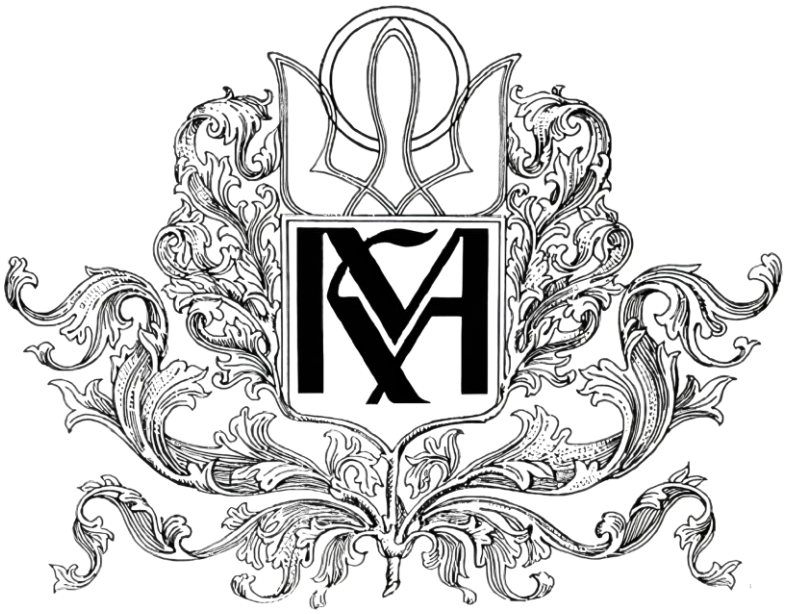
Міністерство освіти і науки України НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ» Кафедра інформатики факультету інформатики



Генерація 3Д світів на основі даних користувача

Текстова частина до курсової роботи за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
Афонін А.О

_____

(підпис)
“____” _____ 2022 р.

Виконав студент
Сурженко В.О
“____” _____ 2022 р.

Київ 2022

Тема: Генерація 3Д світів на основі даних користувача

Календарний план виконання роботи:

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової роботи	01.11.2021	
2.	Вивчення предметної області	03.12.2021	
3.	Проведення досліджень	13.01.2022	
4.	Побудова технічного завдання	25.01.2022	
5.	Написання першої частини курсової роботи	13.02.2022	
6.	Написання другої частини курсової роботи	20.02.2022	
7.	Написання висновків курсової роботи	15.04.2022	
8.	Аналіз отриманих результатів з керівником	21.04.2022	
9.	Коригування роботи	13.05.2022	
10.	Створення презентації	28.05.2022	
11.	Захист роботи		

Студент Сурженко В.О

Керівник Афонін А.О

“____” _____

ЗМІСТ

Анотація 3
Вступ 4
РОЗДІЛ 1. Віртуальні світи 6
1.1 Що таке віртуальний світ 6
1.2 Що таке процедурна генерація 7
1.3 Сфери використання процедурно згенерованих віртуальних світів 8
1.3 Висновок до розділу 1 9
РОЗДІЛ 2. Алгоритми генерації 3Д світів 10
2.1 Генерація полігональних карт 10
2.1.1 Ідея та алгоритм 10
2.2.2 Аналіз алгоритму 12
2.2 Згортання хвильової функції 12
2.3 Шум Перлина 14
2.4 Клітинний автомат 16
2.4 Висновок до розділу 2 18
РОЗДІЛ 3. Реалізація алгоритму генерування 3Д світу на основі вхідних даних користувача 19
3.1 Створення програмному додатку 19
3.2 Генерування мапи шуму 21
3.3 Генерація мапи світу 22
3.4 Отримання результату 23
3.5 Висновок до розділу 3 23
ВИСНОВОК 24
ВИКОРИСТАНІ ДЖЕРЕЛА 25
ВИКОРИСТАНІ АССЕТИ 26

Анотація

У цій курсовій роботі розглянуті основні принципи генерування 3Д світів для використання їх надалі у кіно, ігро та настільних індустріях. Описані принципи та нюанси різних методів генерування. А також створений тестовий додаток задля демонстрації принципу та нюансів роботи з одним з поданих алгоритмів.

Вступ

Генерація об’єктів, ландшафтів та світів, останнім часом, це досить актуальна та поширена тема. З кожним роком все більше індустрій та проєктів потребують створення великої кількості об'єктів або земних мас, що будуть підпорядковуватися певному алгоритму, який можна біло б спокійно налаштувати під свої потреби, але з умовою унікальності та неповторності таких структур. Саме тому потреба у процедурній генерації з кожним роком більшає.

Для повноцінного створення віртуальних світів та об’єктів не досить просто задати певний алгоритм, також треба ввести або отримати вхідні дані, які й будуть лежати в основі генерації комп'ютерного простору. І якщо раніше, таке завдання могло бути майже непосильним через брак як вхідних даних, так і комп'ютерних потужностей, то зараз, задав правильний алгоритм та налаштування, можна генерувати не тільки ландшафти, а цілі всесвіти.

Вже сьогодні, багато ігрових студій пишуть свої алгоритми генерації світів задля подальшого створення гри на отриманій карті, що драматично спрощує та скорочує терміни розробки ААА проєктів з відкритим світом, де карти можуть досягати сотень квадратних кілометрів у розмірі. Що вже там казати про проєкти, які цілком і повністю базуються на процедурно згенерованих світах.

Є багато методів та технік генерування світів та ландшафтів, але всі вони потребують детального проєктування та правильного налаштування перед початком генерації. Саме тому вкрай важливо обрати потрібний метод для певного результату, адже навіть один алгоритм при різних налаштуваннях дає 2 кардинально різні результати, що казати вже про різні методи генерування.

Мета цієї курсової роботи - розглянути різні методи процедурного генерування світів, та дослідити певний з них, написавши до нього застосунок, який би наочно демонстрував його можливості.

Текстова частина курсової роботи складається з трьох розділів.

У першому розділі описується теоретична інформація, потрібна для побудови алгоритму та подальшого процедурного генерування світу.

У другому розділі розглянуті найпоширеніші алгоритми та методи процедурної генерації. Їх можна виділити 4:

- Генерування полігональних карт
- Згортання хвильової функції
- Шум Перлина
- Клітинний автомат

У третьому розділі розглянутий створений додаток для процедурної генерації світу на основі вхідних даних користувача та комп'ютерно згенерованого шуму Перлина (Perlin noise). Створений застосунок генерує світ з використанням 3Д моделей та наявно демонструє роботу та принципи використання даного алгоритму.

РОЗДІЛ 1. Віртуальні світи

1.1 Що таке віртуальний світ

Віртуальний світ - комп'ютерне середовище, штучно створене певною групою людей з використанням програм для користувачів, щоб ті могли в певній мірі взаємодіяти з елементами або іншими користувачами даного світу. Віртуальні світи створені для того, щоб користувачі могли проводити там час. Віртуальні світи складаються з різних цифрових та програмних елементів та у своїй сутності схожі з віртуальною 3Д середую. Такі змодельовані світи можуть бути взяті з реального світу, вигаданого чи повністю згенеровані машиную. Через це у машин та комп'ютерів з'являється задача обробки, зберігання та роботи з такими світами, а у розробників створення та використання.

Термін віртуальний світ отримав різне тлумачення в залежності від точки зору та сфери використання:

- *“Синхронна, постійна спільнота людей, що мають певне представлення - аватар”* - **Марк Беллі, 2008р.**
- *“Автоматизоване, спільне та постійне середовище, за допомогою якої люди можуть взаємодіяти одне з одним або з віртуальним простором”* - **Річард Бартл, 2010р.**
- *“Постійне, інтерпретоване середовище, яке надає користувачам аватарки для взаємодії в реальному часі одне з одним або всередині самого світу”* - **Кіріна Гірван, 2013р**

Певного, чіткого поняття віртуального світу дати неможливо, але це об'єкт, симуляція, який дає певне представлення о просторі та підпорядковується певним, своїм правилам.

1.2 Що таке процедурна генерація

Процедурна генерація - процес автоматичного створення контенту за допомогою певного алгоритму. Формулюючи по іншому, процедурна генерація це програма, що може створювати контент самостійно, або за допомогою оператора, людини, що коригує процесс створення. Результатом генерації може бути цілий перелік речей: мапа, світ, квести, предметы, моделі, транспорт, зброя, вороги, музика ті інше. Ключовою властивістю повинна бути функціональність - юзер повинен мати змогу використовувати сгенерированный контент.

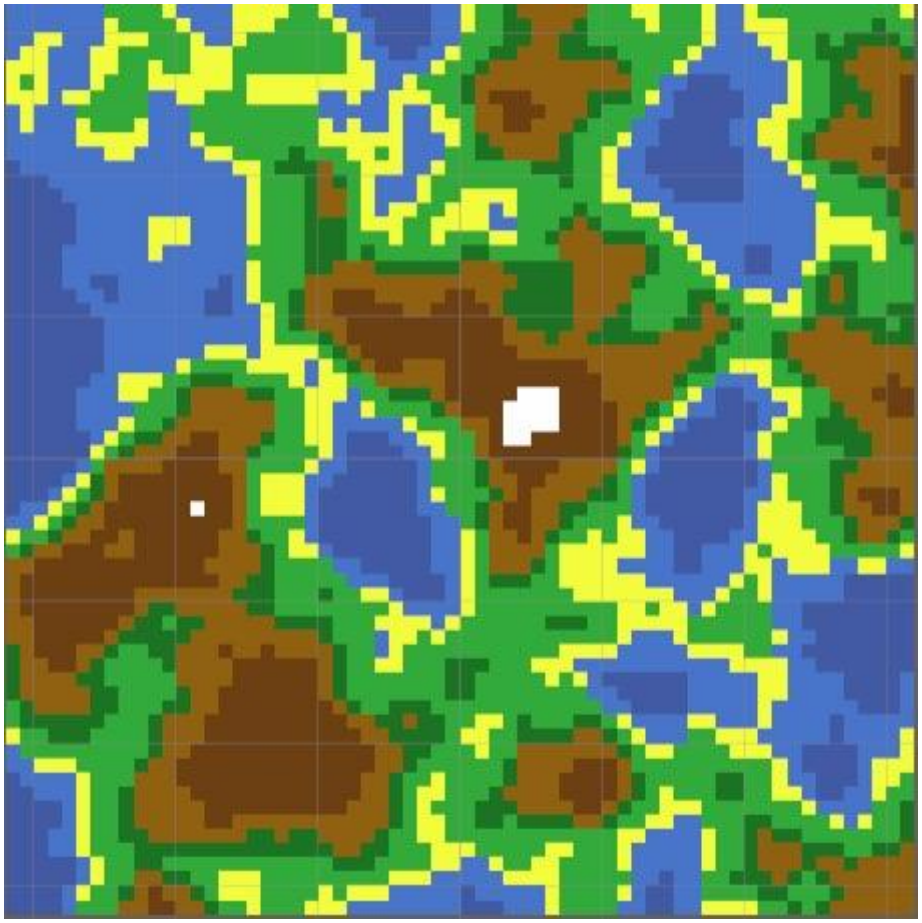


Рисунок 1.1 - Приклад процедурної генерації

Термін процедурній має під собою певну процедуру, процес запущений на комп'ютері для генерації. Основні властивості генерації можна виділити такі:

- **Можливість контролю**: Можливість контролювати та підлаштовувати алгоритм під свої потреби.
- **Швидкість**: в залежності від задачі, потрібен різний час, але, як правило, алгоритм-програма повинна генерувати світ швидко.
- **Надійність**: Генератор повинен гарантувати надійність кінечного результату, збіг бажаного та реального.
- **Різноманітність**: Створення максимально різноманітного, не схожого одне на одне контента, задля отримання унікального результату.
- **Правдивість**: Створений контент повинен підпорядковуватись певним законам та правилам.

1.3 Сфери використання процедурно згенерованих віртуальних світів

Основною метою використання процедурної генерації - це створення контенту без використання людських ресурсів (Моделлерів, дизайнерів тощо), розробка різних типів світів та ігор, покращення реіграбельності, адаптація ігор під гравця, покращення контенту за допомогою алгоритмічних рішень та формалізація гейм дизайну з наученої точки зору.

Перші задокументовані згадки використання процедурно згенерованих світів датуються 1980-ми роками. Тоді, при обмежених ресурсах пк, це дозволяло створювати різноманітні, великі ігрові світи. Характерним прикладом такої генерації слугує гра Rogue, яка надалі започаткувала жанр Roguelike, в якому основною фішкою є саме процедурно згенерований світ.

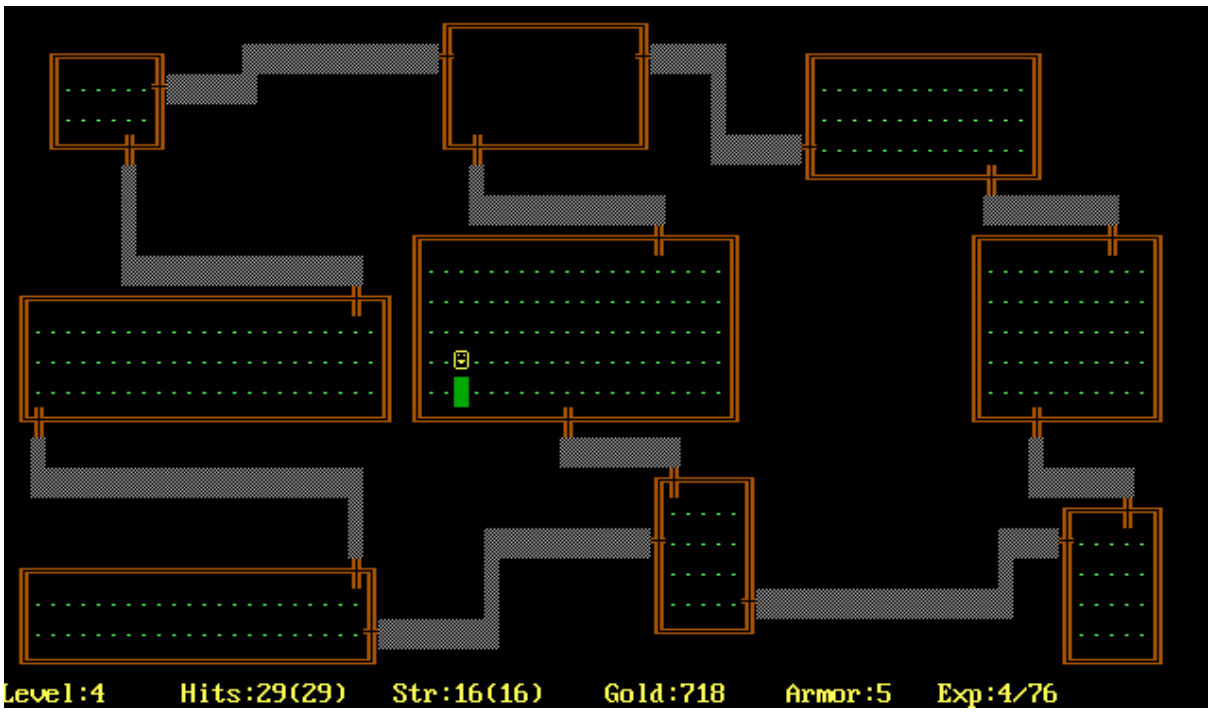


Рисунок 1.2 - Rogue

Найбільш поширена сфера використання процедурної генерації це, звичайно, ігрова індустрія, але вона ще використовується для створення різних детальних та реалістичних локацій для кіноіндустрії. Також процедурна генерація використовується і в настільних іграх. Так, у настільній рольовій грі D&D, майстер підземелля має можливість генерувати данжі та локації на льоту.

1.3 Висновок до розділу 1

У цьому розділі було розглянуто поняття віртуальних світів та процедурної генерації, та сфери їх використання. Було описано можливості та покращення які надають згенеровані світи та основні критерії для оцінки їх алгоритмів.

РОЗДІЛ 2. Алгоритми генерації 3Д світів

2.1 Генерація полігональних карт

2.1.1 Ідея та алгоритм

Алгоритм генерації полігональних карт дозволяє генерувати мапу світу з гарною береговою лінією, річками та горами. Сам алгоритм гарно генерує мапи великих і малих островів повністю оточених водою.

Першим кроком є створення полігонів. Найпростіший варіант це генерація гексагонів та випадкове переміщення їх вершин. Це генерує потрібну нам, хаотичну сітку. Іншим варіантом може бути генерація та поєднання випадкових точок.

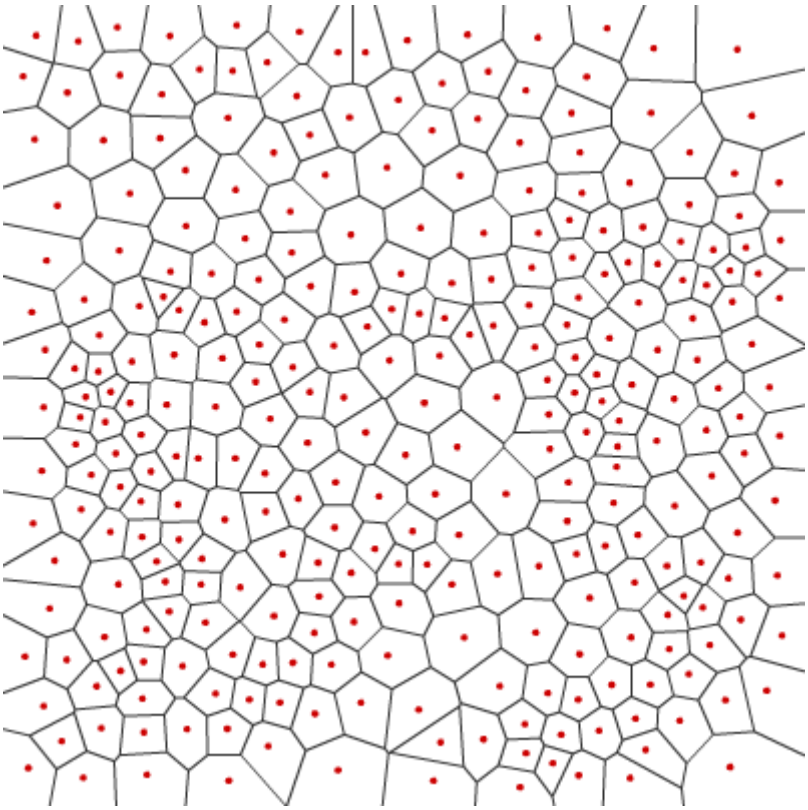


Рисунок 2.1 - Приклад генерації полігонів

Наступний крок - використати триангуляцію Делоне та діаграму Вороного для генерації точок та з'єднань між ними.

Далі, створення суші. В основі алгоритму лежить синусоїдальна функція для генерації меж острова. Далі кожному сегменту назначається тип: земля або вода.

Обрахуємо висоту як відстань від берегової лінії, та внесено деякі корективи задля того, щоб алгоритм не був передбачуваним. В цьому ж кроці, обрахуємо напрямки спаду висоти для подальшої генерації річок. Проведемо річки.

На основі річок та висот, можемо побудувати карту вологості, та надати ці значення нашим сегментам.

Оскільки маємо висоту та вологість кожного, можемо на їх основі вивести тип для кожного сегменту, для цього можемо побудувати просту таблицю перетину та вимог до біомів та використати її.

Зона висот	Зона вологості					
	6 (мокрий)	5	4	3	2	1 (сухий)
4 (високий)	SNOW			TUNDRA	BARE	SCORCHED
3	TAIGA		SHRUBLAND		TEMPERATE DESERT	
2	TEMPERATE RAIN FOREST	TEMPERATE DECIDUOUS FOREST		GRASSLAND		TEMPERATE DESERT
1 (низький)	TROPICAL RAIN FOREST		TROPICAL SEASONAL FOREST		GRASSLAND	SUBTROPICAL DESERT

Рисунок 2.2 - Таблиця біомів

Далі ми можемо надати нашій мапі шуму, або вирівняти краї, це все вже залежить від бажаного результату.

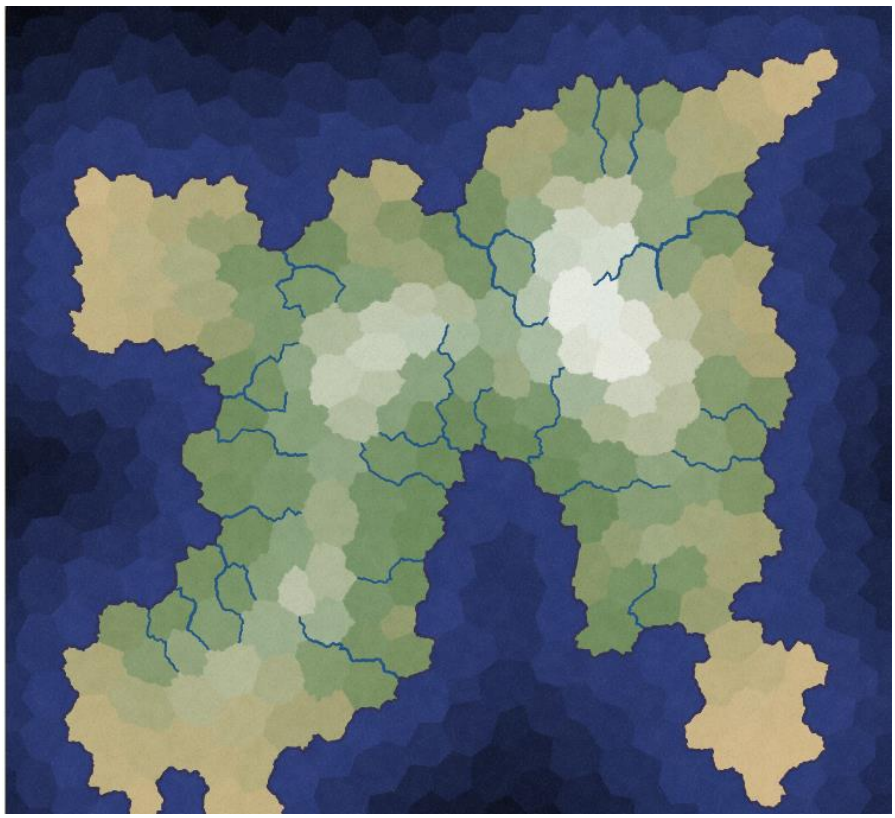


Рисунок 2.3 - Приклад мапи

2.2.2 Аналіз алгоритму

Насамперед алгоритм цікавий структурою побудови землі через геометричні фігури та він генерує досить реалістичні острови та континенти які можна легко використовувати вже за потреби у певних сферах. Його основні переваги в генерації красивої водної лінії, річок та правдивого розподілу біомів.

2.2 Згортання хвильової функції

Згортання хвильової функції - це спеціальний алгоритм процедурної генерації розроблений Максимом Гуміним. Основною метою алгоритму є створення світів на основі спадкових правил та патернів заданого зображення. Ці правила та патерни виводяться з зображення розміру NxM та будує схеми, які вказують як часто повна деталь з'являється у світі, та до якого з тайлів кріпиться.

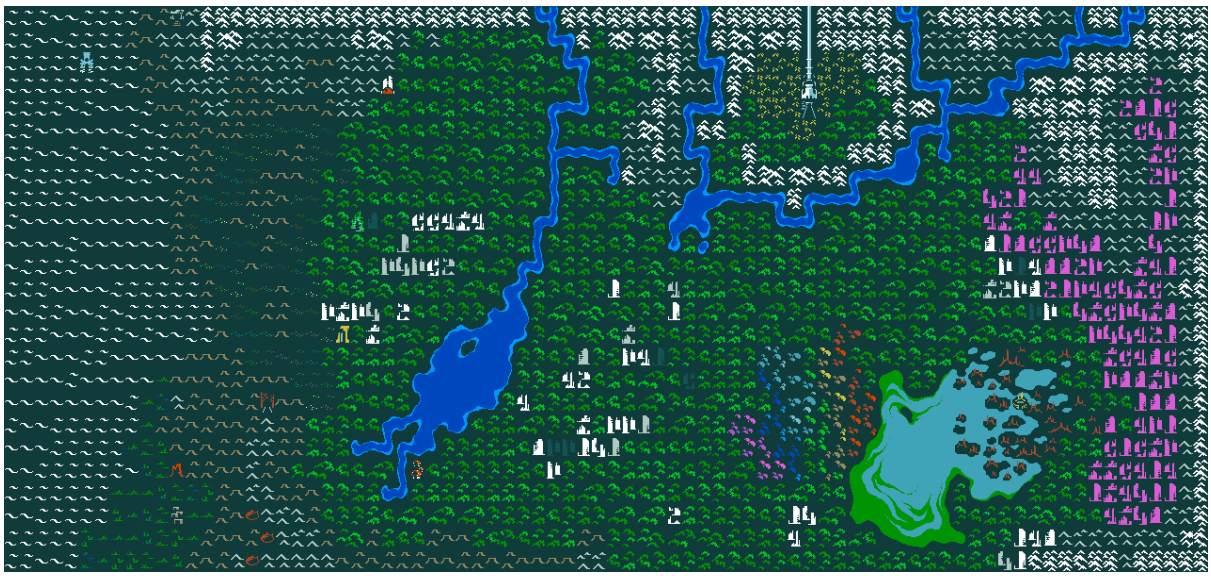


Рисунок 2.4 - Приклад роботи алгоритму згортання хвильової функції

Сам алгоритм може бути як суміжним, так й перекритим. А запускати його можна для 2Д і 3Д, для шестикутної або неправильної сітки.

Перший крок алгоритму - аналіз та патернування вхідного зображення, вивід правил, як кожен елемент може стикуватися до іншого. Далі алгоритм проходить через весь вихідний простір та починає виставляти тайли відповідно до правил, один біля одного. Це схоже на Згортання хвильової функції у квантовій фізиці, де хвильова функція (математична функція, що визначає кут свободи в якому об'єкт, за яким йде спостереження, може зайняти певний стан) напряму наближається та згортається у певний стан.

Після того, як алгоритм закінчує аналіз вхідного зображення, викликається функція, котра переводить отриману інформацію, в умову, для подальшої роботи алгоритму. З всієї інформації, що обробляє алгоритм, найважливішою частиною є - хвиля. Хвиля - це булевий масив, який містити всі можливі комбінації поєднання тайлів.

Останній крок - колапс хвилі. На вихідному просторі, вибирається випадкова точка, де буде побудований перший, випадковий тайл. А далі за правилами поєднання буде побудована решта світу.

Рисунок 2.5 - Робота Алгоритму ЗХФ у грі Bad Norse

2.3 Шум Перлина

Шум Перлина - метод генерування Наборів структурованих, випадкових чисел. Але вихідний результат це не просто набір випадкових чисел, це узгоджені між собою значення. Шум Перлина генерує випадкові числа, які слідують за плавним градієнтом. Отже, генератор у сусідніх точках дасть схожий результат. Такі числа можна генерувати в багатьох програмах: генерація хмар води та місцевості.

Такий алгоритм генерування градієнтів чисел, то його можна використовувати для генерації карти висот. Така карта, буде в градації сірого, і світлі її ділянки будуть наші гори, а темні - ущелини. Хоча така карта має все гарний вигляд, але цього не досить, до реальної карти висот ще далеко. Для цього ми можемо програмно змінити порогове значення та розмір карти. Це дасть нам бажаний результат для подальшої обробки.

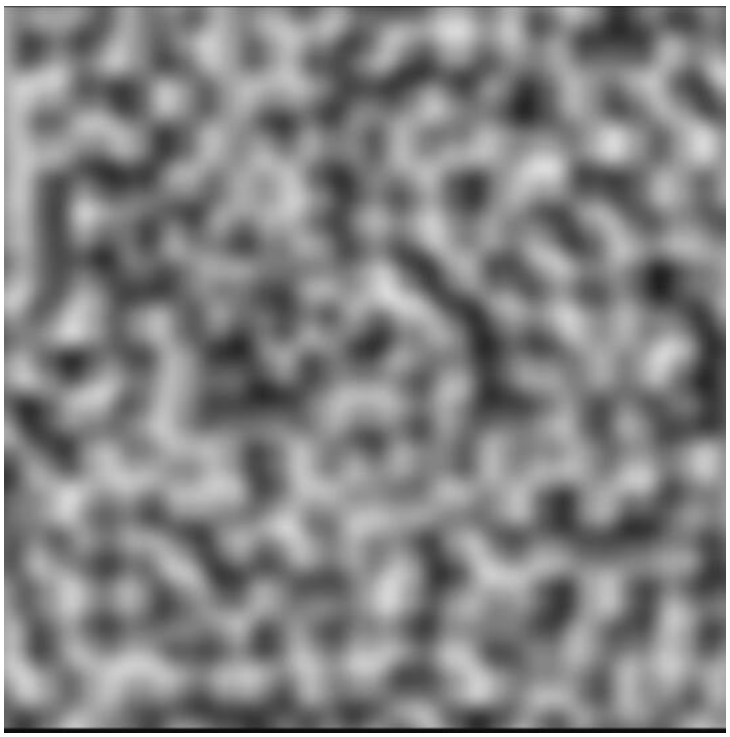


Рисунок 2.6 - Шум Перлина

Далі, перетворимо зображення в масив значень - висоти. Після чого, в нас буде інформація про висоту кожного ділянка нашої мапи. Додаємо воду, біоми, клімати та інші умови нашого світу в залежності від висоти та вхідних умов і результатом такої генерації буде 2 або 3Д світ готовий для подальшого використання.

Алгоритм генерує досить правдиві та різноманітні світи, а також досить легкий в використанні та в модифікації. Надає багато значень та аспектів для маніпуляції та підлаштовування під себе.

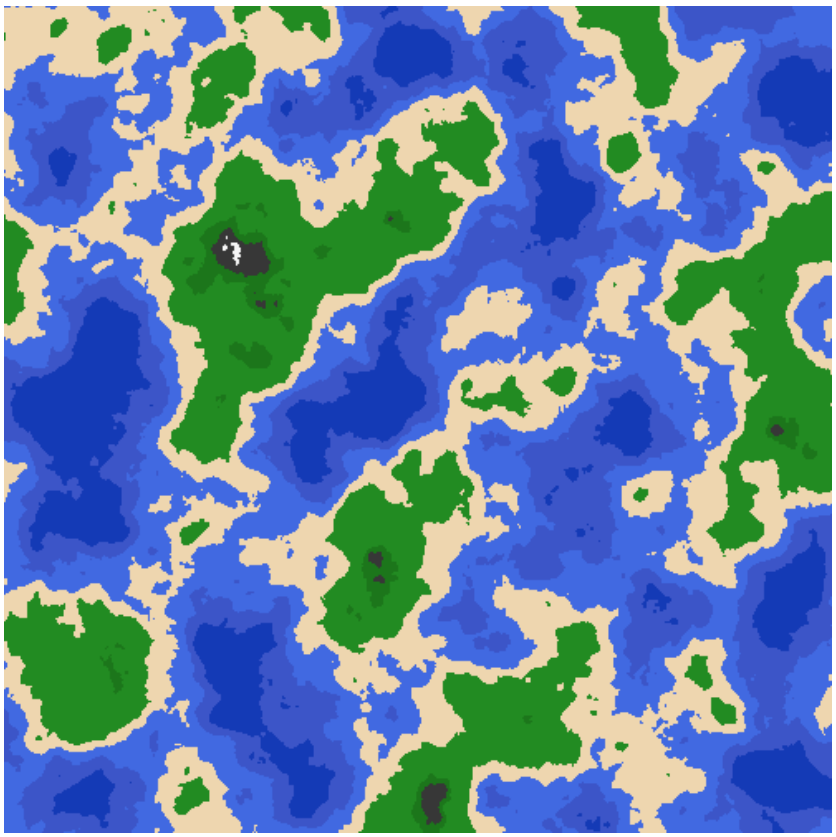


Рисунок 2.7 - Приклад генерації алгоритму

2.4 Клітинний автомат

Клітинний автомат - це дискретна математична модель, вперше використана в 1940 роках. І попри комплексні проблеми, що вирішуються за допомогою цього алгоритму, основна модель досить проста.

В основі, клітинний автомат складається з N комірок та їх правил переходу. Кожна клітинка сітки може перебувати в певному стані (вимкнена або увімкнена як приклад найпростіших станів). А початковий розподіл клітинок - база автомата.

Кожен крок алгоритму переводить всі клітинки одночасно в новий стан, за певними заданими правилами. А сусідство клітинки, визначає як і які клітинки впливають на її стан.

Двома найпоширенішими околицями є околиця Мура і Неймана.
Околиця Мура - квадрат, з восьми клітинок, що оточують дану.
Околиця Неймана - схожа на хрест, складається з 4 клітинок.

Добрим прикладом клітинному автомату слугує гра Життя.

Першим кроком по створенню автомата являється створення сітки та розміщення на ній клітинок. Після чого треба визначити можливі стани кожної клітинки.

Далі ми запускаємо сам алгоритм генерації тайлів. Він проходить в декілька ітерацій і на кожному кроці використовуючи задані правила, генерує та корегую тайли в вихідному просторі.

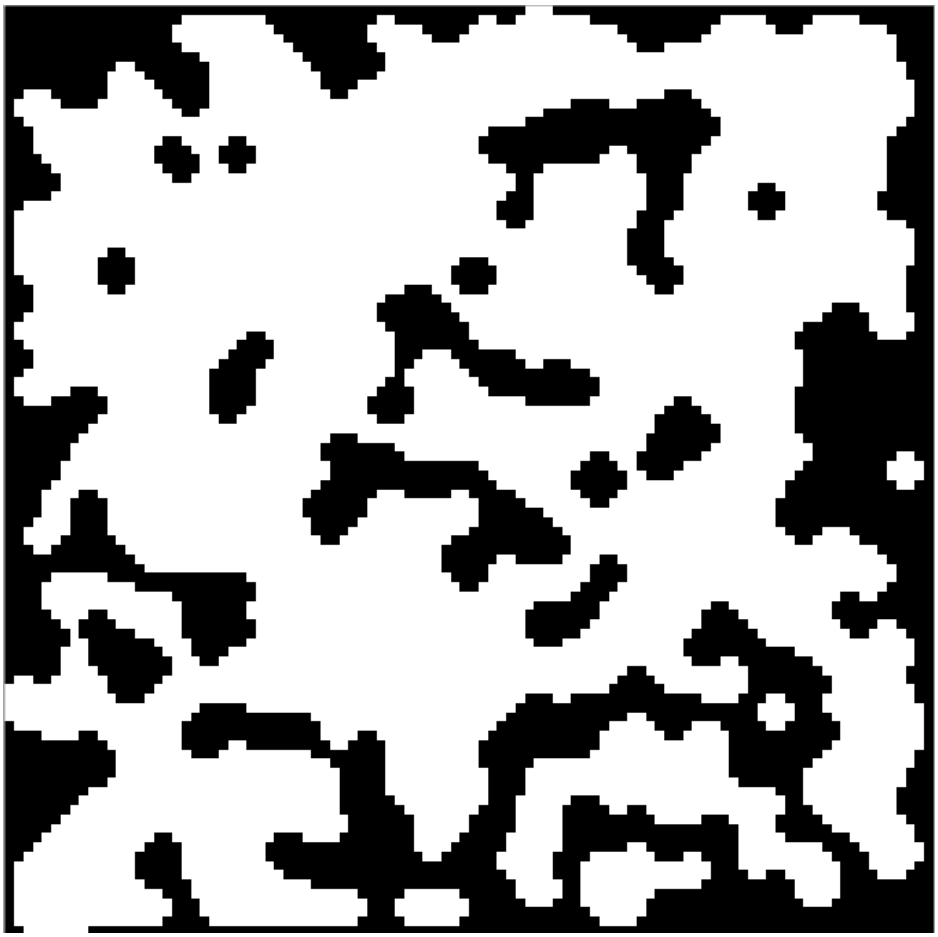


Рисунок 2.8 - Простий приклад роботи клітинному автомату

Наступним кроком буде перетворення 2Д мапи у 3Д простір, для чого можна побудувати плавний перехід висот та спадання низин, або закласти це в початковий автомат.

Алгоритм гарно працює з невеликими ділянками мапи, печерами або природними лабіринтами. Для розгортання генерації величезного 3Д простору треба буде робити багато додаткових маніпуляцій з вихідним результатом, але це цілком можливо.

2.4 Висновок до розділу 2

У цьому розділі було розглянуто основні методи генерування 3Д та 2Д світів, розібрані методи роботи кожного з алгоритмів та коротко підсумовано їх використання та можливості. Важливим критерієм кожного з них є тип вихідного результату та побажання користувача, бо кожний алгоритм дасть певний, унікальний тип ландшафту.

РОЗДІЛ 3. Реалізація алгоритму генерування 3Д світу на основі вхідних даних користувача

3.1 Створення програмному додатку

В якості додаткового, індивідуального завдання, я створив додаток для цієї курсової роботи. Цей додаток - це алгоритм генерації на основі шуму Перлина. Додаток був створений на мові C# з використанням двигуна Unity та різних 3Д ассетів з Unity Asset Store. В наявності й код і .exe файл для демонстрації роботи алгоритму.

У роботі програми використовується набір вхідних даних, та користувач може легко та повністю їх налаштувати або змінити. На рисунку 3.1 показаний графічний інтерфейс вікна налаштування програми-генератора.

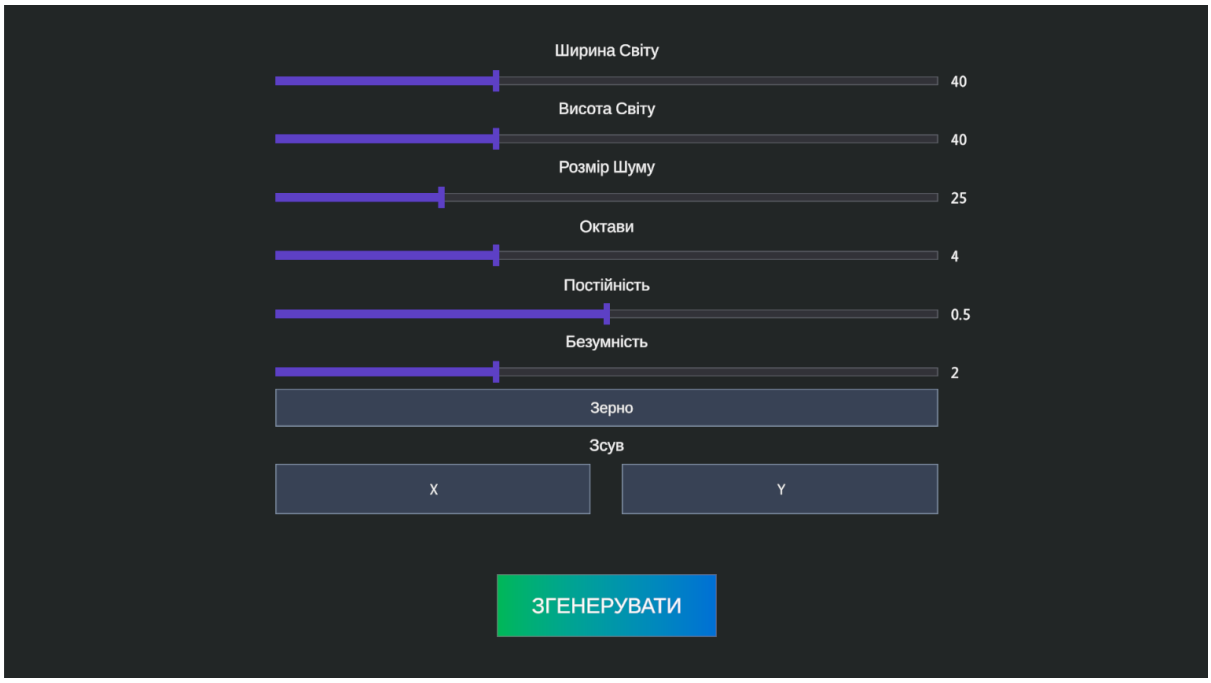


Рисунок 3.1 - Графічний інтерфейс програмного застосунку

Графічний інтерфейс застосунку, хоч й має в собі багато полів та параметрів, але досить простий у використанні та розумінні. Значення по

замовченню згенерують користувачу досить стандартний, красивий 3Д світ готовий до використання.

Параметри

Ширина світу - розмір світу в ширину.

Висота світу - розмір світу в висоту.

Розмір шуму - визначає розмір текстури шуму Перлина, що буде використана для генерації світу.

Октави - параметр для регулювання окремих елементів згенерованого ландшафту.

Постійність - наскільки високий поріг для вирішення типу місцевості.

Безумність - наскільки хаотично спотворення мапи шуму буде зроблено перед генерацією.

Зерно - зерно для генератора, дозволяє налаштувати рандом (При однаковому зерні та налаштуваннях, світ не буде змінюватись при генерації)

Зсув - зсув текстури шуму від центра (X:0 Y:0)

Кнопка “Згенерувати” запускає алгоритм, та через деякий час завантажує модель згенерованого 3Д світу, та надає можливість його роздивитись. Для керування використовується клавіші WASD та RF. Перші - керування камерою, другі - зум камери.

Рисунок 3.2 - Вікно очікування генерації

Рисунок 3.3 - Приклад згенерованого світу

3.2 Генерування мапи шуму

Для генерування мапи шуму, спочатку треба створити масив чисел - наша майбутня мапа висот, задати зерно для генератора та зробити масив зсувів для октав. Після чого, ми міняємо наш зсув згідно зі значеннями в масиві октав. Наступним кроком для кожного елемента масиву мапи шуму генерувати значення на основі вихідних даних та вбудованої у C# функції по генерації шуму Перлина.

```
for (var z = 0; z < height; ++z)
{
    for (var x = 0; x < width; ++x)
    {

        float amplitude = 1;
        float frequency = 1;
        float noiseHeight = 0;

        for (var i = 0; i < octaves; ++i)
        {
            var sampleX float = (x - halfWidth) / scale * frequency + octaveOffsets[i].x;
            var sampleZ float = (z - halfHeight) / scale * frequency + octaveOffsets[i].y;

            var perlinValue float = Mathf.PerlinNoise(sampleX, y sampleZ) * 2 - 1;
            noiseHeight += perlinValue * amplitude;

            amplitude *= persistance;
            frequency *= lacunarity;
        }

        if (noiseHeight > maxHeight)
        {
            maxHeight = noiseHeight;
        }
        else if(noiseHeight < minHeight)
        {
            minHeight = noiseHeight;
        }

        noiseMap[x, z] = noiseHeight;
    }
}
```

Рисунок 3.4 - Основна частина коду генерації шуму Перлина

3.3 Генерація мапи світу

Після отримання наші майбутньої мапи висот, ми можемо почати генерувати світ. Для генерації я створив певні пресети - тайли з ландшафтом і завантажив їх в генератор для подальшого використання. Ці тайли репрезентують Середні віки в Центральній Європі, але їх можна легко замінити на будь-які інші.

Сам алгоритм генерації досить простий. Ми проходимо по текстурі шуму Перлина і в залежності від значення градації сірого вибираємо потрібний нам тип та вид тайла.

```
public void GenerateMap()
{
    ClearMap();

    var noiseMap:float[,] = Noise.GenerateNoiseMap(width, height, seed, noiseScale, octaves, persistance,
        lacunarity, offset);

    for (var x = 0; x < width; ++x)
    {
        for (var z = 0; z < height; ++z)
        {
            var currentHeight:float = noiseMap[x, z];
            for (var i = 0; i < biome.Tiles.Count; ++i)
            {
                if (!(currentHeight <= biome.Tiles[i].Height)) continue;
                var t:GameObject = Instantiate(original:biome.GetRandomTile(i),
                    position:new Vector3(
                        x:x * biome.Tiles[i].Tiles[0].Prefab.transform.localScale.x * biome.TileScale, y:0,
                        z:z * biome.Tiles[i].Tiles[0].Prefab.transform.localScale.z * biome.TileScale),
                    Quaternion.identity, transform);
                var rand:int = Random.Range(0, 5);
                t.transform.rotation = Quaternion.Euler(Vector3.up * 90 * rand);
                _tiles.Add(t);
                break;
            }
        }
    }
}
```

Рисунок 3.5 - Метод для генерування світу

3.4 Отримання результату

Після основних зазначених вгорі кроків ми отримаємо готовий для використання світ на основі шуму Перлина та вхідних даних користувача.

3.5 Висновок до розділу 3

У цьому розділі було розглянуто та реалізовано програму, що будує 3Д світ за допомогою генерації шуму Перлина та подальшого використання його як мапи висот для спавна тайлів.

ВИСНОВОК

Існують різні алгоритми генерації 3Д світів, кожен має свої переваги та недоліки. Щоб підібрати потрібний, треба розуміти свої потреби та вимоги та провести аналіз всіх доступних та популярних алгоритмів генерації. Важливим фактором є значення, що передаються як параметри в алгоритм генерації, бо при різних значеннях, один і той самий алгоритм видає кардинально різні результати.

Були розглянуті найбільш поширені алгоритми генерації 3Д та 2Д світів, проведений їх аналіз та виявлені випадки для їх використання.

Генерація полігональних карт - гарний алгоритм для генерування кліматів, висот для островів та океанічних структур. Хоча сам алгоритм досить складний в реалізації та розумінні.

Згортання хвильової функції - досить новий та молодий алгоритм, який дає можливість на основі згортання хвилі генерувати правдиві мапи невеликих площ світу або ландшафту.

Шум Перлина - напевне найкращий алгоритм для генерації саме величезних світів: континент, океани, гори та моря. Має великий перелік параметрів то модифікацій для гнучкого підлаштування алгоритму під потреби користувача.

Клітинний автомат - гарний алгоритм для генерації печер, відрізків рівнинних ділянок та болт, хоча на цьому його можливості не закінчується. Один з найперших алгоритмів генерації.

В майбутньому варто додати інші алгоритм генерації світів, різні біоми та форми ландшафту. Та можливо додавати експорт світу як окрему 3Д модель**.**

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Процедурна генерація світу з використанням клітинному автомату [Електронний ресурс] / Режим доступу до ресурсу: https://www.raywenderlich.com/2425-procedural-level-generation-in-games-using-a-cellular-automaton-part-1
2. Генерація світу за допомогою шуму перлина [Електронний ресурс] / Режим доступу до ресурсу: https://www.redblobgames.com/maps/terrain-from-noise/
3. Модифікація згортання хвильової функції [Електронний ресурс] / Режим доступу до ресурсу: https://digitalcommons.trinity.edu/cgi/viewcontent.cgi?article=1058&context=compsci_honors
4. Полігональний генератор світів для ігор [Електронний ресурс] / Режим доступу до ресурсу: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/
5. Noor Shaker, Julian Togelius, and Mark J. Nelson Procedural Content Generation in Games [Електронний ресурс] / Режим доступу до ресурсу: http://pcgbook.com/
6. Процедурна генерація рівнів [Електронний ресурс] / Режим доступу до ресурсу: https://habr.com/ru/post/418685/
7. Процедурна генерація [Електронний ресурс] / Режим доступу до ресурсу: https://dtf.ru/gamedev/169117-kak-procedurnaya-generaciya-pomogaet-sozdavat-otkrytye-miry

ВИКОРИСТАНІ АССЕТИ

(Моделі, бібліотеки тощо)

1. Low Poly Ultimate Pack - https://assetstore.unity.com/packages/3d/props/low-poly-ultimate-pack-54733#description
2. Modern UI Pack - https://assetstore.unity.com/packages/tools/gui/modern-ui-pack-201717#description