УДК 004.92

О. І. Стрижалов[1]
О. В. Сілагін[1]

АНАЛІЗ МЕТОДІВ Й АЛГОРИТМІВ ГЕНЕРАЦІЇ ФРАКТАЛЬНЫХ ШУМІВ ДЛЯ ЗАДАЧ ГЕНЕРАЦІЇ АМОРФНИХ ОБ'ЄКТІВ.

[1] Вінницький національний технічний університет;

Анотація
В даній роботі розглянуто та проаналізовано методи й алгоритми генерації фрактальных шумів, які можуть бути використані для виконання задач генерації аморфних об'єктів.
Ключові слова: генерація аморфних об'єктів, фрактал, процедурні шуми.

Abstract *Methods and algorithms for fractal noise generation that can be used to perform amorphous object generation problems are considered and analyzed.*
Keywords*: amorphous objects generation, fractals, procedural noises.*

Вступ

В даний час програмна генерація аморфних об'єктів в комп'ютерній графіці стає все більше актуальною. Зростають вимоги до якості та фотореалістичності зображень, наприклад пейзажів, що в свою чергу підвищує вимоги до пам'яті, так як більш якісні зображення займають все більші обсяги пам'яті. Вирішення даної проблеми - використання процедурних шумів, що вимагає незначної кількості обчислювальних ресурсів для програмної генерації саме аморфних об'єктів. В даній роботі буде розглянуто методи й алгоритми генерації фрактальных шумів, які можуть бути використані для виконання задач генерації аморфних об'єктів..

Основна частина

Для задач генерації аморфних об'єктів використовуються такі методи та алгоритми: шум Уорлі, симплексний шум, хвильовий шум, градієнтний шум, шум OpenSimplex та алгоритм Diamond-Square.

Шум Уорлі - це функція шуму, запроваджена Стівеном Ворлі в 1996 році. У комп'ютерній графіці він використовується для створення процедурних текстур, тобто текстур, які створюються автоматично в довільній точності і їх не потрібно малювати вручну. Шум Уорлі близький до імітації текстур каменю або води. Основна ідея алгоритму полягає в тому, щоб взяти випадкові точки в просторі, а потім для кожної точки простору взяти відстань до n-ї найближчої точки (наприклад, другої найближчої точки) як інформацію про колір. Випадково розподіліть особливі точки в просторі, шум Fn (x) - відстань до n-ї найближчої точки до точки x. У випадку двох вимірів потрібно створити двадцять п'ять квадратів (у розміщенні п'ять на п'ять), щоб точно знайти найближчі. Зазвичай для практичних застосувань вважається достатньо дев'яти точок квадрата (у розташуванні три на три) [6].

Симплексний шум - це метод побудови функцій шуму n-мірних розмірів, порівнянних із шумом Перліна ("класичний" шум), але з меншою кількістю артефактів, у більших розмірах, і з меншими обчислювальними накладними витратами. Кен Перлін розробив алгоритм у 2001 році з метою зняття обмежень класичної шумової функції.

Переваги симплексного шуму над шумом Перліна є те, що він має меншу обчислювальну складність і вимагає меншого застосування множення. Прості шкали шуму до більш високих розмірів (4D, 5D) із значно меншими обчислювальними витратами порівняно з класичним шумом. Симплексний шум не має помітних спрямованих артефактів (візуально ізотропний), хоча шум, що створюється для різних вимірів, візуально відрізняється, наприклад, 2D-шум має інший вигляд, ніж фрагменти 3D-шуму. Симплексний шум можна обчислити без значних витрат пам'яті.

Симплексний шум можна легко реалізувати апаратно. У той час як класичний шум інтерполює між градієнтами в оточуючих кінцевих точках гіперрешітки, симплексний шум ділить простір на симплекти (тобто n-мірні трикутники). Це зменшує кількість точок для обчислень. Симплексний шум корисний для програм комп'ютерної графіки, де шум зазвичай обчислюється для 2, 3, 4 або, можливо, 5 вимірів [4].

Хвильовий шум є альтернативою шуму Перліна, що зменшує проблеми згладжування та втрати деталей, які виникають при зведенні шуму Перліна у фрактал. Значення шуму - це тип шуму, який зазвичай використовується як примітивний текст текстури в комп'ютерній графіці. Він концептуально відрізняється від частотного і градієнтного шумів, прикладами якого є шум Перліна та Симплекс-шум . Цей метод полягає у створенні решітки точок, яким присвоюються випадкові значення. Потім функція шуму повертає інтерпольоване число на основі значень оточуючих точок решітки. Для багатьох застосувань кілька октав цього шуму можна генерувати та потім підсумовувати разом, як це можна зробити з шумом Перліна та шумом Simplex, щоб створити форму фрактального шуму[5].

Градієнтний шум - різновид алгоритмів генерації шуму, що використовуються в комп'ютерній графіці для створення процедурних текстур. При генерації градиентного шуму створюється решітка випадкових градієнтів, які потім інтерполюються для отримання значень в точках, що лежать між вузлами решітки [2].

Шум OpenSimplex - це функція шуму n-мірних градієнтів, яка була розроблена з метою подолання проблем, пов’язаних з патентом навколо симплексного шуму. Також дозволяє уникати візуальних артефактів спрямованості, характерних для шуму Перліна. Алгоритм дещо подібний шуму Simplex, але має дві основні відмінності. Так симплексний шум починається з гіперкубічної соти стільника і збиває її по головній діагоналі, щоб сформувати структуру її сітки, а шум OpenSimplex замість цього змінює косий і зворотно-косий фактори і використовує розтягнуту гіперкубічну соту. Розтягнута гіперкубічна сота після обробки перетворюється на симплектичну соту. Це означає, що 2D Simplex та 2D OpenSimplex використовують різні орієнтації трикутної плитки, тоді як 3D Simplex використовує тетрагональну дисфероїдну соту, а 3D OpenSimplex використовує тетраедрально-октаедричну соту. Шум OpenSimplex використовує більший розмір ядра, ніж шум Simplex [3].

Алгоритм Diamond-Square - це метод генерації аморфних об’єктів для комп'ютерної графіки. Це трохи кращий алгоритм, ніж тривимірна реалізація алгоритму зміщення середньої точки, яка виробляє двовимірні пейзажі. Він також відомий як фрактал випадкового зміщення середньої точки, фрактал хмари або фрактал плазми, через ефект плазми, що утворюється при застосуванні. Алгоритм Diamond-Square починається з двовимірного квадратного масиву ширини і висоти $2n + 1$, де потім випадковим чином генерується значення точок. Спочатку потрібно встановити значення чотирьох кутових точок масиву. Потім кроки Diamond та Square виконуються поперемінно, поки не буде встановлено всі значення масиву. Крок Diamond: для кожного квадрата масиву встановити середню точку цього квадрата як середнє з чотирьох кутових точок плюс випадкове значення. Крок Square: для кожного ромбу в масиві встановити середню точку цього ромбу як середнє з чотирьох кутових точок плюс випадкове значення. При кожній ітерації величина випадкового значення повинна бути зменшена. Під час кроків Square точки, розташовані на краях масиву, матимуть лише три суміжні значення, а не чотири. Існує декілька способів розв'язання цієї проблеми - найпростіше взяти середнє значення тільки трьох суміжних значень. Іншим варіантом є "обгортання", де четверте значення одержуємо з іншої сторони масиву. Цей метод також дозволяє згенерованим фракталам об'єднуватися без розривів. Алгоритм може бути використаний для створення реалістичних ландшафтів, і різні реалізації його широко використовуються в програмному забезпеченні комп'ютерної графіки. Він також застосовується для генерації компонент процедурних текстур [1].

Висновки

Серед розглянутих алгоритмів обрано алгоритм Diamond-Square, який порівняно з іншими краще відповідає вимогам та цілям магістерської кваліфікаційної роботи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. [Електронний ресурс] // Навчальні матеріали – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Алгоритм_Diamond-Square
2. [Електронний ресурс] // Навчальні матеріали – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Градиентный_шум
3. [Електронний ресурс] // Навчальні матеріали – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/OpenSimplex_noise
4. [Електронний ресурс] // Навчальні матеріали – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Simplex_noise
5. [Електронний ресурс] // Навчальні матеріали – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Wavelet_noise
6. [Електронний ресурс] // Навчальні матеріали – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Worley_noise

Стрижалов Олександр Ігорович. — студент групи 1КН-15б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: 1kn15b.stryzhalov@gmail.com.

Науковий керівник: ***Сілагін Олексій Віталійович.*** — к.т.н., доцент кафедри комп'ютерних наук факультету інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця. Відповідальний за організацію бакалаврського дипломного проектування. e-mail: avsilagin@gmail.com.

Stryzhalov Oleksandr I. — Department of information technologies and computer engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: 1kn15b.stryzhalov@gmail.com.

Supervisor: ***Silahin Olexii V.*** — Candidate of Technical Sciences, Associate Professor, Department of Computer Science, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia. Responsible for organization of bachelor diploma design. e-mail: avsilagin@gmail.com.