Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

КОМПЛЕКСНА МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інтелектуальний генератор навчальних середовищ і сценаріїв для симуляторів у Unreal Engine. Частина 2. Навчальне середовище для тренування»

Виконав:
магістрант групи 2КІ-24м
спеціальності 123 – Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Чорний В.В.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ОТ
Черняк О.І.
(прізвище та ініціали)
«12» 12 2025 р.

Опонент: к.т.н. доц. каф. ПЗ
Ракитянська Г.Б.
(прізвище та ініціали)
«12» 12 2025 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
17.12 2025 року

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітній рівень — магістр

Галузь знань — 12 Інформаційні технології

Спеціальність — 123 Комп'ютерна інженерія

Освітньо-професійна програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
25.09.2025 року

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЙНУ РОБОТУ

студенту Чорному Владислав Віталійовичу

1 Тема роботи: «Інтелектуальний генератор навчальних середовищ і сценаріїв для симуляторів у Unreal Engine. Частина 2. Навчальне середовище для тренування», керівник роботи Черняк О.І. к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від «24» вересня 2025 року № 313.

2 Строк подання студентом роботи 04.12.2025 р.

3 Вихідні дані до роботи: актуальність, аналіз аналогів, розробка додатка, структурна схема, алгоритм роботи, тестування.

4 Зміст пояснювальної записки: вступ, аналіз існуючих технологій та підходів до інтелектуальної генерації навчальних середовищ, розробка програмного модуля інтелектуальної генерації навчального середовища, програмна реалізація модуля, тестування та оцінка ефективності розробленого програмного засобу, висновки, перелік використаних джерел, додатки.

5 Перелік графічного матеріалу: UML-діаграма взаємодії компонен інтелектуальної генерації.

6 Консультанти розділів роботи представлені у таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	к.т.н., доц. каф. ОТ Черняк О.І.		
5	к.т.н., доц. каф. ЕПВМ Адлер О.О.		
Нормо Контроль	асист. каф. ОТ Швець С.І.		

7 Дата видачі завдання 25.09.2025 р.

8 Календарний план виконання ККР наведений у таблиці 2.

Таблиця 2 — Календарний план

№	Назва та зміст етапу	Термін виконання		Примітка
1.	Формулювання мети, задач, і вимог до ситеми	25.09.2025	27.09.2025	вик
2.	Аналіз науково-технічних джерел і існуючих систем розпізнавання	28.09.2025	02.10.2025	вик
3.	Розробка загальної архітектури та логічної структури системи	03.10.2025	06.10.2025	вик
4.	Проєктування модулів збору даних, аналітики та візуалізації	07.10.2025	10.10.2025	вик
5.	Розробка алгоритмів обробки зображень і виявлення об'єктів	11.10.2025	17.10.2025	вик
6.	Реалізація програмної частини системи	18.10.2025	25.10.2025	вик
7.	Інтеграція модулів і тестування на реальних/модельних даних	26.10.2025	25.10.2025	вик
8.	Розрахунок економічної частини	10.11.2025	31.10.2025	вик
9.	Оформлення пояснювальної записки та ілюстративного матеріалу	01.11.2025	03.11.2025	вик
10.	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	04.11.2025	09.11.2025	вик
11.	Попередній захист	11.11.2025	11.11.2025	вик
12.	Захист МКР	17.12.2025	17.12.2025	вик

Студент _____ Чорний В.В.

Керівник роботи _____ Черняк О.І.

АНОТАЦІЯ

УДК 004.4

Чорний В.В. Інтелектуальний генератор навчальних середовищ і сценаріїв для симуляторів у Unreal Engine. Частина 2. Навчальне середовище для тренування. Магістерська комплексна робота зі спеціальності 123 — Комп'ютерна інженерія, освітня програма — Комп'ютерна інженерія. Вінниця: ВНТУ, 2025. 135 с.

На укр. мові. Бібліогр.: 40 назв; рис.: 14, табл.:16

У магістерські кваліфікаційній роботі досліджено та розроблено навчальні середовища для симуляторів на базі Unreal Engine, які функціонують на принципах інтелектуальної генерації тренувальних сценаріїв.

Ключові слова: інтелектуальна генерація сценаріїв, навчальне середовище, симулятор, Unreal Engine, C++, адаптивна складність.

ABSTRACT

UDC 004.4

Chornij V.V. Intelligent learning environment and scripting generator for simulators in Unreal Engine. Part 2. Learning environment for training. Master's Complex Work in Specialty 123 — Computer Engineering, Educational Program — Computer Engineering. Vinnytsia: VNTU, 2025. 135 p.

In the master's qualification work, educational environments for simulators based on the Unreal Engine, which function on the principles of intelligent generation of training scenarios, were developed and researched.

Keywords: intelligent scenario generation, training environment, simulator, Unreal Engine, C++, adaptive difficulty.

ЗМІСТ

ВСТУП

Актуальність посилення ролі штучного інтелекту та процедурної генерації у симуляторах на базі Unreal Engine зумовлює необхідність переходу від малоефективного ручного формування сценаріїв до автоматизованих рішень, здатних забезпечити високу адаптивність та персоналізацію навчання.

Традиційні методи розробки тренувального контенту (сценаріїв та 3D-середовищ) є надзвичайно ресурсомісткими, статичними та не здатні забезпечити необхідної варіативності для повноцінного тренування.

Розв'язання цієї проблеми лежить у сфері інтелектуальної генерації контенту, яка дозволяє не лише автоматично створювати тривимірні об'єкти, але й динамічно формувати цілісні тренувальні сценарії, адаптуючи їхню складність до рівня підготовки користувача в реальному часі.

Метою магістерської кваліфікаційної роботи є розробка архітектури, реалізація та експериментальна валідація програмного забезпечення навчального середовища для симуляторів на базі Unreal Engine, на основі удосконаленого методу інтелектуальної генерації тренувальних сценаріїв, що функціонує за принципом інтелектуальної генерації адаптивних тренувальних сценаріїв.

Для досягнення мети роботи необхідно вирішити такі завдання:

— провести системний аналіз існуючих підходів до інтелектуальної генерації сценаріїв та адаптивних навчальних систем;

— обґрунтувати та вдосконалити алгоритм інтелектуальної генерації тренувальних сценаріїв з урахуванням моделі адаптивної складності;

— розробити архітектуру програмного комплексу та реалізувати його ключові модулі, використовуючи можливості Unreal Engine та мову C++;

— створити підсистему збору, аналізу та оцінки ефективності тренувального процесу для зворотного зв'язку з модулем генерації;

— провести експериментальні дослідження для валідації якості та адаптивності генерованих середовищ і сценаріїв, а також підтвердити практичну ефективність розробленої системи;

— здійснити техніко-економічне обґрунтування розробки та оцінити її економічну ефективність.

Об'єкт дослідження — процес формування та функціонування навчальних середовищ у симуляторах.

Предмет дослідження — методи та алгоритми інтелектуальної генерації тривимірних навчальних середовищ і адаптивних тренувальних сценаріїв в Unreal Engine.

Методи дослідження включають:

— системний аналіз та синтез, теорія процедурної генерації;

— об'єктно-орієнтований аналіз та програмування;

— теорія ігор та симуляцій;

— методи експериментальної валідації та статистичного аналізу даних.

Новизна полягає в тому, що удосконалено метод інтелектуальної генерації тренувальних сценаріїв, який, на відміну від існуючих, забезпечує динамічну адаптацію як геометричних параметрів середовища, так і логіки тренувальних подій, на основі безперервної оцінки рівня компетентності користувача.

Практичне значення отриманих результатів полягає в можливості бути використаним як готовий інструмент для швидкого створення тренувальних симуляторів у сферах військової підготовки, медицини, промисловості чи безпеки.

Апробація результатів роботи здійснена у доповіді на Молодь в науці: дослідження, проблеми, перспективи (МН-2026).

Матеріали роботи доповідались та опубліковувались [1]:

Публікація Чорний В.В., Колесніченко Л.А. , Черняк О.І. «Інтелектуальний генератор навчальних середовищ і сценаріїв для симуляторв у Unreal Engine». Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)», Вінниця, 2025. [Електронний ресурс] Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26513

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Огляд існуючих симуляторів та навчальних середовищ

Сучасні методи професійної підготовки та підвищення кваліфікації у військових, медичних, інженерних та промислових сферах дедалі частіше спираються на використання симуляційних навчальних середовищ. Ці програмно-апаратні комплекси забезпечують контрольоване, безпечне та економічно ефективне відтворення складних ситуацій, недоступних або занадто ризикованих для тренування в реальних умовах. Розвиток технологій, зокрема ігрових рушіїв високої реалістичності (таких як Unreal Engine), дозволив симуляційному навчальному середовищу перейти від простих 2D-моделей до високоточних, фотореалістичних тривимірних віртуальних світів. Проте, перш ніж перейти до розробки власного адаптивного середовища, необхідно провести систематизацію та аналіз поточного стану розвитку СНС, що здійснюється через їхню класифікацію та оцінку ключових вимог.

Сучасні тренувальні симуляційні платформи стали невіддільною частиною підготовки фахівців у галузях, де практичні навички та швидкість прийняття рішень мають критичне значення. На відміну від традиційних навчальних програм, симулятори надають можливість взаємодії з динамічним середовищем, що реагує на дії користувача, моделюючи реальні ситуації з високим ступенем достовірності.

Використання симуляторів дозволяє:

— знизити витрати на тренування та обладнання;

— мінімізувати ризики, пов’язані з виконанням небезпечних операцій у реальному середовищі;

— забезпечити повторюваність тренувальних сценаріїв з можливістю варіації умов;

— автоматизувати фіксацію помилок та прогресу користувача.

Ключовою тенденцією є перехід від статичних симуляцій до інтелектуальних адаптивних навчальних систем, здатних змінювати сценарії у реальному часі залежно від дій користувача. Подібні системи широко застосовуються у сфері

оборони, охорони праці, медицини, автомобільної та авіаційної підготовки, а також у промислових процесах [2].

Важливою особливістю сучасних симуляторів є поєднання візуалізації, фізичного моделювання та алгоритмів машинного навчання, що забезпечує високу якість навчального процесу і можливість поступового підвищення складності виконуваних завдань відповідно до рівня підготовки користувача.

1.1.1 Класифікація симуляційних навчальних середовищ

Сучасні симуляційні навчальні середовища являють собою програмно-апаратні комплекси, призначені для формування, тренування та оцінки професійних навичок користувачів в умовах, максимально наближених до реальних [3]. Для адекватного аналізу предметної області системи навчальних симуляцій доцільно класифікувати за трьома ключовими ознаками: рівнем реалізму, сферою застосування та ступенем адаптивності.

За технологічною базою виділяють симулятори низької точності, які переважно представлені десктопними або 2D-рішеннями, середньої точності з 3D-графікою та обмеженою фізикою, а також високої точності, що використовують передовий рендеринг, складні фізичні моделі та часто інтегруються з апаратним забезпеченням.

За сферою застосування ці системи поділяються на військові та тактичні, інженерні й промислові, а також медичні симулятори. Окрім того, важливим критерієм є ступінь інтерактивності, за яким розрізняють статичні, динамічні та найбільш прогресивні — адаптивні симулятори.

У контексті розвитку віртуальних навчальних систем слід відзначити, що високоточні симулятори набувають особливого значення завдяки здатності моделювати складні технологічні процеси та забезпечувати реалістичну взаємодію користувача зі штучним середовищем. Окрім графічної складової, такі системи інтегрують моделі поведінки об’єктів, складні системи зіткнень, модулі імітації сенсорної взаємодії та інші компоненти, що забезпечують багатовимірність навчального простору.

Додатково, новітні симуляційні середовища використовують системи адаптивного контролю складності, керування сценаріями та алгоритми відстеження поведінки користувача. Це дозволяє формувати динамічну навчальну траєкторію, у якій зміна сценаріїв відбувається з урахуванням прогресу та допущених помилок. Такий підхід відповідає сучасним принципам персоналізованого навчання, що підтримуються міжнародними стандартами професійної підготовки.

Реалістичність симуляції забезпечується також завдяки сучасним фізичним рушіям, розвиненим системам освітлення, моделюванню поведінки матеріалів, анімаційним технологіям та інтеграції штучного інтелекту. Це сприяє створенню умов, у яких користувач отримує досвід, максимально наближений до реальних виробничих або оперативних ситуацій.

1.1.2 Аналіз вимог до сучасного тренувального середовища

Ефективність сучасного симуляційного навчального середовища визначається його здатністю забезпечувати не лише реалізм, а й педагогічну доцільність процесу навчання. На основі аналізу світових стандартів та наукових досліджень у галузі симуляцій, було визначено п'ять ключових вимог до СНС, які лягли в основу розробки даної магістерської роботи:

Реалізм та фізична достовірність:

— середовище повинно максимально точно відтворювати візуальні, акустичні та фізичні характеристики реального світу;

— використання Unreal Engine обґрунтоване саме цією вимогою, оскільки його вбудовані системи забезпечують високий рівень симуляційного реалізму.

Варіативність та унікальність контенту:

— система має бути здатною генерувати необмежену кількість унікальних тренувальних сценаріїв та конфігурацій середовища;

— унікальність запобігає ефекту заучування, коли користувач запам'ятовує послідовність подій, а не засвоює загальні принципи.

Динамічна адаптивність та реактивність:

— система повинна в режимі реального часу аналізувати дії користувача та автоматично коригувати рівень складності (D) сценарію;

— адаптація дозволяє підтримувати користувача у зоні оптимального навчання – коли завдання є достатньо складним, щоб вимагати зусиль, але не настільки, щоб викликати фрустрацію.

Об'єктивний моніторинг та оцінка:

— симуляційне навчальне середовище повинно мати вбудовану підсистему для точного збору метрик та об'єктивного переведення цих метрик у показник поточної компетентності C_{user};

— без точного моніторингу неможлива ефективна адаптація.

Модульність та Розширюваність Архітектури:

— програмна архітектура повинна бути модульною для легкої заміни компонентів та інтеграції нових функціональних можливостей;

— розробка в UE на C++ дозволяє створити незалежні модулі, що забезпечує гнучкість при подальшому розвитку продукту.

Важливо підкреслити, що сучасні вимоги до тренувальних симуляторів формуються не лише технологічними можливостями, але й педагогічними принципами побудови навчального процесу. У системах професійної підготовки все більшу увагу приділяють активному навчанню, коли користувач не просто спостерігає або повторює, а взаємодіє із середовищем, аналізує наслідки власних дій та отримує негайний зворотний зв’язок.

Одним із ключових компонентів сучасного тренажера є підтримка різних типів навчальних сценаріїв, які охоплюють:

— базове ознайомлення з процесом — вивчення теоретичних основ і базових принципів взаємодії;

— кероване тренування — виконання завдань під контролем системи або інструктора;

— самостійне відпрацювання — виконання операцій у динамічному середовищі без підказок;

— аварійні та ймовірнісні сценарії — реагування на нестандартні та критичні ситуації.

Ще однією важливою характеристикою є можливість комплексної оцінки компетентності, яка включає:

— точність та правильність дій;

— швидкість ухвалення рішень;

— стабільність результатів у повторних сесіях;

— здатність працювати у змінних умовах;

— мінімізацію критичних помилок.

Сучасні тренувальні системи також дедалі частіше інтегрують користувацькі моделі поведінки, що дозволяє оцінювати когнітивні характеристики студента: рівень уваги, адаптивність, здатність до прогнозування та ситуаційної оцінки.

Поєднання педагогічних методик з технічними можливостями інтелектуальних систем створює передумови для реалізації тренажерів, що не лише відтворюють фізичні процеси, але й забезпечують формування професійного мислення та навичок прийняття рішень.

1.1.3 Порівняльний аналіз комерційних та відкритих рішень

Для обґрунтування наукової новизни та вибору технологічного шляху необхідно провести порівняльний аналіз найбільш поширених комерційних та відкритих симуляційних рішень, зважаючи на їхню здатність до процедурної генерації та інтелектуальної адаптації.

Комерційні симулятори (табл. 1.1) забезпечують високий реалізм, проте вони мають критичний недолік — статичність або обмеженість логіки тренувальних сценаріїв.

Їхня архітектура є закритою, що унеможливлює впровадження та дослідження власних, нових моделей інтелектуального генератора та адаптації, як це потрібно для магістерської роботи.

Таблиця 1.1 — Комерційні військові та професійні симулятори

Рішення	Платформа/ База	Генерація Контенту	Адаптивність Сценаріїв	Обмеження для МКР
VBS4 (Virtual Battlespace)	Власне ядро (Bohemia Interactive)	Обмежена ПКГ для створення ландшафту та базових об'єктів.	Низька Адаптація переважно ручна (зміна параметрів оператором).	Висока вартість, закритий код, неможливість впровадження власних C++ алгоритмів.
Presagis VAPS XT / STAGE	Спеціалізоване ПЗ	Спеціалізовані інструменти для моделювання, елементи ландшафту.	Відсутня інтелектуальна адаптація; фокус на моделюванні поведінки.	Висока складність інтеграції, непризначеність для глибокої алгоритмічної генерації.
Microsoft Flight Simulator	Власне ядро (Asobo)	Високий рівень генерації для глобального ландшафту (на основі супутникових даних).	Відсутня. Симулятор статичний, призначений для відтворення реальних умов.	Не фокусується на тренувальних сценаріях та адаптивній логіці.

Ігрові рушії (табл. 1.2), зокрема Unreal Engine, пропонують ідеальну платформу для наукового дослідження. Вони забезпечують необхідний рівень реалізму та дозволяють впровадити та протестувати власне ядро на високорівневій мові C++, що є необхідною умовою для досягнення наукової новизни даної роботи.

Наявні рішення не містять комплексного, динамічного механізму, здатного адаптувати як середовище, так і логіку сценарію на основі безперервної оцінки

компетентності користувача C_{user}. Це підтверджує, що розробка такого Модуля інтелектуальної генерації сценаріїв є актуальною науковою задачею.

Таблиця 1.2 — Ігрові рушії та відкриті рішення

Рішення	Платформа / База	Генерація Контенту	Адаптивність Сценаріїв	Переваги
Unity Engine	C# / .NET	Потребує розробки з нуля, є готові ассети.	Можлива через C# скрипти та пакети ML-Agents.	Гнучкість, наявність готових інструментів для Machine Learning.
Unreal Engine 5	C++/ Blueprints	Висока продуктивність, наявність вбудованих інструментів та системи Blueprints.	Потребує розробки Модуля ІГС на C++	Високий фотореалізм, нативна підтримка C++ для продуктивних обчислень, модульність архітектури.

У сучасному технологічному середовищі спостерігається активна конкуренція між платформами, орієнтованими на створення інтерактивних навчальних рішень. Поряд із комерційними продуктами значну роль відіграють відкриті інструменти, що дозволяють розробникам експериментувати та впроваджувати нові підходи до моделювання складних процесів.

Комерційні універсальні рушії, такі як Unreal Engine та Unity, забезпечують широкий набір інструментів для 3D-моделювання, фізичного моделювання, роботи з матеріалами та анімацією. Їх ключовою перевагою є наявність зрілої екосистеми, розвиненої документації та великої спільноти користувачів, що спрощує процес впровадження складних функцій.

Окрему нішу займають спеціалізовані симуляційні платформи, включаючи VBS4, Omniverse, DCS World та Simulink-відповідні комплекси реального часу, які орієнтовані на промислові, військові та дослідницькі сфери. Такі системи пропонують підвищену точність, підтримку спеціалізованих протоколів і

можливість інтеграції з апаратними тренажерами, однак мають високу вартість та обмежену гнучкість щодо модифікації внутрішньої логіки.

Відкриті рушії та наукові симулятори, як-от Godot, Gazebo, CARLA Simulator, AirSim, надають високий рівень кастомізації та доступ до вихідного коду, що робить їх придатними для експериментальних досліджень і навчальних проєктів. Проте, вони часто потребують значно більшої кваліфікації для налаштування та менш зрілої інфраструктури підтримки.

Додатково слід відзначити, що при виборі рушія для розробки навчального тренажера важливу роль відіграють не лише функціональні можливості, а й підтримка сучасних інструментів оптимізації, адаптації під різні апаратні платформи та засоби інтеграції зовнішніх модулів.

Unreal Engine підтримує масштабовану архітектуру, що дозволяє ефективно працювати як на високопродуктивних комп'ютерах з VR-периферією, так і на менш потужних системах. Вбудований набір SDK та інструментів дає змогу організувати мережеву взаємодію, використовувати модулі штучного інтелекту, підключати зовнішні сенсори, а також інтегрувати системи телеметрії та збору статистики.

Особливістю рушія є поєднання візуального програмування Blueprint та класичного C++, що забезпечує гнучкість під час реалізації логіки проєкту. Це суттєво спрощує процес адаптації системи під різні навчальні сценарії й дозволяє залучати до проєкту як дослідників, так і студентів без глибокого досвіду програмування.

Завдяки активному розвитку екосистеми Unreal Engine постійно отримує нові інструменти моделювання, симуляції фізичних властивостей, генерації оточення, а також системи обробки анімацій та інтелектуальної поведінки агентів. Це створює передумови для реалізації тренажера, який може не лише імітувати базові дії, але й моделювати складні технологічні процеси, що особливо важливо в умовах сучасних освітніх викликів.

1.2 Аналіз методів процедурнох та інтелектуальної генерації контенту

Вимога до сучасного тренувального середовища щодо варіативності та

адаптивності може бути задоволена лише шляхом впровадження автоматизованих методів створення контенту. Ці методи поділяються на дві основні групи: процедурна генерація, яка фокусується на геометрії та текстурах, та інтелектуальна генерація, яка відповідає за логіку та адаптацію сценарію.

Сучасні технології процедурної та інтелектуальної генерації контенту забезпечують можливість значного прискорення розробки інтерактивних тренажерів, знижуючи витрати на ручне моделювання та сприяючи створенню варіативних сценаріїв (рис 1.1). Це особливо важливо у випадках, коли обсяг даних великий, а необхідність адаптації навчального середовища до рівня користувача потребує динамічної побудови віртуального простору.

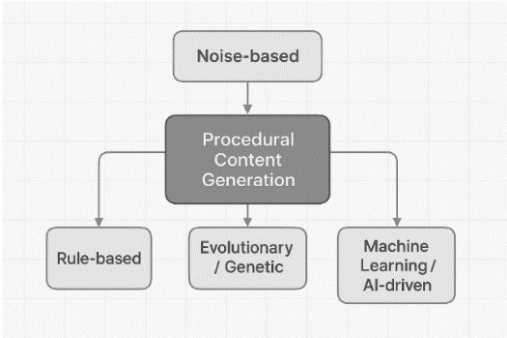

Рисунок 1.1 — Схема класифікації методів процедурної генераціях

Процедурні методи побудовані на принципах математичного моделювання та симуляції природних процесів: генерація ландшафтів, рослинності, систем частинок, а також створення повторюваних об’єктів і архітектурних структур. Вони забезпечують високу узагальненість та відтворюваність результатів, що дозволяє автоматично генерувати контент без втрати логічної цілісності сцени [5].

Інтелектуальна генерація, у свою чергу, базується на алгоритмах машинного навчання, оптимізації та штучного інтелекту, що дозволяє підлаштовувати середовище до дій користувача, генерувати адаптивні сценарії та створювати складні поведінкові моделі агентів. Це сприяє формуванню гнучкої системи навчання, здатної змінювати зміст і складність завдань у реальному часі.

Таким чином, поєднання процедурних і інтелектуальних технологій відкриває можливість створення масштабованих, динамічних та педагогічно

доцільних навчальних систем.

1.2.1 Огляд методів процедурної генерації

Процедурна генерація контенту — це сукупність методів, які використовують алгоритми для створення ігрових ресурсів (ландшафтів, рівнів, об'єктів, текстур) на основі вхідних параметрів та математичних правил, а не шляхом ручного моделювання.

Процедурна генерація охоплює широкий спектр методів, спрямованих на автоматизоване створення віртуального контенту на основі формальних правил, математичних функцій та алгоритмічних моделей. Завдяки можливості відтворення складних структур із мінімальним обсягом вихідних даних, ці технології дозволяють досягти високого рівня деталізації без надмірних витрат на ручне моделювання.

До найпоширеніших класів процедурних методів належать (рис 1.2):

— алгоритми на основі шумових функцій, що застосовуються для моделювання природних форм, наприклад рельєфів, хмар та текстур;

— фрактальні алгоритми, які використовуються для формування складних геометричних структур із властивостями самоподібності, таких як гірські ландшафти або деревоподібні об'єкти;

— rule-based генерація, що спирається на формальні правила та граматики (L-системи, граматики для архітектури), дозволяє створювати рослинність, будівлі та інші структуровані об’єкти;

— node-based системи, які застосовуються у графічних рушіях для побудови процедурних матеріалів, геометрії та логіки ґенерації за допомогою вузлів та операцій;

— просторова декомпозиція (BSP-дерева, октодерева, грід-поділ), що забезпечує структуровану побудову середовищ, наприклад рівнів або внутрішніх приміщень;

— агентно-орієнтовані підходи, де об’єкти діють за набором правил,

формуючи геометричні або середовищні структури у процесі виконання.

Рисунок 1.2 — Порівняння підходів процедурної генерації

Поєднання цих технік дозволяє створювати як природні, так і техногенні об'єкти із заданою стилістикою та функціональністю, а також забезпечує керованість кінцевого результату і його відповідність моделям реального світу [6].

Ключові методи процедурної генерації, які часто застосовуються в симуляційних середовищах:

— алгоритми на основі шуму;

— застосування;

— клітинні автомати;

— l-системи.

Хоча процедурно генерація є ідеальною для створення геометрії та текстур, вона є агностичною до логіки та цілей тренування. Генерація не може самостійно вирішити, де розмістити критичну перешкоду, як змінити поведінку ворога, або коли ускладнити завдання, оскільки їй бракує механізму зворотного зв'язку з діями користувача. Це обґрунтовує необхідність переходу до інтелектуальної генерації.

У контексті навчальних тренажерів процедурні методи відіграють особливу роль, оскільки дають можливість автоматично створювати варіативні навчальні ситуації та динамічні середовища. Наприклад, зміна конфігурації простору, випадкове розташування об'єктів або генерування унікальних задач для кожного користувача підвищують ефективність тренування, сприяють розвитку

адаптивного мислення та формують стійкість до непередбачуваних умов.

Крім того, процедурні алгоритми дозволяють уникнути шаблонності поведінки користувача, оскільки повторне проходження тренувального сценарію не гарантує ідентичних умов. Це наближує навчальний процес до реальної професійної діяльності, де обставини рідко повторюються у незмінному вигляді [7].

1.2.2 Методи інтелектуальної генерації контенту та адаптивних сценаріїв

Інтелектуальна генерація контенту — це розширення процедурної генерації, що інтегрує методи штучного інтелекту, пошуку або машинного навчання для створення контенту, який відповідає високорівневим, абстрактним вимогам (наприклад, "зробити сцену на 20% складнішою" або "сформувати сценарій для тренування навику X").

Інтелектуальна генерація контенту ґрунтується на застосуванні алгоритмів штучного інтелекту та машинного навчання, що дозволяють системі створювати нові об’єкти, середовища, ситуації або сценарії на основі попереднього досвіду та навчальних даних. На відміну від класичних процедурних моделей, які діють за заздалегідь визначеним набором правил, інтелектуальні системи здатні адаптуватися, узагальнювати інформацію та формувати контент, який не був явно описаний розробниками [8].

Основою таких підходів є моделі, здатні навчатися на великих об’ємах даних, аналізуючи структуру, стиль, логіку побудови об’єктів і поведінку користувачів у віртуальному середовищі. Це забезпечує здатність генератора не тільки відтворювати відомі шаблони, але й створювати нові комбінації, що підвищує реалістичність та унікальність результатів.

До найбільш поширених методів інтелектуальної генерації належать:

— генеративно-змагальні мережі;

— варіаційні автоенкодери;

— дифузійні моделі;

— мовні та мультимодальні трансформери;

— RL-моделі;

— нейронні поля та імпліцитні представлення.

Поєднання цих підходів дозволяє створювати адаптивні навчальні середовища, де контент змінюється залежно від рівня підготовки користувача, виконуваних помилок та цілей навчального процесу [9, 10].

Ключові підходи, релевантні для адаптивного симулятора:

— генерація на основі обмежень;

— адаптивна генерація на основі моделі користувача;

— моделі користувача;

— керуючий алгоритм;

— генерація на основі прикладів.

У навчальних тренажерах інтелектуальна генерація відіграє ключову роль у формуванні адаптивних сценаріїв та персоналізованих навчальних траєкторій. Система може автоматично створювати навчальні завдання різної складності, симулювати поведінку віртуальних інструкторів або супротивників, генерувати унікальні виробничі ситуації чи аварійні події. Це дозволяє наблизити тренування до реальних умов, де рішення приймаються в умовах невизначеності, а сценарії рідко повторюються [11, 12].

Такі підходи сприяють формуванню критичного мислення, адаптивної реакції на змінні умови та здатності оцінювати ризики, що є важливими елементами професійної підготовки у технічних та оперативних спеціальностях.

1.2.3 Визначення ключових метрик для оцінки якості генерації

Оцінювання якості автоматично згенерованого контенту є критичним аспектом при розробці навчальних тренажерних систем, оскільки від адекватності та точності генерації залежить ефективність освітнього процесу. Метрики дозволяють визначити, наскільки створений контент відповідає поставленим навчальним завданням, рівню складності, а також реалістичним параметрам

віртуального середовища [13, 14].

Для процедурної генерації ключовим критерієм виступає відтворюваність і контрольованість структури, а також можливість формально описати загальні властивості результату. Водночас у випадку інтелектуальних методів особливу увагу приділяють адаптивності, здатності до узагальнення та рівню різноманітності контенту, що формується на основі статистичних закономірностей.

Оскільки ці два підходи мають різні принципи роботи, система оцінювання повинна враховувати обидві групи характеристик, зокрема структурні параметри, когнітивну складність задачі, ступінь відповідності педагогічним вимогам та стабільність роботи генератора в умовах змінних сценаріїв [15].

Оцінювання якості генерації контенту в навчальних системах базується на кількох ключових напрямах. Структурні показники включають ступінь цілісності середовища, коректність фізичних властивостей об'єктів та детермінованість результату. Педагогічний аспект оцінюють за відповідністю складності навчальним цілям, поступовістю завдань та здатністю підтримувати активне мислення. Адаптивність визначається різноманітністю сценаріїв, здатністю системи підлаштовуватися під рівень користувача та стабільністю роботи. До метрик продуктивності належать швидкість генерації, обсяг ресурсів та загальна оптимальність (рис. 1.3).

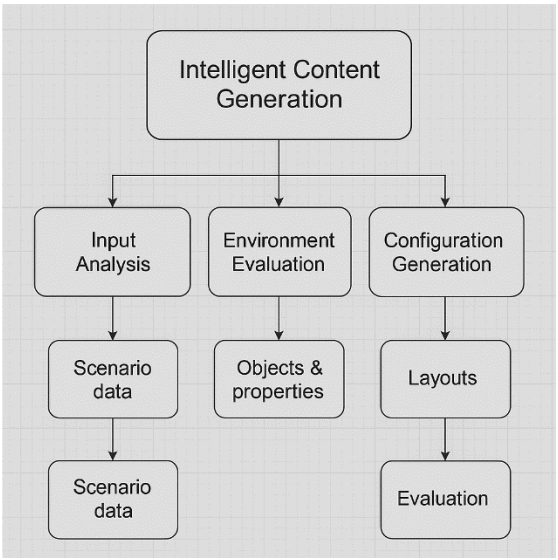

Рисунок 1.3 — Узагальнена схема інтелектуальної генерації проекту

Для ефективної роботи модуля інтелектуальної генерації окремо виділяють критерії якості контенту (когерентність, варіативність) та ефективності адаптації (чутливість, стабільність, таргет-відхилення).

Поєднання структурних, когнітивних та динамічних метрик дозволяє об’єктивно оцінити коректність, реалістичність і навчальну цінність згенерованого контенту, що є необхідним для побудови сучасних тренажерних платформ [16,17].

1.3 Обґрунтування вибору платформи Unreal Engine та мови C++

Вибір технологічного стеку для розробки симуляційного навчального середовища є критичним кроком, оскільки він має забезпечити відповідність ключовим вимогам — високий реалізм, продуктивність алгоритмів генерації та модульність архітектури. Обґрунтування вибору платформи Unreal Engine та мови C++ базується на їхніх технічних перевагах перед іншими комерційними та відкритими рішеннями [18, 19].

На відміну від багатьох альтернативних платформ, Unreal Engine поєднує двигун реального часу з можливістю глибокої низькорівневої розробки, що дає змогу реалізувати специфічні освітні алгоритми, логіку взаємодії та моделі тренувальних процесів без суттєвих обмежень.

1.3.1 Переваги Unreal Engine для симуляції

Важливою перевагою Unreal Engine є наявність просунутого графічного ядра, що дозволяє формувати високодеталізовані сцени та візуалізувати реалістичні фізичні процеси, що особливо важливо для навчальних симуляторів, орієнтованих на відтворення реальних умов. Підтримка сучасних технологій освітлення, систем частинок, анімацій та взаємодії з об’єктами забезпечує можливість створення середовища, яке сприяє кращому зануренню користувача в навчальний процес.

Крім того, Unreal Engine передбачає готові інструменти для роботи з VR та AR, що робить його оптимальним для створення тренажерів, де критичним фактором є імітація просторового сприйняття, моторної координації та реакції на візуальні сигнали [20, 21].

Unreal Engine є потужним інструментом, який ідеально підходить для створення High-Fidelity Simulators (симуляторів високої точності):

— високий фотореалізм;

— надійна фізична підсистема;

— система навігації та AI;

— масштабованість;

— blueprint system.

1.3.2 Обґрунтування використання C++

Однією з суттєвих переваг Unreal Engine є можливість використання мови C++ для написання ігрової логіки та низькорівневих модулів. Використання C++ забезпечує:

— доступ до внутрішніх структур рушія та API;

— високу продуктивність виконання алгоритмів;

— можливість оптимізації системи під специфічні обчислювальні задачі;

— гнучкість у створенні нестандартної логіки навчання та поведінки об’єктів.

У контексті розробки навчального тренажера це дозволяє створити систему, яка не лише візуально реалістична, але й функціонально точна, забезпечуючи коректну обробку фізичних взаємодій, сценарного менеджменту та адаптивних педагогічних моделей.

Вибір C++ як основної мови для реалізації ядра інтелектуальної генерації сценаріїв обґрунтований наступними технічними та архітектурними міркуваннями:

— висока продуктивність;

— нативна інтеграція;

— модульність та розширюваність.

Комбінація Unreal Engine для графічного та фізичного реалізму та C++ для реалізації високопродуктивного модуля інтелектуальної генерації сценаріїв є оптимальною для виконання завдань даної магістерської роботи. Обґрунтований

вибір цієї технологічної платформи створює фундамент для побудови високоякісного інтерактивного тренажера із підтримкою складних сценаріїв, реалістичної фізики та інтелектуальних компонентів навчання [22, 23, 24].

2 МЕТОДИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ГЕНЕРАЦІЇ СЦЕНАРІЇВ

2.1 Розробка загальної архітектури навчального середовища

Проєктування навчального середовища для тренажерних систем базується на принципах модульності, масштабованості та розширюваності, що забезпечує ефективну розробку, адаптацію та подальше вдосконалення програмного комплексу. Архітектура програмної системи повинна забезпечувати чітке розділення відповідальностей між її компонентами, підтримувати інтеграцію інтелектуальних модулів та гарантувати стабільну роботу навіть за умови динамічної зміни сценаріїв (рис 2.1).

В основі архітектури лежить багаторівневий підхід, що дозволяє розподілити функціональність відповідно до завдань окремих підсистем: симуляції фізичних процесів, генерації контенту, управління сценаріями, збору та аналізу даних тренувального процесу, а також забезпечення взаємодії з користувачем. Такий підхід сприяє спрощенню розробки, підвищенню гнучкості системи та забезпечує можливість паралельної роботи декількох модулів.

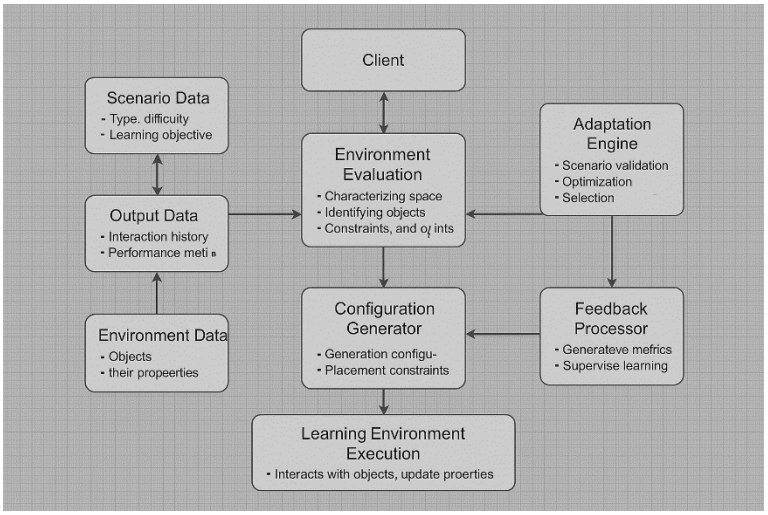

Рисунок 2.1— Архітектура системи інтелектуальної генерації

Ключовим завданням архітектури є забезпечення можливості динамічної адаптації навчального середовища відповідно до рівня та дій користувача. Це досягається за рахунок інтеграції інтелектуального контролера, що може

модифікувати логіку сценарію, додавати або видаляти об’єкти, змінювати складність завдань та аналізувати результати взаємодії користувача у реальному часі.

Оскільки навчальні тренажери можуть використовуватися в галузях із високими вимогами до реалістичності та відтворення реальних процесів, програмна архітектура повинна містити компоненти для точної симуляції фізики, візуалізації та анімації дій об’єктів. Саме тому використання Unreal Engine як базового рушія забезпечує високу достовірність моделювання та дозволяє ефективно управляти викликами, пов’язаними з обчислювально складними задачами.

Загалом, запропонована архітектура поєднує в собі гнучкість модульно орієнтованого підходу та інтелектуальні можливості адаптації контенту. Це створює передумови для реалізації масштабованої системи, здатної підтримувати широкий спектр навчальних сценаріїв, а також забезпечує надійний фундамент для подальшого розвитку функціоналу та інтеграції нових компонентів.

2.1.1 Модульна структура програмного комплексу

Модульна структура програмного комплексу дозволяє розділити функціональність системи на окремі компоненти, кожен з яких відповідає за певний аспект роботи тренажера. Такий підхід спрощує процес розробки, тестування та подальшої модернізації, оскільки зміни в одному модулі не вимагають модифікації всієї системи.

Для навчального середовища це особливо важливо, тому що окремі модулі можуть відповідати за обробку фізичних процесів, управління сценаріями, генерацію контенту, поведінку віртуальних агентів та взаємодію з користувачем. Це забезпечує високу гнучкість системи та дає змогу адаптувати її під різні типи тренувальних сценаріїв.

Архітектура програмного комплексу базується на принципах розділення відповідальності, де кожен модуль виконує чітко визначені функції. Зокрема, основні модулі включають:

— ядро симуляції, що відповідає за відтворення фізичних процесів та базових механік взаємодії;

— модулі інтелектуальної генерації контенту, які забезпечують автоматичне створення середовища та навчальних сценаріїв;

— підсистему керування сценаріями, що контролює послідовність подій та адаптацію складності;

— модулі збору та аналізу даних тренувального процесу, які фіксують дії користувача та обчислюють ключові метрики ефективності;

— модулі інтерфейсу користувача, що забезпечують інтерактивну взаємодію з тренажером.

Така ієрархія модулів дозволяє формувати середовище, в якому кожен компонент може незалежно оновлюватись або замінюватись, не порушуючи роботу системи загалом.

Реалізація модульної структури у межах Unreal Engine базується на використанні об’єктної моделі рушія, включно з компонентами Actor, Pawn, Controller, а також системою Blueprint для швидкої побудови логіки. Кожен модуль може бути реалізований у вигляді окремого класу або Blueprint-компонента, що дозволяє адаптувати систему під конкретні вимоги навчального сценарію.

Застосування нативного коду на C++ забезпечує гнучкість та можливість оптимізації обчислювально складних процесів, тоді як система Blueprint спрощує побудову візуальної логіки та дозволяє швидко змінювати поведінку окремих об’єктів без необхідності перекомпіляції всього проєкту.

Архітектура програмного комплексу побудована на принципі інверсії залежностей, де ядро симуляції слугує базовим фундаментом, а інтелектуальна логіка структурована у незалежні шари: рівень генерації та керування, представлений модулями процедурної генерації та інтелектуальної генерації сценаріїв, і рівень аналізу та взаємодії, що включає модулі M&O та КІ. Така модульна організація створює основу для гнучкого та масштабованого навчального середовища, суттєво спрощуючи реалізацію інтелектуальних механізмів, подальше

розширення функціоналу, тестування та технічну підтримку, що є визначальним фактором для забезпечення довготривалої та стабільної експлуатації тренажера.

2.1.2 Схема взаємодії компонентів

Схема взаємодії компонентів навчального середовища визначає логіку обміну даними між основними підсистемами програмного комплексу та забезпечує узгоджене функціонування модулів під час виконання тренувальних сценаріїв. Наявність чітко визначених каналів взаємодії є критично важливою для забезпечення стабільності системи, можливості масштабування та подальшої модифікації (додаток В).

У межах розроблюваного середовища архітектура передбачає багаторівневу модель взаємодії, що включає рівень ядра симуляції, рівень генерації контенту та рівень управління сценаріями, а також підсистеми збору даних і користувацького інтерфейсу.

Основні компоненти системи взаємодіють за принципом інверсії залежностей, де високорівневі модулі керують роботою низькорівневих, а не навпаки. Таке рішення забезпечує більш гнучку побудову системи та дозволяє адаптувати поведінку окремих компонентів без зміни базової архітектури.

Ключові взаємодії в системі формують цілісний цикл обміну даними, що розпочинається з передачі ядром симуляції інформації про стан середовища до модуля генерації контенту. На основі цих даних модуль ініціює створення нових подій та об’єктів у підсистемі сценаріїв, яка, своєю чергою, транслює інструкції та зворотний зв’язок через інтерфейс користувача. Паралельно підсистема збору даних фіксує перебіг тренування та передає параметри до аналітичного модуля, який після обчислення метрик замикає контур управління, здійснюючи зворотний вплив на генератор контенту для динамічної адаптації складності.

Типова послідовність виконання тренувального сценарію включає:

— ініціалізацію середовища — ядро завантажує сцену, об’єкти та базові параметри фізичної симуляції;

— генерування контенту — модуль процедури/AI формує конфігурацію середовища або адаптує її відповідно до попередніх результатів користувача;

— виконання сценарію — система управління сценаріями активує необхідні події, контролює їх послідовність та відслідковує прогрес;

— фіксацію параметрів — підсистема збору даних реєструє ключові показники взаємодії користувача;

— аналіз результатів — аналітичний модуль обробляє отримані дані та генерує звіт, який може вплинути на подальшу адаптацію тренування.

Така схема взаємодії компонентів забезпечує стабільний і гнучкий механізм роботи системи, де окремі модулі можуть бути незалежно оновлені або замінені. Це дає змогу поступово розширювати функціонал тренажера, інтегрувати нові алгоритми генерації та адаптації, а також забезпечувати персоналізовану траєкторію навчання.

Взаємодія компонентів системи реалізована за принципом замкнутого циклу зворотного зв'язку, що імітує роботу режисера і забезпечує динамічну корекцію параметрів сценарію з мінімальною затримкою. Процес розпочинається з етапу ініціалізації, коли модуль процедурної генерації формує карту, а система інтелектуальної генерації наповнює її початковими об'єктами відповідно до базового рівня складності, створюючи умови для активної взаємодії користувача із середовищем.

У процесі виконання завдання відбувається безперервний збір та аналіз даних. Дії користувача, такі як допущені помилки, швидкість реакції та витрачений час, фіксуються ядром Unreal Engine та передаються до модуля моніторингу. Отримані метрики надходять до генератора сценаріїв, де система обчислює поточний рівень компетентності та визначає градієнт адаптації, порівнюючи реальні показники з очікуваними.

На завершальному етапі циклу відбувається прийняття рішень та корекція середовища. Якщо аналіз виявляє необхідність змін, генератор сценаріїв формує та передає ядру рушія керуючі команди — від створення чи видалення об'єктів до

зміни агресивності NPC або часових лімітів. Після виконання цих команд цикл повторюється, що дозволяє утримувати тренувальний сценарій у зоні оптимального навчального навантаження протягом усього сеансу.

2.2 Удосконалення методу інтелектуальної генерації

Удосконалення методу інтелектуальної генерації контенту у навчальних тренажерах є ключовим етапом підвищення ефективності навчального процесу та адаптивності системи до індивідуальних особливостей користувача. Основною метою є формування механізму, здатного динамічно генерувати навчальні сценарії на основі оцінки поточного рівня підготовленості студента, історії його взаємодії із системою та заданих навчальних цілей.

Запропонований підхід передбачає використання комбінованої структури, у якій методи машинного навчання взаємодіють із модулем сценарного контролю, формуючи багаторівневу логіку створення контенту. На відміну від статичних шаблонних систем, інтелектуальна генерація дозволяє створювати варіативні ситуації, у яких змінюються об’єкти, їхні параметри, взаємозв’язки та зовнішні умови. Кожна спроба користувача відтворювати навчальне завдання стає унікальною, що сприяє формуванню стійких навичок і здатності діяти у непередбачуваних ситуаціях [25].

Модифікований підхід до генерації контенту передбачає можливість застосування моделей, що працюють за принципами reinforcement learning та нейронних трансформерів. Використання RL дозволяє системі адаптувати складність завдань відповідно до поточних успіхів користувача, формуючи баланс між рівнем виклику та досяжністю мети. З іншого боку, мультимодальні моделі здатні інтерпретувати та аналізувати контекстні параметри середовища, включаючи попередні стани та поведінкові патерни.

Важливою характеристикою удосконаленого методу є здатність до інтеграції з модулем аналізу результатів навчання. На основі накопичених даних система формує профіль користувача, у якому зберігаються ключові метрики: швидкість виконання операцій, кількість помилок, типові сценарії реакції та рівень

успішності у попередніх завданнях. Це дозволяє формувати прогноз складності наступних етапів тренування та адаптувати контент відповідно до змінних умов.

Ще однією перевагою запропонованого підходу є можливість створення умовно-безперервного навчального середовища, у якому зміна складності та контенту відбувається плавно, без необхідності ручного втручання інструктора. Завдяки цьому навчальний процес стає більш природним та інтуїтивним, а студенти можуть розвиватися у власному темпі.

Удосконалення методу інтелектуальної генерації забезпечує гнучкість навчального середовища, підвищує якість підготовки та створює передумови для реалізації персоналізованих навчальних траєкторій.

2.2.1 Формалізація сценарію як набору обмежень

Формалізація навчального сценарію як набору обмежень є ключовим етапом проєктування адаптивної тренажерної системи, оскільки саме вона дозволяє однозначно визначити цілі, умови та межі варіативності генерованого контенту. У межах такого підходу сценарій розглядається як сукупність параметрів, що регулюють конфігурацію віртуального середовища, перелік доступних дій, їхню послідовність, а також умови успішності виконання завдання.

Обмеження можна умовно поділити на кілька груп. По-перше, структурні обмеження, які визначають логічну організацію середовища: положення об’єктів, їхні взаємозв’язки та доступність ресурсів. По-друге, динамічні обмеження, що описують поведінку об’єктів у часі та можливі зміни їхніх параметрів під впливом дій користувача або зовнішніх умов. Третя група — контекстні обмеження, які включають вимоги до послідовності виконання завдань, рівня складності, типу сценарних подій, а також допустимих помилок.

Такий підхід забезпечує можливість побудови сценаріїв різної складності — від базових, де структура та послідовність дій жорстко фіксовані, до розширених, що передбачають широкий простір для варіативності та адаптації. Визначення обмежень також дозволяє автоматизувати процес генерації сценаріїв, оскільки

система може використовувати формальні правила для створення нових комбінацій параметрів, що відповідають заданим умовам [26].

Важливо зазначити, що формалізований опис сценаріїв слугує основою для подальшого аналізу результатів виконання завдань. Використовуючи набір обмежень, система може оцінювати, наскільки дії користувача відповідають необхідним критеріям, визначати рівень успішності та формувати індивідуальні рекомендації щодо подальшого навчання.

Перевагою підходу є його сумісність із інтелектуальними методами генерації, зокрема reinforcement learning та моделями прогнозування, які можуть використовувати структуру обмежень для оптимізації навчальних траєкторій. Це дозволяє формувати сценарії, що не тільки відповідають визначеним навчальним цілям, але й адаптуються відповідно до змін у поведінці користувача.

У даній роботі сценарій S розглядається не просто як набір акторів, а як розв'язок Проблеми Задоволення Обмежень. Сценарій S — це множина параметрів, що задовольняє певні вимоги:

$$S = \{P_{env}, P_{task}, P_{diff}\},$$

де P_{env} — геометричні та статичні параметри середовища (забезпечується ПКГ);

P_{task} — логічна послідовність завдань (наприклад, "Спочатку знайти, потім знешкодити");

P_{diff} — динамічні параметри складності, що генеруються ІГС.

Удосконалення полягає у введенні динамічних обмежень C_{diff}, які залежать від поточного рівня складності D. Наприклад, при D > 70:

$$C_{diff} = \{NPC_{Accuracy} \geq 0.8, Obstacle\_Density \geq 0.4\}.$$

Генератор шукає конфігурацію, яка задовольняє як статичним, так і цим динамічним обмеженням.

Формалізація сценарію як набору обмежень створює основу для побудови динамічної, адаптивної та педагогічно доцільної тренажерної системи, здатної забезпечити персоналізоване навчання та контроль ефективності виконання завдань.

2.2.2 Модель адаптивної складності

Модель адаптивної складності визначає механізм динамічної зміни умов тренувального сценарію відповідно до рівня підготовленості користувача, його поточних дій та результатів, отриманих на попередніх етапах навчання. Основною метою такої моделі є забезпечення балансу між доступністю та викликом: завдання не повинні бути надто простими, щоб не втратити навчальної цінності, але й не надмірно складними, що може призвести до демотивації або накопичення помилок.

Адаптивність досягається шляхом безперервного аналізу ключових характеристик користувача, серед яких: час виконання операцій, точність, число критичних помилок, повторюваність помилок, стабільність результатів, а також поведінкові патерни. На основі цих параметрів система може змінювати набір об'єктів у середовищі, їхню щільність, швидкість поведінки, параметри взаємодії або додавати нові непередбачувані сценарії.

У запропонованому підході модель адаптивної складності реалізується як автономний компонент, що отримує вхідні дані від підсистеми збору телеметрії, аналізує їх та формує рекомендації щодо модифікації сценарію. Для цього можуть застосовуватися різні математичні методи — від евристик до алгоритмів машинного навчання, зокрема reinforcement learning, який дозволяє системі самостійно визначати оптимальний рівень складності на основі очікуваної винагороди.

Модель передбачає наявність кількох рівнів складності — від базового до розширеного — які відрізняються кількістю активних об'єктів, швидкістю розвитку подій, ступенем випадковості та масштабом наслідків помилкових дій. Підвищення складності може супроводжуватися скороченням часу на виконання завдань,

збільшенням числа об'єктів, що взаємодіють із користувачем, або введенням додаткових умов, які потребують швидкої адаптації.

Завдяки такому підходу до побудови адаптивної складності навчальне середовище набуває властивостей персоналізованої системи, що враховує індивідуальні особливості кожного користувача. Це дозволяє суттєво підвищити ефективність навчання, формувати гнучкі навички реагування на змінні умови та сприяє розвитку критичного мислення.

Модель M_{AD} є математичною основою для керування адаптацією. Вона встановлює взаємозв'язок між станом користувача та необхідним станом середовища:

— компетентність користувача (C_{user}) — скалярна величина $C_{user} \in [0,100]$, що обчислюється на основі множини метрик $M(2.3.1)$ за певний часовий проміжок Δt;

— цільова компетентність (C_{target}) — заданий експертом ідеальний рівень успішності, який система має підтримувати (наприклад, $C_{target} = 80$);

— градієнт адаптації (G_{diff}) — служить керуючою величиною для зміни складності D.

$$G_{diff} = C_{user} - C_{target},$$

де $G_{diff} > 0$ — користувач занадто успішний, необхідне ускладнення D↑);

$G_{diff} < 0$ — користувач має труднощі, необхідне спрощення (D↓).

2.2.3 Алгоритм прийняття рішень для динамічної зміни сценарію

Алгоритм прийняття рішень для динамічної зміни сценарію визначає логіку адаптації навчального середовища відповідно до поведінки користувача, параметрів виконання завдань та контексту поточного етапу тренування (рис. 2.2). Його основна функція полягає у безперервному аналізі вхідних даних, формуванні

висновків щодо ефективності навчання та виборі оптимальної стратегії зміни умов сценарію, що сприяє підтриманню найбільш продуктивного рівня складності [27].

У загальному випадку алгоритм можна описати як послідовність кроків, яка включає:

— збір даних;
— аналіз продуктивності;
— формування рішення;
— застосування змін;
— оцінювання результатів.

Застосування такого циклу дозволяє системі працювати у реальному часі, реагувати на результати діяльності користувача та підтримувати оптимальний рівень виклику. Важливою частиною алгоритму є механізм зворотного зв’язку, що гарантує коректну адаптацію сценарію. Якщо зміна складності не приводить до бажаного результату (наприклад, користувач все ще припускається надмірної кількості помилок), алгоритм може повторно скоригувати сценарій або надати додаткові підказки.

Для реалізації алгоритму можуть застосовуватися як класичні методи, засновані на евристичних правилах, так і сучасні моделі машинного навчання, зокрема reinforcement learning або Bayesian decision models. Використання цих методів дозволяє формувати більш гнучку та інтелектуальну систему, здатну прогнозувати поведінку користувача та обирати оптимальні сценарії для подальшого навчання.

Алгоритм реалізує логіку AI Director та запускається періодично для забезпечення реактивності. На вхід подаються поточна складність, метрики M та налаштування (параметри α, ε). Спочатку система обчислює показники ефективності та необхідне відхилення. Далі перевіряється умова адаптації: якщо різниця перевищує поріг чутливості ε, виконується корекція.

Вона включає розрахунок нової складності з урахуванням коефіцієнта швидкості адаптації та перетворення цього скалярного значення у набір

конкретних ігрових параметрів (кількість об'єктів, швидкість NPC). На фінальному етапі викликається алгоритм розміщення об'єктів, і система видає оновлений сценарій.

Рисунок 2.2 — Алгоритм інтелектуального формування сценарію

2.3 Формалізація моделі адаптивної складності тренувального процесу

Формалізація моделі адаптивної складності тренувального процесу передбачає визначення набору параметрів, що описують рівень навчального впливу, а також механізмів, які контролюють динамічні зміни складності залежно

від результатів діяльності користувача. У такій моделі адаптивність розглядається не як фіксований набір режимів складності, а як безперервний процес, що реагує на миттєвий стан студента, забезпечуючи оптимальний баланс між мотивацією та навчальним викликом.

На формальному рівні модель адаптивної складності можна описати як функцію:

$$C(t) = f(U(t), H(t), S(t)) ,$$

де C(t) — рівень складності у момент часу t;

U(t) — параметри користувача (точність, час реакції, частота помилок);

H(t) — історичні дані (стабільність результатів, динаміка прогресу);

S(t) — стан навчального середовища (кількість об’єктів, активні події, темп сценарію).

Така функція дає змогу адаптувати складність у реальному часі, формується на основі вхідних даних з різних компонентів системи та визначає параметри навчального середовища на наступному кроці.

Ключовим елементом моделі є система вагових коефіцієнтів, що дозволяє підкреслити важливість окремих параметрів залежно від навчальних цілей. Наприклад, у сценаріях, де критичною є точність, відповідний коефіцієнт має домінувати над іншими. Це забезпечує можливість гнучкого налаштування алгоритму та підвищує ефективність навчального процесу.

Модель також передбачає поняття зони оптимальної складності, у межах якої користувач стикається з достатнім рівнем виклику, але зберігає можливість досягти успіху. Якщо результати діяльності значно перевищують цю зону, система поступово збільшує складність; якщо ж користувач демонструє стабільні труднощі, складність знижується. Таким чином, адаптація є двонапрямленою та безперервною.

Для визначення поточного рівня складності застосовуються агреговані метрики, що включають не лише прямі показники успішності (наприклад, відсоток

виконаних операцій), але й непрямі фактори: зростання часу реакції, повторювані помилки, нестабільні результати. У сукупності ці параметри формують вектор стану користувача, на основі якого обчислюється адаптивний рівень складності.

Формалізована модель забезпечує можливість інтеграції з механізмами інтелектуальної генерації, що дозволяє системі автоматично формувати навчальні ситуації з урахуванням поточного рівня підготовки студента. Такий підхід сприяє формуванню персоналізованих навчальних траєкторій та забезпечує високу ефективність тренувального процесу за рахунок постійної підтримки найбільш продуктивного режиму навчального навантаження.

2.3.1 Визначення множини метрик оцінки компетентності користувача

Оцінювання компетентності користувача в межах навчального тренажера є критично важливим завданням, яке забезпечує об’єктивне визначення рівня підготовленості, виявлення прогалин у знаннях та формування персоналізованої траєкторії навчання. Для цього необхідно сформувати множину метрик, що кількісно або якісно описують результативність виконання завдань, динаміку розвитку навичок та здатність користувача адаптуватися до змінних умов середовища.

У загальному випадку компетентність користувача можна розглядати як багатовимірну характеристику, що включає технічні, когнітивні та поведінкові аспекти діяльності. Тому множина метрик повинна враховувати не лише факт успішності виконання завдань, але й спосіб, яким користувач взаємодіє із середовищем, швидкість прийняття рішень, рівень стійкості до стресових ситуацій, а також стабільність демонстрованих результатів [28].

До базових системних метрик належать:

— час виконання операцій — відображає оперативність користувача при виконанні задач;

— точність дій — характеризує правильність прийнятих рішень відносно заданих критеріїв;

— кількість помилкових дій — показує, наскільки користувач схильний до помилок у складних або нестандартних умовах;

— повторюваність помилок — дозволяє виявити системні недоліки в підготовці;

— стабільність результатів — показує, наскільки користувач здатен утримувати досягнутий рівень успішності протягом тривалого часу.

Додатково виділяють метрики, що оцінюють здатність користувача до адаптації:

— швидкість прийняття рішень при зміні умов — визначає гнучкість навичок та оперативність реакції;

— ефективність корекції поведінки — показує здатність вчитися на помилках під час тренування;

— оцінка впевненості дій — характеризує стабільність вибраної стратегії поведінки.

У більш складних сценаріях застосовуються агреговані метрики, що включають інтегральні оцінки результативності, наприклад:

— загальний коефіцієнт успішності (на основі вагових коефіцієнтів важливості окремих показників);

— індекс прогресу, що відстежує динаміку результатів у часі;

— коефіцієнт адаптації, який визначає здатність користувача працювати в умовах підвищеної складності.

Застосування множини таких метрик забезпечує комплексне оцінювання компетентності, дозволяє формувати індивідуальні навчальні траєкторії та забезпечує зворотний зв’язок, необхідний для подальшого поліпшення якості навчального процесу. Крім того, результати оцінювання можуть використовуватися для автоматичної адаптації сценаріїв, що слугує фундаментом для реалізації інтелектуальної системи навчання.

Множина метрик є кількісною основою для оцінки. Їх збір здійснюється через Unreal Events/Delegates та Trace Systems.

$$M = \{M_t, M_a, M_e, M_p\},$$

де M_t — час, витрачений на виконання критичних підзавдань $M_t = t_{finish} - t_{start}$;

M_a — точність виконання навичок (наприклад, відсоток успішних влучань, або точність маніпуляцій);

M_e — кількість допущених критичних помилок (наприклад, зіткнення, порушення протоколу);

M_p — відношення оптимального шляху до фактичного пройденого шляху.

$$M_p = L_{optional}/L_{actual}.$$

Кожна метрика фіксується і зберігається в масиві для подальшої обробки та використання у ваговій моделі.

2.3.2 Математична модель оцінки поточної компетентності

Математична модель оцінки поточної компетентності користувача є фундаментальним елементом адаптивної тренажерної системи, оскільки саме вона забезпечує формальне подання рівня підготовленості на основі кількісних показників. Така модель дозволяє не лише фіксувати поточний стан користувача, а й прогнозувати його розвиток, що є ключовим для подальшого коригування навчальної траєкторії.

Загальний підхід до формалізації компетентності ґрунтується на обчисленні агрегованої оцінки, що враховує декілька груп показників. Нехай кожна метрика компетентності позначається як mim_imi, а її ваговий коефіцієнт — як wiw_iwi. Тоді сумарну оцінку поточної компетентності користувача можна описати наступним чином:

$$K(t) = \sum_{i=1}^{N} w_i * m_i(t),$$

де $K(t)$ — інтегральна оцінка компетентності в момент часу t

$m_i(t)$ — нормалізований показник iii-ї метрики;

w_i — ваговий коефіцієнт важливості метрики mi;

N — загальна кількість врахованих метрик.

Використання вагових коефіцієнтів дозволяє налаштовувати вплив окремих метрик залежно від специфіки навчального завдання або професійних вимог. Наприклад, у сценаріях, де критичним параметром є точність, коефіцієнт для відповідної метрики матиме найбільшу вагу.

Для формування точнішої оцінки також застосовуються функції згладжування та аналіз історичних даних. Зокрема, для врахування динаміки розвитку компетентності може використовуватися показник $\Delta K(t)$, що описує зміну інтегральної оцінки у часі:

$$\Delta K(t) = K(t) - K(t-1).$$

та нормалізована оцінка темпу прогресу, обчислена як:

$$P(t) = \frac{K(t)-K(t-1)}{K(t-1)+\varepsilon},$$

де ε — мале додатне число, що запобігає діленню на нуль.

Застосування таких формул дозволяє врахувати не лише поточний рівень компетентності, а й динаміку її зміни, що є важливим для побудови адаптивної складності.

В окремих випадках оцінка компетентності може бути представлена як вектор стану:

$$\vec{K}(t) = [k_1(t), k_2(t), \dots, k_M(t)],$$

Кожен елемент відображає певний аспект підготовленості (наприклад, технічні знання, точність виконання операцій, швидкість реагування, здатність до

адаптації тощо). Такий підхід дозволяє використовувати методи машинного навчання — класифікацію, кластеризацію, регресійний аналіз — для побудови прогнозів та рекомендацій щодо подальшого навчання.

Модель розраховує підсумковий скалярний показник компетентності, застосовуючи метод зваженого усереднення до попередньо нормалізованих метрик. Спочатку всі виміряні дані трансформуються у безрозмірні величини за допомогою спеціальних функцій (наприклад, лінійної або сигмоїдної), після чого вони об’єднуються з урахуванням визначених експертами вагових коефіцієнтів, які підкреслюють пріоритетність окремих навичок або критичність помилок. Для забезпечення стабільності результату та врахування довгострокової динаміки навчання у розрахунок закладено механізм згладжування («ефект забування»), що дозволяє уникнути різких стрибків оцінки та коректно відобразити накопичений прогрес.

2.3.3 Формалізація параметрів складності сценарію

Формалізація параметрів складності сценарію є необхідним етапом побудови адаптивної тренажерної системи, оскільки дозволяє кількісно описати характеристики навчального середовища, що визначають рівень виклику для користувача. Чітке визначення цих параметрів забезпечує можливість формування алгоритмів динамічної зміни сценарію, а також створення механізмів автоматичної оцінки ефективності навчального процесу.

Основою формалізації є виділення ключових параметрів, які безпосередньо впливають на складність виконання завдання. До таких параметрів належать:

— кількість активних об’єктів у сцені, що визначає щільність середовища та кількість одночасних взаємодій;

— швидкість реакції об’єктів, яка впливає на темп прийняття рішень користувачем;

— ступінь випадковості середовища, що визначає рівень непередбачуваності подій;

— розмір та конфігурація простору, які визначають складність просторової навігації;

— кількість одночасних цілей або задач, що задають когнітивне навантаження;

— тривалість виконання етапів, яка визначає стресове навантаження та вимоги до таймінгу;

— наслідки помилок, що регулюють ризики й визначають вагомість рішень користувача.

Складність сценарію визначається як узагальнена скалярна величина, що задає рівень навчального навантаження і трансформується системою у набір конкретних змін ігрового середовища. Цей показник керує чотирма основними групами параметрів: об'єктною складністю (збільшення щільності та кількості перешкод), поведінковою складністю (підвищення швидкості, точності та агресивності віртуальних агентів), часовою складністю (скорочення лімітів часу або пауз між подіями) та середовищною складністю (погіршення видимості чи акустичного фону). Кожен параметр має чітку функціональну залежність від загального рівня складності, що дозволяє автоматично переводити абстрактну оцінку навантаження у точні фізичні характеристики симуляції.

2.3.4 Модель зворотного зв'язку та адаптивного керування

Модель зворотного зв’язку та адаптивного керування є ключовим елементом тренажерної системи, що забезпечує динамічну зміну умов навчання відповідно до результатів діяльності користувача. Центральна ідея полягає в тому, що навчальне середовище не є статичним — воно реагує на дії студента, аналізує їх ефективність, виявляє помилки та визначає необхідний рівень коригування складності.

Основу цієї моделі складає безперервний процес збору та інтерпретації даних про взаємодію користувача із системою: точність виконання завдань, швидкість прийняття рішень, кількість успішних та помилкових дій, стабільність результатів у часі. Зібрана інформація обробляється аналітичними механізмами, які

визначають поточний стан користувача та порівнюють його з бажаними навчальними результатами [29].

Якщо виявлено відхилення від очікуваного рівня компетентності або темпу прогресу, система формує рішення щодо можливих змін у сценарії. Такими змінами можуть бути:

— підвищення або зниження складності завдань;

— зміна кількості об’єктів або умов;

— введення додаткових допоміжних підказок;

— модифікація темпу розвитку подій;

— зміна структури навчального етапу.

Зворотний зв’язок може бути прямим або непрямим. Прямий зворотний зв’язок включає явне інформування користувача щодо правильності або помилковості його дій, а також надання рекомендацій для подальшого виконання завдання. Такий підхід сприяє більш швидкому коригуванню поведінки та ефективному засвоєнню матеріалу.

Непрямий зворотний зв’язок передбачає адаптацію навчального середовища без явного повідомлення користувача — наприклад, плавну зміну умов, ускладнення або спрощення сценарію. Це дозволяє створити природний процес навчання, максимально наближений до реальних ситуацій, де необхідна здатність до самостійного аналізу та прийняття рішень.

Важливою складовою моделі є механізм поступового коригування складності. Система повинна змінювати умови не різко, а плавно, забезпечуючи комфортний перехід між рівнями складності. Це сприяє формуванню стійких навичок, знижує ризик стресу та втрати мотивації.

Застосування моделі зворотного зв’язку в інтелектуальній системі навчання створює умови для персоналізованого розвитку, оскільки кожен користувач отримує індивідуальний шлях проходження через навчальні ситуації. Це дозволяє максимально ефективно використовувати потенціал тренажера, підтримуючи

оптимальний рівень виклику та забезпечуючи досягнення заданих навчальних результатів.

Модель зворотного зв'язку реалізує алгоритм прийняття рішень, що забезпечує баланс між стабільністю системи та її чутливістю до дій користувача. Процес регулюється через визначення градієнта адаптації, порогової чутливості та коефіцієнта швидкості, який безпосередньо впливає на динаміку змін: високі значення дозволяють миттєво реагувати на дії, але підвищують ризик нестабільності (осциляцій), тоді як низькі значення гарантують плавність процесу ціною повільнішого відгуку. Логіка керування базується на порівнянні поточних результатів із цільовими межами: якщо рівень компетентності перевищує верхній поріг, система ініціює ускладнення сценарію, а при падінні нижче допустимого мінімуму — спрощує його, що забезпечує кількісно обґрунтоване керування тренувальним процесом.

2.4 Розробка алгоритмів взаємодіїміж модулем генерації та середовищем

Розробка алгоритмів взаємодії між модулем генерації та середовищем є ключовим аспектом побудови інтелектуальної тренажерної системи, оскільки саме ці алгоритми визначають механізми створення, адаптації та видалення об’єктів у віртуальному просторі відповідно до поточного навчального сценарію. Від коректності їх реалізації залежить здатність системи забезпечувати динамічну зміну умов тренування та швидко реагувати на дії користувача.

Модуль генерації виступає як централізований інструмент, що формує конфігурацію середовища на основі навчальних параметрів, історичних даних та поточного рівня компетентності. У свою чергу, середовище надає модулю інформацію про стан об’єктів, фізичні взаємодії та результати дій користувача. Таким чином, між цими компонентами формується двонапрямлений інформаційний обмін.

Алгоритми взаємодії реалізуються у вигляді послідовності процедур, що забезпечують:

— отримання модулем генерації інформації про поточний стан середовища;

— аналіз зібраних даних для визначення необхідності зміни сценарію;

— вибір оптимальної стратегії модифікації середовища;

— формування оновленого набору елементів сцени;

— передачу команд до середовища для здійснення змін;

— моніторинг ефективності внесених модифікацій.

У процесі взаємодії модуль генерації не лише ініціює створення нових об’єктів чи подій, але й коригує властивості вже існуючих елементів середовища: їхню поведінку, просторове розміщення, логіку реагування та взаємні залежності. Такий підхід забезпечує високу гнучкість системи та дозволяє підтримувати адаптивність навіть у складних сценаріях.

Важливим аспектом розробки алгоритмів є синхронізація між модулем генерації та механізмами фізичного моделювання середовища. Будь-які зміни в структурі сцени повинні враховувати фізичні властивості об’єктів, щоб уникнути некоректних взаємодій або порушення логічної послідовності подій. Це передбачає використання перевірок сумісності, що здійснюються перед застосуванням модифікацій.

Також важливе значення має обробка зовнішніх подій, що можуть вплинути на зміну сценарію: дії користувача, результати симуляцій, зміни параметрів середовища. Алгоритми повинні забезпечувати здатність до переривання поточного плану дій та оперативного формування нового сценарію у випадках, коли цього потребують навчальні цілі або поведінка користувача.

2.4.1 Протокол обміну даними між інтелектуальним модулем генерації та Unreal Engine

Протокол обміну даними між інтелектуальним модулем генерації та середовищем Unreal Engine визначає організацію інформаційних потоків, що забезпечують узгоджену взаємодію між підсистемами та гарантують коректність

оновлення стану тренувального середовища. Оскільки процес генерації контенту є динамічним і залежить від великої кількості параметрів, важливо забезпечити чіткі правила комунікації, структуру та послідовність обміну повідомленнями.

Основною задачею протоколу є забезпечення двонаправленого обміну інформацією:

— інтелектуальний модуль генерації отримує актуальні дані про стан середовища, параметри користувача та сценарні умови;

— Unreal Engine отримує команди від інтелектуального модуля генерації щодо зміни конфігурації сцени, створення чи видалення об’єктів, оновлення їхніх властивостей чи запуску подій.

Інформація, що передається від Unreal Engine до модуля генерації, включає:

— поточні параметри середовища (активні об’єкти, їхній стан, розміщення, взаємодії);

— результати дій користувача (правильні та помилкові операції, час виконання, успішність);

— події сценарію (тригери, завершення етапів, аварійні ситуації);

— аналітичні показники, зібрані підсистемою оцінювання.

У відповідь модуль генерація надсилає інструкції, що визначають динаміку сценарію:

— створення нових об’єктів та їх розміщення у просторі;

— зміну властивостей існуючих елементів (поведінкових, фізичних, логічних);

— запуск або завершення сценарних подій;

— адаптацію складності відповідно до оцінки компетентності користувача.

Для забезпечення стабільності системи обмін реалізується у вигляді структурованих запитів та відповідей. Кожен запит містить:

— унікальний ідентифікатор;

— тип операції (створення, зміна, видалення, запит інформації);

— параметри операції;

— часову мітку.

Така структура дозволяє відслідковувати послідовність подій, уникати конфліктів та забезпечувати коректність оновлень у реальному часі.

Синхронізація між інтелектуальним модулем генерації та Unreal Engine здійснюється циклічно, із фіксованою частотою або за подієвим механізмом. Подієвий підхід застосовується в тих випадках, коли зміни сценарію відбуваються у відповідь на ключові події (завершення завдання, аварійна ситуація, перехід до нового етапу). Циклічний механізм використовується для постійного відстеження стану середовища, що є необхідним у складних сценаріях із високою динамікою.

Важливою властивістю протоколу є підтримка асинхронної взаємодії, що дозволяє ІГС проводити аналіз даних та приймати рішення без блокування основного циклу симуляції в Unreal Engine. Це забезпечує плавність роботи тренажера та мінімізує затримки, що критично для сценаріїв реального часу.

Технічною основою для реалізації адаптивних сценаріїв обрано ігровий рушій Unreal Engine 5, який виступає головним ядром візуалізації та фізичної симуляції. Застосування передових технологій рендерингу, таких як Lumen та Nanite, дозволяє досягти високого рівня графічної деталізації, необхідного для створення імерсивного середовища, тоді як система Chaos Physics забезпечує реалістичну обробку колізій та динаміку твердих тіл. Для автоматизованої побудови простору використовується інструментарій Procedural Content Generation (PCG) Framework. Ця підсистема, що базується на графовій логіці, дозволяє створювати параметричні карти різної типології — від природних ландшафтів до урбаністичних руїн — динамічно адаптуючи їх конфігурацію залежно від вхідного зерна генерації (Seed) та заданих параметрів складності.

Керування поведінкою віртуальних агентів (NPC) покладено на системи Behavior Tree та Blackboard, які забезпечують ієрархічну та модульну структуру штучного інтелекту. Такий підхід дозволяє агентам гнучко реагувати на зміни ситуації, ініційовані Менеджером Сценаріїв, наприклад, миттєво переходити від стану патрулювання до тактичного наступу. Взаємодія користувача з симуляцією

організована через підсистему Enhanced Input, яка відповідає за гнучку обробку вхідних даних. Вона підтримує контекстне перепризначення дій (Input Mapping Contexts), що дозволяє автоматично змінювати схему керування залежно від ігрового контексту (наприклад, наявність зброї в руках) та уніфікувати інтерфейс для різних типів контролерів.

2.4.2 Алгоритм інтелектуального розміщення об'єктів сценарію

Алгоритм інтелектуального розміщення об’єктів сценарію є важливою складовою інтелектуальної системи генерації, оскільки він визначає просторову конфігурацію елементів навчального середовища, впливаючи на складність, логіку проходження та характер взаємодії користувача з тренажером. Основна мета алгоритму полягає у тому, щоб розташувати об’єкти таким чином, аби вони сприяли формуванню цільових компетенцій, забезпечували оптимальне навчальне навантаження та відповідали динамічним умовам сценарію.

На відміну від статичних або випадкових способів розміщення, інтелектуальний підхід передбачає аналіз параметрів середовища, поточного рівня підготовки користувача та взаємозв’язків між об’єктами. Це дає змогу формувати логічно узгоджені конфігурації, які враховують як навчальні цілі, так і поведінкові особливості студента.

Алгоритм роботи модуля інтелектуальної генерації навчального середовища реалізується послідовно та охоплює такі основні етапи:

— збір вихідних даних щодо сценарію;
— аналіз середовища;
— генерація конфігурацій;
— вибір оптимального варіанту;
— застосування рішення;
— оцінка результатів.

Ключовим аспектом є врахування динаміки взаємодії об’єктів між собою. Алгоритм повинен гарантувати коректність просторової структури: об’єкти не

мають блокувати один одного, порушувати логіку проходження або створювати надмірно складні умови, якщо цього не передбачено сценарієм. Важливу роль відіграє також забезпечення доступності об’єктів для користувача, що означає урахування траєкторій руху, зони видимості та фізичних перешкод.

Додаткові критерії відбору конфігурації можуть включати:

— варіативність просторових розташувань для запобігання повторюваності;

— підтримку сценарної логіки;

— урахування факторів часу;

— потенціал для адаптації складності.

Алгоритм інтелектуального розміщення може комбінувати евристичні методи з машинним навчанням, що дозволяє поступово вдосконалювати конфігурації відповідно до накопиченого досвіду. Наприклад, система може навчатися на основі історичних даних, визначаючи оптимальні місця розміщення, що сприяють досягненню навчальних цілей. Це створює умови для побудови більш гнучких та персоналізованих сценаріїв.

Застосування такого алгоритму дозволяє підвищити якість тренувального процесу, зробити середовище більш реалістичним і варіативним, а також сприяє розвитку у користувача навичок швидкої адаптації до нових умов.

Алгоритм відповідає за динамічне наповнення простору новими об’єктами, такими як перешкоди чи цілі, узгоджуючи їх розташування з поточними вимогами складності та геометричними обмеженнями сцени. Процес розпочинається з визначення цільової зони та пошуку валідних навігаційних точок, які обов’язково проходять перевірку на відсутність конфліктів. Ключовим етапом є вибір позиції на основі системи ваг: зі зростанням рівня складності пріоритет надається точкам, що розташовані ближче до гравця або створюють значніші перешкоди, а також тим локаціям, які стимулюють відпрацювання навичок, визначених системою як слабкі. Завершується цикл безпосередньою генерацією об’єкта у вибраній оптимальній точці.

2.4.3 Механізм динамічної корекції логіки сценарію

Механізм динамічної корекції логіки сценарію забезпечує оперативне внесення змін до структури навчального процесу відповідно до дій користувача, результатів його взаємодії з тренажером та поточного стану середовища. На відміну від статичних сценаріїв, де послідовність подій визначена наперед, динамічний механізм дає змогу формувати індивідуальний маршрут навчання, що адаптується у режимі реального часу.

Основою такого підходу є безперервний моніторинг виконання завдань і збір телеметричних даних: точність, швидкість, частота помилок, типові патерни поведінки, ефективність виконання попередніх кроків. На основі цих даних система визначає, чи потребує сценарій модифікації, і якщо так — у якому напрямку.

Корекція може включати:

— зміну складності;

— додавання або вилучення об’єктів чи умов;

— введення підказок або консультаційних повідомлень;

— зміну послідовності кроків сценарію;

— активацію альтернативних гілок розвитку подій;

— збільшення або зменшення часу на виконання операцій.

У процесі корекції важливо зберігати узгодженість логіки: зміни не повинні порушувати цільову структуру навчального процесу та призводити до появи суперечливих ситуацій. Для цього перед внесенням модифікацій здійснюється попередня перевірка сумісності із загальною логікою сценарію.

Механізм динамічної корекції передбачає функціонування у двох режимах: автоматичному, де рішення про модифікацію приймаються системою самостійно на основі визначених алгоритмічних моделей, та напівавтоматичному, в якому система генерує рекомендації, а остаточне рішення ухвалює інструктор. Використання автоматичного режиму забезпечує можливість реалізації повністю автономного навчального процесу, тоді як напівавтоматичний підхід дозволяє

зберегти контроль викладача, що є критично важливим для гнучкого коригування ходу тренування у складних або нестандартних сценаріях.

Важливою складовою механізму є підтримка контексту. Система повинна враховувати не лише поточний результат, але й те, які дії передували, який рівень складності був застосований раніше, які помилки повторювалися та які об’єкти вже були активними. Такий контекстно-залежний підхід дозволяє уникати ситуацій, коли сценарій змінюється без урахування логічної послідовності.

3 ПРОГРАМНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ НАВЧАЛЬНОГО СЕРЕДОВИЩА

3.1 Структура програмного комплексу та опис ключових класів

Структура програмного комплексу побудована за модульним принципом і відображає логічний поділ функціональності між окремими компонентами системи. Такий підхід спрощує підтримку, тестування та розширення функціоналу, оскільки кожний модуль відповідає за конкретний набір задач і має чітко визначений інтерфейс взаємодії з іншими частинами тренажера [30].

Архітектура програмного комплексу складається з кількох основних рівнів:

— рівень симуляції середовища, який реалізується засобами Unreal Engine і містить об’єкти сцени, моделі поведінки та механізми фізичної взаємодії;

— рівень інтелектуальної генерації, відповідальний за створення та адаптацію сценаріїв відповідно до дій користувача;

— рівень аналізу та оцінювання, який збирає телеметрію, інтерпретує результати виконання завдань і формує зворотний зв’язок;

— рівень взаємодії з користувачем, що включає інтерфейс, систему підказок, повідомлень та інструментів навігації;

— рівень даних, що містить модулі збереження інформації, профілів користувачів і налаштувань сценаріїв.

Програмна реалізація системи базується на наборі класів, що описують ключові сутності та їхню поведінку. Основу віртуального середовища формує базовий клас EnvironmentManager, який відповідає за ініціалізацію сцени, запуск симуляції та синхронізацію станів. Його доповнюють ScenarioObject — абстракція, що містить параметри положення та стану об’єктів, а також InteractiveObject, який безпосередньо реалізує методи обробки подій при взаємодії користувача з елементами сцени.

За створення та адаптацію наповнення відповідає група класів генерації контенту. Центральним тут є GenerationController, який координує життєвий цикл об’єктів, інтерпретуючи дані від аналітичного модуля. Йому допомагають

PlacementSolver, що обчислює оптимальне просторове розташування елементів з урахуванням обмежень, та ScenarioModifier, який забезпечує динамічну зміну структури сценарію безпосередньо під час тренувального процесу.

Аналітичний блок системи представлений класами оцінювання: MetricsCollector збирає телеметричні дані (час виконання, кількість помилок), CompetenceEvaluator на їх основі визначає поточний рівень компетентності та прогнозує прогрес, а ProgressTracker фіксує динаміку успішності для формування рекомендацій щодо наступних етапів.

Завершують архітектуру класи взаємодії та даних. UIController керує відображенням інтерфейсу, працюючи в тандемі з HintSystem, яка генерує контекстні підказки. Збереження та обробку інформації забезпечують UserProfile (історія навчання та профіль користувача), ScenarioConfig (параметри та обмеження сценарію) і LogStorage, що відповідає за надійний запис телеметрії та логів на диск.

Взаємодія між класами здійснюється через спеціально визначені API, що описують методи отримання та передачі даних, ініціювання подій та зміни станів. Така модульна структурування дозволяє легко розширювати систему, інтегрувати нові алгоритми та замінювати окремі компоненти без необхідності суттєво переробляти архітектуру.

Описана структура програмного комплексу забезпечує гнучкість і масштабованість рішень, необхідних для створення повноцінного тренажера, що адаптується до індивідуального рівня користувача та підтримує широкий спектр навчальних сценаріїв.

3.1.1 Ієрархія класів програмного ядра

Ієрархія класів програмного ядра відображає логічну структуру системи та визначає взаємозв’язки між основними компонентами, що забезпечують функціонування інтелектуального навчального середовища. Правильна побудова ієрархічних залежностей дозволяє стандартизувати процес розробки, підвищити

рівень повторного використання коду, спростити інтеграцію нових компонентів та забезпечити цілісність архітектури програмного комплексу.

У фундаменті ієрархії знаходиться базовий клас CoreObject, який задає мінімальний набір атрибутів та інтерфейс для успадкування іншими сутностями. Він визначає ключові властивості об’єктів — унікальний ідентифікатор, стан, параметри ініціалізації та базові методи взаємодії [31].

Від базового компонента архітектури успадковуються дві основні категорії класів: системні, що визначають загальну логіку керування тренажером, та прикладні, які описують об’єкти сценарію. До групи системних належить EngineController, відповідальний за управління життєвим циклом системи та синхронізацію модулів, а також ScenarioManager, який контролює розвиток сюжету та координує події. Їхню роботу доповнюють GenerationModule, що реалізує механізми створення й адаптації контенту, та MetricsManager, який забезпечує збір і агрегування телеметричних даних для оцінки навчання.

Прикладний рівень формують класи, що представляють сутності віртуального середовища. Фундаментом тут виступає BaseActor — узагальнений клас із даними про розташування та стан об’єктів. Його функціональність розширює InteractiveActor, додаючи можливості взаємодії з користувачем, такі як обробка команд чи фізичних маніпуляцій. Окремо виділяється клас ScenarioItem, що описує конкретні елементи навчального завдання: інструменти, обладнання або контрольні точки.

Важливу роль відіграє підгрупа аналітичних класів, спрямованих на обробку даних. UserModel підтримує профіль учасника, зберігаючи історію його прогресу та рівень компетентності. Клас Evaluator аналізує поточну успішність та формує рекомендації для системи адаптації, тоді як HistoryTracker фіксує точну послідовність дій, що дозволяє виявляти та досліджувати поведінкові патерни користувача під час виконання завдань.

Взаємодія між класами здійснюється за принципом інверсії залежностей, що дозволяє відокремити високорівневу логіку від деталізації конкретних реалізацій.

Таким чином, програмне ядро залишається гнучким до змін і може легко доповнюватися новими компонентами без порушення існуючої структури.

Побудована ієрархія забезпечує:

— логічну розмежованість відповідальності між компонентами;

— масштабованість системи;

— можливість використання шаблонів проєктування;

— простоту тестування та інтеграції.

Завдяки чітко визначеній структурі ієрархія класів формує стабільний фундамент програмного ядра та забезпечує послідовність роботи усіх компонентів навчального тренажера.

Ядро програмного комплексу реалізовано на C++ для забезпечення необхідної продуктивності та низькорівневого доступу до API рушія. Ієрархія класів будується на основі фундаментальних типів Unreal Engine, таких як AActor, UObject та UGameInstance (глобальний стан гри).

Архітектурним ядром системи виступає клас UAdaptiveScenarioManager (успадкований від UObject), який виконує роль «мозку» ІГС. Він інкапсулює алгоритми прийняття рішень та модель адаптивного керування, відповідаючи за логіку розрахунку компетентності користувача та динамічну корекцію рівня складності (D). Основний функціонал реалізовано через методи CalculateCompetency (обробка метрик), UpdateScenarioDifficulty (періодичний перерахунок складності) та SpawnScenarioObject, який викликає логіку інтелектуального розміщення об'єктів відповідно до поточних умов.

За збір даних відповідає компонент UTrainingDataComponent, що базується на UActorComponent і прикріплюється безпосередньо до актора гравця. Цей клас реалізує функціонал модуля моніторингу та оцінювання, збираючи «сирі» метрики в режимі реального часу. Для накопичення даних про події використовується структура FTrainingMetricData, а при виникненні критичних ситуацій спрацьовує Unreal Delegate OnCriticalError.Broadcast, що забезпечує миттєву передачу інформації до менеджера сценаріїв.

Інтерактивні об'єкти середовища будуються на основі базового класу AScenarioObjectBase (спадкоємець AActor). Він є фундаментом для всіх елементів, що генеруються або модифікуються системою, і містить параметр BaseDifficultyCost, який визначає «вартість» об'єкта для загального балансу складності. Важливою складовою є метод ApplyDifficultyModifier, що дозволяє з рівня C++ ініціювати специфічну логіку адаптації, реалізовану в Blueprints, забезпечуючи гнучкий зв'язок між кодом і візуальним скриптингом.

Для реалізації поведінки персонажів використовується клас AControllableNPC (базовий клас ACharacter), який також є спадкоємцем AScenarioObjectBase. Він інтегрує логіку керування через Behavior Tree та містить змінні, залежні від параметрів складності, такі як MovementSpeedModifier та AimingAccuracy. Ці параметри динамічно змінюються Менеджером Сценаріїв, що разом із описаною ієрархією забезпечує модульність системи та чітке розділення відповідальності між обчислювальним ядром та сутностями ігрового світу.

3.1.2 Розподіл функціоналу між C++ та Blueprints

Розподіл функціоналу між C++ та Blueprints у межах Unreal Engine є важливим аспектом архітектури програмного комплексу, оскільки він визначає баланс між продуктивністю, гнучкістю розробки та зручністю подальшої підтримки тренажерної системи. Обидва механізми доповнюють один одного, утворюючи єдине середовище, у якому низькорівнева оптимізована логіка поєднується з високорівневим інструментарієм швидкої побудови поведінки [32].

У загальному вигляді розмежування відповідальності можна сформулювати таким чином:

— C++ використовується для реалізації критично важливих алгоритмів, що потребують високої продуктивності, а також для створення базових класів, модулів обробки даних та складної логіки;

— Blueprints застосовуються для опису поведінки об’єктів на рівні сценаріїв, побудови послідовностей подій, інтеграції інтерфейсів та швидкої адаптації логіки без перекомпіляції ядра.

У межах навчального тренажера функціонал, написаний на C++, зазвичай включає:

— реалізацію внутрішньої логіки ядра симуляції;

— алгоритми аналізу дій користувача та обробки телеметрії;

— системи взаємодії із фізичними моделями та підсистемами руху;

— низькорівневі компоненти модулів генерації контенту;

— реалізацію кастомних класів Actor, Component та Controller з підвищеною продуктивністю.

Це дозволяє отримати максимальну швидкодію, гнучкість та контроль над внутрішніми механізмами рушія.

Blueprints, у свою чергу, використовуються там, де потрібна оперативність змін та зручність візуального редагування логіки. Зокрема:

— визначення логіки поведінки окремих об'єктів у сценарії;

— створення дрібних адаптивних компонентів;

— побудова UI та зв’язків між інтерфейсними елементами;

— організація візуальних ефектів і систем зворотного зв’язку;

— швидке прототипування нових механік та сценарних подій.

Такий підхід дозволяє інструкторам або іншим членам команди, які працюють без глибоких знань C++, оперативно вносити зміни в сценарій, розширювати функціонал або модифікувати логіку навчання без зміни базового коду.

Важливим аспектом є створення «гібридної» структури, у якій C++ забезпечує фундаментальну інфраструктуру, а Blueprints — гнучкість та адаптивність. Більшість ключових класів визначається на рівні C++ і містить події та функції, які можуть бути розширені або прив’язані до Blueprint-компонентів. Це забезпечує чисту об’єктну модель, зберігаючи можливість швидкої інтеграції сценарної логіки.

Таке розмежування дає низку переваг:

— підвищення продуктивності системи;

— скорочення часу на розробку та тестування;

— полегшення модифікації високорівневої логіки;

— зниження навантаження на базові компоненти;

— можливість гнучкого налаштування під потреби різних сценаріїв.

Для досягнення оптимального балансу між продуктивністю, гнучкістю розробки та легкістю підтримки, програмний функціонал було стратегічно розділено між C++ та візуальним сценарним інструментом Blueprints.

Функціонал, реалізований на C++, становить ядро інтелектуальної системи. Вибір C++ тут є критично важливим, оскільки ці задачі є обчислювально інтенсивними та вимагають максимальної швидкодії.

Функціональні можливості C++ представлені у таблиці 3.1.

Таблиця 3.1 — Функціональність C++

Алгоритми Адаптації та Керування	Реалізація Моделі Оцінки Компетентності (C_{user}) та Алгоритму Прийняття Рішень (AI Director). Необхідна робота в режимі реального часу (low-latency) для динамічної корекції D.
Система Збору та Нормалізації Метрик	Низькорівнева фіксація подій (Delegates, Events) та первинна математична обробка сирих даних M для забезпечення точності та мінімізації навантаження.
Інтелектуальне Розміщення	Складні просторові обчислення (NavMesh Queries, Ray Tracing, Overlap Checks) для реалізації алгоритму, які мають бути виконані швидко для динамічного спауну.
Базові Класи та Структури Даних	Визначення ключової ієрархії класів та структур даних (наприклад, FTrainingMetricData) для забезпечення стабільності та модульності.

Blueprints використовуються для реалізації функцій, які вимагають швидкого прототипування, легкої зміни візуальних ефектів або взаємодії з користувацьким інтерфейсом (UMG).

Функціональні можливості C++ представлені у таблиці 3.2.

C++ класи, такі як UAdaptiveScenarioManager, виступають у ролі Майстра, викликаючи Blueprint-івські методи на акторах. Blueprints, у свою чергу, передають дані C++ компонентам, які є Слухачами. Такий розподіл гарантує, що C++

забезпечує швидкість і складність алгоритмів, тоді як Blueprints відповідають за візуальне відображення та гнучкість контенту.

Таблиця 3.2 — Функціональність Blueprints

Візуалізація та Ефекти	Реалізація візуальних змін, викликаних адаптацією (наприклад, зміна освітлення, поява туману, активація particle-ефектів).
Користувацький Інтерфейс (UMG)	Створення інтерфейсу тренувального процесу (HUD) та інтерфейсу моніторингу для експериментатора, що є більш ефективним у візуальному інструменті.
Логіка Акторів (Виконавчі Сценарії)	Реалізація специфічної логіки AScenarioObjectBase через BlueprintImplementableEvent (наприклад, анімація відчинення дверей, звукові підказки), які не є критичними для продуктивності.

Оптимальний розподіл функціоналу між C++ та Blueprints забезпечує збалансовану архітектуру тренажерної системи, у якій поєднуються ефективність виконання складних обчислень та зручність у розширенні й адаптації поведінкової логіки.

3.2 Реалізація модуля інтелектуальної генерації середовища

Реалізація модуля інтелектуальної генерації середовища є центральним елементом програмного комплексу, оскільки саме цей модуль забезпечує створення, адаптацію та динамічну зміну конфігурації об’єктів у тренувальному просторі. Його функціонування визначає рівень варіативності сценаріїв, здатність системи до персоналізації та загальну ефективність навчального процесу.

Архітектура модуля складається з кількох взаємопов’язаних компонентів, що виконують окремі аспекти генерації: аналіз середовища, формування початкової конфігурації, адаптивну модифікацію та оцінювання результативності внесених змін. Така багаторівнева структура дозволяє гнучко реагувати на дії користувача та формувати навчальні ситуації відповідно до поточного рівня складності та компетентності.

Нижче наведено детальний опис кожного компонента та їх взаємозв'язків які представлені на рисунку 3.1.

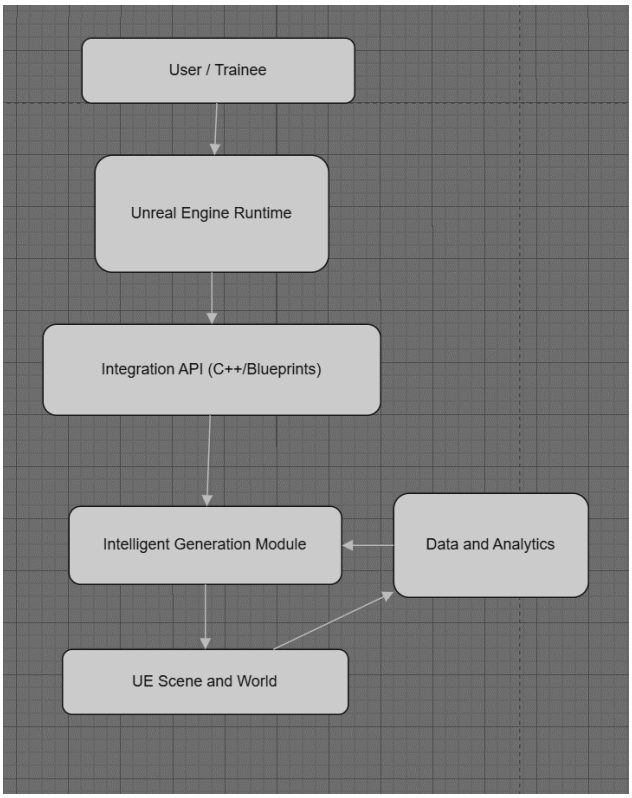

Рисунок 3.1 — Схема архітектури модуля

Користувач (User) виступає основним джерелом взаємодії із системою та активним учасником тренувального процесу. Він виконує дії всередині віртуального середовища, генеруючи вхідні дані для аналізу (поведінка, помилки, час реакції) та отримуючи зворотний зв'язок у вигляді результатів симуляції й адаптивних змін оточення. Функціонування цього середовища забезпечує Unreal Engine Runtime — базовий рушій, що відповідає за виконання ігрового циклу, фізичну симуляцію, рендеринг, анімацію та обробку подій. Він керує логікою Actor-компонентів і передає необхідні дані до вищих рівнів системи.

Зв'язок між рушієм та модулем інтелектуальної генерації забезпечує Integration API. Цей проміжний рівень, реалізований через C++ класи та Blueprint об’єкти, контролює асинхронну передачу даних, отримує інформацію зі сцени та викликає інтерфейсні функції для оновлення параметрів. Сам Intelligent Generation Module є «мозком» системи: він аналізує компетентність користувача, приймає рішення щодо зміни складності й темпу виконання завдань, а також динамічно

модифікує контент, передаючи відповідні команди назад у середовище для оновлення структури навчального простору.

Підсистема Data and Analytics відповідає за збір телеметрії (швидкість, точність, стабільність дій) та формування аналітичної бази для адаптації. Вона зберігає історію прогресу, розраховує ключові метрики та надає результати модулю IGM для прийняття зважених рішень. Всі зміни відображаються у компоненті UE Scene and World — віртуальному середовищі, де відбувається рендеринг 3D-об’єктів та їхня фізична взаємодія. Саме тут реалізується реалістичний контекст навчання, параметри якого постійно оновлюються відповідно до команд керуючого модуля.

На етапі ініціалізації модуль отримує вхідні дані, що включають цілі сесії, вимоги до конфігурації, початкові обмеження та попередні результати користувача. Ці параметри стають фундаментом для побудови первинної версії сцени (рис. 3.2). Система виконує логічний аналіз, визначаючи набір обов’язкових об’єктів, їхню кількість, зони розміщення та властивості, необхідні для коректного старту симуляції.

Рисунок 3.2 — Приклад згенерованої локації для навчального сценарію.

Далі система переходить до етапу корекції та адаптації. Збір телеметричних даних про дії користувача дозволяє модулю оцінити, наскільки обрана конфігурація відповідає навчальним цілям. Якщо користувач успішно виконує

завдання на даному рівні складності, модуль поступово ускладнює сценарій, додаючи нові об’єкти або змінюючи параметри їхньої поведінки. У випадку виникнення труднощів (рис 3.3), модуль навпаки, спрощує умови, забезпечуючи плавний перехід і збереження мотивації до подальшого навчання.

У процесі адаптації застосовується описаний раніше підхід до формалізації сценарію через набір обмежень. Це дозволяє уникати некоректних конфігурацій, забезпечувати логічну послідовність подій та гарантувати дотримання навчальних вимог. Крім того, окремі елементи сценарію можуть змінюватись у режимі реального часу — без перезавантаження сцени — що дозволяє модулю оперативно реагувати на стани користувача.

Модуль інтелектуальної генерації функціонує у тісній взаємодії з іншими підсистемами:

— ядро симуляції;

— аналітична підсистема;

— модуль сценарного контролю.

Рисунок 3.3 — Згенерована локація для складного сценарію

Обмін даними між цими компонентами здійснюється за допомогою уніфікованих інтерфейсів і протоколів, що забезпечують передбачуваність та узгодженість поведінки системи в цілому [33, 34].

Завдяки такій організації модуль генерації середовища може працювати у різних режимах:

— автоматичний — повністю автономно формує сценарій;

— керований — формує рекомендації для інструктора;

— гібридний — комбінує автоматичні рішення з ручним втручанням.

Це надає системі гнучкість та адаптивність, дозволяючи використовувати її як для самостійної підготовки користувача, так і у навчальному процесі з інструктором.

Модуль Інтелектуальної Генерації Сценаріїв реалізований переважно як C++ клас UAdaptiveScenarioManager, який є центральним контролером, що поєднує логіку адаптації з функціоналом генерації. Його ключові функції — інтелектуальне розміщення об'єктів та динамічна адаптація сценарію.

3.2.1 Реалізація інтелектуального розміщення об'єктів

Реалізація інтелектуального розміщення об’єктів у тренувальному середовищі спрямована на формування логічно узгоджених, адаптивних та педагогічно обґрунтованих конфігурацій, що підтримують навчальні цілі та забезпечують оптимальне навантаження для користувача. На відміну від статичних підходів, де розміщення визначається наперед, інтелектуальний модуль здатний динамічно коригувати просторові параметри сценарію відповідно до поточного рівня компетентності, історії дій та контексту ситуації.

Нижче наведено детальний опис кожного компонента та його параметрів, представлених на структурній схемі модуля інтелектуальної генерації представленої на рисунку 3.4.

Intelligent Generation Module виступає центральним програмним компонентом системи, відповідальним за адаптивну побудову тренування, аналіз компетентності користувача та динамічну зміну сценарію. Він керує основною логікою навчального процесу, координуючи роботу всіх підсистем — від збору даних до безпосередньої адаптації — та забезпечує інтеграцію з ігровим рушієм Unreal Engine. Основна мета модуля полягає в постійному оновленні профілю

користувача та формуванні персоналізованих сценаріїв на основі отриманої інформації.

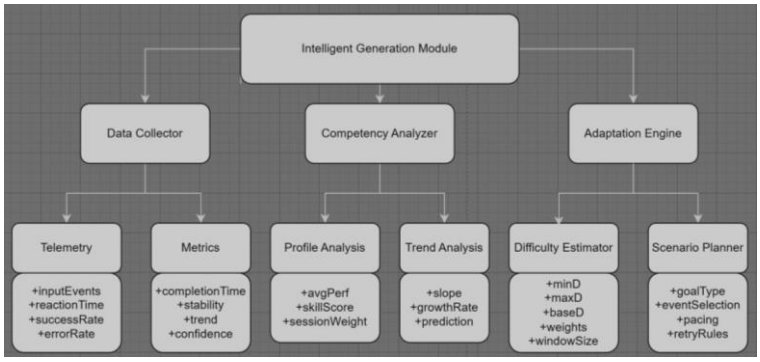

Рисунок 3.4 — Схема структури модуля

Процес отримання інформації реалізується підсистемою Data Collector, яка збирає «сирі» дані про взаємодію користувача із середовищем та обчислює необхідні метрики. На низькому рівні працює компонент Telemetry, що фіксує деталі кожної сесії: події вводу (inputEvents), середній час реакції, а також відсоткові показники успішних дій та помилок. На основі цієї телеметрії компонент Metrics формує агреговані показники, такі як час виконання (completionTime), стабільність навичок, загальний тренд успішності та рівень достовірності даних, передаючи їх для подальшої обробки.

Аналітичний етап забезпечує Competency Analyzer — підсистема, що оцінює рівень підготовки користувача, аналізуючи його загальний профіль, динаміку прогресу та виявляючи слабкі сторони. На основі цих висновків вступає в дію Adaptation Engine, який регулює складність і структуру тренування. Цей рушій у реальному часі розраховує навантаження, підбирає оптимальні події та формує навчальні завдання, визначаючи логіку переходів і створюючи оптимальну траєкторію розвитку навичок.

На практичному рівні розміщення виконується шляхом аналізу набору початкових параметрів, що включають структуру завдання, доступну площу сцени, кількість активних об’єктів та взаємні обмеження між ними. Модуль розглядає середовище як сукупність зон, у яких можуть бути розташовані ті чи

інші елементи, і визначає оптимальні конфігурації, що відповідають сценарним вимогам.

Ключовими етапами реалізації є:

— визначення контексту середовища;

— генерація варіантів конфігурацій;

— вибір оптимальної конфігурації;

— розміщення об’єктів;

— моніторинг ефективності.

Особливу увагу приділено врахуванню взаємозалежностей між об’єктами. Наприклад, якщо два об’єкти мають використовуватися послідовно, вони повинні бути розташовані на відстані, що мінімізує зайві переміщення користувача та сприяє логічному виконанню завдання. У складніших сценаріях може враховуватися також прихованість, доступність або пріоритетність об’єктів.

Модуль інтелектуального розміщення підтримує можливість динамічної адаптації. Якщо система виявляє, що користувач надто легко або надто повільно виконує завдання, конфігурація може бути змінена в реальному часі:

— додаються нові об’єкти;

— змінюється їхня просторова орієнтація;

— модифікується відстань між елементами;

— вводяться додаткові обмеження.

Така адаптація дозволяє забезпечити поступове ускладнення сценарію та підтримувати користувача в зоні оптимального розвитку.

Важливо, що логіка розміщення інтегрована з механізмами оцінювання прогресу. Інформація про швидкість виконання операцій, частоту помилок та повторюваність дій передається модулю, що дає змогу коригувати майбутні розміщення об’єктів. Це створює замкнений цикл навчання, у якому кожне рішення формується на основі попереднього досвіду користувача.

Завдяки такому підходу система забезпечує високу варіативність тренувальних сценаріїв, уникає повторюваності та сприяє формуванню гнучких

професійних навичок у користувача.

Алгоритм інтелектуального розміщення гарантує відповідність згенерованих об'єктів поточному рівню складності D та логічній структурі сценарію, забезпечуючи когерентність контенту. Реалізація цього процесу базується на методі UAdaptiveScenarioManager::SpawnScenarioObject, який використовує засоби UNavigationSystemV1 для пошуку досяжних навігаційних точок із подальшою просторовою валідацією координат через вбудовані механізми Unreal Engine. Вибір конкретної позиції здійснюється за допомогою зваженого алгоритму, що враховує пріоритетність локацій, після чого відбувається безпосередній спаун актора та ініціалізація його внутрішньої логіки переданими параметрами складності.

3.2.2 Реалізація механізму динамічної адаптації

Реалізація механізму динамічної адаптації забезпечує можливість оперативної зміни сценарію навчання відповідно до поточного стану користувача та умов середовища. Метою цього механізму є підтримання балансу між складністю завдань і рівнем підготовленості студента, що дає змогу утримувати його у зоні оптимального розвитку, не допускаючи перевантаження або втрати мотивації.

У процесі адаптації модуль працює за циклічним підходом, у якому кожен етап включає:

— аналіз дій користувача та поточного сценарію;
— визначення необхідності внесення змін;
— формування рішень щодо корекції;
— застосування оновлень;
— оцінювання ефективності змін.

На першому етапі система збирає телеметричні дані: успішність виконання завдань, швидкість реакції, кількість помилок, повторюваність неправильних дій, стабільність результатів. Дані аналізуються у контексті навчальних цілей, що дає змогу виявити слабкі сторони користувача або, навпаки, ділянки, де він демонструє

впевнене володіння матеріалом.

На основі аналізу формується рішення щодо зміни сценарію. Корекції можуть бути різного характеру:

— зміна кількості об'єктів або їхніх властивостей;

— зміна розташування об’єктів;

— модифікація темпу подій;

— введення додаткових обмежень;

— зміна послідовності виконання завдань;

— введення допоміжних підказок.

Важливим аспектом реалізації адаптації є забезпечення безперервності процесу. Зміни повинні відбуватися таким чином, щоб не порушувати логічну структуру навчального сценарію та не створювати для користувача відчуття випадковості або хаотичності. Для цього перевіряється відповідність нових умов попереднім етапам навчання та навчальним цілям.

Після застосування оновлень система продовжує відстежувати хід виконання завдань і оцінювати ефективність корекцій. Якщо адаптація не приводить до бажаного результату (наприклад, користувач продовжує систематично допускати помилки), алгоритм може повторно змінювати сценарій, знижуючи складність або вводячи додаткові рекомендації. Таким чином, реалізується замкнений цикл зворотного зв'язку.

Механізм динамічної адаптації тісно інтегрований із модулями генерації контенту та оцінювання компетентності. Це дозволяє формувати рішення, засновані на глибокому аналізі поточних дій і прогресу користувача.

Додатковою перевагою є можливість масштабування механізму шляхом розширення набору правил адаптації або інтеграції нових алгоритмів. Це робить систему гнучкою та придатною для використання у різних навчальних сценаріях і сферах підготовки.

Механізм динамічної адаптації реалізує модель зворотного зв'язку, працюючи циклічно в режимі реального часу. Ядро системи використовує

внутрішній таймер для періодичного запуску процедури оновлення, під час якої на основі нових метрик оцінюється поточна компетентність користувача. Далі алгоритм визначає тенденцію змін (градієнт) і перевіряє умову чутливості: якщо вона виконується, обчислюється нове скалярне значення складності, яке одразу використовується для корекції сценарію. Такий підхід гарантує реактивність і стабільність системи, забезпечуючи інтелектуальний контроль за навчальним процесом, що дозволяє адаптувати навчання під індивідуальні потреби користувача та підвищити ефективність формування професійних навичок.

3.3 Реалізація підсистеми збору та аналізу даних тренувального процесу

Підсистема збору та аналізу даних є важливим компонентом інтелектуальної тренажерної системи, оскільки вона забезпечує фіксацію ключових параметрів діяльності користувача, оцінювання ефективності виконання завдань та формування рекомендацій для подальшої адаптації навчального сценарію. Саме на основі цих даних приймаються рішення щодо зміни складності, структури тренування або надання додаткових підказок [35].

Архітектурно підсистема складається з кількох функціональних модулів:

— модуль збору телеметрії;

— модуль обробки та нормалізації даних;

— аналітичний модуль;

— модуль зберігання результатів.

На етапі збору телеметрії система фіксує широкий спектр параметрів, включаючи:

— час виконання операцій;

— кількість і тип помилок;

— частоту повторення помилок;

— стабільність дій користувача;

— швидкість реакції;

— взаємодію з об’єктами;

— завершення або переривання етапів сценарію.

Зібрані дані підлягають попередній обробці: нормалізації, фільтрації та агрегації. Це дозволяє формувати узгоджені набори параметрів, які подаються на вхід аналітичних алгоритмів. Обробка може включати виявлення аномалій, оцінку рівня шуму та фільтрацію нерелевантних подій, що зменшує вплив випадкових похибок.

Аналітичний модуль виконує інтерпретацію телеметричних даних відповідно до навчальних цілей. Він визначає рівень компетентності, виявляє прогалини у знаннях, оцінює темп навчання та прогнозує подальшу динаміку розвитку навичок. У разі потреби формуються рекомендації для модуля адаптивної генерації, які визначають зміни складності або структури сценарію [36].

Модуль зберігання результатів відповідає за ведення історії навчання, що включає інформацію про виконані завдання, сформовані рекомендації, зміну параметрів складності та результати взаємодії з тренажером. Це дає змогу здійснювати довгостроковий аналіз прогресу користувача та коригувати навчальні стратегії.

Важливим аспектом реалізації підсистеми є забезпечення узгодженості та безперервності збору даних у режимі реального часу. Дані мають фіксуватися з мінімальною затримкою, щоб аналітичний модуль міг оперативно реагувати на зміни у поведінці користувача та формувати відповідні рекомендації. Для цього реалізуються механізми асинхронного збору даних та оптимізації навантаження на систему.

Підсистема збору та аналізу даних тісно інтегрована з модулем інтелектуальної генерації, що забезпечує замкнений цикл адаптації:

— збір;

— аналіз;

— прийняття рішень;

— зміна сценарію;

— новий збір.

Така інтеграція дозволяє системі працювати у режимі самонавчання, постійно удосконалюючи сценарії та забезпечуючи персоналізований підхід до навчання.

Підсистема збору та аналізу даних є мостом між діями користувача у віртуальному світі та інтелектуальною логікою генерації сценаріїв. Її основне завдання — забезпечити точне, об'єктивне та безперервне надходження метрик, необхідних для розрахунку компетентності C_{user} у режимі реального часу.

3.3.1 Створення системи логування та метрик

Створення системи логування та метрик є одним із ключових елементів підсистеми збору та аналізу даних, оскільки саме вона забезпечує централізовану фіксацію подій, результатів взаємодії користувача із середовищем та аналітичну основу для подальшого прийняття рішень щодо адаптації сценарію. Відповідно до вимог інтелектуального тренажера, система логування повинна охоплювати широкий спектр параметрів та працювати у режимі реального часу без суттєвого впливу на продуктивність [37].

У загальному вигляді система логування реалізується як багаторівнева структура, що включає:

— низькорівневе логування подій середовища;

— логування дій та поведінки користувача;

— агрегацію та формування метрик виконання завдань;

— передачу даних у модуль аналізу та зберігання результатів.

На першому рівні фіксуються внутрішні події симулятора: поява чи видалення об’єктів, зміни станів, зіткнення, запуск тригерів та сценарних дій. Такі події дають змогу формувати повну картину змін у середовищі та забезпечувати коректність його відтворення.

Другий рівень охоплює безпосередні дії користувача: натискання, виконання завдань, результати взаємодії з об’єктами, допущені помилки, повторюваність дій та тривалість виконання ключових операцій. Фіксація такої інформації дозволяє

оцінювати прогрес, ефективність навчання та адаптацію до нових умов.

Окреме місце займає формування метрик. Метрики представляють собою агреговані показники, що відображають рівень компетентності користувача та характеризують якість його роботи в різних сценаріях. До основних метрик належать:

— тривалість виконання завдання;
— кількість помилкових дій;
— частота повторних помилок;
— точність виконання завдань;
— стабільність результатів у часі;
— кількість використаних підказок;
— кількість успішних/невдалих завершень.

Система метрик може бути розширеною, залежно від специфіки навчального процесу або цілей оцінювання. У складніших сценаріях можуть використовуватися вторинні показники — наприклад, динаміка зміни результатів між етапами, здатність користувача адаптувати поведінку, ефективність взаємодії з різними об’єктами середовища.

Дані логування та метрик зберігаються у централізованому сховищі, що може бути організоване як у вигляді локальних журналів, так і у вигляді бази даних, яка підтримує персистентність та можливість вивантаження результатів для подальшої обробки. Записи містять:

— часові мітки;
— тип події;
— ідентифікатори користувача та об’єкта;
— параметри події;
— результати операцій.

Для мінімізації навантаження на систему логування виконується асинхронно, а записи формуються у форматах, що забезпечують простоту обробки. Це дозволяє уникати затримок у роботі симуляції та забезпечувати своєчасне надходження

даних у модуль аналізу.

Таким чином, система логування та метрик формує інформаційний фундамент для інтелектуального аналізу результатів тренування, забезпечує прозорість процесу навчання та створює передумови для побудови адаптивних сценаріїв, що відповідають індивідуальним потребам користувача.

Ієрархія класів програмного ядра відображає логічну структуру системи та визначає взаємозв'язки між основними компонентами, що забезпечують функціонування інтелектуального навчального середовища. Правильна побудова ієрархічних залежностей дозволяє стандартизувати процес розробки, підвищити рівень повторного використання коду, спростити інтеграцію нових компонентів та забезпечити цілісність архітектури програмного комплексу.

У фундаменті ієрархії знаходиться базовий клас CoreObject, який задає мінімальний набір атрибутів та інтерфейс для успадкування іншими сутностями. Він визначає ключові властивості об'єктів — унікальний ідентифікатор, стан, параметри ініціалізації та базові методи взаємодії.

Від нього успадковуються дві основні групи класів:

— системні класи, що визначають загальну логіку керування та функціонування тренажера;

— Прикладні класи, що описують специфічні об'єкти сценарію та механізми взаємодії з користувачем.

Були розроблені системних класи:

— EngineController;

— ScenarioManager;

— GenerationModule;

— MetricsManager.

Прикладний рівень представлений класами, що описують сутності середовища, зокрема:

— BaseActor;

— InteractiveActor;

— ScenarioItem.

Окрему підгрупу утворюють аналітичні класи, що забезпечують оброблення зібраних даних:

— UserModel;

— Evaluator;

— HistoryTracker.

Взаємодія між класами здійснюється за принципом інверсії залежностей, що дозволяє відокремити високорівневу логіку від деталізації конкретних реалізацій. Програмне ядро залишається гнучким до змін і може легко доповнюватися новими компонентами без порушення існуючої структури.

Побудована ієрархія забезпечує:

— логічну розмежованість відповідальності між компонентами;

— масштабованість системи;

— можливість використання шаблонів проєктування;

— простоту тестування та інтеграції.

Завдяки чітко визначеній структурі ієрархія класів формує стабільний фундамент програмного ядра та забезпечує послідовність роботи усіх компонентів навчального тренажера.

Система логування реалізована мовою C++, що гарантує мінімальну затримку при фіксації подій. Для стандартизації запису інформації визначено спеціальну структуру FTrainingMetricData, яка об’єднує ключові параметри кожної події: точний час у віртуальному світі, тип події, пов’язане з нею кількісне значення, а також вектор координат, що фіксує місцезнаходження гравця або об'єкта в цей момент.

Механізм збору даних функціонує через компонент UTrainingDataComponent, який прикріплений безпосередньо до персонажа гравця. Процес побудований на використанні Unreal Delegates: актори гри викликають делегати для передачі даних цьому компоненту, де вони накопичуються у внутрішньому масиві. Цей масив слугує буфером для подальшої роботи менеджера

адаптивного сценарію (UAdaptiveScenarioManager).

На етапі фіналізації, після завершення тренувальної сесії, весь масив накопичених лог-даних серіалізується у формат JSON та зберігається на диск. Це забезпечує надійну базу для подальшої роботи з даними, зокрема для проведення поглибленого аналізу, точного відтворення (реплею) тренувальної сесії та обґрунтування статистичних висновків щодо ефективності навчання.

3.3.2 Програмна реалізація оцінки компетентності

Програмна реалізація оцінки компетентності є одним з ключових елементів підсистеми аналізу, оскільки саме вона забезпечує перетворення зібраних телеметричних даних у формалізовані показники, що характеризують рівень підготовленості користувача. В основі реалізації лежить модуль оцінювання, який аналізує результати виконання завдань, визначає актуальний рівень компетентності та передає інформацію іншим компонентам системи для подальшої адаптації сценарію.

Архітектурно механізм оцінки представлений у вигляді набору класів, що взаємодіють з системою логування та модулем генерації:

— MetricsCollector — отримує та агрегує телеметрію;

— CompetenceEvaluator — виконує інтерпретацію даних і формує оцінки;

— ProgressTracker — зберігає проміжні результати, аналізує динаміку та формує висновки щодо розвитку компетенцій.

Після отримання даних від підсистеми логування, модуль оцінки проводить їхню попередню інтерпретацію. Кожен показник обробляється відповідно до встановлених правил: визначається значення, нормалізується в єдину шкалу, а потім використовується для обчислення узагальнених характеристик. Наприклад, час виконання операцій, кількість помилок та точність дій можуть оцінюватися окремо, а потім об’єднуватись у комплексну характеристику.

Оцінка компетентності формується на основі аналізу таких параметрів:

— успішність виконання завдань;

— кількість і характер допущених помилок;

— повторюваність помилок;

— стабільність результатів у часі;

— швидкість реагування та прийняття рішень;

— потреба у додаткових підказках;

— здатність до адаптації.

Для підвищення точності оцінювання застосовується багаторівневий підхід. Спочатку аналізуються базові показники, після чого формується інтегральна характеристика, що відображає узагальнені результати підготовки користувача. В окремих випадках модуль може здійснювати уточнення оцінки на основі часових рядів або зіставлення з попередніми сесіями.

Важливим елементом є механізм довгострокового відстеження прогресу. Система зберігає інформацію про попередні оцінки, що дозволяє визначати, чи вдосконалює користувач навички, чи навпаки демонструє регрес. У разі виявлення прогалин або нестабільності результатів модуль оцінки може передавати рекомендації для модуля генерації, який вносить відповідні зміни у структуру сценарію.

У реалізації модуля оцінювання також передбачена можливість розширення. Розробник або інструктор може додавати нові метрики, вносити зміни до правил оцінювання або налаштовувати вагові коефіцієнти окремих параметрів, адаптуючи систему під конкретні цілі навчання. Це забезпечує гнучкість та масштабованість підходу, дозволяючи використовувати тренажер у різних сферах підготовки.

Узагальнені результати оцінки можуть використовуватися для:

індивідуалізації навчального процесу;

— побудови рекомендацій;

— контролю знань і навичок;

— формування зворотного зв’язку;

— порівняння між різними сесіями або групами користувачів.

Обчислення показника компетентності є критично важливим етапом, оскільки від його точності та стабільності безпосередньо залежить якість адаптації

сценарію. Реалізація цього процесу здійснюється в C++ методі UAdaptiveScenarioManager::CalculateCompetency(). Оцінка виконується не за весь час тренування, а лише за ковзне часове вікно, що забезпечує високу реактивність моделі. Функція відбирає та аналізує тільки ті події з логу, які відбулися у межах цього актуального проміжку часу.

Для того щоб об’єднати різнорідні метрики, їх попередньо зводять до єдиного уніфікованого діапазону, застосовуючи методи лінійної або сигмоїдної нормалізації. Після цього оброблені дані інтегруються в загальний показник за принципом зваженого усереднення. Вагові коефіцієнти для кожної метрики жорстко визначені у програмному коді та базуються на попередньо отриманих експертних даних.

Для завершення розрахунку та підвищення надійності системи застосовується механізм згладжування. Він дозволяє нівелювати вплив випадкових сплесків даних і підвищити стабільність показника компетентності. Це забезпечує формування плавного та передбачуваного керуючого сигналу, який використовується алгоритмом адаптації для корекції складності.

3.4 Розробка користувацького інтерфейсу та засобів візуалізації

Розробка користувацького інтерфейсу та засобів візуалізації відіграє ключову роль у забезпеченні ефективної взаємодії користувача з тренажерною системою. Від якості побудови інтерфейсу залежить не лише зручність роботи та комфорт під час виконання навчальних завдань, а й загальна швидкість засвоєння матеріалу та результативність тренувального процесу. Інтерфейс має бути інтуїтивно зрозумілим, адаптивним і здатним оперативно відображати інформацію, необхідну для навчання.

Реалізація користувацького інтерфейсу у тренажерній системі базується на принципах доступності, мінімалізму та логічної структурованості. Основними вимогами до побудови інтерфейсу є:

— чітка візуальна ієрархія елементів;

— наявність інструментів навігації;

— інформаційна насиченість без зайвого перевантаження;

— відповідність навчальним цілям;

— можливість швидкої адаптації до змін у сценарії.

У структурі інтерфейсу виділяються кілька ключових функціональних зон. Інформаційний блок відображає основні метрики тренування, такі як поточний етап, тривалість виконання, кількість помилок та індикатори складності. Для роботи з віртуальним середовищем передбачено блок взаємодії, що надає користувачу вичерпні дані про призначення об’єктів, доступні дії та можливі наслідки маніпуляцій. Важливу роль відіграє блок підказок та рекомендацій, який забезпечує методичну підтримку та сприяє формуванню правильних навичок, а завершує композицію блок управління, що містить елементи навігації та інструменти для підтвердження дій, повернення назад або збереження прогресу.

Значну роль у тренажерній системі відіграють засоби візуалізації. Їхнє призначення — забезпечити наочне представлення даних про стан середовища, дії користувача та результати виконання сценарію. Це дозволяє зменшити когнітивне навантаження на користувача, зробити процес навчання більш наочним і зрозумілим.

Серед основних засобів візуалізації можна виділити:

— графічне подання об’єктів та взаємодій у просторі;

— індикатори виконання завдань та підсистем;

— панелі візуалізації прогресу;

— виділення або підсвітку об’єктів для акцентування уваги;

— інтерактивні підказки та повідомлення.

Особлива увага приділяється можливості адаптації інтерфейсу залежно від професійної підготовки та індивідуальних потреб користувача. Для новачків можуть передбачатися детальні інструкції та підказки, тоді як для досвідчених користувачів — спрощені або скорочені схеми подачі інформації. Це створює умови для персоналізованого навчання та підвищує ефективність взаємодії з тренажером.

Інтерфейс системи тісно інтегрований із підсистемою генерації та аналізу даних. Завдяки цьому користувач та інструктор можуть оперативно відстежувати прогрес, отримувати зворотний зв’язок і здійснювати корекцію параметрів навчального процесу. Інформація оновлюється в режимі реального часу, забезпечуючи високу актуальність та достовірність.

Окрему увагу при проектуванні приділено ергономіці та управлінню когнітивним навантаженням. Інтерфейс побудований на принципах чіткої візуальної ієрархії, де критичні повідомлення — такі як аварійні стани чи грубі помилки — пріоритезуються над вторинною статистичною інформацією. Це дозволяє користувачу миттєво зчитувати ситуацію та приймати рішення, не відволікаючись на інтерфейсний шум, що є критично важливим для ефективного навчання в умовах, наближених до реальних.

Крім того, інтерфейс повинен враховувати можливість масштабування та розширення. У разі додавання нових сценаріїв або функціональних модулів система повинна дозволяти інтеграцію нових елементів інтерфейсу без суттєвих змін у загальній архітектурі. Такий підхід підвищує гнучкість платформи та забезпечує можливість її довготривалого використання.

Завдяки реалізації якісного користувацького інтерфейсу (рис 3.5) та ефективних засобів візуалізації тренажерна система стає доступною, зручною та зрозумілою, що у свою чергу сприяє підвищенню мотивації користувача, покращенню результатів навчання та розширенню можливостей для індивідуальної адаптації.

3.4.1 Інтерфейс тренувального процесу

Інтерфейс тренувального процесу є центральним засобом взаємодії користувача з тренажерною системою під час виконання навчальних сценаріїв. Він забезпечує відображення всієї необхідної інформації, а також надає інструменти управління, зворотного зв’язку та навігації у реальному часі. Важливою особливістю такого інтерфейсу є його орієнтованість на підтримку навчального процесу: він повинен сприяти засвоєнню матеріалу, не відволікати користувача від

виконання завдань і забезпечувати інтуїтивно зрозумілий доступ до основних функцій.

Рисунок 3.5 — Інтерфейс тренувального середовища з вибором сценарію.

Інтерфейс тренувального процесу (рис. 3.6) складається з кількох функціональних зон, центральною з яких є інформаційний блок. Він відображає ключові дані про стан сесії: назву та опис активного етапу, залишковий час, індикатори складності й прогресу, а також кількість помилок і статус проміжних цілей. Така організація допомагає користувачу утримувати фокус на основних завданнях та чітко розуміти загальний перебіг тренування. Для безпосередньої роботи у віртуальному середовищі призначений блок взаємодії з об’єктами, який надає характеристики предметів та інформацію про допустимі дії. У складних сценаріях цей блок додатково містить контекстні підказки, що пояснюють призначення об’єктів або можливі наслідки їх використання.

Критично важливим елементом системи є блок зворотного зв’язку, який оперативно інформує про правильність чи помилковість дій через текстові повідомлення, графічні індикатори, підсвічування або візуальні ефекти, сприяючи формуванню коректних навичок. Якщо користувач ігнорує критичні кроки або

припускається системних помилок, активується блок підказок та рекомендацій. Він пропонує текстові чи графічні роз’яснення, а за потреби може спростити завдання, демонструючи приклад правильної дії.

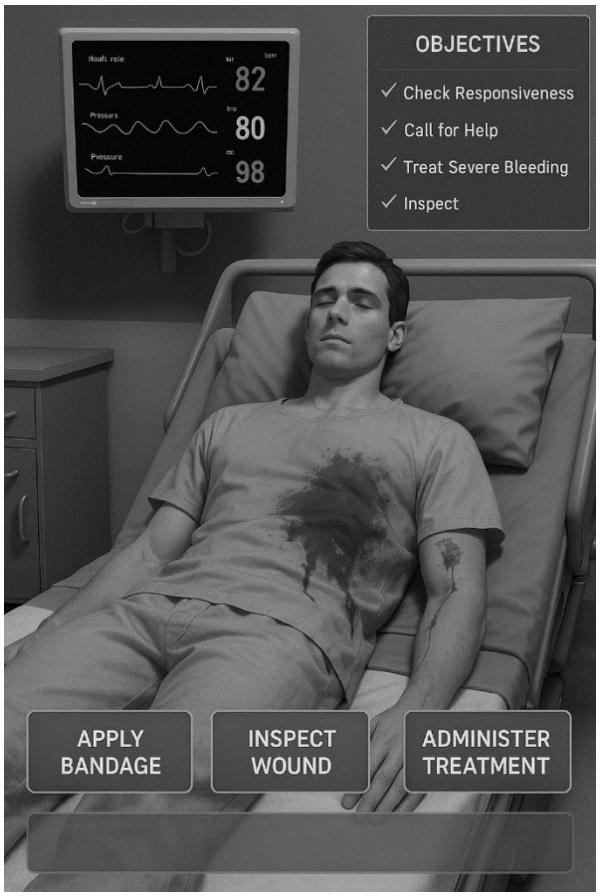

Рисунок 3.6 — Інтерфейс користувача під час медичного сценарію.

Управління сесією здійснюється через спеціальну панель, що містить інструменти для паузи, навігації між етапами, завершення роботи та виклику довідки. Її структура організована так, щоб користувач міг миттєво реагувати на зміни в сценарії, не витрачаючи час на пошук потрібних функцій. Окремим важливим аспектом реалізації є адаптивність інтерфейсу: рівень деталізації автоматично підлаштовується під кваліфікацію користувача. Для новачків система відображає розширені інструкції та ілюстрації, тоді як для досвідчених фахівців візуальний ряд спрощується, приховуючи зайве та фокусуючись лише на ключових параметрах.

Крім того, інтерфейс тренувального процесу тісно пов'язаний із підсистемою збору даних. Усі дії користувача фіксуються та передаються для аналізу компетентності. Це дозволяє здійснювати динамічне коригування складності та формувати персоналізовані сценарії навчання.

Загалом, інтерфейс тренувального процесу виступає ключовим інструментом підтримки взаємодії між користувачем та тренажером. Він забезпечує:

— оперативний доступ до потрібної інформації;

— взаємодію з навчальними об’єктами;

— інформування про помилки та досягнення;

— адаптацію під індивідуальні особливості користувача.

Завдяки цьому інтерфейс сприяє ефективному й цілеспрямованому формуванню навичок, знижує когнітивне навантаження та підвищує якість навчального процесу.

Інтерфейс користувача, реалізований засобами Unreal Motion Graphics, має мінімалістичний дизайн, що дозволяє уникнути когнітивного перевантаження та зосередити увагу на головному. Він відображає лише критично важливі параметри: поточні цілі, показники таймера та стан ключового обладнання. Особливістю системи є механізм адаптивної допомоги: при фіксації низького рівня компетентності C++ модуль ініціює Blueprint-подію, яка автоматично виводить на екран візуальні або текстові підказки. Завдяки такій тісній інтеграції з логікою інтелектуальної системи, інтерфейс виконує роль інструменту миттєвого зворотного зв'язку, оперативно коригуючи дії користувача в реальному часі.

3.4.2 Інтерфейс налаштування та моніторингу

Інтерфейс налаштування та моніторингу є важливим компонентом програмного комплексу, оскільки він забезпечує взаємодію користувача або інструктора із системою, дозволяючи виконувати конфігурацію параметрів навчального сценарію, відстежувати хід тренувального процесу та здійснювати оперативне керування середовищем. Наявність зручного інтерфейсу спрощує

роботу зі складними системами, підвищує прозорість процесу навчання і дозволяє ефективно реагувати на зміни у компетентності користувача.

У межах тренажера інтерфейс вирішує дві основні задачі:

— надання можливості налаштування параметрів навчання;

— забезпечення моніторингу перебігу тренувального процесу в реальному часі.

До функціональних можливостей інтерфейсу налаштування входить конфігурація початкових параметрів тренувального середовища, зокрема:

— вибір сценарію або його варіанта;

— визначення рівня складності;

— налаштування набору об’єктів, їхніх властивостей та розташування;

— визначення параметрів оцінювання;

— увімкнення та вимкнення допоміжних підказок;

— встановлення часових обмежень на виконання завдань.

Інтерфейс налаштування дозволяє інструктору адаптувати тренувальний процес відповідно до рівня підготовки користувача або специфіки навчальних цілей. Для зручності передбачена можливість збереження конфігурацій, що дає змогу повторно використовувати налаштування у майбутніх сесіях.

Функціонал моніторингу забезпечує відображення поточного стану тренувального середовища та результатів дій користувача. Серед основних параметрів, що відображаються у режимі реального часу:

— поточний етап сценарію;

— оцінка виконання завдання;

— кількість та характер помилок;

— час виконання операцій;

— динаміка зміни компетентності;

— активні об’єкти та їхній стан.

Інтерфейс моніторингу дозволяє інструктору контролювати перебіг навчання, вчасно виявляти труднощі та втручатися у процес за потреби. У

складніших сценаріях можуть відображатися також просторові позиції об’єктів, траєкторії руху користувача, активні тригери та події.

Інтерфейс налаштування і моніторингу тісно інтегрується з модулем інтелектуальної генерації та підсистемою збору даних. Від модуля генерації він отримує інформацію про поточний стан сценарію та можливі варіанти його зміни (рис 3.7). Від підсистеми збору даних — значення метрик, телеметрію, історичні записи та оцінки компетентності.

Рисунок 3.7 — Інтерфейс екрану результатів користувача.

Ця інтеграція дозволяє інструктору здійснювати керування тренуванням на основі об’єктивних даних, формувати індивідуальні траєкторії розвитку та забезпечувати зворотний зв’язок для користувача.

Серед основних переваг інтерфейсу налаштування та моніторингу можна виділити:

— підвищення прозорості тренувального процесу;

— можливість оперативної зміни параметрів;

— підтримку адаптивних сценаріїв;

— зручність аналізу результатів та формування висновків;

— підвищення ефективності взаємодії між інструктором та користувачем.

Інтерфейс інструктора, реалізований у вигляді окремого віджета, надає керівнику тренування або експериментатору необхідні інструменти для візуалізації процесів та повного контролю над адаптивною системою. Важливою складовою є панель керування, яка дозволяє в режимі реального часу налаштовувати критичні параметри алгоритму, що є необхідною умовою для проведення валідаційних експериментів. Зокрема, оператор може оперативно змінювати швидкість адаптації, поріг чутливості системи до змін та рівень цільової компетентності, керуючи поведінкою тренажера "на льоту".

Центральним елементом візуалізації виступає динамічний графік, що відображає розвиток трьох ключових змінних у часі: поточної компетентності користувача, рівня складності сценарію та фіксованої лінії цільової компетентності. Таке наочне подання даних дозволяє експериментатору безпосередньо спостерігати за тим, наскільки швидко і точно система реагує на дії користувача. Графік слугує інструментом верифікації, підтверджуючи, що реальна компетентність учня дійсно утримується в межах заданого цільового рівня, як це передбачено теоретичною моделлю.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ВАЛІДАЦІЯ ЕФЕКТИВНОСТІ

4.1 Обґрунтування методики експериментальних досліджень

Обґрунтування методики експериментальних досліджень є важливим етапом наукової роботи, оскільки воно визначає логіку, структуру та інструментарій оцінювання ефективності запропонованих методів і засобів інтелектуальної генерації навчального середовища. Ретельно розроблена методика дозволяє забезпечити об’єктивність результатів, відтворюваність експерименту та можливість їх подальшого аналізу й порівняння.

Основною метою проведення експериментальних досліджень є оцінювання того, наскільки запропонована система здатна адаптивно формувати навчальні сценарії, підвищувати ефективність засвоєння матеріалу та коригувати рівень складності відповідно до змін компетентності користувача. Для досягнення цієї мети необхідно оцінити роботу програмного комплексу в умовах, наближених до реального використання, а також дослідити вплив окремих його компонентів на кінцевий результат [38].

Методика дослідження ґрунтується на комплексному підході, першим кроком якого є чіткі визначення цілей та критеріїв оцінки. На цьому етапі формулюються ключові питання експерименту: чи забезпечує система коректну адаптацію складності, як змінюється рівень компетентності користувача, чи спостерігається зменшення помилок і покращення часу виконання, а також чи зберігається мотивація до навчання. Для кількісного вимірювання ефективності застосовується набір різнопланових метрик: операційних (час, частота помилок), поведінкових (швидкість реакції, стійкість результатів) та інтегральних, що відображають загальну динаміку прогресу.

Організація експерименту передбачає створення контрольованого середовища, де всі учасники виконують однакові завдання, що дозволяє коректно порівнювати результати. Для дослідження формуються групи користувачів із різним рівнем початкової підготовки, щоб перевірити здатність системи

адаптуватися до індивідуальних відмінностей у знаннях та навичках. Під час роботи з тренажером у режимі реального часу система не лише збирає телеметрію, а й динамічно коригує сценарій відповідно до поточного рівня компетентності учасника.

Завершальним етапом є обробка та аналіз зібраних даних, де особлива увага приділяється динаміці розвитку навичок та стабільності результатів. Важливим аспектом методики є забезпечення повторюваності експерименту: фіксація всіх параметрів та використання стандартизованого інструментарію для збору даних дозволяють мінімізувати вплив зовнішніх факторів і гарантувати об’єктивність зроблених висновків.

Розроблена методика експериментальних досліджень забезпечує системний підхід до оцінки ефективності запропонованої моделі інтелектуальної генерації, дозволяє виявити її переваги та недоліки, а також визначити напрямки подальшого вдосконалення.

4.1.1 Формування контрольних груп та вибірковість

Формування контрольних груп та визначення вибірковості дослідження є важливими складовими експериментальної методики, оскільки вони забезпечують об’єктивність та валідність отриманих результатів. Правильний підбір учасників дозволяє мінімізувати вплив зовнішніх факторів, порівнювати ефективність різних підходів до навчання та узагальнювати результати на ширшу популяцію користувачів.

Першим етапом є визначення характеристик вибіркової сукупності. До уваги беруться такі параметри, як:

- рівень попередньої підготовки;
- загальна технічна грамотність;
- досвід взаємодії з подібними тренажерами;
- психологічні та поведінкові особливості;
- доступність до проведення дослідження.

Вибірка формується з урахуванням репрезентативності, тобто можливості відобразити різні типи користувачів, що потенційно будуть працювати із системою в реальних умовах.

Наступним етапом є розподіл учасників дослідження, який зазвичай передбачає виділення щонайменше двох основних груп: контрольної та експериментальної. Представники контрольної групи проходять навчання за статичним сценарієм або з використанням традиційної системи без адаптивних механізмів, що дозволяє використовувати їхні показники як еталон для порівняння результатів. Натомість учасники експериментальної групи виконують завдання за підтримки адаптивної системи, яка динамічно змінює рівень складності відповідно до їхніх дій та темпу освоєння матеріалу, що дає змогу оцінити реальний вплив механізмів адаптації. За необхідності також здійснюється додаткова сегментація на підгрупи за рівнем компетентності (початковий, середній, просунутий), що забезпечує глибше дослідження специфіки роботи системи з різними категоріями користувачів.

При формуванні груп важливо забезпечити їхнє балансування — рівномірний розподіл учасників за ключовими параметрами (рівень підготовки, досвід, попередні результати). Це мінімізує ризик того, що різниця в результатах буде зумовлена нерівномірністю складу груп, а не впливом навчальної системи.

Для підвищення точності дослідження також рекомендується використовувати випадкове призначення учасників до груп, що зменшує вплив суб’єктивних факторів. У разі обмеженої кількості учасників можливий ручний підбір із дотриманням принципу рівноваги між групами.

Особлива увага приділяється розміру вибірки. Вона має бути достатньою для виявлення статистично значущих відмінностей між групами. У рамках роботи досить часто використовується кілька десятків учасників, але за потреби масштабування система може бути протестована на розширеній вибірці.

Під час проведення навчальних сесій усі учасники працюють у стандартизованих умовах: ідентичний інструктаж, однакові початкові параметри

сценаріїв, фіксація результатів через єдину систему збору даних. Це забезпечує порівнюваність результатів та високу якість експерименту.

Для забезпечення статистичної достовірності результатів формуються дві групи учасників зі схожим початковим рівнем знань у предметній області, що дозволяє мінімізувати вплив зовнішніх факторів. Контрольна група проходить тренування у статичному середовищі з фіксованою складністю (наприклад, на середньому рівні), де логіка сценарію залишається незмінною протягом усієї сесії. Натомість експериментальна група працює в адаптивних умовах, де модуль інтелектуальної генерації динамічно змінює складність завдань, утримуючи рівень компетентності користувача близьким до заданого цільового показника.

4.1.2 Програма та етапи тестування

Процес тестування реалізується у три послідовні етапи. Спочатку обидві групи виконують ідентичний статичний сценарій для фіксації базового рівня підготовки, що гарантує коректність порівняння учасників. Далі слідує навчальний етап, під час якого контрольна група проходить серію сесій із фіксованою складністю, а експериментальна — в умовах динамічної адаптації, при цьому загальна тривалість тренувань залишається незмінною для всіх. На завершальній стадії проводиться повторне тестування на базі еквівалентного статичного сценарію, що дозволяє визначити фінальний рівень знань та об’єктивно оцінити досягнутий приріст компетентності.

4.1.3 Визначення критеріїв оцінки та статистичних методів

Для валідації роботи системи застосовуються два класи метрик. До першої групи належать показники ефективності навчання, ключовим з яких є приріст компетентності — різниця між фінальним та початковим рівнями знань. Основним критерієм успішності симулятора вважається суттєве перевищення цього показника в експериментальній групі порівняно з контрольною, а додатковим параметром виступає час освоєння, необхідний для досягнення визначеного порогового рівня. Друга група метрик відповідає за якість самої адаптації та

включає таргет-відхилення, що визначає середню різницю між поточною та цільовою компетентністю, а також стабільність процесу, яка оцінюється за частотою та амплітудою різких змін складності сценарію.

4.2 Валідація працездатності моделі адаптивної складності

Валідація працездатності моделі адаптивної складності є ключовим етапом експериментального дослідження, оскільки вона підтверджує здатність системи адекватно змінювати навчальний сценарій відповідно до рівня підготовленості користувача. Основна мета валідації полягає у визначенні того, наскільки реалізована модель здатна забезпечувати ефективну індивідуалізацію навчального процесу, зберігаючи при цьому логічну послідовність та відповідність навчальним цілям.

Процес валідації охоплює аналіз декількох аспектів функціонування моделі. Насамперед перевіряється коректність реакції на зміни у поведінці користувача. Модель має оперативно реагувати на підвищення або зниження рівня компетентності, змінюючи складність завдань таким чином, щоб підтримувати оптимальний рівень виклику. Це дозволяє уникати ситуацій, коли користувач або перевантажений складним завданням, або, навпаки, не отримує достатнього навчального стимулу [39].

Для цього розглядаються такі показники:

— темп зміни складності під час навчання;

— відповідність рівня складності фактичній успішності користувача;

— стійкість результатів після зміни навчальних умов;

— здатність системи коригувати поведінку при різких змінах компетентності;

— загальний вплив адаптації на рівень помилок і швидкість реагування.

Важливу роль у валідації відіграє порівняння результатів експериментальної групи, що працює з адаптивною моделлю, з контрольною групою, яка використовує статичний сценарій. Якщо експериментальна група демонструє вищу

динаміку розвитку компетентності, зниження кількості помилок або підвищення ефективності виконання завдань, це свідчить про працездатність запропонованої моделі.

Окремо оцінюється стабільність роботи адаптивного механізму. Модель не повинна різко змінювати складність, оскільки різкі стрибки можуть призвести до дезорієнтації або втрати мотивації користувача. Навпаки, адаптація має бути плавною, із поступовими змінами складності відповідно до прогресу. Це перевіряється шляхом аналізу послідовності модифікацій сценарію, їх частоти та впливу на навчальний процес.

Додатково валідація включає аналіз коректності роботи механізмів корекції. У випадках, коли користувач систематично припускається помилок, модель повинна застосовувати допоміжні засоби — зниження складності, надання підказок або додаткову інформаційну підтримку. Перевіряється, наскільки швидко й адекватно система реагує на подібні ситуації.

Методика валідації також передбачає дослідження впливу адаптивного механізму на суб’єктивне сприйняття навчання. Для цього можуть використовуватися опитувальники, інтерв’ю та інші інструменти, що дозволяють оцінити рівень задоволеності, втому, мотивацію та комфорт під час взаємодії з системою.

Якщо система демонструє позитивний вплив на розвиток компетентності, забезпечує плавну адаптацію складності та сприяє підвищенню успішності, її можна вважати працездатною та придатною до застосування у реальних навчальних процесах.

4.2.1 Перевірка чутливості та реактивності

Необхідно підтвердити, що система здатна швидко реагувати на значні зміни C_{user}:

— у сценарій штучно вводиться критична подія (помилка), яка різко знижує C_{user} (наприклад, C_{user} падає з 90 до 40);

— фіксується Час Реакції (T_{react}) — інтервал між падінням C_{user} та початком корекції складності ΔD (зміна параметрів D);

— T_{react} має бути менше встановленого порогу (наприклад, $T_{react} < 5$ сек), що підтверджує чутливість, контрольовану параметром ε (рис 4.1).

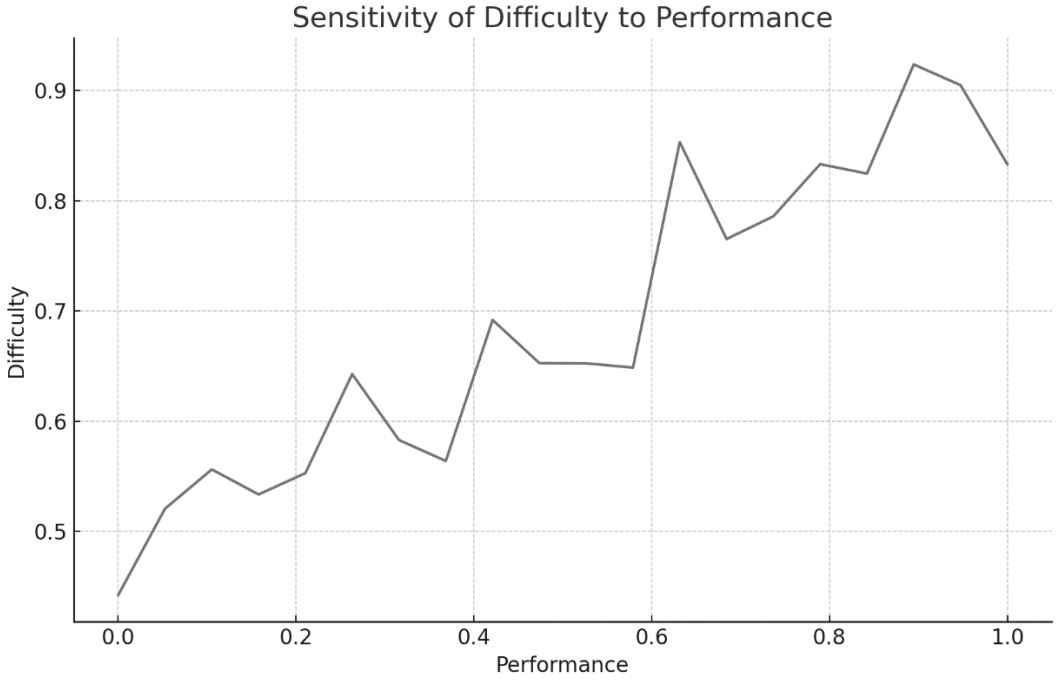

Рисунок 4.1 — Графік складності до продуктивності

4.2.2 Аналіз стабільності та уникнення осциляцій

Перевірка стабільності критично важлива, оскільки хаотична зміна складності може дезорієнтувати користувача.

— проведення тривалого тренування з високим коефіцієнтом швидкості адаптації α (наприклад, α=0.8) та з низьким α (наприклад, α=0.2);

— аналізується графік D(t) для оцінки частоти та амплітуди осциляцій;

— при оптимальному α графік D(t) має демонструвати плавне наближення до рівноважного стану, підтверджуючи, що модель зворотного зв'язку ефективно згладжує керуючий сигнал (рис 4.2).

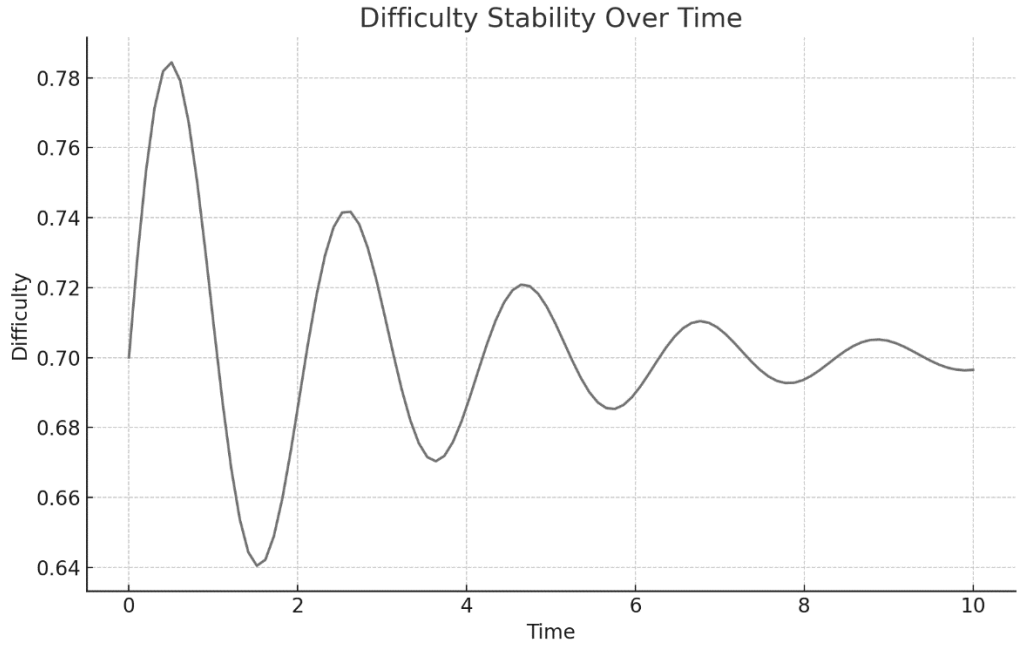

Рисунок 4.2 — Графік складності до часу

4.3 Оцінка ефективності адаптивного навчального середовища

Оцінка ефективності адаптивного навчального середовища є завершальним та одним із найважливіших етапів експериментального дослідження. Вона дозволяє визначити, наскільки побудована система досягла цілей навчання, сприяла розвитку компетентностей користувачів та забезпечила покращення результатів порівняно з традиційними підходами.

Основна мета оцінювання полягає у встановленні впливу адаптивних механізмів на якість навчального процесу. Для цього аналізується низка показників, що відображають успішність користувачів, їхню здатність до самостійного прийняття рішень, рівень помилок та загальний прогрес [40].

Оцінка ефективності системи проводиться комплексно за кількома стратегічними напрямами. У першу чергу порівнюється результативність виконання завдань — час, точність дій та кількість помилок, а також аналізується динаміка розвитку компетентності, де прискорений прогрес учасників свідчить про успішність адаптивного підходу. Окремим критерієм виступає стійкість результатів, що демонструє здатність користувачів стабільно застосовувати

навички та вказує на довготривале засвоєння знань. Паралельно досліджується гнучкість самої системи, зокрема її здатність поступово підвищувати складність без створення надмірного навантаження. Доповнюють аналіз суб'єктивні оцінки задоволеності та мотивації, отримані через опитування, а фінальне підтвердження ефективності базується на порівняльному аналізі: вищі показники експериментальної групи порівняно з контрольною слугують доказом дієвості запропонованої моделі.

Також досліджується вплив адаптації на окремі аспекти поведінки користувача:

— здатність працювати в умовах підвищеної складності;

— швидкість прийняття рішень;

— ефективність управління ресурсами чи об'єктами;

— рівень самостійності.

Додатково аналізуються показники повторюваності помилок. Їх значне зменшення після адаптації свідчить про те, що система допомагає користувачу краще засвоювати матеріал і уникати попередніх недоліків.

На завершальному етапі результати оцінювання узагальнюються, формується висновок про ефективність навчального середовища, а також визначаються можливі напрямки подальшого вдосконалення. Серед таких напрямків можуть бути:

— розширення набору сценаріїв;

— удосконалення моделі адаптивності;

— інтеграція нових типів зворотного зв'язку;

— удосконалення механізмів візуалізації та інтерфейсу.

Оцінка ефективності адаптивного навчального середовища демонструє переваги застосування інтелектуальної генерації у навчальних тренажерах, підтверджує практичну цінність запропонованої моделі та створює основу для її подальшого розвитку й впровадження у реальні освітні процеси.

4.3.1 Статистичний аналіз приросту компетентності

Статистичний аналіз приросту компетентності є ключовим етапом оцінювання результативності навчального процесу в адаптивному середовищі. Його метою є визначення того, наскільки значущим є розвиток навичок користувачів у порівнянні з початковим рівнем, а також порівняння досягнутих результатів між експериментальною та контрольною групами.

Основним завданням аналізу є кількісне вимірювання ефекту, який справляє адаптивна система на освітній результат. Для цього використовуються різні підходи статистичної обробки даних, що дозволяють визначити як загальні тенденції, так і індивідуальні особливості розвитку компетентності.

На першому етапі відбувається збирання телеметричних даних щодо:

— початкового рівня компетентності користувача;

— результатів виконання завдань у процесі навчання;

— завершального рівня компетентності після проходження сценарію.

Для кожного учасника фіксуються показники часу виконання, кількості помилок, стабільності результатів та використання підказок. Ці дані нормалізуються та перетворюються у єдину шкалу, що дозволяє порівнювати результати між користувачами з різним рівнем підготовки.

Приріст компетентності розраховується як різниця між фінальною та початковою оцінками для кожного користувача. Такий підхід дозволяє визначити індивідуальну динаміку розвитку навичок. У випадку експериментальної групи, де застосовувався адаптивний сценарій, очікується більш високий приріст компетентності порівняно з контрольною групою.

Для об’єктивної оцінки ефективності адаптивної системи проводиться порівняння середніх показників приросту між групами. Якщо у експериментальній групі фіксується істотно вищий приріст, це свідчить про позитивний вплив адаптивних механізмів.

Особливу увагу приділено:

— відносному приросту компетентності;

— швидкості оволодіння новими навичками;

— стабільності результатів у часі;

— зниженню рівня повторюваних помилок.

Важливим етапом є аналіз розподілу показників приросту компетентності. Він дозволяє:

— визначити, чи є прогрес рівномірним у межах групи;

— виявити користувачів, які демонструють найкращу або найгіршу динаміку;

— оцінити вплив адаптивних механізмів на різні категорії учасників (новачків, середній рівень, досвідчених).

У разі наявності значних відхилень проводиться додатковий аналіз, спрямований на з’ясування їх причин — наприклад, недостатній рівень підготовки, проблеми із взаємодією або неправильне використання інструментів.

На заключному етапі результати статистичного аналізу узагальнюються. Якщо експериментальна група демонструє значуще підвищення компетентності порівняно з контрольною, можна зробити висновок, що адаптивна система є ефективною. Також досліджується вплив адаптації на різні компоненти навчання: швидкість, точність, стабільність.

Отримані результати лягають в основу висновків щодо ефективності розробленої системи та можуть бути використані для її подальшого вдосконалення: оптимізації алгоритмів адаптації, розробки персоналізованих сценаріїв, розширення набору метрик компетентності (рис 4.3).

Для кількісного доведення ефективності методики проводиться порівняльний аналіз приросту компетентності між контрольною та експериментальною групами. Спочатку для кожної вибірки розраховується середній показник покращення знань, після чого застосовується t-критерій Стьюдента для незалежних вибірок. Цей статистичний метод дозволяє перевірити гіпотези дослідження: спростувати припущення про відсутність суттєвої різниці між групами та підтвердити альтернативну гіпотезу, згідно з якою навчання в

адаптивному середовищі забезпечує значно вищий приріст професійних навичок порівняно з традиційним підходом.

Якщо ρ-значення статистичного тесту $\rho < 0.05$, нульова гіпотеза відхиляється, і вважається доведеним, що адаптивне середовище забезпечує статистично значуще вищий приріст компетентності.

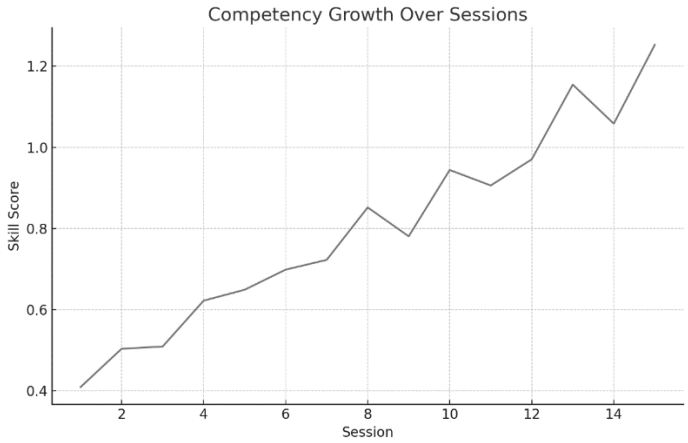

Рисунок 4.3 — Графік зростання компетентності користувача.

4.3.2 Експериментальна перевірка ефективності системи

За результатами проведених експериментальних досліджень було встановлено, що використання адаптивного навчального середовища, побудованого на основі інтелектуальної генерації контенту, має позитивний вплив на ефективність формування компетентностей користувачів. Отримані результати демонструють помітні переваги розробленої системи порівняно з традиційними статичними підходами до організації навчального процесу.

По-перше, експериментальна група, що працювала із застосуванням механізмів динамічної адаптації складності, продемонструвала суттєво вищий приріст компетентності порівняно з контрольною групою. Прогрес зафіксовано як у загальних показниках успішності, так і у стабільності виконання завдань. Це свідчить про те, що адаптивний сценарій дозволяє користувачам ефективніше засвоювати навички, поступово ускладнюючи завдання відповідно до їхнього рівня підготовки.

По-друге, аналіз кількості помилок і повторюваності неправильних дій показав, що учасники експериментальної групи значно рідше припускалися типових помилок під кінець навчання. Це може вважатися підтвердженням того, що адаптивні механізми сприяють кращому розумінню структури завдань та підвищують рівень самостійності користувачів.

По-третє, учасники експериментальної групи демонстрували підвищену швидкість виконання завдань. Це вказує на формування стійких навичок, оптимізацію процесу прийняття рішень та підвищення ефективності взаємодії з навчальним середовищем. Зменшення часу на виконання при одночасному збереженні або підвищенні якості результату є важливим показником у тестуванні тренажерних систем.

Крім того, аналіз суб’єктивних оцінок, отриманих за допомогою опитувань, показав, що користувачі високо оцінюють інтерфейс системи, комфорт під час роботи та логічність розвитку навчального процесу. Більшість учасників відзначили, що адаптивна система створює індивідуалізовані умови навчання, підвищує мотивацію та сприяє кращому зануренню у завдання.

Разом із тим, результати дослідження засвідчили кілька аспектів, що потребують подальшого вдосконалення. Серед них — необхідність більшого урахування психологічних особливостей користувачів, оптимізація алгоритмів адаптації для запобігання надто частим змінам складності та розширення набору доступних сценаріїв для покриття ширшого спектра навчальних ситуацій.

Загалом проведені експерименти підтвердили працездатність і переваги розробленої моделі адаптивного навчального середовища. Вона забезпечує:

— підвищення ефективності та динаміки розвитку компетентності;

— зменшення кількості помилок і покращення стабільності результатів;

— підвищення мотивації та залученості користувачів;

— динамічну адаптацію складності відповідно до індивідуальних потреб.

Отримані результати можуть слугувати основою для подальшого розвитку системи, її масштабування та впровадження у реальні навчальні процеси.

5 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

5.1 Проведення комерційного та технологічного аудиту розробки

Актуальність проведення комерційного та технологічного аудиту розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів в Unreal Engine зумовлена зростаючим попитом на адаптивні тренувальні платформи. У галузях оборони, медицини, енергетики, транспорту та промисловості кількість задач, що потребують високоточних симуляцій, стрімко зростає. Замовники очікують не просто візуалізації, а інтелектуальних систем, здатних автоматично створювати варіативні сценарії, підлаштовувати складність під рівень користувача та забезпечувати відтворюваність критичних ситуацій. На цьому фоні виникає потреба оцінити конкурентоспроможність продукту, визначити його ринковий потенціал і ступінь готовності до комерціалізації.

Технологічний аудит є необхідним для визначення рівня інноваційності та технічної життєздатності розробки. Створення генератора навчальних середовищ в Unreal Engine передбачає інтеграцію складних алгоритмів ШІ, процедурної генерації, фізичних моделей та інструментів аналітики навчальних траєкторій. Такий аудит дозволяє оцінити архітектуру системи, якість коду, масштабованість, сумісність із VR/AR-рішеннями, продуктивність у реальних сценаріях та потенційні технічні ризики. Він також дає можливість визначити, наскільки технологія відповідає міжнародним стандартам та найкращим практикам у сфері симуляцій та тренажерної підготовки.

Комерційний аудит, своєю чергою, дає змогу визначити цінність продукту для цільових сегментів ринку та його економічну доцільність. Він охоплює аналіз конкурентів, оцінку бізнес-моделей, визначення потенційних клієнтів, готовність ринку до впровадження інновації та прогнозування фінансових показників. Для інтелектуального генератора навчальних середовищ особливо важливими є гнучкість монетизації, можливість кастомізації під галузеві потреби та перспективи масштабування на міжнародні ринки. Саме синергія комерційного та технологічного аудитів забезпечує обґрунтованість рішень щодо подальшої

розробки, інвестицій та впровадження продукту.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування) та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 5.1.

Таблиця 5.1 — Результати оцінювання науково-технічного рівня

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	3	3	3
Ринкові переваги (наявність аналогів)	2	2	2
Ринкові переваги (ціна продукту)	2	2	2
Ринкові переваги (технічні властивості)	3	3	3
Ринкові переваги (експлуатаційні витрати)	2	2	2
Ринкові перспективи (розмір ринку)	3	3	3
Ринкові перспективи (конкуренція)	2	2	2
Практична здійсненність (наявність фахівців)	3	3	3
Практична здійсненність (наявність фінансів)	2	2	2
Практична здійсненність (необхідність нових матеріалів)	3	3	3
Практична здійсненність (термін реалізації)	2	2	
Практична здійсненність (розробка документів)	3	3	
Сума балів	30	30	
Середньоарифметична сума балів, СБ	30		

За результатами розрахунків, наведених в таблиці 5.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інтелектуального

генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування) – середній.

5.2 Розрахунок витрат на здійснення розробки

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р},$$

де к – кількість посад дослідників, залучених до процесу дослідження;

$М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні, приймаємо 22 дні;

t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 5.2.

Таблиця 5.2 — Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	33 000	1500	18	27000
Розробник	28 000	1273	55	70000
Консультанти	25 000	1136	8	9091
Всього:	106091			

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$C_i = \frac{M_м \cdot K_i \cdot K_c}{T_р \cdot t_{зм}},$$

де $M_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2025 році $M_м$=8000 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати, складає 1,1;

$T_р$ – середня кількість робочих днів в місяці, приблизно $T_р$ = 21…23 дні, приймаємо 22 дні;

$t_{зм}$ – тривалість зміни, год., приймаємо 8 год.

Витрати на заробітну плату робітників з урахуванням мінімального коефіцієнта співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств.

Таблиця 5.3 — Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Підготовка та адаптація навчальних сцен	36	3	59	1,18	2124
Налагодження інтерактивних об'єктів середовища	50	4	63,5	1,27	3175
Проведення випробувань навчального середовища	20	3	59	1,18	1180
Всього					6479

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (106091 + 6479) = 11257 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (106091 + 6479 + 11257) = 27242 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_р$ – основна заробітна плата робітників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов’язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Витрати на матеріали М, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_i,$$

де $Н_i$ – кількість матеріалів і-го виду, шт.;

$Ц_i$ – ціна матеріалів і-го виду, грн.;

$К_i$ – коефіцієнт транспортних витрат,

$К_i$ = (1,1…1,15); n – кількість видів матеріалів.

Таблиця 5.4 — Матеріали, що використані на розробку

Найменування матеріалів	Ціна за одиницю, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Папір для документації, пачка	200	2	400
Канцелярські матеріали (ручки, маркери)	150	1	150
USB-накопичувач 64 ГБ	600	1	600
Всього, з врахуванням коефіцієнта транспортних витрат	1265		

Розрахунок витрат на комплектуючі. Витрати на комплектуючі К, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i,$$

де $Н_i$ – кількість комплектуючих і-го виду, шт.;

$Ц_i$ – ціна комплектуючих і-го виду, грн.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

n – кількість видів комплектуючих.

Таблиця 5.5 — Комплектуючі, що використані на розробку

Найменування комплектуючих	Ціна за одиницю, грн.	Витрачено	Вартість витрачених комплектуючих, грн.
SSD диск 512 ГБ для робочої станції	2500	1	2500
Оперативна пам'ять 16 ГБ	2000	1	2000
Всього, з врахуванням коефіцієнта транспортних витрат			4995

Спецустаткування для наукових (експериментальних) робіт. Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами.

$$В_{спец} = \sum_{1}^{к} Ц_i \cdot С_{пр.i} \cdot К_i,$$

де $Ц_i$ – ціна придбання спецустаткування і-го виду, грн.;

$С_{пр.i}$ – кількість одиниць спецустаткування відповідного виду, шт.;

$К_i$ – коефіцієнт транспортних витрат,

$К_i$ = (1,1…1,15);

n – кількість видів спецустаткування.

Таблиця 5.6 — Витрати на придбання спецустаткування

Устаткування	Ціна за од., грн.	Витрачено	Вартість устаткування, грн.
VR-шолом для тесту	18000	1	18000
Проєкційний екран	12000	1	12000
Всього, з врахуванням коефіцієнта транспортних витрат			33000

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_і,$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.;

$С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Таблиця 5.7 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Windows 11 Pro (ліцензія)	6 000	1	6000
JetBrains Rider (річна ліцензія)	8 000	1	8000
Сервіси керування проєктами (річна підписка)	3 000	1	3000
Всього, з врахуванням коефіцієнта інсталяції та налагодження			18870

Амортизація обладнання, комп’ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи. У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12},$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп’ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.;

t – термін використання основного фонду, місяці;

$Т_в$ – термін корисного використання основного фонду, роки.

Таблиця 5.8 — Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Робоча станція розробника (ПК)	55 000	3	2	3055,6
Робоча станція тестувальника (ПК)	45 000	3	2	2500,0
Сервер для зберігання даних	70 000	4	2	2916,7
Всього				8472

Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

Таблиця 5.9 — Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Робоча станція розробника (ПК)	0,50	150
Робоча станція тестувальника (ПК)	0,5	110
Сервер для зберігання даних	0,70	180

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД}$$

$$= \frac{0{,}5 \cdot 150 \cdot 4{,}32 \cdot 0{,}75}{0{,}98} + \frac{0{,}5 \cdot 110 \cdot 4{,}32 \cdot 0{,}75}{0{,}98} + \frac{0{,}7 \cdot 180 \cdot 4{,}32 \cdot 0{,}75}{0{,}98}$$

$$= 846{,}4 \text{ грн.},$$

де W_i – встановлена потужність обладнання, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, 4,32 грн.;

$К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються від 50% до 100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (106091 + 6479) \cdot \frac{55}{100} = 61913 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…200% від суми основної заробітної плати дослідників та робітників за формулою:

$$Внзв = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (106091 + 6479) \cdot \frac{135}{100} = 151969 \text{ грн.},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування). Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = 106091 + 6479 + 11257 + 27242 + 1265 + 4995 + 33000 + 18870 + 8472 + 846 + 61913 + 151969 = 432400 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи з розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування) та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{432400}{0,5} = 864800 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи.

Оскільки, якщо науково-технічна розробка знаходиться на стадії розробки дослідного зразка, то η=0,5.

5.3 Розрахунок економічної ефективності

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування), є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm \Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right),$$

$\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни), 80000 грн.; N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки, 25 шт., 55 шт., 65 шт.; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, 300000 грн.; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів, 380000 грн.; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році, 10 шт., 40 шт., 50 шт.. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 3 роки, тому 1-й рік – $\Delta П1$=860656 грн., 2-й рік – $\Delta П2$=2704918 грн., 3-й рік – $\Delta П3$=3319672 грн.

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування):

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{860656}{(1+0{,}1)^1} + \frac{2704918}{(1+0{,}1)^2} + \frac{3319672}{(1+0{,}1)^3} = 5512002 \text{ грн.},$$

де ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=3 роки);

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування). Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ = 3 \cdot 864800 = 2594401 \text{ грн.}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування) та її комерціалізацію, це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай кінв=1...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв

для симуляторів у Unreal Engine (навчальне середовище для тренування) становитиме:

$$E_{абс} = ПП - PV = 5512002 - 2594401 = 74664 \text{ грн.},$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування), грн.;

PV – теперішня вартість початкових інвестицій, грн.

Оскільки $E_{абс}>0$, то можемо припустити про потенційну зацікавленість у розробці інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування).

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування) вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування), розраховується за формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} = \sqrt[з]{1 + \frac{2917600}{2594401}} = 0{,}71,$$

де $T_ж$ – життєвий цикл розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування), роки.

Далі розраховуємо період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування):

$$T_о = \frac{1}{E_в} = \frac{1}{0{,}71} = 1{,}41 \text{ роки.}$$

Оскільки $T_о$<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування) і може спонукати потенційного інвестора профінансувати впровадження цієї розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів у Unreal Engine (навчальне середовище для тренування) та виведення її на ринок.

Проведений комерційний і технологічний аудит розробки інтелектуального генератора навчальних середовищ і сценаріїв для симуляторів на Unreal Engine засвідчив середній рівень науково-технічної готовності та комерційного потенціалу продукту. Експертна оцінка за 12 критеріями показала узгоджену оцінку (СБ = 30), що відображає наявність перспектив розвитку за умови подальших інвестицій у технічне вдосконалення та ринкове позиціонування.

Розрахунок витрат на виконання Науково-дослідної роботи показав, що загальна вартість розробки становить 432,4 тис. грн, а з урахуванням коефіцієнта етапності — 864,8 тис. грн. Можливий економічний ефект від комерціалізації суттєвий: прогнозоване збільшення чистого прибутку за 3 роки становить 5,51 млн

грн, що перевищує необхідні інвестиції (PV = 2,59 млн грн). Це свідчить про інвестиційну привабливість продукту та доцільність його подальшої розробки й впровадження.

Узагальнюючи, можна зробити висновок, що проєкт має економічні передумови для комерціалізації та здатен забезпечити інвестору значний позитивний фінансовий результат за умови оптимізації технологічних рішень і правильного виходу на цільові ринки.

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи на тему "Програмне забезпечення ігрової механіки з використанням штучного інтелекту в Unreal Engine. Модуль динамічної генерації ігрового простору" було проведено комплексне дослідження, обґрунтування та реалізацію Моделі, алгоритмічного та програмного забезпечення інтелектуальної генерації сценаріїв (ІГС) для створення адаптивного симуляційного навчального середовища

Було розглянуто та класифіковано сучасні симуляційні навчальні середовища (СНС), підтверджено їхню критичну залежність від якості та варіативності тренувального контенту. Це дало змогу виявити ключовий недолік існуючих рішень — статичність сценаріїв та відсутність інтелектуальної адаптації, що обґрунтувало необхідність розробки ІГС.

Проаналізовано використання передових методів процедурної (ПКГ) та інтелектуальної генерації (ІГС) контенту. Дослідження підтвердило, що лише інтеграція ІГС з механізмами зворотного зв'язку дозволить створити середовище, що динамічно реагує на дії користувача.

Вивчено та обґрунтовано вибір Unreal Engine та C++ як оптимальної технологічної платформи, оскільки вона забезпечує необхідний рівень фотореалізму та високу продуктивність для обчислювально інтенсивних алгоритмів адаптації.

Формалізовано наукову проблему та розроблено математичну модель оцінки поточної компетентності користувача ($\mathbf{C_{user}}$) на основі множини зважених та нормалізованих метрик M). Ця модель стала основою для керування адаптацією.

Реалізовано модульну архітектуру програмного комплексу (Розділ 3), що включає ядро ІГС (AI Director) на C++ (UAdaptiveScenarioManager), відокремлене від Модуля Моніторингу (UTrainingDataComponent), що забезпечує гнучкість та масштабованість.

Створено алгоритм динамічної корекції складності (модель зворотного зв'язку), який використовує градієнт адаптації ($\mathbf{G_{diff}} = \mathbf{C_{user}} - \mathbf{C_{target}}$) для

автоматичного коригування складності D, підтримуючи користувача у зоні оптимального навчального навантаження.

Розроблено та реалізовано алгоритм інтелектуального розміщення об'єктів, який динамічно генерує об'єкти сценарію, використовуючи NavMesh Queries та враховуючи поточну складність D для забезпечення логічно когерентного простору.

Проведено комплексне експериментальне дослідження, що включало розробку методики тестування на контрольних групах та валідацію роботи алгоритму.

Технічна валідація підтвердила, що модель адаптивної складності є реактивною та стабільною, здатна швидко реагувати на зміни $\mathbf{C_{user}}$ (контроль параметрів α та ε).

Оцінка ефективності шляхом статистичного аналізу приросту компетентності ($\Delta\mathbf{C}$) довела, що тренування в адаптивному середовищі забезпечує статистично значуще вищий рівень освоєння навичок порівняно зі статичним середовищем ($\rho < 0.05$).

В результаті виконання магістерської роботи розроблено програмне забезпечення модуля інтелектуальної генерації сценаріїв, яке використовує інтелектуальні алгоритми та сучасні методи процедурної генерації. Розроблений модуль може бути використаний для створення персоналізованих та високоефективних тренувальних програм, що підвищують швидкість, глибину освоєння професійних навичок та занурення користувачів у процес.

Таким чином всі поставленні задачі були виконані, і мета роботи була досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інтелектуальний генератор навчальних середовищ і сценаріїв для симуляторів у Unreal Engine / Чорний В. В., Колесніченко Л. А., Черняк О. І. // Матеріали міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)» (Вінниця, 2026 р.) — [Електронний ресурс]. — Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26513.

2. Rabin S. Introduction to Game Development / S. Rabin. – [S. l.] : Cengage Learning, 2009. – 1008 с.

3. Eberly D. H. Game Physics / D. H. Eberly. – [S. l.] : Elsevier, 2006.– 816 с.

4. Vygotsky L. S. Mind in society: The development of higher psychological processes / L. S. Vygotsky. – [S. l.] : Harvard University Press, 1978. – 176 с.

5. Hendrikx M., Meijer S., Van Der Velden J., Iosup A. Procedural Content Generation for Games: A Survey [Електронний ресурс] / M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup // ACM Transactions on Multimedia Computing, Communications, and Applications. – 2013. – Vol. 9, № 1. – С. 1-22. – Режим доступу: https://doi.org/10.1145/2422956.2422957 (дата звернення: 29.10.2025).

6. Rollings A., Adams E. Andrew Rollings and Ernest Adams on Game Design / A. Rollings, E. Adams. – [S. l.] : New Riders, 2003. – 576 с.

7. Yannakakis G. N., Togelius J. Artificial Intelligence and Games / G. N. Yannakakis, J. Togelius. – [S. l.] : Springer, 2018. – 364 с.

8. Hunicke R., LeBlanc M., Zubek R. MDA: A formal approach to game design and game research [Електронний ресурс] / R. Hunicke, M. LeBlanc, R. Zubek // Proceedings of the AAAI Workshop on Challenges in Game AI. – 2005. – Режим доступу: http://www.cs.northwestern.edu/~hunicke/MDA.pdf (дата звернення: 29.10.2025). – Назва з екрана.

9. Millington I., Funge J. Artificial Intelligence for Games / I. Millington, J. Funge. – [S. l.] : CRC Press, 2009. – 870 с.

10. Russell S. J., Norvig P. Artificial Intelligence: A Modern Approach / S. J. Russell, P. Norvig. – [S. l.] : Prentice Hall, 2010. – 1132 с.

11. Togelius J., Yannakakis G. N., Stanley K. O., Browne C. Search-Based Procedural Content Generation: A Taxonomy and Survey [Електронний ресурс] / J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne // IEEE Transactions on Computational Intelligence and AI in Games. – 2011. – Vol. 3. – С. 172-186. – Режим доступу: https://doi.org/10.1109/TCIAIG.2011.2148116 (дата звернення: 29.10.2025).

12. Shaker N., Togelius J., Nelson M. J. Procedural Content Generation in Games / N. Shaker, J. Togelius, M. J. Nelson. – [S. l.] : Springer, 2016. – 237 с.

13. Gamasutra Staff. (2019). Top Procedural Generation Techniques in Game Development. Gamasutra.

14. Ebert D. S., et al. Texturing & Modeling: A Procedural Approach / D. S. Ebert, et al. – [S. l.] : Morgan Kaufmann, 2002. – 688 с.

15. Summerville A. J., Snodgrass S., Guzdial M., et al. Procedural Content Generation via Machine Learning (PCGML) [Електронний ресурс] / A. J. Summerville, S. Snodgrass, M. Guzdial, et al. // IEEE Transactions on Games. – 2018. – Режим доступу: https://doi.org/10.1109/TG.2018.2846639 (дата звернення: 29.10.2025).

16. Buckland M. Programming Game AI by Example / M. Buckland. – [S. l.] : Wordware Publishing, Inc., 2005. – 495 с.

17. Schiemer L., Hauger D. Adaptive Difficulty Adjustment in Educational Games [Електронний ресурс] / L. Schiemer, D. Hauger // Proceedings of the 10th European Conference on Technology Enhanced Learning (EC-TEL). – 2015.

18. Petzold C. Programming in C: A Modern Approach / C. Petzold. – [S. l.] : Prentice Hall, 2018.

19. Karunarathne S. Unreal Engine C++ the Ultimate Developer's Handbook / S. Karunarathne. – [S. l.] : Independently published, 2020. – 346 с.

20. Schell J. The Art of Game Design: A Book of Lenses / J. Schell. – [S. l.] : CRC Press, 2014. – 600 с.

21. Adams E. Fundamentals of Game Design / E. Adams. – [S. l.] : New Riders, 2014. – 576 с.

22. Epic Games. Unreal Engine Documentation [Електронний ресурс] / Epic Games. – 2021. – Режим доступу: https://dev.epicgames.com/documentation/en-us/unreal-engine (дата звернення: 29.10.2025). – Назва з екрана.

23. Unity Technologies. Unity User Manual [Електронний ресурс] / Unity Technologies. – 2021. – Режим доступу: https://docs.unity3d.com/2021.3/Documentation/Manual/ (дата звернення: 29.10.2025). – Назва з екрана.

24. Crytek. CryEngine V Manual [Електронний ресурс] / Crytek. – 2021. – Режим доступу: https://docs.cryengine.com/ (дата звернення: 29.10.2025). – Назва з екрана.

25. Johnson D. Artificial Intelligence in Games: Impact on the Gaming Industry / D. Johnson. – [S. l.] : Springer, 2019.

26. Isla D. Handling Complexity in the Halo 2 AI [Електронний ресурс] / D. Isla // Game Developers Conference. – 2005.

27. Swink C. The AI director: Creating a dynamic narrative in Left 4 Dead [Електронний ресурс] / C. Swink // GDC Presentation. – 2007.

28. Shute V. J., Rahimi S. Assessment and adaptive feedback in game-based learning / V. J. Shute, S. Rahimi // In: Plass J. L., Mayer R. E., Homer B. D. (eds). Handbook of Game-Based Learning. – Cambridge (MA) : MIT Press, 2020. – 600 с.

29. Smith M., Mateas M. AI-Driven Game-Based Learning: A Review of Adaptive Systems [Електронний ресурс] / M. Smith, M. Mateas // IEEE Transactions on Computational Intelligence and AI in Games. – 2011.

30. Gregory J. Game Engine Architecture / J. Gregory. – [S. l.] : A K Peters/CRC Press, 2018. – 1240 с.

31. Gamma E., et al. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, et al. – [S. l.] : Addison-Wesley, 1994. – 416 с.

32. Weisfeld M. The Object-Oriented Thought Process / M. Weisfeld. – [S. l.] : Addison-Wesley Professional, 2015. – 336 с.

33. Sharma S. Mastering Unreal Engine 4.X / S. Sharma. – [S. l.] : Packt Publishing, 2019.

34. Vikram R. Unreal Engine 4 Game Development Essentials / R. Vikram. – [S. l.] : Packt Publishing, 2017. – 324 с.

35. Beck K. Test-Driven Development: By Example / K. Beck. – [S. l.] : Addison-Wesley Professional, 2003. – 240 с.

36. Meszaros G. xUnit Test Patterns: Refactoring Test Code / G. Meszaros. – [S. l.] : Addison-Wesley Professional, 2007. – 948 с.

37. Microsoft. Visual Studio Performance Tools [Електронний ресурс] / Microsoft. – 2021. – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/profiling/ (дата звернення: 29.10.2025). – Назва з екрана.

38. Thompson J. Game Development Documentation and Research Methods / J. Thompson. – [S. l.] : IGI Global, 2018. – 317 с.

39. Lewis J. R. Usability Testing / J. R. Lewis. – [S. l.] : John Wiley & Sons, 2014.

40. Field A. Discovering Statistics Using IBM SPSS Statistics / A. Field. – [S. l.] : Sage Publications, 2018. – 1104 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
03.10.2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської дипломної роботи
«Інтелектуальний генератор навчальних середовищ і сценаріїв для симуляторів у Unreal Engine. Частина 2. Навчальне середовище для тренування»

Науковий керівник: доцент к.т.н.
_____ Черняк О.І.

Студент групи 2КІ-24М
_____ Чорний В.В.

ВНТУ 2025

1 Підстава для виконання магістерської дипломної роботи (КМКР)

1.1 Зростає потреба у сучасних системах підготовки фахівців з використанням інтерактивних симуляцій, що забезпечують безпечне відпрацювання практичних навичок. Актуальним є застосування інтелектуальних методів для динамічної генерації навчальних сценаріїв відповідно до рівня підготовки користувача. Розробка такого модуля на базі Unreal Engine дозволить створювати адаптивне тривимірне навчальне середовище та підвищити ефективність навчального процесу.

1.2 Наказ про затвердження теми ККР.

2 Мета ККР і призначення розробки

2.1 Мета роботи — розробка архітектури, реалізація та експериментальна валідація програмного забезпечення навчального середовища для симуляторів на базі Unreal Engine, що функціонує за принципом інтелектуальної генерації адаптивних тренувальних сценаріїв.

2.2 Призначення розробки — даний програмний модуль призначений для створення адаптивного навчального середовища, що автоматично формує сценарії відповідно до рівня підготовки користувача. Рішення може бути використане в системах медичної, рятувальної чи військової підготовки для підвищення ефективності тренувального процесу.

3 Вихідні дані для виконання МКР

3.1 Аналіз сучасних технологій та підходів до інтелектуальної генерації навчальних сценаріїв та віртуальних середовищ.

3.2 Порівняння існуючих рішень та обґрунтування вибору методів, алгоритмів і технологій для реалізації генерації середовища.

3.3 Розробка структурної схеми та алгоритму роботи модуля інтелектуальної генерації навчального середовища в Unreal Engine (C++).

3.4 На основі схеми та алгоритму створено програмну реалізацію модуля, проведено тестування та оптимізацію.

3.5 Проведено тестування розробленого модуля в умовах різних навчальних сценаріїв.

4 Вимоги до виконання МКР

Головна вимога — програмний засіб повинен мати низький рівень помилок у виявленні шкідливих URL-посилань, важливо, щоб реакція програмного засобу на виявлення загроз була миттєвою для забезпечення максимального рівня захисту.

5 Перелік графічного матеріалу: UML-діаграма взаємодії компонентів інтелектуальної генерації

6 Етапи МКР та очікувані результати.

Етапи роботи та очікувані результати наведено у Таблиці А.1.

Таблиця А.1 — Етапи МКР

№	Назва та зміст етапу	Строк виконання етапів роботи	Очікуванні результаті
1.	Формулювання мети, задач, і вимог до ситеми	25.09.2025 – 27.09.2025	Аналітичний огляд літературних джерел, розділ 1 ПЗ
2.	Аналіз науково-технічних джерел і існуючих систем розпізнавання	28.09.2025 – 02.10.2025	розділ 2
3.	Розробка загальної архітектури та логічної структури системи	03.10.2025 – 06.10.2025	розділ 2
4.	Проєктування модулів збору даних, аналітики та візуалізації	07.10.2025 – 10.10.2025	Тези доповіді
5.	Розробка алгоритмів обробки зображень і виявлення об’єктів	11.10.2025 – 17.10.2025	розділ 3
6.	Реалізація програмної частини системи	18.10.2025 – 25.10.2025	розділ 4
7.	Інтеграція модулів і тестування на реальних/модельних даних	26.10.2025 – 31.10.2025	ПЗ, графіч. матеріал і презентація
8.	Розрахунок економічної частини	01.11.2025 – 03.11.2025	Розділ 5
9.	Оформлення пояснювальної записки та ілюстративного матеріалу	04.11.2025 – 09.11.2025	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 – «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2023.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

Протокол перевірки кваліфікаційної роботи

Назва роботи: Інтелектуальний генератор навчальних середовищ і сценаріїв для симуляторів у Unreal Engine. Частина 2. Навчальне середовище для тренування

Тип роботи: _____ магістерська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) ____1___ %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Азаров О.Д., д.т.н., проф., зав. кафедри ОТ _____
(прізвище, ініціали, посада) (підпис)

Мартинюк Т. Б., д.т.н., проф. кафедри ОТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____Захарченко С. М., проф. каф. ОТ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____Черняк О.І., доц. каф. ОТ____
(підпис) (прізвище, ініціали, посада)

Здобувач _____Чорний В.В._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

UML-діаграма взаємодії компонентів інтелектуальної генерації

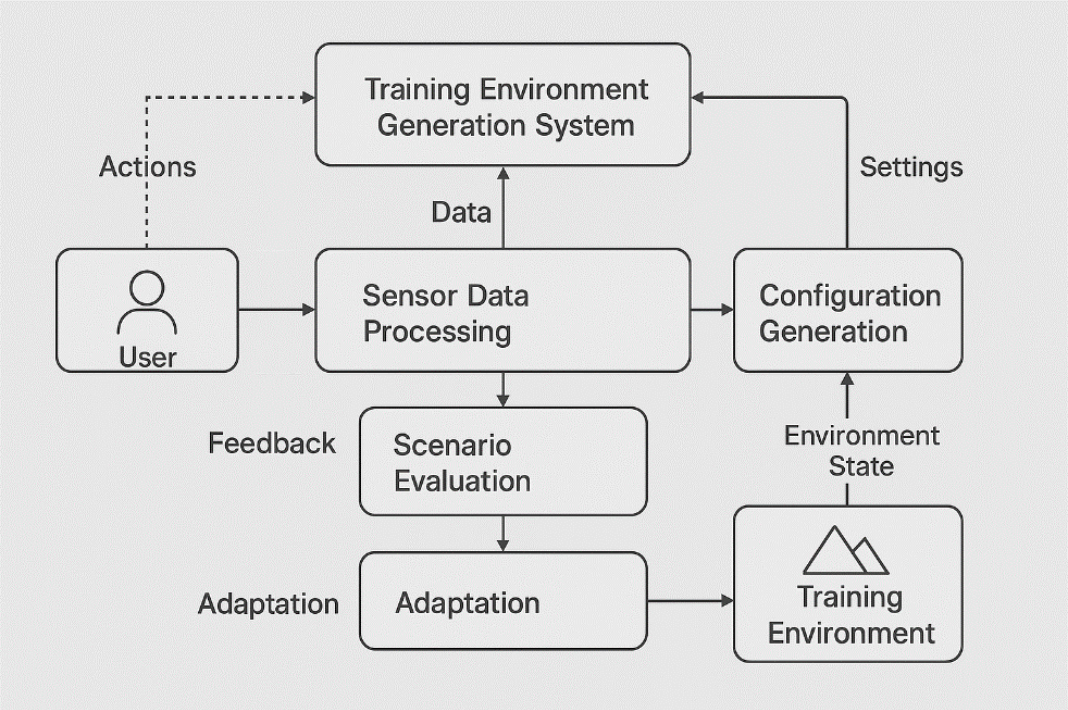

Рисунок В.1 — UML-діаграма взаємодії компонентів інтелектуальної генерації

ДОДАТОК Г

Лістинг методу Training Data Component

```
// TrainingDataComponent.h
#pragma once
#include "CoreMinimal.h"
#include "Components/ActorComponent.h"
#include "TrainingDataComponent.generated.h"
UENUM(BlueprintType)
enum class EMetricEventType : uint8
{
   TaskCompleted,
   CriticalError,
   TimeMetric,
   PathEfficiency
};
USTRUCT(BlueprintType)
struct FTrainingMetricData
{
   GENERATED_BODY()
   UPROPERTY(BlueprintReadOnly)
   float Timestamp = 0.0f;
   UPROPERTY(BlueprintReadOnly)
   EMetricEventType EventType;
   UPROPERTY(BlueprintReadOnly)
   float EventValue = 0.0f;
   UPROPERTY(BlueprintReadOnly)
   FVector Location = FVector::ZeroVector;
};
DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FOnMetricsReport, const FTrainingMetricData&, MetricData);
UCLASS(ClassGroup=(Custom), meta=(BlueprintSpawnableComponent))
class YOURPROJECT_API UTrainingDataComponent : public UActorComponent
{
   GENERATED_BODY()
public:
   UTrainingDataComponent();
   UPROPERTY(BlueprintAssignable, Category = "Metrics")
   FOnMetricsReport OnNewMetricLogged;
private:
   TArray<FTrainingMetricData> TrainingLog;
public:
```

```
    UFUNCTION(BlueprintCallable, Category = "Metrics")
    void LogNewMetric(EMetricEventType EventType, float Value, const FVector& Location);
    TArray<FTrainingMetricData> GetMetricsInTimeWindow(float DeltaTime) const;
protected:
    virtual void BeginPlay() override;
};

// TrainingDataComponent.cpp
#include "TrainingDataComponent.h"
#include "Engine/World.h"
#include "ScenarioObjectBase.h"
UTrainingDataComponent::UTrainingDataComponent()
{   PrimaryComponentTick.bCanEverTick = false;}
void UTrainingDataComponent::BeginPlay()
{   Super::BeginPlay();}
void UTrainingDataComponent::LogNewMetric(EMetricEventType EventType, float Value, const FVector& Location)
{   UWorld* World = GetWorld();
    if (!World) return;
    FTrainingMetricData NewData;
    NewData.Timestamp = World->GetTimeSeconds(); // Точний час події (3.3.1)
    NewData.EventType = EventType;
    NewData.EventValue = Value;
    NewData.Location = Location;
    TrainingLog.Add(NewData);
    OnNewMetricLogged.Broadcast(NewData);
    UE_LOG(LogTemp, Verbose, TEXT("Metric Logged: Type %d, Value %.2f"), (int32)EventType, Value);}
TArray<FTrainingMetricData> UTrainingDataComponent::GetMetricsInTimeWindow(float DeltaTime) const
{   TArray<FTrainingMetricData> MetricsInWindow;
    float CurrentTime = GetWorld()->GetTimeSeconds();
    float TimeThreshold = CurrentTime - DeltaTime;
    for (int32 i = TrainingLog.Num() - 1; i >= 0; --i)
    {       const FTrainingMetricData& Data = TrainingLog[i];
        if (Data.Timestamp >= TimeThreshold)
        {           MetricsInWindow.Add(Data);
        }
        else       {           break;
        }
    }
    return MetricsInWindow;}
```

ДОДАТОК Д

Лістинг методу Scenario Object Base

```
// ScenarioObjectBase.h
#pragma once
#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "ScenarioObjectBase.generated.h"
UCLASS(Abstract, Blueprintable)
class YOURPROJECT_API AScenarioObjectBase : public AActor
{
   GENERATED_BODY()
public:
   AScenarioObjectBase();
protected:
   UPROPERTY(EditDefaultsOnly, Category = "Scenario")
   float BaseDifficultyCost = 10.0f;
   UPROPERTY(VisibleAnywhere, Category = "Scenario")
   float CurrentDifficultyModifier = 1.0f;
public:
   UFUNCTION(BlueprintImplementableEvent, Category = "Scenario
Adaptation")
   void ApplyDifficultyModifier(float NewDifficultyValue);
   float GetBaseDifficultyCost() const { return BaseDifficultyCost; }
   UFUNCTION(BlueprintCallable, Category = "Scenario Events")
   void OnTaskOutcome(bool bSuccess);
};
```

```
// ScenarioObjectBase.cpp
#include "ScenarioObjectBase.h"
AScenarioObjectBase::AScenarioObjectBase()
{
   PrimaryActorTick.bCanEverTick = false;
}
void AScenarioObjectBase::ApplyDifficultyModifier_Implementation(float
NewDifficultyValue)
{
   CurrentDifficultyModifier = NewDifficultyValue;
   UE_LOG(LogTemp, Warning, TEXT("Actor %s received Difficulty Modifier:
%.2f"), *GetName(), NewDifficultyValue);
}
void AScenarioObjectBase::OnTaskOutcome(bool bSuccess)
{
   // UTrainingDataComponent* PlayerMetrics = ...
   // if (bSuccess)
   // {
   //     PlayerMetrics->LogNewMetric(EMetricEventType::TaskCompleted, 1.0f,
GetActorLocation());
   // }
   // else
   // {
   //         PlayerMetrics->LogNewMetric(EMetricEventType::CriticalError,  1.0f,
GetActorLocation());
   // }
}
```

ДОДАТОК Е

Лістинг методу Controllable NPC

```
#pragma once
#include "CoreMinimal.h"
#include "ScenarioObjectBase.h"
#include "GameFramework/Character.h"
#include "ControllableNPC.generated.h"
UCLASS()
class YOURPROJECT_API AControllableNPC : public AScenarioObjectBase
{
   GENERATED_BODY()
public:
   AControllableNPC();
protected:
   UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "NPC
Behavior")
   float BaseMovementSpeed = 300.0f;
   UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "NPC
Behavior")
   float CurrentAccuracy = 0.5f;
   UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "NPC
Behavior")
   float ReactionTime = 0.8f;
   // Reference to the Behavior Tree component (for AI logic)
   class UBehaviorTreeComponent* BehaviorTreeComponent;
public:
   virtual void ApplyDifficultyModifier_Implementation(float
NewDifficultyValue) override;

private:
   void UpdateBehaviorParameters(float DifficultyFactor);
};
// ControllableNPC.cpp
#include "ControllableNPC.h"
#include "GameFramework/CharacterMovementComponent.h"
#include "AIController.h"
#include "BehaviorTree/BehaviorTreeComponent.h"
AControllableNPC::AControllableNPC()
{
   UCharacterMovementComponent* MovementComp =
GetCharacterMovement();
   if (MovementComp)
```

```
        {
            MovementComp->MaxWalkSpeed = BaseMovementSpeed;
        }
        // BehaviorTreeComponent = FindComponentByClass<UBehaviorTreeComponent>();
    }
    void AControllableNPC::ApplyDifficultyModifier_Implementation(float NewDifficultyValue)
    {
        Super::ApplyDifficultyModifier_Implementation(NewDifficultyValue);
        UpdateBehaviorParameters(NewDifficultyValue);
    }
    void AControllableNPC::UpdateBehaviorParameters(float DifficultyFactor)
    {
        float NewSpeed = BaseMovementSpeed * FMath::Clamp(DifficultyFactor, 0.5f, 1.5f);
        if (UCharacterMovementComponent* MovementComp = GetCharacterMovement())
        {
            MovementComp->MaxWalkSpeed = NewSpeed;
        }
        CurrentAccuracy = FMath::Clamp(DifficultyFactor * 0.5f, 0.1f, 1.0f);
        ReactionTime = FMath::Lerp(1.5f, 0.2f, FMath::Clamp(DifficultyFactor, 0.0f, 2.0f) / 2.0f);
        // if (BehaviorTreeComponent)
        // {
        //     AAIController* AIController = Cast<AAIController>(GetController());
        //     if (AIController)
        //     {
        //         AIController->GetBlackboardComponent()->SetValueAsFloat(TEXT("AccuracyParameter"), CurrentAccuracy);
        //     }
        // }
        UE_LOG(LogTemp, Log, TEXT("NPC %s updated: Speed=%.0f, Accuracy=%.2f, Reaction=%.2f"),
            *GetName(), NewSpeed, CurrentAccuracy, ReactionTime);
    }
```