

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____Олексій СМІРНОВ
“_____” _____2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему

**“Програмне забезпечення апаратно-програмного комплексу
прогнозування траєкторії руху об’єктів з використанням алгоритму
Барнса-Хата”**

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____Макаренко М.О.
«_____» _____2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____Минайленко Р.М.
«_____» _____2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань ._12 _____“Інформаційні технології”
Спеціальність 123 “Комп’ютерна інженерія”
Освітньо-професійна (освітньо-наукова) програма “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф. _____Олексій СМІРНОВ
« » 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Макаренку Максиму Олеговичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення апаратно-програмного комплексу прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата*
2. Керівник роботи *Минайленко Роман Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу № 136-02 від 27.02.2026 року
3. Строк подання студентом роботи до захисту *16.05.2026 р.*
4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення апаратно-програмного комплексу прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата*
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
 1. *Призначення та область використання.*
 2. *Перегляд аналогічних існуючих систем.*
 3. *Опис і обґрунтування проектних рішень.*
 4. *Етапи програмування системи.*
 5. *Впровадження системи в промислову експлуатацію.*
 6. *Висновки*
6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

<i>Структурна схема системи</i>	<i>1 аркуш</i>
<i>Функціональна схема системи</i>	<i>1 аркуш</i>
<i>Діаграма процесів</i>	<i>1 аркуш</i>
<i>Блок-схема алгоритму роботи додатку</i>	<i>2 аркуша</i>

7. Дата видачі завдання « » 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	16.05.2026 р.	

Дата видачі завдання
« » р.

Підпис керівника

Минайленко Р.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« » р.

Підпис здобувача

Макаренко М.О.
(прізвище та ініціали)

АНОТАЦІЯ

Макаренко М.О. Програмне забезпечення апаратно-програмного комплексу прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначене для прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата.

Метою розробки є програмне забезпечення прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата та паралельної обробки.

Результат роботи – програмна реалізація системи прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата та паралельної обробки.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11

Програму реалізовано на базі фреймворку .NET 8.

Ключові слова: комп'ютерна інженерія, задача N-тіл, алгоритм Барнса-Хата, GPU, паралельні обчислення, .NET.

ABSTRACT

Makarenko M.O. Software of a hardware-software complex for forecasting the trajectory of objects using the Barnes-Hut algorithm. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this graduation qualification work for the first (bachelor's) level of higher education, software is developed that is intended for forecasting the trajectory of objects using the Barnes-Hut algorithm.

The purpose of the development is software for forecasting the trajectory of objects using the Barnes-Hut algorithm and parallel processing.

The result of the work is a software implementation of a system for forecasting the trajectory of objects using the Barnes-Hut algorithm and parallel processing.

In the process of working on the software model, an analysis of existing hardware and software tools was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program is developed based on the .NET 8 framework.

Keywords: computer engineering, N-body problem, Barnes-Hut algorithm, GPU, parallel computing, .NET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 Призначення системи.....	5
1.2 Область застосування	6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	6
2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування	15
2.3 Розгорнута постановка завдання	17
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	21
3.1 Опис функціонування системи	21
3.2 Розробка структурної схеми.....	24
3.3 Розробка функціональної схеми.....	26
3.4 Розробка діаграми процесів	28
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ	31
4.1 Блок-схеми та опис алгоритмів функціонування системи.....	31
4.2 Захист розробленого програмного забезпечення.....	49
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	51
6 ОСНОВНІ ВИСНОВКИ.....	52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56

					<i>ВКРБ-123.26.0048.00.00.ПЗ</i>			
Змн.	Арк.	№ докум.	Підпис	Дата	Програмне забезпечення апаратно-програмного комплексу прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата	Літ.	Арк.	Акрушіє
Розробив		Макаренко М.О.				Б		
Перевірів		Минайленко Р.М.						59
						ЦНТУ КІ-22-2		
Н. контр.		Коваленко А.С.						
Затв.		Смірнов О.А.						

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AABB	—	обмежувальний паралелепіпед, грані якого паралельні осям
API	—	набір правил і протоколів для взаємодії програмних систем
CPU	—	центральний процесор
GC	—	автоматична система збирання сміття
GPU	—	графічний процесор
LBVH	—	лінійна ієрархія обмежувальних об'ємів
MAC	—	критерій мультипольного прийняття
SIMD	—	виконання однієї інструкції над масивом даних одночасно
SSBO	—	буфер для обміну масивами даних між CPU та GPU
VRAM	—	відеопам'ять графічного адаптера
Θ (Тета)	—	коефіцієнт, що визначає баланс точності та швидкості

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

ВСТУП

Актуальність теми. Класична задача N-тіл є фундаментальною проблемою небесної механіки та астрофізики, що описує рух об'єктів під дією гравітації. Незважаючи на теоретичну вивченість, практична реалізація таких моделей стикається з серйозним обчислювальним бар'єром: метод прямого підсумування (Brute Force) має квадратичну складність $O(N^2)$, що робить моделювання систем з великою кількістю тіл (галактик, зоряних скупчень) надзвичайно часозатратним.

Сучасний етап розвитку інформаційних технологій характеризується переходом від нарощування тактової частоти процесорів до глибокої паралелізації обчислень. Тому ефективність роботи зумовлена необхідністю поєднанні двох підходів:

- Використання ієрархічних структур даних (дерева Octree або Quadtree) в алгоритмі Барнса-Хата для зниження складності до $O(N \log N)$.

- Використання потужностей GPU через обчислювальні шейдери та SIMD інструкцій CPU.

Розробка такої системи на платформі .NET 8 дозволяє поєднати низькорівневу оптимізацію пам'яті з гнучкою архітектурою. Раніше такий рівень контролю був переважно у мов C/C++, але сучасний .NET дає змогу створювати не менш ефективні рішення.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата.

Для досягнення поставленої мети визначено наступні завдання:

- Огляд існуючих систем прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

– Дослідження системи прогнозування траєкторії руху об’єктів з використанням прямого сумування та алгоритму Барнса-Хата.

– Реалізувати програмний комплекс, зокрема модулі для розрахунку сил за допомогою багатопоточності, SIMD та GPU.

– Провести експериментальне порівняння продуктивності різних комбінацій алгоритмів та параметрів.

Практична цінність отриманих результатів полягає у створенні універсального інструменту для високопродуктивного моделювання фізичних процесів, який може використовуватися в астрофізиці для моделювання зоряних скупчень, у навчальному процесі для візуалізації складних фізичних взаємодій у реальному часі для покращення розуміння динамічних систем, а також як еталонний інструмент для тестування продуктивності паралельних обчислювальних систем.

Таким чином, розробка системи прогнозування траєкторії руху об’єктів з використанням алгоритму Барнса-Хата є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система, що розробляється у бакалаврському проекті, призначена для моделювання та прогнозування траєкторій руху великої кількості об'єктів, що взаємодіють між собою за законом всесвітнього тяжіння (задача N-тіл). Метою бакалаврського проекту є створення програмного комплексу, який дозволяє проводити фізичні розрахунки в реальному часі за допомогою алгоритму Барнса-Хата та сучасних методів апаратного прискорення. На даний час використання традиційних методів прямого розрахунку призводить до значних труднощів, а саме:

- Експоненціального зростання обчислювальних витрат при збільшенні кількості тіл.
- Неможливість візуалізувати великі системи на звичайних комп'ютерах.
- Відсутність зручних інструментів для порівняння різних способів обчислень.

Для досягнення зазначених цілей під час проектування були вирішені наступні завдання:

- Проведено аналіз математичних моделей та алгоритмів вирішення задачі N-тіл;
- Розроблено архітектуру системи для візуалізації фізичних процесів у реальному часі;
- Реалізовано алгоритм Барнса-Хата, що дозволило знизити обчислювальну складність задачі з $O(N^2)$ до $O(N \log N)$;
- Розроблено підсистему паралельних обчислень з використанням багатопоточності та SIMD-інструкцій;

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

– Розроблено модуль розрахунків на базі GPU з використанням технології Compute Shaders;

– Створено інструментарій для профілювання продуктивності, бенчмаркінгу та відлагодження точності обчислень.

Платформа розроблена на базі фреймворку .NET 8 (LTS) та бібліотеки OpenTK, що забезпечує низькорівневу взаємодію з GPU на основі специфікації OpenGL 4.5.

Максимальна швидкодія системи досягається завдяки комплексному поєднанню потужностей центрального та графічного процесорів. На рівні CPU задіяно бібліотеку Task Parallel Library для ефективного розпаралелювання процесів побудови та обходу вузлів дерева, а також інструкції SIMD, що прискорюють векторні математичні операції безпосередньо на рівні апаратури. Ключовий приріст продуктивності забезпечує використання обчислювальних шейдерів, які дозволяють перенести найбільш ресурсомісткі розрахунки на сучасні відеокарти з уніфікованою архітектурою, забезпечуючи масивний паралелізм обчислень.

Програмний комплекс оснащений засобами профілювання продуктивності алгоритмів, аналітики рендерингу та графічним інтерфейсом, який дозволяє змінювати параметри симуляції безпосередньо під час її роботи. Таким чином, розробка системи прогнозування траєкторії руху об'єктів з використанням алгоритму Барнса-Хата є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1.2 Область застосування

Розроблена система має широкий потенціал для впровадження у наукових, освітніх та інженерних галузях, де критично важливо поєднувати точність фізичного моделювання з високою швидкістю обчислень. У першу

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

чергу, комплекс орієнтований на потреби астрофізики та космології, оскільки дозволяє відтворювати динаміку еволюції галактик, зоряних скупчень та складних планетарних систем. Алгоритм Барнса-Хата дає можливість вивчати зіткнення галактик або рух планет на звичайних комп'ютерах. Раніше для таких масштабних завдань були потрібні суперкомп'ютери, проте тепер ці процеси можна моделювати завдяки значному прискоренню розрахунків. Це дозволяє спостерігати за складними космічними явищами в реальному часі без використання дорогого обладнання.

Окрім чисто наукового застосування, система є ефективним інструментом для освітньої діяльності. Вона може використовуватися як інтерактивний посібник у закладах вищої освіти для візуалізації законів небесної механіки та вивчення принципів паралельних обчислень. Студенти мають змогу наочно спостерігати, як абстрактні ієрархічні структури даних, зокрема дерева Oстree, дозволяють оптимізувати складні математичні задачі. Водночас програмний комплекс виконує роль еталонного інструменту для тестування продуктивності сучасного апаратного забезпечення. Використання SIMD-інструкцій та обчислювальних шейдерів робить систему якісним бенчмарком для оцінки роботи багатоядерних процесорів та графічних прискорювачів у задачах із неструктурованими даними.

Гнучкість математичного ядра системи дозволяє адаптувати її принципи для суміжних дисциплін, наприклад, для молекулярної динаміки. Оскільки алгоритми розрахунку далекодійних сил є універсальними, їх можна використовувати для моделювання взаємодії атомів і молекул при розробці нових матеріалів чи фармацевтичних препаратів. Крім того, напрацювання проєкту мають значну цінність для індустрії розробки комп'ютерних ігор та візуальних ефектів. Інтеграція ієрархічних методів розрахунку у фізичні рушії дозволяє створювати масштабні системи частинок і гравітаційні аномалії в реальному часі, зберігаючи високу продуктивність навіть на споживчих відеокартах.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Сучасна комп'ютерна інженерія дозволяє переходити від теоретичного аналізу динамічних систем до їх точного моделювання. Задача N-тіл, що полягає у прогнозуванні руху частинок під дією гравітації, залишається однією з найбільш ресурсомістких в обчислювальній фізиці. Аналіз існуючих рішень дозволяє класифікувати їх за алгоритмічною базою, архітектурою обчислень та програмними засобами реалізації.

GADGET-4

Одним із найбільш досконалих інструментів у цій галузі є GADGET-4 (GAlaxies with Dark matter and Gas intERacT) – масивно-паралельна система для космологічних симуляцій. Система є розвитком ідей GADGET-2/3 та використовує методи ієрархічного обчислення гравітації, що безпосередньо корелює з предметом даної роботи.

Алгоритмічні та технічні особливості:

- GADGET-4 використовує алгоритм TreePM, де далекодіючі сили розраховуються за допомогою методу Particle-Mesh (на базі швидкого перетворення Фур'є – FFT), а взаємодія на коротких відстанях – через ієрархічне дерево.

- Код підтримує стандарт MPI-3 для обміну повідомленнями між вузлами та використовує спільну пам'ять всередині багатоядерних вузлів.

- Для максимальної продуктивності система підтримує явну векторизацію обчислювальних ядер через інструкції AVX.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

– Система дозволяє моделювати як класичний ньютонівський простір, так і розширення всесвіту в комобільних координатах.

Прикладом практичного застосування системи є моделювання формування великомасштабної структури всесвіту в межах космологічної моделі Λ CDM (Рисунок 2.1). На знімку візуалізовано розподіл темної матерії, що утворює масивні скупчення, демонструючи здатність системи обробляти колосальні масиви даних гравітаційних взаємодій.

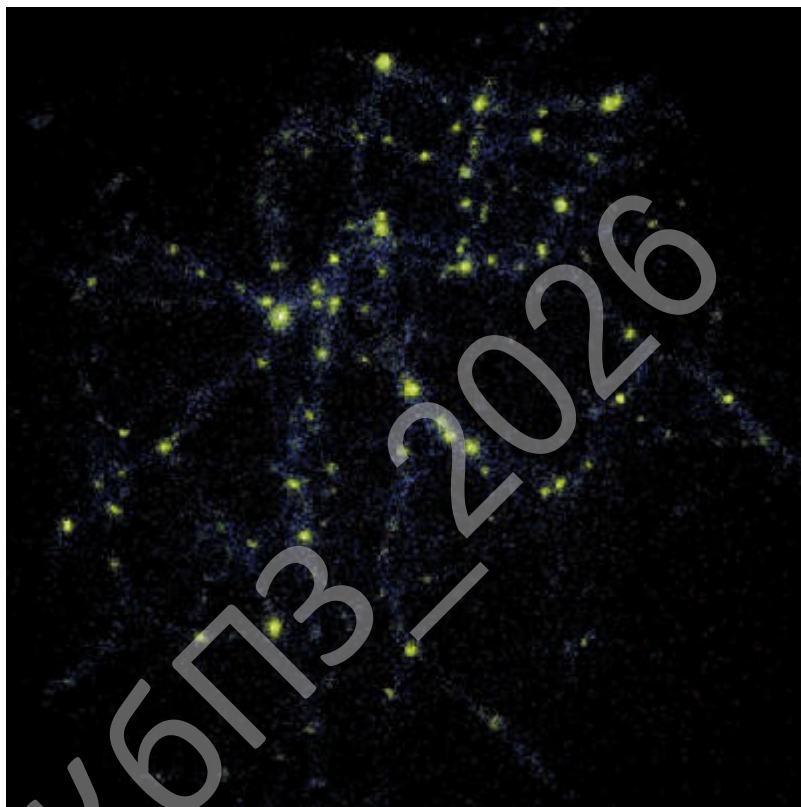


Рисунок 2.1 – Моделювання еволюції структури Всесвіту (GADGET-4).

Незважаючи на значну обчислювальну потужність, GADGET-4 має ряд специфічних особливостей, які роблять його незручним для створення прикладних інтерактивних систем. Основна складність полягає у процесі розгортання та конфігурації: програма вимагає наявності специфічного Linux-оточення та багатьох сторонніх бібліотек, а керування її роботою здійснюється через статичні файли, що редагуються перед компіляцією або запуском. Такий підхід повністю виключає інтерактивність, оскільки система орієнтована на

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

тривалу пакетну обробку даних без можливості втручання користувача в процес. Результати розрахунків зберігаються у форматі Snapshot (переважно HDF5), що вимагає подальшої обробки стороннім програмним забезпеченням для отримання візуальних результатів. Крім того, хоча код і демонструє виняткову ефективність на центральних процесорах, у ньому відсутня вбудована підтримка рендерингу в реальному часі через сучасні графічні API. Це створює суттєвий розрив між процесом обчислення та візуалізацією, що робить використання GADGET-4 недоцільним для освітніх або ігрових симуляторів, де пріоритетом є безперервний зворотний зв'язок та мобільність рішення. Таким чином, для розробки інтерактивного комплексу на базі .NET 8 необхідний інший підхід, який дозволить інтегрувати алгоритмічну потужність ієрархічних методів безпосередньо в графічний конвеєр GPU.

PKDGRAV3

Ще одним вагомим програмним рішенням, яке потребує детального аналізу, є комплекс PKDGRAV3. Ця система розроблена як високоефективний інструмент для проведення масових космологічних симуляцій та дослідження динаміки планетних систем. Архітектурно PKDGRAV3 орієнтована на гетерогенні обчислювальні середовища та відзначається здатністю ефективно використовувати потужності графічних прискорювачів для прискорення гравітаційних розрахунків.

В основі обчислювального ядра лежить використання швидкого мультипольного методу (FMM), який дозволяє розраховувати сили з високою точністю, зберігаючи при цьому лінійну або близьку до лінійної обчислювальну складність. Для просторової декомпозиції система використовує ієрархічні структури, що базуються на деревах f-субдивізії. Цей підхід є еволюційним розвитком принципів, закладених у класичному алгоритмі Барнса-Хата, але забезпечує вищу ефективність на масових паралельних архітектурах завдяки оптимізованому обходу вузлів та кращому балансуванню навантаження.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

Важливою особливістю PKDGRAV3 є підтримка симуляцій з величезною кількістю об'єктів. На рисунку 2.2 наведено приклад візуалізації розподілу матерії у Всесвіті, де завдяки оптимізації алгоритмів вдалося змодельовати систему, що налічує 2 трильйони частинок. Це підтверджує здатність рушія обробляти дані надвисокої роздільної здатності, ілюструючи формування філаментів темної матерії та гравітаційних скупчень у масштабах усього видимого Всесвіту. Продуктивність системи безпосередньо залежить від низькорівневої прив'язки до апаратної специфіки суперкомп'ютерів, що дозволяє PKDGRAV3 демонструвати високу швидкість обчислень навіть при екстремальних навантаженнях.

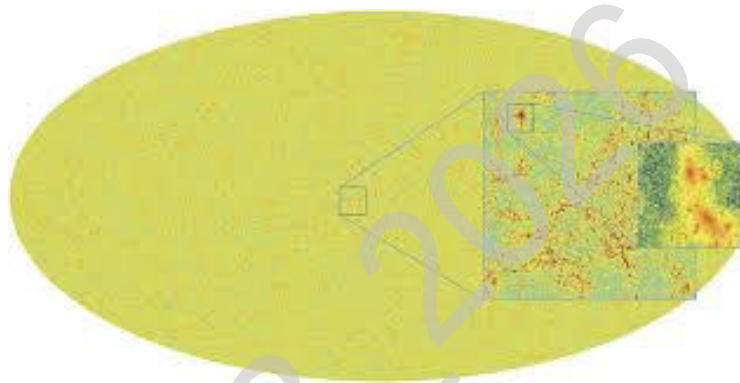


Рисунок 2.2 – Розподіл матерії у Всесвіті (2 трлн частинок, PKDGRAV3).

Незважаючи на видатні технічні показники, застосування PKDGRAV3 як прототипу для поточної розробки має суттєві обмеження. Архітектура системи є жорстко оптимізованою під конкретну фізичну модель та вибраний математичний апарат, що робить її досить статичним інструментом. У PKDGRAV3 практично відсутня можливість гнучкої заміни методів розрахунку (наприклад, перемикання між прямим методом та ієрархічним деревом) для проведення порівняльного аналізу без глибокого переписування вихідного коду.

Крім того, система традиційно орієнтована на тривалі пакетні розрахунки, які запускаються через командний рядок або системи керування чергами на кластерах, і не містить засобів для інтерактивного маніпулювання параметрами в реальному часі. У рамках цієї кваліфікаційної роботи програмне

забезпечення має іншу мету: створити гнучку платформу на базі .NET 8, де користувач може миттєво перемикає алгоритми та спостерігати за зміною продуктивності та точності безпосередньо під час симуляції. Така адаптивність і наявність інтерактивної візуалізації на локальних станціях роблять розроблюваний комплекс зручнішим для прикладних досліджень та наочного аналізу траєкторій, ніж вузькоспеціалізовані наукові системи.

REBOUND

На відміну від масштабних систем-монолітів, REBOUND – це відкрита програмна бібліотека, розроблена спеціально для глибокого дослідження орбітальної динаміки. Ключовою особливістю цього інструменту є його висока гнучкість та орієнтованість на прикладного дослідника. Бібліотека легко інтегрується як пакет для мови Python або підключається як нативна C-бібліотека, що робить її зручною для розробки власних скриптів та моделей.

Важливою перевагою REBOUND є наявність набору вбудованих інтеграторів, таких як WHFast або високоточний IAS15. Це дозволяє користувачеві вибрати оптимальний математичний апарат залежно від специфіки задачі – від швидкого моделювання планетних систем до розрахунків із мінімальною похибкою. Активна спільнота розробників підтримує проєкт великою кількістю готових сценаріїв, що демонструють приклади моделювання сонячних систем або динаміку зіткнень екзопланет. Типовий приклад роботи фреймворку – інтеграцію орбіт багатопланетної системи – наведено на рисунку 2.3. Завдяки досконалому математичному ядру, бібліотека дозволяє розраховувати стабільність орбіт на значних часових проміжках без суттєвої втрати точності.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

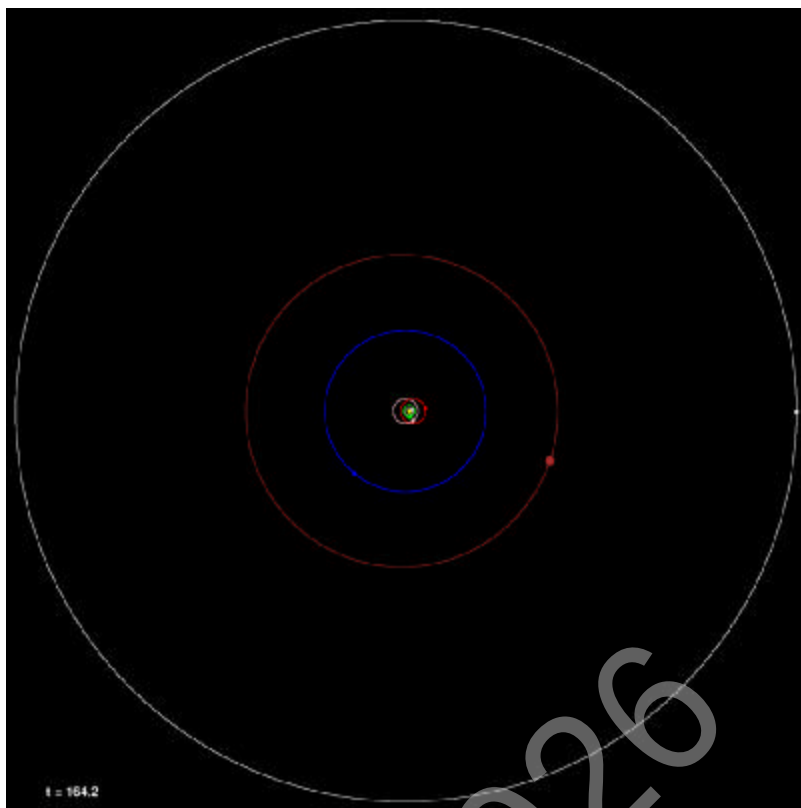


Рисунок 2.3 – Моделювання динаміки планетних орбіт у REBOUND.

Проте, незважаючи на легкість використання та наявність зручного API, REBOUND оперує переважно потужностями CPU. Це накладає певні обмеження на швидкість обробки надвеликих систем частинок. Крім того, для відображення результатів користувачеві доводиться залучати сторонні бібліотеки (наприклад, Matplotlib), які генерують статичні графіки або рендерять анімацію лише після повного завершення розрахунків. Відсутність вбудованого 3D-рушія з прямою підтримкою графічних API (таких як OpenGL) унеможливлює використання цього інструменту для інтерактивної візуалізації мільйонів об'єктів у реальному часі.

У межах даної кваліфікаційної роботи планується усунути цей розрив. Проєктована система на базі .NET 8 та OpenTK не просто виконуватиме математичні розрахунки за допомогою алгоритму Барнса-Хата, а й забезпечуватиме миттєву візуалізацію результатів через графічний конвеєр. Це дозволить поєднати точність, характерну для таких інструментів як REBOUND,

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

із динамічністю сучасних графічних систем, створюючи умови для безперервного інтерактивного дослідження фізичних процесів.

Огляд підходів до апаратного прискорення алгоритмів N-тіл

Окрім готових програмних комплексів, особливої уваги заслуговують сучасні методи апаратного прискорення задачі N-тіл. Класичний алгоритм Барнса-Хата, попри ефективність просторового розбиття, базується на рекурсивному обході структур даних. Хоча такий підхід є природним для CPU, його адаптація під масово-паралельну архітектуру GPU супроводжується значними складнощами. Відсутність апаратного стеку рекурсії на GPU та необхідність постійної синхронізації потоків під час побудови дерева часто призводять до зниження продуктивності порівняно з теоретично очікуваними показниками.

Для розв'язання проблеми побудови просторових ієрархій на графічних процесорах сучасні дослідники пропонують використання LBVN. Найбільш ефективним рішенням у цьому напрямку є алгоритм максимізації паралелізму, запропонований Т. Каррасом. Його підхід базується на квантуванні простору за допомогою кодів Мортон, що дозволяє звести складну задачу просторового розбиття до одновимірного сортування масиву цілих чисел. Ключовою перевагою методу Карраса є можливість паралельного формування топології дерева шляхом порівняння бітових префіксів відсортованих кодів, що повністю виключає необхідність рекурсії та блокування пам'яті. Саме ця технологія визначена як найбільш перспективна для реалізації ядра розроблюваної системи.

За результатами проведеного аналізу існуючих рішень та технологій можна стверджувати, що спеціалізовані системи, такі як GADGET-4 та PKDGRAV3, залишаються еталонними за продуктивністю. Проте вони орієнтовані виключно на кластерні обчислення, складні у розгортанні та позбавлені засобів інтерактивної візуалізації. З іншого боку, дослідницькі бібліотеки, наприклад REBOUND, пропонують високу точність та зручність

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

використання, але не забезпечують достатньої швидкодії для моделювання масштабних систем у реальному часі на локальних станціях.

Саме тому розробка власного програмного забезпечення на базі екосистеми .NET 8 та графічного API OpenTK є актуальним та науково обґрунтованим завданням. Такий підхід дозволить створити гнучкий інструмент, що поєднує високу швидкість обчислень (за рахунок використання алгоритму Карраса для GPU) та інтерактивність (через прямий рендеринг за допомогою OpenGL). Кінцевим результатом стане адаптивне рішення, яке забезпечує високу продуктивність симуляції без вимог до складної серверної інфраструктури, що відкриває нові можливості для прикладних досліджень та візуалізації складних динамічних систем.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

В якості основної платформи для розробки програмного комплексу обрано екосистему .NET 8 та мову програмування C#. Такий вибір зумовлений здатністю платформи поєднувати високу швидкість написання коду, властиву високорівневим мовам, із потужними інструментами для низькорівневої оптимізації. Завдяки механізмам небезпечного коду (unsafe) та можливості прямого керування пам'яттю, розробник отримує контроль над ресурсами, який зазвичай доступний лише у мовах C або C++. Використання саме восьмої версії .NET обґрунтовано її статусом LTS, що гарантує стабільність архітектури та тривалу підтримку системи.

Важливою перевагою обраної платформи для математичного моделювання є підтримка технології Ahead-of-Time (Native AOT) компіляції. На відміну від стандартного підходу, де код оптимізується безпосередньо під час виконання, Native AOT дозволяє транслювати програму в машинний код ще на етапі збірки. Це повністю усуває затримки на динамічну компіляцію під час

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

симуляції, забезпечує миттєвий запуск та робить споживання оперативної пам'яті більш передбачуваним. У результаті система досягає максимальної апаратної продуктивності, що критично важливо при розрахунку траєкторій великої кількості об'єктів.

Архітектурні засоби та проектування

Висока складність симуляції задачі N-тіл вимагає суворого дотримання інженерних принципів чистої архітектури та стандартів SOLID. Екосистема .NET надає для цього зручний інструментарій у вигляді вбудованого контейнера ін'єкції залежностей (Dependency Injection). Використання цього механізму дозволило спроектувати систему модульно, повністю відокремивши математичне ядро від графічного конвеєра та модулів введення-виведення даних.

Завдяки впровадженню поведінкового патерна «Strategy» та використанню абстрактних інтерфейсів, архітектура програми дозволяє легко змінювати методи розрахунку. Це дає змогу системі динамічно перемикає обчислювальні алгоритми – наприклад, миттєво переходити від розрахунків на CPU до використання обчислювальних шейдерів відеокарти – без зупинки або перезавантаження симуляції. Такий рівень абстракції не лише підвищує надійність програми, а й значно спрощує процес профілювання та порівняння точності різних методів у реальному часі.

Багатопотоковість та прямий доступ до SIMD-інструкцій

Для досягнення максимальної швидкодії на центральному процесорі застосовано комплексний підхід, що поєднує паралелізацію та векторизацію обчислень. Оскільки розрахунок гравітаційних сил за своєю природою є масивно-паралельною задачею, для ефективного розподілу навантаження між усіма ядрами процесора використано бібліотеку Task Parallel Library (TPL). Внутрішні механізми TPL дозволяють динамічно та рівномірно завантажувати обчислювальні потужності навіть тоді, коли частинки у просторі розподілені неоднорідно.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

Додатково, для прискорення математичних операцій у межах кожного окремого потоку, реалізовано прямий доступ до SIMD-інструкцій через простір імен `System.Runtime.Intrinsics.X86`. Це дає можливість вручну керувати векторними регістрами процесора, залучаючи розширені набори інструкцій AVX2 та FMA. Завдяки такому підходу складні операції, як-от обчислення відстаней між об'єктами або знаходження зворотного квадратного кореня, виконуються одночасно для восьми частинок за один такт процесора. Це дозволяє радикально пришвидшити найбільш витратні цикли симуляції.

Робота з пам'яттю

Окрему увагу при розробці приділено стратегії керування пам'яттю, оскільки класична об'єктно-орієнтована модель створює значне навантаження на GC та призводить до фрагментації оперативної пам'яті. Щоб уникнути цих проблем, було застосовано парадигму Data-Oriented Design (DOD). Мова C# дозволяє проектувати легковажні типи даних (struct) без зайвих накладних витрат та керувати ними за допомогою вказівників у блоках unsafe коду.

Дані про фізичні частинки розміщуються у вигляді суцільних, неперервних блоків пам'яті у форматах AoS (Array of Structures) або SoA (Structure of Arrays). Це повністю усуває потребу в динамічному виділенні об'єктів у купі та мінімізує втручання GC у процес роботи програми. Такий підхід максимально ефективно використовує кеш-пам'ять процесора рівнів L1 та L2, оскільки дані розташовані послідовно. Це зводить до мінімуму кількість промахів кешу під час ітеративного обходу великих масивів частинок, що критично важливо для стабільної роботи симулятора на високих швидкостях.

Графічний стек та GPU-обчислення

Для реалізації масивно-паралельних обчислень на графічному процесорі було проаналізовано пропрієтарну архітектуру NVIDIA CUDA та відкриті кросплатформні стандарти. У результаті було прийнято рішення відмовитися від CUDA на користь більш універсальних рішень. По-перше, використання пропрієтарної технології жорстко прив'язує продукт до заліза одного виробника,

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

що обмежує аудиторію користувачів. По-друге, інтеграція CUDA вимагає специфічного інструментарію та складних механізмів синхронізації пам'яті для виведення результатів на екран. Хоча низькорівневі API, як Vulkan чи DirectX 12, надають максимальний контроль над ресурсами, вони вимагають значних витрат часу на написання шаблонного коду. Оптимальним вибором став стандарт OpenGL 4.5, інтегрований через бібліотеку OpenTK, оскільки він забезпечує ідеальний баланс між продуктивністю та простотою реалізації.

Замість класичних GPGPU методів у системі використано обчислювальні шейдери, які працюють у єдиному контексті з графічним API. Завдяки використанню SSBO, вдалося досягти безшовної інтеграції обчислень та рендерингу. Обчислювальний шейдер оновлює позиції та швидкості частинок безпосередньо у VRAM, після чого графічний конвеєр миттєво використовує ці ж дані для відмальовки. Це повністю усуває потребу щокандрово передавати гігабайти даних між оперативною пам'яттю та відеокартою через шину PCI-Express, що ліквідує найголовніше вузьке місце гібридних симуляцій.

Графічний інтерфейс користувача (GUI)

Для побудови інтерфейсу користувача, панелі керування та засобів профілювання обрано бібліотеку ImGui.NET. Вибір на користь цієї технології зумовлений її парадигмою Immediate Mode GUI (IMGUI), яка є індустріальним стандартом для розробки інженерного та високопродуктивного програмного забезпечення. На відміну від класичних інтерфейсів (WPF або Windows Forms), що будують складні дерева об'єктів у пам'яті та вимагають постійної синхронізації зі станом програми, підхід IMGUI працює значно ефективніше в умовах динамічної симуляції.

Парадигма Immediate Mode не зберігає стан віджетів між кадрами і не створює зайвих об'єктів у купі, що мінімізує навантаження на збирач сміття та запобігає мікрозатримкам. Інтерфейс формується динамічно за один прохід безпосередньо поверх OpenGL контексту як графічний оверлей. Це гарантує нульову затримку взаємодії та дозволяє користувачеві змінювати параметри

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

середовища, перемикає алгоритми та миттєво аналізує телеметрію під час виконання програми. Такий підхід забезпечує стабільну роботу графічної оболонки навіть при максимальному навантаженні на обчислювальне ядро симулятора.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для високопродуктивного моделювання гравітаційної взаємодії великої кількості космічних тіл (задача N-тіл) з використанням паралельних обчислень на CPU та GPU.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих алгоритмічних рішень та програмних систем для моделювання задачі N-body з метою виявлення переваг та недоліків методів прямого розрахунку та ієрархічних методів (алгоритм Барнса-Хата). Результати аналізу врахувати при проектуванні архітектури обчислювальних рушіїв;

б) вибрати та обґрунтувати методику побудови системи, що підтримує динамічну підміну обчислювальних стратегій (патерн «Strategy») для порівняльного аналізу продуктивності. Розробити структурну схему взаємодії графічного інтерфейсу, системи управління даними частинок та обчислювальних модулів;

в) розробити програмне забезпечення системи на базі фреймворку .NET 8 та бібліотеки OpenTK, що включає реалізацію комплексу обчислювальних рушіїв, зокрема методів прямого розрахунку сил у послідовному, багатопотоковому та SIMD-оптимізованому режимах на CPU, високопродуктивного рушія на GPU з використанням Compute Shaders та Shared

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

Методу, а також ієрархічних алгоритмів Барнса-Хата з паралельною побудовою Остгее-дерева для центрального процесора та на основі LBVN-структур для графічного адаптера;

г) побудувати блок-схеми основних алгоритмів симуляції, зокрема етапів побудови дерева ієрархії, розрахунку сил та інтегрування положень тіл за методом Верле;

д) організувати інтерактивний графічний інтерфейс користувача засобами ImGui.NET для налаштування параметрів симуляції (гравітаційна стала, крок часу, коефіцієнт пом'якшення) та візуалізації метрик продуктивності в реальному часі;

е) реалізувати адаптивну систему візуалізації частинок, що дозволяє відображати динамічні характеристики системи (швидкість, прискорення) та забезпечує стабільну частоту оновлення кадрів при великій кількості об'єктів;

ж) провести експериментальні дослідження та розрахунки ефективності розроблених методів на наборах даних від 10^3 до 10^6 тіл, сформувати порівняльні графіки продуктивності CPU та GPU рушіїв;

з) провести валідацію результатів моделювання, порівнявши точність обчислень оптимізованих рушіїв з еталонним послідовним алгоритмом;

і) сформувати висновки про виконаний обсяг робіт, підтвердити ефективність обраних методів оптимізації та одержані науково-практичні результати.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

В основі функціонування розробленого комплексу лежить фундаментальна проблема небесної механіки – розв'язання задачі N-тіл. Вона полягає у визначенні траєкторій руху системи матеріальних частинок, які взаємодіють між собою згідно із законом всесвітнього тяжіння Ньютона. Сила гравітаційного тяжіння F_{ij} , що діє на i -ту частинку з боку j -тої частинки, обчислюється за формулою (3.1):

$$F_{ij} = G \frac{m_i m_j}{|r_j - r_i|^3} (r_j - r_i), \quad (3.1)$$

де G – гравітаційна стала;

m_j та m_i – маси тіл;

r_j та r_i – їхні просторові вектори-радіуси (координати).

Результуюча сила F_i , що діє на кожну окрему частинку, є векторною сумою сил з боку всіх інших $N - 1$ частинок у системі. Базовий підхід прямого інтегрування вимагає попарного обчислення цих взаємодій для кожної частинки з усіма іншими. Це призводить до квадратичної алгоритмічної складності $O(N^2)$, що робить моделювання мільйонів тіл практично неможливим у реальному часі. Для подолання цього бар'єру використовується ієрархічний алгоритм Барнса–Хата. Замість того, щоб вираховувати силу тяжіння між кожною окремою частинкою, алгоритм об'єднує далеку групу об'єктів у один великий кластер. Якщо група зірок знаходиться досить далеко, програма сприймає їх як одну спільну масу в одній точці, що значно спрощує розрахунки. Таке спрощення дозволяє системі працювати набагато швидше, оскільки кількість необхідних

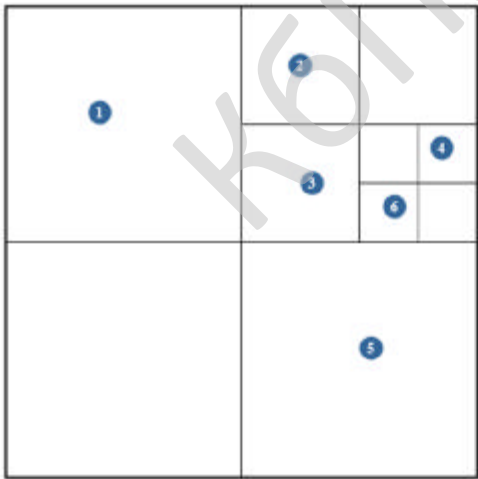
					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

операцій зростає майже пропорційно кількості частинок, а не у квадраті від їх числа.

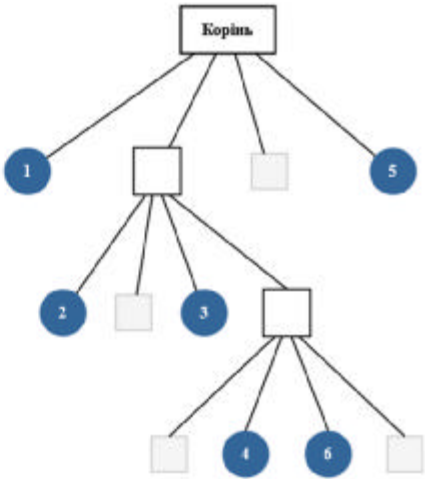
Після визначення результуючої сили для кожної частинки система переходить до етапу кінематичного інтегрування. Відповідно до другого закону Ньютона, прискорення об'єкта прямо залежить від сумарної сили та його маси. Оскільки сили в системі N-тіл безперервно змінюються разом із координатами, знайти точні траєкторії аналітичним шляхом неможливо. Тому алгоритм використовує методи чисельного інтегрування з дискретним часовим кроком Δt . На кожній ітерації оновлюється вектор швидкості на основі поточного прискорення, після чого розраховуються нові координати частинки. Такий покроковий підхід дозволяє з високою точністю симулювати еволюцію багатокomпонентних гравітаційних систем у часі.

Теоретична модель алгоритму Барнса–Хата, спирається на рекурсивний просторовий поділ середовища. Як показано на рисунку 3.1, простір ітеративно ділиться на рівні сектори (чотири квадранти для 2D або вісім октантів для 3D), утворюючи спеціалізовану деревоподібну структуру (Quadtree або Octree).

а) Просторовий поділ (Quadtree)



б) Деревоподібна структура даних



Внутрішній макровузол (Містить центр мас гілки)

Кінцевий листок (Містить реальну частинку)

Пустий вузол

Рисунок 3.1 – Схема просторового поділу (Quadtree)

Цей процес динамічно адаптується до густини розташування об'єктів: області зі щільним скупченням частинок діляться на глибші рівні, тоді як порожні зони залишаються неподіленими. У логічній структурі такого дерева кожен внутрішній вузол акумулює сумарну масу всіх своїх нащадків та зберігає координати їхнього центру мас. Під час розрахунку гравітації достатньо віддалені групи частинок розглядаються як одна «супер-частинка». Це дозволяє замінити тисячі дрібних обчислень одним груповим, що радикально підвищує продуктивність системи без критичної втрати точності моделювання.

Для ефективного перенесення цієї складної ієрархічної структури на масивно-паралельну апаратну архітектуру (GPU), система використовує просторову криву Z-порядку – код Мортон. Цей метод дозволяє без втрат упакувати багатовимірні просторові координати об'єктів в одновимірний цілочисельний масив. Сортування отриманого масиву гарантує високу просторову локальність: об'єкти, що знаходяться фізично поруч у симульованому просторі, автоматично розміщуються у суміжних комірках пам'яті. Це є критично важливим архітектурним рішенням, яке забезпечує високу когерентність кешу та мінімізує затримки звернення до VRAM під час виконання розрахунків на потокових мультипроцесорах.

Кожен крок симуляції виконується як набір простих кроків, що йдуть один за одним. На початку кожної ітерації попереднє просторове дерево знищується, і формується нова ієрархія шляхом послідовного додавання активних частинок. Це викликає рекурсивний поділ простору кожного разу, коли кількість об'єктів у вузлі перевищує встановлений ліміт місткості. Така повна перебудова дерева на кожному кроці гарантує структурну цілісність моделі в умовах високої динаміки частинок та ефективно реалізується завдяки паралельним алгоритмам сортування. Після формування топології система виконує висхідний обхід дерева, під час якого обчислюються агреговані фізичні показники: сумарна маса кожного вузла та координати його центру мас. На основі цих даних ініціюється етап розрахунку гравітаційних сил. Для кожної

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

окремої частинки виконується низхідний обхід дерева, у процесі якого на кожному рівні ієрархії перевіряється критерій МАС, що регулюється параметром Θ . Якщо відношення геометричного розміру вузла до відстані між центром мас вузла та поточною частинкою є меншим за встановлений поріг Θ , алгоритм припиняє заглиблення і розглядає весь кластер частинок у цьому вузлі як єдине масивне тіло. У випадку невиконання критерію алгоритм рекурсивно заглиблюється на наступний, більш деталізований рівень ієрархії.

Для перетворення математичних формул на працюючу програму створено швидку систему зв'язку між обчислювальним ядром та модулем візуалізації. Вона забезпечує миттєву передачу даних, щоб розрахунки та картинка на екрані працювали синхронно. В основі системи лежить гнучкий механізм керування, який об'єднує підготовку середовища, фізичні симуляції та малювання кадрів у єдиний процес.

Такий підхід дозволяє програмі автоматично підлаштовуватися під обладнання користувача. При роботі на процесорі система фокусується на правильному впорядкуванні даних у пам'яті для прискорення доступу, а при використанні відеокарти – на масових паралельних розрахунках безпосередньо у VRAM. Це дозволяє уникнути зайвих пересилань даних між частинами комп'ютера та виводити телеметрію системи в реальному часі. У результаті система функціонує як єдиний механізм, де теоретичні переваги алгоритму Барнса-Хата поєднуються з високою пропускнуою здатністю сучасних апаратних архітектур.

3.2 Розробка структурної схеми

Програмне забезпечення апаратно-програмного комплексу спроектовано на основі багат шарової модульної архітектури. Фундаментом архітектурного рішення виступає принцип інверсії залежностей (Dependency Inversion Principle).

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

Відповідно до цього підходу, високорівнева бізнес-логіка предметної області ізольована від низькорівневих деталей обчислень та графічного API.

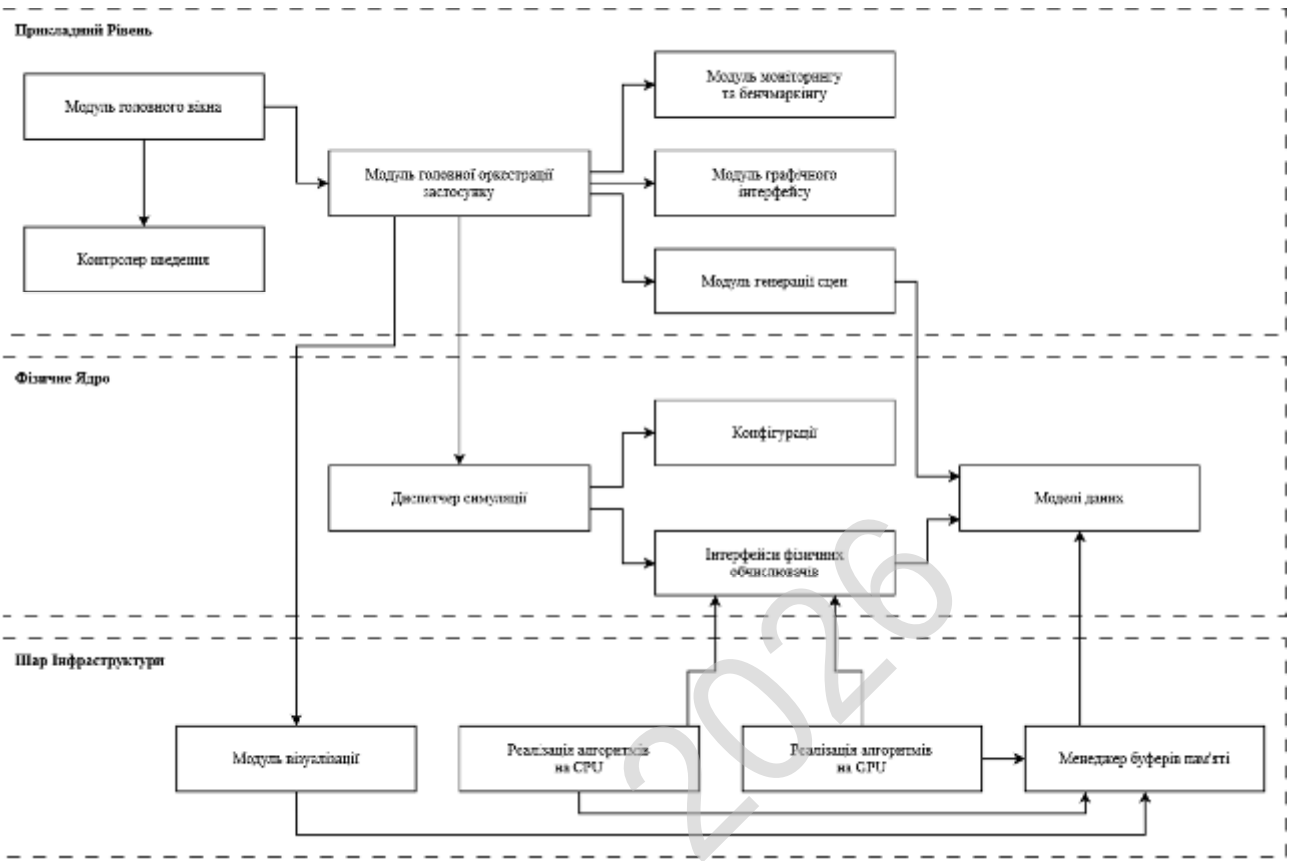


Рисунок 3.1 – Структурна схема системи

Як показано на рисунку 3.1, архітектура поділена на три логічні рівні. Ядро є центральним компонентом системи, який не має жодних вихідних залежностей на інші шари. Воно інкапсулює фундаментальні абстракції задачі N-тіл: моделі даних, конфігурації та диспетчер симуляції. Диспетчер керує життєвим циклом фізичного кроку, не взаємодіючи з конкретним апаратним забезпеченням безпосередньо. Замість цього він покладається на абстрактні контракти – інтерфейси фізичних обчислювачів.

Зовнішні модулі Шару Інфраструктури реалізують ці контракти. Реалізація алгоритмів на CPU використовує центральний процесор та SIMD-векторизацію. Альтернативна реалізація на GPU переносить обчислювальну складність (побудову LBVN, розрахунок сил) безпосередньо на потокові мультипроцесори відеокарти через обчислювальні шейдери. Усі ці модулі

залежать від Ядра, реалізуючи його інтерфейси, що зберігає незалежність базової моделі.

Прикладний рівень здійснює оркестрацію всього застосунку. Модуль головної оркестрації зв'язує Ядро з периферійними підсистемами: генерацією сцен, графічним інтерфейсом та телеметрією. Точкою входу всієї системи залишається модуль головного вікна, який інкапсулює роботу з ОС.

Важливою особливістю розробленої архітектури є застосування виправданого архітектурного рішення для забезпечення високої продуктивності: модуль візуалізації (Шар Інфраструктури) має пряму залежність від менеджера буферів пам'яті (у тому самому шарі). Замість ресурсоємної трансляції масивів з мільйонами об'єктів через абстрактні шари Ядра, модуль візуалізації безпосередньо зчитує підготовлені дані з VRAM, використовуючи графічний конвеєр OpenGL. Таке рішення створює оптимізований шлях читання даних, зберігаючи при цьому сувору ізоляцію фізичних розрахунків від підсистем візуалізації. Запропонована структура гарантує високий рівень масштабованості: додавання нових математичних солверів потребуватиме лише реалізації відповідного інтерфейсу без жодних модифікацій у Ядрі системи.

3.3 Розробка функціональної схеми

Для детального аналізу роботи апаратно-програмного комплексу розроблено функціональну схему, наведену на рисунку 3.2. Схема демонструє послідовний алгоритмічний конвеєр обробки одного розрахункового кадру, чітко розмежовуючи архітектурні рівні системи згідно з принципом інверсії залежностей.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

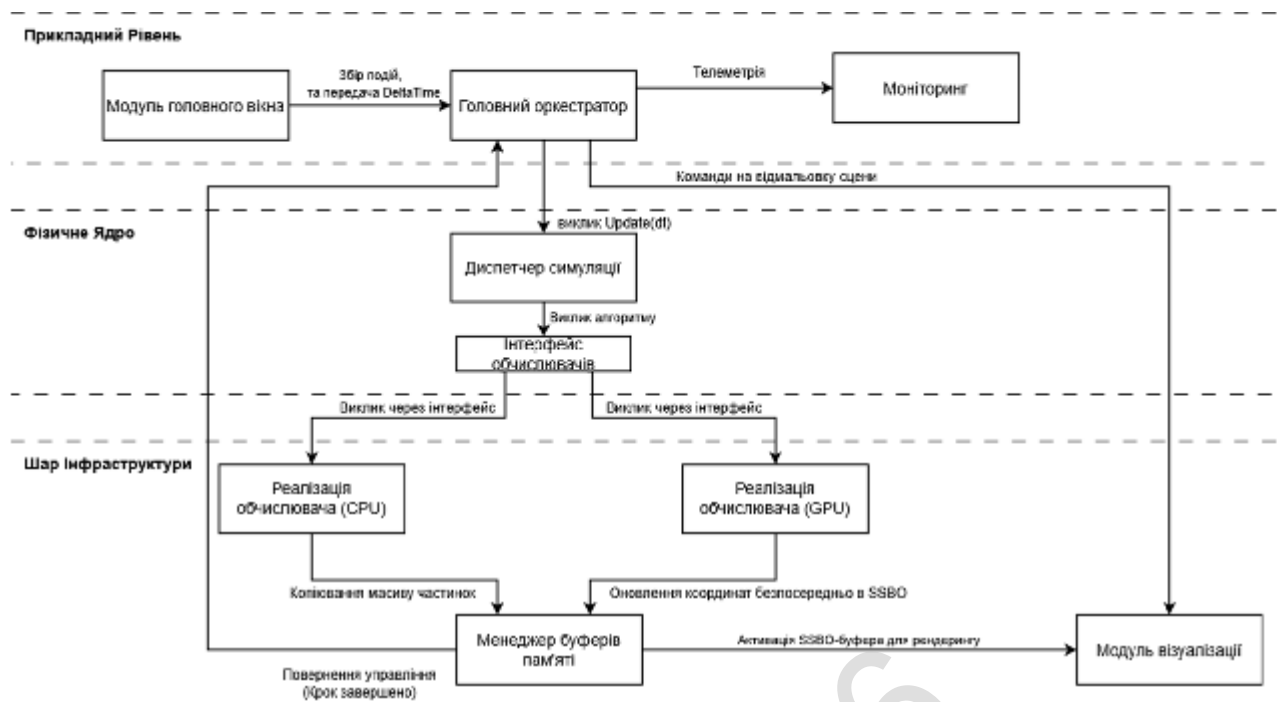


Рисунок 3.2 – Функціональна схема системи

Життєвий цикл кадру ініціюється на Прикладному рівні у модулі головного вікна, який збирає події операційної системи (ввід користувача) та обчислює час, що минув з попереднього кадру. Ці дані у вигляді змінної DeltaTime передаються до головного оркестратора застосунку. Оркестратор виступає центральним контролером життєвого циклу: він запускає фізичний крок, викликаючи відповідний метод у диспетчера симуляції, що розташований у Фізичному Ядрі.

Важливою особливістю конвеєра є абстрагування Ядра від конкретних обчислювальних алгоритмів. Диспетчер симуляції не виконує математичних операцій самостійно, а делегує їх через абстрактний інтерфейс обчислювачів. Залежно від конфігурації, поліморфний виклик переходить однією з реалізацій у Шарі Інфраструктури.

Якщо активна реалізація на CPU, алгоритм виконує обчислення (із залученням SIMD-інструкцій або просторових дерев, в залежності від обраного алгоритму), після чого менеджер буферів пам'яті здійснює ресурсоємну синхронізацію масиву частинок з RAM у VRAM. У випадку використання

реалізації на GPU, масивно-паралельний конвеєр оновлює координати частинок безпосередньо у буферах SSBO, уникаючи передачі масивів через шину PCI-E.

Після завершення апаратних розрахунків та синхронізації буферів, менеджер пам'яті повертає управління головному оркестратору. На фінальному етапі оркестратор паралельно ініціює два процеси: передає час виконання кадру до модуля моніторингу для фіксації телеметрії та відправляє команди модулю візуалізації. Модуль візуалізації зчитує підготовлені дані безпосередньо з відеопам'яті через графічний конвеєр OpenGL, формує кадр і передає його модулю головного вікна для остаточного виводу на екран монітора. Цей процес безперервно повторюється, забезпечуючи інтерактивність та високу кадрову частоту симуляції.

3.4 Розробка діаграми процесів

Для візуалізації алгоритмів взаємодії між системою та користувачем розроблено діаграму процесів, зображену на рисунку 3.3. Згідно з методологією проектування, граф відображає фундаментальні процеси (дії) системи, переходи між якими відбуваються по завершенню їх виконання або за ініціативи користувача.

Життєвий цикл програми розпочинається з обов'язкового процесу ініціалізації компонентів системи. На цьому етапі відбувається виділення пам'яті (зокрема SSBO буферів на відеокарті), компіляція обчислювальних шейдерів та первинна генерація сцени згідно з параметрами за замовчуванням. Після успішного завершення налаштування система переходить у свій центральний базовий процес – «Очікування дій користувача».

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

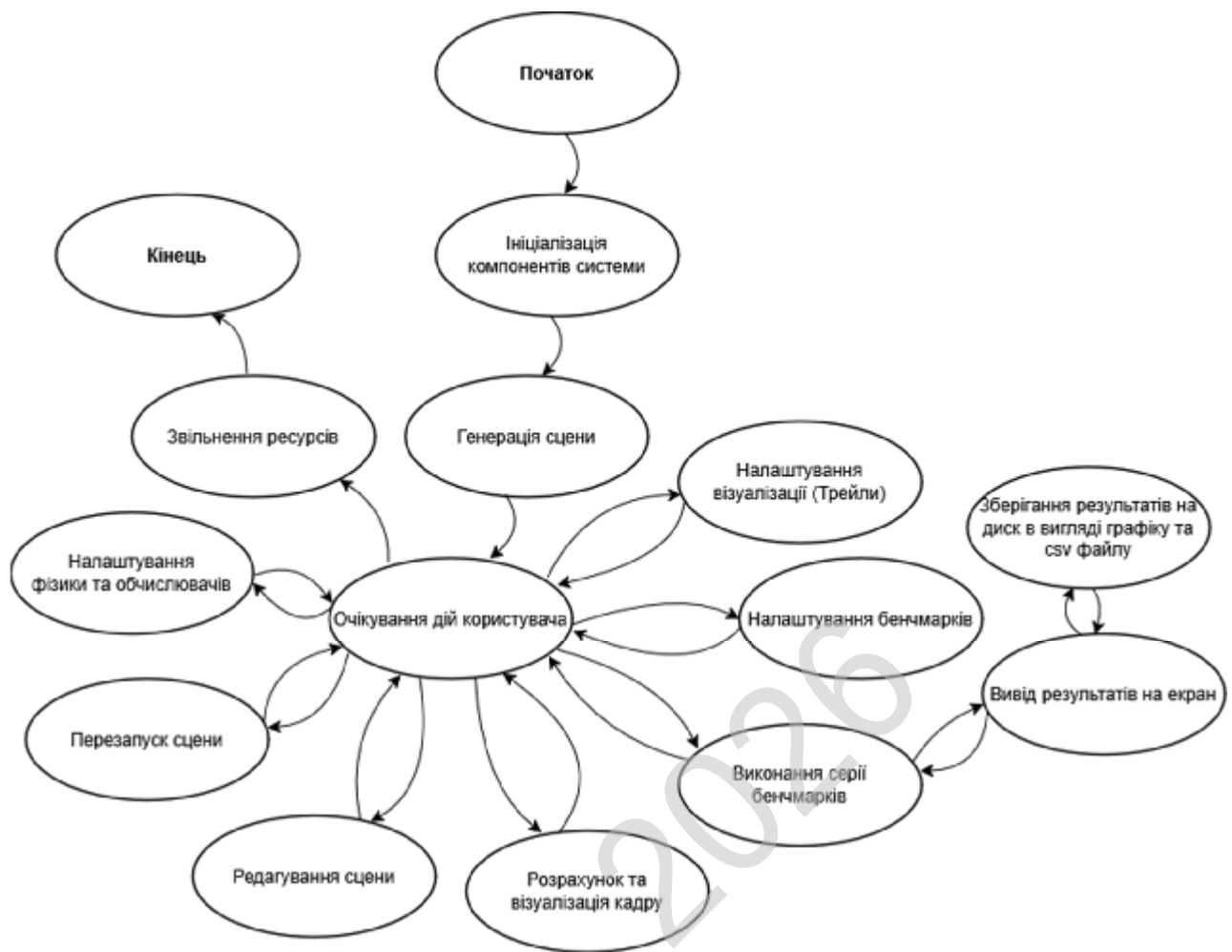


Рисунок 3.3 – Діаграма взаємодії процесів

З цього стану можливі декілька векторів взаємодії. Вибір користувачем відповідних елементів керування у графічному інтерфейсі ініціює переходи до наступних спеціалізованих процесів конфігурації:

- Редагування сцени: процес вибору типу макрооб'єктів (галактик, кластерів) та їх редагування. Критичною архітектурною особливістю є те, що вхід у цей процес з базового стану можливий виключно на нульовому кроці симуляції. Це обмеження зумовлене тим, що об'єкти це лише абстракція для задання початкового стану симуляції.

- Налаштування фізики та обчислювачів: перехід у процес зміни глобальних фізичних сталих та вибору архітектури виконання (CPU або GPU обчислювачі). Цей процес може бути ініційований у будь-який момент часу.

– Налаштування візуалізації: процес модифікації параметрів рендерингу, який дозволяє вмикати або вимикати відображення ієрархічної сітки дерева, слідів частинок, змінювати яскравість частинок, їх градієнт та масштаб.

Ключовим переходом у системі є запуск фізичного розрахунку. Натискання кнопки початку симуляції переводить граф у процес «Розрахунок та візуалізація кадру». Перебуваючи у цьому стані, система безперервно і циклічно обробляє фізику N-тіл та відмальовує результати (інкрементуючи лічильник кроків), доки не отримає тригер «Пауза», що миттєво повертає її до процесу очікування команд.

Окремою гілкою функціонування системи є модуль тестування продуктивності. З базового стану користувач може перейти до процесу «Налаштування бенчмарків», після чого ініціювати «Виконання серії бенчмарків». Цей процес є ізольованим: він призупиняє стандартний рендеринг частинок і виконує задані тести на обраних обчислювачах. По завершенню тестів система автоматично здійснює переходи до процесів виводу результатів на екран та збереження їх на диск (у вигляді графіків та CSV-файлів).

У випадку необхідності скидання симуляції передбачено перехід до процесу «Перезапуск сцени», який обнуляє лічильник кроків, очищує буфери і повертає систему у стан, де знову стає можливим структурне редагування об'єктів. Завершення роботи програми супроводжується переходом у термінальний стан «Кінець», де відбувається безпечне звільнення системних та апаратних ресурсів.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Робота системи базується на чіткій ієрархії процесів, що поєднують підготовку середовища, фізичне моделювання та виведення результатів.

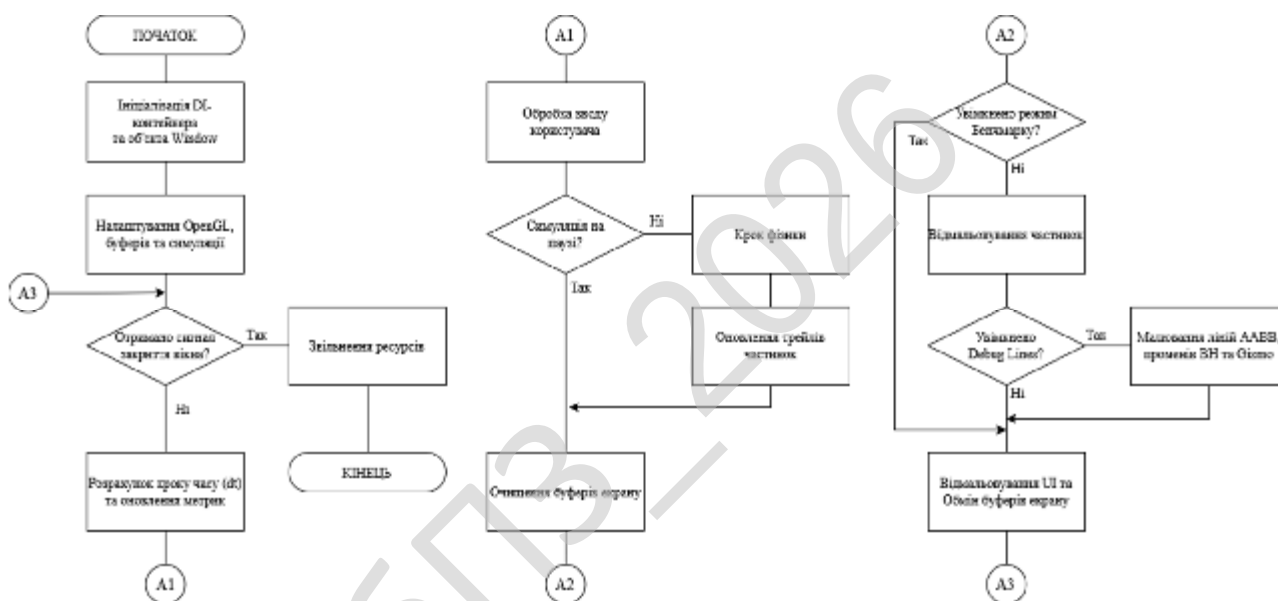


Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.1 наведено блок-схему основного життєвого циклу програми. Функціонування системи розпочинається з етапу ініціалізації, під час якого створюється екземпляр ядра фізичної симуляції, виділяється необхідна пам'ять для базових структур даних, а також налаштовується контекст віконного середовища та графічного інтерфейсу на базі бібліотеки ImGui. Після успішного завершення підготовчих процедур програма переходить до головного циклу обробки, який виконується безперервно до моменту отримання системного сигналу на закриття вікна додатку.

Кожна ітерація циклу розпочинається з процедури очищення графічних буферів, що готує екран для побудови нового кадру. Наступним кроком відбувається зчитування та обробка користувацького вводу: система фіксує переміщення миші, натискання клавіш і взаємодію з елементами інтерфейсу. Це дозволяє користувачеві керувати віртуальною камерою та системними параметрами у реальному часі.

Після оновлення стану вводу перевіряється логічна умова активності симуляції. Якщо вона не поставлена на паузу, ініціюється крок розрахунку фізики з передачею поточної дельти часу. На цьому етапі управління передається обраному обчислювачу (CPU або GPU), який перераховує гравітаційні сили та інтегрує нові координати об'єктів. У випадку активного режиму паузи обчислювальний етап пропускається, і система зберігає поточний просторовий стан сцени без змін.

Далі управління передається підсистемі графічного виводу. Процес візуалізації розділений на два послідовні кроки: спочатку виконується рендеринг фізичних об'єктів за допомогою графічних шейдерів на основі актуальних даних із відеопам'яті, після чого поверх тривимірної сцени накладається двовимірний інтерфейс із панелями налаштувань та метриками продуктивності. Завершується ітерація операцією апаратного перемикавання буферів, яка виводить сформований кадр на монітор, після чого алгоритм зациклюється.

Практична реалізація ініціалізації фізичного кроку в головному циклі (без урахування графічного рендерингу) має наступний вигляд:

```
public void Update(float dt)
{
    if (CurrentSolver == null || Config.IsPaused) return;
    CurrentSolver.Step(dt);
}
```

Така архітектура гарантує строге відокремлення логіки розрахунку від візуалізації. Безпосередній процес інтегрування інкапсульовано у метод Step, що дозволяє динамічно підміняти реалізації алгоритмів за патерном Strategy без втручання у код головного циклу програми.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

Базовим етапом оптимізації обчислювальної складності задачі N-тіл є ієрархічне просторове розділення простору. У рамках підготовки до обчислень за методом Барнса-Хата в системі реалізовано референсну модель рекурсивної побудови просторового октодерева на CPU. Логіка роботи цього алгоритму полягає у послідовному вставленні фізичних частинок у структуру дерева з подальшим динамічним поділом простору (вузлів) у разі виникнення колізій, що гарантує наявність не більше однієї частинки у кожному кінцевому листку дерева.

Цей процес описує логіку вставлення однієї фізичної частинки у структуру дерева, починаючи з кореневого вузла. Алгоритм розпочинається з етапу просторової валідації: система перевіряє, чи знаходяться координати цільової частинки в межах геометричних границь поточного вузла. Якщо частинка виходить за межі простору, який охоплює вузол, гілка виконання алгоритму негайно завершується, відкидаючи некоректні операції.

З метою мінімізації навантаження на GC у платформі .NET, вузли дерева реалізовано з використанням парадигми Data-Oriented Design у вигляді масивів значущих типів (struct). Структура базового вузла має наступний вигляд:

```
public struct OctreeNode
{
    public Vector3 CenterOfMass;
    public float TotalMass;
    public Vector3 Center;
    public float Size;

    public int FirstChild;
    public int ParticleIdx;

    public bool IsLeaf => FirstChild == -1;
    public bool IsEmpty => TotalMass == 0;
}
```

Такий підхід гарантує лінійне розташування даних у пам'яті, що суттєво підвищує ефективність використання кешу центрального процесора та дозволяє

безперешкодно і швидко передавати ці масиви до відеопам'яті графічного адаптера.

Оскільки класична рекурсивна побудова дерева має природні обмеження щодо розпаралелювання, для досягнення максимальної продуктивності основні обчислення переносяться на графічний прискорювач (GPU). Замість класичного вставлення об'єктів у дерево, на GPU реалізовано паралельний метод з побудовою лінійної ієрархії обмежувальних об'ємів (LBVH).

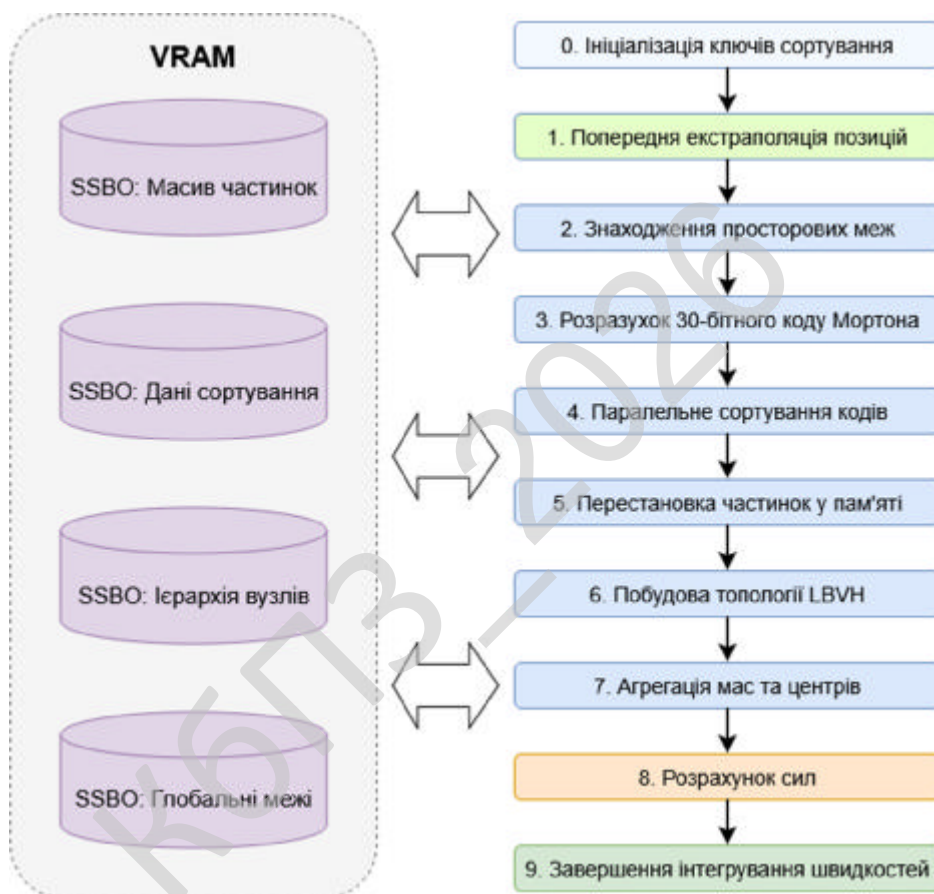


Рисунок 4.2 – Діаграма етапів обчислювального конвеєра LBVH

На рисунку 4.2 зображено діаграму етапів обчислювального GPU-конвеєра. Робота конвеєра в межах одного фізичного кроку симуляції починається з процедури прив'язки масивів даних до контексту OpenGL через механізм SSBO. Це гарантує всім наступним обчислювальним шейдерам глобальний доступ до відеопам'яті без витрат на копіювання даних.

Цикл обчислень ініціюється шейдером скидання (0), який готує масиви ключів для майбутнього сортування. Далі виконується шейдер попередньої екстраполяції (1), який згідно з математичною моделлю інтегрування Velocity Verlet виконує перший напівкрок оновлення швидкостей та зсув позицій частинок. Після цього запускається шейдер просторових меж (2), що використовує швидко паралельну редукцію для пошуку абсолютних мінімальних та максимальних координат усього скупчення об'єктів.

Наступна група шейдерів відповідає за просторове індексування. Система обчислює локальні координати частинок та генерує для них просторові індекси (3), після чого масив ключів проходить етап паралельного сортування (4). Відсортовані дані слугують фундаментом для побудови топології дерева LBVH: послідовно виконуються етапи реорганізації даних (5), встановлення ієрархічних зв'язків між вузлами (6) та висхідного обходу дерева для підрахунку сумарних мас і центрів мас кожного вузла (7).

Кульмінацією конвеєра є виконання алгоритму Барнса-Хата (8), де кожен потік здійснює обхід побудованого дерева для розрахунку результуючих гравітаційних сил. Завершується цикл шейдером фіналізації (9), який виконує другий напівкрок алгоритму Velocity Verlet, остаточно оновлюючи швидкості частинок перед наступним кадром симуляції.

Ключовим викликом при реалізації цього конвеєра на GPU є перетворення тривимірної просторової інформації у лінійну структуру, яка б оптимально оброблялася тисячами потоків. Це завдання вирішується на етапі просторового кодування. На основі глобальних меж, знайдених на попередньому кроці, алгоритм квантує координати кожної частинки та генерує для неї унікальний 30-бітний просторовий код Мортон, ефективно проєктуючи тривимірний простір на одновимірну числову вісь.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

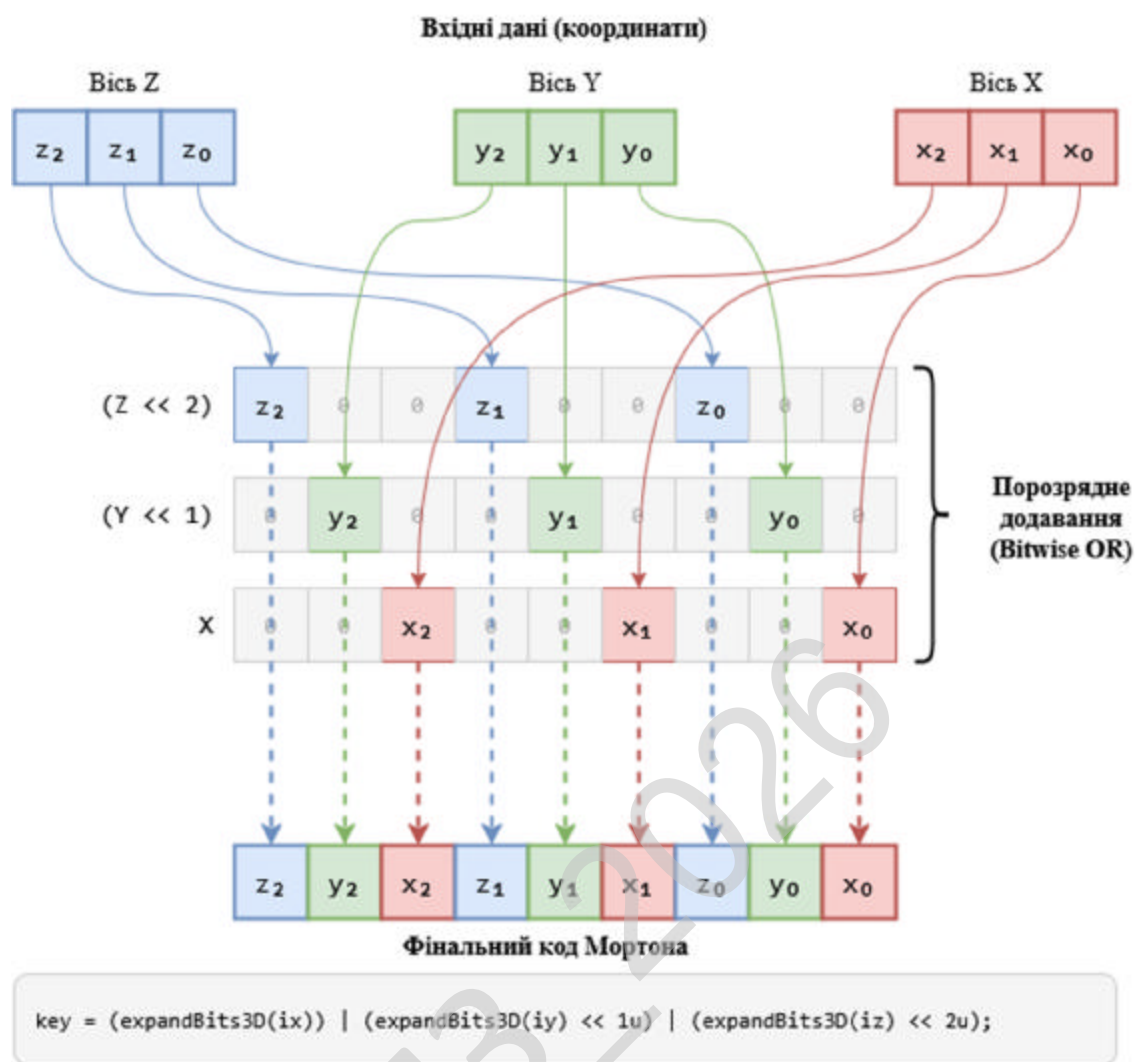


Рисунок 4.3 – Генерація коду Мортон “Гребінцем” (Bit Interleaving)

На рисунку 4.3 зображено процес генерації такого коду (метод "гребінця"), який дозволяє ефективно відобразити багатовимірні координати частинок в одновимірний простір. Основна перевага цього підходу полягає у збереженні просторової локальності: після сортування отриманих значень об'єкти, що знаходяться поруч у фізичному просторі, з високою ймовірністю виявляться поруч і в лінійному масиві пам'яті. У контексті апаратного прискорення це забезпечує критично важливих ефекти:

- Когерентність кешу: під час розрахунку взаємодій графічний процесор звертається до сусідніх у пам'яті даних, що мінімізує кількість затримок та підвищує пропускну здатність шини.

– Побудова LBVH: впорядкований за Мортонем масив дозволяє згенерувати топологію дерева LBVH паралельно, використовуючи лише побітові операції, що є значно швидшим за класичну рекурсивну побудову Oстree.

Для просторового групування розрахованих даних система застосовує багатопрохідну GPU-реалізацію алгоритму порозрядного сортування (Radix Sort). Оскільки ефективне сортування на відеокарті вимагає повної відсутності умовних розгалужень, алгоритм реалізовано за принципом Least Significant Digit (LSD) з основою 256. Сортування 32-бітних кодів Мортонa виконується за чотири ітерації (по 8 біт на прохід) з використанням двох взаємозамінних буферів пам'яті.

Для забезпечення паралелізму кожна ітерація розбита на три жорстко синхронізовані фази, які виконуються окремими обчислювальними шейдерами:

- Підрахунок (Count): паралельне формування локальних гістограм за допомогою швидкої спільної пам'яті (shared memory) груп потоків.
- Префіксна сума (Scan): обчислення глобальних зміщень пам'яті на основі зібраних гістограм.
- Розподіл (Scatter): детермінований запис елементів на їхні нові позиції у вихідному буфері.

Оскільки сортується лише масив числових ключів (з метою економії пропускної здатності шини), після його завершення викликається спеціальний шейдер перестановки. Він фізично впорядковує масив великих структур частинок у пам'яті відповідно до відсортованих індексів, що є критично важливим для забезпечення когерентності кешу GPU під час подальшого обходу дерева.

Друга фаза графічного конвеєра розпочинається зі створення топології дерева. Для ефективної паралельної генерації ієрархії на GPU без використання рекурсії у системі було адаптовано алгоритм максимізації паралелізму, запропонований Т. Каррасом (Тего Karras) [1]. Згідно з цим підходом, кожен внутрішній вузол LBVH генерується незалежно окремим потоком виконання.

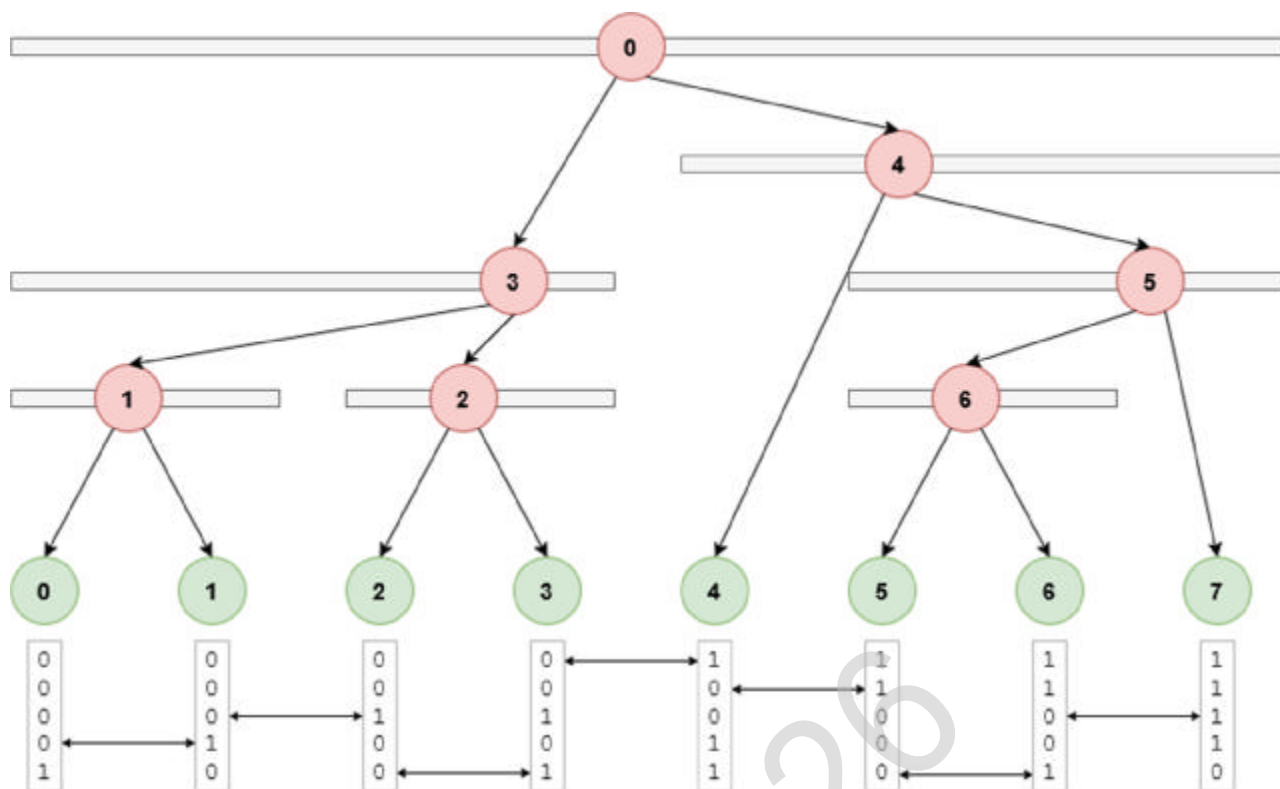


Рисунок 4.4 – Алгоритм Т. Карраса для побудови ієрархії вузлів [1].

Відповідний шейдер аналізує відсортований масив кодів Мортон та визначає діапазон дочірніх елементів для кожного вузла. Це досягається шляхом знаходження найдовшого спільного префікса (тобто порівнюються початкові біти) між кодами Мортон сусідніх об'єктів за допомогою алгоритму бінарного пошуку. Графічну інтерпретацію цього процесу наведено на рисунку 4.4. Як видно зі схеми, довжина спільного префікса визначає просторовий діапазон частинок, який охоплює конкретний внутрішній вузол, а місце зміни старшого біта вказує на точку розколу діапазону. Спираючись на розбіжності цих бітових префіксів, алгоритм паралельно, детерміновано та повністю без використання блокувань потоків формує всі ієрархічні зв'язки (вказівники на батьківські та ліві/праві дочірні вузли) у дереві.

Оскільки алгоритм Карраса будує топологію згори донизу, а подальший розрахунок центрів мас вимагає проходження знизу догори, архітектура потребує механізму гарантованого завершення дочірніх потоків. Для цього після формування зв'язків відбувається обов'язкове ініціалізуюче очищення атомарних

прапорців синхронізації у пам'яті вузлів. Після цього керування передається шейдеру агрегації мас, який використовує ці прапорці для безпечного підйому по дереву від листків до кореня.

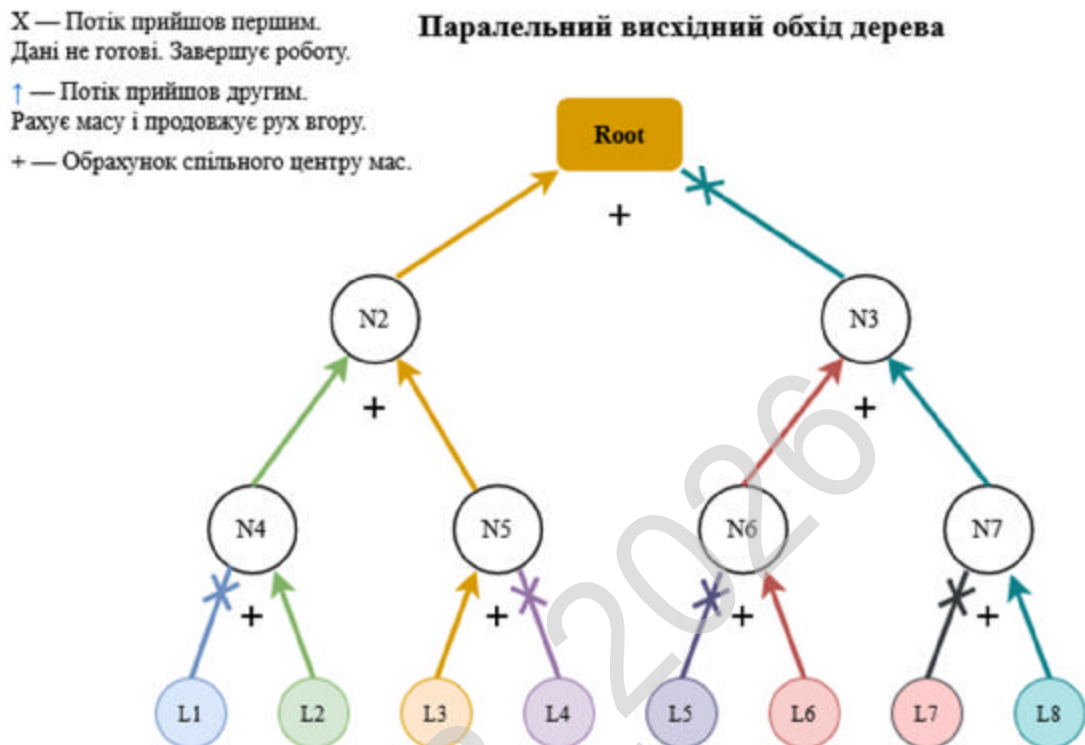


Рисунок 4.5 –Паралельний висхідний обхід дерева (8 листів – 8 потоків)

На рисунку 4.5 зображений обхід дерева знизу вгору – від кінцевих листків до кореня, послідовно сумуючи маси, обчислюючи координати локальних центрів мас та формуючи точні обмежувальні паралелепіпеди для кожної гілки.

Після повного формування просторової структури система переходить до найресурсоємнішого етапу – розрахунку гравітаційних впливів. На цьому кроці GPU виділяє окремий потік для кожної частинки, який здійснює незалежний обхід дерева, використовуючи критерій МАС для відсікання віддалених гілок та їхньої апроксимації. Завершується фізична ітерація викликом шейдера фіналізації, який інтегрує отримані вектори сил у систему, остаточно коригуючи

прискорення, швидкості та позиції частинок, після чого конвеєр зупиняється до початку наступного кадру.

Синхронізація між етапами цього конвеєра є критично важливою для уникнення стану гонитви (Race Condition) у VRAM. Практична реалізація виклику шейдерів обов'язково супроводжується встановленням бар'єрів пам'яті:

```
// Етап 5: Екстраполяція позицій
PredictShader.Dispatch(numGroups, 1, 1);
GL.MemoryBarrier(MemoryBarrierFlags.ShaderStorageBarrierBit);

// Етап 6: Побудова топології дерева LBVN
HierarchyShader.Dispatch(numGroups, 1, 1);
GL.MemoryBarrier(MemoryBarrierFlags.ShaderStorageBarrierBit);
```

Найбільш ресурсоємним етапом описаного конвеєра є розрахунок сил. На рисунку 4.6 наведено детальну блок-схему алгоритму розрахунку гравітаційних сил Барнса-Хата, який виконується незалежним потоком графічного процесора для кожної окремої частинки симуляції.

Виконання алгоритму розпочинається з ініціалізації швидкого локального стеку в пам'яті регістрів відеокарти, куди першим елементом поміщається індекс кореневого вузла просторового дерева LBVN. Далі ініціюється головний цикл обходу ієрархії, який триває до повного вичерпання елементів у стеку. На кожній ітерації циклу верхній вузол вилучається зі стеку та підлягає серії логічних і просторових перевірок. Спочатку алгоритм відкидає порожні вузли з нульовою масою та запобігає математичним сингулярностям шляхом перевірки на самовзаємодію (якщо дистанція до вузла-листка наближається до нуля).

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		40

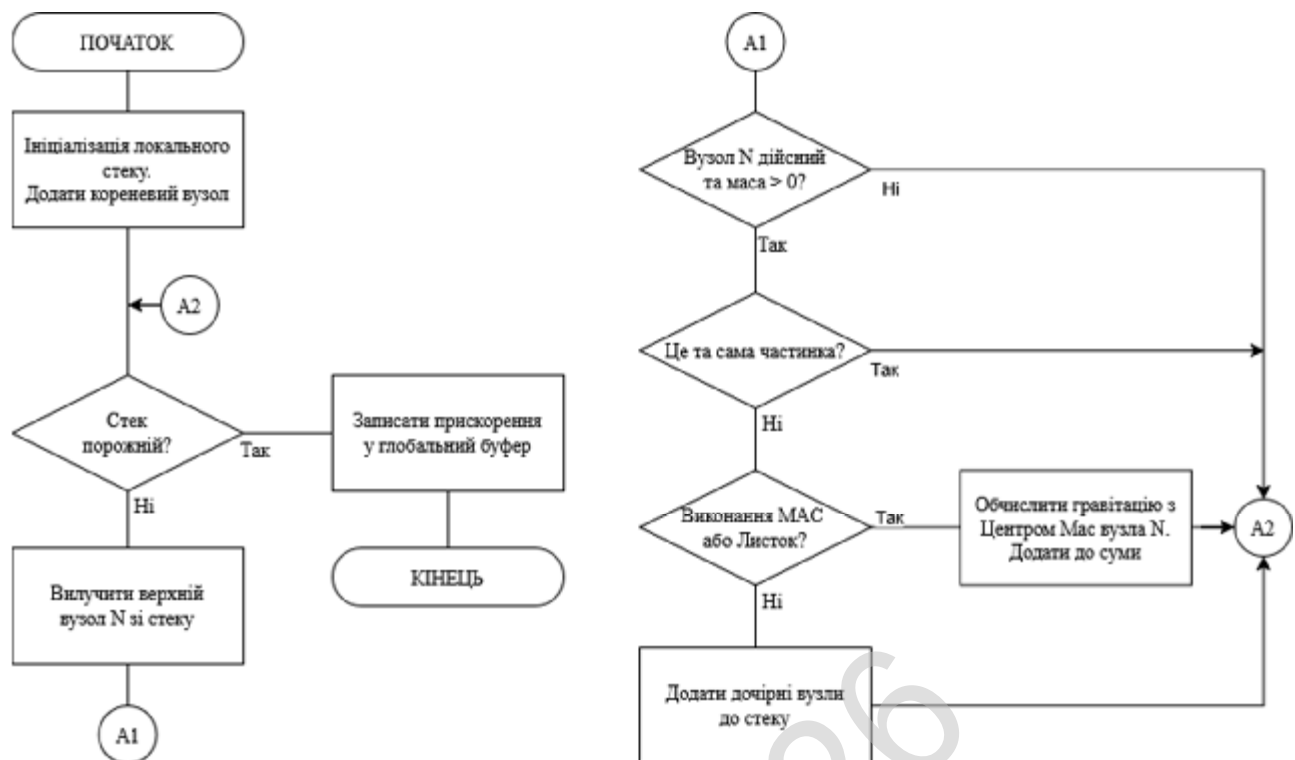


Рисунок 4.6 – Блок-схема підпрограми розрахунку сил Барнса-Хата

Наступним кроком застосовується МАС, який визначає, чи є поточний вузол достатньо віддаленим для просторової апроксимації. Залежно від конфігурації симуляції, відеокарта використовує один із двох режимів МАС: стандартний (що базується на відстані до центру мас вузла) або вдосконалений (що враховує найкоротшу відстань до меж обмежувального об'єму вузла). Вдосконалений режим автоматично вимагає розкриття вузла, якщо цільова частинка знаходиться геометрично всередині його меж, що надійно захищає симуляцію від екстремальних нефізичних аномалій під час проходження частинок крізь щільні кластери.

Якщо критерій МАС виконується або обраний вузол є кінцевим листком, алгоритм апроксимує його загальний вплив як гравітаційну взаємодію з єдиною точковою масою, розташованою у центрі мас вузла. Розрахована сила відразу додається до сумарного вектора прискорення. Якщо ж критерій МАС не виконується, апроксимація відкидається на користь просторової деталізації: дочірні гілки поточного вузла додаються до локального стеку для їх подальшого

рекурсивного аналізу. Після повного опрацювання дерева сумарний вектор сили записується у SSBO як нове прискорення частинки.

З огляду на архітектурні обмеження графічних процесорів, які не підтримують глибоку рекурсію, обхід дерева реалізовано ітеративним методом. Фрагмент коду обчислювального шейдера мовою GLSL, що відповідає за оцінку критерію MAC та обчислення сили взаємодії:

```
bool isFar = false;

if (MacMode == 0) {
    vec3 sizeVec = N[nodeIdx].Max.xyz - N[nodeIdx].Min.xyz;
    float size = max(sizeVec.x, max(sizeVec.y, sizeVec.z));
    isFar = (size * size) < (ThetaSq * distSq);
} else {
    // Вдосконалений MAC з перевіркою перебування всередині AABB
    vec3 minB = N[nodeIdx].Min.xyz;
    vec3 maxB = N[nodeIdx].Max.xyz;
    vec3 closest = clamp(pPos3D, minB, maxB);
    vec3 toBox = closest - pPos3D;
    float dBoxSq = dot(toBox, toBox);

    vec3 sV = maxB - minB;
    float s = max(sV.x, max(sV.y, sV.z));
    isFar = (dBoxSq > 0.0) && ((s * s) < (ThetaSq * dBoxSq));
}

if (isLeaf || isFar) {
    // Апроксимація сили з урахуванням фактора пом'якшення (Softening)
    float invDist = inversesqrt(distSq + SofteningSq);
    acc += r * (G * cm.w * invDist * invDist * invDist);
} else {
    // Деталізація: додавання дочірніх вузлів до локального стеку
    if (stackPtr < 30) {
        int left = N[nodeIdx].Left;
        int right = N[nodeIdx].Right;
        if (right >= 0) stack[stackPtr++] = right;
        if (left >= 0) stack[stackPtr++] = left;
    }
}
```

З метою максимальної апаратної оптимізації та уникнення використання ресурсоемних ALUs-інструкцій графічного процесора (добування квадратного кореня та операцій ділення), базове рівняння критерію MAC $\left(\frac{S}{d} < \Theta\right)$ було математично перетворено на оптимізоване порівняння квадратів значень.

Для стандартного режиму MAC перевірка виконується за формулою (4.1):

$$S^2 < \Theta^2 \times d_{cm}^2, \quad (4.1)$$

де S – найбільший лінійний розмір AABV поточного вузла;

Θ – критерій точності алгоритму Барнса-Хата;

d_{cm} – просторова відстань між цільовою частинкою та центром мас вузла;

Однак стандартний підхід має вразливість: якщо частинка опиняється глибоко всередині або на самій межі гігантського, але розрідженого вузла, відстань d_{cm} до його центру мас може бути формально великою, що призведе до помилкової апроксимації (аномальних прискорень) сусідніх тіл. Для розв'язання цієї проблеми розроблено вдосконалений режим Strict AABV MAC. Його математична модель базується на розрахунку відстані не до віртуального центру мас, а до найближчої фізичної грані обмежувального об'єму (AABV) вузла за формулою (4.2):

$$d_{box}^2 = \sum_{i \in \{x,y,z\}} (\max(\text{Min}_i, \min(pPos_i, \text{Max}_i)) - pPos_i)^2, \quad (4.2)$$

де d_{box} – найкоротша відстань від частинки до межі вузла;

$\text{Min}_i, \text{Max}_i$ – мінімальні та максимальні координати граней вузла за відповідною віссю;

$pPos_i$ – координати цільової частинки.

Базуючись на цій відстані, фінальна умова вдосконаленого критерію приймає вигляд (4.3):

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

$$S^2 < \Theta^2 \times d_{\text{box}}^2, \text{ за умови } d_{\text{box}}^2 > 0 \quad (4.3)$$

де додаткова умова $d_{\text{box}}^2 > 0$ діє як запобіжник: якщо частинка геометрично знаходиться всередині об'єму вузла ($d_{\text{box}} = 0$), будь-яка апроксимація миттєво відхиляється.

Якщо критерій МАС виконується, алгоритм апроксимує загальний вплив вузла. Безпосередньо для розрахунку вектора гравітаційної сили застосовується апаратна функція `inversesqrt`. Вона забезпечує обчислення оберненого квадратного кореня за один машинний такт, що дозволяє максимально ефективно виконати нормалізацію векторів та розрахувати прискорення за законом всесвітнього тяжіння Ньютона. Якщо ж критерій не виконується, вузол розкривається, а його нащадки додаються до локального стеку потоку для подальшого рекурсивного обходу.

Для підтвердження вірності прийнятих проектних та програмних рішень, а також для оцінки ефективності розроблених алгоритмів симуляції N-тіл, було проведено комплексне профілювання системи. Оскільки під час тестування на результати можуть впливати фонові процеси операційної системи та робота збирача сміття у середовищі .NET, використання середнього арифметичного значення для оцінки часу виконання кадру є некоректним і призводить до викривлення статистики. Тому в розробленій системі бенчмаркінгу застосовано методи робастної статистики: для кожного тестового сценарію формується масив часу виконання послідовних кадрів, з якого обчислюється медіана. Аналогічний підхід застосовано і для розрахунку відносної похибки обчислення сил, що дозволило відсікти статистичні аномалії.

Усі заміри продуктивності проводилися на єдиному апаратно-програмному стенді для забезпечення репрезентативності та відтворюваності результатів. Тестування виконувалося на базі портативної робочої станції Acer Nitro 5 з такою конфігурацією: центральний багатопотоковий процесор AMD

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

Ryzen 7 4800H, 16 ГБ оперативної пам'яті стандарту DDR4 (два модулі по 8 ГБ, що функціонують у двоканальному режимі для забезпечення максимальної пропускної здатності), твердотільний накопичувач Western Digital PC SN530 (NVMe SSD) та дискретний графічний прискорювач NVIDIA GeForce GTX 1650 Ti. Програмне середовище тестування: операційна система Windows 10 (64-bit), платформа виконання .NET 8.0 та графічний API OpenGL 4.5.

Під час тестування алгоритму Барнса-Хата було виявлено фундаментальну залежність між топологією просторового дерева та ефективністю MAC. Процесорна реалізація симуляції побудована на базі класичного Октодерева, вузли якого завжди мають форму правильних кубів. Для такої симетричної топології класичний критерій апроксимації є оптимальним. Натомість графічний прискорювач використовує LBVN, вузли якої формуються шляхом щільного облягання кластерів частинок. Використання стандартного критерію для витягнутих (асиметричних) вузлів LBVN призводить до деградації точності. Для вирішення цієї проблеми у GPU-шейдері було імплементовано "Суворий" критерій (Strict AABV MAC), який розраховує найкоротшу відстань від частинки до граней обмежувального об'єму. Профілювання показало (див. рисунок 4.7), що застосування Strict MAC гарантує високу точність: при значенні параметра $\Theta = 0.5$ медіанна похибка становить усього 0.0238%, при цьому час розрахунку залишається в межах високої продуктивності (~4.9 мс).

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

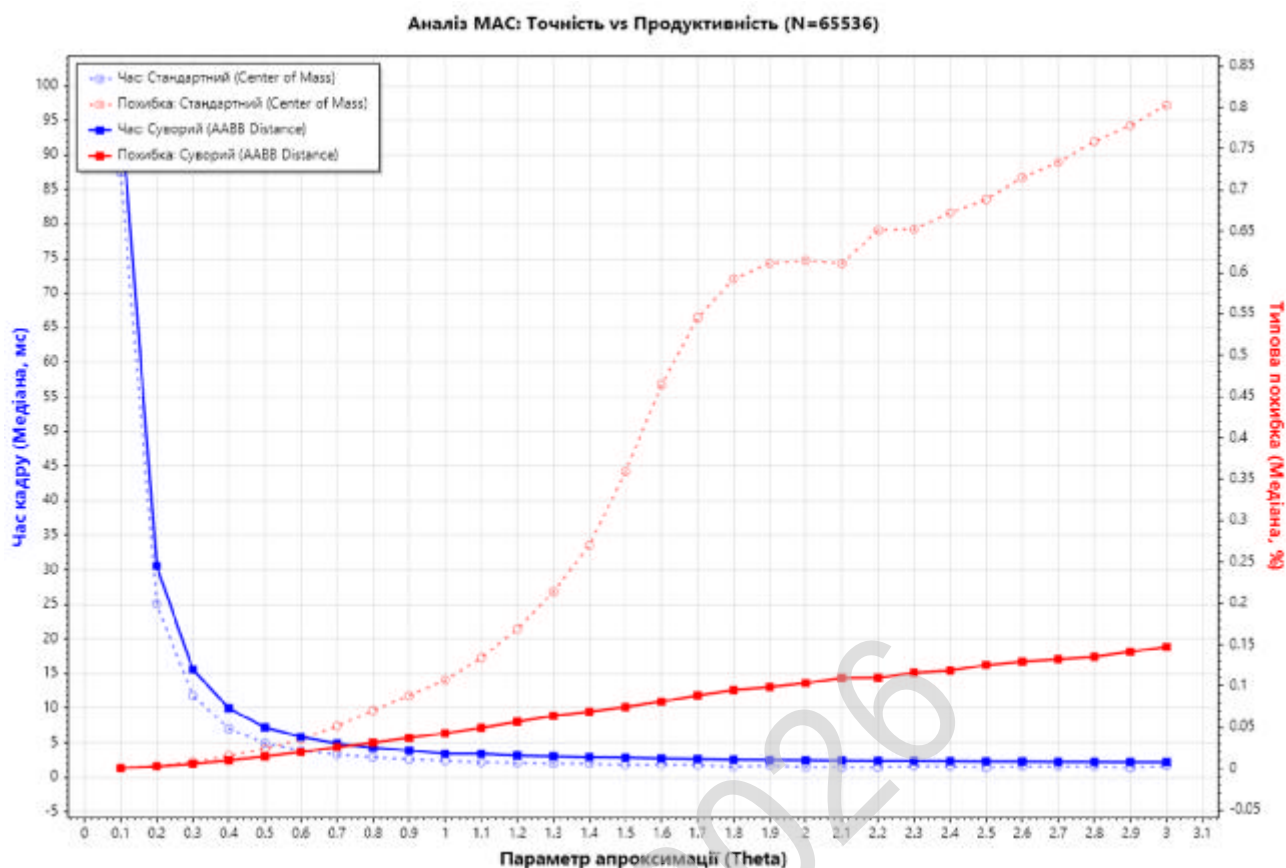


Рисунок 4.7 – Залежність часу та похибки від параметра θ для різних MAC

Для оцінки продуктивності було проведено порівняння прямого розрахунку (Brute Force) та алгоритму Барнса-Хата на вибірках до 262144 об'єктів. Через експоненційну різницю в алгоритмічній складності ($O(N^2)$ проти $O(N \log N)$), результати тестування представлено на графіку з логарифмічною шкалою часу (рис. 4.8). Як свідчать дані, на надмалих вибірках (близько 1000 частинок) центральний процесор демонструє результати, співставні з графічним прискорювачем: процесорна реалізація Барнса-Хата потребує 0.84 мс, тоді як GPU-варіанти виконуються за 0.55–1.05 мс залежно від обраного критерію MAC. Це пояснюється архітектурною пристосованістю CPU до рекурсивних структур та відсутністю накладних витрат на диспетчеризацію шейдерів.

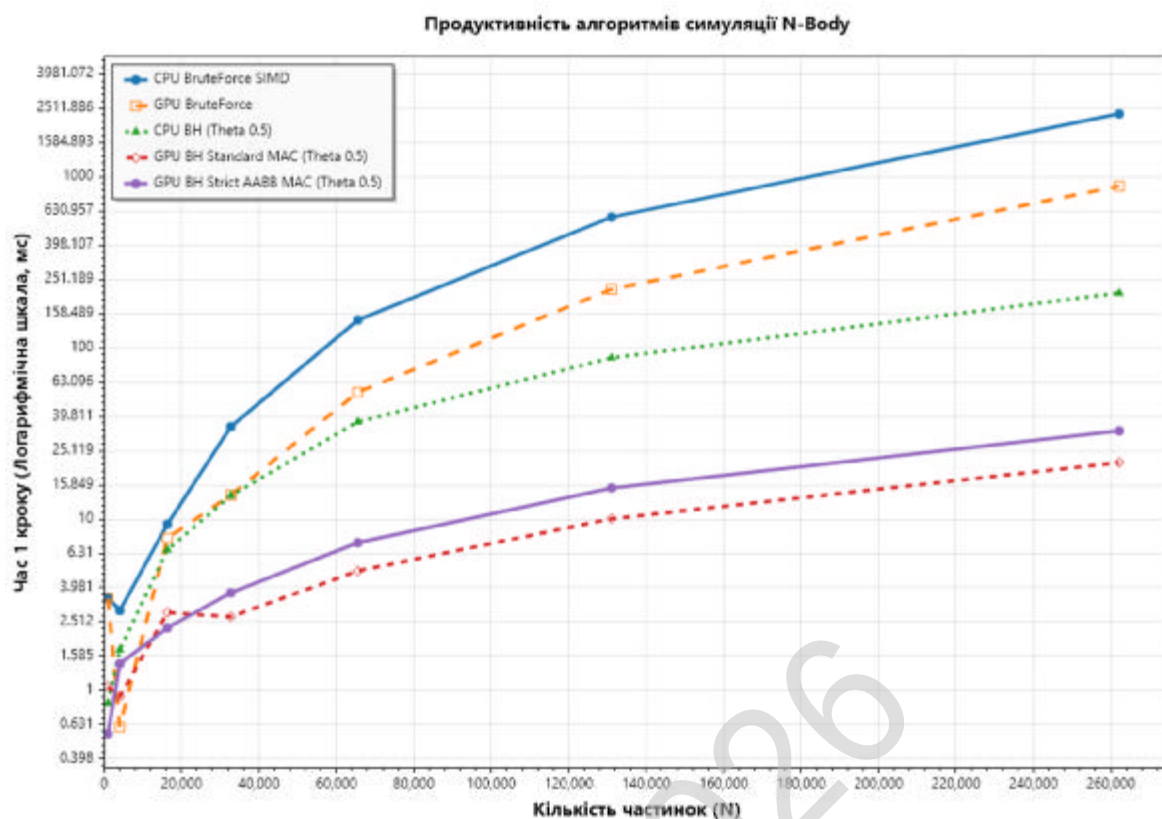


Рисунок 4.8 – Час виконання алгоритмів при різній кількості частинок

Однак, зі збільшенням кількості об'єктів масивний паралелізм GPU повністю нівелює ефект розходження потоків. При симуляції 262144 частинок CPU-реалізація алгоритму Барнса-Хата потребує 209.4 мс на кадр. Водночас GPU виконує аналогічний розрахунок всього за 21.4 мс зі стандартним критерієм апроксимації (Standard MAC) та за 32.8 мс із використанням високоточного «суворого» критерію (Strict AABV MAC). Це забезпечує від 6.4-кратного до майже 10-кратного прискорення порівняно з оптимізованою процесорною версією алгоритму Барнса-Хата.

Порівняно ж із базовим алгоритмом прямого розрахунку (Brute Force SIMD) на CPU, який потребує 2324.9 мс для обробки 262144 об'єктів, розроблений графічний конвеєр Барнса-Хата забезпечує колосальне прискорення від 71 до 108 разів (залежно від обраного режиму точності), дозволяючи симулювати величезні скупчення частинок у реальному часі.

Для визначення граничних можливостей розробленого апаратно-програмного комплексу було проведено тестування графічного конвеєра на надвеликих вибірках даних (рис. 4.9) – до понад 4 мільйонів об'єктів. Тестування виконувалося для двох критеріїв апроксимації: стандартного (Standard MAC) та суворого (Strict AABV MAC) при агресивному значенні параметра $\Theta = 1.0$.

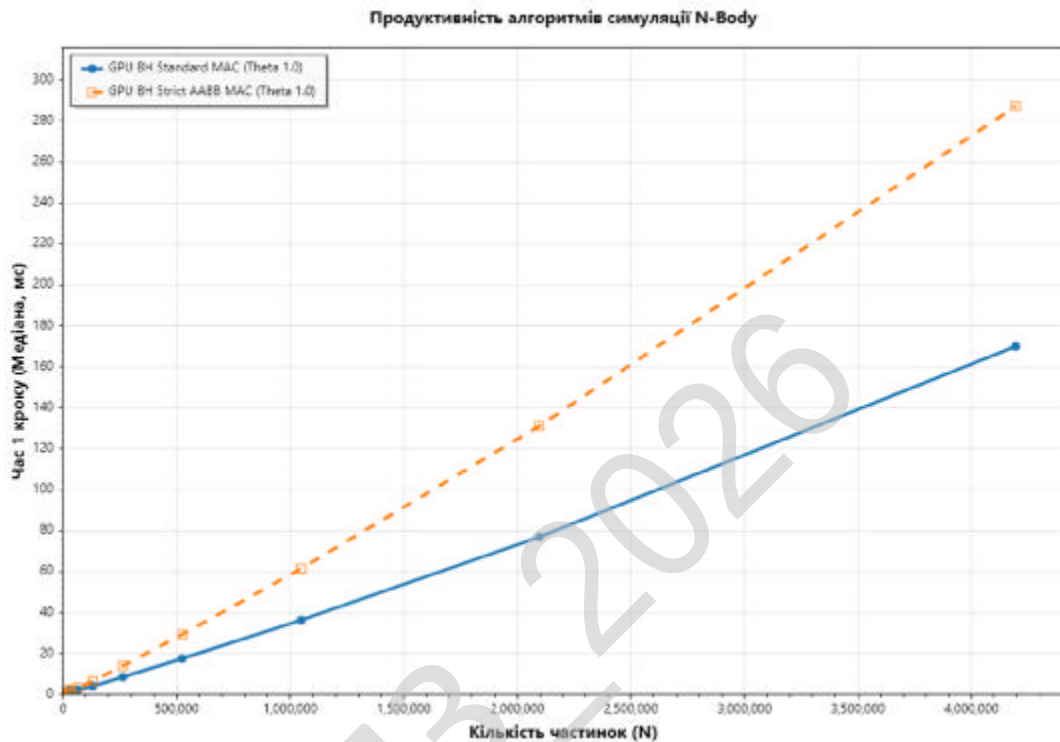


Рисунок 4.9 – Час виконання алгоритму Барнса-Хата при $\theta = 1.0$ (N до 2^{22})

Аналіз отриманих результатів демонструє відмінне масштабування алгоритму Барнса-Хата на відеокарті, що зберігає логіку $O(N \log N)$ на всьому діапазоні даних. При симуляції 1048576 об'єктів конвеєр зі стандартним критерієм виконує розрахунок кадру за 36.359 мс. Це означає, що розроблена система здатна симулювати мільйон тіл з інтерактивною частотою кадрів (близько 27 кадрів на секунду). Навіть при екстремальному навантаженні у 4194304 частинки розрахунок займає прийнятні 169.996 мс на кадр.

Застосування суворого критерію (Strict AABV MAC), який вираховує найкоротшу відстань до граней обмежувального об'єму для гарантування фізичної стабільності, очікувано генерує більше обчислювальне навантаження.

На мікро-вибірках (до 4096 частинок) різниця між алгоритмами нівелюється накладними витратами на диспетчеризацію шейдерів, проте вже з 16384 об'єктів Strict MAC починає стабільно поступатися у швидкості. Це відбувається тому, що суворий критерій більш консервативний: він частіше відкидає можливість апроксимації, змушуючи потік графічного процесора розкривати вузол і спускатися на глибші рівні дерева.

При досягненні 1048576 частинок час розрахунку для суворого критерію становить 61.541 мс. На максимальній тестовій вибірці у 4194304 об'єкти час виконання зростає до 287.273 мс, що майже в 1.7 раза повільніше за стандартний метод, але має суттєво більшу точність.

4.2 Захист розробленого програмного забезпечення

Під час розробки системи особливу увагу приділено питанню захисту інтелектуальної власності, оскільки алгоритми розрахунку N-тіл та методи керування відеопам'яттю є ключовою цінністю проєкту. Традиційно програми на .NET компілюються у проміжний код MSIL, який дуже легко декомпілювати назад у зрозумілий C#. Зазвичай для боротьби з цим використовують обфускатори, які заплутують код. Проте у випадку з фізичним двигуном, де кожна мілісекунда на рахунок, обфускація є неприйнятною. Будь-яке втручання в логіку роботи високонавантажених методів, особливо тих, що використовують векторні SIMD-інструкції, призводить до непередбачуваної втрати продуктивності.

Саме тому для захисту було обрано технологію Native AOT. Замість того, щоб покладатися на сторонні інструменти заплутування коду, проєкт компілюється безпосередньо у машинний код цільової операційної системи. Це повністю виключає етап генерації проміжної мови, тому звичні декомпілятори просто не можуть розпізнати логіку програми. Крім того, у процесі такої компіляції відбувається агресивне видалення метаданих, що перетворює

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

архітектуру системи на монолітний бінарний файл. Для потенційного аналізу алгоритмів зловмиснику довелося б проводити складний розбір на рівні асемблерних інструкцій, що значно підвищує поріг входження та безпеку розробки.

Щодо графічного конвеєра та обчислювальних шейдерів, під час проєктування розглядалася можливість використання бінарного формату SPIR-V для приховування коду алгоритмів Барнса-Хата. Однак такий підхід вимагає наявності сучасного стандарту OpenGL 4.6, що значно обмежило б коло користувачів зі старішим обладнанням. Зважаючи на це, було прийнято рішення відмовитися від SPIR-V на користь максимальної сумісності та стабільності роботи. Для базового захисту шейдерів вони інтегровані безпосередньо в тіло виконуваного файлу як вбудовані ресурси. Завдяки використанню Native AOT ці ресурси «запікаються» всередині машинного коду, що робить їх вилучення набагато складнішим завданням порівняно зі звичайним зберіганням у відкритому вигляді у файловій системі. Такий компроміс дозволив зберегти баланс між захистом авторських напрацювань, високою швидкістю обчислень та підтримкою широкого спектра відеокарт.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		50

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено стартове вікно розробленого апаратно-програмного комплексу. Програма зустрічає користувача інформаційним дашбордом, який виконує роль інтерактивного посібника. Центром композиції є привітання та візуальний логотип проекту, під яким розміщено стислий опис функціональних можливостей системи. Особливу увагу приділено ергономіці: безпосередньо у головному робочому просторі виведено таблицю гарячих клавіш та засобів навігації (керування камерою за допомогою миші, переміщення у просторі клавішами WASD та маніпуляція часовим масштабом). Це дозволяє користувачеві миттєво розпочати роботу з тривимірним середовищем без необхідності вивчення додаткової документації.

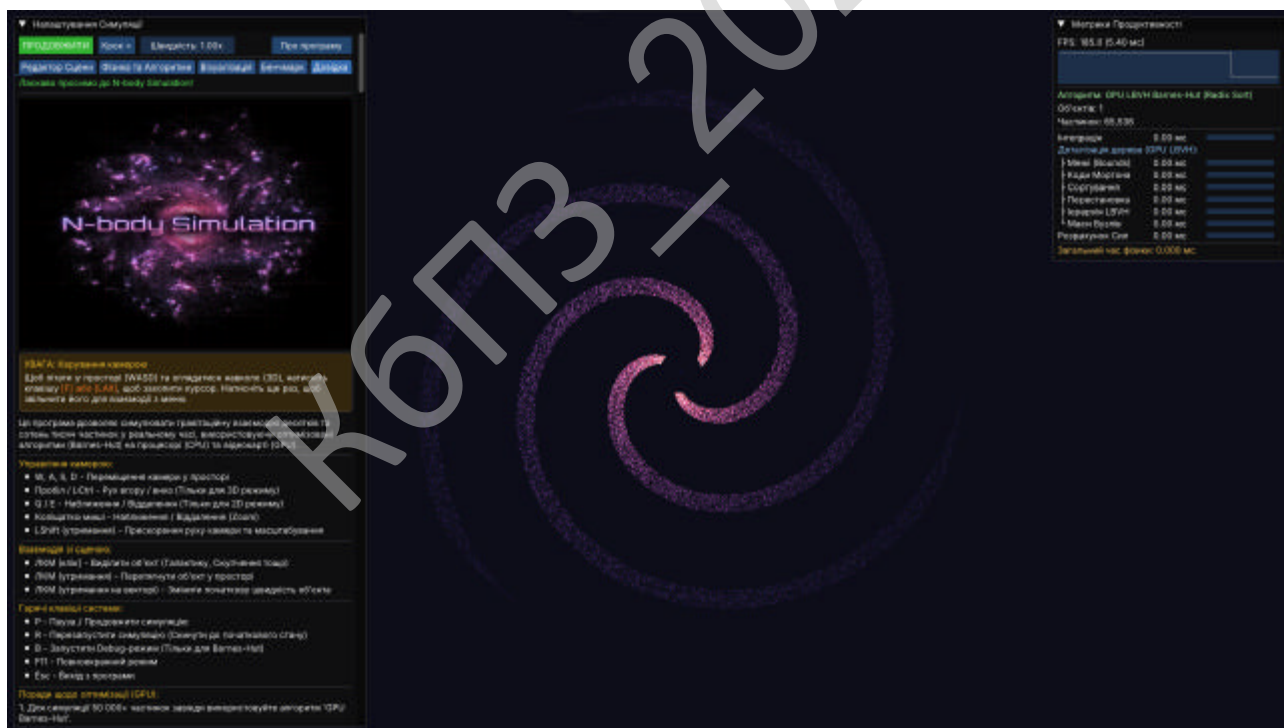


Рисунок 5.1 – Початкове вікно програми

Доступ до всього спектра можливостей системи організовано через лаконічне бокове меню, що містить наступні логічні блоки:

									Арк.
									51
Змн.	Арк.	№ докум.	Підпис	Дата	ВКРБ-123.26.0048.00.00.ПЗ				

– Модуль генерації сцен: дозволяє створювати початкові конфігурації мас, обирати типи галактичних структур та задавати кількість об'єктів для моделювання.

– Блок вибору та налаштування алгоритмів: забезпечує перемикання між обчислювальними ядрами та конфігурування точності методу Барнса-Хата.

– Панель візуалізації: містить інструменти для стилізації графічного виводу, налаштування кольорових схем та візуальних ефектів трасування об'єктів.

– Діагностичний модуль (Бенчмарк): призначений для запуску автоматизованих тестів продуктивності та збору метрик швидкодії апаратного забезпечення.

– Інформаційний блок: містить дані про автора розробки та версію програмного продукту.

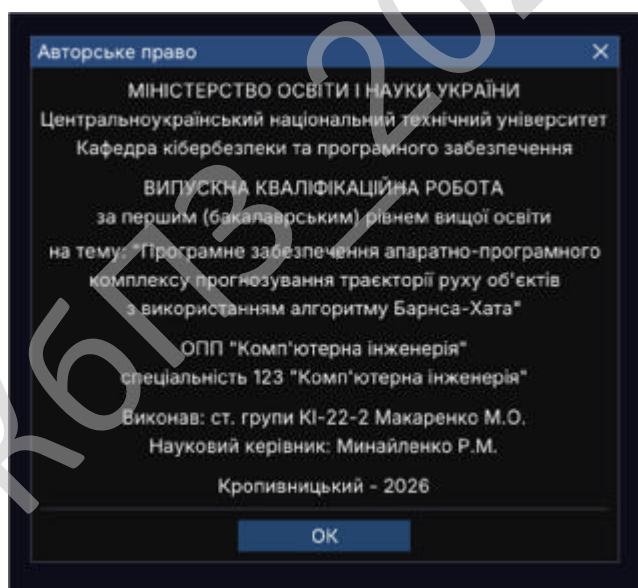


Рисунок 5.2 – Довідка

На рисунку 5.2 зображено форму авторського права, де як модель розповсюдження обрано ліцензію EULA. Такий вибір зумовлений значною науковою та практичною цінністю реалізованих у проєкті алгоритмів для GPU. Використання закритої моделі ліцензування дозволяє зберегти ці напрацювання

як інтелектуальну власність автора та надійно захистити їх від несанкціонованого копіювання чи спроб реверс-інжинірингу.

У межах цієї роботи ліцензія EULA виступає інструментом юридичного закріплення правил використання симулятора. Вона чітко регламентує умови експлуатації продукту, обмежуючи можливість стороннього втручання в структуру програми або її комерційного використання без дозволу розробника. Це гарантує безпеку авторських ідей та збереження цілісності складних архітектурних рішень.

Такий підхід перетворює розробку на захищений і стабільний інструмент для фізичного моделювання. Використання пропрієтарної ліцензії забезпечує надійний юридичний фундамент для подальшого професійного застосування програми або її розвитку в межах спеціалізованих досліджень у майбутньому.

К6ПЗ-2026

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

6 ОСНОВНІ ВИСНОВКИ

У ході виконання дипломної роботи було розроблено високопродуктивний програмний комплекс для моделювання гравітаційної взаємодії в системах N-тіл. Основним завданням проєкту стало ефективне перенесення складних астрофізичних обчислень на архітектуру сучасних графічних процесорів GPU з використанням принципів масивного паралелізму.

Розроблена система базується на модульній архітектурі, де фізичні обчислення суворо відокремлені від графічного інтерфейсу та збору даних. Це дозволило реалізувати можливість перемикання між різними обчислювальними алгоритмами «на льоту» без зупинки симуляції. Головним результатом є створення 10-етапного обчислювального конвеєра на базі шейдерів OpenGL. Конвеєр автоматизує повний цикл роботи з алгоритмом Барнса-Хата: від сортування частинок за просторовими кодами Мортон до швидкої побудови LBVN без блокувань пам'яті.

Особливу увагу в роботі приділено точності та стабільності моделювання. Було виявлено, що стандартні критерії відкриття вузлів дерева можуть давати похибки при роботі з асиметричними структурами. Для розв'язання цієї проблеми впроваджено модифікований критерій Strict AABV MAC, який забезпечив коректний рух частинок навіть у щільних кластерах матерії. Окрім цього, шляхом тонкого налаштування використання регістрів відеокарти вдалося оптимізувати фазу розрахунку сил, що прискорило її на 35-50% без втрати точності.

Практичні випробування підтвердили ефективність обраного підходу. Завдяки переходу від прямого розрахунку сил до деревоподібної апроксимації, загальну складність алгоритму було знижено з $O(N^2)$ до $O(N \log N)$. На практиці це дозволило досягти наступних результатів: система здатна моделювати понад 4 мільйони взаємодіючих частинок у реальному часі. При середніх обсягах даних

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		54

(близько 260 тис. об'єктів) розроблений графічний конвеєр працює в 10 разів швидше за оптимізований процесорний код та у 160 разів швидше за класичні прямі методи розрахунку.

Проект є готовою програмою зі зручним інтерфейсом та інструментами для перевірки швидкості обчислень. Симулятор розповсюджується за ліцензією EULA, що дозволяє чітко встановити правила користування продуктом і захистити авторські алгоритми від копіювання. Такий підхід гарантує, що унікальні технічні рішення залишаться цілісними, а користувачі отримають надійний і стабільний інструмент для наукових розрахунків, тестування фізичних гіпотез та вивчення складних обчислень на відеокартах.

К6ПЗ-2026

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Karras, T. «Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees». High Performance Graphics, 2012, pp. 33-37.
2. Barnes, J., Hut, P. «A hierarchical $O(N \log N)$ force-calculation algorithm». Nature, 1986, 324, pp. 446-449.
3. Burtscher, M., Fang, K. «An efficient GPU implementation of the Barnes-Hut algorithm». Journal of Parallel and Distributed Computing, 2011, 71(12), pp. 1521-1531.
4. Nyland, L., Harris, M., Prins, J. «Fast N-Body Simulation with CUDA». GPU Gems 3, Addison-Wesley, 2007, pp. 677-695.
5. Kokosa, K. «Pro .NET Memory Management: For Better Code, Performance, and Scalability». – [2nd ed.]. – Apress. 2024. – 850 p.
6. Martin, R. C. «Clean Architecture: A Craftsman's Guide to Software Structure and Design». Prentice Hall, 2017. – 432 p.
7. Watson, B. «Writing High-Performance .NET Code». – [2nd ed.]. – Ben Watson. 2018. – 438 p.
8. Bédorf, J., Gaburov, E., Portegies Zwart, S. «A sparse octree gravitational N-body code that runs entirely on the GPU processor». Journal of Computational Physics, 2012, 231(7), pp. 2825-2839.
9. Merrill, D., Garland, M. «Single-pass Parallel Prefix Scan with Decoupled Look-back». Technical Report, NVIDIA, 2016, pp. 1-17.
10. Harris, M., Sengupta, S., Owens, J. D. «Parallel Prefix Sum (Scan) with CUDA». GPU Gems 3, Addison-Wesley, 2007, pp. 851-876.
11. Sengupta, S., Harris, M., Zhang, Y., Owens, J. D. «Scan Primitives for GPU Computing». Graphics Hardware, 2007, pp. 97-106.
12. Kirk, D. B., Hwu, W. W. «Programming Massively Parallel Processors: A Hands-on Approach». – [3rd ed.]. – Morgan Kaufmann. 2016. – 576 p.

					ВКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

13. Khronos Group. «OpenGL 4.5 Core Profile Specification». Khronos. 2017. – 858 p.
14. Sellers, G., Wright, R. S., Haemel, N. «OpenGL SuperBible: Comprehensive Tutorial and Reference». – [7th ed.]. – Addison-Wesley Professional. 2015. – 880 p.
15. Kessenich, J., Sellers, G., Shreiner, D. «OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V». – [9th ed.]. – Addison-Wesley. 2016. – 976 p.
16. Wolff, D. «OpenGL 4 Shading Language Cookbook». – [3rd ed.]. – Packt Publishing. 2018. – 496 p.
17. Rost, R. J., Licea-Kane, B. «OpenGL Shading Language». – [3rd ed.]. – Addison-Wesley. 2009. – 800 p.
18. de Vries, J. «Learn OpenGL: Beginner's guide to 3D rendering and game development». 2020. – 513 p.
19. Evans, E. «Domain-Driven Design: Tackling Complexity in the Heart of Software». Addison-Wesley. 2003. – 560 p.
20. Vernon, V. «Implementing Domain-Driven Design». Addison-Wesley Professional. 2013. – 656 p.
21. Albahari, J. «C# 12 in a Nutshell: The Definitive Reference». O'Reilly Media. 2023. – 1098 p.
22. Price, M. J. «C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals». Packt Publishing. 2023. – 822 p.
23. Richter, J. «CLR via C#». – [4th ed.]. – Microsoft Press. 2012. – 896 p.
24. Cleary, S. «Concurrency in C# Cookbook». – [2nd ed.]. – O'Reilly Media. 2019. – 256 p.
25. Akenine-Möller, T., Haines, E., Hoffman, N. «Real-Time Rendering». – [4th ed.]. – CRC Press. 2018. – 1198 p.
26. Pharr, M., Jakob, W., Humphreys, G. «Physically Based Rendering: From Theory to Implementation». – [3rd ed.]. – Morgan Kaufmann. 2016. – 1266 p.

					БКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

27. Greß, A., Zachmann, G. «GPU-ABVH: Fast AABB Tree Construction for Direct Rendering of Animated Models». Eurographics, 2006, pp. 1-8.
28. Garber, K., Lin, M., Ho, T. «Performance scaling of N-body algorithms on modern GPUs». Parallel Computing, 2015, 41, pp. 23-35.
29. Warren, M. S., Salmon, J. K. «A parallel hashed oct-tree N-body algorithm». Supercomputing, 1993, pp. 12-21.
30. Bentley, J. «Multidimensional binary search trees used for associative searching». Communications of the ACM, 1975, 18(9), pp. 509-517.
31. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M. «Computational Geometry: Algorithms and Applications». – [3rd ed.]. – Springer. 2008. – 386 p.
32. Preparata, F. P., Shamos, M. I. «Computational Geometry: An Introduction». Springer. 1985. – 390 p.
33. Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. «Introduction to Algorithms». – [4th ed.]. – MIT Press. 2022. – 1312 p.
34. Knuth, D. E. «The Art of Computer Programming, Volume 3: Sorting and Searching». – [2nd ed.]. – Addison-Wesley. 1998. – 780 p.
35. Gregory, J. «Game Engine Architecture». – [3rd ed.]. – CRC Press. 2018. – 1240 p.
36. Owens, J. D., Luebke, D., Govindaraju, N., Harris, M. «A Survey of General-Purpose Computation on Graphics Hardware». Computer Graphics Forum, 2007, 26(1), pp. 80-113.
37. Dally, W. J. «The GPU Computing Era». IEEE Micro, 2010, 30(2), pp. 56-69.
38. Flynn, M. J., Rudd, K. W. «Parallel Architectures». CRC Press. 1996. – 416 p.
39. Dally, W. J., Towles, B. «Principles and Practices of Interconnection Networks». Morgan Kaufmann. 2004. – 584 p.
40. Grama, A., Gupta, A., Karypis, G., Kumar, V. «Introduction to Parallel Computing». – [2nd ed.]. – Pearson. 2003. – 656 p.

					БКРБ-123.26.0048.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		58

41. Pacheco, P. S. «An Introduction to Parallel Programming». Morgan Kaufmann. 2011. – 351 p.
42. Quinn, M. J. «Parallel Programming in C with MPI and OpenMP». McGraw-Hill Education. 2003. – 544 p.
43. Matloff, N. «Parallel Computing for Data Science: With Examples in R, C++ and CUDA». Chapman and Hall/CRC. 2014. – 320 p.
44. Arpaci-Dusseau, R. H., Arpaci-Dusseau, A. C. «Operating Systems: Three Easy Pieces». – [1st ed.]. – Arpaci-Dusseau Books. 2018. – 714 p.
45. Tanenbaum, A. S., Bos, H. «Modern Operating Systems». – [4th ed.]. – Pearson. 2014. – 1136 p.
46. Silberschatz, A., Galvin, P. B., Gagne, G. «Operating System Concepts». – [10th ed.]. – Wiley. 2018. – 976 p.
47. Stevens, W. R., Rago, S. A. «Advanced Programming in the UNIX Environment». – [3rd ed.]. – Addison-Wesley. 2013. – 1032 p.
48. Kleppmann, M. «Designing Data-Intensive Applications». O'Reilly Media. 2017. – 616 p.
49. Patterson, D. A., Hennessy, J. L. «Computer Organization and Design: The Hardware/Software Interface». – [5th ed.]. – Morgan Kaufmann. 2013. – 800 p.
50. Jernigan, S. «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.
51. Aila, T., Laine, S. «Understanding the efficiency of ray traversal on GPUs». Proceedings of High Performance Graphics, 2009, pp. 145-149.
52. Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F. «Computer Graphics: Principles and Practice». – [3rd ed.]. – Addison-Wesley Professional. 2013. – 1264 p.
53. Garland, M., Le Grand, S., Nickolls, J., et al. «Parallel Computing Experiences with CUDA». IEEE Micro, 2008, 28(4), pp. 13-27.