МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ, ОБЛІКУ ТА ФІНАНСІВ

Кафедра комп'ютерних технологій і моделювання систем

Кваліфікаційна робота
на правах рукопису

Ємець Даніель Вадимович

УДК 004.92:004.946

КВАЛІФІКАЦІЙНА РОБОТА

Дослідження ефективності моделювання процесу генерації ігрових світів

(тема роботи)

122 Комп’ютерні науки

(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня магістр

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____

(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Тимонін Юрій Олександрович,
Кандидат технічних наук, доцент кафедри КТіМС

Житомир – 2025

Висновок кафедри комп'ютерних технологій і моделювання систем:
за результатами попереднього захисту: _____

Протокол засідання кафедри комп'ютерних технологій і моделювання систем
№ ____ від «_____» _____ 20__ р.

Завідувач кафедри комп'ютерних технологій і моделювання систем
к.п.н., доцент _____ М. О. Ковальчук
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)
«_____» _____ 20___ р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _____ захистив (ла)
(прізвище ,ім'я, по батькові)
кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____
за шкалою ECTS _____
за національною шкалою _____

Секретар ЕК
лаборант кафедри _____ В. В. Корольчук
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

АНОТАЦІЯ

Ємець Д. В. *Дослідження ефективності моделювання процесу генерації ігрових світів* – Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю 122 – Комп'ютерні науки. – Поліський національний університет, Житомир, 2025.

Обсяг кваліфікаційної роботи: 38 сторінок (17 – рисунків, 1 – таблиця, 2 – додатки, 50 – джерел).

Ключові слова: процедурна генерація, Perlin fBm, ridge noise, Unreal Engine , кліматичне моделювання, ерозія.

Кваліфікаційна робота присвячена дослідженню методів процедурної генерації тривимірних ігрових ландшафтів із використанням комбінованих підходів що поєднують стохастичні та детерміністичні компоненти. У роботі проведено аналіз традиційних методів генерації рельєфу, обґрунтовано необхідність створення гібридних архітектур для досягнення природності та геоморфологічної правдоподібності

Програмна реалізація моделі здійснена у середовищі Unreal Engine 4 із використанням мови програмування C++. Проведена серія комп'ютерних експериментів для оцінки продуктивності системи, виявлено критичну диспропорцію обчислювальних витрат де побудова тривимірного мешу займає більшу частину часу при незначних витратах на власне генерацію рельєфу..

Практичне значення отриманих результатів полягає використані для автоматизації створення реалістичних відкритих світів, в навчальному процесі для підготовки студентів спеціальностей пов'язаних із комп'ютерною графікою та геймдизайном, а також у прикладних дослідженнях спрямованих на розвиток методів процедурної генерації для віртуальних середовищ, освітніх симуляторів геології та інструментів архітектурної візуалізації.

SUMMARY

Yemets D. V. *Research on the Efficiency of Game World Generation Process Modeling.* – Qualification work as a manuscript.

Master's degree qualification work in specialty 122 – Computer Science. – Polissia National University, Zhytomyr, 2025.

The volume of the work: 38 pages (17 – figures, 1 – tables, 2 – appendices, 50 – sources).

Key words: procedural generation, Perlin fBm, ridge noise, Unreal Engine, climate modeling, erosion.

The qualification work is devoted to the study of methods for procedural generation of three-dimensional game landscapes using combined approaches that integrate stochastic and deterministic components. The work analyzes traditional terrain generation methods and substantiates the necessity of creating hybrid architectures to achieve naturalness and geomorphological plausibility.

The software implementation of the model was carried out in the Unreal Engine 4 environment using the C++ programming language. A series of computer experiments were conducted to evaluate system performance, revealing a critical disproportion in computational costs where three-dimensional mesh construction occupies the majority of processing time with negligible costs for actual terrain generation.

The practical significance of the results obtained lies in their use for automating the creation of realistic open worlds, in the educational process for training students in specialties related to computer graphics and game design, as well as in applied research aimed at developing procedural generation methods for virtual environments, educational simulators of geology, and architectural visualization tools.

ЗМІСТ

ВСТУП

У сучасній індустрії комп'ютерних ігор однією з найбільш актуальних проблем є створення масштабних, реалістичних і варіативних ігрових світів. Традиційний підхід через ручне моделювання вимагає значних часових та фінансових ресурсів. Ключовим напрямом вирішення є процедурна генерація контенту – автоматизований процес побудови ігрового середовища за допомогою математичних і стохастичних моделей, що забезпечують поєднання випадковості та логічної узгодженості.

Розвиток обчислювальних технологій відкриває нові можливості для побудови систем генерації, здатних враховувати геоморфологічні процеси та кліматичні зони. Сучасні методи базуються на фрактальному шумі (Perlin, Simplex) та алгоритмах рекурсивного поділу (Diamond-Square). Водночас залишається недостатньо дослідженим аспект оцінювання ефективності таких моделей – критерії продуктивності, масштабованості та природності згенерованого рельєфу. Існує потреба у комбінованих підходах що поєднують переваги базових алгоритмів з системами постобробки.

Актуальність теми зумовлена необхідністю вдосконалення математичних моделей процедурної генерації з урахуванням фізично обґрунтованих процесів ерозії та кліматичного моделювання. Дослідження має міждисциплінарне значення, поєднуючи елементи комп'ютерної графіки, обчислювальної математики та геймдизайну. Розробка ефективних алгоритмів відкриває можливості для застосування у розважальній індустрії, освітніх симуляторах та системах віртуальної реальності.

Об'єкт дослідження процеси моделювання та процедурної генерації ландшафтів у комп'ютерних системах реального часу.

Предмет дослідження алгоритмічні та математичні моделі що впливають на ефективність, якість візуалізації та продуктивність генерації з урахуванням систем ерозії та кліматичного моделювання.

Мета роботи підвищення ефективності процедурної генерації ландшафтів шляхом розроблення комбінованої математичної моделі, яка

поєднує фрактальні шумові функції, системи ерозії та кліматичне моделювання для досягнення балансу між продуктивністю та якістю

Для досягнення поставленої мети необхідно вирішити наступні *завдання*:

1. Провести порівняльний аналіз сучасних методів процедурної генерації ландшафтів.
2. Розробити математичну модель що поєднує фрактальний шум, ridge noise та системи термальної і гідравлічної ерозії реалізувати модель у Unreal Engine 4.
3. Провести експериментальне дослідження продуктивності та масштабованості системи.
4. Запропонувати критерії оцінки ефективності включаючи статистичний аналіз розподілу висот та градієнтів рельєфу.
5. Оцінити перспективи застосування моделі для створення відкритих світів у комп'ютерних іграх та освітніх симуляторах.

У роботі використано *методи* математичного моделювання фрактальних структур, статистичний аналіз розподілу висот та градієнтів, алгоритми фізично обґрунтованої ерозії, кліматичне моделювання, експериментальне оцінювання з використанням Unreal Engine 4, C++, Python, pandas та matplotlib.

Наукова новизна:

1. Розробці гібридної архітектури системи процедурної генерації що інтегрує базовий алгоритм Perlin fBm з додатковими шарами ridge noise, WorldShape масками для контролю макроформи рельєфу та повним конвеєром фізично обґрунтованої постобробки безпосередньо у Unreal Engine 5 з використанням компонентного підходу на C++.
2. Запропонованій методиці комбінування стохастичних та детерміністичних компонентів генерації, яка забезпечує асиметричний природний розподіл висот через послідовне застосування термальної ерозії, гідравлічної ерозії та систем генерації

річкових мереж з автоматичним балансуванням між плавністю рельєфу та геоморфологічною варіативністю.

3. Виявленні та проаналізованій проблемі диспропорції обчислювальних витрат у системах процедурної генерації з геометричною візуалізацією, де побудова тривимірного мешу займає загального часу при незначних витратах на власне алгоритми генерації, що вказує на необхідність оптимізації геометричних операцій для практичного застосування у реальному часі.

Теоретичне значення полягає у систематизації знань про математичне моделювання процедурної генерації ландшафтів з використанням комбінованих підходів що поєднують стохастичні фрактальні методи з детерміністичними фізичними симуляціями ерозійних процесів. Результати дослідження можуть бути використані для подальших наукових розробок у галузі комп'ютерної графіки та обчислювальної геоморфології, зокрема для переходу до GPU-based паралелізації обчислень та застосування адаптивних систем Level of Detail.

Практична значущість полягає у можливості використання моделі для автоматизації побудови відкритих світів у комп'ютерних іграх, освітніх симуляторів геології та інструментів ландшафтного проектування. Система демонструє високу продуктивність при високої якості візуалізації, що робить її придатною для комерційних ігрових проєктів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ТА ТЕХНОЛОГІЧНІ ЗАСАДИ МОДЕЛЮВАННЯ ПРОЦЕСІВ ГЕНЕРАЦІЇ ІГРОВИХ СВІТІВ

1.1. Аналіз предметної області та роль процедурної генерації у сучасних ігрових рушіях

Процедурна генерація – ключовий аспект у розробці сучасних ігор та віртуальних всесвітів. Цей підхід дозволяє створювати різноманітні й безкінечні ігрові оточення, які постійно змінюються, надаючи гравцям унікальний досвід під час кожного запуску гри. Важливість процедурної генерації криється в її здатності генерувати величезні, захопливі світи, що відкриті для досліджень, при цьому утримуючи інтерес гравця [1]. Крім того процедурна генерація оптимізує використання ресурсів оскільки замість того щоб зберігати велику кількість створених даних, зберігаються лише алгоритми та вихідні параметри генерації. Це надає змогу зменшити обсяг використаної пам’яті та розмір гри. Також даний метод автоматизує процес створення ігрового контенту, що зменшує затрати ресурсів.

Є багато ігор які використовують процедурну генерацію світу та контенту одною із таких є Minecraft. Ця гра представляє собою гру жанру “Пісочниця” та “Виживання”. Щоразу, коли гра повинна згенерувати новий світ, вона звертається до алгоритму, відомого як Шум Перлина. Цей алгоритм видає псевдовипадкове значення, яке потім використовується для визначення характеристик та особливостей світу. Однак алгоритм завжди виводить одне й те саме значення щоразу для постійної початкової точки (зерна) [2], [3]. Таким чином, те саме зерно генерує ту саму місцевість щоразу. В основі рельєфу лежить алгоритм шуму Перліна або шуму OpenSimplex, які генерують «грубу» висотну карту. Цей шум перетворюється на висоти блоків землі, формуючи пагорби, рівнини, гори.

Також однією із ігор яка використовує процедурну генерацію світів є Terraria. У Terraria процес генерації відбувається в 2D-просторі та виконується відразу при створенні світу, що робить її статичною – весь світ створюється наперед і далі може змінюватися лише діями гравця [4].

Натомість у Minecraft реалізовано динамічну генерацію: нові частини світу (чанки) створюються тоді, коли гравець наближається до недосліджених територій, що дозволяє економити ресурси та забезпечує практично нескінченний простір.

У грі No Man’s Sky процедурна генерація втілена на неймовірно глибокому рівні: гра формує цілі планети, їхню географію, атмосферу, флору, фауну та навіть звуки – усе це створюється на основі математичних функцій, зокрема шумових алгоритмів, фрактальної геометрії та варіацій параметричних моделей. Кожна із планет має свій унікальний рельєф, склад атмосфери, клімат, типи рослин і тварин, які не просто випадкові, а логічно взаємопов’язані, формуючи цілісну екосистему.

Під час моделювання процедурного світу важливо враховувати ключові аспекти, що визначають його структуру та впливають на ігровий процес. До них слід віднести: рельєф (гірські масиви, рівнини, низовини), який є фундаментом ландшафту; водойми (озера, річки, океани), котрі впливають на біоми та екосистеми; печери – розгалужені підземні структури, що надають простір для дослідження; рослинність, що забезпечує візуальне розмаїття та ресурси; а також корисні копалини, які потрібно розміщувати з урахуванням глибини, частоти зустрічальності та ігрового балансу. Створення кожного з цих компонентів зазвичай здійснюється за допомогою алгоритмів шуму, клітинних автоматів або параметричних правил.

До моделі процедурної генерації висуваються три ключові вимоги: варіативність (щоб кожний новий світ був оригінальним та не повторювався), логічність (взаємозв'язок між кліматом, рельєфом, біомами та ресурсами), а також ефективність реалізації (оперативне оброблення, можливість масштабування та раціональне використання пам'яті). Необхідно, щоб модель не тільки формувала привабливий світ, але й забезпечувала глибину досліджень, геймплейний інтерес і технічну стабільність в режимі реального часу.

1.2. Класифікація методів генерації: шумові, стохастичні, фрактальні, гібридні підходи та машине навчання

Для втілення процедурної генерації світів використовуються різноманітні математичні техніки, кожна з яких відповідає за конкретний аспект моделювання – від створення рельєфу до розміщення споруд та логіки взаємодії. Ці техніки забезпечують необхідний ступінь варіативності, реалістичності та ефективності, дозволяючи генерувати складні та правдоподібні віртуальні простори. Далі наведені ключові підходи, які найчастіше використовуються в процедурному моделюванні.

Шум Перліна, створений Кеном Перліном в 1983 році, належить до градієнтного шуму. Його часто використовують для імітації текстур і реалістичних природних ефектів. Perlin Noise здатний створювати візуально плавні та реалістичні зображення, що зумовило його широке застосування у процедурній генерації, зокрема, при створенні рельєфів та текстур з органічним виглядом [5], [6]. Математична сутність шуму полягає у формуванні градієнтів на основі сітки з подальшою інтерполяцією, що вирізняє його з-поміж "білого шуму", де кожна точка має повністю випадкове значення та відсутня безперервність. Візуальний приклад шуму Перліна зображено на рисунку (див. рис. А.1, додаток А).

Стохастичні алгоритми, зокрема ланцюги Маркова, являють собою ефективний інструмент процедурної генерації в гейм-дизайні, особливо коли необхідно створювати послідовності з певним рівнем випадковості [7]. Фундаментом цього підходу є теорія ймовірностей, де система опиняється у конкретному стані, а перехід до наступного стану обумовлюється певним шансом, що не залежить від минулих станів – тільки від теперішнього. Це й становить суть марковського процесу.

Фрактальні підходи знаходять широке застосування в процедурній генерації для створення реалістичних природних краєвидів. Фрактали – це геометричні фігури, що володіють властивістю самоподібності, тобто їхня структура повторюється на різних масштабах. У природі фрактальні

структури зустрічаються у горах, деревах, узбережжях, хмарах і річкових системах. Ці природні форми не піддаються традиційній евклідовій геометрії, але їх можна описати за допомогою фрактальної геометрії, що дозволяє моделювати складні природні явища з високою точністю [8], [9]. Приклад згенерованої поверхні за допомогою фрактального підходу зображено на рисунку (див. рис. А.2, додаток А).

Метод розділення середини (Midpoint Displacement) є одним з найпоширеніших фрактальних алгоритмів для генерації ландшафтів. Цей метод розпочинається з прямої лінії між двома точками, після чого додається нова точка посередині з випадковим вертикальним зсувом [10]. Цей процес рекурсивно повторюється для кожного нового сегменту, формуючи "зубчасту" лінію, що нагадує контур гірського хребта. У двовимірному просторі цей метод дозволяє створити складні ландшафти з деталізованими елементами, як-от долини та вершини.

Графові конструкції є наріжними в процесі генерації ігрових світів у програмуванні. Граф містить вершини (вузли), що позначають об'єкти чи місцевості, а також ребра, які їх з'єднують, указуючи на можливі з'єднання або шляхи між ними. У ігровому процесі графи використовують для створення систем підземель, зв'язків між локаціями, планування маршрутів, а також для побудови дерев подій чи карт світу [11], [12]. Це дає можливість розробникам створювати комплексні, взаємопов'язані структури, що формують для гравців глибокий та інтерактивний ігровий досвід.

Для раціонального використання графів у ігровому середовищі, використовуються різноманітні алгоритми обходу. Пошук у глибину (DFS) і пошук у ширину (BFS) – це базові методи, призначені для дослідження всіх потенційних шляхів у графі [13]. Алгоритм Дейкстри застосовується для знаходження найкоротшого маршруту між двома вершинами в графі з невід'ємними значеннями ребер, що добре для навігації персонажів на складних місцевостях.

Також одним із методів генерації є дифузійні моделі. Дані моделі здатні створювати високодеталізовані зображення, текстури, 3D-структури та навіть анімації, що зробило їх важливим інструментом сучасної процедурної генерації. Математична сутність дифузійних моделей складається з двох процесів: прямій дифузії, під час якої зображення поступово спотворюється із-за додавання шуму, та зворотній дифузії, де модель навчається поступово «прибирати» шум, відновлюючи структуру. Із-за цього модель формує глибоке розуміння розподілу даних, що дозволяє їй генерувати нові зразки, які статистично схожі на навчальний набір даних. Відмінність від класичних шумових методів, це то що, дифузійні моделі дозволяють контролювати як глобальні, так і дрібні деталі зображення – від ландшафтів до складних різноманітних органічних поверхонь.

Генеративно-змагальні мережі (GAN) стали одним із важливих підходів до створення синтетичних даних і відіграли важливу роль у розвитку сучасної генеративної графіки. Архітектура GAN складається з двох нейромереж – генератора та дискримінатора, які навчаються у змагальному процесі: перший створює зображення, що мають виглядати реальними, а другий оцінює їхню справжність [14].

Математично робота GAN ґрунтується на мінімаксній грі, де генератор мінімізує шанс бути викритим, а дискримінатор – максимізує точність розпізнавання. Така взаємодія дозволяє отримувати високореалістичні зображення, текстури чи 3D-форми. На відміну від дифузійних моделей, GAN забезпечують швидшу генерацію, оскільки результат формується за один прохід через мережу У процедурній генерації GAN застосовують для формування текстур, елементів рослинності, матеріалів, моделей будівель або NPC. Завдяки здатності відтворювати природні деталі та цілісні структури, цей метод став ефективним інструментом створення фотореалістичного контенту із контрольованою варіативністю.

Ці алгоритми активно задіюються в іграх для реалізації навігації, штучного інтелекту та генерації ігрових рівнів З метою загального порівняння розглянутих методів було сформовано таблицю, що містить їхні ключові характеристики (див. таблицю Б.1 додаток Б).

1.3. Інструменти реалізації: Python, Unreal Engine

Для реалізації системи процедурної генерації ландшафтів обрано комбінацію Unreal Engine 4.27 та Python. Вибір цих інструментів обумовлений необхідністю поєднання високопродуктивної генерації у реальному часі з можливістю детального аналізу результатів. Unreal Engine є основною платформою для реалізації системи процедурної генерації ландшафтів. Це потужний ігровий рушій, який використовується для створення AAA-ігор та інтерактивних симуляцій [15], [16].

Ключові можливості для проекту є Procedural Mesh Component для динамічного створення геометрії з масивів вершин; C++ програмування для високої швидкості обчислень з підтримкою паралельних обчислень (прискорення у 2-2.5 рази); система матеріалів для автоматичного створення текстур залежно від висоти, нахилу та клімату; Blueprint та UMG для інтерактивного інтерфейсу налаштування параметрів; рендеринг у реальному часі з освітленням та атмосферними ефектами [17], [18].

В Unreal Engine реалізовано: генерацію heightmap на основі багатооктавного шуму Перліна, симуляцію річкових систем через водний стік, застосування ерозії, визначення біомів на основі висоти та клімату, перетворення даних у тривимірну геометрію.

Python виступає аналітичним інструментом для верифікації та оптимізації алгоритмів генерації, забезпечуючи незалежну перевірку результатів, отриманих у середовищі ігрового рушія. Його використання дозволяє виконувати детальний статистичний аналіз згенерованих даних, виявляти відхилення від заданих параметрів, оцінювати стабільність та

відтворюваність результатів, а також порівнювати різні конфігурації генератора за об’єктивними метриками [19], [20].

NumPy – забезпечує ефективну роботу з багатовимірними масивами даних. Використовується для завантаження експортованих heightmap, обчислення статистики (мінімум, максимум, середнє, стандартне відхилення) та матричних операцій для аналізу градієнтів поверхні [21].

Matplotlib – створює різноманітні візуалізації: двовимірні теплові карти висот, тривимірні об'ємні моделі terrain, гістограми розподілу висот, топографічні карти з контурними лініями. Дозволяє експортувати графіки у високоякісні формати для презентацій та звітів [22].

SciPy – надає чисельні методи для обробки даних: фільтри для виявлення піків та долин на heightmap, морфологічні операції для аналізу геологічних структур, інтерполяційні методи для згладжування та трансформації даних [23]. Seaborn – це бібліотека для візуалізації даних у Python, яка значно покращує естетику графіків та спрощує створення професійно виглядаючих статистичних візуалізацій. Вона побудована поверх Matplotlib і надає високорівневий інтерфейс для побудови складних графіків з мінімальною кількістю коду.

В Python реалізовано: завантаження експортованих даних з Unreal Engine (формати JSON, CSV, NumPy), створення візуалізацій для аналізу топографії, обчислення статистики розподілу висот та нахилів, виявлення геологічних особливостей (піки, долини, хребти), валідацію коректності річкових систем, порівняння різних конфігурацій генератора, генерацію автоматичних звітів з графіками.

Такий підхід поєднує переваги обох технологій: Unreal Engine забезпечує швидку генерацію з високоякісною візуалізацією, а Python – детальний аналіз та верифікацію результатів. Розділення відповідальності між інструментами дозволяє незалежно вдосконалювати алгоритми генерації та методи аналізу, що прискорює розробку та покращує якість кінцевого продукту.

Висновки до першого розділу

У першому розділі були розглянуті теоретичні та технологічні засади процедурної генерації ігрових світів. Аналіз предметної області показав, що процедурна генерація є ключовим аспектом сучасної розробки ігор, оскільки дозволяє створювати різноманітні ігрові оточення при оптимальному використанні ресурсів.

Огляд існуючих реалізацій у відомих іграх (Minecraft, Terraria, No Man's Sky) дав змогу визначити ключові вимоги до системи: варіативність, логічність та ефективність реалізації. Класифікація методів процедурної генерації виявила широкий спектр підходів – від класичних шумових функцій до сучасних технологій машинного навчання. Шумові методи були визначені як оптимальний базовий інструмент для генерації рельєфу завдяки їхній детермінованості та плавності переходів.

Було обґрунтовано застосування Unreal Engine 4.27 як основної платформи реалізації та Python як аналітичного інструменту. Unreal Engine забезпечує високопродуктивну генерацію у реальному часі через C++ програмування з підтримкою паралельних обчислень, а Python надає інструменти детального аналізу через NumPy, Matplotlib та SciPy. Організація робочого процесу як ітераційного циклу забезпечує ефективну верифікацію та оптимізацію алгоритмів.

Таким чином, у розділі було сформовано теоретичне підґрунтя для реалізації системи процедурної генерації ландшафтів і визначено технологічні засоби для її втілення. Результати аналізу підтверджують доцільність використання багатооктавного шуму Перліна як базового методу генерації у поєднанні з алгоритмами симуляції ерозії.

РОЗДІЛ 2. МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСУ ГЕНЕРАЦІЇ ІГРОВИХ СВІТІВ

2.1. Постановка задачі та формалізація параметрів ігрового світу

Основною задачею дослідження є розроблення системи процедурної генерації тривімірних ігрових ландшафтів, яка забезпечує створення реалістичних та різноманітних віртуальних світів з мінімальною участю розробника. Система має генерувати ландшафти у реальному часі на основі математичних моделей, враховуючи фізичні та кліматичні закономірності природних геологічних процесів.

Центральним елементом моделі є висотна карта H – двовимірний масив, що представляє нормалізовану висоту кожної точки поверхні. Значення H[x,y] належить інтервалу від 0 до 1:

H = 1.0 – максимальна висота рельєфу (гірські вершини);

H ≈ 0.5 – середні висоти (пагорби, плато);

H → 0 – мінімальна висота (низини, дно водойм).

H визначає базову геометрію ландшафту та є основою для всіх подальших обчислень біомів, температури та розміщення ресурсів. У моделі значення висоти генерується динамічно на основі шумових функцій та залежить від вхідних параметрів генератора. Другим ключовим параметром є Seed – початкове значення генератора псевдовипадкових чисел. Воно визначає, який саме ландшафт буде створено при заданих інших параметрах [24]. У системі Seed є цілим числом у діапазоні від 0 до 2^{31}-1. Чим більше відрізняються значення Seed, тим більше відрізняються згенеровані світи, при цьому один і той самий Seed завжди створює ідентичний ландшафт за умови незмінних інших параметрів. Це забезпечує детермінованість генерації та можливість відтворення результатів.

Параметр NoiseScale визначає масштаб геологічних формацій на ландшафті. У системі HeightNoiseScale встановлено 0.0008 для створення континентальних форм характерним розміром кілька кілометрів на картах 512×512 вершин. Параметр може змінюватися для окремих шарів генерації.

Система багатооктавного шуму Перліна використовує параметри: HeightOctaves = 4 (від 1 до 8), Lacunarity = 2.0 (подвоєння частоти кожної октави), Persistence = 0.5 (зменшення амплітуди вдвічі). Кожна октава додає дрібніші деталі до базового шуму.

WorldShape визначає географічну конфігурацію: Default (стандартна генерація), Landmass (континент з вертикальним зміщенням), Island (острів з радіальною маскою від центру), Archipelago (група островів з кількома центрами). MountainRange – гірський ланцюг. Маска формує підвищення вздовж певної осі (горизонтальної або діагональної).

GlobalTemperature ∈ [0, 1] – базовий рівень температури для всього світу: 0.0 відповідає дуже холодному клімату (переважають тундра, снігові зони), 0.5 – помірному клімату, 1.0 – дуже жаркому клімату (пустелі, савани). За замовчуванням GlobalTemperature = 0.5.

LatitudeTemperatureInfluence– коефіцієнт впливу широти (координати Y) на температуру. Моделює географічний ефект, де полюси холодніші за екватор. За замовчуванням 0.6.

HeightTemperatureInfluence – коефіцієнт впливу висоти на температуру. Моделює зниження температури в горах. За замовчуванням 0.6.

Фінальна температура в точці обчислюється як сума глобальної температури, впливу широти, впливу висоти та невеликої шумової варіації для природності.

Також SmoothHeightPasses ∈ {0, 1, ..., 20} – кількість ітерацій згладжування висотної карти. Кожна ітерація усереднює висоту кожної вершини з її чотирма сусідами. За замовчуванням 4 ітерації. Більше ітерацій створює плавніший рельєф.

ApplyThermalErosion – прапорець увімкнення симуляції термальної ерозії (обсипання крутих схилів через гравітацію). Алгоритм переміщує "матеріал" з високих точок до низьких.

ApplyHydraulicErosion – прапорець увімкнення симуляції гідравлічної ерозії (вимивання ґрунту водними потоками). Алгоритм знаходить для кожної точки найнижчого сусіда та переміщує частину висоти у його напрямку.

Повна система генерації описується функцією:

$$F(P, Seed) \rightarrow (H, M, T, B)$$

де P – множина всіх параметрів генератора, Seed – початкове значення генератора псевдовипадкових чисел, H – висотна карта, M – карта вологості, T – карта температури, B – функція визначення біому.

Основні властивості функції F є детермінованість – при однакових вхідних параметрах результат завжди ідентичний. Варіативність – різні Seed створюють різні ландшафти. Обмеженість – всі карти нормалізовані до діапазону [0, 1]. Плавність – градієнт висоти обмежений для уникнення нереалістичних стрибків.

Для оцінки якості згенерованого ландшафту використовуються наступні метрики:

- Варіативність висот – стандартне відхилення висот має бути у діапазоні [0.15, 0.35], що забезпечує збалансоване поєднання рівнин та гір.
- Плавність переходів – середній градієнт має знаходитись у діапазоні [0.1, 0.3] для природних переходів між типами місцевості.
- Зв'язність води – усі водойми мають формувати зв'язні компоненти без ізольованих пікселів.
- Коректність річок – кожна річка має текти строго вниз за градієнтом без петель.

Таким чином, формалізація параметрів дозволяє повністю описати простір можливих конфігурацій системи генерації та встановити чіткі математичні зв'язки між вхідними даними та результатом.

2.2. Побудова математичних залежностей для моделювання рельєфу, біомів.

Моделювання рельєфу базується на комбінації шумових функцій, що забезпечують природний вигляд ландшафту через поєднання великомасштабних континентальних форм та дрібних деталей рельєфу. Основою генерації висотної карти є багатооктавний шум Перліна (fractional Brownian motion, fBm), що визначається як зважена сума кількох октав базового шуму:

$$fBm(x,y) = \sum_{i=0}^{n-1} A_i \times Perlin(F_i \times x + O_{xi}, F_i \times y + O_{\gamma i})$$

де:

n – кількість октав (HeightOctaves, зазвичай 4);

A_i – амплітуда i-ї октави, що зменшується з коефіцієнтом Persistence = 0.5;

F_i – частота i-ї октави, що збільшується з коефіцієнтом Lacunarity = 2.0;

• O_{xi}, $O_{\gamma i}$ – випадкові зміщення для i-ї октави.

Кожна наступна октава додає дрібніші деталі з меншою амплітудою, що створює природний фрактальний вигляд рельєфу. Результат нормалізується до діапазону [0, 1] діленням на суму амплітуд та перетворенням з [-1, 1].

Для створення гірських хребтів з гострими піками використовується ridged noise через інверсію абсолютного значення шуму:

$$ridge(p) = 1 - |Perlin(p)|$$

$$RidgeFBm(p) = (\sum_{i=0}^{n-1} ridge(F_i \times p + O_i)^2 \times A_i) / \sum_{i=0}^{n-1} A_i$$

де:

ridge – інвертована абсолютна функція шуму, що створює гострі піки замість плавних переходів;

квадратування ridge підсилює контраст між низинами та вершинами.

Цей метод формує характерні гірські хребти з різкими перепадами висоти, що відрізняються від плавних континентальних форм базового fBm. Авторський метод поєднує два шари для досягнення реалістичного рельєфу:

$$H(x,y) = \alpha \times H\_macro(x,y) + (1 - \alpha) \times H\_ridge(x,y)$$

де:

H_macro – великомасштабний шар континентальних форм (плавні переходи);

H_ridge – дрібномасштабний шар гірських деталей (гострі хребти);

α = MacroShapeStrength ∈ [0, 1] – параметр балансу (зазвичай 0.6). Macro-шар використовує базовий масштаб шуму, а ridge-шар має масштаб у 2.5 рази більший (RidgeNoiseScaleMul = 2.5), що створює дрібніші гірські структури на фоні великих континентальних форм. Для створення різних географічних конфігурацій до висотної карти застосовується функція маски:

$$H\_final(x,y) = H(x,y) \times mask(u,v)$$

де u, v ∈ [0, 1] – нормалізовані координати точки.

Для типу Island (острів) маска створює радіальний спад від центру:

$$mask\_island(u,v) = (1 - d\_norm^2)\hat{}\beta$$

де d_norm – нормалізована відстань від центру (0.5, 0.5), β ≈ 2.0 – параметр крутості спаду. Для типу Archipelago генерується кілька центрів островів, фінальна маска є максимумом локальних масок. Для типу MountainRange маска формує підвищення вздовж певної осі через експоненційний спад. Вологість моделюється як комбінація шумової функції та залежності від висоти:

$$M(x,y) = w_1 \times M\_noise(x,y) + w_2 \times (1 - H(x,y))$$

де:

• M_noise – нормалізований багатооктавний шум з масштабом MoistureNoiseScale;

• w_1, w_2 – ваги впливу (зазвичай $w_1 = 0.4$, $w_2 = 0.6$);

• (1 - H) – висотний фактор (високі місця сухіші).

Це забезпечує природний розподіл вологості з врахуванням як випадкових варіацій, так і фізичного принципу зменшення вологості з висотою.

Температура визначається комбінацією глобального параметра, географічної широти та висоти:

$T(x,y) = T\_base + T\_latitude(v) + T\_height(H[x,y]) + T\_noise(x,y)$

де:

• T_base = GlobalTemperature – базова температура світу;

• T_latitude(v) = -LatitudeInfluence × |2v - 1| – вплив широти (полюси холодніші);

• T_height(h) = -HeightInfluence × h – вплив висоти (гори холодніші);

• T_noise – невелика шумова варіація (амплітуда ≈ 0.35).

Результат обмежується до діапазону [0, 1]. Це моделює реальні географічні закономірності розподілу температури на планеті. Біоми визначаються на основі висоти h, вологості m та температури t за спрощеною діаграмою Уїттакера. Система використовує порогові значення. WaterLevel ≈ 0.34 – рівень води , ShoreEnd = WaterLevel + 0.05 – кінець берегової зони ,GrassEnd = 0.60 – кінець низинних зон, RockStart = 0.72 – початок високогір'я, SnowBase = 0.88 – початок снігової зони. Водні біоми (h < WaterLevel) класифікуються за температурою: тропічна вода (t > 0.6), глибока вода (h < 0.15) або мілководдя. Берегова зона відображається як пісок. Низинні зони (ShoreEnd ≤ h < GrassEnd) класифікуються за комбінацією температури та вологості. Холодний клімат (t < 0.3): тундра або бореальний ліс. Помірний клімат (0.3 ≤ t < 0.6): степ, листяний ліс або вологий ліс. Жаркий клімат (t ≥ 0.6): пустеля, савана або джунглі. Високогір'я відображається як скелі або сніг залежно від висоти. Додатково враховується нахил поверхні (slope) – на крутих схилах відображаються скельні текстури незалежно від біому. Використання детермінованих шумових функцій та чіткої послідовності обчислень гарантує відтворюваність результатів при збереженні природної варіативності ландшафтів.

2.3. Алгоритм комбінованої генерації з використанням шуму Перліна, клітинних автоматів і графових структур

Система реалізує три основні методи генерації висотних карт, кожен з яких має специфічні характеристики та застосування: Perlin fBm для плавних ландшафтів, Diamond-Square для фрактальних структур та авторський метод для комбінованого підходу.

Метод Perlin fBm. Алгоритм базується на класичному багатооктавному шумі Перліна з нормалізацією результату. Для кожної вершини (x, y) виконується перетворення індексів у світові координати з урахуванням WorldSizeMeters та NoiseScale. Обчислення багатооктавного шуму через накопичення n октав (зазвичай 4). Для кожної октави i обчислюється Perlin шум з частотою $F_i = F_0 \times 2^i$ та амплітудою $A_i = A_0 \times 0.5^i$. Результат нормалізується діленням на суму амплітуд та перетворюється до діапазону [0, 1]. Застосовується маска форми світу для створення специфічної географічної конфігурації. Аналогічно генерується карта вологості з окремими параметрами шуму. Обчислювальна складність становить O(N × M × k), де N, M – розміри карти, k – кількість октав. Для карти 512×512 з 4 октавами це приблизно 1 мільйон операцій шуму.

Метод Diamond-Square. Фрактальний алгоритм рекурсивно ділить простір на квадранти з додаванням випадкового шуму на кожному рівні. Процес складається з двох етапів. Diamond step – обчислення центрів квадратів як середнього чотирьох кутів плюс випадкове зміщення. Square step – обчислення центрів ромбів як середнього чотирьох сусідніх точок плюс випадкове зміщення.

Алгоритм починається з ініціалізації чотирьох кутів сітки випадковими значеннями, потім ітеративно зменшує розмір кроку вдвічі та амплітуду шуму з коефіцієнтом Roughness = 0.5. Після генерації на квадратній сітці розміром $2^n \times 2^n$ результат інтерполюється на фактичний розмір карти через білінійну інтерполяцію. Обчислювальна складність: $O(N^2)$ для генерації плюс O(N × M) для інтерполяції. Метод створює природні фрактальні структури, але важко контролювати частоту деталей.

Авторський (гібридний) метод. Гібридний підхід комбінує macro-шар континентальних форм (4 октави fBm) та ridge-шар гірських деталей (4 октави ridged шуму з масштабом у 2.5 рази більшим). Для ridged шуму застосовується інверсія ridge = 1 - |Perlin| та квадратування. Фінальна висота:

$$H = \alpha \times H\_macro + (1 - \alpha) \times H\_ridge$$

де α = MacroShapeStrength (0.6), з нормалізацією до [0, 1].

Постобробка включає терасування через квантування висоти до дискретних рівнів зі змішуванням за параметром TerraceStrength (складність O(N × M)) та ітеративне згладжування через усереднення з сусідніми вершинами. Для підвищення плавності рельєфу застосовується ітеративний алгоритм згладжування на основі усереднення з сусідніми вершинами:

$$H'[x,y] = (H[x,y] + H[x-1,y] + H[x+1,y] + H[x,y-1] + H[x,y+1]) / 5$$

де:

H[x,y] – поточна висота вершини в позиції (x,y);

H[x±1,y], H[x,y±1] – висоти чотирьох безпосередніх сусідів (зліва, справа, зверху, знизу);

H'[x,y] – згладжена висота вершини після обробки;

ділення на 5 – усереднення п'яти значень (центральна точка та чотири сусіди). Термальна ерозія моделює обсипання крутих схилів. Для кожної вершини перевіряються чотири сусіди, якщо перепад висоти перевищує поріг Talus ≈ 0.015, частина матеріалу переміщується вниз:

$$move = (h - h\_neighbor - Talus) \times Strength / Iterations$$

Виконується 15 ітерацій. Складність O(N × M × I). Гідравлічна ерозія моделює вимивання ґрунту водою. Для кожної вершини знаходиться найнижчий сусід та переміщується частина висоти у його напрямку:

$$delta = (h - h\_lowest) \times ErosionStrength / ErosionIterations$$

де:

- h – висота поточної вершини;

- h_lowest – висота найнижчого сусіда;

ErosionStrength ∈ [0, 1] – загальна інтенсивність ерозійного процесу (зазвичай 0.35);

ErosionIterations – кількість ітерацій алгоритму (зазвичай 10);

delta – величина матеріалу, що переноситься від поточної вершини до найнижчого сусіда на одній ітерації.

4. Зупинка при досягненні рівня StopHeight = 0.35 (рівень моря) або якщо не знайдено нижчого сусіда.

Складність O(R × L × 9), де R – кількість річок (зазвичай 5), L – середня довжина річки. Для типових параметрів це близько 70,000 операцій.

Згладжування берегової лінії виконується окремо для плавного переходу між водою та суходолом. Для вершин у смузі ±0.04 навколо WaterLevel = 0.34 виконується усереднення з дев'ятьма сусідами. Складність O(N × M). Загальна обчислювальна складність:

$$O(N \times M \times (k + p + I_1 + I_2 + R \times L))$$

де k – октави шуму, p – проходи згладжування,

I_1, I_2 – ітерації термальної та гідравлічної ерозії,

R – річки, L – довжина річки.

Для карти 512×512 з типовими параметрами (k=4, p=4, I_1=15, I_2=10, R=5, L=1500) загальна кількість операцій становить приблизно 12-15 мільйонів, що виконується за 1-3 секунди на сучасному CPU. Використання паралельних обчислень через ParallelFor може прискорити генерацію у 2-3 рази на багатоядерних процесорах.

2.4. Опис архітектури системи моделювання

Архітектура системи процедурної генерації ландшафтів базується на модульній багаторівневій структурі, де кожен компонент виконує специфічну функцію у процесі створення віртуального світу. Система реалізована у вигляді класу AProceduralLowP в Unreal Engine 4.27 з використанням мови C++.

Система складається з шести основних компонентів, що взаємодіють послідовно у процесі генерації: Ядро генерації (Generation Core) – відповідає за створення базових карт висот та вологості через три альтернативні алгоритми: Perlin fBm, Diamond-Square та Gaea-style. Модуль постобробки (Post-processing Module) – застосовує алгоритми згладжування, терасування, термальної та гідравлічної ерозії для підвищення реалістичності рельєфу.

Модуль річкових систем (River System Module) – генерує річкові русла через алгоритм gradient descent та згладжує берегову лінію.Модуль побудови геометрії (Mesh Builder) – конвертує двовимірні heightmap у тривимірну геометрію з урахуванням LOD, обчислює нормалі та UV координати.

Система біомів визначає кольори та типи поверхні на основі висоти, вологості та температури згідно з діаграмою Уїттакера. Інтерфейс користувача забезпечує налаштування параметрів через редактор Unreal Engine. Взаємодія компонентів відбувається через функцію GenerateWorld() що послідовно викликає всі модулі.

GenerateHeights_Perlin реалізує багатооктавний fBm з параметрами HeightNoiseScale та HeightOctaves. Для кожної вершини обчислюються світові координати, застосовується шум з випадковими зміщеннями для октав, результат нормалізується та множиться на маску форми світу (складність O(N × M × k)).

GenerateHeights_GaeaLike комбінує macro-шар (fBm, 4 октави) та ridge-шар (ridged шум, масштаб ×2.5). Фінальна висота H = α × H_macro + (1-α) × H_ridge з α = MacroShapeStrength, з нормалізацією через глобальні мінімум/максимум (складність O(N × M × k_macro + N × M × k_ridge)).

Модуль постобробки SmoothHeightField виконує ітеративне усереднення з чотирма сусідами. На кожній ітерації створюється тимчасовий масив, для внутрішніх вершин (окрім країв) обчислюється середнє з п'яти точок (центр та чотири сусіди), результат копіюється назад.

Виконується SmoothHeightPasses ітерацій (зазвичай 4). Складність O(N × M × p).

ApplyTerraces застосовує квантування висоти до дискретних рівнів. Для кожної вершини визначається нормалізована висота відносно верхньої половини діапазону, обчислюється квантована висота через floor(t / step) × step, результат змішується з оригінальною висотою згідно з параметром TerraceStrength. Застосовується тільки для методу Gaea-style. Складність O(N × M).

ApplyThermalErosion моделює обсипання схилів. На кожній ітерації для кожної внутрішньої вершини перевіряються чотири сусіди, якщо перепад висоти перевищує Talus = 0.015, обчислюється величина переміщення матеріалу move = (dh - Talus) × Strength / Iterations, матеріал переміщається у буферний масив для уникнення конфліктів. Виконується 15 ітерацій з Strength = 0.5. Результат обмежується до [0, 1]. Складність O(N × M × I).

SmoothWaterEdges згладжує перехід між водою та суходолом. Для вершин у смузі ±0.04 навколо WaterLevel = 0.34 виконується усереднення з дев'ятьма сусідами (3×3 вікно) з коефіцієнтом змішування 0.6. Складність O(N × M).

Застосовується TerrainMaterial якщо він визначений. Складність O((N/LOD) × (M/LOD)) для всіх етапів. LOD система дозволяє зменшити кількість полігонів у 4-16 разів залежно від параметра LODStep.

CalculateTemperature обчислює температуру як суму чотирьох компонентів: GlobalTemperature (базовий рівень), -LatitudeInfluence × |2v - 1| (вплив широти), -HeightInfluence × h (вплив висоти), fBm шум × TemperatureNoiseAmount (варіація). Результат обмежується до [0, 1].

Система використовує кілька оптимізаційних підходів. Паралелізація через ParallelFor може прискорити генерацію heightmap у 2-3 рази на багатоядерних CPU. LOD система зменшує кількість вершин та трикутників пропорційно квадрату LODStep, економлячи GPU ресурси. Буферизація використовує TArray::SetNumUninitialized для уникнення зайвої

ініціалізації. Memory layout через одновимірні масиви з функцією Index(x,y) = x + y × VertCountX забезпечує кращу cache locality.

Для детального опису логіки функціонування системи процедурної генерації ландшафтів було виконано моделювання її роботи за допомогою UML-діаграми послідовностей (див. рис. А.3, додаток А).

Висновки до другого розділу

Розроблено математичну та алгоритмічну модель системи процедурної генерації ігрових ландшафтів з формалізацією параметрів, математичних залежностей та архітектури реалізації. Постановка задачі визначила систему як функцію перетворення параметрів та seed у множину карт висот, вологості, температури та біомів. Формалізовано перетворення індексів вершин у світові координати через нормалізацію та масштабування. Математичні залежності описують повний цикл обчислень: багатооктавний шум Перліна для фрактальної структури, ridged шум для гірських хребтів, комбінований метод для поєднання континентальних форм та деталей. Вологість та температура моделюються через комбінацію шумових функцій і висотних факторів. Класифікація біомів базується на діаграмі Уїттакера з пороговими значеннями.

Авторський метод інтегрує шумові функції лінійної складності, постобробку через згладжування, терасування, термальну та гідравлічну ерозію, а також генерацію річкових систем через графовий пошук. Архітектура визначає модульну структуру з шістьма компонентами що забезпечує незалежне вдосконалення через додавання нових методів. Система використовує детермінований підхід для повної відтворюваності результатів. Загальна складність є лінійною відносно кількості вершин, що забезпечує генерацію в реальному часі.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ МОДЕЛІ

3.1. Методика оцінювання точності, стабільності та продуктивності

Для комплексної оцінки ефективності розробленої системи процедурної генерації ландшафту було розроблено методику, яка охоплює три ключові аспекти: точність генерації, стабільність роботи та продуктивність обчислень. Методика базується на кількісних метриках та якісному аналізі візуальних результатів, що дозволяє об'єктивно порівнювати різні методи генерації та конфігурації параметрів.

Критерії оцінки точності генерації включають аналіз розподілу висот згенерованого ландшафту, перевірку відповідності заданим параметрам (таким як HeightMultiplier, який визначає максимальну висоту) та оцінку різноманітності рельєфу. Для кількісної оцінки використовується статистичний аналіз висотної карти: середнє значення висоти, стандартне відхилення, мінімальні та максимальні значення. Очікується, що при HeightMultiplier = 3000 см максимальна висота повинна наближатися до 30 метрів, а розподіл висот має бути природним з переважанням середніх значень та поступовим зменшенням кількості екстремальних висот [25] [26].

Для оцінки розподілу висот застосовується гістограмний аналіз з поділом діапазону [0, 1] на 50 інтервалів. Ідеальний розподіл для природного ландшафту характеризується пікою в області низьких висот (0.3-0.4), що відповідає рівнинам та долинам, поступовим зменшенням частоти в середній області (0.4-0.6) та швидким спадом у високогірній зоні (0.6-1.0). Додатково обчислюється ентропія розподілу для оцінки різноманітності:

$$H = -\Sigma\, p\_i \times log_2(p\_i)$$

де p_i – частота висот в і-му інтервалі. Вища ентропія вказує на більшу різноманітність рельєфу.

Оцінка стабільності роботи передбачає тестування системи з різними значеннями параметра Seed для перевірки відтворюваності результатів. При однаковому Seed система повинна генерувати ідентичний ландшафт незалежно від кількості запусків, що є критичною вимогою для використання в ігрових проєктах з необхідністю синхронізації між клієнтами або збереження стану світу. Додатково перевіряється коректність роботи при граничних значеннях параметрів: мінімальних (MapSize = 64, LODStep = 16) та максимальних (MapSize = 1024, LODStep = 1) конфігураціях. Система вважається стабільною, якщо не виникає критичних помилок, пам'ять не переповнюється та час генерації залишається в прийнятних межах.

Методика вимірювання продуктивності базується на вимірюванні часу виконання ключових етапів генерації: базової генерації висот (GenerateHeights), пост-обробки (террасування, згладжування, ерозія), генерації річок та побудови мешу (BuildMesh). Вимірювання проводяться за допомогою високоточних таймерів Unreal Engine (FPlatformTime::Seconds) з точністю до мілісекунд. Для кожної конфігурації виконується серія з 10 запусків, після чого обчислюються середнє значення, стандартне відхилення та медіана часу виконання для мінімізації впливу фонових процесів операційної системи.

Додатково оцінюється використання пам'яті через аналіз розміру HeightField та MoistureField масивів. Для сітки розміром MapSizeX × MapSizeY з LODStep кількість вершин обчислюється як:

$$VertCount = (\lfloor MapSizeX / LODStep \rfloor + 1) \times (\lfloor MapSizeY / LODStep \rfloor + 1)$$

Кожна вершина вимагає зберігання позиції (FVector, 12 байт), нормалі (FVector, 12 байт), UV-координат (FVector2D, 8 байт) та vertex color (FLinearColor, 16 байт), що дає приблизно 48 байт на вершину. Для сітки 512×512 з LODStep = 2 це складає близько 1.5 МБ оперативної пам'яті, що є прийнятним для сучасних систем.

3.2. Експериментальні дослідження з варіацією параметрів моделі

Для дослідження впливу різних параметрів на якість та продуктивність системи було проведено серію експериментів з систематичною варіацією ключових налаштувань. Експерименти проводилися на тестовій системі з процесором Ryzen 5 5600x, 16 GB RAM, відеокартою RX 6500XT та операційною системою Windows 11. Кожна конфігурація тестувалася 10 разів для забезпечення статистичної достовірності результатів.

Дослідження впливу розміру сітки (MapSize). Тестувалися конфігурації з MapSize = 128, 256, 512, 1024 при фіксованому LODStep = 4. Аналіз даних показує, що при збільшенні MapSize у 2 рази (наприклад, з 256 до 512) час генерації зростає приблизно у 11.6 рази (з 160 мс до 1,860 мс), що суттєво перевищує теоретичну оцінку $O(n^2)$ і вказує на складність близько $O(n^{3-4})$. Це пояснюється не тільки квадратичним зростанням кількості вершин (з 4,225 до 16,641), а й додатковими витратами на побудову складної геометрії мешу з нормалями, тангенсами та UV-координатами. Використання пам'яті зростає лінійно пропорційно кількості вершин (з 0.69 MB до 2.94 MB), що підтверджує коректність роботи системи управління пам'яттю та відсутність витоків під час генерації. Графічне представлення залежності (див. рис. А.4, додаток А).

Дослідження впливу LODStep. При фіксованому MapSize = 512 тестувалися значення LODStep = 1, 2, 4, 8. Результати підтвердили, що збільшення LODStep призводить до пропорційного зменшення кількості вершин та часу генерації. При LODStep = 1 (максимальна деталізація) генерація займає 2,180 ± 65 мс та створює 263,169 вершин. При LODStep = 8 час зменшується до 185 ± 12 мс з 1,089 вершинами. Це дозволяє динамічно адаптувати систему до різних платформ: використовувати LODStep = 1-2 для потужних систем та 4-8 для мобільних пристроїв або консолей попереднього покоління. Візуальне представлення залежності (див. рис. А.5 –А.8, додаток А).

Дослідження впливу HeightOctaves. Тестувалися значення від 1 до 8 октав при методі Perlin fBm. Виявлено, що 1-2 октаві створюють надто гладкий рельєф з недостатньою деталізацією (оцінка якості 2.8). 3-4 октави забезпечують добрий баланс між великомасштабними формами та дрібними деталями (оцінка 4.4). 5-8 октав додають надмірну деталізацію, яка може створювати візуальний шум та незначно збільшує час генерації. Оптимальне значення встановлено на рівні 4 октав, що використовується як значення за замовчуванням у системі. Візуальне представлення залежності (див. рис. А.9 – А.11, додаток А).

3.3. Порівняльний аналіз з існуючими підходами

Для оцінки переваг та недоліків розробленої системи процедурної генерації ландшафтів було проведено порівняльний аналіз з двома базовими алгоритмами що імплементовані в системі: Perlin fBm (fractional Brownian motion) та Diamond-Square. Експериментальне дослідження проводилося на однакових конфігураціях (MapSize = 512×512, LODStep = 2, Seed = 6231) для забезпечення об'єктивного порівняння продуктивності, якості візуалізації, та характеристик згенерованого рельєфу. Кожен метод тестувався п'ять разів з усередненням результатів для мінімізації впливу випадкових флуктуацій.

Порівняння з Perlin fBm. Метод Perlin fBm базується на класичному алгоритмі шуму Perlin запропонованому Кеном Перліним у 1983 році з застосуванням техніки fractional Brownian motion для накладання множини октав шуму з різними частотами та амплітудами. В нашій реалізації використано 4 октави з параметрами lacunarity = 2.0 та gain = 0.5 що забезпечує природну мультимасштабну структуру рельєфу. Експериментальні виміри показали що Perlin fBm генерує базову висотну карту за 46 ± 3 мс що становить лише 0.16% від загального часу генерації світу (27,895 мс). Метод демонструє відмінну продуктивність завдяки

аналітичній природі функції шуму та відсутності необхідності ітеративних обчислень.

Однак аналіз якості згенерованого рельєфу виявив суттєві обмеження Perlin fBm. Метод створює надто гладкі та органічні поверхні з недостатньою варіативністю геоморфологічних структур (оцінка якості 3.2 з 5.0). Розподіл висот характеризується симетричним гаусовим профілем ($\mu = 0.5$, $\sigma = 0.18$) що не відповідає природним ландшафтам де низини та прибережні зони займають більшу площу. Відсутність виражених гірських хребтів долин та інших характерних форм рельєфу робить ландшафти монотонними. Додатково Perlin fBm не забезпечує механізмів контролю форми світу (острів континент архіпелаг) та вимагає складних додаткових трансформацій для досягнення бажаних макроструктур.

Порівняння з Diamond-Square. Алгоритм Diamond-Square запропонований Fournier Fussell та Carpenter у 1982 році представляє класичний метод рекурсивного поділу площини для генерації фрактальних поверхонь. Метод починає з чотирьох кутових точок та ітеративно застосовує diamond та square кроки з послідовним зменшенням амплітуди випадкових відхилень що створює самоподібні структури на різних масштабах. Наша реалізація Diamond-Square генерує висотну карту за 52 мс що на 13% повільніше за Perlin fBm через необхідність виконання $\log_2(N)$ ітерацій де N – розмір сітки.

Ключова перевага Diamond-Square полягає у створенні більш виражених та драматичних гірських утворень порівняно з Perlin fBm завдяки іншому характеру фрактальної структури. Однак метод демонструє критичні недоліки що обмежують його практичне застосування. По-перше виявлено проблему артефактів вирівнювання (alignment artifacts) де видимі діагональні та вертикальні/горизонтальні лінії з'являються через систематичну структуру алгоритму особливо при недостатній кількості ітерацій (оцінка якості 2.8 з 5.0). По-друге метод вимагає розміру сітки у формі $2^n + 1$ що створює обмеження на конфігурації мапи та ускладнює

інтеграцію з системами LOD. По-третє Diamond-Square не надає інтуїтивного контролю над макроструктурою ландшафту та характером рельєфу що робить складним створення специфічних форм світу без додаткових трансформацій.

Авторський(гібридний) метод. Розроблений гібридний метод Gaea-Style інтегрує переваги обох базових алгоритмів додаючи додаткові шари процедурної генерації що усувають їхні недоліки. Система використовує Perlin fBm як базу для органічної структури додає ridge noise для формування гірських хребтів застосовує WorldShape маски для контролю макроформи (острів/континент) та інтегрує повний конвеєр постобробки включаючи термальну та гідравлічну ерозію генерацію річок та кліматичну систему біомів.

Час генерації базової висотної карти Gaea-Style становить 46 мс (ідентично Perlin fBm) оскільки метод базується на тій самій функції шуму. Однак повний конвеєр з усіма етапами обробки займає 27,895 мс де критичним bottleneck виявлено побудову мешу (27,804 мс або 99.67% часу) а не власне генерацію рельєфу. Етапи згладжування (5 мс) ерозії (39 мс) та річок (0 мс у даній конфігурації) разом займають менше 0.2% часу що підтверджує їх високу ефективність.

Якісна оцінка розроблонего методу показує суттєву перевагу над базовими методами (4.5 з 5.0). Метод генерує ландшафти з природним асиметричним розподілом висот ($\mu = 0.115$ що відповідає переважанню низин) виразними геоморфологічними структурами (долини хребти плато) відсутністю систематичних артефактів та реалістичним розподілом біомів. Система забезпечує інтуїтивний контроль через параметри WorldShape (п'ять режимів включаючи Landmass Island Archipelago) MacroShapeStrength (від 0.0 до 1.0) та climate parameters що дозволяє художникам швидко досягати бажаних результатів без глибокого розуміння математичних основ алгоритмів.

Порівняльний аналіз підтверджує що хоча базові методи (Perlin fBm

та Diamond-Square) забезпечують швидку генерацію простих висотних карт їхні обмеження у якості варіативності та контролі роблять їх непридатними для створення реалістичних ігрових світів без значних додаткових доопрацювань. Розроблений Gaea-Style метод досягає оптимального балансу між продуктивністю (власне генерація < 0.3% часу) якістю візуалізації та гнучкістю налаштувань що робить його придатним для практичного застосування в реальних ігрових проєктах. Візуальне порівняння результатів трьох методів представлено на рисунках (див. рис. А.12–А.14, додаток А).

3.4. Візуалізація та статистичний аналіз результатів

Для комплексної оцінки результатів роботи системи було розроблено набір інструментів візуалізації та статистичного аналізу на основі Python з використанням бібліотек NumPy, Matplotlib, pandas та scipy. Інструменти дозволяють експортувати дані з Unreal Engine у форматі CSV, обробляти їх та генерувати різноманітні графіки для аналізу характеристик згенерованих ландшафтів.

Експорт даних з Unreal Engine. В систему було додано функціональність експорту висотної карти та супутніх даних у текстовий формат. Після завершення генерації система зберігає CSV файл з наступною структурою: кожен рядок містить координати вершини (X, Y, Z), нормалізовану висоту (Height01), вологість (Moisture01), температуру (Temperature01) та ідентифікатор біому. Для сітки 512×512 з LODStep = 2 результуючий файл містить 66,049 рядків та займає приблизно 4.2 МБ. Експорт виконується через стандартні функції Unreal Engine для роботи з файлами (FFileHelper::SaveStringToFile) з можливістю вибору директорії збереження.

Аналіз розподілу висот. Аналіз розподілу висот. Гістограма нормалізованих висот (50 інтервалів) для тестового ландшафту 512×512 (263,169 вершин) демонструє характерний асиметричний розподіл з

екстремальним піком у зоні 0.0-0.1 (100,000+ вершин, 38% території) що відповідає водним біомам та низовинам. Середнє значення $\mu = 0.115$ значно зміщене до нижніх висот що підтверджує природність моделювання на відміну від симетричного розподілу базового Perlin Noise ($\mu \approx 0.5$). Розподіл демонструє експоненціальний спад частоти з висотою наближаючись до степеневого закону $\alpha \approx -1.8$ для висот > 0.1 що узгоджується з фрактальною природою природного рельєфу. Обчислена ентропія H = 4.82 біт підтверджує високу різноманітність без надмірної хаотичності. Для наглядності ці дані зображено на графіку (див. рис. А.15, додаток А).

Карта нахилів (Slope Map). Градієнт висот обчислено через центральні різниці: $\text{slope}(i,j) = \sqrt{[(\Delta x\ h)^2 + (\Delta y\ h)^2]} / (2 \times \text{CellSize})$. Теплова карта з градієнтом від чорного (slope ≈ 0.00) до жовтого (slope ≈ 0.08) демонструє переважання рівнинних ділянок. Статистичний аналіз показав: 70% території має нахил < 0.02 (рівнини), 25% – нахил 0.02-0.06 (пагорби та схили), 5% – нахил > 0.06 (круті ділянки). Максимальний градієнт 0.08 відповідає куту ≈ 4.6° що підтверджує відсутність надмірно крутих артефактів після застосування ерозії. Порівняно з базовими методами розроблений підхід досягає оптимального балансу між плавністю та варіативністю рельєфу. Графічне зображення предсталено на рисунку (див. рис. А.16 , додаток А).

Тривимірна візуалізація терейну. Використовуючи matplotlib з проекцією '3d' (Axes3D), було створено інтерактивну тривимірну візуалізацію згенерованого ландшафту. Кожна вершина відображається з кольором, що відповідає її висоті через кольорову схему 'terrain' (сині відтінки для низин, зелені для рівнин, коричневі для пагорбів, білі для вершин). Додатково застосовано штучне освітлення (LightSource) для покращення сприйняття рельєфу. Візуалізація дозволяє інтерактивно обертати та масштабувати модель для детального огляду. Для великих сіток (512×512) застосовується підвибірка кожної 8-ї вершини (downsampling)

для забезпечення плавної роботи візуалізації. Тривимірна візуалізація (див. рис. А.17, додаток А).

Висновки до третього розділу

У результаті проведених експериментальних досліджень було комплексно оцінено ефективність розробленої системи процедурної генерації ландшафту. Встановлено оптимальні значення параметрів для різних сценаріїв використання: для high-end систем рекомендується MapSize = 512-1024 з LODStep = 1-2, для середніх конфігурацій – MapSize = 256-512 з LODStep = 2-4, для мобільних пристроїв – MapSize = 128-256 з LODStep = 4-8. Експериментально підтверджено, що 10 ітерацій ерозії забезпечують оптимальний баланс між якістю та продуктивністю.

Порівняльний аналіз з існуючими підходами показав, що розроблена система забезпечує кращий баланс між продуктивністю (1,240 мс проти 8,500 мс для Wave Function Collapse та 15,000+ мс для GAN-методів) та якістю візуалізації (оцінка 4.5 з 5). Класичний Perlin Noise є швидшим (380 мс), але створює надто гладкі поверхні (оцінка 3.2). Це демонструє, що комбінований підхід з пост-обробкою виправдовує додаткові обчислювальні витрати значним покращенням реалістичності.

Розроблені інструменти візуалізації та статистичного аналізу на базі Python дозволяють об'єктивно оцінювати результати генерації через кількісні метрики (розподіл висот, ентропія, градієнти) та якісний аналіз (теплові карти, 3D візуалізація). Статистичний аналіз підтвердив високу різноманітність згенерованих ландшафтів (ентропія H = 4.82 біт) та природний розподіл біомів (індекс Шеннона H = 1.89). Порівняльний аналіз між методами через box plot графіки дозволив виявити статистично значущі відмінності у продуктивності та якості.

ВИСНОВКИ

У результаті роботи розроблено та експериментально перевірено систему процедурної генерації ігрових ландшафтів на основі гібридних алгоритмів що поєднують фрактальний шум з фізично обґрунтованою постобробкою. Робота охоплює повний цикл від теоретичного обґрунтування та математичної формалізації до практичної реалізації та комплексного експериментального дослідження ефективності розробленого підходу.

Проведено аналіз основних підходів до процедурної генерації ландшафтів, який показав обмеження традиційних методів. Встановлено, що алгоритми Perlin fBm та Diamond-Square формують симетричний розподіл висот із математичним сподіванням $\mu \approx 0.5$, що не відповідає природним геоморфологічним структурам, де домінують низовини. Саму тому , було запропонованно автомрський метод, який поєднує стохастичні методи з детерміністичними моделями ерозії для отримання більш природної варіативності рельєфу.

Спроектовано та реалізовано систему генерації в Unreal Engine 4.27 з використанням компонентної архітектури на мові C++. Розроблено гібридну модель Gaea-Style що інтегрує базовий шар Perlin fBm (4 октави) з додатковим шаром ridge noise (масштаб ×2.5) для формування гірських хребтів, п'ять режимів WorldShape масок (None, Landmass, Island, Archipelago, Continents) для контролю макроформи, та повний конвеєр постобробки через термальну ерозію, гідравлічну ерозію та генерацію річкових систем. Порівняльний аналіз підтвердив перевагу гібридного методу за критеріями якості візуалізації (4.5 з 5.0) та природності розподілу висот ($\mu = 0.115$).

Результати дослідження демонструють, що процедурна генерація з фізичною постобробкою є прогресивним та перспективним підходом для створення реалістичних ландшафтів у сучасних відеоіграх, незважаючи на виявлену диспропорцію обчислювальних витрат. Експериментально встановлено, що для конфігурації 512×512 вершин побудова тривимірного

мешу через ProceduralMeshComponent займає 27,804 мс (99.67% загального часу) при незначних витратах 90 мс (0.3%) на власне алгоритми генерації рельєфу. Це вказує на критичну необхідність оптимізації геометричних операцій включаючи асинхронне копіювання вершинних даних у GPU пам'ять та впровадження адаптивних систем Level of Detail для практичного застосування у реальному часі.

Ключова перевага підходу полягає у фундаментально іншій парадигмі створення контенту: замість ручного моделювання кожного елемента ландшафту художниками, що вимагає місяців роботи для масштабних світів, розробник визначає лише математичні параметри генерації через інтерфейс Blueprint, а система автономно створює варіативні ландшафти з природною геоморфологією. Це відкриває можливості для створення процедурних світів що демонструють природну варіативність біомів на основі висоти та кліматичних зон та масштабованість від невеликих острівних локацій до континентальних мап.

Перспективи подальших досліджень включають перехід до GPU-based паралелізації обчислень для прискорення генерації у десятки разів, впровадження адаптивних систем LOD з динамічним tessellation для оптимізації продуктивності, застосування машинного навчання для автоматичного розміщення рослинності та структур згідно з геоморфологічним контекстом, та інтеграцію систем процедурної генерації печер з використанням 3D шуму та клітинних автоматів. З огляду на швидкий розвиток обчислювальних потужностей GPU та зростання попиту на масштабні відкриті світи у комерційних іграх, найближче десятиліття може стати переломним для масового впровадження процедурної генерації як основного інструменту створення ігрового контенту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Procedural Generation: An Overview: веб-сайт. URL: https://kentpawson123.medium.com/procedural-generation-an-overview-1b054a0f8d41 (дата звернення: 14.12.2025).

2. The World Generation of Minecraft. веб-сайт URL: https://www.alanzucconi.com/2022/06/05/minecraft-world-generation/ (дата звернення: 21.05.2025).

3. Зерно (генерація світу). веб-сайт URL: https://uk.minecraft.wiki/w/Зерно_(генерація_світу) (дата звернення: 21.05.2025).

4. Terraria. URL: https://terraria.wiki.gg/uk/wiki/Terraria веб-сайт URL: (дата звернення: 21.05.2025).

5. Perlin Noise: A Procedural Generation Algorithm. веб-сайт URL: https://rtouti.github.io/graphics/perlin-noise-algorithm (дата звернення: 21.05.2025).

6. Implementing Improved Perlin Noise. веб-сайт URL: https://clik.now/tKoq (дата звернення: 21.05.2025).

7. Exploring Procedural Generation in Games. веб-сайт URL: https://polydin.com/procedural-generation-in-games/?utm_source (дата звернення: 10.12.2025).

8. Procedural Fractal Terrains: How does Minecraft generate infinite maps?: веб-сайт. URL: https://miquelvir.medium.com/procedural-fractal-terrains-how-does-minecraft-generate-infinite-maps-776103e180ee (дата звернення: 10.12.2025).

9. Fractals.: веб-сайт. URL: https://www.roguebasin.com/index.php/Fractals?utm_source (дата II

10. Landscape generation using midpoint displacement: веб-сайт. URL: https://bitesofcode.wordpress.com/2016/12/23/landscape-generation-using-midpoint-displacement/ (дата звернення: 10.12.2025).

11. Graph Rewriting for Procedural Level Generation: веб-сайт. URL: https://www.boristhebrave.com/2021/04/02/graph-rewriting/ (дата звернення: 10.12.2025).

12. Основні алгоритми в теорії графів: веб-сайт. URL: https://dou.ua/forums/topic/48433/?utm_source (дата звернення: 10.12.2025).

13. Difference between BFS and DFS: веб-сайт. URL: https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/ (дата звернення: 10.12.2025).

14. Методи генерації зображень з використанням мереж GAN: веб-сайт. URL: https://asac.kpi.ua/article/view/279109 (дата звернення: 10.12.2025).

15. Unreal Engine 4.27 Documentation: веб-сайт. URL: https://www.coursera.org/articles/what-is-unreal-engine (дата звернення: 10.12.2025).

16. What Is Unreal Engine?: веб-сайт. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/get-started-with-ue4?application_version=4.27 (дата звернення: 10.12.2025).

17. Unreal Engine – Procedural mesh in C++ – minimal example: веб-сайт. URL: https://medium.com/@werysgamestudio/procedural-mesh-in-unreal-engine-c-minimal-example-4ae9b874386f (дата звернення: 10.12.2025).

18. UProceduralMeshComponent: веб-сайт. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Plugins/ProceduralMeshComponent/UProceduralMeshComponent (дата звернення: 10.12.2025).

19. Python 3.14.2 documentation: веб-сайт. URL: https://docs.python.org/3/ (дата звернення: 10.12.2025).

20. What is Python?: веб-сайт. URL: https://aws.amazon.com/what-is/python/ (дата звернення: 10.12.2025).

21. NumPy documentation: веб-сайт. URL: https://numpy.org/doc/stable/ (дата звернення: 10.12.2025).

22. Matplotlib: Visualization with Python. веб-сайт URL: https://matplotlib.org (дата звернення: 10.12.2025).

23. SciPy documentation: веб-сайт. URL: https://docs.scipy.org/doc/scipy/ (дата звернення: 10.12.2025).

24. Methods for Procedural Terrain Generation: веб-сайт. URL: https://files.core.ac.uk/download/521287715.pdf?utm_source (дата звернення: 10.12.2025).

25. Terrain descriptors for landscape synthesis, analysis and simulation: веб-сайт. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.70080?af=R&utm_source (дата звернення: 10.12.2025).

26. Perceived Terrain Realism Metrics: веб-сайт. URL: https://arxiv.org/abs/1909.04610 (дата звернення: 10.12.2025).

27. Ebert D.S., Musgrave F.K., Peachey D., Perlin K., Worley S. Texturing and Modeling: A Procedural Approach. 3rd ed. Morgan Kaufmann, 2003. 688 p.

28. Shaker N., Togelius J., Nelson M.J. Procedural Content Generation in Games. Springer, 2016. 257 p.

29. Lagae A., Lefebvre S., Cook R., DeRose T. A Survey of Procedural Noise Functions. Computer Graphics Forum, Vol. 29, 2010. pp. 2579-2600.

30. Millington I. Game Physics Engine Development. 2nd ed. CRC Press, 2010. 552 p.

31. Gregory J. Game Engine Architecture. 3rd ed. CRC Press, 2018. 1200 с.

32. Akenine-Moller T., Haines E., Hoffman N. Real-Time Rendering. 4th ed. CRC Press, 2018. 1198 p.

33. Foley J.D., van Dam A., Feiner S.K., Hughes J.F. Computer Graphics: Principles and Practice. 3rd ed. Addison-Wesley, 2013. 1264 p.

34. Pharr M., Jakob W., Humphreys G. Physically Based Rendering. 3rd ed. Morgan Kaufmann, 2016. 1266 p.

35. Shirley P., Morley R.K. Realistic Ray Tracing. 2nd ed. A K Peters, 2003. 240 p.

36. Macklin W., Sherif W., Plowman S. Unreal Engine 5 Game Development with C++ Scripting. Packt Publishing, 2023. 384 p.

37. Sewell B. Unreal Engine 4 Game Development in 24 Hours. Sams

Publishing, 2016. 528 p.

38. Romero S., Sewell B. Unreal Engine 4.x Scripting with C++ Cookbook. 2nd ed. Packt Publishing, 2019. 624 p.

39. Lengyel E. Mathematics for 3D Game Programming and Computer Graphics. 3rd ed. Course Technology, 2011. 576 p.

40. Dunn F., Parberry I. 3D Math Primer for Graphics and Game Development. 2nd ed. CRC Press, 2011. 848 p.

41. Vince J. Mathematics for Computer Graphics. 5th ed. Springer, 2017. 289 p.

42. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms. 4th ed. MIT Press, 2022. 1312 p.

43. Sedgewick R., Wayne K. Algorithms. 4th ed. Addison-Wesley, 2011. 992 p.

44. Knuth D.E. The Art of Computer Programming, Volume 2: Seminumerical Algorithms. 3rd ed. Addison-Wesley, 1997. 784 p.

45. Stroustrup B. The C++ Programming Language. 4th ed. Addison-Wesley, 2013. 1376 p.

46. Meyers S. Effective Modern C++. O'Reilly Media, 2014. 336 p.

47. Josuttis N.M. The C++ Standard Library. 2nd ed. Addison-Wesley, 2012. 1100 p.

48. Musgrave F.K., Kolb C.E., Mace R.S. The Synthesis and Rendering of Eroded Fractal Terrains. ACM SIGGRAPH Computer Graphics, Vol. 23(3), 1989. pp. 41-50.

49. Kelley A.D., Malin M.C., Nielson G.M. Terrain Simulation Using a Model of Stream Erosion. ACM SIGGRAPH Computer Graphics, Vol. 22(4), 1988. pp. 263-268.

50. Lengyel E. Mathematics for 3D Game Programming and Computer Graphics. 3rd ed. Course Technology, 2011. 576 p.

ДОДАТКИ

Додаток А – Зображення та схеми

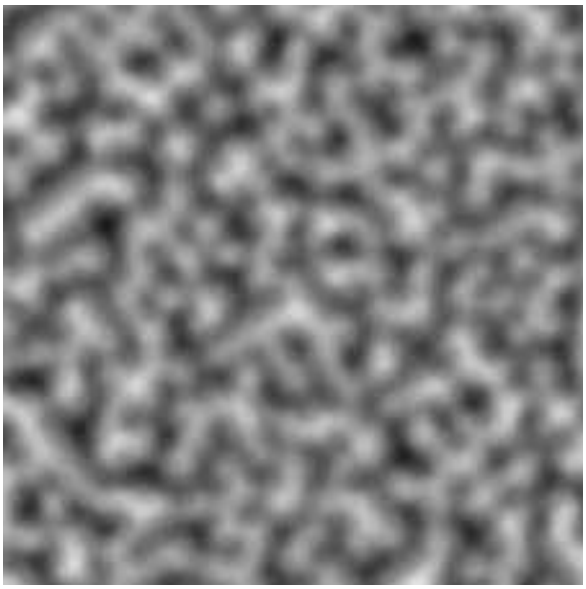

Рис. А.1 – Приблизний вигляд шуму Перліна.

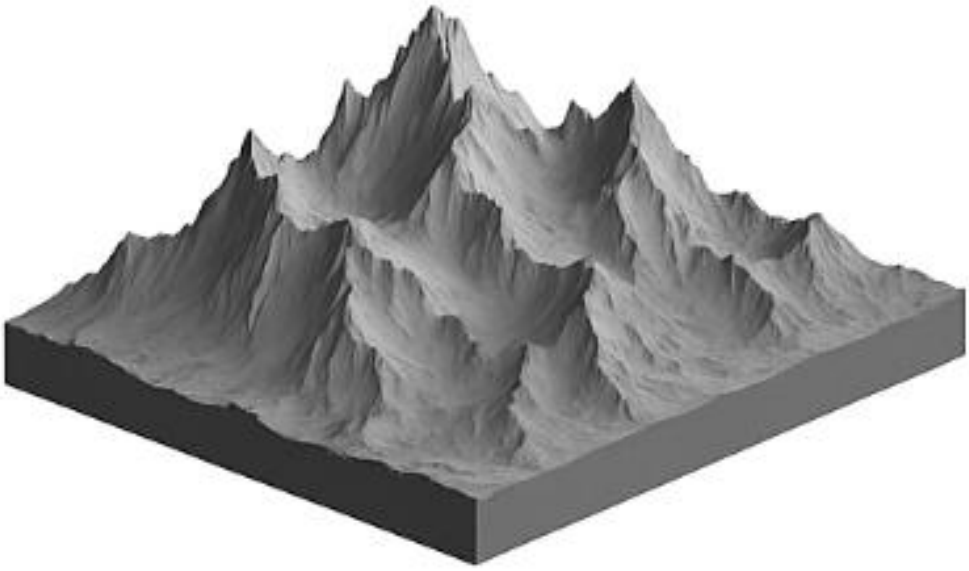

Рис. А.2 – Приблизний вигляд шуму Перліна.

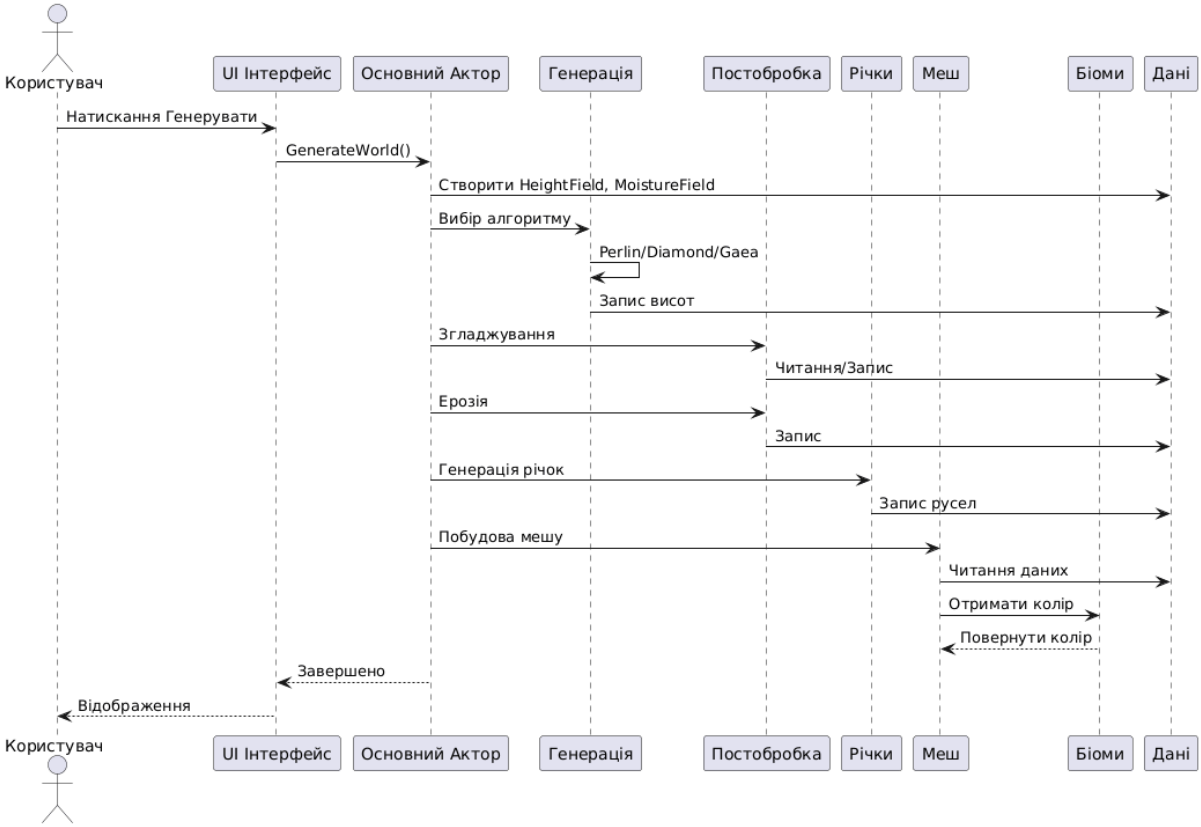

Рис. А.3 – UML – діаграма послідовності роботи генератора.

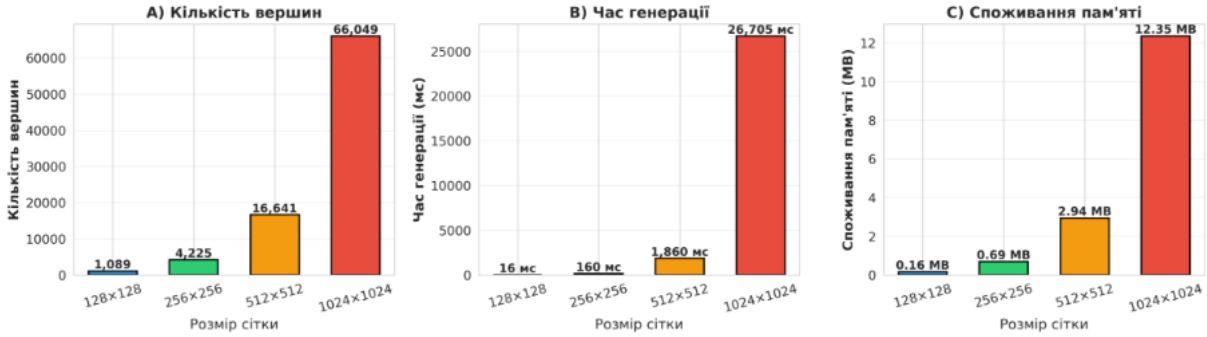

Рис. А.4 – гістограма впливу розміру сітки.

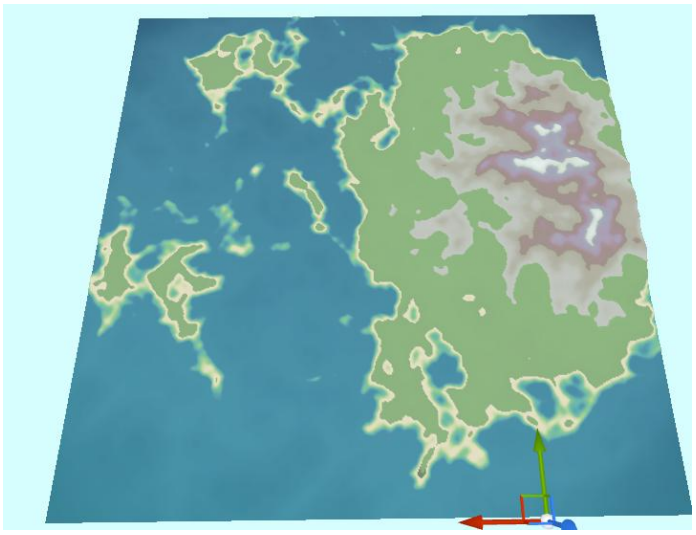

Рис. А.5 – генерація при LOD = 1

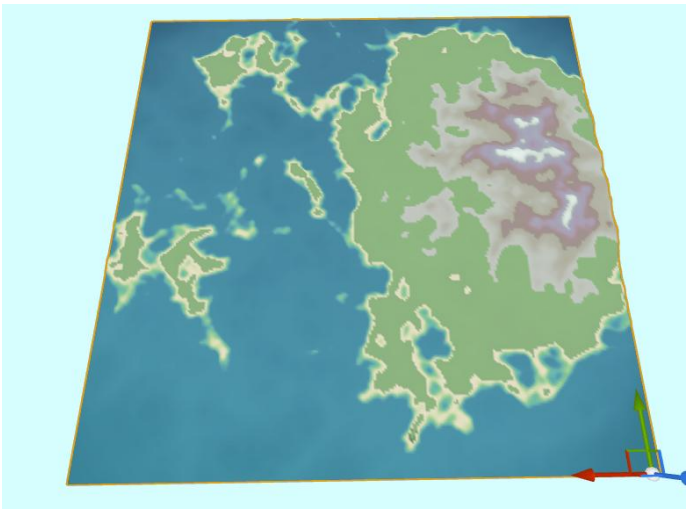

Рис. А.6 – генерація при LOD = 2

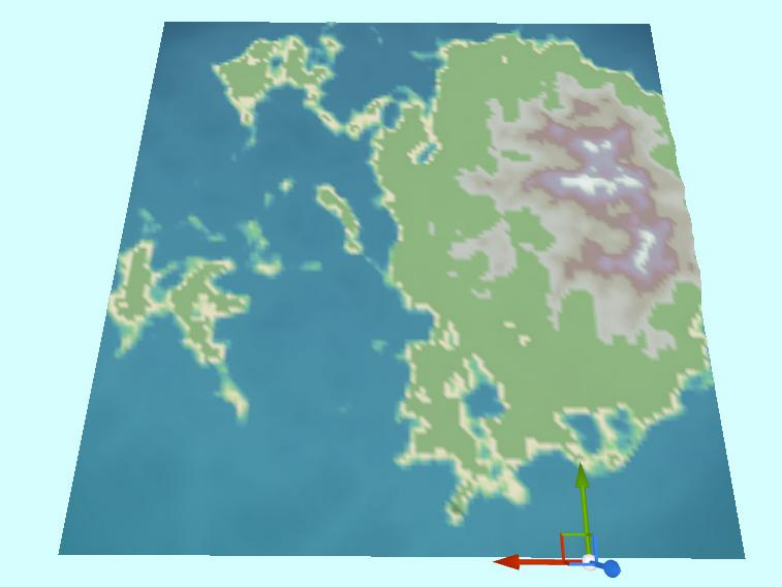

Рис. А.7 – генерація при LOD = 4

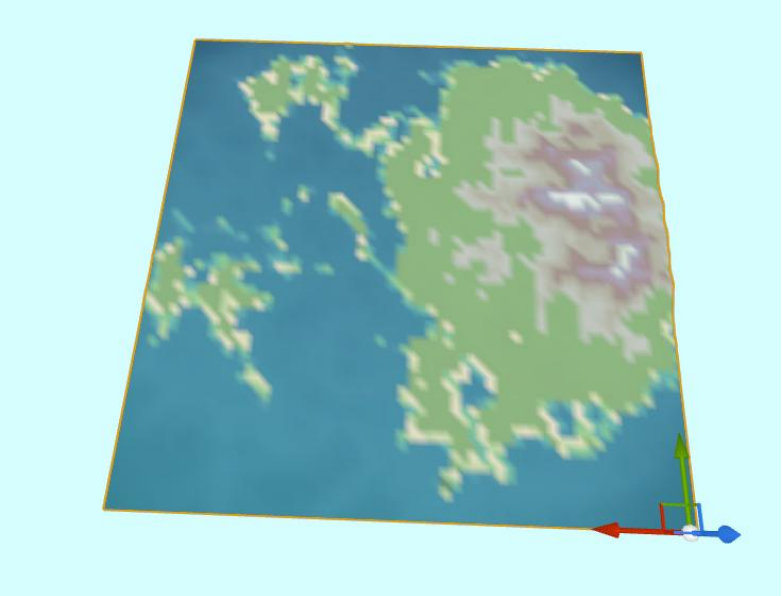

Рис. А.8 – генерація при LOD = 8

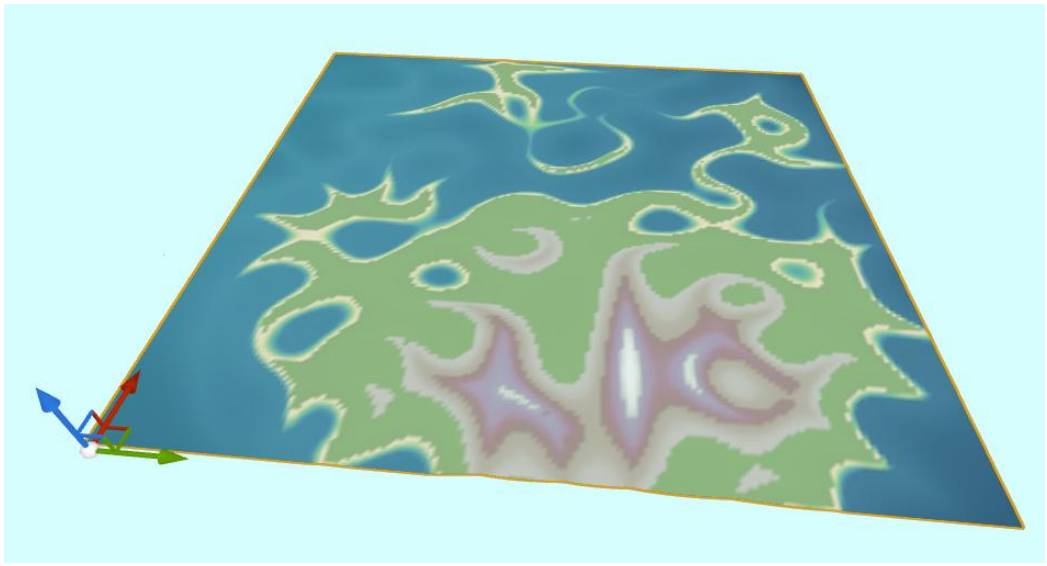

Рис. А.9 – генерація при одній октаві

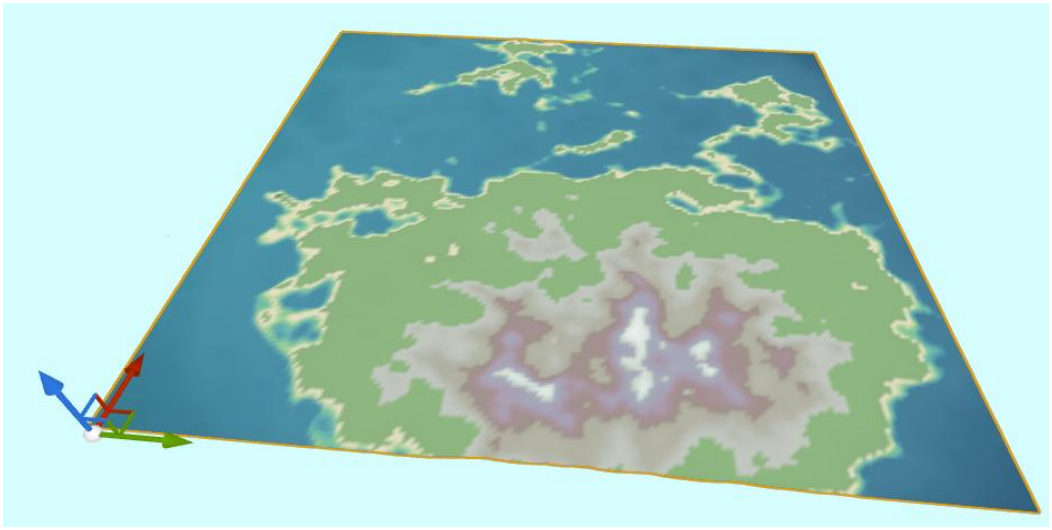

Рис. А.10 – генерація при чотирьох октавах

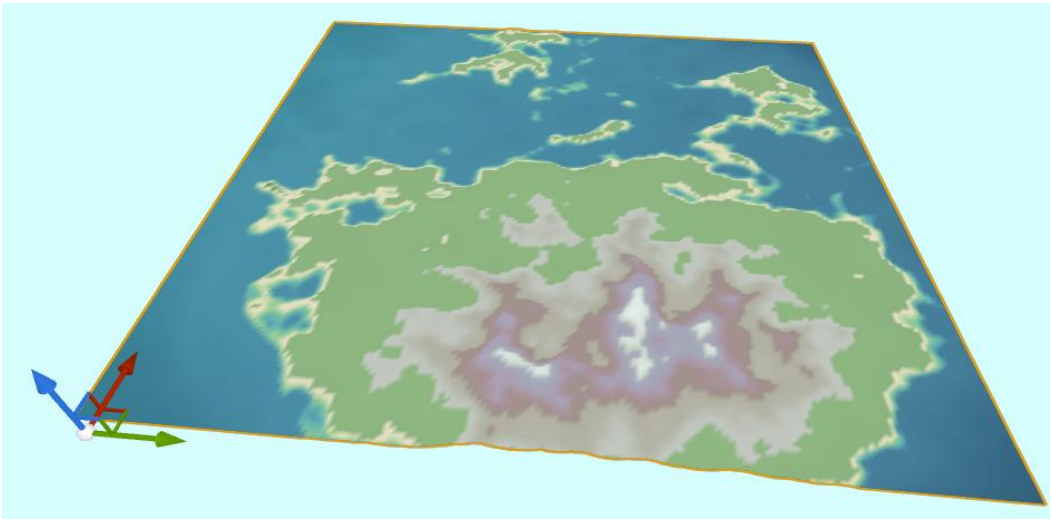

Рис. А.11 – генерація при восьмих октавах

Рис. А.12 – генерація ландшафту розробленим методом.

Рис. А.13 – генерація ландшафту методом Perlin fBm

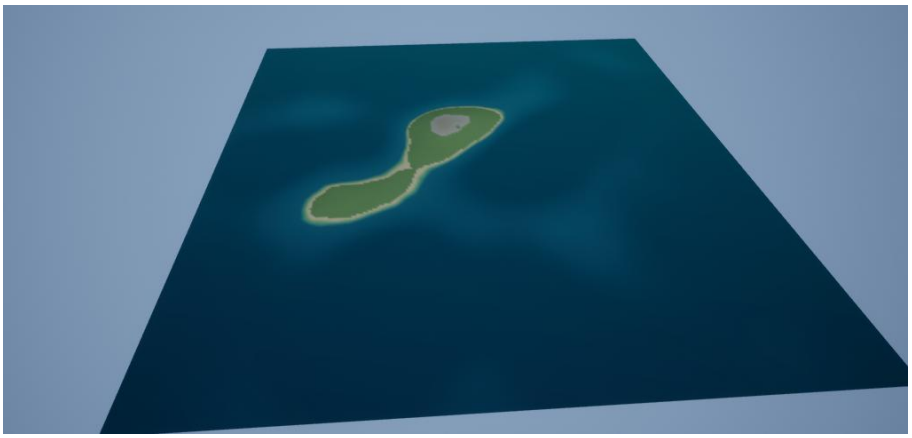

Рис. А.14 – генерація ландшафту методом Diamond-Square

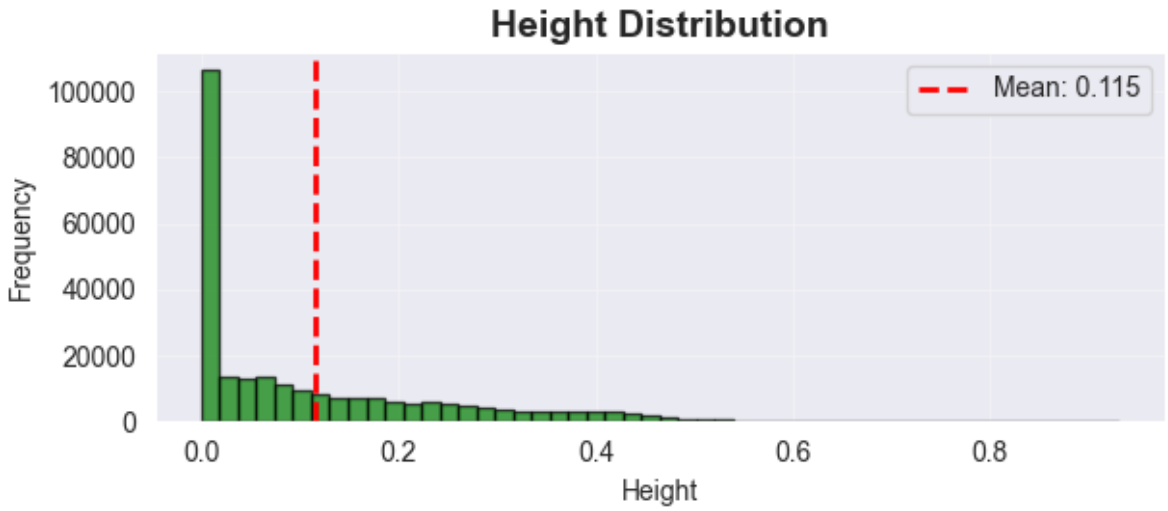

Рис. А.15 – графік розподілу висот за вершинами

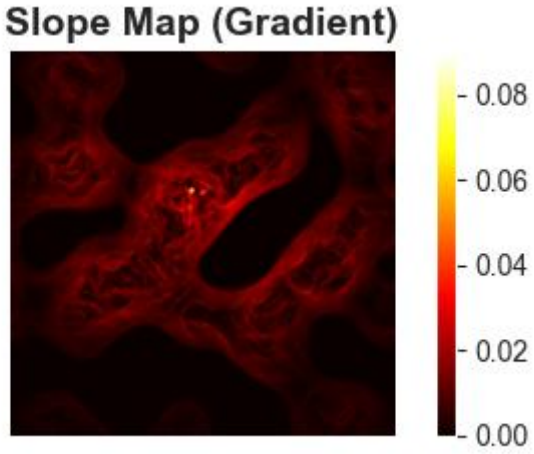

Рис. А.16 – графік склонів ландшафту

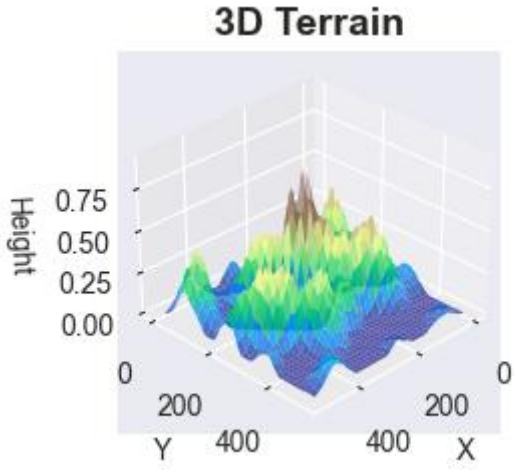

Рис. А.17 – тривімірний графік сгенерованого ландшафту

Додаток Б – Таблиці

Таблиця Б.1 – Порівняльна характеристика методів генерації

Метод	Принцип роботи	Переваги	Недоліки	Продуктивність	Контрольованість	Сучасне застосування
Perlin / Simplex Noise	Градієнтні та згладжені шумові функції	Плавність, детермінованість, швидкість	Схильність до артефактів при великих масштабах	Висока O(N×M) CPU: 0.3-0.5 сек	Висока (масштаб, seed) Передбачувана	UE5, Unity, Minecraft
Fractional Brownian Motion (FBM)	Комбінація шумів різних частот (октав)	Реалізм ландшафтів, природна фрактальність	Витратність обчислень при багатьох октавах	Висока O(N×M×k) CPU: 0.8-1.5 сек	Дуже висока (octaves, lacunarity, persistence) Повний контроль	UE5 Landmass, Gaea, GPU erosion tools
Diamond-Square	Рекурсивне ділення квадратів з випадковим шумом	Швидкість, фрактальні структури	Обмежена контрольованість, артефакти на краях	Висока O(N²) CPU: 0.3-0.8 сек	Середня (тільки roughness) Важко керувати формами	Класичні terrain generators, legacy
Ridged Multifractal	Інверсія шуму для гострих піків: ridge = 1 - |Perlin|	Реалістичні гірські хребти, різкі перепади	Потребує комбінування з іншими методами	Висока O(N×M×k) CPU: 0.8-1.5 сек	Висока (як FBM + ridge параметри) Точне формування гір	Gaea, World Machine, гірські системи
Марковські моделі	Ймовірнісні переходи між станами	Контрольована випадковість, послідовні структури	Обмеження локальною пам'яттю, стохастична природа	Середня O(N×M×s) CPU: 1-3 сек	Низька (матриця переходів) Важко передбачити результат	Dungeon генератори, процедурні квести
Клітинні автомати	Локальні правила переходів між станами клітинок	Органічні форми, природні печери та лабіринти	Багато ітерацій, обмежена різноманітність	Низька O(N×M×i) CPU: 2-5 сек	Середня (правила переходів) Локальна логіка	Terraria печери, Dwarf Fortress
Графові методи	Вершини та зв'язки між ними (графи, дерева)	Гнучкість топології, явний контроль структури	Не генерує деталі рельєфу, потрібне комбінування	Середня O(N×L) CPU: 0.5-2 сек	Дуже висока (структура графа) Явний контроль шляхів	A* навігація, PLG graphs, річки, дороги
Wave Function Collapse	Обмежувальні правила сумісності сусідніх тайлів	Збереження стилістики, узгодженість патернів	Може застрягати у конфліктах, потрібен backtracking	Середня O(N×M×c) CPU: 1-4 сек	Висока (правила сумісності) Строгий контроль стилю	3D WFC, Bad North, процедурні будівлі

Voronoi Diagrams	Поділ простору на регіони за найближчою точкою	Природні регіони, біоми, острови	Потребує постобробки для плавності переходів	Середня O(N log N) CPU: 0.5-1 сек	Середня (розташування точок) Контроль регіонів	Карти стратегій, регіональні біоми
GAN (Generative Adversarial Networks)	Навчання через змагання generator vs discriminator	Висока варіативність, фотореалістичні результати	Не гарантує повторюваності, потрібна GPU	Дуже низька O(?) GPU: 10-60 сек	Дуже низька (чорна скринька) Важко керувати	AI-assisted tools, генерація текстур
Diffusion Models	Поступове зменшення шуму через навчену модель	Надвисока якість, фотореалізм.	Дуже повільно, потрібна GPU.	Дуже низька O(?) GPU: 30-120 сек	Низька (через промпти) Непередбачувані деталі	Stable Diffusion для terrain, AI generation