УДК 681.3
DOI https://doi.org/10.32782/2521-6643-2026-2-72.16

Фірсов О. Д., кандидат фізико-математичних наук, доцент,
доцент кафедри комп'ютерних наук та інженерії програмного забезпечення
Університету митної справи та фінансів
ORCID: 0000-0002-6528-6447

ЗАСТОСУВАННЯ INSTANCEDMESH ДЛЯ ВІЗУАЛІЗАЦІЇ ДИНАМІКИ ПРОЦЕСІВ У ТРИВИМІРНИХ КЛІТИННИХ СТРУКТУРАХ

У роботі розроблено алгоритм та його реалізацію для візуалізації спеціалізованих тривимірних клітинних структур із застосуванням можливостей класу INSTANCEDMESH бібліотеки Three.js мови програмування JavaScript. Об'єктом дослідження виступає процес створення моделей клітинних структур, які змінюються відповідно до теоретичних алгоритмів, тобто відповідно змінюється відображення. У ході роботи було запропоновано, реалізовано і протестовано алгоритм візуалізації тривимірного масиву, значення у котрому змінюються відповідно деяким тактам функціювання системи, що моделюється. Результати дослідження показують, що технологія обробки масиву з розширенням у вигляді класу INSTANCEDMESH є ефективним інструментом для 3D-візуалізації клітинних структур обмеженого розміру, але дозволяє створювати візуалізацію динаміки змін у процесі моделювання. Проаналізовано проблему розміру візуалізуємої структури, у 3D випадку розмір даних необхідний для виявлення значимих структур, на відміну від стандартних одновимірних клітинних автоматів зростає відповідно поліному третього ступеню, що з одного боку дозволяє організувати моделюючий процес, але виникає проблема візуалізації динаміки змін у сенсі можливості проаналізувати тривимірне зображення людиною аналітиком. Початкова чисельність клітин, що дозволяє проводити аналіз отриманих конструкцій змінюється в діапазоні 10^4–10^6. Проблемою візуалізації клітинних структур є не тільки швидкодія, а й візуалізація кожного елементу, так щоб отримати чітке зображення, як кожного елементу так і всіє конструкції цілком.

Запропоноване рішення може бути використане для моделювання динамічних процесів, що можуть бути формалізовані та візуалізовані у вигляді воксельних структур. Підходи до організації процесів обробки та візуалізації даних для тривимірних клітинних структур з урахуванням продуктивності і доступності для аналізу кожного елементу представлення реалізовано та успішно протестовано. Отримані результати будуть слугувати для проведення обчислювальних експериментів для аналізу розвитку тривимірних клітинних автоматів чи подібних структур.

Ключові слова: *3D-моделювання, клітинний автомат, просторові дані, INSTANCEDMESH, візуалізація, інтеграція даних*

Firsov O. D. The application of InstancedMesh for visualizing the dynamics of processes in three-dimensional cellular structures

In the present study, an algorithm along with its software implementation has been developed for the visualization of specialized three-dimensional cellular structures. The solution utilizes the capabilities of the INSTANCEDMESH class provided by the Three.js library in the JavaScript programming language.

The object of research is the process of constructing models of cellular structures that evolve in accordance with theoretical algorithms, resulting in corresponding changes to their visual representation. Within the framework of this work, an algorithm for visualizing a three-dimensional array has been proposed, implemented, and tested. In this array, cell values are updated according to specific time steps (tacts) of the modeled system's operation.

The research findings demonstrate that array processing technology extended by the INSTANCEDMESH class serves as an effective tool for the three-dimensional visualization of cellular structures of limited size. It enables the creation of dynamic visualizations that reflect changes occurring during the modeling process.

The issue of visualized structure size has been analyzed in detail. In the three-dimensional case, the volume of data required to identify significant structures grows according to a third-degree polynomial, in contrast to standard one-dimensional cellular automata. While this growth facilitates the organization of the modeling process, it simultaneously introduces challenges in visualizing the dynamics of changes, particularly regarding the ability of a human analyst to interpret the resulting three-dimensional image effectively.

The initial number of cells that permits meaningful analysis of the generated constructions ranges from 10^4 to 10^6. The visualization of cellular structures is constrained not only by computational performance but also by the necessity to render each individual element with sufficient clarity, ensuring a distinct perception of both separate elements and the overall construction.

The proposed solution can be applied to the modeling of dynamic processes that are amenable to formalization and representation as voxel-based structures. Approaches to the organization of data processing and visualization for three-dimensional

cellular structures – taking into account both performance requirements and the accessibility of each representational element for analysis – have been successfully implemented and tested.

The obtained results will support further computational experiments aimed at analyzing the development of three-dimensional cellular automata and analogous structures.

Key words: *3D modeling, cellular automaton, spatial data, INSTANCEDMESH, visualization, data integration*

Постановка завдання. 3D-моделювання комірок – це потужний метод для представлення та обробки просторових даних у цифровому середовищах. На відміну від традиційного багатокутного моделювання, де об'єкти описуються поверхнями з трикутників або багатокутників, воксельний підхід використовує дискретну решітку об'ємних елементів, які називаються комірками (вокселями). Кожна клітинка має фіксований розмір і може містити інформацію про свій стан: чи вона повна чи порожня, який матеріал або колір їй надано, а також додаткові характеристики, такі як щільність, температура або вектор швидкості. Це дозволяє створювати точні тривимірні моделі за допомогою вибірки природнього простору, що особливо корисно для завдань, пов'язаних із фізичним моделюванням та обробкою об'ємних даних.

Основою воксельного моделювання є регулярна тривимірна сітка. У найпростішому випадку це масив булевих значень (бінарний масив), де 1 означає наявність матеріалу, а 0 – його відсутність. Більш складні реалізації використовують багатовимірні структури даних, такі як розріджене октрі або хеш-карта, для економії пам'яті при роботі з розрідженими томами. Наприклад, у наукових моделюваннях обчислювальної гідродинаміки кожна клітина зберігає не лише факт заповнення, а й векторні поля: швидкість потоку, тиск, температура. Це дозволяє моделювати складні процеси, такі як турбулентність, теплопередача або поширення ударних хвиль, з високою точністю.

Воксельне моделювання використовується у найрізноманітніших сферах. У медицині воно є основою тривимірної реконструкції органів згідно з даними КТ і МРТ. Хірурги використовують воксельні моделі для передопераційного планування, друку окремих імплантів та віртуального моделювання втручань. В інженерному проєктуванні цей метод використовується для топологічної оптимізації конструкцій: програма аналізує розподіл навантажень у тривимірній решітці та видаляє надлишкові елементи, створюючи легкі, але міцні деталі для авіації та автомобілів. У наукових дослідженнях вокселі використовуються для моделювання клімату, поширення забруднення, росту кристалів і навіть біологічних процесів, таких як розвиток пухлин або міграція клітин.

Особливе місце займають комп'ютерна графіка та ігрова індустрія. Сучасні рушії, такі як Unreal Engine 5 із системою Nanite або спеціалізовані воксельні рендерери (MagicaVoxel, Qubicle, Voxatron), дозволяють працювати з мільйонами клітин у реальному часі. Оптимізація досягається за допомогою InstancedMesh у WebGL/Three.js або подібних механізмів у DirectX і Vulkan: замість тисяч окремих об'єктів рендериться один екземпляр геометрії з масивом матриць трансформації. Це зменшує кількість викликів draw з тисяч до одиниць, підвищуючи продуктивність у десятки разів. Крім того, використовуються техніки маршових кубів для створення гладкої поверхні з воксельної сітки, що дозволяє перейти від «кубічного» вигляду до органічних форм.

Природна дискретизація простору спрощує фізичні розрахунки: закони збереження маси, імпульсу та енергії реалізуються безпосередньо на рівні сусідніх комірок, воксельні моделі легко редагувати, шум Перліна, клітинні автомати або алгоритми функціонування хвиль дозволяють автоматично створювати адекватні моделі.

Проблеми моделювання полягають у великій потребі в пам'яті та обчислювальних ресурсах, тому використовуються розріджені структури (розріджене воксельне октрі), де порожні площі зберігаються компактно. Проблема складності анімації також актуальна: деформація воксельного об'єкта вимагає перерахунку всієї сітки.

Поточні тенденції розвитку пов'язані з інтеграцією вокселів із нейронними мережами та апаратним прискоренням. Проєкти на кшталт NeRF (Neural Radiance Fields) та їхні тривимірні аналоги дозволяють перетворювати фотографії у воксельні моделі з фотограмметричною точністю. У GPU-обчисленнях бібліотеки CUDA та Vulkan дозволяють симуляції в реальному часі на мільярдах комірок. Веб-технології, такі як Three.js з InstancedMesh і WebGPU, дозволяють створювати інтерактивні воксельні конструкції безпосередньо в браузері. Завданням подальших досліджень є розробка та реалізація алгоритму візуалізації 3D-структур, придатних для моделювання клітинних автоматів.

Аналіз останніх досліджень і публікацій. Переваги воксельного моделювання підтверджуються практичними результатами і включають простоту булевих операцій, природну підтримку об'ємних даних, зручність фізичних моделей і процедурної генерації.

Недоліки залишаються незмінними протягом багатьох років: висока витрата пам'яті, «сходинкові» поверхні, складнощі з анімацією та рендерінгом великих об'ємів.

Робота [4] дає класичний огляд переваг і недоліків воксельного підходу. Підкреслюється, що вокселі – це «3D-пікселі» (куби), які ідеально імітують реальність на рівні частинок, володіючи при цьому точністю, простотою булевих операцій, природною підтримкою руйнування та CSG.

Науковий огляд алгоритмів вокселізації (перетворення багатокутних/точкових моделей у воксельні моделі), властивостей воксельних структур, розріджених представлень (розріджених), октрісних, хеш-таблиць розглядається у [1]. Особливу увагу приділяється точності, пам'яті та швидкості алгоритмів.

Систематичний огляд використання воксельних моделей в архітектурі та міському плануванні наведено у [5]. Особлива увага приділяється інтеграції гетерогенних даних (клімат, освітлення, доступність, видимість, енергія). Виявлено прогалини в багатодоменній інтеграції та пропонуються напрямки майбутніх досліджень.

Робота [3] надає порівняльний аналіз до трьох основних форматів 3D-даних. Вокселі описуються як регулярна структура, ідеальна для великих даних (медицина, наукові моделювання), з перевагами у простоті індексації та фізичних розрахунків.

У статті [3] представлено результати застосування воксельного підходу для оцінки ергономіки та доступності просторів. Пропонується фреймворк, який поєднує доступність і видимість в єдиній воксельній сітці. Практична цінність цього методу в архітектурному проєктуванні демонструється.

Робота з ефективного воксельного представлення складних форм [6] розглядає алгоритми вокселізації, відображення та аналізу товщини стінок. Вона корисна для розуміння базових технік роботи з тонкостінними та складними об'єктами.

Популярне пояснення суті вокселів і їхнього місця в 3D-моделюванні наведено у [7]. Наведено приклади використання в іграх (Minecraft), медицині, науковій візуалізації. Підкреслюється простота редагування та процедурної генерації порівняно з багатокутними моделями.

Сучасні тенденції спрямовані на інтеграцію з багатодоменними даними в архітектурі та урбанізмі, використання в ергономіці та доступності, поєднання машинного навчання та розріджених структур для збереження пам'яті.

Особливий інтерес викликає використання сучасних програмних рішень, оптимізованих для роботи в браузері.

Мета статті. Моделювання за допомогою воксельного підходу дозволяє створювати алгоритми та розв'язувати прикладні задачі. Існує безліч інструментів для візуалізації таких структур, як 3D-об'єкти. Але виникає потреба у налаштуванні механізму візуалізації під конкретні задачі та обчислювальні процедури. Інша проблема доступність технології візуалізації до користувача-дослідника. Відповідно, актуально отримати 3Д об'єкт з відповідної предметної області за допомогою бібліотек оптимізації обрахунку його елементаргих складових, а саме візуалізовати динаміку ровитку процесів, що можуть бути змодельовані, як клітинний автомат у просторі. Метою работи є розробка алгоритму візуалізації змін у 3Д масиві заданого розміру з заданими параметрами візуалізації із застосуванням технології InstancedMesh.

Виклад основного матеріалу. Оптимізація InstancedMesh у Three.js – один із найефективніших способів рендерингу великої кількості ідентичних об'єктів (у випадку клітинного автомату – кубів) з мінімальними накладними витратами на CPU та GPU.

InstancedMesh забезпечує приріст продуктивності порівняно зі звичайним способом (багато окремих Mesh).

Класичний підхід JavaScript реалізує наступний код

```
for (let i = 0; i < 10000; i++) {
    const mesh = new THREE.Mesh(geometry, material);
    mesh.position.set(...);
    scene.add(mesh);}
```

За цим підходом відбувається наступна послідовність дій:

Кожна геометрія → окремим викликом малювання

Кожен матеріал → окремий виклик малювання (якщо не однаковий)

Кожен об'єкт → окремою передачею матриці трансформації на GPU

Процесор повинен обробляти 10 000+ об'єктів у сцені за кожен кадр

GPU отримує тисячі індивідуальних викликів draw

Результат: при 5–15 тисячах об'єктів FPS падає до 10-30, навіть на поьужних відеокартах.

InstancedMesh – це один об'єкт, який рендерить тисячі (або сотні тисяч) екземплярів однієї геометрії та матеріалу в одному або дуже невеликій кількості викликів малювання.

InstancedMesh дає максимальний ефект, коли кількість ідентичних об'єктів знаходиться в діапазоні 1 000–2 000. Усі об'єкти мають однакову геометрію та однаковий матеріал. Об'єкти відрізняються лише розташуванням, обертанням, масштабом і (за бажанням) кольором. Геометрія не надто складна (куб, сфера, проста модель). InstancedMesh дозволяє рендерити тисячі однакових об'єктів в одному виклику draw замість тисяч окремих викликів. Це ключова оптимізація для воксельної графіки, частинок, трави, дерев, кубічних хмар, процедурних міст і будь-якої сцени з великою кількістю повторюваних простих геометрій.

У випадку тривимірного масиву перехід від окремих Mesh до InstancedMesh дозволяє ефективно рендерити 10-20 тисяч кубів замість 2-5 тисяч при одному FPS.

Алгоритм візуалізації. Нижче наведено покроковий опис логіки у порядку виконання дій під час завантаження сторінки.

1. Підготовка HTML-структури та стилів. Створюється базова HTML-сторінка.

Тіло сторінки містить контейнер з трьома повзунками (Density, Width, Cube Size) та двома кнопками («Regenerate» та «Zoom: ON»).

Стилі CSS визначаються темною тематикою, розташуванням панелі керування у верхньому правому куті та красивим виглядом елементів.

2. Завантаження бібліотек Three.js. Вказано <script type=»importmap»> посилання на модулі Three.js (версія 0.168.0) та OrbitControls.

Імпорт основних класів: THREE и OrbitControls.

3. Ініціалізація основних об'єктів Three.js. Створюється об'єкт сцени, і встановлюється колір фону (0x0e1624).

Перспективна камера створюється з кутом зору 60°, початковою позицією (0, 55, 120) та центральною (0, 0, 0).

Рендерер WebGLRenderer створюється з увімкненим згладжуванням і підтримкою високої роздільної здатності (devicePixelRatio).

Рендерер додається до тіла документа у вигляді <полотна>.

4. Налаштування інтерактивного керування. Створюється об'єкт OrbitControl, який дозволяє обертати сцену мишею.

EnableDamping увімкнено, встановлено обмеження на близькість (minDistance, maxDistance).

5. Налаштування освітлення. Додається однорідне AmbientLight з інтенсивністю 0,9. Додається спрямоване Світло з інтенсивністю 1,8 і положенням (60, 100, 80).

6. Визначення констант і параметрів

Розмір масиву SIZE.

Розміром з одну клітину CELL_SIZE.

Контрольні змінні: щільність, sigmaMult, CUBE_SCALE.

7. Створення геометрії та матеріалу.

Створюється базова геометрія куба BoxGeometry.

MeshStandardMaterial створюється з кольором, блиском, шорсткістю та сяйвом.

8. Створкння InstancedMesh (оптимізований рендеринг). Один об'єкт InstancedMesh створено для 22 000 екземплярів. Об'єкт одразу додається до сцени.

9. Виклик функції regenerate() – це генерація хмари відповідно до незалежних нормальних розподілів. Під час кожного виклику (під час завантаження та натискання кнопки/зміни повзунків) виконується наступне:

Лічильник кубів скидається до нуля.

Для кожної координати x, y, z обчислюється відстань від центру.

Значення одномірного розподілу Гауса обчислюється окремо для кожної осі:

$$px = \exp(-(x - m)^2/(2 \cdot \sigma x^2)).$$

Аналогічно для *py* та *pz*.

Три ймовірності множаться: ймовірність = *px* × *py* × *pz*.

Якщо отримане значення перевищує поріг щільності, то:

положення куба розраховується з урахуванням зсуву OFFSET.

Встановлюється масштаб кубічного CUBE_SCALE.

Матриця трансформації та колір записуються в InstancedMesh.

Після циклу буфери instanceMatrix та instanceColor оновлюються для встановлення фактичної кількості екземплярів instancedMesh.count.

10. Пульсація анімації. У окремій функції pulseAnimation() кожні 8 мілісекунд обчислюється коефіцієнт періодичності. Застосовується масштабування всього instancedMesh.

11. Підключення керування (UI).

12. Виконується базовий цикл анімації.

Функція animate() викликається через requestAnimationFrame.

У кожному кадрі:

Керування (controls.update()) оновлюється.

Виконується Pulse (pulseAnimation()).

Сцена рендериться (renderer.render(сцена, камера)).

13. Обробка зміни розміру вікон. При зміні розміру вікна співвідношення сторін камери перераховується, оновлюються матриця проекції та розмір рендерера.

Фінальна послідовність запуску
HTML + стилі завантажуються.
Модулі Three.js завантажуються.
Створюються сцена, камера, рендерер, керування та світло.
Створюється InstancedMesh.
Викликається Regenerate() – це перший випадок, коли формується хмара.
Запускається нескінченний цикл animate().

Таким чином, створено алгоритм, який дозволяє ефективно реалізувати візуалізацію 3D-об'єкта, що складається з набору ідентичних елементів.

Як тест і приклад програми, що реалізує вищезазначений код, була реалізована функціональна залежність на кроку 9. Приклад безпосередньо демонструє можливості реалізації програмного забезпечення та феноменологічні проблеми візуалізації.

Проблеми візуалізації стосуються не лише продуктивності програмного рішення. Наприклад, ми не можемо створити «тривимірний гауссовий розподіл» (співвідношення трьох величин: X, Y, Z) без переходу до чотиривимірної візуалізації (де ймовірнісна густина буде, наприклад, колір або прозорість у об'ємі). Отже, те, що ми зазвичай називаємо «тривимірним розподілом Гауса», математично є візуалізацією двовимірного розподілу.

На 9-му кроці пропонується схема візуалізації гаусіанів уздовж осей. Це не вирішує проблему переходу у чотиривимірний простір, але дає розуміння проблем і використання додаткових інструментів візуалізації, а також демонструє здатність системи візуалізації працювати з динамічними змінами.

Висновки. Результати дослідження підтвердили, що InstancedMesh як спеціальний клас у бібліотеці Three.js, призначений для ефективного рендерингу великої кількості однакових об'єктів (інстансів) з однаковою геометрією та матеріалом є потужним і ефективним засобом для створення тривимірних моделей клітинних структур у фіксованому 3Д просторі.

Завдяки використанню InstancedMesh вдалося розробити високоякісні та візуально реалістичні 3D-моделі, які легко інтегруються у веб-додатки. Це дозволяє дослідникам, які працюють з клітинними автоматами, зосередитися безпосередньо на вивченні динаміки математичних обєктів.

У ході роботи було створено спеціалізований алгоритм побудови графічної структури, що підтримує динамічну зміну стану кожної комірки, що формує зображення, масштабу та положення об'єктів у просторі. Алгоритм реалізован у вигляді програмного продукту, який протестовано та задіяно у дослідженнях.

Список використаних джерел:

1. Aleksandrov M., Lang W., Papastamatiou K., Pijanowski B. Voxelisation algorithms and data structures: a review // Sensors. 2021. Vol. 21, № 24. P. 8241. DOI: 10.3390/s21248241. URL: https://www.mdpi.com/1424-8220/21/24/8241 (дата обращения: 16.03.2026).

2. Azadi F., Azadi S., Azadi M. Ergonomics of spatial configurations: a voxel-based modelling framework for accessibility and visibility simulations // Frontiers in Built Environment. 2024. Vol. 9. Art. 1300843. DOI: 10.3389/fbuil.2023.1300843. URL: https://www.frontiersin.org/articles/10.3389/fbuil.2023.1300843/full (дата обращения: 16.03.2026).

3. Jha S. Understanding point clouds, meshes, and voxels [Электронный ресурс] // Medium. 2022–2023. URL: https://medium.com/@sanjivjha/a-beginners-guide-to-3d-data-understanding-point-clouds-meshes-and-voxels-385e02108141 (дата обращения: 16.03.2026).

4. Spatial. The main benefits and disadvantages of voxel modeling [Электронный ресурс] // Spatial Blog. 2019. URL: https://blog.spatial.com/the-main-benefits-and-disadvantages-of-voxel-modeling (дата обращения: 16.03.2026).

5. Tyc J., Kania K., Włodarczyk-Sielicka M. A scoping review of voxel-model applications to enable multi-domain data integration in architectural design and urban planning // Architecture. 2023. Vol. 3, № 2. P. 219–242. DOI: 10.3390/architecture3020010. URL: https://www.mdpi.com/2673-8945/3/2/10 (дата обращения: 16.03.2026).

6. Wang J., Oliveira M. M., Kaufman A. E. Voxel-based representation, display and thickness analysis of intricate shapes [Электронный ресурс] // ResearchGate. 2005 (переизд. 2023). URL: https://www.researchgate.net/publication/4228063_Voxel-based_representation_display_and_thickness_analysis_of_intricate_shapes (дата обращения: 16.03.2026).

7. What are voxels and how are they used in 3D modeling? [Электронный ресурс] // MakeUseOf. 2022. URL: https://www.makeuseof.com/what-are-voxels-3d-modeling/ (дата обращения: 16.03.2026).

References:

1. Aleksandrov, M., Lang, W., Papastamatiou, K., Pijanowski, B. (2021). Voxelisation algorithms and data structures: a review. *Sensors*. Vol. 21, № 24. P. 8241. DOI: 10.3390/s21248241. Retrieved from: https://www.mdpi.com/1424-8220/21/24/8241 (дата обращения: 16.03.2026).

2. Azadi, F., Azadi, S., Azadi, M. (2024). Ergonomics of spatial configurations: a voxel-based modelling framework for accessibility and visibility simulations. *Frontiers in Built Environment*. Vol. 9. Art. 1300843. DOI: 10.3389/fbuil.2023.1300843. Retrieved from: https://www.frontiersin.org/articles/10.3389/fbuil.2023.1300843/full (дата обращения: 16.03.2026).

3. Jha S. Understanding point clouds, meshes, and voxels. Medium. 2022–2023. Retrieved from: https://medium.com/@sanjivjha/a-beginners-guide-to-3d-data-understanding-point-clouds-meshes-and-voxels-385e02108141 (дата обращения: 16.03.2026).

4. Spatial. The main benefits and disadvantages of voxel modeling. Spatial Blog. 2019. Retrieved from: https://blog.spatial.com/the-main-benefits-and-disadvantages-of-voxel-modeling (дата обращения: 16.03.2026).

5. Tyc, J., Kania, K., Włodarczyk-Sielicka, M. (2023). A scoping review of voxel-model applications to enable multi-domain data integration in architectural design and urban planning, *Architecture*. Vol. 3, № 2. P. 219–242. DOI: 10.3390/architecture3020010. Retrieved from: https://www.mdpi.com/2673-8945/3/2/10 (дата обращения: 16.03.2026).

6. Wang, J., Oliveira, M. M., Kaufman, A. E. Voxel-based representation, display and thickness analysis of intricate shapes. ResearchGate. 2005 (переизд. 2023). Retrieved from: https://www.researchgate.net/publication/4228063_Voxel-based_representation_display_and_thickness_analysis_of_intricate_shapes (дата обращения: 16.03.2026).

7. What are voxels and how are they used in 3D modeling? MakeUseOf. 2022. Retrieved from: https://www.makeuseof.com/what-are-voxels-3d-modeling/ (дата обращения: 16.03.2026).

Дата першого надходження статті до видання: 24.03.2026
Дата прийняття статті до друку після рецензування: 20.04.2026
Дата публікації (оприлюднення) статті: 30.05.2026