

АЛГОРИТМІЧНІ ТА СТРУКТУРНІ МЕХАНІКИ АЛХІМІЇ У ЧОТИРИВИМІРНИХ ІГРОВИХ ПРОСТОРАХ

Мирошніченко Ратмир Богданович

студент групи МПМ-1-25-1.4-д,

Київського столичного університету імені Бориса Грінченка,

науковий керівник – д.т.н., проф. Бондарчук А.П.

ВСТУП

Актуальність і постановка проблеми

Розвиток сучасної індустрії інтерактивних розваг вимагає постійного пошуку нових геймдизайнерських рішень та ускладнення внутрішніх ігрових систем. Традиційні тривимірні простори з лінійними чи квазілінійними механіками взаємодії об'єктів поступово досягають меж своєї концептуальної виразності. Одним із найбільш перспективних і водночас наукомістких напрямів є інтеграція чотиривимірних (4D) ігрових просторів, де четвертий вимір виступає не просто як шкала часу, а як повноцінна просторова або параметрична координата. Реалізація таких систем пов'язана зі стрімким зростанням обчислювальної складності та виникненням проблеми «прокляття розмірності» (curse of dimensionality), коли обсяг простору станів зростає експоненційно відносно кількості параметрів взаємодії. Керування багатовимірними структурами, забезпечення механік крафтингу, алхімії чи процедурної генерації контенту в таких умовах вимагає порівняльного аналізу та обґрунтованого вибору базових технологій обробки, стиснення та дедуплікації даних з метою оптимізації обчислень у реальному часі.

Мета дослідження

Системне порівняння технологічних рішень та архітектурних підходів до керування чотиривимірними ігровими просторами та забезпечення функціонування складних алгоритмічних і структурних механік в умовах жорстких обмежень обчислювальних ресурсів ігрових рушіїв.

Аналіз останніх досліджень і публікацій

Проблематика побудови та оптимізації складних ігрових механік і математичного моделювання віртуальних просторів досліджується в багатьох фундаментальних і прикладних працях. Загальні алгоритмічні засади організації даних та пошукових структур висвітлено у роботах Кормена і Лейзерсона [1], які заклали базу для реалізації систем динамічної взаємодії об'єктів. Особливості подолання проблеми великої розмірності та оптимізації багатовимірного пошуку детально проаналізовано у дослідженнях Індіка та Мотвані [2]. У контексті ігрової індустрії питання архітектурного проектування сховищ ігрових станів та швидкого двійкового кодування тісно пов'язані з розробками NoSQL-рішень, зокрема LevelDB [3] та RocksDB [4], які демонструють високу пропускну здатність при роботі з упорядкованими ключами. Математична формалізація неперервних трансформацій станів через алгоритми зваженого середнього векторів розглядається у сучасних оптимізаційних моделях (INFO [5], WMOV [6]). Водночас питання процедурної генерації у складних зв'язаних просторах досліджували Шейкер, Тогеліус та

Нельсон [7]. Проте питання комплексного порівняння технологій керування саме чотиривимірними ігровими середовищами та забезпечення детермінованості їхніх механік залишається розрізненим і потребує систематизації.

КОРОТКИЙ ОПИС ДОСЛІДЖЕННЯ, МЕТОДІВ І ЗАСОБІВ

Для розв'язання поставлених задач у роботі використано комплексний методологічний підхід, що поєднує методи лінійної алгебри для моделювання векторних просторів ознак, теорію алгоритмів і структур даних для оптимізації пошуку та обробки колізій, а також методи системного аналізу для порівняльної оцінки технологічних стеків [8]. Моделювання чотиривимірного простору виконувалося шляхом розширення класичного евклідового простору додатковою координатною віссю, що визначає стан або специфічну фізико-хімічну характеристику об'єкта чи інгредієнта.

Основними засобами дослідження виступили концептуальні моделі обробки даних на базі стандарту IEEE 754 для збереження обчислювальної детермінованості чисел із плаваючою комою, а також архітектурні патерни NoSQL-сховищ типу LSM-tree (Log-Structured Merge-tree) для оцінки ефективності серіалізації багатовимірних ключів [9]. Порівняльний аналіз проводився за критеріями обчислювальної складності, витрат оперативної пам'яті, детермінованості обчислень у мережевому середовищі та гнучкості масштабування простору станів [10] [11].

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У результаті проведеного аналізу було структуровано та зіставлено два ключові підходи до забезпечення ігрової механіки у чотиривимірних просторах: векторно-алгебраїчний підхід (із використанням динамічного обчислення зважених середніх) та дискретно-сховищний підхід (із фіксацією станів у високопродуктивних NoSQL-структурах).

Для формалізації механіки керування станом об'єкта у 4D-просторі стан S представлено як кортеж:

$$S = (P, W, F, C)$$

де $P \in R^4$ – чотиривимірний позиційний вектор параметрів,

$W \in R^4$ – вектор вагових коефіцієнтів впливу,

$F \in \{0,1\}^k$ – множина дискретних ознак (прапорів стану),

а C – контекст динамічної взаємодії. Особливе місце в дослідженні займає технологія канонічного кодування стану $encode(S) \rightarrow K$, що дозволяє трансформувати багатовимірні координати у компактний бінарний буфер. На відміну від стандартних хеш-функцій, які генерують незворотні індекси, канонічне кодування забезпечує бієктивність перетворення, зберігаючи точність представлення float-значень без втрат, що є критично важливим для усунення накопичувальних похибок округлення [12].

З метою детального порівняння технологій забезпечення алгоритмічних механік та керування 4D-просторами було сформовано аналітичну таблицю (Таблиця 1), яка відображає функціональні особливості, переваги та недоліки кожного технологічного підходу [13, 15].

Таблиця 1.

Порівняльний аналіз технологій керування 4D-просторами та ігровими механіками

Критерій порівняння	Векторно-алгебраїчні технології (INFO, WMOV, IEEE 754)	LSM-структури та NoSQL-сховища (LevelDB, RocksDB)	Процедурно-інверсні механізми генерації дерев станів
Основна функціональна задача	Обчислення неперервних трансформацій та динамічне змішування параметрів у реальному часі.	Організація компактного зберігання, індексування та швидкого пошуку ($O(1)$) створених станів.	Гарантування досяжності цільових ігрових об'єктів та динамічна побудова рецептурних ланцюжків.
Математичний / алгоритмічний апарат	Апарат лінійної алгебри, зважене середнє векторів, квантування та нормалізація.	Канонічне бінарне кодування, префіксне стиснення, сортування послідовностей байтів у SSTables.	Рекурсивний розклад цільового вектора станів, побудова та обхід спрямованих графів.
Обчислювальна складність	Висока при великій кількості інгредієнтів; вимагає оптимізації операцій над векторами.	Амортизована складність пошуку $O(1)$ завдяки використанню хеш-індексів та фільтрів Блума.	Залежить від глибини рекурсії та розгалуженості графа, контролюється обмеженням простору станів.
Вплив на детермінованість системи	Чутливий до архітектури процесора (вимагає суворого дотримання стандартів IEEE 754 для уникнення розсинхронізації).	Абсолютна детермінованість завдяки байтовому порівнянню канонічних ключів.	Висока детермінованість за умови фіксації початкового псевдовипадкового числа (Seed).
Переваги	Гнучкість, відсутність потреби у жорстко заданій базі статичних рецептів, нескінченна варіативність.	Масштабованість до мільйонів станів, висока швидкість читання/запису, стійкість до колізій.	Мінімізація ручної праці геймдизайнерів, унікальний ігровий досвід для кожного гравця.
Недоліки	Ризик «вибухового» зростання розмірності простору; накопичення помилок обчислень float-значень.	Витрати оперативної пам'яті на підтримку індексів; складність реалізації неперервних модифікацій.	Обчислювальне навантаження в момент ініціалізації або генерації нових великих сегментів дерева.

Реалізація алгоритму змішування інгредієнтів у межах векторно-алгебраїчного підходу покладається на динамічний розрахунок результуючого вектора ваг W_{new} та позиційного вектора P_{new} :

$$W_{new} = \frac{W_{current} \cdot A_{current} + W_{ingredient} \cdot A_{ingredient}}{A_{current} + A_{ingredient}}$$

$$P_{new} = \frac{P_{current} \cdot W_{current} \cdot A_{current} + P_{ingredient} \cdot W_{ingredient} \cdot A_{ingredient}}{W_{current} \cdot A_{current} + W_{ingredient} \cdot A_{ingredient}}$$

де A – кількісна доля відповідного елемента у суміші. Ця математична модель забезпечує високу чутливість системи до пропорцій, перетворюючи дискретний набір правил на неперервне середовище.

Для оптимізації доступу до згенерованих станів без повного перебору (що є критичним при мільйонах комбінацій) технологічний стек NoSQL-сховищ забезпечує механізм триетапної валідації:

- Генерація канонічного бінарного представлення стану K .
- Обчислення швидкого індексного хешу $h = \text{hash}(K)$ для визначення сегмента пам'яті в LSM-таблиці.
- Пошук у відповідному сегменті з фінальним точним потиповим (байтовим) порівнянням ключів для нівелювання колізій.

Паралельно з цим, технологія інверсної генерації розв'язує проблему контролю простору станів через рекурсивний зворотний розклад від цільового об'єкта ($P = 0$) до базових компонентів за формулами зміщення:

$$P_1 = P + d \cdot A_1$$

$$P_2 = P - d \cdot A_2$$

де d – випадковий або зважений вектор зміщення в чотиривимірному просторі. Це гарантує, що побудоване дерево станів міститиме лише досяжні та збалансовані ігрові об'єкти, виключаючи появу «тупикових» гілок еволюції системи.

ВИСНОВКИ

Проведене дослідження дозволяє сформулювати наступні науково-практичні висновки щодо порівняння технологій керування чотиривимірними просторами та реалізації ігрових механік:

Технологічна синергія

Найбільшу ефективність демонструє гібридна архітектура, де векторно-алгебраїчні методи використовуються безпосередньо для розрахунку фізико-параметричних взаємодій у 4D-просторі, тоді як технології на базі LSM-tree (RocksDB/LevelDB) виступають інструментом довготривалого збереження та швидкої дедуплікації станів з амортизованою складністю ($O(1)$).

Забезпечення детермінованості

При використанні динамічних векторних обчислень критично важливим є впровадження канонічного кодування та квантування float-векторів. Це дозволяє повністю нівелювати апаратні розбіжності стандарту IEEE 754, забезпечуючи ідентичність ігрових станів на різних платформах, що є базовою вимогою для мережевої синхронізації [14].

Геймдизайнерський потенціал

Інтеграція технологій інверсної генерації та процедурного обмеження багатовимірних просторів станів усуває необхідність ручного проектування баз даних крафтингу. Побудова динамічних дерев досяжності дозволяє створювати унікальні конфігурації ігрових світів під кожного окремого користувача, суттєво підвищуючи показник повторного проходження (replayability) програмного продукту.

ДЖЕРЕЛА

1. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. 3rd ed. Cambridge: MIT Press, 2009. 1312 p.

2. Indyk P., Motwani R. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality // Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC). 1998. P. 604–613.
3. Dean J., Ghemawat S. LevelDB Implementation. Google, 2011. URL: <https://github.com/google/leveldb> (дата звернення: 05.04.2026).
4. RocksDB Team. RocksDB Documentation. Facebook, 2024. URL: <https://rocksdb.org> (дата звернення: 05.04.2026).
5. Abdel-Basset M., Mohamed R., Mirjalili S. An efficient optimization algorithm based on weighted mean of vectors // Expert Systems with Applications. 2022. Vol. 190.
6. Zhou Y., et al. Weighted mean of vectors algorithm with neighborhood information interaction // Applied Intelligence. 2024.
7. Shaker N., Togelius J., Nelson M. Procedural Content Generation in Games. Cham: Springer, 2016. 237 p.
8. Strang G. Linear Algebra and Its Applications. 4th ed. Boston: Cengage Learning, 2006. 576 p.
9. IEEE Standard for Floating-Point Arithmetic. IEEE 754-2019. New York: IEEE, 2019.
10. Stack Overflow. Good way to hash a float vector?
URL: <https://stackoverflow.com/questions/650175/good-way-to-hash-a-float-vector> (дата звернення: 05.04.2026).
11. Leskovec J., Rajaraman A., Ullman J. D. Mining of Massive Datasets. 2nd ed. Cambridge: Cambridge University Press, 2014. 511 p.
12. Knuth D. E. The Art of Computer Programming. Vol. 2: Seminumerical Algorithms. 3rd ed. Boston: Addison-Wesley, 1997. 762 p.
13. Stack Overflow. Hashing 2D, 3D and nD vectors.
URL: <https://stackoverflow.com/questions/5928725/hashing-2d-3d-and-nd-vectors> (дата звернення: 05.04.2026).
14. Goldberg D. What Every Computer Scientist Should Know About Floating-Point Arithmetic // ACM Computing Surveys. 1991. Vol. 23, No. 1. P. 5–48.
15. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhytsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.
URL: <https://doi.org/10.28925/9720213284km>