КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (27), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.27.768
УДК 004.9

Товсточуб Ігор Сергійович
аспірант
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID ID: 0009-0007-4743-5334
i.tovstochub@stud.duikt.edu.ua

Зінченко Ольга Валеріївна
доктор технічних наук, завідувач кафедри штучного інтелекту
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID ID: 0000-0002-3973-7814
o.zinchenko@duikt.edu.ua

ГЕНЕРАЦІЯ ГРАФІЧНИХ ОБ'ЄКТІВ В ІГРОВОМУ РУШІЇ GODOT ЗА МЕТОДОМ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Анотація. У статті представлено дослідження методів генерації графічних об'єктів в ігровому рушії Godot із застосуванням паралельних обчислень. Основна увага приділяється проблемі ефективного використання обчислювальних ресурсів для підвищення продуктивності генерації графічного контенту в динамічних ігрових середовищах. В ході дослідження було проведено детальний аналіз чотирьох основних категорій алгоритмів генерації графічних об'єктів, що активно застосовуються в екосистемі Godot Engine: алгоритми на основі шуму (Perlin, OpenSimplex), які використовуються для створення природних ландшафтів та текстур; клітинні автомати, що дозволяють моделювати органічні структури; Wave Function Collapse (WFC), який застосовується для генерації структурованих рівнів з правилами з'єднання; та L-системи, що ефективно використовуються для генерації фрактальних структур, зокрема рослинності. Для кожного типу алгоритмів визначено потенціал паралелізації та фактори, що обмежують їх ефективність на багатоядерних системах. Проблемою, яку виявлено при аналізі, є наявність в алгоритмах генерації значної частки послідовних обчислень, які, згідно з законом Амдала, суттєво обмежують максимально можливе прискорення навіть при збільшенні кількості обчислювальних ядер. Запропоновано математичну модель для оцінки потенційного прискорення в контексті генерації графічних об'єктів, що дозволяє точніше прогнозувати продуктивність різних алгоритмів на багатоядерних системах. В рамках дослідження розроблено інноваційний підхід до оптимізації, заснований на методі чанкової генерації, при якому графічні об'єкти розбиваються на просторово локалізовані елементи з мінімальними міжелементними залежностями. Ключовою особливістю запропонованого підходу є асинхронна генерація чанків та локалізація обчислень координат, що дозволяє суттєво знизити частку послідовних операцій і, як наслідок, значно підвищити потенційне прискорення відповідно до закону Амдала. Для об'єктивної оцінки ефективності розробленого методу проведено серію експериментів, що включали бенчмаркінг на різних типах графічних об'єктів з використанням вбудованого інструментарію Godot Engine. Методологія вимірювань базувалася на декомпозиції алгоритму на логічні фази та аналізі зміни часу виконання при поступовому збільшенні кількості потоків. Результати експериментів продемонстрували значне зменшення частки послідовних обчислень для всіх типів графічних об'єктів при застосуванні чанкового підходу, що підтверджує ефективність запропонованого методу. Результати дослідження мають практичне значення для оптимізації розробки ігор на базі Godot Engine та можуть бути адаптовані для інших систем візуалізації з подібною архітектурою.

Ключові слова: генерація графічних об’єктів; Godot; шумові алгоритми; асинхронна генерація; багатопотоковість; бенчмаркінг; паралельне обчислення.

ВСТУП

Генерація графічних об’єктів за методом паралельних обчислень координат є ключовим напрямком оптимізації в сучасній розробці відеоігор. Використання паралельних алгоритмів для обчислення вершин, нормалей та інших атрибутів графічних об’єктів дозволяє створювати масштабні, деталізовані та динамічні ігрові світи. Традиційно цей процес базується на послідовній генерації, де обчислення взаємозалежних елементів відбувається крок за кроком, що значно обмежує продуктивність, особливо для складних сцен.

Рушій Godot надає вбудовану підтримку для паралельних обчислень через багатопотоковість, асинхронне виконання та GPU-прискорення [1], [3]. Архітектура Godot з його вузловою системою та сигнальним підходом створює можливості для ефективної організації незалежних обчислень при генерації графічних об’єктів — від процедурних ландшафтів до складних динамічних структур. Проте паралельні обчислення мають фундаментальні обмеження, описані законом Амдала, згідно з яким максимальне прискорення системи обмежується часткою послідовних обчислень, які неможливо розпаралелити. Це створює значний виклик при генерації графічних об’єктів, які часто мають взаємозалежні елементи, що вимагають послідовної обробки. Дане дослідження пропонує методи подолання обмежень закону Амдала для генерації графічних об’єктів у Godot шляхом декомпозиції задач на незалежні чанки, локалізації залежностей та асинхронної генерації. Розроблені підходи дозволяють мінімізувати частку послідовних операцій і, як наслідок, максимізувати потенційне прискорення при використанні багатопотокової архітектури.

Постановка проблеми. Обмеженням для паралельної генерації графічних об’єктів є високий відсоток послідовних обчислень, особливо для алгоритмів з міжелементними залежностями [3], [4]. Необхідно розробити алгоритми, які дозволять мінімізувати ці обмеження та підвищити ефективність паралельних обчислень координат у контексті Godot Engine.

Аналіз останніх досліджень і публікацій. Сучасні дослідження в галузі паралельних обчислень для генерації графічних об’єктів у відеоіграх виявляють значний прогрес, однак обмеження закону Амдала залишаються фундаментальною проблемою, особливо в контексті рушія Godot.

У роботах [5] – [7] запропоновано класифікацію алгоритмів генерації графічних об’єктів та методи їх паралелізації. Хендрікс та співавтори [5] описують фундаментальні принципи паралельних обчислень для генерації контенту, проте не розглядають специфіку зменшення послідовної частки обчислень у Godot Engine. Аналогічно, Тогеліус та співавтори [7] представляють класифікацію методів генерації на основі пошуку, але не аналізують їх з точки зору потенціалу для паралелізації.

Шейкер, Тогеліус та Нельсон [6] розглядають теоретичні основи процедурної генерації, включаючи алгоритми шуму Перлін, проте не досліджують можливості декомпозиції таких алгоритмів для ефективного розпаралелювання координатних обчислень.

Саммервіль та співавтори [8] вивчають застосування машинного навчання в генерації контенту (PCGML), що відкриває нові можливості для паралельних обчислень, однак не аналізують співвідношення послідовних та паралельних операцій у таких підходах, критичне для оцінки максимального прискорення згідно з законом Амдала.

У контексті алгоритмів для 2D-генерації, Wave Function Collapse, досліджений Смітом [9], демонструє значний потенціал для паралелізації через його ймовірнісний характер, однак практична реалізація в Godot стикається з проблемами зменшення міжелементних залежностей, що обмежують потенційне прискорення [11].

Дослідження Ліндена [10] в контексті оптимізації пам'яті для генерованих світів пропонує чанковий підхід, проте не розглядає його з точки зору зменшення частки послідовних обчислень згідно з законом Амдала, що є критичним для максимізації потенційного прискорення.

Таким чином, існуючі дослідження не пропонують комплексного аналізу методів подолання обмежень закону Амдала при генерації графічних об'єктів у Godot Engine. Відсутній математичний аналіз співвідношення послідовних та паралельних операцій у різних алгоритмах генерації та практичні методи мінімізації послідовної частки обчислень.

У даній роботі ми досліджуємо методи паралельних обчислень координат графічних об'єктів у Godot, з особливим акцентом на мінімізацію частки послідовних обчислень для подолання фундаментальних обмежень закону Амдала.

Мета статті полягає в підвищенні ефективності генерації графічних об'єктів у ігровому рушії Godot шляхом мінімізації частки послідовних операцій при паралельних обчисленнях координат і, як наслідок, максимізації потенційного прискорення згідно з законом Амдала.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Закон Амдала, описує теоретичну межу прискорення комп'ютерної програми при використанні паралельних обчислень. Математично цей закон виражається формулою:

$$S(n) = \frac{1}{p + \frac{1 - p}{n}} \quad (1)$$

де:

- $S(n)$ — прискорення продуктивності при використанні nn;
- p — частка послідовних обчислень, які не можуть бути розпаралелені;
- $(1 - p)$ — частка обчислень, які можуть бути розпаралелені;
- n — кількість потоків.

Тобто, якщо 10% програми повинні виконуватися послідовно ($p = 0.1$), максимальне теоретичне прискорення не може перевищити 10 разів, незалежно від кількості використовуваних процесорів [3]. Для генерації графічних об'єктів у Godot Engine закон Амдала можна адаптувати, враховуючи специфіку процесу генерації:

$$S_{gen}(n) = \frac{1}{p_{gen} + \frac{1 - p_{gen}}{n}} \quad (2)$$

де p_{gen} — частка послідовних обчислень при генерації графічних об'єктів.

Для різних типів алгоритмів генерації значення p_{gen} може суттєво відрізнятися, визначаючи максимальне теоретичне прискорення для кожного з них.

Проведений аналіз виявив чотири основні категорії алгоритмів генерації графічних об'єктів у Godot Engine, кожна з яких має власні особливості з точки зору можливостей паралелізації:

- Алгоритми на основі шуму (Perlin, OpenSimplex) — використовуються для генерації ландшафтів, текстур та природних об'єктів [11], [14]. Ці алгоритми мають значний потенціал для паралелізації, оскільки генерація різних частин об'єкта може виконуватись незалежно. Однак взаємозалежності між елементами створюють певну частку послідовних обчислень, що обмежує максимальне прискорення згідно з законом Амдала.
- Клітинні автомати — застосовуються для створення печер, підземель та органічних структур. Ці алгоритми характеризуються відносно високою часткою послідовних обчислень через сильні міжелементні залежності. Стан кожної клітини залежить від стану сусідніх клітин на попередньому кроці, що створює послідовні залежності між ітераціями та обмежує потенціал для паралелізації.
- Wave Function Collapse (WFC) — використовується для генерації структурованих рівнів з правилами з'єднання. WFC має значну частку послідовних обчислень через складний алгоритм поширення обмежень. Каскадне поширення обмежень після кожного колапсу хвильової функції створює серйозні виклики для паралелізації, суттєво обмежуючи потенційне прискорення.
- L-системи — застосовуються для генерації фрактальних структур, зокрема рослинності [12]. Рекурсивна природа L-систем створює послідовні залежності між рівнями, але паралелізм можливий між гілками одного рівня. Перехід до ітеративної реалізації може зменшити частку послідовних обчислень.

Аналіз підтвердив, що ключовим обмеженням для паралелізації всіх алгоритмів є частка послідовних обчислень, яка відповідно до закону Амдала встановлює теоретичну межу максимального прискорення.

Для подолання обмежень закону Амдала було удосконалено алгоритм паралельної генерації графічних об'єктів, що базується на чанковому підході [15]. Основна ідея удосконалення — це зменшення частки послідовних обчислень (p) шляхом декомпозиції задачі на локалізовані незалежні підзадачі. Блок-схема удосконаленого алгоритму паралельної генерації графічних об'єктів показана на рис. 1.

№ 3 (27), 2025
ISSN 2663 - 4023

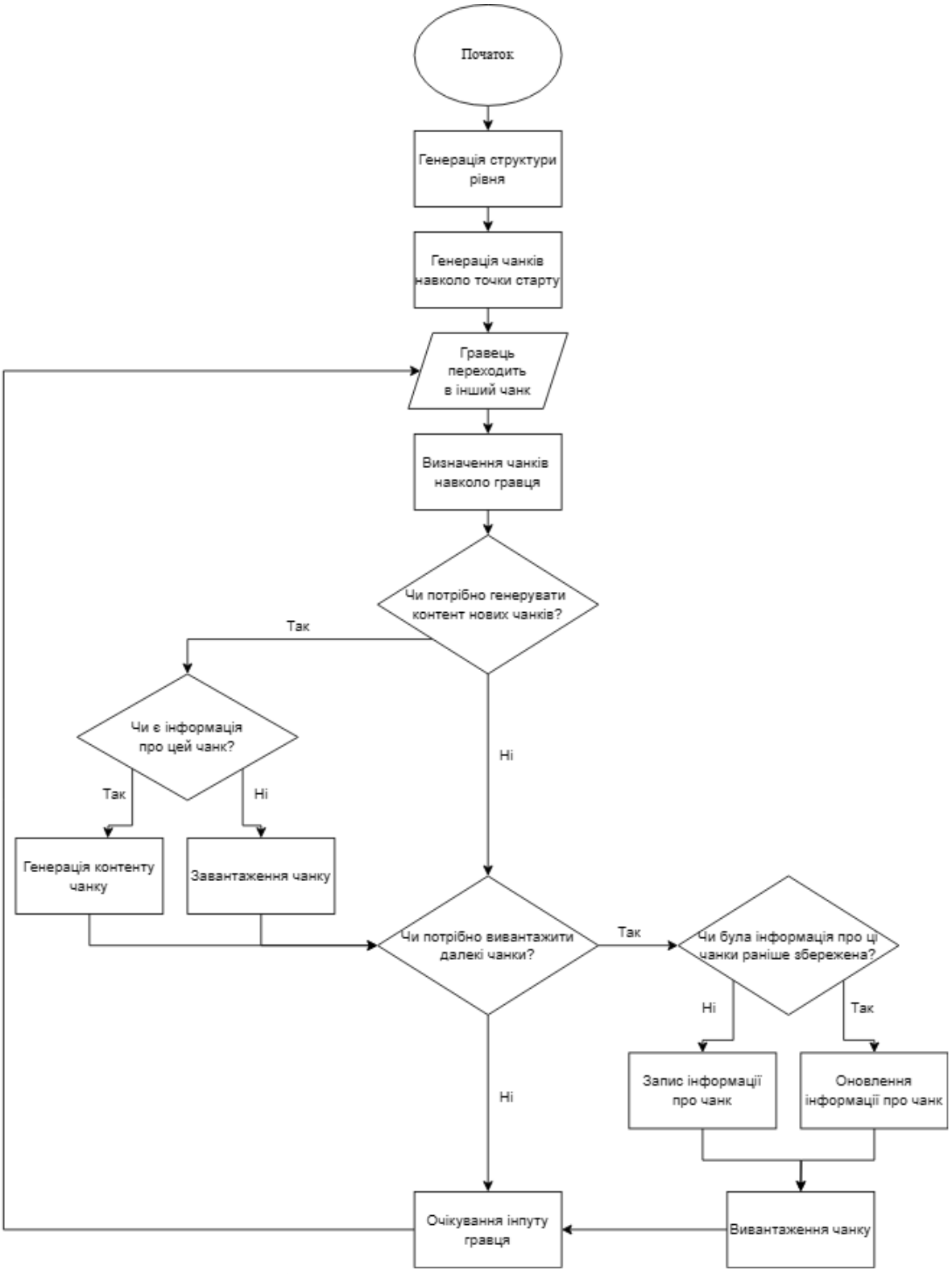

Рис. 1. Удосконалений алгоритм паралельної генерації графічних об'єктів на основі чанкового підходу

Відповідно до адаптованого закону Амдала, при звичайній генерації загальна частка послідовних обчислень p_{full} складається з:

$$p_{full} = p_{init} + p_{topo} + p_{depend} + p_{check} \quad (3)$$

де:

- p_{init} — частка ініціалізації структур даних;
- p_{topo} — частка обчислень загальної топології;
- p_{depend} — частка обробки взаємозалежностей;
- p_{check} — частка ініціалізації структур даних.

В удосконаленому алгоритмі частка послідовних обчислень p_{chunk} зменшується до:

$$p_{chunk} = p_{init\ chunk} + p_{local\ depend} + p_{boundary} \quad (4)$$

де:

- $p_{init\ chunk}$ — ініціалізація для окремого чанка;
- $p_{local\ depend}$ — обробка локальних залежностей;
- $p_{boundary}$ — обчислення на границях.

Оскільки $p_{init\ chunk} < p_{init}$, а $p_{local\ depend} < p_{depend}$ і з'являється лише невелика додаткова складова $p_{boundary}$, загальна частка послідовних обчислень p_{chunk} менша за p_{full}. Відповідно до закону Амдала, це збільшує максимально можливе теоретичне прискорення.

Для оцінки ефективності удосконаленого алгоритму було проведено серію експериментів з генерації різних типів графічних об'єктів у Godot Engine. Бенчмаркінг включав вимірювання частки послідовних обчислень та прискорення при збільшенні кількості потоків для визначення відповідності теоретичним межам закону Амдала.

Процес бенчмаркінгу реалізовано з використанням вбудованого інструментарію Godot Engine. Створено тестові сцени з чотирма типами графічних об'єктів: регулярні та складні структури у 2D і 3D форматах. Налаштовано однакові параметри генерації для забезпечення порівнянності результатів. Вимірювання проводилися за допомогою вбудованого профайлера Godot, активованого через меню Debug > Profiler. Було налаштовано режим High Detail для відстеження часу виконання окремих функцій та додано маркери часу для фіксації початку та кінця кожного етапу генерації. Зібрано метрики виконання для різної кількості потоків (1, 2, 4, 8).

Для визначення частки послідовних обчислень у Godot Engine було проведено декомпозицію алгоритму на логічні фази та послідовні компоненти з використанням наступного процесу:

- Маркування блоків коду — додавання спеціальних викликів профайлера на початку та в кінці кожного логічного блоку генерації:
  - етап ініціалізації структур даних;
  - етап розрахунку глобальної топології;
  - етап генерації окремих чанків;
  - етап фінальної перевірки та інтеграції чанків.
- Запуск тестів з поступовим збільшенням кількості потоків (1, 2, 4, 8) при незмінних інших параметрах генерації, з фіксацією часу виконання кожного блоку.
- Ідентифікація послідовних компонентів через аналіз незмінності часу виконання при збільшенні кількості потоків - блоки, час яких залишається майже однаковим незалежно від кількості потоків, класифікуються як послідовні.

– Розрахунок загальної частки послідовних обчислень за формулою: p = (сума часу всіх послідовних блоків) / (загальний час виконання на одному потоці).
– Частка послідовних обчислень розраховувалася за формулою:

$$p = \frac{T_{seq}}{T_{total}} \quad (5)$$

де:
– T_{seq} — час виконання послідовних компонентів;
– T_{total} — загальний час виконання на одному потоці.

Таблиця 1

Прискорення генерації графічних об'єктів при збільшенні кількості потоків

Тип графічних об'єктів	Частка послідовних обчислень без чанкування (p)	Частка послідовних обчислень з чанкуванням (p)	Теоретичне максимальне прискорення без чанкування (1/p)	Теоретичне максимальне прискорення з чанкуванням (1/p)
2D (регулярна структура)	0.25	0.14	4	7.14
3D (регулярна структура)	0.28	0.16	3.57	6.25
2D (складна структура)	0.34	0.20	2.94	5.00
3D (складна структура)	0.40	0.24	2.50	4.17

Результати бенчмаркінгу підтверджують, що чанковий підхід зменшує частку послідовних обчислень (p) для всіх типів об'єктів, що позитивно впливає на теоретичну межу максимального прискорення згідно з законом Амдала. Важливо відзначити, що зі збільшенням кількості потоків понад 8–16 (залежно від типу процесора) спостерігається уповільнення зростання прискорення.

Впровадження запропонованого методу оптимізації в тестовому проекті підтвердило ефективність розробленого підходу, забезпечивши підвищення продуктивності без втрати якості генерованого контенту.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Паралельні обчислення для генерації графічних об'єктів залишаються критично важливим елементом оптимізації в сучасній розробці відеоігор, оскільки дозволяють створювати масштабні та деталізовані світи з високою продуктивністю. Огляд методів паралельної генерації графічних об'єктів у Godot Engine, проведений у цій статті, показав, що ключовим обмеженням є частка послідовних обчислень, описана законом Амдала.

Проведений аналіз алгоритмів генерації графічних об'єктів (шумові алгоритми, клітинні автомати, Wave Function Collapse та L-системи) виявив, що їх ефективність суттєво обмежується необхідністю послідовної обробки взаємозалежних елементів. Розроблений чанковий підхід дозволяє локалізувати залежності та зменшити частку

послідовних обчислень, що відповідно до закону Амдала підвищує максимально можливе прискорення.

Проте залишається ряд викликів, таких як оптимізація алгоритмів з високою взаємозалежністю елементів (особливо WFC та L-систем) та забезпечення стабільності при асинхронній генерації. Подальші дослідження у цій сфері можуть бути спрямовані на інтеграцію методів машинного навчання для автоматичної оптимізації параметрів чанкової генерації та адаптації до різних апаратних платформ.

Важливо також приділити увагу питанням масштабування розробленого підходу для мобільних платформ з обмеженими ресурсами, де ефективне використання багатоядерних процесорів є особливо актуальним. Перспективним напрямом є вдосконалення методів динамічного балансування навантаження між потоками та адаптивної зміни розміру чанків залежно від складності об’єктів, що дозволить досягти оптимальної продуктивності для широкого спектру ігрових сценаріїв.

Розроблені методи та підходи можуть бути корисними не лише для ігрової індустрії, але й для інших галузей, де використовується процедурна генерація контенту — від архітектурної візуалізації до моделювання природних явищ та процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gregg, C., & Hazelwood, K. (2011). Where is the data? Why you cannot debate CPU vs. GPU performance without the answer. *Software (ISPASS).* https://doi.org/10.1109/ispass.2011.5762730
2. Owens, J. D., et al. (2008). GPU Computing. *IEEE, 96(5),* 879–899. https://doi.org/10.1109/jproc.2008.917757
3. Nickolls, J., & Dally, W. J. (2010). The GPU Computing Era. *IEEE Micro, 30(2),* 56–69. https://doi.org/10.1109/mm.2010.41
4. Volkov, V., & Demmel, J. W. (2008). Benchmarking GPUs to tune dense linear algebra. 2008 SC. *International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX*. https://doi.org/10.1109/sc.2008.5214359
5. Hendrikx, M., et al. (2013). Procedural content generation for games. *ACM Transactions on Multimedia Computing, Communications, and Applications, 9(1),* 1–22. http://dx.doi.org/10.1145/2422956.2422957
6. Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural Content Generation in Games.* Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-42716-4
7. Togelius, J., et al. (2011). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games, 3(3),* 172–186. http://dx.doi.org/10.1109/tciaig.2011.2148116
8. Summerville, A., et al. (2018). Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games, 10(3),* 257–270. http://dx.doi.org/10.1109/tg.2018.2846639
9. Karth, I., & Smith, A. M. (2017). WaveFunctionCollapse is Constraint Solving in the Wild. *Proceedings of the 12th International Conference on the Foundations of Digital Games.* http://dx.doi.org/10.1145/3102071.3110566
10. Linden, R., Lopes, R., & Bidarra, R. (2021). Designing Procedurally Generated Levels. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 9.* http://dx.doi.org/10.1609/aiide.v9i3.12592
11. Sorenson, N., & Pasquier, P. (2010). Towards a Generic Framework for Automated Video Game Level Creation. *Applications of Evolutionary Computation,* 131–140. https://doi.org/10.1007/978-3-642-12239-2_14
12. Shaker, N., et al. (2016). Constructive generation methods for dungeons and levels. *Procedural Content Generation in Games,* 31–55. https://doi.org/10.1007/978-3-319-42716-4_3
13. Liapis, A., Yannakakis, G. N., & Togelius, J. (2015). Constrained Novelty Search: A Study on Game Content Generation. *Evolutionary Computation, 23(1),* 101–129. https://doi.org/10.1162/evco_a_00123
14. Shaker, N., Togelius, J., & Yannakakis, G. N. (2016). The experience-driven perspective. *Procedural Content Generation in Games,* 181–194. https://doi.org/10.1007/978-3-319-42716-4_10
15. Shaker, N., Smith, G., & Yannakakis, G. N. (2016). Evaluating content generators. *Procedural Content Generation in Games,* 215–224. https://doi.org/10.1007/978-3-319-42716-4_12

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (27), 2025
ISSN 2663 - 4023

Ihor Tovstochub
Postgraduate Student
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID ID: 0009-0007-4743-5334
i.tovstochub@stud.duikt.edu.ua

Olha Zinchenko
Doctor of Technical Sciences, Head of the Department of Artificial Intelligence
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID ID: 0000-0002-3973-7814
o.zinchenko@duikt.edu.ua

GENERATION OF GRAPHIC OBJECTS IN THE GODOT ENGINE USING PARALLEL COMPUTATION METHOD

Abstract. This article presents a study of graphic object generation methods in the Godot game engine using parallel computing. The main focus is on the efficient use of computational resources to improve the performance of graphic content generation in dynamic game environments. The research conducted a detailed analysis of four main categories of graphic object generation algorithms actively used in the Godot Engine ecosystem: noise-based algorithms (Perlin, OpenSimplex), which are used to create natural landscapes and textures; cellular automata, which allow modeling organic structures; Wave Function Collapse (WFC), which is used to generate structured levels with connection rules; and L-systems, which are effectively used to generate fractal structures, particularly vegetation. For each type of algorithm, the potential for parallelization and factors limiting their efficiency on multi-core systems were determined. The problem identified during the analysis is the presence of a significant proportion of sequential computations in generation algorithms, which, according to Amdahl's Law, substantially limits the maximum possible speedup even when increasing the number of computational cores. A mathematical model is proposed for estimating potential speedup in the context of graphic object generation, allowing more accurate prediction of the performance of different algorithms on multi-core systems. As part of the research, an innovative optimization approach was developed, based on the chunk generation method, where graphic objects are divided into spatially localized elements with minimal inter-element dependencies. The key feature of the proposed approach is asynchronous chunk generation and localization of coordinate calculations, which significantly reduces the proportion of sequential operations and, consequently, greatly increases the potential speedup according to Amdahl's Law. For an objective evaluation of the developed method's effectiveness, a series of experiments was conducted, including benchmarking on different types of graphic objects using the built-in Godot Engine tools. The measurement methodology was based on decomposing the algorithm into logical phases and analyzing changes in execution time with a gradual increase in the number of threads. The experimental results demonstrated a significant reduction in the proportion of sequential computations for all types of graphic objects when applying the chunk approach, confirming the effectiveness of the proposed method. The research results have practical significance for optimizing game development based on the Godot Engine and can be adapted for other visualization systems with similar architecture.

Keywords: graphic object generation; Godot; noise algorithms; asynchronous generation; multithreading; benchmarking; parallel computation.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Gregg, C., & Hazelwood, K. (2011). Where is the data? Why you cannot debate CPU vs. GPU performance without the answer. *Software (ISPASS).* https://doi.org/10.1109/ispass.2011.5762730
2. Owens, J. D., et al. (2008). GPU Computing. *IEEE, 96(5),* 879–899. https://doi.org/10.1109/jproc.2008.917757
3. Nickolls, J., & Dally, W. J. (2010). The GPU Computing Era. *IEEE Micro, 30(2),* 56–69. https://doi.org/10.1109/mm.2010.41

4. Volkov, V., & Demmel, J. W. (2008). Benchmarking GPUs to tune dense linear algebra. 2008 SC. *International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX*. https://doi.org/10.1109/sc.2008.5214359
5. Hendrikx, M., et al. (2013). Procedural content generation for games. *ACM Transactions on Multimedia Computing, Communications, and Applications, 9(1),* 1–22. http://dx.doi.org/10.1145/2422956.2422957
6. Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural Content Generation in Games.* Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-42716-4
7. Togelius, J., et al. (2011). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games, 3(3),* 172–186. http://dx.doi.org/10.1109/tciaig.2011.2148116
8. Summerville, A., et al. (2018). Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games, 10(3),* 257–270. http://dx.doi.org/10.1109/tg.2018.2846639
9. Karth, I., & Smith, A. M. (2017). WaveFunctionCollapse is Constraint Solving in the Wild. *Proceedings of the 12th International Conference on the Foundations of Digital Games.* http://dx.doi.org/10.1145/3102071.3110566
10. Linden, R., Lopes, R., & Bidarra, R. (2021). Designing Procedurally Generated Levels. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 9.* http://dx.doi.org/10.1609/aiide.v9i3.12592
11. Sorenson, N., & Pasquier, P. (2010). Towards a Generic Framework for Automated Video Game Level Creation. *Applications of Evolutionary Computation,* 131–140. https://doi.org/10.1007/978-3-642-12239-2_14
12. Shaker, N., et al. (2016). Constructive generation methods for dungeons and levels. *Procedural Content Generation in Games,* 31–55. https://doi.org/10.1007/978-3-319-42716-4_3
13. Liapis, A., Yannakakis, G. N., & Togelius, J. (2015). Constrained Novelty Search: A Study on Game Content Generation. *Evolutionary Computation, 23(1),* 101–129. https://doi.org/10.1162/evco_a_00123
14. Shaker, N., Togelius, J., & Yannakakis, G. N. (2016). The experience-driven perspective. *Procedural Content Generation in Games,* 181–194. https://doi.org/10.1007/978-3-319-42716-4_10
15. Shaker, N., Smith, G., & Yannakakis, G. N. (2016). Evaluating content generators. *Procedural Content Generation in Games,* 215–224. https://doi.org/10.1007/978-3-319-42716-4_12