УДК 004.92
DOI https://doi.org/10.32782/2521-6643-2025-2-70.16

Новохатько І. О., аспірант кафедри комп'ютерних наук
Сумського державного університету
ORCID: 0000-0002-9796-3625

Федотова Н. А., кандидат технічних наук, доцент кафедри інформаційних технологій
Сумського державного університету
ORCID: 0000-0001-9304-1693

ПРОЦЕДУРНА ГЕНЕРАЦІЯ У ЧОТИРИВИМІРНОМУ ПРОСТОРІ: ВІД 4D ГРАДІЄНТНОГО ШУМУ ДО ВІЗУАЛІЗАЦІЇ КВАТЕРНІОННИХ ФРАКТАЛІВ

У статті запропоновано комплексний огляд методів процедурної генерації контенту (PCG) у чотиривимірному (4D) просторі, що є актуальним напрямом у сучасній комп'ютерній графіці. Розглянуто теоретичні та практичні аспекти використання четвертого виміру як інструменту для створення складних анімацій та безшовних текстур. Формально визначено евклідовий 4D-простір та його математичні основи. На відміну від 3D, де обертання відбувається навколо осі, у 4D-просторі воно відбувається навколо площини, що допускає унікальне явище подвійних обертань4. Досліджено також геометрію 4-політопів, таких як тесеракт, межі якого складаються з 3D-многогранників. Детально проаналізовано алгоритми градієнтного шуму, зокрема проведено порівняння класичного шуму Перліна з більш продуктивним та якісним симплекс-шумом. Показано, що гіперкубічна сітка шуму Перліна призводить до небажаних візуальних артефактів та має високу обчислювальну складність $O(n^4)$, тоді як симплекс-шум ($O(n^2)$) використовує більш ізотропну симпліціційну сітку. Досліджено методи генерації 4D-фрактальних структур, таких як множини Мандельброта та Жуліа, за допомогою алгебри кватерніонів. Ключовою властивістю алгебри кватерніонів є некомутативність множення, що дозволяє генерувати складні та асиметричні 4D-структури, неможливі у нижчих вимірах. Алгоритм генерації базується на ітераційній формулі, де кватерніонна множина Мандельброта визначається як множина констант C, для яких ітерація $Z_{n+1} = Z_n^2 + C$ залишається обмеженою. Оскільки пряме сприйняття 4D-об'єктів неможливе, у роботі систематизовано ключові техніки візуалізації: 3D-нарізку (зрізи), ортографічні та перспективні проєкції, а також об'ємний рендеринг за допомогою променевого маршингу. Перспективна проєкція, створює відомий ефект "куба в кубі" для тесеракта, даючи відчуття глибини вздовж четвертої осі. Об'ємний рендеринг за допомогою променевого маршингу виявився особливо ефективним для неявно заданих поверхонь, оскільки він використовує функції знакових відстаней (SDF) для фотореалістичної візуалізації. У висновках підсумовано поточний стан галузі та окреслено перспективні напрями досліджень, пов'язані з апаратним прискоренням на GPU та застосуванням генеративних моделей на основі ШІ.

Ключові слова: *4D-геометрія, процедурна генерація, градієнтний шум, симплекс-шум, кватерніонні фрактали, візуалізація, комп'ютерна графіка.*

Novokhatko I. O., Fedotova N. A. Procedural generation in four-dimensional space: from 4D gradient noise to visualization of quaternion fractals

The article offers a comprehensive review of procedural content generation (PCG) methods in four-dimensional (4D) space, which is a relevant direction in modern computer graphics. The theoretical and practical aspects of using the fourth dimension as a tool for creating complex animations and seamless textures are considered. The Euclidean 4D space and its mathematical foundations are formally defined. Unlike 3D, where rotation occurs around an axis, in 4D space it occurs around a plane, which allows for the unique phenomenon of double rotations4. The geometry of 4-polytopes, such as the tesseract, whose boundaries consist of 3D polyhedra, is also studied. Gradient noise algorithms are analyzed in detail, in particular, a comparison of classical Perlin noise with more productive and high-quality simplex noise is carried out. It is shown that the hypercubic Perlin noise mesh leads to undesirable visual artifacts and has a high computational complexity of $O(n^4)$, while the simplex noise ($O(n^2)$) uses a more isotropic simplicial mesh. Methods for generating 4D fractal structures, such as the Mandelbrot and Julia sets, using quaternion algebra are investigated. A key property of quaternion algebra is the noncommutativity of multiplication, which allows the generation of complex and asymmetric 4D structures that are not possible in lower dimensions. The generation algorithm is based on an iteration formula, where the Mandelbrot quaternion set is defined as the set of constants C for which the iteration $Z_{n+1} = Z_n^2 + C$ remains bounded. Since direct perception of 4D objects is impossible, the paper systematizes key visualization techniques: 3D slicing (slices), orthographic and perspective projections, and volumetric rendering using ray marching. Perspective projection creates the well-known "cube-in-cube" effect for the tesseract, giving a sense of depth along the fourth axis. Volumetric rendering using ray marching has been shown to be particularly effective for implicitly specified surfaces, as it

uses sign distance functions (SDFs) for photorealistic visualization. The conclusions summarize the current state of the art and outline promising research directions related to hardware acceleration on GPUs and the use of generative models based on AI.

Key words: *4D geometry, procedural generation, gradient noise, simplex noise, quaternion fractals, visualization, computer graphics.*

Постановка проблеми. Процедурна генерація контенту (PCG) є наріжним каменем сучасної комп'ютерної графіки та означає створення даних за допомогою алгоритмів, а не вручну [1]. Ця парадигма пропонує значні переваги, зокрема здатність генерувати величезні та складні світи, текстури та моделі з компактних алгоритмічних описів, тим самим зменшуючи розміри файлів та накладні витрати на розробку [2]. Хоча PCG найчастіше застосовується у двох та трьох вимірах, розширення цих методів на чотиривимірний (4D) простір відкриває нові горизонти можливостей.

Метою цієї роботи є систематизація знань про процедурну генерацію в 4D, аналіз ключових алгоритмів та методів візуалізації. Мотивація для використання 4D-генерації полягає в її здатності вирішувати специфічні, складні проблеми, які важко розв'язати в трьох вимірах. Четвертий вимір, що часто позначається координатою 'w', може слугувати параметром для безперервної еволюції, найчастіше – часу [4]. Це дозволяє анімувати 3D-ефекти з ідеальною часовою когерентністю. Іншим потужним застосуванням є генерація безшовно замощуваних текстур шляхом відображення 2D-простору на замкнуту 4D-поверхню [6].

Виклад основного матеріалу. Математичні основи чотиривимірного простору.

Для розробки процедурних алгоритмів у 4D необхідне розуміння базової математичної структури. Чотиривимірний евклідів простір, що позначається як R^4, є прямим розширенням звичного тривимірного простору, але з геометричними властивостями, які часто є контрінтуїтивними [15]. Точка в R^4 визначається кортежем.

Концепція обертання в 4D принципово відрізняється на відміну від 3D-простору, де обертання відбувається навколо осі (1D-підпростору), у 4D-просторі обертання відбувається навколо площини (2D-підпростору). Ці обертання описуються бівекторами і допускають унікальне явище подвійних обертань – двох одночасних і незалежних обертань у взаємно ортогональних площинах [17].

Так само, як 3D-простір населений многогранниками, 4D-простір містить їхні аналоги – поліхори або 4-політопи. Це 4D-об'єкти, межі яких складаються з 3D-многогранників. Існує шість правильних опуклих 4-політопів, найвідомішим з яких є тесеракт (гіперкуб), що складається з 8 кубічних комірок [22]. Геометрія цих фігур лежить в основі багатьох алгоритмів генерації.

Алгоритми градієнтного шуму в 4D. Градієнтний шум є основою багатьох технік PCG, створюючи плавні, природні патерни. Його розширення до чотирьох вимірів є концептуально простим, але створює значні обчислювальні виклики.

Алгоритм шуму Перліна, розширений до 4D, працює на сітці 4D-гіперкубів (тесерактів), інтерполюючи значення на основі градієнтних векторів у 16 вершинах навколишньої комірки [10]. Однак цей підхід має значні недоліки. Його обчислювальна складність $O(n^4)$ робить його занадто повільним для застосувань у реальному часі [5]. Крім того, гіперкубічна сітка призводить до візуальних артефактів, створюючи помітну упередженість уздовж осей координат, що робить шум менш природним на вигляд [28].

Для вирішення цих проблем був розроблений симплекс-шум [31]. Цей алгоритм використовує теселяцію простору симплексами (в 4D це 5-комірник) замість гіперкубів. Такий підхід забезпечує суттєві переваги. По-перше, обчислювальна складність зменшується до $O(n^2)$, оскільки в 4D потрібно обробити лише 5 вершин замість 16, що значно прискорює обчислення [25]. По-друге, симпліційна сітка є більш ізотропною, що зменшує кількість артефактів напрямку та створює більш якісний та природний результат [28]. Таким чином, для будь-яких серйозних застосувань 4D-шуму, особливо в реальному часі, симплекс-шум є значно кращим вибором.

Кватерніонні фрактали: Множини Мандельброта та Жуліа. Іншим потужним інструментом PCG є фрактали, що генерують нескінченну складність з простих ітераційних функцій. Класичні 2D-фрактали можна розширити до чотирьох вимірів за допомогою кватерніонів – системи числення, що має форму $a+bi+cj+dk$ [12]. Ключовою властивістю множення кватерніонів є його некомутативність $ij \neq ji$, що дозволяє створювати складні, асиметричні 4D-структури [35].

Генерація 4D-фракталів відбувається за ітераційною формулою, аналогічною до 2D, наприклад, $Z_{n+1}=Z_n^2+C$, *де* Z *та* C – кватерніони [12]. Алгоритм часу виходу визначає, чи належить точка множині, ітеруючи функцію доти, доки модуль точки не перевищить певного значення [38].

- Кватерніонні множини Жуліа: Для кожної фіксованої константи C створюється унікальна 4D-множина Жуліа [35].
- Кватерніонна множина Мандельброта: Це множина всіх констант C, для яких ітерація з початкової точки $Z_0 = 0$ залишається обмеженою [38].

Методи візуалізації 4D-об'єктів

Оскільки людина не може безпосередньо сприймати чотири просторові виміри, для візуалізації 4D-об'єктів необхідні методи зниження розмірності. Не існує єдиного "правильного" способу перегляду 4D-об'єкта; комплексне розуміння вимагає поєднання кількох підходів.

1. 3D-нарізка (перерізи): Це найбільш інтуїтивний метод, аналогічний МРТ-скануванню. 4D-об'єкт "нарізається" на послідовність 3D-зрізів шляхом фіксації однієї з координат (напр., w = const). Анімація значення цієї координати дозволяє побачити, як змінюється внутрішня структура об'єкта вздовж четвертої осі. Цей метод є чудовим інструментом для структурного аналізу, але показує лише одну частину об'єкта за раз, втрачаючи глобальний контекст [36].

2. Проєкція: Цей метод "кидає тінь" 4D-об'єкта на 3D-простір для відображення його загальної топології. Перспективна проєкція створює відомий ефект "куба в кубі" для тесеракта, де дальші частини виглядають меншими, що дає відчуття глибини по четвертій осі [21]. Стереографічна проєкція є більш складною технікою, яка є конформною (зберігає кути) і дозволяє "розгорнути" об'єкт для перегляду всієї його структури, хоча і зі значними спотвореннями [50].

3. Об'ємний рендеринг (променевий маршинг): Це потужний сучасний метод для візуалізації неявно заданих об'єктів, як-от фрактали. Він використовує функції знакових відстаней (SDF), які визначають найкоротшу відстань до поверхні об'єкта, для ітеративного просування променів у просторі. Цей підхід дозволяє створювати фотореалістичні зображення з високоякісним затіненням та освітленням, ідеально підходячи для візуалізації складних поверхонь [52].

Обговорення та майбутні напрямки. Сфера 4D-процедурної генерації стикається з кількома викликами та відкриває нові можливості для досліджень. Окрім розглянутих алгоритмів, існують і інші, наприклад, 4D-шум Ворлі, що створює коміркові структури [62], та 4D-клітинні автомати, здатні генерувати складні еволюціонуючі системи [64].

Основною перешкодою залишається висока обчислювальна вартість. Майбутні дослідження зосереджені на подоланні цієї проблеми шляхом перенесення процесу генерації на GPU та використання нових моделей програмування, як-от GPU work graphs, що краще підходять для динамічної логіки PCG [76].

Крім того, передовий край досліджень зміщується від жорстко закодованих алгоритмів до генеративних моделей на основі ШІ. Нейронні мережі, дифузійні моделі та неявні нейронні представлення (NeRFs) можуть "навчатися" правилам створення складних та узгоджених 4D-світів з даних, що потенційно може подолати однорідність класичних методів і запропонувати новий рівень керованості та різноманітності [73].

Висновки. У роботі представлено систематизований огляд методів процедурної генерації в чотиривимірному просторі. Було показано, що четвертий вимір є потужним концептуальним інструментом для вирішення практичних завдань комп'ютерної графіки, зокрема для створення когерентної анімації та безшовних текстур.

Аналіз алгоритмів градієнтного шуму продемонстрував, що для будь-яких застосувань у реальному часі симплекс-шум має значні переваги над шумом Перліна з точки зору продуктивності та візуальної якості. Дослідження кватерніонних фракталів розкрило їхній потенціал для створення детермінованих структур нескінченної складності.

Справжній виклик полягає у візуалізації цих невидимих об'єктів, що вимагає комбінованого підходу з використанням нарізки для структурного аналізу, проєкції для топологічного розуміння та променевого маршингу для високоякісного рендерингу. Майбутнє цієї галузі лежить у перенесенні обчислень на GPU та інтеграції генеративних моделей на основі ШІ для створення ще багатших та складніших віртуальних світів.

Список використаних джерел:

1. Процедурна генерація – Вікіпедія, URL: https://en.wikipedia.org/wiki/Procedural_generation
2. Процедурна генерація: Огляд | автор: Kenny – Medium, URL: https://kentpawson123.medium.com/procedural-generation-an-overview-1b054a0f8d41
3. 4D симплексний шум з аналітичними градієнтами, заснований на веб-gl-шумі Stegu – Reddit, URL: https://www.reddit.com/r/proceduralgeneration/comments/alihyr/4d_simplex_noise_with_analytical_gradients_based/
4. Яка мета використання 4D OpenSimplex шуму замість 2D Perlin шуму для створення циклічного шуму? – Stack Overflow, URL: https://stackoverflow.com/questions/65540942/what-is-the-intent-of-using-4d-opensimplex-noise-instead-of-2d-perlin-noise-to-c
5. Гарні картинки з полями шуму Перліна – Райан Маркус, URL: https://rmarcus.info/blog/2018/03/04/perlin-noise.html
6. Шум Перліна – Вікіпедія, URL: https://en.wikipedia.org/wiki/Perlin_noise
7. Шум Перліна: реалізація, процедурна генерація та симплексний шум – GarageFarm, URL: https://garagefarm.net/blog/perlin-noise-implementation-procedural-generation-and-simplex-noise
8. Двокватерніонні фрактали Жуліа 12pt22pt-5pt – arXiv, URL: https://arxiv.org/pdf/2303.14827
9. Розділ 2: Шумове обладнання – UMBC, URL: https://userpages.cs.umbc.edu/olano/s2002c36/ch02.pdf
10. Шум Перліна та його вдосконалення: огляд літератури | Прикладна та обчислювальна інженерія, URL: https://www.ewadirect.com/proceedings/ace/article/view/14225
11. Чотиривимірний простір – Вікіпедія, URL: https://en.wikipedia.org/wiki/Four-dimensional_space
12. medium.com, URL: https://medium.com/@nareshshrestha107/understanding-the-fundamentals-of-the-fourth-dimension-a67edda70dbd

13. Розуміння 4D -- Тессеракт – YouTube, URL: https://www.youtube.com/watch?v=iGO12Z5Lw8s
14. Візуалізація – Brown Math, URL: https://www.math.brown.edu/cdaly2/visualization.html
15. 4D вектор – Вікіпедія, URL: https://en.wikipedia.org/wiki/4D_vector
16. Чотиривимірна геометрія -- з Wolfram MathWorld, URL: https://mathworld.wolfram.com/Four-DimensionalGeometry.html
17. 4D фігури – Pardesco, URL: https://pardesco.com/blogs/news/4d-shapes
18. Тессеракт – Вікіпедія, URL: https://en.wikipedia.org/wiki/Tesseract
19. Алгоритм шуму Перліна – Процедурний світ Джейсміто, URL: https://jaysmito101.hashnode.dev/perlins-noise-algorithm
20. Симплексний шум розвінчано – Бременський університет CGVR, URL: https://cgvr.cs.uni-bremen.de/teaching/cg_literatur/simplexnoise.pdf
21. (PDF) Симплексний шум розвінчано – ResearchGate, URL: https://www.researchgate.net/publication/216813608_Simplex_noise_demystified
22. Розділ 5. Впровадження покращеного шуму Перліна – розробник NVIDIA, URL: https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-5-implementing-improved-perlin-noise
23. Розуміння шуму Перліна – детальний огляд самого алгоритму, а також деталей реалізації. Також містить повністю прокоментований та деобфускований код. : r/gamedev – Reddit, URL: https://www.reddit.com/r/gamedev/comments/2d284n/understanding_perlin_noise_an_indepth_look_at_the/
24. Задача Перліна: рух повз квадратний шум – NoisePosti.ng, URL: https://noiseposti.ng/posts/2022-01-16-The-Perlin-Problem-Moving-Past-Square-Noise.html
25. Шум Перліна: алгоритм генерації процедур – блог Рауфа, URL: https://rtouti.github.io/graphics/perlin-noise-algorithm
26. Симплексний шум – Вікіпедія, URL: https://en.wikipedia.org/wiki/Simplex_noise
27. Процедурні ефекти в реальному часі | NVIDIA, URL: https://www.nvidia.it/docs/IO/8343/RealTime-Procedural-Effects.pdf
28. Генерація процедурної карти з використанням шуму Перліна. Як її покращити? – Reddit, URL: https://www.reddit.com/r/proceduralgeneration/comments/16jcsu3/procedural_map_generation_using_perlin_noise_how/
29. N-вимірний генератор шуму Перліна: r/cpp – Reddit, URL: https://www.reddit.com/r/cpp/comments/greq8t/ndimensional_perlin_noise_generator/
30. Кватерніонні фрактали Жуліа – Пол Бурк, URL: https://paulbourke.net/fractals/quatjulia/
31. Розрізання 4-вимірних фракталів, URL: https://new.math.uiuc.edu/math198/MA198-2011/paul23/
32. Фрактальні дерева: Рекурсія, кватерніони та Python. – Блог Хуана Ернесто, URL: https://www.jernesto.com/articles/fractal_trees
33. Теорія кватерніонів – ChaosPro, URL: http://www.chaospro.de/documentation/html/fractaltypes/quaternions/theory.htm
34. Множина Мандельброта – Вікіпедія, URL: https://en.wikipedia.org/wiki/Mandelbrot_set
35. Розширення множини Мандельброта у вищі виміри – Архів Бріджес, URL: https://archive.bridgesmathart.org/2010/bridges2010-247.pdf
36. Фрактальні програми Кватерніон та Гіперкомплекс, URL: http://www.juliasets.dk/Quaternion.htm
37. 3D-рендеринг кватерніонної множини Мандельброта з пам'яттю – Пол Бурк, URL: https://paulbourke.org/papers/fractals2024/prepress.pdf
38. 4D-кватерніон Мандельброта Набір – Пол Найландер – WordPress.com, URL: https://nylander.wordpress.com/2009/07/07/4d-quaternion-mandelbrot-set/
39. Фрактал кватерніонів Жулія – Веньхао Юй, URL: https://wenhaoyu.weebly.com/quaternion-julia-set-fractal.html
40. Інтерактивний 4D-довідник – 3D-зрізи – Бейлі Снайдер, URL: https://baileysnyder.com/interactive-4d/3d-slices/
41. Візуалізація поверхонь у чотиривимірному просторі – Purdue e-Pubs, URL: https://docs.lib.purdue.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1813&context=cstech
42. Інтерактивний 4D-довідник – 4D-сфери – Бейлі Снайдер, URL: https://baileysnyder.com/interactive-4d/4d-spheres/
43. За межами 3D: Крижані кубики: Розрізання 4D-гіперкуба, URL: http://alem3d.obidos.org/en/cubeice/movsl4
44. Візуалізація 4-го виміру. – Mathematics Stack Exchange, URL: https://math.stackexchange.com/questions/2286180/visualizing-the-4th-dimension
45. Стереопроекція 4-го виміру: стереографічна проекція, URL: https://www.math.union.edu/~dpvc/math/4D/stereo-projection/welcome.html
46. Стереографічна проекція – Вікіпедія, URL: https://en.wikipedia.org/wiki/Stereographic_projection

47. Малювання за допомогою математики: ніжне дослідження променевого маршування – Блог Максима Хеккеля, URL: https://blog.maximeheckel.com/posts/painting-with-math-a-gentle-study-of-raymarching/
48. Оцінка відстані 3D-фракталів (частина I) – Syntopia, URL: http://blog.hvidtfeldts.net/index.php/2011/06/distance-estimated-3d-fractals-part-i/
49. Функція відстані зі знаком – Вікіпедія, URL: https://en.wikipedia.org/wiki/Signed_distance_function
50. Розкриття таємниць про функції відстані зі знаком | Scicho – Сейєд Саджад Абеді-Шахрі, URL: https://sadjadabedi.ir/post/demystifying-signed-distance-functions/
51. Неправильний спосіб використання функції відстані зі знаком (sdf) – URL: Winterbloed, https://winterbloed.be/the-wrong-way-to-use-a-signed-distance-function/
52. Функції відстані зі знаком: Моделювання в математиці – Hackaday, URL: https://hackaday.com/2023/04/12/signed-distance-functions-modeling-in-math/
53. Jellevermandere/4D-Raymarching: фреймворк Unity для … – GitHub, URL: https://github.com/Jellevermandere/4D-Raymarching
54. Трасування променів 4D фракталів, візуалізація чотиривимірних властивостей множини Жуліа, URL: http://www.codinginstinct.com/2008/11/raytracing-4d-fractals-visualizing-four.html
55. Трасування променів детермінованих 3-D фракталів – Кафедра комп'ютерних наук, URL: https://www.cs.drexel.edu/~david/Classes/Papers/rtqjs.pdf
56. Шум Ворлі – Вікіпедія, https://en.wikipedia.org/wiki/Worley_noise
57. Вузол VOP шуму Ворлі – SideFX, URL: https://www.sidefx.com/docs/houdini/nodes/vop/worleynoise.html
58. Клітинні автомати в архітектурі, дизайні та мистецтві | автор Степан Кухарський | Medium, URL: https://medium.com/@stepan.kukharskiy/cellular-automata-in-architecture-design-and-art-217fce9c1a5c
59. Процедурна генерація підземель: клітинні автомати – блог jrheard, URL: https://blog.jrheard.com/procedural-dungeon-generation-cellular-automata
60. (PDF) Процедурна генерація підземель – ResearchGate, URL: https://www.researchgate.net/publication/260800341_Procedural_Generation_of_Dungeons
61. Клітинний автомат – Вікіпедія, URL: https://en.wikipedia.org/wiki/Cellular_automaton
62. Процедурна генерація за допомогою клітинних автоматів – Бронсон Згеб, URL: https://bronsonzgeb.com/index.php/2022/01/30/procedural-generation-with-cellular-automata/
63. Процедурні текстури – Лабораторія медіа-досліджень Нью-Йоркського університету, URL: https://mrl.cs.nyu.edu/projects/texture/
64. Процедурне текстурування в 3D: Гнучкий та потужний робочий процес для художників – GarageFarm, URL: https://garagefarm.net/blog/procedural-texturing-in-3d-power-precision-and-possibility
65. Технологічна функція: Шум та фрактали – В іграх божевілля, URL: https://frictionalgames.blogspot.com/2010/11/tech-feature-noise-and-fractals.html
66. Досягнення в 4D-генерації: Огляд – arXiv, URL: https://arxiv.org/html/2503.14501v2
67. Досягнення в 4D-генерації: методи, виклики та майбутні напрямки – arXiv, URL: https://arxiv.org/html/2503.14501v3
68. Найкращий спосіб процедурної генерації великі, скінченні та детальні світи? : r/proceduralgeneration – Reddit, URL: https://www.reddit.com/r/proceduralgeneration/comments/7o5nlr/best_way_to_procedurally_generate_large_finite/
69. Подолання труднощів розробника за допомогою процедурної генерації в Unreal Engine, URL: https://moldstud.com/articles/p-overcoming-developer-challenges-with-procedural-generation-in-unreal-engine
70. Процедурна генерація в реальному часі за допомогою графіків роботи GPU – AMD GPUOpen, URL: https://gpuopen.com/download/Real-Time_Procedural_Generation_with_GPU_Work_Graphs-GPUOpen_preprint.pdf
71. Навігація четвертим виміром: відносність та сприйняття через 3D-лінзу, URL: https://www.researchgate.net/publication/389896982_Navigating_the_Fourth_Dimension_Relativity_and_Perception_Through_a_3D_Lens
72. Матеріали конференції Спеціальної групи інтересів з комп'ютерної графіки та інтерактивних методів, документи конференції – acm siggraph, URL: https://www.siggraph.org/wp-content/uploads/2025/08/Conference-Papers.html
73. ACM Transactions on Graphics (TOG): Том 42, № 4. 2023, URL: https://www.siggraph.org/wp-content/uploads/2024/02/ACM-Transactions-on-Graphics-Volume-42-Issue-4.html

References:

1. Procedural generation – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Procedural_generation
2. Procedural Generation: An Overview | by Kenny – Medium, Retrieved from: https://kentpawson123.medium.com/procedural-generation-an-overview-1b054a0f8d41
3. 4D Simplex Noise with Analytical Gradients, based on stegu's web-gl noise – Reddit, Retrieved from: https://www.reddit.com/r/proceduralgeneration/comments/alihyr/4d_simplex_noise_with_analytical_gradients_based/

4. What is the intent of using 4D OpenSimplex Noise instead of 2D Perlin Noise to create a looping noise? – Stack Overflow, Retrieved from: https://stackoverflow.com/questions/65540942/what-is-the-intent-of-using-4d-opensimplex-noise-instead-of-2d-perlin-noise-to-c

5. Pretty pictures with Perlin noise fields – Ryan Marcus, Retrieved from: https://rmarcus.info/blog/2018/03/04/perlin-noise.html

6. Perlin noise – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Perlin_noise

7. Perlin Noise: Implementation, Procedural Generation, and Simplex Noise – GarageFarm, Retrieved from: https://garagefarm.net/blog/perlin-noise-implementation-procedural-generation-and-simplex-noise

8. Dual-Quaternion Julia Fractals12pt22pt-5pt – arXiv, Retrieved from: https://arxiv.org/pdf/2303.14827

9. Chapter 2: Noise Hardware – UMBC, Retrieved from: https://userpages.cs.umbc.edu/olano/s2002c36/ch02.pdf

10. Perlin noise and its improvements: A literature review | Applied and Computational Engineering, Retrieved from: https://www.ewadirect.com/proceedings/ace/article/view/14225

11. Four-dimensional space – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Four-dimensional_space

12. medium.com, Retrieved from: https://medium.com/@nareshshrestha107/understanding-the-fundamentals-of-the-fourth-dimension-a67edda70dbd

13. Understanding 4D -- The Tesseract – YouTube, Retrieved from: https://www.youtube.com/watch?v=iGO12Z5Lw8s

14. Visualization – Brown Math, Retrieved from: https://www.math.brown.edu/cdaly2/visualization.html

15. 4D vector – Wikipedia, https://en.wikipedia.org/wiki/4D_vector

16. Four-Dimensional Geometry -- from Wolfram MathWorld, Retrieved from: https://mathworld.wolfram.com/Four-DimensionalGeometry.html

17. 4D Shapes – Pardesco, Retrieved from: https://pardesco.com/blogs/news/4d-shapes

18. Tesseract – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Tesseract

19. Perlin's Noise Algorithm – Jaysmito's Procedural World, Retrieved from: https://jaysmito101.hashnode.dev/perlins-noise-algorithm

20. Simplex noise demystified – Uni Bremen CGVR, Retrieved from: https://cgvr.cs.uni-bremen.de/teaching/cg_literatur/simplexnoise.pdf

21. (PDF) Simplex noise demystified – ResearchGate, Retrieved from: https://www.researchgate.net/publication/216813608_Simplex_noise_demystified

22. Chapter 5. Implementing Improved Perlin Noise – NVIDIA Developer, Retrieved from: https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-5-implementing-improved-perlin-noise

23. Understanding Perlin Noise – an in-depth look at the algorithm itself as well as implementation details. Also contains fully commented and deobfuscated code. : r/gamedev – Reddit, Retrieved from: https://www.reddit.com/r/gamedev/comments/2d284n/understanding_perlin_noise_an_indepth_look_at_the/

24. The Perlin Problem: Moving Past Square Noise – NoisePosti.ng, Retrieved from: https://noiseposti.ng/posts/2022-01-16-The-Perlin-Problem-Moving-Past-Square-Noise.html

25. Perlin Noise: A Procedural Generation Algorithm – Raouf's blog, Retrieved from: https://rtouti.github.io/graphics/perlin-noise-algorithm

26. Simplex noise – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Simplex_noise

27. Real-Time Procedural Effects | NVIDIA, Retrieved from: https://www.nvidia.it/docs/IO/8343/RealTime-Procedural-Effects.pdf

28. Procedural map generation using Perlin noise. How to improve? – Reddit, Retrieved from: https://www.reddit.com/r/proceduralgeneration/comments/16jcsu3/procedural_map_generation_using_perlin_noise_how/

29. N-Dimensional Perlin Noise Generator : r/cpp – Reddit, Retrieved from: https://www.reddit.com/r/cpp/comments/greq8t/ndimensional_perlin_noise_generator/

30. Quaternion Julia Fractals – Paul Bourke, Retrieved from: https://paulbourke.net/fractals/quatjulia/

31. Slicing 4-D Fractals, Retrieved from: https://new.math.uiuc.edu/math198/MA198-2011/paul23/

32. Fractal trees: Recursion, quaternions and Python. – Juan Ernesto Blog, Retrieved from: https://www.jernesto.com/articles/fractal_trees

33. Theory of Quaternions – ChaosPro, Retrieved from: http://www.chaospro.de/documentation/html/fractaltypes/quaternions/theory.htm

34. Mandelbrot set – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Mandelbrot_set

35. Expanding the Mandelbrot Set into Higher Dimensions – The Bridges Archive, Retrieved from: https://archive.bridgesmathart.org/2010/bridges2010-247.pdf

36. The Fractal Programs Quaternion and HyperComplex, Retrieved from: http://www.juliasets.dk/Quaternion.htm

37. 3D Rendering of the Quaternion Mandelbrot Set with Memory – Paul Bourke, Retrieved from: https://paulbourke.org/papers/fractals2024/prepress.pdf

38. 4D Quaternion Mandelbrot Set – Paul Nylander – WordPress.com, Retrieved from: https://nylander.wordpress.com/2009/07/07/4d-quaternion-mandelbrot-set/
39. Quaternion Julia Set Fractal – Wenhao Yu, Retrieved from: https://wenhaoyu.weebly.com/quaternion-julia-set-fractal.html
40. Interactive 4D Handbook – 3D Slices – Bailey Snyder, Retrieved from: https://baileysnyder.com/interactive-4d/3d-slices/
41. Visualization of Surfaces in Four-Dimensional Space – Purdue e-Pubs, Retrieved from: https://docs.lib.purdue.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1813&context=cstech
42. Interactive 4D Handbook – 4D Spheres – Bailey Snyder, Retrieved from: https://baileysnyder.com/interactive-4d/4d-spheres/
43. Beyond 3D: Iced Cubes: Slicing 4D hypercube, Retrieved from: http://alem3d.obidos.org/en/cubeice/movsl4
44. Visualizing the 4th dimension. – Mathematics Stack Exchange, Retrieved from: https://math.stackexchange.com/questions/2286180/visualizing-the-4th-dimension
45. 4th Dimension Stereo-projection: Stereographic Projection, Retrieved from: https://www.math.union.edu/~dpvc/math/4D/stereo-projection/welcome.html
46. Stereographic projection – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Stereographic_projection
47. Painting with Math: A Gentle Study of Raymarching – The Blog of Maxime Heckel, Retrieved from: https://blog.maximeheckel.com/posts/painting-with-math-a-gentle-study-of-raymarching/
48. Distance Estimated 3D Fractals (Part I) – Syntopia, Retrieved from: http://blog.hvidtfeldts.net/index.php/2011/06/distance-estimated-3d-fractals-part-i/
49. Signed distance function – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Signed_distance_function
50. Demystifying Signed Distance Functions | Scicho – Seyed Sadjad Abedi-Shahri, Retrieved from: https://sadjadabedi.ir/post/demystifying-signed-distance-functions/
51. The wrong way to use a signed distance function (sdf) – Winterbloed, Retrieved from: https://winterbloed.be/the-wrong-way-to-use-a-signed-distance-function/
52. Signed Distance Functions: Modeling In Math – Hackaday, Retrieved from: https://hackaday.com/2023/04/12/signed-distance-functions-modeling-in-math/
53. Jellevermandere/4D-Raymarching: a Unity framework to … – GitHub, Retrieved from: https://github.com/Jellevermandere/4D-Raymarching
54. Raytracing 4D fractals, visualizing the four dimensional properties of the Julia set, Retrieved from: http://www.codinginstinct.com/2008/11/raytracing-4d-fractals-visualizing-four.html
55. Ray Tracing Deterministic 3-D Fractals – Department of Computer Science, Retrieved from: https://www.cs.drexel.edu/~david/Classes/Papers/rtqjs.pdf
56. Worley noise – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Worley_noise
57. Worley Noise VOP node – SideFX, Retrieved from: https://www.sidefx.com/docs/houdini/nodes/vop/worleynoise.html
58. Cellular Automata in Architecture, Design, and Art | by Stepan Kukharskiy | Medium, Retrieved from: https://medium.com/@stepan.kukharskiy/cellular-automata-in-architecture-design-and-art-217fce9c1a5c
59. Procedural Dungeon Generation: Cellular Automata – jrheard's blog, Retrieved from: https://blog.jrheard.com/procedural-dungeon-generation-cellular-automata
60. (PDF) Procedural Generation of Dungeons – ResearchGate, Retrieved from: https://www.researchgate.net/publication/260800341_Procedural_Generation_of_Dungeons
61. Cellular automaton – Wikipedia, Retrieved from: https://en.wikipedia.org/wiki/Cellular_automaton
62. Procedural Generation with Cellular Automata – Bronson Zgeb, Retrieved from: https://bronsonzgeb.com/index.php/2022/01/30/procedural-generation-with-cellular-automata/
63. Procedural Textures – NYU Media Research Lab, Retrieved from: https://mrl.cs.nyu.edu/projects/texture/
64. Procedural Texturing in 3D: A Flexible and Powerful Workflow for Artists – GarageFarm, Retrieved from: https://garagefarm.net/blog/procedural-texturing-in-3d-power-precision-and-possibility
65. Tech Feature: Noise and Fractals – In The Games Of Madness, Retrieved from: https://frictionalgames.blogspot.com/2010/11/tech-feature-noise-and-fractals.html
66. Advances in 4D Generation: A Survey – arXiv, Retrieved from: https://arxiv.org/html/2503.14501v2
67. Advances in 4D Generation: Techniques, Challenges, and Future Directions – arXiv, Retrieved from: https://arxiv.org/html/2503.14501v3
68. Best way to procedurally generate large, finite, and detailed worlds? : r/proceduralgeneration – Reddit, Retrieved from: https://www.reddit.com/r/proceduralgeneration/comments/7o5nlr/best_way_to_procedurally_generate_large_finite/

69. Overcoming Developer Challenges with Procedural Generation in Unreal Engine, Retrieved from: https://moldstud.com/articles/p-overcoming-developer-challenges-with-procedural-generation-in-unreal-engine

70. Real-Time Procedural Generation with GPU Work Graphs – AMD GPUOpen, Retrieved from: https://gpuopen.com/download/Real-Time_Procedural_Generation_with_GPU_Work_Graphs-GPUOpen_preprint.pdf

71. Navigating the Fourth Dimension: Relativity and Perception Through a 3D Lens, Retrieved from: https://www.researchgate.net/publication/389896982_Navigating_the_Fourth_Dimension_Relativity_and_Perception_Through_a_3D_Lens

72. Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers – acm siggraph, Retrieved from: https://www.siggraph.org/wp-content/uploads/2025/08/Conference-Papers.html

73. ACM Transactions on Graphics (TOG): Vol. 42, No. 4. 2023, Retrieved from: https://www.siggraph.org/wp-content/uploads/2024/02/ACM-Transactions-on-Graphics-Volume-42-Issue-4.html

Дата надходження статті: 16.10.2025
Дата прийняття статті: 10.11.2025
Опубліковано: 30.12.2025