НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: “Процедурна генерація 3D геометрії ігрового контенту”

Виконав: студент IV курсу, групи КВ-52
(шифр групи)

Пасічник Максим Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асистент каф. СПіСКС, Коляда К.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПіСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент ст. викладач каф. ОТ, Виноградов Ю.М. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
Пасічника Максима Юрійовича
(прізвище, ім'я, по батькові)

1. Тема проекту "Процедурна генерація 3D геометрії ігрового контенту", керівник проекту Коляда Костянтин Вячеславович, асистент каф. СПіСКС, затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту «14» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - Аналіз існуючих рішень та обґрунтування теми дипломного проекту;
   - Опис процедурної генерації;
   - Реалізація процедурної генерації ландшафту.

5. Перелік графічного матеріалу:
   - Взаємодія модулів. Схема структурна;
   - Основний потік програми. Схема алгоритму;
   - Потік для оновлення чанків та створення завдань. Схема алгоритму;
   - Пул потоків для виконання завдань. Схема алгоритму;
   - Процедура обрахування шуму Перлина. Схема алгоритму;
   - Презентація.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Клятченко Я.М., доц. каф. СПіСКС, к.т.н		

7. Дата видачі завдання «27» листопада 2018

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	19.11.2018	
2.	Розроблення та узгодження технічного завдання	27.11.2018	
3.	Аналіз існуючих рішень	20.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	09.01.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	05.03.2019	
6.	Розроблення програмного забезпечення	25.04.2019	
7.	Відлагодження та рефакторинг програмного продукту	06.05.2019	
8.	Підготовка матеріалів третього розділу дипломного проекту	17.05.2019	
9.	Підготовка графічної частини дипломного проекту	20.05.2019	
10.	Оформлення документації дипломного проекту	28.05.2019	

Студент _____ М.Ю. Пасічник
(підпис) (ініціали, прізвище)

Керівник проекту _____ К.В. Коляда
(підпис) (ініціали, прізвище)

Анотація

Об’єкт розробки - аналіз існуючих алгоритмів процедурної генерації, вдосконалення та реалізація.

Метою даної роботи є створення комп’ютерної системи для побудови тривимірної геометрії об'єктів з використанням процедурної генерації.

Комп’ютерна система дозволяє: здійснювати ввід вхідних даних, отримувати результат на основі вхідних даних. В процесі розробки було використано бібліотеку GLFW для створення програмного вікна та забезпечення взаємодії з пристроями вводу, математичну бібліотеку GLM яка дає можливість використовувати дані для OpenGL, бібліотеку ImGui для реалізації графічного інтерфейсу.

Вході розробки було виконано наступні задачі:

- проаналізувати існуючі програмні засоби, що дозволяють створювати тривимірну геометрію;
- проаналізувати існуючі алгоритми процедурної генерації;
- розробити програмне забезпечення генерації тривимірної геометрії.

Ключові слова:

Комп’ютерна система, програмне забезпечення, процедурна генерація, тривимірна геометрія, GLFW, GLM, OpenGL, ImGui.

ABSTRACT

The object of development - an analysis of existing algorithms of procedural generation, improvement and implementation.

The purpose of this work is to create a computer system for building three-dimensional geometry of objects using procedural generation.

The computer system allows: to input the data, to receive the result on the basis of input data. In the development process was used the GLFW library to create a software window and provide interoperability with input devices, the GLM mathematical library, which enables the use of data for OpenGL, the ImGui library for implementing the graphical interface.

During the development, the following tasks were performed:

- analyze existing software tools that allow to create three-dimensional geometry;
- analyze existing algorithms of procedural generation;
- develop software for the generation of three-dimensional geometry.

Keywords:

Computer system, software, procedural generation, 3D geometry, GLFW, GLM, OpenGL, ImGui.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЦЛ.045480.002 ТЗ	Процедурна генерація 3D	4		
			геометрії ігрового			
			контенту.			
			Технічне завдання.			
	А4	ІАЦЛ. 045480.003 ТП	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Відомість технічного			
			проекту.			
	А4	ІАЦЛ. 045480.004 ПЗ	Процедурна генерація 3D	52		
			геометрії ігрового			
			контенту.			
			Пояснювальна записка.			
	А4	ІАЦЛ. 045480.005 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Взаємодія модулів.			
			Схема структурна.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЦЛ.045480.001 ОА		
Розробив		Пасічник М.Ю.			Процедурна генерація 3D геометрії ігрового контенту Опис альбому	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Коляда К.В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ, КВ-52	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЦЛ. 045480.006 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Основний потік програми.			
			Схема алгоритму.			
	А4	ІАЦЛ. 045480.007 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Потік для оновлення чанків			
			та створення завдань.			
			Схема алгоритму.			
	А4	ІАЦЛ. 045480.008 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Пул потоків для виконання			
			завдань.			
			Схема алгоритму.			
	А4	ІАЦЛ. 045480.009 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Процедура обрахування			
			шуму Перлина.			
			Схема алгоритму.			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Процедурна генерація 3D геометрії ігрового контенту».

Галузь застосування: автоматичне створення ігрового контенту з допомогою алгоритмів.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання дипломного проекту першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення комп'ютерної системи для побудови тривимірної геометрії об'єктів з використанням процедурної генерації.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна література та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з операційною системою Arch Linux;
- наявнійсь графічного інтерфейсу для взаємодії з програмою.

5.2. Мінімальні вимоги до апаратного забезпечення

- Процесор: Intel Core i5-8250U;

- Оперативна пам’ять: 8 Гб;
- Графіка: Intel UHD Graphics 620.

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Arch Linux;
- Бібліотеки: glm 0.9.9.5-1; glfw-x11 3.3-2; imgui 9-2.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	19.11.2018
2.	Розроблення та узгодження технічного завдання	27.11.2018
3.	Аналіз існуючих рішень	20.12.2018
4.	Підготовка матеріалів першого розділу дипломного проекту	09.01.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	05.03.2019
6.	Розроблення програмного забезпечення	25.04.2019
7.	Відлагодження та рефакторинг програмного продукту	06.05.2019
8.	Підготовка матеріалів третього розділу дипломного проекту	17.05.2019
9.	Підготовка графічної частини дипломного проекту	20.05.2019
10.	Оформлення документації дипломного проекту	28.05.2019
11.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новоорозроблена			
	А4	ІАЦЛ. 045480.004 ПЗ	Процедурна генерація 3D	52		
			геометрії ігрового			
			контенту.			
			Пояснювальна записка.			
	А4	ІАЦЛ. 045480.005 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Взаємодія модулів.			
			Схема структурна.			
	А4	ІАЦЛ. 045480.006 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Основний потік програми.			
			Схема алгоритму.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЦЛ.045480.001 ОА		
Розробив		Пасічник М.Ю.			Процедурна генерація 3D геометрії ігрового контенту Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Коляда К.В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ, КВ-52	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЦЛ. 045480.007 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Потік для оновлення чанків			
			та створення завдань.			
			Схема алгоритму.			
	А4	ІАЦЛ. 045480.008 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Пул потоків для виконання			
			завдань.			
			Схема алгоритму.			
	А4	ІАЦЛ. 045480.009 Д1	Процедурна генерація 3D	1		
			геометрії ігрового			
			контенту.			
			Процедура обрахування			
			шуму Перлина.			
			Схема алгоритму.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Воксель (від англ. Volume та англ. pixel) — елемент простору, позначає значення певної величини в клітинках рівномірної просторової ґратки.

Кросплатформеність - властивість програмного забезпечення працювати більш ніж на одній програмній або апаратній платформі;

Лакунарність (від лат. lacuna — порожнина) — поняття фрактальної геометрії, що позначає міру того як фрактал заповнює простір.

Полігон – геометрична фігура, замкнена ламана.

Рендеринг (англ. rendering — візуалізація, проявлення, відмальовування, подання) — це процес отримання зображення з допомогою комп'ютерної програми.

Симплекс (від лат. simplex — простий) — геометрична фігура, що є багатовимірним узагальненням трикутника і тетраедра.

Чанк (з англ. Chunk – осередок, шматок, уламок) — метод, який використовує гра для розподілу нескінченних карт на частини.

GPU — окремий пристрій персонального комп'ютера або ігрової приставки, виконує графічний рендеринг.

Raytracing — спосіб створення надзвичайно реалістичного зображення тривимірних об'єктів чи сцени.

ВСТУП

В ігровій індустрії постійно необхідно багато якісного контенту. Для цього потрібно наймати високоякісних художників, геймдизайнерів, сценаристів.

Найчастіше для створення контенту необхідно витрачати багато сил, часу та ресурсів. Але останнім часом ведуться різні розробки, які дозволяють частково і навіть повністю перекласти роботу на комп'ютер.

Незважаючи на складність в реалізації, при правильному підході дані алгоритми дозволяються зменшити витрати, на кожному етапі розробки, зменшити кількість найнятого персоналу та збільшити різноманіття в кінцевому продукту. За рахунок цього, процедурна генерація є дуже актуальною.

Напрямів застосування процедурної генерації досить багато навіть в ігровій індустрії. Як найбільш поширену на трудомістку задачу, розглянемо лише генерування ландшафту.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Аналіз систем процедурної генерації ландшафту

1.1.1. Програма Terragen

Terragen — програма ландшафтного моделювання і анімації, що дозволяє створювати фотореалістичні пейзажі. Існують як безкоштовна, так і комерційна версії програми. Безкоштовна версія має деякі обмеження в порівнянні з комерційною.

Переваги:

- Фотореалістичні атмосфера і освітлення, що мають під собою безліч можливостей настройки;
- Потужний засіб відображення з можливістю прорахунку мільярдів полігонів, оптимізований для величезних просторів і дуже великих ландшафтів;
- Підтримка рослин в поширених форматах, в тому числі і зроблених за допомогою Xfrog;
- Робота з процедурними картами з високою деталізацією;
- Необмежене число кроків повернення дії;
- Можливість імпорту тривимірних об'єктів, і анімації створеної з використанням геометричних форм;
- Підтримка 64-розрядних систем, з деякими поліпшеннями для них;
- 3D—перегляд у поточному часу, включаючи можливість швидкої, 2.5—мірної відтворення хмар;
- Експорт геометрії у високій роздільній здатності.

Недоліки:

- Незручний нестандартний інтерфейс;
- Не підтримується прискорення через GPU, внаслідок чого повільне відображення зображення;
- Обмеження безкоштовної версії:
    - Максимальний розмір зображення: 800x600;
    - Максимальний рівень подробиці: 1.0;
    - Максимальний рівень згладжування: 3.

Програма дозволяє генерувати і модифікувати ландшафт як двомірну карту висот. З огляду на те, що карта висот генерується випадковим чином, отримати однакові пейзажі майже неможливо.

На рисунку 1.1 представлено приклад зимового гірського ландшафту, із елементами освітлення.

Рисунок 1.1 – Ландшафт, згенерований в Terragen.

1.1.2. Пакет програм Grome

Grome – це пакет екологічного моделювання, розроблений Quad Software, призначений для процедурної та ручної генерації великих віртуальних зовнішніх світів, придатних для ігор та інших 3D-моделей.

Особливостями даної програми є:

- Простий у використанні користувальницький інтерфейс;
- Попередній перегляд у режимі реального часу на кількох вікнах, з підтримкою апаратного прискорення;
- Можливість налаштувати систему на параметри інструменту;
- Підтримує необмежений розмір місцевості, використовуючи спеціальний механізм перекачування даних;
- Процедурна генерація карти висот з використанням фрактальних, терасових та ерозійних алгоритмів;
- Ручне редагування карти висот;
- Нова система накладання текстур, що дозволяє застосовувати різні методи затінення, процедурної генерації текстур на основі карти висот;
- Візуалізація води в режимі реального часу з використанням піксельного освітлення, відображення рельєфу. Генерація масок шарів води на основі висоти місцевості;
- Дозволяє рендеринг масивної популяції декорацій і трави. Анімовані ефекти трави для імітації вітру.

На рисунку 1.2 наведено приклад згенерованого ландшафту каньйону за допомогою ерозійних та фрактальних алгоритмів.

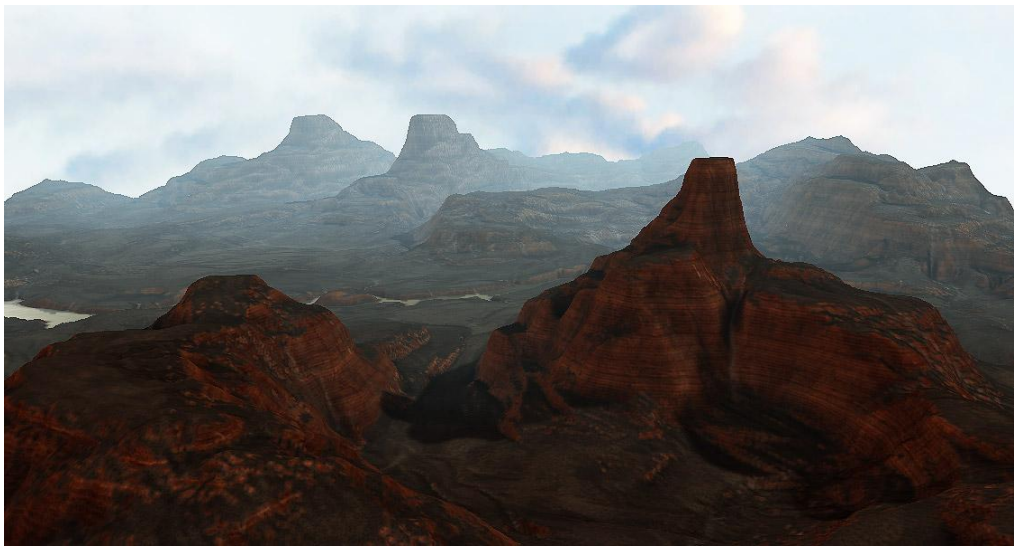

Рисунок 1.2 – Ландшафт, згенерований в Grome.

1.1.3. Програма Bryce

Bryce – програма для 3D-моделювання, рендеринга і анімації, що спеціалізується на генеруванні фрактальних пейзажів.

Програма призначена в основному для створення і візуалізації пейзажів. Можливості створення 3D-моделей внутрішніми засобами моделювання обмежуються деякими примітивами і джерелами світла. У Bryce є такі властивості:

- Нові типи джерел групового освітлення;
- Моделювання дерев;
- Моделювання гірських ландшафтів і їх експорт;
- Моделювання води;
- Редагування неба і хмар;
- Додаткове освітлення сцени від сонця;
- Імпорт об'єктів з більшості 3d—форматів;
- Підтримка імпорту анімації з DAZ Studio, Poser і ін.;
- Редактор матеріалів і текстур;
- Анімація, імпорт / експорт анімації;
- Інтеграція з програмами DAZ 3D;

- Візуалізація з допомогою raytracing алгоритму;
- Експорт та імпорт файлів FBX і COLLADA, а також моделей Google SketchUp
- Створення систем частинок (анімація, управління розміром, вагою, кількістю)

На рисунку 1.3 зображено приклад гірського ландшафту, з елементами освітлення та симуляції води.

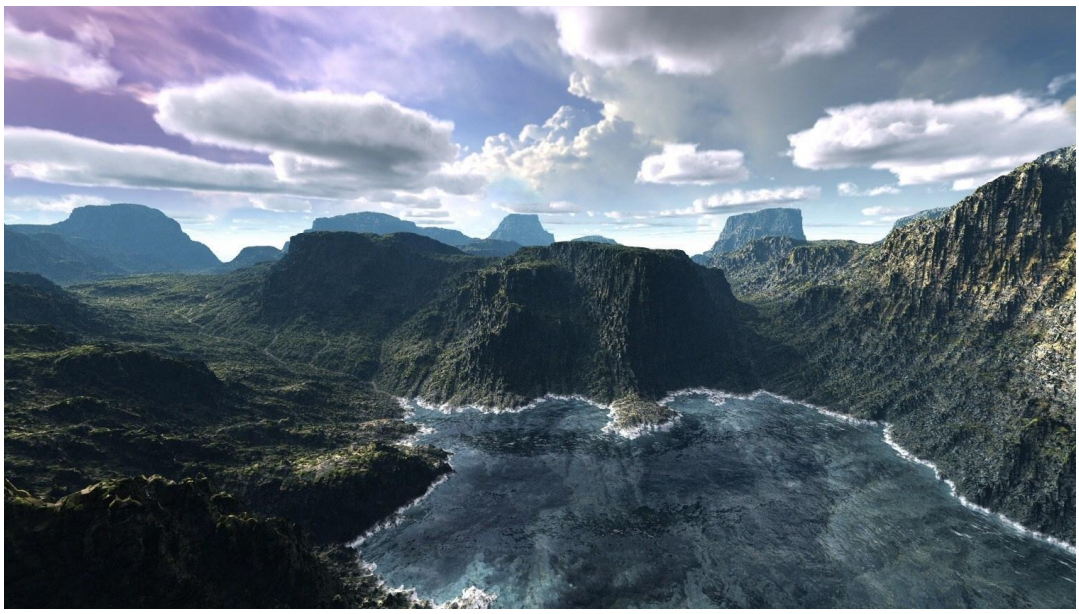

Рисунок 1.3 – Ландшафт згенерований в Bryce.

1.1.4. Генератор ландшафтів World Creator

World Creator — генератор ландшафтів у реальному часі, який виконує всі процеси генерації та проектування повністю на GPU, використовуючи тисячі ядер.

World Creator пропонує найсучасніші засоби проектування, такі як скульптура, штампування, швидке оздоблення, коригування, створення доріг і різних об'єктів.

World Creator дозволяє застосовувати і комбінувати багато різних фільтрів для зміни місцевості такі як розмивання, створення річок та озер, застосування відкладень, також надає змогу трансформувати, стилізувати,

моделювати потоки води та транспортування наносів, а також відкладення осаду, і багато іншого (див. рис. 1.4).

Рисунок 1.4 – Ландшафт згенерований в World Creator.

1.1.5. Графічна програма Houdini

Houdini — професійний програмний пакет для роботи з тривимірною графікою. Головною перевагою даного пакета полягає в тому, що він є середовищем візуального програмування.

Houdini відносно дорогий продукт, але існує і безкоштовна версія Houdini Apprentice - яку можна використовувати в некомерційних цілях.

Основні можливості програми:

- Моделювання
  - Безліч геометричних примітивів;
  - Полігональне моделювання;
  - Воксельне моделювання;
  - Сплайни;
  - L-системи;
  - Групування геометрії.
- Анімація;
- Фізичне моделювання:

- Твердих і м'яких тіл;
- Мотузок;
- Тканин;
- Волосся і вовни;
- Розрахунок фізичної поведінки і візуалізації газів і рідин.

- Освітлення
  - Великий набір джерел світла;
  - Прорахунок тіней, глобального освітлення, каустики.
- Рендеринг
  - Зручна нодова структура рендерингу;
  - Підтримка різних засобів візуалізації: Mantra, Renderman, mental ray;
  - Підтримка основних графічних форматів для експорту сцени з підтримкою багатопланових зображень;
  - Мережевий рендеринг;
  - Рендеринг для окремо взятих джерел світла або об'єктів, що належать до певної групи геометрії.

Houdini 17 SideFX представив нові інструменти, а також вдосконалив існуючі, щоб розширити можливості процедурної генерації ландшафту. Наприклад, моделювання ерозії реалізоване за допомогою алгоритмів які є науково обґрунтованими, що дозволяє отримати надзвичайно реалістичний ландшафт (див. рис. 1.5).

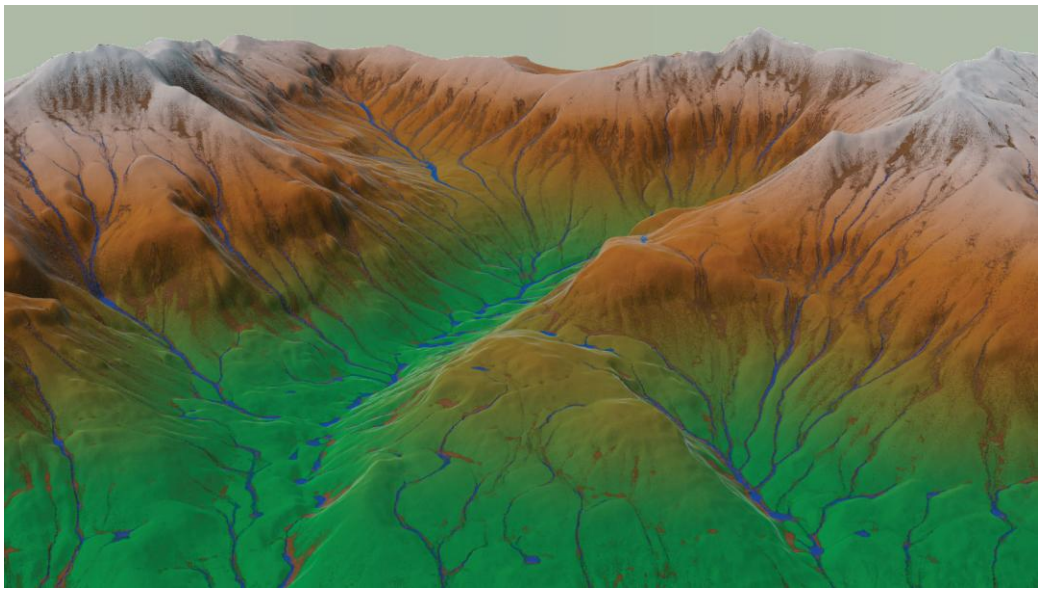

Рисунок 1.5 – Ландшафт згенерований в Houdini.

1.2. Аналіз алгоритмів процедурної генерації ландшафту

1.2.1. Випадкові числа

Найбільш легким способом заповнення карти висот є генерація псевдовипадкових чисел. В їх основі як правило лежать рекурентні формули, вони генерують цілі числа від 0 до деякого максимального m.

Основною проблемою заповнення карти висот випадковими числами є те, що кожне випадкове число є незалежним. В справжньому ландшафті висоти в різних точках є залежними одна від одної. Якщо заповнити текстуру випадковими пікселями, вона буде виглядати як телевізійний шум (див. рис. 1.6).

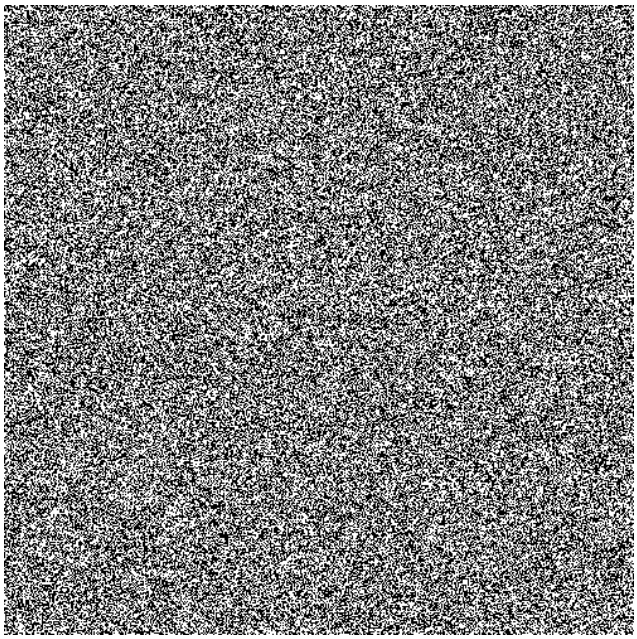

Рисунок 1.6 – Білий шум.

1.2.2. Інтерпольовані випадкові числа

Один із способів запобігти нереалістичні загострення це згладити згенерований ландшафт. Даний підхід полягає в тому, що потрібно згенерувати додаткові точки, і тоді знайти середнє значення висот сусідніх точок. Як і попередній, даний алгоритм також дає неприйнятні результати.

1.2.3. Шум Перлина

Особливість даного шуму (див. рис. 1.7) полягає в тому, що рівномірно розподілені точки з’єднуються плавним градієнтом.

Шум Перлина може бути визначений довільною кількістю вимірів. Реалізація складається з трьох кроків: визначення сітки, обчислення скалярного добутку градієнтних векторів, та інтерполяція між цими значеннями. Альтернативою для шуму Перлина є симплекс-шум що має більшу продуктивність та меншу кількість артефактів.

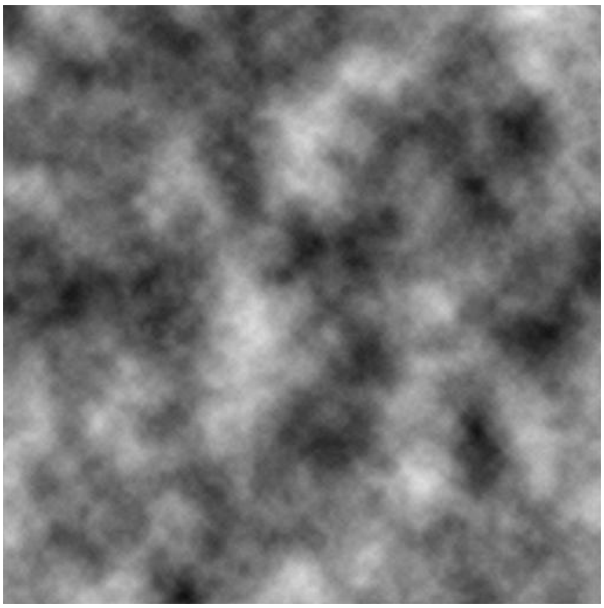

Рисунок 1.7 – Шум Перлина.

1.2.4. Симплекс-шум

Симплекс-шум (див. рис. 1.8) може бути визначений для довільної скінченної кількості вимірів. Реалізація складається з чотирьох кроків: відхилення координат, поділ на симплекси, вибір градієнта і сумування ядер.

Симплекс-шум поділяє простір на симплекси. Це зменшує кількість вузлів, проте при більше чотирьох вимірів, симплекси знаходяться недостатньо щільно, тим самим занулюючи функцію на вільних ділянках.

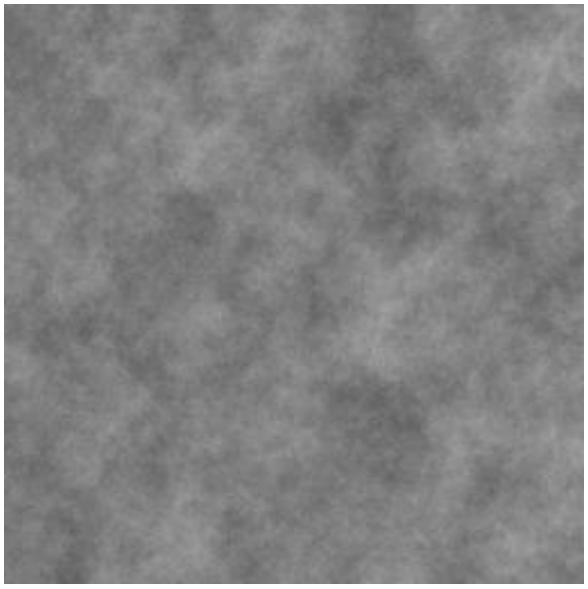

Рисунок 1.8 – Симплекс-шум.

1.2.5. Діаграма Вороного

Діаграма Вороного — це особливий вид розбиття метричного простору (див рис. 1.9), що визначається відстанями до заданої дискретної множини ізольованих точок цього простору.

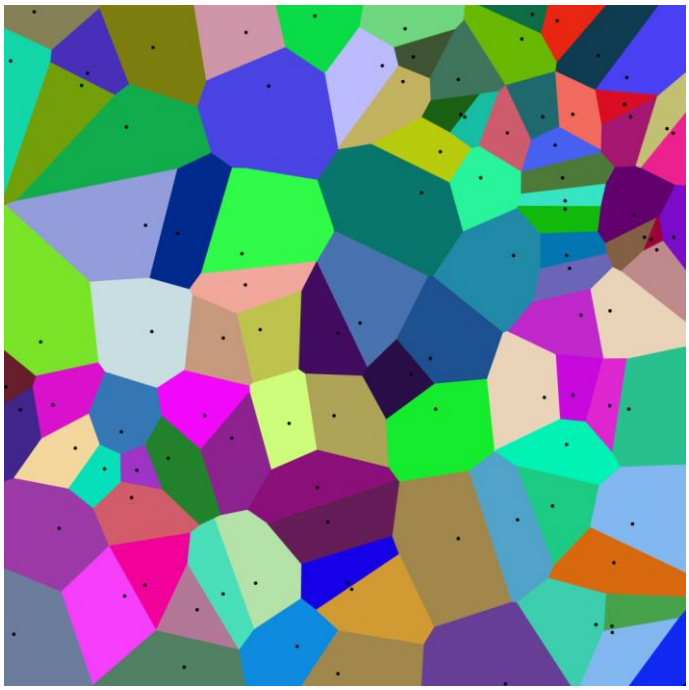

Рисунок 1.9 – Діаграма Вороного.

Існує декілька алгоритмів побудови діаграми Вороного:

- Прямий;
- Шляхом перетину напівплощин;
- Алгоритм Форчуна;
- Рекурсивний.

Оскільки проблема сортування дійсних чисел звідна до задачі обчислення діаграми Вороного, маємо що алгоритм Форчуна — оптимальний.

Алгоритм заснований на застосуванні замітаючої прямої (див рис. 1.10). Замітаюча пряма — це допоміжний об'єкт, що являє собою вертикальну пряму лінію. На кожному кроці алгоритму діаграма Вороного побудована для множини, що складається з замітаючої прямої та точок ліворуч від неї. При цьому межа між областю Вороного прямою та областями точок складається з відрізків парабол. Пряма рухається зліва направо. Щоразу, коли вона

проходить через чергову точку, ця точка додається до вже побудованої ділянки діаграми.

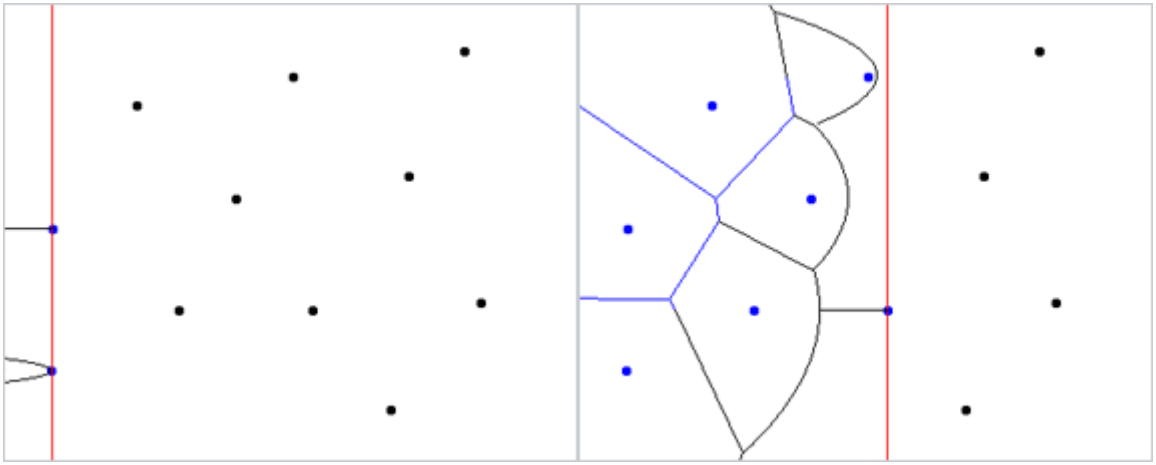

Рисунок 1.10 – Побудова діаграми Вороного алгоритмом Форчуна.

1.2.6. Алгоритм Diamond-Square

Алгоритм diamond-square це алгоритм для генерації фрактальних ландшафтів (див. рис. 1.11). Він заснований на одновимірному алгоритмі midpoint displacement. Цей алгоритм найкраще підходить для генерації реалістичних ландшафтів.

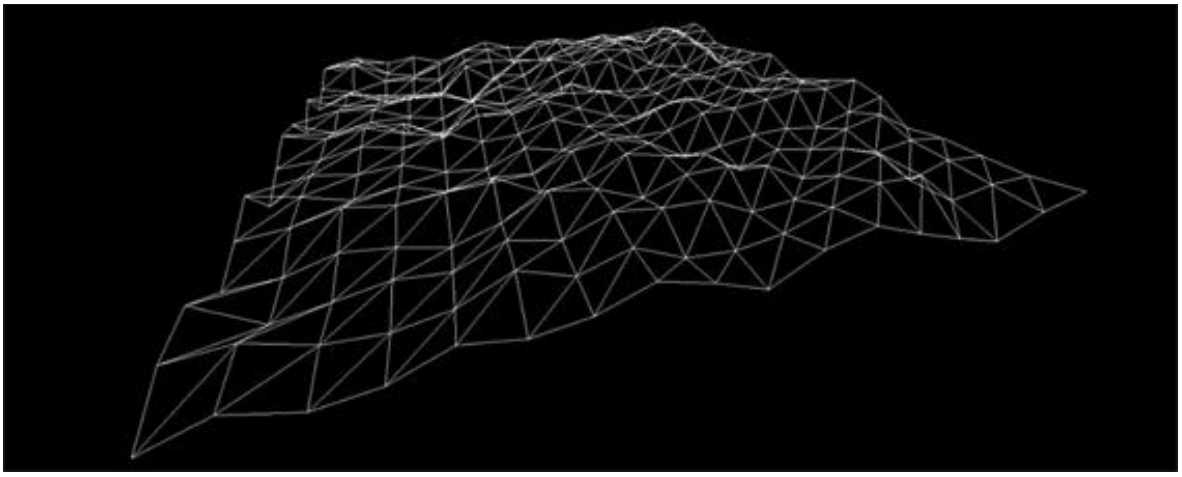

Рисунок 1.11 – Ландшафт створений за допомогою Diamond-Square.

Суть алгоритму полягає в тому, що спочатку встановлюється початкова висота у чотирьох кутових точках масиву.

Далі для кожного масиву знаходиться середня точка, в яку записується середнє значення чотирьох кутових точок зміщене на випадкову величину.

Далі для кожного ромбу в масиві знаходиться середня точка, в яку записується середнє значення чотирьох кутових точок зміщене на випадкову величину.

На кожній ітерації випадкове значення зменшується, кількість ітерацій залежить від бажаного рівня деталізації (див. рис. 1.12).

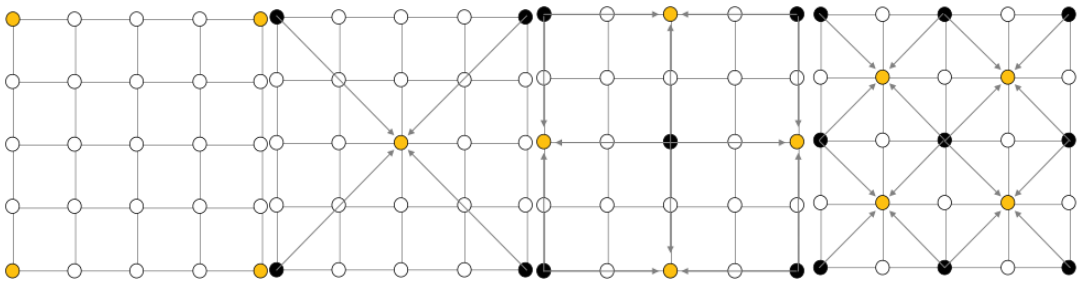

Рисунок 1.12 – Візуалізація алгоритму Diamond-Square.

1.2.7. Моделювання ерозії

Ерозія (від лат. erosio) — процес деформації ландшафту водним потоком (водна ерозія), вітром (вітрова ерозія), льодом. Ерозія — один з головних зовнішніх чинників формування рельєфу земної поверхні. Частина процесу денудації. Розрізняють схилову й руслову водну ерозію. В результаті водної ерозії утворюються яри, балки, річкові долини тощо.

Алгоритм працює таким чином, що спочатку генерується карта висот, використовуючи довільний алгоритм. Далі в залежності від кількості ітерацій моделюється зовнішній вплив на ландшафт.

Таким чином при великій кількості ітерацій досягається реалістичне відображення процедурно-згенерованого ландшафту.

1.3. Обґрунтування теми

Метою даного дипломного проекту є створення програмного забезпечення, яке буде зручно використовувати для процедурної генерації ландшафту.

Особливістю розробки є забезпечення високої швидкості генерації за допомогою векторизації, виконання алгоритмів окремо від основного потоку

програми, а також створення процедурно-згенерованої текстури ландшафту. Користувачеві надається можливість попереднього перегляду згенерованого ландшафту та зміни налаштувань генерації.

2. ОПИС ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ

2.1. Що таке процедурна генерація

Процедурна генерація – це автоматичне створення контенту за допомогою набору алгоритмів, найчастіше для створення ігрового контенту [12].

Засоби процедурної надають можливість створювати контент самостійно, або спільно із користувачем, геймдизайнером.

Процедурна генерація є однією з найбільш актуальних та швидко розвиваючих напрямів досліджень, вона містить алгоритми із багатьох сфер, наприклад статистики або штучного інтелекту.

2.2. Особливості процедурної генерації

Оскільки створюваний контент повинен задовільнити певні критерії, а також вирішувати поставлене перед ним завдання, тому найчастіше виділяють наступні особливості:

- Швидкість: контент повинен генеруватися в задовільних часових рамках по мірі необхідності під час процесу розробки;
- Надійність: генератор повинен задовольнити задані критерії та забезпечувати виконання поставлених умов для кінцевого результату;
- Контрольованість: можливість контролювати процес генерації контенту в доступній можливості свободи;
- Різноманітність: створювати такий контент, щоб не було відчуття одноманітності.
- Правдоподібність: згенерований контент повинен виглядати так, ніби він створений людиною.

2.3. Можливі недоліки процедурної генерації

Ознайомившись із засобами процедурної генерації можна виділити основні недоліки при використанні даної системи. Досить ймовірно що ці проблеми можна оминути при правильному плануванні розробки, наявності альтернативних алгоритмів. Розглянемо основні недоліки які можуть виникнути:

- Надмірна складність:
  - неможливість забезпечити необхідний контроль якості;
  - виникає необхідність покриття великою кількістю тестів;
- Великі витрати часу:
  - Алгоритми процедурної генерації можуть бути дуже складними при розробці;
  - Процес процедурної генерації може негативно впливати на продуктивність виконання гри;
  - Досить складно керувати результатом, потрібно постійно вносити зміни щоб задовольнити потреби.
- Випадковість:

  З такою помилкою зазвичай зіштовхуються початківці. Необхідно мати досвід та навички, щоб процедурно згенерований контент вписався в ігрову механіку. Звичайна випадковість створює відчуття повторюваності, що призводить до не зацікавлення гравця.

2.4. Приклади використання

За допомогою процедурної генерації створюються карти ігрового світу, ігрові рівні, музика, анімація, предмети, квести, текстури, персонажі та багато іншого.

2.4.1. Гра “The Binding of Isaac: Rebirth”

The Binding of Isaac: Rebirth (з англ. - «Жертвопринесення Ісаака: Відродження ») – комп'ютерна гра в жанрі action-adventure з елементами roguelike і шутера з виглядом зверху (див. рис. 2.1).

В даній грі процедурна генерація використовується для генерації лабіринту, параметрів предметів які можна підібрати, а також ворогів.

Рисунок 2.1 – Постер The Binding of Isaac: Rebirth

З допомогою даного методу, у гравця складається ілюзія нескінченної гри. Подібними характеристики мають ігри жанру roguelike. Такі елементи процедурної генерації використовуються для того, щоб створити світ із захопливою різноманітністю та динамікою.

2.4.2. Гра “Minecraft”

Minecraft – інді-ігра жанру пісочниця у відкритому світі (див. рис. 2.2). Minecraft надає можливість гравцю досліджувати процедурно згенерований світ, що складається з вокселів. Перед кожним початком гри гравець може задати параметри генерації світу. В даній грі генерується ландшафт, пещери, поселення, ліси, водойми, кліматичні зони.

Рисунок 2.2 – Постер Minecraft

2.4.3. Гра “No Man’s Sky”

No Man`s Sky (далі NMS) – це науково-фантастична гра про дослідження і виживання в нескінченному процедурно-згенерованому всесвіті (див. рис.2.3). Головною особливістю даної гри – випадково згенеровані галактики, планети, та системи зірок.

В NMS світи заповнюються згенерованими живими створіннями, власною екосистемою та цінними ресурсами.

Рисунок 2.3 – Постер No Man’s Sky

2.4.4. Стратегія “Crusader Kings II”

Crusader Kings II (див. рис. 2.4) – відеогра в жанрі глобальної стратегії в реальному часі. За допомогою процедурної генерації, персонажі стратегій Crusader Kings II стають унікальними, набувають різних рис характеру, що визначають поведінку; реакцію персонажів, які, як правило, дуже прості, але в комплексі можуть здаватися складними.

Рисунок 2.4 – Вікно інформації про персонажа

2.4.5. Гра “Middle-earth: Shadow of Mordor”

Middle-earth: Shadow of Mordor (далі Shadow of Mordor) – мультиплатформна відеогра в жанрі Action.

Shadow of Mordor виходить за рамки звичного уявлення про процедурно-згенерований контент. Гра закладає концепцію налагодження особистих відносин гравців з орками.

Кожний орк унікальний, від імені і зовнішності до манери спілкування та відносин з іншими орками. Вони створюються за допомогою алгоритму Nemesis System (див. рис. 2.5).

Якщо орк вбиває персонажа, він отримує нове звання, а якщо його ранять на ньому з'являється процедурно-згенерований шрам.

Рисунок 2.5 – Процедурно згенерований орк із шрамом

Процедурно-згенеровані характеристики ворогів і шрами, переконують гравців в тому, що вони борються з унікальними противниками, а не одними і тими ж персонажами, доступними для інших користувачів

2.4.6. Рогалик "RymdResa"

Краса RymdResa (див. рис. 2.6) полягає в тому, що сама по собі поетика життя космонавта на борту корабля гравця, загальна атмосфера і стиль — небезпека, самотність, спокій, неосяжність просторів — виникає завдяки алгоритмам процедурної генерації, що формують цілий світ, які під час кожної нової сесії пропонують дослідити унікальний всесвіт.

Рисунок 2.6 – Гемплей RymdResa

2.4.7. Рогалик “Dwarf Fortress”

Dwarf Fortress (див. рис. 2.7) – це безкоштовна відеогра, що поєднує жанри roguelike та симулятор містобудування у фентезійному світі.

В цій грі генерується докладний родовід гномів, яка налічує тисячі років. Це були історії про кохання, втрати, зраду, катастрофи, пригоди, випробуваннях голодом, жахливих пожежах, вбивствах, смішних пригодах і багато іншого.

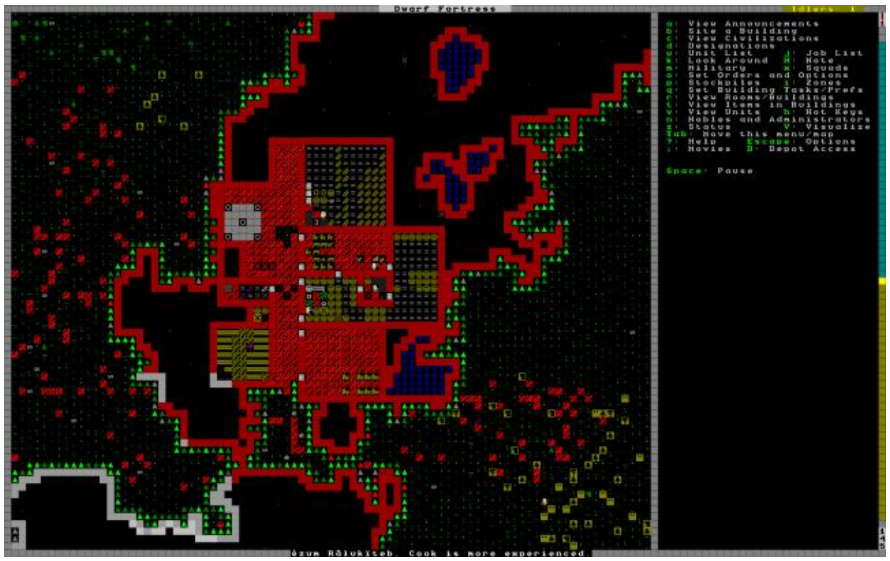

Рисунок 2.7 – Поселення гномів

Dwarf Fortress відрізняється від інших продуктів свого класу не складністю концепції, а тим, як окремі прості елементи гри нашаровуються один на одного, що сприяє процедурної генерації досвіду гравця — завдяки цьому кожна сесія відкриває для нього новий, унікальний, багатовимірний і абсолютно непередбачуваний світ гномів.

3. РЕАЛІЗАЦІЯ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ЛАНДШАФТУ

3.1. Середовище програмування

Важливим критерієм обрання мови програмування було те, що вона має бути відносно безпечною, мати високу швидкість виконання, наявність вільних бібліотек, а також підтримка SIMD. Було проаналізовано декілька існуючих рішень:

- C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET;
- C++ – мова програмування високого [13] рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної;
- C – універсальна, процедурна, імперативна мова програмування загального призначення.

C# як основна мова програмування не підійшов із декількох причин, це:

- Пакування та розпакування. Механізм перетворення розмірних типів даних мови C# із скалярних в вказівникові і назад через задіяння властивостей фундаментального базового класу Object. Пакування та розпакування є досить дорогою операцією;
- Збирання сміття (англ. garbage collection) – це один із методів автоматичного керування оперативною пам'яттю комп'ютера під час виконання програм. Недоліком даного методу є відсутність прямого керування пам’яттю, та значний вплив на швидкодію виконання програми;
- Немає можливості повноцінно використовувати всі переваги SIMD;
- Орієнтованість на Windows OS.

C та C++ є досить схожими, і відповідають очікуваним критеріям:

- Ручне керування пам’яттю;
- Значно більша продуктивність коду і менша вимогливість до ресурсів порівняно із C#;
- Кросплатформеність;
- Самодостатність кінцевого додатку.

Основною перевагою C++ є можливість обрахунку виразів на етапі компіляції за допомогою специфікатора constexpr.

Тому, зважаючи на всі перераховані особливості, C++ було обрано як основну мову програмування.

Оскільки цільовою платформою програмування є Arch Linux, на вибір було декілька середовищ програмування на C++:

- Netbeans for C/C++ Development. Netbeans – це безкоштовний кросплатформений додаток з відкритим кодом. Особливостями Netbeans є:
  - Добре інтегрований коду C++;
  - Засоби відкладки коду;
  - Підтримка тестів;
  - Наявність автоматичної упаковки скомпільованого додатку в архів;
  - Навігація файлами;
  - Автодоповнення;
- Code::Blocks. Code::Blocks – це безкоштовний кросплатформений додаток із пітримкою розширень і кастомізації. Особловостями Code::Blocks є:
  - Підтримка багатьох компіляторів;
  - Висока швидкість компіляції використовуючи інтегрований компілятор.
  - Підтримка відкладки коду.

- JetBrains CLion. CLion – не безкоштовний, але дуже потужний та кросплатформений засіб програмування C/C++. Особливостями Clion є:
    - Підтримка засобів контролю версії GIT, Subversion, Mercurial, CVS, Perforce, TFS;
    - Інтегрований відкладчик коду;
    - Аналіз коду на льоту;
    - Підтримка автогенерації коду та рефакторингу;
    - Легка навігація та перехід між символами.
- CodeLite IDE. Майже безкоштовний кросплатформений додаток із відкритим кодом. Особливостями CodeLite IDE є:
    - Автодоповнення коду;
    - Підтримка багатьох компіляторів C/C++;
    - Відображення помилок в коді;
    - Підтримка відкладки коду;
    - Підтримка рефакторингу;
    - Пітримка засобів контролю версії.
- Eclipse CDT(C/C++ Development Tooling). Eclipse – кросплатформений додаток із відкритим кодом. Особливостями Eclipse є:
    - Підтримка створення проектів;
    - Навігація файлами;
    - Підтримка рефакторингу коду;
    - Підтримка автогенерації коду;
    - Підтримка багатьох компіляторів;
    - Наявність засобів декомпіляції;

Оскільки всі вище перераховані додатки надають схожий функціонал, важливим критерієм стало візуальний вигляд програми. Таким чином було обрано JetBrains CLion.

3.2. Опис роботи алгоритму

Шум Перлина – це надзвичайно потужний алгоритм, який часто використовується при розробці ігор. Шум Перлина може використовуватися як для генерації процедурної геометрії, так і для генерації текстур, ефектів вогню, хмар.

Шум Перлина можна реалізувати для будь якої n-вимірної системи, але при збільшенні вимірів, обрахунку стають дедалі складнішими. Тому у 2002 році була випущена покращена версія алгоритму, яка частково прибирає даний недолік.

Алгоритм працює таким чином, що для заданої точки визначається комірка, в якому вона розміщена (див. рис. 3.1).

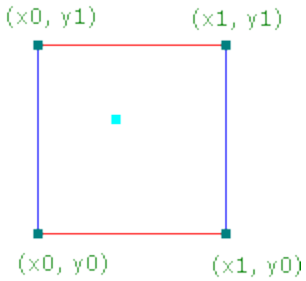

Рисунок 3.1 – Комірка, в якому точка відображає задані координати.

Далі, для кожного вузла комірки визначається випадковий вектор, який вказує напрямок градієнту (див. рис. 3.2). З метою зменшення витрат на обчислення нових градієнтів, деякі реалізації використовують хеш таблиці і таблиці пошуку. Також використання хешу дозволяє задавати сиди для генерації.

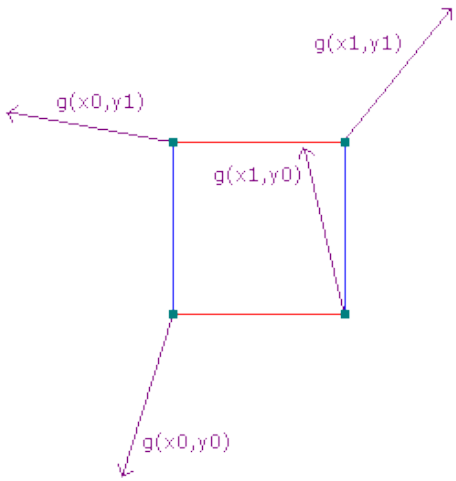

Рисунок 3.2 – Градієнтні вектори.

В покращеному алгоритмі використовуються не випадкові градієнти, а вектори з центру комірки до ребер.

Для кожного вузла необхідно визначити вектор відстані між точкою і координатами вузла (див. рис. 3.3).

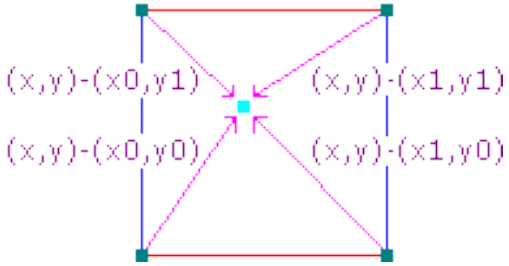

Рисунок 3.3 – Вектори відстані до точки.

Тоді для кожного вузла комірки обраховується скалярний добуток вектора відстані та градієнту. На рисунку 3.4 відображено вплив градієнтів на значення шуму Перлина.

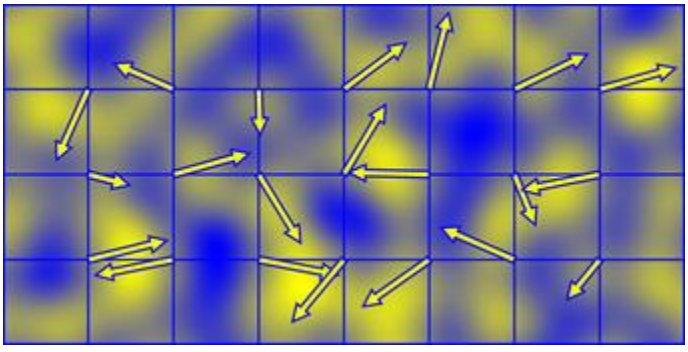

Рисунок 3.4 – Вплив градієнтів на шум Перлина.

Наступний крок – це інтерполяція скалярних добутків, обчислених для кожного вузла. Інтерполяція може виконуватися за допомогою функції:

$$f(x) = a_0(1-x) + a_1 x$$

Таким чином, послідовно виконавши всі кроки, маємо можливість обрахувати значення шуму Перлина для довільної точки.

3.3. Модифікація алгоритму

3.3.1. Багатопотоковість

Багатопотоковість – це властивість операційної системи або програмного забезпечення, в тому, що, процес може складатися з декількох потоків, які виконуються паралельно, або навіть одночасно на багатопроцесорних системах.

Суттю багатопотоковості є квазі-багатозадачність, яка виконується на рівні одного виконуваного процесу, тобто всі потоки виконуються в адресному просторі процесу.

Для значного прискорення генерації, ландшафт розділений на чанки. Кожен чанк зберігає позицію, дані для відображення, параметри, з якими він був згенерований, та додаткові дані (див. додаток А, строка 1).

Було реалізовано три основні частини для взаємодії з чанками, кожна з них має власний функціонал і може працювати незалежно одна від одної.

Перша частина – це створення повідомлень для генерації нових чанків, в залежності від розташування гравця, та оновлення існуючих при змінні параметрів генерації.

Наступна частина відповідає за обробку повідомлень, та генерацію даних. Дану частину було реалізовано за допомогою пулу потоків.

Пул потоків — це набір фіксованої кількості потоків, створених під час запуску програми. Основною особливістю пулу є підвищення продуктивності

виконання задач, та уникнення затримок через створення окремих потоків на кожну задачу.

Кількість потоків які входять до пулу визначається максимальним допустимим числом одночасно виконуваних потоків

Велика кількість потоків призводить до неефективності. OS періодично перемикає один потік на інший на процесорі. Це включає збереження стану одного потоку і завантаження стану іншого потоку з пам'яті. Тому для кожного перемикання контексту потрібно не малий час [4].

Остання частина є основним потоком виконання програми, яка відповідає за керування іншими частини, створення чанків на основі згенерованих даних, та відображення на екрані.

Для взаємодії всіх частин використовуються черги даних, та засоби синхронізації.

3.3.2. Паралельні обчислення

Паралельні обчислення – це форма обчислень, в яких кілька дій проводяться одночасно [15]. Є кілька різних рівнів паралельних обчислень:

- Бітовий;
- Інструкцій;
- Даних;
- Паралелізм задач;

Відповідно до класифікації Флінна визначено чотири основних класи обчислювальних систем:

- SISD (один потік команд – один потік даних);
- SIMD (один потік команд – багато потоків даних);
- MISD (багато потоків команд – один потік даних);
- MIMD (багато потоків команд – багато потоків даних).

Вище наведена класифікація відображає співвідношення потоків команд і потоків даних.

До обчислювальних систем, що здатні реалізувати паралельні обчислення можна віднести два класи: SIMD та MIMD.

Оскільки цільова система підтримує лише SIMD, вона і буде використовуватися під час розробки.

SIMD — це елемент класифікації згідно з таксономією Флінна для паралельних процесорів, де до багатьох елементів даних виконується одна або однакові команди (див. рис. 3.5). SIMD — це одна з головних умов, котра гарантує можливість паралельного виконання алгоритмів.

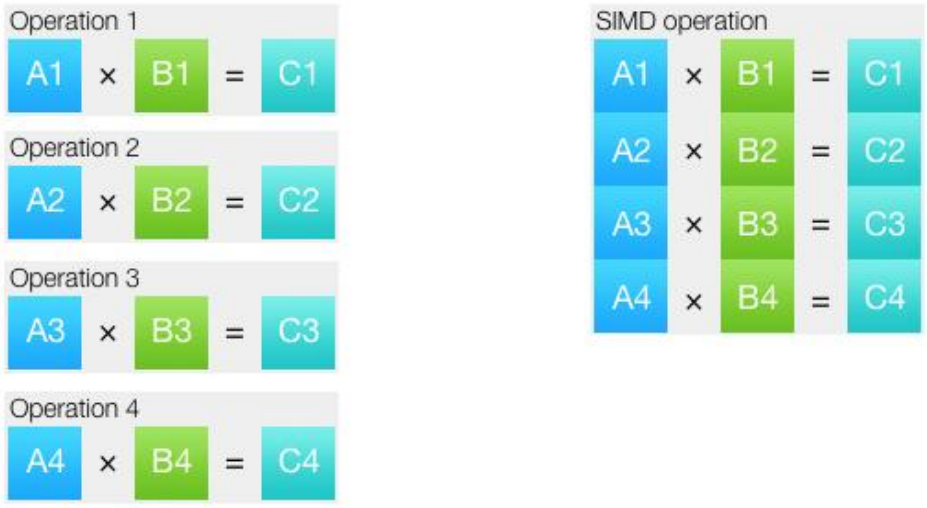

Рисунок 3.5 – Виконання множення послідовно (ліворуч) та SIMD (праворуч)

SIMD використовується при обробці мультимедійної інформації, накладанні фільтрів, де необхідно виконувати однакові дії над кожним пікселем.

Приклади використання SIMD:

- Векторний процесор – процесор, в якому операндами деяких команд можуть слугувати впорядковані масиви даних — вектори;
- GPU – обробка мультимедійної відео інформації;
- Архітектура MAJC – багатоядерний та багатопоточний мікропроцесор, який орієнтований на обробку мультимедійної інформації у мережі;

- Векторні розширення центрального процесора:
  - MMX(Multimedia Extensions);
  - 3DNow!
  - SSE(Streaming SIMD Extensions);
  - AVX(Advanced Vector Extensions).
- Векторні розширення NEON ARM процесорів.

Недоліками SIMD є:

- Не всі алгоритми можливо векторизувати;
- Збільшення споживання енергії процесором;
- Більшість компіляторів не підтримують автоматичну векторизацію алгоритмів. Автоматична векторизація є дуже агресивним видом оптимізації, що найчастіше супроводжується неочікуваною поведінкою програми, та частими збоями. Також, автоматична векторизація є активною областю досліджень інформатики.
- Різні архітектури CPU впроваджують різні розміри регістрів. Таким чином, робота алгоритмів на різних платформах може відрізнятися.
- SIMD накладає обмеження на вирівнювання даних (див. рис. 3.6).

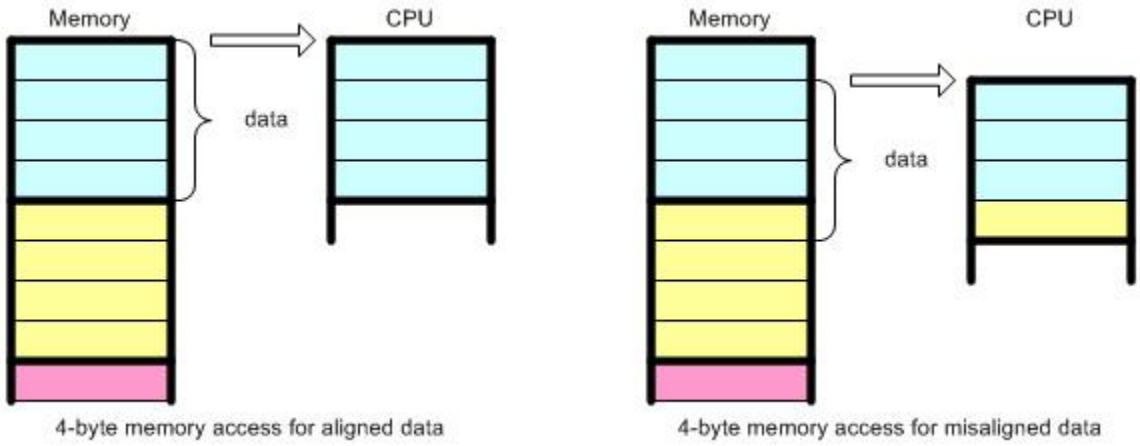

Рисунок 3.6 – Ілюстрація того, як CPU звертається до частини даних розміром 4 байти. Вирівнянні дані по 4 байти (ліворуч) та не вирівнянні (праворуч).

Якщо дані не були вирівнянні, CPU повинен виконати додаткову роботу для доступу до даних. Це сприяє тому, що зменшиться продуктивність виконання програми, і таким чином використання SIMD не матиме жодної переваги.

SIMD-розширень існує багато. Перелік розширень за хронологією появи:

- MMX;
- SSE;
- SSE2;
- SSE3;
- SSSE3;
- SSE4.1;
- SSE4.2;
- AVX;
- AVX2;
- AVX-512.

Зазвичай, наявність нової версії розширень в CPU, свідчить про підтримку всіх попередніх. Найновішою версією SIMD-розширень що реалізований на використовуваному CPU є AVX2, тому даний набір інструкцій буде використовуватися для модифікації алгоритму.

SIMD в основному підтримує наступні типи даних:

- Цілі числа по 8 біт (зі знаком чи без);
- Цілі числа по 16 біт (зі знаком чи без);
- Цілі числа по 32 біта (зі знаком чи без);
- Цілі числа по 64 біта (зі знаком чи без);
- Числа з плаваючою точкою одинарної точності, 32 біта;
- Числа з плаваючою точкою подвійної точності, 64 біта.

AVX2 – розширення системи команд x86 для мікропроцесорів Intel і AMD. AVX2 підтримує набір інструкцій для взаємодії які виконуються блоками по 256 біт.

Для ініціалізації та математичних операції необхідно використовувати спеціальні функції [6]. Дані функції було реалізовано із підтримкою обчислень на етапі компіляції.

Оскільки завдяки AVX2 можна за один раз обраховувати 8 координат, тому для полегшення реалізації необхідно, щоб

$(chunkSize + 1)\ \%\ 8 == 0$, де

chunkSize – розмір чанку в умовних одиницях.

Для заповнення карти висот, необхідно згенерувати шум для кожної координати в чанку (див. додаток А, строка 13), для SIMD (див. додаток А, строка 40).

Для реалізації шуму Перлина, спочатку необхідно визначити комірку в якій знаходиться задані координати (див. додаток А, лінія 66). Для SIMD це буде виглядати наступним чином (див. додаток А, строка 71).

Далі необхідно визначити градієнти на кожному вузлі комірки. Для збільшення продуктивності, було вирішено використовувати хеш таблиці, та таблиці пошуку (див. додаток А, строка 76).

За допомогою SIMD немає можливості реалізувати хеш таблиці та таблиці пошуку, тому було вирішено обраховувати хеш, та визначати градієнт на основі даного хешу. Функція для хешування може бути будь-якою, основне щоб вона забезпечувала високу ступінь унікальності, аби не виникало схожих паттернів (див. Додаток А, лінія 81). Реалізація визначення градієнту на основі хешу[3].

Далі необхідно визначити ваги інтерполяції, та власне обрахувати кінцевий результат (див. Додаток А, строчка 98), для SIMD (див. додаток А, строка 110).

Таким чином, отримаємо реалізацію алгоритму шуму Перлина.

3.4. Процедурне текстурування

3.4.1. Що таке процедурна текстура

Процедурна текстура — це зображення яке створене з використанням математичного опису (див. рис. 3.7). Перевагою такого підходу є низька вартість зберігання, необмежена роздільна здатність і зручне відображення текстур. Процедурно-згенеровані текстури часто використовуються для моделювання поверхневих або об'ємних зображень природних елементів, таких як дерево, мармур, граніт, метал, камінь тощо.

Зазвичай для створення природних текстур використовуються фрактальні шуми і функцій турбулентності. Ці функції використовуються як числове представлення випадковості, що зустрічається в природі.

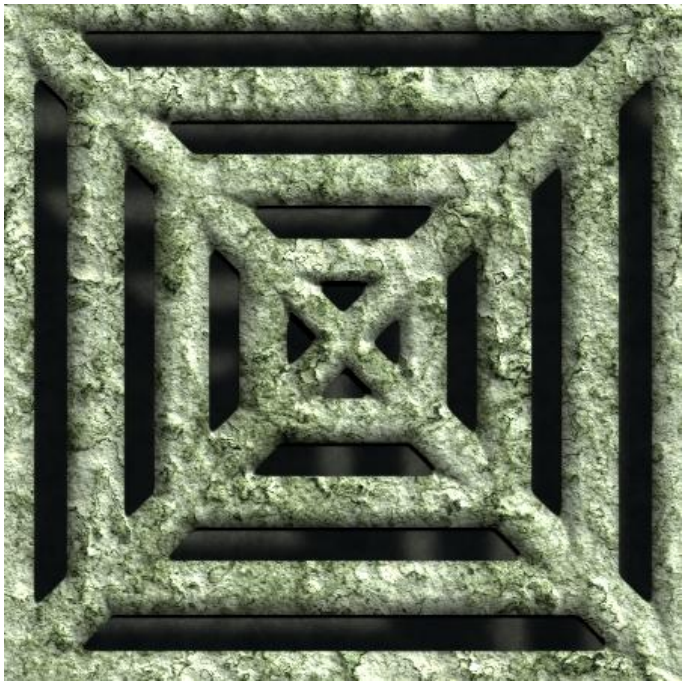

Рисунок 3.7 – Процедурна текстура решітки підлоги, створена редактором текстур Genetica.

Властивості процедурних текстур:

- оборотність. У процедурній текстурі зберігається вся історія її створення;
- малий розмір;
- необмежена кількість варіацій;

- масштабованість до будь-якого розміру.

Для створення природних текстур, таких як дерево, граніт, метал, каміння, лава, використовуються фрактальний шум і ніздрюваті текстури (англ. cellular textures). Існує декілька видів текстурування.

Суцільне текстурування

Суцільне текстурування – це процес, при якому визначається кожна точка в тривимірному просторі видимої поверхні моделі, на основі її властивостей матеріалу, таких як колір, блиск, або нормаль.

Такі текстури не піддаються спотворень простору параметрів поверхні. Вони залишаються послідовними і мають особливості постійного розміру, незалежно від спотворень у системах координат поверхні [8].

Спочатку такі текстури створювалися на основі накладання функцій шуму, таких як Симплекс—шум та шум Перлина. Проте, в даний час, є можливість використовувати як структуровану регулярну сітку, так і структурованих нерівних текстур або виключно стохастичні текстури.

Клітинне текстурування

Клітинне текстурування (див. рис. 3.8) відрізняється від більшості існуючих методів, оскільки вона не залежить від шумових функцій, але вони можуть використовуватися в якості доповнення. Клітинні текстури засновані на характеристичних точках, які розкидані по тривимірному просторі. Потім ці точки використовують для розподілення простору на великі області.

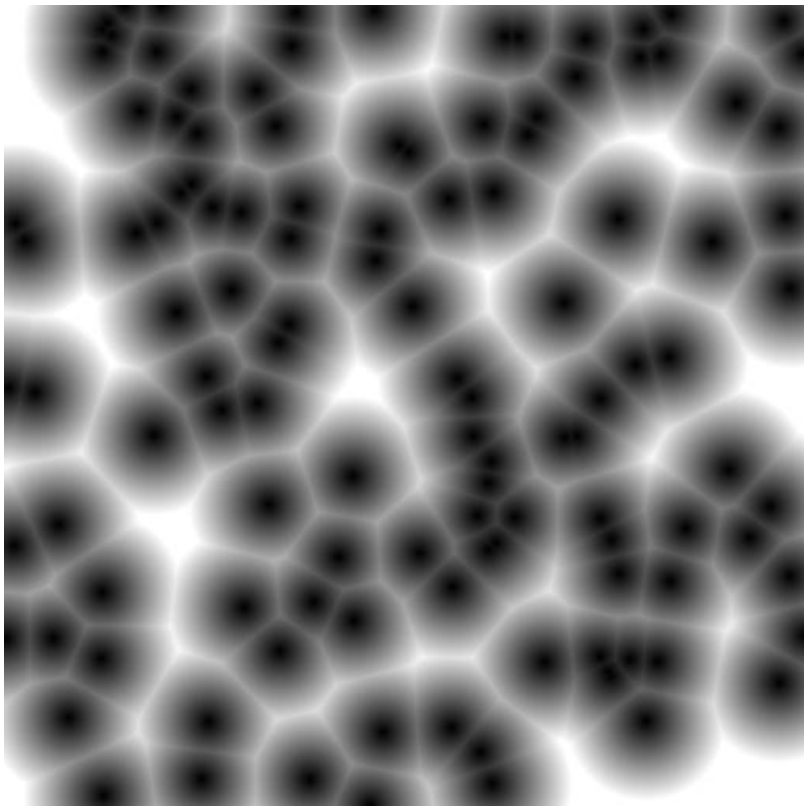

Рисунок 3.8 – Клітинна текстура

Генетичні текстури

Генетично згенеровані текстури є експериментальним підходом для створення текстур. Це автоматизований процес, керований людиною. Потік контролю зазвичай має комп'ютер, що генерує набір текстур. З них користувач вибирає. Комп'ютер потім генерує інший набір текстур на основі обраного користувачем шляхом мутації і кроссоверу. Процес продовжується, поки не буде створено відповідну текстуру для користувача. Оскільки результат важко контролювати, цей метод зазвичай використовується тільки для експериментальних або абстрактних текстур.

Самоорганізовувальні текстури

Починаючи з простого білого шуму, процеси самоорганізації можуть призвести до структурованих моделей, зберігаючи певну випадковість. Реакційно-дифузійні системи є одним із способів генерування таких текстур. Реалістичні текстури можуть генеруватися шляхом моделювання складних хімічних реакцій всередині рідини. Ці системи можуть показувати поведінку,

подібну до реальних процесів, що зустрічаються в природі, такі як маркування тварин.

3.4.2. Генерація текстури

Для генерації текстури було використано раніше реалізований алгоритм шуму Перлина.

В залежності від висоти, кожний піксель текстури був замальований у відповідний колір, далі на текстуру було накладено шум Перлина для збільшення деталізації.

3.5. Використання

Було реалізовано графічний інтерфейс (див. рис. 3.9) для можливості зміни параметрів процедурно-згенерованого ландшафту.

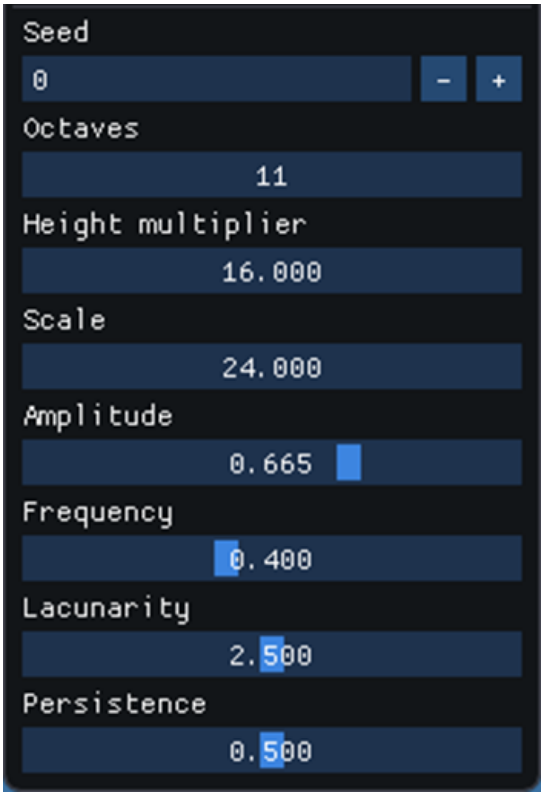

Рисунок 3.9 – Елементи керування параметрами генерації

Графічний інтерфейс дає можливість налаштувати:

- Початкове значення сиду генерації;

- Роздільної здатності (далі октав). Кількість шарів згенерованого шуму що накладаються (див. рис.3.10);
- Коефіцієнт множення висоти;
- Масштабу;
- Амплітуди (максимальне відхилення від нуля);
- Частоту. Коефіцієнт, який обернено пропорційний до масштабу, визначає кількість циклів на одиницю довжини (див. рис. 3.11);
- Лакунарність. Коефіцієнт, який визначає, наскільки швидко частота збільшується для кожної наступної октави;
- Стійкість. Коефіцієнт, який визначає, наскільки швидко згасає амплітуда для кожної наступної октави (див. рис. 3.12).

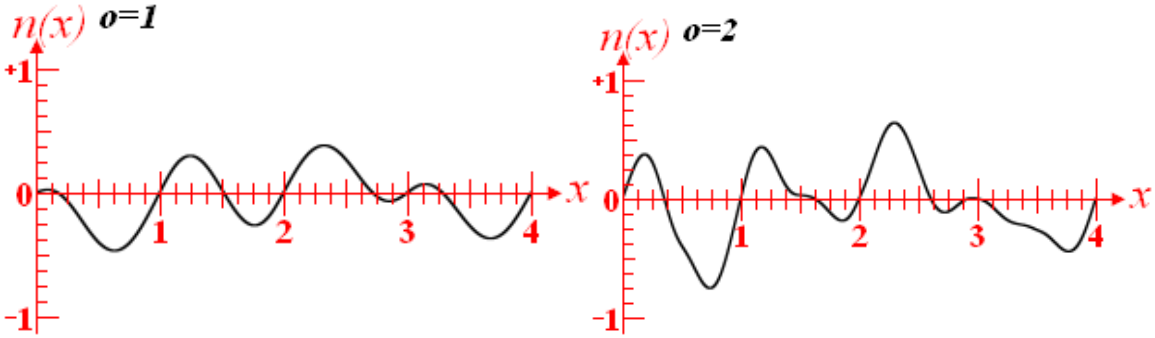

Рисунок 3.10 – Графік залежності шуму від кількості октав при значеннях 1 та 2 відповідно.

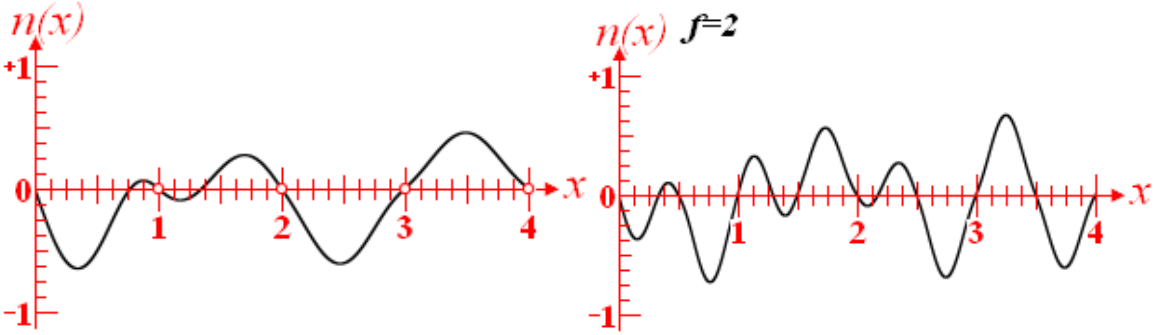

Рисунок 3.11 – Графік залежності шуму від частоти при значеннях 1 та 2 відповідно.

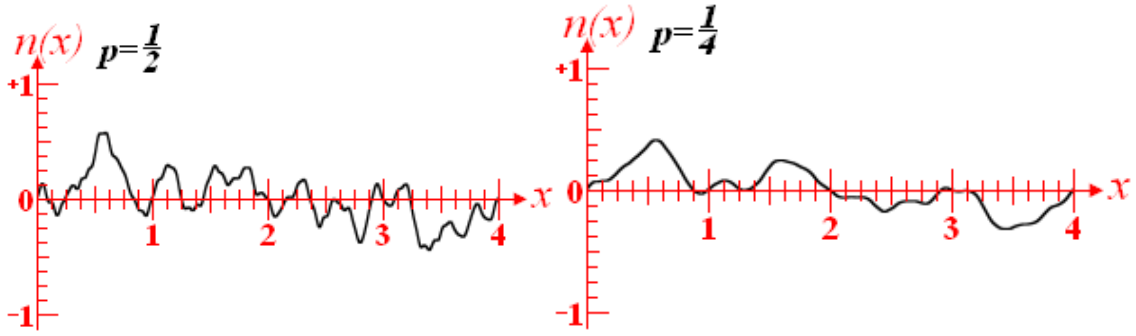

Рисунок 3.12 – Графік залежності шуму від стійкості при значеннях 0.5 та 0.25 відповідно.

Зміна будь-якого параметру відразу супроводжується оновленням процедурно-згенерованого ландшафту в реальному часі. Також, завдяки графічному інтерфейсу (див. рис. 3.13), користувач має додаткові можливості:

- Переглянути кількість кадрів за секунду;
- Час виконання одного фрейму оновлення основного потоку;
- Переглянути кількість чанків, що були згенеровані;
- Переглянути позицію чанку, в якому в даний момент знаходиться користувач;
- Змінити колір фону.

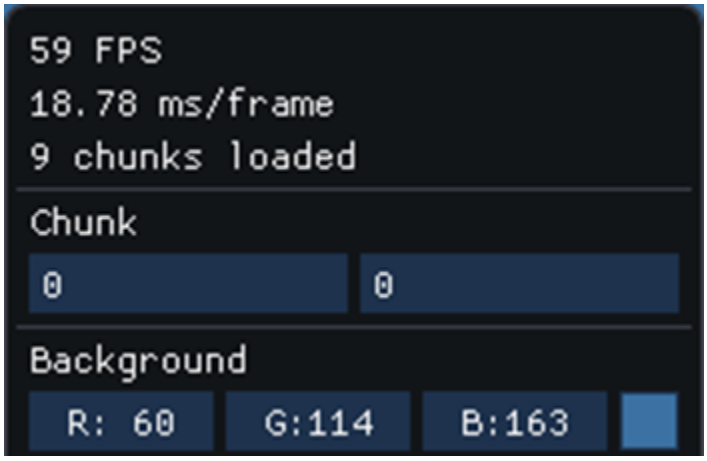

Рисунок 3.13 – Інформаційна частина графічного інтерфейсу

Також, користувач має можливість вільно переміщуватись для перегляду процедурно-згенерованого ландшафту (див. рис. 3.14).

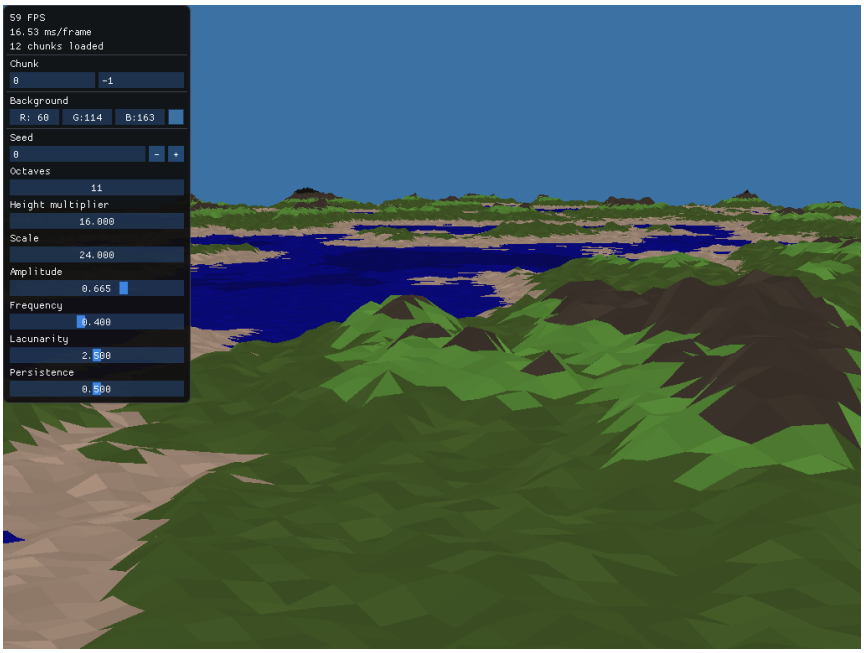

Рисунок 3.14 – Вікно програми для попереднього перегляну процедурно-згенерованого ландшафту.

3.6. Можливі способи покращення

В майбутньому, планується покращувати реалізований алгоритм процедурної генерації.

Для того щоб цього досягти, планується реалізувати динамічне визначення рівня деталізації (англ. Level of detail) згенерованих чанків. Поняття рівня деталізації [10] (далі LOD) означає зменшення складності тривимірної моделі. Зменшення LOD підвищує ефективність рендерингу шляхом зменшення навантаження на етапах графічного конвеєра. Зменшення візуальної якості моделі часто залишається непоміченими через невеликий вплив на зовнішній вигляд об'єкта, коли він знаходиться далеко чи рухається швидко.

3.6.1. Рівень деталізації

Дискретний LOD

Основою дискретного LOD (далі DLOD) є те, що для представлення одного й того ж об’єкту використовуються різні моделі (див. рис. 3.15).

Отримання таких моделей досягається використанням різноманітних методів спрощення тривимірного об’єкту. Недоліком такого способу є відсутність плавного переходу між рівнями LOD.

Рисунок 3.15 – Деталізація сфери в залежності від відстані до спостерігача.

Безперервний LOD

На відміну від DLOD, Безперервний LOD (далі CLOD) реалізує інший підхід до створення спрощеної геометрії. Тим часом, як DLOD створює бажані рівні деталізації під час запуску програми, CLOD створює їх по необхідності під час виконання [14]. Перевагами CLOD є:

- Краща деталізація;
- LOD визначається більш досконало, а не вибирається із скінченної кількості попередньо заданих варіантів;
- LOD використовує мінімальну кількість полігонів;

Ієрархічний LOD

Оскільки апаратні засоби орієнтовані на великі обсяги полігонів, тому велика кількість низько-полігональних об’єктів може негативно вплинути на швидкість відображення. Ієрархічний LOD (далі HLOD) уникає дану проблему шляхом групування різних об'єктів. Це забезпечує більш високу ефективність [11].

Традиційні алгоритми генерації LOD працюють лише на одному об’єкті, і тому мають меншу похибку при створенні спрощеної геометрії. Оскільки HLOD комбінує об’єкти які належать до однієї групи, таким чином створюючи спрощену модель, збільшується похибка та дещо спотворюється кінцевий результат (див. рис. 3.16).

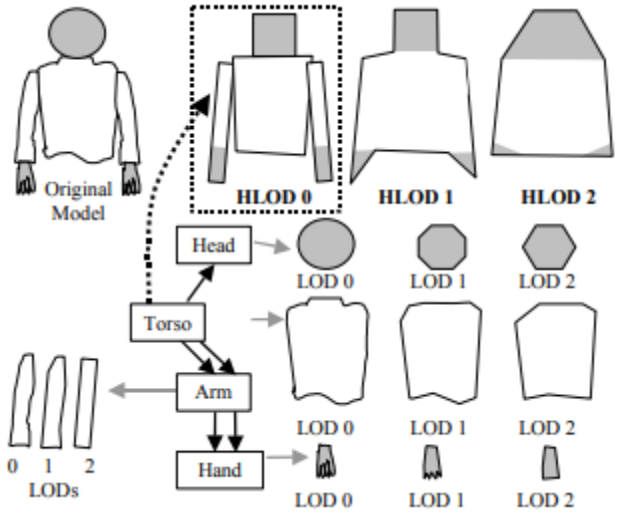

Рисунок 3.16 – Візуалізація моделі використовуючи DLOD та HLOD. Сірі стрілки вказують на DLOD. Чорні та пунктирні вказують на HLOD.

Залежний від спостереження LOD

Залежний від спостереження LOD (далі VDLOD) відрізняється від інших видів LOD тим, що один об'єкт може мати різні рівні деталізації (див. рис. 3.17) [14]. Особливістю VDLOD є те:

- Краща деталізація порівняно з іншими видами LOD;
- Менше використання пам'яті;
- LOD генерується на льоту.

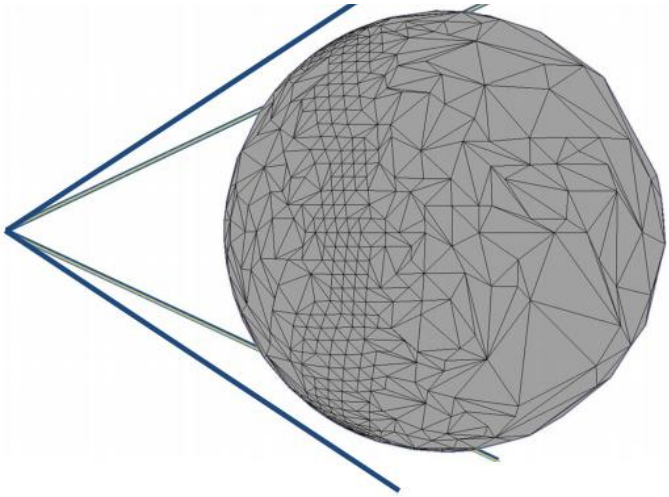

Рисунок 3.17 – Видимі частини мають більшу деталізацію порівняно з невидимими.

3.6.2. Паралельні обчислення

Деякі з останніх процесорів Intel підтримують нове сімейство векторних інструкцій AVX-512. Ці інструкції виконуються блоками по 512 біт. Перевага апаратної підтримки таких великих інструкцій в тому, що за один такт процесор обробляє більше даних. Теоретично, використовуючи інструкції AVX-512, можна прискорити алгоритм процедурної генерації ландшафту який реалізований за допомогою AVX2 ще у 8 разів.

ВИСНОВКИ

Отже, процедурна генерація – інструмент, що дозволяє працювати не тільки з контентом. Даний механізм також застосовується в естетичних цілях, щоб викликати відчуття відчуженості, невідомості або надати дизайну певний відтінок. Процедурно-згенерований контент може виконувати різні функції — адаптації анімаційних систем; підготовки макетів підземель; створення численних рівнів, незнайомих ландшафтів, нескінченних комбінацій геймплею, вдалого музичного супроводу і не тільки.

Важливо розуміти що це ефективний інструмент вирішення проблем, для створення величезної кількості рівнів, елементів дизайну персонажів – і таким чином не бути схожими один на одного. Не варто забувати також про те, що різноманітність це не завжди показник високої якості ігри.

В ході виконання даного дипломного проекту було проаналізовано ряд алгоритмів для процедурної генерації ландшафту та створено програмний застосунок за допомогою модифікованого існуючого алгоритму.

У першому розділі було проаналізовано існуючі застосунки процедурної генерації та існуючі алгоритми. Були розглянуті особливості існуючих рішень. Детальний аналіз показав, що жодне з існуючих рішень не підходить оскільки не задовільняє всім потребам.

Другий розділ дипломного проекту присвячено опису процедурної генерації ландшафту, особливостям та недолікам, які можуть виникнути під час розробки. В даному розділі також було наведено та проаналізовано декілька прикладів використання процедурної генерації.

В останньому розділі було описано весь процес розробки, принцип роботи реалізованого алгоритму та етапи модифікації. Також було описано процедурне текстурування та можливі способи покращення вже реалізованого алгоритму.

Поставлена задача була виконана успішно. Даний проект вже готовий до інтеграції в процес розробки ігрових додатків. Також варто зазначити, що даний проект досить легко модифікувати та кастомізувати функціонал під потреби розробників.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A Survey of Procedural Noise Functions, Computer Graphics Forum, Volume 29 (2010), Number 8, pp 2379—2600.
2. Journal of Computer Graphics Techniques Vol. 3, No. 2, 2014.
3. Ken Perlin's SIGGRAPH 2002 paper: Improving Noise. – Режим доступу: http://staffwww.itn.liu.se/~stegu/TNM022—2005/perlinnoiselinks/paper445.pdf
4. Mastering C++ Multithreading, Maya Posch, 2017.
5. The OpenGL® Graphics System: A Specification (Version 3.3 (Core Profile) — March 11, 2010), 2010.
6. Intel® C++ Intrinsic Reference. – Режим доступу: https://software.intel.com/sites/landingpage/IntrinsicsGuide
7. The Book of Shaders by Patricio Gonzalez Vivo and Jen Lowe. – Режим доступу: https://thebookofshaders.com
8. Ebert et al: Texturing and Modeling A Procedural Approach. Morgan Kaufmann, 2003.
9. Pietroni, Nico; Cignoni, Paolo; Miguel A., Otaduy; Roberto, Scopigno (2010). "A survey on solid texture synthesis".
10. OpenGL Programing Guide, 2nd Edition, 1998. – Режим доступу: http://people.cs.clemson.edu/~dhouse/courses/405/notes/OpenGL-mipmaps.pdf
11. ACM Siggraph Computer Graphics. Vol. 23. No. 3. ACM, 1989. – Режим доступу: http://www.heathershrewsbury.com/dreu2010/wp-content/uploads/2010/07/TheSynthesisAndRenderingOfErodedFractalTerrains.pdf
12. Julian Togelius, Emil Kastbjerg, David Schedl, Georgios N. Yannakakis. What is Procedural Content Generation?: Mario on the Borderline

13. English Dictionary. Brian Kariger; Daniel Fierro. May 14, 1995. – Режим доступу: https://www.dictionary.com/browse/high-level-language
14. Level of Detail: A Brief Overview. Daniel G. Aliaga. Fall 2010. – Режим доступу: https://www.cs.purdue.edu/homes/aliaga/cs535-10/lec-lod.pdf
15. Almasi, G.S. and A. Gottlieb (1989). Highly Parallel Computing. Benjamin-Cummings publishers, Redwood City, CA.