Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів»

Виконав: студент 2-го курсу, групи 2КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Волинець Б.А.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О.В.
(прізвище та ініціали)
«21» грудня 2022 р.

Опонент: к.т.н., доцент каф. АІІТ
Овчинніков К. В.
(прізвище та ініціали)
«21» грудня 2022 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
14. 09 **2022 року**

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Волинець Богдан Андрійович
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів

керівник роботи_ к.т.н., доцент кафедри КН Сілагін О.В.
затверджені наказом вищого навчального закладу від "14" 09 2022_ року №283
2. Строк подання студентом роботи 18 листопада 2022 року
3. Вихідні дані до роботи:
   Мова об'єктно-орієнтованого програмування, використання процедурних фрактальних шумів, параметри регулювання: частота, амплітуда, октава, Кількість октав – не більше 4, кольоровий інтерфейс, мова інтерфейсу – англійська, українська, відповідність стандарту ODBS.
4. Зміст текстової частини:
   вступ, обґрунтування доцільності розробки інформаційної технології, обґрунтування та вибір методів розв'язання задачі; моделювання архітектури інформаційної технології, проектування інформаційної технології; реалізація інформаційної технології, аналіз результатів тестування, економічна частина,
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
   Загальна структурна схема інформаційної технології, схема алгоритму роботи додатку, приклад роботи інформаційної технології, схема взаємодії основних модулів інформаційної технології, UML- діаграма еомпонентів, діаграма послідовності, загальний вигляд інтерфейсу користувача, приклад роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконан прийня
1-3	Сілагін О.В., к.т.н., доц. каф. КН	14.09.2022	14.11.202
4	Бурєннікова Н.В., д.е.н., проф. каф. ЕПВМ	13.09.2022	13.10.202

7. Дата видачі завдання 14. 09 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ етапу	Назва етапу	Термін виконання		Очікувані результ
		початок	кінець	
1	Аналіз сучасного стану розвитку технологій для генерації ландшафтів за допомогою процедурних шумів	14.09.2022	1.10.2022	Аналітичний огляд літературних джер задачі досліджень, розділ 1
2	Обґрунтування методу розв'язання задачі, обґрунтування вибору інструментів технічної реалізації	2.10.2022	16.10.2022	Моделі, розділ 2
3	Проектування інформаційної технології. Програмна реалізація інформаційної технології. Аналіз результатів тестування	17.10.2022	07.11.2022	розділ 2, 3
4	Підготовка економічної частини	08.11.2022	21.11.2022	розділ 4
5	Апробація та/або впровадження результатів дослідження	23.11.2022	01.12.2022	тези доповідей/ак впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.12.2022	14.12.2022	Пояснювальна записка, графічний матеріал, презента

Студент _____ (підпис) Волинець Б.А.

Керівник роботи _____ (підпис) Сілагін О.В.

АНОТАЦІЯ

УДК 004.8

Волинець Б.А. Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - комп'ютерні науки. Вінниця: ВНТУ, 2022. 100 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 18; табл. 3.

Дана магістерська кваліфікаційна робота присвячена розробці технології для генерації ландшафтів за допомогою процедурних шумів. Ця технологія спроектована та реалізована в учбовому додатку для генерування 3D ландшафтів за допомогою алгоритмів «Шум Перліна», з метою демонстрації можливостей цих алгоритмів, графічних бібліотек OpenGL та DirectX12 і візуальної компонентної бібліотеки GLScene (із додатком ігрового ядра) по побудові ландшафту. Додаток створено в програмному середовищі Delphi.

Графічна частина складається з 7 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 288241,25 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника 2 роки та економічний ефект для споживача при використанні даної розробки.

Ключові слова: генерації 3D ландшафтів, процедурні шуми, алгоритм «Шум Перліна», графічні бібліотеки OpenGL та DirectX12.

ABSTRACT

Volynets B.A. Information technology for generating landscapes using procedural noises. Master's qualification thesis on specialty 122 - computer science, educational program - computer science. Vinnytsia: VNTU, 2022. 100 p.

In Ukrainian speech Bibliography: 21 titles; Fig.: 18; table 3.

This master's thesis is devoted to the development of technology for generating landscapes using procedural noises. Existing solutions and their functionality were considered and analyzed. This technology is designed and implemented in an educational application for generating 3D landscapes using Perlin Noise algorithms, with the aim of demonstrating the capabilities of these algorithms, OpenGL and DirectX12 graphics libraries, and the GLScene visual component library (with a game core application) for landscape construction. The application was created in the Delphi software environment.

The graphic part consists of 7 posters.

In the economic section, the amount of costs for the development and production of a new technical solution is calculated, which is 288241,25 hryvnias, the estimated amount of costs for each of the cost items is predicted, the net profit, the payback period for the manufacturer 2 years and the economic effect for the consumer when using this development are calculated.

Keywords: generation of 3D landscapes, procedural noises, Perlin Noise algorithm, OpenGL and DirectX12 graphic libraries.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Природні об’єкти, створені засобами векторної 3D-графіки завжди виглядають штучними і далекими від фото-реалістичності фотографій. Одна з причин цього полягає в її нездатності описати форму хмари, гори, берегової лінії або дерева. Хмари - не сфери, гори - не конуси, берегові лінії - не кола, деревна кора не гладка, блискавка поширюється не по прямій. Багато природних об'єктів настільки іррегулярні і фрагментовані, що в порівнянні зі стандартною геометрією Евкліда природа має не просто більшу складність, а складність зовсім іншого рівня .

Роль фракталів в машинній графіці сьогодні досить велика. Вони приходять на допомогу, наприклад, коли потрібно, за допомогою декількох коефіцієнтів, задати лінії і поверхні дуже складної форми. З точки зору машинної графіки, фрактальна геометрія незамінна при створення штучних хмар, гір, поверхні моря. Фактично знайдений спосіб легкого подання складних неевклідових об'єктів, образи яких дуже схожі на природні.

В магістерській роботі розроблений учбовий програмний додаток для генерування ландшафтів за допомогою процедурних шумів, та з метою демонстрації можливостей цієї технології, бібліотеки OpenGL і візуальної бібліотеки GLScene по побудові ландшафту.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напряму наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей програмних додатків для

генерування ландшафтів із використанням процедурних шумів. Для цього потрібно вирішити наступні задачі.

Задачі дослідження:

1. Проаналізувати сучасні методи, алгоритми та технології, пов’язані з генерацією ландшафтів.
2. Обґрунтувати вибір методів та етапів для створення інформаційної технологія для генерації ландшафтів за допомогою процедурних шумів.
3. Розробити інформаційну технологія для генерації ландшафтів за допомогою процедурних шумів.
4. Реалізувати програмний додаток для генерації ландшафтів.
5. Протестувати програмний додаток для генерації ландшафтів;
6. провести аналіз та розрахунки рівня науково-економічного ефекту;

Об’єктом дослідження є процеси генерації ландшафтів на основі процедурних шумів.

Предметом дослідження є алгоритми та програмне забезпечення, для генерації ландшафтів на основі процедурних шумів.

Методи дослідження - методи генерації процедурних фрактальних шумів, теорія геометричних та алгебраїчних фракталів, методи та підходи до розробки учбово-орієнтованих програмних додатків, методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному:

Розроблено інформаційну технологію генерації ландшафтів на основі інформаційної моделі процедурного шуму, що дозволяє розширити функціональні можливості навчально-орієнтованих програмних додатків.

Практичне значення одержаних результатів полягає у наступному:

Здійснено програмну реалізацію навчального додатку для дослідження процесу генерації ландшафту на основі Шуму Перліна.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація роботи.

Робота апробувалась на Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації 2022 року, доповідь «Застосування алгоритму «Шум Перліна» в учбовому додатку для генерації фрактальних ландшафтів» [1];

Публікації: електронні тези науково-технічної конференції факультету інтелектуальних інформаційних технологій та автоматизації 2022 року [1].

1 АНАЛІЗ ЗАСОБІВ ФРАКТАЛЬНОЇ ГРАФІКИ ДЛЯ ГЕНЕРАЦІЇ ЛАНДШАФТІВ

1.1 Фрактали, та їх застосування в сучасних технологіях

Поняття "фрактал" походить від латинського слова fractus - "зламаний, розбитий" [2]. Це визначення фракталів було запропоноване французьким математиком, професором Сорбони, Бенуа Мандельбротом в 1975 році, в книзі "Фрактальна геометрія природи". Власне, він і зараз є провідним дослідником та керіником наукової школи в галузі фрактальної геометрії. Цей термін вживають, коли об’єкт володіє якими-небудь з наступних властивостей:

- проявляє ознаки повної або наближеної самоподібності;
- володіє складною структурою на всіх шкалах, при цьому збільшення масштабу не веде до її спрощення, як це зазвичай відбувається із звичайними фігурами (де фрагмент, що розглядається в дуже збільшеному масштабі, нагадуватиме пряму);
- володіє дробовою метричною розмірністю або метричною розмірністю, що перевершує топологічну;
- побудований за допомогою рекурсивних процедур (рекурсія - визначення об'єкту з використанням раніше визначених).

Хвилі, хмари, листя та крони дерев, ландшафти, кровоносні судини, капіляри рослин - все це приклади фракталів, тому поява програм фотореалістичного відображення природних ландшафтів - річ більш ніж закономірна [3].

Аналіз ринків показав, що фрактали стали популярним інструментом у трейдерів для аналізу та прогнозування стану біржових ринків.

У фізиці фрактали застосовують при моделюванні нелінійних процесів, таких, як турбулентний рух рідин, процеси адсорбції-дифузії, хмари, полум'я і тому подібне. В нафтохімії фрактали використовуються при створенні різних

пористих матеріалів. У біології їх використовують для опису систем внутрішніх органів (системи лімфовузлів та кровоносних судин), моделювання популяцій і т.д.

Використання фрактальної геометрії при моделюванні та проектуванні пласких настінних антенних пристроїв було вперше застосоване Натаном Коеном, американським інженером, який тоді жив в центрі Бостона, де діяла заборона на установку зовнішніх антенних пристроїв на споруди. Він вирізав антену з алюмінієвої стрічки у формі кривої Кох і наклеював її на листок паперу. Експерименти показали високу ефективність таких антен, що привело до заснування власної компанії і налагодження серійного випуску портативних антен, що широко використовуються у мобільномй зв’язку.

Фрактали широко застосовуються у стиснення зображень. Компанією Microsoft створені алгоритми фрактального стиснення зображень. Вони засновані на ідеї про те, що замість самого зображення можна зберігати стискаюче відображення, для якого це зображення (або деяке близьке до нього) є нерухомою крапкою. Один з варіантів даного алгоритму був використаний при виданні електроних книг.

Система призначення IP-адреси в DNS-серверах мережі Netsukuku використовує фрактальне стиснення інформації для компактного збереження інформації про вузли мережі. Окремий вузол мережі Netsukuku зберігає тільки 4 Кб інформації про стан та походження сусідніх вузлів, при цьому нові вузли підключається до загальної мережі без потреби в центральному регулюванні роздачі IP-адресів, що, характерно для мережі Інтернет. Таким чином, принцип фрактального стиснення інформації гарантує повну децентралізацію, і відповідно, максимально стійку роботу всієї мережі.

1.2 Види та класифікація фракталів

Множина Кантора

Німецький математик Георг Кантор (1845-1918) в 1883 році побудував множину, яка носить його ім'я (множина Кантора) [3]. Так множину Кантора можем побудувати таким чином: одиничний відрізок розріжемо на 3 рівних відрізка; середній відрізок викидається; два відрізки, що залишилися, знову розрізаються, кожен на 3 рівних відрізки, і середні їх частки викидаються. Відрізки, що залишиись, знову розрізаються, кожен на 3 рівних відрізки, і середні частки викидаються і так далі. Як результат одержимо множину Кантора.

Крива Кох

Німецька вчена математик Хельга фон Кох у 1904 році побудувала криву, яка зараз носить її ім'я (крива Кох) [3]. Ця крива представлена на рисунку 1.1. Побудова починається з вирізання у прямої одиничного відрізка. Цей відрізок прямої ділиться на три однакових частини. Частина посередині видаляється, а на її місці зводиться рівносторонній трикутник. У результаті одержуєм ламану лінію, яка складається з 4 відрізків, кожен з яких рівний 1/3.

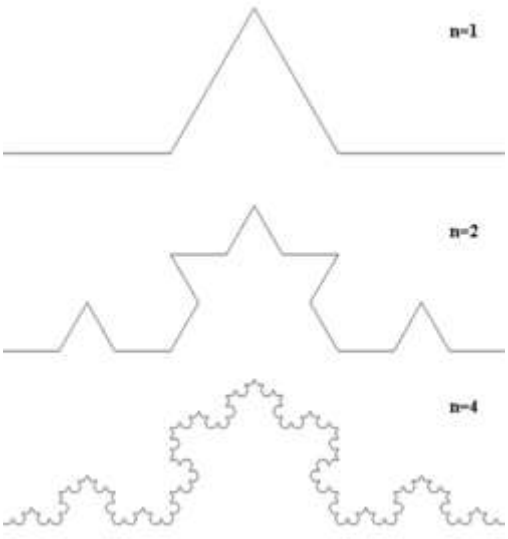


Рисунок 1.1 - «Крива Кох»

Далі, кожен з 4 складових знову ділиться на 3 рівних відрізки, на відрізках, розташованих посередині, зводяться рівносторонні трикутники, з яких середні частинкии видаляються. Ця операція повторюється безкінечно. У результаті крива Кох стає дуже порізаною. Якщо цей процес проводити нескінченно, то одержимо неперервну криву, що не диференціюється ні в одній точці, і ця крива одержує ненульову "площу". Щоб в цьому впевнитися, вирахуємо фрактальну розмірність кривої Кох. На самому початку ми маємо один відрізок завдовжки 1, який можна покрити другим відрізком з довжиною, рівною 1, тобто e = 1 і N(e) = 1. На другому етапі ми відрізаємо 4 рази, кожен з цих відрізків довжиною 1/3, тому для покриття цих відрізань потрібно 4 відрізання завдовжки 1/3, тобто e=1/3 і N(e)=4.

Таким чином, ми вперше стикаємося з ненульовою площею кривої. Звична (або топологічна) розмірність канторової множини рівна 0, а ось фрактальна, як бачимо, нулю не рівна - вона строго більше. Це і є, властивість фрактала.

Фрактали прийнято ділять на класи або види. Кожен вид має свою назву. Розглянемо, для прикладу, геометричні фрактали. Один з найвідоміших представників цього виду - це килимок Серпінського [3] який представлений на рисунку 1.2. Побудова його складається з кількох кроків: ми беремо звичайний рівносторонній трикутник і в середині його вирізуємо отвір у вигляді такого ж рівностороннього трикутника, тільки перевернутого і меншого в чотири рази. Тепер в кожному з кутів у нас з'являться маленькі рівносторонні трикутники. Зробимо з ними те ж саме: в середині кожного виріжемо маленькі трикутники. І так далі, до безкінечності.

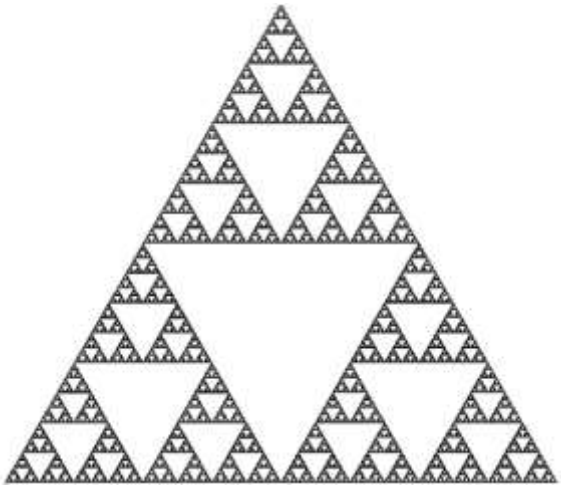


Рисунок 1.2 - Побудова геометричного фрактала - килимка Серпінського

За цим принципом будуються всі геометричні фрактали: берем якусь геометричну фігуру і починаємо застосовувати зміни до неї, а потім до її складових частин, такі геометричні перетворення виконується багато разів. Для фрактала, строго кажучи, процедура зміни повинна повторюватись нескінченну кількість раз.

Алгебраїчні фрактали

Один з методів побудови алгебраїчних фракталів виглядає так. Ви берете якусь формулу, підставляєте в неї число і отримуєте певний результат. Потім в цю ж формулу підставляєте результат і отримуєте наступне число. Багато раз повторимо цю процедуру. У математиці таке повторення називається ітераційним процесом. Як результат, одержуєм набори чисел, які є точками фрактала. Незважаючи на те, що ці формули можуть бути досить простими, фігури, які їм відповідають на графічних моделях, виходять вражаючої складності і краси [4]. Так виглядає, наприклад, фрактал «папороть», зображений на рисунку 1.3.

Рисунок 1.3 – Алгебраїчний фрактал "папороть"

Найпоширенішим видом фрактальної комп’ютерної графіки є стохастичні фрактали [4]. Їх одержують, змінюючи в ітераційному процесі деякі виділені параметри рандомно. За допомогою стохастичних фракталів можна створити такі віртуальні природні об'єкти, як ландшафти, порізані берегові лінії, рельєф місцевості, вогонь, хвилі, хмари, та багато чого іншого. Через те стохастичні фрактальні моделі широко застосовують в комп'ютерних іграх або тренажерах для створення віртуальної реальності, яку вже важко відрізнити від дійсної.

Цікавим є дослідження Бенуа Мандельбротом перетворень комплексної площини. Він розглядав згенеровані траєкторії крапок, що одержують при такому перетворенні, і виявляв залежність вихідної картини від параметрів пертворення. При цьому Мандельброт намагався встановити ті межі на площині, де крапки не "тікають" на безкінченність, а створена в ітераційному процесі послідовність залишається в обмеженій області. Він встановив, що одержані значення таких параметрів створюють деяку зв'язану множину з химерною межею, і одержана форма основної частини множини повторюється безкінечно в різних масштабах. Ця множина і була названа множиною Мандельброта. Вона зображена на рисунку 1.4.

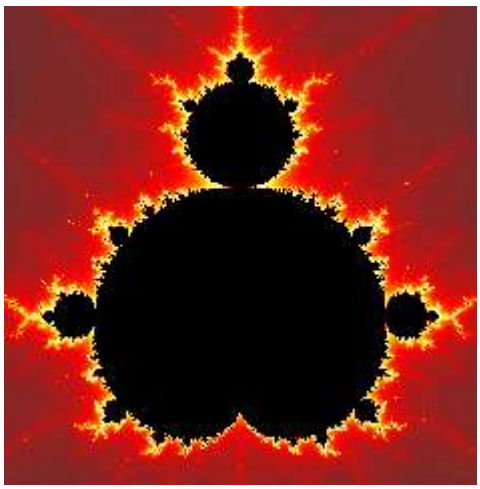


Рисунок 1.4 - Множина Мандельброта

1.3 Аналіз можливостей процедурних шумів для генерації ландшафтів

В підрозділі 1.2 ми зазначали, що фрактали, які находять застосування в сучасних технологіях, можуть бути поділені на 3 класи: генометричні, алгебраїчні та стохастичні. Останні якраз і знайшли широке застосування в моделюванні віртуальної реальності. В свою чергу, стохастичні фрактали (за методологією їх створення) можуть бути поділені на два типи. Перший тип передбачає, що у об’єкта з певною структурою деякі атрибути можуть змінюватись, в певних межах, рандомно. Назвемо їх структури з регульованою хаотичністю. Другий тип характеризується повним початковим хаосом, у який вводяться певні зауономірності, - хаос структурується. Цей тип називається процедурними шумами. У даному підрозділі описуються підходи до генерації шумів, докладно розглядаються найбільш популярні методи генерації процедурних шумів і приводиться відповідна термінологія. На рисунку 1.5 зображено ландшафт, повністю побудований за допомогою фрактальних процедурних шумів.

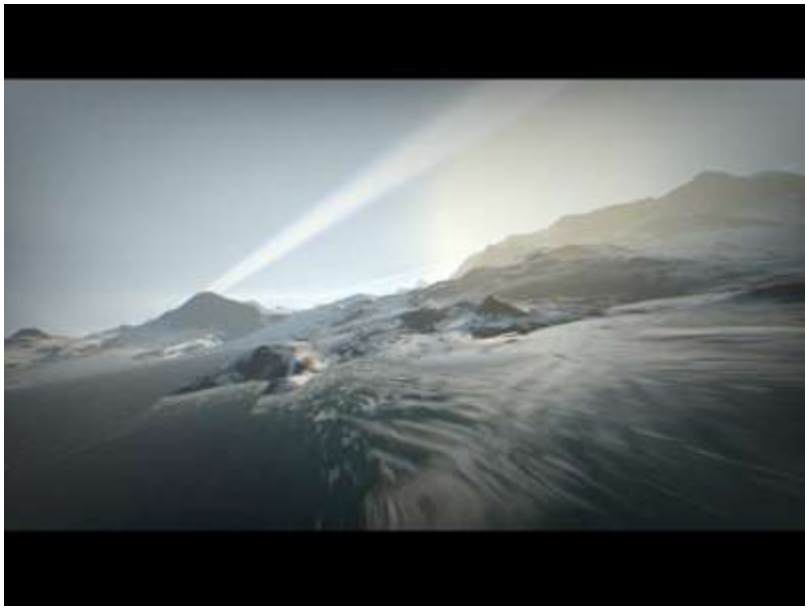


Рисунок 1.5 - Ландшафт і текстура згенеровані з використанням процедурного шуму

Визначення шуму: Шум - безладні коливання різної фізичної природи, що відрізняються складністю тимчасової і спектральної структури.

Надалі в роботі під випадковими числами будемо найчастіше розуміти, без явної вказівки на це, псевдовипадкові числа, як такі, що генеруються детермінованим способом, але на обмежених часових інтервалах сприймаються як випадкові, зокрема з деякого набору статистичних тестів.

Джерела шуму

У програмуванні часто зустрічаються 2 об'єкти, пов'язаних з генерацією шуму [2, 3]:

1. Генератор випадкових чисел (ГВЧ (англ. RNG - random number generator)), або ГПВЧ - генератор псевдовипадкових чисел (в даній роботі під ГВЧ мається на увазі саме ГПВЧ). При виклику ГВЧ видає наступний елемент в послідовності (псевдо) випадкових чисел.

2. Шумова функція (англ. Noise function). Функція, яка аргументу протиставляє результат, що веде себе, в деякому сенсі, як шум. Аргумент може бути як дискретним, так і безперервним, одно- або багатовимірним.

Для безперервного випадку частіше за все нас цікавить когерентна шумова функція [4]:

Когерентний шум генерується функцією когерентного шуму, що має три важливі властивості:

- передача тієї ж вхідної величини завжди поверне таке ж вихідне значення;
- невелика зміна вхідного значення призведе до невеликої зміни вихідного значення;
- велика зміна вхідного значення призведе до випадкової зміни вихідного значення.

Функція n-мірного когерентного шуму вимагає n-розмірного вхідного значення. Його вихідне значення завжди є скаляром.

Прикладом ГВЧ є функція rand стандартної бібліотеки C. Для багатьох цілей дана конкретна функція є невдалою, так як її стан (визначає наступний елемент) - невидима глобальна змінна. Відповідно підсистеми, що її використовують будуть впливати один на одного.

Більш зручними в цьому сенсі є об'єкти генераторів, які зберігають свій стан, наприклад генератори, що визначені в заголовочному файлі <random> стандартної бібліотеки C ++ (<random> присутній в стандартній бібліотеці починаючи з C ++ 11).

Шумову функцію від дискретного аргументу концептуально можна реалізувати через ГВЧ, відкинувши відповідну кількість початкових значень, наприклад так:

```
float noise(int i)
{
   std::minstd_rand rng(0); // Initialize RNG with the internal state of 0.
   rng.discard((unsigned long long)i); // Advance the internal state by i steps.
   return std::uniform_real_distribution<float>(0.0f,1.0f)(rng);
}
```

Але з практичної точки зору це не дуже вдалий код, так як обчислення занадто повільні, причому час роботи залежить від аргументу. Тому зазвичай шумові функції обчислюють за допомогою явних формул.

Шумовими функціями часто зручно оперувати на концептуальному рівні, тому що вони є функціями в звичайному математичному сенсі (є "чистими функціями" в програмістської термінології), і маніпуляції з ними можна проводити за звичайними правилами. Для практичних цілей в багатьох випадках досить прямого використання ГВЧ для отримання послідовних значень.

"Кольори" шуму

Важливою характеристикою шуму є його спектр, тому що він сильно впливає на параметри шуму і при цьому надає зручний математичний опис. Під спектром, як правило, розуміють "спектральну щільність" - квадрат модуля (амплітуди) перетворення Фур'є, рідше - саму амплітуду (тому що амплітуда, в загальному випадку, є комплексною величиною, крім модуля, іноді, становить інтерес також і фаза).

Розглянемо білий шум - стаціонарний шум, спектральні складові цього шуму рівномірно розподілені на всьому діапазоні задіяних частот [5]. Називається він так за аналогією з білим світлом, що відповідно містить кольори всіх частот (в оптичному діапазоні).

Безперервний ідеальний білий шум (для якого два значення нескорельовані, якщо вони не одночасні) є фізично некоректним, так як має нескінченну потужність. Для дискретного білого шуму ця проблема відсутня.

Також, по аналогу з оптикою, використовують терміни рожевий (з переважанням низьких частот) і синій (з переважанням високих частот). Часто також користуються більш вузькими значеннями: червоний, рожевий, блакитний і фіолетовий [6].

Одним з методів отримання дискретного шуму з заданими спектральними характеристиками є добуток спектра дискретного білого шуму і певної спектральної обвідної (Fourier Spectral Synthesis): Нехай $\boldsymbol{X} = \{x_i\}$ - дискретний білий шум. Тоді шум з обвідної (в частотному просторі) $w(f)$ можна отримати як

$$\boldsymbol{Y} = \mathcal{F}^{-1}\{w(f) \cdot \mathcal{F}\{\boldsymbol{X}\}\}, \tag{1.1}$$

що виглядає як

$$y_j = \frac{1}{n}\sum_{k=0}^{n-1} w_k e^{2\pi i \frac{jk}{n}} \widetilde{x}_k, \tag{1.2}$$

де

$$\widetilde{x}_k = \sum_{j=0}^{n-1} e^{-2\pi i \frac{jk}{n}} x_j \tag{1.3}$$

Категорії процедурного шуму.

Різні методи генерації процедурного шуму доступні в [7]. Розглянемо стисло найбільш важливі.

Один метод – Перетворення Фур’є (Fourier Spectral Synthesis) ми вже розглянули.

Шуми на решітці (Lattice Noises) є найбільш популярними. В них задаються рандомні значення в вузлах решітки, виходячи з яких знаходиться значення в проміжних точках (будується безперервна шумова функція). Це найбільш відомі:

1. Чисельний шум (value noise) – передбачає, що в кожному вузлі решітки задане (детермінованим способом) псевдовипадкове число. Тоді значення шумової функції буде інтерполяція (лінійна або кубічна) між значеннями в межах того інтервалу, куди потрапляє аргумент. Види чисельного шуму показані на рисунку 1.6.

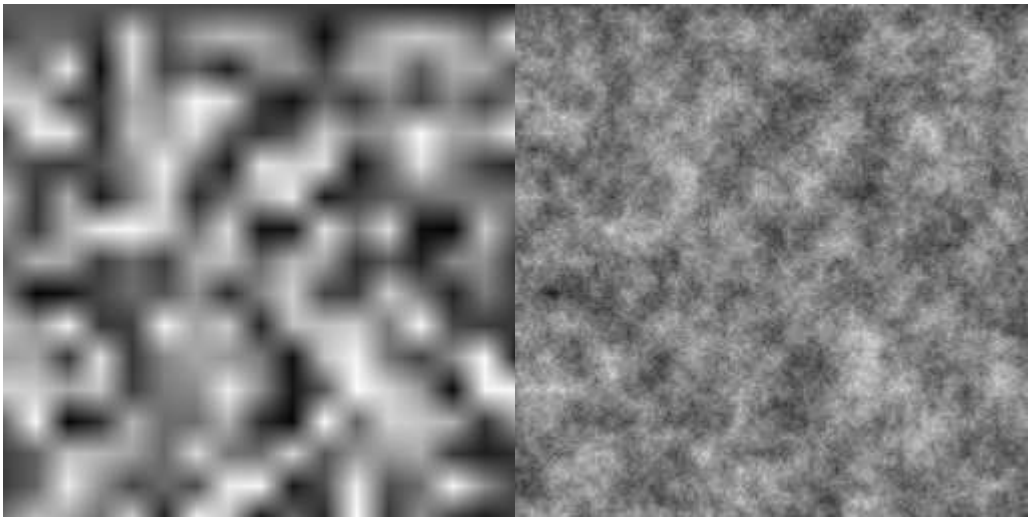


Рисунок 1.6 - Двовимірний чисельний шум. Зліва - одна октава, праворуч - 5 октав (коефіцієнт 0.73 та білінійна інтерполяція)

2. Метод градієнтного шуму (gradient noise) - для кожного вузла решітки задається (детермінованим способом) псевдовипадковий вектор (званий градієнтом). Значення шумової функції знаходять, виходячи зі значень в вузлах інтервалу і напрямків векторів. Найбільш популярним представником є шум Перліна.

Рисунок 1.7 - Двовимірний шум Перліна.

Метод зміщення середнього значення (Midpoint displacement method) Одна із найбільш популярних реалізацій цього метода - Diamond-square algorithm [8], зображений на рисунку 1.8.

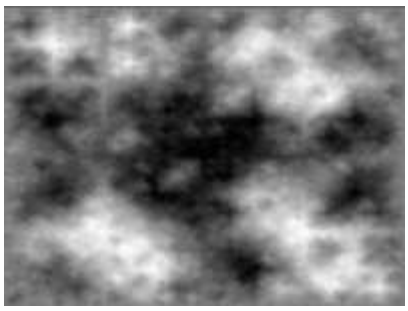


Рисунок 1.8 - Plasma fractal, згенерований Diamond-square algorithm

Вейвлет-шум (Wavelet noise)

Реалізується при генерації заповненого випадковим чином зображення R (білий шум), і вирахування з нього upsample (downsample (R)), прибираючи таким чином деталі, представлені в зменшеній роздільній здатності.

Метод розсіяного згорання (Sparse Convolution Noise).

Являє собою згортку деяким чином обраного ядра і пуассонівського шумового процесу (Poisson process noise). Пуассонівський шумовий процес - це множина нескорельованих по амплітуді імпульсів, розташованих в

рандомно вибраних точках. Таким чином, цей вид шуму являє суперпозицію ядер з центрами і інтенсивностями, що задаються рандомним набором імпульсів, на зразок інтерференційного зображення. Найбільш відомий представник - Gabor Noise [10].

Шум Уорлі (Worley noise).

Запропоновано Стівеном Уорлі в 1996 році в статті A cellular texture basis function. Являє собою діаграму Вороного n-го порядку для випадкового набору точок. Таким чином значенням функції є відстань до n-ї найближчої точки (відстань не обов'язково використовує евклідоу метрику, іноді застосовується, наприклад, манхеттенська). Часто використовується для створення пористих текстур (комбінація 1-го і 2-го порядків). Хороший опис ефективних алгоритмів побудови пористих текстур можна знайти в статтях [11, 12, 13].

Існують різні способи генерації набору точок для шуму Уорлі. Jittered grid (регулярна сітка з певним рандомно зміщенням вершин) є досить популярною, але призводить до анізотропії. В статті Уорлі був використаний метод з розрізанням простору на куби і генерацією для куба випадкових точок кількістю, що визначена розподілом Пуассона;

Більшість з цих методів генерують шум в досить вузькому діапазоні частот (іншими словами - з певним розміром деталей). Він може бути не дуже корисний сам по собі, але буде зручним будівельним блоком для поєднань, що створять більш цікавий шум.

Розповсюдженим підходом є сума октав шуму:

$$f_{\sum}(\boldsymbol{x}) = \frac{1}{\sum_{k=0}^{n-1} a^k} \sum_{k=0}^{n-1} a^k f(b^k \boldsymbol{x}) \tag{1.4}$$

Коефіцієнт a носить різні назви. Бібліотека libnoise використовує термін

Persistence, як і стаття Хьюго Еліаса , де згадується, що термін persistence щодо шуму був введений Бенуа Мандельбротом як обернена величина ($1/a$). Коефіцієнт bв libnoise названий Lacunarity, і в більшості випадків вибирається рівним 2.

Коефіцієнт a найчастіше вибирають в діапазоні [0; 1], хоча це і не обов'язково.

Використання суми октав для чисельного шуму і шуму Перліна широко відомо, але цей підхід також може бути застосований, наприклад, і для шуму Уорлі, з перспективними результатами.

Шум Перліна

Запропонований Кеном Перліном в 1983 році, Шум Перліна є прикладом градієнтного шуму.

Алгоритм реалізує шумову функцію в довільному діапазоні точності. Початкова вихідна реалізація була 3-мірної.

Оригінальний алгоритм визначення функції в заданій точці наступний:

1. На вузлах решітки (це точки з цілочисельними координатами) детермінованим способом задаються псевдовипадкові n-мірні одиничні (або однакові по довжині) вектори, так звані градієнти.
2. Для вузла знаходиться цілочисельна комірка (одиничний куб з цілочисельними вершинами), яка її містить.
3. Для кожного кута комірки створюється вектор з цього кута в точку, і обчислюється його скалярний добуток з градієнтом для цього кута.
4. Одержані скалярні добутки інтерполюють (може бути лінійна інтерполяція з вагами, що відповідають S-curve від дробної частини координат: (важливо, щоб 1-і похідні від функції інтерполяції дорівнювали 0 на краях). Така інтерполяція буде сепарабельною. З практичних міркувань, обчислюються n значень S-curve (по 1 для кожної координати) і 2 ^ {n} раз

проводимо лінійну інтерполяцію (2 ^ {n-1} на першій координаті, потім 2 ^ {n-2} - на 2-й для результатів попереднього кроку, і т. д.).

Легко показати, що якщо буде виконана умова рівності похідної 0 на краях, то градієнт отриманої скалярної функції в цілочисельних вузлах буде дорівнювати заданим в них векторам.

Незважаючи на те, що ця функція дещо повільніше інтерполяції чисельного шуму, що розглянутий у попередньому пункті, але вона дає більш якісний шум. Попри те, що шум Перліна не є ізотропним, його анізотропія набагато менше виділяється, ніж у чисельного шуму на сітці.

Public Domain реалізація 3-мірного шуму Перліна реалізує дещо іншу модифікацію алгоритму. У stb-версії випадковий вектор - одна з 12 можливих перестановок {± 1, ± 1,0}, обрана за допомогою хеш-функції від (цілочисельних) координат (revised Perlin noise) і процедура вибору теж дещо відрізняється.

Також шум Перліна реалізований в складі бібліотеки libnoise. Як легко бачити алгоритм має складність O (2 ^ n), де n - розмірність простору. Для високих розмірностей його швидкодія є дуже низькою. Для боротьби з цим Кен Перлін в 2001 році запропонував симплекс-шум, який розбиває простір на симплекси , а не на гіперкуби, що дає можливість знизити складність алгоритму до O (n ^ 2). Симплекс-шум підпадає під дію патенту (принаймні в США), але є альтернативні відкриті реалізації (напр. OpenSimplex noise). Стаття Перліна з описом алгоритму доступна за посиланням [14].

1.4. Обгрунтування та вибір алгоритму «Шум Перліна» для моделювання ландшафтів.

Сформулюємо ряд вимог, яким повинен відповідати шум для моделювання 3-d аморфних об’єктів:

- шум має бути когерентним;

- нести в собі ознаки фрактала;
- повинен бути процедурним по реалізації;
- повинен бути хаотичним і одночасно гладким;
- повинна бути регульованою ступінь хаотичності;
- процедури шуму повинні виконуватись в режимі реального часу;
- забезпечення генерації функції шуму в багатовимірному просторі.

Фрактальний шум або Шум Перліна, як правило, використовується в двомірній і тривимірній комп'ютерній графіці для створення таких візуальних ефектів, як дим, хмари, туман, вогонь і т. д. Він також може використовуватися як проста текстура, що вкриває геометричну модель. На протилежність від растрових текстур, шум Перліна є процедурним, і тому він не перевантажує пам'ять, але разом з тим виконання алгоритму передбачає використання обчислювальних ресурсів. Використання шуму Перліна також поширене в демосценах. Кпім того, на цьому алгоритмі працюють такі програми для створення та генерації ландшафтів, як Terragen, а також всі ігрові застосування, в яких є потреба хоч якось відобразити небо та згенерувати ландшафт [16, 17].

Алгоритм Шум Перліна був створений Кеном Перліном (англ.) в 1983 році і потім був названий на його честь. Перлін створював свій алгоритм, працюючи в Mathematical Applications Group Inc. У 1997 році Кен Перлін виборов нагороду Academy Award for Technical Achievement від Академії кінематографічних мистецтв і наук Америки за великий внесок в створення фільму «Трон» випуску 1982 року.

Використання Perlin Noise для моделювання ландшафту і хмар є найбільш поширеним способом через його можливості генерувати одно- дво- та багатовимірні об’єкти. Більш того, в алгоритм можна використати четверте часове вимірювання, дозволяючи програмі динамічно змінювати текстуру в часі. Також його характеризує високий ступінь реалістичності.

Як відомо, хмарна та ландшафтна поверхні володіють плавною структурою, але, в той же час, і хаотичною. Таким чином, для формування хмар і ландшафту потрібна якась хаотична функція, ядро якої, одночасно, підкоряється певному закону. Таким вимогам задовольняють функції Perlin Noise, які є своєрідною комбінацією шумової функції та інтерполяційної функції (рис. 1.9). Аргументами для шумової функції виступають цілі числа (integer).

а) шумова функція

б) комбінація шумової та інтерполяційної функції

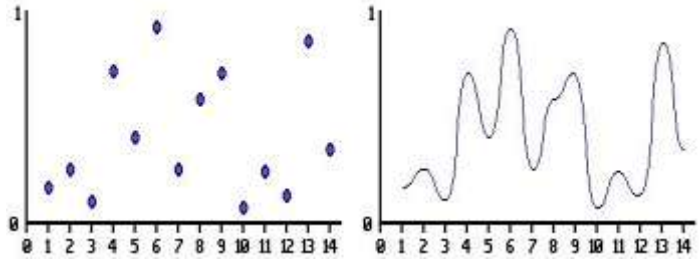


Рисунок 1.9 – Шум Перліна, як комбінація шумової функції та інтерполяції

Як результат послідовного застосування цих двох функцій виходить, як показано на малюнках, функція, що володіє гладкістю і в той же час достатнім ступенем хаотичності (рис. 1.10).

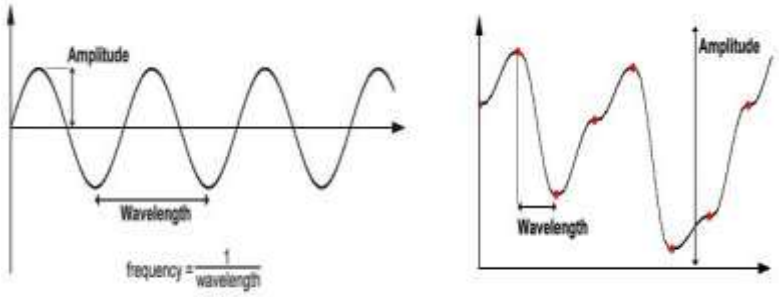


Рисунок 1.10 – Хвильові якості Шуму Перліна

Основними параметрами тут є амплітуда (amplitude) і частота (frequency). Амплітуда визначає висоту хвилі, частота є оберненою величиною від довжини хвилі, довжина хвилі - це, по відношенню до шумової функції, інтервал від однієї червоної позначки до іншої.

Але якщо ми фіксуємо амплітуду і частоту, то отримана функція все ще буде володіти достатнім ступенем рівномірності. Рішенням є наступне: ми будуємо кілька функцій з різними параметрами, а потім сумуємо їх.

Отримана в результаті функція тепер задовольняє всім необхідним умовам, що приведені на початку підрозділу 1.4.

1.5 Висновок до розділу 1

В першому розділі магістерської кваліфікаційної роботи зроблено детальний огляд та аналіз засобів фрактальної графіки, термінологія, історія їх виникнення та еволюція з точки зору використання в реалістичному моделюванні 3-D об’єктів. В результаті робимо висновок, що найкращим інструментом для цього є шумова фрактальна функція. В подальшому зроблено огляд та аналіз популярних методів генерації шумової функції та сформована низка вимог до них. Цим вимогам якнайкраще відповідає процедурний метод генерації шумової функції – Шум Перліна.

Вважаю за потрібне в наступному розділі провести математичне та програмне моделювання процедурного методу генерації «Шум Перліна» на предмет дослідження його використання в реалізації ландшафтів.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ГЕНЕРАЦІЇ ЛАНДШАФТІВ НА ОСНОВІ ПРОЦЕДУРНОГО ШУМУ

2.1 Моделювання процедурного Шуму Перліна

2.1.1 Моделювання в одновимірному просторі

Для кожного цілого числа x_i = 0,1,2,3,..., для початку виберемо випадкове дійсне число з діапазону [-1;1]. Це число означатиме тангенс кута нахилу, градієнт прямої, що проходить через x_i на числовій осі (рис. 2.1).

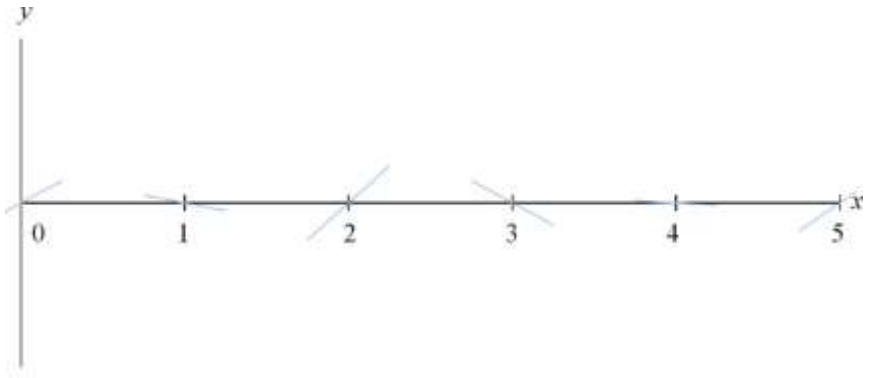


Рисунок 2.1 – Формування випадкового градієнта

Покажемо дві похилі лінії з кожного боку, що продовжують градієнти і складаються з точок (рис. 2.2).

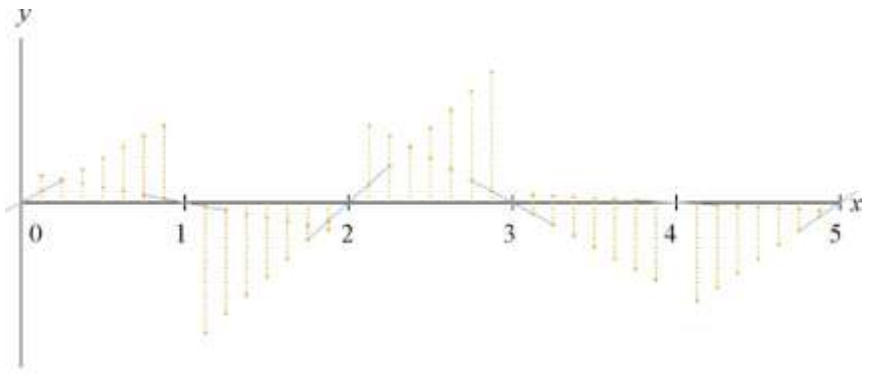


Рисунок 2.2 – Визначення точок перетину

Тепер вирішуєм питання: як можна пари точок для кожного x_i трансформувати в гладку криву? Очевидна відповідь – лінійна інтерполяція, так як це показано на рисунку 2.3

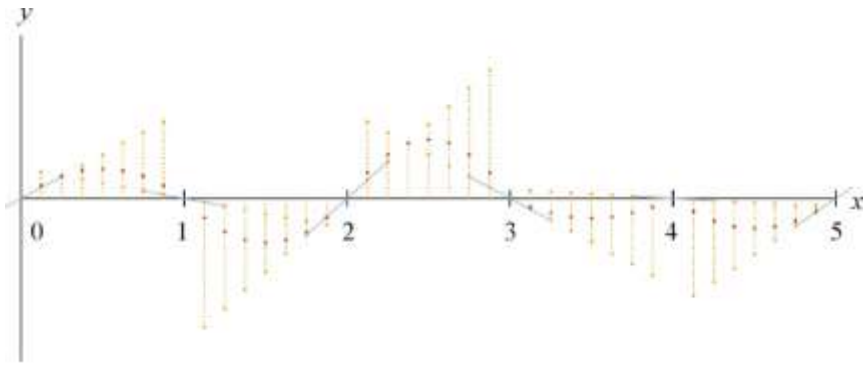


Рисунок 2.3 – Приклад застосування лінійної інтерполяції

Всі ці криві - параболи, задані рівнянням $y=(a-b)(x_i-x_i^2)$. Для цього випадку можна використати криві Безьє, що задані повторною лінійною інтерполяцією. Для квадратичної кривої Безьє маємо три опорні точки; при цьому ми інтерполюємо між першою, другою і третьою точками, для кубічної кривої Безьє маємо чотири опорні точки і будемо робити на одну ітерацію більше, і так далі. Одержані результати застосування кривих Безьє спостерігаємо на рисунку 2.4.

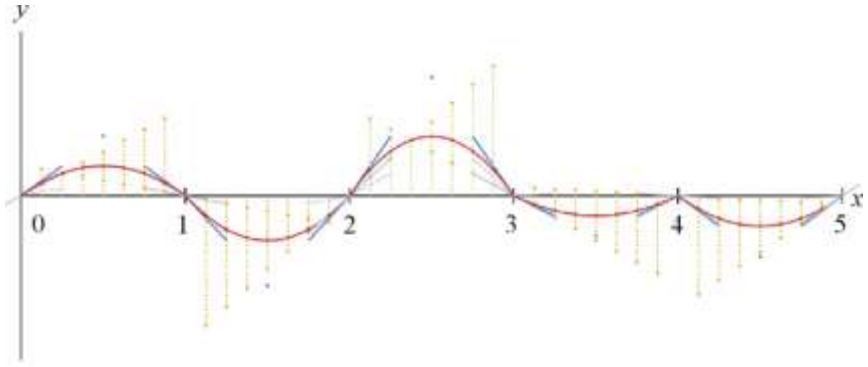


Рисунок 2.4 – Недоліки лінійної інтерполяції

На четвертій точці спостерігаємо злам. Це демонструє те, що лінійна інтерполяція для певних застосувань може бути неприйнятною. Уникнути зламів можемо застосувуючи функцію smoothstep. Це s-подібна крива, схожа на пряму y=x, але сплющена в нулі і одиниці. Посередині вона не сильно змінюється - 0.5 залишається 0.5, але вона досить сильно зміщує кінці. У точці 0.9 вона буде рівною 0.972.

Ця крива має четверту ступінь (2.1),

$$y=2(a-b)x^4-(3a-5b)x^3-3bx^2+ax, \qquad (2.1)$$

і на графіку позначена пунктиром (рис. 2.5).

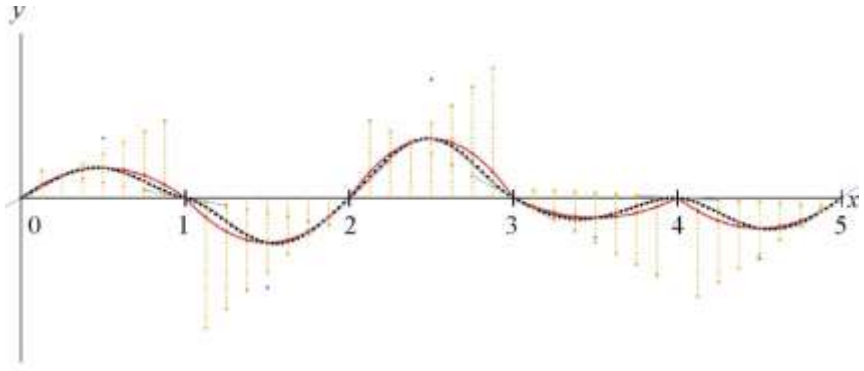


Рисунок 2.5 – Інтерполяція функцією smoothstep

Нові точки наближаються до квадратичних кривих, але в районі зламів вони ближче прилягають до вихідних градієнтів. Точно збігаються тільки середні точки, тому що smoothstep не міняє 0.5.

Для проведення експериментів можна взяти інтерактивний графік.

Використання октав

Для того щоб одержана в попередньому тексті крива стала фракталом, їй бажано надати ітераційну самоподібність. Це можна зробити, за допомогою октав. Поняття октави запозичено з музики, де частота кожної наступної

октави дорівнює попередній помноженій на 2. Можемо змінювати також амплітуду хвилі. Вона, як правило, пов’язана із частотою через константу або функцію, що одержала назву «persistans»

Створимо октавну криву Перліна (або ту ж саму), але вже з подвійною частотою - випадкові градієнти тепер будуть в точках x=0,0.5,1,.... Поділимо на два значення виходу і додамо до початкової кривої. Результат демонструється на рисунку 2.6.

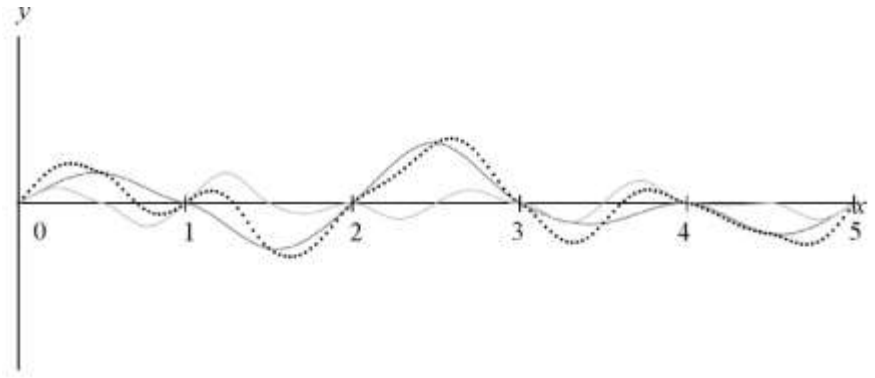


Рисунок 2.6 – Крива Перліна в дві октави

Цю операцію можна повторювати ще скільки завгодно раз, роблячи вдвічі менше попереднього кожен новий графік. Кожен окремий графік буде називатися октавою (в музиці кожна наступна октава має частоту в два рази більша за попередню). Для отримання ефекту «гористого краєвиду» буде достатньо 4-5 октав (рис. 2.7).

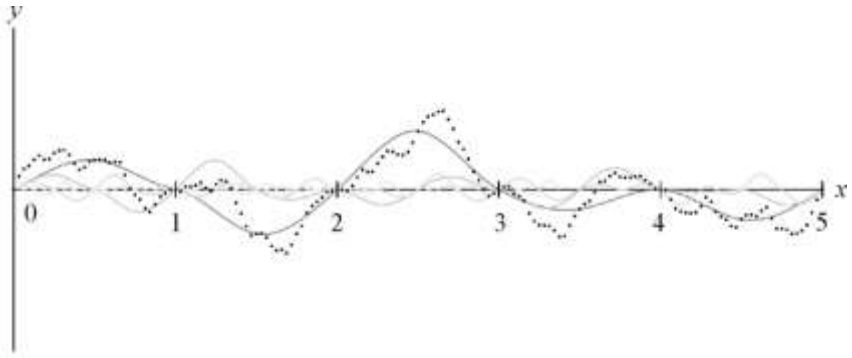


Рисунок 2.7 – Крива Перліна в 4-ох октавах

2.1.2 2-D моделювання Шуму Перліна

Ідея та ж сама, але замість осі з похилими лініями ми маємо на початку сітку. Також для розширення поняття «нахилу» в двох градієнтах, в кожному вузлі сітки ставимо вектор, так як це показано на рисунку 2.8:

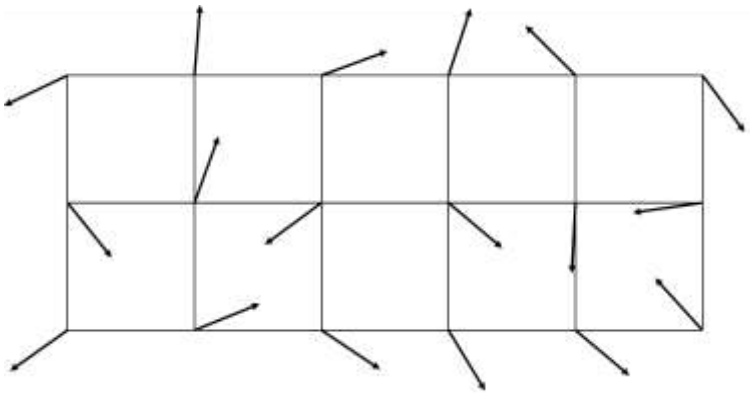


Рисунок 2.8 – Сітка для 2-D представлення градієнтів

В вузлах сітки одиничні вектори - тобто, кожен вектор насправді має довжину 1, і нам важлива лише інформація про напрям, а не про довжину. Є деяка схожість з попереднім представленням, але тут є важлива відмінність. Раніше вхід був горизонтальним - позиція на осі x - і вихід був вертикаллю. Зараз вхід двовимірний – x,y на площині а вихід – вертикаль, перпендикулярна

площині. Як правило це інтенсивність білого точки монітора. Послідовність дій для одномірного представлення була наступна:

1. знайти відстань до найближчої лівої точки і помножити її на тангенс кута нахилу;
2. зробити те ж саме для правої точки;
3. інтерполювати ці значення.

Розглянемо відмінність двовимірного представлення (рис. 2.9). Кожна точка тепер має чотири сусідніх вузла сітки, і додатково потрібна якась подоба добутку нахилу на відстань.

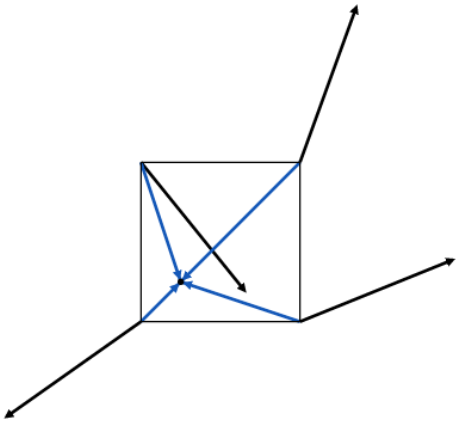


Рисунок 2.9 – Представлення градієнтів на площині

Вирішення проблеми полягає в використанні скалярних добутків. Якщо у вас є вектори (a, b) і (c, d), їх скалярний добуток є сума попарних добутків відповідних компонент: ac + bd. У діаграмі вище, кожна точка сітки має два вихідних вектора: випадковий і другий, що вказує на поточну точку. Добуток нахилу на відстань може бути заміщена скалярним добутком цих двох векторів. Геометрична інтерпретація скалярного добутку визначає, що це добуток довжин векторів на косинус їхнього кута. Таким чином скалярний добуток - відстань від поточної точки до точки на сітці, помножене на косинус кута між векторами.

Косинус демонструє, наскільки малий цей кут - або, в нашому випадку, наскільки зближені два вектора. Якщо вони співпадають, косинус дорівнює одиниці. У міру їх віддалення, косинус меншає: для прямого кута він - нуль (і скалярний добуток стає нулем), а якщо вони протилежні, косинус буде дорівнювати -1.

Щоб візуалізувати це, представим скалярний добуток між точкою і найближчим до неї точкою сітки єдиним градієнтом, так як це показано на рисунку 2.10.

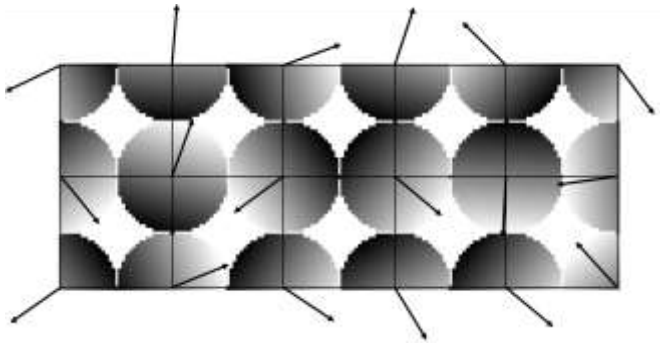


Рисунок 2.10 - Представлення градієнтів в вузлах сітки

Це вже буде щось на зразок тривимірного рельєфу, на який ми дивимося зверху. Координати x і y - це вхід, точки, які вибрали, а z - це вихід. Зазвичай її подають у відтінках сірого, як зроблено тут, але можна також її уявити як глибину. Білі точки виходять з екрану в нашу сторону, сірі йдуть в глибину екрану. Така візуалізація представляє кожен вузол сітки як похилу площину, закріплену за ним. Вектор вказує в сторону білого кінця, того кінця, який виходить з екрану в нашу сторону. Для точок, близьких до вузла сітки, скалярний добуток зводиться до нуля, як і на одновимірному графіку; для точок подалі від вузла значення ближче до екстремалбних.

Тепер, коли в кожній точці є чотири скалярних добутки, нам потрібно їх якось інтегрувати. Лінійна інтерполяція працює тільки для 2 значень, тому ми

будемо виконувати її попарно. Один прохід інтерполяції залишить нас з двома значеннями, другий дасть остаточне значення, яке і буде нашим виходом.

Три і більше вимірів

З цього моменту ми можемо розширити ідею на будь яку кількість вимірів. Вибераємо більше векторів в потрібній кількості вимірів, обчислюємо більше скалярних добутків, інтерполюємо. За кожен інтерполяційний прохід обробляється один вимір.

Варіації

Ідея шуму Перліна досить проста, і для одержання різних ефектів можна змінити будь-які її частини.

Вибір градієнтів

Відомо три способи вибору градієнтів. Перший спосіб - вибрати n випадкових чисел з діапазону [-1;1] (для n вимірювань), використовувати теорему Піфагора, нормувати вектор на довжину. Тепер у нас буде одиничний вектор.

Недолік цього методу в тому, що він генерує більше діагональних векторів, ніж уздовж осей, нам потрібен випадковий кут, а не координата.

Для двох вимірів це дуже легко: візьмемо випадковий кут! Виберемо випадкове число з діапазону [0; 2π]. Позначимомо його θ. Наш вектор (cos θ, sin θ).

Для трьох вимірів теж існує рішення, яке працює у будь-якому вимірі. Принцип такий же: вибрати n випадкових чисел і потім нормувати вектор, отриманий з них. Відмінність така, що числа беруться з нормального розподілу (використовується розподіл Гаусса). В бібліотеках для цього передбачена функція random.gauss ().

Третій метод, що запропонований Перліном в роботі «Improving Noise», пропонує взагалі не брати випадкові градієнти! Замість цього використовується фіксований набір векторів і вибирається випадковий з них. Як наслідок - вирішується проблема «злипання», яка частково була обумовлена зсувом орієнтації градієнтів до діагональних. А другою причиною є оптимізація.

Ключовою відмінністю оригінальної реалізації Кена Перліна було те, що не готувався набір випадкових градієнтів для кожної точки сітки; а готувався набір випадкових градієнтів фіксованого розміру, потім використовувалась технологія вибору градієнта для кожного окремо взятого вузла сітки. Це економило пам'ять і працювало більш швидко, і, втой же час, дозволяло точкам мати низькі або високі значення.

Пропозиції по «поліпшенню» покінчили з випадковими градієнтами зовсім, використовуючи замість них набір з 16 градієнтів, у яких одна координата дорівнює нулю, а дві інших приймають значення ± 1. Ці вектора уже не одиничні, проте скалярні добутки значно легше рахувати. Елемент випадковості зберігається схемою вибору градієнта в кожному конкретному вузлі сітки. Для генерації великих масивів шуму виявилося, що це працює непогано. Ці оптимізації не потрібні для отримання шуму Перліна, але без них не обійтися, якщо ми беремось генерувати шум в реальному часі.

Smootherstep

Вибір функції smoothstep досить вдалий. Можемо використати будь-яку функцію, яка сплющує в нулі і в одиниці.

«Improving Noise» пропонує альтернативу під назвою smootherstep, яка представлена формулою (2.2):

$$y = 6x^5-15x^4+10x^3 \quad (2.2)$$

Можна вибрати схожі поліноми більш високого порядку, хоча вони, збільшать час роботи алгоритму.

Використаємо, наприклад, інтерполяцію синусоїдою (2.3):

$$y = 0.5\sin(\pi\,(x-0.5)) + 0.5 \qquad (2.3)$$

Звичайно це більш низька швидкість в порівнянні з поліномами, але цілком допустимий варіант інтерполяції. На рисунку 2.11 показано використання трьох найпопулярніших способів інтерполяції.

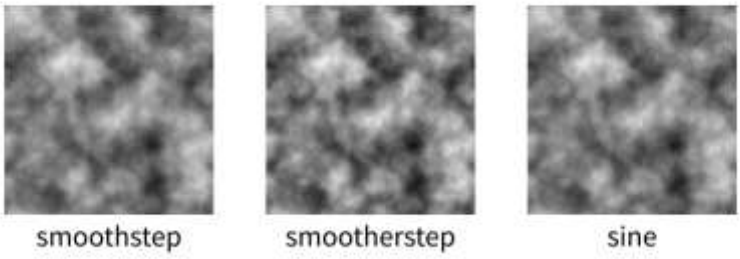


Рисунок 2.11 – Використання різних видів інтерполяції

Картинки досить подібні. У smootherstep чіткіші екстремуми, що виглядає трохи краще.

2.1.3 Моделювання використання октав

З використанням октав є багато перспектив та простору для експерементів. При додаванні більш детального шуму з меншою амплітудою можливо отримати цікаві рішення, як на рисунку 2.12.

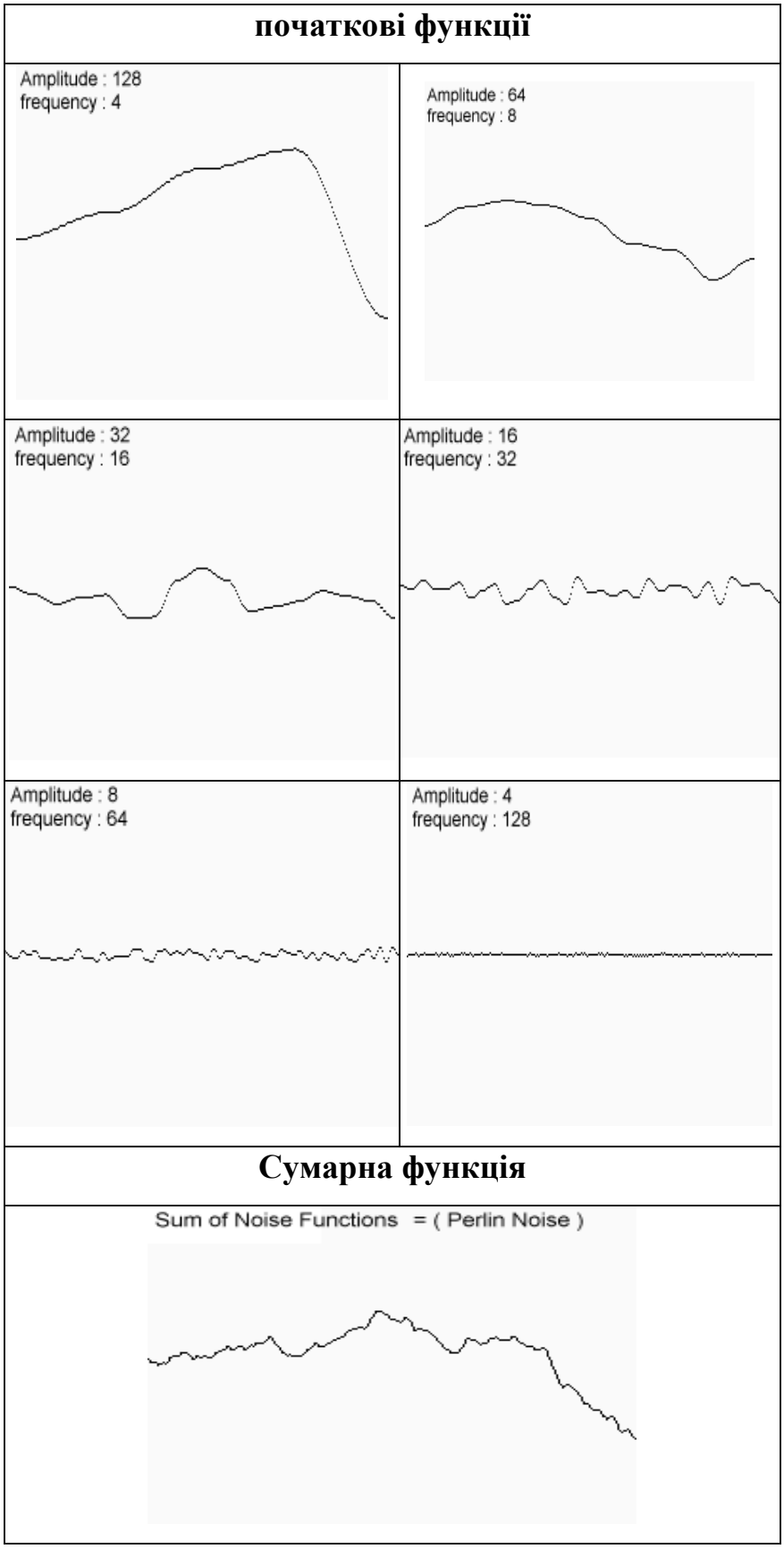


Рисунок 2.12 - Приклад на 6 октав в 1- вимірі

Амплітуду і частоту на кожному кроці змінюють по наступному закону:

frequency = 2i amplitude = persistence,

де і - номер функції, яка генерується. Величина persistence спеціально введена для зв’язування амплітуди з частотою. На рисунку 2.13 октави в двовимірному просторі.

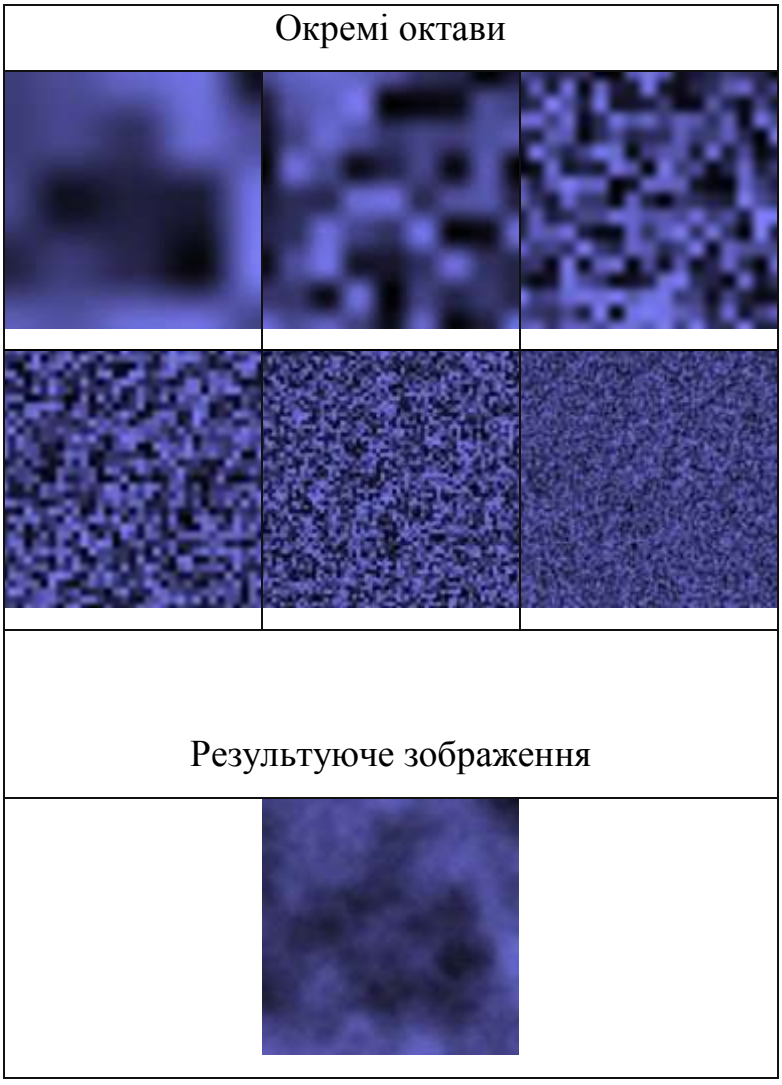


Рисунок 2.13 – Сума 6 октав в двовимірному просторі

Октавою (octave) може бути будь яка функція, що згенерувана і додана в сумарний потік. Кількість октав, також, є базовою величиною в даному процесі і впливає на зображення.

У презентаціях Кена Перліна є кілька прикладів, таких як взяття модуля кожної октави перед складанням для додавання ефекту турбулентності, або взяття синуса для ефекту завихреного мармуру [26].

Симплекс-шум

Багавимірні застосування, навіть при генерації шуму для однієї точки вимагають багато обчислень. Навіть в трьох вимірах є вісім навколишніх точок: вісім скалярних добутків, сім лінійних інтерполяцій, три проходи smoothstep.

Симплекс-шум є варіантом, який використовує трикутники замість квадратів. У трьох вимірах це тетраедр, якому досить всього 4 точки замість 8. У чотирьох вимірах досить 5 точок замість 16. Для генерації чотирьох- або пятимірного шуму це може бути дуже продуктивно.

2.1.4 Деякі властивості Шуму Перліна

Значення шуму Перліна в кожній точці сітки при будь-якому числі вимірів дорівнює нулю.

Шум Перліна безперервний - виключає різкі перепади. Навіть при використанні великої кількості октав, при тому, що результат дуже рваний, можна побачити, що крива все одно гладка.

Всупереч поширеній думці - яку навіть підтримував Кен Перлін - область значень шуму Перліна не дорівнює [-1; 1]. Вона дорівнює ± 0.5n, де n - число вимірів. Для одновимірного випадку цей діапазон дорівнює [-0.5; 0.5], а двовимірний шум затиснутий в діапазоні ± 0.52≈ ± 0.707. Це потрібно враховувати коли є потреба, щоб шум Перліна займав деякий діапазон. Октави, звичайно, збільшать вихідний діапазон. Одна октава збільшить його на 50%; дві - на 75%; три - на 87.5% і так далі.

Вихід шуму Перліна не розподілений рівномірно, навіть для великих вибірок. Ось гістограма двовимірного шуму Перліна на одну октаву, згенерованого в Matlab і показаного на рисунку 2.14:

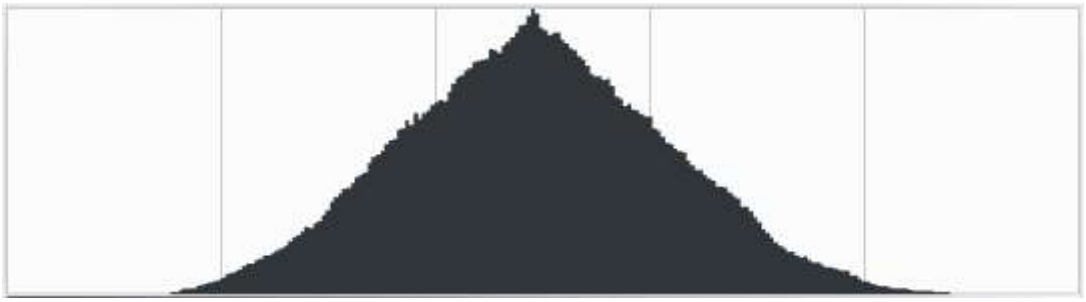


Рисунок 2.14 - Гістограма двовимірного шуму Перліна

Є безліч застосувань шуму Перліна. Можна створити текстуру дерева, отриману за допомогою застосування модульної арифметики. Ми можемо легко отримати хмари, обрізаючи шум на довільним рівні. Дим - це шум Перліна, в якому позитивні значення належать диму, а негативні вирізані.

Завдяки безперервності, можна використати один вимір в якості часу і отримати шум, який буде динамічно змінюватися в часі.

2.2 Розробка інформаційної технології для генерації ландшафтів за допомогою процедурних шумів

Інформаційна технологія, яка пропонується в магістерській кваліфікаційній роботі призначена для розробки та демонстрації навчальних додатків, в яких можна досліджувати процеси створення алгоритмів і самі алгоритми для генерації ландшафтів та текстур з використанням різних типів процедурних шумів, методів та особливостей їх перетворення.

Серед відомих шумів, які використовуються для генерації ландшафтів та текстур, відмітимо наступні:

– Білий шум (White noise);

- Шуми на решітці (Lattice noises);
- Чисельний шум (value noise);
- Градієнтний шум (gradient noise);
- Вейвлет-шум (Wavelet noise);
- Пуассонівський шумовий процес (Poisson process noise);
- Шум Габора (Gabor noise);
- Шум Уорлі (Worley noise);
- Шум Перліна (Perlin noise);
- Симплекс-шум (Simplex noise).

При цьому до них можуть бути застосовані наступні методи цифрової трансформації:

- Метод отримання дискретного шуму з заданими спектральними характеристиками через добуток спектра дискретного білого шуму і певної спектральної обвідної (Fourier Spectral Synthesis);
- Перетворення Фур'є;
- Метод градієнтного шуму;
- Метод зміщення середнього значення (Midpoint displacement method);
- Метод - Diamond-square algorithm;
- Метод розсіяного згорання (Sparse Convolution Noise);
- Діаграма Вороного n-го порядку;
- Метод регулярної сітки з певним рандомним зміщенням вершин (Jittered grid);
- Метод з розрізанням простору на куби і генерацією для куба випадкових точок кількістю, що визначена розподілом Пуассона;
- Метод використання суми октав.

Сформулюємо ряд вимог, яким повинен відповідати шум для моделювання 3-d аморфних об'єктів:

- шум повинен бути когерентним;
- нести в собі всі ознаки фрактала;
- по реалізації повинен бути процедурним;
- повинен бути гладким і одночасно хаотичним;
- ступінь хаотичності повинна бути регульованою;
- процедури шуму повинні забезпечувати режим реального часу;
- повинен забезпечувати генерацію функції шуму в багатовимірному просторі.

Робота інформаційної технології для генерації ландшафтів за допомогою процедурних шумів показана на рисунку 2.15. На першому етапі з використанням користувацького інтерфейсу здійснюється вибір одного із відомих процедурних шумів, що перераховані на сторінці 46. Потім ми вибираємо алгоритм перетворенн процедурного шуму, який орієнтований на використання одного із методів, що приведені на сторінці 46. На другому етапі, знову ж з використанням користувацького інтерфейсу, ми задаємо конкретні атрибути та параметри вибраного метода. І нарешті третій етап, коли по вибраним із застосуванням користувацького інтерфейсу режимам, генеруємо заданий ландшафт. Процес генерації дозволяє в інтерактивному режимі використовувати набагато ширший, у порівнянні з аналогами, набір параметрів для генерації вихідної шумової функції. Це параметри для функції корельованого генерування псевдо-випадкових градієнтів, параметр вибору фукції інтерполяції та параметри для функції управління процесом змішування октав.

Рисунок 2.15 – Структурна схема роботи інформаційної технології

2.3 Розробка архітектури програмного додатку для генерації ландшафтів

В якості технологічного підходу до розробки проекту був обраний об'єктно-орієнтований підхід. Цей вибір обумовлюється легкістю і швидкістю розробки програми. В основу майбутньої архітектури програмного додатку варто закласти архітектурний шаблон-концепцію MVC, тобто відділити процедурну частину від інтерфейсної та візуалізації. До створення програмного додатку варто застосувати компонентний підхід з використанням можливостей бібліотеки OpenGL і візуальної бібліотеки GLScene (із завдатком ігрового ядра). Програмний комплекс складається з трьох модулів: редактора, візуалізації та бібліотеки базових функцій.

Головним класом редактора є інтерфейс користувача, який об’єднує слідуючі компоненти (агрегація), Як це показано в діаграмі компонентів на рисунку 2.16:

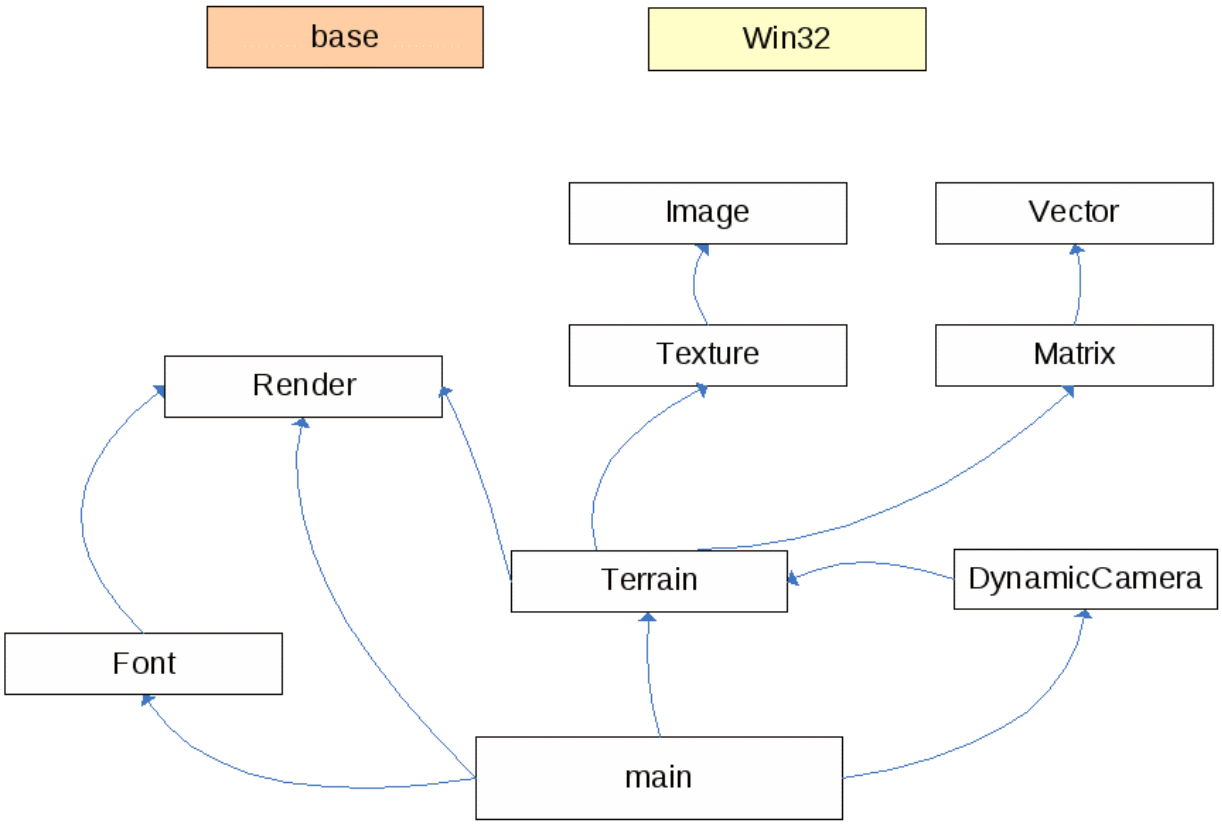


Рисунок 2.16 – Діаграма компонентів додатку

Main - модуль, в якому відбувається зв'язування всіх компонентів програми. В ньому містяться такі функції:

завантаження налаштувань, таких як шлях до карти висот, карті освітленості і текстурі ландшафту і неба, з файлу

void LoadSettings (char * FileName)

В цю подію вказуються дії, що здійснюються в момент старту програми

bool appPreInit ()

В цю подію вказується послідовність дій при візуалізації кожного кадру (кілька разів на секунду)

bool appRender ()

В цю подію вказуються дії, що здійснюються при закритті програми (звільнення ресурсів)

void appShutdown ()

Оброблювач натискань клавіш

bool ProcessKey (HWND hWnd, WPARAM wParam)

Terrain - Описує безпосередньо геометрію ландшафту (вершини, індекси, нормалізує обмін речовин), текстури ландшафту, неба, води.

Також цей клас відповідає за візуалізацію всіх цих об'єктів.

DynamicCamera - Реалізація камери, її пересування, обертання, а також зв'язок з ландшафтом (пересування по ньому).

Font - модуль для ініціалізації і виведення тексту на екран

Render - модуль для реалізації візуалізації в 3-вимірному і 2-вимірному режимах

Image - модуль для завантаження і обробки зображень

Texture - модуль для завантаження зображень і ініціалізацій структур

Vector - модуль для роботи з векторами (додавання, віднімання, скалярний добуток, векторний добуток)

Matrix - модуль для роботи з матрицями (множення, транспонування, знаходження оберненої). Матриці використовуються для реалізації композиції трансформацій (переміщення, обертання, поворот), для реалізації геометричних алгоритмів

Base - модуль для включення основних стандартних бібліотек

Win 32 - модуль, пов'язаний з включенням бібліотек середовища Windows

2.2.1 Використання графічних бібліотек для написання 3D – програм

Для прискорення розробки 3D - програм та ігор використовуються спеціалізовані прикладні програмні (графічні) бібліотеки. Ці бібліотеки можуть бути як стандартними (розробленими лідерами 3D - індустрії), так і фірмовими (розробленими виробниками 3D - прискорювачів). Функції бібліотеки доступні через відповідний API (Application Programming Interface) - програмний інтерфейс розробника. Можна приблизно сказати, що API - мова опису тривимірної графіки. Відповідно, кожна 3D - програма написана з використанням деякого API і відповідно працюватиме у вашій системі тільки

в тому випадку, якщо відео карта підтримує відповідний API, Від самого API багато в чому залежить якість і продуктивність роботи відеоадаптера [11].

В даний час відео карти підтримують два стандартних API:

OpenGL корпорації Silicon Graphic (SGI). Доступний в ОС Windows NT. У Windows 9x реалізований не повністю. У свою чергу, Microsoft створила бібліотеку MCD (Mini Client Driver), що дозволяє задіювати основні можливості OpenGL в Windows 9x.

Direct3d корпорації Microsoft для ОС Windows 9x і NT, скорочено D3d. Він є частиною Microsoft DirectX, який стандартно вбудований до цієї ОС.

2.2.2 Використання графічної бібліотеки OpenGL

OpenGL є на даний момент одним з найпопулярніших програмних інтерфейсів для розробки застосувань в області двовимірної і тривимірної графіки [6, 15]. Наприклад в таких програмних продуктах як для моделювання 3D – графіки це є 3Ds Max, Alias Maya, Auto CAD, Google SketchUp.

Стандарт OpenGL був розроблений і затверджений в 1992 році провідними фірмами в області розробки програмного забезпечення, а його основою стала бібліотека IRIS GL, розроблена Silicon Graphics.

Характерними особливостями OpenGL, які забезпечили розповсюдження і розвиток цього графічного стандарту, є:

Стабільність - доповнення і зміни в стандарті реалізуються так, щоб зберегти сумісність з розробленим раніше програмним забезпеченням.

Надійність і переносимість - застосування, використовуючи OpenGL, гарантують однаковий візуальний результат незалежно від типу використовуваної операційної системи і організації відображення інформації. Крім того, ці застосування можуть виконуватися як на персональних комп'ютерах, так і на робочих станціях і суперкомп'ютерах.

Легкість застосування - стандарт OpenGL має продуману структуру і інтуїтивно зрозумілий інтерфейс, що дозволяє з меншими витратами

створювати ефективні застосування, що містять менше рядків коду, чим з використанням інших графічних бібліотек. Необхідні функції для забезпечення сумісності з різним устаткуванням реалізовані на рівні бібліотеки і значно спрощують розробку застосувань.

Основні можливості OpenGL:

- набір базових примітивів: точки, лінії, багатокутники і тому подібне;
- видові і координатні перетворення;
- видалення невидимих ліній і поверхонь (z-буфер);
- використання сплайнів для побудови ліній і поверхонь;
- накладення текстури і застосування освітлення;
- додавання спеціальних ефектів: туману, зміна прозорості, сполучення кольорів (blending), усунення ступінчастості (anti-aliasing).

Також, існує реалізація OpenGL для різних платформ, для чого було зручно розділити базові функції графічної системи і функції для відображення графічної інформації і взаємодії з користувачем.

На даний момент реалізація OpenGL включає декілька бібліотек (опис базових функцій OpenGL, Glu,glut).

Не дивлячись на те, що бібліотека OpenGL (скорочено GL) надає практично всі можливості для моделювання і відтворення тривимірних сцен, деякі з функцій, які потрібні при роботі з графікою, безпосередньо відсутні в стандартній бібліотеці OpenGL. Тому для OpenGL існують так звані допоміжні бібліотеки.

Перша з цих бібліотек називається GLU. Ця бібліотека вже стала стандартом і поставляється разом з головною бібліотекою OpenGL. До складу цієї бібліотеки увійшли складніші функції, наприклад для того, щоб визначити циліндр або диск буде потрібно всього одну команду. Також до бібліотеки увійшли функції для роботи реалізовані додаткові операції над матрицями і додаткові види проекцій.

Наступна бібліотека, також широко використовувана - це GLUT. Це також незалежна від платформи бібліотека. Вона реалізує не тільки додаткові функції OpenGL, але і надає функції для роботи з вікнами, клавіатурою і мишкою. Для того, щоб працювати з OpenGL в конкретній операційній системі наприклад Windows, треба провести деяку попередню настройку і ця попередня настройка залежить від конкретної операційної системи. З бібліотекою GLUT все набагато спрощується, буквально декількома командами можна визначити вікно, в якому працюватиме OpenGL, визначити переривання від клавіатури або мишки і все це не залежатиме від операційної системи. Бібліотека надає також деякі функції, за допомогою яких можна визначати деякі складні фігури, такі як конуси, тетраедри, і навіть можна за допомогою однієї команди визначити чайник.

2.4 Розробка алгоритмів генерації ландшафтів на основі Шуму Перліна

2.4.1 Розробка одновимірного алгоритму

Схема алгоритму генерації одновимірного Шуму Перліна показана на рисунку 2.17. На початку задається кількість інтервалів інтерполяції (n) та кількість точок в середині інтервала. Потім з використанням ГПВЧ (вершини 1,2) генерується n+1 випадкове число в інтервалі [1;-1]. Це число є значенням тангенсу кута нахилу одиничних векторів-градієнтів. У вершинах 4-6 користувачем вибирається вид інтерполяції. Інтерполяція здійснюється циклічно для кожного інтервалу (7,8). Після інтерполяції генерується потрібна кількість октав (9,10). Параметри генерації октав - частота, амплітуда, фаза, persistans (див. розділ 2.1). Октави сумуються (11) і одержаний одновимірний Шум Перліна візуалізується.

Використовується для генерації:

– лінії обрію;

- берегової лінії;
- хаотичних треків руху об’єктів;
- лінійних текстур різного призначення;
- художніх творів комп’ютерної графіки.

2.3.2 Розробка двовимірного алгоритму

Схема алгоритму генерації одновимірного Шуму Перліна показана на рисунку 2.18. Відмінність двовимірного представлення полягає в тому що замість точок на осі x входом двовимірної шумової функції є точка на площині x,y, а виходом координата z точки на шумовій поверхні, сформованій векторами-градієнтами так як це показано рисунку 2.10 і 2.11 підрозділу 2.1.

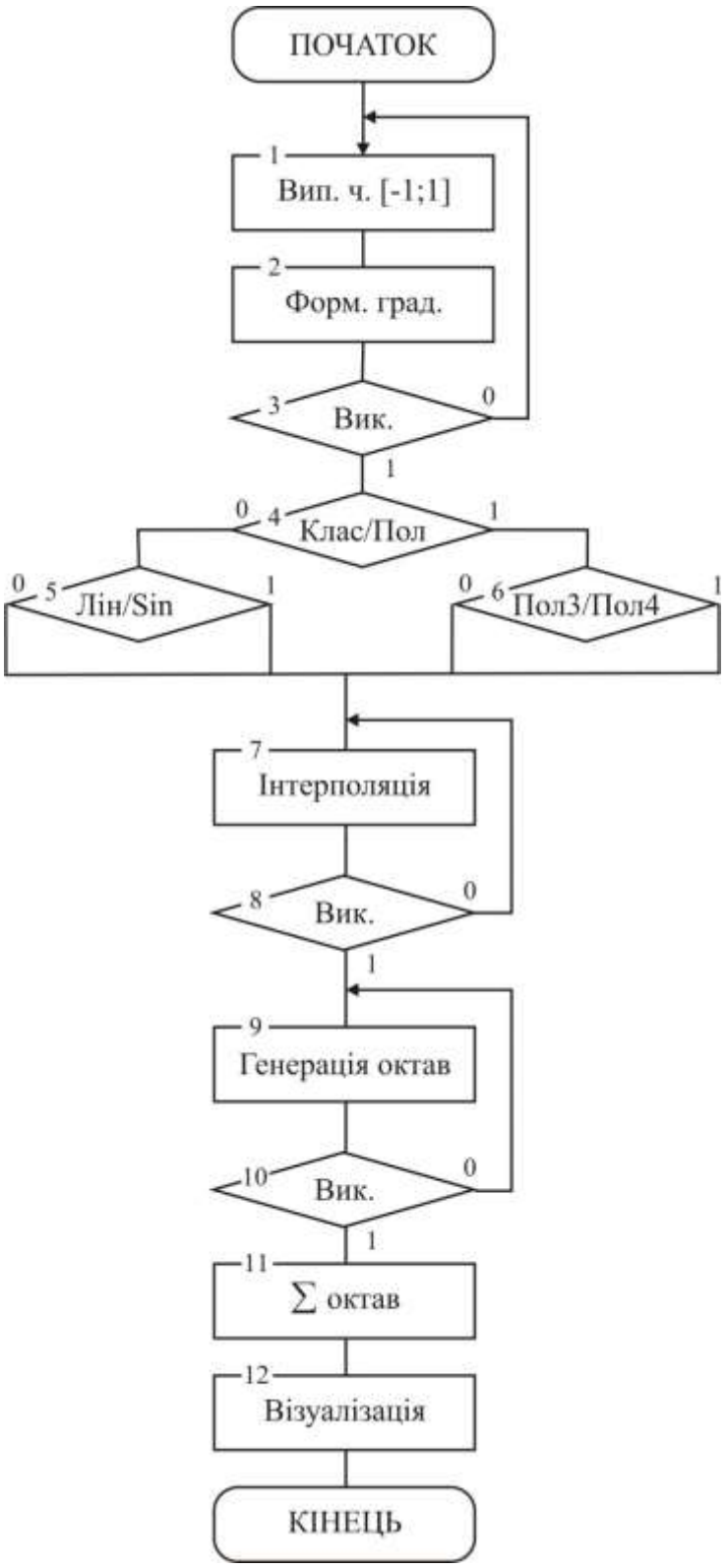


Рисунок 2.17 - Алгоритм генерації одновимірного Шуму Перліна

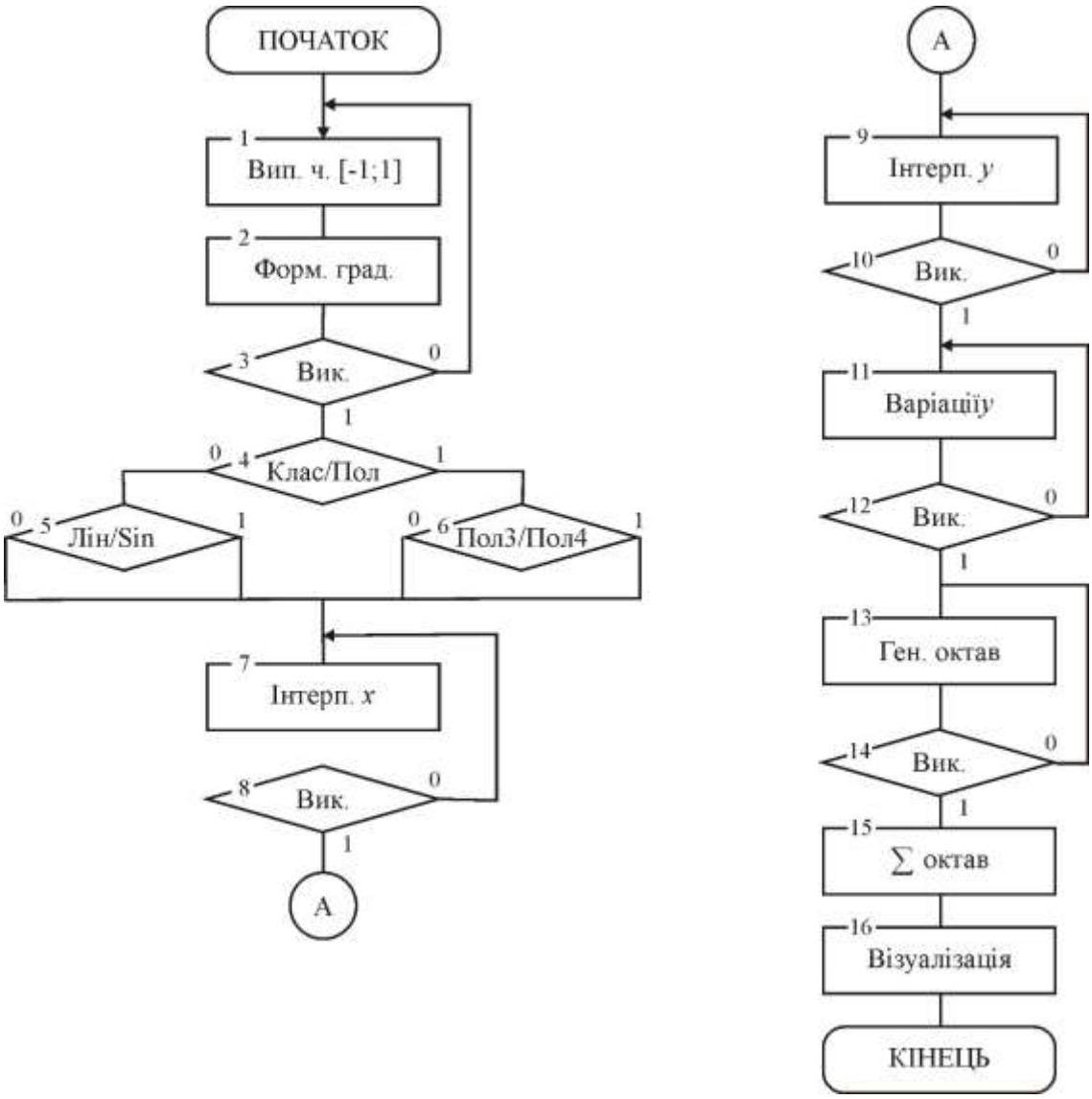


Рисунок 2.18 - Алгоритм генерації двовимірного Шуму Перліна

Інтерполяція виконується спочатку в середині інтервалів по осі x, а потім в середині інтервалів по осі y в два етапи. Відтепер ми маємо набагато більше комбінацій параметрів для формування октав – 2^{2p} в порівнянні з 2^p, де p кількість параметрів. В кінці алгоритма вихід шумової функції візуалізується. Тут є два шляхи. В першому випадку координата z є координатою тривимірного простору (застосовується для формування 3D-ландшафтів або 3D-текстур), а в другому випадку координата z э інтенсивністю горіння

пікселю екрану (застосовується для генерації зображень псевдотривимірних аморфних об’єктів).

2.5 Висновок до розділу 2

В розділілі 1 ми зробили висновок, що найбільш перспективними для розробки технології генерації фрактальних ландшафтів є процедрні шуми. Крім того одним із можливих реалізацій учбового додатку в даній технології є Шум Перліна. Тому в 2 розділі ми провели детальне моделювання Шуму Перліна, розробили етапи та структурну схему дехнології, розробили архітектуру учбового додатку для генерації ландшафтів у вигляді діаграми компонентів а такаж алгоритми роботи технології при генерації карт висот для двовимірного та тривимірного ландшафту.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ ГЕНЕРАЦІЇ ЛАНДШАФТІВ

3.1 Обгрунтування вибору мови та середовища для програмування

Програмний додаток для дослідження «Шуму Перліна» був розроблений за допомогою середовища Delphi.

Delphi - це середовище розробки і мова програмування Object Pascal достатньо гнучкі, щоб створити повноцінну тривимірну гру практично будь-якого жанру з сучасним рівнем графіки або 3D - програм. Багато хто заперечить, що стандартом розробки комп'ютерних ігор є MSVC++ або інші середовища на основі C++, безумовно, могутніше, ніж Object Pascal [5].

Але він і менш високорівневий, тобто в рази складніше. Object Pascal же не тільки простий, але і достатньо гнучкий, щоб на нім можна було розробити повноцінну комп'ютерну гру сучасного рівня. Тепер про середовища. Тут так категорично не скажеш. Середовище розробки - справа смаку і звички кожного конкретного програміста. MSVC++ генерує трохи більш швидкий код, чим Delphi. Власне, на цьому переваги закінчуються.

Козирі Delphi - велика швидкість компіляції (у десятки і навіть сотні разів швидше, ніж MSVC++), висока якість засобів відладки (в більшості випадків Delphi указує точно той рядок коду, в якому міститься помилка, тоді як MSVC ++ може вказати строчку за декілька сторінок від шуканої) і зручний інтерфейс.

Серед великої різноманітності продуктів для розробки програм, Delphi займає одне з провідних місць. Delphi віддають перевагу розробники з різним стажем, звичками, професійними інтересами. За допомогою Delphi написана колосальна кількість додатків, десятки фірм і тисячі програмістів-одинаків розробляють для Delphi додаткові компоненти [12].

В основі такої загальновизнаної популярності лежить той факт, що Delphi, як ніяка інша система програмування, задовольняє викладеним вище вимогам. Дійсно, додатки за допомогою Delphi розробляються швидко, причому взаємодія розробника з інтерактивним середовищем Delphi не викликає внутрішнього відторгнення, а навпаки, залишає відчуття комфорту.

3.2 Використання бібліотек візуальних компонентів Delphi

Компоненти, що використовуються при розробці в Delphi, вбудовані в середовище розробки додатків, представляють з себе набір типів об'єктів, що використовуються як фундамент при будівництві додатку [7, 8].

Це називається Visual Component Library (VCL). В VCL є такі стандартні елементи управління, як рядки редагування, статичні елементи управління, рядки редагування із списками, списки об'єктів. Ще є такі компоненти, які раніше були доступні тільки в бібліотеках третіх фірм: табличні елементи управління, закладки, багатосторінкові записники. Всі об'єкти розбиті на сторінки по своїй функціональності і представлені в палітрі компонент.

VCL містить спеціальний об'єкт, який представляє інтерфейс графічних пристроїв Windows, і дозволяє розробникам малювати, не піклуючись про звичайні для програмування в середовищі Windows деталі.

Ключовою особливістю Delphi є можливість не тільки використовувати візуальні компоненти для будівництва додатків, але і створення нових компонент. Така можливість дозволяє розробникам не переходити в інше середовище розробки, а навпаки, вбудовувати нові інструменти в існуюче середовище. Крім того, можна поліпшити або повністю замінити існуючі по замовчуванню в Delphi компоненти.

Тут слід зазначити, що звичайних обмежень, властивих середовищам візуальної розробки, в Delphi немає. Сама Delphi написана за допомогою Delphi, що говорить про відсутність таких обмежень.

Класи об'єктів побудовані у вигляді ієрархії, які складається з абстрактних, проміжних, і готових компонент. Розробник може користуватися готовими компонентами, створювати власні на основі абстрактних або проміжних, а також створювати власні об'єкти

Проект - це сукупність файлів, з яких складається Delphi - програма.

Інструменти середовища Delphi.

Основними інструментами є:

- головне меню;
- панель інструментів;
- палітра компонентів (Component Palette);
- інспектор об'єктів (Object Inspector);
- вікно форми;
- редактор коду (Code Editor).

Ці інструменти стають доступними після запуску програми Delphi: три знаходяться в головному вікні, а решта — в окремих вікнах.

Головне меню та панель інструментів.

Головне меню складається з таких елементів: File, Edit, Search, View, Project, Run, Component, Database, Tools, Help (рис. 3.1).

Меню File містить стандартні команди роботи з файлами проекту. За допомогою цих команд можна створити новий проект (New Application), нову форму або модуль (New Form і New Unit),відкрити чи закрити файл проекту (Open і Close), закрити всі відкриті файли (Close All), зберегти файл, проект або все відразу (Save, Save As, Save Project As, Save All).

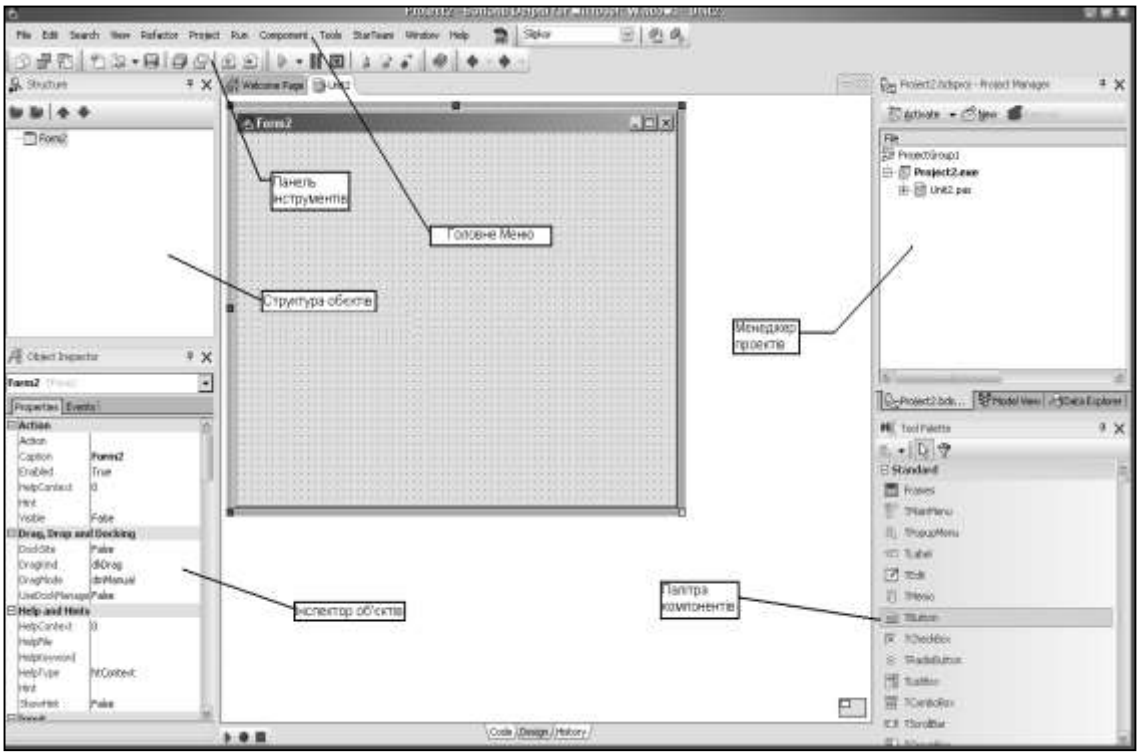


Рисунок 3.1 – Головне меню та панель інструментів

За допомогою команд меню Edit можна вирівнювати компоненти відносно сітки або між собою (Align to Grid, Align), задавати порядок відображення компонентів, які перетинаються (Bring to Front, Senid to Back), змінювати розмір вибраного компонента (Size), масштабувати візуальні компоненти (Scale) тощо. Меню Search містить стандартні команди пошуку та заміни фрагмента тексту (Find, Replace, Search Again, Incremental Search) та інші. У меню View знаходяться команди візуалізації елементів середовища. Меню Project містить команди компіляції (Compile, Build All) та перевірки синтаксису програми (Syntax Check).

Меню Run містить команди налагодження та запуску програми. Меню Component використовують для створення та інсталяції нових компонентів.

Меню Database містить команди виклику інструментів бази даних. У меню Tools знаходяться команди коригування параметрів середовища.

На панелі інструментів розташовані кнопки інструментів. На ній можуть міститися кнопки всіх згаданих команд.

Палітра компонентів.

Палітра компонентів розташована у головному вікні і має вигляд багатосторінкового блокнота. Кожній сторінці відповідає свій набір компонентів. Щоб помістити компонент у центрі вікна форми, двічі клацають на його піктограмі. Якщо потрібно розмістити компонент десь на формі, клацають один раз на його піктограмі і один раз у потрібному місці форми. Для багаторазового вставляння одного й того ж компонента потрібно натиснути клавішу Shift і клацнути на його піктограмі — тепер можна клацати у вікні форми. Щоб відмовитися від цього режиму, треба натиснути на кнопку палітри компонентів з зображенням стрілки. Вибраний компонент можна переміщати на формі, а також змінювати розміри, перетягуючи його маркери.

Інспектор об'єктів.

За допомогою інспектора об'єктів можна задавати початкові значення властивостей об'єктів та їхню реакцію на стандартні події. Вікно інспектора об'єктів містить список компонентів поточної форми, а також дві закладки: властивостей (Properties) та подій (Events). Щоб активізувати вікно інспектора об'єктів, використовують клавішу F11. Розглянемо це вікно (рис. 3.2). Закладка властивостей складається з двох стовпчиків: лівий містить назви властивостей компонентів, а правий — їхні значення. Властивості можуть бути простими або комплексними. Комплексні властивості складаються з набору інших властивостей. Такі властивості в інспекторі об'єктів позначені символом "+", наприклад, +Font. Закладка подій також

має два стовпці. У лівому відображаються назви (імена) стандартних подій, на які об'єкт може реагувати, а в правому - назви методів (процедур), які будуть реалізовувати реакцію на подію.

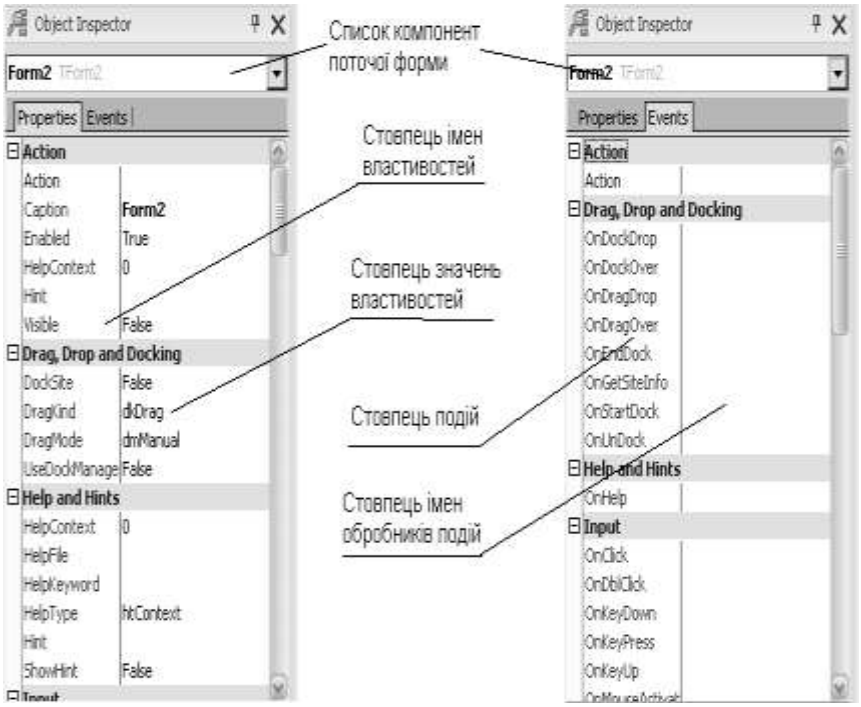


Рисунок 3.2 – Вікно інспектора об’єктів

Кожній стандартній події можна поставити у відповідність деякий метод, назва якого з'явиться після подвійного клацання мишею у правому стовпчику. Одночасно у вікно програми додається шаблон базового коду (процедури) для цього методу. Шаблон потрібно заповнити текстом процедури.

Для введення значень властивостей числового та текстового типу (Width, Name тощо) використовується стандартне поле введення. Значення властивостей перерахованого типу (Align, Cursor тощо) задаються комбінованим списком, звідки вибирають потрібне. Деякі комплексні властивості (Font, Pictures, Glyph тощо) використовують діалогові вікна, набір керуючих елеііентів яких залежить від конкретної властивості.

Вікно форми.

Форма — це вікно Windows, яке утворюється в одному з можливих для вікон стилів. Увесь внутрішній простір є робочою областю, яка має сітку вирівнювання для зручного розташування компонентів на формі. Для виконання групових операцій декілька компонентів можна об'єднувати. Для цього необхідно натиснути на ліву клавішу миші і переміщенням вказівника охопити всі потрібні компоненти. У групу додаються компоненти, які хоча б частково попадають в охоплену область. Можна також додати/вилучити окремий елемент. Для цього потрібно натиснути на клавішу Shift та, не відпускаючи її, вибрати мишею потрібний компонент на формі. Вилучення виокремлених компонентів чи групи виконується клавішею Delete**.** Переміщення виокремленого компонента в межах форми здійснюється мишею. Над компонентами та їхніми групами. можна виконувати операції вирізування, копіювання в буфер обміну і вставляння з буфера.

Вирівнювати компоненти можна як відносно вікна форми, так і відносно один одного. Для цього використовується команда Edit => Align головного меню або палітра вирівнювання (команда View => Alignment Palette головного меню). Інша можливість — безпосередньо задати властивості Left та Top компонентів. Компоненти у групі вирівнюються відносно того компонента, який попав у групу першим.

Редактор коду.

Редактор коду (програми) є в окремому вікні. Це вікно організоване як багато сторінковий блокнот відкритих на даний час файлів. У момент відкриття нового проекту в модуль Unitl, який відповідає формі Forml, редактор автоматично додає програмний код опису цієї форми. Під час, додавання нових компонентів у вікно форми до програми опису форми автоматично додається програмний код опису візуальних параметрів цих

компонентів (висота, ширина, розташування, стиль тощо). Додавання у програму обробника подій певного об'єкта веде до появи заготовки базового коду відповідної процедури у вікні редактора. Заготовка (шаблон) складається з заголовка процедури та стандартних слів begin і end.

3.3 Розробка інтерфейсу додатку для генерації ландшафтів

Програма «Шум Перліна» має простий та інтуїтивно зрозумілий інтерфейс користувача. В ній використано стандартні для Windows елементи керування.

Ця програма дозволяє налаштувати завантаження зображень карти висот і текстури, а також текстур неба і води з файлу або згенерувати їх автоматично.

Програма складається з таких елементів управління як це показано на рисунку 3.3.

Параметри генерації карти висот налаштовуються в розділі «Параметри генерації». У цьому розділі можна завантажити готовий шаблон або встановити параметри вручну.

Поле розмір - установка ширини, довжини карти висот, а висота потрібна для подальшої побудови карти освітленості (регулювання пропорцій ландшафту)

Поле матеріал - установка відбивної здатності матеріалу ландшафту для фонового і дифузного відбиття відповідно (білий колір - відображення всіх кольорів).

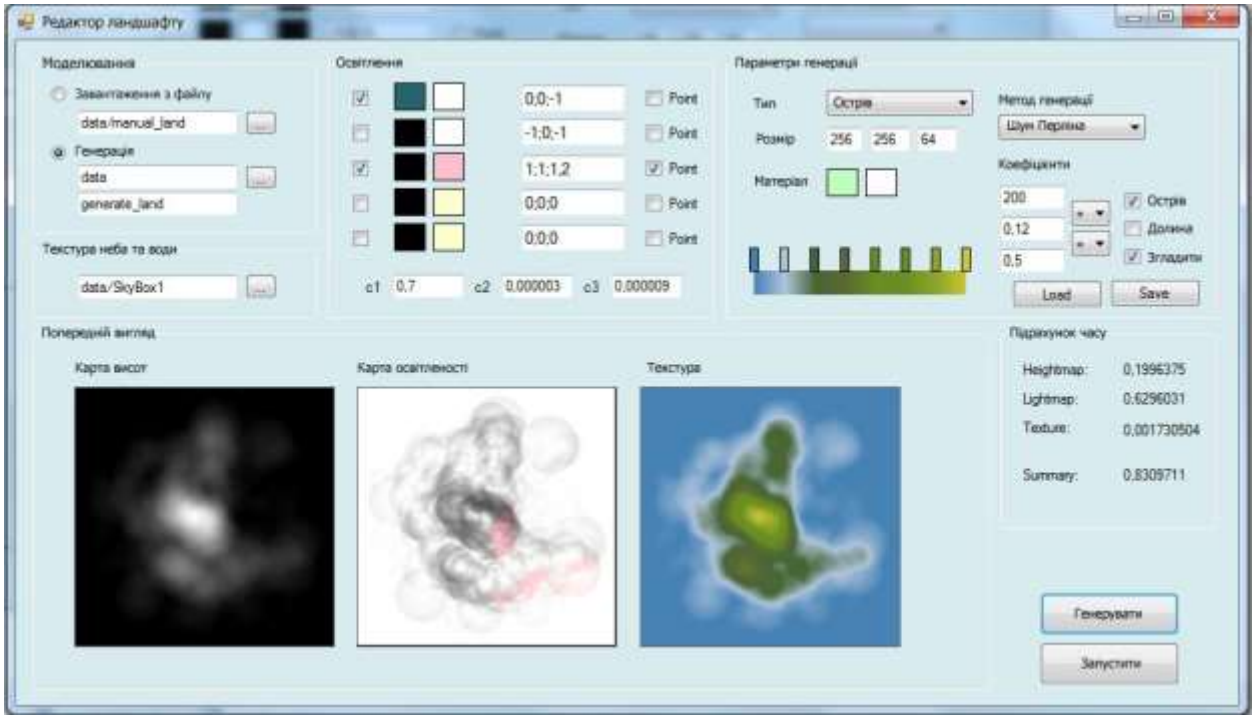


Рисунок 3.3 – Користувацький інтерфейс

Шкала кольорів використовується при побудові текстури ландшафту (кожному значенню висоти відповідає колір з цієї шкали кольорів)

Далі можна вибирати алгоритм генерації ландшафту: за допомогою шуму Перліна або іншими методами.

Прапорець «Острів» використовується для генерації таких типів ландшафтів, на кордоні яких висота дорівнює нулю (для плавного спуску в воду).

Прапорець «Долина» використовується для долінізації ландшафту.

У даній програмі можна створювати карту освітленості з різному налаштованими джерелами світла в кількості не більше 5.

Для кожного джерела світла також можна налаштувати колір фонового освітлення, дифузного освітлення, вектор напрямку (або позицію з коефіцієнтами згасання c1 c2 c3 для точкових джерел).

3.4 Тестування програмного додатку

Генерація карт.

Створити карту висот можна двома способами:

- завантажити вже наявне зображення у форматі BMP (підтримується робота тільки з 24-бітовим кольором).
- згенерувати нову карту

При створенні карти висот, автоматично створюються карта освітленості і текстура.

Збереження карт.

Збереження проводиться автоматично при натисканні на кнопку «Генерувати». При збереженні створюються 3 файли:

- карта висот * .bmp
- карта освітленості * .bmp
- текстура * .bmp

Програма Viewer

Ця програма дозволяє переглядати ландшафти в інтерактивному режимі, побудовані з раніше згенерованих карти висот, текстур і карт освітленості. Також в цій програмі є підтримка неба, водної поверхні, відображень від води (рис. 3.4).

W, A, S, D – навігація по простору.

[] – збільшення / зменшення рівня моря.

Z, X, C – включення / відключення відповідно текстури, карти освітленості і карти деталей.

V – прив'язка камери до поверхні ландшафту (не можна буде спуститися під поверхню).

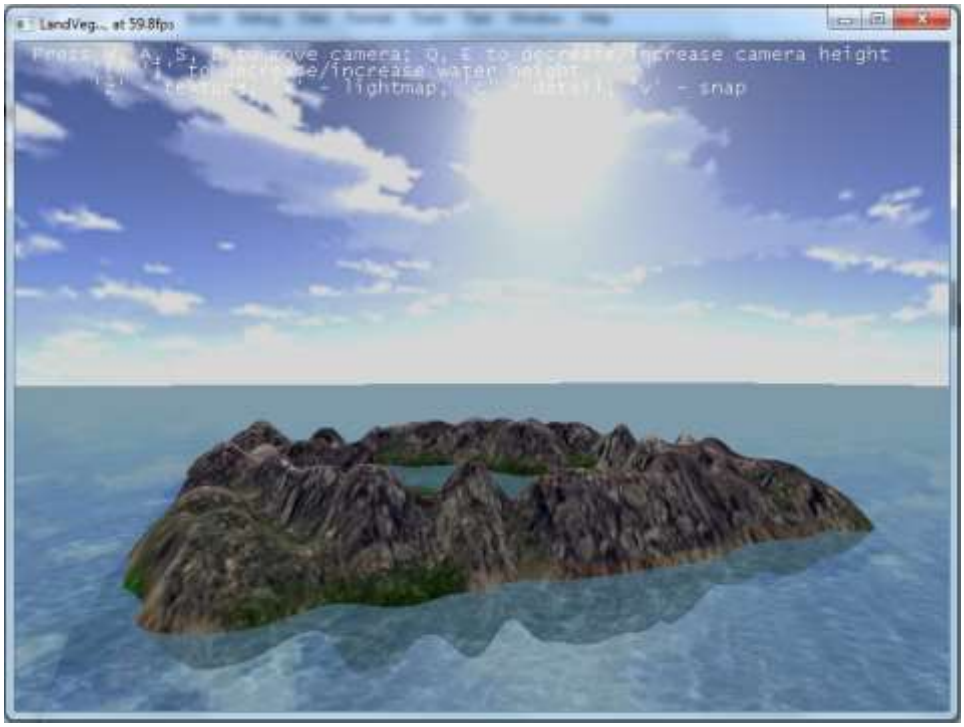


Рисунок 3.4 – Результат тестової генерації ландшафту

3.5 Висновок до розділу 3

В третьому розділі магістерської кваліфікаційної роботи зроблено програмну реалізацію додатку для генерації ландшафтів. Вибрано мову, середовище програмування, бібліотеку візуальних компонентів та розроблено користувацький інтерфейс. Вхідними даними для даної програми є карта висот, карта освітленості і текстура ландшафту, 5 кубічних текстур неба і текстура води в графічних файлах в форматі ".bmp", також файл конфігурації, де вказано шляхи до цих ресурсів. У ньому міститься вся інформація, необхідна для роботи програми. Так само при зображенні ландшафту приймаються дані з клавіатури і мишки, які інтерпретуються як команди користувача, і в залежності від них будується наступне зображення.

Вихідними даними програми є анімаційний ряд із зображенням тривимірного ландшафту, побудованого і відображеного на підставі вхідних даних. Також вихідними даними є інформація про кількість кадрів, що виводяться на екран за секунду.

4 ЕКОНОМІЧНА ЧАСТИНА

Виконання науково-дослідної роботи завжди передбачає отримання певних результатів і вимагає відповідних витрат. Результати виконаної роботи завжди дають нам нові знання, які в подальшому можуть бути використані для удосконалення та/або розробки (побудови) нових, більш продуктивних зразків техніки, процесів та програмного забезпечення.

Дослідження на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів» може бути віднесено до фундаментальних і пошукових наукових досліджень і спрямоване на вирішення наукових проблем, пов’язаних з практичним застосуванням. Основою таких досліджень є науковий ефект, який виражається в отриманні наукових результатів, які збільшують обсяг знань про природу, техніку та суспільство, які розвивають теоретичну базу в тому чи іншому науковому напрямку, що дозволяє виявити нові закономірності, які можуть використовуватися на практиці.

Для цього випадку виконаємо такі етапи робіт:

1. здійснимо проведення наукового аудиту досліджень, тобто встановлення їх наукового рівня та значимості;
2. проведемо планування витрат на проведення наукових досліджень;
3. здійснимо розрахунок рівня важливості наукового дослідження та перспективності, визначимо ефективність наукових досліджень.

4.1 Оцінювання наукового ефекту

Основними ознаками наукового ефекту науково-дослідної роботи є новизна роботи, рівень її теоретичного опрацювання, перспективність, рівень розповсюдження результатів, можливість реалізації. Науковий ефект НДР на тему «Інформаційна технологія для генерації ландшафтів за допомогою

процедурних шумів» можна охарактеризувати двома показниками: ступенем наукової новизни та рівнем теоретичного опрацювання.

Значення показників ступеня новизни і рівня теоретичного опрацювання науково-дослідної роботи в балах наведені в табл. 4.1 та 4.2.

Таблиця 4.1 – Показники ступеня новизни науково-дослідної роботи виставлені експертами

Ступінь новизни	Характеристика ступеня новизни	Значення ступеня новизни, бали		
		Експерти (ПІБ, посада)		
		1	2	3
Принципово нова	Робота якісно нова за постановкою задачі і ґрунтується на застосуванні оригінальних методів дослідження. Результати дослідження відкривають новий напрям в даній галузі науки і техніки. Отримані принципово нові факти, закономірності; розроблена нова теорія. Створено принципово новий пристрій, спосіб, метод	0	62	0
Нова	Отримана нова інформація, яка суттєво зменшує невизначеність наявних значень (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту). Проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів	55	0	57
Відносно нова	Робота має елементи новизни в постановці задачі і методах дослідження. Результати дослідження систематизують і узагальнюють наявну інформацію, визначають шляхи подальших досліджень; вперше знайдено зв’язок (або знайдено новий зв’язок) між явищами. В принципі відомі положення розповсюджені на велику кількість об’єктів, в результаті чого знайдено ефективне рішення. Розроблені більш	0	0	0

Продовження табл. 4.1

	прості способи для досягнення відомих результатів. Проведена часткова раціональна модифікація (з ознаками новизни)			
Традиційна	Робота виконана за традиційною методикою. Результати дослідження мають інформаційний характер. Підтверджені або поставлені під сумнів відомі факти та твердження, які потребують перевірки. Знайдено новий варіант рішення, який не дає суттєвих переваг в порівнянні з існуючим	0	0	0
Не нова	Отримано результат, який раніше зафіксований в інформаційному полі, та не був відомий авторам	0	0	0
Середнє значення балів експертів		58,0		

Згідно отриманого середнього значення балів експертів ступінь новизни характеризується як нова, тобто отримана нова інформація, яка суттєво зменшує невизначеність наявних знань (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту) та проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів.

Таблиця 4.2 – Показники рівня теоретичного опрацювання науково-дослідної роботи виставлені експертами

Характеристика рівня теоретичного опрацювання	Значення показника рівня теоретичного опрацювання, бали		
	Експерт (ПІБ, посада)		
	1	2	3
Відкриття закону, розробка теорії	0	0	0

Продовження табл. 4.2

Глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу	64	68	66
Розробка способу (алгоритму, програми), пристрою, отримання нової речовини	0	0	0
Елементарний аналіз зв'язків між фактами та наявною гіпотезою, класифікація, практичні рекомендації для окремого випадку тощо	0	0	0
Опис окремих елементарних фактів, викладення досвіду, результатів спостережень, вимірювань тощо	0	0	0
Середнє значення балів експертів	66,0		

Згідно отриманого середнього значення балів експертів рівень теоретичного опрацювання науково-дослідної роботи характеризується як глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу.

Показник, який характеризує рівень наукового ефекту, визначаємо за формулою [18]:

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор}, \quad (4.1)$$

де $k_{нов}, k_{теор}$ - показники ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи, $k_{нов} = 58,0$, $k_{теор} = 66,0$ балів;

0,6 та 0,4 – питома вага (значимість) показників ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи.

$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор} = 0,6 \cdot 58,0 + 0,4 \cdot 66,00 = 61,20$ балів.

Визначення характеристики показника $E_{нау}$ проводиться на основі висновків експертів виходячи з граничних значень, які наведені в табл. 4.3.

Таблиця 4.3 – Граничні значення показника наукового ефекту

Досягнутий рівень показника	Кількість балів
Високий	70…100
Середній	50…69
Достатній	15…49
Низький (помилкові дослідження)	1…14

Відповідно до визначеного рівня наукового ефекту проведеної науково-дослідної роботи на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів», даний рівень становить 61,20 балів і відповідає статусу - середній рівень. Тобто у даному випадку можна вести мову про потенційну фактичну ефективність науково-дослідної роботи.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [18]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=24 дні.

$$З_о = 16800{,}00 \cdot 24 / 24 = 16800{,}00 \; грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Науковий керівник дослідної роботи	16800,00	700,00	24	16800,00
Інженер-програміст 1-ї категорії	15200,00	633,33	24	15200,00
Консультант (фахівець-геолог)	15000,00	625,00	10	6250,00
Технік 1-ї категорії	7150,00	297,92	15	4468,75
Всього				42718,75

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.3)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.4)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [18];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 24 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (24 \cdot 8) = 63{,}34$ *грн.*

$З_{р1} = 63{,}34 \cdot 8{,}00 = 506{,}69$ *грн.*

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця інженера-програміста	8,00	2	1,10	63,34	506,69

Продовження табл. 4.5

Інсталяція програмного забезпечення математичного моделювання ландшафтів	5,52	3	1,35	77,73	429,07
Введення програмних блоків моделювання процедурних шумів	18,10	4	1,50	86,37	1563,25
Налагодження програмних блоків математичної моделі	6,45	5	1,70	97,88	631,34
Формування (введення) бази даних дослідження моделі ландшафтів	32,00	3	1,35	77,73	2487,38
Тестування взаємодії досліджуваних моделей	9,50	5	1,70	97,88	929,89
Контроль циклічного експерименту	22,00	3	1,35	77,73	1710,07
Всього					8257,68

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \qquad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод} = (42718,75 + 8257,68) \cdot 10 / 100\% = 5097,64$ *грн.*

4.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (42718,75 + 8257,68 + 5097,64) · 22 / 100% = 12336,30 грн.

4.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів».

Витрати на матеріали на даному етапі проведення досліджень в основному пов’язані з використанням моделей елементів та моделювання роботи і досліджень за допомогою комп’ютерної техніки та створення експериментальних математичних моделей або програмного забезпечення, тому дані витрати формуються на основі витратних матеріалів характерних для офісних робіт.

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}\,, \quad (4.7)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1 = 4,0 \cdot 270,00 \cdot 1,12 - 0 \cdot 0 = 1209,60$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір Офісний А4 500	270,00	4,0	0	0	1209,60
Папір для записів А4 250	144,00	6,0	0	0	967,68
Органайзер офісний	183,00	3,0	0	0	614,88
Канцелярське приладдя	202,00	4,0	0	0	904,96
Картридж для принтера	967,00	2,0	0	0	2166,08
Диск оптичний CD-R	23,00	4,0	0	0	103,04
Flesh-пам'ять 64 GB	596,00	1,0	0	0	667,52
Всього					6633,76

4.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів» відсутні.

4.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 30264,00 \cdot 1 \cdot 1,12 = 33895,68$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ЕОМ для генерації ландшафтів за допомогою процедурних шумів типу (КОМП'ЮТЕР VINGA WOLVERINE A5257 (I5M16G1650.A5257)	1	30264,00	33895,68
Всього			33895,68

4.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 9720,00 \cdot 1 \cdot 1,12 = 10886,40$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Математичне середовище розробки Delphi	1	9720,00	10886,40
Прикладне програмне забезпечення мови програмування Object Pascal	1	8350,00	9352,00
Програмні блоки підтримки графічної бібліотеки OpenGL	1	6730,00	7537,60

Продовження табл. 4.9

Програмні блоки візуальної компонентної бібліотеки GLScene	1	4890,00	5476,80
Всього			33252,80

4.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (8420{,}00 \cdot 1) / (2 \cdot 12) = 350{,}83$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
ОС Windows	8420,00	2	1	350,83

Продовження табл. 4.10

Прикладний пакет Microsoft Office	7350,00	5	1	122,50
Пристрої передачі даних	7690,00	4	1	160,21
Оргтехніка	8260,00	5	1	137,67
Приміщення лабораторії досліджень	426000,00	25	1	1420,00
Робоче місце інженера-програміста	8320,00	7	1	99,05
Медіапроєктор візуалізації ландшафту	22840,00	5	1	380,67
ЕОМ типу НОУТБУК HP 250 G8 (2W8X8EA)	27460,00	3	1	762,78
Всього				3433,70

4.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (4.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 6,20 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,05 \cdot 192,0 \cdot 6,20 \cdot 0,95 / 0,97 = 59,52$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Пристрої передачі даних	0,05	192,0	59,52
Оргтехніка	0,42	50,0	130,20
Пристої відображення інформації	0,25	11,0	17,05
Робоче місце інженера-програміста	0,06	192,0	71,42
Медіапроєктор візуалізації ландшафту	0,36	100,0	223,20
ЕОМ типу НОУТБУК HP 250 G8 (2W8X8EA)	0,03	150,0	27,90
ЕОМ для генерації ландшафтів за допомогою процедурних шумів типу (КОМП'ЮТЕР VINGA WOLVERINE A5257 (I5M16G1650.A5257)	0,25	150,0	232,50
Всього			761,79

4.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.13)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 25%.

$В_{св}$ = (42718,75 + 8257,68) · 25 / 100% = 12744,11 грн.

4.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (42718,75 + 8257,68) · 30 / 100% = 15292,93 грн.

4.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 75%.

$I_в$ = (42718,75 + 8257,68) · 75 / 100% = 38232,32 грн.

4.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв}$ = (42718,75 + 8257,68) · 120 / 100% = 61171,72 грн.

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія для генерації ландшафтів за допомогою

процедурних шумів» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.17)$$

$В_{заг}$= 42718,75 +8257,68 +5097,64 +12336,29653 +6633,76 +0,00 +33895,68 + 33252,80 + 3433,70 +761,79 +12744,11 +15292,93 +38232,32 +61171,72 = =273829,19 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 273829,19 / 0,95 = 288241,25 грн.

4.3 Оцінювання важливості та наукової значимості науково-дослідної роботи

Оцінювання та доведення ефективності виконання науково-дослідної роботи фундаментального чи пошукового характеру є достатньо складним процесом і часто базується на експертних оцінках, тому має вірогідний характер.

Для обґрунтування доцільності виконання науково-дослідної роботи на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів» використовується спеціальний комплексний показник,

що враховує важливість, результативність роботи, можливість впровадження її результатів у виробництво, величину витрат на роботу.

Комплексний показник K_P рівня науково-дослідної роботи може бути розрахований за формулою:

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t}, \qquad (4.19)$$

де I – коефіцієнт важливості роботи. Приймемо $I = 4$;

n – коефіцієнт використання результатів роботи; $n = 0$, коли результати роботи не будуть використовуватись; $n = 1$, коли результати роботи будуть використовуватись частково; $n = 2$, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках; $n = 3$, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок. Приймемо n=3;

T_C – коефіцієнт складності роботи. Приймемо $T_C = 2$;

R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то $R = 4$; якщо результати роботи відповідають відомому рівню, то $R = 3$; якщо нижче відомих результатів, то $R = 1$. Приймемо $R = 3$;

B – вартість науково-дослідної роботи, тис. грн. Приймемо B =288241,25 грн;

t – час проведення дослідження. Приймемо t = 0,08 років, (1 міс.).

Визначення показників *I, n, T_C, R, B, t* здійснюється експертним шляхом або на основі нормативів [18].

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t} = 4^3 \cdot 2 \cdot 3 \,/\, 288{,}2 \cdot 0{,}08 = 15{,}99.$$

Якщо $K_P > 1$, то науково-дослідну роботу на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів»

можна вважати ефективною з високим науковим, технічним і економічним рівнем.

4.4 Висновок до розділу 4

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів» складають 288241,25 грн. Відповідно до проведеного аналізу та розрахунків рівень науково-економічного ефекту проведеної науково-дослідної роботи на тему «Інформаційна технологія для генерації ландшафтів за допомогою процедурних шумів» є середній, а дослідження актуальними, рівень доцільності виконання науково-дослідної роботи $K_P > 1$, що свідчить про потенційну ефективність з високим науковим, технічним і економічним рівнем.

ВИСНОВКИ

В процесі виконання магістерської кваліфікаційної роботи було проаналізовано предметну область програмних засобів по створенню комп’ютерних ландшафтів, визначено її границі, системні та програмні вимоги. Для вирішення цієї задачі була вибрана методологія використання когерентного фрактального шуму. Серед багатьох існуючих методів генерування фрактального шуму був обраний метод «Шум Перліна», як найбільш широковживаний та універсальний. На основі проведених досліджень було проведено проектування та реалізація учбового програмного додатку для дослідження цього методу. Всі задачі, поставлені перед магістерською кваліфікаційною роботою виконані в повному об’ємі, а саме:

- проаналізовано сучасні методи, алгоритми та технології, пов’язані з генерацією ландшафтів;
- обґрунтовано вибір методів та етапів для створення інформаційної технологія генерації ландшафтів за допомогою процедурних шумів;
- розроблено інформаційну технологія для генерації ландшафтів за допомогою процедурних шумів;
- реалізовано та протестовано додаток для генерації ландшафтів;
- розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 288241,25 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат.

Мета – розширення функціональних можливостей реалізованого додатку досягнута за рахунок того, що на відміну від аналогів (відомих бібліотечних реалізацій), він дозволяє в інтерактивному режимі використовувати набагато ширший набір параметрів для генерації вихідної шумової функції. Це параметри для функції корельованого генерування псевдо-випадкових градієнтів, параметр вибору фукції інтерполяції та параметри для функції управління процесом змішування октав.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

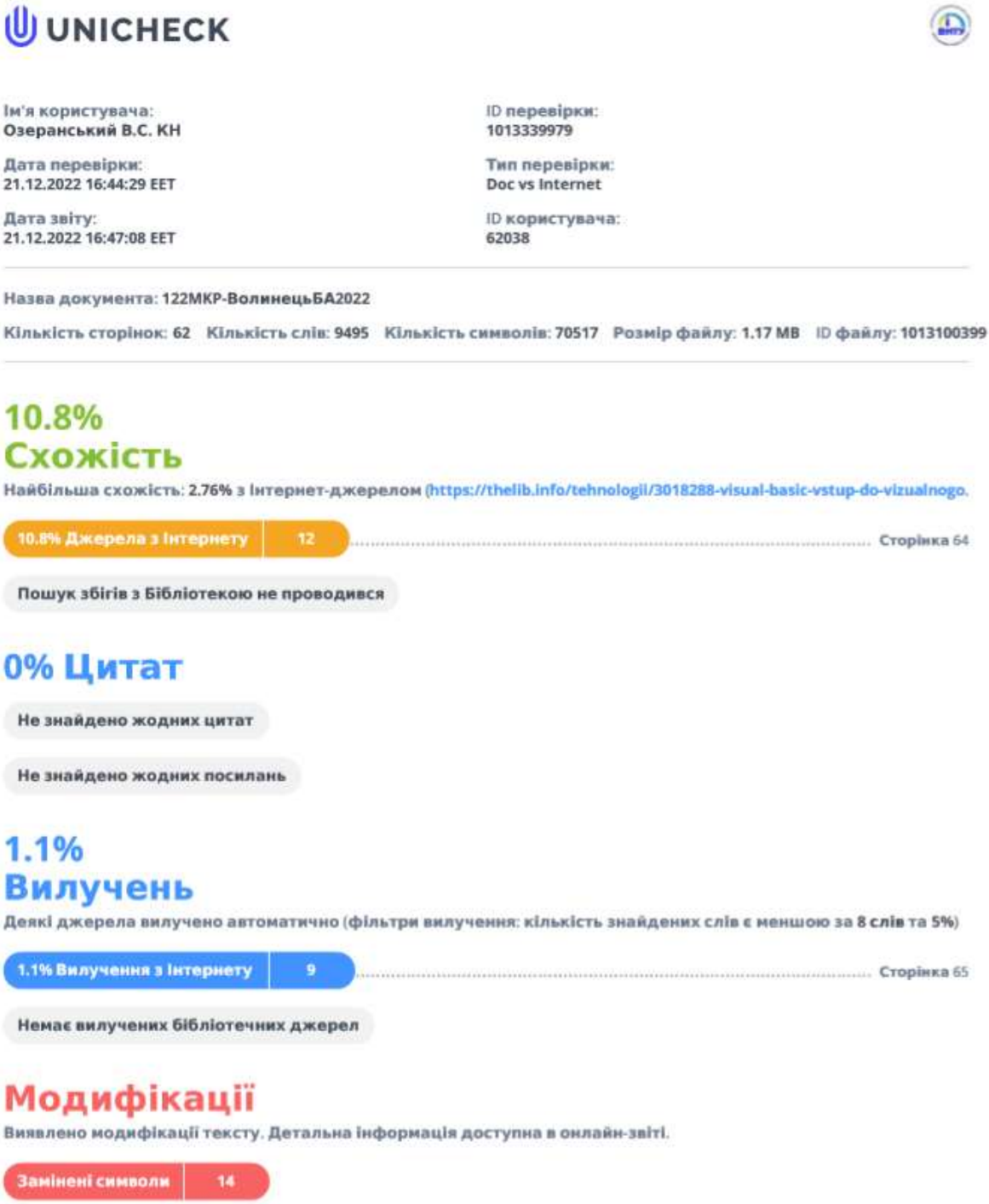
1. Волинець Б.А. Сілагін О.П. Застосування алгоритму «Шум Перліна» в учбовому додатку для генерації фрактальних ландшафтів» Матеріали Науково-технічної конференції факультету інтелектуальних інформаційних технологій та автоматизації (2022): Тез. доп. – м. Вінниця, Україна, 2022, 2с. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/11649
2. Сілагін О., Лученко М., Поліщук Б. Застосування горбкового алгоритму в учбовому додатку фрактальні ландшафти: Матеріали доповіді на Регіональній НПК «ФІТКІ ВНТУ- 2018». – Вінниця: Електронний репозитарій ВНТУ. Режим доступу: http://ir.lib.vntu.edu.ua/handle/123456789/20591
3. Сілагін О., Лученко М., Поліщук Б. Навчальний додаток для дослідження горбкового алгоритму: Матеріали доповіді на МНПК «ІОН-2018». – Вінниця: «Універсум», ВНТУ. – 2018.
4. Г.Снук - 3D ландшафти в реальному часі на C ++ і DirectX9: пер. з англ. - М.: Изд.дом «Вильямс». – 2008.
5. Роджерс Д. Алгоритмічні основи машинної графіки: пров. з англ М .: Мир, 1989.- 512 с.: іл.
6. Авдєєва С.М., Куров А.В. Алгоритми тривимірної машинної графіки: Навчальний посібник. - М .: Изд-во МГТУ ім. Н.е. Баумана, 1996. - 60 с .: іл.
7. Граді Буч “Об’єктно-орієнтоване проектування”.—М.: Мир.—1992.
8. Элиенс Т. Объектно-ориентированное программирование. – М.: Изд.дом «Вильямс». – 2003.

9. Сілагін О.В. Мови об’єктно-орієнтованого програмування. Навч.посібн.. – Вінниця: «Універсум», ВНТУ. – 2004.
10. Программное обеспечение персональных ЭВМ. Под ред. Верлонь // Вища школа, 1993.
11. Г. Джойс. Языки систем искусственного интеллекта.—М.: Мир, 1994.
12. Ротштейн О.П. Інтелектуальні технології ідентифікації: нечіткі множини, генетичні алгоритми, нейронні мережі. – Вінниця: Універсам – Вінниця, 1999. – 320с., іл..
13. Искусственный интеллект. Справочник в 3-х томах. - М.: Радио и связь, 1990.
14. Касаткин В.И. Алгоритмы и игры. - К.: Техніка, 1984.
15. Ковальски Р. Логика в решении проблем. - М.: Наука, 1990.
16. Гарднер М. Математические досуги: Пер. с англ. - М.: Мир, 1972.
17. М. Краснов, “OpenGL. Графика в проектах DELPHI” г.Москва 2000г.
18. “Delphi 7 – Для профессионалов” ЗАО Издательский Дом “Питер”, 2004г.
19. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
20. Д. Кузан, В. Шарапов, “Программирование в Delphi” г.Санкт-Петербург, 2005г.
21. Сайт Delphi World 6.0. Графика и Игры.Режим доступу: http://delphiworld.narod.ru/_graphic_.html
22. Сайт MirGames. Статьи\Уроки по GLScene. Режим доступу: http://www.mirgames.ru/articles/glscene.html
23. Сайт GLScene. Статьи по GLScene. Режим доступу: http://www.glscene.ru/content.php?article
24. Сайт Мир 3D. Природное самоподобие. Режим доступу: http://www.mir3d.ru/articles/84/

25. Клуб программистов Delphi programming. Фракталы – геометрия природы. Режим доступу: http://programmersclub.ru/gambler-fractali1/
26. Сайт GameDev. Реализация Шума Перлина. Режим доступу: http://www.gamedev.ru/code/articles/?id=4212
27. Russian Computer Graphics. Генерация изображения облаков (Clouds image generation). Режим доступу: http://community.livejournal.com/ru_compgraphics/
28. Сайт Введение в компьютерную графику. Визуализация природных явлений. Режим доступу: http://graphicon.ru/oldgr/courses/cg02b/assigns/hw-5/hw5_cld.htm
29. Сайт Вікіпедія. Фрактали. Режим доступу: https://uk.wikipedia.org/wiki/Фрактал
30. «Ken Perlin's Homepage» - http://mrl.nyu.edu/perlin/
31. Е. Федер ,“Фракталы” г. Москва, 1991г. – 249с.
32. Сайт 3D Accelerator. Статья Генерация трехмерных ландшафтов. Режим доступу: http://www.3daccelerator.com.ua/articles.html

Додаток А

Результат перевірки на плагіат в онлайн-системі UNICHECK



UNICHECK

Ім'я користувача:
Озеранський В.С. КН

ID перевірки:
1013339979

Дата перевірки:
21.12.2022 16:44:29 EET

Тип перевірки:
Doc vs Internet

Дата звіту:
21.12.2022 16:47:08 EET

ID користувача:
62038

Назва документа: 122МКР-ВолинецьБА2022

Кількість сторінок: 62 Кількість слів: 9495 Кількість символів: 70517 Розмір файлу: 1.17 MB ID файлу: 1013100399

10.8%
Схожість

Найбільша схожість: 2.76% з Інтернет-джерелом (https://thelib.info/tehnologii/3018288-visual-basic-vstup-do-vizualnogo.

10.8% Джерела з Інтернету 12 ..... Сторінка 64

Пошук збігів з Бібліотекою не проводився

0% Цитат

Не знайдено жодних цитат

Не знайдено жодних посилань

1.1%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 5%)

1.1% Вилучення з Інтернету 9 ..... Сторінка 65

Немає вилучених бібліотечних джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 14

Додаток Б

Лістинг програми

```
unit GenFL;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, BusinessSkinForm, bsSkinData, bsSkinCtrls,ExtCtrls, bsSkinExCtrls,
  bsSkinBoxCtrls, StdCtrls, Mask, bsSkinShellCtrls;
type
  PLongA = ^TLongA;
  TLongA = array [0..0] of LongWord;
  PByteA = ^TByteA;
  TByteA = array [Word] of Byte;
  PRGBQuadA = ^TRGBQuadA;
  TRGBQuadA = array [Word] of TRGBQuad;
  TLongArray = array of LongWord;

type
  TGenerationForm = class(TForm)
    bsCompressedSkinList1: TbsCompressedSkinList;
    bsSkinData1: TbsSkinData;
    bsBusinessSkinForm1: TbsBusinessSkinForm;
    bsSkinPanel1: TbsSkinPanel;
    bsSkinPanel2: TbsSkinPanel;
    bsSkinButton1: TbsSkinButton;
    bsSkinStatusBar1: TbsSkinStatusBar;
    Image1: TImage;
    Label1: TbsSkinShadowLabel;
    bsSkinShadowLabel2: TbsSkinShadowLabel;
    bsSkinShadowLabel3: TbsSkinShadowLabel;
    bsSkinShadowLabel4: TbsSkinShadowLabel;
    bsSkinShadowLabel5: TbsSkinShadowLabel;
    tedtOfX: TbsSkinTrackEdit;
    tedtOfY: TbsSkinTrackEdit;
    ScaleX: TbsSkinSpinEdit;
    ScaleY: TbsSkinSpinEdit;
    ComboBoxOctav: TbsSkinComboBox;
    bsSkinSavePictureDialog1: TbsSkinSavePictureDialog;
```

```
  bsSkinButtonLabel1: TbsSkinButtonLabel;
  procedure FormCreate(Sender: TObject);
  procedure bsSkinButton1Click(Sender: TObject);
  procedure FormDestroy(Sender: TObject);
  procedure ScaleXChange(Sender: TObject);
  procedure ScaleYChange(Sender: TObject);
  procedure tedtOfXChange(Sender: TObject);
  procedure tedtOfYChange(Sender: TObject);
  procedure ComboBoxOctavChange(Sender: TObject);
  procedure bsSkinButtonLabel1Click(Sender: TObject);
private
 bmg : TBitmap;
  NrFrame:   Integer;
  st2, startTime: double;
  Map,Map1: TLongArray;
  Pal: array[0..255] of TRGBQuad;

  procedure CreateMap;

  procedure InitNoise(seed: Integer);
  function  noise1D(const x: double): double;
  function noise2D(const x, y: double): double;
  function noise3D(const x, y, z: double): double;
  function turbulence1D(x: double; const n: integer): double;
  function turbulence2D(x, y: double; const n: integer): double;
  function turbulence3D(x, y, z: double; const n: integer): double;
  function turbulence2Di(x, y: Integer; const n: integer): Integer;
  function noise2Di(const x, y: Integer): Integer;
  function turbulence3Di(x, y, z: Integer; const n: integer): Integer;
  function noise3Di(const x, y, z: Integer): Integer;

  procedure CreatePalette;
  function GenerateMap(const SizeX, SizeY: Word;
                       ScaleX: double = 0;
                       ScaleY: double = 0;
                       Octaves: integer = -1;
                       OffsetX: double = 0;
                       OffSetY: double = 0): TLongArray; overload;
  procedure GenerateMap(Map: TLongArray;
                       const SizeX, SizeY: Word;
```

```
                ScaleX: double = 0;
                ScaleY: double = 0;
                Octaves: integer = -1;
                OffsetX: double = 0;
                OffSetY: double = 0); overload;
    function turbulence2Dia(x, y: Integer; const n: integer): Integer;
    procedure ColorizeBaseMap(Map: TLongArray; Color: integer = 0);
    procedure PowMap(Map: TLongArray);
    procedure MultMap(Map1, Map2: TLongArray);

  public
    procedure WMEraseBkgnd(var Message: TWMEraseBkgnd); message WM_ERASEBKGND;
  end;

const
  MapSizeX = 256;
  MapSizeY = 256;

const
  B:  Integer = $100;
  BM: Integer = $FF;
  N:  Integer = $1000;
  Ni:  Integer = $400000;
  NM: Integer = $FFF;
  NP: Integer = 12;
  B2: Integer = $202; // B + B + 2;
var
  P:  array[0..$202] of Integer;
  G1: array[0..$202] of Double;
  G2: array[0..$202,0..1] of Double;
  G2i: array[0..$202,0..1] of Integer;
  G3: array[0..$202,0..2] of Double;
  G3i: array[0..$202,0..2] of Integer;


var
  GenerationForm: TGenerationForm;
  sX,sY : Double;
  oX,oY : Integer;
  Oct: integer;
```

```
implementation

{$R *.dfm}
uses
  math, mmsystem, mainFL;


procedure TGenerationForm.bsSkinButton1Click(Sender: TObject);
Var
 s: double;
begin
  CreateMap;
  Image1.Canvas.Draw(0,0,bmg);
  //
  Inc(NrFrame);
  s := (timeGetTime * 0.001) - StartTime;
    startTime := timeGetTime * 0.001;
    NrFrame := 0;
  invalidate;
  Form1.GenLands(Image1);
  Form1.Image1.Picture.Bitmap.Assign(Image1.Picture.Bitmap);
  Form1.bsSkinPanel4.Enabled:=true;
  Form1.loadData;
  Form1.bsSkinStatusPanel1.Caption:='Triangles: '+
    intToStr(Form1.Landscape.MeshObjects.Items[0].TriangleCount);
end;

procedure TGenerationForm.FormCreate(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clBlack;
  Image1.Canvas.Rectangle(256,256,-256,-256);
  Randomize;
  startTime := timeGetTime * 0.001;
  NrFrame := 0;
  bmg := Tbitmap.Create;
  bmg.Width := MapSizeX;
  bmg.Height := MapSizeY;
  bmg.PixelFormat := pf32bit;
  SetLength(Map, MapSizeX * MapSizeY);
  SetLength(Map1, MapSizeX * MapSizeY);
```

```
  CreatePalette;
  InitNoise(100);
  st2 := now;
  sX:=ScaleX.Value;
  sY:=ScaleY.Value;
  oX:=tedtOfX.Value;
  oY:=tedtOfY.Value;
  Oct:=0;
end;

procedure TGenerationForm.FormDestroy(Sender: TObject);
begin
 bmg.free;
end;

function TGenerationForm.GenerateMap(const SizeX, SizeY: Word; ScaleX,
  ScaleY: double; Octaves: integer; OffsetX, OffSetY: double): TLongArray;
var
  map: TLongArray;
begin
  if (SizeX * SizeY) = 0 then
    Exit;
  SetLength(Map, Sizex * SizeY);
  GenerateMap(Map, SizeX, SizeY, ScaleX, ScaleY, Octaves, OffsetX, OffsetY);
  Result := map;
end;

procedure TGenerationForm.GenerateMap(Map: TLongArray; const SizeX, SizeY: Word; ScaleX,
  ScaleY: double; Octaves: integer; OffsetX, OffSetY: double);
var
  x,y,i: integer;
  v: integer;
  Xi,Yi,DXi,DYi: integer;
begin
  if (SizeX * SizeY) = 0 then
    Exit;

  if High(Map) <> (Sizex * SizeY -1) then
    SetLength(Map, Sizex * SizeY);
```

```
 if octaves < 0 then
   Octaves := Round(Oct);

 if ScaleX = 0 then
   ScaleX := sqrt(random + sX);

 if ScaleY = 0 then
   ScaleY := sqrt(random + sY);

 if OffsetX = 0 then
   OffsetX := random * oX;

 if OffsetY = 0 then
   OffsetY := random * oY;

 Xi := Round(OffsetX * 1024);
 Yi := Round(OffsetY * 1024);
 DXI := Round(1024 * ScaleX/SizeX);
 DYI := Round(1024 * ScaleY/SizeY);

 i := 0;
 for y := 0 to SizeY - 1 do begin
   for x := SizeX - 1 downto 0 do begin
     v := turbulence2Dia(XI,YI, Octaves) shr 2;

     map[i] := (v shl 16) + (v shl 8) + v;
     Inc(i);
     Inc(XI, DXI);
   end;
   XI := Round(OffsetX * 1024);
   Inc(YI, DYI);
 end;
end;

procedure TGenerationForm.ComboBoxOctavChange(Sender: TObject);
begin
 if ComboBoxOctav.ItemIndex>-1 then
   Oct:=strToint(ComboBoxOctav.Items.Strings[ComboBoxOctav.ItemIndex]);
end;
function HMPath : string;
```

```
begin
  Result := ExtractFilePath(Paramstr(0))+'Data\HeightMap\';
end;
procedure TGenerationForm.bsSkinButtonLabel1Click(Sender: TObject);
begin
 bsSkinSavePictureDialog1.InitialDir:=HMPath;
if (bsSkinSavePictureDialog1.Execute) and (bsSkinSavePictureDialog1.FileName<>'')
 then
   begin
    Image1.Picture.Bitmap.PixelFormat:=pf24bit;
    Image1.Picture.SaveToFile(bsSkinSavePictureDialog1.FileName);
   end;
end;

procedure TGenerationForm.ColorizeBaseMap(Map: TLongArray; Color: integer);
var
 pal: array[0..255] of LongWord;
 r,g,b: Integer;
 i: integer;
begin
 if Color <> 0 then begin
   r := (Color and $FF0000) shr 16;
   g := (Color and $FF00) shr 8;
   b := Color and $FF;
 end else begin
   repeat
     r := Round(255);
     g := Round(255);
     b := Round(255);
   until ((r + g + b) > 550){ and ((r + g + b) < 250)};
 end;

 for i:= 0 to 255 do begin
   Pal[i] := round(min((i*r)/127,255)) shl 16 +
             round(min((i*g)/127,255)) shl 8 +
             round(min((i*b)/127,255));
 end;

 for i := 0 to High(Map) do begin
   Map[i] := Pal[Map[i] and $FF];
```

```
  end;
end;

procedure TGenerationForm.PowMap(Map: TLongArray);
var
  r,g,b,v,i: integer;
begin
  for i := 0 to High(Map) do begin
    v := Map[i];
    r := 255 - ((v and $FF0000) shr 16);
    g := 255 - ((v and $FF00) shr 8);
    b := 255 - (v and $FF);

    r := r * r div 256;
    g := g * g div 256;
    b := b * b div 256;
    map[i] := r shl 16 + g shl 8 + b;
  end;
end;

procedure TGenerationForm.ScaleXChange(Sender: TObject);
begin
  sX:=ScaleX.Value;
end;

procedure TGenerationForm.ScaleYChange(Sender: TObject);
begin
  sY:=ScaleY.Value;
end;

procedure TGenerationForm.MultMap(Map1, Map2: TLongArray);
var
  v,w,i: integer;
  rv,gv,bv,rw,gw,bw,r,g,b: integer;
begin
  for i := 0 to High(Map) do begin
    v := Map1[i];
    w := Map2[i];
    rv := (v and $FF0000) shr 16;
    gv := (v and $FF00) shr 8;
```

```
    bv := v and $FF;
    rw := (w and $FF0000) shr 16;
    gw := (w and $FF00) shr 8;
    bw := w and $FF;
    r := min(255, (((60 * 3) div 2 + 127) * rw) div 256);
    g := min(255, (((60 * 3) div 2 + 127) * gw) div 256);
    b := min(255, (((60 * 3) div 2 + 127) * bw) div 256);
    map1[i] := r shl 16 + g shl 8 + b;
  end;
end;

procedure TGenerationForm.CreateMap;
var
  x,y,i: longword;
  Row: PRGBQuadA;
begin
  GenerateMap(Map, MapSizeX, MapSizeY);
  ColorizeBaseMap(Map);
  PowMap(Map);
  GenerateMap(Map1, MapSizeX, MapSizeY);
  ColorizeBaseMap(Map1);
  PowMap(Map1);
  MultMap(Map, Map1);
  GenerateMap(Map1, MapSizeX, MapSizeY);
  ColorizeBaseMap(Map1);
  PowMap(Map1);
  MultMap(Map, Map1);

  i := 0;
  for y := 0 to MapSizeY - 1 do begin
    Row := bmg.Scanline[y];
    for x := 0 to MapSizeX - 1 do begin
      Row[x] := tRGBQUAD(map[i]);
      Inc(i);
    end;
  end;

end;

procedure TGenerationForm.WMEraseBkgnd(var Message: TWMEraseBkgnd);
```

```
begin
  Message.Result := 1;
end;

function TGenerationForm.noise1D(const x: double): double;
var
  bx0,bx1: integer;
  rx0,rx1,sx,t,u,v: double;
begin
  t := x + N;
  bx0 := Trunc(t) and BM;
  bx1 := (bx0 + 1) and BM;
  rx0 := t - trunc(t);
  rx1 := rx0 - 1;

  sx := rx0 * rx0 * (3 - 2 * rx0);
  u  := rx0 * G1[P[bx0]];
  v  := rx1 * G1[P[bx1]];
  result := u + sx * (v - u);
end;

function TGenerationForm.noise2D(const x,y : double): double;
var
  bx0,bx1,by0,by1: integer;
  b00,b10,b01,b11: integer;
  rx0, rx1, ry0, ry1: double;
  sx, sy,t,a,b,u,v: double;
begin
  t := x + N;
  bx0 := Trunc(t) and BM;
  bx1 := (bx0 + 1) and BM;
  rx0 := t - trunc(t);
  rx1 := rx0 - 1;

  t := y + N;
  by0 := Trunc(t) and BM;
  by1 := (by0 + 1) and BM;
  ry0 := t - trunc(t);
  ry1 := ry0 - 1;
```

```
  b00 := P[P[bx0]+by0];
  b10 := P[P[bx1]+by0];
  b01 := P[P[bx0]+by1];
  b11 := P[P[bx1]+by1];

  sx := rx0 * rx0 * (3 - 2 * rx0);
  sy := ry0 * ry0 * (3 - 2 * ry0);

  u := rx0 * G2[b00][0] + ry0 * G2[b00][1];
  v := rx1 * G2[b10][0] + ry0 * G2[b10][1];
  a := u + sx * (v - u);

  u := rx0 * G2[b01][0] + ry1 * G2[b01][1];
  v := rx1 * G2[b11][0] + ry1 * G2[b11][1];
  b := u + sx * (v - u);

  result := a + sy * (b - a);
end;

function TGenerationForm.noise2Di(const x,y: Integer): Integer;
var
  bx0,bx1,by0,by1: integer;
  b00,b10,b01,b11: integer;
  rx0, rx1, ry0, ry1: integer;
  sx, sy,t,a,b,u,v: integer;
  i,j: integer;
  r: integer;
begin
  t := x + $40000000;
  bx0 := (t shr 10) and $FF;
  bx1 := (bx0 + 1) and $FF;
  i := P[bx0];
  rx0 := t and 1023;
  rx1 := rx0 - 1024;

  t := y + $40000000;
  by0 := (t shr 10) and $FF;
  by1 := (by0 + 1) and $FF;
  j := P[bx1];
  ry0 := t and 1023;
```

```
  ry1 := ry0 - 1024;

  b00 := P[i + by0];
  b10 := P[j + by0];
  b01 := P[i + by1];
  b11 := P[j + by1];

  sx := (rx0 * rx0 * (3072 - 2 * rx0)) shr 20;
  sy := (ry0 * ry0 * (3072 - 2 * ry0)) shr 20;

  u := (rx0 * G2i[b00][0] + ry0 * G2i[b00][1]);
  v := (rx1 * G2i[b10][0] + ry0 * G2i[b10][1]);
  a := (u shl 10) + sx * (v - u);
asm
  SAR a,20
end;

  u := (rx0 * G2i[b01][0] + ry1 * G2i[b01][1]);
  v := (rx1 * G2i[b11][0] + ry1 * G2i[b11][1]);
  b := (u shl 10)  + sx * (v - u);
asm
  SAR b,20
end;

  r := a shl 10  + sy * (b - a);
asm
  SAR r,10
end;

  result := r;
end;


function TGenerationForm.noise3Di(const x,y,z: Integer): Integer;
var
  bx0,bx1,by0,by1,bz0,bz1: integer;
  b00,b10,b01,b11: integer;
  rx0, rx1, ry0, ry1, rz0, rz1: integer;
  sx, sy, sz, t,a,b,c,d,u,v: integer;
  i,j: integer;
```

```
  r: integer;
begin
  t := x + $40000000;
  bx0 := (t shr 10) and $FF;
  bx1 := (bx0 + 1) and $FF;
  i := P[bx0];
  rx0 := t and 1023;
  rx1 := rx0 - 1024;

  t := y + $40000000;
  by0 := (t shr 10) and $FF;
  by1 := (by0 + 1) and $FF;
  j := P[bx1];
  ry0 := t and 1023;
  ry1 := ry0 - 1024;

  t := z + $40000000;
  bz0 := (t shr 10) and $FF;
  bz1 := (bz0 + 1) and $FF;
  rz0 := t and 1023;
  rz1 := rz0 - 1024;

  b00 := P[i + by0];
  b10 := P[j + by0];
  b01 := P[i + by1];
  b11 := P[j + by1];

  sx := (rx0 * rx0 * (3072 - 2 * rx0)) shr 20;
  sy := (ry0 * ry0 * (3072 - 2 * ry0)) shr 20;
  sz := (rz0 * rz0 * (3072 - 2 * rz0)) shr 20;

  u := rx0 * G3i[b00 + bz0][0] + ry0 * G3i[b00 + bz0][1] + rz0 * G3i[b00 + bz0][2];
  v := rx1 * G3i[b10 + bz0][0] + ry0 * G3i[b10 + bz0][1] + rz0 * G3i[b10 + bz0][2];
  a := (u shl 10) + sx * (v - u);
asm
  SAR a,20
end;

  u := rx0 * G3i[b01 + bz0][0] + ry1 * G3i[b01 + bz0][1] + rz0 * G3i[b01 + bz0][2];
  v := rx1 * G3i[b11 + bz0][0] + ry1 * G3i[b11 + bz0][1] + rz0 * G3i[b11 + bz0][2];
```

```
  b := (u shl 10)  + sx * (v - u);
asm
  SAR b,20
end;

  c := a shl 10 + sy * (b - a);

  u := rx0 * G3i[b00 + bz1][0] + ry0 * G3i[b00 + bz1][1] + rz1 * G3i[b00 + bz1][2];
  v := rx1 * G3i[b10 + bz1][0] + ry0 * G3i[b10 + bz1][1] + rz1 * G3i[b10 + bz1][2];
  a := u shl 10 + sx * (v - u);
asm
  SAR a,20
end;

  u := rx0 * G3i[b01 + bz1][0] + ry1 * G3i[b01 + bz1][1] + rz1 * G3i[b01 + bz1][2];
  v := rx1 * G3i[b11 + bz1][0] + ry1 * G3i[b11 + bz1][1] + rz1 * G3i[b11 + bz1][2];
  b := u shl 10 + sx * (v - u);
asm
  SAR b,20
end;

  d := a shl 10 + sy * (b - a);

  r := c shl 10  + sz * (d - c);
asm
  SAR r,20
end;

  result := r;
end;

function TGenerationForm.noise3D(const x,y,z : double): double;
var
  bx0,bx1,by0,by1,bz0,bz1: integer;
  b00,b10,b01,b11: integer;
  rx0, rx1, ry0, ry1, rz0, rz1: double;
  sx,sy,sz, t,a,b,c,d,u,v: double;
begin
  t := x + N;
  bx0 := Trunc(t) and BM;
```

```
bx1 := (bx0 + 1) and BM;
rx0 := t - trunc(t);
rx1 := rx0 - 1;

t := y + N;
by0 := Trunc(t) and BM;
by1 := (by0 + 1) and BM;
ry0 := t - trunc(t);
ry1 := ry0 - 1;

t := z + N;
bz0 := Trunc(t) and BM;
bz1 := (bz0 + 1) and BM;
rz0 := t - trunc(t);
rz1 := rz0 - 1;

b00 := P[P[bx0]+by0];
b10 := P[P[bx1]+by0];
b01 := P[P[bx0]+by1];
b11 := P[P[bx1]+by1];

sx := rx0 * rx0 * (3 - 2 * rx0);
sy := ry0 * ry0 * (3 - 2 * ry0);
sz := rz0 * rz0 * (3 - 2 * rz0);

u := rx0 * G3[b00 + bz0][0] + ry0 * G3[b00 + bz0][1] + rz0 * G3[b00 + bz0][2];
v := rx1 * G3[b10 + bz0][0] + ry0 * G3[b10 + bz0][1] + rz0 * G3[b10 + bz0][2];
a := u + sx * (v - u);

u := rx0 * G3[b01 + bz0][0] + ry1 * G3[b01 + bz0][1] + rz0 * G3[b01 + bz0][2];
v := rx1 * G3[b11 + bz0][0] + ry1 * G3[b11 + bz0][1] + rz0 * G3[b11 + bz0][2];
b := u + sx * (v - u);

c := a + sy * (b - a);

u := rx0 * G3[b00 + bz1][0] + ry0 * G3[b00 + bz1][1] + rz1 * G3[b00 + bz1][2];
v := rx1 * G3[b10 + bz1][0] + ry0 * G3[b10 + bz1][1] + rz1 * G3[b10 + bz1][2];
a := u + sx * (v - u);

u := rx0 * G3[b01 + bz1][0] + ry1 * G3[b01 + bz1][1] + rz1 * G3[b01 + bz1][2];
```

```
  v := rx1 * G3[b11 + bz1][0] + ry1 * G3[b11 + bz1][1] + rz1 * G3[b11 + bz1][2];
  b := u + sx * (v - u);

  d := a + sy * (b - a);

  result := c + sz * (d - c);
end;

procedure TGenerationForm.initnoise(seed: Integer);
var
  I,J,T: integer;
  len: double;
begin
//  randseed := seed;
  randomize;

  for i := 0 to B - 1 do begin
    P[i] := i;

    G1[i] := (Trunc(Random * 2 * B) - B)/B;

    G2[i,0] := (Trunc(Random * 2 * B) - B)/B;
    G2[i,1] := (Trunc(Random * 2 * B) - B)/B;
    len := sqrt(G2[i,0] * G2[i,0] + G2[i,1] * G2[i,1]);
    if len > 1E-5 then begin
      G2[i,0] := G2[i,0] / len;
      G2[i,1] := G2[i,1] / len;
    end;

    G2i[i,0] := trunc(G2[i,0] * 1024);
    G2i[i,1] := trunc(G2[i,1] * 1024);

    G3[i,0] := (Trunc(Random * 2 * B) - B)/B;
    G3[i,1] := (Trunc(Random * 2 * B) - B)/B;
    G3[i,2] := (Trunc(Random * 2 * B) - B)/B;
    len := sqrt(G3[i,0] * G3[i,0] + G3[i,1] * G3[i,1] + G3[i,2] * G3[i,2]);
    if len > 1E-5 then begin
      G3[i,0] := G3[i,0] / len;
      G3[i,1] := G3[i,1] / len;
      G3[i,2] := G3[i,2] / len;
```

```
    end;

    G3i[i,0] := trunc(G3[i,0] * 1024);
    G3i[i,1] := trunc(G3[i,1] * 1024);
    G3i[i,2] := trunc(G3[i,2] * 1024);
  end;

  for i := 0 to B - 1 do begin
    j := Trunc(Random * B);
    T := P[i];
    P[i] := P[j];
    P[j] := T;
  end;

  for i := 0 to B + 1 do begin
    P[B + i] := P[i];

    G1[B + i] := G1[i];

    G2[B + i][0] := G2[i][0];
    G2[B + i][1] := G2[i][1];

    G2i[B + i][0] := G2i[i][0];
    G2i[B + i][1] := G2i[i][1];

    G3[B + i][0] := G3[i][0];
    G3[B + i][1] := G3[i][1];
    G3[B + i][2] := G3[i][2];

    G3i[B + i][0] := G3i[i][0];
    G3i[B + i][1] := G3i[i][1];
    G3i[B + i][2] := G3i[i][2];
  end;
end;

procedure TGenerationForm.tedtOfXChange(Sender: TObject);
begin
  oX:=tedtOfX.Value;
end;
```

```
procedure TGenerationForm.tedtOfYChange(Sender: TObject);
begin
  oY:=tedtOfY.Value;
end;

function TGenerationForm.turbulence1D(x: double; const n: integer): double;
var
  freq: double;
  i: integer;
begin
  result := 0;
  freq := 1.0;

  for i := n - 1 downto 0 do begin
    result := result + Noise1D(x) * freq;
    x := x * 2;
    freq := freq * 0.5;
  end;
end;

function TGenerationForm.turbulence2D(x,y: double; const n: integer): double;
var
  freq: double;
  i: integer;
begin
  result := 0;
  freq := 1.0;

  for i := n - 1 downto 0 do begin
    result := result + abs(Noise2D(x,y)) * freq;
    x := x * 2;
    y := y * 2;
    freq := freq * 0.5;
  end;
end;

function TGenerationForm.turbulence2Di(x,y: Integer; const n: integer): Integer;
var
  r, i: integer;
  a: integer;
```

```
begin
  r := 0;
  a := 1;

  for i := n - 1 downto 0 do begin
    Inc(r, Noise2Di(x,y) div a);
    x := x shl 1;
    y := y shl 1;
    a := a shl 1;
  end;
  Result := abs(r);
end;

function TGenerationForm.turbulence2Dia(x,y: Integer; const n: integer): Integer;
var
  r, i: integer;
  a: integer;
begin
  r := 0;
  a := 1;

  for i := n - 1 downto 0 do begin
    Inc(r, abs(Noise2Di(x,y)) div a);
    x := x shl 1;
    y := y shl 1;
    a := a shl 1;
  end;
  Result := r;
end;

function TGenerationForm.turbulence3D(x,y,z: double; const n: integer): double;
var
  freq: double;
  i: integer;
begin
  result := 0;
  freq := 1.0;

  for i := n - 1 downto 0 do begin
    result := result + Noise3D(x,y,z) * freq;
```

```
    x := x * 2;
    y := y * 2;
    z := z * 2;
    freq := freq * 0.5;
  end;
  result := abs(Result)
end;

function TGenerationForm.turbulence3Di(x,y,z: Integer; const n: integer): Integer;
var
  r, i: integer;
  a: integer;
begin
  r := 0;
  a := 1;

  for i := n - 1 downto 0 do begin
    Inc(r, Noise3Di(x,y,z) div a);
    x := x shl 1;
    y := y shl 1;
    z := z shl 1;
    a := a shl 1;
  end;
  Result := abs(r);
//  Result := r;
end;

procedure TGenerationForm.CreatePalette;
var
  i: integer;
begin
  for i := 0 to 255 do begin
     Pal[i].rgbRed   := i;
    Pal[i].rgbGreen := i;
    Pal[i].rgbBlue  := round(sqrt(i/255) * 255);
  end;
end;
end.
```

Додаток В

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ ГЕНЕРАЦІЇ
ЛАНДШАФТІВ ЗА ДОПОМОГОЮ ПРОЦЕДУРНИХ ШУМІВ

Виконав: студент 2-го курсу,
групи 2КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Волинець Б.А.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О.В.
(прізвище та ініціали)
«21» грудня 2022 р.

Вінниця ВНТУ - 2022 рік

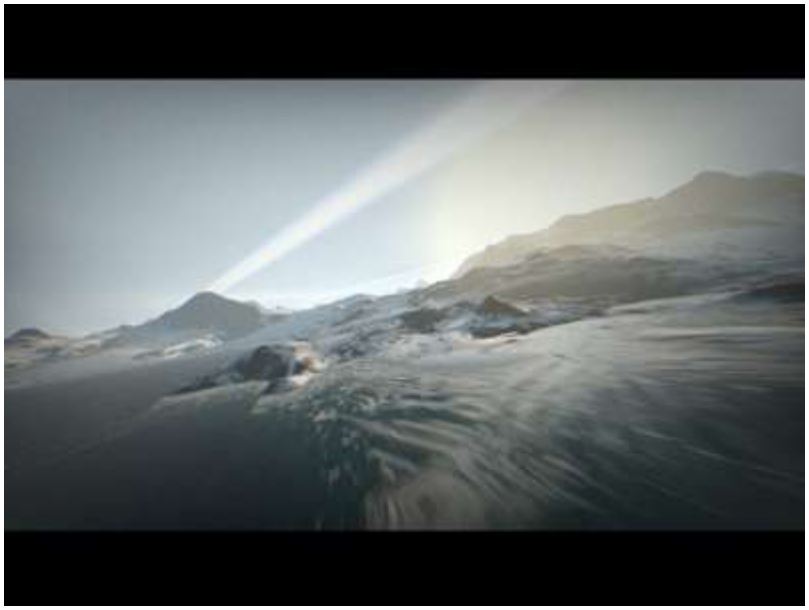


Рисунок В.1 - Ландшафт і текстура згенеровані з використанням процедурного шуму

Рисунок В.2 – Структурна схема роботи інформаційної технології для генерації ландшафтів за допомогою процедурних шумів

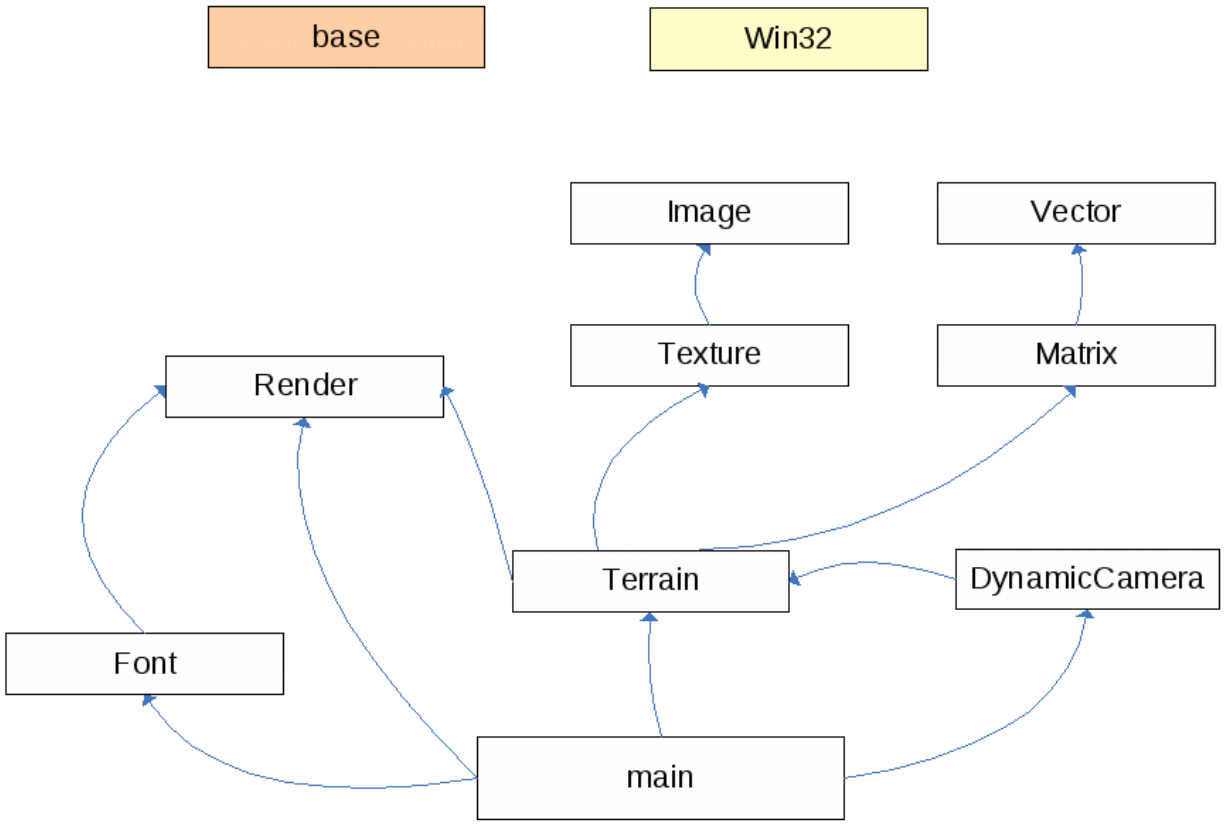


Рисунок В.3 – Діаграма компонентів додатку

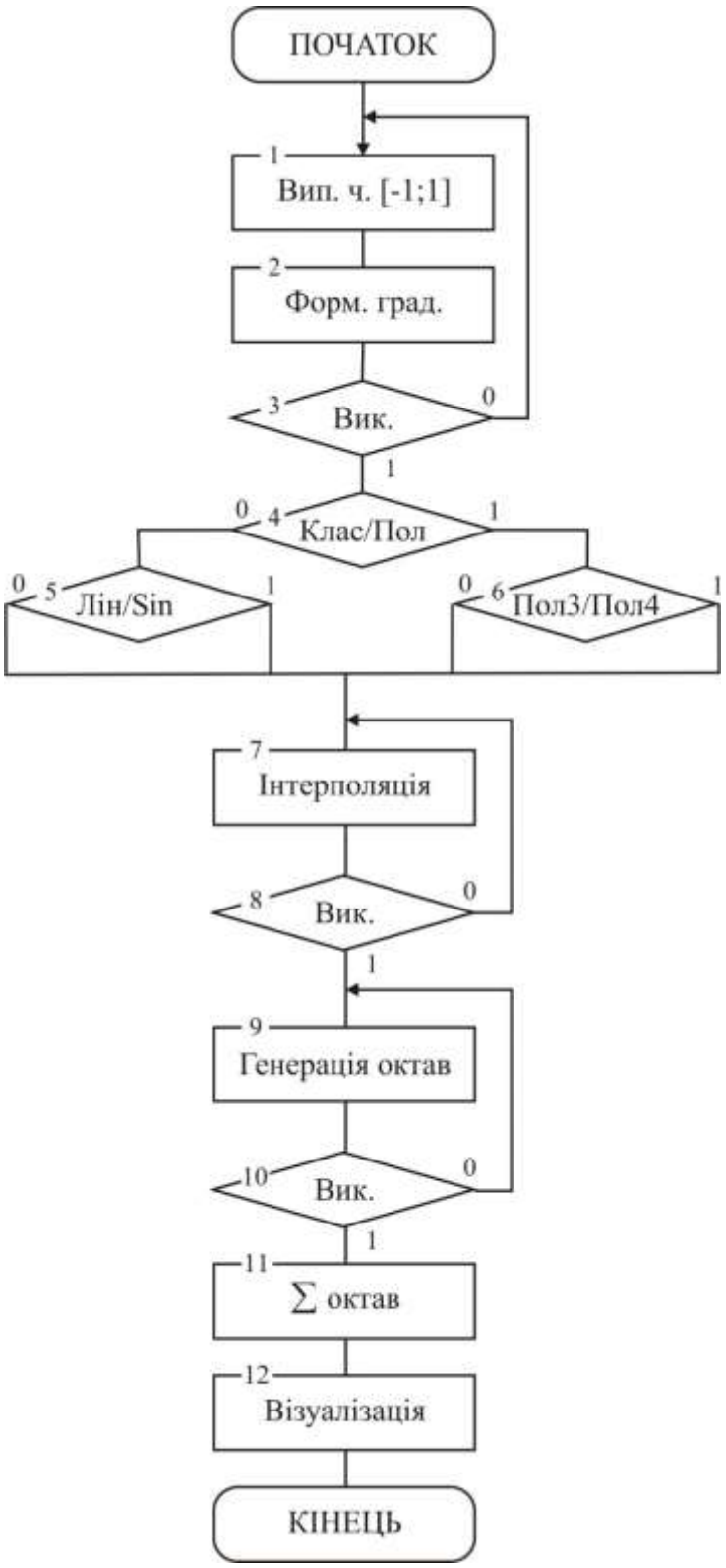


Рисунок В.4 - Алгоритм генерації одновимірного Шуму Перліна

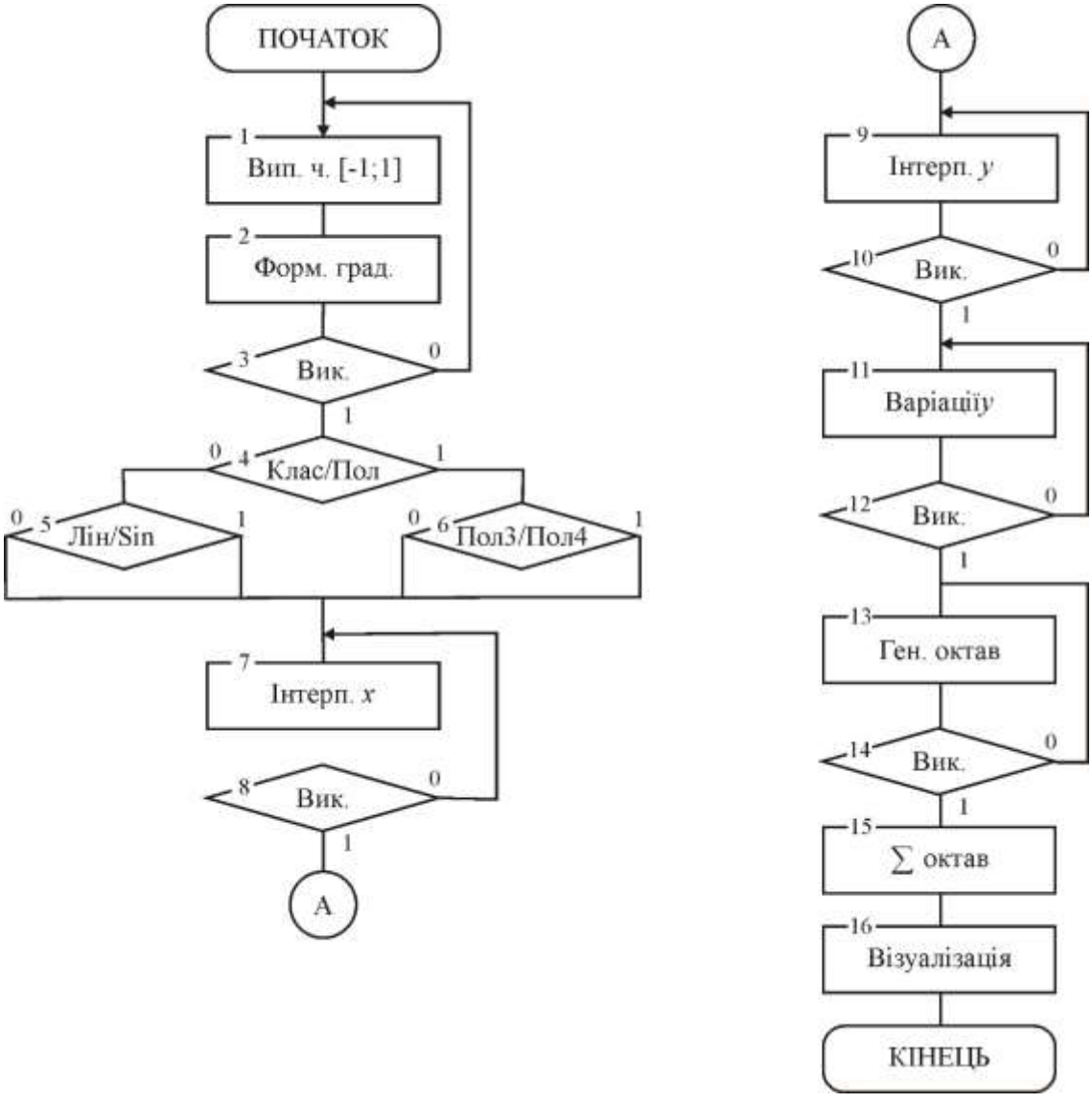


Рисунок В.5 - Алгоритм генерації двовимірного Шуму Перліна

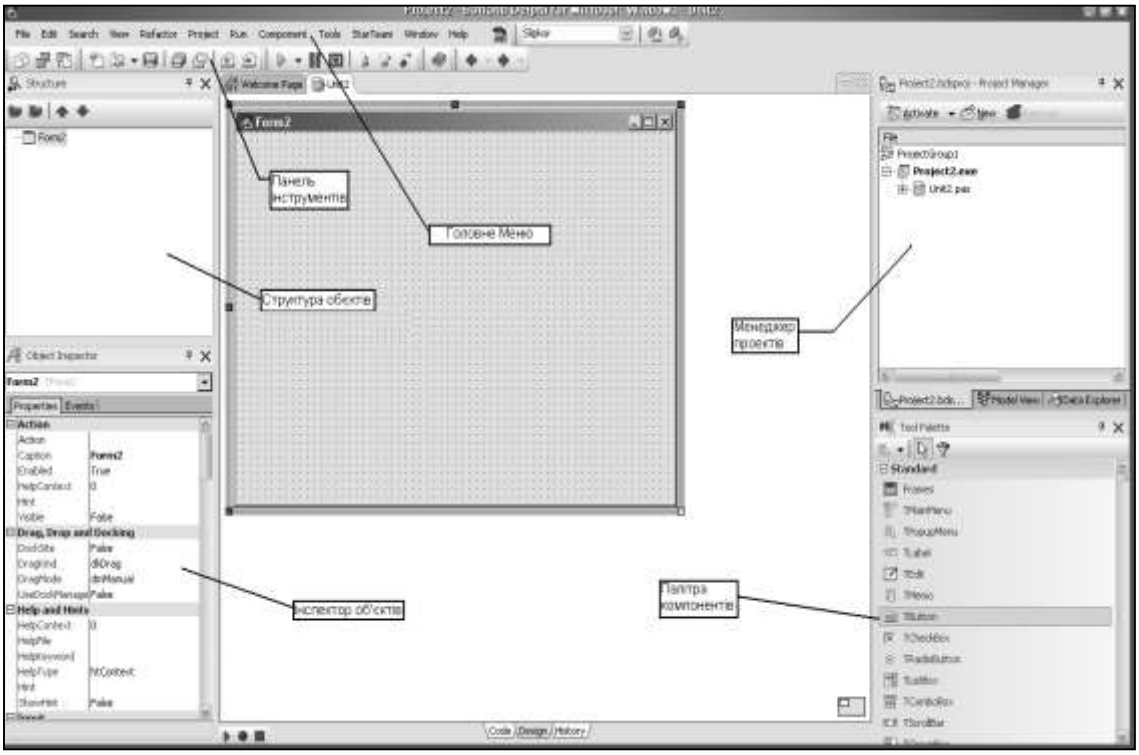


Рисунок В.6 – Головне меню та панель інструментів

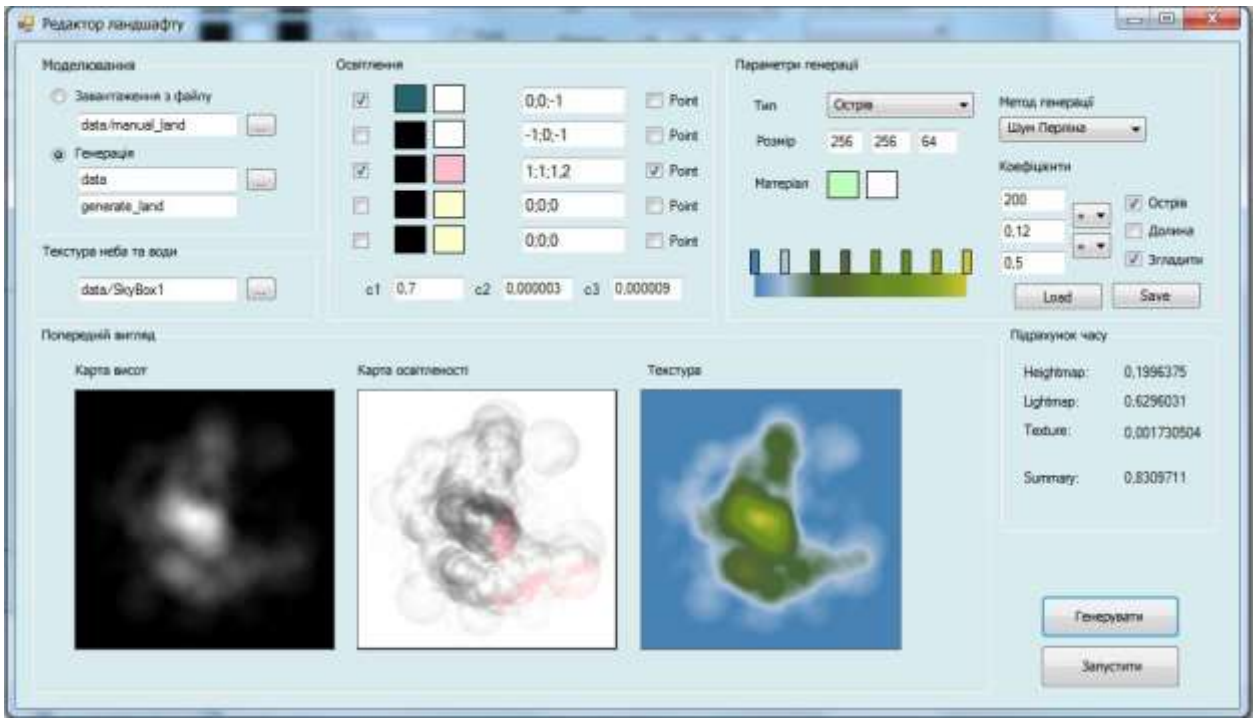


Рисунок В.7 – Користувацький інтерфейс