УДК 004.925.8:004.932.72

МЕТОДИ КОНТРОЛЮ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ РЕЛЬЄФІВ

CONTROL METHODS FOR PROCEDURAL TERRAIN GENERATION

Кудрявцев А.В.
Kudriavtsev A.V.
Чорноморський національний університет імені Петра Могили, м. Миколаїв, Україна
ORCID: https://orcid.org/0009-0008-7935-5678
Email: extosis.vt@gmail.com

DOI: 10.15673/atbp.v17i3.3261

Анотація. *У роботі здійснено дослідження методів контролю процедурної генерації фрактальних поверхонь, що є актуальною проблемою сучасної комп’ютерної графіки та ігрової індустрії. У якості об’єкта експерименту обрано моделювання кратеру, який, попри видиму простоту, характеризується достатнім рівнем структурної складності, що дозволяє продемонструвати ключові принципи процедурної генерації та перевірити ефективність методів керування. Особлива увага приділяється алгоритмам, заснованим на принципах стохастичних фракталів, серед яких виокремлюється алгоритм Diamond-Square, що забезпечує баланс між випадковістю та передбачуваністю і є одним із найпоширеніших у практиці створення карт висот. У роботі розглянуто та експериментально досліджено різні підходи до контролю, зокрема обмеження процесу генерації контрольними поверхнями, використання частково направленої рандомізації, застосування ефекту стохастичної інтерполяції, а також авторський «пружинний метод». Кожен із методів проаналізовано з точки зору можливостей наближення згенерованої структури до бажаної форми, рівня збереження випадковості, а також стійкості до появи артефактів. Окремо розглянуто характерні недоліки алгоритму Diamond–Square, зокрема утворення специфічних сіткових артефактів, а також наведено способи їх мінімізації шляхом комбінації з контрольними поверхнями та введення додаткових елементів рандомізації. Отримані результати показують, що поєднання контрольованих підходів дозволяє досягати оптимального балансу між керованістю та природністю рельєфу. Результати можуть бути застосовувані для вдосконалення генераторів ландшафтів у комп’ютерних іграх, анімаційних фільмах та інших сферах, де важлива як реалістичність, так і контрольована випадковість рельєфу, відкриваючи нові перспективи для розвитку процедурних методів генерації.*

Abstract. *The paper presents a study of methods for controlling procedural generation of fractal surfaces, which is a relevant issue in modern computer graphics and the gaming industry. As the object of the experiment, crater modeling was chosen, which, despite its apparent simplicity, has a sufficient level of structural complexity to demonstrate key principles of procedural generation and to test the effectiveness of control methods. Special attention is paid to algorithms based on the principles of stochastic fractals, among which the Diamond–Square algorithm stands out as it provides a balance between randomness and predictability and is one of the most widely used in the practice of heightmap generation. The paper considers and experimentally investigates various control approaches, including limiting the generation process with control surfaces, applying partially directed randomization, using the effect of stochastic interpolation, as well as the author’s “spring method”. Each method is analyzed in terms of its ability to approximate the generated structure to the desired form, the degree of randomness preservation, and resistance to artifact occurrence. Particular attention is given to the inherent drawbacks of the Diamond–Square algorithm, such as the formation of specific grid artifacts, as well as possible ways of minimizing them through the combination with control surfaces and the introduction of additional elements of randomization. The obtained results show that the combination of controlled approaches makes it possible to achieve an optimal balance between controllability and the natural appearance of the relief. The findings can be applied to improving landscape generators in computer games, animated films, and other areas where both realism and controlled randomness of the terrain are important, opening new perspectives for the development of procedural generation methods.*

Ключові слова: Процедурна генерація, фрактальна поверхня, diamond–square, криві Безьє, стохастична інтерполяція, генерація ландшафтів, комп'ютерна графіка, пружинний метод, TMF алгоритм.

Key words: Procedural generation, fractal surface, diamond–square, Bézier curves, stochastic interpolation, landscape generation, computer graphics, spring method, TMF algorithm.

Вступ. Алгоритми процедурної генерації – це тип комп'ютерних алгоритмів, що автоматично створюють контент у реальному часі відповідно до заданих правил, коефіцієнтів і обмежень. На відміну від зумовленого контенту, коли художник або дизайнер вручну визначає кожен об'єкт і всі параметри створюваного контенту, при процедурної генерації результат може бути в деякій мірі непередбачуваний і при кожному запуску може генерувати давати контент який відрізняється від попередньо-згенерованого. Одна з ключових переваг процедурної генерації - це скорочення витрат розробки, різних комерційних продуктів. У ігровій індустрії процедурна генерація підвищує рівень дизайну та вдосконалює основні ігрові системи.

Однак в розумінні і застосуванні процедурної генерації легко зробити помилки: дуже важливо зрозуміти, що це не є інструментом для вирішення всіх проблем. Їй можна користуватися для отримання безлічі цифрового контенту, або ж для внесення елемента випадковості в об'єкти, які довго і важко робити вручну. Написання та тестування алгоритмів часозатратна операція, особливо, коли вони взаємодіють з іншими системами додатку. До того ж, процедурна генерація – це не тільки засіб створення контенту. Процедурно генерований контент може служити самим різним цілям: будь то адаптивна анімації, нескінченні ігрові рівні, безшовні текстури, нескінченна реіграбельність, інтуїтивно зрозумілі системи, адаптивна музика або що завгодно ще.

Аналіз літературних даних і постановка проблеми. Алгоритми процедурної генерації зазвичай мають певні обмеження щодо можливості точного контролю за створюваним контентом, що ускладнює генерацію конкретних структур на кшталт того, як це реалізовано у генеративних моделях штучного інтелекту. У більшості випадків результат визначається випадковістю або потребує додаткового додавання заздалегідь підготовлених елементів після завершення основного процесу генерації. Так, в [1] запропоновано використання дескрипторів ландшафту для синтезу, аналізу та симуляції складних поверхонь. Автори роблять акцент на багаторівневих характеристиках рельєфу, проте їхній підхід більше орієнтований на узагальнений опис та аналіз, ніж на контрольовану генерацію локальних структур. У дослідженні [2] представлено метод NNProc, який наближує процедурні моделі 3D-об'єктів за допомогою нейронних мереж. Такий підхід дозволяє відтворювати складні геометричні форми, але він зосереджений переважно на апроксимації готових моделей, а не на контролі стохастичної генерації рельєфу в режимі реального часу. Робота [3] присвячена використанню алгоритму WaveFunctionCollapse для генерації рельєфів на основі даних дистанційного зондування. Метод демонструє потенціал для відтворення великих ландшафтів із реальних даних, проте основний фокус дослідження спрямований на використання вхідних карт висот, а не на гнучке керування параметрами генерації. В [4] автори систематизують сучасні підходи до процедурної генерації контенту в іграх і підкреслюють інтеграцію великих мовних моделей (LLM) як перспективний інструмент для підвищення керованості. Проте огляд зосереджується на геймдизайні й не надає детального аналізу специфічних алгоритмів генерації рельєфу. У роботі [5] дослідники пропонують неявні нейронні представлення для створення й редагування складних поверхонь. Підхід забезпечує неперервне відтворення даних і можливість інтерактивного дослідження, але він залишається ближчим до задач реконструкції, ніж до класичної процедурної генерації на основі стохастичних алгоритмів.

Проведений огляд показує, що більшість сучасних робіт спрямовані або на розширення описових можливостей моделей рельєфу, або на інтеграцію нейронних підходів для підвищення реалістичності. Однак питання контрольованості стохастичної генерації поверхонь, мінімізації артефактів та поєднання детермінованих і випадкових компонентів залишаються відкритими. Саме ці аспекти є предметом дослідження, де розглядаються методи обмеження генерації контрольними поверхнями, частково спрямована рандомізація та стохастична інтерполяція на прикладі моделювання кратерів. Ідея впровадження систем контролю за генерацією з'явилася у контексті спроб створення кратерів, які виступають об'єктом експерименту у цій статті. Попри відносну простоту структури кратера, яка не потребує складних алгоритмів ерозії чи їхньої імітації, він залишається достатньо складним, щоб формуватися у «випадковому» порядку, демонструючи основні принципи процедурної генерації та підкреслюючи важливість розробки методів контролю за результатами.

Мета і завдання дослідження. Метою даної роботи є дослідження методів контролю процедурної генерації поверхонь, які працюють з алгоритмами стохастичної інтерполяції та фрактальних підходів, зокрема алгоритму Diamond–Square. Робота спрямована на аналіз та порівняння різних способів керування генерації, як наприклад: контрольні поверхні та частково направлену рандомізацію – а також на оцінку їх ефективності з точки зору наближення до бажаної форми та мінімізації артефактів.

Окремою метою є розробка практичних рекомендацій щодо застосування цих методів для створення фрактальних поверхонь у цифрових моделях, комп'ютерних іграх та візуалізаційних системах, з можливістю регулювання варіативності та детальності поверхні.

Методи і матеріали досліджень. У контексті цієї статті алгоритми процедурної генерації представлені методами, що використовують принципи стохастичних фракталів/поверхонь, зокрема, Diamond-Square. Вони дозволяють отримувати самоподібні структури з випадковими відхиленнями, що є характерним для природних

форм рельєфу. Стохастичні фрактали поєднують у собі детерміновані закономірності та елементи випадковості – саме ця подвійність забезпечує природність і неоднорідність згенерованих поверхонь.

Завдяки цьому підходу можна моделювати різноманітні ландшафтні форми: від кратерів до гірських хребтів і долин. При цьому зберігається контроль над масштабними характеристиками й амплітудою нерівностей, без необхідності залучати складні алгоритми ерозії чи фізичної імітації процесів.

Алгоритм Diamond-Square, як найвідоміший приклад застосування стохастичних фракталів, дозволяє проводити більш керовану генерацію, забезпечуючи баланс між випадковістю та структурною організацією [6]. Незважаючи на те, що результати його роботи інколи поступаються за природністю аналогам, створеним на основі шуму Перліну (Perlin Noise) — це компенсується високою передбачуваністю форм і простотою налаштування параметрів.

Алгоритм Diamond-Square

Алгоритм «diamond-square» – це метод для генерації карт висот в комп'ютерній графіці. Його позиціонують як більш кращий алгоритмом, ніж тривимірна реалізація алгоритму «midpoint displacement», яка створює двовимірні ландшафти. Його також відомо як фрактал випадкового відхилення середини, хмарний фрактал чи фрактал плазми через ефект плазми, який він створює при застосуванні.

Ідею вперше представили Alain Fournier, Don Fussell та Loren Carpenter на конференції SIGGRAPH у 1982 році. Фрактальна поверхня яку він створює - поверхня, фрактальний об'єкт, який імітує зовнішній вигляд природної місцевості. Іншими словами, фрактальна поверхня з'являється не в результаті жорстко заданої процедури, а є, скоріше, псевдо-випадковим об'єктом, що володіє властивостями фрактала. Багато природніх об'єктів володіють в тій чи іншій формі властивістю статистичної самоподібності, яке може бути змодельоване за допомогою фрактальних поверхонь. Крім того, зміни текстури поверхні містять важливу інформацію про орієнтацію і нахил поверхні, а використання майже самоподібних об'єктів може допомогти змоделювати природні візуальні ефекти [6].

Алгоритм починає роботу з двовимірної сітки, потім, з чотирьох початкових, кутових значень, випадковим чином генерує карту висот, впорядковану у вигляді сітки з точок так, щоб вся площина була покрита квадратами, рис.1.

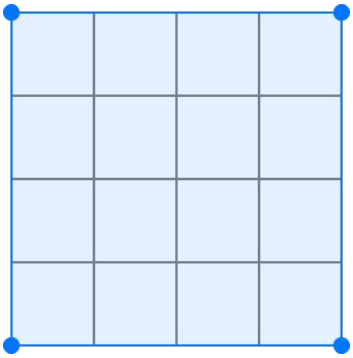

Рис. 1– Початкова сітка з чотирма опорними точками
Fig.1 – Initial grid with four reference points

Причому 2D сітка повинна бути розміром nxn, де $n = 2x + 1$, $x \in \mathbb{Z}$. У чотирьох кутових точках масиву встановлюються початкові значення – згенеровані псевдорандомні числа, рис.2.

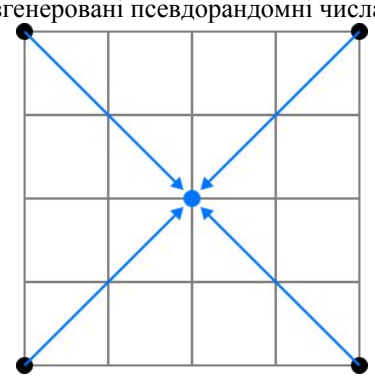

Рис. 2 – Обчислення центральної точки.
Fig.2 – Calculation of the central point

Крок «diamond». Для квадрата з чотирьох початкових значень, знаходиться центральна точка, в яку встановлюється середнє значення чотирьох кутових точок плюс псевдовипадкове значення (яке може бути від'ємним), рис.3.

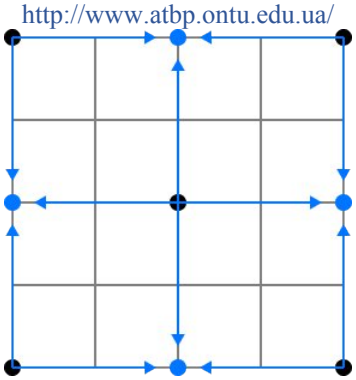

Рис. 3 – Обчислення чотирьох центральних точок
Fig.3 – Calculation of four central points

Крок «square». Для кожного ромба в масиві, встановлюється центральна точка, якій присвоюється середнє значення з чотирьох кутових точок плюс псевдовипадкове значення. У кроках «square», точки, розташовані по краях загального масиву будуть мати тільки три сусідніх значення а не чотири (четверта точка буде виходити за межі масиву), рис.4.

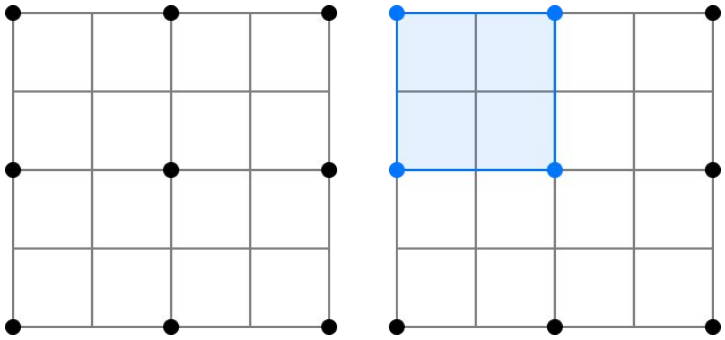

Рис. 4 – Результат першої ітерації
Fig.4 – First iteration result

Далі для отриманих чотирьох квадратів кроки «diamond» і «square» повторюються поки значення всіх точок не будуть знайдені. Алгоритм diamond-square – рекурсивний, умовою виходу є різниця х координат двох сусідніх точок яка дорівнює одиниці при чому $x_1 - x_0 = 1$, де $x_1 > x_0$.

При роботі цього алгоритму будуть виникати випадки подвійного визначення значення точок тому для правильної роботи слід помічати вже знайдені значення для того, щоб не розраховувати у другий раз – це може призвести до суттєвих візуальних артефактів. Також псевдовипадкове значення, яке додається на кожному кроці, потрібно тримати у змінному діапазоні з урахуванням відстані між опорними точками, щоб запобігти різкої зміни висоти суміжних точок, наприклад:

$$f(a_c) = \frac{a_0 + a_1 + a_2 + a_3}{4} + rand(x_1 - x_0) - \frac{(x_1 - x_0)}{2} \qquad (1)$$

де a_c – значення поточної точки, $a_0 - a_3$ значення опорних точок, x_0, x_1 – координати x опорних точок, *rand(max)* – деяка функція, яка вертає псевдовипадкове значення в інтервалі [0, *max*]

Алгоритм Triangle Mesh Fractal

Алгоритм Triangle Mesh Fractal базується на принципах роботи алгоритма «Diamond-Square», тобто він також є варіантом стохастичного фракталу, поверхні. Також, як і згаданий попередньо алгоритм, він імітує зовнішній вигляд природної поверхні та має єфект плазми, і може бути використаний для генерації складних ландшафтів, хмар тощо.

Алгоритм також починає роботу з двовимірної сітки, але на відміну від базового алгоритму, для цього алгоритму не потрібно мати чотири початкові точки для старту генерації, достатньо буде трьох для того щоб заповнити половину сітки значеннями та відтворити той самий процес для іншої половини щоб отримати повністю заповнену сітку, рис.5. Також нова процедура включає у себе меншу кількість кроків для створення подібного до базового алгоритму результату [7].

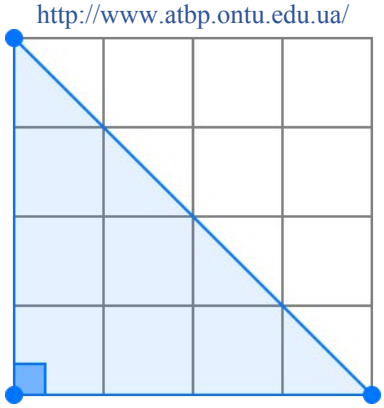

Рис. 5 – Початкова сітка з трьома опорними точками
Fig.5 – Initial mesh with three reference points

Залежність розмірів 2D сітки є такою самою і вона повинна бути розміром *nxn*, де $n = 2x + 1$, $x \in \mathbb{Z}$. У трьох кутових точках масиву встановлюються початкові значення які є згенерованими псевдорандомними числами, рис.6.

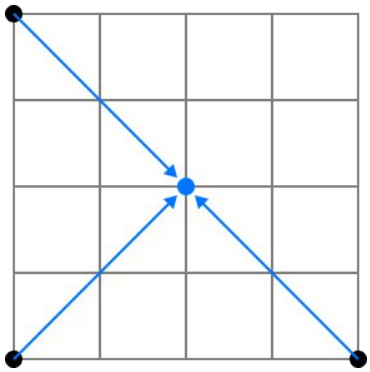

Рис. 6 – Обчислення центральной точки гіпотенузи
Fig.6 – Calculation of the central point of the hypotenuse

На наступному кроці обчислюється центральна точка найбільшої сторони трикутника, у даному випадку це гіпотенуза рівнобедреного, прямокутного трикутника. У визначену точку встановлюється середнє значення трьох вершин трикутника плюс псевдовипадкове значення, рис.7.

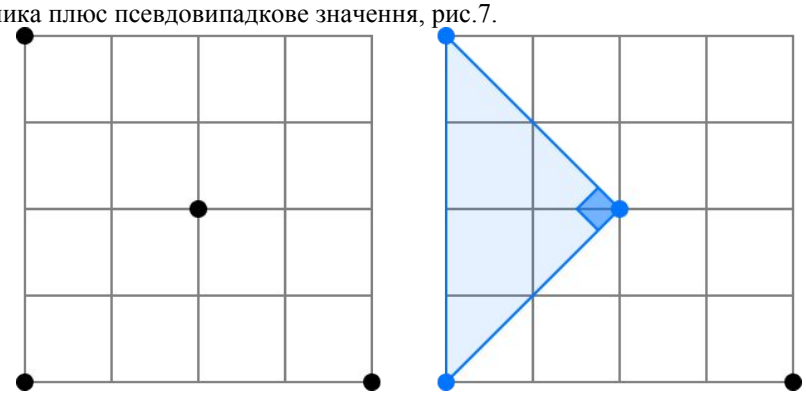

Рис. 7 – Результат першої ітерації
Fig.7 – Result of the first iteration

Знайдена точка поділяє попередній трикутник на два нових прямокутних, рівнобедрених трикутника, до яких застосовується та ж сама оперція доки значення різниці двох координат двох сусідніх точок не будуть дорівнювати одиниці: $x_1 - x_0 = 1$, де $x_1 > x_0$.

Формальне визначення кроку алгоритму не передбачає необхідності мати квадратну двовимірну сітку, але така сітка значно спрощує виконання алгоритму, бо у такому випадку відпадає необхідність шукати найбільшу сторону трикутника, вона завжди відома – це гіпотенуза, також центри цих, найбільших сторін будуть співпадати з сіткою, а тому для того щоб знайти правильні значення не доведеться використовувати методи інтерполяції.

При цьому, при роботі цього алгоритму будуть виникати випадки подвійного визначення значення точок, тому для правильної роботи слід помічати вже знайдені значення для точок, щоб не розраховувати їх у другий раз. Також псевдо-випадкове значення, яке додається на кожному кроці, потрібно тримати у змінному діапазоні з урахуванням відстані між вершинами найбільшої сторони трикутника, наприклад:

$$f(a_c) = \frac{a_0 + a_1 + a_2}{3} + rand(dist((x_0, y_0)(x_1, y_1))) - \frac{dist((x_0, y_0)(x_1, y_1))}{2} \quad (2)$$

де a_c – значення необхідної точки, $a_0 - a_2$ – значення вершин, x_0, y_0, x_1, y_1 – координати вершин, заданих парою (x, y), де $x, y \in Z$, $rand(max)$ – деяка функція, яка вертає псевдовипадкове значення в інтервалі [0, *max*], *dist*((*x*, *y*), (*x*, *y*)) – функція, яка повертає відстань між двома координатами точок.

Результати досліджень. Усі далі описані методи контролю генерації, у тій чи іншій мірі, використовують контрольну поверхню або контрольні поверхні як еталонну форму, у даному випадку – кратера, що дозволяє порівнювати отримані результати з ідеальною моделлю та коригувати процес генерації для досягнення бажаного вигляду.

Для наочної репрезентації результатів роботи застосовуються чорно-білі карти висот, які відображають рельєф поверхні у вигляді градацій сірого: світліші ділянки відповідають точкам, що розташовані вище; тоді як темніші тим, що розташовані нижче, рис.8. Такі карти висот, з деякою зручністю, дозволяють візуально оцінити топографію згенерованої структури, виявляти відхилення від контрольної форми та проводити подальшу оптимізацію алгоритмів процедурної генерації або методів їх контролю.

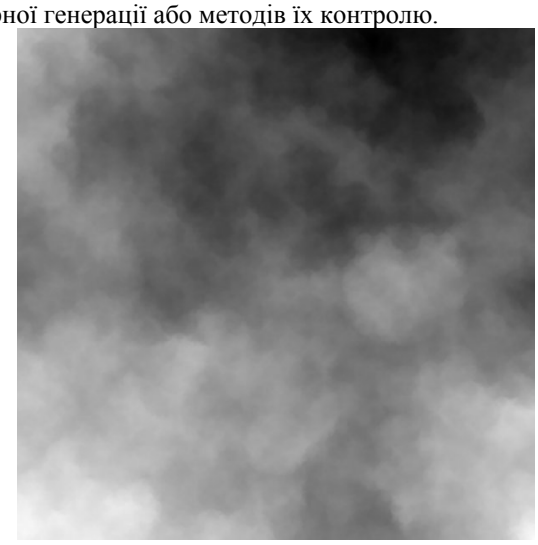

Рис. 8 – Приклад карти висот, побудованої за допомогою Triangle Mesh Fractal
Fig.8 – Example of a heightmap generated using the Triangle Mesh Fractal

Також, карти висот є зручним інструментом для обробки даних та симуляції, оскільки їх можна безпосередньо використовувати для моделювання текстур, освітлення та інших графічних чи фізичних обчислень, що підвищує практичну значущість застосованих методів візуалізації.

Для моделювання тривимірного вигляду поверхні використовується програмне забезпечення Blender. У контексті процедурної генерації застосування цього програмного забезпечення полягає у перетворенні згенерованих карт висот у тривимірні моделі поверхні, що дає змогу оцінювати топографію структури не лише у вигляді плоскої карти, а й у вигляді об'ємного представлення, з виразним рельєфом, деталями, переходами. Більше того, створені поверхні можна інтегрувати у більші сцени або симуляційні середовища.

Створення контрольних поверхонь

Для контрольних поверхонь застосовується підхід у вигляді функцій типу *f*(*x, y*), що дозволяють визначати значення в будь-якій точці двовимірної площини за координатами *x* та *y*. Контрольні поверхні виявились доволі комплексними об'єктами, оскільки при їх створенні застосовується кілька різних рівнів математичних абстракцій. Така багаторівнева структура дозволяє більш точно впливати на вигляд контрольної поверхні, а отже і на результат генерації.

Для створення такої контрольної поверхні застосовувався метод обертання функції виду *f(x)* навколо заданої центральної точки, або, як його ще називають, побудова поверхні за принципом радіальної симетрії, що дозволяє формувати об'ємні форми на основі одновимірного профілю. Такий підхід передбачає, що значення висоти у будь-якій точці поверхні визначається відстанню від центра, що забезпечує природну округлу симетрію, характерну для кратерів та інших подібних структур: $x = dist((x_i, y_i)(x_c, y_c))$, де (x_i, y_i) – вхідна точка, (x_c, y_c) – центральна точка.

Крива *f*(*x*) задається як кусочно-задана функція, де окремі сегменти визначаються окремо та незалежно, що дозволяє поєднувати різні функції для отримання потрібної форми кривої. Кожен сегмент формується за допомогою кривих Безьє, які задаються як $f(t) = (x(t), y(t)), t \in [0, 1]$ – параметр кривої Безьє, рис.9.

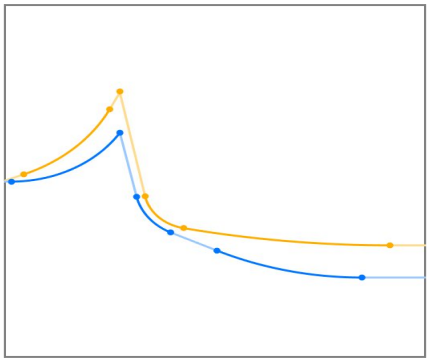

**Рис. 9 – Використані криві з визначеними сегментами,
насиченим кольором позначені кубічні криві Безьє, ненасиченим – лінійні**
**Fig.9 – The curves with defined segments: cubic Bézier curves are shown in solid color,
and linear ones in light color**

У роботі використовуються лінійні та кубічні криві Безьє, які формують сегменти кусочно-заданої функції. Опорні точки задані у нормалізованому вигляді, що дозволяє підлаштувати контрольну поверхню до необхідного розміру. Такий підхід дозволяє ефективно визначати значення функції у будь-якій точці сітки, забезпечуючи точний контроль форми кривих і надійне відтворення базових контурів поверхні. Перетворення кривих у вигляд $f(x)$ здійснюється з використанням методу бісекції для пошуку значення параметра t, яке найбільш точно відповідає вхідному значенню x – цей підхід застосовується для кубічних кривих, тоді як лінійні криві Безьє обробляються без додаткових обчислень, оскільки їхнє перетворення у звичайну лінійну функцію є тривіальною задачею [8]. Хоча застосований метод бісекції не є найоптимальнішим з точки зору обчислень, він був обраний через простоту реалізації та достатньо високу точність для задач моделювання, рис.10.

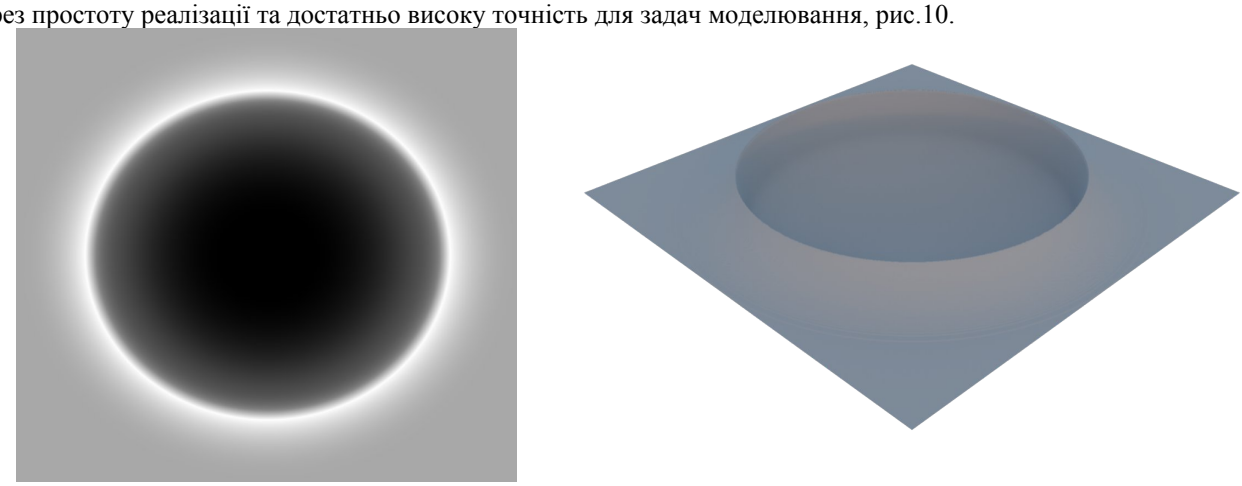

Рис. 10 – Карта висоти (ліворуч) та тривимірна модель (праворуч) нижньої контрольної поверхні
Fig.10 – Heightmap (left) and 3D model (right) of the lower control surface

Обмеження генерації контрольними поверхнями

У попередніх дослідженнях застосовуваних алгоритмів генерації використовувалися максимальні та мінімальні значення (у контексті карт висот – висоти), що дозволяло визначати область формування генерованих даних. Фактично такі обмеження являють собою дві поверхні виду $f(x, y) = const$, що визначають верхню та нижню межі. Проте ці плоскі поверхні можна узагальнити та замінити будь-якими іншими обмежувальними поверхнями виду $f(x, y)$, між якими і повинні виконуватися кроки алгоритму процедурної генерації.

Крок перевірки обмежень та корекції значення виконується для кожної згенерованої точки після всіх інших обчислень алгоритму, наприклад для алгоритму Diamond-Square цей крок відбувається після обчислення середнього арифметичного значень суміжних точок та додавання псевдовипадкового числа, що і визначає локальну варіативність рельєфу. Таким чином це дозволяє забезпечити відповідність кожної точки встановленим контрольним поверхням, зберігаючи при цьому стохастичну, псевдовипадкову природу процедурної генерації та можливість формування складних структур, одночасно обмежуючи значення у визначених межах для підтримки бажаного вигляду результуючої поверхні.

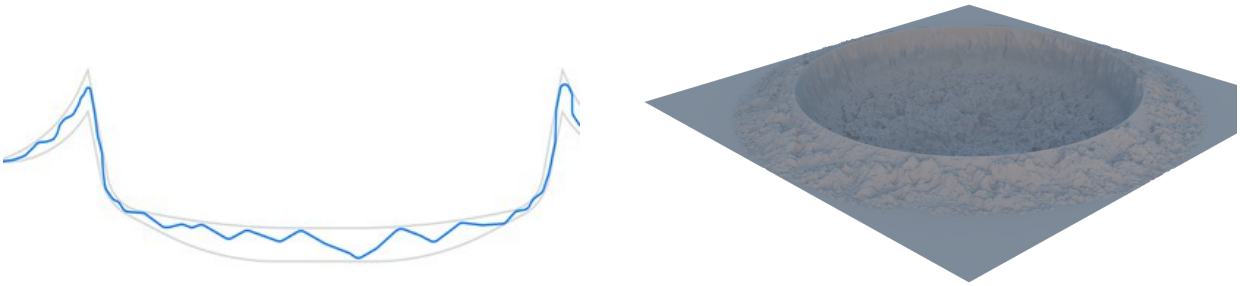

Рис. 11 – Приклад контрольованої генерації у розрізі (ліворуч) та тривимірна модель згенерованої поверхні (праворуч)
Fig.11. Example of controlled generation in cross-section (left) and 3D model of the generated surface (right)

Такий підхід забезпечує більш гнучке і водночас просте керування амплітудою та формою генерованого рельєфу, а також дозволяє використовувати складні контрольні поверхні, підвищуючи передбачуваність результату без втрати стохастичного характеру генерації.

Алгоритм ефективно виконує свою основну задачу, проте має певні недоліки через використання «ідеальних» контрольних поверхонь. Одним із таких недоліків є обмежена кількість хаотичних елементів у формуванні рельєфу: через обмежувальні контрольні поверхні, наприклад, неможливо відтворити руйнування стінки кратера, оскільки нижня поверхня просто не дозволить цього зробити. Крім того, діапазон псевдовипадкових значень залежить від відстані між точками, і часто він може перевищити проміжок між обмежувальними поверхнями, що призводить до значних коливань, у даному випадку, висот у рельєфі. Додатково, у тих місцях, де контрольні поверхні розташовані дуже близько одна до одної або під крутим кутом, алгоритм майже гарантовано «упирається» у ці обмеження, що обмежує природність і варіативність згенерованої структури. Таким чином, хоча алгоритм забезпечує достатньо високу точність у відтворенні заданих форм, він водночас демонструє обмежену гнучкість при моделюванні складних структур або структур з нестабільним характером висот.

Частково зазначені вище проблеми можна вирішити різними способами. Наприклад, з використанням лише однієї контрольної поверхні, тобто обмежувати роботу алгоритму лише знизу або лише зверху – це дозволяє досягти більшої варіативності рельєфу та надає алгоритму більше свободи у формуванні локальних нерівностей. Також, можна змінити спосіб обчислення максимального значення рандомізації, включивши у розрахунок відстань між верхньою та нижньою контрольними поверхнями, що дозволяє згладжувати різкі перепади та робить рельєф більш природним та прогнозованим. Додатково, доцільним є введення додаткової рандомізації, як окремого кроку, для випадків, коли результат наближується до значення контрольної поверхні, що дає можливість алгоритму частково «виходити» за межі встановлених обмежень і формувати більш реалістичні та різноманітні структури, не порушуючи загальної форми контрольної поверхні, рис.12.

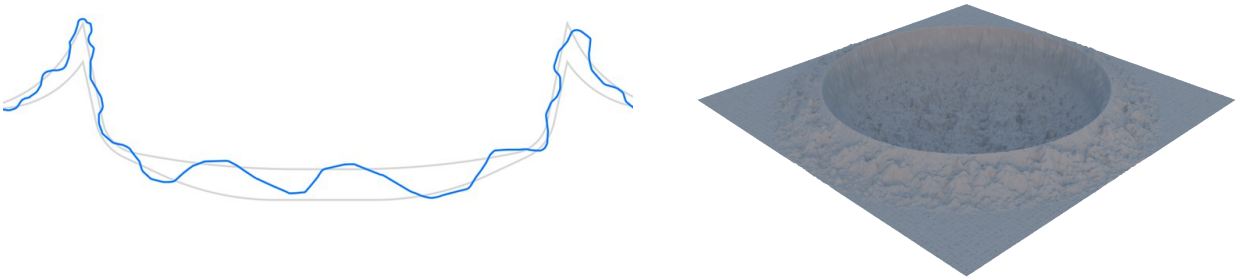

Рис. 12 – Приклад контрольованої генерації з додаткової рандомізацією у розрізі (ліворуч) та тривимірна модель згенерованої поверхні (праворуч)
Fig.12 – Example of controlled generation with additional randomization in cross-section (left) and 3D model of the generated surface (right)

Обмеження генерації

Іншим підходом до контролю генерації є застосування так званого «пружинного методу» (авторський термін), суть якого полягає в спробі направити, «притягнути» згенеровану поверхню до контрольної з силою, пропорційною відстані між ними. Ілюстративно це можна уявити так, ніби між кожною точкою згенерованої поверхні та відповідною точкою контрольної поверхні натягнута пружина, яку відпускають на певний проміжок часу, що і зумовлює поступове наближення точки до бажаної висоти. Використання цього методу реалізується так само як і попередній – як окремий крок після виконання основних обчислень алгоритму, дозволяючи

коригувати рельєф, згладжувати різкі перепади та забезпечувати більш точну відповідність контрольній поверхні, при цьому зберігаючи часткову випадковість та природність структури.

Алгоритм «пружинного методу» працює з однією контрольної поверхнею замість двох обмежувальних, що є його перевагою, оскільки це забезпечує контроль з меншими зусиллями та, теоретично, дозволяє досягти більшої варіативності рельєфу. В цілому алгоритм виконує свою задачу, проте він не позбавлений проблем. Основним недоліком є те, що функція, яка відповідає за силу притягування згенерованої точки до контрольної поверхні, є лінійною і фактично задається рівнянням $y = x$, що передбачає необхідність введення якогось коефіцієнта, який би визначав «тривалість» впливу цього зусилля, і якщо цей коефіцієнт буде дорівнювати одиниці – згенерована поверхня повністю співпаде з контрольної. Було здійснено спроби застосувати нелінійну функцію для притягування, проте такий підхід може призвести до небажаних ефектів: якщо значення функції перевищить відповідне значення лінійної функції $y = x$, то кінцеве значення точки може виявитися ще далі від контрольної поверхні, ніж до дії кроку пружинного методу.

Найбільшою проблемою також залишаються візуальні артефакти, які виникають через природу використаного алгоритму Diamond-Square та посилюються при застосуванні пружинного методу. Алгоритм Diamond-Square без кроку рандомізації фактично працює як окремий випадок лінійної рекурсивної інтерполяції, що призводить до утворення характерних чотирикінцевих «зірок» на поверхні – специфічних, для цієї інтерполяції, структур, від яких дуже складно позбутися через повторювану сіткову рекурсію, рис.12. Застосування спрямованої рандомізації на кожному кроці алгоритму додатково підсилює ці прояви: зміщення точок відповідно до контрольної поверхні разом із локально випадковими значеннями призводить до посилення контрасту артефактів, через що практично кожний результат генерації Diamond-Square із цим методом містить помітні візуальні дефекти.

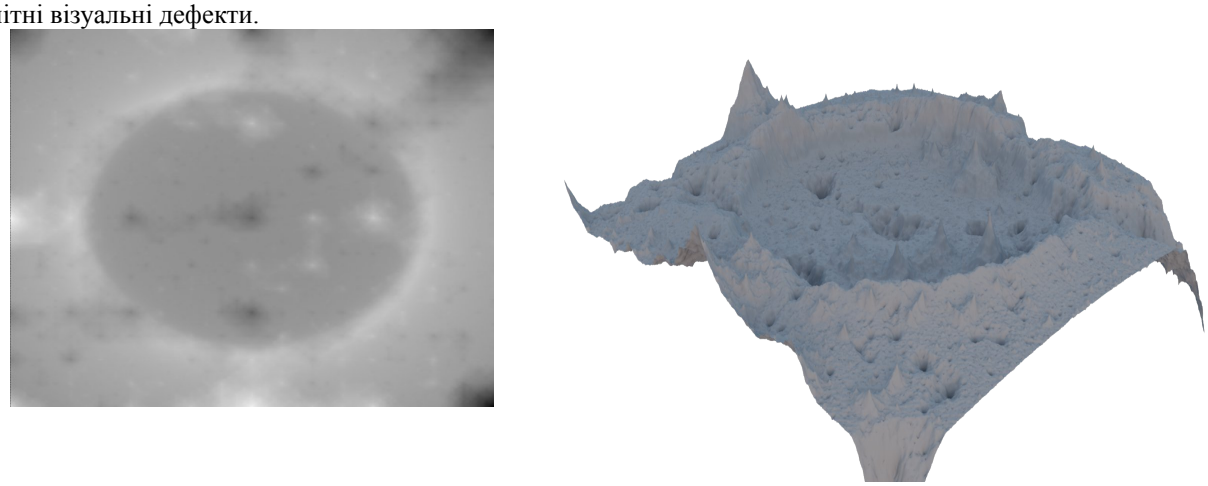

Рис. 13 – Гіперболізований приклад артефактів генерації при використанні пружинного методу (ліворуч) та згенерована поверхня з великою кількістю артефактів (праворуч)
Fig.13 – Exaggerated example of generation artifacts using the spring method (left) and a generated surface with a large number of artifacts (right)

Такі артефакти є наслідком як самої сіткової структури алгоритму, так і обмеженої гнучкості при корекції точок у кроці пружинного методу, що потребує додаткових підходів для згладжування поверхні та мінімізації небажаних ефектів.

Теоретично можливо використати інший, вже згаданий, алгоритм – Triangle Mesh Fractal (TMF algorithm). Під час його розробки був неусвідомлено порушений «принцип» лінійної рекурсивної інтерполяції, завдяки чому при відключенні рандомізації вихідні значення не збігаються з результатом багаторазово повторюваної лінійної інтерполяції. Через це цей алгоритм не має характерних чотирикінцевих артефактів, властивих для Diamond-Square, проте проявляє інший специфічний ефект у вигляді клубоподібних завихрень на поверхні, що є природним наслідком топології трикутної сітки та особливостей рекурсивного розбиття, і, ймовірно, їх інтенсивність посилиться при застосуванні пружинного методу, оскільки корекція точок із прив'язкою напряму генерації до контрольної поверхні може локально збільшувати кривизну та створювати концентровані хвильові структури на рельєфі.

Частково направлена рандомізація

Сама суть ідеї «пружинного методу» залишається дієвою, і вона полягає в наближенні згенерованого значення до значень контрольної поверхні, що дозволяє коригувати рельєф у потрібному напрямку. Водночас важливо уникати постійного застосування цього впливу, оскільки при безперервній корекції згенеровані значення зрештою повністю збігатимуться з контрольними, втрачаючи природну випадковість та варіативність. Такий підхід формує основу для частково направленої рандомізації, де контрольна поверхня виступає лише частковим орієнтиром.

Як приклад, можна частково гібридизувати крок додавання псевдовипадкових значень, встановивши умову, що якщо відстань між опорними точками перевищує відстань до контрольної поверхні, то слід генерувати псевдовипадкове значення пропорційно відстані до контрольної поверхні та додавати або віднімати це значення до обчисленого середнього значення опорних точок в залежності від положення контрольної поверхні, тим самим наближаючи результат до контрольного значення. У протилежному випадку, коли відстань між точками менше або дорівнює відстані до контрольної поверхні, генерується псевдовипадкове значення з урахуванням відстані між точками і додається до середнього значення опорних точок. Це дозволяє підтримувати варіативність рельєфу та уникати надмірної кореляції з контрольними значеннями. Такий підхід забезпечує баланс між керованістю та випадковістю, одночасно зберігаючи природність і динамічність згенерованої поверхні, рис.14.

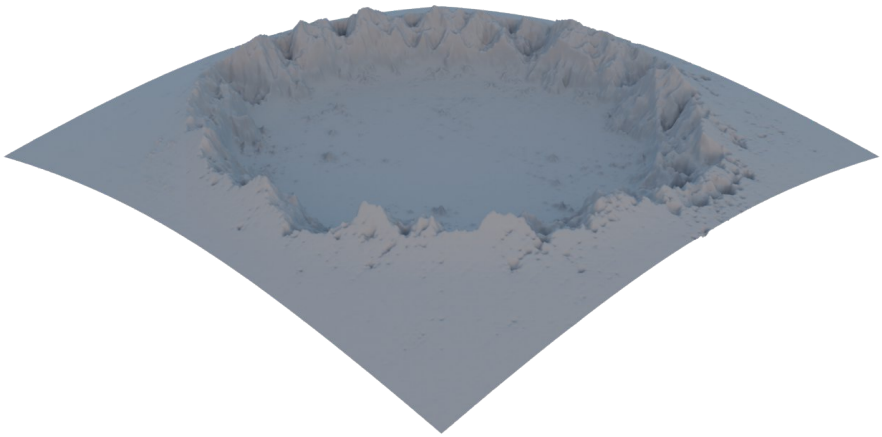

Рис.14 – Приклад згенерованої поверхні з частково направленою рандомізацією
Fig.14 – Example of a generated surface with partially directed randomization

Хоча цей метод, суб'єктивно, дає досить хороші результати та дозволяє досягти вискої різноманітності поверхні з дотриманням основних форм, він залишається доволі нестабільним і часто потребує кількох циклів генерації для отримання бажаного результату. Хоча такий підхід і є перспективним, оскільки поєднує керованість та варіативність рельєфу, проте його правильна реалізація та використання складні, тому цей метод потребує додаткового доопрацювання, щоб підвищити стабільність та покращити кінцеві результати генерації.

Стохастична інтерполяція

У строгому сенсі застосовані алгоритми не є класичною стохастичною інтерполяцією, проте все ж проявляють її властивості, а саме: інтерполяція передбачає обчислення невідомих значень на основі відомих; стохастичний компонент додає псевдовипадкове значення, створюючи характерну випадковість.

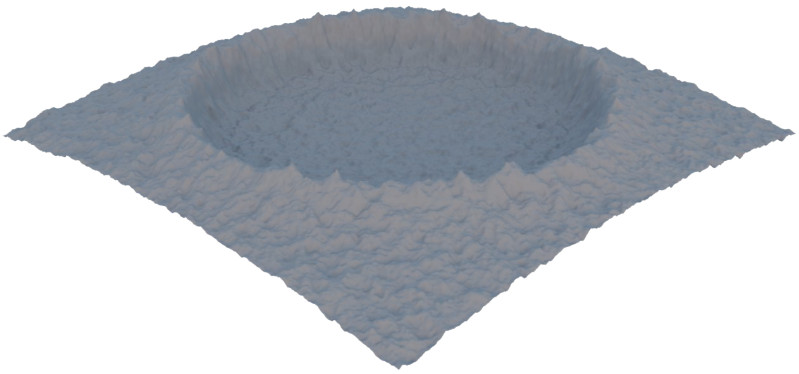

Рис. 15 – Згенерована поверхня методом стохастичної інтерполяції
Fig.15 – Generated surface using the stochastic interpolation method

Саме цими властивостями і можна ефективно скористатися для контролю генерації. Ідея є досить простою: на початкових ітераціях рекурсії алгоритму замість звичних кроків обчислення значень точок використовуються безпосередньо значення контрольної поверхні, що задає базову форму рельєфу. На пізніших ітераціях застосовуються стандартні кроки алгоритму для заповнення та обчислення проміжних значень між фіксованими точками, отриманими з контрольної поверхні на попередніх ітераціях. Такий підхід дозволяє поєднувати керованість генерації на рівні основної топології та збереження фрактальної варіативності на дрібних деталях рельєфу, забезпечуючи баланс між структурованістю та псевдовипадковістю кінцевої поверхні, рис.15.

Стохастична інтерполяція є доволі прийнятним способом контролю генерації, оскільки дозволяє зберегти базову форму рельєфу, проте кінцевий результат потребує доопрацювання. Основними проблемами є необхідність ручного підбору оптимальної кількості ітерацій перед початком фази стохастичної інтерполяції, оскільки їхня кількість безпосередньо залежить від розміру сітки генерації. Значення цієї величини не може бути близьким до нуля або до максимального числа ітерацій: занадто ранній початок фази стохастичної інтерполяції призведе до того, що контрольна поверхня фактично не буде впливати на рельєф, надто пізній – і, крім значень контрольної поверхні, нічого не буде згенеровано. Також через особливості алгоритму та обмеження сітки округлі структури, як у випадку з кратером, формуються з певними спотвореннями форм. Крім того, хоч ця проблема і менш помітна, частина точок, перенесених з контрольної поверхні, завжди залишається однаковою, що зменшує загальну випадковість і варіативність згенерованої поверхні.

Висновки. Дослідження продемонструвало, що контроль алгоритмів процедурної генерації є цілком можливим, і існують різні підходи для реалізації цього. Було вивчено кілька методів контролю, включно з обмеженням значень за допомогою контрольних поверхонь, застосуванням «пружинного методу», частково направленою рандомізацією та використанням стохастичної інтерполяції. Для кожного з окреслених методів охарактеризовано переваги та обмеження.

Метод обмеження контрольними поверхнями показав свою простоту та надійність, дозволяючи ефективно регулювати межі генерованих значень і забезпечувати відповідність базовій формі рельєфу. Частково направлена рандомізація, хоча і складніша в реалізації, дає цікаві результати, дозволяючи поєднувати збереження основної форми та варіативність дрібних деталей поверхні, що робить її перспективною для подальшого розвитку.

Інші підходи, такі як «пружинний метод» або стохастична інтерполяція, хоча й мають певні переваги, виявилися більш складними та нестабільними у використанні, потребуючи ручного налаштування параметрів та додаткових циклів обчислень, що знижує їх практичну ефективність.

У комплексі проведене дослідження показує, що що поєднання контрольних поверхонь із частково направленою рандомізацією забезпечує оптимальний баланс між керованістю генерації та природною варіативністю фрактального рельєфу.

Результати можуть бути застосовувані для вдосконалення генераторів ландшафтів у комп'ютерних іграх, анімаційних фільмах та інших сферах, де важлива як реалістичність, так і контрольована випадковість рельєфу, відкриваючи нові перспективи для розвитку процедурних методів генерації.

Список використаних джерел

1. Argudo O., Guérin E., Schott H., Galin E. **Terrain descriptors for landscape synthesis, analysis and simulation.** Computer Graphics Forum, vol. 44, No. 2, 2025. DOI: 10.1111/cgf.70080
2. Hossain I., Shen I., van Kaick O. **Approximating Procedural Models of 3D Shapes with Neural Networks (NNProc), vol.44 (2),** 2025. DOI: 10.1111/cgf.70024
3. Dajkhosh S. **Utilizing WaveFunctionCollapse Algorithm for Procedural Generation of Terrains using Remotely Sensed Elevation Data,** 2024 . DOI: arxiv.org/abs/2412.04688
4. Farrokhi Maleki, M., & Zhao, R. Procedural Content Generation in Games: A Survey with Insights on Emerging LLM Integration. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 20(1), 167-178. 2024. DOI: 10.1609/aiide.v20i1.31877
5. Yi-Tang Chen, Haoyu Li, Neng Shi, Xihaier Luo, Wei Xu, Han-Wei Shen. Explorable INR: An Implicit Neural Representation for Ensemble Simulation Enabling Efficient Spatial and Parameter Exploration, 2025. https://arxiv.org/html/2504.00904v1
6. Fangxia Z., Ahmad K. 3D Animation Simulation of Computer Fractal and Fractal Technology Combined with Diamond-Square Algorithm. Applied Mathematics and Nonlinear Sciences, 2023, 8(1): 467-474.
7. Kudriavtsev, A., Bulgakova, O. Procedural Recursive Algorithm of Generation Based on Triangles. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham. 2025. DOI: 10.1007/978-3-031-84228-3_34
8. Fangxia Z., Ahmad K. 3D Animation Simulation of Computer Fractal and Fractal Technology Combined with Diamond-Square Algorithm. Applied Mathematics and Nonlinear Sciences, 2023, 8(1): 467-474.

Отримана в редакції 12.06.2025. Прийнята до друку 18.06.2025. Received 12 June 2025. Approved 18 June 2025. Available in Internet 30 June 2025