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

ВИКОРИСТАННЯ ШУМІВ ПЕРЛІНА ДЛЯ ГЕНЕРАЦІЇ ІГРОВОГО ПОЛЯ ПРИ РОЗРОБЦІ ІГОР

Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
Корнійчук М. А.__
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2023 р.

Виконав студент _____
___Перун Є.Т._____
(прізвище та ініціали)
“____” _____ 2023 р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,

доцент, к.ф-м.н.

_____О. П. Жежерун (підпис)

„____”_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Перун Єлизаветі Тарасівні факультету інформатики 4-го курсу

ТЕМА Використання шумів Перліна для генерації ігрового поля при розробці ігор

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1 Процедурна генерація в іграх

2 Шум Перліна та його використання в іграх

3 Інструменти та методи використані у курсовій роботі

Висновки

Список прийнятих скорочень

Список джерел

Дата видачі „___” _____ 2023 р. Корнійчук М. А. _____ (підпис)

Завдання отримав _____ (підпис)

Тема: Використання шумів Перліна для генерації ігрового поля при розробці ігор

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проект (роботи)	Термін виконанн етапу	Примітка
1.	Отримання завдання на курсов роботу	01.10.2022	
2.	Аналіз матеріалів за темою	01.11.2022	
3.	Розробка та програмуванн алгоритму	01.12.2022	
4.	Написання текстової частини д курсової роботи	01.02.2023	
5.	Коригування виконаної роботи	01.04.2023	
6.	Створення слайдів для доповід та написання доповіді.	01.05.2023	
7.	Остаточне оформлення роботи т слайдів	21.05.2023	
8.	Захист курсової роботи	31.05.2023	

Перун Є. Т. _____

Корнійчук М. А. _____

“_____” _____

Зміст

Анотація

Робота присвячена розробці інструмента для процедурної генерації ігрової карти для відеоігор за допомогою шуму Перліна.

Ключові слова: шум Перліна, генерація ігрового світу, процедурна генерація

Вступ

В даній кваліфікаційній роботі була поставлена задача розробки утиліти для генерації ігрового світу, яка допоможе прискорити процес створення різноманітних відеоігор, що потребують тривимірної реалістичної ігрової карти. Головною метою є розробка ефективного і гнучкого інструменту, який можна використовувати для створення різних типів ігрових світів з різними характеристиками та стилями.

Перший розділ кваліфікаційної роботи присвячений огляду процедурної генерації в відеоіграх. У цьому розділі буде розглянуто основні принципи процедурної генерації, її переваги та застосування в ігровій індустрії. Також будуть проаналізовані приклади відомих ігор, які використовують процедурну генерацію для створення своїх ігрових світів.

Другий розділ кваліфікаційної роботи присвячено шуму Перліна та одному з методів його отримання. Шум Перліна є одним з основних інструментів у процедурній генерації. У цьому розділі будуть розглянуті принципи роботи шуму Перліна, його модель та основні характеристики.

Третій розділ містить детальну інформацію про методи, алгоритми та інструменти, що використовуються в дипломній роботі. Також будуть представлені інструменти, які були використані для реалізації утиліти, включаючи фреймворк Unity для розробки.

Основна частина

Розділ 1: Процедурна генерація в іграх

1.1 Огляд

Процедурна генерація – метод створення асетів до відеоігор за допомогою поєднання вручну створених елементів, алгоритмів та випадкових даних. Використовується на противагу ручному створенню мап, текстур, 3D-моделей тощо.

Своєрідна процедурна генерація використовувалася в настільних рольових іграх ще до появи комп'ютерних ігор. В настільній грі "Dungeons&Dragons" правила дозволяли ведучому створювати мапи підземелля за допомогою випадкових чисел, що випадають на кубику, та заздалегідь створених окремих кімнат підземелля.

Раннім прикладом процедурної генерації саме в комп'ютерних іграх є процес створення ігрової мапи в грі "Rogue"(1980) – вона процедурно генерувалася з ASCII символів[1].

Інший приклад використання – гра "Elite"(1984). Розробники спершу планували закласти в неї 2^{48} процедурно згенерованих галактик для дослідження гравцем. Однак згодом кількість доступних галактик зменшили до 8, адже видавець вважав, що таке неприродньо велике число доступних галактик може відвернути гравців від гри[2].

Гра "Rescue on Fractalus"(1985) використовувала фрактали аби в реальному часі процедурно генерувати гори.

Один із визначних сучасних прикладів – гра "No Man's Sky". Вважається найбільшою грою в історії, адже містить 18 квінтільйонів процедурно згенерованих планет до дослідження, при цьому для кожного гравця всі планети однакові, адже для генерації використовується один сід (seed) для випадкового числа[3].

1.2 Методи генерації та застосування

Як було зазначено вище, одним із методів процедурної генерації є використання фракталів. Геометричні фрактали використовуються для генерації природніх структур – дерев, кущів тощо. Алгебраїчні та стохастичні фрактали

гарно підходять для моделювання гір, а також того, що стосується механіки рідин та газів – поверхні води чи полум’я[4].

Ланцюг Маркова може використовуватися для генерації природніх явищ чи генерації ігрової мапи, так само як і шум Перліна чи симплексний шум. Також у генерації геометрії ігрового рівня (мапи) можуть використовуватися випадкові числа, діаграми Вороного, алгоритм Diamond-Square тощо.

Процедурно генерувати можна також текстури, різноманітні 3D-моделі, звук (мовлення та музику).

Розділ 2: Шум Перліна та його використання в іграх

2.1 Шум Перліна

В той час як звичайний шум є набором псевдовипадкових значень від 0 до 1, шум Перліна є градієнтним шумом, тобто набором псевдовипадкових одиничних векторів у просторі та інтерпольованих функцією згладжування між ними.

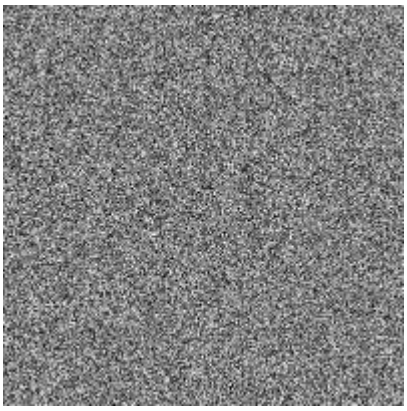

Рисунок 1 Звичайний шум

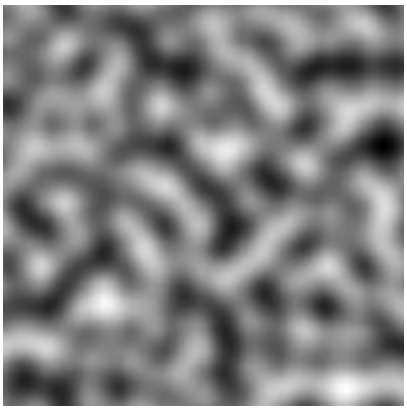

Рисунок 2 Шум Перліна

Шум Перліна було розроблено Кеном Перліном в результаті його розчарування надто машинним та піксельним виглядом тогочасної комп’ютерної графіки[5].

Розглянемо що таке функція шума в загальному. В одновимірному варіанті функції, якщо уявити собі графік, що складається з нескінченної осі Х та осі У, що сягає одиниці, та складається з випадкових точок, що інтерпольовані між собою, ми отримуємо функцію шуму[6].

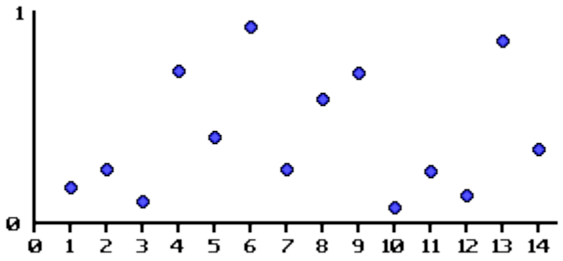

Рисунок 3 Випадкові точки на графіку

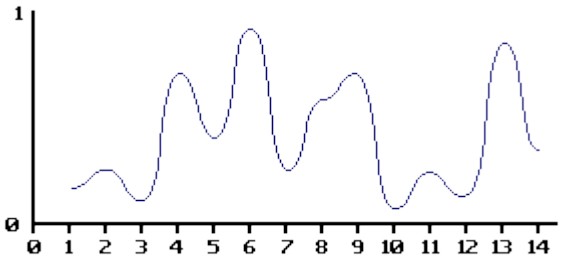

Рисунок 4 Інтерпольовані точки

Важливо також розуміти такі поняття, як амплітуда, частота.

Візьмемо за приклад функцію синуса. Відстанню між хвилями називають відстань від одного піку функції до іншої. Тоді висота однієї хвилі й буде амплітудою, частотою ж буде значення обернене до відстані між хвилями.

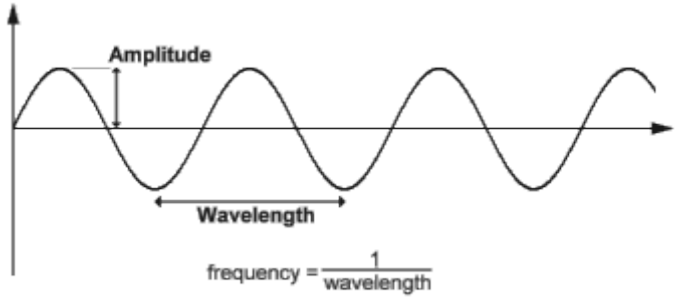

Рисунок 5 Функція синуса

Для функції шуму амплітуда та частота матимуть такий вигляд: червоними точками на графіку позначено обрані випадкові значення між якими здійснювалася інтерполяція, тож у цьому випадку амплітудою буде відстань між мінімальним та максимальним значенням, що набуває графік, а відстанню між хвилями буде відстань між двома сусідніми червоними точками. Частота ж залишається значенням оберненим до відстані між хвилями.

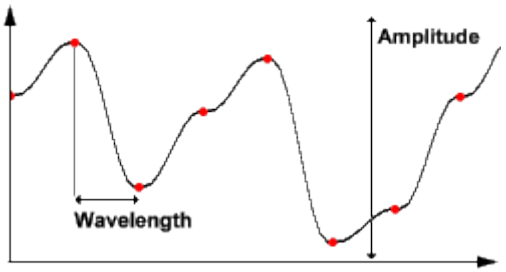

Рисунок 6 Амплітуда та частота функції хвилі

Тепер, якщо взяти кілька різних функцій шуму та додати їх одна до одної, отримуємо функцію шума Перліна.

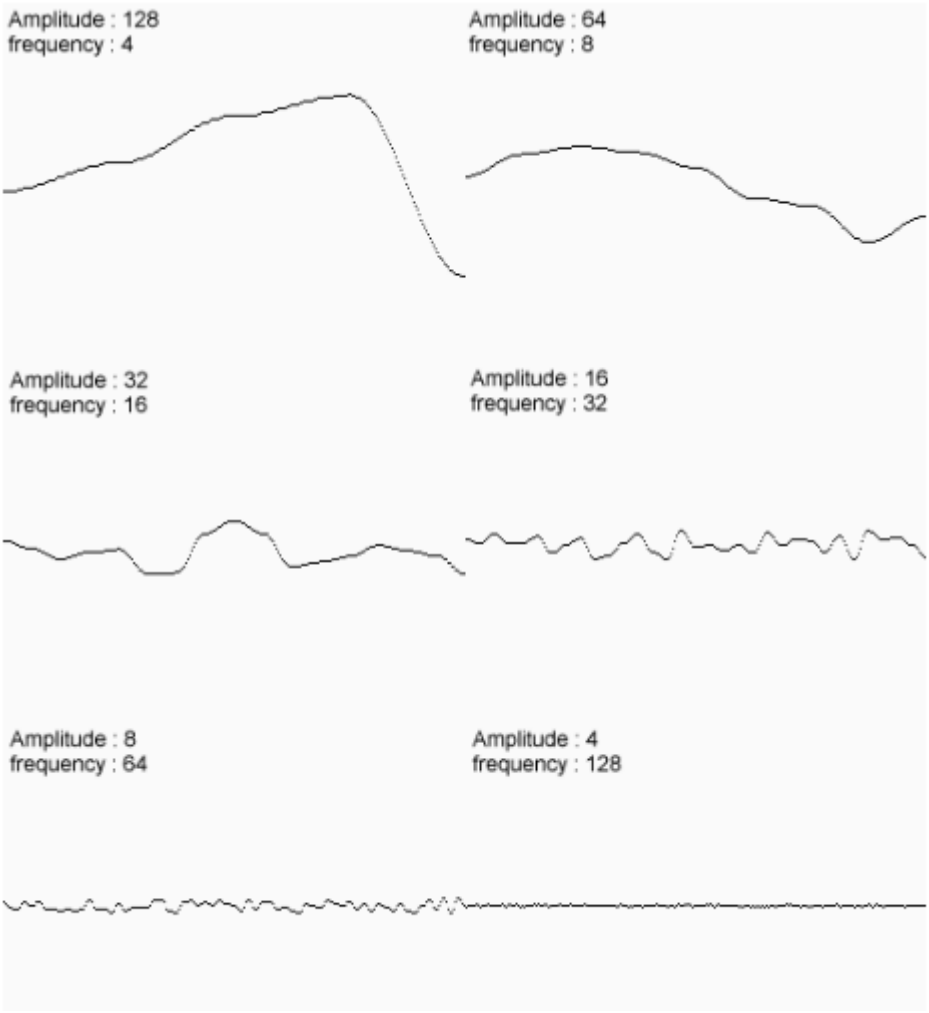

Рисунок 7 Різні функції шуму

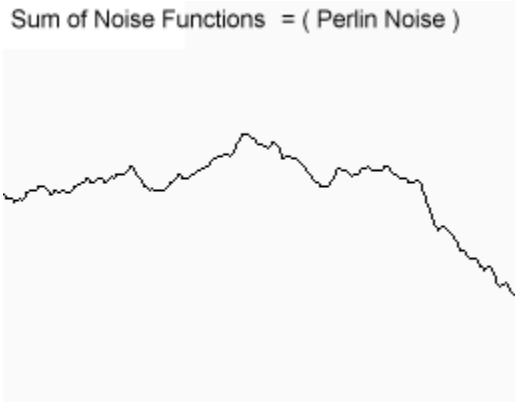

Рисунок 8 Результуючий шум Перліна

Наступним важливим терміном є стійкість шуму Перліна.

В даному випадку для кожної доданої функції використовувалося подвоєне значення амплітуди та половина значення частоти. Але це не єдина можлива конфігурація. Отже, стійкістю шуму Перліна є число, що визначає кожну амплітуду:

Амплітуда = стійкість^{i}, де i – номер кожної доданої функції.

Лакунарністю ж називають число, що визначає частоту кожної наступної функції.

Частота = лакунарність^{i}, де i – номер кожної доданої функції.

Наступна діаграма відображає вплив значення стійкості на кожну наступну додану функцію – чим більше значення стійкості, тим більше кожна наступна додана функція впливає на результуючу функцію шуму Перліна. В даному прикладі лакунарність дорівнює 2.

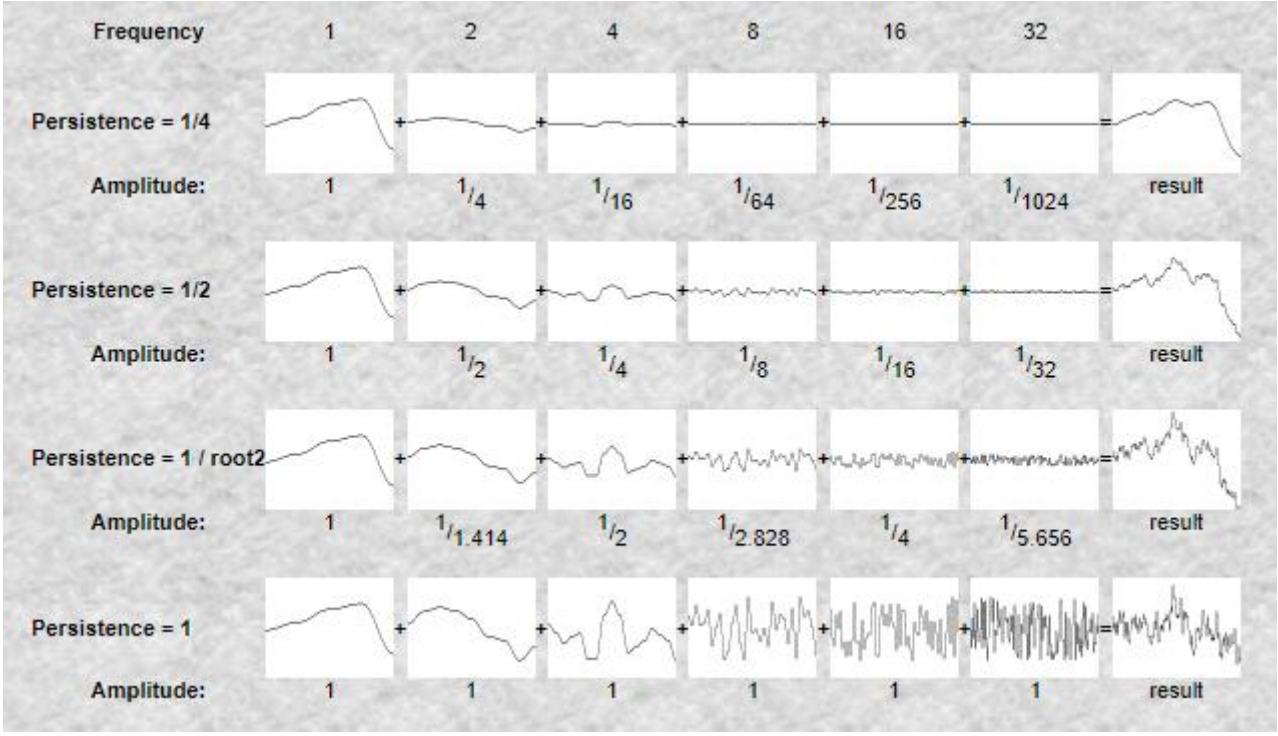

Рисунок 9 Залежність доданих функцій від стійкості

Кожна така наступна додана функція називається октавою – кожна матиме частоту вдвічі більшу за попередню.

Отже, якщо проводити аналогію з горами, можна розглядати першу функцію що додається, як функцію, що визначає обрис гори. Наступна функція визначатиме велике каміння, ще наступна – каміння меншого розміру. Таким чином, кожна наступна функція матиме все менш значний вплив на результуючу гору, однак змінюючи значення лакунарності та стійкості можна досягнути ефекту, коли навіть остання додана функція значно змінюватиме обрис гори.

2.2 Застосування

Одновимірний шум Перліна можна використовувати для створення айдлової анімації персонажа в грі – значення шуму можна використовувати для корекції позиції вузла скелета персонажа в момент часу.

Двовимірний шум Перліна використовується для генерації хмар та двовимірних текстур, що накладаються на об’єкти, а також для створення ландшафту у гру, що й висвітлено в цій дипломній роботі.

Тривимірний шум використовується для генерації об’ємних хмар, їх анімації, а також для тривимірних текстур.

Чотиривимірний шум (де четвертим виміром є час) використовується для анімацій.

Розділ 3: Інструменти та методи використані у курсовій роботі

3.1 Ігровий рушій Unity

В роботі над кваліфікаційної роботою було обрано ігровий рушій Unity.

Unity - це популярний ігровий рушій, розроблений компанією Unity Technologies. Він використовується для створення різноманітних ігор, а також інтерактивних додатків і симуляцій.

Основні характеристики Unity включають:

1. Кросплатформність: Unity підтримує багато платформ, включаючи Windows, macOS, Linux, Android, iOS, Xbox, PlayStation і багато інших. Це дозволяє розробникам створювати ігри для різних пристроїв з використанням одного і того ж коду.

2. Візуальний редактор: Unity має потужний візуальний редактор, який дозволяє розробникам створювати ігрові об'єкти, налаштовувати їх властивості, створювати сцени та взаємодіяти зі зображеннями, звуками та іншими ресурсами гри без необхідності писати код.

3. Скриптованість: Unity використовує скриптову мову C#, що дозволяє розробникам контролювати поведінку об'єктів у грі. C# є потужною та простою у використанні мовою програмування, що робить Unity доступним для розробників з різним рівнем досвіду.

4. Розширюваність: Unity надає можливість розширювати функціональність рушія за допомогою додатків (плагінів). Існує широкий спектр додатків, які дозволяють розширити можливості Unity у різних областях, таких як фізика, штучний інтелект, графіка тощо.

3.2 Структура застосунку

Застосунок працює за рахунок семи скриптів [7][8], а саме:

1. EndlessTerain - виконує ряд функцій для створення безкінечного рельєфу у грі. Основна ідея полягає у тому, що рельєф розбивається на невеликі частини (чанки), які завантажуються та відображаються в залежності від позиції гравця у грі.

2. FaloffGenerator - містить метод GenerateFalloffMap. Цей метод використовується для генерації карти спаду (falloff map) для ефектів, які використовують зміну значень на основі відстані від центру. В наведеному в кваліфікаційній роботі застосунку дозволяє генерувати острови на мапі.

3. MapDisplay - використовується для відображення текстур і мешів на об'єкті у сцені Unity.

4. MapGenerator - використовується для генерації карти та керування процесом відображення карти у сцені Unity.

5. MeshGenerator - включає два класи: MeshGenerator і MeshData. Клас MeshGenerator містить статичний метод GenerateTerrainMesh, який використовується для генерації мешу ландшафту на основі заданої карти висот. Клас MeshData представляє дані мешу, включаючи вершини, трикутники та текстурні координати.

6. Noise – генерує шум Перліна.

7. TextureGenerator - містить два методи для генерації текстур на основі кольорової мапи та карти висот.

8. CameraController – керування камерою під час гри.

3.3 Огляд застосунку

Застосунок представляє собою редактор карти всередині інтерфейсу Unity. Розробнику дається доступ до таких параметрів мапи:

Режим відображення мапи – чи мапа відображається у вигляді рельєфної мапи, пласкої кольорової, мапи спаду чи шуму Перліна.

Режим нормалізації – визначає чи буде мапа безшовною.

Режим деталізації на превью – наскільки деталізована мапа генерується на превью.

Розмір шуму Перліна.

Октави шуму Перліна.

Стійкість шуму Перліна.

Лакунарність шуму Перліна .

Зерно випадкового числа.

Зміщення по координатами мапи шуму.

Чи використовувати мапу спаду.

Модифікатор висоти найвищих точок мапи.

Крива висот мапи.

Чи оновлювати мапу автоматично на превью.

Редактор регіонів мапи – океану, моря, суші тощо.

Кнопка, що запускає генерацію.

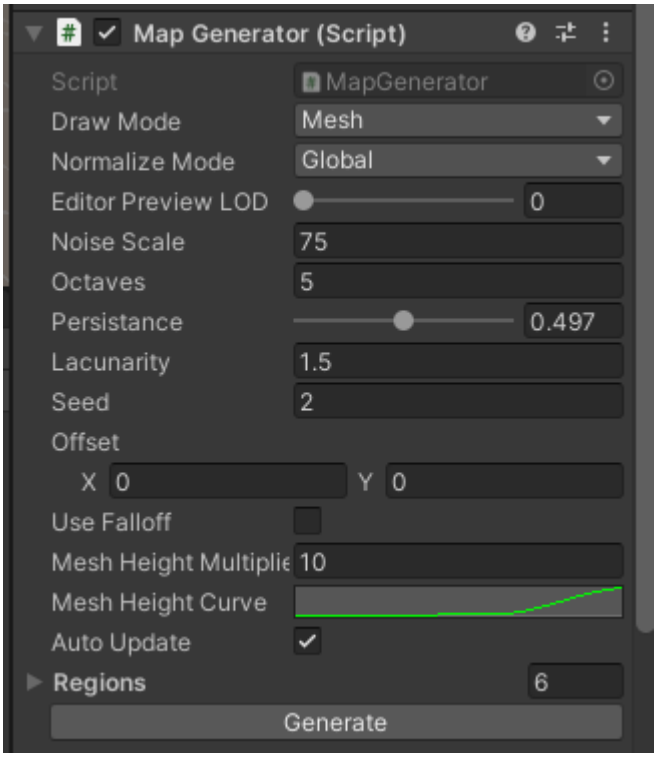

Рисунок 10 Інтерфейс застосунку

В застосунку є два режими роботи – генерація мапи для превью з можливістю її подальшого використання в грі та генерація мапи безпосередньо під час гри в реальному часі.

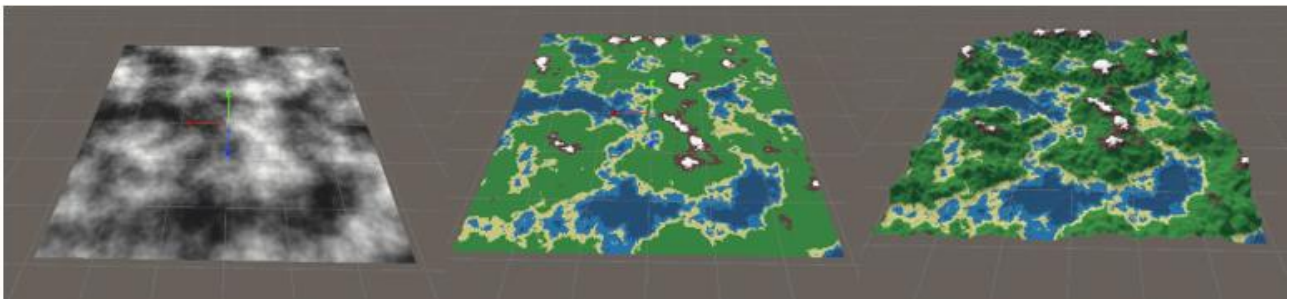

Рисунок 11 Згенерована мапа в різних режимах превью

Висновки

Отримані результати свідчать про успішну реалізацію поставленої мети - розробку утиліти для генерації ігрового світу з використанням процедурної генерації та шуму Перліна. Реалізована генерація мапи з використанням шуму Перліна дозволяє отримувати різноманітні ігрові світи з реалістичними деталями та рівнем деталізації.

Процес розробки включав огляд процедурної генерації в відеоіграх, вивчення принципів шуму Перліна та розробку алгоритмів інтеграції цього шуму для створення реалістичних ігрових мап. Були використані сучасні інструменти, зокрема фреймворк Unity, що дозволило забезпечити ефективну і реалістичну генерацію мап.

Отримана утиліта має потенціал використовуватися в ігровій індустрії для створення різноманітних ігрових світів, що відповідають потребам розробників ігор. Вона дозволяє швидше створювати реалістичні ігрові мапи з різними характеристиками та стилями, що сприяє покращенню якості ігрового досвіду для користувачів.

Однак, дослідження та реалізація процедурної генерації мапи є лише початком ширшого обсягу роботи в галузі розробки ігор. Є можливість подальшого розширення функціональності утиліти, вдосконалення алгоритмів генерації та впровадження додаткових функцій, що дозволять розробникам ігор створювати ще більш реалістичні ігрові світи.

Список прийнятих скорочень

3D – тривимірна розмірність простору, від англійського “3-dimensional”

C# – об’єктно-орієнтована мова програмування для платформи .NET від компанії Microsoft.

Асет – той чи інший об’єкт, що використовується в грі, ним може бути тривимірна модель, звук, скрипт тощо.

LOD – від англійського “Level of detail”. Рівень деталізації того чи іншого асету.

Список джерел

[1] - http://pc.gamespy.com/pc/ftl-faster-than-light/1227287p1.html

[2] - https://www.theguardian.com/books/2003/oct/18/features.weekend

[3] - https://business.blogthinkbig.com/realistic-worlds-procedural-generation-artificial-intelligence-video-games/

[4] - https://www.dstu.dp.ua/Portal/Data/74/66/7st-0-2.pdf

[5] - https://web.archive.org/web/20160308055501/http://noisemachine.com/talk1/4.html

[6] - https://www.arendpeter.com/Perlin_Noise.html

[7] - https://www.youtube.com/playlist?list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3

[8] - https://github.com/SebLague/Procedural-Landmass-Generation