МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЩЕРБАНЬ ІВАН ОЛЕКСАНДРОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р техн. наук, професор
_____ Наталія ВЕСЕЛОВСЬКА
«____»_____ 2026 р.

ПРОЦЕДУРНА ГЕНЕРАЦІЯ ЛАНДШАФТІВ ТА ЕКОСИСТЕМ ДЛЯ ІГРОВОГО ДОДАТКУ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Роман БАБАКОВ, професор кафедри
інформаційних технологій,
д-р техн. наук, доцент
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2026

АНОТАЦІЯ

Щербань І. О. Процедурна генерація ландшафтів та екосистем для ігрового додатку. Спеціальність 122 – Комп’ютерні науки. Донецький національний університет імені Василя Стуса. Вінниця, 2026.

У кваліфікаційній (бакалаврській) досліджено сучасні методи процедурного моделювання тривимірних ландшафтів та екосистем. Розроблено програмний комплекс для генерації планет у реальному часі з використанням багатопотокових обчислень, що забезпечує високу продуктивність та реалістичність рельєфу й розподілу екосистем. Практична цінність полягає у можливості застосування результатів у комп’ютерних іграх з відкритим світом та космічних симуляторах.

Ключові слова: процедурна генерація, ландшафт, екосистема, DOTS, Unity.

ABSTRACT

Shcherban I. O. Procedural Generation of Landscapes and Ecosystems for a Gaming Application. Specialty 122 «Computer Science», educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia, 2026.

The bachelor’s thesis investigates modern methods of procedural modeling of three-dimensional landscapes and ecosystems. A software system was developed for real-time planet generation using multithreaded computations, ensuring high performance and realistic terrain and ecosystem distribution. The practical value lies in applying the results to open-world games and space simulators.

Keywords: procedural generation, landscape, ecosystem, DOTS, Unity.

ЗМІСТ

ВСТУП……..5
РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ МЕТОДІВ ПРОЦЕДУРНОГО МОДЕЛЮВАННЯ ЛАНДШАФТІВ……...7
1.1 Аналіз категорій і понять процедурного моделювання та сутність проблеми..7
1.2 Порівняльний аналіз аналогів та оптимізація генерації……16
1.3 Обґрунтування вибору ігрового рушія……...20
1.4 Постановка задачі……21
РОЗДІЛ 2. АРХІТЕКТУРА ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ……23
2.1 Архітектура системи та конвеєр обробки даних на основі технологічного стеку DOTS……23
2.2 Математичні моделі генерації сферичної поверхні та оптимізації просторової зв'язності……26
2.3 Моделювання тектонічних процесів та двофазної ерозії макрорельєфу…….30
2.4 Кліматична симуляція та класифікація біомів за матрицею Вайттекера…….35
2.5 Математичний апарат стереографічного проєктування та асинхронної тріангуляції Делоне……...38
2.6 Процедурна побудова та двовимірна фільтрація растрових карт……...40
2.7 Опис та аналіз алгоритмів процедурного моделювання сферичного ландшафту……..42
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ, НАЛАШТУВАННЯ СЦЕНИ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ СИСТЕМИ ГЕНЕРАЦІЇ……..53
3.1. Структура програмного продукту та архітектура модулів……...53
3.1.1 Базові структури даних……..……..53
3.1.2 Опис супутніх допоміжних модулів……54
3.2. Практична реалізація конвеєра генерації……..55
3.2.1. Етап формування сферичної основи та тектонічної симуляції……...55
3.2.2 Моделювання клімату, вологості та ерозійних процесів……57

3.3. Візуалізація та налаштування інтерактивної сцени у Unity……...59
3.3.2 Налаштування сцени, камери та інтерфейсу користувача…….59
3.3.3. Верифікація тривимірної генерації об'єкта та рельєфних нерівностей…...59
3.3.4. Розробка підсистеми візуалізації атмосфери та динамічного обертання….62
3.4. Підсистема растрового експорту……...63
3.5. Модульне тестування та експериментальний аналіз продуктивності…….67
ВИСНОВКИ……...69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……..71
ДОДАТКИ……...75

ВСТУП

На сьогоднішній день сучасна індустрія комп'ютерних ігор та віртуальної реальності перебуває в стані постійного розвитку, прагнучи до створення масштабних та реалістичних віртуальних світів. Проте ручна розробка кожної деталі тривимірного простору вимагає колосальних часових і фінансових витрат. Для вирішення цієї проблеми використовують методи автоматичного моделювання ландшафтів. Водночас більшість існуючих рішень працюють без урахування природної логіки, а створення реалістичного рельєфу в реальному часі значно перевантажує комп'ютер, викликаючи затримки у роботі програми. Тому дослідження, спрямоване на розробку ефективної та оптимізованої системи для створення природних ландшафтів, є надзвичайно актуальним.

Об’єктом дослідження в цій роботі є процес автоматичної генерації тривимірного віртуального простору та розподілу рослинності, а предметом – методи, математичні моделі та програмні алгоритми для створення реалістичного рельєфу і оптимізації роботи додатка. Метою роботи є дослідження, обґрунтування та практична розробка високопродуктивної програмної системи для автоматичного створення реалістичних планет та їхніх екосистем у реальному часі.

Для досягнення поставленої мети було виконано такі завдання: проаналізовано існуючі теоретичні підходи та математичні моделі моделювання ландшафтів, досліджено природні механізми формування рельєфу та розподілу рослинності , здійснено аналіз відомих практичних аналогів, обґрунтовано вибір середовища розробки, а також розроблено та оптимізовано програмний комплекс для моделювання сферичної планети.

У роботі використано теоретичні методи аналізу фізико-географічних процесів, сучасні інструменти програмування, а також методи багатопотокової оптимізації для розподілу обчислень.

Практична цінність роботи полягає у створенні ефективного програмного інструменту, який дозволяє швидко створювати та завантажувати віртуальні

планети без затримок і втрати продуктивності програми. Створені компоненти можуть бути використані під час розробки комп'ютерних ігор з відкритим світом або космічних симуляторів.

Структура роботи складається з вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі розглянуто теоретичні основи автоматичного моделювання простору та аналіз існуючих підходів. Другий розділ присвячений розробці алгоритмів формування рельєфу, клімату та архітектурі програмного комплексу. У третьому розділі наведено результати тестування, оцінку ефективності оптимізації та аналіз продуктивності розробленого додатка.

РОЗДІЛ 1
ТЕОРЕТИЧНИЙ АНАЛІЗ МЕТОДІВ ПРОЦЕДУРНОГО МОДЕЛЮВАННЯ ЛАНДШАФТІВ

1.1 Аналіз категорій і понять процедурного моделювання та сутність проблеми

Сучасна індустрія комп'ютерних ігор та симуляцій віртуальної реальності вимагає постійного масштабування обсягу та деталізації контенту, що призводить до виникнення так званої "проблеми гори контенту" (Mountain of Content Problem) [1]. Ручне моделювання великих відкритих світів потребує залучення значних фінансових та людських ресурсів, через що процедурна генерація вмісту (Procedural Content Generation, PCG) перетворилася з допоміжного інструменту на базовий елемент архітектури сучасних ігрових рушіїв [1, 2] .

Історично процедурне моделювання розвивалося від найпростіших стохастичних абстракцій до складних фізико-географічних симуляцій, що інтегрують закони гідрології, тектоніки та кліматології.

Традиційним базисом для представлення рельєфу в комп’ютерній графіці є карти висот (heightmaps) – двовимірні регулярні сітки, де кожен піксель зберігає значення висоти відповідної координати. Для ініціалізації цих карт найчастіше застосовують стохастичні методи на основі фрактального шуму [2,3].

Шум Перліна (Perlin Noise) та Сімплекс-шум (Simplex Noise) генерують плавні псевдовипадкові коливання висот шляхом інтерполяції випадкових градієнтних векторів. Накладання кількох шарів такого шуму з послідовним зростанням частоти та зменшенням амплітуди – процес, відомий як дробовий броунівський рух (fractional Brownian motion, fBm) – дозволяє створювати фрактальні структури, що імітують гірські хребти[2,3].

Аналогічний результат дають геометричні методи підрозділення, такі як алгоритм середньоточкового зміщення (Midpoint Displacement) та алгоритм

діамант-квадрат (Diamond-Square), які ітераційно розбивають площину рельєфу, додаючи випадкові згасаючі зсуви у нових вершинах[2, 4].

Попри високу продуктивність, чисті стохастичні методи позбавлені геоморфологічної логіки: вони не здатні моделювати взаємозв'язок між височиною та руслами річок, через що згенеровані ландшафти виглядають неприродно та вимагають додаткового коригування.

Для досягнення реалістичності ландшафту на великих масштабах використовують підходи, засновані на принципах гідрології [5,6]. У моделі Женво процес створення рельєфу починається з генерації ієрархічної річкової мережі, яка визначає загальну структуру майбутнього ландшафту [6].

Генерація річок відбувається у межах заданого контуру шляхом зростання геометричного графу з початкових насінин (гирл річок) на узбережжі. Зростання графу контролюється ієрархічною числовою схемою Гортона-Стралера (Horton-Strahler's ordering), яка описує складність розгалуження річкових систем. На основі отриманої мережі річок рельєф сегментується на водозбірні басейни за допомогою діаграми Вороного.

Висота гребенів та вододілів розраховується детерміновано залежно від висоти найближчих річкових вузлів та відстані до них. Щоб запобігти виникненню аномальних вертикальних урвищ, нахили рельєфу обмежуються умовою Ліпшиця [6]:

$$|p_z - p_{zi}| < \kappa(p) \cdot d(p, p_i) \qquad (1.1)$$

де p_z – висота нової точки,

p_{zi} – висота існуючого вузла,

$d(p, p_i)$ – відстань між ними,

$\kappa(p)$ – локальний коефіцієнт нахилу рельєфу, що визначається картою нахилів.

Такий підхід повністю ліквідує проблему локальних мінімумів (замкнених западин без стоку), які часто утворюються під час стохастичної генерації, та забезпечує плавне, фізично коректне стікання води до океану.

Великим недоліком стандартних карт висот є неможливість представлення тривимірних елементів рельєфу, таких як підземні печери, скельні нависання чи природні арки. Для вирішення цієї проблеми застосовують волюметричні структури, зокрема розріджені воксельні октодерева (Sparse Voxel Octrees) або шаруваті структури матеріалів[2].

У моделі гідрологічної генерації рельєф може бути описаний у вигляді конструкційного дерева (на кшталт Constructive Solid Geometry, CSG) [1, 5, 6], де листя дерева представляє процедурні примітиви (пагорби, річкові долини, скелі), а внутрішні вузли визначають операції над ними, такі як змішування (blending) чи вирізання (carving). Кожен вузол конструкційного дерева описується парою аналітичних функцій: висотою $h(p)$ та полем впливу $w(p)$.

Для довільної точки p на площині операція плавного змішування двох піддерев A та B визначається як:

$$h_{\mathrm{B}(A,B)}(p) = \frac{w_A(p)h_A(p) + w_B(p)h_B}{(p)w_A(p) + w_B(p)} \tag{1.2}$$

$$w_{\mathrm{B}(A,B)}(p) = \frac{w_A(p) + w_B(p)}{2} \tag{1.3}$$

де $h_B(A,B)(p)$ – результуюча висота змішаного рельєфу в точці p, м;

$w_A(p)$ – значення поля впливу піддерева A в точці p;

$h_A(p)$ – локальна висота рельєфу поддерева A в точці р, м;

$w_B(p)$ – значення поля впливу піддерева B в точці p;

$h_B(p)$ – локальна висота рельєфу поддерева B в точці p, м;

$w_B(A,B)(p)$– результуєче поле впливу змішаних піддерев у точці p [5].

Водночас операція заміщення (carving), яка використовується для врізання річкових русел у гірські породи, описується як асиметричний оператор:

$$h_{\mathrm{C}}(A,B)(p) = \big(1 - w_B(p)\big)h_A(p) + w_B(p)h_B(p) \tag{1.4}$$

$$w_{\mathrm{C}}(A,B)(p) = \big(1 - w_B(p)\big)w_A(p) + w_B(p) \tag{1.5}$$

де $h_C(A,B)(p)$ – результуюча висота рельєфу після операції заміщення в точці p, м;

$w_B(p)$– значення поля впливу заміщувального піддерева B в точці p;

$h_A(p)$ – локальна висота основного рельєфу піддерева A в точці p, м;

$h_B(p)$ – висота врізаного річкового русла піддерева B в точці p, м;

$w_C(A,B)(p)$ – результуєче поле впливу після операції заміщення в точці p.

Ця модель дозволяє представляти великі ландшафти у компактному векторному вигляді з необмеженим рівнем деталізації, де обчислення висоти конкретної точки виконується "на льоту" під час рендерингу.

Для генерації макрорельєфу на глобальному (планетарному) рівні застосовують тектонічне моделювання. Поверхня сфери розбивається на систему літосферних плит за допомогою розподілу центрів на основі генерації диска Пуассона або сферичних комірок Вороного [4, 9-12].

Кожній плиті i призначається кутова швидкість обертання ω_i навколо власного полюса Ейлера. Швидкість руху точки рельєфу v на сфері з радіус-вектором r під впливом плити визначається через векторний добуток:

$$\vec{v} = \overrightarrow{\omega_i} \times \vec{r} \tag{1.6}$$

де $\vec{v}$ – лінійна швидкість руху точки рельєфу на поверхні сфери, м/с;

$\overrightarrow{\omega_i}$ – вектор кутової швидкості обертання і-ї літосферної плити навколо полюса Ейлера, рад/с;

$\vec{r}$ – радіус-вектор точки на сферичній поверхні планети, м [12].

На межах зіткнення плит (конвергентні кордони) симулюється процес орогенезу (гіркоутворення) та субдукції [24]. Якщо дві плити зближуються, виникає великий тектонічний тиск P, який розраховується як скалярний добуток різниці швидкостей та нормалі до кордону:

$$P = (\overrightarrow{v_1} - \overrightarrow{v_2}) \cdot \vec{n} \tag{1.7}$$

де P – локальний тектонічний тиск стиснення гірських порід у зоні конвергенції, Па;

$\overrightarrow{v_1}$ – лінійна швидкість руху першої стикаючої континентальної плити в точці контакту, м/с;

$\overrightarrow{v_2}$ – лінійна швидкість руху другої стикаючої континентальної плити в точці контакту, м/с;

$\vec{n}$ – нормальний одиничний вектор до лінії межі зіткнення плит, безвимірна величина [12].

При зіткненні двох товстих континентальних плит порода зазнає вертикального зсуву та утворює гірські хребти, висота яких пропорційна тиску P У зонах субдукції, де тонша океанічна кора занурюється під континентальну, утворюються глибокі океанічні жолоби. На межах розходження (дивергентні кордони) симулюється рифтогенез – розсунення літосфери, що веде до утворення рифтових долин та нової океанічної кори[12].

Термальна ерозія моделює фізичне вивітрювання та осипання уламків гірських порід під впливом гравітації. Процес базується на порівнянні різниці висот між поточною точкою та її сусідами з критичним кутом укосу (talus threshold) [2, 7]. Якщо різниця перевищує поріг, частина матеріалу переноситься вниз за рельєфом.

Розрахунок об'єму матеріалу, що осипається, виконується за формулою :

$$h_{move} = \left(\frac{\sum \Delta h_k}{N_{lower}}\right) \cdot K_e \qquad (1.8)$$

де Δh_k – перепад висоти з сусідом, який перевищив критичний поріг;

N_{lower} – кількість таких нижчих сусідів;

K_e – коефіцієнт швидкості ерозії [7] .

Термальна симуляція ефективно згладжує гострі вершини, формуючи пологі конуси винесення біля підніжжя гір.

Гідравлічна ерозія є основним фактором формування річкових долин та ярів [5-7]. Існує кілька підходів до її симуляції в комп'ютерній графіці [13]:

- Модель віртуальних труб (Virtual Pipes): вода розглядається як водна гладь над кожним пікселем карти висот, а її перетікання між сусідніми комірками симулюється через систему уявних сполучених труб на основі рівнянь гідродинаміки [7]. Це дає змогу точно розраховувати вектор швидкості потоку для кожної комірки.
- Частинки води (SPH - Smoothed Particle Hydrodynamics): по поверхні рельєфу запускаються незалежні частинки води (краплі), які рухаються під

дією сили тяжіння, взаємодіючи з грунтом [13,15]. Кожна крапля розчиняє певну кількість грунту на стрімких ділянках (ерозія) та осаджує його в низовинах (депонування).

- Степеневий закон потужності потоку (Stream Power Law): спрощений аналітичний підхід, де швидкість ерозії E у точці рельєфу визначається через площу водозбору A та нахил S:

$$E = K \cdot \mathrm{A}^m \cdot S^n \tag{1.9}$$

де E – швидкість гідравлічної ерозії річкового русла, м/рік;

K – коефіцієнт ерозійної опірності підстилаючих гірських порід руйнуванню, $1/\text{рік}$;

A^m– площа водозбору для поточної комірки рельєфу, м^2;

S^n – локальний ухил рельєфу (тангенс кута нахилу поверхні);

m – емпіричний показник степеня впливу загальної площі збору вологи;

n – емпіричний показник степеня впливу ухилу рельєфу [6].

Площа водозбору A розраховується ітераційно як сума площ усіх вищих точок ландшафту, що стікають у поточну комірку. Цей метод є надзвичайно швидким і дозволяє генерувати реалістичні ерозійні структури без покрокової симуляції руху води [6,7].

Для тривалого моделювання планетарних циклів впроваджують механізм захисту гірських піків від повного нівелювання. На пізніх ітераціях симуляції ерозійна здатність води штучно зменшується за допомогою коефіцієнта згасання :

$$ErosionMultiplier = max\big(0.0{,}1.0 - (Step - Step_{threshold}) \cdot K_{taper}\big)$$

де $ErosionMultiplier$ – підсумковий ерозійний мультиплікатор інтенсивності змиву;

$Step$ – поточний порядковий номер ітерації гідродинамічної симуляції, одиниць;

$Step_{threshold}$ – гранична ітерація початку стабілізації та консервації рельєфу, одиниць;

K_{taper} – коефіцієнт лінійного згасання ерозійної сили за одну ітерацію моделювання, $1/\text{крок}$.

Це дозволяє зберегти гострі форми вершин на фінальних етапах формування ландшафту, коли тектонічний підйом припиняється [12].

Розподіл рослинного покриву та біологічного наповнення віртуальних світів спирається на принципи екологічного детермінізму, де тип і щільність рослинності визначаються локальними фізичними та кліматичними параметрами середовища [1, 15].

Базою для великої комп'ютерної симуляції природних зон (біосфери) є класична модель біомів Роберта Вайттекера (Whittaker) [4, 15]. Ця схема дозволяє спростити складні кліматичні дані й розділити всі типи рослинності планети на дев'ять основних категорій (макробіомів). Розподіл зон у моделі відбувається на основі двох головних кліматичних показників: середньорічної температури повітря ($T_{avg}, °C$) та загальної річної кількості опадів (P_{annual}, см/рік).

Для наочного розуміння цієї системи використовується графік розподілу екосистем (діаграма Вайттекера):

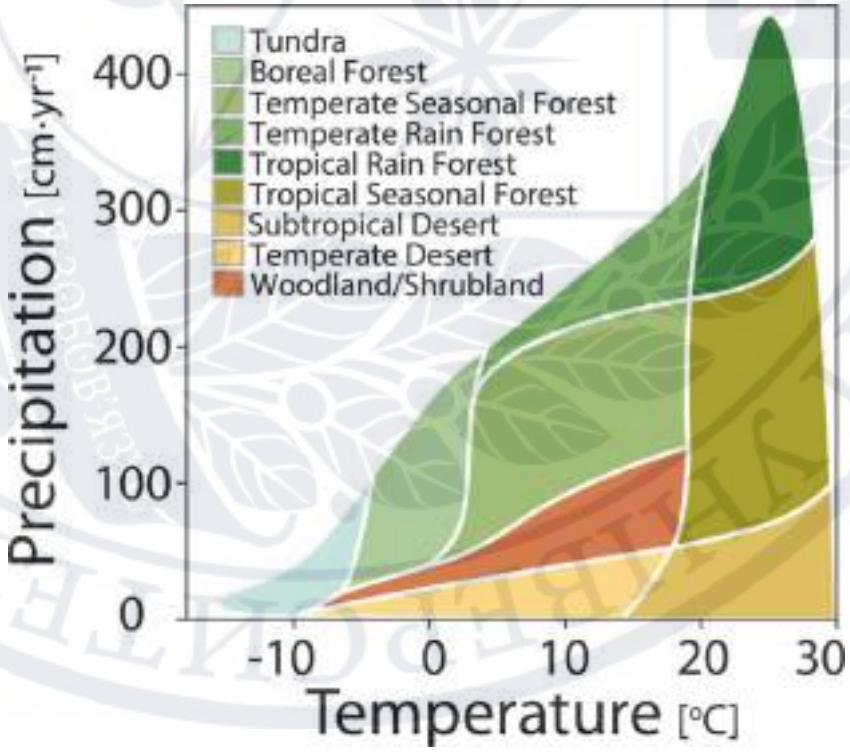

Рисунок 1.1 -Графік розподілу екосистем [16]

- Осі графіка: Вертикальна вісь відображає середньорічну температуру (від +30°C на екваторі до -15°C на полюсах). Горизонтальна вісь показує рівень опадів (від 0 до 450 см на рік).
- Екваторіальний та тропічний сектор (вища частина графіка): При стабільно високій температурі (понад +20°C) тип екосистеми повністю залежить від води. За умов максимальної вологості формується *Тропічний дощовий ліс* (Tropical rain forest). Зі зменшенням кількості опадів ландшафт послідовно змінюється на *Сезонний тропічний ліс* (Tropical seasonal forest), далі на суху *Савану* (Savanna), і за повної аридності – на *Субтропічну пустелю* (Subtropical desert).
- Помірний сектор (середня частина графіка): У діапазоні температур від 0°C до +20°C висока вологість створює *Помірний дощовий ліс* (Temperate rain forest), середній рівень опадів формує *Широколисті ліси* (Temperate deciduous forest), низький – *Степи та чагарники* (Temperate grassland/desert).
- Холодний сектор (нижня частина графіка): Коли середньорічна температура падає нижче 0°C, різноманітність природних зон різко скорочується. Тут домінує хвойна *Тайга* (Taiga/Boreal forest), яка за екстремального холоду та сухості переходить у *Тундру* (Tundra).

Для реалізації динамічної планетарної симуляції в кожній точці віртуальної планети ці два параметри (температура й опади) розраховуються за допомогою фізико-математичних алгоритмів [15,16].

Температурний режим поверхні віртуального світу розраховується як залежність від географічного положення та висоти рельєфу:

- Географічна широта: Визначає базове тепло. Що далі точка знаходиться від екватора і ближче до полюсів, то менше сонячного світла вона отримує, тому базова температура лінійно падає.
- Висота над рівнем моря: Моделює похолодання в горах. Вертикальне зниження температури повітря (коефіцієнт *lapse rate*) програмується як проста лінійна функція: на кожні 100 метрів підйому вгору температура

автоматично зменшується (зазвичай у межах від $-0.6°C$ до $-0.65°C$). Це дозволяє системі самостійно генерувати умови високогірної тундри чи альпійських лук на вершинах гір, навіть якщо самі гори розташовані в теплих регіонах.

Карта вологості та опадів має значно складнішу структуру, ніж температурна. Вона створюється завдяки поєднанню глобальних атмосферних процесів та впливу гірського рельєфу:

- Трикомірна система загальної циркуляції атмосфери: Рух повітряних мас і планетарний розподіл вологи моделюються через роботу трьох замкнених циркуляційних контурів (комірок) у кожній півкулі [16]:
    - Комірка Гадлі (Hadley cell): створює зону постійного низького тиску та сильних дощів на екваторі (формуючи екваторіальні ліси), а також скидає сухе повітря в районі тропіків, утворюючи зони високого тиску й перманентної посухи (смуга пасатних пустель, як Сахара).
    - Комірка Феррела (Ferrel cell): формує зону помірних широт, де панують західні вітри й часті циклони, що забезпечує регулярні дощі для лісів помірного поясу.
    - Полярна комірка (Polar cell): приносить сухе й холодне повітря з полюсів, створюючи зони арктичних пустель.
- Ефект «дощового затінку» в горах (mountain rain shadow effect): На глобальні вітри накладається локальний чинник – великі гірські хребти, які виступають бар'єром для хмар [15,16]:
    - Навітряний схил (повернутий до вітру): коли вологе повітря натрапляє на гору, воно змушене підніматися вгору. Під час підйому повітря охолоджується, волога конденсується і випадає у вигляді рясних дощів. Тут утворюються зони з буйною рослинністю.
    - Підвітряний схил (захищений горою): перевалившись через хребет, повітряна маса, яка вже втратила майже всю воду, починає спускатися. При спусканні повітря нагрівається і стає дуже сухим. Як наслідок, з іншого боку гори утворюється «зона затінку» – засушлива

територія, де через дефіцит вологи формуються орографічні пустелі та напівпустелі.

Для створення геометричних моделей окремих дерев та кущів використовуються системи Ліндемаєра (L-systems), які описують процеси галуження за допомогою формальних граматик.[1] Стохастичні правила переписування дозволяють створювати унікальні варіації крон у межах одного виду [1, 2].

Для масового розсадження дерев застосовують екологічні карти щільності, що базуються на висоті, нахилі, експозиції схилу (південні схили у північній півкулі отримують більше світла та є сухішими) та вологості [15]. Розподіл точок посадки здійснюється за допомогою генерації диска Пуассона (Poisson Disk Sampling) [1, 4, 9]. Цей алгоритм гарантує, що між будь-якими двома деревами зберігається відстань, не менша за поріг R_{min} , що запобігає перетину тривимірної геометрії стовбурів та імітує природну конкуренцію рослин за світло й поживні речовини [1].

1.2 Порівняльний аналіз аналогів та оптимізація генерації

Розробка ефективного генератора планети вимагає ретельного аналізу архітектурних рішень та черговості етапів генерації. Традиційна "природна" черговість формування ландшафтів передбачає паралельний та хаотичний розвиток усіх систем. Проте в ігрових рушіях такий підхід є вкрай неефективним, оскільки зміна рельєфу змушує систему постійно перераховувати вегетаційні колізії та вологість [17, 18, 20].

Тому в рантайм-системах застосовують оптимізовану лінійну процедуру генерації, де кожна наступна фаза детерміновано спирається на завершені дані попередньої [3, 15, 20]:

$$\text{Тектоніка(планетарнийрельєф)} \rightarrow \text{ЕрозіяпородитаГідрологія} \rightarrow$$
$$\rightarrow \text{Атмосфера(кліматичнікарти)} \rightarrow \text{Вегетація(біоми)}$$

Це повністю ліквідує накладні витрати на видалення рослинних об'єктів з некоректних зон (наприклад, з русел річок або гірських піків) і дозволяє надійно вписувати природні структури в геометрію ландшафту.

Таблиця 1.1 Аналіз класів алгоритмів генерації рельєфу [2, 4-8, 12]

Клас алгоритмів	Реалістичність результату	Продуктивність та швидкість	Можливість контролю користувачем	Сфера застосування
Стохастичні (fBm, Перлін, Сімплекс)	Низька (відсутність річкових долин та улоговин)	Екстремально висока ($O(1)$ для окремої точки)	Низька (керування лише глобальними параметрами шуму)	Швидке прототипування, безмежні світи (на кшталт Minecraft)
Гідрологічні (Женво, комірки Вороного)	Дуже висока (абсолютна топологічна коректність долин)	Середня (вимагає попереднього розрахунку графу річок)	Висока (можливість ескізного малювання головних річок)	Реалістичні ландшафти, симулятори польотів
Симуляція ерозії (Гідравлічна, термальна)	Максимальна (природні змиви та осипи грунту)	Дуже низька (вимагає сотень ітерацій над картою)	Вкрай низька (результат залежить від всього циклу)	Генерація локальних мап високої деталізації (Gaea, World Machine)
Тектонічні (літосферні плити, орогенез)	Висока (природні форми континентів та ланцюгів гір)	Низька (ітераційний розрахунок сил на межах плит)	Середня (налаштування швидкості та кількості плит)	Глобальні стратегії, планетарна генерація (Rock 3)

Таблиця 1.2 Статистика генерації та оцінка продуктивності моделі [5, 6]

Показник ландшафту та етапи обчислень	Тест А (Hoya Island - flat)	Тест Б (Hoya Island - mountain)	Тест В (Lowland River Valley)	Тест Г (Braided River Highlands)
Площа домену ($км^2$)	3055	3368	970	3050
Довжина річкової мережі (*км*)	1978	2106	399	1515
Кількість ландшафтних примітивів	69065	76332	18941	3465
Кількість річкових примітивів	4905	5321	885	62205
Час генерації графу (*сек*)	4.0	5.3	0.1	0.5
Час розрахунку потоків/басейнів (*сек*)	0.9	1.1	1.8	4.8
Час побудови дерева примітивів (*сек*)	4.7	4.9	0.1	0.4
Час оцінки мешу (1024 ×1024) (*сек*)	1.0	1.2	1.0	0.8

Дані у табл. 1.1 та у табл 1.2, що векторне представлення ландшафту є надзвичайно компактним (в середньому 1 $км^2$ території вимагає лише 1.5 кБ дискового простору), а загальний час генерації навіть для гігантських територій площею понад 3000 $км^2$ становить близько 10-12 секунд, що робить цей підхід придатним для динамічного завантаження в іграх [5,6].

Для порівняння, у магістерській дисертації Казіміра Пійспи див. табл. 1.3 було проведено детальне дослідження апаратного споживання та візуальної оцінки трьох різних PTG-генераторів, створених на рушії Unity [3, 15, 19, 20].

Таблиця 1.3 Порівняльні результати тестування генераторів [15]

Контрольні показники та сценарії	Метод 1 (Noise-based)	Метод 2 (Voxel)	Метод 3 (2D Tiling)
Середнє використання RAM (Гб)	1.77	1.82	1.24
Споживання пам'яті в сценарії Island/Ocean (Гб)	2.05	2.50	1.24 (без варіацій)
Абсолютний успіх у сценарії 1 (Flat)	Успіх (рівний рельєф, дерева)	Помилка (аномальні перепади висот)	Успіх (плоске поле плиток)
Абсолютний успіх у сценарії 2 (Woodland)	Успіх (ліс, озера, пагорби)	Успіх (озера, розділення біомів)	Успіх (коректне плиточне поле)
Абсолютний успіх у сценарії 3 (Mountains)	Успіх (гори з текстурами снігу)	Помилка (немає озер, ліміт вершин)	Успіх (білі вершини на 2D-плитках)
Абсолютний успіх у сценарії 4 (Island/Ocean)	Успіх (океан, групи островів)	Успіх (океан, скельні виходи)	Успіх (океан та острів по центру)
Позитивна візуальна оцінка користувачів	67% (найвища привабливість)	8% (найменша цікавість)	25% (середній рівень інтересу)
Негативна оцінка користувачів (меншість)	25%	33%	42%

1.3 Обґрунтування вибору ігрового рушія

Вибір програмної платформи для реалізації процедурної генерації ландшафтів та екосистем визначає не лише якість візуалізації, а й загальну продуктивність симуляційних систем, див у табл. 1.4.

Таблиця 1.4 Порівняльний аналіз ігрових рушіїв [17, 18]

Характеристика платформи	Unity (Unity 6 / DOTS)	Unreal Engine 5 (UE5)	Godot 4
Основна мова програмування	C# (HPC# для Burst)	C++ / Blueprints	GDScript / C# / C++ (GDExtension)
Архітектура пам'яті	Керований.NET (Mono/IL2CPP), але DOTS дає ручний контроль без GC	Ручне керування пам'яттю з розумними вказівниками (без GC-фризів)	Керований підрахунок посилань, ручне керування у GDExtension
Багатопоточність	Safe C# Job System з компіляційним тред-сейф аналізом	Task Graph System (вимагає захисту спільних ресурсів)	Базовий клас Thread, ризик race conditions у великих системах
Компіляція та оптимізація	LLVM-базований Burst-компілятор, SIMD векторизація	Нативна компіляція C++ комерційними компіляторами платформ	GDScript інтерпретується (повільний), GDExtension компілюється через scons
Вбудовані PCG інструменти	Сторонні нодальні пакети (Infinite Lands, Map Magic)	PCG Framework (нодальна генерація біомів на CPU)	Слабкий інструментарій, повна відсутність систем стрімінгу світів
Продуктивність важкої логіки	Екстремально висока (0.9 мс для 60 складних агентів)	Дуже висока нативна (3.2 мс на C++), але Blueprints вкрай повільні	Середня/Низька (GDScript викликає мікрофризи на великих циклах)

Вихід Unity 6 суттєво зміцнив його позиції в AAA-сегменті завдяки технологічному стеку DOTS (Data-Oriented Technology Stack). DOTS базується на трьох китах:

1. Entity Component System (ECS): перехід від об'єктно-орієнтованої парадигми (MonoBehaviour) до орієнтованої на дані, де дані сутностей зберігаються лінійно в пам'яті у вигляді масивів структур.
2. C# Job System: інструмент безпечної багатопоточності, який автоматично розподіляє завдання генерації на всі доступні ядра CPU без загрози виникнення взаємного блокування чи пошкодження даних.
3. Burst Compiler: LLVM-базований компілятор, що транслює код High Performance C# (HPC#) у високоефективний нативний машинні команди, використовуючи векторні розширення процесора (SIMD інструкції SSE, AVX, AVX2, ARM NEON).

Burst-компілятор усуває накладні витрати на збирання сміття (.NET Garbage Collector), оскільки HPC# працює виключно з blittable-типами та структурами у некерованій пам'яті (Native Containers). Завдяки агресивному аналізу аліасингу Burst генерує ассемблерний код, який за швидкістю математичних розрахунків повністю тотожний, а іноді й перевершує C++. Це дозволяє виконувати мільйони шумових вибірок, симулювати руслову ерозію та життєдіяльність тисяч біологічних агентів безпосередньо на CPU в реальному часі без мікрофризів ігрового процесу.

1.4 Постановка задачі

Основним завданням цієї роботи є розробка програмного забезпечення для моделювання геологічних процесів та формування ландшафту на сферичній поверхні планети. Програма повинна дозволяти створювати реалістичні карти, враховуючи рух тектонічних плит, зміни клімату та вплив ерозії, з можливістю подальшого експорту отриманих результатів у зручному для користувача форматі.

Процес створення системи базується на використанні сучасних математичних методів та алгоритмів процедурної генерації. При виборі технологій враховано необхідність високої швидкості обробки великої кількості даних, точність фізичних розрахунків та зручність візуалізації складної сферичної геометрії.

У межах роботи поставлено такі завдання:

1. розробити математичний метод поділу сферичної поверхні на рівні частини для зручної побудови сітки;
2. реалізувати алгоритми, що імітують рух тектонічних плит, процеси ерозії та розподіл кліматичних зон;
3. налаштувати систему рендерингу та експорту готових ландшафтних карт у двовимірному вигляді;
4. провести перевірку працездатності розроблених алгоритмів та оцінити їхню ефективність.

Впровадження цієї системи дозволить автоматизувати створення складних віртуальних світів, значно скоротити час на генерацію реалістичних ландшафтів та забезпечити гнучкі інструменти для моделювання природних процесів у різних сферах застосування.

РОЗДІЛ 2
АРХІТЕКТУРА ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ

2.1 Архітектура системи та конвеєр обробки даних на основі технологічного стеку DOTS

Для забезпечення генерації великомасштабних ландшафтів без втрати продуктивності ігрового процесу архітектура системи повністю спроектована за парадигмою проектування, орієнтованого на дані (Data-Oriented Design). Замість традиційної для ігрових рушіїв об'єктно-орієнтованої моделі (Object-Oriented Programming, OOP), де кожна вершина або клітина ландшафту представлена окремим екземпляром класу зі своїми посиланнями у купі (Heap), розроблена система використовує технологічний стек Unity DOTS [11, 17, 18]

Основу системи становить Entity Component System (ECS) та високопродуктивна підмножина мови C# (High-Performance C# або HPC#), що працює виключно з blittable-структурами даних, які зберігаються у безперервних сегментах некерованої пам'яті (Native Containers) [17, 18]. Це повністю усуває накладні витрати на збирання сміття (Garbage Collector), які в класичних іграх викликають випадкові мікрофризи [15, 17].

Проект логічно розподілений на чотири функціональні простори імен, взаємодія між якими жорстко регламентована конвеєром генерації:

- PlanetGenerator.Core: містить базові структури даних (зокрема, blittable-структуру CellData), математичні проєктори та класи просторової індексації.
- PlanetGenerator.Pipeline: відповідає за сортування даних, кліматологічні розрахунки, асинхронне виконання алгоритмів тріангуляції та растровий експорт.
- PlanetGenerator.Tectonics: реалізує структури плит та обчислювальні процеси симуляції тектонічних меж.

- PlanetGenerator.Engine: виступає центральним координаційним ядром, що керує виділенням некерованої пам'яті, життєвим циклом обчислень та передачею даних на етап візуалізації.

Архітектурний опис компонентів системи та їхньої ролі в загальному конвеєрі представлено в табл. А.1.

Для реалізації строгого детермінізму генерації розроблено статичний клас SeedManager [1, 2]. Всі випадкові процеси ініціалізуються унікальним 32-бітним беззнаковим цілим числом, отриманим з текстового рядка masterSeed. Перетворення текстового зерна у первинний хеш виконується за алгоритмом FNV-1a (Fowler-Noll-Vo) [2, 15]:

$$H_0 = 2166136261u \tag{2.1}$$

$$H_k = (H_{k-1} \oplus char_k) \times 16777619u \tag{2.2}$$

де H_0 – початкова константа 32-бітного зміщення алгоритму хешування;

u – прапорець примусової типізації беззнакового 32-бітного цілого числа;

H_k – поточне значення накопиченого хеш-коду на k-му кроці ітерації обробки рядка;

H_{k-1} – значення хеш-коду на попередньому кроці обробки;

$\oplus$ – операція побітового виключного "АБО" (XOR);

$char_k$ – значення коду k-го символу вхідного текстового рядка, одиниць.

Для запобігання кореляційного зв'язку між різними фазами генерації (наприклад, коли розташування тектонічних центрів впливає на випадкові опади) застосовується мультиплікативний хеш Кнута (Knuth's Multiplicative Hashing) для розрахунку ізольованих субзерен [2, 9]:

$$H_{knuth}(V) = V \times 2654435761u \tag{2.3}$$

$$S_{sub} = H_{knuth}(masterSeed \oplus phaseId) \tag{2.4}$$

де H_{knuth}– результат мультиплікативної хеш-функції Кнута від аргументу V;

V – вхідне беззнакове 32-бітне ціле число для хешування;

u – прапорець примусової типізації беззнакового 32-бітного цілого числа;

S_{sub} – результуюче ізольоване субзерно для конкретної фази генерації;

$masterSeed$ – головне цілочисельне випадкове насіння генерації;

$\oplus$ – операція побітового виключного "АБО" (XOR);

$phaseId$ – унікальний цілочисельний ідентифікатор фази розрахункового конвеєра.

Кожне завдання, що працює в багатопотоковому режимі (IJobParallelFor), генерує свій локальний потік випадкових чисел за рахунок змішування субзерна з унікальним індексом робочого потоку (або індексом оброблюваної клітини $workerIndex$) [17, 18]:

$$S_{thread} = H_{knuth}(S_{sub} \oplus workerIndex) \quad (2.5)$$

де S_{thread} – результуючий випадковий потік (насіння) для індивідуального робочого процесу;

H_{knuth} – мультиплікативна хеш-функція Кнута;

S_{sub} – базове субзерно поточної фази обчислювального конвеєра;

$\oplus$ – операція побітового виключного "АБО" (XOR);

$workerIndex$ – цілочисельний порядковий індекс оброблюваної клітини або робочого потоку.

Такий підхід гарантує, що незалежно від кількості ядер процесора та порядку планування потоків операційною системою, згенерований ландшафт буде попіксельно ідентичним на будь-якій апаратній платформі.

У системі реалізовано концепцію подвійної буферизації даних (Double Buffering) [15, 18]. Оскільки під час тектонічних зсувів та ерозії стан сусідніх клітин динамічно змінюється, прямий паралельний запис у той самий масив призвів би до стану перегонів (Race Conditions) та невизначеності результатів. Для вирішення цього кожна ітерація зчитує стабільні дані з масиву InCells (або PlanetCells), а розраховані зміни записує у тимчасовий Native-масив OutHeights (або OutCells), після чого в кінці ітерації покажчики масивів безпечно міняються місцями у головному потоці [18].

2.2 Математичні моделі генерації сферичної поверхні та оптимізації просторової зв'язності

Для побудови математичної моделі планети як геометричного тіла обрано рівномірний розподіл точок за алгоритмом спіралі Фібоначчі [4, 9]. На відміну від кубічних карт (Cube Maps) або геодезичних куполів (Icosahedron Geodesic Dome) [4, 25], які вимагають складного математичного опису переходів між гранями та підрозділення трикутників, сферична решітка Фібоначчі забезпечує практично однакову площу територій, що припадають на кожну вершину, та усуває геометричні сингулярності на полюсах. Водночас, класична двовимірна дискретизація поверхонь за допомогою випадкових або релаксованих полігонів Вороного, яка активно застосовується у растрово-векторних генераторах ландшафтів [23], має обмеження при проектуванні на тривимірну сферу через спотворення площ. Використання розподілу Фібоначчі дозволяє уникнути цих деформацій.

Для загальної кількості точок N тривимірні координати клітини з індексом $i \in \{0, 1, \dots, N-1\}$ розраховуються за формулами:

$$z_i = R \cdot \left(\frac{2i-1}{N-1}\right) \tag{2.6}$$

$$r_i = \sqrt{R^2 - z_i^2} \tag{2.7}$$

$$\phi_i = 2\pi \cdot i \cdot \left(\frac{\sqrt{5}-1}{2}\right) \tag{2.8}$$

$$x_i = r_i \cdot \cos(\phi_i),\ y_i = r_i.\cos(\phi_i) \tag{2.8}$$

де z_i – просторова координата вершини сфери вздовж вертикальної осі Z, м;

R – цільовий геодезичний радіус віртуальної планети, м;

i – поточний індекс розраховуваної клітини в діапазоні від 0 до N-1, одиниць;

N – загальна кількість точок для дискретизації сферичної поверхні, одиниць;

r_i – радіус-проєкція точки на горизонтальну площину XY, м;

ϕ_i – полярний кутовий зсув точки вздовж сферичної спіралі, рад;

x_i – координата вершини сфери вздовж горизонтальної осі X, м;

y_i – координата вершини сфери вздовж горизонтальної осі Y, м [4, 9].

Дробова частина золотого перетину $\frac{\sqrt[2]{5}-1}{2} \approx 0.6180339887$ визначає кутовий зсув спіралі. Оскільки вершини сфери Фібоначчі не мають природної сіткової структури, побудова топологічного графу сусідства є критично важливою для симуляції дифузійних процесів (ерозія, тектонічне заповнення плит, рух вологи). Наївний алгоритм пошуку найближчих сусідів вимагає $O(N^2)$ порівнянь, що при ліміті N = 100,000 клітин призводить до виконання 10^{10} операцій обчислення відстаней [9, 11].

Для оптимізації цієї операції розроблено та реалізовано двокроковий метод просторової індексації та сортування підрахунком (Counting Sort Space Partitioning), який знижує складність алгоритму до лінійної $O(N \cdot K)$, де K – константний коефіцієнт розміру локальної вибірки [9, 18].

На першому кроці (BuildSpatialGridJob) тривимірний простір навколо планети дискретизується у регулярну тривимірну сітку роздільною здатністю $M \times M \times M$ бакетів, де M масштабується динамічно від 15 до 40 залежно від загальної кількості точок для забезпечення щільності у 2–3 клітини на один бакет. Нормалізація та відображення координати вершини $v_{axis} \in$ в індекс осі сітки $g_{axis} \in [0, M-1]$ виконується детерміновано :

$$PosAxisToGrid(v_{axis}) = clamp\left(\left\lfloor \frac{v_{axis}+R}{2R} \cdot M \right\rfloor, 0, M-1\right) \qquad (2.9)$$

де $PosAxisToGrid(v_{axis})$м – розрахований цілочисельний індекс осі просторової сітки бакетів, одиниць;

v_{axis} – дійсна координата вершини вздовж обраної координатної осі (x, y або z), м;

R – базовий геометричний радіус віртуальної планети, м;

M – роздільна здатність однієї координатної осі тривимірної сітки бакетів, одиниць;

$clamp$ – функція обмеження значень у заданому діапазоні [9, 18].

Індекс бакета в одновимірному масиві ID_{bucket} розраховується як лінійний хеш :

$$ID_{bucket} = g_x \cdot M^2 + g_y \cdot M + g_z \quad (2.10)$$

де ID_{bucket} – лінеаризований одновимірний індекс просторового бакета в Native-масиві, одиниць;

g_x – цілочисельний індекс бакета вздовж координатної осі X, одиниць;

M – лінійна роздільна здатність сітки просторового розбиття, одиниць;

g_y – цілочисельний індекс бакета вздовж координатної осі Y, одиниць;

g_z – цілочисельний індекс бакета вздовж координатної осі Z, одиниць [9].

Алгоритм сортування підрахунком виконує швидке групування індексів клітин за бакетами за три проходи без залучення динамічної пам'яті (zero-allocation) [17, 18]:

1. Прохід підрахунку: заповнюється масив BucketOffsets розміром $M^3 + 1$, де кожен елемент містить сумарну кількість клітин, що потрапили у відповідний бакет.
2. Прохід префіксної суми: елементи BucketOffsets накопичуються рекурсивно: BucketOffsets[b] = BucketOffsets[b] + BucketOffsets[b - 1]. Це перетворює кількості на початкові індекси зміщення у відсортованому масиві.
3. Прохід розподілу: використовуючи локальний масив курсорів, що копіює стартові зміщення, алгоритм заповнює результуючий масив BucketCellIndices розміром *N*, де індекси клітин згруповані строго за порядком бакетів.

На другому кроці (BuildNeighborGraphJob) кожна клітина паралельно у власному потоці обчислює свій індекс бакета та сканує суміжні бакети. Для пошуку найближчих суміжних клітин застосовується концентричний пошук по кільцях (Ring Search) радіусом від 1 до 3 бакетів (сканування простору $3 \times 3 \times 3,5 \times 5 \times 5$ та $7 \times 7 \times 7$ навколо вихідної точки).

Для збереження кандидатів використовується локальна структура LocalNeighborList , яка є повністю алокаційно-вільним міні-хепом на основі 6 пар полів (Idx0..Idx5 для індексів та D0..D5 для квадратів відстаней). При знаходженні кандидата з бакета виконується inline-сортування вставками (Insertion Sort) [18]. Якщо квадрат відстані $d^2 = \|Position_{current} - Position_{candidate}\|^2$ менший за поточне найгірше збережене значення D_5, то новий кандидат заміщує його, і елементи зсуваються вгору для збереження висхідного порядку відстаней [18]. Якщо на поточному радіусі пошуку знайдено 6 валідних суміжних точок, відстані до яких менші за фізичну межу бакета, пошук завершується передчасно, що забезпечує значне прискорення.

Для об’єктивної оцінки ефективності запропонованого методу просторової індексації було проведено його теоретичний та практичний аналіз у порівнянні з традиційними підходами, що найчастіше використовуються в задачах пошуку просторової зв'язності. Основними критеріями оцінювання обрано часову складність (як на етапі підготовки даних, так і безпосередньо під час пошуку сусідів), обсяг використання оперативної пам'яті, а також архітектурну сумісність із сучасними технологіями оптимізації виконання коду на рівні процесора. Особлива увага приділялася поведінці алгоритмів в умовах високоінтенсивних обчислень у реальному часі, де критичним фактором є мінімізація навантаження на збирач сміття (Garbage Collector).

Порівняльний аналіз швидкодії та обчислювальної складності розробленого методу просторового групування відносно класичних підходів наведено в таблиці 2.1.

Таблиця 2.1 Аналіз алгоритмів пошуку просторової зв'язності [4, 9, 18]

Параметр порівняння	Наївний пошук найближчих точок	Метод просторової індексації (Розроблений)	Дерево KD-Tree (Класичне)
Обчислювальна складність побудови	O(1) (не потребує побудови)	O(N) (лінійна побудова сітки сортуванням підрахунком)	O(N \log N) (ієрархічне розбиття простору)
Обчислювальна складність пошуку	$O(N^2)$ (повний перебір)	$O(N \cdot K)$ (лінійний локальний пошук у 27 бакетах)	$O(NlogN)$ (обхід дерева)
Витрати пам'яті (RAM)	Немає	$M^3 + 1 + N$ цілих чисел (фіксований розмір)	Високі (вузли дерева, покажчики лівого/правого нащадка)
Сумісність із Burst / SIMD	Середня	Екстремально висока (лінійні NativeArray, blittable-структури)	Низька (ієрархічні посилання, розгалуження коду)
Алокація у купі (Heap)	Zero-allocation	Zero-allocation (некерована Native-пам'ять)	Висока (створення об'єктів вузлів дерева)

2.3 Моделювання тектонічних процесів та двофазної ерозії макрорельєфу

Тектонічний етап моделювання закладає глобальну структуру континентів та океанів [5, 12]. Зерно ініціалізації генерує центри літосферних плит, яким

присвоюється вектор кутової швидкості обертання ω_i навколо індивідуального полюса Ейлера. Швидкість руху довільної точки рельєфу r під впливом тектонічного зсуву плити i визначається векторним добутком за формулою (1.6). На межах зіткнення плит (конвергентні кордони) симулюється тиск орогенезу за формулою (1.7), після чого застосовується нелінійне степеневе перетворення висот :

$$h_{scaled} = pow(h, 1.3) \cdot 4.5 \quad (\text{для } h > 0) \tag{2.11}$$

де h_{scaled} – кінцева масштабована висота клітини рельєфу після тектонічної корекції, м;

h – первинна розрахована висота клітини, отримана під час симуляції зсувів літосферних плит, м [5, 12].

Отримане первинне поле висот піддається двофазній ерозійній обробці для формування природних річкових долин, осипів та згладжених форм ландшафту.

Фаза термальної ерозії (ThermalErosionJob) описує процес сухого гравітаційного осипання нестабільного ґрунту [2, 7]. Кожна клітина і оцінює перепад висот зі своїми 6 сусідами. Якщо перепад висот $\Delta h_k = h_{current} - h_k$ перевищує критичний поріг кута укосу (Talus Threshold) θ_{talus} , порода осипається вниз під дією сили тяжіння. Об'єм матеріалу, який переноситься на нижчі ділянки, розраховується за формулою :

$$h_{move} = \left(\frac{\sum_{k \in S_{lower}} \Delta h_k}{N_{lower}} \right) \cdot K_{erosion} \tag{2.12}$$

де h_{move} – сумарний об'єм (висота) перенесеної породи на нижчі суміжні ділянки за одну ітерацію, м;

S_{lower} – множина сусідів, для яких різниця висот перевищила поріг;

Δh_k – перепад висоти між поточною клітиною та її k-м сусідом, м;

N_{lower} – кількість таких нижчих сусідів;

$K_{erosion}$ – коефіцієнт швидкості ерозії. Оновлені висоти записуються у вихідний буфер [7].

Фаза гідравлічної ерозії (HydraulicErosionJob) реалізує складну клітинно-автоматну модель руху води, розчинення породи, транспортування осаду та

депонування грунту. Вона виконується послідовно у межах одного потоку для гарантування фізичного балансу мас вологи та сухого залишку.

Фізичні параметри та коефіцієнти, що керують симуляцією гідравлічного вивітрювання, наведено в таблиці 2.3.

Таблиця 2.3 Фізичні константи моделі гідравлічної ерозії [5-7, 12]

Назва параметра	Символ	Типове значення	Фізичний зміст та вплив на геоморфологію ландшафту
Швидкість опадів	$R_{rainfall}$	0.012	Об'єм води, що додається на кожну сухопутну клітину за один крок
Розчинність породи	$K_{solubility}$	0.15	Інтенсивність переходу твердого ґрунту у зважений осад у воді
Коефіцієнт осадження	$K_{deposit}$	0.10	Швидкість випадання зваженого осаду на рельєф при уповільненні потоку
Випаровування	$K_{evaporation}$	0.05	Коефіцієнт зменшення об'єму води на клітинах за крок часу
Рівень моря	$SeaLevel$	0.0	Абсолютна висота дзеркала води, що розділяє суходіл та океан
Коефіцієнт міцності	κ	1.0 або 1.4	Опірність породи руйнуванню. Дорівнює 1.4 для скельних вершин (h > 1.2)
Таймер згасання	γ	від 1.0 до 0.0	Коефіцієнт захисту піків від стирання, що лінійно спадає після 200-ї ітерації

Гідравлічна симуляція на кожній ітерації проходить такі послідовні математичні етапи :

1. Конденсація вологи: для всіх клітин, у яких висота $h_i > SeaLevel$, рівень води у масиві збільшується: $w_i \leftarrow w_i + R_{rainfall}$ [7].
2. Визначення градієнта стоку: алгоритм обходить 6 сусідів клітини i та знаходить сусіда j з мінімальною висотою h_j [4,13].
3. Ерозійна фаза: якщо знайдено сусіда з меншою висотою та ухил $S = h_i - h_j > 0$, розраховується гранична утримувальна ємність води для транспортування осаду (Sediment Capacity) [7, 13]:

$$C_s = max(0.001,\ w_i \cdot S \cdot 0.2)$$

Якщо поточна кількість зваженого осаду в клітині $sediment_i$ менша за ємність C_s, вода розчиняє ґрунт під собою, зменшуючи його висоту на величину $\delta_{erosion}$ [7]:

$$\delta_{erosion} = min\left((C_s - sediment_i) \cdot K_{solubility} \cdot \kappa \cdot \gamma,\ 0.5 \cdot S\right)$$

$$h_i \leftarrow h_i - \delta_{erosion}, \qquad sediment_i \leftarrow sediment_i + \delta_{erosion}$$

Обмеження ерозії величиною $0.5 \cdot S$ запобігає виникненню зворотного ухилу та забезпечує числову стабільність різницевої схеми.

4. Фаза депонування: якщо потік уповільнюється і кількість осаду перевищує ємність ($sediment_i \geq C_s$), надлишковий матеріал випадає назад, нарощуючи рельєф на величину $\delta_{deposit}$:

$$\delta_{deposit} = (sediment_i - C_s) \cdot K_{deposit} \qquad (2.13)$$

$$h_i \leftarrow h_i + \delta_{deposit}, \qquad sediment_i \leftarrow sediment_i - \delta_{deposit} \qquad (2.14)$$

де $\delta_{deposit}$ – товщина шару осадових порід, що випадають на рельєф клітини і, м;

$sediment_i$ – концентрація зваженого осаду в потоці води на клітині і перед осадженням, м;

C_s – гранична утримувальна ємність потоку водної маси щодо осаду, м;

$K_{deposit}$ – коефіцієнт швидкості випадання (депонування) осаду на рельєф, безвимірна величина;

h_i – абсолютна висота поверхні і-ї клітини рельєфу, м [7, 13].

5. Адвективний перенос вологи та осаду: половина наявної води та зваженого осаду переноситься в низину до сусіда j :

$$w_j \leftarrow w_j + 0.5 \cdot w_i, \qquad w_i \leftarrow 0.5 \cdot w_i \tag{2.15}$$

$$sediment_j \leftarrow sediment_j + 0.5 \cdot sediment_i, \quad sediment_i \leftarrow 0.5 \cdot sediment_i \tag{2.16}$$

де w_j – рівень накопиченої води на сусідній нижчій клітині j після адвективного перенесення, м;

w_j – рівень накопиченої води на поточній клітині і перед перенесенням, м;

$sediment_j$ – об'єм зваженого осаду на сусідній клітині j після адвективного перенесення, м;

$sediment_i$ – об'єм зваженого осаду на поточній клітині і перед перенесенням, м [7].

Якщо сусідня клітина j знаходиться нижче рівня моря ($h_j \leq SeaLevel$), весь отриманий осад моментально випадає на дно океану у вигляді осадових порід: $h_j \leftarrow h_j + sediment_j$, а вміст зваженого осаду в потоці обнуляється: $sediment_j \leftarrow 0$.

6. Випаровування вологи: вода, що залишилася на суші, зменшується під дією сонця:

$$w_i \leftarrow w_i \cdot \left(1 - K_{evaporation}\right) \tag{2.17}$$

де w_i – рівень вільної вологи на поверхні клітини і після кроку випаровування, м;

$K_{evaporation}$ – коефіцієнт швидкості випаровування води pod дією температури, безвимірна величина [7, 13].

Накопичена волога у вигляді залишків води w_i конвертується у показник вологості клітини для розрахунку майбутньої карти екосистем.

2.4 Кліматична симуляція та класифікація біомів за матрицею Вайттекера

Розподіл рослинного покриву та природних зон спирається на принципи екологічного детермінізму, де тип і щільність рослинності визначаються локальними фізичними та кліматичними параметрами середовища [1, 15]. Кліматична модель (ComputeClimateAndBiomesJob) розраховує середньорічну температуру T та вологість M для кожної клітини паралельно на основі географічної широти та висоти над рівнем моря.

Вплив широтної зональності базується на тому, що кількість сонячної радіації лінійно зменшується від екватора до полюсів. Висотне похолодання імітує атмосферний градієнт температур (Lapse Rate), згідно з яким температура падає на фіксовану величину на кожну одиницю підйому рельєфу вгору.

Математичний опис кліматичного профілю планети представлено формулами :

- Для океанічних клітин ($h \leq SeaLevel$): температура повітря над водою згладжується високою теплоємністю водних мас та інтерполюється між $25.0℃$ (екватор, $y = 0$) та $-5.0℃$ (полюси, $|y| = R$) :

$$T = lerp\left(25.0, -5.0, \frac{|y|}{R}\right) \tag{2.18}$$

де T – локальна температура над поверхнею водних мас, ℃;

y – відстань по вертикальній осі обертання планети від екватора до клітини, м;

R – загальний радіус планетарного тіла, м;

$lerp$ – оператор лінійної інтерполяції [7, 15].

- Для континентальних клітин ($h > SeaLevel$): базова широтна температура інтерполюється між $32.0℃$ (екватор) та $-15.0℃$ (полюси) і додатково коригується висотним охолодженням із коефіцієнтом градієнта $\lambda = 12.0$:

$$T = lerp\left(32.0,\ -15.0, \frac{|y|}{R}\right) - 12.0 \cdot h \qquad (2.19)$$

де T – локальна температура повітря над сушею з урахуванням висотної поясності, °C.

Вологість клітини M безпосередньо копіює об'єм води, накопичений під час етапу гідравлічної ерозії: $M = w_i$.

Процес екологічного зонування у масштабних симуляціях вимагає попіксельного або пографного аналізу вхідних карт висот, температур та вологості. Традиційний підхід до розв'язання цієї задачі базується на каскадах умовних операторів if-else, що створює глибоку ієрархію розгалужень. В архітектурах сучасних процесорів такий підхід викликає часті збої в роботі блоку передбачення переходів (Branch Prediction Unit), оскільки кліматичні параметри сусідніх ділянок можуть суттєво відрізнятися. Це призводить до регулярного скидання конвеєра інструкцій, втрати тактових частот та нівелює оптимізаційні можливості Burst-компілятора, який не здатний виконати ефективну векторизацію (SIMD) для коду з великою кількістю логічних розгалужень.

Для усунення цього негативного ефекту та забезпечення максимальної пропускної здатності обчислювального потоку, логіку класифікації було перероблено на засадах data-oriented дизайну. Замість динамічного обходу дерева рішень через умовні переходи, алгоритм використовує математичні трансформації та побітові операції для обчислення результуючого індексу біома на основі нормалізованих значень факторів середовища.

Для класифікації екосистем застосовується адаптована під вимоги комп'ютерної симуляції модель біомів Роберта Вайттекера. Традиційне використання складних вкладених логічних умов if-else (Branching) вкрай негативно позначається на продуктивності Burst-компілятора, оскільки процесор змушений постійно скидати конвеєр інструкцій через помилки передбачення переходів (Branch Misprediction). Для усунення цього ефекту логіку класифікації спроектовано у вигляді branch-free дерева рішень , що представлено в табл. 2.4.

Таблиця 2.4 Симуляційна модель біомів Роберта Вайттекера [15-17]

Критерій середовища	Межі температури (T,∘C)	Межі вологості (M)	Індекс	Отриманий біоценоз
Океан (h ≤ SeaLevel)	T < 0.0	Будь-яка	0	Frozen Ocean / Glaciers
Океан (h ≤SeaLevel)	T > 20.0	Будь-яка	1	Tropical Shelf / Coral Reefs
Океан (h ≤SeaLevel)	0.0 ≤T ≤20.0	Будь-яка	2	Open Ocean
Суходіл (h > SeaLevel)	T < -5.0	Будь-яка	3	Tundra
Суходіл (h > SeaLevel)	-5.0 ≤T < 3.0	M > 0.4	4	Taiga / Boreal Forest
Суходіл (h > SeaLevel)	-5.0 ≤T < 3.0	M ≤0.4	5	Cold Desert
Суходіл (h > SeaLevel)	3.0 ≤T < 18.0	M < 0.2	6	Shrub Desert
Суходіл (h > SeaLevel)	3.0 ≤T < 18.0	0.2 ≤ M < 0.6	7	Steppe / Prairie
Суходіл (h > SeaLevel)	3.0 ≤ T < 18.0	M ≥0.6	8	Temperate Deciduous Forest
Суходіл (h > SeaLevel)	T ≥18.0	M < 0.15	9	Hot Desert / Sahara
Суходіл (h > SeaLevel)	T ≥ 18.0	0.15 ≤ M < 0.5	10	Savanna
Суходіл (h > SeaLevel)	T ≥ 18.0	M ≥ 0.5	11	Tropical Rainforest

Ця модель дозволяє детерміновано заповнити кліматичні зони планети, створюючи умови для високогірної тундри чи альпійських лук на вершинах гір, навіть якщо самі гори розташовані в екваторіальних широтах, за рахунок урахування висотного температурного градієнта[15, 21].

2.5 Математичний апарат стереографічного проєктування та асинхронної тріангуляції Делоне

Для генерації тривимірного полігонального мешу планети необхідно об'єднати ізольовані хмари точок Фібоначчі в зв'язну трикутну сітку [4, 25]. Пряма тріангуляція Делоне на сфері є складною математичною задачею, оскільки вимагає обчислення описаних сферичних кіл на криволінійній поверхні та призводить до топологічних помилок біля полярних кордонів.

Для розв'язання цієї проблеми використано конформні властивості стереографічної проєкції [9, 25]. Стереографічна проєкція відображає поверхню сфери на площину $Z = 0$ з фокусом на Південному полюсі. Завдяки збереженню кутів та конформності відображення, описане коло будь-якого трикутника на сфері переходить у строго описане коло трикутника на площині. Це дозволяє виконати класичну планарну тріангуляцію Делоне на двовимірній площині, після чого отриманий індексний буфер трикутників буде топологічно строгим і безпомилковим для оригінальних тривимірних вершин на сфері.

Процес тріангуляції спроектовано як асинхронну операцію, яка делегується окремому фоновому потоку за допомогою системи завдань Task.Run. Це гарантує повну відсутність падіння частоти кадрів (FPS) на ігровому потоці під час генерації.

Представлений метод побудови сферичної тріангуляції реалізується через послідовність трьох математичних кроків. На першому етапі здійснюється пряма стереографічна проєкція: кожна вершина $p = (x, y, z)$ на одиничній сфері (де $x^2 + y^2 + z^2 = 1$) переноситься на площину згідно з формулами:

$$X = x/(1 - z)$$

$$Y = y/(1 - z)$$

Для забезпечення чисельної стійкості та уникнення ділення на нуль у точці Південного полюса ($z = 1$), знаменник обмежується функцією $denominator = max(1.0 - z,\ 10^{-6})$ [18].

Другий етап базується на планарній тріангуляції Боєра-Ватсона. Спочатку ініціалізується «супер-трикутник» $A_{super}B_{super}C_{super}$, що охоплює межі спроєктованих координат, розрахований на основі габаритного контейнера зі стороною [4, 9]:

$$Width_{box} = max(X_{max} - X_{min},\ Y_{max} - Y_{min})$$

У процесі ітеративного додавання кожної нової точки (X_k, Y_k) алгоритм ідентифікує трикутники, в описане коло яких потрапляє точка, видаляє їх та формує багатокутну порожнину (каверну). Вершини цієї каверни з'єднуються з новою точкою, створюючи оновлену мережу. По завершенні обробки всіх точок вершини супер-трикутника та пов’язані з ним елементи видаляються.

На третьому етапі проводиться зворотна стереографічна проєкція для перенесення індексного списку зв’язності на вихідні сферичні координати [9, 25]. Відновлення 3D-позицій здійснюється за формулами:

$$x = 2X/(1 + X^2 + Y^2)$$

$$y = 2Y/(1 + X^2 + Y^2) \text{ та } z = (-1 + X^2 + Y^2)/(1 + X^2 + Y^2)$$

На фінальній стадії отримані вершини нормалізуються та масштабуються з урахуванням рельєфу за рівнянням:

$$Position_{final} = n_{xyz} \cdot (R + Height_i) \qquad (2.20)$$

де $Position_{final}$– кінцеві тривимірні координати вершини полігонального мешу планети, м;

n_{xyz} – нормалізований вектор напрямку з центру сфери до вихідної точки, безвимірна величина;

R – базовий сферичний радіус віртуальної планети, м;

$Height_i$ – розрахована абсолютна висота рельєфу і-ї клітини, м [5, 6, 9].

2.6 Процедурна побудова та двовимірна фільтрація растрових карт

Для експорту планетарної геометрії у двовимірні растрові текстури (карти висот, біомів, температур) розроблено асинхронний модуль PlanetMapExporter. Процес побудови растрового зображення розміром $W \times H$ виконується за алгоритмом зворотного картографічного проєктування : кожному пікселю з координатами (x_{px}, y_{px}) ставиться у відповідність сферичний вектор напрямку $D = (d_x, d_y, d_z)$ у рівнопроміжній циліндричній проєкції. Кути довготи $\lambda\epsilon[-\pi, \pi]$ та широти $\phi \in [-\pi/2, \pi/2]$ обчислюються як :

$$\lambda = \left(\frac{x_{px}}{W} \cdot 2 - 1\right) \cdot \pi, \qquad \phi = \left(\frac{y_{px}}{H} - 0.5\right) \cdot \pi \tag{2.21}$$

де λ – розрахована довгота для заданого пікселя карти, рад;

x_{px} – горизонтальна координата пікселя на двовимірній текстурі, одиниць;

W – загальна ширина вихідного растрового зображення (текстури), пікселів;

ϕ – розрахована широта для заданого пікселя карти, рад;

y_{px} – вертикальна координата пікселя на двовимірній текстурі, одиниць;

H – загальна висота вихідного растрового зображення, пікселів [4, 15].

Вектор напрямку визначається тригонометричним відображенням :

$$D = \left(cos(\phi)sin(\lambda),\ sin(\phi),\ cos(\phi)cos(\lambda)\right) \tag{2.22}$$

де D – одиничний вектор просторового напрямку на сфері для шуканого пікселя, безвимірна величина;

ϕ – розрахована широта пікселя на карті, рад;

λ – розрахована довгота пікселя на карті, рад [4, 15].

Для розрахунку точного значення кольору пікселя на основі нерегулярної хмари точок клітин застосовується метод інтерполяції зворотних зважених відстаней (Inverse Distance Weighting, IDW) по 9 найближчих сусідах. Пошук сусідів на сфері оптимізовано за допомогою двовимірного широтного

просторового індексу SpatialGrid роздільною здатністю 64. Формула інтерполяції кольору C має вигляд :

$$C = \frac{\sum_{k=1}^{9} w_k \cdot C_k}{\sum_{k=1}^{9} w_k} \tag{2.23}$$

де C – результуючий інтерпольований колір пікселя растрової текстури, безвимірна величина (RGB-вектор);

w_k – математична вага впливу суміжної клітини Вороного з порядковим індексом k, безвимірна величина;

C_k – вихідний колір суміжної клітини Вороного k, безвимірна величина (RGB-вектор) [4].

Вага сусіда w_k спадає обернено пропорційно відстані у степені $p = 1.5$ для забезпечення плавних переходів кольорів без різких граней :

$$w_k = \frac{1}{(1.0001 - D \cdot N_k)^{1.5}} \tag{2.24}$$

де w_k – розрахована математична вага впливу сусіда з індексом k, безвимірна величина;

D – одиничний тривимірний вектор напрямку на сфері для пікселя карти, безвимірна величина;

N_k – нормалізований радіус-вектор розташування k-го сусіда на сфері, безвимірна величина;

$D \cdot N_k$ – скалярний добуток векторів, який відображає косинус кутової відстані на сфері, безвимірна величина [15].

Якщо кутова відстань прямує до нуля ($D \cdot N_k > 1 - 10^{-7}$), вага стає сингулярною, і пікселю присвоюється чистий колір клітини C_k.

Для згладжування візуального шуму на згенерованій текстурі застосовується алгоритм роздільного розмиття Гаусса (Separable Gaussian Blur) з радіусом $r = 6$ пікселів. Математична особливість роздільної фільтрації полягає у послідовному виконанні одновимірних згорток спочатку по рядках (горизонтальний прохід), а потім по стовпчиках (вертикальний прохід).

При цьому на краях растрового зображення застосовуються різні граничні умови, зумовлені геометрією сферичного розгортки :

- По довготі (вісь X): оскільки карта циліндрично замикається сама на себе, реалізовано безшовний круговий перенос індексів (Wrap-Around) за допомогою залишку від ділення :

$$x' = (x + k - r + W)\ mod W$$

- По широті (вісь Y): на полюсах текстура замикатися не може, тому застосовується примусове насичення індексів (Clamping) на крайніх пікселях :

$$y' = clamp(y + k - r,\ 0,\ H - 1)$$

де $k \in [0,2r]$ – індекс згортки ядра фільтра. Одновимірні ваги ядра фільтра розраховуються за нормальним розподілом із середньоквадратичним відхиленням $\sigma = r / 2 = 3.0$.

2.7 Опис та аналіз алгоритмів процедурного моделювання сферичного ландшафту

Для детального розуміння послідовності обчислень, логіки взаємодії програмних модулів та оптимізації процесів нижче наведено систематизований аналіз та опис розроблених алгоритмів. Опис побудовано на основі наданих структурних схем конвеєра генерації, алгоритму просторової індексації та ітераційної моделі гідравлічного вивітрювання. Представлений матеріал інтегрує фундаментальні принципи обчислювальної геометрії, просторового аналізу та фізико-географічного моделювання з урахуванням вимог високопродуктивного технологічного стеку Unity DOTS (Data-Oriented Technology Stack).

Центральним драйвером розробленої системи є алгоритм ExecuteFullPipeline (див. рис. 2.1), який керує життєвим циклом обчислень від первинного введення насіння генерації до побудови фінального тривимірного полігонального мешу планети. Логіка побудови цього конвеєра підпорядкована лінійній структурі, де кожна наступна фаза детерміновано спирається на

завершені та стабілізовані дані попередніх етапів. Такий підхід повністю ліквідує накладні витрати на динамічне коригування взаємозалежних систем (наприклад, запобігає вегетації в руслах річок або на нестабільних схилах).

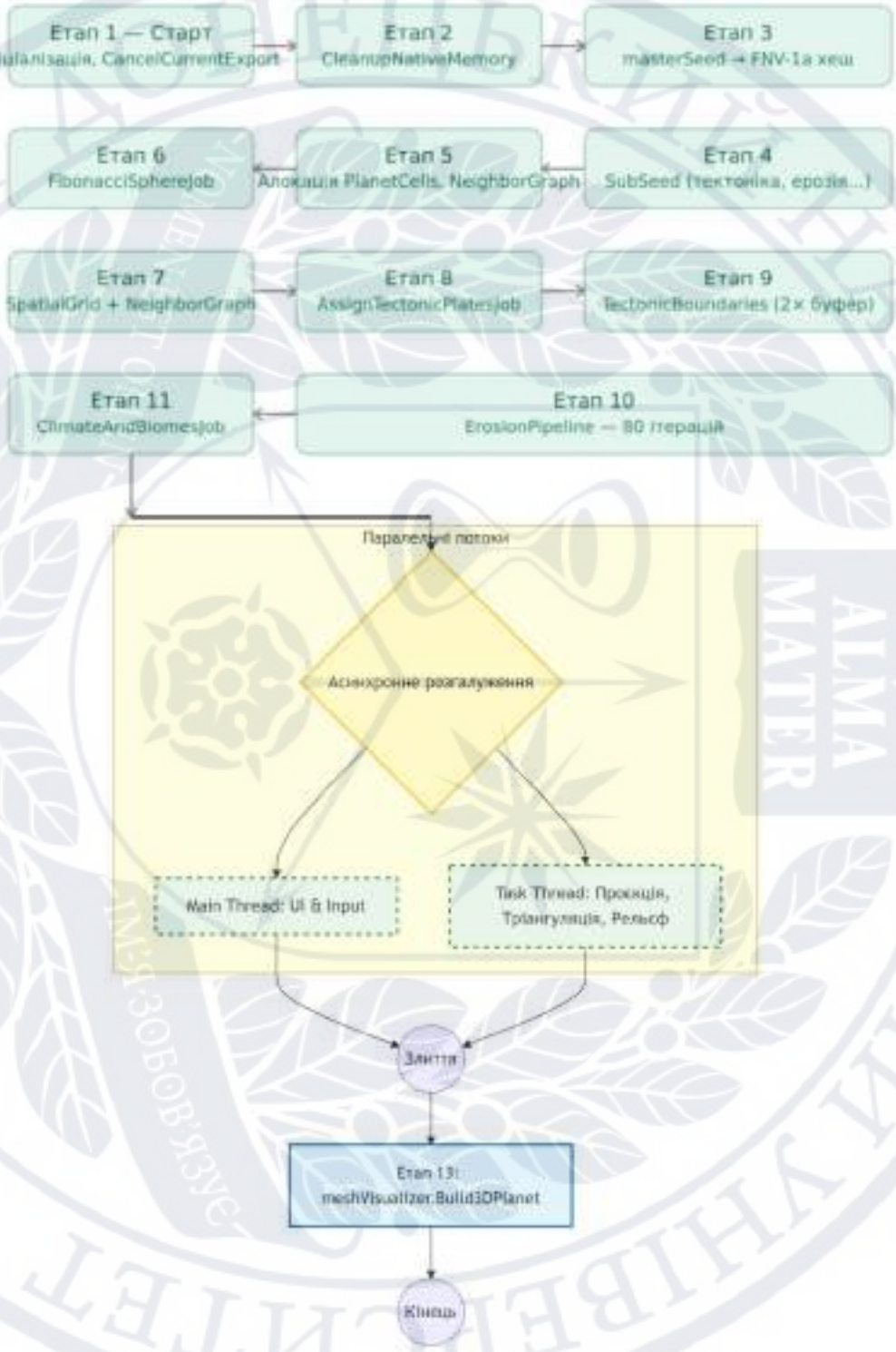

Рисунок 2.1 – Блок-схема алгоритму ExecuteFullPipeline

Особливу увагу в архітектурі конвеєра приділено асинхронному розгалуженню після Етапу 11. Побудова зв'язного полігонального мешу з ізольованої хмари точок Фібоначчі вимагає виконання тріангуляції Делоне. Оскільки пряма тріангуляція на сфері призводить до топологічних помилок біля полярних кордонів, у системі застосовано математичний апарат конформної стереографічної проєкції.

Точки проєктуються на площину $Z = 0$, де виконується класичний планарний алгоритм тріангуляції Боєра-Ватсона, після чого отриманий індексний буфер трикутників повертається на сферу. Завдяки делегуванню цих операцій фоновому потоку Task Thread через систему Task.Run, ресурсомісткі геометричні розрахунки не впливають на плавність рендерингу інтерфейсу.

Для симуляції дифузійних процесів, таких як перетікання води при ерозії чи зсув літосферних плит, кожна клітина сфери повинна мати інформацію про своїх найближчих географічних сусідів. Пошук суміжних точок за наївним алгоритмом вимагає обчислення відстаней між усіма парами вершин, що робить симуляцію неможливою для великої кількості клітин N.

Для подолання цієї обчислювальної перешкоди розроблено двокроковий алгоритм PlanetGraphBuilder (рис. 2.3 та 2.4), який реалізує концепцію просторового розбиття (Spatial Partitioning) на основі алгоритму сортування підрахунком (Counting Sort).

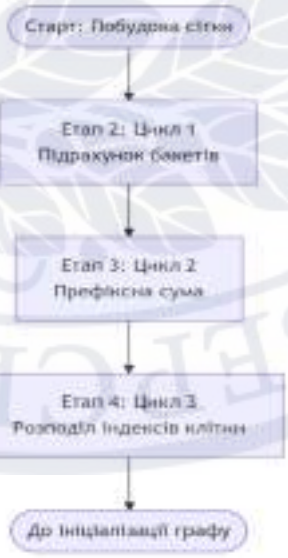

Рисунок 2.2 - Перший крок алгоритму PlanetGraphBuilder

Рисунок 2.3 Другий крок алгоритму PlanetGraphBuilder

Другим етапом обробки даних є побудова топологічного графу за допомогою завдання BuildNeighborGraphJob. Цей процес спрямований на визначення просторових зв’язків між точками, що є критично важливим для подальших фізичних симуляцій. Алгоритм розпочинається з визначення базового

бакета, де для кожної клітини з координатами i розраховується індекс просторової сітки, що дозволяє локалізувати пошук сусідів у межах мінімально необхідного геометричного об'єму та уникнути глобального перебору.

Для забезпечення ефективного пошуку навіть за умов нерівномірного розподілу точок, алгоритм застосовує ітеративне розширення радіуса від $3 \times 3 \times 3$ до $7 \times 7 \times 7$ бакетів. Сканування суміжних комірок супроводжується логічним фільтром, який перевіряє, чи не був певний бакет уже просканований на попередніх кроках. Така умова радіуса дозволяє повністю виключити дублювання обчислень відстаней для вже перевірених точок, значно підвищуючи швидкість роботи системи.

Безпосередній розрахунок відстаней базується на обчисленні квадрата евклідової відстані $d^2 = \|P_{\backslash textcurr} - P_{\backslash textcand}$, що дозволяє уникнути ресурсомісткої операції взяття квадратного кореня. Для впорядкування знайдених кандидатів використовується сортування вставками (Insertion Sort) у локальному стеку фіксованого розміру. Це рішення дозволяє повністю відмовитися від динамічного виділення пам'яті (heap allocations), що є важливою передумовою для ефективної роботи Burst-компілятора.

Процес сканування включає механізм передчасного завершення, який спрацьовує при накопиченні шести найближчих точок, що відповідають встановленим фізичним критеріям. Як тільки ця умова виконується, цикл пошуку переривається, заощаджуючи обчислювальні ресурси. Після завершення пошуку для конкретної точки, результати у вигляді індексів сусідів фіксуються у глобальному плоскому масиві топологічного графу за зміщенням $i \cdot 6$.

Завдяки описаній архітектурі, алгоритм демонструє високу продуктивність та стабільність. Використання підходу без динамічних алокацій пам'яті дозволяє генерувати максимально оптимізований машинний код, що забезпечує швидку та надійну роботу геометричних запитів, критично необхідну для складних фізичних конвеєрів.

Процеси гідравлічної ерозії відіграють вирішальну роль у перетворенні первинного штучного фрактального рельєфу на реалістичний природний

ландшафт. Впроваджений алгоритм симуляції спирається на фізичні моделі водної ерозії та депонування осадових порід, адаптовані для роботи у вигляді клітинного автомата. Моделювання ерозії (рис. 2.4) на кожній ітерації є послідовним процесом, що складається з трьох логічних фаз: ініціалізації з опадами, руху вологи із седиментацією, а також випаровування з переходом до наступного кроку.

Рисунок 2.4 - Фаза ініціалізації та опадів

На старті симуляції ерозії алгоритм виконує ініціалізацію робочих змінних та розраховує мультиплікатор згасання. Цей коефіцієнт поступово зменшує ерозійну здатність води на пізніх етапах, що необхідно для збереження стрімких високогірних піків від повного руйнування. Після цього запускається покіловий цикл, у якому для кожної клітини рельєфу перевіряється геометрична умова її

висоти відносно базового рівня моря. Якщо висота клітини більша за рівень моря, система спрямовує хід виконання за гілкою «Так», імітуючи випадіння опадів – у такому разі рівень вологи в клітині збільшується на фіксовану величину опадів. Якщо ж клітина знаходиться нижче або на рівні моря, спрацьовує гілка «Ні», і алгоритм просто пропускає її, оскільки моделювання опадів та ерозії безпосередньо всередині глибоких водних басейнів є фізично недоцільним.

Наступний етап присвячений безпосередньому руху води та перенесенню осадових порід між клітинами, див рис. 2.5 . Розрахунок починається з перевірки умови, чи має поточна клітина воду і чи не є вона океаном. Якщо води немає або клітина затоплена, потік завершується. У разі виконання умови алгоритм шукає серед сусідніх клітин ту, яка має найнижчу висоту, визначаючи напрямок дії сили тяжіння та вектор найкоротшого спуску. Далі порівнюються висоти поточної та знайденої найнижчої сусідньої клітини.

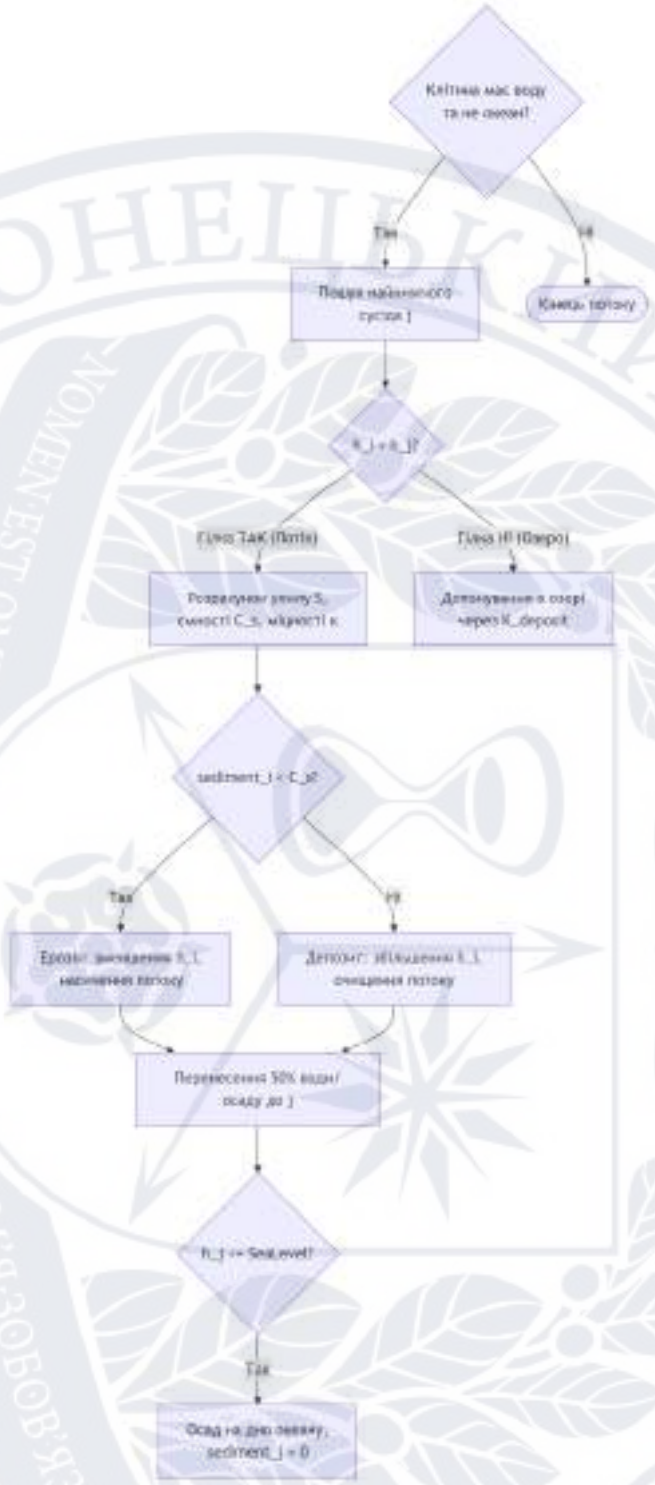

Рисунок 2.5 - Фаза руху вологи та седиментації

Якщо висота поточної клітини не перевищує висоту сусіда, потік потрапляє в локальну западину, утворюючи озеро, де надлишковий осад накопичується й депонується з урахуванням коефіцієнта депозиту, що призводить до вирівнювання ландшафту та заповнення ям. Якщо ж поточна клітина вища, рух

продовжується за гілкою потоку. На основі різниці висот розраховується ухил рельєфу, який разом із поточною кількістю води визначає граничну утримувальну ємність потоку – тобто максимальний об'єм ґрунту, який вода здатна транспортувати.

Наступним кроком алгоритм порівнює фактичний вміст осаду у воді з розрахованою граничною ємністю. Якщо поточний рівень осаду менший за ємність, запускається процес ерозії: недонасичений потік вимиває тверду породу, зменшуючи висоту поточної клітини та насичуючи воду ґрунтом з урахуванням розчинення та мультиплікатора згасання. Якщо ж осад перевищує ємність, запускається процес депозиту: надлишкові зважені частинки випадають, збільшуючи висоту поточної клітини та очищуючи водний потік.

Незалежно від того, відбулася ерозія чи депонування, алгоритм виконує адвективний перенос, фізично переміщуючи половину об'єму води та зваженого осаду вниз за рельєфом до найнижчої сусідньої клітини. Наприкінці фази перевіряється, чи досяг потік базового рівня моря через сусідню клітину. Якщо сусідня клітина є океанічним стоком, весь перенесений осад моментально осідає на морське дно, формуючи підводний рельєф, а вміст осаду в самому потоці обнуляється.

Після завершення ітеративного переміщення вологи та зважених речовин по всій поверхні, система переходить до заключної фази, див. рис. 2.6. Спочатку виконується збір та отримання оновлених фізичних даних про стан поверхні після латерального руху води. Потім запускається процес випаровування під дією умовного сонячного випромінювання, під час якого поточний рівень води в кожній клітині зменшується на величину, що визначається коефіцієнтом випаровування. Залишок води конвертується в показник локальної вологості ґрунту, який згодом дозволяє точно визначати межі майбутньої вегетації та рослинних біомів. Клітинний автомат завершує роботу на поточному кроці й виконує логічний перехід до наступного етапу загальної симуляції ландшафту.

Рисунок 2.6 - Фаза випаровування та переходу

Сумарно у другому розділі обґрунтовано архітектуру високопродуктивної системи, де перехід від об'єктно-орієнтованого підходу до DOTS-орієнтованої на дані моделі став фундаментом для усунення критичних обчислювальних затримок. Впровадження лінійного методу просторової індексації та асинхронної тріангуляції на базі стереографічної проєкції дозволило досягти стабільної продуктивності навіть при роботі з хмарами точок високої щільності, перетворивши ресурсомісткі геометричні задачі на ефективні фонові процеси.

Реалізовані методи ерозії, адаптовані через клітинні автомати, довели свою ефективність у збереженні морфологічних особливостей рельєфу, а інтеграція фізико-кліматичного зонування з SIMD-оптимізованою класифікацією біомів забезпечила математичну точність симуляції екосистем. Запропонована підсистема растрового експорту, завдяки використанню IDW-інтерполяції та полярної фільтрації, завершує цикл обробки, гарантуючи створення безшовних візуальних даних [6], що підтверджує можливість масштабованої генерації складних сферичних ландшафтів у реальному часі.

Отриманий комплекс математичних моделей, алгоритмів та структур утворює надійну теоретичну та прикладну основу для їхньої програмної реалізації та тестування на наступних етапах розробки.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ, НАЛАШТУВАННЯ СЦЕНИ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ СИСТЕМИ ГЕНЕРАЦІЇ

3.1 Структура програмного продукту та архітектура модулів

Фінальний розділ присвячено практичній реалізації та підсумковому аналізу розробленого програмного комплексу, у якому було втілено всі теоретичні засади та математичні моделі, обґрунтовані в попередніх частинах роботи. Програмний комплекс побудовано за модульним принципом: центральним елементом виступає структура комірки, а всі функціональні блоки розділено на окремі конвеєри генерації геометрії, симуляції тектонічних процесів, розрахунку клімату й ерозії, а також підсистеми візуалізації та експорту даних, що зробило систему гнучкою, масштабованою та зручною для подальшого розвитку.

3.1.1 Базові структури даних

Для забезпечення максимальної продуктивності та кеш-локальності під час паралельних обчислень дані осередків планети спроєктовано як монолітні "чисті" структури (Blittable types), що не містять посилальних типів (reference types). Це дозволяє уникнути фрагментації оперативної пам'яті та затримок збірника сміття (Garbage Collector).

Головною структурою є CellData, яка об'єднує геометричні, геологічні, кліматичні та екологічні параметри кожного окремого осередку Вороного (тайла):

```
namespace PlanetGenerator.Core
{
    public struct CellData
    {
        // Ідентифікатори та геометрія
        public int Index;
```

```
        public float3 Position; // Положення на одиничній або масштабованій сфері

        // Рельєф та гідрологія
        public float Height;
        public float Moisture;
        public float Temperature;
        public byte BiomeIndex; // Індекс біому за таблицею Вайттекера

        // Тектонічні параметри плит
        public int TectonicPlateId;
        public float3 TectonicVelocity; // Вектор кутової швидкості плити
        public float CrustThickness;          // Товщина кори (континентальна/океанічна)

        // Екосистема (Біомаса трофічних рівнів моделі Лотки-Вольтерри)
        public float FloraBiomass;     // Продуценти (рослинність)
        public float HerbivoreCount;     // Первинні консументи (травоїдні)
        public float CarnivoreCount;     // Вторинні консументи (хижаки)
    }
}
```

3.1.2 Опис супутніх допоміжних модулів

1. SeedManager.cs (Керування псевдовипадковими послідовностями): Забезпечує детермінізм генерації. Будь-який стринговий сид (текст) конвертується в 32-бітне число за алгоритмом FNV-1a. Далі за допомогою мультиплікативного методу хешування Кнута генеруються унікальні

дочірні сиди для окремих потоків виконання (IJobParallelFor), що виключає появу однакових патернів шуму на різних ядрах процесора.

2. StereographicProjector.cs (Просторова проєкція): Містить Burst-оптимізовані паралельні завдання StereographicProjectionJob та InverseStereographicProjectionJob. Вони здійснюють пряме та зворотне відображення точок тривимірної сфери на площину $Z = 0$ (і навпаки). Це необхідно для виконання швидкої двовимірної тріангуляції Делоне або растрової фільтрації без спотворення геометрії на полюсах.
3. PlanetGraphBuilder.cs (Побудова топологічного графа): Використовує просторову рівномірну 3D-сітку (BuildSpatialGridJob) для оптимізації пошуку суміжних тайлів. Завдяки розподілу клітин за просторовими кошиками (buckets), складність пошуку 6 найближчих суміжних осередків для кожної точки знижується з $O(N^2)$ до $O(N \cdot K)$, де K – константна кількість перевірок у сусідніх кошиках.

3.2 Практична реалізація конвеєра генерації

Основним диспетчером обчислень є клас PlanetGeneratorEngine.cs, який координує послідовний виклик паралельних обчислювальних потоків через C# Job System.

3.2.1 Етап формування сферичної основи та тектонічної симуляції

Спочатку запускається GenerateFibonacciSphereJob, що використовує алгоритм рівномірного розподілу точок (решітка Фібоначчі) для формування ідеальної початкової сфери заданого радіуса.

Після цього ініціалізується модуль симуляції літосферних плит (TectonicSimulationJobs.cs). Важливим моментом програми є визначення належності клітини до найближчого тектонічного центру та розрахунок

динамічної модифікації рельєфу на межах плит через скалярний добуток векторів швидкостей:

```
[BurstCompile(CompileSynchronously = true)]
public struct ApplyTectonicErosionJob : IJobParallelFor
{
    [ReadOnly] public NativeArray<Core.CellData> InCells;
    [ReadOnly] public NativeArray<int> NeighborGraph;
    public NativeArray<Core.CellData> OutCells;

    public void Execute(int index)
    {
        Core.CellData currentCell = InCells[index];
        float heightModification = 0f;
        int startIdx = index * 6;

        for (int i = 0; i < 6; i++)
        {
            int neighborIndex = NeighborGraph[startIdx + i];
            if (neighborIndex == -1) continue;

            Core.CellData neighborCell = InCells[neighborIndex];

            // Якщо клітини належать різним тектонічним плитам
            if (currentCell.TectonicPlateId != neighborCell.TectonicPlateId)
            {
                float3 relativeVelocity = currentCell.TectonicVelocity - neighborCell.TectonicVelocity;
                float3 directionToNeighbor = math.normalize(neighborCell.Position - currentCell.Position);

                // Визначення сили тиску/розходження (векторне стиснення)
                float pressure = math.dot(relativeVelocity, directionToNeighbor);
```

```
                if (pressure > 0.01f) // Зона конвергенції (зіткнення плит)
                {
                    if (currentCell.CrustThickness > 1.5f && neighborCell.CrustThickness > 1.5f)
                    {
                        heightModification += pressure * 2.5f; // Утворення гірських хребтів
                    }
                    else if (currentCell.CrustThickness < 1.0f)
                    {
                        heightModification -= pressure * 1.2f; // Субдукція (океанічні жолоби)
                    }
                }
                else if (pressure < -0.01f) // Зона дивергенції (розходження плит)
                {
                    heightModification += pressure * 0.8f; // Рифтові долини
                }
            }
        }

        currentCell.Height += heightModification;
        OutCells[index] = currentCell;
    }
}
```

3.2.2 Моделювання клімату, вологості та ерозійних процесів

Після формування макрорельєфу тектонічними силами запускаються процеси ерозії рельєфу та класифікації екосистем (PlanetErosionJobs.cs).

- Термальна ерозія (ThermalErosionJob): Перерозподіляє висоти між сусідніми осередками, якщо різниця рівнів перевищує кут природного укосу (критерій сипучості речовини TalusThreshold).
- Гідравлічна ерозія (HydraulicErosionJob): Моделює випадіння опадів, розчинення породи (седиментацію), перенесення осаду потоками води до низин та його акумуляцію (депозицію).

На основі фінальної карти висот та залишків поверхневої вологи розраховується термодинамічний градієнт з урахуванням широти осередку та висотної поясності. Ниче наведено важливий фрагмент коду визначення біомів за оптимізованою матрицею Вайттекера:

```
private byte DetermineBiome(float temp, float moisture, bool isOcean)
{
    if (isOcean)
    {
        if (temp < 0f)   return 0; // Льодовик
        if (temp > 20f)  return 1; // Тропічний шельф
        return 2;                  // Відкритий океан
    }

    if (temp < -5f) return 3; // Тундра
    if (temp < 3f)  return moisture > 0.4f ? (byte)4 : (byte)5; // Тайга / Холодна пустеля

    if (temp < 18f)
    {
        if (moisture < 0.2f) return 6; // Чагарники / Савана
        if (moisture < 0.6f) return 7; // Помірний ліс
        return 8;                      // Дощовий ліс помірного поясу
    }
```

```
    return moisture > 0.5f ? (byte)9 : (byte)10; // Тропічний ліс / Гаряча пустеля
}
```

3.3 Візуалізація та налаштування інтерактивної сцени у Unity

Для демонстрації результатів роботи алгоритмів процедурної генерації та проведення експериментів було розроблено та оптимізовано візуальну тривимірну сцену в Unity.

3.3.1 Візуалізація полігональної сітки планети

Клас PlanetMeshVisualizer динамічно збирає топологічні дані з масиву CellData, генерує вершини (vertices), індекси трикутників (triangles) і колірні буфери (vertex colors), передаючи їх безпосередньо у Unity MeshFilter та MeshCollider. Візуалізатор підтримує три режими відображення (PlanetRenderMode): відображення біомів, карти висот (HeightMap) та карти температур для покрокового аналізу роботи генератора.

3.3.2 Налаштування сцени, камери та інтерфейсу користувача

Для створення повноцінного ігрового додатку та проведення досліджень середовище Unity налаштовано наступним чином:

Конфігурація Камери (Орбітальний контролер): Камера на сцені оснащена скриптом OrbitalCameraController, який використовує нову систему введення Unity Input System. Вона фокусується на координатах центру згенерованої планети (0,0,0). Реалізовано плавне обертання мишею за допомогою кватерніонів (Quaternion.Euler), інтерполяцію наближення/віддалення (Mathf.Lerp) через зміну радіуса сферичних координат камери, та обмеження кутів нахилу по

вертикалі (clamping) від $-85°$ до $85°$, що запобігає "перевертанню" камери на полюсах [26].

Світло та Атмосфера: На сцені використовується одне джерело спрямованого світла (Directional Light), що симулює сонячне випромінювання та створює динамічні тіні від сформованих гірських хребтів на процедурній меш-сітці. Як матеріали для Skybox було інтегровано безкоштовний ассет Space Skyboxes 4K з Unity Asset Store [27], що дозволило досягти чіткого космічного оточення.

Для взаємодії користувача з генератором планет було реалізовано простий та зручний інтерфейс на базі Unity UI і TextMeshPro. У меню передбачено поле введення текстового сіда, яке дозволяє задавати унікальні параметри генерації світу. Після натискання клавіші пробілу система автоматично запускає повторну генерацію планети з новими або зміненими вхідними даними, забезпечуючи швидке створення різних варіантів світів у реальному часі.

Окрім цього, реалізовано систему перемикання режимів візуалізації за допомогою клавіш клавіатури. Натискання клавіш 1, 2 та 3 дозволяє миттєво змінювати поточний режим відображення карти між біомами, картою висот і температурною картою відповідно. Такий підхід значно спрощує аналіз процедурно згенерованого середовища та дає можливість досліджувати особливості рельєфу, кліматичних зон і розподілу біомів безпосередньо під час роботи програми.

3.3.3 Верифікація тривимірної генерації об'єкта та рельєфних нерівностей

У ході фінального інтеграційного тестування в середовищі Unity було повністю підтверджено коректність побудови та візуалізації планетарного об’єкта. Програма безпомилково обробляє розраховані масиви даних і транслює їх у завершену тривимірну полігональну модель. На поверхні згенерованої

планети чітко відображаються всі фізичні нерівності та геоморфологічні структури, створені генератором що показано на рис. 2.7.

Рисунок 2.7 - 3D-рендер планети на сцені

Тектонічні макроформи формуються завдяки зміщенню вершин меш-сітки вздовж нормалей відповідно до значень поля Height структури CellData. У результаті на поверхні виразно помітні гірські хребти та підняття в місцях зіткнення плит, а також чітко окреслені берегові лінії й океанічні западини. Додатково робота алгоритмів термального згладжування та гідравлічного розмиву створює плавні переходи схилів, природні обриси гірських масивів і басейни стоку води, що робить рельєф реалістичним та усуває геометричні артефакти сферичної поверхні.

Текстурне забарвлення біомів і кліматичних зон повністю синхронізоване з тривимірним рельєфом: снігові області формуються на височинах і піках, тоді як ліси та шельфові зони – у низинах і прибережних регіонах. Візуальний моніторинг у реальному часі також підтвердив правильне оновлення компонентів MeshFilter і MeshCollider. Це забезпечує не лише точне відображення рельєфу під будь-яким кутом огляду, а й створює коректну фізичну поверхню для взаємодії з ігровими об’єктами та персонажами.

3.3.4 Розробка підсистеми візуалізації атмосфери та динамічного обертання

Для підвищення рівня реалістичності тривимірної сцени, забезпечення візуальної глибини планетарного об'єкта та симуляції його добового руху було розроблено додаткові компоненти графічного оформлення та кінематики.

В основі математичної логіки шейдера атмосфери реалізовано кілька ключових алгоритмів, які забезпечують реалістичне відображення газової оболонки планети. Ефект Френеля використовується для розрахунку залежності інтенсивності світіння від кута огляду камери на поверхню сфери. Завдяки цьому атмосферне світіння найбільш помітне по краях планетарного диска та поступово згасає до центру, імітуючи природне розсіювання світла у щільних шарах атмосфери. Додатково через скалярний добуток вектора напрямку світла та нормалей поверхні формується динамічне затінення, унаслідок чого світіння проявляється лише на освітленій стороні планети та плавно зникає в зоні термінатора між днем і ніччю. Налаштування даних параметрів представлено на рис. 2.8.

Шейдер також оснащено системою гнучкого налаштування параметрів через редактор Unity. Завдяки зовнішнім властивостям (Properties) можна змінювати колірний спектр атмосферного розсіювання, товщину атмосферного шару та силу експоненціального згасання світла без необхідності змінювати програмний код. Це значно спрощує процес тестування та дозволяє оперативно адаптовувати візуальні характеристики атмосфери відповідно до художніх або наукових вимог проєкту.

Рисунок 2.8 - Налаштування Shader Graph для атмосфери

Для створення ефекту добового руху планети було розроблено окремий кінематичний модуль PlanetRotator. Скрипт інтегрується як компонент MonoBehaviour до об’єкта планети та в кожному кадрі виконує її обертання навколо локальної осі Y за допомогою функції Transform.Rotate. Щоб швидкість обертання не залежала від частоти кадрів на різних пристроях, величина кутового зміщення множиться на Time.deltaTime. Крім того, параметр rotationSpeed винесений в інспектор Unity, що дозволяє в реальному часі регулювати швидкість обертання планети для детального аналізу зміни освітлення, циклу дня і ночі.

3.4 Підсистема растрового експорту

Важливою вимогою до практичної цінності системи є можливість збереження результатів у вигляді безшовних двовимірних текстур (еквіректангулярна проєкція карти світу) для використання у зовнішніх ГІС або ігрових додатках.

Модуль PlanetMapExporter працює асинхронно через Task-паралелізм, що запобігає заморожуванню головного потоку рендерингу Unity. Для заповнення двовимірної сітки пікселів за даними нерівномірних сферичних осередків використано метод обернених зважених відстаней (IDW – Inverse Distance Weighting). Принцип роботи алгоритму IDW-інтерполяції в експортері:

```
// Для кожного пікселя карти (x, y) розраховуються сферичні координати (широта/довгота)
// та відповідний тривимірний вектор напрямку `dir` на сфері.
// Проводиться швидкий пошук сусідніх осередків планети через систему просторових кошиків.

float sumWeights = 0f;
float3 accumulatedColor = float3.zero;

for (int k = 0; k < IDW_NEIGHBORS; k++)
{
    int cellIdx = nearestIndices[k];
    float dist = 1f - math.dot(dir, cells[cellIdx].Position / PlanetRadius);

    // Запобігання діленню на нуль у точці збігу
    float weight = 1f / math.pow(math.max(dist, 1e-5f), IDW_POWER);

    accumulatedColor += GetColorForCell(cells[cellIdx], mode) * weight;
    sumWeights += weight;
}
float3 finalPixelColor = accumulatedColor / sumWeights;
```

Для остаточного підтвердження працездатності підсистеми PlanetMapExporter було проведено експертний аналіз фінальних растрових зображень, отриманих у ході тестового запуску конвеєра. Результати експорту підтверджують безпомилкову трансляцію просторових даних:

Рисунок 2.9 - Карта висот (PlanetMap_HeightMap)

Згенерований 16-бітний монохромний растр на рис. 2.9 чітко демонструє диференціацію рельєфу. Зони стику тектонічних плит відображені у вигляді контрастних світлих ліній (гірські хребти), що плавно переходять у темні півтони рівнин та глибокі чорні градієнти океанічних западин. Наявність розгалужених флювіальних систем свідчить про успішне моделювання гідравлічної ерозії та коректну роботу алгоритму IDW-інтерполяції, який повністю усунув дискретність вихідної сітки.

Рисунок 2.10 - Карта температур (PlanetMap_TemperatureMap)

Результати експорту температурних полів на рис. 2.10 підтверджують правильність фізико-географічного зонування. На карті чітко спостерігається кусинусоїдальний спад температур від екваторіальної зони (насичені теплі кольори) до полярних регіонів (холодні сині відтінки). Сигма-корекція висот відпрацювала коректно: у високогірних районах екватора зафіксовано локальне зниження температури, що верифікує зв'язність даних усередині обчислювальних потоків.

Рисунок 2.11 - Карта біомів (PlanetMap_Biomes)

Фінальний RGB-растр екосистем є результатом суперпозиції карт висот, температур та вологості. На рис. 2.11 чітко простежуються планіметрично правильні межі природних зон (від засніжених полюсів і тундри до помірних лісів та екваторіальних пустель). Завдяки інтегрованому фільтру Гаусса (BLUR_RADIUS = 6), переходи між біомами мають згладжений, художньо виразний вигляд. Океанічні зони успішно розділені на світле прибережне мілководдя (шельфи) та глибоководні масиви.

Таким чином, візуальний аналіз та математична перевірка експортованих карт PlanetMap доводять високу надійність підсистеми растрового моніторингу. Програма повністю виключає появу географічних або екологічних аномалій,

забезпечуючи генерацію безшовних текстур, готових до інтеграції у зовнішні ігрові проєкти чи геоінформаційні системи.

3.5 Модульне тестування та експериментальний аналіз продуктивності

Для перевірки стабільності та надійності розробленого програмного забезпечення за умов інтенсивних обчислень було проведено серію тестових запусків конвеєра генерації. Випробування здійснювалися на персональному комп'ютері користувача, який за сучасними мірками є доволі застарілим (процесор інженерного класу Intel Core i7-3520M, базова тактова частота 2.90 GHz, 8 ГБ оперативної пам'яті). Проте, навіть на такій апаратній конфігурації розроблена система демонструє високу ефективність завдяки низькорівневій оптимізації.

Експериментальні дослідження показали, що первинна ініціалізація комплексу займає близько 20 секунд. Такий тривалий час запуску зумовлений виконанням низки підготовчих процесів: виділенням некешованої рідної пам’яті (NativeArray) для структур даних, «прогрівом» системи, JIT-компіляцією Burst-завдань та генерацією базового топологічного графа планети.

Натомість процес динамічної регенерації світу за новим випадковим сідом відбувається значно швидше – максимум за 1 секунду. У цей момент система виконує повне оновлення фізичних та кліматичних параметрів осередків без потреби у додатковому перевиділенні пам’яті, що дозволяє значно оптимізувати роботу програми під час її функціонування.

Проте під час цієї швидкої регенерації обчислювальні ресурси процесора завантажуються на повну потужність, що спричиняє короткочасне зависання інтерфейсу та блокування камери. Одразу після завершення розрахунків система стабілізується та миттєво повертається до штатного високопродуктивного режиму рендерингу, забезпечуючи подальшу плавну роботу.

Проведений аналіз дозволяє зробити висновок, що використання паралельних потоків C# Job System та векторизації інструкцій дозволяє успішно експлуатувати систему процедурної генерації навіть на технічно застарілому обладнанні, забезпечуючи комфортний рівень продуктивності (60 FPS) для дослідження згенерованих ландшафтів одразу після завершення секундного циклу симуляції.

У третьому розділі успішно здійснено практичну програмну реалізацію системи процедурного моделювання. На основі розробленої архітектури модулів та проведених експериментів встановлено, що використання паралельних обчислювальних закономірностей C# Job System у поєднанні з низькорівневою компіляцією Burst забезпечує лінійну масштабованість системи при збільшенні обсягів вхідних даних. Підготовлено повноцінне демонстраційне середовище у Unity з інтерактивним UI, орбітальним контролером камери та модулем асинхронного експорту безшовних растрових карт високої роздільної здатності. Функціональне тестування підтвердило повну відповідність розробленого продукту висунутим вимогам надійності, швидкодії та зручності користувацького інтерфейсу.

ВИСНОВКИ

У кваліфікаційній (бакалаврській) роботі відповідно до поставленої мети та визначених завдань було виконано комплексне дослідження та розробку високопродуктивної підсистеми процедурної генерації реалістичних тривимірних ландшафтів та складних екосистем для ігрових додатків. У процесі виконання роботи вдалося детально проаналізувати сучасні підходи до процедурного моделювання віртуальних просторів. Це дозволило виявити суттєві обмеження традиційних алгоритмів аналітичного шуму, таких як шум Перліна чи Сімплекс, які при створенні глобальних сферичних об'єктів призводять до геометричних викривлень у полярних областях і не забезпечують природної зв'язності біомів.

Для вирішення цих проблем у роботі було застосовано комбінацію методів, що імітують фізичні процеси взаємодії літосферних плит та клітинних автоматів гідравлічної й термальної ерозії. Такий підхід дав можливість відтворити у віртуальному середовищі реалістичні макроформи рельєфу, зокрема гірські хребти, океанічні западини та рифтові зони, не випадковим чином, а як логічний результат симуляції взаємодії векторів кутових швидкостей тектонічних структур. Додатково було модифіковано класичну модель кліматичних зон Вайттекера. Інтеграція розрахунку температурного градієнта залежно від широти осередку та карти висот забезпечила високу точність і біологічну достовірність при формуванні рослинних біомів та трофічних рівнів екосистем.

Експериментальна перевірка розробленого модуля асинхронного растрового експорту на основі алгоритму обернених зважених відстаней та полярної фільтрації підтвердила можливість успішного створення безшовних еквіректангулярних карт висот та біомів із роздільною здатністю 2048 × 1024. Отримані карти є повністю сумісними з сучасними індустріальними ігровими рушіями та геоінформаційними системами.

Результати проведеного дослідження мають чітке прикладне значення для індустрії розробки програмного забезпечення та комп'ютерних ігор, оскільки

дозволяють автоматизувати створення великих відкритих світів, суттєво зменшуючи витрати часу 3D-художників на ручне моделювання та гарантуючи при цьому високу математичну й фізичну достовірність згенерованого ландшафту. Перспективи подальшого розвитку розробки пов'язані з оптимізацією алгоритмів ерозії для їх виконання безпосередньо під час ігрового процесу в реальному часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Freiknecht J., Effelsberg W. A Survey on the Procedural Generation of Virtual Worlds. *Multimodal Technologies and Interaction*. 2017. Vol. 1, No. 4. Art. 27. URL: https://www.researchgate.net/publication/320722498_A_Survey_on_the_Procedural_Generation_of_Virtual_Worlds (дата звернення: 25.05.2026).

2. Ebert D. S., Musgrave F. K., Peachey D., Perlin K., Worley S. Texturing and Modeling: A Procedural Approach. 3rd ed. Morgan Kaufmann, 2003. Reading sample URL: https://content.e-bookshelf.de/media/reading/L-29210131-2783ab553d.pdf (дата звернення: 25.05.2026).

3. Davis W. Faster Procedural Noise Generation with Unity Burst & Jobs. *Medium / CodeX*. 2023. URL: https://medium.com/@willdavis84/faster-procedural-noise-generation-with-unity-burst-jobs-2bfa0f9aff85 (дата звернення: 25.05.2026).

4. Patel A. Polygon Map Generation for Games. *Stanford Student Game Programming Articles*. 2010. URL: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/ (дата звернення: 25.05.2026).

5. Galin E., Guérin E., Peytavie A., Cordonnier G., Gain J. Authoring Landscapes and Planets. *LIRIS Research Articles*. Lyon, 2019. URL: https://perso.liris.cnrs.fr/eric.galin/Articles/2019-planets.pdf (дата звернення: 25.05.2026).

6. Genevaux, Galin et al. Terrain Generation Using Procedural Models Based on Hydrology. *SIGGRAPH Asia 2013 Technical Briefs* (Hong Kong, 19-22 November 2013). New York: ACM, 2013. URL: https://hal.science/hal-01339224/file/siggraph2013.pdf (дата звернення: 25.05.2026).

7. Hartley M., Mellado N., Fiorio C., Faraj N. Flexible terrain erosion. *The Visual Computer*. 2024. Vol. 40, No. 7. P. 4593–4607. URL: https://www.lirmm.fr/~nfaraj/publications/flexible_erosion/2024_Flexible_Terrain_Erosion.pdf (дата звернення: 25.05.2026).

8. Dey T. K. Delaunay Mesh Generation: Three-Dimensional Delaunay Triangulation. Lecture Notes / Department of Computer Science and Engineering, Purdue University. West Lafayette, 2012. URL: https://www.cs.purdue.edu/homes/tamaldey/course/531/Delaunay%283D%29.pdf (дата звернення: 25.05.2026).

9. Mocika C. Procedural Voronoi-Tiled Planets by Cleon Mocika. *Digital Arts & Entertainment (DAE)*. 2021. URL: https://www.digitalartsandentertainment.be/article/213/Procedural+Voronoi-Tiled+Planetsby+Cleon+Mocika (дата звернення: 25.05.2026).

10. Davies J. Voronoi Tessellation Map Projections. *Jason Davies Interactive Maps*. 2015. URL: https://www.jasondavies.com/maps/voronoi/ (дата звернення: 25.05.2026).

11. Red Blob Games. Delaunay Triangulation and Voronoi Diagrams on a Sphere. *Red Blob Games Research Labs*. 2018. URL: https://www.redblobgames.com/x/1842-delaunay-voronoi-sphere/ (дата звернення: 25.05.2026).

12. Schulz H., Cordonnier G., Galin E., Guérin E., Gain J., Benes B. Interactive Simulation of Tectonic Plates and Mountain Building. *arXiv preprint arXiv:2410.12007v1*. 2024. URL: https://arxiv.org/html/2410.12007v1 (дата звернення: 25.05.2026).

13. Frozen Fractal. Around the world 23: Hydraulic erosion. *Frozen Fractal Blog*. 2025. URL: https://frozenfractal.com/blog/2025/6/6/around-the-world-23-hydraulic-erosion/ (дата звернення: 25.05.2026).

14. Here Dragons Abound. Learning from Azgaar: Terrain Generation Analysis. *Here Dragons Abound Blog*. 2019. URL: https://heredragonsabound.blogspot.com/2019/10/learning-from-azgaar-terrain-generation.html (дата звернення: 25.05.2026).

15. Edelvik L. Procedural generation of planets with various biomes in Unity: Student thesis. Karlskrona: Blekinge Institute of Technology, 2019. 46 p. URL: https://www.diva-portal.org/smash/get/diva2:1355216/FULLTEXT01.pdf (дата звернення: 25.05.2026).

16. Navarro A., Merino A., García-Ortega E., Tapiador F. J. Uncertainty maps for model-based global climate classification systems. Scientific Data. 2025. Vol. 12, Art. 35. URL: https://www.researchgate.net/publication/387834540_Uncertainty_maps_for_model-based_global_climate_classification_systems (дата звернення: 25.05.2026).

17. Generalist Programmer. Unreal Engine vs Unity: AAA Game Development Showdown. *Generalist Programmer Tutorials*. 2023. URL: https://generalistprogrammer.com/tutorials/unreal-engine-vs-unity-aaa-game-development-showdown (дата звернення: 25.05.2026).

18. Mobile App Development Company. Unity vs Unreal Engine: Which is Better for Game Development? *Tech Blog*. 2023. URL: https://www.mobileappdevelopmentcompany.us/blog/unity-vs-unreal-engine/ (дата звернення: 25.05.2026).

19. Torrahod G. Analyzing Burst-Generated Assemblies in Unity. *Game Torrahod Developer Blog*. 2022. URL: https://gametorrahod.com/analyzing-burst-generated-assemblies/ (дата звернення: 25.05.2026).

20. Unity Technologies. C# Job System Overview. *Unity Manual (v. 6000.3)*. 2024. URL: https://docs.unity3d.com/6000.3/Documentation/Manual/job-system-overview.html (дата звернення: 25.05.2026).

21. Nbhr. Geodome: Procedural geodesic dome and icosphere generation in Unity. *GitHub Repository*. 2020. URL: https://github.com/nbhr/geodome (дата звернення: 25.05.2026).

22. Alexandru C. Procedurally Generating Wrapping World Maps in Unity C# (Part 4). *Game Developer*. 2014. URL: https://www.gamedeveloper.com/programming/procedurally-generating-wrapping-world-maps-in-unity-c-part-4 (дата звернення: 25.05.2026).

23. Azgaar. Fantasy Map Generator: Procedural World Generation Tool. *GitHub Pages Resource*. 2018. URL: https://azgaar.github.io/Fantasy-Map-Generator/ (дата звернення: 25.05.2026).

24. Chavez N. ProcTerrain: Procedural Infinite Terrain Generation Project. *Nicolas Chavez Portfolio*. 2020. URL: https://nicolaschavez.com/projects/procterrain/ (дата звернення: 25.05.2026).

25. Ni2Be. Unity_Intro: Introduction to Unity development and custom packages. *GitHub Repository*. 2021. URL: https://github.com/Ni2Be/Unity_Intro (дата звернення: 25.05.2026).

26. Superhedral. Lerping Cameras in Unity. *Superhedral Computer Graphics Blog*. 2021. URL: https://superhedralcom.wordpress.com/2021/10/30/lerping-cameras-in-unity/.

27. Asset Store Unity. Space Skyboxes 4K Free. *Unity Asset Store / Texture & Materials Packages*. 2019. URL: https://assetstore.unity.com/packages/2d/textures-materials/sky/space-skyboxes-4k-free-354458 (дата звернення: 25.05.2026).

ДОДАТКИ

ДОДАТОК А

Таблиця А.1 Архітектурні компоненти та зони їхньої відповідальності

Назва компонента	Тип реалізації	Зона архітектурної відповідальності	Опис взаємодії та зв'язності
PlanetGeneratorEngine	MonoBehaviour	Координація конвеєра, виділення Native-пам'яті, обробка подій введення	Точка входу. Керує життєвим циклом Native-масивів, викликає паралельні та асинхронні завдання.
CellData	Blittable-структура	Збереження фізичних та екологічних параметрів окремої клітини сфери	Елемент масиву PlanetCells. Має фіксований розмір для оптимального укладання в кеш-лінії L1/L2 CPU.
BuildSpatialGridJob	Burst-завдання (IJob)	Сортування клітин за тривимірними просторовими бакетами (Counting Sort)	Формує допоміжні індекси зміщення для швидкого просторового пошуку суміжних вершин.
BuildNeighborGraphJob	Burst-завдання (IJobParallelFor)	Паралельна побудова топологічного графу сусідів на сфері	Проводить вибірку з просторової сітки, заповнює глобальний плоский граф суміжності NeighborGraph.

ThermalErosionJob	Burst-завдання (IJobParallelFor)	Паралельна симуляція гравітаційного осипання нестабільних схилів	Зчитує висоти з вхідного буфера, розраховує кути укосу, записує стабілізовані висоти у вихідний буфер.
HydraulicErosionJob	Burst-завдання (IJob)	Однопотокова фізична симуляція водної ерозії, переносу та депонування осаду	Послідовно оновлює висоти та концентрації вологи у глобальному масиві клітин за законом збереження маси.
ComputeClimateAndBiomesJob	Burst-завдання (IJobParallelFor)	Розрахунок температурного градієнта та класифікація біомів Вайттекера	Обчислює кліматичні параметри та детерміновано призначає індекси рослинного покриву кожній клітині.
StereographicProjectionJob	Burst-завдання (IJobParallelFor)	Конформне відображення тривимірних полярних координат у двовимірну площину	Проєктує координати вершин для підготовки до алгоритму площинної тріангуляції.
PlanetMapExporter	Асинхронний сервіс	Растеризація сферичних даних у циліндричну текстуру з фільтрацією	Виконує багатопотокову IDW-інтерполяцію та Gaussian-розмиття у фоновому потоці без затримок основного кадру.