УДК 004.421 **DOI: 10.15330/itee.2026.4.06**

Ю.Ю. ІЛЯШ, канд. техн. наук, Д.А. КОЗАЧОК, студентка

ПРОЦЕДУРНА ГЕНЕРАЦІЯ 2D-КАРТ НА ОСНОВІ КЛІТИННИХ АВТОМАТІВ

Вступ

У даний час створення ігрових рівнів та віртуальних середовищ у комп'ютерних іграх здійснюється як за допомогою ручного проєктування, так і із застосуванням алгоритмів процедурної генерації [1–3]. Традиційний підхід передбачає детальну розробку кожного рівня дизайнером, що забезпечує високий рівень контролю над структурою середовища, однак потребує значних часових і трудових витрат та обмежує масштаб ігрових світів.

Процедурна генерація контенту дозволяє автоматизувати процес створення ігрових карт, забезпечуючи варіативність, масштабованість та унікальність кожного проходження гри [1, 2]. У сучасних дослідженнях застосовуються різноманітні алгоритмічні підходи, зокрема шумові методи, фрактальні алгоритми [4-6], а також генерація складних ігрових елементів, таких як наративні структури або завдання [7, 8].

У роботі [9] клітинні автомати розглядаються як один із ефективних підходів до процедурної генерації карт, що дозволяє формувати природоподібні структури, проте питання впливу параметрів генерації на топологічні властивості сформованих карт не отримали детального висвітлення.

Лазарос Лазарідіс та Джордж Ф. Фрагіліс [10] запропонували алгоритм генерації рівнів із автоматичним розміщенням кімнат та перевіркою їх взаємного розташування. Запропонований підхід орієнтований на модульні кімнатні структури, тоді як генерація безперервних просторів потребує окремих досліджень.

Одним із перспективних підходів до розв'язання цієї задачі є використання клітинних автоматів — математичних моделей, що базуються на локальній взаємодії елементів і здатні формувати складні просторові структури [11–15]. Додатково їх ефективність підтверджується у прикладних задачах, зокрема обробці зображень та фільтрації шумів [16], що свідчить про універсальність даного підходу.

Таким чином, застосування клітинних автоматів у задачах процедурної генерації двовимірних карт є актуальним напрямом досліджень, який дозволяє поєднати простоту алгоритмічної реалізації з можливістю формування природоподібних ігрових середовищ.

Постановка проблеми

Сучасна індустрія комп'ютерних ігор характеризується постійним зростанням вимог до масштабності, складності та варіативності ігрових світів. Користувачі очікують великих відкритих просторів, унікальних локацій та високого потенціалу для повторного проходження, що ускладнює процес розробки контенту. Традиційне ручне проєктування рівнів потребує значних часових і трудових ресурсів, а також обмежує обсяг створюваного середовища. За умов збільшення розмірів карт і кількості ігрових елементів ручний підхід стає малоефективним та занадто трудомістким.

Автоматизація створення ігрових світів за допомогою алгоритмів процедурної генерації дозволяє вирішити проблему надмірних витрат часу та ресурсів. Застосування алгоритмічних

методів дає можливість формувати великі карти за короткий проміжок часу, забезпечувати унікальність кожного запуску гри та підтримувати динамічну зміну середовища.

Окремою проблемою є контрольованість випадкових алгоритмів. Більшість методів процедурної генерації базуються на використанні псевдовипадкових чисел, що може призводити до хаотичних або нефункціональних структур. Тому важливо розробити такі підходи, які поєднують елемент випадковості з чітко визначеними правилами формування простору. Одним із перспективних інструментів для розв'язання цієї задачі є клітинні автомати, що дозволяють отримувати природні структури шляхом застосування локальних правил переходу станів клітин.

Аналіз досліджень і публікацій

У сучасних дослідженнях процедурної генерації ігрового контенту значна увага приділяється автоматизації створення карт, рівнів та ігрових світів. У роботах [1-3] наведено узагальнений огляд алгоритмів процедурної генерації, включаючи генерацію рівнів, ландшафтів, об'єктів та сюжетних елементів. Автори підкреслюють, що використання таких підходів дозволяє значно зменшити трудові витрати розробників та забезпечити високу варіативність ігрового контенту. Проте більшість оглядових досліджень зосереджені на комплексних системах генерації, що інтегрують різноманітні алгоритмічні підходи та часто орієнтовані на великі комерційні проєкти. Це ускладнює їх застосування у невеликих програмних системах або інді-іграх, де ключовими вимогами є простота реалізації алгоритмів, низька ресурсомісткість та можливість швидкої модифікації генератора карт.

Окремий напрям досліджень присвячений генерації конкретних типів ігрових рівнів, зокрема лабіринтів або підземель. У роботах [4-6, 14] розглядаються методи генерації dungeon-рівнів та лабіринтоподібних структур, що базуються на використанні еволюційних та пошукових алгоритмів, а також спеціальних функцій пристосованості для визначення якості сформованих карт. Такі підходи дозволяють отримувати карти із заданими характеристиками, наприклад необхідною довжиною маршрутів або рівнем зв'язності областей. Водночас суттєвим недоліком подібних методів є складність підбору та налаштування критеріїв оцінки якості генерації, а також значні обчислювальні витрати, пов'язані з роботою оптимізаційних алгоритмів. У результаті такі підходи можуть виявитися надто складними для використання в системах реального часу або в невеликих ігрових проєктах.

У роботах [7, 8] розглядаються підходи до процедурної генерації складніших елементів ігрового середовища, таких як наративні світи або ігрові завдання. Зокрема пропонується використовувати шаблони наративних структур або генетичні алгоритми для створення взаємопов'язаних об'єктів і подій у межах ігрового світу. Хоча такі системи дозволяють формувати складні сюжетні структури, їх реалізація вимагає попереднього формування великої бази знань про ігровий світ, а також застосування складних механізмів планування та оптимізації. Це обмежує їх використання у задачах генерації геометричної структури ігрових карт, де необхідна швидка та ефективна генерація великої кількості варіантів середовища.

Значну роль у розвитку процедурної генерації відіграють дослідження клітинних автоматів як універсальних моделей самоорганізації складних систем. У роботах [11-15] показано, що прості локальні правила взаємодії між елементами системи можуть призводити до формування складних структур та патернів. Ці властивості роблять клітинні автомати перспективним інструментом для генерації природоподібних структур, зокрема печерних систем або

ландшафтів. Разом з тим більшість таких досліджень мають переважно теоретичний характер і спрямовані на моделювання фізичних процесів або аналіз складних систем, що потребує додаткової адаптації цих підходів для практичного застосування у генерації ігрових карт.

У деяких прикладних роботах клітинні автомати використовуються для задач обробки зображень. Зокрема у дослідженні [16] показано ефективність таких методів для фільтрації шумів та виявлення контурів на зображеннях. Хоча результати демонструють високу ефективність локальних правил взаємодії клітин, подібні підходи спрямовані переважно на аналіз існуючих даних, а не на генерацію нових структур. Тому їх використання у задачах процедурної генерації карт потребує модифікації правил переходу клітин та адаптації алгоритму до вимог генерації ігрових рівнів.

Таким чином, аналіз сучасних досліджень показує, що більшість існуючих підходів до процедурної генерації ігрового контенту або орієнтовані на складні системи генерації великих віртуальних світів, або потребують використання ресурсомістких алгоритмів оптимізації. У зв'язку з цим актуальним є використання більш простих і ефективних алгоритмів, здатних забезпечити прийнятну якість генерації при мінімальних обчислювальних витратах. Одним із таких підходів є використання клітинних автоматів, які дозволяють формувати природоподібні структури карт на основі простих локальних правил взаємодії між клітинами.

Формулювання мети дослідження

Метою роботи є дослідження можливостей використання клітинних автоматів для процедурної генерації двовимірних карт. У межах роботи передбачається реалізувати алгоритм генерації карти на основі клітинного автомата, провести експериментальне тестування параметрів генерації та дослідити вплив кількості ітерацій і початкової щільності стін на структуру сформованої карти. Отримані результати дозволяють оцінити ефективність даного підходу для створення природних та варіативних ігрових середовищ. У межах дослідження також виконано порівняльний аналіз результатів клітинних автоматів із початковою випадковою генерацією карти, з метою оцінити ефективність запропонованого підходу.

Основні принципи програмної генерації та клітинних автоматів

Процедурна генерація контенту є одним із ключових підходів до створення ігрових середовищ, який передбачає автоматичне формування об'єктів, рівнів або карт за допомогою алгоритмів та математичних моделей. Вона значно спрощує процес розробки великих ігрових світів та забезпечує високу варіативність отриманого результату [1].

Застосування процедурної генерації особливо актуальне у випадках, коли необхідно створити масштабні та складні віртуальні простори. За допомогою алгоритмів можна автоматично генерувати рельєф місцевості, рівні гри, підземні структури, текстури та карти [1]. Завдяки використанню псевдовипадкових чисел та математичних правил стає можливим формування різноманітних конфігурацій середовища, що забезпечує унікальний досвід під час кожного запуску програми.

Однією з головних переваг процедурної генерації є можливість створення великої кількості варіантів ігрового простору при мінімальних витратах ресурсів. Алгоритмічний підхід дозволяє значно скоротити час розробки, а також адаптувати середовище до дій користувача. Завдяки цьому ігрові світи можуть змінюватися динамічно, що підвищує рівень реалізму та зацікавленість гравця [2].

У контексті процедурної генерації важливе місце займають двовимірні карти, які використовуються для представлення ігрового простору у багатьох застосунках та відеоіграх. Найпоширенішим типом таких карт є модульні або грід-карти, які складаються з регулярної сітки клітин. Кожна клітина має власні координати та описує певний елемент середовища, наприклад землю, воду, перешкоду або інший об'єкт.

Іншим поширеним типом є висотні карти, що представляють рельєф місцевості у вигляді матриці числових значень. Також використовуються біомні карти, які визначають кліматичні або природні зони світу на основі параметрів температури та вологості. Окрім цього застосовуються топологічні карти, що базуються на графових структурах і дозволяють описувати взаємозв'язки між різними частинами рівня.

Поєднання різних типів карт та алгоритмів генерації дозволяє створювати складні та різноманітні ігрові середовища, які можуть змінюватися залежно від заданих параметрів або випадкових факторів [1].

Для автоматичного створення двовимірних карт використовується велика кількість алгоритмів процедурної генерації. Найбільш поширеними є шумові алгоритми, фрактальні методи, стохастичні алгоритми, діаграмні методи та клітинні автомати. Для більш наочного аналізу різних підходів до процедурної генерації карт в роботі здійснено їх порівняння за основними характеристиками: контрольованість – наскільки розробник впливає на процес генерації, точність – відповідність результату завдань специфікації, різноманітність – здатність алгоритму генерувати різні результати при кількох запусках, складність – рівень деталізації сформованих карт, локальна взаємодія – використання інформації про сусідні клітини або елементи, а також типові види структур, при яких використовують кожен метод (табл. 1).

Таблиця 1. Порівняння методів процедурної генерації карт

Методи генерації	Контрольованість	Точність	Різноманітність	Складність структури	Локальна взаємодія	Типове застосування
Шумові алгоритми	Середня	Середня	Висока	Висока	Відсутня	Ландшафти, висотні карти, текстури
Фрактальні методи	Середня	Середня	Середня	Висока	Часткова	Рельєф місцевості,
Стохастичні алгоритми	Низька	Низька	Висока	Низька	Відсутня	Початкове заповнення карти,
Діаграмні методи	Висока	Висока	Середня	Середня	Часткова	Біоми, зони карт
Клітинні автомати	висока	Висока	Середня	Висока	На основі сусідніх клітин	Печери, підземелля, лабіринти

За результатами аналізу встановлено, що клітинні автомати поєднують достатньо високий рівень контрольованості процесу генерації з можливістю формування складних структур на основі локальної взаємодії клітин. Саме ця властивість робить їх ефективним інструментом для створення печерних систем, підземель та інших природоподібних структур у двовимірних ігрових картах.

До шумових алгоритмів належить Perlin Noise, який використовується для генерації плавних випадкових структур. Цей метод дозволяє створювати природні ландшафти та текстури,

що мають плавні переходи між значеннями. Подібним підходом є Simplex Noise, який працює за схожим принципом, але використовує іншу геометричну структуру сітки.

Фрактальні методи також широко застосовуються у процедурній генерації. Одним із прикладів є алгоритм Diamond–Square, який використовується для створення гірських рельєфів та висотних карт. Завдяки фрактальним властивостям такі алгоритми дозволяють отримувати природні нерівні поверхні.

До стохастичних методів належить випадкове заповнення карти, при якому кожна клітина отримує випадкове значення. Хоча цей метод є дуже простим у реалізації, він зазвичай використовується як початковий етап перед застосуванням більш складних алгоритмів.

Діаграмні методи, зокрема діаграми Вороного, дозволяють розділяти простір на області та використовуються для створення біомів або топологічних структур. Окреме місце серед методів процедурної генерації займають клітинні автомати, які дозволяють формувати складні природні структури за допомогою простих локальних правил.

Клітинний автомат є дискретною математичною моделлю, що використовується для моделювання складних систем через взаємодію простих елементів. Основою такого підходу є регулярна сітка клітин, кожна з яких може перебувати в одному з декількох можливих станів [4]. Зміна стану клітини визначається певними правилами переходу, які враховують її поточний стан та стан сусідніх клітин.

Важливим елементом клітинних автоматів є поняття сусідства. Найпоширенішими типами є сусідство фон Неймана, при якому враховуються чотири сусідні клітини, та сусідство Мура, що включає вісім клітин, розташованих навколо поточної [16].

У задачах процедурної генерації карта зазвичай представляється у вигляді двовимірної матриці станів. Кожен елемент такої матриці відповідає окремій клітині та зберігає інформацію про її тип або стан. Найчастіше використовують просте кодування, наприклад значення 0 для позначення порожнього простору та 1 для позначення стіни або перешкоди.

Початковий стан карти зазвичай формується випадковим чином або за допомогою шумових алгоритмів. Після цього алгоритм клітинного автомата виконує кілька ітерацій, під час яких стан кожної клітини змінюється відповідно до заданих правил. Завдяки багаторазовому застосуванню цих правил випадкова структура поступово перетворюється на більш впорядковану та природну форму.

Таким чином, клітинні автомати є ефективним інструментом процедурної генерації, що дозволяє отримувати складні структури з простих початкових конфігурацій. Цей підхід широко використовується для генерації печер, островів, лабіринтів та інших елементів ігрових карт.

Викладення основного матеріалу дослідження

Алгоритм генерації карти за допомогою клітинного автомата базується на ітеративному застосуванні правил переходу до клітин двовимірної сітки. Кожна клітина має певний стан, наприклад: 0 — порожній простір (прохід), 1 — твердий блок (стіна). Карта представляється у вигляді двовимірної матриці розміру $N \times M$, де N — висота карти, а M — її ширина. Кожен елемент матриці відповідає окремій клітині ігрового простору.

Робота алгоритму складається з кількох основних етапів:

1. Ініціалізація карти випадковими значеннями.
2. Визначення правил переходу клітинного автомата.

3. Виконання декількох ітерацій оновлення карти.

4. Візуалізація отриманої карти.

Початкове випадкове заповнення створює базову структуру карти, яка потім поступово змінюється під час ітерацій алгоритму. Завдяки цьому випадкові шумові структури поступово перетворюються на більш природні та впорядковані форми(рис. 1(б)).

На першому етапі алгоритму виконується початкове випадкове заповнення карти. Для кожної клітини визначається її початковий стан на основі заданої ймовірності.

Наприклад, якщо ймовірність появи стіни становить **приблизно 50 %**, то для кожної клітини генерується випадкове число. Якщо воно перевищує заданий поріг, клітина отримує стан стіни, інакше — стан порожнього простору.

Проведені дослідження показали, що початкова ймовірність появи стін істотно впливає на структуру згенерованої карти. Занадто низькі значення цього параметра призводять до формування великої кількості відкритих просторів, тоді як занадто високі значення спричиняють появу надмірно щільних та непрохідних структур. Практичні експерименти показали, що оптимальним є діапазон 45–50 %, який забезпечує баланс між відкритими та закритими областями карти.

Такий підхід дозволяє отримати випадкову карту, яка має хаотичну структуру (рис. 1(а)). Однак ця структура є лише початковим етапом генерації. Під час подальших ітерацій клітинного автомата вона буде змінюватися відповідно до заданих правил.

Після ініціалізації карти визначаються правила переходу клітинного автомата. Вони описують, як змінюється стан клітини залежно від станів її сусідів.

У більшості алгоритмів генерації печер використовується **сусідство Мура**, яке враховує вісім клітин навколо поточної.

Правила переходу клітинного автомата залежать від поточного стану клітини та кількості сусідніх клітин-стін у сусідстві Мура. Якщо поточна клітина є стіною, вона зберігає цей стан за наявності чотирьох або більше сусідніх стін, інакше переходить у стан проходу. Якщо поточна клітина є проходом, вона перетворюється на стіну за наявності п’яти або більше сусідніх стін, інакше залишається проходом.

Ці правила виконують роль фільтра, який усуває випадковий шум у структурі карти та формує більш компактні області.

Після кількох ітерацій застосування правил дрібні випадкові області зникають, а великі структури стають більш вираженими (табл. 2).

Таблиця 2. Правила переходу клітинного автомата

Поточний стан клітини	Кількість сусідів-стін	Новий стан
Стіна(1)	>=4	Стіна(1)
Стіна(1)	<4	Прохід(0)
Прохід(0)	>=5	Стіна(1)
Прохід(0)	<5	Прохід(0)

У процесі дослідження також було проаналізовано вплив порогового значення кількості сусідніх клітин у сусідстві Мура. Зменшення цього значення призводить до надмірного розростання областей стін та утворення щільних малопрохідних структур, тоді як його збільшення

сприяє фрагментації карти та появі великої кількості ізольованих порожнин. Проведені експерименти показали, що використання порогового значення, близького до п’яти сусідніх клітин, забезпечує баланс між процесами розширення та руйнування областей, завдяки чому формуються зв'язані простори з плавними межами та природоподібними переходами між окремими ділянками карти.

Після визначення правил переходу алгоритм виконує кілька ітерацій оновлення карти (рис. 1 (в)).

Під час кожної ітерації:

1. для кожної клітини підраховується кількість сусідніх клітин зі станом «стіна»;
2. відповідно до правил переходу визначається новий стан клітини;
3. нові значення записуються в окрему матрицю.

Використання двох матриць дозволяє забезпечити **паралельне оновлення клітин**, оскільки зміна стану однієї клітини не повинна впливати на обчислення інших у межах тієї ж ітерації.



а) б) в)

Рис. 1 – Етапи формування карти за допомогою клітинного автомата

Одним із ключових параметрів алгоритму клітинного автомата є кількість ітерацій оновлення карти, яка зазвичай становить від 4 до 10. Проведені експериментальні дослідження показали, що зі збільшенням кількості ітерацій випадкова початкова структура поступово згладжується, формуючи більш зв’язані та компактні області, що наближаються до природоподібних печерних структур. Водночас надто мала кількість ітерацій призводить до збереження значної частини хаотичних елементів початкової генерації. За результатами експериментального тестування встановлено, що використання приблизно 4–7 ітерацій забезпечує оптимальний баланс між структурованістю карти та різноманітністю отриманих конфігурацій. Критерієм такого вибору є зменшення кількості ізольованих областей та формування зв'язаних просторів без надмірного згладжування структури карти, що дозволяє зберегти достатню варіативність конфігурацій.

Після завершення основних ітерацій алгоритму можуть застосовуватися додаткові методи обробки карти.

До них належать:

- Згладжування – усуває різкі переходи між областями карти та робить структуру більш природною.

• Видалення ізольованих областей – невеликі ізольовані порожнини або острови можуть бути видалені для покращення ігрового процесу.

• З'єднання печер – якщо на карті утворилися ізольовані печери, між ними можуть бути створені штучні коридори.

Для реалізації алгоритму клітинного автомата було використано мову програмування **C++**. Карта представлена у вигляді двовимірного масиву, який зберігає стан кожної клітини.

Програмна реалізація алгоритму клітинного автомата побудована за модульним принципом. Основні компоненти системи включають модуль конфігурації, модуль генерації карти, модуль правил клітинного автомата та модуль візуалізації. Такий підхід дозволяє розділити логіку програми на окремі функціональні блоки, що спрощує зміну параметрів алгоритму та подальше розширення системи.

Модуль конфігурації містить основні параметри генерації карти: її розміри, розмір клітини для відображення, кількість ітерацій роботи автомата та ймовірність появи стіни під час початкової ініціалізації.

Фрагмент коду конфігурації:

```
const int WIDTH = 100;
const int HEIGHT = 100;
const int CELL_SIZE = 6;
const int ITERATIONS = 10;
const int WALL_CHANCE = 47;
```

Для зберігання карти використовується двовимірна структура даних типу `Grid`, що представляє клітинну сітку, де кожна клітина може мати значення стіни або проходу:

```
using Grid = vector<vector<int>>;
Grid createEmptyGrid()
{
    return Grid(HEIGHT, vector<int>(WIDTH, 0));
}
```

Модуль генерації карти відповідає за створення початкового стану та виконання ітерацій клітинного автомата. Початкова карта формується випадковим заповненням клітин відповідно до параметра `WALL_CHANCE`:

```
Grid randomInit()
{
    Grid g = createEmptyGrid();
    for (int y = 0; y < HEIGHT; y++)
        for (int x = 0; x < WIDTH; x++)
            g[y][x] = rand() % 100 < WALL_CHANCE;
    return g;
}
```

Правила клітинного автомата реалізуються через підрахунок сусідів за схемою сусідства Мура та застосування правил переходу для кожної клітини:

```
int countNeighbors(const Grid& g, int x, int y)
{
    int c = 0;
    for (int dy = -1; dy <= 1; dy++)
        for (int dx = -1; dx <= 1; dx++)
        {
```

```
            if (dx == 0 && dy == 0) continue;
            int nx = x + dx;
            int ny = y + dy;
            if (nx < 0 || ny < 0 || nx >= WIDTH || ny >= HEIGHT || g[ny][nx])
                c++;
        }
    return c;
}
```

На кожній ітерації виконується оновлення стану карти відповідно до кількості сусідніх клітин:

```
Grid step(const Grid& cur)
{
    Grid next = cur;
    for (int y = 0; y < HEIGHT; y++)
        for (int x = 0; x < WIDTH; x++)
        {
            int n = countNeighbors(cur, x, y);
            if (cur[y][x])
                next[y][x] = n >= 4;
            else
                next[y][x] = n >= 5;
        }
    return next;
}
```

Модуль візуалізації реалізований за допомогою бібліотеки SFML та відповідає за відображення карти у вигляді клітинної сітки, де кожна клітина представлена прямокутником відповідного кольору:

```
void drawGrid(RenderWindow& window, const Grid& grid)
{
    RectangleShape cell(Vector2f((float)CELL_SIZE, (float)CELL_SIZE));
    for (int y = 0; y < HEIGHT; y++)
        for (int x = 0; x < WIDTH; x++)
        {
            if (grid[y][x])
                cell.setFillColor(Color(255, 200, 210));
            else
                cell.setFillColor(Color(220, 220, 220));
            cell.setPosition(Vector2f(
                (float)(x * CELL_SIZE),
                (float)(y * CELL_SIZE)));
            window.draw(cell);
        }
}
```

Покрокове виконання алгоритму здійснюється в головному циклі програми, де після кожної ітерації клітинного автомата карта оновлюється та відображається на екрані:

```
        if (iteration < ITERATIONS &&
            clock.getElapsedTime().asSeconds() >= stepDelay)
        {
            map = step(map);
            ++iteration;
            clock.restart();}
```

Такий підхід дозволяє спостерігати поступове формування структури карти та аналізувати вплив параметрів генерації на результат роботи алгоритму.

Програма реалізує базовий алгоритм клітинного автомата для генерації карти. Вона виконує випадкову ініціалізацію карти, після чого застосовує кілька ітерацій правил переходу та виводить результат у консоль.

Для наочного демонстрування процесу формування структури карти було проаналізовано еволюцію клітинного автомата на різних етапах виконання алгоритму(рис. 1(а-в)).

У процесі виконання ітерацій клітинного автомата відбувається поступове згладжування випадкових структур та формування більш зв’язаних областей. На фінальному етапі генерації утворюються природоподібні печерні структури з компактними областями проходів і стін.

Для більш об'єктивного оцінювання ефективності запропонованого підходу було проведено експериментальне дослідження характеристик згенерованих карт. Під час аналізу враховувалися такі показники, як щільність стін, частка прохідних клітин, кількість зв'язних компонент та середній час генерації. Отримані результати дозволяють оцінити вплив кількості ітерацій клітинного автомата на структуру сформованого середовища та простежити зміни його топологічних характеристик. Основні результати експериментального дослідження та порівняння початкової випадкової генерації з результатами застосування клітинного автомата наведено в таблиці 3.

Таблиця 3. Залежність характеристик карти від параметрів генерації

Метод	Початкова ймовірність появи стін ,%	Кількість ітерацій	Щільність стін, %	Прохідність,%	Кількість компонентів	Час генерації, мс
Випадкове заповнення	50	0	50	50	70	0,3
Клітинний автомат	50	4	53	47	16	1,2
Клітинний автомат	50	7	57	43	7	1,9
Клітинний автомат	50	10	61	39	4	2,6

Як видно з таблиці 3, збільшення кількості ітерацій клітинного автомата призводить до зростання щільності стін та зменшення кількості зв'язних компонент, що свідчить про поступове згладжування випадкових структур і формування більш компактних областей. Водночас надмірне збільшення кількості ітерацій зменшує частку прохідних клітин та різноманітність конфігурацій. За результатами проведених експериментів найбільш збалансовані характеристики спостерігаються при 4–7 ітераціях алгоритму.

Розроблена програма процедурної генерації двовимірних карт на основі клітинних автоматів може бути застосована у широкому спектрі задач, що потребують автоматичного створення структурованих віртуальних середовищ. Отримані результати показують, що запропонований підхід забезпечує формування зв’язаних та придатних для навігації карт при невисоких обчислювальних витратах, що робить його придатним для використання у відеоіграх жанрів roguelike, dungeon crawler та sandbox, де необхідна генерація різноманітних рівнів, печерних систем і підземель. Крім того, алгоритм може бути використаний у навчальних симуля-

ціях та освітніх застосунках для моделювання просторових середовищ, а також у задачах прототипування та тестування ігрових механік. Завдяки можливості змінювати параметри генерації та правила переходу клітин запропонований підхід може застосовуватися і в дослідницьких проєктах, пов’язаних із вивченням алгоритмів процедурної генерації та адаптивних систем.

Загалом, простота реалізації, гнучкість налаштування параметрів та модульна архітектура роблять розроблену програму ефективним інструментом для створення процедурних середовищ у різноманітних ігрових, освітніх і наукових застосуваннях, де важливими є варіативність, масштабованість та природність сформованих структур.

Висновки

У роботі досліджено алгоритмічні підходи до процедурної генерації двовимірних ігрових карт із використанням клітинних автоматів як інструменту формування структурованого ігрового простору. Реалізовано та досліджено алгоритм процедурної генерації двовимірних карт на основі клітинного автомата, виконано аналіз впливу основних параметрів генерації та проведено порівняння отриманих результатів із початковим випадковим заповненням карти.

У межах дослідження розроблено програмну реалізацію алгоритму мовою програмування C++ із використанням графічної бібліотеки SFML, що забезпечує візуалізацію процесу генерації карти в режимі послідовних ітерацій. Модульна архітектура програмної системи дозволяє змінювати параметри генерації та розширювати функціональні можливості програми.

Проведене експериментальне дослідження показало, що найбільший вплив на структуру карти мають початкова щільність стін, кількість ітерацій клітинного автомата та порогові значення правил переходу. Встановлено, що початкова ймовірність появи стін у межах 45–50 % забезпечує баланс між відкритими та закритими областями карти, а використання 4–7 ітерацій дозволяє зменшити кількість ізольованих областей та сформувати зв'язані простори без надмірного згладжування структури. Використання порогового значення, близького до п'яти сусідніх клітин, сприяє формуванню природоподібних переходів між областями та зменшує фрагментацію карти. Якість отриманих карт оцінювалася за щільністю стін, часткою прохідних клітин, кількістю зв'язних компонент та середнім часом генерації. Експериментальні результати показали, що застосування клітинного автомата дозволяє отримувати більш структуровані та придатні для навігації карти порівняно з початковим випадковим заповненням.

Практична реалізація підтвердила ефективність використання клітинних автоматів для генерації ігрових карт, оскільки цей підхід поєднує простоту реалізації, можливість керування параметрами генерації та високу варіативність результатів. Розроблений алгоритм може застосовуватися під час створення прототипів відеоігор, навчальних симуляцій, дослідницьких програм і систем моделювання середовищ, де необхідне автоматичне формування складних просторових структур.

Разом із тим запропонований підхід не гарантує повної зв'язності всіх областей карти та може потребувати застосування додаткових алгоритмів для видалення ізольованих областей або з'єднання окремих печерних структур. Перспективним напрямом подальших досліджень є автоматичний підбір параметрів клітинного автомата, удосконалення механізмів оцінювання якості згенерованих карт, а також використання алгоритмів пошуку шляхів для перевірки доступності окремих ділянок і забезпечення коректної навігації у сформованому середовищі.

Список використаної літератури

1. T. Gao, J. Zhang, and Q. Mi, “Procedural generation of game levels and maps: A review,” in Proc. 2022 Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC), Jeju Island, Korea, Feb. 2022, pp. 248–253. doi: https://doi.org/10.1109/icaiic54071.2022.9722624 ..
2. B. M. F. Viana and S. R. dos Santos, “Procedural dungeon generation: A survey,” J. Interact. Syst., vol. 12, no. 1, pp. 83–101, 2021. doi: https://doi.org/10.5753/jis.2021.999.
3. E. W. Hidayat, E. N. F. Dewi, and I. S. Ramadhan, “Application of procedural content generation system in forming dungeon level in Dungeon Diver game,” J. Tek. Inform. (Jutif), vol. 5, no. 3, pp. 873–881, 2024. doi: https://doi.org/10.52436/1.jutif.2024.5.3.1465.
4. Z. Shen, “Procedural generation in games: Focusing on dungeons,” SHS Web Conf., vol. 144, p. 02005, 2022. doi: https://doi.org/10.1051/shsconf/202214402005.
5. J. Vijaya et al., “Analysis of maze generation algorithms,” in Proc. 2024 5th Int. Conf. Innov. Trends Inf. Technol. (ICITIIT), Kottayam, India, Feb. 2024, pp. 1–6. doi: https://doi.org/10.1109/ICITIIT61487.2024.10580178.
6. G. N. Yannakakis and J. Togelius, “Procedural content generation by content type,” in Artificial Intelligence and Games, Cham, Switzerland: Springer, 2025, pp. 287–312. doi: https://doi.org/10.1007/978-3-031-83347-2_9.
7. J. T. Balint and R. Bidarra, “Procedural generation of narrative worlds,” IEEE Trans. Games, vol. 15, no. 2, pp. 143–153, Jun. 2023. doi: https://doi.org/10.1109/tg.2022.3216582.
8. E. Soares de Lima, B. Feijó, and A. L. Furtado, “Procedural generation of quests for games using genetic algorithms and automated planning,” in Proc. 2019 18th Brazilian Symp. Comput. Games Digit. Entertain. (SBGames), Rio de Janeiro, Brazil, Oct. 2019, pp. 495–504. doi: https://doi.org/10.1109/sbgames.2019.00028. .
9. О. Трофименко, О. Задерейко, О. Янковський, В. Каіров, Г. Морозова, “Системний аналіз алгоритмів генерації лабіринтів для інтерактивних ігрових середовищ”, Кібербезпека: освіта, наука, техніка, т. 2, № 30, с. 259–279, 2025. doi: https://doi.org/10.28925/2663-4023.2025.30.972.
10. Wu, Z. Mao, Y. Li, Q. “Procedural game map generation using multi-leveled cellular automata by machine learning”. In: Proceedings of the 2nd International Symposium on Artificial Intelligence for Medicine Sciences. 2021. p. 168-172.. doi: https://doi.org/10.1145/3500931.3500962
11. M. Metkar, N. Sah, Y. Zhou, “Cellular Automata: From Structural Principles to Transport and Correlation Methods”. arXiv preprint arXiv:2601.01278, 2026. doi: https://doi.org/10.48550/arXiv.2601.01278
12. K. Vayadande, et al. “Simulation of Conway’s game of life using cellular automata”. Simulation, 2022
13. H. Sayama, C. L Nehaniv,. “Self-reproduction and evolution in cellular automata: 25 years after evoloops”. Artificial Life, 2024, 31.1: 81-95. doi: https://doi.org/10.1162/artl_a_00451
14. Joel Reyes, et al.” Game of sensors: An energy-efficient method to enhance network lifetime in wireless sensor networks using the game of life cellular automaton”. IEEE Access, 2022, 10: 129687-129701. doi: https://doi.org/10.1109/ACCESS.2022.3228585
15. Ю. Іляш. В. Горєлов, В.С. Антоняк , “Алгоритми клітинних автоматів для генерації рівнів у комп’ютерних іграх”. Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки, Том 37 (76) № 2 2026. Частина 1, с. 108 – 113 doi: https://doi.org/10.32782/2663-5941/2026.2.1/16
16. O. Zhezherun, B. Kalitovskyi, “Використання клітинних автоматів для вирішення задач фільтрації шумів та виявлення контурів зображень”. Наукові записки НаУКМА. Комп’ютерні науки, 2, 66–72. doi https://doi.org/10.18523/2617-3808.2019.2.66-72

Надійшла до редколегії 23.05.2026

Відомості про авторів:

Іляш Юрій Юрійович – кандидат технічних наук, доцент, завідувач кафедри комп’ютерних наук та інформаційних систем, Карпатський національний університет імені Василя Стефаника / Vasyl Stefanyk Carpathian National University, Україна; email: yurii.iliash@cnu.edu.ua; ORCID: https://orcid.org/0000-0003-3724-4183

Козачок Діана Андріївна – студентка кафедри комп’ютерних наук та інформаційних систем, Карпатський національний університет імені Василя Стефаника / Vasyl Stefanyk Carpathian National University, Україна; email: diana.kozachok.24@pnu.edu.ua; ORCID: https://orcid.org/0009-0009-9312-9973