Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота магістра

з теми: «Розробка та дослідження алгоритмів процедурної генерації ігрових всесвітів»

Виконав: здобувач вищої освіти групи _KN1-M24_

спеціальності 122 Комп’ютерні науки

_____Медвідь Юрій Володимирович_____
(прізвище, ім’я та по батькові здобувача вищої освіти)

Керівник:___Смалько Олена Аркадіївна_____
_____кандидат педагогічних наук, доцент_____
(прізвище, ім’я та по батькові, науковий ступінь, вчене звання керівника)

Рецензент:_____
_____
(прізвище, ім’я та по батькові, науковий ступінь, вчене звання рецензента)

м. Кам’янець-Подільський – 2025 р.

ЗМІСТ

АНОТАЦІЯ 3
ВСТУП 5
РОЗДІЛ 1. МЕТОДОЛОГІЧНІ ОСНОВИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ТА АНАЛІЗ АЛГОРИТМІВ 8
1.1 Класифікація методів процедурної генерації 8
1.2 Дослідження ефективності генеративних алгоритмів 14
Висновки до розділу 1 17
РОЗДІЛ 2. ЗАСТОСУВАННЯ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ У ПРОЦЕДУРНІЙ ГЕНЕРАЦІЇ 20
2.1 Використання нейронних мереж у генеративних алгоритмах 20
2.2 Вдосконалення методів процедурної генерації на основі нейромережевого навчання 24
2.3 Перспективи розвитку нейромережевих генеративних моделей 28
Висновки до розділу 2 31
РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ 33
3.1 Формалізація вимог до програмного забезпечення 33
3.2 Алгоритмічне забезпечення та реалізація процедурної генерації 35
3.3 Оптимізація та аналіз продуктивності розробленої системи 38
Висновки до розділу 3 40
ВИСНОВОК 43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 46
ДОДАТОК. Програмні інструменти для процедурної генерації 51

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню та розробці алгоритмів процедурної генерації ігрових всесвітів з використанням методів машинного навчання, що є важливим напрямом сучасної розробки відеоігор. Процедурна генерація дозволяє автоматично створювати великі, деталізовані та логічно структуровані ігрові середовища, мінімізуючи потребу в ручному проєктуванні та суттєво скорочуючи витрати ресурсів. В роботі розглядається, як методи машинного навчання можуть бути використані для оптимізації процедурної генерації, покращення варіативності та якості створюваних світів.

У роботі проведено аналіз існуючих методів процедурної генерації, їхніх переваг та недоліків, а також розглянуто підходи до генерації ландшафтів, архітектурних структур, ігрових об’єктів та інших елементів, що формують цілісний ігровий світ. Особливу увагу приділено використанню машинного навчання для покращення продуктивності, оптимізації алгоритмів та забезпечення високої якості згенерованого контенту.

На основі проведеного аналізу реалізовано алгоритми, що інтегрують машинне навчання для створення логічно узгоджених, реалістичних та естетично привабливих ігрових просторів. Запропоновані методи пройшли тестування, результати якого підтвердили їхню ефективність у створенні ігрових світів з високим рівнем деталізації та оптимальним використанням обчислювальних ресурсів.

Результати дослідження можуть бути використані у розробці відеоігор з відкритими світами, процедурно генерованими рівнями та адаптивними середовищами. Крім того, запропоновані алгоритми мають потенціал для застосування у суміжних галузях, таких як віртуальна реальність, комп’ютерна графіка, геопросторове моделювання та архітектурне проєктування.

ABSTRACT

The qualification thesis is dedicated to the research and development of procedural generation algorithms for game universes using machine learning methods, which is a crucial direction in modern video game development. Procedural generation enables the automatic creation of vast, detailed, and logically structured game environments, minimizing the need for manual design and significantly reducing resource costs. This study examines how machine learning methods can be utilized to optimize procedural generation, enhance variability, and improve the quality of generated worlds.

The work analyzes existing procedural generation methods, their advantages and drawbacks, and explores approaches to generating landscapes, architectural structures, game objects, and other elements that form a cohesive game world. Special attention is given to the application of machine learning for improving performance, optimizing algorithms, and ensuring high-quality generated content.

Based on the conducted analysis, algorithms integrating machine learning were implemented to create logically consistent, realistic, and aesthetically appealing game spaces. The proposed methods were tested, and the results confirmed their effectiveness in generating highly detailed game worlds with optimal computational resource utilization.

The research findings can be applied in the development of video games featuring open worlds, procedurally generated levels, and adaptive environments. Additionally, the proposed algorithms have potential applications in related fields such as virtual reality, computer graphics, geospatial modeling, and architectural design.

ВСТУП

У сучасній індустрії відеоігор одним із ключових викликів є створення великих, різноманітних та захопливих ігрових світів, які забезпечують унікальний досвід для кожного гравця. Традиційні методи ручного проєктування ігрового середовища потребують значних ресурсів, часу та зусиль, що робить їх малоефективними, особливо для проєктів з відкритим світом або великою кількістю генерованого контенту. Вирішенням цієї проблеми є використання алгоритмів процедурної генерації, які дозволяють автоматично створювати складні віртуальні простори, зберігаючи баланс між випадковістю та логічною структурованістю.

З розвитком апаратного забезпечення та програмних технологій зростає потреба у вдосконаленні методів процедурної генерації. Використання таких алгоритмів дозволяє не лише зменшити витрати на розробку контенту, а й зробити ігровий процес більш динамічним та варіативним. Процедурна генерація активно застосовується в різних жанрах: рольових іграх, стратегіях, симуляторах виживання, пригодницьких іграх тощо. Вона дозволяє створювати реалістичні ландшафти, міста, печери, підземелля, розташування об'єктів та навіть системи квестів без необхідності вручного опрацювання кожного елемента.

Актуальність дослідження обумовлена тим, що попри широке використання процедурної генерації, існують проблеми, пов’язані з оптимізацією продуктивності, якістю створюваних світів та досягненням балансу між випадковістю й контрольованою структурою. Недостатньо розроблені алгоритми можуть створювати нелогічні або неестетичні середовища, що негативно впливає на ігровий досвід. Крім того, складність розробки та налаштування таких алгоритмів ускладнює їх ефективне використання, особливо в умовах великомасштабних проєктів.

З розвитком методів машинного навчання все більше розробників звертаються до цих технологій для покращення якості процедурної генерації. Машинне навчання дозволяє оптимізувати генерацію контенту, покращити

варіативність та логічну узгодженість світів, зменшити витрати часу на розробку та забезпечити більш динамічний ігровий процес. Проте ці методи все ще перебувають на стадії активного розвитку, що створює додаткові виклики, зокрема в контексті реальних ігрових проєктів.

Об'єктом дослідження є процеси та методи процедурної генерації ігрових світів, які використовуються для автоматизованого створення контенту в комп'ютерних іграх.

Предметом дослідження є алгоритмічні підходи до процедурної генерації, які впливають на якість, продуктивність та гнучкість створюваного ігрового середовища, зокрема методи генерації ландшафтів, об'єктів та структур, а також їх оптимізація для зменшення обчислювальної складності.

Метою дослідження є розробка та аналіз алгоритмів процедурної генерації ігрових світів, що забезпечують оптимальне поєднання продуктивності, логічної узгодженості та різноманітності створюваного контенту за допомогою машинного навчання.

Для досягнення цієї мети необхідно вирішити такі **завдання**:

1. Провести аналіз існуючих методів процедурної генерації та їх застосування в ігровій індустрії.
2. Визначити ключові вимоги до алгоритмів процедурної генерації.
3. Розробити та реалізувати алгоритми, що використовують машинне навчання для покращення процедурної генерації.
4. Провести тестування запропонованих алгоритмів, оцінити їхню ефективність, продуктивність та якість створюваного контенту.

Для досягнення поставлених цілей у роботі використовуються такі **методи**:

1. Аналіз наукової та технічної літератури з процедурної генерації та алгоритмічних методів моделювання віртуальних світів.
2. Реалізація та тестування алгоритмів на експериментальних наборах даних з використанням машинного навчання.
3. Порівняльний аналіз продуктивності та якості створюваного контенту.

Практична цінність дослідження полягає у можливості застосування розроблених алгоритмів у процесі створення відеоігор, що дозволить: зменшити витрати на розробку контенту, підвищити якість та логічну узгодженість ігрових середовищ, надати гравцям унікальний досвід взаємодії з динамічними ігровими світами. Результати роботи можуть бути використані розробниками відеоігор, а також у навчальних і дослідницьких проєктах, пов’язаних із процедурною генерацією.

Основні результати дослідження були представлені на наукових конференціях [263, 4], а також за результатами роботи II Міжнародної наукової конференції «Штучний інтелект у науці та освіті» (AISE 2025) [5] підготовлено розділ з назвою «Формування основ ігрового штучного інтелекту» для колективної монографії, це підтверджує актуальність та практичну значущість напрацьованих матеріалів. Окрім того, було проведено тестування алгоритмів на реальних ігрових проєктах, що дало змогу оцінити їхню ефективність в умовах реального використання.

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатку.

РОЗДІЛ 1. МЕТОДОЛОГІЧНІ ОСНОВИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ТА АНАЛІЗ АЛГОРИТМІВ

1.1 Класифікація методів процедурної генерації

Процедурна генерація ігрових всесвітів є важливою складовою сучасного геймдеву, що дозволяє автоматизувати процес створення складних середовищ із мінімальними витратами часу та ресурсів. Її ключовими аспектами є забезпечення унікальності згенерованих світів, їх реалістичності та відповідності потребам гри. У науковій літературі основна увага приділяється базовим алгоритмам, таким як шум Перліна, фрактальні методи та клітинні автомати. Кожен із них має свої переваги й особливості, які визначають їхню сферу застосування.

Процедурна генерація контенту (PCG) є ключовою технологією для створення динамічних ігрових світів. У статті [36] представлено різні методи PCG, зокрема генерацію рівнів, карт, квестів та персонажів. Обговорюється важливість використання штучного інтелекту та евристичних алгоритмів для покращення якості та реіграбельності ігор. Генерація базується на різних алгоритмах, зокрема генерації шумів, фрактальній геометрії та клітинних автоматах, що дозволяє створювати варіативні та унікальні рівні з мінімальним втручанням розробників [2].

Одним із найбільш популярних методів процедурної генерації є шум Перліна, розроблений Кеном Перліном у 1983 році [30]. Цей алгоритм створює гладкі перехідні текстури шляхом інтерполяції випадкових значень у просторовій сітці, що робить його незамінним для генерації висотних карт, хмар, води та інших природних елементів у відеоіграх.

Формула для 1D шуму Перліна на основі градієнтів виглядає наступним чином (1.1):

$$P(x) = \sum_{i=-\infty}^{\infty} G(x-i) * fade(x) \tag{1.1}$$

Де G(x) — це градієнт функції для кожної одиничної клітинки (зазвичай, це випадкові значення), fade(x) — це функція згладжування, яка визначає, як значення шуму змінюються в залежності від відстані між точками.

Функція fade використовується для створення більш плавних переходів (1.2):

$$\text{fade}(t) = t^3 \cdot (t \cdot (t \cdot 6 - 15) + 10) \quad (1.2)$$

Інтерполяція між значеннями шуму також є важливою частиною алгоритму (1.3):

$$\text{lerp}(a, b, t) = a + t \cdot (b - a) \quad (1.3)$$

Шум Перліна використовується для створення плавних і природних текстур, зокрема для генерації рельєфів, хмар, океанських хвиль і навіть розподілу рослинності. Його принцип роботи базується на генерації згладжених псевдовипадкових значень у просторовій сітці.

Особливістю алгоритму є те, що він формує згладжені градієнти (див. рис. 1.1) між значеннями, уникаючи різких переходів, які характерні для звичайного випадкового шуму. Для 2D шуму формула розширюється, де координати є двовимірними (1.4):

$$P(x, y) = \sum_{i,j} G(x - i, y - j) \cdot \text{fade}(x - i) \cdot \text{fade}(y - j) \quad (1.4)$$

Популярність шумових алгоритмів серед розробників пояснюється їхньою здатністю генерувати великий обсяг контенту без потреби в детальному ручному моделюванні, а також тим, що вони забезпечують достатній рівень варіативності, що дозволяє створювати унікальні світи для кожної нової гри або рівня.

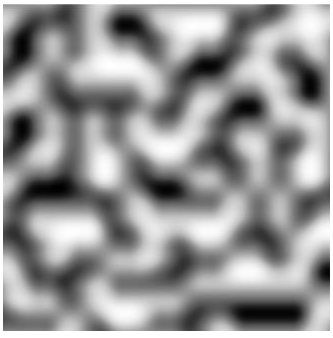

Рисунок 1.1. Приклад згенерованого шуму Перліна

Однією з найбільших переваг алгоритмів шуму є те, що їх можна масштабувати (див. рис. 1.2). Це означає, що шумові алгоритми можна використовувати для створення як малих, так і великих ігрових світів, зберігаючи їх цілісність і природність. Завдання створення масивних відкритих світів вимагає від алгоритму роботи на різних масштабах, від глобального ландшафту до локальних деталей, таких як текстури та елементи навколишнього середовища.

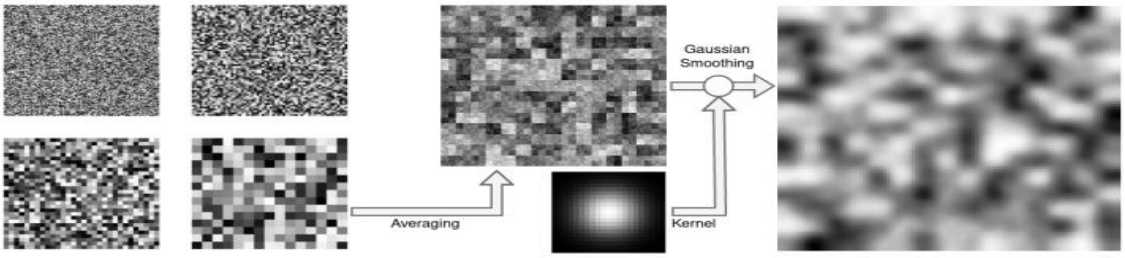

Рисунок 2.2. Усереднення багатомасштабних випадкових матриць

У своїй роботі Перлін підкреслював, що однією з основних переваг його алгоритму є можливість створення плавних переходів між різними значеннями, що імітують природні ландшафти. Подальший розвиток цього підходу призвів до створення Simplex Noise [31], який суттєво оптимізує обчислення, усуваючи артефакти, що виникали на великих масштабах. Варто зазначити, що шум Перліна часто використовується у поєднанні з іншими методами, наприклад, фрактальними алгоритмами, для покращення деталізації рельєфу. Формула для Simplex Noise в 2D виглядає так (1.5):

$$f(x,y) = \sum_{k=0}^{K-1} w_k \cdot G\big(g \cdot (x - x_k)\big) \qquad (1.5)$$

де x — координати точок (x, y), $G\big(g \cdot (x - x_k)\big)$ — функція градієнта, w_k — ваговий коефіцієнт для кожного з внесків.

Градієнтний вектор для Simplex Noise обчислюється як випадковий вектор, орієнтований на один із північних напрямків. Градієнт визначається за такою формулою (1.6):

$$G(g) = g_x \cdot x + g_y \cdot y \qquad (1.6)$$

Де g_x, g_y — це компоненти випадкового вектора градієнта для кожної клітинки.

Для коректної генерації шуму Simplex необхідно виконати певні математичні перетворення для координат (1.7):

$$f = \frac{x + y}{\sqrt{2}} \qquad (1.7)$$

Шумові алгоритми дозволяють ефективно рендерити світи, забезпечуючи природний вигляд і плавні переходи між різними типами середовищ. Таким чином, алгоритми шуму є потужним інструментом для рендерингу різних аспектів ігрового світу, таких як ландшафти, текстури та атмосферні умови.

Фрактали є математичними структурами, які характеризуються властивістю самоподібності – кожна частина об’єкта має подібну структуру до всього об’єкта в цілому. Завдяки цій властивості фрактальні алгоритми широко використовуються для створення складних і реалістичних ігрових ландшафтів, таких як гори, узбережжя, річкові системи, хмари та навіть цілі планети. Одним із найбільш популярних алгоритмів фрактальної генерації є алгоритм розподілу за середнім значенням (midpoint displacement algorithm). Його застосування добре ілюструє, як фрактальна геометрія може створювати нерівності поверхні, схожі на природні.

Метод середньої точки працює за принципом рекурсивного поділу відрізків, до яких додаються випадкові значення, що забезпечує створення плавних, але нерегулярних поверхонь (див. рис. 1.3). Алгоритм діамантово-квадратного поділу є розширенням цього підходу і використовується для двовимірних площин. Такі алгоритми забезпечують ефективність у створенні реалістичних ландшафтів із мінімальними витратами ресурсів [32].

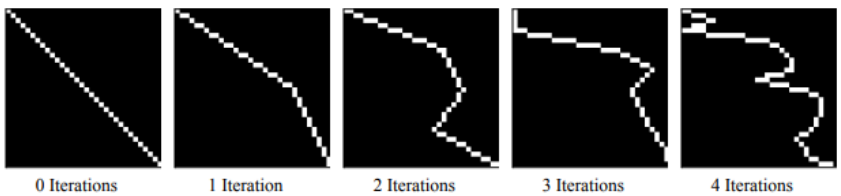

Рисунок 3.3. Наочний приклад роботи фрактального методу

Алгоритм починається з двох крайніх точок на лінії або межах області, що має бути згенерована. Початкові значення на кінцях лінії або квадрата (для 2D простору) задаються випадковими висотами (див. рис. 1.4).

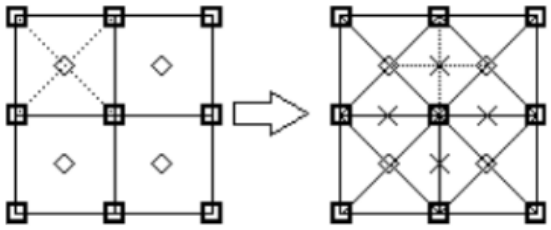

Рисунок 4.4. Метод діамантового квадрату

Нехай точки на межах лінії (для 1D) або квадрату (для 2D) мають значення $P_0 = H_0, \quad P_n = H_n$, де Hо і Hn — це випадкові значення для кожної точки на межах. Для кожного інтервалу між двома сусідніми точками додається нова точка, яка є серединою між двома точками (1.8).

$$P_{\text{mid}} = \frac{P_i + P_{i+1}}{2} + \Delta \tag{1.8}$$

У випадку 2D-генерації (наприклад, для ландшафтів), алгоритм працює на квадратних або прямокутних сітках. Початково ініціалізуються чотири кути квадрата. Для кожного квадрата (або ромба) обчислюється середина (1.9).

$$P_{\text{mid}} = \frac{P_1 + P_2 + P_3 + P_4}{4} + \Delta \tag{1.9}$$

Де P_1, P_2, P_3, P_4 — це кути квадрата (ромба), а Δ — випадкове зміщення. Якщо розглядати кожен крок алгоритму як ітерацію, то для кожної точки на сітці висота може бути виражена як (1.10).

$$H_{\text{new}} = \frac{H_{\text{previous}} + H_{\text{next}}}{2} + \text{random} \times \text{amplitude} \tag{1.10}$$

Де висота нової точки залежить від висоти попередньої і наступної точки на лінії. Алгоритм часто включає в себе коефіцієнт масштабування α, який регулює, наскільки сильно нові точки будуть відрізнятися від сусідніх (1.11).

$$\Delta = \alpha \times \text{random\_value} \tag{1.11}$$

Фрактальна генерація також часто поєднується з іншими алгоритмами для створення багатошарових середовищ. Таке багаторівневе поєднання

дозволяє створювати реалістичні світи, які виглядають логічно і природно. Завдяки своїй універсальності, масштабованості та здатності генерувати реалістичні середовища фрактальна генерація є важливим інструментом для розробників.

Клітинні автомати – це підхід, який поєднує в собі простоту і вражаючу здатність до створення складних структур, що імітують природні або організовані простори. В основі методу лежить ітеративна модель, яка дозволяє згенерувати карту або рівень, починаючи з випадкового розподілу клітин [10]. Завдяки цьому клітинні автомати стають особливо корисними в контексті ігрової розробки, де важливо швидко створювати унікальні, реалістичні світи з мінімальними витратами часу та ресурсів (див. рис. 1.5). Еволюція кожної клітини в клітинному автоматі залежить від станів її сусідів. Це можна записати таким чином (1.12):

$$S_{i,j}^{t+1} = f\left(S_{i-1,j}^{t}, S_{i+1,j}^{t}, S_{i,j-1}^{t}, S_{i,j+1}^{t}, \dots\right) \qquad (1.12)$$

$S_{i,j}^{t}$ — стан клітини на позиції (i,j) в момент часу t, f — функція, що описує правило переходу станів клітини на основі її сусідів.

Рисунок 5.5. Процес створення мапи клітинними автоматами

Популярність клітинних автоматів серед розробників пояснюється їхньою здатністю забезпечувати високий рівень контролю над результатом при одночасній випадковості та варіативності [45]. Одним з найвідоміших прикладів є правило "Життя" Джона Конвея, яке визначає наступні правила для кожної клітини (1.13).

$$S_{i,j}^{t+1} = \begin{cases} 1, \text{якщо } (S_{i,j}^{t} = 0 \text{ та кількість живих сусідів } = 3) \\ 1, \text{якщо } (S_{i,j}^{t} = 1 \text{ та кількість живих сусідів } = 2 \text{ або } 3) \\ 0, \text{в інших випадках} \end{cases} \qquad (1.13)$$

Кількість живих сусідів — це кількість клітин, які знаходяться в стані 1 навколо поточної клітини. Перевага клітинних автоматів полягає в їхній простоті та високій адаптивності. Особливо ефективним цей підхід є у жанрі roguelite-ігор, де випадково згенеровані рівні забезпечують високу реіграбельність та адаптивність геймплею [1]. Вони дозволяють не лише генерувати статичні світи, а й створювати динамічні середовища, які можуть змінюватися в реальному часі відповідно до дій гравця. Це робить клітинні автомати універсальним інструментом для процедурної генерації, який ефективно поєднується з іншими методами, такими як шум Перліна чи фрактальні алгоритми.

1.2 Дослідження ефективності генеративних алгоритмів

Оцінка ефективності генеративних алгоритмів є складним завданням, яке вимагає врахування різних аспектів, таких як час виконання, якість отриманих результатів, а також використання ресурсів, зокрема пам'яті та процесорного часу. У цьому контексті необхідно застосовувати як теоретичні, так і практичні методи оцінки.

Теоретичний аналіз ефективності генеративних алгоритмів базується на математичних моделях, що описують поведінку алгоритмів у певних умовах. Для цього використовуються різноманітні метрики, наприклад, кількість ітерацій, необхідних для досягнення стабільного результату, або час, необхідний для генерації великої кількості даних.

Практичний аналіз полягає в проведенні чисельних експериментів, де порівнюються різні алгоритми за їхнім часом виконання і точністю. Наприклад, для генерування ландшафтів у відеоіграх, таких як Minecraft [26], можна порівнювати алгоритм шуму Перліна з іншими методами за критеріями швидкості генерації карт і їхньої відповідності бажаним характеристикам. Більш детально про техніки процедурної генерації та їх застосування описується у статті [2]. Важливою метою є мінімізація часу виконання при збереженні високої якості результату. Для цього можна використовувати

стратегії паралельної обробки, оптимізації пам'яті або вибору найбільш ефективних алгоритмів для конкретних завдань.

Час виконання алгоритму можна виразити через складність алгоритму, що дозволяє визначити його ефективність (1.14).

$$T(n) = O\big(f(n)\big) \tag{1.14}$$

де $T(n)$ — час виконання, n — кількість елементів для обробки, $f(n)$ — функція складності алгоритму. Для деяких генеративних алгоритмів складність може бути лінійною або навіть підлінійною, що забезпечує високу ефективність при генерації великих обсягів даних. Загальна продуктивність алгоритмів залишається на прийнятному рівні (див. табл. 1.1), і навіть великі карти створюються за лічені секунди.

Таблиця 1.1 Порівняння швидкодії різних алгоритмів генерації

Алгоритм	**Карта 256×256**	**Карта 512×512**	**Карта 1024×1024**
Шум Перліна	0.34 с	1.28 с	5.12 с
Фрактали	0.28 с	1.05 с	4.87 с
Клітинні автомати	0.41 с	1.64 с	6.23 с

Адаптивність генеративних алгоритмів є важливим фактором, оскільки багато з них працюють в умовах зміни параметрів, на основі яких вони генерують нові структури або дані. Важливою характеристикою є здатність алгоритму підлаштовувати свої параметри в залежності від зовнішніх змін або від нових вхідних даних, що дозволяє досягати оптимальних результатів навіть у змінюваних умовах.

У випадку генерації ландшафтів або текстур генетичний алгоритм може адаптувати параметри шуму Перліна або Simplex Noise, покращуючи результат на основі зворотного зв'язку від оцінки якості згенерованих даних. Адаптивність таких алгоритмів може бути виражена через формулу (1.15).

$$A(p) = \frac{E(p)}{T(p)} + \lambda \cdot \Delta p \tag{1.15}$$

де $A(p)$ — адаптивність алгоритму, $E(p)$ — ефективність при параметрах p, $T(p)$ — час виконання алгоритму при цих параметрах, λ — коефіцієнт адаптації, Δp — зміна параметрів, що відображає адаптивність.

У разі процедурної генерації в ігрових світах адаптивність є важливою для створення змінних і динамічних умов, які забезпечують унікальність кожного ігрового процесу. Наприклад, в іграх на кшталт No Man’s Sky [28] генерація планет з використанням фрактальних алгоритмів вимагає високого рівня адаптації до численних варіацій, таких як зміни у фауна, флора та кліматичні умови. Оскільки на основі алгоритмів генеруються трильйони різних планет, адаптивність дозволяє забезпечити, щоб кожен світ виглядав унікальним, навіть при наявності певних спільних характеристик.

Важливим аспектом дослідження ефективності та адаптивності генеративних алгоритмів є їх взаємозв'язок. Алгоритм, який має високу ефективність, здатний швидко обробляти великі обсяги даних, але при цьому він може бути менш адаптивним до зміни зовнішніх умов. З іншого боку, високий рівень адаптивності може супроводжуватись підвищеним споживанням ресурсів і більшою складністю обчислень. Це взаємозв'язок можна моделювати через функцію, яка поєднує обидва аспекти (1.16).

$$F(E, A) = \alpha \cdot E + \beta \cdot A \qquad (1.16)$$

де $F(E,A)$ — загальна ефективність і адаптивність алгоритму, α та β — коефіцієнти, що визначають вагу кожного з аспектів. Збалансування ефективності та адаптивності є важливим для досягнення оптимальних результатів при мінімальних витратах ресурсів.

Для зручності розглянемо три основні підходи: клітинні автомати, фрактальну генерацію та шумові алгоритми (див. табл. 1.2). Кожен із них пропонує свої переваги, недоліки та особливості, які можуть бути визначальними в залежності від типу ігрового середовища.

Таблиця 1.2 Порівняльний аналіз методів генерації

Метод генерації	Переваги	Недоліки	Сфера застосування
Клітинні автомати	- Легкість у створенні складних структур (наприклад, печер) - Простота реалізації правил	- Вимагає ручного налаштування правил - Може бути неприродна структура при неправильних параметрах	Генерація печер, лабіринтів, рельєфів

Метод генерації	Переваги	Недоліки	Сфера застосування
Фрактальна генерація	- Природність у вигляді - Підходить для масштабних ландшафтів	- Вимагає великої обчислювальної потужності - Менше контролю над деталями	Створення гір, річок, планет
Шумові алгоритми	- Простота реалізації - Висока варіативність - Ідеально для природних ландшафтів	- Не підходить для створення складних структур - Вимагає налаштування параметрів	Ландшафти, рівні з природою

Ці методи дозволяють досягти балансу між реалістичністю, керованістю та унікальністю створених середовищ. Вибір підходу залежить від ігрових вимог, апаратних можливостей та бажаного стилю гри.

У статті [17] досліджується підхід до створення пазлів за допомогою генеративних алгоритмів, що гарантує збалансованість складності та реіграбельність рівнів. Використання процедурних методів дозволяє автоматично генерувати логічні задачі, що відповідають заданим критеріям складності та унікальності. Дослідження [22] розглядає новітні підходи до генерації середовищ у різних жанрах, включаючи стратегічні ігри та платформери, де алгоритмічні рішення дозволяють адаптувати ігровий процес під гравця, створюючи щоразу унікальний досвід. У статті [41] запропоновано методи створення поселень, що використовують алгоритми на основі штучного інтелекту для адаптивного розташування структур, що підвищує реалістичність та інтеграцію в ігровий світ.

У сучасному ігровому дизайні існує безліч програмних рішень, які дають змогу використовувати різні методи генерації. Кожна з таких програм має свої особливості, що визначають її ефективність у певних сценаріях використання. Деякі інструменти орієнтовані на створення реалістичних ландшафтів, тоді як інші спеціалізуються на створенні лабіринтів, підземель чи цілих планет. У додатку розглянуто деякі з програм, які підтримують різні підходи до процедурної генерації та їхню роль у розробці ігрових світів (див. Додаток).

Висновок до розділу 1

У рамках дослідження були проаналізовані основні методи процедурної генерації, серед яких шумові алгоритми, клітинні автомати та фрактальна

генерація, кожен з яких має свої унікальні характеристики, переваги та обмеження.

Шумові алгоритми, зокрема алгоритм Перліна, відзначаються своєю здатністю створювати плавні та природні переходи в ландшафтах, що особливо важливо для моделювання природних об’єктів, таких як гори, річки та моря. Клітинні автомати, з іншого боку, дозволяють генерувати складні та детализовані структури, що є корисними для створення рівнів, підземель або міст. Фрактальна генерація, завдяки своїй здатності моделювати складні геометричні форми через прості математичні правила, є особливо ефективною для створення реалістичних природних ландшафтів, таких як гори, ріки, озера та інші географічні об’єкти.

Вибір методу генерації безпосередньо залежить від вимог проекту, і використання комбінацій різних методів дозволяє досягти більш ефективних та багатих результатів. Процедурні алгоритми забезпечують ефективну автоматизацію створення ігрових світів, знижують необхідність ручної роботи та підвищують варіативність контенту. Наприклад, використання шумових функцій у поєднанні з клітинними автоматами дозволяє створювати більш складні та реалістичні ігрові рівні, а застосування фрактальних алгоритмів — генерувати точні та деталізовані рельєфи.

Процедурна генерація дає змогу значно зменшити витрати часу та ресурсів на розробку великих ігрових світів та контенту, забезпечуючи при цьому високу варіативність та динамічність ігрового процесу. Інтеграція цих методів в ігрові рушії дозволяє розробникам створювати інтерактивні, масштабовані світи, що змінюються в реальному часі, і адаптуються до дій гравця.

Враховуючи наявні досягнення в галузі машинного навчання та нейронних мереж, перспективи розвитку процедурної генерації виглядають дуже обнадійливими. Використання нейронних мереж для вдосконалення процедурних алгоритмів дає змогу створювати ще більш реалістичні та

адаптивні ігрові середовища, що підвищує загальний рівень якості та інтерактивності в ігрових світах.

Таким чином, сучасні інструменти та методи процедурної генерації відкривають нові можливості для розробників ігор, дозволяючи їм створювати більш реалістичні, складні та масштабовані світи за допомогою ефективних алгоритмів. Подальші дослідження у цій сфері можуть привести до ще більших інновацій, зокрема в напрямку інтеграції штучного інтелекту та нейронних мереж, що дозволить створювати динамічніші та більш адаптовані до дій гравців світи.

РОЗДІЛ 2. ЗАСТОСУВАННЯ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ У ПРОЦЕДУРНІЙ ГЕНЕРАЦІЇ

2.1 Використання нейронних мереж у генеративних алгоритмах

Машинне навчання значно розширило можливості процедурної генерації контенту, дозволяючи системам адаптивно генерувати контент на основі вивчених патернів. Використання нейронних мереж у генеративних алгоритмах є значним кроком уперед у сфері комп'ютерних наук і має широке застосування у створенні складного контенту. Зокрема, методи глибинного навчання, такі як генеративні змагальні мережі (GAN), автоенкодери та інші моделі, використовуються для генерації реалістичних зображень, текстур, ландшафтів та інших елементів, що мають великий потенціал для таких галузей, як відеоігри, моделювання та наукові дослідження [19]. У статті [3] розглядається, як саме використовуються нейронні мережі у процедурній генерації ігрових світів.

Генеративно-змагальні мережі (GAN) є потужним інструментом для створення нових об’єктів, текстур та навіть рівнів у відеоіграх. Дослідження [15] пояснює механізм роботи GAN, у якому дві нейромережі (генератор і дискримінатор) змагаються між собою, покращуючи якість синтезованих даних, що має велике значення для розширення можливостей процедурної генерації. Архітектура GAN складається з двох компонентів: генератора, який створює дані, і дискримінатора, що оцінює їх реалістичність (див. рис. 2.1).

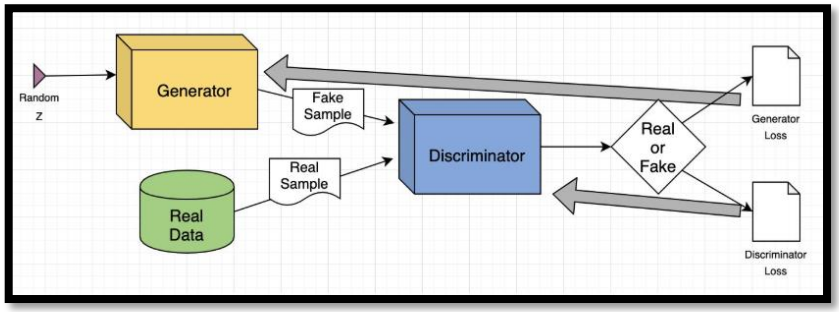

Рисунок 2.1. Архітектура генеративних змагальних мереж

Цей підхід продемонстрував чудові результати у генерації зображень, але також став основою для вирішення завдань процедурної генерації

контенту. GAN дозволяють генерувати не лише окремі елементи, як зображення або текстури, але й цілі віртуальні світи з різноманітними ландшафтами та географічними елементами, такими як острови, річки, гори тощо. Так, у роботі [16] було показано, як використання GAN здатне генерувати нові зображення на основі реалістичних даних, що може бути адаптовано для створення складних ландшафтів для відеоігор або віртуальних середовищ. Розглянемо формулу для тренування, яку використовує GAN (2.1).

$$L_{GAN} = E_{x\sim p_{data}}[\log D\,(x)] + E_{z\sim p_z}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \qquad (2.1)$$

Ця функція втрат є основною для тренування GAN, де $D(x)$ — це дискримінатор, що оцінює ймовірність того, що зразок x є реальним, а $G(z)$ — це генератор, що створює нові зразки на основі випадкового шуму z (рис. 2.2).

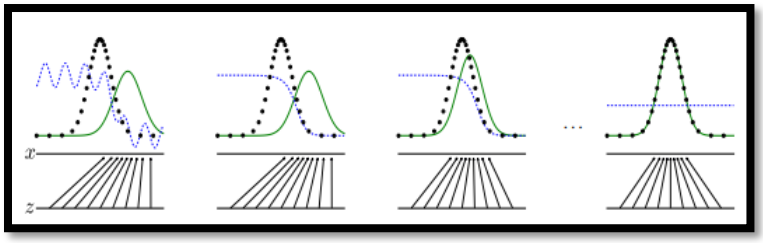

Рисунок 2.2. Результати тренування GAN

Wasserstein GAN (WGAN) є покращеною версією звичайної генеративно-змагальної мережі, що використовує функцію відстані Васерштейна для більш стабільного та якісного навчання.

Однак забезпечення того, щоб згенерований контент відповідав конкретним вимогам гри, таким як розв’язуваність чи складність, залишається викликом. Стаття [42] представляє підхід до покращення умовної генерації рівнів у пазл-іграх за допомогою автоматизованої валідації. Ми тренуємо умовну GAN на наборі даних розв’язуваних рівнів та включаємо механізм автоматизованої валідації під час тренування для фільтрації невалідних рівнів. Результати показують, що цей підхід значно покращує якість та розв’язуваність згенерованих рівнів порівняно з базовою GAN без валідації.

За останні роки також було проведено дослідження з використанням генеративних змагальних мереж, однак GAN загалом не дуже добре

підтримують обмеження, необхідні для контенту, використовуваного в іграх [33]. У статті [6] розглядається спосіб мінімізації нестабільності навчання та покращення якості створюваних даних, що може бути корисним у процедурній генерації ігрових світів та текстур.

Але GAN не є єдиним методом у процедурній генерації. Автоенкодери, які спочатку використовувалися для стиснення та відновлення даних, також можуть бути ефективно застосовані для генерування нових структур. Одним із таких підходів є використання варіаційних автоенкодерів (VAE) (рис. 2.3), які можуть навчатися на великих наборах даних і генерувати нові варіанти карт, текстур або інших елементів. Наприклад, у роботах [21] було запропоновано автоенкодери для моделювання ймовірності генерації даних, що дозволяє генерувати нові варіанти ландшафтів, текстур або 3D-об'єктів, що є основою для процедурної генерації віртуальних світів. Вони використовують ймовірнісні моделі для побудови представлень, що можуть бути згенеровані на основі навчальних даних і дозволяють генерувати високоякісні результати (2.2).

$$q_\phi(z|x) = \mathcal{N}(z; \mu(x), \sigma(x)^2) \tag{2.2}$$
$$p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \sigma_\theta(z)^2)$$

Ці рівняння описують процеси варіаційного автоенкодера, де $q_\phi(z|x)$ є апостеріорним розподілом для латентної змінної z, а $p_\theta(x|z)$ — ймовірнісним розподілом для відновлення вихідних даних x.

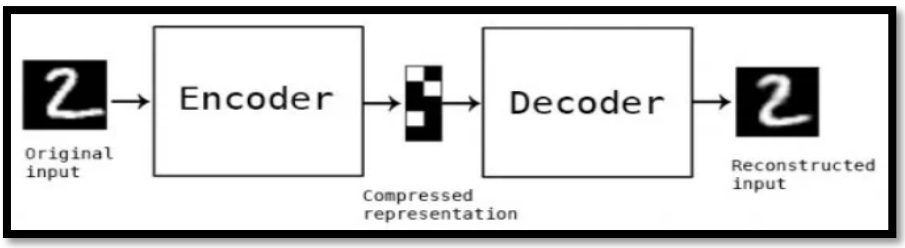

Рисунок 2.3. Приклад роботи автоенкодерів

Автоенкодери є важливим інструментом для зменшення розмірності даних і генерації нових зразків контенту. У дослідженні [20] пояснюється механізм роботи автоенкодерів, їхня роль у створенні реалістичних текстур та

рівнів у відеоіграх, а також можливості застосування у стисканні та реконструкції ігрових середовищ.

З іншого боку, комбінування шумових алгоритмів, таких як Perlin noise або Simplex noise, з нейронними мережами також виявилося корисним для генерації реалістичних ландшафтів. Використання Perlin noise дає змогу генерувати плавні перехідні лінії між різними частинами карти, створюючи таким чином природніший вигляд ландшафтів. Коли цей підхід поєднується з нейронними мережами, результат може бути значно покращений. В дослідженнях [43] було продемонстровано використання нейронних мереж для створення процедурно генерованого контенту, зокрема, текстур і ландшафтів, що адаптуються до заданих умов, таких як клімат чи тип місцевості. Формула (2.3) описує комбінований шумовий сигнал, який поєднує класичний шум Перліна із шумом, згенерованим генеративною змагальною мережею

$$\mathrm{Noise}_{\mathrm{final}} = \mathrm{Noise}_{\mathrm{Perlin}} + \alpha \times \mathrm{Noise}_{\mathrm{GAN}} \qquad (2.3)$$

де $\mathrm{Noise}_{\mathrm{final}}$ — фінальний шумовий сигнал, що використовується в процедурній генерації. $\mathrm{Noise}_{\mathrm{Perlin}}$— класичний шум Перліна, який забезпечує згладжені, природні зміни висот або текстур. $\mathrm{Noise}_{\mathrm{GAN}}$ — шум, згенерований генеративною змагальною мережею, що додає варіативність, враховуючи навчання на великій вибірці реальних даних. α — ваговий коефіцієнт, що регулює рівень впливу $\mathrm{Noise}_{\mathrm{GAN}}$ на фінальний шум.

Традиційні методи процедурної генерації контенту часто покладаються на правило-базовані або випадкові техніки генерації, які, хоча й ефективні до певної міри, можуть не мати адаптивності та не забезпечувати грабельність або баланс. В останні роки підйом глибокого навчання з підкріпленням (DRL) дозволив розробку систем, які можуть вчитися з взаємодій з їхнім середовищем, пропонуючи підхід, керований даними, до генерації контенту [7], [12].

Зокрема, недавні досягнення в ШІ, особливо в машинному навчанні, ще більше розширили потенціал процедурної генерації контенту. Глибокі нейронні мережі сприяють генерації різноманітного ігрового контенту, від ігрових середовищ і поведінки персонажів до створення наративів [24]. Сучасні відеоігри вимагають великої кількості медіа високої чіткості, які потенційно можуть генеруватися за допомогою підходів глибокого навчання. Наприклад, перспективні недавні методи генерації фотореалістичних облич можуть використовуватися для створення персонажів в іграх [23].

Незважаючи на значний прогрес у використанні нейронних мереж для генерації процедурного контенту, існують певні обмеження, які потребують вирішення. Одним із основних викликів є велика потреба в обчислювальних ресурсах для тренування моделей, а також необхідність великих наборів навчальних даних для досягнення бажаного рівня якості. Крім того, інтеграція різних типів нейронних мереж з традиційними процедурними алгоритмами потребує ретельної налаштування та оптимізації. Однак з розвитком технологій і підвищенням обчислювальних потужностей ці проблеми можуть бути подолані.

2.2 Вдосконалення методів процедурної генерації на основі нейромережевого навчання

Процедурна генерація контенту є основною складовою в багатьох галузях, зокрема у відеоіграх, симуляціях та віртуальних світах, де необхідно генерувати великі та складні простори або об'єкти без участі людини. Одним із основних викликів для традиційних методів є створення контенту, що відповідає вимогам реалістичності та адаптивності. Оскільки класичні алгоритми, такі як шумові функції (наприклад, Perlin noise), дозволяють генерувати ландшафти на основі математичних рівнянь, вони часто не можуть ефективно враховувати складні взаємодії між різними елементами світу, що виникають у реальних умовах. Проблеми виникають також через обмежену здатність традиційних алгоритмів до адаптації і навчання на нових даних.

В останні роки значний прогрес у вдосконаленні процедурної генерації досягнутий завдяки використанню сучасних інструментів машинного навчання, таких як TensorFlow та ML-Agents. Використання глибинних нейронних мереж, таких як генеративні змагальні мережі (GAN), варіаційні автоенкодери (VAE), а також методи навчання з підкріпленням (RL), дозволяють створювати більш складні і адаптивні системи генерації контенту.

TensorFlow — це популярний фреймворк для створення та тренування моделей машинного навчання, який підтримує великі набори даних та численні типи нейронних мереж. Однією з основних переваг TensorFlow є можливість використовувати його для навчання нейронних мереж на великих обсягах даних (див. рис. 2.4), що дозволяє створювати високоякісні ландшафти, текстури та інші елементи контенту. Завдяки цим можливостям, нейронні мережі, побудовані на TensorFlow, можуть комбінувати традиційні алгоритми (наприклад, Perlin noise) з новими методами глибинного навчання для створення більш природних та варіативних ландшафтів.

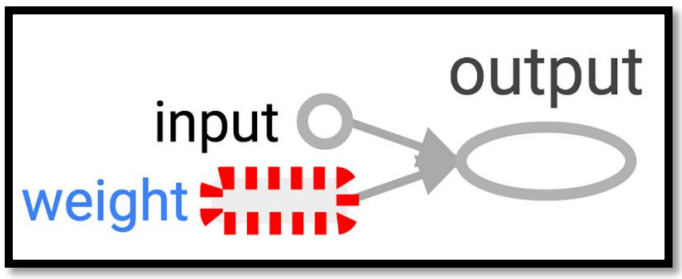

Рисунок 2.4. Візуалізація найпростішого нейрону TensorFLow

Наприклад, генеративні змагальні мережі (GAN), які працюють у рамках TensorFlow, можуть створювати нові варіанти контенту на основі великих наборів даних. Вони можуть навчатися генерувати ландшафти, що відрізняються від класичних шумових структур, надаючи більше варіативності та складності.

ML-Agents є платформою для тренування агентів у віртуальних середовищах, що дозволяє інтегрувати методи навчання з підкріпленням для оптимізації процедурної генерації контенту [39]. Вона забезпечує взаємодію агентів з навколишнім середовищем (див. рис. 2.5), що дозволяє агентам

самостійно вивчати, як створювати контент на основі взаємодій з різними факторами (географічними, екологічними тощо).

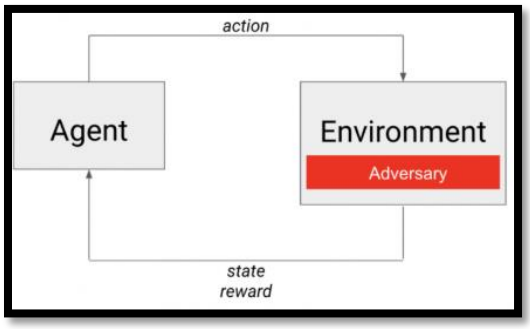

Рисунок 2.5. Цикл тренування агентів в ML-Agents

ML-Agents дозволяє навчати штучний інтелект у симуляціях, що використовується для вдосконалення поведінки NPC та інших ігрових об’єктів. У статті [8] розглядається налаштування ML-Agents у Python з TensorFlow [38] для навчання агентів у середовищі Unity, що відкриває нові можливості в автоматизації ігрових процесів.

У випадку процедурної генерації ландшафтів, ML-Agents може використовувати навчання з підкріпленням для оптимізації параметрів генерації. Агент може вивчати, як змінювати параметри, такі як висота, текстури або типи місцевості, для досягнення найбільш реалістичних результатів. Основна ідея полягає в тренуванні моделі на прикладах, відібраних з деякого розподілу, а потім використанні цієї моделі для виробництва нових зразків [37]. У дослідженні [29] пропонується новий метод процедурного дизайну рівнів за допомогою глибокого навчання з підкріпленням (DRL) в середовищі на основі Unity 3D. Система складається з двох агентів: агента, який діє як розв’язувач, та агента, відповідального за генерацію та розміщення колекційних об’єктів (квіток) на терені в реалістичний та контекстно-усвідомлений спосіб. Колібрі тренується за допомогою алгоритму Proximal Policy Optimization (PPO) з набору інструментів Unity ML-Agents.

Одним із прикладів може бути використання глибинних нейронних мереж або Q-навчання (2.4) для оптимізації цих параметрів, де функція

винагороди $Q(s,a)$ допомагає агенту оцінювати ефективність обраної стратегії генерації на основі якості отриманого контенту [34].

$$Q(s,a) = Q(s,a) + \alpha\left(R(s,a) + \gamma \max_{a'} Q\,(s',a') - Q(s,a)\right) \qquad (2.4)$$

де $Q(s,a)$ — це функція корисності для станyss і дії a, $R(s,a)$ — винагорода за виконану дію, а γ — коефіцієнт дисконтування для майбутніх винагород. Завдяки ML-Agents агент може вивчати найкращі стратегії генерації контенту, базуючись на отриманих результатах і адаптуючи свої дії для досягнення бажаного результату.

Глибоке навчання, що використовується у Unity ML-Agents, дозволяє створювати складні поведінкові моделі для віртуальних агентів. Дослідження [35] демонструє, як використання підходів глибокого підкріпленого навчання може допомогти створювати більш реалістичних і автономних NPC, здатних адаптуватися до змін у середовищі. Використання TensorFlow та ML-Agents у процедурній генерації надає значні переваги, серед яких:

- Покращення якості та різноманітності контенту, оскільки нейронні мережі можуть навчатися на великих обсягах даних, створюючи більш складні та реалістичні ландшафти.
- Адаптивність до нових умов, завдяки чому можна легко змінювати параметри генерації, враховуючи різні екологічні та географічні фактори.
- Зниження потреби в ручному налаштуванні параметрів за рахунок автоматичного навчання моделей.

Нейронні мережі, треновані за допомогою TensorFlow або ML-Agents, можуть поєднуватися з традиційними шумовими алгоритмами, такими як Perlin noise або Simplex noise. Шумові алгоритми використовуються для створення природних, плавних переходів між різними частинами ландшафтів. Комбінація цих методів дозволяє нейронним мережам створювати більш реалістичні та різноманітні результати, враховуючи як алгоритми шуму, так і знання, отримані через навчання на великих наборах даних.

Проте ці методи також мають певні виклики:

- Великі обчислювальні ресурси для тренування моделей, що потребує використання спеціалізованих апаратних засобів, таких як графічні процесори (GPU) або TPU.
- Необхідність великих навчальних наборів даних для ефективного тренування моделей.
- Складність оптимізації та інтеграції різних підходів, що вимагає значного часу на налаштування і тестування.

Попри численні виклики, використання TensorFlow та ML-Agents у процедурній генерації має значний потенціал для вдосконалення створення контенту в реальному часі. З розвитком обчислювальних потужностей і покращенням архітектур нейронних мереж, ці технології зможуть надати ще більше можливостей для генерації складних, реалістичних і динамічних віртуальних світів. Це відкриває нові горизонти в ігровій індустрії, наукових симуляціях та віртуальних середовищах.

2.3 Перспективи розвитку нейромережевих генеративних моделей

Процедурна генерація контенту в ігровій індустрії значно еволюціонувала завдяки впровадженню нейромережевих підходів. Використання генеративних змагальних мереж (GAN), варіаційних автоенкодерів (VAE) та глибинного навчання дозволяє створювати більш реалістичні та варіативні середовища у порівнянні з традиційними методами шумової генерації. Оцінка ефективності таких підходів ґрунтується на аналізі якості, швидкодії та можливостей адаптації до різних типів ігрових світів. Важливим фактором є також їх масштабованість та стійкість до змін у процесі генерації контенту. Сучасні дослідження демонструють, що поєднання класичних алгоритмів процедурної генерації з методами машинного навчання дозволяє значно покращити реалістичність та контрольованість результатів.

Нейромережеві генеративні моделі дозволяють розробникам створювати складні, унікальні та реалістичні віртуальні середовища без необхідності ручного проектування кожного окремого елемента.

Використання таких підходів значно зменшує трудовитрати, дозволяючи створювати великі світи в значно коротші терміни. Одним із ключових досягнень останніх років стало використання генеративних мереж для створення складних природних ландшафтів, архітектурних структур та навіть процедурної генерації сюжетів, що відкриває нові можливості в розвитку інтерактивних ігор.

Для визначення ефективності нейромережевих моделей у процедурній генерації контенту, проведено порівняння основних підходів за кількома ключовими параметрами, такими як реалістичність, обчислювальна складність та масштабованість (див. табл. 2.1).

Таблиця 2.1 Порівняльний аналіз різних методів процедурної генерації

Метод	**Реалістичність**	**Обчислювальна складність**	**Гнучкість у навчанні**	**Масштабованість**
Perlin Noise	Низька	Низька	Відсутня	Висока
Cellular Automata	Середня	Низька	Відсутня	Висока
GAN	Висока	Висока	Висока	Середня
VAE	Висока	Середня	Висока	Середня

Попри це, перспектива розвитку процедурної генерації з використанням нейромереж виглядає дуже обнадійливо. Одним із ключових напрямів є інтеграція нейромереж з іншими технологіями, зокрема підкріпленим навчанням і трансферним навчанням. Це дозволяє досягати кращих результатів у генерації, а також застосовувати моделі на практиці без необхідності великої кількості навчальних даних.

Процедурна генерація ландшафту є основою комп’ютерної графіки та ігор протягом десятиліть, з техніками від функцій шуму до симуляції ерозії. У цій статті [25] досліджується використання нейронних мереж для покращення процедурної генерації ландшафту, зокрема фокусуючись на варіантах шуму Перліна, інтегрованих з моделями глибокого навчання для більш реалістичних та різноманітних ландшафтів. Нейронні мережі дозволяють вчитися складним патернам з даних реального світу, покращуючи традиційні методи шляхом генерації теренів, які адаптуються до конкретних вимог гри.

Ще одним перспективним напрямом розвитку є підвищення контролю над генерацією. Використання нейромереж, які здатні навчатися на стилістичних особливостях конкретних художників або дизайнерських концепцій, дозволяє створювати рівні та світи, які відповідають заданим параметрам, включаючи атмосферу, композицію та наративний стиль. Це особливо важливо для ігор із відкритим світом, де рівні повинні залишатися унікальними, але водночас гармоніювати з загальним художнім баченням проєкту. Дослідження [27] демонструє, що генеративний ШІ може вийти за межі традиційних застосувань і стати інтегральним, інтерактивним елементом у відеоіграх. Оскільки технологія ШІ продовжує еволюціонувати, її потенціал у дизайні ігор є величезним, пропонуючи нові способи створення динамічних, керованих гравцем досвідів.

Однією з найбільш перспективних тенденцій є розвиток моделей, що дозволяють оптимізувати обчислювальні ресурси. Завдяки новим алгоритмам та потужнішим апаратним засобам, нейромережі зможуть генерувати контент більш ефективно, зменшуючи час на тренування моделей і покращуючи якість результатів. Це дозволить застосовувати їх на більш широкому спектрі пристроїв, у тому числі в реальному часі, без значних затрат ресурсів. У майбутньому також очікується, що більше моделей включатимуть текстовий ввід та механізми зворотного зв’язку. Генеративний ШІ надає різноманітні методи для PCG, показуючи потенціали та прогалини в дослідженнях для дослідників чи розробників ігор [11].

Загалом, перспективи розвитку процедурної генерації з використанням нейромереж є дуже обнадійливими. Вони відкривають нові можливості для створення високоякісного, адаптивного та персоналізованого контенту, що має велике значення як для індустрії розваг, так і для багатьох інших сфер, таких як симуляції, планування міст, архітектурне проектування та багато інших. У майбутньому можна очікувати, що технології нейромереж в галузі процедурної генерації будуть не тільки розвиватися, а й досягатимуть нових рівнів інтерактивності та адаптивності, створюючи реалістичніші і більш персоналізовані віртуальні середовища.

Висновок до розділу 2

Узагальнюючи результати, викладені в попередніх розділах, можна стверджувати, що використання нейронних мереж у процедурній генерації контенту має великий потенціал для подальшого розвитку та удосконалення технологій, що використовуються в сучасній ігровій індустрії та віртуальних середовищах. Традиційні підходи до процедурної генерації контенту, засновані на алгоритмах шуму, таких як Perlin noise, Cellular Automata та інші, мають свої обмеження. Вони, хоча й дають непогані результати для створення ландшафтів та текстур, не здатні повною мірою враховувати складність взаємодій між елементами світу, що обмежує можливості для створення високоякісного та адаптивного контенту. Водночас, новітні підходи, засновані на нейронних мережах, таких як генеративні змагальні мережі (GAN) та варіаційні автоенкодери (VAE), дозволяють значно покращити реалістичність та варіативність генерованих середовищ, забезпечуючи вищий рівень адаптації до різних умов і вимог.

Одним із найбільших досягнень у застосуванні нейронних мереж для процедурної генерації є здатність цих мереж виводити складні та високоякісні віртуальні середовища без потреби в ручному моделюванні кожного елемента. Це не тільки значно зменшує час розробки, але й дозволяє створювати більші і складніші ігрові світи в коротші терміни, що є важливою перевагою для індустрії, де швидкість розробки має велике значення. Такі підходи дозволяють створювати різноманітні типи контенту, починаючи від природних ландшафтів і закінчуючи складними архітектурними структурами, а також навіть генерувати сюжетні лінії для ігор.

Застосування нейронних мереж дозволяє зменшити трудозатрати, пов'язані з процесом створення контенту, дозволяючи розробникам зосередитися на інших аспектах ігрового процесу. Це, в свою чергу, дає можливість знижувати витрати на створення великих ігрових світів, що є важливим чинником в умовах постійної конкуренції в індустрії. Однак, попри значні переваги, застосування нейронних мереж у процедурній генерації контенту не є безперешкодним. Одним з основних викликів є потреба у великих наборах даних для тренування моделей, що може бути

проблематичним, особливо для специфічних контекстів, де доступ до відповідних даних обмежений.

Також варто зазначити, що тренування складних нейронних мереж потребує значних обчислювальних ресурсів, що може стати серйозною перешкодою для широкого застосування таких підходів у малих студіях або для персональних проектів. Проте, з розвитком апаратного забезпечення, таких як графічні процесори (GPU) і спеціалізовані модулі для машинного навчання, ці обмеження поступово знижуються, що дає підстави для оптимізму щодо застосування нейромереж у процедурній генерації контенту в майбутньому.

Перспективи для подальшого розвитку в цій сфері є обнадійливими. В майбутньому, завдяки покращенню алгоритмів та апаратного забезпечення, нейронні мережі зможуть ще більш ефективно генерувати контент, зменшуючи час тренування моделей і покращуючи якість результатів. Це дозволить створювати більш складні та реалістичні віртуальні світи, що значно підвищить рівень інтерактивності та адаптивності віртуальних середовищ для користувачів.

Загалом, можна зробити висновок, що нейронні мережі представляють собою потужний інструмент для розвитку процедурної генерації контенту, і їх застосування обіцяє значні зміни в підходах до створення віртуальних середовищ у майбутньому. Ці технології дозволяють не тільки створювати реалістичніші та більш адаптивні ігрові світи, але й забезпечують нові можливості для персоналізації контенту, що є важливим для задоволення вимог сучасних користувачів та розвитку індустрії в цілому.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1. Формалізація вимог до програмного забезпечення

У процесі розробки програмного продукту важливим етапом є збір, аналіз та систематизація вимог до системи. Чітке визначення вимог дозволяє забезпечити відповідність створеного продукту очікуванням користувачів, а також уникнути помилок на етапах проєктування та реалізації. Саме тому особлива увага була приділена формулюванню функціональних та нефункціональних вимог, які визначають ключові аспекти роботи системи.

Для створення ефективної системи генерації ігрового світу важливо врахувати потреби користувачів, орієнтуючись на сучасні тенденції у галузі розробки ігор. Генерація світу повинна бути не тільки функціональною, але й інтуїтивно зрозумілою, щоб полегшити взаємодію кінцевого користувача із системою. Значна увага приділяється масштабованості системи, яка дозволяє ефективно працювати з різними обсягами даних та підтримувати проєкти різного масштабу. Іншими важливими критеріями є стабільність роботи, що передбачає мінімізацію помилок та збоїв, а також зручність інтерфейсу, яка полегшує навчання і роботу із системою.

Для більш детальної конкретизації вимог до програмного продукту нижче представлена таблиця (див. табл. 3.1), що містить ключові вимоги системи із зазначенням їх категорій, пріоритетів, критеріїв прийомки та методів перевірки.

Таблиця 3.1 Вимоги до програмного продукту

Код	Вимога	Категорія	Пріоритет	Критерій прийомки	Метод перевірки
1	Алгоритм повинен створювати різноманітні ландшафти	Функціональна	Високий	Створюються світи з різними біомами	Тестування генерації
2	Система повинна підтримувати налаштування параметрів	Функціональна	Середній	Інтерфейс дозволяє задавати параметри перед запуском	Функціональ не тестування

Код	Вимога	Категорія	Пріоритет	Критерій прийомки	Метод перевірки
3	Генерація світу повинна бути ефективною за часом	Нефункціональна	Високий	Час генерації не перевищує 10 секунд для карт середнього розміру	Замір часу роботи
4	Забезпечення стабільної роботи системи	Нефункціональна	Високий	Система не видає критичних помилок протягом 10 тестів	Навантажувальне тестування
5	Масштабованість для роботи з великими картами	Нефункціональна	Високий	Система стабільно працює із картами 1024x1024 одиниць	Замір ресурсів під час роботи

Розробка програмного продукту потребує створення моделей, які описують його структуру, функціональність та взаємозв'язки між компонентами. Це дозволяє розробникам, аналітикам і замовникам краще розуміти архітектуру системи, що спрощує процес розробки та подальшої підтримки. Крім того, використання моделей на ранніх етапах проєктування дає змогу своєчасно виявити можливі проблеми та усунути їх ще до початку реалізації.

Одним із ключових етапів розробки є побудова концептуальної та логічної моделей. Концептуальна модель визначає загальну структуру системи та її основні компоненти без заглиблення в деталі реалізації. Основна мета концептуального моделювання – створення зрозумілої візуальної структури, яка допоможе на етапах аналізу, проектування та комунікації між розробниками та замовниками. Оскільки система призначена для процедурної генерації ігрових рівнів, до її складу увійдуть такі основні компоненти:

- Менеджер генерації світу – відповідальний за вибір алгоритму генерації та управління його параметрами.
- Модуль шумових алгоритмів – містить реалізації алгоритмів Перліна, фрактального шуму та клітинних автоматів.
- Компонент візуалізації – забезпечує відображення згенерованої карти.
- Інтерфейс користувача – дозволяє змінювати параметри генерації та спостерігати за результатами.

Головною перевагою такого підходу є можливість створення гнучкої, розширюваної та легко підтримуваної системи. Чітке розмежування обов’язків між модулями дозволяє легко змінювати або покращувати окремі компоненти без значного впливу на всю архітектуру. Наприклад, якщо потрібно оновити алгоритм генерації шуму або додати підтримку нових типів ландшафтів, це можна зробити без масштабного переписування коду, що значно економить час і ресурси розробників.

У процесі розробки програмного продукту ключовим рішенням є вибір платформи, яка забезпечить необхідний рівень продуктивності, гнучкості та сумісності з вимогами проєкту. Для створення програмного продукту було обрано Unity – один із найпопулярніших рушіїв у світі розробки ігор і інтерактивних застосунків [40]. Це рішення обґрунтоване низкою факторів (див. табл. 3.2), які роблять Unity оптимальним варіантом для реалізації задуманої системи.

Таблиця 3.2 Порівняння рушіїв

Критерій	**Unity**	**Unreal Engine**	**Godot**
Мова програмування	C#	C++, Blueprint	GDScript, C#
Кросплатформність	+	+	+
Графічні можливості	Високі	Високі	Середні
Процедурна генерація	+	+	+
Простота у використанні	+	-	+
Відкрите ПЗ	-	-	+

Ключовим аспектом вибору стала можливість реалізації процедурної генерації ігрового світу, яка є центральним елементом проєкту. Unity надає широкий спектр інструментів для створення випадково згенерованих рівнів, ландшафтів і об'єктів.

3.2 Алгоритмічне забезпечення та реалізація процедурної генерації

Процес створення процедурної генерації в моїй роботі є результатом поєднання класичних алгоритмів з сучасними методами на основі нейронних мереж. Завдяки цьому підходу я отримую високоякісні, варіативні та реалістичні ігрові світи, здатні адаптуватися під конкретні умови або вимоги проєкту.

На першому етапі система створює шумову карту — двовимірний масив значень висот, який визначає особливості рельєфу. Використовуючи алгоритми основних методів процедурної генерації або ж з використанням генеративно-змагальної нейромережжі, генератор формує плавні переходи між різними зонами. На основі цих даних визначаються області для лісів, гір, водойм та інших біомів.

Для досягнення більш високого рівня адаптивності та реалістичності контенту я переходжу до більш складного етапу — інтеграції TensorFlow і створення локальної нейронної мережі. На цьому етапі я застосовую методи машинного навчання для покращення якості та адаптивності генерованого контенту (див. рис. 3.1). Моя нейронна мережа здатна вдосконалювати згенеровані карти, змінюючи їх відповідно до заданих параметрів.

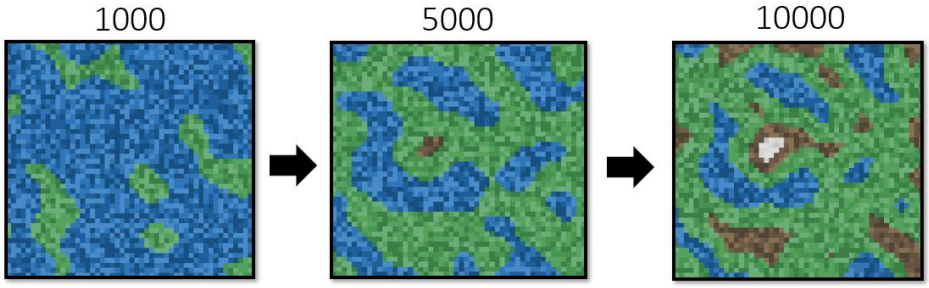

Рисунок 3.1. Порівняння генерації від кількості ітерацій навчання

Агент отримує позитивні чи негативні сигнали в залежності від того, наскільки добре генерована карта відповідає заданим критеріям, таким як рівень деталізації, природність ландшафту чи відповідність специфікаціям проекту. З часом агент починає генерувати карти, які стають все більш адаптованими до конкретних умов, на яких проходить навчання.

Крім того, програмний продукт дозволяє інтегрувати систему для перетворення двовмиріної сітки висот у тривимиріну модель (див. рис. 3.2), це дозволить застосовувати генерацію не тільки для створення двовимірних проєктів, а й для тривимірних. Програма автоматично відображає результати генерації у тривимірній інтерпретації, надаючи можливість швидкого аналізу якості карт та їх відповідності вимогам проекту.

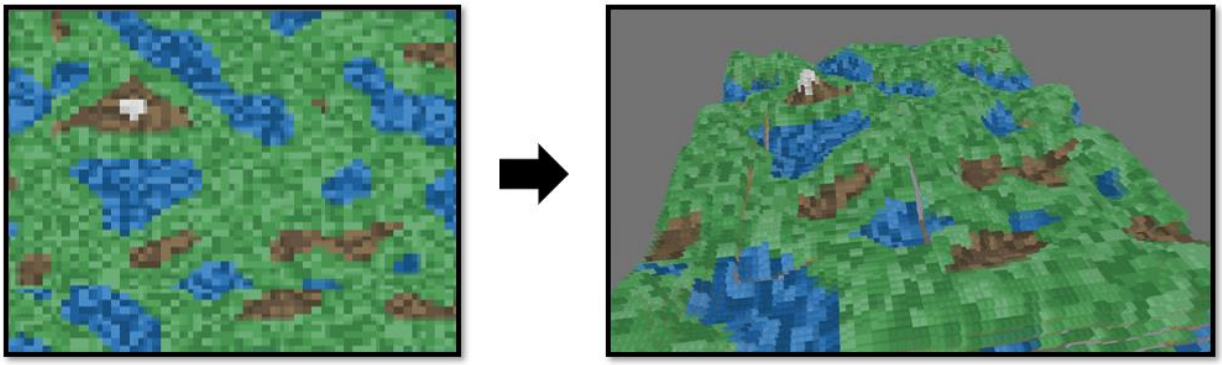

Рисунок 3.2. Перетворення двовимірної сітки висот у тривимірну модель

Створення інтерфейсу було спрямоване на забезпечення простоти налаштувань параметрів генерації та інтеграції нейронної мережі. У програмі є можливість вибору типу генерації, глибини рельєфу, та інших ключових параметрів, що визначають зовнішній вигляд генерованої карти. Така гнучкість дозволяє адаптувати процес генерації під конкретні вимоги проекту або ігрові механіки. Для користувача передбачена панель налаштувань, де можна вручну коригувати параметри генерації, такі як масштаби або деталізацію, а також задавати специфічні умови для генерації ландшафтів (див. рис. 3.3).

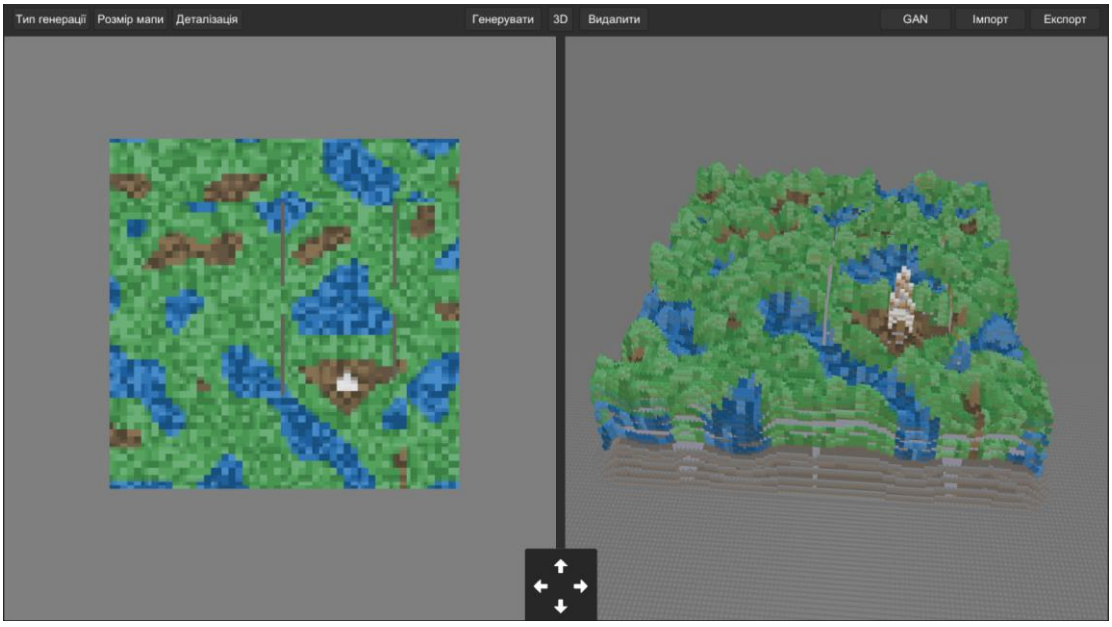

Рисунок 3.3. Інтерфейс програмного продукту

Інтерфейс програми, в якій реалізовано таку генерацію, був розроблений з урахуванням максимального зручності для користувача. Це дозволяє швидко

налаштовувати основні параметри генерації, такі як тип шуму, глибина рельєфу, текстури та інші важливі фактори.

Завдяки інтеграції цих методів і створенню зручного інтерфейсу програма стає потужним інструментом для процедурної генерації контенту. Вона дозволяє не лише створювати реалістичні і адаптивні ландшафти, але й дає можливість значно оптимізувати цей процес, зменшуючи час, необхідний для створення високоякісних карт.

Таким чином, запропонована система генерації світу не тільки створює унікальні та цікаві карти, а й дозволяє легко масштабувати та розширювати ігрове середовище, роблячи кожну сесію неповторною. Це забезпечує більш захопливий ігровий процес, де кожне дослідження території приносить нові виклики та можливості

3.3. Оптимізація та аналіз продуктивності розробленої системи

Тестування, оптимізація та аналіз продуктивності розробленої системи є невід'ємними етапами роботи, що дозволяють оцінити ефективність генератора карт, виявити можливі недоліки та покращити його продуктивність. Головна мета тестування полягала у визначенні, наскільки швидко працює генерація, наскільки стабільно поводиться система при зміні параметрів, чи здатний генератор створювати різноманітні карти та наскільки зручно ним користуватися. Оптимізація ж була спрямована на покращення швидкодії та зменшення споживання ресурсів, щоб генератор міг ефективно працювати навіть у великих проєктах.

Один із перших аспектів, який я перевіряв, – це швидкість роботи генератора. Важливо, щоб користувач отримував карту в мінімальні терміни, навіть при складних параметрах генерації. Я тестував різні алгоритми та аналізував їхню продуктивність. Щоб оптимізувати швидкість генерації, я використав кілька підходів:

1) Кешування проміжних розрахунків, щоб уникнути повторного обчислення однакових значень.

2) Багатопотокову обробку, яка дозволяє розподілити навантаження на процесор і прискорити генерацію.
3) Зменшення використання пам’яті, що особливо важливо при роботі з великими картами.

Окремо я аналізував, як генератор працює у великих масштабах, оскільки важливо було впевнитися, що система здатна створювати великі карти без надмірного навантаження на обчислювальні ресурси. Під час тестування було перевірено, як змінюється продуктивність при збільшенні розміру карти. Результати показали, що навіть при великих розмірах сцена залишається стабільною, а час генерації збільшується поступово, без стрибкоподібного зростання навантаження (див. табл. 3.3).

Таблиця 3.3 Результати ефекивності системи

Розмір карти (тайлів)	**Час генерації (с)**	**Використання пам’яті (МБ)**
256 × 256	0.7	45
512 × 512	1.7	110
1024 × 1024	7.1	300
2048 × 2048	8.9	920

Другим важливим етапом тестування була оцінка стабільності роботи. Однією з проблем процедурної генерації є можливі зависання або збої при зміні параметрів, особливо якщо карта містить велику кількість деталей. У процесі тестування я аналізував, як генератор поводиться при зміні налаштувань, чи не виникають критичні помилки та чи правильно оновлюється карта без необхідності перезавантаження сцени. Випробування показали, що система стабільно працює навіть при збільшенні розміру карти або використанні складних алгоритмів. При цьому вдалося уникнути зависань завдяки ефективному керуванню пам’яттю. Було впроваджено динамічне видалення об’єктів, які знаходяться за межами видимості, що значно зменшило навантаження на графічний процесор. Крім того, усі зміни параметрів застосовуються в реальному часі без необхідності перезапуску генератора, що підвищує зручність роботи з інструментом.

Машинне навчання допомогло підвищити варіативність та логічну зв'язність генерованих карт. Нейронна мережа вивчала особливості генерації карт, аналізуючи велику кількість різноманітних даних про рельєф, біоми, розташування об'єктів тощо. Цей підхід дозволив не лише покращити якість карт, а й забезпечити більш природний розподіл об'єктів у просторі, забезпечуючи більш реалістичну та логічну структуру кожного згенерованого світу.

Оцінка продуктивності включає аналіз стабільності роботи системи. Під час тестування генератор працював без збоїв та помилок навіть при зміні великої кількості параметрів. Оптимізація алгоритмів та грамотне управління пам’яттю дозволяє уникнути зависань та непередбачуваних збоїв. Генератор успішно обробляє різні типи даних та коректно реагує на дії користувача, що свідчить про його високу надійність.

Таким чином, проведене тестування дозволило визначити сильні сторони генератора та вдосконалити його продуктивність. Оптимізація дозволила зменшити час генерації, зробити систему більш стабільною та знизити навантаження на апаратні ресурси. Розроблений інтерфейс значно полегшує роботу з генератором, роблячи його доступним як для досвідчених користувачів, так і для тих, хто вперше працює з процедурною генерацією. Загальний аналіз показав, що генератор ефективно виконує поставлені задачі та може використовуватися у різних сценаріях, від тестових експериментів до інтеграції в масштабні ігрові проєкти.

Висновки до розділу 3

У ході розробки програмного продукту було проведено комплексний аналіз і реалізацію ключових аспектів, необхідних для створення ефективного генератора карт. Вибір рушія відіграв вирішальну роль у цьому процесі. Серед різних варіантів було обрано Unity як оптимальну платформу для розробки, враховуючи його широкі можливості щодо кросплатформності, розвинену екосистему та велику кількість інструментів, що дозволяють ефективно

реалізовувати процедурну генерацію. Використання мови програмування C# сприяло зручному написанню алгоритмів, їхній оптимізації та тестуванню. Проведений порівняльний аналіз із іншими ігровими рушіями, такими як Unreal Engine та Godot [14], підтвердив, що Unity найкраще підходить для реалізації поставлених завдань, оскільки він поєднує потужний функціонал і зручність у роботі.

Окрім вибору технологічної основи, важливим етапом стало створення користувацького інтерфейсу. Для забезпечення максимальної зручності взаємодії користувача з генератором карт було розроблено інтуїтивний та функціональний UI, який дозволяє легко змінювати параметри генерації в реальному часі. Використання інтерактивних елементів, таких як повзунки, кнопки, випадаючі списки та панелі налаштувань, дало можливість швидко адаптувати параметри генерації під потреби конкретного користувача. Такий підхід дозволяє змінювати характеристики карти без необхідності внесення змін у вихідний код або перезавантаження системи, що значно підвищує продуктивність роботи з програмою.

За допомогою технології ML-Agents в Unity було інтегровано систему, яка автоматично вчилася на попередніх результатах генерації та оптимізувала процес створення карт, вивчаючи закономірності та покращуючи розподіл об'єктів у просторі. Машинне навчання сприяло розвитку адаптивності генератора, дозволяючи нейронній мережі оптимізувати параметри генерації, такі як рівень деталізації, висота рельєфу та густота об'єктів. Завдяки навчанню на великих наборах даних система змогла покращити розподіл біомів, водних просторів, а також логіку взаємодії різних елементів карти. Це дозволило згенерувати карти, які відповідають природній логіці і виглядають більш реалістично, що позитивно вплинуло на ігровий процес.

Особливу увагу було приділено оцінці ефективності та продуктивності роботи генератора. Проведені тестування показали високу швидкість роботи алгоритмів навіть при значному ускладненні карти. Оптимізація використання пам’яті та впровадження ефективних методів процедурної генерації дозволили

досягти балансу між продуктивністю та якістю створюваних ландшафтів. Було перевірено стабільність генератора в різних умовах, включаючи створення великих карт із високим рівнем деталізації. Тестування підтвердило, що реалізовані алгоритми можуть швидко і без збоїв обробляти складні параметри генерації, що є важливим критерієм для програм подібного типу.

Таким чином, розроблений програмний продукт повністю відповідає всім основним вимогам, що висувалися до продуктивності, стабільності роботи, якості створюваних карт та зручності використання. Реалізовані функції забезпечують користувачеві можливість швидко та ефективно генерувати ландшафти, змінювати їхні характеристики, експериментувати з параметрами та отримувати якісний результат у найкоротший термін.

ВИСНОВОК

У процесі виконання даної кваліфікаційної роботи було проведено комплексне дослідження алгоритмів процедурної генерації ігрових світів, їхніх особливостей, переваг і недоліків. Визначено основні принципи побудови процедурно згенерованих середовищ, здійснено аналіз існуючих методів генерації та їхньої ефективності для використання у відеоіграх. Окрему увагу було приділено розгляду актуальних проблем у сфері процедурної генерації, таких як недостатня варіативність створюваних світів, проблеми з логічною зв’язністю об’єктів у просторі та продуктивність алгоритмів.

У ході роботи було виявлено, що процедурна генерація є потужним інструментом у сучасній ігровій розробці, дозволяючи створювати великі, складні й унікальні ігрові світи без необхідності ручного опрацювання кожного елемента. Вона забезпечує значну економію ресурсів при розробці та відкриває широкі можливості для створення динамічного ігрового середовища. Однак використання процедурної генерації вимагає ретельного налаштування параметрів, оскільки без достатньої контролюваності процесу результати можуть виявитися неструктурованими або навіть неіграбельними.

Проведений аналіз дозволив сформулювати вимоги до алгоритмів процедурної генерації. Передусім, такі алгоритми повинні бути продуктивними, тобто працювати з мінімальними витратами ресурсів і забезпечувати швидке створення ігрових світів навіть у реальному часі. Важливим критерієм є різноманітність, оскільки однотипні світи швидко набридають гравцеві та знижують рівень занурення в гру. Генерація має забезпечувати високу варіативність результатів, щоб уникати повторюваності. Також алгоритми мають бути керованими, що дає змогу розробникам змінювати параметри генерації відповідно до потреб гри. Це дозволяє створювати ігрові світи, що відповідають концепції гри, а також адаптувати алгоритм під різні жанри та стилістику.

Ще одним важливим фактором є адаптивність алгоритмів, оскільки система повинна легко інтегруватися в сучасні ігрові рушії та працювати в межах різних технічних обмежень. Окрім того, логічна зв’язність світу є необхідною умовою для забезпечення цілісності та реалістичності. Процедурно згенерований світ повинен мати структуроване середовище, в якому розташування біомів, річок, гір, міст, лісів та інших об’єктів підкоряється природній логіці, а не є випадковим набором елементів.

На основі цих вимог було розроблено алгоритм процедурної генерації, який реалізує кілька методів створення ігрового світу. Він враховує особливості рельєфу, розподіл біомів, розміщення об’єктів та структуру населених пунктів. Запропонований алгоритм дозволяє генерувати світи з високою варіативністю, забезпечуючи баланс між випадковістю та керованістю процесу. Крім того, алгоритм використовує механізми контролю якості результату, що дозволяє уникати появи нелогічних або некоректних конфігурацій світу.

Практична реалізація алгоритму продемонструвала його ефективність та відповідність поставленим вимогам. Проведене тестування показало, що створений інструмент дозволяє отримувати деталізовані ігрові світи з можливістю масштабування, що робить його придатним для використання в різних жанрах ігор. Використані методи забезпечують швидке виконання генерації навіть у великих масштабах, що підтверджує доцільність їх застосування.

Особливу увагу було приділено розробці системи параметризації алгоритму. Це дозволило реалізувати механізм, за допомогою якого розробники можуть гнучко змінювати характеристики генерації, такі як щільність об’єктів, особливості рельєфу, рівень деталізації середовища тощо. Такий підхід дає можливість використовувати один і той самий алгоритм у різних ігрових проєктах, адаптуючи його до конкретних вимог.

Ще одним важливим результатом є аналіз продуктивності алгоритму. Проведені експерименти показали, що запропоноване рішення здатне

створювати великі ігрові світи з мінімальними витратами обчислювальних ресурсів, що є важливим чинником у сучасних іграх. Оптимізація процесу генерації дозволила значно знизити навантаження на процесор та оперативну пам’ять, що робить алгоритм придатним навіть для мобільних пристроїв.

Одним із ключових етапів роботи стало застосування технології ML-Agents для інтеграції нейронної мережі, яка дозволила оптимізувати генерацію карт, враховуючи закономірності попередніх результатів і забезпечуючи більш реалістичне розподілення елементів у просторі. Завдяки машинному навчанню було досягнуто значного покращення якості генерованих ландшафтів, зокрема в аспектах рівня деталізації, висоти рельєфу та взаємодії різних біомів.

Виконана робота підтверджує актуальність процедурної генерації для сучасної ігрової індустрії. Використання таких методів відкриває нові можливості для створення масштабних та унікальних ігрових світів, значно спрощує розробку контенту та дозволяє досягти більшої різноманітності в іграх.

Отримані результати можуть бути використані як основа для подальших досліджень у сфері процедурної генерації. Зокрема, перспективними напрямками є вдосконалення алгоритмів для підвищення якості згенерованих світів, інтеграція штучного інтелекту для розумного керування генерацією, а також дослідження гібридних методів, що поєднують ручне створення контенту з автоматизованими процедурами.

Таким чином, виконана робота зробила внесок у розвиток методів процедурної генерації ігрових світів та їх застосування в сучасній ігровій індустрії. Запропоновані рішення можуть знайти практичне застосування у створенні ігор різних жанрів, забезпечуючи більш високу якість контенту при менших витратах ресурсів. Дослідження в цьому напрямку є перспективними, і подальша робота може бути спрямована на покращення алгоритмів, розширення їхніх можливостей та інтеграцію з іншими технологіями, що використовуються в сучасних відеоіграх.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Боровік Н. І. Розробка roguelite-гри з процедурною генерацією оточення (підземель). Київ : Нац. ун-т «Києво-Могилянська академія», 2023. 31 с. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/5d3e1ada-adaf-456c-9d0e-617f29967d50/content (дата звернення: 14.11.2025).
2. Марчук Г. В. , Левківський В. Л., Харченко А. В., Марчук Д. К. Огляд і аналіз алгоритмів процедурної генерації ігрових світів. Технічний журнал. 2022. URL: https://journals.ksauniv.ks.ua/index.php/tech/article/view/644/603 (дата звернення: 14.11.2025).
3. Медвідь Ю. Використання нейронних мереж у процедурній генерації ігрових світів / Збірник матеріалів наукової конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам’янець-Подільського національного університету імені Івана Огієнка у 2024-2025 н.р., 9-10 квітня 2025 року. Кам’янець-Подільський, 2025. С. 49–52. URL: http://elar.kpnu.edu.ua:8081/xmlui/handle/123456789/9195 (дата звернення: 14.11.2025).
4. Медвідь Ю. Сучасні техніки процедурної генерації у створенні ігрових світів / Вісник Кам’янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Кам’янець-Подільський : Кам’янець-Подільський національний університет імені Івана Огієнка, 2024. Вип. 17. С. 93–97. URL: http://elar.kpnu.edu.ua:8081/xmlui/handle/123456789/8685 (дата звернення: 14.11.2025).
5. Смалько О., Іванюк В., Мястковська М., Медвідь Ю. Формування основ ігрового штучного інтелекту / Штучний інтелект у науці та освіті (AISE 2025). Artificial intelligence in science and education: збірник матеріалів міжнародної наукової конференції (Київ, 15 квітня 2025 р.) Київ: УкрІНТЕІ, 2025 URL: https://sites.google.com/view/aise-2025 (дата звернення: 14.11.2025).

6. Arjovsky M., Chintala S., Bottou L. Wasserstein GAN. arXiv. 2017. arXiv:1701.07875. URL: https://arxiv.org/pdf/1701.07875 (дата звернення: 14.11.2025).
7. Arulkumaran K., Deisenroth M. P., Brundage M. та ін. A brief survey of deep reinforcement learning. IEEE Signal Processing Magazine. 2017. arXiv:1708.05866. DOI: 10.1109/MSP.2017.2743240.
8. Chung T. M. Setting up a Python environment with Unity ML-Agents and TensorFlow for macOS. Medium. 2018. URL: https://medium.com/@indiecontessa/setting-up-a-python-environment-with-tensorflow-on-macos-for-training-unity-ml-agents-faf19d71201 (дата звернення: 14.11.20254).
9. Epic Games. The most powerful real-time 3D creation tool: Unreal Engine. URL: https://www.unrealengine.com/ (дата звернення: 14.11.2025).
10. Fachada N., Rodrigues A. R., de Andrade D., Lopes P. Generating 3D terrain with 2D cellular automata. arXiv. 2024. URL: https://arxiv.org/pdf/2406.00443 (дата звернення: 14.11.2025).
11. Farrokhi Maleki M., Zhao R. Procedural content generation in games: a survey with insights on emerging LLM integration. Proceedings of the Twentieth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-24). 2024. Vol.20, №1. URL: https://arxiv.org/abs/2410.15644 (дата звернення: 14.11.2025).
12. Francois-Lavet V., Henderson P., Islam R. An introduction to deep reinforcement learning. Foundations and Trends in Machine Learning. 2018. Vol. 11, № 3-4. P. 219–354. arXiv:1811.12560. DOI: 10.1561/2200000071.
13. Gaea 2.0. QuadSpinner. URL: https://quadspinner.com/gaea (дата звернення: 14.11.2025).
14. Godot Engine. Godot Engine: free and open source 2D and 3D game engine. URL: https://godotengine.org/ (дата звернення: 14.11.2025).
15. Goodfellow I. J. та ін. Generative adversarial nets. arXiv. 2014. URL: https://arxiv.org/abs/1406.2661 (дата звернення: 14.11.2025).

16. Goodfellow I. NIPS 2016 tutorial: generative adversarial networks. arXiv. 2017. arXiv:1701.00160. URL: https://arxiv.org/pdf/1701.00160 (дата звернення: 14.11.2025).
17. Hald A., Hansen J. S., Kristensen J., Burelli P. Procedural content generation of puzzle games using conditional generative adversarial networks. arXiv. 2023. arXiv:2306.15696. URL: https://arxiv.org/pdf/2306.15696 (дата звернення: 14.11.2025).
18. Houdini. SideFX. URL: https://www.sidefx.com/products/houdini/ (дата звернення: 14.11.2025).
19. Joshi A. S. Reinforcement learning-enhanced procedural generation for dynamic narrative-driven AR experiences. arXiv. 2025. arXiv:2501.08552. DOI: 10.5220/0013373200003912.
20. Kingma D. P., Welling M. An introduction to variational autoencoders. Foundations and Trends in Machine Learning. 2019. Vol. 12, № 4. P. 307–392. arXiv:1906.02691. URL: https://arxiv.org/pdf/1906.02691 (дата звернення: 14.11.2025).
21. Kingma D. P., Welling M. Auto-encoding variational Bayes. arXiv. 2013. URL: https://arxiv.org/abs/1312.6114 (дата звернення: 14.11.2025).
22. Lazaridis L., Fragulis G. F. Creating a newer and improved procedural content generation (PCG) algorithm with minimal human intervention for computer gaming development. Computers. 2024. Vol. 13, № 11. P. 304. DOI: 10.3390/computers13110304.
23. Liu J., Snodgrass S., Khalifa A. та ін. Deep learning for procedural content generation. Neural Computing and Applications. 2021. Vol. 33, № 1. P. 19–37. arXiv:2010.04548. DOI: 10.1007/s00521-020-05383-8.
24. Mao X., Yu W., Yamada K. D. та ін. Procedural content generation via generative artificial intelligence. arXiv. 2024. arXiv:2407.09013. DOI: 10.48550/arXiv.2407.09013.
25. Merizzi F. Procedural terrain generation with style transfer. arXiv. 2024. URL: https://arxiv.org/pdf/2403.08782 (дата звернення: 14.11.2025).

26. Minecraft. Welcome to the official site of Minecraft. URL: https://www.minecraft.net/ (дата звернення: 14.11.2025).
27. Nair R., Merino T., Togelius J. God's innovation project – empowering the player with generative AI. arXiv. 2025. arXiv:2504.13874. DOI: 10.48550/arXiv.2504.13874.
28. No Man's Sky. No Man's Sky. URL: https://www.nomanssky.com/ (дата звернення: 14.11.2025).
29. Özkan M. B. Procedural game level design with deep reinforcement learning. arXiv. 2025. arXiv:2510.15120. URL: https://arxiv.org/abs/2510.15120 (дата звернення: 14.11.2025).
30. Perlin K. An image synthesizer. ACM SIGGRAPH Computer Graphics. 1985. Vol. 19, № 3. P. 287–296. DOI: 10.1145/325165.325247.
31. Perlin K. Perlin's "Simplex" Noise. 2001. URL: https://mrl.cs.nyu.edu/~perlin/noise/ (дата звернення: 14.11.2025).
32. Ramstedt R., Smed J. Midpoint displacement in multifractal terrain generation ResearchGate. 2016. URL: https://www.researchgate.net/publication/313367166_Midpoint_Displacement_in_Multifractal_Terrain_Generation (дата звернення: 14.11.2025).
33. Rodriguez Torrado R., Khalifa A., Cerny Green M. Bootstrapping conditional GANs for video game level generation. arXiv. 2019. arXiv:1910.01603. URL: https://arxiv.org/abs/1910.01603 (дата звернення: 14.11.2025).
34. Salimans T., Goodfellow I., Zaremba W. та ін. Improved techniques for training GANs. arXiv. 2016. arXiv:1606.03498. URL: https://arxiv.org/pdf/1606.03498 (дата звернення: 14.11.2025).
35. Skevofylakas M. Deep reinforcement learning using Unity ML-Agents. part I. Medium. 2021. URL: https://medium.com/@m.skevofylakas/deep-reinforcement-learning-using-unity-ml-agents-part-i-62a020e96dd (дата звернення: 14.11.2025).

36. Summerville A., Snodgrass S., Guzdial M. та ін. Procedural content generation via machine learning (PCGML). arXiv. 2017. arXiv:1702.00539. URL: https://arxiv.org/pdf/1702.00539 (дата звернення: 14.11.2025).
37. Summerville A., Snodgrass S., Guzdial M. та ін. Procedural content generation via machine learning (PCGML)]. IEEE Transactions on Games. 2018. Vol. 10, № 3. P. 257–270. arXiv:1702.00539. URL: https://arxiv.org/abs/1702.00539 (дата звернення: 14.11.2025).
38. TensorFlow. TensorFlow: open source machine learning framework. URL: https://www.tensorflow.org/ (дата звернення: 14.11.2025).
39. Unity Technologies. Unity Machine Learning Agents Toolkit (ML-Agents). GitHub. 2025. URL: https://github.com/Unity-Technologies/ml-agents (дата звернення: 14.11.2025).
40. Unity Technologies. Unity real-time development platform: 3D, 2D, VR & AR engine. URL: https://unity.com/ (дата звернення: 14.11.2025).
41. Van der Staaij A., Prins J., Prins V. L. та ін. Believable Minecraft settlements by means of decentralised iterative planning. arXiv. 2023. URL: https://arxiv.org/pdf/2309.10871 (дата звернення: 14.11.2025).
42. Villanueva Aylagas M., Bergdahl J., Gillberg J. та ін. Improving conditional level generation using automated validation in match-3 games. arXiv. 2024. arXiv:2409.06349. DOI: 10.1109/TG.2024.3440214.
43. Vollmer B., Nehlig F., Ibata R. The flaring HI disk of the nearby spiral galaxy NGC 2683. Astronomy & Astrophysics. 2016. Vol. 586. URL: https://arxiv.org/abs/1512.07058 (дата звернення: 14.11.2025).
44. World Machine. URL: https://www.world-machine.com/ (дата звернення: 14.11.2025).
45. Wu Z. Procedural game map generation using multi-leveled cellular automata by machine learning. ResearchGate. 2023. URL: https://www.researchgate.net/publication/357273855_Procedural_Game_Map_Generation_using_Multi-leveled_Cellular_Automata_by_Machine_learning (дата звернення: 14.11.2025).

ДОДАТОК

Програмні інструменти для процедурної генерації

World Machine — один із найпотужніших інструментів для генерації природних ландшафтів, що використовується в розробці ігор та візуалізації середовищ для кіноіндустрії [44]. Основний принцип роботи програми базується на фрактальній генерації, яка дозволяє отримувати деталізовані гори, рівнини, пагорби, річкові системи та інші природні об’єкти (рис. 1).

Програма пропонує вузлову систему побудови, що дає змогу користувачам створювати складні ландшафти шляхом комбінування різних алгоритмів. Однією з ключових особливостей є генерація ерозії, яка робить ландшафти ще більш реалістичними, додаючи ефекти вивітрювання, стікання води та накопичення осадових порід.

Рисунок 1. Інтерфейс World Machine

Окрім візуалізації, World Machine підтримує експорт карт висот у різні формати, що робить її сумісною з ігровими рушіями, такими як Unity та Unreal Engine. Завдяки цьому розробники можуть швидко інтегрувати згенеровані ландшафти у свої проєкти та налаштовувати їх відповідно до потреб гри.

Houdini — це один із найпотужніших інструментів для процедурної генерації, що активно використовується у розробці ігор, анімації та візуальних ефектах. Houdini пропонує гнучкі можливості для створення як природних ландшафтів, так і урбаністичних структур, підземель та міст [18].

Основною особливістю Houdini є вузловий (node-based) підхід до створення контенту, що дозволяє комбінувати різні алгоритми генерації (див. рис. 2). Це робить його ефективним як для шумових алгоритмів (Perlin noise, Simplex noise), так і для методів на основі графів та клітинних автоматів. Завдяки цьому можна створювати не лише природні ландшафти, а й процедурні рівні, архітектурні структури та інші складні об’єкти.

Houdini також підтримує динамічне моделювання фізичних процесів, таких як ерозія, рух води, вітер та осадові процеси, що дозволяє отримати більш реалістичні результати.

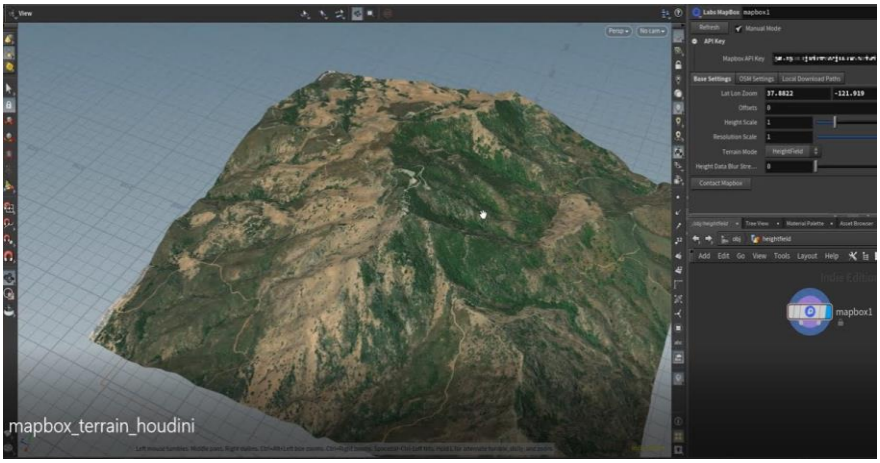

Рисунок 2. Інтерфейс Houdini

Однією з найважливіших переваг Houdini є інтеграція з рушіями Unreal Engine та Unity через Houdini Engine, що дозволяє змінювати параметри генерації у реальному часі без необхідності перезапуску програми. Це робить Houdini незамінним інструментом для розробників, які працюють з великими відкритими світами або складними процедурними рівнями.

Gaea — сучасний генератор ландшафтів, який поєднує методи фрактальної генерації та шумових алгоритмів для створення реалістичних природних світів (див. рис. 3). Вона орієнтована на професійне використання в ігровій та кіноіндустрії, дозволяючи створювати складні географічні структури з високим рівнем деталізації. Програма використовує нодову систему редагування, що дозволяє користувачам будувати складні карти висот

шляхом комбінації різних алгоритмів. Gaea підтримує динамічну симуляцію ерозії, що дозволяє отримати більш природні результати [13].

Однією з ключових особливостей Gaea є можливість експорту високоякісних карт висот у формати, сумісні з рушіями Unreal Engine, Unity, а також іншими професійними інструментами, такими як Houdini та Blender.

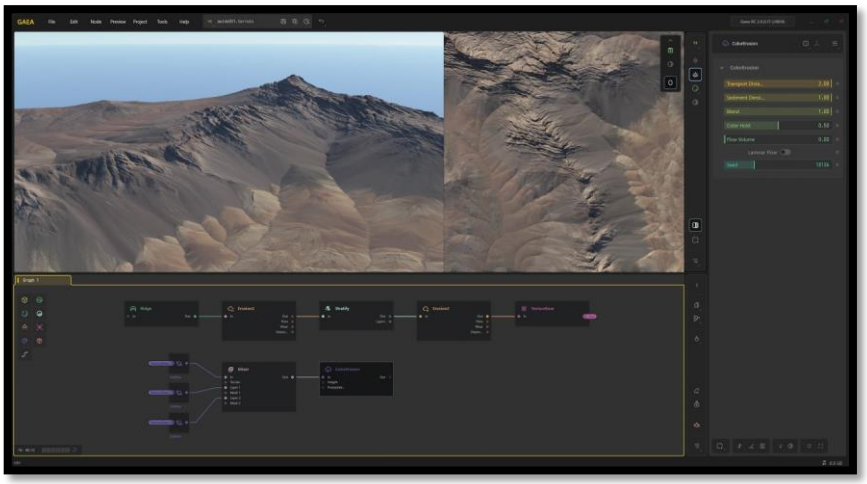

Рисунок 3. Інтерфейс Gaea

Програма ідеально підходить для створення великих відкритих світів, але, на відміну від MapGenie, вона не призначена для генерації внутрішніх структур, таких як підземелля або міста.

Досліджені інструменти використовують різні алгоритми, зокрема фрактальну генерацію, шумові функції та клітинні автомати, кожен із яких має свої особливості, переваги та обмеження. Для узагальнення отриманих результатів наведемо порівняльну таблицю, яка містить ключові характеристики розглянутих програм (див. табл.).

Таблиця. Порівняльний аналіз програм для генерації

Програма	**Методи генерації**	**Переваги**	**Недоліки**	**Сфера застосування**
World Machine	Фрактальна генерація	Висока деталізація, підтримка ерозії, експорт у рушії	Вимагає багато ресурсів, не підходить для печер	Генерація відкритих світів, природних ландшафтів

Програма	Методи генерації	Переваги	Недоліки	Сфера застосування
Houdini	Шумові алгоритми, клітинні автомати, процедурні графи, фізичне моделювання	Гнучкість, підтримка складних структур, інтеграція з ігровими рушіями	Високий поріг входу, потребує потужного обладнання	Генерація як природних, так і міських середовищ, рівнів, процедурних карт
Gaea	Фрактальна генерація, шумові алгоритми	Висока деталізація, реалістичні ландшафти	Вимагає потужного заліза	Створення природних світів, рельєфів

Як видно з таблиці, кожен із розглянутих інструментів орієнтований на певні завдання та має свою специфіку. Наприклад, World Machine більше підходить для створення відкритих природних ландшафтів, тоді як MapGenie є більш універсальним рішенням для генерації рівнів у різних жанрах. Gaea, у свою чергу, забезпечує високу деталізацію, що робить його ефективним для реалістичних рельєфів.

Вибір інструмента залежить від конкретних вимог ігрового проєкту. Використання процедурної генерації дає змогу значно спростити процес розробки, зменшити витрати ресурсів на створення контенту та підвищити різноманітність ігрового світу. У подальших дослідженнях можна розглянути можливості комбінування різних методів для отримання ще більш ефективних рішень.