Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота

магістра

з теми: «Процедурна генерація цифрового представлення рельєфу місцевості та дорожньої мережі в 3D-графіці»

Виконав: здобувач вищої освіти, групи KN1-M23
спеціальності 122 Комп’ютерні науки
Станіславів Олександр Сергійович

Керівник: Смалько О. А., кандидат педагогічних наук, доцент, доцент кафедри комп’ютерних наук

Рецензент: Зеленський О. В., кандидат фізико-математичних наук, доцент, доцент кафедри математики

Кам’янець-Подільський – 2024 р.

ЗМІСТ

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню існуючих та розробці ефективних алгоритмів генерування природних і рукотворних компонентів цифрових ландшафтів для 3D-візуалізації, а також для ефективної обробки даних з подальшою їх інтеграцією у розроблюваний програмний застосунок.

Дослідження зосереджено на вдосконаленні методів моделювання природних процесів та інтеграції штучних елементів, таких як мережі доріг, у цифровий рельєф, акцентуючи увагу на ефективності обчислень і точності візуалізації. Вдосконалено окремі методи маніпулювання картою висот, розроблено модульний підхід для реалізації режимів змішування та метод інтеграції доріг до ландшафту. Також реалізовано ефективні структури даних для оптимізації обчислювальних процесів, що забезпечують безперебійну роботу та оптимізацію, і є важливим для реальних застосунків, таких як середовища віртуальної реальності та системи моделювання. Визначено та обґрунтовано найбільш корисні передові математичні підходи, включаючи дискретні гідродинамічні моделі та моделювання ерозії на основі фізики, для відтворення природних геологічних і гідрологічних процесів, завдяки яким досягається виняткова схожість синтезованих ландшафтів на реальні.

Матеріали кваліфікаційної роботи можуть бути корисними в подальших академічних дослідженнях, а також для дослідників і фахівців, які працюють в галузях моделювання та візуалізації цифрових природних середовищ, що поєднують в собі окремі антропогенні об’єкти. Це сприятиме розвитку розуміння прийомів поєднання природного рельєфу та штучно створених об’єктів, а також пошуку найбільш раціональних методів синтезу реалістичних і функціональних цифрових середовищ.

Результати цього дослідження можуть бути застосовані в проєктах розробників відеоігор, інженерів графічного програмного забезпечення, дизайнерів віртуального простору та інших суміжних галузях.

The qualification work is dedicated to researching existing and developing efficient algorithms for generating natural and man-made components of digital landscapes for 3D visualization, as well as for efficient data processing and subsequent integration into the developed software application.

The research focuses on improving methods for modeling natural processes and integrating artificial elements, such as road networks, into digital terrain, emphasizing computational efficiency and visualization accuracy. Specific methods for manipulating heightmaps have been refined, a modular approach for implementing blending modes has been developed, and a method for integrating roads into the landscape has been proposed. Efficient data structures have also been implemented to optimize computational processes, ensuring smooth operation and optimization, which is crucial for real-world applications such as virtual reality environments and simulation systems. The most effective advanced mathematical approaches, including discrete hydrodynamic models and physics-based erosion simulation, have been identified and substantiated to recreate natural geological and hydrological processes. These methods ensure exceptional realism of the synthesized landscapes, closely resembling real-world terrain.

The materials of the qualification work can be valuable for further academic research, as well as for researchers and professionals working in the fields of modeling and visualization of digital natural environments that integrate individual anthropogenic objects. This contributes to a deeper understanding of techniques for combining natural terrain with artificially created objects, as well as to identifying the most rational methods for synthesizing realistic and functional digital environments.

The results of this research can be applied to projects by video game developers, graphic software engineers, virtual space designers, and other related fields.

ВСТУП

Процедурна генерація цифрових ландшафтів представляє важливу область дослідження на перетині комп’ютерної графіки, обчислювальної математики та розробки програмного забезпечення. Ця галузь була обрана для дослідження через її всезростаюче значення в сучасних застосунках, де попит на реалістичні, масштабовані та інтерактивні цифрові середовища продовжує зростати. Здатність генерувати складний рельєф і бездоганно інтегрувати природні особливості та штучно створені об’єкти, такі як дороги та водні шляхи, є практичною необхідністю. Важливість даної сфери полягає в її потенціалі вирішення ключових проблем у створенні контенту. Ручне моделювання цифрових ландшафтів займає багато часу, є надзвичайно трудомістким і часто обмежується людськими можливостями. Процедурна генерація долає ці проблеми, використовуючи алгоритмічні підходи для автоматизації та оптимізації створення ландшафтів, забезпечуючи послідовність, масштабованість і високий рівень деталізації.

Дана сфера хоч і відзначається значними досягненнями, але вона залишається недостатньо розвиненою в кількох критичних областях, які обмежують її потенціал. Одна з головних проблем полягає в точному та ефективному моделюванні природних явищ та їх інтеграції в процедурно створені ландшафти. Хоча існуючі методи надають інструменти для створення як базових форм ландшафту, таких як пагорби, долини та плато, так і більш складніших, але вони часто не в змозі відтворити складність і реалістичність геологічних процесів, включаючи ерозію, транспортування осаду та вивітрювання із забезпеченням достатньої оптимізації обчислень. Технології, програмні інструменти, алгоритми та методи, досліджені та розроблені в ході цієї роботи, сприяють підвищенню доступності та ефективності процесу моделювання цифрових ландшафтів, дозволяючи розробникам створювати більш захоплюючі та реалістичні віртуальні середовища без значних вимог до апаратної складової комп’ютерних систем.

Об’єкт дослідження: процедурна генерація цифрових ландшафтів.

Предмет дослідження: методи та алгоритми процедурної генерації цифрового представлення тривимірного рельєфу місцевості та дорожньої мережі.

Мета кваліфікаційної роботи. Дослідження існуючих і розробка ефективних алгоритмів генерування природних та рукотворних компонентів цифрових ландшафтів для 3D-візуалізації та ефективної обробки даних з подальшою їх інтеграцією в розроблюваний програмний застосунок.

Завдання дослідження:

1. Дослідження існуючих моделей, алгоритмів і методів генерації природних особливостей місцевості та деяких рукотворних об’єктів, зокрема дорожніх мереж, з використанням сучасних підходів до обробки даних і 3D-візуалізації.
2. Розробка ефективних методів і технологій поєднання природних і штучних об’єктів у цифрових ландшафтах з акцентом на оптимізацію процесів обробки, передачі та зберігання даних для інтеграції в розроблюваний програмний застосунок.
3. Пошук найбільш ефективних алгоритмів та методик для створення реалістичних рельєфів з високою візуальною достовірністю та ефективною обробкою даних в контексті комп’ютерних систем та 3D-графіки.
4. Проєктування та реалізація генератора ландшафтів, що автоматизує процес моделювання різноманітних цифрових рельєфів з урахуванням оптимізації обчислювальних процесів і зручності користувацького інтерфейсу.

Методи дослідження: аналіз джерел, математичне моделювання, експериментальний метод, комп’ютерне моделювання, алгоритмічний метод, метод порівняльного аналізу.

Практичне значення одержаних результатів полягає у розробленому комплексному наборі методів та алгоритмів процедурної генерації цифрових

ландшафтів, що включають в себе вдосконалені методи маніпулювання картою висот, способи змішування шарів та нелінійні перетворення, які підвищують точність і контроль створення рельєфу. Крім того, інтеграція фізичних моделей, у тому числі ерозії та транспортування осаду, дозволяє реалістично моделювати природні геологічні процеси. Також розроблені алгоритми цілісної генерації дорожньої мережі і адаптивні методи згладжування забезпечують природне злиття доріг з рельєфом. Дані методи розроблені з модульною структурою, що робить їх не тільки ефективними в рамках дослідницького застосування, але й адаптованими для використання в різноманітних зовнішніх системах.

Апробація результатів дослідження відбувалася впродовж роботи трьох Міжнародних конференцій, зокрема у форматі наукових статей [20] (категорія «А»), [19] (категорія «Б»), [47] (категорія «А») та тез [24]; шляхом публікації в науково-практичному журналі «Сучасна спеціальна техніка» (категорія «Б»), у Віснику К-ПНУ імені Івана Огієнка. Фізико-математичні науки [4] та двох тез [8], [54] у Збірнику наукових праць студентів та магістрантів К-ПНУ імені Івана Огієнка.

Структура роботи. Кваліфікаційна робота складається з анотації, вступу, трьох розділів, висновків, списку використаних джерел, що містить 54 посилання, та трьох додатків.

РОЗДІЛ 1
МЕТОДИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ПРИРОДНИХ ЕЛЕМЕНТІВ ЛАНДШАФТУ ТА ДОРОЖНІХ МЕРЕЖ

1.1 Основи процедурної генерації цифрових ландшафтів

Сфера створення процедурного ландшафту в комп'ютерній графіці зазнала значних успіхів з моменту свого заснування під впливом паралельних розробок математики та обчислювальних технологій. У 1960-х і 1970-х роках процедурне моделювання рельєфу було нішевим, вимагаючи значного ручного введення. Ранні моделі, які створювалися, базувалися на простих математичних поверхнях, яким бракувало природної складності. Однак робота Бенуа Мандельброта над фракталами в 1970-х роках надихнула на використання самоподібних і рекурсивних функцій для моделювання природних особливостей, що призвело до фрактальних ландшафтів у 1980-х роках [1]. До середини 1980-х фрактальні інструменти, такі як алгоритм ромбоподібних квадратів (англ. «Diamond square algorithm»), стали основою для створення ландшафтів [2]. Швидші комп'ютери в 1990-х роках дозволили розробникам впроваджувати шумові функції, зокрема шум Перліна, як альтернативу, революціонізуючи генерацію цифрового ландшафту з покращеним контролем над текстурою та мінливістю [3]. З тих пір шумові функції стали центральними для створення реалістичних ландшафтів в іграх, симуляторах і наукових програмах.

Сама процедурна генерація відноситься до створення даних з використанням комбінацій детермінованих алгоритмів, поєднаних з випадковістю [4]. У контексті цифрових ландшафтів процедурна генерація дозволяє створювати величезні, унікальні та деталізовані ландшафти, не вимагаючи значного ручного введення. Цей підхід особливо корисний у сферах, де створення вмісту є або повторюваним, або має охоплювати великі області, наприклад, у відеоіграх, симуляторах або віртуальній реальності. На

відміну від статичних моделей рельєфу, процедурні ландшафти дозволяють створювати динамічні середовища, де ландшафти можна будувати і комбінувати, кожного разу, задаючи різні правила чи параметри генерації для створення нескінченних варіацій [5].

В ігровій індустрії процедурна генерація допомагає уникнути дорогого та трудомісткого процесу ручного проєктування кожного елементу ігрового світу, дозволяючи розробникам наповнювати ігрові світи різноманітними ландшафтами [6]. Такі проєкти, як Minecraft, No Man's Sky і серія ігор The Elder Scrolls, використовують процедурну генерацію для створення захоплюючого та непередбачуваного середовища [7]. Окрім ігор, процедурні ландшафти відіграють важливу роль у моделюванні для військової підготовки, географічних досліджень і планування реагування на катастрофи. У віртуальній реальності (VR) процедурна генерація є особливо цінною, оскільки вона дозволяє створювати великі віртуальні середовища з мінімальним використанням ресурсів, покращуючи занурення. Крім того, процедурна генерація дозволяє цим середовищам адаптуватися в режимі реального часу, реагуючи на взаємодію користувачів, створюючи більш захоплюючий та інтерактивний досвід.

Одним із найважливіших елементів створення процедурного ландшафту є карта висот, яка представляє собою двовимірний масив, кожен елемент якого відповідає значенню висоти в певній точці ландшафту. Ця структура є основою більшості цифрових рельєфів і даний підхід робить її практичним та досить ефективним способом представлення 3D-рельєфу [8]. Як правило вони генеруються за допомогою шумових функцій або інших математичних моделей для імітації природних форм ландшафтів. У процесі створення цифрового ландшафту карта висот функціонує як план, де темніші ділянки представляють нижчу висоту, а світліші – вищу. Далі це 2D-представлення перетворюється на 3D-рельєф шляхом відображення кожного пікселя на вершину в 3D-сітці, зміщуючи його по вертикалі на основі значень карти висот.

Формат карт висот може значно відрізнятися, включаючи такі формати зображень, як PNG і TIFF, які дозволяють відображати дані висоти в градаціях сірого, а також більш спеціалізовані формати, такі як файли RAW або DEM (цифрова модель рельєфу), призначені для наукових застосувань. Карти висот, створені за допомогою шуму Перліна (рис. 1.1), наприклад, створюють рельєф із більш природною та плавною зміною висот, на відміну від простішого випадкового шуму, який може виглядати надто хаотичним [9]. Змінюючи параметри шумової функції, розробники можуть контролювати частоту й амплітуду особливостей рельєфу, створюючи більш плавні або нерівні ландшафти. Під час візуалізації карта висот створює цілісну безперервну поверхню, яка може нагадувати рельєф реального світу, такі як пагорби, гори, долини чи рівнини.

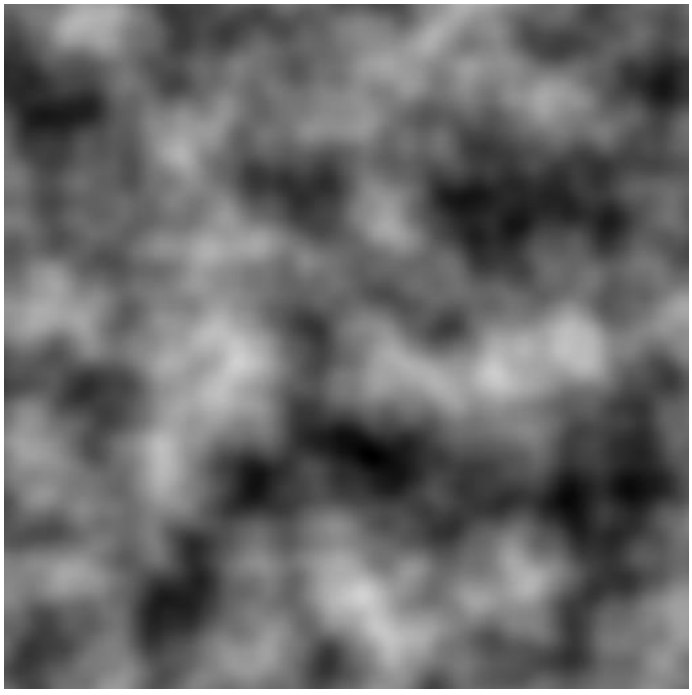


Рис. 1.1. Карта висот, згенерована шумом Перліна

Карта висот особливо цінна через свою гнучкість. Її використання дозволяє нашаровувати кілька процедурних елементів для досягнення більшої реалістичності. Поєднуючи кілька карт висот або накладаючи різні шумові функції, розробники можуть імітувати складні геологічні утворення, такі як плато, скелі чи гірські хребти. Наприклад, карту висот, згенеровану за допомогою шуму Перліна, можна комбінувати з іншими типами шуму,

наприклад, фрактальним броунівським рухом, щоб представити дрібніші деталі поверхні (дрібне каміння, гравій тощо). Також структура карти висоти забезпечує ефективне зберігання даних, оскільки вона стискає складні особливості рельєфу в двовимірне представлення.

1.2 Шумові функції та їх математична реалізація

Випадковий шум

Випадковий шум забезпечує базову структуру для процедурної генерації рельєфу, оскільки вводить початковий рівень мінливості висоти по сітці. Якщо застосовувати ізольовано, випадковий шум є відносно рудиментарним, що призводить до різких, непередбачуваних перепадів висоти [10]. Незважаючи на свою простоту, він служить корисним базовим рівнем, на якому можуть спиратися більш розширені шумові функції, поєднуючи випадковість зі структурованою мінливістю.

Для генерації випадкового шуму, нехай $H(x, y)$ – функція висоти, де x і y представляють координати сітки, а N – загальна кількість точок уздовж кожної осі. Використовуючи функцію випадкового генератора на основі зерна (англ. «seed»), початкового числа, що використовується для ініціалізації генератора псевдовипадкових чисел, $R_s(x, y)$, ми забезпечуємо відтворюваність, контролюючи випадковість. Математично висота в кожній координатній сітці визначається як (1.1):

$$H(x, y) = R_s(x, y), \qquad x, y \in \{0, \dots, N-1\}. \tag{1.1}$$

Випадкові значення $R_s(x, y)$ рівномірно розподілені в межах вибраного нами діапазону, а саме $[0, 1]$, який представляє нормалізований діапазон висоти. Щоб адаптувати випадкові значення шуму для конкретних типів місцевості, вводиться додатковий коефіцієнт масштабування:

$$H(x, y) = \alpha R_s(x, y),$$

де α контролює інтенсивність висоти.

Випадковий шум також можна комбінувати в декілька шарів. Наприклад, додавання двох випадкових шарів шуму з різними масштабами дає (1.2):

$$H(x,y) = \alpha_1 R_s(x,y) + \alpha_2 R_s(2x,2y). \quad (1.2)$$

Це генерує більш складну випадкову місцевість із дрібнішими деталями.

Чисельний шум

Чисельний шум згладжує випадковий шум шляхом інтерполяції між псевдовипадковими значеннями в точках сітки, що призводить до поступових переходів по сітці. Цей тип шуму, який часто використовується для ландшафтів із ледь помітними варіаціями, дуже підходить для моделювання рівнин та пологих пагорбів.

Для його реалізації ми будуємо сітку базових значень, $G(i,j)$, присвоєну кожній цілочисельній точці сітки (i,j) на 2D-площині. Кожна точка сітки містить псевдовипадкове значення висоти в діапазоні $[0;1]$, а для визначення значення в будь-якій точці між цими точками сітки використовується функція інтерполяції $I(u,v,t)$. Для будь-якої точки (x,y) на сітці визначаємо цілі координати $i = \lfloor x \rfloor$ та $j = \lfloor y \rfloor$, разом із дробовими частинами $u = x - i$ та $v = y - j$.

Щоб досягти плавних переходів, ми інтерполюємо між чотирма значеннями сітки:

$$H(x,y) = I(I(G(i,j), G(i+1,j), u), I(G(i,j+1), G(i+1,j+1), u), v),$$

де $I(a,b,t) = a(1-t) + bt$ є лінійною інтерполяційною функцією. Досконалішим методом інтерполяції є кубічна інтерполяція Ерміта або сплайн-інтерполяція, яка може забезпечити більш плавні градієнти шляхом регулювання впливу суміжних значень сітки.

Додавання кількох частотних шарів або «октав» покращує візуальну складність. Якщо $H_o(x,y)$ представляє шум, створений в октаві o, тоді (1.3):

$$H(x,y) = \sum_{o=1}^{O} \alpha_o H_o(2^{o-1}x, 2^{o-1}y), \quad (1.3)$$

де α_o – коефіцієнт амплітуди для октави o.

На рис. 1.2 зображено згенеровану карту висот із початковою сіткою розміром 5 на 5 клітин та кількістю октав $O = 2$.

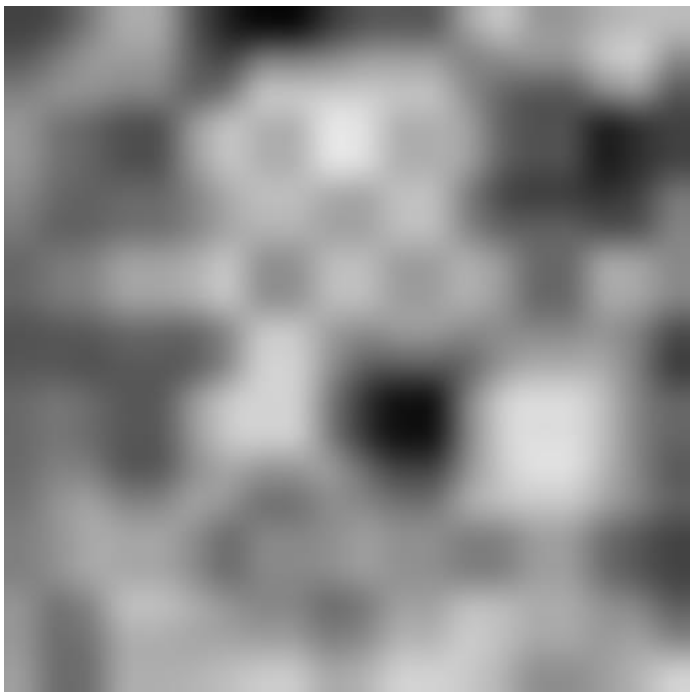


Рис. 1.2. Карта висот, згенерована чисельним шумом

Шум Перліна

Шум Перліна, метод градієнтного шуму, є більш складним, ніж чисельний шум, оскільки він вводить градієнтні вектори в точках сітки, що дозволяє будувати більш плавні та природніші переходи у висоті. Цей підхід став промисловим стандартом для створення текстур і рельєфу, які дуже нагадують природні ландшафти.

У шумі Перліна кожній точці сітки (i, j) присвоюється градієнтний вектор $\overrightarrow{g_{ij}}$ замість скалярного значення. Оскільки вектор $\overrightarrow{g_{ij}}$ є одиничним вектором, то його побудову ми будемо здійснювати шляхом генерації псевдовипадкового кута $\alpha \in [0; 2\pi)$, як скалярної величини, а вже сам вектор обчислювати як (1.4):

$$\overrightarrow{g_{ij}} = (\sin\alpha\,, \cos\alpha). \tag{1.4}$$

Для точки (x, y) у клітинці, визначеній цілочисельною сіткою координат (i, j), обчислюємо вектори відстаней:

$$\begin{cases} \overrightarrow{d_{00}} = (x - i, y - j), \\ \overrightarrow{d_{10}} = (x - (i+1), y - j), \\ \overrightarrow{d_{01}} = \big(x - i, y - (j+1)\big), \\ \overrightarrow{d_{11}} = \big(x - (i+1), y - (j+1)\big). \end{cases}$$

Внесок шуму від кожного кута комірки визначається скалярним добутком між градієнтним вектором та вектором відстані:

$$N_{ij}(x, y) = \overrightarrow{g_{ij}} \cdot \overrightarrow{d_{ij}}.$$

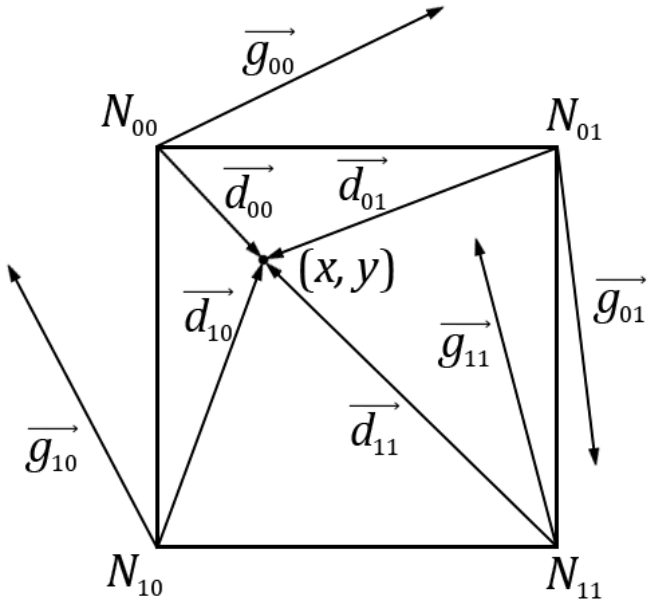


Рис. 1.3. Вектори $\overrightarrow{g_{ij}}$ та $\overrightarrow{d_{ij}}$ побудовані для точки (x, y)

Остаточне значення шуму отримується шляхом інтерполяції цих внесків за допомогою поліному п’ятого степеня (1.5):

$$f(t) = 6t^5 - 15t^4 + 10t^3 \tag{1.5}$$

для плавного змішування по сітці:

$$H(x, y) = I(I(N_{00}, N_{10}, u), I(N_{01}, N_{11}, u), v),$$

де $u = x - i$ та $v = y - j$. Щоб створити складніший рельєф, шум Перліна можна застосувати із кількома октавами. На рис. 1.1 показана карта висот із трьома октавами на основі шуму Перліна.

Шум Ворлі

Шум Ворлі, або стільниковий шум (шум Вороного), створює візерунки, які нагадують природні стільникові структури, що робить його дуже

ефективним для скелястих або фрагментованих рельєфів. Він обчислюється шляхом знаходження відстані від кожної точки сітки до найближчих довільно розміщених «особливих точок», створюючи стільниковий шаблон.

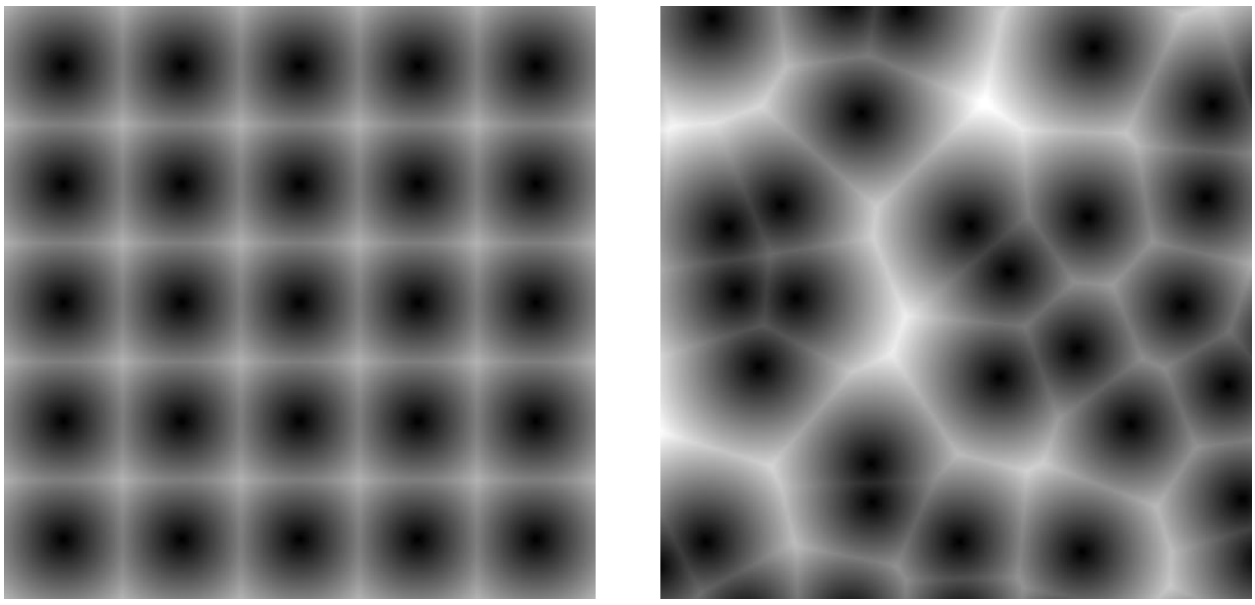
Нехай P – набір характерних точок, кожна випадково розподілена у двовимірній сітці. Для кожної точки (x, y), ми обчислюємо евклідову відстань до кожної характерної точки (x_p, y_p) по сусідству. Значення шуму Ворлі в (x, y) тоді визначається як (1.6):

$$H(x,y) = \min_{(x_p,y_p)\in P} \sqrt{(x - x_p)^2 + (y - y_p)^2} \tag{1.6}$$

Для псевдовипадкового розподілу точок у відповідних комірках сітки генеруються два псевдовипадкових числа r_x і r_y з проміжку $[-0{,}5; 0{,}5]$. Оскільки координати кожної точки всередині комірки змінюються в межах $[0; 1]$, тоді їх зміщення обчислюється:

$$\begin{cases} x = 0{,}5 + \alpha r_x, \\ y = 0{,}5 + \alpha r_y. \end{cases}$$

де $\alpha \in [0; 1]$ – коефіцієнт зміщення заданий користувачем. При $\alpha = 0$ зміщення відсутні і всі точки знаходяться строго по центру своїх комірок (рис. 1.4a), а при $\alpha = 1$ – точки розподілені максимально випадково (рис. 1.4b).



a *b*

Рис. 1.4. Карти висот, згенеровані шумом Ворлі:
a – коефіцієнт зміщення α рівний нулю;
b – коефіцієнт зміщення α рівний одиниці.

Алгоритм ромбоподібних квадратів

Алгоритм ромбоподібних квадратів (англ. «Diamond square algorithm») – це метод на основі фракталів, який рекурсивно розбиває сітку, створюючи рельєф із нерівними гірськими особливостями. Він працює шляхом коригування значень висоти на основі середніх значень сусідніх точок і додавання випадкових зсувів, що призводить до реалістичного рельєфу з самоподібними фрактальними якостями.

Для функціонування даного алгоритму, на відміну від інших підходів, карта висот повинна мати строго визначений розмір, а саме – бути квадратом зі стороною $2^n + 1$. Це пояснюється тим, що під час усіх кроків в яких здійснюється розбиття сітки (допоки розмір клітинок не стане 1 на 1) не повинно виникнути жодного дробового значення. Оскільки розмір карти висот задається довільним чином з боку користувача, виникає необхідність у створенні оптимального способу побудови карти висот довільного розміру.

Нехай x та y – ширина і висота карти висот відповідно, яку користувач хоче отримати. Розмір карти висот, що буде генеруватися повинен бути не менший, ніж максимальне значення заданої ширини і висоти, тобто нам достатньо лише підібрати найменше значення $n \in \mathbb{N}$, для якої буде виконуватися нерівність (1.7):

$$\max(x, y) \leq 2^n + 1. \tag{1.7}$$

Після чого, здійснюємо інтерполяцію одержаних значень для отримання кінцевого результату.

Сам алгоритм починається з ініціалізації псевдовипадкових значень зміщення R_s з діапазону $[-1; 1]$ в чотирьох кутах сітки $H(0, 0)$, $H(N, 0)$, $H(0, N)$, $H(N, N)$, де $N = 2^n + 1$. Далі циклічно виконуємо кроки «ромб» та «квадрат» допоки розмір кроку $s > 1$:

- *Крок «ромб»*: центральній точці кожного квадрата призначається середнє значення кутових точок плюс випадкове зміщення R_s домножене на масштаб шорсткості r:

$$H\left(A_{\text{центр.}}\right) = \frac{H(A_{00}) + H(A_{01}) + H(A_{10}) + H(A_{11})}{4} + rR_s.$$

- *Крок «квадрат»*: кожна середня точка вздовж країв перераховується шляхом усереднення суміжних значень із додаванням випадкового зміщення із масштабом шорсткості аналогічно до кроку «ромб».

Після кожного виконання тіла циклу, крок s, що на початку виконання рівний $s = N - 1$, зменшується вдвічі, а масштаб шорсткості r домножається на 2^{-h}, де $h \in [0; 1]$ – шорсткість рельєфу (більші значення дають більш згладжений рельєф, а менші – навпаки). Цей процес повторюється у все менших масштабах, створюючи більш дрібні деталі на кожному рівні рекурсії (рис. 1.5).

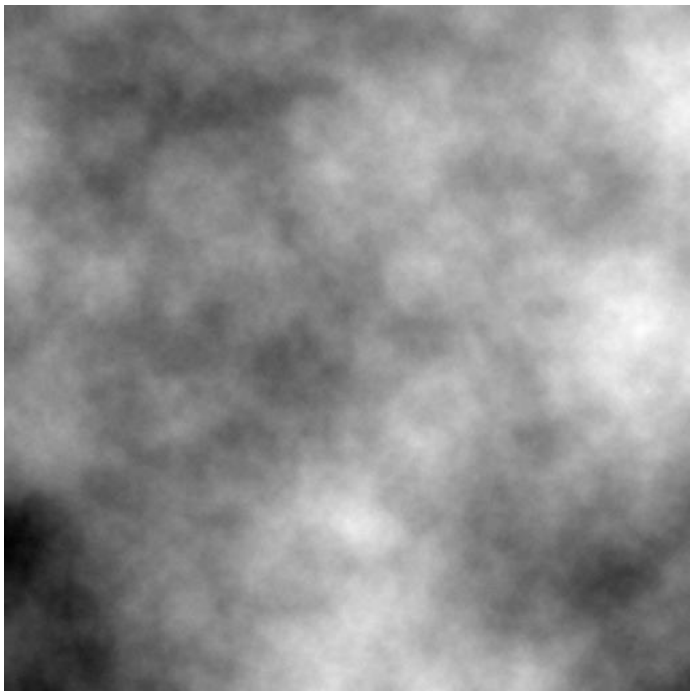


Рис 1.5. Карта висот, згенерована алгоритмом ромбоподібних квадратів

Синусоїдальна хвиля

Карти висот на основі синусоїдальних хвиль забезпечують математично простий, але візуально привабливий спосіб створення рельєфу з повторюваними хвильовими візерунками. Програмно реалізовуються два підходи, які дозволяють досягти різних ефектів залежно від того, яка форма хвилі потрібна: прямокутна чи радіальна.

Прямокутна форма будується функцією (1.8):

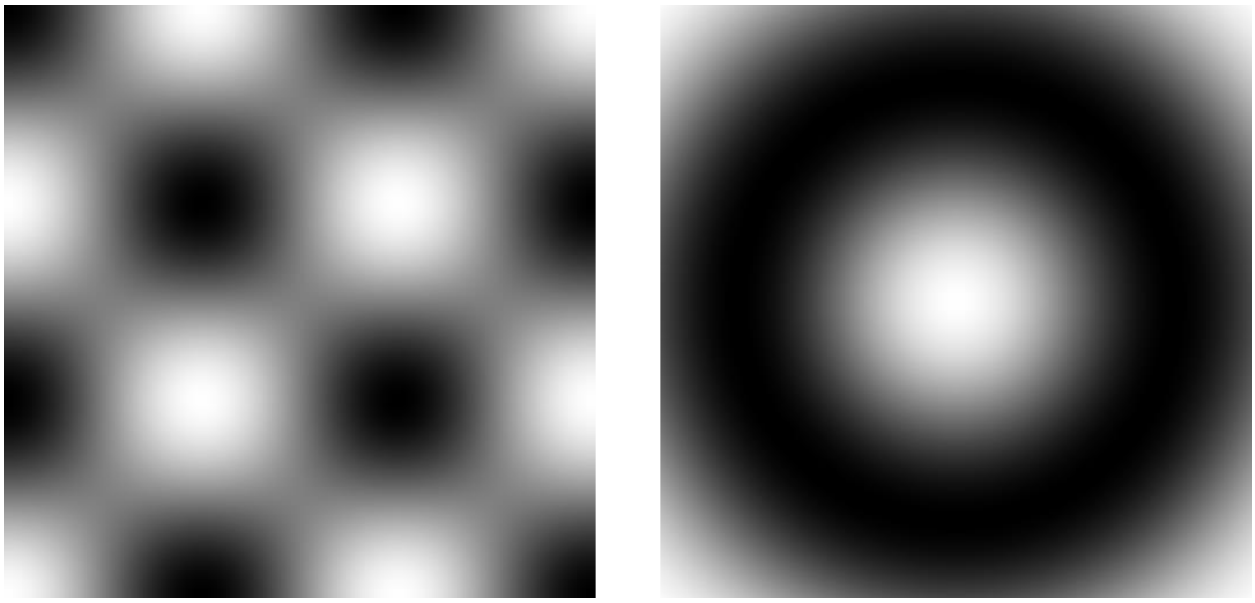
$$H(x, y) = \sin(k_x x + \varphi_x) \cdot \sin\big(k_y y + \varphi_y\big), \tag{1.8}$$

де k_x та k_y є параметрами частоти вздовж відповідних осей, а φ_x і φ_y – відповідні фазові зсуви (рис. 1.6a).

Радіальна форма, в свою чергу, будується функцією (1.9):

$$H(x, y) = \sin\left(k\sqrt{x^2 + y^2} + \varphi\right), \tag{1.9}$$

де параметр k регулює радіальну частоту, а φ – відповідний фазовий зсув (рис. 1.6b). Фрагмент коду реалізації подано у Додатку В.



a *b*

Рис 1.6. Карти висот, згенеровані із використанням синусоїдальної хвилі:
a – прямокутна форма;
b – радіальна форма.

Фрактальні алгоритми

Фрактальні алгоритми у створенні рельєфу використовують рекурсивні процеси для створення самоподібних візерунків, часто шляхом шарування шумових функцій на різних частотах і амплітудах. Наприклад, дробовий броунівський рух (fBm) повторює шум Перліна на вищих частотах із зменшенням амплітуди (1.10):

$$H(x, y) = \sum_{i=0}^{O} \alpha^i P\big(\beta^i x, \beta^i y\big), \tag{1.10}$$

де $P(x, y)$ – шум Перліна, α контролює спад амплітуди, а β контролює збільшення частоти. Фрактальні підходи корисні для створення місцевостей зі складними, надзвичайно деталізованими характеристиками.

Гібридний ландшафт

Генерація гібридного рельєфу поєднує численні шумові функції, підвищуючи реалістичність за рахунок комбінування різноманітних характеристик поверхонь. Наприклад, поєднання шуму Перліна з шумом Ворлі може створити рельєф із гладкими пагорбами, перемежованими скелястими виступами (1.11):

$$H(x, y) = w_1 P(x, y) + w_2 W(x, y), \quad (1.11)$$

де $P(x, y)$ і $W(x, y)$ – шуми Перліна та Ворлі відповідно, а w_1 і w_2 – ваги змішування. Крім того, даний підхід можна значно розширити завдяки імплементації режимів змішування, таких як множення, освітлення, лінійне комбінування тощо, які окрім зміни ступеню непрозорості дозволяють глибоко налаштовувати вплив окремих значень карти висот на кінцевий результат.

1.3 Способи інтеграції дорожніх мереж в цифрові середовища

Ручне та процедурне розміщення дорожніх мереж

Розміщення дорожніх мереж у цифровому середовищі місцевості може бути досягнуто як ручними, так і процедурними методами. Ручне розміщення часто використовується, коли потрібен точний контроль, наприклад, у випадках, коли дороги мають відповідати певним географічним особливостям ландшафту. У цьому методі дизайнери розміщують вузли та сегменти доріг вручну, вказуючи координати для кожного сегмента дороги для забезпечення точності. Наприклад, задана карта висот місцевості $H(x, y)$, де кожна координата (x, y) має певну висоту h, ручне розміщення передбачає визначення кожного сегмента (x_i, y_i) таким чином, щоб дорога дотримувалася потрібного градієнту $\frac{\partial h}{\partial x}$ чи $\frac{\partial h}{\partial y}$ для імітації реалістичних схилів [11].

Процедурне розміщення, з іншого боку, дозволяє автоматично створювати дорожні мережі на основі попередньо визначених критеріїв. Для цього підходу застосовуються математичні обмеження та правила для імітації реалістичних планів доріг. Базовий процедурний підхід може створити дороги, які з'єднують ключові місця за допомогою мінімального остовного дерева (MST) над вузлами, що представляють важливі точки на місцевості, такі як пагорби або переправи через річку [12]. Нехай маємо граф $G = (V, E)$, де V представляє вузли, а E – потенційні сегменти дороги, алгоритм MST мінімізує загальну довжину L доріг шляхом ітеративного додавання найкоротших можливих ребер без утворення циклів, які можна представити як (1.12):

$$L = \sum_{(u,v)\in E'} d(u,v), \qquad E' \subset E, \tag{1.12}$$

де $d(u, v)$ – евклідова відстань між вузлами u та v.

Алгоритми маршрутизації та генерації доріг: A та діаграми Вороного*

Для маршрутизації та створення доріг зазвичай застосовуються такі алгоритми, як A* та діаграми Вороного, завдяки їх ефективності та адаптованості до різних обмежень рельєфу. A* (A-зірка) – це алгоритм пошуку шляху, який використовує евристику для пошуку найкоротшого шляху між двома точками на сітці рельєфу. Цей алгоритм особливо ефективний для генерації дорожньої мережі на місцевості, оскільки він може враховувати зміни відстані та висоти. У A* кожен вузол n на сітці рельєфу місцевості оцінюється на основі функції витрат (1.13):

$$f(n) = g(n) + h(n), \tag{1.13}$$

де $g(n)$ представляє вартість від початкового вузла до вузла n, а $h(n)$ – евристична оцінка вартості з n до цільового вузла [13]. Оцінка $h(n)$ враховує евклідову або манхеттенську відстань, дозволяючи A* ефективно визначати пріоритет вузлів, які ведуть до цілі.

Діаграми Вороного пропонують альтернативний підхід для створення дорожньої мережі, особливо корисний при створенні доріг між кількома

місцями, наприклад містами. На діаграмі Вороного рельєф ділиться на регіони, кожен з яких пов'язаний з найближчим місцем. Кордони між ділянками природним чином утворюють шляхи, які можна використовувати як траси доріг, дотримуючись особливостей місцевості, де це необхідно [14]. Комірка Вороного для ділянки P_i визначається як (1.14):

$$V(P_i) = \{Q \in \mathbb{R}^2 \big| d(Q, P_i) < d(Q, P_j), \forall j \neq i\}, \quad (1.14)$$

де $d(Q, P_i)$ – відстань від будь-якої точки Q до ділянки P_i. Ця структура дає значну перевагу для з'єднання кількох місць у мережі, одночасно оптимізуючи відстані маршруту [15].

Використання дорожніх карт

Розміщення доріг на основі дорожньої карти використовує наявні дані про дороги, для імітації реалістичних мереж у цифрових ландшафтах. Цей підхід використовує точки і шляхи з географічних інформаційних систем (ГІС) для накладання мереж доріг на місцевість. Нехай маємо набір точок $P = \{p_1, p_2, \dots, p_n\}$, де кожна точка p_i являє собою координати сегмента дороги, алгоритм дорожньої карти розміщує дороги шляхом інтерполяції між цими точками, коригуючи висоту місцевості. Плавну криву $C(t)$, що представляє дорогу, можна згенерувати за допомогою кубічної інтерполяції, гарантуючи, що дорога відповідає існуючим географічним особливостям (1.15):

$$C(t) = \sum_{i=1}^{n} b_i(t) p_i, \quad (1.15)$$

де b_i – базисні функції, зазвичай B-сплайнові або кубічні, які забезпечують плавні переходи між сегментами.

Вплив рельєфу місцевості на планування та розміщення доріг

Вплив рельєфу на розміщення доріг є одним із критичних факторів у створенні цифрових доріг, оскільки нахил, висота та перешкоди безпосередньо впливають на розміщення дороги. Обмеження місцевості моделюються математично шляхом включення обмежень нахилу та даних про висоту

безпосередньо в алгоритм створення дороги. Наприклад, нахил S між двома точками (x_1, y_1, h_1) та (x_2, y_2, h_2) заданий як (1.16):

$$S = \frac{|h_2 - h_1|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}, \quad (1.16)$$

повинен залишатися нижче порогового, наперед заданого користувачем значення для забезпечення оптимізації генерованого шляху [16, 17]. Окрім нахилу, алгоритм враховує обмеження висоти та параметри різних типів місцевості (наприклад, рівнинна, гірська або берегова зона) для формування логічних маршрутів, які виглядають природно. У симуляціях також часто враховуються обхідні шляхи, які дозволяють уникати різких перепадів висоти та зберігати цілісність цифрової карти доріг.

Висновки до розділу 1

Процедурна генерація цифрових ландшафтів перетворилася з базових методів візуалізації на складні, керовані алгоритмами середовища. Центральним у цьому є використання карт висот, 2D-масивів, що зберігають значення висоти, де різні шумові функції задають структурну різноманітність, що дозволяє створювати масштабовані, динамічні та детальні цифрові ландшафти.

Найкращі варіанти використання цих алгоритмів підкреслюють їхні унікальні переваги. Випадковий шум дозволяє створювати пересічену місцевість для абстрактних або чужих пейзажів, що ідеально підходить для фантастичних середовищ. Чисельний шум із його більш плавними переходами часто використовується в біомах для створення пагорбів або плато. Шум Перліна імітує більш природні, поступові зміни висоти, що робить його основним елементом для пейзажів в іграх з відкритим світом і VR. Шум Ворлі генерує стільникові візерунки, які ідеально підходять для імітації еродованих ландшафтів або регіонів із чіткими нерівними деталями, такими як скелі та кам’яниста місцевість. Алгоритм ромбоподібних квадратів забезпечує

генерацію фракталоподібних рельєфів, який підходить для великих гірських хребтів або островів зі скелястими вершинами, часто використовується в симуляції, де потрібні різноманітні, непередбачувані ландшафти. Синусоїдальні хвилі, в свою чергу, можна застосовувати для створення коливальних візерунків, таких як піщані дюни або океанські хвилі, додаючи повторювані, ритмічні зміни висоти до однорідної місцевості.

Включення дорожніх мереж у процедурно створені рельєфи поєднує в собі ручні та алгоритмічні підходи, забезпечуючи реалістичний і функціональний вигляд доріг у різноманітних ландшафтах. Ручне розміщення дозволяє дизайнерам вказувати точні маршрути, корисні для створення фокусних шляхів на ігрових картах або під час моделювання, тоді як процедурне розміщення використовує алгоритми для динамічної адаптації доріг до навколишнього середовища. Алгоритми маршрутизації, такі як A*, забезпечують ефективне визначення найкоротших і доступних маршрутів через місцевість, тоді як діаграми Вороного допомагають створювати мережі, розділяючи регіони на природні шляхи, що ідеально підходить для органічних, схожих на міста структур. Використання дорожніх карт дозволяє наявним дорожнім даним інформувати про розміщення, підвищуючи реалістичність шляхом розміщення доріг у знайомих планах.

Ці підходи створюють складні, правдоподібні мережі доріг, які плавно інтегруються з процедурно згенерованим рельєфом, збагачуючи як візуальну автентичність, так і функціональність віртуальних світів і симуляційних моделей.

РОЗДІЛ 2
МЕТОДИ ТА ПРИЙОМИ НАДАННЯ РЕАЛІСТИЧНОСТІ ГЕНЕРОВАНИМ ЛАНДШАФТНИМ СЕРЕДОВИЩАМ

2.1 Способи вдосконалення рельєфних форм місцевості у цифровому представленні

У цифровому представленні місцевості підвищення реалістичності та точності форм рельєфу має важливе значення для цілісних і візуально привабливих 3D-ландшафтів. В межах даної кваліфікаційної роботи пропонується ряд підходів для вдосконалення генерації рельєфних форм місцевості. Ці вдосконалення передбачають математичні перетворення та методи створення, обробки та комбінування різних карт висот. Ретельно впроваджуючи ці підходи, можна одержати цифрові ландшафти, які краще імітують природні структури рельєфу.

Нелінійне масштабування

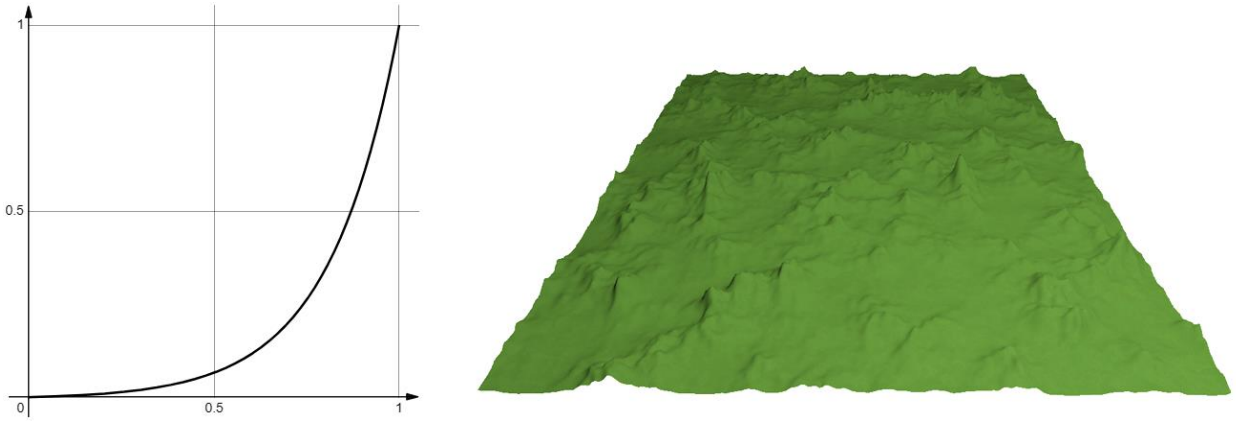
Метод нелінійного масштабування дозволяє уточнювати характеристики рельєфу на картах висот при створенні ландшафту. Застосовуючи нелінійні перетворення до значень висоти, ми отримуємо точний контроль над її розподілом, уможливлюючи покращення тонких особливостей рельєфу. Ці коригування дозволяють створити рельєф місцевості з більш реалістичними пропорціями, наприклад пологі долини, гори з гострими вершинами або хвилясті рівнини.

Одним з ефективних методів є застосування функції експоненціального коригування, а саме (2.1):

$$H_{\text{рез.}}(x,y) = \begin{cases} \dfrac{\alpha^{H(x,y)} - 1}{\alpha - 1}, & \alpha \neq 1 \\ H(x,y), & \alpha = 1 \end{cases} \qquad (2.1)$$

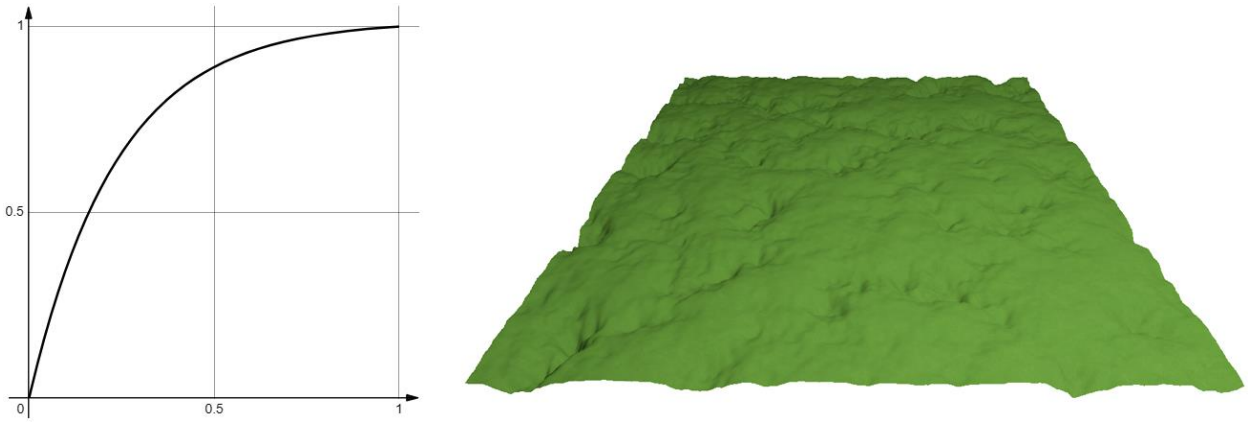
де α – визначений користувачем параметр ($\alpha > 0$), що контролює розподіл висоти. Ця функція, що регулює крутизну вершин і гладкість долин,

враховує різні характеристики рельєфу. Коли $\alpha > 1$ – отримана місцевість демонструє крутіші схили та гостріші вершини (рис. 2.1), коли $0 < \alpha < 1$ – рельєф виглядає більш гладким з більш поступовими переходами (рис. 2.2). Ця гнучкість дає змогу створювати гірські ландшафти, пагорби чи широкі рівнини шляхом простого налаштування параметра α з боку користувача.



a *b*

Рис. 2.1. Згенерований ландшафт при $\alpha > 1$:
a – графік функції (2.1) при $\alpha = 200$;
b – відповідний згенерований ландшафт.



a *b*

Рис. 2.2. Згенерований ландшафт при $0 < \alpha < 1$:
a – графік функції (2.1) при $\alpha = 0{,}015$;
b – відповідний згенерований ландшафт.

Режими змішування

Режими змішування (англ. «Blending modes») пропонують структурований спосіб шарувати зображення, зокрема в контексті розроблюваного програмного застосунку їх можна застосувати при накладанні карт висот, контролюючи, як одна карта висот взаємодіє з іншою, регулюючи

їхні відповідні внески за допомогою певних математичних операцій із регульованою непрозорістю. Налаштування непрозорості дозволяє точно налаштувати, наскільки сильно кожен шар впливає на кінцевий результат, дозволяючи налаштувати змішування для створення складного рельєфу. На відміну від базової лінійної інтерполяції, яка рівномірно змішує карти висот, режими змішування забезпечують складну, локалізовану взаємодію між шарами, імітуючи моделі ерозії, вплив погодних умов або геологічні шарування. Порівняння впливів усіх впроваджених режимів змішування на форму ландшафту подано у Додатку А.

Розглянемо дві карти висот $H_1(x, y)$ та $H_2(x, y)$, де $H_1(x, y)$ представляє базову карту висот, а $H_2(x, y)$ – вторинна карта, яка буде змішуватися. Обидві карти висот мають нормалізовані значення між 0 і 1. Кожен режим змішування застосовується з непрозорістю $\alpha \in [0; 1]$, що визначає вплив $H_2(x, y)$ на остаточну карту висот $H_{\text{рез.}}(x, y)$. Тобто коефіцієнт непрозорості α застосовується для керування інтенсивністю змішування $H_2(x, y)$ із $H_1(x, y)$. Математично результат $H_{\text{рез.}}(x, y)$ можна представити у вигляді лінійної інтерполяції між $H_1(x, y)$ та вибраної операції змішування $B(H_1, H_2)$, застосованої до $H_1(x, y)$ та $H_2(x, y)$ (2.2):

$$H_{\text{рез.}}(x, y) = (1 - \alpha)H_1(x, y) + \alpha B(H_1, H_2). \quad (2.2)$$

Розглянемо деякі зі режимів змушування.

Режим «Нормальний» (англ. «Normal») передбачає звичайне накладання $H_2(x, y)$ на $H_1(x, y)$ з регульованою непрозорістю. Це найпростіший режим, де $H_2(x, y)$ лише поступово замінює $H_1(x, y)$ по мірі збільшення коефіцієнту непрозорості. Формула даного режиму наступна (2.3):

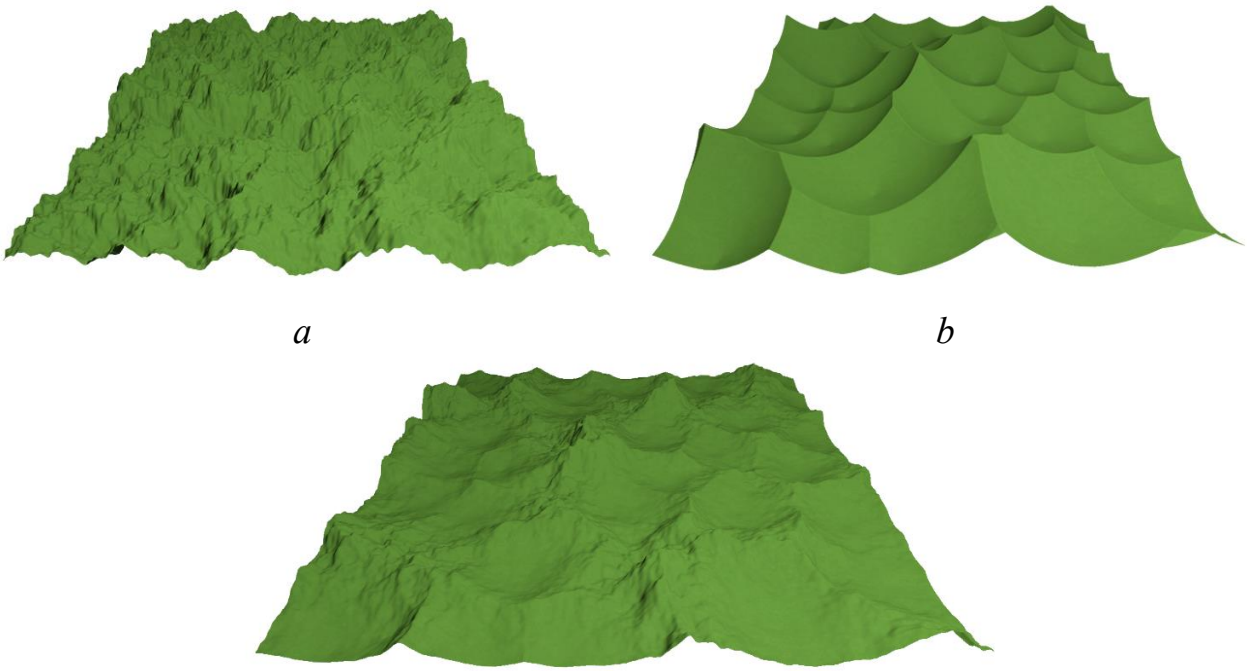
$$B(H_1, H_2) = H_2. \quad (2.3)$$

Підставивши (2.3) у формулу результуючої карти висот (2.2) одержуємо звичайну лінійну інтерполяцію між $H_1(x, y)$ та $H_2(x, y)$ (2.4):

$$H_{\text{рез.}}(x, y) = (1 - \alpha)H_1(x, y) + \alpha H_2(x, y). \quad (2.4)$$

Режим «Множення» (англ. «Multiply»), в свою чергу, підсилює деталі висоти шляхом множення відповідних значень $H_1(x,y)$ і $H_2(x,y)$, що підкреслює області, де обидві карти висот мають високі значення і приглушує ділянки, де хоча б одна із карт має низькі значення висоти, таким чином створюючи різкіші піки, більш згладжені рівнини, а також дозволяє ефективно поєднувати особливості рельєфу кожної з карти висот (рис. 2.3). Даний режим змішування має наступну формулу (2.5):

$$B(H_1, H_2) = H_1 \cdot H_2. \tag{2.5}$$



a *b* *c*

Рис. 2.3. Ландшафт, отриманий шляхом поєднання двох карт висот з режимом змішування «Множення»:
a – ландшафт, згенерований за допомогою шуму Перліна;
b – ландшафт, згенерований за допомогою шуму Ворлі;
c – результат накладання шуму Ворлі на шум Перліна із режимом змішування «Множення».

Режим «Лінійне висвітлювання» (англ. «Linear Dodge») дозволяє збільшувати загальну висоту рельєфу шляхом підсумовування відповідних значень з обох карт висот. Цей режим корисний для шарування місцевості, де кожен наступний шар піднімає ландшафт, створюючи ефект накопичення. Формула даного режиму змішування має вигляд (2.6):

$$B(H_1, H_2) = \begin{cases} H_1 + H_2, & H_1 + H_2 \leq 1 \\ 1, & H_1 + H_2 > 1 \end{cases} \quad (2.6)$$

Оскільки результуюча карта висот $H_{\text{рез.}}(x, y)$ повинна бути в межах від 0 до 1, тоді і значення $B(H_1, H_2)$ повинні також належати проміжку $[0; 1]$.

Режим «Накладання» (англ. «Overlay») поєднує в собі ефекти режимів «Множення» та «Екран», посилюючи контрасти шляхом затемнення темних областей і освітлення світлих областей карт висот, що дозволяє підкреслити як долини, так і вершини на місцевості. Цей режим задається такою формулою (2.7):

$$B(H_1, H_2) = \begin{cases} 2H_1H_2, & H_1 < 0{,}5 \\ 1 - 2(1 - H_1)(1 - H_2), & H_1 \geq 0{,}5 \end{cases} \quad (2.7)$$

Такий підхід, за рахунок аналізу базової карти висот $H_1(x, y)$ і застосування на основі її значень відповідних режимів змішування, забезпечує створення збалансованого, але детального профілю місцевості, де підкреслюються як глибина, так і висота рельєфу.

Описані режими змішування реалізуються за допомогою методів, що включають в себе базову функцію змішування для повторення сіток карти висот із застосуванням вибраної математичної формули в кожній точці сітки (x, y) і диспетчера режимів, який вибирає відповідну операцію змішування – нормальний, множення, лінійне висвітлювання тощо – на основі заданих користувачем параметрів. Кожен режим змішування інкапсульовано в окремому методі для забезпечення модульності, що дозволяє динамічно підключати логіку змішування до базової функції. Налаштування непрозорості обробляється методом лінійної інтерполяції, поєднуючи базову карту висот із змішаним результатом. Ця модульна структура підтримує масштабованість і повторне використання, забезпечуючи ефективне й гнучке впровадження на різних шарах карти висот.

2.2 Фізично-обґрунтовані методи створення реалістичних ландшафтів

У природі такі геологічні процеси, як тектонічні рухи, вулканічна діяльність і вивітрювання, змінюють ландшафт протягом тисяч або навіть мільйонів років. Симуляція цих процесів у обчислювальному контексті спрямована на створення реалістичного ландшафту шляхом застосування фізичних і математичних моделей, які імітують ці природні сили. Тектонічний рух, одну з основних сил геологічних перетворень, можна моделювати, змінюючи висоту та форму рельєфу. Для простоти побудови моделі достатньо припустити, що тектонічні сили застосовують лінійну або синусоїдальну функцію зміщення до частин сітки карти висот [18]. Припустимо, у нас є карта висот $H(x, y)$, визначена над 2D площиною. Тектонічне підняття або западину можна представити як (2.8):

$$H_{\text{рез.}}(x, y) = H(x, y) + D(x, y, t), \tag{2.8}$$

де $D(x, y, t)$ являє собою зміщення в часі t. Функція зміщення може змінюватися залежно від геологічних особливостей: наприклад, гори можуть зазнавати лінійного зсуву вгору, тоді як долини зазнають періодичних коливань через ерозію та осадження.

Вулканічну активність можна змоделювати локальними змінами висоти та розподілу матеріалу. Модель вулканічного утворення починається з додавання шарів лави в кругових або еліпсоїдних областях на карті висот. Висота зростає в радіальному градієнті, причому центр має найбільшу висоту. Визначивши точку (x_c, y_c), як центр вулкана, одержуємо функцію (2.9):

$$f(x, y) = e^{-\left(\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}\right)}. \tag{2.9}$$

Сама ж функція висоти приймає форму (2.10):

$$H_{\text{рез.}}(x, y) = H(x, y) + A \cdot e^{-\left(\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}\right)}, \tag{2.10}$$

де A керує максимальною висотою лавового купола, а σ – його поширення. Такий підхід дозволяє імітувати Гауссівську зміну висоти навколо центру вулкана (рис. 2.4).

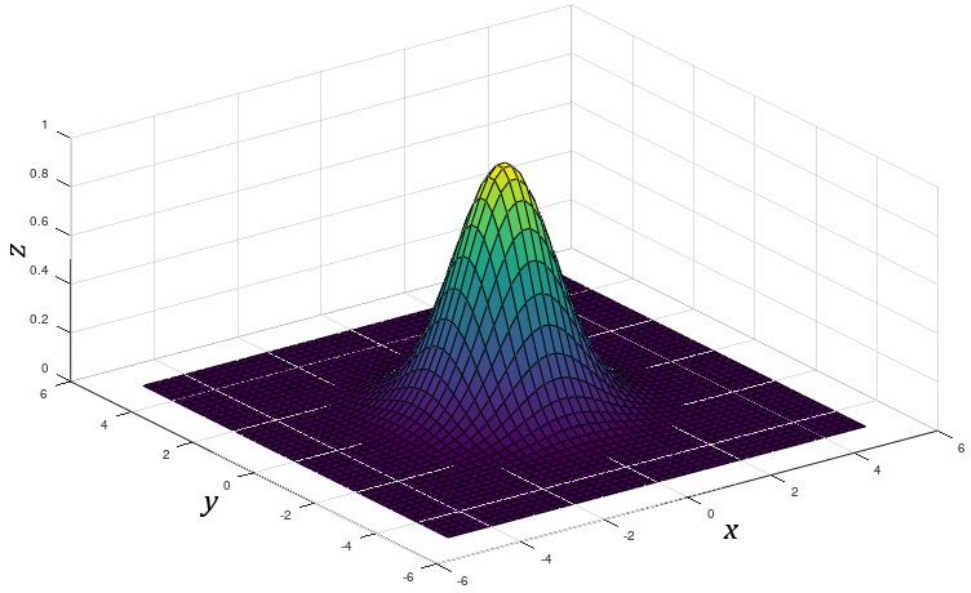


Рис 2.4. Графік функції (2.9) з центром в точці $(0; 0)$ *та* $\sigma = 1$

Хімічне вивітрювання, в свою чергу, дозволяє моделювати розчинення гірських порід і може бути представлено реакційно-дифузійними моделями. Ці моделі використовують диференціальні рівняння для врахування хімічних реакцій і дифузії агентів вивітрювання. Наприклад, якщо припустити, що концентрація реакційноздатного агента $C(x, y, t)$, що дифундує по місцевості, концентрацію в будь-якій точці можна описати як (2.11):

$$\frac{\partial C}{\partial t} = D \cdot \nabla^2 C - kC, \tag{2.11}$$

де D – коефіцієнт дифузії, k – швидкість реакції. Потім висота ландшафту оновлюється на основі взаємодії між $C(x, y, t)$ та властивостями матеріалу $H(x, y)$. Даний підхід фіксує поступову ерозію поверхонь, які піддаються впливу реактивних елементів, таких як кислотні дощі або ґрунтові води з високим вмістом мінеральних речовин [19].

Взаємодія води з ландшафтом

Взаємодія води з рельєфом є ключовим у формуванні природних ландшафтів. Такі процеси, як утворення річок, транспортування осадів і

гідравлічна ерозія, відповідають за постійну трансформацію форм рельєфу. При обчислювальній генерації рельєфу взаємодію води можна моделювати за допомогою принципів гідродинаміки, які моделюють рух і вплив води на поверхню рельєфу.

Щоб імітувати рух води по рельєфу, ми починаємо з представлення як рельєфу, так і висоти води. Нехай $H(x, y)$ позначає висоту місцевості в точці (x, y), а $h(x, y, t)$ – висота води в цій самій точці в момент часу t. Основним методом моделювання динаміки води на місцевості є використання рівнянь мілководдя (англ. «shallow-water equations»), які спрощують рівняння Нав'є-Стокса, припускаючи, що горизонтальні розміри водного потоку набагато більші за його вертикальні [20]. Це спрощення враховує ключову поведінку води, наприклад річковий потік, зберігаючи обчислювальну ефективність.

Рівняння мілководдя складаються з двох частин: рівняння неперервності, яке описує збереження маси, і рівняння імпульсу, що описує потік води. Рівняння неперервності має вигляд (2.12):

$$\frac{\partial h}{\partial t} + \nabla h\vec{v} = 0, \tag{2.12}$$

де h – висота води, $\vec{v}$ – вектор швидкості потоку води. Це рівняння, по суті, відстежує швидкість, з якою вода надходить або виходить з певної області, забезпечуючи збереження маси. Рівняння імпульсу, яке включає сили тяжіння, тертя та градієнти тиску, визначає те, як вода тече по місцевості. Це можна представити як (2.13):

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} = -g \cdot \nabla h + u\nabla^2\vec{v}, \tag{2.13}$$

де g – прискорення вільного падіння, а u – коефіцієнт в'язкості [21]. Ці рівняння керують змінами швидкості та висоти води з часом, дозволяючи їй текти вниз, накопичуватися в водоймах або розтікатися на плоских поверхнях.

Водний потік чинить зсувну силу на поверхню місцевості, що призводить до ерозії та переносу осаду. Ерозія моделюється шляхом видалення невеликої частини висоти з рельєфу в точках, де потік води інтенсивний.

Швидкість ерозії $E(x, y, t)$ є функцією швидкості та об'єму води. Наприклад, ерозія може бути прямо пропорційною швидкості води (2.14):

$$E(x, y, t) = k \cdot |\vec{v}| \cdot h, \tag{2.14}$$

де k є константою, яка контролює інтенсивність ерозії [22]. Коли потік води досягає більш пологого схилу або зовсім припиняється, може відкладатися осад, який переноситься водою. Це відкладення осаду можна змоделювати шляхом зменшення висоти води та збільшення висоти рельєфу на основі кількості осаду, який несла вода.

Коли вода рухається ландшафтом, вона прорізає річкові русла, відкладає осад у долинах і формує схили. Завдяки постійному застосуванню рівнянь мілководдя та правил ерозійних відкладень рельєф динамічно розвивається, імітуючи довготерміновий вплив річок і струмків на ландшафти. Це моделювання створює реалістичні річкові долини, дельти та моделі ерозії, характерні для рельєфу природної форми.

Гідравлічна ерозія на основі частинок

Гідравлічна ерозія, зокрема її моделювання на основі руху частинок, моделює механічний вплив окремих крапель води на ландшафт. Цей процес імітує те, як дощ і поверхневий стік витісняють осад і транспортують його по місцевості. Гідравлічна ерозія на основі частинок особливо ефективна при процедурному створенні рельєфу, оскільки вона працює на гранульованому рівні, дозволяючи створювати детальні та різноманітні схеми ерозії, які імітують природне вивітрювання та водну ерозію.

Алгоритм гідравлічної ерозії на основі частинок починається зі створення частинок на поверхні місцевості. Кожна частинка являє собою краплину води з масою, швидкістю та здатністю переносити осад. Початковий стан кожної частинки визначається її масою m, початковою швидкістю v та положенням (x, y). Маса частинки пропорційна кількості води, яку вона представляє, що впливає на її здатність розмивати та транспортувати осад.

Швидкість $\vec{v}$ задається на основі сили тяжіння, гарантуючи, що частинка буде рухатися вниз, якщо не буде перешкод [23].

Після ініціалізації кожна частинка рухається по поверхні на основі принципів класичної механіки. Сила, яка діє на частинку, є сумою сили тяжіння, поверхневого тертя та потенційної взаємодії з особливостями рельєфу. Рівняння, що визначає прискорення $\vec{a}$ має вигляд (2.15):

$$\vec{a} = \frac{\vec{F}}{m}, \tag{2.15}$$

де сила $\vec{F}$ включає силу гравітації та силу тертя. Положення та швидкість частинки оновлюються на кожному кроці часу, у результаті чого формується вектор руху $\vec{v}$, що веде її по ландшафту.

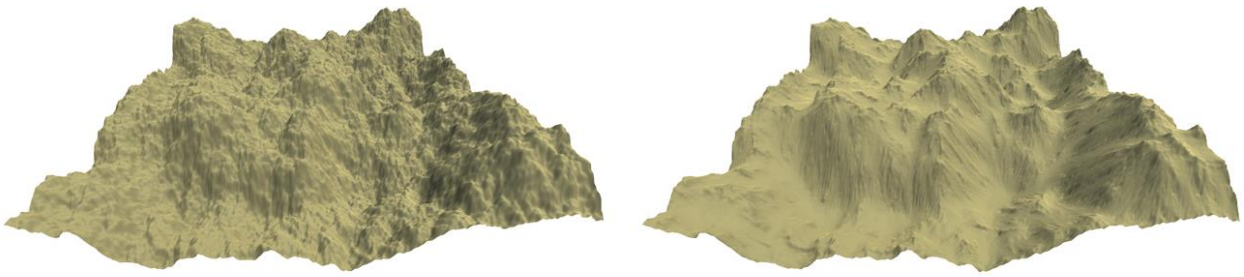
Коли частинка рухається, вона взаємодіє з рельєфом, збираючи або відкладаючи осад. Перенесення осаду залежить від крутизни рельєфу та швидкості частинок. Якщо частинка рухається вниз по крутому схилу, вона збирає осад відповідно до функції масообміну (2.16):

$$S(x, y) = \alpha \cdot |\vec{v}| \cdot \Delta h(x, y), \tag{2.16}$$

де $S(x, y)$ являє собою кількість зібраного осаду, α – ерозійна константа, $\vec{v}$ – швидкість частинки, $\Delta h(x, y)$ – різниця висот між поточним і попереднім положеннями. Чим швидше рухається частинка, тим більше осаду вона може розмивати та переносити. І навпаки, якщо частинка сповільнюється, вона відкладає осад залежно від кількості зібраного осаду та залишкової швидкості.

Після певної кількості взаємодій або коли частинка досягає рівноважного стану (наприклад, коли її швидкість надто мала), частинка випаровується, що означає втрату води через умови навколишнього середовища. Цей процес випаровування зменшує масу частинки, впливаючи на її здатність переносити осад і знижуючи її потенціал до подальшої ерозії. Життєвий цикл частинки закінчується, коли вона стає надто малою для продовження або виходить за межі місцевості, закінчуючи свій ерозійний вплив [24].

За допомогою повторюваних ітерацій частинки моделюють поступове руйнування особливостей рельєфу, вирізаючи канали, згладжуючи нерівні ділянки та відкладаючи осад на рівнинних ділянках. Це поетапне переміщення відкладень імітує дуже деталізовані та реалістичні особливості реального ландшафту, що відображає кумулятивний вплив гідравлічної ерозії (рис. 2.5).



a *b*

Рис. 2.5. Приклад моделювання гідравлічної ерозії:
a – ландшафт, згенерований шляхом поєднання різних шумових функцій;
b – результат моделювання гідравлічної ерозії на основі ландшафту a.

2.3 Інтеграція та адаптація дорожніх мереж до топографії рельєфу

У той час як методи створення рельєфу місцевості часто наголошують на таких природних об’єктах, як гори, долини та річки, створення рукотворних споруд, таких як дороги, представляє певні труднощі. Дороги повинні плавно інтегруватися з рельєфом, що лежить під ними, зберігаючи візуальну безперервність та структурну правдоподібність.

Запропонований в межах магістерського дослідження алгоритм для створення доріг приймає як вхідні дані карту висот, що представляє рельєф, і набір визначених користувачем контрольних точок, що визначають шлях дороги. Використовуючи сплайн Catmull-Rom для плавної інтерполяції між точками, алгоритм вирівнює рельєф уздовж траєкторії дороги та поєднує краї дороги з навколишнім ландшафтом, щоб забезпечити природний перехід.

Результатом є модифікована карта висот, яку можна безпосередньо використовувати для створення 3D-моделей з інтегрованою мережею доріг.

Дано карту висот $H(x, y)$, яка визначається як двовимірний масив чисел з плаваючою комою, де x і y – координати на площині, а значення $H(x, y)$ представляє висоту в цій точці, мета полягає в тому, щоб створити дорогу, яка слідує шляху, визначеному набором контрольних точок $P = \{P_1, P_2, \dots, P_n\}$. Кожна контрольна точка P_i представляється у вигляді двовимірного вектора $P_i = (x_i, y_i)$. Дорога повинна проходити через усі контрольні точки, підтримувати визначену користувачем ширину w та мати плавний перехід в навколишню місцевість.

Щоб створити безперервну та гладку дорогу через контрольні точки, визначені користувачем, ми використовуємо сплайн Catmull-Rom. Цей тип сплайна особливо добре підходить для програм, що вимагають плавної інтерполяції, оскільки він забезпечує C^1-неперервність, що робить переходи між послідовними сегментами візуально плавними.

Сплайн Catmull-Rom визначається для чотирьох послідовних контрольних точок P_{i-1}, P_i, P_{i+1}, P_{i+2} і параметризується змінною $t \in [0; 1]$. Інтерпольована позиція $R(t)$ уздовж сплайна визначається як (2.17):

$$R(t) = \frac{1}{2}(2P_i + (P_{i+1} - P_{i-1})t + (2P_{i-1} - 5P_i + 4P_{i+1} - \\ -P_{i+2})t^2 + (3P_i - P_{i-1} - 3P_{i+1} + P_{i+2})t^3). \quad (2.17)$$

Це рівняння обчислюється ітераційно для кожного сегмента між контрольними точками, створюючи серію точок $R_k = (x_k, y_k)$, які визначають шлях дороги.

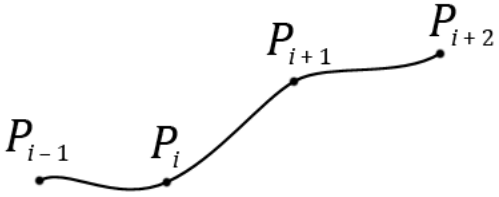


Рис. 2.6. Сплайн-інтерполяція Catmull-Rom

Після визначення траєкторії дороги R наступним кроком є зміна карти висот для вирівнювання рельєфу уздовж дороги. Для кожної інтерпольованої точки R_k, ми виділяємо комірки місцевості в радіусі, що дорівнює половині ширини дороги $\frac{w}{2}$. Евклідова відстань від кожної клітини (x, y) до центру дороги (x_k, y_k) обчислюється як (2.18):

$$d(x, y) = \sqrt{(x - x_k)^2 + (y - y_k)^2}. \tag{2.18}$$

Якщо $d(x, y) \leq \frac{w}{2}$, тоді значення висоти $H(x, y)$ регулюється відповідно до висоти центру дороги $H(x_k, y_k)$.

Також не менш важливим аспектом створення доріг є змішування країв дороги з навколишньою місцевістю. Різкі переходи можуть призвести до візуально нереалістичних результатів і порушити безперервність ландшафту. Щоб вирішити цю проблему, до карти висот поблизу країв дороги застосовується фільтр згладжування Гауса. Функція Гауса визначається як (2.19):

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{d^2(x,y)}{2\sigma^2}}, \tag{2.19}$$

де σ – контролює ступінь згладжування. Згладжене значення висоти $H(x, y)$ обчислюється як середнє зважене сусідніх значень висоти (2.20):

$$H_s(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} H(x + i, y + j) G(i, j). \tag{2.20}$$

Операція згладжування забезпечує поступовий перехід від рівної поверхні дороги до природного рельєфу, підвищуючи її візуальну реалістичність.

Рис. 2.7. Результат генерації доріг та їх інтеграції з ландшафтом

Однією з сильних сторін алгоритму є його гнучкість. Регулюючи такі параметри, як ширина дороги w та коефіцієнт згладжування σ, користувачі можуть контролювати зовнішній вигляд доріг, роблячи їх придатними для різних типів ландшафтів і застосувань. Крім того, використання сплайнів дозволяє створювати складні дорожні мережі з кривими та перехрестями.

Висновки до розділу 2

Проведене дослідження демонструє різноманітні підходи до покращення генерації тривимірних цифрових ландшафтів, інтегруючи математичні підходи, фізичні методи та алгоритми взаємодії з користувачем. Використання методу нелінійного масштабування, що забезпечує уточнювання характеристик ландшафту на картах висот при його створенні, покращує тонкі особливості рельєфу, зокрема пологі долини, гори з гострими вершинами або хвилясті рівнини.

За допомогою розробленого модульного підходу реалізуються різноманітні режими накладання шарів, що відбувається за таким алгоритмом: базова функція ітераційно проходить через сітку висот, динамічно застосовує вибрані користувачем операції за допомогою інкапсульованих методів, а також інтегрує непрозорість за допомогою лінійної інтерполяції для масштабованого та ефективного накладання шарів. Ці методи надають користувачам інтуїтивно

зрозумілі, але потужні інструменти для маніпулювання даними рельєфу, забезпечуючи побудову навіть дуже складних рельєфів з мінімальними обчислювальними витратами.

Впровадження базованих на фізиці методів дозволяє значно підвищити реалістичність і автентичність створюваних ландшафтів. Включення геологічного моделювання, такого як вивітрювання та тектонічний рух, а також взаємодія води з ландшафтом дозволяє симулювати розвиток рельєфу з плином часу, імітуючи реальні природні процеси. Алгоритм гідравлічної ерозії на основі частинок виділяється в цьому відношенні, використовуючи принципи класичної механіки для точного моделювання транспортування та осадження. Відстежуючи рух частинок, перенесення маси та випаровування, алгоритм фіксує складну взаємодію між поверхнею та ерозійними силами.

Розроблений метод генерації та інтеграції дорожніх мереж у рельєф значно розширює функціональні можливості застосунку. Поєднуючи сплайн-інтерполяцію Catmull-Rom, вирівнювання рельєфу та згладжування країв, алгоритм створює реалістичні дороги, які плавно інтегруються із місцевістю. Запропонований метод є обчислювально ефективним і може бути легко інтегрований в існуючі системи генерації рельєфу.

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Використовувані технології розробки

Розробка застосунку для генерації тривимірного ландшафту вимагає ретельного відбору технологій розробки, кожна з яких вибирається для виконання певних аспектів функціональності.

Інтегроване середовище розробки (IDE): Visual Studio 2022

Visual Studio 2022, потужне інтегроване середовище розробки (IDE), яке забезпечує комплексне та гнучке середовище для створення та відлагодження програмного застосунку. Вибір Visual Studio пояснюється його надійним набором інструментів, ефективними можливостями відлагодження та бездоганною інтеграцією з іншими технологіями, необхідними для цього проєкту, такими як OpenGL, GLFW та ImGui. Крім того, Visual Studio пропонує широкий набір шаблонів проєктів і конфігурацій, які допомагають оптимізувати розробку на мові C++, забезпечуючи сумісність з API OpenGL та іншими бібліотеками.

Мова програмування: C++

Вибір C++ як мови програмування ґрунтується на його високій продуктивності, широкій підтримці об’єктно-орієнтованого програмування та можливостях точного керування пам’яттю. У застосунку для створення 3D рельєфу, де кожен кадр може включати мільйони вершин і тисячі складних математичних операцій, накладні витрати, пов’язані з динамічним розподілом пам’яті та збором сміття, повинні бути мінімізовані. Використання C++ також полегшує інтеграцію OpenGL, GLFW і GLM, оскільки усі вони оптимізовані для C++ і забезпечують вбудовану підтримку високопродуктивної графіки та математичних операцій.

Графічний API: OpenGL

OpenGL було обрано як графічний API для цього проєкту через його широке застосування, розвинену екосистему та крос-платформні можливості. Його сумісність із GLFW та ImGui забезпечує повну інтеграцію компонентів рендерингу, вікон та інтерфейсу користувача. Реалізація OpenGL у застосунку починається з ініціалізації контексту візуалізації, створеного за допомогою GLFW для керування вікном програми. Налаштування включає визначення вікна перегляду, конфігурацію кадрових буферів і підготовку шейдерів для обробки вершин. Шейдери займають центральне місце в процесі візуалізації, оскільки вони обробляють дані вершин і застосовують ефекти освітлення та затінення, які надають рельєфу глибину та реалістичність.

Підтримка OpenGL об’єктів буфера вершин (VBO) і об’єктів масиву вершин (VAO) забезпечує ефективне зберігання та обробку даних вершин. У розроблюваному застосунку результуюча карта висот перетворюється на сітку вершин, які, в свою чергу, зберігаються в VBO, тоді як VAO керують організацією атрибутів вершин, таких як положення, нормалі та текстурні координати. Далі конвеєр візуалізації обробляє ці дані для створення тривимірної моделі рельєфу, яка вже відображається у вікні програми.

Математична бібліотека: GLM (OpenGL Mathematics)

GLM – це математична бібліотека, спеціально розроблена для графічних додатків, яка забезпечує набір функцій і структур даних, адаптованих до вимог векторної математики. У контексті нашого застосунку GLM використовується для виконання обчислень над векторами, матрицями та іншими математичними об’єктами, які визначають форму, орієнтацію та трансформацію моделей рельєфу.

Матричні та векторні класи GLM використовуються для обробки операцій перетворення та проєктування, необхідних у конвеєрі візуалізації OpenGL. Наприклад, при створенні 3D-рельєфу кожна точка на сітці карти висот відповідає вершині в 3D-просторі. Використовуючи векторні операції

GLM, цими вершинами можна легко маніпулювати для створення схилів, піків і долин, які імітують рельєф реального світу. Крім того, GLM має важливе значення для розрахунку векторів нормалей, освітлення та затінення.

Бібліотека вікон: GLFW

GLFW – це міжплатформна бібліотека, яка використовується для керування вікнами, введенням і подіями. GLFW відповідає за створення та керування головним вікном застосунку в Windows, а також за обробку введення користувача за допомогою клавіатури та миші. Вибір GLFW мотивований його простотою та сумісністю з OpenGL, що робить його ідеальним для застосунку, орієнтованого на продуктивність.

Ініціалізація GLFW передбачає створення вікна з OpenGL-сумісністю та налаштування зворотних викликів подій для обробки взаємодії користувача. Наприклад, в інтерфейсі програми користувачі можуть генерувати або завантажувати карти висот, регулювати режими змішування та переглядати рельєф. Коли користувач вносить коригування, GLFW обробляє подію, передає її в конвеєр OpenGL і забезпечує відтворення оновленої моделі рельєфу у вікні. Цей керований подіями підхід оптимізує інтерфейс користувача та забезпечує оперативну взаємодію, оскільки вхідні дані обробляються паралельно із завданнями візуалізації.

Бібліотека інтерфейсу користувача: ImGui

ImGui – бібліотека, яка надає графічний інтерфейс користувача негайного режиму, що дозволяє створювати динамічні та інтерактивні інтерфейси. Інтеграція ImGui з OpenGL дуже корисна в цьому проєкті, оскільки дозволяє користувачам взаємодіяти з елементами керування генерацією рельєфу в режимі реального часу, не перериваючи процес візуалізації.

ImGui ініціалізується шляхом створення контексту та зв’язування його з системою візуалізації OpenGL. Кожен кадр ImGui малює елементи інтерфейсу безпосередньо в кадровому буфері, яким керує OpenGL, накладаючи елементи

керування на відрендерену місцевість. Обробка подій ImGui у поєднанні з GLFW дозволяє швидко реагувати на введення даних, дозволяючи користувачам коригувати параметри рельєфу в реальному часі. Наприклад, користувачі можуть регулювати непрозорість окремих шарів карти висот, а ImGui миттєво оновлює інтерфейс відповідно до цих змін.

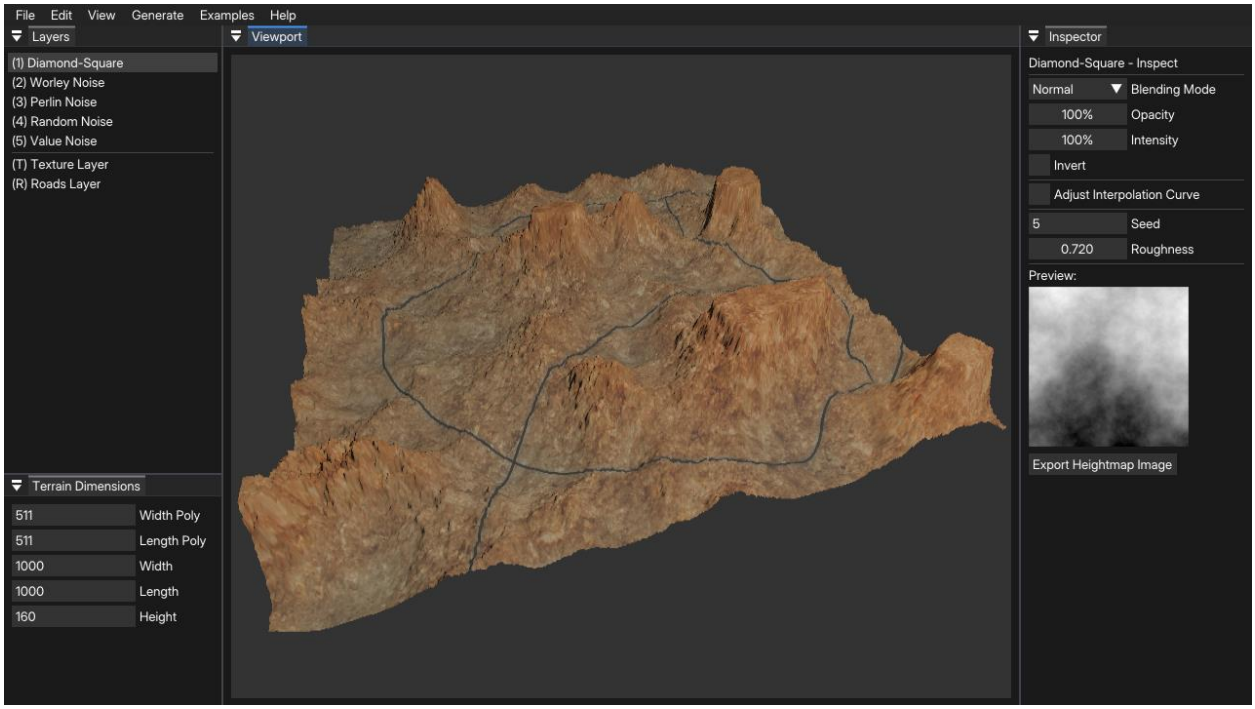


Рис. 3.1. Скріншот інтерфейсу застосунку

3.2 Проєктування програмного застосунку

Структура зберігання даних про ландшафт

Формування рельєфу починається з внутрішньої організації даних ландшафту. Кожен шар рельєфу має бути параметризований, щоб дозволити ефективне збереження та налаштування відповідних параметрів. Щоб досягти цього, програма використовує двовимірний цілочисельний масив під назвою «parametersArray», де кожен рядок відповідає різному шару карти висот, а кожен стовпець містить окремий параметр, пов’язаний із цим шаром. Ця конструкція підтримує легкодоступну та масштабовану систему зберігання, що дозволяє виконувати складні маніпуляції та перетворення кожного рівня (таблиця 3.1).

Таблиця 3.1. Структура масиву «parametersArray»

	Тип шару	***Режим змішування***	***Непрозорість***	***Інтенсивність***	***Інвертованість***	***Зерно***	***Коефіцієнт коригування***	***…***
Шар 1	3	0	100	100	0	61728	0	…
Шар 2	5	2	100	100	0	718	1	…
Шар 3	1	2	100	100	0	521	0	…
Шар 4	2	0	50	100	0	12345	0	…
Шар 5	0	0	50	100	0	0	0	…
…	…	…	…	…	…	…	…	…

Двовимірна структура «parametersArray» покращує модульність, дозволяючи незалежно модифікувати, змінювати порядок або видаляти кожен шар, не впливаючи на цілісність даних рельєфу в цілому. Кожним із цих шарів можна керувати незалежно, дозволяючи користувачеві створювати рельєфи з виразними геологічними особливостями, регулюючи такі параметри шару, як тип, змішування, непрозорість тощо. Вплив цих шарів, об’єднаних за допомогою визначеного порядку змішування, сприяє остаточному профілю висоти рельєфу. Крім того, формат зберігання полегшує операції пошуку, оскільки кожен параметр зіставляється з певним індексом стовпця, що дозволяє швидко здійснювати коригування під час маніпуляцій у реальному часі.

Реалізація функціональності «Custom Heightmap»

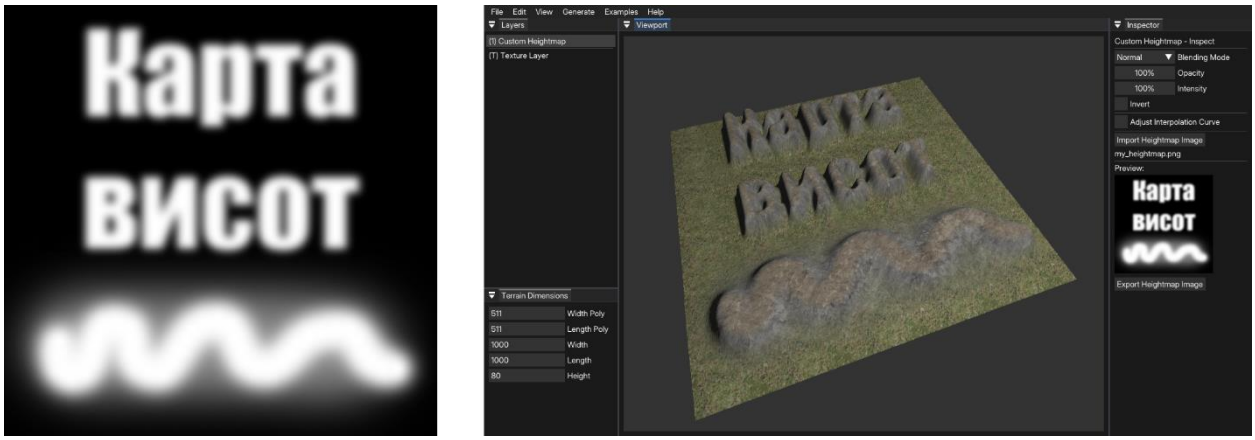
Щоб ще більше підвищити гнучкість, застосунок містить функцію завантаження власних карт висот із зовнішніх файлів зображень. Ця можливість дозволяє користувачам використовувати будь-яке зображення як карту висот, відкриваючи можливості для створення рельєфу на основі

реальних даних або унікального художнього введення. Ця функція використовує «stbi_image», бібліотеку для завантаження зображень у різних форматах.

Процес завантаження починається зі зчитування зображення з шляху до файлу, наданого користувачем. Якщо зображення містить значення RGB, воно перетворюється на відтінки сірого, щоб відповідати вимогам до структури карт висот, використовуючи метод яскравості (3.1):

$$gray = 0{,}2126r + 0{,}7152g + 0{,}0722b. \quad (3.1)$$

Після трансформації роздільна здатність зображення масштабується відповідно до визначених користувачем розмірів рельєфу за допомогою білінійної інтерполяції, яка гарантує, що будь-яке зображення, незалежно від його початкового розміру, може служити як карта висот, зберігаючи усі його деталі. Після успішного завантаження та масштабування зображення воно перетворюється на дані карти висот шляхом зіставлення інтенсивності кожного пікселя зі значенням висоти. Кожен піксель у відтінках сірого відповідає значенню висоти від 0 до 1, яке додається в стек шарів ландшафту.



a *b*

Рис. 3.2. Приклад застосування функції «Custom Heightmap»:
a – заздалегідь створена карта висот;
b – згенерований на її основі ландшафт.

Цей процес, хоч і простий, але додає суттєвої універсальності застосунку, дозволяючи використовувати як процедурні, так і створені користувачем дані для побудови ландшафту.

Збереження проєктів

Щоб керувати та зберігати складні конфігурації рельєфу, в застосунку реалізований спеціальний формат файлу .tgf (Terrain Generation File). Формат файлу .tgf дозволяє користувачам зберігати всі відповідні дані місцевості, включаючи параметри карти висот, режими змішування та спеціальні налаштування, у текстовий файл, який можна перезавантажувати в будь-який час. Цей формат файлу дозволяє користувачам продовжувати працювати над місцевістю, не втрачаючи своїх конфігурацій, полегшуючи спільне та ітераційне проєктування.

Сам процес збереження (завантаження) файлу починається із вибору користувачем з меню «File» кнопки «Save As» («Open») або використовуючи комбінацію клавіш «Ctrl + S» («Ctrl + O»), що відкриває діалогове вікно вибору шляху куди буде збережений (зчитаний) файл (рис. 3.3).

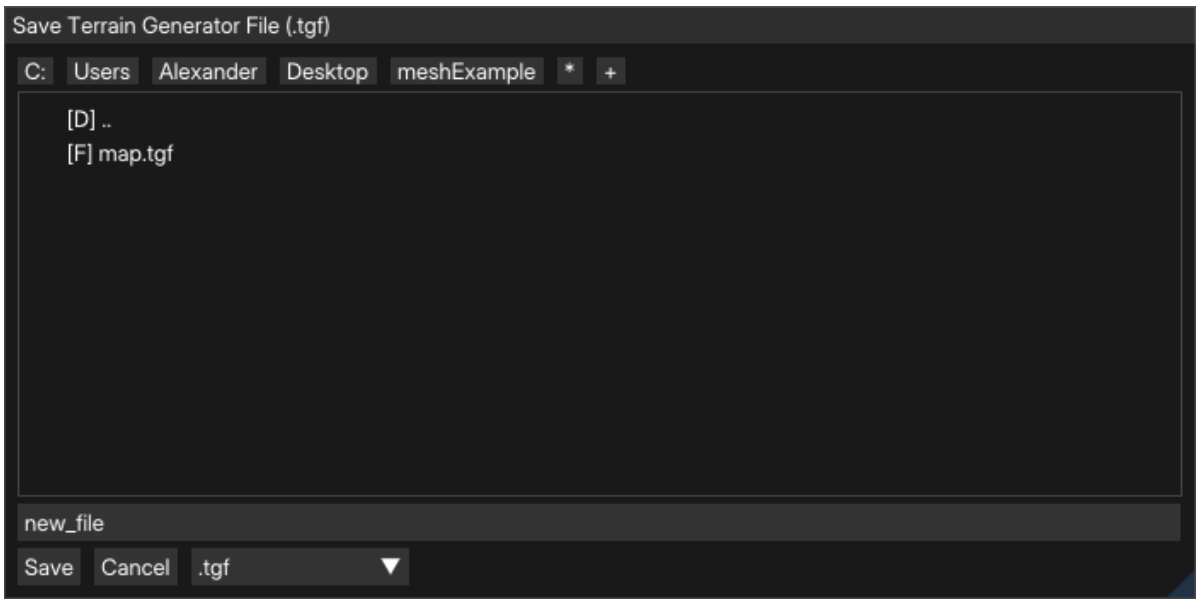


Рис. 3.3. Діалогове вікно збереження файлу

Залежно від вибраної операції, що має бути виконана (збереження чи завантаження) запускається відповідна функція («SaveFileByPath(filepath)» чи «OpenFileByPath(filepath)») в яку подається параметр «filepath» – вказаний користувачем шлях.

Під час збереження проєкту кожен рядок у «parametersArray» та «directoryArray» (масив усіх заданих користувачем шляхів до карт висот,

текстур тощо) серіалізуються з параметрами, розділеними комами, утворюючи один рядок у текстовому файлі. Цей підхід зрозумілий людині та легко аналізується програмою після завантаження, що дозволяє швидко відновити всі налаштування рельєфу.

Для завантаження файл .tgf читається рядок за рядком, і кожен параметр подається назад у масиви «parametersArray» та «directoryArray». Цей метод текстової серіалізації покращує сумісність і прозорість, оскільки файли .tgf можна відкривати та перевіряти в будь-якому текстовому редакторі. Формат .tgf дозволяє зберігати великі проєкти місцевості з мінімальним розміром файлу, що робить його ефективним для зберігання та швидким для завантаження.

Експорт 3D-моделей та текстур

Після створення ландшафту в застосунку реалізована можливість експортувати 3D-модель у форматі .obj – широко використовуваному стандарті збереження 3D-моделей. Формат .obj був обраний в якості основного формату експорту завдяки своїй простоті використання та підтримці збереження усіх необхідних даних про ландшафт: положення вершин, текстурні координати та нормалі, які є необхідними для детального відтворення рельєфу в іншому застосунку, в який експортується модель.

Розглянемо структуру формату .obj більш детально на прикладі простої площини, яка складається із чотирьох вершин (рис. 3.4).

```
v -1.000000 0.000000 -1.000000
v 1.000000 0.000000 -1.000000
v -1.000000 0.000000 1.000000
v 1.000000 0.000000 1.000000
vt 0.000000 0.000000
vt 1.000000 0.000000
vt 0.000000 1.000000
vt 1.000000 1.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
f 4/4/4 2/2/2 1/1/1 3/3/3
```

Рис. 3.4. Структура формату .obj

Дані про 3D-модель зберігаються в текстовому форматі окремими блоками. Перший блок містить координати вершин моделі в тривимірному просторі, які записуються послідовно з нового рядка із позначкою «v». В контексті моделі ландшафту, вершини визначають 3D-координати кожної точки місцевості, представляючи дані висоти, відповідно до карт висот.

Другий блок зберігає текстурні координати в двовимірному (UV) просторі, які записуються у файл аналогічно до вершин, але із позначкою «vt». Текстурні координати відповідають за правильне відображення текстур на 3D-моделі шляхом перерозподілу областей вхідного зображення відповідно до зазначених координат.

Третій блок складається із переліку одиничних векторів нормалей, які йдуть рядок за рядком з позначкою «vn». Вектори нормалей необхідні для імітації реалістичних ефектів освітлення під час візуалізації в 3D-середовищі, оскільки вони позначають орієнтацію поверхні в кожній вершині і визначають як світло повинно відбиватися від неї під різними кутами.

Четвертий блок містить перелік граней (чотирикутних поверхонь) із позначкою «f», які вказують на те, як рендерер повинен візуалізовувати подані вище дані. Грані зберігаються як послідовність сукупностей індексів вершин у форматі «v/vt/vn». Важливо також враховувати порядок у якому здійснюється перелік, оскільки він вказує на орієнтацію нормалі грані $\vec{n}$ (рис. 3.5).

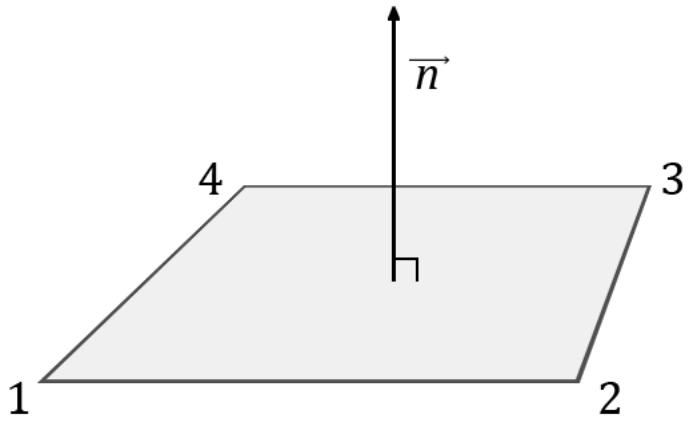
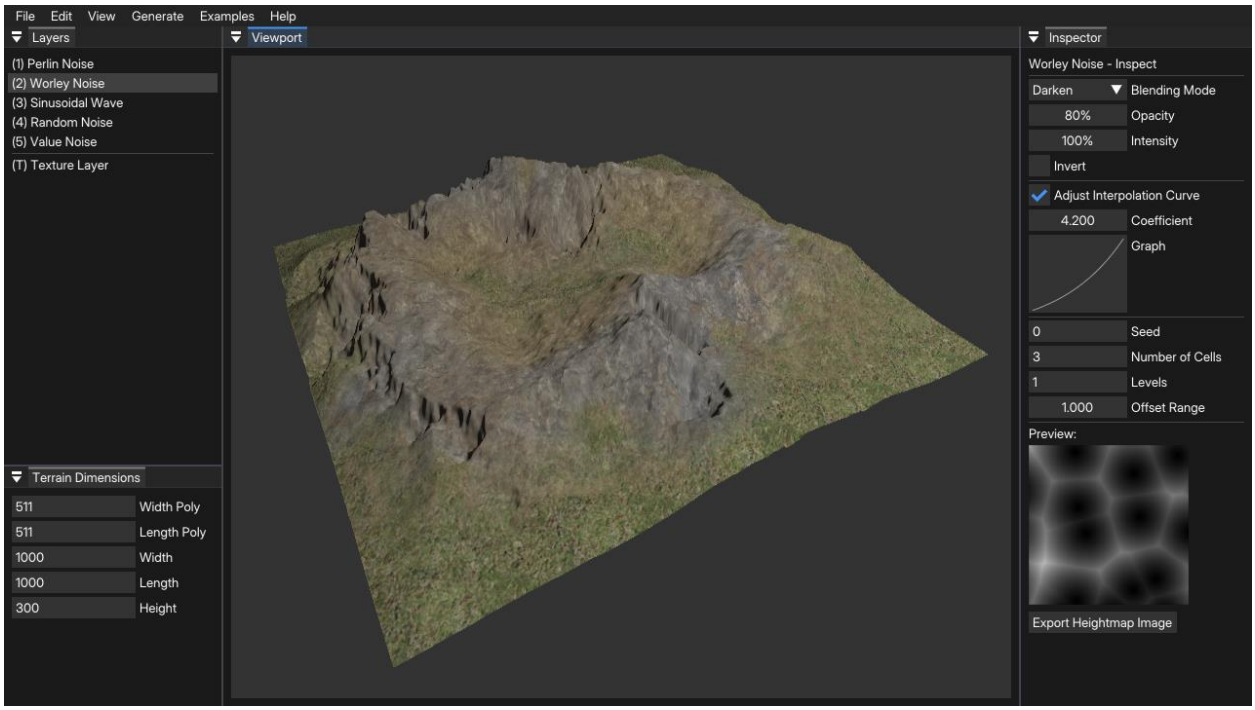
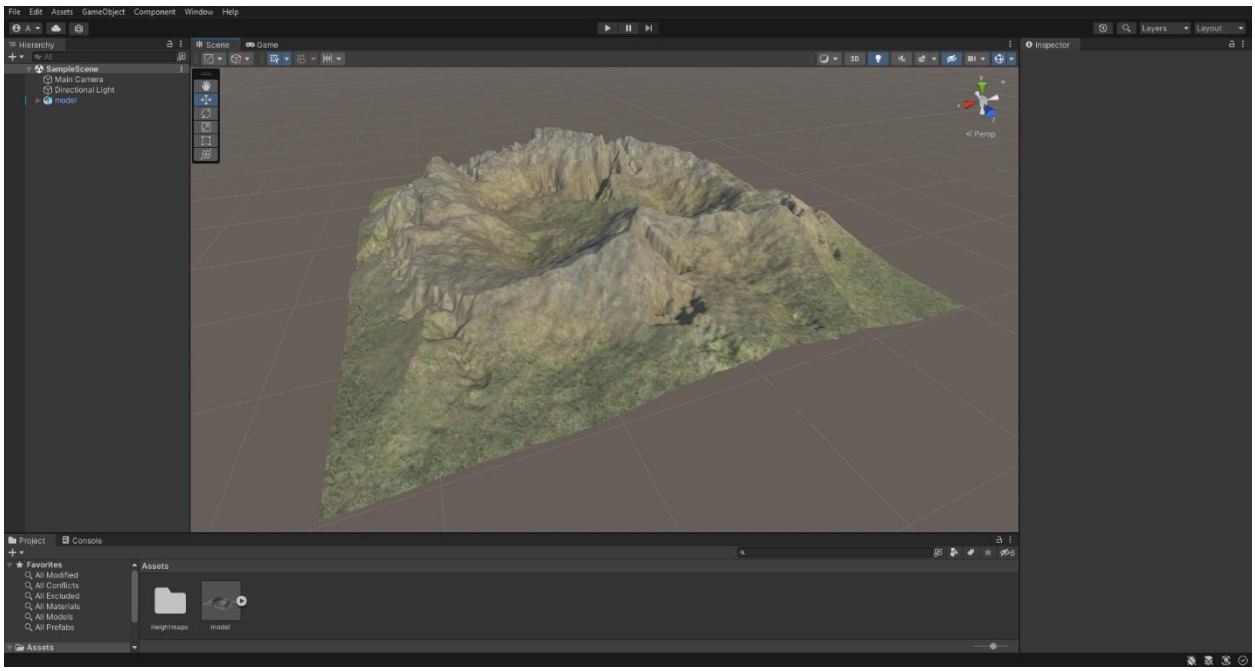


Рис. 3.5. Орієнтація нормалі грані, якщо перерахунок робити проти годинникової стрілки

Оскільки текстури генеруються процедурно на основі зчитаних зображень та параметрів заданих користувачем, потрібно також передбачати і їх експорт. Для цього в застосунку передбачено автоматичний експорт згенерованої текстурної карти разом із моделлю у форматі .bmp.



a



b

Рис. 3.6. Приклад експорту згенерованої моделі та її подальший імпорт в інший застосунок:
a – створена в застосунку модель ландшафту;
b – імпортована модель в ігровий рушій Unity.

3.3 Реалізація цифрового представлення елементів ландшафту

Процес побудови тривимірної моделі ландшафту починається з ініціалізації координат вершин, які будуть представляти топологічну структуру місцевості. Кожній вершині сітки ставиться у відповідність певна точка в 3D-просторі, яка обчислюється на основі параметрів, визначених користувачем, такими як ширина «Width», довжина «Length», висота «Height» та кількість полігонів вздовж кожної із осей горизонтальної площини («Width Poly» та «Length Poly» для ширини і довжини відповідно). Дані параметри є базовими для кожного ландшафту, а тому їх задання передбачається в окремому вікні (рис. 3.7).

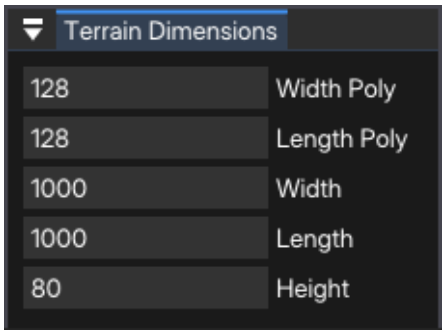


Рис. 3.7. Вікно «Terrain Dimensions» із заданими за замовчуванням параметрами генерації

Координати x і z для кожної вершини визначаються на основі заданої ширини w і довжини l місцевості, поділеної на кількість чотирикутних полігонів, визначених уздовж кожної осі. Якщо n_x та n_z представляють кількість полігонів у напрямках x та z відповідно, тоді кількість вершин буде рівна $n_x + 1$ та $n_z + 1$, за рахунок включення крайніх вершин. Координата y, в свою чергу, задається за рахунок множення висоти місцевості h на результуючу карту висот розмірністю $n_x + 1$ на $n_z + 1$ під назвою «resultHeightmap» (математично будемо позначати як $H_r(x, z)$), що складається із нормалізованих значень висоти, одержаних шляхом поєднання попередньо згенерованих карт висот із зазначеними користувачем параметрами накладання, непрозорості,

інтенсивності тощо. Таким чином, координати вершини v_{ij} в і-му рядку та j-му стовпці визначаються як (3.2):

$$\begin{cases} x_{ij} = w \cdot \left(\dfrac{i}{n_x} - 0{,}5\right), \\ y_{ij} = h \cdot H_r(i,j), \\ z_{ij} = l \cdot \left(\dfrac{j}{n_z} - 0{,}5\right), \end{cases} \tag{3.2}$$

де $i = \overline{0, n_x}$ та $j = \overline{0, n_z}$.

Щоб забезпечити симетричний центр рельєфу в 3D-просторі, початкова точка фіксується на $(0; 0; 0)$, створюючи збалансовану структуру, навколо якої розподіляються всі вершини (рис. 3.8).

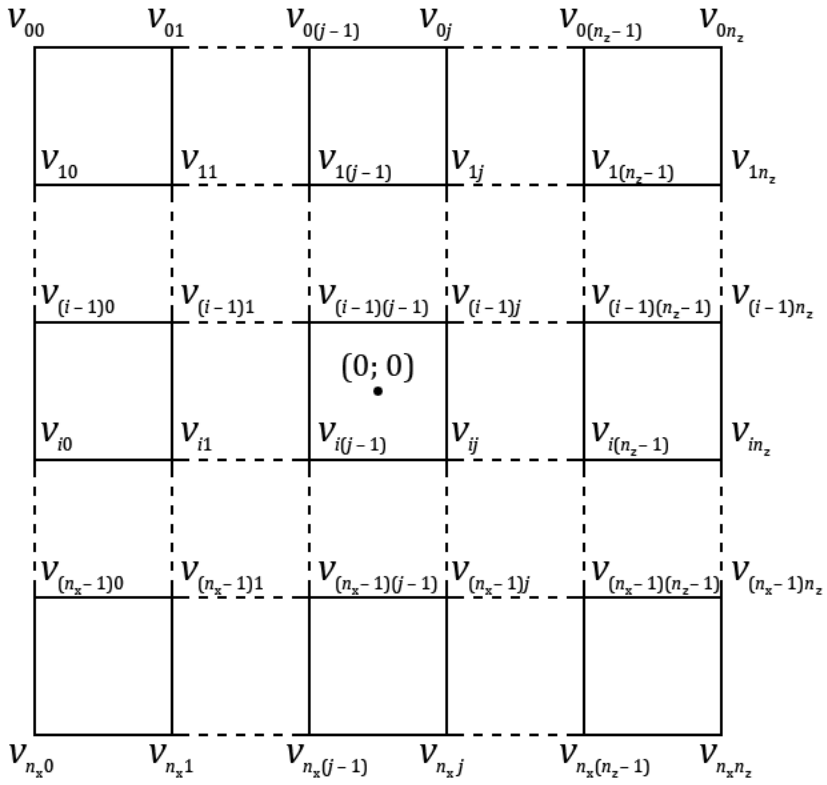


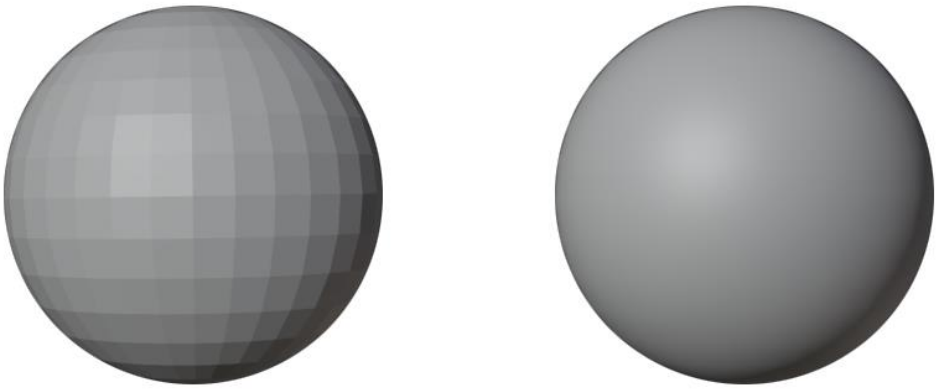
Рис. 3.8. Структура полігональної сітки, утвореної шляхом розподілу вершин на площині

Наступним кроком, після побудови усіх вершин, необхідно обчислити текстурні координати для кожної з них. Усі UV-координати (u, v) призначаються на основі ширини та довжини сітки в діапазоні від $(0; 0)$ у верхньому лівому куті до $(1; 1)$ у нижньому правому куті. Отже, для вершини v_{ij} UV-координати визначаються як (3.3):

$$\begin{cases} u_{ij} = \dfrac{i}{n_x}, \\ v_{ij} = \dfrac{j}{n_z}. \end{cases} \tag{3.3}$$

Такий підхід є найоптимальнішим, оскільки він не передбачає зчитування вже існуючих координат вершин та їх трансформування. Це є можливим завдяки тому, що значення карти висот задають лише розміщення вершин по вертикалі, не впливаючи на їх горизонтальне положення. Іншими словами, проекція моделі ландшафту на горизонтальну площину завжди дає заздалегідь відому прямокутну сітку.

Для досягнення правильного рендерингу генерованої моделі необхідно виконувати обчислення одиничних векторів нормалей для кожної з вершин. Загалом можна виділити два основних підходи до побудови нормалей – «Flat shading» та «Smooth shading» (рис. 3.9).



a *b*

Рис. 3.9. Модель сфери з різними підходами до згладжування:
a – метод «Flat shading»;
b – метод «Smooth shading».

Метод «Flat shading» прирівнює вектори нормалей до нормалі суміжного полігону, чітко виділяючи при цьому ребра 3D-моделі. Метод «Smooth shading», в свою чергу, усереднює напрямки нормалей усіх суміжних для вершини граней, створюючи при цьому плавний градієнт без необхідності у використанні великої кількості полігонів, що дозволяє ефективно імітувати плавну форму рельєфу [25].

Процес обчислювання векторів нормалей в програмному застосунку здійснюється у декілька етапів. Перший етап полягає в обчисленні нормалей для кожної грані моделі, необхідних для подальших обчислень, які будуть зберігатися в тимчасовому масиві. Слід зазначати, що через довільне зміщення вершин по вертикалі, не завжди чотири вершини які утворюють грань будуть лежати на одній площині, а тому постає необхідність у попередній тріангуляції та усередненні нормалей двох трикутників, що формують чотирикутний полігон. Важливо також відмітити, що вершини, які знаходяться на краю сітки мають менше чотирьох суміжних полігонів, а тому для спрощення обчислень їх значення можна продублювати, одержавши цим самим тимчасовий масив розмірністю $n_x + 3$ на $n_z + 3$. Крім цього, даний масив можна значно спростити залишивши лише значення висоти вершин, оскільки із приведеної раніше структури ландшафтної сітки випливає, що довжини ребер комірок сітки рівні між собою і їх можна обчислити заздалегідь (рис. 3.10).

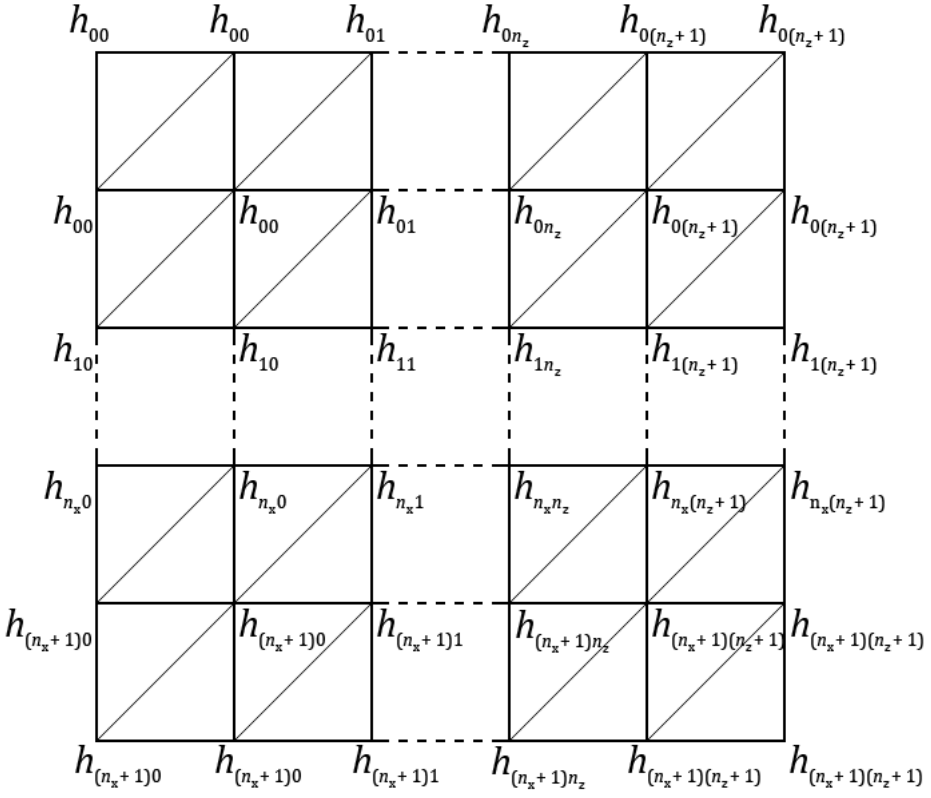


Рис. 3.10. Тимчасовий масив із продубльованими значеннями висот

Далі створюється новий тимчасовий масив тривимірних одиничних векторів, який вже буде безпосередньо зберігати нормалі для кожної грані

розширеного масиву, тобто його розмірність повинна бути $n_x + 2$ на $n_z + 2$. Розглянемо одну із клітинок (рис. 3.11).

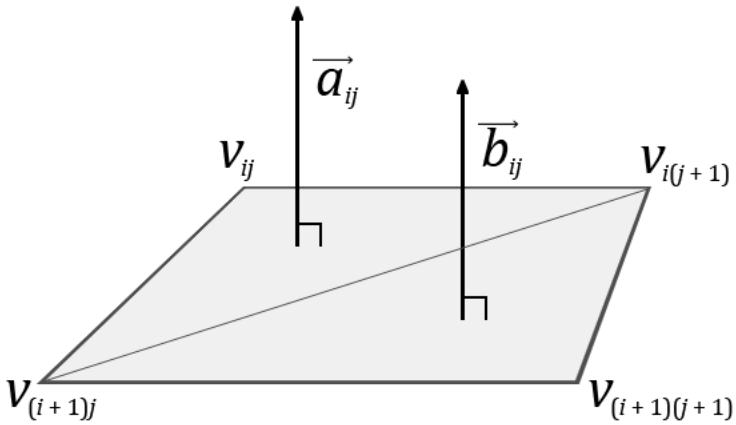


Рис. 3.11. Клітинка сітки, яка являє собою тріангульований чотирикутний полігон

Маємо чотири вершини v_{ij}, $v_{(i+1)j}$, $v_{(i+1)(j+1)}$ і $v_{i(j+1)}$, які формують представлений на рисунку полігон. Наша задача полягає у знаходженні нормалі грані $\overrightarrow{n_{ij}}$, яка, в свою чергу, одержується із нормалей двох складових трикутників $\overrightarrow{a_{ij}}$ та $\overrightarrow{b_{ij}}$ наступним чином (3.4):

$$\overrightarrow{n_{ij}} = \frac{\overrightarrow{a_{ij}} + \overrightarrow{b_{ij}}}{\left|\overrightarrow{a_{ij}} + \overrightarrow{b_{ij}}\right|}. \tag{3.4}$$

Щоб знайти значення $\overrightarrow{a_{ij}}$ і $\overrightarrow{b_{ij}}$ необхідно на катетах кожного із трикутників побудувати вектори. Для першого трикутника $\Delta v_{ij} v_{(i+1)j} v_{i(j+1)}$ ми будуємо два вектори $\overrightarrow{v_{ij} v_{(i+1)j}}$ та $\overrightarrow{v_{ij} v_{i(j+1)}}$, векторний добуток яких, дає шуканий вектор $\overrightarrow{a_{ij}}$. Аналогічно знаходиться і вектор $\overrightarrow{b_{ij}}$ шляхом векторного добутку $\overrightarrow{v_{(i+1)(j+1)} v_{i(j+1)}}$ та $\overrightarrow{v_{(i+1)(j+1)} v_{(i+1)j}}$. В результаті, із урахуванням порядку, в якому необхідно здійснювати множення, одержуємо (3.5):

$$\begin{cases} \overrightarrow{a_{ij}} = \left[\overrightarrow{v_{ij} v_{(i+1)j}}, \overrightarrow{v_{ij} v_{i(j+1)}}\right], \\ \overrightarrow{b_{ij}} = \left[\overrightarrow{v_{(i+1)(j+1)} v_{i(j+1)}}, \overrightarrow{v_{(i+1)(j+1)} v_{(i+1)j}}\right]. \end{cases} \tag{3.5}$$

В наступному етапі, після знаходження усіх нормалей граней, кожній вершині присвоюється усереднене нормалізоване значення чотирьох суміжних граней, які являють собою шукані вектори нормалі.

Таким чином, ми отримали загальний підхід до знаходження векторів нормалей, але його можна значно оптимізувати використовуючи особливості моделі ландшафту та побудовані в попередніх кроках масиви даних.

Оптимізацію будемо здійснювати шляхом спрощення алгоритму побудови векторів на ребрах полігону. В загальному випадку дані вектори знаходяться шляхом віднімання від координат кінцевої вершини координати початкової. Із побудованої структури моделі, представленої на рис. 3.8, слідує, наступне (3.6), (3.7):

$$\begin{cases} x_{0j} = x_{1j} = \cdots = x_{n_x j}, & j = \overline{0, n_z} \\ z_{i0} = z_{i1} = \cdots = x_{i n_z}, & i = \overline{0, n_x} \end{cases} \tag{3.6}$$

$$\begin{cases} x_{i(j+1)} - x_{ij} = \dfrac{w}{n_x}, & i = \overline{0, n_x}, j = \overline{0, n_z - 1} \\ z_{(i+1)j} - z_{ij} = \dfrac{l}{n_z}, & i = \overline{0, n_x - 1}, j = \overline{0, n_z} \end{cases} \tag{3.7}$$

Значення висот будемо брати із тимчасового масиву представленого на рис. 3.10. Підставивши усі одержані значення виразів у формулу знаходження векторів одержуємо наступну систему (3.8):

$$\begin{cases} \overrightarrow{v_{ij} v_{(i+1)j}} = \left(0, h_{(i+1)j} - h_{ij}, \dfrac{l}{n_z}\right), \\ \overrightarrow{v_{ij} v_{i(j+1)}} = \left(\dfrac{w}{n_x}, h_{i(j+1)} - h_{ij}, 0\right), \\ \overrightarrow{v_{(i+1)(j+1)} v_{i(j+1)}} = \left(0, h_{i(j+1)} - h_{(i+1)(j+1)}, -\dfrac{l}{n_z}\right), \\ \overrightarrow{v_{(i+1)(j+1)} v_{(i+1)j}} = \left(-\dfrac{w}{n_x}, h_{(i+1)j} - h_{(i+1)(j+1)}, 0\right). \end{cases} \tag{3.8}$$

Отже, оптимізований алгоритм знаходження векторів нормалей зводиться лише до попереднього обчислення значень ширини $\frac{w}{n_x}$ та довжини $\frac{l}{n_z}$ клітинок сітки, підстановки їх значень та значень із тимчасового масиву висот у (3.5) для знаходження $\overrightarrow{a_{ij}}$ і $\overrightarrow{b_{ij}}$, підставлення одержаних векторів у (3.4) для знаходження $\overrightarrow{n_{ij}}$ та присвоювання їх усереднених нормалізованих значень кожній із вершин.

Останнім кроком у побудові моделі ландшафту є подання індексів вершин, які формують полігон. Цей процес виконується аналогічним до експорту 3D-моделі чином, але він має певну відмінність. Для правильного рендерингу моделі її потрібно тріангулювати та подавати не індекси чотирикутного полігону, а трикутного. Щоб спростити цей процес не відходячи від спроєктованої структури сітки можна згрупувати індекси двох трикутних полігонів, що утворюють чотирикутний в один блок циклу, тобто для полігону представленому на рис. 3.5, замість подання 1-2-3-4, подавати 1-3-4, 1-2-3. Фрагмент коду відповідального за побудову моделі ландшафту подано у Додатку Б.

3.4 Впровадження інтерактивних елементів у програмі

Інтерфейс застосунку поділений на кілька логічних розділів, включаючи керування базовими параметрами генерації ландшафту, параметризацію рельєфу, змішування шарів і 3D-візуалізацію (рис. 3.12).

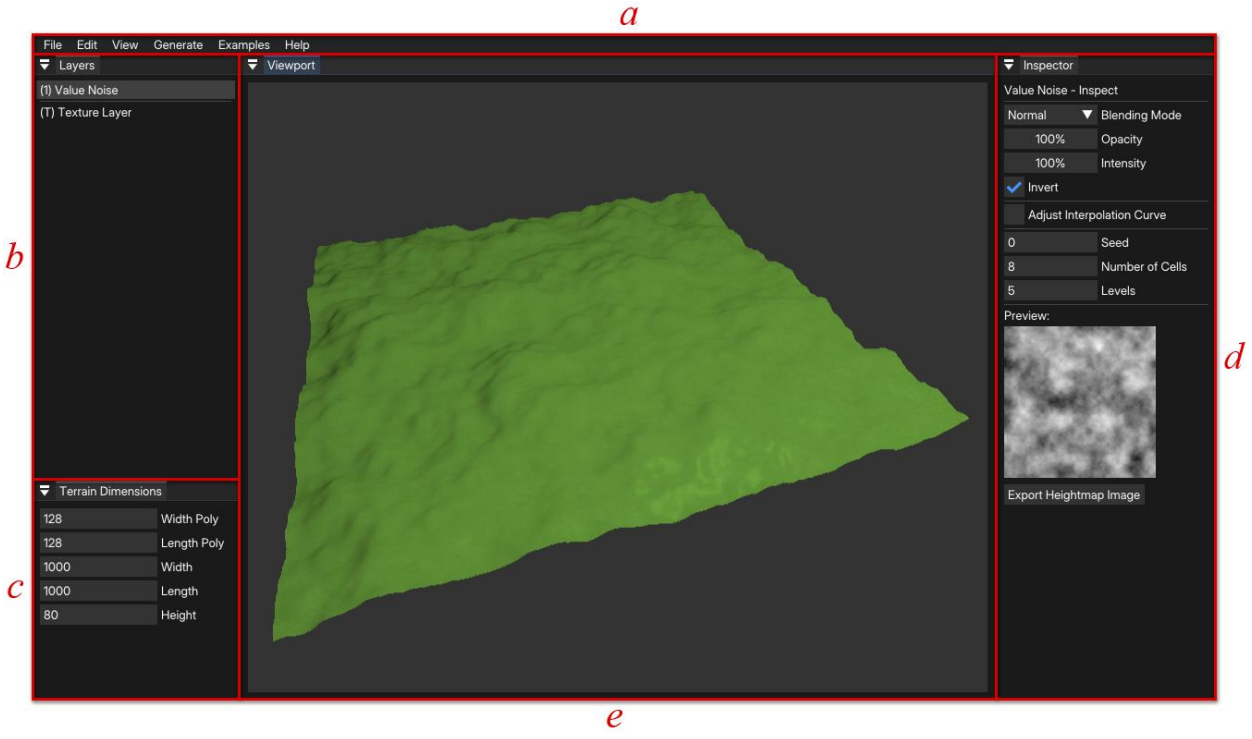


Рис. 3.12. Інтерфейс програмного застосунку:
a – панель меню застосунку;
b – вікно перегляду та вибору застосованих шарів;

c – вікно для задання базових параметрів генерації ландшафту;
d – вікно детального налаштування параметрів для вибраного шару із списку застосованих;
e – вікно попереднього перегляду одержаних результатів.

Структура графічного інтерфейсу розроблена з акцентом на інтуїтивно зрозумілу навігацію користувача. Основне вікно відображає тривимірну візуалізацію рельєфу, тоді як допоміжні панелі забезпечують елементи керування для зміни параметрів рельєфу, керування шарами карти висот і налаштування параметрів генерації тощо. Ці панелі відображаються як плаваючі вікна, які користувачі можуть переміщувати та змінювати їх розмір відповідно до своїх уподобань.

Панель меню слугує центральною точкою доступу до основних функцій програми та розташована у верхній частині інтерфейсу. Вона містить кілька спадних меню, кожне з яких надає певний набір параметрів:

- «File» – це меню містить параметри для відкриття, збереження та експорту файлів місцевості. Користувачі можуть завантажувати існуючі проєкти або експортувати створений рельєф як 3D-модель у таких форматах, як .obj.
- «Edit» – надає базові функції редагування, а саме операції скасування та повторення дій. Даний функціонал надає користувачам можливість керувати змінами без втрати прогресу.
- «View» – пропонує параметри для налаштування макета інтерфейсу, перемикання видимості певних вікон і налаштування параметрів вікна перегляду.
- «Generate» – це меню, що є центральним у процесі створення ландшафту, дозволяє користувачам створювати нові шари рельєфу та застосовувати шумові функції.
- «Examples» – надає попередньо налаштовані приклади різних типів місцевості, що дозволяє користувачам швидко зрозуміти та вивчити можливості програми.

- «Help» – містить документацію, навчальні посібники та інформацію про програму, яка допомагає користувачам орієнтуватися в її функціях.

Вікно шарів містить список усіх шарів карти висот, застосованих до рельєфу. Кожен шар представляє процедурний алгоритм, математичну функцію, користувацьку карту висот тощо, яка робить свій внесок у остаточну структуру генерованого ландшафту. Користувачі можуть вибирати, змінювати порядок, дублювати або видаляти шари, забезпечуючи гнучку та динамічну композицію рельєфу. Вибір шару зі списку активує вікно інспектора, де можна налаштувати детальні параметри цього шару. Дане вікно є контекстно-залежним і відображає настроювані параметри для поточного вибраного шару з вікна шарів. Воно забезпечує гнучкий контроль над їх властивостями, дозволяючи користувачам детально налаштовувати вплив кожного з них.

Наприклад, якщо при застосуванні двох шарів – шуму Перліна та шуму Ворлі – у вікні шарів вони відображатимуться як окремі записи. Користувачі можуть регулювати їх порядок, щоб контролювати, як вони змішуються (процес накладання шарів виконується у порядку їх нумерації – зверху вниз). Якщо вибраний шар є шумом Перліна, у вікні інспектора відображатимуться відповідні для нього параметри, а також попередній перегляд карти висот цього шару із можливістю її експорту як зображення у форматі .bmp. Користувачі можуть змінювати ці значення, щоб вплинути на структуру шуму та одразу побачити ефект у вікні перегляду.

Вікно перегляду – це центральне вікно, де можна переглядати створений рельєф у реальному часі. Воно забезпечує інтерактивний 3D-перегляд, дозволяючи користувачам обертати камеру, збільшувати та зменшувати масштаб. Ця взаємодія дає можливість швидко оцінювати зовнішній вигляд місцевості та вносити обґрунтовані коригування її параметрів. Управління камерою здійснюється за допомогою миші та клавіатури, а GLFW фіксує ці події та відповідно оновлює положення та орієнтацію камери.

Здатність змінювати параметри рельєфу в режимі реального часу є однією із найголовніших функцій програми, яка дозволяє користувачам миттєво візуалізувати результати своїх коригувань. Це досягається шляхом використання ефективних методів обробки даних і візуалізації, щоб забезпечити безперебійну роботу навіть під час роботи з великими наборами даних. Коли користувач взаємодіє з елементом GUI, таким як повзунок або спадне меню, програма оновлює базову структуру даних рельєфу та повторно відображає модель рельєфу. Наприклад, налаштування режиму змішування певного шару вимагає перерахунку комбінованої карти висот і оновлення даних вершин, які використовуються для візуалізації.

Висновки до розділу 3

Спроєктовано та розроблено програмний застосунок для створення 3D-моделей рельєфу. Він працює на основі процедурних методів і сучасних обчислювальних інструментів, придатних для застосовування при вирішенні завдань цифрового представлення ландшафту. Програмний застосунок розроблено із застосуванням ретельно підібраного стеку технологій, включаючи мову C++ для критичних до продуктивності алгоритмів, OpenGL для рендерингу графіки, GLFW для ефективного управління вікнами та введенням, GLM для векторних математичних операцій та ImGui для гнучкого користувацького інтерфейсу.

Поєднуючи у розробленому застосунку ефективні структури даних, інструменти налагодження гнучкої взаємодії з користувачем у режимі реального часу та високопродуктивні методи візуалізації, вдалося розробити програму, яка є чудовим прикладом практичної інтеграції теоретичних концепцій в реальних програмних застосунках. Методи та технології, що використовуються в цьому проєкті, застосовні не лише для моделювання рельєфу, але також можуть бути узагальнені для інших областей, що вимагають генерації та обробки складних просторових даних.

Модульна структура зберігання даних пропонує гнучкий підхід до управління та процесу змішування кількох процедурних шарів, а також, подібним чином, реалізація підтримки користувацьких карт висот і настроюваних параметрів забезпечує зручний механізм для включення зовнішніх даних у процедурні робочі процеси.

Здатність програми експортувати моделі в такий широко підтримуваний формат, як (.obj), підкреслює її потенціал для інтеграції у великі цифрові конвеєри. Ця функціональність гарантує, що процедурно згенеровані ландшафти можна легко впровадити в різноманітні робочі процеси, включаючи візуалізацію, моделювання та середовища віртуальної реальності. Крім того, зберігання проєктів у розробленому форматі файлу (.tgf) демонструє цінність адаптованих підходів до серіалізації даних для ефективного керування складними процедурними даними.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було досягнуто поставленої мети та виконані усі завдання. Зокрема, перше завдання полягало у дослідженні існуючих моделей, алгоритмів і методів для створення як природних особливостей місцевості, так і штучних структур, таких як мережі доріг. Було ретельно проаналізовано численні методики, включно з підходами, заснованими на фізиці, і математичними моделями, які покращують реалістичність синтезованого рельєфу. Досліджуючи гідродинамічні моделі, процеси ерозії та методи інтеграції дорожніх мереж, було визначено ключові стратегії підвищення як візуальної точності, так і обчислювальної ефективності цифрових ландшафтів. Ці фундаментальні знання сприяли подальшій розробці інноваційних методів і алгоритмів з подальшим практичним їх впровадженням.

Друге завдання передбачало розробку ефективних методів і технологій для поєднання природних і штучних особливостей у цифрових ландшафтах, одночасно оптимізуючи обробку, передачу та зберігання даних. Було вдосконалено окремі методи маніпулювання картою висот, розроблено модульний підхід для реалізації режимів змішування та метод інтеграції доріг до ландшафту. Крім того, було реалізовано ефективні структури даних для оптимізації обчислювальних процесів, що забезпечує безперебійну роботу та оптимізацію, і є важливим для реальних застосунків, таких як середовища віртуальної реальності та системи моделювання.

Третє завдання полягало у пошуку найбільш ефективних алгоритмів і методів для створення реалістичних ландшафтів з високою візуальною точністю та ефективною обробкою даних. В результаті було визначено та обґрунтовано найбільш корисні передові математичні підходи, включаючи дискретні гідродинамічні моделі та моделювання ерозії на основі фізики, для відтворення природних геологічних і гідрологічних процесів. Спрощені форми складних рівнянь, таких як рівняння Нав’є-Стокса та Ейлера, були ефективно адаптовані для досягнення реалістичних результатів без надмірних

обчислювальних витрат. Ці методи надають нові можливості побудови цифрових ландшафтів, які точно імітують такі природні явища, як відкладення осаду, повзучість матеріалу та ерозія русла річки.

Четверте завдання полягало у проєктуванні та розробці системи генерації ландшафту, здатної автоматизувати моделювання різноманітних цифрових рельєфів, одночасно забезпечуючи обчислювальну ефективність і зручність для користувача. Це було виконано, зокрема сформована програмна система, що має модульну структуру, в якій усі алгоритми реалізовані як незалежні компоненти, що можна повторно використовувати в різних програмах. Генератор ландшафтів об’єднує методи синтезу рельєфу, включаючи змішування карт висот, моделювання ерозії та створення дорожньої мережі, пропонуючи користувачам інтуїтивно зрозумілий інтерфейс для створення складних рельєфів. Модульний підхід не тільки підвищує гнучкість системи, але й розширює можливості її інтеграції в інші програмні платформи, що робить її цінним інструментом для розробників.

Всі види робіт, проведені в межах даного кваліфікаційного дослідження, є вагомими для сфери процедурної генерації рельєфу, оскільки завдяки інтеграції принципів гідродинаміки та моделювання на основі фізики стало можливим моделювати природні процеси з високим рівнем реалістичності з невеликими вимогами до комп’ютерно-мережевих систем. Розроблені методи ефективно збалансовують обчислювальну ефективність із візуальною точністю, вирішуючи ключові проблеми у створенні інтерактивного та захоплюючого середовища.

Крім того, дослідження підкреслює важливість оптимізації обчислювальних методів для забезпечення візуалізації в реальному часі та інтерактивного дизайну. Можливість створювати візуально привабливі рельєфи без надмірних потреб у ресурсах дозволяє впроваджувати розроблені методи в галузях із різноманітними технологічними можливостями, усуваючи розрив між теоретичними досягненнями та практичним впровадженням.

Гармонійне поєднання у застосунку методів, що забезпечують інтерактивність та модульність, покращують ефективність і реалістичність

процедурної генерації рельєфу, підкреслюють його важливість для практичного застосування в багатьох галузях, де використовують цифровий візуальний контент з реалістичними ландшафтними формами. Також корисним результатом проведеного дослідження є вирішена задача плавної інтеграції згенерованої дорожньої мережі в природний рельєф місцевості, яка реалізовується за допомогою алгоритму, що вирівнює рельєф уздовж траєкторії дороги та згладжує її краї з навколишнім ландшафтом.

Варто зазначити, що на основі проведених досліджень було розроблено програмне рішення, в якому реалізовується поєднання значної кількості розроблених ефективних методів та алгоритмів.

Матеріали даної кваліфікаційної роботи можуть бути корисними в подальших академічних дослідженнях, а також для дослідників і фахівців, які працюють в галузях моделювання та візуалізації цифрових природних середовищ, що поєднують в собі окремі антропогенні об’єкти. Це сприятиме розвитку у них розуміння прийомів поєднання природного рельєфу та штучно створених об’єктів, а також пошуку найбільш раціональних методів синтезу реалістичних і функціональних цифрових середовищ. Результати цього дослідження та розроблені автором методи стануть основою подальших науково-практичних досліджень націлених на вдосконалення процедур генерації цифрових ландшафтів шляхом розширення спектру модельованих природних та антропогенних процесів, на підвищення обчислювальної ефективності та впровадження методів машинного навчання для забезпечення оптимізації синтезу рельєфу. Майбутня робота може бути зосереджена на розробці адаптивних алгоритмів, здатних динамічно підлаштовуватися під різноманітні умови навколишнього середовища та вимоги користувачів, а також на застосуванні передових методів моделювання фізичних процесів для створення більш складних взаємодій між елементами рельєфу та дією сторонніх сил. Це сприятиме створенню багатомасштабних, високодеталізованих цифрових середовищ, які можуть подолати існуючий розрив між науковою точністю і практичним застосуванням, зокрема, у таких сферах, як геопросторовий аналіз, віртуальна реальність, ігри та міське планування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lawick van Pabst J., Jense H. Dynamic Terrain Generation Based on Multifractal Techniques. *High Performance Computing for Computer Graphics and Visualization*, 1996. DOI:10.1007/978-1-4471-1011-8_13.
2. Game Programmer Martz P. Generating random fractal terrain. URL: https://web.archive.org/web/20060420054134/http://www.gameprogrammer.com/fractal.html (date of access: 03.08.2024).
3. Perlin K. Chapter 4. In the beginning: The pixel stream editor. URL: https://redirect.cs.umbc.edu/~olano/s2002c36/ch04.pdf (date of access: 05.08.2024).
4. Станіславів О. С. Цифрове представлення ландшафтів за допомогою методів процедурної генерації. *Вісник Кам'янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки*. Випуск 16. Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2023. С. 79–84.
5. Miller G. S. P. The definition and rendering of terrain maps. *ACM SIGGRAPH Computer Graphics*, 1986. Vol. 20, Issue 4. DOI:10.1145/15886.15890.
6. Blatz M., Korn O. A Very Short History of Dynamic and Procedural Content Generation. *Game Dynamics*. Springer International Publishing, Cham, 2017. pp. 1–13. DOI:10.1007/978-3-319-53088-8_1.
7. Hendrikx M., Meijer S., Van Der Velden J., Iosup A. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1), 2013. pp. 1–22. DOI:10.1145/2422956.2422957.
8. Станіславів О. С. Розробка програмного застосунку для процедурної генерації рельєфу місцевості. *Збірник наукових праць студентів та магістрантів Кам'янець-Подільського національного університету імені Івана Огієнка*. Випуск 18. Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2024. С. 224–225.

9. Perlin K. An image synthesizer. *ACM SIGGRAPH Computer Graphics*. Vol. 19, Issue 3. pp. 287–296, July 1985.
10. Lagae A., Lefebvre S., Cook R., Derose T., Drettakis G., Ebert D. S., Lewis J. P., Perlin K., Zwicker M. A Survey of Procedural Noise Functions. *Computer Graphics Forum*, 29(8), 2010. Pp. 2579–2600. DOI:10.1111/j.1467-8659.2010.01827.x.
11. Marechal N., Galin E., Peytavie A., Maréchal N., Guérin E. Procedural Generation of Roads. *Computer Graphics Forum*, 2010. Vol. 29, Issue 2. Pp. 429-438. DOI:10.1111/j.1467-8659.2009.01612.x.
12. Zhang X., Zhong M., Liu S., Zheng L., Chen Y. Template-Based 3D Road Modeling for Generating Large-Scale Virtual Road Network Environment. *ISPRS International Journal of Geo-Information*, 8(9), 2019. DOI:10.3390/ijgi8090364.
13. Hong Z., Sun P., Tong X., Pan H., Zhou R., Zhang Y., Han Y., Wang J., Yang S., Xu L. Improved A-Star Algorithm for Long-Distance Off-Road Path Planning Using Terrain Data Map. *ISPRS International Journal of Geo-Information*, 10(11), 2021. DOI:10.3390/ijgi10110785.
14. Liu X., Zhan B., Ai T. Road selection based on Voronoi diagrams and “strokes” in map generalization. *International Journal of Applied Earth Observation and Geoinformation*, 12(1), 2010. Pp. 194–202. DOI:10.1016/j.jag.2009.10.009.
15. Wu J., Zhao Y., Yu M. et al. A new Voronoi diagram-based approach for matching multi-scale road networks. *Journal of Geographical Systems*, 25(2), 2023. Pp. 265–289. DOI:10.1007/s10109-023-00409-w.
16. Kelvin L. Z., Bhojan A. Procedural Generation of Roads with Conditional Generative Adversarial Networks. *Special Interest Group on Computer Graphics and Interactive Techniques Conference*. 2020. DOI:10.1145/3388770.3407422.
17. Galin E., Peytavie A., Marechal N., Guerin E. Procedural Generation of Roads. *Computer Graphics Forum*, 29(2), 2010. Pp. 429–438. DOI:10.1111/j.1467-8659.2009.01612.x.

18. Chen A., Darbon J., Morel J.-M. Landscape evolution models: A review of their fundamental equations. *Geomorphology*. Vol. 219, 2014. Pp. 68-86. DOI:10.1016/j.geomorph.2014.04.037.
19. Станіславів О. С., Жолтовський О. О., Смалько О. А. Комп'ютерне моделювання деяких природних процесів для генерації ландшафтів. Математичне та комп'ютерне моделювання. Серія: Технічні науки: зб. наук. праць. DOI:10.32626/2308-5916.2024-25.106-113.
20. Zholtovskyi O. O., Stanislaviv O. S., Smalko O. A. Landscape generation using discrete hydrodynamic models. *2024 IEEE 5th KhPI Week on Advanced Technology*. URL: https://2cm.es/MX5g
21. Beneš B. Real-time erosion using shallow water simulation. *Virtual reality interactions and physical simulation "VRIPHYS"*, Eurographics Association, 2007. DOI:10.2312/PE/vriphys/vriphys07/043-050.
22. 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)*, 2007. DOI:10.1109/PG.2007.15.
23. McDonald N. Simple particle-based hydraulic erosion. URL: https://nickmcd.me/2020/04/10/simple-particle-based-hydraulic-erosion/ (date of access: 12.10.2024).
24. Станіславів О. С. Генерація тривимірного ландшафту з використанням методу моделювання гідравлічної ерозії на основі частинок. Сучасні проблеми математичного моделювання, прогнозування та оптимізації: тези доповідей 10-ї Міжнародної наукової конференції. Пам'яті почесного професора Кам'янець-Подільського національного університету імені Івана Огієнка, д.т.н., професора, почесного академіка НАПНУ А.Ф. Верланя. Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2024. URL: https://2cm.es/MX5L
25. Belyaev A., Ohtake Y. A comparison of mesh smoothing methods. *Proc. Conf. on Geometric Modeling and Computer Graphics IKBN* 2003, Tel-Aviv, 2003. Pp. 83–87.

26. Anderson D. A., Tannehill J. C., Pletcher R. H., Munipalli R., Shankar V. Computational Fluid Mechanics and Heat Transfer. Fourth Edition. Abingdon: CRC Press, Taylor & Francis Group, 2021. DOI:10.1201/9781351124027.
27. Appel A. Some techniques for shading machine renderings of solids. URL: https://graphics.stanford.edu/courses/Appel.pdf (date of access: 16.10.2024).
28. Bridson R. Fluid simulation for computer graphics, 2nd ed. Boca Raton, FL: CRC Press, 2016. 276 p.
29. Cai X., Mengyao X., Yu N., Yang Z. A Terrain Elevation Map Generation Method Based on Self-Attention Mechanism and Multifeature Sketch. *Computational Intelligence and Neuroscience*, 2022. Pp. 1–19. DOI:10.1155/2022/9481445.
30. Camassa R., Holm D. D. An integrable shallow water equation with peaked solitons. *Physical Review Letters*. Vol. 71, Issue 11. 1993. Pp.1661-1664. DOI:10.1103/PhysRevLett.71.1661.
31. Campos R., Quintana J., Garcia R., Schmitt T., Spoelstra G., Schaap D. M. A. 3D simplification methods and large scale terrain tiling. *Remote Sensing*. Vol. 12, Issue 3. 2020. DOI:10.3390/rs12030437.
32. Foley J. D., Van Dam A. Fundamentals of interactive computer graphics (the systems programming series). Massachusetts : Addison-Wesley, 1983. 664 p.
33. Fournier A., Fussell D., Carpenter L. Computer rendering of stochastic models. *Communications of the ACM*, 1982. Vol. 25, Issue 6. DOI:10.1145/358523.358553.
34. Gagniere S., Hyde D., Marquez-Razon A., Jiang C. et al. A hybrid Lagrangian/Eulerian collocated advection and projection method for fluid simulation. *Computer Graphics Forum*. Vol. 39, Issue 8. 2020. Pp. 1–14.
35. Gotoh H., Sakai T., Shibahara T. Lagrangian flow simulation with sub-particle-scale turbulence model. *Proceedings of hydraulic engineering*, Vol. 44, pp. 575–580, February 2000.

36. Harlow F., Welch J. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids*. Vol. 8, No 12. 1965. Pp. 2182–2189.
37. Kamal K. R., Uddin Y. S. Parametrically controlled terrain generation. *Proceedings of the 5th international conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*. 2007. Pp. 17–23. DOI:10.1145/1321261.1321264.
38. Karsznia I., Wereszczyńska K., Weibel R. Make It Simple: Effective Road Selection for Small-Scale Map Design Using Decision-Tree-Based Models. *ISPRS International Journal of Geo-Information*, 11(8). 2022. DOI:10.3390/ijgi11080457.
39. Kodama Y. Solitons in Two-Dimensional Shallow Water. Philadelphia, PA: *SIAM-Society for Industrial and Applied Mathematics*, 2018. DOI:10.1137/1.9781611975529.
40. Li H., Kim S. Mesh simplification algorithms for rendering performance. *International journal of engineering research and technology*. Vol. 13, N. 6. 2020. DOI:10.37624/IJERT/13.6.2020.1110-1119.
41. McKee S., Tome M. F., Ferreira V. G., Cuminato J. A., Castelo A., Sousa F. S., Mangiavacchi N. The mac method. *Computers and Fluids*, Vol. 37, Issue 8. Pp. 907-930, 2008. DOI:10.1016/j.compfluid.2007.10.006.
42. Musgrave F. K., Kolb C. E., Mace R. S. The synthesis and rendering of eroded fractal terrains. *ACM SIGGRAPH Computer Graphics*, 1989. Vol. 23, Issue 3. DOI:10.1145/74334.74337.
43. Osher S., Fedkiw R. Level Set methods and dynamic implicit surfaces. *Applied Mathematical Sciences*. Vol. 153. Springer, 2003. Pp. 210–211.
44. Pharr M., Jakob W., Humphreys G. Physically based rendering: from theory to implementation. URL: https://pbr-book.org/4ed/contents (date of access: 15.10.2024).
45. Ruban A. I., Gajjar J. B., Fluid Dynamics. *Classical Fluid Dynamics*. Oxford: Oxford University Press, 2014. DOI:10.1093/acprof:oso/9780199681730.002.0003.

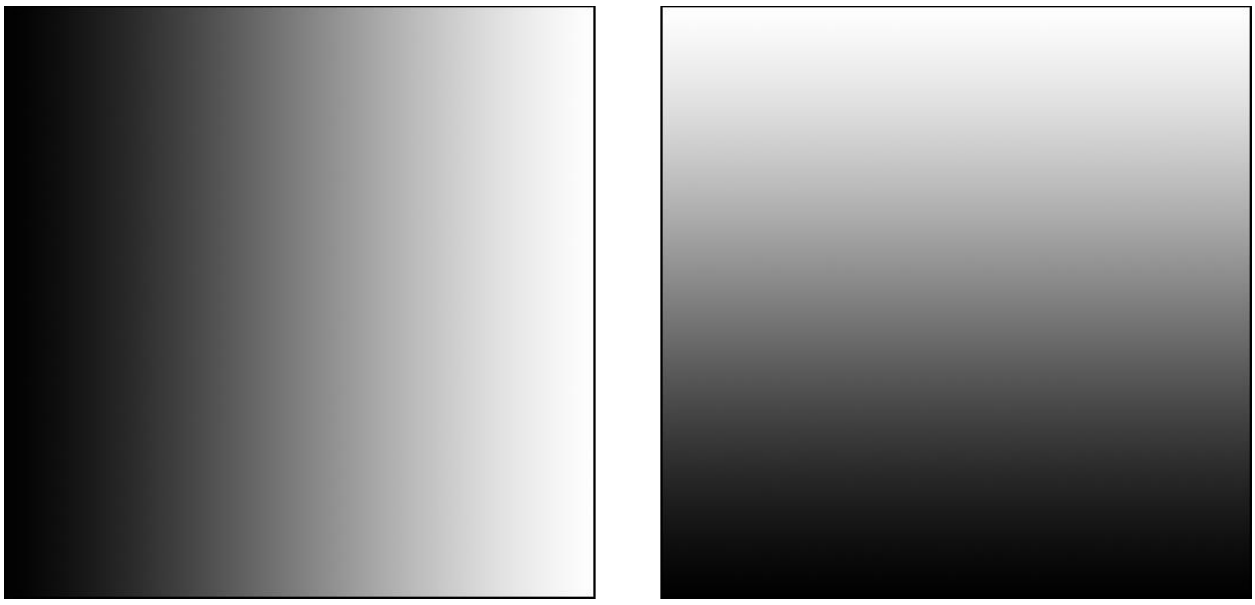
46. Schumacker R. A. Brand B., Gilliland M. G., Sharp W. H. Study for applying computer-generated images to visual simulation. Apollo Systems Department of General Electric Company. URL: https://2cm.es/Plmv (date of access: 16.10.2024).
47. Stanislaviv O. S., Zholtovskyi O. O., Smalko O. A. Generation techniques for realistic landscape informational models in immersive systems. *2024 IEEE 5th International Conference on Advanced Trends in Information Theory (ATIT)*. URL: https://2cm.es/N4C4
48. Tzathas P., Gailleton B., Steer P., Cordonnier G. Physically-based analytical erosion for fast terrain generation. *Computer Graphics Forum*, 43(2), 2024. DOI:10.1111/cgf.15033.
49. Vieira-e-Silva A. L., Brito C., Almeida M. W., Teichrieb V. Improved MPS method and its variations for simulating incompressible fluids on GPU. *SBC Journal on Interactive Systems*. Vol. 9, No 2. 2018. Pp. 52–67.
50. Williams L. Pyramidal Parametrics. *Computer Graphics*, 1983. Vol. 17, N. 3. DOI:10.1145/964967.801126.
51. Wu X. An efficient antialiasing technique. *ACM SIGGRAPH Computer Graphics*. Vol. 25, Issue 4. 1991. DOI:10.1145/127719.122734.
52. Yuan L., Li D., Tao F. Optimization of Terrain Generation Algorithm in Three-dimensional Real-time Modeling. *Geomatics and Information Science of Wuhan University*, 42(10), 2017. Pp. 1387–1393. DOI:10.13203/j.whugis20160100.
53. Смалько О. А., Костромицький А. І., Станіславів О. С., Коваленко О. М., Яковенко О. В. Метод кодування текстурної інформації з урахуванням особливостей пересилання та захисту відеозображень в інфокомунікаційних мережах. *Сучасна спеціальна техніка*. 2023. № 4. С. 62–69.
54. Станіславів О. С. Digital terrain representation using procedural generation methods. *Збірник наукових праць студентів та магістрантів Кам'янець-Подільського національного університету імені Івана Огієнка*. Випуск 18. Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2024. С. 37–39.

ДОДАТКИ

Додаток А

Порівняння впливів впроваджених режимів змішування на форму генерованого рельєфу

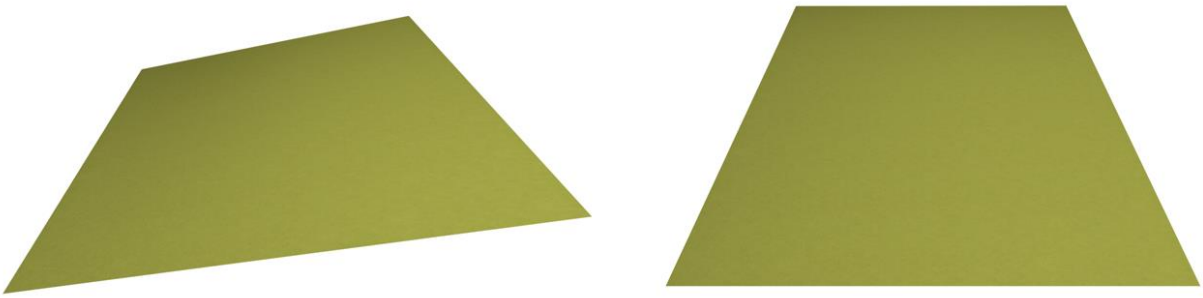
Кожен із впроваджених режимів змішування дозволяє імітувати різні властивості рельєфних структур при накладанні карт висот, контролюючи їх взаємодію шляхом регулювання відповідних внесків за допомогою певних математичних операцій. Порівняння будемо здійснювати шляхом накладання двох градієнтних карт висот розмірністю 512 на 512, що мають значення від 0 до 1 по ширині та висоті відповідно (рис. А.1). Застосовуючи такий підхід можна наочно продемонструвати дію цих режимів для усіх пар значень в діапазоні [0; 1].



a *b*

Рис. А.1. Карти висот утворені за допомогою лінійного градієнту:
a – значення змінюються лінійно горизонтально по всій ширині карти висот від 0 до 1;
b – значення змінюються лінійно вертикально по всій висоті карти висот від 0 до 1.

Якщо побудувати ландшафти на основі карт висот з рис. А.1, ми отримаємо наступний результат (рис. А.2).



a *b*

Рис. А.2. Згенеровані на основі карт висот ландшафти:
a – ландшафт згенерований на основі карти висот рис. А.1a;
b – ландшафт згенерований на основі карти висот рис. А.1b.

В межах дослідження пропонується використання наступного переліку режимів змішування, які, в свою чергу, інтегруються в розроблюваний програмний застосунок:

1. «Нормальний» (англ. «Normal») є найпростішим і використовується для відображення карти висот без змін. Він ідеально підходить для базового рельєфу, який потрібно створити без впливу інших карт. Даний режим часто використовується як початковий шар для експериментів з іншими режимами або як основа, до якої додаються нові деталі. Наприклад, якщо ви бажаєте створити рівнину чи базовий контур гірської місцевості, цей режим дозволяє зберегти природні форми без спотворення. Якщо припустити, що H_1 – карта висот на рис. А.1a, а H_2 – на рис. А.1b, то при накладанні H_2 на H_1 одержимо $H = H_2$ (рис. А.2b).
2. «Затемнення» (англ. «Darken») вибирає найнижчі значення висоти між двома картами, дозволяючи виділити западини чи долини. Це особливо корисно, коли потрібно створити більш глибокий рельєф або обмежити підвищення, зробивши рельєф плоскішим. Наприклад, якщо є дві карти, одна з яких представляє гори, а друга – долини, цей режим допоможе інтегрувати їх так, щоб збереглися тільки западини (рис. А.3).

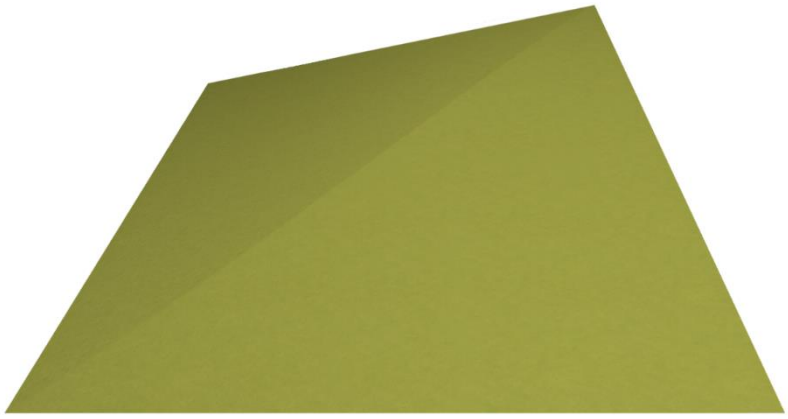


Рис. А.3. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Затемнення»

3. «Множення» (англ. «Multiply») підсилює взаємодію між двома картами висот. Кожна деталь першої карти помножується на значення другої, що дозволяє будувати складніші й динамічніші форми рельєфу, створюючи різкіші піки, більш згладжені рівнини, а також дозволяє ефективно поєднувати особливості рельєфу кожної з карти висот (рис. А.4).

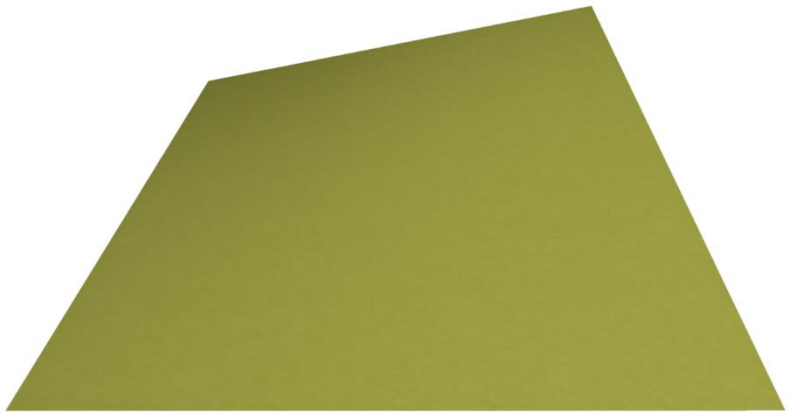


Рис. А.4. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Множення»

4. «Кольорове вигоряння» (англ. «Color Burn») надає рельєфу більш насиченіший вигляд, акцентуючи найглибші елементи місцевості. Це дозволяє створити рельєф із різкими ущелинами, скелями або долинами, підкреслюючи контрасти між височинами та низинами. Режим може використовуватися для драматичного відображення природних ландшафтів, таких як каньйони або розломи, де важливо акцентувати деталі.

5. «Лінійне вигоряння» (англ. «Linear Burn») зменшує значення висот, створюючи відчуття стискання рельєфу. Це дозволяє моделювати більш компактні форми місцевості, наприклад, вузькі долини або ущелини. Якщо перша карта представляє гладку поверхню, а друга – більш різкі структури, цей режим інтегрує їх, створюючи загальну карту, де кожна висота буде приглушена. Це корисно для підготовки низинних місцевостей, які мають бути детальнішими, але не такими помітними в загальній картині.
6. «Освітлення» (англ. «Lighten») залишає найвищі значення висоти між двома картами, підкреслюючи вершини та підвищення. Його можна використовувати для створення складних гірських ландшафтів, де важливо акцентувати височини, або для розробки плато чи гребенів. Наприклад, якщо одна карта представляє плавний горбистий рельєф, а друга – різкі підвищення, цей режим дозволяє інтегрувати їх так, щоб збереглися лише найвищі точки. Режим корисний для створення ландшафтів із чітко вираженими вершинами, які виділяються на загальному фоні (рис. А.5).

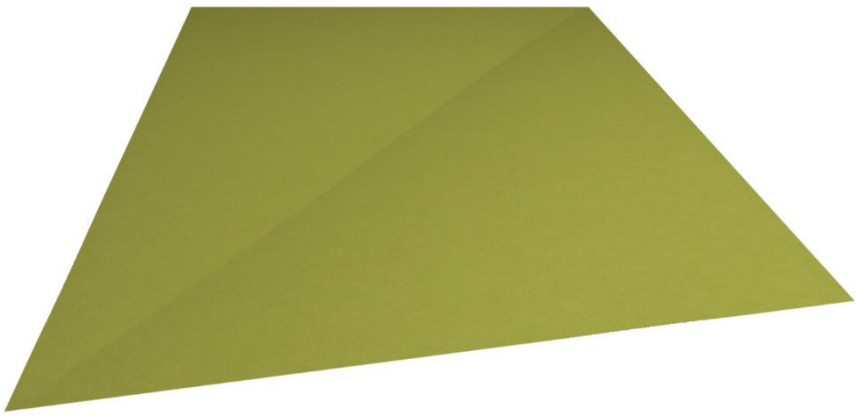


Рис. А.5. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Освітлення»

7. «Екран» (англ. «Screen») освітлює карту висот, зменшуючи контраст між різними висотами. Це створює більш плавний і вирівняний рельєф, що може бути корисним для розробки рівнин, піщаних дюн або інших поверхонь, де необхідно уникнути різких переходів. Цей режим також

допомагає зменшити візуальну складність ландшафту, що корисно для відображення великих територій із мінімальними деталями (рис. А.6).

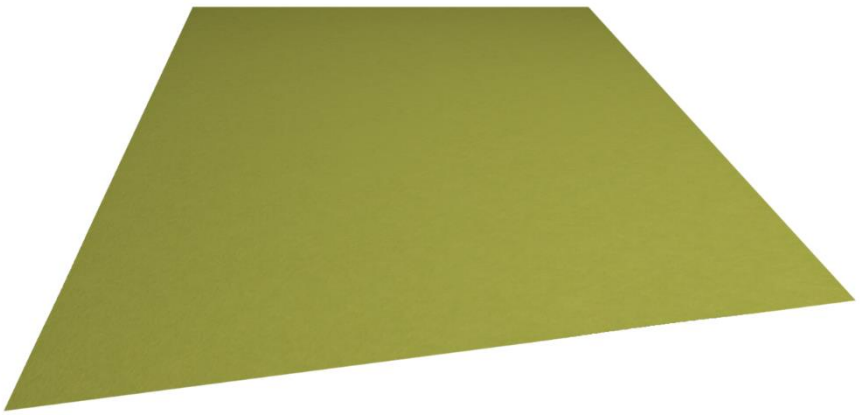


Рис. А.6. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Екран»

8. «Висвітлювання кольорів» (англ. «Color Dodge») підкреслює світлі ділянки карти висот, створюючи яскраві підвищення. Він ефективний для акцентування піків, особливо в умовах, коли рельєф потребує деталізації підвищених ділянок.
9. «Лінійне висвітлювання» (англ. «Linear Dodge») додає значення висот двох карт, створюючи більш насичений і виражений рельєф. Це дозволяє збільшити загальну висоту місцевості, що особливо корисно при моделюванні драматичних гірських ландшафтів, крутих схилів чи обривів. Наприклад, цей режим може використовуватися для злиття базового шару з деталізованими піками або хвилеподібними елементами, що додають текстурності. Також режим підходить для створення великих плато чи підняттів, які поступово зливаються в загальний ландшафт.
10. «Накладання» (англ. «Overlay») поєднує властивості режимів «Множення» та «Екран», підсилюючи контрасти між високими та низькими ділянками рельєфу. Темніші області стають ще темнішими, а світліші – світлішими, що дозволяє створювати яскраво виражені ландшафти з сильними перепадами висот. Наприклад, цей режим можна використовувати для моделювання вулканічних місцевостей, де

поєднуються глибокі кратери та високі вершини. Він також підходить для створення складних ландшафтів із вираженими деталями, таких як скельні утворення або урвища (рис. А.7).

Рис. А.7. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Накладання»

11. «М’яке світло» (англ. «Soft Light») створює м’який перехід між висотами, роблячи рельєф менш контрастним і більш природним. Це ідеальний вибір для створення плавних схилів, дюн або хвилястих рівнин, де не потрібна різка деталізація. Наприклад, цей режим добре підходить для моделювання пустельних ландшафтів, пологих пагорбів або берегових ліній. Він також використовується для корекції висот, коли необхідно зробити карту висот менш різкою і більш гармонійною.
12. «Жорстке світло» (англ. «Hard Light») підсилює контраст між висотами, створюючи виразний і різкий рельєф. Темні області стають глибшими, а світлі – піднятими, що дозволяє створювати динамічний ландшафт із яскраво вираженими деталями. Наприклад, цей режим можна використовувати для моделювання гірських масивів із крутими схилами, стрімкими урвищами та різкими піками.
13. «Лінійне світло» (англ. «Linear Light») значно змінює висоти рельєфу, додаючи сильний контраст і роблячи деталі вираженішими. Цей режим корисний для створення драматичних ландшафтів, таких як стрімкі скелі, каньйони чи гірські плато. Він також може використовуватися для інтеграції двох карт висот, де одна відповідає за основний рельєф, а друга

додає текстуру або дрібні деталі. Наприклад, цей режим ефективний для створення детальних карт скелястої місцевості з виразними вершинами та різкими переходами (рис. А.8).

Рис. А.8. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Лінійне світло»

14. «Яскраве світло» (англ. «Vivid Light») додає рельєфу сильний контраст, підсилюючи як темні, так і світлі області. Це дозволяє створювати візуально насичений ландшафт із глибокими ущелинами, високими вершинами та виразними текстурами. Режим корисний для акцентування деталей, наприклад, у гірських масивах або лісових територіях, де висота має суттєве значення для загальної композиції.
15. «Точкове світло» (англ. «Pin Light») виділяє окремі області карти висот, роблячи їх більш різкими залежно від початкового значення. Цей режим корисний для створення унікальних деталей у рельєфі, таких як ізольовані піки, западини чи невеликі структури, які мають привертати увагу. Наприклад, він добре підходить для моделювання ізольованих гірських вершин або окремих скель у пустелі (рис. А.9).

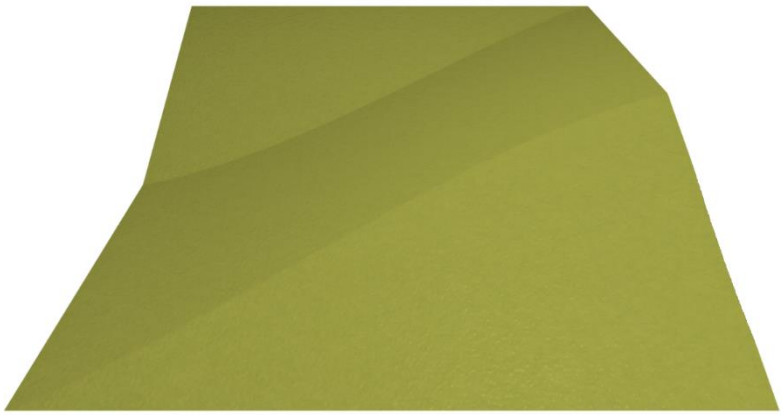


Рис. А.9. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Точкове світло»

16. «Різниця» (англ. «Difference») порівнює дві карти висот і залишає відмінності між ними. Це дозволяє аналізувати зміни рельєфу або створювати нові форми на основі взаємодії двох карт. Наприклад, цей режим можна використовувати для створення складних ландшафтів, де рельєф постійно змінюється, таких як вулканічні місцевості чи території з ерозійними процесами (рис. А.10).

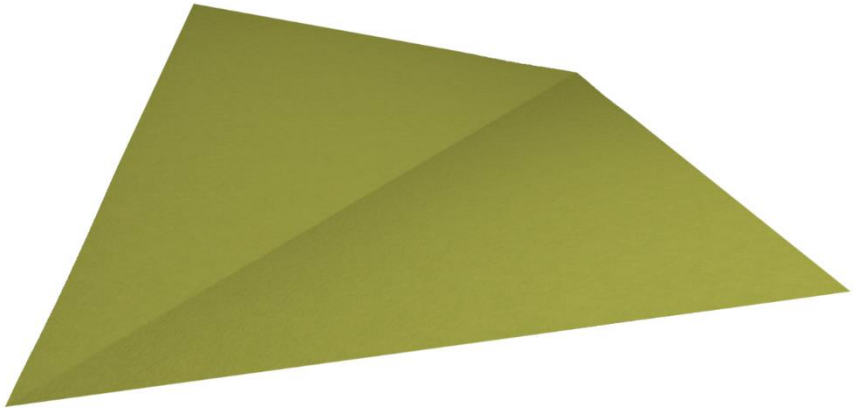


Рис. А.10. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Різниця»

17. «Виключення» (англ. «Exclusion») згладжує відмінності між двома картами висот, створюючи плавний, але контрастний рельєф. Його можна використовувати для створення природних ландшафтів, які виглядають гармонійно, але мають легкі варіації висот, наприклад, хвилясті пагорби або рівнини з мінімальними деталями.

18. «Віднімання» (англ. «Subtract») зменшує висоти першої карти, віднімаючи значення другої. Це дозволяє створювати западини, каньйони або інші форми рельєфу, які акцентують глибокі ділянки. Наприклад, цей режим ефективний для моделювання русел річок, ям або заглиблень у ландшафті.
19. «Ділення» (англ. «Divide») збільшує значення висоти першої карти, створюючи більш різкий рельєф. Цей режим корисний для створення гірських хребтів або підняття загальної висоти місцевості, що дозволяє формувати надзвичайно круті переходи (рис. А.11).

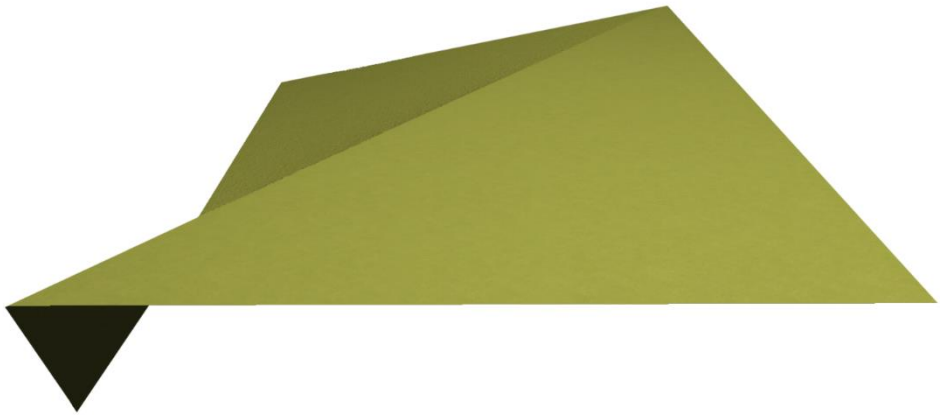


Рис. А.11. Ландшафт згенерований шляхом накладання H_2 на H_1 з режимом змішування «Ділення»

Додаток Б

Фрагмент програмного коду для формування моделі ландшафту

```
mVertexIndices.clear();
mVertices.clear();

//CalculatingNormals
float* expandedHeightmap = new float[(widthPoly + 3) * (lengthPoly + 3)];
glm::vec3* faceNormals = new glm::vec3[(widthPoly + 2) * (lengthPoly + 2)];

//SettingMiddleValues
for (int z_count = 0; z_count < lengthPoly + 1; z_count++)
{
    for (int x_count = 0; x_count < widthPoly + 1; x_count++)
    {
        expandedHeightmap[(z_count + 1) * (widthPoly + 3) + x_count + 1] =
resultHeightmap[z_count * (widthPoly + 1) + x_count] * terHeight * 0.004f;
    }
}
//SettingUpperAndBottomBorders
for (int x_count = 0; x_count < widthPoly + 1; x_count++)
{
    //UpperBorder
    expandedHeightmap[x_count + 1] = expandedHeightmap[x_count + widthPoly + 4];

    //BottomBorder
    expandedHeightmap[(lengthPoly + 2) * (widthPoly + 3) + x_count + 1] =
expandedHeightmap[(lengthPoly + 1) * (widthPoly + 3) + x_count + 1];
}
//SettingLeftAndRightBorders
for (int z_count = 0; z_count < lengthPoly + 1; z_count++)
{
    //LeftBorder
    expandedHeightmap[(z_count + 1) * (widthPoly + 3)] = expandedHeightmap[(z_count
+ 1) * (widthPoly + 3) + 1];

    //RightBorder
    expandedHeightmap[(z_count + 1) * (widthPoly + 3) + widthPoly + 2] =
expandedHeightmap[(z_count + 1) * (widthPoly + 3) + widthPoly + 1];
}
//SettingCornersBorders
expandedHeightmap[0] = expandedHeightmap[widthPoly + 4];
expandedHeightmap[widthPoly + 2] = expandedHeightmap[2 * widthPoly + 4];
expandedHeightmap[(lengthPoly + 2) * (widthPoly + 3)] =
expandedHeightmap[(lengthPoly + 1) * (widthPoly + 3) + 1];
expandedHeightmap[(lengthPoly + 3) * (widthPoly + 3) - 1] =
expandedHeightmap[(lengthPoly + 2) * (widthPoly + 3) - 2];

//CalculatingFaceNormals
glm::vec3 vectorA;
glm::vec3 vectorB;
glm::vec3 crossAxB_1;
glm::vec3 crossAxB_2;
float faceWidth;
float faceLength;

faceWidth = (2.0f / widthPoly) * terWidth * 0.002f;
faceLength = (2.0f / lengthPoly) * terLength * 0.002f;

for (int z_count = 0; z_count < lengthPoly + 2; z_count++)
{
```

```
    for (int x_count = 0; x_count < widthPoly + 2; x_count++)
    {
        vectorA = { 0, expandedHeightmap[(z_count + 1) * (widthPoly + 3) + x_count]
- expandedHeightmap[z_count * (widthPoly + 3) + x_count], faceLength};
        vectorB = { faceWidth, expandedHeightmap[z_count * (widthPoly + 3) + x_count
+ 1] - expandedHeightmap[z_count * (widthPoly + 3) + x_count], 0 };
        crossAxB_1 = glm::cross(vectorA, vectorB);

        vectorA = { 0, expandedHeightmap[z_count * (widthPoly + 3) + x_count + 1] -
expandedHeightmap[(z_count + 1) * (widthPoly + 3) + x_count + 1], -faceLength };
        vectorB = { -faceWidth, expandedHeightmap[(z_count + 1) * (widthPoly + 3) +
x_count] - expandedHeightmap[(z_count + 1) * (widthPoly + 3) + x_count + 1], 0 };
        crossAxB_2 = glm::cross(vectorA, vectorB);

        faceNormals[z_count * (widthPoly + 2) + x_count] =
glm::normalize((crossAxB_1 + crossAxB_2) * 0.5f);
    }
}

//AddingVertices
for (int z_count = 0; z_count < lengthPoly + 1; z_count++)
{
    for (int x_count = 0; x_count < widthPoly + 1; x_count++)
    {
        VertexHolder vh;
        vh.mPos = { (2.0f * x_count / widthPoly - 1) * terWidth * 0.002f,
resultHeightmap[z_count * (widthPoly + 1) + x_count] * terHeight * 0.004f ,(2.0f *
z_count / lengthPoly - 1) * terLength * 0.002f };
        vh.mNormal = glm::normalize((faceNormals[z_count * (widthPoly + 2) +
x_count] + faceNormals[z_count * (widthPoly + 2) + x_count + 1] +
faceNormals[(z_count + 1) * (widthPoly + 2) + x_count] + faceNormals[(z_count + 1) *
(widthPoly + 2) + x_count + 1]) * 0.25f);
        vh.mTexture = { 1.0f * x_count / widthPoly, 1.0f * z_count / lengthPoly };

        add_vertex(vh);
    }
}

//AddingTriangleFacesIndices
for (int z_face = 0; z_face < lengthPoly; z_face++)
{
    for (int x_face = 0; x_face < widthPoly; x_face++)
    {
        add_vertex_index((z_face + 1) * (widthPoly + 1) + x_face);
        add_vertex_index(z_face * (widthPoly + 1) + 1 + x_face);
        add_vertex_index(z_face * (widthPoly + 1) + x_face);

        add_vertex_index((z_face + 1) * (widthPoly + 1) + x_face);
        add_vertex_index((z_face + 1) * (widthPoly + 1) + 1 + x_face);
        add_vertex_index(z_face * (widthPoly + 1) + 1 + x_face);
    }
}
```

Додаток В

Реалізація карти висот ландшафту із синусоїдальними формами

```
void SinusoidalWave(int number)
{
    for (int z_count = 0; z_count < lengthPoly + 1; z_count++)
    {
        for (int x_count = 0; x_count < widthPoly + 1; x_count++)
        {
            float value = 0.0f;
            float x_val = (2.0f * x_count / widthPoly - 1) * pi * x_frequency +
x_offset;
            float z_val = (2.0f * z_count / lengthPoly - 1) * pi * z_frequency +
z_offset;

            if (isRadial)
            {
                value = std::sinf(std::sqrtf(x_val * x_val + z_val * z_val) +
radial_offset);
            }
            else
            {
                value = std::sinf(x_val) * std::sinf(z_val);
            }

            value = 0.5f * value + 0.5f;

            value = value * (1 - isInverted) + (1 - value) * isInverted;

            if (isAdjust)
            {
                if (coef > 0)
                {
                    if (coef > 1 || coef < 1)
                    {
                        value = (std::powf(coef, value) - 1) / (coef - 1);
                    }
                }
            }

            layerHeightmaps[number].push_back(value * intensity);
        }
    }
}
```