МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

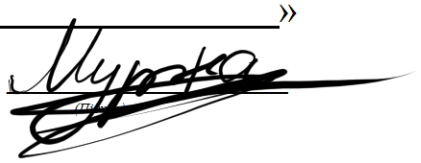
ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему: «Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC»

здобувачки Муржі Надії Анатоліївни
(ПІБ)



Науковий керівник:
д.т.н., доцент
_____Шевченко К.Л.

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент
_____Тішков Б.О.

Київ 2025

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:
Керівник проєктної групи(гарант)
освітньо-професійної програми
_____Помазун О.М.
"____" _____2025 р.

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____Тішков Б.О.
"____" _____2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувачки вищої освіти ***Муржі Надії Анатоліївни***

очної (денної) форми навчання

на підготовку кваліфікаційної бакалаврської роботи
на тему: «*Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC*»

Тему затверджено наказом ректора Університету від « *7* » *березня 2025* р. № *466-ст*.

Кваліфікаційна бакалаврська робота виконується на матеріалах
офіційної документації Unity, наукових публікацій про поведінку агентів у віртуальних середовищах, матеріалах з адаптивного геймдизайну

План кваліфікаційної бакалаврської роботи
Розділ І Характеристика та аналіз предметної галузі

Розділ ІІ Розробка вимог та моделювання інформаційної системи

Розділ ІІІ Проєктування та реалізація компонентів гри

Об'єкт дослідження: процеси поведінки неігрових персонажів (NPC) в умовах віртуального ігрового середовища

Предмет дослідження: механізми адаптивної реакції NPC на дії гравця, зміни контексту та власні емоційні стани у процесі діалогу та бойової взаємодії

Мета кваліфікаційної бакалаврської роботи: розробка інтерактивного прототипу гри з адаптивною поведінкою NPC, які здатні змінювати свої реакції залежно від внутрішніх параметрів (довіри, настрою, тривожності) та зовнішнього ігрового контексту._____

Конкретні завдання, які здобувачка повинна виконати для досягнення поставленої мети:

У розділі I Визначити основні характеристики предметної галузі та об'єкта дослідження. Провести аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

У розділі II Провести аналіз вимог до інформаційної системи, сформулювати основні задачі. Розробити алгоритм розв'язнання задачі. Провести моделювання інформаційної системи та її поведінки.

У розділі III Розробити інформаційне забезпечення ігрової платформи. Розробити структуру технічного забезпечення. Розробити програмне забезпечення та документацію до нього. Навести приклади практичної реалізації розробленої гри.

Завдання підготував
науковий керівник _____ **Шевченко Костянтин Леонідович**

«9 » червня 2025 р.

Завдання одержав
здобувач _____ **Муржа Надія Анатоліївна**

«9 » червня 2025 р..

Відгук
про кваліфікаційну бакалаврську роботу
здобувачки навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми
«Комп'ютерні науки»

Муржі Надії Анатоліївни

на тему
« Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC »

1. Актуальність теми: враховуючі стрімкий розвиток ігрової індустрії та зростання попиту ігрові продукти, де ігрові персонажі адаптивно реагують на дії гравця, створючі ефект живого середовища, тема роботи є актуальною.
2. Позитивні риси кваліфікаційної бакалаврської роботи: в роботі проведено грунтовний аналіз літературних джерел, сформульовано вимоги та основні задачі дослідження, розроблено інформаційне та програмне забезпечення, визначено структуру технічного забезпечення
3. Наявність самостійних розробок автора: автором самостійно виконані всі поставлені в індивідуальному завданні задачі.
4. Цінність теоретичних висновків та практичних рекомендацій: на основі моделі адаптивної поведінки NPC створено працездатний ігровий прототип, який може бути використаний як основа для подальших розробок освітніх або ігрових проєктів
5. Наявність недоліків: в роботі зустрічаються незначні відхилення у форматуванні тексту, суттєвих змістовних зауважень немає.
6. Загальна оцінка кваліфікаційної бакалаврської роботи та її допущення до захисту перед ЕК: кваліфікаційна бакалаврська робота на тему «Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC» відповідає встановленим вимогам та рекомендується до захисту, а її автор, Муржа Надія Анатоліївна заслуговує на присвоєння освітньо-кваліфікаційного рівня «бакалавр» за спеціальністю 122 «Комп'ютерні науки».

Науковий керівник
д.т.н., професор кафедри
інформаційних систем в економіці Шевченко К.Л.
(підпис)

" 15 " червня 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НДІ АВТОМАТИЗАЦІЇ
ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

А Е Д

03056, Київ, проспект Берестейський 37,
КПІ-4222
Тел: (044) 204-98-93
E-mail: tuz@aer.ntu-kpi.kiev.ua

ЄДРПОУ 02070921, розрахунковий рахунок 31254226113853
КПКВ 25010100 в ДКСУ, код банку 820172, БН. 545

№

Рецензія

на кваліфікаційну бакалаврську роботу
здобувачки вищої освіти
Муржі Надії Анатоліївни
На тему «Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC»

Актуальність теми кваліфікаційної бакалаврської роботи і доцільність її розроблення: ринок комп'ютерних ігрових продуктів останніми роками є одним з таких, що має найбільшу рентабельність Особливим попитом користуються продукти, у яких ігрові персонажі адаптивно реагують на дії гравця. Тому тему представленої роботи можна вважати актуальною.

Якість проведеного дослідження: робота містить грунтовний аналіз літературних джерел, аргументований вибір архітектурного та програмного забезпечення, все це виконано з достатньою якістю.

Позитивні риси кваліфікаційної бакалаврської роботи: всю роботу, від постановки задачі до практичної реалізації побудовано логічно. Кінцевим результатом є створення робочого прототипу.

Зауваження: до зауважень слід віднести некоректне розташування назв таблиць. Як правило, в технічній документації вони вирівнюються по правому краю сторінки.

Практична значимість висновків і рекомендацій: автором створено працездатний ігровий прототип, який може бути використаний для подальших розробок освітніх або ігрових застосунків.

Вважаю, що кваліфікаційна бакалаврська робота на тему «Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC» відповідає встановленим вимогам та заслуговує оцінки "відмінно", а її автор, Муржа Надія Анатоліївна, заслуговує на присвоєння освітньо-кваліфікаційного рівня «бакалавр» за спеціальністю 122 «Комп'ютерні науки».

Місце роботи та посада рецензента

Директор НДІ автоматизації експериментальних досліджень, к.т.н., доцент Самарцев Ю.М

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Науково-дослідний інститут автоматизації експериментальних досліджень

2024/6/18 09:09

АНОТАЦІЯ

кваліфікаційної бакалаврської роботи
здобувачки першого (бакалаврського) рівня вищої освіти, 4 курсу,
виконаної на тему: «__Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC_____»
Київ: кафедра інформаційних систем в економіці, 2025 р.

У роботі розглянуто проєктування адаптивної поведінки неігрових персонажів (NPC) у комп'ютерній грі. Розроблено ігровий прототип, в якому NPC реагують на дії гравця залежно від внутрішніх параметрів: довіри, настрою та тривожності. Реалізовано систему діалогів, бойову логіку та збереження даних у форматі JSON. Поведінка NPC моделюється через дискретну функцію реакції, що враховує контекст середовища. Архітектура гри побудована з використанням C# та середовища Unity. Результати дослідження можуть бути використані для розробки інтелектуальних агентів у навчальних, демонстраційних або ігрових проєктах.

РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить 71 сторінку, 10 таблиць, 17 рисунків, список використаних джерел з 30 найменувань, 1 додаткок на 72-74 сторінках.

«Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC»

Об'єкт дослідження: Поведінка неігрових персонажів (NPC) в умовах віртуального ігрового середовища.

Предмет дослідження: Механізми адаптивної реакції NPC на дії гравця, зміни контексту та власні емоційні стани у процесі діалогу та бойової взаємодії.

Метою кваліфікаційної роботи є розробка інтерактивного прототипу гри з адаптивною поведінкою NPC, які здатні змінювати свої реакції залежно від внутрішніх параметрів (довіри, настрою, тривожності) та зовнішнього ігрового контексту. Проєкт має продемонструвати можливість емоційно-чутливої взаємодії між гравцем і віртуальними агентами у реальному часі.

Відповідно до поставленої мети визначені такі *завдання*:

1. Проаналізувати сучасні підходи до реалізації адаптивної поведінки NPC у відеоіграх.
2. Розробити модель NPC, яка враховує параметри довіри, настрою та тривожності.
3. Реалізувати діалогову систему з умовною логікою реакцій NPC.
4. Побудувати бойову підсистему з урахуванням зміни поведінки залежно від внутрішніх станів персонажів.
5. Спроєктувати модульну архітектуру, що дозволяє розширення та повторне використання компонентів.
6. Реалізувати механізми збереження ігрового прогресу з використанням формату JSON.
7. Побудувати UML-діаграми для структурного й поведінкового моделювання системи.
8. Провести апробацію роботи через функціональне тестування контрольних сценаріїв.

Теоретична значущість полягає в обґрунтуванні моделі адаптивної поведінки NPC на основі параметрів емоційного стану й подієвої логіки. *Методична значущість* полягає в описі архітектурного підходу до інтеграції діалогових, бойових і поведінкових систем у межах єдиної ігрової логіки. *Практична цінність* роботи полягає у створенні функціонального прототипу, який може бути використаний для демонстрацій, освітніх цілей, або слугувати основою для подальших досліджень у галузі інтелектуальних віртуальних агентів.

Рік виконання кваліфікаційної бакалаврської роботи – 2025.

Рік захисту роботи – 2025.

Ключові слова: адаптивна поведінка NPC, емоційна модель, діалогова система, Unity, бойова логіка, довіра, тривожність.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІЗ – інформаційне забезпечення

ІС – інформаційна система

NPC (Non-Player Character) – не ігровий персонаж

AAA (Triple-A) - використовується для класифікації відеоігор, вироблених або розповсюджених середнім або великим видавцем.

АРМ – автоматизоване робоче місце

ПЗ – програмне забезпечення

ВСТУП

В умовах стрімкого розвитку ігрової індустрії та відповідного зростання кількості продуктів на ринку, зростає попит на віртуальні світи, у яких персонажі реагують не за заздалегідь визначеним шаблоном, а відповідно до змін ситуації, поведінки гравця чи власного внутрішнього стану. Розробка адаптивної поведінки NPC є важливою складовою такого підходу, адже саме це створює ефект живого середовища. Враховуючи потенціал подібних систем як у розважальних, так і в навчальних або дослідницьких проєктах, їх розробка набуває особливої актуальності – зокрема й в Україні, де останніми роками спостерігається активне зростання незалежної геймдев-спільноти.

Проблематика моделювання поведінки агентів досліджується у працях як вітчизняних, так і зарубіжних авторів. Зокрема, у дослідженнях Schwab та Bohle аналізується емоційно-чутлива взаємодія в інтерактивному наративі. Unity Technologies у своїй документації пропонує інструменти реалізації сценарних систем на базі ScriptableObject та State Machine. Також існують проєкти, у яких застосовуються моделі настрою, тривожності та довіри для прийняття рішень NPC (Short, 2020; Dignum, 2018).

Метою дослідження є розробка інтерактивної гри-прототипу з адаптивною поведінкою неігрових персонажів (NPC), які здатні реагувати на дії гравця під час діалогів або бойових ситуацій. Така система має демонструвати елементи емоційної чутливості та прийняття рішень на основі внутрішніх станів персонажа.

Завданнями дослідження є:

1. аналіз підходів до побудови поведінки NPC у сучасних ігрових системах;
2. розробка моделі NPC з параметрами довіри, настрою та тривожності;
3. реалізація діалогової системи з можливістю зміни реплік залежно від внутрішнього стану NPC;
4. побудова бойової системи з адаптивною логікою переходу NPC до агресії;
5. проєктування модульної архітектури для забезпечення масштабованості та гнучкості;

6. організація серіалізації та збереження даних у форматі JSON;
7. моделювання структури системи з використанням діаграм UML;
8. проведення апробації функціональності на контрольному прикладі взаємодії гравця з NPC.

Об'єктом дослідження є поведінка NPC в умовах ігрового середовища.

Предметом дослідження є механізм адаптивної реакції NPC на дії гравця та внутрішні параметри персонажа під час діалогу та бою.

У ході дослідження застосовано структурно-логічний та вербально-описовий методи, елементне моделювання, побудову UML-діаграм. Практична частина реалізована засобами C# у середовищі Unity з використанням ScriptableObject, інтерфейсів та станів анімації. Також проаналізовано наявні підходи до проєктування адаптивної поведінки NPC у відкритих джерелах.

Теоретична значущість полягає в обґрунтуванні моделі адаптивної поведінки NPC на основі змінних емоційного стану та логіки прийняття рішень.

Практична значущість полягає у створенні працездатного ігрового прототипу, який може бути використаний як основа для подальших досліджень, освітніх або демонстраційних проєктів.

Інформаційну базу дослідження становлять офіційна документація Unity, наукові публікації про поведінку агентів у віртуальних середовищах, матеріали з адаптивного геймдизайну, а також приклади реалізацій діалогових систем і бойової логіки з відкритим кодом, доступні на GitHub і Unity Asset Store.

Результати роботи апробовані у рамках демонстрації проєкту на кафедрі під час консультацій із науковим керівником. В межах апробації протестовано функціонал взаємодії з NPC у різних сценаріях. Публікації за темою кваліфікаційної роботи не здійснювались.

Структура роботи. Загальний обсяг роботи становить 72 сторінку друкованого тексту, 17 рисунків на 13 - 58 сторінках, 10 таблиць на 11 – 73 сторінках, 1 додаток на 72 - 74 сторінках. Список використаних джерел налічує 30 найменувань.

РОЗДІЛ 1
ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об’єкта дослідження

Для такої динамічної індустрії розваг, як комп’ютерні та мобільні ігри, характерний постійний пошук нових способів покращити користувацький досвід та поглибити інтеграцію гравця в ігровий світ. Як великі ААА-компанії, так і невеликі інді-розробники прагнуть привабити нових користувачів не лише якісним геймплеєм, а й глибокою, емоційно залученою взаємодією з персонажами, зокрема через динамічні діалоги, що імітують інтелектуальну реакцію неігрових персонажів.

Якщо в перших ігрових проєктах NPC виконували переважно функціональну або декоративну роль (наприклад, простих продавців або «вказівників» для гравця) то сьогодні вони виконують значно глибші наративні функції. Неігрові персонажі дедалі частіше виступають як джерело сюжетних розгалужень, як моральні «дзеркала», що реагують на вибір гравця, і як елементи симуляції живого суспільства всередині віртуального середовища. Їхні реакції здатні формувати емоційний фон гри, викликати емпатію, впливати на мотивацію користувача до дослідження світу та бажання перепройти гру після її завершення.

Важливу роль відіграє також здатність NPC зберігати контекст, або пам’ятати попередні дії гравця. У найуспішніших іграх сучасності саме це створює ілюзію «живого» світу, коли гравець відчуває, що його дії мають довготривалі наслідки. Таким чином, NPC перестають бути лише технічними елементами інтерфейсу й перетворюються на повноцінних учасників наративного процесу.

Розвиток сучасних технологій, зокрема ігрових рушіїв, також вплинув на еволюцію NPC. Завдяки інструментам на кшталт Unity, Unreal Engine, Godot, навіть інді-розробники отримали можливість створювати складні логіки поведінки персонажів, використовувати системи станів, діалогові дерева, анімаційні графи,

інтегрувати елементи штучного інтелекту та скриптової логіки. Проте ці можливості часто залишаються невикористаними або реалізуються за допомогою жорстко фіксованих сценаріїв, що унеможливлює масштабування та адаптацію.

Водночас із розвитком ігор та зростанням кількості випущених проєктів зростають і очікування гравців щодо адаптивності NPC по відношенню до дій гравця, що враховували б індивідуальність користувача. Від того, наскільки переконливо зреагує персонаж, враховуючи його внутрішній стан, погодні умови, час доби, ситуацію, в якій він знаходиться, залежить, наскільки гравець буде насолоджуватись процесом гри і скільки разів після закінчення він буде повертатись, щоб випробувати нові комбінації для впливу на світ.

У сучасному геймдизайні дедалі більшого поширення набуває концепція реактивного світу, коли ігрове середовище змінюється під впливом дій гравця, і NPC є ключовим інструментом цієї реактивності. Здатність персонажів адаптуватися до змін у зовнішньому чи внутрішньому світі гри не лише підвищує ігрову реалістичність, але й розширює дизайнерські можливості щодо гілкування сюжету, побудови моралі в грі, формування індивідуального досвіду користувача [2].

Проте з цим пов’язані й нові виклики. Для того щоб система NPC була справді адаптивною, необхідна відповідна архітектура, яка дозволить легко оновлювати діалогові гілки, враховувати параметри стану персонажа, забезпечувати зв’язок між діалогом, анімацією та подієвою логікою. Універсального рішення для цієї задачі поки не існує: великі студії створюють власні фреймворки, а малі команди часто імпровізують, створюючи кастомні інструменти. Це створює простір для дослідницьких прототипів [15].

У зв’язку з цим фокусом даного дослідження є проєктування інтерактивної комп’ютерної гри з акцентом на систему адаптивної поведінки NPC, яка реагує на змінні параметри внутрішнього стану персонажа (довіра, настрій, тривожність) та на контекст ігрового середовища, зокрема час доби або дії гравця.

Об’єктом дослідження є поведінка неігрових персонажів у динамічному віртуальному середовищі, а предметом є проєктування та реалізація архітектури

адаптивної системи, що дозволяє моделювати реакції NPC залежно від заздалегідь визначених параметрів і поточної взаємодії з гравцем. Такий підхід сприяє створенню живішого, гнучкішого світу, що здатний змінюватись у відповідь на вибір користувача навіть у рамках невеликого прототипу.

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Попри бажання розробників додавати в гру багаторівневі діалоги, які б змінювались залежно від різних факторів і мали вплив як на локальний сюжет, так і на загальну наративну структуру гри, більшість таких рішень залишаються досить затратними у реалізації. Доступні інструменти для створення діалогових систем або занадто обмежені у своїх можливостях, або вимагають великого обсягу ручної роботи при масштабуванні [8]. При цьому більш дешеві або умовно безкоштовні рішення далеко не завжди дозволяють досягти бажаної адаптивності без додаткового програмування, що ускладнює розробку, особливо в малих командах чи навчальних проєктах.

Серед дослідницьких підходів до побудови поведінки NPC особливу увагу привертає концепція «правдоподібних агентів» (believable agents), запропонована J. Bates (1994) [2], яка пізніше отримала розвиток у роботах Mateas і Stern (2003) [11] у рамках проєкту *Façade*. У ньому було реалізовано інтерактивну драму, де персонажі змінювали свою поведінку залежно від тону голосу гравця, змісту реплік та контексту ситуації. Хоча технічна реалізація була обмеженою, цей експеримент став еталонним прикладом емоційно-адаптивного діалогу у цифрових медіа.

У сфері геймдизайну дослідження адаптивних діалогів також пов’язані з напрямом інтерактивного сторітелінгу, що розглядається в роботах P. Cavazza, F. Charles і S. Mead [3]. Вони описують використання планувальних агентів для формування діалогових сценаріїв, які динамічно змінюються в залежності від взаємодії між гравцем і персонажами. Ці підходи орієнтовані на автономність NPC і можливість їхньої реакції в умовах змінної гри без жорсткого скриптування.

Практичне втілення адаптивних NPC широко представлено в іграх студій BioWare, Bethesda та Rockstar. Наприклад, серія ігор *Mass Effect* запровадила систему моралі (Paragon/Renegade), де вибір гравця змінює діалоги, стосунки з персонажами й навіть фінал гри. У *The Elder Scrolls V: Skyrim* NPC демонструють реакцію на дії гравця, його репутацію та стан (злочинець, герой, незнайомець). *Red Dead Redemption 2* реалізує складну поведінку NPC, яка залежить від емоцій, часу доби, погоди, зовнішнього вигляду гравця та його дій у світі [26].

Одним із сучасних підходів до моделювання поведінки в інтерактивних наративах є концепція емоційно-чутливої взаємодії, що аналізується у роботах Schwab і Bohle (2021) [17]. У своїх дослідженнях вони розглядають ігрових агентів як суб'єкти, що мають внутрішній емоційний стан, здатний змінюватись залежно від дій гравця, що впливає на подальшу траєкторію діалогу. Такі моделі підтримують багатоваріантну реакцію NPC та слугують основою для створення більш гнучких, «живих» систем. Їх підхід передбачає врахування афективних параметрів (напр., емоційної напруги або співпереживання), що робить поведінку персонажів менш передбачуваною та більш природною.

З технічного боку, Unity Technologies у своїй офіційній документації рекомендує використовувати архітектурні підходи на основі ScriptableObject, а також State Machine для побудови сценарних систем [22]. ScriptableObject дозволяє відокремити дані від логіки, що сприяє повторному використанню й простішому тестуванню. У рамках цієї роботи саме така архітектура була обрана для побудови діалогової системи, що дало змогу реалізувати адаптивну логіку реакцій NPC без значного ускладнення структури коду [5].

Додатково варто згадати роботи Short (2020) [19] та Dignum (2018) [7], які стосуються прийняття рішень агентами на основі соціальних і психологічних змінних. Зокрема, Dignum досліджує, як параметри довіри, страху або тривожності впливають на вибір дії NPC у динамічному середовищі. У цих підходах підкреслюється важливість контекстуальної поведінки, яка базується на змінних внутрішнього стану та історії взаємодій.

Водночас, навіть у комерційних проєктах адаптивність часто залишається умовною: більшість реакцій все ще базуються на заздалегідь прописаних сценаріях із чіткими гілками. Динамічна зміна діалогу на основі множинних параметрів (внутрішнього стану персонажа, тривалості знайомства, контексту сцени тощо) реалізується вкрай рідко через складність тестування та підтримки таких систем [9].

Більшість доступних інструментів, як комерційних, так і з відкритим кодом, пропонують базову підтримку діалогових дерев, але слабо охоплюють адаптивність на рівні внутрішніх емоцій NPC. Рішення, які дозволяють редагувати поведінку залежно від настрою, довіри, страху або інших змінних, часто створюються з нуля або потребують складної інтеграції сторонніх систем [18].

Таким чином, у рамках цієї роботи реалізовано спрощену, але ефективну модель адаптивної поведінки NPC, яка базується на змінних довіри, настрою та тривожності персонажа. У поєднанні з діалоговою системою на основі ScriptableObject і контекстом часу доби (день/ніч), це дозволяє досягти рівня адаптивності, який є досяжним у межах студентського проєкту, але при цьому демонструє ключові властивості реактивної поведінки: варіативність, емоційну чутливість і змінність залежно від взаємодії [1].

РОЗДІЛ 2
РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз і специфікація вимог до інформаційної системи

У межах цієї роботи передбачається створення прототипу інтерактивної комп'ютерної гри, основним функціональним призначенням якої є демонстрація адаптивної поведінки неігрових персонажів залежно від внутрішніх параметрів та контексту ігрового середовища [1]. З погляду інформаційного підходу, гра розглядається як інформаційна система, що поєднує кілька взаємопов'язаних підсистем: діалогову систему, підсистему бойової взаємодії, систему відстеження емоційного стану NPC, а також базову логіку гри [22].

З метою формування вимог до розроблюваної системи доцільно проаналізувати її концептуальну модель. Запланований програмний продукт є однокористувацьким дослідницьким прототипом у жанрі умовної рольової пригоди типу RPG-lite. Гравець керує персонажем, який взаємодіє з NPC у різних ігрових ситуаціях: під час діалогу, конфлікту, прохання або прояву агресії. Основна мета гри не передбачає проходження традиційного сюжету, а полягає у вивченні реакцій NPC на дії гравця з урахуванням таких факторів, як рівень довіри, настрій, тривожність, а також зовнішній контекст такий, як зміна дня і ночі [19].

Кожен NPC має набір базових характеристик та змінних, що модифікуються в процесі гри. Залежно від поточних значень цих параметрів NPC можуть:

- відповідати різними репліками в межах діалогу [15];
- відмовлятися від спілкування [3];
- ініціювати бій [10];
- просити про помилування в ситуаціях критично низького рівня здоров'я, якщо гравець атакує [20].

Також у системі планується реалізація умовних логічних сценаріїв, що відображають зміну поведінки залежно від попередніх взаємодій [13]. Наприклад,

у разі грубої репліки з боку гравця довіра NPC знижується; при критично низькому рівні довіри персонаж відмовляється надавати квест або навіть починає бій. NPC також може втікати від гравця в безпечне місце, якщо цей NPC не є бойовим типом [21].

Візуальна складова гри передбачається мінімалістичною, з базовим інтерфейсом та анімацією. Основний акцент ставиться на архітектурній побудові системи адаптивної поведінки NPC та взаємодії між її ключовими підсистемами: діалоговою, бойовою та логікою станів [22].

Для реалізації концепції було сформульовано функціональні (Таблиця 2.1) та нефункціональні (Таблиця 2.2) вимоги, що відображають очікувану поведінку системи та визначають її структуру, можливості розширення і підтримку [6].

Таблиця 2.1 – Основні функціональні вимоги до ІС

ID вимоги	Назва	Опис
FR-01	Наявність системи діалогів	Система забезпечувати інтерактивну взаємодію гравця з NPC через репліки з вибором.
FR-02	Підтримка реплік з вибором	Система повинна дозволяти гравцеві обирати одну з кількох реплік у межах діалогу з NPC.
FR-03	Залежність реплік від стану NPC	Зміст реплік NPC повинен змінюватися залежно від поточного емоційного стану персонажа.
FR-04	Завершення діалогу	Після завершення діалогу система повинна визначати, чи переходити до бою або завершити взаємодію, або відкриття додаткових меню (панель завдань чи панель мазазину).
FR-05	Наявність моделі емоційного стану	Система має відповідати за змінні довіри, настрою і тривожності NPC, що впливають на їхню реакцію в діалозі та бою.

Продовження таблиці 2.1

ID вимоги	Назва	Опис
FR-06	Параметри довіри, настрою, тривожності	У моделі NPC мають бути реалізовані змінні: довіра, настрій, тривожність.
FR-07	Вплив параметрів на діалог і бій	Параметри мають впливати на зміст реплік і поведінку в бою.
FR-08	Наявність контексту NPC	Система дозволяє змінювати поведінку NPC залежно від часу доби та попередніх дій гравця.
FR-09	Зміна поведінки залежно від часу доби	NPC повинен змінювати поведінку залежно від часу доби в ігровому світі (день або ніч).
FR-10	Зміна станів залежно від дій гравця	Система повинна реагувати на попередні дії гравця, змінюючи внутрішні параметри NPC.
FR-11	Наявність бойової системи	Має забезпечувати базовий набір атак та захисту аби була можливість імітувати бій.
FR-12	Ініціація бою з діалогу	Гравець повинен мати змогу ініціювати бій через вибір репліки в діалозі.
FR-13	Базова анімація атак	Система повинна містити базову анімацію атак, пов’язаних із бойовою взаємодією.
FR-14	Реакція NPC на критичний стан	NPC повинен мати здатність реагувати на критично низький рівень здоров’я.
FR-15	Передача станів між підсистемами	Стан NPC має передаватися між різними підсистемами: діалоговою, бойовою та контекстною.

Джерело: сформовано автором на основі виконаного дослідження

Для зручності аналізу і візуалізації логічної структури функціональних вимог було побудовано діаграму, на якій відображено їх ієрархію і залежності від інших вимог.

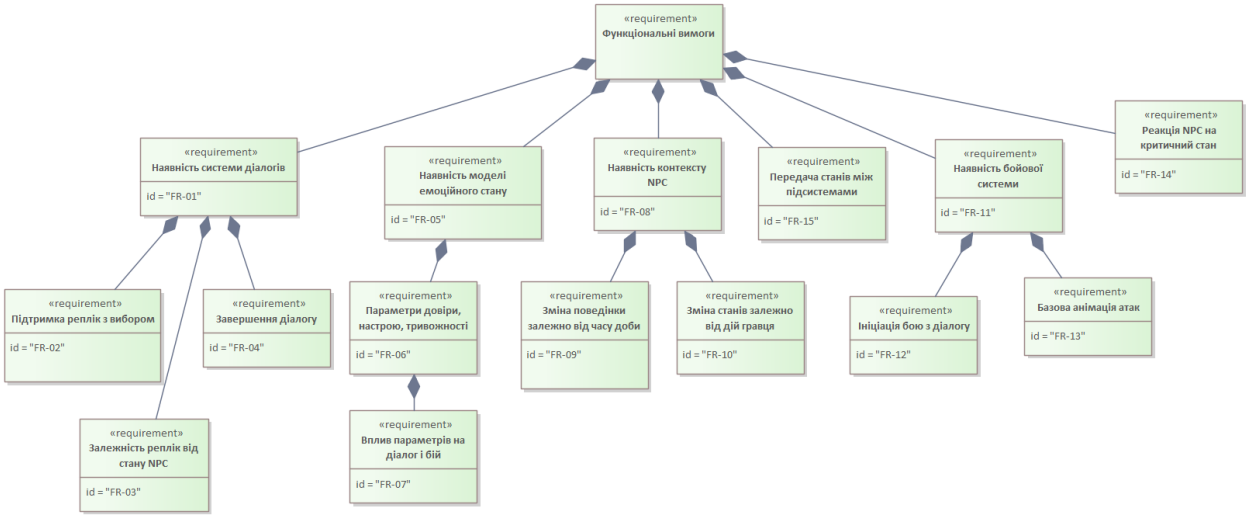


Рисунок 2.1 – Діаграма ієрархії функціональних вимог

Джерело: сформовано автором на основі виконаного дослідження

Таблиця 2.2 – Основні не функціональні вимоги до ІС

ID вимоги	Назва	Опис
FR-01	Модульність підсистем	Кожна підсистема має бути реалізована як окремий модуль для спрощення розробки й тестування.
FR-02	Можливість редагування діалогів без зміни коду	Система має підтримувати додавання нових NPC і сценаріїв без зміни коду.
FR-03	Продуктивність на середньому ПК	Гра має стабільно працювати в середовищі Unity або на звичайному ПК.
FR-04	Зрозумілий користувацький інтерфейс	Інтерфейс користувача повинен бути простим для сприйняття та зручним у використанні.

Джерело: сформовано автором на основі виконаного дослідження

Для не функціональних вимог також було сформовано діаграму ієрархії, яка зображена на *рисунку 2.2*.

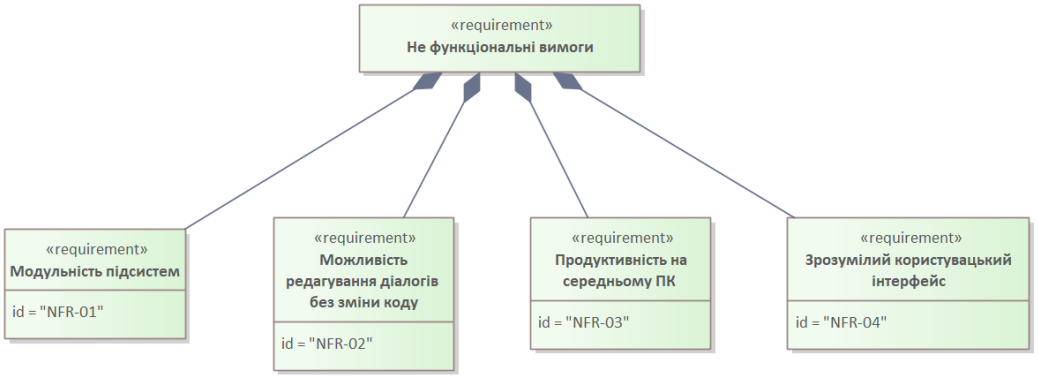


Рисунок 2.2 – Діаграма ієрархії не функціональних вимог

Джерело: сформовано автором на основі виконаного дослідження

Таким чином, сформульовані функціональні та нефункціональні вимоги окреслюють не лише загальні очікування до інформаційної системи, але й конкретизують логіку взаємодії окремих підсистем, їхній зв'язок та ролі в реалізації ігрового процесу. Вимоги узгоджуються з початковою концепцією гри, в якій адаптивна поведінка NPC є ключовим елементом. Ретельне розділення вимог за типами, а також побудова їхньої внутрішньої ієрархії, дозволяють не тільки краще структурувати подальшу розробку, але й зробити систему гнучкою до змін і розширення [24].

2.2 Постановка та алгоритм розв'язнання задачі

2.2.1 Постановка задачі

Задача полягає у створенні інтерактивної інформаційної системи у формі комп'ютерної гри, в якій неігрові персонажі адаптують свою поведінку відповідно до змінного контексту та власного внутрішнього стану [19]. Така поведінка включає переважно емоційно-наративну складову (варіативні діалоги), що

обирається на основі параметрів, таких як довіра, настрій, тривожність та час доби [2].

Призначенням задачі є підвищення реалістичності поведінки NPC для забезпечення глибшого занурення гравця у симульоване середовище. Подібна адаптивність надає грі нелінійності та варіативності, що є важливими трендами в сучасній індустрії цифрових розваг [3].

З огляду на обсяг інформації, що має оброблятися, та необхідність реагування в реальному часі, для реалізації цієї задачі застосовується автоматизована обробка даних за допомогою ЕОМ, зокрема на основі ігрового рушія Unity та мови програмування C# [22]. Система має працювати в інтерактивному режимі, забезпечуючи зв’язок між підсистемами: діалоговою, бойовою, системою емоційного стану та середовищем гри [5].

Об’єктами управління в межах даної задачі виступають неігрові персонажі, сценарії діалогів, бойові реакції та графічний інтерфейс. NPC мають змінні емоційні параметри, зокрема довіру, настрій і тривожність, які динамічно оновлюються в процесі гри залежно від дій гравця [7]. Сценарії діалогів активуються відповідно до контексту ситуації, а самі репліки змінюються залежно від внутрішнього стану NPC та попереднього досвіду взаємодії [2].

Усі ці елементи мають бути синхронізовані з інтерфейсом, який дозволяє гравцеві отримувати зворотний зв’язок та контролювати перебіг взаємодії у зручній формі [15].

Вихідна інформація, яку формує система, є ключовим інструментом адаптації ігрового процесу. Вона охоплює не лише вибрані репліки персонажів, а й бойові дії та ситуації, коли NPC відмовляється від взаємодії [14]. Ця інформація використовується для оновлення внутрішнього стану NPC, формування реакцій в інтерфейсі та візуалізацій (зокрема анімацій), а також зберігається у форматі JSON [6]. Такий підхід забезпечує можливість повторного використання даних у тій самій або новій ігровій сесії [23].

Розв’язання задачі має подієво-асинхронний характер: кожна дія гравця запускає окремий цикл обробки, що дозволяє досягти високої гнучкості та швидкої

реакції системи на зміну контексту [24]. У свою чергу, автоматизоване розв’язання задачі завершується після завершення діалогу або бойового етапу, а також при завершенні ігрової сесії або аварійному припиненні роботи програми [27].

Задача має тісні зв’язки з іншими функціональними модулями інформаційної системи. Зокрема, вона взаємодіє із підсистемою збереження та завантаження прогресу гри, обробкою інвентарю гравця та управлінням виконанням квестів [15].

Розподіл дій між користувачем та автоматизованою системою є чітко визначеним: гравець виконує зовнішні дії, такі як ініціація діалогу, вибір реплік або атака, а інформаційна система, у свою чергу, обробляє отримані події, визначає відповідну реакцію з боку NPC та оновлює внутрішній стан гри [11]. Результати цієї обробки одразу виводяться через інтерфейс, а також, за потреби, зберігаються для подальшого використання [12].

Для кращого розуміння логіки взаємодії між підсистемами адаптивної поведінки NPC, системою введення та оточенням гри, нижче наведено інформаційну модель задачі. Ця схема відображає основні джерела вхідних даних, процеси їх обробки та формування вихідної інформації, що передається іншим підсистемам або зберігається у відповідних файлах [25].

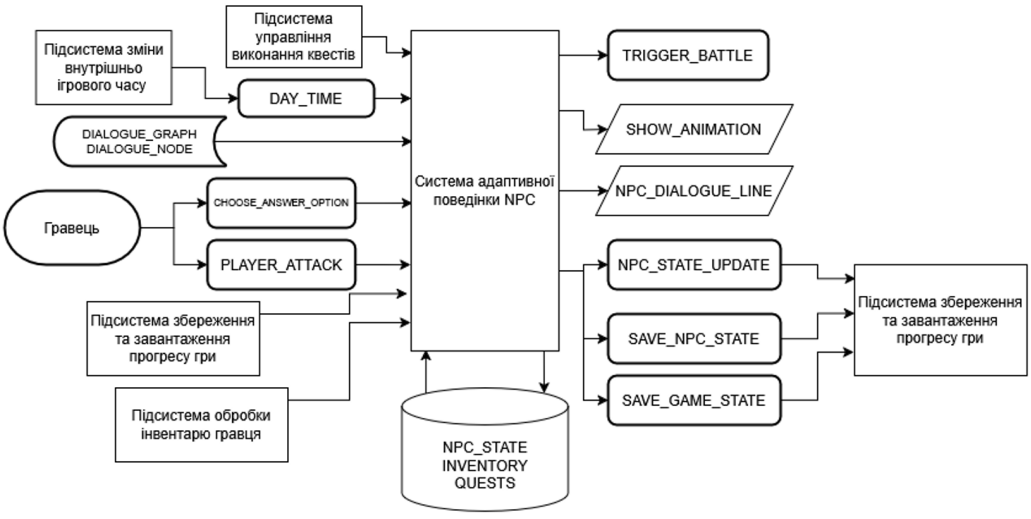


Рисунок 2.3 – Інформаційна модель задачі

Джерело: сформовано автором на основі виконаного дослідження

У процесі реалізації задачі з моделювання адаптивної поведінки NPC формується низка вихідних повідомлень, що відіграють критичну роль у взаємодії між ключовими підсистемами гри [19]. У межах цього проєкту до основних таких

підсистем належать: логіка діалогів, бойовий модуль, модуль збереження, а також інтерфейс користувача [22].

Під час розробки особлива увага приділялась чіткому розмежуванню вихідної інформації залежно від її призначення: реактивна інформація передається миттєво після взаємодії користувача (наприклад, обрання репліки чи атака), тоді як інша накопичується або готується до збереження (наприклад, оновлений стан NPC, який фіксується в JSON для подальшого відновлення у наступних сесіях гри) [6]. Це дозволило досягти збалансованого підходу до продуктивності системи та гнучкості її архітектури [18].

Найбільш типові вихідні повідомлення передаються у різних формах: прості логічні сигнали, рядки, складні структуровані дані. У багатьох випадках формування повідомлення супроводжується додатковою логікою перевірки стану персонажа, ігрового часу чи наявності інших умов, що відповідає вимогам до адаптивності [7].

Форма подання вихідної інформації у межах системи реалізована відповідно до специфіки обраного середовища розробки Unity з використанням мови програмування C#. Наприклад, реакції, пов’язані з діалогами, інтегровані через ScriptableObject-структури, а збереження параметрів відбувається шляхом серіалізації в JSON-файли з прив’язкою до унікального ідентифікатора кожного NPC [22]. Це забезпечує збереження контексту ігрового стану навіть у випадку багатьох одночасно активних персонажів [13].

У таблиці нижче наведено основні типи вихідної інформації, яка формується в процесі розв’язання задачі, із зазначенням її ідентифікаторів, форми подання, частоти формування, а також користувачів, які її споживають у межах інформаційної системи.

Таблиця 2.3 – Перелік і опис вихідних повідомлень

Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувачі інформації
Ініціація бойового режиму	TRIGGER_BATTLE	Логічний сигнал, що запускає бойовий сценарій.	Подійна	Миттєво після обробки; затримка до 0,2 с	Підсистема бою, анімаційна система
Вивід анімацій	SHOW_ANIMATION	Команда на запуск певної анімації.	Подійна	Протягом 0,1 – 0,3 с після події	Анімаційна система, візуальний інтерфейс
Репліка NPC у діалозі	NPC_DIALOGUE_LINE	Рядок тексту. Виводиться на екран гравця	Подійна	Одразу після переходу до репліки, до 0,1 с	Система UI, гравець
Оновлення стану NPC	NPC_STATE_UPDATE	Оновлення внутрішніх змінних (довіра, тривожність тощо)	Після кожної взаємодії, що змінює стан	Протягом 1 с після завершення взаємодії	Модуль логіки NPC, зберігаюча підсистема
Збереження стану NPC	SAVE_NPC_STATE	JSON-структура; містить емоційні параметри NPC	Після суттєвих змін стану або завершення діалогу/бою	До 5 с після тригеру збереження	Підсистема збереження, модуль повторного завантаження NPC
Збереження загального стану гри	SAVE_GAME_STATE	JSON-файл з інформацією про інвентар, позицію, квести	Після завершення ключової взаємодії або вручну	До 5 с; може зберігатись асинхронно	Підсистема збереження, модуль завантаження сесії

Джерело: сформовано автором на основі виконаного дослідження

Для забезпечення адаптивної поведінки NPC у межах розроблюваної інформаційної системи необхідною є обробка низки вхідних повідомлень, що надходять від гравця, ігрового оточення та структур діалогів [22]. Ці повідомлення виступають ініціаторами змін внутрішніх станів NPC, запуску нових сценаріїв взаємодії та бойових ситуацій [19].

Під час розробки було встановлено, що більшість вхідних повідомлень мають подієву природу та виникають у момент конкретної дії з боку користувача або зміни стану світу. Вони обробляються через Unity-сценарії на C#, з опорою на архітектурні шаблони взаємодії між підсистемами (наприклад, Event-driven або ScriptableObject-based підходи) [5].

Особливу роль серед вхідних повідомлень відіграють дані, отримані з ScriptableObject, які містять заздалегідь підготовлену інформацію про поведінку NPC (наприклад, TraderDialogueGraph або Guard_Graph) [15]. Ці структури дозволяють створити повторно використовуваний, гнучкий і зручний для редагування формат взаємодії [4].

Форма подання вхідних повідомлень варіюється: від простих логічних значень до складних об’єктів (наприклад, вузол діалогу з усіма доступними варіантами відповіді та умовами переходу) [15]. Частота надходження таких даних залежить від типу події. Наприклад, DAY_TIME оновлюється кожну секунду, вибір репліки оновлюється виключно за ініціативою гравця [10].

Таким чином, вхідна інформація є не просто джерелом запуску сценаріїв, а й ключовим елементом, який дозволяє реалізувати адаптивну логіку NPC без потреби в жорсткому програмуванні поведінки [17]. Завдяки модульній структурі система легко масштабується та допускає розширення в майбутньому [3].

У таблиці нижче (Таблиця 2.4) наведено структурований опис основних типів вхідної інформації, що використовується в межах проєкту.

Таблиця 2.4 – Перелік і опис вхідних повідомлень

Назва вхідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність надходження	Джерело
Поточний ігровий час (день/ніч)	DAY_TIME	Значення типу bool (день = true / ніч = false)	Періодичне переключення (1 раз на зміну доби)	Підсистема ігрового часу
Граф діалогу NPC	DIALOGUE_GRAPH	ScriptableObject-файл	Завантажується при старті діалогу з конкретним NPC	Файлова система Unity / Редактор Unity
Поточний вузол діалогу	DIALOGUE_NODE	Об’єкт вузла в межах діалогового графа	Визначається при кожному кроці діалогу	Система діалогів
Вибраний варіант відповіді гравця	CHOOSE_ANSWER_OPTION	Ціле число або ідентифікатор вибраного варіанта	Подієвий, генерується одразу після вибору варіанту гравцем	Інтерфейс користувача (UI)
Атака гравця	PLAYER_ATTACK	Сигнал з бойового контролера гравця	Подієвий	Контролер гравця

Джерело: сформовано автором на основі виконаного дослідження

2.2.2 Алгоритм розв’язання задачі

У рамках алгоритму адаптивної поведінки NPC інформаційна система використовує низку масивів, що відображають як поточний стан ігрового

середовища, так і дії користувача. Ці масиви формуються на основі внутрішньоігрових подій, заздалегідь створених структур, а також параметрів, які зберігаються у зовнішніх джерелах у форматі JSON або ScriptableObject у межах середовища Unity [5].

Одним із ключових джерел даних є граф діалогу (DIALOGUE_GRAPH), що є структурованим об’єктом, що описує логіку проходження діалогів між гравцем і NPC. Він містить вузли реплік, переходи між ними та умови активації залежно від стану персонажа [15]. Кожен граф пов’язаний із конкретним NPC або групою NPC, що забезпечує індивідуалізовану поведінку [19].

Вузли діалогу (DIALOGUE_NODE) - це окремі елементи графа, які містять репліки NPC або варіанти відповіді гравця. У момент вибору репліки система обробляє дані з вузлів, аналізуючи умови переходу та змінюючи стан NPC [15].

Масив CHOOSE_ANSWER_OPTION формується під час взаємодії користувача з інтерфейсом вибору. Вибір однієї з опцій передається в систему для подальшої обробки: збереження довіри, ініціація бою, вплив на гілки діалогу тощо [3].

PLAYER_ATTACK - це масив, що формується під час бою та включає дані про атаки гравця. Система NPC отримує цю інформацію, щоб ініціювати бойову поведінку, змінювати тактику, або ж у випадку критичного пошкодження - змінювати настрій і, за потреби, просити про пощаду [9].

DAY_TIME - масив, який передає інформацію про час доби в грі (день або ніч). Ці параметри впливають на поведінку NPC (наприклад, деякі з них взагалі не спілкуються у нічний час) [18].

NPC_EMOTION_STATE містить внутрішній стан NPC: довіра, настрій, тривожність. Ці параметри впливають на прийняття рішень та зміну реакцій NPC.

REACTION_RULES це масив правил, що визначають залежності між станом NPC, діями гравця та можливими реакціями. Правила зберігаються у вигляді наборів умов та тригерів [24].

Уся використана інформація формується на початку або під час виконання системи, її обробка відбувається асинхронно, залежно від дій гравця та внутрішніх подій [1].

Таблиця 2.5 - Перелік масивів використовуваної інформації

Масив	Ідентифікатор	Максимальна кількість записів
1	2	3
Граф діалогу	DIALOGUE_GRAPH	Обмежено кількістю NPC з унікальними графами діалогів
Вузол діалогу	DIALOGUE_NODE	Принаймні в 2 рази більше, ніж графів діалогів
Вибір відповіді гравця	CHOOSE_ANSWER_OPTION	Не більше 4 на один вузол діалогу
Подія атаки	PLAYER_ATTACK	Без обмежень
Поточний стан доби	DAY_TIME	2
Параметри емоцій NPC	NPC_EMOTION_STATE	300
Сценарії адаптивної реакції	REACTION_RULES	100

Джерело: сформовано автором на основі виконаного дослідження

Математичне забезпечення реалізованої задачі полягає у побудові моделі адаптивної поведінки NPC на основі сукупності змінних внутрішнього стану персонажа та зовнішнього контексту [19]. Основними об’єктами математичного моделювання є емоційні стани NPC (довіра, настрій, тривожність), дії гравця та умови середовища (час доби), які впливають на вибір реакції NPC у діалозі або в бойовій ситуації [17].

Умовні позначення:

- T - рівень довіри (Trust), числове значення в інтервалі [0, 1];
- M - рівень настрою (Mood), числове значення в інтервалі [-1, 1];
- A - рівень тривожності (Anxiety), числове значення в інтервалі [0, 1];

- $D_t \in$ {день,ніч} - поточний ігровий час;
- R - репліка, яку обирає NPC;
- P - дія гравця, що може бути реплікою, атакою або запитом.

1) Функція адаптивної реакції NPC описується наступним чином:

$$R = f(P,\ T,\ M,\ A,\ D_t) \tag{2.1}$$

Це функція прийняття рішення, яка визначає, яку репліку або дію має виконати NPC у відповідь на дію гравця залежно від його емоційного стану та контексту.

2) Зміна стану NPC відбувається за кожної взаємодії:

$$T' = T + \Delta T(P), \qquad M' = M + \Delta M(P), \qquad A' = A + \Delta A(P), \tag{2.2}$$

де $\Delta T(P)$, $\Delta M(P)$, $\Delta A(P)$ - змінення відповідних параметрів залежно від дії гравця PPP. Ці зміни зберігаються у таблицях сценаріїв реакції в рамках логіки ScriptableObject.

3) Початок бою ініціюється, якщо виконуються всі умови:

$$T < T \quad , \qquad A > A \quad , \qquad P = \qquad , \tag{2.3}$$

де $T_{кр}$ - критичний поріг довіри, а $A_{кр}$ - поріг тривожності. Значення цих порогів задаються вручну для кожного NPC у конфігураційному файлі.

У процесі тестування моделі було визначено, що зміни стану NPC мають очікуваний вплив на поведінку лише за умови правильного балансування коефіцієнтів ΔT, ΔM, ΔA. У спрощеній моделі коефіцієнти є сталими, однак система може бути розширена шляхом введення функцій залежності від контексту або історії взаємодії [19].

Розроблена модель є дискретною, подієвою, і добре узгоджується з архітектурою ігрових систем у середовищі Unity. Вона допускає масштабування як по NPC, так і по кількості параметрів стану без потреби зміни основної логіки [5].

Алгоритм розв’язання задачі адаптивної поведінки NPC реалізується у вигляді подієво-орієнтованого процесу, що забезпечує динамічну реакцію неігрових персонажів на дії гравця, контекст ігрового середовища та власний внутрішній стан [14]. Його реалізація здійснюється в середовищі Unity з

використанням мови програмування C# [1], а також низки допоміжних компонентів: ScriptableObject-файлів для структурування діалогів, JSON-файлів для збереження стану, Animator Controller для керування анімаціями тощо [11].

Розв’язання задачі передбачає обробку вхідної події (наприклад, ініціація діалогу з боку гравця), зчитування поточного емоційного стану NPC, перевірку зовнішніх умов (часу доби та історії взаємодій), а також формування відповідної реакції [17]. Усі дії виконуються в режимі реального часу з тісною інтеграцією між підсистемами: діалоговою, бойовою, станів NPC, інтерфейсом користувача та системою збереження [4].

Процес взаємодії починається з ініціації події гравцем. Відповідний обробник визначає NPC, з яким відбувається взаємодія, та завантажує ключові параметри: ім’я, тип, рівень довіри, настрій, тривожність тощо. Ці значення зберігаються в оперативній пам’яті та, за потреби, зчитуються з локального JSON-файлу [23].

Наступним кроком є аналіз контексту середовища: система враховує час доби, кількість попередніх взаємодій із цим NPC, розташування в просторі, а також наявність провокаційних або агресивних дій з боку гравця. Це дозволяє формувати більш гнучку та правдоподібну поведінку NPC [6].

У разі діалогу, обрана репліка гравця інтерпретується як DialogueNode, що викликає відповідну гілку у дереві діалогу (DialogueGraph). Якщо стан NPC не відповідає умовам гілки (наприклад, надто низька довіра), відбувається перехід до альтернативного (fallback) вузла. Аналогічно, за відповідних умов (агресія з боку гравця або системна перевірка параметрів), NPC може ініціювати бій або звернутись із проханням про помилування (у випадку критично низького рівня здоров’я) [20].

Усі наслідки вибору гравця впливають на параметри NPC, які змінюються відповідно до реакції. Зміни записуються у внутрішні структури й дублюються у зовнішні JSON-файли, що забезпечує можливість продовження гри з актуальними даними [5].

Після завершення відповідної дії система повертається в очікування наступної події. Така подієво-орієнтована архітектура дозволяє NPC реагувати природно, а гравцеві відчувати послідовність і глибину взаємодій [6]. На *рисунку 2.4* наведено узагальнену блок-схему, яка відображає логіку обробки подій адаптивної поведінки NPC у контексті діалогу, квестів і бойових ситуацій.

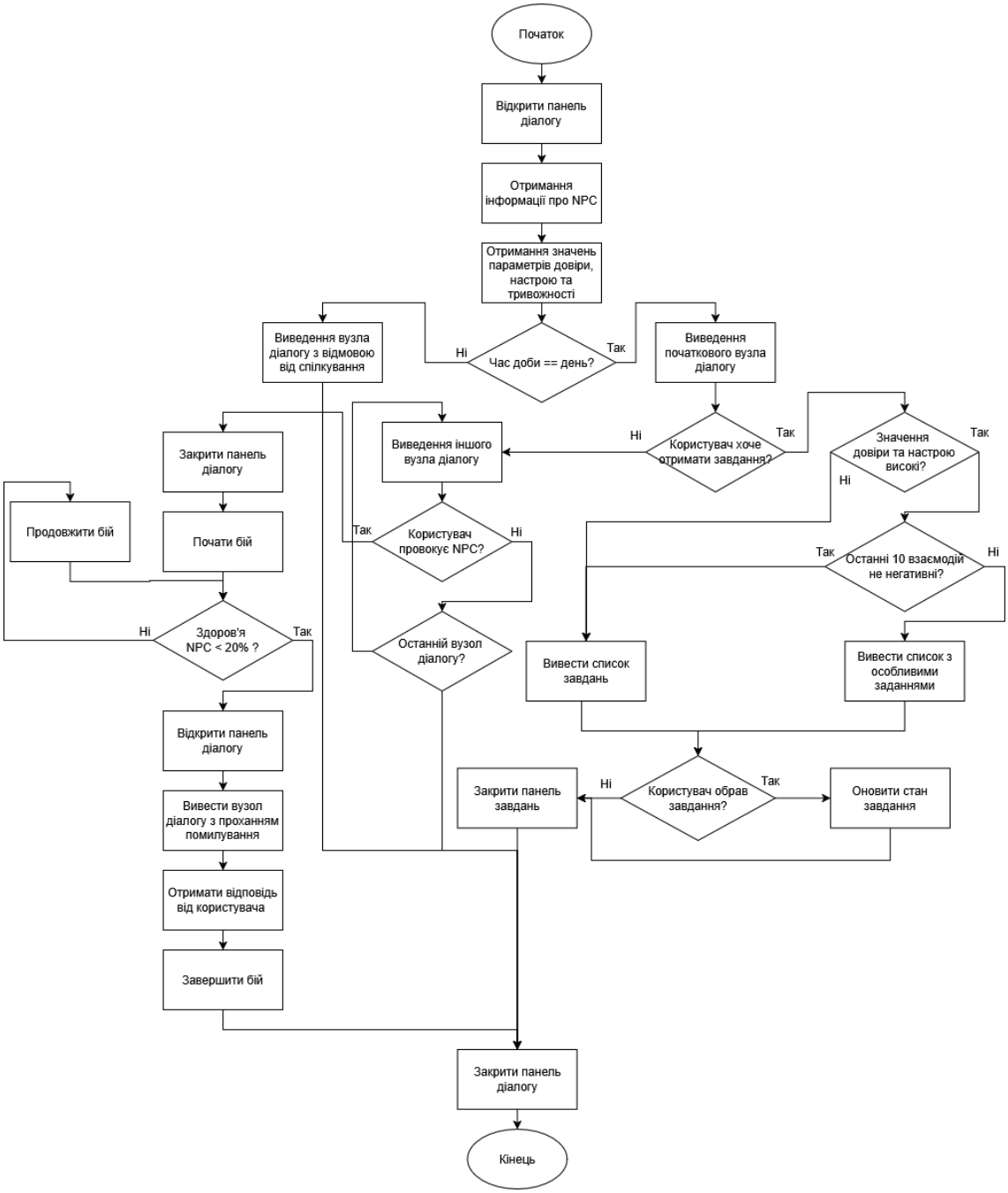


Рисунок 2.4 – Узагальнена блок-схема алгоритму

Джерело: сформовано автором на основі виконаного дослідження

2.3 Моделювання інформаційної системи

2.3.1 Моделювання поведінки системи

У процесі розробки інформаційної підсистеми, яка забезпечує адаптивну поведінку NPC в інтерактивному ігровому середовищі, критично важливим є моделювання поведінки системи [12]. Це дозволяє на концептуальному рівні описати основні сценарії взаємодії між гравцем і системою, встановити взаємозв’язки між функціональними компонентами, а також визначити межі відповідальності кожної підсистеми [19].

Моделювання поведінки є етапом, на якому формалізуються дії користувача, реакції NPC, логіка переходів між станами та способи обміну даними між модулями. Воно допомагає не лише структурувати логіку системи, але й виявити потенційні колізії чи пропущені сценарії ще до початку реалізації. У контексті системи, що проектується, це дозволяє забезпечити узгоджену роботу таких складових, як система діалогів, бойова система, модель емоційного стану NPC, інтерфейс користувача та модуль збереження.

Моделювання реалізується з використанням стандарту UML, який дає змогу описувати як поведінкову, так і структурну сторону системи. Зокрема, в межах цього підрозділу використовуються:

- Діаграма прецедентів для узагальненого подання основних сценаріїв взаємодії між гравцем і підсистемою NPC;
- Діаграма послідовності для конкретизації порядку обміну повідомленнями між об’єктами у межах одного прецеденту;
- Діаграма діяльності для опису логіки переходів між станами та варіантами дій NPC залежно від умов.

Моделі будуються на основі вже визначених функціональних вимог, постановки задачі та структури системи, а також враховують поточну реалізацію в ігровому середовищі Unity. Створення таких моделей є основою для реалізації адаптивної логіки, її тестування та подальшого масштабування.

Нижче представлено діаграму прецедентів, яка демонструє ключові сценарії використання підсистеми адаптивної поведінки NPC, взаємодію гравця з елементами системи та зв’язки між окремими прецедентами.

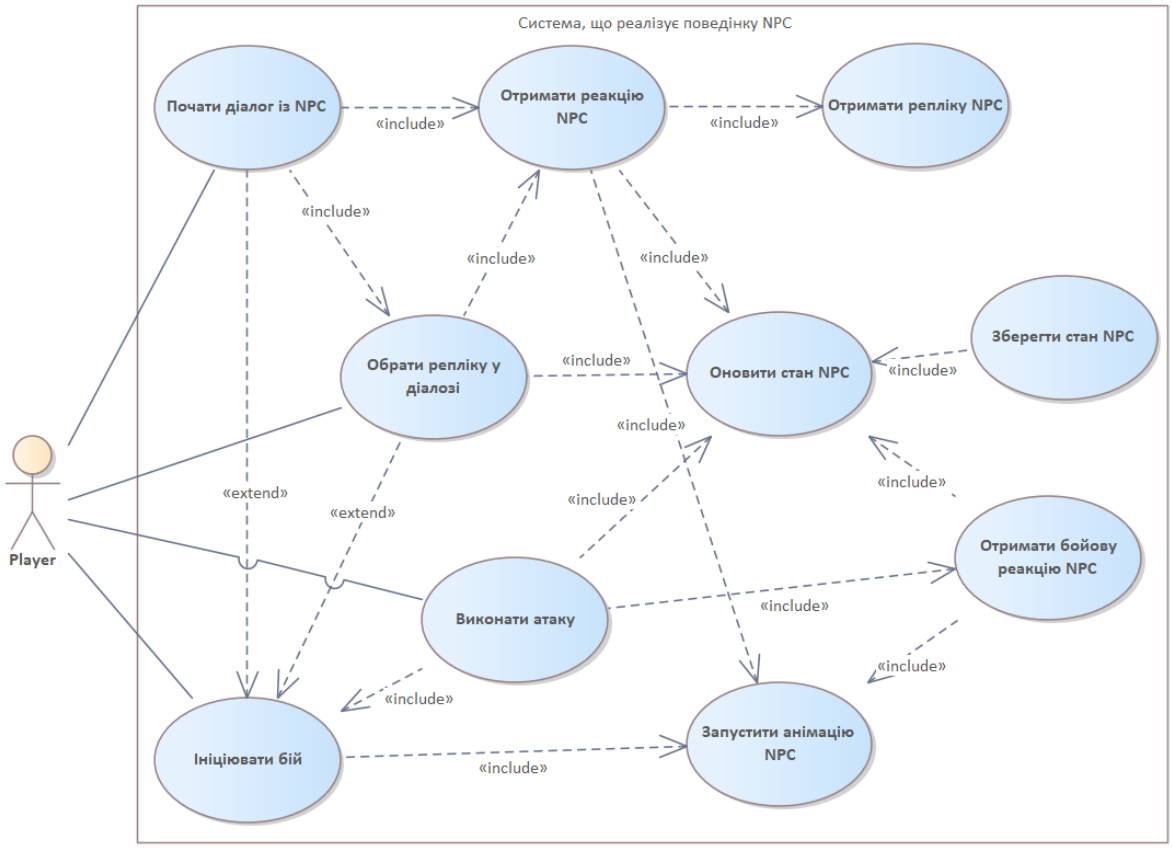


Рисунок 2.5 – Діаграма прецедентів

Джерело: сформовано автором на основі виконаного дослідження

У межах інформаційної підсистеми, що забезпечує адаптивну поведінку NPC, діаграми послідовності використовуються для моделювання детальної взаємодії між компонентами системи у відповідь на дії користувача. Вони демонструють порядок викликів методів, обмін повідомленнями та взаємозв’язки між об’єктами, задіяними у сценаріях.

Оскільки гравець є єдиним зовнішнім актором, усі представлені нижче діаграми послідовності описують сценарії, ініційовані саме ним.

На *рис. 2.6* показано діаграму, на якій представлено сценарій початку взаємодії між гравцем і неігровим персонажем (NPC), що є одним із ключових елементів адаптивної логіки гри. Діаграма демонструє процес активації діалогу,

завантаження стану NPC, визначення його типу, а також відображення першої репліки у діалоговому вікні.

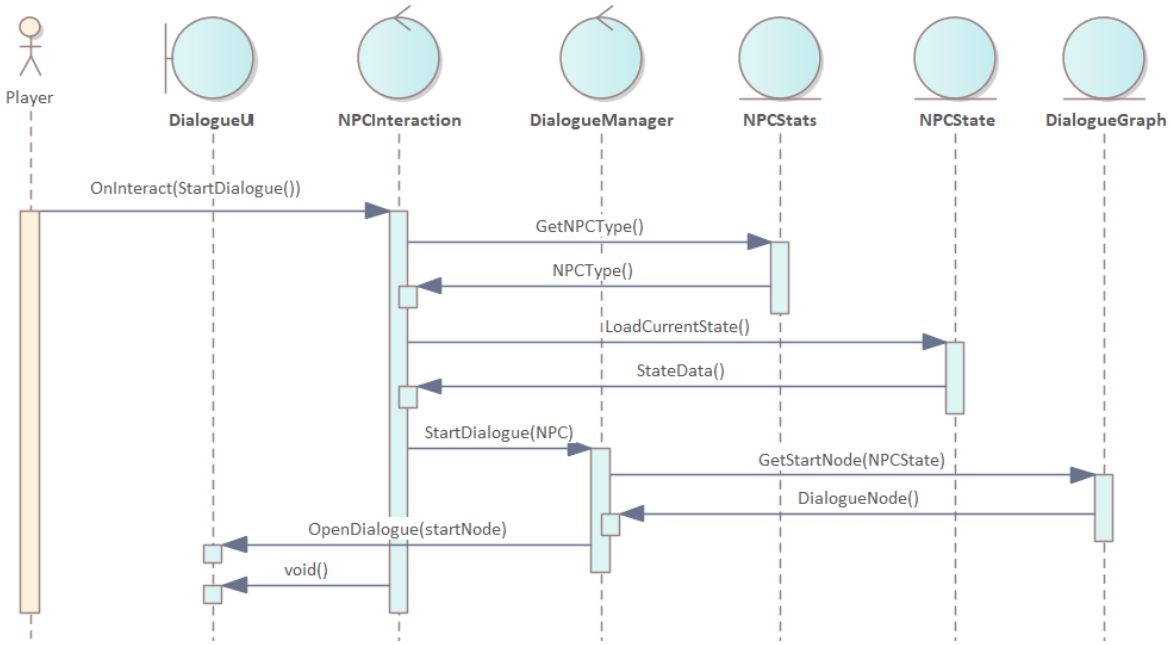


Рисунок 2.6 – Діаграма послідовності для сценарію «Почати діалог із NPC»

Джерело: сформовано автором на основі виконаного дослідження

Ця діаграма є центральною для розуміння процесу адаптації діалогу до стану NPC. Вона демонструє, як інформація про NPC використовується для динамічного вибору стартового вузла діалогу, що забезпечує відчуття реалістичної поведінки персонажів та дозволяє гравцеві отримати персоналізовану реакцію вже на перших етапах взаємодії.

Наступна діаграма на *рис.2.7* відображає процес вибору варіанта відповіді гравцем у рамках активного діалогу з NPC. Вона ілюструє, як дія гравця на інтерфейсі перетворюється на логічний запит до системи, яка аналізує структуру діалогу, знаходить наступний вузол і повертає результат для виводу або завершення взаємодії.

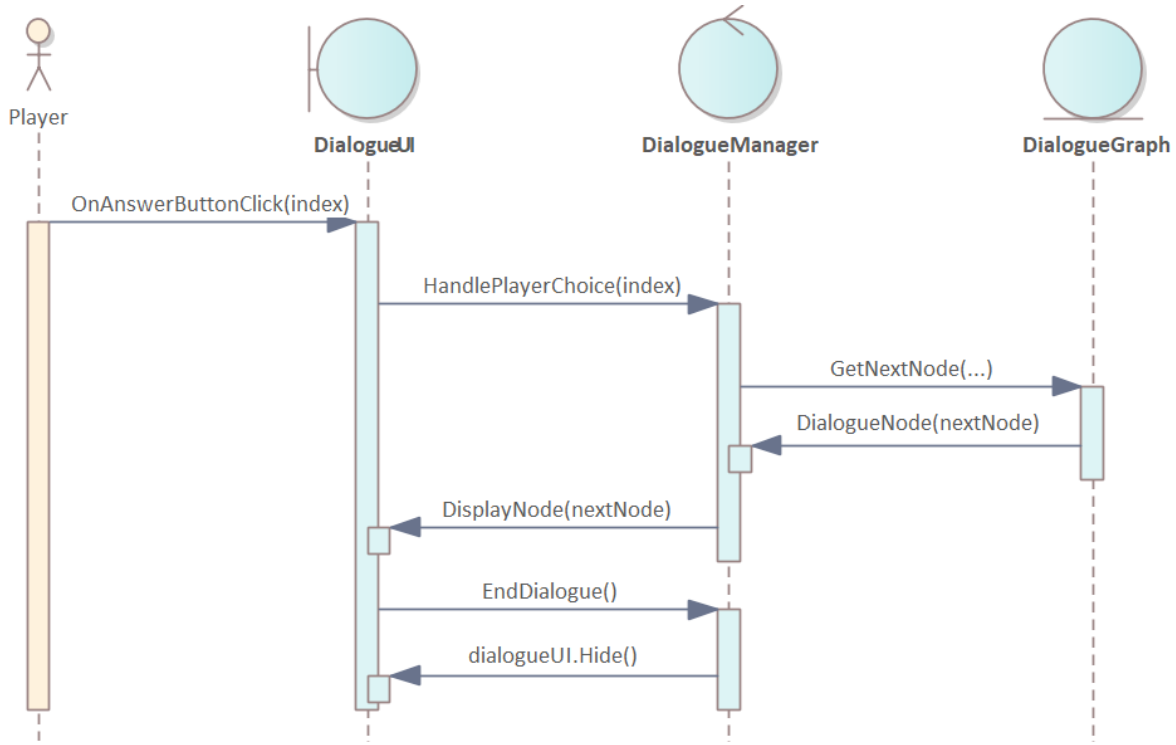


Рисунок 2.7 – Діаграма послідовності для сценарію «Обрати репліку у діалозі»

Джерело: сформовано автором на основі виконаного дослідження

Цей сценарій демонструє основний цикл взаємодії гравця з NPC у діалозі. Він є ключовим для реалізації гілок у системі діалогів і для динамічної побудови реакцій NPC на вибір користувача. Залежно від репліки гравця та внутрішнього стану NPC, структура дерева може мати різні переходи, що дозволяє реалізовувати адаптивну поведінку в межах одного діалогу.

Діаграма на *рис.2.8* моделює сценарій атаки гравцем NPC, під час якої неігровий персонаж може або загинути, або при досягненні критичного рівня здоров'я попросити пощади. Даний сценарій демонструє взаємодію бойової системи з логікою NPC, анімаційним контролером, UI, та діалоговим менеджером.

Ця діаграма моделює критичний сценарій адаптивної поведінки NPC, а саме момент, коли агресивна дія гравця провокує NPC на прохання пощади або спричиняє його загибель. Вона ілюструє гнучкість системи: при різних значеннях здоров'я NPC система або завершує бій, або переключається у новий адаптивний діалог.

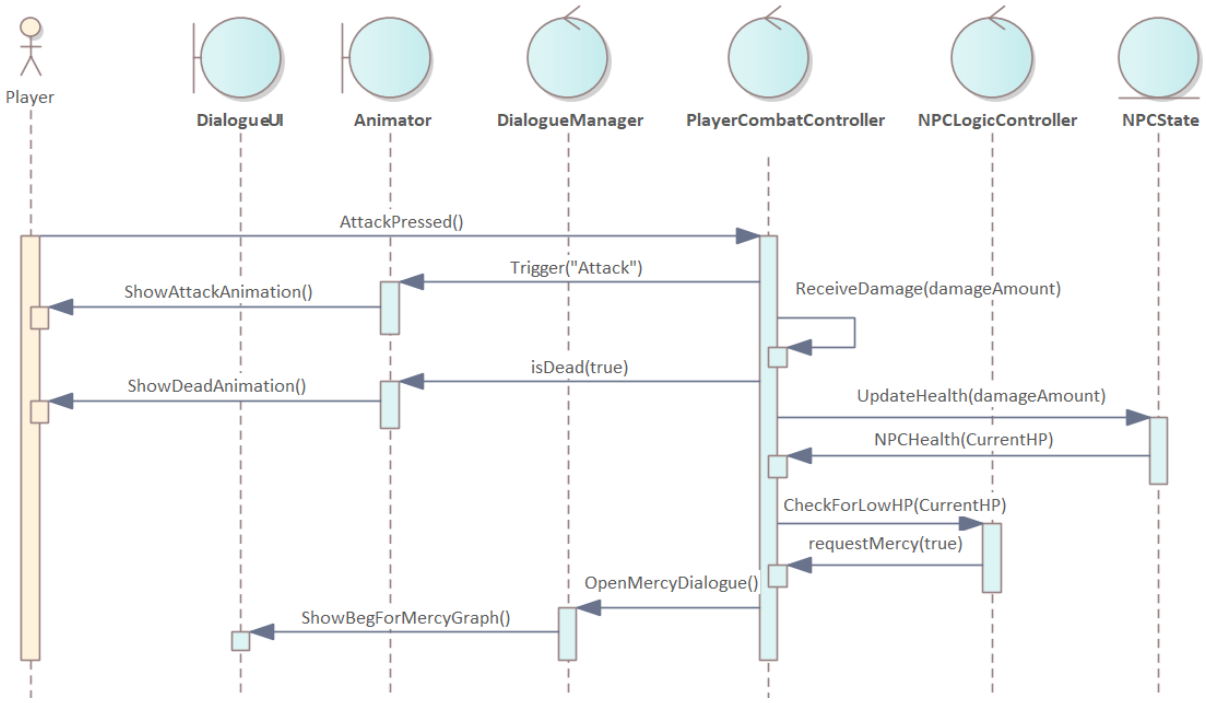


Рисунок 2.8 – Діаграма послідовності для сценарію «Виконати атаку»

Джерело: сформовано автором на основі виконаного дослідження

У цьому сценарії одночасно працюють кілька підсистем: бойова, діалогова, анімаційна та система збереження станів. Таке поєднання забезпечує реалістичність поведінки NPC і глибше занурення гравця в ігровий процес.

Діаграма на *рис.2.9* ілюструє сценарій переходу від діалогу з NPC до бойового режиму, що ініціюється гравцем через обрану репліку. Вона демонструє, як діалогова система взаємодіє з підсистемою бою, станом NPC та анімаційним контролером для підготовки NPC до бою.

Цей сценарій ілюструє ключову особливість адаптивної поведінки NPC, а саме можливість логічного переходу від діалогу до бою на основі дій гравця. Він охоплює зв’язок між діалоговою системою, станом NPC, бойовою логікою та анімацією.

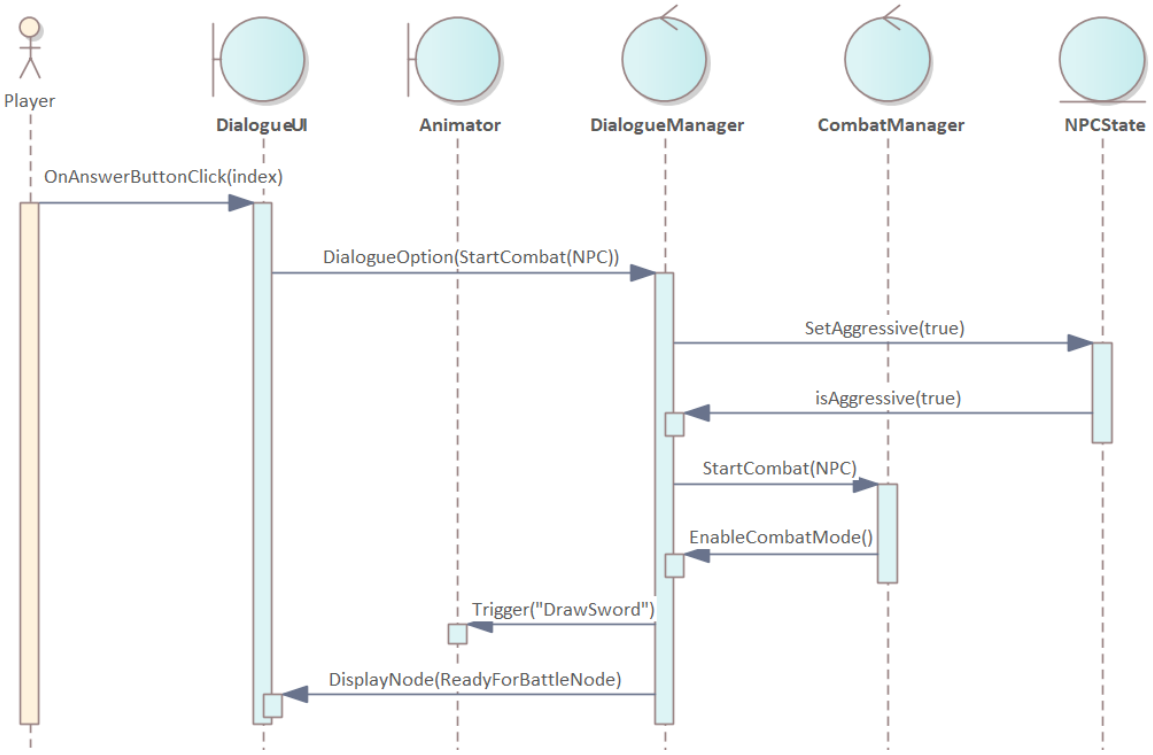


Рисунок 2.9 – Діаграма послідовності для сценарію «Ініціювати бій»

Джерело: сформовано автором на основі виконаного дослідження

Побудовані діаграми послідовностей дозволяють не лише зафіксувати логіку реакцій NPC на дії гравця, але й продемонструвати взаємозв'язки між окремими підсистемами: діалоговою, бойовою, анімаційною та контролем внутрішнього стану. Вони відображають динаміку обробки вхідних подій у реальному часі, врахування контексту, зміну станів та формування вихідних реакцій. Таким чином, діаграми послідовностей забезпечують необхідний рівень деталізації для подальшої реалізації функціоналу, тестування системи й адаптації логіки під нові сценарії.

Для відображення загальної логіки функціонування адаптивної підсистеми NPC використано діаграму активності (*рис.2.10*), що дозволяє описати послідовність дій та умов переходів у межах певного процесу або логічної поведінки системи. У контексті розробки інтерактивної гри така діаграма дає змогу візуалізувати алгоритм прийняття рішень NPC залежно від параметрів внутрішнього стану, дій гравця та зовнішніх умов.

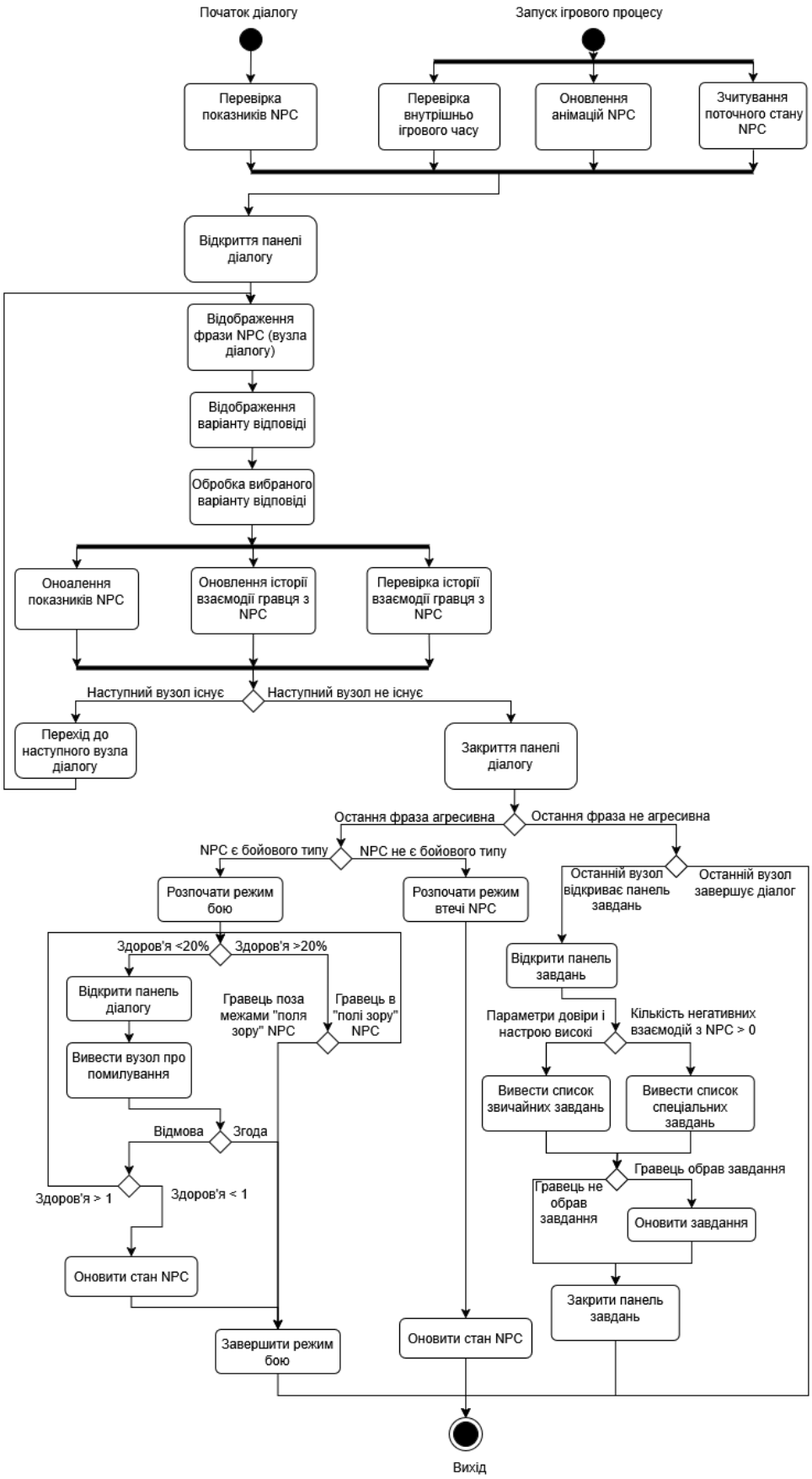


Рисунок 2.10 – Діаграма активності

Джерело: сформовано автором на основі виконаного дослідження

На діаграмі активності зображено загальну логіку роботи інформаційної підсистеми адаптивної поведінки NPC у грі. Вона демонструє, як NPC реагує на дії гравця в контексті діалогу, бойових сценаріїв та взаємодії з квестами, залежно від внутрішнього стану персонажа й умов середовища.

Діаграма починається з паралельного запуску кількох фонових процесів, що працюють постійно протягом усієї гри: перевірка показників NPC, оновлення анімацій, зчитування поточного стану та оновлення внутрішньоігрового часу. Ці процеси є незалежними й забезпечують динамічну адаптацію NPC до ігрового світу.

Після ініціації діалогу гравцем відкривається панель діалогу, зчитуються фрази NPC та варіанти відповіді. Гравець обирає одну з відповідей, після чого система:

- оновлює параметри NPC;
- оновлює історію взаємодії;
- перевіряє, як попередні дії гравця вплинули на NPC;
- ухвалює рішення щодо подальших дій.

Залежно від контексту та стану NPC, система:

- або переходить до наступного вузла діалогу,
- або запускає бойову взаємодію,
- або переводить NPC у режим втечі,
- або відкриває панель із квестами (звичайними чи спеціальними),
- або відкриває гілку діалогу з проханням про помилування (якщо здоров’я персонажа критично низьке).

Кожна з цих гілок супроводжується оновленням стану NPC, збереженням даних і, за потреби, завершенням бою. Наприкінці процесу панель діалогу або бойовий режим закривається, і система повертається до очікування нової події.

Загалом, діаграма підкреслює подієво-орієнтовану архітектуру, де адаптивна реакція NPC формується на основі станів, історії взаємодій і вибору гравця, з інтеграцією діалогової логіки, бойових дій та квестової системи.

2.3.2 Моделювання структури системи

Для опису структури інформаційної підсистеми гри, яка реалізує адаптивну поведінку NPC, було створено діаграму класів UML. Вона дозволяє відобразити основні класи, їх атрибути, методи, а також зв’язки між ними. Це дає змогу чітко структурувати систему, визначити відповідальність окремих компонентів та забезпечити цілісність архітектури.

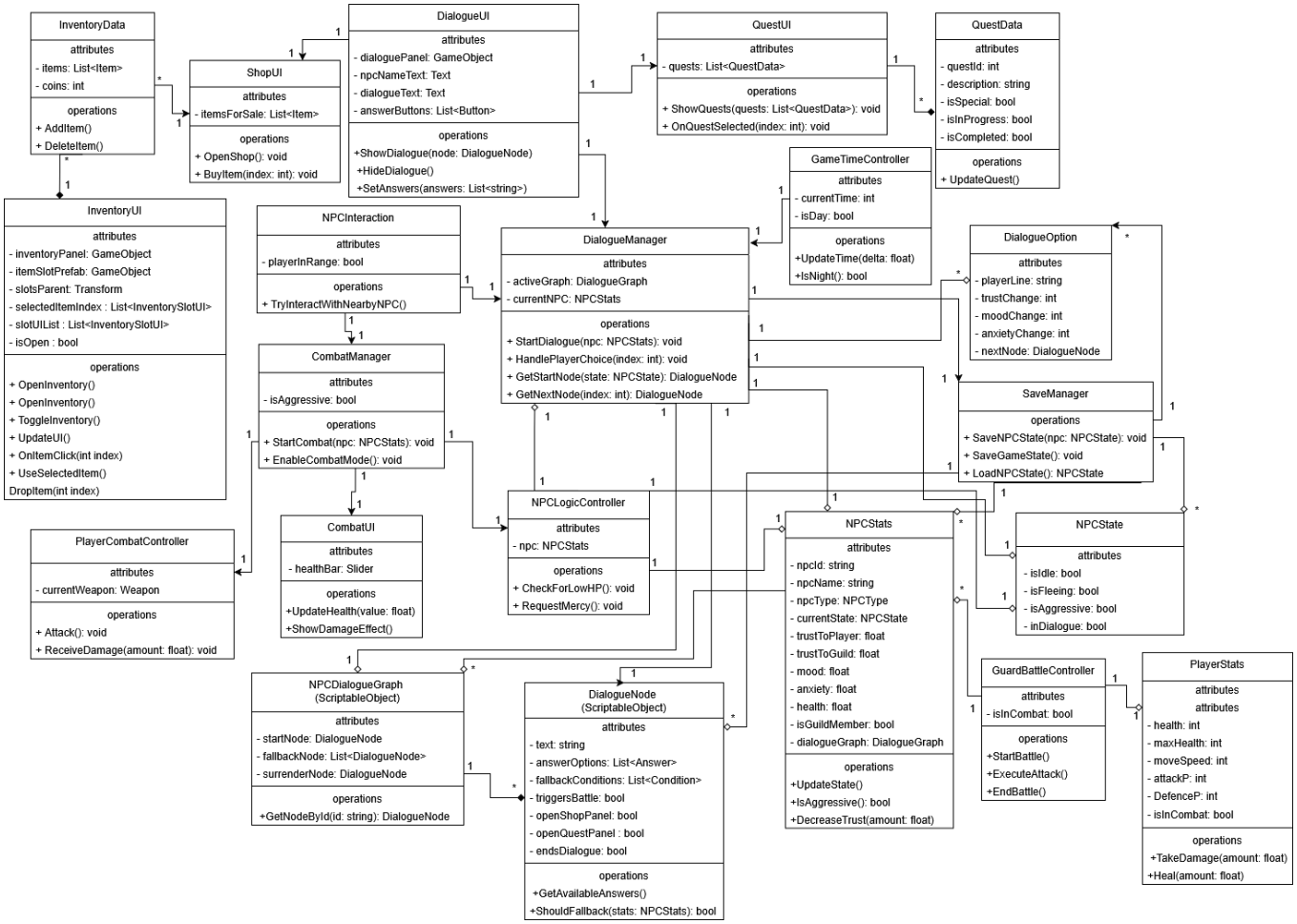


Рисунок 2.11 – Діаграма класів

Джерело: сформовано автором на основі виконаного дослідження

На діаграмі представлені такі класи:

1) Граничні класи (інтерфейси UI):

DialogueUI, ShopUI, QuestUI, InventoryUI, CombatUI відповідають за вивід інформації гравцеві та обробку його взаємодій з інтерфейсом.

2) Класи-контролери (керуючі логікою):

DialogueManager, CombatManager, PlayerCombatController, NPCLogicController, GuardBattleController реалізують логіку, приймають рішення та координують взаємодії між даними, станами та інтерфейсами.

3) Класи-сутності:

NPCStats: зберігає емоційний та поведінковий стан NPC.

NPCState: містить інформацію про поточну активність або стан NPC.

PlayerStats: дані гравця, включно зі здоров’ям та іншими параметрами.

InventoryData, QuestData: структури даних інвентарю та квестів.

4) Дані та конфігурації (ScriptableObject):

NPCDialogueGraph, DialogueNode забезпечують збереження діалогів у вигляді вузлів і структур діалогу. DialogueOption окремі варіанти реплік у діалозі.

5) Системні модулі:

SaveManager відповідає за збереження станів гри.

GameTimeController відповідає за керування внутрішньоігровим часом.

Створена діаграма класів чітко ілюструє логічну організацію адаптивної інформаційної підсистеми, що охоплює ключові компоненти гри: діалогову систему, бойовий модуль, управління часом, збереженням, інвентарем та завданнями. Завдяки виділенню граничних класів, класів-сутностей та керуючих компонентів забезпечено прозоре розмежування обов’язків між елементами системи. Виявлені зв’язки — як асоціації, так і композиції — дають змогу прослідкувати потоки даних і залежності між об'єктами. Це полегшує не лише реалізацію, а й подальшу підтримку та масштабування гри, оскільки кожен клас виконує чітко визначену функцію, а їхня взаємодія ґрунтується на структурованій логіці.

2.3.3 Розподіл вимог за компонентами системи

На завершальному етапі моделювання інформаційної підсистеми важливим завданням є перевірка відповідності реалізованих архітектурних компонентів поставленим вимогам.

Процес трасування забезпечує логічну зв’язаність між рівнями абстракції: від загального опису потреб до конкретних технічних рішень. Завдяки цьому розробник може впевнено встановити, що жодна з критичних вимог не залишилася поза увагою, а структура системи дійсно підтримує задану поведінку.

Для реалізації трасування було використано дві діаграми:

- Діаграма трасування, яка наочно демонструє взаємозв’язки між функціональними вимогами, прецедентами та класами системи. Ця діаграма дозволяє встановити, які саме частини коду безпосередньо відповідають за обробку того чи іншого сценарію взаємодії з гравцем.
- Матриця відповідності , яка є табличною формою опису відповідності між вимогами і модулями реалізації. Вона слугує зручним інструментом для оцінки повноти покриття реалізації вимог і подальшого супроводу системи.

Трасування проводиться як для функціональних, так і для нефункціональних вимог. При цьому функціональні сценарії співвідносяться з відповідними класами-контролерами, інтерфейсами та логікою взаємодії, а нефункціональні вимоги співвідносяться з архітектурними рішеннями, структурою даних та використанням стандартів проєктування.

Перша діаграма трасування на *рисунку 2.12* ілюструє зв’язки між функціональними та нефункціональними вимогами, прецедентами та класами, що реалізують логіку діалогової системи гри та емоційної адаптації NPC. Вона охоплює основні механізми ведення діалогу, реагування NPC на обрану репліку, залежність відповіді від емоційного стану персонажа та змінний контекст середовища.

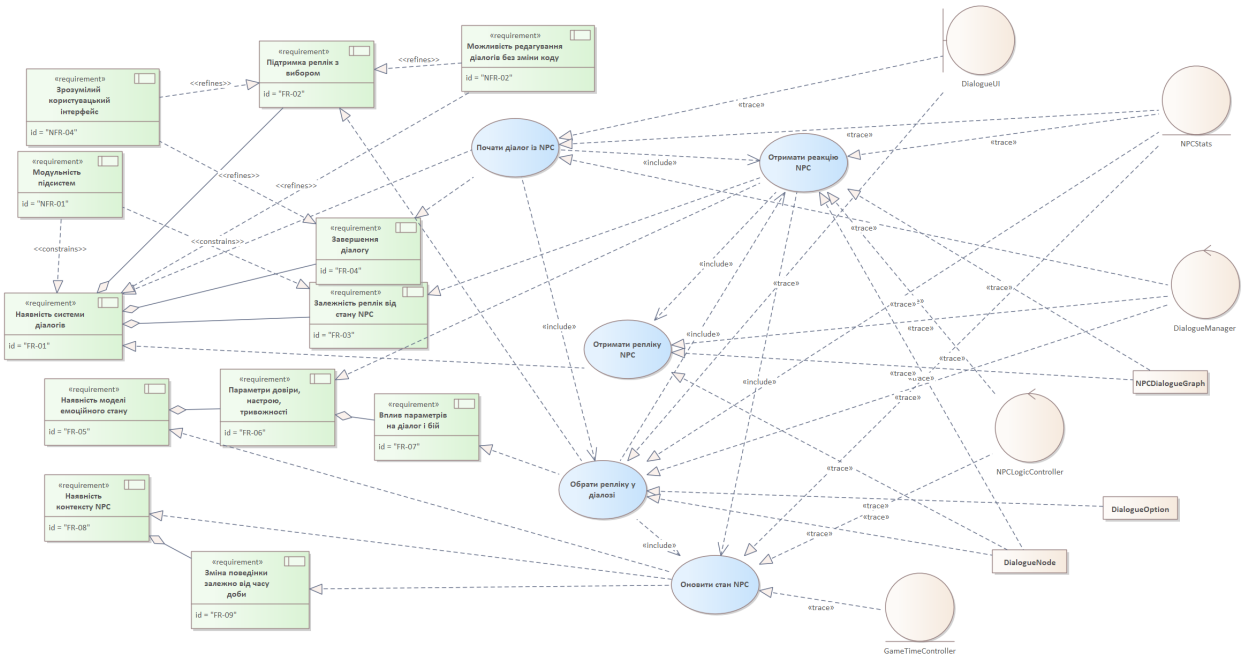


Рисунок 2.12 – Діаграма трасування для діалогової системи та емоційної моделі NPC

Джерело: сформовано автором на основі виконаного дослідження

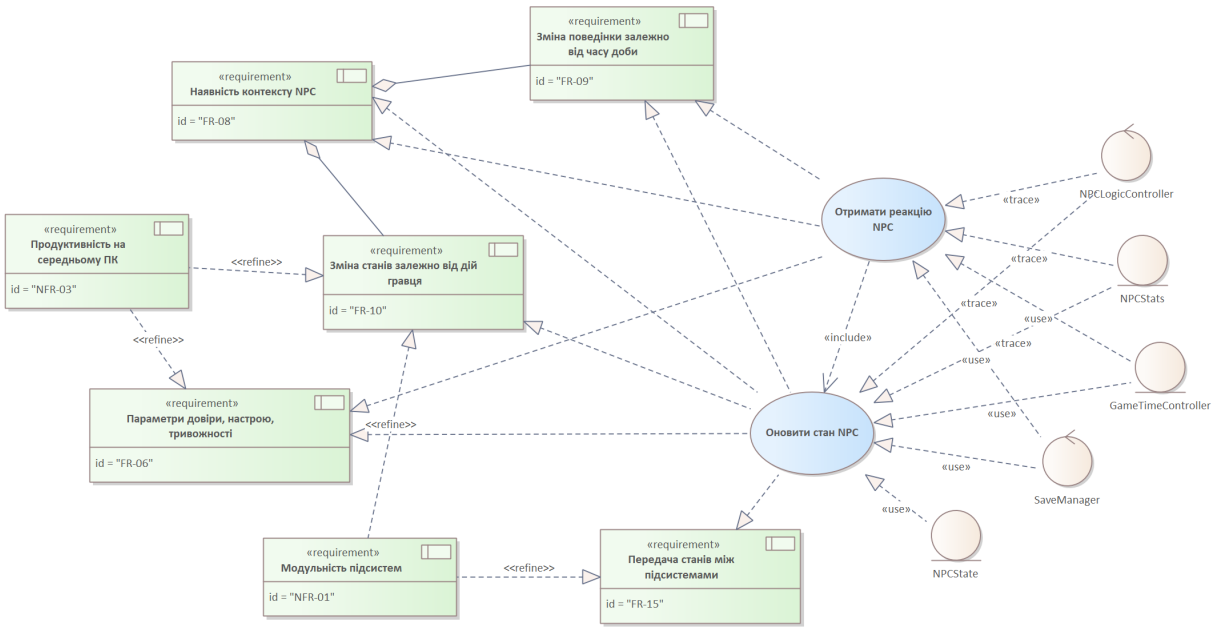


Рисунок 2.13 – Діаграма трасування для системи реакції NPC та оновлення стану

Джерело: сформовано автором на основі виконаного дослідження

Діаграма трасування на *рисунку 2.13* демонструє зв’язки між функціональними вимогами, прецедентами та класами, які реалізують логіку

реакції NPC на події в грі та оновлення їхнього внутрішнього стану. Основний акцент зроблено на параметрах емоційної моделі персонажів (довіра, настрій, тривожність), контексті середовища, а також забезпеченні узгодженості станів між підсистемами.

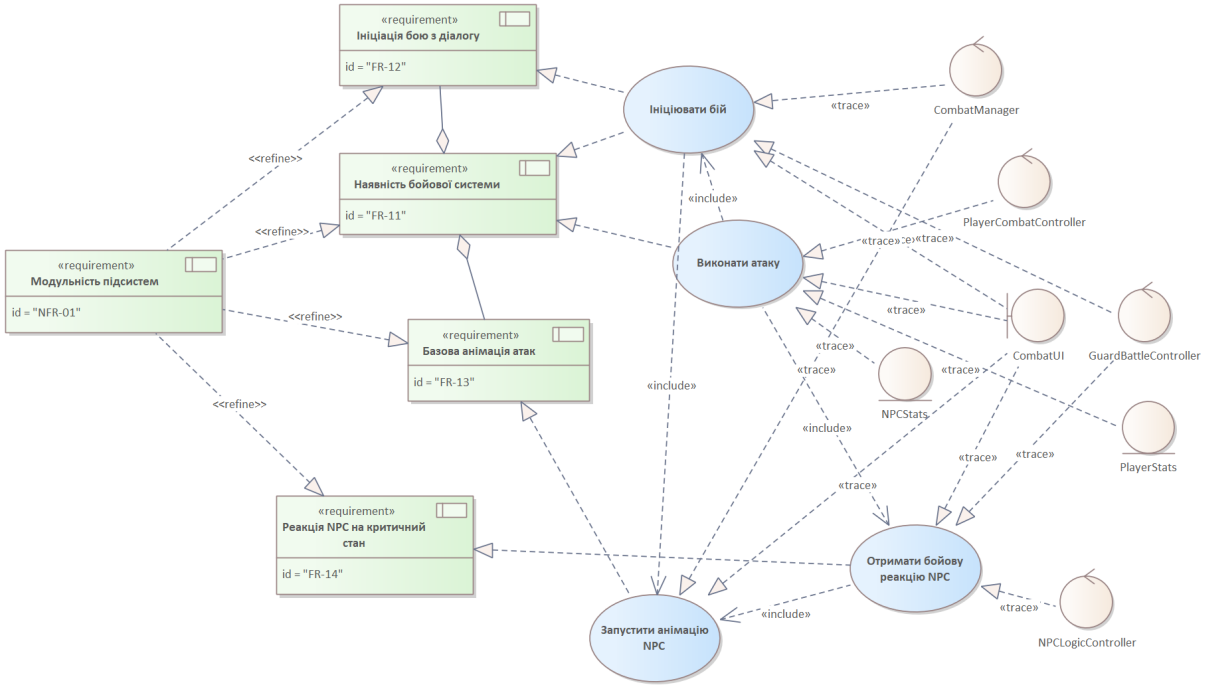


Рисунок 2.14 – Діаграма трасування для бойової системи

Джерело: сформовано автором на основі виконаного дослідження

Діаграма на *рисунку 2.14* відображає трасування функціональних і нефункціональних вимог до бойової підсистеми гри, включно з логікою бою, реакцією NPC на критичний стан, а також візуалізацією бойових подій. У центрі уваги прецеденти, що охоплюють ключові дії гравця під час бою.

Матриця трасування на *рисунку 2.15* відображає відповідність між функціональними вимогами до системи та відповідними прецедентами, реалізованими у проєкті. Такий підхід дозволяє перевірити повноту реалізації функціональних вимог і переконатись, що кожна з них підтримується хоча б одним прецедентом системи.

Source \ Target	Базова анімація атак	Вплив параметрів на діалог і б	Завершення діалогу	Залежність реплік від стану NP	Зміна поведінки залежно від	Зміна станів залежно від дій г	Зрозумілий користувацький ін	Ініціація бою з діалогу	Модульність підсистем	Можливість редагування діал	Наявність бойової системи	Наявність контексту NPC	Наявність моделі емоційного	Наявність системи діалогів	Не функціональні вимоги	Параметри довіри, настрою, т	Передача станів між підсистем	Підтримка реплік з вибором	Продуктивність на середньом	Реакція NPC на критичний ста	Функціональні вимоги
Виконати атаку											↑										
Запустити анімацію NPC	↑																				
Зберегти стан NPC																					
Ініціювати бій								↑			↑										
Обрати репліку у діалозі		↑																↑			
Оновити стан NPC					↑	↑						↑	↑			↑	↑				
Отримати бойову реакці...																				↑	
Отримати реакцію NPC				↑	↑							↑				↑					
Отримати репліку NPC														↑							
Почати діалог із NPC			↑											↑							

Рисунок 2.15 – Матриця трасування

Джерело: сформовано автором на основі виконаного дослідження

По горизонталі (Target) подано функціональні та нефункціональні вимоги, що визначають архітектуру, поведінку NPC, логіку бою, діалогів, інтерфейс користувача, контекстну поведінку та інші аспекти. По вертикалі (Source) подано основні прецеденти взаємодії гравця з NPC як у рамках діалогової взаємодії, так і бойових сценаріїв.

Побудована матриця забезпечує повну трасованість між вимогами та функціональністю системи, що відповідає принципам структурного моделювання відповідно до стандартів розробки програмного забезпечення.

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ГРИ

3.1 Інформаційне забезпечення

3.1.1 Загальна характеристика інформаційного забезпечення

Інформаційне забезпечення гри ґрунтується на принципах локального збереження, гнучкої структури даних та модульного доступу до інформаційних об’єктів [1]. В якості постійного сховища інформації використовуються файли формату JSON, які зберігають поточні стани NPC, історію взаємодій, прогрес гравця, а також внутрішньоігрові параметри (Такі, як час доби, здоров’я, виконані квести, інвентар граівця) [3]. Крім цього, для зберігання конфігурацій, діалогових графів та шаблонних даних використовуються об’єкти типу ScriptableObject, які є частиною середовища Unity і не потребують зовнішньої СКБД.

У грі немає централізованої реляційної бази даних. Натомість, з метою забезпечення швидкого доступу до конфігураційних даних, їх ініціалізації в пам’яті та подальшого оновлення під час виконання гри, використовується структура об'єктів, серіалізованих у JSON-файли та ScriptableObject-файли. Всі зміни, що відбуваються під час ігрового процесу, можуть бути збережені до JSON-файлів через SaveManager, який виконує функцію обробника транзакцій збереження [8].

Методи контролю інформації реалізовано через валідацію даних при серіалізації та завантаженні: усі критичні об’єкти проходять перевірку на повноту та коректність. Вимоги до надійності забезпечуються за рахунок дублювання збережень, а також логічного контролю з боку SaveManager. Достовірність інформації підтримується шляхом ізоляції доступу до важливих станів NPC, які доступні виключно через відповідні контролери (NPCLogicController, DialogueManager, CombatManager).

Носіями інформації є локальні ресурси гри (в Assets), а також внутрішні runtime-файли у форматі .json, що генеруються при збереженні стану гри. Тип

носія: файл на локальному диску. Обсяг кожного JSON-файлу не перевищує 50–100 КБ, а завантаження відбувається лише при потребі.

Таким чином, інформаційне забезпечення гри є подієво-орієнтованим, не централізованим, а розподіленим за принципом компонентного підходу. Вся інформація розділена за функціональними сферами (бойова логіка, діалоги, емоційний стан, квести, внутрішньо ігровий час) та обслуговується відповідними класами-контролерами, що відповідають за доступ, оновлення та інтеграцію між підсистемами [2].

Умовно-узагальнена схема інформаційного забезпечення включає:

1. Постійне сховище конфігурацій у вигляді ScriptableObject-об’єктів (NPCDialogueGraph, DialogueNode, QuestData).
2. Поточний стан гри у вигляді JSON-файлів (npc_stats.json, player_save.json, worldstate.json).
3. Тимчасові структури в пам’яті такі, як класи NPCStats, PlayerStats, InventoryData, які оновлюються під час гри.

Такий підхід відповідає нефункціональній вимозі модульності підсистем (NFR-01), оскільки дозволяє змінювати спосіб зберігання даних, не впливаючи на решту логіки гри.

3.1.2 Організація збору і передання первинної інформації

У грі процес збору первинної інформації не пов’язаний з традиційними джерелами, такими як зовнішні бази даних або інформаційні підрозділи. Натомість, джерелами первинної інформації виступає сама гра, а саме дії гравця, внутрішній стан NPC, контекст середовища та часові події. Уся інформація збирається та обробляється в реальному часі, під час виконання гри в середовищі Unity.

Джерела первинної інформації:

- гравець – ініціює взаємодії (початок діалогу, вибір репліки, атака тощо);
- NPC – мають власний емоційний стан, який формується та змінюється залежно від дій гравця;

- система часу (GameTimeController) – передає значення поточної доби, що впливає на поведінку NPC;
- контролери логіки (NPCLogicController, DialogueManager, CombatManager) - обробляють усі зміни в поведінці та реакціях NPC і гравця;
- інтерфейс користувача (DialogueUI, CombatUI, ShopUI, QuestUI) – фіксує вибрані гравцем дії, які виступають подіями, що запускають оновлення станів.

Спосіб передання інформації:

Інформація передається між компонентами системи гри у вигляді:

1. викликів методів та передачі об’єктів у пам’яті між класами-контролерами, сутностями та UI;
2. збереження стану гри у форматі JSON при ключових подіях, як-от завершення діалогу, бою, прийняття квесту тощо.

Періодичність збору та оновлення:

- інформація оновлюється лише при виникненні події – взаємодія, репліка, бойова дія;
- зміна доби перевіряється періодично з певним інтервалом;
- при запуску гри завантажуються всі JSON-файли, необхідні для відновлення станів.

Носії інформації:

1. усі поточні стани NPC та гравця зберігаються в оперативній пам’яті під час гри;
2. файли JSON.

Відповідальні компоненти:

- SaveManager – відповідає за збереження та завантаження всіх ключових станів у JSON;
- NPCLogicController, DialogueManager, CombatManager – формують, змінюють та передають інформацію між підсистемами;
- GameTimeController – передає час доби та дату іншим системам;

- UI-класи – ініціюють збір інформації на основі дій гравця.

Таким чином, система збору і передання первинної інформації в грі є реактивною, подієво-орієнтованою та модульною, що відповідає загальним принципам архітектури і забезпечує цілісну поведінку NPC в реальному часі.

3.1.3 Побудова системи класифікації та кодування

У процесі створення гри з адаптивною поведінкою NPC була реалізована внутрішня система класифікації та кодування, що забезпечує унікальну ідентифікацію об’єктів гри та підтримку логіки взаємодії між компонентами. Через відсутність централізованої системи керування базами даних, дані зберігаються у вигляді JSON-файлів та ScriptableObject-об’єктів у середовищі Unity, тому кодування застосовується локально — для забезпечення унікальності та структурування.

Класифікація NPC передбачає присвоєння кожному неігровому персонажу унікального ідентифікатора, який записується вручну. Такий ідентифікатор використовується як ключ у збереженні даних про NPC, включно з рівнем довіри, настроєм, тривожністю, станом здоров’я та іншими характеристиками. Крім того, NPC класифікуються за ролями (наприклад, цивільний, охоронець, майстер гільдії), що визначає їхню поведінку, хоча структурованого поділу діалогів за типами NPC на поточному етапі ще не реалізовано.

Завдання також мають власну систему ідентифікації. Вона базується на поєднанні номера локації та номера квесту всередині цієї локації. Це дозволяє впорядковано зберігати й ідентифікувати завдання незалежно від їхнього місця розташування у світі гри.

Предмети інвентарю класифікуються за категоріями та конкретними видами всередині кожної категорії. Для кожного предмета створено унікальний код, який дозволяє точно визначити його тип і використовувати при завантаженні чи оновленні інвентаря гравця з файлу.

Зброя має складнішу ієрархічну систему кодування. Вона враховує призначення зброї (наприклад, для атаки чи захисту), її клас (меч, спис, лук тощо), а також унікальний номер конкретної зброї. Це дозволяє зберігати до сотень унікальних екземплярів кожного типу зброї у відповідних JSON-структурах.

Діалоги представлені у вигляді ScriptableObject-структур. Графи діалогів формуються вручну і мають унікальні назви, що виконують функцію ідентифікатора. Пряме поле з ID для кожного вузла діалогу (наприклад, окремої репліки) поки не передбачено, але вузли пов’язуються між собою через об’єкт-граф, а також мають поведінкові параметри (наприклад, перехід до торгівлі, виклик бою, або відкриття квесту). Крім цього, деякі вузли відповідають умовам (довіра, настрій, тривожність), що дозволяє трактувати їх як поведінкові категорії.

На даному етапі всі ідентифікатори присвоюються вручну через інспектор Unity або заздалегідь підготовлені JSON-файли. Це забезпечує повний контроль над структурою і дозволяє уникнути конфліктів. У подальшому можливе впровадження автоматичної генерації ідентифікаторів, зокрема при створенні великої кількості NPC або квестів.

Таблиця 3.1 – Таблиця класифікаторів ідентифікаторів

Назва об’єкта	Формат ID	Приклад ID	Примітка
NPC (звичайний)	NPC_NORM_XXX	NPC_NORM_001	Для звичайний NPC, які не мають особливого призначення.
Merchant (купець)	NPC_MERCH_XXX	NPC_MERCH_001	Кількість цих NPC сильно обмежена, тому немає сенсу ускладнювати ID, як з предметами інвентарю.
Guild Master (Майстер гільдії)	NPC_GMaster_XX	NPC_GMaster_01	
Guard (охоронець)	NPC_GUA_XXX	NPC_GUA_001	Охоронець є бойовим типом NPC.

Продовження таблиці 3.1

Назва об’єкта	Формат ID	Приклад ID	Примітка
Завдання (Quest)	QUESTLLNNN	QUEST01001	LL – локація, NNN – номер квесту
Предмет (Item)	ITEMCCVV	ITEM0101	CC – категорія, VV – вид у категорії
Зброя (Weapon)	W_TTVVV	W_101004	T – тип (1 – атака, 2 – захист), TT – клас, VVV – варіант
Діалоговий граф	Назва SO-файлу	GuildMaster_Graph	Назва файлу ScriptableObject
Вузол діалогу	Назва SO-файлу	GMaster_FD_Response_01	Ідентифікується через структуру графу

Джерело: сформовано автором на основі виконаного дослідження

У проєкті реалізовано кілька паралельних систем класифікації та кодування, які охоплюють NPC, квести, предмети, зброю та діалоги. Їхнє впровадження забезпечує впорядкованість структури даних, простоту масштабування та підтримку цілісності гри без використання зовнішніх баз даних. Усі ідентифікатори підтримують зрозумілу логіку побудови, яка уможливлює відстеження походження, типу та ролі об’єкта в межах системи.

3.1.4 Проєктування форм первинних документів та відеокадрів

У контексті розробки комп’ютерної гри традиційне розуміння первинних документів трансформується в інтерактивні візуальні елементи, які забезпечують збирання, обробку та подання інформації в межах геймплейної взаємодії. Такими елементами є UI-форми (панелі інтерфейсу користувача) та відеокадри, що виконують функції інформаційного забезпечення замість класичних машинограм і форм обліку, притаманних інформаційним системам.

У проєкті реалізовано низку форм UI, кожна з яких виконує специфічну функцію. Зокрема, інтерфейс діалогу (DialogueUI) забезпечує взаємодію між гравцем і неігровими персонажами (NPC), а саме відображає репліки, дозволяє

вибирати варіанти відповіді, а також запускає відповідні події, як зміна настрою NPC, початок бою, чи виклик інших інтерфейсів (магазин або дошка завдань).

Інтерфейс інвентарю (InventoryUI) відображає доступні гравцеві предмети, дозволяє переглядати їх характеристики, використовувати, а також сортувати за категоріями. Аналогічно, інтерфейс квестів (QuestUI) надає інформацію про поточні та завершені завдання, їх опис, цілі та нагороду.

Окремо реалізовано інтерфейс торгівлі (ShopUI), який дозволяє купувати й продавати предмети між гравцем і NPC, та бойовий інтерфейс (CombatUI), що відображає бойові стани, індикатори здоров'я, бойові повідомлення тощо.

Важливим елементом інформаційного забезпечення є також внутрішньоігрові відеокадри. Це або короткі анімовані сценки, або візуальні переходи між станами, які супроводжують ігрові події (наприклад, реакція NPC на поведінку гравця, перемога в бою).

Хоча традиційні документи у грі відсутні, кожна UI-форма має унікальний скрипт, що реалізує її логіку, та стандартизовані методи керування (OpenPanel, ClosePanel, UpdateUI тощо). Це дозволяє контролерам гри централізовано управляти виведенням інформації та обробкою дій гравця.

Таким чином, в адаптованому до ігрового середовища контексті формами первинного інформаційного забезпечення виступають графічні інтерфейси та відеокадри, що виконують функції подання, збору й обробки ігрової інформації.

3.1.5 Структура інформаційних масивів

У проєкті не використовується класична реляційна база даних, проте структура інформаційних масивів реалізована за допомогою зовнішніх файлів формату JSON. Вони зберігають ключові дані, що забезпечують функціонування гри: стани персонажів, інвентар, квести, час доби та прогрес гравця. У цьому випадку JSON-файли виступають аналогами інформаційних масивів, що розмежовані за логікою роботи відповідних підсистем.

Кожен масив має власну структуру й організований як набір об’єктів з фіксованими атрибутами. Всі масиви мають унікальні ідентифікатори записів, що забезпечують цілісність і зручність звертання до окремих об'єктів гри. Оскільки збереження інформації відбувається локально, без серверної частини чи СКБД, масиви залишаються відкритими для модифікації, що підвищує гнучкість при тестуванні та доопрацюванні гри.

До основних інформаційних масивів у проєкті належать:

- NPCData – містить параметри кожного неігрового персонажа (довіра до гравця та гільдії, настрій, тривожність, здоров’я, належність до типу).
- InventoryData – відображає предмети, якими володіє гравець. Ідентифікатори предметів кодуються за категорією та підкатегорією, наприклад: ITEM0101, WPN101004.
- QuestData – фіксує активні та завершені квести з відповідним статусом. Ідентифікатори мають формат QUEST01001, де перші дві цифри після слова QUEST означають номер локації, а наступні порядковий номер квесту.
- GameTimeData – фіксує поточний час у грі, що впливає на поведінку NPC залежно від частини доби.
- ScriptableObject-файли – використовуються для опису діалогових структур. Наприклад, NPCDialogueGraph містить початковий вузол діалогу, вузли умовного fallback-переходу тощо. Хоча такі об’єкти не мають ID у звичному розумінні, логіка гри забезпечує унікальність шляхом структури з'єднань і зв’язків між вузлами.

Доступ до кожного з цих масивів здійснюється через спеціальні керуючі компоненти, зокрема SaveManager, DialogueManager, InventoryManager. Ці класи відповідають за зчитування, оновлення та запис відповідних JSON-файлів.

У межах реалізованої структури також дотримано базових вимог до цілісності інформації. Для параметрів, таких як настрій, тривожність, довіра, визначено допустимі діапазони значень (від 0 до 100). Ідентифікатори не дублюються, що гарантує коректність зв’язків між даними.

Отже, структура інформаційних масивів у проєкті є чітко організованою, модульною і здатна до масштабування. Такий підхід дозволяє уникнути використання повноцінної СКБД, забезпечуючи водночас гнучкість і контроль над даними гри.

3.1.6 Вибір СКБД

У межах розробки однокористувацької гри з адаптивною поведінкою NPC використання класичної системи керування базами даних, такої як MySQL, PostgreSQL або SQLite, не було доцільним. Замість цього в проєкті застосовано локальну файлову систему збереження, засновану на структурованих файлах у форматі JSON.

Використання JSON-файлів забезпечує низку важливих переваг:

1. Формат JSON легко інтегрується із середовищем Unity. Завдяки цьому збереження і завантаження ігрових даних (NPC, інвентар, квести, параметри гравця) реалізовано без залучення сторонніх СКБД або серверної інфраструктури.
2. Рішення не потребує окремого середовища виконання або встановлення СКБД. Усі дані зберігаються локально, що ідеально підходить для автономної офлайн-гри.
3. JSON-файли можна редагувати вручну, що спрощує налаштування, налагодження та балансування ігрових параметрів без потреби компіляції або створення SQL-запитів.
4. Дані поділені на окремі логічні блоки, зокрема: npc_data.json для NPC, player_data.json для гравця, inventory_data.json для інвентарю, quest_data.json для квестів. Така структура полегшує підтримку і масштабування проєкту.
5. Обсяги даних у грі відносно невеликі, тому читання і запис JSON-файлів здійснюється швидко, без помітних затримок.

Збереженням і завантаженням гри опікуються спеціальні менеджери:

- SaveManager це основний компонент, що відповідає за серіалізацію та десеріалізацію даних;
- GameDataManager координує доступ до інвентаря, квестів, характеристик;
- DialogueManager працює з діалоговими структурами, представленими у вигляді ScriptableObject.

У рамках попереднього аналізу також розглядалося використання SQLite як вбудованої СКБД. Проте ця опція була відкинута через надлишкову складність і відсутність необхідності в реляційних структурах або складних запитах.

3.1.7 Інфологічна модель бази (сховища) даних

У даному проєкті класична СКБД не використовується. Натомість реалізовано формат зберігання даних у вигляді JSON-файлів, які забезпечують структурований і незалежний від платформи спосіб збереження ігрового прогресу. Проте, попри відсутність реляційної СКБД, для цілей логічного аналізу та забезпечення послідовності даних побудована інфологічна модель — абстрактне логічне представлення основних сутностей гри та зв’язків між ними.

Інфологічна модель базується на описі логічних об’єктів, що використовуються в грі, таких як гравець, NPC, діалоги, квести, предмети інвентарю, бойовий стан, а також описує, яким чином ці об’єкти взаємодіють та зв’язуються між собою.

Основні сутності інфологічної моделі:

Player (Гравець) – логічна сутність, яка містить основні параметри гравця: здоров’я, інвентар, виконані квести, позицію у світі гри. Має зв’язки з об’єктами інвентарю, квестами, бойовою логікою, а також із системою збереження.

NPC (Неігровий персонаж) – абстрактне уявлення про персонажа, з яким взаємодіє гравець. Містить посилання на емоційний стан, тип NPC (наприклад, торговець, охоронець), та пов’язані діалогові графи.

NPCStats – набір параметрів NPC, що характеризують його поточний стан: рівень довіри до гравця/гільдії, настрій, тривожність, здоров’я тощо. Ці параметри безпосередньо впливають на діалогову систему та бойову поведінку.

NPCDialogueGraph – об’єкт типу ScriptableObject, який визначає діалогову логіку NPC. Містить стартовий вузол, вузол капітуляції (наприклад, при низькому здоров’ї), список fallback-вузлів, які активуються залежно від стану NPC.

DialogueNode – окрема репліка NPC у діалозі. Містить текст, список можливих варіантів відповіді гравця та дії, які виконуються при активації вузла (наприклад, відкриття магазину або початок бою).

DialogueOption – варіант відповіді гравця у вузлі діалогу. Веде до іншого вузла або викликає зміну станів (емоційних, поведінкових).

Quest (Квест) – логічна одиниця завдання. Має унікальний ідентифікатор, опис, статус (активний, виконаний, провалений) і посилання на NPC або події, що його активують. Дані зберігаються в окремому масиві, доступному через систему збереження.

InventoryItem – абстракція об’єкта в інвентарі. Кожен предмет має унікальний ідентифікатор, категорію, тип і стан (наприклад, активний, використаний). Пов’язаний із гравцем.

GameTimeController – логічний компонент, що зберігає та керує внутрішньоігровим часом. Його значення впливають на поведінку NPC, наприклад, відмову від спілкування вночі.

SaveManager – центральний модуль, що керує збереженням та завантаженням даних гри. Взаємодіє з усіма ключовими сутностями: Player, NPC, Inventory, Quest, Dialogue.

У проєкті ігрової системи ключові зв’язки між сутностями визначаються на основі логіки їх взаємодії у грі. Кожен клас у моделі відіграє конкретну роль у процесах збереження, діалогу та взаємодії з гравцем. Опис нижче деталізує характер зв’язків.

Гравець (Player) має:

- множину предметів інвентарю (InventoryItem): гравець може мати бачаго предметів, а кожен предмет належить лише одному гравцеві. Зв’язок типу 1 до 0..*;
- множину завдань (Quest), які гравець отримує, виконує або провалює. Гравець може мати багато завдань одночасно Зв’язок 1 до 0..*;
- зв’язок зі SaveManager: дані гравця зберігаються централізовано. Зв’язок 1 до 1.

Неігровий персонаж (NPC) має:

- один об’єкт NPCStats, який містить поточний стан (довіру, настрій, тривожність тощо). Зв’язок 1 до 1;
- діалоговий граф (NPCDialogueGraph), що визначає сценарій взаємодії з гравцем. NPC може не мати графа (наприклад, якщо він не взаємодіє через діалоги), або мати більше 1 графа. Зв’язок 1 до 0..*.

NPCDialogueGraph складається з:

- одного або кількох вузлів діалогу (DialogueNode), які визначають репліки NPC. Зв’язок 1 до 1..*.

DialogueNode містить:

- множину варіантів відповіді гравця (DialogueOption), що ведуть до інших вузлів або виконують певні дії. Зв’язок 1 до 0..*.

NPCStats:

- взаємодіє з GameTimeController, який визначає внутрішньоігровий час. Стан NPC може змінюватися в залежності від часу доби. Зв’язок 1 до 1;
- зв’язаний із SaveManager, оскільки емоційні параметри NPC зберігаються між сесіями гри. Зв’язок 1 до 1.

Квест (Quest):

- Виконання завдання може потребувати взаємодії з більш ніж одним NPC, один NPC може бути джерелом більш ніж одного завдання. Не

кожен NPC є джерелом завдання. Завдання не обов’язково можна отримати від NPC, для його виконання також не завжди обов’язково взаємодіяти з NPC. Зв’язок 0..* до 0..*.

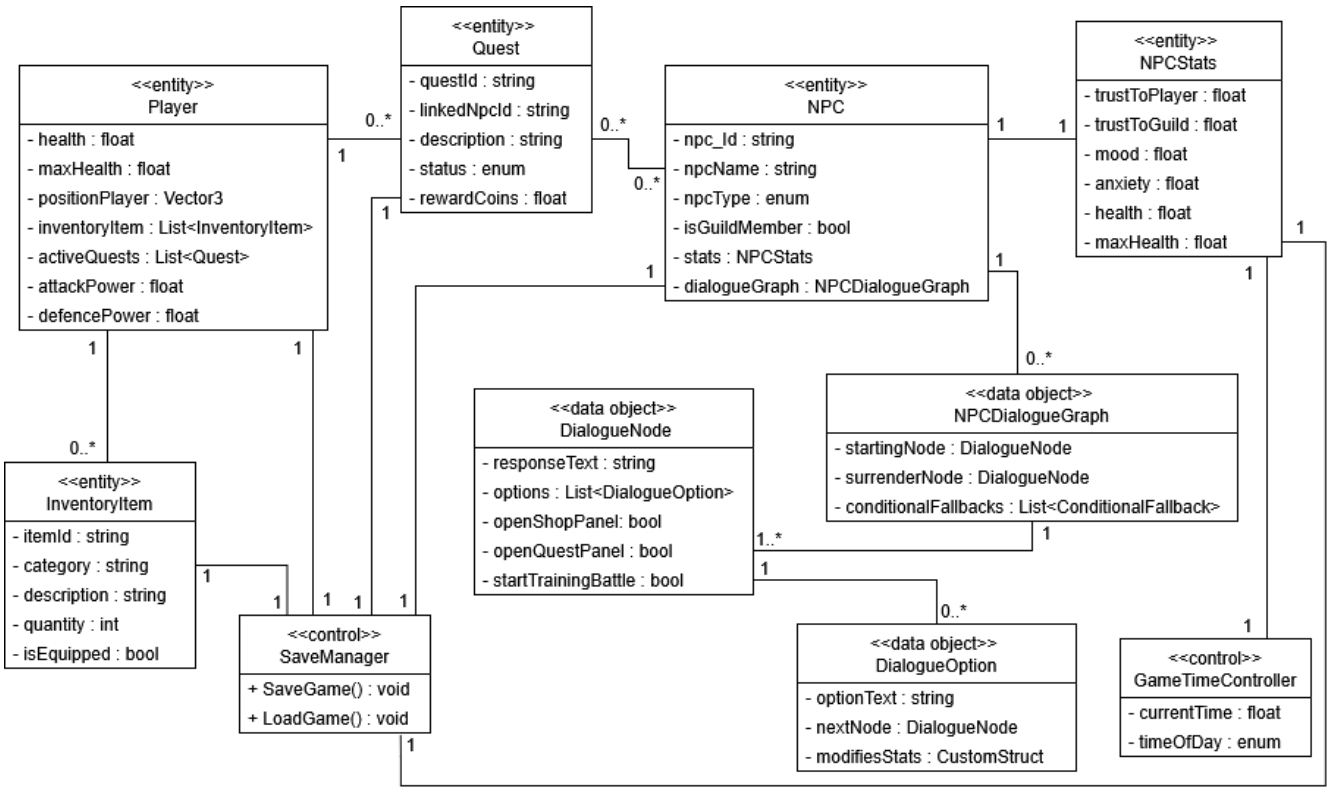


Рисунок 3.1 – Діаграма інфологічної моделі

Джерело: сформовано автором на основі виконаного дослідження

3.1.8 Даталогічна модель бази (сховища)даних

У проєкті гри не використовується класична система керування базами даних. Натомість для зберігання і обробки інформації застосовано формат JSON, який є зручним і гнучким рішенням у контексті однокористувацького ігрового середовища. Такий підхід дає змогу серіалізувати об’єкти гри у вигляді структурованих текстових файлів, що зберігаються локально та відновлюються під час запуску гри.

Дані у форматі JSON поділені за логічними блоками відповідно до основних підсистем гри. Окремі файли відповідають за:

- збереження параметрів гравця (здоров’я, інвентар, прогрес),

- стан NPC (довіра, настрій, тривожність, здоров’я, тип),
- інформацію про квести (ідентифікатор, статус, зв’язок із NPC),
- час доби та інші змінні.

Кожен об’єкт у файлі містить набір ключів і значень, які безпосередньо відповідають властивостям класів у коді проєкту. Наприклад, дані про NPC записуються у вигляді полів npcId, mood, trustToPlayer, anxiety тощо, а об’єкт InventoryItem містить такі атрибути як itemId, category, type, isEquipped.

На відміну від традиційних таблиць у реляційних БД, модель зберігання побудована за ієрархічним принципом, що дозволяє гнучко масштабувати систему та легко додавати нові поля. Структура кожного JSON-файлу визначається класами, які використовуються в ігровій логіці, і відповідає актуальному стану об’єктів у грі.

Для забезпечення централізованої обробки інформації використовується модуль SaveManager, який відповідає за збереження та завантаження даних. Він взаємодіє з іншими компонентами, зокрема NPC, гравцем, діалоговою системою, інвентарем та квестами.

Таким чином, даталогічна модель проєкту реалізована через набір JSON-файлів, які замінюють собою традиційні таблиці БД і зберігають структуровану інформацію про всі ключові елементи гри.

3.2 Технічне забезпечення

3.2.1 Загальні положення та схема автоматизації

Автоматизація задачі проєктування адаптивної поведінки NPC реалізується у вигляді локального програмного продукту, що функціонує в середовищі персонального комп’ютера. Вся інформаційна система побудована як автономний додаток без потреби у зовнішніх мережевих з’єднаннях або серверній інфраструктурі.

Програмна реалізація виконана у середовищі Unity, яке забезпечує повну інтеграцію інструментів для побудови сцени, створення ігрової логіки, організації взаємодії з NPC, керування UI, запуску анімацій, серіалізації даних та збереження прогресу. Уся обробка здійснюється локально з використанням внутрішньої логіки C#-скриптів.

Схема автоматизації передбачає взаємодію таких підсистем, як введення (ініціація дій гравцем), обробка (прийняття рішень NPC на основі внутрішніх станів і контексту), виведення (візуалізація реакцій через UI та анімацію), а також збереження (локальні JSON-файли).

Апаратне середовище складається з персонального комп'ютера, монітора, клавіатури та миші. Зберігання та обробка даних реалізовані у межах єдиної системи без використання баз даних або віддалених сховищ. Комунікація між модулями реалізується через подієво-орієнтовану архітектуру Unity та структури типу ScriptableObject.

3.2.2 Структура комплексу технічних засобів

Комплекс технічних засобів, що забезпечує функціонування розробленої інформаційної системи, має автономну локальну архітектуру. Уся система розгортається, виконується та тестується на одному персональному комп'ютері без залучення серверної інфраструктури чи мережевих сервісів.

Серед основних елементів комплексу є:

Обчислювальний модуль – персональний комп'ютер, на якому розміщено середовище розробки (Unity) і всі компоненти гри. Він виконує обробку логіки, симуляцію внутрішніх станів NPC, візуалізацію ігрового середовища, а також взаємодію з користувачем.

Графічний модуль – система відображення інтерфейсу, анімацій та сцен у реальному часі. Реалізується через GPU комп'ютера у зв'язці з монітором.

Підсистема керування даними – локальна файлово-орієнтована система збереження, що оперує файлами у форматі JSON. Вона відповідає за зберігання ігрового прогресу, станів NPC, параметрів середовища та іншої інформації.

Засоби введення/виведення – класичні периферійні пристрої (миша, клавіатура, монітор), через які здійснюється управління і візуальний зворотний зв’язок.

Програмне забезпечення – середовище Unity, мова програмування C#, набір скриптів, ScriptableObject-файлів, системи анімацій та обробки подій, що реалізують основну логіку адаптивної поведінки NPC.

Уся система є незалежною від зовнішніх джерел даних або мережевого доступу, що спрощує її розгортання, експлуатацію та тестування. Така структура дозволяє забезпечити стабільну роботу гри навіть в умовах обмежених ресурсів.

3.2.3 Опис автоматизованого робочого місця

Розробка гри здійснюється на персональному комп’ютері, який виконує функції повноцінного автоматизованого робочого місця (АРМ) розробника. У контексті цього проєкту автоматизоване робоче місце призначене для повного циклу проєктування, програмування, тестування та налагодження гри в середовищі Unity.

Для забезпечення коректної роботи усіх модулів гри, включаючи обробку анімацій, рендеринг 3D-сцен, діалогові дерева, бойову систему та систему збереження, використовуються такі технічні характеристики АРМ, які описані в *таблиці 3.2*.

Таблиця 3.2 – Характеристика АТМ

Компонент	Характеристика
Процесор (CPU)	12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
Оперативна пам’ять (RAM)	40.0 ГБ
Накопичувач	SSD, > 1Тб
Відеокарта (GPU)	NVIDIA GeForce RTX 3070Ti
Операційна система	Windows 11 Home, 64-розрядна операційна система, процесор на базі архітектури x64

Джерело: сформовано автором на основі виконаного дослідження

До складу АРМ також входить базовий набір периферії, характеристики якого описані в *таблиці 3.3*.

Таблиця 3.3 – Характеристика додаткових пристроїв

Пристрій	Характеристика
Монітор	З роздільною здатністю не менше Full HD (1920×1080)
Клавіатура та миша	Стандартні дротові

Джерело: сформовано автором на основі виконаного дослідження

На робочому місці встановлено наступне середовище розробки та інструменти:

Unity (версія 2022.3.44f1) – основне середовище розробки, побудови сцен, обробки діалогів, систем NPC та UI.

Visual Studio з підтримкою C# – для написання коду і підключення логіки

Blender – для редагування моделей.

Krita – для редагування текстур.

3.2.4 Схема мережі передачі даних

У межах реалізації гри як однокористувацького застосунку, мережевий режим взаємодії між клієнтами або із зовнішніми сервісами не передбачається. Уся функціональність гри реалізована на локальній машині, а обмін даними між модулями відбувається всередині системи шляхом передачі інформації між об’єктами в оперативній пам’яті або зчитуванням/записом у JSON-файли.

3.3 Програмне забезпечення

Програмне забезпечення гри реалізовано за модульним принципом із чітким поділом відповідальностей між компонентами. Кожен функціональний блок гри (персонажі, діалоги, бойова система, інвентар, збереження) представлений окремим набором скриптів та об’єктів Unity, що взаємодіють між собою через чітко визначені інтерфейси або дані.

3.3.1 Структура програмного забезпечення

Структура програмного забезпечення проєкту являє собою багаторівневу організацію компонентів, необхідних для функціонування прототипу інтерактивної гри з адаптивною поведінкою NPC. Архітектура побудована відповідно до принципів модульності, що забезпечує розмежування між базовим середовищем, ігровою логікою та документацією, яка супроводжує розробку.

Системне програмне забезпечення забезпечує базову інфраструктуру для виконання прикладного коду. У проєкті використано операційну систему Windows 11, інтегроване середовище розробки Unity 2022.3.44f1. Додатково застосовуються сторонні бібліотеки Unity Asset Store та редактор Visual Studio 2022, що забезпечує написання, налагодження та організацію коду гри.

Прикладне програмне забезпечення становить основу розробленої інформаційної системи. Складається з набору скриптів та модулів, що відповідають за функціональність окремих підсистем гри. Сюди входять:

- Модуль управління системою діалогів (DialogueManager, DialogueUI, NPCDialogueGraph),
- Бойова система (CombatManager, PlayerCombatController, NPCLogicController),
- Модуль управління поведінкою NPC (NPCStats, NPCInteraction),
- Управління ігровою логікою (GameDataManager, QuestManager, GameTimeController),
- Інтерфейс користувача (UIPanelManager, InventoryUI),
- Система збереження (SaveManager),
- та інші допоміжні компоненти.

Логіка кожного модуля інкапсульована, що дає змогу розвивати та модифікувати підсистеми незалежно одна від одної.

Програмна документація супроводжує створену гру та охоплює як технічні, так і користувацькі аспекти. До складу документації входить:

- Керівництво користувача, що описує основні механіки гри, принципи взаємодії з NPC, використання інтерфейсу, інвентаря та квестів.
- Інструкція з встановлення містить технічні вимоги та послідовність дій для запуску гри в середовищі Unity.
- Керівництво з тестування функціональності, з прикладами перевірки діалогової логіки, збереження гри та бойових сценаріїв.
- Структуру збережених даних формату JSON-файлів для NPC, гравця, інвентаря, а також правила формування ідентифікаторів внутрішньоігрових об’єктів.

Загальна схема структури ПЗ представлена у вигляді діаграми (*рисунок 3.2*), що демонструє взаємодію між зазначеними рівнями. Такий підхід дозволяє гнучко

підтримувати розробку, адаптувати систему під нові вимоги та забезпечити підтримку користувачів і тестувальників у процесі розгортання.

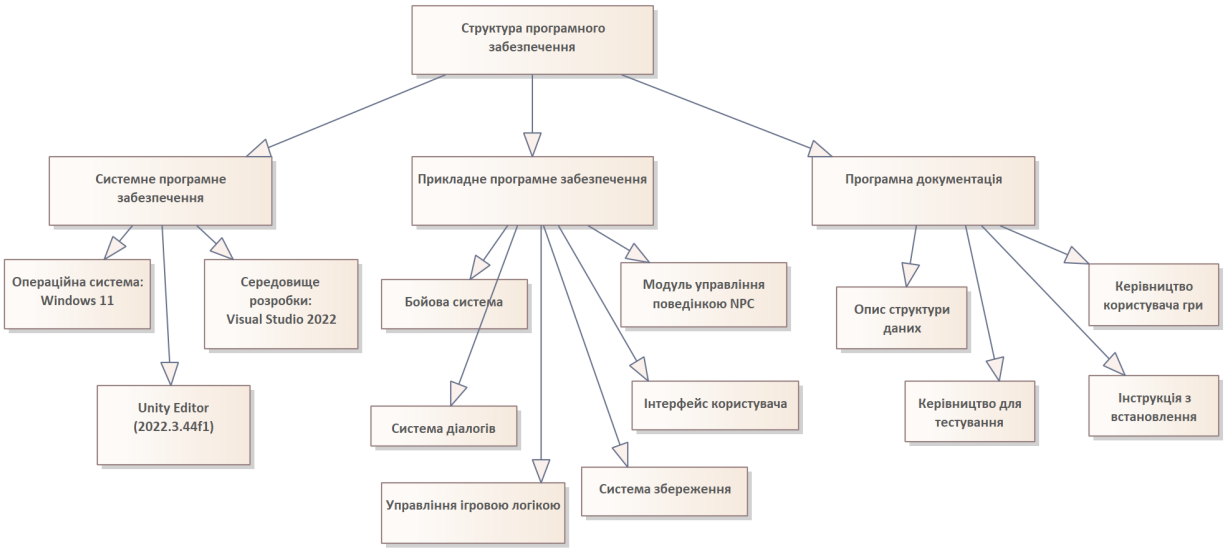


Рисунок 3.2 – Загальна схема структури програмного забезпечення

Джерело: сформовано автором на основі виконаного дослідження

3.3.2 Системне програмне забезпечення

Системне програмне забезпечення в рамках розробки інтерактивної гри з адаптивною поведінкою NPC забезпечує функціонування середовища розробки, інструментів програмування, а також взаємодію з апаратною частиною персонального комп’ютера. Його вибір базується на вимогах до продуктивності, сумісності з інструментами розробки, стабільності, а також підтримці необхідних API для роботи з тривимірною графікою, анімацією, ігровою фізикою тощо.

Основні компоненти системного ПЗ, які використовуються у проєкті, описані нижче.

Операційна система Windows 11. Сучасна платформа з підтримкою драйверів та інструментів, необхідних для стабільної роботи Unity та середовищ розробки. Вона забезпечує сумісність з інструментами візуального та інтегрованого програмування, підтримку систем шрифтів, файлових структур, візуалізації в реальному часі тощо.

Середовище розробки Visual Studio 2022, що надає розширене середовище для написання, налагодження та профілювання C#-коду, включаючи інструменти автодоповнення, аналізу помилок, інтеграцію з Unity через відповідний плагін.

Unity Editor (версія 2022.3.44f1): інтегроване середовище розробки ігор, що забезпечує редактор сцен, імпорт ассетів, роботу з анімаціями, візуалізацію 3D-графіки, створення UI, компонування сцен, а також побудову фінального білду гри. Unity також виконує роль рушія (game engine), що є серцем усього ігрового процесу.

Додатково застосовуються:

- Unity Asset Store як джерело сторонніх бібліотек, які використовуються для прискорення розробки (наприклад, анімаційні контролери, UI-компоненти).
- .NET Framework, що поставляється разом із Unity виконує роль інструментального середовища виконання для C#-коду та забезпечує доступ до стандартних бібліотек.

Враховуючи специфіку ігрової розробки, було обрано саме Unity як основний рушій, оскільки він забезпечує повну сумісність із сучасними системами рендерингу, має відкриту структуру роботи з анімаціями, об’єктами, станами і збереженням, а також дозволяє гнучко працювати з внутрішніми форматами даних через ScriptableObject та JSON.

3.3.3 Прикладне програмне забезпечення

Прикладне програмне забезпечення є центральним елементом реалізації гри з адаптивною поведінкою NPC. Воно забезпечує повноцінну ігрову логіку, збереження даних, виведення інтерфейсу, реакцію на дії гравця, а також динамічну зміну поведінки персонажів. Усі компоненти були реалізовані мовою програмування C# в інтегрованому середовищі Unity 2022.3.44f1, яке надає можливості швидкої інтеграції графіки, сценаріїв, анімацій та логіки в єдиному редакторі.

Вся логіка гри побудована на структурі окремих модулів, що відповідають за конкретні аспекти функціонування проєкту.

Система діалогів реалізує адаптивну комунікацію між гравцем і NPC. Компоненти DialogueManager, DialogueUI відповідають за керування перебігом діалогу, вибір реплік гравцем і виведення їх на екран. Діалоги побудовані на структурі NPCDialogueGraph (ScriptableObject), який містить вузли (DialogueNode) з текстами NPC, варіантами відповіді (DialogueOption) і можливими діями (наприклад, відкриття панелі магазину чи ініціація бою). Система підтримує умовні переходи залежно від поточного емоційного стану NPC.

Бойова система включає CombatManager, PlayerCombatController, NPCLogicController і GuardBattleController. Ці компоненти відповідають за ініціацію та перебіг бою, зміну анімацій, обчислення шкоди та контроль бойових станів NPC. У залежності від дій гравця NPC можуть змінювати свою поведінку: капітулювати, тікати чи переходити в агресивний режим.

Поведінкова система NPC реалізується через NPCStats, який зберігає поточні значення довіри, настрою, тривожності, здоров’я тощо. Ці параметри впливають на вибір реакції NPC в діалозі або поза ним. Додаткові компоненти NPCInteraction і NPCDetectionZone реалізують виявлення гравця в радіусі, зміну станів (наприклад, з Idle в Aggressive) та ініціювання відповідних дій, таких як вступ у бій чи втеча.

Інтерфейс користувача реалізується через окремі компоненти: UIPanelManager, InventoryUI, QuestUI, ShopUI, CombatUI. Вони відповідають за зручне відображення ігрової інформації: список предметів, виконані квести, репліки у діалозі, шкалу здоров’я під час бою тощо. Інтерфейс розроблено таким чином, щоб забезпечити інтуїтивну взаємодію без зайвих відволікань.

Глобальна логіка гри об'єднується у модулі GameDataManager, QuestManager і GameTimeController. Останній виконує ключову роль у моделюванні ігрового часу: зміна дня і ночі впливає на емоційний стан NPC, їхню доступність для взаємодії, а також на внутрішні скрипти поведінки. Наприклад, NPC можуть відмовити у взаємодії вночі.

Система збереження гри побудована на компоненті SaveManager, який формує та зчитує JSON-файли зі всіма ключовими даними гри: інформацією про гравця, NPC, квести, діалоги, інвентар та стан часу. Це дозволяє зберігати прогрес гри та відновлювати його після перезапуску.

Усі компоненти розроблені з дотриманням принципів модульності та інкапсуляції. Кожен модуль функціонує незалежно, має чітко визначені межі відповідальності й взаємодіє з іншими лише через стандартизовані інтерфейси або сервіси. Це дозволяє легко розширювати систему новими механіками, тестувати окремі модулі автономно та забезпечувати зручну підтримку під час експлуатації проєкту.

3.3.4 Програмна документація

Програмна документація супроводжує створену інформаційну систему та виконує роль інструктивного і довідкового матеріалу як для кінцевого користувача, так і для технічних спеціалістів. Вона забезпечує повноцінне ознайомлення з принципами роботи гри, підтримку її експлуатації, а також можливість подальшої модифікації.

До складу документації входить кілька ключових компонентів. Зокрема, керівництво користувача містить опис основних ігрових механік: взаємодії з неігровими персонажами, ведення діалогів, зміни станів NPC, використання інвентаря та виконання квестів. Такий документ дозволяє гравцю швидко зорієнтуватись у правилах гри та можливостях управління.

Інструкція з встановлення містить інформацію про мінімальні технічні вимоги для запуску гри, порядок встановлення середовища розробки (Unity), конфігурацію проєкту та послідовність дій для відкриття або компіляції проєкту на локальному пристрої.

Керівництво з тестування описує основні функціональні сценарії, що підлягають перевірці. Це включає запуск діалогів, обробку вибору реплік, зміни емоційного стану NPC, запуск бойових дій, а також збереження і завантаження

стану гри. Наведені приклади в *додатку А* дозволяють провести функціональне тестування з прогнозованими результатами.

Окремим елементом є опис структури збережених даних, зокрема JSON-файлів, які містять інформацію про стан NPC, гравця, інвентар, квести та часові параметри. Для кожного типу даних наведено приклади структури та правила формування унікальних ідентифікаторів об’єктів. Це дозволяє розробникам і тестувальникам впевнено працювати з даними гри, а також спрощує налагодження у разі збоїв.

Крім того, документація містить загальні відомості про призначення програми, області її застосування та обмеження щодо мінімальної конфігурації обладнання, необхідної для її стабільного функціонування.

Також надається опис архітектури системи, який містить стислий огляд модулів, таких як система діалогів, бойова логіка, взаємодія з NPC, збереження даних та інтерфейс користувача. Такий опис є важливим для підтримки проєкту, модернізації та перенесення до інших платформ або середовищ.

Документація підготовлена у форматах DOCX та PDF та розміщується у відповідному каталозі разом із проєктом. Її структура відповідає потребам як кінцевих користувачів гри, так і розробників, які підтримуватимуть або розвиватимуть систему надалі.

3.4 Результати реалізації гри

У результаті виконання бакалаврської кваліфікаційної роботи було створено функціональний прототип інтерактивної однокористувацької гри, в основі якої реалізована адаптивна поведінка неігрових персонажів (NPC). Розробка здійснювалася в середовищі Unity з використанням мови програмування C# та орієнтована на дослідження можливостей динамічного реагування NPC на дії гравця. Проєкт поєднує ряд оригінальних рішень у сфері діалогових систем, керування станами персонажів, логіки бою та взаємодії внутрішньоігрових підсистем.

Ключовим елементом є адаптивна система діалогів, яка формує відповіді NPC залежно від змінних параметрів — довіри до гравця, довіри до гільдії, настрою та тривожності. Ці характеристики змінюються в процесі гри під впливом дій гравця, результатів бою та сценарних подій. Це дозволяє досягти ефекту живого світу, де NPC реагують на ситуацію контекстуально, змінюючи свою поведінку та репліки.

У межах реалізації було створено також модулі:

- бойової системи, що підтримує комбінації атак та переходи між озброєними/неозброєними станами;
- поведінкової логіки NPC, яка включає зміну станів (нейтральний, агресивний тощо) залежно від параметрів довіри та здоров'я;
- системи керування ігровим часом (GameTimeController);
- системи збереження та відновлення прогресу у форматі JSON.

Архітектура програмного забезпечення реалізована за модульним підходом: окремі підсистеми ізольовані, що дозволяє підтримувати їх незалежно, масштабувати або замінювати. Усі збереження, конфігурації та дані NPC реалізовано у вигляді ScriptableObject-структур і зовнішніх JSON-файлів.

Розроблений прототип було апробовано на контрольному прикладі з участю кількох NPC (торговець, охоронець, лідер гільдії), з якими гравець міг взаємодіяти за допомогою адаптивних діалогів.

Проведено тестування таких аспектів гри:

- зміна поведінки NPC залежно від тривожності та довіри;
- діалогові гілки з fallback-варіантами;
- бойові сценарії;
- збереження і відновлення станів NPC;
- відкриття торгового чи квестового інтерфейсу за результатами діалогу.

Зібрані результати демонструють, що розроблена система дозволяє досягти цілей гнучкої поведінки NPC. Її потенційне використання може стосуватись ігор з

наративним ухилом, симуляцій соціальної взаємодії або інструментів для створення адаптивного контенту.

Розроблений прототип має широкі *перспективи подальшого вдосконалення* та масштабування. У межах майбутніх досліджень і розробок можуть бути реалізовані низка напрямів, що значно підвищать інтелектуальну складову системи, її адаптивність та зручність у розробці.

Одним із найперспективніших напрямів є інтеграція методів машинного навчання для моделювання поведінки NPC. Замість ручного налаштування правил і сценаріїв, поведінка персонажів може навчатися на основі ігрових ситуацій, дій гравця та статистики взаємодій. Це дозволить створити дійсно непередбачуваних та «живих» NPC, які адаптуються до стилю гри користувача.

Окрім цього, перспективною є реалізація реакцій NPC на глобальні події у грі, такі, як зміна погоди, політичні зміни у фракціях, великі битви чи інші сюжетні катаклізми. NPC зможуть обговорювати події між собою, змінювати настрій і плани залежно від загальної ситуації, що ще більше посилить ефект «живого світу».

Таким чином, розроблений прототип підтвердив свою практичну ефективність, сформував підґрунтя для подальших досліджень та продемонстрував реальність впровадження концепцій адаптивної взаємодії у внутрішньоігровому просторі.

ВИСНОВКИ

У ході виконання випускного бакалаврського проєкту реалізовано прототип інтерактивної гри, який досліджує концепцію адаптивної поведінки неігрових персонажів у змінному ігровому середовищі.

Основною метою проєкту було створення системи, здатної враховувати внутрішні параметри персонажів (такі, як довіра, настрій, тривожність) під час діалогів і бою, що успішно досягнуто шляхом побудови модульної архітектури та застосування об’єктно-орієнтованих підходів у розробці.

У процесі реалізації випускного бакалаврського проєкту було досягнуто низку важливих результатів, які підтверджують ефективність обраних технічних і методологічних рішень.

Було реалізовано адаптивну систему діалогів, що дозволяє NPC динамічно реагувати на дії гравця. Поведінка персонажів змінюється залежно від внутрішніх станів, зокрема довіри, настрою та тривожності. Система підтримує можливість дострокового завершення діалогу або автоматичного переходу NPC до бойового режиму, якщо репліка гравця викликає ворожу реакцію.

Було створено бойову систему, яка дозволяє NPC ініціювати бій не лише як відповідь на обрану гравцем репліку, а й автоматично у випадках, коли гравець виконує ряд провокуючих дій або здійснює напад на NPC. Такий підхід забезпечує нелінійність розвитку подій і сприяє глибшому зануренню у світ гри.

Також було здійснено чіткий поділ проєкту на такі логічні модулі: система діалогів, бойова система, система взаємодії з NPC, система збереження, управління квестами, інтерфейс користувача тощо. Така модульна архітектура забезпечує гнучкість і розширюваність. Підсистеми можуть розвиватися незалежно одна від одної без потреби в глобальних змінах коду.

У проєкті реалізовано серіалізацію даних у форматі JSON, що дало змогу ефективно зберігати й відновлювати стани NPC, гравця, інвентаря та інших об’єктів гри. Це рішення дозволило обійтись без реляційних баз даних, зберігаючи простоту та автономність системи.

Крім цього, була виконана побудова UML-моделей, що описують ключові аспекти структури та поведінки системи. Зокрема, створено діаграми класів, прецедентів, послідовностей і трасування вимог. Це забезпечило повноцінну програмну документацію, що відповідає сучасним вимогам до інженерії програмного забезпечення.

Нарешті, система була апробована на контрольному прикладі, що включає взаємодію гравця з кількома NPC. У результаті тестування підтверджено, що персонажі реагують відповідно до закладеної логіки, змінюють свою поведінку залежно від попередньої взаємодії, вибору гравця та власного емоційного стану. Це засвідчує працездатність системи та її відповідність поставленим вимогам.

Серед ключових переваг розробленого програмного забезпечення слід виділити його автономність. Система функціонує повністю локально, без потреби в постійному інтернет-з’єднанні або зверненні до зовнішніх серверів. Це забезпечує стабільну роботу гри в ізольованому середовищі, підвищує безпеку збережених даних та робить проєкт доступним для використання на пристроях із обмеженим або відсутнім доступом до мережі.

Ще однією важливою перевагою є гнучкість архітектури. Система легко адаптується до впровадження нових сценаріїв, поведінкових моделей NPC або геймплейних механік. Завдяки використанню параметризованих об’єктів та умовної логіки, поведінка NPC може бути розширена без необхідності переробки вже існуючого функціоналу.

Окремо слід відзначити модульність розробленої системи. Логічний поділ функціональності на незалежні компоненти дозволяє ефективно масштабувати гру, замінювати або оновлювати окремі частини без втручання в інші. Такий підхід полегшує супровід проєкту, спрощує тестування та створює основу для подальшого розвитку системи.

Серед перспектив розвитку проєкту варто виділити впровадження методів машинного навчання для формування реакцій NPC на основі накопиченого досвіду взаємодії з гравцем. Такий підхід дозволить системі адаптуватися до стилю гри

конкретного користувача, поступово змінюючи поведінкові моделі залежно від ігрового контексту.

Також до перспективних напрямів належить розширення реакцій NPC на глобальні зміни у світі гри, що дозволить реалізувати глибше занурення користувача в ігровий процес. Зміна поведінки NPC відповідно до подій, які відбуваються поза межами поточного діалогу чи сцени, сприятиме створенню більш динамічного і живого ігрового світу.

Особистий внесок студентки полягає в розробці ключових функціональних модулів гри, а саме системи діалогів, логіки бойових дій, поведінки NPC та збереження стану. Уся реалізація велась з нуля без використання сторонніх фреймворків для ігрової логіки, з опорою на власне розуміння принципів адаптивності в ігровому дизайні. Також студентка виконала моделювання структури системи та створила технічну документацію. Робота має прикладне значення як навчальний або дослідницький проєкт, який може бути використаний для демонстрації адаптивної поведінки в інтерактивних системах. Водночас проєкт поки не призначений для використання як готовий комерційний продукт. Для цього потрібна додаткова розробка, розширення контенту та тестування.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alonso E. et al. *Deep Reinforcement Learning for NPC Navigation*. arXiv, 2020. URL: https://arxiv.org/abs/2011.04764 (дата звернення: 11.06.2025)
2. Bates J. *The Role of Emotion in Believable Agents*. 1994. URL: https://www.somearchive.org/bates_emotion_believable_agents.pdf (дата звернення: 11.06.2025)
3. Cavazza P., Charles F., Mead S. *Interactive Storytelling in Games*. 2002. URL: https://www.igi-global.com/gateway/book/interactive-storytelling-games/ (дата звернення: 11.06.2025)
4. Černý M. et al. *Behavior Objects in Virtual Worlds*. arXiv, 2015. URL: https://arxiv.org/abs/1508.00377 (дата звернення: 11.06.2025)
5. Code With K. *Scriptable Objects in Unity – a powerful tool for data management*. Medium, 2021. URL: https://medium.com/@Code_With_K/scriptable-objects-in-unity-a-powerful-tool-for-data-management-... (дата звернення: 11.06.2025)
6. Deductor International. *Deductor – Data Mining Suite*. URL: https://www.deducer.org/ (дата звернення: 11.06.2025)
7. Dignum V. *Trust and Emotion in Agent-Based Systems*. Springer, 2018. URL: https://link.springer.com/chapter/10.1007/978-3-319-67522-5_4 (дата звернення: 11.06.2025)
8. GameDevBeginner. *Dialogue Systems in Unity*. 2022. URL: https://gamedevbeginner.com/dialogue-systems-in-unity/ (дата звернення: 11.06.2025)
9. Improved Non-Player Character behavior using evolutionary approaches. ScienceDirect, 2024. URL: https://www.sciencedirect.com/science/article/abs/pii/S187595212400243X (дата звернення: 11.06.2025)
10. Intelligent Adaptation of Difficulty and NPC Behavior in Serious Games. ScienceDirect, 2024. URL:

https://www.sciencedirect.com/science/article/pii/S2405896324002313 (дата звернення: 11.06.2025)

11. Mateas M., Stern A. *Façade: An Experiment in Building a Fully-Realized Interactive Drama*. 2003. URL: http://www.interactivestory.net/publications/façade/ (дата звернення: 11.06.2025)
12. MineSet Team. *MineSet Visual Data Mining Software*. URL: https://www.altair.com/mineset/ (дата звернення: 11.06.2025)
13. Nakatsu R. et al. *Evaluating Believable Characters in Conversational Agents*. 2000. URL: https://dl.acm.org/doi/10.1145/336939.336946 (дата звернення: 11.06.2025)
14. Partlan N. et al. *EvolvingBehavior: Co-Creative Evolution of NPCs*. arXiv, 2022. URL: https://arxiv.org/abs/2209.01020 (дата звернення: 11.06.2025)
15. PixelCrushers. *Dialogue System for Unity: Scripting*. 2023. URL: https://www.pixelcrushers.com/dialogue_system/manual2x/html/scripting.html (дата звернення: 11.06.2025)
16. Reynold C. W. *Flocks, Herds, and Schools: A Distributed Behavioral Model*. SIGGRAPH, 1987. URL: https://dl.acm.org/doi/10.1145/37401.37406 (дата звернення: 11.06.2025)
17. Schwab A., Bohle C. *Emotionally Sensitive Interactions in Interactive Narratives*. ACM, 2019. URL: https://dl.acm.org/doi/10.1145/3359985 (дата звернення: 11.06.2025)
18. Sestini A. et al. *Deep Policy Networks for NPC Behaviors*. arXiv, 2020. URL: https://arxiv.org/abs/2012.03532 (дата звернення: 11.06.2025)
19. Short E. *Mood, Anxiety and Trust in NPC Decision Making*. arXiv, 2020. URL: https://arxiv.org/abs/2004.01234 (дата звернення: 11.06.2025)
20. SSRN. *Adaptive NPC in Serious Games Using Artificial Intelligence*. 2023. URL: https://papers.ssrn.com/abstract=4806061 (дата звернення: 11.06.2025)
21. Suzue K. *Adaptive NPC Behavior In Maze Chase Game Using Genetic Algorithms*. Wooster Independent Study, 2022. URL:

https://openworks.wooster.edu/independentstudy/9972/ (дата звернення: 11.06.2025)

22. Unity Technologies. *ScriptableObject*. Unity Manual, 2023. URL: https://docs.unity3d.com/Manual/class-ScriptableObject.html (дата звернення: 11.06.2025)
23. Unity Technologies. *Separate Game Data and Logic with ScriptableObjects*. Unity Learn, 2023. URL: https://unity.com/how-to/separate-game-data-logic-scriptable-objects (дата звернення: 11.06.2025)
24. Wikipedia. *Action selection*. URL: https://en.wikipedia.org/wiki/Action_selection (дата звернення: 11.06.2025)
25. Wikipedia. *Agent-based model*. URL: https://en.wikipedia.org/wiki/Agent-based_model (дата звернення: 11.06.2025)
26. Wikipedia. *Artificial intelligence in video games*. URL: https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games (дата звернення: 11.06.2025)
27. Wikipedia. *Behavior tree (artificial intelligence)*. URL: https://en.wikipedia.org/wiki/Behavior_tree_(artificial_intelligence) (дата звернення: 11.06.2025)
28. Wikipedia. *Non-player character*. URL: https://en.wikipedia.org/wiki/Non-player_character (дата звернення: 11.06.2025)
29. Wired. *Wild Things: AI in Games* (2002). URL: https://www.wired.com/2002/06/wild-things-ai/ (дата звернення: 11.06.2025)
30. Yannakakis G. N. *Game AI Revisited*. ACM Computing Frontiers, 2012. URL: https://dl.acm.org/doi/10.1145/2212908.2212923 (дата звернення: 11.06.2025)

ДОДАТКИ

Додаток А

Керівництво з тестування

Керівництво з тестування містить перелік функціональних сценаріїв, які перевіряють коректність реалізації основних механік гри. Для кожного сценарію наведено:

1. Унікальний ідентифікатор тесту (Test ID);
2. назву тесту;
3. передумови (preconditions);
4. вхідні дії (inputs);
5. очікуваний результат (expected result);
6. фактичний результат (заповнюється вручну при тестуванні);
7. статус проходження (Pass/Fail).

Таблиця А.1 – Приклад заповнення звіту про тестування

Test ID:	T03
Назва:	Адаптивна реакція NPC на агресивну репліку за умови низької довіри та високої тривожності.
Передумови:	npcType = civil, trustToPlayer = 20, anxiety = 80, відкритий діалог
Вхідні дії:	Вибір загрозливої репліки в діалозі
Очікуваний результат:	NPC біжить в точку «безпечне місце»
Фактичний результат:	NPC залишився спокійним, діалог завершився
Статус:	Fail
Коментар:	У вузлі діалогу не спрацювала перевірка на пороги. Можливо, помилка у FallbackLogic.

Нижче представлена таблиця функціональних тестів (таблиця А.2).

Додаток А

Таблиця А.2 – Таблиця функціональних тестів

ID	Назва тесту	Передумови	Вхідні дії	Очікуваний результат
T01	Ініціація діалогу	Гравець знаходиться в радіусі дії NPC, NPC не зайнятий боєм	Натиснути E при наведеному «полі зору» на NPC	Відкривається панель діалогу, показано перший вузол репліки
T02	Вибір репліки	Відкритий діалог	Натиснути на одну з запропонованих кнопок з текстом	Відображається відповідна реакція NPC, зміщується поточний вузол
T03	Адаптивна реакція NPC на агресивну репліку за умови низької довіри та високої тривожності.	npcType = civil, trustToPlayer = 20, anxiety = 80, відкритий діалог	Вибір загрозливої репліки в діалозі	NPC біжить в точку «безпечне місце»
T04	Відмова від діалогу при низькій довірі	trustToPlayer < 1	Спробувати ініціювати діалог	Виводиться fallback-репліка або NPC відмовляється говорити
T05	Зміна параметрів NPC	Діалог із NPC, обрана репліка має вплив на trustToPlayer, mood, anxiety	Завершити діалог, перевірити нові значення trustToPlayer, mood, anxiety	Значення параметрів NPC змінено згідно з логікою вузла
T06	Збереження стану NPC	Встановлено значення NPCStats, SaveManager активний	Завершити взаємодію, зберегти гру	У npc_stats.json оновлено актуальні значення

Продовження таблиці А.2.

Додаток А

ID	Назва тесту	Передумови	Вхідні дії	Очікуваний результат
T07	Завантаження стану NPC	Існує npc_stats.json із записом для поточного NPC	Запустити гру, підійти до NPC	Значення параметрів завантажено, діалог відкривається відповідно до них
T08	Ініціація бою з NPC за умови низької довіри і високої тривожності	NPC бойового типу, trustToPlayer <=10, anxiety >= 80	Підійти в радіус дії NPC та дістати зброю (не атакувати)	NPC реагує агресивно, переходить у стан бою, запускається анімація атаки
T09	NPC просить про помилування	Здоров’я NPC < 20%	Гравець атакує NPC	NPC не атакує у відповідь, з’являється панель діалогу з вузлом SurrenderNode
T10	Перемикання зброї	Гравець в режимі з не активованою зброєю	Натиснути клавішу «1»	Персонаж виймає зброю

Джерело: сформовано автором на основі виконаного дослідження

Додаток Б

StrikePlagiarism.com

Дата звіту 6/15/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kyiv National Economic University named after Vadym Hetman KNEU
Заголовок
Проєктування інтерактивної комп'ютерної гри з адаптивною поведінкою NPC
Автор Науковий керівник / Експерт
Мужа Надія АнатоліївнаШевченко К.Л.
підрозділ
кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

2.52% 2.52% КП 1

0.82% 0.82% КП 2

0.44% 0.44% КЦ

25
Довжина фрази для коефіцієнта подібності 2

13490
Кількість слів

103231
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	5
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	23

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/f2d2da61-14eb-4ba9-b82f-2663cc9ae485/download	67 0.50 %
2	https://ir.kneu.edu.ua/bitstreams/b20c497c-320d-46ce-8d1d-a56f3c2eec3f/download	43 0.32 %
3	https://ir.kneu.edu.ua/bitstreams/b20c497c-320d-46ce-8d1d-a56f3c2eec3f/download	24 0.18 %
4	https://ir.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	24 0.18 %
5	https://ir.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	20 0.15 %