Харченко А. В., магістрант,
Левківський В. Л., викладач,
Колос К. Р., д.пед.н., професор
Державний університет «Житомирська політехніка»

ПРОЦЕДУРНА ГЕНЕРАЦІЯ ІГРОВИХ СВІТІВ

Процедурна генерація є важливим та актуальним елементом у розробці сучасних ігор і віртуальних світів. Цей підхід дозволяє створювати дуже різноманітні та нескінченно ігрові середовища, що змінюються, забезпечуючи унікальний досвід для гравця кожного разу, коли він запускає гру. Важливість процедурної генерації полягає в її здатності створювати величезні, яскраві світи з куточками, які можна досліджувати, зберігаючи при цьому зацікавленість гравця, кидаючи йому виклик. Крім того, цей підхід оптимізує використання ресурсів і дозволяє розробляти ігрові проекти з меншим бюджетом і в коротші терміни. Процедурна генерація продовжує впливати на індустрію розробки ігор і стала важливим інструментом для досягнення варіативності, реалістичності та глибини ігрових світів.

Метою дослідження є аналіз основних алгоритмів і підходів до генерації ігрових світів.

Під час процедурної генерації використовуються різноманітні алгоритми та підходи. Одним з найпоширеніших є використання шуму Перліна. Це тип градієнтного шуму, розроблений Кеном Перліном у 1983 році. Він має багато застосувань, включаючи, але не обмежуючись: процедурне генерування рельєфу, застосування псевдовипадкових змін до змінної та допомога у створенні текстур зображення. Найчастіше він реалізується у двох, трьох або чотирьох вимірах, але може бути визначений для будь-якої кількості вимірів [1].

Іншим доволі популярним алгоритмом є Diamond-Squarealgorithm або алгоритм «Ромбоподібний Квадрат». Це метод створення карт висот для комп'ютерної графіки. Це кращий алгоритм, ніж тривимірна реалізація алгоритму зміщення середньої точки, яка створює двовимірні пейзажі.

Алгоритм ромбовидного квадрата починається з двовимірного квадратного масиву шириною та висотою 2n + 1. Для чотирьох кутових точок масиву спочатку потрібно встановити початкові значення. Потім по черзі виконуються кроки ромба та квадрата, доки не буде встановлено всі значення масиву.

Кожне випадкове значення множиться на константу, що відповідає за масштаб, яка зменшується з кожною ітерацією на коефіцієнт 2-h, де h є значенням від 0,0 до 1,0. Під час квадратних кроків точки, розташовані на краях масиву, матимуть лише три суміжні значення, а не чотири. Існує кілька способів впоратися з цим ускладненням. Найпростішим є взяти середнє лише трьох суміжних значень. Іншим варіантом є «обгортання», беручи четверте значення з іншого боку масиву. При використанні з узгодженими початковими кутовими значеннями цей метод також дозволяє з'єднати згенеровані фрактали без розривів [2].

Ще одним підходом при генерації ігрових світів є використання фракталів. У математиці фрактал – це геометрична фігура, що містить детальну структуру в довільно малих масштабах, зазвичай має фрактальну розмірність, що строго перевищує топологічну розмірність. Багато фракталів виглядають подібними в різних масштабах, як показано на послідовних збільшеннях множини Мандельброта. Ця демонстрація подібних візерунків у дедалі менших масштабах називається самоподібністю, також відомою як розширена симетрія або розгорнута симетрія. Якщо ця реплікація абсолютно однакова в кожному масштабі, як у губки Менгера, форма називається афінною самоподібною.Фрактали відіграють доволі важливу роль у сучасній процедурній генерації ігрових світів, пропонуючи можливість створювати складні та реалістичні ландшафти [3].

Отже, процедурна генерація ігрових світів завжди була важливим елементом ігрової індустрії, а використання різних алгоритмів і підходів демонструє неймовірну силу інновацій у цій галузі. Описані вище алгоритми та методи можуть надати гравцям неймовірно різноманітні та унікальні ігрові світи. Ці підходи допомагають оптимізувати ресурси, покращити процес розробки ігор та розширити можливості створення віртуальних світів з високим рівнем занурення. Використання різних алгоритмів і підходів є важливим кроком в еволюції ігрового дизайну, що дозволяє гравцям насолоджуватися іграми, які пропонують більше свободи, варіативності та реалізму.

Список використаних джерел

1. Perlin Noise: A Procedural Generation Algorithm.Raouf's blog. URL: https://rtouti.github.io/graphics/perlin-noise-algorithm .
2. TerrainGeneration – Diamond Square Algorithm. Daniel Beard's Programming Blog. URL: https://danielbeard.wordpress.com/2010/08/07/terrain-generation-and smoothing/.
3. The Mandelbrot Set – Fractals – Mathigon. Mathigon. URL: https://uk.mathigon.org/course/fractals/mandelbrot.