

**Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка
Інженерно-педагогічний факультет
Кафедра комп'ютерних технологій**

Кваліфікаційна робота

**МЕТОДИКА НАВЧАННЯ СТУДЕНТІВ ФАХОВИХ КОЛЕДЖІВ
РОЗРОБКИ ІГРОВИХ ДОДАТКІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО
ІНТЕЛЕКТУ**

**Спеціальність: 015 Професійна освіта;
спеціалізація: 015.39 Цифрові технології
Освітньо-професійна програма
«Професійна освіта (Комп'ютерні технології)»**

ВИКОНАВ:
здобувач вищої освіти
освітнього рівня «магістр»
Антон СЛОБОДЯНЮК

НАУКОВИЙ КЕРІВНИК:
кандидат педагогічних наук, доцент
Сергій КОЗІБРОДА

РЕЦЕНЗЕНТ:
кандидат технічних наук, доцент
кафедри машинознавства
та транспорту ТНПУ ім. В. Гнатюка
Назар БУРЕГА

Робота захищена з оцінкою:
Національна шкала _____
Кількість балів: _____ Оцінка: ECTS _____

Тернопіль 2025

**Тернопільський національний педагогічний університет
імені Володимира Гнатюка
Інженерно-педагогічний факультет
Кафедра комп'ютерних технологій**

**ЗАВДАННЯ
ДЛЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ
Антону СЛОБОДЯНЮКУ**

на тему «Методика навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту»

Спеціальність: 015 Професійна освіта, спеціалізація: 015.39 Цифрові технології

Освітня програма: Професійна освіта (Комп'ютерні технології)

НАУКОВИЙ КЕРІВНИК: канд. пед. наук, доцент Сергій КОЗІБРОДА

Термін подання студентом на кафедру роботи і супроводжувальних документів: до 05 грудня 2025 р.

Зміст (перелік основних питань, які потрібно розкрити):

1. Проаналізувати можливості сучасних ігрових рушіїв та інструментів для створення ігрових додатків з елементами штучного інтелекту та обґрунтувати доцільність розробки методики їх створення засобами Unity.
2. Розкрити технологію використання середовища Unity та елементів штучного інтелекту у процесі навчання студентів фахових коледжів у процесі розробки ігрових додатків та розробити методику їх проектування.
3. Експериментально перевірити ефективність методики навчання студентів фахових коледжів розробці ігрових додатків з елементами штучного інтелекту на основі використання Unity.

Перелік додаткових матеріалізованих результатів роботи (моделей, виробів, ілюстрацій та ін.): методики навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту, діаграма контрольного зрізу знань студентів ЕГ і КГ.

Графік підготовки магістерської роботи до захисту

№пп	Перелік роботи	Термін виконання, I рік навчання	Термін виконання, II рік навчання	Відмітка наукового керівника
1.	Вибір теми, затвердження на засіданні кафедри та вибір керівника	Жовтень-Листопад		
2.	Складання плану роботи і графіку її підготовки, узгодження з керівником	Листопад		
3.	Систематизація аналітичного матеріалу у вигляді розділу роботи. Розробка методики експерименту.	Грудень-Квітень		
4.	Проміжний контроль.	Під час практики		
5.	Написання кваліфікаційної роботи відповідно до складеного змісту роботи, ознайомлення керівника з її першим варіантом	Під час практики	Під час практики	
6.	Корегування роботи, оформлення її згідно вимог, подання на рецензію.		Листопад	
7.	Попередній захист роботи		Листопад	
8.	Подання кваліфікаційної роботи та супроводжуючих документів на кафедру.		Грудень	
9.	Публічний захист магістерської роботи.		Грудень	

Графік узгоджено: 24.10.2024 р.

Науковий керівник _____ Сергій КОЗІБРОДА Магістрант _____ Антон Слободянюк

АНОТАЦІЯ

Слободянюк А. О. Методика навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту. Магістерська робота за спеціальністю 015.39 Професійна освіта. Цифрові технології. Тернопільський національний педагогічний університет імені Володимира Гнатюка. Тернопіль, 2025. 62 с.

Магістерська робота присвячена розробці та експериментальній перевірці ефективності методики навчання студентів фахових коледжів розробці ігрових додатків з елементами штучного інтелекту (ШІ). Актуальність дослідження зумовлена високим попитом на фахівців, які володіють сучасними інструментами створення інтерактивного контенту, зокрема, навичками програмування ШІ для ігрової індустрії.

У роботі було проаналізовано можливості провідних сучасних ігрових рушіїв, мов програмування та інструментів, що застосовуються для розробки ігрових додатків. На основі цього аналізу обґрунтовано доцільність використання ігрового рушія Unity та мови програмування C# як найбільш ефективних та поширених засобів для реалізації поставлених освітніх завдань.

Розкрито технологію використання середовища Unity, мови C# та основних елементів штучного інтелекту (наприклад, алгоритмів пошуку шляху, поведінкових дерев, кінцевих автоматів) у процесі професійної підготовки студентів.

Ключові слова: Unity, C#, ігрові додатки, штучний інтелект (ШІ), методика навчання, фаховий коледж, професійна підготовка, проєктування ігор.

SUMMARY

Slobodyanyuk A. O. Methodology for Teaching Vocational College Students to Develop Game Applications with Artificial Intelligence Elements. Master's Thesis in Specialty 015.39 Professional Education. Digital Technologies. Volodymyr Hnatyuk Ternopil National Pedagogical University. Ternopil, 2025. 62 p.

The Master's Thesis is dedicated to the development and experimental verification of the effectiveness of the methodology for teaching vocational college students to develop game applications with elements of Artificial Intelligence (AI). The relevance of the study is driven by the high demand for specialists who possess modern tools for creating interactive content, particularly skills in AI programming for the gaming industry.

The paper analyzed the capabilities of leading modern game engines, programming languages, and tools used for developing game applications. Based on this analysis, the feasibility of using the Unity game engine and the C# programming language was substantiated as the most effective and widely used means for achieving the stated educational objectives.

The technology for utilizing the Unity environment, the C# language, and fundamental elements of artificial intelligence (such as pathfinding algorithms, behavior trees, finite-state machines) in the professional training process of students was revealed. The developed methodology is based on competence-based and project-based approaches, involving practical tasks closely resembling real-world development conditions.

Key words: Unity, C#, game applications, Artificial Intelligence (AI), teaching methodology, vocational college, professional training, game design.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ	7
ВСТУП.....	8
I. ХАРАКТЕРИСТИКА СУЧАСНИХ ТЕХНОЛОГІЙ РОЗРОБКИ ІГРОВИХ ДОДАТКІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ.....	12
1.1. Огляд та вибір кросплатформенного середовища розробки.....	12
1.2. Використання процедурного та статичного генерування дизайну ігрових додатків у навчанні студентів фахових коледжів.	17
1.3. Аналіз алгоритмів на основі штучного інтелекту для інтеграції у ігрові додатки.....	20
Висновки до першого розділу	26
II. МЕТОДИЧНІ АСПЕКТИ НАВЧАННЯ СТУДЕНТІВ ФАХОВИХ КОЛЕДЖІВ РОЗРОБКИ ІГРОВИХ ДОДАТКІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ	28
2.1.Методика побудови середовища ігрового додатку з елементами штучного інтелекту	28
2.2. Методика реалізації моделей штучного інтелекту ворожих об'єктів	34
2.3. Методика навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту.....	41
Висновки до другого розділу.....	47
III. ВИЗНАЧЕННЯ ЕКФЕКТИВНОСТІ МЕТОДИКИ НАВЧАННЯ СТУДЕНТІВ ФАХОВИХ КОЛЕДЖІВ РОЗРОБКИ ІГРОВИХ ДОДАТКІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ	49
3.1. Організація та проведення експерименту	49
3.2. Оцінка ефективності запропонованої розробки	50
Висновки до третього розділу	57
ВИСНОВКИ.....	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

- ШІ – Штучний Інтелект.
- ООП – об’єктно-орієнтоване програмування
- ЕГ – Експериментальна Група
- КГ – Контрольна Група
- ФПО – Фахова Передвища Освіта
- ШНМ – Штучна Нейронна Мережа
- PCG – Procedural Content Generation (Процедурна Генерація Контенту)
- NPC – Non-Player Character (Неігровий Персонаж)
- NavMesh – Navigation Mesh (Навігаційна Сітка)
- FSM – Finite State Machine (Скінченний Автомат)
- GA – Genetic Algorithm (Генетичний Алгоритм)
- ML-Agents – Machine Learning Agents (Інструментарій Машинного Навчання)
- RL – Reinforcement Learning (Навчання з Підкріпленням)
- FPS – Frames Per Second (Кадрів за секунду)
- C# – C Sharp (Об’єктно-орієнтована мова програмування)
- API – Application Programming Interface (Прикладний Програмний Інтерфейс)
- IDE – Integrated Development Environment (Інтегроване Середовище Розробки)
- SDK – Software Development Kit (Комплект для Розробки Програмного Забезпечення)

ВСТУП

У сучасному світі цифрових технологій, де інновації визначають динаміку розвитку суспільства, ігрова індустрія (GameDev) посідає одне з ключових місць, демонструючи стрімке зростання та економічний потенціал. Зростає попит на висококваліфікованих фахівців, здатних створювати складні, інтерактивні та захоплюючі ігрові додатки. Критичним компонентом, що визначає якість та реалістичність сучасних ігор, є штучний інтелект (ШІ), який відповідає за поведінку неігрових персонажів, логіку ігрового світу та адаптацію складності.

Одними з провідних та найбільш затребуваних інструментів для розробки ігрових додатків є мультиплатформовий ігровий рушій Unity та мова програмування C#. Unity є основою для створення широкого спектра 2D/3D ігор та симуляцій, а C# забезпечує потужні можливості для реалізації складної ігрової логіки, включно з розробкою та інтеграцією елементів ШІ.

Підготовка майбутніх фахівців ігрової індустрії та цифрових технологій у фахових коледжах передбачає не лише освоєння сучасних інструментів, але й формування здатності розробляти та впроваджувати ефективні цифрові рішення, особливо ті, що вимагають розуміння принципів роботи ШІ. У цьому контексті методика навчання розробки ігрових додатків з елементами штучного інтелекту стає одним із ключових елементів професійної підготовки. Вона дозволяє студентам здобути практичні навички роботи з Unity/C# та розвинути інженерний підхід до вирішення задач моделювання поведінки в ігровому середовищі.

Дослідження спрямоване на розробку та вдосконалення методики, яка базується на використанні Unity та C# для навчання створенню ігрових додатків із інтегрованими алгоритмами ШІ. Це дозволить майбутнім фахівцям не лише оволодіти технічними навичками програмування, а й інтегрувати передові рішення у свою професійну діяльність для створення конкурентоспроможного ігрового контенту.

Актуальність теми полягає в тому, що в умовах глобального стрімкого розвитку цифрових технологій, зокрема в галузі ігрової індустрії (GameDev), штучного інтелекту (ШІ) та комп'ютерної графіки, суттєво зростає необхідність у фахівцях, здатних створювати високоякісні, інтерактивні ігрові додатки з елементами складних алгоритмів AI. Ігрові додатки, що інтегрують ШІ, набувають широкого застосування не лише в індустрії розваг, а й у критично важливих сферах: освіті, медицині, інженерному моделюванні та віртуальних тренажерах. Створення таких програмних продуктів вимагає не лише глибокого розуміння технічних аспектів програмування, а й практичних навичок використання сучасних ігрових рушіїв, зокрема Unity та мови C#, для реалізації AI-алгоритмів.

Зі зростанням попиту на молодших фахівців, здатних створювати сучасні програмні продукти з використанням AI-технологій, актуальним є впровадження інноваційних методик навчання у систему фахової передвищої освіти, які відповідають вимогам сучасної IT-індустрії. Фахові коледжі відіграють важливу роль у формуванні практично орієнтованих компетентностей, однак традиційні підходи до навчання програмування часто не враховують новітні тенденції розвитку ШІ та його інтеграції в ігрові середовища.

Особливого значення ця тема набуває для підготовки студентів фахових коледжів, адже сучасні умови вимагають від них здатності ефективно використовувати цифрові інструменти та базові моделі штучного інтелекту для вирішення прикладних задач моделювання поведінки. Відсутність систематизованої та експериментально перевіреної методики навчання розробки ігрових додатків з елементами AI ускладнює підготовку студентів до реальних виробничих завдань та знижує їх конкурентоспроможність на ринку праці.

Таким чином, тема магістерської роботи є актуальною, оскільки вона спрямована на вдосконалення методики підготовки майбутніх молодших фахівців галузі цифрових технологій відповідно до сучасних вимог суспільства

й ринку праці, забезпечуючи їх знаннями та навичками роботи з передовими технологіями Unity, C# та штучним інтелектом.

Об'єктом дослідження є процес підготовки студентів фахових коледжів до розробки ігрових додатків з елементами штучного інтелекту.

Предмет дослідження – технологія навчання студентів фахових коледжів розробки ігрових додатків з використанням інструментів Unity та алгоритмів штучного інтелекту.

Метою даної роботи є обґрунтування ефективності застосування методики навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту у процесі їх професійної підготовки.

Виходячи з мети дослідження було поставлено наступні **завдання**:

1. Проаналізувати можливості сучасних ігрових рушіїв та інструментів для створення ігрових додатків з елементами штучного інтелекту та обґрунтувати доцільність розробки методики їх створення засобами Unity.
2. Розкрити технологію використання середовища Unity та елементів штучного інтелекту у процесі навчання студентів фахових коледжів у процесі розробки ігрових додатків та розробити методику їх проєктування.
3. Експериментально перевірити ефективність методики навчання студентів фахових коледжів розробці ігрових додатків з елементами штучного інтелекту на основі використання Unity.

Методи дослідження: *теоретичні* (аналіз, узагальнення літературних джерел і проєктування навчально-методичного курсу); *емпіричні* (педагогічне спостереження, опитування, анкетування).

Наукова новизна дослідження: отримала подальший розвиток методика навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту, що сприяє формуванню професійних компетентностей у сфері розробки сучасних інтерактивних рішень із застосуванням Unity для моделювання поведінки об'єктів (ІІІ) відповідно до актуальних вимог ринку праці та освітніх тенденцій.

Практичне значення одержаних результатів: розроблено методичний супровід навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту, що включає навчальні матеріали, практичні завдання та кейсові проєкти для формування навичок розробки програмних рішень на основі Unity з інтеграцією базових алгоритмів штучного інтелекту.

I. ХАРАКТЕРИСТИКА СУЧАСНИХ ТЕХНОЛОГІЙ РОЗРОБКИ ІГРОВИХ ДОДАТКІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

1.1. Огляд та вибір кросплатформенного середовища розробки.

Стрімкий розвиток індустрії розробки ігрових додатків (GameDev) висуває високі вимоги до інструментарію, який використовується для створення якісного, високопродуктивного та кросплатформенного інтерактивного контенту. Вибір оптимального кросплатформенного середовища розробки (ігрового рушія) є фундаментальним етапом проектування, оскільки він визначає архітектуру, функціональні можливості, швидкість розробки, а також спектр кінцевих платформ (ПК, консолі, мобільні пристрої), на яких може бути розгорнутий ігровий додаток. У контексті підготовки студентів фахових коледжів, цей вибір повинен також враховувати поширеність технології на ринку праці та її доступність для освітнього процесу..

Концептуальна ідея ігрового рушія полягає в абстрагуванні та централізації виконання спільних, ресурсомістких ігрових задач, таких як графічний рендеринг, фізичні симуляції, обробка вводу даних, управління пам'яттю та, що особливо важливо для нашої теми, реалізація базового штучного інтелекту . Така уніфікація дозволяє розробникам, особливо початківцям, зосередитися безпосередньо на унікальній ігровій механіці, сюжеті та інтеграції складних поведінкових алгоритмів, які забезпечують оригінальність кінцевого продукту.

Ігровий рушій являє собою комплексне прикладне програмне забезпечення, яке обов'язково включає: фізичний рушій, візуалізатор (для 2D/3D графіки), звуковий рушій, систему анімації, підтримку скриптингу (програмування ігрової логіки) та, звичайно, модулі для реалізації штучного інтелекту (AI). Саме функції, реалізовані програмістами в скриптах, визначають кінцевий ігровий процес та інтерактивність.

На сучасному ринку лідерами серед кросплатформенних рушіїв є Unity та Unreal Engine (рис. 1.1). Їхня архітектура рендерингу дозволяє створювати високоякісні візуальні ефекти, які при цьому можуть ефективно працювати навіть на пристроях з невисокою продуктивністю, що важливо для розробки додатків для фахових коледжів.

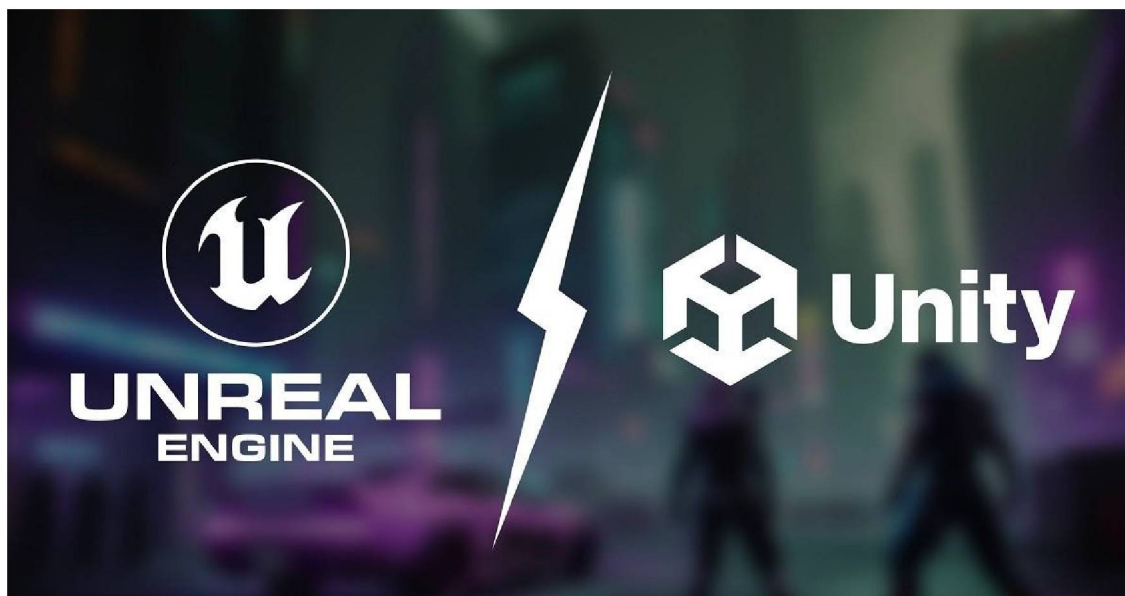


Рис. 1.1 – Лідери серед кросплатформенних рушіїв Unity та Unreal Engine

З точки зору доступності, Unity та Unreal Engine пропонують вигідні моделі ліцензування. Обидва рушії доступні відносно безкоштовно для навчання та некомерційної розробки. Unity надає безкоштовну версію, яка вимагає відображення логотипу при запуску гри, а комерційні відрахування (близько 10% від прибутку) починаються лише після досягнення значного доходу (зазвичай, понад \$100 тис. на рік). Unreal Engine 4 також є безоплатним до досягнення доходу, але вимагає 5% роялті після перевищення певного порогу прибутковості (близько \$3 тис. на квартал).

Ключовим фактором при виборі рушія для освітнього процесу є екосистема інструментів. Обидва рушії мають власні крамниці асетів (Asset Stores/Marketplaces), де розробники можуть знайти готові 3D-моделі, текстури, звуки, системи частинок та, що важливо для нашої теми, готові AI-скрипти та

інструменти. При цьому Unity Asset Store часто відзначається більшою кількістю доступних асетів та розширеною документацією, що робить цей рушій більш зручним та доступним для новачків і студентів.

Нарешті, вирішальною відмінністю для навчання є мова програмування. Unreal Engine 4 використовує мову C++, яка має складніший поріг входження. Натомість, Unity використовує мову C# (C Sharp), яка є об'єктно-орієнтованою, має більш зрозумілий синтаксис та краще підходить для швидкого засвоєння студентами фахових коледжів основ ігрової логіки та програмування елементів ІІІ.

Unity є потужним кросплатформенним середовищем розробки, що дає можливість створювати та розгортати ігрові додатки для більш ніж 20 різних платформ (від мобільних пристроїв та ПК до консолей). Це забезпечує студентам фахових коледжів універсальність навичок, що є високою вимогою на сучасному ринку праці.

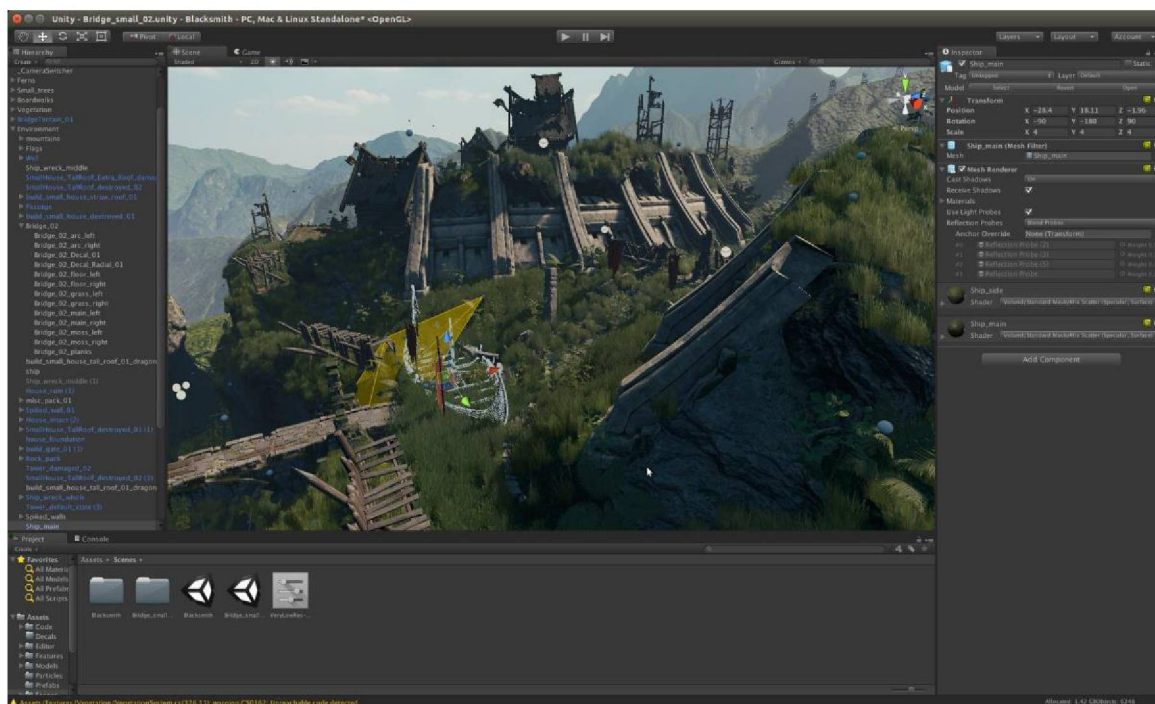


Рис. 1.2 – Редактор Unity

Ключовою перевагою Unity для освітнього процесу є його інтуїтивно зрозуміле візуальне середовище розробки (Drag & Drop інтерфейс). Редактор

Unity (рис. 1.2), який легко налаштовується, складається з різних вікон (Scene, Game, Inspector, Project), що дозволяє здійснювати налаштування ігрових об'єктів, логіки та параметрів ІІІ прямо в редакторі без потреби постійного перемикавання між програмами. Це значно спрощує процес прототипування та тестування для студентів, особливо на початкових етапах засвоєння матеріалу.

Дві основні переваги Unity, що вирізняють його в контексті навчання:

1. **Наявність візуального середовища розробки:** Окрім інструментарію візуального моделювання, Unity пропонує інтегровану лінію складання (Build Pipeline), що суттєво підвищує продуктивність розробників. Для 2D-розробки доступні спеціалізовані інструменти (Sprite Creator, Sprite Editor), а також окремий фізичний рушій, що дозволяє швидко опанувати основи ігрової механіки.

2. **Кросплатформенна підтримка:** Можливість однією кодовою базою створювати продукти для різних платформ критично важлива для формування професійної компетентності.

Крім того, екосистема Unity сприяє навчанню завдяки Unity Asset Store. Тут доступні такі візуальні інструменти, як Playmaker, які дозволяють студентам навіть без глибокого знання програмування (на етапі вивчення логіки) проектувати власні ігрові механіки та поведінку AI, використовуючи скінченні автомати та логічні блоки. Це дозволяє розділити процес вивчення гейм-дизайну та логіки ІІІ від вивчення суто синтаксису C#.

Наявність безкоштовної версії без явних функціональних обмежень для освітніх цілей, а також широка база готових асетів, робить Unity унікальним і оптимальним середовищем для підготовки розробників ігрових додатків з елементами штучного інтелекту у фахових коледжах.

Вибір Unity як освітнього середовища обумовлений його широким набором інтегрованих інструментів, які орієнтовані як на швидке прототипування, так і на оптимізацію кінцевого продукту. Для цілей навчання

студентів фахових коледжів розробці ігрових додатків з ІІІ, особливо важливими є такі функціональні можливості:

- *Редактор "Все-в-одному" та Візуальне Середовище:* Редактор, доступний на Windows, Linux і Mac, включає практичний інструментарій для візуального проєктування ігрових середовищ. Це дозволяє студентам легко налаштовувати сцени, швидко створювати ігрову логіку та експериментувати з високопродуктивним геймплеєм, що є ідеальним для освоєння основ.

- *Ефективні Робочі Процеси (Prefabs):* Використання Unity Prefabs (заздалегідь налаштованих ігрових об'єктів) дозволяє студентам працювати впевнено, використовуючи шаблони для створення багаторазових елементів (наприклад, ворогів або інтерактивних об'єктів). Це значно спрощує навчання об'єктно-орієнтованому підходу та масштабуванню проєкту.

- *Вбудована Навігаційна Система для ІІІ (AI Pathfinding Tools):* Unity надає потужну навігаційну систему (NavMesh System), призначену для створення неігрових персонажів (NPC), які можуть розумно переміщатися всередині ігрового простору. Система використовує навігаційні сітки, які автоматично генеруються з геометрії сцени. Це є фундаментальним елементом для навчання студентів основним алгоритмам пошуку шляху та моделюванню інтелектуальної поведінки.

- *Розширюваність та Спеціальні Інструменти:* Редактор Unity може бути розширений за допомогою спеціальних інструментів, що відповідають вимогам робочого процесу. Студенти можуть створювати власні розширення (для оптимізації роботи з ІІІ) або використовувати тисячі ресурсів і розширень з Asset Store, що прискорює навчальні проєкти. Це розвиває навички адаптації та використання готових рішень.

- *Інструменти Вдосконаленого Профілювання:* Наявність функцій профілювання (Performance Profiler) дозволяє студентам постійно оптимізувати вміст гри, особливо логіку ІІІ, у процесі розробки. Це важливий елемент для

підготовки фахівців, здатних створювати не лише функціональні, а й високопродуктивні додатки.

- *Підтримка 2D & 3D Розробки:* Unity підтримує обидва напрямки, що дозволяє методиці навчання охоплювати широкий спектр ігрових жанрів та відповідати різноманітним інтересам студентів.
- *Командна Співпраця (Unity Teams):* Інструменти для командної роботи (навіть у хмарному форматі) дозволяють студентам ефективно співпрацювати над спільними проєктами, що є критично важливим для імітації реального виробничого процесу.

Таким чином, Unity є не просто інструментом розробки, а повноцінною освітньою платформою, що забезпечує студентів фахових коледжів усіма необхідними засобами для засвоєння теорії та практики розробки ігрових додатків, зокрема, у сфері штучного інтелекту.

1.2. Використання процедурного та статичного генерування дизайну ігрових додатків у навчанні студентів фахових коледжів.

Ефективна розробка сучасних ігрових додатків, особливо великомасштабних та тих, що інтегрують елементи штучного інтелекту, вимагає від фахівців володіння різними підходами до створення ігрового контенту та дизайну рівнів. Існують два основні методи формування ігрового простору: ручне (або статичне) проєктування та процедурне генерування (Procedural Content Generation, PCG).

Ручний дизайн передбачає деталізоване, індивідуальне створення кожного елемента сцени гейм-дизайнером, забезпечуючи високу якість, художню цілісність та точний контроль над ігровим досвідом. Натомість, процедурне генерування використовує алгоритми для автоматичного створення великого, різноманітного та унікального контенту (ландшафтів, рівнів, об'єктів) на основі заданих правил та параметрів.

У контексті підготовки студентів фахових коледжів, важливо навчити їх ефективно комбінувати ці підходи. Розуміння PCG, яке часто ґрунтується на алгоритмах випадковості та логіці, тісно пов'язане з принципами функціонування штучного інтелекту (наприклад, для генерації завдань або адаптації рівня).

Ефективне проєктування ігрових світів базується на усвідомленому виборі між ручним дизайном та процедурним генеруванням (PCG). Обидва підходи мають свої переваги та недоліки (таблиця 1.1-1.4), розуміння яких є критично важливим для фахівців, яких готують фахові коледжі.

Таблиця 1.1 – Переваги процедурного генерування (PCG) для навчання

№	Перевага	Значення для навчання та ІІІ
1.	Легке масштабування та обсяги контенту.	Дозволяє створювати карти/проєкти, значно більші, ніж ті, що можна було б створити вручну. Це навчає студентів роботі з великими даними та ефективному використанню систем управління ресурсами.
2.	Зменшення залежності від постійної пам'яті.	Створення фрагментів ландшафту "на льоту" дозволяє уникнути необхідності зберігати кожен карту в пам'яті пристрою. Це розвиває навички оптимізації та динамічного завантаження контенту.
3.	Вища варіативність та реґрабельність.	Різноманітність і непередбачуваність згенерованих карт підвищує зацікавленість користувача. Це є важливим фактором для ІІІ, оскільки вимагає від NPC (неігрових персонажів) адаптивної поведінки в умовах постійно мінливого середовища.
4.	Гнучкість у внесенні глобальних змін.	Широко поширені зміни в місцевості досягаються зміною лише кількох параметрів, на відміну від трудомісткого ручного редагування.

Хоча процедурне генерування пропонує значні переваги у масштабованості та варіативності, воно не позбавлене технічних та логічних складностей, які необхідно враховувати в навчальному процесі.

Таблиця 1.2 – Недоліки процедурного генерування (PCG)

№	Недолік	Значення для навчання та ІІІ
1.	Висока початкова складність.	Вимагає значних зусиль для гарантування "пристойного вигляду" та функціональності згенерованого ландшафту. Це підвищує вимоги до навичок алгоритмічного мислення студентів.
2.	Проблеми гарантією геймплею.	Без ретельного тестування існує ризик генерації рівнів, де ігровий процес повністю переривається (наприклад, непрохідні ділянки). Це підкреслює необхідність інтеграції верифікаційних алгоритмів ІІІ в процес генерації.
3.	Складність розміщення дрібних деталей.	Розміщення предметів та "ручної" логіки потребує обережного підходу, хоча меншою мірою стосується ігор із видом зверху.

На противагу автоматизованим методам, статичний дизайн зберігає свою актуальність, особливо для освоєння студентами основ гейм-дизайну та забезпечення контрольованого наративного досвіду.

Таблиця 1.3 – Переваги ручного дизайну для навчання

№	Перевага	Значення для навчання
1.	Гарантований геймплей.	Розробник може бути впевнений, що ігровий процес функціонує відповідно до плану на кожному рівні. Це ідеально для вивчення основних принципів гейм-дизайну та наративу.

2.	Просте розміщення дрібних деталей.	Створення та розміщення унікальних художніх або логічних елементів значно простіше.
3.	Точний контроль над просуванням гравця.	Легше направляти гравця "лінійним шляхом" і створювати керований ігровий досвід.

Таблиця 1.4 – Переваги ручного дизайну для навчання

№	Недолік	Значення для навчання
1.	Трудомісткість і час.	Широко поширені зміни в місцевості займають багато часу.
2.	Вимоги до пам'яті.	Необхідність зберігати кожну карту в пам'яті збільшує постійні вимоги до ресурсів, що для великих ігор є неприйнятним.
3.	Потреба у додаткових інструментах.	Може знадобитися створення спеціального редактора карт і системи завантаження файлів.

Таким чином, у процесі навчання студентів фахових коледжів необхідно забезпечити баланс, вивчаючи ручний дизайн для освоєння основ гейм-дизайну та застосовуючи PCG для розвитку алгоритмічних навичок та навичок роботи з великими ігровими світами, що є критичним для розуміння принципів роботи AI.

1.3. Аналіз алгоритмів на основі штучного інтелекту для інтеграції у ігрові додатки

Штучний інтелект (ШІ) є наріжним каменем сучасної ігрової індустрії. Саме ШІ відповідає за створення динамічного, реалістичного та захоплюючого ігрового досвіду, моделюючи поведінку неігрових персонажів (NPC), управління ресурсами, адаптацію складності та процедурну генерацію контенту. Для ефективної підготовки студентів фахових коледжів необхідно забезпечити

глибоке розуміння основних алгоритмів ШІ, які використовуються в комерційній розробці.

Одним із найбільш ефективних і гнучких алгоритмів PCG є **Алгоритм колапсу хвильової функції (WFC)**. Цей метод дозволяє генерувати текстури та ігрові світи, які є одночасно випадковими та відповідають чітким, визначеним правилам (наприклад, "ліс не може примикати до води, а дорога має бути продовжена").

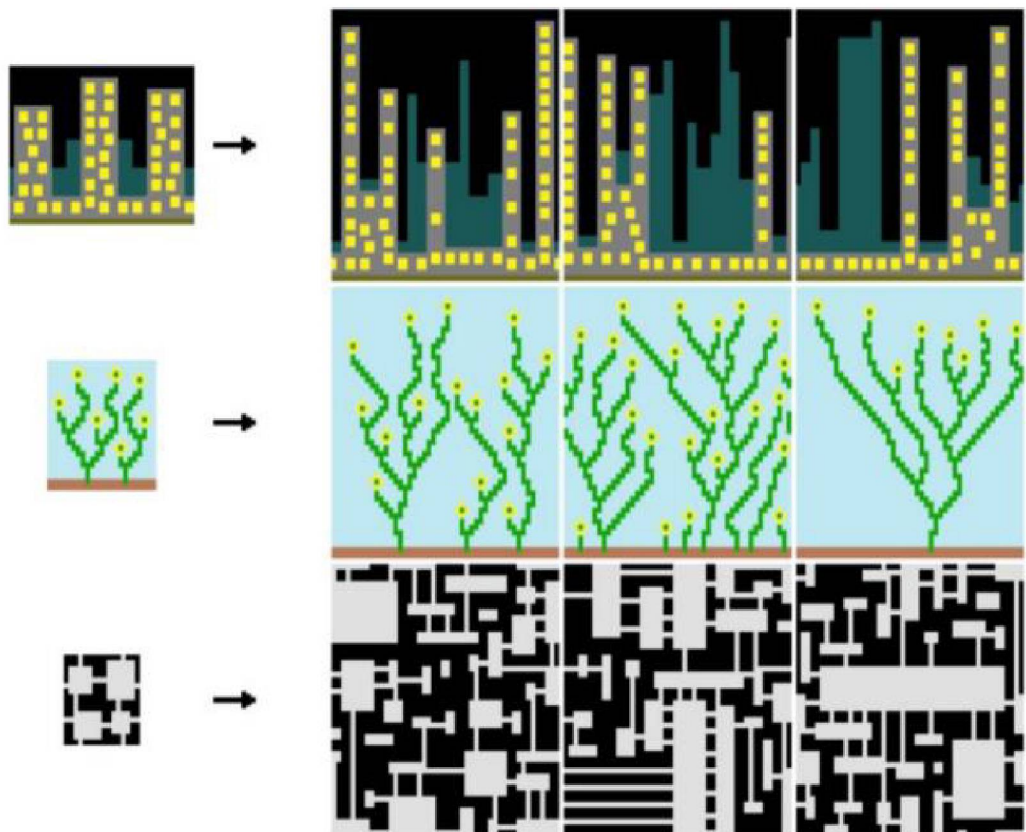


Рис. 1.3 – Алгоритм колапсу хвильової функції (WFC).

Принцип роботи WFC в контексті ігрового світу:

1. *Суперпозиція (Непостережуваний стан):* Спочатку весь масив клітинок (осередків) ігрового світу розглядається як "хвильова функція" у непоміченному стані. Це означає, що кожній клітинці відповідає множина модулів (плиток, елементів ландшафту), які потенційно можуть у ній

знаходитися. У термінах WFC, осередок перебуває у суперпозиції всіх можливих модулів.

2. *Колапс (Вибір)*: Процес генерації починається з "колапсу" (з'єднання) осередків, одного за одним. Для кожного осередку випадковим чином вибирається лише один модуль із усіх можливих.

3. *Поширення Обмежень (Constraint Propagation)*: Після вибору модуля настає найважливіший і найбільш ресурсовитратний етап. Для кожного обраного модуля заздалегідь визначена підмножина модулів, яким дозволено бути суміжними з ним. При кожному "колапсі" система оновлює підмножини можливих модулів у сусідніх (суміжних) осередках, відсіюючи ті варіанти, які суперечать правилам. Це гарантує логічну узгодженість згенерованого світу.

4. *Вибір наступного осередку (Ентропія)*: Ключовий аспект алгоритму полягає у виборі наступного осередку для "колапсу". Алгоритм завжди вибирає осередок із найменшою ентропією. Ентропія тут означає мінімальну кількість варіантів вибору (найменшу хаотичність). Цей принцип гарантує, що найбільш "обмежені" зони вирішуються першими, запобігаючи логічним конфліктам і підвищуючи ймовірність успішної генерації.

Навчання WFC у Unity та C# є високоцінним для студентів, оскільки воно не лише демонструє потужність алгоритмічного дизайну, а й слугує містком до складніших концепцій Штучного Інтелекту та Алгоритмічного мислення.

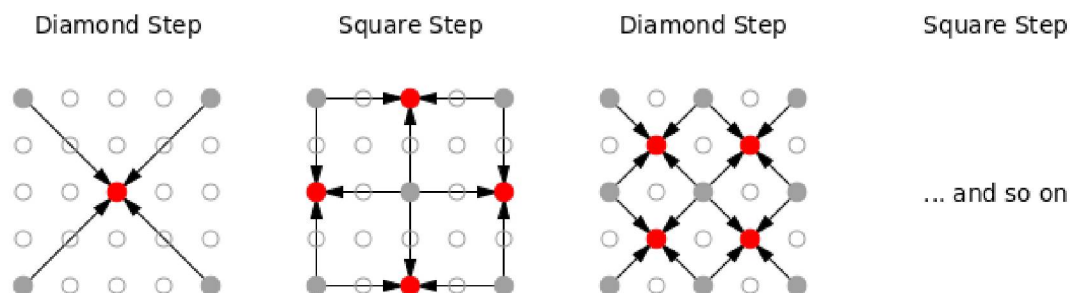


Рис. 1.4 – Алгоритм Diamond-Square (Діамент-Квадрат).

Для навчання студентів створенню відкритих, реалістичних ігрових світів одним із фундаментальних алгоритмів процедурної генерації є **Алгоритм Diamond-Square (Діамент-Квадрат)**. Цей метод широко використовується для ефективного створення карт висот (heightmaps) — цифрових моделей рельєфу, які є основою для генерації гір, долин та іншого природного ландшафту в ігровому рушії Unity.1.4.2.

Алгоритм Diamond-Square працює на двовимірному масиві, розмір якого має бути $(2^n+1)*(2^n+1)$ (наприклад, $5*5$, $17*17$, $33*33$ і т. д.).

Етапи виконання алгоритму:

1. *Ініціалізація:* Алгоритм починається із заповнення початкових значень (висот) лише в чотирьох кутах масиву.
2. *Циклічна ітерація:* Далі він циклічно перебирає поступово дедалі менші розміри кроків, послідовно виконуючи дві ключові операції: «Алмазний крок» і «Квадратний крок», доки кожне значення у масиві не буде встановлено.

Ключові кроки:

- *Діамантний крок (Diamond Step):* Береться квадратний фрагмент масиву (визначений чотирма кутами), і обчислюється його центральна точка. Значення висоти цієї центральної точки встановлюється як середнє арифметичне значень чотирьох кутів, плюс випадкове зміщення у визначеному діапазоні. Цей крок створює ромбоподібний (діамантний) візерунок у сітці.

- *Квадратний крок (Square Step):* Береться «діамантний» візерунок, і обчислюються середні точки ребер. Значення висоти цих точок встановлюється як середнє арифметичне значень чотирьох навколишніх точок (які утворюють ромб), плюс випадкове зміщення.

Зменшення діапазону випадкового зміщення на кожній ітерації циклу забезпечує створення більш гладких і реалістичних деталей на дрібніших рівнях, тоді як початкові великі зміщення створюють загальний масштаб рельєфу. Навчання цьому алгоритму є важливим для розвитку у студентів навичок

математичного моделювання та алгоритмічного керування генерацією великих ігрових середовищ.

Алгоритм Perlin noise (Шум Перліна). Шум Перліна є фундаментальним процедурним примітивом текстури, який належить до типу градієнтного шуму. Його використання є майже всюдисущим у комп'ютерній графіці та ігровій індустрії (CGI) для підвищення візуального реалізму .

Функція генерує псевдовипадковий вигляд, де всі візуальні деталі мають однаковий розмір. Ця контрольована випадковість дозволяє легко керувати ним: декілька шарів (октав) шуму Перліна можуть бути скомбіновані в математичних виразах для створення широкої різноманітності реалістичних процедурних текстур (таких як хмари, вогонь, дим, поверхні води чи гірський рельєф), імітуючи контрольовану випадковість природних текстур. Це робить Шум Перліна критично важливим елементом для вивчення PCG у Unity.

У сучасному GameDev його наступники, такі як фрактальний шум та симплексний шум, широко використовуються в модулях обробки графіки для генерації як текстур у режимі реального часу, так і статичних процедурних текстур.

Шум Перліна найчастіше реалізується як дво-, три- або чотиривимірна функція і включає **три основні етапи**:

1. *Визначення сітки випадкових градієнтних векторів.* Визначається n -вимірний сітка. Кожен вузол сітки пов'язаний із фіксованим випадковим градієнтним вектором . У двовимірному випадку, ці вектори визначають напрямки «підйому» або «спаду» шуму від вузла.

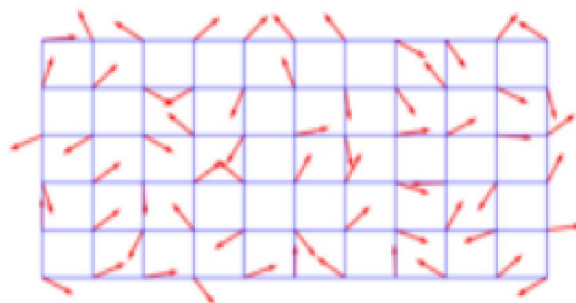


Рис. 1.5 – Двовимірний сітка градієнтних векторів.

2. *Обчислення точкового добутку.* Для обробки значення будь-якої точки-кандидата (де ми хочемо знайти значення шуму), спочатку визначається унікальна комірка сітки, в якій вона знаходиться. Потім ідентифікуються 2^n кутів цієї комірки та пов'язані з ними градієнтні вектори.

- Для кожного кута обчислюється **вектор зміщення** — вектор переміщення від точки кандидата до цього кута.
- Далі для кожного кута обчислюється **точковий добуток** (скалярний добуток) між його градієнтним вектором та вектором зміщення до точки кандидата. Цей точковий добуток показує, наскільки сильно впливає градієнтний вектор кута на значення в цій точці. Складність обчислень має масштабування $O(2^n)$.

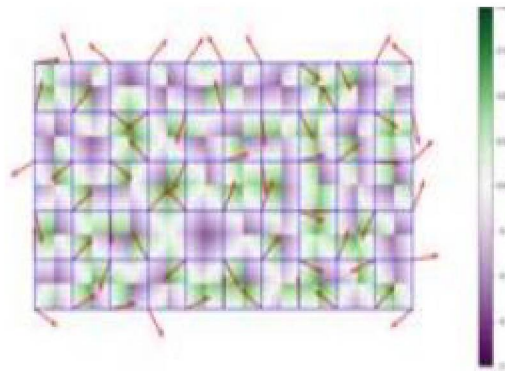


Рис. 1.6 – Двовимірна сітка градієнтних векторів.

3. *Інтерполяція (Згладжування).* Останній крок — інтерполяція (згладжування) між отриманими 2^n точковими добутками. .

- Інтерполяція виконується за допомогою функції, яка має нульову першу похідну (наприклад, smoothstep), що забезпечує плавний перехід і уникнення різких зламів (нульовий перехід) у вузлах сітки.
- Це надає шуму Перліна його характерний, "природний" вигляд.

Навчання реалізації Шуму Перліна та його подальших версій (Simplex) у Unity з використанням C# є фундаментальним для студентів, оскільки дозволяє їм освоїти створення динамічних процедурних світів та текстур, що є необхідною навичкою для розробки сучасних ігрових додатків.

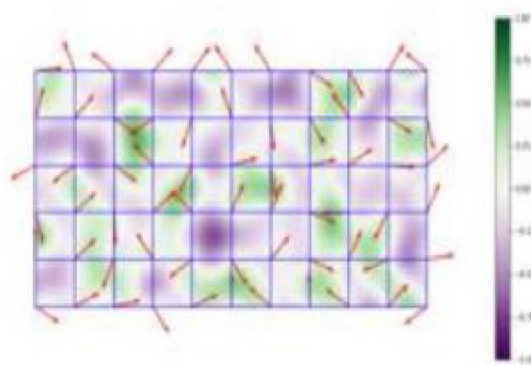


Рис. 1.7 – Остаточний інтерпольований результат.

Висновки до першого розділу

Таким чином, розглянуто ключові відмінності між провідними ігровими рушіями та мовами програмування, що допомогло чітко визначити, що Unity (через інтуїтивно зрозуміле середовище та потужну NavMesh System) у поєднанні з C# (через його об'єктно-орієнтований, зрозумілий синтаксис) є оптимальним вибором для підготовки молодших фахівців. Окрему увагу було приділено особливостям функціонування ігрових рушіїв, зокрема їхній здатності до кросплатформенної розробки та інтеграції функцій ШІ. Це стало основою для подальшого обґрунтування вибору інструментарію для розробки методики навчання студентів фахових коледжів.

Розглянуто структуру типового ігрового додатка та важливість алгоритмічного керування контентом. Описані принципи роботи з процедурною генерацією (PCG), зокрема, детально проаналізовано такі алгоритми, як Колапс хвильової функції (WFC), Diamond-Square та Шум Перліна. Знання цих аспектів стало основою для визначення вимог до майбутньої методики навчання, яка повинна охоплювати не лише ручний (статичний) дизайн, а й вміння програмувати складні алгоритми створення світу.

Було проведено аналіз основних проблем та обмежень, які можуть виникати при використанні ручного дизайну порівняно з PCG, та підкреслено необхідність інтеграції верифікаційних алгоритмів ШІ для забезпечення ігрової

логіки в згенерованих світах. Виявлено, що Unity надає можливість створювати масштабовані та інтерактивні середовища, що є важливим для розробки систем, де поведінка NPC (неігрових персонажів) контролюється елементами штучного інтелекту.

На основі проведеного аналізу, варто відзначити доцільність використання Unity для створення ігрових додатків з елементами ІІІ, що дозволяє ефективно вирішувати завдання формування комплексних професійних компетентностей студентів.

II. МЕТОДИЧНІ АСПЕКТИ НАВЧАННЯ СТУДЕНТІВ ФАХОВИХ КОЛЕДЖІВ РОЗРОБКИ ІГРОВИХ ДОДАТКІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

2.1.Методика побудови середовища ігрового додатку з елементами штучного інтелекту

Успішна розробка сучасних ігрових додатків неможлива без інтеграції ефективного та реалістичного штучного інтелекту (ШІ). На відміну від академічного (наукового) ШІ, метою ігрового ШІ є не досягнення ідеального інтелекту, а створення ілюзії розумної, передбачуваної, але водночас цікавої та адаптивної поведінки неігрових персонажів (NPC). Побудова ігрового ШІ вимагає від розробника глибокого розуміння балансу між обчислювальною ефективністю, правдоподібністю та впливом на загальний ігровий досвід.

У рамках практичної частини дослідження та для цілей реалізації розробленої методики навчання студентів було створено власний навчальний алгоритм генерації ігрового середовища. Цей алгоритм, що ґрунтується на базовому генеруванні випадкових чисел (рандомізації), слугує яскравим і простим прикладом для демонстрації студентам основних принципів процедурного створення контенту (PCG).

Незважаючи на те, що створений алгоритм може бути ресурсозатратним порівняно з комерційними рішеннями (що не є пріоритетом у навчальній роботі), його головними перевагами для освітнього процесу є:

1. *Легкість у проєктуванні:* Він має просту логіку, що швидко засвоюється студентами фахових коледжів.
2. *Швидкість дії:* Дозволяє швидко створювати тестові полігони для інтеграції елементів ШІ.

Основний принцип роботи полягає у послідовній рандомізації значень, яка далі використовується для:

- побудови елементів ігрового середовища (перешкод, стін, ресурсів);
- проведення подальших логічних перевірок (наприклад, перевірка на замкнутість простору, гарантування прохідності).

Для візуалізації та демонстрації принципів роботи цього алгоритму у навчальному проєкті створюється базова сітка (поле), яка нумерується. Наприклад, на початковому етапі сітка має розмір 5x5 (рис. 2.1).

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

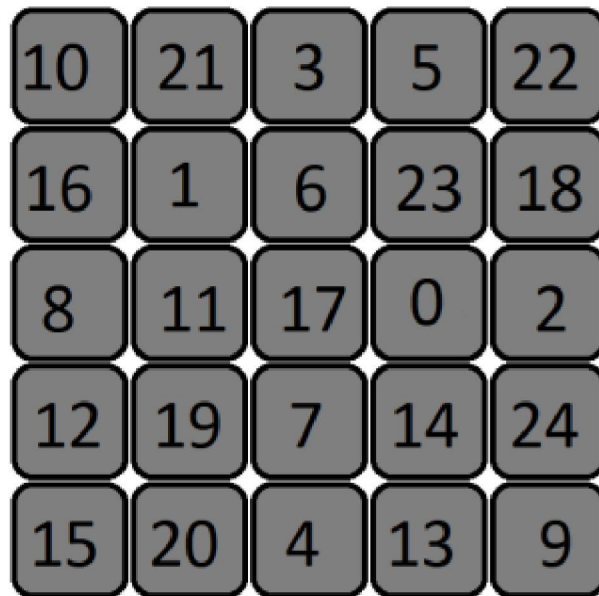
Рис. 2.1 – Початкова сітка.

Наступним кроком у реалізації навчального алгоритму є забезпечення випадкової послідовності обробки елементів сітки, що є необхідною умовою для створення непередбачуваного ігрового середовища.

1. *Ініціалізація структури даних:* Усі індекси або унікальні ідентифікатори (номери) клітинок сітки фіксуються та завантажуються до спеціальної структури даних, наприклад, стеку (Stack) або списку (List).
2. *Випадкове перемішування (Shuffle):* Далі ця послідовність піддається операції випадкового перемішування (shuffling), використовуючи функцію генератора псевдовипадкових чисел.

Мета цього етапу: Рандомізація порядку витягування клітинок зі списку є критично важливою. Вона гарантує, що при подальшій побудові перешкод та

логічних перевірок (наприклад, на прохідність) не виникне візуальних закономірностей (артефактів), які б з'явилися при послідовній обробці (наприклад, зліва направо). Таким чином, студенти на практиці вивчають, як простий рандом використовується для формування органічного та нелінійного ігрового середовища.



10	21	3	5	22
16	1	6	23	18
8	11	17	0	2
12	19	7	14	24
15	20	4	13	9

Рис. 2.2 – Сітка після алгоритму випадкових чисел.

Після того, як номери всіх клітинок були випадково перемішані та занесені у список, розпочинається процес ітеративного розміщення перешкод на полі.

1. *Початок обробки:* Процес починається з вилучення першої комірки (з індексом 0 у перемішаному списку, незалежно від її початкового номера на сітці).
2. *Створення перешкоди:* У вибраній комірці створюється перешкода (наприклад, стіна, камінь або непрохідна зона). Це моделює випадкове розміщення елементів на ігровій карті.
3. *Логічна перевірка:* Наступні кроки алгоритму, які будуть деталізовані далі, обов'язково включатимуть логічну перевірку після розміщення кожної перешкоди. Ця перевірка має гарантувати, що:

- Нова перешкода не порушує правил дизайну (наприклад, не блокує вхід чи вихід).
- Збережено можливість проходу між ключовими точками на полі.

10	21	3	5	22
16	1	6	23	18
8	11	17	0	2
12	19	7	14	24
15	20	4	13	9

Рис. 2.3 – Перша перешкода.

Для студентів цей етап є важливим для вивчення того, як програмувати рандомізацію з логічними обмеженнями, що є основою для багатьох простих елементів штучного інтелекту та процедурної генерації.

Найважливішою умовою функціонування будь-якого процедурно згенерованого ігрового середовища є його логічна узгодженість та прохідність. У контексті нашої методики навчання, це демонструє студентам, як інтегрувати елементи алгоритмів пошуку шляху (типовий елемент ШІ) у процес генерації.

Критичний етап (Валідація): Перед кожною спробою розмістити нову перешкоду у вибраній випадковій комірці, система обов'язково ініціює перевірку карти на наявність недосяжних місць (ізольованих зон).

Як це працює (Принцип, близький до ШІ-алгоритмів):

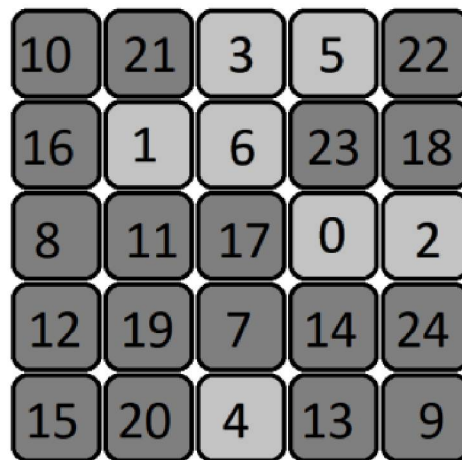
1. *Моделювання проходу*: Фактично, ця перевірка використовує спрощений алгоритм пошуку шляху (наприклад, поширення хвилі або Breadth-First Search/BFS).

2. *Точки контролю:* З однієї "стартової" точки на карті (наприклад, початкова позиція гравця) ініціюється пошук, який має досягти всіх інших незаблокованих клітинок.

3. *Визначення успіху/невдачі:*

- Якщо розміщення потенційної перешкоди призведе до того, що будь-яка інша клітинка стане недосяжною з початкової точки (тобто, створиться повністю ізольована зона), спроба розміщення цієї перешкоди ігнорується, і комірка залишається вільною.
- Якщо всі клітинки залишаються досяжними, перешкода фіксується.

Цей механізм є прямим прикладом того, як принципи, що лежать в основі Навігаційної системи Unity (NavMesh) та алгоритмів пошуку шляху для NPC, застосовуються не лише для руху ворогів, але й для валідації самого ігрового світу. Це ключовий момент для навчання студентів, оскільки вони бачать практичне застосування алгоритмічної логіки.



10	21	3	5	22
16	1	6	23	18
8	11	17	0	2
12	19	7	14	24
15	20	4	13	9

Рис.2.4 – Перше замкнуте середовище.

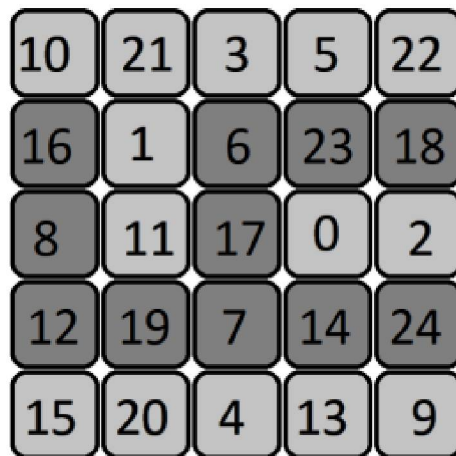
Деталізований процес перевірки наявності недосяжних місць (ізольованих зон) відбувається за допомогою методу, схожого на алгоритм пошуку в глибину або в ширину (DFS/BFS), але адаптованого для валідації ігрового простору:

1. *Ініціація пошуку:* Пошук (трасування) починається з центральної або іншої фіксованої стартової комірки, яка гарантовано є доступною.

2. *Помітка вільних комірок:* Алгоритм ітеративно обходить усі сусідні комірки, позначаючи їх як "досяжні" (пусті). Цей процес триває до тих пір, поки не будуть обійдені всі клітинки, до яких можна дістатися зі стартової точки.
3. *Фінальна валідація:* Після завершення обходу виконується підрахунок. Якщо різниця між загальною кількістю комірок на полі та кількістю пустих (досяжних) комірок не дорівнює сумі вже поставлених перешкод, це однозначно означає, що запропонована перешкода створює ізольовану (недосяжну) зону. У такому випадку спроба розміщення перешкоди відхиляється.

Цей механізм гарантує, що згенерований світ завжди буде логічно цілісним та прохідним. Для студентів це є чудовим прикладом того, як алгоритмічна логіка (яка є основою ШІ для пошуку шляху) інтегрується для валідації самого контенту.

Повторюючи цей процес, що включає випадковий вибір клітинки та сувору логічну перевірку, алгоритм поступово заповнює ігрове поле перешкодами.



10	21	3	5	22
16	1	6	23	18
8	11	17	0	2
12	19	7	14	24
15	20	4	13	9

Рис. 2.5 – Готовий лабіринт.

У фінальному результаті, коли кількість перешкод була доведена до максимально можливої кількості (з дотриманням умови прохідності), виходить функціональний лабіринт або складне ігрове середовище. Це середовище

ідеально підходить для навчальної інтеграції елементів штучного інтелекту, таких як:

- Тестування алгоритмів пошуку шляху (наприклад, A*) для NPC.
- Перевірка логіки прийняття рішень (Behavior Trees) для навігації в складному просторі.

2.2. Методика реалізації моделей штучного інтелекту ворожих об'єктів

У цьому розділі, присвяченому створенню навчального ігрового додатку, було інтегровано функціональний штучний інтелект (ШІ) для моделювання поведінки ворожих елементів. Завдяки цьому, вороги здатні самостійно відстежувати гравця та атакувати його, рухаючись до цілі по оптимальному маршруту. Ворожі об'єкти будуть постійно йти до гравця по найкоротшому обраному шляху, завдаючи шкоди при безпосередньому контакті.

Пошук шляху (Pathfinding) є класичним та фундаментальним завданням для ігрового ШІ, що вимагає ефективного розв'язання оптимізаційної задачі на графах. У навчальному процесі студенти ознайомлюються як з теоретичними (академічними) алгоритмами, так і з їхніми практичними ігровими реалізаціями:

- *Академічні алгоритми:* Алгоритм Дейкстри (надійний, але повільний) та Алгоритм A* (швидший, але вимагає більше пам'яті). Обидва алгоритми добре працюють на невеликих графах, проте з ростом масштабу ігрового світу їхня обчислювальна складність різко зростає, що вимагає застосування їхніх сильно модифікованих версій у комерційних іграх.

- *Альтернативні методи:* Також використовуються хвильові алгоритми, просте "обходження перешкод" та численні комбінації цих підходів.

З огляду на цілі методики навчання студентів фахових коледжів, найкращим і найбільш доцільним варіантом для швидкої та ефективної реалізації пошуку найкоротшого шляху була обрана система NavMesh (Навігаційна Сітка), вбудована безпосередньо в ігровий рушій Unity.

Переваги NavMesh для навчання:

1. *Інтеграція та Простота:* NavMesh автоматично генерує прохідні зони (сітку) на основі геометрії ігрового світу (створеної, наприклад, за допомогою PCG). Це дозволяє студентам швидко інтегрувати базовий ІШ, фокусуючись на логіці поведінки (на C# скриптах), а не на оптимізації самого алгоритму пошуку шляху.

2. *Висока Продуктивність:* Unity NavMesh використовує високооптимізовані внутрішні алгоритми, що забезпечує швидкий і надійний пошук шляху навіть на великих, складних картах, згенерованих нашим алгоритмом.

Використання NavMesh дозволяє студентам наочно побачити, як NPC (представлений компонентом NavMeshAgent в Unity) розумно переміщується, оминаючи перешкоди (які ми розмістили на попередньому етапі).

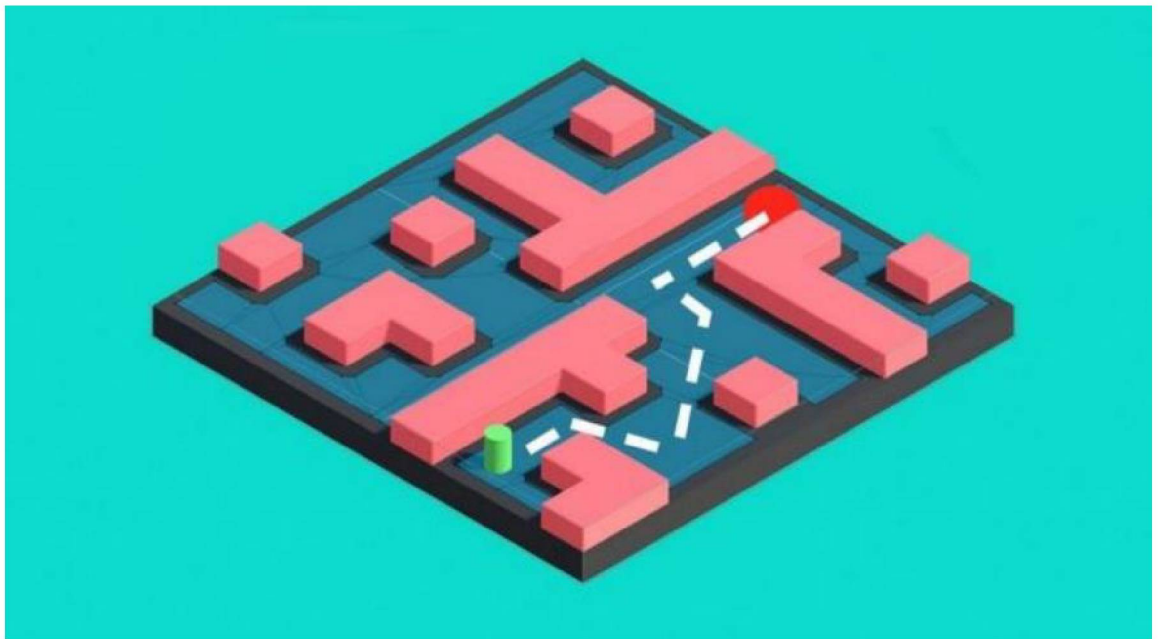


Рис. 2.6 – Вибраний шлях до точки.

Хоча система NavMesh (Навігаційна Сітка) в Unity є високооптимізованою вбудованою функцією, в її основі лежить класичний алгоритм пошуку шляху, найчастіше варіант алгоритму A*. Для навчання студентів важливо розуміти

базову послідовність дій, яку виконує компонент NavMeshAgent при знаходженні маршруту для неігрового персонажа (NPC).

Дано:

- S — Початковий стан пошуку (позиція NPC).
- T — Цільовий стан (позиція гравця або іншої точки).
- Q — Набір вузлів (частин NavMesh-сітки), які необхідно перевірити.

Алгоритм пошуку шляху:

1. *Ініціалізація (Крок 1):* Система ініціалізує список для перевірки (Q), додаючи до нього початковий вузол S як єдиний вхід.
2. *Перевірка списку (Крок 2):* Алгоритм перевіряє, чи порожній список Q. Якщо так, пошук неможливий, і він зупиняється. В іншому випадку, вибирається найкращий елемент (n) із Q (зазвичай, той, що має найменшу очікувану загальну вартість шляху).
3. *Перевірка цілі (Крок 3):* Якщо обраний стан (n) збігається з цільовим станом (T), шлях знайдено, і алгоритм повертає n.
4. *Розширення шляху (Крок 4-5):*
 - Вузол n видаляється зі списку перевірки Q.
 - Алгоритм знаходить усіх нащадків (сусідні прохідні вузли NavMesh) стану n, які ще не були перевірені.
 - Створюються розширені шляхи до n від кожного знайденого нащадка.
5. *Повторення (Крок 6):* Усі новостворені розширені шляхи додаються до списку Q, і процес повторюється, повертаючись до Кроку 2.

Розуміння цієї циклічної логіки дозволяє студентам фахових коледжів не просто використовувати вбудований функціонал NavMesh, а й діагностувати проблеми з навігацією та, за необхідності, модифікувати логіку руху NPC у своїх C# скриптах.

Для демонстрації студентам передових методів ІІІ та формування навичок роботи з адаптивною поведінкою, у навчальному проєкті застосовано штучну

нейронну мережу (ШНМ). ШНМ — це низка алгоритмів, що імітують роботу людського мозку для розпізнавання складних взаємозв'язків у наборі даних. У нашому випадку це допомагає боту приймати оптимальні рішення в динамічному ігровому середовищі.

Для реалізації та навчання бота було використано Unity ML-Agents — спеціалізований інструментарій, який дозволяє інтегрувати машинне навчання (Reinforcement Learning) безпосередньо в ігрове середовище Unity. Це дає студентам можливість освоїти актуальні інструменти, що використовуються в сучасній індустрії.

Алгоритм роботи машинного навчання бота:

1. *Вхідні дані (Сенсори)*: Бот аналізує навколишній світ, використовуючи сенсори — він "пускає" 12 променів в радіусі до 100 юнітів. Це є імітацією зору NPC.

- *Типи вхідних даних*: Кожен промінь повертає інформацію одного з трьох типів: наявність ворога, порожнечі або перешкоди.

2. *Обробка (Ваги)*: Зібрані дані (12 значень) надсилаються на ваги нейронної мережі, де відбувається аналіз інформації.

3. *Навчання (Кількість прогонів)*: Шляхом багаторазових прогонів та оцінки винагороди (Reinforcement Learning) бот поступово коригує свої ваги, вибираючи все більш точні та ефективні команди.

4. *Вихідні дані (Команди)*: На виході нейронна мережа генерує команди, які визначають:

- Куди боту потрібно йти.
- Чи варто стріляти.
- Напрямок стрільби.

Для керування персонажем гравця, використано повнозв'язну нейронну мережу (Fully Connected Network).

- Топологія: 12-12-12-6.
- Вхідний шар: 12 нейронів (відповідає 12 сенсорним променям).

- Приховані шари: Два приховані шари по 12 нейронів.
- Вихідний шар: 6 нейронів (відповідає 6 можливим командам або діям боту: рух вперед/назад, поворот, стрільба тощо).

Навчання студентів конструюванню та тренуванню такої ШНМ є вершиною практичної підготовки в цьому модулі, демонструючи перехід від класичних алгоритмів (NavMesh, FSM) до адаптивних систем ІІІ.

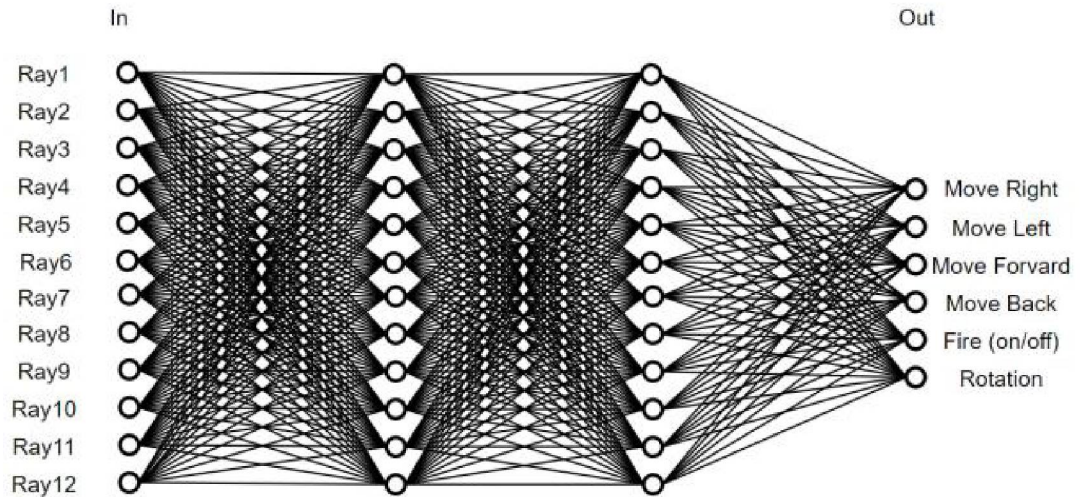


Рис. 2.7 – Діаграма нейронної мережі.

Unity ML-Agents працює за класичним принципом навчання з підкріпленням (Reinforcement Learning), де агент (бот) навчається оптимальній поведінці шляхом постійної взаємодії із симуляційним середовищем. Нейронна мережа не просто приймає рішення, вона керує повним циклом, у якому вихідні дані (дії) агента впливають на входні дані (спостереження), які він отримує наступного разу.

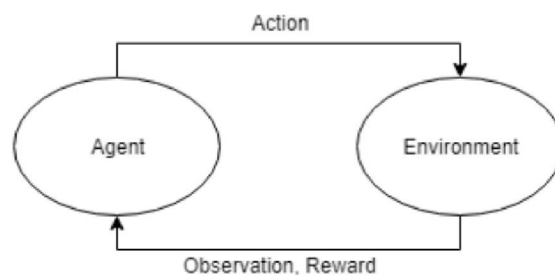


Рис. 2.8 – Діаграма взаємодії агента та середовища

Unity ML-Agents контролює вхідні та вихідні дані залежно від реакції середовища. Цей цикл можна описати так:

1. *Спостереження (Вхід)*: Бот аналізує середовище (за допомогою 12 променів-сенсорів). Ці дані є входом для його нейронної мережі.
2. *Рішення (Процес)*: Нейронна мережа обробляє вхідні дані та генерує команди (6 вихідних нейронів).
3. *Дія (Вихід)*: Команди перетворюються на дії в ігровому світі (рух, стрільба).
4. *Винагорода / Покарання (Реакція Середовища)*: Середовище реагує на дію агента, надаючи йому винагороду (наприклад, за наближення до гравця чи влучний постріл) або покарання (за отримання шкоди чи зіткнення з перешкодою).
5. *Навчання*: Система використовує цю винагороду для коригування ваг нейронної мережі, оптимізуючи її майбутні рішення.

Саме цей постійний зворотний зв'язок дозволяє боту адаптивно навчатися оптимальній стратегії гри. Навчання цьому циклу демонструє студентам не лише програмування ІІІ, а й принципи моделювання складних адаптивних систем.

Для оптимізації поведінки ігрового бота, керованого нейронною мережею, використовується Генетичний Алгоритм (Genetic Algorithm, GA). Цей метод належить до класу еволюційних обчислень і є потужною демонстрацією того, як принципи природного відбору можуть бути застосовані для розв'язання складних оптимізаційних задач у ІІІ.

Генетичний алгоритм імітує еволюційний процес, адаптуючи параметри нейронної мережі (її ваги) до вимог ігрового середовища:

1. *Ініціалізація та Оцінка Ефективності (Fitness Evaluation)*: На початку генерується набір різних випадкових нейронних мереж (початкова популяція ботів). Кожна мережа тестується в ігровому середовищі (на "арені"). Оцінка ефективності ("пристосованості") визначається метрикою, наприклад,

часом, протягом якого бот успішно підтримувався на арені до його знищення ворогами.

2. *Селекція (Відбір)*: З популяції відбираються лише найкращі особини (ті, чия оцінка вища за середню).

3. *Формування Нової Популяції*: На основі відібраних "батьківських" мереж формується нова популяція. Ваги зв'язків у нейронах випадковим чином змінюються через мутації та рекомбінацію.

4. *Повторення*: Нова популяція знову тестується, і процес повторюється. З кожним новим поколінням відбираються боти з кращими результатами, що веде до еволюційного покращення їхньої поведінки.

Навчання студентів фахових коледжів цим процедурам є критичним для розуміння механізмів адаптації ІІІ:

Селекція (Selection): Завданням селекції є відбір найуспішніших «генотипів» (нейронних мереж) для розмноження. У цьому проєкті використовується метод Remainder Stochastic Sampling. Після моделювання розраховується середня оцінка успішності по популяції. Відбираються особини з оцінкою, вищою за середню, і клонуються пропорційно їхній успішності. Це дозволяє зберегти розмір популяції, одночасно забезпечуючи переважне розмноження найкращих генотипів.

Мутація (Mutation): Це процес, у якому випадковим чином вибирається певна кількість нейронних мереж, і в них деякі випадкові зв'язки (ваги) змінюються на невелику випадкову величину. У результаті отримується нова ШНМ з трохи іншими вагами, що забезпечує дослідження нових, потенційно кращих стратегій.

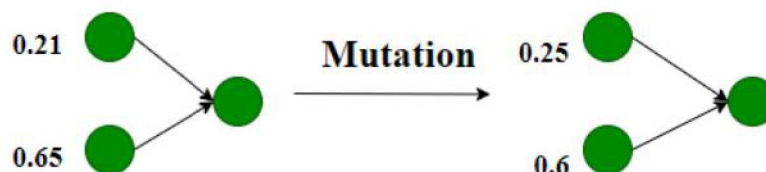


Рис. 2.9 – Схематичне зображення зміни ваг у нейроні внаслідок мутації.

Рекомбінація (Crossover): Це процес обміну генами (тобто вагами нейронних зв'язків) між двома "батьківськими" ботами, відібраними для розмноження. Для рекомбінації випадковим чином відбираються дві ШНМ, і відбувається обмін частинами ваг між ними

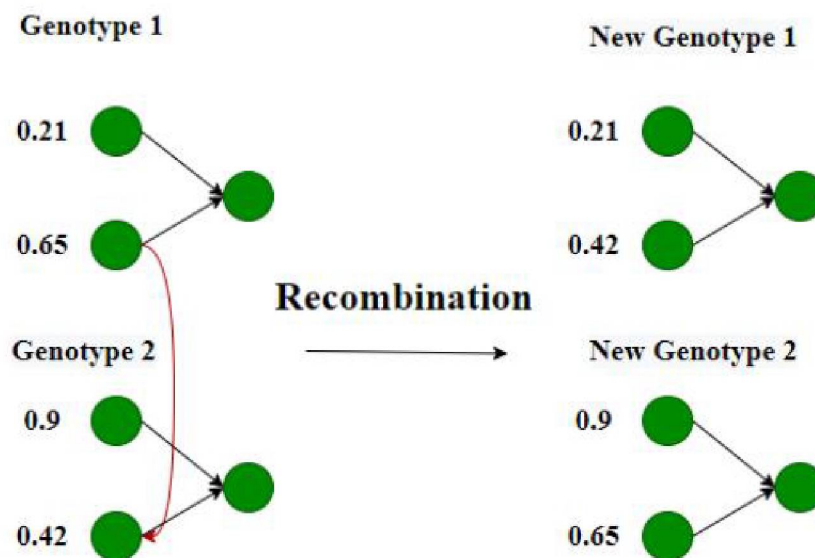


Рис. 2.10 – Схема обміну частинами ваг між двома нейронними мережами.

У результаті отримуються дві нові особини, які поєднують у собі найкращі зв'язки двох предків, сприяючи швидкій конвергенції до оптимального рішення.

2.3. Методика навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту.

Методика навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту вимагає наявності чіткої, функціональної та архітектурно правильно побудованої моделі-проєкту. Третій розділ є безпосереднім відображенням цієї моделі. Він демонструє, як теоретично обґрунтовані алгоритми ШІ та процедурної генерації трансформуються у прикладний програмний код на C# в середовищі Unity.

Суть методики навчання студентів на цьому етапі включає:

1. *Формування навичок об'єктно-орієнтованого програмування (ООП):* Навчання проєктуванню складної, масштабованої структури класів та інтерфейсів.
2. *Освоєння процедурного генерування (PCG):* Практична реалізація алгоритмів створення динамічного ігрового середовища (карт, перешкод).
3. *Інтеграція ігрового ШІ:* Освоєння вбудованих інструментів Unity (NavMesh) та власних рішень (генетичний алгоритм) для моделювання поведінки NPC.
4. *Професійне використання інструментарію:* Набуття навичок роботи з Unity Editor та Visual Studio, що повністю імітує реальний робочий процес.

Як основне середовище розробки ігрового додатка було обрано Unity, що є індустріальним стандартом та найдоцільнішим інструментом для освітнього процесу.

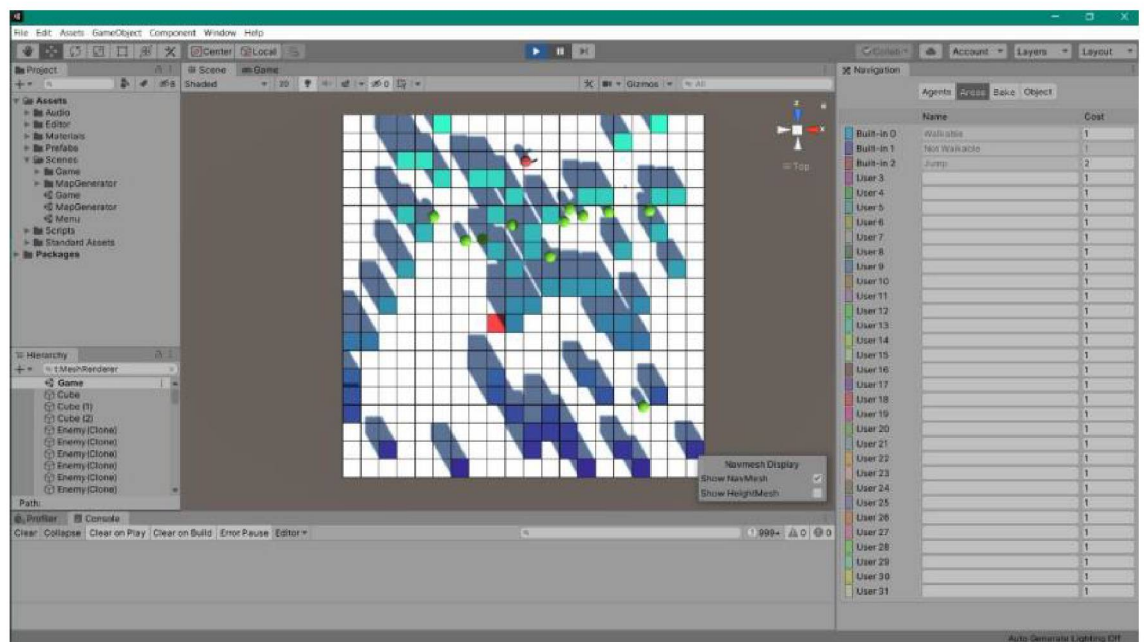


Рис. 2.11 – Середовище розробки.

Для редагування та компілювання коду на C# використовувалася інтегрована платформа Visual Studio. Вибір обумовлений зручністю графічного

інтерфейсу Unity, потужними засобами налагодження та широкою підтримкою професійного програмування.

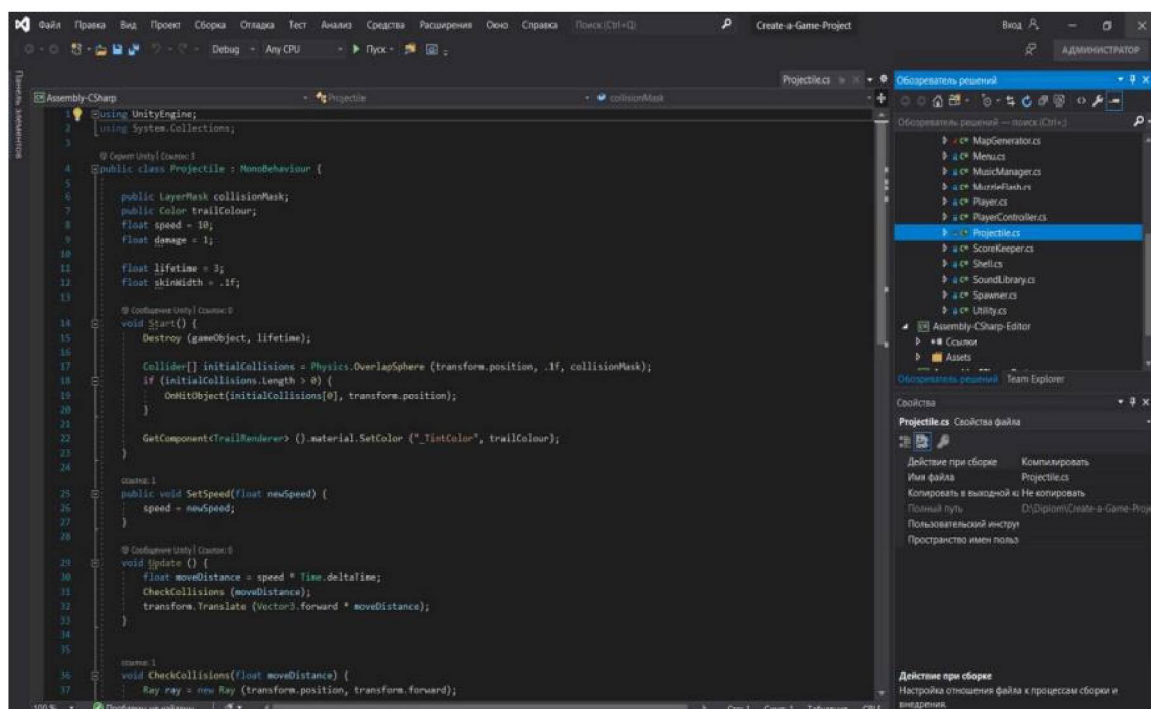


Рис. 2.12 – Редактор додатку.

Елементи Unity, які є критичними для реалізації методики навчання:

- *Інтуїтивний Інтерфейс та Налагодження:* Unity забезпечує зручне налаштування ігрового інтерфейсу та безшовну інтеграцію у повноцінний програмний код C#, що мінімізує поріг входження для студентів.
- *Система Prefabs (Префаби):* Використання префабів є частиною методики, що вчить студентів принципам об'єктно-орієнтованого підходу (ООП). Настроєний об'єкт може бути збережений та використаний багаторазово, що спрощує масштабування проєкту.
- *Моноповедінка (MonoBehaviour):* Усі класи, що взаємодіють з ігровою сценою, є нащадками класу MonoBehaviour. Це гарантує інтеграцію класу з ігровим циклом і є обов'язковим елементом для освоєння програмування логіки в Unity.

Проект повинен мати чітку об'єктно-орієнтовану структуру (ООП), що є основою для формування навичок системного програмування. Діаграма класів є ключовим елементом моделювання (подана на Рисунку 2.13).

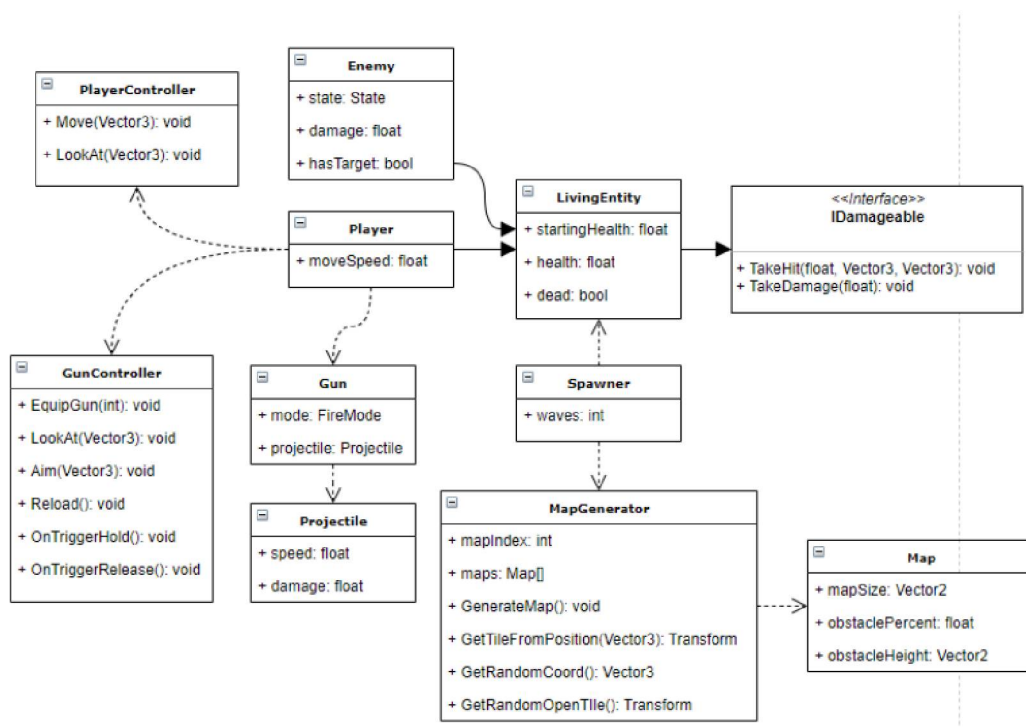


Рис. 2.13 – Діаграма класів.

Таблиця 2.1 – Ключові класи, що демонструють ООП-зв'язки.

Клас / Інтерфейс	Призначення та Зв'язок (Навчальний фокус)
LivingEntity	Базовий клас для абстрагування спільних властивостей живих об'єктів.
Enemy	Клас ворога, наслідує LivingEntity. Критичний для ІІІ-модуля.
IDamageable	Інтерфейс, що забезпечує поліморфізм для обробки шкоди.
MapGenerator	Клас, що генерує карту. Критичний для РСG-модуля.
PlayerController	Клас, що контролює рух та приціл гравця.
Projectile	Клас, що відповідає за кулю, наслідується від Gun.
Spawner	Клас, що відповідає за хвилі ворогів.

При створенні гри варто використовувати три основні алгоритми, реалізовані в C#, що демонструють освоєння студентами алгоритмічного підходу:

1. Алгоритм випадкових чисел (метод ShuffleArray).
2. Алгоритм генерування перешкод (метод GenerateMap).
3. Алгоритм перевірки доступності середовища (метод MapIsFullyAccessible).

Для підтвердження функціональності розроблених модулів та демонстрації результатів роботи алгоритмів процедурної генерації контенту (PCG) була ініційована нова ігрова сесія.

На початку сесії була згенерована навчальна сцена розміром 20×20 клітинок. Це слугує полігоном для подальшої інтеграції та тестування елементів штучного інтелекту. На Рисунку 2.14 демонструється початковий вигляд ігрового середовища, створеного процедурним алгоритмом (Map Generator), що підтверджує успішну реалізацію логіки генерації перешкод та прохідності карти.

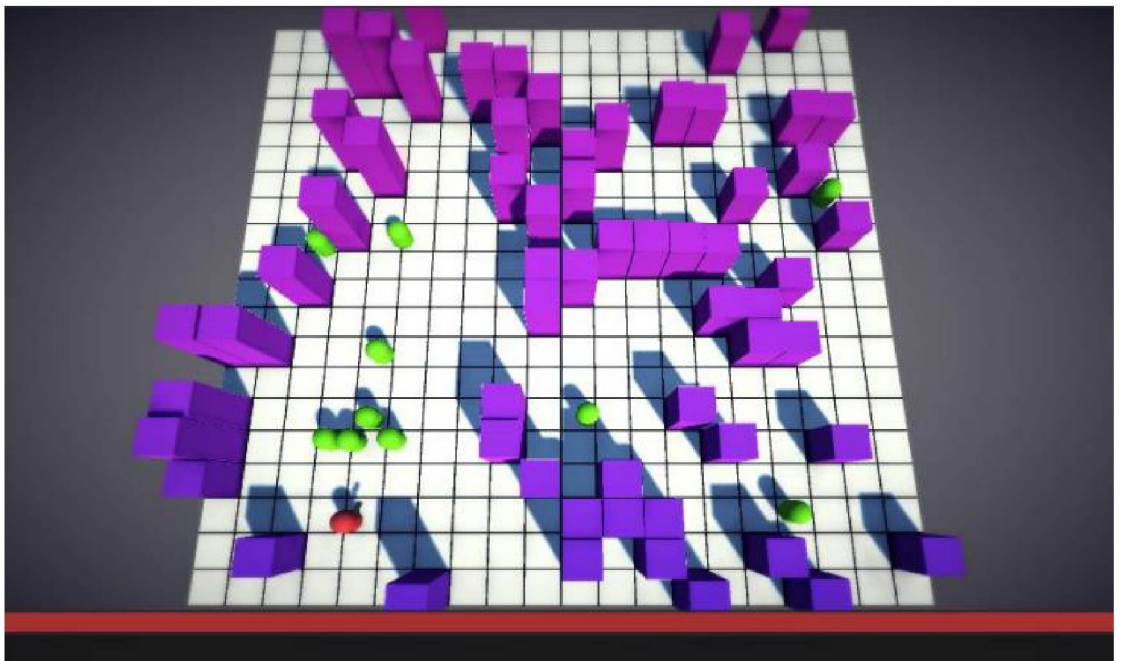


Рис. 2.14 – Початковий вигляд ігрового середовища

Це тестування є обов'язковим етапом методики, оскільки воно візуально підтверджує, що:

1. *Алгоритми PCG* (зокрема, перевірка доступності середовища) працюють коректно.

2. *Базова ігрова архітектура* здатна завантажувати та відображати згенерований світ

У рамках демонстрації функціоналу класу *Spawner* та елементів управління ігровим процесом, реалізовано механізм випадкового спауну ворогів.

Методичний фокус: Цей етап показує студентам, як інтегрувати елементи випадковості та механіки "попередження" для забезпечення справедливого та динамічного ігрового досвіду.

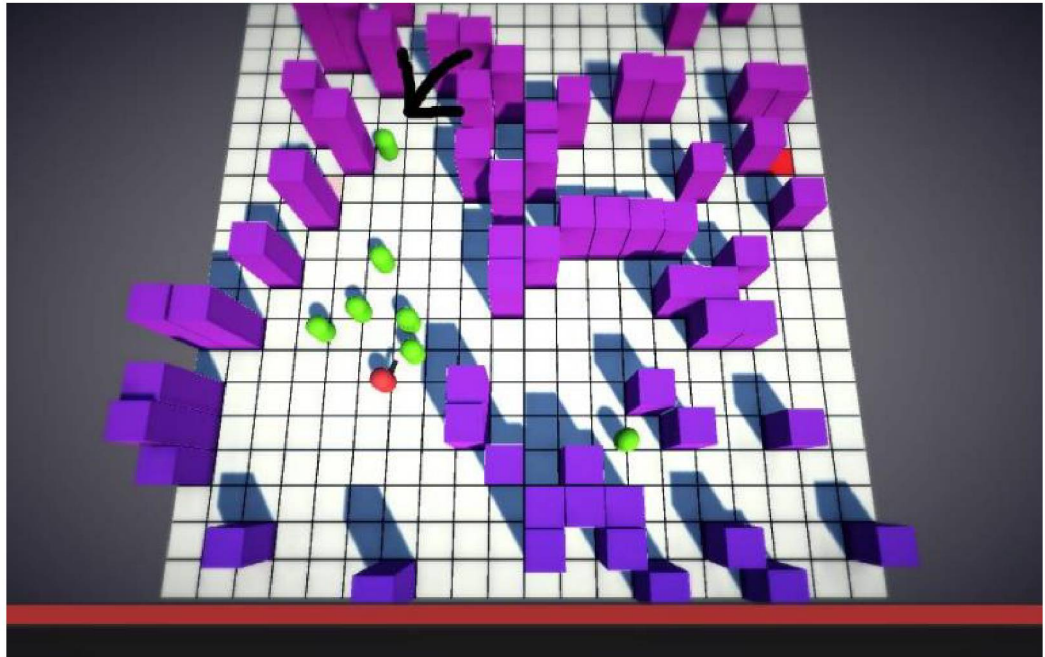


Рис. 2.15 – Генерація ворогів

Реалізація:

1. Випадкова генерація позиції: Позиція спауну ворога обирається випадковим чином у межах прохідної зони згенерованої карти.

2. Система попередження: Для підвищення прозорості ігрової логіки, за 3 секунди до появи ворога система активує візуальне попередження. Це відбувається шляхом підсвічування клітинки спауну червоним кольором.

3. Демонстрація: На Рисунках 3.8 та 3.9 візуально продемонстровано послідовність етапів спауну: від активації попереджувального підсвічування до появи ворожого NPC, готового до активації свого ІП.

Така реалізація дозволяє студентам освоїти не тільки базову логіку випадкового розміщення об'єктів, але й принципи створення дружнього до гравця (user-friendly) ігрового дизайну.

Висновки до другого розділу

Таким чином, у розділі розглянуто теоретичні та практичні аспекти побудови навчальної моделі ігрового додатку, що є основою розробленої методики. Проведений аналіз та методика побудови середовища ігрового додатку дозволили створити ефективний навчальний алгоритм процедурної генерації контенту (PCG). Цей алгоритм, що базується на рандомізації та логічній валідації (перевірка на недосяжність), є потужним інструментом для розвитку у студентів алгоритмічного мислення та навичок програмування логічних обмежень, що є основою функціонування ігрового ІШ. Методика реалізації моделей штучного інтелекту ворожих об'єктів базується на комплексному підході, що включає використання вбудованої системи Unity NavMesh для швидкої інтеграції пошуку шляху та застосування Штучної нейронної мережі (ШНМ), навченої за допомогою Генетичного алгоритму (з топологією 12-12-12-6), для моделювання адаптивної поведінки NPC. Аналіз цих механізмів дозволяє студентам освоїти як класичні, так і еволюційні методи ІШ.

Розроблена методика навчання студентів фахових коледжів включає чотири ключові компоненти, необхідні для формування повноцінної професійної компетентності: Формування навичок об'єктно-орієнтованого програмування (ООП): Навчання проєктуванню складної, масштабованої структури класів (LivingEntity, Enemy) та інтерфейсів (IDamageable) на C#. Освоєння процедурного генерування (PCG): Практична реалізація алгоритмів генерації карт та їх логічної перевірки. Інтеграція ігрового ІШ: Освоєння вбудованих інструментів Unity (NavMesh) та власних рішень (генетичний алгоритм, Unity

ML-Agents). Професійне використання інструментарію: Набуття навичок роботи з Unity Editor та Visual Studio, що повністю імітує реальний робочий процес.

Розглянуті аспекти сприяють формуванню у майбутніх фахівців галузі цифрових технологій необхідних компетентностей для створення інноваційних та ефективних ігрових додатків, здатних забезпечити високий рівень реалізму та адаптивної поведінки.

ІІІ. ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ МЕТОДИКИ НАВЧАННЯ СТУДЕНТІВ ФАХОВИХ КОЛЕДЖІВ РОЗРОБКИ ІГРОВИХ ДОДАТКІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

3.1. Організація та проведення експерименту

Для перевірки ефективності розробленої методики навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту був проведений педагогічний експеримент.

Дослідження організовано на базі ВСП «Фаховий коледж економіки, права та інформаційних технологій ЗУНУ» за участі 22 студентів спеціальності «Комп'ютерна інженерія». Методика впроваджувалась у навчальний процес дисципліни «Спеціалізовані мови програмування», оскільки вона забезпечує вивчення мови C# та принципів ООП, необхідних для роботи з Unity. Метою експерименту стало визначення змін у навчально-пізнавальній діяльності студентів та рівні сформованості професійних компетентностей після впровадження розробленої методики.

У ході експерименту застосовувався метод анкетування та контрольних зрізів знань, що забезпечило кількісний та якісний аналіз отриманих результатів. Для об'єктивної оцінки ефективності методики було сформовано дві групи:

- Експериментальна група (ЕГ): 11 студентів.
- Контрольна група (КГ): 11 студентів.

Експеримент тривав протягом чотирьох навчальних тижнів. У контрольній групі навчання здійснювалось за традиційною методикою викладання програмування та алгоритмізації, тоді як в експериментальній групі використовувалась запропонована методика, яка передбачає проєктний підхід та безпосереднє застосування Unity, C# та інструментів ІІІ (NavMesh, ML-Agents).

У педагогічному експерименті забезпечувалась умова однакового обсягу навчальної інформації для студентів обох груп, а також постановка ідентичних

за складністю дидактичних завдань (наприклад, реалізація алгоритму пошуку шляху чи процедурної генерації). Відмінність полягала виключно у використанні різних методів, засобів і прийомів навчання.

Для перевірки впливу запропонованої методики на якість знань студентів застосовувалась методика поелементного та поопераційного аналізу контрольних робіт. Згідно з цим підходом, контрольні завдання аналізуються для отримання точної кількісної оцінки рівня:

1. Засвоєних теоретичних знань (ІІІ-алгоритми, принципи PCG).
2. Сформованих практичних навичок (проєктування класів, інтеграція NavMesh).
3. Здатності вирішувати комплексні завдання (наприклад, створення функціонального NPC).

Під час експерименту було заплановано два контрольних зрізи знань: початковий (констатувальний, для підтвердження рівності груп) та фінальний (формульвальний, для виявлення динаміки змін у навчальній діяльності).

Результати фінального контрольного зрізу дали змогу провести порівняльний аналіз між якісними показниками навчальної діяльності студентів ЕГ та КГ, що дозволило обґрунтувати ефективність запропонованої методики навчання та її позитивний вплив на формування професійних компетентностей.

3.2. Оцінка ефективності запропонованої розробки

Для об'єктивного визначення динаміки знань і навичок студентів у процесі експерименту застосовувався комплексний підхід до оцінювання.

На додаток до основних контрольних зрізів, для визначення рівня засвоєння теоретичного матеріалу (алгоритми ІІІ та PCG) використовувалось усне опитування. Це забезпечувало безпосередню вербальну взаємодію між студентом і викладачем, дозволяючи оцінити глибину розуміння концепцій, таких як принцип роботи NavMesh чи логіка Генетичного алгоритму. Отримані

результати були систематизовані та занесені до відповідних таблиць для подальшого кількісного аналізу.

Для забезпечення об'єктивності оцінювання, контрольні перевірки (практичні завдання) проводились за умов самостійного виконання студентами. Це досягалось завдяки розробці різноманітних варіантів завдань, що виключало можливість простого копіювання рішень.

Основним показником під час експерименту було співвідношення середніх арифметичних балів, отриманих студентами експериментальної ($\bar{X}_e$) та контрольної ($\bar{X}_k$) груп. Це дозволило об'єктивно порівняти ефективність традиційного підходу до навчання з розробленою методикою [17].

$$\eta = \frac{X_e}{X_k} \quad (1)$$

Для коректного визначення ефективності запропонованої методики навчання розробки ігрових додатків з елементами ІІІ важливим є правильний вибір показників, які забезпечують наукову обґрунтованість висновків. Ці висновки формулюються виключно на основі експериментальних даних, представлених у кількісних показниках.

Показники ефективності визначаються шляхом аналізу структури навчальної діяльності студентів та послідовності виконання практичних операцій. Для забезпечення об'єктивної оцінки рівня сформованості професійних компетентностей встановлено наступні вимоги до поелементного та поопераційного аналізу контрольних завдань:

1. **Коректна ініціалізація та реалізація PCG:** Перевірка правильності коду C# у методах, що відповідають за процедурну генерацію (наприклад, реалізація логіки ShuffleArray та GenerateMap).

2. **Забезпечення логічної цілісності середовища:** Оцінка коректності алгоритму MapIsFullyAccessible (перевірка прохідності карти), що є прямим застосуванням алгоритмічної логіки, критичної для ШІ.

3. **Правильне використання ООП та Unity-компонентів:** Оцінка здатності студента наслідувати класи (MonoBehaviour, LivingEntity) та коректно використовувати префаби та інтерфейси (IDamageable).

4. **Інтеграція та функціональність ШІ-модуля:** Перевірка успішної інтеграції та коректного функціонування вбудованого NavMesh Agent для пошуку шляху ворожим NPC.

5. **Розуміння та конфігурація адаптивної логіки:** Оцінка здатності студента налаштовувати ключові параметри для навчання нейронної мережі (наприклад, конфігурація сенсорів та винагород у Unity ML-Agents).

У протоколі результатів навпроти кожного студента фіксуються позначки: «+» ставиться проти засвоєних елементів (операцій), а «-» – проти тих, які потребують доопрацювання. Приклад заповнення такого протоколу оцінювання представлено в таблиці 3.1.

Таблиця 3.1 – Протокол по-елементного аналізу

№	Перелік вимог	Студент										
		1	2	3	4	5	6	7	8	9	10	11
1	Коректна ініціалізація та реалізація PCG	+	+	-	+	+	-	+	-	+	+	+
2	Забезпечення логічної цілісності середовища.	+	-	+	-	+	-	-	+	+	+	+
3	Правильне використання ООП та Unity-компонентів	+	+	-	-	-	+	-	-	-	+	-
4	Інтеграція та функціональність ШІ-модуля	-	+	+	-	-	+	-	-	+	-	-
5	Розуміння та конфігурація адаптивної логіки	-	-	-	-	-	+	-	-	+	-	+
	Всього	3	3	2	1	2	3	1	1	4	3	3

На основі зібраних даних розраховується коефіцієнт повноти засвоєння навчального матеріалу за спеціальною формулою, що дозволяє оцінити рівень сформованості знань та практичних навичок у студентів.

На основі протоколу визначається коефіцієнт повноти засвоєння змісту понять за формулою:

$$K = \frac{\sum_{i=1}^N n_i}{nN} \quad (2)$$

де n_i – кількість ознак, засвоєних i -м студентом;

n – кількість ознак (елементів) поняття, які на даний момент повинні бути засвоєні;

N – кількість студентів, що виконували роботу.

Коефіцієнт ефективності в цьому випадку визнається як відношення значень коефіцієнтів повноти засвоєння змісту K_e експериментальної та K_k контрольної груп:

$$\eta = \frac{K_e}{K_k} \quad (3)$$

Ключовим критерієм кількісної оцінки ефективності розробленої методики навчання є коефіцієнт повноти виконання операцій. Цей показник розраховується на основі поопераційного аналізу практичних завдань, виконаних студентами, і відображає рівень сформованості їхніх практичних та алгоритмічних навичок. У процесі перевірки до протоколу вносять перелік усіх критичних операцій, які повинні бути виконані для досягнення поставленої мети – навчити студентів розробляти ігрові додатки з елементами ІШ, використовуючи Unity та C#.

Операції записуються в логічній послідовності: кожна наступна операція залежить від правильного виконання попередньої. Для забезпечення точності оцінки встановлено наступні вимоги до поопераційного аналізу завдань:

1. Ініціалізація та структурування проєкту: Коректне налаштування середовища Unity, створення та успадкування ключових класів ООП (MonoBehaviour, LivingEntity) на C#.

2. **Програмування PCG-алгоритму:** Правильна реалізація алгоритмів процедурної генерації (наприклад, GenerateMap) та забезпечення коректної валідації ігрового середовища (перевірка на прохідність).

3. **Інтеграція навігаційної логіки (III):** Успішне використання вбудованого функціоналу Unity (запікання NavMesh) та коректне написання C# скриптів для керування агентом (NavMeshAgent) NPC.

4. **Моделювання поведінки III:** Реалізація логіки прийняття рішень для NPC (наприклад, реалізація простих скінченних автоматів або інтеграція Unity ML-Agents) для відстеження та атаки гравця.

5. **Тестування та оптимізація:** Здатність студента самостійно діагностувати та виправляти помилки у взаємодії між PCG-середовищем та AI-агентами, а також перевірка продуктивності (наприклад, за допомогою Unity Profiler).

У протоколі, навпроти кожного студента, позначають виконані операції знаком «+», а ті, які були виконані неправильно або не виконані, – знаком «-». У графі «Всього» зазначають загальну кількість операцій, які повинен був виконати студент. Приклад заповнення протоколу, що відображає поопераційний аналіз, представлено в таблиці 3.2.

Таблиця 3.2 – Протокол поопераційного аналізу

№	Перелік вимог	Студент										
		1	2	3	4	5	6	7	8	9	10	11
1	Ініціалізація та структурування проекту	+	+	-	+	+	-	+	-	+	+	+
2	Програмування PCG-алгоритму	+	-	+	-	+	-	-	+	+	+	+
3	Інтеграція навігаційної логіки (III)	+	+	-	-	-	+	-	-	-	+	-
4	Моделювання поведінки III	-	+	+	-	-	+	-	-	+	-	-
5	Тестування та оптимізація	-	-	+	-	-	+	-	-	+	-	+
	Всього	3	3	3	1	2	3	1	1	4	3	3

Скориставшись формулою (3), оцінимо знання кожного студента у відносних одиницях, де оцінка знаходиться в межах $0 < p < 1$. Чим ближче до 1 значення p , тим краща оцінка [19]. За даними таблиць визначимо середнє

арифметичне значення коефіцієнта виконання операції студентами експериментальної та контрольної груп:

$$\bar{p}_k = \frac{\sum_{i=1}^N p_i}{pN}, \quad (4)$$

де p_i – кількість правильно виконаних операцій студентом;

p – кількість операцій, які повинні бути виконані;

N – кількість студентів, що виконували роботу.

Далі, за результатами поелементного та поопераційного аналізу, знаходимо середній бал контрольної та експериментальної груп на вхідному та підсумковому етапі експерименту. За значеннями коефіцієнтів $\bar{p}$ для ЕГ та КГ визначаємо коефіцієнт ефективності запропонованої розробки за формулою 4.

$$\eta_{\bar{p}} = \frac{\bar{p}_e}{\bar{p}_k} \quad (5)$$

Отримані дані заносимо у таблицю 3.5.

Таблиця 3.3 – Визначення коефіцієнту ефективності запропонованої розробки.

Етап експерименту	Поелементний аналіз		η	Поопераційний аналіз		η
	ЕГ	КГ		ЕГ	КГ	
ВК	0,443	0,421	1,05	0,473	0,452	1,04
ПК	0,636	0,502	1,16	0,709	0,594	1,19

Проаналізувавши динаміку коефіцієнтів повноти засвоєння теоретичних знань (представлена на Рисунку 3.1) та динаміку коефіцієнтів повноти виконання практичних операцій (представлена на Рисунку 3.2). На обох графіках чітко видно, що значення коефіцієнтів збільшуються з часом як для ЕГ, так і для КГ, що свідчить про позитивні зрушення у навчальній діяльності студентів загалом. Проте, суттєво більший приріст та вищі підсумкові значення в

Експериментальній Групі (ЕГ) порівняно з Контрольною Групою (КГ) підтверджують ефективність запропонованої методики.

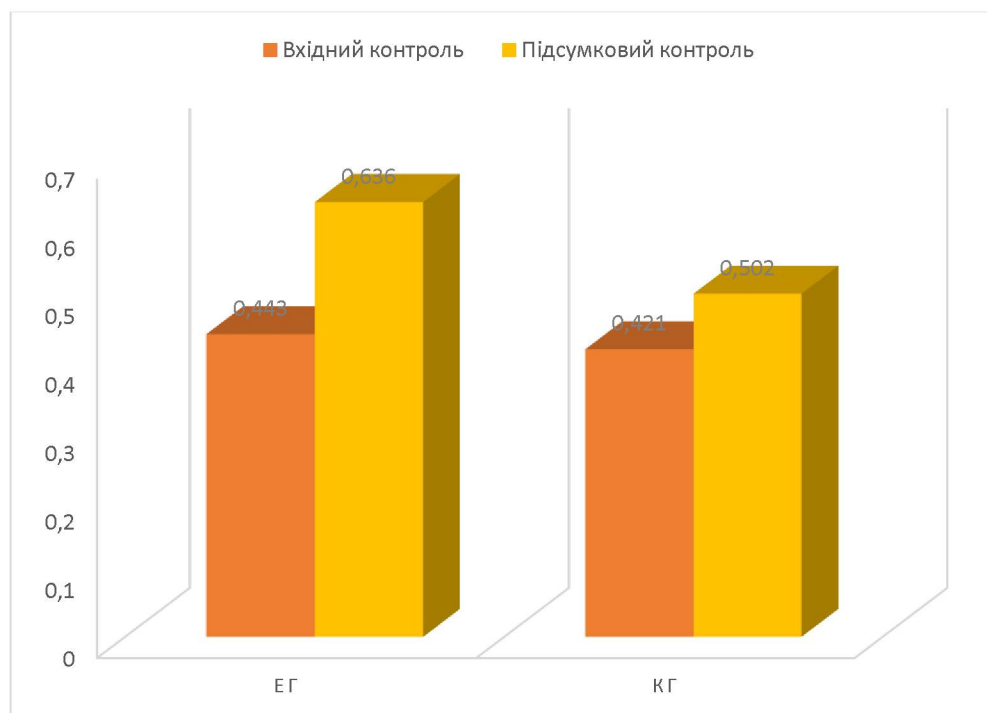


Рис. 3.1 – Динаміка коефіцієнтів повноти засвоєння знань.

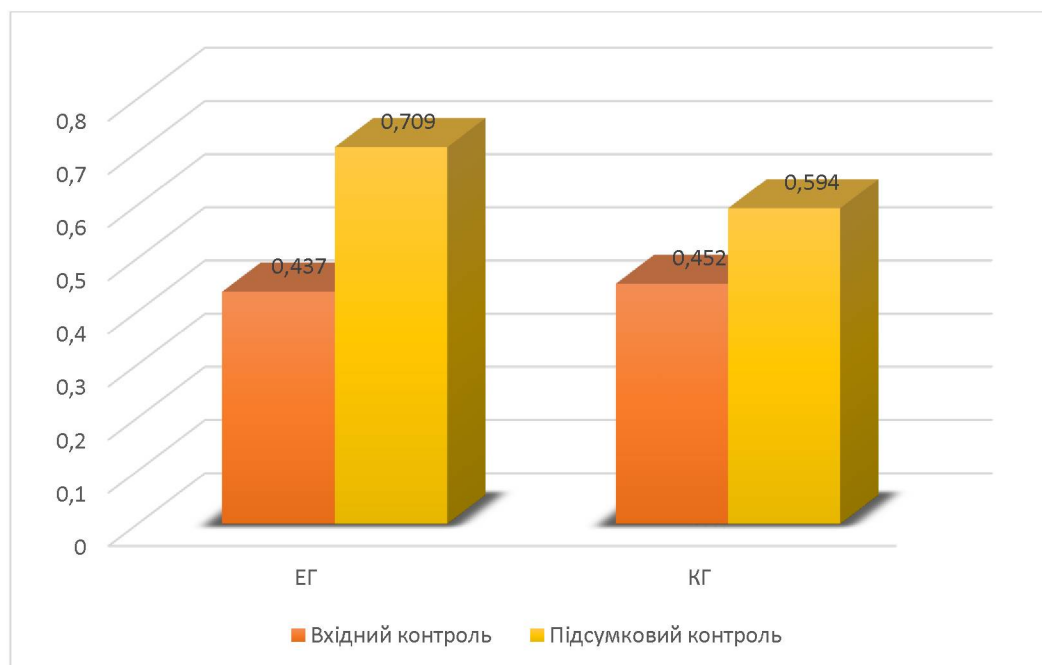


Рис. 3.2 – Динаміка коефіцієнтів повноти виконання операцій

Важливо, що різниця підсумкових балів у студентів ЕГ та КГ не носить випадкового характеру, а свідчить про цілеспрямовану та ефективну експериментальну роботу. Впровадження методики, орієнтованої на проєктний підхід та безпосереднє використання Unity/C# для інтеграції ІІІ та PCG, забезпечило кращу якість засвоєння матеріалу та формування практичних навичок.

Висновки до третього розділу

Таким чином, для проведення педагогічного експерименту було використано метод контрольних зрізів та поопераційного аналізу, який дозволив здійснити кількісний та якісний аналіз результатів дослідження. Для об'єктивної оцінки ефективності розробленої методики навчання розробки ігрових додатків з елементами штучного інтелекту, студенти були поділені на експериментальну (ЕГ) та контрольну (КГ) групи, які були рівнозначні за рівнем підготовки на попередніх етапах навчання.

Аналіз отриманих результатів свідчить про те, що впровадження розробленої методики у навчальний процес підготовки студентів спеціальності «Комп'ютерна інженерія» сприяє підвищенню їхнього інтересу до навчання, інтенсивному розвитку практичних навичок, необхідних для ООП-проєктування та інтеграції AI, а також формуванню відкритої, проєктно-орієнтованої освітньої системи. Це, у свою чергу, позитивно впливає на зростання якості знань і компетентностей студентів у галузі цифрових технологій та розробки ігрових додатків з елементами штучного інтелекту, що експериментально підтверджує ефективність запропонованої методики.

ВИСНОВКИ

У магістерському дослідженні обґрунтовано ефективність застосування методики навчання студентів фахових коледжів розробки ігрових додатків з елементами штучного інтелекту у процесі їх професійної підготовки і запропоновано нове вирішення щодо технології використання кросплатформного ігрового рушія та інструментарію машинного навчання (ML-Agents) для створення інтерактивних AI-систем.

1. Проаналізовано можливості сучасних ігрових рушіїв та інструментів для створення ігрових додатків з елементами штучного інтелекту та обґрунтовано доцільність розробки методики їх створення засобами Unity. Проведено аналітичний огляд сучасних ігрових рушіїв (Unreal Engine, Unity) та інструментів, що застосовуються для розробки ігрових додатків з елементами ШІ. Визначено, що Unity є найбільш доцільним вибором для освітнього процесу завдяки його інтуїтивно зрозумілому візуальному середовищу, високій кросплатформенній підтримці та використанню мови C#, що має нижчий поріг входження порівняно з C++. Обґрунтовано, що ігровий рушій слугує платформою, яка абстрагує складні завдання (рендеринг, фізика), дозволяючи студентам зосередитися на ключових компонентах ШІ. Проаналізовано класичні алгоритми, такі як Шум Перліна та Алгоритм Diamond-Square, як основу для процедурної генерації контенту (PCG), що є невіддільною частиною динамічних AI-світів. Це обґрунтовує потребу в методиці навчання студентів, орієнтованій на Unity.

2. Розкрито технологію використання середовища Unity та елементів штучного інтелекту у процесі навчання студентів фахових коледжів у процесі розробки ігрових додатків та розроблено методику їх проєктування. Досліджено методичні аспекти використання Unity та елементів ШІ, що реалізуються через виконання комплексного навчального проєкту. Розроблено методику проєктування ігрових додатків, яка включає послідовне освоєння ключових

модулів. Методика дозволяє формувати у студентів практичні навички роботи з об'єктно-орієнтованим програмуванням (ООП), створюючи масштабовану структуру класів (LivingEntity, IDamageable). Методика реалізації моделей ІІІ базується на комплексному підході, що поєднує:

- Використання вбудованої системи NavMesh для пошуку шляху NPC (Pathfinding).
- Застосування Штучної нейронної мережі (з топологією 12-12-12-6) та Генетичного алгоритму для навчання адаптивної поведінки ботів (через Unity ML-Agents).

Розроблена методика навчання включає чотири ключові етапи: Формування навичок ООП:

- Навчання проєктуванню складної, масштабованої структури класів та інтерфейсів.
- Освоєння процедурного генерування (PCG): Практична реалізація алгоритмів генерації карт та їх логічної валідації.
- Інтеграція ігрового ІІІ: Освоєння вбудованих інструментів Unity (NavMesh) та власних адаптивних рішень (генетичний алгоритм).
- Професійне використання інструментарію: Набуття навичок роботи з Unity Editor та Visual Studio.

3. Експериментально перевірено ефективність методики навчання студентів фахових коледжів розробці ігрових додатків з елементами штучного інтелекту на основі використання Unity. Проведено педагогічний експеримент на базі ВСП «Фаховий коледж економіки, права та інформаційних технологій ЗУНУ» з метою перевірки ефективності розробленої методики навчання. Студентів поділено на експериментальну (ЕГ) та контрольну (КГ) групи з рівними початковими знаннями. ЕГ навчалась за розробленою методикою, а КГ — за традиційною. Результати експерименту, отримані через поопераційний аналіз та кількісне порівняння коефіцієнтів засвоєння, підтвердили, що використання запропонованої методики у навчальному процесі значно підвищує

інтерес студентів до роботи, їхню залученість та якість здобутих знань і практичних навичок. Це свідчить про те, що розроблена методика є ефективним інструментом для підготовки кваліфікованих фахівців у галузі розробки ігрових додатків з інтегрованими системами ІІІ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bates, B. (2024). *Game Design* (2-ге вид.). Course Technology PTR. 377 p.
2. Bethke, E. (2023). *Game Development and Production*. Texas: Wordware Publishing, Inc. 432 p.
3. Brathwaite, B., та Schreiber, I. (2019). *Challenges for Game Designers*. Charles River Media. 347 p.
4. Chandler, H. M. (2019). *The Game Production Handbook* (2-ге вид.). Hingham, Massachusetts: Infinity Science Press. 442 p.
5. Heaton, R. (2018). *The Wavefunction Collapse Algorithm explained very clearly*. [Електронний ресурс]. URL: <https://robertheaton.com/2018/12/17/wavefunction-collapse-algorithm/> (Дата звернення: 05.12.2025).
6. Huttar, L. (б.д.). *Learning how Perlin noise works*. [Електронний ресурс]. URL: <http://www.huttar.net/lars-kathy/graphics/perlin-noise/perlin-noise.html> (Дата звернення: 05.12.2025).
7. McGuire, M. (2019). *Creating Games: Mechanics, Content, and Technology*. Wellesley, Massachusetts: A K Peters. 500 p.
8. McShaffry, M. (2019). *Game Coding Complete*. Hingham, Massachusetts: Charles River Media. 754 p.
9. Moore, M. E., та Novak, J. (2015). *Game Industry Career Guide*. Delmar Cengage Learning. 108 p.
10. Oxland, K. (2024). *Gameplay and Design*. Addison Wesley. 349 p.
11. Perry, D. (2008). *David Perry on Game Design: A Brainstorming Toolbox*. Rusel DeMaria. 304 c.
12. Rollings, A., та Adams, E. (2023). *On Game Design*. New Riders Publishing. 648 p.

13. Salen, K., та Zimmerman, E. (2015). *The Game Design Reader: A Rules of Play Anthology*. MIT Press. Бартенєва І. О. Педагогіка вищої школи : Навчальний посібник. Одеса: ПДПУ імені К. Д. Ушинського, 2002. – 344 с.
14. Salen, K., та Zimmerman, E. (2023). *Rules of Play*. MIT Press. 688 p.
15. Wong, J. (2014). *Metaballs and Marching Squares*. [Електронний ресурс]. URL: <http://jamiewong.com/2014/08/19/metaballs-and-marching-squares/> (Дата звернення: 05.12.2025).
16. Дудка В. В. Формування у студентів умінь застосовувати комп'ютерні редактори в майбутній професійній діяльності : автореферат дис. на здобуття наук. ст. кан-та пед. наук : спец. 13.00.04 «Теорія і методика професійної освіти» / В.В. Дудка – Київ, 1996. – 22 с.
17. Завгородня Т.К., Стражнікова І.В. Методологічні засади педагогічних досліджень: навчально-методичний посібник. Івано-Франківськ, 2021. 120 с
18. Крушельницька О.В. Методологія та організація наукових досліджень: навч. посіб. Київ: Кондор, 2003. 192 с.
19. Підласий І.П. Діагностика і експертиза педагогічних проектів: навч. посіб. Київ: Україна, 1998. 343 с