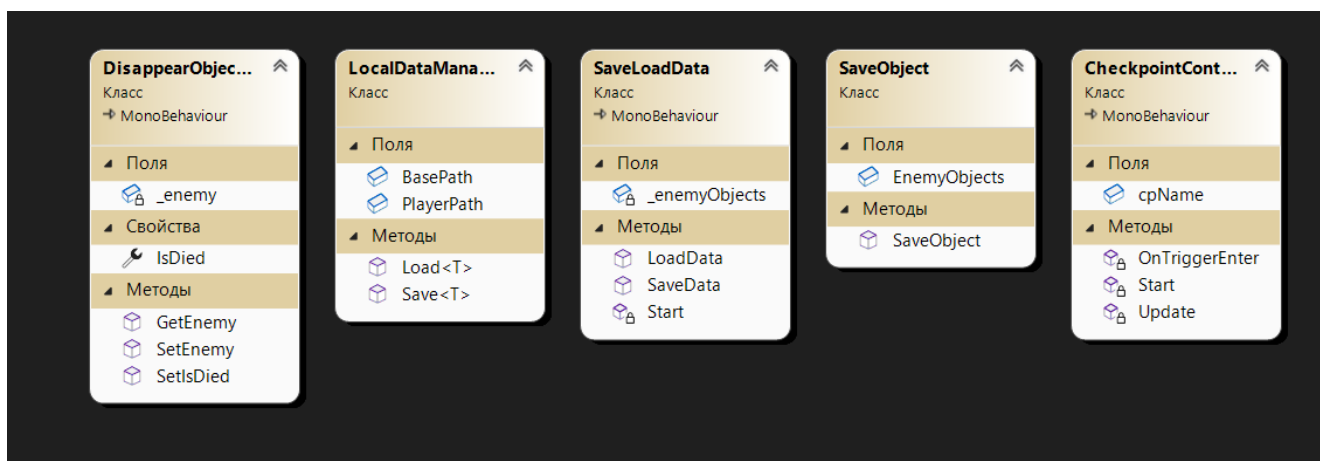


Рис. 2.8. Діаграма класів збереження даних в контрольній точці

Клас CheckpointController відповідає за керування контрольними точками у грі. Клас DisappearObjectData представляє дані про об'єкти, що зникають у грі. Він містить посилання на групу об'єктів, а також визначає, чи об'єкт вже зник. Клас LocalDataManager використовується для збереження та завантаження даних на локальному пристрої. Він містить методи для збереження та завантаження даних у форматі JSON. Клас SaveLoadData є основним класом, що керує збереженням та завантаженням даних гри. Клас SaveObject відповідає за створення об'єкта, який містить дані для збереження.

2.2 Проектування та реалізація алгоритмів роботи системи

Основним модулем системи є керування гравцем. Даний модуль складається з багатьох скриптів, що контролюють:

- рух гравця;
- стрибки гравця;
- стрільбу гравця;
- прицілювання гравця;
- підбирання зброї, аптечки, патронів;
- відображення здоров'я та кількості патронів гравця;
- звукові ефекти;
- збереження в контрольній точці.

При реалізації системи було найбільш точно реалізовано процес керування гравцем. Аналогічні дії виконуються і для реальних об'єктів.

На початку гри гравець з'являється на початковій точці з якої йому необхідно дістатися до кінцевої точки, де буде знаходитися телепорт на інший рівень, щоб цього досягти необхідно пройти або пробігти перешкоди. Щоб можна було краще пересуватися гравець може використовувати стрибок та подвійний стрибок також буде наявний батут, щоб перестрибувати високі перешкоди. На його шляху будуть зустрічатися вороги, гуманоїдні та башта, щоб через них пройти є 2 варіанти: знищити їх, або швидко пробігти. Для того, щоб знищити ворогів потрібна зброя, за замовчування гравець матиме пістолет. На його шляху будуть зустрічатися різні види зброї: лазерний пістолет, гвинтівка, ракетниця. Так як гравець буде знищувати ворогів він може прицілюватися для кращого знищення, при підбиранні зброї її можна змінювати. При перестрілці з ворогами можна зазнати пошкоджень тому важливо відновлювати здоров'я використовуючи аптечку, також для поповнення набоїв на шляху гравця будуть знаходитися патрони, підібравши їх збільшиться магазин патронів. Отже, на рис. 2.8 наведено діаграма активності гравця.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		40

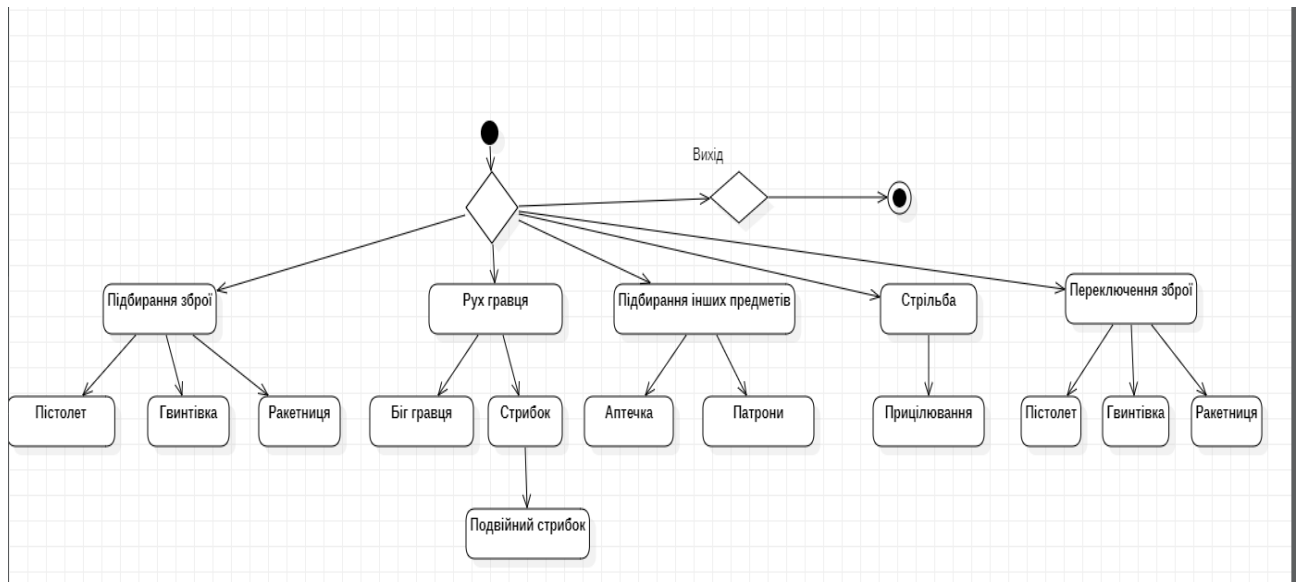


Рис. 2.9. Діаграма активності гравця

2.3. Реалізація функціоналу додатку

Додаток у жанрі FPS повинен мати модель поведінки, що є якнайбільш наближена до реалістичної, тому розглянемо реалізацію найважливіших класів додатку.

Розглянемо клас `PlayerController`, який відповідає за дії гравця:

Перевіряємо умови для активності руху персонажа: чи не є екран паузи або закінчення рівня активними, щоб уникнути руху персонажа коли гра зупинена або рівень завершений. Отримаємо входні дані гравця для руху та забезпечуємо рівномірний рух за допомогою функції `Normalize()`.

```
void Update()
{
    if (!UIController.Instance.pauseScreen.activeInHierarchy && !GameManager.Instance.levelEnding)
    {
        float yStore = moveInput.y;

        Vector3 vertMove = transform.forward * Input.GetAxis("Vertical");
        Vector3 horiMove = transform.right * Input.GetAxis("Horizontal");

        moveInput = horiMove + vertMove;
        moveInput.Normalize();
    }
}
```

Керування швидкістю руху персонажа.

```

if (Input.GetKey(KeyCode.LeftShift))
{
    moveInput = moveInput * runSpeed;
}
else
{
    moveInput = moveInput * moveSpeed;
}

```

Рух персонажа по вертикалі (вгору та вниз) та встановлюється значення гравітації, що дозволяє персонажу стояти на землі.

```

moveInput.y = yStore;

moveInput.y += Physics.gravity.y * gravityModifier * Time.deltaTime;

if (charCon.isGrounded)
{
    moveInput.y = Physics.gravity.y * gravityModifier * Time.deltaTime;
}

```

Перевіряється можливість стрибка та подвійного стрибка персонажа

```

canJump = Physics.OverlapSphere(groundCheckPoint.position, .25f, whatIsGround).Length > 0;

if (Input.GetKeyDown(KeyCode.Space) && canJump)
{
    moveInput.y = jumpPower;

    canDoubleJump = true;

    AudioManager.Instance.PlaySFX(8);
}

else if (canDoubleJump && Input.GetKeyDown(KeyCode.Space))
{
    moveInput.y = jumpPower;

    canDoubleJump = false;

    AudioManager.Instance.PlaySFX(8);
}

```

Переміщення персонажа, отримання вхідних даних миші.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (bounce)
{
    bounce = false;
    moveInput.y = bounceAmount;

    canDoubleJump = true;
}

charCon.Move(moveInput * Time.deltaTime);

Vector2 mouseInput = new Vector2(Input.GetAxisRaw("Mouse X"), Input.GetAxisRaw("Mouse Y")) * mouseSensitivity;

if (invertX)
{
    mouseInput.x = - mouseInput.x;
}

if (invertY)
{
    mouseInput.y = - mouseInput.y;
}

```

Обертання персонажа, камери.

```

transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles.x, transform.rotation.eulerAngles.y + mouseInput.x, transform.rotation.eulerAngles.z);
camTrans.rotation = Quaternion.Euler(camTrans.rotation.eulerAngles + new Vector3(-mouseInput.y, 0f, 0f));

if (camTrans.rotation.eulerAngles.x > maxViewAngle && camTrans.rotation.eulerAngles.x < 180f)
{
    camTrans.rotation = Quaternion.Euler(maxViewAngle, camTrans.rotation.eulerAngles.y, camTrans.rotation.eulerAngles.z);
}
else if (camTrans.rotation.eulerAngles.x > 180f && camTrans.rotation.eulerAngles.x < 360f - maxViewAngle)
{
    camTrans.rotation = Quaternion.Euler(-maxViewAngle, camTrans.rotation.eulerAngles.y, camTrans.rotation.eulerAngles.z);
}

muzzleFlash.SetActive(false);

```

Керування вистрілами зі зброї гравця на основі вхідних даних гравця та стану зброї.

```

if (Input.GetMouseButtonDown(0) && activeGun.fireCounter <= 0)
{
    RaycastHit hit;
    if (Physics.Raycast(camTrans.position, camTrans.forward, out hit, 50f))
    {
        if (Vector3.Distance(camTrans.position, hit.point) > 2f)
        {
            firePoint.LookAt(hit.point);
        }
        else
        {
            firePoint.LookAt(camTrans.position + (camTrans.forward * 30f));
        }
    }

    FireShot();
}

```

Автоматичний вистріл зі зброї, переключення зброї, збільшення та зменшення масштабу прицілювання.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

```

if (Input.GetMouseButton(0) && activeGun.canAutoFire)
{
    if (activeGun.fireCounter <= 0)
    {
        FireShot();
    }
}

if (Input.GetKeyDown(KeyCode.Tab))
{
    SwitchGun();
}

if (Input.GetMouseButtonDown(1))
{
    CameraController.Instance.ZoomIn(activeGun.zoomAmount);
}

if (Input.GetMouseButton(1))
{
    gunHolder.position = Vector3.MoveTowards(gunHolder.position, adsPoint.position, adsSpeed * Time.deltaTime);
} else
{
    gunHolder.localPosition = Vector3.MoveTowards(gunHolder.localPosition, gunStartPos, adsSpeed * Time.deltaTime);
}

if (Input.GetMouseButtonUp(1))
{
    CameraController.Instance.ZoomOut();
}

```

Метод, що відповідає за вистріл з активної зброї гравця.

```

2 references
public void FireShot()
{
    if (activeGun.currentAmmo > 0)
    {
        activeGun.currentAmmo--;

        Instantiate(activeGun.bullet, firePoint.position, firePoint.rotation);

        activeGun.fireCounter = activeGun.fireRate;

        UIController.Instance.ammoText.text = "Патрони: " + activeGun.currentAmmo;

        muzzleFlash.SetActive(true);
    }
}

```

Метод, що відповідає за зміну активної зброї гравця.

```

public void SwitchGun()
{
    activeGun.gameObject.SetActive(false);

    currentGun++;

    if (currentGun >= allGuns.Count)
    {
        currentGun = 0;
    }

    activeGun = allGuns[currentGun];
    activeGun.gameObject.SetActive(true);

    UIController.Instance.ammoText.text = "Патрони: " + activeGun.currentAmmo;

    firePoint.position = activeGun.firepoint.position;
}

```

Метод, який додає нову зброю до гравця.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public void AddGun(string gunToAdd)
{
    bool gunUnlocked = false;

    if (unlockableGuns.Count > 0)
    {
        for (int i = 0; i < unlockableGuns.Count; i++)
        {
            if (unlockableGuns[i].gunName == gunToAdd)
            {
                gunUnlocked = true;

                allGuns.Add(unlockableGuns[i]);

                unlockableGuns.RemoveAt(i);

                i = unlockableGuns.Count;
            }
        }
    }

    if (gunUnlocked)
    {
        currentGun = allGuns.Count - 2;
        SwitchGun();
    }
}

```

Завантаження та збереження даних гравця використовуючи PlayerPrefs.

```

public void LoadData()
{
    if (PlayerPrefs.HasKey("PlayerHealth"))
    {
        playerHealth.SetCurrentHealthPoints(PlayerPrefs.GetInt("PlayerHealth"));
    }

    foreach (var gun in allGuns)
    {
        if (PlayerPrefs.HasKey($"{gun.name}"))
        {
            gun.SetAmmo(PlayerPrefs.GetInt($"{gun.name}"));
        }
    }
}

4 references
public void SaveData()
{
    PlayerPrefs.SetInt("PlayerHealth", playerHealth.currentHealth);
    foreach (var gun in allGuns)
    {
        PlayerPrefs.SetInt($"{gun.name}", gun.currentAmmo);
    }
}

```

Висновки до розділу 2

Система симуляції реалізує функціонал FPS та моделювання реальних подій, що зв'язані з керуванням гравця, знищенням ворогів. Симуляція забезпечується наборами скриптів.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

Було розглянуто алгоритми основних процесів системи та проаналізовано взаємодію класів системи під час процесів гри.

В останньому пункті наводиться детальний розгляд реалізації дій гравця.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ СИМУЛЯЦІЇ

3.1 Порядок встановлення та налаштування системи.

Система стрільби у жанрі FPS це додаток, що було створено на рушії U3D. Після побудови релізної версії систему можна переносити на будь-який девайс, що відповідає цільовій операційній системі збірки, адже рушій має підтримку багатьох платформ. Наприклад, якщо збірку було виконано під Windows, гра комфортно запускатиметься на будь-якій системі, конфігурація якої задовольняє наступним мінімальним вимогам:

- процесор з тактовою частотою 2.4 ГГц або вище;
- 4 ОЗУ;
- 1 ГБ вільного місця на жорсткому диску;
- Графічний пристрій з підтримкою DirectX 9 (бажано DirectX 11).

Таким чином, розроблена система матиме достатню частоту кадрів (30 fps) на системах, конфігурація яких не старіше за 4-5 років та комфортну (60fps) на більшості сучасних систем.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з грою.

Після завантаження користувач опиняється у головному меню (рис. 3.1).

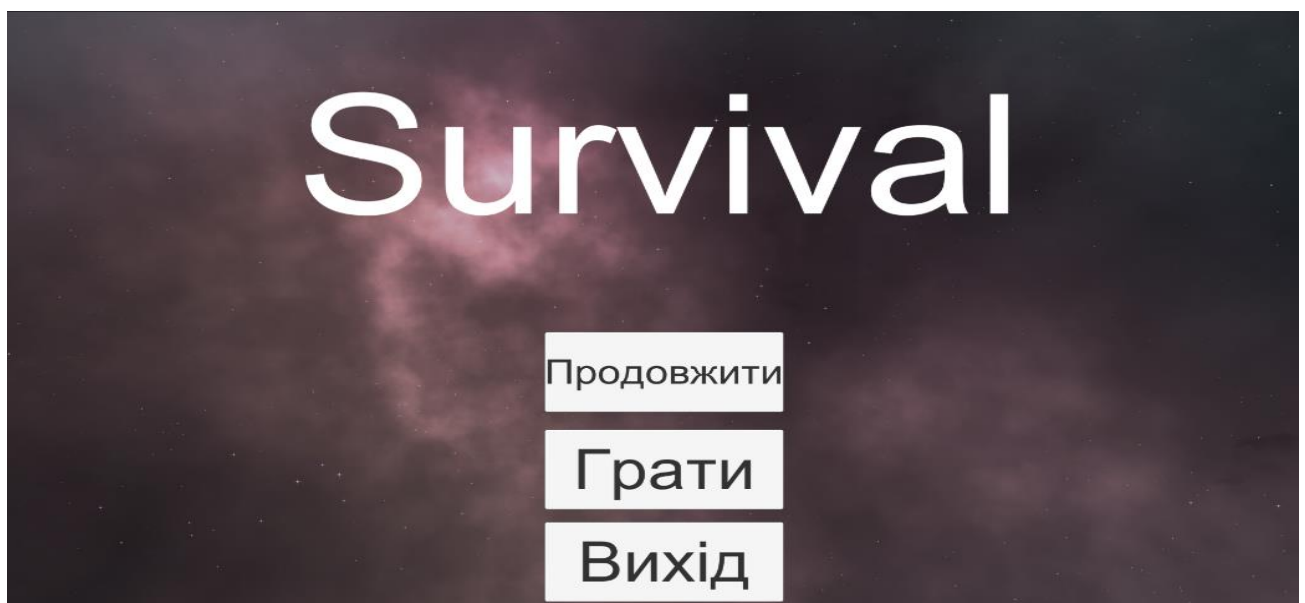


Рис. 3.1. Головне меню

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

Зображено три елементи керування, що представлені у вигляді кнопок: “Трати”, “Продовжити” та “Вихід”. При натисканні кнопки “Трати” починається гра, якщо користувач вийшов у головне меню чи вийшов з гри після проходження одного з рівнів він може натиснути на кнопку “Продовжити” і тоді гра продовжується. Після натискання кнопки “Вихід” користувач може вийти з гри.

Після запуску гри користувач з’являється на першому рівні, де він повинен пройти шлях до кінця усіх рівнів знищуючи своїх ворогів.

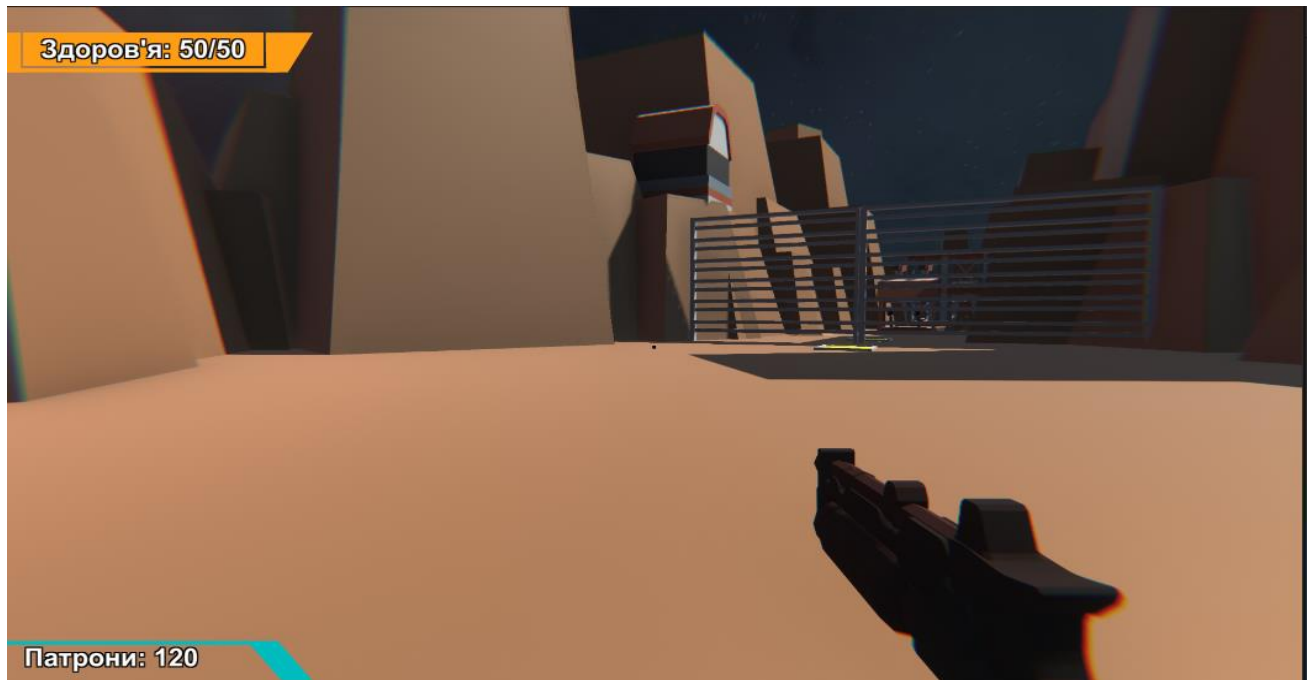


Рис. 3.2. Початок гри

Загальне керування. На екрані гравця з’являється сцена проходження гри, у лівому верхньому кутку відображається здоров’я гравця, у нижньому лівому кутку відображається кількість патронів, у кожної зброї своя кількість патронів.

Для того щоб пересуватися користувач натискає клавіші W S A D, де:

- W – рух вперед;
- S – рух назад;
- A – рух вліво;
- D – рух вправо.

Для бігу користувач може натиснути на клавішу Shift, яка збільшує швидкість гравця у 1.5 рази.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		48

Також користувач може робити стрибок та подвійний стрибок, що дозволяє краще уникати куль ворогів та краще пересуватися по різних локаціям.

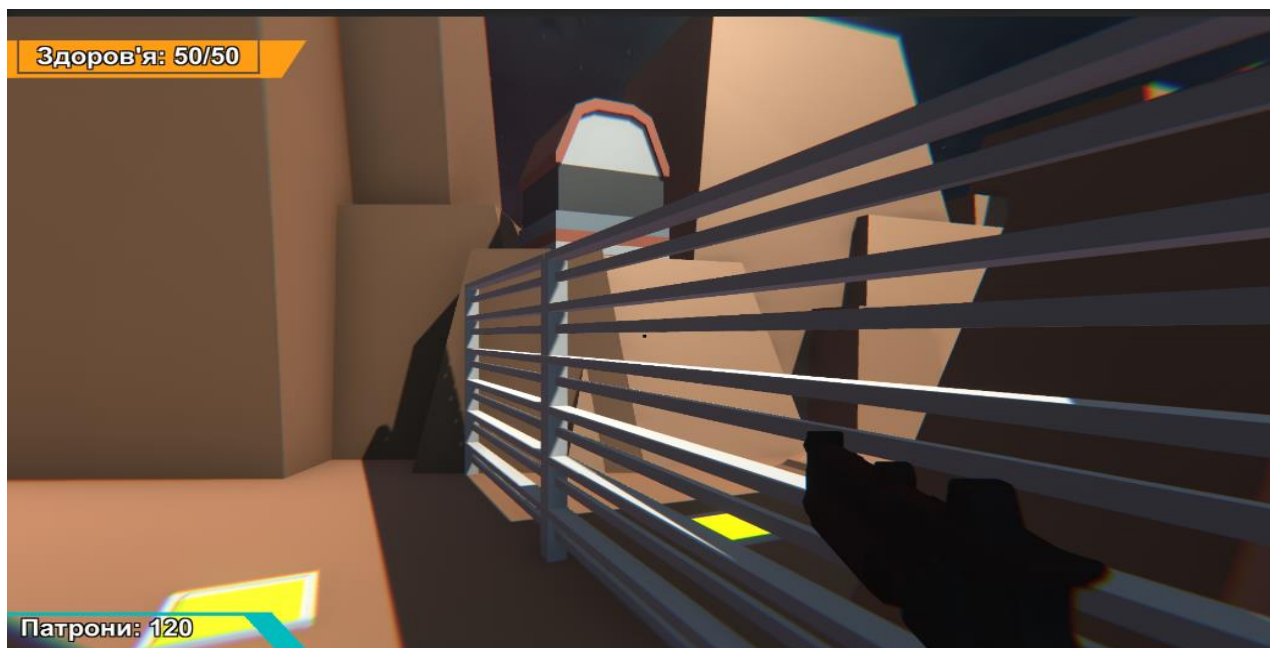


Рис. 3.3. Стрибки гравця

Для того, щоб скоріше проходити бар'єри користувач може використовувати батути, які дозволяють перепригнути перешкоду просто пригнувши або ставши на них.



Рис. 3.4. Взаємодія гравця з батутом

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

Знищення ворогів. Пересуваючись по локації гравець може натрапити на ворогів, тому їх потрібно знищити, для того щоб знищити ворога потрібно направити зброю на ворога та натиснути ЛКМ і стріляти до тих пір поки ворог не буде знищений та не буде заважати пересуватися по ігровому світу.

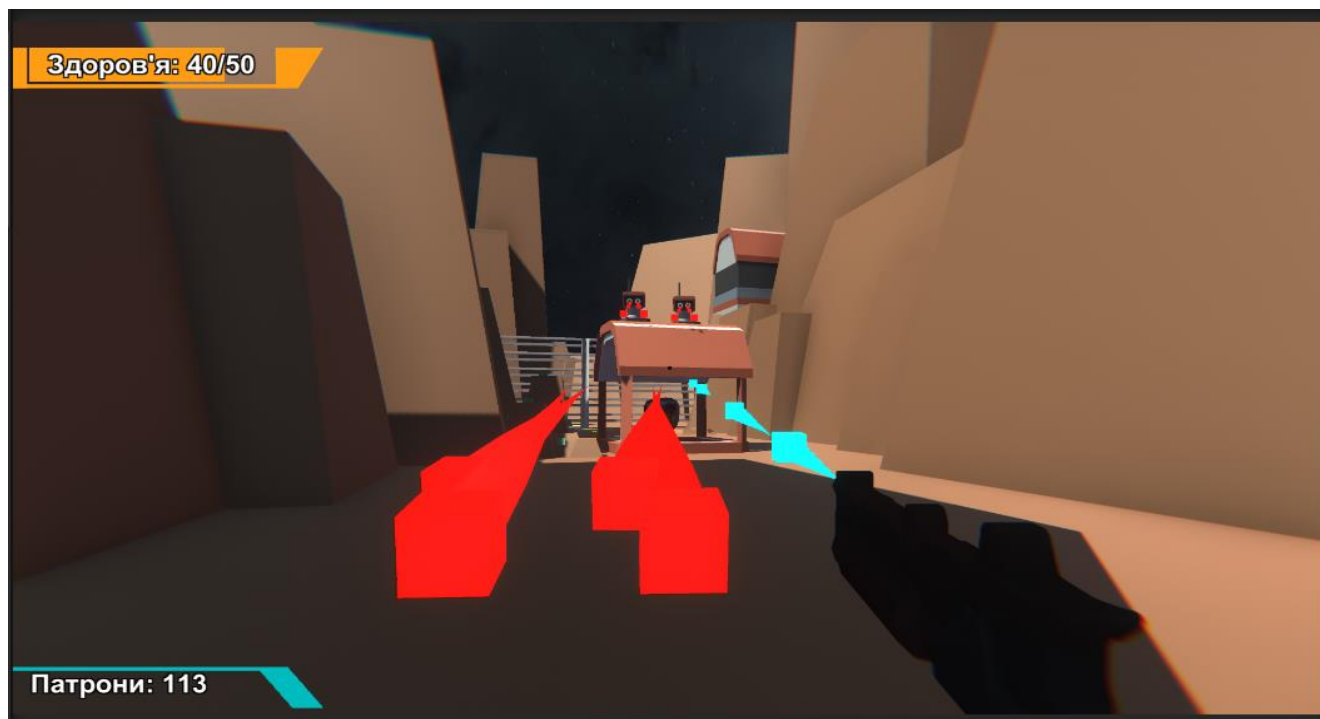


Рис. 3.5. Знищення ворогів

Є два типи ворогів: гуманоїдний та башта. Гуманоїдний ворог може рухатися по ігровому світу та стріляти по гравцеві. Башта стоїть на місці, але вона весь час обертається і якщо увійти в її зону ураження, то вона почне випускати пулі по гравцю безкінечно поки ворог її не знищить, або вийде з її зони ураження.

Для того щоб краще попадати по ворогові є масштабування прицілювання, яке дозволяє збільшити зону видимості перед собою і краще прицілитися.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

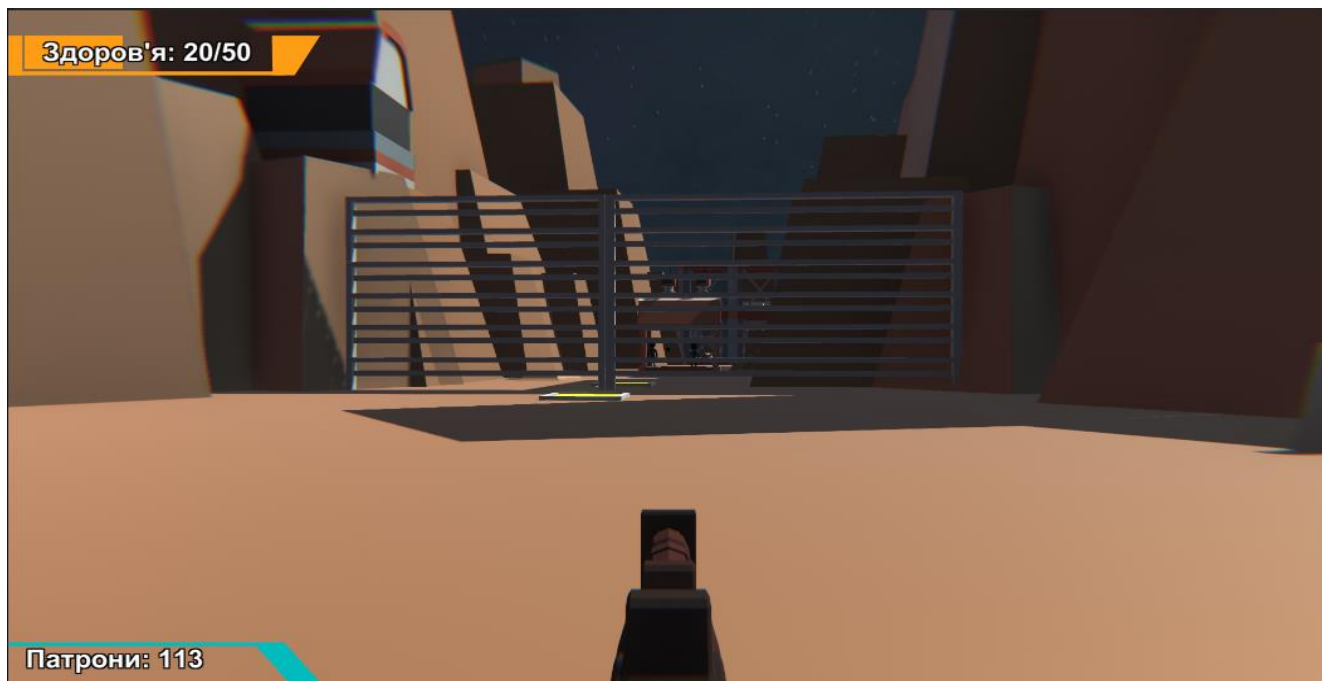


Рис. 3.6. Прицілювання гравця

Зброя. Гравець може підібрати зброю, яка знаходиться у різних місцях ігрового світу, всього є 4 типи зброї:

- Пістолет 1 типу(за замовчуванням)
- Пістолет 2 типу
- Гвинтівка
- Ракетниця

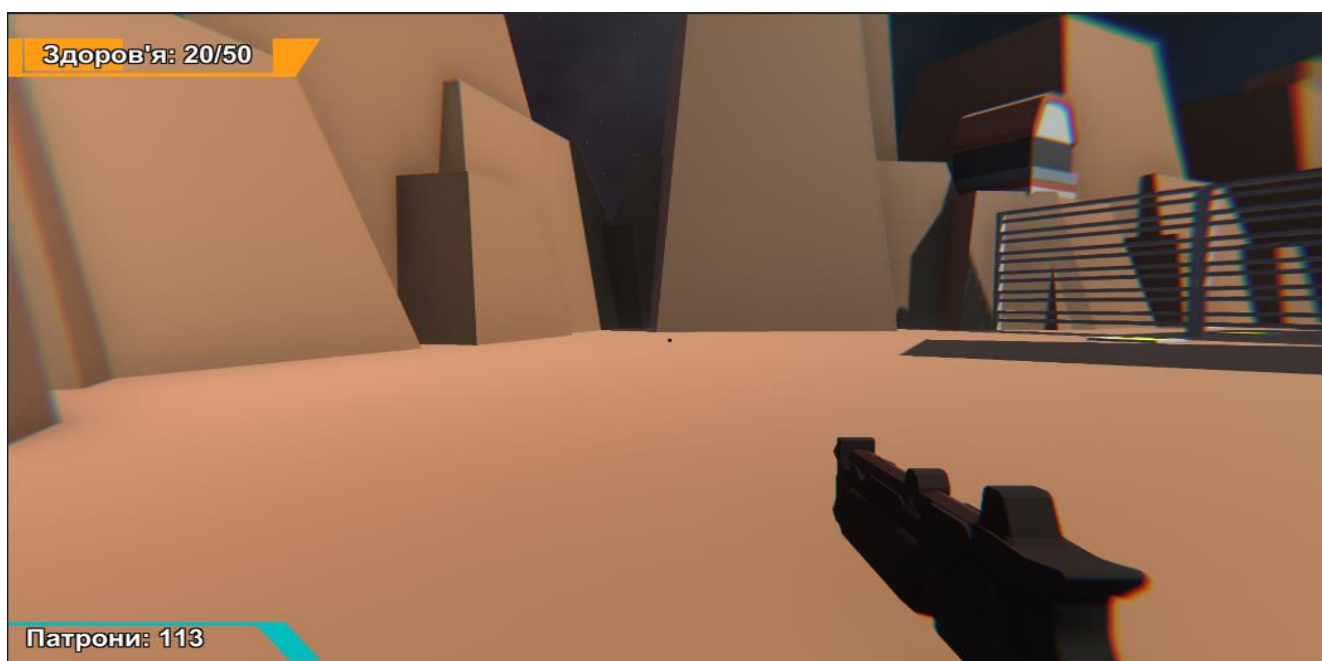


Рис. 3.7. Пістолет 1 типу

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

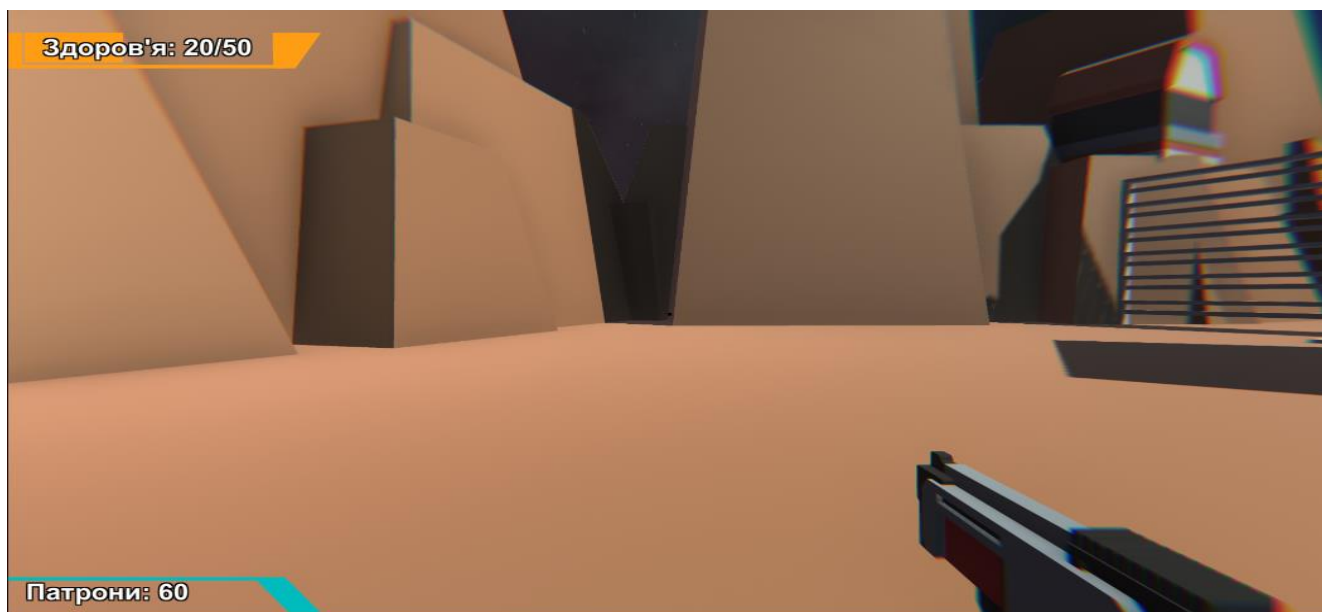


Рис. 3.8. Пістолет 2 типу



Рис. 3.9. Гвинтівка

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

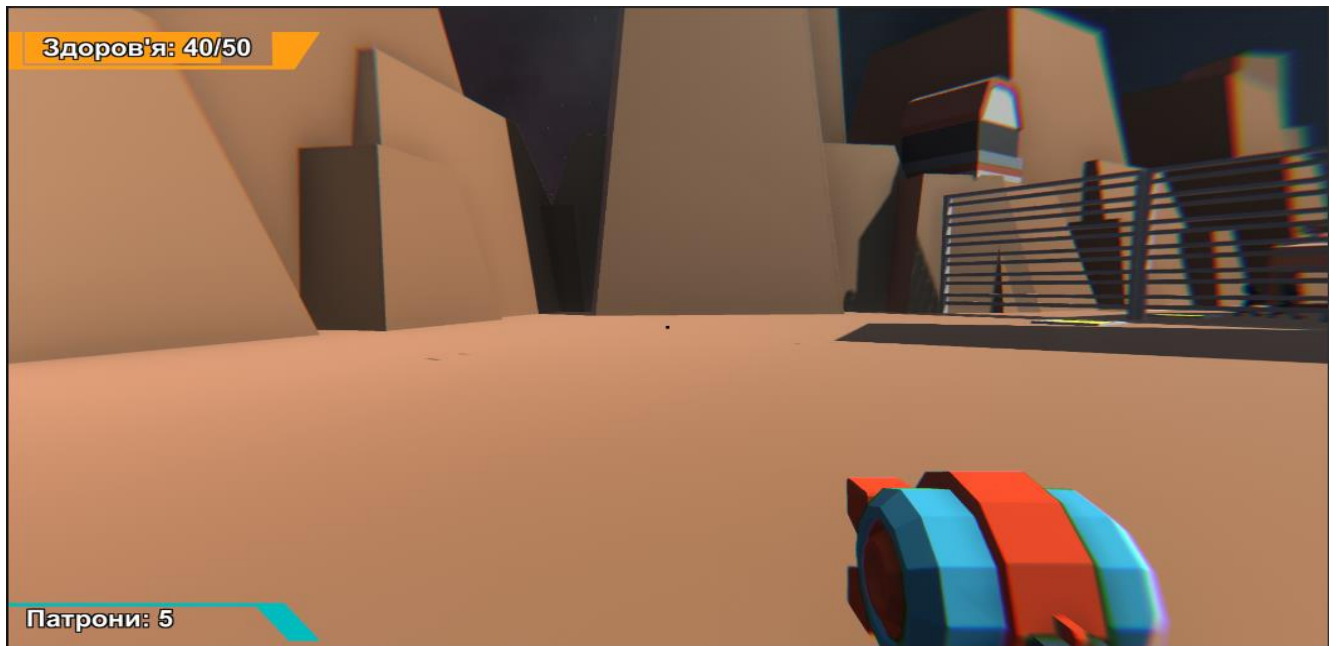


Рис. 3.10. Ракетниця

Аптечка. Для того, щоб гравець міг відновити здоров'я є аптечка, яка відновлює здоров'я на 15 пунктів із 50. Аптечку можна підібрати на локаціях ігрового світу.

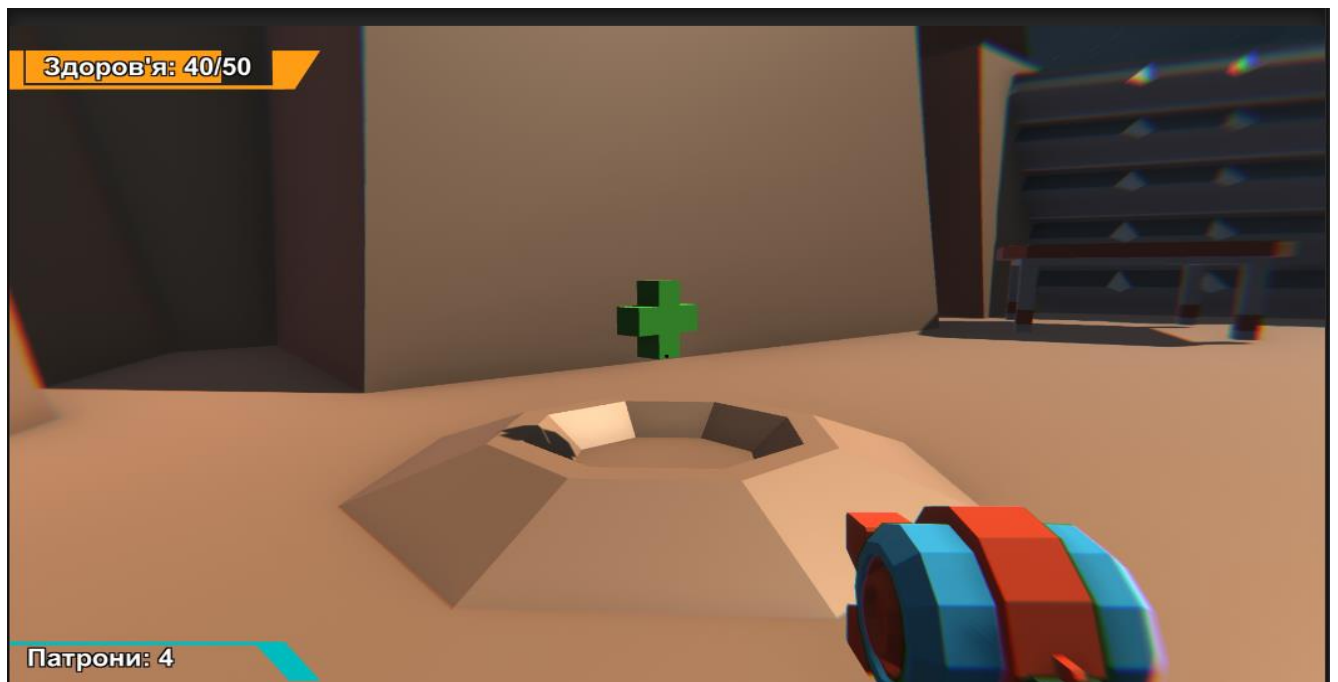


Рис. 3.11. Аптечка

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

Патрони. При стрільби патрони можуть досить швидко закінчуватися, тому потрібно їх відновлювати, для цього на локаціях ігрового світу є предмет патрону, який можна підібрати і збільшити кількість патронів.

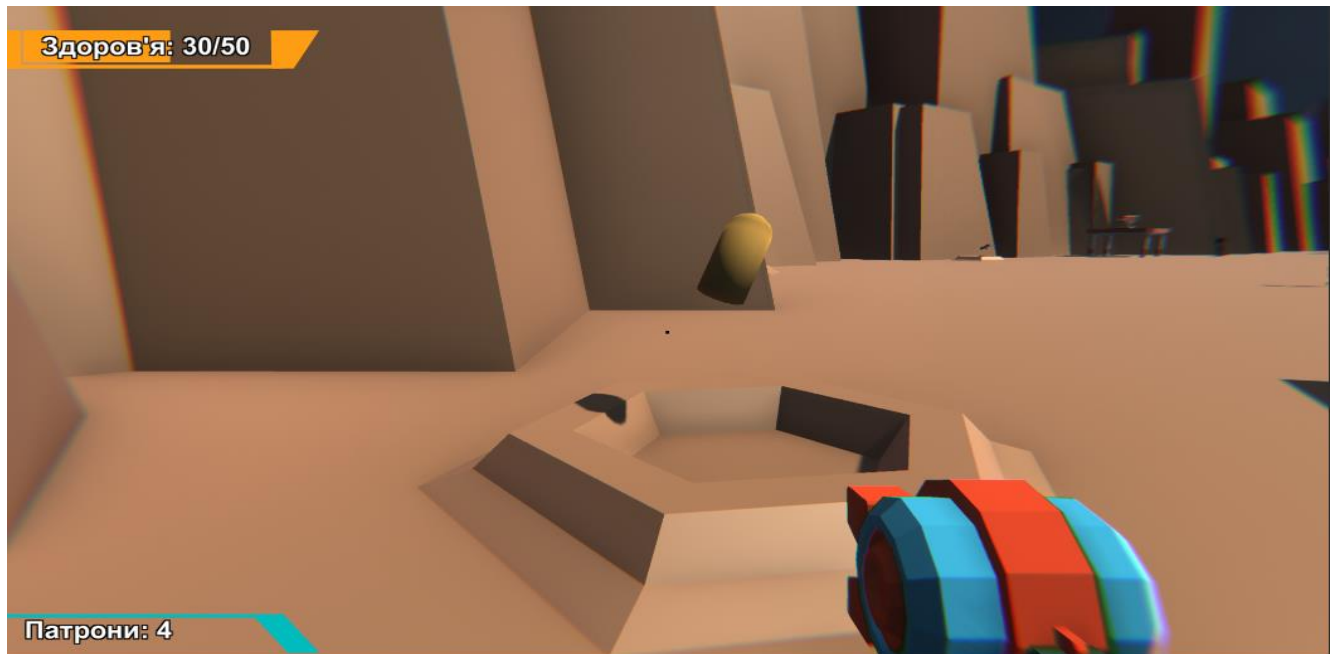


Рис. 3.12. Патрони

Перехід на інший рівень. Користувач має можливість перейти на інший рівень, де буде нова локація, яку потрібно пройти.

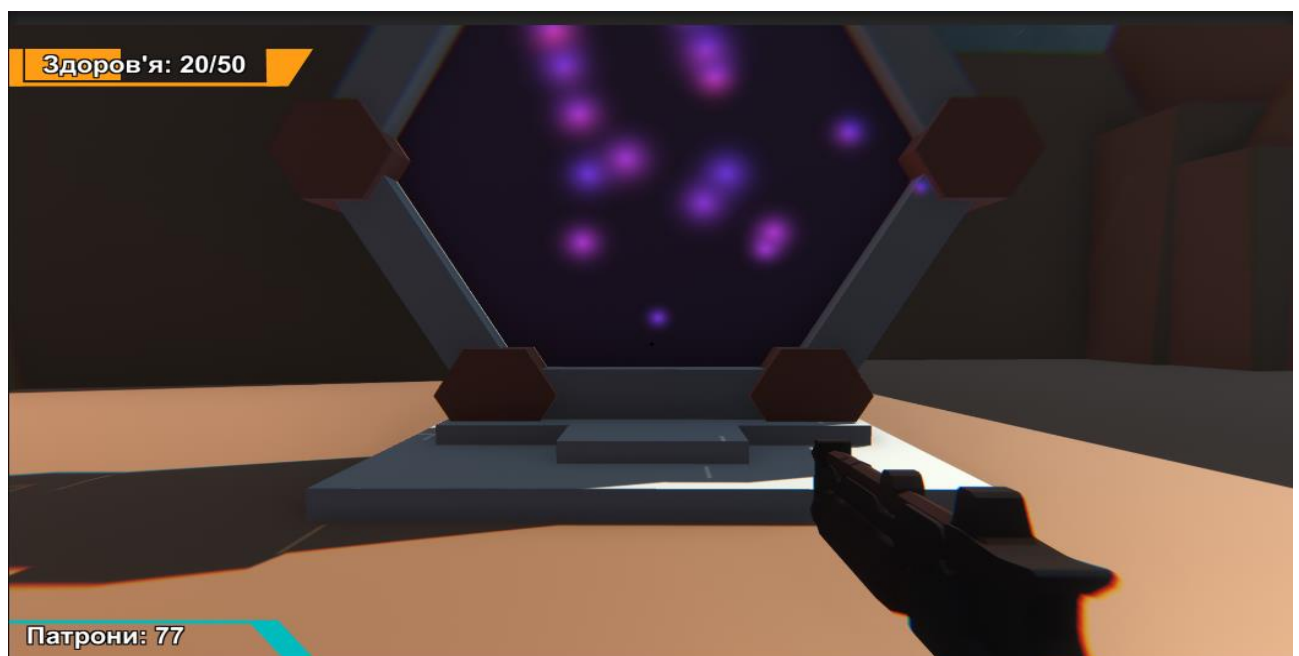


Рис. 3.13. Перехід між рівнями

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Тестування роботи системи

Перевірка на працездатність – невід’ємний процес під час розробки програмного коду системи [14]. Ігрове тестування в певних аспектах відрізняється від тестування інтернет-ресурсів, мобільних додатків та інших продуктів [15].

Тестування ігрових систем містить такі основні види:

- тестування функціональності;
- тестування контенту;
- тестування геймплею;
- користувацьке тестування.

Тестування функціональності. Нижче наведено тест-кейси по основним діям гравця.

Кнопка «Грати»:

Передумови: Гравець повинен знаходитись у головному меню;

Кроки: Натискання кнопки «Грати»;

Очікуваний результат: Перехід гравця на 1 рівень;

Фактичний результат: Перехід гравця на 1 рівень;

Кнопка «Продовжити»:

Передумови: Гравець повинен знаходитись у головному меню;

Кроки: Натискання кнопки «Продовжити»;

Очікуваний результат: Перехід гравця на будь-який із рівнів;

Фактичний результат: Перехід гравця на будь-який із рівнів;

Кнопка «Вихід»:

Передумови: Гравець повинен знаходитись у головному меню;

Кроки: Натискання кнопки «Вихід»;

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

Очікуваний результат: Вихід користувача з гри;

Фактичний результат: Вихід користувача з гри;

Тестування геймплею. Було проведено тестування геймплейних особливостей, включаючи основну – керування гравцем його рух, стрільба і тд. Під час тестування доступного функціоналу у нього не було виявлено жодних недоліків чи помилок у роботі.

Висновки до розділу 3

В даному розділі було наведено порядок встановлення та налаштування додатку та визначено мінімальну конфігурацію системи для комфортної гри.

Реалізована система симуляції містить інтерфейси головного меню, меню паузи та основний ігровий інтерфейс, робота та функції елементів якого були наведені. Також було проведено функціональне та геймплейне тестування додатку. Знайдені помилки було виправлено, система працює коректно та виконує поставлені задачі.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано гру у жанрі FPS для цього було вирішено наступні завдання:

1. Проведений аналіз предметної області дозволив визначити основні аспекти ігор в жанрі FPS. Огляд наявних аналогів та виявлення протиріч між наявними можливостями та потребами показав, що головними функціями майбутньої системи має бути: сучасний фізичний рушій, наявність зручного GUI, можливість вільно пересуватись гравцем ігровим світом, взаємодія із дрібними ігровими елементами, звукові ефекти. Наявність підтримки користувацьких модифікацій, економічна складова та мультиплеєрний режим є також бажаними функціями для таких програмних систем. Визначившись з оптимальною архітектурою Component-Based Architecture, обрано необхідний та достатній набір програмних продуктів для реалізації поставлених задач: рушій Unity3D, 3D-редактор Blender3d.

2. Було проаналізовано функціональні та нефункціональні вимоги до системи симуляції, для чого побудовано діаграму варіантів використання та на її основі розроблено діаграму класів для аналізу структури системи.

3. Було розглянуто алгоритми основних процесів системи та проаналізовано взаємодію класів системи під час процесів симуляції.

4. Було наведено детальний розгляд основного класу гравця.

5. Реалізована система симуляції містить інтерфейси головного меню, меню паузи та основний ігровий інтерфейс, робота та функції елементів якого були наведені.

6. Також було проведено функціональне та геймплейне тестування додатку. Знайдені помилки було виправлено, система працює коректно та виконує поставлені задачі.

Реалізований програмний комплекс готовий до використання. Система може бути використана для розваг.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

В подальшому дана система може бути розширена шляхом впровадження нових функціональностей та розширенням ігрового світу. Завдяки обраній архітектурі розширення системи відбуватиметься значно швидше.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		58

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Чистий код: Посібник з майстерності гнучкого програмного забезпечення", Роберт К. Мартін, Prentice Hall (2008) – 441 с.
2. Jason van Gumster "Blender For Dummies" – 640 с.
3. Andrew Trolelsen, Philip Japikse Pro C# 7: With .NET and .NET Core – 1372 с.
4. Білас С. "A Data-Driven Game Object System" / Скот Білас., 2013. – 251 с.
5. Документація C# [Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/dotnet/csharp/>
6. Фрімен Е. Паттерни проектування / Е. Фрімен, Е. Фрімен. – 2011. – 656 с.
7. Використання Singleton в Unity3D [Електронний ресурс] – Режим доступу до ресурсу: <https://habr.com/post/341830/>
8. Документація Unity [Електронний ресурс] – Режим доступу до ресурсу:
<https://forum.unity.com/>
9. Документація Unity [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.unity3d.com/>
10. "Component-Based Software Engineering: Putting the Pieces Together" by William C. Wake – 226с.
11. Торн А. Mastering Unity Scripting / Алан Торн. – Бірмінгем: Packt Publishing, 2015. – 380 с.
12. James Kronister Blender Basics – 153 с.
13. Heather Brown "Visual Studio Code Tips and Tricks" – 150 с.
14. Glenford J. Myers, Corey Sandler, Tom Badgett "The Art of Software Testing" – 256 с.
15. Alan Page, Ken Johnston, and Bj Rollison "How We Test Software at Microsoft" – 405 с.

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас PlayerController

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    public static PlayerController Instance;
    public PlayerHealthController playerHealth;
    public float moveSpeed;
    public float gravityModifier;
    public float jumpPower;
    public float runSpeed = 12;
    public CharacterController charCon;
    private Vector3 moveInput;
    public Transform camTrans;
    public float mouseSensitivity;
    public bool invertX;
    public bool invertY;
    private bool canJump;
    private bool canDoubleJump;
    public Transform groundCheckPoint;
    public LayerMask whatIsGround;
    public Animator anim;
    public Transform firePoint;
    public Gun activeGun;
    public List<Gun> allGuns = new List<Gun>();
    public List<Gun> unlockableGuns = new List<Gun>();
    [field: SerializeField] private PlayerData _playerData;
    public int currentGun;
    public Transform adsPoint;
    public Transform gunHolder;
    private Vector3 gunStartPos;
    public float adsSpeed = 2f;
    public GameObject muzzleFlash;
    public AudioSource footstepFast;
    public AudioSource footstepSlow;
    private float bounceAmount;
    private bool bounce;
    public float maxViewAngle = 60;

    private void Awake()
    {
        Instance = this;
    }

    // Start is called before the first frame update
    void Start()

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

```

{
    currentGun--;
    SwitchGun();

    gunStartPos = gunHolder.localPosition;
}

// Update is called once per frame
void Update()
{
    if (!UIController.Instance.pauseScreen.activeInHierarchy &&
        !GameManager.Instance.levelEnding)
    {
        float yStore = moveInput.y;

        Vector3 vertMove = transform.forward * Input.GetAxis("Vertical");
        Vector3 horiMove = transform.right * Input.GetAxis("Horizontal");

        moveInput = horiMove + vertMove;
        moveInput.Normalize();

        if (Input.GetKey(KeyCode.LeftShift))
        {
            moveInput = moveInput * runSpeed;
        }
        else
        {
            moveInput = moveInput * moveSpeed;
        }

        moveInput.y = yStore;
        moveInput.y += Physics.gravity.y * gravityModifier * Time.deltaTime;

        if (charCon.isGrounded)
        {
            moveInput.y = Physics.gravity.y * gravityModifier * Time.deltaTime;
        }

        canJump = Physics.OverlapSphere(groundCheckPoint.position, .25f, whatIsGround).Length > 0;

        if (Input.GetKeyDown(KeyCode.Space) && canJump)
        {
            moveInput.y = jumpPower;

            canDoubleJump = true;

            AudioManager.Instance.PlaySFX(8);
        }

        else if (canDoubleJump && Input.GetKeyDown(KeyCode.Space))
        {
            moveInput.y = jumpPower;
        }
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        canDoubleJump = false;

        AudioManager.Instance.PlaySFX(8);

    }

    if (bounce)
    {
        bounce = false;
        moveInput.y = bounceAmount;

        canDoubleJump = true;
    }

    charCon.Move(moveInput * Time.deltaTime);

    Vector2 mouseInput = new Vector2(Input.GetAxisRaw("Mouse X"), Input.GetAxisRaw("Mouse Y")) * mouseSensitivity;

    if (invertX)
    {
        mouseInput.x = - mouseInput.x;
    }

    if (invertY)
    {
        mouseInput.y = - mouseInput.y;
    }

    transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles.x,
        transform.rotation.eulerAngles.y + mouseInput.x, transform.rotation.eulerAngles.z);

    camTrans.rotation = Quaternion.Euler(camTrans.rotation.eulerAngles + new Vector3(-
        mouseInput.y, 0f, 0f));

    if (camTrans.rotation.eulerAngles.x > maxViewAngle && camTrans.rotation.eulerAngles.x <
        180f)
    {
        camTrans.rotation = Quaternion.Euler(maxViewAngle, camTrans.rotation.eulerAngles.y,
            camTrans.rotation.eulerAngles.z);
    }
    else if (camTrans.rotation.eulerAngles.x > 180f && camTrans.rotation.eulerAngles.x < 360f -
        maxViewAngle)
    {
        camTrans.rotation = Quaternion.Euler(-maxViewAngle, camTrans.rotation.eulerAngles.y,
            camTrans.rotation.eulerAngles.z);
    }

    muzzleFlash.SetActive(false);

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

if (Input.GetMouseButtonDown(0) && activeGun.fireCounter <= 0)
{

    RaycastHit hit;
    if (Physics.Raycast(camTrans.position, camTrans.forward, out hit, 50f))
    {   if (Vector3.Distance(camTrans.position, hit.point) > 2f)
        {
            firePoint.LookAt(hit.point);
        }

    } else
    {
        firePoint.LookAt(camTrans.position + (camTrans.forward * 30f));
    }

    FireShot();
}

if (Input.GetMouseButton(0) && activeGun.canAutoFire)
{
    if (activeGun.fireCounter <= 0)
    {
        FireShot();
    }
}

if (Input.GetKeyDown(KeyCode.Tab))
{
    SwitchGun();
}

if (Input.GetMouseButtonDown(1))
{
    CameraController.Instance.ZoomIn(activeGun.zoomAmount);
}

if (Input.GetMouseButton(1))
{
    gunHolder.position = Vector3.MoveTowards(gunHolder.position, adsPoint.position, adsSpeed
        * Time.deltaTime);
} else
{
    gunHolder.localPosition = Vector3.MoveTowards(gunHolder.localPosition, gunStartPos,
        adsSpeed * Time.deltaTime);
}
if (Input.GetMouseButtonUp(1))
{
    CameraController.Instance.ZoomOut();
}

anim.SetFloat("moveSpeed", moveInput.magnitude);
anim.SetBool("onGround", canJump);

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}

public void FireShot()
{
    if (activeGun.currentAmmo > 0)
    {
        activeGun.currentAmmo--;

        Instantiate(activeGun.bullet, firePoint.position, firePoint.rotation);

        activeGun.fireCounter = activeGun.fireRate;

        UIController.Instance.ammoText.text = "Патрони: " + activeGun.currentAmmo;

        muzzleFlash.SetActive(true);
    }
}

public void SwitchGun()
{
    activeGun.gameObject.SetActive(false);

    currentGun++;

    if (currentGun >= allGuns.Count)
    {
        currentGun = 0;
    }

    activeGun = allGuns[currentGun];
    activeGun.gameObject.SetActive(true);

    UIController.Instance.ammoText.text = "Патрони: " + activeGun.currentAmmo;

    firePoint.position = activeGun.firepoint.position;
}

public void AddGun(string gunToAdd)
{
    bool gunUnlocked = false;

    if (unlockableGuns.Count > 0)
    {
        for (int i = 0; i < unlockableGuns.Count; i++)
        {
            if (unlockableGuns[i].gunName == gunToAdd)
            {
                gunUnlocked = true;

                allGuns.Add(unlockableGuns[i]);
            }
        }
    }
}

```

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		65

```

        unlockableGuns.RemoveAt(i);

        i = unlockableGuns.Count;
    }
}

if (gunUnlocked)
{
    currentGun = allGuns.Count - 2;
    SwitchGun();
}

}

public void Bounce(float bounceForce)
{
    bounceAmount = bounceForce;
    bounce = true;
}

public void LoadData()
{
    if(PlayerPrefs.HasKey("PlayerHealth"))
    {
        playerHealth.SetCurrentHealthPoints(PlayerPrefs.GetInt("PlayerHealth"));
    }

    foreach (var gun in allGuns)
    {
        if(PlayerPrefs.HasKey($"{gun.name}"))
        {
            gun.SetAmmo(PlayerPrefs.GetInt($"{gun.name}"));
        }
    }
}

public void SaveData()
{
    PlayerPrefs.SetInt("PlayerHealth", playerHealth.currentHealth);
    foreach (var gun in allGuns)
    {
        PlayerPrefs.SetInt($"{gun.name}", gun.currentAmmo);
    }
}
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		66

Клас PlayerHealthController

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerHealthController : MonoBehaviour
{
    public static PlayerHealthController Instance;
    public int maxHealth;
    public int currentHealth;
    public float invincibleLength = 1f;
    private float invincCounter;

    private void Awake()
    {
        Instance = this;
    }

    // Start is called before the first frame update
    void Start()
    {
        // currentHealth = maxHealth;
        UIController.Instance.healthSlider.maxValue = maxHealth;
        UIController.Instance.healthSlider.value = currentHealth;
        UIController.Instance.healthText.text = "Здоров'я: " + currentHealth + "/" + maxHealth;
    }

    // Update is called once per frame
    void Update()
    {
        if (invincCounter > 0)
        {
            invincCounter -= Time.deltaTime;
        }
    }

    public void DamagePlayer(int damageAmount)
    {
        if (invincCounter <= 0 && !GameManager.Instance.levelEnding)
        {
            AudioManager.Instance.PlaySFX(7);

            currentHealth -= damageAmount;

            UIController.Instance.ShowDamage();

            if (currentHealth <= 0)
            {
                gameObject.SetActive(false);
            }
        }
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        currentHealth = 0;

        GameManager.Instance.PlayerDied();

        AudioManager.Instance.StopBGM();
        AudioManager.Instance.PlaySFX(6);
        AudioManager.Instance.StopSFX(7);
    }

    invincCounter = invincibleLength;

    UIController.Instance.healthSlider.value = currentHealth;
    UIController.Instance.healthText.text = "Здоров'я: " + currentHealth + "/" + maxHealth;
}
}

public void HealPlayer(int healAmount)
{
    currentHealth += healAmount;

    if (currentHealth > maxHealth)
    {
        currentHealth = maxHealth;
    }

    UIController.Instance.healthSlider.value = currentHealth;
    UIController.Instance.healthText.text = "Здоров'я: " + currentHealth + "/" + maxHealth;
}

public void SetCurrentHealthPoints (int healthPoints)
{
    currentHealth = healthPoints;
    UIController.Instance.healthSlider.value = currentHealth;
    UIController.Instance.healthText.text = "Здоров'я: " + currentHealth + "/" + maxHealth;
}
}

```

Клас EnemyController

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class EnemyController : MonoBehaviour
{
    private bool chasing;
    public float distanceToChase = 10f;
    public float distanceToLose = 15f;
    public float distanceToStop = 2f;
    private Vector3 targetPoint;
    private Vector3 startPoint;
    public NavMeshAgent agent;
    public float keepChasingTime = 5f;
    private float chaseCounter;
    public GameObject bullet;
    public Transform firePoint;
    public float fireRate;
    public float waitBetweenShots = 2f, timeToShoot = 1f;
    private float fireCount;
    private float shootWaitCounter, shootTimeCounter;
    public Animator anim;
    private bool wasShot;

    // Start is called before the first frame update
    void Start()
    {
        startPoint = transform.position;
        shootTimeCounter = timeToShoot;
        shootWaitCounter = waitBetweenShots;
    }

    // Update is called once per frame
    void Update()
    {
        targetPoint = PlayerController.Instance.transform.position;
        targetPoint.y = transform.position.y;

        if (!chasing)
        {
            if (Vector3.Distance(transform.position, targetPoint) < distanceToChase)
            {
                chasing = true;

                shootTimeCounter = timeToShoot;
                shootWaitCounter = waitBetweenShots;
            }
        }
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (chaseCounter > 0)
{
    chaseCounter -= Time.deltaTime;

    if (chaseCounter <= 0)
    {
        agent.destination = startPoint;
    }
}

if (agent.remainingDistance < .25f)
{
    anim.SetBool("isMoving", false);
} else
{
    anim.SetBool("isMoving", true);
}
}
else
{
    if (Vector3.Distance(transform.position, targetPoint) > distanceToStop)
    {
        agent.destination = targetPoint;

    } else
    {
        agent.destination = transform.position;
    }

    if (Vector3.Distance(transform.position, targetPoint) > distanceToLose)
    {
        if (!wasShot)
        {
            chasing = false;

            chaseCounter = keepChasingTime;
        }

    } else
    {
        wasShot = false;
    }

    if (shootWaitCounter > 0)
    {
        shootWaitCounter -= Time.deltaTime;

        if (shootWaitCounter <= 0)
        {
            shootTimeCounter = timeToShoot;
        }
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        anim.SetBool("isMoving", true);

    } else {

        if (PlayerHealthController.Instance.gameObject.activeInHierarchy)
        {
            shootTimeCounter -= Time.deltaTime;

            if (shootTimeCounter > 0)
            {
                fireCount -= Time.deltaTime;

                if (fireCount <= 0)
                {
                    fireCount = fireRate;

                    firePoint.LookAt(PlayerController.Instance.transform.position + new Vector3(0f, 1.2f,
0f));
                    // кут нахилу до гравця

                    Vector3 targetDir = PlayerController.Instance.transform.position - transform.position;
                    float angle = Vector3.SignedAngle(targetDir, transform.forward, Vector3.up);

                    if (Mathf.Abs(angle) < 30f)
                    {
                        Instantiate(bullet, firePoint.position, firePoint.rotation);

                        anim.SetTrigger("fireShot");
                    } else
                    {
                        shootWaitCounter = waitBetweenShots;
                    }

                }

                agent.destination = transform.position;

            } else
            {
                shootWaitCounter = waitBetweenShots;
            }
        }
        anim.SetBool("isMoving", false);
    }
}

public void GetShot()
{
    wasShot = true;

    chasing = true;
}
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас EnemyHealthController

```

using System.Collections;
using System.Collections.Generic;
using SaveLoadData;
using UnityEngine;

public class EnemyHealthController : MonoBehaviour
{
    public int currentHealth = 5;
    public EnemyController theEC;
    [SerializeField] private EnemyData _enemyData;
    public EnemyData GetEnemyData()
    {
        return _enemyData;
    }

    void Awake()
    {
        // _enemyData.SetEnemy(this.gameObject);
    }

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

    public void DamageEnemy(int damageAmount)
    {
        currentHealth -= damageAmount;

        if (theEC != null)
        {
            theEC.GetShot();
        }
        if (currentHealth <= 0)
        {
            gameObject.SetActive(false);
            _enemyData.SetIsDied(true);
            AudioManager.Instance.PlaySFX(2);
        }
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		72

Клас CameraController

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraController : MonoBehaviour
{
    public static CameraController Instance;
    public Transform target;
    private float startFOV;
    private float targetFOV;
    public float zoomSpeed = 1f;
    public Camera theCam;

    private void Awake()
    {
        Instance = this;
    }
    // Start is called before the first frame update
    void Start()
    {
        startFOV = theCam.fieldOfView;
        targetFOV = startFOV;
    }

    // Update is called once per frame
    void LateUpdate()
    {
        transform.position = target.position;
        transform.rotation = target.rotation;
        theCam.fieldOfView = Mathf.Lerp(theCam.fieldOfView, targetFOV, zoomSpeed *
            Time.deltaTime);
    }

    public void ZoomIn(float newZoom)
    {
        targetFOV = newZoom;
    }

    public void ZoomOut()
    {
        targetFOV = startFOV;
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		73

Клас BulletController

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BulletController : MonoBehaviour
{
    public float moveSpeed;
    public float lifeTime;
    public Rigidbody theRB;
    public GameObject impactEffect;
    public int damage = 1;
    public bool damageEnemy;
    public bool damagePlayer;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        theRB.velocity = transform.forward * moveSpeed;
        lifeTime -= Time.deltaTime;

        if (lifeTime <= 0)
        {
            Destroy(gameObject);
        }
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Enemy" && damageEnemy)
        {
            other.gameObject.GetComponent<EnemyHealthController>().DamageEnemy(damage);
        }
        if (other.gameObject.tag == "Headshot" && damageEnemy)
        {
            other.transform.parent.GetComponent<EnemyHealthController>().DamageEnemy(damage * 2);
            Debug.Log("Headshot hit");
        }
        if (other.gameObject.tag == "Player" && damagePlayer)
        {
            PlayerHealthController.Instance.DamagePlayer(damage);
        }
        Destroy(gameObject);
        Instantiate(impactEffect, transform.position + (transform.forward * (-moveSpeed * Time.deltaTime)), transform.rotation);
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас GameManager

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour
{
    public static GameManager Instance;
    public SaveLoadData.SaveLoadData SaveLoadData;
    public float waitAfterDying = 2f;
    [HideInInspector]
    public bool levelEnding;
    private List<GameObject> objectsToDestroy;

    private void Awake()
    {
        Instance = this;
        if (Instance == null)
            Instance = this;
        if (Instance != this)
        {
            Destroy(this.gameObject);
            return;
        }
    }

    // Start is called before the first frame update
    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            PauseUnpause();
        }
    }

    public void PlayerDied()
    {
        StartCoroutine(PlayerDiedCo());
        //SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

    }

    public IEnumerator PlayerDiedCo()
    {
        yield return new WaitForSeconds(waitAfterDying);

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

    public void PauseUnpause()
    {
        if (UIController.Instance.pauseScreen.activeInHierarchy)
        {
            UIController.Instance.pauseScreen.SetActive(false);
            Cursor.lockState = CursorLockMode.Locked;
            Cursor.visible = false;
            Time.timeScale = 1f;
            PlayerController.Instance.footstepFast.Play();
            PlayerController.Instance.footstepSlow.Play();

        } else
        {
            UIController.Instance.pauseScreen.SetActive(true);
            Cursor.lockState = CursorLockMode.None;
            Cursor.visible = true;
            Time.timeScale = 0f;
            PlayerController.Instance.footstepFast.Stop();
            PlayerController.Instance.footstepSlow.Stop();
        }
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас Gun

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

public class Gun : MonoBehaviour
{
    public GameObject bullet;
    public bool canAutoFire;
    public float fireRate;
    [HideInInspector]
    public float fireCounter;
    public int currentAmmo;
    public int pickupAmount;
    public Transform firepoint;
    public float zoomAmount;
    public string gunName;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (fireCounter > 0)
        {
            fireCounter -= Time.deltaTime;
        }
    }

    public void GetAmmo()
    {
        int maxAmmo = GetMaxAmmo();

        if (currentAmmo < maxAmmo)
        {
            currentAmmo += pickupAmount;
            if (currentAmmo > maxAmmo)
            {
                currentAmmo = maxAmmo;
            }
            UIController.Instance.ammoText.text = "Патрони: " + currentAmmo;
        }
    }

    private int GetMaxAmmo()
    {
        if (gunName == "pistol")
```

```

{
    return 60;
}
else if (gunName == "repeater")
{
    return 120;
}
else if (gunName == "sniper")
{
    return 15;
}
else if (gunName == "rocket launcher")
{
    return 5;
}
return 0;
}

public void SetAmmo(int ammo)
{
    currentAmmo = ammo;
    UIController.Instance.ammoText.text = "Патрони: " + currentAmmo;
}
}

```

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас TargetMovement

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

public class TargetMovement : MonoBehaviour
{
    public bool shouldMove;
    public bool shouldRotate;
    public float moveSpeed;
    public float rotateSpeed;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (shouldMove)
        {
            transform.position += new Vector3(moveSpeed, 0f, 0f) * Time.deltaTime;
        }

        if (shouldRotate)
        {
            transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles + new Vector3(0f,
            rotateSpeed * Time.deltaTime, 0f));
        }
    }
}
```

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас Explosion

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Explosion : MonoBehaviour
{
    public int damage = 25;
    public bool damageEnemy;
    public bool damagePlayer;

    private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Enemy" && damageEnemy)
        {
            other.gameObject.GetComponent<EnemyHealthController>().DamageEnemy(damage);
        }

        if (other.gameObject.tag == "Player" && damagePlayer)
        {
            PlayerHealthController.Instance.DamagePlayer(damage);
        }
    }
}

```

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас Turret

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Turret : MonoBehaviour
{
    public GameObject bullet;
    public float rangeToTargetPlayer;
    public float timeBetweenShots = .5f;
    private float shotCounter;
    public Transform gun;
    public Transform firepoint;
    public float rotateSpeed = 45f;

    // Start is called before the first frame update
    void Start()
    {
        shotCounter = timeBetweenShots;
    }

    // Update is called once per frame
    void Update()
    {
        if (Vector3.Distance(transform.position, PlayerController.Instance.transform.position) <
            rangeToTargetPlayer)
        {
            gun.LookAt(PlayerController.Instance.transform.position + new Vector3(0f, 1.2f, 0f));

            shotCounter -= Time.deltaTime;

            if (shotCounter <= 0)
            {
                Instantiate(bullet, firepoint.position, firepoint.rotation);
                shotCounter = timeBetweenShots;
            }
        } else
        {
            shotCounter = timeBetweenShots;

            gun.rotation = Quaternion.Lerp(gun.rotation, Quaternion.Euler(0f, gun.rotation.eulerAngles.y
                + 10f, 0f), rotateSpeed * Time.deltaTime);
        }
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас CheckpointController

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class CheckpointController : MonoBehaviour
{
    public string cpName;

    // Start is called before the first frame update
    void Start()
    {
        if (PlayerPrefs.HasKey(SceneManager.GetActiveScene().name + "_cp"))
        {
            if (PlayerPrefs.GetString(SceneManager.GetActiveScene().name + "_cp") == cpName)
            {
                PlayerController.Instance.transform.position = transform.position;
                Physics.SyncTransforms();
                Debug.Log("Player starting at " + cpName);
                GameManager.Instance.SaveLoadData.LoadData();
                GameManager.Instance.SaveLoadData.SaveData();
                PlayerController.Instance.LoadData();
                PlayerController.Instance.SaveData();
            }
            else
            {
                GameManager.Instance.SaveLoadData.SaveData();
                PlayerController.Instance.SaveData();
            }
        }
        else
        {
            GameManager.Instance.SaveLoadData.SaveData();
            PlayerController.Instance.SaveData();
        }
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.L))
        {
            PlayerPrefs.SetString(SceneManager.GetActiveScene().name + "_cp", "");
        }
    }

    private void OnTriggerEnter(Collider other)

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    if (other.gameObject.tag == "Player")
    {
        PlayerPrefs.SetString(SceneManager.GetActiveScene().name + "_cp", cpName);
        Debug.Log("Player hit " + cpName);
        GameManager.Instance.SaveLoadData.SaveData();
        PlayerController.Instance.SaveData();
        AudioManager.Instance.PlaySFX(1);
    }
}
}

```

		Надворний М.Ю			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		83

Клас Bouncepad

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

public class Bouncepad : MonoBehaviour
{
    public float bounceForce;

    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
        {
            PlayerController.Instance.Bounce(bounceForce);
            AudioManager.Instance.PlaySFX(0);
        }
    }
}
```

Клас AudioManager

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AudioManager : MonoBehaviour
{
    public static AudioManager Instance;
    public AudioSource bgm;
    public AudioSource victory;
    public AudioSource[] soundEffects;
    private void Awake()
    {
        Instance = this;
    }
    // Start is called before the first frame update
    void Start()
    {

    }

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

    }

    public void StopBGM()
    {
        bgm.Stop();
    }

    public void PlayLevelVictory()
    {
        StopBGM();
        victory.Play();
    }

    public void PlaySFX(int sfxNumber)
    {
        soundEffects[sfxNumber].Stop();
        soundEffects[sfxNumber].Play();
    }

    public void StopSFX(int sfxNumber)
    {
        soundEffects[sfxNumber].Stop();
    }
}

```

		Надворний М.Ю			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		