УДК 519.21:004.421.2
DOI DOI https://doi.org/10.32782/2663-5941/2026.2.1/16

Іляш Ю.Ю.
https://orcid.org/0009-0006-5786-7205
Карпатський національний університет імені Василя Стефаника

Горєлов В.О.
https://orcid.org/0000-0002-2106-8704
Карпатський національний університет імені Василя Стефаника

Антоняк В.С.
https://orcid.org/0009-0004-2076-5606
Карпатський національний університет імені Василя Стефаника

АЛГОРИТМИ КЛІТИННИХ АВТОМАТІВ ДЛЯ ГЕНЕРАЦІЇ РІВНІВ У КОМП'ЮТЕРНИХ ІГРАХ

Стаття присвячена дослідженню алгоритмів процедурної генерації ігрових рівнів у 2D roguelike-іграх на основі клітинних автоматів та методів морфологічної обробки двовимірних дискретних структур. Обґрунтовано актуальність використання стохастичних і детермінованих алгоритмічних підходів для формування варіативних ігрових середовищ, що забезпечують високу реіграбельність, адаптивність та масштабованість ігрового процесу. Розглянуто особливості застосування клітинних автоматів із використанням околу Мура та вплив параметрів ініціалізації (щільність заповнення, кількість ітерацій згладжування) на топологічні характеристики згенерованих карт.

Запропоновано модульну архітектуру генератора рівнів, яка включає початкову стохастичну ініціалізацію карти, ітеративне згладжування за правилом сусідства Мура, видалення ізольованих регіонів за допомогою алгоритму пошуку в глибину, побудову коридорів між кімнатами на основі графових зв'язків та морфологічне розширення прохідних областей. Додатково враховано можливість генерації другорядних об'єктів ігрового середовища на основі тих самих принципів клітинних автоматів, що забезпечує структурну узгодженість рівня.

Практичну реалізацію виконано в рушії Unity мовою C# із використанням Tilemap-системи. Проте, завдяки універсальності підходу, подані алгоритми легко адаптуються до інших мов програмування та середовищ розробки, зокрема C++. Показано, що поєднання клітинних автоматів із додатковими топологічними та морфологічними операціями дозволяє формувати природоподібні печерні структури, характерні для жанру roguelike, при збереженні повної відтворюваності результатів через використання параметра seed.

Проведений аналіз демонструє, що варіювання параметрів генерації безпосередньо впливає на зв'язність рівня, розмір і форму ігрових просторів, а також на баланс між відкритими зонами та перешкодами. Отримані результати можуть бути використані як основа для систем навчання з підкріпленням та інших методів машинного навчання, що працюють із процедурно згенерованими середовищами та як базис для подальших досліджень із використанням методів машинного навчання в генеративному геймдизайні.

Ключові слова: *процедурна генерація, клітинні автомати, морфологічні операції, Unity, Tilemap, C#, C++, компонент зв'язності, алгоритм, згладжування.*

Постановка проблеми. Сучасні тенденції розвитку комп'ютерних ігор засвідчують, що користувачі очікують не лише цікавий сюжет, або красиву графіку, а насамперед – варіативності в методах проходження гри, та в отриманні кожного разу унікального ігрового досвіду.

Представники жанру roguelike, характеризуються високим рівнем варіативності ігрового процесу, що досягається шляхом використання процедурної генерації контенту (Procedural Content Generation, PCG) – автоматичне створення ігрових рівнів, які повинні відповідати низці вимог: забез-

печувати зв'язність простору, логічну структуру, збалансовану складність, можливість навігації гравця та коректне розміщення ігрових об'єктів (ворогів, предметів тощо).

При використанні випадкових методів генерації без додаткової структурної обробки виникає низка проблем. Зокрема, карта може містити ізольовані області, надто вузькі проходи, непридатні для ігрової механіки зони або геометрично некоректні конфігурації. Такі дефекти негативно впливають на ігровий процес та можуть призвести до ситуацій, коли рівень є непрохідним або порушує баланс гри.

Автори Лоуренс Джонсон, Юліан Торгеліус та Георгіс Яннакакіс в [1], та Метью Крайцер, Деніел Ешлок та Раджеш Перейра в [10] розглядають класичні алгоритми ручного або напівручного проєктування рівнів, такі як метод фіксованих шаблонів, алгоритми розбиття простору типу Binary Space Partitioning (BSP), а також побудова рівнів на основі заздалегідь визначених кімнат і коридорів забезпечують повний контроль над структурою простору. Однак описані алгоритми є ресурсоємними та не дозволяють створювати велику кількість унікальних варіантів. Таким чином, виникає необхідність у розробці алгоритмічного підходу, який поєднує випадковість із формальними правилами структурної обробки.

Особливо актуальною є задача формування природоподібних печерних структур, характерних для roguelike-ігор, із забезпеченням топологічної зв'язності всієї карти. Додатково постає проблема інтеграції алгоритму генерації в середовищ розробки із можливістю масштабування, модифікації параметрів та детермінованого відтворення результатів через використання початкового зерна випадковості (seed).

Аналіз останніх досліджень і публікацій. Процедурна генерація контенту (PCG) у іграх визначається як автоматична або комп'ютерно-підтримувана генерація контенту – рівнів, ландшафтів, предметів, правил, квестів тощо. Нур Шейкер, Юліан Тогеліус та Марк Джуніор Нельсон підкреслюють, що PCG охоплює різні класи алгоритмів – фрактальні, граматичні, пошукові та еволюційні методи, методи на основі обмежень, а також генерацію наративу, ландшафтів і підземель [1]. Серед цих підходів клітинні автомати (далі по тексту КА) є простим конструктивним методом генерації рівнів, здатним створювати лабіринтоподібні структури доволі простою схемою правил.

Лоуренс Джонсон, Юліан Торгеліус та Георгіс Яннакакіс [1,2] продемонстрували ефективність застосування КА для генерації печерних рівнів у реальному часі. Алгоритм базується на випадковій ініціалізації сітки та подальшому ітеративному застосуванні порогового правила, що дозволяє регулювати щільність стін і ширину тунелів. Попри низьку обчислювальну складність, базовий підхід потребує додаткових процедур забезпечення зв'язності, що свідчить про необхідність постобробки структури.

У роботі Юрія Іляша, Віталія Горєлова та Віктора Ровінського [3] розглянуто алгоритми та методи процедурної генерації ігрового контенту та адаптивної складності рівнів у 2D-платформерах. Автори пропонують підхід, який орієнтований переважно на відкриті або платформні простори та враховує зміну складності відповідно до поведінки гравця. Водночас така модель менш придатна для формування замкнутих печероподібних структур із жорсткими вимогами до топологічної зв'язності.

У працях Ендрю Печа, Філіпа Хінгстона, Мартіна Масека та Чіу Пен Лама [4,5] досліджено еволюційний підбір правил клітинних автоматів для генерації рівнів із заданими властивостями. Запропоновано використання генетичних алгоритмів для автоматичного налаштування локальних правил або відтворення топології зразкових карт. Однак такі підходи є обчислювально складнішими, що знижує їх практичність для легковагових рушіїв і швидкої інтеграції у проєкти з обмеженими ресурсами.

У дослідженні Чада Адамса, Хірава Парекха та Сушила Луїса [6] запропонувано інтерактивну еволюцію правил КА з участю користувача. Алгоритм адаптується на основі суб'єктивної оцінки згенерованих рівнів, що підвищує персоналізацію процесу. Водночас залежність від участі користувача обмежує автономність генератора та ускладнює автоматизоване тестування.

Ізабелла Антонюк в [7] розглядає гібридний підхід, у якому КА поєднуються з попередньо заданим дизайнерським "скелетом" рівня. Це дозволяє зберігати контроль над глобальною структурою карти, використовуючи природність форм КА. Однак такий метод частково повертає процес до напівручного проектування та залежить від попередньої структурної схеми.

У вітчизняному дослідженні проведеному Оленою Трофименко, Олександром Задарейком та співавторами [8], КА розглядаються як окремий клас решіткових алгоритмів PCG поряд із шумовими та графовими методами. Підкреслюється їх здатність формувати органічні структури при

низькій обчислювальній складності, однак зазначається проблема ізольованих областей і необхідність додаткових механізмів контролю.

Лазарос Лазарідіс та Джордж Ф. Фрагіліс [9] запропонували удосконалений конструктивний алгоритм на базі КА з автоматичним розміщенням кімнат і перевіркою колізій. Підхід забезпечує високу варіативність і придатність карт до використання без додаткового тестування. Водночас алгоритм орієнтований на модульну кімнатну структуру, що відрізняється від безперервних печероподібних просторів, характерних для roguelike-середовищ.

У праці Метью Крайцера, Деніела Ешлока та Раджеша Перейра [10] досліджено застосування клітинних автоматів для генерації ігрових рівнів із подальшим автоматизованим оцінюванням їх ігрових властивостей. Автори акцентують увагу на поєднанні конструктивної генерації з метриками придатності рівня (прохідність, зв'язність, просторовий баланс), що дозволяє формалізувати критерії якості. Підхід демонструє можливість інтеграції КА у більш комплексні системи контролю якості контенту, однак передбачає додатковий етап аналітичної перевірки, що ускладнює імплементацію підходу в прості проекти.

Огляд сучасних досліджень свідчить про зростання інтересу до інтеграції різноманітних алгоритмічних підходів у системи процедурної генерації контенту. Більшість робіт поєднують стохастичні механізми з еволюційними, графовими або адаптивними методами для підвищення варіативності та керованості структури рівнів. Водночас існує потреба у простому, обчислювально ефективному та параметризованому алгоритмі, здатному формувати замкнуті 2D-простори з гарантованою зв'язністю без складної еволюційної оптимізації або ручного втручання. Саме на вирішення цієї задачі спрямовано запропонований у даній роботі підхід.

Постановка завдання. Таким чином, розробка алгоритмів процедурної генерації рівнів на основі клітинних автоматів є перспективним напрямом у сфері створення ігрових середовищ, зокрема для жанру roguelike. Поєднання стохастичних методів ініціалізації з формальними правилами еволюції клітинних структур дозволяє формувати природоподібні, структурно зв'язні та варіативні карти з можливістю масштабування складності. У цьому контексті створення модульного алгоритму генерації 2D-рівнів із використанням клітинних автоматів та подальшої морфологічної обробки може розглядатися як практична реалізація сучасних підходів до процедурного геймдизайну та основа для подальших досліджень у сфері адаптивних ігрових систем і навчальних середовищ.

Виклад основного матеріалу. У межах дослідження було реалізовано алгоритм процедурної генерації 2D-рівнів для гри жанру roguelike на основі клітинних автоматів із подальшою структурною обробкою отриманої карти та генерацією ігрового наповнення. На відміну від класичних підходів, що обмежуються лише локальними правилами трансформації клітин, запропонований алгоритм доповнено механізмами постобробки, перевірки зв'язності та модульною системою розміщення додаткових об'єктів. Це дозволяє формувати топологічно коректні карти з керованими параметрами щільності, складності та варіативності простору. Запропонований підхід забезпечує поєднання стохастичної різноманітності з контролем структурної цілісності, що підвищує природність сформованої геометрії рівня та придатність карт для практичного використання в ігровому середовищі. Рівень представлено у вигляді двовимірної дискретної сітки розміром W×H, де кожна клітина може перебувати в одному з двох станів: прохідна область або стіна. Таким чином, карта інтерпретується як бінарна матриця, що описує геометрію ігрового простору.

Початковий етап генерації полягає у стохастичному заповненні карти. Для кожної клітини визначається початковий стан відповідно до ймовірності r появи стіни: $P(S_0(x,y)=1)=r$,
де $S_0(x,y)$ – початковий стан клітини. Граничні клітини примусово ініціалізуються як стіни з метою формування замкненого простору. Для забезпечення відтворюваності результатів використовується початкове зерно випадковості (seed), що дозволяє отримувати однакову карту при повторному запуску алгоритму з однаковими параметрами.

У спрощеному вигляді програмна реалізація алгоритму за допомогою мови C# у середовищі Unity матиме такий вигляд: public static int[,] Generate(int width, int height, string seed, bool useRandomSeed, int randomFillPercent, int smoothPasses)

```
    {
    int[,] map = new int[width, height];
    RandomFillMap(map, width, height, seed, useRandomSeed,
randomFillPercent);
    for (int i = 0; i < smoothPasses; i++){
    SmoothMapStep(map);
}
    return map;
}
```

Подальше формування структури рівня здійснюється за допомогою клітинного автомата із сусідством Мура, яке враховує вісім навколишніх клітин. На кожній ітерації обчислюється кількість сусідніх клітин у стані «стіна», після чого застосовується порогове правило переходу:

$$S_{t+1}(x,y) = \begin{cases} 1, & N(x,y) \geq T \\ 0, & N(x,y) < T \end{cases}$$

де $N(x,y)$ – кількість сусідів зі станом «стіна», а T – порогове значення. Ітеративне застосування цього правила протягом декількох кроків призводить до згладжування випадкових шумових фрагментів і формування печероподібної структури, характерної для ігор жанру roguelike. Практичні експерименти показали, що вибір параметрів y та 4–6 ітерацій забезпечує збалансоване співвідношення між щільністю стін і відкритих просторів.

У спрощеному вигляді програмна реалізація алгоритму за допомогою мови C# у середовищі Unity матиме такий вигляд:

```
static void SmoothMapStep(int[,] map){
int width = map.GetLength(0);
int height = map.GetLength(1);
int[,] newMap = (int[,])map.Clone();
for (int x = 0; x < width; x++){
for (int y = 0; y < height; y++){
int count = GetSurroundingWallCount(map, x, y);
if (count > 4) newMap[x, y] = 1;
else if (count < 4) newMap[x, y] = 0;
else newMap[x, y] = map[x, y];
}
}
for (int x = 0; x < width; x++){
for (int y = 0; y < height; y++){
map[x, y] = newMap[x, y];
}
}
}
```

Архітектура генерації побудована за модульним принципом, де кожен компонент відповідає за окрему стадію побудови рівня. Кожен етап працює незалежно, використовуючи спільні параметри, що передаються через об'єкт MapGeneratorManager. Такий підхід забезпечує легку масштабованість і можливість заміни будь-якого модуля без впливу на решту системи.

Основні модулі системи:

CellularAutomata: основний модуль, який генерує карту на основі заданих параметрів (розмір карти, зерно випадковості, відсотка початкового заповнення та кількості ітерацій згладжування).

RegionProcessor: відповідає за пошук компонент зв'язності та фільтрацію малих областей.

RoomDetector: визначає регіони, які пройшли фільтрацію через модуль RegionProcessor.

CorridorBuilder: здійснює побудову тунелів між регіонами карти, у випадку якщо вони є достатньо великими, але не пов'язані одне з одним.

EnemySpawner: відповідає за створення ворогів у межах одного регіону.

ObstacleSpawner: відповідає за створення перешкод у межах одного регіону.

У процесі експериментального дослідження встановлено, що вибір параметра початкової щільності заповнення (randomFillPercent) та кількості ітерацій згладжування (smoothPasses) суттєво впливає на топологію карти.

За низьких значень коефіцієнта (менше 45%) формуються великі відкриті області з недостатньою структурною варіативністю. За надмірно високих значень (понад 55%) утворюється велика кількість дрібних ізольованих замкнених областей та непрохідних зон.

За відсутності згладжування карта має хаотичну структуру з великою кількістю дрібних артефактів та різких переходів між станами клітин. Така конфігурація може випадково створювати природні перешкоди, однак не забезпечує стабільної геометричної цілісності. Зі збільшенням кількості ітерацій згладжування відбувається агрегація стін у більші масиви та усунення поодиноких клітин, що формує більш плавні та природні контури простору. Водночас надмірна кількість ітерацій призводить до надмірного укрупнення структур і втрати варіативності карти.

Найбільш збалансована структура спостерігається у діапазоні 46–53%, за умови виконання 5–8 ітерацій згладжування. Саме в цьому інтервалі забезпечується поєднання природності форм, достатньої зв'язності та ігрової прохідності простору.

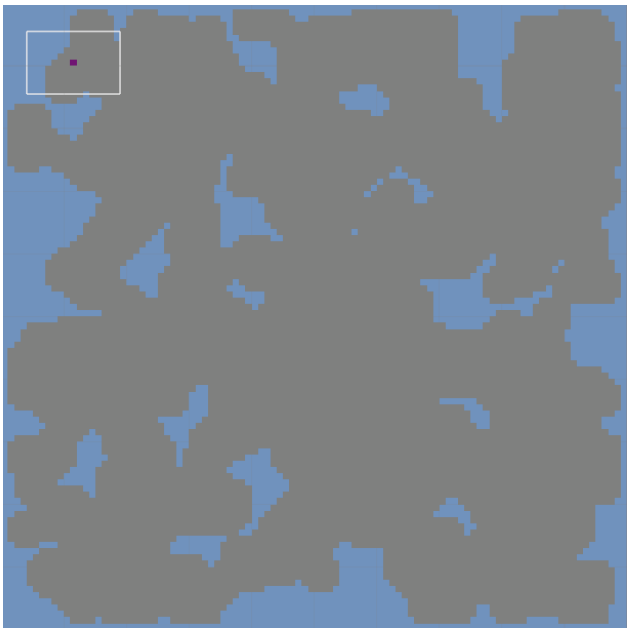

Рис. 1. Приклад згенерованої мапи, за низького значення параметра randomFillPercent

Таким чином, розроблений алгоритм поєднує стохастичну генерацію, ітеративну дискретну модель клітинного автомата та подальшу структурну обробку, що забезпечує формування варіативних, зв'язних і придатних до використання ігрових середовищ.

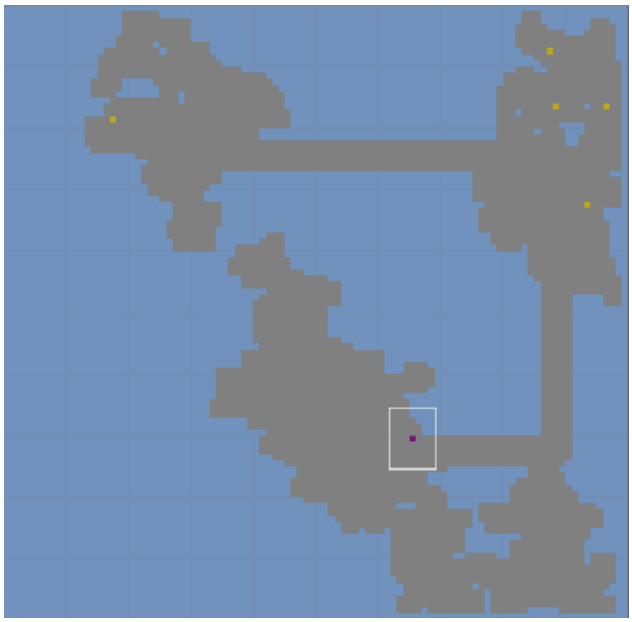

Рис. 2. Приклад згенерованої мапи, без використання згладжування

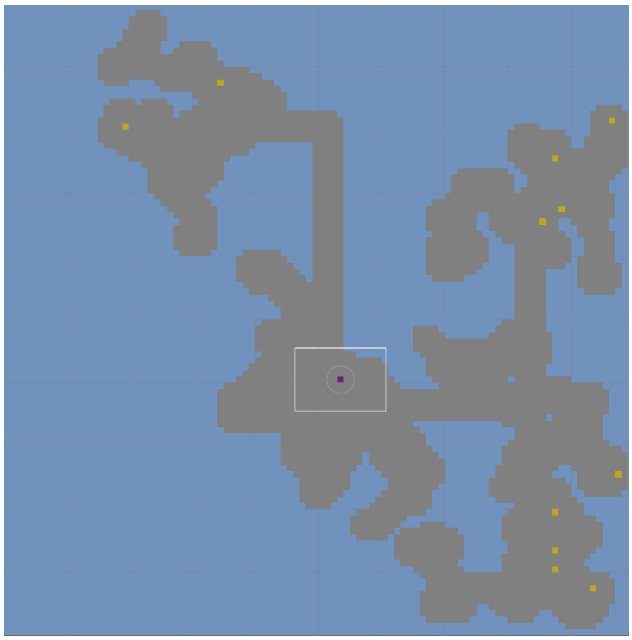

Рис. 3. Приклад згенерованої мапи з використанням згладжування

Запропонований алгоритм можуть бути використані в середовищах розробки інтерактивних застосунків та ігор, зокрема у Unity (C#), Unreal Engine (C++), Cocos2d-x (C++) та інших рушіях, що підтримують двовимірне або тайлове представлення простору. Завдяки параметризованій структурі генератора (розмір карти, відсоток початкового заповнення, кількість ітерацій згладжування) алгоритм може бути адаптований до різних жанрів – від roguelike та dungeon-crawler до навчальних або симуляційних систем.

Висновки. У роботі досліджено алгоритмічний підхід до процедурної генерації 2D-рівнів у комп'ютерних іграх жанру roguelike на основі клітинних автоматів із подальшою структурною та морфологічною обробкою простору, запропоновано доповнення та експериментально визначено оптимальні параметри вхідних даних. Запропоновано формалізовану модель генерації ігрового середовища у вигляді двовимірної дискретної сітки з бінарним представленням станів клітин, що дозволило описати процес формування рівня у термінах ітеративної еволюції локальних правил.

На відміну від підходів, що обмежуються виключно застосуванням клітинного автомата, запропонований алгоритм доповнено механізмами перевірки топологічної зв'язності, фільтрації ізольованих компонент, побудови коридорів між регіонами та модульною системою розміщення ігрових об'єктів. Це забезпечує не лише природоподібність геометрії, а й структурну цілісність та ігрову придатність згенерованих рівнів.

Експериментальний аналіз впливу параметрів початкової щільності заповнення та кількості ітерацій згладжування дозволив визначити діапазони значень, за яких досягається баланс між варіативністю, прохідністю та топологічною зв'язністю карти. Показано, що алгоритм має лінійну залежність обчислювальної складності від розміру сітки, що робить його придатним для використання в системах реального часу.

Практична реалізація в середовищі Unity підтвердила ефективність модульної архітектури генератора, яка забезпечує масштабованість, параметризованість і можливість детермінованого відтворення результатів через використання початкового зерна випадковості.

Отримані результати можуть бути використані як основа для побудови адаптивних систем процедурної генерації, інтеграції механізмів автоматичного балансування складності, а також для створення навчальних і тестових середовищ у задачах машинного навчання та навчання з підкріпленням. Перспективами подальших досліджень є розробка методів автоматичного підбору параметрів генерації, застосування еволюційних і нейромережевих підходів до оптимізації локальних правил клітинного автомата та розширення алгоритму для багаторівневих або тривимірних ігрових просторів.

Список літератури:

1. Shaker N., Togelius J., Nelson M. J. Procedural Content Generation in Games. Springer, 2016. P. 1-55. DOI: 10.1007/978-3-319-42716-4

2. Breno M. F. Viana, Selan R. dos Santos. Procedural Dungeon Generation: A Survey. Journal on Interactive Systems. 2021. Vol. 12. № 1. P. 83–101. DOI: 10.5753/jis.2021.999.

3. Іляш Ю.Ю., Горєлов В.О., Ровінський В.А. Алгоритми та методи процедурної генерації та адаптивної складності рівнів в іграх-платформерах. Вісник Херсонського національного технічного університету. 2025. № 4. С. 107–116. DOI: 10.35546/kntu2078-4481.2025.4.3.12

4. Pech A., Hingston P., Masek M., Lam P. Lecture Notes in Computer Science (Artificial Life and Computational Intelligence). Springer, 2015. Vol. 8955. P. 112–124. DOI: 10.1007/978-3-319-14803-8_9

5. Pech A., Hingston P., Masek M., Lam P. Game level layout generation using evolved cellular automata. Connection Science. 2016. Vol. 28. № 1. P. 63–82. DOI: 10.1080/09540091.2015.1130020

6. Adams C., Parekh H., Louis S. J. Procedural level design using an interactive cellular automata genetic algorithm. Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). 2017. P. 85–86. DOI: 10.1145/3067695.3075614

7. Antoniuk I. Generating layout for complex cave-like levels with schematic maps and Cellular Automata. Machine Graphics and Vision. 2023. Vol. 32. № 2. P. 45–64. DOI: 10.22630/MGV.2023.32.2.3

8. Трофименко О.Г., Задрейко О.В., Янковський О.Г., Каіров В. О., Морозова Г.С. Системний аналіз алгоритмів генерації лабіринтів для інтерактивних середовищ. Кібербезпека: освіта, наука, практика. 2025. № 30. С. 259–279. DOI: 10.28925/2663-4023.2025.30.972

9. Lazaridis L., Fragulis G. F. Creating a Newer and Improved Procedural Content Generation (PCG) Algorithm with Minimal Human Intervention for Computer Gaming Development. Computers. 2024. Vol. 13. № 11. P. 155–175. DOI: 10.3390/computers13110304

10. Kreitzer M., Ashlock D., Pereira R. (2019). Automatic Generation of Diverse Cavern Maps with Morphing Cellular Automata. 2019 IEEE Conference on Games (CoG). 2019. P. 933–940. DOI: 10.1109/CIG.2019.8847947

Iliash Yu.Yu., Horielov V.O., Antoniak V.S. CELLULAR AUTOMATION ALGORITHMS FOR LEVEL GENERATION IN COMPUTER GAMES

This article is devoted to the study of algorithms for the procedural generation of game levels in 2D roguelike games based on cellular automata and methods of morphological processing of two-dimensional discrete structures. The relevance of using stochastic and deterministic algorithmic approaches to generate variable game environments that ensure high replayability, adaptability, and scalability of the game process is substantiated. The paper examines the specifics of applying cellular automata using Moore's neighborhood and the influence of initialization parameters (filling density, number of smoothing iterations) on the topological characteristics of the generated maps.

A modular level generator architecture is proposed, which includes initial stochastic map initialization, iterative smoothing according to Moore's neighborhood rule, removal of isolated regions using a depth-first search algorithm, construction of corridors between rooms based on graph connections, and morphological expansion of passable areas. Additionally, the ability to generate secondary game environment objects based on the same cellular automaton principles has been incorporated, ensuring structural consistency within the level.

The practical implementation was carried out in the Unity engine using the C# language and a tilemap system. However, thanks to the versatility of the approach, the presented algorithms can be easily adapted to other programming languages and development environments, including C++. It is shown that combining cellular automata with additional topological and morphological operations allows for the generation of nature-like cave structures characteristic of the roguelike genre, while maintaining full reproducibility of results through the use of a seed parameter.

The analysis demonstrates that varying the generation parameters directly affects the level's connectivity, the size and shape of game spaces, as well as the balance between open areas and obstacles. The results obtained can be used as a foundation for reinforcement learning systems and other machine learning methods that work with procedurally generated environments, as well as a basis for further research on the use of machine learning methods in generative game design.

Keywords: *procedural generation, cellular automata, morphological operations, Unity, Tilemap, C#, C++, connectivity component, algorithm, smoothing.*

Дата першого надходження статті до видання: 19.03.2026
Дата прийняття статті до друку після рецензування: 15.04.2026
Дата публікації (оприлюднення) статті 11.05.2026