*DOI: https://doi.org/10.25140/2411-5363-2026-2(44)-268-277*
*УДК 004.275:004.738.5*

***Вікторія Віталіївна Гуменюк***
аспірантка кафедри комп'ютерних наук та інформаційних технологій
Житомирський державний університет імені Івана Франка (Житомир, Україна)
**E-mail:** Humeniuk-V@zu.edu.ua. **ORCID** https://orcid.org/0000-0002-4966-6300. **ResearcherID:** GQP-0005-2022

# ВПЛИВ КЛАСТЕРИЗАЦІЇ ОБ'ЄКТІВ НА ЕФЕКТИВНІСТЬ ВІДСІКАННЯ НЕВИДИМИХ ПОВЕРХОНЬ

*У статті досліджено вплив просторової кластеризації об'єктів на ефективність відсікання невидимих поверхонь у складних тривимірних сценах, зокрема BIM-моделях. Проаналізовано особливості застосування Frustum Culling та Occlusion Culling і визначено роль кластеризації як чинника, що впливає на точність перевірок видимості та продуктивність рендерингу. Узагальнено переваги й обмеження підходів Grid, Octree та BVH. Показано, що для просторово складних і нерівномірних сцен найбільш перспективною є ієрархія обмежувальних об'ємів, яка забезпечує точніше групування об'єктів і зменшує кількість надлишкових перевірок.*

***Ключові слова**: кластеризація об'єктів; відсікання невидимих поверхонь; Frustum Culling; Occlusion Culling; рендеринг; BIM-моделі; Octree; BVH.*
*Рис.: 2. Табл.: 1. Джерел: 11.*

**Актуальність теми дослідження.** Візуалізація складних тривимірних сцен у реальному часі стала базовою вимогою для сучасних інженерних застосунків, зокрема під час роботи з BIM-моделями та цифровими двійниками, де сцена містить тисячі об'єктів і значну полігональність. Процес формування кадру є циклічним і виконується багаторазово, а ключовими джерелами втрат продуктивності виступають надмірна кількість об'єктів, що надсилаються у графічний конвеєр, та перевитрати на обчислення видимості. О. В. Тотосько, А. Г. Микитишин і П. Д. Стухляк підкреслюють, що задачі видалення невидимих поверхонь належать до фундаментальних у комп'ютерній графіці та визначають ресурсоємність рендерингу складних сцен [1].

У практичних системах відсікання об'єктів поза пірамідою огляду (Frustum Culling) і відсікання за ознакою перекриття (Occlusion Culling) зменшують кількість геометрії та викликів відмальовування, однак ефективність цих методів прямо залежить від того, як організовано сцену. І. В. Коваленко показує, що поєднання Frustum Culling та Occlusion Culling знижує навантаження на апаратні ресурси під час рендерингу складних сцен, але результат визначається структурою даних і розподілом об'єктів у просторі [2]. Порівняння ієрархій для оклюзійного відсікання на архітектурних сценах також підтверджує, що вибір способу кластеризації (октадерева, BVH-подібні структури тощо) впливає на час кадру та стійкість алгоритму на великих моделях. Отже, дослідження впливу кластеризації об'єктів на ефективність відсікання невидимих поверхонь прямо відповідає запиту на прогнозовану продуктивність рендерингу складних BIM-сцен у реальному часі.

**Постановка проблеми.** Сучасні системи візуалізації тривимірних сцен працюють в умовах постійного зростання геометричної складності моделей, а для BIM-середовищ ця проблема посилюється великою кількістю окремих об'єктів, високою деталізацією та потребою забезпечувати інтерактивну частоту кадрів. M. Johansson і M. Roupé прямо зазначають, що дані, отримані з BIM, через велику кількість індивідуальних елементів і високу геометричну складність не можуть ефективно використовуватися для рендерингу в реальному часі без додаткової обробки; саме тому в таких системах застосовують механізми culling для відхилення об'єктів, які не формують фінальне зображення [3].

О. В. Тотосько, А. Г. Микитишин і П. Д. Стухляк підкреслюють, що завдання видалення невидимих поверхонь належить до найскладніших у комп'ютерній графіці, а ефективність відповідних алгоритмів суттєво залежить від організації процесу сортування та вибору способу роботи з простором сцени [1]. У свою чергу, І. В. Коваленко стверджує, що

використання Frustum Culling та Occlusion Culling істотно знижує навантаження на апаратні ресурси під час рендерингу складних тривимірних сцен [2]. Разом із тим J. Bittner та співавтори показують, що наївне застосування occlusion queries здатне не підвищувати, а знижувати загальну продуктивність через затримки CPU та простої GPU, тобто саме по собі впровадження алгоритму відсікання ще не гарантує ефективності [4].

Проблема полягає в тому, що результативність Frustum Culling та Occlusion Culling визначається не лише самим методом відсікання, а й способом просторової кластеризації об'єктів, який задає точність перевірок, кількість запитів видимості та обсяг службових витрат на обхід ієрархії. Порівняння ієрархічних структур для occlusion culling підтверджує, що різні підходи до групування об'єктів дають різний вплив на час побудови структури та час кадру; зокрема, LBVH демонструє близьку до BVH SAH продуктивність і швидше будується. Саме тому наукова проблема зводиться до визначення того, як параметри та спосіб кластеризації змінюють ефективність відсікання невидимих поверхонь у складних сценах і за яких умов ієрархічна організація простору забезпечує реальний приріст продуктивності рендерингу [3; 5].

**Аналіз останніх досліджень і публікацій.** У сучасних дослідженнях із комп'ютерної графіки проблема відсікання невидимих поверхонь розглядається як одна з базових для забезпечення продуктивного рендерингу складних тривимірних сцен. У вітчизняному науково-освітньому просторі О. В. Тотосько, А. Г. Микитишин і П. Д. Стухляк послідовно підкреслюють, що методи видалення невидимих ліній і поверхонь належать до ключових складників сучасної графіки, а сама ця галузь не є завершеною в теоретичному сенсі, оскільки продовжують активно розвиватися підходи до растрового сканування, рендерингу та просторової оптимізації. Такий підхід є важливим для формування теоретичної основи дослідження, оскільки він фіксує загальне наукове розуміння: підвищення швидкодії рендерингу досягається не лише апаратними ресурсами, а й раціональною організацією геометрії сцени та процедур її відбору перед передаванням до графічного конвеєра [1].

Окремий напрям досліджень пов'язаний із вдосконаленням Frustum Culling. У. Ассарссон і Т. Меллер у роботах, присвячених оптимізованому відсіканню за пірамідою огляду, показали, що прискорення перевірок досягається завдяки використанню ієрархій обмежувальних об'ємів AABB та OBB, а також за рахунок міжкадрової когерентності, кешування попередніх результатів і маскування перевірок площин. Це означає, що вже на рівні Frustum Culling ефективність алгоритму визначається не лише формулою перевірки перетину, а передусім тим, як об'єкти попередньо згруповані в ієрархічну структуру. Для теми кластеризації це має принципове значення, оскільки саме межі кластера, його просторовий об'єм і глибина вкладеності формують кількість перевірок, які виконує система в кожному кадрі. Отже, у працях цього напряму Frustum Culling розглядається не як ізольований етап, а як механізм, чутливий до структури просторової організації сцени [6].

Інший великий масив публікацій присвячений Occlusion Culling та його ієрархічним формам. Дж. Біттнер, М. Віммер, Г. Пірінгер і В. Пургатгофер довели, що наївне використання hardware occlusion queries не гарантує приросту продуктивності, оскільки може створювати затримки на CPU та простої GPU. Саме з цієї причини вони запропонували підхід Coherent Hierarchical Culling, у якому просторову й часову когерентність видимості використано для зменшення кількості запитів і кращого узгодження роботи центрального та графічного процесорів. Подальший розвиток цього напряму представлено в роботі О. Маттауша, Дж. Біттнера та М. Віммера, де алгоритм CHC++ удосконалено через адаптивне прогнозування видимості, пакетування запитів і побудову щільніших обмежувальних об'ємів; автори прямо вказують, що це дає приріст продуктивності до порядку величини порівняно з попередніми рішеннями. Таким чином, у сучасних дослідженнях Occlusion Culling домінує висновок про те, що результативність методу визначається насамперед якістю ієрархії, а не самим фактом використання occlusion queries [4].

Важливе місце в новіших прикладних публікаціях займають роботи, що переносять акцент із суто алгоритмічних схем на практичну архітектуру обробки сцени. У матеріалах Intel щодо Masked Software Occlusion Culling наголошено, що програмне оклюзійне відсікання з маскованим поданням глибини підтримує чергування растеризації оклюдерів і запитів видимості без штрафу продуктивності, а отже, краще інтегрується в обхід scene graph і прикладний код рендерингу. У той самий час сучасна документація Unity фіксує, що система під час попередньої обробки ділить сцену на комірки, формує дані про геометрію в межах комірок та видимість між сусідніми комірками, а потім за можливості об'єднує їх для зменшення обсягу даних; при цьому камери виконують і Frustum Culling, і Occlusion Culling одночасно. Unity прямо вказує, що такий підхід найкраще працює у сценах із невеликими, чітко відокремленими зонами, наприклад кімнатами та коридорами. Це фактично підтверджує на рівні промислової практики, що розмір і конфігурація кластерів прямо впливають на якість відсікання. [7]

Подібну логіку демонструють і дослідження та інструменти, орієнтовані на великі інженерні моделі. М. Йоганссон і М. Рупе, аналізуючи рендеринг BIM, встановили, що через велику кількість індивідуальних об'єктів і високу геометричну складність такі моделі не забезпечують інтерактивної частоти кадрів без спеціальних механізмів відсікання; тому автори запропонували систему, яка поєднує hardware occlusion queries із додатковими механізмами, що спираються на просторові відношення та семантику BIM, а також розвиває cells-and-portals підхід. У практиці Unreal Engine ця ж ідея реалізується через попередньо обчислену visibility-структуру: стан видимості зберігається в комірках, прив'язаних до позиції камери або гравця, а сам підхід рекомендований для малих і середніх середовищ, особливо там, де динамічне оклюзійне відсікання апаратно обмежене. Отже, останні дослідження й прикладні рішення одностайно показують, що центральним чинником ефективності є не окремий алгоритм Frustum Culling чи Occlusion Culling, а спосіб кластеризації сцени – поділ на комірки, вузли або ієрархічні об'єми, який визначає точність, швидкість і стабільність відбору видимих об'єктів [3; 8].

**Виділення недосліджених частин загальної проблеми.** Аналіз наявних праць показує, що в літературі переконливо обґрунтовано ефективність Frustum Culling та Occlusion Culling як засобів зниження обчислювального навантаження під час рендерингу складних сцен. Водночас основний акцент більшості досліджень зосереджено або на оптимізації окремого алгоритму оклюзійного відсікання, або на побудові спеціалізованої системи для конкретного типу моделей. Зокрема, у роботі про CHC доведено ефективність ієрархічного планування occlusion queries, а в CHC++ показано, що накладні витрати таких запитів у частині сценаріїв можуть знижувати загальну швидкодію настільки, що підхід поступається навіть простому view-frustum culling. Це означає, що питання продуктивності визначається не лише самим методом відсікання, а й властивостями просторової структури, у межах якої він реалізується [4].

Недостатньо дослідженим залишається прямий зв'язок між способом кластеризації об'єктів і підсумковою ефективністю відсікання невидимих поверхонь. У наявних публікаціях не систематизовано, як саме розмір кластера, глибина ієрархії, щільність заповнення вузлів і просторова неоднорідність сцени змінюють кількість перевірок видимості, число викликів відмальовування та час формування кадру. Для BIM-моделей ця прогалина є особливо суттєвою, оскільки дослідники прямо вказують, що вибір алгоритму залежить від типу сцени, а якість автоматично сформованих просторових розбиттів для деталізованих будівельних моделей залишається невизначеною. Саме тому потребує окремого дослідження вплив параметрів кластеризації на результативність Frustum Culling та Occlusion Culling у складних просторово насичених сценах [3].

**Мета дослідження**. Метою статті є теоретичне узагальнення та порівняльний аналіз впливу просторової кластеризації об'єктів сцени на ефективність відсікання невидимих поверхонь під час застосування Frustum Culling і Occlusion Culling у складних тривимірних сценах, зокрема BIM-моделях. Досягнення поставленої мети передбачає зіставлення основних підходів до кластеризації, а саме Grid, Octree та BVH, за їхнім впливом на точність перевірок видимості, кількість надлишкових перевірок, характер накладних витрат і придатність до сцен із різною просторовою структурою. Теоретичною основою такого аналізу є положення сучасних досліджень про те, що результативність ієрархічного відсікання визначається не лише самим алгоритмом, а і способом організації простору сцени та якістю побудови кластерів.

**Виклад основного матеріалу.** Ефективність відсікання невидимих поверхонь визначається тим, наскільки рано в конвеєрі рендерингу система відсікає геометрію, що не впливає на фінальне зображення. У класичному підході первинне відсікання виконується за межами піраміди огляду, тоді як додаткове скорочення обсягу відмальовування забезпечується перевіркою перекриття об'єктів ближчою геометрією. У роботі J. Bittner, M. Wimmer, H. Piringer і W. Purgathofer наголошено, що головною метою visibility culling є недопущення передавання невидимих об'єктів у рендеринговий конвеєр, а view-frustum culling розглядається як базова, але недостатня техніка, оскільки вона не усуває об'єкти, які залишаються всередині фрустума, але фактично не видимі через перекриття. Аналогічно Unity фіксує, що стандартне frustum culling усуває лише ті об'єкти, які не потрапляють у межі огляду камери, тоді як occlusion culling прибирає об'єкти, повністю закриті ближчими елементами сцени [4; 9].

На рисунку 1 показано базову логіку такого відбору: спочатку з поля розгляду вилучаються об'єкти поза view frustum, далі усуваються грані, не орієнтовані до спостерігача, а після цього виконується відсікання об'єктів, які не формують жодного видимого фрагмента через оклюзію. Саме ця схема є вихідною точкою для подальшого вдосконалення механізму перевірки видимості, оскільки вона демонструє, що між загальним набором геометрії сцени та стадією відмальовування існує кілька послідовних рівнів відбору. У праці D. Cohen-Or, Y. Chrysanthou, C. T. Silva і F. Durand така послідовність розглядається як фундаментальна основа visibility culling для walkthrough-застосувань [9; 10].

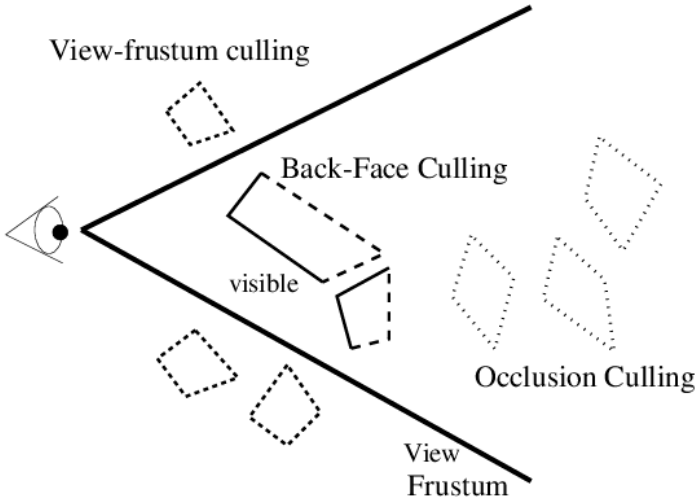


*Рис. 1. Типи відсікання невидимих елементів: view-frustum culling, back-face culling та occlusion culling [10]*

Для теми цього дослідження принциповим є те, що між етапом подання сцени та перевіркою видимості вводиться додаткова стадія – кластеризація об'єктів. Її сутність полягає в тому, що система працює не з кожним окремим об'єктом як із незалежною одиницею, а з просторовими групами, вузлами ієрархії або комірками сцени. Якщо межі кластера повністю розташовані поза фрустумом, увесь набір об'єктів усередині нього відсікається однією перевіркою. Якщо кластер залишається в межах огляду, але його обмежувальний об'єм не дає видимих фрагментів у запиті оклюзії, відсікається вся група без переходу до відмальовування дочірніх елементів. Саме такий підхід лежить в основі ієрархічного відсікання, яке, за J. Bittner та співавторами, мінімізує кількість запитів і скорочує затримки, пов'язані з латентністю результатів occlusion queries, завдяки обробці вузлів просторової ієрархії [4].

Найпростішим варіантом кластеризації є Grid, тобто рівномірний поділ простору на комірки однакового розміру. Така модель дає передбачувану структуру обходу, спрощує адресацію даних і добре масштабується в тих сценах, де об'єкти розподілені відносно рівномірно. Водночас її слабким місцем є низька адаптивність: у розріджених або просторово неоднорідних сценах частина комірок залишається майже порожньою, а в щільних фрагментах, навпаки, одна комірка може містити надто багато різнорідних об'єктів. У такому разі coarse-grained кластер має занадто великий обмежувальний об'єм, що погіршує точність frustum culling і збільшує частку «хибно позитивних» об'єктів, які потрапляють до наступного етапу перевірки. Для систем із попередньо підготовленими даними про видимість схожа логіка використовується й у промислових рушіях: Unity під час bake-процесу ділить сцену на cells, формує дані про геометрію в комірках і видимість між сусідніми комірками, а потім об'єднує їх там, де це зменшує обсяг службових даних [9].

Більш гнучким підходом є Octree, у межах якого простір поділяється ієрархічно: щільні ділянки сцени дробляться глибше, а менш насичені зони зберігаються на вищих рівнях дерева. Унаслідок цього щільність перевірок розподіляється нерівномірно, але значно точніше відповідає реальній структурі моделі. Для frustum culling така схема є вигідною, оскільки великі області поза пірамідою огляду відсікаються на верхніх рівнях, а детальний спуск до нижчих вузлів відбувається лише там, де це справді необхідно. Для occlusion culling перевага полягає в тому, що дрібніші вузли в зоні високої просторової щільності формують точніші proxy-об'єми для перевірки видимості. Разом із тим надмірне поглиблення дерева збільшує вартість обходу, а в динамічних сценах може ускладнювати оновлення структури. Отже, octree забезпечує кращу селективність відсікань, але потребує зваженого вибору глибини розбиття. На рівні загальної теорії це узгоджується з висновком D. Cohen-Or та співавторів про те, що в задачах видимості точні рішення часто замінюються кластеризацією, а ефективність визначається якістю такого групування [10].

Третій підхід – BVH (Bounding Volume Hierarchy) – принципово відрізняється від grid та octree тим, що він групує не порожній простір, а самі об'єкти через вкладені обмежувальні об'єми. Для складних BIM-сцен це особливо важливо, оскільки геометрія часто є нерівномірною: частина об'єктів зосереджена у вузлах інженерних комунікацій, а частина розміщена довгими протяжними структурами – стінами, перекриттями, шахтами, коридорами. У такій ситуації BVH формує щільніші межі навколо реальних груп об'єктів, ніж фіксовані просторові комірки, а це зменшує надлишкові перевірки. Підтвердженням цього є висновки O. Mattausch, J. Bittner і M. Wimmer: скорочення кількості запитів, зменшення змін стану рендерингу та використання щільніших bounding volumes безпосередньо підвищують продуктивність і можуть забезпечувати приріст до порядку величини порівняно з попередніми online-підходами. Це дає підстави вважати BVH найбільш придатним для сцен із нерівномірною геометричною насиченістю та складною просторовою структурою [11].

На рис. 2 подано порівняння трьох підходів до кластеризації сцени. Його логіка полягає в тому, що в усіх трьох випадках кількість і базове розташування об'єктів залишаються однаковими, але змінюється спосіб їх просторового групування. Саме таке порівняння є методично правильним для статті, оскільки воно наочно показує, що ключовим фактором є не сама сцена, а структура, через яку виконуються перевірки видимості. У контексті дослідження рис. 2 безпосередньо пояснює, чому та сама геометрія може давати різний результат під час Frustum Culling та Occlusion Culling залежно від обраного способу кластеризації.

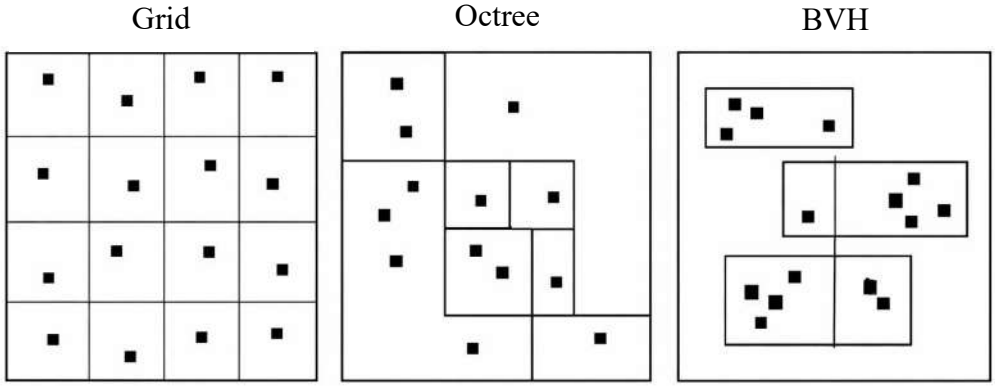


*Рис. 2 – Порівняння підходів до кластеризації сцени: рівномірний поділ простору (Grid), адаптивний ієрархічний поділ (Octree) та ієрархія обмежувальних об'ємів (BVH)*

Джерело: розроблено автором.

У табл. 1 узагальнено порівняльні характеристики розглянутих підходів до кластеризації з позиції їхнього впливу на ефективність Frustum Culling та Occlusion Culling.

*Таблиця 1 – Порівняльна характеристика підходів до кластеризації сцени для задач відсікання невидимих поверхонь*

| Підхід | Grid | Octree | BVH |
|---|---|---|---|
| Принцип побудови | Рівномірний поділ простору на комірки фіксованого розміру | Ієрархічний адаптивний поділ простору з поглибленням у щільних ділянках | Вкладені обмежувальні об'єми навколо груп реальних об'єктів |
| Вплив на Frustum Culling | Швидкий і простий обхід; ефективний за відносно рівномірного розподілу об'єктів | Дає точніше відсікання на різних рівнях дерева; зменшує кількість зайвих перевірок | Забезпечує точніші межі кластерів для складної та нерівномірної геометрії |
| Вплив на Occlusion Culling | Дає стабільні, але грубі результати; часто зберігає зайві перевірки через великі або порожні комірки | Формує більш локальні вузли для перевірки оклюзії; підвищує селективність відсікань | Зменшує кількість запитів завдяки щільнішим proxy-об'ємам і кращому групуванню |
| Основні обмеження | Низька адаптивність; неефективний у неоднорідних сценах | Зростають витрати на обхід і підтримку структури при надмірній глибині | Потребує якісної побудови ієрархії; чутливий до способу оновлення в динаміці |
| Тип сцени, для якого підхід є найкращим | Сцени з близькою щільністю заповнення простору | Сцени зі змінною щільністю та локальними зонами концентрації об'єктів | Сцени з нерівномірною геометрією, складною топологією та BIM-структурами |

Джерело: розроблено автором.

Дані, наведені в табл. 1, дають підстави стверджувати, що ключовим чинником ефективності є гранулярність кластера. Якщо кластер надто великий, система виконує малу кількість перевірок, але знижує точність відсікання: разом із потенційно видимими об'єктами до рендерингу потрапляє значний обсяг зайвої геометрії. Якщо ж кластер занадто дрібний, точність різко зростає, однак збільшується кількість тестів, обхід дерева, число occlusion queries та навантаження на CPU. Unity прямо вказує, що вбудоване occlusion culling виконує runtime-обчислення на CPU і саме тому не завжди автоматично покращує продуктивність; виграш спостерігається насамперед тоді, коли проєкт є GPU-bound через overdraw. У свою чергу CHC++ показує, що надлишкова кількість запитів і змін стану рендерингу здатна зробити навіть оклюзійне відсікання повільнішим за просте VFC у певних сценаріях. Отже, оптимальна кластеризація – це не максимально дрібний поділ, а такий рівень деталізації, за якого економія на відмальовуванні перевищує накладні витрати на перевірку видимості. [9; 11]

З огляду на це, робочий механізм оптимізованого рендерингу для складних сцен можна описати так. На першому етапі формується структура кластерів (grid, octree або BVH) і для кожного вузла задається обмежувальний об'єм. На другому етапі система виконує frustum culling у порядку зверху вниз: вузли, повністю розташовані поза фрустумом, одразу відсікаються, а для вузлів, що перетинають межі огляду, виконується подальший спуск у дочірні елементи. На третьому етапі для вузлів, які залишилися після перевірки фрустума, запускається occlusion culling – або через дані попередньо обчисленої видимості, або через runtime-запити до GPU. Після цього до draw calls потрапляють лише ті листові вузли або об'єкти, для яких підтверджено потенційну видимість. Саме така послідовність відповідає принципу, зафіксованому в CHC, де вузли просторової ієрархії обробляються у front-to-back порядку, а перевірки видимості поєднуються з відмальовуванням раніше підтверджених вузлів. [4; 9]

У межах дослідження вплив кластеризації на ефективність відсікання невидимих поверхонь доцільно оцінювати через сукупність взаємопов'язаних метрик: час кадру, кількість draw calls, число запитів видимості, частку відсічених об'єктів і процесорні витрати на обхід структури. Саме такий набір показників дозволяє відокремити реальну економію на рендерингу від формального збільшення кількості перевірок. Якщо, наприклад, частка відсічених об'єктів зростає, але при цьому різко збільшується CPU-час на обробку структури, підхід не можна вважати ефективним. Якщо ж при стабільній або помірній кількості запитів скорочуються draw calls і зменшується frame time, кластеризація працює результативно. На підставі праць J. Bittner та O. Mattausch можна зробити висновок, що саме узгодження структури кластера з просторовою і часовою когерентністю сцени є головною умовою зниження сумарних накладних витрат. [4; 11]

Для конкретизації подальшої емпіричної перевірки сформульовано узагальнену експериментальну постановку задачі. Порівняльне оцінювання ефективності Grid, Octree та BVH доцільно здійснювати на кількох типах сцен, зокрема на рівномірно заповнених синтетичних сценах, архітектурних сценах із локальними зонами концентрації об'єктів і BIM-моделях зі складною топологією та нерівномірним розподілом геометрії. Для кожного типу сцени варто враховувати число об'єктів і полігональність моделі, формуючи набори різної складності. Побудову кластерів у Grid доцільно виконувати на основі фіксованого кроку комірки, в Octree – через адаптивне просторове подрібнення до заданої глибини або порогу заповнення вузла, а в BVH – шляхом ієрархічного групування об'єктів за обмежувальними об'ємами. Як основні метрики оцінювання слід використовувати середній час кадру, кількість draw calls, частку відсічених об'єктів, кількість перевірок видимості та процесорні й

графічні витрати на обхід структури. Для забезпечення відтворюваності результатів порівняльне тестування має виконуватися на фіксованій апаратній конфігурації із зазначенням моделі процесора, відеокарти, обсягу оперативної пам'яті та параметрів програмного середовища. Така постановка задачі створює методичну основу для подальшої кількісної перевірки положень, узагальнених у статті.

Отже, у межах основного матеріалу встановлено, що кластеризація є не допоміжним, а визначальним елементом системи відсікання невидимих поверхонь. Вона задає одиницю перевірки, визначає точність відбору видимих елементів і безпосередньо впливає на співвідношення між скороченням обсягу рендерингу та накладними витратами на самі перевірки. Для простих і відносно рівномірних сцен достатнім може бути grid-поділ, однак для складних BIM-моделей із неоднорідною просторовою структурою більш обґрунтованими виглядають octree та BVH. На підставі проаналізованих підходів найбільш перспективним для таких сцен є BVH, оскільки він краще узгоджується з реальною геометрією моделі та забезпечує щільніші межі кластерів, тоді як octree залишається ефективним компромісом між адаптивністю та структурною прозорістю. Це є авторським висновком, сформованим на основі аналізу сучасних підходів до visibility culling та ієрархічного відсікання.

**Висновки.** Теоретичним результатом статті є узагальнення сучасних підходів до відсікання невидимих поверхонь у складних тривимірних сценах та обґрунтування визначальної ролі просторової кластеризації в забезпеченні ефективності Frustum Culling і Occlusion Culling. На основі аналізу наукових джерел установлено, що кластеризація не є допоміжним компонентом рендерингового конвеєра, а формує одиницю перевірки видимості, визначає точність відсікань і безпосередньо впливає на співвідношення між скороченням обсягу відмальовування та накладними витратами на обхід структури. Порівняльний аналіз підходів Grid, Octree та BVH показав, що їхня ефективність залежить від характеру просторового розподілу об'єктів, геометричної щільності сцени та ступеня її неоднорідності.

Прикладний результат дослідження полягає в обґрунтуванні логіки вибору способу кластеризації залежно від типу сцени. Для відносно однорідних сцен доцільним є використання Grid як простої та передбачуваної схеми організації простору. Для сцен зі змінною щільністю заповнення більш ефективним є Octree, який забезпечує адаптивне ієрархічне групування. Для складних інженерних і BIM-моделей найбільш перспективним підходом є BVH, оскільки ієрархія обмежувальних об'ємів формує щільніші межі навколо реальних груп об'єктів, зменшує кількість надлишкових перевірок і краще відповідає нерівномірній геометрії сцени. Практичне значення одержаних положень полягає в можливості використання їх як теоретичної основи для вибору структури просторової організації даних під час проєктування систем візуалізації BIM-моделей і складних 3D-сцен у реальному часі.

Перспективи подальших досліджень пов'язані з експериментальною перевіркою узагальнених положень на реальних і синтетичних сценах різного типу. Насамперед доцільно провести порівняльне тестування Grid, Octree та BVH за показниками часу кадру, кількості draw calls, частки відсічених об'єктів, кількості перевірок видимості та накладних витрат CPU/GPU. Окремий інтерес становить аналіз впливу числа об'єктів, полігональності моделі, глибини ієрархії та параметрів побудови кластерів на підсумкову продуктивність рендерингу. Це дозволить перейти від теоретичного узагальнення до кількісно підтвердженої моделі вибору оптимального способу кластеризації для сцен із різною просторовою структурою.

**Заява про використання генеративного ШІ та технологій на основі ШІ в процесі написання тексту статті.**

Під час підготовки статті авторка використовувала ChatGPT (OpenAI) як допоміжний інструмент для структуризації матеріалу, уточнення формулювань і мовного редагування. Після використання цього інструменту авторка переглянула та відредагувала текст за потреби й взяла на себе повну відповідальність за зміст публікації.

## Список використаних джерел

1. Тотосько, О. В., Микитишин, А. Г., & Стухляк, П. Д. (Уклад.). (2017). *Комп'ютерна графіка : навчальний посібник : в 2-х кн. Кн. 1. для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»*. Тернопільський національний технічний університет імені Івана Пулюя.

2. Коваленко, І. В. (2024). *Оптимізація ігор на рушії Unity за допомогою алгоритмів Occlusion Culling та Frustrum Culling [Магістерська дисертація, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»]*. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

3. Johansson, M., & Roupé, M. (2009). Efficient real-time rendering of building information models. *Proceedings of the 2009 international conference on computer graphics and virtual reality* (CGVR09) (p. 97).

4. Bittner, J., Wimmer, M., Piringer, H., & Purgathofer, W. (2004). Coherent hierarchical culling: Hardware occlusion queries made useful. *Computer Graphics Forum*, 23(3), 615–624. https://www.cg.tuwien.ac.at/research/vr/chcull/bittner-eg04-chcull.pdf.

5. Johansson, M. (2013). Integrating occlusion culling and hardware instancing for efficient real-time rendering of building information models. In *Proceedings of the International Conference on Computer Graphics Theory and Applications* (Vol. 2, pp. 197–206). SCITEPRESS.

6. Assarsson U., & Moller, T. (2000). Optimized view frustum culling algorithms for bounding boxes. *Journal of graphics tools*, 5(1), 9-22.

7. Hasselgren, J., Andersson, M., & Akenine-Möller, T. (2016). Masked software occlusion culling. *High Performance Graphics* (pp. 23-31).

8. Epic Games. (n.d.). *Precomputed visibility volume*. Epic Developer Community. https://dev.epicgames.com/documentation/en-us/unreal-engine/precomputed-visibility-volume?application_version=4.27.

9. Unity Technologies. (n.d.). Occlusion culling. Unity Manual. https://docs.unity3d.com/6000.3/Documentation/Manual/OcclusionCulling.html.

10. Cohen-Or, D., & Chrysanthou, Y. L. (2003). A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization & Computer Graphics*, 9(03), 412-431.

11. Mattausch, O., Bittner, J., & Wimmer, M. (2008). CHC++: Coherent hierarchical culling revisited. *Computer Graphics Forum*, 27(2), 221–230.

## References

1. Totosko, O. V., Mykytyshyn, A. H., & Stukhliak, P. D. (Uklad.). (2017). Kompiuterna hrafika: navchalnyi posibnyk: v 2-kh kn. Kn. 1. dlia studentiv spetsialnosti 151 "Avtomatyzatsiia ta kompiuterno-intehrovani tekhnolohii" [Computer graphics: textbook: in 2 books. Book 1. For students of specialty 151 "Automation and computer-integrated technologies"]. *Ternopilskyi natsionalnyi tekhnichnyi universytet imeni Ivana Puliuia Ternopil National Technical University named after Ivan Pulii.*

2. Kovalenko, I. V. (2024). Optymizatsiia ihor na rushii Unity za dopomohoiu alhorytmiv Occlusion Culling ta Frustrum Culling [Optimization of games on the Unity engine using Occlusion Culling and Frustrum Culling algorithms] [Mahisterska dysertatsiia, Natsionalnyi tekhnichnyi universytet Ukrainy "Kyivskyi politekhnichnyi instytut imeni Ihoria Sikorskoho"]. *Natsionalnyi tekhnichnyi universytet Ukrainy "Kyivskyi politekhnichnyi instytut imeni Ihoria Sikorskoho" – Igor Sikorsky Kyiv Polytechnic Institute, National Technical University of Ukraine.*

3. Johansson, M., & Roupé, M. (2009). Efficient real-time rendering of building information models. *Proceedings of the 2009 International Conference on Computer Graphics and Virtual Reality* (CGVR 2009) (p. 97).

4. Bittner, J., Wimmer, M., Piringer, H., & Purgathofer, W. (2004). Coherent hierarchical culling: Hardware occlusion queries made useful. *Computer Graphics Forum*, 23(3), 615–624. https://www.cg.tuwien.ac.at/research/vr/chcull/bittner-eg04-chcull.pdf.

5. Johansson, M. (2013). Integrating occlusion culling and hardware instancing for efficient real-time rendering of building information models. *Proceedings of the International Conference on Computer Graphics Theory and Applications* (Vol. 2, pp. 197–206). SCITEPRESS.

6. Assarsson, U., & Möller, T. (2000). Optimized view frustum culling algorithms for bounding boxes. *Journal of Graphics Tools*, 5(1), 9–22.

7. Hasselgren, J., Andersson, M., & Akenine-Möller, T. (2016). Masked software occlusion culling. *High Performance Graphics* (pp. 23–31).

8. Epic Games. (n.d.). Precomputed Visibility Volume. Epic Developer Community. https://dev.epicgames.com/documentation/en-us/unreal-engine/precomputed-visibility-volume?application_version=4.27.

9. Unity Technologies. (n.d.). Occlusion culling. Unity Manual. https://docs.unity3d.com/6000.3/Documentation/Manual/OcclusionCulling.html.

10. Cohen-Or, D., & Chrysanthou, Y. L. (2003). A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, 9(3), 412–431.

11. Mattausch, O., Bittner, J., & Wimmer, M. (2008). CHC++: Coherent hierarchical culling revisited. *Computer Graphics Forum*, 27(2), 221–230.

Дата першого надходження статті до видання: 17.03.2026
Дата прийняття статті до друку після рецензування: 26.03.2026

*UDC 004.275:004.738.5*

***Viktoriia Humeniuk***

Graduate Student of the Department of Computer Science and Information Technologies
Zhytomyr Ivan Franko State University (Zhytomyr, Ukraine)
**E-mail:** Humeniuk-V@zu.edu.ua. **ORCID**: https://orcid.org/0000-0002-4966-6300. **ResearcherID**: GQP-0005-2022

## THE IMPACT OF OBJECT CLUSTERING ON THE EFFICIENCY OF HIDDEN SURFACE REMOVAL

*The article examines the impact of object clustering on the efficiency of hidden surface removal in complex three-dimensional scenes, especially BIM models with dense geometry and many separate elements. The study is relevant because rendering performance in real-time visualization depends not only on scene complexity, but also on the accuracy of visibility determination before drawing. The paper analyzes the role of Frustum Culling and Occlusion Culling in the rendering pipeline and shows that their efficiency depends on the spatial organization of the scene. Clustering defines the unit of visibility testing, the granularity of exclusion, and the balance between rendering savings and the overhead of hierarchy traversal and visibility checks. Three clustering approaches are compared: Grid, Octree, and BVH. Grid provides a simple and predictable partitioning scheme, but loses efficiency in scenes with uneven object distribution. Octree improves the selectivity of checks through adaptive subdivision, although excessive hierarchy depth increases traversal costs. BVH is the most effective for complex BIM-oriented scenes because it groups objects into tighter bounding volumes, reduces redundant checks, and better matches actual scene geometry. The practical value of the results lies in selecting a clustering method according to scene topology and rendering workload. The analysis indicates that clustering is a key structural factor in rendering optimization. Grid is suitable for relatively uniform scenes, Octree for scenes with local density variations, and BVH for complex engineering and BIM models with irregular spatial structure.*

***Keywords:*** *object clustering; hidden surface removal; Frustum Culling; Occlusion Culling; rendering optimization; BIM models; Grid; Octree; BVH.*

*Fig.: 2. Table: 1. References: 11.*

Гуменюк В. В. Вплив кластеризації об'єктів на ефективність відсікання невидимих поверхонь. *Технічні науки та технології,* 2026. № 2(44). С. 268-277. DOI: https://doi.org/10.25140/2411-5363-2026-2(44)-268-277.