КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (30), 2025
Спеціальний випуск ISSN 2663 – 4023
Special issue
DOI 10.28925/2663-4023.2025.30.972
УДК 004.421:519.17

Трофименко Олена Григорівна
кандидат технічних наук, доцент,
доцент кафедри інформаційних технологій,
Національний університет "Одеська юридична академія", м. Одеса, Україна,
ORCID: 0000-0001-7626-0886
trofymenko@onua.edu.ua

Задерейко Олександр Владиславович
кандидат технічних наук, доцент,
доцент кафедри інформаційних технологій,
Національний університет «Одеська юридична академія», м. Одеса, Україна,
ORCID: 0000-0003-0497-9861
zadereyko@onua.edu.ua

Янковський Олег Георгійович
кандидат технічних наук, доцент,
доцент кафедри телекомунікаційних систем та мереж,
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, м. Київ, Україна,
ORCID: 0000-0001-8041-1843
yanoleg71@gmail.com

Каіров Володимир Олексійович
кандидат технічних наук, доцент,
доцент кафедри програмного забезпечення автоматизованих систем,
Національний університет кораблебудування імені адмірала Макарова, м. Миколаїв, Україна,
ORCID: 0000-0003-0502-7942
volodymyr.kairov@nuos.edu.ua

Морозова Ганна Сергіївна
старший викладач кафедри інформаційних управляючих систем та технологій,
Національний університет кораблебудування імені адмірала Макарова, м. Миколаїв, Україна,
ORCID: 0009-0005-9149-7378
ganna.morozova@nuos.edu.ua

СИСТЕМНИЙ АНАЛІЗ АЛГОРИТМІВ ГЕНЕРАЦІЇ ЛАБІРИНТІВ ДЛЯ ІНТЕРАКТИВНИХ ІГРОВИХ СЕРЕДОВИЩ

Анотація. Лабіринти у відеоіграх слугують не лише інструментом навігації, а й комплексним елементом дизайну, який поєднує технічні, естетичні та ігрові функції. Використання процедурної генерації, інтерактивних елементів та адаптивних систем дозволяє реалізувати інноваційні підходи до побудови віртуального простору. Мета дослідження полягає у формуванні системної класифікації алгоритмів процедурної генерації лабіринтів для застосування в розробці відеоігор, а також у визначенні їх функціональних характеристик, переваг і обмежень з урахуванням технічних і геймдизайнерських вимог. Актуальність теми дослідження зумовлена зростанням ролі процедурного контенту в сучасному геймдеві, що забезпечує підвищену реіграбельність, адаптивність і зниження витрат на ручне проєктування рівнів. У межах дослідження проведено огляд і порівняльний аналіз сучасних алгоритмів генерації лабіринтів, зокрема класичних (DFS, Прима, Крускала, Еллера, Вілсона, Олдоса-Бродера), клітинних автоматів (Rule 4/5, Conway CA, Maze CA, Mazectric, Hybrid CA), шумових функцій (Perlin, Simplex, Worley), фрактальних систем (L-системи, криві Гільберта) та алгоритмів на основі машинного навчання (нейроеволюція, WFC, марковські моделі). Запропоновано класифікацію алгоритмів за типом базової структури (граф, решітка, автомат,

шум, ML-модель), що дозволяє систематизувати підходи до генерації лабіринтів залежно від архітектурних і функціональних ознак. Встановлено, що класичні алгоритми забезпечують високу передбачуваність і продуктивність, тоді як клітинні автомати та гібридні підходи дозволяють створювати складні, органічні або декоративні структури. Наукова новизна полягає у створенні уніфікованої класифікації алгоритмів генерації лабіринтів, яка враховує як структурні, так і геймплейні параметри, що дозволяє обґрунтовано вибирати оптимальні рішення для конкретних ігрових задач. Практичне значення роботи полягає в можливості використання результатів для побудови адаптивних систем генерації рівнів, створення навчальних платформ для вивчення алгоритмів, а також розробки рекомендаційних систем вибору алгоритмів залежно від жанру гри, технічних обмежень та очікуваної складності.

Ключові слова: алгоритми; оптимальність алгоритмів; графи; генерація лабіринтів; розробка ігор; лабіринт; машинне навчання.

ВСТУП

Постановка проблеми. Генерація лабіринтів є актуальною задачею у розробці комп'ютерних ігор, позаяк велика кількість сучасних ігрових проєктів використовують процедурно створені рівні, що дозволяє значно зекономити ресурси на ручному дизайні, забезпечити високу реіграбельність та адаптацію під дії гравця. Лабіринти, як особлива форма таких рівнів, не лише відіграють роль середовища для навігації гравця, а й як механіки випробування, де від структури лабіринту залежить складність, емоційне навантаження та стратегія проходження рівня.

Для генерації лабіринтів використосують різні алгоритми, серед яких найбільш поширеними є: обхід вглиб (Depth-First Search, DFS), Прима, Крускала, Еллера, рекурсивний поділ, вирощування дерева тощо. Суттєво, що зазначені алгоритми генерації лабіринтів у сучасному геймдеві мають принципово відмінні топології, час побудови, рівень складності навігації та вимоги до ресурсів. Наприклад, алгоритм DFS на практиці формує лабіринти з великою кількістю тупиків, що ускладнює орієнтацію, а тому цей алгоритм підходить для жанру головоломок. Алгоритм Прима генерує менш заплутані структури, зручні для початкових рівнів або дитячих ігор. Зі свого боку алгоритм Еллера дозволяє створювати нескінченні стрічкові лабіринти в режимі реального часу, що особливо корисно для жанрів типу endless runner чи roguelike. Через відсутність загальноприйнятих критеріїв коректного вибору алгоритму для конкретної задачі розробники часто використовують одні й ті самі підходи незалежно від контексту, що призводить до неефективності або невідповідності геймдизайну.

Аналіз останніх досліджень і публікацій. Приклади використання тих чи інших алгоритмів для процедурної генерації алгоритмів висвітлені в численних наукових публікаціях. Так, у статті [1] досліджено алгоритм полювання та знищення (Hunt and Kill), де автори дійшли висновку про його потенційно довгий час виконання при великих розмірах лабіринту. У роботі [2] розглянуто три алгоритми генерації лабіринтів (рекурсивного зворотного шляху (Recursive Backtracker), Прима, Крускала) і три алгоритми пошуку шляху (DFS, BFS, A*) в них з позиції пояснення студентам і порівняння показників продуктивності. Автори [3] розглянули дещо інші три алгоритми генерації 2D-лабіринтів (DFS, рекурсивний опис (Recursive description), Прима) з позиції можливого усунення їхніх недоліків. У дослідженні [4] на прикладі ігрового мобільного Android-застосунку для дітей розглянуто відмінності генерації лабіринтів різними алгоритмами (двійкового дерева (Binary Tree Algorithm), повторюваного шаблону, випадкового вибору, Вілсона, Олдоса-Бродера, полювання та знищення, рекурсивного

зворотного шляху) з метою вивчення дітьми (вік 5+) базових понять програмування та заохочення їх до обчислювального мислення. У роботі [5] порівняли складність та середній час проходження невеликих лабіринтів розміром 5x5, 10x10 та 15x15, згенерованих алгоритмами Крускала, Прима, Еллера та зворотного відстеження (Backtracking). Її автори дійшли висновку про те, що лабіринти, створені алгоритмом зворотного відстеження, є найскладнішими і потребують найбільший з-посеред інших час проходження, а лабіринти, створені алгоритмом Прима, є найменш складними з найменшим часом проходження. Дещо інші результати продемонстровано у досліджені [6], в якому порівняно п’ять алгоритмів (Крускала, Олдоса-Бродера, рекурсивного зворотного шляху, Hunt and Kill, двійкового дерева) для створення лабіринтів розмірами 10x10, 50x50, 100x100 та 250x250. Найшвидшим для генерації великих лабіринтів виявився алгоритм двійкового дерев. Стаття [7] порівнює продуктивність семи алгоритмів (Прима, Крускала, DFS, Hunt and Kill, Еллера, Вілсона, Олдоса-Бродера) за двома параметрами – часом генерації лабіринту й просторовою складністю лабіринту. Найшвидшим для створення лабіринтів виявився DFS. Загалом аналіз наявних публікацій свідчить про те, що порівняння алгоритмів генерації лабіринтів має високу практичну та наукову актуальність, особливо в контексті сучасної розробки ігор, інтелектуальних систем і навчального програмування.

Метою статті є системний аналіз алгоритмів процедурної генерації лабіринтів, які застосовуються в розробці відеоігор, з урахуванням їх структурних типів, функціональних властивостей та впливу на геймплей, а також формування класифікації цих алгоритмів для розробки практичних рекомендацій щодо вибору оптимального підходу відповідно до технічних обмежень та дизайнерських вимог.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Важливою є класифікація алгоритмів генерації лабіринтів за типом базової структури, оскільки тип структури (граф, решітка, шум, автомати тощо) визначає архітектурну основу генерації (рис. 1). Така класифікація допомагає систематизувати різні підходи до генерації лабіринтів. Виконане групування за методологією охоплює як класичні алгоритми на основі графів.

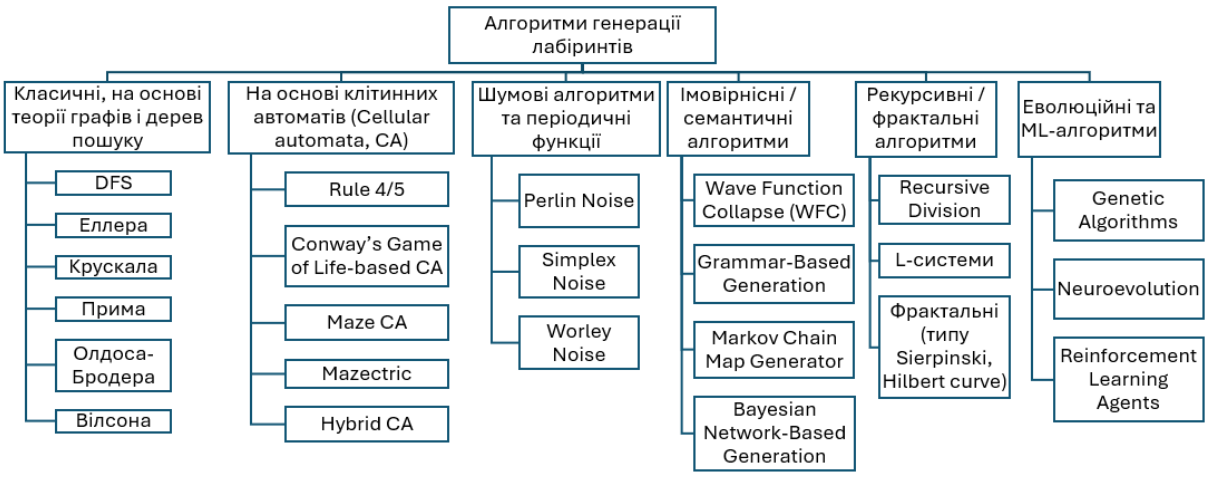

Рис. 1. Класифікація алгоритмів процедурної генерації лабіринтів за типом базової структури

Класичні алгоритми на основі теорії графів та дерев пошуку розглядають простір лабіринту як граф, в якому вузли є кімнатами або клітинами, а ребра – це потенційні проходи. Типовим прикладом такого алгоритму є *обхід графа вглиб* (*Depth-first search, DFS*). Цей простий рекурсивний алгоритм генерує дерева без циклів і створює лабіринти з довгими коридорами. Іншим прикладом є *алгоритм Еллера* (Eller), ітеративний підхід якого здатен рядок за рядком генерувати нескінченні лабіринти у потоковому режимі, але він менш гнучкий у створенні лабіринтів довільної топології. *Алгоритми Прима* (*Prim*) і *Крускала* (*Kruskal*) належать до алгоритмів побудови мінімального остовного дерева, але діють по-різному: Прима починає з однієї клітини і розширюється на сусідів, тоді як Крускала працює з усіма ребрами, з'єднує компоненти, уникаючи циклів. Як результат лабіринти, згенеровані за допомогою Прима, виглядають більш «розтягнутими», тоді як алгоритм Крускала створює рівномірно розподілені проходи. *Алгоритм Олдоса-Бродера* (*Aldous-Broder*) базується на випадковому проході по графу, доки всі клітини не буде відвідано; він створює рівномірно випадкове остовне дерево, але має низьку продуктивність, бо може довго «блукати» вже відвіданими клітинами. *Алгоритм Вілсона* (*Wilson*) виправляє цей недолік за допомогою «loop-erased random walk»: починає з однієї клітини і додає нові шляхи без циклів, що значно пришвидшує генерацію при збереженні рівномірності. Загалом алгоритми Прима і DFS є швидкими й контрольованими, хоча формують менш випадкову топологію, а Крускала, Вілсона і Олдоса-Бродера – більш випадкові, особливо останні два, хоча й менш ефективні для великих карт [8]. Узагальнене порівняння ключових метрик класичних алгоритмів генерації лабіринтів, яке охоплює всі основні підходи, застосовувані в розробці ігор, наведено в табл. 1.

Таблиця 1

Порівняльний аналіз класичних алгоритмів процедурної генерації лабіринтів

Алгоритм	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
DFS	Низький: генерація виконується в лінійному або майже лінійному часі	Низький: складно керувати топологічними характеристиками	Низька: реалізація базується на простих операціях на графах Складність: O(N), де N – кількість клітин	Низький: гарантують зв'язність і відсутність циклів	Забезпечують передбачувану структуру («ідеальні» лабіринти), ефективні для базових ігрових задач або мобільних застосунків
Прима та Крускала		Середній: можливе часткове керування щільністю через вибір ребер	Середня: реалізація на основі теорії графів. Складність: O(N log N)		Добре збалансовані за швидкодією та випадковістю структури
Еллера та Вілсона	Середній	Середній: контроль обмежений структурою рядків або дерев	Середня. Складність їх: O(N) та O(N log N)		Використовуються для генерації нескінченних або потокових лабіринтів

Перевагами усіх алгоритмів цієї категорії є контроль над топологією та генерування лабіринтів з гарантованим єдиним шляхом між будь-якими двома точками. З іншого боку такі алгоритми не завжди доречні для складних структур або адаптивного дизайну.

Алгоритми на основі клітинних автоматів (Cellular Automata, CA) становлять окрему групу методів генерації лабіринтів, які базуються на локальних правилах оновлення стану клітин залежно від їхнього оточення. Вони характерні тим, що не використовують явних структур графів або дерев, а працюють із двовимірною сіткою клітин, де кожна клітина змінює свій стан за заданими правилами [9]. Залежно від правил ці алгоритми дозволяють створювати лабіринти з різним рівнем хаотичності, зв'язності та природності форм (табл. 2). Одним із найпоширеніших є так зване *правило 4/5* (*Rule 4/5* або «*згладжування печери*»). Його суть полягає в тому, що клітинка перетворюється на стіну, якщо має чотири або п'ять сусідів-стін, інакше вона стає прохідною. Застосування кількох ітерацій такого згладжування дозволяє отримати структури, які нагадують печери або природні підземелля. Цей підхід активно використовується в ігровій індустрії, зокрема в генерації карт для roguelike-ігор, де важливо отримати природну, але складну для навігації геометрію простору.

Таблиця 2

Порівняльний аналіз алгоритмів на основі клітинних автоматів процедурної генерації лабіринтів

Алгоритм	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
Rule 4/5	Низький: виконується швидко навіть на великих сітках	Середній: правила дозволяють регулювати щільність стін	Низька: проста логіка переходів	Низький: стабільний результат при фіксованих параметрах	Ефективний для ігор з великою кількістю рівнів; забезпечує повторювані структури
Conway's Game of Life-based CA	Середній: потребує кількох десятків ітерацій для стабілізації	Низький, без додаткових обмежень; можливе хаотичне зростання	Середня: необхідне налаштування стартового патерна та кількості сусідів	Високий: часто утворюються ізольовані або непрохідні ділянки	Придатний для органічних або «хаотичних» рівнів; потребує постпроцесингу
Maze CA	Середній: залежить від кількості ітерацій	Високий: спеціально налаштований для створення прохідних лабіринтів	Середня – потребує ретельного налаштування кількості ітерацій і правил виживання	Середній – можливі зайві замкнені області	Добре збалансований підхід; створює «гладкі» коридори і природні переходи
Mazectric (модифікований CA)	Середній, оптимізований під генерацію на льоту	Високий: зберігає цілісність лабіринту завдяки модифікованим правилам	Середня	Низький: рідко формує некоректні області	Оптимальний для реального часу, дає цікаві «електричні» патерни, які нагадують кореневі системи або нейронні мережі

Алгоритм	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
Hybrid CA (поєднання кількох правил)	Високий: потребує кількох фаз симуляції	Дуже високий: дозволяє контролювати структуру на макро- і мікрорівні	Висока: складна реалізація і вибір правил	Середній: залежить від комбінації фаз	Використовується у сучасних рушіях (напр. Unreal Procedural Tools) для створення складних печер або міст

Іншим прикладом CA є *адаптація правил «Життя» Конвея* (*Conway's Game of Life-based CA*), де модифіковані умови народження і виживання клітин призводять до утворення складних, саморегульованих структур. Клітинка залишається «живою» (прохідною), якщо має 4–5 сусідів, інакше стає «мертвою» (стіною). Хоча такий підхід не завжди дає контрольовані або прохідні лабіринти, він корисний для дослідження динамічних середовищ або для створення еволюційно змінюваних просторів у художньо-орієнтованих системах. Алгоритм *Maze CA* у форматі B3/S123 дозволяє генерувати вузькі, зигзагоподібні проходи без великих порожнин. Його перевагою є здатність формувати справжні лабіринти зі складними маршрутами, придатні для головоломок або інтерфейсних задач, де важлива щільність простору. Особливий інтерес становить варіант CA-алгоритму під назвою *Mazectric*, який використовує правило Life-like автоматів B3/S12345678: B3 – клітина народжується, якщо має 3 сусідів, S12345678 – клітинка виживає, якщо має від 1 до 8 сусідів. Тобто Mazectric відрізняється менш жорсткими правилами виживання клітин, що дозволяє утворювати лабіринти з густішими структурами, більшим числом петлястих шляхів та складною морфологією. Тому Mazectric застосовується для декоративного або естетично спрямованого дизайну рівнів, коли важлива візуальна складність, а не обов'язково оптимальна прохідність. Окремої уваги заслуговують *гібридні алгоритми CA*, які поєднують клітинні автомати з класичними алгоритмами, зокрема DFS, Growing Tree тощо [10]. У таких підходах спочатку створюється базова структура лабіринту, а потім застосовується клітинний автомат для додавання нерегулярності, варіацій або природного шуму. Це дозволяє досягнути балансу між керованістю шляху і візуальною складністю, що корисно для пригодницьких та екшн-ігор із відкритим світом, проте в них не гарантується зв'язність карти, а тому доцільним є додатковий етап з'єднання регіонів. Загалом клітинні автомати забезпечують високу варіативність у генерації просторових структур, хоча й поступаються класичним графовим методам у керованості прохідністю. Їх застосування доцільне там, де пріоритетом є природність форм, художня виразність або непередбачуваність середовища.

Шумові алгоритми та періодичні функції використовують псевдовипадкові функції шуму для створення лабіринтів і середовищ, в яких важлива природна, нерегулярна або псевдореалістична структура простору. Ці методи базуються на математичних функціях [11], що створюють псевдовипадкові або згладжені варіації значень на двовимірному полі, які потім інтерпретуються як стіни чи проходи (табл. 3).

Таблиця 3

Порівняльний аналіз алгоритмів шумових автоматів процедурної генерації лабіринтів

Алгоритм	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
Perlin Noise	Середній: потребує обчислення градієнтів для кожної точки сітки	Середній: легко контролювати частоту, амплітуду, рівень деталізації	Середня: реалізація базується на інтерполяції градієнтів	Середній: без порогової фільтрації, можливі непрохідні області	Класичний алгоритм процедурної генерації; забезпечує плавні зміни, використовується для карт висот і «м'яких» лабіринтів
Simplex Noise	Низький: оптимізований варіант Perlin, працює швидше в багатовимірних просторах	Високий: дозволяє гнучко керувати частотою та ізотропністю шуму	Висока: реалізація складніша, потребує розуміння симплексів	Низький: стабільні результати при фіксованих параметрах	Підходить для реального часу; часто використовується в ігрових рушіях (Unity, Unreal) для процедурних ландшафтів і печер
Worley Noise (Cellular Noise)	Середній або високий, залежно від кількості осередків і відстаней	Високий: дозволяє моделювати структури типу камер, вузлів, тунелів	Висока: потребує складних обчислень відстаней між точками	Середній: можливі артефакти при недостатньому семплінгу	Формує унікальні, «біологічні» або «камерні» структури; часто застосовується для печер, нейронних сіток або систем коридорів
Комбіновані шумові функції (Fractal Brownian Motion, Turbulence)	Високий: потребує багатошарової генерації	Дуже високий: контроль над складністю, деталізацією, масштабом	Висока: об'єднання кількох шумів із різними параметрами	Середній: залежить від кількості шарів і параметрів	Дає реалістичні, багаторівневі карти; використовується у великих ігрових світах для створення природних патернів лабіринтів

Одним із найвідоміших прикладів є *алгоритм Перліна* (*Perlin noise*), який генерує плавно змінюваний шум без різких переходів. Завдяки своїй згладженості цей шум дозволяє створювати м'які, хвилясті лабіринти, схожі на природні ландшафти: печери, скелясті структури або підводні тунелі [12]. Алгоритм підходить для візуально складного середовища, де не вимагається чітко контрольована прохідність, і застосовується, зокрема, в іграх з відкритим світом для генерації рельєфу або динамічних підземель. *Алгоритм Simplex noise,* розроблений як удосконалення методу Перліна, має схожі властивості, але показує кращу продуктивність у великих розмірностях і менше артефактів у діагональних напрямках. Він частіше використовується для 3D-генерації лабіринтів або багаторівневих структур, де важливо уникати періодичних повторів і треба забезпечити природну топологію. Ще одним прикладом є алгоритм *шуму Ворлі* (*Worley noise*), який ґрунтується на відстані до найближчих точок-сусідів і створює клітиноподібні, острівні або сегментовані структури. На відміну від Перліна, цей шум дає чітко розділені області, що корисно для генерації лабіринтів у вигляді островів, поселень або окремих зон з різною функціональністю. Така сегментація підходить для проєктування ігрових мап із чітким поділом на регіони. Загальною специфікою шумових алгоритмів є їхня здатність генерувати природну нерегулярність без потреби зберігати складну інформацію про структуру. Вони не гарантують зв'язність простору і потребують додаткового оброблення для перетворення шуму на прохідну лабіринтну структуру. При цьому їх перевагами є високий рівень варіативності, швидкість генерації та можливість зберігати візуальну цілісність при масштабуванні [13]. Сфери їхнього застосування охоплюють генерацію середовищ у

пригодницьких і симуляційних іграх (наприклад, Minecraft), створення рельєфів, печер, підземель і процедурних мап, де важливі естетика, випадковість і органічна форма, а не точний контроль над маршрутами [4].

Імовірнісні семантичні алгоритми процедурної генерації, на відміну від класичних, не покладаються на детерміновані правила чи фіксовані шаблони, а використовують статистичні моделі для опису залежностей між елементами лабіринту. Їх характерною ознакою є акцент не лише на геометричній цілісності, як у класичних алгоритмах на кшталт DFS або Прима, а й на дотриманні ймовірнісних залежностей, семантичної відповідності або стилістичної узгодженості простору. *Колапс хвильової функції* (*Wave Function Collapse, WFC*) – один із найбільш відомих методик, яка поєднує ймовірнісну логіку із семантичними обмеженнями [14]. Її робота базується на принципі суперпозиції можливих плиток (елементів карти), кількість варіантів яких поступово зменшується до єдиного варіанта згідно з правилами сумісності сусідніх елементів та обчисленнями ентропії Шеннона. Алгоритм забезпечує локальну когерентність і може генерувати лабіринти або рівні, які зберігають стиль, візуальний ритм та логіку референсної структури (табл. 4). В ігровій розробці WFC застосовується для процедурної генерації мап з урахуванням локальних правил, створення квазіреалістичних світів, а також у дизайні архітектурних середовищ або міських структур [15].

Алгоритм граматичної генерації (*Grammar-Based Generation*) описує структуру лабіринту формальними правилами граматики (контекстно-вільної або рекурсивної). Кожне з цих правил визначає, як можуть бути трансформовані частини структури (наприклад, «вузол → вузол + коридор + кімната»), що дозволяє створювати лабіринти з ієрархічною, смислово обґрунтованою побудовою [16]. Такий підхід корисний у наративних іграх та освітніх системах, де простір має відображати логіку подій або завдань [17].

Таблиця 4

Порівняльний аналіз імовірнісних алгоритмів процедурної генерації лабіринтів

Алгоритм	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
Wave Function Collapse (WFC)	Середній: залежить від розміру карти та кількості обмежень	Високий: задається через набір правил сусідства та шаблонів	Висока: потребує ручного проєктування шаблонів, наборів тайлів і правил сумісності	Середній: можливі «тупики» або неможливість знайти валідну конфігурацію	Формує структури з локально узгодженими патернами; ідеальний для стилізованих або архітектурних лабіринтів
Grammar-Based Generation (на основі граматик)	Низький або середній, залежно від глибини розгортання граматичних правил	Високий: можна жорстко задати структурні та топологічні властивості через правила	Середня: потребує побудови граматик і керування стохастичними правилами	Низький: якщо граматика коректна, результати завжди валідні	Дає змогу створювати лабіринти з певними архітектурними мотивами (наприклад, симетричні або модульні структури); підходить для генерації рівнів і dungeon-систем
Markov Chain Map Generator (на основі марківських ланцюгів)	Низький: ефективний у реальному часі	Середній: властивості контролюють через розподіл імовірностей переходів між станами	Середня: потребує побудови матриці переходів і навчання на прикладах	Середній: можливі повторювані або надто однорідні структури	Дає змогу імітувати стиль наявних лабіринтів або карт; часто використовують для генерації рівнів за прикладом (наприклад, у Super Mario AI Framework)

Алгоритм генератора карт на основі ланцюга Маркова (*Markov Chain Map Generator*) базується на використанні марківських ланцюгів для моделювання послідовностей станів,

зокрема для розташування кімнат або вузлів у лабіринті. На відміну від традиційних підходів, тут враховуються статистичні залежності між елементами простору, які можуть бути отримані з навчального набору (наприклад, з наявних карт). Це дозволяє створювати варіативні, але стилістично узгоджені структури. Алгоритм Markov Chain Map Generator ефективно використовується в роґлайк-іграх, симуляторах або навчальних середовищах із потребою збереження впізнаваних шаблонів рівнів для генерації наступного кроку на основі ймовірностей попередніх кроків [18]. *Генерація на основі байєсівських мереж* (*Bayesian Network-Based Generation*) – менш розповсюджений, але потужний метод, що дозволяє враховувати ймовірнісні залежності між компонентами лабіринту на основі умовної ймовірності. Такий підхід забезпечує глобальну когерентність структури та дозволяє адаптувати лабіринт до заданих вимог (наприклад, наявність певної кількості тупиків, зон тощо). Баєсівська мережева генерація може застосовуватися в адаптивних ігрових системах або в генерації сценаріїв навчального моделювання [19]. Спільною рисою всіх вказаних алгоритмів цієї групи є здатність створювати простори, які задовольняють не лише геометричним, а й функціональним, естетичним або геймплейним критеріям. Імовірнісні алгоритми зазвичай забезпечують статистичну різноманітність при збереженні загального стилю, тоді як семантичні – формують змістовно навантажені структури з логічними зв'язками між частинами простору. У практиці ці методи знаходять застосування у відеоіграх (особливо – роґлайках, пригодницьких або платформерах), для генерації віртуальних світів, в архітектурному моделюванні, освітніх симуляторах і навіть у тестуванні штучного інтелекту, де потрібні варіативні, але структурно осмислені середовища.

Таблиця 5

Порівняльний аналіз рекурсивних та фрактальних алгоритмів процедурної генерації лабіринтів

Алгоритм	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
Recursive Division	Низький: алгоритм працює швидко навіть на великих сітках	Високий: можна керувати симетрією, шириною коридорів, частотою розгалужень	Низька: реалізація базується на простій рекурсії	Низький: алгоритм детермінований і майже не генерує помилкових структур	Створює лабіринти з чіткою ієрархічною структурою та прямолінійними проходами; часто використовується у класичних dungeon-генераторах та навчальних симуляціях
Lindenmayer systems (L-системи)	Середній або високий – залежить від глибини рекурсії та кількості правил	Високий – дозволяє керувати формою, симетрією та щільністю шляхом зміни граматики	Висока – потребує проєктування граматичних правил та ітераційного розгортання	Середній: при некоректних правилах можливі самоперетини або ізольовані області	Використовується для генерації природних або деревоподібних структур; може формувати лабіринти з біоморфною геометрією (наприклад, у стилі «живих печер» чи «кореневих» структур)
Фрактальні алгоритми (наприклад, на основі плазмових фракталів, midpoint displacement, diamond-square)	Середній: зростає з розміром карти та кількістю ітерацій	Середній: контроль обмежений параметрами roughness і recursion depth	Середня: потребує розуміння фрактальної геометрії та процедур згладжування	Середній: можливі некоректні або непридатні для проходження топології	Формують природні, органічні лабіринти із самоподібною структурою; підходять для ландшафтів, печер, підземель або біомів із складною топографією

Рекурсивні та фрактальні алгоритми процедурної генерації лабіринтів ґрунтуються на самоподібних, повторюваних структурах або ієрархічному поділі простору. Вони займають проміжну позицію між класичними детермінованими й стохастичними підходами. Ці алгоритми забезпечують високий рівень впорядкованості та симетрії, що робить їх особливо корисними у створенні візуально виразних та математично обґрунтованих лабіринтів (табл. 5).

Одним із найпоширеніших представників цього класу є алгоритм *Recursive Division*, який реалізується шляхом рекурсивного поділу простору на підобласті з подальшим прорізанням проходів. Цей підхід зумовлює його здатність створювати лабіринти з чіткою ієрархічною структурою та високою навігаційною ефективністю [20]. Загалом Recursive Division вирізняється низькою обчислювальною складністю та зручною реалізацією для двовимірних ігор. Інший потужний інструмент – це *L-системи (Lindenmayer systems)*, які спочатку були запропоновані для моделювання росту рослин, проте згодом були адаптовані для процедурної генерації, зокрема лабіринтів та ігрових рівнів. Суть цього підходу полягає у використанні формальних граматик, які ітеративно розгортають просту початкову структуру в складну геометрію [21]. У контексті лабіринтів це дозволяє створювати топологічно послідовні та самоподібні маршрути, які можуть бути використані як в архітектурних середовищах, так і в фантастичних світах через їхню придатність для генерації навколишнього середовища з високим ступенем повторюваності, а також легкість керування параметрами складності. Окремої уваги заслуговують *фрактальні алгоритми*, які базуються на побудові простору за відомими математичними кривими, такими як крива Гільберта (Hilbert curve) або трикутник Серпінського (Sierpinski triangle). Ці фрактальні структури дозволяють отримувати лабіринти з високим ступенем самоподібності, які можуть бути ефективно використані для побудови ігрових світів із непрямолінійною навігацією. Криві заповнення площини, як-от крива Гільберта, особливо цікаві в контексті генерації шляхів, що максимально покривають доступний простір із мінімальним перетином маршрутів [2]. Фрактальні лабіринти демонструють високу стабільність у поведінці користувача та потенціал до адаптивної генерації у відповідь на ігрові дії. Загалом рекурсивні та фрактальні алгоритми доречно використовувати в ігрових середовищах, коли потрібні лабіринти з високою структурною цілісністю, передбачуваною складністю та естетичною симетрією. Їх застосування не лише забезпечує варіативність рівнів, а й дозволяє легко масштабувати середовище з урахуванням різних вимог до дизайну гри або симуляції.

Еволюційні та ML-алгоритми створюють лабіринти, навчаючись або оптимізуючи функцію придатності. Еволюційні методи орієнтовані на пряме оптимізаційне формулювання проблеми: лабіринт кодується в генотипі (бінарний масив стін/прохідних комірок, перерахування коридорів або параметричні подання), для нього визначається фенотип (реальна карта), а через функцію пристосованості (fitness function) оцінюється придатність згенерованого рівня за набором критеріїв, наприклад: довжина, кількість розгалужень, ступінь симетрії, ігрова складність для агента. На практиці генетичні алгоритми є ефективними для генерації різноманітних лабіринтів та оптимізації ігрової складності [22]. Еволюційні алгоритми дають дизайн-інженеру високий рівень контролю через явне задання функції пристосованості і дозволяють включати багатокритеріальну оптимізацію та техніки на кшталт novelty search або MAP-Elites для забезпечення різноманітності генерацій [23]. Дещо протилежний підхід мають методи машинного навчання (ML, Machine Learning), які в останні роки стали потужним інструментом процедурної генерації контенту (PCG, Procedural Content Generation). На відміну від еволюційних методів, які використовують явну оптимізацію за заданою

фітнес-функцією, моделі машинного навчання навчаються відтворювати або узагальнювати структурні закономірності на основі навчальних прикладів [24]. До таких підходів належать *автокодери та варіаційні автокодери* (VAE, Variational Autoencoders), *мережі довготривалої короткочасної пам'яті* (LSTM, Long Short-Term Memory networks) для послідовних представлень, а також *генеративні змагальні мережі* (GAN, Generative Adversarial Networks), адаптовані для побудови рівнів і карт [17]. Машинне навчання дозволяє створювати узгоджені, стилістично єдині варіанти лабіринтів із високою швидкістю після етапу навчання. Проте цей підхід потребує ретельно підібраного навчального набору даних і може продукувати некоректні структури (наприклад, без виходу або з ізольованими зонами) без додаткового етапу перевірки обмежень. У межах ML-парадигми виокремився напрям навчання з підкріпленням (RL, Reinforcement Learning), який застосовується для з'явився окремий напрям для PCG. У RL агент навчається послідовно будувати рівень або лабіринт, отримуючи винагороду залежно від досягнення певних критеріїв, як-от: прохідність, ігрова збалансованість чи відповідність очікуваній поведінці тестових агентів. Такий підхід відкриває можливість формулювати більш вибагливі вимоги, наприклад, «згенерувати лабіринт, який буде складним для автоматизованого пошукового алгоритму A* (A-star), але водночас прийнятним для людини-гравця». Водночас RL-методи потребують значних обчислювальних ресурсів і ретельного проєктування функції винагороди, щоб уникнути ситуацій, коли агент навчається неадекватним або епізодично оптимальним стратегіям [25]. Отже, порівняльний аналіз специфіки показує, що еволюційні алгоритми відрізняє детермінованість управління: інженер явно формує критерії якості і може гарантувати певні властивості (наприклад, наявність шляху, мінімальна довжина рішення, певна складність), тоді як ML-методи краще відтворюють статистичні властивості й здатні генерувати естетично узгоджені варіанти без ручного налаштування для кожного критерію [26]. Порівняння основних ML-підходів до генерації лабіринтів наведено в табл. 6.

Таблиця 6

Порівняльна характеристика ML-підходів до процедурної генерації лабіринтів

Підхід	**Концептуальна основа**	**Точність відтворення структурних закономір-ностей**	**Контроль складності та властивостей лабіринту**	**Потреби у навчальних даних**	**Стабільність результату та узгодженість генерацій**	**Типові сфери застосу-вання**
Варіаційні автоко-дери (VAE)	Статистичне відображення структури лабіринту у латентному просторі з можливістю інтерполяції між варіантами	Висока точність відтворення глобальних структур; локальні деталі можуть бути спрощеними	Обмежений контроль: зміна латентних змінних дає загальні варіації без прямої семантичної інтерпретації	Високі: потребують великої кількості збалансованих прикладів для навчання	Висока при стабільному латентному просторі, але можлива деградація якості при перенавчанні	Генерація нових карт на основі наявних прикладів, дослідження простору форм лабіринтів
Мережі довго-тривалої коротко-часної пам'яті (LSTM)	Моделювання лабіринту як послідовності клітин або рішень під час його побудови	Середня: добре відтворює послідовності руху, але схильна до накопичення помилок у довгих шляхах	Високий контроль над локальними структурами; обмежений над глобальною топологією	Помірні: достатньо набору прикладів з типових послідов-ностей побудови	Середня: стабільність залежить від довжини послідовностей і налаштувань станів пам'яті	Генерація покрокових лабіринтів, навчання агентів на основі історій дій

Підхід	**Концептуальна основа**	**Точність відтворення структурних закономір-ностей**	**Контроль складності та властивостей лабіринту**	**Потреби у навчальних даних**	**Стабільність результату та узгодженість генерацій**	**Типові сфери застосу-вання**
Генера-тивні змагальні мережі (GAN)	Змагання між генератором і дискримінатором для створення реалістичних конфігурацій лабіринтів	Висока: здатні відтворювати як глобальні, так і локальні закономірності форми	Низький, без спеціального керування: складно гарантувати контроль над складністю або прохідністю	Високі: потрібен великий та різноманітний набір даних для уникнення «залипання»	Висока після збалансованого навчання; проте можливі коливання якості між епохами	Генерація рівнів у комп'ютерних іграх, дизайн карт з естетичною варіативністю
Навчання з підкріп-ленням (RL)	Агент послідовно будує лабіринт, отримуючи винагороду за досягнення мети або дотримання властивостей	Висока для функціональних критеріїв (прохідність, баланс, складність)	Високий: можливо явно формулювати вимоги через функцію винагороди	Низькі: достатньо симульованого середовища без потреби навчального набору прикладів	Середня: результат залежить від стабільності політики агента та налаштування винагороди	Автоматична генерація рівнів у реальному часі, адаптивні ігрові середовища, дослідження роботизованої навігації

Еволюційні підходи легко інтегрувати з різними заходами валідації (симуляція гравця, формальні тести), але вони часто більш ресурсомісткі в контексті часу виконання при великому просторі пошуку і потребують ретельного проєктування fitness-функції. ML-моделі GAN та VAE забезпечують швидке генерування після тренування і добре масштабуються для створення великих і різноманітних лабіринтів, однак їхня поведінка при екстраполяції за межі тренувального набору менш передбачувана і часто потребує комбінованих перевірок на коректність [27]. З іншого боку, аналіз переваг і недоліків вказує на практичні обмеження еволюційних алгоритмів. По-перше, вони вимагають визначення адекватних fitness-метрик, що само по собі є складним дослідницьким завданням і може призводити до несподіваних «ігрових обхідних шляхів». По-друге, вони масштабуються гірше при високій розмірності лабіринту. Методи ML потребують різноманітні, стилістично узгоджені дані для навчання і ризикують продукувати семантично неконсистентні карти; крім того, інтерпретованість отриманих моделей обмежена [28]. Методи RL вирізняються нестабільністю навчання та чутливістю до формулювання винагороди, що ускладнює реплікацію результатів. Недавні дослідження та бенчмарки підкреслюють необхідність розробки загальних метрик якості та еталонних наборів задач для коректного порівняння методів.

Як видно з табл.6, кожен підхід має власну специфіку та баланс між точністю, контролем і потребами у даних. Варіаційні автокодери демонструють найкращу узгодженість у структурній генерації, але обмежені в точному управлінні параметрами складності. LSTM-мережі забезпечують більшу керованість локальними елементами, однак менш ефективні у підтриманні глобальної цілісності. Підхід на основі генеративних змагальних мереж має найвищий потенціал для відтворення складних патернів, але вимагає великих обсягів навчальних даних і обчислювальних ресурсів. Натомість методи навчання з підкріпленням надають максимальний контроль за властивостями лабіринту та придатні для динамічних або адаптивних сценаріїв, хоча й характеризуються високою складністю налаштування.

Загалом класичні методи (DFS, Прима, Крускала) залишаються найшвидшими, але обмеженими у варіативності; вони залишаються оптимальними, коли критичними є

продуктивність, передбачуваність і простота впровадження, особливо в освітніх, симуляційних або низькорівневих ігрових середовищах. Клітинні автомати й шумові підходи забезпечують природну морфологію, проте вимагають післяобробки для забезпечення ігрової прохідності. Імовірнісні алгоритми надають гнучкість і структурність, але мають значну обчислювальну складність. Еволюційні та ML-підходи демонструють найбільший потенціал у процедурній генерації контенту нового покоління і найвищий рівень контрольованості створюваних лабіринтів з точно заданими параметрами (довжина оптимального шляху, кількість тупиків); вони дозволяють не просто створювати лабіринти, а адаптувати їх до гравця, стиля гри та контексту геймплею. У табл. 7 узагальнено основні характеристики сучасних підходів до процедурної генерації лабіринтів: від класичних алгоритмів (DFS, Прима, Крускала, Еллера, Вілсона) до еволюційних та ML.

Вибір алгоритму процедурної генерації лабіринтів тісно пов’язаний із жанровою специфікою гри. Класичні графові методи забезпечують контроль і передбачуваність; клітинні та шумові – природність і атмосферність; WFC і граматичні – семантичну узгодженість; еволюційні та ML-підходи – гнучкість і адаптивність до ігрових цілей. Порівняння алгоритмів процедурної генерації лабіринтів за типами ігор (табл. 8) відображає взаємозв’язок між класом алгоритму, його придатністю до різних жанрових структур і особливостями отримуваного геймплейного простору. Аналіз показав, що ефективність алгоритму визначається не лише технічними характеристиками (швидкодією чи контролем властивостей лабіринту), а й семантичною відповідністю ігровому жанру, зокрема стилем рівнів, бажаною складністю та варіативністю простору.

Таблиця 7

Порівняльний узагальнений аналіз алгоритмів процедурної генерації лабіринтів

Алгоритм / Підхід	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
Класичні алгоритми (DFS, Прима, Крускала, Еллера, Вілсона)	Низький: генерація виконується в лінійному або майже лінійному часі	Обмежений: контроль щільності та топології складний	Низька: реалізація базується на простих графових операціях	Низький: гарантують зв’язність і відсутність циклів	Забезпечують передбачувану структуру («ідеальні» лабіринти), ефективні для базових ігрових задач або мобільних застосунків
Алгоритми на основі клітинних автоматів (Conway’s Game of Life, Rule 4/5, Maze CA)	Середній: залежить від кількості ітерацій і розміру сітки	Високий: можна задавати щільність, коридорність, «біоморфність»	Середня: потребує налаштування правил переходу	Середній: можливі ізольовані області без виходу	Дають природні, органічні форми лабіринтів; часто застосовуються у survival-іграх, open-world середовищах
Шумові алгоритми та періодичні функції (Perlin Noise, Simplex Noise, Worley Noise)	Середній: залежить від складності функції та фільтрації	Високий: можна регулювати періодичність, текстуру, топологію	Висока: потребує знань про генерацію процедурних карт	Середній: можливі невалідні топології без додаткової обробки	Використовуються для створення природних патернів (печери, тунелі), забезпечують плавність переходів і візуальне розмаїття
Імовірнісні / семантичні	Середній або високий,	Високий: можна описувати	Висока: потребує	Середній: можливі dead-	Дають контроль над стилем і структурою;

Алгоритм / Підхід	**Час генерації**	**Контроль властивостей лабіринту**	**Складність реалізації**	**Ризик помилок (некоректних лабіринтів)**	**Характерні особливості / коментар**
алгоритми (Wave Function Collapse, Grammar-Based Generation, Markov Chains)	залежно від складності обмежень	правила сумісності блоків, шаблони структури	ручного проєктування шаблонів і семантичних правил	ends або неможливість знайти валідне рішення	активно використовуються для рівнів у жанрах puzzle та RPG
Рекурсивні та фрактальні алгоритми (Recursive Division, L-системи, криві Гільберта, Сєрпінського)	Середній: залежить від кількості рекурсивних кроків	Середній: контроль здійснюється через параметри рекурсії	Середня: потребує математичного моделювання фрактальної структури	Низький: побудова детермінована, помилки рідкісні	Забезпечують естетичну симетрію та предиктивність; часто використовуються в puzzle-рівнях або симуляціях
Еволюційні та ML-алгоритми (Genetic Algorithms, Neuroevolution, Reinforcement Learning)	Високий під час навчання, низький під час використання	Дуже високий: можна задавати цільові метрики складності, стилю, прохідності	Дуже висока: вимагає навчального набору та налаштування моделі	Середній: можливі некоректні або неігрові конфігурації без перевірок	Дають найвищу адаптивність; дозволяють створювати лабіринти, що підлаштовуються під гравця або динаміку геймплею

Таблиця 8

Порівняльна таблиця алгоритмів за типами ігор

Алгоритм	**Roguelike**	**Adventure**	**Puzzle / Logic**	**Коментар**
DFS (Backtracking)	Хороший	Придатний	Підходить	Дає ідеальні лабіринти; контрольований рівень складності
Прима	Гарний вибір	Можливий	Обмежено	Більш «хаотичний», добре для повторюваності
Крускала	Підходить	Придатний	Підходить	Дозволяє більше контролю через ваги
Вілсона	Повільний	Малоефективний	Підходить	Рівномірна генерація; складний контроль
Олдоса-Бродера	Реалістичний	Повільний	Не рекомендовано	Випадковий вигляд, але дорогий по ресурсах
Еллера	Ідеально для скролінгу	Обмежено	Обмежено	Ідеальний для side-scrolling dungeon crawler
CA	Атмосферний	Природний	Не підходить	Формує «печери», але не завжди прохідні
Perlin / Simplex Noise	Візуально цікаво	Можливо	Не рекомендовано	Потрібна післяобробка для прохідності
WFC	Можливий	Відмінний	Використовується	Добре для сюжетних / семантичних просторів
WFC + Grammar	Рідко	Ідеально	Придатний	Поєднання структури та семантики
GA	Добре масштабується	Можливий	Дуже добре	Працює на оптимізацію за цілями
RL	Підлаштовується	Потенційно	Обмежено	Динамічне коригування складності

Тобто розробка процедурної генерації лабіринтів для ігор вимагає не лише технічного розуміння алгоритмів, а й усвідомлення їхньої жанрової відповідності. Вибір конкретного підходу має спиратися на структурні, динамічні та естетичні вимоги ігрового світу. Це означає оптимальне поєднання формальних характеристик алгоритму (контрольованість, обчислювальна складність, відтворюваність результатів) з художньо-ігровими аспектами, такими як атмосфера, нелінійність або семантична логіка простору.

Для *roguelike ігор*, в яких провідну роль відіграють випадковість, повторюваність проходжень і тактична варіативність, найбільш доцільними є алгоритми на основі графових структур, зокрема DFS, Прима або Крускала. Їхня здатність формувати зв'язні і при цьому непередбачувані лабіринти забезпечує баланс між складністю та ігровою справедливістю. У класичних прикладах, як-от NetHack або Dungeon Crawl Stone Soup, ці алгоритми використовуються для створення рівнів, де кожне проходження має унікальну конфігурацію, але при цьому зберігається логічна структура маршруту.

Для *пригодницьких ігор* (*adventure*) доцільно застосовувати семантичні або граматичні підходи, наприклад, Wave Function Collapse (WFC) або Grammar-Based Generation, оскільки вони дозволяють будувати локації, які відповідають наративним вимогам. Такі ігрові системи, як у «Legend of Zelda: Breath of the Wild» або «No Man's Sky», демонструють ефективність поєднання процедурної структури з попередньо визначеними правилами дизайну. Алгоритми цього типу забезпечують не просто топологічну різноманітність, а й семантичну узгодженість простору, що особливо важливо для сюжетних ігор із відкритим світом.

У *puzzle* та *logic-іграх*, де пріоритетом є контрольована складність і наявність єдиного або передбачуваного розв'язку, ефективно використовуються DFS, Крускала або рекурсивні методи поділу (Recursive Division). Наприклад, у навчальних або мобільних логічних іграх (The Witness, Monument Valley) подібні алгоритми дозволяють точніше регулювати геометрію та послідовність проходження. Їхня детермінованість і передбачуваність створює можливість адаптації рівня складності до когнітивного навантаження гравця.

У *survival* або *rogue-lite* жанрах, де важливими є природність середовища, його варіативність і атмосфера, перевагу варто навати клітинним автоматам (Cellular Automata) та шумовим алгоритмам (Perlin, Simplex, Worley Noise). Вони дають змогу створювати печери, тунелі та біоморфні структури, які імітують природні форми, зберігаючи при цьому процедурну непередбачуваність. Такі підходи використовуються у Terraria, Spelunky, Don't Starve, де структура карти не лише функціональна, а й естетично узгоджена з тематикою гри.

Еволюційні алгоритми та ML-підходи відкривають нові можливості для адаптивної генерації ігрового простору. У динамічних іграх або симуляторах, де середовище має змінюватися залежно від дій гравця, такі алгоритми дозволяють оптимізувати лабіринти за конкретними критеріями: складністю проходження, ресурсною насиченістю або поведінкою супротивників. Дослідження [22 Khalifa] показують, що поєднання генетичних стратегій із нейронними моделями дозволяє досягати високої керованості структури рівнів при збереженні стохастичної варіативності.

Для *ігор із відкритим світом або sandbox-підходом*, де необхідна поєднаність численних регіонів із різними топологічними характеристиками, ефективним є використання гібридних схем, тобто поєднання шумових, фрактальних та граматичних методів. Такі системи дозволяють створювати багаторівневі світи, в яких локальні патерни поєднані в масштабну, узгоджену структуру. Яскравим прикладом є Minecraft,

де поєднання Perlin Noise з імовірнісними правилами формує природний, але структурно контрольований світ.

Отже, вибір алгоритму процедурної генерації лабіринтів повинен базуватися на поєднанні технічних характеристик і жанрової семантики гри, щоб забезпечити відповідну ігрову динаміку, баланс складності та продуктивність. Класичні графові методи оптимальні для контрольованих ігрових просторів, клітинні та шумові – для природних і нелінійних середовищ, а еволюційні та машинно-навчальні – для адаптивних систем, здатних підлаштовуватися під гравця чи ігрові сценарії. Це вказує на поступовий перехід у сучасній ігровій індустрії від детермінованих алгоритмів до когнітивно-керованих систем генерації контенту, де лабіринт стає не лише топологічною структурою, а інтерактивною моделлю ігрового досвіду.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

За результатами проведеного аналізу класифіковано алгоритми процедурної генерації лабіринтів за типом базової структури, що дозволило систематизувати підходи до побудови віртуального простору в контексті розробки відеоігор. Встановлено, що вибір алгоритму має суттєвий вплив на топологію лабіринту, складність навігації, естетичне сприйняття рівня та загальну динаміку геймплею. Класичні алгоритми на основі графів забезпечують передбачувану структуру та високу продуктивність, тоді як алгоритми на основі клітинних автоматів дозволяють створювати органічні, хаотичні або декоративні форми з гнучким контролем щільності та прохідності. Запропонована класифікація враховує не лише архітектурну основу алгоритмів, а й топологічні властивості згенерованих лабіринтів (ідеальність, циклічність, прохідність), що є критично важливим для адаптації алгоритмів до жанрових і геймплейних вимог.

Проведено порівняння класичних алгоритмів (DFS, Прима, Крускала, Еллера, Вілсона, Олдоса-Бродера) за критеріями часу генерації, складності реалізації, контролю топології та ризику помилок. Встановлено, що: DFS забезпечує найшвидшу генерацію, але має низький рівень контролю над структурою; Прима та Крускала демонструють баланс між випадковістю та керованістю; Вілсона та Олдоса-Бродера – найбільш випадкові, але менш ефективні для великих карт. Алгоритми на основі клітинних автоматів (Rule 4/5, Maze CA, Mazectric, Hybrid CA) є ефективними для створення органічних, складних або декоративних лабіринтів, особливо в жанрах roguelike, dungeon crawler та puzzle. Визначено, що гібридні та ML-орієнтовані підходи до генерації алгоритмів мають високий потенціал для адаптивного дизайну, але потребують значних обчислювальних ресурсів і складної інтеграції.

Запропонована класифікація та порівняльні таблиці можуть слугувати основою для створення рекомендаційних систем вибору алгоритму залежно від технічних і дизайнерських параметрів проєкту. Визначено, що відсутність єдиних критеріїв вибору алгоритму для конкретного ігрового контексту є актуальною проблемою, яка потребує подальшого дослідження. Результати дослідження можуть бути використані для: розробки адаптивних систем генерації рівнів у відеоіграх; створення навчальних платформ для вивчення алгоритмів; оптимізації генерації контенту в мобільних іграх з обмеженими ресурсами.

Перспективи подальших досліджень можуть стосуватися розробки метрик для автоматизованої оцінки якості згенерованих лабіринтів (наприклад, рівень складності, час проходження, когнітивне навантаження) та інтеграції алгоритмів генерації

лабіринтів з системами машинного навчання для адаптації структури рівня до стилю гри конкретного користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Marchuk G.V., Liubchenko D.V. Generation of mazes using the hunt-and-kill algorithm. *Scientific notes of Taurida National V.I. Vernadsky University*. *Series: Technical Sciences*. 2024. Issue 35 (74), № 3. P. 130-135. DOI: https://doi.org/10.32782/2663-5941/2024.3.1/20
2. Binayak B. Interactive Maze Generation and Pathfinding Algorithms: An Educational Visualization Platform for Algorithmic Analysis. *Zenodo*. 2025. DOI: https://doi.org/10.5281/zenodo.15824242
3. Vijaya J., Gopu A., Vinayak K., Singh J., Malhotra K. Analysis of Maze Generation Algorithms. *5th IEEE International Conference on Innovative Trends in Information Technology (ICITIIT'24)*. Kottayam, India, 15-16 March 2024. P. 1-6. DOI: https://doi.org/10.1109/ICITIIT61487.2024.10580178
4. Čarapina M., Staničić O., Dodig I., Cafuta D. A Comparative Study of Maze Generation Algorithms in a Game-Based Mobile Learning Application for Learning Basic Programming Concepts. *Algorithms*. 2024. Vol. 17(9). P. 404. DOI: https://doi.org/10.3390/a17090404
5. Cahyakusuma I., Istiono, W. Algorithmic diversity in maze generation: Comparative study of Backtracking, Kruskal's, Prim's, and Eller's algorithms. *International Journal of Intelligent Systems and Applications in Engineering*. 2024. Vol. 12(3). P. 1281–1287. URL: https://ijisae.org/index.php/IJISAE/article/view/5518
6. Марчук Г. В., Граф М. С., Левківський В. Л., Венгловська Ю. В. Аналіз та порівняння існуючих методів генерації лабіринтів в комп'ютерних іграх. *Вісник Херсонського національного технічного університету*. 2024. № 3(90). С. 228-237. DOI**:** https://doi.org/10.35546/kntu2078-4481.2024.3.29
7. Mane D., Harne R., Pol T., Asthagi R., Shine S., Zope B. An Extensive Comparative Analysis on Different Maze Generation Algorithms. *International Journal of Intelligent Systems and Applications in Engineering*. 2023. Vol. 12(2). P. 37-47. URL: https://ijisae.org/index.php/IJISAE/article/view/3557
8. Piao J., Hu X., Zhou Q. Maze and navigation algorithms in game development. *Applied and Computational Engineering*. 2024. Vol. 79. P. 11-19. DOI**:** https://doi.org/10.54254/2755-2721/79/20241081.
9. Malem F. Wall Object Design in Maze Game Using Cellular Automata Algorithm. *Journal of Computer Engineering: Progress, Application and Technology*. 2023. Vol. 2(1). P. 27-35. DOI**:** https://doi.org/10.25124/cepat.v2i01.5781.
10. Yang K., Lin S., Dai Yu., Li W. Optimization and comparative analysis of maze generation algorithm hybrid. *Applied and Computational Engineering*. 2024. Vol. 79. P. 20-33. DOI**:** https://doi.org/10.54254/2755-2721/79/20241082.
11. Трофименко О. Г., Задерейко О. В., Баландіна Н. М., Толокнов А. А., Гусельніков І. М. Векторна алгебра для розробки ігор. *Кібербезпека: освіта, наука, техніка*. 2024. № 2(26). С. 71-80. DOI: https://doi.org/10.28925/2663-4023.2024.26.626.
12. Xu J., Morris J. Procedural generation in 2D metroidvania game with answer set programming and perlin noise. *International Journal of Artificial Intelligence & Applications*. 2023. Vol. 14. P. 21-35. DOI**:** https://doi.org/10.5121/ijaia.2023.14302.
13. Трофименко О. Г., Задерейко О. В., Логінова Н. І., Шевченко О. В. Інструменти штучного інтелекту в розробці графічних елементів, 3D-моделей та інших компонентів відеоігор. *Інформатика та математичні методи в моделюванні*. 2025. Т.15, № 2. С. 276-287. DOI: https://doi.org/10.15276/imms.v15.no2.276
14. Alaka Sh., Bidarra R. Hierarchical Semantic Wave Function Collapse. *Proceedings of the 18th International Conference on the Foundations of Digital Games (FDG '23)*. 2023. Vol. 68. P. 1-10. DOI**:** https://doi.org/10.1145/3582437.3587209.
15. Linden R., Lopes R., Bidarra R. Procedural Generation of Dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*. 2014. Vol. 6, Issue 1. P. 78-89. DOI**:** https://doi.org/10.1109/TCIAIG.2013.2290371.
16. Tanaka T., Simo-Serra E. Grammar-based Game Description Generation using Large Language Models. *arXiv.* DOI**:** https://doi.org/10.48550/.2407.17404
17. Shaker N., Liapis A., Togelius Ju., Lopes R., Bidarra R. Constructive generation methods for dungeons and levels. *Computational Synthesis and Creative Systems (CSACS'16)*, 2016. P. 31-55. DOI**:** https://doi.org/10.1007/978-3-319-42716-4_3.
18. Markov Chains for Procedural Buildings. URL: https://nickmcd.me/2019/10/30/markov-chains-for-procedural-buildings/

19. Dou H., Du J., Jiang X., Li H., Yao W., Deng Y. Image-to-Image Bayesian Flow Networks With Structurally Informative Priors. *IEEE Transactions on Image Processing*. 2025. Vol. 34. P. 4968-4982. DOI: https://doi.org/10.1109/TIP.2025.3592546.
20. Setiadharma E., Husniah L., Kholimi A. Algoritma Maze Generator Recursive Backtracking Untuk Membuat Prosedural Labirin Pada Game Petualangan Labirin 3D. *Jurnal Repositor*. 2024. Vol. 2(3). P. 373-384. DOI: https://doi.org/10.22219/repositor.v2i3.30509
21. Duffy Ch., Hillis S., Khan U., McQuillan I., Shan S. Inductive inference of lindenmayer systems: algorithms and computational complexity. *Natural Computing*. 2025. P. 1-11. DOI: https://doi.org/10.1007/s11047-025-10024-x.
22. Yuichi N., Akinori I., Norihiko O. A Genetic Algorithm for the Picture Maze Generation Problem. *Computers & Operations Research*. 2019. Vol. 115. P. 104860. DOI: https://doi.org/10.1016/j.cor.2019.104860.
23. Krishnaa K., Kaluri J., Konjeti S., Aravind T., Gurusamy J. A Genetic Algorithm Framework to Solve Two-Dimensional Maze Problem. *Proceedings of 3rd International Conference on Machine Learning, Advances in Computing, Renewable Energy and Communication.* 2022, 18 September, P. 277-284. DOI: https://doi.org/10.1007/978-981-19-2828-4_27
24. Трофименко О.Г., Дика А.І., Лобода Ю.В., Толокнов А.А., Бондаренко М.Є. Штучний інтелект у розробці ігор. Кібербезпека*: освіта, наука, техніка*. 2025. № 3(27). С. 109-119. DOI: https://doi.org/10.28925/2663-4023.2025.27.701.
25. Khalifa A., Bontrager P., Earle S., Togelius J. PCGRL: Procedural Content Generation via Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* 2020. Vol. 16(1). P. 95-101. DOI: https://doi.org/10.1609/aiide.v16i1.7416
26. Трофименко О.Г., Дика А.І., Задерейко О.В., Шевченко О.В. Сфери застосування штучного інтелекту в розробці та тестуванні ігор. *Кібербезпека: освіта, наука, техніка*. 2025. № 1(29). С. 41-59. DOI: https://doi.org/10.28925/2663-4023.2025.27.701
27. Summerville A. et al. Procedural content generation via machine learning (PCGML). *arXiv*. Vol. 1702.00539. DOI: https://doi.org/10.48550/arXiv.1702.00539
28. Alaguna C., Gomez Jo. Maze benchmark for testing evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '18).* 2018. P. 1321-1328. DOI: https://doi.org/10.1145/3205651.3208285

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (30), 2025
Спеціальний випуск ISSN 2663 – 4023
Special issue

Olena G. Trofymenko
PhD, Associate Professor,
Associate Professor at the Department of Information Technologies
National University "Odessa Law Academy", Odesa, Ukraine,
ORCID: 0000-0001-7626-0886
trofymenko@onua.edu.ua

Olexander V. Zadereyko
PhD, Associate Professor,
Associate Professor at the Department of Information Technologies
of the National University «Odessa Law Academy», Odesa, Ukraine,
ORCID: 0000-0003-0497-9861
zadereyko@onua.edu.ua

Oleg G. Iankovskii
PhD, Associate Professor,
Associate Professor at the Department Telecommunications Systems and Networks
Kruty Heroes Military Institute of Telecommunications and Information Technology, Kyiv, Ukraine,
ORCID: 0000-0001-8041-1843
yanoleg71@gmail.com

Volodymyr O. Kairov
PhD, Associate Professor,
Associate Professor at the Department of Software for Automated Systems,
Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine,
ORCID: 0000-0003-0502-7942
volodymyr.kairov@nuos.edu.ua

Hanna S. Morozova
Senior Lecturer at the Department of Information Management Systems and Technologies,
Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine,
ORCID: 0009-0005-9149-7378
ganna.morozova@nuos.edu.ua

SYSTEMATIC ANALYSIS OF MAZE GENERATION ALGORITHMS IN INTERACTIVE GAME ENVIRONMENTS

Abstract. Labyrinths in video games serve not only as navigation tools but also as complex design elements that combine technical, aesthetic, and gameplay functions. The use of procedural generation, interactive elements, and adaptive systems enables innovative approaches to construct virtual spaces. The purpose of this study is to develop a systematic classification of algorithms for the procedural generation of labyrinths for use in video game development and to determine their functional characteristics, advantages, and limitations, considering technical and game design requirements. The relevance of the research lies in the growing importance of procedural content in modern game development, which enhances replayability, adaptability, and reduces the cost of manual level design. This study presents a review and comparative analysis of modern maze generation algorithms, including classical approaches (DFS, Prim's, Kruskal's, Eller's, Wilson's, Aldous-Broder), cellular automata (Rule 4/5, Conway's Game of Life, Maze CA, Mazectric, Hybrid CA), noise functions (Perlin, Simplex, Worley), fractal systems (L-systems, Hilbert curves), and machine learning-based algorithms (neuroevolution, Wave Function Collapse, Markov models). A classification of algorithms by the type of underlying structure (graph, grid, automaton, noise, ML model) is proposed, allowing for the systematization of maze generation approaches based on architectural and functional characteristics. It is established that classical algorithms offer high predictability and performance, while cellular automata and hybrid methods enable the creation of complex, organic, or decorative structures. The scientific novelty of the study lies in the

development of a unified classification of maze generation algorithms that consider both structural and gameplay parameters, thus enabling informed choices of optimal solutions for specific game design tasks. The practical significance of this work lies in its applicability to build adaptive level generation systems, creating educational platforms for studying algorithms, and developing recommendation systems for selecting algorithms based on game genre, technical constraints, and desired complexity.

Keywords: algorithms; optimality of algorithms; graphs; maze generation; game development; maze; machine learning.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Marchuk, G.V. & Liubchenko, D.V. (2024). Generation of mazes using the hunt-and-kill algorithm. *Scientific notes of Taurida National V.I. Vernadsky University*. *Series: Technical Sciences*, 35 (74), 3, 130-135. https://doi.org/10.32782/2663-5941/2024.3.1/20
2. Binayak, B. (2025). Interactive Maze Generation and Pathfinding Algorithms: An Educational Visualization Platform for Algorithmic Analysis. *Zenodo*. https://doi.org/10.5281/zenodo.15824242
3. Vijaya, J., Gopu, A., Vinayak, K., Singh, J., Malhotra, K. (2024). Analysis of Maze Generation Algorithms. *5th IEEE International Conference on Innovative Trends in Information Technology (ICITIIT)*, Kottayam, India, 15-16 March 2024, 1-6. https://doi.org/10.1109/ICITIIT61487.2024.10580178
4. Čarapina, M., Staničić, O., Dodig, I., & Cafuta, D. (2024). A Comparative Study of Maze Generation Algorithms in a Game-Based Mobile Learning Application for Learning Basic Programming Concepts. *Algorithms*, 17(9), 404. https://doi.org/10.3390/a17090404
5. Cahyakusuma, I. & Istiono, W. (2024). Algorithmic diversity in maze generation: Comparative study of Backtracking, Kruskal's, Prim's, and Eller's algorithms. *International Journal of Intelligent Systems and Applications in Engineering*, 12(3), 1281–1287. https://ijisae.org/index.php/IJISAE/article/view/5518
6. Marchuk, G. V., Graf, M. S., Levkivskyi, V. L., & Venhlovska, Yu. V. (2024). Analysis and comparison of existing maze generation methods in computer games. *Visnyk of Kherson National Technical University*, 3(90), 228-237. https://doi.org/10.35546/kntu2078-4481.2024.3.29
7. Mane, D., Harne, R., Pol, T., Asthagi, R., Shine, S., & Zope, B. (2023). An Extensive Comparative Analysis on Different Maze Generation Algorithms. *International Journal of Intelligent Systems and Applications in Engineering*, 12(2), 37-47. https://ijisae.org/index.php/IJISAE/article/view/3557
8. Piao, J., Hu, X., & Zhou, Q. (2024). Maze and navigation algorithms in game development. *Applied and Computational Engineering*, 79, 11-19. https://doi.org/10.54254/2755-2721/79/20241081
9. Malem, F. (2023). Wall Object Design in Maze Game Using Cellular Automata Algorithm. *Journal of Computer Engineering: Progress, Application and Technology*. 2(1), 27-35. https://doi.org/10.25124/cepat.v2i01.5781
10. Yang, K., Lin, S., Dai, Yu., & Li, W. (2024). Optimization and comparative analysis of maze generation algorithm hybrid. *Applied and Computational Engineering*, 79, 20-33. https://doi.org/10.54254/2755-2721/79/20241082
11. Trofymenko, O., Zadereyko O., Balandina, N., Tolocknov, A., & Huselnikov, I. (2024). Vector algebra for gamedev. *Electronic Professional Scientific Journal «Cybersecurity: Education, Science, Technique»*, 2(26), 71-80. https://doi.org/10.28925/2663-4023.2024.26.626
12. Xu, J. & Morris, J. (2023). Procedural generation in 2d metroidvania game with answer set programming and perlin noise. *International Journal of Artificial Intelligence & Applications*, 14, 21-35. https://doi.org/10.5121/ijaia.2023.14302
13. Trofymenko, O.G., Zadereyko, O.V., Loginova, N.I., & Shevchenko, O.V. (2025). Artificial intelligence tools in the development of graphic elements, 3D models and other video game components. *Informatics and mathematical methods in simulation*, 15(2), 276-287. https://doi.org/10.15276/imms.v15.no2.276
14. Alaka, Sh. & Bidarra, R. (2023). Hierarchical Semantic Wave Function Collapse. *FDG '23: Proceedings of the 18th International Conference on the Foundations of Digital Games*, 68, 1-10. https://doi.org/10.1145/3582437.3587209
15. Linden, R., Lopes, R., & Bidarra, R. (2014). Procedural Generation of Dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*. Volume: 6, Issue: 1, March 2014, 78-89. https://doi.org/10.1109/TCIAIG.2013.2290371

16. Tanaka T. & Simo-Serra E. Grammar-based Game Description Generation using Large Language Models. *arXiv.* https://doi.org/10.48550/.2407.17404
17. Shaker, N., Liapis, A., Togelius, Ju., Lopes, R., & Bidarra, R. (2016). Constructive generation methods for dungeons and levels. *Computational Synthesis and Creative Systems (CSACS)*, 31-55. https://doi.org/10.1007/978-3-319-42716-4_3
18. Markov Chains for Procedural Buildings. https://nickmcd.me/2019/10/30/markov-chains-for-procedural-buildings/
19. Dou, H., Du, J., Jiang, X., H, Li., Yao, W., & Deng, Y. (2025). Image-to-Image Bayesian Flow Networks With Structurally Informative Priors. *IEEE Transactions on Image Processing*, 34, 4968-4982. https://doi.org/10.1109/TIP.2025.3592546
20. Setiadharma, E., Husniah, L., & Kholimi, A. S. (2024). Algoritma Maze Generator Recursive Backtracking Untuk Membuat Prosedural Labirin Pada Game Petualangan Labirin 3D. *Jurnal Repositor*, 2(3), 373-384. https://doi.org/10.22219/repositor.v2i3.30509
21. Duffy, Ch., Hillis, S., Khan, U., McQuillan, I., & Shan, S. (2025). Inductive inference of lindenmayer systems: algorithms and computational complexity. *Natural Computing*. 1-11. https://doi.org/10.1007/s11047-025-10024-x.
22. Yuichi, N., Akinori, I., & Norihiko, O. (2019). A Genetic Algorithm for the Picture Maze Generation Problem. *Computers & Operations Research*, 115, 104860. https://doi.org/10.1016/j.cor.2019.104860
23. Krishnaa, K., Kaluri, J., Konjeti, S., Aravind, T., & Gurusamy, J. (2022). A Genetic Algorithm Framework to Solve Two-Dimensional Maze Problem. *Proceedings of 3rd International Conference on Machine Learning, Advances in Computing, Renewable Energy and Communication*, 2022, 18 September, 277-284. https://doi.org/10.1007/978-981-19-2828-4_27.
24. Trofymenko, O., Dyka, A., Loboda, Y., Tolocknov, A., & Bondarenko, M. (2025). Artificial intelligence in the gamedev. *Electronic Professional Scientific Journal «Cybersecurity: Education, Science, Technique»*, *3*(27), 109-119. https://doi.org/10.28925/2663-4023.2025.27.701
25. Khalifa, A., Bontrager, P., Earle, S., & Togelius, J. PCGRL: Procedural Content Generation via Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* 2020. 16(1), 95-101. https://doi.org/10.1609/aiide.v16i1.7416
26. Trofymenko, O., Dyka, A., Zadereyko O., & Shevchenko O. (2025). Areas of application of artificial intelligence in game development and testing. *Electronic Professional Scientific Journal «Cybersecurity: Education, Science, Technique»*, *1*(29), 41-58. https://doi.org/10.28925/2663-4023.2025.29.832
27. Summerville, A., et al. Procedural content generation via machine learning (PCGML). *arXiv*. 1702.00539. https://doi.org/10.48550/arXiv.1702.00539
28. Alaguna C., Gomez Jo. Maze benchmark for testing evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '18).* 2018. 1321-1328. https://doi.org/10.1145/3205651.3208285