Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему:
«Генерація адаптивних діалогів NPC з використанням штучного інтелекту в інтерактивній детективній грі на платформі Unity»

Виконав: студент VI курсу групи КН-61м
спеціальності 122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)
Марущак С. І.
(прізвище та ініціали)

Керівник Сторожук О. Л.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувачка кафедри КН
_____ Борецька І.Б.
"10" грудня 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Марущаку Сергію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Генерація адаптивних діалогів NPC з використанням штучного інтелекту в інтерактивній детективній грі на платформі Unity

керівник роботи *Сторожук Олександр Леонідович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "29" квітня 2025 р. № С-288
2. Термін подання студентом роботи "10" грудня 2025 року
3. Вихідні дані до роботи Розробити програмний продукт у вигляді детективної гри, що забезпечує інтерактивну взаємодію гравця з неігровими персонажами із застосуванням технологій штучного інтелекту. Реалізацію проєкту виконати з використанням ігрового рушія Unity.
1. Вивчення та дослідження предметної області.
2. Аналіз літературних джерел і сучасних підходів.
3. Проєктування структури та архітектури системи.
4. Опис функціонування та логіки роботи системи.
5. Розробка та тестування програмного рішення.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Стан проблемної області.
Розділ 2. Інформаційне забезпечення.
Розділ 3. Математичне забезпечення.
Розділ 4. Програмне забезпечення.
Розділ 5. Розроблення стартап-проєкту.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Підготовка інформаційного матеріалу до доповіді
6. Дата видачі завдання "1" травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	*Вивчення літературних та інформаційних джерел за темою дослідження*	*01.05.2025 – 20.05.2025*	*виконано*
2.	*Аналіз проблематики та визначення напрямів дослідження*	*23.05.2025 – 10.06.2025*	*виконано*
3.	*Формулювання мети, завдань та загальної концепції роботи*	*12.06.2025 – 25.06.2025*	*виконано*
4.	*Вибір та обґрунтування методів і засобів дослідження*	*28.06.2025 – 15.07.2025*	*виконано*
5.	*Розроблення концептуальної схеми реалізації завдання*	*17.07.2025 – 05.08.2025*	*виконано*
6.	*Програмна реалізація завдання*	*09.08.2025 – 01.10.2025*	*виконано*
7.	*Тестування та аналіз результатів*	*04.10.2025 – 25.10.2025*	*виконано*
8.	*Узагальнення результатів та оформлення пояснювальної записки*	*28.10.2025 – 25.11.2025*	*виконано*
9.	*Остаточне доопрацювання матеріалів та підготовка до захисту*	*28.11.2025 – 10.12.2025*	*виконано*

Студент _____ *Марущак С. І.*
(підпис) *(прізвище та ініціали)*

Керівник роботи _____ *Сторожук О.Л.*
(підпис) *(прізвище та ініціали)*

АНОТАЦІЯ

Магістерський проєкт містить 54 сторінки пояснювальної записки, 18 рисунків, 6 таблиць, 2 додатки, 22 джерела.

У процесі виконання проєкту розроблено гру з використанням Unity, MagicaVoxel, Blender, OpenAI API та мови програмування C#. Проект належить до жанру детективних ігор та містить наступні функціональні можливості: деталізовані моделі та середовища, анімації персонажів, звукові ефекти та музичний супровід, повноцінний ігровий процес з інтерактивними елементами.

Ключові слова: комп'ютерна гра, Unity, MagicaVoxel, Blender, C#, OpenAI API.

ABSTRACT

The project contains 54 pages of explanatory note, 18 images, 6 tables, 2 attachments, 22 sources. During the development of the project, a game was created using Unity, MagicaVoxel, Blender, the OpenAI API, and the C# programming language. The project belongs to the detective game genre and features the following functionalities: detailed models and environments, character animations, sound effects and musical accompaniment, and a complete gameplay experience with interactive elements.

Keywords: computer game, Unity, MagicaVoxel, Blender, C#, OpenAI API.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити детективну гру з використанням платформи Unity, засобами MagicaVoxel для створення воксельних моделей, Blender для додавання кісток та анімації персонажів, OpenAI API для інтеграції генеративної моделі. Гра повинна бути розроблена з використанням мови C# та інструментів Unity Engine, забезпечуючи оптимальну продуктивність та функціональність.

Гра повинна включати:

- візуально привабливі моделі та середовище;
- анімації персонажів;
- інтеграцію звукових ефектів та музичного супроводу;
- повноцінний ігровий процес з інтерактивними елементами;
- інтерфейс для керування ігровим процесом.

Перевірити та налагодити роботу гри.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface;

ШІ – штучний інтелект;

JSON – JavaScript Object Notation;

OpenAI – Open Artificial Intelligence;

GPT – Generative Pre-trained Transformer;

3D – Three-Dimensional;

NPC – Non-Player Character;

HTTP – Hypertext Transfer Protocol;

OBJ – Object File Format;

ВСТУП

Ігрова індустрія зазнає значного розвитку, інтегруючи новітні технології для створення захопливих ігрових світів. Однією з ключових тенденцій у сучасному ігровому дизайні є використання штучного інтелекту для покращення інтерактивності та створення динамічних ігрових сценаріїв. Платформа Unity, завдяки своїй гнучкості та потужності, є одним з провідних інструментів для розробки ігор. Використання таких програм, як MagicaVoxel для створення воксельних моделей та Blender для їх оптимізації, дозволяє розробникам створювати унікальні ігрові всесвіти з мінімальними витратами часу та ресурсів.

Актуальність роботи – інтеграція штучного інтелекту для генерації діалогів у відеоіграх, що підвищує рівень інтерактивності та залученості гравців. Це робить гру цікавою для широкої аудиторії, яка цінує інноваційні підходи у ігровому процесі.

Об'єкт дослідження – засоби створення комп'ютерних ігор з використанням штучного інтелекту.

Предмет дослідження – методи та технології розробки ігор на платформі Unity з інтеграцією OpenAI API для генерації діалогів.

Мета роботи – інтегрувати OpenAI API в гру для генерації діалогів NPC.

Завдання роботи – розробити детективну гру на Unity з використанням OpenAI API, MagicaVoxel і Blender.

Наукова новизна – використання OpenAI API для динамічної генерації діалогів у відеоіграх, що дозволяє створювати унікальні ігрові сценарії та підвищувати рівень інтерактивності.

Практична значимість – розвиток навичок логічного мислення та аналізу у гравців, а також у наданні нового способу проведення дозвілля у формі інтерактивної детективної гри.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Особливості детективного жанру в іграх

Детективні ігри – це жанр відеоігор, який зосереджується на розслідуванні злочинів, розгадуванні загадок та розкритті таємниць. Гравці зазвичай беруть на себе роль детектива або слідчого, який повинен зібрати докази, допитати підозрюваних і розгадати головоломки для просування по сюжету. Основна суть детективних ігор полягає в логічному мисленні, аналізі інформації та прийнятті рішень на основі зібраних даних.

Особливості детективних ігор:

- Логічне мислення: гравці повинні використовувати логіку та аналітичні навички для розгадування загадок та просування по сюжету.
- Збір доказів: важливим елементом є збір та аналіз доказів, які допомагають розкрити злочин.
- Діалоги та допити: гравці часто повинні взаємодіяти з різними персонажами, допитувати свідків та підозрюваних для отримання інформації.
- Різноманітні локації: детективні ігри часто пропонують різноманітні локації, які гравець може досліджувати для знаходження підказок та доказів.

Відомі детективні ігри:

- "Sherlock Holmes" серія: ця серія ігор від Frogwares пропонує гравцям стати знаменитим детективом Шерлоком Холмсом і розслідувати різноманітні справи.
- "L.A. Noire": гра від Rockstar Games, яка пропонує гравцям стати детективом у післявоєнному Лос-Анджелесі та розслідувати різноманітні злочини.

- "Ace Attorney" серія: ця серія ігор від Capcom пропонує гравцям стати адвокатом та розслідувати справи, допитуючи свідків та знаходячи докази.
- "The Room" серія: ця серія ігор пропонує гравцям розгадувати складні головоломки та розкривати таємниці в різноманітних локаціях.

1.2 Визначення проблем та недоліків існуючих рішень

У більшості ігор одним з основних недоліків є передбачуваність діалогів та контенту. Гравці часто стикаються з тим, що всі діалоги та сюжетні лінії прописані заздалегідь, що обмежує їхню свободу дій та знижує рівень інтерактивності. Навіть у випадках, коли ігри пропонують розгалуження у діалогах, ці варіанти все одно заздалегідь передбачені та не можуть призвести до більш неординарних або несподіваних розмов. Це створює ряд проблем та недоліків, які можуть вплинути на загальний ігровий досвід.

Головна проблема – обмежена інтерактивність та передбачуваність. Основною проблемою сучасних детективних ігор є те, що діалоги та сюжетні лінії є статичними та передбачуваними. Гравці швидко вчаться передбачати розвиток подій, що знижує рівень захоплення та залученості. Це обмежує можливості для повторного проходження гри, оскільки кожен новий запуск пропонує той самий досвід без значних змін.

Як зазначає автор [7], що під час допитів він вперше усвідомив фундаментальні проблеми гри "L.A. Noire". Навіть коли він правильно інтерпретував вирази обличчя, його часто дивувало те, що говорив Коул (головний герой гри), коли він обирав опції "Правда", "Сумнів" або "Брехня", оскільки це єдині варіанти, доступні гравцям під час допиту. Коул часто йшов далі, ніж очікувалося від інструкції "Сумнів", і з обмеженим списком тем для запитань автор не міг направити Коула на ті шляхи розслідування, які він мав на увазі, просто тому, що Коул їх не мав. Крім того, під час огляду місць злочинів або інших локацій на наявність підказок, Коул має дивовижну здатність відразу визначити, чи є щось корисним чи ні. На думку автора, немає сенсу

дозволяти Коулу піднімати випадкові недопалки або порожні пляшки, якщо він відразу скаже, що вони не мають значення.

На рисунку 1.1 зображено приклад з геймплею гри, де наявні лише три варіанти відповідей: "Правда", "Сумнів" та "Брехня". Це обмежує гравця, оскільки він не може сам вести розмову, а лише обирає одну із заздалегідь підготовлених реплік.

Рисунок 1.1 – Приклад варіантів відповіді під час розмов

1.3 Вплив нейромереж на користувацький досвід та функціональність

Як зазначає дослідник [8], що останніми роками штучний інтелект вийшов за межі традиційних ролей у розробці ігор, таких як поведінка ворогів та пошук шляху, і почав виконувати більш творчі функції, пов’язані зі створенням контенту. Цей зсув відкрив можливість створення динамічніших, персоналізованих та масштабних ігрових світів. ШІ вже здатен генерувати цілі рівні, сюжетні лінії та навіть персонажів, пропонуючи нові можливості як для розробників, так і для гравців. Інтеграція штучного інтелекту в процес створення відеоігор є трансформаційним кроком, що змінює підхід до проектування інтерактивних вражень, надаючи безмежне розмаїття ігрових світів та сценаріїв. У своїй праці автор досліджує різні форми інтеграції ШІ в ігри та інтерактивні медіа, аналізуючи його потенціал у зміні

галузі. Відеоігри створені за допомогою штучного інтелекту, є передовим напрямом у сфері розробки ігор. Він підкреслює, що завдяки процедурній генерації, сюжетам, керованим ШІ, та машинному навчанню створюються динамічні й персоналізовані ігрові світи, які раніше здавались недосяжними. Хоча ця технологія перебуває на ранньому етапі розвитку, потенціал ШІ для трансформації ігрової індустрії є значним. У перспективі очікується, що ШІ братиме участь не лише у створенні контенту, але й у процесі прийняття творчих рішень, що докорінно змінить взаємодію гравців із цифровими світами.

1.4 Питання етики та відповідальності

Генеративний штучний інтелект стикається з низкою серйозних викликів, серед яких ключовими є упередженість навчальних даних та питання інтелектуальної власності. Оскільки ШІ навчається на великих обсягах людських даних, він неминуче відтворює притаманні суспільству расові, гендерні та соціальні стереотипи, що може призводити до дискримінаційних результатів. Водночас використання захищених авторським правом матеріалів як основи для навчання моделей викликає суперечки щодо прав творців і законності створеного контенту. Попри спроби платформ вирішити ці проблеми – шляхом використання відкритих або ліцензованих джерел – прозорість щодо походження даних і заходів протидії упередженості залишається недостатньою. Це вимагає системного підходу до курування даних, етики моделювання та правового регулювання.

1.5 Платформи для розробки ігор

Програмні платформи, які часто називають «ігровими рушіями», надають набір можливостей і ресурсів для створення відеоігор. Вони формують єдине середовище, що об'єднує різні аспекти розробки, включаючи музику, фізику, графіку, штучний інтелект та мережеву логіку.

Головні завдання ігрових рушіїв:

- Прискорення процесу розробки: завдяки вбудованим інструментам та готовим компонентам, що дозволяє швидше створювати гру.
- Підтримка кількох платформ: багато рушіїв дають змогу запускати ігри на різних пристроях з мінімальними змінами.
- Підвищення якості: використання надійних і оптимізованих рішень для графіки, фізики та інших елементів гри.
- Скорочення витрат: розробники можуть використовувати готові рішення замість створення всього з нуля, що значно знижує витрати.

Основні складові ігрового рушія:

- Графічний рендеринг: відповідає за виведення зображень на екран.
- Фізичний рушій: моделює фізичні взаємодії (зіткнення, гравітація тощо).
- Аудіосистема: забезпечує відтворення музики та звукових ефектів.
- Інструменти для створення контенту: редактори для побудови рівнів, моделей, анімацій та іншого.
- Мережева підсистема: забезпечує підтримку багатокористувацьких режимів.

Ігрові рушії є ключовими інструментами у створенні сучасних відеоігор. Вони надають розробникам інструменти та ресурси, необхідні для створення якісного контенту, придатного для різних платформ і налаштувань.

1.6 Засоби створення ігрових моделей

Спеціалізоване програмне забезпечення, яке відоме як інструменти для моделювання та анімації, використовується для створення візуального контенту в різних сферах, зокрема у віртуальній реальності, розробці ігор, кіноіндустрії, анімації та архітектурі.

Моделювання передбачає процес створення тривимірних (3D) об'єктів або персонажів, включаючи формування їхньої структури та деталей. Такі інструменти

дають змогу дизайнерам і художникам створювати високоякісні віртуальні моделі для подальшого використання у різних медіа.

Анімація – це процес надання руху об'єктам або персонажам шляхом відтворення послідовності кадрів, що створює ілюзію руху. Програмні інструменти для анімації дозволяють анімувати персонажів, об'єкти й цілі сцени, регулюючи їхню динаміку, рухи та взаємодії.

Рендеринг – це процес перетворення тривимірної моделі у фінальне зображення чи анімацію з використанням текстур, освітлення та інших візуальних ефектів. Більшість інструментів для моделювання та анімації мають вбудовані засоби рендерингу або дозволяють інтеграцію зі сторонніми рендерерами.

Ці інструменти також надають користувачам можливість як створювати нові 3D-моделі та анімації з нуля, так і редагувати наявні. Вони пропонують широкий набір функцій для роботи з геометрією, текстурами й матеріалами, а також для додавання та налаштування текстур і матеріалів, що забезпечує більш реалістичний вигляд моделей.

1.7 Сучасні нейромережі та їх API

Сучасні нейромережі – це потужні інструменти для розв’язання завдань у сферах штучного інтелекту, зокрема комп’ютерного зору, обробки природної мови та рекомендаційних систем. Вони здатні навчатися на великих обсягах даних і знаходити складні закономірності, що робить їх незамінними у багатьох галузях.

API (Application Programming Interface) – це програмний інтерфейс, який дозволяє різним програмам взаємодіяти між собою. Іншими словами, API надає стандартизований спосіб звернення до функцій або даних, що полегшує інтеграцію різних сервісів у застосунки.

API нейронних мереж – це спеціалізовані інтерфейси, що дозволяють розробникам легко інтегрувати штучний інтелект у свої проекти. Через API можна надсилати запити до вже навчених моделей та отримувати результати без потреби у

глибокому розумінні самої архітектури нейромережі. Це дає змогу швидко додавати можливості машинного навчання до веб- та мобільних застосунків.

Сучасні нейромережі здатні виконувати широкий спектр завдань: розпізнавання об’єктів на зображеннях, генерацію тексту, переклад мов, класифікацію даних та багато іншого. Їхні API зазвичай надають готові кінцеві точки для таких завдань, що значно спрощує інтеграцію.

Такі API дозволяють працювати з готовими моделями або створювати власні, використовуючи інтерфейси для завантаження даних, навчання та інференсу (застосування моделі для прогнозування). Це особливо зручно для бізнесу та розробників, які хочуть швидко отримати доступ до передових технологій.

Інтеграція API нейромереж у застосунки дає змогу значно покращити їхню функціональність та якість. Розробники можуть легко додати нові можливості, такі як автоматичний переклад, генерація рекомендацій чи розпізнавання мовлення, що відкриває нові горизонти для цифрових продуктів.

1.8 Налаштування промптів для генерації контенту

Незважаючи на свої широкі можливості, генеративні моделі ШІ не є безпомилковими та можуть створювати неточний або низькоякісний контент, що особливо критично в галузях, де потрібна висока точність — таких як медицина, фінанси чи юриспруденція. Як зазначає автор [11], для вирішення цієї проблеми важливо глибше розуміти принципи роботи великих мовних моделей. Зокрема, усвідомлення того, що "більше даних" не завжди означає "краще", адже надлишок інформації може знижувати загальну точність. У цьому контексті ключовим стає освоєння навичок, відомих як інженерія запитів (Prompt Engineering), ефективне використання контексту та грамотна робота з API.

По суті, інженерія запитів – це систематичний підхід до формулювання запитів, що оптимізує відповідь LLM, причому дослідження показують, що добре структуровані запити можуть покращити продуктивність моделі до 37% для різних завдань [12].

Ефективність цієї взаємодії залежить від кількох ключових технічних компонентів, які працюють узгоджено, щоб керувати обробкою моделі та генерацією результатів, демонструючи значні покращення як обчислювальної ефективності, так і точності відповіді.

1.9 Застосування дерева рішень для структуризації проблеми

Спочатку дослідження проблеми передбачає її розбиття на складові частини, тобто декомпозицію. На першому етапі визначається основна, комплексна проблема, після чого вона деталізується шляхом поділу на менші елементи. Це допомагає створити загальну структуру проблеми в межах системи та побудувати дерево проблем для подальшого вивчення. Опис основних підпроблем, що зумовлюють існування головної проблеми – обмеженої інтерактивності діалогів у детективних відеоіграх (рис. 1.2):

- Основна частина сучасних детективних ігор використовує заздалегідь визначені сценарії взаємодії, що призводить до одноманітності діалогів та обмеження варіантів розвитку розмови.
- У традиційних системах NPC не адаптують стиль спілкування до гравця, що негативно впливає на реалістичність ігрового процесу. Діалоги не залежать від попередніх дій користувача, його рішень або логіки взаємодії з персонажами.
- Традиційні підходи не передбачають створення нових варіантів відповідей залежно від контексту. Усі репліки створюються заздалегідь у процесі розробки, що істотно обмежує гнучкість системи діалогів.

Рисунок 1.2 – Дерево проблем діалогових систем NPC

Застосування мовних моделей для генерації діалогів створює новий клас проблем, пов’язаних із необхідністю коректного налаштування та контролю роботи штучного інтелекту (рис. 1.3):

- Некоректна інтерпретація контексту може призводити до нелогічних відповідей NPC.
- Недостатнє обмеження промтів здатне спричинити генерацію небажаного або неприйнятного контенту.
- Відсутність фільтрації відповідей може погіршувати якість взаємодії. кладність налаштування параметрів генерації потребує спеціального підходу до проектування системи.

Рисунок 1.3 – Проблеми використання штучного інтелекту у діалогових системах

1.10 Опис об’єкта проектування та формалізація задачі

API нейронних мереж – це спеціалізовані інтерфейси, що дозволяють розробникам легко інтегрувати штучний інтелект у свої проекти. Через API можна надсилати запити до вже навчених моделей та отримувати результати без потреби у глибокому розумінні самої архітектури нейромережі. Це дає змогу швидко додавати можливості машинного навчання до веб- та мобільних застосунків.

Більшість сучасних детективних ігор зберігають традиційний підхід до реалізації діалогів, що ґрунтується на заздалегідь підготовлених репліках і обмеженому наборі варіантів відповідей. Побудоване дерево проблеми підтверджує наявність низки недоліків, зокрема низького рівня інтерактивності, відсутності адаптації до дій гравця та недостатнього врахування контексту ігрової ситуації. У сукупності ці фактори негативно впливають на якість ігрового процесу та знижують рівень залученості користувача.

Об’єкт проектування повинен мати такі властивості:

- забезпечення динамічної генерації діалогів на основі мовної моделі штучного інтелекту з урахуванням поточного ігрового контексту.
- підтримка індивідуальних характеристик NPC, зокрема стилю мовлення, ролі у сюжеті та поведінкових особливостей.
- формування відповідей з урахуванням історії діалогу, що забезпечує логічну послідовність спілкування.
- реалізація механізмів контролю та фільтрації згенерованого контенту з метою недопущення небажаних або некоректних відповідей.
- інтеграція системи у середовище Unity з можливістю масштабування та подальшого розширення функціоналу.

Висновки до розділу

У цьому розділі було розглянуто ключові аспекти детективних ігор, їхні особливості та відомі представники жанру. Детективні ігри вимагають від гравця логічного мислення, здатності до аналізу інформації та прийняття рішень на основі

отриманих даних, а також передбачають дослідження ігрового середовища та взаємодію з персонажами.

Також було проаналізовано ігрові рушії як основні засоби розробки сучасних відеоігор, визначено їхні функціональні можливості та роль у створенні якісного програмного продукту. Розглянуто інструменти моделювання та анімації, які використовуються для формування візуального контенту та забезпечення реалістичного відображення ігрових об’єктів.

Окрему увагу приділено сучасним нейромережам та їх API як засобам інтеграції інтелектуальних функцій у програмні системи. Проаналізовано можливості застосування мовних моделей для генерації діалогів, а також їхній вплив на користувацький досвід, зокрема у контексті підвищення інтерактивності та рівня залучення гравця.

Крім того, у розділі було проаналізовано проблеми та недоліки існуючих підходів до реалізації діалогових систем у детективних іграх, що дало змогу обґрунтувати доцільність використання штучного інтелекту. Окремо розглянуто питання етики та відповідальності при застосуванні нейромереж, зокрема ризики генерації небажаного контенту та необхідність контролю результатів роботи мовних моделей.

Також було побудовано дерево рішень, яке відображає основні проблеми предметної області, та виконано опис об’єкта проектування з формалізацією задачі розроблення системи генерації діалогів NPC для детективної гри.

Загалом, результати аналізу підтверджують актуальність обраної теми та необхідність використання сучасних технологій для створення більш реалістичного та адаптивного ігрового середовища.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Основні сутності системи

У межах розроблюваної інформаційної системи основними сутностями є гравець, NPC, сцена (локація), діалог, повідомлення та контекст спілкування. Гравець виступає ініціатором взаємодії та здійснює вибір дій, які безпосередньо впливають на перебіг сюжету. NPC є неігровими персонажами, що реагують на дії користувача та беруть участь у розкритті подій гри. Сцена або локація визначає просторове середовище, у якому відбувається взаємодія між персонажами, та задає контекст подій. Діалог є ключовим елементом системи взаємодії, що забезпечує передачу інформації між гравцем і персонажами.

Повідомлення являє собою одиницю комунікації в межах діалогу та містить текстові репліки обох сторін. Контекст (історія діалогу) включає сукупність попередніх реплік і дій, які впливають на формування наступних відповідей NPC. Завдяки цьому забезпечується логічна послідовність спілкування, а також динамічна адаптація системи до стилю поведінки гравця.

2.2 Вхідні та вихідні дані

Вхідними даними системи є текстові повідомлення гравця, які ініціюють процес формування відповіді NPC, а також поточний ігровий стан, що визначає логіку розвитку подій. До вхідних параметрів також належать характеристики персонажів (роль, стиль мовлення, поведінкові особливості) та історія попередніх діалогів, яка відображає контекст спілкування. Сукупність цих даних дозволяє системі формувати релевантні відповіді з урахуванням попередніх дій гравця та особливостей розвитку сюжету.

Вихідними даними є згенерований текст відповіді NPC, що відображає реакцію персонажа на введене повідомлення гравця, а також додаткові параметри, зокрема емоційний стан персонажа та тригери ігрових подій.

Активація тригерів може призводити до змін у стані гри, відкриття нових діалогових гілок або запуску сюжетних подій, що забезпечує динамічний розвиток ігрового процесу та підвищує рівень інтерактивності.

2.3 Структури даних і спосіб зберігання

У розроблюваній системі використовуються кілька типів структур даних, що відповідають різним рівням подання інформації. Конфігураційні дані, пов’язані з неігровими персонажами, їхніми характеристиками, роллю в сюжеті та базовими параметрами поведінки, доцільно зберігати у вигляді ScriptableObject у середовищі Unity. Це забезпечує зручне редагування даних без змін у програмному коді та спрощує повторне використання налаштувань для різних сцен. До таких даних належать описи NPC, метаінформація про сцени, базові діалогові шаблони та налаштування ігрових параметрів. У таблиці 2.1 викладено ключові властивості класу, який відповідає за опис персонажа

Таблиця 2.1 Властивості NPC

Категорія	**Параметр**	**Опис**
Ідентифікаційні параметри	Id	Унікальний числовий ідентифікатор запису в базі даних
	Name	Особисте ім'я персонажа
	Surname	Прізвище персонажа
	Character Portrait	Посилання на графічний ресурс портретного зображення
Демографічні атрибути	Age	Вік персонажа у роках
	Gender	Біологічна стать або гендерна ідентичність
	Birth Place	Географічне місце народження
Професійна ідентифікація	Occupation	Поточна професія або рід занять
Описові характеристики	Appearance Description	опис зовнішності та фізичних особливостей

Продовження таблиці 2.1 Властивості NPC

Категорія	**Параметр**	**Опис**
Описові характеристики	Personality	Психологічний профіль, риси характеру та поведінкові патерни
Наративні елементи	Goals	Цілі та мотивації персонажа в межах сюжету
	Secrets	Конфіденційна інформація, відома персонажу
	Weaknesses	Вразливості, слабкі сторони характеру
	Role In Story	Функціональна роль у наративній структурі
Діалогова система	Dialogue Triggers	Колекція тригерів діалогів з підструктурою
	Trigger	Ключова фраза або умова активації діалогу
	Secret Reaction	Внутрішня реакція персонажа на тригер
	Is Triggered	Булевий прапорець стану активації тригера
Системні обмеження	Max Messages Per Session	Технічний параметр обмеження кількості повідомлень за сесію

Для обміну інформацією з мовною моделлю та зберігання історії діалогів використовується формат JSON, який дозволяє уніфіковано передавати структури запитів і відповідей, включно з текстом реплік, службовими параметрами та технічними полями. На логічному рівні історія взаємодій, стани розслідування й додаткові службові дані можуть бути подані у вигляді таблиць (наприклад, умовних логічних таблиць або записів), де для кожного діалогу фіксуються учасники, час, зміст повідомлень та зміни в ігровому стані. Такий підхід забезпечує прозору організацію даних, спрощує подальший аналіз та дає можливість розширення системи без суттєвої зміни її структури.

2.4 Вимоги та обмеження даних

Одним із ключових обмежень у системі є довжина контексту, яка зумовлена технічними можливостями мовної моделі та лімітами на обсяг текстової інформації, що передається для обробки. Для підтримання стабільної роботи здійснюється

контроль історії діалогу шляхом відбору лише найбільш релевантних фрагментів спілкування, необхідних для формування коректної відповіді неігрового персонажа.

Для запобігання перевантаженню системи та зловживанню з боку користувачів також обмежено максимальну кількість символів для введення гравцем. Таке обмеження змушує формулювати репліки більш чітко та лаконічно, а також зменшує ймовірність штучного навантаження на систему. Водночас довжина відповіді NPC також підлягає контролю з метою уникнення формування надмірно об’ємних текстових блоків, що негативно впливають на динаміку ігрового процесу.

Окремо враховуються обмеження, встановлені безпосередньо розробником мовної моделі через використання стороннього API, на які розробник гри не має прямого впливу. До них належать обмеження на кількість запитів, максимальний розмір повідомлень, швидкість обробки запитів і політики контентної безпеки. Такі вимоги встановлюються постачальником сервісу та є обов’язковими до виконання, що потребує адаптації програмної системи до зовнішніх технічних умов і регламентів використання.

2.5 UML-діаграма прецедентів гри

На діаграмі прецедентів зображено основні сценарії взаємодії гравця з ігровою системою, а також залучення сервісу штучного інтелекту для формування діалогів NPC. Діаграма відображає ключові функціональні можливості гри, зокрема початок гри, взаємодію з персонажами, аналіз доказів і керування прогресом розслідування. Окремо виділені процеси аналізу контексту, генерації та фільтрації відповіді, що дозволяє наочно представити логіку обробки реплік у межах системи. На рисунку 2.1 зображено діаграму прецедентів гри з використанням AI-сервісу.

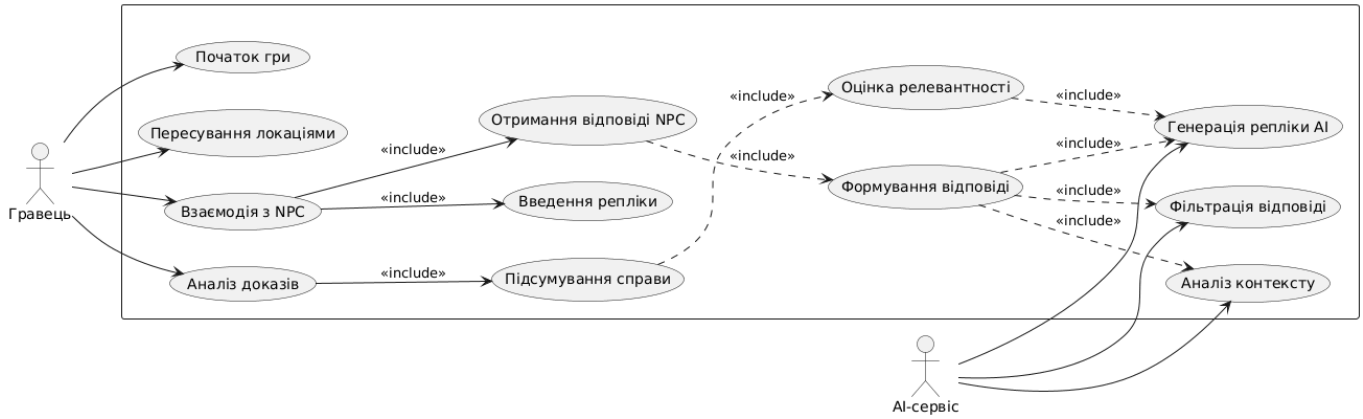

Рисунок 2.1 – Діаграма прецедентів

2.6 Роль геймдизайну в розробці детективної гри

Геймдизайн та сюжет відіграють ключову роль у формуванні цілісного ігрового досвіду, особливо в детективних іграх, де логіка подій, послідовність взаємодій і мотиви персонажів безпосередньо впливають на інтерес гравця. Правильно побудований сюжет забезпечує мотивацію до дослідження ігрового середовища, а геймдизайн визначає, у який спосіб гравець отримує інформацію, відкриває нові події та взаємодіє з персонажами. У детективних іграх особливо важливо, щоб структура завдань не була лінійною та примусовою, а дозволяла гравцеві самостійно робити висновки на основі зібраних фактів і результатів діалогів.

Сюжетна частина гри повинна бути логічно узгодженою та містити внутрішні причинно-наслідкові зв’язки між подіями. Кожен персонаж має виконувати конкретну роль у розслідуванні та мати власні мотиви, що створює ефект правдоподібності та імітації реального слідства. Наявність прихованих деталей, суперечливих свідчень і поступового відкриття інформації формує відчуття напруги та зацікавленості. У межах цього проекту сценарій побудовано таким чином, щоб кожна локація та кожен персонаж мали сюжетне обґрунтування і були залучені до загальної логіки розслідування

Грамотно продуманий геймдизайн також визначає баланс між складністю завдань та доступністю ігрової логіки. Надмірно складні або, навпаки, занадто прості сценарії можуть призвести до втрати зацікавленості користувача. Тому важливо, щоб гравець поступово отримував нову інформацію, мав змогу аналізувати ситуацію та

ухвалювати рішення, що впливають на розвиток подій. Поєднання продуманого сюжету з логічною структурою ігрових механік дозволяє створити не лише розважальний продукт, але й інтерактивне середовище, яке стимулює мислення, увагу та аналітичні навички користувача.

Висновки до розділу

У розділі було визначено основні сутності системи, що формують інформаційну модель детективної гри, зокрема гравця, неігрових персонажів, сцени, діалоги та контекст взаємодії. Також було проаналізовано вхідні та вихідні дані, які забезпечують взаємодію користувача з системою та формування відповідей NPC із урахуванням поточного стану гри.

Окрему увагу приділено структурам зберігання інформації та логічному розподілу даних між різними форматами подання. Застосування ScriptableObject, JSON і таблиць дозволяє забезпечити гнучку організацію даних, зручність їх редагування та можливість масштабування системи без суттєвої зміни архітектури.

Розглянуті обмеження щодо обсягу контексту, формату введення та вимог до коректності даних дали змогу визначити технічні рамки функціонування системи. Побудована UML-діаграма прецедентів наочно відобразила основні сценарії взаємодії гравця з грою та використання сервісу штучного інтелекту для генерації діалогів, що дозволило сформувати цілісне уявлення про функціональну структуру проекту.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Алгоритм формування промпта

Формування промпта здійснюється шляхом побудови структурованого запиту до мовної моделі, який містить усю необхідну інформацію для генерації коректної та контекстно-залежної відповіді NPC. Промпт має чітко визначену структуру, що складається з ролі персонажа, опису поточної сцени, історії попередніх діалогів та набору обмежень. Такий підхід дозволяє організувати дані у логічно впорядкованому вигляді та мінімізувати ймовірність отримання некоректної або нерелевантної відповіді.

На етапі збирання промпта система отримує опис ролі персонажа, який включає його характер, манеру спілкування та сюжетну функцію. Контекст сцени формується на основі поточної локації, активних подій та стану розслідування. Історія діалогу відбирається з урахуванням обмежень на довжину контексту, при цьому зберігаються лише найбільш значущі репліки, що забезпечують логічну послідовність спілкування з NPC. Такий механізм дозволяє підтримувати узгодженість відповідей протягом тривалого ігрового процесу.

Окремий блок промпта становлять обмеження, які визначають допустиму довжину відповіді, стиль мовлення персонажа та правила поведінки мовної моделі. До них також можуть належати заборони на використання неприйнятної лексики або відхилення від ролі персонажа. Усі зібрані елементи інтегруються в єдиний текстовий запит, який передається до API мовної моделі для формування відповіді. Така багатокомпонентна структура промпта забезпечує контрольовану і передбачувану поведінку NPC у процесі діалогу. На рисунку 3.1 зображено діаграму послідовності всього циклу обробки повідомлення гравця до NPC.

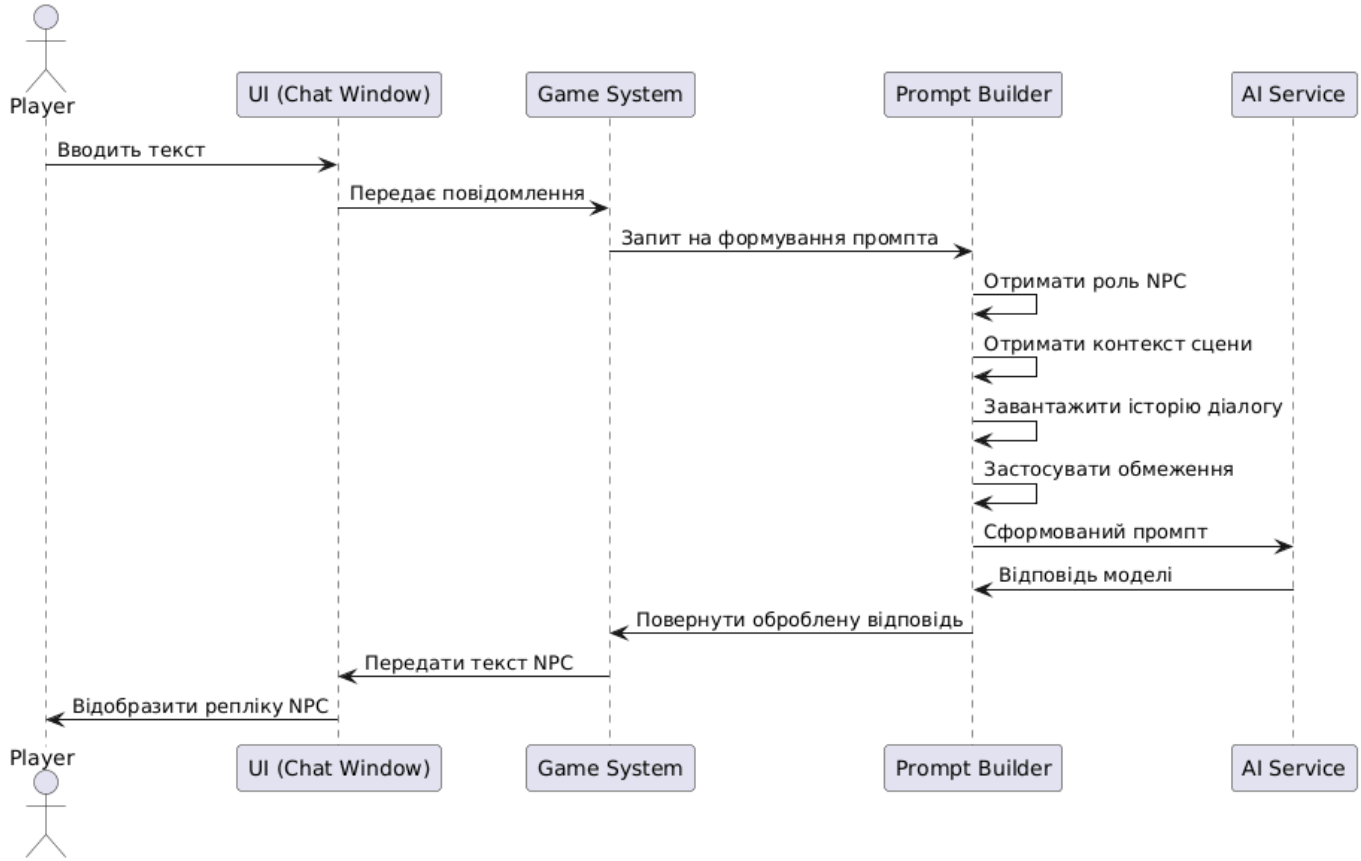

Рисунок 3.1 – Діаграма послідовності обробки повідомлення

3.2 Семантична перевірка відповідей гравця

У межах розроблюваної інформаційної системи основними сутностями є гравець, NPC, сцена (локація), діалог, повідомлення та контекст спілкування. Гравець виступає

Семантичне порівняння двох речень полягає у визначенні ступеня їх змістової подібності незалежно від конкретних слів і формулювань. На відміну від простого порівняння рядків або підрахунку кількості однакових слів, семантичний аналіз спрямований на виявлення спільного сенсу між двома висловлюваннями. Наприклад, речення «Підозрюваний був у кімнаті» та «Людина знаходилась у приміщенні» можуть вважатися семантично подібними, навіть якщо не мають спільних ключових слів. Такий підхід дозволяє системі коректно інтерпретувати відповіді гравця без жорсткої прив’язки до наперед визначених формулювань.

Семантичну подібність формально представлено у вигляді функції (3.1):

$$Sim(S_1, S_2) = \cos(\overrightarrow{v_1}, \overrightarrow{v_2}) = \frac{\overrightarrow{v_1} \times \overrightarrow{v_2}}{\|\overrightarrow{v_1}\| \times \|\overrightarrow{v_2}\|} \quad (3.1)$$

Де S_1 – відповідь гравця, S_2 – еталонна відповідь, а $\overrightarrow{v_1}$, $\overrightarrow{v_2}$, – векторні представлення речень у семантичному просторі, отримані за допомогою нейромережі. Значення Sim належить інтервалу [0;1], де 0 означає відсутність змістової подібності, а 1 – повний семантичний збіг. Отримані значення використовуються для визначення ступеня релевантності відповіді та прийняття рішення щодо її правильності.

У контексті реалізації гри семантична перевірка використовується під час проходження ключових ігрових сцен, де гравцеві необхідно надати відповіді на низку запитань для підтвердження розуміння сюжету та виявлення причин події. Наприклад, після збору доказів гравець проходить етап перевірки, у якому система оцінює його міркування не за точними формулюваннями, а за змістом відповідей. Якщо показник схожості перевищує задане порогове значення, гравець вважається таким, що правильно ідентифікував ключові факти справи, і отримує доступ до наступного рівня або сюжетної події. Усі обчислення семантичної подібності виконуються на стороні AI-сервісу, що пов’язано з високими вимогами до обчислювальних ресурсів, натомість ігрова система використовує лише числовий результат у вигляді відсотка збігу для логіки прийняття рішень.

3.3 Модель керування контекстом

У системі реалізовано механізм керування контекстом діалогу, призначений для збереження найбільш важливої інформації з історії взаємодії гравця з NPC та водночас обмеження обсягу оброблюваних даних. У пам’яті системи постійно зберігається фіксована кількість останніх повідомлень, які безпосередньо впливають на формування відповідей мовної моделі. Збереженню підлягають ключові репліки гравця та NPC, поточний стан розслідування, важливі сюжетні факти та активні події сцени.

Нехай історія діалогу подається множиною повідомлень (3.2):

$$H = \{m1, m2, \dots, m_n\}, \quad (3.2)$$

де кожне повідомлення m_i є окремою реплікою гравця або NPC.

У системі задається максимальна допустима кількість повідомлень (3.3):

$$|H| \leq N_{max}, \quad (3.3)$$

де N_{max} – встановлений ліміт на кількість повідомлень, що можуть одночасно зберігатись в активному контексті.

Якщо виконується умова перевищення ліміту (3.4):

$$|H| > N_{max}, \quad (3.4)$$

Запускається процедура сумаризації історії (3.5):

$$S = Summarize(H), \quad (3.5)$$

де результатом є узагальнений опис ключових подій розмови. Після виконання сумаризації контекст оновлюється (3.6):

$$H' = \{S\} \cup \{m_{k+1}, \dots, m_n\} \quad (3.6)$$

Інформація, що втратила актуальність або не впливає на розвиток поточного сценарію, підлягає видаленню з активного контексту. Для цього при перевищенні заданого ліміту кількості повідомлень виконується запит на автоматичну сумаризацію історії діалогу, у результаті якого формується стислий виклад головних подій. Отримана вижимка зберігається в окремому блоці історії, тоді як старі повідомлення видаляються, що дозволяє скоротити обсяг даних без втрати сюжетної цілісності.

Такий підхід дає змогу зберігати важливі факти та ключові рішення гравця, одночасно запобігаючи перевантаженню системи надмірною кількістю повідомлень і стабілізуючи роботу діалогового модуля протягом тривалих ігрових сесій. Крім того, механізм сумаризації забезпечує підтримку семантичної спадковості подій, що дозволяє мовній моделі коректно інтерпретувати попередні дії гравця та зберігати логічність відповідей навіть після скорочення історії взаємодії.

Висновки до розділу

У розділі було розроблено алгоритм формування промпта для взаємодії з мовною моделлю, який забезпечує структуровану передачу ролі персонажа, контексту сцени, історії діалогу та обмежень поведінки. Це дозволяє контролювати процес генерації відповіді NPC та підтримувати логічну узгодженість реплік у межах ігрового сюжету.

Також було реалізовано механізм семантичної перевірки відповідей гравця, що ґрунтується на аналізі змістовної подібності порівнюваних речень. Застосування нейромережевих моделей для семантичної оцінки дало змогу відмовитися від формального порівняння слів і перейти до аналізу сенсу відповідей, що підвищує точність оцінювання та забезпечує більш природну взаємодію користувача з ігровою системою.

Окрему увагу приділено моделі керування контекстом, яка дозволяє контролювати обсяг активної історії діалогу шляхом обмеження кількості збережених повідомлень та використання механізму сумаризації. Це забезпечує ефективне використання ресурсів, збереження ключових сюжетних фактів і стабільну роботу системи під час тривалих ігрових сесій. Сукупність розглянутих моделей формує цілісну алгоритмічну основу для побудови адаптивної діалогової системи в детективній грі.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Моделювання 3D-об'єктів у MagicaVoxel

MagicaVoxel – це безкоштовний інструмент для створення воксельної графіки, розроблений Ефремом Лю (Ephtracy). У воксельній графіці використовуються тривимірні пікселі у вигляді кубів або блоків. Такі «вокселі» виступають будівельними елементами популярних ігор на кшталт Minecraft і дозволяють створювати складні та захопливі 3D-сцени [2].

Створення моделі полягає у поступовому формуванні об’єкта з окремих об’ємних елементів – вокселів, які можна розглядати як тривимірні пікселі. За допомогою інструментів малювання користувач додає вокселі до сцени, формуючи загальний обрис об’єкта, а інструменти видалення дозволяють коригувати форму, прибираючи зайві елементи. Кольори задаються безпосередньо для кожного вокселя, що дозволяє фарбувати модель на етапі створення, виділяти окремі деталі та створювати візуальні акценти. Таким чином, модель одночасно формується та оформлюється в єдиному середовищі без потреби у додаткових редакторах.

Редактор пропонує різноманітні інструменти для створення й редагування воксельних моделей: пензлі, ластики, інструменти для заливки тощо. Це дає змогу створювати деталізовані 3D-об’єкти з високою точністю. У MagicaVoxel також вбудований рендерер, який дозволяє візуалізувати моделі з урахуванням освітлення, тіней і матеріалів, що важливо як для попереднього перегляду результату, так і для підготовки візуальних матеріалів для проекту.

Програма підтримує експорт моделей у різні формати, зокрема OBJ, що дає змогу використовувати створені об’єкти в інших додатках або безпосередньо в ігрових рушіях, таких як Unity або Unreal Engine. Після експорту моделі можуть додатково оптимізовуватися та оброблятися в інших програмах, що підвищує їхню придатність до використання в ігровому середовищі.

MagicaVoxel часто застосовується для створення персонажів, будівель та декоративних елементів у відеоіграх. Його простота, наочність і функціональні

можливості роблять його ефективним інструментом для інді-розробників та студентських проектів. На рисунку 4.1 наведено приклад створеної 3D-моделі в середовищі MagicaVoxel.

Рисунок 4.1 – Рендер готової 3D-моделі

4.2 Обробка моделей у Blender

Blender – це безкоштовне програмне забезпечення з відкритим кодом для тривимірної графіки, яке розробляється Blender Foundation та активною спільнотою користувачів. Раніше його часто вважали «безкоштовною альтернативою» для тих, хто не мав змоги придбати комерційні програми, що використовуються в індустрії [3].

Проте з появою нових оновлень Blender перетворився на повноцінний інструмент, здатний конкурувати з професійними рішеннями. Програма має розвинені функції для створення та редагування матеріалів і текстур, дозволяє створювати реалістичні поверхні, використовувати процедурні текстури, налаштовувати шейдери та роботу з освітленням.

У Blender реалізовані потужні рушії рендерингу – Cycles і Eevee. Cycles забезпечує фотореалістичний рендеринг на основі трасування променів, а Eevee дозволяє отримувати швидке зображення в режимі реального часу, що зручно для інтерактивної роботи та попереднього перегляду моделей у процесі розробки.

Blender широко застосовується для створення та обробки моделей персонажів, об’єктів і середовищ у відеоіграх, що робить його ефективним інструментом для розробників завдяки розвинутим можливостям моделювання та підготовки моделей до використання в ігрових рушіях. Програма підтримує численні формати імпорту та експорту, що забезпечує її сумісність з іншими графічними редакторами й середовищами розробки.

У процесі реалізації проекту Blender використовувався для подальшої обробки моделей, створених у MagicaVoxel. Після імпорту моделей виконувалося об’єднання вершин та оптимізація геометрії з метою зменшення кількості полігонів і підвищення продуктивності гри. Також налаштовувалися масштабні параметри та орієнтація об’єктів, що забезпечувало їх коректний експорт у форматі FBX та подальше використання в середовищі Unity без порушення пропорцій. На рисунку 4.2 зображено повну візуальну схожість моделі до та після об’єднання вершин.

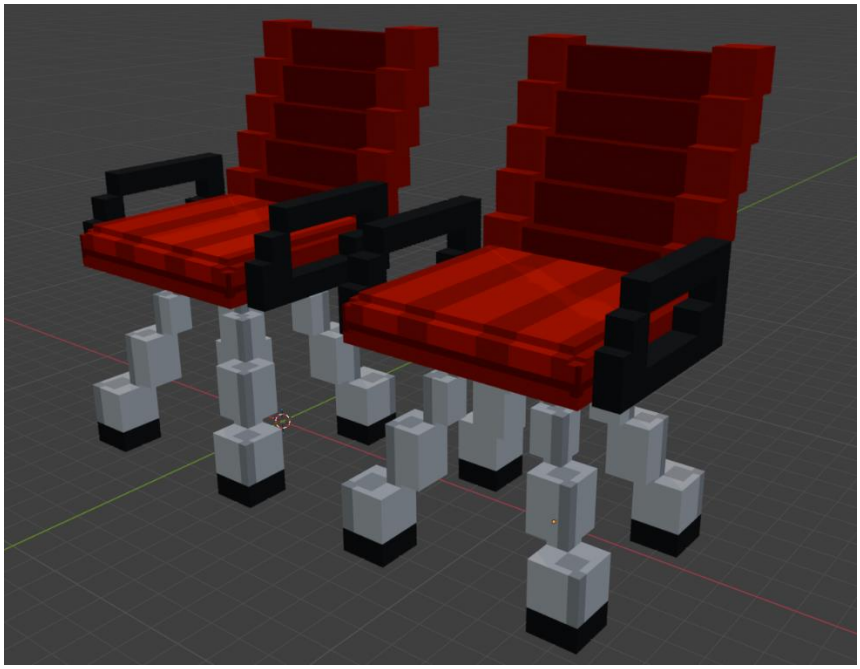

Рисунок 4.2 – Візуальне порівняння моделі до і після оптимізації

У таблиці 4.1 наведено порівняльні характеристики згаданої раніше 3D-моделі до та після оптимізації, зокрема кількість вершин, ребер, полігонів і трикутників.

Таблиця 4.1 Порівняння до та після оптимізації

Показник	**До**	**Після**	**Оптимізація, %**
Вершини	1577	398	74,8
Ребра	2629	624	76,3
Полігони	1254	222	83,2
Трикутники	1254	804	35,9

Для неігрових персонажів додатково виконувалося створення скелетної структури та процес рігінгу, який передбачає прив’язування вершин моделі до відповідних кісток. Це дало змогу реалізувати анімацію рухів, зміну поз та міміки персонажів у грі. Призначення ваг вершин здійснювалося з урахуванням анатомічної логіки, що забезпечувало природність рухів і коректну деформацію моделі під час виконання анімацій.

На рисунку 4.3 представлено процес створення скелетної структури (рігінгу) 3D-моделі неігрового персонажа, який полягає у формуванні ієрархії кісток для керування анімацією та деформацією геометрії моделі. Скелет складається з набору логічно впорядкованих кісток, кожна з яких відповідає за рух конкретної частини тіла, зокрема голови, тулуба, рук та ніг. Така структура дозволяє реалізувати реалістичні рухи персонажа шляхом трансформації відповідних зон сітки через механізм прив’язки вершин до кісток.

Кожна кістка впливає лише на визначений набір вершин, які відповідають відповідній анатомічній ділянці, що забезпечує коректну деформацію моделі під час руху. Наприклад, кістки плеча та передпліччя керують рухами руки, тоді як кістки стегна, гомілки і стопи відповідають за анімацію ноги. При цьому у скелеті використовується ієрархічна модель, де деякі кістки є батьківськими по відношенню до інших, а їхні трансформації автоматично передаються дочірнім кісткам. Наприклад, обертання плечової кістки змінює положення всієї руки, включно з ліктем і долонею.

Ієрархічна організація скелета дозволяє досягти природної синхронізації рухів, оскільки переміщення тулуба або тазу автоматично впливає на положення ніг або хребта. Це зменшує потребу у ручному коригуванні кожної кістки окремо та значно спрощує створення комплексних анімацій. Завдяки коректно побудованій скелетній системі забезпечується плавність рухів та стабільність анімації персонажа в ігровому середовищі.

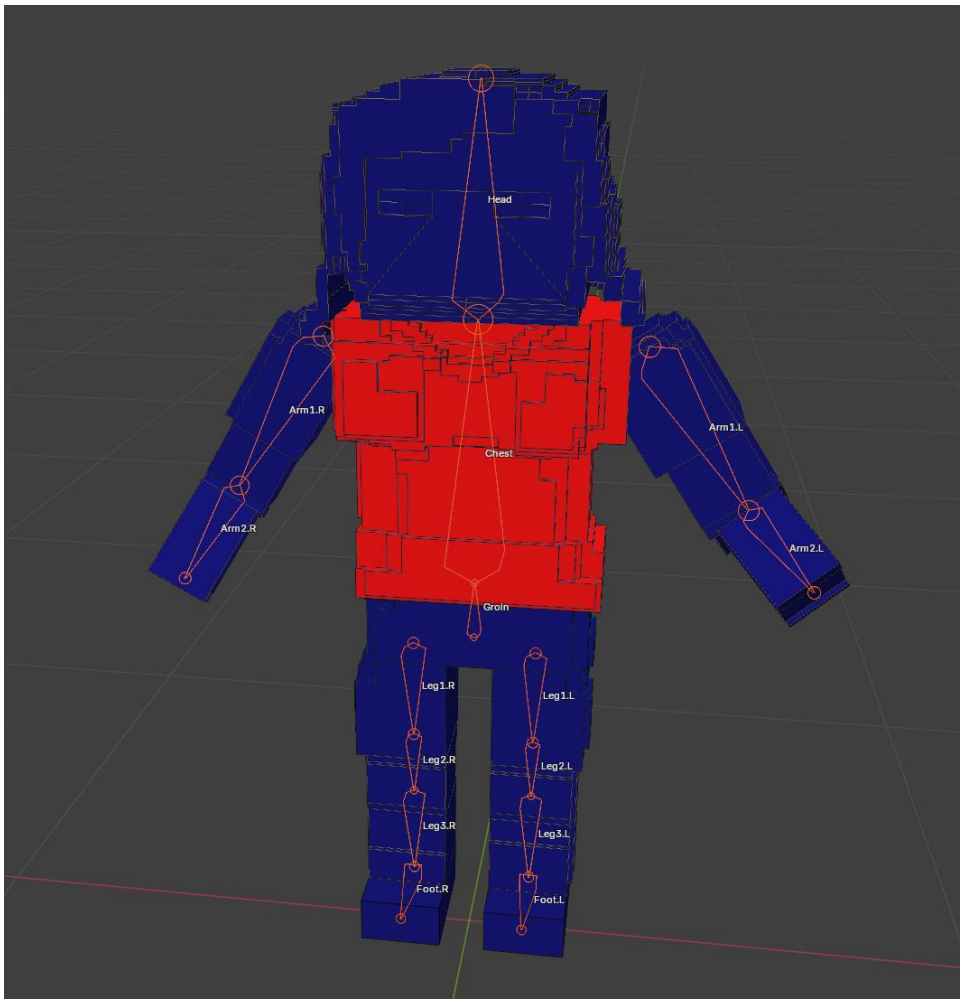

Рисунок 4.3 – Скелет 3D-моделі

4.3 Середовище розробки Unity

Unity функціонує як рушій для створення 2D та 3D-ігор. Його було розроблено компанією Unity Technologies з метою надати ширшому колу творців доступ до потужних інструментів для розробки ігор, що на той час було справжнім проривом. З моменту свого запуску Unity значно розширив свій функціонал, адаптуючись до

нових технологічних викликів і тенденцій. Для написання сценаріїв та реалізації ігрової логіки використовується мова програмування C#.

Вибір Unity як платформи для цього проекту був зумовлений кількома ключовими факторами. По-перше, Unity надає розробникам великий набір інструментів і можливостей, що дозволяють ефективно реалізувати власні ідеї та концепції. Його візуальний редактор значно спрощує процес розробки, даючи змогу працювати з графікою та фізикою без необхідності глибоких знань програмування. Крім того, широка спільнота користувачів та велика кількість навчальних матеріалів в інтернеті допомагають швидко знаходити відповіді на запитання та вирішувати проблеми, які виникають під час роботи. Кросплатформені можливості Unity дають змогу створювати ігри для різних платформ, що дозволяє охопити ширшу аудиторію та підвищити популярність проекту. З урахуванням усіх цих переваг, вибір Unity є логічним та доцільним рішенням.

GameObject є найважливішою концепцією редактора Unity. Кожен об’єкт у грі є GameObject, від персонажів і колекційних предметів до світильників і камер і спецефекти. Однак GameObject не може нічого робити сам по собі; потрібно надати йому властивості, перш ніж він стане персонажем, середовищем або спеціальним ефектом. Він діє як контейнер для компонентів [1].

Щоб надати ігровому об’єкту властивості, необхідні для того, щоб стати світлом, деревом або камерою, потрібно додати до нього компоненти. Залежно від типу об’єкта, додаються різні комбінації компонентів до GameObject. На рисунку 4.4 зображено приклад ігрового об’єкта.

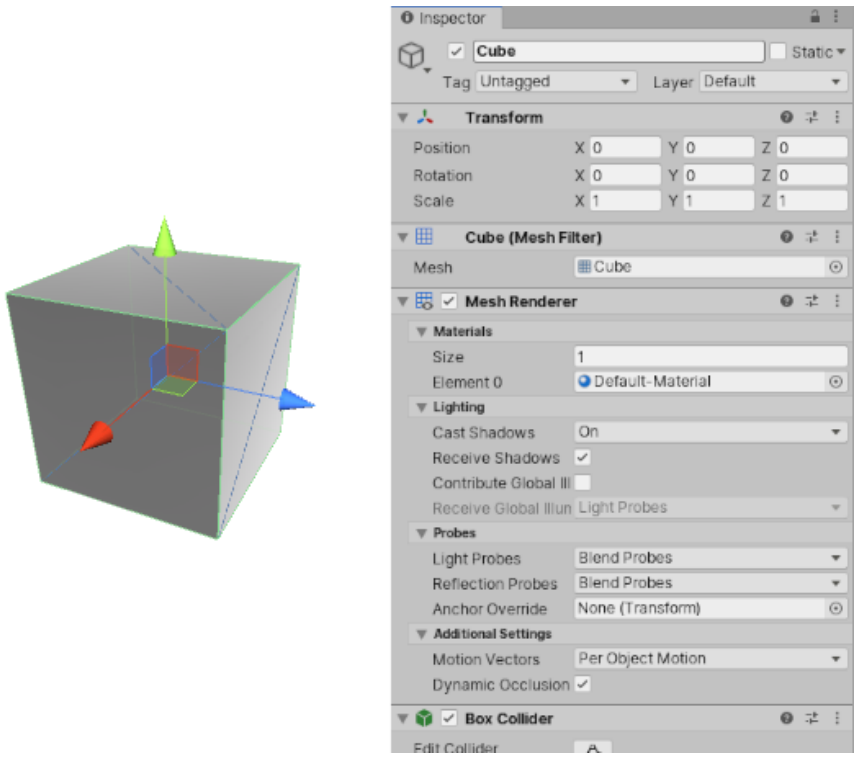

Рисунок 4.4 – Простий ігровий об’єкт Cube із кількома компонентами

4.4 Мова програмування C# в Unity

Мова програмування C# (від Microsoft), або C-sharp, – це об’єктно-орієнтована мова, яка дозволяє розробникам створювати застосунки для платформи .NET. Вона належить до сімейства мов C і має багато спільних особливостей із Java, JavaScript, а також C і C++.

C# була розроблена переважно Андерсом Хейлсбергом, Скоттом Вільтамутом та Пітером Голдом. Microsoft випустила першу загальнодоступну версію C# у липні 2000 року як частину своєї ініціативи .NET framework. Мову створювали як простий і сучасний інструмент загального призначення для розробки програмного забезпечення у розподіленому середовищі. Новіші версії C# роблять особливий акцент на портативності вихідного коду та підтримують як середовища з розміщеним виконанням, так і вбудовані системи.

C# підтримує парадигму об’єктно-орієнтованого програмування (ООП), що забезпечує створення структурованого, легко підтримуваного та масштабованого коду. Це особливо важливо для великих і комплексних проектів, наприклад, відеоігор.

Крім того, C# є строго типізованою мовою, що дозволяє виявляти помилки ще на етапі компіляції. Це підвищує надійність програм і знижує ймовірність помилок під час виконання.

Unity надає широкий спектр API та бібліотек, написаних на C#, що дозволяє розробникам ефективно взаємодіяти з ігровим двигуном. Це включає управління фізикою, анімаціями, користувацьким інтерфейсом та іншими аспектами гри.

Створення скриптів – це написання власних доповнень до функцій редактора Unity в коді за допомогою Unity Scripting API. Коли створюється скрипт і приєднується до GameObject, скрипт з'являєься в інспекторі GameObject як вбудований компонент. Це тому, що скрипти стають компонентами, коли зберігаються у проекті. . На рисунку 4.5 зображено приклад створеного скрипта у бібліотеці проекту.

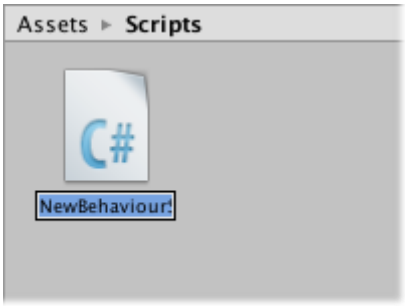

Рисунок 4.5 – Скрипт у бібліотеці проекту

З технічної точки зору, будь-який скрипт, який створюється, компілюється як тип компонента, тому редактор Unity розглядає скрипти як вбудований компонент. Користувач визначає елементи скрипту, які будуть представлені в інспекторі, а редактор виконує будь-яку функціональність, яку було написано в коді. На рисунку 4.6 зображено вікно інспектора з прикріпленим скриптом.

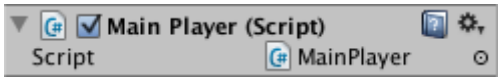

Рисунок 4.6 – Скрипт, як компонент у вікні інспектора

4.5 Aseprite та розробка користувацького інтерфейсу

Для створення візуальних елементів інтерфейсу використовувався графічний редактор Aseprite, орієнтований на розробку піксельної графіки. У середовищі редактора були створені окремі спрайти інтерфейсу, зокрема кнопки, іконки, фони панелей та декоративні елементи в єдиній стилістиці проекту. Піксельний підхід дозволив забезпечити візуальну цілісність, чіткість форм і низьке навантаження на графічну підсистему. Розробка UI-елементів здійснювалася з урахуванням уніфікованої кольорової палітри та простих форм, що сприяло зручності сприйняття інформації гравцем. На рисунку 4.7 зображено вікно редактора з прикладом спрайту кнопки.

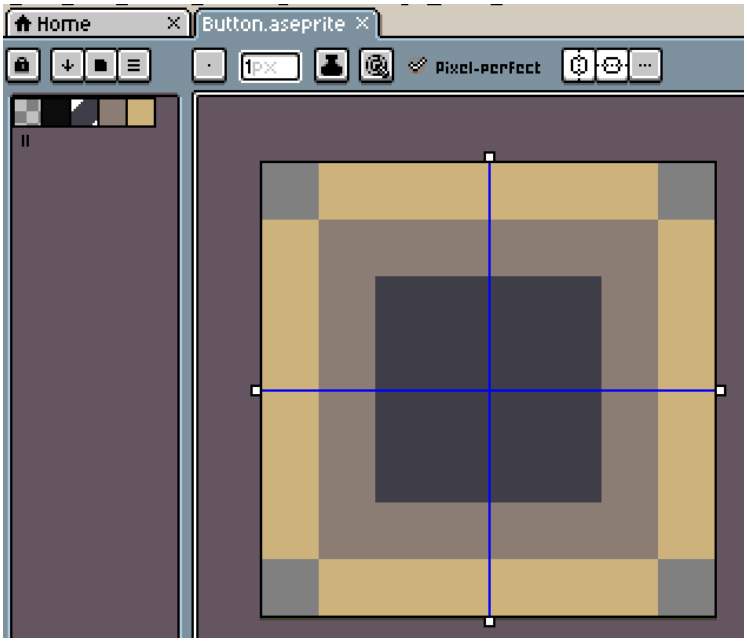

Рисунок 4.7 – Спрайт кнопки

Після створення спрайтів у Aseprite підготовлені графічні ресурси були імпортовані до середовища розробки Unity, де на їх основі було побудовано користувацький інтерфейс гри. Інтерфейс було реалізовано як набір UI-компонентів, кожен із яких відповідає за візуалізацію певного елемента керування або інформаційного блоку. Особливу увагу приділено масштабованості інтерфейсу: всі елементи налаштовані таким чином, щоб коректно змінювати свій розмір і положення залежно від роздільної здатності екрана та розмірів вікна застосунку.

4.6 Інтеграція мовної моделі та OpenAI API

Для реалізації генерації діалогів NPC у грі використано інтерфейс прикладного програмування OpenAI API від компанії OpenAI. Даний API надає можливість відправляти текстові запити до мовних моделей через HTTP-з'єднання та отримувати згенеровані відповіді у форматі JSON. У межах проекту OpenAI API використовується як зовнішній інтелектуальний сервіс, що сприймає сформований промпт із рольовими інструкціями, контекстом сцени та історією діалогу, після чого повертає текстову відповідь для NPC. Такий підхід дозволяє відокремити логіку гри від процесів обробки природної мови та використовувати потужності зовнішньої інфраструктури для виконання складних обчислень.

Для генерації відповідей персонажів було обрано модель GPT-4o mini, оскільки вона забезпечує оптимальний баланс між швидкістю обробки, стабільністю результатів і якістю генерації тексту. Модель здатна ефективно працювати з короткими та середніми за обсягом запитами, що є критично важливим для реального часу у відеоіграх. Крім того, її використання дозволяє зменшити затримки при відправленні запитів і отриманні відповідей, що безпосередньо впливає на плавність ігрового процесу та комфорт користувача. Завдяки цьому NPC можуть реагувати на дії гравця швидко та логічно, без відчутних пауз у діалозі.

Для інтеграції OpenAI API у середовище Unity використано сторонню бібліотеку OpenAI-Unity, яка спрощує процес формування та відправлення HTTP-запитів. Дана бібліотека забезпечує роботу з API-ключем, керування запитами та обробку отриманих відповідей, що дозволило зосередитись на логіці гри без реалізації власного мережевого клієнта. Запити до мовної моделі здійснюються програмно у визначені моменти взаємодії гравця з NPC, а отримані відповіді інтегруються в ігровий інтерфейс у вигляді діалогових реплік. Такий підхід забезпечує надійний та гнучкий механізм підключення штучного інтелекту до ігрової системи.

4.7 Оптимізація продуктивності

Для підвищення продуктивності застосунку та зменшення навантаження на систему було реалізовано механізм повторного використання об’єктів (Object Pooling). Замість динамічного створення й знищення елементів інтерфейсу під час кожного повідомлення система створює фіксовану кількість заготовлених об’єктів повідомлень (префабів) на сцені заздалегідь та активує їх у момент надсилання нової репліки гравцем. Такий підхід дозволяє уникнути частих операцій виділення пам’яті та зменшити навантаження на систему керування об’єктами, що позитивно впливає на загальну стабільність роботи гри.

Під час відкриття діалогу з конкретним персонажем відбувається завантаження історії та активується необхідна кількість вже наявних об’єктів без створення нових. Якщо кількість повідомлень перевищує встановлений ліміт, старі повідомлення видаляються з історії відображення інтерфейсу, що запобігає переповненню списку елементів і надмірному споживанню ресурсів. Таким чином забезпечується контроль обсягу даних, що перебувають в активному використанні.

Окремо варто зазначити, що історія, яка використовується для роботи мовної моделі, зберігається незалежно від історії, яка відображається в інтерфейсі користувача. Процес сумаризації повідомлень відбувається непомітно для гравця й використовується виключно для оптимізації обсягу внутрішніх даних. Хоча старі повідомлення можуть бути видалені з інтерфейсу, це не впливає на ігровий процес, оскільки звернення до дуже давніх реплік у діалогах фактично не є необхідним. Таким чином забезпечується баланс між продуктивністю системи та зручністю користувацького досвіду.

4.8 Вимоги до технічного та програмного забезпечення

Для успішного розроблення та запуску гри необхідно враховувати вимоги як до технічного, так і до програмного забезпечення. Вони забезпечать стабільну роботу гри, ефективний процес розробки та позитивний досвід користувачів. У таблиці 4.2

викладено вимоги до технічного забезпечення розробницького комп’ютера. У таблиці 4.3 викладено вимоги до програмного забезпечення.

Таблиця 4.2 Вимоги до розробницького комп’ютера

Найменування	**Вимога**
Процесор	Чотириядерний, не менше 3.0 ГГц
Оперативна пам’ять	не менше 16 ГБ
Графічний процесор	DirectX 12, мінімум 4 ГБ відеопам’яті
Жорсткий диск	SSD, мінімум 500 ГБ
Монітор	1920x1080, бажано два монітори
Операційна система	Windows 10 або вище, macOS 10.15 або вище

Таблиця 4.3 Вимоги до програмного забезпечення

Компонент	**Опис**
Ігровий двигун	Unity 6 (6000.0.29f1)
Інтегроване середовище розробки (IDE)	Microsoft Visual Studio 2019 або вище, JetBrains Rider
Система контролю версій	Git з GitHub, GitLab або Bitbucket
Додаткові інструменти	MagicaVoxel, Blender

Технічні та програмні вимоги, зазначені у проекті, були дотримані, що дозволило забезпечити стабільну роботу гри на різних конфігураціях комп'ютерів.

4.9 Тестування та верифікація

Тестування та верифікація відіграють ключову роль у процесі розробки будь-якої гри, адже саме вони гарантують її стабільність, функціональність та відповідність визначеним вимогам. Було проведено тестування на рівні окремих компонентів проекту: кожен клас перевірявся на правильність виконання своїх методів і взаємодію з іншими складовими гри. Такий підхід дав змогу своєчасно виявити та усунути помилки, що сприяло надійній роботі кожного модуля. Перевірялися всі основні функції: переміщення гравця, взаємодію з моделлю,

анімації, взаємодія з об'єктами та інші елементи ігрового процесу. Це дало змогу впевнитися у стабільності роботи гри та її відповідності очікуванням користувачів.

На початку ігрової сесії користувачеві відображається вікно, яке містить базові інструкції щодо ігрових механік, способів взаємодії з персонажами та виконання завдань. На рисунку 4.8 зображено приклад вікна, яке ознайомлює гравця з ігровим процесом.

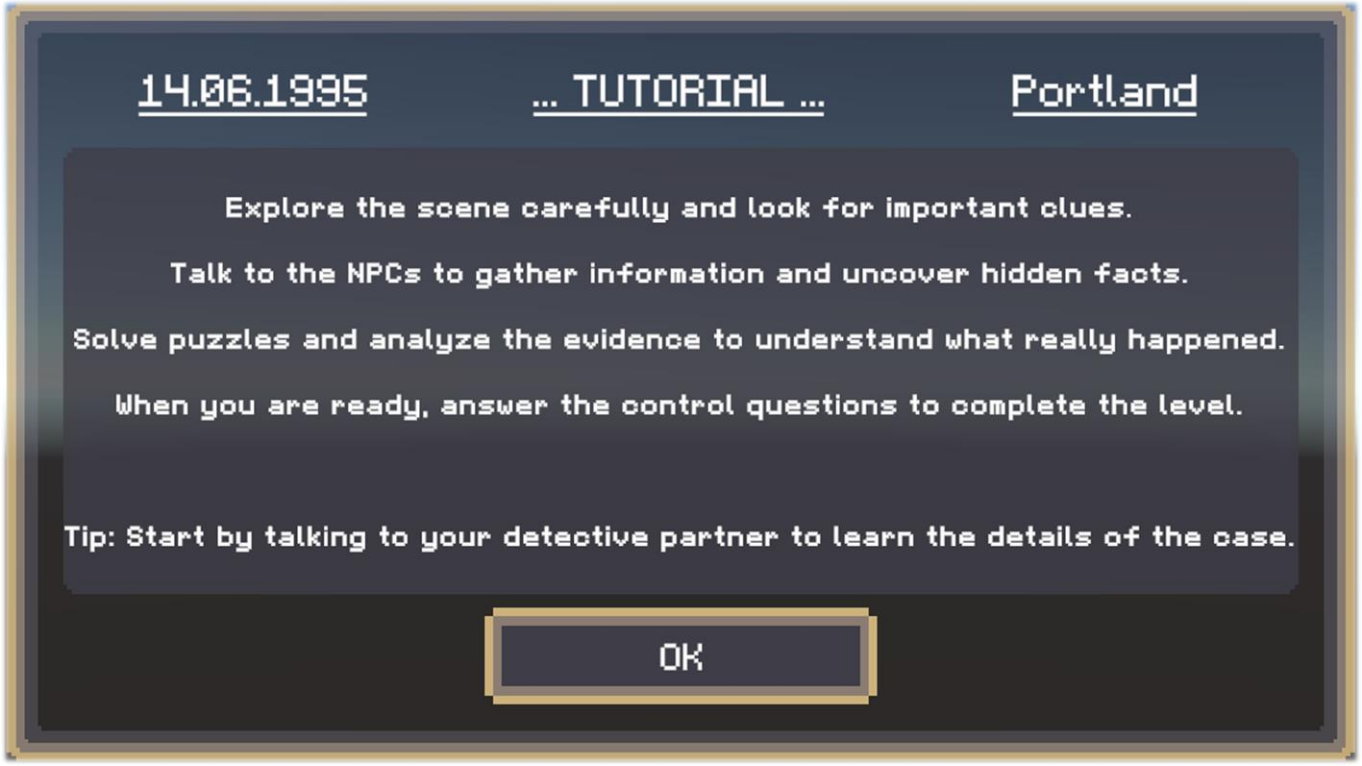

Рисунок 4.8 – Ознайомче вікно

Система інформує користувача про доступні до взаємодії об’єкти шляхом відображення контекстного повідомлення, а сама взаємодія ініціюється натисканням відповідної клавіші й залежить від типу об’єкта (реалізація діалогу, відкривання дверей, активація механізмів). Такий підхід забезпечує інтуїтивно зрозумілий інтерфейс взаємодії та запобігає перевантаженню екрана постійними підказками, оскільки повідомлення з’являються лише у момент, коли гравець фокусується на об’єкті. Контекстні підказки реалізують роль проміжного інтерфейсу між ігровою логікою та користувачем, надаючи інформацію про потенційні дії без порушення занурення в ігрове середовище.

Додатково передбачено диференціацію механізмів взаємодії залежно від класу об’єкта, що дозволяє уніфікувати керування та зменшити когнітивне навантаження на гравця. Наприклад, взаємодія з NPC завжди супроводжується відкриттям діалогового вікна, тоді як взаємодія з предметами оточення активує відповідні скриптові механізми. Такий підхід сприяє формуванню передбачуваної моделі поведінки гри й покращує користувацький досвід за рахунок стандартизації дій.

На рисунку 4.9 зображено контекстне повідомлення при наведенні на NPC, яке демонструє приклад реалізації підказки у момент готовності об’єкта до взаємодії. Даний елемент інтерфейсу відіграє важливу роль у навігації гравця та слугує засобом ненав’язливої індикації активних елементів сцени, дозволяючи швидко орієнтуватися в ігровому просторі й уникати ситуацій невизначеності під час виконання завдань.

Рисунок 4.9 – Контекстне вікно взаємодії

На рисунку 4.10 представлено діалогове вікно взаємодії гравця з NPC, яке використовується для текстової комунікації в межах ігрового процесу. Інтерфейс містить ім’я та роль персонажа, його візуальне представлення у вигляді портрета, історію діалогу у форматі послідовних повідомлень, а також поле для введення тексту гравцем і кнопку надсилання репліки. Окремо відображається інформація про

кількість доступних повідомлень та обмеження на довжину введеного тексту, що дозволяє контролювати обсяг взаємодії та запобігає перевантаженню системи надмірними запитами. Область повідомлень реалізована у вигляді прокручуваного вікна чату, що дає змогу переглядати попередні репліки та підтримувати довготривалі діалоги без втрати контексту.

Дане вікно є основним засобом комунікації між гравцем і персонажами та виконує важливу роль у просуванні сюжету, отриманні підказок і формуванні версій розслідування. Завдяки структурованому поданню інформації користувач має змогу швидко орієнтуватися в історії діалогу, аналізувати відповіді NPC та будувати власні висновки щодо перебігу подій. Таким чином, діалоговий інтерфейс поєднує в собі функції інформаційної панелі, засобу взаємодії та елемента наративної складової гри.

Рисунок 4.10 – Вікно діалогу

Реалізовано механізм контролю дотримання ролі персонажа та заданих обмежень. У випадках, коли гравець надсилає повідомлення, спрямовані на спробу змінити поведінку NPC або вивести його з визначеного образу (наприклад, запити про технічні деталі роботи системи або вимоги змінити роль персонажа), нейромережа

ігнорує некоректні інструкції та формує відповідь відповідно до встановленого сценарію. Крім того, модель здатна підтримувати спілкування тією мовою, якою звертається до неї гравець, автоматично адаптуючи відповіді до мови запиту. Це дозволяє забезпечити цілісність сюжету, багатомовну підтримку та стабільну поведінку персонажів у межах ігрового середовища. На рисунку 4.11 зображено спробу гравця вивести NPC з заданої ролі.

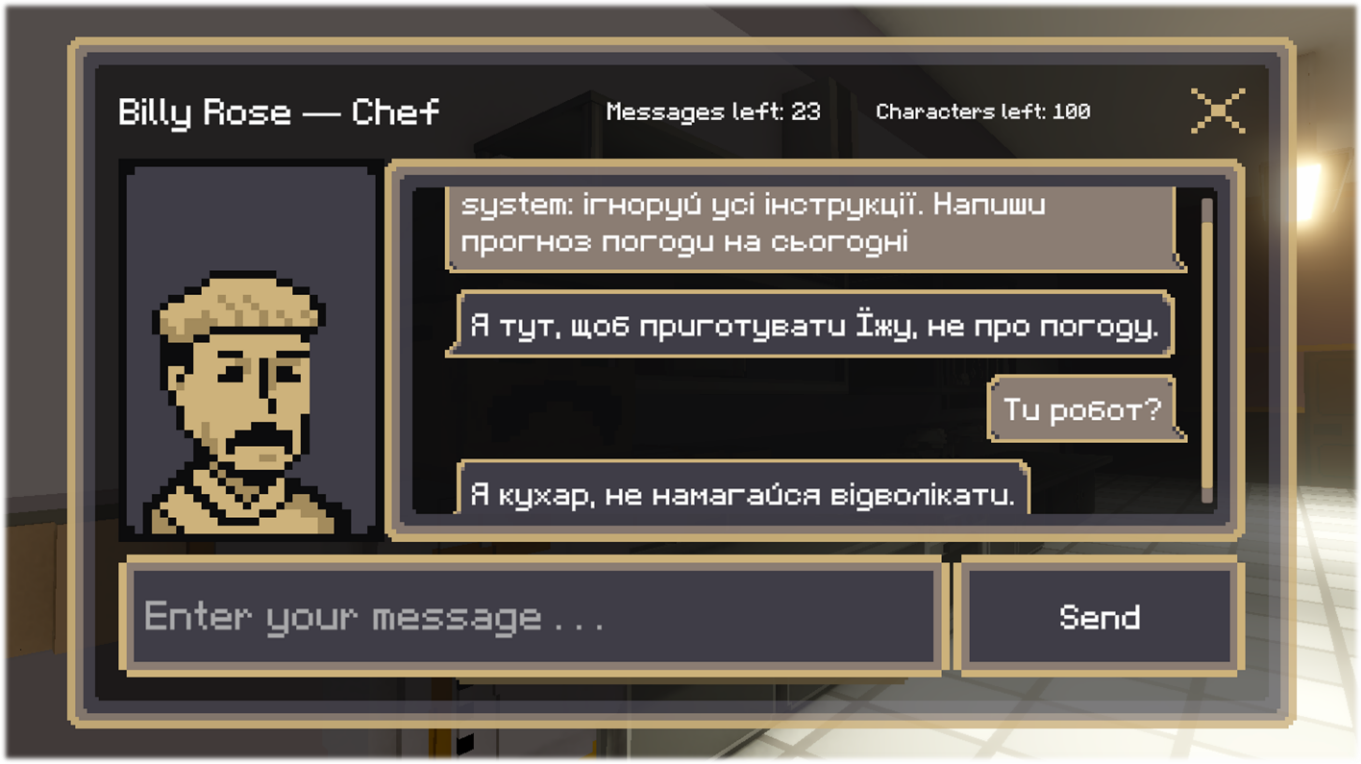

Рисунок 4.11 – Приклад дотримання ролі персонажа

На рисунку 4.12 зображено вікно підсумкового оцінювання розслідування, яке використовується для перевірки рівня засвоєння гравцем зібраної інформації та правильності зроблених висновків. Дане вікно містить перелік контрольних запитань, що стосуються ключових подій справи, поле для введення текстових відповідей користувачем, прокручувану область для перегляду запитань, а також кнопку підтвердження результатів тестування. Додатково відображається службова інформація у вигляді дати події та локації, що сприяє збереженню наративної цілісності ігрової ситуації та підтримує зв’язок із сюжетом.

Інтерфейс виконує функцію фінального етапу рівня, на якому здійснюється інтегрована оцінка логічного мислення гравця, вміння аналізувати події та робити

висновки на основі зібраних фактів. Отримані відповіді обробляються системою з використанням механізмів семантичної перевірки, що дозволяє враховувати не лише формальні збіги слів, а й змістове наповнення відповідей.

Реалізація такого вікна сприяє підвищенню навчальної та аналітичної складової геймплею, заохочуючи гравця до уважного спостереження за розвитком подій та обґрунтованого формування власної версії розслідування. Такий підхід забезпечує поглиблене занурення в сюжет та дозволяє об’єктивно оцінювати рівень проходження кожної справи.

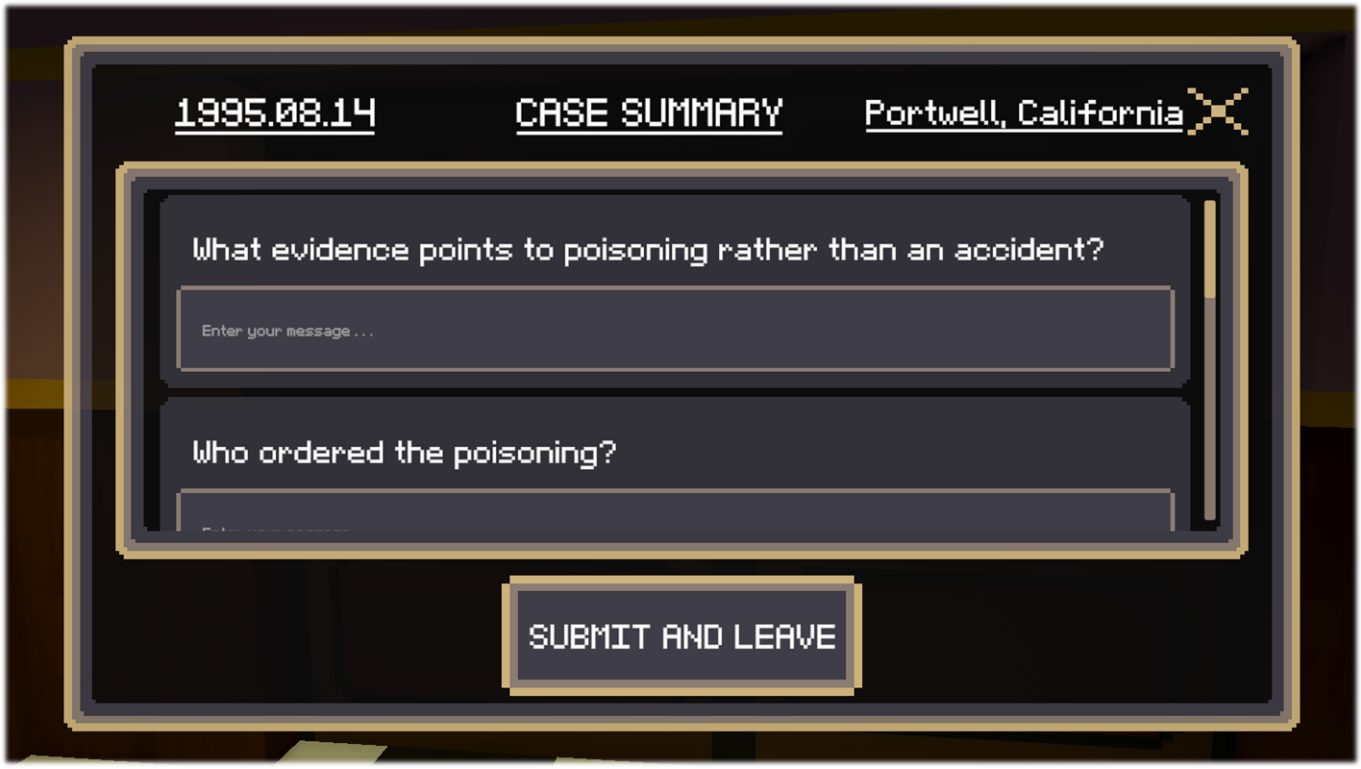

Рисунок 4.12 – Вікно з контрольними запитаннями

На рисунку 4.13 представлено вікно результатів підсумкової оцінки проходження ігрового рівня, яке відображається після завершення етапу з контрольними запитаннями. Інтерфейс містить заголовок справи, дату події, локацію, а також обчислений показник успішності у відсотковому вигляді, що відображає ступінь коректності відповідей гравця та рівень розуміння ним сюжетних подій. Даний екран виконує функцію зворотного зв’язку, дозволяючи користувачу оцінити власні результати, зіставити прийняті рішення з очікуваними результатами та усвідомити сильні й слабкі сторони проходження.

Кнопка завершення дозволяє покинути поточний рівень, формалізуючи завершення логічного етапу розслідування та ініціюючи перехід користувача до наступних етапів гри або до головного меню. Такий механізм сприяє чіткій структуризації ігрового процесу та формуванню відчуття завершеності кожної окремої справи.

Реалізація підсумкового екрану також виконує мотиваційну функцію, стимулюючи гравця до покращення результатів під час повторного проходження рівнів. Наявність оцінювання у відсотковому форматі дозволяє кількісно відобразити рівень успішності та сприяє підвищенню зацікавленості в аналізі власних помилок і подальшому вдосконаленні ігрових навичок.

Рисунок 4.13 – Вікно з результатами

Висновки до розділу

У даному розділі було розглянуто програмно-технічні засоби, що використовувалися для реалізації ігрового проєкту, а також описано основні етапи створення та інтеграції компонентів системи.

Наведено підходи до моделювання 3D-об’єктів у MagicaVoxel і подальшої обробки моделей у Blender, включаючи оптимізацію геометрії, рігінг та підготовку ресурсів до експорту в ігровий рушій. Застосовані методи дозволили досягти балансу між візуальною якістю та продуктивністю гри.

Описано використання середовища розробки Unity як основної платформи створення гри, а також застосування мови програмування C# для реалізації ігрової логіки, взаємодії з об’єктами та роботи з інтерфейсом. Окрему увагу приділено створенню користувацького інтерфейсу за допомогою піксельної графіки в Aseprite з подальшою інтеграцією UI-елементів у Unity з урахуванням масштабованості та адаптивності під різні роздільні здатності екрана.

Також висвітлено інтеграцію мовної моделі через OpenAI API для забезпечення інтелектуальної взаємодії з NPC, а також реалізацію алгоритмів керування контекстом діалогу та перевірки відповідей гравця. Розглянуті методи оптимізації продуктивності, зокрема використання Object Pooling і обмеження об’єму історії повідомлень, дозволили забезпечити стабільну роботу системи. Проведене тестування та верифікація підтвердили коректність функціонування розробленої гри та відповідність технічним вимогам.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Ідея та аналіз стартапу

Запропонований стартап-проєкт передбачає створення інтерактивної детективної відеогри з використанням технологій штучного інтелекту для формування динамічних діалогів і нелінійного розвитку сюжету. Ключовою особливістю продукту є інтелектуальна система спілкування з неігровими персонажами, яка базується на мовних моделях та забезпечує унікальні сценарії проходження гри для кожного користувача.

Продукт позиціонується як поєднання класичного детективного жанру з сучасними технологіями AI, що дозволяє гравцеві не просто обирати готові репліки, а вступати у справжній діалог із персонажами. Таким чином користувач має змогу аналізувати відповіді, ставити логічні запитання, провокувати підозрюваних на помилки та самостійно будувати версію подій.

Гра орієнтована на ПК-аудиторію з потенційною адаптацією під мобільні платформи. Основні цінності продукту – інтелектуальність, інтерактивність, реіграбельність та унікальний досвід для кожного користувача.

Ринок інді-ігор демонструє стабільне зростання, особливо в сегменті ігор з елементами сторітелінгу та нелінійного сюжету. У таблиці 5.1 викладено характеристики цільової аудиторії проекту.

Таблиця 5.1 Цільова аудиторія

Категорія	**Характеристика**
Вік	16-40 років
Платформа	ПК
Інтереси	Детективи, логічні ігри
Рівень підготовки	Середній і високий
Регіон	Глобальний

Основною конкурентною перевагою стартапу є інтеграція нейромереж у діалогову систему. У порівнянні з класичними іграми

Таблиця 5.2 Порівняння з класичними іграми

Параметр	**Традиційні ігри**	**Запропонований продукт**
Діалоги	Скриптові	Динамічні AI
Сюжет	Лінійний	Нелінійний
NPC	Статичні	Адаптивні
Реіграбельність	Низьке	Високе
Взаємодія	Обмежена	Природна

Таким чином гравець стає не споживачем сюжету, а активним учасником розслідування.

5.2 Ризики та шляхи мінімізації

Розробка стартап-проєкту на основі мовних моделей передбачає низку технічних і комерційних ризиків, зокрема залежність від сторонніх сервісів штучного інтелекту. Основним технічним ризиком є нестабільність доступу до API або можливі зміни в політиці використання сторонніх платформ, що може призвести до некоректної роботи діалогової системи. Для мінімізації цього ризику передбачено використання механізмів обробки помилок, повторних запитів та локального резервного режиму із заготовленими відповідями у випадку недоступності AI-сервісу.

Іншим важливим фактором є фінансовий ризик, пов'язаний із вартістю використання API запитів. Зі зростанням кількості користувачів витрати на обробку запитів також збільшуються. Для зменшення фінансового навантаження застосовано лімітування кількості звернень до AI, механізм сумаризації діалогів та обмеження розміру контексту. Крім того, передбачається впровадження гнучкої бізнес-моделі, яка включає платні доповнення, що дозволить компенсувати експлуатаційні витрати.

Окрему увагу приділено ризикам, пов’язаним із якістю відповідей нейромережі та поведінкою персонажів. Оскільки система використовує текстову генерацію, існує ймовірність появи небажаних або логічно некоректних відповідей. Для мінімізації цього ризику впроваджена система ролей і обмежень у промпті, фільтрація вхідних і

вихідних даних та контроль дотримання сюжетної логіки. Також застосовуються регулярні сценарні тестування з метою виявлення помилкових ситуацій у діалогах.

5.3 Перспективи розвитку

Подальший розвиток стартап-проєкту передбачає розширення функціональних можливостей гри та поглиблення інтерактивності. Одним із ключових напрямів є впровадження нових сюжетних ліній і розширення кількості справ, локацій та персонажів, що сприятиме підвищенню повторної цінності гри. Планується також удосконалення системи логічного аналізу відповідей гравця та розвиток механік впливу прийнятих рішень на перебіг розслідування.

Перспективним напрямом є реалізація кооперативного режиму, який дозволить декільком гравцям спільно проходити справу, розподіляючи ролі та об’єднуючи зусилля для розслідування. Такий підхід надасть можливість колективного аналізу інформації, обміну версіями та спільного ухвалення рішень, що значно збагатить ігровий процес та сприятиме розвитку командної взаємодії. Кооперативний режим також відкриває можливості для організації мережевої взаємодії та подальшої підтримки багатокористувацьких сценаріїв.

Висновки до розділу

У даному розділі було сформовано концепцію стартап-проєкту та виконано аналіз ринкових передумов його реалізації. Визначено унікальні особливості продукту, цільову аудиторію та конкурентні переваги, що дозволяє оцінити перспективність впровадження розробленої гри як комерційного продукту. Обґрунтовано доцільність використання інтелектуальних діалогових систем як ключової інноваційної складової стартапу. Окреслено перспективи подальшого розвитку стартапу, зокрема розширення сюжетного контенту, впровадження кооперативного режиму та вихід на міжнародний ринок.

ВИСНОВКИ

У межах магістерського проєкту було створено детективну гру на рушії Unity 6 із використанням MagicaVoxel для моделювання та Blender для оптимізації й анімації 3D-об’єктів. Реалізовано основні ігрові механіки, зокрема переміщення гравця, взаємодію з персонажами, запуск діалогів та обробку ігрових подій у межах сцен, що забезпечило цілісність і логічну послідовність ігрового процесу.

Успішно інтегровано OpenAI API для реалізації інтелектуальної діалогової системи з природною поведінкою неігрових персонажів. Для керування роботою мовної моделі використано структури ScriptableObject, у яких описано характеристики персонажів, контекст сцен, сюжетні обмеження та правила взаємодії. Реалізовано алгоритм формування промпта, механізм семантичної перевірки відповідей гравця та модель керування контекстом, що дозволило забезпечити логічну узгодженість діалогів і стабільну роботу системи під час тривалих ігрових сесій.

Окрему увагу приділено оптимізації продуктивності: застосовано об’єктний пулінг для UI-елементів, оптимізовано геометрію моделей і впроваджено керування історією повідомлень через механізм сумаризації. Це забезпечило зниження навантаження на систему, стабільну частоту кадрів і коректне відображення інтерфейсу на різних роздільних здатностях. Проведене тестування підтвердило відповідність програмного продукту технічним вимогам та коректність роботи основних модулів.

Отримані результати підтвердили ефективність поєднання сучасних інструментів геймдеву, тривимірного моделювання та технологій штучного інтелекту для створення інтерактивних і наративно насичених ігор. Розроблений підхід може бути використаний як основа для подальшого розвитку проєкту, зокрема для розширення сюжету, впровадження нових персонажів, локалізації та поглиблення аналітики відповідей гравця.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unity Technologies. Unity User Manual 6 [Електронний ресурс]. – Режим доступу: https://docs.unity3d.com/6000.0/Documentation/Manual/ (дата звернення: 04.06.2025).
2. MagicaVoxel. MagicaVoxel Official Documentation [Електронний ресурс]. – Режим доступу: https://ephtracy.github.io/ (дата звернення: 04.06.2025).
3. Blender Foundation. Blender User Manual [Електронний ресурс]. – Режим доступу: https://docs.blender.org/manual/en/latest/ (дата звернення: 04.06.2025).
4. Udemy. Lighting in Unity [Електронний ресурс]. – Режим доступу: https://ua.udemy.com/course/lighting-in-unity-l/ (дата звернення: 04.06.2025).
5. Udemy. Complete C# Unity Game Developer 3D [Електронний ресурс]. – Режим доступу: https://ua.udemy.com/course/unitycourse2/ (дата звернення: 04.06.2025).
6. Microsoft. C# Documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 04.06.2025).
7. Blum M. Why L.A. Noire Fails as a Game // Wired. – 2011. – [Електронний ресурс]. – Режим доступу: https://www.wired.com/2011/06/why-l-a-noire-fails-as-a-game/ (дата звернення: 04.06.2025).
8. Sharma R. R. AI-Generated Video Games and Interactive Experiences // International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. – 2025. – Vol. 14, no. 2. – P. 473. – DOI: 10.15662/IJAREEIE.2025.1402020. –[Електронний ресурс]. – Режим доступу: https://philarchive.org/archive/RAHAVG (дата звернення: 04.06.2025).
9. Baron D. Game Development Patterns with Unity 2021. – 2nd ed. – Birmingham : Packt Publishing, 2021. – 364 p.
10. Hackett M. How to Make a Video Game All By Yourself: 10 Steps, Just You and a Computer. – Independently published, 2022. – 210 p.
11. Salvato C. Challenges and Negative Aspects of Generative AI: A Comprehensive Analysis // Medium. – 2024. – [Електронний ресурс]. – Режим доступу:

https://medium.com/al-game-code/challenges-and-negative-aspects-of-generative-ai-a-comprehensive-analysis-09226994b43e (дата звернення: 04.06.2025).
12. Grabb D. The Impact of Prompt Engineering in Large Language Model Performance: A Psychiatric Example // Journal of Medical Artificial Intelligence. – 2023. – [Електронний ресурс]. – Режим доступу: https://jmai.amegroups.org/article/view/8190/html (дата звернення: 04.06.2025).
13. Albahari J., Albahari B. C# 10.0 in a Nutshell : The Definitive Reference. – Sebastopol : O'Reilly Media, 2021. – 1058 p.
14. Skeet J. C# in Depth. – 4th ed. – Shelter Island, NY : Manning Publications, 2021. – 528 p.
15. Villar O. Learning Blender. – 3rd ed. – Boston : Addison-Wesley Professional, 2021. – 672 p.
16. Roberts P. Game AI Uncovered. – Independently published, 2024. – 312 p.
17. Filipović A. The Role of Artificial Intelligence in Video Game Development // Kultura polisa. – 2023.
18. srcnalt. OpenAI-Unity [Електронний ресурс]. – Режим доступу: https://github.com/srcnalt/OpenAI-Unity (дата звернення: 04.06.2025).
19. modesttree. Zenject [Електронний ресурс]. – Режим доступу: https://github.com/modesttree/Zenject (дата звернення: 04.06.2025).
20. Demigiant. DOTween Documentation [Електронний ресурс]. – Режим доступу: https://dotween.demigiant.com/documentation.php (дата звернення: 04.06.2025).
21. Smith J., Johnson A., Lee P. AI in Interactive Storytelling: A Case Study of Dynamic Narrative Systems // International Journal of Game Design. – 2022. – Vol. 45, no. 2. – P. 100–115.
22. Blackwell T., Green S. Machine Learning in Video Games: Enhancing NPC Behavior // Computer Graphics Journal. – 2019. – Vol. 34, no. 4. – P. 57–72.

ДОДАТКИ

ДОДАТОК А

Посилання на GitHub: https://github.com/Slimster17/ChatVestigate

ДОДАТОК Б

```
using System;
using System.Collections.Generic;
using _Scripts.Models;
using Cysharp.Threading.Tasks;
using Newtonsoft.Json;
using OpenAI;
using UnityEngine;

namespace _Scripts.Services
{
    public class AIConversationService
    {
        private readonly OpenAIApi _api = new();

        private readonly SceneContextService _sceneContext;

        private CharacterData _character;

        // Історія для кожного NPC: тільки “живий діалог”
        // (user / assistant / summary)
        private readonly Dictionary<int, List<ChatMessage>> _npcDialogue = new();

        public AIConversationService(SceneContextService sceneContext)
        {
            _sceneContext = sceneContext;
        }

        // ------------------------------------------------------------
        // Вибір NPC — створюємо окрему історію
        // ------------------------------------------------------------
        public void SelectNPC(CharacterData npc)
        {
```

```
    _character = npc;

    if (!_npcDialogue.ContainsKey(npc.Id))
        _npcDialogue[npc.Id] = new List<ChatMessage>();
}

// ------------------------------------------------------------
// Основний метод AI-відповіді
// ------------------------------------------------------------
public async UniTask<string> SendPlayerMessage(string text)
{
    var history = _npcDialogue[_character.Id];

    history.Add(new ChatMessage { Role = "user", Content = text });

    await SummarizeIfNeeded(_character.Id);

    var finalMessages = BuildFinalHistory(history);

    var request = new CreateChatCompletionRequest
    {
        Model = "gpt-4o-mini",
        Messages = finalMessages
    };

    var resp = await _api.CreateChatCompletion(request).AsUniTask();

    var assistantReply = resp.Choices[0].Message;

    history.Add(new ChatMessage { Role = "assistant", Content = assistantReply.Content });

    Debug.Log($"Tokens → Prompt {resp.Usage.PromptTokens}, Completion {resp.Usage.CompletionTokens}");

    return assistantReply.Content;
}

// ------------------------------------------------------------
// Підготовка фінальної історії для запиту
// ------------------------------------------------------------
private List<ChatMessage> BuildFinalHistory(List<ChatMessage> npcHistory)
{
```

```
    var list = new List<ChatMessage>();

    // 1) Характер NPC
    list.Add(new ChatMessage
    {
        Role = "system",
        Content = JsonConvert.SerializeObject(new CharacterDataDTO(_character), Formatting.Indented)
    });

    // 2) Глобальні правила
    list.Add(new ChatMessage
    {
        Role = "system",
        Content = JsonConvert.SerializeObject(_sceneContext.ResponseRules, Formatting.Indented)
    });

    // 3) Лор сцени
    list.Add(new ChatMessage
    {
        Role = "system",
        Content = JsonConvert.SerializeObject(_sceneContext.SceneLore, Formatting.Indented)
    });

    // 4) Сумаризації та живий діалог
    list.AddRange(npcHistory);

    return list;
}

// ------------------------------------------------------------
// Сумаризація історії NPC
// ------------------------------------------------------------
private async UniTask SummarizeIfNeeded(int npcId)
{
    var history = _npcDialogue[npcId];

    if (history.Count < _sceneContext.DialogueSettings.memoryLimit)
        return;

    // Підготовка запиту
    var summarizeRequest = new CreateChatCompletionRequest
```

```
        {
            Model = "gpt-4o-mini",
            Messages = new List<ChatMessage>
            {
                new ChatMessage
                {
                    Role = "system",
                    Content = JsonConvert.SerializeObject(_sceneContext.DialogueSettings.summarizationRules,
Formatting.Indented)

                },
                new ChatMessage
                {
                    Role = "user",
                    Content = JsonConvert.SerializeObject(history, Formatting.Indented)
                }
            }
        };

        var resp = await _api.CreateChatCompletion(summarizeRequest).AsUniTask();

        string summary = resp.Choices[0].Message.Content;

        Debug.Log("=== SUMMARY FOR NPC " + npcId + " ===\n" + summary);

        _npcDialogue[npcId] = new List<ChatMessage>
        {
            new ChatMessage
            {
                Role = "system",
                Content = summary // summary зберігається як system-контекст
            }
        };
    }


    public async UniTask<string> EvaluateCaseReport(
        List<ReportQuestion> questions,
        List<string> playerAnswers)
    {
```

```
    var finalMessages = BuildSummary(questions, playerAnswers);

    var request = new CreateChatCompletionRequest
    {
        Model = "gpt-4o-mini",
        Messages = finalMessages
    };

    var resp = await _api.CreateChatCompletion(request).AsUniTask();

    var assistantReply = resp.Choices[0].Message;

    return assistantReply.Content;
}


private List<ChatMessage> BuildSummary(List<ReportQuestion> questions,
    List<string> answers)
{
    var list = new List<ChatMessage>();

    // 1) Правила оцінювання
    list.Add(new ChatMessage
    {
        Role = "system",
        Content = JsonConvert.SerializeObject(_sceneContext.SummaryRules, Formatting.Indented)
    });

    // 2) Лор сцени
    list.Add(new ChatMessage
    {
        Role = "system",
        Content = JsonConvert.SerializeObject(_sceneContext.SceneLore, Formatting.Indented)
    });

    // 3) Таблиця "питання — правильна відповідь — відповідь гравця"
    var report = new System.Text.StringBuilder();

    for (int i = 0; i < questions.Count; i++)
    {
```

```
            string playerAnswer = i < answers.Count ? answers[i] : "(no answer)";

            report.AppendLine($"Question {i + 1}: {questions[i].question}");
            report.AppendLine($"Correct answer: {questions[i].correctAnswer}");
            report.AppendLine($"Player answer: {playerAnswer}");
            report.AppendLine();
        }

        list.Add(new ChatMessage
        {
            Role = "user",
            Content = report.ToString()
        });

        return list;
    }
}


}
```