МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Процедурна генерація ландшафту та рендеринг у OpenGL**

Виконав: студент 2 курсу, групи 8.1214-пзс-2
ціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
Н. С. Кузенний
(ініціали та прізвище)
Керівник доцент, к.т.н., В.І.Заяц
(посада,вчене звання,науковий ступінь,підпис,ініціали та прізвище)
Рецензент Директор ТОВ «Дискус»,
Р.О. Лютий
(посада,вчене звання,науковий ступінь,підпис,ініціали та прізвище)

Запоріжжя
2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙНАВЧАЛЬНО-НАУКОВИЙІНСТИТУТ
ім.Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра: електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти другий (магістерський)
Спеціальність 121 Інженеріяпрограмногозабезпечення
(код та назва)

Освітня програма Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Т.В. Критська

"_1_" квітня 2025_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кузенному Нікіті Сергійовичу
(прізвище,ім'я,по батькові)

1. Тема роботи Процедурна генерація ландшафту та рендеринг у OpenGL

керівник роботи Заяц Валерій Іванович, доцент, к.т.н.
(прізвище, ім'я ,по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 28.03. 2025 р. № 361-е

2. Строк подання студентом кваліфікаційної роботи 01.12.2025 р.
3. Вихідні дані магістерської роботи
   - комплект нормативних документів ;
   - технічне завдання до роботи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   - огляд та збір літератури стосовно теми кваліфікаційної роботи;
   - огляд та аналіз існуючих рішень та аналогів;
   - дослідження проблеми розробки генератора ландшафту для OpenGL;
   - створення програмного продукту та його опис;
   - дослідження поставленої проблеми та розробка висновків.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   слайдів презента

Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис,дата Завданняприйняв

6. Дата видачі завдання 01.04.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	15.04–20.04.25	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	21.04–30.04.23	виконано
3	Аналіз існуючих рішень	01.05– 10.05.25	виконано
4	Узгодження подальших дій з науковим керівником	11.05– 20.05.25	виконано
5	Визначення вимог	21.05– 06.08.25	виконано
6	Вибір технологій реалізації	26.09-27.09.25	виконано
7	Створення дизайн інтерфейсу користувача за допомогою Qt Framework	28.09-12.10.25	виконано
8	Імплементація попереднього перегляду у реальному часі	13.10-15.10.25	виконано
9	Імплементація алгоритмів процедурної генерації ландшафту	16.10-17.10.25	виконано
10	Реалізація алгоритму симуляції ерозії ландшафту у реальному часі	19.10-08.11.25	виконано
11	Апробація застосунку	09.11-14.11.25	виконано
12	Оформлення звіту	15.11-22.11.25	виконано
13	Оформлення презентації	25.11-01.12.25	виконано

Студент _____ Кузенний Н.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Заяц В.І.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено
Нормоконтролер _____ Скрипник. І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок — 78

Рисунків — 23

Таблиці — 1

Джерел — 30

Кузенний Н.С. Процедурна генерація ландшафту та рендеринг у OpenGL: кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник В. І. Заяц. Запоріжжя : ЗНУ, 2025. 78 с.

У цій дисертації представлено розробку інтерактивного застосунку для генерації ландшафту, який поєднує процедурні методи шуму з моделюванням гідравлічної ерозії. Метою роботи є створення системи, яка може генерувати різноманітний рельєф місцевості за допомогою кількох методів на основі шуму та додатково його уточнювати за допомогою процесу ерозії.

Застосунок реалізує кілька шумових функцій, включаючи шум Перліна, фрактальний броунівський рух, шум, деформований за доменом, та варіант шуму Перліна за абсолютним значенням. Ці алгоритми формують початковий рельєф місцевості, який користувачі можуть змінювати за допомогою моделювання гідравлічної ерозії.

Ключовою частиною роботи є порівняння реалізованих методів генерації та аналіз їх візуальних та обчислювальних характеристик. Систему ерозії також було покращено за допомогою невеликої модифікації.

Кінцевим результатом є функціональний застосунок, який поєднує генерацію рельєфу на основі шуму з оптимізованою моделлю ерозії, пропонуючи візуалізацію в режимі реального часу.

Ключові слова: ландшафт, генерація, шум Перліна, ерозія, OpenGL, сцена, візуалізація, у реальному часі.

ABSTRACT

Pages — 78

Drawings — 23

Tables — 1

Sources — 30

Kuzennyi N.S. Procedural landscape generation and rendering in OpenGL: qualification work of the master of specialty 121 "Software Engineering" / Science. manager V.I. Zayats. Zaporizhzhia: ZNU, 2025. 78 p.

This thesis presents the development of an interactive landscape generation application that combines procedural noise techniques with a hydraulic erosion simulation. The goal of the work is to create a system that can generate varied terrain using several noise-based methods and refine it further through an erosion process.

The application implements multiple noise functions, including Perlin noise, fractal Brownian motion, domain-warped noise, and an absolute-value variant of Perlin noise. These algorithms form the initial terrain, which users can modify through hydraulic erosion simulation.

A key part of the thesis is the comparison of the implemented generation techniques and the analysis of their visual and computational characteristics. The erosion system was also improved through a small modification.

The final outcome is a functional application that combines noise-based terrain generation with an optimized erosion model, offering real-time visualization.

Keywords: landscape, generation, Perlin noise, erosion, OpenGL, scene, rendering, real-time.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ АНАЛОГІВ ПРЕДМЕТНОЇ ОБЛАСТІ 12
1.1 Огляд літературних джерел 12
1.2 Аналіз аналогів програмного забезпечення 14
1.2.1 Terragen 14
1.2.2 World Machine 16
1.2.3 QuadSpinner Gaea 18
1.2.4 Висновки аналізу аналогів 20
1.3 План робіт 21
2 ТЕОРЕТИЧНІ ОСНОВИ ДЛЯ РОЗРОБКИ СИСТЕМ ДЛЯ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ЛАНДШАФТУ 23
2.1 Загальні відомості про процедурну генерацію ландшафту 23
2.2 Аналіз сучасних технологій розробки генератора ландшафту 25
3 РОЗРОБКА ГЕНЕРАТОРА ЛАНДШАФТУ У РЕАЛЬНОМУ ЧАСІ 29
3.1 Опис предметної області 29
3.2 Архітектура системи 31
3.3 Алгоритми генерації шуму 33
3.3.1 Шум Перліна 33
3.3.2 Фрактальний броунівський рух 36
3.3.3 Шум деформованих ділянок 38
3.3.4 Абсолютний шум 41
3.4 Симуляція ерозії 43
3.4.1 Теоретичні основи гідравлічної ерозії 43
3.4.2 Реалізація алгоритму гідравлічної ерозії 45
3.5 Модуль візуалізації 47
3.6 Функціонал системи 50
3.7 Вимоги до інтерфейсу 60
3.8 Аналіз вимог до користувачів генератора ландшафту 62

3.9 Аналіз вимог до апаратного та програмного забезпечення 64
4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ ГЕНЕРАЦІЇ ЛАНДШАФТУ У РЕАЛЬНОМУ ЧАСІ 65
4.1 Аналіз результатів генерації рельєфу на основі шуму 65
4.1.1 Шум Перліна 65
4.1.2 Фрактальний броунівський рух (fBm) 66
4.1.3 Шум деформованих ділянок 67
4.1.4 Абсолютний шум 68
4.1.5 Порівняння результатів 69
4.2 Аналіз результатів симуляції гідравлічної ерозії 70
ВИСНОВКИ 75
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 76

ВСТУП

Актуальність теми

Генерація реалістичного ландшафту стала невід'ємним компонентом сучасної комп'ютерної графіки, систем моделювання та інтерактивних програм. Зі зростанням масштабу та складності віртуальних середовищ зростає попит на методи, здатні ефективно та з високим ступенем реалізму створювати природні ландшафти. Процедурна генерація рельєфу стала одним з найефективніших підходів до задоволення цього попиту, оскільки вона дозволяє створювати величезні, різноманітні та деталізовані середовища без необхідності ручного моделювання. Методи, засновані на математичних шумових функціях, фрактальних структурах та фізично основаних процесах, дозволяють динамічно генерувати рельєф, пропонуючи як гнучкість, так і обчислювальну ефективність.

Окрім розважальних програм, таких як відеоігри, генерація рельєфу відіграє значну роль у науковій візуалізації, геопросторовому аналізі, віртуальному прототипуванні та освітніх інструментах моделювання. Реалістичні форми рельєфу необхідні для моделювання природних явищ, вивчення ерозії та потоку води, планування інфраструктури та візуалізації геологічних процесів. У цих сценаріях здатність алгоритмічно генерувати ландшафти забезпечує не тільки візуальну точність, але й відтворюваність та контроль, що дозволяє проводити експерименти, які було б непрактично або неможливо проводити в реальних умовах.

Особливо актуальною галуззю в цій сфері є моделювання ерозії, яке значно підвищує реалізм, вводячи структури, сформовані природними силами. Моделі гідравлічної ерозії імітують вплив опадів, переносу води та руху осаду, в результаті чого місцевість відображає природні закономірності вивітрювання, русла річок та долини. Інтеграція ерозії в процедурну генерацію місцевості усуває розрив між штучною геометрією та автентичною геологічною поведінкою, роблячи її важливою як для візуальних, так і для наукових застосувань. Оскільки обчислювальні ресурси продовжують

зростати, а візуалізація у реальному часі стає все більш доступною, потреба в ефективних, інтерактивних алгоритмах ерозії стає ще більшою.

У цьому контексті дослідження, що зосереджені на вдосконаленні моделей місцевості на основі шуму, покращенні алгоритмів ерозії та розробці інструментів для візуалізації в реальному часі, залишаються дуже актуальними. Така робота сприяє як теоретичному розумінню, так і практичному прогресу в генерації місцевості, дозволяючи створювати більш реалістичні, керовані та обчислювально ефективні ландшафти.

Мета дослідження

Основною метою цього дослідження є розробка застосунку, здатного процедурно генерувати цифровий рельєф, використовуючи комбінацію алгоритмів на основі шуму та моделювання гідравлічної ерозії.

Завдання дослідження

Цілі цього дослідження зосереджені на вивченні теоретичних та практичних основ процедурної генерації рельєфу та використанні цих знань для розробки та впровадження ефективного програмного рішення.

Об’єкт дослідження

Об'єктом дослідження є процес розробки процедурного генератора рельєфу та методи, що використовуються для створення цифрових ландшафтів.

Предмет дослідження

Предметом дослідження в цьому дослідженні є методи процедурної генерації цифрового рельєфу. Це охоплює різні методи на основі шуму, включаючи шум Перліна, фрактальний броунівський рух, шум деформації ділянок, та абсолютний шум, а також алгоритми для моделювання процесів ерозії, які формують рельєф з часом. Дослідження зосереджено на розумінні

того, як ці методи можна застосовувати, поєднувати та оптимізувати для створення реалістичних та візуально привабливих ландшафтів в інтерактивному програмному середовищі.

Методи дослідження

Методи дослідження, що використовуються в цій дисертації, поєднують теоретичний аналіз та практичне впровадження.

Перший метод включає вивчення та систематичний огляд літературних джерел, включаючи наукові статті, книги та технічну документацію, для вивчення існуючих підходів до процедурної генерації рельєфу та моделювання ерозії. Цей огляд забезпечує основу для розуміння сильних сторін, обмежень та сучасних тенденцій у цій галузі.

Другий метод полягає в розробці програми, яка служить практичним експериментом для впровадження, тестування та оцінки різних методів генерації рельєфу. Поєднуючи ці підходи, дослідження інтегрує теоретичні знання з практичним експериментуванням, що дозволяє проводити як аналіз, так і перевірку запропонованих рішень.

Практичне значення одержаних результатів

Практичне значення цього дослідження полягає в розробці та модифікації алгоритму гідравлічної ерозії в рамках процедурної програми генерації рельєфу. Підвищуючи ефективність та продуктивність процесу ерозії, дослідження сприяє створенню більш реалістичних та динамічно змінюваних ландшафтів у режимі реального часу. Модифікований алгоритм зменшує обчислювальні витрати, зберігаючи при цьому візуальну якість деформації рельєфу, що робить його придатним для використання в інтерактивних програмах, системах моделювання та освітніх інструментах. Ці результати забезпечують практичну основу для розробників програмного забезпечення та дослідників, які прагнуть запровадити ефективні методи

генерації рельєфу та ерозії в графічних двигунах або середовищах моделювання в реальному часі.

Глосарій

OpenGL – кросплатформний графічний API, що використовується для рендерингу 2D та 3D графіки.

OpenGL Shading Language (GLSL) – мова програмування, що використовується для написання шейдерів, що виконуються на графічному процесорі.

Функція шуму – математична функція, яка генерує псевдовипадкові значення, що використовуються для створення текстур та рельєфу.

Ерозія – процес, за допомогою якого ґрунт видаляється з поверхні рельєфу за допомогою імітації потоку води.

Осадження – процес переносу ґрунту краплями води, назад на рельєф.

Градієнтна сітка – представлення схилів рельєфу, що використовується для розрахунку напрямку потоку води та ерозії.

Qt Framework – кросплатформний фреймворк для розробки графічних інтерфейсів користувача.

VBO (Vertex Buffer Object) – буфер пам'яті графічного процесора, що використовується для зберігання даних вершин для ефективного рендерингу.

VAO (Vertex Array Object) – об'єкт OpenGL, який зберігає конфігурацію вказівників атрибутів вершин.

EBO (Element Buffer Object) – буфер графічного процесора, який зберігає індекси для відображення вершин у певному порядку.

1 АНАЛІЗ АНАЛОГІВ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

Процедурна генерація рельєфу є важливою галуззю дослідження та застосування в комп'ютерній графіці, розробці ігор та геопросторовому моделюванні. Її актуальність випливає зі зростаючого попиту на великомасштабні, деталізовані та реалістичні віртуальні середовища, які можна ефективно створювати без необхідності тривалого ручного проектування. Оскільки віртуальні світи розширюються як у сфері розваг, так і в науковій візуалізації, методи процедурної генерації забезпечують потужне рішення для автоматизації складних природних ландшафтів.

Процес процедурної генерації рельєфу включає математичні моделі, які наближають природний рельєф за допомогою шумових функцій та моделювання ерозії. Однак цей процес створює кілька технічних та обчислювальних проблем. Однією з ключових проблем є досягнення реалізму при збереженні інтерактивності в реальному часі. Наприклад, хоча прості методи на основі шуму, такі як шум Перліна [1], є обчислювально ефективними, вони часто створюють надмірно гладкі або повторювані візерунки, яким бракує геологічної правдоподібності. Фрактальний броунівський рух (fBm) [2] підвищує реалізм, поєднуючи кілька шарів шуму на різних частотах, але йому все ще бракує таких функцій, як річкові мережі або ефекти вивітрювання, які природним чином виникають на реальних рельєфах.

Ще однією серйозною проблемою є моделювання процесів ерозії, які є важливими для додавання фізично реалістичних деталей, таких як долини та канали. Алгоритми гідравлічної ерозії, описані Масгрейвом та ін. [3] та додатково вдосконалені Кордоньє та ін. [4], намагаються змоделювати, як вода взаємодіє з рельєфом з часом. Ці алгоритми є обчислювально ресурсоємними

та часто вимагають компромісів між точністю моделювання та продуктивністю в реальному часі.

Методи деформації ділянок, спочатку досліджені Перліном, а пізніше розширені в контексті генерації рельєфу [5], допомагають подолати повторювані артефакти простих шумових функцій. Спотворюючи вхідну область шумової функції, ці методи вводять більш природні, нерегулярні особливості, які імітують складність реальних геологічних формацій. Однак поєднання деформації ділянок з ерозією створює нові проблеми, пов'язані з обчислювальними витратами.

Крім того, реалізація інтерактивної системи, де користувачі можуть маніпулювати параметрами генерації, додає ще більшої складності [23]. Візуалізація та зворотний зв'язок у реальному часі вимагають ефективних обчислень на основі графічного процесора та добре структурованих моделей параметризації, щоб забезпечити, щоб зміни призводили до значущих та передбачуваних варіацій у згенерованому рельєфі. Дослідження Смеліка та ін. [6] щодо інтерфейсів процедурного моделювання підкреслює важливість балансування художнього контролю з алгоритмічною автономією в системах генерації рельєфу.

Таким чином, у розглянутій літературі наголошується на тому, що розробка процедурного застосунку для генерації рельєфу, здатного створювати різноманітні, реалістичні рельєфи з інтерактивним моделюванням ерозії, вимагає вирішення кількох взаємопов'язаних проблем: підтримки обчислювальної ефективності, забезпечення природного реалізму та підтримки експериментів, керованих користувачем. Поєднання класичних методів на основі шуму, сучасного деформування ділянок та фізично натхненних моделей ерозії формує основу для вирішення цих проблем як у дослідженнях, так і в практичних впровадженнях.

1.2 Аналіз аналогів програмного забезпечення

1.2.1 Terragen

Terragen — це професійний додаток, призначений для процедурної генерації рельєфу, моделювання ландшафтів та фотореалістичного рендерингу. Розроблений Planetside Software, він широко використовується у створенні візуальних ефектів, науковій візуалізації та створенні віртуального ландшафту завдяки своїй здатності імітувати природні явища з високим рівнем реалізму. Програма дозволяє користувачам створювати складні рельєфи за допомогою процедурних функцій та карт висот, імітувати атмосферні умови, такі як хмари, туман та освітлення, а також рендерити великомасштабні сцени. Terragen інтегрує як процедурну генерацію рельєфу на основі шуму, так і імпорт даних реального світу, що робить його гнучким як для художніх, так і для наукових застосувань.

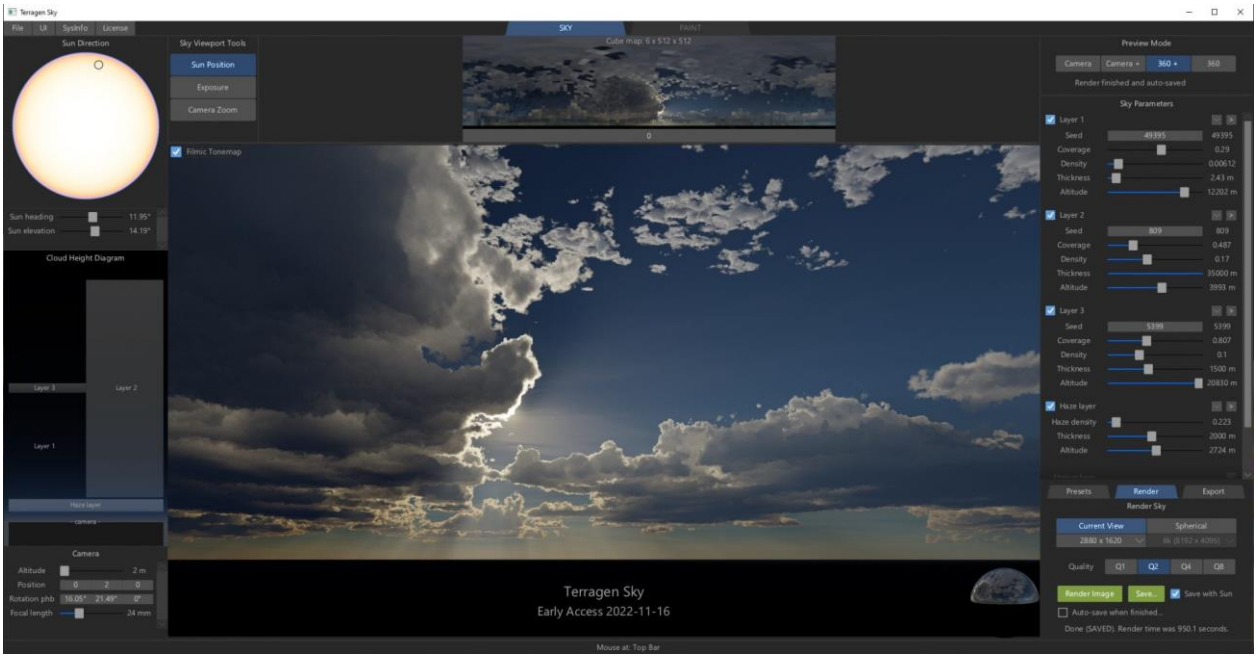


Рисунок 1 — *Інтерфейс користувача Terragen*

Terragen надає інтерфейс (див. Рисунок 1), який дозволяє користувачам визначати структуру рельєфу, текстурні шари та параметри навколишнього середовища за допомогою взаємопов'язаних компонентів. Ця система надає

користувачам значний контроль над кожним аспектом згенерованого світу, що дозволяє точно маніпулювати процедурним шумом, ефектами ерозії та матеріалами поверхні. Його механізм рендерингу підтримує глобальне освітлення, трасування променів та фізичні шейдери, створюючи візуально реалістичні результати, які використовувалися у кіновиробництві та високоякісних симуляціях.

Головна перевага Terragen полягає в його здатності створювати візуально приголомшливі ландшафти, що точно імітують природні ландшафти. Його гнучкість та точність імітації атмосферних умов роблять його особливо корисним для професійного рендерингу та наукової візуалізації. Крім того, процедурний робочий процес на основі вузлів гарантує, що сцени можна легко змінювати та повторно використовувати, що підвищує ефективність у творчих та дослідницьких контекстах.

Однак Terragen також має кілька обмежень. Високий поріг навчання та складність програмного забезпечення роблять його складним у вивченні або для інтерактивної роботи у реальному часі. Хоча воно чудово справляється з офлайн-рендерингом, його продуктивність не оптимізована для симуляції в реальному часі, що обмежує його використання в інтерактивній генерації ландшафту або динамічних системах, керованих користувачем. Крім того, більшість його розширених функцій доступні лише в платних версіях, що може обмежувати експерименти в освітніх або дослідницьких цілях.

Загалом, Terragen являє собою один з найсучасніших інструментів генерації ландшафту, зосереджуючись переважно на реалізмі та візуальній точності, а не на інтерактивності в реальному часі. Він слугує важливим прикладом для процедурних систем генерації рельєфу, демонструючи, чого можна досягти, надаючи пріоритет якості та фізичній точності над обчислювальною продуктивністю.

1.2.2 World Machine

World Machine — це процедурний інструмент для генерації рельєфу, який широко використовується в розробці ігор, кінематографі та наукових симуляціях. Він зосереджений на створенні детальних та природних карт висот за допомогою комбінації процедурного шуму та симуляції ерозії. Розроблений Стівеном Шміттом, додаток надає користувачам робочий процес, зосереджений на побудові мереж генерації рельєфу, які визначають форму, текстуру та структуру ландшафтів. Він особливо цінується за свою здатність інтегрувати складні моделі ерозії, які додають реалістичні геологічні формації до рельєфу, що робить його кращим інструментом для створення базових даних рельєфу для ігрових рушіїв таких, як Unreal Engine або Unity.

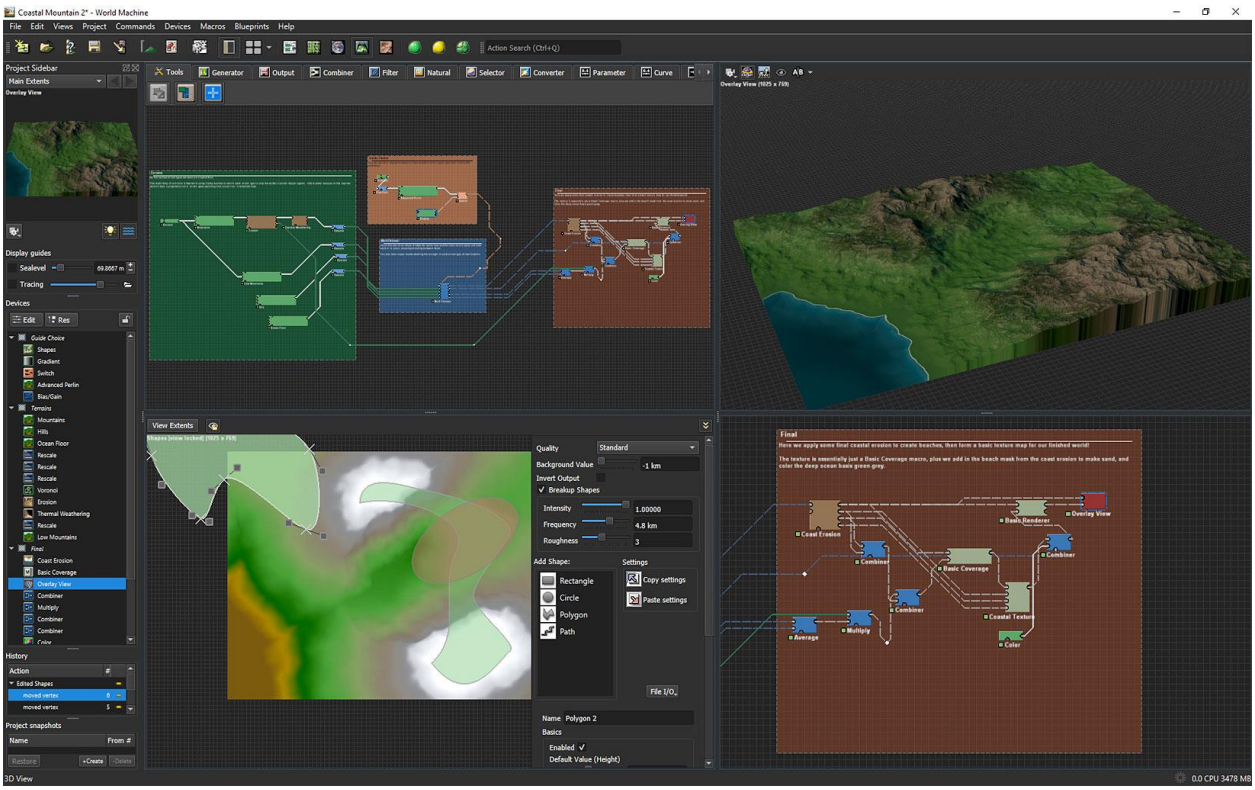


Рисунок 2 — *Інтерфейс користувача World Machine*

World Machine має інтерфейс (див. Рисунок 2) на основі вузлів, подібний до блок-схеми, де кожен вузол представляє процес генерації або модифікації, такий як генерація шуму, ерозія або змішування текстур. Цей візуальний

робочий процес дозволяє користувачам експериментувати з різними процедурними комбінаціями, підключати кілька модифікаторів та переглядати результати. Додаток надає можливість користувачам інтерактивно орієнтуватися на місцевості та досліджувати дрібні деталі в режимі реального часу.

Головна перевага World Machine полягає в його можливостях процедурної генерації та моделювання ерозії. Його система ерозії відома тим, що створює реалістичні моделі потоку води, перенесення осаду та вивітрювання місцевості, які дуже нагадують природні ландшафти. Додаток також пропонує багато опцій експорту, що дозволяє інтегрувати його результати з іншими 3D-інструменти. World Machine забезпечує баланс між автоматизацією та контролем, що робить його придатним як для творчих, так і для технічних завдач.

Однак World Machine має кілька помітних недоліків. Продуктивність програмного забезпечення в реальному часі обмежена, оскільки моделювання з високою роздільною здатністю може стати обчислювально ресурсоємним та вимагати значного часу обробки. Хоча його інтерфейс на основі вузлів ефективний для досвідчених користувачів, він може важким у вивченні через технічну природу процедурного проектування місцевості. Крім того, програма зосереджена переважно на генерації карт висот і не забезпечує розширеного текстурування, розміщення рослинності чи динамічного моделювання погоди, що означає, що її часто потрібно доповнювати іншими інструментами для повноцінного робочого процесу створення ландшафту.

Підсумовуючи, World Machine — це надійна та універсальна система генерації рельєфу, розроблена для точності та геологічного реалізму. Її процедурні та ерозійно-орієнтовані принципи проектування роблять її важливим прикладом для розуміння потенціалу алгоритмів ерозії, незважаючи на обмеження в інтерактивності та продуктивності в реальному часі.

1.2.3 QuadSpinner Gaea

QuadSpinner Gaea — це сучасне програмне забезпечення для генерації ландшафту, яке поєднує процедурні та художні робочі процеси для створення реалістичних ландшафтів. Воно розроблене з акцентом на зручність використання, візуальний зворотний зв'язок та ефективність, пропонуючи як генеративні, так і ручні інструменти. Gaea дозволяє користувачам генерувати ландшафти за допомогою процедурних функцій, таких як шум, генерація фракталів та ерозія, а також дозволяє ручне налаштування для більшого творчого контролю. Цей гібридний підхід робить його придатним як для технічних, так і для творчих користувачів, які прагнуть створювати високодеталізований та природний ландшафт.

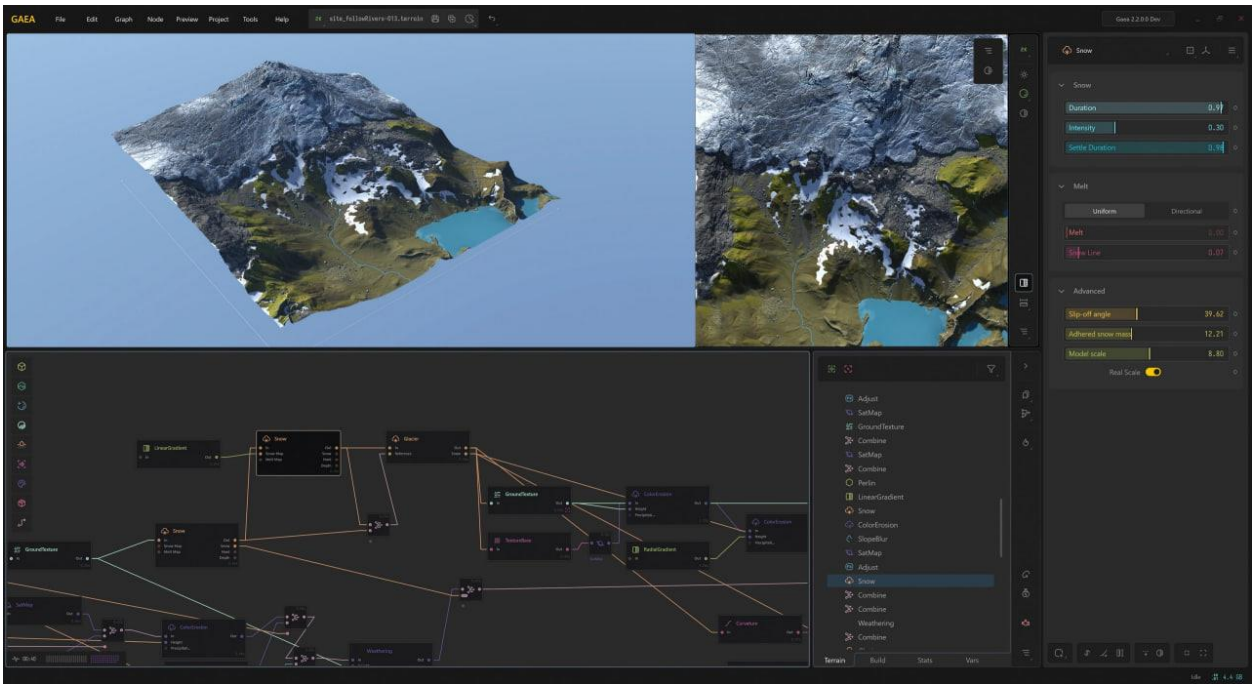


Рисунок 3 — Інтерфейс користувача QuadSpinner Gaea

Інтерфейс (див. Рисунок 3) Gaea інтегрує вікно попереднього перегляду 3D у реальному часі поруч із графом вузлів, забезпечуючи негайний візуальний зворотний зв'язок під час процесу створення ландшафту. Граф вузлів використовується для побудови складних конвеєрів генерації ландшафту, поєднуючи такі операції, як генерація шуму, деформація

ландшафту та різні типи ерозії. Також додаток дозволяє користувачам вручну формувати елементи ландшафту, зберігаючи процедурну гнучкість. Цей дворежимний робочий процес дозволяє ітеративне вдосконалення, поєднуючи сильні сторони процедурної генерації з прямим користувацьким контролем.

Найбільша перевага Gaea полягає в балансі між реалізмом, зручністю використання та продуктивністю. Його передові інструменти для ерозії моделюють геологічні процеси, такі як гідравлічна та теплова ерозія, з високою візуальною точністю, створюючи ландшафти, які виглядають природними та цілісними. Попередній перегляд у реальному часі дозволяє проводити інтерактивні експерименти, роблячи його більш зручним, ніж інші інструменти генерації ландшафту. Gaea також підтримує широкий спектр форматів експорту, що забезпечує інтеграцію у процес розробки ігор, візуальних ефектів та наукової візуалізації.

Незважаючи на свої сильні сторони, Gaea має певні обмеження. Хоча його інтерфейс зручніший для користувача, ніж у інших інструментів генерації ландшафту, він все ще вимагає часу на навчання, щоб повністю опанувати процедурну систему вузлів. Деякі з його розширених функцій, такі як експорт з високою роздільною здатністю або спеціалізовані модулі ерозії, обмежені ліцензіями вищого рівня, що може обмежувати доступність для академічних користувачів. Крім того, хоча Gaea забезпечує попередній перегляд у реальному часі, надзвичайно складні налаштування вузлів або високодетальні симуляції ерозії все ще можуть бути обчислювально вимогливими.

На завершення, QuadSpinner Gaea представляє категорію програм для генерації ландшафту, які успішно поєднують процедурну генерацію з прямими змінами ландшафту від користувача. Акцент на інтерактивності, зворотному зв'язку в режимі реального часу та гібридному управлінні робить його потужним прикладом для розробки програм, спрямованих на балансування алгоритмічної потужності з доступністю для користувача.

1.2.4 Висновки аналізу аналогів

Аналіз існуючих програмних аналогів: Terragen, World Machine та QuadSpinner Gaea — демонструє різноманітність підходів, що використовуються в процедурній генерації ландшафту, та демонструє ключові аспекти, які безпосередньо стосуються розробки поточної програми. Кожен із цих інструментів підкреслює різні пріоритети, від візуального реалізму та якості рендерингу до процедурної гнучкості та інтерактивності користувача.

Terragen демонструє потенціал процедурної генерації ландшафту в поєднанні з розширеним рендерингом та атмосферним моделюванням. Однак його зосередженість на високоякісному офлайн-рендерингу відбувається за рахунок малої інтерактивності та слабкого зворотного зв'язку в режимі реального часу. Це підкреслює важливість розробки системи, яка надає пріоритет швидкості візуалізації в режимі реального часу — важливому аспекту цього проекту, де користувачі можуть динамічно спостерігати за еволюцією ландшафту під час моделювання ерозії.

World Machine надає приклад в процедурному управлінні та реалізмі на основі ерозії. Його складні алгоритми ерозії створюють ландшафти, які точно імітують природні форми рельєфу, доводячи важливість моделювання ерозії для досягнення геологічної достовірності. Тим не менш, відсутність продуктивності в реальному часі та обмежена інтерактивність демонструють необхідність оптимізації алгоритмів ерозії для ефективної роботи на сучасному обладнанні.

QuadSpinner Gaea впроваджує ефективний баланс між процедурною генерацією та художнім контролем завдяки своєму гібридному робочому процесу. Попередній перегляд у реальному часі надає приклад розробки зручного інтерфейсу, який поєднує точність з негайним зворотним зв'язком. Водночас, часткова складність та обчислювальні вимоги Gaea вказують на те, що оптимізована система, зосереджена на основних аспектах процедурної генерації та ерозії, може бути більш доступною для дослідницького та освітнього використання.

Підсумовуючи, аналіз показує, що хоча існуючі програми для генерації рельєфу мають переваги у певних областях – реалістичність візуалізації, точність ерозії або зручність у використанні – жодна з них не інтегрує всі ці аспекти у відкритому середовищі реального часу та з високим рівнем налаштування. Розробляємий застосунок прагне подолати ці недоліки, розробляючи легкий, інтерактивний додаток, який поєднує методи процедурної генерації з моделюванням гідравлічної ерозії в реальному часі. Це об'єднання не лише відповідає сучасним тенденціям процедурної моделювання, але й сприяє дослідженню ефективних, інтерактивних систем генерації рельєфу, придатних як для наукових досліджень, так і для практичного впровадження.

1.3 План робіт

Основною метою є розробка програми, здатної процедурно генерувати рельєф за допомогою кількох методів, а також виконати порівняльний аналіз цих методів з точки зору їх ефективності та технічної продуктивності. Програма використовуватиме OpenGL для візуалізації, дозволяючи глибше досліджувати, як процедурні алгоритми інтегруються з низькорівневими графічними API у середовищі реального часу.

Для досягнення цієї мети був сформульований такий план:

1. Дослідження предметної області — Комплексне дослідження методів процедурної генерації, алгоритмів моделювання рельєфу та існуючих реалізацій з метою розуміння їх переваг, обмежень і практичного застосування.
2. Аналіз і вибір технологій розробки — Оцінка доступних бібліотек, інструментів і API для визначення найбільш підходящого стека технологій для впровадження. В якості платформи візуалізації для візуалізації місцевості буде обрано OpenGL.

3. Створення архітектури системи програмного забезпечення — розробка модульної та масштабованої архітектури програмного забезпечення, яка відокремлює логіку генерації рельєфу від компонентів візуалізації, що забезпечує гнучкість у тестуванні різних методів генерації.
4. Реалізація запланованої архітектури — Розробка програми відповідно до визначеної архітектури, включаючи інтеграцію різних методів генерації рельєфу та візуалізацію в реальному часі за допомогою OpenGL.
5. Перевірка правильності роботи та доступності функціональних вимог — Тестування та перевірка системи програмного забезпечення, щоб переконатися, що вона відповідає функціональним вимогам, працює надійно та забезпечує значущі показники порівняння між різними методами генерації рельєфу.

2 ТЕОРЕТИЧНІ ОСНОВИ ДЛЯ РОЗРОБКИ СИСТЕМ ДЛЯ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ЛАНДШАФТУ

2.1 Загальні відомості про процедурну генерацію ландшафту

Процедурна генерація ландшафтів — це галузь комп'ютерної графіки та моделювання, яка зосереджена на створенні ландшафтів за допомогою алгоритмічних методів, а не ручного моделювання. Метою цих методів є ефективне створення складних, природних середовищ з високим рівнем варіативності. Актуальність процедурної генерації продовжує зростати, оскільки програми все частіше вимагають масштабних, високодеталізованих середовищ, які було б непрактично проектувати вручну. Сучасні відеоігри, системи моделювання, середовища віртуальної реальності, інструменти географічної візуалізації та наукові програми покладаються на процедурні методи для розширення масштабу та реалістичності своїх ландшафтів.

В основі більшості систем процедурної генерації лежить концепція математичної функції, яка перетворює вхідні параметри на значення висоти місцевості. Залежно від обраної функції та структури її параметрів, результуючі ландшафти можуть варіюватися від плавних пагорбів до пересічених гірських регіонів. Одним з найбільш широко використовуваних підходів є генерація ландшафту на основі шуму. Градієнтний шум, такий як шум Перліна, введений Кеном Перліном у 1980-х роках, дозволяє створювати безперервні псевдовипадкові поверхні, що нагадують природні утворення. Шум Перліна є основою для багатьох похідних методів, включаючи фрактальний броунівський рух (fBm), де кілька шарів шуму з різними частотами та амплітудами поєднуються для збільшення візуальної складності. Шум, деформований за ділянками, додатково трансформує вхідну область шумової функції, дозволяючи створювати спотворені та більш хаотичні особливості рельєфу, яких неможливо досягти лише простим нашаруванням. Іншим варіантом є використання абсолютного значення шумових функцій, що

створює характерні хребтоподібні структури. Ці методи є фундаментальними для побудови широкого спектру форм рельєфу та широко використовуються в системах процедурного моделювання [1].

Однак самі по собі шумові функції часто не можуть відтворити складні схеми ерозії та процеси седиментації, які формують природні ландшафти. З цієї причини багато сучасних систем інтегрують методи на основі моделювання, такі як гідравлічна або теплова ерозія. Гідравлічна ерозія моделює, як вода взаємодіє з поверхнею рельєфу, прокладаючи канали, транспортуючи осад та поступово змінюючи форми рельєфу. Навіть спрощені моделі ерозії можуть значно підвищити реалізм, вводячи особливості, які не враховують виключно шумові методи. У системах реального часу або інтерактивних системах ці алгоритми ерозії повинні бути оптимізовані для ефективної роботи, часто на центральному або графічному процесорі, залежно від вимог реалізації. Моделювання ерозії в реальному часі, включене до розробленого застосунку, дотримується цього підходу шляхом ітеративного оновлення сітки рельєфу на основі параметрів потоку води, перенесення осаду та випаровування. Можливість для користувача інтерактивно візуалізувати процес ерозії посилює освітні та експериментальні аспекти системи, дозволяючи водночас точно контролювати регульовані параметри.

На доповнення до алгоритмічної генерації, деякі системи спираються на реальні географічні дані. Набори даних про висоту, отримані зі супутникових або аерофотознімків, можуть бути оброблені та поєднані з процедурними операціями для заповнення відсутніх даних або покращення візуальної насиченості. Гібридні робочі процеси, що поєднують шумовий рельєф, моделювання ерозії та реальні дані про рельєф, стали сучасним стандартом у дослідженнях та виробництві генерації ландшафту. Поєднання різних методів генерації є важливим, оскільки жоден метод не може повністю відтворити складність природних форм рельєфу [7].

Розробляємий додаток інтегрує кілька ключових методів в єдину систему. Завдяки шуму Перліна, шуму fBm, деформації ділянок, перетворень

абсолютного шуму та алгоритму гідравлічної ерозії в реальному часі, система демонструє, як різні процедурні компоненти можуть бути поєднані. Ці методи разом утворюють репрезентативну підмножину сучасних методів генерації процедурних ландшафтів, що використовуються як у застосунках реального часу, так і в робочих процесах моделювання місцевості. Сучасна література також підтримує цей підхід, заснований на комбінаціях, підкреслюючи, що реалістичні ландшафти вимагають як стохастичного створення місцевості, так і моделювання на основі процесів для досягнення достовірних результатів [8].

Теоретичні принципи, описані в цьому розділі, забезпечують основу для розробляємої системи. Вони показують, як математичні моделі, фізичні симуляції та дані реального світу сприяють формуванню ландшафтів.

2.2 Аналіз сучасних технологій розробки генератора ландшафту

Додаток для генерації ландшафту – це програмна система, призначена для алгоритмічного створення та візуалізації цифрових рельєфів. Такі додатки зазвичай включають математичні моделі, функції шуму та процедури моделювання для побудови складних та природних середовищ. В інтерактивних системах механізм рендерингу повинен працювати ефективно, дозволяючи користувачеві налаштовувати параметри та негайно спостерігати за змінами рельєфу. Додаток, розроблений у цій дисертації, дотримується цієї парадигми: він генерує карти рельєфу на основі шуму, застосовує моделювання ерозії в реальному часі та візуалізує ці процеси за допомогою шейдерів OpenGL. З цієї причини вибір відповідних технологій для рендерингу, числових обчислень та розробки інтерфейсу користувача є важливим.

Мова програмування. Основною мовою програмування, обраною для системи, є C++. Вона залишається однією з найефективніших та найширше використовуваних мов для графічних додатків у реальному часі завдяки низькорівневому керуванню пам'яттю, високій продуктивності та сумісності з

графічними API. Оскільки процедурна генерація ландшафту включає безперервне оновлення, обчислення для пікселів у шейдерах та кроки моделювання, що виконуються в кожному кадрі, ефективність, яку забезпечує C++, є важливою для забезпечення стабільної продуктивності. Сучасні спільноти документації та розробників широко визнають C++ основою для графічно інтенсивних систем [9].

Графічний API. OpenGL слугує основним графічним API програми. Він надає програмований конвеєр, придатний для візуалізації карт висот та модифікацій місцевості в реальному часі [27]. Його кросплатформність та широка підтримка в академічних середовищах та середовищах з відкритим кодом роблять його придатним вибором для дослідницьких проектів. OpenGL також бездоганно інтегрується з програмами шейдерів, що дозволяє виконувати операції на основі графічного процесора, які є критично важливими для інтерактивної візуалізації карт шуму та процесів ерозії. Офіційні специфікації OpenGL підкреслюють його гнучкість у реалізації користувацьких робочих процесів візуалізації, що є перевагою для завдань процедурної генерації [10].

Мова графічного програмування. GLSL, мова шейдерів OpenGL, використовується для реалізації вершинних та фрагментних шейдерів, відповідальних за перетворення геометрії та обчислення зовнішнього вигляду поверхні місцевості. GLSL дозволяє переносити ресурсоємні операції на графічний процесор, забезпечуючи оновлення згенерованого ландшафту в реальному часі, коли користувач змінює параметри [26]. Для цієї програми шейдери необхідні для застосування освітлення, візуалізації градієнтів кольору на основі висоти та підтримки динамічних перетворень сітки. Дизайн GLSL як C-подібної мови спрощує інтеграцію з кодом C++ та забезпечує ефективне виконання в конвеєрі візуалізації [11].

Фреймворк користувацького інтерфейсу. Для розробки графічного інтерфейсу користувача використовується Qt Framework. Системи генерації ландшафту часто потребують великої кількості регульованих параметрів,

таких як частоти шуму, сила ерозії, тривалість симуляції та параметри рендерингу. Qt надає надійний набір компонентів інтерфейсу користувача, кросплатформне керування вікнами та зручну інтеграцію з OpenGL за допомогою спеціальних віджетів. Це дозволяє програмі поєднувати складну логіку рендерингу з інтуїтивно зрозумілою панеллю керування [12].

Середовище розробки. QtCreator IDE використовується як основне середовище розробки для програми. Воно пропонує інтегровані інструменти для організації проектів, налагодження, профілювання та дизайну інтерфейсу користувача, що робить його придатним для управління графічно-орієнтованим проектом на C++. Вбудована підтримка CMake та qmake у поєднанні з безшовною інтеграцією компонентів інтерфейсу Qt та модулів OpenGL спрощує робочий процес розробки та зменшує часові витрати на налаштування. Структурований інтерфейс QtCreator дозволяє ефективно орієнтуватися у великих кодових базах, що особливо корисно в проектах, які поєднують логіку рендерингу, алгоритми моделювання та компоненти інтерфейсу.

Математична бібліотека. Для математичних операцій, таких як векторні перетворення, матриці, інтерполяція та розрахунки шуму, програма спирається на бібліотеку GLM (OpenGL Mathematics). GLM розроблена для відображення структури та умовностей GLSL, що робить її особливо придатною для графічного програмування. Вона забезпечує чистий та оптимізований інтерфейс для операцій лінійної алгебри, забезпечуючи узгодженість між обчисленнями на стороні процесора та логікою шейдерів на стороні графічного процесора, що є важливим під час генерації ландшафту [13].

Загалом, вибір C++, OpenGL, GLSL, Qt Framework, QtCreator та GLM формує цілісну та ефективну технологічну основу для створення інтерактивної системи генерації ландшафту. Ці технології забезпечують високопродуктивний рендеринг у реальному часі, модульну структуру коду та адаптивний інтерфейс, придатний для експериментів з процедурними

алгоритмами. Їх широке впровадження в галузі та академічній сфері ще більше підтверджує їх використання в дослідницьких застосунках.

3 РОЗРОБКА ГЕНЕРАТОРА ЛАНДШАФТУ У РЕАЛЬНОМУ ЧАСІ

3.1 Опис предметної області

Предметна область дипломної роботи охоплює перетин комп'ютерної графіки, обчислювальної геометрії та моделювання природних процесів. Основна мета – створення складних, реалістичних ландшафтів за допомогою алгоритмічних методів, а не ручного моделювання. Цей підхід став важливим компонентом сучасних інтерактивних додатків, особливо в системах реального часу, таких як відеоігри, віртуальні симуляції та середовища наукової візуалізації [28]. Основна перевага процедурних методів полягає в їхній здатності створювати різноманітні та деталізовані рельєфи з мінімальними вимогами до зберігання, а також дозволяти динамічну модифікацію ландшафтних елементів у відповідь на введення користувача або фізичне моделювання.

У цій предметній області алгоритми на основі шуму відіграють фундаментальну роль. Такі технології, як шум Перліна, фрактальний броунівський рух, деформація ділянок та варіанти шуму за абсолютними значеннями, формують основу для створення карт висот, які імітують природну нерівність реальних ландшафтів. Ці методи дозволяють розробникам контролювати структуру, частоту та амплітуду елементів рельєфу, що дозволяє моделювати все: від гладких пагорбів до різко еродованих гірських хребтів. Однак, рельєфу, керованому виключно шумом, часто бракує реалізму з точки зору геологічних процесів, що призвело до зростаючого використання моделей моделювання ерозії. Гідравлічна ерозія, зокрема, вносить ефекти потоку води, перенесення осаду та ітеративного перетворення рельєфу, що призводить до ландшафтів, які демонструють більш природні візерунки та утворення. Інтеграція шумових функцій з алгоритмами ерозії відображає сучасний стан досліджень, в яких гібридні процедурні

підходи спрямовані на досягнення як ефективності, так і візуальної автентичності.

Предметна галузь також підкреслює важливість обчислень у реальному часі. Генерація та модифікація рельєфу в реальному часі вимагають оптимізованих алгоритмів та ефективного управління пам'яттю. Програми, які працюють з цими обмеженнями, повинні балансувати візуальну якість з обчислювальною продуктивністю, особливо коли дозволяють користувачам інтерактивно налаштовувати параметри, спостерігати за симуляціями або перемикатися між різними методами генерації. Це вводить додатковий рівень складності, який безпосередньо впливає на розробку інструментів та інтерфейсів, що використовуються для дослідження процедурних ландшафтів.

Розробка представленого застосунку мотивована необхідністю дослідити та порівняти різні підходи до генерації рельєфу в контрольованому, інтерактивному середовищі. Завдяки включенню кількох алгоритмів на основі шуму та моделі гідравлічної ерозії в реальному часі, застосунок дозволяє детально проаналізувати, як різні методи впливають на результуючу структуру рельєфу. Він забезпечує експериментальну платформу для вивчення переваг та обмежень кожного підходу, оцінки їхніх обчислювальних властивостей та дослідження потенційних удосконалень або гібридних методів. Такі дослідження є актуальними як для академічного розуміння, так і для практичних застосувань, що вимагають ефективного, але реалістичного ландшафту.

На завершення, предметна область процедурної генерації ландшафту охоплює алгоритмічне моделювання, моделювання природних явищ та візуалізацію в режимі реального часу. Створений додаток робить внесок у цю галузь, пропонуючи гнучкий інструмент для вивчення методів створення ландшафту, порівняння їх характеристик та аналізу процесів ерозії в міру їх розвитку. Його інтеграція кількох процедурних моделей та інтерактивність у режимі реального часу відповідає сучасним тенденціям у цій галузі та

підтримує подальші дослідження з розробки більш точних та ефективних методів генерації ландшафту.

3.2 Архітектура системи

Архітектура генератора ландшафту у реальному часі базується на шаблоні проектування Model–View–Controller (MVC). MVC – це широко використовуваний архітектурний підхід в інтерактивних та графічних додатках, оскільки він розділяє обробку даних, рендеринг та взаємодію з користувачем на незалежні компоненти. Такий поділ відповідає цілям цього проекту, забезпечуючи модульність, спрощуючи обслуговування та дозволяючи окрему розробку логіки генерації рельєфу та функцій інтерфейсу користувача [30]. Шаблон MVC сприяє чіткій організації коду та покращує масштабованість додатків, які передбачають безперервну взаємодію з користувачем та оновлення даних у режимі реального часу [14]. На Рисунок 4 — *UML діаграма архітектури застосунка* зображені зв'язки між компонентами Моделі, Представлення та Контролера, а також потік даних та керування ними.

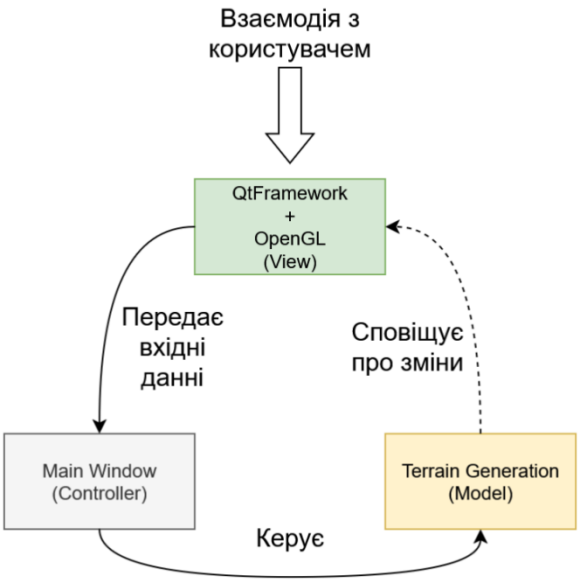


Рисунок 4 — *UML діаграма архітектури застосунка*

Модель у цій системі слугує основним обчислювальним модулем. Вона реалізує всі алгоритми, відповідальні за генерацію рельєфу, включаючи шум Перліна, фрактальний броунівський рух, шум деформації ділянок, шум абсолютного значення та моделювання гідравлічної ерозії. Цей компонент працює незалежно від графічного інтерфейсу та зберігає структури даних, що представляють карту висот. Його конструкція гарантує, що процеси генерації рельєфу та ерозії можна змінювати, розширювати або оптимізувати, не впливаючи на інші частини програми.

Компонент представлення – це інтерфейс користувача Qt Framework разом з інтегрованим конвеєром рендерингу OpenGL. Qt надає вікна, панелі, елементи керування параметрами та інтерактивні віджети, за допомогою яких користувач налаштовує параметри генерації рельєфу. OpenGL обробляє рендеринг згенерованих карт висот та моделювання ерозії в реальному часі. Таке розділення гарантує, що візуальне представлення точно відображає поточний стан Моделі, залишаючись гнучким для майбутніх удосконалень методів рендерингу або структури інтерфейсу користувача.

Контролер реалізовано як головний клас вікна, який координує всі взаємодії між Моделлю та Представленням. Він обробляє дії користувача, оновлює значення параметрів, запускає процедури генерації або ерозії та гарантує, що Представлення отримує оновлені дані щоразу, коли відбуваються зміни. Контролер інкапсулює логіку програми та діє як посередник, який підтримує узгодженість між обчислювальними процесами та графічним виводом.

На завершення, архітектура MVC забезпечує чітку та структуровану основу для розробки генератора ландшафту в реальному часі [24]. Її модульна природа підтримує гнучке експериментування з різними методами генерації ландшафту та сприяє майбутнім розширенням системи. Вибір архітектури відповідає найкращим практикам, що потребують інтеграції обчислювальних модулів, графічних інтерфейсів та взаємодії в реальному часі.

3.3 Алгоритми генерації шуму

3.3.1 Шум Перліна

Шум Перліна є одним з фундаментальних алгоритмів, що використовуються для процедурної генерації рельєфу завдяки його здатності створювати плавні, природні градієнти, що імітують особливості реальних ландшафтів. Розроблений Кеном Перліном, цей метод генерує шум шляхом змішування псевдовипадкових градієнтних векторів на сітці, створюючи безперервні варіації, ідеальні для побудови карт висот та інших процедурних текстур.

У реалізованому генераторі рельєфу шум Перліна служить основним механізмом для створення початкового розподілу висоти перед застосуванням подальших перетворень або операцій ерозії. Процес починається з побудови сітки градієнтних векторів за допомогою функції `generateVectorGrid`, наведеної у Лістинг 1. Ця функція визначає напрямок векторів в кожній точці сітки, а її детермінована генерація на основі заданного числа забезпечує відтворюваність рельєфу.

Лістинг 1 – *функція "generateVectorGrid"*

```
std::vector<std::vector<glm::vec2>>
TerrainGenerator::generateVectorGrid(int seed, int gridCount)
{
    std::vector<std::vector<glm::vec2>>
gradientVectorGrid(gridCount + 1,
std::vector<glm::vec2>(gridCount + 1));

    for(int i = 0; i < gridCount + 1; i++)
    {
        for(int j = 0; j < gridCount + 1; j++)
        {
            float random = c_utils::randomNum(i * seed, j *
seed);

            glm::vec2 vector(sin(random), cos(random));
            gradientVectorGrid[i][j] = vector;
        }
```

```
    }

    return gradientVectorGrid;
}
```

Після генерації градієнтної сітки алгоритм переходить до обчислення значень шуму для кожної точки карти висот. Це робиться всередині циклу, який перебирає кількість шарів, визначених для фрактального броунівського руху. Коли використовується лише один шар, отриманий рельєф відповідає класичному шуму Перліна. Коли застосовується кілька шарів, кожен з яких масштабується за амплітудою та частотою відповідно до стандартних правил fBm, кінцева карта висот демонструє багатшу деталізацію та складніші топологічні варіації. Основне обчислення значення шуму для однієї точки виконується функцією `perlinNoise`, наведеної у Лістинг 2, яка обчислює значення шуму для заданих координат.

Лістинг 2 – *функція "perlinNoise"*

```
float TerrainGenerator::perlinNoise(int x, int y,
    const std::vector<std::vector<glm::vec2>> &grid)
{
    int localGridSize = (m_noiseParams.gridCount *
m_noiseParams.gridSize) / (grid.size() - 1);
    int locX = x / localGridSize;
    int locY = y / localGridSize;
    int gx = locX * localGridSize;
    int gy = locY * localGridSize;
    auto dot = [&](int ix, int iy, float dx, float dy) {
        const auto &g = grid[ix][iy];
        return dx * g.x + dy * g.y;
    };
    float dx = float(x - gx) / localGridSize;
    float dy = float(y - gy) / localGridSize;
    float tl = dot(locX, locY, dx, dy);
    float tr = dot(locX + 1, locY, dx - 1.0f, dy);
    float br = dot(locX + 1, locY + 1, dx - 1.0f, dy - 1.0f);
    float bl = dot(locX, locY + 1, dx, dy - 1.0f);
    float u = c_utils::fade(dx);
    float v = c_utils::fade(dy);
    float ix0 = c_utils::lerp(tl, tr, u);
```

```
    float ix1 = c_utils::lerp(bl, br, u);
    return c_utils::lerp(ix0, ix1, v) * 1.5f;
}
```

Лістинг 3 демонструє, як генерується повна карта висот функцією perlinNoiseHeightMap, яка викликає генератор градієнтної сітки, виконує ітеративні обчислення, накопичує значення шуму в локальну матрицю та повертає завершену карту висот після обробки всіх шарів.

Лістинг 3 – *функція "perlinNoiseHeightMap"*

```
std::vector<std::vector<float>>
TerrainGenerator::perlinNoiseHeightMap(int seed)
{
    int mapSize = m_noiseParams.gridCount *
m_noiseParams.gridSize;
    std::vector<std::vector<float>> localHeightMap(mapSize,
std::vector<float>(mapSize, 0.0f));
    float frequency = 1.0f;
    float amp = 1.0f;
    for(int l = 0; l < m_noiseParams.layersCount; l++)
    {
        int localGridCount = m_noiseParams.gridCount *
frequency;
        int localGridSize = m_noiseParams.gridSize / frequency;
        std::vector<std::vector<glm::vec2>> vectorGrid =
generateVectorGrid(seed, localGridCount);
        for(int i = 0; i < localGridCount * localGridSize; i++)
        {
            for(int j = 0; j < localGridCount * localGridSize;
j++)
            {
                float noiseFactor = perlinNoise(i, j,
vectorGrid);
                noiseFactor = std::clamp((noiseFactor + 1.0f) *
0.5f, 0.0f, 1.0f);
                localHeightMap[i][j] =
c_utils::lerp(localHeightMap[i][j], noiseFactor, amp);
            }
        }
        frequency = frequency *
m_noiseParams.layerFrequencyFactor;
        amp = amp / m_noiseParams.layerAmplitubeFactor;
    }
    return localHeightMap;
```

```
}
```

Результатом (див. Рисунок 5) цього алгоритму є гладка, безперервна структура рельєфу, яка формує основу для подальших перетворень шуму або моделювання ерозії.

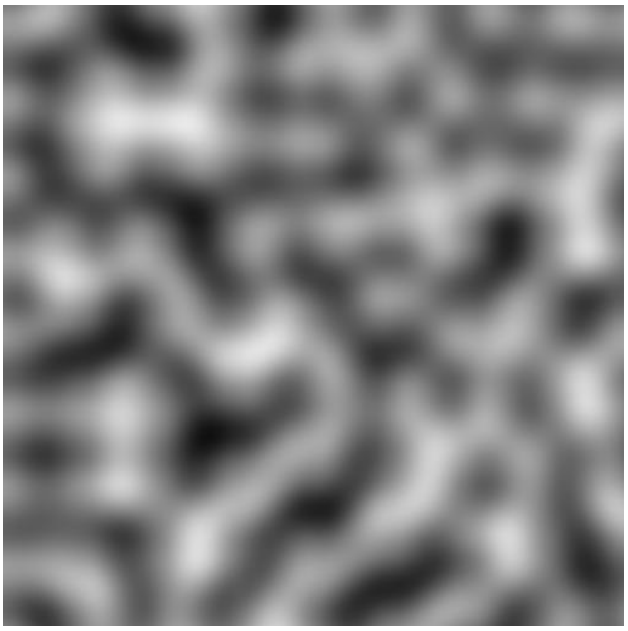


Рисунок 5 — *Результат шуму Перліна*

Підсумовуючи, реалізація шуму Перліна забезпечує основу для генерації рельєфу, підтримуючи як класичний одношаровий шум, так і багатошарові варіації fBm. Його детермінована побудова сітки, плавна інтерполяція та сумісність з додатковими модулями обробки роблять його придатним та надійним компонентом генератора рельєфу.

3.3.2 Фрактальний броунівський рух

Фрактальний броунівський рух (fBm) розширює класичний шум Перліна, об'єднуючи кілька шарів шуму (див. Рисунок 6) в одну, візуально складнішу карту висот. Кожен шар, який часто називають октавою, додає додаткової деталізації залежно від своєї частоти та амплітуди. Низкочастотні шари встановлюють великі, широкі особливості рельєфу, тоді як

високочастотні шари вводять дрібні варіації, такі як хребти або ледь помітні нерівності поверхні. Комбінуючи ці шари зі зменшенням амплітуди та збільшенням частоти, отриманий рельєф набуває структурного різноманіття, якого неможливо досягти за один прохід лише шуму Перліна. Цей метод широко використовується в процедурній генерації завдяки своєму природному вигляду та здатності імітувати фрактальні візерунки, що спостерігаються в горах, пагорбах та ерозійно утворених ландшафтах.

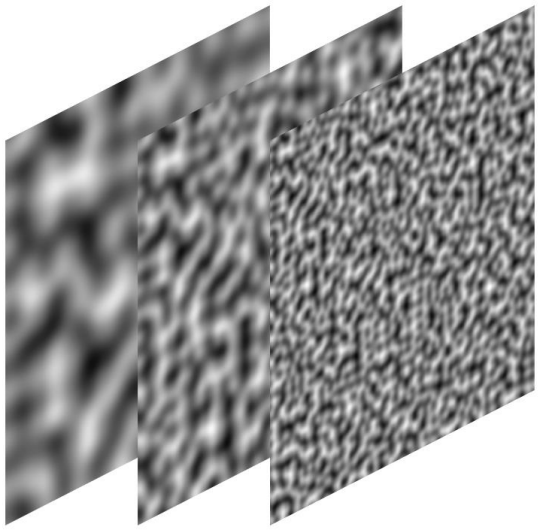


Рисунок 6 — *Ілюстрація поєднання шарів fBm*

У реалізованому генераторі рельєфу fBm не є окремим алгоритмом, а скоріше структурованим розширенням генерації шуму Перліна, описаного в попередньому розділі. Карта висот будується за допомогою циклу, який ітерується по кількох шарах шуму. Для кожного шару алгоритм оцінює функцію шуму Перліна в масштабованих координатах та зі зміненою роздільною здатністю сітки. Отримані значення шуму накопичуються в проміжну карту висот, де внесок кожного шару зважується відповідно до правил амплітуди, визначених теорією fBm. Частота збільшується для кожного наступного шару, що стискає шумову картину та вводить деталі вищих частот, тоді як амплітуда зменшується, щоб гарантувати, що вищі шари надають внесок у уточнення, а не перезавантажують рельєф.

Кінцевий результат, отриманий в результаті шаруватого підсумовування, показано на Рисунок 7. Цей шум створює більш складний та візуально привабливий рельєф порівняно з одношаровим шумом Перліна, зберігаючи при цьому переходи та природні варіації схилів.

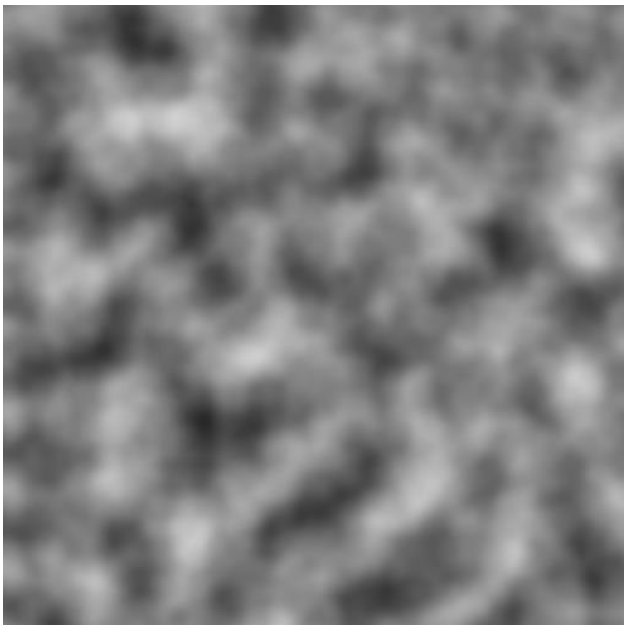


Рисунок 7 — Результат fBm шуму

Завдяки цій стратегії нашарування, fBm забезпечує гнучку структуру, яка дозволяє користувачеві контролювати рівень деталізації та складності згенерованого рельєфу. У контексті генератора рельєфу цей метод діє як основа, яку можна додатково покращити шляхом деформації ділянок або ерозії, що дозволяє створювати широкий спектр ландшафтів, придатних для моделювання, візуалізації або подальших експериментів.

3.3.3 Шум деформованих ділянок

Деформація ділянок – це техніка, яка використовується для збільшення візуальної складності та природної варіації процедурно згенерованого рельєфу шляхом зміни вхідних координат базової функції шуму [15]. Замість того, щоб вибірково контролювати шум Перліна безпосередньо за регулярними координатами сітки, алгоритм обчислює два допоміжні поля шуму, які

служать картами зміщення. Одне з цих полів контролює зміну координати *X*, а інше – зміну координати *Y*. Зміщуючи позиції вибірки відповідно до цих додаткових значень шуму, результуючий рельєф демонструє закручені, скручені та дуже нерегулярні структури, які неможливо створити лише за допомогою стандартних методів Перліна або fBm. Цей підхід є ефективним методом введення нелінійної просторової варіації у функції шуму.

У реалізованому генераторі рельєфу шум деформації ділянок створюється функцією `perlinNoiseDomainWarpedHeightMap`, наведеної у Лістинг 4. Алгоритм спочатку генерує дві незалежні карти висот, використовуючи шум Перліна, кожна з яких представляє поле зміщення. Одна карта зміщення використовується для деформації координати *X*, а інша — для деформації координати *Y*. Обидві карти масштабуються за допомогою визначеної користувачем ваги деформації, яка визначає інтенсивність ефекту спотворення. Після обчислення цих значень зміщення алгоритм змінює координати вибірки перед викликом стандартної функції `perlinNoise`. Деформовані координати потім використовуються для генерації кожного зразка кінцевої карти висот, створюючи місцевість з деформацією схожою на тектонічний рух поверхні землі.

Лістинг 4 – *функція "perlinNoiseDomainWarpedHeightMap"*

```
std::vector<std::vector<float>>
TerrainGenerator::perlinNoiseDomainWarpedHeightMap(int seed,
const float domainWarpWeight)
{
    std::vector<std::vector<float>> xDomainWarpMap =
perlinNoiseHeightMap(2.0f);
    std::vector<std::vector<float>> yDomainWarpMap =
perlinNoiseHeightMap(4.0f);
    float frequency = 1.0f;
    float amp = 1.0f;
    int mapSize = m_noiseParams.gridCount *
m_noiseParams.gridSize;
    std::vector<std::vector<float>> localHeightMap(mapSize,
std::vector<float>(mapSize, 0.0f));
    for(int l = 0; l < m_noiseParams.layersCount; l++)
    {
```

```
        int localGridCount = m_noiseParams.gridCount *
frequency;
        int localGridSize = m_noiseParams.gridSize / frequency;
        std::vector<std::vector<glm::vec2>> vectorGrid =
generateVectorGrid(seed, localGridCount);
        for(int i = 0; i < localGridCount * localGridSize; i++)
        {
            for(int j = 0; j < localGridCount * localGridSize;
j++)
            {
                int warpedX = std::round(i +
((((xDomainWarpMap[i][j] * 2.0f) - 1.0f) * localGridSize) *
domainWarpWeight));
                warpedX = std::clamp(warpedX, 0, (localGridCount
* localGridSize) - 1);
                int warpedY = std::round(j +
((((yDomainWarpMap[i][j] * 2.0f) - 1.0f) * localGridSize) *
domainWarpWeight));
                warpedY = std::clamp(warpedY, 0, (localGridCount
* localGridSize) - 1);
                float noiseFactor = perlinNoise(warpedX,
warpedY, vectorGrid);
                noiseFactor = std::clamp((noiseFactor + 1.0f) *
0.5f, 0.0f, 1.0f);
                localHeightMap[i][j] =
c_utils::lerp(localHeightMap[i][j], noiseFactor, amp);
            }
        }
        frequency = frequency *
m_noiseParams.layerFrequencyFactor;
        amp = amp / m_noiseParams.layerAmplitubeFactor;
    }
    return localHeightMap;
}
```

Після завершення процедури деформації та вибірки функція виводить карту висот (див. Рисунок 8 — *Результат шуму з деформацією ділянок*), яка візуально відрізняється від класичного шуму Перліна та fBm. Ефект деформації порушує передбачувану градієнтну структуру стандартного шуму та вводить складні візерунки, що нагадують геологічну складчастість.

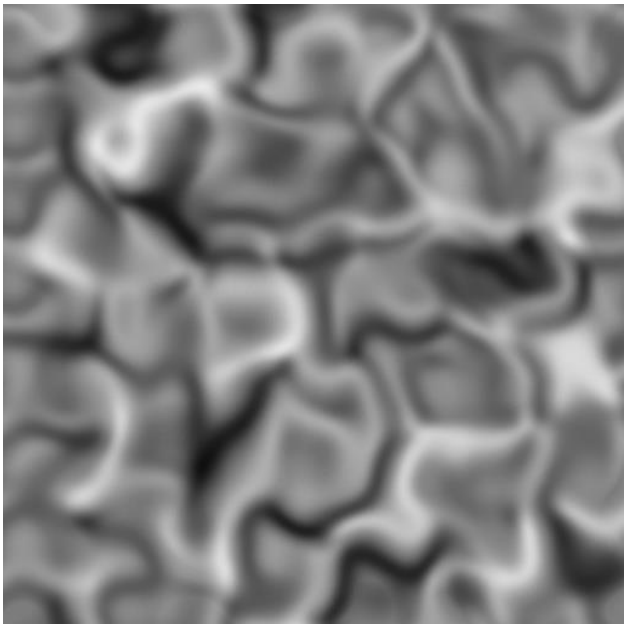


Рисунок 8 — *Результат шуму з деформацією ділянок*

Деформація ділянок забезпечує ефективний спосіб покращення різноманітності рельєфу без значного збільшення складності алгоритму. Це особливо ефективно в поєднанні з fBm або при використанні як вхідних даних для подальшого моделювання ерозії, де деформована топографія призводить до більш складних та реалістичних результатів.

3.3.4 Абсолютний шум

Абсолютний шум – це проста, але ефективна модифікація класичного шуму Перліна, яка перетворює діапазон вихідних значень з [-1, 1] у невід'ємний інтервал [0, 1]. Це перетворення досягається шляхом взяття абсолютного значення функції шуму. В результаті згенерований рельєф демонструє характерний візерунок, що характеризується хребтоподібними структурами, оскільки від'ємні значення, які зазвичай утворюють западини, відбиваються вгору. Цей підхід часто використовується в процедурній генерації текстур та моделюванні рельєфу, коли потрібні гострі хребти або симетричні варіації висоти.

У реалізованому генераторі рельєфу функція, відповідальна за створення цього типу карти висот, – це

`perlinNoiseAbsoluteHeightMap`. Внутрішньо він дотримується того ж робочого процесу, що й звичайний генератор шуму Перліна: будується градієнтна векторна сітка, координати обробляються шар за шаром, якщо ввімкнено fBm, а значення шуму Перліна обчислюються за допомогою стандартної функції. Єдина відмінність полягає в тому, як перетворюється результуюче значення шуму. Замість того, щоб записувати необроблений результат шуму Перліна на карту висот, алгоритм перетворює його за допомогою виразу абсолютного значення, як це показано у Лістинг 5.

Лістинг 5 – *Фрагмент перетворення до абсолютного шуму з функції perlinNoiseAbsoluteHeightMap*

```
for(int i = 0; i < localGridCount * localGridSize; i++)
        {
            for(int j = 0; j < localGridCount * localGridSize; 
j++)
            {
                float noiseFactor = perlinNoise(i, j, vectorGrid);

                noiseFactor  =  std::clamp(std::abs(noiseFactor), 
0.0f, 1.0f);

                localHeightMap[i][j]                                =
c_utils::lerp(localHeightMap[i][j], noiseFactor, amp);
            }
        }
```

Ця формула гарантує, що кожен зразок вносить невід'ємне значення висоти. При застосуванні до всієї місцевості ця операція створює більш виражені гребені та усуває глибокі западини, типові для немодифікованого шуму Перліна. Оскільки алгоритм зберігає всі інші кроки генерації шуму, він підтримує як одношаровий абсолютний шум, так і багатошаровий абсолютний fBm, що дозволяє створювати високодетальні гребеноподібні утворення місцевості.

Отримана карта висот (див. Рисунок 9) демонструє характерну дзеркальну структуру абсолютного шуму, де спочатку негативні області однаково впливають на висоту місцевості.

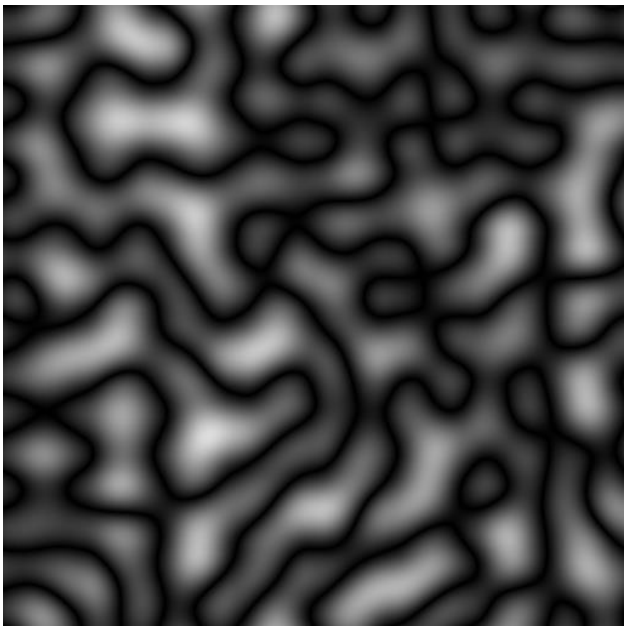


Рисунок 9 — Результат абсолютного шуму

Застосовуючи мінімальну математичну модифікацію до вихідного шуму Перліна, цей метод створює новий вид ландшафту, який можна використовувати самостійно або в поєднанні з ерозією для створення гороподібних утворень з гострими гребенями.

3.4 Симуляція ерозії

3.4.1 Теоретичні основи гідравлічної ерозії

Гідравлічна ерозія – це метод, який моделює поступову трансформацію рельєфу, спричинену рухом води по його поверхні [16]. У природному середовищі дощ та річковий потік поступово змінюють рельєф розчиняючи, переносячи та відкладаючи ґрунт у нижчих регіонах ландшафту. Цей процес створює характерні геологічні утворення, такі як канали, русла річок та гірські схили. Моделювання цих явищ на комп'ютерно-згенерованому рельєфі забезпечує засоби створення ландшафтів, які виглядають більш реалістичними та геологічно цілісними, ніж ті, що створюються виключно за допомогою алгоритмів на основі шуму.

Фундаментальний принцип гідравлічної ерозії базується на взаємодії між водою та осадом [29]. Типова модель припускає, що кожна точка рельєфу може утримувати певну кількість води, яка потім рухається у відповідь до градієнту висоти поверхні. Коли вода перетинає рельєф, вона збирає осад у районах з високим схилом або високою швидкістю води. Ця ємність осадів залежить від таких факторів, як швидкість, місцевий нахил та кількість присутньої води. Коли вода сповільнюється або зустрічається з більш рівною місцевістю, вона більше не може витримувати навантаження осадів, що призводить до їх відкладення. Протягом багатьох ітерацій цей цикл ерозії та відкладень змінює форму місцевості, поглиблюючи канали, згладжуючи схили та формуючи осадові структури.

Ще один компонент теорії гідравлічної ерозії включає моделювання випаровування, яке зменшує кількість води, що утримується на поверхні з часом. Коли вода випаровується, будь-який осад, що все ще знаходиться в ній, осідає на місцевості. Це явище сприяє поступовій стабілізації поверхні після припинення потоку води. Важливими аспектами теорії є вимога до ретельного налаштування таких параметрів, як швидкість ерозії, ємність осадових порід, швидкість випаровування та швидкість перенесення води, щоб забезпечити реалістичні та візуально узгоджені результати.

Моделі гідравлічної ерозії були широко досліджені в дослідженнях, пов'язаних з моделюванням природних явищ та процедурною генерацією місцевості. Вони забезпечують фізично натхненну основу, яка підвищує реалістичність синтетичних ландшафтів та допомагає подолати розрив між процедурними шумовими функціями та фактичними геологічними процесами. Фізично засновані та гібридні системи ерозії можуть створювати структури ерозії зі значно більшою структурною правдоподібністю порівняно з чисто алгоритмічними методами генерації місцевості [17].

3.4.2 Реалізація алгоритму гідравлічної ерозії

Реалізація алгоритму гідравлічної ерозії в застосунку відповідає моделі симуляції на основі крапель, що виконується на процесорі. Щоб уникнути блокування головного потоку, відповідального за рендеринг та взаємодію з користувачем, процес ерозії починається зі створення окремого потоку процесора. У цьому потоці функція `erosionSimulation` викликається неодноразово протягом заданої користувачем кількості ітерацій, які представляють кількість крапель для симуляції. Наприклад, якщо користувач вводить тисячу ітерацій, функція виконується десять разів, кожне виконання імітує сто крапель.

У контексті гідравлічної ерозії термін «ітерації» стосується кількості крапель, що моделюються під час процесу ерозії, де кожна крапля являє собою незалежну частинку, що рухається по місцевості, розмиваючи та відкладаючи ґрунт вздовж свого шляху. Налаштування дозволяють контролювати вплив окремої краплі на місцевість. Збільшуючи такі параметри, як радіус ерозії, накопичувальна здатність або гравітація, одна крапля може імітувати ефект багатьох крапель, що дозволяє створювати правдоподібні результати навіть за відносно невеликої кількості ітерацій. Зниження цих значень призводить до фізично реалістичного моделювання, в якому кожна крапля поводиться незалежно та впливає лише на локальну область місцевості.

Сама симуляція ерозії структурована навколо концепції часу життя краплі. Для кожної краплі алгоритм починає з вибору випадкової початкової позиції на карті висот. Потім крапля перетинає рельєф місцевості відповідно до інформації про локальний схил. Для обчислення напрямку використовується карта градієнта. Замість попереднього обчислення всієї сітки градієнта, система обчислює значення градієнта коли крапля досягає точки, для якої градієнт ще не обчислено, алгоритм отримує його на вимогу та зберігає. Такий підхід значно зменшує обчислювальні витрати, оскільки обраховуються лише ті градієнти, які дійсно потрібні.

На кожному кроці моделювання крапля оцінює крутизну напрямку свого руху, обчислюючи різницю висот між її поточним положенням та наступним. Це значення разом зі швидкістю краплі визначає її максимальну ємність для утворення осаду, для обчислення запропонована наступна формула 3.1.

$$SedimentCapacity = V \times h\Delta \times CapacityFactor, \quad (3.1)$$

де *SedimentCapacity* – ємність краплі, *V* – швидкість краплі, *h*Δ – різниця у висоті між наступною та поточною позицією, *CapacityFactor* – множник формули для контролю результатів від користувача.

Якщо крапля наразі переносить більше осаду, ніж її максимальна ємність, вона відкладає частину цього осаду на рельєф. Якщо ж крапля переносить менше осаду, ніж максимальна ємність, крапля розмиває поверхню.

Ключовим компонентом у контролі ерозії є швидкість краплі, яка впливає як на захоплення, так і на перенесення осаду. Швидкість оновлюється на кожному кроці моделювання відповідно до запропонованої формули 3.2.

$$V = \sqrt{V_{prev}{}^2 + h\Delta \times g}, \quad (3.2)$$

де *V* – це швидкість, V_{prev} – це швидкість у попередній точці, *h*Δ – різниця у висоті між наступною та поточною позицією, g – це гравітація.

Після того, як нова швидкість відома, кількість ґрунту, що розмивається з рельєфу, визначається за допомогою запропонованої формули 3.3.

$$E = V \times h\Delta \times ErosionFactor, \quad (3.3)$$

де E – це кількість ґрунту, яку потрібно вимити, V – це швидкість, $h\Delta$ – різниця у висоті між наступною та поточною позицією, *ErosionFactor* – це множник формули для контролю результатів від користувача.

Після обчислення ерозії карта висот відповідно змінюється, а кількість осаду, який крапля утримує, збільшується на обраховану величину. У цій реалізації модифікація рельєфу додатково формується спеціалізованим класом `TerrainBrush`, який визначає, як розподіляються ефекти ерозії та осадження між сусідніми комірками. Клас складається з попередньо обчисленого вектора `BrushPoints`, кожен з яких визначається зміщенням X та Y від центрального положення краплі та відповідним значенням ваги. Коли застосовується ерозія або осадження, ці зважені зміщення забезпечують згладжування та масштабування модифікації рельєфу відповідно до заданих користувачем параметрів радіуса, забезпечуючи точніший контроль над тим, наскільки сильно рельєф реагує на кожну краплю.

Якщо крапля досягає точки, де немає шляху спуску, або якщо її життєвий час закінчується, вона відкладає залишки осаду, перш ніж бути видаленою. Цей процес повторюється для кожної краплі на всіх ітераціях, поступово змінюючи рельєф, щоб відобразити кумулятивний вплив гідравлічної ерозії.

3.5 Модуль візуалізації

Модуль візуалізації програми являє собою комплексну підсистему, призначену для візуалізації процедурно згенерованих ландшафтів у реальному часі. Його дизайн відповідає модульній та розширюваній архітектурі, в якій набір спеціалізованих класів співпрацює для створення цілісної 3D-сцени [18]. Ці компоненти керують ресурсами графічного процесора, організовують геометричні дані, налаштовують шейдери, обробляють освітлення та взаємодію з камерою, а також взаємодіють з користувачем через вбудовані віджети Qt. Структура цієї підсистеми проілюстрована на Рисунок 10.

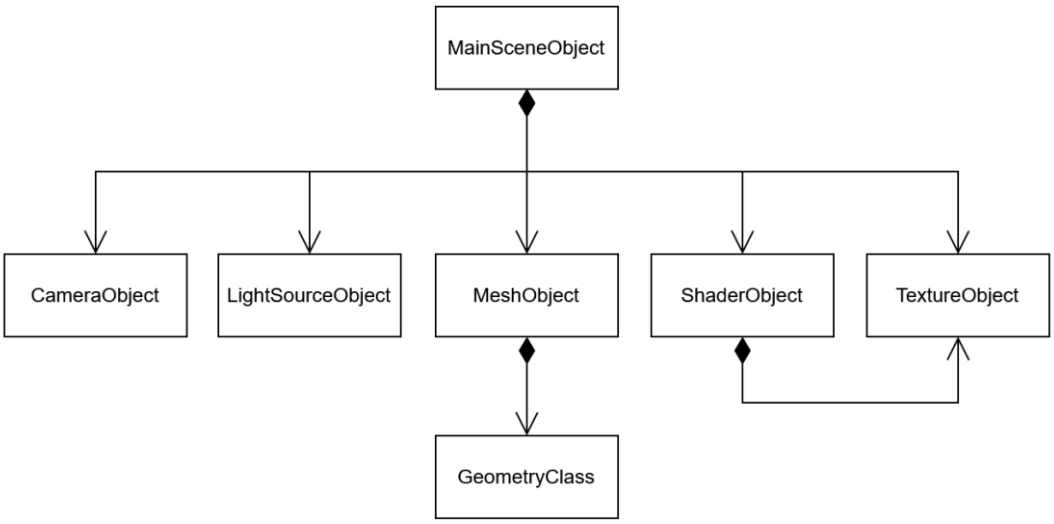


Рисунок 10 — *Діаграма класів модулю візуалізації*

Віджет Qt, `SceneViewPort` знаходиться між графічним інтерфейсом та механізмом візуалізації, і відповідальний за ініціалізацію та підтримку контексту OpenGL. Він служить основним середовищем, де відбуваються операції рендерингу, і гарантує, що взаємодія користувача, така як рух камери, зміна розміру області перегляду та оновлення, керовані інтерфейсом користувача, належним чином відображаються у 3D-сцені. `SceneViewPort` використовує `MainSceneObject`, який функціонує як центральний контролер візуалізації. У цій ролі `MainSceneObject` зберігає посилання на поточну геометрію рельєфу, список доступних шейдерів, завантажені у пам’ять текстури, джерело світла та камеру. Він ефективно керує взаємодією між усіма елементами візуалізації, забезпечуючи послідовну синхронізацію геометрії, освітлення та параметрів шейдерів.

Значна частина візуальної конфігурації рельєфу керується підсистемою шейдерів. Цю логіку імплементує клас `ShaderObject`, який інкапсулює завантаження, компіляцію та зв'язування шейдерних програм GLSL. Окрім складання етапів шейдерів, він забезпечує структурований доступ до змінних шейдера, дозволяючи програмі динамічно оновлювати матриці перетворення, налаштування освітлення та параметри специфічні для рельєфу [19]. Ця гнучкість є важливою в середовищі процедурної генерації, де геометрія та

характеристики затінення можуть швидко змінюватися, коли користувач налаштовує функції шуму, налаштування ерозії або стилі візуалізації. У комбінації з цим класом працює `TextureObject`, який перетворює необроблені дані зображення на текстури OpenGL. Це гарантує, що текстури, такі як карти висот, карти нормалей та допоміжні візуальні зображення, ефективно передаються на графічний процесор, правильно форматуються та доступні для шейдерних програм.

Освітлення значною мірою сприяє реалістичності відображення рельєфу. `LightSourceObject` інкапсулює такі властивості, як положення світла, колір та інтенсивність. Ці значення передаються до шейдерних програм, що дозволяє системі візуалізації обчислювати взаємодію освітлення по всій поверхні рельєфу. Це робить рельєфні особливості та деталі ерозії більш помітними, тим самим підвищуючи аналітичну корисність візуального зображення. Доповненням до освітлення є `CameraObject`, який керує перспективою користувача в 3D-сцені. Камера підтримує вільну навігацію та орбітальний режим для огляду ландшафту. Вона також містить систему перемикання між ортографічним та ізометричним режимами проекції.

Роботою з геометричними даними у системі займається клас `MeshObject`, відповідальний за представлення об'єктів, що малюються, в сцені. Він зберігає положення вершин, нормалі, координати текстур та інші атрибути, а також перетворення для кожного об'єкта, такі як переміщення, обертання та масштабування. Для оптимізації зв'язку з OpenGL, `MeshObject` делегує створення та керування буфером до класа `RenderObjectManager`. Цей допоміжний клас підтримує структури VAO, VBO та EBO, необхідні для ефективного рендерингу. Централізуючи взаємодію буферів, система гарантує, що ресурси графічного процесора організовані послідовно та правильно оновлюються щоразу, коли змінюється базова геометрія.

Сама геометрія зберігається в `GeometryClass`, який узагальнює представлення даних вершин для різних типів геометрії. Він організовує всі атрибути вершин в єдиному векторі та включає метадані, що пояснюють, як

слід інтерпретувати атрибути, включаючи розмір та порядок компонентів. Ця абстракція спрощує взаємодію з модулем візуалізації та дозволяє кільком типам геометрії використовувати єдину структуру даних. `GeometryClass` також зберігає буфери індексів вершин для оптимізованих операцій малювання за допомогою індексованого рендерингу.

На цьому загальному представленні базується спеціалізований клас `TerrainGeometry`. Цей підклас розроблений спеціально для процедурної генерації рельєфу та інтегрує алгоритми модифікації та перетворення безпосередньо в геометричне представлення. Він може конвертувати карти висот у масиви вершин, оновлювати геометрію у відповідь на події ерозії або осадження, експортувати рельєф як зображення або як 3D-об’єкт. Успадковуючи від `GeometryClass`, він гарантує, що геометрія рельєфу залишається сумісною з `MeshObject` та робочим процесом візуалізації, водночас забезпечуючи спеціалізовану поведінку, необхідну для генерації ландшафту. У типовому режимі роботи `MeshObject` отримує екземпляр `TerrainGeometry`, що дозволяє в режимі реального часу візуалізувати модифікації, створені алгоритмами шуму або під час симуляції ерозії.

Разом ці компоненти утворюють інтегровану структуру візуалізації, яка доповнює та підтримує підсистеми генерації та ерозії рельєфу. Архітектура забезпечує плавне оновлення візуального виводу по мірі використання користувачем застосунку, що робить модуль візуалізації не лише механізмом відображення, але й важливим аналітичним інструментом. Його модульна конструкція дозволяє майбутні розширення, такі як розширені моделі освітлення, додаткові шари візуалізації або більш складні методи оптимізації геометрії, тим самим забезпечуючи міцну основу для подальшої розробки генерації рельєфу в режимі реального часу.

3.6 Функціонал системи

Функціональність програмної системи пов'язана з якістю та зрозумілістю її графічного інтерфейсу користувача. У застосунках, що

передбачають візуалізацію в реальному часі та інтерактивне налаштування параметрів, таких як генератор ландшафту, інтерфейс відіграє головну роль у тому, наскільки ефективно користувачі можуть отримати доступ до можливостей системи. Добре розроблений графічний інтерфейс не лише організовує доступні інструменти, але й визначає робочий процес, за допомогою якого користувач генерує рельєф, застосовує ерозію, налаштовує параметри візуалізації та експортує результати. Основний інтерфейс (див. Рисунок 11), розробленого застосунку, служить центральною точкою, через яку вся функціональність системи стає доступною для користувача.

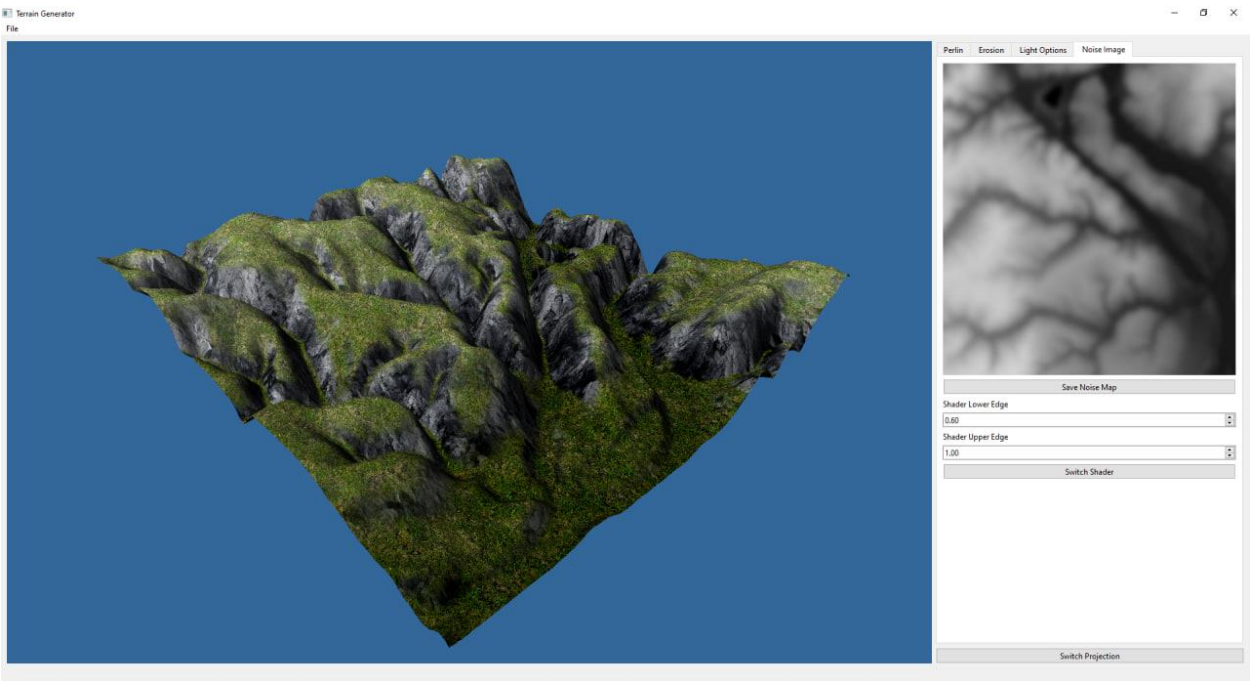


Рисунок 11 — *Графічний інтерфейс розробляємого генератора ландшафту*

Перш ніж детально описувати внутрішню функціональність генератора рельєфу, також важливо встановити основні взаємодії між користувачем та системою. Ці взаємодії можна узагальнити у вигляді діаграми (див. Рисунок 12) варіантів використання, яка окреслює основні дії, які може виконувати користувач, та як ці дії пов'язані з операційними компонентами системи.

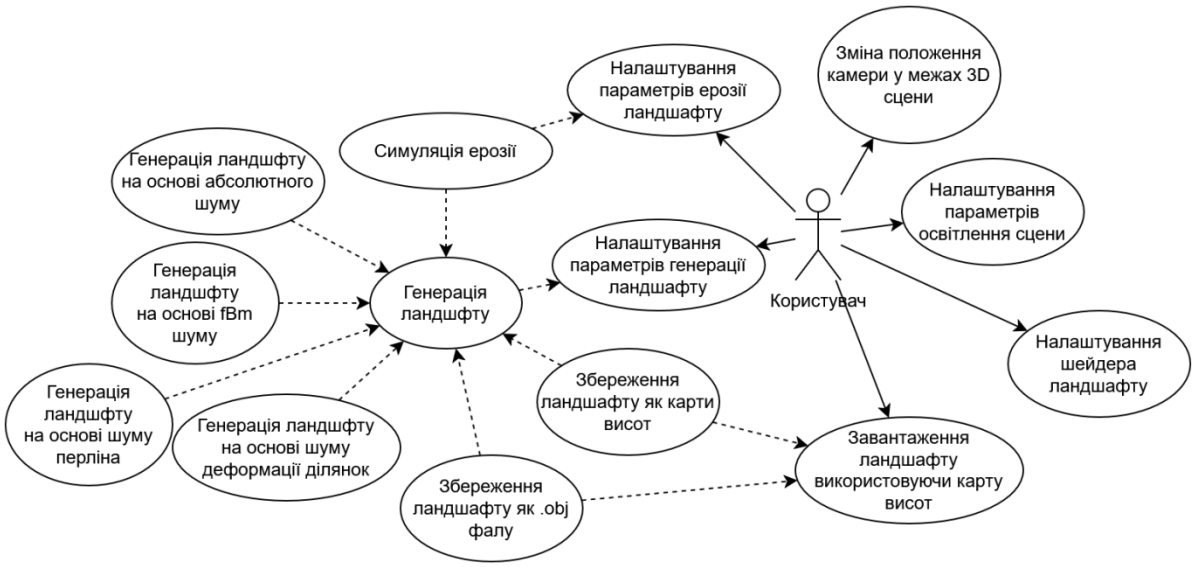


Рисунок 12 — *Use Case діаграма генератора ландшафту*

Генерація ландшафту

Додаток надає гнучкий інструмент генерації ландшафту, який дозволяє користувачеві створювати різноманітні ландшафти, налаштовуючи набір параметрів генерації. Ці параметри впливають як на структуру основних шумових функцій, так і на кінцеву форму карти висот. Для підтримки користувачів, які можуть не мати попереднього досвіду роботи з процедурним шумом, система представляє ці елементи керування в інтуїтивно зрозумілому вигляді, тоді як внутрішня обробка виконується автоматично.

Перша група параметрів стосується структури шумової сітки. Значення «Grid Size» визначає, скільки точок оцінки існує всередині одного квадрата сітки, що використовується під час генерації шуму. Більше значення збільшує щільність вибірки в кожній ділянці, що призводить до плавніших переходів на згенерованій карті висот. Параметр «Grid Count» визначає, скільки таких квадратів сітки створюється вздовж кожної осі. У класичній генерації шуму Перліна створюється двовимірна сітка градієнтних векторів, і значення шуму інтерполюються між ними; таким чином кількість квадратів сітки впливає на загальну частоту та розподіл особливостей ландшафту. Налаштовуючи ці два

значення, користувач ефективно контролює базову просторову роздільну здатність шумового поля.

Фізичний розмір згенерованого рельєфу залежить від параметра «Patch Size», який визначає масштаб кінцевої сітки в одиницях світового виміру. Це дозволяє візуалізувати ту саму конфігурацію шуму як невелику ділянку або як широкий ландшафт, залежно від потреб користувача. Параметр «Amplitude» функціонує як множник висоти, масштабуючи всі значення висоти, що генеруються шумом, і визначаючи, наскільки високим або плоским виглядає результуючий рельєф.

Для генерації фрактального броунівського руху (fBm) система вводить параметри, які визначають, як поєднуються додаткові шари шуму. Параметр «Layer Amount» визначає, скільки послідовних шарів шуму буде накопичено для формування кінцевої карти висот. Кожен додатковий шар використовує модифіковані значення частоти та амплітуди, і ці модифікації контролюються за допомогою параметрів «Next Layer Frequency» та «Next Layer Amplitude», які визначають, наскільки збільшується частота кожного нового шару та наскільки сильно його внесок у висоту впливає на результат. Разом ці елементи керування дозволяють створювати рельєфи від гладких пагорбів до високодеталізованих, нерівних поверхонь.

Коли вибрано шум, деформований доменом, система застосовує додаткове перетворення, яке зміщує координати вибірки на основі іншої функції шуму. Параметр «Domain Warp Weight» визначає силу цього спотворення. Невелика вага створює незначні зміни, які додають ледь помітні нерівності до рельєфу, тоді як більші значення створюють значно більш хаотичні варіації.

Усі ці параметри можна налаштовувати інтерактивно, і кожна зміна запускає генерацію рельєфу, яка негайно відображається у вікні перегляду OpenGL. Діаграму послідовності, що ілюструє типовий потік взаємодії під час генерації рельєфу, зображено на Рисунок 13.

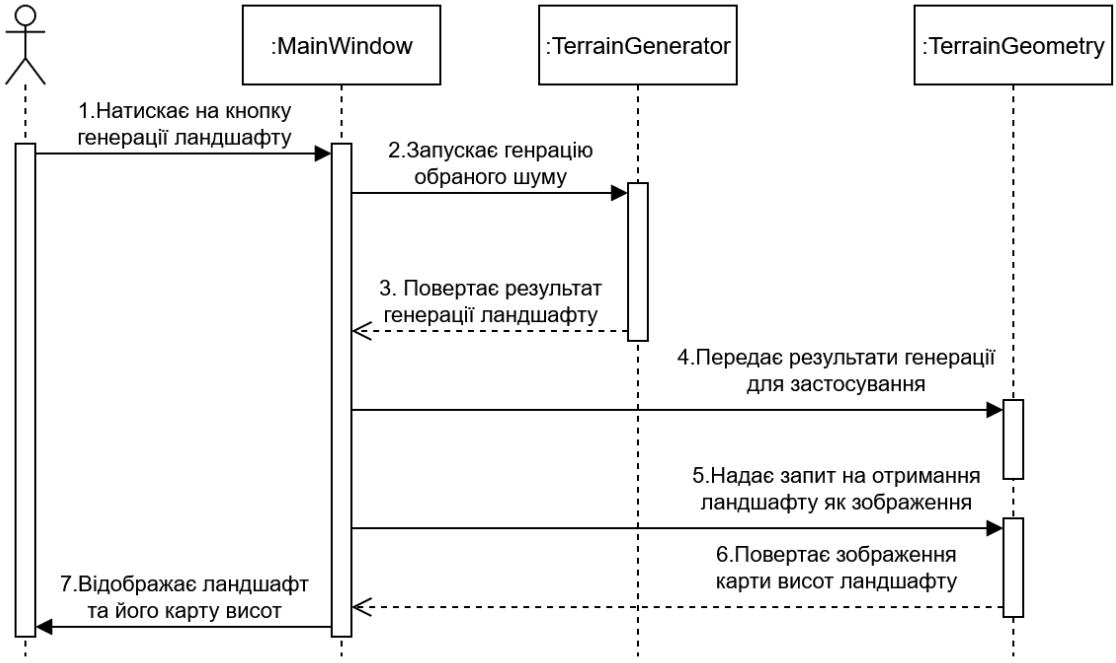


Рисунок 13 — *Sequence діаграма "Генерація ландшафту"*

Симуляція ерозії

Підсистема ерозії працює на існуючій карті висот, яка може бути створена генератором рельєфу на основі шуму програми або імпортована із зовнішнього зображення карти висот. Перед початком моделювання дані рельєфу повинні бути повністю завантажені в пам'ять, щоб значення висоти можна було вибирати, змінювати та візуалізувати в режимі реального часу. Після підготовки рельєфу користувач може налаштувати кілька параметрів, які визначають поведінку алгоритму гідравлічної ерозії та визначають, як краплі взаємодіють з поверхнею.

Параметр «Number Of Iterations» визначає загальну кількість крапель, які будуть моделюватися. Кожна крапля являє собою невеликий об'єм води, який рухається по ландшафту, розмиваючи та відкладаючи осад під час руху. Більша кількість ітерацій підвищує точність і реалістичність кінцевого рельєфу, а також подовжує час обчислення. Значення «Droplet Lifetime» визначає, скільки кроків моделювання може виконати крапля, перш ніж вона припинить своє існування. Протягом свого життя крапля постійно оцінює

нахил рельєфу під нею, щоб визначити напрямок свого руху, поступово формуючи ландшафт.

Під час руху крапель вони накопичують осад відповідно до коефіцієнта ємності, який вказує, скільки ґрунту може переносити одна крапля. Цей фактор безпосередньо впливає на баланс між ерозією та відкладенням на місцевості. Коли крапля має залишкову ємність, вона видаляє ґрунт з ділянок з високим схилом; як тільки вона насичується, вона починає відкладати матеріал на більш пологих ділянках. Швидкість розмивання рельєфу визначається коефіцієнтом ерозії, який визначає величину зменшення висоти на кожному кроці. Разом ці параметри контролюють агресивність моделювання та плавність або нерівність результуючих форм ландшафту.

На рух і поведінку кожної краплі додатково впливає сила тяжіння, яка контролює інтенсивність, з якою краплі спускаються вздовж схилів. Вищі значення сили тяжіння створюють більш виражений низхідний потік, що призводить до глибших каналів і більш чітких річкоподібних структур. Параметр «Brush Size» визначає область впливу навколо положення краплі. Замість того, щоб змінювати одну точку, алгоритм ерозії впливає на невелику околицю точок місцевості, плавно розподіляючи ерозію та відкладення по локалізованій області. Це запобігає різким змінам висоти та дозволяє створювати природні утворення.

Після встановлення параметрів система виконує симуляцію ерозії, у паралельному до застосунка потоці [20], та оновлює рельєф через визначені проміжки часу, залежно від обмежень продуктивності. Вікно перегляду OpenGL негайно відображає ці зміни, дозволяючи користувачеві спостерігати за процесом ерозії у реальному часі. Діаграму послідовності, що ілюструє типовий робочий процес моделювання ерозії, зображено на Рисунок 14.

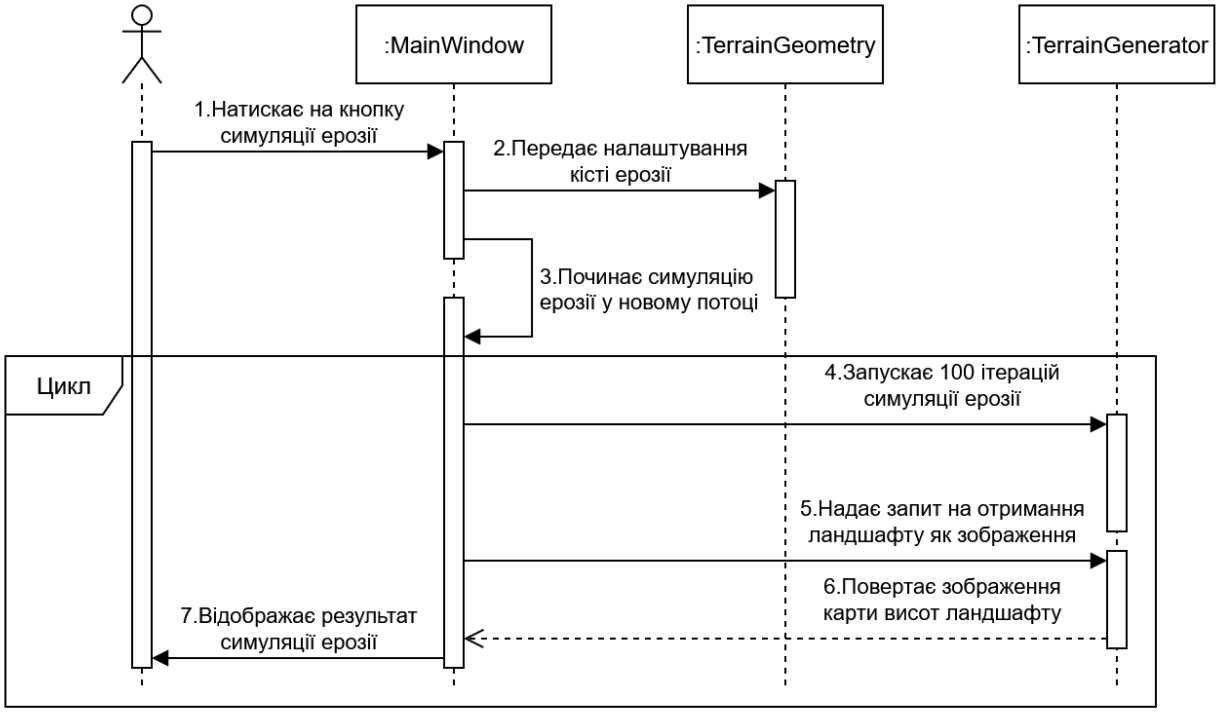


Рисунок 14 — *Sequence діаграма "Симуляція ерозії"*

Збереження ландшафту

Додаток пропонує два способи збереження згенерованого або зміненого ландшафту, що дозволяє користувачам експортувати свої результати у формати, придатні для візуалізації, подальшого редагування або інтеграції у зовнішні робочі процеси. Збереження ландшафту як 3D-об'єкту здійснюється через панель інструментів, де користувач може експортувати поточний ландшафт у широко підтримуваному форматі файлу *.obj*. Ця функція експорту перетворює внутрішньо збережену карту висот у структурований набір вершин, індексів та нормалей, створюючи файл, який можна відкрити в програмному забезпеченні для моделювання або використовувати в інших середовищах візуалізації. Система гарантує, що експортована геометрія точно відображає всі застосовані перетворення шуму, ефекти ерозії та налаштування масштабування.

Окрім експорту геометричного представлення, додаток дозволяє зберігати саме зображення карти висот. За допомогою кнопки «Save Noise Map», розташованої на вкладці «Noise Image», користувач може створити файл

.png, що містить зображення значень висоти ландшафту у градаціях сірого. Ця опція корисна, коли користувач бажає повторно використовувати карту висот в іншому генераторі ландшафту, виконати обробку на основі текстур, зберегти процедурну конфігурацію шуму без експорту геометрії або у цілях подальшого дослідження.

Обидва методи збереження мають схожу операційну структуру: користувач ініціює експорт, система збирає необхідні дані з активного представлення місцевості, форматує їх відповідно до вибраного типу виводу та записує файл на диск. Діаграму послідовності, що ілюструє експорт місцевості у форматі *.obj*, зображено на Рисунок 15.

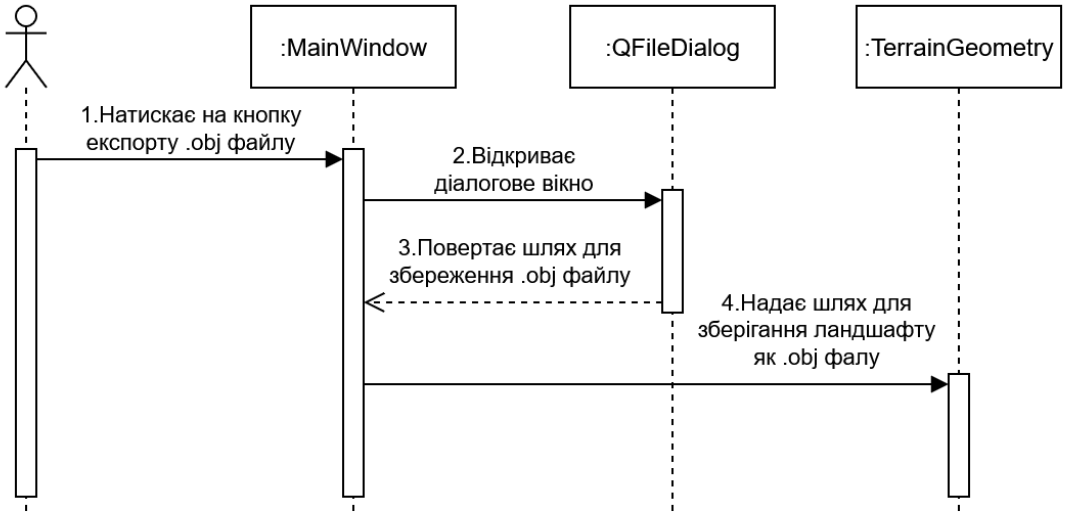


Рисунок 15 — *Sequence діаграма "Експорт ландшафту як .obj файлу"*

Завантаження ландшафту

Система підтримує завантаження ландшафту із зовнішнього зображення карти висот, що дозволяє користувачам працювати не лише з процедурно згенерованими ландшафтами, але й з користувацькими або вже існуючими даними. Завантаження ландшафту ініціюється через панель інструментів, де користувач вибирає зображення *.png*, що представляє інформацію про висоту у відтінках сірого. Після вибору файлу програма пропонує користувачеві

вказати значення масштабу, в якому одна точка геометрії ландшафту може відповідати одному, двом, трьом або чотирьом пікселям у вхідному зображенні. Цей крок масштабування забезпечує контроль над кінцевою роздільною здатністю завантаженого ландшафту, що дозволяє отримати або дуже детальну реконструкцію, або більш гладку, узагальнену поверхню, отриману з того ж зображення.

Після підтвердження коефіцієнта масштабування система зчитує карту висот, перетворює інтенсивність кожного пікселя на значення висоти та створює внутрішній масив висот відповідно до вибраного відображення. Потім сітка ландшафту генерується з використанням цих значень висоти, гарантуючи правильне перетворення імпортованих даних у геометричну форму. Будь-який раніше згенерований або змінений ландшафт замінюється під час цього процесу. Вікно перегляду OpenGL оновлюється негайно, щоб відобразити щойно завантажений ландшафт, що дозволяє користувачеві продовжувати роботу з моделюванням ерозії або операціями експорту за потреби.

Діаграму послідовності, що ілюструє робочий процес завантаження ландшафту із зображення, починаючи з дії користувача на панелі інструментів, а потім діалогове вікно вибору масштабу, розбір зображення, перетворення висоти та остаточну візуалізацію реконструйованого ландшафту, зображено на Рисунок 16.

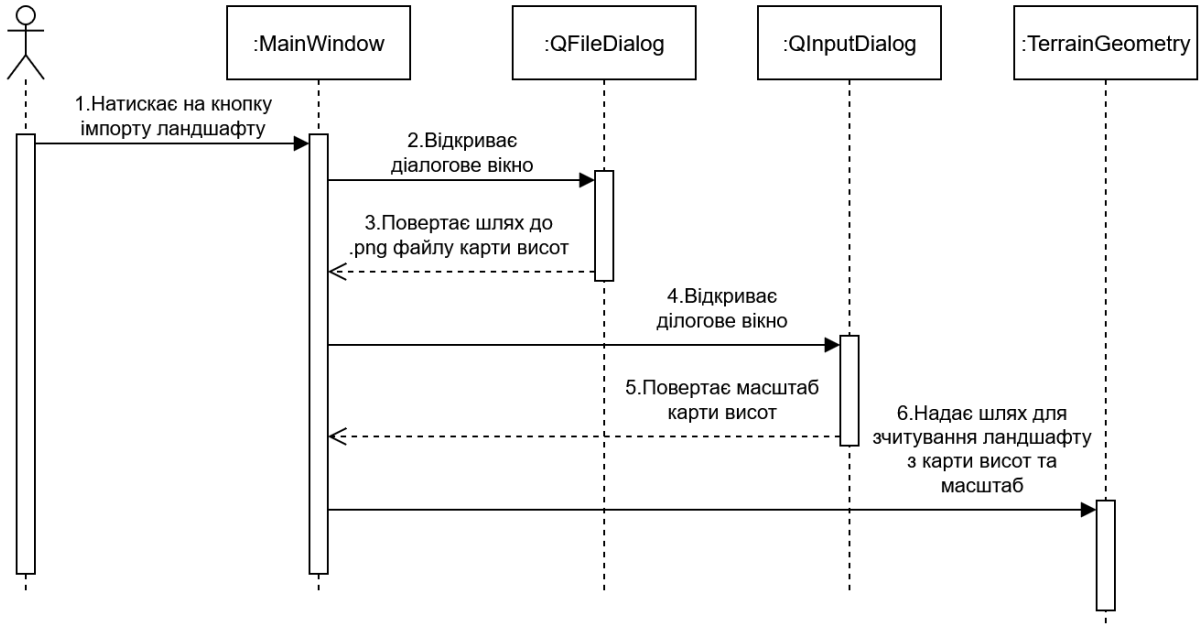


Рисунок 16 — *Sequence діаграма "Імпорт ландшафту"*

Візуалізація результатів

Візуалізація рельєфу здійснюється безперервним циклом рендерингу, який працює з фіксованою частотою 60 кадрів на секунду. Підтримка цієї частоти кадрів гарантує, що кожна зміна, внесена користувачем — чи то налаштування параметрів шейдерів, зміна умов освітлення, навігація сценою за допомогою камери чи застосування ерозії в режимі реального часу — негайно відображається на екрані. Цикл рендерингу оновлює область перегляду OpenGL через певні проміжки часу, забезпечуючи плавний зворотний зв'язок і дозволяючи користувачам інтуїтивно спостерігати, як кожна дія впливає на зовнішній вигляд рельєфу [25].

У кожній ітерації циклу система отримує актуальний стан геометрії рельєфу, конфігурації шейдерів, параметрів освітлення та трансформації камери. Якщо користувач запускає симуляцію ерозії, цикл відображає кожне проміжне оновлення карти висот, коли краплі рухаються та змінюють форму поверхні. Коли налаштування шейдерів змінюються, цикл запускає реконфігурацію активної програми шейдерів, щоб колір, освітлення або режими візуалізації змінювалися миттєво. Безперервно перемальовуючи

сцену, цикл рендерингу діє як об'єднуючий механізм, що поєднує взаємодію користувача з візуальним виходом.

Діаграму послідовності, яка ілюструє процес циклу рендерингу, зображено на Рисунок 17. Починаючи з запланованого оновлення кадру, далі йде отримання даних сцени, виконання програм шейдерів, рендеринг ландшафту та відображення кадру на екрані.

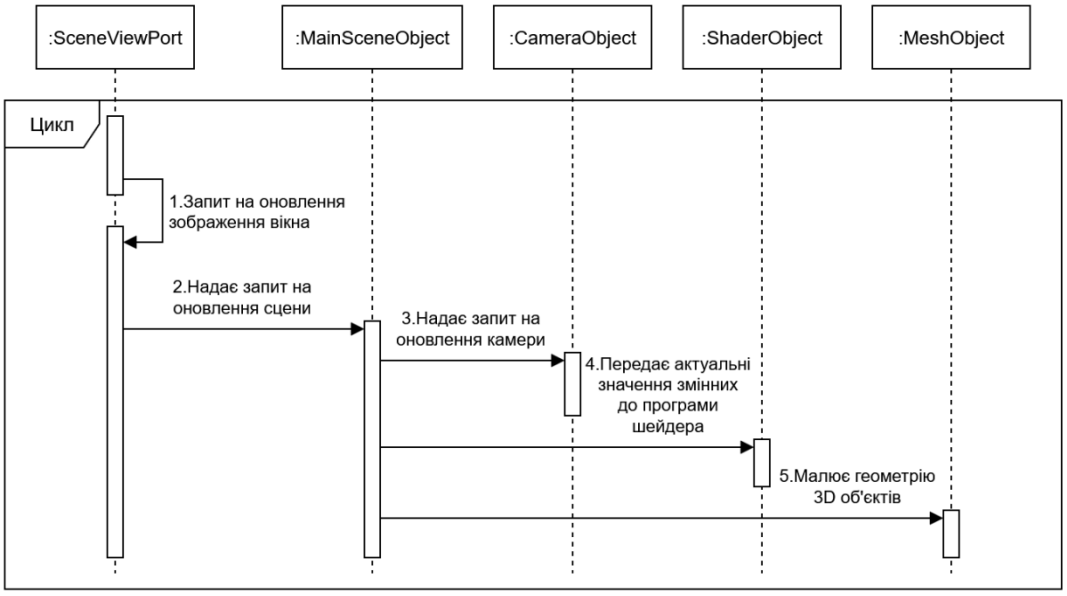


Рисунок 17 — *Sequence діаграма "Цикл візуалізації ландшафту"*

3.7 Вимоги до інтерфейсу

Інтерфейс програми для генерації ландшафтів у реальному часі повинен надавати користувачам інтуїтивно зрозумілі та гнучкі інструменти для побудови, модифікації та візуалізації процедурно згенерованих рельєфів. Оскільки програма підтримує кілька методів генерації на основі шуму, моделювання гідравлічної ерозії в реальному часі та інтерактивну 3D-візуалізацію, інтерфейс повинен забезпечувати легкий доступ та логічну організацію всіх функцій на основі наступних вимог до інтерфейсу програми:

1. Елементи керування генерацією рельєфу

   a. Варіанти генерації рельєфу за допомогою шуму Перліна, фрактального шуму броунівського руху, шуму деформації ділянок, та абсолютного шуму.
   b. Налаштовувані параметри для кожного типу шуму, включаючи частоту, амплітуду, кількість октав, інтенсивність деформації та інші налаштування, що стосуються методу.
   c. Можливість завантаження зовнішнього зображення карти висот для ініціалізації рельєфу замість його процедурної генерації.
2. Елементи керування моделюванням ерозії
   a. Елементи інтерфейсу для симуляції гідравлічної ерозії.
   b. Налаштовувати параметри, такі як кількість крапель, радіус ерозії, швидкість осадження, швидкість моделювання та інших параметрів моделі ерозії.
   c. Відображення часу симуляції ерозії за заданими параметрами.
3. Керування освітленням
   a. Можливість налаштування положення спрямованого світла у сцені.
   b. Керування кольором спрямованого світла для кращої візуалізації особливостей рельєфу.
4. Параметри рендерингу та шейдерів
   a. Можливість перемикання між реалістичним шейдером рельєфу та абстрактним шейдером візуалізації.
   b. Застосування змін шейдерів у реальному часі в області перегляду OpenGL.
5. Керування камерою
   a. Режим орбітальної камери, що дозволяє обертання навколо рельєфу з регульованою відстанню та кутом.
   b. Режим вільного польоту камери для більш гнучкого дослідження згенерованого ландшафту.

   c. Перемикання між режимами ортографічної та ізометричної проекції камери.

6. Вимоги до області перегляду
   a. Інтегрована область перегляду OpenGL, призначена для попереднього перегляду рельєфу в реальному часі.
   b. Плавна продуктивність рендерингу під час перемикання між шейдерами, налаштування освітлення або запуску симуляції ерозії.
7. Експорт та збереження
   a. Можливість збереження згенерованого рельєфу як зображення карти висот.
   b. Можливість експорту рельєфу у форматі .obj для використання в зовнішніх інструментах.

Ці вимоги до інтерфейсу гарантують, що програма залишається зручною для користувача, забезпечує детальний контроль над створенням рельєфу та моделюванням ерозії, а також підтримує ефективні робочі процеси візуалізації та експорту, що відповідають цілям цієї дисертації.

3.8 Аналіз вимог до користувачів генератора ландшафту

Розуміння потреб та очікувань користувачів є важливою частиною розробки практичного та ефективного інструменту для генерації ландшафту. Оскільки програма призначена для візуалізації рельєфу в реальному часі, генерації на основі шуму та інтерактивного моделювання ерозії, визначення основних вимог користувачів гарантує, що розроблений інтерфейс та робочий процес повністю підтримуватимуть типові випадки використання, такі як:

1. Користувачі повинні мати можливість генерувати рельєф за допомогою кількох попередньо визначених методів на основі шуму та змінювати параметри генерації простим та інтуїтивно зрозумілим способом.

2. Програма повинна дозволяти користувачам швидко переглядати зміни рельєфу у вікні перегляду OpenGL без складних кроків налаштування.
3. Користувачі повинні мати доступ до налаштованих параметрів моделювання ерозії та можливість спостерігати за прогресом ерозії в реальному часі.
4. Інструмент повинен забезпечувати прості елементи керування освітленням та вибором шейдерів для покращення візуалізації рельєфу.
5. Користувачі повинні мати можливість завантажувати власні зображення карти висот та експортувати кінцеві результати у поширені формати, такі як зображення карти висот або файлу .obj.
6. Елементи керування камерою повинні бути адаптивними та простими у використанні, підтримуючи як орбітальний, так і режим вільної навігації.

Окрім розуміння очікувань користувачів, також важливо визначити рівень навичок, необхідних для ефективної роботи з програмою. Це гарантує, що складність інтерфейсу відповідає цільовій аудиторії та не створює зайвих бар'єрів. Користувачі повинні мати:

1. Базове знайомство з 3D-програмами, включаючи такі концепції, як рух камери та навігація по сцені.
2. Розуміння карт висот та представлення рельєфу для правильного використання функцій імпорту та експорту.
3. Загальні знання процедурних функцій шуму та основних параметрів, таких як частота та амплітуда, для ефективного налаштування параметрів генерації.
4. Розуміння того, як гідравлічна ерозія впливає на форми рельєфу, для інтерпретації результатів моделювання та налаштування параметрів ерозії.

Ці вимоги та очікувані навички користувача допомагають спрямувати дизайн програми в напрямку зручності використання, зрозумілості та

доступності, гарантуючи, що інструмент придатний для використання за призначенням у генерації ландшафту в режимі реального часу.

3.9 Аналіз вимог до апаратного та програмного забезпечення

Розробка генератора ландшафту потребує глибокого розуміння як апаратних, так і програмних аспектів. Ці параметри не лише визначають можливості та швидкодію програми, а й гарантують її роботу на різноманітних системах.

Вимоги до обладнання:

1. Графічний адаптер із підтримкою OpenGL 3.3 або новішої версії.
2. Багатоядерний процесор із частотою від 2,0 ГГц для стабільного функціонування.
3. Щонайменше 4 ГБ оперативної пам’яті, більший обсяг покращує продуктивність, особливо під час роботи з деталізованими ландшафтами.
4. Жорсткий диск або SSD з мінімум 1 GB вільного місця.

Щодо програмних вимог, застосунок потребує лише операційну систему Windows, macOS або Linux.

4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ ГЕНЕРАЦІЇ ЛАНДШАФТУ У РЕАЛЬНОМУ ЧАСІ

4.1 Аналіз результатів генерації рельєфу на основі шуму

4.1.1 Шум Перліна

Шум Перліна служить базовим генератором для системи та являє собою найпростіший з реалізованих методів. Його вихід характеризується плавними, безперервними градієнтами, що утворюють пологі пагорби, та широкими, низькочастотними перепадами висот. Це робить його придатним як базовий рельєф або як перший етап у складніших алгоритмах. Технічно, шум Перліна є високоефективним, потребуючи відносно мало операцій вибірки, які добре впливають на продуктивність у реальному часі. Його обчислювальна передбачуваність дозволяє системі підтримувати стабільну частоту кадрів навіть при вищій роздільній здатності рельєфу.

Однак, шум Перліна також має кілька обмежень. Створений рельєф, як правило, немає дрібніших деталей і тому є менш візуально привабливим, коли використовується окремо. У деяких випадках стають видимими артефакти, вирівняні по сітці, особливо при зменшенні масштабу або використанні високочастотного масштабування. Ці артефакти виникають через базову градієнтну структуру решітки. Рельєф на основі Перліна (див. Рисунок 18) видається найгладшим і найменш нерівним з усіх оцінених методів. Незважаючи на ці недоліки, його ефективність і стабільність роблять його цінним компонентом системи.

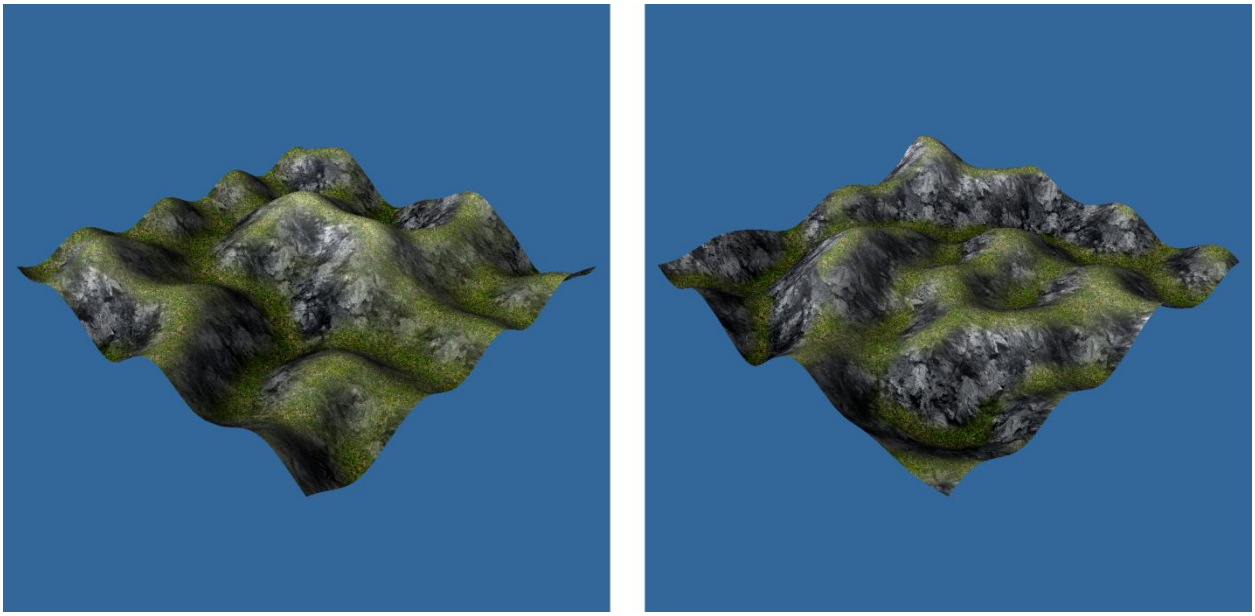


Рисунок 18 — *Згенерований ландшафт на основі шуму Перліна*

4.1.2 Фрактальний броунівський рух (fBm)

Шум fBm базується на шумі Перліна шляхом нашарування кількох октав шуму, кожна з яких має свою частоту та амплітуду. Цей підхід додає поступово тонші деталі до рельєфу, створюючи багату різноманітність форм, включаючи гостріші хребти, більш виражені долини та фрактальну шорсткість, що імітує природні гірські утворення. Оскільки шум накопичується протягом кількох октав, отримані карти висот демонструють збалансоване поєднання глобальної структури та локальних деталей.

З технічної точки зору, fBm є більш обчислювально вимогливим, ніж стандартний шум Перліна, оскільки вимагає багаторазових оцінок функції шуму для кожної точки рельєфу. Тим не менш, підвищена вартість залишається прийнятною для генерації в реальному часі в межах типових розмірів рельєфу, що використовуються в застосунку. Згенерований шум (див. Рисунок 19) демонструє наскільки виразнішим та реалістичнішим виглядає рельєф fBm порівняно з базовим варіантом Перліна. Його складність робить його добре придатним для моделювання природних ландшафтів та для сцен, що вимагають візуально різноманітного гірського рельєфу.

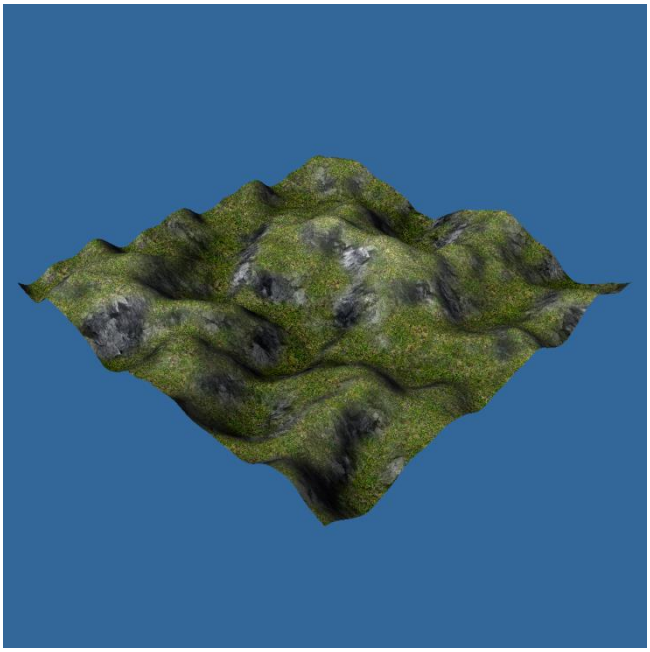


Рисунок 19 — *Згенерований ландшафт на основі шум fBm*

4.1.3 Шум деформованих ділянок

Шум деформованих ділянок є найскладнішим з реалізованих методів. У цій техніці одна шумова функція використовується для спотворення координат вибірки іншої, створюючи нелінійні деформації, які різко змінюють просторову структуру місцевості. Це призводить до дуже нерегулярних форм ландшафту, таких як скручені хребти, спіралеподібні долини, розгалужені канали, подібні до ерозії, та хаотичний розподіл висот, які неможливо отримати за допомогою простіших методів. Ці закономірності можуть нагадувати як природні геологічні утворення, так і стилізовані фантастичні середовища.

Обсяг обчислень деформації ділянок значно вища, ніж у шуму Перліна або fBm. Оскільки ділянку необхідно вибірково використовувати принаймні двічі, а часто й кілька разів, кількість пошуків шуму значно зростає. Незважаючи на це, система підтримує швидкість реагування в режимі реального часу при помірній роздільній здатності місцевості. Шум деформації ділянок (див. Рисунок 20) дає найбільш візуально нетрадиційні результати, що робить його предатним для випадків, де потрібна нелінійна структура місцевості.

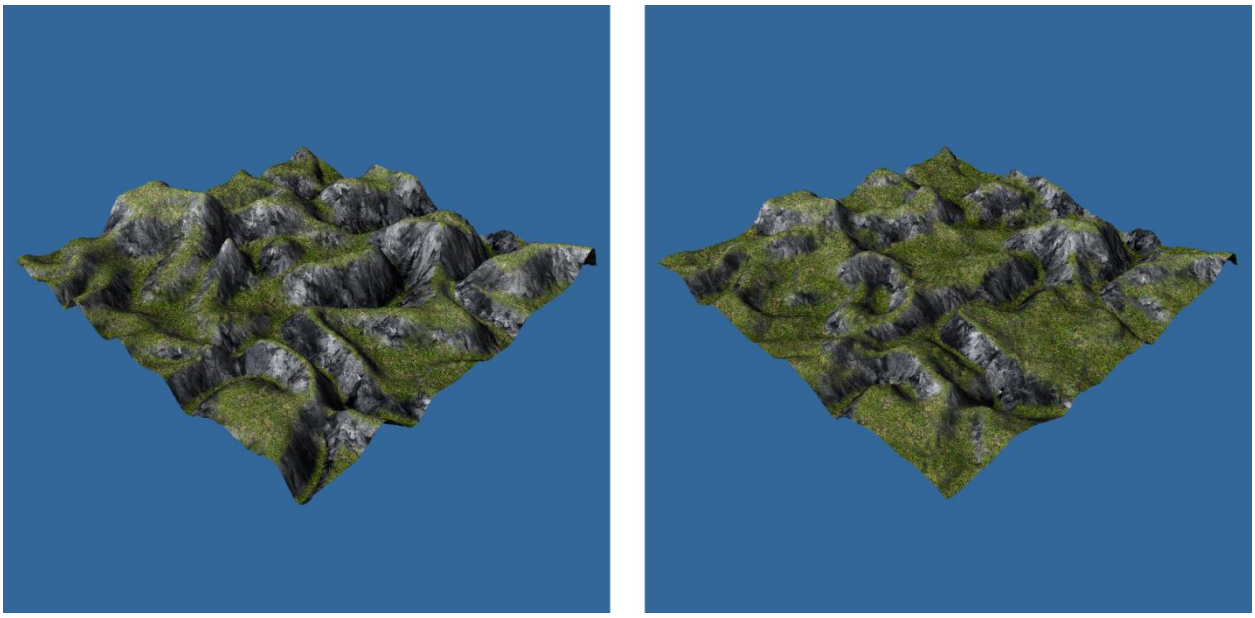


Рисунок 20 — *Згенерований ландшафт на основі шум деформації ділянок*

4.1.4 Абсолютний шум

Абсолютний шум модифікує шум Перліна, застосовуючи абсолютну функцію до його значень перед масштабуванням їх у діапазоні висот. Ця операція призводить до симетричних градієнтних структур з різкими, хребтоподібними формами, які повторюються по всій місцевості. Оскільки абсолютна операція є недорогою, обчислювальні витрати залишаються майже ідентичними стандартним шумом Перліна.

Візуально абсолютний шум створює висококонтрастні переходи висот, круті схили та виражені хребтоподібні структури, які виділяються навіть на низьких частотах. Однак симетрія, введена абсолютною функцією, може призвести до повторюваних або штучно виглядаючих форм місцевості, що може бути небажаним для застосувань, які вимагають природного реалізму. Результати згенрованого абсолютного шуму (див. Рисунок 21), підкреслюють ці характеристики, демонструючи, як абсолютний шум створює скелеподібні структури та дзеркальні утворення, відсутні в інших методах. Хоча він менш підходить для реалістичних ландшафтів, його унікальні особливості роблять його цінним для стилізованих або експериментальних типів місцевості.

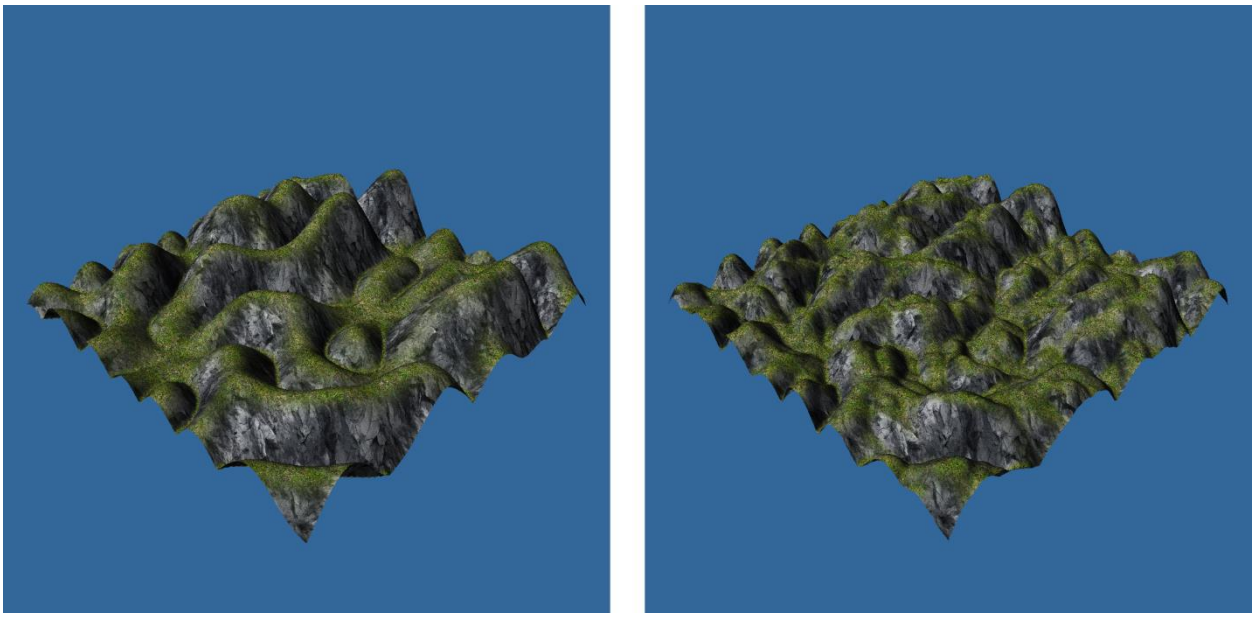


Рисунок 21 — *Згенерований ландшафт на основі абсолютного шум*

4.1.5 Порівняння результатів

Порівняльна оцінка чотирьох шумових функцій виявляє широкий спектр обчислювальних та візуальних характеристик. Шум Перліна є найефективнішим і створює найплавніші ландшафти, але йому бракує деталей. fBm додає природної фрактальної складності за помірних витрат, що робить його потужним методом загального призначення. Шум з деформацією домену досягає найвищого рівня варіації та більшого діапазону візуальних результатів, хоча й зі значно більшими обчислювальними витратами. Абсолютний шум залишається ефективним, пропонуючи при цьому дуже чіткі структури хребтів, але має вплив штучної симетрії.

Ці результати підтверджують, що жодна окрема функція шуму не є універсально оптимальною. Натомість кожен метод підтримує різні цілі генерації рельєфу. Пропонуючи різний набір алгоритмів шуму, програма дозволяє користувачам або орієнтуватися на фотореалістичні ландшафти, або експериментувати з високо стилізованими ландшафтами відповідно до їхніх творчих та технічних вимог.

4.2 Аналіз результатів симуляції гідравлічної ерозії

Алгоритм, що використовується в цій роботі, базується на моделі гідравлічної ерозії на основі крапель, добре зарекомендувавши себе в процедурних дослідженнях рельєфу [21]. У цьому методі тисячі змодельованих крапель води рухаються по поверхні рельєфу, кожна з яких представляє невелику та локалізовану подію ерозії. Коли краплі прискорюються вниз по крутих схилах, вони розмивають ґрунт; коли вони сповільнюються або досягають більш рівних ділянок, вони відкладають осад. Така динамічна поведінка призводить до появи реалістичних структур, таких як канали ерозії, розгалужені русла річок та природно згладжені хребти рельєфу. Сукупний ефект від багатьох крапель, що застосовуються з часом, призводить до створення рельєфу, який візуально нагадує реальні геологічні утворення, сформовані потоком води.

Для ілюстрації візуальних результатів моделі як процедурний, так і реальний рельєф були оброблені за допомогою алгоритму ерозії. Ці приклади демонструють, як система обробляє різні початкові умови рельєфу.

У процедурних прикладах рельєфу (див. Рисунок 22) моделі дренажу виникають природним чином із потоку крапель вздовж локальних схилів, прокладаючи правдоподібні шляхи в поверхні рельєфу. Високочастотні шумові артефакти, часто видимі на картах висот на основі чистого шуму, зменшуються або усуваються завдяки комбінованому впливу ерозії та відкладення осаду. Карти висот на основі шуму, спочатку характеризувалися гострими краями або повторюваними візерунками, набувають значно більш природного вигляду після ерозії.

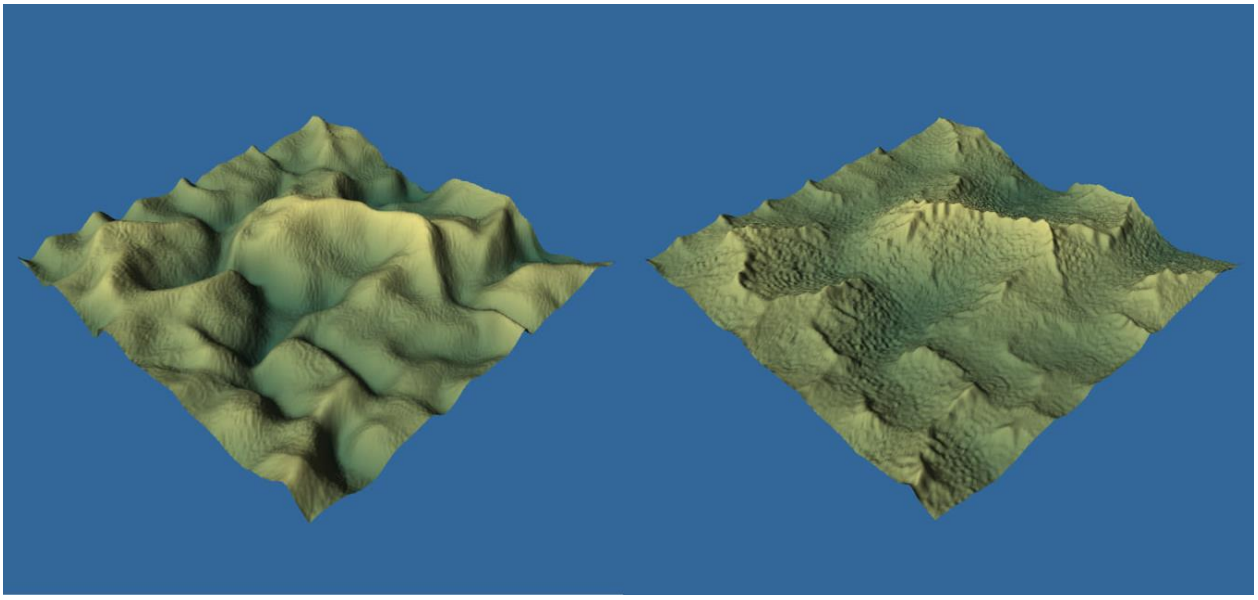


Рисунок 22 — *Симуляція ерозії на основі процедурно згенерованого ландшафту*

Натомість, коли ерозія застосовується до імпортованих даних про рельєф реального світу, система показує, як існуючі форми рельєфу можуть розвиватися під постійним гідрологічним впливом. Порівняння «до» та «після» (див. Рисунок 23) показує потенційне поглиблення долин, розширення площ басейнів та формування зон відкладень. Це робить програму корисною не лише для візуальних ефектів, але й для освітніх та дослідницьких цілей, дозволяючи користувачам аналізувати гіпотетичні геологічні процеси.

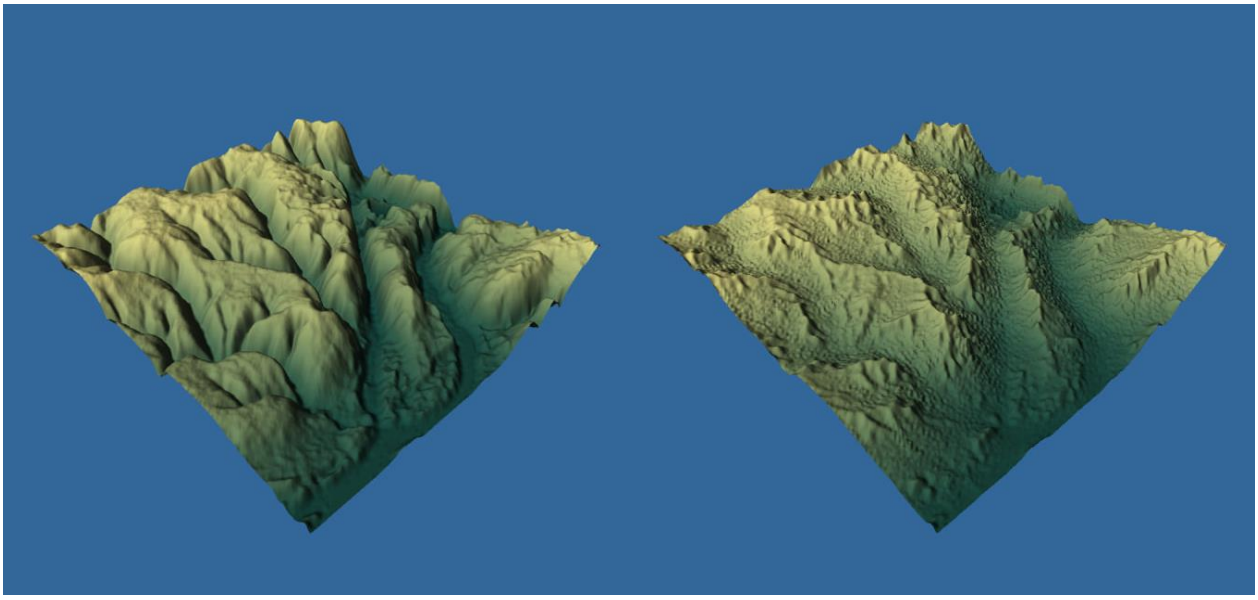


Рисунок 23 — *Симуляція ерозії на основі карти висот реального ландшафту*

Також важливою частиною цього аналізу є ступень реалістичності зображення після ерозії [22]. Спочатку система спиралася на класичний метод, за якого значення градієнтів для всієї сітки рельєфу обчислювалися заздалегідь, перш ніж можна було розпочати будь-які кроки моделювання. Хоча теоретично цей підхід був ефективним, він призвів до значного часу попередньої обробки, особливо для рельєфів з вищою роздільною здатністю. Обчислення градієнтів для кожної комірки, навіть тих, що не торкнулися шляхів крапель, призвело до непотрібних обчислювальних витрат та помітної затримки, перш ніж користувачі змогли спостерігати процес ерозії. Це обмеження значно знизило залученість користувача та зробило ітеративне експериментування менш ефективним, особливо під час роботи з великими ландшафтами.

Щоб подолати ці проблеми, алгоритм було вдосконалено стратегією обчислення градієнта лише за потреби. В модифікованій версії градієнт у певній комірці місцевості обчислюється у разі необхідності, а саме, коли крапля вперше потрапляє в це місце. Після обчислення градієнт зберігається та повторно використовується для всіх наступних крапель, запобігаючи надлишковим перерахункам. Це перетворення переносить обчислювальні

витрати з етапу попередньої обробки на етап моделювання, але дуже ефективним способом: обчислюються лише градієнти, які фактично впливають на рух крапель. В результаті час попередньої обробки майже виключається, а загальний час моделювання значно скорочується в сценаріях, коли великі ділянки місцевості залишаються недоторканими краплями.

Дані про продуктивність, що порівнюють класичний та модифікований методи, наведені в Таблиця 1, яка містить детальні вимірювання для різних роздільних здатностей місцевості. Тести з вищою роздільною здатністю показують найбільший приріст продуктивності, оскільки класичний метод погано масштабується, коли кожна комірка сітки вимагає попереднього обчислення. Модифікований метод на вимогу, навпаки, масштабується більш ефективно, уникаючи роботи, яка безпосередньо не сприяє фізичному моделюванню. Дані показують, що ця модифікація може суттєво скоротити час виконання ерозії без шкоди для точності результатів.

Таблиця 1 — *Заміри роботи класичного алгоритму та модифікованого варіанту, у секундах*

Розмір ділянки ландшафту n×n	**Класичний алгоритм**	**Модифікований алгоритм**
128	6.31	5.53
256	28.72	19.88
512	93.04	83.90

Важливо, що модифікація не змінює логіку ерозії, а це означає, що візуальні результати залишаються повністю однаковими з результатами класичного підходу. Моделі ерозії, поведінка перенесення осаду та гідрогеологічні перетворення, отримані обома методами, ідентичні. Це зберігає наукову цілісність моделювання, що робить модифікований алгоритм таким же придатним для аналітичних цілей, як і немодифіковану версію.

Загалом, модифікована реалізація гідравлічної ерозії досягає балансу між обчислювальною ефективністю, продуктивністю в реальному часі та

візуальною точністю. Покращення, запроваджені завдяки обчисленню градієнтів у разі потреби, дозволяють швидке моделювання без шкоди для реалізму чи узгодженості результатів ерозії. Поєднання даних про продуктивність та візуальних демонстрацій підтверджує, що система добре підходить як для креативного проектування місцевості, так і для наукових досліджень. Зберігаючи точну геологічну поведінку та значно знижуючи обчислювальні витрати, розроблений модуль ерозії посилює загальну функціональність та застосовність інструментарію для генерації ландшафту в реальному часі.

ВИСНОВКИ

1. Проведене дослідження існуючих аналогів у галузі процедурної генерації рельєфу та моделювання ерозії. Детальний огляд наукової літератури та сучасних програмних рішень дав уявлення про сильні сторони, обмеження та можливості різних підходів до генерації рельєфу. Розуміння цих існуючих рішень стало міцною основою для проектування та розробки запропонованого застосування.
2. У ході цього дослідження були впроваджені, вивчені та порівнянні різні методи процедурної генерації рельєфу, щоб визначити їх придатність для створення реалістичних ландшафтів та для інтеграції з моделюванням ерозії. Порівняльний аналіз цих методів дозволяє глибше зрозуміти, як різні шумові функції впливають на структуру рельєфу, ступень реалістичності зображення та обчислювальну продуктивність.
3. Значним результатом дослідження є модифікаія алгоритму гідравлічної ерозії, що використовується в додатку. Завдяки впровадженню модифікації, обчислювальна продуктивність алгоритму була покращена, зберігаючи при цьому візуальний реалізм ефектів ерозії.
4. Розроблен додаток процедурної генерації рельєфу з інтегрованим моделюванням ерозії, який дозволяє генерувати ландшафти, використовуючи кілька методів шуму, застосовувати ерозію в реальному часі та налаштовувати параметри для дослідження різних результатів.
5. Додаток демонструє практичну можливість поєднання моделей рельєфу на основі шуму з модифікованими алгоритмами ерозії.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Perlin K. An image synthesizer. ACM SIGGRAPH Computer Graphics. 1985. Т. 19, № 3. С. 287–296. URL: https://doi.org/10.1145/325165.325247 (дата звернення: 04.06.2025).

2. Mandelbrot B. B. Fractal Geometry of Nature. Echo Point Books & Media, LLC., 2021. 500 с.

3. Musgrave F. K., Kolb C. E., Mace R. S. The synthesis and rendering of eroded fractal terrains. ACM SIGGRAPH Computer Graphics. 1989. Т. 23, № 3. С. 41–50. URL: https://doi.org/10.1145/74334.74337 (дата звернення: 18.09.2025).

4. Large Scale Terrain Generation from Tectonic Uplift and Fluvial Erosion / G. Cordonnier та ін. Computer Graphics Forum. 2016. Т. 35, № 2. С. 165–175. URL: https://doi.org/10.1111/cgf.12820 (дата звернення: 14.04.2025).

5. A Survey of Procedural Noise Functions / A. Lagae та ін. Computer Graphics Forum. 2010. Т. 29, № 8. С. 2579–2600. URL: https://doi.org/10.1111/j.1467-8659.2010.01827.x (дата звернення: 04.06.2025).

6. A Survey on Procedural Modelling for Virtual Worlds / R. M. Smelik та ін. Computer Graphics Forum. 2014. Т. 33, № 6. С. 31–50. URL: https://doi.org/10.1111/cgf.12276 (дата звернення: 10.06.2025).

7. Кузенний Н. С. СУЧАСНІ МЕТОДИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ЛАНДШАФТУ. ІНЖЕНЕРНІ ІННОВАЦІЇ ТА РОЗБУДОВА НАЦІОНАЛЬНОЇ ЕКОНОМІКИ : Матеріали II Міжнар. науково-практ. конф., м. Запоріжжя, 15–16 трав. 2025 р. Запоріжжя, 2025. С. 479–481.

8. Texturing and Modeling: A Procedural Approach / D. S. Ebert та ін. Morgan Kaufmann Publishers, 1998. 415 с.

9. ISO C++ Documentation. URL: https://isocpp.org (дата звернення: 18.11.2025).

10. OpenGL Architecture Review Board. Official OpenGL Specifications. The Khronos Group. URL: https://www.khronos.org/opengl/ (дата звернення: 02.06.2025).

11. Core Language (GLSL) - OpenGL Wiki. The Khronos Group. URL: https://www.khronos.org/opengl/wiki/Core_Language_(GLSL) (дата звернення: 02.06.2025).

12. Qt Documentation. URL: https://doc.qt.io (дата звернення: 02.06.2025).

13. OpenGL Mathematics. GLM. URL: https://glm.g-truc.net/0.9.9/ (дата звернення: 02.06.2025).

14. Shalloway A., Trott J. Design Patterns Explained: A New Perspective on Object-Oriented Design (2nd Edition) (Software Patterns Series). 2-ге вид. Addison-Wesley Professional, 2004. 480 с.

15. Domain Warping - 2002. Inigo Quilez. URL: https://iquilezles.org/articles/warp/ (дата звернення: 21.09.2025).

16. Benes B., Forsbach R. Layered data representation for visual simulation of terrain erosion. Spring Conference on Computer Graphics, м. Budmerice, Slovakia. URL: https://doi.org/10.1109/sccg.2001.945341 (дата звернення: 22.09.2025).

17. Interactive Terrain Modeling Using Hydraulic Erosion / O. Stava та ін. Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation, м. Goslar, 2008 p. Switzerland, 2008. С. 201–210. URL: https://doi.org/10.2312/SCA/SCA08/201-210 (дата звернення: 25.09.2025).

18. Vries J. D. Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion. Kendall & Welling, 2020. 522 с.

19. Kessenich J., Baldwin D., Rost R. The OpenGL Shading Language, Version 4.60.8 / ред. G. Leese AddisonWesley Professional, 2023. 244 с.

20. Gain J., Merry B., Marais P. Parallel, Realistic and Controllable Terrain Synthesis. Computer Graphics Forum. 2015. Т. 34, № 2. С. 105–116. URL: https://doi.org/10.1111/cgf.12545 (дата звернення: 22.10.2025).

21. Mei X., Decaudin P., Hu B.-G. Fast Hydraulic Erosion Simulation and Visualization on GPU. 15th Pacific Conference on Computer Graphics and Applications (PG'07), м. Maui, HI, USA, 29 жовт. – 2 листоп. 2007 р. 2007. URL: https://doi.org/10.1109/pg.2007.15 (дата звернення: 28.10.2025).

22. PTRM: Perceived Terrain Realism Metric / S. D. Rajasekaran та ін. ACM Transactions on Applied Perception. 2022. URL: https://doi.org/10.1145/3514244 (дата звернення: 26.11.2025).

23. Angel E. Interactive Computer Graphics: A Top-down Approach Using OpenGL. Pearson Education, Limited, 2005. 816 с.

24. Mizutani W. K., K. Daros V., Kon F. Software architecture for digital game mechanics: A systematic literature review. Entertainment Computing. 2021. Т. 38. С. 100421. URL: https://doi.org/10.1016/j.entcom.2021.100421 (дата звернення: 26.11.2025).

25. Computer graphics: Principles and practice / J. F. Hughes та ін. 2-ге вид. Reading, Mass : Addison-Wesley, 1995. 1175 с.

26. Rost R. J. OpenGL shading language. 3-тє вид. Upper Saddle River, NJ : Addison Wesley, 2009. 743 с.

27. OpenGL programming guide: The official guide to learning OpenGL, version 2 / ред.: S. Dave, O. A. R. Board. 5-те вид. Upper Saddle River, NJ : Addison-Wesley, 2006. 838 с.

28. Bezditnyi V. M., Chebanyuk O. V. Design methodology of multimedia applications for gaming engines with a component-oriented architectural stile. PROBLEMS IN PROGRAMMING. 2024. № 2-3. С. 92–98. URL: https://doi.org/10.15407/pp2024.02-03.092 (дата звернення: 28.11.2025).

29. Terrain generation using procedural models based on hydrology / J.-D. Génevaux та ін. ACM Transactions on Graphics. 2013. Т. 32, № 4. С. 1–13. URL: https://doi.org/10.1145/2461912.2461996 (дата звернення: 28.11.2025).

30. Visualising Game Engine Subsystem Coupling Patterns / G. C. Ullmann та ін. Entertainment Computing – ICEC 2023. Singapore, 2023. С. 263–274. URL: https://doi.org/10.1007/978-981-99-8248-6_22 (дата звернення: 30.11.2025).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Кузенний Нікіта Сергійович, студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми "Програмне забезпечення систем", адреса електронної пошти ipz20bd-105@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Процедурна генерація ландшафту та рендеринг у OpenGL**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи , а також на архівування моєї роботи в базі даних цієї системи.

Дата 03.12.2025 _____ Кузенний Нікіта Сергійович (студент)
(підпис) (прізвищета ініціали)

Дата 03.12.2025 _____ Заяц Валерій Іванович (керівник)
(підпис) (прізвищета ініціали)