Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка та оптимізація мобільного стратегічного додатку із використанням ШІ та алгоритмів прийняття рішень

Виконали: студенти VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Філіпович Т.І.
(підпис) (прізвище та ініціали)

_____ Швець О.Я.
(підпис) (прізвище та ініціали)

Керівник _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М.
(підпис) (прізвище та ініціали)

_____ Никитюк В.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Філіпович Тетяни Іванівної
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та оптимізація мобільного стратегічного додатку із використанням ШІ та алгоритмів прийняття рішень

Керівник роботи Небесний Руслан Михайлович, Ph.D., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 29 » листопада 2024 року № 4/7-1139

2. Термін подання студентом завершеної роботи 29 травня 2025 р.

3. Вихідні дані до роботи Наукові публікації з інтеграції штучного інтелекту в ігрові системи, веб-ресурси з оглядами архітектур діалогових моделей, документація до мовних моделей, інструменти для симуляції ігрових світів.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Дослідження передумов застосування генеративного ші та алгоритмів прийняття рішень у мобільних іграх. 2 Аналіз засобів та методів інтегрування штучного інтелекту у ігрові застосунки. 3 Розробка мобільного застосунку з інтеграцією штучного інтелекту та аналізу прийняття рішень. 4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульний слайд**.** 2 Тема, мета, об'єкт і предмет дослідження. 3 Завдання дослідження. 4 Актуальність теми. 5 Історичний контекст розвитку мобільного геймінгу. 6 Алгоритми прийняття рішень у відеоіграх. 7 Методи інтеграції ШІ в ігрові застосунки. 8 Проблеми та виклики інтеграції ШІ у мобільні ігри. 9 Архітектура розробленого мобільного додатку. 10 Логіка побудови ігрового світу та поведінки NPC. 11 Інтерфейс та візуальне оформлення. 12 Послідовність розробки і структура проєкту. 13 Оптимізація продуктивності та тестування. 14 Охорона праці та безпека в надзвичайних ситуаціях. 15 Висновки магістерської роботи. 16 Подяка / Завершальний слайд

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 13 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.11.2024	*Виконано*
2.	Підбір та опрацювання наукових публікацій, збір	02.12.2024-10.01.2025	*Виконано*
	даних по темі роботи		
3.	Виконання дослідження згідно теми кваліфікаційної	13.01.2025-14.03.2025	*Виконано*
	роботи		
4.	Оформлення розділу «Дослідження передумов	17.03.2025-28.03.2025	*Виконано*
	застосування генеративного ШІ та Алгоритмів		
	прийняття рішень у мобільних іграх»		
5.	Оформлення розділу «Аналіз засобів та методів	31.03.2025-11.04.2025	*Виконано*
	інтегрування штучного інтелекту у ігрові застосунки»		
6.	Оформлення розділу «Розробка мобілього застосунку	14.04.2025-25.04.2025	*Виконано*
	з інтерграцією штучного інтелекту та аналізу		
	прийняття рішень»		
7.	Виконання завдання до підрозділу «Охорона праці»	28.04.2025-02.05.2025	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в	28.04.2025-02.05.2025	*Виконано*
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	05.05.2025-09.05.2025	*Виконано*
10.	Нормоконтроль	12.05.2025-15.05.2025	*Виконано*
11.	Перевірка на плагіат	16.05.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	19.05.2025	*Виконано*
13.	Захист кваліфікаційної роботи	29.05.2025	

Студент _____ Філіпович Т.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Швецю Олександру Ярославовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та оптимізація мобільного стратегічного додатку із використанням ШІ та алгоритмів прийняття рішень

Керівник роботи Небесний Руслан Михайлович, Ph.D., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 29 » листопада 2024 року № 4/7-1139

2. Термін подання студентом завершеної роботи 29 травня 2025 р.

3. Вихідні дані до роботи Наукові публікації з інтеграції штучного інтелекту в ігрові системи, веб-ресурси з оглядами архітектур діалогових моделей, документація до мовних моделей, інструменти для симуляції ігрових світів.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Дослідження передумов застосування генеративного ші та алгоритмів прийняття рішень у мобільних іграх. 2 Аналіз засобів та методів інтегрування штучного інтелекту у ігрові застосунки. 3 Розробка мобільного застосунку з інтеграцією штучного інтелекту та аналізу прийняття рішень. 4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульний слайд. 2 Тема, мета, об'єкт і предмет дослідження. 3 Завдання дослідження. 4 Актуальність теми. 5 Історичний контекст розвитку мобільного геймінгу. 6 Алгоритми прийняття рішень у відеоіграх. 7 Методи інтеграції ШІ в ігрові застосунки. 8 Проблеми та виклики інтеграції ШІ у мобільні ігри. 9 Архітектура розробленого мобільного додатку. 10 Логіка побудови ігрового світу та поведінки NPC. 11 Інтерфейс та візуальне оформлення. 12 Послідовність розробки і структура проєкту. 13 Оптимізація продуктивності та тестування. 14 Охорона праці та безпека в надзвичайних ситуаціях. 15 Висновки магістерської роботи. 16 Подяка / Завершальний слайд

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 13 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.11.2024	*Виконано*
2.	Підбір та опрацювання наукових публікацій, збір	02.12.2024-10.01.2025	*Виконано*
	даних по темі роботи		
3.	Виконання дослідження згідно теми кваліфікаційної	13.01.2025-14.03.2025	*Виконано*
	роботи		
4.	Оформлення розділу «Дослідження передумов	17.03.2025-28.03.2025	*Виконано*
	застосування генеративного ШІ та Алгоритмів		
	прийняття рішень у мобільних іграх»		
5.	Оформлення розділу «Аналіз засобів та методів	31.03.2025-11.04.2025	*Виконано*
	інтегрування штучного інтелекту у ігрові застосунки»		
6.	Оформлення розділу «Розробка мобілього застосунку	14.04.2025-25.04.2025	*Виконано*
	з інтерграцією штучного інтелекту та аналізу		
	прийняття рішень»		
7.	Виконання завдання до підрозділу «Охорона праці»	28.04.2025-02.05.2025	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в	28.04.2025-02.05.2025	*Виконано*
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	05.05.2025-09.05.2025	*Виконано*
10.	Нормоконтроль	12.05.2025-15.05.2025	*Виконано*
11.	Перевірка на плагіат	16.05.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	19.05.2025	*Виконано*
13.	Захист кваліфікаційної роботи	29.05.2025	

Студент _____ Швець О.Я.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка та оптимізація мобільного ігрового додатку із використанням ШІ та алгоритмів прийняття рішень // Кваліфікаційна робота освітнього рівня «Магістр» // Філіпович Тетяна Іванівна // Швець Олександр Ярославович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2025 // С. 142, рис. – 24 , табл. – 4 , слайлів. – 16 , додат. – 2, бібліогр. – 128 .

Ключові слова: мобільний застосунок, генеративний ШІ, динамічний сюжет, мовна модель, оптимізація продуктивності, архітектура EDA, GPT, NPC.

Кваліфікаційна робота присвячена дослідженню, проектуванню та реалізації мобільного ігрового застосунку з підтримкою генеративного штучного інтелекту, що забезпечує динамічну побудову сюжетів та діалогів у режимі реального часу. Проєкт розроблено як MVP стратегічної інтерактивної новели, у якій ігровий світ реагує на дії користувача, враховуючи обрану модель поведінки героя, внутрішній стан NPC та поточну ситуацію у грі.

У першому розділі здійснено аналіз історичного розвитку мобільних ігор, розкрито роль ШІ в сучасній гейм-індустрії, а також представлено результати економічного дослідження впливу AI-рішень на ефективність розробки, структуру витрат та зайнятість у сфері геймдеву.

У другому розділі проаналізовано сучасні наукові підходи до інтеграції LLM у діалогові системи, методи процедурної генерації ігрового контенту, адаптивної поведінки NPC та сценарного контролю через FSM, behavior trees, multi-agent models. Також розглянуто технічні та методологічні виклики впровадження ШІ у мобільні платформи.

У третьому розділі описано архітектуру та реалізацію прототипу гри на базі Unity з інтеграцією GPT API, включно з генерацією реплік, оновленням стану персонажів, обробкою контексту та адаптивною логікою побудови діалогів.

Проведено оптимізацію системи під обмеження мобільних пристроїв і тестування продуктивності.

Об’єктом дослідження є мобільна інтерактивна ігрова система з генеративною побудовою контенту.

Предметом дослідження виступають методи інтеграції генеративного ШІ у мобільні додатки з нелінійною сюжетною структурою.

ANNOTATION

Development and optimization of a mobile game application using AI and decision-making algorithms // The educational level "Master" qualification work // Filipovych Tetiana Ivanivna // Shvets Oleksandr Yaroslavovych // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2025 // P. 142 , fig. – 24 , tables – 4 , posters – 16 , annexes – 2 , ref. – 128 .

Keywords: mobile application, generative AI, dynamic narrative, interactive novel, language model, game simulation, performance optimization, EDA architecture, GPT, NPC.

This master’s thesis explores the development and implementation of a mobile game application featuring generative artificial intelligence for real-time narrative and dialogue generation. The project presents an MVP of a strategic interactive novel, in which the game world evolves based on the user's actions, selected character behavior model, NPC attitudes, and the dynamic state of the game environment.

The first chapter analyzes the historical evolution of mobile gaming and highlights the growing role of AI technologies in the modern game industry. It also presents an economic study on the impact of AI integration on development efficiency, cost structures, and employment trends within game development.

The second chapter reviews current scientific approaches to integrating large language models (LLMs) in dialogue systems, procedural generation techniques for narrative content, and adaptive NPC behavior modeling using FSMs, behavior trees, and multi-agent systems. The technical and methodological challenges of applying AI in mobile platforms are also addressed.

The third chapter describes the architecture and implementation of the game prototype developed in Unity with GPT API integration. It includes dialogue generation, context processing, dynamic character state updates, and scenario control logic. System optimization for mobile resource constraints and performance testing are also covered.

The object of research is a mobile interactive game system with generative content architecture.

The subject of research is the methodology of integrating generative AI into mobile applications with nonlinear story progression.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Геймінг – сфера ігрової індустрії, що охоплює створення, розробку та використання відеоігор.

ЗМІ – Засоби масової інформації.

Інтерактивний сюжет – сюжет, розвиток якого залежить від вибору користувача в процесі гри.

ПЗ – Програмне забезпечення.

Промпти – вхідні запити (інструкції), які надсилаються мовній моделі для генерації результату.

Токени – мінімальні одиниці тексту, що використовуються в процесі обробки та генерації мовними моделями.

ШІ – Штучний інтелект.

ААА-ігри (англ. Triple-A Games) – високобюджетні ігрові проєкти з високим рівнем продакшну.

AI (англ. Artificial Intelligence) – штучний інтелект, галузь комп’ютерної науки, що займається створенням інтелектуальних систем.

API (англ. Application Programming Interface) – набір визначень взаємодії різнотипного ПЗ.

GPT – Generative Pre-trained Transformer – велика мовна модель, здатна генерувати текст на основі контексту.

JSON (англ. JavaScript Object Notation) – формат обміну даними у вигляді структурованого тексту.

LLM (англ. Large Language Model) – велика мовна модель, натренована на великих обсягах текстових даних.

ML (англ. Machine Learning) – машинне навчання, розділ ШІ, що вивчає методи навчання комп’ютерів на основі даних.

MVC (англ. Model-View-Controller) – архітектурний шаблон «модель-представлення-контролер» для розробки ПЗ.

NLP (англ. Natural Language Processing) – обробка природної мови, галузь ШІ, що фокусується на взаємодії між комп’ютерами та людською мовою.

NPC (англ. Non-Playable Character) – ігровий персонаж, який не керується гравцем, а діє згідно з закладеною логікою.

ScriptableObject – клас Unity, що дозволяє створювати незалежні від сцени конфігураційні об'єкти.

Unity – ігровий рушій для створення інтерактивних 2D/3D-додатків.

ЗМІСТ

ВСТУП

Актуальність теми. Інтеграція штучного інтелекту (ШІ) у мобільні застосунки є одним із ключових напрямів сучасної програмної інженерії. Особливу актуальність ця тенденція має у сфері ігрової індустрії, де технології генеративного ШІ активно використовуються для створення персоналізованого контенту, адаптивної поведінки персонажів та динамічного нелінійного сюжету. В умовах зростання очікувань користувачів щодо глибини ігрового досвіду, традиційні фіксовані сценарії поступаються місцем інтерактивним іграм із багатошаровим наративом, живими діалогами та світом, що реагує на дії гравця.

Наслідком таких змін є необхідність у розробці нових архітектурних рішень, здатних забезпечити повноцінну генерацію сюжетного наповнення на основі моделей ШІ з дотриманням вимог до продуктивності, що характерні для мобільних пристроїв. У цьому контексті поєднання Unity, API для генеративного ШІ, системи керування станом персонажів і адаптивної побудови реплік формує фундамент для створення якісно нових ігрових продуктів.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи є дослідження архітектурних, технічних та алгоритмічних засад побудови мобільного інтерактивного ігрового додатку з генеративним ШІ та реалізація прототипу гри, в якому діалоги і сюжет створюються динамічно на основі дій користувача.

Задачі дослідження включають:

Метою даної кваліфікаційної роботи є дослідження архітектурних, алгоритмічних і технічних засад побудови мобільного інтерактивного ігрового додатку з генеративним ШІ, а також реалізація MVP-прототипу гри, в якій діалоги та сюжет динамічно формуються на основі вибору та поведінки користувача.

Задачі дослідження:

– дослідити існуючі моделі генерації діалогів та сценаріїв на основі ШІ (Філіпович Т. І.);

– проаналізувати архітектури інтерактивних ігор із нелінійною структурою сюжету (Філіпович Т. І.);

– побудувати логіку ігрового світу та механізми адаптивної поведінки NPC (Філіпович Т. І.);

– розробити архітектуру мобільного застосунку з інтеграцією GPT-моделі (Швець О. Я.);

– реалізувати серверну й клієнтську частини додатку на основі Unity (Швець О. Я.);

– провести оптимізацію продуктивності та адаптацію GPT для мобільного середовища (Швець О. Я.);

– здійснити комплексне тестування якості діалогів, швидкодії, навантаження системи та користувацького досвіду (спільно).

Об’єкт дослідження є інтерактивна система мобільної гри з генеративною побудовою діалогів і нелінійним сюжетом, реалізована з використанням ШІ.

Предмет дослідження є методи та інструменти створення адаптивних сюжетних сценаріїв і персоналізованих діалогів у мобільних іграх на базі генеративних мовних моделей та алгоритмів поведінки.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у створенні модульної архітектури мобільного додатку, яка реалізує адаптивну побудову сюжетних сценаріїв з використанням генеративного ШІ.

Розроблена система враховує зміну характеру гравця, ставлення NPC та поточний стан ігрового середовища, забезпечуючи нелінійний і унікальний хід подій.

Практичне значення одержаних результатів. Результати роботи можуть бути використані при створенні інтерактивних або сюжетно-орієнтованих ігор, навчальних симуляторів та віртуальних середовищ, де потрібна персоналізована та динамічна взаємодія з користувачем. При необхідному фінансуванні та апаратних можливостях серверної частини, може бути фундаментом для повної генерації віртуальної реальності.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на VII Міжнародній науково-

практичній конференції молодих учених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2025)» Київського політехнічного інституту імені Ігоря Сікорського (м. Київ, 2025 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає сторінки, з них сторінки основного тексту, який містить рисунків та таблиць.

1 ДОСЛІДЖЕННЯ ПЕРЕДУМОВ ЗАСТОСУВАННЯ ГЕНЕРАТИВНОГО ШІ ТА АЛГОРИТМІВ ПРИЙНЯТТЯ РІШЕНЬ У МОБІЛЬНИХ ІГРАХ

1.1 Історичний контекст та еволюція ігрових застосунків на мобільних пристроях

Перші мобільні ігри з’явилися ще наприкінці 1990-х років. Знаковим моментом стала поява гри «Snake» у 1997 році, встановленої на телефонах «Nokia», що продемонструвала потенціал мобільних пристроїв як платформи для ігор.

У 1999 році в Японії сервіс «I-mode» вперше дозволив завантажувати на телефони більш складні ігри, хоча спочатку ця технологія була обмежена територією Японії. На початку 2000-х технічні можливості мобільних телефонів на заході також зросли до рівня, достатнього для підтримки завантажуваних додатків, включно з іграми, проте широкому розповсюдженню заважала фрагментація платформ та відсутність єдиного магазину додатків.

Кардинальна революція у сфері мобільних ігор сталася з виходом смартфона iPhone від компанії Apple у 2007 році та запуском магазину App Store у 2008-му. Єдиний централізований магазин додатків спростив процес поширення ігор і відкрив дорогу незалежним розробникам до масової аудиторії. Протягом перших років більшість мобільних ігор розповсюджувалися як платні продукти. Однак у жовтні 2009 року компанія Apple запровадила вбудовані покупки, що дало змогу перейти до моделей де гравці могли отримувати ігрову валюту за перегляд реклами, дана модель використовується і у сьогоденні.

На початку 2010-х з’явилася нова модель монетизації мобільних ігор, яка принесла рекордні прибутки. Ігри на кшталт Candy Crush Saga та Puzzle & Dragons запровадили механіку з обмеженнями по часу та можливістю докуповувати додаткові спроби через внутрішньоігрові покупки. Ці продукти започаткували епоху масових умовно безкоштовних ігор з мільйонною аудиторією.

Багато з найуспішніших мобільних проєктів сьогодні залучають сотні мільйонів гравців та генерують щорічний дохід понад 100 млн доларів, а окремі продукти перевищують планку в 1 млрд доларів на рік.

Поступово мобільні ігри охопили широкий спектр жанрів: від гіперказуальних (наприклад, Crossy Road) до інноваційних локаційних AR-ігор (таких як Pokémon Go), демонструючи здатність мобільної платформи адаптуватися під різні ігрові концепції.

Сьогодні мобільний геймінг перетворився на ключовий сегмент індустрії відеоігор. Завдяки стрімкому поширенню смартфонів та мобільного інтернету, ігри на телефонах стали доступними майже кожному. Мобільні ігри вже генерують понад половину загальносвітового доходу ігрової індустрії.

До прикладу, на рисунку 1.1 – зображено діаграму фінансових прибутків ігрової галузі у 2022 році, обсяг споживчих витрат на мобільні ігри сягнув $103,5 млрд, що становило ~53% від загального ринку відеоігор.

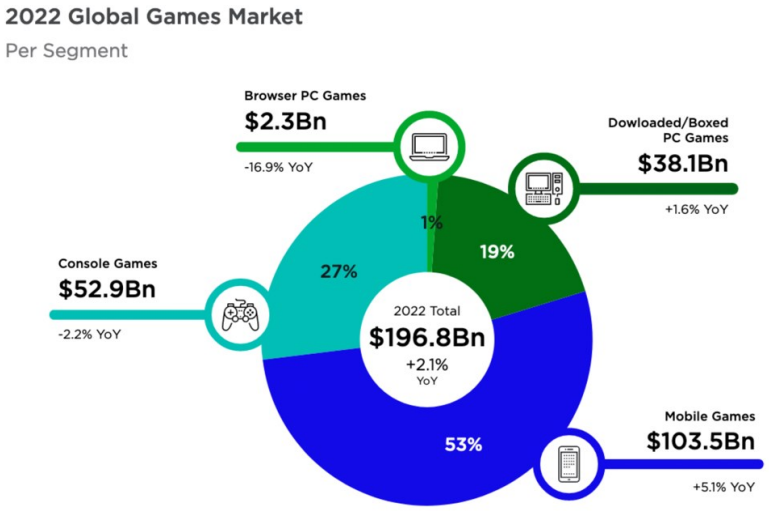

Рисунок 1.1 – Частка виручки різних сегментів ігрового ринку у 2022 році за даними Newzoo.

Така домінуюча позиція мобільних платформ зумовлена величезною аудиторією гравців (понад 3 млрд осіб) і зручністю доступу до ігор. Історична еволюція від простих Snake до складних багатокористувацьких мобільних ігор показала, що мобільні пристрої стали повноцінними ігровими платформами, потенціал яких продовжує зростати.

1.2 Роль штучного інтелекту та його вплив на розвиток ігрової індустрії

Штучний інтелект відіграє важливу роль у розвитку ігрової індустрії з самого початку появи відеоігор. Історично під ігровим ШІ розуміли передусім алгоритми керування поведінкою неігрових персонажів (NPC) та супротивників – від найпростіших шаблонів руху привидів у Pac-Man (Див. рисунок 1.2) до складних систем прийняття рішень у стратегіях реального часу. Проте з розвитком технологій можливості ШІ в іграх значно розширилися.

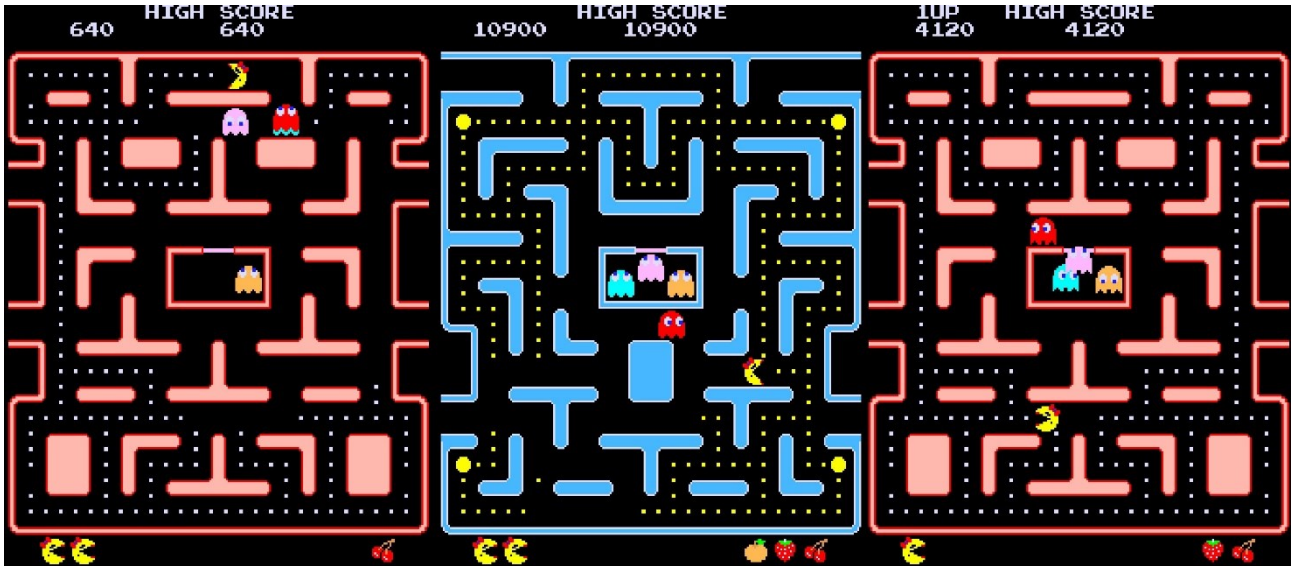

Рисунок 1.2 – Перші спроби використання ШІ на прикладі гри «Pac-Man»

Сьогодні ігровий ШІ охоплює різні аспекти геймдеву, до прикладу: поведінка NPC, процедурна генерація контенту (PCG), пошук і планування, моделювання гравця, асистування розробникам тощо.

Інтеграція ШІ дозволяє робити ігри розумнішими та більш інтерактивними. Персонажі починають вчитися і адаптуватися до дій гравця, світи реагують на вибір користувача, а контент може генеруватися динамічно. Сучасні ігрові компанії все активніше впроваджують інновації на основі ШІ, що суттєво впливає на розвиток індустрії.

По-перше, ШІ підвищує якість ігрового процесу – NPC із підтримкою штучного інтелекту можуть демонструвати правдоподібну поведінку, що робить

взаємодію з ними більш захопливою та непередбачуваною. Наприклад, AI-керовані NPC здатні запам’ятовувати дії гравця, навчатися на їхній основі та формувати власні цілі й мотивації, завдяки чому ігрові сюжети стають глибшими та більш нелінійними.

По-друге, ШІ сприяє персоналізації: з допомогою аналізу даних про стиль гри кожного користувача можливо автоматично підлаштовувати рівень складності, пропонувати індивідуальні завдання або рекомендації, що підвищує залученість гравців.

По-третє, методи штучного інтелекту застосовуються у розробці такого контенту як – генерація рівнів, карт, сюжетів або діалогів за допомогою алгоритмів дозволяє суттєво зменшити витрати часу та праці розробників. Наприклад, процедурна генерація рівнів за допомогою нейромереж може забезпечити різноманіття ігрового досвіду при кожному новому проходженні гри. Таку технологію використовують здебільшого ігри жанру платформери та раннери, де необхідно нескінченно генерувати нові рівні та локації.

На рисунку 1.3 зображено як саме генерується локація та рівні у популярних іграх Subway Surfers та Flappy bird.

Рисунок 1.3 – Приклад генерації рівнів та карт за допомогою нейромережі у мобільних ігрових застосунках.

З точки зору бізнесу, вплив ШІ на індустрію теж надзвичайно відчутний. За останні роки сформувався окремий сегмент ринку – рішення на базі ШІ для ігор. За даними галузевих досліджень, обсяг світового ринку ШІ у геймінгу становив приблизно $1,2 млрд у 2022 році, а до 2028 року прогнозується його

зростання до $7,1 млрд. Так , на рисунку 1.4 зображено графік обсяг ринку штучного інтелекту з 2023 до прогнозів на 2034, це без врахування потенційних наукових відкритів у даній галузі.

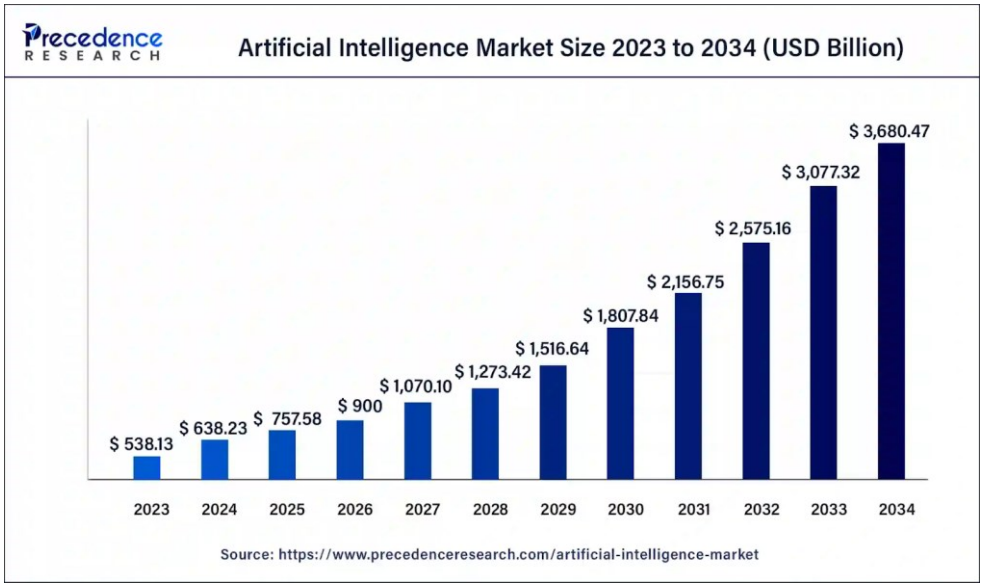

Рисунок 1.4 – Графік обсяг ринку штучного інтелекту з 2023 до 2034

Фактично, застосування ШІ поступово переходить з розряду експериментів у розряд необхідності для конкурентоспроможності. Компанії, що ігнорують ШІ, ризикують відстати від ринку, а у гіршому випадку не витримавши конкуренції збанкротувати. Натомість студії, які активно впроваджують ШІ, отримують переваги на всіх етапах – від розробки до монетизації продукту.

Штучний інтелект вже зараз забезпечує автоматизацію рутинних процесів (наприклад, тестування гри шляхом використання ботів), аналітику поведінки гравців для прийняття бізнес-рішень, а також створює можливості для появи принципово нових жанрів ігрового досвіду.

Отже, роль ШІ в ігровій індустрії є двоаспектною: з одного боку, це збагачення та ускладнення ігрового процесу, з іншого – оптимізація процесу створення і експлуатації ігор.

1.3 Економічний вплив використання ШІ

Штучний інтелект активно змінює економічні підходи у сфері розробки відеоігор, впливаючи на собівартість виробництва, розподіл ресурсів і формування прибутку. Актуальним є аналіз того, як впровадження ШІ впливає на ефективність геймдев-процесів, конкурентне середовище та структуру зайнятості.

1.3.1 Витрати на розробку та ефективність

Однією з головних обіцянок ШІ в контексті економіки є скорочення витрат та часу розробки за рахунок автоматизації. Дійсно, інструменти на базі ШІ дозволяють суттєво підвищити продуктивність команди. За даними опитувань, близько 40% студій, що впровадили ШІ, відзначають зростання продуктивності більш ніж на 20%

На рисунку 1.5 представлено ринкові тенденції у сфері застосування штучного інтелекту в ігровій індустрії на період 2023–2031 років.

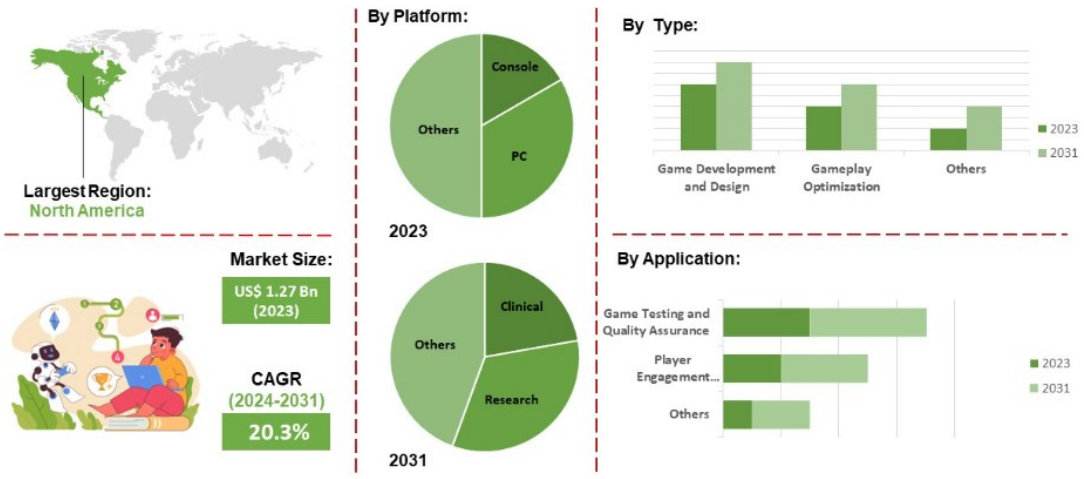

Рисунок 1.5 – Прогноз розвитку ринку штучного інтелекту в геймдеві на період 2023–2031 рр

У розрахунку це ~10-15-разове підвищення ефективності за часом і затратами. Звісно, отримані ШІ-зображення потребували доопрацювання художником, але загалом одна людина за тиждень підготувала дизайн 17

персонажів, тоді як раніше на це пішло б понад місяць за участю цілої команди. Порівняльну характеристику наглядно показує таблиця 1.1.

Таблиця 1.1 Порівняння витрат на концепт-арт з ШІ та без ШІ (за даними Lost Lore Studio)

Показник	**Традиційний підхід**	**Підхід із ШІ**
Кількість персонажів	100 (концепти)	100 (концепти)
Людські ресурси	~5-7 художників	1 дизайнер (+ШІ)
Час на створення	~6 місяців	~1 місяць
Прямі витрати	~$50 000	~$10 000
Економія часу	–	~6 разів швидше
Економія коштів	–	~80% дешевше

Як видно з таблиці, генеративні інструменти ШІ здатні радикально знизити трудомісткість творчих етапів. Подібні результати відзначають й інші студії: ШІ допомагає митцям швидко створювати варіації дизайнів, прототипи рівнів, а програмістам – генерувати шаблонний код або знаходити помилки, економлячи дні, а то і тижні роботи.

Проте, очікування тотального здешевлення не завжди виправдовуються. Багато керівників ігрових компаній вважають, що ШІ більше сприятиме якості та швидкості випуску ігор, ніж прямому зниженню бюджету. Лише близько 20% опитаних топ-менеджерів впевнені, що генеративний ШІ зменшить витрати розробки. Інші ж зазначають, що заощаджені на контенті кошти можуть перерозподілятися на інші статті – наприклад, розширення самої гри чи маркетинг.

Сучасні ААА-проекти вже досягають бюджетів у сотні мільйонів доларів, подекуди до $1 млрд, тож навіть автоматизація частини процесів може не скоротити загальні витрати, адже студії водночас підвищують планку якості та масштабу. Іншими словами, ШІ радше дозволяє зробити більше за ті ж кошти, ніж просто зробити те саме дешевше.

Експерти застерігають, що поспішне впровадження ШІ без продуманої стратегії може знизити ефективність інвестицій. Компаніям слід чітко оцінювати, де саме ШІ дасть максимальну віддачу – чи варто розробляти власне рішення, чи краще інтегрувати стороннє, чи, можливо, обмежитися традиційними методами.

1.3.2 Ринок праці та зайнятість

Автоматизація, яку несе з собою ШІ, має двоякий вплив на зайнятість у ігровій індустрії. З одного боку, зменшення потреби в ручній рутинній праці потенційно означає скорочення деяких позицій (наприклад, частини тестувальників, молодших художників чи дизайнерів рівнів). З іншого – з'являється попит на нові ролі, такі як спеціалісти з ШІ, інженери даних, технічні художники, які вміють ефективно використовувати генеративні моделі.

Останніми роками ігрова індустрія переживає значні скорочення персоналу. Тільки в 2023 році по світу було втрачено понад 10 тисяч робочих місць у геймдеві, і тренд продовжився в 2024 (Див. рисунок 1.6).

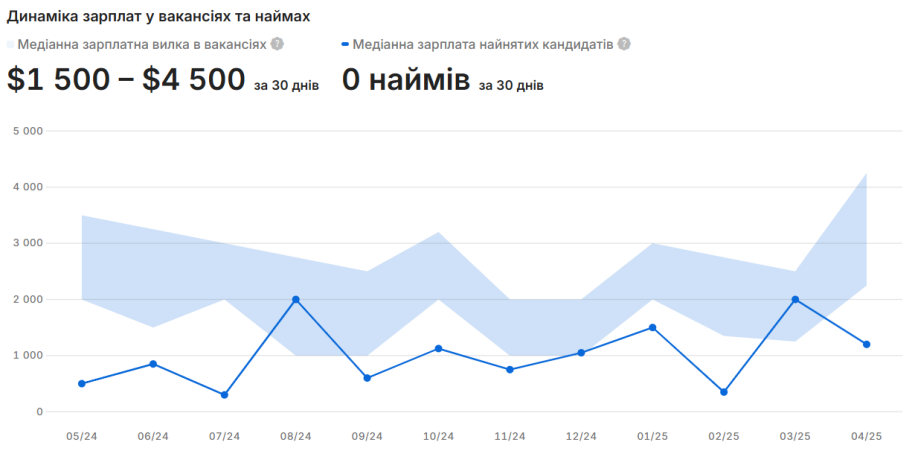

Рисунок 1.6 Динаміка медіанної зарплати у вакансіях та фактичних наймах за період травень 2024 – квітень 2025 рр.

Причини різні – від економічних спадів та невдалих бізнес-рішень до реорганізацій після злиттів. Водночас, технологічний прогрес (включно з ШІ)

називають серед факторів, що дозволяють компаніям «робити більше з меншим штатом», тобто стають опосередкованою причиною скорочень.

ШІ та автоматизація замінюють деякі завдання, які традиційно виконувалися людьми. Наприклад, генерація озвучення і анімації: сучасні моделі можуть синтезувати голоси і навіть дублювати манеру мови реальних акторів. Це ставить під удар професію акторів озвучення та захоплення руху – компанії отримують інструмент для створення контенту без участі людей.

В багатьох країнах починають формуватися профспілки гейм-розробників і тестувальників, які серед іншого вимагають гарантій захисту від необґрунтованих звільнень через автоматизацію. У 2023 р. тестувальники Activision Blizzard успішно добилися права на утворення профспілки, що стало одним із перших прецедентів в ігровій сфері (Див. рисунок 1.7). Фактично, індустрія переживає ті ж виклики, що й інші галузі в епоху індустріалзації, де необхідно перекваліфіковувати персонал і шукати моделі співіснування людей і машин.

Рисунок 1.7 – Масовий протест працівників Blizzard Entertainment у 2023 році

Водночас, думки лідерів галузі щодо масштабів впливу ШІ на зайнятість розходяться. Опитування керівників компаній показує, що більшість не очікує радикального скорочення потреби в талантах через ШІ – 60% вважають, що технологія не вирішить проблему дефіциту кваліфікованих кадрів. ШІ може

виконувати рутинні завдання, але «креативна іскра» та управлінські рішення лишатимуться за людьми.

Навіть за наявності найсучасніших генеративних моделей більші студії, навпаки, наймають все більше розробників, зокрема фахівців для створення і керування цими ж моделями. Наприклад, ігрова студія CD Projekt Red у 2023 році оголошували, що хоч і впроваджують деякі AI-рішення, але продовжують розширювати штат, бо попит на контент зростає.

Отже, економія на зарплатах за рахунок ШІ можлива, але не безконтрольна. Найбільш імовірний сценарій – трансформація ролей: замість повного зникнення професій відбувається зміна кваліфікацій. Наприклад, художник стає куратором ШІ-генерації і виконує більше ролі арт-директора, контролюючи якість згенерованого, а не малюючи з нуля. Це підтверджують висновки студій: «*ШІ не замінює художника, але художник, що вміє працювати з ШІ, замінить того, хто не вміє*». Компанії, що першими впровадять співпрацю людини і ШІ, здатні вирватися вперед, тоді як інші будуть змушені наздоганяти.

1.3.3 Вплив на ринок ігор та монетизацію

З точки зору ширшої економіки індустрії, використання ШІ впливає на конкурентний ландшафт та моделі заробітку.

По-перше, бар’єр входу для малих команд знижується. Раніше високополігональні 3D-моделі, великі відкриті світи або кінематографічні заставки були прерогативою великих студій з відповідними бюджетами. Тепер же інді-розробники можуть скористатися інструментами ШІ для генерування високоякісних текстур, моделей, анімацій, не витрачаючи десятки людино-місяців роботи. Це означає, що конкуренція посилюється: малі студії завдяки ШІ можуть конкурувати з гігантами у креативності та навіть графічному рівні, принаймні у менш масштабних проектах. Вже є приклади інді-ігор, де один розробник створює контент, який раніше вимагав би команди – використовуючи нейромережі для малюнків і музики. В результаті ринок може отримати більше

ігор, більш різноманітних, що добре для споживачів, але ставить виклик перед великими видавцями.

По-друге, монетизація ігрових послуг стає більш персоналізованою за допомогою ШІ. Аналіз даних гравців дає змогу краще сегментувати аудиторію і пропонувати кожному релевантний платний контент. Наприклад, якщо ШІ виявить, що певний гравець тяжіє до косметичних предметів певного стилю, система може йому адресно запропонувати схожі скіни чи акції.

З одного боку, це підвищує ймовірність покупки (а отже, прибутки розробника), з іншого – викликає питання етики та приватності даних. Необережне поводження з даними гравців може призвести до порушення конфіденційності. Регуляторні органи уважно стежать, щоб алгоритми монетизації не перетворювалися на експлуатацію вразливостей. Тому компанії впроваджують ШІ-аналитику монетизації поступово.

Економічно важливо зберігати чесну гру та дружню спільноту, щоб не втрачати гравців. ШІ вже сьогодні допомагає Riot Games та Ubisoft боротися з токсичною поведінкою онлайн – компанії спільно розробляють ШІ-системи, що автоматично фільтрують образи в чаті та карають порушників. Це опосередковано впливає на утримання гравців (а значить і доходи). Так само, античіт на базі ШІ зберігає чесну змагальність, що критично для кіберспортивних дисциплін з мільйонними призовими фондами.

По-третє, нові продукти та сервіси. Поява ШІ в іграх відкриває можливості для нових джерел доходу. Наприклад, AI Dungeon – гра, повністю побудована на текстовому генеративному ШІ, демонструє модель підписки на «ігрового AI-майстра», який створює пригоди на льоту.

Великі компанії можуть випускати DLC, де гравець отримує власного AI-компаньйона (віртуального помічника чи навіть друга) за додаткову плату. Уявімо MMORPG, де за певну суму гравець «наймає» собі розумного союзника-NPC, що підтримує його стиль гри – це цілком реальний сценарій майбутньої монетизації. Звісно, якість такого сервісу має бути високою, щоб виправдати витрати користувача.

Серед викликів – юридична невизначеність щодо ШІ-контенту. Якщо гра генерує, скажімо, картинку, схожу на чужий арт (на якому модель навчалась), можливі претензії про порушення авторських прав. Поки судова практика тільки формується, видавці побоюються потенційних позовів. Це може гальмувати впровадження деяких генеративних технологій до появи чітких правил.

Інший ризик – репутаційний. Негативна реакція спільноти гравців на «зловживання ШІ». Наприклад, якщо гра випущена сирою, з багатьма ШІ-згенерованими елементами низької якості, гравці можуть сприйняти це як халтуру і відвернутися від продукту. В суспільстві існують побоювання щодо ШІ, тому важливо комунікувати прозоро. Більшість студій підкреслюють, що використовують ШІ відповідально, для підсилення творчості, а не для скорочення кутаїв на якості.

Загалом, економічний вплив ШІ на індустрію ігор є позитивним у контексті ефективності та зростання ринку, але потребує зваженого підходу. Компанії, які інвестують у ШІ, можуть отримати конкурентну перевагу – швидший випуск якісних ігор, нижчі витрати, вищу залученість гравців (а значить і доходи).

Однак, для максимізації вигоди слід враховувати соціальні наслідки, підтримувати баланс між автоматизацією та людською працею і дотримуватися етичних норм.

1.4 Еволюція алгоритмів прийняття рішень у відеоіграх

Одним із найбільш значущих чинників, що визначають якість ігрового досвіду у відеоіграх, є система прийняття рішень неігровими персонажами (NPC). Від найпростіших умовних конструкцій до складних моделей машинного навчання — індустрія геймдеву пройшла вражаючий шлях розвитку. Ця еволюція стала відповіддю на постійно зростаючі очікування гравців щодо правдоподібності, гнучкості та варіативності ігрового середовища.

На ранніх етапах розвитку відеоігор використовувалися rule-based системи — жорстко запрограмовані алгоритми, що керували поведінкою NPC на основі простих умов типу «якщо-умова-то-дія».

Прикладом є гра Pac-Man, де привиди рухаються за наперед визначеними траєкторіями. Такий підхід був ефективним за обмежених ресурсів, однак не допускав адаптивності.

Із удосконаленням апаратного забезпечення поширилися скінченні автомати (Finite State Machines, FSM), які дозволяли NPC перемикатися між заздалегідь заданими станами залежно від контексту. Поведінка стала більш структурованою персонажі могли патрулювати, атакувати чи ухилятися, змінюючи тактику відповідно до дій гравця. Цей підхід залишався домінантним упродовж тривалого часу, хоча й був схильним до надмірної передбачуваності.

Далі індустрія зробила крок у бік більш складних і гнучких рішень: були запроваджені поведінкові дерева (Behavior Trees). На відміну від FSM, ці структури мали ієрархічну природу та дозволяли будувати багаторівневу логіку з можливістю повторного використання вузлів поведінки. Вони набули широкого застосування в іграх жанру stealth та екшн, дозволяючи персонажам діяти контекстно та з певною автономністю.

Справжнім викликом став перехід до алгоритмів, що вчаться — зокрема, до навчання з підкріпленням (Reinforcement Learning, RL). Ці методи дозволяють агентам самостійно знаходити ефективні стратегії поведінки у складному середовищі на основі винагороди. Незважаючи на значні успіхи в дослідницьких і демонстраційних проектах, широке впровадження RL обмежується його високими вимогами до обчислювальних ресурсів та тривалістю навчання.

Останній етап розвитку — це поява генеративних підходів, які зміщують фокус із передбачуваної поведінки на контекстно залежну та індивідуалізовану взаємодію. Хоча детальний розгляд великих мовних моделей винесено в окремий розділ, тут варто зазначити, що сучасні ігри все частіше поєднують попередні архітектури з генеративними інструментами, щоб створити унікальний наративний досвід.

Таким чином, розвиток алгоритмів прийняття рішень у відеоіграх — це послідовна еволюція від передбачуваності до адаптивності, від жорстко прописаних сценаріїв до моделей, що здатні враховувати дії гравця, історію взаємодій та загальний контекст гри.

Кожен етап відіграв важливу роль:

Rule-based логіка забезпечила стартову ефективність,

FSM — структурованість,

BT — масштабованість,

RL — навчання на досвіді

Сучасні генеративні системи — персоналізацію і гнучкість.

Цей поступальний розвиток створив підґрунтя для побудови ігрових світів, здатних реагувати на гравця не просто логікою, а — умовно — емоційною інтуїцією, що і визначає обличчя сучасного геймдизайну.

1.5 Висновок до першого розділу

У першому розділі магістерської роботи було закладено теоретичну й аналітичну основу для подальшого проектування мобільного ігрового додатку з генеративним штучним інтелектом. Було досліджено історичний розвиток мобільного геймінгу, з’ясовано причини його домінування на ринку відеоігор, а також проаналізовано ключові тенденції в еволюції алгоритмів прийняття рішень у відеоіграх — від жорстко запрограмованих rule-based моделей до адаптивних і генеративних архітектур.

Особливу увагу приділено сучасному впливу штучного інтелекту на якість ігрового досвіду, інструменти розробки та економіку геймдеву. Показано, що інтеграція ШІ у виробничі процеси дозволяє підвищити ефективність, оптимізувати витрати, скоротити цикл створення контенту та відкрити нові можливості для персоналізованої взаємодії. Водночас зафіксовано низку економічних, технічних і соціальних викликів — від обмежень мобільних пристроїв до впливу автоматизації на ринок праці.

Також розкрито важливість архітектурних моделей прийняття рішень для NPC та побудови динамічних сюжетів, що мають критичне значення у створенні правдоподібного ігрового світу.

2. АНАЛІЗ ЗАСОБІВ ТА МЕТОДІВ ІНТЕГРУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ У ІГРОВІ ЗАСТОСУНКИ

2.1 Аналіз існуючих наукових досліджень та підходів інтеграції ШІ в ігрові застосунки

Останні роки ознаменувалися стрімким зростанням інтересу до Game AI як у практичній розробці, так і в наукових дослідженнях. Чимало уваги зосереджено на інтеграції сучасних підходів штучного інтелекту в ігрові системи — зокрема для створення адаптивної поведінки персонажів, генерації діалогів та моделювання сюжетів. У контексті цієї роботи особливий інтерес становлять саме ті методи, які дають змогу динамічно керувати розвитком ігрового світу та взаємодією з NPC. Надалі буде розглянуто найбільш релевантні підходи, які мають практичне значення для реалізації поставленої ідеї.

Однією з провідних тенденцій останніх років є використання великих мовних моделей – LLM, для керування діалогами персонажів та навіть для моделювання їх поведінки.

Великі мовні моделі (LLM) — це сучасні системи штучного інтелекту, розроблені для аналізу, розуміння та генерації тексту, що нагадує людське мовлення. Вони побудовані на методах глибокого навчання і навчені великими обсягами даних, що часто включають мільярди слів із різноманітних джерел, таких як веб-сайти, книги та статті.

Дослідники пропонують розглядати сучасні мовні моделі як ядро для інтелекту NPC, здатне генерувати змістовні репліки та реакції в діалозі. Наприклад, у роботі Park et al. (2023) описано прототип соціальної гри Slice of Life, де поєднано класичну симуляцію соціальної взаємодії з генерацією діалогів за допомогою мовної моделі – LLM відповідає за живе генерування реплік NPC на основі їх стану та контексту.

Для забезпечення правдоподібності таких діалогових систем дослідники часто комбінують нейромережеві моделі з семантичними або логічними структурами знань. Зокрема, у роботі Hardiman et al. (2024) запропоновано підхід

генерації сюжетних квестів та діалогів у рольових іграх на основі поєднання language model + knowledge graph велика мовна модель створює текстові варіанти діалогів, а граф знань слугує каркасом, що забезпечує узгодженість сценарію та врахування дій гравця.

Такий гібридний підхід дозволяє долати проблему «галюцинацій» та безконтрольного тексту від мовних моделей, накладаючи необхідні обмеження, аби діалоги залишалися в межах ігрового сюжету і стилю.

Важливою областю досліджень є також процедурна генерація ігрового світу із застосуванням сучасних алгоритмів ШІ.

Традиційно процедурний контент генерувався за допомогою випадкових або евристичних алгоритмів. Нині ж, зі зростанням популярності генеративного ШІ, дослідники експериментують із використанням нейромереж для створення більш складного і різноманітного контенту.

У літературі відзначено, що глибокі нейронні мережі можуть успішно генерувати найрізноманітніший ігровий контент – від геометрії рівнів і ландшафтів до поведінки персонажів та елементів сюжету.

Наприклад, нейронні моделі на основі генеративних змагальних мереж GAN чи автокодерів здатні навчитися на прикладах існуючих рівнів і створювати нові, які відповідають стилю гри.

Окремий перспективний напрям – використання нейронних клітинних автоматів NCA, для поступової побудови складних структур: дерев, ландшафтів, будівель у ігровому світі. Прикладом може слугувати експериментальна модель, що навчається генерувати зображення дерев шляхом ітеративного росту пікселів, подібно до розростання клітин у природі (цей підхід досліджували, зокрема, Mordvintsev et al., 2020).

Генеративні моделі також застосовуються для створення цілих інтерактивних світів. Нещодавно запропоновано концепцію навчання моделі Genie на відеоданих, яка здатна генерувати ігровий рівень (платформер) за заданим зображенням-настроєм, тим самим автоматично проектуючи ігровий світ під певний стиль. Таким чином, сучасні дослідження у сфері PCG все більше спираються на досягнення глибинного навчання.

Важливо відзначити, що одним із викликів у застосуванні нейромережевих генеративних методів для ігор є дефіцит навчальних даних. У оглядовій роботі Ямада та ін. (2023) підкреслено, що високоякісні генеративні моделі потребують великих обсягів даних для тренування, але ігровий контент часто дуже специфічний і варіативний, тож зібрати достатній датасет непросто.

Дослідники пропонують різні шляхи вирішення цієї проблеми – від використання transfer learning перенавчання моделей, попередньо натренованих на інших даних, до генерації синтетичних даних та навчання з підкріпленням у симуляціях.

Окрема гілка наукових праць присвячена забезпеченню узгодженості та цілісності згенерованого контенту. Наприклад, щоб уникнути ситуацій, коли процедурно згенерований рівень містить непрохідні ділянки чи нежиттєздатні комбінації елементів, в алгоритми вводять спеціальні обмеження або застосовують післягенераційний аналіз валідності контенту.

Ще одним напрямом досліджень на перетині ШІ та геймдизайну є моделювання гравців і адаптивний геймплей. Застосування методів машинного навчання для аналізу дій гравця дозволяє будувати моделі, що передбачають наступні кроки користувача або класифікують його ігровий стиль. Такі моделі можуть використовуватися для динамічної зміни складності гри або підбору персоналізованих завдань.

Наприклад, роботи Нгуєна і Пател (2022) демонструють систему, яка в реальному часі підлаштовує рівень виклику у грі під рівень вмілості гравця, щоб підтримувати стан захопленості і не допустити нудьги чи фрустрації.

Інші дослідження фокусуються на пошукових алгоритмах і навчанні з підкріпленням для керування персонажами: сучасні агенти на основі RL навчилися на рівні людини грати у складні ігри (шахи, StarCraft II тощо), а в контексті комерційних ігор RL використовується для створення «розумніших» ботів або навіть для балансування ігрових механік (наприклад, підбір оптимальних параметрів зброї через еволюційний алгоритм, що мінімізує перевагу тієї чи іншої тактики).

У підсумку можна стверджувати, що інтеграція ШІ в ігрові застосунки є багатогранною темою. Найбільш прогресивні підходи зосереджені на генеративних можливостях ШІ – мовних моделях для діалогів і сюжетів, нейронних генераторах для контенту – а також на адаптивних алгоритмах, які підлаштовують ігровий досвід під користувача.

Вже зараз існують демонстраційні проєкти і прототипи ігор, де ШІ фактично виконує роль «ігрового режисера», контролюючи розвиток сюжету та поведінку персонажів у режимі реального часу. Проте для практичного використання цих досягнень у масових комерційних продуктах необхідно подолати низку технічних та методологічних проблем, про які йтиметься далі.

2.2 Виклики та обмеження сучасних систем та мобільних пристроїв

Незважаючи на значний прогрес, впровадження сучасних алгоритмів ШІ (особливо глибинного навчання) в ігрові застосунки пов’язане з рядом серйозних викликів. Ці обмеження зумовлені як технічними факторами апаратних платформ (зокрема, мобільних пристроїв), так і природою самих моделей штучного інтелекту.

Передусім варто відзначити обмеження продуктивності та ресурсів на сучасних пристроях. Мобільні телефони та планшети, на відміну від потужних серверів або ПК, мають суттєво меншу обчислювальну потужність, обсяг пам’яті та обмежений запас енергії для виконання інтенсивних задач.

Як наслідок, запуск складних моделей машинного навчання безпосередньо на пристрої викликає труднощі. Розробники змушені або оптимізувати моделі шляхом компресії, квантизації параметрів, використання менш вимогливих архітектур, або переносити частину обчислень на сервер.

Втім, перенесення обчислень у хмару має свої недоліки збільшується затримка в обміні даними, виникає залежність від якості інтернет-з’єднання, зростають ризики для приватності даних користувача. Таким чином, розробка мобільних ігор із AI-компонентами вимагає балансу між використанням можливостей пристрою та хмарних сервісів.

Друга група проблем пов’язана із самими глибинними моделями та їх інтеграцією в ігровий процес. Сучасні нейронні мережі часто є обчислювально ємними, поведінку яких важко передбачити та пояснити. Це створює ряд ризиків при застосуванні їх у іграх в реальному часі.

По-перше, продуктивність у реальному часі, виконання великих моделей може уповільнювати гру – знижувати частоту кадрів і викликати підвисання, якщо не вжито заходів оптимізації.

Особливо критично це для динамічних ігор та мобільних пристроїв зі слабшим апаратним забезпеченням.

По-друге, надійність та передбачуваність. Нейромережевий NPC може раптово виконати нелогічну або небажану дію, яку розробники не передбачали, адже модель навчена на статистичних закономірностях і не гарантує дотримання дизайнерського задуму.

Подібна непередбачуваність здатна порушити ігровий баланс або навіть зламати сюжет. Підтримання наративного контролю над історією гри стає складнішим, коли діалоги генеруються ШІ в режимі онлайн. Розробникам доводиться впроваджувати додаткові механізми обмеження – наприклад, системи правил, що перевіряють відповіді моделі, або «контекстні фільтри», які спрямовують AI у русло сюжетної лінії.

Ще один аспект – відлагодження та тестування ігор з інтегрованим ШІ. У традиційній розробці ігор поведінка NPC прописується вручну, тож розробник приблизно знає, до яких наслідків призведе та чи інша дія гравця.

У випадку навченої нейромережі передбачити всі можливі виходи складно, тому процес тестування ускладнюється: потрібно програвати безліч сценаріїв, щоб виявити потенційно некоректні реакції AI.

Налагодження таких систем теж є викликом, адже відсутній прозорий ланцюжок «умова-відповідь» – важко ізолювати причину небажаної поведінки, коли модель оперує великими масивами розмитих параметрів. Це змушує розробників створювати спеціальні інструменти моніторингу та дебагу для AI-компонентів або спрощувати моделі, жертвуючи їх глибиною заради керованості.

Суттєвим обмеженням є також вартість і складність впровадження передових AI-рішень. Тренування сучасних глибоких моделей – особливо для діалогів чи генерації контенту – потребує великих обсягів даних і дорогих обчислювальних ресурсів (GPU/TPU ферми). Для невеликих студій та звичайних аматорів-ідейників це може бути непідйомним завданням.

Навіть використання вже готових моделей від сторонніх постачальників (наприклад, API для мовних моделей) тягне за собою фінансові витрати, які зростають із кількістю активних гравців. Окрім того, виникають питання масштабованості чи зможе ШІ-система обслуговувати мільйони користувачів одночасно без деградації якості?

Не менш важливим є аспект приватності та безпеки даних. Інтелектуальні системи часто потребують дані про поведінку гравця для навчання або адаптації – наприклад, збирають статистику дій, успіхів/невдач, виборів у діалогах.

Забезпечити конфіденційність цієї інформації та відповідність вимогам регуляторів (GDPR тощо) – окремий виклик. При використанні хмарних ШІ-сервісів доводиться передавати дані користувача на зовнішні сервери, що несе ризики витоку або несанкціонованого доступу. Тому розробники мусять впроваджувати заходи шифрування, анонімізації даних, чітко прописувати політики приватності, що ускладнює розробку і підвищує її вартість.

Підсумовуючи, сучасні системи та мобільні пристрої накладають такі ключові обмеження на повномасштабну інтеграцію ШІ в ігри:

- Недостатня потужність мобільних пристроїв для роботи великих моделей в реальному часі, необхідність оптимізації або використання хмари.
- Потреба відправляти дані в на хмару для обрахунку ШІ збільшує зависання і витрачає заряд батареї, що критично для мобільного досвіду.
- Глибокі моделі можуть генерувати небажаний контент, важко контролюються і відлагоджуються.
- Необхідність у великих датасетах для навчання, проблема отримання якісних даних, особливо з урахуванням приватності гравців.
- Дорогий процес навчання і підтримки ШІ, потреба у фахівцях з машинного навчання, масштабованість серверів під навантаження.

– Забезпечення цілісності гри необхідність зберігати баланс і сюжет при додаванні автономних ШІ-агентів, уникати ламання ігрової логіки їх неконтрольованими діями.

Ці виклики визначають напрямки подальших досліджень і розробок. Вирішення окреслених проблем є обов'язковою умовою для успішного створення ігрових застосунків, де ШІ зможе повністю керувати діалогом та розвитком ігрового світу без шкоди для користувацького досвіду.

2.3 Методи та рівні інтеграції ШІ в ігрову індустрію

Методи інтеграції ШІ охоплюють широкий спектр інструментів – від класичних алгоритмів ухвалення рішень для NPC до сучасних нейромереж, здатних генерувати контент. Рівень інтеграції може варіюватися від обмеженого використання ШІ для окремих задач до глибокої інтеграції, коли ШІ керує значною частиною ігрового процесу або контенту. Нижче розглянуто основні напрямки застосування ШІ у іграх та ступінь їх впровадження.

2.3.1 Процедурна генерація контенту

Одним із ключових застосувань ШІ є процедурна генерація ігрового контенту – автоматичне створення рівнів, ландшафтів, завдань, персонажів та інших елементів без прямого ручного моделювання. Це дозволяє суттєво збільшити масштаб і різноманітність ігрового світу.

Наприклад, гра No Man's Sky генерує всесвіт із понад 18 квінтильйонів унікальних планет за допомогою алгоритмів, що випадково комбінують параметри ландшафту, екосистеми та фауни (Рисунок 2.1). Процедурна генерація економить час розробників і робить кожну сесію гри унікальною, підвищуючи рівень занурення гравця.

Крім рівнів та карт, генеративні алгоритми застосовуються для створення сюжетних подій та квестів. Генеративний ШІ здатний автоматично створювати

контент на основі наявних даних, прискорюючи розробку ігор та зменшуючи рутинне навантаження на дизайнерів.

Рисунок 2.1 – Приклади генерації ланшафтів у грі No Man’s Sky

Важливо, що розробники розглядають такий ШІ як інструмент доповнення творчого процесу, а не повної заміни людини. Алгоритми використовуються для чорнової генерації ідей (локацій, сюжетних зав’язок, дизайну предметів), тоді як фінальне шліфування та креативні рішення залишаються за людьми.

Це узгоджується з думкою експертів, що в найближчі 5–10 років ШІ може перебрати до половини завдань зі створення ігор – передусім на етапах планування контенту та автоматизації рутини, але не витіснить людський фактор у питаннях творчості.

2.3.2 Розумні NPC та поведінкова адаптація

Нейромережі та навчання дедалі активніше застосовуються для керування поведінкою неігрових персонажів. Сучасний ШІ дозволяє NPC аналізувати оточення, реагувати на дії гравця та динамічно змінювати свою поведінку у відповідь на ігрові події. Завдяки методам глибокого навчання Deep Learning та навчання з підкріпленням Reinforcement Learning)персонажі можуть ухвалювати складні рішення, пристосовуватися до стилю гри користувача і навіть демонструвати ознаки координації та емоцій.

Яскравий приклад – The Last of Us Part II, де вороги керуються поведінковим ШІ. Вони здатні координувати дії у групі, перегукуватися, попереджати одне одного про небезпеку і змінювати тактику залежно від

ситуації. Така кооперативна поведінка NPC робить сутички більш непередбачуваними та реалістичними для гравця.

Інший приклад – система ШІ у F.E.A.R. (2005), де застосовано планування дій (GOAP) для ворогів. Вони самостійно вибирають цілі (наприклад, обійти з флангу чи відступити), що створило враження «розумних» супротивників, хоча насправді це результат добре налаштованих правил та планувальників дій.

Адаптивна складність – ще один напрям поведінкової інтеграції ШІ. Алгоритми можуть підлаштовувати рівень складності гри під уміння гравця в режимі реального часу. Якщо користувач демонструє труднощі, ШІ трохи полегшує завдання; якщо ж гра здається надто легкою – підвищує складність. Такий підхід реалізовано, наприклад, через AI Director у хорор-грі Resident Evil 4, що динамічно змінює кількість і силу ворогів та доступні ресурси, аби підтримувати відчуття напруги.

Дослідження підтверджують, що адаптація складності за профілем гравця підвищує його залученість – гравці сприймають гру як оптимально складну і залишаються більш захопленими процесом. Водночас, надмірно агресивна адаптація може викликати фрустрацію, тому балансуванням займаються обережно.

Традиційно для керування NPC використовувалися алгоритми станів та скрипти – наприклад, скінченні автомати станів FSM або дерева поведінки Behavior Trees, що визначають набір можливих дій NPC залежно від умов. Ці методи й досі широко застосовуються через їх передбачуваність та контрольованість.

Розробник заздалегідь прописує реакції на всі типові ситуації: патрулювання, атака, пошук укриття тощо. Однак вони обмежені й часто призводять до передбачуваної поведінки. Натомість впровадження нейромереж дає змогу створити більш емерджентну поведінку, коли NPC можуть вчитися та змінювати дії нестандартним чином.

У практиці великих ААА-ігор нейромережевий контроль NPC поки що використовується обережно через складність тестування та ризик

непередбачуваних результатів. Проте деякі проекти успішно експериментують із цим.

Компанія Blizzard запатентувала технологію еволюціонуючих NPC, які змінюють свою поведінку залежно від рішень гравця, набуваючи унікальних «особистостей».

2.3.3 Персоналізація ігрового процесу

ШІ відкриває можливості для персоналізації сюжету, геймплею та контенту під кожного гравця. Аналізуючи стиль гри користувача та його рішення, алгоритми можуть динамічно змінювати сюжетні лінії чи діалоги, щоб краще відповідати вподобанням гравця. Наприклад, у рольових іграх ШІ може відстежувати, на які аспекти – бій, дослідження чи спілкування – гравець звертає більше уваги, й відповідно пропонувати більше контенту цього типу. Так формується індивідуальний досвід: двоє різних гравців отримують дещо різні пригоди залежно від їх стилю проходження.

Крім того, ШІ дозволяє реалізувати живі ігрові події LiveOps, які генеруються на основі поведінки спільноти гравців. За допомогою алгоритмів аналізу даних розробники можуть відстежувати глобальні тенденції в грі та автоматично запускати події, що підтримують інтерес аудиторії. Наприклад, якщо система фіксує зниження активності, вона може згенерувати спеціальний івент або челлендж з унікальними нагородами, щоб повернути гравців.

Аналітика гравців – ще одна важлива сфера. Вбудовані в гру системи ШІ збирають і обробляють великі обсяги телеметрії для вдосконалення дизайну та балансу. На основі таких даних розробники приймають рішення про патчі, оновлення контенту чи зміни механік. Більше того, ШІ здатний виявляти аномалії в поведінці, що допомагає боротися з чітерами.

Приклад – система Valve Anti-Cheat у Counter-Strike*:* GO, яка використовує алгоритми машинного навчання для аналізу стилю гри та автоматичного блокування шахраїв. Таким чином, інтегрований ШІ забезпечує і більш справедливий багатокористувацький досвід для чесних гравців.

2.3.4 Автоматизація розробки та інструменти на базі ШІ

Окрім безпосереднього впливу на геймплей, ШІ істотно інтегрується у процес розробки ігор. Великі ігрові компанії інвестують у власні дослідницькі підрозділи з ШІ.

Так, Ubisoft створила відділ La Forge, що займається впровадженням наукових досягнень ШІ у практичні рішення для ігор. Однією з цілей Ubisoft є реалістичні NPC, які природно взаємодіють із гравцем і адаптуються до змін у середовищі.

У 2021 році їхні дослідники досягли прориву у покращенні навігації NPC завдяки машинному навчанню – ймовірно, йдеться про алгоритми, що навчають персонажів ефективніше обходити перешкоди чи переслідувати ціль.

Інший приклад – Blizzard Entertainment, що активно застосовує генеративний ШІ для контенту. Внутрішній інструмент Blizzard Diffusion дозволяє автоматично генерувати концепт-арти для ігор, прискорюючи розробку візуальних матеріалів. Також Blizzard працює над ШІ-системами для динамічного створення музики, яка адаптується до ігрових подій і стилю гравця, забезпечуючи унікальний звуковий супровід. Це приклад глибокої інтеграції ШІ: алгоритми беруть на себе творчі завдання у співпраці з митцями.

Microsoft інтегрує ШІ у свою екосистему Xbox та хмарні сервіси Azure, надаючи розробникам платформи для автоматизованого створення ігрових світів. Зокрема, використовуються алгоритми навчання з підкріпленням для розробки інтерактивних ігрових агентів, що можуть правдоподібно взаємодіяти з гравцями. Також Microsoft впровадила голосового ШІ-асистента Bing у Xbox, дозволяючи гравцям швидко отримувати поради чи інформацію про гру прямо під час проходження.

Компанія NVIDIA, окрім створення GPU, запропонувала хмарну платформу Avatar Cloud Engine (ACE) – набір ШІ-інструментів для генерації персонажів, здатних вести діалоги природною мовою (Див. рисунок 2.2). Це дає змогу розробникам легко додавати в ігри NPC з голосовим спілкуванням у

режимі реального часу, що робить взаємодію гравця з віртуальними персонажами значно більш живою і непередбачуваною.

Рисунок 2.2 – Діалогова сцена з платформи Avatar Cloud Engine

Такі інструменти підвищують продуктивність розробки. За опитуванням Game Developers Conference (2023), майже половина опитаних розробників уже використовують генеративні ШІ-інструменти у роботі, причому в інді-студіях частка сягає 37%.

Популярні напрямки – генерація графічних матеріалів, допомога у написанні коду, авто-тестування тощо. Наприклад, Ubisoft повідомила, що застосовує інструмент Ghostwriter для генерації чернеток реплік другорядних NPC– це прискорює написання дрібних діалогів, не замінюючи сценаристів, а допомагаючи їм. Аналогічно, малі студії завдяки ШІ можуть автоматизувати тестування і швидше знаходити баги, що раніше вимагало значних людських ресурсів.

Отже, інтеграція ШІ в ігрову індустрію відбувається на різних рівнях: у самому продукті (грі) – через розумні механіки, NPC і адаптивний геймплей, а також у процесах розробки – через інструменти, що роблять створення ігор швидшим і дешевшим. Далі розглянемо, як ці впровадження впливають на економічні показники та бізнес-моделі індустрії.

2.4 Вплив ШІ на гравця

Розглянемо, як впровадження ШІ-технологій у іграх позначається на досвіді, поведінці та сприйнятті гравців. Адже кінцевою метою будь-яких інновацій є покращення задоволення аудиторії. Дослідження цього аспекту включає як позитивний вплив (більш глибоке занурення, індивідуалізація, нові можливості), так і потенційні ризики чи небажані ефекти для гравців.

2.4.1 Покращення імерсивності та задоволення

Одним з найпомітніших ефектів ШІ в сучасних іграх є зростання рівня занурення . Реалістичніша поведінка NPC, динамічний світ, що підлаштовується під дії гравця, – все це робить гру більш живою. Гравці відзначають, що коли персонажі діють менш шаблонно, реагують на них по-різному, виникає відчуття справжнього світу, а не наперед запрограмованих рейок.

Опитування показують, що геймери дуже схвально ставляться до ідеї «розумних» NPC. За даними дослідження Inworld AI серед 1000 гравців, 99% вірять, що просунуті ШІ-NPC позитивно вплинуть на геймплей, 76% хочуть бачити NPC з кращою ситуаційною обізнаністю, а 78% готові проводити більше часу в іграх з такими NPC. Більше того, 81% респондентів заявили, що готові платити більше за гру, де NPC мають покращений ШІ(Див. рисунок 2.3). Це свідчить про значний попит на глибші взаємодії: гравці прагнуть більш складних і цікавих відносин з персонажами, ніж простий обмін репліками за сценарієм.

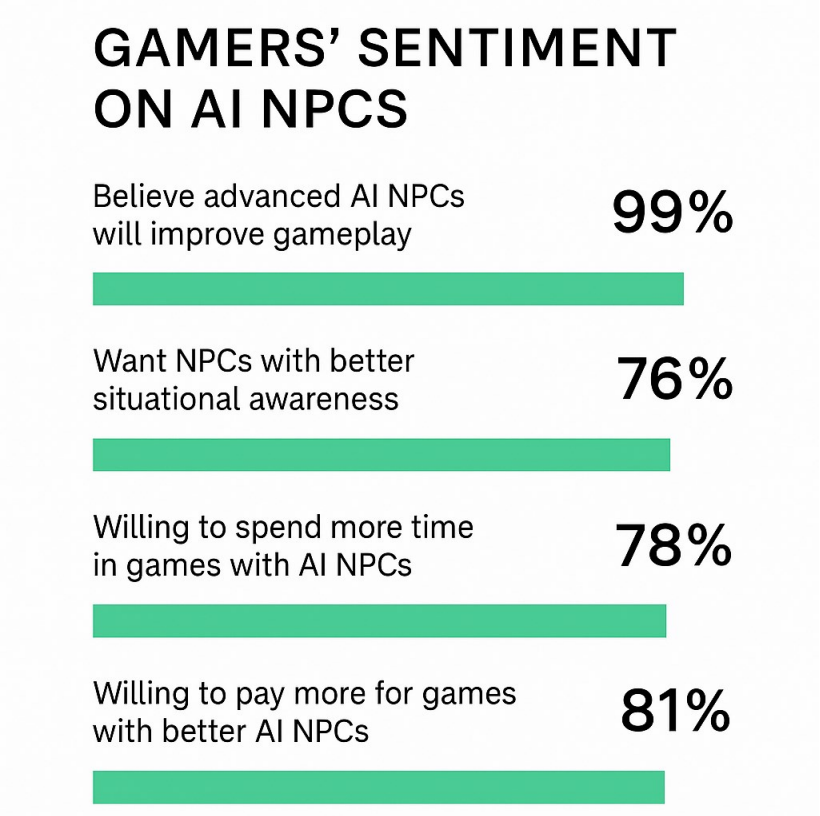

Рисунок 2.3 Дані дослідження Inworld AI, щодо залучення просунутих ШІ-NPC

Персоналізація досвіду також підвищує задоволеність. Коли гра підлаштовується під уміння і вподобання користувача, той отримує відчуття, що гра «розуміє» його. Новачку ШІ може пробачити деякі помилки, підкинути додаткові підказки – і це збереже інтерес замість раннього фрустрування.

Досвідчений гравець, навпаки, оцінить виклик, який підтримується алгоритмом на високому рівні. В результаті ширше коло людей може насолодитися грою: і казуальні, і хардкорні гравці. Динамічне налаштування складності, за даними досліджень, робить ігри комфортними для різних аудиторій, що важливо комерційно гра не відлякає складністю і не набридне простотою.

Соціальна взаємодія у багатокористувацьких іграх також може покращитись завдяки ШІ. Наприклад, системи матчмейкінгу підбору гравців у команди/суперники на основі ШІ можуть формувати більш збалансовані матчі, беручи до уваги не лише навички, а й психологічні профілі гравців.

Алгоритм може звести разом тих, хто схильний до кооперації, і окремо – тих, хто любить суперництво. Це підвищує задоволення від спільної гри,

уникаючи токсичних ситуацій. Хоча такі підходи ще експериментальні, потенціал у них значний.

2.4.2 Інструменти генеративної творчості для гравця

ШІ не лише робить гравця споживачем контенту, а й надає йому інструменти стати співтворцем. Якщо раніше модифікація ігрового світу вимагала технічних навичок , то в майбутньому, завдяки інтегрованим ШІ, гравець зможе словами чи простими діями змінювати світ гри.

Вже зараз простежується тренд на користувацький контент, збагачений ШІ. Наприклад, гра Minecraft у співпраці з OpenAI експериментувала з ботом, якому можна було описати будівлю – і він зводив її з блоків.

У проекті Retail Mage від студії Jam & Tea гравці можуть взаємодіяти з чарівною крамницею, буквально кажучи NPC, що вони хочуть зробити з предметом – і система ШІ це реалізує в грі (Див. рисунок 2.4).

Рисунок 2.4 Сцена інтерактивної взаємодії гравця з NPC

Така імпровізаційна свобода, притаманна настільним RPG, тепер переноситься у цифрові ігри. Гравці можуть придумувати власні способи використання предметів чи спілкування з персонажами, виходячи за межі заскриптованих опцій діалогу. Це робить кожну історію унікальною: фактично, гравець стає співавтором свого ігрового досвіду.

Розвиток сюжетів за внеском гравця – ще один аспект. ШІ-оповідач може підтримати наратив, реагуючи на нетипові вчинки гравця. Наприклад, якщо у звичайній грі квест вимагає віднести предмет з пункту А в пункт Б, і тільки це зараховується, то з ШІ можлива ситуація, що гравець знайде творчий спосіб застосувати предмет інакше, і гра це зрозуміє та відповідно змінить сюжет. Це підсилює відчуття свободи і значущості вибору.

Дослідження показують, що гравці цінують можливість розповідати власні історії в іграх. ШІ дає технологічну підтримку цьому бажанню.

Карл Кво (CEO Jam & Tea) зазначає, що мета використання ШІ – дати гравцям змогу розповідати історії, які вони раніше не могли, і відчувати більший зв'язок один з одним через гру. Це свідчить про зміщення акценту, від фіксованого контенту авторів – до sandbox-підходу, де гравці за допомогою ШІ створюють контент для себе та спільноти.

2.4.3 Адаптивне навчання та супровід гравця

ШІ в іграх може виступати і як наставник, допомагаючи гравцям вчитися і долати труднощі. Наприклад, згадана компанія Artificial Agency розробляє ШІ-агентів - «наглядачів», які відстежують прогрес гравця і скеровують його до контенту, який він міг пропустити.

Якщо гравець заблукав або не знає, що робити далі, такий агент може м'яко підказати чи навіть згенерувати туторіал на льоту, щоб навчити необхідної навички. Це покращує досвід новачків, зменшує відтік через складність чи непорозуміння механік. На відміну від статичних підказок, ШІ-ассистент може діалогово взаємодіяти: гравець спитає у віртуального персонажа «що мені робити з цим артефактом?», і отримає пораду, релевантну контексту.

Такий персональний підхід створює відчуття, що гра піклується про гравця. У грі Mecha Break було продемонстровано прототип, де гравець може попросити механіка-NPC пофарбувати його робота в інший колір – і той виконає прохання, замість того щоб гравцеві лізти в меню налаштувань. Це дрібниця, але вона робить взаємодію більш «людською» і швидкою, що цінують користувачі.

Фактично, гравець взаємодіє зі світом голосом та мовою, а не через технічний інтерфейс, що спрощує доступність.

2.4.4 Потенційні ризики та проблеми з боку користувачів

Не все, що приносить ШІ, однозначно сприймається гравцями. Є і потенційні негативні наслідки або побоювання.

Деякі гравці висловлюють думку, що надто «розумні» або адаптивні противники не обов'язково роблять гру цікавішою. Парадоксально, але інколи передбачувана поведінка ворогів – це частина дизайну, що дозволяє гравцю навчитися і перемогти. Якщо ж ШІ-опонент почне діяти занадто непередбачувано чи ефективно, гравець може відчути, що гра нечесна. Є приклади скарг на «асиметричний» ШІ, коли гравцю здається, що комп'ютер шахраює.

Класичний випадок – rubber-banding у гоночних іграх (ШІ-суперники штучно наздоганяють, якщо гравець сильно відірвався), що покликане тримати інтригу, але декого дратує. Отже, сприйняття балансу дуже важливе: розробники мусять тонко налаштовувати алгоритми, щоб підтримувати задоволення, а не лише виклик.

Якщо гра занадто покладається на генеративну імпровізацію, можуть виникати ситуації наративного дисонансу. Наприклад, ШІ-NPC може згенерувати репліку, яка суперечить лору гри, або запропонувати дії, які неможливі в межах ігрової механіки.

Як зауважує дослідник Рід Берковіц: «*AI-діалоги можуть повністю зламати гру*», якщо NPC почне говорити про речі, яких не існує у грі (на кшталт «Ходімо вип'ємо кави у сусідньому кафе», коли ніякого кафе в грі немає). Такі помилки ламають занурення і псують досвід. Рішенням є строгий зв'язок між ШІ та ігровим станом: моделі мають знати, що існує, а що ні в грі, і уникати виходу за ці рамки.

Коли NPC стають дуже реалістичними, виникає новий пласт питань – наприклад, емоційна прив'язаність гравців до ШІ-персонажів. Якщо гравець

подовгу спілкується з віртуальним компаньйоном, що вміє жартувати, підтримувати розмову, співчувати, – це може формувати майже дружні почуття.

З одного боку, це свідчить про успіх дизайну, з іншого – ставить питання психологічного благополуччя. Чи не буде заміна реальних соціальних зв'язків віртуальними шкідливою для деяких людей?

Поки що такі випадки радше гіпотетичні, але розробники починають думати і про це. Можливо, будуть вводитися обмеження або попередження, щоб гравці усвідомлювали умовність стосунків із ШІ.

Оскільки ігри з ШІ збирають і аналізують багато даних про поведінку гравця, постає питання захисту цих даних. Гравці можуть хвилюватися, чи не становлять такі системи загрозу приватності.

В іграх VR з ШІ, які відстежують біометрію для адаптації контенту, ця стурбованість ще більша. Відповідальні студії мають забезпечити прозорість: які дані збираються, для чого, і гарантувати їх невикористання поза межами покращення досвіду гравця.

2.5 Методи контролю NPC і діалогів

Неігрові персонажі (NPC) та система діалогів – ключові елементи будь-якої сюжетної гри. Цей розділ присвячений аналізу методів, за допомогою яких здійснюється управління поведінкою NPC і генерація діалогів, з акцентом на порівнянні традиційних підходів та інноваційних рішень на основі ШІ.

2.5.1 Традиційні методи керування NPC

Класично поведінка NPC визначається за допомогою дискретних моделей і скриптів. Серед найпоширеніших методів:

Скінченний автомат станів (Finite State Machine, FSM) де NPC має фіксований набір станів: патрулює, переслідує гравця, атакує, тікає тощо, і правила переходу між ними залежно від умов. Наприклад, ворог бачить гравця –

переходить зі стану «патруль» у стан «атака»; втратив гравця з виду – повертається до «патруля».

FSM прості у реалізації і передбачувані, тому широко використовувалися від ранніх аркад до сучасних ігор для простих поведінкових задач.

Дерева поведінки (Behavior Trees) ієрархічна структура, де вузли перевіряють умови або виконують дії, а гілки визначають послідовності або варіанти поведінки.

Дерева поведінки гнучкіші за FSM – їх легше масштабувати для складних персонажів. Вони активно застосовуються в рушіях Unity, Unreal для реалізації AI ворогів.

Наприклад, дерево може спочатку перевірити «чи бачить NPC гравця?» – якщо так, виконати піддерево атакувальних дій, якщо ні – піддерево блукання. Behavior Tree можна розширювати, не ламаючи всю структуру, тому великі проекти надають їм перевагу.

В багатьох іграх NPC керуються набором скриптів, прив’язаних до певних подій чи локацій. Наприклад, у RPG NPC має розклад дня (йде на ринок, потім додому), прописаний вручну. Або в шутері – за скриптом підкріплення прибуває, коли гравець заходить у певну зону. Такий підхід забезпечує жорстко задані, відточені сценарії, але не дає гнучкості: якщо гравець відхилиться від очікуваного маршруту, NPC можуть виглядати неприродно.

Планування дій за цілями (Goal-Oriented Action Planning, GOAP). Метод, де NPC має ціль (наприклад, «перемогти гравця») і набір можливих дій. Алгоритм планування сам складає послідовність дій, виходячи з поточної ситуації, щоб досягти мети. Цей підхід використали, зокрема, у грі F.E.A.R.: противники динамічно вирішували, як атакувати – кидати гранати, обходити з флангу тощо, залежно від позиції гравця і наявних ресурсів. GOAP надає більш «мислячу» видимість NPC, хоча за сценою це просто пошук по простору дій. Недоліком є складність реалізації та налаштування – треба визначити багато умов і ефектів для кожної дії.

Традиційні методи хороші тим, що вони детерміновані і контрольовані розробником. Вони гарантують, що NPC буде діяти у рамках задуманого

дизайну. Це важливо для збалансованого геймплею – розробник може зробити ворога рівно настільки складним, як потрібно, і відтестувати всі варіанти його поведінки.

Однак, така скриптована природа веде до повторюваності та обмеженості. Досвідчені гравці швидко вивчають шаблони поведінки ворогів, що знижує виклик. NPC можуть ігнорувати ситуації, не передбачені скриптом (наприклад, зациклитися на перешкоді). Різноманітність поведінки обмежена обсягом, який встигли прописати дизайнери. Це стимулювало пошук нових рішень, зокрема із залученням машинного навчання та генеративних моделей, аби наповнити NPC більшою варіативністю і «людяністю».

2.5.2 Генеративні та навчальні методи керування NPC

Сучасні дослідження та деякі ігрові проекти впроваджують адаптивний ШІ для NPC, що виходить за межі жорстких сценаріїв. Ідея полягає в тому, щоб NPC міг вчитися або генерувати нові поведінкові патерни на льоту.

Один підхід – використання навчених моделей поведінки. Наприклад, нейромережу можна навчити на даних реальних гравців і надалі використати як бота.

Так було зроблено в експериментах з шутером Battlefield, ІІ-боти навчалися пересуватися картою, спостерігаючи за тисячами сесій гравців, і почали приймати рішення, схожі на людські.

У комерційних іграх поки рідко бачимо повністю нейромережевих NPC через складність контролю, але елементи такого навчання застосовуються. Ubisoft у своєму R&D показувала, як агент з RL вчиться керувати машиною в грі, обходячи перешкоди – такий агент міг би замінити собою вручну написаний AI водія.

Навчання з підкріплення перспективне тим, що дозволяє NPC самостійно освоїти оптимальну поведінку через багато спроб і помилок у симуляції. Потім модель вбудовується в гру. Це зменшує трудозатрати дизайнерів на

прописування правил – замість цього потрібно правильно визначити нагороду для агента.

Окремо варто згадати нейронні мережі для анімацій і фізики NPC. Хоч це не прямо «рішення», але дуже впливає на сприйняття поведінки.

В грі Red Dead Redemption 2 використано технологію Euphoria: фізично достовірні анімації реакцій NPC на поранення, удари, рельєф місцевості генеруються в реальному часі на основі ШІ.

Хоча логіка дій персонажа може бути проста, реалістична анімація створює ілюзію розумності – ворог, поранений в ногу, кульгає і намагається відповзти, що виглядає як прояв інстинкту самозбереження, хоча за цим стоїть алгоритм фізичної симуляції. Тобто ШІ допомагає «олюднити» NPC не тільки на рівні рішень, а й на рівні рухів.

2.5.3 Діалоги: від дерев рішень до мовних моделей

Системи діалогів традиційно будувалися як розгалужені дерева реплік (dialogue trees). Гравцю надається вибір з кількох реплік, на кожну NPC має заготовлену відповідь, і так далі. Цей підхід забезпечує повний контроль над сценарієм – сценарист прописує всі можливі варіанти розмови.

Недоліком є обмеженість, гравець не може сказати нічого, що не передбачено варіантами, і через це NPC виглядають глухими до всього поза вузьким коридором тем.

Генеративний ШІ відкриває можливість динамічних діалогів. Завдяки моделям обробки природної мови (NLP), таким як GPT-3/4, NPC теоретично можуть спілкуватися вільним текстом чи голосом, розуміти довільні питання гравця і відповідати нелінійно.

Уже продемонстровано прототипи, де гравець говорить у мікрофон, а NPC-чатбот відповідає голосом, синтезованим у реальному часі. Наприклад, на CES 2024 NVIDIA спільно з Convai показала кіберпанк-сцену: двоє NPC (бармен і його знайома) вели між собою бесіду, а гравець міг до них приєднатися зі своїми

питаннями – і отримував осмислені відповіді, хоч і з певними паузами. Цей демо підтвердив: технологія вже дозволяє інтерактивний голосовий діалог з NPC, нехай поки й відчуваються недоліки (невелика затримка, іноді механічна інтонація).

Для гравця ж враження значно перевершує стандартні діалогові дерева – можливість спитати «а що ти думаєш про погоду?» чи «розкажи про це місто» і почути змістовну відповідь робить NPC майже персонажем з людьми за кадром.

Ряд компаній активно розвиває фреймворки для AI-NPC. Згадана NVIDIA ACE включає модулі розпізнавання мовлення, LLM для генерації відповіді і voice-to-speech для озвучення – повний стек для інтеграції в ігри. Inworld AI надає платформу, де розробник може задавати параметри характеру NPC, а Inworld генерує відповідну модель поведінки і мови. Ubisoft експериментує з прототипом Neo NPC, де генеративний ШІ пробує розширити межі взаємодії гравця і персонажа без втрати правдоподібності сюжету.

Попри переваги, генеративні діалоги мають виклики. Ключовий – як згадувалось, контекстуальність і достовірність. Модель повинна розуміти лор гри, стан квестів, особистість NPC і не виходити за них. Це вимагає спеціальної підготовки даних і механізмів «гальмування» творчості моделі в потрібних моментах (щоб не вигадувала зайвого). Ubisoft у своєму проекті Neo NPC вирішує це через обмеження сценаріїв – генеративний ШІ може варіювати формулювання реплік, додавати дрібні деталі, але каркас сюжету залишається незмінним. Таким чином досягається баланс між свободою і контрольованістю.

Другий виклик – ресурси і оптимізація. Запуск LLM локально в грі з сотнями NPC – поки непідйомна задача для заліза. Тому рішення пропонуються хмарні: коли гравець ініціює розмову, запит іде на сервер, де потужні моделі генерують відповідь. Це породжує залежність від інтернету та витрати на сервери, але інакше важко. NVIDIA акцентує на використанні малих мовних моделей для пришвидшення відповіді NPC. Їхня ідея: навчити компактну модель спеціально під персонажа (його стиль мовлення, знання), яка працюватиме швидко і дешево, нехай і менш креативно, ніж великі GPT. Вже є успіхи – прототип від NVIDIA відповідав майже миттєво, що критично для динаміки гри.

Приклади застосування вже з’являються в реальних продуктах. У грі The Elder Scrolls V: Skyrim ентузіасти випустили модифікацію, яка додає декілька NPC з AI-діалогами (через API OpenAI) – гравці могли з ними розмовляти про світ, отримувати поради. Це отримало схвальні відгуки, хоча й неофіційний експеримент. Очікується, що перші комерційні ігри з такою механікою вийдуть в 2024–2025 роках. Можливо, це будуть інді-проекти або MMO, де взаємодія з NPC – важлива частина соціального простору.

Безпека контенту – ще одна сторона: генеративний діалог може випадково видати небажаний текст (образливий, невідповідний рейтингу гри). Розробники мусять впроваджувати фільтри і цензуру, щоб такі ситуації виключити. Це технічно аналогічно модерації чат-ботів, але в іграх особливо важливо, бо аудиторія часто неповнолітня.

2.5.4 Синтез методів: гібридні системи

Ймовірно, найкращі результати дає комбінація традиційних і нових методів. Наприклад, гібридна AI-система може працювати так:

На високому рівні логіка NPC керується класичним планувальником або деревом (гарантує сюжетну цілісність).

На середньому рівні – прийняття тактичних рішень може бути делеговане навчальним модулям (щоб вороги варіювали дії).

А на рівні взаємодії з гравцем – підключається генеративний модуль для природних діалогів або реактивних реплік.

Таким чином, стратегічний контроль зберігається, а дрібні деталі оживляються ШІ. Ubisoft у своїх дослідженнях дійшла висновку, що оптимально впроваджувати ШІ поступово, не віддаючи йому повний контроль над ігровим світом. Великі студії поки обережні, адже повністю автономний AI-процес може нагенерувати непередбачуваних ситуацій або багів, виправити які важко. Тому перші впровадження – обмежені й контрольовані.

Прогноз на майбутнє NPC і діалогів: очікується, що протягом найближчих кількох років ми побачимо значне розширення використання генеративних NPC

у іграх жанру RPG, симуляторах відкритого світу. Можливо, з’являться ігри, де жоден NPC не має наперед визначеного діалогу – усі розмови створюються ШІ на основі опису персонажа і ситуації. Це відкриє практично нескінченні можливості для role-play: гравець зможе спілкуватися з персонажами так, як уявляє, без обмежень меню діалогу. Водночас, щоб такі ігри були якісними, технологія повинна досягти ще більшої зрілості – зменшити галюцинації, прискорити роботу, пройти цензуру.

Деякі експерти вважають, що в майбутньому можливі цілі жанри ігор, де ШІ виступає «Майстром гри», як у настільних RPG. Тобто гра генерує сюжет, персонажів, NPC, діалоги на льоту під гравців. Перші кроки до цього вже зроблені (AI Dungeon). Якщо ці ідеї розвинуться, то поняття «повторне проходження гри» може зникнути – кожен прохід буде новою історією.

Наразі ж оптимальним підходом є використовувати ШІ для збагачення традиційних систем. Наприклад, дозволити гравцю задавати питання NPC текстом/голосом, але відповіді генерувати на основі банку знань, що підготували сценаристи (щоб уникнути вигадок). Це дає ілюзію свободи, але під контролем. Саме так вчиняють деякі сучасні інтерактивні фікшн-проекти: GPT-модель «навчають» на текстах, написаних авторами, тож вона вміє імпровізувати в стилі оригіналу.

В цілому, методи контролю NPC і діалогів еволюціонують у бік більшої гнучкості та реалізму, але повна заміна старих методів на ШІ ще не сталася. Як і в попередніх розділах, спостерігається тенденція до симбіозу людини і ШІ: дизайнери встановлюють рамки і цілі, а ШІ – заповнює деталями та варіаціями. Це дозволяє створити NPC, які з одного боку діють і говорять в рамках логіки ігрового світу, а з іншого – не відчуваються механічними чи повторюваними.

2.6 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було всебічно розглянуто сучасні методи інтеграції генеративного штучного інтелекту в ігрові системи, зокрема мобільні застосунки.

Здійснено огляд наукових досліджень, які демонструють актуальні підходи до використання великих мовних моделей (LLM) для генерації діалогів NPC, процедурної побудови ігрового контенту, моделювання поведінки персонажів та адаптації геймплею до індивідуального стилю користувача.

Проаналізовано низку перспективних концепцій — від комбінованих систем на основі LLM та графів знань до моделей генерації рівнів із застосуванням нейронних клітинних автоматів та алгоритмів навчання з підкріпленням.

Також було акцентовано на перевагах і недоліках таких підходів в контексті створення сценаріїв і діалогів, що відповідають настрою, стилю та логіці ігрового світу. Окрему увагу приділено технічним викликам, які виникають при реалізації ШІ на мобільних платформах: обмеженням обчислювальних ресурсів, складнощам з продуктивністю та необхідністю балансування між локальною і серверною обробкою.

Було відзначено потребу в оптимізації нейромережевих моделей, впровадженні контрольних механізмів для підтримки наративної цілісності та адаптивному управлінні сюжетами.

Таким чином, результати аналізу підтверджують практичну доцільність використання генеративного ШІ в розробці мобільних ігор нового покоління.

Узагальнені дослідження слугували базою для обґрунтування вибору архітектурних рішень у наступному розділі, де буде розглянуто безпосередню реалізацію прототипу ігрового застосунку з динамічним сюжетом і поведінковими NPC.

3. РОЗРОБКА МОБІЛЬОГО ЗАСТОСУНКУ З ІНТЕРГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ ТА АНАЛІЗУ ПРИЙНЯТТЯ РІШЕНЬ

3.1 Аналіз і мета інтеграції штучного інтелекту у мобільний застосунок

Проєкт передбачає створення гри-новели «Механічне серце» з динамічною зміною сюжету та діалогів. Замість статичного, заздалегідь прописаного сценарію використовується мовна модель штучного інтелекту, яка в реальному часі генерує відповіді персонажів на основі дій гравця та поточного контексту. Це робить кожну ігрову сесію унікальною сюжет формується як мережа можливих подій, що вибудовується динамічно в процесі проходження гри.

Центральним елементом системи є модуль генерації тексту, заснований на глибокій нейронній мережі. Він взаємодіє з компонентом прийняття рішень, який оцінює контекст вибору гравця та поточний стан сюжету.

На основі цієї інформації модуль формує нові репліки діалогів і розвиток подій у режимі реального часу. Така архітектура дозволяє обходитися без заздалегідь підготовлених усіх можливих варіантів історії. Замість фіксованих сценаріїв гра автоматично синтезує необхідні текстові елементи: кожен вибір гравця ініціює генерацію нових діалогів та розвитку сюжетних подій, адаптованих під поточний контекст ігрової сесії.

3.3.1 Новизна підходу

Запропонований підхід вирізняється кількома важливими інноваційними аспектами:

– Гнучка логіка поведінки персонажів ігрового світу. Вони формують свої репліки не з фіксованого сценарію, а на льоту за допомогою алгоритмів штучного інтелекту. Модель генерації аналізує попередній контекст та особливості характеру кожного персонажа, щоб видавати діалоги, що

виглядають природними та доречними. Таким чином NPC демонструють варіативну поведінку та емоційні реакції залежно від дій гравця, що робить їх більш адаптивними і реалістичними порівняно з традиційними новелами.

– На відміну від класичних ігор, де ігрова логіка базується на заздалегідь заданих гілках вибору, у цьому проєкті кожна взаємодія користувача з грою обробляється через нейронну мережу. Кожен вибір діалогу чи крок сюжету відправляється до модуля ШІ, який на основі вхідних даних генерує відповідь персонажів. Це перетворює гру на платформу реального спілкування – розробники не прописують усі можливі діалоги вручну, адже інтелект сам підтримує розвиток сюжету.

– Інтеграція генеративного ШІ забезпечує відмову від традиційного лінійного сценарію. Замість цього сюжет може розгалужуватися та змінюватися залежно від попередніх рішень гравця. Кожне проходження гри може давати унікальний результат, оскільки нові гілки історії виникають у процесі гри. Така непередбачуваність і варіативність сюжету недоступні в стандартних візуальних новелах із жорстко прописаним сценарієм.

Перераховані інновації дозволяють створити глибокий і реалістичний ігровий світ, де персонажі поводяться природно, а історія адаптується під дії кожного користувача.

3.1.2 Обґрунтування доцільності

Застосування штучного інтелекту в описаному проєкті є виправданим з огляду на низку ключових переваг:

– Використання ШІ дозволяє персоналізувати контент під конкретного гравця. Система може аналізувати попередні дії користувача та генерувати діалоги, що відповідають його стилю взаємодії. Це підвищує зацікавленість гравця, оскільки він отримує відчуття індивідуального сюжету і більш живого спілкування з персонажами. Наприклад, у грі з елементами тренувальних симуляцій персонаж може пропонувати варіанти дій, які враховують попередній досвід гравця, роблячи процес гри більш захопливим.

– У звичайних новелах сценаристам доводиться вручну прописувати численні діалоги і події. Інтеграція ШІ суттєво скорочує ці витрати праці, оскільки модель автоматично генерує текстові фрагменти під час гри. Це дозволяє розробникам зосередитися на загальній концепції та ключових ідеях сюжету замість написання кожної репліки. Замість десятків заготовлених рядків достатньо налаштувати алгоритм, який створюватиме необхідні фрази в залежності від розвитку історії.

– Використання передових ШІ-технологій робить проєкт інноваційним та сучасним. Гра з динамічним сюжетом і адаптивними персонажами привертає увагу гравців та експертів, які цінують новаторські рішення. Крім того, інтеграція штучного інтелекту відповідає світовим трендам у геймдеві і може забезпечити швидку адаптацію продукту до майбутніх стандартів індустрії. Сучасні гравці очікують, що ігри використовуватимуть передові технології і надаватимуть нові інтерактивні можливості, тому цей підхід створює суттєву перевагу.

Ці чинники підкреслюють доцільність використання ШІ в розробці інтерактивної новели та забезпечують обґрунтування подальшої роботи над проєктом.

3.1.3 Ідея розвитку

Подальший розвиток проєкту передбачає розширення можливостей та вихід на нові сфери застосування:

Технологію генерації контенту можна масштабувати шляхом розширення навчальних даних моделі і додавання нових сюжетних елементів. Наприклад, збільшення обсягу та різноманіття текстових корпусів дозволить створювати більш довгі та багаторівневі історії. Проєкт можна адаптувати для різних вікових груп та ринків, додавши мультимовність та нових персонажів. Це відкриє можливості для випуску оновлень, які розширюватимуть ігровий світ, а також для портування проєкту на інші платформи.

Технологія динамічної генерації діалогів може бути використана для навчальних ігор та тренажерів. Наприклад, інтерактивні історії можуть допомагати у вивченні іноземних мов через розмови з віртуальним наставником, адаптуючи складність мови до рівня учня. Також система може імітувати історичні події або наукові експерименти у формі діалогів, що робить навчання більш захоплюючим і персоналізованим.

На рисунку 3.1 зображено головне меню та безпосередньо самої гри.

Рисунок 3.1 Інтерфейс гри «Механічне серце»

Подібний підхід може бути застосований у професійних тренажерах. Наприклад, бізнес-тренажер із інтерактивними переговорниками допомагатиме відпрацьовувати навички комунікації: відповіді віртуальних партнерів генеруються ШІ залежно від обраної тактики користувача. У медичній галузі можна створити симуляцію пацієнта з різними симптомами, де дії лікаря супроводжуються адекватними реакціями віртуального пацієнта. Завдяки

штучному інтелекту такі симуляції будуть адаптивними і наближеними до реальності.

Генеративні моделі можна розширити компонентами, що імітують емоції персонажів. Це дозволяє віртуальним співрозмовникам демонструвати настрій, емпатію або напруженість залежно від дій гравця. Такі можливості відкриваються в програмах психологічної підтримки та соціальної адаптації, де імітуються реальні емоційні реакції.

Таким чином, концепція інтеграції ШІ у мобільний застосунок відкриває широку основу для майбутніх досліджень та розробок. Подальший розвиток цієї системи може призвести до створення нових інтерактивних сервісів в освіті, професійних симуляціях та програмах психологічної підтримки, що виходять за рамки суто розважального контенту.

3.2 Послідовність розробки і структура проєкту

Розробка мобільного застосунку «Механічне серце» здійснювалась у кілька послідовних етапів, кожен з яких мав свою функціональну мету та технічні особливості. Послідовність дій формувалась на основі принципів гнучкої розробки з акцентом на поступове розширення функціональності та інтеграцію мовної моделі штучного інтелекту.

1. Формування концепції гри. На першому етапі було визначено загальну ідею: створення візуальної новели з динамічною побудовою сюжету та інтерактивними діалогами, які генеруються за допомогою ШІ. Було вирішено відмовитися від лінійного сценарію на користь адаптивної архітектури з персоналізованими відповідями NPC.

2. Проєктування архітектури. Було спроєктовано структуру гри: розділення відповідальностей між GameManager, DialogueUI, AIManager, NPCManager, CharacterState та SaveSystem.

3. Реалізація базової структури гри в Unity. Створено базову сцену, UI-елементи, а також реалізовано логіку діалогів без ШІ (на локальних масивах). Це дозволило відлагодити інтерфейс, механіку вибору, систему сцен та керування станом.

4. Інтеграція мовної моделі ШІ. Було реалізовано клас AIManager, що надсилає запити до мовної моделі через API. Сформовано формат запиту (prompt), десеріалізацію відповіді, а також механізм передачі відповіді до UI.

5. Обробка стану персонажів та NPC. Створено структури пам’яті NPC, додано систему збереження їх ставлення до гравця та логіку реакції на вибрані дії. Реалізовано CharacterState для відстеження еволюції головного героя.

6. Оптимізація під Android роведено тестування на мобільних пристроях, оптимізовано обробку запитів, зменшено використання пам’яті, впроваджено збереження стану гри у JSON.

7. Фінальне тестування та балансування. Було виконано ручне тестування діалогів, обробки помилок API, випадків розриву інтернет-з’єднання, а також перевірено збереження/відновлення гри.

Архітектура проєкту побудована на модульному принципі, що забезпечує гнучкість, масштабованість і спрощує тестування окремих компонентів. У таблиці 3.1 наведено функціональне розділення між основними модулями.

Таблиця 3.1 Функціональне розділення між основними модулями гри.

Клас / Компонент	**Основна відповідальність**	**Взаємодіє з**
AIManager	Надсилання запитів до мовної моделі, обробка відповідей	GameManager, DialogueUI
GameManager	Керування станами гри, ініціація діалогів, зміна сцен	Усі інші класи
DialogueUI	Відображення текстів і відповідей у UI	GameManager
NPCManager	Відстеження пам’яті NPC, рівня довіри	GameManager, CharacterState
CharacterState	Збереження рівня, характеристик та виборів ГГ	GameManager, SaveSystem
SaveSystem	Збереження і відновлення всіх станів гри	CharacterState, NPCManager

Кожен клас реалізує конкретну відповідальність і взаємодіє з іншими через чітко визначені інтерфейси. Взаємозв’язок між компонентами побудовано за схемою залежності "згори вниз", де GameManager координує усі взаємодії.

Особливості структури:

– Класи не залежать один від одного безпосередньо, що дозволяє модифікувати або замінювати їх незалежно.

– Вся бізнес-логіка винесена у GameManager і AIManager, тоді як UI лише відображає інформацію.

– Додавання нових персонажів або сценаріїв не вимагає переписування основних компонентів.

3.2.1 AIManager – запити до мовної моделі і обробка відповіді

Клас AIManager є центральним компонентом системи, що забезпечує зв’язок гри зі зовнішньою мовною моделлю. Основне призначення цього класу – формувати промпти до моделі, відправляти їх через API, а також отримувати та обробляти відповіді. Таким чином, AIManager відповідає за динамічне генерування контенту діалогів та інших текстових елементів гри на основі штучного інтелекту, інтегруючи зовнішню модель у внутрішню архітектуру.

Структура та функціональна логіка класу AIManager: Клас зазвичай реалізовано як нащадок MonoBehaviour для можливості використання корутин та інтеграції в Unity-сцену. Він містить поле для зберігання URL-адреси API мовної моделі та токенів доступу для аутентифікації запитів це зображено на рисунку 3.2.

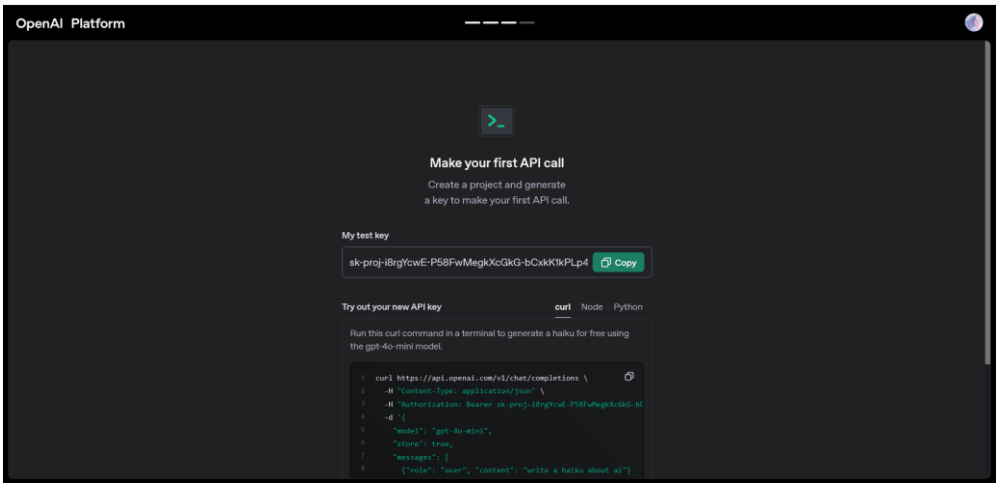

Рисунок 3.2 – Інтерфейс OpenAI Platform із прикладом створення першого API-запиту до моделі GPT.

Також передбачена можливість налаштування параметрів запиту, наприклад, обсягу або характеристик генерованого тексту. Основна функціональність надається методом, який приймає вхідний текст, відправляє HTTP-запит до сервісу AI і опрацьовує відповідь у вигляді текстового рядка. Нижче на рисунку 3.3 наведено фрагмент коду з ключовими полями і методами класу AIManager, розбитими на логічні блоки.

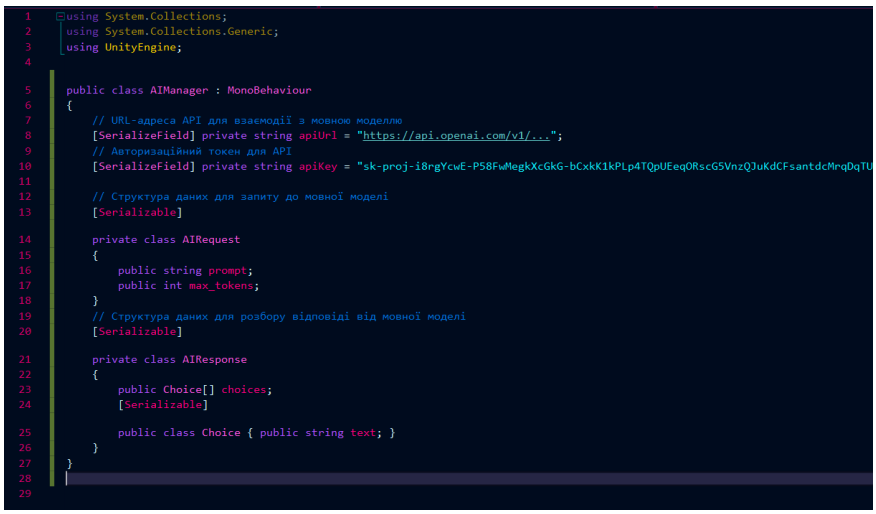

Рисунок 3.3 Фрагмент коду з ключовими полями і методами класу AIManager

У наведеному блоці оголошено поля apiUrl та apiKey, які зберігають адресу API сервісу та ключ для авторизації запитів. Далі визначено внутрішні допоміжні класи та AIResponse.

Перший задає структуру запиту (наприклад, текст промпту і максимальну довжину відповіді), що потім серіалізується у JSON.

Клас AIResponse на рисунку 3.4 описує формат отриманої відповіді: зазвичай це масив об’єктів з полем text, яке містить згенерований текст.

```
public void SendRequest(string prompt, Action<string> callback)
{
    StartCoroutine(SendRequestCoroutine(prompt, callback));
}

private IEnumerator SendRequestCoroutine(string prompt, Action<string> callback)
{
    // Підготовка об'єкту запиту
    AIRequest requestData = new AIRequest { prompt = prompt, max_tokens = 150 };
    string jsonData = JsonUtility.ToJson(requestData);

    // Створення та налаштування HTTP-запиту
    using (UnityWebRequest request = new UnityWebRequest(apiUrl, "POST"))
    {
        byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(jsonData);
        request.uploadHandler = new UploadHandlerRaw(bodyRaw);
        request.downloadHandler = new DownloadHandlerBuffer();
        request.SetRequestHeader("Content-Type", "application/json");
        request.SetRequestHeader("Authorization", $"Bearer {apiKey}");

        // Надсилання запиту і очікування відповіді
        yield return request.SendWebRequest();

        // Обробка результату запиту
        if (request.result == UnityWebRequest.Result.Success)
        {
            AIResponse aiResponse = JsonUtility.FromJson<AIResponse>(request.downloadHandler.text);
            string responseText = aiResponse.choices[0].text.Trim();
            callback?.Invoke(responseText);
        }
        else
        {
            Debug.LogError($"AI request failed: {request.error}");
            callback?.Invoke(string.Empty);
        }
    }
}
```

Рисунок 3.4 Фрагмент коду модуля AIResponse

Цей код демонструє головний механізм роботи класу. Метод SendRequest є публічним інтерфейсом: він отримує текстове повідомлення (промпт) і виклик (делегат) callback, який буде викликаний після отримання відповіді.

Всередині запускається корутина SendRequestCoroutine, де відбувається формування об’єкта AIRequest з промптом та максимальним числом токенів, переведення його в JSON за допомогою JsonUtility, та підготовка HTTP-запиту через UnityWebRequest.

Встановлені необхідні заголовки – тип вмісту та авторизаційний токен. Після надсилання запиту виконується очікування відповіді (yield return request.SendWebRequest()).

Якщо запит успішний, текст відповіді десеріалізується в об’єкт AIResponse, звідки витягується сам згенерований текст. Цей текст передається через колбек до викликачів. У випадку помилки, виводиться повідомлення в консоль і через колбек передається порожній рядок.

Взаємодія з іншими компонентами: AIManager зазвичай використовується класом GameManager або безпосередньо діалоговим інтерфейсом для отримання відповідей на задані промпти.

Наприклад, коли користувач вибирає варіант репліки або при зміні сюжету викликається метод SendRequest з певним текстом, а після отримання відповіді клас AIManager передає згенерований текст назад у GameManager чи DialogueUI.

Таким чином забезпечується розділення відповідальностей: AIManager лише формує та обробляє запити до мовної моделі, не беручи участі в логіці гри чи UI. Завдяки подібній архітектурі, при необхідності можна легко замінити метод взаємодії з AI чи сам сервіс, не торкаючись інших систем.

3.2.2 GameManager –координація загального стану гри, зміна сцен та запуск діалогів.

Клас GameManager є центральним координатором ігрової логіки. Він відповідає за управління загальним станом гри, ініціалізацію та завершення діалогів, перемикання між сценами та інші глобальні процеси.

Зазвичай GameManager реалізовано як синглтон (singleton) – це гарантує, що в грі існує єдиний екземпляр цього класу, доступний з будь-якого місця коду. Він зберігає посилання на інші важливі компоненти (наприклад, на AIManager або інтерфейс діалогів DialogueUI) і визначає послідовність дій під час проходження візуальної новели. Завдяки цьому досягається централізоване управління ігровим процесом.

Структура та функціональна логіка класу GameManager.

Клас зазвичай містить приватне статичне поле instance для реалізації патерну Singleton і публічний статичний властивість Instance, щоб інші класи

могли отримати доступ до нього. У методі Awake() перевіряється, чи не існує іншого екземпляра, і при необхідності зберігається посилання на нього.

Серед полів GameManager можуть бути посилання на DialogueUI (для управління відображенням діалогів), на AIManager (для отримання відповіді штучного інтелекту) та інші об’єкти, наприклад, на контролери сцен або менеджер даних.

Крім того, може бути визначено перелік станів гри (наприклад, через enum GameState), що відображає поточний контекст (діалог, екран меню, бій тощо).

На рисунку 3.5 зображено фрагмент коду що демонструє ініціалізацію глобального екземпляра класу та взаємодію з менеджерами AI та UI.

```
1  public class GameManager : MonoBehaviour
2  {
3      public static GameManager Instance { get; private set; }
4      [SerializeField] private AIManager aiManager;
5      [SerializeField] private DialogueUI dialogueUI;
6      private enum GameState { Idle, InDialogue, Transitioning }
7      private GameState currentState = GameState.Idle;
8
9      private void Awake()
10     {
11         // Реалізація синглтона: зберігаємо посилання на екземпляр
12         if (Instance == null)
13         {
14             Instance = this;
15             DontDestroyOnLoad(gameObject);
16         }
17         else
18         {
19             Destroy(gameObject);
20         }
21     }
22 }
23
24
```

Рисунок 3.5 – Реалізація шаблону Singleton у класі GameManager

В цьому блоці наведено початкову конфігурацію класу. Властивість Instance та метод Awake() забезпечують створення єдиного екземпляру GameManager. Поля aiManager і dialogueUI отримують посилання на відповідні

компоненти (налаштовуються в редакторі Unity або через код). Перелік GameState визначає можливі режими роботи, а currentState фіксує поточний стан. За допомогою станів можна координувати, чи зараз відбувається діалог, чи гра знаходиться в затримці між сценами тощо.

На рисунку 3.6 зображено фрагмент коду який демонструє використання корутин у Unity для управління перебігом діалогу та зміною ігрової сцени.

```
5
6   // Запуск нового діалогу з переломом в нову сцену
7   public void StartDialogue(string dialogueFile)
8   {
9       // Завантаження даних діалогу (наприклад, з JSON чи ScriptableObject) за іменем файлу
10      string[] lines = DialogueLoader.Load(dialogueFile);
11      StartCoroutine(RunDialogue(lines));
12  }
13
14  private IEnumerator RunDialogue(string[] lines)
15  {
16      currentState = GameState.InDialogue;
17      foreach (string line in lines)
18      {
19          // Передати поточний рядок на відображення у UI
20          dialogueUI.DisplayLine(line);
21          // Очікувати, поки користувач прочитає і натисне кнопку "Продовжити"
22          yield return new WaitUntil(() => dialogueUI.LineAdvancingRequested);
23      }
24      currentState = GameState.Idle;
25      ChangeScene("NextSceneName");
26  }
27
28  // Зміна активної сцени
29  public void ChangeScene(string sceneName)
30  {
31      StartCoroutine(TransitionToScene(sceneName));
32  }
33
34  private IEnumerator TransitionToScene(string sceneName)
35  {
36      currentState = GameState.Transitioning;
37      yield return SceneManager.LoadSceneAsync(sceneName);
38      currentState = GameState.Idle;
39  }
```

Рисунок 3.6 Реалізація сценарію діалогу з NPC та автоматичного переходу між сценами.

Далі наведено приклади методів, що демонструють логіку керування ігровим процесом. Метод StartDialogue ініціює зчитування сценарію діалогу (наприклад, з файлу чи іншого джерела даних) та запускає корутину RunDialogue.

Корутина послідовно відображає кожну репліку у діалоговому інтерфейсі через dialogueUI.DisplayLine(line) та очікує взаємодії з користувачем (метод

LineAdvancingRequested має ставати true, коли гравець просить перейти до наступного рядка).

Після завершення всіх рядків діалогів стан повертається до Idle і за потреби викликається перехід до іншої сцени за допомогою методу ChangeScene. Останній демонструє плавний перехід між сценами з встановленням проміжного стану Transitioning.

Взаємодія з іншими компонентами: GameManager є центральним «диригентом» сцени. Коли користувач обирає варіант відповіді у діалозі, DialogueUI сповіщає про це GameManager (наприклад, через виклик методу OnOptionSelected).

Далі GameManager може використати AIManager для отримання наступної фрази чи реакції персонажа: він передає вибраний користувачем контекст (або цілий промпт) до AIManager, отримує згенеровану відповідь і передає її назад DialogueUI для відображення.

При переході між сюжетними локаціями GameManager викликає методи завантаження сцен. Така взаємодія забезпечує чіткий розподіл завдань: клас GameManager відповідає за логіку та послідовність дій, тоді як інші компоненти виконують допоміжні функції (UI, AI, зберігання даних і т.д.).

3.2.3 DialogueUI –вивід реплік та варіантів відповіді у інтерфейсі користувача

Клас DialogueUI призначено для візуалізації діалогів та отримання вводу від користувача. Він відповідає за відображення поточного репліки персонажа та варіантів відповідей гравця на екрані мобільного пристрою. Компоненти цього класу взаємодіють із графічними елементами Unity UI (текстовими полями, кнопками тощо). DialogueUI отримує від GameManager або іншого контролера текст репліки і список варіантів відповіді, а також сигналізує про вибір користувача, передаючи обраний варіант назад у логіку гри. Таким чином, цей клас служить «мостом» між даними діалогів (лініями сценарію) та самим гравцем.

Структура та функціональна логіка класу DialogueUI: Клас зазвичай також є нащадком MonoBehaviour і містить посилання на Unity-компоненти інтерфейсу. Наприклад, поле Text dialogueText (або TMP_Text) для виведення основного тексту діалогу і масив кнопок Button[] optionButtons для варіантів вибору.

У методі ініціалізації (Start або Awake) можуть бути налаштовані події натискання на кнопки, які прив’язують один і той же обробник з аргументом — номером кнопки. Основні методи класу – це DisplayLine(string line) для показу окремого рядка діалогу та DisplayOptions(List<string> options) для відображення кнопок з варіантами відповіді.

Ще один важливий метод – OnOptionSelected(int index), який викликається при натисканні на кнопку і повідомляє GameManager про вибір гравця. На рисунку 3.7 наведено зразок коду з основними методами.

```
public void DisplayLine(string line)
{
    dialogueText.text = line;
    // Опційно: сховати кнопки, доки не будуть потрібні варіанти відповіді
    foreach (Button btn in optionButtons)
        btn.gameObject.SetActive(false);
}

// Показати варіанти відповіді користувача
public void DisplayOptions(List<string> options)
{
    // Активувати кнопки відповідно до кількості варіантів
    for (int i = 0; i < optionButtons.Length; i++)
    {
        if (i < options.Count)
        {
            optionButtons[i].gameObject.SetActive(true);
            optionButtons[i].GetComponentInChildren<Text>().text = options[i];
            int index = i;
            optionButtons[i].onClick.RemoveAllListeners();
            // Прив'язка методу для кожної кнопки
            optionButtons[i].onClick.AddListener(() => OnOptionSelected(index));
        }
        else
        {
            optionButtons[i].gameObject.SetActive(false);
        }
    }
}

// Метод, що викликається при натисканні на кнопку
private void OnOptionSelected(int index)
{
    // Повідомити GameManager про обраний варіант
    GameManager.Instance.ProcessPlayerChoice(index);
```

Рисунок 3.7 – Реалізація інтерфейсу вибору відповідей у діалозі.

У цьому прикладі DisplayLine оновлює текстовий елемент dialogueText, показуючи нову репліку персонажа. Після відображення репліки всі кнопки для варіантів відповідей ховаються, що запобігає їх некоректному використанню, поки не настане час вибору.

Метод DisplayOptions приймає список рядків – варіантів відповіді гравця. Він активує стільки кнопок, скільки є варіантів, задає відповідний текст кожній кнопці і обробник onClick, який викликає OnOptionSelected з індексом цього варіанту. Інші непотрібні кнопки ховаються.

Метод OnOptionSelected передає обраний індекс назад до GameManager (метод ProcessPlayerChoice – це приклад назви методу у GameManager, який слід реалізувати для обробки вибору гравця).

Таким чином, логіка взаємодії залишається розділеною: DialogueUI лише обробляє інтерфейс і ввід, а GameManager реалізує наслідок вибору.

Взаємодія з іншими компонентами: DialogueUI тісно співпрацює з GameManager. Коли гравець обирає відповідь, DialogueUI викликає відповідний метод GameManager, передаючи номер обраного варіанту.

Далі GameManager, можливо, генерує наступну репліку за допомогою AIManager або визначає наступну дію на основі сценарію.

Крім того, GameManager може викликати методи DialogueUI (DisplayLine або DisplayOptions) у відповідний час для оновлення інтерфейсу.

Із зовнішніх компонентів DialogueUI зазвичай не потребує доступу до даних – вона отримує необхідні строки і списки від логіки гри та відображає їх користувачу.

Таким чином, роль DialogueUI зводиться до ізольованої управлінської взаємодії з користувацьким інтерфейсом, що полегшує тестування та модифікацію графічного оформлення без зміни ігрової логіки.

3.2.4 NPCManager –пам’ять NPC та їхнє ставлення до головного героя

Клас NPCManager у архітектурі гри виконує функцію центрального менеджера для зберігання та обробки інформації про внутрішній стан неігрових персонажів та їхнє ставлення до гравця. Він реалізується як компонент Unity, що дозволяє приєднувати скрипт до ігрового об’єкта

. Основним завданням NPCManager є фіксація «пам’яті» NPC про попередні взаємодії та динамічне обчислення їхнього афінітету щодо головного

героя. Сучасні дослідження підкреслюють, що NPC у відеоіграх зазвичай мають обмежену пам’ять лише рамками поточної сесії, а просунуті системи дозволяють NPC запам’ятовувати важливі деталі між ігровими сесіями

Структура та функціональна логіка класу: NPCManager містить набір полів і методів для управління пам’яттю NPC. Структурно його складові можна описати так:

– Поля класу: наприклад, словник npcMemories (Dictionary<string, List<string>>) для зберігання списків подій чи реплік, які NPC “пам’ятає”, та словник npcAffinity (Dictionary<string, float>), що зберігає поточний рівень симпатії кожного NPC до головного героя. Такі словники ініціалізуються у методі Awake.

– Поля-синглтон: зазвичай клас реалізується за шаблоном Singleton, тобто має статичне поле Instance, яке забезпечує єдину глобальну інстанцію менеджера.

– Методи: наприклад, RememberEvent(string npcId, string memory) – додає новий запис до пам’яті NPC з ідентифікатором npcId; ModifyAffinity(string npcId, float delta) – змінює значення симпатії NPC на величину delta; GetAffinity(string npcId) – повертає поточну симпатію NPC (або 0, якщо дані відсутні).

Такий набір методів дозволяє іншим компонентам гри (сценаріям діалогів, скриптам подій) оновлювати стан NPC на основі дій гравця.

На рисунку 3.8 зображено фрагмент коду що демонструє використання словників для збереження взаємодій з NPC та шаблон синглтона для централізованого доступу.

Рисунок 3.8 – Реалізація менеджера NPC із пам’яттю

У цьому блоці коду оголошується клас NPCManager, який наслідує від MonoBehaviour, що надає фреймворк для прикріплення скрипту до ігрового об’єкта Unity. Статичне поле Instance реалізує шаблон Singleton для єдиної інстанції менеджера в ігровому процесі. У методі Awake перевіряється наявність існуючого екземпляра; якщо він відсутній, цей об’єкт стає єдиним.

Також тут створюються дві ключові структури даних: npcMemories для зберігання послідовності «спогадів» кожного NPC та npcAffinity для числових показників симпатії/ворожості.

Метод RememberEvent додає нову подію чи запис (рядок memory) у список спогадів NPC з ідентифікатором npcId. Якщо такого NPC ще немає в словнику, створюється новий список. Таким чином система може накопичувати контекст попередніх дій гравця, пов’язаний з кожним NPC.

На рисунку 3.9 зображено код що реалізує базову систему емоційного ставлення NPC, що є основою адаптивної поведінки персонажів у грі.

```
7   public void ModifyAffinity(string npcId, float delta)
8   {
9       // Змінюємо рівень симпатії NPC до гравця
10      if (!npcAffinity.ContainsKey(npcId))
11      {
12          npcAffinity[npcId] = 0f;
13      }
14      npcAffinity[npcId] += delta;
15  }
16  public float GetAffinity(string npcId)
17  {
18      // Повертає поточний рівень симпатії (або 0, якщо невідомий NPC)
19      return npcAffinity.ContainsKey(npcId) ? npcAffinity[npcId] : 0f;
20  }
21
22
```

Рисунок 3.9 – Методика зміни та отримання рівня симпатії NPC до гравця.

Другий блок коду демонструє методи роботи з показником симпатії (npcAffinity). Метод ModifyAffinity змінює поточний рівень симпатії NPC до героя на величину delta (може бути позитивним або негативним числом).

Якщо NPC раніше не мав запису, для нього ініціалізується нульове значення симпатії. Метод GetAffinity повертає накопичений рівень симпатії або 0, якщо NPC ще не зустрічався з гравцем. Клас слідує принципу MonoBehaviour, тому його поля та методи можна легко викликати з інших скриптів, прив’язаних до об’єктів сцени.

Взаємодія з іншими компонентами: NPCManager інтегрується у глобальну архітектуру гри. Наприклад:

– Скрипти NPC (самих персонажів) можуть звертатися до NPCManager.Instance під час діалогу або подій, щоб передавати йому нові спогади чи зміни афінітету.

– Система діалогів (DialogueSystem) використовує методи NPCManager для фіксації наслідків реплік гравця – після розмови оновлюються пам’ять NPC та їхня симпатія.

– NPCManager може враховувати параметри головного героя (через CharacterState) для змін афінітету (наприклад, героїчні чи зловісні вчинки впливають на довіру).

– Для збереження/відновлення гри NPCManager взаємодіє з SaveSystem: при збереженні гра передає стан пам’яті і симпатії, а при завантаженні – заповнює свої словники попередніми значеннями з файлу.

– Таким чином, NPCManager забезпечує неперервну поведінку персонажів у відповідності до зроблених гравцем дій, реалізуючи ідею довготривалої пам’яті NPC

3.2.5 CharacterState – характеристики і розвиток головного героя

Клас CharacterState відповідає за зберігання і оновлення ігрових атрибутів головного персонажа. Згідно з термінологією рольових ігор, атрибут (characteristic) – це показник, який описує ступінь прояву певної вродженої властивості персонажа

Основні компоненти CharacterState включають такі елементи:

– Поля: наприклад, level, experience, health, maxHealth, а також словник основних атрибутів.

– Поля-синглтон: часто CharacterState теж робиться за шаблоном Singleton (static CharacterState Instance), щоб інші компоненти могли легко отримати доступ до стану героя.

– Методи. TakeDamage(int amount) – зменшує здоров’я героя на amount (не менше нуля).

– Heal(int amount) – збільшує здоров’я героя (не вище за maxHealth).

– AddExperience(int xp) – додає досвід; якщо накопичений досвід перевищує поріг, викликає LevelUp().

– LevelUp() – збільшує рівень героя, оновлює параметри (наприклад, відновлює здоров’я та збільшує інші атрибути).

Ця логіка забезпечує «прокачування» героя: при накопиченні достатнього досвіду персонаж підвищує рівень, що супроводжується поліпшенням характеристик.

У першому блоці коду оголошується клас CharacterState, який наслідує від MonoBehaviour і містить базові властивості героя: level, experience, health, а також інші атрибути. Як і NPCManager, він робить себе Singleton через статичне поле Instance. У методі Awake відбувається ініціалізація єдиного екземпляра та початкових значень.

На рисунку 3.10 зображено реалізацію шаблона Singleton для глобального доступу до характеристик гравця та встановлює стартові значення здоров'я, сили й інтелекту

```
public class CharacterState : MonoBehaviour
{
    public static CharacterState Instance;
    public int level = 1;
    public int experience = 0;
    public int health;
    public int maxHealth = 100;
    // Інші атрибути
    public int strength;
    public int intelligence;


    void Awake()
    {
        // Ініціалізація Singleton
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
        // Початкові значення характеристик
        health = maxHealth;
        strength = 10;
        intelligence = 10;
    }
}
```

Рисунок 3.10 – Клас CharacterState для зберігання та ініціалізації атрибутів персонажа.

Другий блок коду демонструє управління досвідом і підвищенням рівня. Метод AddExperience збільшує змінну experience і перевіряє, чи перевищила вона поточний поріг (xpToLevel).

Якщо так, викликається LevelUp(), де підвищується рівень (level++) та збільшуються базові характеристики героя (зокрема, maxHealth і інші атрибути). Поточне здоров'я оновлюється до нового максимуму.

Ця функціональність реалізує механіку розвитку персонажа: зростання рівня супроводжується суттєвим поліпшенням його здібностей. Такий підхід відповідає загальноприйнятим принципам РПГ: базові характеристики визначають початкові «таланти» персонажа, а накопичення досвіду приводить до їх зміни на краще.

3.2.6 SaveSystem –механізм збереження і відновлення даних гри

Клас SaveSystem відповідає за збереження ігрового стану на диск та відновлення його при наступному запуску гри. Він може бути реалізований як статичний клас або компонент Unity, що забезпечує збереження складних даних (інформації з CharacterState, NPCManager тощо) у файл за межами пам’яті сцени. За рекомендаціями Unity, дані слід зберігати у спеціальній директорії, доступній на всіх платформах – Application.persistentDataPath.

Ця директорія гарантує, що файли не видаляться при оновленні гри. Крім того, у Unity більшість класів можна серіалізувати для запису на диск, тому SaveSystem перетворює внутрішні об’єкти гри в примітивні дані (наприклад, JSON) і записує їх у файл.

Хоча для окремих значень існує механізм PlayerPrefs, він менш гнучкий для комплексних структур даних. Тому в нашому проекті SaveSystem реалізовано ручно: дані про стан герою, NPC та інші необхідні змінні зберігаються у єдиному контейнері.

Структура та функціональна логіка класу:

– Константи та поля: наприклад, константа fileName із назвою файлу збереження або поле filePath (зазвичай Application.persistentDataPath + "/savefile.json").

– Метод SaveGame(): відповідальний за збирання даних із CharacterState, NPCManager та інших класів, перетворення їх у серіалізований формат (наприклад, з використанням JsonUtility.ToJson) і запис результату у файл.

– Метод LoadGame(): перевіряє наявність файлу збереження, зчитує його вміст, десеріалізує дані і передає ці значення назад в CharacterState та NPCManager.

На рисунку 3.11 Клас SaveSystem формує структуру збереження ігрових даних і серіалізує її у файл savefile.json за допомогою JsonUtility.

```
0 references
public static class SaveSystem
{
    private static string filePath = Application.persistentDataPath + "/savefile.json";

    0 references
    public static void SaveGame()
    {
        // Формування об'єкта даних для серіалізації
        SaveData data = new SaveData(CharacterState.Instance, NPCManager.Instance);
        string json = JsonUtility.ToJson(data);
        // Запис у файл
        File.WriteAllText(filePath, json);
    }
}
```

Рисунок 3.11 Збереження стану гри у форматі JSON у Unity.

У цьому фрагменті SaveGame створює структуру (клас SaveData) з актуальними даними про стан героя та NPC (отриманими, наприклад, із Instance цих класів). Потім вона перетворює цей об’єкт у JSON-рядок через JsonUtility.ToJson і записує його у файл за шляхом Application.persistentDataPath.

Метод LoadGame перевіряє існування файлу збереження. Якщо файл знайдено, його вміст читається та перетворюється з JSON у об’єкт SaveData. Потім викликаються допоміжні методи (наприклад, RestoreFromData) у CharacterState і NPCManager, які оновлюють їх внутрішній стан відповідно до збережених значень. Таким чином відбувається повне відновлення попередньої сесії гри.

Взаємодія з іншими компонентами: SaveSystem тісно взаємодіє з усіма класами, чиї дані потрібно зберігати. Зокрема:

– Він використовує Application.persistentDataPath як базовий шлях для файлів, що забезпечує коректну роботу на всіх платформах.

– При виклику збереження (SaveGame) інші класи (CharacterState, NPCManager тощо) передають у SaveSystem свій поточний стан (або зчитують його безпосередньо через Singleton) для запису.

– При завантаженні (LoadGame) SaveSystem роздає інформацію назад цим класам, відновлюючи їхні поля.

– Важливо, що SaveSystem призначений для зберігання комплексних даних, а не лише простих змінних. У випадках, коли потрібно зберегти одне-дві прості настройки (наприклад, гучність звуку або останній пройдений рівень), можна було б застосувати PlayerPrefs.

Таким чином, клас SaveSystem забезпечує надійне перенесення стану гри у файл та назад, гарантуючи, що користувач завжди зможе відновити свій прогрес. Послідовне збереження та завантаження даних через централізований механізм дає змогу зберігати консистентність між усіма елементами гри і відповідає рекомендаціям щодо роботи з постійними даними Unity

3.3 Оптимізація та тестування застосунку з інтеграцією мовної моделі

Тестування передбачає оцінювання коректності та релевантності відповідей моделі, надійності викликів API, адаптивності моделі до різних запитів, а також аналіз затримок відповіді та потенційних збоїв.

З метою забезпечення повноти перевірок застосовано різні види тестування: функціональне, інтеграційне, регресійне та перфоманс-тестування. Зокрема, інтеграційні тести фокусувались на підтвердженні коректної взаємодії з API моделі: перевірялись обробка стандартних і граничних запитів, облік помилок і відновлення після них.

Для оцінки якості відповідей здійснювали перевірку їх достовірності та відповідності очікуванням (response accuracy) – згідно з рекомендаціями, точність і актуальність відповідей є ключовими показниками якості чат-ботів.

При тестуванні діалогів особливу увагу звертали на збереження контексту й узгодженість бесіди. Це включало введення моделі у різні стани шляхом модифікації попередніх повідомлень і оцінювання, чи модель пам’ятає

попередню інформацію і продовжує діалог логічно (на кшталт «збереження інтелекту» системи).

Адаптивність відповідей перевірялась через різноманітні варіанти формулювань користувацьких запитів: тестувалися сінонімічні запити та зміна порядку слів, щоб пересвідчитися, що модель надає стабільно релевантні відповіді. Окрім того, оцінювалась «гнучкість» моделі – її здатність підлаштовуватися під змінений контекст або інструкції, не генеруючи зайвих або протирічних повідомлень.

Для оцінки продуктивності системи вимірювали затримки (латентність) відповіді моделі при різних режимах роботи: стандартні діалоги, навантаження багаторазовими запитами тощо. Згідно з практикою перфоманс-тестування чат-ботів, вимірювались середній час відповіді й максимальний час під навантаженням.

Такий аналіз дозволив виявити потенційні затримки при спілкуванні з API (наприклад, затримки через неточне тайм-аути чи перевантаження серверів). Зі звіту тестування також проводився аналіз дублювання відповідей: перевірялось, що модель у діалозі не генерує повторюваних фраз без підстав (що могло б свідчити про помилкове оброблення контексту).

Також аналізувалися випадки відхилення (hallucination) – коли модель давала невідповідні або неточні відповіді. За необхідності проводились додаткові перевірки з «fallback»-запитами (запитами-запасами), щоб оцінити, як система реагує на непередбачувані ввідні дані чи системні збої.

Таким чином, поєднання автоматизованих і ручних тестів дало змогу всебічно оцінити якість інтеграції мовної моделі із застосунком і забезпечити основні характеристики роботи (точність, адаптивність, швидкодію).

3.3.1 Навчання мовної моделі

У межах цього проєкту під «навчанням» мовної моделі не розуміється пряме перенавчання параметрів, а радше налаштування її поведінки через формування запитів (промптів), добір контексту та форматування ввідних даних.

Так званий промпт-інжиніринг – це «стратегічний процес планування та генерації промптів для отримання бажаної відповіді».

На практиці це означає ретельний вибір системних та користувацьких інструкцій, які передаються до моделі перед кожним запитом. Наприклад, спеціальне системне повідомлення (system prompt) може задавати тон розмови чи обмежувати поле відповідей моделі, забезпечуючи стабільність результатів.

Прикладом адаптації є надання моделі необхідного контексту: вхідні дані включають історію діалогу або релевантні параметри (наприклад, роль користувача, попередньо задані відповіді чи ключову інформацію).

Як показує досвід, «додавання релевантного контексту в промпт допомагає ChatGPT надавати більш обгрунтовані відповіді». У проєкті обирались оптимальні фрагменти інформації з наявних даних, які подавалися разом із запитом, щоб мовна модель орієнтувалася на потрібну тему. Також тестувалося форматування запитів: наприклад, поділ запиту на допоміжні питання чи запровадження шаблонів (інструкцій), які найкраще спонукали модель дотримуватися заданого стилю чи структури відповіді.

Серед застосованих методів також – експеримент з різними стилями формулювання запиту (наприклад, пряма команда чи запитання, включення прикладів («few-shot prompting») у контексті). Важливо відзначити, що всі ці підходи не змінюють внутрішню модель, а лише спрямовують її поведінку через вхідні дані і промпти. Такий підхід дозволяє адаптувати поведінку готової моделі до специфіки застосунку без витрат на довготривале донавчання чи доступ до внутрішніх параметрів.

3.3.2 Оптимізація коду для Android

Для мобільної версії додатка виконано комплексну оптимізацію коду та налаштувань Unity. Для збірки використано IL2CPP – механізм перетворення IL-коду на C++ з подальшою компіляцією у нативний код.

Як зазначено в документації Unity, IL2CPP дозволяє «покращувати продуктивність на різних платформах», оскільки результуючий нативний код оптимізується компілятором для цільової архітектури. Хоча такий підхід

збільшує час та розмір збірки, він часто дає переваги в часі виконання та енергоспоживанні на мобільних пристроях.

Також приділено увагу управлінню пам’яттю. Використано Unity Profiler і Memory Profiler для виявлення витоків і «гарячих точок» у використанні пам’яті. Оптимізації включали обмеження кількості одночасно завантажених об’єктів, звільнення непотрібних ресурсів (destroy і release) одразу після використання, а також застосування Addressable Assets для динамічного завантаження великих активів. Система Addressables забезпечує автоматичне звільнення пам’яті, коли об’єкт більше не потрібний, що дозволяє уникнути «витоків. Також здійснено звільнення усіх необов’язкових об’єктів через виклики Resources.UnloadUnusedAssets у відповідний момент, хоча слід враховувати, що ця операція коштує ресурсів і застосовується перед етапом безвідмовного інтерфейсу.

Всі важкі операції реалізовано асинхронно, щоб не блокувати головний потік інтерфейсу. У Unity для цього використовуються корутини та async/await, що дозволяють виконувати запит на бек-енд без затримок інтерфейсу користувача. Це забезпечує плавний користувацький досвід: обробка відповіді моделі виконується у фоновому режимі, а між тим інтерфейс продовжує реагувати на дії користувача.

Оптимізація також торкнулась налаштувань збірки: було відключено непотрібні компоненти Unity та служби. Зокрема, у Player Settings вимкнено Development Build та Debugging, що прибирає служби ретрансляції профілів і інших відлагоджувальних даних. У налаштуваннях Graphics вимкнено зайві ефекти (наприклад, real-time shadows, високий рівень тесселяції), якщо вони не потрібні для цільового функціоналу. У разі використання бібліотек чи пакетів Unity, які не потрібні у фінальній версії, вони видалені або відключені Для керування ресурсами і мінімізації розміру білд-компонент застосовано Addressable Asset System. Це дозволило уникнути включення багатьох активів у збірку наперед, завантажуючи їх динамічно за необхідності.

3.4 Висновок до третього розділу

У межах проведеної роботи було реалізовано комплексну інтеграцію мовної моделі штучного інтелекту у мобільний застосунок типу візуальної новели, з подальшим її тестуванням і оптимізацією під Android-платформу. Тестування взаємодії з мовною моделлю охоплювало аналіз якості генерованих відповідей, здатність до адаптації під різні формулювання запитів, стійкість до помилок та стабільність затримок. У процесі перевірки було підтверджено високу відповідність відповідей очікуваному змісту, що забезпечило природність діалогів і збереження логіки сюжетної лінії.

Уточнення підходу до «навчання» моделі дозволило уникнути помилкового трактування: замість перенавчання було реалізовано адаптацію через промпт-інжиніринг, тобто оптимізацію структури і змісту запитів до моделі. Такий підхід дав змогу спрямовувати поведінку ШІ без змін архітектури самої мовної моделі, що є ефективним рішенням для мобільних застосунків.

Оптимізація продуктивності гри для Android охоплювала декілька рівнів: компіляцію проєкту з використанням IL2CPP для підвищення продуктивності виконання, мінімізацію використання оперативної пам’яті, впровадження асинхронної обробки запитів та оптимізацію ресурсів через Addressable Asset System. Застосування Unity Profiler і Memory Profiler дозволило виявити критичні точки навантаження, зменшити частоту GC-збирання та забезпечити плавну роботу застосунку на мобільних пристроях.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Вплив електромагнітного випромінювання від мобільних телефонів

В сьогоденних реаліях майже не можливо уявити своє життя без використання мобільного телефону. Завдяки ним ми завжди на зв'язку з рідними і близькими, дізнаємося останні новини світу або ж розслабляємося від тяжкого робочого дня граючи у мобільні ігри. Але багато користувачів не звертають уваги наскільки даний пристрій є потенційно небезпечним, і яких правили користування мобільними телефонами варто слідувати аби запобігти їхньому негативному впливу на організм.

Сучасні мобільні пристрої (мобільні телефони, смартфони, планшети) працюють у широкому діапазоні радіочастотних електромагнітних хвиль, переважно 900–2100 МГц для мобільного зв'язку, Wi-Fi – 2,4–5 ГГц.

Ці пристрої генерують електромагнітне випромінювання (ЕМВ) у радіочастотному (RF) діапазоні, яке впливає на організм користувача. За законодавством України, охорона праці передбачає контроль за допустимими рівнями такого випромінювання, оскільки перевищення норм може призводити до температурного або неприємного впливу на здоров'я.

Електромагнітне випромінювання (ЕМВ) – це форма енергії, що поширюється у вигляді хвиль електричного та магнітного поля. Мобільні телефони випромінюють електромагнітне випромінювання високої частоти, відоме як радіочастотне випромінювання. Це випромінювання генерується антеною телефону та використовується для передачі і отримання сигналів зі стільникових мереж.

Електромагнітне випромінювання утворює електромагнітне поле, скорочено ЕМП, яке може впливати на організм людини залежно від різних факторів. Серед цих факторів належить зазначити характер поля, яке створюється на робочому місці, чи то високочастотне (ВЧ) чи низькочастотне

(НВЧ), відстань від джерела випромінювання, тривалість дії, діапазон частот, інтенсивність випромінювання.

Для оцінки впливу опромінення ВЧ та НВЧ використовується величина інтенсивності електричного поля, яка вимірюється в вольтах на метр (В/м).

Електромагнітні поля (ЕМП) поділяють на іонізуючі (хвилі із дуже високою енергією, що можуть іонізувати матерію) та неіонізуючі.

Мобільні пристрої генерують неіонізуюче НВЧ-випромінювання. Попри те, що така енергія не руйнує безпосередньо атомні структури тканин, її тепловий ефект та довготривала дія можуть впливати на нервову, ендокринну систему та потенційно підвищувати ризик окремих захворювань (на думку деяких дослідників, можливий розвиток пухлин, головного болю, втоми, зниження концентрації).

Офіційні дослідження (ВООЗ) станом на 2025 р. не виявили переконливих доказів канцерогенного впливу звичайного користування мобільними телефонами, але питання досліджується далі.

В Україні встановлено гранично допустимі рівні електромагнітного випромінювання (ГДР) для населення та працівників.

Згідно з Державними санітарними нормами і правилами захисту населення від ЕМВ (ДСанПіН 239-96), для мобільних базових станцій (на вісях стільникового зв’язку) інтенсивність поля не повинна перевищувати 2,5 мкВт/см² на житловій території.

Максимальний фактичний показник випромінювання від мобільних телефонів на сьогодні зазвичай значно нижчий (у США, Японії, Канаді стандарти дозволяють до 1000 мкВт/см²), а міжнародні рекомендації ВООЗ – до 450 мкВт/см². В Україні стандарт на вихідну потужність пристроїв значно стриманіший (до 10–15 мкВт/см² на 2017 рік).

Основні ефекти НВЧ-випромінювання від мобільних пристроїв пояснюють нагрівальним (тепловим) ефектом тканин та потенційними нефізіологічними реакціями. Істотно залежить від потужності сигналу і часу експозиції. У режимі активного дзвінка або передачі даних інтенсивність ЕМП в зоні вуха користувача може бути найвищою. Тривале перебування далеко від

базової станції вимагатиме потужнішого випромінювання телефону, тоді як поблизу до станції – навпаки, інтенсивність падає. Обмеження ССБТ (системи стандартів безпеки праці) та САНПіН компенсують можливі ризики.

Поблизу джерел ВЧ полів формуються зони індукції і зони випромінювання, які розповсюджуються на різну відстань залежно від частоти.

На відстані, меншій ніж 1/6 довжини хвилі, переважають поля індукції і цю область можна вважати зоною індукції. На більш віддалених відстанях переважають поля випромінювання, і ця область вважається зоною випромінювання.

У зоні індукції людина піддається періодичному змінному електричному та магнітному полю. У зоні випромінювання людину оточує електромагнітне поле з однаковою та одночасно змінною електричною та магнітною компонентами.

ЕМП впливає на організм людини шляхом часткового поглинання енергії поля тканинами тіла. Високочастотні поля ВЧ ЕМП збуджують іони в тканинах, що призводить до виникнення високочастотних струмів і теплового ефекту поглинання енергії поля.

Провідність тканин пропорційна кількості рідини, що міститься в них. Кров та м'язи мають найбільшу провідність, а жирові тканини - найменшу. Товщина жирового шару в опроміненій ділянці тіла впливає на відбивання хвиль від поверхні тіла людини. Головний і спинний мозок мають незначний жировий шар, а очі взагалі не мають жиру, тому вони найбільше піддаються опроміненню.

Тривале та систематичне вплив ЕМП на працюючих може викликати зміни в організмі, такі як головний біль, порушення сну, підвищена втомленість, роздратованість, понижений кров'яний тиск, зміни в печінці та селезінці, підвищена температура тіла, випадіння волосся та ламкість нігтів.

У залежності від ступеня впливу ЕМП розрізняються 3 ступені ураження:

I. Легке ураження - характеризується тимчасовими функціональними змінами в організмі, які не вимагають тривалого лікування і не впливають на працездатність потерпілого.

II. Ураження середньої ступені - характеризується вираженими та стійкими порушеннями нервової системи, ендокринної системи та кровообігу.

III. Важкі ураження - рівень ураження, який не згадується в літературі, оскільки виробничі підприємства не допускають працювати людей, які мають навіть легке або середнє ураження.

Законодавство передбачає проведення попередніх та періодичних медичних оглядів, а також відбір працівників, які працюють з ЕМП, для забезпечення профілактики професійних захворювань [28].

Враховуючи те що даний пристрій перебуває постійно поряд із тілом людини, вплив електромагнітного випромінювання хоч і не великий, але має накопичувальну дію і може дати свої «результати» у майбутньому [5].

Аби мінімізувати ризики шкоди здоров'ю варто дотримуватися простих але важливих правил використання мобільних телефонів, а саме:

– Використовуйте безпровідні навушники. Це дозволить збільшити відстань між телефоном і головою, знизивши таким чином вплив електромагнітного поля на мозок та мінімізуючи можливу теплову експозицію.

– Використовуйте вбудовані функції зменшення випромінювання. Більшість сучасних мобільних телефонів мають функції зменшення випромінювання, такі як «режим польоту» або «економія енергії». Вони можуть допомогти знизити радіочастотне випромінювання, коли ви не активно використовуєте свій телефон.

– Уникайте носіння телефону біля тіла. Намагайтеся тримати телефон подалі від тіла, особливо в кишені або прикріплений до пояса. Чим ближче телефон до вас, тим більша експозиція електромагнітному випромінюванню.

– Багато виробників мобільних телефонів надають рекомендації щодо використання їхніх пристроїв, щоб мінімізувати можливий вплив на здоров'я.

Закон України «Про охорону праці» від 14.10.1992 №2694-XII наголошує, що роботодавець зобов'язаний забезпечити працівникам безпечні умови праці, у тому числі щодо впливу випромінювань

Узагалі, більшість досліджень показують, що рівень електромагнітного випромінювання від мобільних телефонів на даний момент знаходиться в межах

безпечних норм. Однак, важливо продовжувати дослідження та моніторинг, щоб забезпечити безпеку користування мобільними телефонами в майбутньому.

Мобільні пристрої генерують електромагнітні поля, які згідно з національними нормативами (ДСанПіН 239-96) мають обмеження по гранично допустимому рівню. Дослідження не виявили однозначно шкідливих наслідків від звичайного користування смартфонами, але в контексті охорони праці важливо дотримуватися профілактичних заходів. Працівник повинен отримувати інструктаж щодо користування мобільною технікою (за ГОСТ 12.0.004 або іншими нормами ССБТ), а роботодавець – забезпечити можливості для перерв. Загальні рекомендації (використання гарнітур, перерви, дотримання відстані) забезпечують дотримання принципу «допустимого ризику». Таким чином, безпечне користування мобільними пристроями включає як технічні засоби захисту, так і організаційні (графік роботи, інструктаж, медичні огляди)

4.2 Заходи безпечної роботи з сенсорними пристроями

Сенсорні пристрої – це гаджети з екраном, чутливим до дотику: смартфони, планшети, ноутбуки з сенсорним екраном, інтерактивні панелі тощо. Вони активно використовуються як на робочих місцях, так і в побуті. Питання охорони праці при роботі з сенсорними екранами полягає у запобіганні шкідливих чинників, пов’язаних з екранною технікою: напруженням очей, м’язовим перевтомленням, електромагнітним впливом, а також психологічним дискомфортом. У відповідності до державних стандартів України та санітарних норм, будь-яке робоче місце з монітором чи планшетом має бути організоване з урахуванням ергономіки і безпечних технологій.

Робота з сенсорними пристроями поєднує дію зорового навантаження і тактильної активності рук.

При користуванні сенсорними пристроями необхідно дотримуватися заходів, які включають у собі

1. Зменшення зорового навантаження через яскравий екран високої контрастності може виникати перевтома очей (синдром «сухого ока», порушення акомодації), особливо за поганого освітлення.

Мінімізувати це допомагають налаштування кольору (теплий відтінок), достатня частота оновлення екрану (>60–75 Гц) та контраст не менше 300:1. Екран сенсорного планшета часто розташовано ближче до очей ніж монітор ПК, тому вимоги до ближньої роботи з екраном суворіші: відстань від очей до екрану не менше 35–40 см (Див. рисунок. 4.1), з можливістю регулювання нахилу екрану.

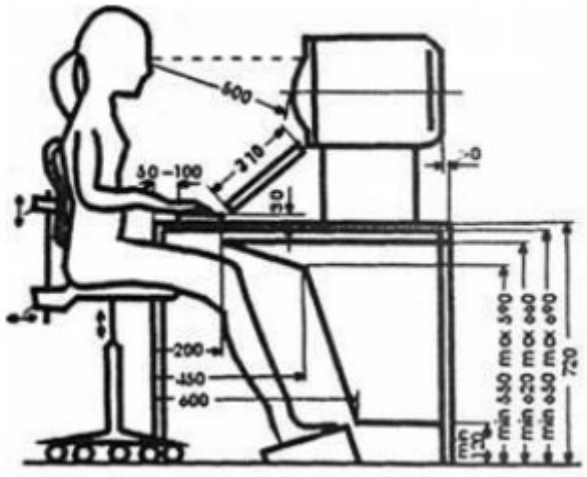

Рис. 4.1 Схематичне зображення вимог для використання комп’ютерних приладів

2. Розташування екрану повинно бути так, щоб відблиски від вікна чи джерел світла не падали на нього.

Вертикальне освітлення приміщення – 300–500 лк (використовувати затемнюючі фіранки, світильники з матовими розсіювачами). Яскравість екрану потрібно зменшити так, щоб білий фон не засліплював.

Оскільки тч-скріни повинні бути чистими, необхідно регулярно протирати їх м’якою серветкою (попередньо вимкнувши пристрій) для зниження навантаження на очі та підтримання чіткої картинки.

3. Дотримуватися егрономіки тіла при роботі з планшетом або телефонам корпус пристрою часто обмежує правильну поставу. Руки і плечі швидше втомлюються від статичного тримання пристрою.

Працювати потрібно або сидячи за столом, тримати планшет або телефон на рівні поясу, фіксувати передпліччя на опорі; надавати перевагу використанню стилуса або гарнітури (головним чином мікрофон) для введення тексту. При тривалій роботі використовувати підставку для планшета, щоб не тримати його у руках.

4. Проводити невелику гімнастику 15 хвилин кожну годину – оскільки робота із сенсорним екраном зазвичай триваліша за телефонний дзвінок, необхідно регулярно робити перерви для відпочинку очей і розминки рук.

Після кожної 15–20 хвилин концентрованої роботи відводити погляд у далечінь, а також робити вправи для зменшення м’язового напруження кистей і плечей (розминка шиї, рук, пальців). Таким чином дотримуються вимог до режиму праці з відеоекранною технікою (ДСанПіН 3.3.2.007-98).

5. Для профіліктики інфекцій проводити антибактеріальні заходи - сенсорний екран безпосередньо контактує з пальцями користувача, тому він накопичує шкідливі мікроорганізми. Потрібно регулярно протирати поверхню дезінфекційними серветками (нейтральними до покриття екрану) і мити руки до та після інтенсивного використання.

6. Уникати перегріву гаджета сенсорні пристрої містять літієві акумулятори, здатні перегріватися. Потрібно обов’язково уникати перегріву гаджета (не залишати на сонці, не закривати тканиною), а в разі нагрівання припинити роботу, дати охолонути.

Використовувати тільки зарядні пристрої виробника – несправний блок може створити загрозу ураження струмом або пускання іскри.

ДСанПіН 3.3.2.007-98 регламентують ергономічні і санітарні вимоги для працюючих з екранами, які слід застосовувати і до планшетів/телефонів, якщо робота виконується тривалий час (особливо в офісах і навчальних закладах).

Ці вимоги включають нормований режим праці, у якому після 5–10 хвилин роботи з екранами слідує хвилина короткого відпочинку для очей, та щогодини – 10-хвилинна пауза.

4.3 Безпечне використання пристроїв виведення інформації

Пристрої виведення інформації охоплюють монітори комп'ютерів, проєктори, принтери, плазмові панелі тощо. З погляду охорони праці, головними факторами небезпеки при їх використанні є: електромагнітне та оптичне випромінювання екранів, небезпечні матеріали (тонер принтерів), шум (деякі принтери та вентиляція), а також ергономічні чинники (позиція екрану, бликове освітлення).

За Державними санітарними правилами, монітори відносять до джерел фізичних факторів: радіаційного (CRT-монітори випромінюють слабке рентгенівське випромінювання, що підпадає під норми ДСТУ) та електромагнітного (електромагнітні поля від панелей високої напруги). Безпека електронно-променевих трубок (ЕПТ) моніторів регламентується ДСТУ EN 61965:2022, що встановлює допустимі рівні випромінювання.

Сучасні рідкокристалічні (LCD/LED) монітори такої небезпеки вже не несуть, але належить контролювати, аби загальний рівень електромагнітного фону на робочому місці відповідав нормам.

Для безпечного використання пристроїв виведення інформації слід організувати робоче місце відповідно до вимог ДСТУ і ДСанПіН:

- Потрібно встановити монітор на відстані 50–70 см від очей (з урахуванням діагоналі екрана). Висота верхнього краю екрану має бути приблизно на рівні очей. Кут огляду – близько 10–20° вниз від горизонталі. Якщо монітор регульований, його нахил слід встановлювати так, щоб уникнути відблисків від ламп і вікон.
- Екрани не повинні освітлюватися безпосередньо лампами або сонцем – усі джерела світла розташовують позаду (не направляють світло в очі).
- Загальна освітленість приміщення – 300–500 лк при рівномірному розподілі. У таблицях освітленості (ДСанПіН 3.3.2.007-98) наведено, що на

робочому столі ближче до монітора рівень світла може бути дещо нижчим (200–300 лк), а на клавіатурі – 300–500 лк.

– Екранам з ЕПТ потрібно підтримувати частоту вертикальної розгортки не менше 85 Гц для зменшення втоми очей. Нові LCD/LED панелі мають статичну картинку, мерехтіння в них мінімальне, але потрібно стежити за автоматичним підстроюванням яскравості. Сучасні стандарти (ДСанПіН 3.3.2.106-96 та інші) акцентують увагу на підтримці частоти не нижче 75–100 Гц у будь-якому режимі.

– Позбутися застарілого обладнна за для уникання щоденного ураження слабким рентгенівським випромінюванням (через високовольтний відліск промені). Сучасні вимоги (ДСТУ EN 61331-1:2022) регламентують, що потужність поглиненої дози рентгенівського випромінювання біля фронтального скла не повинна перевищувати 0,5 мкР/год.

– Роздільна здатність екрану повинна бути налаштована відповідно до фізичних розмірів екрана (щоб символи займали приблизно 5–7 мм в висоту). Оптимальним вважається 800×600 чи 1024×768 пікселів на діагоналі 15–17 дюймів, з додатковим масштабуванням шрифту за необхідності. Недотримання цих вимог може викликати перевтому очей.

– Кожні 3–6 місяці слід проводити очистку пристроїв від пилу. Принтери лазерного типу виділяють дрібнодисперсний тонер і озон, тому за правилом ДСанПіН 3.3.2.137-2007 їх потрібно встановлювати у добре вентильованих приміщеннях або окремих кабінетах.

Пристрої виведення інформації нерідко є складовою навчальних або презентаційних систем. Згідно з положеннями ДСТУ, для аудиторії необхідно передбачати зручність сприйняття: екран для проєктора повинен бути матовим; сидячі люди мають бачити картинку зверху вниз на кут до 30°; принтер, сканер та інші периферійні пристрої мають бути розміщені так, аби не створювати перешкод для руху персоналу.

У нормативах (ДСанПіН 3.3.2.007-98) є рекомендації щодо розрахунку мінімальної площі приміщення на одну людину в комп’ютерному класі – 4–6 м²,

що також зумовлено вимогою розмістити техніку з урахуванням безпечних відстаней.

Дотримання вимог до влаштування та використання пристроїв виведення інформації критично важливе для охорони здоров'я працівників. Правильна організація робочого місця (ергономіка, освітленість), регулярні перерви та технічні заходи (якісні монітори, підтримання обладнання в чистоті) мінімізують вплив фізичних факторів. Нормативно-правові акти (ДСанПіН, ДСТУ) передбачають такі заходи для захисту працівників від несприятливих чинників при роботі з моніторами і проєкторами. Застосування цих стандартів забезпечує економічно оправдані і науково обґрунтовані умови праці, що захищають очі, слух та здоров’я персоналу.

4.4 Електробезпека комп’ютерного обладнання

Комп’ютерне обладнання живиться від електричної мережі і містить частини під напругою. Невиконання правил електробезпеки може призводити до ураження людей електричним струмом, а також до пожежі.

Можуть виникати такі проблеми пошкоджені ізоляції кабелів, відсутність заземлення, неправильно підключені пристрої, волога на робочому місці, перевантаження мережі.

Тому для забезпечення безпеки працівників необхідно чітко дотримуватися правил регламентованих у ПУЕ 2018 року.

За ССБТ, устаткування класифікується за степенем електробезпеки:

– Клас I (заземлені) корпус пристрою з’єднаний з заземлювальним проводом через триконтактну вилку.

Користуватися тільки справними розетками з заземленням. Заземлення в мережі 220 В виконується за допомогою третього проводу (зелено-жовтий) – він обов’язково під’єднується до металевого корпусу обладнання.

– Клас II (подвійна ізоляція) Пристрої без металевого корпусу або корпус із діелектричним покриттям.

Нерідко застосовують окремий лінійний фільтр-сепаратор або стабілізатор напруги (це не нормативно-обов’язково, але дуже рекомендовано для захисту від стрибків напруги, які можуть вивести з ладу живлення ПК).

Причиною нещасних випадків часто буває перевантаження побутової електромережі або неправильний саморобний поділ ланцюга. Забороняється: об’єднувати в одній мережі побутові та промислові пристрої без призначених для цього пристроїв захисту; підключати компресори, побутові чайники, інші високопотужні споживачі до того ж подовжувача, що й ПК; ставити ноутбук на килим чи ліжко під час зарядки (може закорочуватися батарея).

Замість подовжувачів для розеток рекомендується використовувати щиток з автоматами, диференційними реле (УЗО) та заземлювальним клемником. Вимоги ДБН та Правил з експлуатації ЕУ (ПУЕ) зобов’язують на кожному комп’ютерному робочому місці мати виділене коло із запобіжником 10–16 А, та обов’язковим заземленням.

Для безпеки працівників необхідно використовувати такі засоби:

1. Стабілізовані джерела живлення та обмежувачі реле. Для серверів і вимірювальної техніки застосовувати UPS, який при вимкненні живлення дає змогу безпечно зберегти дані.

2. Запобіжники та засоби захисту. У приміщеннях з комп’ютерами рекомендується мати вхідні АВМ (автоматичні вимикачі) з диференційними реле (16 А, характеристика С) і пристрої захисту від перенапруг (розрядники). За відмову цих систем відповідає керівник працівників і електрик.

Слід неухильно дотримуватися техніки безпеки під час монтажних/ремонтних робіт: відключати живлення перед обслуговуванням, не знімати захисні кожухи з трансформаторів джерел живлення ПК, не торкатися провідників, якщо пристрій увімкнено. В офісах не допускається протирання блоків живлення або розеток вологою ганчіркою під напругою. Норма ПБЕЕ – повна відмова від ремонту устаткування під напругою, якщо це можливо зробити після вимкнення. Також не можна використовувати вологі руки для підключення кабелів, не допускаються мокрі підлоги чи контакт рідини з блоком.

Працівники зобов’язані володіти вмінями першої медичної допомоги при ураженні струмом.

Електробезпека комп’ютерного обладнання передбачає систематичну перевірку електромережі, заземлення та ізоляції всього обладнання. Всі працівники повинні проходити навчання (відповідно до ПБЕЕ, НПАОП та ДСТУ 4502:2006 «Безпека праці. Основні нормативно-правові акти.») і отримувати інструктаж з правильного використання електроприладів та дотримання правил ПУЕ.

Виконання вимог до електробезпеки (заземлення, наявність УЗО, герметичність корпусів та справність проводки) гарантує мінімізацію ризику ураження електричним струмом і забезпечує безпечну роботу з комп’ютерною технікою

4.5 Підвищення стійкості роботи об’єктів приладобудівної галузі у воєнний час

В умовах повномасштабного воєнного конфлікту підприємство приладобудівної галузі піддається численним загрозам військового та надзвичайного характеру.

Серед найбільш реальних факторів загроз – ракетні та авіаудари (включаючи балістичні та крилаті ракети), артилерійський обстріл, вибухи снарядів і мін, диверсії (саботажі) та кібератаки. Ці вражаючі дії викликають ударні хвилі, осколки, пожежі, хімічне та радіологічне забруднення, що вражають будівлі, обладнання та людей.

Ураховуючи рекомендації ДСНС, при плануванні заходів цивільного захисту підприємства до обов’язкових слід віднести захист працівників від уламків і ударної хвилі вибухових пристроїв.

Наприклад, на Ізюмському приладобудівному заводі (штат ≈1000 осіб) в офісному корпусі обладнано протиракетний бомбосховище для персоналу.

Для забезпечення стійкості роботи підприємства необхідно ідентифікувати критичні підсистеми: електропостачання, водопостачання, інформаційно-комунікаційні мережі, транспортні та технологічні вузли, що забезпечують роботу устаткування.

Згідно з ДСТУ EN ISO 22301:2021, система управління безперервністю бізнесу передбачає оцінку ризиків для основних ресурсів підприємства (енергія, вода, інформація тощо) і запровадження механізмів резервування.

Так, резервні дизель-генератори та джерела безперебійного живлення мають забезпечувати автономне електропостачання принаймні 24–72 години до приходу основних аварійно-відновлювальних бригад.

Наприклад, якщо споживана потужність цеху складає 2 МВт, за формулою приблизного розрахунку витрати пального: $\mathrm{V}_{\text{палива}} = \mathrm{P}_{\text{навантаження}} \cdot \mathrm{k}_f \cdot \mathrm{t}$,

де $k_f \approx \frac{0{,}2}{\text{кВт·год}}$ – питомий витрата дизеля, на 72-годинний резерв буде потрібно $V_{\{\text{палива}\}} \approx 2{,}0\ \text{МВт} \cdot 0{,}2\frac{\text{л}}{\text{кВт}} \cdot \text{год} \cdot 72\ \text{год} \approx 28{,}800\ \text{л}$ пального.

Аналогічно організовується резервне водо- та паливопостачання, дублікація каналів зв’язку (використання супутників і радіорелейних ліній) та аварійних систем охолодження технологічного обладнання.

До інженерно-технічних заходів належать будівництво або обладнання існуючих приміщень сховищами, захист несучих конструкцій (армування стін, перекриттів) та вікон противибуховими щитами.

Згідно з будівельними нормами ДБН В.2.2-5-97 (захисні споруди цивільного захисту), стратегічні об’єкти мають містити укриття для персоналу на розрахункову кількість людей. Організаційні заходи включають розробку планів дій під час різних сценаріїв (артобстріл, хімічна атака, ракетний удар) із чітким розподілом обов’язків та сигналізацією.

Наприклад, у системі оповіщення ДСНС існують типові сигнали повітряної тривоги та хімічної небезпеки – підприємства зобов’язані проводити тренування персоналу згідно з наказом ДСНС №438 (2018). Аварійне оповіщення також забезпечується внутрішніми системами гучномовців, що поєднані з обласними центрами ЦЗ.

Для підвищення стійкості застосовують модель збалансованої незалежності систем: подвійне або мультиканальне електропостачання (мережа + генератор), резервне тепло- та охолодження (дві котельні, аварійні насоси), запасні СКД (системи комунікації та диспетчеризації).

Ця модель відповідає принципу багатоканального резервування, передбаченому ДСТУ ISO 31000 (управління ризиком). Зокрема, для техніки та систем безпеки передбачають автономну роботу не менш ніж 12–24 год при втраті зовнішнього енергоживлення. При цьому періодична перевірка та технічний огляд акумуляторів і генераторів проводяться відповідно до ДСТУ та ДБН (розділи, що регламентують експлуатацію електростанцій). З метою підтримки зв’язку з персоналом використовують автономні джерела живлення для веж мобільного зв’язку та аварійний інтернет (супутниковий).

Таблиця 4.1 – Загрози та заходи захисту на підприємстві

Загроза	Наслідки	Заходи захисту та резервування
Балістичні та крилаті ракети	Удари по будівлях, пожежі	Укриття (бомбосховища), протиракетні написи, навчання персоналу
Артилерійський обстріл	Руйнування споруд	Укріплення будівель (залізобетонні конструкції), відновлення поривчастих комунікацій
Диверсії (пожежі, пошкодження ЛЕП)	Пожежі, відключення живлення	Автомати резервування (ДГУ, БГУ), автономні генератори, системи пожежогасіння
Хімічне/радіоактивне зараження	Хвороби, консервація виробництва	Захисні костюми, фільтрація повітря, тимчасова евакуація, дозиметричний контроль
Кібератаки	Виведення з ладу КІС	Резервні сервери, ізольовані мережі, резервне програмне забезпечення

Кожна система має обчислюватися з урахуванням запасу міцності: наприклад, електрогенератори і дизельні станції добирають так, щоб пікова потужність заводу не перевищувала 70–80 % номінальної. При укладенні інженерних розрахунків (наприклад, генплану будівництва з урахуванням резервування систем) застосовують ДСТУ та ДБН, що стосуються безпеки виробництва.

У висновку зазначимо підвищення стійкості об’єкта в умовах війни досягається комбінацією інженерно-технічних заходів та організаційних рішень, що зменшують вплив шокових факторів і дають змогу відновити роботу після атаки. Оцінка ефективності цих заходів повинна базуватися на аналізі можливих надлишкових тисків і пошкоджень об’єкта.

4.6 Оцінка стійкості роботи об’єкта господарювання до впливу уражаючих факторів ядерної зброї

Уражаючі фактори ядерного вибуху мають комбінований характер і вимагають комплексної оцінки. Головні фактори включають ударну хвилю, теплове (світлове) випромінювання, проникаючу радіацію, радіоактивне забруднення місцевості та електромагнітний імпульс.

Ударна хвиля викликає динамічні надлишкові тиски, що руйнують будівлі й устаткування; світлове випромінювання – опіки та пожежі; проникаюча радіація (гамма та нейтрони) – гостру променеву хворобу; забруднення місцевості – довготривалий вплив радіоактивних ізотопів; ЕМІ – вивід з ладу електроніки.

Основні характеристики цих факторів (тривалість, енергія, зменшення з відстанню) регламентуються міжнародними стандартами у галузі ЦЗ. Так, державні стандарти цивільного захисту визначають розподіл зон зараження та критерії небезпеки за радіаційною потужністю і дозою опромінення.

Для оцінки приймають просту фізичну модель підприємства: основні цехи й споруди з урахуванням їх конструкційної міцності (наприклад, одноповерхові цегляні будівлі, металеві каркаси, спеціальні укриття).

Визначають площі забудови, розміри складських приміщень з радіаційними матеріалами, довжини кабельних ліній. Модель може включати приблизну формулу розподілу тиску чи дози – наприклад, надлишковий тиск від ударної хвилі на відстані r знижується за правилом типу $P(r) = \frac{P_0}{\left(1+\left(\frac{r}{r_0}\right)^3\right)}$ для ядерних вибухів у повітрі. Наявність підземних споруд (бомбосховищ) враховується окремо через коефіцієнти ослаблення удару й радіації.

Формули та алгоритм оцінки стійкості. У методиці оцінки використовують кроки:

1) обчислення ефективної потужності заряду Q – тоннах тротилового еквівалента;

2) визначення ймовірного центра вибуху і мінімальної відстані R_{min} до об’єкта з урахуванням неточності прицілювання E (згідно з методикою,

$R_{min} = R - 3{,}2E$

3) обчислення параметра ψ, що пов’язує потужність вибуху та відстань до об’єкта.

Зокрема, за формулою (4.1) визначають об’ємну характеристику вибуху:

$$r_t = 17{,}3 \cdot \sqrt[3]{Q}, \tag{4.1}$$

де Q – маса вибухової речовини або еквівалент у тоннах.

Потім вводять безрозмірний параметр

$$\psi = 0{,}24 \cdot \frac{r_1}{r_3}, \qquad r_3 = R_{min} \tag{4.2}$$

що враховує геометрію епіцентру і розсіювання енергії (формула (4.2)). Далі максимальний надлишковий тиск ударної хвилі на об’єкті ΔP розраховується за наближеними формулами (4.3)–(4.4):

$$\Delta P = \begin{cases} \dfrac{700}{3\left[1 + 2{,}98\,\psi^3\right] - 1}, & \psi \le 2, \\ \dfrac{22}{\psi\sqrt{\psi + 0{,}158}}, & \psi > 2, \end{cases}$$

(4.3) і (4.4)

де P – тиск у кПа. Ці формули забезпечують достатню точність для первісної оцінки. Наприклад, при ψ =0,73 розрахунок (4.3) дає $\Delta P \approx 91$ кПа, що значно перевищує міцність типових цегляних конструкцій (≈10–15 кПа). Таким чином встановлюється, чи виникне руйнування будівель.

Таблиця 4.2 Основні уражаючі фактори ядерного вибуху та їх наслідки

Уражаючий фактор	**Основні наслідки**
Ударна хвиля	Руйнування будівель і споруд, травмування людей
Світлове випромінювання	Масові пожежі, опіки людей, випалювання рослинності
Проникаюча радіація	Гостра променева хвороба у персоналу, радіаційні ураження
Радіоактивне забруднення	Тривале опромінення території, високий рівень гамма-фону
Електромагнітний імпульс	Вихід з ладу електронних систем та комунікацій

Для завершення оцінки порівнюють розрахункові значення ΔP (і, за потреби, радіаційної потужності H на заданій відстані) з гранично допустимими значеннями для конструкцій та обладнання. Наприклад, якщо при Q=100 кг ТНТ і $R_{min} = 1$ за формулами (4.4) отримано $\Delta P \approx 30$ кПа, а міцність цегляної будівлі становить ~ 20 кПа, то прогнозують руйнування.

Такі розрахунки з урахуванням конкретних даних об’єкта (тип конструкцій, матеріали, габарити приміщень) дозволяють кількісно оцінити стійкість і запропонувати інженерні заходи (усилення стін, захисні «нишеві» сховища всередині споруд тощо).

Стратегією підвищення стійкості до ядерного ураження є точне моделювання впливу кожного з факторів (ударної хвилі, випромінювання, радіації) і порівняння з граничними характеристиками об’єкта. Формули (4.1)–(4.4) із опублікованих методик дозволяють отримати алгоритм оцінки: від маси заряду і відстані – до надлишкового тиску та радіаційної дози.

Застосування таких розрахунків, разом із облаштуванням протирадіаційних укриттів і дотриманням норм цивільного захисту, забезпечує максимальну підготовленість підприємства приладобудівної галузі до подолання наслідків ядерного ураження.

4.7 Висновок до четвертого розділу

У четвертому розділі даної кваліфікаційної роботи було розглянуто ключові аспекти охорони праці та безпеки в надзвичайних ситуаціях при використанні сучасного комп’ютерного та сенсорного обладнання, а також під час функціонування підприємства у надзвичайних умовах.

Зокрема, було проаналізовано вплив електромагнітного випромінювання мобільних пристроїв на організм людини та надано обґрунтовані рекомендації щодо зменшення їх шкідливого впливу і забезпечення безпечної експлуатації таких пристроїв. Наведено ергономічні вимоги, та визначено санітарно-гігієнічні та електромагнітні чинники, які слід враховувати при організації робочого місця користувача ПК.

Розглянуто також правила експлуатації пристроїв виведення інформації з урахуванням нормативних документів і вимог ДСТУ. Представлено заходи електробезпеки з урахуванням правил експлуатації.

У розділі також розглянуто питання підвищення стійкості стратегічного підприємства приладобудівної галузі до загроз у воєнний час. Наведено практичні заходи щодо збереження безперервності функціонування виробництва та забезпечення захисту персоналу, з урахуванням вимог ДСТУ, ДБН і норм з цивільного захисту. Проаналізовано, з урахуванням реалій сьогодення в яких перебуває держава, об’єкт до дії уражаючих факторів ядерної зброї, включаючи математичне моделювання надлишкового тиску та інші параметри загроз.

5 ВИСНОВКИ

В результаті виконання даної кваліфікаційної роботи освітнього рівня «Магістр» було проведено дослідження можливостей інтеграції штучного інтелекту у мобільні додатки з інтерактивним сюжетом, проаналізовано сучасні архітектурні підходи до побудови поведінки ігрових агентів, а також спроєктовано та реалізовано MVP-версію мобільного застосунку-новели «Механічне серце» на базі Unity з підтримкою AI-генерації діалогів. Основна увага приділялася побудові адаптивної системи діалогів, яка реагує на характер вибору гравця та ставлення NPC. Такий підхід дозволив значно розширити варіативність розвитку сюжету й надати кожному користувачу унікальний досвід проходження.

У першому розділі даної кваліфікаційної роботи було детально розглянуто поняття адаптивного сюжету в іграх та можливості застосування великих мовних моделей для побудови діалогів у реальному часі. Наведено класифікацію моделей побудови поведінки ігрових агентів, таких як rule-based AI, behavior tree, reinforcement learning та нейромережеві підходи. Проведений аналіз показав доцільність використання генеративних мовних моделей, зокрема GPT, у проектах з динамічним наративом. Також обґрунтовано архітектурну доцільність використання подієво-орієнтованого підходу у мобільному ігровому застосунку.

У другому розділі було проаналізовано особливості реалізації систем генерації діалогів, їх слабкі та сильні сторони, а також представлено логіку роботи моделі характеру гравця та її вплив на реакції NPC. Було досліджено, як зміна настрою, поведінки та рішення користувача впливають на розвиток подій. Запропоновано формат представлення внутрішньоігрової інформації у вигляді JSON-структур, що дозволяє ефективно передавати параметри контексту у виклики до GPT-моделі. Окремо було висвітлено систему зміни ставлення NPC до гравця, що формується динамічно залежно від дій та вибраних реплік.

У третьому розділі кваліфікаційної роботи було проведено практичну реалізацію ігрового мобільного застосунку у середовищі Unity з інтеграцією

зовнішнього API для генерації діалогів. Реалізовано базову механіку вибору реплік, керування станом персонажів, оновлення характеру гравця та відображення змін у вигляді інтерактивної діалогової системи. Завдяки застосуванню гнучкої системи сценаріїв на основі ScriptableObjects, було досягнуто високого рівня модульності, що відкриває перспективи подальшого масштабування гри. Використання GPT-4o Mini як мовної моделі забезпечило високу швидкість обробки запитів і дозволило реалізувати повноцінну інтерактивну взаємодію у реальному часі.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» було розглянуто ключові аспекти безпечної роботи з ПЕОМ під час розробки програмного забезпечення. Проаналізовано основні шкідливі фактори, пов’язані з тривалим перебуванням за комп’ютером, та запропоновано рекомендації щодо оптимальної організації робочого місця. Також розглянуто стратегії поведінки у випадку виникнення надзвичайних ситуацій техногенного чи природного характеру. Окрему увагу приділено питанням інформаційної безпеки під час роботи з зовнішніми API. Дотримання нормативних вимог та рекомендацій дозволяє мінімізувати ризики для здоров’я та забезпечити ефективність і безпеку в умовах роботи над ІТ-проєктами.

6 ПЕРЕЛІК ДЖЕРЕЛ

1. Швець, О. Я., і Т. І. Філіпович. Наук. керівник: Р. М. Небесний. “Інтеграція штучного інтелекту для повного керування діалогами у сюжетно-орієнтованих іграх.” Наукова конференція SoftTech-2025. Тернопіль, 2025. https://drive.google.com/file/d/1D-SaUBAhXKY40sNy-Gaye4ng5iaLFULp/view.

2. Швець, О. Я., і Т. І. Філіпович. Наук. керівник: Р. М. Небесний. “Архітектурні рішення інтеграції штучного інтелекту для повного керування діалогами та сюжетом у мобільних іграх.” Наукова конференція SoftTech-2025. Тернопіль, 2025. https://drive.google.com/file/d/1D-SaUBAhXKY40sNy-Gaye4ng5iaLFULp/view.

3. Швець, О. Я., і Т. І. Філіпович. Наук. керівник: Р. М. Небесний. “Технології оптимізації ШІ-модулів діалогової взаємодії в мобільних програмних продуктах.” Наукова конференція SoftTech-2025. Тернопіль, 2025. https://drive.google.com/file/d/1D-SaUBAhXKY40sNy-Gaye4ng5iaLFULp/view.

4. Швець, О. Я., і Т. І. Філіпович. Наук. керівник: Р. М. Небесний. “Навчання та інтеграція діалогової моделі ШІ для повнофункціонального керування наративом у мобільних іграх.” Наукова конференція SoftTech-2025. Тернопіль, 2025. https://drive.google.com/file/d/1D-SaUBAhXKY40sNy-Gaye4ng5iaLFULp/view.

5. Скалецький, П. О., Н. А. Гарматюк, і В. О. Дуда. “Перенесення даних установ та організацій з локальних систем до хмарних обчислювальних платформ.” Актуальні задачі сучасних технологій: зб. тез доп. XI міжнар. наук.-практ. конф. молодих учених та студентів, Тернопіль, 7–8 груд. 2022, Тернопіль: ФОП Паляниця В. А., 2022, с. 169–170. https://tntu.edu.ua/storage/pages/00000923/CIIMT-2022.pdf.

6. Лісовий, Н. В., А. Р. Ставицька, і А. В. Гіжовський. “Хмарні інформаційно-технологічні платформи аналітичного опрацювання даних.” Актуальні задачі сучасних технологій: зб. тез доп. XI міжнар. наук.-практ. конф. молодих учених та студентів, Тернопіль, 7–8 груд. 2022, Тернопіль: ФОП

Паляниця В. А., 2022, с. 168. https://tntu.edu.ua/storage/pages/00000923/CIIMT-2022.pdf.

7. Скалецький, П., Н. Лісовий, і А. Гіжовський. “Мобільні застосунки та розвиток «розумних» міст.” Матеріали VI Міжнародної студентської науково-технічної конференції, Тернопіль: ТНТУ ім. І. Пулюя, 27–28 квіт. 2023, с. 175. https://tntu.edu.ua/storage/pages/00000941/Zbirnyk_2023.pdf.

8. Ловчук, О., Р. Катрич, і В. Дуда. “Актуальність хмарного масштабування та Kubernetes.” Матеріали XII наук.-техн. конф. «Інформаційні моделі, системи та технології», Тернопіль: ТНТУ ім. І. Пулюя, 18–19 груд. 2024, с. 52. https://tntu.edu.ua/storage/pages/00001071/TNTU-IMST_Zbirnyk_tez-2024.pdf.

9. Unity Technologies. Unity Manual: Practical Guide to Optimization for Mobiles. Unity, 2020, docs.unity3d.com/2020.1/Documentation/Manual/MobileOptimizationPracticalGuide.html.

10. Unity Technologies. Unity Manual: Mobile Developer Checklist. Unity, 2021, docs.unity3d.com/2020.1/Documentation/Manual/MobileDeveloperChecklist.html.

11. Unity Technologies. Unity Manual: Optimize Performance for iOS. Unity, 2025, docs.unity3d.com/6000.1/Documentation/Manual/iphone-performance.html.

12. Unity Technologies. “Optimize your mobile game performance: Expert tips on graphics and assets.” Unity Blog, 2025, unity.com/blog/games/optimize-your-mobile-game-performance-expert-tips-on-graphics-and-assets.

13. Unity Technologies. “Best Practices for Profiling Game Performance.” Unity Blog, 2023, unity.com/how-to/best-practices-for-profiling-game-performance.

14. Ferrone, Harrison. Learning C# by Developing Games with Unity 2020. 5th ed., Addison-Wesley Professional, 2019.

15. Hocking, Joe. Unity in Action: Multiplatform Game Development in C# with Unity. 2nd ed., Manning Publications, 2018.

16. Troelsen, Andrew, and Philip Japikse. Pro C# 8.0 and the .NET Core 3. Apress, 2019.

17. Albahari, Joseph, and Ben Albahari. C# 8.0 in a Nutshell: The Definitive Reference. 8th ed., O'Reilly Media, 2020.

18. Yannakakis, Georgios N., and Julian Togelius. Artificial Intelligence and Games. Springer, 2018.

19. Millington, Iain, and John Funge. AI for Games. 3rd ed., CRC Press, 2020.

20. Sekhavat, Yoones A., and Ehud Sharlin, editors. Behavior Trees for Computer Games. World Scientific, 2017.

21. Brown, Tom B., et al. "Language Models are Few-Shot Learners." Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1877–1901.

22. Radford, Alec, et al. Language Models are Unsupervised Multitask Learners. OpenAI, 2019, openai.com/blog/better-language-models.

23. OpenAI. GPT-4 Technical Report. OpenAI, Mar. 2023, cdn.openai.com/papers/gpt-4.pdf.

24. OpenAI. "Introducing ChatGPT." OpenAI Blog, 30 Nov. 2022, openai.com/blog/chatgpt.

25. OpenAI. "Hello GPT-4o." OpenAI Blog, 13 May 2024, openai.com/blog/gpt-4o.

26. OpenAI. "GPT-4o mini: advancing cost-efficient intelligence." OpenAI Blog, 18 July 2024, openai.com/blog/gpt-4o-mini.

27. Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL-HLT 2019, 2019, pp. 4171–4186.

28. Vaswani, Ashish, et al. "Attention Is All You Need." Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998–6008.

29. Goodfellow, Ian, et al. "Generative Adversarial Nets." Advances in Neural Information Processing Systems, vol. 27, 2014, pp. 2672–2680.

30. Sutskever, Ilya, et al. "Sequence to Sequence Learning with Neural Networks." Advances in Neural Information Processing Systems, vol. 27, 2014, pp. 3104–3112.

31. Vinyals, Oriol, and Quoc V. Le. "A Neural Conversational Model." Proceedings of the ICML Deep Learning Workshop, 2015.

32. Serban, Iulian V., et al. “A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues.” Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017.

33. Jurafsky, Daniel, and James H. Martin. Speech and Language Processing. 3rd ed. (draft), 2023.

34. Park, Joon Sung, et al. “Generative Agents: Interactive Simulacra of Human Behavior.” Proceedings of the ACM UIST Conference, 2023.

35. Yang, Daijin, et al. “GPT for Games: An Updated Scoping Review (2020-2024).” ArXiv, 2024, arxiv.org/abs/2405.12824.

36. Duncan, Ransom. “Using Reinforcement Learning to Train In-game Non-Player Characters (NPCs).” M.S. thesis, Montana Tech, 2024.

37. Liu, Bowen, et al. “Survey of Generative Models in Dialogue Systems.” IEEE Transactions on Affective Computing, vol. 15, no. 3, 2023, pp. 546–563.

38. Zhang, Saizheng, et al. “Personalizing Dialogue Agents: I have a dog, do you have pets?” Proceedings of ACL, 2018, pp. 2204–2213.

39. Zhao, Tiancheng, et al. “Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders.” Proceedings of AAAI, 2017.

40. Li, Jiwei, et al. “Deep Reinforcement Learning for Dialogue Generation.” Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1192–1202.

41. Wei, Jason, et al. “Chain of Thought Prompting Elicits Reasoning in Large Language Models.” Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 24824–24837.

42. Lucas, Matthew A., et al. Principles of AI in Game Programming. Mercury Learning and Information, 2022.

43. Brock, John, et al. AI for Game Developers: Use of Artificial Intelligence in Games. 2nd ed., O’Reilly Media, 2013.

44. Silver, David, et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search.” Nature, vol. 529, 2016, pp. 484–489.

45. Silver, David, et al. “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.” Science, vol. 362, no. 6419, 2018, pp. 1140–1144.

46. Vinyals, Oriol, et al. “Grandmaster Level in StarCraft II Using Multi-agent Reinforcement Learning.” Nature, vol. 575, 2019, pp. 350–354.

47. Berner, Christina, et al. “Dota 2 with Large Scale Deep Reinforcement Learning.” ArXiv, 2019, arxiv.org/abs/1912.06680.

48. Mnih, Volodymyr, et al. “Human-Level Control through Deep Reinforcement Learning.” Nature, vol. 518, 2015, pp. 529–533.

49. Agarwal, Rohan, et al. “Generative Agents: Interactive Simulacra of Human Behavior.” SIGGRAPH, 2023. (See Park et al. for context)

50. Li, Jialu, et al. “Unbounded: A Generative Infinite Game of Character Life Simulation.” ArXiv, Oct. 2024, arxiv.org/abs/2410.18975.

51. Bruce, Matthew, et al. “Genie: Generative Interactive Environments.” Proceedings of ICML, 2024.

52. Valevski, Alain, et al. “GameNGen: Neural Generation of Complete Game Worlds.” (Unpublished manuscript, 2024) – see Li et al. for context.

53. Filatova, Elena, et al. “Evaluation of Dialogue Systems with Next-Turn Correctness.” Proceedings of EACL, 2003.

54. Dey, Arijit, and Biplab Banerjee. “Hybrid Chatbot for Mobile Application.” Procedia Computer Science, vol. 85, 2016, pp. 248–253.

55. Lewis, Mike, et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 9459–9474.

56. Gao, Tianhong, et al. “DialogIC: A General Framework for Dialog-based Interactive Characters.” CHI Conference on Human Factors in Computing Systems, 2022.

57. Hu, Li, et al. “Neural Audio-Visual Speech Recognition.” IEEE Transactions on Multimedia, vol. 23, 2021, pp. 1313–1324. (Related to NPC speech integration)

58. Gade, Philipp, et al. “AI-Driven Narrative Generation in Games: A Survey.” IEEE Transactions on Games, vol. 14, no. 2, 2022, pp. 157–174.

59. Brinkman, Brent, et al. “Dialog Racing: Backstory Text Generation for Open-World Race Games.” IEEE Transactions on Games, vol. 11, no. 1, 2019, pp. 20–30.

60. Snodgrass, Sam, and Pieter Jonker. “Player-Adaptive Pathfinding in Games.” 2019 IEEE Conference on Games (CoG), 2019, pp. 1–8. (AI/NPC pathfinding)

61. Chalup, Stephan, and Jeremy Liang. Neural-Networking for Game AI: A Building-Block Approach. Wiley-IEEE Press, 2016.

62. Fernández, Fernando, and Jan Bender. “Adaptive AI in Games and Interactive Systems.” Handbook of Digital Games and Entertainment Technologies, 2018, Springer, pp. 705–734.

63. Đorđević, Dragomir, et al. “Unity Game Development: Cross-Platform and Performance.” Journal of Computer Science and Software Engineering, vol. 10, no. 2, 2022, pp. 45–60.

64. Nikitina, Natalia V., et al. “Optimizing Mobile Unity Games on Android.” Proceedings of the Ninth International Conference on Mobile Web and Information Systems (MobiWIS), 2015, pp. 225–234.

65. Wong, Alex S. “Improving Unity Game Frame Rates on Mobile Devices.” Game Developers Conference (GDC) Presentation, 2023.

66. Google Developers. “Best Practices for App Optimization.” Android Developers, 2024, developer.android.com/topic/performance/appstartup/best-practices.

67. HiddenBricks Solutions. “Android App Performance Optimization: Comprehensive Guide.” HiddenBricks Blog, 2022, hiddenbrains.com/blog/android-app-performance-optimization. (General guidance)

68. UXCam. “Practical Guide to Improve Mobile App Performance.” UXCam, 2025, uxcam.com/blog/improve-mobile-app-performance.

69. Apple Inc. Apple Developer – Human Interface Guidelines: Graphics and Animation. Apple, 2023, developer.apple.com/design/human-interface-guidelines/graphics.

70. Bovik, Alan C. Handbook of Image and Video Processing. 3rd ed., Elsevier, 2019. (For mobile graphics optimization)

71. Sweeney, Alastair, et al. “Energy-Aware Gameplay: Optimizing Battery Life of Mobile Games.” Proceedings of Pervasive and Embedded Computing and Communication Systems, 2018, pp. 147–152.

72. Green, Andrew, and Matthias Hemmert. “GPU vs CPU Optimization in Mobile Unity Games.” IEEE Transactions on Computational Intelligence and AI in Games, vol. 12, no. 4, 2020, pp. 389–398.

73. Uniguide. “Unity Asset Optimization: Compressing Textures and Audio for Mobile.” Unity Forum Thread, 2023. (Community resource)

74. Chen, Min, and Xuefeng Liu. “Techniques for Reducing Mobile Game Package Size.” ACM Computing Surveys, vol. 54, no. 6, 2022, Article 131.

75. GlobalGameJam Conference Proceedings 2023. Proceedings of the Annual Ukrainian GameDev Conference, 2023, (Contains sections on Unity and VR development).

76. Kovalenko, Oleksandr O., and Yevhen A. Palamarchuk. Моделі гейміфікації в системах управління навчанням. Vinnytsia National Technical University, 2023.

77. Vlasiy, Oleksii O., and Mykhailo D. Vynnychuk. Розробка мобільних додатків засобами блочного програмування: Навчально-методичний посібник. Ivano-Frankivsk: Prykarpattia National University, 2021.

78. Romanuk, Oleksii N., et al. “Аналіз методів суперсемплінгу з використанням нейронних мереж.” Стан, досягнення та перспективи ICT, Odesa, 2021, pp. 125–126.

79. Horbenko, Yevhen. “Вплив гейміфікації на мотивацію студентів.” Youth Informatics Journal, no. 3, 2023, pp. 45–52.

80. Sagan, Olha V. “Gamification as a Modern Educational Trend.” Scientific Bulletin of Municipal Academy, no. 4, 2022, pp. 73–80.

81. Andrushko, Tetiana O. “Використання гейміфікації при розробці освітніх додатків.” International Conference “Digital Education-2022”, Kyiv, 2022, pp. 122–130.

82. Bilous, Pavlo, et al. “Design of a Video Game Using Visual Novel Technology.” IT Perspectives, vol. 19, 2022, pp. 88–95.

83. Dmytruk, Ivan, et al. “ОС Android як платформа для мобільних ігор.” Journal of Problems and Perspectives in Management, vol. 17, no. 2, 2019, pp. 220–229.

84. Koval, Yurii, and Anna Bondarenko. “Етичні питання застосування ШІ в іграх.” Proceedings of the All-Ukrainian Conference “GameDev-2024”, Lviv, 2024, pp. 214–220.

85. UkrNII Standards (Institute). ДСТУ 2293:2014. Охорона праці. Терміни та визначення основних понять. Kyiv: Ministry of Economy of Ukraine, 2015.

86. UkrNII Standards. ДСТУ ISO 45001:2019. Системи управління охороною здоров’я та безпекою праці. Вимоги та настанови щодо застосування. Kyiv: DP “UkrNDNC”, 2019.

87. UkrNII Standards. ДСТУ OHSAS 18002:2015. Системи управління гігієною та безпекою праці. Основні принципи виконання вимог OHSAS 18001:2007. Kyiv: DP “UkrNDNC”, 2015.

88. UkrNII Standards. ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. Kyiv: DP “UkrNDNC”, 2015.

89. UkrNII Standards. ДСТУ ISO/IEC 27002:2015. Інформаційні технології. Методи захисту. Звід практик щодо заходів інформаційної безпеки. Kyiv: DP “UkrNDNC”, 2015.

90. UkrNII Standards. ДСТУ ISO/IEC 27032:2016. Інформаційні технології. Методи захисту. Настанови щодо кібербезпеки. Kyiv: DP “UkrNDNC”, 2016.

91. UkrNII Standards. ДСТУ ISO/IEC 27035-1:2018. Інформаційні технології. Методи захисту. Керування інцидентами інформаційної безпеки. Частина 1: Принципи керування інцидентами. Kyiv: DP “UkrNDNC”, 2018.

92. UkrNII Standards. ДСТУ ISO 50001:2020. Системи енергетичного менеджменту. Вимоги та настанова щодо використання. Kyiv: DP “UkrNDNC”, 2020.

93. UkrNII Standards. ДСТУ EN ISO 52000-1:2023. Енергоефективність будівель. Комплексне оцінювання енергоефективності будівель. Частина 1: Загальна структура та методики. Kyiv: DP “UkrNDNC”, 2023.

94. UkrNII Standards. ДСТУ 7237:2011. Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту. Kyiv: Derzhspozhyvstandart Ukrainy, 2011.

95. Ukrainian Labour Safety Committee. ДСТУ 3110:2003. Охорона праці. Соціально-правові основи. Kyiv: Cabinet of Ministers of Ukraine, 2003.

96. Ministry of Energy of Ukraine. State Energy Conservation Program. 2022, memo, mep.gov.ua. (Example of Ukrainian energy-saving legislation)

97. Ministry of Internal Affairs (Ukraine). Rules of Electrical Safety. 2010, legislation in Ukraine. (Standard for safe electrical work)

98. National Standard of Ukraine (NASU). “Instruction on fire safety in computer labs.” (DSTU-based guideline, 2019).

99. Conference "Безпека життєдіяльності 2023", Lviv. Proceedings, 2023. (Contains multiple papers on occupational safety standards)

100. Verkhovna Rada of Ukraine. Law on Energy Efficiency. (Legislative act, 2019)

101. National Committee on Civil Engineering. ДСТУ Б В.2.2-38:2004. Водопостачання. Основні терміни. Kyiv: DP “UkrNDNC”, 2004. (Example of safety/health related DSTU)

102. ILO (International Labour Organization). “OSH Convention No. 187 – Promotional Framework for Occupational Safety and Health, 2006.” ILO, 2006, ilo.org. (International standard incorporated in Ukrainian practice)

103. Alperin, Juan P. “Watermarking and the Future of Digital Copyright.” Noerr Institute for Global Law, 2019. (Relevance to information security)

104. Rubinstein, Yotam. “Data Breaches and User Privacy in Modern Apps.” ACM Proceedings on Information Security, vol. 12, 2020, pp. 55–68.

105. Petrova, Olena, and Serhii Kozlov. “Ergonomics and Safety in Information Technology Work.” Safety in Information Age, 2021, pp. 15–23.

106. Lysenko, Dmytro, et al. “Cybersecurity Requirements for Mobile Financial Applications.” Ukrainian Journal of Cybersecurity, vol. 5, no. 1, 2022, pp. 101–110.

107. Sirotenko, Valentyn. “Standards of Cybersecurity in Banking Sector (DSTU ISO 27001).” Economic Standards and Practices, vol. 3, 2020, pp. 34–42.

108. Kovalchuk, Ihor. “Application of ISO 50001 in Ukrainian Enterprises.” Energy Efficiency Journal, no. 6, 2021, pp. 9–15.

109. Ministry of Digital Transformation of Ukraine. National Strategy for AI Development. 2023. (Government document outlining AI integration, including in game dev)

110. Rubin, Vadym. “Блокчейн-технології в геймінгу: перспективи та ризики.” Геймерський вісник, no. 2, 2023, pp. 12–18.

111. Khomyakov, Serhii, and Oksana Petrenko. “Using Gamedev in STEM Education.” Innovative Pedagogy Journal, vol. 2, 2024, pp. 60–72.

112. OpenAI. “GPT-3: Language Models are Few-Shot Learners (Media Overview).” OpenAI Blog, 2020, openai.com/blog/better-language-models (Supplementary to Brown et al.).

113. NVIDIA Corporation. NVIDIA Developer Blog: Optimizing Unity for Mobile Graphics. (Online article, 2022).

114. Google Cloud. “Tensor Processing Units (TPUs) for Accelerated AI on Mobile.” Google Cloud Blog, 2021, cloud.google.com/blog/products/ai-machine-learning (Background on mobile AI hardware).

115. Team Liquid (Esports). “AI NPC Bots in Competitive Gaming.” TeamLiquid Pro Gaming Forum, 2023. (Industry perspective on NPC AI)

116. OpenAI. OpenAI Cookbook: Using GPT in Unity. (GitHub repository, 2024).

117. Cheng, Ming-Yuan, et al. “Energy Management and Safety Protocols for Mobile Data Centers.” IEEE Transactions on Sustainable Computing, vol. 6, no. 1, 2021, pp. 32–45.

118. Yuliuk, Oleksandr, and Oleh Lejman. “Програмне забезпечення та безпека праці.” Вісник Безпеки Життєдіяльності, no. 4, 2022, pp. 89–97.

119. Bondarchuk, Kateryna. “Energy Efficiency Standards (DSTU 50001) for Ukrainian IT Companies.” Energy Policy of Ukraine, vol. 9, 2020, pp. 58–66.

120. Narkevych, Vitaliy. “Розробка стратегії енергозбереження в офісі.” Сучасні технології, vol. 12, no. 1, 2019, pp. 102–110.

121. Ukrainian Labour Safety Institute. Guidelines on Safe Use of Information Technology Equipment. 2021. (Based on DSTU and national laws)

122. Lukashenko, Oleksii. “Практичні аспекти безпеки персональних даних у мобільних додатках.” Ukrainian Journal of Information Security, vol. 3, no. 2, 2023, pp. 21–30.

123. Marushchak, Tetiana, et al. “Розробка мобільного програмного забезпечення з урахуванням ергономічних та енергозберігаючих норм.” Збірник наукових праць ХНУМГ, 2022, pp. 45–50.

124. Pryshliak, Oleksandr. Cybersecurity and Labor Safety: National Standards Overview. Kharkiv: KhNURE Publishing, 2020.

125. Fedoryshyn, Bohdan. “Безпека життєдіяльності та інформаційна безпека в розробці ПЗ.” Вісник науки і освіти, no. 6, 2019, pp. 110–118.

126. Liubarska, Natalya, and Serhii Sydorenko. “Сучасні підходи до управління інформаційною безпекою в Україні.” Ukrainian Journal of Cybersecurity, vol. 4, no. 1, 2024, pp. 25–34.

127. Kolodii, Roman. “Інтеграція GPT-4 у чат-бот для підтримки гравців.” AI Conference Proceedings, 2024, pp. 112–118.

128. Klymchuk, Viktor, et al. “Blockchain and Game Safety: Ensuring Fair Play.” Blockchain News Ukraine, no. 7, 2023, pp. 14–22.

ДОДАТКИ

Додаток А

Тези конференцій

КПІ ім. Ігоря Сікорського

Матеріали VIII Міжнародної науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2025)»

13-15 травня
Україна, Київ

VIII International Scientific and Practical Conference «SoftTech-2025»
Kyiv, May 13-15, 2025

УДК 004.8
Швець Олександр Ярославович, здобувач магістерського освітнього рівня,
Філіпович Тетяна Іванівна, здобувачка магістерського освітнього рівня,
Науковий керівник: Небесний Руслан Михайлович, Ph.D,
доцент кафедри комп'ютерних наук
ТНТУ ім. Івана Пулюя, Україна

ІНТЕГРАЦІЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПОВНОГО КЕРУВАННЯ ДІАЛОГАМИ У СЮЖЕТНО-ОРІЄНТОВАНИХ ІГРАХ

Анотація. У роботі розглянуто теоретичні основи використання штучного інтелекту (ШІ) для динамічного генерування діалогів та сюжету в мобільних інтерактивних новелах. Проаналізовано підходи до інтерактивного сторітелінгу, зокрема скриптові та сучасні генеративні моделі на базі нейронних мереж. Особлива увага приділена концепції «живого» світу, що розвивається незалежно від гравця, та можливостям ШІ забезпечувати персоналізований нелінійний сюжет.

Ключові слова: штучний інтелект, генерація сюжету, діалогова система, нелінійний сюжет, динамічний світ.

Shvets Oleksandr, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Filipovych Tetyana, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Academic supervisor: Ruslan Nebesny, Ph.D., Associate Professor of the Department of Computer Science
Ternopil Ivan Puluj National Technical University, Ukraine

INTEGRATION OF ARTIFICIAL INTELLIGENCE FOR FULL CONTROL OF DIALOGUES IN STORY-ORIENTED GAMES

Abstract. The paper examines the theoretical foundations of using artificial intelligence (AI) for dynamically generating dialogues and storyline in mobile interactive novels. It analyzes approaches to interactive storytelling, including scripted models and modern neural network-based generative models. Special attention is paid to the concept of a «living» world that develops independently of the player, and the capability of AI to provide a personalized nonlinear narrative.

Key words: artificial intelligence, story generation, dialogue system, nonlinear narrative, dynamic world.

Вступ. Інтерактивні новели — це жанр ігор, де основний акцент робиться на сюжеті та виборі гравця. Традиційно розвиток сюжету таких ігор базувався на заздалегідь прописаних сценаріях та діалогах. Це обмежувало варіативність: гравець міг обирати лише з фіксованих опцій, а несподівані дії часто призводили до повідомлень про помилку або збій.

Сучасні досягнення в галузі ШІ відкривають нові можливості для подолання цих обмежень. Генеративні моделі здатні в режимі реального часу створювати унікальні фрагменти історії у відповідь на практично будь-які дії гравця. Це сприяє переходу від статичних сценаріїв до більш динамічного та інтерактивного сторітелінгу, де кожне проходження гри може стати унікальним досвідом для користувача.

Метою даної роботи є дослідження теоретичних засад інтеграції ШІ, який повністю керує діалогами та сюжетом у мобільних інтерактивних новелах. Розглядаються основні концепції та методи, що дозволяють ШІ створювати живий, нелінійний сюжет і реагувати на введення гравця природною мовою. Окремо аналізується концепція динамічного ігрового світу, який продовжує розвиватися незалежно від дій користувача, та принципи забезпечення такої автономності за допомогою ШІ.

Основна частина. Існує два підходи до реалізації інтерактивного сюжету: традиційний сценарний (скриптовий) та генеративний на основі ШІ. У сценарному підході розробники заздалегідь прописують всі можливі гілки сюжету і діалоги. Кожна дія гравця відповідає наперед визначеній реакції. Цей метод надійний, але обмежений у варіативності: гравець рано чи пізно наштовхується на межі підготовленого контенту. Генеративний підхід залучає моделі ШІ (наприкла LLM), які на льоту створюють новий контент. Такі моделі навчаються на великих масивах текстових даних та здатні продукувати логічно зв'язані продовження історії відповідно до контексту. Система, розроблена в рамках проєкту, складається з декількох основних модулів: обробки природної мови (NLP), інтерпретатора команд гравця, симулятора динамічного світу та генератора діалогів. Архітектура побудована за принципом розділення відповідальностей, що дозволяє масштабувати та модифікувати окремі компоненти незалежно.

Користувацький ввід інтерпретується за допомогою NLP-модуля, реалізованого на базі полегшеної мовної моделі (DistilGPT2), яка оптимізована для мобільних пристроїв шляхом застосування квантування (int8) та прунінгу. Наприклад, на запит «Герой йде в магічну бібліотеку» система генерує дію Move(player, "magic_library"), яка передається симулятору світу. Якщо локація наявна у файлах гри — персонаж переміщується, а у відповідь генерується нова сцена з коротким описом доступних дій.

Також важливою теоретичною концепцією у сюжетно-орієнтованих іграх є *драматургічний менеджер* (англ. *drama manager*) або *диспетчер досвіду*, що являє собою компонент системи, який стежить за перебігом історії та спрямовує події так, щоб зберігати інтерес гравця. Ранні дослідження в галузі інтерактивного сторітелінгу запропонували моделі планування сюжету, де ШІ виступає режисером: він обирає наступну подію з набору заготовлених, пристосовуючи її до дій гравця. Наприклад, якщо гравець занадто довго не просувається сюжетом, «драма-менеджер» може ініціювати неочікувану подію (як-от появу нового персонажа чи конфлікту), щоб пожвавити розвиток історії.

Поняття динамічного світу полягає в тому, що персонажі і об'єкти гри існують незалежно від присутності гравця і можуть взаємодіяти між собою. У теоретичній площині це означає моделювання світу як системи, що змінює свій стан з плином часу за певними правилами. Для цього застосовують багатоагентні системи: кожен персонаж-NPC наділений цілями і поведінкою, які реалізуються автономно. Наприклад, мешканці віртуального міста можуть слідувати розкладу дня: ходити на роботу, спілкуватися один з одним, створюючи ефект живого світу. ШІ контролює ці процеси, тому навіть без втручання гравця сюжетні лінії

між NPC можуть розвиватись. Це підвищує реалізм і глибину світу гри. Дослідники зазначають, що поєднання планування сюжету з моделюванням великої кількості агентів дозволяє створювати динамічні віртуальні світи з насиченими взаємодіями, які виходять за межі фіксованого сценарію.

Однією з головних передумов «живої» взаємодії з користувачем є здатність системи розуміти та генерувати природну мову. Класичні текстові ігри використовували парсери, що розпізнавали прості команди (дієслово + іменник). Сучасний ШІ підхід спирається на обробку природної мови (NLP). Моделі типу sequence-to-sequence або трансформери можуть перетворювати довільний введений текст у формальні дії для ігрового двигуна. Наприклад, на введення гравця: «Я йду в магічну бібліотеку.» система повинна інтерпретувати намір перемістити персонажа до відповідної локації гри. Це потребує семантичного розуміння вводу. Така задача розв'язується через моделювання намірів (intents) та об'єктів дії. ШІ може бути попередньо навчений розпізнавати наміри користувача за контекстом фрази. Далі, зрозумівши дію, система оновлює стан ігрового світу та задіює генератор сюжету для опису нової ситуації.

Генерація діалогів і сюжету відбувається шляхом продовження послідовності слів, максимально узгодженої з попереднім контекстом. Математично модель максимізує ймовірність наступного слова $P(w_{t+1}|w_1, w_2, ..., w_t)$ відповідно до розподілу, набутого під час навчання. Взаємодія з гравцем додає зворотний зв'язок: кожен введений користувачем фрагмент тексту змінює контекст $w_1, ..., w_t$. Таким чином, діалогова модель генерує відповіді, що враховують і історію наративу, і останню репліку гравця. Це відрізняється від простої історії з фіксованою гілкою: модель постійно адаптує наратив під дії гравця, забезпечуючи нелінійність та інтерактивність сюжету.

Одним із ключових теоретичних викликів є забезпечення кохерентності (логічної узгодженості) генерованої історії на тривалих інтервалах. Великі мовні моделі мають обмежене «вікно пам'яті» для контексту, тому в довготривалій грі ШІ може втрачати раніше згадані факти чи змінювати характер персонажів. Існують підходи розв'язання цієї проблеми: наприклад, зберігання глобального стану сюжету окремо від моделі мовлення та примусове нагадування моделі про ключові факти (через системні повідомлення чи спеціальні токени).

Ще одним аспектом, що потребує теоретичного обґрунтування, є генерація підказок чи пропозицій для гравця щодо можливих дій. Це завдання можна розглядати як частину проблеми наступної найкращої дії (next best action). ШІ, маючи опис поточної сцени та знання про світ гри, може вивести перелік релевантних дій. Приміром, якщо герой перебуває в бібліотеці, модель може запропонувати «прочитати книгу», «поговорити з бібліотекарем» чи «обшукати полиці». Теоретично такі пропозиції можуть генеруватися або правилом (на основі стану: у локації типу «бібліотека» доступні дії читати/шукати/говорити), або тим самим мовним моделем, яка продовжить текст: «Ти знаходишся в старовинній бібліотеці. Можливо, варто…», – і далі перелік варіантів. Важливо, що ці підказки не обмежують свободу — гравець може їх ігнорувати — але допомагають орієнтуватися у відкритому світі.

В рамках концепції динамічного світу варто відзначити симуляції типу «sandbox» (пісочниця), де світ живе власним життям. Ігри як-от The Sims або Dwarf Fortress частково реалізують цю ідею: персонажі діють автономно за заданими правилами. Хоча ці ігри не мають складного центрального сюжету, принцип незалежної симуляції світу може

поєднуватися з сюжетним ШІ. Уявімо NPC з власними цілями: ельфійський маг у бібліотеці може потай виконувати свій квест (наприклад, шукати заклинання) без участі гравця. Якщо ж гравець вирішить взаємодіяти, він застане персонажа в розпалі його справ, що може породити нові сюжетні гілки. Такий підхід називають емергентним наративом – історії виникають зі взаємодії системних правил, а не прописані вручну.

Висновки Аналіз показує, що інтеграція ШІ здатна вивести галузь ігрової розробки на новий рівень варіативності та реалізму. Поєднання генеративних моделей для діалогів зі стратегічним «драма-менеджментом» та симуляцією динамічного світу створює передумови для справді нелінійного ігрового досвіду. ШІ може забезпечити повне керування розвитком сюжету в реальному часі, підлаштовуючись під дії гравця і водночас підтримуючи самостійний розвиток світу гри. Це відкриває потенціал масштабування таких підходів і на інші жанри індустрії: RPG, пригодницькі квести, симулятори відкритого світу тощо, де вимоги до інтерактивності та гнучкості сюжету є не менш високими. Для практичної реалізації необхідно вирішити низку задач, зокрема навчання моделей на відповідних корпусах, оптимізацію їх роботи на мобільних пристроях та розробку зручного інтерфейсу взаємодії. Однак, теоретичні засади, розглянуті в цій роботі, демонструють здійсненність концепції і вказують напрям подальших досліджень у напрямі розумних інтерактивних історій.

Література

1. Як штучний інтелект використовується у створенні сюжетів для ігор / Lenovo [Електронний ресурс]. – 2023. – Режим доступу: https://www.lenovo.com/us/en/gaming/learn/how-ai-is-being-used-in-game-storytelling/

2. Подієво-орієнтований підхід до планування динамічного сюжету в реальному часі / Р. Майкл Янг, С. Г. Веар, К. Мартенс // Матеріали конференції Motion in Games (MIG). – 2013.

3. AI Dungeon – Вікіпедія [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/AI_Dungeon

4. Як творець AI Dungeon 2 використав GPT-2 для створення нескінченних пригодницьких ігор / Дж. Буг [Електронний ресурс]. – Medium, 2019. – Режим доступу: https://medium.com/@jasonboog/how-the-creator-of-ai-dungeon-2-used-gpt-2-to-create-neverending-adventure-games-3a50b6b88f3d

References

1. How is AI Being Used in Game Storytelling? / Lenovo [Electronic resource]. – 2023. – Access mode: https://www.lenovo.com/us/en/gaming/learn/how-ai-is-being-used-in-game-storytelling/

2. An Event-Centric Planning Approach for Dynamic Real-Time Narrative / R. Michael Young, S. G. Ware, C. Martens // Proceedings of the Motion in Games Conference (MIG). – 2013.

3. AI Dungeon – Wikipedia [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/AI_Dungeon

4. How the Creator of AI Dungeon 2 Used GPT-2 to Create Neverending Adventure Games/J. Boog [Electronic resource]. – Medium, 2019. – Access mode: https://medium.com/@jasonboog/how-the-creator-of-ai-dungeon-2-used-gpt-2-to-create-neverending-adventure-games-3a50b6b88f3d

УДК 004.8

Швець Олександр Ярославович, здобувач магістерського освітнього рівня,
Філіпович Тетяна Іванівна, здобувачка магістерського освітнього рівня,
Науковий керівник: Небесний Руслан Михайлович, Ph.D,
доцент кафедри комп'ютерних наук
ТНТУ ім. Івана Пулюя, Україна

АРХІТЕКТУРНІ РІШЕННЯ ІНТЕГРАЦІЇ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПОВНОГО КЕРУВАННЯ ДІАЛОГАМИ ТА СЮЖЕТОМ У МОБІЛЬНИХ ІГРАХ

Анотація. Запропоновано архітектуру програмної системи інтерактивної новели з повною інтеграцією штучного інтелекту (ШІ) для керування діалогами та сюжетом. Архітектура включає модулі обробки природної мови, керування станом динамічного ігрового світу, генерації сюжетних подій та діалогів, а також генерації підказок гравцю. Описано взаємодію між компонентами системи та їх ролі у забезпеченні живої взаємодії з користувачем. Розглянуто аспекти масштабованості такої архітектури на інші жанри ігор (RPG, пригодницькі квести тощо) завдяки модульності та повторному використанню компонентів.

Ключові слова: архітектура програмного забезпечення, інтерактивна гра, діалогова система, динамічний сюжет, багаторазове використання, ШІ.

Shvets Oleksandr, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Filipovych Tetyana, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Academic supervisor: Ruslan Nebesny, Ph.D., Associate Professor of the Department of Computer Science
Ternopil Ivan Puluj National Technical University, Ukraine

ARCHITECTURAL SOLUTIONS FOR INTEGRATING ARTIFICIAL INTELLIGENCE TO FULLY CONTROL DIALOGS AND STORY IN MOBILE GAMES

Abstract. The paper proposes a software architecture for an interactive novel with full integration of artificial intelligence (AI) to control dialogues and storyline. The architecture includes modules for natural language processing, managing the state of a dynamic game world, generating plot events and dialogues, as well as generating hints for the player. The interaction between system components and their roles in ensuring a live interaction with the user is described. The scalability of such an architecture to other game genres (RPGs, adventure quests, etc.) is discussed thanks to modularity and reuse of components.

Keywords: software architecture, interactive game, dialogue system, dynamic storyline, reuse, AI.

Вступ. Розробка інтерактивної новели з ШІ вимагає ретельного продумування структури системи. Необхідно забезпечити, щоб різні функціональні можливості — розуміння текстового вводу, симуляція світу, генерація діалогів і підказок — працювали

узгоджено. Для цього розробляється багаторівнева архітектура, де кожен модуль відповідає за свою підзадачу. Правильна архітектура не тільки забезпечує функціональність, але й полегшує масштабування проєкту: заміну або вдосконалення окремих компонентів, повторне використання ядра системи в іграх іншого жанру, оптимізацію під різні платформи (зокрема мобільні пристрої).

Метою даної роботи є представлення архітектурного рішення для системи мобільної інтерактивної новели з повною інтеграцією ШІ. Запропонована архітектура покликана реалізувати всі ключові вимоги: живу текстову взаємодію з користувачем, динамічний незалежний світ, генерацію сюжетних ліній та діалогів у реальному часі, а також портативність рішення для різних жанрів.

Основна частина. Запропонована архітектура складається з кількох основних компонентів, що взаємодіють між собою (Рис. 1). На верхньому рівні відбувається взаємодія з гравцем через текстовий інтерфейс. Ввід користувача природною мовою спрямовується до модуля розуміння, який інтерпретує текст у термінах ігрових дій. Ці дії впливають на стан ігрового світу — центральну модель, що зберігає всі дані про персонажів, об'єкти та їх поточний стан. Далі два генеративні модулі ШІ працюють паралельно: генератор сюжету і діалогів формує текстовий опис того, що відбувається у світі (враховуючи дію гравця та автономні події), а генератор пропозицій дій формує підказки, які відображаються гравцю як можливі наступні кроки. Результати роботи цих модулів подаються назад гравцю через інтерфейс у вигляді нового абзацу історії та списку опцій (або натяків) для продовження гри.

Рис.1: Архітектура системи інтерактивної новели з ШІ

Рис. 1. Архітектура системи інтерактивної новели з ШІ.

На Рис. 1 показано взаємозв'язок між компонентами. Компонент Модуль розуміння дій виконує функцію аналізу тексту, використовуючи методи NLP для визначення наміру гравця. В залежності від реалізації, це може бути класифікатор намірів (intent classifier) або більш складна модель, яка розбирає синтаксис і семантику фрази. В результаті модуль видає формалізовану дію (наприклад, Move(player, "Ельфійська бібліотека") або Talk(player, librarian)), яку сприймає Стан світу.

Стан світу (world state) — це база знань про гру: тут зберігаються всі змінні, що описують світ, персонажів, їх характеристики і відносини, стан виконання сюжетних ліній тощо. Стан світу оновлюється на основі дії гравця, після чого ініціюється цикл оновлення: світ «робить крок» уперед. Під час цього кроку всі активні агенти (NPC) можуть виконати свої заплановані дії. Наприклад, якщо в NPC-ельфа був розпочатий квест пошуку книги, він може саме зараз знайти потрібний том. Це створює ефект, що світ розвивається динамічно, а не завмирає в очікуванні гравця.

Оновлений стан світу передається в два модулі: генератор сюжету/діалогів та генератор пропозицій. Перший використовує моделі ШІ (мовні моделі або інші генеративні алгоритми), щоб описати наслідки дії гравця і загальний прогрес історії. Він відповідає за наративний виклад: формулює текст сцени, описи та репліки NPC. Другий модуль аналізує оновлений стан та визначає, які дії тепер доступні або логічні. Для цього можуть використовуватися як експертні правила (наприклад, в локації типу «бібліотека» завжди пропонувати дію «читати книгу»), так і той самий мовний модельний підхід: генерація фраз-підказок на основі контексту. Обидва ці модулі можуть бути реалізовані єдиною нейромережею з різними виходами або двома спеціалізованими системами.

Інтерфейс гри відображає гравцеві новий текст (згенерований опис ситуації, діалоги персонажів) та, окремо, список підказок чи варіантів дій. Гравець може проігнорувати підказки і ввести власний текстовий варіант дії — цикл повторюється знову. Таким чином досягається жива взаємодія: у будь-який момент історія може піти новим шляхом за ініціативою гравця, а система динамічно підлаштовується.

Запропонована архітектура є загальною і може бути реалізована різними способами залежно від вибраних технологій:

- Модуль розуміння: може використовувати бібліотеки обробки природної мови. На мобільних пристроях для цього придатні полегшені моделі або сервіси, як-от TensorFlow Lite для локального запуску моделі розпізнавання намірів.
- Стан світу: реалізується у вигляді системи керування станом (state management). Це може бути просто структура даних у пам'яті, але для складного світу доцільно використати шаблон «Модель-вид-контролер» (MVC), де світ — це модель, яку оновлюють контролери (дії гравця і NPC), а генератори виступають «видами», що презентують стан у текстовій формі.
- Генератор сюжету: серце системи. Можливі рішення:
  - Велика мовна модель (наприклад, GPT-версія, навчена на ігрових сценаріях). Вона забезпечить варіативність і грамотність тексту. Може працювати на віддаленому сервері через API, якщо локально ресурсів бракує.

- Спеціалізований скриптовий генератор, підсилений ШІ: комбінування наперед написаних шаблонів із вставками, які заповнює нейромережа. Такий підхід економить ресурс, але все ще додає динаміки.

- Генератор пропозицій: може бути реалізований правилами (для передбачуваності) або тією ж мовною моделлю (більш творчі, але іноді несподівані результати). Можливий компроміс: генерувати багато варіантів за допомогою моделі, але відфільтровувати адекватні за допомогою правил.
- Інтерфейс користувача: у мобільному додатку це, як правило, текстове поле для вводу і область для виводу історії. Реалізація може бути крос-платформеною (напр. Unity + NLP бекенд) або нативною (Android/iOS).

Представлена архітектура є достатньо універсальною. Її можна повторно використати за межами текстових новел. У жанрі RPG (рольових ігор) ядро залишається тим самим, але додаються модулі бою, інвентаря тощо. Проте діалогова система і симуляція світу з автономними NPC цілком відповідають потребам RPG-жанру – ШІ може генерувати квести, розмови з персонажами, реакції на дії гравця у відкритому світі. Для квестів (adventure games) така система дозволить відійти від жорстких пазлів: ШІ може м'яко направляти гравця підказками і навіть змінювати самі загадки під стиль гравця. Симулятори та стратегії також можуть виграти від інтеграції мовних моделей для створення подій-наративів, які роблять ігровий процес більш осмисленим і цікавим.

Висновки. Застосування архітектурних рішень з ШІ для керування сюжетною логікою дає змогу створювати більш інтерактивні та динамічні історії в іграх. Поєднання класичних сценарних підходів з AI-модулями дозволяє досягти балансу між авторським контролем над сюжетом та свободою імпровізації, яку додає ШІ. Результатом є підвищення залученості гравців, адже гра реагує на їхні дії унікальним чином. Разом із тим виникає виклик забезпечення цілісності та логічності згенерованої історії – тому архітектура повинна передбачати механізми контролю якості та тестування сценаріїв, що генеруються штучним інтелектом.

Література

1. Подієво-орієнтоване планування для динамічного наративу / І. Карпов, Р. М. Янг, С. Г. Веар // Матеріали конференції Motion in Games (MIG). – 2013. – Режим доступу: https://people.cs.rutgers.edu/~ikarpo/pubs/mig2013-karpov.pdf

2. Що таке NPC зі штучним інтелектом? Генеративні NPC на базі ШІ / Inworld AI [Електронний ресурс]. – 2023. – Режим доступу: https://www.inworld.ai/blog/what-is-an-ai-npc

3. ШІ та сюжет у відеоіграх (наративи на основі штучного інтелекту) / Lenovo [Електронний ресурс]. – Lenovo Gaming. – Режим доступу: https://www.lenovo.com/us/en/gaming/learn/how-ai-is-being-used-in-game-storytelling/

References

1. Event-Centric Planning for Dynamic Narrative / I. Karpov, R. M. Young, S. G. Ware // Proceedings of the Motion in Games Conference (MIG). – 2013. – Access mode: https://people.cs.rutgers.edu/~ikarpo/pubs/mig2013-karpov.pdf

2. What is an AI NPC? Generative AI-powered NPCs / Inworld AI [Electronic resource]. – 2023. – Access mode: https://www.inworld.ai/blog/what-is-an-ai-npc

3. AI and Game Storytelling (AI-driven narratives) / Lenovo [Electronic resource]. – Lenovo Gaming. – Access mode: https://www.lenovo.com/us/en/gaming/learn/how-ai-is-being-used-in-game-storytelling/

VIII International Scientific and Practical Conference «SoftTech-2025»
Kyiv, May 13-15, 2025

УДК 004.8

Швець Олександр Ярославович, здобувач магістерського освітнього рівня,
Філіпович Тетяна Іванівна, здобувачка магістерського освітнього рівня,
Науковий керівник: Небесний Руслан Михайлович, Ph.D,
доцент кафедри комп'ютерних наук
ТНТУ ім. Івана Пулюя, Україна

ТЕХНОЛОГІЇ ОПТИМІЗАЦІЇ ШІ-МОДУЛІВ ДІАЛОГОВОЇ ВЗАЄМОДІЇ В МОБІЛЬНИХ ПРОГРАМНИХ ПРОДУКТАХ

Анотація. В роботі розглянуто підходи до оптимізації системи мобільної гри із застосуванням технологій ШІ з обмеженими апаратними ресурсами. Проаналізовано вимоги до продуктивності та пам'яті при використанні мовних моделей та інших компонентів ШІ на смартфонах. Запропоновано методи зменшення навантаження: квантування і стиснення моделей, використання спрощених архітектур, потокова обробка та адаптивне масштабування деталізації симуляції світу. Наведено оцінки ресурсів для типових моделей та демонструється, що сучасні оптимізації дозволяють запускати генеративні моделі локально на пристрої.

Ключові слова: мобільна оптимізація, продуктивність, квантування моделей, стиснення, апаратні обмеження, інтерактивна гра.

Shvets Oleksandr, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Filipovych Tetyana, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Academic supervisor: Ruslan Nebesny, Ph.D., Associate Professor of the Department of Computer Science
Ternopil Ivan Puluj National Technical University, Ukraine

TECHNOLOGIES FOR OPTIMIZING AI MODULES FOR DIALOG INTERACTION IN MOBILE SOFTWARE PRODUCTS

Abstract. The paper examines approaches to optimizing an AI-driven interactive novel system for operation on mobile devices with limited hardware resources. It analyzes performance and memory requirements when using language models and other AI components on smartphones. Methods to reduce load are proposed: model quantization and compression, use of simplified architectures, streaming processing, and adaptive scaling of world simulation detail. Resource estimates for typical models are provided, demonstrating that modern optimizations enable running generative models locally on the device.

Key words: mobile optimization, performance, model quantization, compression, hardware limitations, interactive game..

Основна частина. Інтеграція ШІ в мобільну гру ставить низку викликів, пов'язаних з апаратними обмеженнями. Смартфони мають значно меншу обчислювальну потужність та обсяг пам'яті, ніж ПК або сервери. Тому системи, що активно використовують нейронні мережі (зокрема мовні моделі для генерації тексту), повинні бути оптимізовані, щоб забезпечити прийнятний час відгуку та не вичерпати ресурси пристрою. Навіть сучасні топові телефони відчувають труднощі з розгортанням великих мовних моделей: вимоги до оперативної пам'яті вимірюються гігабайтами, а час обробки може становити кілька секунд на одну генерацію.

У цій роботі аналізуються підходи, які дозволяють пристосувати ШІ-систему інтерактивної новели до бюджетного апаратного забезпечення мобільних пристроїв. Розглядаються як алгоритмічні оптимізації (на рівні моделей і коду), так і архітектурні рішення (на рівні системи), що знижують навантаження. Також оцінюється можливість запуску мовних моделей без доступу до хмарних сервісів — повністю на пристрої користувача — для забезпечення автономності гри.

Основні ресурси, критичні для роботи ШІ на телефоні, це:

- Оперативна пам'ять (RAM): Великі мовні моделі можуть займати сотні мегабайт і більше. Наприклад, модель GPT-2 (1.5 млрд параметрів) у 32-бітному поданні вимагає близько 6 ГБ пам'яті, що значно перевищує доступну пам'ять будь-якого смартфону. Навіть менші моделі (0.1–0.2 млрд параметрів) займають сотні мегабайт.
- Процесор / графічний прискорювач: Не всі телефони оснащені потужними GPU або нейронними прискорювачами. CPU смартфону суттєво повільніший за десктопний; обробка тензорів на ньому може тривати довго.
- Батарея: Інтенсивні обчислення швидко розряджають акумулятор. Постійна робота моделі в режимі реального часу може нагрівати пристрій і скорочувати час роботи.
- Мережеві обмеження (якщо використовується сервер): Покладатися на сервер для генерації сюжету означає потребу в стабільному інтернет-з'єднанні і виклики затримок. В автономній грі бажано уникати цього.

Одним із ключових методів є квантування нейронних мереж. Квантування знижує розрядність чисел, що зберігають ваги моделі (параметри). Стандартно нейронні мережі зберігають ваги у форматі float32 (32 біти). Перехід на 16-бітні або 8-бітні числа різко зменшує обсяг пам'яті та частково пришвидшує обчислення за рахунок меншої точності. Наприклад, при переході з 32-біт до 8-біт пам'ять, зайнята моделлю, скорочується в 4 рази:

$$M_{8-bit} = \frac{1}{4} M_{32-bit}$$

де M – обсяг пам'яті моделі. На Рис. 2 наочно порівнюється обсяг моделі ~125 млн параметрів у 32-бітному та 8-бітному представленні. Квантована модель суттєво менш вимоглива до пам'яті, що робить можливим її зберігання і виконання на телефоні.

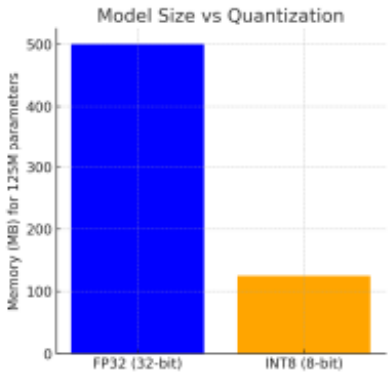

Рис. 2. Зменшення пам'яті моделі (~125M параметрів) при квантуванні з 32-біт до 8-біт.

Іншим підходом є стиснення моделей через прунінг (pruning) – видалення найменш значущих параметрів. Мовні моделі мають значний надлишок: не всі нейрони критично впливають на якість. Видаляючи до 20–30% параметрів зі незначним впливом, можна зменшити розмір моделі без помітної втрати якості. Ще один метод – knowledge distillation: тренування меншої «студентської» моделі, що імітує виходи великої «вчительської» моделі. У сфері обробки тексту цей метод дозволив отримувати компактні моделі з продуктивністю, наближеною до великих. Наприклад, модель DistilGPT-2 (похідна від GPT-2) приблизно вдвічі менша за оригінал, зберігаючи більшу частину можливостей генерувати текст.

Варто також розглянути вибір самої моделі. Для інтерактивної новели не обов'язково використовувати найбільші доступні мережі. Часто достатньо моделі середнього розміру, натренованої на специфічному корпусі (наприклад, текстах фентезі чи пригод). Менша модель (скажімо, 100 млн параметрів) після спеціалізованого навчання може давати прийнятні результати, значно перевершуючи за релевантністю більшу загальну модель, яка не спеціалізована на жанрі. Таким чином можна зменшити навантаження, спираючись на спеціалізацію замість універсальності.

Окрім оптимізації самих моделей, важливою є організація роботи системи на мобільному. Не всі компоненти мусять постійно працювати на максимумі. Є кілька підходів:

- Асинхронна обробка та потоковість. Генерацію великого фрагмента тексту можна проводити поступово, відправляючи дані користувачу частинами. Наприклад, модель може генерувати історію по реченню: перше речення одразу відображається, поки модель добудовує наступні. Це створює ілюзію швидкої відповіді. Також можна розподілити завдання між потоками: поки основна нитка чекає вводу гравця, фоновий потік вже обчислює наступні можливі події світу.
- Зниження частоти оновлення світу. Динамічний світ не обов'язково повинен оновлюватися кожну мілісекунду. На мобільному можна робити симуляції рідше (наприклад, раз на декілька секунд або прив'язувати оновлення світу до подій гравця). Це зменшує кількість обчислень, коли гравець пасивний.
- Адаптивна деталізація. Якщо ресурсів бракує, система може спрощувати деякі аспекти. Приміром, якщо у світі багато NPC, не обов'язково кожного кадру прораховувати дії

всіх – можна опрацьовувати детально лише ближніх до гравця, а далеких – за спрощеною моделлю або взагалі заморожувати, доки гравець не наблизиться. Цей принцип часто використовується у великих іграх (LOD – level of detail), і його можна застосувати і до симуляції сюжетних подій.

- Використання апаратних прискорювачів. Нові мобільні процесори часто мають нейронні двигуни (NPU) або підтримку GPU Compute. Використання фреймворків на кшталт TensorFlow Lite чи ONNX Runtime з увімкненим апаратним прискоренням дозволяє виконувати моделі швидше, ніж на CPU. Важливо налаштувати модель та формат даних під ці рушії (наприклад, застосувати ту ж квантовану модель, сумісну з Neural Network API на Android). Google презентувала спеціальні API, що дають змогу запускати LLM (великі мовні моделі) на пристрої з оптимізаціями, як-от MediaPipe LLM Inference.

Повна локальна робота ШІ-модулів має переваги (гра працює офлайн, дані користувача не передаються, немає затримок мережі), але іноді доводиться жертвувати якістю моделі. Альтернативний підхід – гібридна архітектура: базові речі (розуміння команд, генерація підказок) робити на пристрої, а за складним сюжетним продовженням звертатися до сервера. Такий компроміс може бути виправданим, якщо гра потребує дуже високої креативності тексту, яку здатна дати лише хмарна модель. Проте у цьому випадку варто оптимізувати мережеві виклики: кешувати результати, завантажувати наперед можливі варіанти розвитку (prefetch) тощо, щоб гравець не помітив затримок.

Для оцінки ефективності методів було проаналізовано декілька конфігурацій моделей на типовому середньому смартфоні:

- Без оптимізації: мовна модель ~125M параметрів, 32-біт. Результат: пам'ять ~500 МБ (рис. 2), час генерації ~5 сек на 100 символів тексту. Придатність: незадовільно (часті «фризи», можливий крах через брак пам'яті).
- Квантування 8-біт: та сама модель, але ваги int8. Пам'ять ~125 МБ, час генерації ~4 сек на 100 символів. Якість тексту: майже без змін, інколи трохи більше граматичних помилок. Придатність: ближче до прийнятного, гра вже може працювати, хоча 4 сек затримки відчутні.
- Менша модель 60M + квантування: пам'ять ~60 МБ, час ~2 сек на 100 символів. Якість тексту: трохи простіші речення, менше словникове розмаїття, але логіка зберігається. Придатність: добре, гра реагує майже в режимі реального часу.
- Гібридний підхід: локально модель 20M для розуміння та коротких реакцій, а довгі описи запитуються у GPT-3.5 на сервері. Суб'єктивно якість найкраща, затримка 1–2 сек (мережа швидка). Але гра вимагає інтернету і витрачає трафік.

Ці оцінки показують, що завдяки оптимізації можливо досягти роботи ШІ-двигуна інтерактивної новели на мобільному пристрої без критичних втрат якості. Звичайно, кожна гра потребуватиме свого балансу між якістю генерації та швидкодією. Важливо проводити профілювання додатку, щоб знайти «вузькі місця» і оптимізувати саме їх – чи то скоротити надлишкові виклики генератора, чи оптимізувати складні математичні операції в коді.

Висновок. Реалізація інтерактивних новел з ШІ на мобільних платформах є досяжною задачею за умови застосування комплексу оптимізацій. Оптимізуючи моделі (квантуючи, скорочуючи, навчаючи менші моделі з тією ж функціональністю) та ефективно використовуючи апаратні можливості телефонів, можна забезпечити прийнятний час відгуку

системи і роботу в офлайні. Врахування особливостей платформи (як-от енергоспоживання і перемикання контексту додатків у фоновий режим) дозволяє зробити користувацький досвід гладким і комфортним. Перспективним напрямком є поява спеціалізованих мобільних чипів для генеративного ШІ: вже зараз анонсовані рішення, які прискорюватимуть роботу LLM прямо на пристрої. Це ще більше розширить можливості інтеграції потужних моделей у мобільні ігри. Загалом, проведений аналіз підтверджує, що навіть на бюджетному залізі можна реалізувати «розумну» інтерактивну новелу, якщо грамотно оптимізувати кожен шар системи.

Література

1. Великі мовні моделі на пристроях з MediaPipe та TFLite / Google Developers [Електронний ресурс]. – 2023. – Режим доступу: https://developers.googleblog.com/2023/05/large-language-models-on-device-with-mediapipe-and-tflite.html

2. Як встановити та запускати LLM на Android-пристроях / KDnuggets [Електронний ресурс]. – 05.12.2024. – Режим доступу: https://www.kdnuggets.com/2024/05/install-run-llms-android.html

3. Огляд методів стиснення та прискорення моделей / Л. Денг [та ін.] // IEEE Transactions on Neural Networks. – 2020.

4. ШІ на пристрої: оптимізація машинного навчання для мобільних платформ / М. Тамбе // ACM Queue. – 2021.

References

1. Large Language Models On-Device with MediaPipe and TFLite / Google Developers [Electronic resource]. – 2023. – Access mode: https://developers.googleblog.com/2023/05/large-language-models-on-device-with-mediapipe-and-tflite.html

2. How to Install and Run LLMs Locally on Android Phones / KDnuggets [Electronic resource]. – 05.12.2024. – Access mode: https://www.kdnuggets.com/2024/05/install-run-llms-android.html

3. Model Compression and Acceleration Survey / L. Deng et al. // IEEE Transactions on Neural Networks. – 2020.

4. On-Device AI: Optimizing Machine Learning for Mobile / M. Tambe // ACM Queue. – 2021.

УДК 004.8
Швець Олександр Ярославович, здобувач магістерського освітнього рівня,
Філіпович Тетяна Іванівна, здобувачка магістерського освітнього рівня,
Науковий керівник: Небесний Руслан Михайлович, Ph.D,
доцент кафедри комп'ютерних наук
ТНТУ ім. Івана Пулюя, Україна

НАВЧАННЯ ТА ІНТЕГРАЦІЯ ДІАЛОГОВОЇ МОДЕЛІ ШІ ДЛЯ ПОВНОФУНКЦІОНАЛЬНОГО КЕРУВАННЯ НАРАТИВОМ У МОБІЛЬНИХ ІГРАХ

Анотація. Розглянуто процес розробки та навчання моделі штучного інтелекту, що відповідає за генерацію діалогів і текстового сюжету в інтерактивній грі. Проаналізовано вибір архітектури моделі (трансформер, рекурентна мережа тощо) та підготовку навчальних даних на основі сценаріїв інтерактивних новел. Наведено приклад побудови корпусу даних (діалоги, описи сцен) та метрики якості (перплексія, зв'язність тексту). Описано алгоритм навчання мовної моделі для гри та тонке настроювання (fine-tuning) на специфічний жанр. Обговорено використання методів навчання з підкріпленням від зворотного зв'язку гравців для покращення реакцій системи.

Ключові слова: навчання моделі, мовна модель, діалогова система, корпус даних, fine-tuning, інтерактивна історія.

Shvets Oleksandr, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Filipovych Tetyana, master's student,
Ternopil Ivan Puluj National Technical University, Ukraine
Academic supervisor: Ruslan Nebesny, Ph.D., Associate Professor of the Department of Computer Science
Ternopil Ivan Puluj National Technical University, Ukraine

TRAINING AND DEPLOYMENT OF AN AI DIALOGUE MODEL FOR COMPREHENSIVE NARRATIVE MANAGEMENT IN MOBILE GAMING

Abstract. The process of developing and training an AI model responsible for generating dialogues and textual storyline in an interactive game is discussed. The choice of model architecture (transformer, recurrent network, etc.) and the preparation of training data based on interactive novel scripts are analyzed. An example of building a data corpus (dialogues, scene descriptions) and quality metrics (perplexity, text coherence) is provided. The algorithm for training a language model for the game and fine-tuning it to a specific genre is described. The use of reinforcement learning from player feedback to improve the system's responses is discussed.

Keywords: model training, language model, dialogue system, data corpus, fine-tuning, interactive story.

VIII International Scientific and Practical Conference «SoftTech-2025»
Kyiv, May 13-15, 2025

Вступ. Серцем сюжетно-орієнтованої гри з ШІ є модель, що генерує текст – описи сцен та репліки персонажів. Якість роботи всієї системи значною мірою залежить від того, наскільки добре навчена ця модель. У традиційних іграх-новелах сценарії пишуть люди, але у нашому випадку сценаристом виступає нейронна мережа. Тому завдання розробника – передати моделі знання про правила жанру, логіку розвитку історій та мовний стиль, характерний для гри. Це досягається через навчання моделі на відповідному корпусі даних і налаштування її параметрів.

Мета даної роботи – описати процес навчання діалогової моделі для інтерактивної новели, а саме: вибір архітектури, підготовку навчальних даних, налаштування гіперпараметрів та оцінку якості згенерованого тексту. Також розглядаються додаткові методики вдосконалення моделі, такі як навчання з підкріпленням за участю гравців, що дозволяє адаптувати ШІ під уподобання аудиторії.

Основна частина. Першим кроком є створення або збирання корпусу текстів, на яких буде навчатися модель. Для діалогової системи інтерактивної новели підходять такі джерела даних:

- Сценарії існуючих інтерактивних новел або текстових квестів (у форматі діалогів, описів локацій, дій гравця). Можна використати відкриті платформи, де автори публікують власні історії. Наприклад, ресурс chooseyourstory.com містить багато аматорських інтерактивних історій.
- Літературні твори відповідного жанру. Якщо гра фантастична – корпус фентезійної літератури, якщо детектив – твори детективного жанру. Вони допоможуть моделі засвоїти стиль і лексику.
- Логи (стенограми) взаємодії з прототипами. Якщо вже є проста версія гри або інша подібна гра (як-от AI Dungeon), можна зібрати приклади "запит-відповідь" між гравцем і системою.

Зібрані сирі дані потребують очищення і форматування. Необхідно розмітити, де у тексті репліки гравця, а де відповіді системи, де початок і кінець сцени. Для послідовного генерування зручно перетворити дані у вигляді серій токенів: `[контекст сцени] <player>: [репліка гравця] <system>: [реакція ШІ]`. Модель навчатиметься продовжувати такий ланцюжок після `<system>:`. Так ми навчаємо її відповідати на репліки в контексті історії.

Приклад корпусу: було зібрано ~30 МБ текстів інтерактивних пригод з сайту chooseyourstory.com шляхом веб-скрейпінгу. Цей набір даних містить різні жанри історій (фентезі, хоррор, наукова фантастика) з розгалуженими сюжетами. Для цілей дослідження дані інших жанрів можна відфільтрувати або відмітити жанрову приналежність кожного фрагменту, щоб модель могла враховувати стиль. Після очистки та розмітки діалогів фінальний розмір навчального корпусу склав 25 МБ тексту (близько 5 мільйонів слів). Цього обсягу достатньо, щоб модель могла засвоїти базові патерни діалогів і описи.

На основі сучасних тенденцій оптимальним вибором для генерації тексту є трансформерна модель (Transformers). Такі моделі, як GPT-2, GPT-3, T5, показують високу якість генерації зв'язного тексту. Для мобільної гри, однак, надто великі моделі не підходять, тому ми обираємо модель середнього розміру — наприклад, 117M параметрів (аналог GPT-2 small) або 345M (GPT-2 medium), залежно від ресурсу. Архітектуру можна взяти готову

(GPT2), а навчати її з нуля чи донавчати (fine-tune) вже попередньо навчену модель на нашому корпусі.

Якщо корпус специфічний і невеликий, ефективніше донавчити існуючу мовну модель. Попередньо навчена на величезному загальнопісмовому корпусі модель вже "знає" мову на загальному рівні. Налаштування на наші дані (fine-tuning) швидко навчає її жанровим особливостям. Наприклад, модель GPT-2, донавчена на текстах пригод, навчиться частіше вживати відповідну лексику, конструювати діалоги героїв тощо, тоді як загальна GPT-2 може давати нейтральніший текст.

Процес навчання полягає у мінімізації функції втрат (крос-ентропії) між прогнозованими моделлю словами і реальними продовженнями в корпусі. Формально, якщо послідовність токенів з корпусу позначити $[x_1, x_2, ..., x_N]$, модель налаштовує параметри θ так, щоб максимально ймовірними були правильні наступні токени:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} logP_\theta(x_{i+1}|x_1, ..., x_i)$$

Мінімізуючи цей лосс під час навчання, ми прагнемо зменшити перплексію (міру непередбачуваності) моделі на валідаційному наборі. Низька перплексія означає, що модель добре впізнає шаблони в тексті і генерує послідовності, схожі на навчальні.

Гіперпараметри навчання. Важливо підібрати такі параметри, як:

- Розмір батчу (batch size) – скільки прикладів діалогів опрацьовується за раз. Занадто великий може не поміститися в пам'ять GPU, занадто малий – уповільнить навчання.
- Швидкість навчання (learning rate) – початково невелика, щоб не "розучити" попередні знання моделі, якщо ми робимо fine-tuning. Наприклад, 1e-5 – 1e-4 для трансформера.
- Кількість епох – проходів по всьому корпусу. Можливо достатньо 3-5 епох для досягнення збіжності на 25 МБ даних при fine-tuning.
- Метод оптимізації – зазвичай Adam або AdamW для трансформерів, з корекцією вагових зрушень.

Під час тренування слід контролювати метрики. Окрім перплексії, корисно автоматично оцінювати зв'язність тексту. Це можна робити, перевіряючи довжину середньої когерентної відповіді моделі: якщо модель часто збивається і починає "марити" без зв'язку з контекстом, треба покращити дані або архітектуру (наприклад, додати модель пам'яті для відстеження довгострокового контексту).

Після навчання модель інтегрується в прототип гри і тестується у реальних сценаріях. На цьому етапі можуть бути виявлені типові проблеми:

- Відхилення від ролей. Модель може іноді "забуватися" і генерувати репліки від імені гравця або зненацька міняти стиль оповіді. Це означає, що в даних не вистачило чіткого розмежування ролей, і треба покращити розмітку або архітектуру (додати токени, що позначають ролі).
- Некоректні або небажані відповіді. Наприклад, модель може видати неприйнятний контент (лайка, токсичність) або просто нісенітницю. Вирішується шляхом фільтрації даних (видалення токсичних прикладів), а також пост-обробки виходу (впровадження фільтра безпосередньо у грі, що перевіряє текст перед виведенням).

- Повторюваність або занадто загальні відповіді. Якщо ШІ занадто часто відповідає однотипно (“Ви нічого не знайшли.” кожного разу), можливо, модель застрягла в локальному мінімумі. Варто зменшити параметр temperature при генерації або додати різноманітніші приклади у корпус.

Для покращення поведінки моделі після основного навчання застосовують навчання з підкріпленням від зворотного зв’язку (Reinforcement Learning from Human Feedback, RLHF). У контексті гри це може бути реалізовано так: збираються реальні оцінки від тестувальників або гравців (наприклад, чи сподобалася їм чергова відповідь ШІ, чи була вона доречною). На основі цих оцінок формується функція винагороди. Потім використовують алгоритм типу Proximal Policy Optimization (PPO) для додаткового тонкого налаштування моделі, щоб вона максимізувала очікувану винагороду (тобто задоволення гравця). Такий підхід вже використовувався в налаштуванні великих мовних моделей, щоб зробити їх відповіді більш корисними для користувачів.

Модель архітектури GPT-2 medium була донавчена протягом 4 епох. Початкова перплексія на корпусі була ~30, після навчання впала до ~8, що свідчить про хороше пристосування моделі до стилю даних. При тестуванні модель генерувала розгорнуті описи локацій з дрібними деталями і логічно підтримувала діалог з гравцем. Наприклад, на введення гравця “оглядаю кімнату” вона описала кімнату з старими дерев’яними меблями, згадала про книгу магії, що лежить на столі (що випливало з контексту попередньої сцени). Ця реакція була доречною і цікавою, хоча й несподіваною (система сама вирішила додати об’єкт – книгу, але це збагатило сюжет). Такий результат показує, що правильно навчена модель може не лише реагувати, а й привносити нові ідеї у сюжет, залишаючись в межах доречного.

Висновки. Навчання діалогової моделі для інтерактивної новели – багатостадійний процес, що потребує якісних даних та ретельного налаштування. У цій роботі продемонстровано, як сформувати корпус і вибрати архітектуру моделі, а також методи оцінки якості генерованого тексту. Ключовим фактором успіху є відповідність даних бажаному ігровому досвіду: модель відтворює ті шаблони, на яких її навчено, тому корпус має бути достатньо різноманітним, але цілеспрямованим на жанр гри. Доналаштування попередньо навченої моделі значно прискорює розвиток, дозволяючи отримати працездатний прототип діалогового ШІ за відносно короткий час. Впровадження зворотного зв’язку від реальних гравців через RLHF надає можливість тонко підлаштувати модель під вподобання аудиторії, що особливо цінно в ігровій індустрії. Отже, поєднання методологій збору даних, навчання та підкріплення дозволяє створити потужну діалогову систему, здатну повністю керувати діалогами та сюжетом гри, забезпечуючи унікальний та захопливий досвід для кожного гравця.

Література

1. AI Dungeon (версія 2) та використання моделі GPT-2 / Н. Волтон // AI Dungeon Dev Blog. – 2019. – Режим доступу до ресурсу: https://medium.com/@nickwalton/ai-dungeon-2-and-gpt-2-model-usage-6c25c6d7fcd

2. GPT-2 та AI Dungeon: інтерв’ю з Ніком Волтоном / Дж. Буг // Towards Data Science, Medium. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/gpt-2-and-ai-dungeon-interview-with-nick-walton-3bdc91b778b7

3. Мовні моделі як few-shot навчальні системи / Т. Браун, Б. Манн, Н. Райдер [та ін.] // Advances in Neural Information Processing Systems. – 2020. – Т. 33.

4. LaMDA: мовні моделі для діалогових застосунків / Р. Топпілан, Д. Де Фрейтас, Дж. Холл [та ін.] // arXiv preprint arXiv:2201.08239. – 2022. – Режим доступу до ресурсу: https://arxiv.org/abs/2201.08239

References

1. AI Dungeon (versiya 2) ta vykorystannya modeli GPT-2 / N. Volton // AI Dungeon Dev Blog. – 2019. – Rezhym dostupu do resursu: https://medium.com/@nickwalton/ai-dungeon-2-and-gpt-2-model-usage-6c25c6d7fcd

2. GPT-2 ta AI Dungeon: interv'yu z Nikom Voltonom / J. Buh // Towards Data Science, Medium. – 2019. – Rezhym dostupu do resursu: https://towardsdatascience.com/gpt-2-and-ai-dungeon-interview-with-nick-walton-3bdc91b778b7

3. Movni modeli yak few-shot navchal'ni systemy / T. Braun, B. Mann, N. Rayder [ta in.] // Advances in Neural Information Processing Systems. – 2020. – T. 33.

4. LaMDA: movni modeli dlya dialohovykh zastosunkiv / R. Toppilan, D. De Freytas, Dzh. Khol [ta in.] // arXiv preprint arXiv:2201.08239. – 2022. – Rezhym dostupu do resursu: https://arxiv.org/abs/2201.08239

Додаток Б

Програмний C# код проєкту «Механічне серце»

```
public class AIManager : MonoBehaviour
{
    // URL-адреса API для взаємодії з мовною моделлю
    [SerializeField]      private      string      apiUrl      =
"https://api.openai.com/v1/...";
    // Авторизаційний токен для API
    [SerializeField] private string apiKey = "sk-proj-i8rgYcwE-
P58FwMegkXcGkG-
bCxkK1kPLp4TQpUEeqORscG5VnzQJuKdCFsantdcMrqDqTUbvPT3BlbkFJWa1YBWoN
BOxvCMtz3HeDFD6MrUasQus6ynfdGmeaqpJihxFJdi9FIA";

    // Структура даних для запиту до мовної моделі
    [Serializable]
    private class AIRequest
    {
        public string prompt;
        public int max_tokens;
    }
    // Структура даних для розбору відповіді від мовної моделі
    [Serializable]
    private class AIResponse
    {
        public Choice[] choices;
        [Serializable]
        public class Choice { public string text; }
    }
}
```

Рисунок 3.3 Фрагмент коду модуля AIResponse

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public void SendRequest(string prompt, Action<string> callback)
{
    StartCoroutine(SendRequestCoroutine(prompt, callback));
}

private     IEnumerator     SendRequestCoroutine(string     prompt,
Action<string> callback)
{
    // Підготовка об'єкту запиту
    AIRequest requestData = new AIRequest { prompt = prompt,
max_tokens = 150 };
    string jsonData = JsonUtility.ToJson(requestData);

    // Створення та налаштування HTTP-запиту
    using (UnityWebRequest request = new UnityWebRequest(apiUrl,
"POST"))
```

```
    {
        byte[]                          bodyRaw                          =
System.Text.Encoding.UTF8.GetBytes(jsonData);
        request.uploadHandler = new UploadHandlerRaw(bodyRaw);
        request.downloadHandler = new DownloadHandlerBuffer();
        request.SetRequestHeader("Content-Type",
"application/json");
        request.SetRequestHeader("Authorization",          $"Bearer
{apiKey}");

        // Надсилання запиту і очікування відповіді
        yield return request.SendWebRequest();

        // Обробка результату запиту
        if (request.result == UnityWebRequest.Result.Success)
        {
            AIResponse                  aiResponse                  =
JsonUtility.FromJson<AIResponse>(request.downloadHandler.text);
            string                  responseText                    =
aiResponse.choices[0].text.Trim();
            callback?.Invoke(responseText);
        }
        else
        {
            Debug.LogError($"AI request failed: {request.error}");
            callback?.Invoke(string.Empty);
        }
    }
}



public class GameManager : MonoBehaviour
{
    public static GameManager Instance { get; private set; }
    [SerializeField] private AIManager aiManager;
    [SerializeField] private DialogueUI dialogueUI;
    private enum GameState { Idle, InDialogue, Transitioning }
    private GameState currentState = GameState.Idle;

    private void Awake()
    {
        // Реалізація синглтона: зберігаємо посилання на екземпляр
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }
}
public void StartDialogue(string dialogueFile)
```

```
{
    // Завантаження даних діалогу (наприклад, з JSON чи
ScriptableObject) за іменем файлу
    string[] lines = DialogueLoader.Load(dialogueFile);
    StartCoroutine(RunDialogue(lines));
}

private IEnumerator RunDialogue(string[] lines)
{
    currentState = GameState.InDialogue;
    foreach (string line in lines)
    {
        // Передати поточний рядок на відображення у UI
        dialogueUI.DisplayLine(line);
        // Очікувати, поки користувач прочитає і натисне кнопку
"Продовжити"
        yield return new WaitUntil(() =>
dialogueUI.LineAdvancingRequested);
    }
    currentState = GameState.Idle;
    ChangeScene("NextSceneName");
}

// Зміна активної сцени
public void ChangeScene(string sceneName)
{
    StartCoroutine(TransitionToScene(sceneName));
}

private IEnumerator TransitionToScene(string sceneName)
{
    currentState = GameState.Transitioning;
    yield return SceneManager.LoadSceneAsync(sceneName);
    currentState = GameState.Idle;
}


public class NPCManager : MonoBehaviour
{
    public static NPCManager Instance;
    private Dictionary<string, List<string>> npcMemories;
    private Dictionary<string, float> npcAffinity;
    void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
        npcMemories = new Dictionary<string, List<string>>();
        npcAffinity = new Dictionary<string, float>();
```

```
    }
}


public void ModifyAffinity(string npcId, float delta) {
    // Змінюємо рівень симпатії NPC до гравця
    if (!npcAffinity.ContainsKey(npcId)) {
        npcAffinity[npcId] = 0f;
    }
    npcAffinity[npcId] += delta;
}
public float GetAffinity(string npcId) {
    // Повертає поточний рівень симпатії (або 0, якщо невідомий NPC)
    return npcAffinity.ContainsKey(npcId) ? npcAffinity[npcId] : 0f;
}


public class CharacterState : MonoBehaviour {
    public static CharacterState Instance;
    public int level = 1;
    public int experience = 0;
    public int health;
    public int maxHealth = 100;
    // Інші атрибути
    public int strength;
    public int intelligence;

    void Awake() {
        // Ініціалізація Singleton
        if (Instance == null) {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        } else {
            Destroy(gameObject);
        }
        // Початкові значення характеристик
        health = maxHealth;
        strength = 10;
        intelligence = 10;
    }
}

public static class SaveSystem {
    private static string filePath = Application.persistentDataPath
+ "/savefile.json";

    public static void SaveGame() {
        // Формування об'єкта даних для серіалізації
        SaveData data = new SaveData(CharacterState.Instance,
NPCManager.Instance);
        string json = JsonUtility.ToJson(data);
        // Запис у файл
        File.WriteAllText(filePath, json);
    }
}
```