Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент групи КН-21-1* Підпис *Іван БОЖИК*
Група виконавця Ім'я, ПРІЗВИЩЕ

Керівник: *д.т.н., проф. каф. КН* Підпис *Едуард МАНЗЮК*
Науковий ступінь, посада Ім'я, ПРІЗВИЩЕ

Нормоконтроль: *к.т.н., доц. каф. КН* Підпис *Руслан БАГРІЙ*
Науковий ступінь, посада Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
зав. кафедри КН, д.т.н., професор Підпис *Олександр БАРМАК*
Ім'я, ПРІЗВИЩЕ

16 *06* 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор Олександр БАРМАК
«*10*» *02* *2025* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників»*

2. Завдання видано студенту *Івану Божику*
(Ім'я, прізвище)

3. Керівник роботи *Професор кафедри КН Едуард МАНЗЮК*
(посада, ім'я, прізвище)

4. Затверджено наказом університету від «*07*» *02* *2025* р. № *23*

5. Дата видачі завдання студенту: «*10*» *02* *2025* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета кваліфікаційної роботи бакалавра – розроблення методу процедурної генерації ігрового контенту для 2D-платформера з інтелектуальним керуванням противників.

Щоб досягти мети кваліфікаційної роботи, потрібно виконати кілька завдань: спочатку провести аналіз методів процедурної генерації та способів реалізації ігрового штучного інтелекту, далі спроєктувати власний метод генерації рівнів і систему поведінки противників, після чого реалізувати все це у вигляді настільного застосунку за допомогою Unity, а на завершення – провести експериментальне тестування створеної програми.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником, складання календарного графіка виконання роботи*	*січень 2025*	Виконано
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*лютий 2025*	Виконано
3	*Проектування та розробка методу, інтерфейсу користувача, вибір засобів реалізації програмного забезпечення*	*березень 2025*	Виконано
4	*Створення та тестування програмного забезпечення*	*квітень 2025*	Виконано
5	*Написання пояснювальної записки, урахування зауважень керівника, оформлення згідно вимог*	*травень 2025*	Виконано
6	*Розробка презентаційних матеріалів та попередній захист кваліфікаційної роботи*	*травень 2025*	Виконано
7	*Отримання відгуку керівника, рецензії, перевірка на плагіат, нормоконтроль*	*червень 2025*	Виконано
8	*Підготовка до захисту та захист кваліфікаційної роботи бакалавра*	*червень 2025*	Виконано

Виконавець: *студент групи КН-21-1* (Група виконавця) _____ (Підпис) *Іван БОЖИК* (Ім'я, ПРІЗВИЩЕ)

Керівник: *д.т.н., проф. каф. КН* (Науковий ступінь, посада) _____ (Підпис) *Едуард МАНЗЮК* (Ім'я, ПРІЗВИЩЕ)

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-21-1 Іван БОЖИК*

Керівник кваліфікаційної роботи бакалавра: *доктор технічних наук, професор кафедри КН Едуард МАНЗЮК*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
50	33	3	53	2

Мета кваліфікаційної роботи бакалавра полягає у підвищенні варіативності ігрового процесу шляхом розробки методу процедурної генерації ігрового контенту з інтелектуальною системою керування поведінкою противників для 2D-платформера.

Розроблений застосунок дає змогу автоматично створювати рівні платформи із варіативними елементами середовища та забезпечує противникам інтелектуальну поведінку, що підвищує динамічність ігрового процесу та робить гру цікавішою для гравця. Реалізована система має потенціал для подальшого розширення механік гри та ускладнення поведінки противників.

Ключові слова: процедурна генерація, 2D-платформер, керування противників, Unity, C#.

Виконавець: *студент групи КН-21-1* (Група виконавця) _____ (Підпис) *Іван БОЖИК* (Ім'я, ПРІЗВИЩЕ)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
NPC	Не ігровий персонаж
ШІ	Штучний інтелект
ІС	Інформаційна система
ІТ	Інформаційні технології
КРБ	Кваліфікаційна робота бакалавра
РПГ	Рольова гра
ПК	Персональний комп’ютер
BSP	Розділення бінарного простору
FSM	Скінчені автомати станів

Вступ

Кваліфікаційна робота бакалавра присвячена розробці методу процедурної генерації ігрового контенту для 2D-платформера з інтелектуальною системою керування поведінкою противників та створенню настільного ігрового застосунку на його основі за допомогою рушія Unity та мови програмування C#

Актуальність – Процедурна генерація контенту є важливим напрямом сучасної ігрової розробки, оскільки дозволяє створювати великі та різноманітні світи з мінімальними витратами ресурсів. У поєднанні зі штучним інтелектом противників це підвищує варіативність ігрового процесу та забезпечує більшу цікавість для гравців. Створення ефективного методу процедурної генерації рівнів і розумної поведінки противників є актуальним завданням у контексті розвитку ігрової індустрії.

Об'єкт дослідження – процес процедурної генерації ігрового контенту для 2D-платформерів.

Предмет дослідження – методи та технології створення інтелектуальних систем керування поведінкою противників у процедурно згенерованих рівнях 2D-платформера.

Мета кваліфікаційної роботи бакалавра – підвищення варіативності ігрового процесу шляхом розробки методу процедурної генерації ігрового контенту з інтелектуальною системою керування поведінкою противників для 2D-платформера.

Завданням кваліфікаційної роботи бакалавра є аналіз предметної області процедурної генерації контенту та ігрового ШІ, вивчення існуючих методів генерації рівнів для 2D-платформерів і особливостей побудови ігрового світу. Також дослідити моделі реалізації інтелектуальної поведінки противників, розробити метод процедурної генерації, спроєктувати архітектуру застосунку з інтеграцією систем генерації рівнів і ШІ, реалізувати ігровий застосунок у середовищі Unity, провести його тестування та оцінити ефективність запропонованих рішень.

Розділ 1 Характеристика предметної області: аналіз моделей, методів та реалізацій

1.1 Аналіз інформаційних моделей

Ігрові застосунки за останні роки стали невід'ємною частиною сучасності. Вони захоплюють гравців своїми історіями, викликами, головоломками та чудовою графікою, що дає як можна більше занурення у світ гри [**Ошибка! Источник ссылки не найден.**]. Але ігри не можна назвати звичайною розвагою, їх вплив не обмежується лише проведеним часом, їх можна розгортати на різноманітних платформах, таких як смартфони, ПК та консолі, що може повністю задовольнити потреби у розвагах, освіті та комунікації [2].

Завдяки використанню процедурної генерації віртуальні середовища можуть імітувати складність та випадковість, що існують у реальному світі. Це призводить до більш захопливих та реалістичних ігрових сеттингів, на відміну від статичних, заздалегідь визначених ландшафтів. Це додає шар непередбачуваності, який відображає органічну природу нашого оточення [3].

Також процедурна генерація використовується при створенні різних ефектів у фільмах кіно та телеіндустрії, процедурні методи 2D та 3D застосовуються переважно у візуальних ефектах, де вони допомагають створювати реалістичні симуляції природних явищ, таких як горіння, сніг та інші погодні ефекти, поведінка натовпу чи віртуальні середовища [4].

Процедурні методи генерації ШІ пропонують безмежні можливості дизайну, підвищуючи цінність повторного проходження. Створюючи унікальний ігровий досвід, ШІ підтримує залученість та утримання гравців. Індивідуальний ігровий досвід адаптується до індивідуальних уподобань, покращуючи занурення гравців. Крім того, ШІ спрощує розробку, оптимізуючи ефективність для творців. Інтеграція ШІ в ігровий дизайн стимулює інновації, прокладаючи шлях для нових ігрових можливостей та досліджуючи незвідані території в галузі [**Ошибка! Источник ссылки не найден.**].

З кожним роком з’являється все більше ігор із різними жанрами та механіками. Серед найпопулярніших – стратегії, багатокористувацькі ігри та рольові (RPG) [6]. RPG мають піджанри: наприклад, відкритий світ-пісочниця, JRPG – японські RPG із покроковою бойовою системою [7], та souls-like – ігри, що випробовують гравця на логіку, реакцію і стресостійкість, не маючи налаштувань складності [8][9].

Окремо виділяються мобільні ігри, що найчастіше створюються як таймкілери – нескінченні або процедурно згенеровані рівні, як у жанрі "три в ряд" [10][11].

Підсумовуючи вище сказане можна дійти висновку, що ігри мають велике різноманіття цікавих та корисних жанрів, що дає змогу людям кожного віку знайти те що підійде саме їм, та дасть можливість чудово провести свій вільний час.

1.2 Огляд теоретичних підходів до розв’язку подібних задач

Процедурна генерація – це метод автоматичного створення контенту в іграх, що дозволяє розробникам економити час на ручному проєктуванні ландшафтів, рівнів чи об'єктів [12]. Вона використовує алгоритми та математичні функції для швидкого й узгодженого формування ігрового середовища, забезпечуючи нескінченний ігровий досвід.

У Unity цей підхід часто застосовують для генерації рельєфу, рівнів або текстур, що особливо корисно для створення деталізованих світів із мінімальними зусиллями [13].

Є багато алгоритмів процедурної генерації, але найпопулярнішими є такі як шум Перлина, рандомайзер та клітинний автомат.

Шум Перлина це математичний алгоритм генерування процедурної текстури псевдо-випадковим методом. Використовується в комп'ютерній графіці задля збільшення реалізму чи графічної складності поверхні геометричних об'єктів. Також може використовуватися для створення ефектів диму чи туману.

Шум Перлина являє собою градієнтний шум, що складається з набору псевдовипадкових одиничних векторів (напрямів градієнта), розташованих у певних точках простору та інтерпольованих функцією згладжування між цими точками. Для генерації шуму Перлина в одновимірному просторі необхідно для кожної точки цього простору обчислити значення шумової функції, використовуючи напрямок градієнта (або нахил) у зазначеній точці [**Ошибка! Источник ссылки не найден.**].

Рандомайзер визначає шанс на появу того чи іншого елемента на полотні, є найпримітивнішим алгоритмом процедурної генерації, шанс виставляється в діапазоні від 0 до 1, та впливає на частоту появи того чи іншого елементу [15].

Клітинний автомат це дискретна математична модель, його основою є N комірок та їх правил переходу, початковий розподіл клітинок формує стартову конфігурацію автомату [16]. На кожному кроці алгоритму одночасно всі клітинки переходять у новий стан згідно правил, тип сусідства клітинок визначає як і які клітинки будуть впливати на зміну її стану. Одними із найпоширеніших типів сусідства є околиці Мура та Неймана.

Околиця Мура охоплює вісім клітинок навколо центральної, включаючи як прямі, так і діагональні напрямки. Завдяки цьому вона дозволяє враховувати ширший контекст оточення, що корисно при генерації природніших або менш суворо структурованих форм, зокрема для плавного переходу між типами поверхні чи при згладжуванні шуму [**Ошибка! Источник ссылки не найден.**].

Околиця Неймана включає чотири клітинки, розташовані зліва, справа, зверху і знизу від поточної. Вона використовується в задачах, де важливо обмежити взаємодію лише прямими напрямками, наприклад, при генерації коридорів, хвильовому поширенні або контролі зв’язності простору без діагоналей [18].

Карти Дейкстри – це інструмент, що базується на алгоритмі Дейкстри для пошуку найкоротших шляхів, який активно використовується у процедурній генерації ігрових рівнів. Вони дозволяють створити сітку, де кожна клітинка містить відстань до заданої цілі, що дає змогу формувати логічно пов’язану

структуру світу. Такий підхід застосовується для з’єднання кімнат у підземеллях, розміщення об’єктів з урахуванням доступності, а також для навігації персонажів та ворогів, які можуть орієнтуватися за даними карти без потреби у складніших алгоритмах. Карти Дейкстри є універсальним і ефективним способом створення взаємопов’язаного простору в процедурно згенерованому середовищі [19].

Розділення бінарного простору (BSP) – це метод рекурсивного поділу простору на підобласті, який застосовується для створення кімнат у підземеллях. Після поділу кожна область може стати кімнатою, а спеціальний алгоритм з'єднує їх коридорами, що забезпечує логічну ігрову структуру [20].

Також не можна забувати про штучний інтелект ворогів в іграх, це дає змогу краще погрузитись у ігровий процес, та не знудьгувати під час гри.

Штучний інтелект має велику роль під час розробки ігор, може покращувати багато ігрових аспектів, до прикладу поведінку NPC, якусь ігрову механіку чи створення процедурного наповнення [**Ошибка! Источник ссылки не найден.**].

NPC це неігровий персонаж, яким керує ШІ, це може бути як друг так і ворог, вони можуть бути як і важливими у концепті гри так і другорядним аспектом, але при цьому вони все одно створюють атмосферу наповненості світу, що дає змогу гравцю як можна краще зануритись у світ [**Ошибка! Источник ссылки не найден.**].

Деколи при створені поведінки ворогів застосовують FSM (скінченні автомати станів), це коли об’єкт перебуває в якомусь стані, до прикладу патрулювання чи атака, може переходити між станами в залежності від умов середовища [23]. Логіка переходів між станами керується правилами, що визначають реакцію ворога на деякі події це (Rule-Based AI), до прикладу якщо при виявлені прірви ворог змінює напрям руху, а при наближені гравця розпочинає переслідування яке призведе до подальшої атаки [24].

Також є такий аспект поведінки як (Reactive AI) він забезпечує швидку реакцію на наявні в даний момент правила, без ускладненого планування наперед, це досить непогане рішення для простого штучного інтелекту в іграх [25].

Дерево поведінки це формалізованіший підхід побудови поведінки мобів. Його особливість полягає в тому, що всі стани персонажа організовані у вигляді структури, що гілкується зі зрозумілою ієрархією. Дерево поведінки містить у собі всі можливі стани, у яких може бути моб. Коли в грі відбувається якась подія, ШІ перевіряє, в яких умовах знаходиться NPC, і перебирає всі стани у пошуках того, що підійде для тієї чи іншої ситуації [26].

Генетичні алгоритми (Genetic Algorithms) – це методи, натхнені принципами еволюції та природного відбору, які можуть бути використані для створення адаптивних поведінок у ігрових персонажів чи середовищах. Генетичні алгоритми працюють шляхом популяції, які еволюціонують через механізми відбору, схрещування та мутації. Ці алгоритми можуть бути використані для вирішення складних задач, таких як оптимізація маршрутів, розробка тактик ворогів чи навіть визначення найкращих стратегій поведінки NPC у різних ситуаціях. Цей підхід дозволяє створювати системи, які можуть адаптувати свою поведінку на основі результатів попереднього досвіду та змін у грі [**Ошибка! Источник ссылки не найден.**].

При створені ігор процедурною генерацією часто використовують таке поняття як чанк, це невеликі ділянки ігрового світу, які створюються, обробляються та зберігаються окремо, що дає змогу зменшити затрати ресурсів на обробку катри гри, гра завантажує та обробляє тільки ті чанки які є поруч з гравцем, у той час чанки які є віддалені від гравця “засинають”, задля економії ресурсів, що дає змогу не втрачати в продуктивності та зберігати стабільний фрейм-рейт [28].

В загальному є дуже багато різних підходів для створення процедурно згенерованих світів, та багато методів керування поведінки NPC, це дає змогу розробникам знайти саме свою комбінацію засобів для розробки продукту.

1.3 Аналіз існуючих програмних засобів та наукових рішень

Наразі є багато ігор під час створення яких використовувалась процедурна генерація, разом із штучним інтелектом ворогів на основі правил та обмежень, і деякі з цих ігор стали справді успішними [29], до прикладу одна з таких ігор це Rouge Legacy [30], у ній кожен наступний запуск створює повністю різні схеми замку, майже не можливе створення ідентичної комбінації розміщення кімнат, також є система сімейного дерева, що дає змогу після втрати одного героя, грати за його наслідника, і так до безкінечності, чи поки не буде пройдена гра. На рисунку 1.1 зображено реалізацію поколінь.

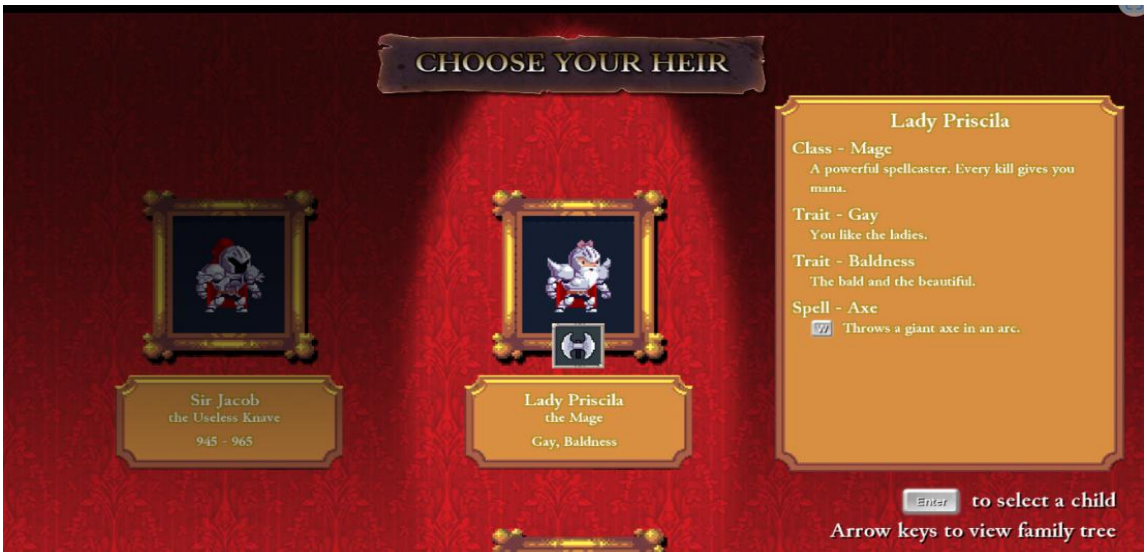


Рисунок 1.1 – Вибір наслідника у грі Rouge Legacy [31]

Кожен наступний наслідник складається із різних параметрів, таких як клас, магія, та якісь особливості до прикладу кольорова сліпота, боязнь курей і тому подібне. На рисунку 1.2 відображено згенеровану мапу в грі Rouge Legacy

Рисунок 1.2 – Згенерована мапа у грі Rouge Legacy [32]

У цій грі генерація кімнат на мапі працює так, світ складається з кімнат, які створені заздалегідь силами дизайнерів. Під час запуску гри, вона випадково вибирає шаблони кімнат із великої бібліотеки готових приміщень, після чого поєднує їх між собою за заздалегідь визначеними правилами, щоб була гарантована проходимість кімнати, це називається гібридним підходом, при поєднанні ручного дизайну та випадкового розташування [33].

Spore це ще один ігровий продукт який використовує процедурну генерацію, до прикладу у цій грі є декілька планет на яких можна розпочати гру, усі вони створені за допомогою генерації [34]. На рисунку 1.3 відображено планети які були згенеровані.

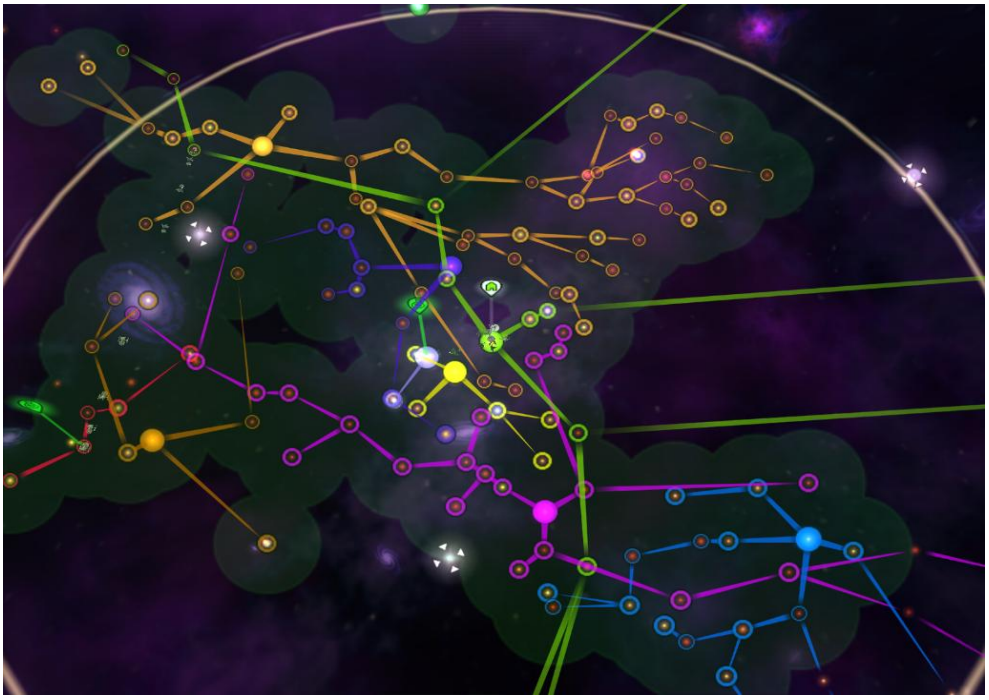


Рисунок 1.3 – Згенеровані планети [35]

Кожна з планет повністю унікальна та доступна для візиту на космічному кораблі, у кожної своя екосистема та наповнення, яке також є згенерованим, планети населяють різноманітні створіння, зі своїми унікальними параметрами та виглядом.

На планеті у випадкових місцях створюються гнізда які населяють створіння [36], їх поведінка також визначається автоматично при створенні, вони можуть бути як агресивними так і дружелюбними чи нейтральними, в залежності від їх настрою буде вирішено чи нападуть вони чи будуть намагатись подружитись [37].

Ще одним дуже популярним ігровим застосунком є Teraria [38], ця гра по принципу генерації світу є сильно наближена до теми КРБ, при генерації світу використовується процедурна генерація методом Перлина, та автоматичне розташування противників з інтелектуальною поведінкою [39]. На рисунку 1.4 відображено один із згенерованих світів.

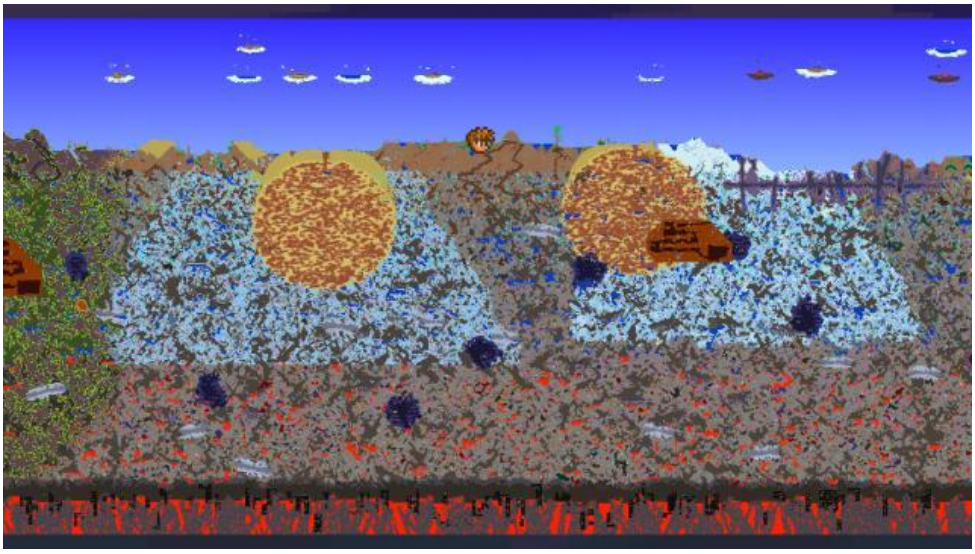


Рисунок 1.4 – Згенерований світ терарії [40]

Загалом є дуже багато різних застосунків які використовують процедурну генерацію за допомогою багатьох методів, так же у більшості ігрових програмних продуктів реалізовано інтелектуальну поведінку противників основану на обмеженнях чи навчанні.

1.4 Мета, задачі та вимоги до реалізації інформаційної системи

Метою кваліфікаційної роботи бакалавра є підвищення варіативності ігрового процесу шляхом розробки методу процедурної генерації ігрового контенту з інтелектуальною системою керування поведінкою противників.

Задачами кваліфікаційної роботи бакалавра є:

1. Проведення аналізу предметної області процедурної генерації контенту та ігрового штучного інтелекту.
2. Аналіз існуючих методів процедурної генерації рівнів для 2D-платформерів.
3. Визначення особливостей побудови ігрового світу з процедурною генерацією.
4. Дослідження моделей та методів реалізації інтелектуальної поведінки ігрових противників.

5. Розроблення методу процедурної генерації ігрового контенту для 2D-платформера.

6. Проєктування архітектури застосунку із системами процедурної генерації та керування поведінкою противників.

7. Розроблення настільного ігрового застосунку з використанням рушія Unity і мови програмування C#.

8. Проведення тестування створеного застосунку.

9. Оцінювання ефективності розробленого методу процедурної генерації та роботи інтелектуальної системи керування противниками.

Основними вимогами для розробки методу є визначення функціональної структури системи, розробка архітектури ШІ та створення діаграм для відображення основних елементів роботи методу, визначити спеціалізовані програмні компоненти які потрібні для розробки методу.

Також потрібно визначити шляхи дослідження та засоби для розробки методу, реалізувати програмні складові системи, протестувати розроблену систему, створити інструкцію користувача та визначити результати досліджень в межах методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників.

Загалом наведених вище аспектів достатньо для розробки концепту методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників, у подальшому функціонал та можливості можуть бути розширені та покращені.

Розділ 2 Метод процедурної генерації з інтелектуальною системою керування поведінкою противників

2.1 Метод процедурної генерації ігрового контенту

В межах КРБ розроблено схему методу процедурної генерації ігрового контенту, схема дозволить краще зрозуміти суть роботи методу, та покращить його розуміння. На рисунку 2.1 зображено схему методу.

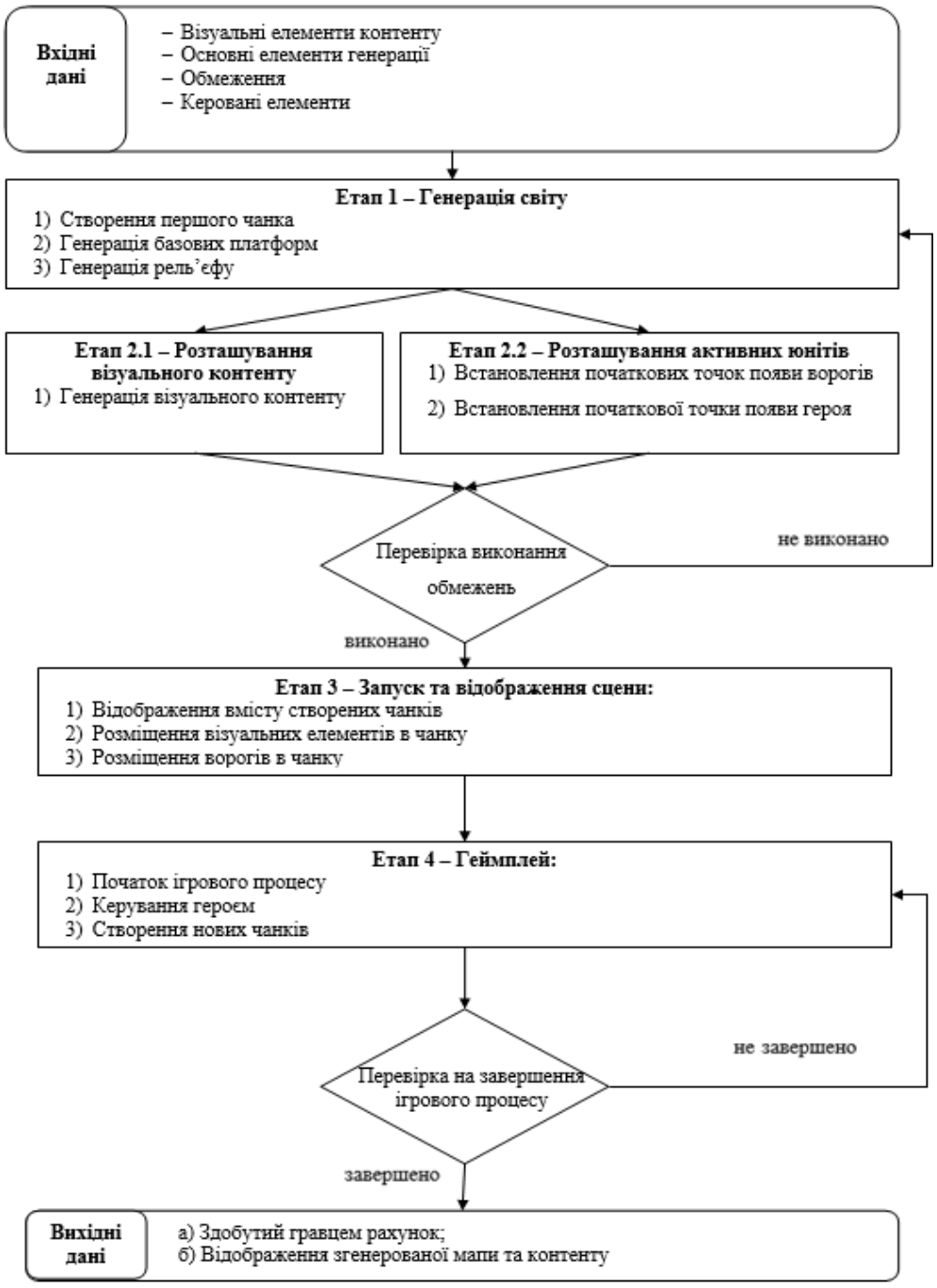


Рисунок 2.1 – Схема методу процедурної генерації ігрового контенту

У вхідних даних містяться основні та візуальні елементи генерації, вони відповідають за візуальну складову рівня, обмеження це набір правил які дають змогу правильно згенерувати світ, щоб була можливість пересуватися та для візуально-естетичного відображення, вороги та герой це елементи які взаємодіють між собою.

Етап 1 – для початку створюється стартовий чанк в якому генерується вміст, який складається із платформ та рель'єфу, платформи це елементи по яким можна пересуватись, вони містять колайдери, рель'єф створює перешкоди які буде потрібно оминати.

Етап 2.1 – генерує візуальний контент, такий як свічки, шафи чи статуї, які розміщуються згідно обмежень.

Етап 2.2 – визначає точки в чанках де будуть з'являтись противники, також згідно обмежень, щоб запобігти перевищенню кількості ворогів на один чанк, також відповідає за початкову позицію героя.

Перевірка на виконання обмежень – перевіряє чи виконані умови обмежень, якщо так то розпочинається 3 етап якщо ні то повернення до 1 етапу.

Етап 3 – відповідає за запуск сцени та відображення результату генерації, розміщує ворогів та візуальні елементи на сцені згідно визначених раніше точок, також відображає згенеровані елементи рівня.

Етап 4 – розпочинає ігровий процес, запускає інтелектуальну систему ворогів, дає змогу керувати героєм та розпочинає створення чанків в залежності від точки перебування героя, що забезпечує безкінечну генерацію рівня, якщо ігровий процес не завершено то він продовжується, якщо ж завершено то можна переходити до вихідних даних.

Вихідні данні – показують фінальний результат згенерованого контенту та кінцевий рахунок гравця.

Узагальнюючи дана схема методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників показує усі основні алгоритми роботи методу.

2.2 Архітектура інтелектуальної поведінки противників

Для реалізації навчального штучного інтелекту для інтелектуальної поведінки ворогів у проєкті використано фреймворк Unity ML-Agents, який дає змогу створювати агентів, здатних навчатися шляхом взаємодії з ігровим середовищем. Основу архітектури ШІ становить штучна нейронна мережа, яка в процесі тренування формує політику дій ворога на основі отриманих спостережень і зворотного зв'язку з оточення. На рисунку 2.2 зображено схему структури навчання ворога.

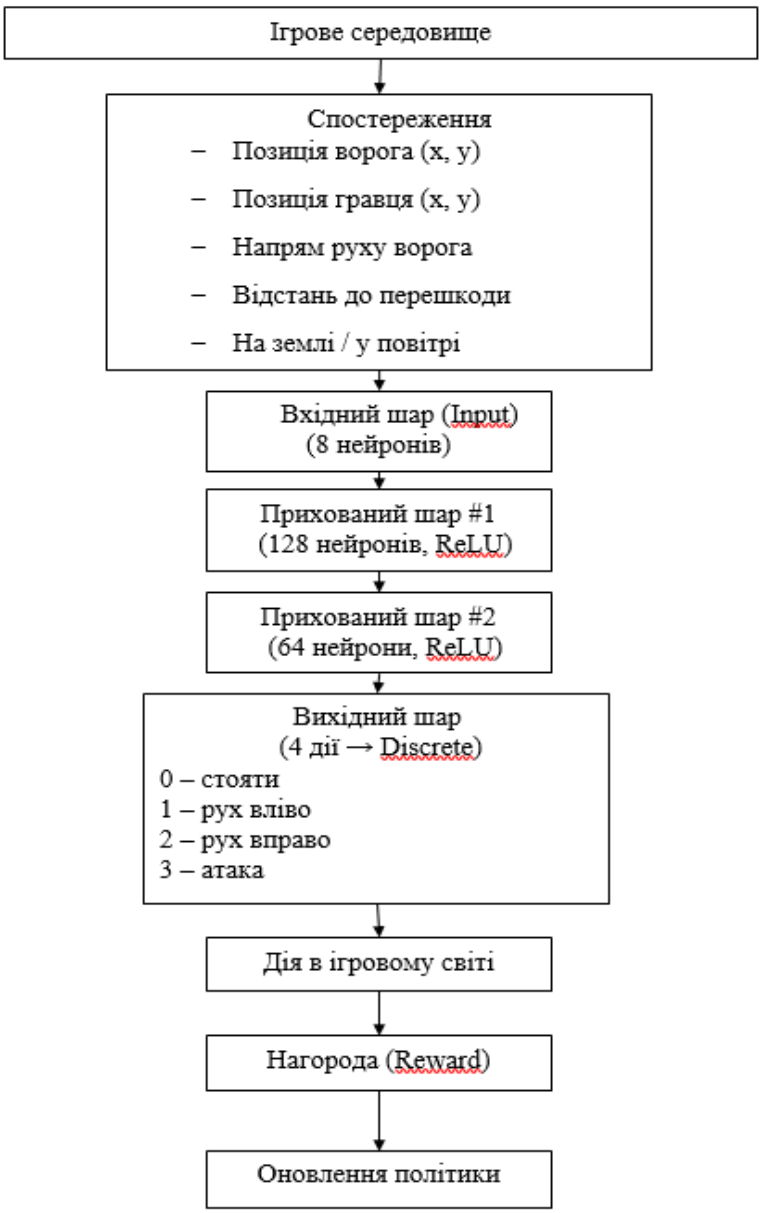


Рисунок 2.2 – Схема структури навчання

Вхідний шар мережі опрацьовує вісім параметрів, які описують ситуацію з точки зору агента. До них належать поточні координати ворога та гравця, напрямок руху ворога, відстань до найближчої перешкоди, а також інформація про те, чи перебуває ворог на землі, чи у повітрі. Це дозволяє агенту оцінити своє положення, визначити напрям для руху або атаки, уникати зіткнень і враховувати гравітаційний стан під час прийняття рішень.

Нейронна мережа має два приховані шари: перший з них містить 128 нейронів, а другий – 64. Для обох використовується функція активації ReLU (Rectified Linear Unit), яка забезпечує ефективне навчання складних залежностей та стабільну роботу моделі. Ці шари перетворюють вхідні дані у внутрішнє представлення, яке дозволяє агенту виробити стратегію поведінки.

На виході мережі формується одне з чотирьох можливих рішень: залишитися на місці, рухатись ліворуч, рухатись праворуч або атакувати. Агент обирає дію з максимальним очікуваним результатом відповідно до поточного стану середовища.

Для навчання нейронної мережі застосовується алгоритм Proximal Policy Optimization (PPO) – один з найефективніших методів глибокого навчання з підкріпленням. Його переваги полягають у стабільності, контролі за оновленням політики (що запобігає її “зламу” під час тренування) та здатності працювати як з дискретними, так і з неперервними просторами дій. PPO добре масштабується і широко застосовується у складних середовищах, таких як симуляції поведінки в іграх.

У першій частині схеми – спостереження агента, які включають вісім параметрів: координати ворога й гравця, напрямок руху, відстань до перешкоди та інформацію про стан на землі чи в повітрі. Ці параметри надходять на вхідний шар нейронної мережі, де їх опрацьовує вісім нейронів.

Далі сигнал передається до двох прихованих шарів із 128 та 64 нейронами відповідно, які використовують функцію активації ReLU для забезпечення нелінійності та стабільного навчання.

Вихідний шар генерує одну з чотирьох дискретних дій: стояти на місці, рухатись вліво, рухатись вправо або атакувати. Вибір дії базується на політиці, сформованій під час навчання, й визначає поведінку агента у грі.

Після виконання дії середовище генерує нагороду (reward), яка сигналізує агенту, наскільки вдалою була обрана дія. Цей зворотний зв’язок використовується алгоритмом PPO для оновлення політики – тобто адаптації майбутньої поведінки агента.

Завдяки такій архітектурі ворог у грі здобуває гнучку та адаптивну модель поведінки, яка не ґрунтується на наперед прописаних станах або правилах, як це відбувається у класичних підходах типу FSM. Замість цього, модель самостійно формує стратегію дій у процесі тренування, що дає змогу реагувати на динамічні зміни в оточенні, ефективно взаємодіяти з гравцем і демонструвати більш природну та непередбачувану поведінку. Такий підхід не лише покращує реалізм ігрового процесу, але й відкриває можливості для подальшого розвитку штучного інтелекту.

2.3 Вхідні дані для процедурної генерації

Вхідними даними для системи є спрайти, вони дають змогу візуалізувати роботу генерації світу, на рисунку 2.3 відображено усі вхідні спрайти які потрібні для візуалізації згенерованого світу.

Рисунок 2.3 – Спрайти для генерації

Для створення підлоги та підйомів зі спусками було використано чотири спрайти [41], їх виділено на рисунку 2.4, також із цього ж джерела було взято усі декоративні елементи.

Рисунок 2.4 – Основні елементи землі

Також для створення перешкод було використано спрайт вогню [42].

Ще використано платформу по яких проводяться стрибки [43].

Для створення героя використано спрайти із розкадровкою для створення анімацій, базовий спрайт зображений на рисунку 2.5

Рисунок 2.5 – Спрайт головного героя [44]

Та останнім використаним спрайтом який бере участь у генерації світу та ігровому процесі це ворог, він також має розкадровки для анімацій, його показано на рисунку 2.6.

Рисунок 2.6 – Спрайт ворога [45]

Загалом цих елементів візуалізації цілком достатньо для початку розробки методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників.

2.4 Розробка інформаційної системи

2.4.1 Функціональна структура системи

Для методу процедурної генерації ігрового контенту представлено схему функціональної структури системи, у якій показано етапи послідовності роботи методу, її показано на рисунку 2.7.

Підсистема керування грою – визначає початкові точки генерації, де буде створено в наступних підсистемах візуалізацію генерації.

Підсистема генерації світу – відповідає за генерацію чанку та його вмісту, такого як підлога, прогалини, платформи, зміни висоти рель’єфу та розміщення декорацій із ворогами.

Підсистема керування гравцем – контролює усіма аспектами які пов’язані із головним героєм, такі як рух, стрибки, атака, та анімація цих аспектів. Також відповідає за пересування камери для динамічного відображення локації, камера слідує за гравцем і визначає зону для генерації нових чанків.

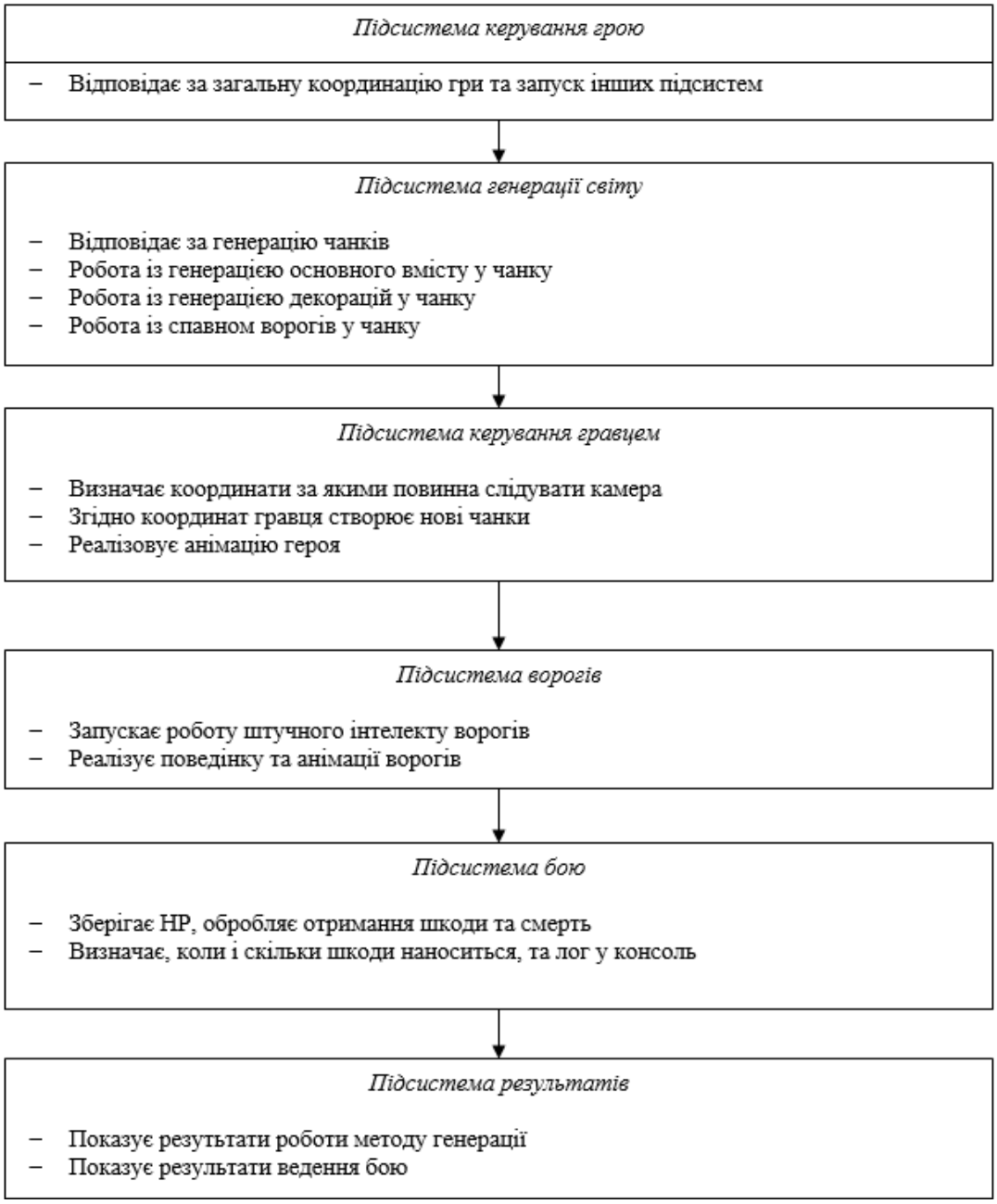


Рисунок 2.7 – Схема функціональної структури системи

Підсистема ворогів – запускає роботу штучного інтелекту кожного створеного моба, що дозволяє їм виконувати такі функції як рух, переслідування, перевірка на прірву чи стіну попереду, задля своєї безпеки, та атака по герою.

Підсистема бою – зберігає стартовий показник HP (Health Point – значення здоров’я) усіх задіяних елементів, таких як герой та вороги, та динамічно оновлює їх в залежності від того ким і кому була нанесена шкода, результат динамічного оновлення показників відображається в логах.

Підсистема результатів – показує фінальні результати як роботи методу генерації так і рахунок якого досягнув гравець, що дає змогу оцінити роботу методу та майстерність користувача.

2.4.2 Проектна архітектура системи та взаємозв'язок компонентів

Для кращого розуміння роботи програмного застосунку змодельовано діаграми варіантів використання та діяльності.

На рисунку 2.8 відображено діаграму варіантів використання.

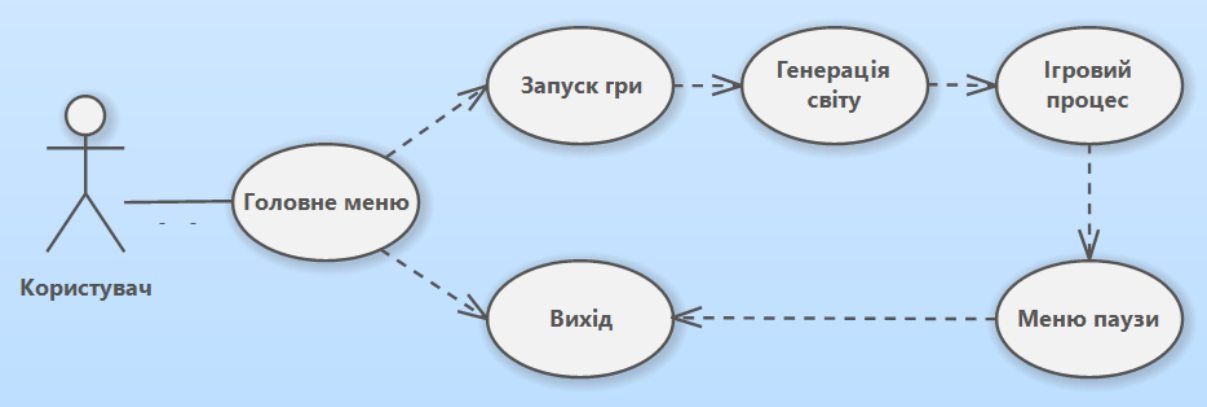


Рисунок 2.8 – діаграма варіантів використання

Головне меню, дає змогу вийти із застосунку, або ж запустити гру, що в свою чергу запустить генерацію світу, після чого буде можливість розпочати ігровий процес, під час якого можна відкрити меню паузи задля виходу.

На рисунку 2.9 зображено діаграму діяльності ШІ ворога.

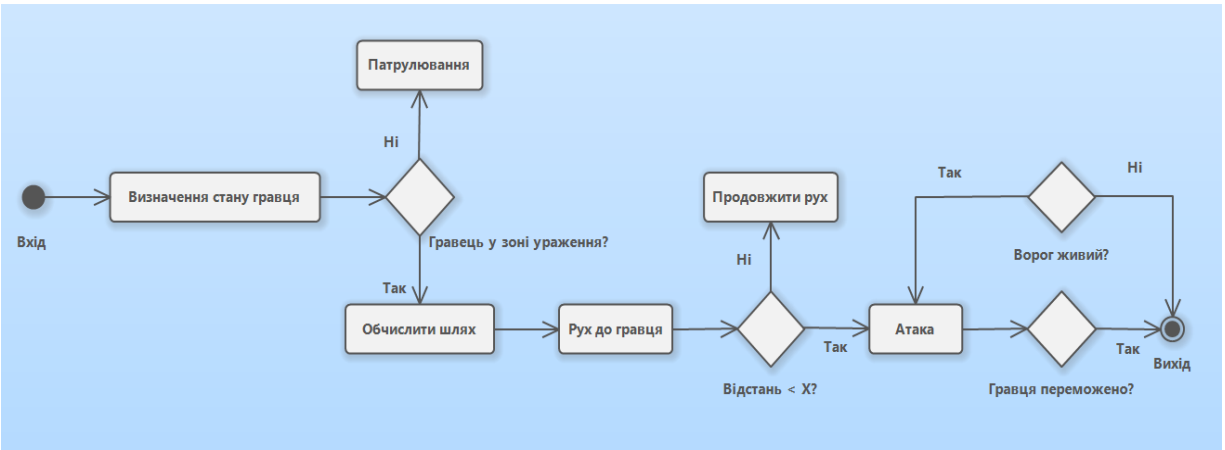


Рисунок 2.9 – діаграма діяльності ворога

Для початку визначається положення гравця до ворога, якщо він не у зоні ураження то ворог просто продовжує патрулювання території, якщо ж у зоні ураження то після обчислення відстані розпочинає рух до гравця, після чого йде перевірка на те чи достатня дистанція для нанесення атаки чи ні, якщо ні то рух продовжується, а коли відстань достатньо мала розпочинається атака, далі йде перевірка на те чи гравця переможено, якщо так то це буде означати кінець гри, якщо ж ні то проводиться перевірка чи живий ворог, якщо так то продовжується атака, якщо ж ні то також кінець гри, але для ворога.

Загалом представлено загальну структуру програмної системи та взаємодію її основних компонентів. За допомогою діаграм варіантів використання та діяльності ворога вдалося наочно показати основні сценарії використання програми та логіку роботи окремих елементів, зокрема штучного інтелекту ворога. Це дало змогу краще зрозуміти загальну архітектуру застосунку, спростити його реалізацію та забезпечити цілісність функціонування системи.

2.4.3 Особливості використання спеціалізованих програмних компонентів

На етапі планування програмної частини проєкту проаналізовано доцільність використання сторонніх або спеціалізованих бібліотек, зокрема рішень для роботи зі штучним інтелектом, розширеною фізикою чи мережею. Однак у межах поставлених завдань було вирішено зосередитися переважно на використанні стандартних бібліотек, що постачаються разом із середовищем розробки Unity та мовою програмування C#. Такий підхід дозволив забезпечити стабільність, сумісність з різними платформами, а також уникнути зайвих залежностей, що могли б ускладнити підтримку проєкту.

Зокрема, передбачалося використання таких бібліотек, як System.Collections [46] та System.Collections.Generic [47] для роботи з колекціями, а також UnityEngine, яка є основною бібліотекою для створення ігрових проєктів у Unity. System.Collections забезпечує базову підтримку переліків та ітераторів, необхідних, наприклад, для корутин. Розширена бібліотека

System.Collections.Generic дає змогу ефективно працювати з типізованими структурами даних, зокрема зі списками, які використовуються для зберігання ігрових об'єктів, точок генерації або маршрутів.

Найширше використання отримала бібліотека UnityEngine, що забезпечує доступ до всіх основних елементів рушія: управління об’єктами сцени, трансформацією, фізикою, анімаціями та обробкою подій. Такий вибір бібліотек був цілком виправданим, оскільки дозволив реалізувати всі необхідні функції проєкту, зберігаючи при цьому простоту архітектури, зрозумілість коду та мінімальні вимоги до системи.

Окрім базових інструментів, для реалізації навчального штучного інтелекту ворогів було інтегровано спеціалізовану бібліотеку Unity.MLAgents [48], що є офіційним розширенням Unity для створення агентів, які навчаються за допомогою алгоритмів навчання з підкріпленням, у межах цього підходу агент навчається шляхом взаємодії з середовищем: виконуючи дії, він отримує винагороди або штрафи залежно від результатів. Метою агента є максимізація сумарної винагороди за епізод – тобто знаходження оптимальної стратегії поведінки. До складу ML-Agents входить декілька важливих компонентів:

– Unity.MLAgents – основний простір імен, що містить класи Agent, BehaviorParameters, DecisionRequester, які використовуються для визначення логіки агентів і способів прийняття рішень.

– Unity.MLAgents.Actuators – модуль, відповідальний за реалізацію дій агентів.

– Unity.MLAgents.Sensors – інтерфейси для збирання спостережень із середовища.

– Unity.MLAgents.Policies – механізми керування політиками, які застосовуються під час тренування та інференсу.

Завдяки цим бібліотекам було реалізовано повноцінну систему поведінки ворогів, яка ґрунтується на гнучкому машинному навчанні, а не на наперед заданих правилах. ML-Agents дозволив проводити тренування агентів за допомогою Python-бібліотек (через окремий навчальний сервер) із використанням

алгоритму PPO (Proximal Policy Optimization), що був обраний за стабільність і високу продуктивність у ігрових середовищах.

Таким чином, використання лише базових бібліотек Unity у поєднанні з ML-Agents дозволило зберегти чисту архітектуру, реалізувати сучасний ШІ та забезпечити масштабованість проєкту без необхідності залучення сторонніх нестандартних рішень.

2.5 Висновки до розділу 2

Під час розробки методу процедурної генерації ігрового контенту з інтелектуальною системою керування ворогами створено набір схем і діаграм, які визначають архітектуру майбутнього застосунку. Серед них – схема методу, інформаційна структура системи, структура навчання ШІ, діаграми варіантів використання та діяльності. Ці матеріали забезпечують розуміння логіки функціонування системи та послідовності виконання основних процесів.

Опрацьовано архітектуру штучного інтелекту ворогів, а також описано спеціалізовані програмні компоненти, необхідні для реалізації функціоналу. Визначено набір вхідних даних, які використовуються для генерації рівня: ландшафту, ворогів з керованою поведінкою та декоративних елементів.

У результаті сформовано цілісне уявлення про структуру системи, що дозволяє ефективно реалізувати метод процедурної генерації ігрового контенту з підтримкою адаптивного ШІ.

Розділ 3 Програмна реалізація інформаційної системи

3.1 Опис засобів розробки інформаційної системи для методу

Мова програмування C# є високорівневою, об'єктно-орієнтованою мовою, що підтримується корпорацією Microsoft [49]. Вона має зручний синтаксис, широкий набір засобів для роботи з пам’яттю, подіями, потоками та колекціями, а також є стандартною для розробки в Unity. Її використання дозволяє писати структурований, зрозумілий і масштабований код, що важливо для будь-якого ігрового проєкту.

Основа для виконання C#-коду – це платформа .NET, яка надає великий набір бібліотек [50], включно з System.Collections та System.Collections.Generic, які забезпечують підтримку колекцій і типізованих структур даних. Саме ці бібліотеки активно використовувалися для управління списками об'єктів, збереження параметрів гри та реалізації корутин.

Ключовим інструментом став ігровий рушій Unity, який надає всі необхідні засоби для створення інтерактивного середовища, фізичних взаємодій, графіки, анімацій та звуку [51]. Бібліотека UnityEngine, яка постачається разом із рушієм, забезпечує доступ до основних класів, таких як MonoBehaviour, GameObject, Transform, Collider2D, Animator та інші. З їх допомогою реалізується більшість ігрових механік, включаючи рух, зіткнення, обробку подій, взаємодію об’єктів тощо. Unity також має зручний візуальний редактор сцен, що значно спрощує роботу з контентом.

Для написання та редагування коду було обрано Visual Studio – сучасне інтегроване середовище розробки, що має повну підтримку C#, інтеграцію з Unity, підсвічування синтаксису, автодоповнення, систему відлагодження та керування проєктами. Visual Studio забезпечує зручність у роботі з великими обсягами коду, а також дозволяє ефективно тестувати та виправляти помилки ще на етапі розробки [52].

Для навчання штучного інтелекту противників у грі використовується мова програмування Python [53], яка широко застосовується в галузі машинного

навчання та штучного інтелекту. Завдяки простому синтаксису та великій кількості спеціалізованих бібліотек Python дозволяє ефективно розробляти і тренувати нейронні мережі, зокрема з використанням бібліотек TensorFlow, PyTorch та інших. Unity ML-Agents інтегрує Python у процес навчання агентів, що дозволяє створювати адаптивний інтелект з можливістю навчання на основі взаємодії з ігровим середовищем. Такий підхід забезпечує високу гнучкість та якість поведінки ворогів у грі.

Враховуючи все вищезазначене, вибір інструментів – C#, .NET, Unity, Python та Visual Studio – є цілком обґрунтованим. Вони забезпечили необхідний рівень продуктивності, стабільності, гнучкості та зручності, що дозволило реалізувати поставлені цілі проєкту без потреби у сторонніх бібліотеках чи складних додаткових компонентах.

3.2 Діаграма класів застосунку

Програмна архітектура розробленого застосунку побудована з урахуванням принципів модульності, масштабованості та зручності підтримки. Основні компоненти були виокремлені ще на етапі проєктування для забезпечення чіткого розмежування логіки, візуалізації та взаємодії з користувачем. Загальна структура реалізована за патернами компонентної архітектури, що характерна для ігрових проєктів на базі Unity. На рисунку 3.1 зображено діаграму класів.

Програмний продукт складається з таких основних модулів:

– GameManager модуль загального управління грою. Відповідає за ініціалізацію, перехід між станами гри (меню, гра, поразка), збереження стану та координування інших компонентів.

– WorldGenerator відповідає за процедурну генерацію світу. На основі заздалегідь визначених параметрів (висота, ширина, тип блоків, частота прогалин тощо) створює ігровий простір.

– PlayerController компонент керування гравцем. Реалізує обробку вводу, рух, стрибки, атаки, а також взаємодію з навколишнім середовищем.

– EnemyController модуль логіки ворогів. Здійснює патрулювання, виявлення гравця, атаку, а також реакцію на зміну стану середовища.

– ChunkManager реалізує поділ світу на чанки для покращення продуктивності. Дозволяє завантажувати та видаляти ділянки карти залежно від позиції гравця.

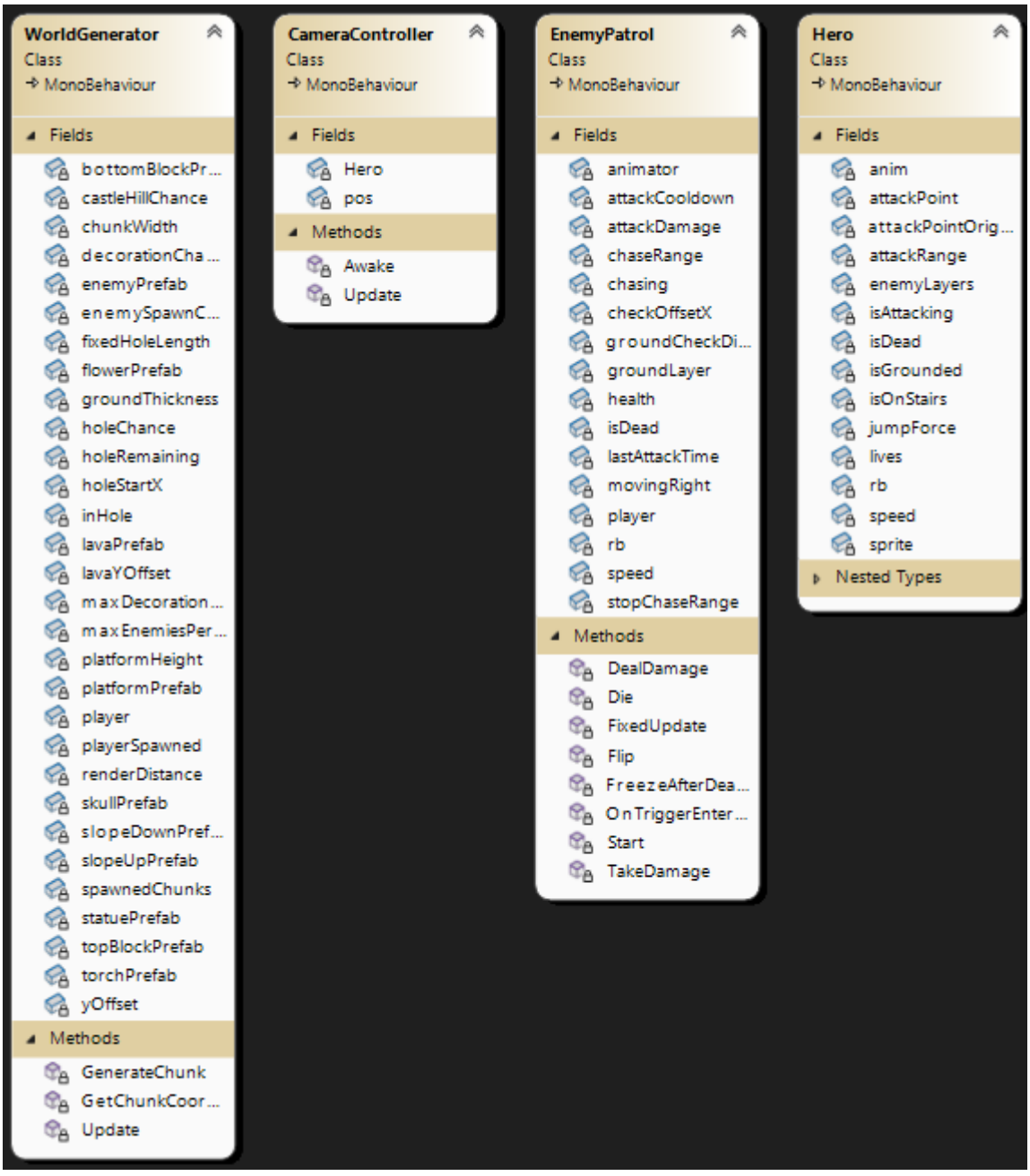


Рисунок 3.1 – Діаграма класів

– ObjectSpawner відповідає за розміщення об'єктів (декорації, факели, статуї, вороги) під час генерації кожного чанку.

– UIManager забезпечує управління елементами інтерфейсу (панелі здоров’я, текстові повідомлення, меню паузи).

Класи мають чітко визначені функції: наприклад, GameManager, WorldGenerator і ChunkManager належать до логічного шару, тоді як PlayerController, EnemyController і UIManager до шару взаємодії з користувачем та середовищем.

Взаємозв’язки між модулями реалізовані через доступ до спільних інтерфейсів та єдиних точок керування. Наприклад, WorldGenerator передає ChunkManager згенеровані дані для створення візуальних блоків, а ObjectSpawner взаємодіє з WorldGenerator для отримання координат розміщення об'єктів. Взаємодія між гравцем і ворогами реалізується через події: при досягненні певної анімаційної фази PlayerController викликає атаку, яка обробляється відповідними методами ворога.

3.3 Особливості реалізації програмних складових системи

Для початку створено персонажа, та написаний базовий скрипт для його переміщення, на рисунку 3.2 відображено самого персонажа та опис того що дає змогу керувати персонажем.

Рисунок 3.2 – Персонаж

Методи Run() та Jump() реалізують базову поведінку гравця – переміщення та стрибок. Вони враховують поточний стан об’єкта, напрям руху, змінюють зовнішній вигляд спрайта та коригують позицію точок атаки відповідно до напряму.

Можливість динамічних елементів знаходитись на поверхні а не провалюватись крізь них надає 2D колайдер, на рисунку 3.3 відображено колайдери персонажа та поверхні.

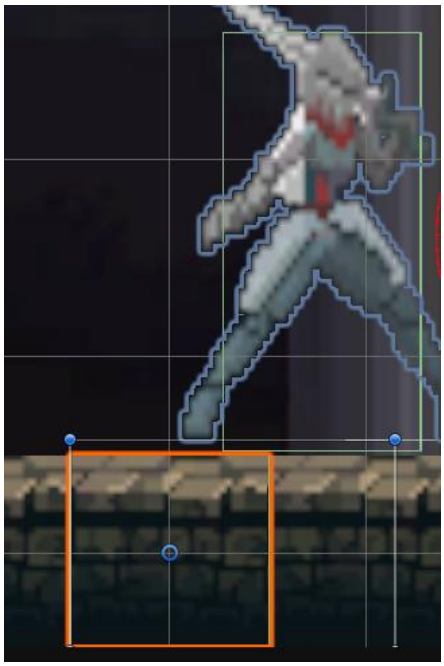


Рисунок 3.3 – Відображення 2D колайдерів

Колайдерами є зелені чотирикутники, вони визначають фізику об’єкту, тобто дають можливість динамічним елементам стояти на статичних елементах із колайдером.

Наступним важливим елементом застосунку є камера, задля забезпечення динамічності потрібно щоб камера слідувала за героєм, на рисунку 3.4 показано те як камера захоплює героя.

Скрипт CameraController відповідає за плавне стеження камери за головним героєм у 2D-грі. Він автоматично знаходить героя на сцені і кожен кадр коригує позицію камери, щоб вона залишалась позаду героя та трохи вище,

забезпечуючи зручний огляд ігрового простору. Завдяки згладженому руху (Lerp), камера не ривками переміщується за героєм, а м’яко слідує за ним, створюючи комфортне візуальне сприйняття для гравця.

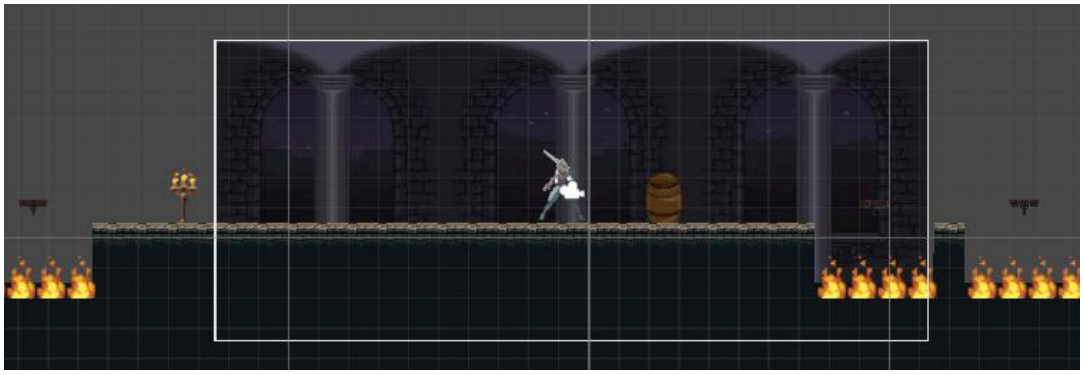


Рисунок 3.4 – Контроль камери

Також у застосунку реалізовано процедурну генерацію контенту, такого як підлога, прірви, вогонь в прірвах, декорації та вороги. На рисунку 3.5 відображено реалізований згенерований світ у згенерованому чанку.

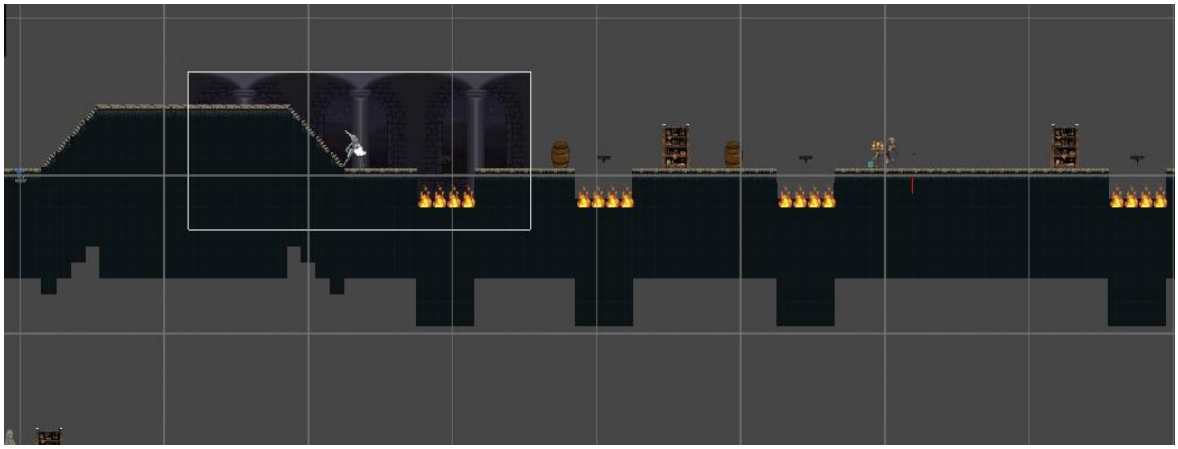


Рисунок 3.5 – Процедурно згенерований світ

Створення підлоги та блоків під нею реалізується частиною процедурної генерації в 2D-платформері. Його мета – побудувати платформи що складаються з одного верхнього блоку і кількох нижніх блоків, які йдуть під нею.

Утворення прогалин та платформи по центру відповідає за генерацію провалля під час процедурного створення світу, і є важливою частиною системи, що контролює змінність рельєфу та геймплейні особливості. Коли генератор

досягає позиції, де має бути яма, він вмикає режим inHole, що сигналізує про необхідність припинити створення землі в поточному місці. Параметр createGround примусово встановлюється у false, щоб уникнути встановлення блоків платформи в середині прірви. Далі код перевіряє, чи поточна позиція є початком або кінцем провалля, і додає відповідну координату до списку holeEdges. Це дозволяє зберегти важливу інформацію про межі прірви, щоб зрозуміти чи робити проганину далі, або почати генерувати звичайну землю. Особливу роль відіграє перевірка на середину провалля – якщо holeRemaining дорівнює половині довжини фіксованої ями, то над центром прірви створюється окрема платформа яка розміщується трохи вище основної площини рівня, створюючи можливість для гравця перестрибнути яму, використовуючи її як тимчасову опору.

Утворення вогню у прогалинах під ним додатково інстанціюються кілька блоків землі, щоб вогонь виглядав наче розміщений у заглибленні. Усі об’єкти додаються до загальної структури чанку.

Також у префабі вогню є наявний тригер, при зіткнені гравця чи ворога із цим тригером програється анімація смерті, та колайдери стають неактивними і блокується можливість до пересування. На рисунку 3.6 зображено тригер у префабі вогню.

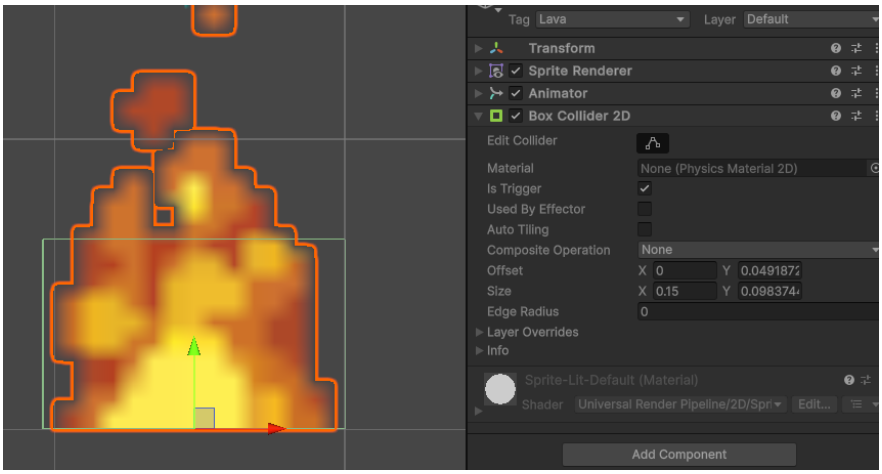


Рисунок 3.6 – Тригер у префабі вогню

На рисунку можна побачити галочку перед полем Is Trigger що означає що бокс колайдер вогню не є фізичним елементом, тобто на нього не можна стати, але при цьому при зіткненні із тригерним колайдером герой чи ворог помирає. На рисунку 3.7 зображено останній кадр анімації смерті від зіткнення із вогнем.

Рисунок 3.7 – Смерть героя при зіткненні із вогнем

При зіткненні із тригером розпочинає програватись анімація смерті, яка блокується на останньому фреймі задля того щоб запобігти зацикленій анімації.

На рисунку 3.8 наведено дерево анімації героя.

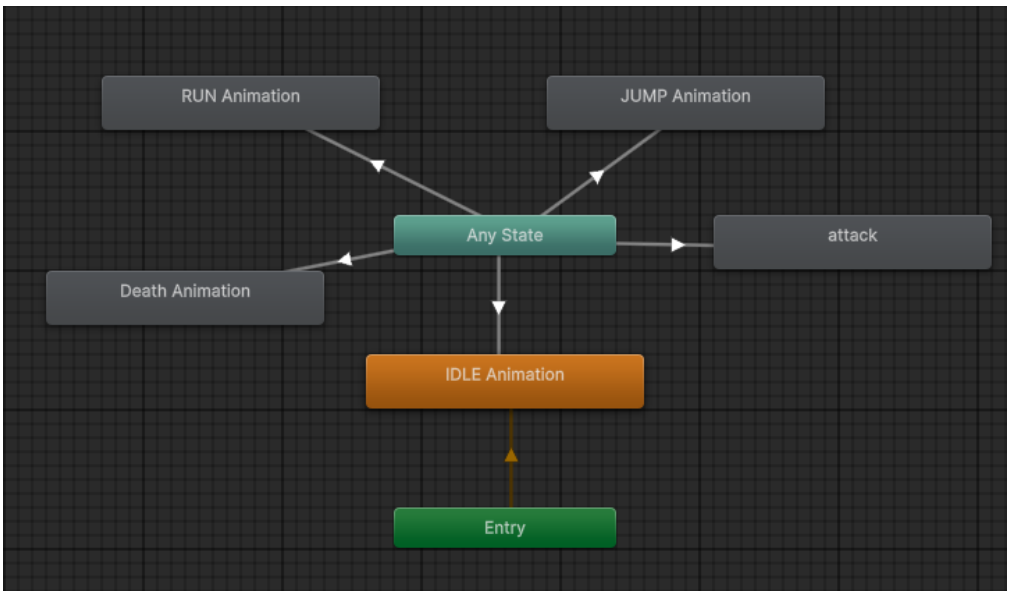


Рисунок 3.8 – Аніматор героя

По цій схемі можна зрозуміти що початковою анімацією є анімація спокою, після чого із любого стану можна перейти до іншого в зв’язку із діями героя, вийнятком є анімація смерті, оскільки вона може програтися лише один раз.

Далі на рисунку 3.9 наведено візуалізацію зору противника, по результатам якого він приймає рішення що йому робити.

Рисунок 3.9 – Зір ворога

Червона вертикальна лінія потрібна для того щоб перевіряти наступний нижній блок перед ворогом, це допомагає бачити прогалини та запобігає провалу у вогонь. Тут йде перевірка на наступний нижній блок, якщо там шар землі то ворог продовжує рух, якщо ж там якийсь інший шар то він розвертається і продовжує рух.

Синій відрізок відповідає за перевірку на стіни та інші перешкоди, за вийнятком шару героя, у такому випадку розпочинається анімація атаки, та наноситься шкода герою.

На рисунку 3.10 відображено дерево анімацій ворога.

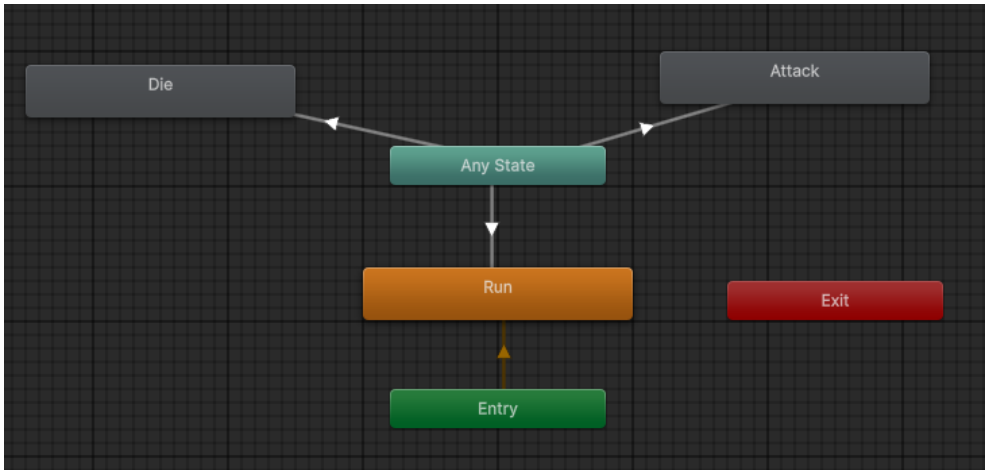


Рисунок 3.10 – Аніматор ворога

Початковий стан це анімація бігу, так як ворог майже завжди в русі, єдиним вийнятком є момент коли він атакує героя, тоді ж програється анімація атаки, і так же як у героя після анімації смерті усі фізичні властивості ворога вимикаються.

На рисунку 3.11 відображено анімацію атаки у героя.

Рисунок 3.11 – Анімація атаки героя

У цій анімації на конкретному кадрі є ключ, який визначає момент нанесення шкоди ворогу, на рисунку 3.12 відображено розміщення ключа на анімації атаки.

Рисунок 3.12 – Ключ моменту атаки

3.4 Тестування інформаційної системи та вимоги до розгортання

У рамках проекту, присвяченого розробці методу процедурної генерації ігрового контенту для 2D-платформера з інтелектуальною системою керування поведінкою противників, тестування реалізовано через Functional Testing, це коли копіями екрану доводиться можливість коректного виконання кожної з функцій. Вони охоплюють основні функціональні модулі системи, що забезпечують якість і стабільність роботи гри.

Основні напрямки тестування:

– Процедурна генерація світу: перевірка правильності та варіативності генерації рівнів, включаючи розміщення платформ, провалів, сходів, декорацій та інших елементів. Тестується унікальність згенерованих ділянок, відсутність критичних помилок (наприклад, непрохідних зон).

– Інтелектуальна поведінка ворогів: оцінка коректності роботи системи штучного інтелекту, яка забезпечує патрулювання, виявлення гравця, переслідування та атаку. Тестується плавність переходів між станами та реакція на навколишнє середовище.

– Взаємодія героя з ворогами та світом: перевірка системи пошкоджень, анімаційних станів (рух, атака, смерть), а також фізичних взаємодій із платформами, сходами і перешкодами.

Першочергово проводиться тестування стартової генерації ігрового контенту, для цього потрібно запустити генерацію світу, на рисунку 3.13 відображено створений світ.

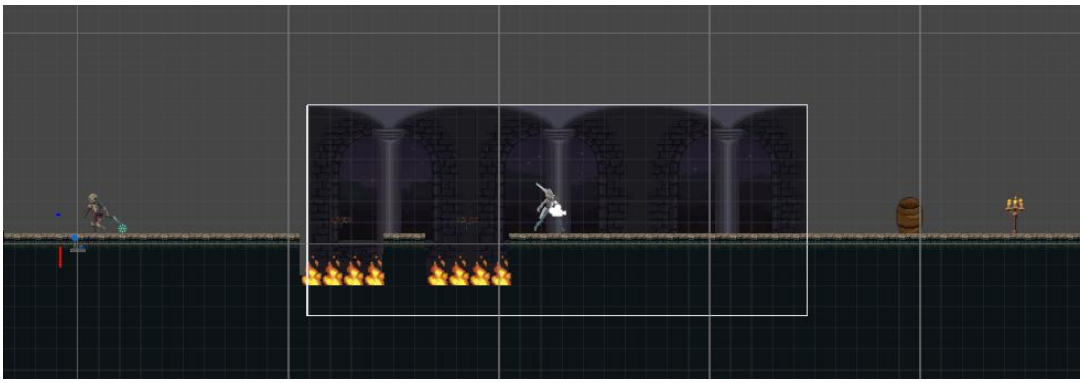


Рисунок 3.13 – Результат тестування стартової генерації

Далі потрібно протестувати коректне створення нових чанків із вмістом у них, для цього потрібно перемістити персонажа до краю поточного чанка, на рисунку 3.14 відображено результат тестування створення нового чанку.

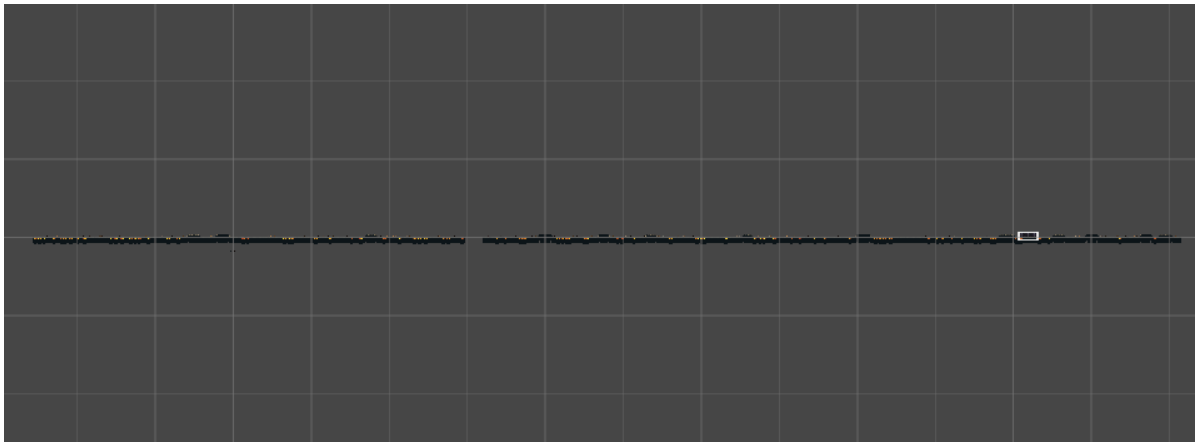


Рисунок 3.14 – Результат тестування створення нових чанків

На рисунку можна побачити збільшену локацію, це означає що генерація нових чанків виконується коректно, та без помилок.

Наступним етапом тестування є перевірка коректності роботи ШІ ворогів, для цього потрібно прослідкувати за поведінкою ворога коли він не стикається із героєм та навпаки. На рисунку 3.15 відображено результат роботи ШІ без контакту із героєм.

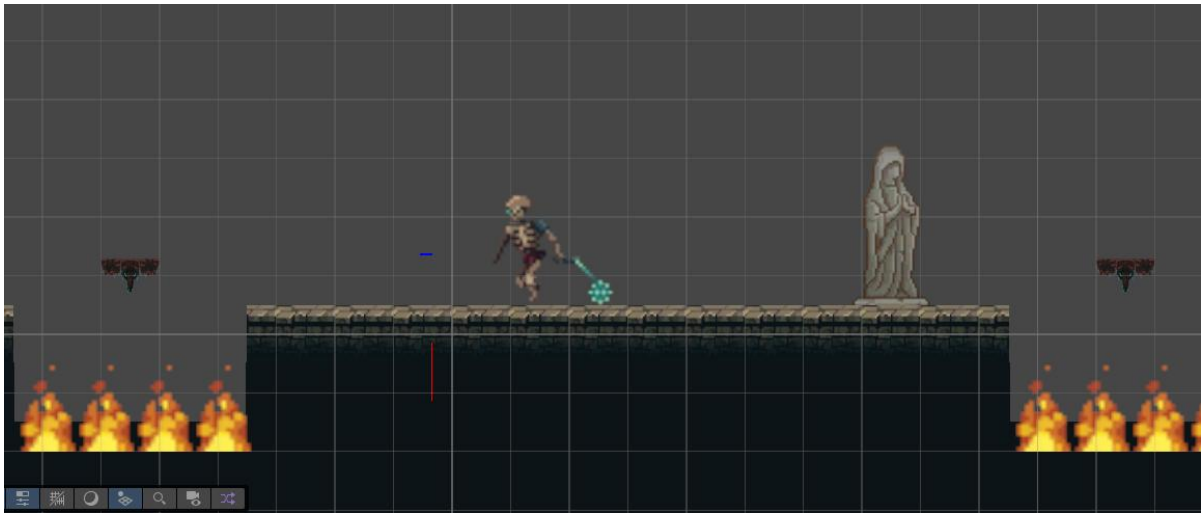


Рисунок 3.15 – Результат тестування поведінки ШІ без участі героя

Тестування поведінки ШІ ворога без взаємодії з героєм пройдено успішно, так як ворог уникає падіння у прірву, та просто патрулює територію в очікувані зміни ситуації на мапі, такої як поява героя в полі зору.

На рисунку 3.16 відображено тестування поведінки ворога коли поблизу присутній герой, по задумці він повинен переслідувати героя та розпочинати атаку коли той буде у зоні ураження.

Рисунок 3.16 – Результат тестування поведінки ШІ поблизу героя

Тестування пройдене успішно, так як ворог розпочав атаку коли герой був до нього надто близько, отже тестування поведінки ШІ пройдене успішно.

Також потрібно провести тестування функцій героя, таких як нанесення та отримання шкоди, а також коректність роботи системи керування.

На рисунку 3.17 відображено тестування системи керування, а саме стрибок.

Рисунок 3.17 – Результат тестування системи керування героєм

Тестування системи керування пройдено успішно, герой вдало виконав команду стрибку, кожен стрибок має рівномірну висоту та швидкість підйому.

На рисунку 3.18 відображено тестування нанесення шкоди ворогу.

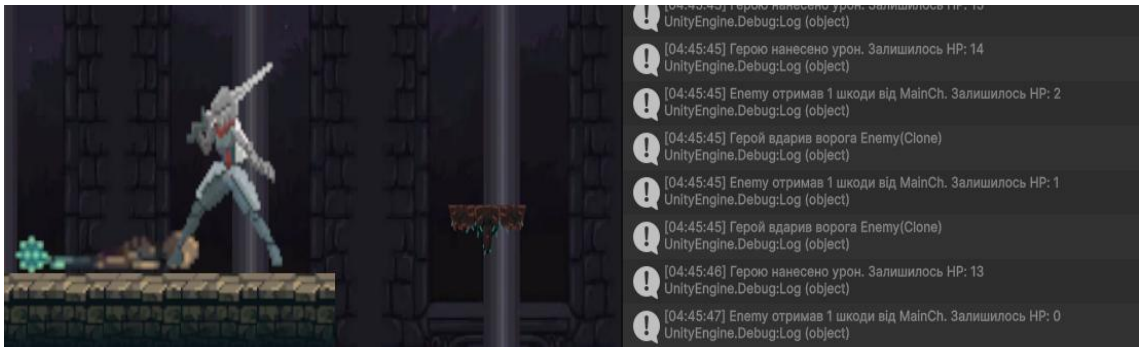


Рисунок 3.18 – Результат тестування нанесення шкоди

Тестування нанесення шкоди пройдено успішно, ворог після отримання критичної шкоди виконав анімацію смерті та деактивувався.

Залишилось протестувати взаємодію героя із тригером на спрайті вогню, для цього потрібно просто впасти у прірву із вогнем, герой повинен виконати

анімацію смерті та деактивуватись, на рисунку 3.19 відображено процес тестування роботи тригеру.

Рисунок 3.19 – Результат тестування роботи тригеру

Тестування роботи тригеру вогню пройдено успішно, герой виконав очікувані анімацію смерті та деактивацію, що означає неможливість подальшого керування без перезапуску світу.

Вимоги до розгортання системи

Інформаційна система реалізована з використанням рушія Unity 2021 що дозволяє створювати кросплатформенні збірки, зокрема для Windows (у вигляді .exe) та WebGL.

Для коректної роботи десктопної версії білду передбачаються такі мінімальні системні вимоги:

– Операційна система: Windows 7/10/11 (x64).

– Процесор: двоядерний з тактовою частотою не нижче 2 ГГц.

– Оперативна пам’ять: 4 ГБ або більше.

– Відеокарта: сумісна з DirectX 11 або OpenGL 3.0+.

– Вільне місце на диску: від 500 МБ.

Таким чином, проведене тестування підтвердило стабільність і працездатність основних механік гри. Розроблена система готова до створення білду та розгортання на цільових пристроях кінцевих користувачів.

3.5 Результати досліджень інтелектуальної поведінки противників

У рамках даного дослідження реалізовано та протестовано навчальний підхід до поведінки ворогів у 2D-платформері. Порівняно ефективність традиційного скриптового підходу та використання навчання з підкріпленням (reinforcement learning) для керування ворогами. Основна увага приділялася навчанню агентів атакувати героя та уникати падіння в прірву, на таблиці 1 відображено параметри навчання..

План дослідження:

– Підготовка середовища: ворог у 2D-просторі, герой як ціль, присутність прірв.

– Навчання агента: використано ML-Agents.

– Метрика ефективності: кількість влучних атак, середній час виживання, частка перемог над героєм.

– Порівняння поведінки навчального агента та скриптового ворога.

У таблиці 1 наведено параметри для навчання агента.

Таблиця 3.1 - параметри навчання

Параметр	Значення
Алгоритм	PPO
Кількість епізодів	150000
Максимальна тривалість епізодів	30 секунд
Середовище	Герой, ворог, платформи
Критерії оцінювання	Успішна атака, падіння, смерть

Таблиця параметрів навчання подає ключові налаштування, за якими відбувалося навчання агентів у середовищі Unity з використанням ML-Agents. Як алгоритм навчання було обрано PPO (Proximal Policy Optimization) – один з найбільш стабільних методів навчання з підкріпленням, що добре підходить для ігрових середовищ.

Протягом 150 000 епізодів агенти тренувалися у середовищі, яке складалося з героя, ворога та платформ, включаючи прірви. Максимальна тривалість одного епізоду становила 30 секунд реального часу, після чого епізод завершувався незалежно від результатів.

Для ефективного навчання визначено критерії винагороди, що стимулювали цілеспрямовану поведінку агента:

– Успішна атака на героя – позитивна винагорода.

– Падіння у прірву – негативна винагорода.

– Смерть агента – сильна негативна винагорода.

Ці параметри дозволили агенту вчитися взаємодіяти зі світом і приймати рішення, спрямовані на досягнення ігрових цілей. На таблиці 2 показано результати навчання.

Таблиця 3.2 - результати навчання ворога

Метрика	Епізод 10 000	Епізод 50 000	Епізод 100 000	Епізод 150 000
Частка перемог ворога (%)	7	24	41	56
Середній час життя (с)	3.2	6.5	10.8	13.1
Середня кількість атак	0.4	1.2	2.1	2.6

Таблиця результатів навчання ворога демонструє, як змінювалися ключові метрики поведінки ворожого агента на різних етапах навчання: після 10 000, 50 000, 100 000 та 150 000 епізодів.

– Частка перемог ворога (%) показує відсоток завершення епізодів на користь ворога, тобто ситуацій, коли ворог успішно атакував героя. Збільшення цього показника з 7% до 56% свідчить про прогрес у навчанні та поступове набуття ефективної бойової стратегії.

– Середній час життя (с) відображає середню тривалість існування ворога в епізоді. Показник зріс з 3,2 с до 13,1 с, що вказує на здатність агента уникати ризикованих ситуацій і вести себе обережніше у процесі бою.

– Середня кількість атак – це середнє число спроб завдати удару за один епізод. Збільшення значення з 0,4 до 2,6 свідчить про активнішу участь агента в бою та вдосконалення здатності до ініціативної атаки.

Для більш наглядного порівняння різниці між результатами навчання за різні кількості епізодів сформовано діаграму на рисунку 3.20.

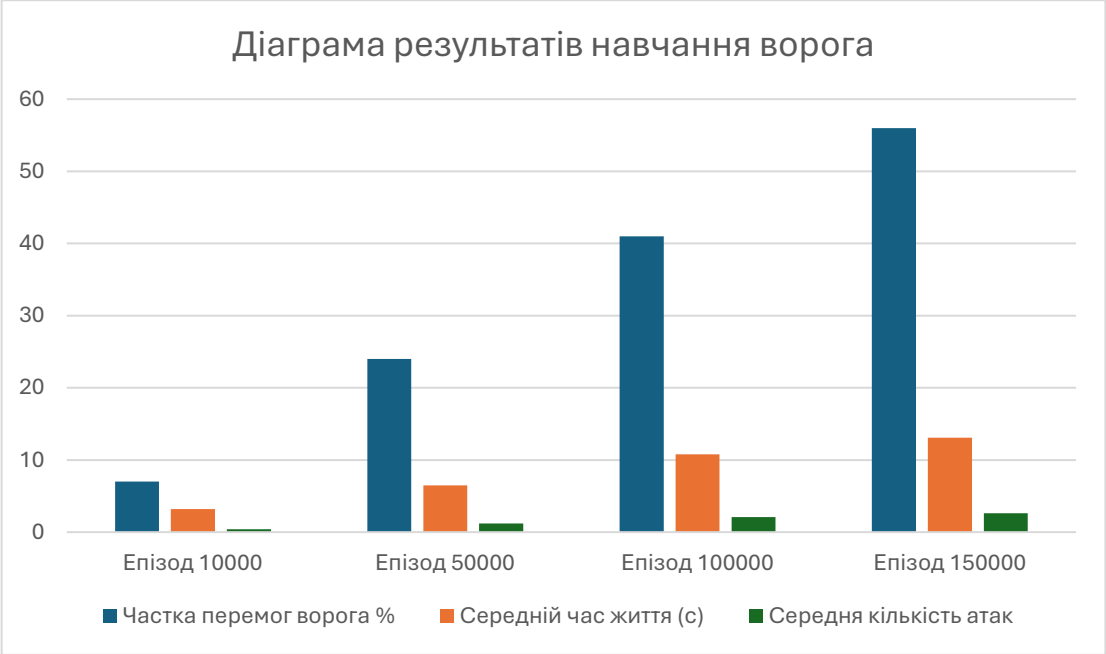


Рисунок 3.20 – Діаграма результатів навчання ворога

У сукупності наведені дані свідчать про суттєве покращення навичок агента в результаті навчання, що підтверджує ефективність використаного алгоритму та обраних параметрів середовища.

Для оцінки ефективності різних підходів до моделювання поведінки ворогів було проведено серію тестів у межах однакового рівня 2D-платформера. Усі експерименти виконувалися за однакових умов: наявність героя, платформ, прірв та обмеження часу епізоду. Поведінка ворога аналізувалася в двох варіантах:

– Скриптовий ворог, реалізований через класичний підхід зі статичними правилами.

– Навчений агент, керований алгоритмом PPO після 150 000 епізодів тренування.

Результати результату дослідження ефективності різних підходів до моделювання поведінки показано у 3 таблиці.

Таблиця 3.3 – результати дослідження ефективності підходів

Параметр	Ворог на основі FSM	Навчений агент
Середній час життя	6.2 с	13.1 с
Кількість атак / епізод	1.0	2.6
Здатність уникати прірв	Частково реалізована	Адаптивна
Поведінкова гнучкість	Залежить від кількості станів	Висока, формується під час навчання
Реакція на нові ситуації	Обмежена (в межах FSM)	Гнучка, здатна до узагальнення
Можливість еволюції стратегії	Немає без ручного втручання	Є – за рахунок подальшого навчання
Складність реалізації	Низька, потребує ручного опису станів	Вища, потребує тренування
Простота налагодження	Висока – стан видно явно	Складніша – політика неявна

Аналіз результатів дослідження ефективності:

– Скриптові вороги та FSM-підхід загалом добре працюють у статичних сценаріях, де середовище не змінюється і ворогові не потрібно пристосовувати поведінку до нових умов. Проте така система має обмежену гнучкість, адже вимагає ручного опису всіх можливих сценаріїв та переходів.

– Навчений агент, на відміну від FSM, демонструє здатність адаптуватися до нових ситуацій, що особливо корисно в умовах, коли рівні генеруються

динамічно або присутня варіативність у поведінці гравця. Крім того, поведінка агента може вдосконалюватися автоматично за допомогою подальшого навчання без необхідності змінювати код вручну.

Порівняння показує, що в умовах сучасних динамічних ігор із ускладненою взаємодією навчальні агенти мають суттєву перевагу над традиційними методами. Використання навчання з підкріпленням дає змогу моделювати більш реалістичну, гнучку та стратегічно обґрунтовану поведінку ворогів, що підвищує ігрову привабливість та складність.

Результати підтверджують доцільність використання навчального ШІ для керування ворогами навіть у відносно простих бойових сценаріях. Навчений агент демонструє адаптивну поведінку та здатність до узагальнення в межах заданого середовища. У подальших дослідженнях можливе розширення поведінки.

3.6 Висновки до розділу 3

Під час розробки застосунку для методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників визначено загальний підхід до створення системи та підібрано відповідні інструменти, такі як мова C#, середовище Visual Studio та рушій Unity.

Структуру проєкту представлено у вигляді діаграми класів, що наочно демонструє логіку побудови та взаємозв’язки між складовими.

Реалізовано основні функціональні елементи, зокрема генерацію рівнів, взаємодію об’єктів і логіку поведінки ворогів. Усі ключові механіки функціонують коректно та стабільно, що підтверджено під час тестування системи, інструкція користувача дозволяє швидко ознайомитись із можливостями застосунку.

У рамках дослідження проаналізовано ефективність роботи різних підходів до ШІ: класичного у вигляді скінченних автоматів і навчального, що відзначається високою адаптивністю та гнучкістю в управлінні ворогами.

Загальні висновки

У кваліфікаційній роботі бакалавра було досягнуто поставленої мети а саме розроблено метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників. Реалізацію методу виконано з використанням ігрового рушія Unity, мови програмування C# та середовища розробки Visual Studio.

У процесі роботи було виконано повний цикл дослідження: здійснено аналіз предметної області процедурної генерації контенту та штучного інтелекту в іграх, розглянуто існуючі методи генерації рівнів для 2D-платформерів, визначено особливості побудови ігрового світу з використанням процедурної генерації. Також досліджено підходи до реалізації інтелектуальної поведінки противників на основі вбудованих засобів Unity та можливостей навчання агентів.

Розроблено метод процедурної генерації рівнів, який дозволяє будувати 2D-ігровий світ з варіативними структурними особливостями. Розроблено архітектуру програмної системи, що об’єднує модулі генерації контенту та ШІ-систему управління противниками. Реалізовано настільний ігровий застосунок, який підтримує основні ігрові механіки, генерацію нових сцен, динамічне розміщення ворогів та інтелектуальну поведінку супротивників у межах обраної моделі.

У результаті тестування підтверджено коректність роботи системи. Система забезпечує автоматичне формування рівнів із дотриманням заданих правил розміщення елементів та коректну реакцію супротивників на дії гравця. Ефективність запропонованого методу оцінювалась за критеріями варіативності контенту, стабільності роботи системи та адаптивності поведінки ворогів. Результати засвідчили доцільність запропонованого підходу та можливість його масштабування.

Одержана інформаційна система виконує функції створення та управління ігровими рівнями, включаючи автоматизовану генерацію ландшафту, розміщення

ворогів та реалізацію їх інтелектуальної поведінки у відповідь на зміну ігрової ситуації.

У рамках дослідження ефективності навчального підходу до керування ворогами, було проведено серію експериментів з використанням бібліотеки ML-Agents та алгоритму PPO. Результати показали суттєве покращення характеристик агента у процесі навчання: частка перемог ворога зросла з 7% до 56%, середній час життя збільшився до 13,1 с, а кількість атак – до 2,6 за епізод. Це свідчить про високу ефективність застосованого методу та його здатність до адаптивної поведінки в ігровому середовищі.

Розроблений метод повністю відповідає поставленим завданням. Його можна використовувати як основу для розширення функціоналу, зокрема для впровадження навчального ШІ, покращення генератора рівнів, підтримки нових типів ландшафтів та сценаріїв гри.

Перелік посилань

1. Менжулін Д. В. Роль ігрових додатків у сучасному світі. Інтернет конференція. URL: http://www.konferenciaonline.org.ua/ua/article/id-1807/ (дата звернення: 28.04.2025).

2. Яка найкраща платформа для відеоігор?. *ЖУК*. URL: https://zhuk.ua/poradi-ta-rekomendatsii/yaka-naikrashcha-platforma-dlia-videoihor/?srsltid=AfmBOooMkI4wJS4sBJ82rTvL2lZ6oCoAGx3tpWWgGPO6X22jhidxW_yR (дата звернення: 05.05.2025).

3. Understanding procedural generation in games | lenovo US. Offizielle Lenovo DE Website. URL: https://www.lenovo.com/us/en/glossary/procedural-generation/?orgRef=https%3A%2F%2Fwww.google.com.ua%2F (дата звернення: 03.05.2025).

4. Procedural generation. Autodesk. URL: https://www.autodesk.com/solutions/procedural-generation#:~:text=Use%20for%20film%20and%20TV&text=In%20the%20film%20and%20TV,crowd%20behavior,%20and%20virtual%20environments. (дата звернення: 03.05.2025).

5. Farrokhi Maleki M., Zhao R. Procedural Content Generation in Games: A Survey with Insights on Emerging LLM Integration. arXiv preprint arXiv:2410.15644. 2024. URL: https://arxiv.org/abs/2410.15644 (дата звернення: 30.05.2025).

6. Kowal M., Toth A. J., Campbell M. J., et al. The tangled ways to classify games: A systematic review of how games are classified in psychological research. Frontiers in Psychology. 2023. Vol. 14. Article 11195997. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11195997/ (дата звернення: 30.05.2025).

7. What is a JRPG?. *Geek to Geek Media*. URL: https://geektogeekmedia.com/geekery/video-games/what-is-a-jrpg-definition-japanese-jrpgs-game/ (дата звернення: 05.05.2025).

8. FromSoftware, Inc. elden ring. FromSoftware, Inc., 2022. URL: https://store.steampowered.com/app/1245620/ELDEN_RING/ (дата звернення: 28.04.2025).

9. Admin. Ігрові жанри. QATestLab. URL: https://training.qatestlab.com/blog/technical-articles/games-genres/#:~:text=До%20найбільш%20популярних%20жанрів%20на,карткові%20ігри,%20пазли%20і%20головоломки. (дата звернення: 28.04.2025).

10. Ігрові жанри. QATestLab. URL: https://training.qatestlab.com/blog/technical-articles/games-genres/ (дата звернення: 03.05.2025).

11. Nsdg. Процедурне генерування світу в єдності. Sharp Coder Blog. URL: https://uk.sharpcoderblog.com/blog/world-generation-in-unity (дата звернення: 28.04.2025).

12. What is procedural generation. BorisTheBrave.Com. URL: https://www.boristhebrave.com/2024/05/25/what-is-procedural-generation/ (дата звернення: 06.05.2025).

13. Nsdg. Вступ до процедурної генерації в Unity. Sharp Coder Blog. URL: https://uk.sharpcoderblog.com/blog/an-introduction-to-procedural-generation-in-unity (дата звернення: 28.04.2025).

14. Kopel M., Maciejewski G. Comparison of procedural noise-based environment generation methods computational collective intelligence. 2020. URL: https://link.springer.com/chapter/10.1007/978-3-030-63007-2_69

15. Procedural generation and randomness. unseen diplomacy 2. URL: https://unseendiplomacy2.com/2020/01/procedural-generation-and-randomness/ (дата звернення: 03.05.2025).

16. Schatzeder D. The logic of procedural generation. medium. URL: https://schatzeder.medium.com/the-logic-of-procedural-generation-3043368a6a06 (дата звернення: 03.05.2025).

17. Omirov B. A., Redjepov Sh. B., Usmonov J. B. 2D Moore CA with new boundary conditions and its reversibility. arXiv preprint arXiv:2406.06195. 2024. URL: https://arxiv.org/abs/2406.06195

18. У Salo V. On von neumann regularity of cellular automata. arXiv preprint arXiv:2209.13373. 2022. URL: https://arxiv.org/abs/2209.13373

19. GeeksforGeeks. A* search algorithm. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/a-search-algorithm/ (дата звернення: 06.05.2025).

20. Binary space partitioning. *Valve Developer Community*. URL: https://developer.valvesoftware.com/wiki/Binary_space_partitioning (дата звернення: 06.05.2025).

21. Jagli D., Chandra S., Dhanikonda S. R., Laxmi N.Artificial Intelligence Usage in Game Development. SSRN. 2024. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4929567

22. Zhang L. Application of Artificial Intelligence Technology in Game NPC. proceedings of the first international conference on Science, engineering and technology practices for sustainable development. 2024. URL: https://doi.org/10.4108/eai.17-11-2023.2342705.

23. Скінченні автомати. Flashcards World. URL: https://flashcards.world/flashcards/sets/45ce2205-1c96-4c9b-98f3-84b99862c4c2/ (дата звернення: 28.04.2025).

24. Rule-Based AI. deepgram. URL: https://deepgram.com/ai-glossary/rule-based-ai (date of access: 28.04.2025).

25. Bernard Marr. What are the four types of AI?. Bernard Marr. URL: https://bernardmarr.com/what-are-the-four-types-of-ai/ (date of access: 28.04.2025).

26. Principles of game trees. saylor academy. URL: https://learn.saylor.org/mod/page/view.php?id=80822 (дата звернення: 03.05.2025).

27. ML and AI in game development in 2025. analytics vidhya. 2025. URL: https://www.analyticsvidhya.com/blog/2023/03/ml-and-ai-in-game-development/ (дата звернення: 30.05.2025).

28. Styles M. Procedural 2D terrain. trederia. URL: https://trederia.blogspot.com/2016/11/procedural-2d-terrain-part-2-creating.html (date of access: 28.04.2025).

29. Top procedural generation games. Steam 250. URL: https://steam250.com/tag/procedural_generation (дата звернення: 06.05.2025).

30. Rogue legacy 2. PlayStation. URL: https://www.playstation.com/ru-ua/games/rogue-legacy-2/ (дата звернення: 29.04.2025).

31. Tag Archives: rogue legacy. asolemdream. URL: https://asolemdream.wordpress.com/wp-content/uploads/2013/08/rogue-legacy.png (date of access: 29.04.2025).

32. White S. Rogue legacy 2 guide. IGN. URL: https://www.ign.com/wikis/rogue-legacy-2/Essential_Tips_and_Tricks (дата звернення: 29.04.2025).

33. Procedural content generation in games. arXiv.org. URL: https://arxiv.org/html/2410.15644v1 (дата звернення: 29.04.2025).

34. Contributors to SporeWiki. Spore. SporeWiki. URL: https://spore.fandom.com/wiki/Spore (дата звернення: 29.04.2025).

35. Contributors to SporeWiki. Planets. SporeWiki. URL: https://spore.fandom.com/wiki/Fiction:Delpha_Coalition_of_Planets/History/Development (дата звернення: 29.04.2025).

36. Contributors to SporeWiki. nest. *sporewiki*. URL: https://spore.fandom.com/wiki/Nest (дата звернення: 06.05.2025).

37. Contributors to SporeWiki. alliance. *sporewiki*. URL: https://spore.fandom.com/wiki/Alliance (дата звернення: 06.05.2025).

38. Terraria. *Terraria wiki*. URL: https://terraria.fandom.com/wiki/Terraria_Wiki (дата звернення: 06.05.2025).

39. Contributors to terraria wiki. world generation. terraria wiki. URL: https://terraria-archive.fandom.com/wiki/World_Generation#:~:text=World%20generation%20is%20

decided%20by,up%20the%20world%20creation%20screen. (дата звернення: 29.04.2025).

40. WorldGen previewer. ModsLab. URL: https://modslab.net/en/terraria/mods/f9ke9EAD86/ (дата звернення: 29.04.2025).

41. Medieval castle defense. itch.io. URL: https://masker.itch.io/medieval-castle-defense (дата звернення: 05.05.2025).

42. Fire animation - pixel art FX. itch.io. URL: https://brullov.itch.io/fire-animation (дата звернення: 05.05.2025).

43. Rocky world platformer. itch.io. URL: https://szadiart.itch.io/rocky-world-platformer-set (дата звернення: 05.05.2025).

44. Hero knight. itch.io. URL: https://luizmelo.itch.io/hero-knight (дата звернення: 05.05.2025).

45. Animated pixel art skeleton. itch.io. URL: https://astrobob.itch.io/animated-pixel-art-skeleton (дата звернення: 05.05.2025).

46. System.Collections namespace. Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/dotnet/api/system.collections?view=net-8.0 (дата звернення: 05.05.2025).

47. System.Collections.Generic. Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/dotnet/api/system.collections.generic?view=net-9.0 (дата звернення: 05.05.2025).

48. Juliani, A., et al. (2018). Unity: A General Platform for Intelligent Agents. arXiv preprint arXiv:1809.02627. URL: https://arxiv.org/abs/1809.02627

49. C# guide. Microsoft learn. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 05.05.2025).

50. .NET Framework documentation. microsoft learn. URL: https://learn.microsoft.com/uk-ua/dotnet/framework/ (дата звернення: 05.05.2025).

51. Учасники проектів Вікімедіа. Unity (ігровий рушій). Вікіпедія. URL: https://uk.wikipedia.org/wiki/Unity_(ігровий_рушій) (дата звернення: 05.05.2025).

52. Visual studio. Visual Studio. URL: https://visualstudio.microsoft.com/ (дата звернення: 05.05.2025).

53. What is python used for?. Coursera. URL: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (дата звернення: 30.05.2025).

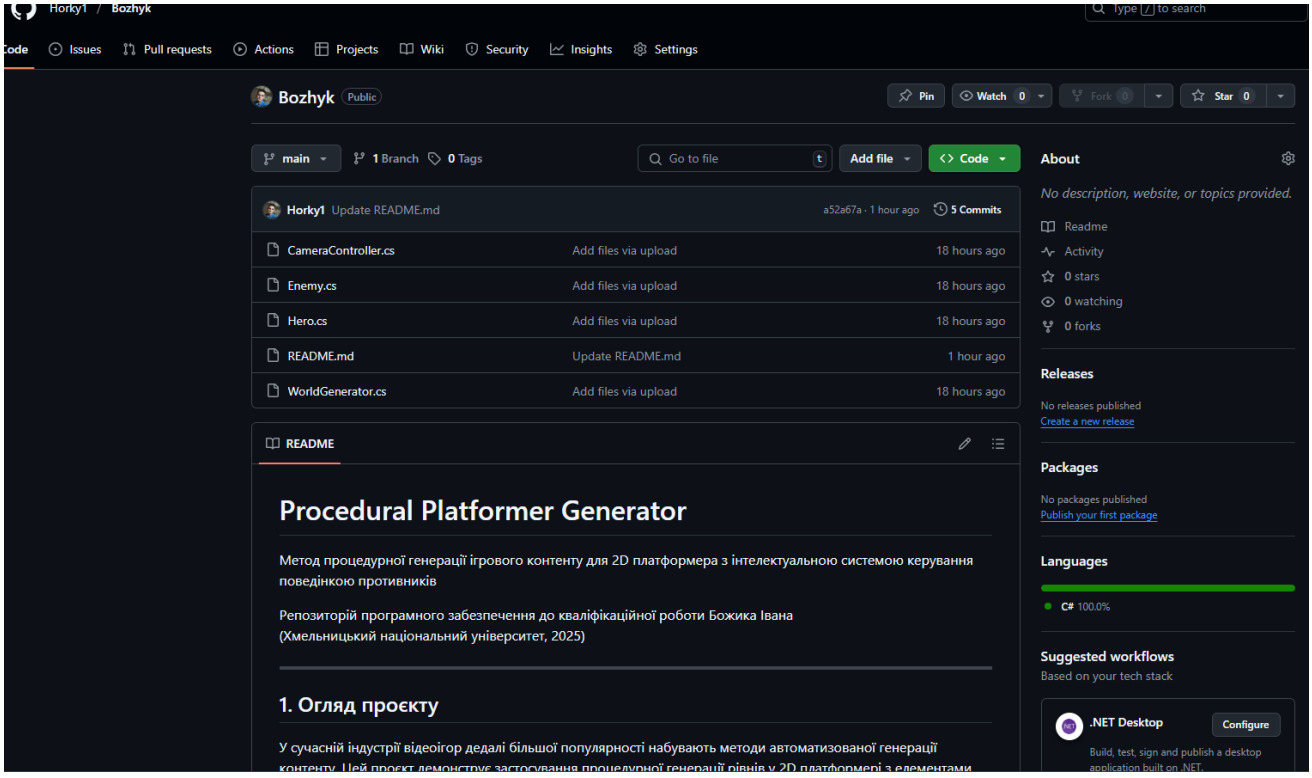
ДОДАТКИ

Додаток А

Програмні коди

Посилання на репозиторій: https://github.com/Horky1/Bozhyk

Вигляд сторінки репозиторію:



Опис вмісту репозиторію:

– CameraController, скрипт який відповідає за переміщення камери, для того щоб герой був завжди у її полі зору.

– Enemy, скрипт який відповідає за роботу ШІ ворога, також можна змінювати налаштування здоров’я та швидкості.

– Hero, скрипт за допомогою якого можна керувати героєм, змінювати його параметри, такі як здоров’я, швидкість та силу стрибку, також у ньому визначено у який кадр анімації буде проводитись нанесення шкоди ворогу.

– WorldGenerator, скрипт за допомогою якого генерується світ, визначається зміна висот ландшафту, утворення прірв, розміщення пасток та ворогів, також його можна налаштувати змінивши параметри на основі яких відбувається генерація, до прикладу в одному чанку можна утворювати не одного а чотирьох ворогів та тому подібне.

Додаток Б

Презентаційний матеріал

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

МЕТОД ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ІГРОВОГО КОНТЕНТУ ДЛЯ 2D ПЛАТФОРМЕРА З ІНТЕЛЕКТУАЛЬНОЮ СИСТЕМОЮ КЕРУВАННЯ ПОВЕДІНКОЮ ПРОТИВНИКІВ

Виконав:
студент 4 курсу, групи КН-21-1
Божик Іван Сергійович

Керівник:
д.т.н., професор кафедри КН
Манзюк Едуард Андрійович

Актуальність

Процедурна генерація контенту є важливим напрямом сучасної ігрової розробки, оскільки дозволяє створювати великі та різноманітні світи з мінімальними витратами ресурсів.

У поєднанні зі штучним інтелектом противників це підвищує варіативність ігрового процесу та забезпечує більшу цікавість для гравців.

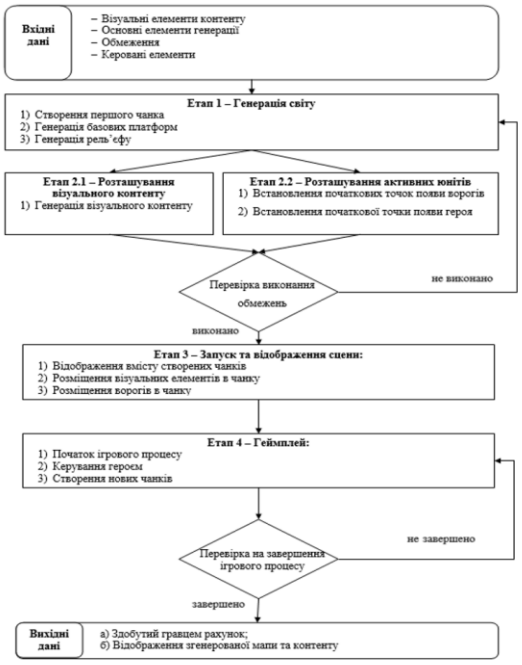
Мета і задачі роботи

Мета роботи – підвищення варіативності ігрового процесу шляхом розробки методу процедурної генерації ігрового контенту з інтелектуальною системою керування поведінкою противників.

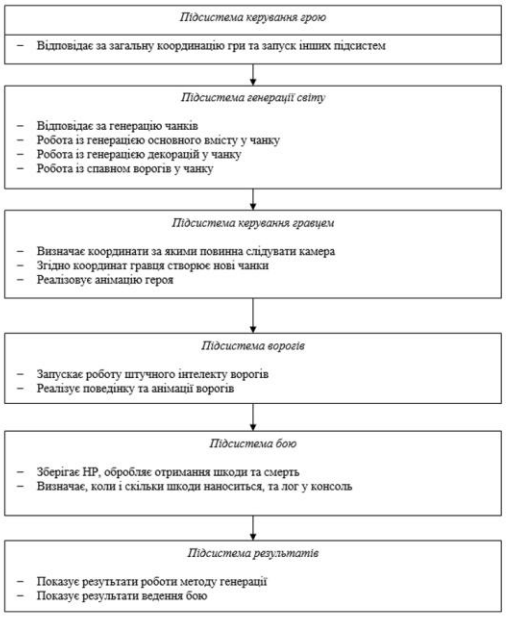
Основні завдання:

- Провести аналіз підходів до процедурної генерації та ігрового ШІ;
- Дослідити методи створення рівнів у 2D-платформерах;
- Розробити власний метод генерації ігрового світу;
- Реалізувати систему ШІ для ворогів;
- Провести тестування та оцінити ефективність реалізованих рішень.

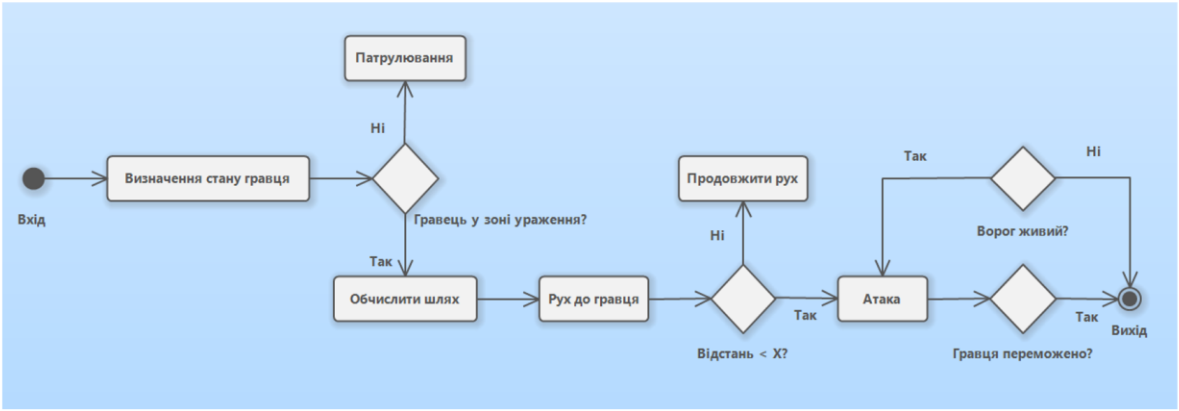
Схема методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників



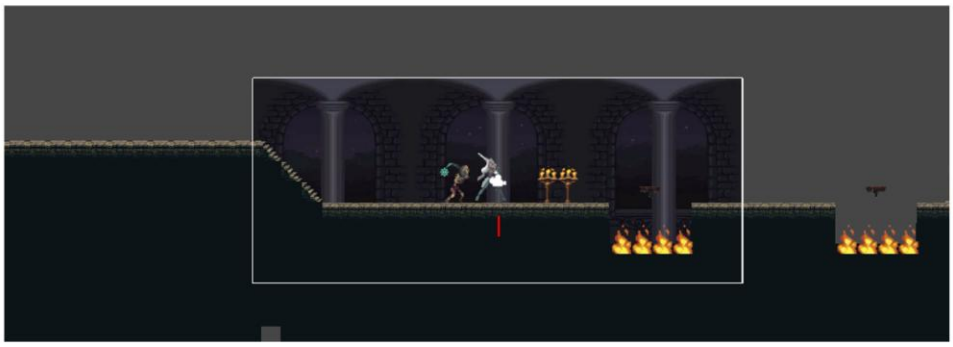
Схемаі нформаційної структури методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників



Діаграма діяльності ворога



Результат роботи методу



На світлині можна побачити коректну роботу методу, світ процедурно згенерований.

Результат роботи методу

Ворог після розміщення на мапі після вдалого переслідування розпочав атакувати героя.

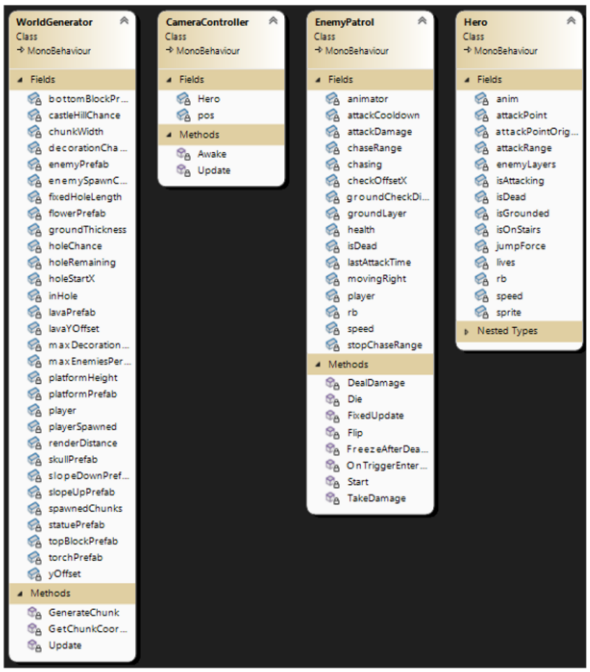
9

Результат роботи методу

По мірі переміщення героя створюються нові чанки, що забезпечує безкінечну генерацію світу.

10

Діаграма класів програмної реалізації методу процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників



11

Висновки

В межах кваліфікаційної роботи бакалавра розроблено метод процедурної генерації ігрового контенту для 2D-платформера з інтелектуальною системою керування поведінкою противників. Реалізацію виконано в Unity на C#.

Проведено аналіз методів генерації та ігрового ШІ, спроєктовано архітектуру системи, реалізовано автоматичне створення рівнів, розміщення ворогів і їхню поведінку.

Результати тестування підтвердили стабільність системи, варіативність контенту та коректну реакцію ворогів. Також досліджено можливості навчального ШІ з використанням ML-Agents і алгоритму PPO.

12

ДЯКУЮ ЗА УВАГУ!

Sun Jun 08 21:58:24 EEST 2025, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 4.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 11%**

	Document		Sum coincidence on the DB	
ID: 244119 Title: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників Added in a DB: 2025-06-08 Authors: Іван БОЖИК Heads: Едуард МАНЗЮК Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	57110	842	4010 (7%)	55 (7%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Іван БОЖИК

Співавтор:

Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників

Науковий керівник: Едуард МАНЗЮК, д.т.н., доцент

Підрозділ: Кафедра комп'ютерних наук

Коефіцісит подібності 1:7%

Коефіцісит подібності 2:4.2%

Мікропробіли: 0

Заміна букв: 2

Інтервали: 0

Білі знаки: 17

Дата створення звіту: 2025-06-08 20:32:08.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 8.06.25 експерт Петровський Р.О.

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи *Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників*
Автор *студент групи КН-21-1 Іван БОЖИК*
Освітня програма *Комп'ютерні науки*
Рівень вищої освіти *перший (бакалаврський)*
Спеціальність *122 – Комп'ютерні науки*
Науковий керівник: *д.т.н., проф. каф. комп'ютерних наук Едуард МАНЗЮК*

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмними засобами комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	***відповідає***
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	***відсутні***

Підтвердження:

Запозичення, виявлені в роботі Івана БОЖИКА, не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти, які не мають авторства і містять поширені конструкції та загальновідомі терміни, скорочення. Рівень подібності не перевищує допустимої межі. Таким чином, робота є законною та приймається до захисту.

Обсяг запозичень, визначений системами виявлення збігів/ідентичності/схожості:
- за системою Anti-Plagiarism: 4%;
- за системою StrikePlagiarism КП1: 7%, КП2: 4,2%.

08.06.2025

Завідувач кафедри _____ *Олександр БАРМАК*

Гарант освітньої програми _____ *Олександр МАЗУРЕЦЬ*

Керівник кваліфікаційної роботи _____ *Едуард МАНЗЮК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-21-1 Божик Іван Сергійович*
за темою: *Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників*

1. Актуальність обраної теми

Тема є актуальною, оскільки поєднує сучасні підходи до процедурної генерації ігрових рівнів та елементи штучного інтелекту, що мають широке застосування в ігровій індустрії.

2. Повнота розкриття мети та завдань роботи

Мета та завдання роботи сформульовані чітко та логічно. Усі поставлені завдання виконані у повному обсязі.

3. Зміст кожного розділу роботи

Перший розділ містить аналіз інформаційних моделей, огляд теоретичних підходів до подібних задач, аналіз подібних існуючих програмних засобів та наукових рішень, також визначено мету, задачі та вимоги до реалізації.

У другому розділі представлено схему методу, архітектуру інтелектуальної поведінки противників, визначено вхідні данні для генерації також представлено розробку інформаційної системи.

Третій розділ присвячений програмній реалізації інформаційної системи, у ньому представлено засоби для розробки, діаграму класів, особливості реалізації програмних складових системи, проведено тестування системи, а також представлені результати досліджень.

4. Оцінка розробленого методу та його практична цінність

Розроблений метод демонструє стабільну роботу, дозволяє створювати варіативні рівні та містить інтелектуальну поведінку ворогів. Практична цінність полягає в можливості використання підходу у створенні 2D-ігор із динамічним контентом.

5. Якість оформлення кваліфікаційної роботи бакалавра

Робота оформлена згідно з вимогами до академічних робіт. Матеріал структурований, наявні ілюстрації, діаграми та результати тестування.

6. Недоліки кваліфікаційної роботи бакалавра

Недоліки є незначними й не впливають на загальну якість роботи. Доцільно було б розширити тестування за різними сценаріями поведінки ворогів або надати порівняльний аналіз альтернативних методів генерації.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка « відмінно*»*

Рецензент _____ Т. Говорущенко

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-21-1 Івана БОЖИКА*

за темою *Метод процедурної генерації ігрового контенту для 2D платформера з інтелектуальною системою керування поведінкою противників*

1. Актуальність теми

Тема кваліфікаційної роботи є актуальною у контексті сучасного розвитку ігрової індустрії. Процедурна генерація контенту стала однією з ключових технологій, що дозволяє створювати різноманітні та непередбачувані ігрові світи з мінімальними витратами ресурсів. Поєднання автоматизованого створення рівнів з інтелектуальними системами керування противниками відкриває нові можливості для підвищення варіативності та цікавості ігрового процесу.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Виконана робота повністю відповідає вимогам стандарту спеціальності 122 "Комп'ютерні науки". Дослідження охоплює розробку інформаційних моделей, алгоритмів та програмних систем для автоматизованого створення ігрового контенту. У роботі застосовано сучасні методи машинного навчання, зокрема навчання з підкріпленням, для реалізації інтелектуальної поведінки ігрових агентів. Використано передові технології програмування, включаючи Unity ML-Agents та алгоритм PPO.

3. Професійні та особистісні якості бакалавра

Під час виконання кваліфікаційної роботи студент Божик Іван продемонстрував високий рівень професійної підготовки та відповідальне ставлення до навчального процесу. Він своєчасно виконував усі етапи дослідження, проявляв ініціативність у вирішенні складних технічних завдань. Студент продемонстрував достатні компетенції у галузі алгоритмічного програмування, машинного навчання та розробки ігрових застосунків.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Усі результати, представлені в роботі, отримані завдяки самостійній роботі студента. Він особисто розробив архітектуру системи, реалізував алгоритми

процедурної генерації, створив та протестував інтелектуальну систему керування противниками.

5. Ступінь оволодіння методами дослідження

Студент продемонстрував впевнене володіння сучасними методами дослідження в галузі комп'ютерних наук. Він ефективно застосував методи процедурної генерації, алгоритми машинного навчання з підкріпленням, технології об'єктно-орієнтованого програмування. Особливо слід відзначити успішне використання фреймворку Unity ML-Agents та алгоритму Proximal Policy Optimization для навчання ігрових агентів.

6. Повнота та якість розкриття теми роботи

Тема роботи розкрита повністю та всебічно. Проведено ґрунтовний аналіз існуючих підходів до процедурної генерації та методів реалізації штучного інтелекту в іграх. Розроблено оригінальний метод, що поєднує автоматизоване створення рівнів з адаптивною поведінкою противників. Експериментальні дослідження довели переваги навчального підходу над традиційними методами.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Робота має чітку логічну структуру та послідовний виклад матеріалу. Усі розділи органічно пов'язані між собою та спрямовані на досягнення поставленої мети. Теоретичні положення підкріплені практичними результатами. Стиль викладу науковий, аргументований, літературно грамотний. Використано сучасну наукову термінологію та відповідні стандарти оформлення.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений метод має значний потенціал для практичного застосування в ігровій індустрії. Система процедурної генерації може бути використана для створення різноманітних 2D-платформерів з динамічним контентом. Інтелектуальна система керування противниками придатна для впровадження в різні типи ігор. Окремі компоненти, зокрема алгоритми генерації рівнів та навчальні агенти, можуть знайти застосування в освітніх програмах та наукових дослідженнях.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи актуальність теми, високу якість виконання, практичну значущість результатів та відповідність усім вимогам до кваліфікаційних робіт бакалавра, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**» .*

Керівник _____ *д.т.н., проф. каф. КН Едуард МАНЗЮК*