

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА 2D ГРИ ЖАНРУ BULLET HEAVEN «SOUL HARVEST»
ЗАСОБАМИ UNITY**

**DEVELOPMENT OF A 2D GAME «SOUL HARVEST» IN THE BULLET
HEAVEN GENRE USING UNITY**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Вакуленко Денис Мирославович

Керівник:
Суринович Олена Миколаївна

Кваліфікаційну роботу
допущено до захисту
«__» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Вакуленку Денису Мирославовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка 2D гри жанру Bullet Heaven «Soul Harvest» засобами Unity»

Керівник роботи: доцент Суринович О. М.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026 р.

3. Вихідні дані до роботи: предметна область розробки 2D ігор жанру Bullet Heaven, технології розробки ігор із використанням мови програмування C# та рушія Unity, UML-моделювання, методи організації ігрового процесу, створення користувацького інтерфейсу та оптимізації продуктивності ігрових застосунків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз особливостей ігор жанру Bullet Heaven та існуючих аналогів, постановка завдання, визначення вимог до програмного забезпечення, проєктування архітектури гри та UML-моделювання, обґрунтування вибору технологій і засобів розробки, практична реалізація 2D гри «Soul Harvest», розробка основних ігрових механік, організація ігрового процесу та баланс складності, тестування, налагодження й оптимізація програмного забезпечення.

5. Перелік графічного матеріалу: 6 рисунків, 6 лістингів,

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	<i>Суринович О. М.</i>		
Специфікація вимог до розробленої системи	<i>Суринович О. М.</i>		
Розробка об'єкта проектування	<i>Суринович О. М.</i>		
Нормоконтроль	<i>Суринович О. М.</i>		
Гарант ОП	<i>Ліщина Н. М.</i>		
Показник запозичень тексту	<u> </u> %		
Академічна доброчесність	<i>Суринович О. М.</i>		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Денис ВАКУЛЕНКО

Керівник кваліфікаційної роботи _____

Олена СУРИНОВИЧ

АНОТАЦІЯ

Вакуленко Д. М. Розробка 2D гри жанру Bullet Heaven «Soul Harvest» засобами Unity. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі проведено аналіз предметної області, пов'язаної з розробкою комп'ютерних ігор жанру Bullet Heaven, досліджено особливості сучасних 2D survival-ігор, проаналізовано існуючі аналоги та обґрунтовано доцільність створення власного програмного продукту. У другому розділі визначено функціональні та нефункціональні вимоги до гри, виконано проєктування архітектури програмного забезпечення за допомогою UML-діаграм, а також обґрунтовано вибір рушія Unity, мови програмування C# та інших інструментальних засобів розробки. У третьому розділі описано процес практичної реалізації 2D гри жанру Bullet Heaven «Soul Harvest», розглянуто створення основних ігрових механік, організацію ігрового процесу, баланс складності, реалізацію користувацького інтерфейсу, а також тестування, налагодження й оптимізацію програмного забезпечення.

У висновках підсумовано результати виконаної роботи, підтверджено досягнення поставленої мети та повне виконання завдань кваліфікаційної роботи.

Ключові слова: 2D гра, Bullet Heaven, Unity, C#, розробка ігор, ігровий рушій, ігрові механіки, система прогресії, UML-моделювання, користувацький інтерфейс, Object Pooling.

ABSTRACT

Vakulenko D. M. Development of a 2D game “Soul Harvest” in the Bullet Heaven genre using Unity. Manuscript. Bachelor's qualification work of the OP “Software Engineering” specialty “Software Engineering”. Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three sections, conclusions, and a list of sources used.

The first section analyzes the subject area related to the development of computer games of the Bullet Heaven genre, examines the features of modern 2D survival games, analyzes existing analogues, and justifies the feasibility of creating your own software product. The second section identifies functional and non-functional requirements for the game, performs software architecture design using UML diagrams, and justifies the choice of the Unity engine, the C# programming language, and other development tools. The third section describes the process of practical implementation of the 2D game of the Bullet Heaven genre “Soul Harvest”, considers the creation of basic game mechanics, organization of the gameplay, balance of complexity, implementation of the user interface, as well as testing, debugging and optimization of the software.

The conclusions summarize the results of the work performed, confirm the achievement of the set goal and the full completion of the tasks of the qualification work.

Keywords: 2D game, Bullet Heaven, Unity, C#, game development, game engine, game mechanics, progression system, UML modeling, user interface, Object Pooling.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ	9
1.1 Аналіз сучасного стану проблеми	9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	12
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ...	13
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення	13
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання ...	18
РОЗДІЛ 3 РОЗРОБКА 2D ГРИ ЖАНРУ BULLET HEAVEN «SOUL HARVEST».....	23
3.1 Практична реалізація об'єкта проєктування.....	23
3.2 Аналіз ігрових механік жанру Bullet Heaven та їх адаптація у грі «Soul Harvest».....	32
3.3 Організація ігрового процесу та баланс складності	35
3.4 Тестування та налагодження 2D гри жанру Bullet Heaven «Soul Harvest»	38
ВИСНОВКИ	41
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	44

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку індустрії відеоігор особливої популярності набувають ігри жанру Bullet Heaven, основою яких є динамічний ігровий процес, велика кількість супротивників, система розвитку персонажа та використання різноманітних здібностей. Завдяки простому керуванню, високій динаміці та можливості постійного вдосконалення персонажа такі ігри привертають увагу широкої аудиторії користувачів. Розвиток комп'ютерної графіки, ігрових рушіїв та мов програмування створює сприятливі умови для розробки власних ігрових проєктів різного рівня складності [1].

Сьогодні ринок комп'ютерних ігор активно розширюється, а незалежні розробники отримують можливість створювати повноцінні ігрові продукти за допомогою сучасних засобів розробки. Одним із найбільш популярних інструментів для створення 2D та 3D ігор є ігровий рушій Unity, який забезпечує широкий набір функцій для роботи з графікою, фізикою, анімацією та взаємодією об'єктів. Використання мови програмування C# дозволяє реалізовувати складну ігрову логіку, оптимізувати продуктивність гри та забезпечувати зручність подальшого масштабування проєкту .

Особливістю жанру Bullet Heaven є необхідність обробки великої кількості ігрових об'єктів одночасно: ворогів, снарядів, ефектів та елементів інтерфейсу. Це створює потребу в ефективній організації архітектури програмного забезпечення, оптимізації використання ресурсів та реалізації механізмів керування ігровими процесами. Невдало реалізована система генерації ворогів або обробки зіткнень може призводити до зниження продуктивності, затримок у роботі гри та погіршення користувацького досвіду.

Окремої уваги потребує створення систем розвитку персонажа, балансу складності та мотивації гравця до подальшого проходження. Для сучасних ігор важливими є не лише технічна реалізація, а й забезпечення цікавого ігрового процесу, зручного інтерфейсу користувача та системи прогресії, яка підтримує

інтерес гравця протягом тривалого часу. Саме тому розробка гри жанру Bullet Heaven потребує комплексного підходу до проектування ігрових механік та програмної реалізації.

Таким чином, розробка 2D гри жанру Bullet Heaven «Soul Harvest» засобами Unity з використанням мови програмування C# є актуальною темою, оскільки поєднує сучасні підходи до створення ігор, реалізацію інтерактивних механік та вирішення задач оптимізації продуктивності й організації ігрової логіки.

Метою кваліфікаційної роботи є розробка 2D гри жанру Bullet Heaven «Soul Harvest» засобами Unity з використанням мови програмування C#.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати особливості та принципи побудови ігор жанру Bullet Heaven;
- дослідити можливості ігрового рушія Unity та мови програмування C#;
- визначити функціональні та нефункціональні вимоги до гри;
- спроектувати архітектуру програмного забезпечення гри;
- реалізувати основні ігрові механіки та розробити систему керування персонажем, ворогами та здібностями;
- реалізувати користувацький інтерфейс і систему прогресії;
- провести тестування та оптимізацію програмного продукту.

Об'єктом дослідження є процес розробки комп'ютерних 2D ігор жанру Bullet Heaven.

Предметом дослідження є методи та засоби створення 2D гри засобами Unity з використанням мови програмування C#.

Практичне значення роботи полягає у створенні ігрового програмного продукту «Soul Harvest», який демонструє можливості використання Unity та мови C# для реалізації сучасних 2D ігор жанру Bullet Heaven, а також – як основа для подальшого розвитку та вдосконалення ігрового проекту.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Сучасна індустрія комп'ютерних ігор є однією з найбільш динамічних сфер розвитку інформаційних технологій. Завдяки розвитку програмного забезпечення, ігрових рушіїв та апаратного забезпечення значно зросли можливості створення інтерактивних ігрових продуктів різних жанрів та рівнів складності. Особливої популярності останніми роками набули 2D ігри, які поєднують відносно невисокі вимоги до ресурсів комп'ютера, простоту розробки та широкий спектр ігрових механік. Одним із жанрів, що активно розвивається, є Bullet Heaven.

Жанр Bullet Heaven характеризується великою кількістю ворогів на ігровій сцені, автоматизованою або частково автоматизованою системою атак персонажа, динамічним ігровим процесом та поступовим розвитком здібностей гравця. Основною метою таких ігор є виживання персонажа протягом певного часу, знищення хвиль супротивників та покращення характеристик героя. Популярність жанру значно зросла після виходу гри Vampire Survivors, яка продемонструвала високий інтерес користувачів до простих за графікою, але динамічних та захопливих ігор [2].

Однією з основних проблем при розробці ігор жанру Bullet Heaven є необхідність обробки великої кількості одночасних об'єктів. Під час гри на сцені можуть перебувати сотні або навіть тисячі ворогів, снарядів, ефектів та інших елементів. Це створює значне навантаження на процесор і графічну підсистему, що потребує оптимізації алгоритмів руху, зіткнень та генерації об'єктів. Неефективна реалізація таких механік може призвести до падіння продуктивності, затримок у роботі гри та погіршення взаємодії користувача з програмним продуктом.

Ще однією важливою задачею є забезпечення балансу ігрового процесу. Для ігор жанру Bullet Heaven характерна система прогресії персонажа, у межах якої гравець поступово отримує нові здібності, покращує характеристики та відкриває нові можливості. Неправильний баланс між складністю ворогів і можливостями персонажа може зробити гру занадто легкою або надмірно складною, що негативно впливає на зацікавленість користувачів.

Для створення сучасних 2D ігор широко використовуються спеціалізовані ігрові рушії. Одним із найпоширеніших серед них є Unity. Даний рушій надає засоби для роботи з 2D-графікою, анімацією, фізикою, аудіо та системою керування ресурсами. Важливою перевагою Unity є підтримка мови програмування C#, яка дозволяє реалізовувати складну ігрову логіку, створювати гнучку архітектуру програмного забезпечення та забезпечувати масштабованість проєкту [3].

На сьогодні існує значна кількість ігор жанру Bullet Heaven, які можна розглядати як аналоги для дослідження. Однією з найвідоміших є гра Vampire Survivors, що вирізняється мінімалістичною графікою, великою кількістю ворогів та системою випадкового покращення здібностей персонажа (рис. 1.1). Перевагами цієї гри є динамічний ігровий процес, проста система керування та високий рівень повторюваності проходження. До недоліків можна віднести обмежену різноманітність взаємодії з оточенням та спрощену систему бойових механік [4].



Рисунок 1.1 – Vampire Survivors [5]

Іншим прикладом є гра Brotato, у якій реалізовано хвильову систему виживання та можливість одночасного використання декількох видів зброї. Перевагою гри є велика кількість варіантів розвитку персонажа та швидкий темп гри. Водночас недоліком можна вважати перевантаження екрану великою кількістю візуальних ефектів, що інколи ускладнює сприйняття ігрового процесу (рис. 1.2) [6].



Рисунок 1.2 – Brotato [7]

Також до популярних представників жанру належить 20 Minutes Till Dawn, яка поєднує елементи Bullet Heaven та roguelike. Гра вирізняється атмосферним стилем оформлення, системою ручного прицілювання та великою кількістю комбінацій здібностей. Недоліком гри є вищий рівень складності для нових користувачів та підвищені вимоги до навичок керування персонажем [8].

Аналіз існуючих аналогів показує, що сучасні ігри жанру Bullet Heaven мають значний потенціал розвитку, проте багато з них зосереджуються або на простоті реалізації, або на вузькому наборі ігрових механік. Це створює передумови для розробки нового програмного продукту, який поєднуватиме динамічний ігровий процес, оптимізовану систему генерації ворогів, систему розвитку персонажа та зручний користувацький інтерфейс.

У межах кваліфікаційної роботи планується розробка 2D гри жанру Bullet Heaven «Soul Harvest» засобами Unity з використанням мови програмування C#. Основна увага приділятиметься реалізації ефективної архітектури гри,

оптимізації продуктивності, створенню системи розвитку персонажа та забезпеченню комфортної взаємодії користувача з ігровим середовищем. Це дозволить створити сучасний ігровий продукт, який відповідатиме актуальним тенденціям розвитку індустрії комп'ютерних ігор.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

У межах кваліфікаційної роботи передбачається розробка комп'ютерної гри жанру Bullet Heaven під назвою «Soul Harvest». Гра створюється з використанням ігрового рушія Unity та мови програмування C# і орієнтована на реалізацію динамічного ігрового процесу, системи розвитку персонажа, взаємодії з ворогами та використання різноманітних здібностей.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати особливості та принципи побудови ігор жанру Bullet Heaven;
- дослідити можливості ігрового рушія Unity та мови програмування C#;
- визначити функціональні та нефункціональні вимоги до гри;
- спроектувати архітектуру програмного забезпечення гри;
- реалізувати основні ігрові механіки та розробити систему керування персонажем, ворогами та здібностями;
- реалізувати користувацький інтерфейс і систему прогресії;
- провести тестування та оптимізацію програмного продукту.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Розробка сучасної комп'ютерної гри потребує комплексного підходу до аналізу вимог, проєктування архітектури програмного забезпечення та моделювання взаємодії окремих компонентів системи. Для створення 2D гри жанру Bullet Heaven «Soul Harvest» необхідно врахувати особливості реалізації динамічного ігрового процесу, забезпечення стабільної продуктивності та створення зручного користувацького інтерфейсу.

Одним із ключових етапів створення програмного продукту є вибір моделі проєктування системи. Для розробки гри «Soul Harvest» доцільно використовувати об'єктно-орієнтований підхід, оскільки він забезпечує модульність, масштабованість та можливість повторного використання програмних компонентів. Використання об'єктно-орієнтованого програмування у поєднанні з можливостями рушія Unity дозволяє реалізувати незалежні компоненти для керування персонажем, ворогами, системою атак, інтерфейсом користувача та іншими елементами гри.

Під час проєктування програмного забезпечення було використано метод аналізу функціональних вимог, що дозволив визначити основні можливості гри та взаємодію користувача із системою [9]. Основним користувачем програмного продукту є гравець, який взаємодіє з ігровим середовищем за допомогою клавіатури та миші. Система повинна забезпечувати безперервний ігровий процес, коректне відображення об'єктів, обробку зіткнень та збереження ігрових параметрів.

Для формалізації структури програмного забезпечення доцільно використовувати UML-моделювання. У межах проєктування гри можуть бути створені такі UML-діаграми:

- діаграма варіантів використання (Use Case Diagram), що описує взаємодію гравця з основними функціями гри;
- діаграма класів (Class Diagram), яка відображає структуру основних програмних компонентів;
- діаграма діяльності (Activity Diagram), що демонструє послідовність виконання ігрових процесів.

Для аналізу взаємодії користувача з основними функціями гри була створена UML-діаграма варіантів використання (Use Case Diagram) (рис. 2.1). Дана діаграма дозволяє відобразити основні сценарії взаємодії гравця із системою, зокрема запуск гри, керування персонажем, взаємодію з меню налаштувань, отримання покращень персонажа та завершення ігрової сесії. Використання діаграми варіантів використання дозволяє краще визначити функціональні вимоги до програмного продукту та структурувати основні можливості гри.

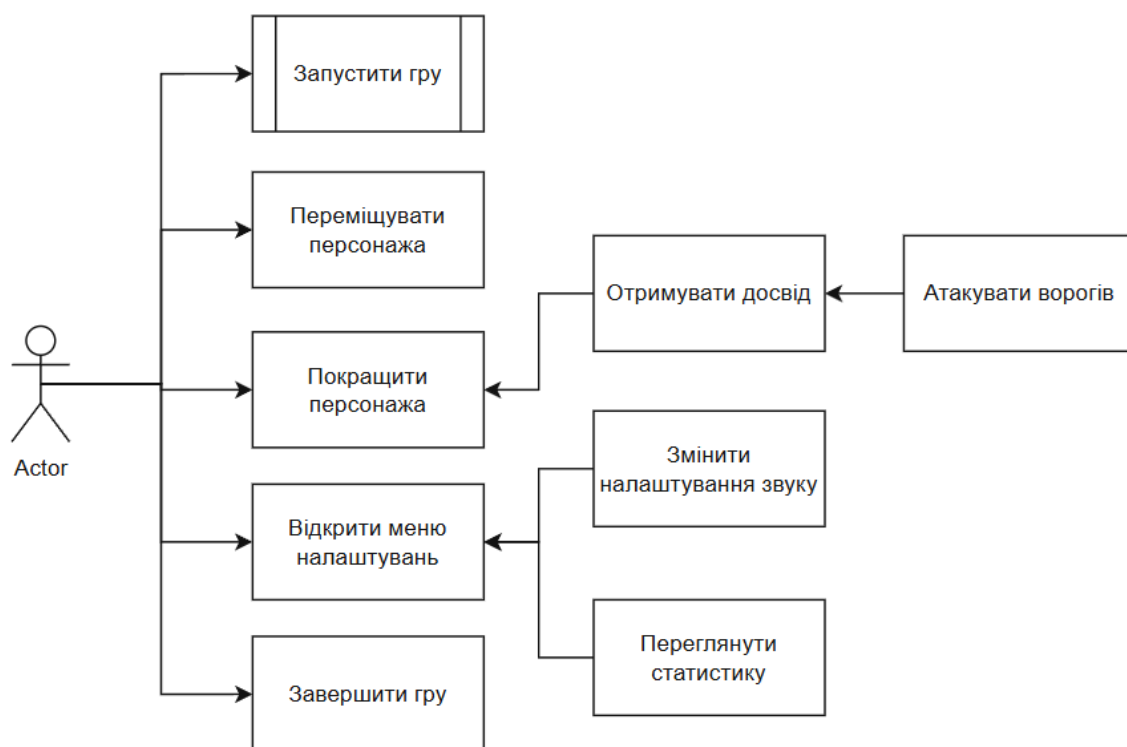


Рисунок 2.1 – UML-діаграма варіантів використання

Для відображення внутрішньої структури програмного забезпечення була розроблена UML-діаграма класів (Class Diagram) (рис. 2.2). Дана діаграма демонструє основні класи системи, їх атрибути, методи та взаємозв'язки між компонентами гри. На діаграмі представлені класи персонажа, ворогів, системи атак, менеджера гри, системи генерації ворогів, інтерфейсу користувача та інших елементів. Використання діаграми класів дозволяє забезпечити модульність програмного забезпечення та спрощує процес подальшого розширення функціоналу гри.

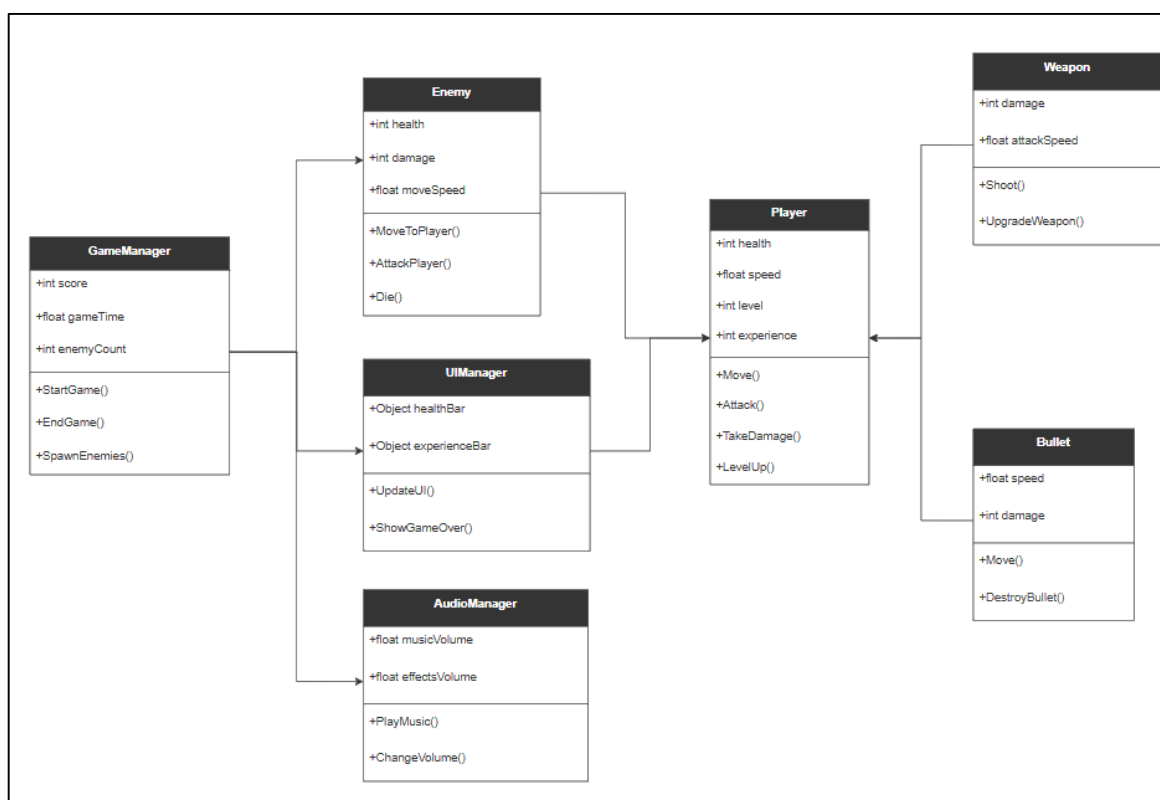


Рисунок 2.2 – UML-діаграма класів

Для моделювання послідовності виконання основних ігрових процесів була створена UML-діаграма діяльності (Activity Diagram) (рис. 2.3). Вона відображає алгоритм роботи гри від моменту запуску до завершення ігрової сесії. На діаграмі представлено процес запуску гри, вибір дій користувача, генерацію ворогів, отримання досвіду, підвищення рівня персонажа та обробку завершення гри у випадку втрати всіх очок здоров'я. Діаграма діяльності дозволяє наочно

продемонструвати логіку функціонування програмного продукту та взаємозв'язок між окремими етапами ігрового процесу.

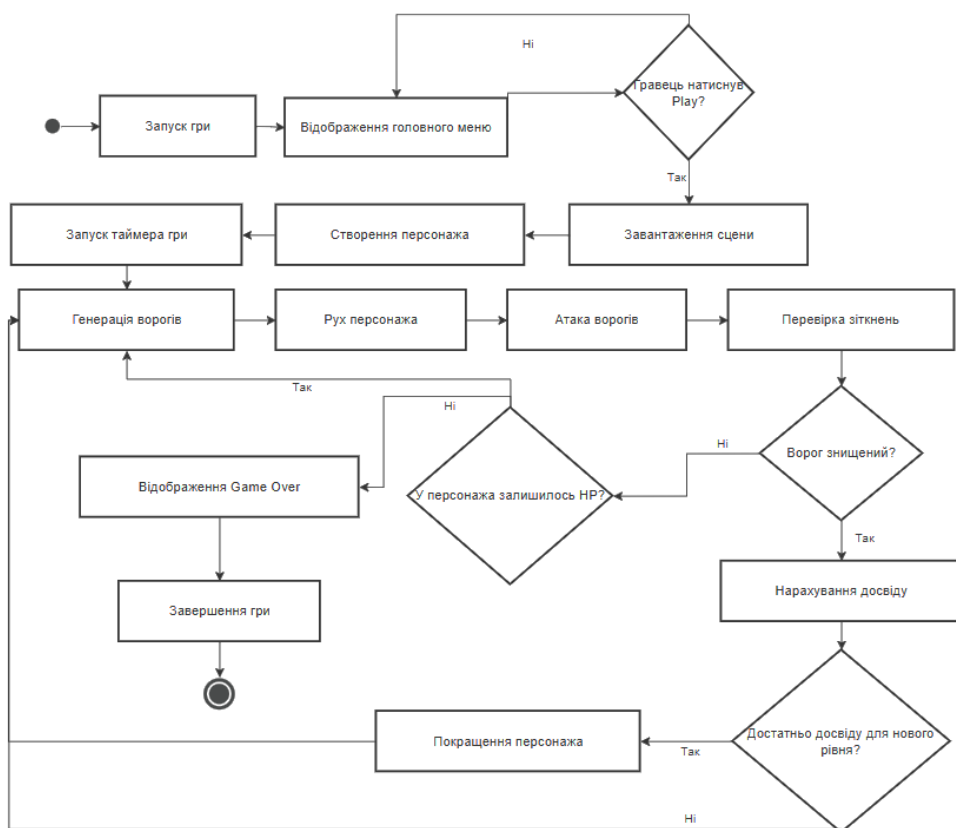


Рисунок 2.3 – UML-діаграма діяльності

Основною метою програмного продукту є створення динамічної 2D гри жанру Bullet Heaven із системою виживання, розвитку персонажа та поступового ускладнення ігрового процесу. Для досягнення цієї мети система повинна забезпечувати реалізацію комплексу функціональних можливостей.

До основних функцій гри «Soul Harvest» належать:

- запуск та завершення ігрової сесії;
- керування персонажем;
- генерація хвиль ворогів;
- автоматична або керована система атак;
- система зіткнень між персонажем, ворогами та снарядами;
- система накопичення досвіду та підвищення рівня персонажа;

- покращення здібностей та характеристик героя;
- відображення статистики гри та характеристик персонажа;
- реалізація головного меню та меню налаштувань;
- відтворення звукових ефектів та музичного супроводу;
- збереження основних ігрових параметрів.

Функціональні вимоги до програмного забезпечення визначають поведінку системи та набір можливостей, які повинні бути реалізовані у грі. Система повинна забезпечувати плавне пересування персонажа по ігровій карті, автоматичне створення ворогів відповідно до рівня складності та коректну взаємодію між усіма ігровими об'єктами. Гра повинна підтримувати систему розвитку персонажа, у межах якої користувач отримує нові здібності або покращення після підвищення рівня.

Важливим елементом системи є механізм керування ресурсами та оптимізації продуктивності. Оскільки у жанрі Bullet Heaven на екрані одночасно може перебувати велика кількість ворогів та снарядів, програмне забезпечення повинно забезпечувати ефективне використання оперативної пам'яті та процесорних ресурсів. Для цього доцільно застосовувати механізми Object Pooling, оптимізацію обробки фізики та мінімізацію кількості одночасних обчислень.

До нефункціональних вимог системи належать:

- забезпечення стабільної роботи гри без критичних помилок;
- підтримка комфортної частоти кадрів під час великої кількості об'єктів;
- зручність користувацького інтерфейсу;
- простота керування персонажем;
- можливість подальшого розширення функціоналу гри;
- сумісність із сучасними операційними системами;
- швидке завантаження сцен та ігрових ресурсів.

Під час проєктування інтерфейсу користувача особлива увага приділяється UX/UI-дизайну. Інтерфейс гри повинен бути інтуїтивно зрозумілим та не перевантажувати користувача зайвою інформацією. Основними елементами інтерфейсу є головне меню, кнопки початку гри та налаштувань, індикатор здоров'я персонажа, шкала досвіду, відображення поточного рівня, кількості знищених ворогів та часу виживання.

У межах UX/UI-дизайну також передбачається візуальне оформлення гри, воно повинне відповідати стилістиці жанру та забезпечувати комфортне сприйняття великої кількості ігрових об'єктів на екрані [10].

Інформаційні потоки в системі організовані таким чином, щоб забезпечити безперервну взаємодію між ігровими компонентами. Вхідними даними є дії користувача, зокрема натискання клавіш та взаємодія з елементами інтерфейсу. На основі цих дій система обробляє логіку гри, змінює стан персонажа, створює нові об'єкти та оновлює інформацію на екрані. Результатом роботи системи є відображення актуального стану гри, статистики та анімаційних ефектів.

Таким чином, визначення вимог та проєктування програмного забезпечення гри «Soul Harvest» дозволяє сформуванню основи для реалізації ефективного, масштабованого та зручного у використанні ігрового продукту. Використання сучасних методів моделювання та можливостей рушія Unity забезпечує створення програмної системи, яка відповідатиме функціональним та технічним вимогам сучасних 2D ігор жанру Bullet Heaven.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Розробка 2D гри жанру Bullet Heaven «Soul Harvest» потребує використання сучасних інструментів та технологій, які забезпечують ефективне створення ігрового програмного забезпечення, реалізацію складної ігрової логіки та оптимізацію продуктивності. Для досягнення поставленої мети було

проведено аналіз існуючих засобів розробки ігор, методів організації програмної архітектури та алгоритмів обробки ігрових процесів.

Одним із найважливіших етапів є вибір ігрового рушія. Серед найбільш популярних сучасних рушіїв можна виділити Unity, Unreal Engine та Godot. Unreal Engine забезпечує широкі можливості для створення високоякісної 3D-графіки та складних візуальних ефектів, проте потребує значних апаратних ресурсів та є більш складним для розробки 2D ігор. Godot є безкоштовним та легким у використанні рушієм, однак має меншу кількість готових інструментів і навчальних матеріалів у порівнянні з Unity [11].

Для реалізації гри «Soul Harvest» було обрано ігровий рушій Unity. Основними перевагами Unity є підтримка 2D та 3D розробки, велика кількість вбудованих інструментів, зручна система компонентів, підтримка фізики, анімації та UI-елементів. Крім того, Unity має велику спільноту розробників, значну кількість документації та готових програмних рішень, що спрощує процес створення гри.

У якості основної мови програмування використовується C#. Дана мова добре інтегрується з Unity та забезпечує можливість реалізації об'єктно-орієнтованого підходу до розробки програмного забезпечення. Використання C# дозволяє створювати модульну архітектуру гри, спрощує підтримку коду та дає можливість реалізовувати складні алгоритми взаємодії між компонентами системи.

Для створення програмного коду було обрано середовище розробки Visual Studio. Це середовище забезпечує підтримку роботи з Unity, автоматичне доповнення коду, засоби налагодження та аналізу помилок. Використання Visual Studio дозволяє прискорити процес розробки та підвищити якість програмного забезпечення.

Для створення візуальної складової гри були використані готові графічні ресурси з набору Undead Survivor Assets Pack, доступного в магазині ресурсів Unity Asset Store. Даний набір містить готові 2D-спрайти персонажів, ворогів, анімації, елементи оточення та візуальні ефекти, що дозволило значно

пришвидшити процес розробки гри та зосередити основну увагу на реалізації ігрової логіки й механік. Використання готових текстур також забезпечує єдиний художній стиль гри та покращує загальне візуальне сприйняття програмного продукту. Для роботи зі звуковими ефектами та аудіосупроводом можуть використовуватися вбудовані засоби Unity або спеціалізовані аудіоредактори, зокрема Audacity.

При розробці програмного забезпечення гри використовується об'єктно-орієнтований метод проєктування. Основною перевагою цього підходу є розділення функціоналу на окремі класи та компоненти. Наприклад, окремі класи відповідають за керування персонажем, поведінку ворогів, систему атак, відображення інтерфейсу та керування ресурсами гри. Такий підхід забезпечує зручність підтримки та можливість подальшого розширення функціоналу [12].

Одним із ключових методів реалізації гри є компонентний підхід, який використовується в Unity. Кожен ігровий об'єкт складається з набору компонентів, що відповідають за певні функції. Наприклад, персонаж може містити компоненти переміщення, анімації, системи здоров'я та обробки зіткнень. Це дозволяє гнучко налаштовувати поведінку об'єктів та повторно використовувати створені компоненти.

Для реалізації системи руху персонажа використовується алгоритм обробки введення користувача. Система зчитує натискання клавіш клавіатури та змінює координати персонажа відповідно до напрямку руху. Оновлення позиції персонажа виконується у кожному кадрі гри, що забезпечує плавність переміщення.

Алгоритм генерації ворогів базується на системі періодичного створення об'єктів у випадкових точках навколо гравця. Частота появи ворогів та їх характеристики змінюються залежно від часу гри або рівня персонажа. Такий підхід дозволяє поступово підвищувати складність гри та підтримувати інтерес користувача.

Для реалізації взаємодії між ігровими об'єктами використовується система колізій Unity. Під час зіткнення об'єктів система перевіряє їхні типи та виконує

відповідні дії, наприклад зменшення рівня здоров'я ворога або персонажа, знищення снаряда чи запуск анімації ефекту.

Важливим елементом є система розвитку персонажа. Алгоритм її роботи базується на накопиченні очок досвіду після знищення ворогів. Після досягнення певного значення досвіду рівень персонажа підвищується, а користувач отримує можливість обрати нове покращення або здібність. Це забезпечує різноманітність проходження та підвищує зацікавленість гравця.

Оскільки жанр Bullet Heaven передбачає велику кількість одночасних об'єктів, важливою задачею є оптимізація продуктивності. Для цього використовується метод Object Pooling. Суть даного методу полягає у повторному використанні вже створених об'єктів замість постійного створення та видалення нових. Це дозволяє значно зменшити навантаження на оперативну пам'ять та процесор, а також підвищити стабільність роботи гри [13].

Для проектування структури програмного забезпечення використовуються UML-діаграми. Діаграма класів дозволяє відобразити структуру основних компонентів гри та взаємозв'язки між ними. Діаграма пакетів використовується для логічного групування компонентів системи, а діаграма компонентів демонструє взаємодію окремих модулів програмного забезпечення. Також може бути використана діаграма розміщення для опису середовища виконання програмного продукту.

У межах розробки гри «Soul Harvest» також застосовуються методи UX/UI-проектування. Основною метою є створення інтуїтивно зрозумілого інтерфейсу користувача, який забезпечить швидкий доступ до основних функцій гри та комфортне сприйняття інформації. Для цього використовуються мінімалістичний дизайн, зручне розташування елементів інтерфейсу та система візуального виділення важливих показників.

Також під час розробки гри використовуються методи модульного програмування та повторного використання компонентів. Це дозволяє розділити програмний код на окремі незалежні модулі, кожен з яких відповідає за конкретну частину функціоналу гри. Наприклад, окремо реалізуються системи

керування персонажем, генерації ворогів, обробки пошкоджень, інтерфейсу користувача та звукового супроводу. Такий підхід спрощує процес тестування, налагодження та подальшого вдосконалення програмного продукту.

Окрему увагу приділено алгоритмам організації ігрового процесу та підтримання балансу складності. Для цього використовується система поступового збільшення кількості ворогів і покращення їх характеристик залежно від часу гри. Подібний підхід дозволяє підтримувати динамічність ігрового процесу та забезпечує поступове зростання складності без різких змін навантаження на користувача. Крім того, реалізовані механізми прогресії персонажа дозволяють компенсувати підвищення складності шляхом покращення характеристик героя та відкриття нових можливостей під час проходження гри.

Таким чином, вибір засобів, методів та алгоритмів розробки забезпечує створення ефективної та масштабованої програмної системи. Використання рушія Unity, мови програмування C#, об'єктно-орієнтованого підходу та сучасних методів оптимізації дозволяє реалізувати 2D гру жанру Bullet Heaven «Soul Harvest» із високою продуктивністю, стабільною роботою та зручним користувацьким інтерфейсом.

РОЗДІЛ 3

РОЗРОБКА 2D ГРИ ЖАНРУ BULLET HEAVEN «SOUL HARVEST»

3.1 Практична реалізація об'єкта проєктування

Практична реалізація 2D гри жанру Bullet Heaven «Soul Harvest» виконувалась із використанням ігрового рушія Unity та мови програмування C#. Основною метою етапу реалізації було створення повноцінного ігрового програмного продукту з динамічним ігровим процесом, системою виживання, генерацією ворогів, бойовою системою та механізмами розвитку персонажа. У процесі розробки особлива увага приділялася організації архітектури проєкту, оптимізації продуктивності та реалізації взаємодії між основними компонентами системи.

Для створення гри було використано середовище розробки Unity, яке забезпечує підтримку 2D-графіки, фізики, анімацій, системи UI та засобів керування ресурсами. У якості мови програмування використовувалась C#, оскільки вона є основною мовою програмування для Unity та забезпечує реалізацію об'єктно-орієнтованого підходу до розробки програмного забезпечення. Написання програмного коду виконувалось у середовищі Visual Studio, яке забезпечує підтримку автоматичного доповнення коду, засобів налагодження та аналізу помилок.

На початковому етапі реалізації було створено структуру проєкту та організовано основні папки для зберігання скриптів, спрайтів, анімацій, аудіофайлів та сцен гри. Для забезпечення зручності підтримки програмного коду структура проєкту була поділена на окремі модулі відповідно до функціонального призначення компонентів системи. Зокрема, були створені окремі каталоги для системи керування персонажем, системи ворогів, менеджерів гри, UI-компонентів та системи аудіо.

Для реалізації персонажа було створено окремий об'єкт Player, який містить компоненти Sprite Renderer, Rigidbody2D, Collider2D та набір власних скриптів для керування логікою персонажа. Основним завданням даного об'єкта

є обробка введення користувача, переміщення персонажа, взаємодія з ворогами та система отримання пошкоджень.

Система руху персонажа реалізована за допомогою класу `PlayerMovement`, який відповідає за зчитування введення з клавіатури та зміну позиції персонажа у просторі сцени. Для цього використовується компонент `Rigidbody2D` та метод `Input.GetAxisRaw()`, який дозволяє отримувати значення напрямку руху по горизонталі та вертикалі, приклад коду наведено у лістингу 3.1.

Лістинг 3.1 – Реалізація руху персонажа

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    public float moveSpeed = 5f;

    private Rigidbody2D rb;
    private Vector2 movement;

    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    private void Update()
    {
        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");
    }

    private void FixedUpdate()
    {
        rb.MovePosition(rb.position + movement * moveSpeed *
Time.fixedDeltaTime);
    }
}
```

кінець лістингу 3.1

У наведеному фрагменті коду здійснюється обробка натискань клавіш користувача та переміщення персонажа по ігровій сцені. Метод `FixedUpdate()` використовується для коректної роботи фізики, а застосування `Time.fixedDeltaTime` забезпечує плавність руху незалежно від частоти кадрів.

Однією з ключових особливостей гри жанру Bullet Heaven є автоматична система атак. Для реалізації даної механіки було створено окремий клас `WeaponController`, який відповідає за створення снарядів через певний інтервал часу. Система атак використовує Prefab-об'єкти куль та механізм `Instantiate` для створення нових екземплярів снарядів, приклад його реалізації наведено у лістингу 3.2.

Лістинг 3.2 – Реалізація автоматичної атаки

```
using UnityEngine;

public class WeaponController : MonoBehaviour
{
    public GameObject bulletPrefab;
    public Transform firePoint;
    public float fireRate = 1f;

    private float fireTimer;

    private void Update()
    {
        fireTimer += Time.deltaTime;

        if (fireTimer >= fireRate)
        {
            Shoot();
            fireTimer = 0f;
        }
    }

    void Shoot()
    {
        Instantiate(bulletPrefab, firePoint.position,
Quaternion.identity);
    }
}
```

кінець лістингу 3.2

У даному алгоритмі змінна `fireTimer` використовується для відстеження часу між атаками. Після досягнення встановленого значення виконується створення нового снаряда. Такий підхід дозволяє легко змінювати швидкість атаки персонажа та реалізовувати систему покращення зброї.

Для реалізації поведінки ворогів було створено клас EnemyAI, який забезпечує рух ворогів у напрямку персонажа. Алгоритм роботи базується на постійному визначенні позиції гравця та обчисленні напрямку руху ворога наведено у лістингу 3.3.

Лістинг 3.3 – Реалізація логіки ворогів

```
using UnityEngine;

public class EnemyAI : MonoBehaviour
{
    public float speed = 2f;

    private Transform player;

    private void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    private void Update()
    {
        Vector2 direction = (player.position -
transform.position).normalized;
        transform.position += (Vector3)direction * speed *
Time.deltaTime;
    }
}
```

кінець лістингу 3.3

У наведеному коді ворог постійно рухається в напрямку персонажа, що створює основну механіку виживання. Для забезпечення коректної роботи використовується тег Player, за допомогою якого здійснюється пошук об'єкта гравця на сцені.

Важливим елементом реалізації є система зіткнень. Для цього використовуються компоненти Collider2D та механізми фізики Unity. При зіткненні кулі з ворогом виконується зменшення рівня здоров'я ворога, а у випадку досягнення нульового значення об'єкт ворога знищується. Даний код реалізує перевірку зіткнення снаряда з ворогом та передачу пошкодження через

метод `TakeDamage()`. Після зіткнення снаряд автоматично видаляється зі сцени, приклад коду наведено у лістингу 3.4.

Лістинг 3.4 – Реалізація обробки зіткнення

```
using UnityEngine;

public class Bullet : MonoBehaviour
{
    public int damage = 1;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Enemy"))
        {
            collision.GetComponent<EnemyHealth>().TakeDamage(damage);
            Destroy(gameObject);
        }
    }
}
```

кінець лістингу 3.4

У процесі реалізації також було створено систему генерації ворогів, яка забезпечує періодичне створення супротивників у випадкових точках навколо гравця. Це дозволяє поступово збільшувати складність гри та підтримувати постійний ігровий процес.

Для реалізації системи генерації ворогів було створено окремий клас `EnemySpawner`, який відповідає за періодичне створення нових супротивників на сцені. Генерація ворогів здійснюється у випадкових точках навколо гравця, що дозволяє створити ефект постійної небезпеки та підтримувати динамічний ігровий процес. Частота створення ворогів може збільшуватись із часом, що забезпечує поступове ускладнення гри.

У даному фрагменті коду використовується метод `Random.insideUnitCircle`, який дозволяє отримати випадкову позицію навколо персонажа. Після визначення координат виконується створення нового об'єкта ворога наведено у лістингу 3.5.

Лістинг 3.5 – Реалізація генерація ворогів

```
using UnityEngine;

public class EnemySpawner : MonoBehaviour
{
    public GameObject enemyPrefab;
    public Transform player;

    public float spawnRadius = 10f;
    public float spawnRate = 2f;

    private float timer;

    private void Update()
    {
        timer += Time.deltaTime;

        if (timer >= spawnRate)
        {
            SpawnEnemy();
            timer = 0f;
        }
    }

    void SpawnEnemy()
    {
        Vector2 randomPosition = Random.insideUnitCircle.normalized *
spawnRadius;
        Vector2 spawnPosition = (Vector2)player.position +
randomPosition;

        Instantiate(enemyPrefab, spawnPosition, Quaternion.identity);
    }
}
```

кінець лістингу 3.5

Однією з основних механік гри є система накопичення досвіду та підвищення рівня персонажа. Для реалізації даної функціональності було створено окремий клас `PlayerExperience`, який відповідає за облік отриманого досвіду та перевірку умов переходу на новий рівень наведено у лістингу 3.6.

Після накопичення необхідної кількості досвіду виконується виклик методу `LevelUp()`, який підвищує рівень персонажа та збільшує вимоги до

наступного рівня. У майбутньому дана система може бути розширена додаванням вибору здібностей або покращення характеристик персонажа.

Лістинг 3.6 – Реалізація системи досвіду

```
using UnityEngine;

public class PlayerExperience : MonoBehaviour
{
    public int currentExperience;
    public int experienceToNextLevel = 10;
    public int level = 1;

    public void AddExperience(int amount)
    {
        currentExperience += amount;

        if (currentExperience >= experienceToNextLevel)
        {
            LevelUp();
        }
    }

    void LevelUp()
    {
        level++;
        currentExperience = 0;
        experienceToNextLevel += 5;

        Debug.Log("LevelUp: " + level);
    }
}
```

кінець лістингу 3.6

Під час розробки гри особлива увага приділялась оптимізації продуктивності. Оскільки у жанрі Bullet Heaven на сцені одночасно можуть знаходитися десятки або сотні ворогів і снарядів, часте створення та видалення об'єктів може негативно впливати на продуктивність гри. Для вирішення цієї проблеми використовується технологія Object Pooling.

Суть даного методу полягає у створенні певного набору об'єктів заздалегідь та їх повторному використанні замість постійного створення нових

екземплярів. Це дозволяє значно зменшити навантаження на систему керування пам'яттю та уникнути різких просідань частоти кадрів.

Для реалізації Object Pooling було створено менеджер пулу об'єктів, який зберігає список неактивних куль або ворогів та активує їх при необхідності. Після завершення використання об'єкт повертається назад у пул замість повного видалення.

Важливою частиною реалізації стала система користувацького інтерфейсу. Для створення UI використовувались інструменти Unity Canvas та UI Toolkit. У грі було реалізовано головне меню, меню налаштувань, панель здоров'я персонажа, шкалу досвіду та відображення статистики гри.

Для оновлення інформації на екрані було створено клас UIManager, який відповідає за зміну текстових елементів та індикаторів у реальному часі.

У коді використовується компонент Slider, який дозволяє наочно відображати поточний рівень здоров'я та досвіду персонажа. Значення шкал оновлюються відповідно до змін стану гравця (рис. 3.1).



Рисунок 3.1 – Відображення здоров'я та досвіду персонажа

Окрему увагу було приділено реалізації аудіосистеми. У грі використовуються звукові ефекти атак, отримання пошкоджень, підвищення рівня та фонові музика. Для керування звуком створено окремий клас

AudioManager, який відповідає за відтворення аудіофайлів та регулювання рівня гучності.

Використання окремих аудіоканалів для музики та звукових ефектів дозволяє незалежно регулювати рівень гучності кожного типу звуків у меню налаштувань.

Під час реалізації гри активно використовувалась система Prefab-об'єктів Unity. Prefab дозволяє створювати шаблони об'єктів, які можуть повторно використовуватись у різних сценах гри. Це значно спрощує роботу з ворогами, кулями, ефектами та іншими елементами ігрового середовища.

Для забезпечення коректної роботи всіх компонентів було використано систему менеджерів гри. Зокрема, GameManager відповідає за загальний стан гри, запуск та завершення ігрової сесії, підрахунок часу виживання та обробку стану програшу. Такий підхід дозволяє централізовано керувати основною логікою гри.

У процесі розробки також було реалізовано систему анімацій персонажа та ворогів. Для цього використовувались компоненти Animator та Animation Controller. Анімації змінюються залежно від напрямку руху персонажа, атаки або отримання пошкоджень. Це дозволяє зробити ігровий процес більш динамічним та візуально привабливим.

Після завершення основної реалізації проводилось тестування програмного продукту. Перевірялась коректність роботи системи руху персонажа, генерації ворогів, бойової системи, UI-компонентів та механізмів збереження ігрових параметрів. Також виконувалось тестування продуктивності гри при великій кількості активних об'єктів на сцені.

У результаті практичної реалізації було створено 2D гру жанру Bullet Heaven «Soul Harvest», яка забезпечує динамічний ігровий процес, систему розвитку персонажа, генерацію ворогів, систему атак та зручний користувацький інтерфейс. Використання рушія Unity та мови програмування C# дозволило реалізувати гнучку та масштабовану архітектуру програмного забезпечення, що

забезпечує можливість подальшого вдосконалення та розширення функціоналу гри.

3.2 Аналіз ігрових механік жанру Bullet Heaven та їх адаптація у грі «Soul Harvest»

Жанр Bullet Heaven є одним із сучасних напрямів розвитку 2D action-ігор, основною особливістю якого є поєднання динамічного бойового процесу, великої кількості ворогів на екрані та системи постійного розвитку персонажа. Ігри даного жанру орієнтовані на виживання гравця протягом максимально довгого часу в умовах безперервного зростання складності ворогів. Одними з найбільш відомих представників жанру є Vampire Survivors, Soulstone Survivors та Brotato [14].

Основною концепцією жанру Bullet Heaven є створення ігрового процесу, у якому персонаж автоматично або напіваавтоматично атакує ворогів, а основним завданням гравця є переміщення по ігровій карті, уникнення пошкоджень та вибір оптимальної стратегії розвитку персонажа. На відміну від жанру Bullet Hell, де головний акцент робиться на ухиленні від великої кількості ворожих снарядів, Bullet Heaven фокусується на масовому знищенні супротивників та поступовому посиленні героя.

Під час розробки гри «Soul Harvest» було проведено аналіз основних механік жанру Bullet Heaven та визначено елементи, які доцільно адаптувати у власному програмному продукті. Особлива увага приділялась механікам виживання, системі розвитку персонажа, генерації ворогів та побудові циклу ігрового процесу.

Однією з ключових механік жанру є безперервний ігровий процес із поступовим ускладненням. У грі «Soul Harvest» дана механіка реалізована шляхом поступового збільшення кількості ворогів та підвищення їх характеристик із плином часу. На початкових етапах гри вороги мають невелику швидкість руху та низький рівень здоров'я, що дозволяє гравцю ознайомитися з

базовими механіками керування. У подальшому складність гри поступово зростає, що створює постійне відчуття напруги та стимулює користувача вдосконалювати власну стратегію виживання.

Важливим елементом жанру Bullet Heaven є система автоматичних атак. У більшості ігор даного жанру персонаж самостійно атакує найближчих ворогів або здійснює атаки через певний проміжок часу. У грі «Soul Harvest» було реалізовано автоматичну систему створення снарядів, що дозволяє гравцю зосередитися на переміщенні персонажа та ухиленні від ворогів. Такий підхід робить ігровий процес більш динамічним та знижує навантаження на користувача під час гри.

Ще однією характерною особливістю жанру є наявність великої кількості ворогів на екрані. Для реалізації цієї механіки у грі використовується система хвильового створення супротивників. Вороги з'являються у випадкових точках навколо персонажа та постійно рухаються у його напрямку. Така механіка створює ефект постійного переслідування та підтримує високий темп гри.

У процесі аналізу жанру було встановлено, що одним із головних факторів утримання інтересу гравця є система прогресії персонажа. У грі «Soul Harvest» реалізовано механізм накопичення досвіду після знищення ворогів. Після досягнення певної кількості досвіду персонаж підвищує рівень та отримує можливість покращити власні характеристики або здібності. Дана система забезпечує поступове посилення героя та створює мотивацію для продовження гри.

Для адаптації механіки розвитку персонажа було використано декілька типів покращень, серед яких:

- збільшення рівня здоров'я персонажа;
- підвищення швидкості руху;
- збільшення швидкості атаки;
- покращення сили пошкодження;
- збільшення кількості одночасних снарядів;
- підвищення ефективності здібностей персонажа.

Використання системи випадкового вибору покращень після підвищення рівня дозволяє зробити кожне проходження гри унікальним та підвищує реіграбельність програмного продукту.

Однією з важливих особливостей ігор жанру Bullet Heaven є короткий цикл отримання винагороди. Гравець постійно отримує досвід, нові здібності, покращення та візуальні ефекти, що створює відчуття прогресу та підтримує інтерес до гри. У «Soul Harvest» даний принцип реалізується через часте підвищення рівня персонажа, систему візуальних ефектів та збільшення кількості ворогів у процесі проходження.

Для створення більш динамічного ігрового процесу було адаптовано систему поступового збільшення інтенсивності гри. Із часом на сцені з'являється більша кількість ворогів, а також можуть змінюватися їх характеристики, швидкість руху або тип поведінки. Такий підхід забезпечує постійне ускладнення гри та підтримує баланс між складністю та можливостями персонажа.

Окрему увагу під час розробки було приділено візуальному стилю гри. Для створення графічної складової використовувались готові 2D-ресурси з набору Undead Survivor Assets Pack, які відповідають стилістиці жанру Bullet Heaven. Використання готових графічних ресурсів дозволило забезпечити єдиний художній стиль гри та прискорити процес розробки програмного продукту.

Під час адаптації механік жанру також враховувались особливості користувацького досвіду. Інтерфейс гри було спроектовано таким чином, щоб він не перевантажував користувача великою кількістю інформації. Основні елементи інтерфейсу, зокрема шкала здоров'я, індикатор досвіду та статистика гри, розташовані у зручних частинах екрана та не заважають сприйняттю ігрового процесу.

У результаті аналізу та адаптації механік жанру Bullet Heaven було сформовано структуру ігрового процесу гри «Soul Harvest», яка поєднує динамічний бойовий процес, систему виживання, розвиток персонажа та поступове ускладнення гри. Реалізовані механіки забезпечують високу динаміку

ігрового процесу, підтримують зацікавленість користувача та створюють умови для подальшого розширення функціональних можливостей програмного продукту.

3.3 Організація ігрового процесу та баланс складності

Організація ігрового процесу є одним із найважливіших етапів розробки комп'ютерної гри, оскільки саме від побудови основних механік залежить рівень зацікавленості користувача та загальна якість взаємодії з програмним продуктом. Для гри жанру Bullet Heaven особливо важливими є динамічність ігрового процесу, поступове ускладнення гри, система винагород та забезпечення балансу між можливостями персонажа і рівнем загрози з боку ворогів.

Основою ігрового процесу у грі «Soul Harvest» є механіка виживання. Головним завданням гравця є якомога довше залишатися живим у середовищі постійного нападу ворогів. Ігровий цикл побудований таким чином, щоб забезпечити безперервну взаємодію користувача з ігровим світом та підтримувати високий рівень динаміки протягом усієї ігрової сесії.

Ігровий процес починається із запуску гри та переходу до головного меню, де користувач може розпочати нову гру. Після початку гри створюється персонаж, запускається система генерації ворогів та активується таймер ігрової сесії. Основний цикл гри складається з переміщення персонажа, знищення ворогів, накопичення досвіду та поступового підвищення рівня героя.

Для забезпечення комфортного керування було реалізовано просту систему управління персонажем за допомогою клавіатури. Основна увага користувача зосереджується не на складних комбінаціях дій, а на правильному переміщенні персонажа по ігровому полю та уникненні зіткнень із ворогами. Такий підхід відповідає концепції жанру Bullet Heaven, де важливу роль відіграє стратегія руху та вибір оптимального маршруту пересування.

Одним із ключових елементів організації ігрового процесу є система автоматичних атак. Персонаж автоматично створює снаряди через певний

проміжок часу, що дозволяє гравцю зосередитися на виживанні та аналізі ситуації на екрані. Зі збільшенням рівня персонажа параметри атаки можуть покращуватися, що створює відчуття прогресу та підсилення героя.

Для підтримання інтересу користувача було реалізовано систему поступового ускладнення гри. На початкових етапах кількість ворогів є відносно невеликою, а їх характеристики дозволяють гравцю адаптуватися до механік керування. Із плином часу складність поступово зростає шляхом:

- збільшення кількості ворогів;
- підвищення швидкості їх руху;
- збільшення рівня здоров'я супротивників;
- скорочення інтервалу між появою нових ворогів;
- ускладнення поведінки окремих типів супротивників.

Поступове підвищення складності дозволяє уникнути різких стрибків навантаження на користувача та забезпечує природний розвиток ігрового процесу. Такий підхід позитивно впливає на утримання уваги гравця та підтримує інтерес до проходження гри.

Важливим елементом балансу складності є система розвитку персонажа. У процесі гри користувач отримує очки досвіду за знищення ворогів. Після накопичення необхідної кількості досвіду персонаж переходить на новий рівень та отримує покращення характеристик. Це дозволяє компенсувати поступове зростання складності гри та забезпечує баланс між силою героя та кількістю ворогів.

Під час проєктування балансу було враховано співвідношення між темпом розвитку персонажа та швидкістю ускладнення гри. Якщо персонаж розвивається надто швидко, гра втрачає складність та перестає бути цікавою. У випадку надмірної складності користувач може швидко втратити інтерес через постійні поразки. Тому система прогресії була налаштована таким чином, щоб складність гри зростала поступово та відповідала можливостям персонажа на певному етапі проходження.

У грі «Soul Harvest» також реалізовано механізм випадкового вибору покращень після підвищення рівня. Гравець може отримати різні бонуси, серед яких:

- збільшення швидкості атаки;
- покращення сили пошкодження;
- збільшення швидкості пересування;
- підвищення максимального рівня здоров'я;
- збільшення кількості одночасних снарядів;
- покращення спеціальних здібностей.

Дана система забезпечує варіативність проходження та підвищує реіграбельність гри. Користувач може формувати власний стиль проходження залежно від вибраних покращень.

Для підтримання динаміки ігрового процесу було використано механізм постійної активності на сцені. У будь-який момент часу на екрані присутні вороги, снаряди, візуальні ефекти та анімації. Це створює відчуття інтенсивного бойового процесу та відповідає особливостям жанру Bullet Heaven.

Під час організації ігрового процесу також враховувались психологічні аспекти взаємодії користувача з грою. Одним із важливих факторів є система короткострокових винагород. Гравець регулярно отримує нові рівні, покращення, візуальні ефекти та збільшення кількості знищених ворогів, що формує відчуття постійного прогресу та стимулює продовження гри.

Для забезпечення балансу між складністю та продуктивністю системи було реалізовано обмеження максимальної кількості активних ворогів на сцені. Це дозволяє уникнути надмірного навантаження на процесор та забезпечує стабільну частоту кадрів навіть під час тривалих ігрових сесій.

Окрему увагу приділено системі появи ворогів. Супротивники генеруються у випадкових точках навколо персонажа, однак на безпечній відстані від гравця. Це дозволяє уникнути ситуацій миттєвого отримання пошкоджень після появи ворога та забезпечує справедливість ігрового процесу.

У межах організації ігрового процесу також було реалізовано систему завершення гри. Після втрати всіх очок здоров'я гра переходить у стан Game Over, а користувачу відображається статистика проходження, зокрема час виживання та кількість знищених ворогів. Це мотивує гравця покращувати власні результати під час наступних ігрових сесій.

Таким чином, організація ігрового процесу у грі «Soul Harvest» базується на поєднанні механік виживання, автоматичних атак, поступового ускладнення та системи розвитку персонажа. Реалізований баланс складності забезпечує комфортне входження у гру для нових користувачів та підтримує високий рівень динаміки для більш досвідчених гравців. Використані підходи дозволили створити цілісний ігровий процес, який відповідає сучасним особливостям жанру Bullet Heaven та забезпечує тривале утримання інтересу користувача.

3.4 Тестування та налагодження 2D гри жанру Bullet Heaven «Soul Harvest»

Одним із важливих етапів розробки програмного забезпечення є тестування та налагодження системи. Основною метою тестування гри «Soul Harvest» була перевірка коректності роботи реалізованих ігрових механік, забезпечення стабільності програмного продукту та виявлення можливих помилок під час виконання гри. У процесі тестування також оцінювалась продуктивність системи при великій кількості одночасно активних об'єктів, що є особливо важливим для ігор жанру Bullet Heaven.

Тестування гри виконувалось протягом усього процесу розробки. Після реалізації кожного нового функціонального модуля проводилась перевірка його працездатності та взаємодії з іншими компонентами системи. Такий підхід дозволив своєчасно виявляти помилки та уникати накопичення критичних проблем у програмному коді.

У процесі тестування використовувались методи функціонального, інтеграційного та системного тестування. Функціональне тестування

застосовувалось для перевірки окремих елементів гри, зокрема системи керування персонажем, генерації ворогів, бойової системи, інтерфейсу користувача та механізму підвищення рівня персонажа. Інтеграційне тестування дозволило перевірити взаємодію між різними модулями програмного забезпечення, а системне тестування використовувалось для оцінки роботи гри в цілому [15].

На початковому етапі тестування перевірялась система керування персонажем. Було протестовано коректність переміщення гравця в усіх напрямках, стабільність роботи фізики та реакцію персонажа на натискання клавіш. Особлива увага приділялась плавності руху та відсутності затримок при обробці введення користувача.

Наступним етапом стало тестування системи генерації ворогів. Перевірялась правильність створення ворогів у випадкових точках навколо персонажа, коректність їхнього руху та поступове збільшення складності гри. У процесі тестування було виявлено проблему надмірного накопичення ворогів на сцені, що призводило до зниження продуктивності гри. Для усунення даного недоліку було оптимізовано систему генерації та реалізовано механізм Object Pooling.

Під час тестування бойової системи перевірялась коректність створення снарядів, обробка зіткнень між кулями та ворогами, а також правильність нарахування пошкоджень. Було встановлено, що при великій кількості одночасних зіткнень виникали короткочасні просідання частоти кадрів. Для вирішення цієї проблеми було оптимізовано систему фізики та зменшено кількість одночасних обчислень зіткнень.

Особлива увага приділялась тестуванню системи розвитку персонажа. Перевірялась правильність нарахування очок досвіду, підвищення рівня персонажа та робота механізмів покращення характеристик. У результаті тестування було усунуто помилки, пов'язані з некоректним оновленням UI-елементів після зміни рівня персонажа.

Тестування користувацького інтерфейсу включало перевірку роботи головного меню, меню налаштувань, шкали здоров'я, індикатора досвіду та відображення статистики гри. Також перевірялась адаптація інтерфейсу до різних роздільних здатностей екрана. У процесі тестування було покращено розташування UI-елементів та збільшено читабельність тексту.

Для налагодження програмного коду використовувались вбудовані інструменти Unity та середовище розробки Visual Studio. Основними засобами налагодження були:

- використання консолі Unity для виведення повідомлень про помилки;
- покрокове виконання коду у Visual Studio;
- перевірка значень змінних у процесі виконання програми;
- аналіз продуктивності за допомогою Unity Profiler;
- тестування фізики та зіткнень у режимі реального часу.

У процесі тестування також проводився аналіз продуктивності гри. Для цього використовувався інструмент Unity Profiler, який дозволяє відстежувати навантаження на процесор, використання оперативної пам'яті та частоту кадрів.

Після завершення всіх етапів тестування було підтверджено стабільну роботу гри та коректність функціонування основних механік. Виявлені помилки та недоліки були усунуті шляхом оптимізації програмного коду, покращення системи керування ресурсами та вдосконалення користувацького інтерфейсу.

Таким чином, проведення тестування та налагодження дозволило забезпечити стабільність роботи 2D гри жанру Bullet Heaven «Soul Harvest», підвищити продуктивність програмного забезпечення та покращити загальну якість взаємодії користувача з ігровим середовищем.

ВИСНОВКИ

У кваліфікаційній роботі було досягнуто поставлену мету, яка полягала у розробці 2D гри жанру Bullet Heaven «Soul Harvest» засобами Unity з використанням мови програмування C#. У процесі виконання роботи було реалізовано повноцінний програмний продукт із динамічним ігровим процесом, системою виживання, бойовими механіками та системою розвитку персонажа.

У межах роботи було проведено аналіз особливостей та принципів побудови ігор жанру Bullet Heaven. Досліджено характерні механіки сучасних ігор даного жанру, серед яких система автоматичних атак, велика кількість ворогів на сцені, поступове ускладнення ігрового процесу та система прогресії персонажа. На основі проведеного аналізу було визначено основні механіки, які доцільно реалізувати у грі «Soul Harvest».

Під час виконання роботи було досліджено можливості ігрового рушія Unity та мови програмування C#. Визначено переваги використання Unity для створення 2D ігор, зокрема підтримку компонентного підходу, системи фізики, анімації, UI-елементів та інструментів оптимізації. Використання мови C# дозволило реалізувати модульну архітектуру програмного забезпечення та забезпечити гнучкість у процесі розробки.

У роботі було визначено функціональні та нефункціональні вимоги до програмного продукту. Сформовано перелік основних функцій гри, зокрема систему керування персонажем, генерацію ворогів, бойову систему, систему розвитку героя, інтерфейс користувача та механізми збереження ігрових параметрів. Також було визначено вимоги до продуктивності, стабільності роботи гри та зручності взаємодії користувача з ігровим середовищем.

Виконано проєктування архітектури програмного забезпечення гри. Побудовано UML-діаграми варіантів використання, класів та діяльності, що дозволило формалізувати структуру системи та визначити взаємодію між основними компонентами гри. Використання об'єктно-орієнтованого підходу

забезпечило модульність програмного коду та спростило процес подальшого розширення функціональних можливостей програмного продукту.

У процесі практичної реалізації було створено основні ігрові механіки гри «Soul Harvest». Реалізовано систему керування персонажем, систему автоматичних атак, генерацію хвиль ворогів, обробку зіткнень між об'єктами та систему отримання пошкоджень. Також було створено механізм накопичення досвіду та підвищення рівня персонажа із можливістю покращення характеристик героя.

Було реалізовано користувацький інтерфейс гри, який включає головне меню, меню налаштувань, шкалу здоров'я, індикатор досвіду та статистику проходження. Для створення візуальної складової використовувались готові графічні ресурси з набору Undead Survivor Assets Pack, що дозволило забезпечити єдиний художній стиль гри та прискорити процес розробки.

Особлива увага у роботі приділялась питанням тестування та оптимізації програмного забезпечення. Було проведено функціональне, інтеграційне та системне тестування гри. У процесі тестування виявлено та усунено недоліки, пов'язані з продуктивністю та обробкою великої кількості об'єктів на сцені. Для підвищення стабільності роботи гри було використано механізм Object Pooling та оптимізовано систему фізичних обчислень.

Результати виконаної роботи можуть бути використані у сфері розробки незалежних комп'ютерних ігор, навчальних проєктів та досліджень у галузі створення 2D ігрових застосунків. Практичне значення роботи полягає у створенні повноцінного програмного продукту, який може бути основою для подальшого розвитку та вдосконалення гри.

Отримані результати можуть бути використані у навчальному процесі під час вивчення дисциплін, пов'язаних із розробкою комп'ютерних ігор, програмуванням мовою C# та використанням ігрового рушія Unity. Також результати роботи можуть бути корисними для подальших досліджень у напрямку оптимізації ігрових систем та вдосконалення механік жанру Bullet Heaven.

Подальший розвиток програмного продукту може включати реалізацію нових типів ворогів, системи босів, додаткових ігрових рівнів, розширення системи здібностей персонажа, реалізацію багатокористувацького режиму та вдосконалення штучного інтелекту ворогів. Також перспективним напрямом є адаптація гри для мобільних платформ та розширення системи візуальних ефектів.

Таким чином, усі поставлені завдання кваліфікаційної роботи було виконано у повному обсязі, а розроблена 2D гра жанру Bullet Heaven «Soul Harvest» відповідає поставленим функціональним і технічним вимогам та демонструє можливості використання рушія Unity і мови програмування C# для створення сучасних ігрових програмних продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ігри XXI сторіччя. URL: <https://mezha.ua/articles/novi-zhanri-videoigor-306481/> (дата звернення: 28.04.2026).
2. Вікіпедія. Vampire Survivors-like, також відомий як bullet heaven. URL: https://uk.wikipedia.org/wiki/Vampire_Survivors%E2%80%93like (дата звернення: 28.04.2026).
3. Розробка ігор на Unity (C#). URL: <https://arionisgames.com/uk/services/gamedev-engines/unity/> (дата звернення: 28.04.2026).
4. Vampire Survivors. URL: https://uk.wikipedia.org/wiki/Vampire_Survivors (дата звернення: 28.04.2026).
5. Stage. Vampire Survivors Wiki. URL: <https://vampire.survivors.wiki/w/Stages> (дата звернення: 28.04.2026).
6. Brotato – неймовірна різноманітність Vampire Survivors. URL: https://kutok.io/pxperspective/brotato_neimovirna_riznovydnist_vampire_survivors_ta_the_binding_of_isaac-ai1 (дата звернення: 28.04.2026).
7. Mod: Dami's Overhaul Mod. URL: https://brotato.wiki.spellsandguns.com/Mod:Dami%27s_Overhaul_Mod (дата звернення: 28.04.2026).
8. 20 Minutes Till Dawn. URL: https://uk.wikipedia.org/wiki/20_Minutes_Till_Dawn (дата звернення: 28.04.2026).
9. Що таке функціональні вимоги: приклад та шаблони. URL: <https://surl.li/uarokm> (дата звернення: 05.05.2026).
10. Що таке UX/UI дизайн ігрових інтерфейсів та хто ними займається. URL: <https://varto.school/blog/shho-take-ux-ui-dyzajn-igrovyh-interfejsiv-ta-hto-nym-zajmayetsya/> (дата звернення: 05.05.2026).
11. Найпопулярніші ігрові рушії Steam: Unity, Unreal, а які ж інші? URL: <https://mezha.ua/articles/naypopuliarnishi-ihrovi-rushii-steam/> (дата звернення: 05.05.2026).

12. Що таке об'єктно-орієнтоване програмування: принципи, переваги та недоліки. URL: <https://dan-it.com.ua/uk/blog/chto-takoe-obektno-orientirovannoe-programmirovanie-principy-preimushhestva-i-nedostatki/> (дата звернення: 05.05.2026).

13. Як оптимізувати ігровий проект за допомогою патерну ObjectPool. URL: <https://gamedev.dou.ua/blogs/optimizing-a-game-project-with-object-pool-pattern/> (дата звернення: 05.05.2026).

14. Найкращі ігри в дусі Vampire Survivors. URL: <https://mezha.ua/articles/naykrashchi-igri-v-dusi-vampires-survivors-308363/> (дата звернення: 05.05.2026).

15. Види функціонального та нефункціонального тестування. URL: <https://dan-it.com.ua/uk/blog/vidy-funkcionalnogo-i-nefunkcionalnogo-testirovaniya/> (дата звернення: 05.05.2026).