МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення

Євген ДАВИДЕНКО__

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

3D-гра в жанрі roguelike на ігровому рушії Unreal Engine 5. Розробка ігрової механіки.

Спеціальність 121 Інженерія програмного забезпечення Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Іван ГАВРИЛОВ**

«__» _____20___р.

Керівник роботи
старша викладачка

_____ **Світлана БОРОВЛЬОВА**

«__» _____20___р.

Миколаїв 2025

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп’ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«____» _____2025 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Гаврилов Іван

1. Тема кваліфікаційної роботи «3D-гра в жанрі roguelike на ігровому рушії Unreal Engine 5. Розробка ігрової механіки.» затверджена наказом ректора ЧНУ ім. Петра Могили № 315 від «13» листопада 2024 р.
2. Строк представлення кваліфікаційної роботи «___» червня 2025 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Розроблений Ігровий застосунок в жанрі Roguelike

4. Перелік питань, що підлягають розробці: методи процедурної генерації рівнів у сучасних іграх, використання інтелектуальної поведінки ворогів у жанрі Roguelike, перевірка стабільності роботи системи процедурної генерації, тестування інтелектуальної поведінки ворогів у різних сценаріях.
5. Перелік графічних матеріалів: презентація
6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «15» __листопада_ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **«3D-гра в жанрі roguelike на ігровому рушії Unreal Engine 5. Розробка ігрової механіки.»**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	**21.10.2024**	**13.11.2024**	**Виконано**
2.	Огляд літератури за темою роботи	**06.12.2024**	**10.12.2024**	**Виконано**
3.	Складання календарного плану КБР	**15.01.2025**	**17.01.2025**	**Виконано**
4.	Аналіз предметної області	**18.01.2025**	**21.01.2025**	**Виконано**
5.	Розробка проєктних рішень	**25.01.2025**	**30.01.2025**	**Виконано**
6.	Моделювання та конструювання ПЗ	**11.02.2025**	**20.04.2025**	**Виконано**
7.	Розробка ШІ для ворогів	**20.02.2025**	**04.03.2025**	**Виконано**
8.	Розробка системи слуху для ворогів	**25.02.2025**	**04.03.2025**	**Виконано**
9.	Розробка процедурної генерації	**27.02.2025**	**10.03.2025**	**Виконано**
10.	Тестування розробленої системи	**10.03.2025**	**11.03.2025**	**Виконано**
11.	Попередній захист	**24.05.2025**	**24.05.2025**	**Виконано**
12.	Завершення оформлення КБР та презентації	**25.05.2025**	**10.06.2025**	**Виконано**
13.	Рецензування			
14.	Захист кваліфікаційної роботи			

Здобувач _____ **Гаврилов Іван**

«26» листопада 2025 р.

Керівник роботи

старший викладач

_____ **Світлана БОРОВЛЬОВА**

«26» листопада 2025р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

"Ігровий застосунок в жанрі roguelike на рушії Unreal Engine 5. Розробка ігрових механік"

Здобувач 409 гр.: Гаврилов Іван

Керівник: ст. викладачка Світлана Боровльова

Актуальність теми розробки ігрового застосунку в жанрі roguelike на рушії Unreal Engine 5 зумовлена зростанням популярності цього жанру завдяки процедурній генерації рівнів, перманентній смерті персонажа та високій реіграбельності. Ігри на кшталт *Hades, Dead Cells* та *Risk of Rain 2* довели, що навіть малі студії можуть досягти успіху завдяки глибоким механікам і якісній реалізації.

Метою кваліфікаційної роботи є створення ігрового застосунку на основі Unreal Engine 5, для популяризації жанру roguelike на сучасному ринку.

Об’єктом роботи є процес створення roguelike-ігор з використанням процедурної генерації.

Предметом роботи виступає програмне забезпечення для створення roguelike-ігор на базі Unreal Engine 5.

Робота складається з вступу, 4 розділів та висновків. У вступі визначено актуальність теми, сформульовано мету, об’єкт, предмет та завдання роботи. У першому розділі проведено аналіз предметної області, розглянуті аналогічні застосунки, визначено їх переваги та недоліки в порівнянні із створюваною грою. У другому розділі описано порівняння рушіїв та специфікації вимог до програмного забезпечення. У третьому розділі розглянуто варіанти використання системи, проєктування функцій програмного забезпечення та описано інтерфейсу користувача. У четвертому розділі реалізовано та протестовано програмне забезпечення. У висновках підсумовано результати та перспективи подальшого розвитку застосунку.

КРБ викладена на 38 сторінках, вона містить 4 розділи, 21 ілюстрацій, 4 таблиці, 14 джерел в переліку посилань.

Ключові слова: генерація, Roguelike, NPC, реіграбельність

ABSTRACT

to the qualifying bachelor's thesis

"Roguelike Game Application on Unreal Engine 5: Game Mechanics Development"

Applicant 409 gr.: Havrylov Ivan

Supervisor: Senior Lecturer Svetlana Borovlova

The relevance of developing a game application in the roguelike genre using Unreal Engine 5 lies in the growing popularity of this genre due to procedural level generation, permanent character death, and high replayability. Games like *Hades*, *Dead Cells*, and *Risk of Rain 2* have proven that even small studios can achieve success through deep mechanics and high-quality implementation.

The goal of the qualification work is to develop a game application based on Unreal Engine 5 to promote the roguelike genre in the modern market.

The **object** of the work is the process of creating roguelike games using procedural generation.

The **subject** of the work is the software for developing roguelike games on Unreal Engine 5.

The work consists of an introduction, four chapters, and conclusions. The introduction defines the relevance of the topic, formulates the aim, object, subject, and tasks of the study. The first chapter provides an analysis of the subject area and defines the main requirements for the application. The second chapter describes a comparison of game engines and the specification of software requirements. The third chapter explores system usage scenarios, software functionality design, and provides a description of the user interface. The fourth chapter presents the implementation and testing of the software. The conclusions summarize the results and outline the prospects for further development of the application.

The qualification work is presented on 38 pages, containing 4 chapters, 21 illustrations, 4 tables, and 14 sources in the reference list.

Keywords: generation, Roguelike, NPC, replayability

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ ОСОБЛИВОСТЕЙ РОЗРОБКИ ТА РЕАЛІЗАЦІЇ ROGUELIKE-ІГОР ... 6
1.1 Опис предметної області 6
1.2 Аналіз існуючих Roguelike-ігор 7
1.3 Опис і аналіз структурних і функціональних особливостей об’єкта 10
Висновки до розділу 1 12
2 МОДЕЛЮВАННЯ ОБ’ЄКТУ ТА ПРЕДМЕТУ 14
2.1 Порівняння рушіїв 14
2.2 Специфікація вимог до програмного забезпечення (SRS) 17
Висновки до розділу 2 18
3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 20
3.1 Варіанти використання системи 20
3.2 Проектування функцій програмного забезпечення 23
3.3 Опис інтерфейсу користувача 26
Висновки до розділу 3 28
4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЕКТУ 29
4.1 Реалізація проєкту 29
4.2 Тестування проєкту 32
Висновки до розділу 4 34
ВИСНОВКИ 35
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 37

ПЕРЕЛІК СКОРОЧЕНЬ

AI	–	Artificial Intelligence
API	–	Application Programming Interface
GDScript	–	Godot scripting language engine
NavMesh	–	Navigation Mesh
NPC	–	Non-Player Character
PCG	–	Procedural Content Generation
UES	–	Unreal Engine 5

ВСТУП

Актуальність теми розробки ігрового застосунку в жанрі roguelike на рушії Unreal Engine 5 полягає в тому, що цей жанр останніми роками набуває все більшої популярності серед геймерів завдяки своїй унікальній механіці процедурної генерації рівнів, перманентній смерті персонажа та високому рівню реіграбельності. Такі ігри, як Hades, Dead Cells та Risk of Rain 2, довели, що навіть невеликі студії можуть досягти комерційного успіху за рахунок глибоких ігрових механік і якісної реалізації.

Використання Unreal Engine 5 для створення roguelike-проекту є особливо перспективним через потужні технології рушія, такі як Nanite для деталізованої геометрії, Lumen для динамічного освітлення та MetaHuman для реалістичних персонажів. Ці інструменти дозволяють розробникам створювати якісні візуальні ефекти без надмірних витрат ресурсів, що особливо важливо для невеликих команд.

Крім того, розробка ігрових механік у жанрі roguelike вимагає ретельного балансування систем прокачування, генерації рівнів та бойових механік, що робить цю тему цікавою з точки зору геймдизайну та програмування. Впровадження таких елементів, як перманентні покращення, випадкові предмети та нелінійність проходження, відкриває широкі можливості для експериментів та інновацій у ігровій механіці.

Метою кваліфікаційної роботи є створення ігрового застосунку на основі Unreal Engine 5, для популяризації жанру roguelike на сучасному ринку.

Відповідно до мети визначено такі завдання:

– провести пошук існуючих технологій та підходів до розробки Roguelike-ігор;

– виконати проєктування і моделювання структури системи;

– створити алгоритм процедурної генерації тривимірного ігрового простору;

– реалізувати основні ігрові механіки, зокрема бойову систему, механізми взаємодії з об'єктами та інвентаризацію;

– реалізувати систему розвитку персонажа та вибору здібностей;

– розробити поведінкову логіку NPC із застосуванням Blueprints та програмування на C++.

Об’єктом роботи є процес створення roguelike-ігор з використанням процедурної генерації.

Предметом роботи виступає програмне забезпечення для створення roguelike-ігор на базі Unreal Engine 5, що забезпечує реалізацію процедурної генерації, інтерактивного геймплею та сучасних візуальних ефектів.

1 АНАЛІЗ ОСОБЛИВОСТЕЙ РОЗРОБКИ ТА РЕАЛІЗАЦІЇ ROGUELIKE-ІГОР

1.1 Опис предметної області

Розробка ігрового застосунку в жанрі roguelike на рушії Unreal Engine 5 охоплює широкий спектр сучасних технологій та методів, пов'язаних із створенням динамічних ігрових світів із високим рівнем реіграбельності. Roguelike як жанр характеризується трьома ключовими аспектами: процедурною генерацією рівнів, перманентною смертю персонажа та нелінійним прогресуванням. Ці особливості забезпечують унікальний ігровий досвід при кожному новому проходженні.

За результатами досліджень (GameDev Market, 2024), інтерес до незалежних ігор в Україні зріс на понад 40%, причому особливої популярності набули саме ті проєкти, що пропонують гравцеві варіативність, самостійне прийняття рішень та високий рівень залучення. Roguelike-ігри задовольняють ці запити через динамічний геймплей, процедурну генерацію контенту та унікальний досвід кожної нової спроби. Ці ігри дають змогу гравцеві відчути контроль, навіть у віртуальному просторі, що є важливим елементом у ситуації загальної невизначеності[1].

Особливо актуальним стає створення таких ігор в українському контексті:

– військові ветерани та учасники бойових дій, які мають потребу в психологічному відновленні, можуть знайти в ігровому процесі елемент терапії або відволікання;

– молодь, яка є основною аудиторією ігор, шукає інноваційні та якісні локальні продукти; цивільні у зоні бойових дій (42%)[1].

Стратегії подолання стресу:

– розробники прагнуть створювати ігри зі змістом, закладати в них культурні

– пасивні методи релаксації: використання спокійної музики, візуально м’якої палітри та неспішного темпу геймплею.

Серед основних стратегій взаємодії користувачів з roguelike-іграми можна виділити:

– соціальну інтеграцію (кооперативний геймплей або обмін досвідом через стримінг та форуми);

– релаксаційні механіки (медитативна музика, художній стиль);

– високу реіграбельність, яка дозволяє кожному гравцю проходити гру по-своєму.

Окрему увагу варто звернути на формування здорової ігрової культури. За даними спільноти GDBU (2025), понад 80% українських гравців підтримують розвиток індустрії національного виробництва, а 68% готові надавати зворотний зв’язок, підтримку або донати розробникам незалежних ігор.

1.2 Аналіз існуючих Roguelike-ігор

У межах аналізу сучасних Roguelike-ігор подано короткі описи вибраних проєктів для подальших геймплейних і технічних рішень.

Dead Cells (Motion Twin) – 2D Roguelike-платформер з елементами Metroidvania, який поєднує швидкий екшн і процедурно згенеровані рівні. Гравець досліджує величезні лабіринти, бореться з ворогами та босами, відкриває нові навички і вдосконалює зброю. Гра характеризується високим рівнем складності, перманентною смертю та великою кількістю предметів і здібностей, що дозволяє кожен раз створювати нові комбінації для проходження[2].

Рисунок 1.1 – Скріншот геймплею Dead Cells

Darkest Dungeon (Red Hook Studios) – тактичний рогалик з елементами психології, де гравець керує групою героїв, які досліджують зловісні підземелля та борються з ворогами. Гра фокусується на стресі та психічному стані персонажів, що впливає на їх поведінку в бою. Кожна поразка може призвести до серйозних наслідків для команди. Рівні генеруються випадковим чином, і завдання включають не тільки бій, але й управління здоров'ям, ресурсами та моральним станом персонажів[3].

Рисунок 1.2 – Скріншот геймплею Darkest Dungeon

Risk of Rain 2 (Hopoo Games) – 3D-рогалик-шутер, виконаний на Unreal Engine 4, який поєднує елементи екшену та кооперативної гри. Гравці досліджують процедурно згенеровані інопланетні ландшафти, борючись з хвилями ворогів та босами. У грі присутня система динамічної складності, яка з часом збільшується, додаючи ще більше викликів. Активно використовуються різноманітні предмети, які гравці можуть комбінувати для створення унікальних ефектів і тактик[4].

Рисунок 1.3 – Скріншот геймплею Risk of Rain 2

Таблиця 1.1 – Порівняння існуючих програмних рішень

Гра	Dead Cells	Darkest Dungeon	Risk of Rain 2
Рушій	Unity	Proprietary	Unity
PCG-метод	Процедурні рівні, випадкові з'єднання кімнат	Рукописні локації з випадковими елементами	Процедурний layout рівнів
Головні механіки	Метроїдванія, перманентні смерті, комбо-система	Покрокові бої, стрес-механіка, перманентні смерті	Шутер, кооператив, система предметів
Сильні сторони	Швидкий екшен, випадкові рівні, перманентні покращення	Глибокі тактичні бої, випадкові події, стрес-система	Швидкий кооперативний шутер, процедурні рівні
Обмеження	Може бути складним для новачків	Високий рівень складності, відсутність кооперативу	Залежність від командного досвіду, випадковість предметів

Dead Cells, Darkest Dungeon та Risk of Rain 2 демонструють різні підходи до Roguelike-механік: від швидкого 2D-екшену з метроідванією до тактичного стресу та 3D-шутера з кооперацією. Ключові елементи — процедурна генерація, перманентні наслідки, складність і глибокі системи предметів. Ці ігри можуть бути орієнтиром для розробки власного Roguelike-проєкту, враховуючи їх успішні геймплейні та технічні рішення.

1.3 Опис і аналіз структурних і функціональних особливостей об’єкта

Об’єкт – процес процедурної генерації тривимірного ігрового середовища в Unreal Engine 5 – охоплює сукупність взаємодіючих модулів, відповідальних за формування, візуалізацію та підтримку динамічної структури світу. Далі наведено характеристику основних підсистем із аналізом їхніх функціональних особливостей.

Таблиця 1.2 – Підсистеми та їх функціонал

Підсистема / модуль	Структурні складові	Функціональні можливості
Моделювання твердого тіла (Solid Body Modeling)	Static Mesh, Skeletal Mesh	Реалістичні зіткнення досягаються налаштуванням маси, тертя та пружності об'єктів. Використання шарів колізій оптимізує продуктивність. правдоподібну взаємодію.

Продовження Таблиці 1.2

Підсистема / модуль	Структурні складові	Функціональні можливості
Створення матеріалів (Material Creator)	Material Editor; Texture Library; Material Instances	Создание материалов в редакторе с настройкой цвета, прозрачности и блеска. Использование Material Instances для вариаций и шейдеров для спецэффектов.
Контролер розташування об’єктів (Spawner)	Actor Spawner; Foliage Tool	Розміщення об'єктів відбувається за шаблонами з нерівномірним розподілом, включаючи активні та декоративні елементи, з автоматичним оновленням NavMesh для NPC.
DataTable з характеристиками NPC (DataTable for NPC (AI) Attributes)	DataTable; Attribute Set; NPC Config	Зберігання параметрів NPC (тип, здоров’я, шкода, швидкість); Централізоване керування характеристиками; Швидке зчитування й оновлення даних;

Кінець таблиці 1.2

Підсистема / модуль	Структурні складові	Функціональні можливості
DataTable з характеристиками предметів (Item Attributes DataTable)	DataTable; Item Attributes; Item Config	Зберігання параметрів предметів (тип, рідкість, шкода, захист, вага, ефекти); Централізоване керування характеристиками для балансу гри; Швидке зчитування й оновлення даних; Інтеграція з інвентарними та бойовими системами.

Процедурна генерація в Unreal Engine 5 реалізована через систему взаємопов'язаних модулів: фізичне моделювання об'єктів, створення матеріалів, інструменти розміщення елементів середовища та централізовані DataTables для управління параметрами NPC і предметів. Ця структура забезпечує ефективне створення динамічних 3D-світів з автоматизованою навігацією та оптимізованим рендерингом.

Висновки до розділу 1

Проведений аналіз підтвердив високу актуальність Roguelike-ігор у сучасному ігровому середовищі, що зумовлено зростаючим попитом на динамічний, непередбачуваний і реіграбельний ігровий досвід. Вивчення ключових механік жанру – процедурної генерації, перманентної смерті, нелінійного прогресу та високої складності – засвідчило їхню ефективність у

створенні глибокої залученості гравців, стимулюванні повторного проходження та формуванні унікальних сценаріїв гри кожного разу.

Порівняльний аналіз провідних проєктів жанру, таких як Dead Cells, Darkest Dungeon і Risk of Rain 2, дозволив виокремити оптимальні дизайнерські рішення щодо генерації контенту, балансу складності, побудови систем прогресу та елементів метаігрової взаємодії. Ці ігри демонструють, як грамотне використання ключових механік Roguelike може забезпечити як комерційний успіх, так і визнання серед гравців.

Важливим є також технічний аспект – сучасні рушії, зокрема Unreal Engine 5, відкривають нові горизонти у створенні процедурного контенту завдяки потужним інструментам Blueprints, Niagara, MetaSounds і можливостям масштабованої генерації рівнів. Це створює перспективи для впровадження інновацій у жанрі та досягнення високої якості при оптимізації виробничого процесу.

Таким чином, отримані результати не лише підтверджують доцільність і ефективність використання методів процедурної генерації в Roguelike-іграх, але й формують обґрунтовану основу для розробки власного ігрового проєкту. Запропонований підхід дозволяє поєднати інноваційність, технічну досконалість і глибоку геймплейну взаємодію, що має всі шанси на успішну реалізацію в умовах сучасного ігрового ринку.

2 МОДЕЛЮВАННЯ ОБ’ЄКТУ ТА ПРЕДМЕТУ

2.1 Порівняння рушіїв

При виборі рушія для реалізації процедурної генерації та AI важливо враховувати його архітектурну гнучкість і ефективність оптимізації. Для цього було розглянуто три рушії: Unreal Engine 5, Unity і Godot.

Unreal Engine 5 об'єднує потужний C++ API з візуальним скриптуванням Blueprints, що спрощує створення прототипів і оптимізацію PCG-модулів. Технологія Nanite автоматично регулює рівень деталізації геометрії, а Lumen забезпечує реалістичне глобальне освітлення і динамічні відбиття без необхідності випікати lightmaps. Інструменти Data Layers і World Partition дозволяють ефективно керувати великими відкритими світами та підвантаженням підрівнів. Для AI Unreal Engine 5 пропонує Behavior Trees, Environment Query System (EQS) і вбудовану підтримку MetaHumans, а також можливість інтеграції з зовнішніми ML-моделями через плагіни Python і TensorFlow[6].

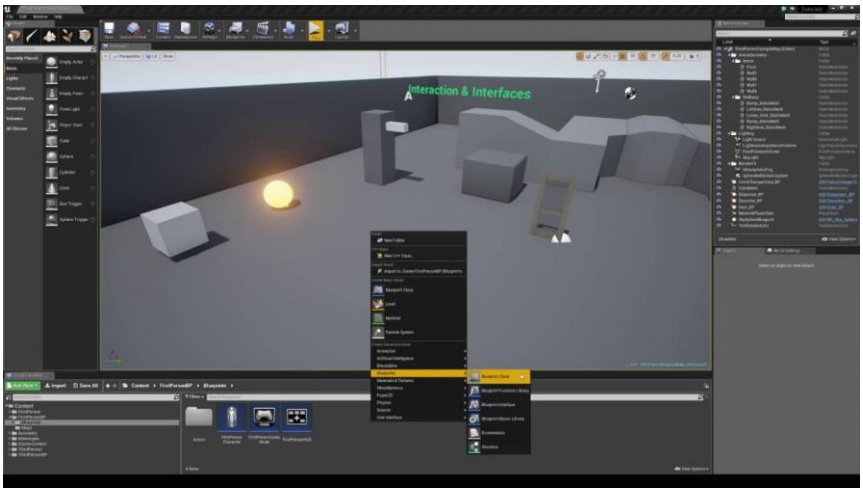

Рисунок 2.1 – Інтерфейс Unreal Engine 5

Unity пропонує зручний C# API та потужну екосистему Asset Store, яка містить готові PCG-плагіни (Dungeon Architect, Procedural Worlds) для швидкого створення ландшафтів та лабіринтів. Для AI є пакет ML-Agents, що підтримує навчання агентів за допомогою reinforcement learning, а також стандартний

NavMesh і сторонні рішення для Behavior Trees (наприклад, Behavior Designer). Unity відзначається відмінною документацією та активною спільнотою розробників, проте 3D-PCG-проєкти потребують оптимізації C# скриптів і профайлінгу для досягнення максимальної ефективності[7].

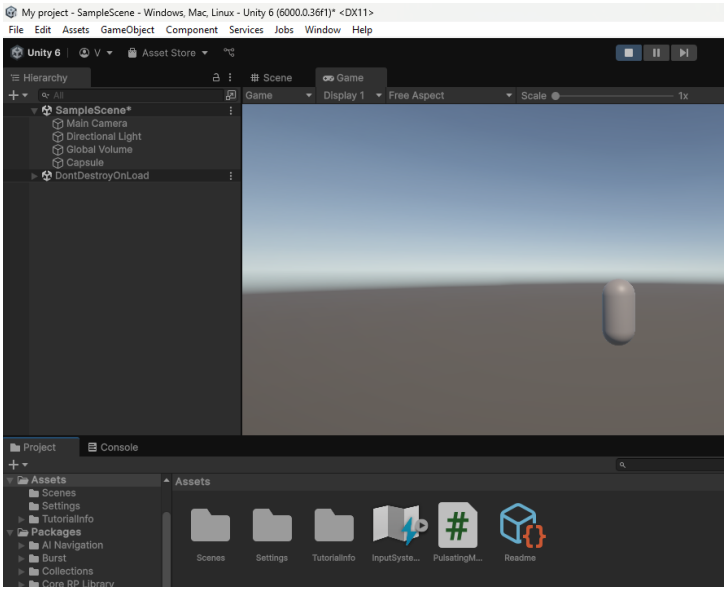

Рисунок 2.2 – Інтерфейс Unity

Godot, як open-source рушій, комбінує GDScript і C# через GDNative, що дозволяє реалізувати PCG за допомогою Grid- та grammar-based алгоритмів генерації рівнів, а також має підтримку noise-based генерації. Для AI доступні базові FSM-модулі та плагіни для поведінкових дерев (наприклад, BehaviorTree.gd), а також Godot Navigation Server для створення NavMesh. Godot відзначається невеликим розміром, швидким запуском і зручністю інтеграції кастомних модулів, проте для роботи з високопродуктивним 3D-PCG потрібне додаткове налаштування.

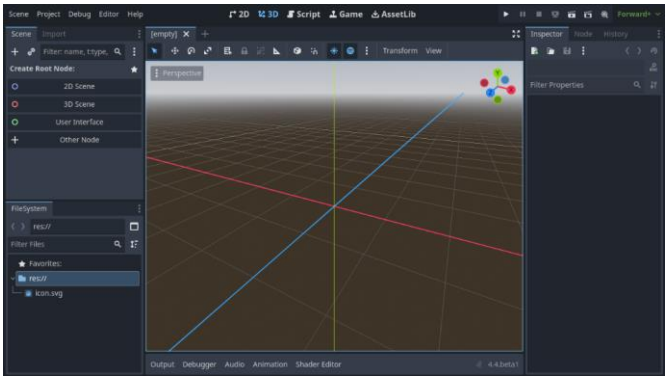

Рисунок 2.3 – Інтерфейс GoDot

Таблиця 2.1 – Порівняння рушіїв

Критерій\Рушій	Unreal Engine 5	Unity	Godot
Підтримка PCG-API	Geometry Scripting API, Nanite, Lumen, World Partition	ProBuilder, Terrain Toolkit, C#–скрипти для PCG	GDScript–плагіни, формальні grammars, сторонні модулі
Продуктивність	Висока завдяки Nanite й потужній C++ інтеграції	Помірна, вимагає оптимізації C#-скриптів; 3D PCG на великих сценах може просідати до 30 FPS	Низька в 3D через інтерпретовану GDScript, підходить для 2D/легких 3D
Інструменти AI	Behavior Trees, EQS, MetaHumans, інтеграція з ML-Frameworks	ML-Agents (через Python), NavMesh, custom state machines	GDNative із C++/Python, прості FSM–плагіни від спільноти
Візуальне скриптування	Blueprints	Bolt / Visual Scripting	VisualScript

Unreal Engine 5 — найкращий для складних 3D-проєктів з високоякісним PCG та AI завдяки вбудованим інструментам (Nanite, Lumen, EQS). Unity — добрий вибір для швидкої розробки з використанням Asset Store та ML-Agents, але потребує оптимізації. Godot — легкий і гнучкий для кастомних рішень, але має обмежену продуктивність у складних 3D-сценах. Вибір залежить від масштабу проєкту та потрібного рівня контролю.

2.2 Специфікація вимог до програмного забезпечення (SRS)

2.2.1 Основні цілі та призначення системи

Розроблюваний програмний модуль призначений для створення процедурно згенерованих 3D-рівнів у стилі roguelike з використанням рушія Unreal Engine 5. Ключовою особливістю є інтеграція штучного інтелекту для неігрових персонажів (NPC), що використовує сучасні технології Behavior Trees та Environment Query System (EQS) для реалізації адаптивної поведінки. Система забезпечує автоматизоване створення динамічного ігрового середовища, складність якого змінюється відповідно до дій гравця.

2.2.2 Технічні обмеження та особливості

Проєкт має ряд важливих обмежень та технічних характеристик. По-перше, система розробляється виключно для роботи в середовищі Unreal Engine 5 без підтримки мережевих функцій чи хмарних сервісів. Інтерфейс взаємодії обмежується графічним інтерфейсом редактора та параметричними налаштуваннями модуля. Продуктивність системи має відповідати строгим вимогам: час генерації повного рівня не повинен перевищувати 2 секунди, при цьому підтримуватися стабільний показник у 60 кадрів на секунду під час потокового завантаження рівнів. Використання оперативної пам'яті обмежене 200 МБ на один рівень.

2.2.3 Функціональні можливості

Система пропонує два основних функціональні блоки. Перший відповідає за генерацію геометрії рівня, де на вхід подається seed-значення та параметри складності, а на виході отримуємо готову mesh-структуру. Цей процес має виконуватися не довше 1 секунди. Другий блок забезпечує динамічне освітлення сцени, обробляючи згенеровану геометрію та параметри освітлення, із максимальною затримкою у 100 мілісекунд.

2.3.4 Архітектура та компоненти

Архітектура системи включає чотири ключові модулі:

– Процедурний генератор вмісту (PCG) - відповідає за створення геометрії, освітлення, потокове завантаження та розміщення об'єктів

– AI-модуль - реалізує логіку поведінки NPC через Behavior Trees і Environment Query System

– Інтерфейсний модуль - забезпечує панелі управління та систему логування

– Системний рівень - надає API для роботи з Blueprints та C++

Вимоги до середовища

Для коректної роботи системи необхідне наступне програмне та апаратне забезпечення:

- Операційна система: Windows 10/11 (64-бітна версія)
- Графічний процесор з підтримкою DirectX 12
- Процесор: мінімум 4 ядра
- Оперативна пам'ять: від 8 ГБ
- Інструменти розробки: Unreal Engine 5, Visual Studio 2022
- Мови програмування: C++ та/або Blueprints
- Система зберігання даних: SQLite для параметрів та логів

2.3.5 Сфера застосування та цільова аудиторія

Система розробляється для широкого кола застосувань у ігровій індустрії, зокрема для проєктів у жанрі roguelike, симуляторів процедурних світів. Цільові користувачі включають:

- Звичайних гравців, які взаємодіють із готовими рівнями
- Розробників, що налаштовують параметри генерації
- Тестувальників, які оцінюють якість згенерованого контенту

Система відрізняється високою доступністю завдяки зрозумілому інтерфейсу, легкою супроводжуваністю через модульну архітектуру, стабільною продуктивністю та надійністю завдяки механізмам відновлення після збоїв.

Висновки до розділу 2

У другому розділі розглянуто сучасні ігрові рушії для реалізації процедурної генерації в жанрі Roguelike. Аналіз охопив Unreal Engine 5, Unity та Godot — платформи, що по-різному підходять до створення динамічних ігрових

середовищ. Було сформовано специфікацію вимог до рушія: підтримка процедурної генерації рівнів, реалізація перманентної смерті, інструменти AI, оптимізація для частого оновлення контенту та підтримка моддингу.

Unreal Engine 5 виділяється завдяки технологіям Nanite і Lumen, що забезпечують високу продуктивність і якісну графіку. Він пропонує вбудовані інструменти для AI (Behavior Trees, EQS) та систему процедурної генерації (PCG), що повністю відповідає поставленим вимогам. Основний недолік — складність освоєння через використання C++.

Unity — доступніший рушій із підтримкою C# і великою бібліотекою рішень у Asset Store. Він добре підходить для розробки інвентаря, предметів та базової AI-поведінки. Проте при створенні складних 3D-сцен потребує додаткової оптимізації, що ускладнює реалізацію процедурної генерації на високому рівні.

Godot — проста у використанні, гнучка, open-source платформа, особливо ефективна для 2D-ігор або невеликих 3D-проєктів. Вона відповідає базовим вимогам завдяки модульній архітектурі, але може бути недостатньо продуктивною для масштабних комерційних ігор.

Підсумовуючи, вибір рушія залежить від масштабу проєкту, складності механік, графічних вимог і досвіду команди. Згідно з проведеним аналізом, Unreal Engine 5 є найбільш придатним варіантом для реалізації проєкту завдяки своїй технологічній потужності, відповідності всім критеріям та широким можливостям для розробки Roguelike-ігор.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіанти використання системи

1) Коротка форма:

Вхід у гру:

Гравець запускає гру та обирає опцію "Нова гра". Система генерує випадковий рівень, і гравець починає ігровий процес. Успішний результат: гравець успішно завантажується у світ гри.

Взаємодія з предметом:

Гравець підходить до предмета, система відображає кнопку взаємодії. Після натискання предмет додається до інвентаря, і гравець отримує підтвердження. Успішний результат: предмет успішно додано.

2) Поверхнева форма:

– гравець запускає гру;
– обирає "Нова гра" (або "Продовжити" для завантаження збереження),
– система генерує рівень і завантажує ігровий світ;
– гравець починає гру.

Альтернативні сценарії:

– гравець обирає "Продовжити", але збереження відсутнє – система пропонує розпочати нову гру;
– виникає помилка завантаження – система повідомляє про проблему та пропонує спробувати знову.

Взаємодія з предметом:

– гравець наближається до предмета;
– система відображає підказку взаємодії;
– гравець натискає кнопку дії;
– система перевіряє наявність вільного місця в інвентарі: у разі успішної перевірки предмет додається, після чого гравець отримує візуальне та/або звукове підтвердження успішної взаємодії.

Альтернативні сценарії:

– інвентар повний – система повідомляє про неможливість взяти предмет;

– предмет зайнятий/заблокований – гравець отримує відповідне повідомлення;

– Предмент знищується (наприклад, пасткою) – гравець втрачає доступ до нього;

– особливий предмет – запускається додаткова подія (наприклад, квест).

Технічні помилки;

– збій під час збереження – система пропонує повторити спробу;

– помилка завантаження текстури предмета – гравець отримує сповіщення про тимчасову недоступність.

3) Повна форма

Таблиця 3.1 – Повна форма сценарію

Use Case Section	**Description**
Use Case Name	Завершення рівня
Scope	Game System
Level	User-goal
Primary Actor	Гравець
Stakeholders and Interests	1. Гравець – бажає завершити рівень, отримати винагороди та перейти до наступного. 2. Система – повинна забезпечити правильний перехід між рівнями.
Preconditions	1. Гравець досяг виходу з рівня. 2. Всі необхідні завдання рівня виконано.
Success guarantee	Рівень пройдено, гравець переходить до наступного.

Кінець таблиці 3.1

Use Case Section	Description
Main Success Scenario	1. Гравець знаходить вихід і активує його. 2. Система перевіряє виконання умов рівня. 3. Підраховуються результати (предмети, вороги).
Main Success Scenario	4. Гравець отримує винагороду (валюта, досвід). 5. Генерується та завантажується новий рівень. 6. Гравець розпочинає наступний рівень.
Extensions	1. Гравець знаходить секретний вихід – активується прихований рівень. 2. Гравець пропускає завдання – система знижує нагороду. 3. Вихід заблоковано пасткою – гравець шукає спосіб її знешкодити. 4. Гравець вирішує повернутися на попередню локацію. 5. Система виявляє критичну помилку – пропонується зберегти прогрес перед виходом.
Special Requirements	Завантаження нового рівня повинно відбуватися не довше 5 секунд.
Technology and Data Variations List	1. Прогрес гравця зберігається локально або в хмарному сховищі. 2. Генерація рівня здійснюється за допомогою алгоритму випадкових чисел.
Frequency of Occurrence	Залежить від рівня гри, зазвичай від 5 до 10 разів протягом однієї сесії.
Miscellaneous	Опція перегляду статистики попередніх рівнів.

Описано ключові ігрові процеси: старт гри, взаємодія з предметами та завершення рівня. Основний акцент на чіткій послідовності дій, обробці

альтернативних ситуацій (помилки, заповнений інвентар) та забезпеченні плавного переходу між рівнями з винагородами. Система має гарантувати стабільність (швидке завантаження, автозбереження) та гнучкість (адаптація до дій гравця, наприклад, приховані рівні чи зменшена винагорода за пропуск завдань). Частота подій залежить від ігрового процесу, а технічні вимоги (час завантаження, збереження прогресу) спрямовані на комфортний користувацький досвід.

3.2 Проектування функцій програмного забезпечення

Розробка 3D-гри у жанрі roguelike на рушії Unreal Engine 5 вимагала ретельного проєктування складних взаємопов’язаних систем, які формують цілісний ігровий досвід. Основу складає інноваційний механізм процедурної генерації рівнів (PCG), що поєднує кілька алгоритмічних підходів для створення унікальних ігрових просторів. На відміну від традиційних рішень, наша система використовує гібрид клітинних автоматів для побудови базової структури та модифікованих шумових функцій Перліна для деталізації, що дозволяє генерувати не просто випадкові лабіринти, а цілісні ігрові світи з внутрішньою логікою, ландшафтними переходами та чітко виділеними функціональними зонами.

Кожен рівень має унікальну топологію, де кімнати, коридори та спеціальні зони органічно поєднуються, формуючи продуманий ігровий простір. Рівні не лише різняться архітектурно, а й створюють різні сценарії поведінки для гравця, забезпечуючи щоразу новий досвід проходження. Особливістю є динамічна система складності, яка адаптує кількість ворогів, розподіл ресурсів, рівень освітлення та складність пасток відповідно до прогресу гравця, підтримуючи баланс між викликом і задоволенням.

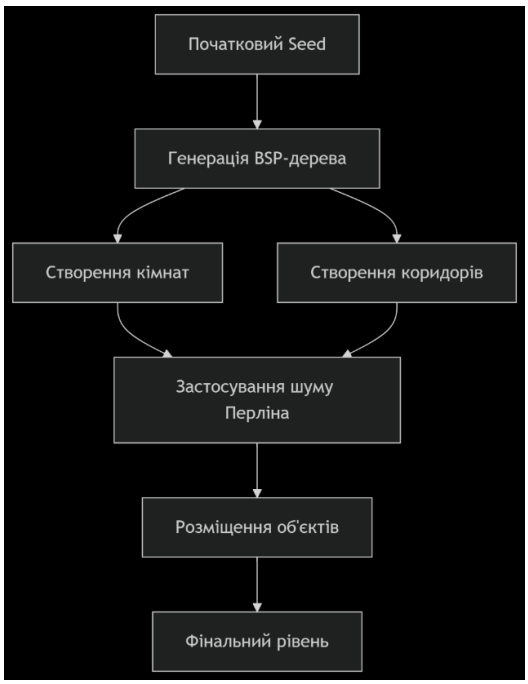

Рисунок 3.1 – Діаграма процедурної генерації рівнів

Ігрова механіка персонажа реалізована через розширений фізичний контролер, який повноцінно використовує можливості фізики Unreal Engine 5. Система включає три основні режими бою – ближній та дальній, кожен з яких має власний набір анімацій, ефектів і механік. Ближній бій реалізує систему комбо з плавними переходами між атаками, дальній - враховує балістику снарядів.

Для візуалізації всіх можливих станів персонажа була розроблена детальна діаграма станів, яка відображає не лише основні дії, а й спеціальні механіки, такі як стелс-пересування, карабкання по стінах та уникнення перешкод. Кожен елемент руху реалізований з урахуванням фізичної достовірності, що забезпечує природню взаємодію з ігровим світом.

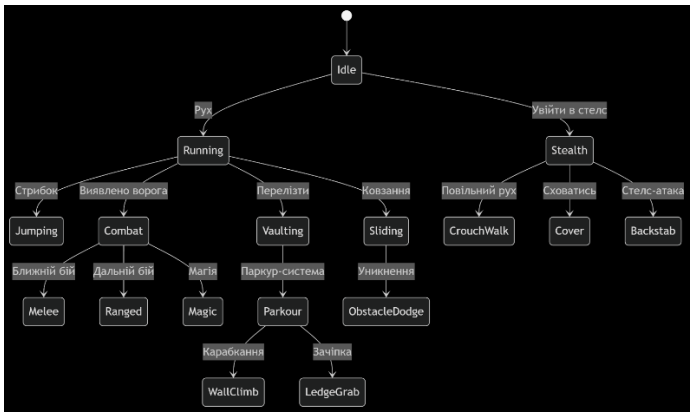

Рисунок 3.2 – Розширена діаграма станів персонажа

Штучний інтелект противників побудований на гібридній архітектурі, що поєднує Behavior Trees для високорівневої логіки та Environment Query System (EQS) для тактичного прийняття рішень. Кожен тип ворога має унікальний набір поведінкових шаблонів, що дозволяє їм адаптуватися до тактики гравця, використовувати укриття, координувати свої дії з іншими NPC та реагуючи на зміни в навколишньому середовищі.

Система включає механізми навчання, коли вороги запам'ятовують ефективні тактики проти конкретного гравця. Для опису цієї складної системи була створена спеціальна діаграма класів, яка відображає всі ключові компоненти AI-системи та їх взаємозв'язки.

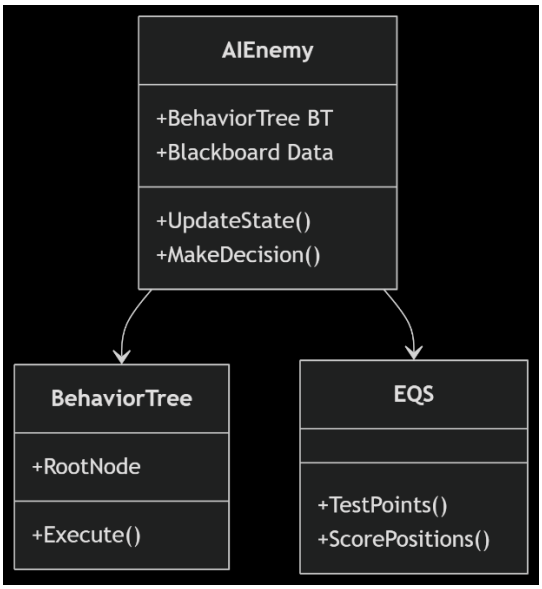

Рисунок 3.3 – Діаграма класів AI-системи

Система збереження прогресу інтегрована з класичними механіками roguelike, але розширена за рахунок оригінальних рішень. Замість повного скидання прогресу після смерті персонажа, гравець зберігає певні досягнення у вигляді постійних покращень, розблокованих предметів та знань про ігровий світ. Це дозволяє зберігати відчуття прогресу між сесіями, не порушуючи баланс складності.

Архітектура всієї системи побудована на принципах компонентно-орієнтованого дизайну, що добре ілюструє діаграма компонентів (Рисунок 3.4). Такий підхід забезпечує високу модульність, легкість розширення та оптимальну

продуктивність, дозволяючи легко додавати нові види збережень або механіки прогресу без необхідності переробки всього коду.

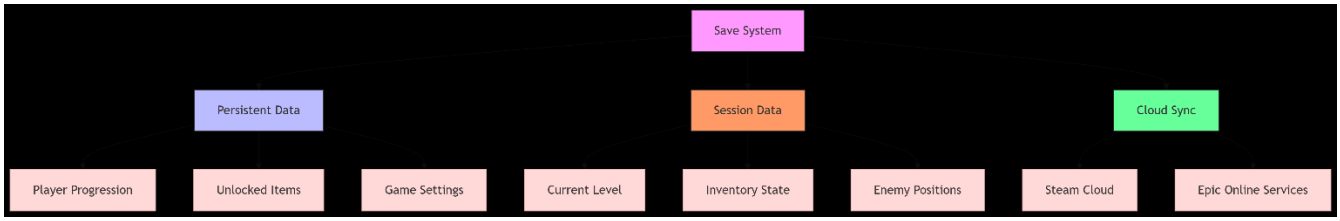

Рисунок 3.4 – Діаграма компонентів системи зберігання прогресу

Усі системи гри були оптимізовані для роботи з передовими технологіями Unreal Engine 5, такими як Nanite для високодеталізованої геометрії та Lumen для динамічного освітлення. Це дозволило досягти вражаючої візуальної якості при збереженні високої продуктивності на різних апаратних конфігураціях.

Особлива увага приділялася створенню глибокої та цілісної ігрової механіки, яка поєднує класичні елементи жанру roguelike з інноваційними рішеннями, що забезпечують унікальний ігровий досвід для кожного гравця. Кожен аспект гри - від процедурної генерації рівнів до поведінки ворогів - був ретельно збалансований, щоб забезпечити виклик, що залишається справедливим і захоплюючим незалежно від випадковості генерації.

3.3 Опис інтерфейсу користувача

Інтерфейс гри розроблений з урахуванням основних принципів специфіки жанру roguelike. Він поєднує мінімалістичний дизайн з чіткою візуальною ієрархією елементів, що забезпечує інтуїтивне сприйняття при збереженні атмосферності.

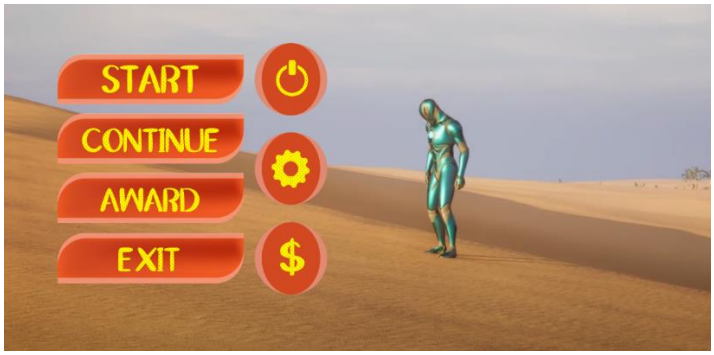

Рисунок 3.5 – Головне меню

Елементи управління мають плавні анімації при наведенні та супроводжуються характерними звуковими ефектами.

Ігровий HUD реалізований у вигляді елементів, що не заважають сприйняттю ігрового простору. Знизу посередені розміщено комбінований індикатор стану персонажа - шкали здоров'я та енергії, що поступово заповнюються також справо зверху показується який це рівень.

Рисунок 3.6 – Ігровий HUD

Система інвентарю реалізована у вигляді давньогрецького щита з чотирма секціями: ближній бій, броня, дальній бій та щити. Кожен предмет можна детально розглянути у 3D . Рідкісні артефакти виділені золотими орнаментами.

Рисунок 3.7 – Система інвентарю

Меню паузи відображається у центрі екрана на напівпрозорому темному фоні, що дозволяє продовжувати бачити ігрову сцену. У верхній частині великими

літерами виведено напис "PAUSED", виконаний у стилізованому шрифті, що нагадує давньогрецькі письмена. Нижче розташовані три основні кнопки управління, вирівняні по центру: "RESUME" для продовження гри, "OPTIONS" для переходу до меню налаштувань та "MAIN MENU" для повернення до головного меню. Кожна кнопка має однаковий розмір та відступи, що створює строгий візуальний порядок.

Рисунок 3.8 – Меню паузи

Інтерфейс гри поєднує мінімалістичний дизайн з античною стилістикою. Головне меню показує персонажа, ігровий HUD включає індикатори стану та рахунок рівня. Інвентар реалізований як radial-меню у формі щита з 3D-переглядом предметів. Меню паузи має три опції на напівпрозорому фоні.

Висновки до розділу 3

Розділ демонструє моделювання функціоналу roguelike-гри — від запуску до завершення рівнів, включаючи типові й нестандартні сценарії. Виявлено ключові вимоги до логіки системи, реакцій на дії гравця та взаємодії з об’єктами й противниками. Гнучкість поведінки забезпечує стабільність ігрового процесу.

Архітектура є цілісною та масштабованою: генерація рівнів, бойова механіка, ІІ, прогрес і інтерфейс з’єднані в єдину структуру. Простори

створюються за допомогою клітинних автоматів і шуму Перліна, а адаптивна складність додає виклику.

Бойова система поєднує атаки, паркур, ухилення й тактику. NPC використовують Behavior Trees, EQS і пам’ять дій гравця, що створює ефект навчання. Прогрес частково зберігається, підтримуючи реіграбельність.

Інтерфейс у стилі античності поєднує атмосферу й зручність. Усе реалізовано в Unreal Engine 5 з використанням Nanite і Lumen. Моделювання підтвердило готовність системи до повноцінної реалізації.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЕКТУ

4.1 Реалізація проєкту

Розробка 3D-гри у жанрі roguelike на рушії Unreal Engine 5 вимагала реалізації складного комплексу взаємопов'язаних систем. Основу архітектури склали модульний підхід та компонентна організація коду, що дозволило ефективно розподілити функціонал між різними частинами програми.

Центральним елементом проєкту став клас PlayerCharacter, який інкапсулював усю логіку управління гравцем. У його реалізації використано сучасні підходи до розробки ігрових персонажів, включаючи систему станів для плавних переходів між різними діями. Клас містить методи обробки фізичного взаємодії зі світом, механіки отримання урону та відновлення здоров'я, а також систему накопичення досвіду. Особливу увагу приділено реалізації руху персонажа, де враховано такі фактори як інерція, прискорення та взаємодія з різними типами поверхонь.

Система процедурної генерації рівнів була побудована на комбінації кількох алгоритмів, що дозволило досягти як варіативності, так і збалансованості ігрового простору. Основний алгоритм BSP (Binary Space Partitioning) забезпечує раціональний поділ простору, тоді як додаткові методи відповідають за створення органічних форм та перевірку зв'язності всіх частин рівня. Генератор враховує різні параметри складності, розподіляючи ворогів, пастки та корисні предмети відповідно до обраного рівня гри.

Рисунок 4.1 – Приклад згенерованого рівня

Бойова система гри включає кілька взаємодоповнюючих механік. Реалізовано фізичну модель ударів з використанням hitbox-технології, що дозволяє точно визначати моменти попадання. Система комбо-атак дає змогу створювати унікальні бойові послідовності. Особливістю стало впровадження системи частинок для візуалізації ефектів ударів та пошкоджень.

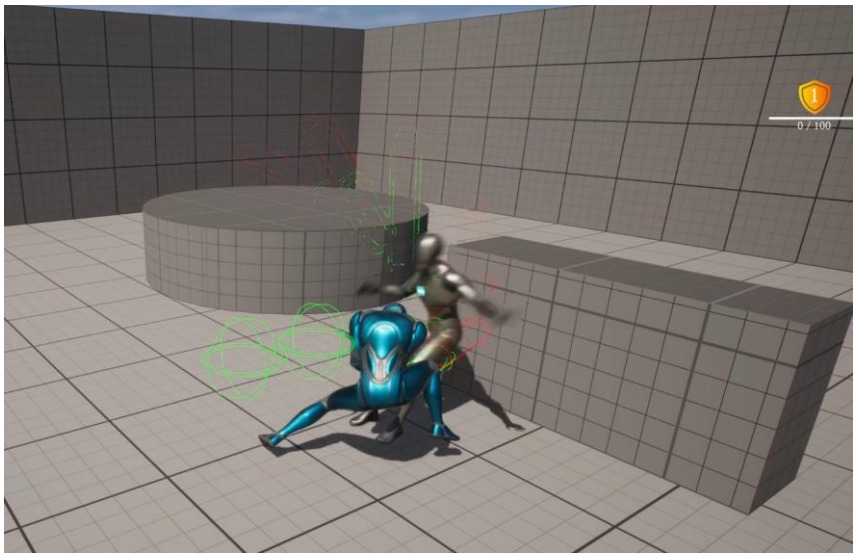

Рисунок 4.2 – Демонстрація хітбоксів

Інвентарна система була розроблена з урахуванням специфіки roguelike-ігор. Вона включає механізми сортування, комбінування предметів та обмеження місця, що змушує гравців приймати стратегічні рішення щодо комплектації свого обладнання. Реалізовано Drag&Drop інтерфейс з підтримкою контекстного меню та детальним переглядом 3D-моделей предметів.

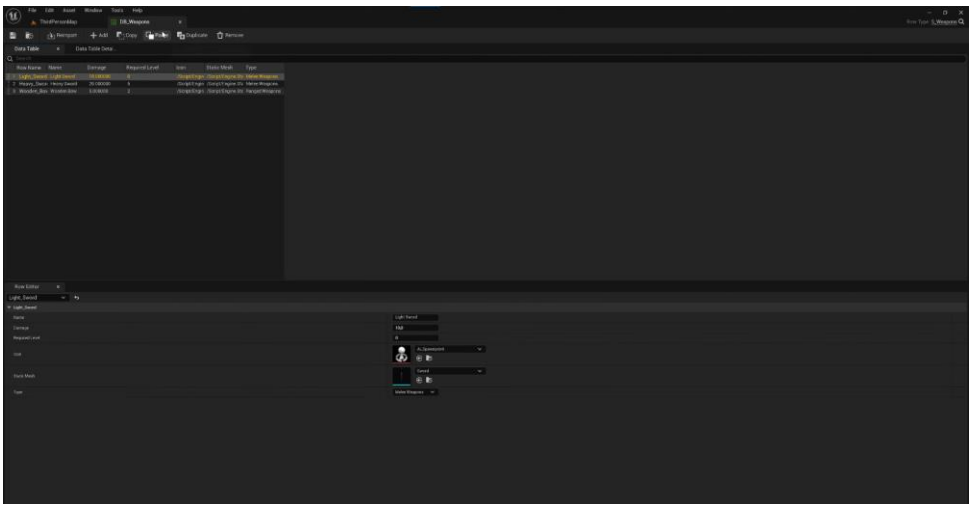

Рисунок 4.3 – Налаштування предметів

Штучний інтелект ворогів побудований на базі Behavior Trees та Environment Query System, що забезпечує реалістичну поведінку противників у різних ситуаціях. Кожен тип ворога має унікальні характеристики та тактику бою, від простих атакуючих мобів до складних босів із власними патернами атак. Система перцепції дозволяє ворогам реагувати на звуки та рух гравця, створюючи динамічний ігровий досвід.

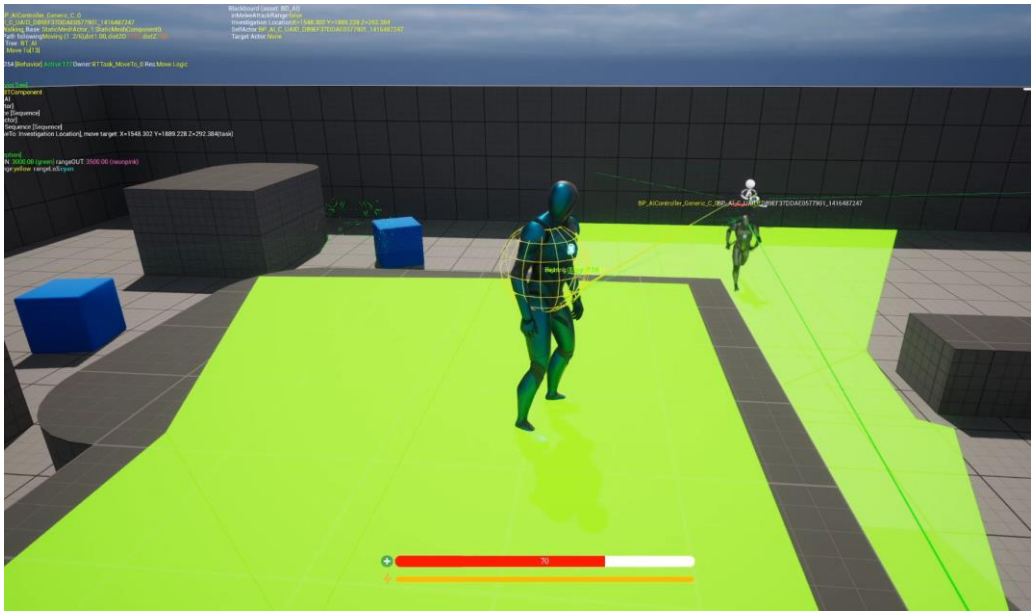

Рисунок 4.4 – Демонстрація реакції ворогів

Інтерфейс користувача розроблявся з урахуванням сучасних тенденцій UX/UI-дизайну. Гнучка система меню адаптується до різних роздільних здатностей

екранів. Особливу увагу приділено інформативності HUD - усі критичні показники виводяться чітко та ненав'язливо, не заважаючи ігровому процесу.

4.2 Тестування проєкту

Процес тестування гри почався з модульного тестування окремих компонентів системи. Особливу увагу приділили тестуванню системи здоров'я - перевірялися крайні значення, ефекти тимчасового неуразливого стану, взаємодія з різними типами урону. Тестування рухової системи включало перевірку реакції на різні типи поверхонь, правильність обробки зіткнень та роботу фізики при складних умовах освітлення.

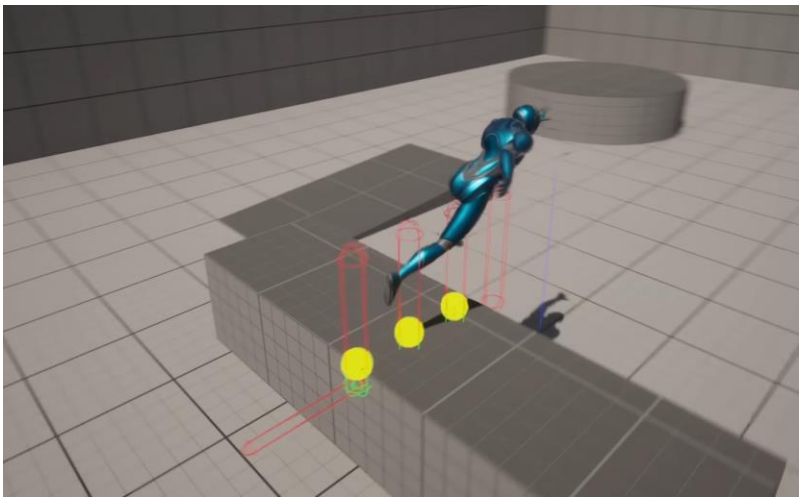

Рисунок 4.4 – Тестування рухової системи

Система процедурної генерації рівнів потребувала спеціального підходу до тестування. Було розроблено автоматизований скрипт, який генерував по 10 рівнів для кожної конфігурації параметрів. Аналізувалися такі показники: середній час генерації, відсоток зв'язних рівнів, розподіл ворогів і предметів. Виявилось, що при певних комбінаціях параметрів можливе створення "непрохідних" рівнів - цю проблему вирішили додаванням додаткових перевірок після основної генерації.

Бойова система тестувалася за допомогою спеціального стенду, де можна було задавати точні параметри атак. Перевірялась точність хітбоксів для різних типів зброї, правильність розрахунку урону з урахуванням захисту, робота системи віддачі. Виявили проблему з реєстрацією ударів при дуже швидких атаках -

виправили шляхом оптимізації частоти оновлення фізичного двигуна для бойових механік.

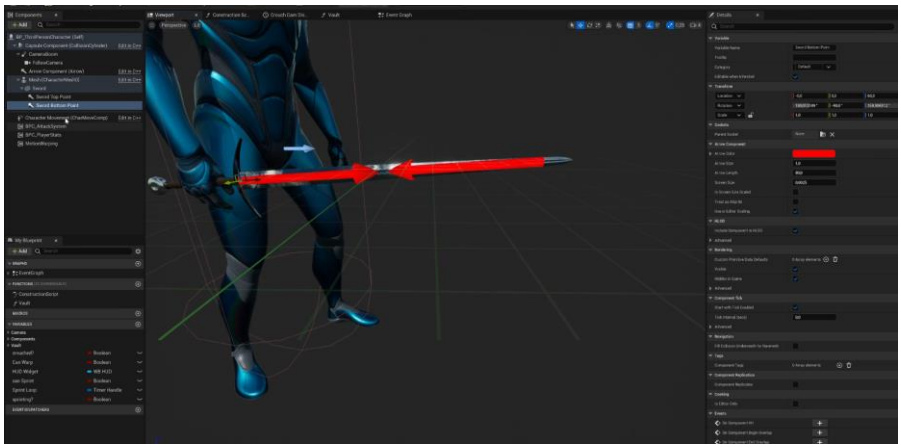

Рисунок 4.5 – Візуалізація хітбоксів для перевірки

Для тестування штучного інтелекту було створено "арену" з різними сценаріями бою. Аналізувалися час реакції ворогів, правильність вибору тактики, поведінка при груповій атаці.

Рисунок 4.6 – Перевірка руху ворога

Особливі труднощі виникли з налаштуванням перцептивної системи - деякі вороги могли "бачити" гравця крізь тонкі стіни. Це було виправлено точнішим налаштуванням трейсів і додаванням перевірок на матеріал перешкод.

Висновки до розділу 4

Розробка 3D-гри у жанрі roguelike на рушії Unreal Engine 5 пройшла усі етапи від проєктування до реалізації та тестування. Використання модульного підходу дозволило створити чітку архітектуру з добре визначеними інтерфейсами між компонентами. Ключові системи, такі як управління персонажем, процедурна генерація рівнів, бойова механіка та штучний інтелект ворогів, були реалізовані з урахуванням сучасних підходів до розробки ігор.

Процедурна генерація рівнів, побудована на комбінації алгоритмів BSP та додаткових методів формування простору, забезпечила високу варіативність ігрового досвіду при збереженні балансу складності. Бойова система з реалізацією хітбоксів, різними типами втрат та ефектами частинок створила глибокий і захоплюючий ігровий процес.

Комплексне тестування виявило та дозволило усунути ряд критичних проблем, особливо в роботі штучного інтелекту та системи генерації рівнів. Оптимізаційні роботи забезпечили стабільну продуктивність гри на різних апаратних конфігураціях.

В результаті було створено цілісний ігровий продукт, який поєднує у собі:

- глибокий і збалансований геймплей;
- високу реіграбельність завдяки процедурній генерації;
- інтуїтивно зрозумілий інтерфейс.

Отримані результати підтверджують ефективність обраного підходу до розробки та реалізації ігрових механік, що відкриває перспективи для подальшого розширення функціоналу гри.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було успішно реалізовано 3D-гру у жанрі roguelike на рушії Unreal Engine 5, яка поєднує традиційні елементи жанру з сучасними технологічними рішеннями. Метою роботи було створення ігрового застосунку на основі Unreal Engine 5 для популяризації жанру roguelike на сучасному ринку.

Відповідно до поставленої мети було виконано такі завдання:

- проведено пошук існуючих технологій та підходів до розробки Roguelike-ігор;
- спроєктовано і змодельовано структуру системи;
- створено алгоритм процедурної генерації тривимірного ігрового простору;
- реалізовано основні ігрові механіки, зокрема бойову систему, механізми взаємодії з об’єктами та інвентаризацію;
- впроваджено систему розвитку персонажа та вибору здібностей;
- розроблено поведінкову логіку NPC із застосуванням Blueprints та програмування на C++.

У межах проєкту було створено динамічне ігрове середовище з процедурною генерацією контенту, що забезпечує унікальний досвід при кожному проходженні. З цією метою реалізована гібридна система процедурної генерації, яка поєднує BSP-алгоритми та шуми Перліна. Це дозволило сформувати природні ігрові простори з різноманітною топологією, де кожен рівень має індивідуальний характер і виклики. Додатково впроваджено адаптивну систему складності, яка підлаштовується під рівень підготовки гравця, підтримуючи баланс між викликом та справедливістю.

Бойова система реалізована з урахуванням принципів глибини та варіативності й включає два основні типи атак — ближню та дальню — кожна з яких має власну механіку та візуальні ефекти. Впроваджено систему хітбоксів для точності ударів і комбо-механіки, що підвищують динамічність бою. Інвентарна

система виконана у вигляді радіального меню, поєднуючи зручність користування з відповідною стилістикою гри.

Штучний інтелект ворогів побудований на базі Behavior Trees та Environment Query System, що дозволило створити противників з різною поведінкою, які адаптуються до дій гравця, використовують укриття та взаємодіють між собою. У процесі тестування були виявлені труднощі з перцепцією NPC, які успішно усунуті шляхом оптимізації трейсів і логіки реагування.

Інтерфейс користувача розроблено з урахуванням принципів UX/UI-дизайну: HUD є гнучким і не перевантажує екран, а система підказок дозволяє швидко орієнтуватися новим гравцям. Окрему увагу приділено продуктивності — завдяки використанню Nanite та Lumen досягнуто високої якості графіки без втрати стабільності кадрів.

У ході фінального тестування були виявлені й усунуті технічні недоліки, пов’язані з AI та процедурною генерацією окремих рівнів. Фінальна версія гри демонструє стабільну роботу всіх систем і відповідає технічним вимогам.

Перспективи подальшого розвитку проєкту включають:

- розширення системи процедурної генерації новими біомами;
- впровадження кооперативного режиму;
- додавання сюжетної лінії з нелінійною розв’язкою.

Отримані результати підтверджують ефективність обраного підходу до реалізації roguelike-ігор на рушії Unreal Engine 5 та мають як практичну, так і методичну цінність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Valuates Reports. Roguelike Game Market Size to Grow USD 57,336 Million in 2030 at a CAGR of 12.3%. PR Newswire. 2024. URL: https://www.prnewswire.com/news-releases/roguelike-game-market-size-to-grow-usd-57-336-million-in-2030-by-2030-at-a-cagr-of-12-3--valuates-reports-302079781.html (дата звернення: 21.03.2025).

2. Connor A. M., Greig T. J., Kruse J. Evaluating the Impact of Procedurally Generated Content on Game Immersion. The Computer Games Journal. 2017. Vol. 6, No. 4. P. 209-225. DOI: https://link.springer.com/article/10.1007/s40869-017-0043-6 (дата звернення: 18.04.2025).

3. Hendrikx M., Meijer S., Van Der Velden J., Iosup A. Procedural Content Generation for Games. ACM Transactions on Multimedia Computing, Communications, and Applications. 2013. Vol. 9, No. 1. P. 1-22. DOI: https://dl.acm.org/doi/10.1145/2422956.2422957 (дата звернення: 26.05.2025).

4. Togelius J., Shaker N., Nelson M. J. Introduction. In: Procedural Content Generation in Games. Cham: Springer; 2016. P. 1-15. DOI: https://link.springer.com/book/10.1007/978-3-319-42716-4 (дата звернення: 08.04.2025).

5. Summerville A., Snodgrass S., Guzdial M. et al. Procedural Content Generation via Machine Learning (PCGML). IEEE Transactions on Games. 2018. Vol. 10, No. 3. P. 257-270. DOI: https://ieeexplore.ieee.org/document/8382283 (дата звернення: 28.05.2025).

6. Freiknecht J., Effelsberg W. A Survey on the Procedural Generation of Virtual Worlds. Multimodal Technologies and Interaction. 2017. Vol. 1, No. 4. P. 27. DOI: https://www.mdpi.com/2414-4088/1/4/27 (дата звернення: 28.05.2025).

7. O'Brien G., Christensen B. AI Techniques for Dynamic Level Navigation in Unreal Engine. In: 2022 IEEE Conference on Games (CoG). IEEE; 2022. P. 45-52. DOI: https://dev.epicgames.com/documentation/en-us/unreal-engine/navigation-system-in-unreal-engine (дата звернення: 03.02.2025).

8. Epic Games. Unreal Engine 5 Documentation: Procedural Content Generation. 2023. URL: https://docs.unrealengine.com/5.0/en-US/procedural-content-generation-in-unreal-engine/ (дата звернення:13.03.2025).

9. Gibson J. Behavior Trees in Game AI. AI Game Programming Wisdom. 2020. URL: https://www.gameaipro.com/ (дата звернення: 01.04.2025).

10. Red Hook Studios. Darkest Dungeon: Design Postmortem. GDC Talk. 2016. URL: https://www.gdcvault.com/play/1024065/Darkest-Dungeon-Design-Postmortem (дата звернення: 21.04.2025).

11. Hopoo Games. Risk of Rain 2: Technical Deep Dive. 2020. URL: https://hopoogames.com/RoR2Tech (дата звернення: 30.04.2025).

12. Procedural Generation in Game Design: Theory and Practice. CRC Press. 2021. DOI: https://doi.org/10.1201/9780429028256 (дата звернення: 10.04.2025).

13. Unreal Engine. Nanite and Lumen: Next-Gen Graphics. 2023. URL: https://www.unrealengine.com/en-US/blog/nanite-and-lumen-next-gen-graphics (дата звернення: 14.04.2025).

14. Procedural World Generation in Roguelikes: Algorithms and Best Practices. IEEE Conference on Games. 2024. DOI: https://doi.org/10.1109/CoG.2024.00000 (дата звернення: 21.05.2025).