

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:

**«Комп'ютерна гра у жанрі метроїдванія “Dark
Watcher” – система битв, ігровий інтелект супротив-
ників»**

Виконав студент групи ПЗ-19-1
ЛЕУТ Назарій Віталійович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2023

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: **ЛЕУТ Назарій Віталійович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Комп'ютерна гра у жанрі метроїдванія “Dark Watcher” – система битв, ігровий інтелект супротивників»**,

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

Вихідні дані для роботи: **Об'єктом дослідження є комп'ютерна гра "Dark Watcher" у жанрі метроїдванія. Предметом дослідження є система битв та ігровий інтелект супротивників, які використовуються у грі.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	14.02.2023	14.02.2023
2	Морозов А.В.	14.02.2023	14.02.2023
3	Морозов А.В.	14.02.2023	14.02.2023

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.	02 березня 2023 – 05 квітня 2023	
2	Проектування структури системи	6 квітня 2023 – 12 квітня 2023	
3	Написання програмного коду	13 квітня 2023 – 26 квітня 2023	
4	Тестування	27 квітня 2023 – 04 травня 2023	
5	Підготовка пояснювальної записки	5 травня 2023 – 11 червня 2023	

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу гри в жанрі метроїдванія “Dark Watcher” та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 71 сторінці. В роботі наведено 20 ілюстрацій та 28 таблиць, один додаток. Список використаних джерел містить 19 найменувань і займає 2 сторінки. Загальний обсяг роботи – 74 сторінки.

Метою роботи є розробка гри у жанрі метроїдванія, за використання засобів рушія Unity та мови програмування C#.

В роботі встановлено ключові завдання для розробки системи, проведено аналіз аналогічних проєктів. Наведено сценарії роботи програмного комплексу, його загальна структура, опис алгоритмів взаємодії окремих компонентів системи, об’єктної структури системи. Крім цього наведено результати тестування готового продукту.

Ключові слова: Dark Watcher, метроїдванія, 2D, платформер, бойова система, ігровий інтелект, боси.

					ІПЗ.КР.Б – 121 – 23 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.	Н.В. Леут				Комп’ютерна гра у жанрі мет- роїдвання “Dark Watcher” – сис- тема битв, ігровий інтелект су- противників. Пояснювальна записка			Лім.	Арк.	Аркушів	
Керівник	А.В. Морозов									4	75
								Житомирська політехніка, група ІПЗ-19-1			
Н. контр.	О.П. Тичина										
Зав. каф.	А.В.Морозов										

ABSTRACT

The bachelor's thesis consists of the software complex of the game in the metroidvania genre "Dark Watcher" and an explanatory note. The explanatory note to the final thesis contains a text part laid out on 72 pages. The work contains 20 illustrations and 28 tables, one appendix. The list of used sources contains 19 names and occupies 2 pages. The total volume of work is 75 pages.

The purpose of the work is to develop a game in the metroidvania genre using the Unity engine and the C# programming language.

In the work, the key tasks for the development of the system were established, and the analysis of similar projects was carried out. Scenarios of the software complex, its general structure, a description of the interaction algorithms of individual system components, and the object structure of the system are presented. In addition, the results of testing the finished product are given.

Keywords: Dark Watcher, metroidvania, 2D, platformer, combat system, game intelligence, bosses.

		<i>Н.В. Леут</i>			ІПЗ.КР.Б – 121 – 23 – ПЗ	<i>Арк.</i>
		<i>А.В. Морозов</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		5

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР	11
1.1. Постановка задачі.....	11
1.2. Аналіз аналогів програмного продукту	12
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	30
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	30
2.2 Розробка супротивників	36
2.3. Проектування та реалізація алгоритмів роботи системи.....	38
2.4. Реалізація ігрового додатку	45
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ В ЖАНРІ МЕТРОЇДВАНІЯ “DARK WATCHER”	58
3.1. Порядок встановлення та налаштування системи.....	58
3.2. Структура інтерфейсу та порядок роботи з системою битв.....	59
3.3. Тестування роботи системи	62
ВИСНОВКИ	69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	70
ДОДАТКИ	72
Додаток А. Відгуки на кваліфікаційну роботу.....	73

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Той, хто спостерігає уважно, не міг пропустити очевидне: сучасний світ все більше і більше захоплюють комп'ютерні ігри, які вже зайняли важливе місце у повсякденному житті людей. За останні роки, ігровиробнича індустрія неймовірним чином зростає і розвивається, завойовуючи серця мільйонів гравців по всьому світу. Сьогодні багато людей регулярно проводять свій вільний час за ігровим процесом, щоб отримати задоволення, відпочити чи пережити емоції, недоступні для реального світу.

Серед розмаїття жанрів відеоігор особливе місце займає "метроїдванія". Назва жанру походить від найменування гри "Metroid", яка була випущена в 1986 році. Цей жанр характеризується відкритим світом, безліччю секретів і прихованих проходів, а також необхідністю повертатися назад у попередні локації після отримання нових здібностей, щоб розблокувати раніше недоступні області. Гравцеві дається велика свобода дослідження гри. "Метроїдванія" містить в собі також елементи гри "Castlevania: Symphony of the Night" (1997) такі як темний фентезійний світ з монстрами, та здатністю персонажа до покращення своїх здібностей та екіпірування.

Зважаючи на це, представимо комп'ютерну гру у жанрі "метроїдванія" – "Dark Watcher". Вже з перших моментів ігрового процесу гравець занурюється в її унікальний та неповторний світ, насичений неординарним сюжетом, приємною стилізованою графікою та відточеними ігровими механіками, притаманним жанру.

Практична цінність "Dark Watcher" проявляється у наданні гравцям можливості розвинути свою фантазію, увагу, моторику та критичне мислення, навчить швидкому прийняттю рішень навіть у непередбачуваних ситуаціях під час ігрового процесу, ставлячи їх перед різноманітними труднощами та викликами зокрема у сутичках з противниками та під час виборів.

Спираючись на невпинний процес популяризації та збільшення впливу комп'ютерних ігор на суспільство, орієнтування в особливостях та сильних

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		7

сторонах жанру "метроїдванія", дослідження ігроладу "Dark Watcher" необхідне усім, хто зацікавлений у геймінгу і бажає глибше вивчити цю захоплюючу сферу. Саме тому, у рамках даного проєкту, ми присвятимо особливу увагу одній з основних та невід'ємних частин ігор даного жанру, а саме бойовій системі та ігровому інтелекту супротивників.

Об'єкт та предмет дослідження:

Об'єктом дослідження є комп'ютерна гра "Dark Watcher" у жанрі метроїдванія. Предметом дослідження є система битв та ігровий інтелект супротивників.

В процесі дослідження об'єкту та предмету, проаналізуємо особливості системи битв, які є невідкладною складовою ігрового процесу та реалізує взаємодію гравця з противниками, що стоять на шляху дослідження ігрового світу та подальшого просування. Крім цього розглянемо механіки сутичок, різноманітні види й типи атак доступні гравцю й противникам та їхній ігровий інтелект під час ведення бою.

Метою дослідження цих сторін є розкриття значимості та ефекту на ігровий процес гри "Dark Watcher". Детальний аналіз результатів, отриманих безпосередніми учасниками розробки гри та наявних даних щодо подібних проєктів даної області, дозволить досягнути кращої обізнаності в сучасних тенденціях розробки відеоігор та поліпшенню отриманого безпосередньо гравцями досвіду.

Застосування методів дослідження, таких як аналіз, порівняння та моделювання, дозволить висвітлити ключові аспекти системи битв та ігрового інтелекту супротивників у грі "Dark Watcher" та з'ясувати їхню функціональність та взаємозв'язок з ігроладом.

Таким чином, дослідження гри "Dark Watcher" визначить особливості системи битв та ігрового інтелекту супротивників, їх вплив на геймплей та сприятиме поліпшенню досвіду, отриманого в процесі гри.

Методи аналізу та інформаційна база дослідження:

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		8

В процесі розробки дипломного проєкту було застосовано такі методи: огляд літературних джерел та проведення досліджень, аналіз ігрового досвіду та експериментальне моделювання.

Дослідницький матеріал включав в себе наукові статті, публікації, документацію та коментарі розробників комп'ютерних ігор. Крім того, було взято до уваги інформацію з офіційних сторінок в мережі та форумів з обговоренням відеоігор, подібних до "Dark Watcher".

В процесі розробки продукту джерелами натхнення стали наступні ігри: "Hollow Knight", "Ori and the Will of Wisps", "Blasphemous" та "Dark Souls". Вони стали основою ідей під час аналізу системи битв та ігрового інтелекту супротивників, ігроладу "метроїдваній".

Гра "Dark Watcher" була розроблена з використанням Unity - платформи, що надає широкий інструментарій, зручний та зрозумілий інтерфейс для створення відеоігор за високими стандартами якості. Завдяки Unity були успішно втілені продумані системи битв та ігровий інтелект супротивників у грі "Dark Watcher".

В результаті, гра "Dark Watcher" була розроблена шляхом використання аналітичних методів, що ґрунтуються на літературних джерелах та ігровому досвіді, разом із використанням дослідницької інформаційної бази, яка включала наукові ресурси та думки розробників. Це посприяло створенню гри "Dark Watcher" на платформі Unity, почерпавши натхнення у вдалих прикладах ігор жанру.

Практичне значення отриманих результатів:

Одержані висновки мають значимість для розробників відеоігор і геймерської спільноти вцілому.

По-перше, дослідження системи битв та ігрового інтелекту супротивників в грі "Dark Watcher" надає розробникам цінні уявлення про ефективні підходи та принципи їх втілення в "метроїдваніях". Це сприяє поліпшенню якості геймплею та набутого користувацького досвіду під час гри.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Крім того, в отриманих результатах досліджень є потенціал бути використаними з метою збільшення естетичної привабливості та створення атмосфери гри "Dark Watcher". Розуміння ігрового процесу, який успішно використовується в популярних іграх, дозволить використовувати найкращі підходи та створювати унікальну взаємодію.

Також, отримані результати можуть послужити основою для подальших досліджень і розробок в сфері відеоігор. Вони сприяють розвитку новаторських підходів до системи битв та ігрового інтелекту супротивників, що збагачує спільноту з досліджень у цій області, відкриваючи нові можливості для творчості та нових ідей.

Отже, практичне значення отриманих результатів полягає в покращенні процесу розробки відеоігор, поліпшенні та вдосконаленні ігрового процесу, появи приємних та ефектних ігрових практик гравців.

Структура та обсяг проекту (роботи). Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 19 найменувань (2 сторінки), 1 додатку (2 сторінки), 28 таблиць та 20 ілюстрацій по тексту. Повний обсяг роботи становить 74 сторінки, з них 71 сторінка основного тексту

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		10

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР

1.1. Постановка задачі

Основна задача дослідження полягає у розробці та реалізації комп'ютерної гри у жанрі метроїдванія під назвою "Dark Watcher". Для досягнення цієї мети потрібно вирішити наступні підзадачі:

1. Розробка системи битв

1.1.Визначення правил та механік, що будуть використовуватись під час битв

1.2.Визначення видів атак, заклять та інших дій, які гравець може виконати під час битви

1.3.Створення графічного інтерфейсу користувача для відображення важливої інформації під час битви – очок мари, здоров'я гравця та ворогів

1.4.Розробка системи анімацій та візуальних ефектів для зрозумілого відображення дій гравця та ворогів під час битви

1.5.Здійснення корекцій у механіках та параметрах для досягнення балансу між задоволенням та викликом

1.6.Розробка алгоритму визначення умов перемоги та поразки у битвах

1.7.Створення системи нагород в разі успіху

2. Розробка ігрового інтелекту супротивників

2.1.Реалізація анімацій для різних дій противників

2.2.Впровадження механік атак та взаємодії супротивників з гравцем

2.3.Розробка системи поведінки супротивників в залежності від дій гравця та з елементом випадковості

Після успішної реалізації цих проміжних завдань, комп'ютерна гра "Dark Watcher" буде готова для переходу до етапу тестування та кінцевого

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		11

полірування. Основним завданням є створення захоплюючої гри у жанрі метроїдванія, яка забезпечить гравцям неймовірну графіку, різноманітність бойової системи та супротивників, щоб забезпечити їм цікавий ігровий процес.

1.2. Аналіз аналогів програмного продукту

Опираючися на вимоги, висунуті до гри, та врахування специфіки предметної галузі, відбувається дослідження. Отримані результати будуть підґрунтям для прийняття рішення щодо вибору платформи проектування, технологій та інтегрованим середовища розробки.

Hollow Knight – це захоплююча гра в жанрі метроїдванія, розроблена і видана незалежною австралійською студією Team Cherry [11]. Гра вийшла в 2017 році і швидко здобула популярність завдяки своїй атмосфері темного фентезі, глибокому геймплею та неймовірному художньому стилю. У Hollow Knight гравець бере на себе роль безіменного подорожнього, який опиняється в підземному світі королівства, що давно занепало. Головна мета гравця – розгадати та дослідити таємниці цього світу, зіштовхуючися з небезпеками, зустрічаючи різноманітних істот та прокладаючи шлях через складні пастки та босів. Гра пропонує великий відкритий світ з численними гілками розвитку, секретами і викликами, що чекають на дослідження.



Рисунок 1.1 – Hollow Knight

Переваги:

– вишукана атмосфера: гра окутує містичною та мрійною атмосферою. Чудово промальовані пейзажі, меланхолійна музика і затишні, та одночасно небезпечні локації створюють неперевершену ігрову обстановку

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		12

– глибокий геймплей: Hollow Knight пропонує надзвичайно розгалужену систему бою та здібностей, що дає гравцеві безліч варіантів для підлаштування під свій стиль гри. Ви зможете отримувати нове спорядження, навички та властивості для боротьби зі зростаючими викликами гри;

– розмаїтість монстрів і босів: ви зустрінете багато унікальних монстрів, кожен з яких має свої унікальні особливості і стиль бою. Боси гри особливо вражають своєю силою і дизайном та потребують унікального підходу для здобуття перемоги над ними;

– великий світ для дослідження: світ гри є великим і розширеним, з численними секретами і гілками, які можна досліджувати. Гра надає вам свободу вибору маршруту, і завдяки цьому кожен гравець може знайти свій унікальний шлях через гру;

– присутні безкоштовні доповнення, що доповнюють основну гру та додають новий ігровий вміст;

– йде робота над грою-продовженням.

Недоліки:

– моменти надмірної складності: гра вимагає від гравця великої витривалості і вміння подолати певні складні випробування, проте ці виклики не є основною частиною гри і не обов'язкові до проходження;

– високий поріг входу: механіки гри прості в розумінні, однак для майстерного володіння ними необхідний певний час або попередній досвід з подібними іграми;

– обмежені вказівки: Hollow Knight не надає багато вказівок або пояснень, іноді залишаючи гравця в розгубленості. Це може бути проблемою для тих, хто шукає більше направленості та настанови;

– нестандартна подача сюжету: сюжет у грі поданий через діалоги з неігровими персонажами та написи, розміщені по світу, що може бути незвичним для людей, які звикли до стандартної прямолінійної сюжетної історії;

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		13

- обмежені налаштування графіки та інтерфейсу;
- відсутність української локалізації.

Ori and the Will of the Wisps – це захоплююча метроїдванія, розроблена студією Moon Studios і видана Xbox Game Studios [14]. Гра є продовженням популярної "Ori and the Blind Forest" і знову веде гравця в неймовірну казкову пригоду у ролі маленького духа світла на ім'я Орі. "Ori and the Will of the Wisps" пропонує неперевершену графіку, мальовничий арт-дизайн та вражаючу атмосферу. Гра розташована у магічному світі, де гравець зустрічається з різноманітними істотами, розв'язує загадки та бореться з ворожими силами. Геймплей "Ori and the Will of the Wisps" є динамічним та різноманітним. Гравець має рухатися по грі, виконувати складні стрибки, збирати предмети та вдосконалювати навички свого героя. Гра пропонує багато різних ворогів, босів та головоломок, що додають грі викликів та розваг. Головна мета полягає у порятунку світу, дослідженні глибокої емоційної історії про любов, жертвність і надію.

Переваги:

- вражаюча графіка та дизайн: гра вражає мальовничими краєвидами, деталізацією персонажів та кольоровими схемами, що створюють неперевершену атмосферу;
- глибока історія: гра пропонує захоплюючий наратив, який спонукає гравця відчувати емоції та зануритися у світ Орі;
- різноманітний ігровий процес: гра пропонує різноманітність викликів та завдань, від виконання стрибків та акробатичних трюків до розв'язування головоломок та боротьби з ворогами;
- великий світ: "Ori and the Will of the Wisps" пропонує великий та розгалужений світ для дослідження, з численними секретами, прихованими областями та побічними завданнями;
- музика та звуковий дизайн: гра має чудовий саундтрек та вражаючий звуковий дизайн, які додають глибини та емоцій до геймплею.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Недоліки:

- можливі проблеми з просуванням по грі: проблеми з прохідністю деяких рівнів або складних стрибків можуть призвести до перебування в геймплейному тупику;
- незначна нестабільність частоти кадрів: у деяких випадках можуть виникати дрібні зниження продуктивності або дрібні затримки, особливо на менш потужних системах;
- присутність не схожих одна на одну механік: гра пропонує різноманітні механіки, що може відштовхнути гравців, які орієнтуються лише на один жанр гри;
- відсутня українська локалізація.



Рисунок 1.2 – Ori and the Blind Forest

Blasphemous – це екшн-пригодницька метроїдванія, розроблена компанією The Game Kitchen [12] і видана студією Team17 [13]. Гра випущена у вересні 2019 року і швидко заробила визнання завдяки своїй готичній атмосфері, художньому стилю і високоякісному геймплею. Blasphemous заснована на досить похмурому та дорослому фентезійному світі. Гравець бере на себе роль миротворця, який прокидається від прокляття, щоб пробитися через ворожі землі, зустрічаючи різноманітних ворогів і розв'язуючи загадки, щоб очистити світ від гріховної зневаги.

Переваги:

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		15

- вражаючий готичний арт-стиль і дизайн: гра відрізняється ретельно промальованою піксельною графікою, які створюють атмосферу похмурості і відчуття загрози;
- кожен персонаж, локація та ворог пропрацьовані з деталізацією і виявляються неймовірно жахливими і красивими одночасно;
- глибока тематика: гра пропонує глибоку та змістовну сюжетну лінію, яка досліджує релігійні та філософські теми, примушуючи гравців роздумувати над ними;
- розширене дерево навичок, яке дозволяє гравцеві налаштувати свій стиль гри;
- висока реграбельність;
- ведеться робота над сиквелом.

Недоліки:

- можлива складність для новачків: висока складність гри може відлякати новачків, які не звикли до такого рівня виклику;
- відсутність української локалізації.



Рисунок 1.3 – Blasphemous

"DARK SOULS™: Prepare To Die Edition" – це екшн-рольова гра, розроблена студією FromSoftware та видана компанією Bandai Namco Entertainment [15]. Гра вперше вийшла у 2012 році та стала першою частиною культової серії Dark Souls. У грі "DARK SOULS™: Prepare To Die

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Edition" гравці відправляються у суворий світ, наповнений монстрами, загадками та небезпеками. Головний акцент робиться на високому рівні виклику та безпощадності. Гра відрізняється від багатьох інших ігор свого жанру глибиною сюжету та відмінною системою бою.

Переваги:

- висока складність: граючи у DARK SOULS, гравець стикається з безкомпромісними випробуваннями, що вимагають великої уваги, майстерності та терпіння. Це стимулює розвиток навичок і стратегічне мислення;
- вага рішень: кожне рішення гравця в битвах з монстрами та босами має велике значення і може вплинути на подальший розвиток історії гри. Це надає грі велику глибину та різноманітність варіантів проходження;
- атмосферний світ: гра створює постапокаліптичну ауру, наповнену темрявою, готичною архітектурою та містичними створіннями. Кожна локація виглядає унікально та деталізовано, створюючи захоплюючу атмосферу;
- музика та звуковий дизайн: надзвичайна музика та звуковий дизайн гри підсилюють почуття небезпеки та загадковості. Вони допомагають зануритися у гру і створюють незабутні враження.

Недоліки:

- високий поріг входження: гра може бути дуже важкою для новачків, особливо для тих, хто не знайомий зі специфікою серії Dark Souls або подібних ігор. Це може відлякати деяких гравців і створювати перешкоди для їх насолоди геймплеем;
- проблеми з оптимізацією на ПК: гра може мати деякі проблеми з оптимізацією на ПК, що може вплинути на плавність геймплею та загальний досвід. Це може бути дратуючим для гравців, які очікують безперебійного та комфортного геймплею;
- припинення підтримки онлайн-складової гри, через вихід оновленої версії;

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		17

- технічні проблеми на старших платформах: гра може мати деякі технічні проблеми на старших платформах, які можуть вплинути на задоволення гравців від гри. Це можуть бути помітні збої, падіння кадрів або інші проблеми, які впливають на геймплей та загальний досвід;
- відсутність української локалізації.



Рисунок 1.4 – Dark Souls

Для глибокого розуміння процесу розробки схожих додатків та відповідального підходу до цієї роботи необхідно сформувати таблицю, яка включатиме інформацію про використані технології та кількість розробників, задіяних при розробці ігор з аналогічного жанру та рівня складності [16]. Ця порівняльна таблиця, що включає дані про використані рушії та мови програмування, має значення з наступних причин:

- технічний аспект: знання про використані рушії та мови програмування, які використовувалися для створення гри, може бути корисним для розробників та тих, хто цікавиться технічними аспектами розробки. Це дає уявлення про інструменти та технології, використовувані під час розробки різних ігор;
- вивчення підходів: знання про використані рушії та мови програмування може допомогти вивчити підходи розробників до створення ігор. Це особливо корисно для тих, хто цікавиться незалежною розробкою ігор або бажає зрозуміти, які технології використовуються в конкретних проєктах;

– вибір технологій: при плануванні власної гри, знання про використані рушії та мови програмування може вплинути на вибір технології. Порівняння рушіїв та мов допомагає зрозуміти переваги та недоліки, а також знайти найбільш підходящу технологію для конкретного проєкту;

– слідування за трендами: порівняльна інформація про рушії та мови програмування, використані в популярних іграх, допомагає розуміти тренди та напрямки розвитку галузі розробки ігор.

Це дозволяє бути в курсі нових технологій та приймати обґрунтовані рішення.

Таблиця 1.1. Інструментарій розробки додатків-аналогів

Гра	Рушій	Мова програмування	Кількість розробників
Hollow Knight	Unity	C#	15-20
Ori and the Will of the Wisps	Unity	C#	8-12
Blasphemous	Game Maker Studio 2	GML	10-15
"DARK SOULS™: Prepare To Die Edition"	Katana (Dantelion)	C++	230

Варто зазначити, що в числі розробників здійснювався підрахунок лише тих, хто брав безпосередню участь у активній розробці кожного з проєктів.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

При виборі стеку програмування потрібно врахувати наявну кількість тематичної літератури, документованих матеріалів та розмір спільноти, що зможе посприяти уникненню та пришвидшенню вирішення потенційних проблем.

Під час вибору інструментів, було обрано мову програмування C#.

1.3.1. C#:

C# (C-Sharp) є однією з найпопулярніших мов програмування, використовуваних в сфері розробки ігор. Вона є об'єктно-орієнтованою мовою, розробленою компанією Microsoft і базується на платформі .NET. C# здатна працювати на різних платформах, включаючи Windows, Xbox, PlayStation та інші.

Однією з найбільших переваг C# для розробки ігор є його зручне інтегроване середовище розробки (IDE) – Visual Studio. Visual Studio надає розробникам широкі можливості для створення ігрових проєктів, включаючи підтримку візуального проєктування, інтеграцію зі спеціалізованими інструментами, такими як Unity. Це робить процес розробки більш ефективним і швидким.

C# пропонує широкий спектр функцій і можливостей, які сприяють розробці як 2D, так і 3D ігор [17]. Вона має потужну систему керування пам'яттю, що дозволяє уникнути багатьох проблем, пов'язаних зі сміттєзбіркою. Бібліотеки і фреймворки, такі як Unity і MonoGame, надають розширені можливості для створення ігор з використанням C#.

C# також має можливості управління графікою, звуком та фізикою за допомогою вбудованих бібліотек, що дозволяє розробникам створювати реалістичні та захопливі ігри. Мова має вбудовану підтримку мультимедіа, а також доступ до графічних бібліотек і API, що спрощують розробку графіки в іграх.

Ще однією перевагою C# для розробки ігор є велика спільнота розробників, яка активно підтримується Microsoft і іншими компаніями. Розробники можуть знайти безліч документації, підручників, форумів і ресурсів, які допоможуть їм вирішити будь-які проблеми, з якими вони можуть зіткнутися під час розробки своїх ігор.

1.3.2. C# Mono

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		20

C# Mono є однією з важливих технологій для розробки програмного забезпечення, зокрема в галузі геймдеву [1]. Mono є реалізацією відкритого стандарту ECMA-334 для мови програмування C#. Вона надає розширені можливості для розробки мультиплатформного програмного забезпечення, що означає, що програми, написані на C# Mono, можуть працювати на різних операційних системах, таких як Windows, macOS та Linux, забезпечуючи широку аудиторію користувачів.

C# Mono використовується в багатьох сферах розробки програмного забезпечення, включаючи розробку ігор. Вона надає розширений набір бібліотек та інструментів, необхідних для створення графічних інтерфейсів, обробки даних, мережевого програмування, управління пам'яттю та багатьох інших функцій.

C# Mono також підтримує розвиток розширень та модульну архітектуру, завдяки чому розробники можуть створювати додаткові компоненти, бібліотеки та плагіни, які розширюють функціональність основної програми. Це дозволяє створювати розширення, які додають нові функції та можливості до вже існуючих ігор.

Крім того, C# Mono має розширену підтримку інтегрованих середовищ розробки, таких як Visual Studio та Unity. Це забезпечує зручне середовище для програмістів, яке допомагає в розробці, налагодженні та тестуванні ігор.

Узагальнюючи, технологія C# Mono є потужним інструментом для розробки програмного забезпечення, зокрема відеоігор. Вона надає переносимість, розширюваність, багатофункціональність та зручність розробки, що робить її популярним вибором для створення ігор на різних платформах.

Зваживши це, зупинимось на цій мові програмування.

Спираючись на обрану мову програмування, розглянемо доступні ігрові рушії, для мови програмування C#. Серед найпопулярніших:

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		21

– Unity – це кросплатформний рушій розробки програмного забезпечення, який широко використовується для створення відеоігор, віртуальної реальності, доповненої реальності та інших інтерактивних додатків. Він надає зручні інструменти для моделювання, анімації, фізики, штучного інтелекту та розробки ігрових сценаріїв, що дозволяє розробникам створювати додатки для різних платформ, включаючи комп'ютери, консолі, мобільні пристрої та веб [2];

– Godot Engine - це безкоштовний, відкритий та кросплатформний рушій розробки програмного забезпечення, призначений для створення відеоігор та інтерактивних додатків. Заснований на мові програмування Python і C++, Godot Engine надає розробникам потужні інструменти для створення графічних об'єктів, анімації, фізики, штучного інтелекту та розробки ігрових сценаріїв. Він підтримує різні платформи, включаючи комп'ютери, консолі, мобільні пристрої та веб, і має активну спільноту розробників, яка надає додаткові ресурси та підтримку [9];

– Xenko - це кросплатформний рушій розробки програмного забезпечення, спеціально призначений для створення відеоігор. Він пропонує розробникам широкий інструментарій для моделювання, анімації, фізики, штучного інтелекту та розробки ігрових сценаріїв. Xenko має велику базу платформ. Рушій має зручний інтерфейс користувача і надає можливості для розробки як 2D, так і 3D ігор з високою якістю графіки та звуку. Завдяки своїй відкритій архітектурі, Xenko також надає розробникам можливість налаштовувати рушій під їхні потреби та використовувати розширення, що підтримуються спільнотою [9];

– Wave Engine - це кросплатформний рушій розробки програмного забезпечення, який дозволяє створювати відеоігри та інтерактивні додатки. Він надає розробникам зручні інструменти для моделювання, анімації, фізики, звуку та розробки відеоігор. Рушій також надає можливість розробникам працювати з графічними ефектами, освітленням, штучним інтелектом

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		22

та іншими аспектами гри, що дозволяє створювати вражаючі ігрові візуальні ефекти [8].

Зважаючи всі плюси та мінуси було обрано Unity в ролі ігрового рушія для розробки проєкту. Ключовими причинами для рішення стали:

- популярність: Unity є одним з найбільш поширених ігрових рушіїв у сфері розробки відеоігор. Велика спільнота розробників Unity є однією з найбільших спільнот розробників галузі. Існує безліч форумів, блогів, посібників і ресурсів, де можна знайти допомогу та поради від інших розробників;
- програмування на C#: C# – сучасна мова, яка має широкую екосистему з великою кількістю ресурсів, бібліотек та інструментів, які полегшують розробку ігор. Будучи основною мовою програмування в Unity, робить процес розробки комфортнішим та продуктивнішим для спеціалістів, які стикалися з роботою з використанням C# або .NET-платформи;
- кросплатформність: Unity підтримує розробку ігор для різних платформ, включаючи Windows, macOS, Linux, iOS, Android, Xbox, PlayStation. Це дозволяє легко переносити гру на різні пристрої без значного зусилля та необхідності переписування коду;
- спеціальні інтеграції: в Unity присутні потужні інтегровані інструменти, для створення, налаштування та управління ігровими об'єктами, сценами та ресурсами. Розробники можуть взаємодіяти з графічним інтерфейсом, щоб швидко створювати та редагувати елементи гри, швидко прототипувати та відлагоджувати проєкт, що пришвидшує процес розробки та робить його ефективнішим.

Вцілому, обрання Unity та C# для розробки гри – це вигідний вибір, оскільки Unity надає потужні інструменти для створення ігрових середовищ, а C# забезпечує легку вивченість та високу продуктивність, що дозволяє розробникам швидко та ефективно створювати вражаючі ігрові проєкти.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Скриптовані об'єкти – це тип об'єктів що дозволяє зберігати, організовувати та маніпулювати даними ігрових станів всередині гри. Вони є контейнерами що містять в собі поля, властивості та методи схожі на класи, але їх можна створювати, налаштовувати безпосередньо в редакторі Unity без повторного написання коду [3].

Переваги використання Scriptable Objects:

- параметризація даних: Scriptable Objects дозволяють розробникам створювати параметризовані об'єкти, що надає можливість легко змінювати властивості об'єктів і налаштовувати їхню поведінку, не змінюючи вихідний код;
- відокремлені дані: Scriptable Objects дозволяють відокремити дані від об'єктів, що робить їх легкими для управління та обміну між різними об'єктами;
- використання в редакторі Unity: Scriptable Objects можна змінювати та налаштовувати безпосередньо в редакторі Unity, що спрощує розробку та тестування гри;
- підтримка серіалізації: Scriptable Objects підтримують серіалізацію, що дозволяє зберігати їх стан та передавати через різні сцени або між сеансами гри.

Недоліки використання Scriptable Objects:

- велика кількість файлів: використання багатьох Scriptable Objects може призвести до великої кількості файлів, що може бути складним для управління та підтримки;
- обмеження на редагування: Scriptable Objects можуть бути важкими для редагування, особливо якщо вони містять складну логіку або велику кількість даних. Редагування таких об'єктів може вимагати додаткового зусилля і часу;

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		24

– вимоги до пам'яті: якщо використовуються великі об'єкти Scriptable Objects або їх багато, це може призвести до збільшення вимог до пам'яті, що може вплинути на продуктивність гри;

– висока складність: використання Scriptable Objects може бути викликом через їх високу складність, що передбачає розуміння концепцій та особливостей роботи з ними. Це може ускладнити проєкт і вимагати від розробників додаткового зусилля.

Незважаючи на недоліки, використання Unity Scriptable Objects може бути ефективним інструментом для розробки гри, забезпечуючи гнучкість та легкість управління даними.

C# Action – є одним із вбудованих делегатів в мові програмування C# [19]. Він може визначатись з різною кількістю параметрів, але не повертає значення. Action може бути використаний для передачі методу як параметра в інші методи, для зберігання методу в колекціях або для створення лямбда-виразів. Він дозволяє зручно передавати функціональність без необхідності визначати новий інтерфейс або делегат. Після визначення типу Action можна створити його екземпляр та передати в нього методи або лямбда-вирази, які викликатимуться при використанні цього екземпляра. Action є вигідним інструментом для функціонального програмування в C#, оскільки він дозволяє передавати функції як аргументи, і в результаті створювати більш гнучкі та повторно використовувані фрагменти коду [18].

Переваги використання C# Action:

– гнучкість: C# Action дозволяє передавати функціональність в якості параметрів, що надає гнучкість в проєктуванні програми. Можна передавати різні методи або лямбда-вирази, що дозволяє використовувати Action для виконання різноманітних функцій без необхідності створювати новий інтерфейс або делегат;

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		25

- простота використання: використання C# Action досить просте. Не потрібно визначати новий інтерфейс або делегат для передачі методу як параметра. Достатньо просто використати вбудований делегат Action, визначивши його тип і передавши в нього потрібний метод або лямбда-вираз;
- підтримка анонімних методів: C# Action дозволяє використовувати анонімні методи або лямбда-вирази, що дозволяє створювати і використовувати функції без необхідності виносити їх в окремі методи;
- збереження коду: використання C# Action дозволяє зберігати функції як об'єкт і використовувати її повторно, що дозволяє знизити дублювання коду і спростити його обслуговування;
- розширюваність: завдяки використанню C# Action, код стає більш розширюваним і модульним. Ви можете легко додавати нові функціональності, передаючи додаткові методи або лямбда-вирази через Action без необхідності змінювати основну логіку програми.
- Недоліки використання C# Action:
 - обмеження типів даних: C# Action приймає функції без поверненого значення (void), що унеможливорює використання Action для передачі функцій, які повертають значення;
 - відсутність перевантаження: Action не підтримує перевантаження методів з однаковою кількістю параметрів, що може створити обмеження при проектуванні складних систем з необхідністю передачі методів з різними параметрами;
 - залежність від лямбда-виразів: незважаючи на те, що лямбда-вирази є потужним інструментом, вони можуть стати складними для розуміння і підтримки, що може збільшити складність коду і зробити його менш прозорим;
 - передача стану: оскільки Action передається як параметр, він може бути обмежений у передачі стану або контексту між функціями і може

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		26

ускладнити спільне використання даних або змінних між різними методами, переданими через Action;

- відсутність повного контролю: використання Action може обмежити ваш контроль над послідовністю виконання функцій. Передаючи функції в Action, відсутній повний контроль над тим, яка функція буде виконана першою або як будуть оброблятися помилки.

Загалом, використання C# Action дозволяє покращити чистоту, гнучкість та розширюваність коду, спрощує його обслуговування та дозволяє створювати більш повторно використовуваний код. Враховуючи ці недоліки, важливо уважно розглянути ситуацію і вибрати відповідний інструмент для своїх потреб, наприклад інші типи делегатів або підходи.

Щодо мінімальних системних вимог для 2D гри на Unity, було визначено наступні значення: ОС Windows 7, процесор Intel Core 2 Duo E5200 чи аналогічний, 4 ГБ оперативної пам'яті, відеокарта GeForce 9800GTX+ з 1 ГБ відеопам'яті, DirectX 10 версії та 2 ГБ доступного місця на диску, що зумовлено такими чинниками:

- сумісність зі старішими системами: обрання ОС Windows 7 як мінімальної вимоги може бути обґрунтовано тим, що Windows 7 була популярною операційною системою, особливо серед геймерів, і багато користувачів досі використовують її на своїх комп'ютерах. Враховуючи розповсюдженість Windows 7 гра, що підтримує цю ОС забезпечить доступність більш широкої аудиторії гравців;

- достатній рівень обчислювальної потужності: процесор Intel Core 2 Duo E5200, хоча і вважається старим, все ще може забезпечити достатній рівень обчислювальної потужності для відеогри такого типу. Обрання цього процесора може дозволити гравцям з менш потужними комп'ютерами відтворювати гру без помітних проблем з продуктивністю;

- мінімальні вимоги щодо оперативної пам'яті: вказані 4 ГБ оперативної пам'яті є мінімальним рівнем, який може забезпечити достатню кількість пам'яті для плавної роботи гри без помітних затримок або зависань;

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		27

– старша, але все ще прийнятна відеокарта: GeForce 9800GTX+ з 1 ГБ відеопам'яті є відносно старою відеокартою, проте вона може задовольнити мінімальні вимоги для відтворення 2D-графіки, що означає, що гравці з такою відеокартою зможуть відтворювати гру з прийнятною якістю графіки і швидкістю кадрів;

– підтримка DirectX 10: DirectX 10 є розповсюдженою версією графічної бібліотеки, яка використовується для розробки ігор. Обрання DirectX 10 як мінімальної вимоги дозволяє гарантувати, що гра відтворюватиметься на системах з підтримкою цієї версії DirectX та забезпечить необхідний функціонал;

– вимоги до місця на диску: зазначених 2 ГБ доступного місця на диску можна обґрунтувати як мінімальну кількість, необхідну для встановлення гри. Вона дозволяє гравцям з обмеженим дисковим простором відведеному для ігор встановлювати та грати в цю гру без проблем.

Зваживши ці чинники, вимоги до проєктованої системи було встановлено для гарантування функціонування гри на різних комп'ютерах, надаючи оптимальну продуктивність та візуалізації двовимірної графіки на задовільному рівні.

Висновки до першого розділу:

Впродовж аналізу аналогічних проєктів було визначено необхідний стек для розробки, а також спеціалізовані інструменти для вирішення задачі теми кваліфікаційної роботи: мову програмування C#, інтерфейс Scriptable Object та систему делегатів C# Action. Зважаючи всі плюси та мінуси було обрано Unity в ролі ігрового рушія. Вибір зумовлений наступними факторами:

Мова програмування C#:

- популярність;
- сучасна мова;
- кросплатформність;

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		28

- спеціальні інтеграції.

Scriptable Object:

- параметризація даних;
- відокремлені дані;
- використання в редакторі Unity;
- підтримка серіалізації.

C# Action:

- гнучкість;
- простота використання;
- підтримка анонімних методів;
- збереження коду;
- розширюваність.

Unity як ігровий рушій:

- популярність;
- програмування на C#;
- кросплатформність;
- спеціальні інтеграції.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		29

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

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

Перед початком розробки проєкт необхідно розділити на окремі етапи загального проєктування, що міститимуть в собі важливі елементи для покращення процесу розробки програмного продукту. В цьому розділі проводиться детальне планування та опис проєкту, що гарантує його якісну і успішну реалізацію. Крім цього було проведено розподілення ролей між учасниками команди, кожен з яких виконує конкретну задачу та працює над окремою частиною загального проєкту. Ролі учасників наведені в таблиці 2.1.

В цьому розділі буде представлено наступні частини:

1. Технічне завдання: містить детальний опис характеристик та функціональних вимог до програмного продукту. Цей опис включає інформацію про цільову аудиторію, необхідні інтеграції та інші релевантні деталі, які допомагають точно уявити та зрозуміти проєкт.
2. Методики взаємодії з ворожими об'єктами: в галузі комп'ютерних ігор та симуляцій вивчає та розробляє технології, які дозволяють моделювати та керувати взаємодією з ворожими об'єктами в ігрових середовищах.

Кожен учасник команди приймає на себе кілька ролей та типів завдань для виконання. Вибір кожної ролі здійснювався на підставі вмінь та навичок кожного члена команди, враховуючи їх особисті побажання, і отримання згоди від кожного учасника. Розподіл ролей можна переглянути у таблиці 2.1. Ця кваліфікаційна робота обмежується на дослідженні тієї частини проєкту, що відображена у ролях та темі кваліфікаційної роботи.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		30

Таблиця 2.1 – Розподіл ролей

Учасник	Ролі
Харченко Віталій Андрійович	Team Leader, Lead Developer
Леут Назарій Віталійович	Developer, QA
Козакевич Наталія Олександрівна	Developer, Director
Буханевич Оксана Андріївна	Developer, Designer

2.1.1. Технічне завдання

Сформуємо технічне завдання для частини проєкту, що стосується розробки системи битв та ігрового інтелекту супротивників. Це дозволить сформувати представлення розробки частини групового проєкту.

1. Опис проєкту

"Dark Watcher" - це комп'ютерна гра у жанрі метроїдванія з елементами екшену та пригод, в стилі soulslike. Гравець – темний вартівний, який досліджує невідомий темний світ та бореться зі злими духами в ньому. Гра пропонує нелінійний переплетений світ, ігровий процес заснований на подоланні платформ з елементами розв'язання головоломок та сутичок з ворожими неігровими персонажами.

2. Основні функціональні вимоги

2.1.Бойова система

2.1.1. Реалістична бойова механіка

2.1.1.1.Різноманітність атак, ударів і захисту.

2.1.1.2.Використання близьких та віддалених магічних атак.

2.1.1.3.Можливість ухилятися та контратакувати.

2.1.2. Битви з босами

2.1.2.1.Розробка унікальних босів зі своїми стилями бою та слабкостями.

2.1.2.2.Використання унікальних підходів для перемоги над босами.

2.1.2.3.Поступове ускладнення битв з прогресом у грі.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		31

2.2.Ігровий інтелект супротивників

2.2.1. Реалістичні рухи та поведінка супротивників.

2.2.1.1.Різні типи супротивників зі своїми унікальними навичками та атаками.

2.2.1.2.Здатність супротивників реагувати на дії гравця та змінювати свою стратегію.

2.2.2. Агресивна та оборонна поведінка

Супротивники можуть атакувати гравця, захищатися від його атак або намагатися уникати зіткнень.

3. Технічні вимоги

3.1.Операційна система: Windows 7.

3.2.Процесор: Intel Core 2 Duo E5200.

3.3.Оперативна пам'ять: 4 ГБ.

3.4.Відеокарта: GeForce 9800GTX+ з 1 ГБ пам'яті

3.5.DirectX: версія 10.

3.6.Місце на диску: 2 ГБ доступного місця.

4. Завдання для розробника

4.1.Розробка механік персонажу для ведення бою.

4.2.Реалізація системи нагород після перемоги над противниками.

4.3.Програмування механік супротивників та їхньої поведінки.

4.4.Створення рівнів та перешкод для подолання.

4.5.Тестування та налагодження гри з метою виявлення та усунення помилок.

5. Критерії успішності

5.1.Функціонування механік бою.

5.2.Плавне та швидке реагування противників.

5.3.Працездатне випадіння та отримання нагород.

5.4.Успішне тестування, відсутність помилок.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Це технічне завдання визначає потік роботи над розробкою гри та допомагає визначити вимоги до розроблюваної гри, завершеності та забезпечення інструментів для реалізації.

2.1.2. Методики взаємодії з ворожими персонажами.

Методики взаємодії з ворожими об'єктами в галузі комп'ютерних ігор та симуляцій широко використовуються і включають в себе проджектайли, gaucast та колайдери [3]. Проджектайли моделюють рух летючих об'єктів, таких як кулі або ракети, та їх взаємодію з оточенням. Rucast використовується для обчислення перетину променів з об'єктами та дозволяє виявляти зіткнення або моделювати освітлення. Колайдери визначають фізичні межі та властивості об'єктів і використовуються для виявлення зіткнень та взаємодії між ними.

Використання проджектайлів є важливою складовою при розробці ігор у Unity, однієї з найпопулярніших ігрових рушіїв. Проджектайли відповідають за моделювання руху та взаємодії летючих об'єктів у грі, таких як кулі, ракети, стріли або заклинання. Ця техніка дозволяє створювати захоплюючі битви, засновані на фізиці, та візуальні ефекти.

У Unity використовується система компонентів для реалізації проджектайлів. Зазвичай кожен проджектайл має компонент руху, який відповідає за обчислення його шляху та швидкості руху. Цей компонент встановлюється на об'єкт, який представляє проджектайл у грі.

Після створення проджектайла встановлюються його початкові координати і швидкість. Зазвичай ці значення задаються на початку або під час пострілу гравцем або іншим об'єктом. Кожен кадр симуляції рушієм Unity оновлює положення проджектайла, використовуючи його швидкість та напрямок руху.

Одна з ключових функцій проджектайлів – виявлення зіткнень з іншими об'єктами у грі. У Unity це досягається за допомогою компонента ко-

лайдера, який додається до об'єктів, та з якими можуть взаємодіяти проджектайли. Колайдери визначають форму та розміри об'єкту, а також визначають області зіткнень.

При зіткненні проджектайла з колайдером іншого об'єкта виконуються певні дії. Це може включати віднімання здоров'я, нанесення пошкоджень, виклик подій або взаємодію з іншими об'єктами. Взаємодія може бути реалістичною, враховуючи фізичні параметри об'єктів, або стилізованою для створення ефектів чи особливих механік гри.

Для досягнення бажаного візуального ефекту, проджектайли можуть супроводжуватись частинками, світловими ефектами, вибухами або анімаціями. Unity надає різноманітні інструменти та можливості для розробки красивих та реалістичних ефектів взаємодії з проджектайлами.

Застосування проджектайлів у грі можуть бути дуже різноманітними, від пострілів гравця до атак монстрів або заклинань магів. Ця техніка дозволяє створювати динамічні та захоплюючі битви, які залучають гравця до геймплею та надають реалістичності та візуальної привабливості грі.

Загалом, використання проджектайлів у Unity є потужним інструментом для створення взаємодії з ворожими об'єктами у грі. Вони додають реалістичність, ефективність та візуальну привабливість, що робить геймплей більш захоплюючим та зацікавлюючим для гравців.

При розробці гри у Unity, використання техніки raycast, також відомої як "променева перевірка" або "променевий вибух", може бути надзвичайно корисним для різних аспектів ігрового процесу та візуальних ефектів. Raycast використовується для виявлення перетину променя з об'єктами у просторі і може бути використаний для виявлення зіткнень, освітлення, візуалізації лазерів та багато іншого.

У Unity реалізація raycast базується на використанні класу Raycast, який дозволяє кинути промінь з початкової точки в певному напрямку та виявити перетин з колайдерами об'єктів. Клас Raycast має різні методи та

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		34

властивості, що дозволяють вам виконувати різні дії, пов'язані з розрахунками променя та отриманням результатів.

Одним з основних застосувань Raycast є виявлення зіткнень з об'єктами. Наприклад, коли гравець наводиться на ціль або спробує стріляти, можна сконструювати промінь від позиції гравця у напрямку цілі або в напрямку пострілу. За допомогою Raycast можна виявити першу точку перетину променя з колайдером і отримати інформацію про зіткнення, таку як позиція зіткнення, нормаль колайдеру, відстань від початкової точки тощо. Це дозволяє виконувати різні дії відповідно до результатів, наприклад, наносити пошкодження, запускати анімацію, взаємодіяти з ігровими об'єктами та інше.

Крім того, raycast може бути використаний для освітлення та візуалізації. Наприклад, можна використовувати промені для визначення видимих областей у грі та розрахунку освітлення від джерел світла. Також raycast може використовуватись для візуалізації лазерів, пеленгів, відбиття світла та інших ефектів, де точність і точне визначення перетину променя є важливими.

Використання raycast у Unity вимагає визначення початкової точки променя, напрямку та інших параметрів, які впливають на його поведінку. Ви можете контролювати кількість променів, їх напрямки, довжину та інші параметри для досягнення потрібних ефектів та рівня деталізації.

Загалом, використання raycast у розробці гри на Unity відкриває широкі можливості для виявлення зіткнень, освітлення та візуалізації. Вона дозволяє реалістично моделювати фізичні взаємодії та створювати захоплюючі ефекти, які підвищують якість геймплею та візуальне враження гравців.

Методики взаємодії з ворожими об'єктами в галузі комп'ютерних ігор та симуляцій широко використовуються і включають в себе проджектайли, raycast та колайдери. Проджектайли моделюють рух летючих об'єктів, таких як кулі або ракети, та їх взаємодію з оточуючим середовищем. Raycast вико-

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		35

ристовується для обчислення перетину променів з об'єктами та дозволяє виявляти зіткнення або моделювати освітлення. Колайдери визначають фізичні межі та властивості об'єктів і використовуються для виявлення зіткнень та взаємодії між ними.

Використання цих методик залежить від потреб ігрового проєкту. Проджектайли можуть бути використані для реалістичного моделювання стрільби або кидання предметів. Raycast може бути використаний для створення складних візуальних ефектів, таких як лазерні промені або ефекти світла. Колайдери є необхідним елементом для моделювання фізичних властивостей об'єктів та їх взаємодії, таких як стикання або відштовхування.

Ці методики є лише деякими з численних технологій, які застосовуються в розробці ігор та симуляцій для створення реалістичного геймплею та візуальних ефектів. Вони дозволяють взаємодіяти з ворожими об'єктами та створювати захоплюючі ігрові враження.

2.2 Розробка супротивників

Розробка рядових супротивників є важливою складовою частиною процесу створення відеогри. Рядові супротивники – це персонажі або сутності, які вступають у сутички з персонажем гравця.

Перший крок у розробці рядових супротивників – це проєктування їхнього зовнішнього вигляду та характеристик. Важливо визначити їх вигляд, анімацію, рухи та можливі дії, такі як атака або захист. Здатність рядових супротивників до взаємодії з гравцем, стратегія атаки та поведінка відіграють важливу роль у сприйнятті гри [5].

Після проєктування наступним кроком є реалізація ворогів. Це включає створення анімаційних спрайтів, їх програмування та інтеграцію з системою фізики Unity. Крім цього необхідно реалізувати логіку поведінки: переміщення, реакція на дії гравця та атаки [10].

Процес тестування є неодмінною частиною розробки супротивників. Тестування допомагає виявити та виправити помилки, забезпечити баланс

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		36

гри та забезпечити належну взаємодію ворогів з гравцем, що включає включати тестування їхнього ігрового інтелекту в залежності від дій гравця.

Невід’ємною складовою розробки системи битв є балансування. Характеристики супротивників, такі як сила атаки, швидкість дій та переміщення повинні відповідати рівню складності гри а кожен противник мати власний підхід до перемоги над ним.

Розробка супротивників та босів у програмному забезпеченні може бути складним завданням, яке вимагає ретельного підходу та використання ефективних інструментів. Використання C# Action є одним з методів, які дозволяють створювати багатосторонніх та гнучких супротивників.

При розробці супротивників, спочатку необхідно визначити їх характеристики та поведінку. C# Action може бути використаний для опису різних дій, які супротивники можуть виконувати. Наприклад, можна створити Action для атаки, відступу, використання навичок тощо. Це дозволяє задати поведінку супротивника на основі його характеристик та поточної ситуації в грі.

Після визначення характеристик та поведінки супротивника, можна перейти до реалізації цих дій у коді. Використання C# Action дозволяє створювати функціональні об’єкти, які можна передавати і викликати відповідно до потреб гри. Наприклад, можна створити Action для атаки супротивника і викликати його, коли супротивник нападає на гравця.

Крім того, C# Action може бути використаний для реалізації босів – могутніх та складних супротивників, які виконують велику кількість різних дій та мають унікальні характеристики. Завдяки гнучкості C# Action, можна створити складні поведінкові шаблони для босів, де вони можуть виконувати послідовність дій або реагувати на певні події у грі.

Розробка супротивників та босів з використанням C# Action дозволяє створити гнучких та ефективних противників у програмному забезпеченні. Використовуючи цей підхід, розробники можуть створювати реалістичну та

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		37

захоплюючи геймплей-динаміку, де гравці змушені боротися з різноманітними ворогами, кожен з яких має свої унікальні властивості та стратегію.

Для кращого розуміння роботи над бізнес процесом рекомендуємо ознайомитись з діаграмою бізнес-процесу противника-боса. Варто зазначити, що на діаграмі зображено лише базову методику роботи, та лише одну фазу битв з таким персонажем. Переглянути діаграму, можна на рисунку 2.1.

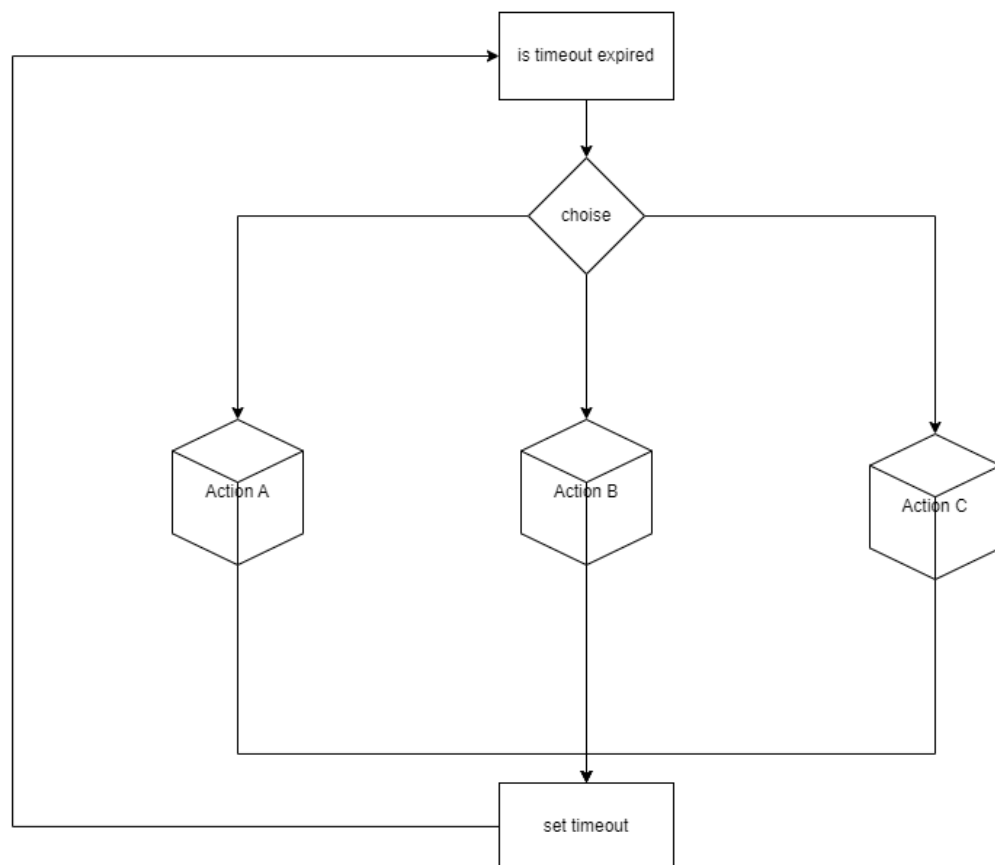


Рисунок 2.1 – Діаграма бізнес процесу супротивника боса.

2.3. Проектування та реалізація алгоритмів роботи системи

Успішне проектування та реалізація алгоритмів роботи системи є невід’ємною складовою в розробці будь-якого програмного продукту або системи. З точки зору тестувальника, існує кілька ключових аспектів, що підкреслюють важливість цього етапу.

Перш за все, ефективно спроектовані та оптимізовані алгоритми дозволяють переконатися в ефективному функціонуванні системи. Під час тестування можна виявити можливі проблеми з продуктивністю та швидкістю системи, а також забезпечити оптимальне використання ресурсів.

Надійність системи також залежить від проектування надійних алгоритмів. Тестування алгоритмів допомагає перевірити систему на наявність помилок, вразливостей або ситуацій зі збоєм, які можуть призвести до некоректної роботи системи. Це забезпечує високу якість та надійність роботи системи.

Чітко визначені та розроблені алгоритми спрощують процес тестування системи. Наявність грамотно спроектованих алгоритмів полегшує розуміння системи тестувальниками і дозволяє швидше виявляти та виправляти помилки. Крім того, ці алгоритми допомагають зменшити час, необхідний для проведення тестування, та полегшують супровід та розширення функціональності системи.

З точки зору розробника, ефективність також є важливим аспектом. Розробники прагнуть створювати алгоритми, що працюють швидко та ефективно, використовуючи мінімальну кількість ресурсів. Це допомагає задовольнити потреби користувачів та підвищити конкурентоспроможність системи на ринку.

Надійність є ще одним аспектом, який розробники враховують при проектуванні алгоритмів. Створення надійних алгоритмів, що працюють безперебійно та уникати помилок, допомагає забезпечити стабільну роботу системи та запобігти ситуаціям зі збоєм або втратою даних. Ретельне тестування алгоритмів гарантує їхню надійність.

Грамотне проектування алгоритмів також сприяє зменшенню складності управління системою для розробників [7]. Якщо алгоритми чітко визначені та легко розуміються, розробники можуть швидше розробляти та налагоджувати систему. Це полегшує супровід, підтримку та розширення функціональності системи в майбутньому.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Крім того, розробники повинні створювати алгоритми, які дозволяють системі масштабуватися та працювати зі зростаючим обсягом даних або користувачів. Тестування алгоритмів допомагає перевірити, наскільки система впорається зі зростаючим навантаженням. Розробники повинні забезпечити масштабованість системи та переконатися, що вона може ефективно функціонувати при збільшенні обсягів даних чи навантаження.

Отже, проектування та реалізація алгоритмів є ключовим етапом у розробці програмного продукту чи системи, який дозволяє забезпечити ефективність, надійність, складність управління та масштабованість системи. Грамотні алгоритми гарантують ефективне та стабільне функціонування системи, забезпечуючи задоволення користувачів та конкурентні переваги на ринку.

Діаграма активності "Взаємодія з противником" є цінним інструментом для уявлення послідовності дій, які відбуваються під час взаємодії гравця з противником у грі. Вона дозволяє описати сценарій взаємодії в діловому стилі, забезпечуючи чітку візуалізацію взаємодій та поведінки сторін.

Основне призначення діаграми активності "Взаємодія з противником" полягає в детальному описі послідовності дій, які виконуються під час взаємодії між гравцем і противником, що дозволяє чітко вказати, які дії здійснюються і в якому порядку, забезпечуючи логічний розпорядок подій.

Діаграма допомагає розібратися у різних взаємодіях, які відбуваються між гравцем і противником. Вона надає можливість з'ясувати, які дії виконуються з обох сторін і визначити характерну поведінку противника в різних ситуаціях.

Також є можливість аналізу послідовності подій, які відбуваються під час взаємодії з противником. Це допомагає виявити потенційні проблеми, помилки або недоліки в поведінці противника або гравця, що дозволяє внести необхідні зміни для поліпшення геймплею.

Загалом, діаграма активності "Взаємодія з противником" є важливим інструментом для проектування та аналізу взаємодії гравця з противником

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		40

у грі. Вона допомагає краще розуміти та вдосконалювати логіку, поведінку та геймплей відносно противника, забезпечуючи захоплюючий досвід гри.

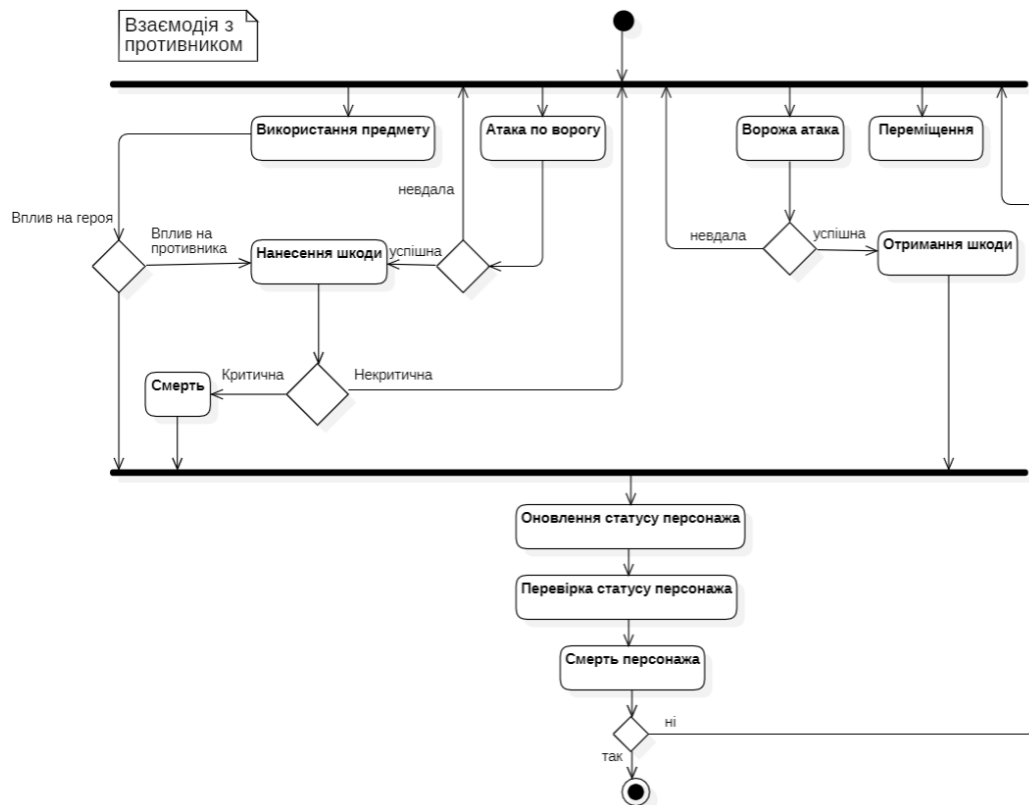


Рисунок 2.2 – Діаграма активності «Взаємодія з противником»

Діаграма активності "Оновлення статусу персонажа" використовується для моделювання послідовності активностей та процесів, пов'язаних з оновленням статусу персонажа в програмному продукті або системі. Ця діаграма надає візуальне представлення дій та взаємодій, які відбуваються для зміни стану персонажа на різних етапах гри або додатку. Основна мета діаграми активності - показати логіку та послідовність дій, які здійснюються для оновлення статусу персонажа

Діаграма активності "Оновлення статусу персонажа" використовується для візуалізації та моделювання процесу оновлення статусу персонажа в програмному продукті, такому як гра або додаток. Головна ціль цієї діаграми - показати послідовність дій та процесів, які відбуваються для зміни статусу персонажа.

Діаграма активності дозволяє розібрати процес оновлення статусу персонажа на окремі дії, взаємозв'язки між ними та прийняття рішень на основі різних умов.

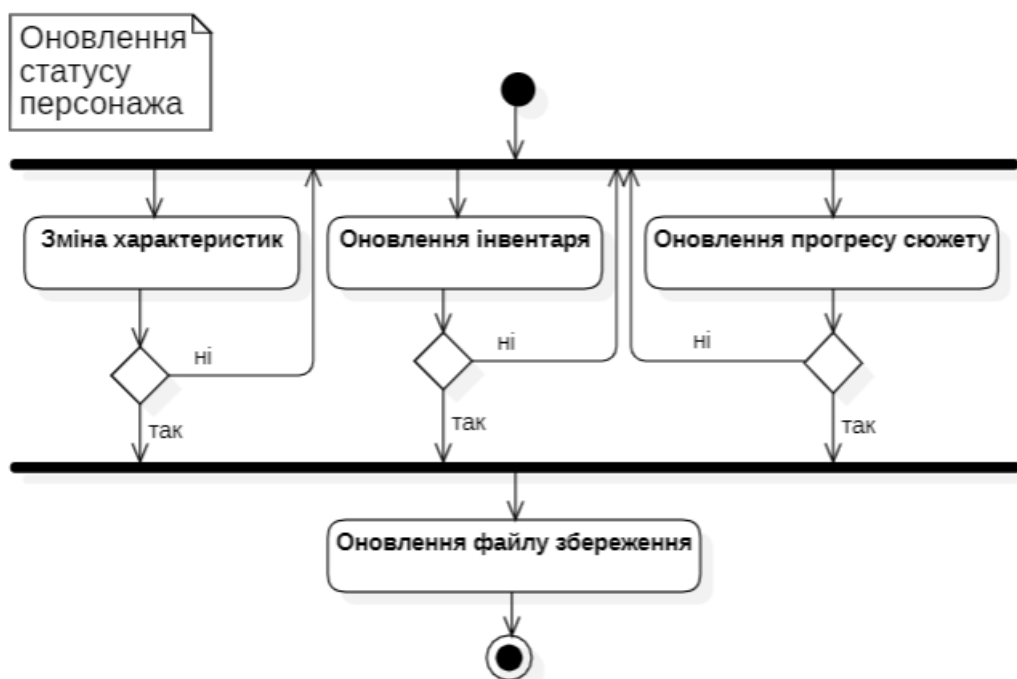


Рисунок 2.3 – Діаграма активності «Оновлення статусу персонажа»

Діаграма послідовності є одним з типів діаграм, який використовується для моделювання взаємодії між об'єктами або компонентами системи у певній послідовності. Вона дозволяє візуалізувати, як об'єкти взаємодіють між собою шляхом обміну повідомленнями, звернень або викликів методів.

Головна мета діаграми послідовності – показати порядок виконання взаємодії між різними об'єктами системи та часові лінії, на яких вони відбуваються. Це дозволяє розібрати послідовність взаємодій між об'єктами та виявити потенційні проблеми або помилки в логіці програми.

Діаграма послідовності дозволяє розібрати логіку взаємодії об'єктів у системі, ідентифікувати потребу в методах або функціональності, виявити залежності та послідовність дій для успішного виконання завдання. Вона є важливим інструментом при проєктуванні та аналізі систем, особливо при розробці програмного забезпечення.

Діаграма послідовності використовує наступні елементи:

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		42

- об'єкти: представлені прямокутниками з назвою об'єкта всередині. Кожен об'єкт представляється як учасник взаємодії;
- життєві лінії: відображають послідовність дій та взаємодій між об'єктами. Показують, коли об'єкти активні та коли вони взаємодіють між собою;
- повідомлення: представлені стрілками, які показують виклики методів або передачу повідомлень між об'єктами. Вони вказують напрямок та порядок передачі даних.

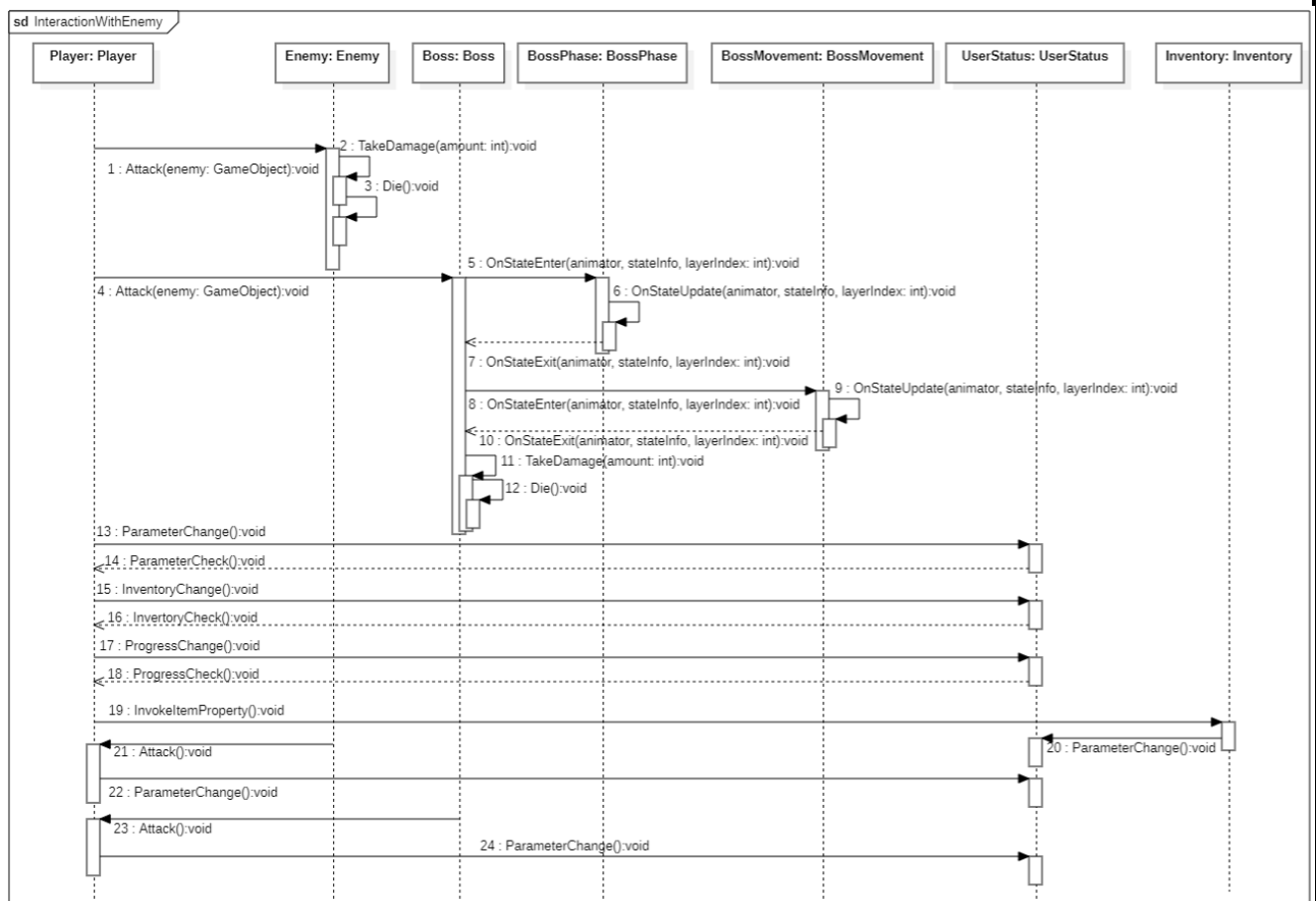


Рисунок 2.4 – Діаграма послідовності

Діаграма компонентів є структурною діаграмою та використовується для моделювання структури системи, а саме її компонентів та взаємозв'язків між ними. Вона дозволяє візуально представити архітектурний розподіл компонентів системи та їх взаємозв'язки.

Головна мета діаграми компонентів – показати структуру системи на високому рівні абстракції, відображаючи компоненти, з яких складається

система, та зв'язки між ними. Компонент може представляти собою окремий модуль, підсистему, бібліотеку, клас або функціональну одиницю, яка виконує певні завдання в системі.

Діаграма компонентів допомагає розібрати структуру системи, виділити основні компоненти та їх взаємозв'язки, визначити модульність системи та можливість заміни або розширення компонентів. Вона є важливим інструментом при проєктуванні архітектури програмного забезпечення, дозволяє зрозуміти взаємодію компонентів та спрощує подальше розроблення та тестування системи.

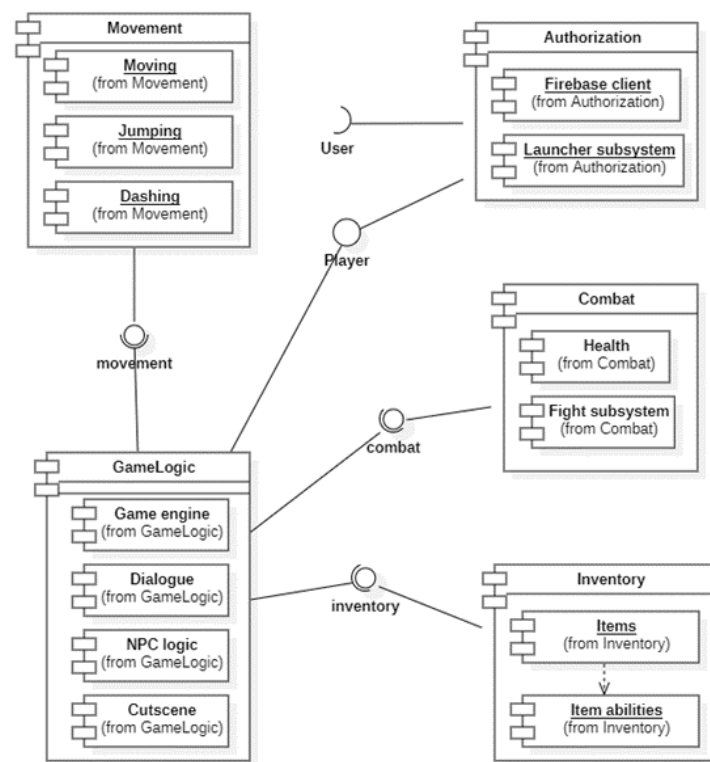


Рисунок 2.5 – Діаграма компонентів

Діаграма компонентів використовує наступні елементи:

- компоненти: представлені у вигляді прямокутників з назвою компонента всередині. Кожен компонент відображається як окрема частина системи зі своїм функціоналом.
- зв'язки: вказують на взаємозв'язки між компонентами. Це можуть бути залежності, використання інших компонентів, спадкування або інші типи взаємодій.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		44

– інтерфейси: показують спосіб взаємодії зовнішнього середовища з компонентом, описують доступні методи, параметри та повернені значення.

Діаграма розгортання – використовується для візуального зображення фізичного розташування компонентів системи та їх взаємозв'язків на апаратному рівні. Вона відображає, як компоненти системи фізично розташовані на обладнанні, такому як сервери, комп'ютери, мережеві пристрої та інші пристрої.

Основна мета діаграми розгортання полягає в показі структури та конфігурації фізичних компонентів системи, включаючи апаратне забезпечення, мережеві взаємозв'язки, середовища виконання та їх взаємодію. Вона надає інформацію про фізичну архітектуру системи та розташування компонентів на різних вузлах апаратного забезпечення.

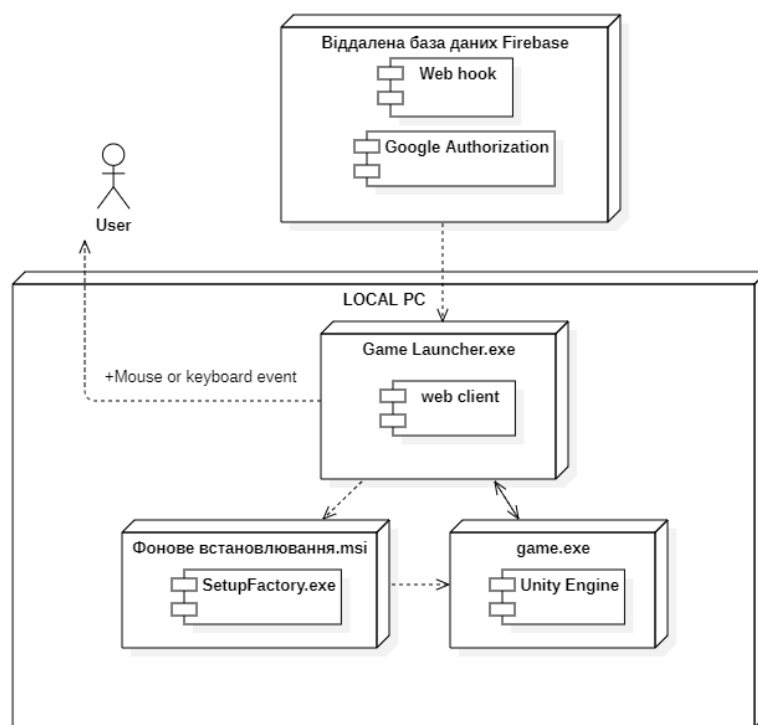


Рисунок 2.6 – Діаграма розгортання

2.4. Реалізація ігрового додатку

Розробка відеогри, як і будь-якого проєкту, – складний процес, що містить різні етапи та задачі.

Перший етап – це визначення концепції гри. Встановлюється загальна ідея гри, визначають її жанр, цільову аудиторію та головні механіки гри.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Після того, як концепція визначена, відбувається проєктування, що включає розробку сюжету, геймплею, рівнів, персонажів, музики, звуків та інших елементів гри.

Після завершення проєктування розпочинається фактична розробка програмного забезпечення. Цей етап включає програмування, створення графіки, анімації, звукового супроводу та всіх необхідних елементів.

На етапі, коли розробка завершена, система проходить етап тестування. Тестування включає виявлення та виправлення помилок, перевірку на відповідність гральним механікам, збалансованість та загальну якість гри.

Після успішного тестування і виправлення всіх помилок гра готова до випуску. Розробники підготовлюють необхідні матеріали для рекламної кампанії, друку та розповсюдження гри. Гра може бути випущена на різних платформах, таких як персональні комп'ютери, ігрові консолі або мобільні пристрої.

Випустивши гру, розробники можуть продовжувати підтримувати її шляхом випуску оновлень та додаткового контенту, включно з виправленням помилок, вдосконаленням ігроладу, додаванням нових рівнів, персонажів або режимів гри, а також взаємодію з гравцями через форуми чи онлайн-служби.

Система битв гри повинна включати в себе взаємодію гравця з ворожими персонажами, для чого знадобиться реалізація відповідного функціоналу та механік як для головного героя, так і для кожного ворога [4]. Під час битв гравець може наносити шкоду ворогам, отримувати шкоду від пропущених ударів, відновлювати очки здоров'я, мана, створювати заклинання зі здатністю нанесення шкоди ворогам чи лікування. В разі отримання критичної шкоди гравець помирає та відроджується на останній точці збереження. Вороги в свою чергу можуть атакувати гравця, отримують шкоду від його ударів, а при досягненні критичної шкоди помирають й мають можливість залишити після себе нагороду.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Для реалізацій ключових механік гравця, що стосуються бойової системи гри, було створено окремий клас CombatScript, де було реалізовано їх.

Таблиця 2.2 – Опис методів класу CombatScript

Назва	Призначення
Update()	Перевіряє можливість проведення атаки та викликає відповідні натисненням клавішам дії, за можливості
Attack()	Викликає анімацію атаки за допомогою відповідного триггеру та програє відповідний звуковий ефект
Damage()	Перевіряє чи влучила атака по противникам, якщо так – кожен з них отримує шкоду в розмірі атаки гравця. Нанесена шкода додається до статистики.
CastCrystal()	За наявності очок мани, створює ігровий об'єкт кристалу та витрачає відповідну ціні кількість очок мани.
CastAOE()	За наявності очок мани, наносить шкоду в розмірі атаки гравця всім ворогам в радіусі заклинання та витрачає відповідну ціні кількість очок мани. Нанесена шкода додається до статистики.
OnDrawGizmosSelected()	Відображає зону атаки та зону досягнення заклинання зі шкодою по області у вигляді кола при вибраному ігровому об'єкту в редакторі Unity.

Для врахування максимальної та поточної кількості здоров'я та мани персонажа, а також обробки зміни їх значень було створено клас

PlayerHealth. Завдяки цьому класу буду здійснюватися механіки, необхідні під час бою з ворогами.

Таблиця 2.3 – Опис методів класу “PlayerHealth”

Назва	Призначення
TakeDamage(int damage)	Якщо гравець не є невразливим, від його поточного здоров'я віднімається вказане значення, викликається метод heartsControl(), викликається анімація та звуковий ефект отримання шкоди, отримана шкода враховується в статистику. В разі досягнення поточного здоров'я 0 чи нижче, викликається метод Die().
SpendMana(int amount)	Поточна мана гравця зменшується на вказане значення, викликається метод heartsControl().
RestoreHealth(int amount)	Збільшує поточне здоров'я гравця на вказане значення, проте не вище максимального, програв звуковий ефект зцілення.
RestoreMana(int amount)	Збільшує поточні очки мани гравця на вказане значення, проте не вище максимального, програв звуковий ефект зцілення.
Die()	Викликає анімацію смерті, змінює значення параметру isDead на true, збільшує кількість смертей у статистиці, програв звуковий ефект смерті.
LoadScene()	Перезавантажує поточну сцену.

Продовження таблиці 2.3

DamageAnimation()	Перемикає кадри анімації, використовуючи асинхронні події Unity.
Start()	Ініціалізує змінні, оновлює шкалу здоров'я та мани.
Update()	Візуальне відображення зміни шкал очків здоров'я та мани.
heartsControl()	Метод-контролер шкал здоров'я та мани.

Щоб впровадити аналогічну логіку для ворожих персонажів було створено клас Enemy.

Таблиця 2.4 – Опис методів класу “Enemy”

Назва	Призначення
Start()	Ініціалізує змінні.
HitEffect()	Відштовхує противника після отримання шкоди.
TakeDamage(int damage)	Якщо противник не є невразливим, від його поточного здоров'я віднімається вказане значення, викликається метод HitEffect(), викликається метод DamageAnimation() та звуковий ефект отримання шкоди. В разі досягнення поточного здоров'я 0 чи нижче, викликається метод Die().
Die()	Викликає анімацію повідомлення про перемогу, змінює значення параметру isDead на true, програє звуковий ефект смерті. За наявності нагороди, розміщує предмет на ігровій сцені.
DamageAnimation()	Перемикає кадри анімації, використовуючи асинхронні події Unity.

Для повноцінної системи битв необхідна логіка дій противників, що відштовхується від поточної ситуації у грі. З такою метою було створено клас EnemyAI, який відповідає за переміщення ворожих персонажів під час

їхньої взаємодії з гравцем в залежності від їхньої позиції та позиції персонажа.

Таблиця 2.5 – Опис методів класу “EnemyAI”

Назва	Призначення
Start()	Ініціалізує змінні, викликає метод InvokeRepeating().
UpdatePath()	Оновлює поточний шлях в залежності від позиції цілі та поточної позиції.
OnPathComplete(Path p)	Скидає поточну позицію до нуля.
FixedUpdate()	Якщо кінцеву точку шляху не досягнуто, обраховується наступна точка для руху та проводиться переміщення.

За ключові битви гравця необхідно нагороджувати, щоб заохотити кидати виклики складнішим супротивникам та для подальшого просування по сюжету, наприклад отримавши предмети, які додають варіативності до ігрового процесу чи дозволяють потрапити у раніше недоступні області ігрового світу.

З метою можливості реалізації механік пов’язаними з отриманнями нагород було розроблено класи Loot, LootBag та LootPickup. Вони відповідають за створення об’єктів нагород та логіку їхнього випадіння з противників.

Таблиця 2.6 – Опис методу класу “Loot”

Назва	Призначення
Loot(string lootName, int dropChance)	Конструктор для створення об’єкту нагороди

Таблиця 2.7 – Опис методів класу “LootBag”

Назва	Призначення
GetDroppedItem()	Випадковим чином обирає нагороду з доступних, в залежності від шансу випадіння.
InstantiateLoot(Vector2 spawnPosition)	Викликає метод GetDroppedItem() після чого розміщує обрану нагороду на ігровій сцені.

Таблиця 2.8 – Опис методів класу “LootPickup”

Назва	Призначення
Start()	Знищує об’єкт нагороди, в разі наявності ідентичної в гравця.
OnTriggerEnter2D(Collider2D hitInfo)	Додає нагороду в інвентар гравця, зберігає інвентар, програє звуковий ефект отримання нагороди та виводить відповідне повідомлення на екран.

Для рядових противників, а саме їхньої логіки ведення бою, було створено клас Monster.

Таблиця 2.9 – Опис методів класу “Monster”

Назва	Призначення
Attack()	Нанесення шкоди гравцю, в разі його присутності у зоні нанесення шкоди.
OnDrawGizmosSelected()	Відображає зону атаки у вигляді кола при вибраному ігровому об’єкту в редакторі Unity.

Для ускладнених ключових противників – босів, було вирішено розробити абстрактний клас IBoss, в якому міститимуться всі необхідні поля та методи, що будуть реалізовані для кожного унікального боса індивідуально. Таким чином класи кожного боса відрізнятимуться один від одного. Для демонстрації роботи наведемо класи для боса «Wolf», який у свою чергу має 2 фази.

Таблиця 2.10 – Опис методів класу “WolfPhase1”

Назва	Призначення
void Start()	Ініціалізація змінних та створення екшенів.
Action jump	Виклик триггеру Jump
Action bite	Виклик триггеру Bite
Action walk	Виклик триггеру Walk
Action paw	Виклик триггеру Paw
Action howl	Виклик триггеру Howl
Update()	Виклик методів detectPlayer() та LookAtPlayer(), обрання можливої дії в залежності від шансу та виклик відповідного екшену.
detectPlayer()	Отримання поточної позиції гравця.
LookAtPlayer()	Виконання повороту обличчям до гравця.
Attack(Hitbox hitbox)	Нанесення шкоди гравцю, в разі його присутності у вказаній зоні нанесення шкоди.

Продовження таблиці 2.10

Bite()	Виконання переміщення під час екшену bite
Jump()	Виконання переміщення під час анімації Jump.
BiteAttack()	Виклик функції Attack() для зони нанесення шкоди під час екшену bite.
JumpAttack()	Виклик функції Attack() для зони нанесення шкоди під час екшену jump.
PawAttack()	Виклик функції Attack() для зони нанесення шкоди під час екшену paw.
OnDrawGizmosSelected()	Відображає зони нанесення шкоди у вигляді кол при вибраному ігровому об'єкті в редакторі Unity.

Таблиця 2.11 – Опис методів класу “WolfPhase2”

Назва	Призначення
void Start()	Ініціалізація змінних та створення екшенів.
Action headDrop	Виклик триггеру HeadDrop.
Action headDownWalk	Виклик триггеру HeadDownWalk.
Action headShake	Виклик триггеру HeadShake.
Action run	Виклик триггеру Run.
Update()	Виклик методів detectPlayer() та LookAtPlayer(), обрання можливої дії в залежності від шансу та виклик відповідного екшену.
detectPlayer()	Отримання поточної позиції гравця.

LookAtPlayer()	Виконання повороту обличчям до гравця.
Attack(Hitbox hitbox)	Нанесення шкоди гравцю, в разі його присутності у вказаній зоні нанесення шкоди.
HeadlessAttack()	Виклик функції Attack() для зони нанесення шкоди під час екшену headDrop та headDownWalk.
OnDrawGizmosSelected()	Відображає зони нанесення шкоди у вигляді кол при вибраному ігровому об'єкті в редакторі Unity.
OnDestroy()	Записує факт перемоги в досягнення.

У певних випадках є необхідність для розділення функціоналу на окремі підскрипти у ролі машини-станів, що дає дозволяє більш зручно керувати анімаціями під час роботи та дозволяє візуально продемонструвати конкретну механіку. Таким чином було реалізовано стани різних фаз, їх переходу а також станів анімацій для певних атак боса.

Таблиця 2.12 – Опис методів класу “Wolf_Phase1”

Назва	Призначення
OnStateEnter()	Викликає тригер переходу в наступну фазу якщо здоров'я досягло половини, інакше скидає стан InAction до false.

Таблиця 2.13 – Опис методів класу “Wolf_ShiftPhase”

Назва	Призначення
OnStateEnter()	Вимикає скрипт WolfPhase1 та вмикає WolfPhase2, зберігаючи при цьому значення змінної isFlipped.

Таблиця 2.14 – Опис методів класу “Wolf_Phase2”

Назва	Призначення
OnStateEnter()	Скидає стан InAction до false.

Таблиця 2.15 – Опис методів класу “Wolf_Headless_Walk”

Назва	Призначення
OnStateEnter()	Переміщує боса фізично, відповідно до анімації, вимкнувши гравітацію та задавши прискорення.
OnStateExit()	Повертає початкову гравітацію та прибирає прискорення.

Таблиця 2.16 – Опис методів класу “Wolf_Phase2”

Назва	Призначення
OnStateEnter()	Скидає стан InAction до false.

Оскільки різні атаки противників мають різні анімації, для точного розміщення зони нанесення шкоди було вирішено створити клас Hitbox. Він дозволяє розділити зони, в яких будуть спрацьовувати різні атаки.

Таблиця 2.17 – Опис методів класу “Hitbox”

Назва	Призначення
Hitbox(Transform attackPoint, int attackDamage, float attackRange)	Конструктор для створення об'єкту області нанесення шкоди.

Ще один приклад реалізації боса – противник, який створює об'єкти-вогняні кулі, які переслідують гравця та наносять шкоду. Такий тип босів заснований на системі станів, що є вбудованою в Unity.

Таблиця 2.18 – Опис методів класу “BossRun”

Назва	Призначення
OnStateEnter()	Ініціалізація змінних.
OnStateUpdate()	Виклик функції LookAtPlayer(), рух відносно гравця та виклик триггеру атаки, в разі досягнення.

OnStateExit()	Скидання триггеру Attack.
---------------	---------------------------

Таблиця 2.19 – Опис методів класу “Boss_Attack”

Назва	Призначення
OnStateEnter()	Ініціалізація точки атаки, виклик функції Shoot().
Shoot()	Виклик функції Instantiate() для створення об'єкту вогняної кулі.

Таблиця 2.20 – Опис методів класу “HomingFireball”

Назва	Призначення
Start()	Ініціалізація змінних, отримання позиції гравця.
FixedUpdate()	Обрахування напрямку та рух в сторону гравця.
OnTriggerEnter2D()	Нанесення шкоди гравцю в разі влучання, знищення об'єкта.

Гравець у свою чергу має змогу створення заклинань.

Таблиця 2.21 – Опис методів класу “Crystal”

Назва	Призначення
Start()	Ініціалізація, задання руху в напрямку повороту гравця.
OnTriggerEnter2D()	Нанесення шкоди противнику в разі влучання, знищення об'єкта.

Детальний опис Unity Event System, який задіяний при створенні таких типів супротивників наведено у кваліфікаційних роботах інших учасників команди.

Висновки до другого розділу

У цьому розділі було поставлено технічне завдання, згідно проаналізованих вимог до продукту, визначено потік роботи над подальшою розробкою відеогри. Також було розглянуто та проаналізовано різні механіки, які можна застосувати для взаємодії з ворожими персонажами у розроблюваній системі битв.

В процесі проєктування алгоритмів було враховано різноманітні аспекти, такі як функціонал, працездатність, безпека, зручність та зрозумілість експлуатації. Розроблено модель програмного комплексу бойової системи та системи супротивників та взаємодії з ними, системи отримання нагород, битв з босами. Це забезпечить її ефективність та працездатність під час майбутнього ігрового процесу.

В результаті, було визначено та реалізовано необхідні елементи системи відеогри, які задовольняють потреби та вимоги ігрового додатку.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		57

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ В ЖАНРІ МЕТРОЇДВАННЯ “DARK WATCHER”

3.1. Порядок встановлення та налаштування системи

Гра «Dark Watcher» є відеогрою, розробленою на платформі Unity 2D. Після випуску офіційної версії застосунок доступний для встановлення на будь-якому пристрої з підтримуваних релізних платформ. Основними платформами, для яких створювалась гра, є Windows 11 і MacOS. Незалежно від операційної системи, додаток буде встановлюватись ідентично. Щоб встановити гру, необхідно створити безкоштовний аккаунт в магазині цифрової дистрибуції «Steam» та увійти в додаток. Після цього необхідно перейти до на сторінку гри в крамниці вказати назву гри, користуючися пошуком та придбати цифрову копію ліцензії, або, за наявності ключа, ввести його у відповідне поле, вибравши опцію активації продукту в Steam. Після придбання, гра з'явиться у бібліотеці користувача, звідки він зможе її завантажити.

Приклад оптимальної системи з операційною системою Windows для досягнення мінімум 120 стабільних кадрів в секунду (fps) для комфортної гри:

1. Процесор (CPU): Intel Core i7 або AMD Ryzen 7 чи краще.
2. Графічна карта (GPU): NVIDIA GeForce GTX 1660 Super або AMD Radeon RX 5700 з обсягом відеопам'яті 6ГБ або більше.
3. Оперативна пам'ять (RAM): мінімум 8 гігабайт (ГБ) оперативної пам'яті.
4. Жорсткий диск (HDD) або твердотільний накопичувач (SSD): рекомендовано використовувати SSD для швидкого завантаження ігор та швидкого доступу до даних.
5. Монітор: роздільна здатність Full HD (1920x1080) з частотою оновлення 120 Гц або вище для плавності зображення.
6. Операційна система: Windows 11 64-біт.
7. Живлення: достатньо потужний блок живлення для живлення всієї системи, з урахуванням споживання енергії від компонентів.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Варто зазначити, що це лише орієнтовні характеристики системи, які гарантують максимальну ефективність, фактичні мінімальні вимоги було зазначені на етапі проектування.

3.2. Структура інтерфейсу та порядок роботи з системою битв

Необхідно відзначити, що в цьому розділі буде описана конкретна частина гри, яка була наведена у темі кваліфікаційної роботи. Інші частини можна переглянути у кваліфікаційних роботах інших членів команди.

Після запуску гри та переміщення гравця по рівню він може зіткнутися з ворожими персонажами, наприклад «трав'яним монстром» (рис 3.1), босом «Сокира» (рис 3.2), босом «Деревний монстр» (рис 3.3), босом «Вовк» (рис 3.4) чи іншими.



Рисунок 3.1 – Противник «трав'яний монстр»



Рисунок 3.2 – Бос «Сокира»

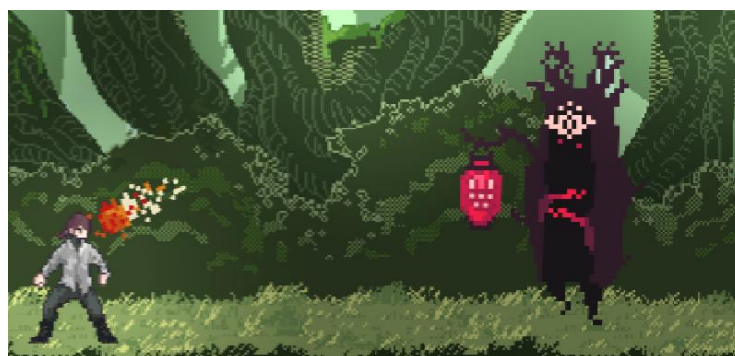


Рисунок 3.3 – «Деревний монстр»

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		59



Рисунок 3.4 – Бос «Вовк»

Гравець може виконувати атаки мечем перед собою, натиснувши кнопку «Enter» чи використовувати заклинання, натиснувши кнопку «F», які наносять шкоду противникам, в разі успішного влучання. Візуальна складова відображена на рисунках 3.5 – 3.7.



Рисунок 3.5 – Атака персонажа мечем



Рисунок 3.6 – Різновид атаки мечем



Рисунок 3.7 – Використання заклинання «Кристал»

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				60
Змн.	Арк.	№ докум.	Підпис	Дата		

В разі отримання гравцем шкоди чи витрачення очків мани відповідно оновлюється шкала в інтерфейсі користувача, що розміщений у верхній частині ігрового екрану (рис. 3.8).



Рисунок 3.8 – Зміна поточних очків здоров'я та мани на шкалі персонажа

В разі перемоги над босом відповідне повідомлення про перемогу виводиться на екран; в разі отримання нагороди за вбивство супротивника вона з'являється на ігровому екрані (рис. 3.9).



Рисунок 3.9 – Результат перемоги над босом

Коли гравець отримує нове спорядження, підібравши нагороду, отриману після перемоги над супротивником, на екран виводиться відповідне повідомлення (рис. 3.10).



Рисунок 3.10 – Повідомлення про отримання нового спорядження

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		61

3.3. Тестування роботи системи

Тестування ігрової системи – це процес перевірки та оцінки якості та функціональності ігрового продукту. Його основна мета - виявлення помилок, дефектів та проблем, що можуть виникнути під час гри, а також забезпечення задоволення користувачів.

До основних видів тестування [6] відеоігор можна віднести:

- функціональне тестування: перевірка роботи основних функцій гри, таких як керування персонажем, взаємодія з противниками;
- стабільності: перевірка вміння гри працювати без збоїв або перерв у роботі, включаючи тестування міцності, продуктивності та оптимізації;
- тестування ігроладу: перевірка якості ігрових механік та атмосфери з точки зору враження від гри;
- користувацьке тестування: перевірка зручності та ефективності інтерфейсу, включаючи навігацію, взаємодію з меню.
- Функціональне тестування містило такі тест-кейси:

Тестування зміни фази боса «Вовк»:

Таблиця 3.1 – Тест-кейс зміни фази

Test name: Wolf Boss Shift Phase (Issue id #15)			
Status	Passed		
Created date	2023-06-04	Tested at	2023-06-05
Reporter	Vitalii Kharchenko	Assigned to	Nazarii Leut
Pre Conditions: register new account			
Steps	Description	Expected results	
1	Enter the boss battle with the Wolf Boss.	The player character is positioned in the boss arena, and the Wolf Boss is in its initial phase.	
2	Attack the Wolf Boss using regular attacks.	The Wolf Boss's health decreases as a result of the player's attacks.	
3	Wait for the Wolf Boss to reach a certain health threshold.	The Wolf Boss's health drops to a specific value, indicating the transition to the shift phase.	

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		62

Продовження таблиці 3.1

4	Observe the transition of the Wolf Boss.	The Wolf Boss undergoes a visual transformation, indicating the shift to a different phase.
5	Analyze the new abilities and patterns of the Wolf Boss.	The Wolf Boss exhibits different attack patterns or abilities compared to the previous phase. The player character needs to adapt to these changes.
6	Continue battling the Wolf Boss in its shifted phase.	The player engages with the Wolf Boss using appropriate strategies and tactics, aiming to defeat it while avoiding its new attacks and abilities.

Тестування отримання нагороди:

Таблиця 3.2 – Тестування отримання нагороди

Test name: Loot Pickup (Issue id #16)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-06
Reporter	Vitalii Kharchenko	Assigned to	Leut Nazarii
Pre Conditions: Authorization/Registration			
Steps	Descriprion	Expected results	
1	Position the player character near the loot item.	The player character is in proximity to the loot item, which could be a chest, dropped item, or any other object containing loot.	
2	Interact with the loot item.	The player interacts with the loot item, such as by pressing a designated button or clicking on it, to initiate the pickup process.	
3	Observe the loot pickup animation or feedback.	The game provides visual and audio cues indicating that the loot is being picked up, such as an animation or sound effect.	
4	Add the loot to the player's inventory.	The loot item is removed from the game world and added to the player's inventory, ensuring it is now accessible to the player.	

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		63

Продовження таблиці 3.2

5	Update the player's statistics or resources.	If the loot item grants bonuses or affects the player's statistics (e.g., health, damage, or attributes), those values are updated.
6	Verify the presence of the loot in the player's inventory.	The player can check their inventory and confirm that the looted item is present, ready for future use or further interactions.

Тестування використання заклинання:

Таблиця 3.3 – Тестування використання заклинання

Test name: Spell Cast (Issue id #17)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-06
Reporter	Nataliia Kozakevych	Assigned to	Leut Nazarii
Pre Conditions: Authorization/Registration, game started			
Steps	Descripion	Expected results	
1	Launch the game and navigate to the spell casting menu.	The game's main menu is displayed, and the player is able to access the spell casting menu option.	
2	Select a spell from the available options.	The chosen spell is highlighted or selected, indicating that it's ready to be cast.	
3	Aim the spell at a target or area.	The player can aim the spell using the provided controls, directing it towards the desired target or location.	
4	Trigger the spell cast.	Upon initiating the spell cast, the corresponding visual and audio effects associated with the spell are displayed, indicating its activation.	
5	Observe the spell's impact on the target.	The spell effect interacts with the target or area, applying the intended outcome, such as damage, healing, or other defined spell properties.	
6	Verify the cooldown or resource depletion status.	After casting the spell, the player's resources or cooldowns associated with the spell are updated to reflect the appropriate values.	

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Тестування оновлення шкали здоров'я:

Таблиця 3.4 – Тестування оновлення шкали здоров'я

Test name: Health Bar Update (Issue id #18)			
Status	Passed		
Created date	2023-06-06	Tested at	2023-06-07
Reporter	Nataliia Kozakevych	Assigned to	Leut Nazarii
Pre Conditions: Authorization/Registration, game started			
Steps	Description	Expected results	
1	Begin the game or load a game save.	The game starts, and the player character's health bar is displayed on the screen, reflecting the initial value.	
2	Perform an action that affects the player's health.	The player character takes damage or receives healing, resulting in a change in their health value.	
3	Observe the health bar after the action is performed.	The health bar visually updates to reflect the new health value, accurately representing the change.	
4	Verify the display of critical health or status effects.	If the player character's health drops to critical levels or if there are any status effects, they are indicated.	
5	Confirm the accuracy of the health value and visuals.	The displayed health value matches the actual health of the player character, and the visuals are consistent.	

Тестування відображення повідомлення про перемогу:

Таблиця 3.5 – Тестування повідомлення про перемогу

Test name: Victory Message (Issue id #19)			
Status	Passed		
Created date	2023-06-06	Tested at	2023-06-08
Reporter	Oksana Bukhanevych	Assigned to	Leut Nazarii
Pre Conditions: Authorization/Registration, game started			
Steps	Description	Expected results	
1	Engage in the boss battle and defeat the boss.	The player character engages in combat with the boss and successfully defeats it.	

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Продовження таблиці 3.5

2	Observe the boss's defeat animation or visual cues.	The boss displays a specific defeat animation or other visual cues that clearly indicate its demise.
3	Display a message on the screen after the boss is defeated.	A message or notification appears on the screen, congratulating the player on defeating the boss or providing relevant information.
4	Include relevant details in the message.	The message may include details such as the boss's name, the player's achievement, rewards, or progression to the next stage.
5	Verify the appropriate transition or game state after dismissal.	After dismissing the message, the game transitions to the next stage, level, or relevant state as expected.

Тестування відображення повідомлення про отримання нового спорядження:

Таблиця 3.6 – Тестування повідомлення про отримання спорядження

Test name: Pickup Message (Issue id #20)			
Status	Passed		
Created date	2023-06-06	Tested at	2023-06-08
Reporter	Oksana Bukhanevych	Assigned to	Leut Nazarii
Pre Conditions: Authorization/Registration, game started			
Steps	Description	Expected results	
1	Open a loot container or defeat an enemy to obtain loot.	The player character interacts with a loot container, defeats an enemy, or performs any action that results in receiving loot items.	
2	Display a message on the screen after receiving the loot.	A message or notification appears on the screen, informing the player about the loot received or providing relevant information about it.	
3	Include details of the loot in the message.	The message includes information such as the names, quantities, rarities, or properties of the received loot items.	

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		66

4	Update the player's inventory with the received loot items.	After acknowledging the message, the loot items mentioned in the message are added to the player's inventory or appropriate storage system.
---	---	---

Тестування ігроладу було проведено і містило в собі перевірку механік, пов'язаних з битвами проти ворожиш неігрових персонажів, як рядових супротивників, так і складніших – босів, їхню поведінку та логіку дій. Також було протестовано можливість отримання нагород й оновлення різноманітних статусів на користувацькому інтерфейсі, які безпосередньо стосуються сутичок з ворогами. Атмосфера гри гарантує занурення і відповідає задумці розробників. В процесі перевірки наявного наповнення гри, не було виявлено несправностей, адже всі невідповідності були успішно скореговані в процесі активної розробки та проведення тестуванні системи.

Висновки до третього розділу

У цьому розділі була описана процедура встановлення та конфігурації відеогри, розробленої в результаті проведеної роботи. При цьому було висвітлено основні кроки взаємодії, від встановлення до ігрового процесу.

Перш за все було наведено орієнтовні характеристики системи для запуску проєкту з максимальною ефективністю для найкращого досвіду гравців. Також було ретельно описано порядок взаємодії гравця з супротивниками та систему битв у грі. Було розкрито основні механіки битв та надано детальний опис процесу взаємодії гравця з різними противниками.

Після цього система була піддана тестуванню, оцінена її на дієздатність. Виявлені в процесі розробки, які виникали під час розробки, були усунені з метою покращення роботи системи. Цей процес дозволив забезпечити високу якість та безперебійну роботу відеогри.

В результаті, проведеної в даному розділі роботи, було описано порядок дій при роботі з готовим додатком, ідентифіковано та виправлено наявні

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		67

й передбачено для подальшого уникнення будь-які недоліки для забезпе-
чення найкращого враження від розробленої відеогри.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		68

ВИСНОВКИ

В процесі розробки кваліфікаційної роботи бакалавра було спроектовано комп'ютерну гру в жанрі метроїдванія «Dark Watcher». Бажаних результатів було досягнуто завдяки крокам, наведеним нижче:

- було проведено аналіз предметної області та наявних аналогів, що дозволило визначити потреби користувачів і встановити основні механіки та характеристики гри. Було обрано Unity як ігровий рушій, а також мову програмування C#, інтерфейс Scriptable Object та систему делегатів C# Action;

- встановлено технічне завдання, визначено потік роботи та проаналізовано механіки взаємодії з ворожими персонажами. Розроблено модель програмного комплексу бойової системи та здібностей персонажа, системи супротивників та взаємодії з ними, а також системи отримання нагород і битв з босами;

- наведено характеристики системи для оптимального запуску проєкту із найкращим досвідом гравців. Ретельно описано порядок взаємодії гравця з супротивниками та системою битв. Проведено тестування системи, виявлені й усунені помилки з метою поліпшення роботи відеогри.

В результаті проведеної роботи було розроблено елементи системи відеогри, а саме бойову систему й ігровий інтелект супротивників, які відповідають вимогам та потребам ігрового додатку. Також було забезпечено високу якість та безперебійну роботу гри шляхом ідентифікації та усунення помилок.

Отримана в результаті відеогра готова і доступна для широкої аудиторії та розповсюджується через платформу цифрової дистрибуції «Steam», де всі охочі мають можливість завантажити остаточну версію гри, придбавши її або спробувати демонстраційну версію.

В майбутньому планується розширення комп'ютерної гри додатковим завантажувальним вмістом, новими механіками, що зумовлено можливостями обраних для реалізації технологій.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		69

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Harrison F. Learning C# by Developing Games with Unity / Ferrone Harrison. – Birmingham: Packt Publishing, 2022. – 460 p. – (7th ed.). – (978-1837636877).
2. Felicia P. Unity From Zero to Proficiency (Foundations) / Patrick Felicia., 2019. – 234 с. – (Third Edition). – (978-1795806633).
3. Felicia P. Unity from Zero to Proficiency (Beginner) / Patrick Felicia., 2019. – 500 с. – (978-1091872028).
4. Nandy A. Neural Networks in Unity / A. Nandy, M. Biswas. – Kolkata, West Bengal: apress, 2018. – 166 с. – (978-1-4842-3672-7).
5. Шпигорь І. Боти для комп'ютерних ігор / Ілля Шпигорь., 2021. – 318 с. – (Leanpub).
6. Martin R. Clean Code / Robert Martin. – Boston: Pearson Education Inc., 2009. – 462 с. – (2). – (978-0-13-235088-4).
7. UNITY ПОГЛИБЛЕНИЙ [Електронний ресурс] // ITVDN. – 2023. – Режим доступу до ресурсу: <https://itvdn.com/ua/video/unity-advanced>.
8. Unity and C# basics [Електронний ресурс] // class central. – 2023. – Режим доступу до ресурсу: <https://www.classcentral.com/course/unity-and-c-basics-182599>.
9. Beginner Unity Bundle [Електронний ресурс] // GameDev.tv. – 2023. – Режим доступу до ресурсу: https://www.gamedev.tv/p/beginner-unity-bundle/?coupon_code=WOWZER.
10. Unity Game Developer [Електронний ресурс] // Robot Dreams. – 2023. – Режим доступу до ресурсу: <https://robotdreams.cc/uk/course/unity-game-developer?gad=1>.
11. Team Cherry [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://www.teamcherry.com.au/>.
12. The Game Kitchen [Електронний ресурс] // The Game Kitchen. – 2023. – Режим доступу до ресурсу: <https://thegamekitchen.com/>.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				70
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Team 17 [Электронный ресурс] // Team 17. – 2023. – Режим доступа до ресурсу: <https://www.team17.com/>.
14. Moon Studios [Электронный ресурс] // Moon Studios. – 2023. – Режим доступа до ресурсу: <https://www.orithegame.com/moon-studios/>.
15. From Software [Электронный ресурс] // From Software. – 2023. – Режим доступа до ресурсу: <https://www.fromsoftware.jp/jp/>.
16. Steam DB [Электронный ресурс] // Steam DB – 2023. – Режим доступа до ресурсу: <https://steamdb.info/app/>.
17. DOT NET FOR ALL [Электронный ресурс] // DOT NET FOR ALL. – 2023. – Режим доступа до ресурсу: <https://www.dotnetforall.com/>.
18. Learn Microsoft [Электронный ресурс] // Microsoft. – 2023. – Режим доступа до ресурсу: <https://learn.microsoft.com/en-us/dotnet/api/system.action?view=net-7.0>.
19. Dot Net Pattern [Электронный ресурс] // DotNetPattern. – 2023. – Режим доступа до ресурсу: <https://dotnetpattern.com/csharp-action-delegate>.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		71

ДОДАТКИ

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				72
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток А. Відгуки на кваліфікаційну роботу



КОМП'ЮТЕРНА АКАДЕМІЯ IT STEP

вул. Перемоги, буд. 29, м. Житомир, Україна, 10001
тел.: +38 (067) 303-05-33; +38(093) 087-30-33
e-mail: HYPERLINK "mailto:itstep.zhitomir@gmail.com" itstep.zhitomir@gmail.com

Шановний Назарію Леут,

Після детального огляду вашої роботи, ми хотіли б висловити наші враження та думки про вашу професійну діяльність. Наша команда визнає ваші навички в галузі розробки Unity та ваш внесок у систему битв та тестування.

Ваші досягнення у створенні системи битв вражають своєю якістю та функціональністю. Ви продемонстрували глибоке розуміння механік битв та зуміли ефективно реалізувати їх у продукті. Ваша робота свідчить про вашу здатність створювати захоплюючі та динамічні ігрові елементи.

Крім того, ми високо оцінюємо вашу роль у тестуванні. Ваші знання та досвід допомогли покращити якість продукту та забезпечити його стабільність. Ваші зусилля у галузі QA внесли вагомий внесок у проект.

Узагалі, ваша робота підтверджує ваші здібності та професійну підготовку. Ви проявилися як обізнаний розробник з глибоким розумінням геймдевелопменту. На основі вашої праці, ми можемо охарактеризувати ваш рівень як Junior Game Developer та Junior QA.

Ви володієте потенціалом стати цінним активом у галузі розробки ігор, і ми бажаємо вам успіху в подальшій кар'єрі. Ваш внесок у цей проект був високо оцінений.

Рекомендую оцінити роботу на високому рівні, у літерованій міжнародній системі, як 90A+.

З повагою,
Гармаш Олександр
Представник компанії ITSTEP Academy



		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Шановний Назарій Леут,

Після детального огляду вашої роботи, я хотів би висловити свої враження та думки про вашу професійну діяльність. Ваші навички у розробці системи битв та тестуванні демонструють ваше зацікавлення та знання у галузі геймдевелопменту.

Ваш внесок у розробку системи битв в проекті вражає своєю працездатністю та вмінням вирішувати завдання. Ви проявили знання механік гри та здатність створювати захоплюючі ігрові взаємодії. Ваша робота свідчить про ваш потенціал для подальшого росту у цій області.

Я також високо оцінюю вашу роботу в сфері тестування. Ви добре знаєте процес тестування програмного забезпечення та вмієте знаходити та документувати помилки. Ваше уважне та систематичне підхід до тестування сприяє покращенню якості продукту.

На основі представленої роботи, я можу охарактеризувати ваш рівень як Junior Game Developer та Junior QA (Quality Assurance) Specialist. Ви проявляєте потенціал для подальшого росту та розвитку у галузі геймдевелопменту.

Бажаю вам успіху в подальшій кар'єрі та надалі розширювати свої навички та знання у цій галузі.

Рекомендую оцінити роботу на високому рівні.

З повагою,

Представник компанії INP-SOFTWARE



Рудницький С. Г.

		Н.В. Леут			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		74