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

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Ігровий десктоп-застосунок в жанрі adventure на основі рушія Unreal Engine 5

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Дмитро МУТІДЗЕ**

«__» _____ 20__ р.

Керівник роботи

PhD, Доцентка
(б.в.з.) _____ **Катерина АНТІПОВА**

«__» _____ 20__ р.

м. Миколаїв – 2026 рік

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
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Мутідзе Дмитра

1. Тема кваліфікаційної роботи «Ігровий десктоп-застосунок в жанрі adventure на основі рушія Unreal Engine 5» затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від «26» грудня 2025 р.
2. Строк представлення кваліфікаційної роботи «___» червня 2026 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Розроблений десктоп-застосунок в жанрі adventure

4. Перелік питань, що підлягають розробці:
   - аналіз предметної області;
   - проєктування архітектури застосунку;

– моделювання та конструювання ПЗ;
– розробка інтерфейсу користувача;
– інтеграція аудіовізуальних ефектів;
– оптимізація продуктивності;
– тестування та валідація.

5. Перелік графічних матеріалів: презентація
6. Консультанти:

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

Дата видачі завдання « 8 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Ігровий десктоп-застосунок в жанрі adventure на основі рушія UE5**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	05.01.2026	08.01.2026	Виконано
2.	Огляд літератури за темою роботи	12.01.2026	18.01.2026	Виконано
3.	Складання календарного плану КБР	20.01.2026	21.01.2026	Виконано
4.	Аналіз предметної області	23.01.2026	27.01.2026	Виконано
5.	Розробка проєктних рішень	02.02.2026	20.02.2026	Виконано
6.	Моделювання та конструювання ПЗ	25.02.2026	28.03.2026	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	01.04.2026	10.05.2026	Виконано
8.	Відгук керівника КБР	22.05.2026	03.06.2026	Виконано
9.	Оформлення КБР та презентації	16.05.2026	24.05.2026	Виконано
10.	Попередній захист	25.05.2026	25.05.2026	Виконано
11.	Рецензування	12.06.2026	12.06.2026	Виконано
12.	Завершення оформлення КБР та презентації	26.05.2026	11.06.2026	Виконано
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Дмитро МУТІДЗЕ**

«__»_____ 2026 р.

Керівник роботи

PhD, доцентка

(б.в.з.) _____ **Катерина АНТІПОВА**

«__» _____ 2026 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Ігровий десктоп-застосунок в жанрі adventure

на основі рушія Unreal Engine 5»

Здобувач 408 гр.: Дмитро Мутідзе

Керівник: PhD, доцент (б.в.з.) Катерина Антіпова

Сучасна індустрія відеоігор стрімко розвивається, пропонуючи гравцям все більш імерсивні та технологічно складні світи. Жанр adventure залишається популярним завдяки можливостям глибокого сюжету, дослідження світів та емоційного занурення, що робить його ідеальним полем для експериментів. Однак багато існуючих проєктів використовують застарілі рушії, обмежуючи свій візуальний і функціональний потенціал, а доступність сучасних інструментів, таких як Unreal Engine 5 , для навчальних та некомерційних цілей за останні роки починає набирати популярність.

Розробка ігор залишається дуже складним процесом, який вимагає не лише програмування, а й розуміння дизайну, психології гравця та сучасних трендів, а також розуміння мети та поставленої мети. Освіта в галузі програмної інженерії наголошує на важливості практичного досвіду, адже саме досвід дозволяє трансформувати теоретичні знання у реальні навички.

Обрана тема КБР — створення десктоп-застосунку в жанрі adventure на основі рушія UE5 — відображає сучасні тенденції як у геймдеві, так і в освіті. Вона демонструє здатність студента працювати з інструментами високого рівня, а також підкреслює важливість творчого підходу, адже жанр adventure вимагає не лише технічної досконалості, а й художнього бачення. Він поєднує в собі технічну складність, а також творчий потенціал. Це ідеальна тема для експериментів, де можна відточити та продемонструвати навички аналізу проблем, прийняття дизайн-рішень та тестування гіпотез.

Важливим аспектом є й те, що розробка ігор на UE5 активно використовується не лише в ігровій індустрії. Цей рушій застосовується в

архітектурі, у багатьох можливих симуляціях (медицина,дрони), освітніх програмах — це означає, що навички, отримані під час практики, мають широкий спектр застосування.

Об’єктом кваліфікаційної роботи є ігрові механіки та дизайн гри у жанрі adventure.

Предметом кваліфікаційної роботи є інструменти розробки десктоп-гри на основі рушія Unreal Engine 5.

Метою кваліфікаційної роботи є розробка ігрового застосунку на основі рушія Unreal Engine 5 для популяризації ігор у жанрі adventure. Це дозволить:

– оцінити взаємозв’язок між теоретичними концепціями та їх практичною реалізацією;

– розвинути навички роботи з професійними інструментами розробки;

– сформувати системне мислення щодо організації життєвого циклу програмного продукту.

У роботі застосовано **теоретичні методи** дослідження, зокрема аналіз предметної області, порівняння існуючих ігрових рішень, вивчення функціональних можливостей Unreal Engine 5 та методів проєктування інтерактивних систем. **Практичні методи** включали проєктування ігрового середовища, реалізацію програмної логіки за допомогою Blueprints і C++, створення головоломок, системи взаємодії, елементів штучного інтелекту, а також тестування й оптимізацію застосунку.

У **першому розділі** досліджено предметну область, проаналізовано особливості жанру adventure, сучасні тенденції в розробці ігор та виконано огляд аналогів.

У **другому розділі** проведено аналіз інструментарію, методів та вимог. Сформовано вимоги до ігрового застосунку.

У **третьому розділі** наведено процес моделювання програмного продукту на базі Unreal Engine 5, описано структуру проєкту, реалізацію системи керування персонажем, взаємодії з об’єктами, головоломок, динамічного середовища,

аудіовізуальних ефектів і поведінки ігрових сутностей.

У **четвертому розділі** подано результати тестування, оцінку працездатності та продуктивності застосунку, а також визначено напрями подальшого вдосконалення розробленого програмного продукту.

Основним результатом роботи є створений прототип пригодницької гри для персонального комп'ютера, який реалізує дослідження лабіринту, систему головоломок, сюжетно-наративні елементи, атмосферне аудіовізуальне оформлення та скриптові ігрові події. Розроблений застосунок може бути використаний як основа для подальшого розвитку повноцінного ігрового продукту, а також як приклад практичного застосування сучасних технологій розробки інтерактивного програмного забезпечення

КБР містить вступ, два розділи, висновки, перелік джерел посилання. Вступ зазначає актуальність, мету, об'єкт та предмет роботи. Перший розділ описує предметну область та аналоги застосунку.

Ключові слова: adventure, AI, аналіз, застосунок, індексація, особливості, прототип, розробка, рушій, тестування, C++, UE5.

ABSTRACT

of the Bachelor’s Thesis

“A desktop adventure gaming application based on Unreal Engine 5”

Applicant of 408th group: Dmytro Mutidze

Supervisor: PhD, Associate Professor (w/o a.d.) Kateryna Antipova

The modern video game industry is developing rapidly, offering players more and more immersive and technologically complex worlds. The adventure genre remains popular due to the possibilities of deep storytelling, world exploration, and emotional immersion, making it an ideal field for experimentation. However, many existing projects use outdated engines, limiting their visual and functional potential, and the availability of modern tools such as Unreal Engine 5 for educational and non-commercial purposes has been gaining popularity in recent years.

Game development remains a very complex process that requires not only programming, but also an understanding of design, player psychology and current trends, as well as an understanding of the goal and objective. Education in software engineering emphasizes the importance of practical experience, because it is experience that allows you to transform theoretical knowledge into real skills. The qualifying bachelor's thesis teaches you to plan projects, work with limited resources, adapt to changes, and, most importantly, see the product holistically - from idea to implementation.

The chosen topic of the internship - creating a desktop adventure application based on the UE5 engine - reflects current trends in both game development and education. It demonstrates the student's ability to work with high-level tools and emphasizes the importance of creativity, as the adventure genre requires not only technical excellence but also artistic vision.

Another important aspect is that UE5 game development is actively used not only in the gaming industry. This engine is used in architecture, in many possible simulations (medicine, drones), and educational programs, which means that the skills acquired during the internship have a wide range of applications.

The **object** of this thesis is game mechanics and game design in the adventure genre.

The **subject** of this thesis is the tools used to develop a desktop game based on the Unreal Engine 5.

The **purpose** of this thesis is to develop a game application based on the Unreal Engine 5 to promote games in the adventure genre. This will allow:

– to assess the relationship between theoretical concepts and their practical implementation;

– to develop skills in working with professional development tools;

– to develop systematic thinking regarding the organization of a software product's lifecycle.

The work employed **theoretical research** methods, including domain analysis, comparison of existing game solutions, and examination of the functional capabilities of Unreal Engine 5 and methods for designing interactive systems. **Practical methods** included designing the game environment, implementing program logic using Blueprints and C++, creating puzzles, interaction systems, and artificial intelligence elements, as well as testing and optimizing the application.

The **first chapter** explores the subject area, analyzes the characteristics of the adventure genre and current trends in game development, and provides an overview of similar titles.

The **second chapter** an analysis of the tools, methods, and requirements was conducted. Requirements for the game application were defined.

The **third chapter** details the process of implementing the software product using Unreal Engine 5, describes the project structure, the implementation of the character control system, interactions with objects, puzzles, the dynamic environment, audiovisual effects, and the behavior of game entities.

The **fourth chapter** presents the test results, an assessment of the application's performance and efficiency, and identifies areas for further improvement of the developed software product.

The **main result** of this work is a prototype adventure game for a personal computer that implements maze exploration, a puzzle system, story-driven narrative

elements, atmospheric audiovisual design, and scripted game events. The developed application can be used as a basis for the further development of a full-fledged game product, as well as an example of the practical application of modern interactive software development technologies

The qualifying bachelor's thesis internship includes an introduction, two chapters, conclusions, and a list of references. The introduction outlines the relevance, purpose, object, and subject of the work. The first chapter describes the subject area and similar applications. The second chapter presents the software requirements specification and describes the functionality. The conclusions summarize the stages of the application’s development.

Keywords: adventure, AI, analysis, application, C++, development, engine, features, indexing, prototype, testing, UE5.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 2
ВСТУП 3
1 АНАЛІЗ СУЧАСНОГО СТАНУ ІГРОВОЇ ІНДУСТРІЇ 5
1.1 Опис предметної області 5
1.2 Існуючі аналогічні застосунки 7
1.3 Особливості об’єкта роботи 10
Висновки до розділу 1 14
2 ФОРМУЛЮВАННЯ ВИМОГ ДО ІГРОВОГО ЗАСТОСУНКУ 15
2.1 Аналіз інструментарію, моделей та методів 15
2.1 Специфікація вимог до програмного забезпечення 19
Висновки до розділу 2 24
3 МОДЕЛЮВАННЯ ІГРОВОГО ЗАСТОСУНКУ 25
3.1 Архітектура програмного забезпечення 25
3.2 Система керування персонажем та взаємодії 30
3.3 Ігрові механіки та головоломки 34
3.4 Реалізація штучного інтелекту 39
3.5 Вибір технологій та компонентів 41
3.6 Опис інтерфейсів 43
Висноки до розділу 3 45
4 ТЕСТУВАННЯ ТА КЕРВІНИЦТВО КОРИСТУВАЧА 46
4.1 Кодування та тестування програмного забезпечення 46
4.2 Тестування програмного застосунку 51
4.3 Керівництво користувача 54
Висновки до 4 розділу 58
Висновки 59
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 60

ПЕРЕЛІК СКОРОЧЕНЬ

AI	–	Artificial Intelligence
CPU	–	Central Processing Unit
FPS	–	Frame Per Second
GPU	–	Graphics Processing Unit
NPC	–	Non Player Character
PC	–	Personal Computer
PCG	–	Procedural Content Generation
RPG	–	Role-Playing Game
UE5	–	Unreal Engine 5

ВСТУП

Світ зараз у епоху інформаційних технологій постійно змінюється. Однією з ключових складових підготовки є практичне застосування теоретичних знань у реальних проєктах, це дозволяє не лише закріпити навички, але й сформувати гнучкість та креативність. Актуальність обумовлена потребою у поглибленні компетенцій, пов’язаних із сучасними інструментами розробки програмного забезпечення, зокрема в ігровій індустрії. Такий жанр як adventure, з акцентом на сюжеті та взаємодії гравця зі світом, залишається важливим для демонстрації можливостей сучасних технологій. Використання потужних інструментів, як Unreal Engine 5, відкриває великі можливості для експериментів з інноваційними підходами до створення імерсивних досвідів, що відповідає світовим трендам у геймдеві або навпаки, задати тренд.

Отже жанр adventure, який обраний для проєкту, не випадковий, він поєднує в собі технічну складність, а також творчий потенціал, бо це ідеальна тема для експериментів, де можна відточити та продемонструвати навички аналізу проблем, прийняття дизайн-рішень та тестування гіпотез. Сьогодні, користуючись такими потужними інструментами, як Unreal Engine 5, розробник має доступ до технологій, які раніше були доступні лише великим студіям. Це відкриває можливості для багатьох розробників. За допомогою цього рушія , навіть невелика команда може розробити гру , яка буде популярна у всьому світі.

Наступне , це важливість практики, вона полягає також у тому, що вона формує системне бачення бо розробка – це не тільки про написання коду: це управління ресурсами, комунікація з потенційними користувачами, усвідомлення етичних аспектів. Робота з сучасними рушіями, таким як UE5, навчає вчитися в умовах швидкої еволюції технологій – адже інструменти оновлюються що кілька місяців, і вміння швидко освоювати нове стає критичним.

Також не менш важливом аспектом цієї теми практики є розвиток креативного мислення, комерційні застосунки є конкретними і не має творчого креативу, свободи, процес створення гри вимагає художнього підходу: створення

атмосфери, продумування персонажів, балансу між складністю та доступністю. Це виклик, який змушує експериментуватита придумувати своє, що є основою для інженерної майстерності.

Об’єктом кваліфікаційної роботи є ігрові механіки та дизайн гри у жанрі adventure.

Предметом кваліфікаційної роботи є інструменти розробки десктоп-гри на основі рушія Unreal Engine 5.

Метою кваліфікаційної роботи є розробка ігрового застосунку на основі рушія Unreal Engine 5 для популяризації ігор у жанрі adventure. Це дозволить:

– оцінити взаємозв’язок між теоретичними концепціями та їх практичною реалізацією;

– розвинути навички роботи з професійними інструментами розробки;

– сформувати системне мислення щодо організації життєвого циклу програмного продукту.

Відповідно до мети визначено такі завдання:

– аналіз вимог до сучасних adventure-ігор та можливостей їх реалізації;

– оволодіння інструментарієм Unreal Engine 5 для створення інтерактивних сценаріїв;

– моделювання системи проєкту;

– розробка архітектури проєкту з урахуванням принципів масштабованості та оптимізації;

– впровадження механік, спрямованих на залучення гравця через сюжет та візуальну складову;

– тестування готового проєкту.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ІГРОВОЇ ІНДУСТРІЇ

1.1 Опис предметної області

У сучасному світі розробка відеоігор є багатогранною сферою. Вона може поєднувати як технічні навички так і творче мислення та розуміння того , що потребує аудиторія. Жанр adventure, має довгу історію. Починаючи з текстових квестів багато років тому і до сучасних нарративних великих проєктів на кшталт "The Legend of Zelda", "Uncharted" або "Red Dead Redemption 2" його головна особливість — акцент на сюжеті, дослідженні світу та емоційному зануренні гравця. Шутери чи RPG використовують бойові механіки або прокачку гравця. А ігри жанру adventure часто фокусуються на розгадуванні таємниць, діалогах і прийнятті моральних рішень.

Отже на сьогодні цей жанр є популярним завдяки технологіям, які дозволяють створювати детальні світи з високим рівнем інтерактиву. Однак розробникам доводиться знаходити баланс між художнім задумом і технічними обмеженнями. Наприклад, створення великих локацій де є можливість вільно дослідити світ вимагає оптимізації ресурсів. Unreal Engine 5 сучасний рушій, який надає розробникам інструменти для того щоб могти реалізувати такі ідеї. Але не дивлячись на велику кількість доступних інструментів, розробники можуть розробити свої плагіни чи інструменти для більшої зручності або реалізації завдання.

UE5 став проривом в індустрії завдяки таким інструментам, як Nanite [5,6] та Lumen[7]. Nanite дозволяє імпортувати високополігональні 3D-моделі без втрати продуктивності, що критично важливо для adventure-ігор, де кожен об’єкт може мати нарративне значення, але поки Nanite[5,6] ще не достатньо дороблений. Тобто на будь-який mesh його не варто накладати, так як для деяких категорій Nanite може особливо нічого не змінити, або навпаки трохи погіршити, тому тут треба підбирати рішення де і для чого використовувати Nanite[5,6]. Lumen[7] забезпечує динамічне глобальне освітлення, створюючи реалістичні тіні та відблиски, які

підкреслюють атмосферу локацій. Це дуже крутий інструмент , адже світло відіграє однин з найголовніших компонентів для локації, за допомогою такого оствітлення десь можна пожертвувати якістю інший деталей на локації, адже при правильному світлі гравець не відчує ніякої різниці. Також завдяки технології Lumen[7] стало простіше налаштовувати оствітлення , так як налаштувати її відносно легко.

Також окрім технічних аспектів, жанр adventure вимагає уваги до деталей у сюжеті та персонажах. Гравець має відчувати, що його дії впливають на світ, навіть якщо зміни неочевидні. Наприклад, можливість вибору у діалозі може змінити ставлення NPC до гравця або відкрити альтернативну кінцівку. Реалізація таких механік вимагає ретельного планування сценарію, аби це все відчувалося живим та мало логіку.

Ігри сьогодення використовують технології AI, щоб оживити NPC. UE5 пропонує систему Behavior Trees, яка дозволяє налаштувати логіку поведінки NPC. Наприклад, як вони будуть реагують на гравця, пересуватися локацією чи взаємодіяти з об’єктами. І саме це додає глибини ігровому світу, роблячи його більш "живим".

UE5 має свої виклики, на вигляд може здаватись що це доволі простий рушій, інтуітивний. Але має свої проблеми, і щоб повністю розумітись треба прочитати досить велику документацію, "набити" руки, та постійно слідкувати за оновлення , щоб нічого не упустити. І висока деталізація графіки може призвести до падіння FPS на слабких ПК. Отже виходячи з цього, оптимізація стає ключовим етапом розробки. Рушій надає інструменти для аналізу продуктивності, такі як Profiler, який показує, які процеси "навантажують" CPU чи GPU. Також важливо правильно використовувати Virtual Texturing — технологію, що зменшує обсяг пам’яті, який використовують текстури.

Для будь-яких ігор важливим етапом є тестування. І навіть найкраще продуманий сюжет може "зламатися" через технічні баги , приклад цьому з останніх ігор на UE5 є Stalker 2, де досить велика кількість гравців не могла

просунутись по сюжету через такі баги. Необхідно перевіряти всі механіки, чи усі вони коректно працюють.

1.2 Існуючі аналогічні застосунки

Ринок adventure-ігор розроблених на UE5 досить великий. Дуже багато ігор розроблених на цьому рушії отримали всесвітння визнання та популярність. У цьому підрозділі я хочу продемонструвати різні підходи до використання технологій який надає UE5 та відповідно проаналізувати ігри.

Hellblade II: Senua’s Saga (рис 1.1, табл. 1.1) – поєднує елементи психологічного хорору та пригодницького жанру. Гра розповідає історію Сенеу, вона бореться зі своїми демонами в скандивській міфології. Рушій було використано для створення кінематографічної графіки, демонстрування чудових анімацій у тому числі обличчя та дуже атмосферних локацій з динамічним оствітленням.

Таблиця 1.1 – Характеристики Hellblade II: Senua’s Saga

Назва	**Hellblade II: Senua’s Saga**
Студія	Ninja Theory
Рік випуску	2024
Ключові технології UE5	Nanite, Lumen, MetaHuman, MetaSounds
Особливості нарративу	Кінематографічні катсцени, діалоги
Рушій	Unreal Engine 5.2

Рисунок 1.1 – Використання технології UE5 у Hellblade II: Senua’s Saga

Layers of Fear (рис 1.2, табл 1.2) – серія хорор-пригод від Bloober Team, де гравець досліджує психологічно напружені простори, розкриваючи таємниці головного героя. Студія використовуючи можливості UE5 створили досить цікавий сюжет та локації, які могли перетворюватись досить швидко. Ця команда використовувала World Partition для управління локаціями та системою Niagara для візуальних ефектів , а також освітлення Lumen.

Таблиця 1.2 – Характеристики Layers of Fear

Назва	**Layers of Fear**
Студія	Bloober Team
Рік випуску	2023
Ключові технології UE5	Niagara, Lumen, World Partition
Особливості нарративу	Психологічні ілюзії, перетворення локацій
Рушій	Unreal Engine 5.1

Рисунок 1.2 – Використання технології UE5 у Layers of Fear

The Talos Principle 2 (рис 1.3, табл. 1.3) – adventure гра від Croteam, філософська гра, де гравець має вирішувати головоломки в антиутопічному світі. Середовище розробки Unreal Engine 5 було використано задля створення великих локацій з детальним середовищем. Студія використовувала технологію Nanite для рендеру статуй та різних архітектур , щоб вони були у найвищій якості. Також для NPC використано Behavior Trees , які спостерігали за гравцем та коментували його дії.

Таблиця 1.3 – Характеристики The Talos Principle 2

Назва	**The Talos Principle 2**
Студія	Croteam
Рік випуску	2023
Ключові технології UE5	Nanite, Lumen, Behavior Trees
Особливості наративу	Діалоги з філософськими питаннями
Рушій	Unreal Engine 5.0

Рисунок 1.3 – Використання технології UE5 у The Talos Principle 2

Такі ігри дають розуміння у якому напрямку треба працювати створюючи власний застосунок. А також , як і де використовувати технології для кращого відображення ідеї.

1.3 Особливості об’єкта роботи

Застосунки у жанрі adventure популярні через сюжету, атмосферу, емоційність , що у свою чергу накладає певні задачі для розробника на кожен етап під час створення гри. Отже сюжет є основою зв'язку до людини яка грає за гравця, що означає, кожен діалог, кожен предмет або локація мають бути пов’язані так, щоб задовільняти загальну ідею. Важливу роль у створенні атмосфери та імерсивності відіграють аудіо та візуальні компоненти. Музика, звукові ефекти, освітлення — все це має бути органічним , імерсивним, щоб гравець відчував себе частиною ігрового світу.

Однак інтеграція цих елементів — це не лише креатив або мистецтво, а й технічна задача. Неправильно налаштований звук може перевантажити сцену звуком, а надто яскраве освітлення — може зменшити напругу від ситуації.

Сучасні технології, як Unreal Engine 5, можуть спростити процес створення, але в той час ставлять нові виклики. Технології — це лише спосію щоб втілити

ідею. Навчитися обмежувати себе, навіть маючи доступ до нескінченних можливостей — це один із тих аспектів роботи де ти проявляєш свою професійність.

Не менш важливою є робота з гравцем. Наприклад, додавання системи непрямих підказок.необхідність адаптації під різні платформи. Наприклад, гра, створена для ПК, може вимагати допрацювань для консолей: зміни в управлінні, оптимізація інтерфейсу під геймпади, перевірка продуктивності на різному залізі. Не можна забувати і про тестування. Навіть сильно продуманий сюжет не зможе вивезти якщо будуть технічні баги.

Розробка adventure-гри — це не лише технічний процес, а й особистисний досвід. Кожен розробник хоче вкладсти у гру частинку себе: свої страхи, надії, роздуми, сенси, натяки. І саме ця складова яка є осболивою робить ігри цього жанру такими особливими. Вони можуть бути як розважальними так і змушувати задуматися, співпереживати, дивуватися. Також дуже важливим у такому жанрі є й те, що аудиторія таких ігор залишається вірною цьому жанру протягом багатьох років і чекають на нові ігри у такому жанрі. Гравці, які грали у ігри з цим жанром колись у дитинстві, досі шукають схожі ігри, але вже у сучасному виконанні. При цьому покоління яке росте зараз, може бути втомленим від шаблонних шутерів, також може відкрити для себе цей жанр. На мою думку дуже великий успіх мав "Disco Elysium" і відповідно продемонстрував, що гра де потрібно в більшості читати текст і обирати рішення, теж може стати хітом.Не потрібно забувати і про те, що ігри у цьому жанрі стали полем для експериментів з новими можливостями , механіками. Віртуальна реальність, інтерактивні фільми, ігри з елементами документалістики — все це вперше з’являється саме в цьому жанрі. До прикладу, "The Walking Dead: Saints & Sinners" у VR продемонстрував, як можна відродити класичні механіки дослідження та діалогів.

Розробка ігрового застосунку в жанрі adventure за допомогою рушія Unreal Engine 5 має багато унікальних особливостей, які зумовлені як вибором рушія, так і жанром гри. Насамперед, UE5 надає доступ до чудових інструментів високого

рівня, таких як Nanite, Lumen, Niagara та MetaSounds, що дозволяє створювати великі проєкти, навіть маленьким студіям або взагалі декільком людям . Однак робота з цими технологіями має бути з розумінням цих технологій. Розуміння їх можливостей та обмежень. Особливо коли може бути обмеження у ресурсах.

Перше це використання **Nanite** для високодеталізованих об’єктів, наприклад, для створенні локації з палацом (рис 1.4) імпортовано 3D-моделі з мільйонами полігонів на сцену, локацію. Завдяки Nanite рушій може сам автоматично оптимізувати їх для рендерингу, а це в свою чергу дозволяє зберегти деталізацію та покращити FPS.

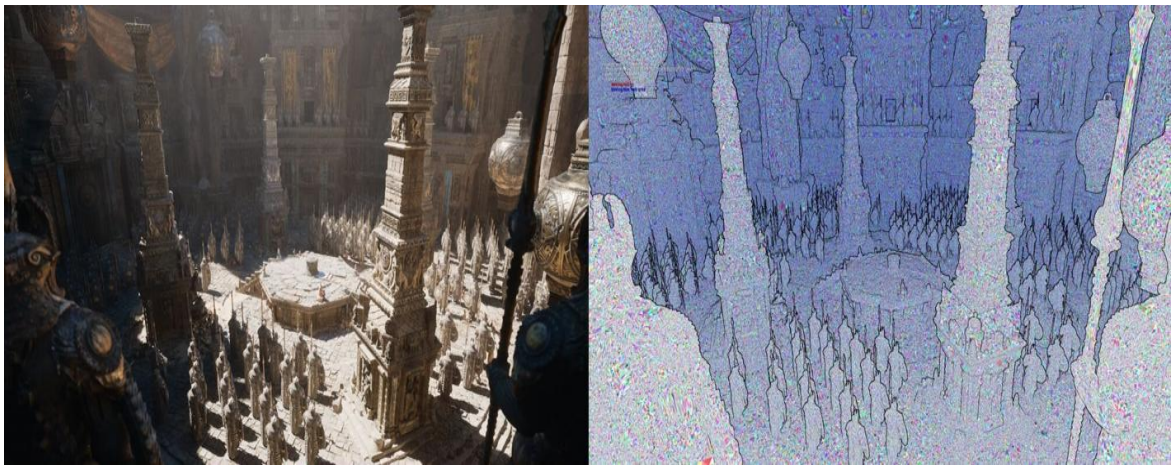

Рисунок 1.4 – Приклад використання технології Nanite

Другою особливістю є динамічне освітлення Lumen. У грі є сцени, де гравець переходить з яскраво освітлених зон у темніші або темряву, і Lumen забезпечує плавні переходи світла, наприклад, у сцені є печера з отвором (рис 1.5) , через отвір світло проходить дуже реалістично та забезпечую цьому місцю атмосферно, а також світло рефлектує з іншими об'єктами в печері. Це потребує оптимізації, щоб система сильно не навантажувалась.

Рисунок 1.5 – Приклад використання технології Lumen

Отже оптимізація продуктивності буде окремим етапом роботи. До прикладу, після додавання динамічних ефектів дощу через Niagara, FPS може просідати. Виходить щоб вирішити цю проблему, доведеться зменшити кількість частинок і можливо активувати Virtual Texturing для текстур. Також можна використовувати такий інструмент , як Profiler UE5, щоб виявити місця де використовується занадто багато ресурсів CPU/GPU. Virtual Texturing технологія для оптимізації використанння текстур.

Тепер проаналізую кожну технологію детальніше. Nanite , він дозволяє імпортувати 3D-моделі з мільйонами полігонів без ручної оптимізації, наприклад в UE4 рекомендована кількість полігонів була до 150 тисяч. З цієї технологією відповідно розширились можливості для більш реалістичної , імерсивної, деталізованої графіки. Керуючись статистикою від Epic Games ця технологія зменшує використання VRAM на як менше 30%, якщо порівнювати з лодами.
Єдиним обмеженням для цієї технології підтримка тільки статичних об’єктів.

Динамічне освітлення Lumen забезпечує глобальне оствітлення з багатьма джерелами світла у реальному часі. Для такого ж ефекту у минулі версії рушія потрібно було використовувати бейкінг , що могло займати багато часу. Якщо

говорити про продуктивність , то за даними Digital Foundry , Lumen використовує 15-20% потужності GPU на RTX 3070 у 4K. Наразі все більше компаній які працюють на цьому рішію , все більше використовують цю технологію.

Висновки до розділу 1

У першому розділі було проаналізовано особливості розробки засносунку у жанрі adventure на основі рушія Unreal Engine 5 та проаналізувано ключові моменти, які роблять цей жанр актуальним і перспективниму майбутньому. Перш за все, ігри у жанрі adventure залишаються унікальними у світі відеоігор завдяки їхньому фокусу на сюжет, атмосферу та емоційний зв’язок .

Аналіз предметної області продемонстрував, що технології сьогодення, такі як UE5 та його інструменти, відкривають нові можливості для реалізації творчих ідей. Було досліджено аналоги (Hellblade II, Layers of Fear, The Talos Principle 2 тощо) що продемонстрували, що UE5 активно використовується великими компаніями і невеликими командами, кожен з цих проєктів продемонстрував різні підходи: від кінематографічної графіки до психологічних ілюзій. Особливості об’єкта роботи розробки гри у жанрі adventure на UE5 виявили низку викликів. Важливим етапом стала оптимізація: навіть з такими технологіями, як Virtual Texturing, Profiler.Також важливим висновком стало те, що технології UE5, хоч і спрощують багато процесів, не вирішують усіх проблем. Окремо від іншого варто відзначити роль тестування. Процес вияву та виправлення помилок стане важливим етапом. Він вимагає уваги до деталей . Відповідно треба бути уважним до коректності діалогів та фізики взаємодії об’єктів.

Підсумовуючи, аналіз предметної області довів, що ігри у жанрі adventure залишаються актуальними через свої ідеї та реалізації.

2 ФОРМУЛЮВАННЯ ВИМОГ ДО ІГРОВОГО ЗАСТОСУНКУ

2.1 Аналіз інструментарію, моделей та методів

Завжди у розробці відеоігор рушії займали ключову роль як основний інструмент. Тому для цього завдання був обраний UE5. У підрозділі буде проведено аналіз інструментів, моделей і методів, що дозволяють виконати поставлені задачі.

Перше це Інструментарій розробника. Основними технологіями Unreal Engine 5 що будуть застосовуватись при розробці є:

– Nanite — технологія яка дозволяє деталізувати об'єкти без особливої втрати продуктивності;[5,6]

– Lumen — система глобального освітлення у реальному часі, яка покращує імерсивність;[7]

– Blueprints — візуальна мова програмування, що дозволяє створювати логіку без написання коду;

– Unreal Motion Graphics — інструмент для створення інтерфейсу.;

– MetaHuman — інструмент для генерації реалістичних персонажів;

– Behavior Trees — шаблони поведінки для створення AI.

Допоміжні інструменти:

– Quixel Bridge — для імпорту 3D-об’єктів;

– Visual Studio — основне IDE для роботи з кодом.

Сучасні методи моделювання ігрового процесу передбачає застосування:

– UML-діаграм для моделювання логіки систем, класів, взаємодії між об'єктами гри;

– архітектурних шаблонів для структурування ігрових систем;

– поведінкові моделі для NPC;

– методологій Agile/Scrum для організації процесу розробки.

Методи створення adventure-ігор зазвичай фокусуються на:

– дослідженні оточення;

– взаємодії з об'єктами та персонажами;

– розв'язанні головоломок та проходженні сюжетних місій.

Методи, які використовуються для реалізації цього функціоналу:

– Event-driven programming — запуск логіки у відповідь на дії користувача;

– Use-case analysis — для побудови сценаріїв взаємодії гравця з грою;

– Narrative Design — побудова діалогів, нелінійних сюжетів, розвиток персонажів.

Для аналізу сучасного стану індустрії було опрацьовано низку наукових публікацій. Актуальні тенденцій:

– застосування UE5 для розробки ігор з відкритим світом;

– розвиток інструментів для procedural content generation;

– інтеграція UE5 з системами штучного інтелекту;

– упровадження методів трасування променів у реальному часі.

Розглядаючи моделі, які використовуються для реалізації геймплейної логіки, потрібно зауважити про важливість внутрішньоігрових сценаріїв, які визначають послідовність подій, реакцію гри на вибір гравця та наслідки дій. В іграх цього жанру важливо забезпечити відчуття цілісного сюжетного простору, де рішення гравця мають реальний вплив. Саме тому рушії нового покоління, мають інструменти для сценарного моделювання, які можна адаптувати до різних сюжетних структур лінійних, розгалужених або циклічних. Крім того, активна робота спільноти Unreal сприяє постійному оновленню методик розробки, появі нових шаблонів, готових механік та систем інтеграції зі сторонніми сервісами.

Розробка десктопного застосунку в жанрі adventure з використанням рушія Unreal Engine 5 передбачає використання потужного та різноманітного інструментарію, що забезпечує як візуальну, так і технічну складову продукту. Вибір Unreal Engine 5 зумовлений його широкими можливостями у створенні інтерактивних 3D-середовищ, підтримкою сучасної графіки, а також активною спільнотою розробників і постійною еволюцією функціоналу.

UE5 поєднує в собі новітні технології для створення реалістичних, глибоких та продуктивних ігрових проєктів. Однією з основних інновацій рушія є Nanite[5,6] технологія віртуалізованої геометрії, яка дозволяє імпортувати високополігональні 3D-моделі без необхідності створення лодів. Це значно зменшує час підготовки моделей та підвищує деталізацію об'єктів без втрати продуктивності.

Ще однією технологією, важливою для створення атмосфери є Lumen[7] система глобального освітлення у реальному часі. Lumen[7] дозволяє симулювати складні світлові взаємодії без попереднього запікання освітлення, що дає змогу гнучко змінювати середовище та адаптувати освітлення до динамічних подій гри. Це особливо корисно в сюжетних сценах або при зміні часу доби.

Для створення логіки ігрового процесу у UE5 використовується система Blueprints візуальне програмування. Це значно спрощує розробку та дозволяє тестувати ігрові сценарії в реальному часі. Blueprints застосовуються для керування подіями, створення систем квестів, інтеракції з об’єктами та контролю персонажів.

Важливим елементом будь-якого ігрового застосунку є інтерфейс користувача. У UE5 для цього передбачений модуль Unreal Motion Graphics . UMG дозволить створювати кастомізовані UI-елементи, включаючи інвентар, діалоги, меню тощо.

Трендом у створенні персонажів , які будуть виглядати реальними є використання MetaHuman Creator що дозволяє генерувати таких персонажів. Так як metaHuman інтегрований у UE5 це відповідно дозволяє швидко створювати прототипи AI.

Для створення поведінки NPC застосовуватимуться Behavior Trees система яка відповідає за шаблони поведінки, вона дозволяє задавати різні сценарії дій. Такий підхід забезпече гнучкість у керуванні штучним інтелектом.

Додатковими інструментами для роботи з контентом є Quixel Bridge, який дає доступ до бібліотеки Megascans . Це значно пришвидшить процес наповнення світу .

Для написання коду на C++ використовується Visual Studio середовище розробки, яке повністю інтегрується з UE5. Через Visual Studio реалізуються складні системи гри, зокрема геймплейні компоненти, системи збереження, AI-модулі, логіка боїв тощо.

Окрім безпосередніх інструментів рушія, важливе місце в архітектурі гри займає моделювання логіки та структури гри. Для цього використовуються UML-діаграми класів, об'єктів, послідовностей, що дозволяють візуалізувати взаємозв’язки між елементами системи. Це дає змогу оптимізувати взаємодію об’єктів гри.

Архітектурні шаблони Model-View-Controller, Game Loop, Component-Based Design використовуються для структуризації коду гри. Це дозволяє створювати масштабовану, легко підтримувану та тестовану архітектуру. У жанрі adventure важливо дотримуватись принципу розділення логіки сюжету, візуалізації та взаємодії з гравцем.

Поведінкові моделі NPC включають патерни з використанням finite state machines, планувальників дій, а також умовних блоків логіки, що керуються на основі даних гравця або подій гри. У комплексі ці підходи дозволяють створювати багатовимірних персонажів, які реагують на події, здійснюють вибір та демонструють реалістичну поведінку.

Narrative design відповідає за побудову сюжету. Для цього використовуються діалогові системи, нелінійні сюжетні дерева, варіативні кінцівки та збереження виборів гравця. Це дозволяє занурити гравця у світ гри та дати йому відчуття контролю над історією.

Також застосовується Data-Driven Design підхід, за якого дані про об'єкти, місії, діалоги, інвентар зберігаються у зовнішніх файлах, що дозволяє швидко змінювати контент без редагування коду.

Тестування продуктивності проєкту потрібно буде здійснювати за допомогою Unreal Insights. Який буде збирати детальну інформацію про FPS, використання пам'яті та затримку обробки подій. І також не менш важливо

використовувати Stat Commands . Який потрібен для моніторингу системних параметрів у реальному часі, що дозволяє швидко виявити проблемні місця. Для аналізу графічної продуктивності застосовується GPU Visualizer, він допомагає оптимізувати тіньові карти, постобробку та рендеринг.Також для реалізації складної фізики використовується Chaos Physics Engine. Він підтримує руйнування об’єктів, симуляцію тканини, зіткнення та інші фізичні явища.

2.1 Специфікація вимог до програмного забезпечення

1) Призначення та межі проєкту

1.1)Призначення системи: Розробка інтерактивного пригодницького ігрового десктоп-застосунку у жанрі adventure, орієнтованого на дослідження ігрового світу, взаємодію з предметами, виконання завдань та розкриття сюжетної лінії.

1.2)Погодження, що ухвалені в програмній документації: ігровий застосунок повинен бути сумісний із Windows 10/11. Основним рушієм є Unreal Engine 5. Графіка повинна бути максимально як дозволяє ПК. Всі скрипти та логіка мають бути організовані згідно з архітектурою, прийнятою в UE5 (Blueprint/C++).

1.3) Межі проєкту: Гра буде доступна лише на ПК. Багатокористувацький режим не підтримується. Розробка мобільної або VR-версії не передбачена в рамках цього проєкту.

2) Загальний опис

2.1) Сфера застосування: Десктоп-застосунок призначений для геймерів віком від 12 років, для геймерів які полюбляють пригодницькі ігри з сюжетною лінією. Гра також може бути використана для демонстрації навичок розробки в Unreal Engine 5.

2.2)Характеристики користувачів: Гравці повинні мати базові навички користування ПК та знайомство з іграми жанру adventure. Мінімальний

рівень знань з того , як встановити гру, налаштувати графіку та використовувати клавіатуру/мишку.

2.3) Загальна структура і склад системи:

- Головне меню;
- Ігровий рівень (місцевість, NPC, завдання);
- Підсистема інвентаря;
- Підсистема діалогів;
- Підсистема збереження прогресу;
- Підсистема музики та звукових ефектів;
- Система виконання квестів;
- Карта світу;
- Взаємодія з об’єктами, які не підлягають збору в інвентар.

2.4) Загальні обмеження:

- Мінімальні вимоги:Windows 10, CPU i5, RAM 8GB, GPU GTX 1050Ti;
- Гра не повинна перевищувати 80 GB на диску.

3) Функції системи

3.1) Система запуску гри:

3.1.1) Опис: Гравець може запустити гру безпосередньо через виконуваний файл.

3.1.2) Вхідна інформація: Клік мишкою, вибір через меню.

3.1.3) Вихідна інформація: Запуск сцени головного меню.

3.1.4) Функціональні вимоги: Підтримка завантаження необхідних ресурсів при старті.

3.2) Система збереження прогресу:

3.2.1) Опис: Гравець може зберегти свій прогрес вручну або автоматично.

3.2.2) Вхідна інформація: Команда збереження.

3.2.3) Вихідна інформація: Файл збереження у форматі JSON/UE5 SaveGame.

3.2.4) Функціональні вимоги: Можливість зчитувати та перезаписувати стан гри.

3.3) Взаємодія з NPC

3.3.1) Опис: Гравець може розмовляти з неігровими персонажами.

3.3.2) Вхідна: Підхід до NPC, натискання клавіші.

3.3.3) Вихідна: Відкриття діалогового вікна.

3.3.4) Функціональні вимоги: Реалізація дерев діалогу.

3.4) Переміщення по локації:

3.4.1) Опис: Гравець може вільно рухатися світом гри.

3.4.2) Вхідна: Клавіші WASD, мишка.

3.4.3) Вихідна: Анімація руху персонажа.

3.4.4) Функціональні вимоги: Реалізація контролера руху.

3.5) Система інвентарю:

3.5.1) Опис: Гравець може збирати, переглядати та використовувати предмети.

3.5.2) Вхідна: Взаємодія з предметами, відкриття меню інвентаря.

3.5.3) Вихідна: Відображення предметів, їх опис.

3.5.4) Функціональні вимоги: Зберігання та редагування масиву предметів.

3.6) Система виконання квестів:

3.6.1) Опис: Гравець отримує та виконує сюжетні або побічні завдання.

3.6.2) Вхідна: Активізація квесту через діалог або об'єкт.

3.6.3) Вихідна: Прогрес виконання, винагорода, оновлення статусу квесту.

3.6.4) Функціональні вимоги: Структура для зберігання стану квестів, можливість відслідковувати виконання.

3.7) Карта світу

3.7.1) Опис: Гравець може переглядати карту поточної локації або всього світу.

3.7.2) Вхідна: Натискання клавіші, відкриття меню карти.

3.7.3) Вихідна: Відображення карти з позначками.

3.7.4) Функціональні вимоги: Підтримка масштабування, динамічне оновлення інформації на мапі.

3.8) Взаємодія з об'єктами, які не можна підібрати:

3.8.1) Опис: Гравець може активувати механізми, читати записки, відкривати двері тощо.

3.8.2) Вхідна: Підхід до об'єкта, натискання клавіші взаємодії.

3.8.3) Вихідна: Зміна стану об'єкта (анімоване відкриття, показ тексту).

3.8.4) Функціональні вимоги: Реалізація системи тригерів, інтерактивні компоненти об'єктів.

4) Вимоги до інформаційного забезпечення:

4.1) Джерела і зміст вхідної інформації: Дані вводяться користувачем або надходять з ігрового рушія (сценарії, дані збереження, конфігурації).

4.2) Нормативно-довідкова інформація:

4.2.1)Стандарти UE5.

4.2.2) Класифікація предметів, типів NPC, локацій згідно внутрішніх довідників проєкту.

4.3) Вимоги до способів організації, збереження та ведення інформації:

4.3.1) Усі збереження мають створюватися у вказаній системній директорії користувача.

4.3.2)Використання структурованих форматів типу JSON або proprietary SaveGame UE5.

5) Вимоги до технічного забезпечення:

5.1) Комп'ютер користувача повинен відповідати мінімальним системним вимогам гри.

5.2)Має бути наявне підключення до інтернету для завантаження оновлень.

6) Вимоги до програмного забезпечення

6.1) Архітектура програмної системи: Розподілена на модулі: ядро, інтерфейс, логіка, збереження, UI, звук.

6.2) Системне програмне забезпечення: Windows 10/11 x64.

6.3) Мережне програмне забезпечення: Не використовується (гра офлайн).

6.4) Програмне забезпечення ведення інформаційної бази: Внутрішнє сховище UE5.

6.5) Мова і технологія розробки ПЗ: C++, Blueprint, Unreal Engine 5.

7) Вимоги до зовнішніх інтерфейсів:

7.1) Інтерфейс користувача: Меню гри, HUD, інвентар, діалоги, карти.

7.2) Апаратний інтерфейс: Клавіатура, мишка.

7.3) Програмний інтерфейс: Взаємодія між модулями UE5 через Blueprint або C++ API.

7.4) Комунікаційний протокол: Внутрішні виклики функцій UE5, міжмодульна взаємодія.

8) Властивості програмного забезпечення:

8.1) Доступність:

Доступна для завантаження на офіційному сайті розробника або через лаунчер.

8.2) Кросформленість: Проєкт може бути портований на інші платформи з підтримкою UE5 у майбутньому.

8.3) Продуктивність: Гра має запускатися з FPS не менше 30 на мінімальних налаштуваннях.

8.4) Надійність: Система не повинна аварійно завершуватись при стандартних сценаріях гри.

8.5) Безпека: Захист даних користувача

9) Інші вимоги:

9.1) Підтримка української та англійської мов.

9.2) Адаптація інтерфейсу під різні роздільності екрану.

Висновки до розділу 2

У другому розділі було детально розглянуто інструментарій, моделі та методи, необхідні для створення сучасного пригодницького ігрового застосунку за допомогою рушія Unreal Engine 5. Вибір саме UE5 зумовлений його потужністю, універсальністю.Усі інструменти для розробки та підходи спрямовані на створення якісного продукту що відповідає поставленому завданню. Врахування сучасних тенденцій, таких як процедурна генерація та AI, забезпечує завданню бути потрібним продуктом зараз. Таким чином, моделювання об’єкту та предмету у рамках даного завдання відповідає сучасним стандартам індустрії відеоігор і дозволяє створити масштабний, технологічно просунутий застосунок.

3 МОДЕЛЮВАННЯ ІГРОВОГО ЗАСТОСУНКУ

3.1 Архітектура програмного забезпечення

Архітектура для програмного забезпечення виконана за подульним принципом. Це підхід , коли замість збирання усього в одну купу, розподіляють на окремі незалежні частини - модулі. Відповідно ці частини можна буде в будь-який момент відредагувати , замінити, комбінувати. При такому підході це все можна робити не змінюючи усю або більшу частину структури та можна не переживати , що через цю зміну зламається щось важливе.

3.1.1 Структура програмного застосунку

Розділення на різні модулі дозволяє оптимізувати процес розробки, та спростити покращення або виправлення під час розробки та в майбутньому.
Нижче представлені основні модулі.

Таблиця 3.1 – Основні модулі

Модуль	**Призначення**
PCG Maze Generator	Генерація лабіринту
Puzzle	Логіка головоломок
Interaction	Система взаємодії
Notes System	Система записок
AI Event	Система поведінки Мінотавра
Save	Збереження стану гри
UI	Меню, HUD, мапа
Progression System	Система для контролю проходження лабіринту

Задля генерації лабіринту використовується PCG. Модуль Puzzle відповідає за головолмки зі світлом чи об’єктами. Також присутній модуль взаємодії з об’єктами , такими як записки або об’єкти у головоломках. Система записок реалізована для подачі наративу, поведнка Мінотавра потрібна для реалізації погоні за гравцем. Реалізовано збереження , контрольована прогресія та інтерфейс користувача.

3.1.2 UML-діаграма класів

Щоб відобразити структуру програмного забезпечення створено UML-діаграма класів, на якій продемонстровано основні зв’язки та системи гри. Усе це реалізовано на взаємодії окремих модулів.

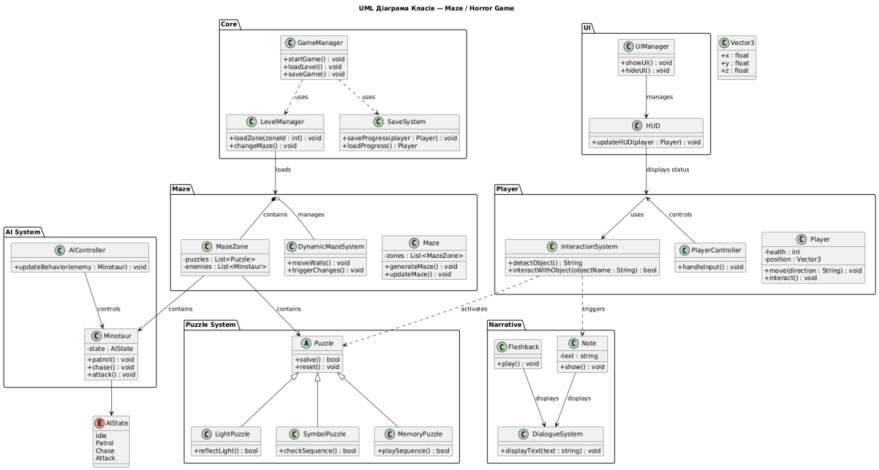

Рисунок 3.1 – UML-діаграма класів

Maze відповідає за процедурну генерацію лабіринту та створенням маршруту для квестів і відповідно головного маршруту. Компонентом який є основним тут буде клас Maze.

Player дозволяє керувати персонажем та взяємодіяти з інтерактивними об’єктами. Клас Player відповідає за основні параметри гравця, а клас InteractionSystem дозволяє зрозуміти які предмети або об’єкти є інтерактивними.

Puzzle System відповідає за логіку головоломок та відповідно за тим щоб умови для проходження головоломок були виконані. Використовуються різні типи головоломок , як от світлові чи інтерактивні.

AI System реалізує та коригує поведінку Мінотавра. Це відбувається під час різних заскриптованих подій задля створення атмосфери та напруги на гравця. Minotaur це клас який має декілька станів як: стан спокою, переслідування, атака.

UI відповідає за відображення інтерфейсу для гравця. Гравець бачитиме HUD, квести , витривалість та меню гри. Також UI буде відповідати за відображення записок після того , як гравець з ними почав взаємодіяти.

UML-діаграма демонструє описану вище модульну структуру програмного застосунку та відображає взаємодію різних молулів. Це дозволяє спростити та зробити більш комфортним розробку.

3.1.3 Діаграма розгортання

Розроблена UML-діаграма розгортання задля відображення структури програмного застосунку. Ця діаграма демонструє взяємодіє програмних модулів під час гри.

Архітектура програмного застосунку була реалізована як singleplayer застосунок. Застосунок який виконується на PC та стан гри зберігається локально.

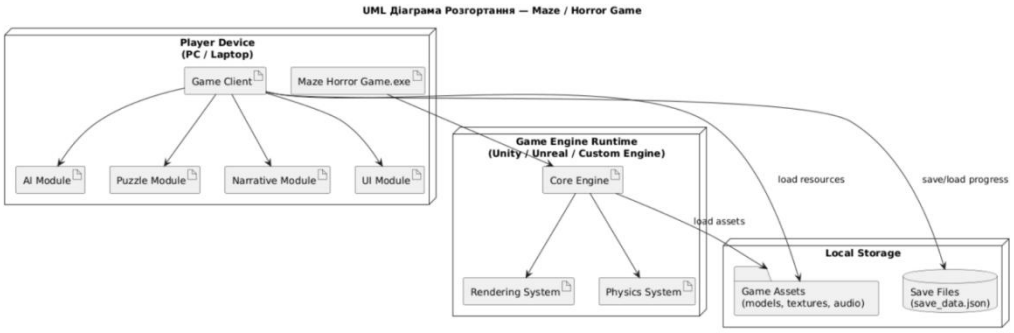

Рисунок 3.2 - UML-діаграма розгортання

Виконання програмного застосунку забезпечує Game Engine Runtime, для роботи фізики , рендеренгу та логіки гри. На PC гравця є застосунок гри у якому містяться потрібні модулі для коректної роботи гри. Збереження прогресу гравця у грі відбувається за допомогою локальних збережень, де відповідно зберігатимуться стан світу , виконані квести або не виконані, дані гравця, та прогрес по грі.

3.1.4 PCG генерація лабіринту

Головна складова програмного застосунку є генерація досить великого лабіринту за допомогою PCG. Процедурна генерація дозволяє створювати лабіринт будь-якого розміру. Сама генерація відбувається шляхом комбінуванням вже створених Level Instances.

Також дуже важливим є те що у створених Level Instances об’єкт який використовується для побудови стін мусить мати pivot у нижньому куті цього об’єкту. Без цього генерація буде не вдалою тому , що Level Instances не зможуть правильно зістикуватись одним з одним , тому буде багато пустих ділянок та Level Instances будуть генеруватись один на одному , також шляхи виходів не будуть правильно приєднуватись.

Все це відбувається за допомогою PCG Graph , у ньому ми зазначаєму усе потрібне для генерації, що за чим має бути, аналізуємо точки генерації, формуємо основний маршрут , формування правильного приєднання Level Instances один до одного , та створення основного шляху задля проходження гри. Також через те що ми використовужмо вже готові Level Instances , вони мають різну кількість виходів, через це по всіх краях лабіринту є виходи. Задля вирішення цієї проблеми проаналізовано точки які потрібно закрити , та на цих точках спавнити об’єкт стіни , такий самий як і по всьому лабіринту та задати правильно форму.

Усі Level Instances вже можна сказати повністю дороблені частинки лабіринту. Тобто там розставлена рослиність та об’єкти. Також відповідно до тестування проводилась оптимізація.

В результаті виходить готова система генерації лабіринту, з урахованням усіхх тонкостей та проблем. Генерується лабіринт з різних Level Instances які мають по 2, 3, 4 виходи. Генерується головний шлях для проходження лабіринту з одного Level Instance який має 4 виходи та не використовувався при генерації основного масиву лабіринту. Два окремі шляхи генеруються на основі того як проходить головний маршрут, тобто від головного маршрути є 2 шляхи у різні сторони, також усі непотрібні виходи закриваються.

Сам PCG Graph можна розділити на 3 частини:

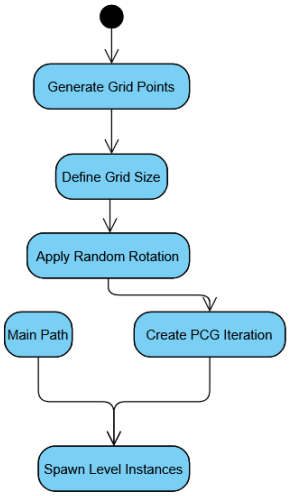

Рисунок 3.3– Схема генерації структури лабіринту

Перше що відбувається у PCG graph - це сама генерація лабіринту за допомогою Level Instances, без основного шляху та з купою виходів.

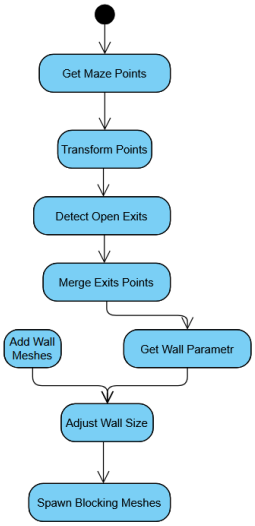

Рисунок 3.4 – Схема спавну блокуючих стін

Друге - спавн блокуючих стін. Це потрібно для того аби прибрати всі зайві виходи.

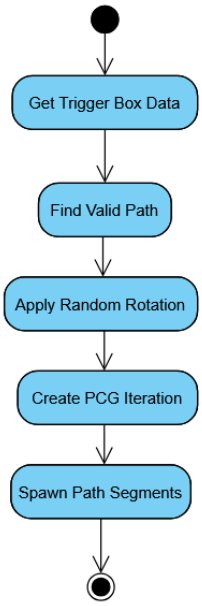

Рисунок 3.5 – Схема генерації основного шляху

Третє - це генерація основного шляху. Це відбувається коли вже лабіринт згенерований, і аналізуючи точки та розмір який потрібен, на головному шляху спавняться Level Instances які ще не використовувались і потрібні тільки для цього.

Така функціональність дозволяє спростити підримку чи розширення у подальшому цього застосунку. Запропонована реалізація підвищує варіантивність як самого лабіринту так й ігрового процесу загалом.

3.2 Система керування персонажем та взаємодії

Реалізовано систему керування персонажем . Буде описано які стани матиме гравець, які отримає можливості. Також описано з чим і як персонаж буде взаємодіяти.

3.2.1 Керування персонажем

Керування персонажем реалізовано через використання Enchanced Input Unreal Engine 5, що обробляє введення корисувача та взаємодію зі світом.

Персонаж маж базові можливості : ходьба, біг, огляд камери, взаємодія з об’єктами , ліхтар.

Таблиця 3.2 - Основні дії персонажа

Дії	**Клавіші та миша**
Рух	WASD
Огляд	Mouse
Взаємодія	E
Біг	Left Shift
Ліхтар	F
Карта	M
Пауза	ESC

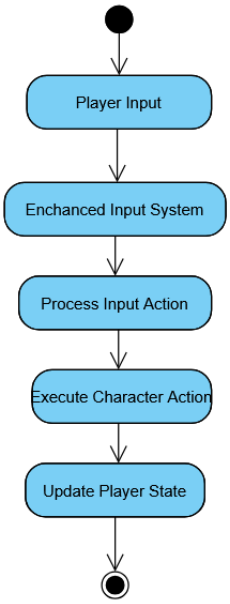

Рисунок 3.6 - Алгоритм обробки введення користувача

BP_PlayerCharacter клас у яку реалізовано всю логіку керування персонажем.Під час будь-якої з дій описаних у таблиці 3.2 використовуються Input Actions та Input Mapping Context, саме це дозволяє керувати системою введення для персонажа.

Коли персонаж отримує команду від користувача зробити відповідну дію система за це відпоповідає і змінює або стан гравця .

3.2.2 Ігрові механіки персонажа та інтеракція з об’єктами

Взаємодія відбувається за допомогою окремої системи інтеракції. Це дозволяє працювати з записками , які є один з найголовніших для наративу. Проводити інтеракції з головоломками та різними об’єктами для виконання головоломок.

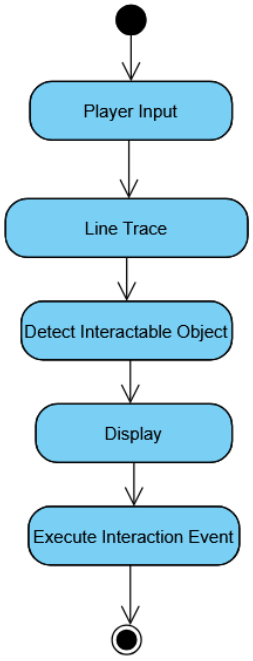

Рисунок 3.7 - Алгоритм взаємодії з об’єктами

Цей алгоритм відбувається під час натискання клавіші для взаємодії з об’єктом. Після натискання клавіші перед камерує персонажа виконується Line Trace для перевірки чи дійсно тут є інтерактивний об’єкт. Якщо це успішно виконається відповідний скрипт.

Такий алгорит дозволить будувати взаємодію з будьякими типами інтерактивних об’єктів, що значно спрощує процес розробки та реалізацію.

Система витривалості гравця

Механіка яка не дозволяє гравцю весь час швидко пересуватись по лабіринту. Реалізація витривалості є критичною для атмосфери проходження. Під час бігу запас питривалості відповідно зменшується , і коли гравець переходить на ходьбу - витривалість відновлюється автоматично.

Використання витривалості дозволяє грацю швидше пересуватись лабіринтом, але ще однією функцією є те , що це дозволить тікати від Мінотавра. Який буде змушений переслідувати гравця деякий час після скриптованого івенту, що надасть гравцю напруження під час проходження гри.

Ліхтар та приховані підказки

Механіка ліхнатря використовується для освтілення собі шляху та загалом світу, бо не усідю в лабіринті потрапляє світло. Також використовується як елемент для знаходження підказок шляхом наведення на них ліхтарем.

Саме ліхтар дозволяє віднайти підказки або символи які були раніше приховані. Такі підказки та символи розташовані на поверхнях ігрового середовища.

Основою такої механіки є промені світла та матеріали з шейдерами які їх приховують, тобто Emissive Materials. Цей метод я найбільш підходящим та оптимізованим.

Логіка системи така :

– Увімкнення ліхтаря, активується світло;

– Перевірка поверхні, для пошуку матеріалу, коли гравець заходить у зону головоломок з ліхтаря пускаються Line Trace для пошуку матеріалу;

– Поява схованого матеріалу, якщо гравець тримав ліхтар на такому матеріалі, він починає поступово проявлятись, і його буде видно тепер завжди.

Взаємодія з записками

Наратив у грі реалізований тільки через записки , які ще потрібно вдішукати та щоб отримати пройти головоломки. У них містяться фрагменти історії

персонажа, саме через що він і потрапив у цей лабіринт. Цей наратив є основою для вибору у фіналі гри.

Процес інтеракції відбувається з того що гравець бачить записку, підходить та натискає кнопку взаємодії. Після цього відбувається взаємодія з UI яка відображає записку та її вміст. У деяких випадках грацю треба буде перевертати записки. Це потрібно для вирішення головоломки, отримавши записку гравець побачить пустий лист, перевернувши отримає підказку що робити та поле для вводу символів , щоб активувати скрипт для прояви тексту з іншої сторони.

Підсумовуючи можна сказати що системи керування та взаємодії що були реалізовані забезпечують для гравця повну функціональність для проходження лабіринту та квестів. Використання такого підходу дозволило об’єднати деякі механіки в єдину систему.

3.3 Ігрові механіки та головоломки

Для цікавого проходження гри було реалізовано низку механік та головоломки. Вони допомагають цікавіше проводити час бродячи лабіринтом, та просувають далі згідно наративу. Присутні як взаємодія з навколишнім середовищем так і scripted events для яких потрібно виконати певні умови.

Система головоломок

Система головоломок є однією з найважливіших у цьому програмному застосунку. Вони необхіжні для сворення інтерактивних завдань , щоб було цікавіше проходити лабіринт. Є декілька типів головоломок які реалізовані у застосунку , зокрема головоломка зі свілом та з послідовністю дій.

Якщо головоломка була активована перевіряються умови для успішного проходження. Після чого граця чекає відповіднийрезультат.

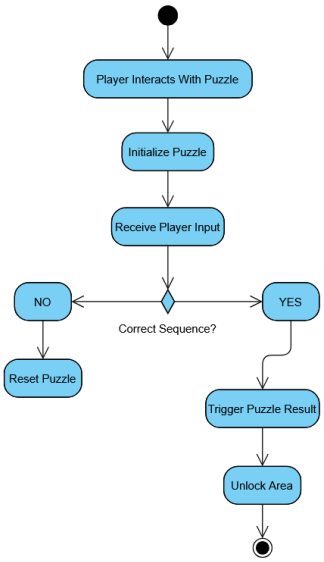

Рисунок 3.9 - Puzzle Activity Diagram

Як тільки головоломка пройде перевірку на успішність активується відповідна дія у разі успіху. Якщо ж головломка не пройдена або була неправильна послідовність дій , головоломка скидається до початкового стану.

Система прогресії

Задля комфортного проходження гри , реалізована система прогресії. Яка відповідає за закриття локацій з головоломками , щоб пройшовши гравець вже чітко розумів що він тут був. Також при проходжені головоломки надається карта, з частиною зони головоломки, і коли гравець вийде по мапі на головний шлях , цей прохід закриється . Незважаючи на те що буде закрит головний прохід, туди можна потрапити ще через інші , це вже потрібно дослджувати ходи лабіринту.

Таке буде відбуватися 2 рази, віжповідно така кількість записок та головоломок. Так закривши ці 2 зони та маючи мапу цих двох зон, гравець можне приблизно оцінити як йому пройти до виходу.

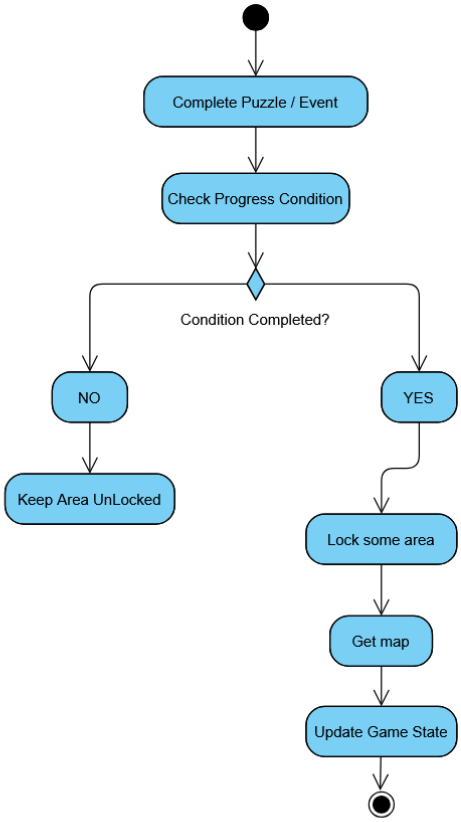

Рисунок 3.10 - Activity diagram системи прогресії

Дивлячись на activity diagram можна побачити що система прогресі перевіряє умови, якщо вони не виконані все залишається як було. Але якщо виконані, от як, наприклад головолмка завершена і відбувся scripted event надає карту гравцю цієї зони. Після виходу на головний шлях закриває проходи до цієї зони, але не всі. Це зроблено для того щоб гравець все таки міг заблукати, але і при цьому мав підказку дивлячись на мапу що він тут вже був чи приблизно зрозуміти куди йти, щоб вийти через фінальний прохід.

Scripted events

Scripted events потрібні задля реалізацій деяких механік. А саме звукового супроводу, спавну мінотавру, або для інтеракції з обєктами.

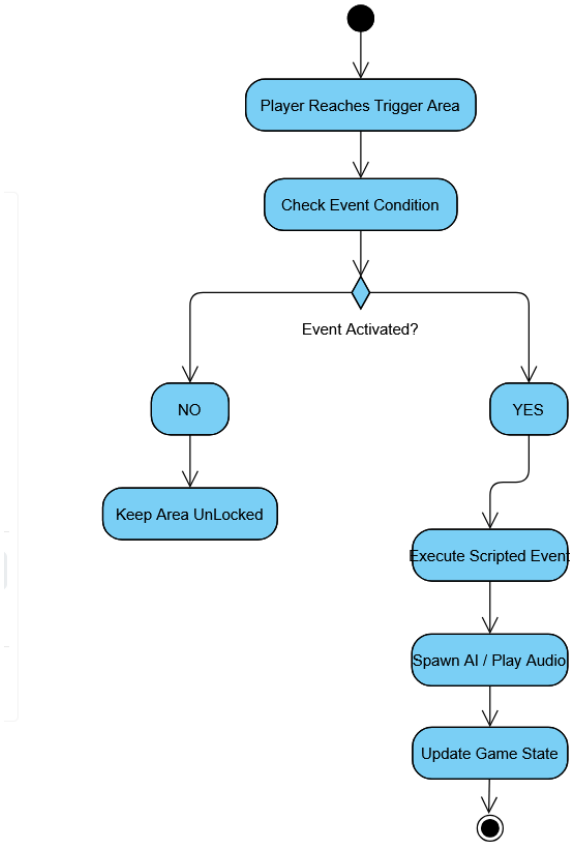

Рисунок 3.11 - Activity diagram для scripted events

Такі scripted events в основному реалізовані для спавну Мінотавра та звукових ефектів. Саме після того як гравець прочитає записки такі scripted events запускають погоню мінотавра та звукову напругу.

Але ще такі тригери розроблені для деяких об’єктів як от під час головоломок. Або для системи прогресії де перекриваються шляхи для гравця, та отримування мапи для персонажа.

Реалізація фінального вибору

Фінальна частина гри є одним з головних етапів. Формує завершення гри шляхом визначення кінцівки гравцем. Гравець має обрати надані вибори, його рішення має базуватися на інформації якій отримав під час проходження лабіринту. Весь наратив подається через записки які гравець має знайти та прочитати. З іншої сторони , гравець не може прийти з нестачею інформації так , як без прочитування обох записок гравець просто не зможе вийти з лабіринту.

Інформацію яку гравець знайде у записках, відображає персонажа , який потрапив до цього лабіринту через те , що був покараний богами. Але цього він не знатиме бо повністю вратив пам’ять. В записках гравець дізнаватиметься через що персонажа покарали. В результаті на основі такого наративу , гравцю потрібно буде вирішити фінал персонажа.

Гравцю будуть пропонуватися три варіанти вибору, кожен варіант має власну окрему кінцівку. Цей підхід реалізує елемент нелінійного наративу. Та підвищує цікавість у досліжені лабіринту.

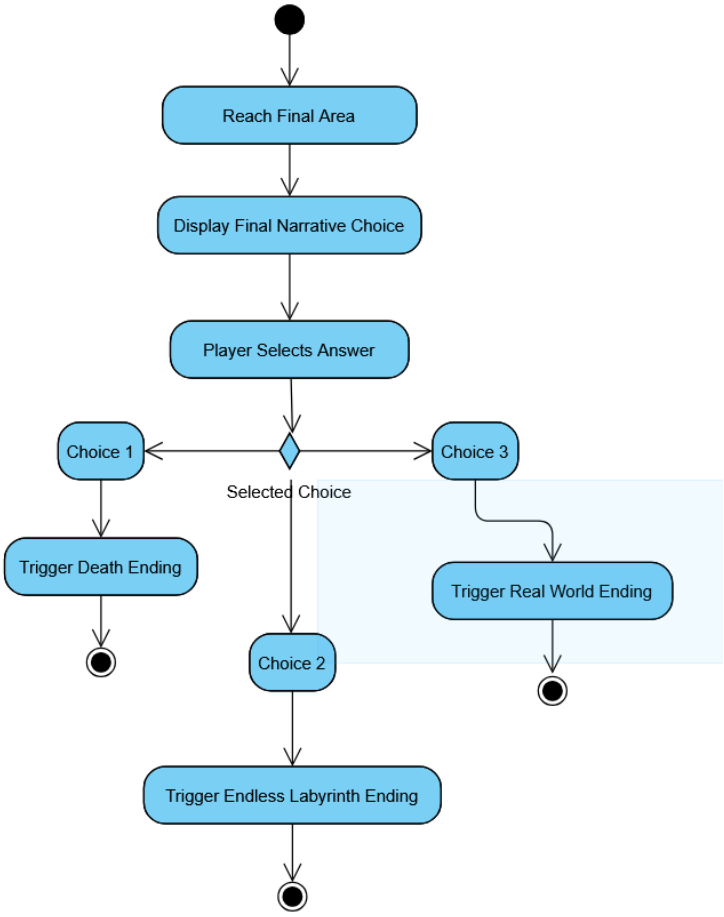

Рисунок 3.12 - Activity diagram для фінального вибору

Система вибору використовує наратив який гравець має дізнаватись по ходу проходження та дослідження лабіринту. На основі записок розкриватиметься сюжетна лінія , інформація на основі якої гравець прийматиме рішеня.

Перший варіант призводить до гибелі гравця. Другий варіант призводить до нескінченого проходження цього лабіринту. Третє повертає гравця у його світ, реальний світ.

3.4 Реалізація штучного інтелекту

Задля створення інколи тимчасової напруги на гравця, було реалізовано штучний інтелект у вигляді Мінотавру. Його задачею є створення відчуття у гравця небезпеки обмеження часту на обдумування дії.

Мінотавр має декілька станів, між якимми штучний інтелекст сам перемикається відповідно до ситуації або активованих тригер боксів.

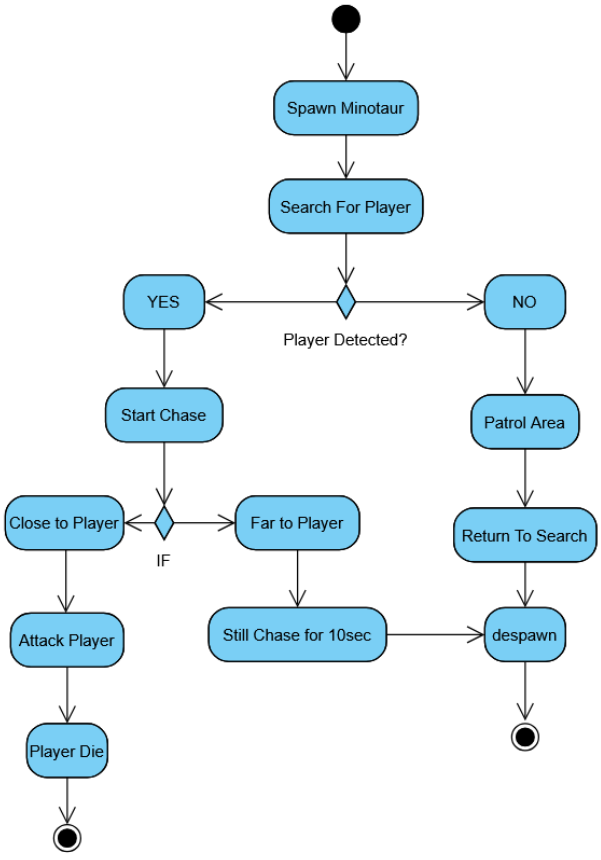

Рисунок 3.13 – Діаграма діяльності для штучного інтелекту

Задля механіки відчуття небезпеки у грі , реалізовано систему штучного інтелекту Мінотавра. Основною задачею Мінотавра є створення постійного

відчуття загрозу. Це призведе до психологічного тиску на гравця під час дослідження лабіринту у самий несподіваний момент для гравця.

Сам Мінотавр є динамічним елементом для геймплею, що робить його цікавим в плані гри. Він реагує на гравця та змінює свої стани залежно від ситуації поточного стану гру або від дій гравця.

Поведінка станів AI має чітко визначений набір. Штучний інтелект сам вирішує залежно від ситуації у ігровому світі , який стан потрібно обрати на даний момент. Цей підхід створює контрольоване напруження та надає розуміння , що непердбачуваності або прошення завдання немає відбутися.

Спавн самого звіра відбувається тільки і тільки через scripted events, на початку гри Мінотавра ще не існує у цьому левелі гри. Це рішення є коректним з точки зору дизайну самої гри та є свідомим рішенням. Спочатку гравець адаптується до дослідження лабіринту, те що все спокійно , нічого не відбувається окрім пошуку виходу та виконання квестів. Після того як гравець прочитає записку відбувається відповідний scripted events що спавнить поряд Мінотавра у завчасно визначеній точці лабіринту. Місце обрано безпосердньо так щоб це не було прямо навпроти граця, це б дуже порушивало імерсивність.

Як тільки звір з’явився у лабіринті має застосуватись стан патрулювання задля пошуку гравця. У цьому стані використовується система навігації Unreal Engine 5 . Під час патрулювання , швидкість звіра є не великою, відбувається імітація помірного патрулювання та пошуку цілі.

Коли гравця виявлено AI має перемикатись у стан переслідування. З цим допомагає AIController , який має передавати де знаходиться гравець та оновлювати дані цілі. Будує оптимальний маршрут з урахуванням геометрії рівня лабіринту. Також головним є те , що швидкість під час пересідування помітно збільшується і коли гравець це бачитиме, відчуавтиме небезпеку та напругу під час втечі, але на поворотах швидкість звіра зменшується, тому грацю реально втекту від Мінотавра.

Важливим є ще й те , що у граця немає ніякої зброї чи механік окрім бігу для супротиву Мінотавру. Це було спроєктовано навмисно для усунення загрози

Мінотавру від гравця, адже в лабіринті він має бути непереможний. Його не можливо знищити чи заблокувати , можливо тільки втекти та сподіватись що він більше не переслідує. Сам гравець має усвідомлювати що він не безсмертний і може поверти в лабіриті та потрібно буде проходити з останнього збереження. Що відповідно змушує діяти гравця обережно на певний момент гри, навіть якщо звіра немає поки в лабіринті. Тому після якихось скріптів гравець може вже очікувати зі страхом на Мінотавра , а його і не буде.

Також важливим є аудіо складова , так як від цього і залежить уся імерсивність цієї механіки. Кроки , рик та гуркіт Мінотавра , усе це формує враження граця і має бути виконано у найвищій формі. Також аудіо дизайн дозволяє гравцю розуміти наскільки він відірвався під час втечі або наскільки близько до персонажа звір. Відповідно гучність аудио дизайну регулюється через MetaSounds з просторовим звуком. Також при активації стану переслідування звіра застосовується легкий візуальний евект на камері гравця через PostProcessVolume, що теж має підсилювати імерсивність , небезпеку та напругу.

З технічної сторони , логіка поведінки Мінотавра інкапсульована в окремомому класі BP_Minotaur_AIController, це відповідає вже описаному модульному принциму архітектури проєкти раніше. Відповідно такий принцип дозволяє змінювати будь-які параметри , такі як швидкість, радіус чи реакція на гравця без необхідності змінювати логіку у інших системах. Все це спрощує загалом як і балансування станів Мінотавра , складності оскільки все це можна легко скоригувати.

3.5 Вибір технологій та компонентів

У цьому підрозділі буде обґрунтовано вібір технологічного стеку для розробки ігрового десктоп-застосунку в жанрі adventure на основі рушія Unreal Engine 5. Всі рішення приймалися з урахуванням різних вимог як от до персональних можливостей так і до продуктивності чи візуальної якості. Обрані

інструменти дозволять забезпечити чудові технічні можливості та ефективний процес самої розробки.

Unreal Engne 5

Обраний як основний рушій для розробки застосунку - Unreal Engine 5. Якщо порівнювати з конкурентам таким як Unity, UE5 має великі переваги у графіці та деяким технологіям такими як Nanite та Lumen, які не присутні у Unity без використання стороніх налаштувань.

Також головним фактором є те , що рушій є безкоштовним, як і усі інструменту у рушії. Та і зараз є найпопулярнішим у індустрії розробці сучасних ігор як великих так й інді. Це ще й дозволяє набути навичок які можуть знадобитися у майбутній професійній діяльності.

Blueprints та C++

Ігрова логіка реалізована в основному через Blueprints , так як ця система відуального програмування є досить простою у користуванні. Але й застосовувалися знання у C++ так , як було вирішено написати деякі модулі саме мовою програмування, задля зручності керування та оптимізації процесу. Blueprint в основному використовується для генерації самого лабіринту, деяких гейплейних логік та тригерів і головоломок. C++ же використовується для реальзації персонажа , збережень, штучного інтелекту , тому що тут важливий чіткий потроль на пам’яттю. А занадто багато blueprints можу навантажувати ігровий процес. Такий підхід є чудов під час розробки подібних проєктів так як поєднує в собі швидкість з технічністю.

Nanite

Технологія Nanite - віртуалізована геометрія, дозволяє мати на сценах 3д моделі з багатьма полігонами . Звісно мільйон полігонів на 1 дерево це сильно сильно багато, але можна створити наприклад ме з 10к полігонів , який є досить детальним, і для нього можна не робити LOD , так як Nanite сам коригує відображення об’єкту.

Lumen

Система глобального освітлення Lumen даєє динамічний розрахунок непрямого світла у реально часі без запікання світла. Для гри з лабіринтом я вважаю це є важливою технологією задля створення імерсивності та реалістичного відображення взаємодію світтля зі світом граця.

Unreal Motion Graphics

Використовується для розробки інтерфейсу корисувача. UMG є частиною вбудованих інструментів UE5. UMG надає редактор для створення UI , з анімаціями та адаптивністю під різну роздільну здатність екрану. У застосунку реалізовано головне меню, HUD, відображення записок.

3.6 Опис інтерфейсів

Інтерфейс є одним з важливих компонентів для першого враження при заході у гру. Також забезпечує розуміння між гравцем та грою з імерсивністю. У програмному застосунку спроєктовано мінімалістичний HUD. Більша частина інформації подається через інші компоненти світу. Увесь UI реалізований за допомогою Unreal Motion Graphics та адаптовано під різну роздільність екрану.

Головне меню

Головне меню це перше враження гравця , і воно має надати ще більшу зацікавленість гравцю. Реалізовано як окрема сцена з динамічним показом усього лабіринту з горами та деревами. Меню містить чотири основні кнопки : **Нова гра** для початку проходження гри, **Завантажити** задля завантаження збереження процесу, **Налаштування** для зммін графіки та аудіо і також роздільності екрану, **Вихід** відповідно дозволяє вийти та повністю закрити гру. Візуально це оформлено загалом у темних тонаї задля атмосфери , щоб сформувати певний емоційний настрій.

Рисунок 3.14 - Головне меню

HUD

Ігровий HUD є дуже простий , якщо увімкнено то по центру завжди є біла крапка, якщо вимкнено, то при бігу з’являється полоса витривалості для спринту. Також є підказка з іконкою у вигляді букви E , там присутній короткий опис та з’являється відповідно лише коли гравець достатньо близько розташований. Цей підхід ідеально підходить , якщо хочеться уникнути візуального перевантаження на екрані та зберігти атмосферу імерсивності під час гуляння в лабіринті.

Рисунок 3.15 - HUD програмного застосунку

Інтерфейс записок

Коли гравець взаємодіє ат підіймає записку для читання на карті відображається UI записка посередині екрану і ззаду затемнений фон. Записка імітує вигляд жовтого аркушу паперу з текстом. Гравець може перглядати запуску та інколи за потреби перевертати на іншо сторону.Під час читання ігровий процес не зупиняється.

Висноки до розділу 3

У третьому розділі виконано повний обсяг робіт з архітектури проєктування програмного забезпечення, моделювання а також конструювання програмного забезпечення відповідно до вимог сформованих у другому розділі.

У підрозділі 3.1 розроблено модульну архутектуру застосунку на основі принципу Component-Based Design. Основні модулі описані та визначені. PCG відповідає за генерацію лабіринту, Puzzle System реалізує логіку для головоломок. Також є поведінка Мінотавра за допомогою штучного інтелекту, та збереження з інтерфейсом.

У підрозділах 3.2 та 3.3 детально описано вже реалізовані ігрові механіки : керування персонажем з взаємодіями, ліхтар , витривалість, систему головоломок з перевіркою умов проходження, scripted events, механіка фінального виборо з трьома варіантами кінцівки. До кожної наведено UML activity diagram та детальне пояснення роботи з призначеням.

У підрозділі 3.5 обґрунтовано вибір усього технологічного стеку. Обґрунтовано використання UE5 як рушій та визначенно застосування Blueprints та C++ . Було описано роль кожної технології у проєкті, як от Nanite для об’єктів чи Lumen для кращого освітлення.

У підрозділі 3.6 описано та обґрунтовано інтерфейс застосунку. Головне меню, HUD, відображення записок. Відображено що інтерфейс спроєктований для мінімального втручання у процес гри .

4 ТЕСТУВАННЯ ТА КЕРВІНИЦТВО КОРИСТУВАЧА

4.1 Кодування та тестування програмного забезпечення

У цьому підрозділі описано основні програмні класи застосунку, їх структуру, змінні та ключові методи. Реалізація застосунку виконана з використанням гібридного підходу: Blueprint Visual Scripting для геймплейної логіки та C++ для базових компонентів і систем що вимагають підвищеної продуктивності. Весь програмний код організований відповідно до модульної архітектури описаної у розділі 3.

Реальзація механік гравця

Лістинг коду LabyrintosCharacter.h:

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Camera") class UCameraComponent FirstPersonCamera;*

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Flashlight") class USpotLightComponent Flashlight;*

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement") float WalkSpeed = 300.f; UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement") float RunSpeed = 600.f;

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Stamina") float MaxStamina = 100.f;

UPROPERTY(BlueprintReadOnly, Category = "Stamina") float Stamina;

Макроси такі , як **UPROPERTY** визначає як рушій взаємодії зі змінними. **EditAnywhere** надає змінівуати значення в редакторі без необхідності перекомпіляції. **BlueprintReadWrite** дозволяє бачити змінну у Blueprint-графі, наприклад , UI-шкала витривалості може читати значення Stamina і відображите

без необхідності якогось додаткового коду. **VisibleAnywhere** дозволяє бачити змінні у панелі деталей в редакторі.

Повна реалізація міститься у файлі **LabirinthosCharacter.cpp** , де присутні усі методи класу персонажа. Кожен метод має виконувати лише одну задачу, це було зроблено для спрощення подальшої модифікації коду.

Конструктор класу виконується лише одиного разу. Компонент **UcameraComponent** прикріпляється до голови скелетного мешу персонажа. Компонент **USpotLightComponent** прикріпляється саме до камери , через це він завжди направлений туди, куди дивитиметься гравець. Параметр **SetIntensity** задає відповідно яскравість віжповідно до освітлення Lumen. Метод **BeginPlay** також викликається лише на початку гри , коли вже усі актори у світі були заспавнені та ініціалізовані.

Початкову швидкість для персонажа встановлюється через **GetCharacterMovement() -> MaxWalkSpeed**. Метод **Tick** викликається рушієм відповідно кожен кадр і він отримує параметр **DeltaTime**. Саме так реалізована логіка витривалості. Тобто витривалість у нас оновлюється кожен тік, також присутня перевірка **Stamina < 5.f** у **StartRun** для запобігання миттєвого активування бігу після того як витривалість була вичерпана. Методи витривалості такі як **DrainStamina** та **RegenerateStamina** використовують функцію **Fmath::Clamp** , вона гарантує що діапазон в даному випадку завжди буде від 0 до максимального значення **MaxStamina**. Без Clamp з витривалість могла б не правильно прораховуватись, тобто витривалість могла набувати від’ємного значення або перевищувати допустиме. Швидкість витрати після тестування встановлення як 15.f на секунду , а відновлення вже 8f на секунду. Це дає оцінку , що використовувати безперервно біг можна приблизно 8 секунд, в той час як на відновлення потрібно приблизно 12 секунд.

Interact це метод , який реалізує саме виявлення об’єктів, які є інтерактивними для гравця через лінійнене трасування **LineTraceSingleByChannel**. Присутній параметр **ECC_Visibility** , який взаємодіє тільки з об’єктами які видимі для камери, це запобігає взаємодії з об’єктами крізь стіни чи інші об’єкти. Перевірка через **Implements<ULabyrinthosInteractInterface>()** та виклик **Execute_OnInteract** є реалізацією “Interface” – тобто пересонаж є повністю відокремленим від конкретних об'єктів з якими взаємодіє.

ToggleFlashlight – це метод, який відповідає за ліхтар. **bFlashlightOn** – це булева змінна, **!bFlashlightOn** інвертує поточний стан ліхтаря, за допомогою **SetVisibility(bFlashlightOn)** світло вмикається або вимикається.

Лістинг коду LabyrinthosInteractInterface.h:

```
UINTERFACE(MinimalAPI, Blueprintable)
class ULabyrinthosInteractInterface : public UInterface
{
    GENERATED_BODY()
 };
class LABYRINTHOS_API ILabyrinthosInteractInterface
{
    GENERATED_IINTERFACE_BODY()
public:
```

UFUNCTION(BlueprintNativeEvent, BlueprintCallable, Category = "Interaction")

void OnInteract(ACharacter Interactor);*

};

Цей файл оголошує інтерфейс взаємодії , який реалізують усі інтерактивні об’єкти у програграмному застосунку. **BlueprintNativeEvent** на методі OnInteract каже про те , що поведінку можна перевизначити й у Blueprint-графі конкретного об’єкту. Завдяки цьому реалізовани різні Blueprint у який використовується OnInteract , але кожен реагує на взаємодіє по-різному.

Першим з таких Blueprint є **BP_Note**. Це граф для відображення записок, які знаходитиме гравець протягом гри. Для цього був написаний хедер файл , де були описані потрібні змінні. А основою є короткий **cpp:**

AlabyrinthosNote::AlabyrinthosNote()

{

PrimatyActirTick.bCanEverTick = false;

}

void AlabyrinthosNote::OnInteract_Implementation(Acharacter Interactor)*

{ }

Метод **OnInteract_Implementation** є порожнім , так як сама логіка реалізується через blueprint-граф.

Клас BP_PuzzleBase

BP_PuzzleBase є базовим класом для всіх головоломок гри та містить спільну логіку перевірки умов завершення. Ключова змінна bIsSolved типу Boolean зберігає стан головоломки, використовується як константа , може змінитися лиша один раз , якраз при успішнуому вирішенні головоломки. Функція CheckSolution викликається після кожної дії гравця та перевіряє поточний стан елементів головоломки відносно правильної відповіді. Використовується для головоломки з цеглинами, викликається після кожного натискання цеглини і перевіряє на правильність, тобто чи у правильному порядку була натиснута цеглина. Якщо ж гравець натисне неправильну , відповідно йдучи не за правильним порядком – усі цеглини які були натиснуті повертаються у свою початкову позицію, відповідно увесь прогрес головоломки скидається. Але якщо порядок правильний, викликається **OnPuzzleSolved.**

Відповідно **OnPuzzleSolved** активує **scripted event** який має відбутися після успішного виконання цієї головоломки та відбудеться збереження гри через **BP_SaveGame**.

Клас BP_SaveGame

BP_SaveGame – це клас , який успадковується від базового класу USaveGame рушія та містить всі дані необхідні для збереження та відновлення стану гри.

Таблиця 4.1 — Змінні класу

Змінна	**Тип**	**Призначення**
PlayerLocation	FVector	Позиція гравця у світі
BlockedPassages	Tarray<int32>	Індекси перекритих проходів у лабіринті
SolvedPuzzlesCount	Int32	Кількість вирішених головоломок
ReadNotesCount	Int32	Кількість прочитаних записок

Саме збереження завжди відбувається автоматично після кожного важливого моменту. Збереження виконується через виклик функції SaveGameToSlot з іменем слоту та індексом гравця, тобто гравець немає контролю над моментом збереження програмного застосунку.

BlockedPassages це масив цілих чисел , де кожне число є відповідним індексом перекритого проходу у лабіринті. При завантаження гравця у гру через **LoadGameFromSlot** система має пройтися по цьому масиву і перекрити відповідні проходи у лабіринті.

При завантаженні гравця у лабіринт PlayerLocation передається напряму у SetActorLocation актора. А SolvedPuzzlesCount та ReadNotesCount відновлюються відповідні стани.

4.2 Тестування програмного застосунку

Тестування розробленого програмного застосунку проводилось відповідно до мети тестування, а саме перевірки коректності реалізації всіх ігрових механік та відповідності функціональним вимогам визначеним у другому розділі. Використовувались два методи тестування: верифікація для перевірки технічної коректності реалізації кожної функції та валідація для підтвердження відповідності застосунку вимогам специфікації.

Перше – це верифікація, яка проводилась на рівні окремих методів або компонентів. Під час тестування перевірялося коректність роботи системи витривалості, в основному це були моменти коли витривалість повністю вичерпувалался та миттєвого бігу без витривалості. Також тестувалась система лінійного трасування для методу Interact, перевірялось саме коректність об’єктів , також неможливість взаємодіяти з ними крізь стіни, а також чи бачить система ці об’єкти аби передати цю інформацію іншим скриптам. Збереження та завантаження

перевірялися шляхом перезапуску програмного застосунку після різних моментів автоматичного зберережння гри.

Відповідно друге – це валідація , яка проводилась шляхом повного проходження застосунку 5 разів, для перевірки на відповідність як правильної поведінки різних механік та на очікавані результати. Для правильної валідації, тестування проводилось не як звичайне проходження, окрім такого плану, інші були направленні на нестандартні варіанти проходження для виявлення багів у роботі програмного застосунку.

Тестування продуктивності застосунку проводилось з використанням відповідних вбудованих інструментів Unreal Engine 5: такі команди як – stat fps та stat unit для отримання даних у реальному часі, а також інструменту GPU Profiler для аналізу навантаження на відеокарту. Тестування виконувалось на обладнанні з процесором AMD Ryzen 2700x, відеокартою NVIDIA GeForce RTX 2070 та 32 ГБ оперативної пам'яті при роздільності 1920×1080.

Таблиця 4.2 — Результати вимірювань продуктивності при різних налаштуваннях якості

Налаштування	**FPS середній**	**FPS мінімальний**	**Пам'ять (МБ)**
Low	120+	95	2100
Medium	87	68	2985
High	58	44	3900
Epic	38	29	5100

Аналіз результатів показує що основним bottleneck застосунку є GPU, а не CPU, на сьогодні це є типовою ситуацією для графічно насичених програмних

застосунків з увімкненими технологіями Lumen та Nanite. На налаштування Low застосунок має середній FPS який перевищує 120, з мінімумом у 95 FPS, це відповідно забезбечує більш ніж комфортну роботу застосунку на слабшому обладнанні. Налаштування такі , як Medium та High вже суттєво відрізняються, але застосунок все ж демонструє комфортні для проходження гри показники продуктивності, маємо на Medium налаштуваннях середній показник у 87 FPS та мінімальним у 58 FPS ,що вже відповідає стандарту та є прийнятним, також можу наголосоти що на налаштуваннях High під час проходження більшої частини гри присутній стабільні 60 FPS, але звісно в деякий моментах є просадки у 44 FPS , які гравець наврядчи відчує під час проходження.

На максмальних налаштуваннях, Epic, навантаження на GPU зростає суттєво на обладнанні яке використовувалось для тестування, це виникає в більшості через технологію Lumen – повне трасування променів з великою кількістю об’єктів як рослиності так і загалом в лабіринті сильно навантажує GPU для обрахування та малювання усіх тонкостей які пропонує ця технологія. Через що маємо зниження середнього показника до 38 FPS що є прийнятним для таких налаштування, на такому обладнанні.

Стосовно опертивної пам’яті , в залежності від налаштування, але зростає від 2100 мегабайт на Low до 5100 мегабайт на Epic, що для максимальних налаштування є ще дуже оптимізовано. Загалом такий об’єм пов’язаний з розміром текстур які завантажуються у відопам’ять відповідно до рівня деталізації зазначеного у налаштуваннях.

Отимізація – важливий крок, для цього було застосовано наступні рішення: обмеження відстанні відображження об’єктів за допомогою **CullDistance** , це дозволяє не рендерити грі об’єкти як занадто далеко. Таке рішення є обов’язковим короком в оптимізації. Зменшення кількості динамічного світла на рівні та

використання Virtual Shadow Maps замість звичайний тіней, забезпечує більш кращу якість при вже відносно меншому навантаженні.

4.3 Керівництво користувача

У цьому розділі описано керівництво користувача. Тобто будуть описані кроки для повного проходження гри.

Рисунок 4.1 – Головне меню

Гравець починає з завантаження гри та потрапляння у головне меню програмного застосунку. На початковому єкрані гравець може зайти в налаштування та обрати потрбні параметри. Для початку гри треба натиснути кнопку Start , після чого гравець буде вантажитись на основну мапу, Лабіринт.

Рисунок 4.2 – Початкова позиція гравця

Гравець з’являється на початку лабіринту , звідки починається шлях проходження гри. У перші хвилини гравець чує голос , який каже що гравцю потрібно зробити аби звідси вибратися.

Рисунок 4.3 – Перша головоломка

Блукаючи лабіринтом , гравець обов’язково знайде одну з двох головоломок які потрібно пройти для проходження гри, на рисунку 4.3 показана перша головоломка, а саме світло, потрібно використовуючи ліхтар знайти усі потрібні букви , вони проявляються тільки після того як на них попаде світло з ліхтаря. Коли усі символи будуть знайдені , заспавниться записка з історією , з якою гравцю треба ознайомитися.

Рисунок 4.4 – Поява записки

Записка з’являється на дереві, гравець має прочитати її для ознайомлення з історією персонажа. Після чого на гравця чекає scripted event.

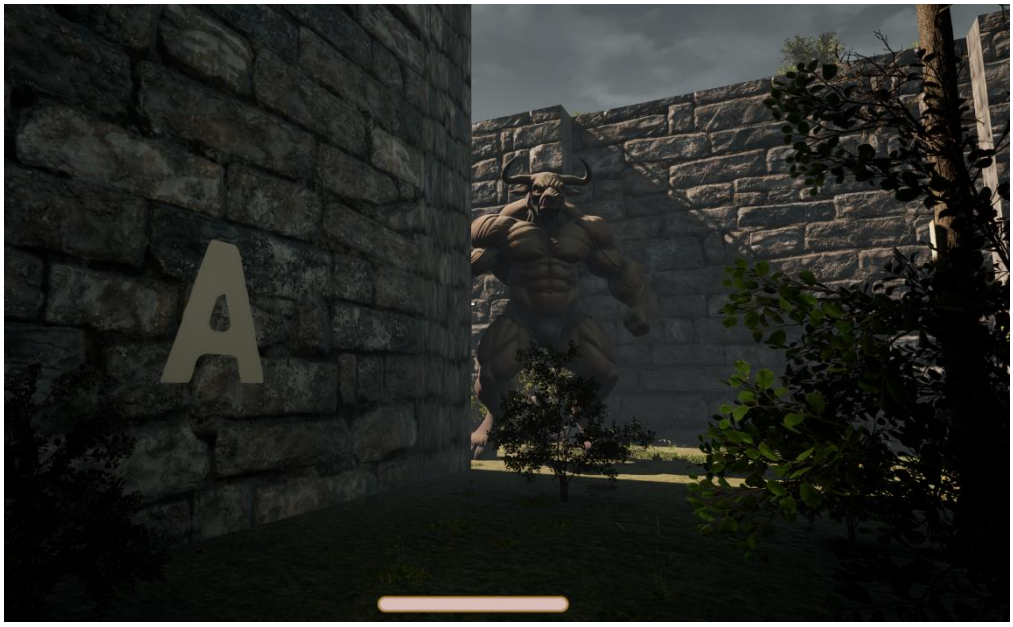

Рисунок 4.5 – Поява Мінотавра

Перший scripted event , поява Мінотавра, прочитавши записку гравець дізнається інформацію, щоб ускладнити та зробити геймплей у лабіринті більш напруженим , після кожної прочитаної записки поруч з гравцем буде з’являтися Мінотавр, який шукатиме гравця та переслідуватиме його деякий час поки не втратить з поля зору.

Рисунок 4.6 – Друга головоломка

Блукаючи лабіринтом , гравець після першої головоломки обов’язково натрапить на другу і відповідно останню головоломку. Гравцю потрібно буде у правильному порядку натиснути на цегли для проходження головоломки, адже поки він тут , усі шляхи виходу заблоковані, саме тому гравцю треба пройти цю головоломку , а також задля отримання останньої записки. Після чого гравець матиме змогу вийти з лабіринту. Але звісно після вирішення цієї головоломки , з’явиться Мінотавр , який в цей раз буде довше переслідувати гравця.

Рисунок 4.7 – Вихід з лабіринту

На виході з лабіринту гравця чекають високі сходи, піднявшись на гору гравець втратить свідомість. Звідси починається фінальний вибір гравця , який вирішить долю персонажу.

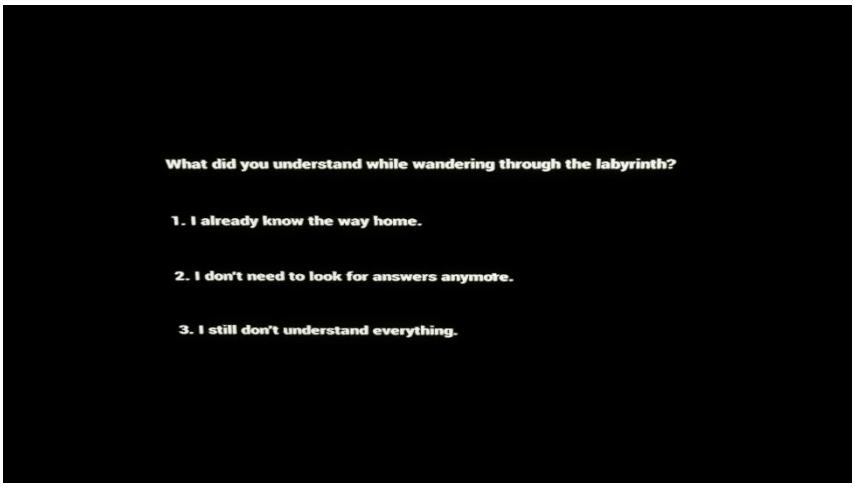

Рисунок 4.8 – Фінальний вибір

Фінал гри зроблений так, що у гравця є 3 варіанти вибору. Отримуючи наратив з записок гравець має зрозуміти чому та через що покарали персонажа.

Перший вибір – це показуватиме , що гравець не до кінця зрозумів покарання персонажа та його історію, через що персонаж буде блукати лабіринтом вічно.

Другий вибір – персонаж помирає після блукання лабіринтом. Гравець не зрозумів моралі осторії через , що втрачає персонажа. Також можливо , що гравець ознайомившись з історією персонажа вирішив, що той робив усе правильно і не заслуговує покарання.

Третій вибір – персонаж повертається у світ де народився. Гравець правильно зрозумів мораль історії, ознайомився та зрозумів історії персонажа та його покарання. Пройшовши лабіринт персонаж усвідомив свої помилки минулого та замислюється про майбутнє.

Висновки до 4 розділу

В ході тестування програмного застосунку усі баги або невідповідні очікуванням результати були виправлені. Продуктивність програмного застосунку було оцінено як задовільно. Користувача чекає комфортний геймплей з напруженням. Також було розроблено керівництво користувача , де показано та описано кокретні ключові події , які потрібні для проходженя програмного застосунку.

Висновки

Під час виконання кваліфікаційної бакалаврської роботи було розроблено ігровий десктоп-застосунок в жанрі adventure на основі рушія Unreal Engine 5 який має назву Labyrinthos.

У першому розділі було проведено аналіз сучасного стану ігрової індустрії, також досліджено особливості жанру програмного застосунку та проведено аналіз існуючих аналогів. Основу геймплею визначено за ключовими характеристиками жанру програмного застосунку, а саме : дослідження лабіринту, наратив через записки та головоломки, додаткове напруження через взаємодію з Мінотавром.

У другому розділі сформофані функціональні та нефункціональні вимоги для застосунку , також було обгрунтовано вибір інструментарію для розробки. Unreal Engine 5 з такими технологіями як Lumen та Nanite є найкращим вибором для реалізації програмного застосунку, відповідно створено атмосферне середовище лабіринту з динамічним освітленням.

У третьому розділі описано виконання моделювання та саму реалізацію застосунку. Спроєктовано архітектуру застосунку. Реалізовано систему керування персонажем , що включає мехінки ходьбі, біг та витривалість. Реалізовано систему взаємодії з об’єктами та головоломки двох типів – світлова та послідовна. AI реалізований через Behaviot Tree , також присутнє аудіо формлення геймплею та виконано систему збереження прогресу.

У четвертому розділі описано реалізацію ключових класів програмного застосунку з використанням як C++ так і Blueprint-графів. Виконано тестування з верифікацією та валідацією. Результати тестування є кофортими , що означає оцінку , як задовільно.

В підсумку , результатом робити є реалізований програмний застосунок який є функціональним для реалізації гемплею. Застосунок можу бути використай як основу для реалізації більшого масштабу у майбутньому. Програмний застосунок демонструє навички з застосування сучасних технологій для розробки на база UE5.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Doe J. Unreal Engine 5 Requirements - Catness Game Studios. *Catness Game Studios*. URL: https://catnessgames.com/blog/unreal-engine-5-requirements/ (date of access: 25.04.2026).

2. Unreal Engine 5 - Starloop Studios. *Starloop Studios*. URL: https://www.starloopstudios.com/ebook/unreal-engine-5/ (date of access: 18.04.2026).

3. LI Z. G., Roberts E. W. Unreal Engine 5 Game Development with C++ Scripting: Become a Professional Game Developer and Create Fully Functional, High-Quality Games. de Gruyter GmbH, Walter, 2023. 384 c.

4. Unreal Engine 5.5 documentation. *Epic Games*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-documentation (date of access: 24.04.2026).

5. Nanite as a disruptive technology for the interactive visualisation of cultural heritage 3D models: A case study. URL: https://doi.org/10.3390/heritage6080295 (date of access: 21.04.2026).

6. Li, Tianshu. Real–time performance comparison of environments created using traditional geometry rendering versus unreal nanite technology in virtual reality. MS thesis. Purdue University, 2024. 47 c. URL: https://doi.org/10.25394/PGS.25676631 (date of access: 23.04.2026)

7. Skorobogatova A. Real-Time Global Illumination in Unreal Engine 5 : thesis / Faculty of Informatics, Masaryk University. 2022. 66 p. URL: https://is.muni.cz/th/n1qq4/real-time_GI_in_UE5.pdf

8. DENG, Zhian; SHI, Minhua; CHEN, Yuxuan. Research on UE5 Pathfinding Based on Improved A* Algorithm. In: *2025 10th International Symposium on Advances in Electrical, Electronics and Computer Engineering (ISAEECE)*. IEEE, 2025. p. 462-466. (date of access: 21.04.2026)

9. Літвінова, О. О. "Розробка RPG-гри в Unreal Engine 5 із застосуванням засобів штучного інтелекту для моделювання поведінки некерованих персонажів." (2025). (date of access: 16.04.2026)

10. Romero, Marcos, and Brenden Sewell. *Blueprints Visual Scripting for Unreal Engine 5: Unleash the true power of Blueprints to create impressive games and applications in UE5*. Packt Publishing Ltd, 2022. 531 p. (date of access: 10.04.2026)

11. Sang, YuanZi, and KiHong Kim. "How to Optimize Tilling Effect of Environmental Materials in UE5." *International Journal of Internet, Broadcasting and Communication* (2025): 119-129. (date of access: 15.04.2026)

12. PAJERSKY, BC RICHARD. "Procedural Level Generation for Unreal Engine 5." (date of access: 22.04.2026)

13. Tan, Tiow Wee. "Getting Started with Unreal Engine 5." *Game Development with Unreal Engine 5 Volume 1: Design Phase*. Berkeley, CA: Apress, 2024. 1-22. (date of access: 21.03.2026)

14. Koivisto, Olivia. "Creating a 3D environment using Unreal Engine 5.1." (2023). (date of access: 12.03.2026)

15. Butler, Stuart, and Tom Oliver. *Game Development Patterns with Unreal Engine 5: Build maintainable and scalable systems with C++ and Blueprint*. Packt Publishing Ltd, 2024. 232 p.(date of access: 09.02.2026)

ДОДАТКИ

LabirinthosCharacter.cpp

```
#include "LabyrinthosCharacter.h"
#include "Camera/CameraComponent.h"
#include "Components/SpotLightComponent.h"
#include "GameFramework/CharacterMovementComponent.h"
#include "Engine/World.h"
#include "LabyrinthosInteractInterface.h"
ALabyrinthosCharacter::ALabyrinthosCharacter()
{
    PrimaryActorTick.bCanEverTick = true;
    FirstPersonCamera = CreateDefaultSubobject<UCameraComponent>(
        TEXT("FirstPersonCamera"));
    FirstPersonCamera->SetupAttachment(GetMesh(), FName("head"));
    FirstPersonCamera->bUsePawnControlRotation = true;
    Flashlight = CreateDefaultSubobject<USpotLightComponent>(
        TEXT("Flashlight"));
    Flashlight->SetupAttachment(FirstPersonCamera);
    Flashlight->SetIntensity(5000.f);
    Flashlight->SetOuterConeAngle(35.f);
```

```
    Stamina = MaxStamina;
}
void ALabyrinthosCharacter::BeginPlay()
{
    Super::BeginPlay();
    GetCharacterMovement()->MaxWalkSpeed = WalkSpeed;
}
void ALabyrinthosCharacter::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    if (bIsRunning)
        DrainStamina(DeltaTime);
    else
        RegenerateStamina(DeltaTime);
}
void ALabyrinthosCharacter::SetupPlayerInputComponent(
    UInputComponent* PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);
    PlayerInputComponent->BindAxis("MoveForward", this,
```

```
        &ALabyrinthosCharacter::MoveForward);
    PlayerInputComponent->BindAxis("MoveRight", this,
        &ALabyrinthosCharacter::MoveRight);
    PlayerInputComponent->BindAxis("LookUp", this,
        &ALabyrinthosCharacter::LookUp);
    PlayerInputComponent->BindAxis("LookRight", this,
        &ALabyrinthosCharacter::LookRight);
    PlayerInputComponent->BindAction("Run", IE_Pressed, this,
        &ALabyrinthosCharacter::StartRun);
    PlayerInputComponent->BindAction("Run", IE_Released, this,
        &ALabyrinthosCharacter::StopRun);
    PlayerInputComponent->BindAction("Interact", IE_Pressed, this,
        &ALabyrinthosCharacter::Interact);
    PlayerInputComponent->BindAction("Flashlight", IE_Pressed, this,
        &ALabyrinthosCharacter::ToggleFlashlight);
}
void ALabyrinthosCharacter::MoveForward(float Value)
{
    if (Value != 0.f)
        AddMovementInput(GetActorForwardVector(), Value);
```

```
}
void ALabyrinthosCharacter::MoveRight(float Value)
{
    if (Value != 0.f)
        AddMovementInput(GetActorRightVector(), Value);
}
void ALabyrinthosCharacter::LookUp(float Value)
{
    AddControllerPitchInput(Value);
}
void ALabyrinthosCharacter::LookRight(float Value)
{
    AddControllerYawInput(Value);
}
void ALabyrinthosCharacter::StartRun()
{
    if (Stamina < 5.f) return;
    bIsRunning = true;
    GetCharacterMovement()->MaxWalkSpeed = RunSpeed;
}
```

```
void ALabyrinthosCharacter::StopRun()
{
    bIsRunning = false;
    GetCharacterMovement()->MaxWalkSpeed = WalkSpeed;
}
void ALabyrinthosCharacter::DrainStamina(float DeltaTime)
{
    Stamina = FMath::Clamp(Stamina - 15.f * DeltaTime, 0.f, MaxStamina);
    if (Stamina <= 0.f)
        StopRun();
}
void ALabyrinthosCharacter::RegenerateStamina(float DeltaTime)
{
    Stamina = FMath::Clamp(Stamina + 8.f * DeltaTime, 0.f, MaxStamina);
}
void ALabyrinthosCharacter::Interact()
{
    FVector Start = FirstPersonCamera->GetComponentLocation();
    FVector End = Start +
        FirstPersonCamera->GetForwardVector() * InteractRange;
```

```
    FHitResult HitResult;
    FCollisionQueryParams Params;
    Params.AddIgnoredActor(this);
    bool bHit = GetWorld()->LineTraceSingleByChannel(
        HitResult, Start, End, ECC_Visibility, Params);
    if (bHit && HitResult.GetActor())
    {
        AActor* HitActor = HitResult.GetActor();
        if (HitActor->Implements<ULabyrinthosInteractInterface>())
        {
            ILabyrinthosInteractInterface::Execute_OnInteract(
                HitActor, this);
        }
    }
}
void ALabyrinthosCharacter::ToggleFlashlight()
{
    bFlashlightOn = !bFlashlightOn;
    Flashlight->SetVisibility(bFlashlightOn);
}
```