

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

**«3D-гра в жанрі "action" із акцентом на складову
шутера»**

Виконав студент групи КН-19-2
МУРЕВИЧ Владислав Юрійович

Керівник роботи:
ЛЕВЧЕНКО Антон Юрійович

Консультант:
КУЗЬМЕНКО Олександр Вікторович

Рецензент:
ПЕТРОСЯН Руслан Валерікович

Житомир – 2023

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

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

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

Здобувач вищої освіти: **МУРЕВИЧ Владислав Юрійович**

Керівник роботи: **ЛЕВЧЕНКО Антон Юрійович**

Тема роботи: **«3D-гра в жанрі "action" із акцентом на складову шутера»**,
затверджена наказом закладу вищої освіти від **«05» січня 2023 р., №10/с**

Термін здачі закінченої роботи: **«20» червня 2023 р.**

Вихідні дані роботи: об'єктом дослідження є ігровий додаток жанру екшен з акцентом на складову шутера. Предмет дослідження: розробка комп'ютерної гри в середовищі Unity із застосуванням мови програмування C#

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

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Кузьменко О.В.	03.02.2023р.	03.02.2023р.
2	Кузьменко О.В.	04.04.2023р.	04.04.2023р.
3	Кузьменко О.В.	16.05.2023р.	16.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2023 – 25 квітня 2023	Виконано
2	Проектування структури системи	25 квітня 2023 – 15 травня 2023	Виконано
3	Написання програмного коду	14 травня 2023 – 16 червня 2023	Виконано
4	Тестування системи	16 червня 2023 – 20 червня 2023	Виконано
5	Оформлення пояснювальної записки	16 червня 2023 – 25 червня 2023	Виконано

Здобувач вищої освіти **Владислав МУРЕВИЧ**
Керівник **Антон ЛЕВЧЕНКО**

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему «3D-гра в жанрі "action" із акцентом на складову шутера» складається вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на **61** сторінці друкованого тексту.

Пояснювальна записка має **4** сторінки додатків. Список використаних джерел містить **11** найменувань і займає **1** сторінку. В роботі наведено **32** рисунки і 24 таблиці. Загальний обсяг роботи – **65** сторінки.

У першому розділі були визначені основні вимоги та критерії до проекту, були оглянуті альтернативні ігрові додатки, було брано середовища для розробки компонентів гри.

У другому розділі було проведено проектування ігрового додатку, було створено 3D моделі для подальшого використання в грі, а також написані та пояснені скрипти для реалізації основних механік гри

У третьому розділі був розроблений користувацький інтерфейс, проведено тестування компонентів гри, щоб виявити і усунути проблеми. Було створено чек лісти

Висновок містить в собі результати виконаної роботи з розробки ігрового додатку.

					КН.КР.Б – 122 – 23 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	3D-гра в жанрі "action" із акцентом на складову шутера Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.	В.Ю. МУРЕВИЧ						3	65
Керівник	А.Ю. ЛЕВЧЕНКО					Житомирська політехніка, група КН-19-2		
Рецензент	Р.В. ПЕТРОСЯН							
Н. контр.								
Зав. каф.	М.С. ГРАФ							

ABSTRACT

The explanatory note to the qualification work on the topic "A 3D game in the genre of "action" with an emphasis on the shooter component" consists of an introduction, three chapters, conclusions, a list of used literature and an appendix.

The text part is laid out on **61** pages of printed text. The explanatory note has **4** pages of annexes. The list of used sources contains **11** names and occupies **1** page. The work contains **32** figures and **24** tables. The total volume of work is **65** pages.

In the first section, the main requirements and criteria for the project were determined, alternative game applications were reviewed, and the environment for the development of game components was selected

In the second section, the design of the game application was carried out, 3D models were created for further use in the game, and scripts were written and explained to implement the main mechanics of the game

In the third chapter, the user interface was developed, the game components were tested to identify and eliminate problems. Check sheets were created

The conclusion contains the results of the work done on the development of the game application.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

РЕФЕРАТ	3
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ.....	9
1.1 Постановка задачі	9
1.2 Концепт-документ	9
1.3 Існуючі аналоги програмного продукту	10
1.4 Обґрунтування вибору інструментальних засобів	12
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	16
2.1 Визначення варіантів використання	16
2.2 Алгоритм роботи та стани програмної системи	17
2.3 Об'єктно-орієнтовна модель системи	20
2.4 Взаємодія об'єктів системи	22
2.5 Взаємодія компонентів системи	22
2.6 Реалізація ігрового проекту	24
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ “From The Darkness”	45
3.1. Порядок встановлення та налаштування системи	45
3.2. Структура інтерфейсу	45
3.3. Тестування роботи системи	49
ЛІТЕРАТУРА.....	61
ДОДАТКИ	62

ВСТУП

Актуальність теми: Тема створення відеоігор залишається дуже актуальною і важливою в сучасному світі з різноманітними можливостями технологій.

Популярність відеоігор: Відеоігри продовжують зростати у популярності серед різних груп користувачів, включаючи дітей, підлітків, молодь і навіть дорослих. Вони стали важливим розважальним засобом і джерелом розваг для мільйонів людей по всьому світу.

Розвиток технологій: З постійним розвитком технологій, таких як штучний інтелект, віртуальна реальність, розширена реальність та обlačні рішення, відеоігри отримують нові можливості для реалізації інноваційних ідей. Розробники можуть створювати більш реалістичні графічні ефекти, інтерактивні світи та складні сценарії.

Комерційний успіх: Відеоігрова індустрія є великим ринком зі значним комерційним потенціалом. Ігрові компанії та розробники можуть заробляти значні прибутки від продажу ігор, мікротранзакцій, додаткових контентів та інших способів монетизації. Відомі ігри можуть стати глобальними франшизами, які включають не тільки саму гру, а й книги, фільми, іграшки та інші продукти.

Інновації та творчість: Створення відеоігор є місцем для розкриття творчого потенціалу. Розробники можуть втілювати свої ідеї, створювати унікальні світи, персонажів та історії. Це дозволяє їм експериментувати з різними жанрами, геймплеєм та інноваційними механіками.

Соціальна взаємодія та спільнота: Відеоігри також стають місцем для соціальної взаємодії, коли гравці можуть грати разом у мережі, спілкуватися, спільно розв'язувати завдання та досліджувати віртуальні світи. Відеоігри створюють спільноти фанатів, які активно обговорюють, спілкуються та співпрацюють один з одним.

Усі ці фактори підтримують актуальність теми створення відеоігор, надають можливості для інновацій, творчості, комерційного успіху та

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

соціальної взаємодії. Ця галузь продовжує розвиватися, пропонуючи нові можливості для розробників та незабутні враження для гравців.

Мета: дослідження особливостей моделювання та аналізу, мови програмування C# в середовищі розробки Unity, створення UML-діаграм, поставлення вимог до програмного продукту, опис моделі програмного комплексу на логічному рівні, створення фізичної моделі для розробки, створення макету рівнів та 3D моделей для гри.

Об'єкт дослідження: уніфікована мова моделювання (UML), її методи та засоби проектування комп'ютерної гри, використання технологій Unity, створення 3D моделей.

Предмет дослідження: розробка комп'ютерної гри в середовищі Unity із застосуванням мови програмування C#, використанням UML для проектування, створення 3D моделей.

Практичне значення: програмний продукт заснований на інтерактивній взаємодії з користувачем(гравцем), через яку гравець буде проходити під час проходження гри у розважальних цілях.

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

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

1.1 Постановка задачі

Назва системи: «From the Darkness» - комп'ютерна гра у жанрі екшн із акцентом на складову шутеру.

Призначення системи: програмний продукт оснований на інтерактивній взаємодії з користувачем (гравцем), через який гравець буде проходити під час користування програмним додатком (проходити гру) у розважальних цілях. Спрямований на гравців, яких цікавлять динамічні та швидкі події, основані на реагуванні та прийнятті рішень користувача.

Вимоги до програмного додатку:

- комфортне керування, основне заняття в грі це взаємодія з ворогами, механіка стрільби має бути лояльною до гравця у керуванні
- зрозумілий інтерфейс, має бути зручним та зрозумілим, повинен мати підказки
- присутні налаштування гри
- гра має відповідати поставленому жанру – екшн з ухилом на шутерну складову
- графіка у грі в стилі low-poly, низькополігональна графіка

1.2 Концепт-документ

Жанр і аудиторія: гра належить до жанру екшн, є ухил на шутерну складову, від першої особи, основна аудиторія це підлітки та дорослі.

Основні особливості гри: шутер, екшн, велика кількість зброї.

Геймплей: гра представляє собою арену на якій гравець вступає в бій з різними видами ворогів, використовуючи широкий арсенал зброї.

Платформа: гра буде розроблятися тільки під PC-платформу.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3 Існуючі аналоги програмного продукту

В даному підрозділі знайдемо аналогічні програмні продукти у відповідному ігровому жанрі, виявимо їх функціональні можливості.

Borderlands - відеогра жанру науково-фантастичного шутера від першої особи з елементами RPG, розроблена компанією Gearbox.



Рис.1.1. Геймплей гри Borderlands

Переваги:

- динамічний геймплей
- система прокачування характеристик
- величезна кількість різноманітної зброї
- особливий графічний стиль
- чудовий левел-дизайн
- добрий саундтрек та звукові ефекти
- присутня кооперативна гра

Недоліки:

- недостатнє поєднання між зброєю на спеціальним навичками ігрових персонажів

Ultrakill - це швидкий ультражорстокий шутер від старої школи, який поєднує в собі класичні шутери.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис.1.2. Геймплей гри Ultrakill

Переваги:

- унікальна динаміка гри та усього, що відбувається у ній
- широка механіка переміщення, має свої навички
- гарний саундтрек та звукові ефекти
- цікаві механіки зброї

Недоліки:

- досить складна для проходження, хардкорна

Redmatch 2 - безкоштовний онлайн-швидкий шутер від першої особи для кількох гравців із 15 картами, декількома видами зброї та ігровими режимами на вибір.



Рис.1.3. Скріншот геймплею гри Redmatch 2

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

Переваги:

- динамічний, швидкісний геймплей
- цікава механіка переміщення
- у вільному доступі (безкоштовна)
- гнучка система налаштувань кімнат (ігрових сесій)
- чудова кастомізація персонажа, інтерфейсу та звукових ефектів
- привабливі досягнення, які заохочують гравця до їх виконання

Недоліки:

- відсутня одиночна гра, немає сюжетної компанії
- через велику систему налаштувань кімнат, можливе створення дизбалансу у онлайн грі
- невелика кількість зброї та режимів гри

1.4 Обґрунтування вибору інструментальних засобів

Розробка гри буде вестись у середовищі Unity.

Unity — багатоплатформовий інструмент для розроблення відеоігор і застосунків, і рушій, на якому вони працюють. Створені за допомогою Unity програми працюють на настільних комп'ютерних системах, мобільних пристроях та гральних консолях у дво- та тривимірній графіці, та на пристроях віртуальної чи доповненої реальності. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL.

Unity є одним з найпопулярніших інтегрованих середовищ розробки (IDE) для створення ігор, і воно має численні переваги:

1. Кросплатформеність: Unity дозволяє розробляти ігри для різних платформ, таких як Windows, macOS, Linux, Android, iOS, Xbox, PlayStation та багатьох інших. Розробники можуть створювати гру один раз і потім експортувати її на різні платформи без необхідності переписувати весь код.
2. Легкість використання: Unity має інтуїтивний і простий у використанні інтерфейс, що робить його доступним для розробників з будь-

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

яким рівнем досвіду. Є велика кількість документації, уроків та спільнот, які допомагають новачкам вивчити інструменти та розв'язувати проблеми.

3. Велика екосистема: Unity має широкий вибір розширень, активів, плагінів та ресурсів, які допомагають розробникам прискорити процес створення гри. Існує велика спільнота розробників, де можна знайти відповіді на питання, отримати поради та навіть знайти співробітників для спільної розробки.

4. Графічний движок: Unity має потужний графічний движок, який дозволяє створювати візуально привабливі та реалістичні графічні ефекти. Він підтримує різні види освітлення, тіней, частинок та фізичної симуляції.

5. Скриптованість: Unity використовує скриптову мову C# для програмування гри. C# є потужною мовою з великою кількістю функціональних можливостей. Це дає розробникам багато гнучкості і контролю над розробкою гри.

6. Фізична симуляція: Unity має вбудовану фізичну систему, що дозволяє реалістично моделювати фізику об'єктів у грі. Це дозволяє створювати реалістичні фізичні ефекти та поведінку об'єктів.

7. Анімація: Unity надає розробникам потужні інструменти для створення і анімації персонажів і об'єктів у грі. Розробники можуть створювати складні рухи, змінювати форму об'єктів та контролювати анімацію за допомогою вбудованих інструментів анімації.

8. Можливості розробки мультиплеєра: Unity надає можливості для створення мультиплеєрних ігор, як локальних, так і мережових. Розробники можуть легко створювати з'єднання між гравцями, обмінюватися даними та створювати онлайн-ігри.

В цілому, Unity є потужним та універсальним інструментом для розробки ігор, який забезпечує зручний робочий процес, гнучкість та широкі можливості для реалізації ідей розробників.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

Для створення 3D моделей та анімацій для них буде використано тривимірний графічний редактор 3DS Max

3ds Max є одним з провідних тривимірних графічних редакторів і має численні переваги:

1. Розширені можливості моделювання: 3ds Max надає розробникам широкий набір інструментів для створення складних тривимірних моделей. Він підтримує різні методи моделювання, включаючи полігональне моделювання, моделювання з використанням поверхонь та NURBS-моделювання. Це дозволяє розробникам створювати деталізовані об'єкти з різними формами та складностями.

2. Анімація і кінематика: 3ds Max має потужні інструменти для створення анімації об'єктів. Розробники можуть створювати рухи, інтерактивні анімаційні сценарії, анімувати персонажів та об'єкти. Крім того, 3ds Max підтримує різні методи кінематики, такі як кістякова кінематика (skeletal animation), кінематика інверсного кінця (IK) та інші, що дозволяють розробникам створювати складні рухи об'єктів.

3. Велика бібліотека матеріалів і текстур: 3ds Max має широкий вибір попередньо налаштованих матеріалів, текстур і освітлення, які можна використовувати для створення реалістичних зображень і сцен. Це дозволяє розробникам швидко надати об'єктам візуальну привабливість та деталізацію.

4. Симуляція фізики і реалістичні ефекти: 3ds Max має вбудовані інструменти для симуляції фізики, що дозволяє розробникам створювати реалістичні фізичні ефекти, такі як симуляція руху тканин, рідини, волосся та інших об'єктів. Він також має широкий вибір ефектів, які дозволяють додати спеціальні ефекти, такі як вибухи, дим, вогонь і багато інших.

5. Інтеграція з іншими інструментами: 3ds Max має добру підтримку обміну даними з іншими популярними програмами для тривимірної графіки, такими як Photoshop, After Effects, ZBrush та багатьма іншими. Це дозволяє розробникам легко і ефективно працювати з різними інструментами в своєму розробчому процесі.

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

6. Велика спільнота та ресурси: 3ds Max має широку спільноту розробників, яка активно ділиться знаннями, уроками, ресурсами та плагінами. Це дозволяє розробникам швидко отримувати допомогу та поради в разі потреби.

Загалом, 3ds Max є потужним інструментом для тривимірної графіки, який надає розробникам широкий набір інструментів для створення складних моделей, анімації, реалістичних ефектів та симуляції фізики. Його зручний інтерфейс та можливості інтеграції роблять його популярним в індустрії тривимірної графіки.

Код пишеться на мові програмування C#, за допомогою якої можна створити об'єктно орієнтовну модель системи та в цілому зручно писати скрипти для логіки гри.

Висновки до першого розділу

У ході виконання першого розділу було поставлено завдання та висунуті певні вимоги до розробки системи для кваліфікаційної роботи. Було створено концепт-документ гри. Були розглянуті аналогічні ігрові додатки за тематикою, тобто які підходять до жанру екшн-шутер. Було обрано середовище розробки компонентів гри.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення варіантів використання

У даному розділі буде описано проектування та розробка програмного забезпечення. Відповідно до поставленого завдання і задач, почнемо з підрозділу про визначення варіантів використання, що опишемо у відповідних UML-діаграмах

Діаграма варіантів використання - це UML діаграма за допомогою якої в графічному вигляді можна зобразити вимоги до розроблюваної системи, а також відношення між акторами та прецедентами в системі.

Було розроблено дві діаграми для більш чіткого розуміння – меню і сама гра(ігровий процес). В меню бачимо усі можливі використання, такі як вибір гри, налаштування та вихід.

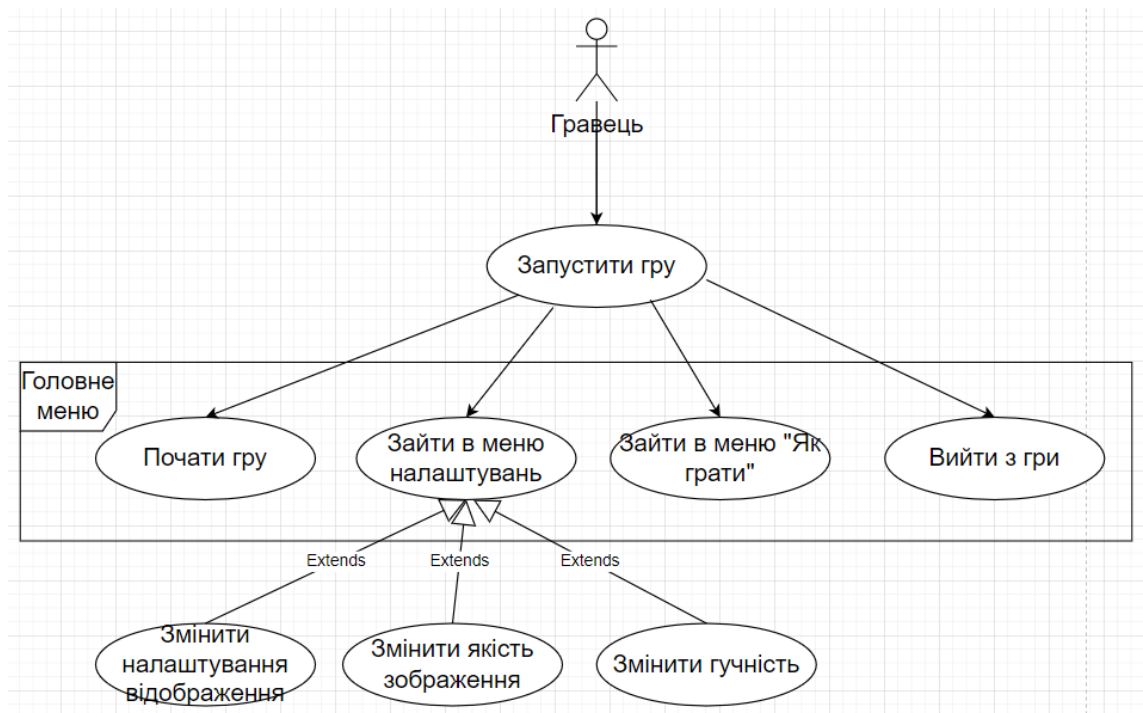


Рис. 2.1 Діаграма варіантів використання (меню)

Дана діаграма описує ігровий процес, зображено три актори – гравець, ігрова логіка та ворожі НІП-и (неігрові персонажі). Гравець має певні основні ігрові

механіки (переміщення, бій, взаємодія зі світом). Гравець пов'язаний з ігровою логікою та ворожими НІП-ами, це показано у взаємодії між гравцем та ворогом. Також на діаграмі показана взаємодія з друзелюбними НІП-ами, але вони не є акторами, адже у них не передбачається набір характеристик актора.

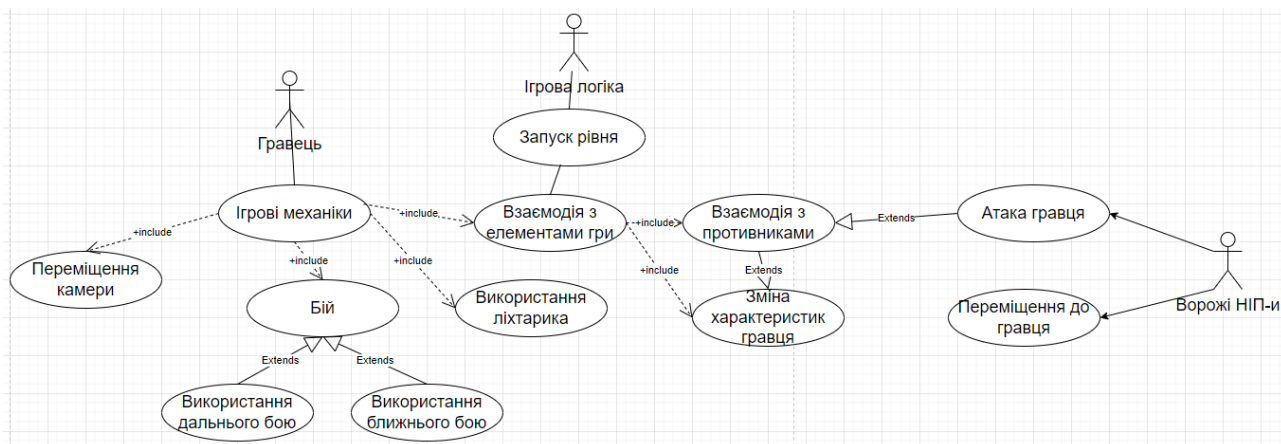


Рис. 2.2 Діаграма варіантів використання (ігровий процес)

2.2 Алгоритм роботи та стани програмної системи

У даному підрозділі будуть описані алгоритми роботи та стани програмної системи у виді діаграм активності або діаграм діяльності.

Діаграма активності складається з таких дій як: action – певне завдання, етап діяльності, decision node – умовна гілка, включає один або декілька виходів, control flows – з'єднувачі між кроками, start node – початковий вузол, початок діяльності, end node – останній крок у дії, кінцевий вузол.

Гра має досить багато активностей, тому було розроблено декілька діаграм діяльності для гри. Почнемо з меню. Початковий стан – момент перед запуском гри, після чого у користувача є вибір, почати гру, що в подальшому приведе до вибору: продовжити гру чи почати нову гру, при продовженні гри постане вибір рівня, після чого відбуватиметься перехід на рівень, що буде кінцевою точкою. В налаштуваннях, будуть активності – зміни налаштувань, після підтвердження або не згоди зі зміною, відбувається повернення до вибору у головному меню.

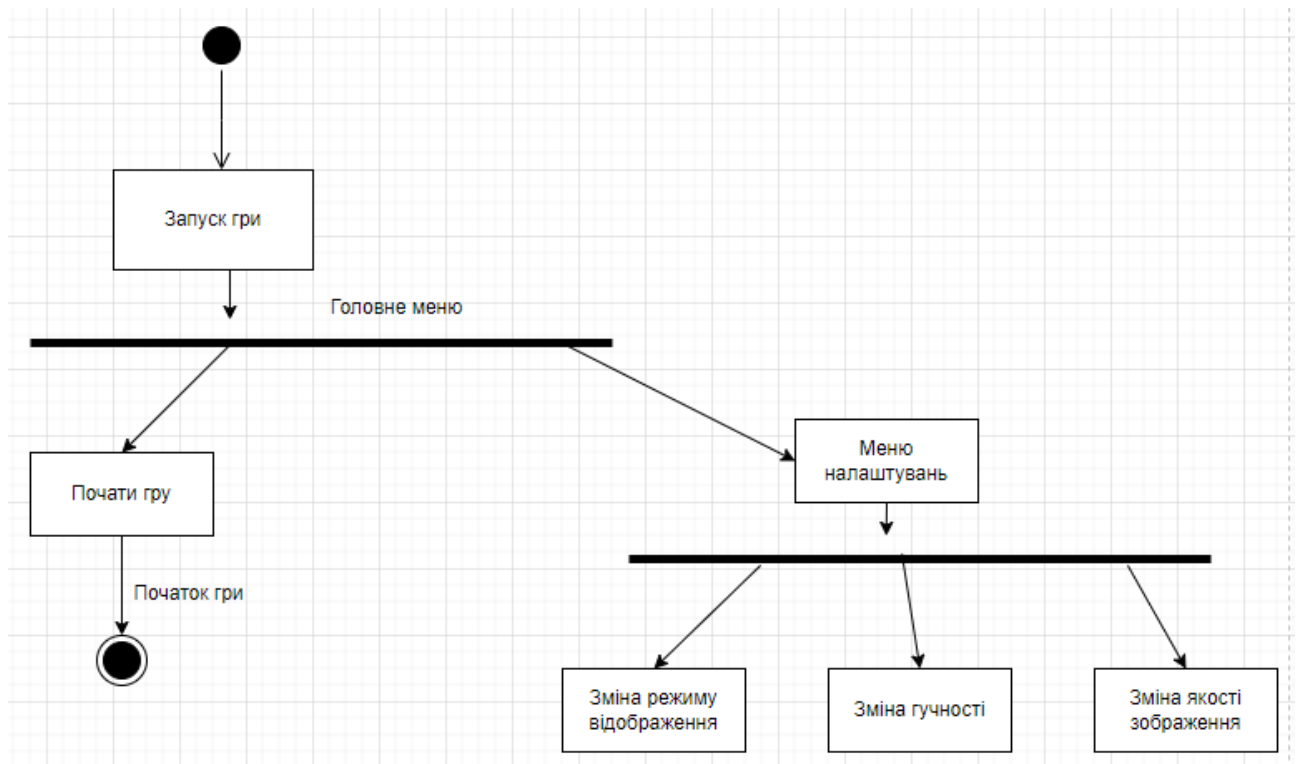


Рис. 2.3 Діаграма активності меню гри

Далі іде діаграма діяльності, які описує загальний процес гри на рівні.

Перейшовши до гри, гравець потрапляє на арену, де матиме можливість відкрити меню з якого він може перейти до головного меню або продовжити гру.

У разі програшу, гравець потрапляє на екран кінця гри, де можна побачити кількість очок зароблених за ігровий сеанс, а також матиме можливість почати гру спочатку, вийти в головне меню, або вийти з гри

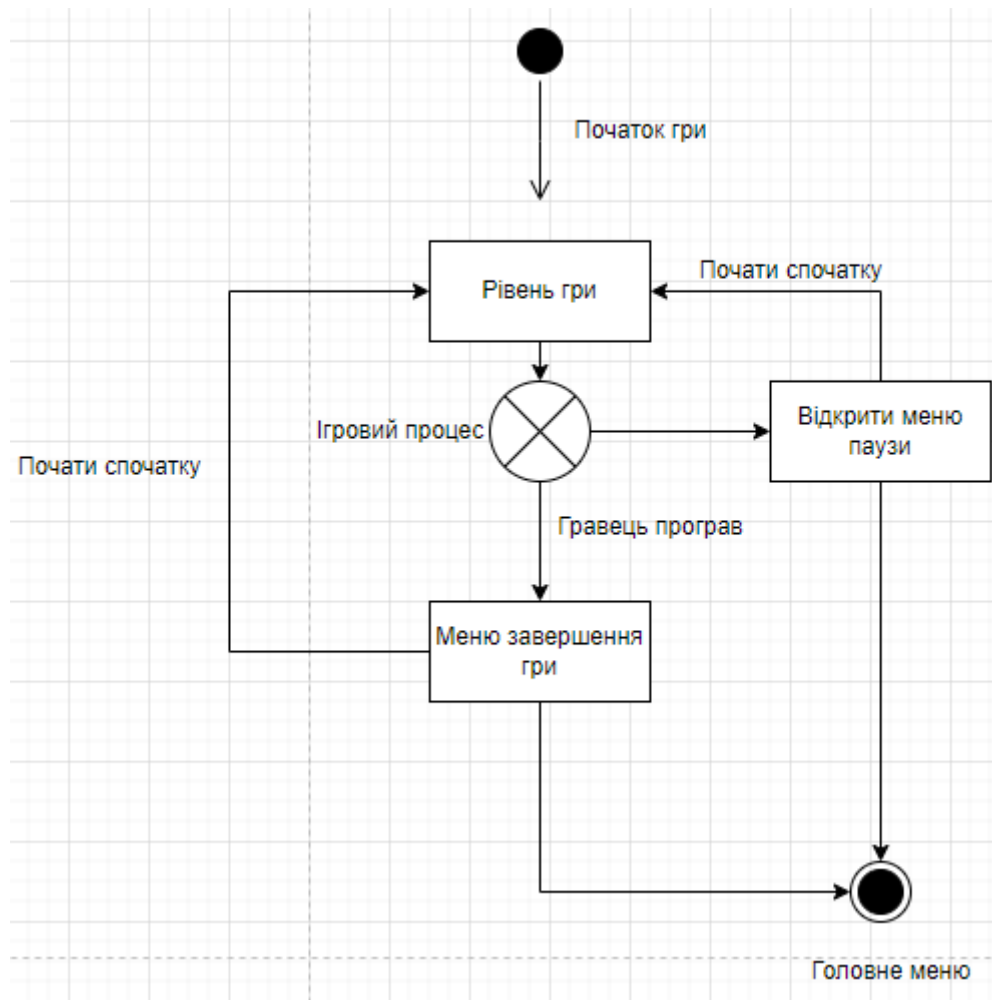


Рис.2.4. Діаграма активності загальний процес гри

Одна з головних діаграм активності – взаємодія гравця з ворогами, оскільки це буде основним заняттям в грі.

Початкова точка це початок дій гравця, таких як переміщення – гравець може уникнути сутички або зайняти більш зручну позицію. Атака по ворогу, у грі передбачено два види зброї. При влучанні у ворога, противнику буде наноситись шкода, в іншому випадку, повертаємось до початкової точки вибору. При нанесенні шкоди, іде перевірка кількості здоров'я ворога, якщо вона менша нуля, взаємодія з цим ворогом припиняється.

У разі атаки ворога по гравцю, при влучанні знімається здоров'я героя. Далі переходимо у стан перевірки здоров'я гравця, у разі відсутності здоров'я, переходимо у кінцеву точку, смерть гравця. Якщо ж гравець має здоров'я, взаємодія продовжується.

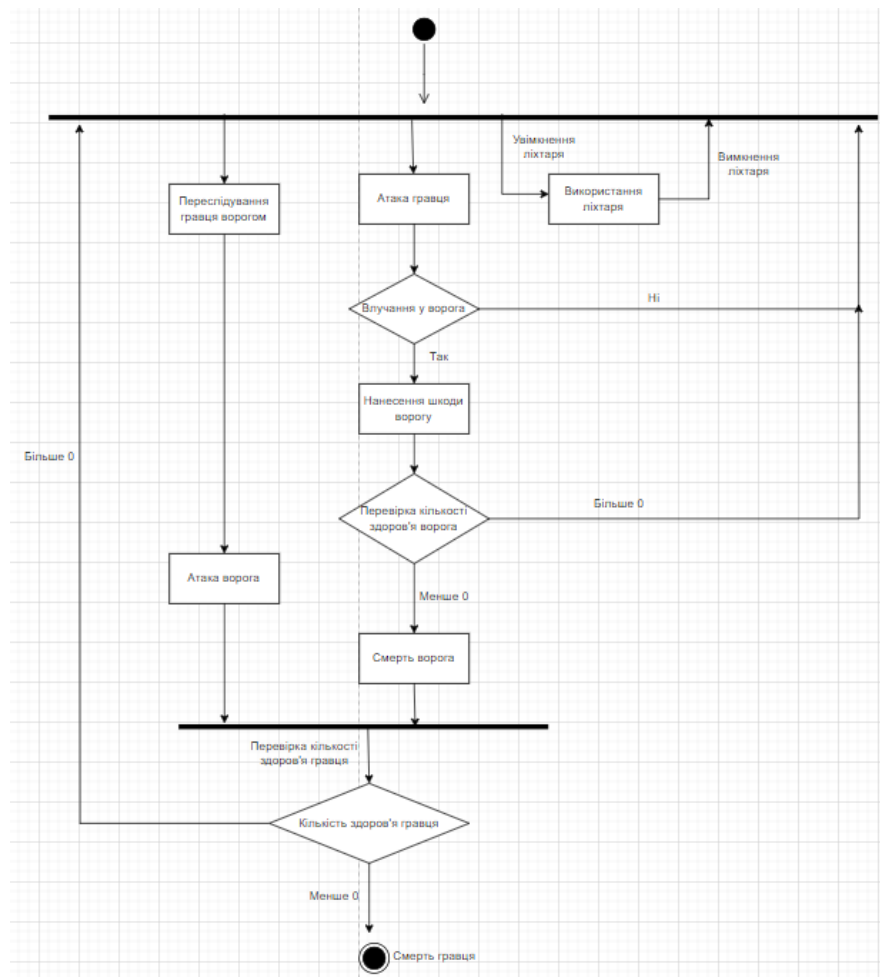


Рис.2.5. Діаграма активності взаємодії гравця з противником

2.3 Об'єктно-орієнтовна модель системи

Розглянемо розроблену діаграму класів до гри та патерни проектування, які будуть використовуватись.

Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На діаграмі показують класи, інтерфейси, об'єкти й кооперації, їх відносини.

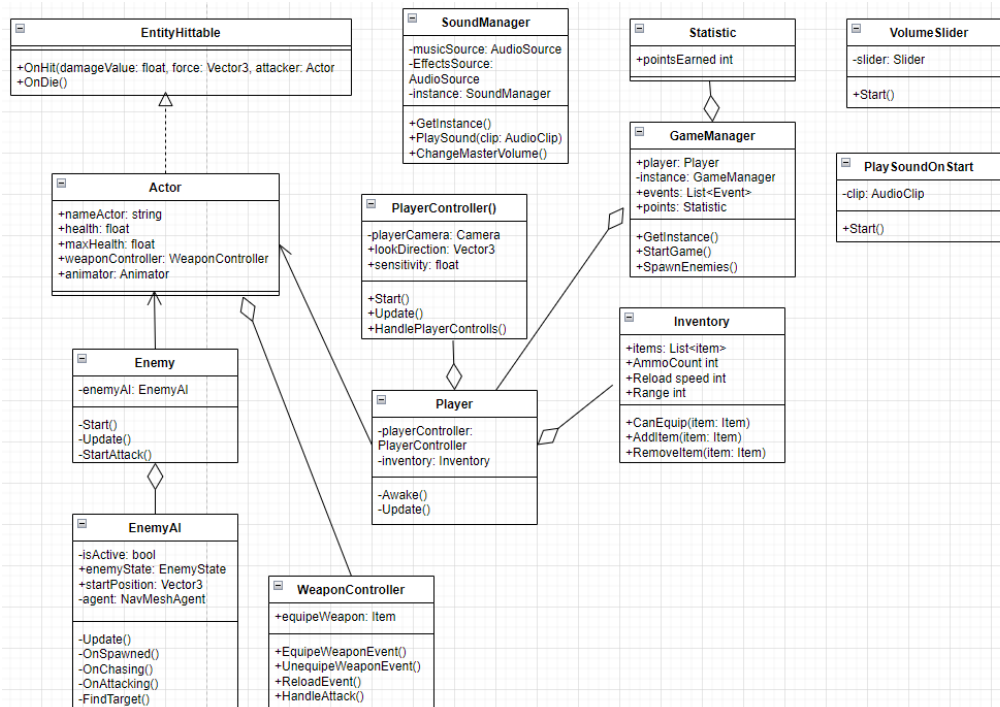


Рис.2.6 Діаграма класів

Actor – діючі персонажі на карті, а саме гравець та вороги. Має реалізацію інтерфейсу IEntityHittable, який має у собі операції для класу Actor.

IEntityHittable – інтерфейс, має операції які використовуються для об'єктів, які можуть отримувати шкоду, можуть руйнуватися та помирати у грі

Enemy – дочірній клас від Actor, клас супротивників, ворогів.

EnemyAI – клас, який містить у собі штучний інтелект ворога, усі його можливі дії та стани.

Player – дочірній клас від Actor, клас гравця, описує його логіку.

PlayerController – описує логіку керування гравця.

WeaponController – описує логіку керування зброєю для гравця і для ворога.

GameManager – описує логіку гри, реалізований як патерн одинак.

Inventory – клас інвентаря, містить у собі предмети гравця, які можна з нього екіпірувати.

SoundManager() – клас, який описує логіку звуків та музики у грі.

2.4 Взаємодія об'єктів системи

На даному етапі розглянемо взаємодію об'єктів системи за допомогою діаграм послідовностей.

Основна частина геймплею це взаємодія з ворогами. Тому перш за все слід розглянути її.

Зображено клас GameManager який запускає гру і генерує ворогів. Вороги до взаємодії з гравцем використовують методи з класу EnemyAI, які описують їх стартову позицію та далі патрулювання.

Гравець використовує своє управління, та виконує постріл зі зброї. При влучанні з ворога знімається здоров'я.

Логіка отримання шкоди у гравця і ворога також однакові, методи для цього беруться з інтерфейса класу Actor.

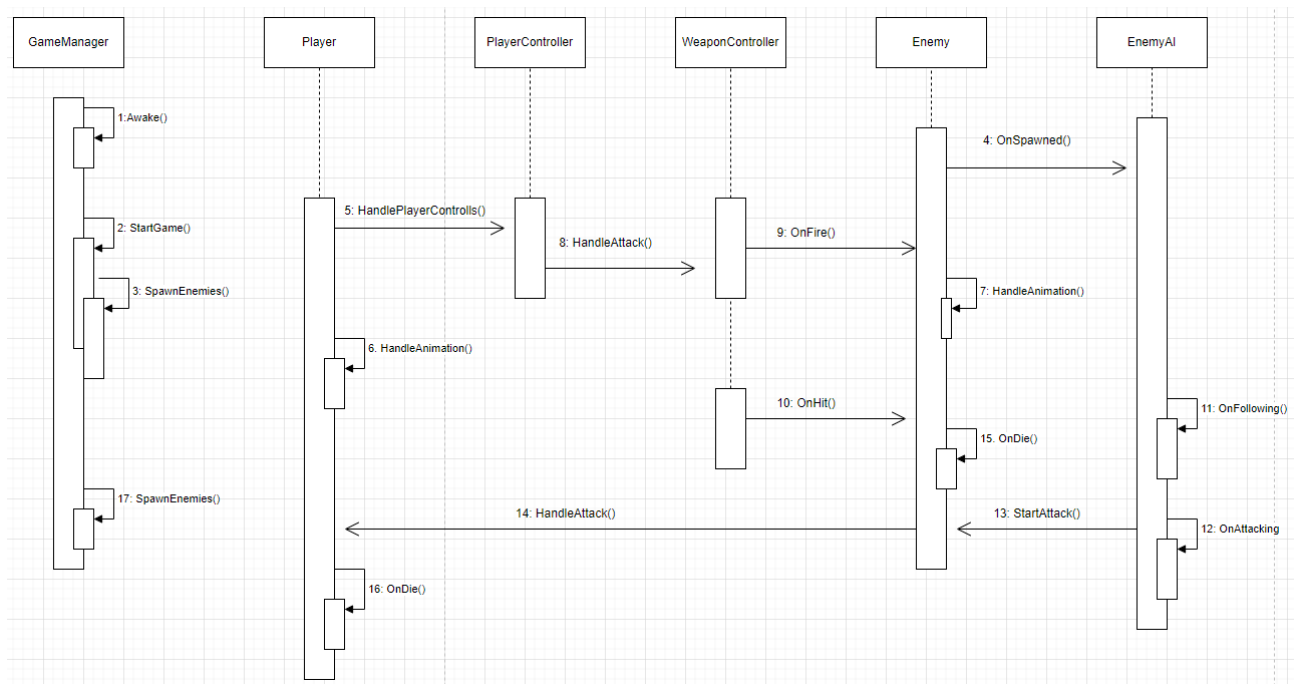


Рис.2.7 Діаграма послідовності взаємодії з ворогами

2.5 Взаємодія компонентів системи

Діаграма компонентів відображає залежності між компонентами програмного забезпечення. Модуль програмного забезпечення може бути представлено як компонент. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми.

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

На діаграмі присутні такі компоненти як: ігрові механіки гравця та інвентар, які мають прямий зв'язок з компонентом Гравця. Ігрові логіка – пов'язана з Гравцем та Ворогами. Ворог пов'язаний з Гравцем через бій

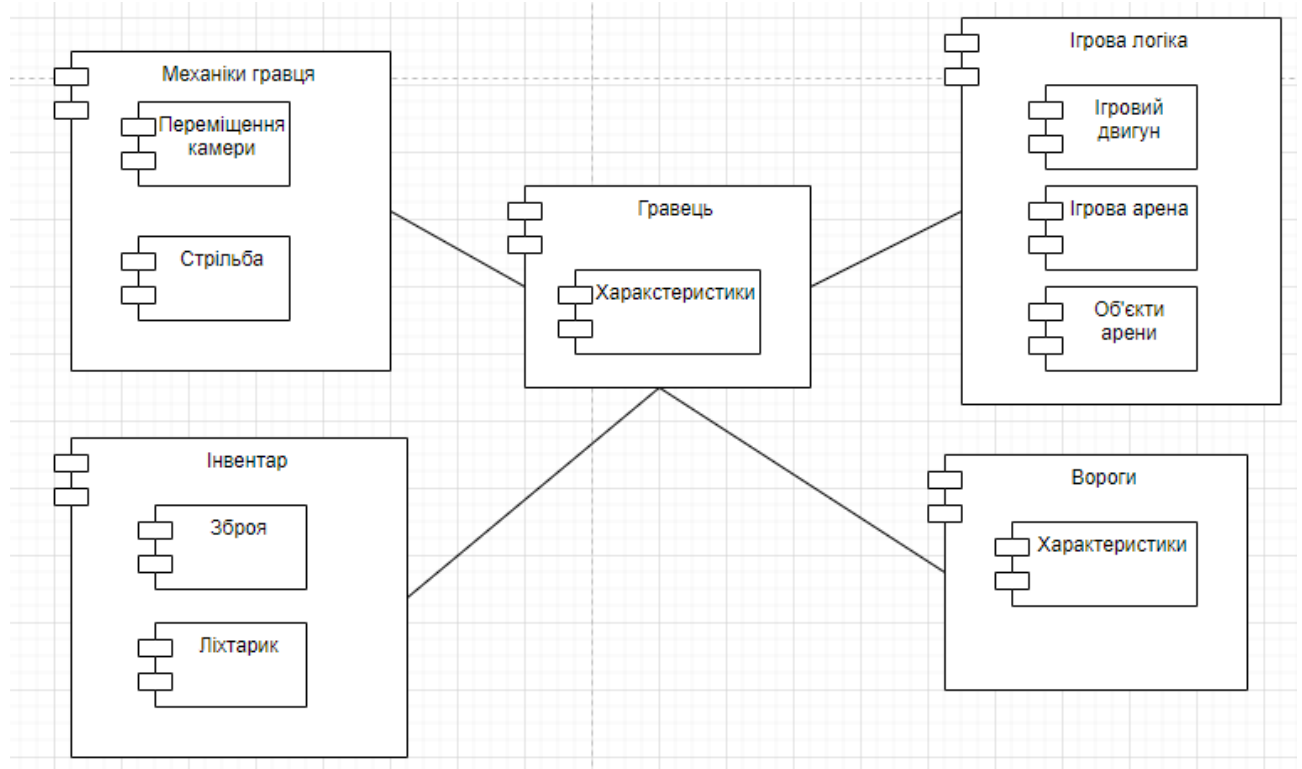


Рис.2.8 Діаграма компонентів

Діаграма розгортання - це тип діаграми UML, який показує архітектуру виконання системи, включаючи такі вузли, як середовища виконання апаратного чи програмного забезпечення, і проміжне програмне забезпечення, що їх з'єднує.

З діаграми видно, що починаємо з запуску гри на локальному комп'ютері, а саме game.exe, який в свою чергу включає в себе налаштування рушія, по суті ігровий двигун. Ігровий рушій зв'язаний з Game tools – конфігураційні файли та Game content – контент гри, який включає такі модулі як графіка, інтерфейс, аудіо частина та налаштування.

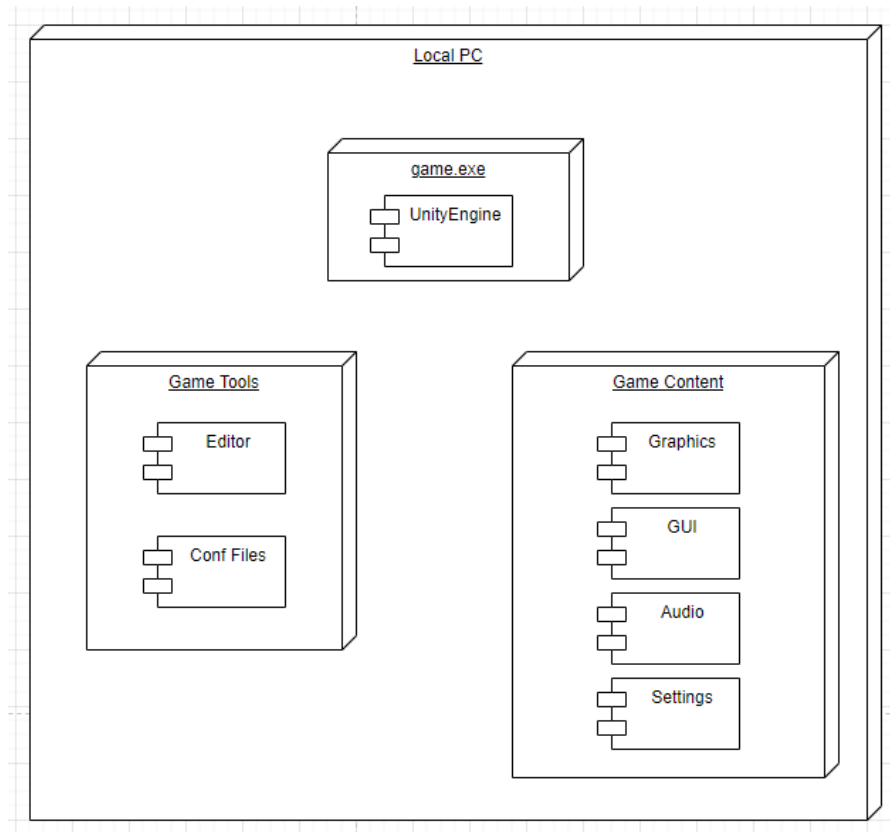


Рис.2.9 Діаграма розгортання

2.6 Реалізація ігрового проекту

Реалізація ігрового додатку зазвичай включає кілька етапів. Нижче наведено загальні етапи, які можуть виникнути під час розробки ігрового додатку:

1. Планування і концепція: На цьому етапі визначається загальна ідея гри, включаючи жанр, механіку гри, цільову аудиторію та історію. Також проводиться аналіз конкурентів і розробляється концепція гри.

2. Прототипування: Розробляється прототип гри, що демонструє основні механіки та функції гри. Це допомагає перевірити, наскільки вдалим є концепт гри та виявити можливі проблеми.

3. Дизайн гри: Розробляється дизайн гри, включаючи графіку, звукові ефекти, музику та інтерфейс користувача. Важливо створити привабливу та зручну графічну оболонку, яка буде відповідати концепції гри.

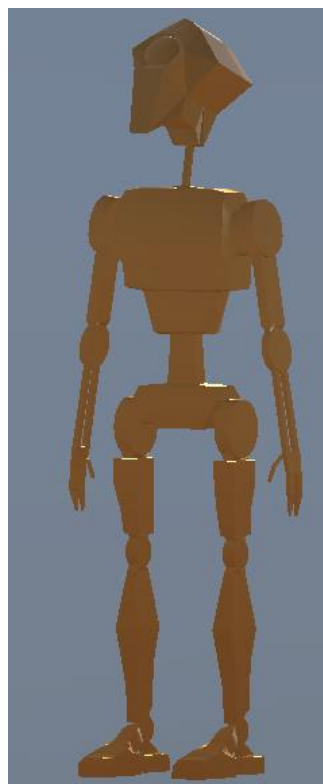
4. Розробка: Розроблюється програмне забезпечення гри. Розробники створюють логіку гри, програмують функції, оброблюють введення користувача та виводять необхідні візуальні та звукові ефекти.

5. Тестування: Проводяться внутрішні та зовнішні тестування гри для виявлення помилок, багів, неполадок та недоліків. Тестування також допомагає переконатися, що гра є веселою та викликає задоволення у гравців.

6. Випуск: Після успішного завершення тестування гра готова до випуску. Розробники виконують фінальні налаштування, пакують гру та розміщують її на відповідних платформах (наприклад, App Store, Google Play або інші цифрові ринки).

7. Підтримка та оновлення: Після випуску гри можуть з'являтися патчі, оновлення та виправлення помилок. Розробники надають підтримку гравцям, відповідають на їх запитання та забезпечують продовження розвитку гри.

Однією з основних задач було створення персонажів для гри. В результаті роботи було змодельовано 4 персонажі: 1 – персонаж гравця, 3 – персонажі вороги.



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

Рис. 2.10 Модель персонажу гравця



Рис. 2.11 Модель швидкого ворога

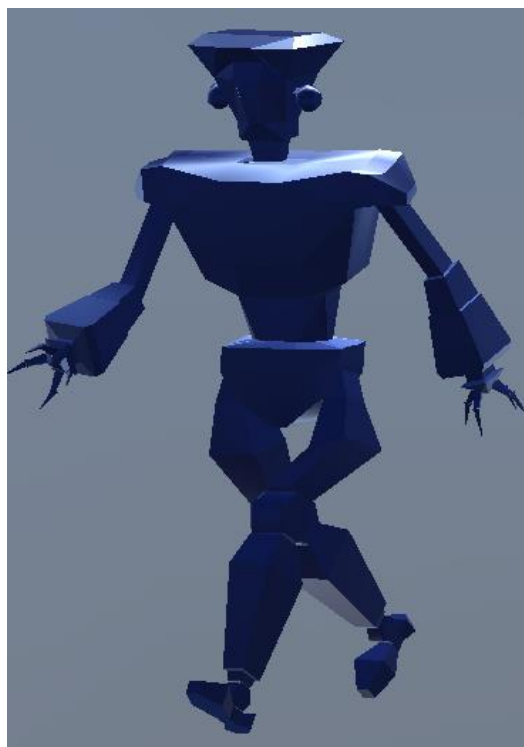


Рис. 2.12 Модель середнього ворога



Рис. 2.13 Модель важкого ворога

Також було використано 5 моделей зброї для використання гравцем:



Рис. 2.14 Модель зброї «Ніж»

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



Рис. 2.15 Модель зброї «Пістолет»



Рис. 2.16 Модель зброї «Штурмова гвинтівка»



Рис. 2.17 Модель зброї «Дробовик»



Рис. 2.18 Модель зброї «Снайперська гвинтівка»

Для кожного персонажа гри були створені свої власні аніматори, які відповідають за активацію і переключенням між анімаціями за певної умови

Аніматор для гравця включає в себе кілька гілок, які відповідають за анімації для кожної доступної для використання зброї (ніж, пістолет, штурмова гвинтівка, дробовик, снайперська гвинтівка).

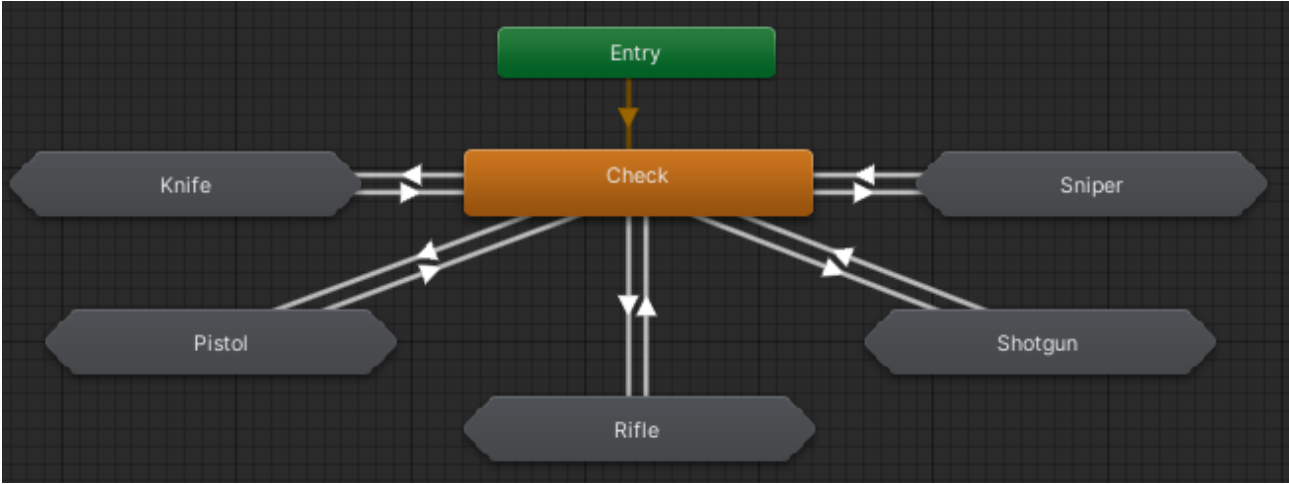


Рис. 2.19 Аніматор персонажу гравця

В свою чергу, кожна гілка має в собі ряд анімацій (анімація бездіяння, дістати зброю, сховати зброю, вистрілити зі зброї, перезарядити зброю)

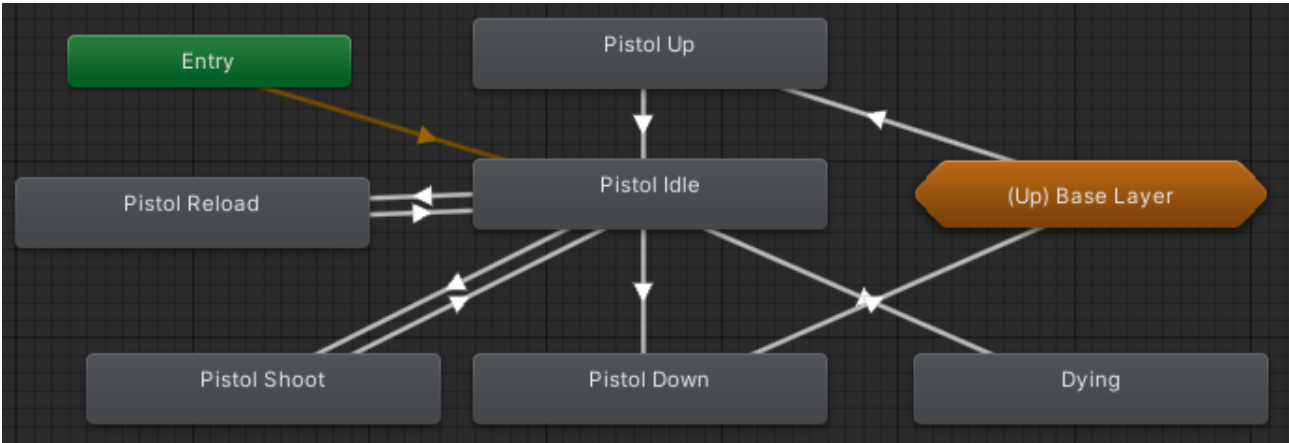


Рис. 2.20 Гілка з анімаціями для Пістолета в аніматорі персонажу гравця

Анімація ворогів включає у себе анімацію бездіяння, анімацію переміщення, анімацію атаки, анімацію вмирання.

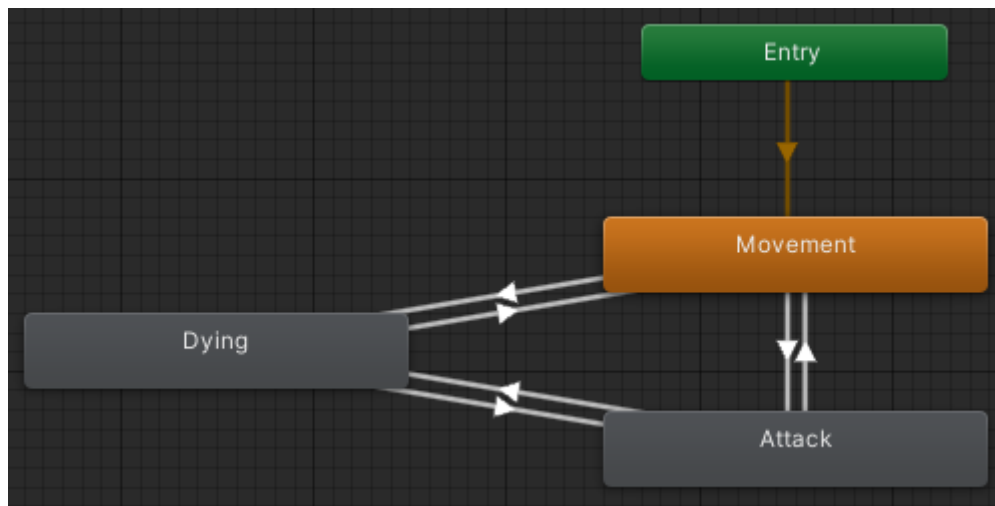


Рис. 2.21 Аніматор для ворогів

При появі на арені, ворог переходить у анімацію переміщення. Зміна анімації залежить від швидкості переміщення ворога. Дійшовши і зону зупинки, яка визначається у компоненті NavMeshAgent, швидкість ворога стає 0, що сприяє переходу у анімацію бездіяння. В свою чергу перехід з анімації бездіяння у анімацію атаки відбувається із заданою для кожного ворога затримкою. Після атаки ворог повертається в анімацію бездіяння і цикл повторюється.

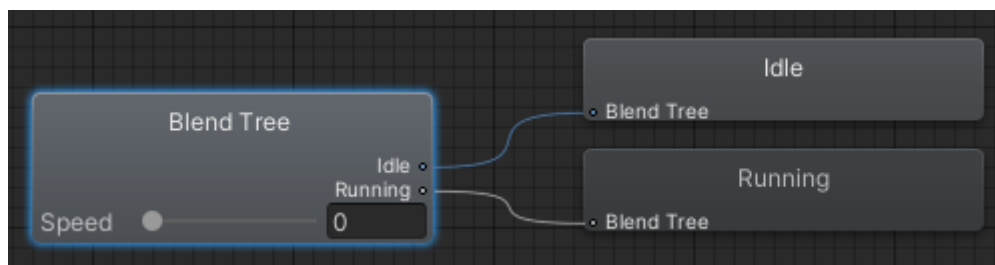


Рис. 2.22 Гілка Movement в аніматорі для ворогів

В момент, коли кількість очок здоров'я ворога дійде до 0, почнеться анімація вмирання після якої об'єкт «ворог» зникне з арені.

Для реалізацій механік, пов'язаних з переміщенням камери для прицілювання було реалізовано скрипт CameraController, який включає:

Таблиця 2.1 Опис вмісту скрипту CameraController

Назва	Пояснення
float mouseSensitivity	Поле, яке зберігає значення чутливості миші при її переміщенні
Transform arms	Поле, яке зберігає позицію моделі персонажа гравця, для її коректного обертання при переміщенні камери
Transform body	Поле, яке зберігає позицію об'єкту «гравець», для його коректного обертання при переміщенні камери
void Start()	Метод, який викликає метод LockCursor
void Update()	Метод, який викликає і оновлює метод HandleMouseLook
void HandleMouseLook()	Метод, який відповідає за переміщення камери за рухами мишки. Враховує рухи по осям X та Y, а також значення mouseSensitivity
void LockCursor()	Метод, який використовується для приховання курсору миші, коли викликаний
void UnlockCursor()	Метод, який використовується для показування курсору миші, коли викликаний

Для реалізації механік пов'язаних з характеристиками персонажів, було створено скрипт CharacterStats, а також скрипти, які наслідують його - PlayerStats, PlayerStats

Таблиця 2.2 Опис вмісту скрипту CharacterStats

Назва	Пояснення
int health	Поле, яке відповідає за теперішні очки здоров'я персонажа
int maxHealth	Поле, яке відповідає за максимальні очки здоров'я персонажа
bool isDead	Поле, яке приймає значення true
void Start	Метод, який викликає метод InitVariables
void CheckHealth()	Метод, який перевіряє значення теперішнього здоров'я і викликає метод Die у разі, якщо здоров'я персонажа менше 0
void Die()	Метод, який визначає значення isDead як true і викликається у разі смерті персонажа
void IsDead()	Метод, який повертає значення поля isDead
void SetHealthTo(healthToSetTo)	Метод, який прирівнює значення теперішніх очок здоров'я, health, до значення максимальних очок здоров'я, maxHealth
void TakeDamage(int damage)	Метод, який обраховує теперішнє значення очок здоров'я, health, відповідно до отриманих очок шкоди
void InitVariables()	Метод, який ініціалізує значення полів maxHealth, isDead. Також викликає метод SetHealthTo

Таблиця 2.3 Опис вмісту скрипту PlayerStats

Назва	Пояснення
void CheckHealth()	Метод, який викликає метод CheckHealth зі скрипту CharacterStats. Також викликає метод UpdateHealth зі скрипту PlayerHUD, в який передає значення теперішніх очок здоров'я і максимальних очок здоров'я.
void Die()	Метод, який викликає метод Die зі скрипту CharacterStats. Також викликає метод SetActiveHUD зі скрипту PlayerHUD, в який передає значення false. Відбувається програвання анімації вмирання

Таблиця 2.4 Опис вмісту скрипту EnemyStats

Назва	Пояснення
int enemyHealth	Поле, яке містить інформацію про очки здоров'я ворога
int enemyDamage	Поле, яке містить інформацію про очки шкоди, які нанесе ворог під час атаки
float enemyAttackSpeed	Поле, яке містить інформацію про швидкість атаки ворога
void Start()	Метод, який викликає метод InitVariables

void DealDamage(CharacterStats statsToDamage)	Метод, який відповідає за зняття очок здоров'ю гравця, відповідно до кількості очок шкоди ворога
void InitVariables()	Метод, який ініціалізує характеристики ворога, очки здоров'я, очки шкоди, швидкість атаки
void Die()	Метод, який викликає метод Die в скрипті CharacterStats, метод UpdateScoreAmount у скрипті PlayerHUD. Також запускає анімацію вмирання ворога

Для реалізації основних механік гравця пов'язаних зі зброєю було створено низку методів – Weapon, Inventory, EquipmentManager, WeaponShooting

Таблиця 2.5 Опис вмісту скрипту Weapon

int damage	Поле, яке відповідає за кількість одиниць шкоди, яку буде нанесено ворогу у разі попадання за один вистріл
int magazineSize	Поле, яке відповідає за кількість патронів у магазині, який використовується
int storedAmmo	Поле, яке відповідає за максимальну кількість патронів. Звертаючись до цього поля, ми будемо поновлювати кількість патронів при перезарядці відповідно до заданого значення
float fireRate	Поле, яке відповідає за скорострільність зброї, тобто за

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

	кількість вистрілів за певний проміжок
float range	Поле, яке відповідає за максимальну дальність атаки, на якій ворогам може бути зараховано пошкодження від даної зброї

Таблиця 2.6 Опис вмісту скрипту EquipmentManager

int currentlyEquippedWeapon	Поле, яке зберігає значення зброї в масиві Weapon. Саме цю зброю буде екіпіровано при початку гри
void Update()	Метод, який змінює теперішню екіпіровану зброю, відповідно до натиснутої клавіші. Викликає методи UnequipWeapon і EquipWeapon
void UnequipWeapon(Weapon weapon)	Метод, який знищує теперішню екіпіровану зброю, при екіпіруванні іншої зброї. Викликається внаслідок спрацювання Event у відповідній анімації деікіпірування зброї
void EquipWeapon()	Метод, який створює зброю в руках гравця, при екіпіруванні даної зброї. Спрацьовує після методу UnequipWeapon

Таблиця 2.7 Опис вмісту скрипту Inventory

Weapon []	Масив, який зберігає доступну для використання зброю
-----------	--

Weapon GetItem(int index)	Метод, який відповідає за повернення значення зброї з масиву Weapon
---------------------------	---

Для реалізацію механік ворогів були створені наступні скрипти: EnemyController, EnemyStats, EnemySpawner

Таблиця 2.8 Опис вмісту скрипту EnemyController

float timeOfLastAttack	Поле, яке використовується для врахування часу до початку наступної атаки ворога
bool hasStopped	Поле, яке використовується для перевірки чи ворог зупинився для початку атакування гравця
void Update()	Метод, який відповідає за виклик методу MoveToTarget
void MoveToTarget()	Метод, який скеровує ворога до координат гравця. Запускає анімацію переміщення ворога
void RotateToTarget()	Метод, який розвертає ворога до гравця під час переміщення
void AttackTarget(Character Stats)	Метод, який використовується для нанесення очок шкоди гравцю під час атаки ворога. Запускає анімацію атаки ворога

Таблиця 2.9 Опис вмісту скрипту EnemySpawner

Transform[] spawners	Масив, який зберігає точки появи ворогів на ігровій арені
----------------------	---

Trasnform[] enemyRobot	Масив, який зберігає ворогів, які будуть з'являтися з точок появи на ігровій арені
void SpawnEnemy()	Метод, який обирає випадкову точку появи ворога з масиву spawners на ігровій арені, а також ворога р vfc bde enemyRobot, який з'явиться з цієї точки

Для реалізації стрільби було створено скрипт WeaponShooting, який включає в себе:

Таблиця 2.10 Опис вмісту скрипту WeaponShooting

Назва	Пояснення
float lastShootTime	Поле, яке використовується для обрахунку часу від останнього пострілу
bool canShoot	Поле, яке використовується для перевірки можливості зробити вистріл в даний момент
bool canRealod	Поле, яке використовується для перевірки можливості зробити перезарядку зброї в даний момент
int CurrentAmmo	Поле, яке зберігає інформацію про поточну кількість патронів в магазині в екіпірованій зброї
int CurrentAmmoStorage	Поле, яке зберігає інформацію про максимальну кількість патронів в магазині в екіпірованій зброї

bool MagazineEmpty	Поле, яке використовується для перевірки наявності патронів в магазині
GameObject hitParticles	Поле, яке приймає об'єкт hitParticles, які з'являються при попаданні при вистрелі по ворогу
void Start()	Метод, який присвоює полям canShoot і canReload значення true
void Update()	Метод, який перевіряє, чи персонаж гравця живий і в такому випадку при натисканні лівої клавіші миші викликається метод Shoot і здійснюється вистріл; при натисканні кнопки R викликається метод Reload і здійснюється перезарядка
void RaycastShoot(Weapon currentWeapon)	Метод, який відповідає за появу луча із визначеної точки екрану, який при потраплянні в об'єкт слою «Enemy» викликає метод TakeDamage і передає в нього кількість очок шкоди поточної зброї, які будуть зняті від очок здоров'я ворога по якому було здійснено попадання. Також викликається метод SpawnHitParticles і на місці попадання з'являться hitParticles
void Shoot()	Метод, який викликає метод CheckCanShoot для перевірки можливості вистрілити в даний момент, після чого викликається

	метод RaycastShoot, UseAmmo, PlayShootAnimation
void UseAmmo(int slot, int currentAmmoUsed, int currentStoredAmmoUsed)	Метод, який відповідає за використання наявних патронів в магазині під час стрільби. Також викликається метод UpdateWeaponAmmoUI скрипту PlayerHUD для оновлення інформації про наявні патрони і оновлення інформації про них в графічному інтерфейсі
void AddAmmo(int slot, int currentAmmoAdded, int currentStoredAmmoAdded)	Метод, який присвоює полю CurrentAmmo значення CurrentAmmoStored, поточної зброї
void Reload(slot)	Метод, який перевіряє можливість перезарядки в даний момент, після чого викликає методи AddAmmo, UseAmmo, CheckCanShoot при натисканні кнопки «R». Також проходить анімація перезарядки поточної зброї
void CheckCanShoot(int slot)	Метод, який перевіряє наявність патронів в магазині, присвоюючи при цьому значення true або false полю canShoot
void InitAmmo(int slot, Weapon weapon)	Метод, який ініціалізує змінні CurrentAmmo і CurrentAmmoStorage відповідними значеннями які містить поточна зброя

PlayShootAnimation()	Метод, який запускає анімацію перезарядки поточної зброї
SpawnHitParticles(Vector3 position, Vector3 normal)	Метод, який відповідає за появу hitParticles

Для реалізації графічного інтерфейсу гравця, було створено скрипти ProgressBar, WeaponUI, PlayerHUD

Таблиця 2.11 Опис вмісту скрипту ProgressBar

float baseValue	Поле, яке зберігатиме теперішнє значення для заливки для шкал відображення очок здоров'я і рівня заряду ліхтарика
float maxValue	Поле, яке зберігатиме максимальне значення для заливки для шкал відображення очок здоров'я і рівня заряду ліхтарика
Image fill	Поле, яке зберігає зображення, яке використовується для заливки внутрішньої області шкал відображення очок здоров'я і рівня заряду ліхтарика
Text amount	Поле, яке зберігатиме значення теперішнього значення очок здоров'я у текстовому варіанті
void SetValues(float _baseValue, float _maxValue)	Метод, який слугуватиме ініціалізації поточних, а також максимальних значень очок здоров'я гравця і рівня заряду ліхтаря
void CalculateFillAmount()	Метод, який слугуватиме для обчислення значення теперішнього

	здоров'я на шкалі здоров'я гравця, а також для обчислення теперішнього рівню заряду ліхтарика у відношенні: теперішнє значення до максимального значення
--	--

Таблиця 2.12 Опис вмісту скрипту WeaponUI

Image icon	Поле, яке зберігає іконку зброї
Text magazineSizeText	Поле, яке зберігає інформацію про теперішню кількість патронів в магазині зброї у текстовому варіанті
Text storedAmmoText	Поле, яке зберігає інформацію про максимальну кількість патронів в магазині зброї у текстовому варіанті
void UpdateInfo (Sprite weaponIcon, int magazineSize, int storedAmmo)	Метод, який використовується для визначення потрібної іконки, теперішнього магазину і максимальної кількості патронів в магазині для теперішньої зброї
void UpdateWeaponUI(int magazineSize, int storedAmmo)	Метод, який використовується для визначення потрібної теперішнього магазину і максимальної кількості патронів в магазині для теперішньої зброї для показу у графічному інтерфейсі

Таблиця 2.13 Опис вмісту скрипту PlayerHUD

Назва	Пояснення
void UpdateHealth(int currentHealth, int maxHealth)	Метод, який викликає метод SetValues зі скрипта ProgressBar і передає значення int currentHealth, int maxHealth
void UpdateFlashLight(float currentCharge, float maxCharge)	Метод, який викликає метод SetValues зі скрипта ProgressBar і передає значення float currentCharge, float maxCharge
void UpdateWeaponUI(Weapon newWeapon)	Метод, який викликає метод UpdateInfo зі скрипту WeaponUI і передає дані про іконку, початкову кількість патронів в магазині і максимальну кількість патронів в магазині для екіпірованої зброї
void UpdateWeaponAmmoUI(int currentAmmo, int storedAmmo)	Метод, який викликає метод UpdateWeaponUI і передає теперішню кількість патронів в магазині і максимальну кількість патронів в магазині для екіпірованої зброї для відображення цієї інформації у графічному інтерфейсі
void UpdateScoreAmount()	Метод, який викликає метод AddToScore зі скрипту Score, щоб оновити кількість набраних очок і відобразити зміни в графічному інтерфейсі

Висновки до другого розділу

У ході виконання другого розділу були розібрані варіанти використання, сформовані алгоритми роботи та стани програмної системи, зроблені необхідні діаграми активності. Було розроблено діаграму класів, описано всі атрибути та операції.

Були сформовані діаграми послідовності, визначено взаємодію між компонентами системи

Було створено 3D моделі, а також налаштовані анімації для них. Були описані скрипти, які реалізують основні механіки гри

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ “From The Darkness”

3.1. Порядок встановлення та налаштування системи

“From the Darkness” – відеогра, розроблена на платформі Unity 3D. Після випуску релізної версії, гру можна буде завантажити на пристроях підтримуваних платформ. Платформа, для якої створювалась гра – Windows.

Мінімальні вимоги для запуску програмного продукту:

- ОС: Windows 7 або вище
- Процесор: 2.6 GHz Dual Core або схожий
- Оперативна пам'ять: 4 GB ОП
- Відеокарта: GeForce GTX 500 серії або схожа
- DirectX: версії 11
- Місце на диску: 1 GB доступного місця

3.2. Структура інтерфейсу

Після запуску гри, гравця зустріне головне меню з кількома опціями на вибір.

Кнопка “Start Game” – запустить основну геймплейну сцену

Кнопка “How to Play” – запустить меню, в якому будуть описані правила і поради щодо гри

Кнопка “Options” – перенесе в меню налаштувань, де можна налаштувати гру

Кнопка “Quit Game” – припинить роботу ігрового додатку

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

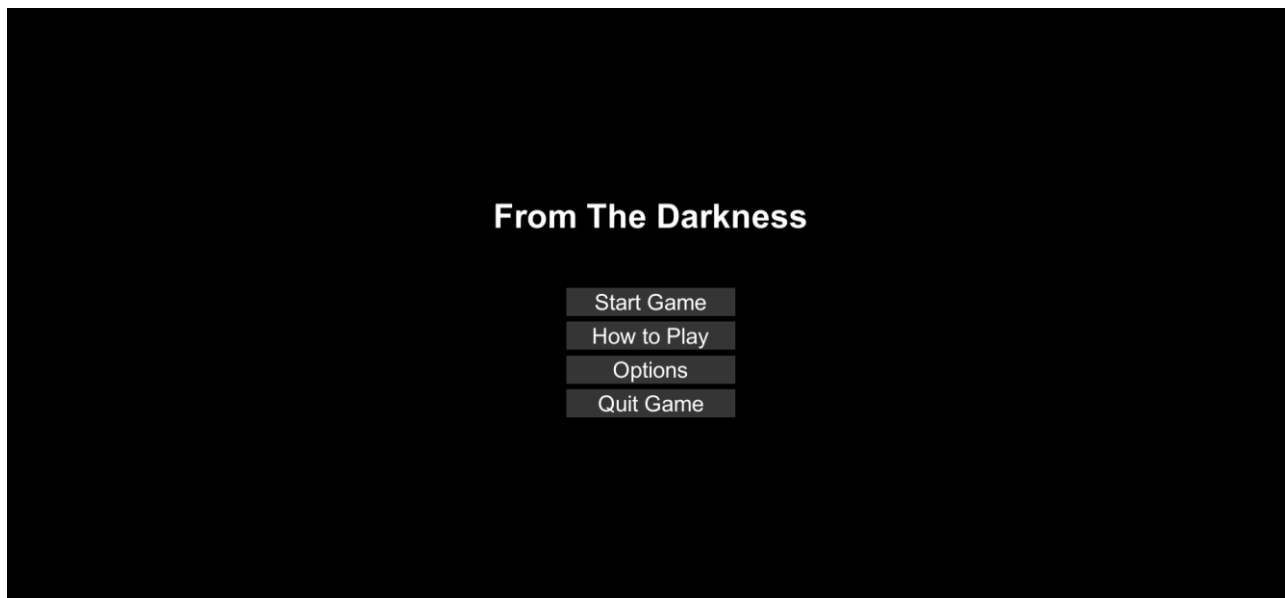


Рис. 3.1 Головне меню гри

Меню налаштувань гри включає у себе можливість змінити ежим відображення з повноекранного на віконний або навпаки (перша опція); змінити якість зображення (друга опція); змінити гучність звуків гри (третя опція).

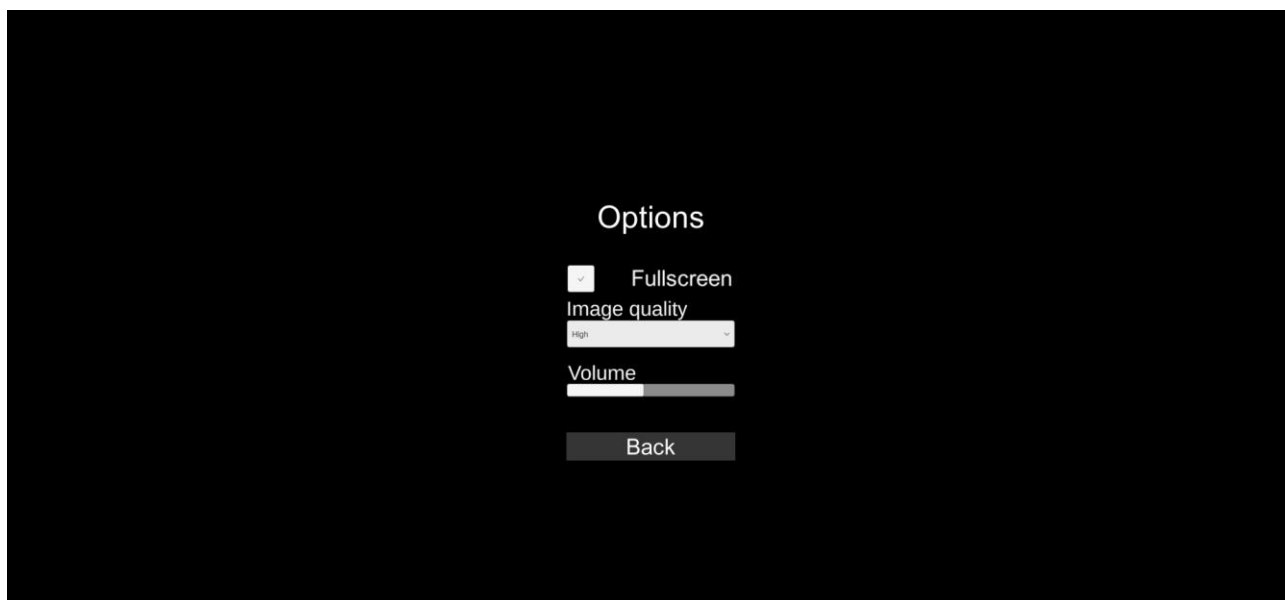


Рис. 3.2 Меню налаштувань гри

В самій грі ми можемо побачити нашого персонажа зліва екрану, а також графічний інтерфейс, який включає в себе зароблені за вбивство ворогів очки (зверху екрану), шкала очок здоров'я гравця (знизу зліва), іконка поточної зброї,

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

а також її боезапас (справа знизу), шкала заряду ліхтарика (справа зверху), приціл (центр екрану)

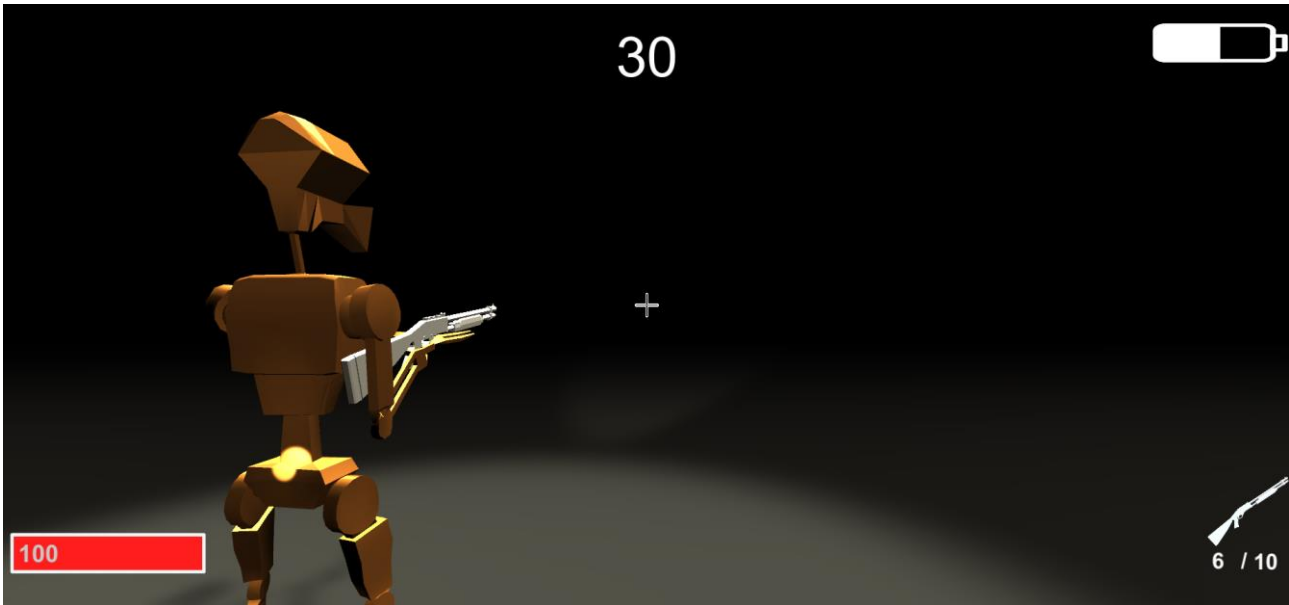


Рис. 3.3 Загальний вигляд гри та елементів інтерфейсу

При здійсненні вистрілу вогнепальною зброєю, з’явиться ефект вистрілу, а при попаданні по ворогу з’являться іскри.

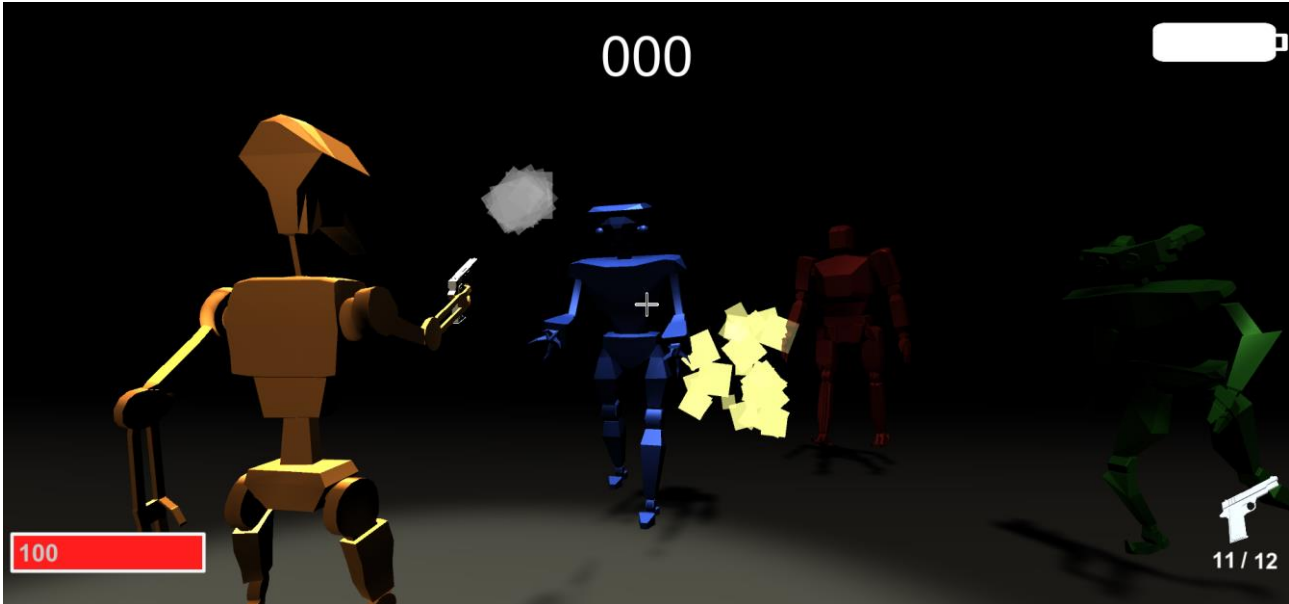


Рис. 3.5 Димові часточки при вистрілі з Пістолету і іскри від попадання по ворогу

Використання ліхтаря дає можливість гравцю знаходити ворогів на більш далекій дистанції. При цьому ліхтар витрачає свій заряд і стає менш яскравим. Вимкнення ліхтаря дає можливість йому перезарядитися



Рис. 3.6 Використання ліхтаря

В процесі гри у гравця є можливість викликати меню паузи, натиснувши кнопку «Escape». Дане меню дозволить гравцю перезапстити гру кнопкою “Restart”, вийти до головного меню кнопкою “Main Menu”, вийти з гри кнопкою “Quit Game”. Якщо гравець хоче продовжити гру, достатньо ще раз натиснути кнопку “Escape”



Рис. 3.4 Меню паузи

Якщо очки здоров'я граця будуть рівні 0, з'явиться кінцеве меню гри. Дане меню відображатиме кількість очок, набраних за період ігрового сеансу, а також надасть декілька варіантів на вибір: перезапуск гри кнопкою “Restart”, вихід до головного меню кнопкою “Main Menu”, вихід з гри кнопкою “Quit Game”

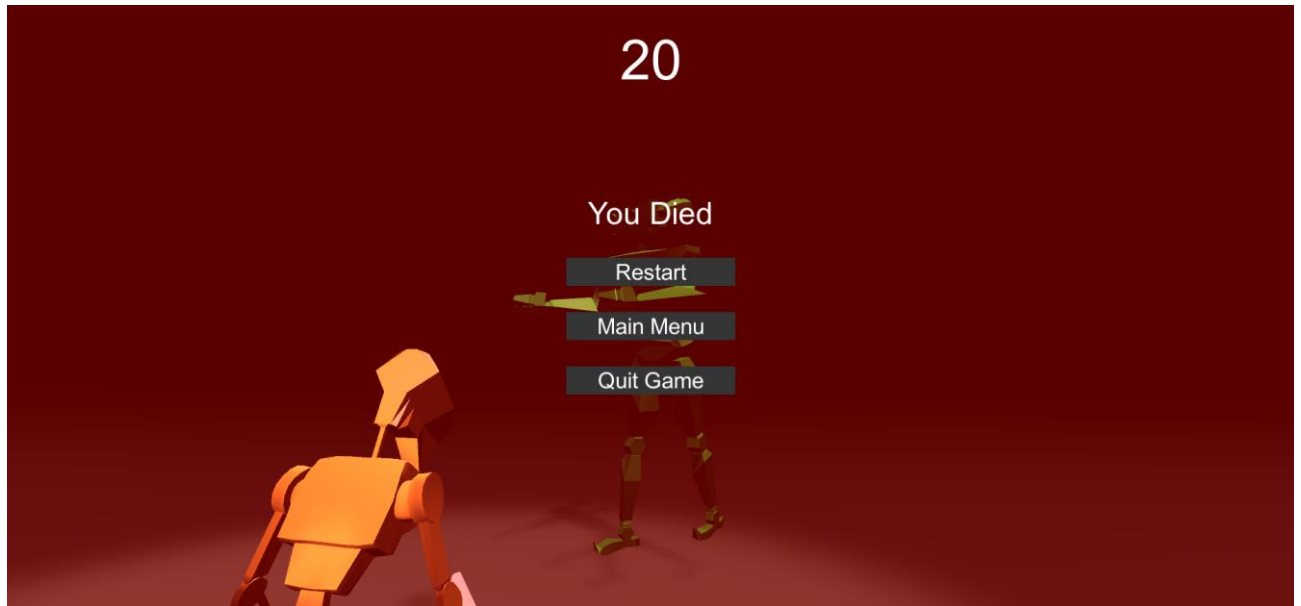


Рис. 3.5 Кінцеве меню гри

3.3. Тестування роботи системи

Таблиця 3.1 Чек ліст перевірки роботи головного меню

Назва кроку	Статус	Коментар
Натиснути кнопку Start Game	Пройшов	Кнопка натискається. Відбувається перехід на ігрову сцену
Натиснути кнопку Options	Пройшов	Кнопка натискається. Відбувається перехід в меню Options
Натиснути кнопку How to Play	Пройшов	Кнопка натискається. Відбувається перехід на сторінку з правилами і порадами для гри

Натиснути кнопку Quit Game	Пройшов	Кнопка натискається. Відбувається закриття ігрового додатку
----------------------------	---------	--

Таблиця 3.2 Чек ліст перевірки роботи меню налаштувань

Назва кроку	Статус	Коментар
Зняти галочку в боксі Fullscreen	Пройшов	Галочка знялася. Гра змінила відображення на віконне
Поставити галочку в боксі Fullscreen	Пройшов	Галочка поставилась. Гра зміна відображення на повноекранний режим

Таблиця 3.3 Чек ліст перевірки роботи меню паузи

Назва кроку	Статус	Коментар
Натиснути кнопку Escape на клавіатурі перебуваючи в грі	Пройшов	Відбувається зупинка гри і перехід в меню паузи
Натиснути кнопку Escape на клавіатурі перебуваючи в мен	Пройшов	Відбувається продовження гри
Натиснути кнопку Restart	Пройшов	Кнопка натискається. Перезапуск ігрової сцени
Натиснути кнопку Main Menu	Пройшов	Кнопка натискається. Відбувається перехід на сцену Головного меню

Натиснути кнопку Quit Game	Пройшов	Кнопка натискається. Відбувається закриття ігрового додатку
----------------------------	---------	---

Таблиця 3.4 Чек ліст перевірки роботи ігрової камери

Назва кроку	Статус	Коментар
Провести мишкою горизонтально по осі X	Пройшов	Камера повернулася в сторону відповідно до руху мишки
Провести мишкою горизонтально по осі Y	Пройшов	Камера повернулася в сторону відповідно до руху мишки
Провести мишкою горизонтально по осям X + Y	Пройшов	Камера повернулася в сторону відповідно до руху мишки

Таблиця 3.5 Чек ліст перевірки роботи зміни зброї в грі і відповідних анімацій

Назва кроку	Статус	Коментар
Натискання клавіші 1 на клавіатурі	Пройшов	Відбулося ховання попередньої зброї разом з анімацією, зброя змінилася на Пістолет з відповідною анімацією дістання
Натискання клавіші 2 на клавіатурі	Пройшов	Відбулося ховання попередньої зброї разом з анімацією, зброя змінилася на Штурмову

		гвинтівку з відповідною анімацією дістання
Натискання клавіші 3 на клавіатурі	Пройшов	Відбулося ховання попередньої зброї разом з анімацією, зброя змінилася на Дробовик з відповідною анімацією дістання
Натискання клавіші 4 на клавіатурі	Пройшов	Відбулося ховання попередньої зброї разом з анімацією, зброя змінилася на Снайперську гвинтівку з відповідною анімацією дістання
Натискання клавіші 5 на клавіатурі	Пройшов	Відбулося ховання попередньої зброї разом з анімацією, зброя змінилася на Ніж з відповідною анімацією дістання

Таблиця 3.6 Чек ліст роботи ліхтаря

Назва кроку	Статус	Коментар
Увімкнення ліхтаря клавішою F на клавіатурі	Пройшов	Ліхтар увімкнувся, заряд почав зменшуватись, а з ним і яскравість
Вимкнення ліхтаря клавішою F на клавіатурі	Пройшов	Ліхтар вимкнувся, заряд почав збільшуватись, а з ним і яскравість

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.7 Чек ліст перевірки роботи стрільби/ударів відповідною зброєю

Назва кроку	Статус	Коментар
Натискання лівої кнопки миші з поточною зброєю Ніж	Пройшов	Відбулася відповідна анімація удару Ножа
Натискання лівої кнопки миші з поточною зброєю Ніж, націлившись на ворога	Пройшов	Відбулася відповідна анімація удару Ножа, в місці попадання з'явилися hitParticles, ворог отримав пошкодження
Натискання лівої кнопки миші з поточною зброєю Пістолет	Пройшов	Відбулася відповідна анімація стрільби з Пістолета
Натискання лівої кнопки миші з поточною зброєю Пістолет, націлившись на ворога	Пройшов	Відбулася відповідна анімація стрільби з Пістолета, в районі ствола зброї з'явилися shootParticles, в місці попадання з'явилися hitParticles, ворог отримав пошкодження
Натискання лівої кнопки миші з поточною зброєю Пістолет з повністю порожнім магазином	Пройшов	Стрільба не відбулася, магазин порожній
Натискання лівої кнопки миші з поточною зброєю Штурмова гвинтівка	Пройшов	Відбулася відповідна анімація стрільби з Штурмової гвинтівки

Натискання лівої кнопки миші з поточною зброєю Штурмова гвинтівка, націлившись на ворога	Пройшов	Відбулася відповідна анімація стрільби з Штурмової гвинтівки, в районі ствола зброї з'явилися shootParticles, в місці попадання з'явилися hitParticles, ворог отримав пошкодження
Натискання лівої кнопки миші з поточною зброєю Штурмова гвинтівка з повністю порожнім магазином	Пройшов	Стрільба не відбулася, магазин порожній
Натискання лівої кнопки миші з поточною зброєю Дробовик	Пройшов	Відбулася відповідна анімація стрільби з Дробовика
Натискання лівої кнопки миші з поточною зброєю Дробовик, націлившись на ворога	Пройшов	Відбулася відповідна анімація стрільби з Дробовика, в районі ствола зброї з'явилися shootParticles, в місці попадання з'явилися hitParticles, ворог отримав пошкодження
Натискання лівої кнопки миші з поточною зброєю Дробовик з повністю порожнім магазином	Пройшов	Стрільба не відбулася, магазин порожній

Натискання лівої кнопки миші з поточною зброєю Снайперська гвинтівка	Пройшов	Відбулася відповідна анімація стрільби з Снайперської гвинтівки
Натискання лівої кнопки миші з поточною зброєю Снайперська гвинтівка, націлившись на ворога	Пройшов	Відбулася відповідна анімація стрільби з Снайперської гвинтівки, в районі ствола зброї з'явилися shootParticles, в місці попадання з'явилися hitParticles, ворог отримав пошкодження
Натискання лівої кнопки миші з поточною зброєю Снайперська гвинтівка з повністю порожнім магазином	Пройшов	Стрільба не відбулася, магазин порожній

Таблиця 3.8 Чек лісти перевірки роботи перезарядки зброї

Назва кроку	Статус	Коментар
Перезарядка зброї Ніж	Пройшов	Перезарядка не відбулася, Ніж не використовує набої
Перезарядка зброї Пістолет з повним магазином	Пройшов	Перезарядка не відбулася, магазин зброї повний
Перезарядка зброї Пістолет з частково повним/повністю пустим магазином	Пройшов	Перезарядка відбулася, кількість патронів в магазині повенулася до початково значення,

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

		відбулася анімація перезарядки Пістолету
Перезарядка зброї Штурмова гвинтівка з повним магазином	Пройшов	Перезарядка не відбулася, магазин зброї повний
Перезарядка зброї Штурмова гвинтівка з частково повним/повністю пустим магазином	Пройшов	Перезарядка відбулася, кількість патронів в магазині повенулася до початково значення, відбулася анімація перезарядки Штурмової гвинтівки
Перезарядка зброї Дробовик з повним магазином	Пройшов	Перезарядка не відбулася, магазин зброї повний
Перезарядка зброї Дробовик з частково повним/повністю пустим магазином	Пройшов	Перезарядка відбулася, кількість патронів в магазині повенулася до початково значення, відбулася анімація перезарядки Дробовика
Перезарядка зброї Снайперська гвинтівка з повним магазином	Пройшов	Перезарядка не відбулася, магазин зброї повний
Перезарядка зброї Снайперська гвинтівка з частково повним/повністю пустим магазином	Пройшов	Перезарядка відбулася, кількість патронів в магазині повенулася до початково значення, відбулася анімація

		перезарядки Снайперської гвинтівки
--	--	---------------------------------------

Таблиця 3.9 Чек лісти перевірки роботи ворогів

Назва кроку	Статус	Коментар
Поява різних ворогів у різних точках арени	Пройшов	Різні вороги з'являються в різних точках появи із заданим часовим інтервалом
Переслідування персонажа гравця з відповідною анімацією	Пройшов	Після появи вороги слідують до точки, в якій знаходиться персонаж гравця, супроводжуючи це відповідною анімацією переміщення
Зупинка ворогів поблизу персонажу гравця	Пройшов	Вороги зупиняються в певному радіусі від гравця і переходять в анімацію бездіяння
Атака ворогів по гравцю	Пройшов	Після анімації бездіяння слідує атака ворога з відповідною анімацією атаки. Відбувається зменшення очок здоров'я гравця
Атака гравця по ворогу	Пройшов	Атака гравця по ворогу сприяє появі hitParticles, ворог втрачає здоров'я після кожного попадання. Коли

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

		здоров'я ворога досягає нуля, проходить анімація помирання і ворог зникає з арени
--	--	---

Таблиця 3.10 Чек ліст роботи графічного інтерфейсу

Назва кроку	Статус	Коментар
Зміна шкали очок здоров'я гравця	Пройшов	Після отримання шкоди від ударів ворогів, шкала здоров'я гравця зменшилася
Зміна очок за перемогу над ворогами	Пройшов	Після вбивста ворога, кількість зароблених очок збільшилось
Зменшення шкали заряду увімкненого ліхтарика	Пройшов	Після увімкнення ліхтаря, шкала заряду почала зменшуватись
Збільшення заряду вимкненого ліхтаря	Пройшов	Після вимкнення ліхтаря, шкала заряду почала збільшувались

Таблиця 3.11 Чек ліст перевірки меню завершення гри

Назва кроку	Статус	Коментар
Поява меню завершення гри, коли здоров'я гравця дорівнює 0	Пройшов	Меню завершення гри з'явилося після смерті персонажа гравця. Графічний інтерфейс вимкнувся

Натиснути кнопку Restart	Пройшов	Кнопка натискається. Перезапуск ігрової сцени
Натиснути кнопку Main Menu	Пройшов	Кнопка натискається. Відбувається перехід на сцену Головного меню
Натиснути кнопку Quit Game	Пройшов	Кнопка натискається. Відбувається закриття ігрового додатку

Висновки до третього розділу

У ході виконання даного розділу було розглянуто порядок встановлення та налаштування системи. Було описано представлено елементи інтерфейсу як самої гри, так і ігрових меню

Було здійснено тестування, усунення знайдених проблем і сформовано чек лісти на основі виконаного тестування

ВИСНОВКИ

В процесі розробки кваліфікаційної роботи було розроблено 3D-гра в жанрі "action" із акцентом на складову шутера - «From the Darkness»

У ході виконання роботи було визначено завдання, які мають бути виконані. Було проаналізовано існуючі програмні продукти, визначено їх переваги та недоліки.

Для створення проекту було обрано середовище розробки Unity 3D, разом з додатком для тривимірного моделювання 3Ds Max, де було створено 3D моделі, а також зроблені анімації для них. Написання скриптів, які відповідали за основні ігрові механіки, було реалізовано на мові програмування C#.

Було проведено прототипування ігрового продукту з використанням діаграм активностей, класів, послідовностей.

Було розглянуто порядок встановлення та налаштування системи.

В кінці було проведено ряд тестів, які мали на меті знаходження і усунення проблем.

Кінцевий проект відповідає початковий вимогам і є максимально наближеним до початкової задумки

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

ЛІТЕРАТУРА

1. StarUML документація, вступ [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.staruml.io>.
2. Патерни проектування [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns>.
3. Unity. Documentation Version: 5.3 (switch to 5.4b) [Електронний ресурс] / Unity. – 2015. – Режим доступу до ресурсу: <https://docs.unity3d.com/ru/530/Manual/AnimationGlossary.html>.
4. Приклади та статті на різну айті тематику, включаючи про архітектуру програмних додатків [Електронний ресурс] – Режим доступу до ресурсу: <https://www.wafrat.com/>.
5. Патерни для ігрових додатків [Електронний ресурс] - Режим доступу до ресурсу: <http://gameprogrammingpatterns.com/>.
6. 3ds Max Forums - Autodesk Community [Електронний ресурс] - Режим доступу до ресурсу: <https://forums.autodesk.com/t5/3ds-max/ct-p/area-c1>.
7. Level up your code with game programming patterns / E-book / Unity. 2021
8. Game Programming Gems, Mark DeLoura. 2000.
9. Документація Unity [Електронний ресурс] - Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/index.html>.
10. System Modeling and Analysis: a Practical Approach / Gerrit Muller.
11. YouTube канал Brackeys присвячений створенню ігр на Unity та програмуванню на мові C# та інших – Режим доступу: https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA

ДОДАТКИ

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

Лістинг скрипту EnemyController, який відповідає за поведінку ворогів на ігровій арені:

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

public class EnemyController : MonoBehaviour
{
    Rigidbody m_Rigidbody;

    private float timeOfLastAttack = 0;
    private bool hasStopped = false;

    private NavMeshAgent agent = null;
    private Animator anim = null;
    private EnemyStats stats = null;
    private Transform target;

    private void Start()
    {
        GetReferences();
        m_Rigidbody = GetComponent<Rigidbody>();
    }

    private void Update()
    {
        MoveToTarget();
    }

    private void MoveToTarget()
    {
        agent.SetDestination(target.position);
        anim.SetFloat("Speed", 1f, 0.3f, Time.deltaTime);

        RotateToTarget();

        float distanceToTarget = Vector3.Distance(target.position, transform.position);
        if(distanceToTarget <= agent.stoppingDistance+0.1)
        {
            anim.SetFloat("Speed", 0f);

            if(!hasStopped)
            {
                hasStopped = true;
                timeOfLastAttack = Time.time;
            }

            m_Rigidbody.constraints = RigidbodyConstraints.FreezePosition;

            if(Time.time >= timeOfLastAttack + stats.attackSpeed)
            {
                timeOfLastAttack = Time.time;
                CharacterStats targetStats = target.GetComponent<CharacterStats>();
                AttackTarget(targetStats);
            }
        }
    }
}
```

```

        else
        {
            if(hasStopped)
            {
                hasStopped = false;
            }
        }
    }

    private void RotateToTarget()
    {
        transform.LookAt(target);
    }

    private void AttackTarget(CharacterStats statsToDamage)
    {
        anim.SetTrigger("Attack");
        stats.DealDamage(statsToDamage);
    }

    private void GetReferences()
    {
        agent = GetComponent<NavMeshAgent>();
        anim = GetComponentInChildren<Animator>();
        stats = GetComponent<EnemyStats>();
        target = PlayerController.instance;
    }
}

```

Додаток Б

Лістинг скрипту UIManager, який відповідає за відображення меню гри, а також за роботу кнопок в меню:

```

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

public class UIManager : MonoBehaviour
{
    private bool isPaused = false;

    [SerializeField] private GameObject hudCanvas = null;
    [SerializeField] private GameObject pauseCanvas = null;
    [SerializeField] private GameObject endCanvas = null;

    private CameraController camController = null;
    private PlayerStats stats = null;

    private void Start()
    {
        GetReferences();
        SetActiveHud(true);
    }

    private void Update()
    {
        if(!stats.IsDead())
        {
            if (Input.GetKeyDown(KeyCode.Escape) && !isPaused)

```

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

```

        SetActivePause(true);
    else if (Input.GetKeyDown(KeyCode.Escape) && isPaused)
        SetActivePause(false);
    }
}

public void SetActiveHud(bool state)
{
    hudCanvas.SetActive(state);
    endCanvas.SetActive(!state);

    if(!stats.IsDead())
        pauseCanvas.SetActive(!state);
}

public void SetActivePause(bool state)
{
    pauseCanvas.SetActive(state);
    hudCanvas.SetActive(!state);

    Time.timeScale = state ? 0 : 1 ;
    if(state)
        camController.UnlockCursor();
    else
        camController.LockCursor();
    isPaused = state;
}

public void Restart()
{
    Time.timeScale = 1f;
    SceneManager.LoadScene(1);
}

public void MainMenu()
{
    SceneManager.LoadScene(0);
}

public void Quit()
{
    Application.Quit();
}

private void GetReferences()
{
    camController = GetComponentInChildren<CameraController>();
    stats = GetComponent<PlayerStats>();
}
}

```