Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія створення аморфних об’єктів для фотореалістичних зображень»

Виконав: студент 2 курсу, групи 1КН-19м
спеціальності 122 «Комп'ютерні науки»
Стрижалов О. І.

Керівник: канд. техн. наук, доцент Сілагін О. В.

Рецензент: доктор. техн. наук, професор Ліщинська Л.Б.

Вінниця, 2020 рік

ЗАТВЕРДЖУЮ

Заступник директора ТОВ «ІТІ»

Бодяк В. М.

(підпис)

“__.__.__” _____2020 р.

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

д.т.н., проф.. Яровий А.А.

(підпис)

“___.__._” _____2020 р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.013.19.000.ПЗ

Магістранта групи 1КН-19м Стрижалова Олександра Ігоровича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія створення аморфних об’єктів для фотореалістичних зображень»

Вхідні дані: розмір зображення 720*1280, формат збереженого зображення .bmp, мова програмування PascalABC.Net.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: структура інформаційної технології; схема загального алгоритму; UML-діаграма компонентів програмного засобу; головне вікно програмного засобу; приклад роботи інформаційної технології; порівняльний графік тривалості побудови зображень залежно від розміру зображення звичайного та вдосконаленого алгоритму Diamond-Square.

2. Текстова (пояснювальна записка): вступ, аналіз предметної області створення роботи інформаційної технології створення аморфних об’єктів для фотореалістичних зображень, розробка інформаційної технології створення аморфних об’єктів для фотореалістичних зображень, програмна реалізація інформаційної технології створення аморфних об’єктів для фотореалістичних зображень, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасних методів створення інформаційної технології створення аморфних об'єктів для фотореалістичних зображень. Постановка задач дослідження			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1
2	Побудова алгоритму інформаційної технології створення аморфних об'єктів для фотореалістичних зображень			Блок-схеми алгоритмів, розділ 2
3	Практичне застосування та оцінка ефективності розробленої інформаційної технології			розділ 3
4	Підготовка економічної частини			розділ 4
5	Апробація та/або впровадження результатів дослідження			тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ к.т.н, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
"____" _____20___ р. О. В. Сілагін
ініціали та прізвище

2. Економічна частина _____ к.е.н., доц, доц. кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
"____" _____20___ р. М. В. Бальзан
ініціали та прізвище

Дата попереднього захисту роботи "____" _____20___ р.

Рецензент _____ д.т.н. проф, проф. кафедри ПЗ
(підпис) наук. ступінь, вчене звання (посада)
Ліщинська Л. Б.
ініціали та прізвище

Завдання видав
науковий керівник _____ к.т.н, доц., доц., кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
О. В. Сілагін
ініціали та прізвище

"____" _____20___ р.

Завдання отримав магістрант _____ О. І. Стрижалов
(підпис) ініціали та прізвище

"____" _____20___ р.

АНОТАЦІЯ

Магістерська кваліфікаційна робота присвячена розробці інформаційної технології створення аморфних об’єктів для фотореалістичних зображень на основі алгоритму Diamond-Square. Дана технологія забезпечує створення аморфних об’єктів для фотореалістичних зображень.

В ході роботи проведено аналіз предметної області створення аморфних об’єктів для фотореалістичних зображень. Розглянуто аналоги для створення аморфних об’єктів для фотореалістичних зображень. Удосконалено структуру алгоритму таким чином, щоб збільшити швидкість створення аморфних об’єктів для фотореалістичних зображень. Розроблено інформаційну технологію створення аморфних об’єктів для фотореалістичних зображень на основі алгоритму Diamond-Square та реалізовано програмний засіб на мові програмування PascalABC.Net.

ABSTRACT

The master's qualification work is devoted to the development of information technology for the creation of amorphous objects for photorealistic images based on the Diamond-Square algorithm. This technology provides the creation of amorphous objects for photorealistic images.

In the course of the work the analysis of the subject area of creation of amorphous objects for photorealistic images is carried out. Analogs for creating amorphous objects for photorealistic images are considered. The structure of the algorithm has been improved to increase the speed of creating amorphous objects for photorealistic images. The information technology for creating amorphous objects for photorealistic images based on the Diamond-Square algorithm has been developed and a software tool in the PascalABC.Net programming language has been implemented.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. На сьогоднішній день відбувається розвиток технологій в усіх галузях діяльності людства, в тому числі комп'ютерна графіка. Сучасні засоби створення зображень з допомогою обчислювальної техніки дозволяють знімати фільми, про які не так давно навіть ніхто не мріяв та розробляти комп'ютерні ігри, в яких зображення настільки якісні, що їх можна сприймати як реальний світ. Звичайно новий рівень будь-яких технологій тягне за собою зростання потреб, які вимагають підвищення якості технології. Справедливе це твердження і для комп'ютерної графіки. Очевидно що зростання якості та реалістичності зображень відображається на об'ємах пам'яті або швидкості їх створення. Тому щоб ощадливо використовувати час роботи та пам'ять обчислювальної техніки потрібно покращувати технології. Якщо звернути увагу на сучасні фантастичні фільми та комп'ютерні ігри, то можна зробити висновок, що значна частка зображення – це аморфні об'єкти. Характерною рисою яких є відносна не важливість їх форми та внутрішньої структури. Також це можна помітити в деяких текстурах. Як відомо багато речей, предметів та явищ природи мають фрактальну природу коли частина подібна цілому. Тому для вирішення проблем створення якісних реалістичних зображень аморфних об'єктів можна використовувати фрактальну графіку. Зважаючи на те, що необхідно створювати велику кількість зображень, які повинні бути не схожими між собою доцільно використовувати випадкові величини. В даний час існує кілька високоякісних реалізацій для генерації аморфних об'єктів для фотореалістичних зображень за допомогою процедурних шумів, серед яких досить відомі шум Перліна, симплексний шум, шум Уорлі, хвильовий шум, градієнтний шум та алгоритм Diamond-Square, який також базується на процедурних шумах. Однак достатньо висока обчислювальна складність багатьох фрактальних алгоритмів підтверджує доцільність їх покращення для

збільшення швидкодії створення зображень. Саме тому дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології створення аморфних об’єктів для фотореалістичних зображень, швидкодія якої буде підвищена шляхом модифікації алгоритму.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка спеціалізованих засобів штучного інтелекту на основі інтелектуального аналізу даних та машинного навчання».

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є підвищення швидкодії створення аморфних об’єктів для фотореалістичних зображень за допомогою зменшення кількості випадкових зміщень середніх точок.

Для досягнення поставленої мети слід розв’язати такі завдання:

- розглянути та проаналізувати існуючі програмні реалізації розв’язання задачі створення аморфних об’єктів для фотореалістичних зображень;
- запропонувати математичну модель для інформаційної технології створення аморфних об’єктів для фотореалістичних зображень;
- навести стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології створення аморфних об’єктів для фотореалістичних зображень;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес створення аморфних об’єктів для фотореалістичних зображень.

Предмет дослідження – інформаційна технологія створення аморфних об’єктів для фотореалістичних зображень.

Область застосування – створення аморфних об’єктів для фотореалістичних зображень та анімацій.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи оброблення цифрової інформації, теорія фракталів та процедурних шумів для реалізації інформаційної технології створення аморфних об’єктів для фотореалістичних зображень, методи математичної статистики для обрахунків результатів отриманих за допомогою програмного засобу, програмування на мовах високого рівня.

Наукова новизна одержаних результатів полягає в покращенні технології створення аморфних об’єктів для фотореалістичних зображень за допомогою вдосконаленого алгоритму Diamond-Square, що відрізняється від відомих реалізацій зменшенням кількості зміщень середніх точок, що забезпечило збільшення швидкості створення аморфних об’єктів для фотореалістичних зображень.

Практичне значення одержаних результатів:

1. Розроблено алгоритм створення аморфних об’єктів для фотореалістичних зображень з використанням процедурних шумів.

2. Розроблено програмний продукт для створення аморфних об’єктів для фотореалістичних зображень з використанням процедурних шумів.

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, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

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

Апробація результатів роботи. Результати досліджень апробовані на XLIX науково-технічній конференції підрозділів ВНТУ, XII міжнародній науково-практичній конференції «ІОН-2020» [1, 2].

Публікації. За результатами досліджень опубліковано тезу доповіді міжнародної науково-практичної конференції [1, 2].

1 АНАЛІЗ СУЧАСНОГО РІВНЯ РОЗВИТКУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ СТВОРЕННЯ ФОТОРЕАЛІСТИЧНИХ ЗОБРАЖЕНЬ

1.1 Аналіз задачі створення фотореалістичних зображень

Які тільки можливості не надають сьогодні персональні комп'ютери! За умови установки на них відповідних програмних засобів вони самостійно генерують паролі і шифрують, пишуть музику і вірші, створюють зображення і анімацію і т.п. Звичайно, не все у них виходить ідеально, тому повністю замінити користувачів їм навряд чи вдасться, але варто визнати, що раціональне зерно в тому, щоб комп'ютери самостійно генерували, наприклад, зображення, є.

За допомогою генераторів зображень можна по-перше, автоматизувати ряд стандартних процесів. Особливо корисно використання генераторів при розробці текстур і різноманітних Web-елементів. Наприклад, професійні кнопки з їх допомогою ви зможете створити в лічені секунди. По-друге, генератори зображень можуть непогано виручити в тому випадку, якщо малювати ви не вмієте (не всі ж народжуються художниками!), А створити те чи інше зображення (мова не йде, звичайно, про шедеври Леонардо да Вінчі або Пікассо) вам необхідно . Тому сьогодні ми і поговоримо про програми, що генерують зображення.

Правда, відразу варто визнати, що дати чітке визначення поняттю «генератор зображень» вельми складно, оскільки самі розробники відносять до даної категорії ПО найрізноманітніший софт. Очевидно тільки одне: це програма, за допомогою якої можна повністю автоматично або з деякою корекцією якихось налаштувань і параметрів порівняно швидко створити зображення без стомлюючого малювання. А ось далі починаються нюанси.

Одні генератори дозволяють отримувати зображення на основі шаблонів. Здебільшого ці пакети орієнтовані на розробку елементів Web-

дизайну, але є і виключення, наприклад в особі Images Generator, спектр застосування якого охоплює найрізноманітніші сфери, починаючи від Web-дизайну і закінчуючи створенням ескізів деталей.

До інших генераторів, що створює свої графічні шедеври шляхом математичних розрахунків, відносяться різноманітні пакети зі створення фрактальної графіки, причому з використанням фракталів можуть будуватися не тільки якісь ірреальні зображення, а й цілком реалістичні. Використовувати отримані таким шляхом зображення можна в самих різних областях - від створення звичайних текстур до фантастичних ландшафтів для комп'ютерних ігор або книжкових ілюстрацій.

Третю групу представляють генератори ландшафтів - вони призначені для створення реалістичних пейзажів і в кінцевому рахунку відносяться вже до сфери 3D-графіки, але вміють створювати і незвичайні по красі природні або космічні пейзажі, які нічим не відрізняються від звичайних високоякісних фотографій. До речі, що зберігаються у багатьох користувачів шпалери для робочого столу, що представляють собою природні пейзажі, в більшості своїй створені за допомогою професійних генераторів ландшафтів, а не за допомогою фотоапарата. Сфера застосування змодельованих таким способом зображень надзвичайно широка і не обмежується заставками і шпалерами для робочого столу або зберіганням в фотоколекції - вони можуть стати фоном для подальшої роботи в середовищі графічного пакета при створенні, наприклад, фотомонтажу або фотоколлажа, послужити основою для різноманітних 3D-сцен і т.п. Дуже складний принцип роботи генераторів ландшафтів заснований на розумінні ними внутрішньої структури природного пейзажу і знанні географічних даних різноманітних екосистем, причому принципи реалізації можуть бути найрізноманітніші. Багато генератори ландшафтів базуються на використанні прийнятої в картографії так званої карти висот, але існують і інші варіанти. Тут ми не будемо вдаватися в подробиці, оскільки генерація ландшафтів - тема окремої

серйозної статті, а розглянемо лише найпростіші з них в контексті з іншими типами генераторів зображень. Для створення фотореалістичних зображень необхідно використовувати засоби, які здатні відтворити форму реальних об’єктів.

Фрактали - це геометричні об'єкти з незвичайними властивостями: будь-яка частина фрактала містить його зменшене зображення. Тобто, скільки фрактал не збільшувати, з будь-якої його частини на вас буде дивитися його маленька копія.

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 [3].

Розроблені й успішно застосовуються три основних принципи представлення графічних зображень – растрова, векторна та фрактальна графіка. В основі кожного способу лежать математичні моделі зображень. Для растрової графіки – це масив (матриця) чисел, що описують координати і колірні параметри кожної точки малюнка, для векторної графіки – математичні формули, що описують геометричні фігури (об’єкти) зображення.

Фрактальна графіка оперує математичними формулами, які описують процес автоматичної генерації зображення за допомогою рівнянь.

Растрову графіку застосовують, в основному, при розробці електронних (мультимедійних) і поліграфічних видань. Для кодування малюнок розбивають на невеликі одноколірні частини. Усі кольори, використані в зображенні, нумерують і для кожної частини записують номер її кольору. Запам'ятавши послідовність розташування частин і номер кольору частини, можна однозначно описати будь-який малюнок. Ілюстрації,

виконані засобами растрової графіки, рідко створюють вручну за допомогою комп'ютерних програм. Тому більшість графічних редакторів, призначених для роботи з растровими ілюстраціями, орієнтовані не стільки на створення зображень, скільки на їхню обробку. Програмні засоби для роботи з векторною графікою призначені для створення зображень і, в меншій мірі, для їх обробки. У векторному способі кодування геометричні фігури, криві і прямі лінії, що складають малюнок, зберігаються в пам'яті комп'ютера у вигляді математичних формул і геометричних абстракцій, таких як: коло, квадрат, еліпс і подібні фігури. За допомогою математичних формул можна описати найрізноманітніші фігури. Саме векторна графіка використовується в системах автоматизованого проектування для виготовлення та редагування різноманітної технічної документації при виконанні проектно-конструкторських робіт . Векторна графіка широко використовується у рекламних агентствах, дизайнерських бюро, редакціях і видавництвах. Дизайнерські роботи, що базуються на застосуванні шрифтів і найпростіших геометричних елементів, легше вирішуються засобами векторної графіки.

Програмні засоби для роботи з фрактальною графікою призначені для автоматичної генерації зображень шляхом математичних розрахунків.

Створення фрактальної художньої композиції з програмування. Фрактальну графіку рідко застосовують для створення друкованих чи електронних документів. Здатність фрактальної графіки моделювати образи живої природи обчислювальним шляхом часто використовують для автоматичної генерації незвичних ілюстрацій .У машинній графіці фрактали використовують при зображені дерев, кущів, моделюванні рельєфу місцевості чи поверхні моря і т.п. Фактично, знайдений спосіб легкого представлення складних, з точки зору геометрії, об’єктів, образи яких дуже подібні на природні, тому що фрактальними властивостями володіють багато об’єктів живої і неживої природи [4].

В даний час існує кілька високоякісних реалізацій для генерації аморфних об’єктів для фотореалістичних зображень за допомогою фрактальних шумів, серед яких досить відомі шум Перліна [5], алгоритм Diamond-Square[6], який також базується на процедурних шумах, хвильовий шум [7], симплексний шум[8], шум Уорлі[9], та градієнтний шум [10]. В користь необхідності застосування фрактальної графіки при створенні фотореалістичних зображень можна сказати, що фрактальна графіка найкраще підходить для створення реалістичних зображень об’єктів живої та неживої природи, для яких характерна самоподібність. Також серед переваг фрактальної графіки можна відзначити здатність до практично необмеженої масштабованості без значної втрати якості зображення та невеликий обсяг пам’яті порівняно з растровою графікою. Проте потрібен час для створення зображень за допомогою алгоритму. У випадку коли зображень небагато та вони статичні растрова графіка може забезпечити високий рівень фотореалістичності, але наприклад в сучасних комп’ютерних іграх або фантастичних фільмах велика кількість текстур та рухомих об’єктів для зображення яких за допомогою растрової графіки знадобиться велика кількість пам’яті та часу, або значна втрата якості зображень. В такому випадку необхідно використовувати засоби фрактальної графіки, які здатні детально відтворювати об’єкти живої та неживої природи, а крім того не потребують значних обсягів пам’яті.

1.2 Аналітичний огляд існуючих аналогів

Фрактальний шум або Шум Перліна [5] широко використовується в двомірній і тривимірній комп'ютерній графіці для створення таких візуальних ефектів, як дим, хмари, туман, вогонь і так далі. Він також дуже часто використовується як проста текстура, що покриває геометричну модель. На відміну від растрових текстур, шум Перліна є процедурною

текстурою, і тому він не займає пам'ять, але разом з тим виконання алгоритму вимагає якихось обчислювальних ресурсів. Використання шуму Перліна дуже поширене в демосценах. Також на такому алгоритму працюють, практично всі ігрові застосування, в яких робляться спроби відобразити небо і згенерувати ландшафт.

Шум Перліна був створений Кеном Перліном в 1983 році і згодом був названий на честь свого творця. Перлін створив свій алгоритм, працюючи в Mathematical Applications Group, Inc.

Використання Perlin Noise для моделювання ландшафту і хмар є найбільш поширеним способом із-за його можливості масштабувати одно-двовимірного і тривимірного вигляду.

Взагалі кажучи, хмарна і ландшафтна поверхня володіє нерівномірною структурою, але є хаотичною. Звідси, для формування хмар і ландшафту необхідна якась хаотична функція, ядро якої підкоряється деякому певному закону. Цим умовам цілком задовольняють функції Perlin Noise, які є суть комбінація шумової функції (показана на рисунку 1.1) і інтерполяційної функції (показана на рисунку 1.2). Аргументами для шумової функції виступають цілі числа.

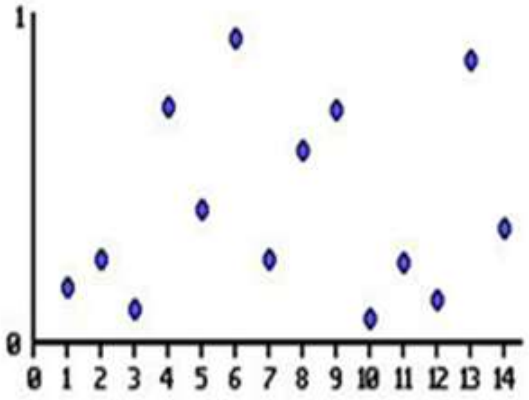

Рисунок 1.1 – Шумова функція

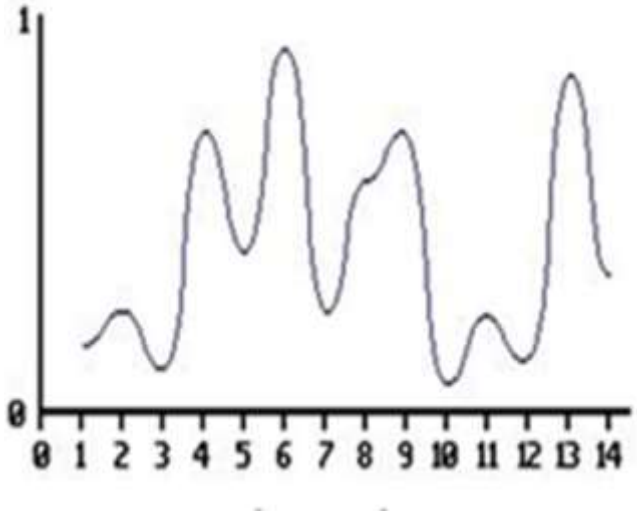

Рисунок 1.2 – Інтерполяційна функція

В результаті послідовного застосування цих двох функцій виходить, як показано на малюнках, функція, що володіє гладкістю і достатнім ступенем хаотичності.

Основними величинами тут, як і в синусній функції, яка показана на рисунку 1.3, є амплітуда і частота . Амплітуда відповідає за висоту хвилі, частота рівна зворотному від довжини хвилі, довжина хвилі - це, у випадку шумової функції, яка зображена на рисунку 1.4, відстань від однієї червоної точки до іншої.

Рисунок 1.3 – Приклад створення синусної функції

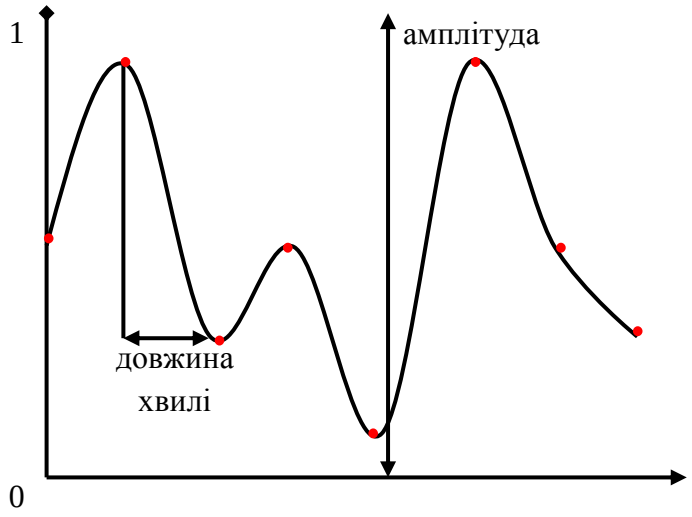

Рисунок 1.4 – Приклад створення шумової функції

Але якщо фіксувати амплітуду і частоту, то отримувана функція все ще володіє достатнім ступенем рівномірністю. Рішення наступне: ми будуємо декілька функцій з різними амплітудами і частотами що показано на рисунках 1.5 – 1.10, а потім додаємо їх значення. Отримувана в результаті функція,яка зображена на рисунку 1.11 якраз задовольняє всім необхідним умовам.

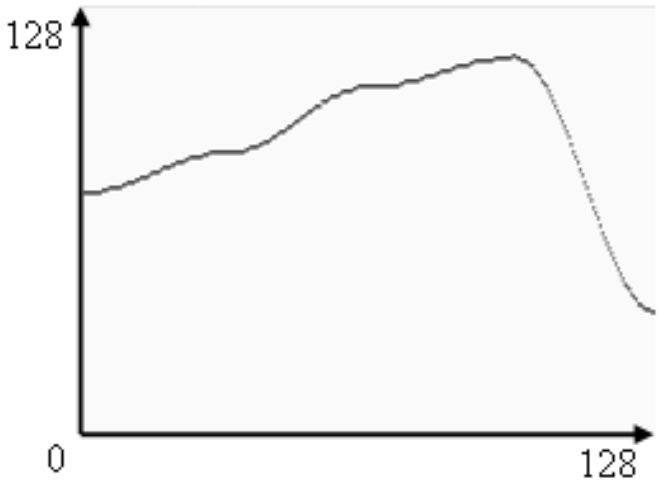

Рисунок 1.5 – Шумова функція зі значенням амплітуди – 128 і значенням частоти – 4

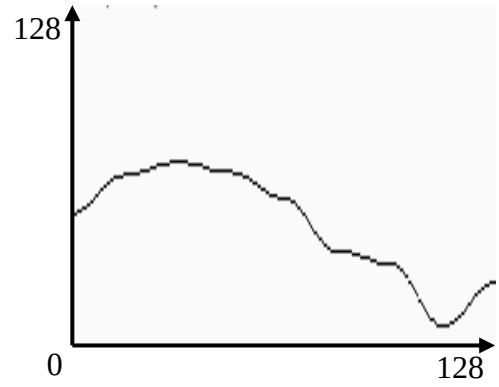

Рисунок 1.6 – Шумова функція зі значенням амплітуди – 64 і значенням частоти – 8

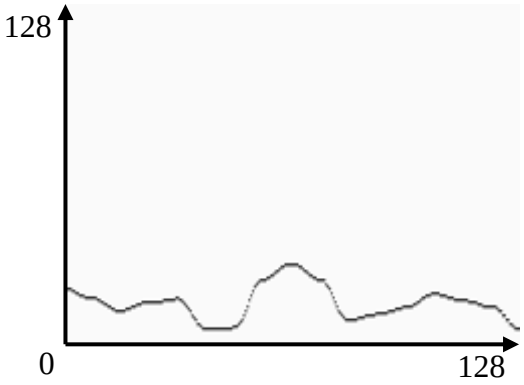

Рисунок 1.7 – Шумова функція зі значенням амплітуди – 32 і значенням частоти – 16

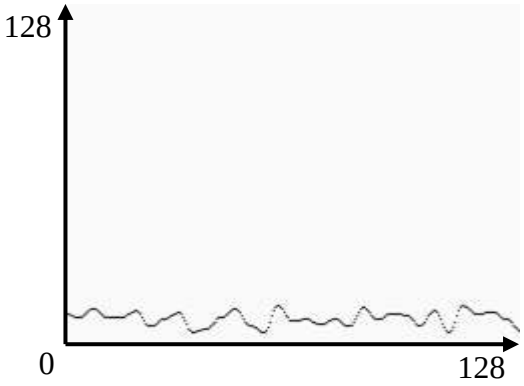

Рисунок 1.8 – Шумова функція зі значенням амплітуди – 16 і значенням частоти – 32

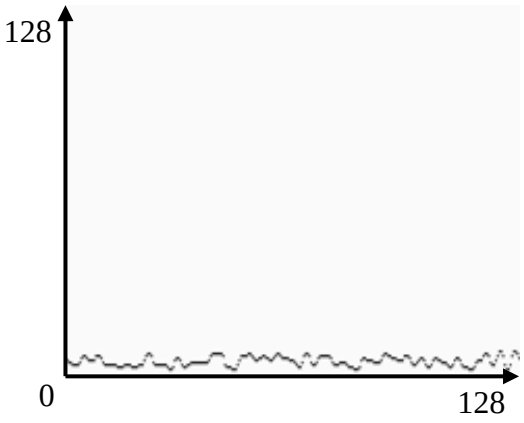

Рисунок 1.9 – Шумова функція зі значенням амплітуди – 8 і значенням частоти – 64

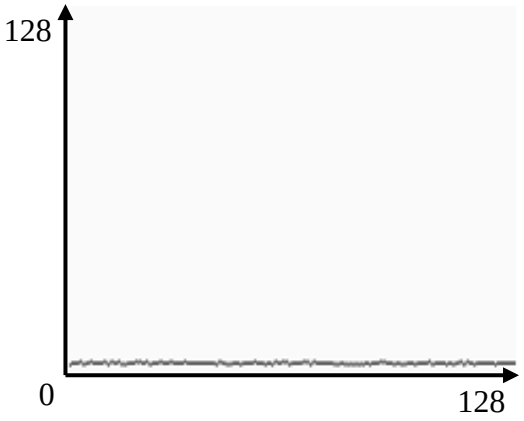

Рисунок 1.10 – Шумова функція зі значенням амплітуди – 4 і значенням частоти – 128

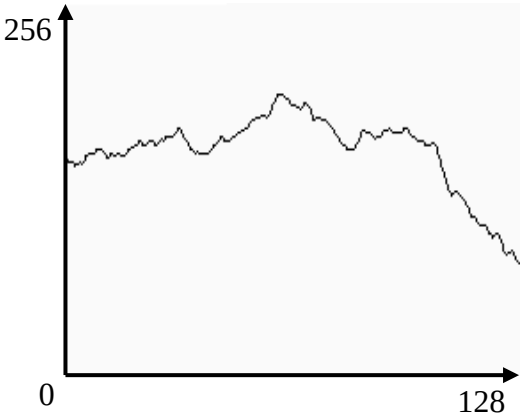

Рисунок 1.11 – Сумарна функція

Амплітуду і частоту на кожному кроці змінюють за наступними формулами:

$$F = 2^i \tag{1.1}$$

$$A = p^i \tag{1.2}$$

де F – частота;

A – амплітуда;

i – номер функції, що генерується;

p – спеціально введена величина для забезпечення залежності амплітуди від частоти.

Створення зображення за допомогою шумів Перліна відбувається аналогічним чином: створюються функції з заданими частотами та амплітудами, згідно формул 1.1 та 1.2, ці функції називають октавами та додають в загальний потік. Зображення, які можна отримати з кожної окремої октави продемонстровані на рисунках 1.12 – 1.17, а сумарне зображення показано на рисунку 1.18. Кількість октав, також, є основною величиною в даному процесі і впливає на якість зображення [5].

Рисунок 1.12 – Зображення першої октави

Рисунок 1.13 – Зображення другої октави

Рисунок 1.14 – Зображення третьої октави

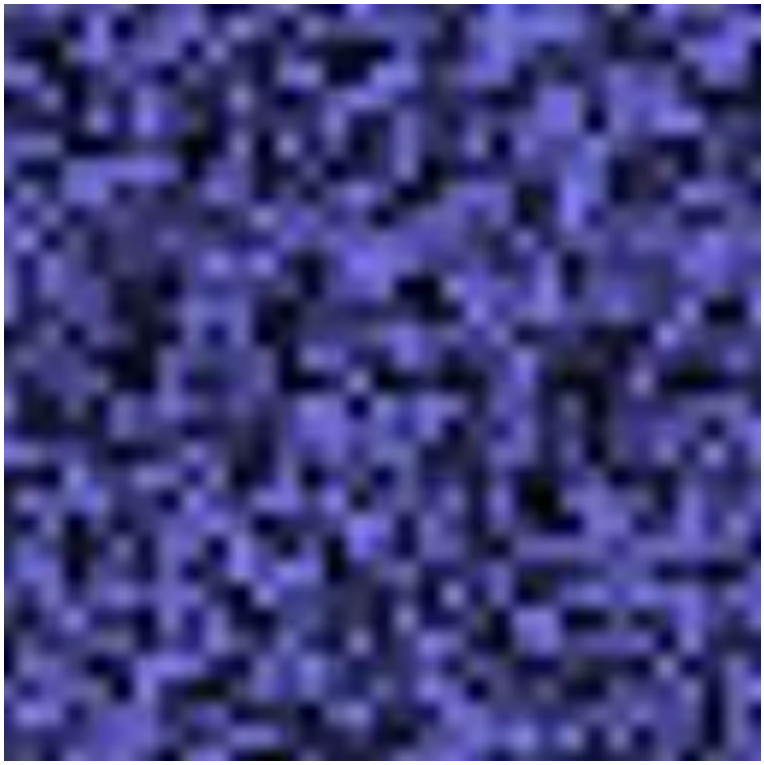

Рисунок 1.15 – Зображення четвертої октави

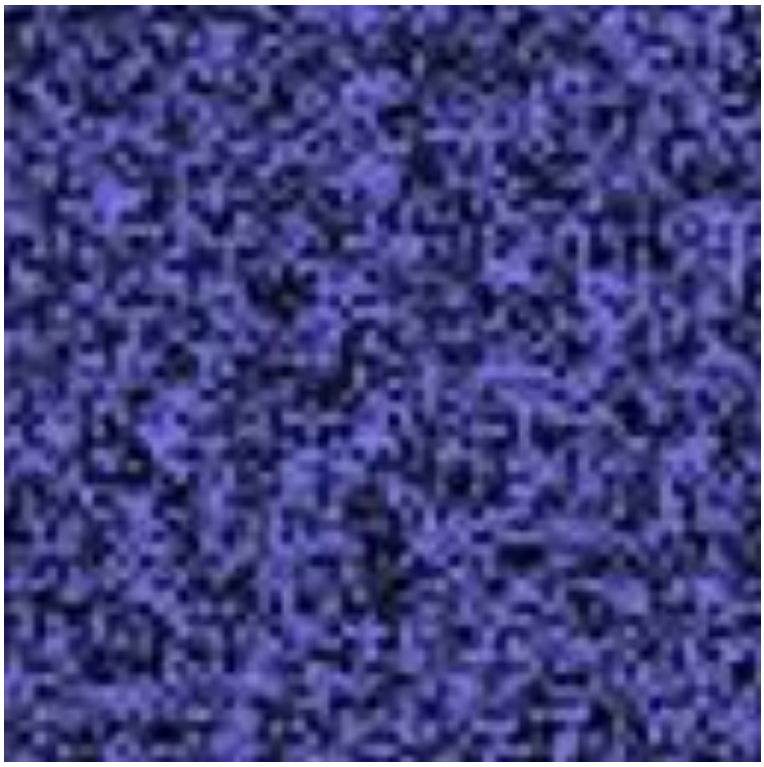

Рисунок 1.16 – Зображення п’ятої октави

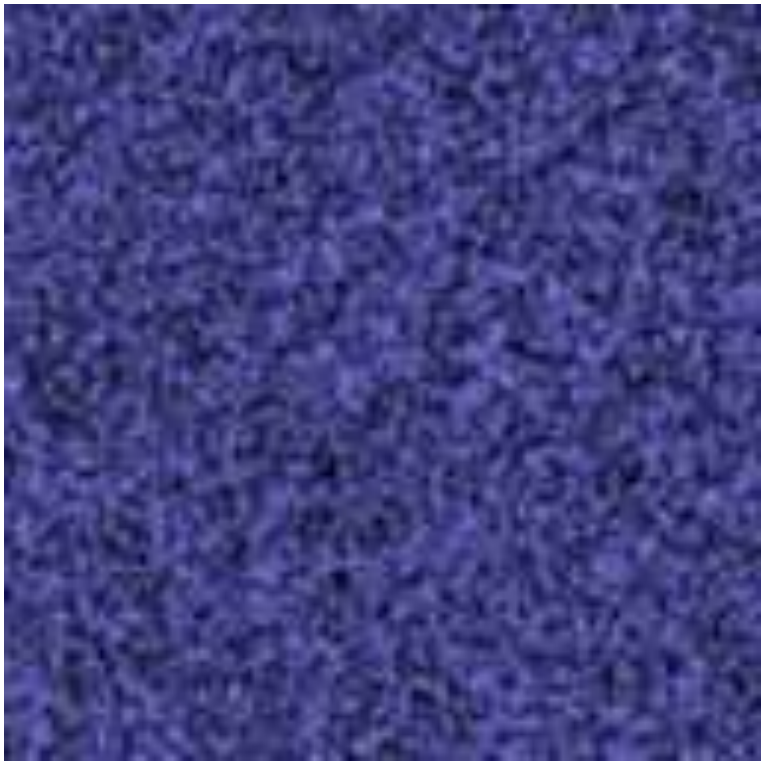

Рисунок 1.17 – Зображення шостої октави

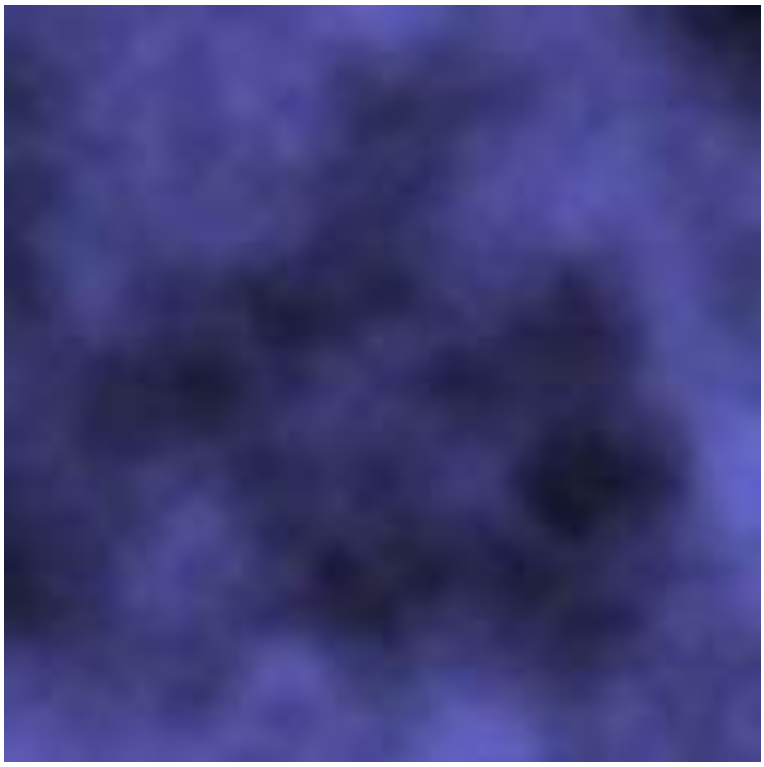

Рисунок 1.18 – Сумарне зображення шести октав

Октавою (octave) називається кожна згенерована і додана в сумарний потік функція. Кількість октав, також, є основною величиною в даному процесі і впливає на якість зображення [5].

Алгоритм diamond-square

Одним із самих поширених і таких, що дають одні з найбільш реалістичних результатів є алгоритм diamond-square, розширення одновимірного алгоритму midpoint displacement на двовимірну площину. Зображення, що виходять з його допомогою, як правило, називають фрактальними.

Почнемо з простішого алгоритму midpoint displacement. Як уже сказано, він працює не на двовимірної площині, а на одновимірному відрізку.

Те, що ріднить цей алгоритм з фракталами – це його рекурсивна поведінка. Спочатку ми будь-яким чином задаємо значення на кінцях відрізка (h1 і h5) і розбиваємо його крапкою (h3) посередині на два менших відрізка,що показано на рисунку 1.19.

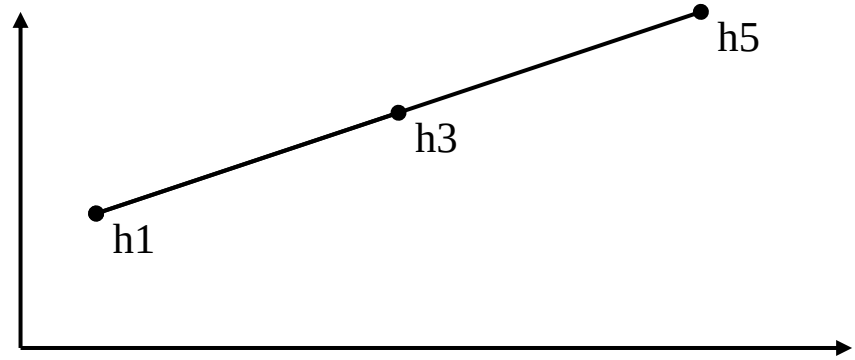

Рисунок 1.19 – Розділення відрізка на два менших відрізка

Цю точку ми зміщуємо на випадкову величину, що показано на рисунку 1.20 і повторюємо розбиття і зміщення для кожного з отриманих менших відрізків що показано на рисунку 1.21. І так далі – поки відрізки не стануть довжиною в один піксель.

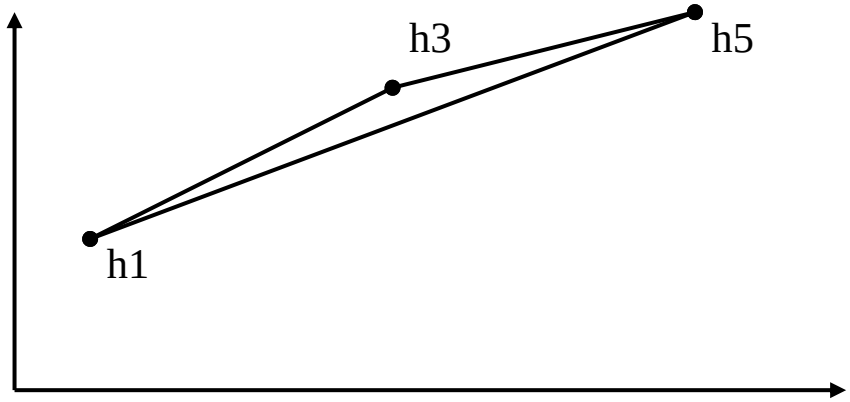

Рисунок 1.20 – Зміщення середньої точки

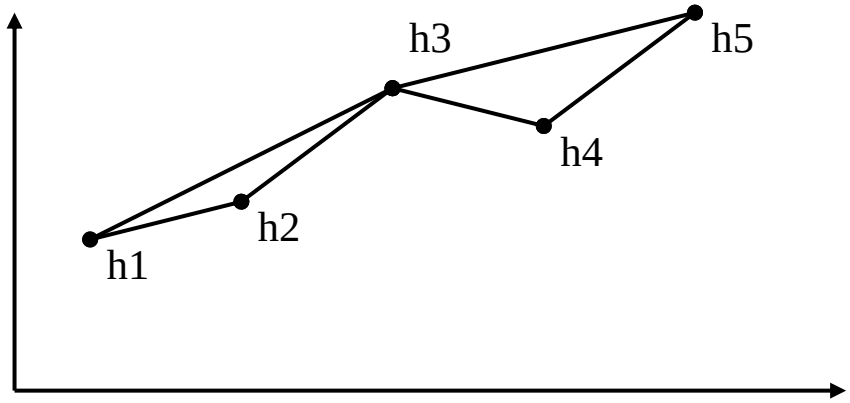

Рисунок 1.21 – Наступна ітерація роботи алгоритму midpoint displacement

Важливе зауваження: випадкові зміщення повинні бути пропорційні довжині відрізків, на яких вироблено розбиття.

Наприклад, ми розбиваємо відрізок h1 – h5 довжиною 16 тоді точка h3 посередині нього повинна мати висоту розраховану за формулою 1.3.

$$h3 = \frac{h1+h5}{2} + random(-16R, 16R) \quad (1.3)$$

де h1 – значення лівого кінця відрізка;

h5 – значення правого кінця відрізка;

random – функція вибору випадкової величини з вказаного проміжку;

R – константа яка визначає «шорсткість» (roughness) ламаної і є головним параметром в даному алгоритмі).

Спробуємо узагальнити цей алгоритм для двовимірного зображення. Почнемо з присвоєння випадкових значень чотирьом кутах всієї матриці зображення цілком і розіб’ємо її на чотири рівних квадрата. Таким же чином, як і в одновимірному midpoint displacement знайдемо решту значень матриці – точка в центрі виходить усередненням висот всіх 4 кутових точок, а кожна серединна точка на стороні великого квадрата – усередненням двох значень точок, що лежать на стороні квадрата. Розраховані значення зміщуються випадковим чином в межах, пропорційних стороні квадрата. Наступні ітерації виконуються аналогічно для менших квадратів.

Це ще не diamond-square – даний алгоритм, як правило, теж називають алгоритмом midpoint displacement і незважаючи на те, що він дає вже відносно прийнятні результати, в готовому зображенні без особливих зусиль можна помітити її «прямолінійну» натуру.

Алгоритм diamond-square – той самий, який дозволяє отримувати «справжні» фрактальні зображення – відрізняється від двовимірного midpoint displacement тим, що складається з двох кроків. Перший – так званий «Diamond» точно так само визначає центральну точку в квадраті шляхом усереднення кутових і додаванням власне випадкової величини. Другий же крок – «Square» – покликаний визначити висоту точок, що лежать на серединах сторін. Тут усереднюються не дві точки, а всі чотири сусідні. Важливо зауважити, що ці значення, які дісталися нам на попередньому кроці, повинні бути вже пораховані - тому обрахування потрібно вести «шарами», спочатку для всіх квадратів виконати крок «Diamond» – потім для всіх ромбів виконати крок «Square» – і перейти до менших квадратів.

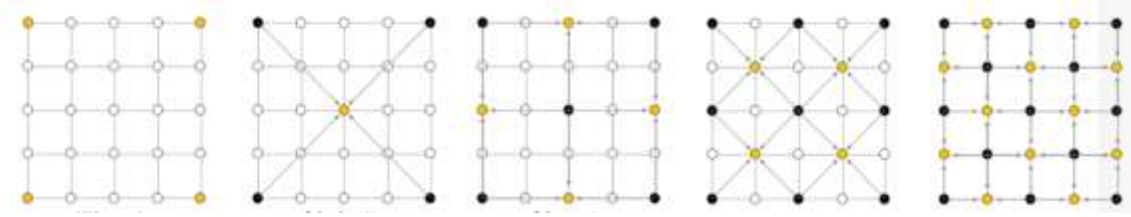

Рисунок 1.22 – Послідовність заповнення матриці розмірністю 5×5

Ситуація на краях ландшафту: справа в тому, що на етапі «Square» алгоритм використовує висоту точок, яких знаходяться за межами поточного квадрата і, можливо, всієї матриці. Можна або вважати ці висоти рівними 0 або будь-який інший константі), або намагаючись дізнатися висоту точки, що лежить на 64 пікселя лівіше лівої межі карти, ми дізнаємося висоту точки, що лежить на 64 пікселя лівіше правої межі. Реалізується взяттям координат по модулю, рівному розміру карти [6].

Wavelet Noise (хвильовий шум). З моменту винайдення шуму Перліна шумові функції були важливі для створення текстур у 3D-комп'ютерній графіки. Шум Перлін будує зображення з октав шуму, кожна з яких обмежений діапазоном частот. Октави використовуються як будівельні блоки для побудови складних шаблонів шуму, де частота спектр контролюється зважуванням внесків різних діапазонів. Наприклад, фрактальний шум може створюватися шляхом створення амплітуди кожної октави обернено пропорційно її частоті.

Оскільки цей підхід використовує одну вагу на октаву, він не підтримує детальне формування спектру. Для вирішення цієї проблеми Lewis представив aWiener-метод інтерполяції, який дозволяє будувати текстури з довільним спектром. Це забезпечує порівняно більший контроль за характеристиками шуму, але це значно підвищує обчислювальну складність.

Незважаючи на значний обсяг досліджень щодо визначення та побудови шуму, оригінальна версія шуму Перліна продовжувала залишатися найпоширенішим засобом галузі. Для цього були вагомі причини: швидкість та простота, а октави забезпечують достатній спектральний контроль для більшості додатків.

Кожна октава шуму Перліна призначений для обмеження діапазону, щоб він містив лише частоти в діапазоні степені 2. Але кожна група насправді містить значно ширший діапазон частот; це видно з перетворень Фур'є на рисунках 1.23 – 1.26.

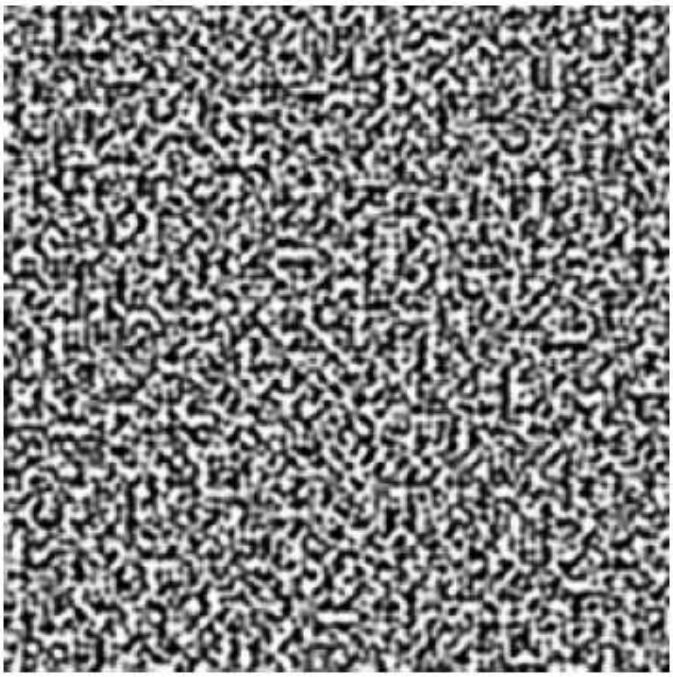

Рисунок 1.23: Шаблон двовимірного шуму Перліна.

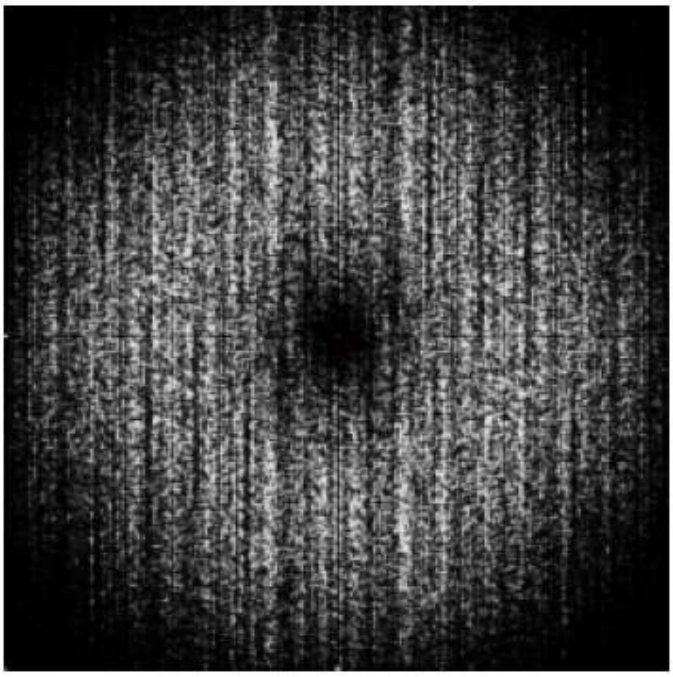

Рисунок 1.24 – Перетворення Фур'є шаблону двовимірного шуму Перліна.

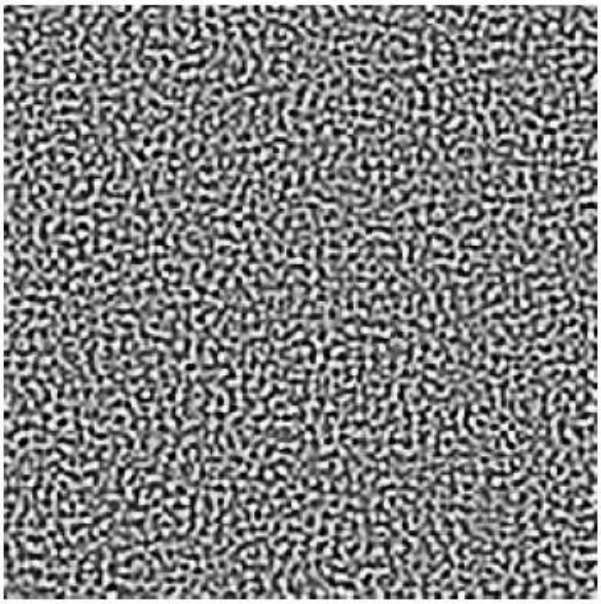

Рисунок 1.25: Шаблон двовимірного хвильового шуму

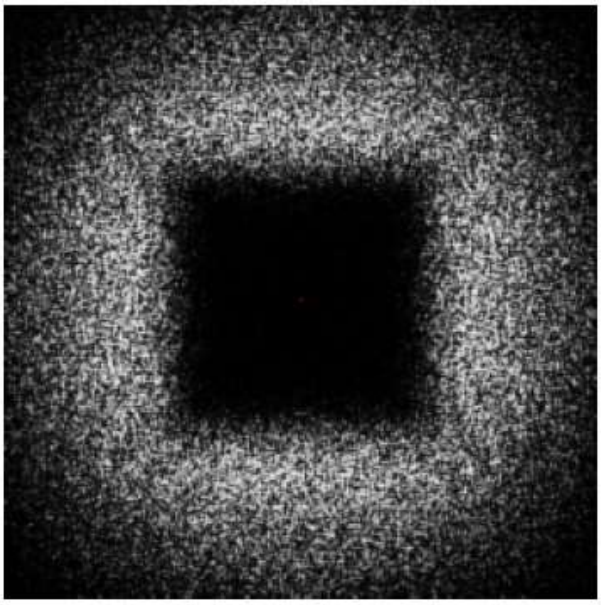

Рисунок 1.26 – Перетворення Фур'є шаблону двовимірного хвильового шуму

Ці обмеження слабких смуг призводять до серйозних проблем.Діапазон Перліна поблизу межі Найквіста містить обидві частоти, які є досить низькими, щоб бути репрезентабельними (тобто вони містять деталі, які має бути на зображенні) і частот, які є достатньо високими, щоб бути нерепрезентативні (тобто вони можуть спричинити псевдоніми). Без групи

спричиняє втрату деталізації артефактів, але в тому числі викликає псевдоніми.

Врівноваження цього компромісу між втратою деталей та псевдонімами було постійним джерелом розчарувань для авторів шейдерів. Тому що псевдоніми зазвичай є більш неприйнятні ніж втрата деталей, тому при створенні анімації, смуги агресивно послаблюються.

Невдалим наслідком цього є те, що при збільшенні масштабу у сцені деталь текстури стає помітною пізніше геометрії деталі, тому текстура, здається, не пов’язана з її геометрією. Натомість текстура, здається, зникає неприродно, як ніби там був серпанок, який затуляв лише деякі частини зовнішнього вигляду поверхні. Метод розрідженої згортки був введений для вдосконалення обмежень діапазону шуму, але він теж не повністю вирішити проблему втрати деталізації проти псевдонімів.

Однак є ще більш фундаментальна проблема, яка мучить кожен існуючий метод створення шуму. Під час візуалізації, це є загальним для текстури двовимірних поверхонь шляхом вибірки функції тривимірного шуму , але отримана двовимірна-текстура загалом не буде обмежена діапазоном,навіть якщо тривимірна шумова функція абсолютно обмежена діапазоном. Це означає що втрата деталізації та компромісний компроміс не може бути вирішена просто побудова смугової тривимірної функції. Наскільки нам відомо, це проблема навіть ніколи не була визначена, тим більше не розглядалась.

Хвильовий шум, який вирішує всі ці проблеми базується на спостереженні, що побудова Перліна з використанням сум масштабованих та ослаблених версій функції з обмеженим діапазоном були раннім прикладом того, що стало відомим як багатозначна функція. Після роботи Перліна, хвильовий аналіз, також відомий як багаторезолюційний аналіз, має виникла як потужний спосіб аналізу та побудови таких функцій.

Тому природно використовувати цей підхід для аналізу та побудови шум.

Хвильовий шум майже ідеально обмежений діапазоном, забезпечуючи хороші деталі з мінімальними псевдонімами, як показано на рисунку 1.27, можна використовувати для текстурування двовимірної поверхні.

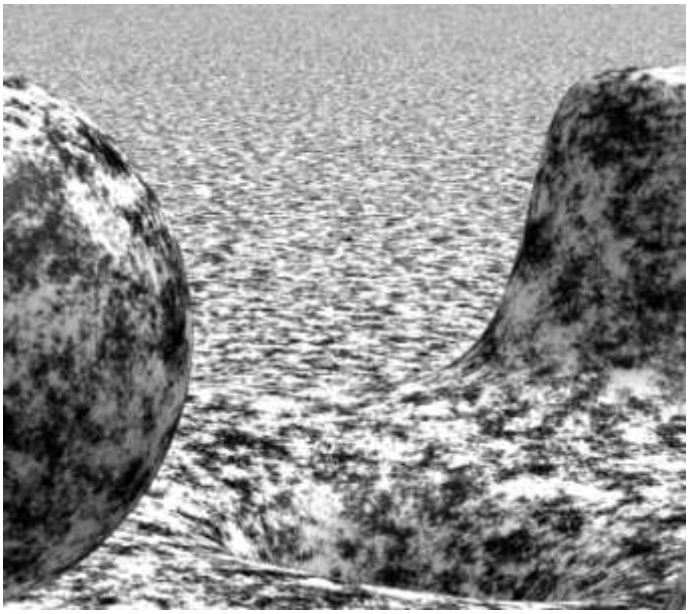

Рисунок 1.27 – Накладання текстури хвильового шуму на двовимірну поверхню

Більше того, тривимірний хвильовий шум можна використовувати для текстурування двовимірної поверхні таким чином, щоб зберегти обмежений характер [7].

Отже, такі основні алгоритми створення зображень аморфних об’єктів за допомогою процедурних шумів. На мою думку, найбільш реалістичні результат дає останній з них, diamond-square, але і він не позбавлений деяких недоліків. Наприклад, створивши зображення, яке добре виглядає при сильному наближенні – при перегляді цілком можна побачити суцільний шум. Виправити це можна комбінуванням констант для випадкових зміщень. Їх значення можна задати функціонально ввести вручну або зберегти добре налаштовану множину коефіцієнтів.

1.3 Постановка задачі

В ході виконання дипломного проектування необхідно вивчити предметну область процесу створення аморфних об’єктів для фотореалістичних зображень. Крім того потрібно ознайомитися з теперішнім станом справ сфері створення фотореалістичних зображень, розглянути аналоги та зробити висновок про необхідність розробки інформаційної технології, сформулювати вимоги до програмного засобу, обрати засоби розробки: мову програмування та середовище розробки.

В результаті аналізу програмних рішень та алгоритмів створення аморфних об’єктів, обрано алгоритм Diamond-Square для подальшого вдосконалення та сформульовано вимоги до розробки технології та реалізації програмного засобу. Мета розробки інформаційної технології – підвищення швидкодії створення аморфних об’єктів для фотореалістичних зображень за допомогою зменшення кількості розрахунків шляхом спрощення обрахунку значення середніх точок та зменшення кількості випадкових зміщень середніх точок. Враховуючи можливість невідповідності розмірності матриці з розмірністю готового зображення потрібно реалізувати варіант алгоритму з можливістю «зшивання» однакових матриць. В такому випадку матриця буде подібна розгортці тороїдної фігури. Досягається такий ефект врахуванням значень елементів матриці з протилежного боку при розрахунку значень елементів, що знаходяться на межі матриці. Готове зображення розглядається як двовимірний масив пікселів, колір яких визначається значенням відповідного елемента матриці зображення. Таким чином колір одного пікселя буде розрахований програмними функціями, окремими для кожного типу зображених аморфних об’єктів. Необхідно реалізувати наступні можливості програмного засобу: розрахунок матриці зображення вдосконаленим алгоритмом Diamond-Square заданої користувачем розмірності з урахуванням обраного користувачем початкового значення

матриці та значення максимального зміщення середньої точки, виводити зображення на екран, зберігати створенні зображення. Крім того програмний засіб має задовольняти двом основним вимогам: фотореалістичність готового зображення аморфних об’єктів та швидкодія вища ніж швидкодія аналогічного програмного засобу,який заснований на звичайному алгоритмі Diamond-Square.

Згідно з цими вимогами здійснюється робота над розробкою інформаційної технології та реалізацією програмного засобу.

1.4 Висновок

В першому розділі проаналізовано предметну область створення фотореалістичних зображень, розглянуто існуючі програмні рішення для створення аморфних об’єктів, проведено аналіз основних алгоритмів за якими вони функціонують, визначено досить велику обчислювальну складність цих алгоритмів, обґрунтовано необхідність підвищення швидкодії програмних засобів створення зображень.

Проаналізувавши існуючі аналоги створення аморфних об’єктів для фото реалістичних зображень для подальшого вдосконалення було обрано алгоритм Diamond-Square, ідея якого дозволяє скоротити кількість обчислень без значних втрат якості, що дозволить підвищити швидкість роботи програмного засобу заснованого на цьому алгоритмі.

Виконано постановку задачі, виконання якої дозволить підвищити швидкодію створення аморфних об’єктів для фотореалістичних зображень.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ АМОРФНИХ ОБ'ЄКТІВ ДЛЯ ФОТОРЕАЛІСТИЧНИХ ЗОБРАЖЕНЬ

2.1 Опис вибраної теоретичної основи інформаційної технології

Як теоритичну основу інформаційної технології створення аморіфних об'єктів для фотореалістичних зображень було обрано алгоритм «Diamond-Square». Щоб досягти загальну мету магістерської кваліфікаційної роботи, а саме збільшення швидкодії створення фотореалістичних зображень необхідно зменшити кількість обчислень, які потрібні дя створення мариці зображення.

Проаналізуємо покрокову роботу алгоритму при створенні матриці розмірністю 9×9 елементів.

Першим кроком буде присвоєння випадкових значень початковим елементам матриці зображення Це показано в таблиці 2.1.

Таблиця 2.1 – Присвоєння випадкових значень початковим елементам матриці зображення

110								75
90								125

Наступним буде виконання кроку Diamond, а саме розрахунок значення центрального елемента матриці за формулою 2.1. Результат наведено в таблиці 2.2.

$$M_{i;j} = \frac{M_{i+k;j+k}+M_{i+k;j-k}+M_{i-k;j+k}+M_{i-k;j-k}}{4} + random(k * r) \quad (2.1)$$

де $M_{i-k;j+k}$; $M_{i+k;j-k}$; $M_{i+k;+kj}$; $M_{i-k;j-k}$; – елементи сусідні с розрахованим елементом;

k – відстань між елементами на даному кроці;

r – константа для розрахунку випадкових зміщень.

Таблиця 2.2 – Виконання кроку diamond

110								75
				144				
90								125

Наступний крок – Square, а саме розрахунок значення елементів матриці, що лежать на серединах сторін квадрата, за формулою 2.2. Результат наведено в таблиці 2.3.

$$M_{i;j} = \frac{M_{i;j+k}+M_{i;j-k}+M_{i-k;j}+M_{i+k;j}}{4} + random(k * r) \quad (2.2)$$

де $M_{i-k;j}$; $M_{i+k;j}$; $M_{i;j+k}$; $M_{i;j-k}$; – елементи сусідні с розрахованим елементом;

k – відстань між елементами на даному кроці;

r – константа для розрахунку випадкових зміщень.

Таблиця 2.3 – Виконання кроку square

110				123				75
132				144				118
90				140				125

Далі кроки Diamond та Square виконуються поперемінно за формулами 2.1 та 2.2, зі зменшенням параметру k в два рази на кожній ітерації, результати подальших розрахунків показано в таблицях 2.4 – 2.7

Таблиця 2.4 – Виконання кроку diamond

110				123				75
		129				109		
132				144				118
		121				125		
90				140				125

Таблиця 2.5 – Виконання кроку square

110		115		123		99		75
125		129		126		109		103
132		134		144		130		118
111		121		138		125		133
90		113		140		115		125

Таблиця 2.6 – Виконання кроку diamond

110		115		123		99		75
	117		128		115		93	
125		129		126		109		103
	122		143		129		96	
132		134		144		130		118
	137		127		120		129	
111		121		138		125		133
	95		134		131		130	
90		113		140		115		125

Таблиця 2.7 – Виконання кроку square

110	111	115	132	123	118	99	91	75
108	117	120	128	120	115	111	93	98
125	121	129	135	126	128	109	101	103
119	122	137	143	134	129	118	96	107
132	131	134	139	144	133	130	117	118
122	137	124	127	131	120	122	129	126
111	106	121	123	138	132	125	128	133
104	95	117	134	137	131	124	130	128
90	100	113	132	140	122	115	106	125

2.2 Аналіз та вдосконалення математичної моделі

Для підвищення швидкодії алгоритму та збереження властивостей ззображення необхідно скоротити кількість обчислень.

По перше для збереження «зшитості» правого та лівого, а також верхнього та нижнього краю матриці та скорочення обчислень значення елементів правого краю прирівняємо до елементів лівого краю, а значення елементів нижнього краю прирівняємо до значень елементів верхнього краю. Очевидно, що в такому випадку всі значення «кутових» елементів будуть однаковими. Для частини ітерацій крім кількох перших та кількох останніх крок Diamond буде виконуватися за формулою 2.3.

$$M_{i;j} = \frac{M_{i+k;j+k}+M_{i+k;j-k}+M_{i-k;j+k}+M_{i-k;j-k}}{4} \quad (2.3)$$

де $M_{i-k;j+k}$; $M_{i+k;j-k}$; $M_{i+k;+kj}$; $M_{i-k;j-k}$; – елементи сусідні с розрахованим елементом;

k – відстань між елементами на даному кроці;

Крок Square поділиться на два підкроки для розрахунку значень елемента середини «горизонтальної» сторони та значення елемента середини «вертикальної» сторони, але це спрощення не поширюється на елементи, які лежать на стороні «великого»квадрата. «Горизонтальна» сторона буде розрахована за формулою 2.4, а вертикальна за формулою 2.5.

$$M_{i;j} = \frac{M_{i;j+k}+M_{i;j-k}}{2} \quad (2.4)$$

де $M_{i;j+k}$; $M_{i;j-k}$ – елементи сусідні с розрахованим елементом;

k – відстань між елементами на даному кроці;

$$M_{i;j} = \frac{M_{i+k;j}+M_{i-k;j}}{2} \quad (2.5)$$

де $M_{i\text{-}k;j}$; $M_{i+k;j}$– елементи сусідні с розрахованим елементом;

k – відстань між елементами на даному кроці;

Приклад покрокового створення матриці зображення розмірністю 9*9 за допомогою вдосконаленого алгоритму Diamond-Square Показано в таблицях 2.8 – 2.14.

Таблиця 2.8 – Присвоєння початковихелементів матриці

110								110
110								110

Таблиця 2.9 – Виконання кроку diamond

110								110
				139				
110								110

Таблиця 2.10 – Виконання кроку square

110				71				110
228				139				228
110				71				110

Таблиця 2.11 – Виконання кроку diamond

110				71				110
		142				85		
228				139				228
		121				153		
110				71				110

Таблиця 2.12 – Виконання кроку square

110		47		71		77		110
167		142		116		85		167
228		99		139		205		228
212		121		165		153		212
110		47		71		77		110

Таблиця 2.13 – Виконання кроку diamond

110		47		71		77		110
	104		123		84		80	
167		142		116		85		167
	134		92		120		170	
228		99		139		205		228
	188		135		174		212	
212		121		165		153		212
	128		108		102		165	
110		47		71		77		110

Таблиця 2.14 – Виконання кроку square

110	81	47	70	71	89	77	90	110
111	104	126	123	109	84	83	80	111
167	124	142	111	116	108	85	101	167
198	134	133	92	111	120	138	170	198
228	143	99	111	139	143	205	220	228
222	188	123	135	147	174	203	212	222
212	175	121	151	165	128	153	171	212
138	128	117	108	88	102	131	165	138
110	81	47	70	71	89	77	90	110

Можна побачити, що створена за вдосконаленим алгоритмом матриця не втрачає якості в тій мірі, яка зробить зображення аморфного об’єкта занадто неприродним, або таким що не задоволить критеріям фотореалістичності. Ефективність даного вдосконаленого алгоритму «Diamond-Square» порівняно зі звичайним алгоритмом «Diamond-Square» буде перевірено, після програмної реалізації інформаційної технології створення аморфних об’єктів для фотореалістичних зображень. Очікується підвищення швидкодії створення матриці зображення в межах від 7% до 10% залежно від розмірності матриці.

2.3 Вибір колірної моделі для вирішення задачі

Колірна модель – це сукупність абсолютних або відносних параметрів кольору, що описують даний колір в даному колірному просторі.

Колір має різну фізичну природу. На моніторі ми бачимо колір, який випромінюється екраном, на папері - колір, відбитий аркушем паперу. Кольорові моделі призначені для опису кольорів, утворених різними методами.

Колірні моделі в графічних програмах підтримуються спеціальними графічними режимами. Всі використовувані в даний час колірні моделі можна умовно класифікувати наступним чином:

1. Монохромні: двоградаційні (дуплексні), напівтонові (з відтінками сірих кольорів);

2. Кольорові: індексні, повнокольорові: адитивні (RGB), засновані на додаванні кольорів; субтрактивні (CMYK), засновані на відніманні кольорів; перцепційне (HSV, HSB, HLS, LAB, і т.д.), засновані на сприйнятті [11].

RGB [11]. Ця модель отримала свою назву за першими літерами англійських слів Red (Червоний), Green (Зелений), Blue (Синій).

Модель RGB називають адитивною, тому що будь-який колір в цій моделі утворюється шляхом змішування в різних пропорціях трьох основних кольорів: червоного, зеленого і синього, які називаються первинними. При попарному змішуванні первинних кольорів утворюються вторинні кольори: блакитний, пурпурний і жовтий. Первинні і вторинні кольори називаються базовими кольорами, які показані на рисунку 2.1.

Базовими кольорами називаються кольори, за допомогою яких можна отримати практично весь спектр видимих кольорів.

Колір в даній колірній моделі описується трьома значеннями в діапазоні від 0 до 255.

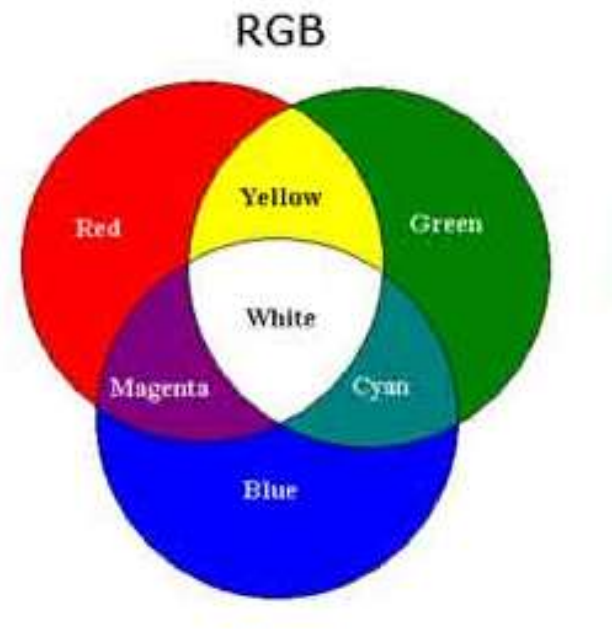

Рисунок 2.1 – Базові кольори моделі RGB

Подання колірної моделі RGB наступному порядку: червона, зелена і синя складові. У цих точках відповідні складові мають максимальні значення: чистий червоний колір – 255, 0, 0 (рівень червоного максимальний, а зелена і синя складові відсутні); зелений колір – 0, 255, 0;синій – 0, 0, 255. чорний колір – 0, 0, 0 (жоден колір не випромінюється, всі складові рівні 0); білий колір – 255, 255, 255; жовтий колір – 255, 255, 0; шкала сірого – містить 256 відтінків сірого. На цій осі значення червоного, зеленого і синього складової однакові наприклад – 50,50,50.

Таким чином, будь-який колір в цій моделі може бути представлений в колірному просторі за допомогою вектора.

Обмеження RGB-моделі

1. Апаратна залежність:колір, що виникає в результаті змішування колірних складових RGB елемента, залежить від типу світлопередавальних елементів. У технології виробництва сучасних моніторів знаходять застосування різні типи світлопередавальних елементів, які мають різну спектральну характеристику і установка одних і тих же інтенсивностей

кольорів в разі використання різних світлопередавальних елементів приведе до синтезу різного кольору.

Для усунення або принаймні мінімізації залежно RGB-моделі від апаратних засобів використовуються різні пристрої і програми градуювання.

2. Обмеження гами кольорів.

Колірний обхват – це діапазон кольорів, який може розрізняти людина або відтворювати пристрій незалежно від механізму отримання кольору випромінювання або відображення.

Обмеженість гами пояснюється тим, що за допомогою адитивного синтезу принципово неможливо отримати всі кольори видимого спектру це доведено теоретично. Зокрема, деякі кольори, такі як чистий блакитний або чистий жовтий, не можуть бути точно відтворені на екрані. Але, не дивлячись на те, що людське око здатне розрізняти кольорів більше, ніж монітор, RGB-моделі цілком достатньо для створення кольорів і відтінків, необхідних для відтворення фотореалістичних зображень на екрані вашого комп'ютера.

Колірна модель Lab [11]. Колір в даній колірній моделі визначається трьома параметрами наведено на рисунку 2.2, два з яких хроматичні компоненти:

a – кольоровість в діапазоні від зеленого до червоного;

b – кольоровість в діапазоні від синього до жовтого;

L – світлота (Lightnesss), що представляє собою аналог яскравості.

Параметри a і b задаються числами, що знаходяться в діапазоні від -120 до 120. Для параметра a значення -120 відповідає зеленому кольору, а +120 - червоному. Для параметра b значення -120 - це синій колір, а значення +120 - жовтий. Все за умови, що L дорівнює 100%. Светлота змінюється в діапазоні від 0 до 100%.

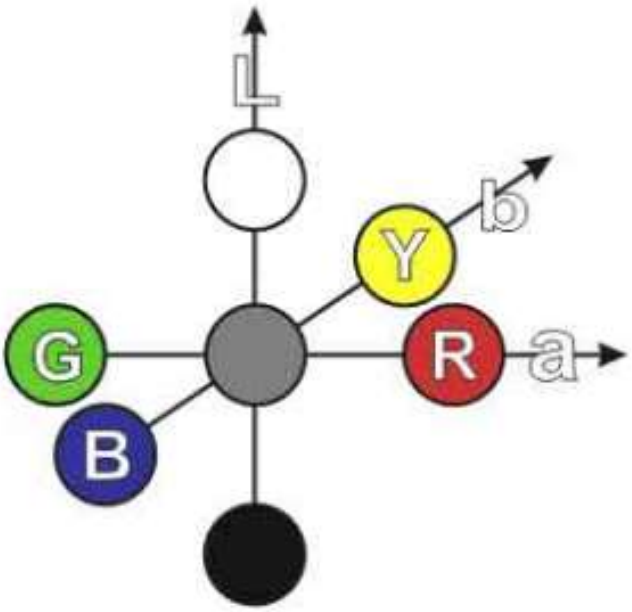

Рисунок 2.2 – Кольорова палітра Lab

Використовувані в Lab - моделі колірні координати узгоджуються з біологічним механізмом сприйняття кольору, відкритим в 1981 році американськими вченими Давидом Хьюблом (David H. Hubel) і Торстеном Вайзелом (Torsten N. Wiesel), які отримали Нобелівську премію за дослідження зору. Вони довели, що очей надає в мозок зовсім не інформацію про червоний, зелений і синій. Замість цього мозок отримує:

• різницю між світлим і темним;

• різницю між зеленим і червоним;

• різницю між синім і жовтим, де жовтий - сума червоного і зеленого.

Схема колірного зору

На горизонтальному зрізі всі кольори мають однакову яскравість. Це означає, що кожен колір може бути точно описаний в колірних координатах a й b.

Переваги моделі Lab:

1. Апаратна незалежність;

2. Більший колірний обхват в порівнянні з моделями RGB і CMYK.

3. На базі параметрів цієї колірної системи можна визначити параметри інших колірних моделей.

Колірна модель HSB [11]. Колірна модель HSB розроблена з максимальним урахуванням особливостей сприйняття кольору людиною. Колір описується трьома компонентами: відтінком (Hue), насиченістю (Saturation) і яскравістю (Brigfitness).

Значення кольору вибирається як вектор, що виходить із центру кола. Точка в центрі відповідає білому кольору, а точки по периметру кола – чистим спектральним кольорам, що показано на рисунку 2.3. Напрямок вектора задається в градусах і визначає колірний відтінок. Довжина вектора визначає насиченість кольору. На окремій осі, яка названа ахроматичною, задається яскравість, при цьому нульова точка відповідає чорному кольору.

Колірний обхват моделі HSB перекриває всі відомі значення реальних кольорів. Модель HSB прийнято використовувати при створенні зображень на комп'ютері з імітацією прийомів роботи і інструментарію художників.

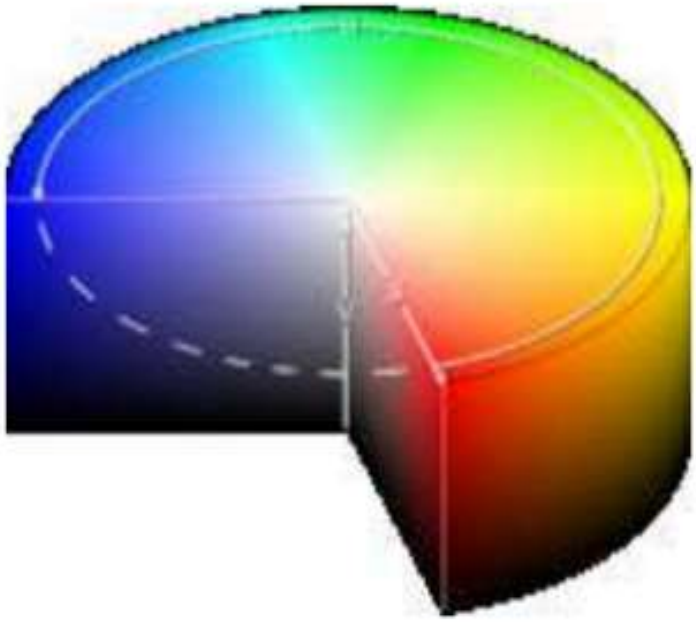

Рисунок 2.3 – Спектр кольорів моделіHSB

Під колірним тоном розуміється світло з домінуючою довжиною хвилі. Параметр Hue (Тон) характеризується становищем на колірному колі і визначається вершиною кута (від 0 до 360).

На колірному колі первинні кольори розташовані на рівній відстані один від одного. Вторинні кольори знаходяться між первинними. У свою чергу, кожен колір розташований навпроти доповнює його (компліментарного) кольору і знаходиться між двома кольорами, з яких отримано.

Параметр Saturation визначає насиченість кольору. Збільшення насиченості призводить до збільшення концентрації кольору, а зменшення - до його розбавлення.

Насиченість має максимальне значення на окружності - 100% і мінімальне в центрі кола - 0%.

Параметр Brightness (яскравість) визначає ступінь освітленості або затемненості кольору - це інтенсивність, з якою енергія світла впливає на рецептори нашого ока. Величина яскравості вимірюється в відсотках в діапазоні від 0% (чорний колір) до 100% (білий колір).

Ахроматичні кольори, тобто білі, сірі та чорні, характеризуються тільки яскравістю.

Цю модель можна представити у вигляді циліндра, в якому:

• контур підстави (окружність) відповідає осі зміни параметра Hue,

• радіус підстави – осі зміни параметра Saturation,

• бічна сторона – осі зміни параметра Brightness.

Ця модель більше, ніж інші відповідає традиційному сприйняттю кольору людиною і найбільш проста в розумінні: спочатку можна визначити колірний тон, а потім задати йому насиченості і яскравість. Крім того, модель HSB зручно використовувати при редагуванні малюнків. Наприклад, ви хочете замінити зелений лист на жовтий редагованої фотографії. Досить поміняти тільки колірну складову використовуваних квітів, не змінюючи

яскравість і насиченість. Малюнок при цьому не зміниться, але прийме інший відтінок.

Яскравість і колірний тон не є повністю незалежними параметрами. Зміни яскравості зображення впливає на зміну колірного тону, що створює небажаний колірне зрушення в зображенні.

Наприклад, при значному зменшенні яскравості зелені кольори синіють, сині наближаються до фіолетовим, жовті – до помаранчевих, а помаранчеві – до червоних. Сильне збільшення яскравості випромінювання викликає інший ефект. Червоні кольори переходять в помаранчеві, потім в жовті і, нарешті, – в білі.

Недолік колірної моделі HSB: так само як і в попередніх колірних моделях – обмежене колірне простір.

переваги:

1. Апаратна незалежність.

2. Більш простий і інтуїтивно зрозумілий механізм управління кольору.

Порівнявши вище наведені моделі та узгодивши їх із загальною метою роботи було прийнято рішення використовувати модель RGB, яка може бути реалізована на будь-якій мові програмування, а недоліки даної моделі не є критичними для досягнення мети розробки інформаційної технології створення аморфних об’єктів для фотореалістичних зображень.

2.4 Складові інформаційної технології

Визначення складових інформаційної технології створення аморфних об’єктів для фотореалістичних зображень.

Для виконання вимог до інформаційної технології вона повинна містити такі етапи роботи: зчитування набору вхідних даних; розрахунок матриці зображення з використанням випадкових величин та перетворення

матриці зображенння в готове зображення обраної колірної моделі. Ці складові наведено на рисунку 2.4.

Рисунок 2.4 – Структура інформаційної технології створення аморфних об’єктів для фотореалістичних зображень

Зчитування набору вхідних даних полягає у введені користувачем розмірності матриці зображення, початкового значення кутових елементів матриці зображення та коефіцієнта випадкових зміщень нових розрахованих елементів матриці зображення.

Розрахунок матриці зображення базується на вдосконаленому алгоритмі «Diamond-Square» з використанням даних введених користувачем.

Перетворення матриці зображення в готове зображення обраної колірної моделі полягає в нормалізації значень матриці в межах [0:255], яке здійснюється за формулою 2.6 та присвоєння пікселю зображення кольору, який визначається функцією, аргументом якої є значення відповідного елемента матриці.

$$Mn_{ij} = \frac{255 \times (M_{ij} - M_{min})}{(M_{max} - M_{min})} \quad (2.6)$$

Де Mn_{ij} – нормалізоване значення елемента матриці зображення;

M_{ij} – початкове значення елемента матриці зображення;

M_{max} та M_{min} – максимальне та мінімальне значення елементів матриці.

2.5 Висновок

У другому розділі магістерської кваліфікаційної роботи було розроблено інформаційну технологію створення аморфних об’єктів для фотореалістичних зображень на основі алгоритму «Diamond-Square». Проаналізовано та вдосконалено математичну модель алгоритму створення матриці зображення результатом чого стало зменшення кількості обчислень. Було обрано колірну модель RGB, в якій планується реалізація інформаційної технології. На основі цього було визначено та описано основні складові інформаційної технології створення аморфних об’єктів для фотореалістичних зображень.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ ФОТОРЕАЛІСТИЧНИХ ЗОБРАЖЕНЬ

3.1 Розробка структури програмного забезпечення

Відповідно до поставлених вимого до розробки необхідно реалізувати наступні можливості:

1) Введення користувачем вхідних .
2) Розрахунок елементів матриці зображеня аморфних об’єктів.
3) Нормалізація матриці зображення
4) Перетворення нормалізованої матриці зображення в готове зображення в обраній колірній моделі
5) Виведення готового зображення на екран.
6) Збереження готових зображень.

Згідно з цими пунктами розроблено структурні елементи програмного модуля, а саме введеня параметрів через графічний інтерфейс, створення матриці зображень, розрахунок значення елементів матриці зображення, нормалізація матриці зображення, перетворення матриці зображення в готове зображення колірної моделі RGB за функціями налаштованими дослідним шляхом для кожного типу аморфних об’єктів. Для диму: SetPixel(ix,iy,RGB(color[ix,iy] ,color[ix,iy] , color[ix,iy])); для неба: SetPixel(ix,iy,RGB(255-(color[ix,iy] div 2),255-(color[ix,iy] div 2), 255)); для моря: SetPixel(ix,iy,RGB(0 ,255-(255 -color[ix,iy]) div 2 ,255-2*((255-color[ix,iy]) div 3))); для вогню: SetPixel(ix,iy,RGB(color[ix,iy] ,color[ix,iy] div 2 , color[ix,iy] div 16)). Збереження готового зображення відбувається в тій самій директорії де розміщено виконавчий файл. Структурна схема алгоритму програмного забезпечення зображена на рисунку 3.1.

Рисунок 3.1 – Структурна схема алгоритму програмного забезпечення

3.2 Побудова UML-діаграм розробленого програмного забезпечення

Універсальна мова моделювання (UML) − це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови системи, але ця мова допомагає наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками.

UML розроблено для проектування об’єктно-орієнтованих структур, ця мова має дуже обмежену користь для програмування на основі інших парадигм. Основні типи діаграм:

Діаграма випадків використання показує дієвих осіб (людей або інших користувачів системи), випадки використання (сценарії використання системи) та їх взаємодію.

Діаграми класів.

Діаграми послідовності, на яких показано об’єкти і послідовність методів, якими ці об’єкти викликають інші об’єкти.

Діаграми співпраці, на яких буде показано об’єкти та їх взаємозв’язок з наголосом на об’єкти, які беруть участь у обміні повідомленнями.

Діаграми стану, на яких буде показано стани, зміну станів і події у об’єкті або частині системи.

Діаграми діяльності, на яких буде показано дії та зміни однієї дії іншою, які є наслідком подій, що сталися у певній частині системи.

Діаграми компонентів.

Діаграми впровадження, на яких буде показано екземпляри компонентів та їх взаємодію.

Діаграми взаємозв’язку сутностей, на яких буде показано дані, взаємозв’язки і умови обмеження зв’язків між даними [12].

Під час проектування та розробки інформаційної технології створення аморфних об'єктів для фотореалістичних зображень було побудовано діаграму діяльності, яка показана на рисунку 3.2. На ній відображено основні дії програмного засобу та переходи від одного стану до іншого.

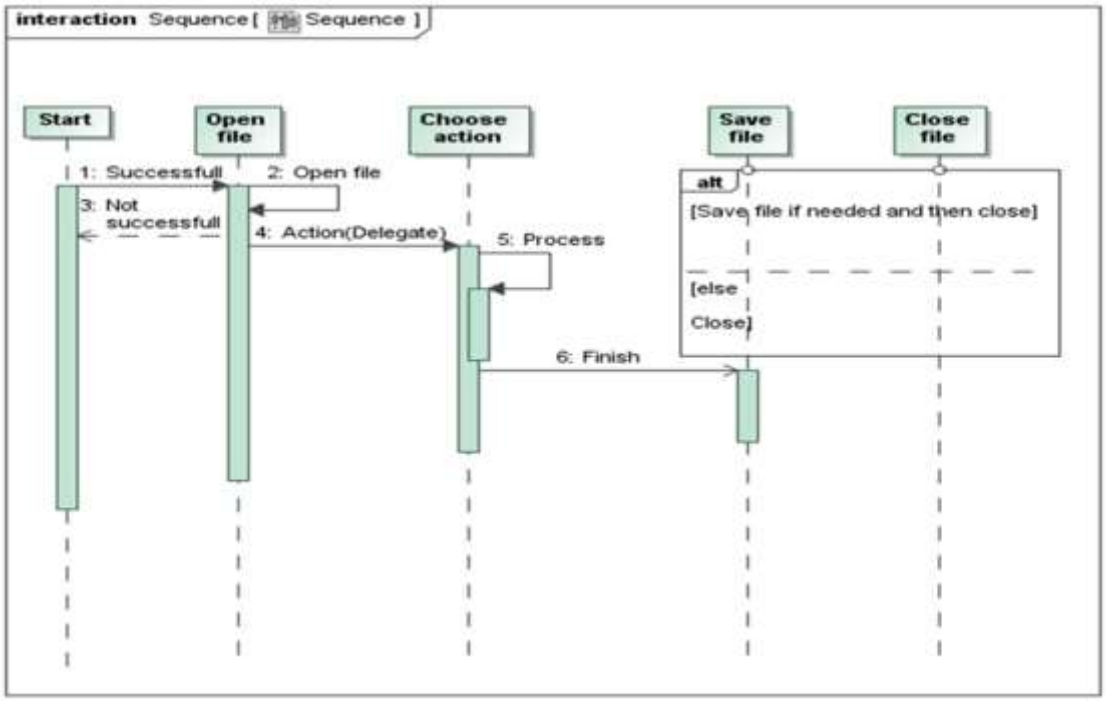

Рисунок3.2 – Діаграма діяльності

Також побудовано діаграму компонентів,яка зображена на рисунку 3.3 на якій позначено взаємодію компонентів програмного засобу.

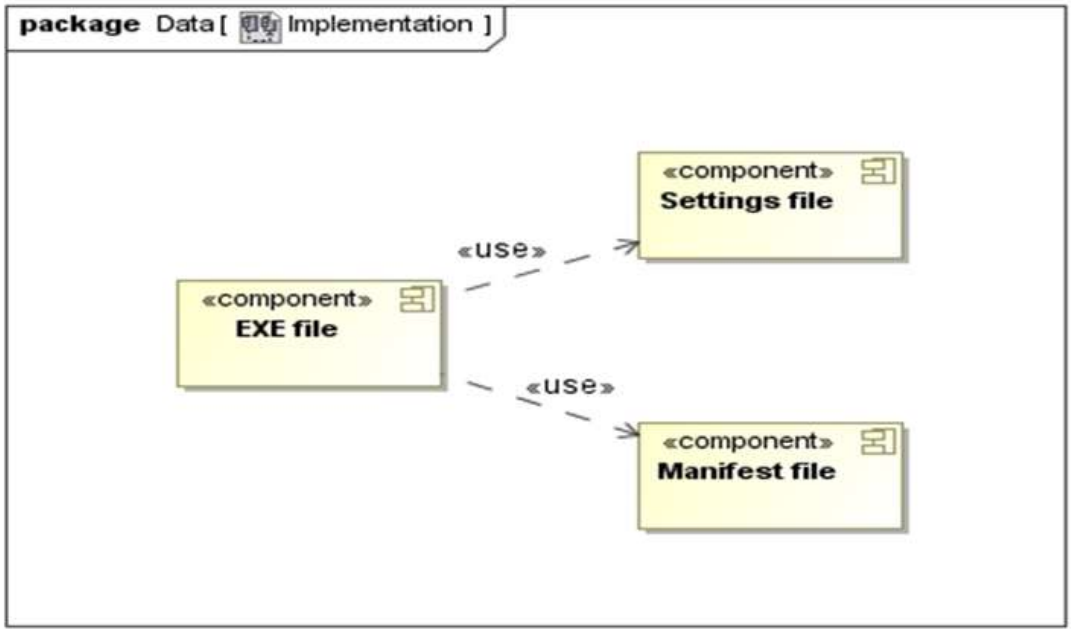

Рисунок 3.3 – Діаграма компонентів

3.3 Розробка алгоритму роботи програмного забезпечення

Алгоритм diamond-square починає роботу з двовимірного масиву розміру $2^n + 1$. У чотирьох кутових точках масиву встановлюються початкові значення. Кроки diamond і square виконуються по черзі до тих пір, поки всі значення масиву не будуть встановлені.

Крок square (квадрат). Для кожного квадрата в масиві, знаходиться серединна точка, в яку встановлюється середнє значення чотирьох кутових точок плюс випадкове значення.

Крок diamond (ромб). Для кожного ромба в масиві, встановлюється серединна точка, якої присвоюється середнє арифметичне з чотирьох кутових точок плюс випадкове значення.

На кожній ітерації випадкове значення, що додають до серединних точок, зменшується.

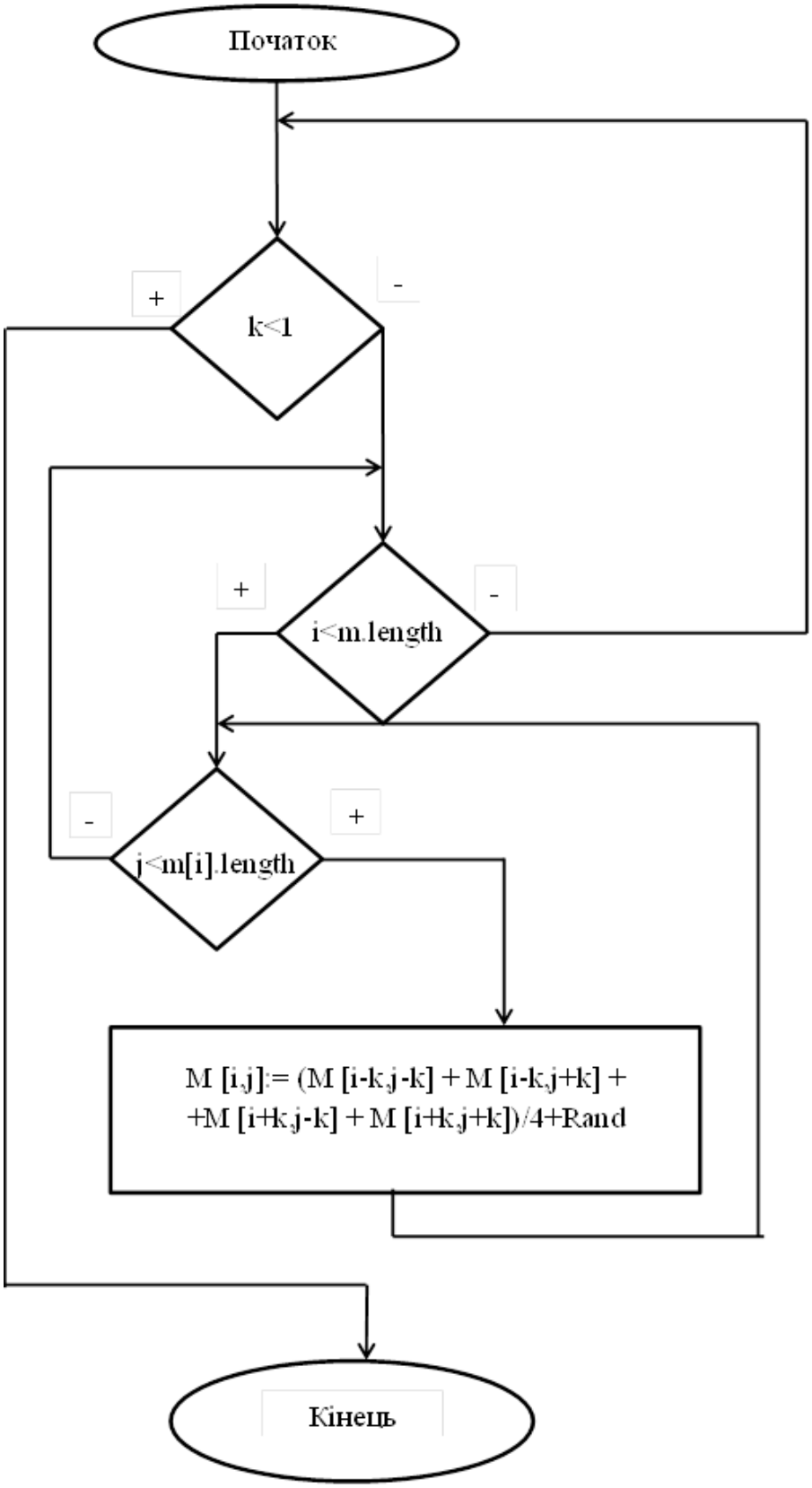

Рисунок 3.4 – Схема алгоритму кроку DIAMOND

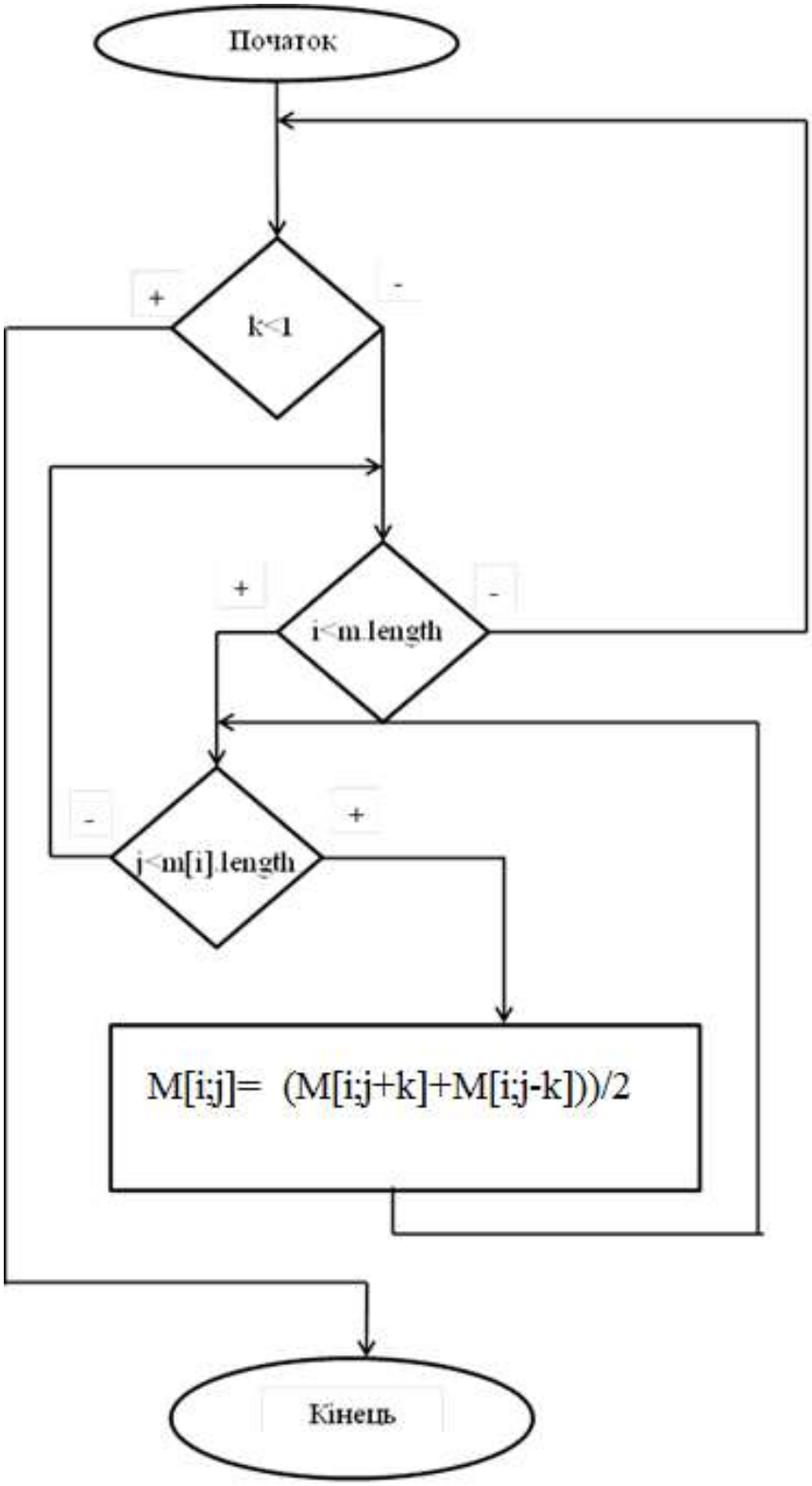

Рисунок 3.5 – Схема алгоритму кроку SQUARE для горизонтальних сторін

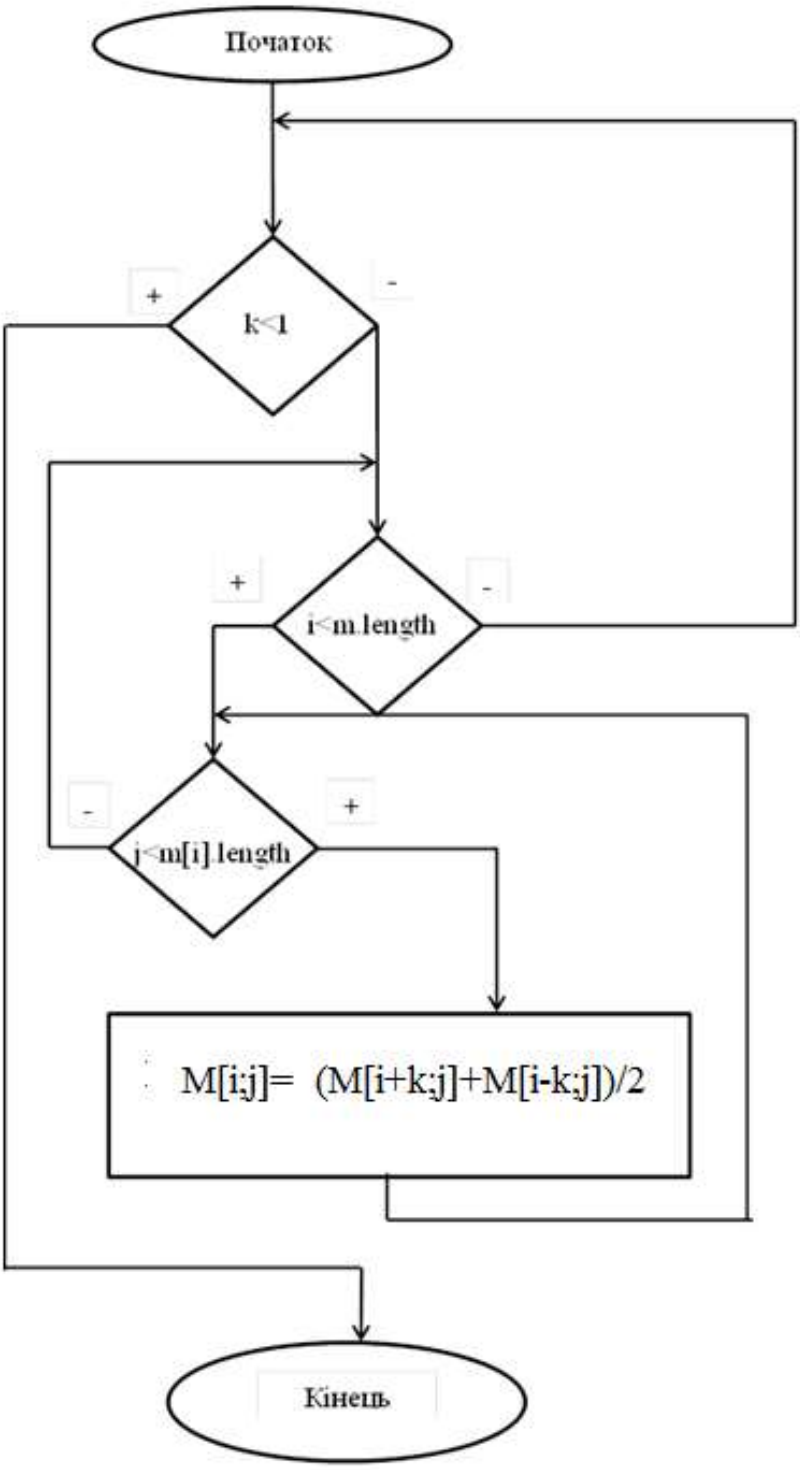

Рисунок 3.6 – Схема алгоритму кроку SQUARE для вертикальних сторін

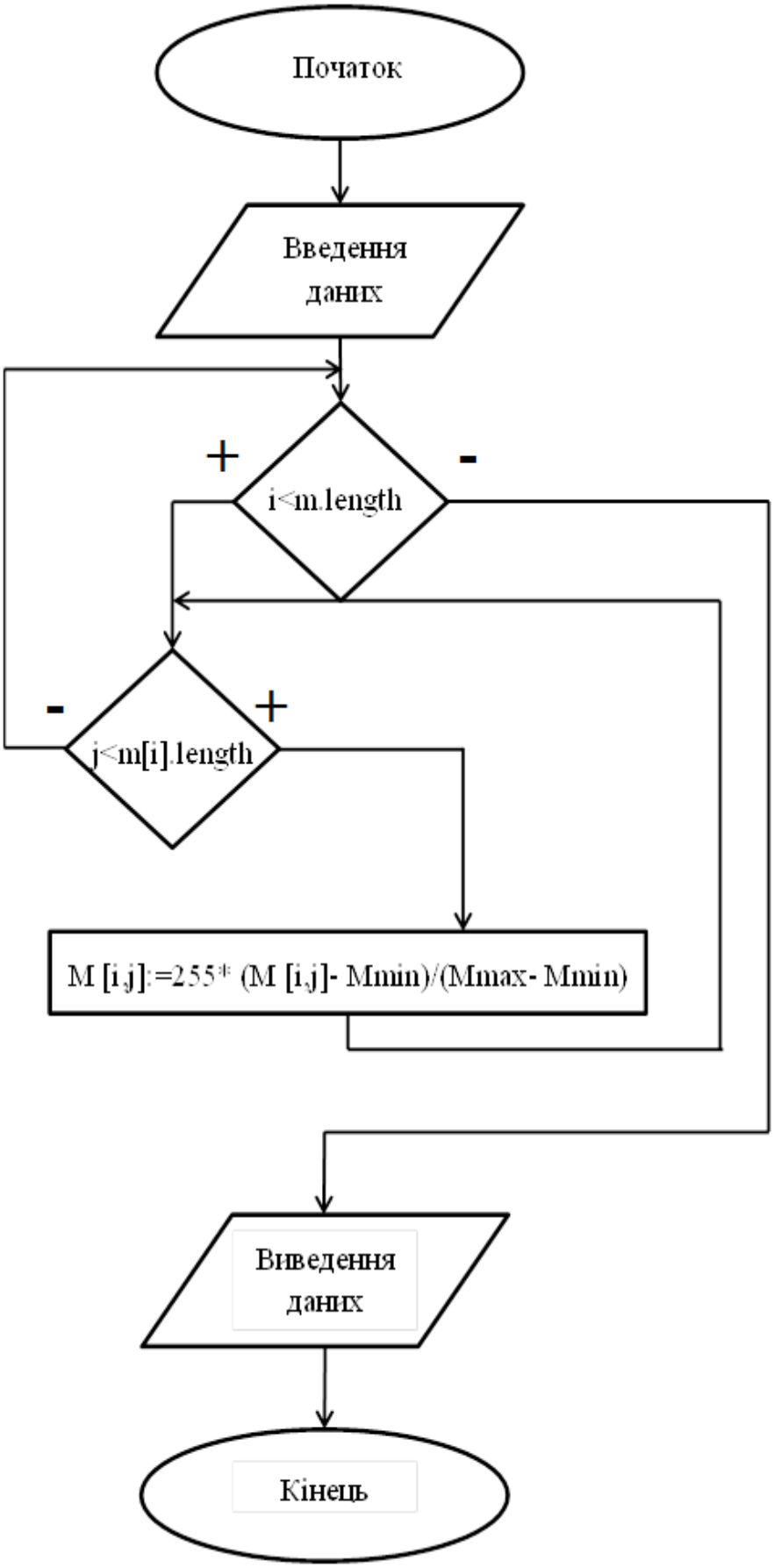

Рисунок 3.7 – Схема алгоритму нормалізації матриці

3.4 Обґрунтування вибору мови та середовища програмування

Дана магістерська кваліфікаційна робота реалізована на мові програмування Pascal, в середовищі розробки PascalABC.NET.

PascalABC.NET – це система програмування і мова Pascal нового покоління для платформи Microsoft .NET. Мова PascalABC.NET містить всі основні елементи сучасних мов програмування: модулі, класи, перевантаження операцій, інтерфейси, виключення, узагальнені класи, збірку сміття, лямбда-вирази, а також деякі засоби паралельності, в тому числі директиви OpenMP. Система PascalABC.NET включає в себе також просте інтегроване середовище, орієнтоване на ефективне навчання сучасного програмування.

Мова Паскаль була розроблена швейцарським вченим Ніколаусом Віртом в 1970 р як мова зі строгою типізацією і інтуїтивно зрозумілим синтаксисом. У 80-ті роки найбільш відомої реалізацією став компілятор Turbo Pascal фірми Borland, в 90-і роки двадцятого сторіччя йому на зміну прийшло середовище програмування Delphi, яке стала одним з кращих середовищ для швидкого створення додатків під Windows. Delphi ввела в мову Паскаль ряд вдалих об'єктно-орієнтованих розширень, оновлена мова отримала назву Object Pascal. З версії Delphi 7 мову Delphi Object Pascal став називатися просто Delphi. З альтернативних реалізацій Object Pascal слід зазначити багатоплатформений відкритий компілятор Free Pascal.

Створення PascalABC.NET диктувалося двома основними причинами: старіння стандартної мови Pascal і систем, побудованих на її основі (Free Pascal та інші), а також необхідність в сучасному простому, безкоштовному та потужному інтегрованому середовищі програмування.

PascalABC.NET базується на передовій платформі програмування Microsoft.NET, яка забезпечує мову PascalABC.NET величезною кількістю стандартних бібліотек і дозволяє легко поєднувати її з іншими .NET-мовами:

C #, Visual Basic.NET, керований C ++, Oxygene і ін. Платформа. NET надає також такі мовні засоби як єдиний механізм обробки виключень, єдиний механізм управління пам'яттю у вигляді збірки сміття, а також можливість вільного використання класів, успадкування, поліморфізму і інтерфейсів між модулями, написаними на різних .NET-мовами.

Мова PascalABC.NET близька до реалізації Delphi (Object Pascal). У ньому відсутній ряд специфічних мовних конструкцій Delphi, деякі конструкції змінені. Крім цього, доданий ряд можливостей: є автовизначення типу при описі, можна описувати змінні всередині блоку, є операції + =, - =, * =, / =, методи можна описувати безпосередньо в тілі класу або записи, можна користуватися вбудованими в стандартні типи методами і властивостями, пам'ять під об'єкти управляється збиральником сміття , множини можуть бути створені на основі довільних типів, введені оператори foreach, змінні циклів for і foreach можна описувати безпосередньо в заголовку циклу та ін.

Близькою за ідеологією до PascalABC.NET є мова RemObjects Oxygene (Object Pascal 21 століття). Однак вона сильно змінена у бік .NET: немає глобальних описів, всі описи поміщаються в клас, що містить статичний метод Main, відсутній ряд стандартних підпрограм мови Паскаль. Крім того, система RemObjects Oxygene - платна і не містить власної оболонки (вбудовується в Visual Studio і інші IDE).

Інтегроване середовище PascalABC.NET забезпечує підсвічування синтаксису, підказку за кодом (підказка по точці, підказка параметрів підпрограм, спливаюча підказка за кодом), форматування тексту програми за запитом, перехід до визначення і реалізації імені, елементи рефакторінга [14].

Сучасна мова програмування Object Pascal

Мова PascalABC.NET включає в себе практично весь стандартний мову Паскаль, а також більшість мовних розширень мови Delphi. Однак, цих засобів недостатньо для сучасного програмування. Саме тому PascalABC.NET розширено поруч конструкцій, а його стандартний модуль -

поруч підпрограм, типів і класів, що дозволяє створювати легко читаються додатки середньої складності.

Крім цього, мова PascalABC.NET використовує більшість засобів, що надаються платформою .NET: єдина система типів, класи, інтерфейси, виключення, делегати, перевантаження операцій, узагальнені типи (generics), методи розширення, лямбда-вирази.

Стандартний модуль PABCSystem, автоматично підключається до будь-якої програми, містить величезну кількість стандартних типів і підпрограм, що дозволяють писати ясні і компактні програми.

У розпорядженні PascalABC.NET знаходяться всі засоби .NET-бібліотек класів, постійно розширюються найсучаснішими можливостями. Це дозволяє легко створювати програми на PascalABC.NET додатки для роботи з мережею, Web, XML-документами, використовувати регулярні вирази і багато іншого.

Мова PascalABC.NET дозволяє програмувати в класичному процедурному стилі, в об'єктно-орієнтованому стилі і містить безліч елементів для програмування в функціональному стилі. Вибір стилю або комбінації цих стилів - справа смаку програміста, а при використанні в навчанні – методичний підхід викладача.

Поєднання багатих і сучасних мовних засобів, можливостей вибору різних траєкторій навчання дозволяє рекомендувати PascalABC.NET з одного боку як мову для навчання програмуванню (від школярів до студентів молодших і середніх курсів), з іншого - як мова для створення проектів та бібліотек середньої складності.

Проста і потужна середовище розробки

Інтегроване середовище розробки PascalABC.NET орієнтована на створення проектів малої та середньої складності. Вона досить легковажно і в той же час забезпечує розробника усіма необхідними засобами, такими як вбудований налагоджувач, засоби Intellisense (підказка по точці, підказка по

параметрам, підказка по імені), перехід до визначення і реалізації підпрограми, шаблони коду, автоформатування коду.

У середовище PascalABC.NET вбудований також дизайнер форм, що дозволяє створювати повноцінні віконні додатки в стилі RAD (Rapid Application Development – швидке створення додатків).

На відміну від багатьох професійних середовищ, середовище розробки PascalABC.NET не має громіздкого інтерфейсу і не створює безліч додаткових допоміжних файлів на диску при компіляції програми. Для невеликих програм це дозволяє дотримати принцип "Одна програма - один файл на диску".

У середовищі PascalABC.NET велику увагу приділено зв'язку запущеної програми з оболонкою: консольна програма, запущена з-під оболонки, здійснює введення-виведення в спеціальне вікно, вбудоване в оболонку. Можна також запустити кілька програм одночасно – всі вони будуть контролюватися оболонкою.

Інтегроване середовище PascalABC.NET дозволяє перемикати в налаштуваннях російську та англійську мову, при цьому локалізовані не тільки елементи інтерфейсу, але і повідомлення про помилки.

Крім цього, внутрішні уявлення PascalABC.NET дозволяють створювати компілятори інших мов програмування і вбудовувати їх в середу розробки за допомогою спеціальних плагінів.

Спеціалізовані модулі для навчання. Платформа Microsoft.NET забезпечує PascalABC.NET стандартної бібліотекою, що складається з величезної кількості класів для вирішення практично будь-яких завдань: від алгоритмічних до прикладних. Саме тому в PascalABC.NET відсутня необхідність в розробці широкого кола власних модулів.

Власні модулі, які є в PascalABC.NET, орієнтовані саме на початкове навчання програмуванню.

Крім цього, середовище PascalABC.NET містить модуль електронного задачника Programming Taskbook, що дозволяє здійснювати автоматичну постановку і перевірку завдань. Є також модулі для викладача, що дозволяють створювати завдання для виконавців і електронного задачника.

Модуль растрової графіки GraphWPF і модуль векторної графіки об'єктів WPFObjects і модуль тривимірної графіки Graph3D можуть бути використані для створення найпростіших графічних. а також інтерактивних анімаційних програм, керованих подіями [14].

3.5 Програмна реалізація інформаційної технології створення фотореалістичних зображень

В даного програмного модуля графічний інтерфейс складатиметься з діалогового вікна, кнопок для внесення змін в параметри, тектових повідомлень про зміни до параметрів та зображень .

1. Створення діалогового вікна.

```
begin
SetWindowCaption('Dimond-Square');
 SetWindowSize(1355,720);
 CenterWindow;
 init;
  end.
```

2. Створення кнопок для введення вхідних даних

```
procedure init;
  begin
   B1 := new ButtonABC(100, 250, 200, 50, 'Diamond-Square',
RGB(0,255,255));
B2 := new ButtonABC(100, 200, 100, 50, '+1', RGB(0,255,255));
```

```
B3 := new ButtonABC(200, 200, 100, 50, '-1', RGB(0,255,255));

B4 := new ButtonABC(100, 100, 50, 50, '+1', RGB(0,255,255));

B5 := new ButtonABC(150, 100, 50, 50, '-1', RGB(0,255,255));

B6 := new ButtonABC(200, 100, 50, 50, '+10', RGB(0,255,255));

B7 := new ButtonABC(250, 100, 50, 50, '-10', RGB(0,255,255));

B8 := new ButtonABC(100, 350, 50, 50, '+1', RGB(0,255,255));

B9 := new ButtonABC(150, 350, 50, 50, '-1', RGB(0,255,255));

B10 := new ButtonABC(200, 350, 50, 50, '+10', RGB(0,255,255));

B11 := new ButtonABC(250, 350, 50, 50, '-10', RGB(0,255,255));

B12:=new ButtonABC(100, 450, 200, 50, 'ДЫМ', RGB(0,255,255));
B13.OnClick := init;

B14 := new ButtonABC(100, 500, 200, 50, 'КАРТА', RGB(0,255,255));

B15 := new ButtonABC(100, 550, 200, 50, 'НЕБО',  RGB(0,255,255));

B16 := new ButtonABC(100, 600, 200, 50, 'МОРЕ',  RGB(0,255,255));

B17 := new ButtonABC(100, 650, 200, 50, 'ОГОНЬ',  RGB(0,255,255));

B18:=new ButtonABC(WindowWidth-301, 100, 200, 100, 'ВЫХОД',
RGB(255,0,0));
```

3. присвоєння та процедур для кожної кнопки графічного інтерфейсу

```
B1.OnClick := diamondsquare;

B2.OnClick := one;

B3.OnClick := eno;

B4.OnClick := sone;

B5.OnClick := seno;

B6.OnClick := sten;

B7.OnClick := snet;
```

```
B8.OnClick := mone;
B9.OnClick := meno;
B10.OnClick := mten;
B11.OnClick := mnet;
 B12.OnClick := smog;
 B13:=new ButtonABC(100, 0, 200, 50, 'Обновить', RGB(0,255,255));
B14.OnClick := map;
B15.OnClick := sky;
B16.OnClick := sea;
B17.OnClick := fire;
B18.OnClick := CloseWindow;
 N:=8;
 start:=128;
 maxrand:=100;
   Font.Size := 20;
Textout(100,165,('N=  '+inttostr(N)+'   '));
Textout(100,65,('Start='+inttostr(start)+'   '));
Textout(100,315,(' maxrand='+inttostr(maxrand)+'   '));
end;
```

4. опис поведінки кнопок

```
procedure mone;
begin
maxrand+=1;
if (maxrand<=127) and(maxrand>=0) then
 start:=start
```

```
 else
 maxrand:=((maxrand+128) mod 128) ;Font.Size := 20;
 Textout(100,315,('maxrand=  '+inttostr(maxrand)+'        '));
 end;
procedure meno;
begin
maxrand-=1;
if (maxrand<=127) and(maxrand>=0) then
 start:=start
 else
 maxrand:=((maxrand+128) mod 128) ;Font.Size := 20;
 Textout(100,315,('maxrand=  '+inttostr(maxrand)+'        '));
 end;
 procedure mten;
begin
maxrand+=10;
if (maxrand<=127) and(maxrand>=0) then
 start:=start
 else
 maxrand:=((maxrand+128) mod 128) ;Font.Size := 20;
 Textout(100,315,('maxrand=  '+inttostr(maxrand)+'        '));
 end;
 procedure mnet;
begin
maxrand-=10;
```

```
if (maxrand<=127) and(maxrand>=0) then
 start:=start
 else
 maxrand:=((maxrand+128) mod 128) ;Font.Size := 20;
 Textout(100,315,('maxrand=  '+inttostr(maxrand)+'        '));
 end;
```

5.створення функції виведення та збереження зображень

```
procedure smog ;
 begin
 for var ix:=0 to WindowWidth-1 do
   begin
 for var iy:=0 to WindowHeight-1 do
 begin
color:=(matrix[iy mod (L),ix mod (L)]-2*(matrix[iy mod (L),ix mod (L)]
div 255)*(matrix[iy mod (L),ix mod (L)] mod 255));
    SetPixel(ix,iy,RGB(color ,color , color ))
   end;
 end;
 SaveWindow('Дым.bmp');
 ClearWindow;
Textout(100,0,('Обновить '));
 end;
```

3.6 Тестування та аналіз результатів роботи програмного засобу

Розроблений програмний засіб створення аморфних об'єктів для фотореалістичних зображень був протестований, коректність його роботи була підтверджена. Було проведено 500 запусків програми, перевірено роботу його компонентів. Після запуску виконавчого файлу відкривається головне вікно, в якому виконуються всі дії. Головне вікно зображено на рисунку 3.8.

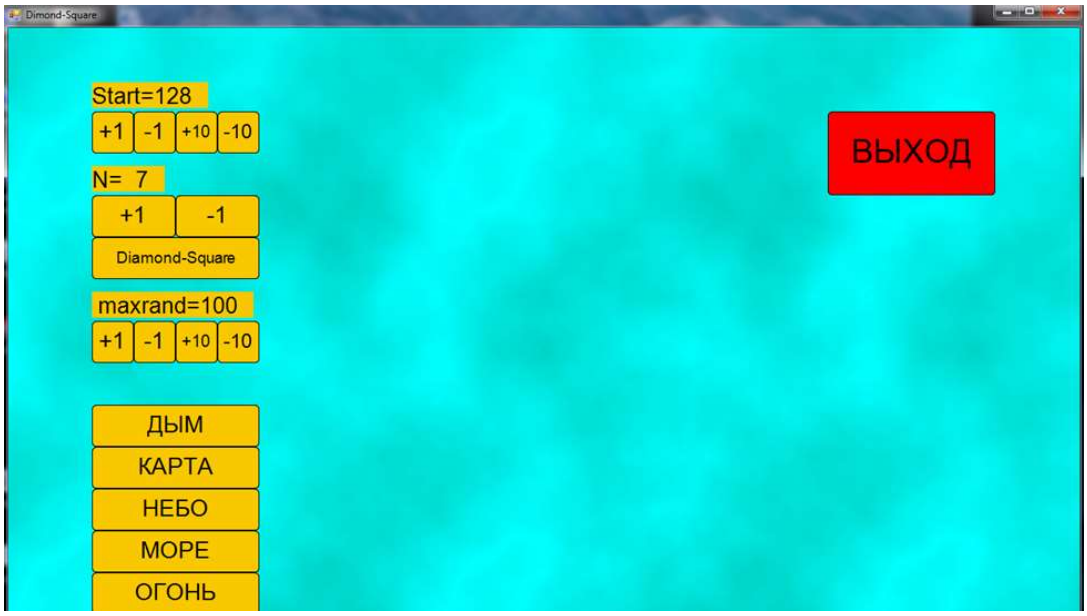

Рисунок 3.8 – Головне вікно програмного засобу

Після відкриття головного вікна програми можна приступати до введення початкових даних, які визначатимуть всі параметри створеного зображення. Вікно містить такі елементи: поле початкового значення матриці та кнопки регулювання цієї величини; поле порядку розмірності зображення та кнопки регулювання цієї величини; поле максимально можливої випадкової зміни значення елемента матриці та кнопки регулювання цієї змінної; кнопка розрахунку матриці; кнопки виведення на екран та збереження зображень

відповідного типу; кнопка «ВЫХОД» для завершення роботи програми та кнопка «ОБНОВИТЬ» для повернення до початкового стану програми.

Після введення початкових даних було створено зображення. Результат роботи програми показано на рисунку 3.9.

Після перевірки коректності роботи програми було проведено порівняльний аналіз швидкодії звичайного алгоритму Diamond-Square та вдосконаленого алгоритму Diamond-Square. Час виконання розрахунків матриць різних розмірностей наведено в таблиці 3.1.

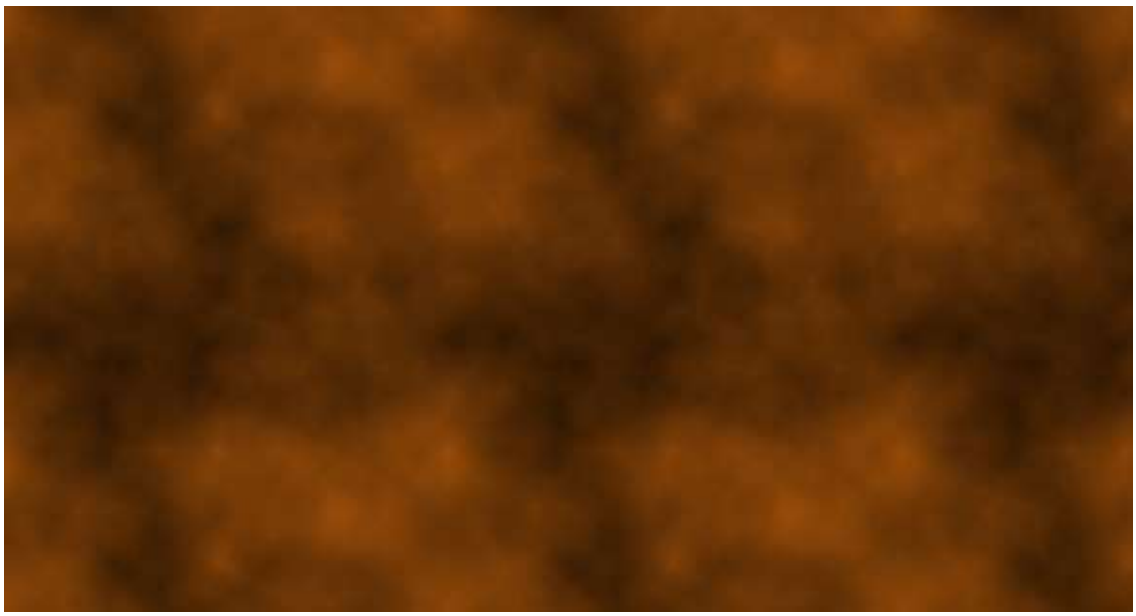

Рисунок 3.9 – Приклад створеного зображення.

Таблиця 3.1 – Час побудови матриці зображення в мілісекундах

Порядок матриці	Вдосконалений алгоритм	Звичайний алгоритм
7	3	4
8	12	14
9	49	54
10	205	227
11	817	889
12	2879	3123
13	10849	11978
14	42771	47338

На основі даних таблиці 3.1 побудовано графік залежності часу побудови матриці зображення від порядку розмірності цієї матриці. На рисунку 3.10 графік залежності на лінійній шкалі з ціною поділки 5000 мілісекунд. На рисунку 3.11 графік залежності на логарифмічній шкалі

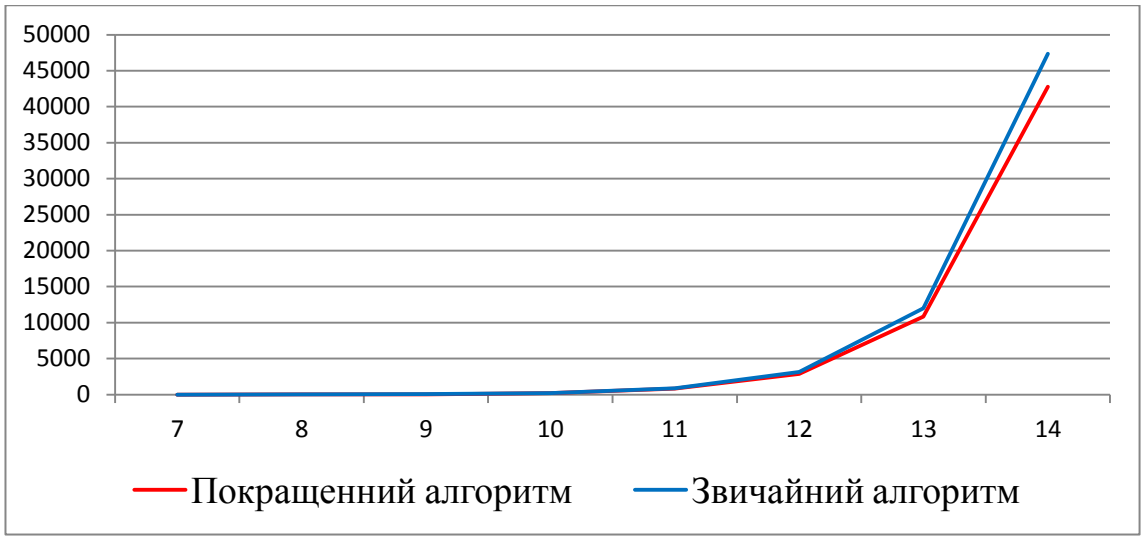

Рисунок 3.10 – Графік залежності часу розрахунку матриці від порядку її розмірності на лінійній шкалі

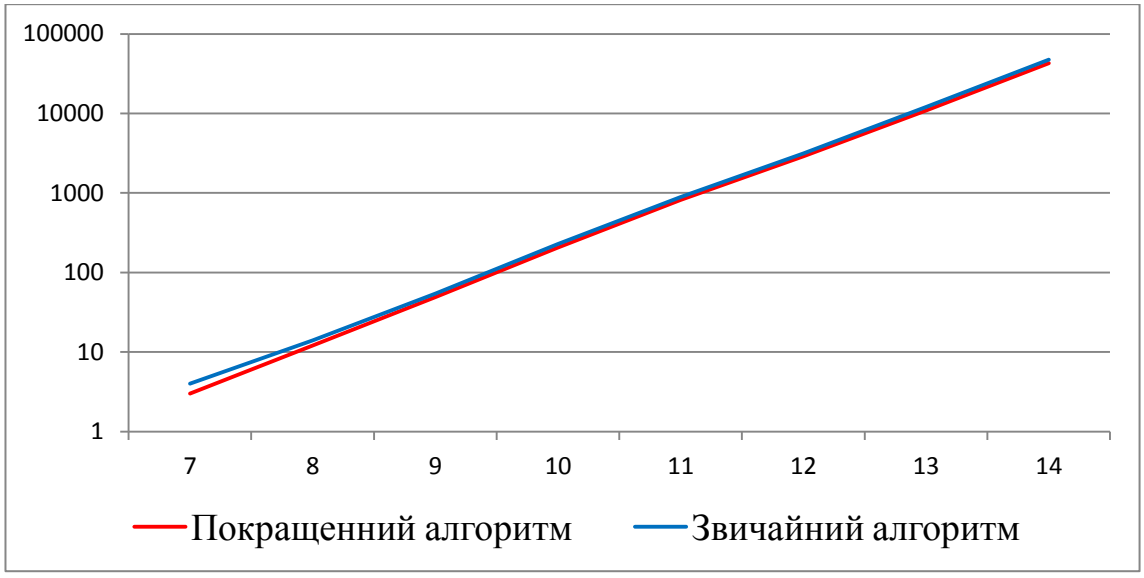

Рисунок 3.11– Графік залежності часу розрахунку матриці від порядку її розмірності на логарифмічній шкалі

3.7 Висновок

В третьому розділі магістерської кваліфікаційної роботи було відображено реалізацію інформаційної технології за допомогою мови Pascal в середовищі розробки PascslABC.net. Було обґрунтовано вибір мови програмування та середовища розробки. В реалізації інформаційної технології було використано вбудовані в середовище розробки засоби GraphABC і ABCObjects.

Також було наведено список додаткових бібліотек, що використовувались для спрощення програмної реалізації графічного інтерфейсу. У підсумку було проведено тестування функціоналу розробленого програмного засобу , яке показало коректність його роботи. Також виконано порівняння швидкості роботи вдосконаленого алгоритму Diamond-Square зі звичайним алгоритмом Diamond-Square, що показало зростання швидкодії на 9,8% . В цілому результати тестування співпадають з очікуваннями, а отже можна перейти до загальних висновків по роботі.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 3-х незалежних експертів.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5 бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	Яровий А. А.	Колесницький О. К.	Савчук Т. О.
	Бали виставлені експертами		
1	3	3	3
2	3	3	3
3	4	3	3
4	4	3	3
5	4	4	4
6	3	3	4
7	3	4	4
8	3	4	4
9	4	3	3
10	4	4	4
11	4	3	4
12	3	4	4
Сума балів	$СБ_1$=42	$СБ_2$=41	$СБ_3$=43
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{i=1}^{3} СБ_i}{3} = 42$		

Отже з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько-технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати:

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника;

$Т_р$ – кількість робочих днів у місяці, $Т_р$ = 22 дні;

t – число днів роботи розробника, t=50 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	6600	300	5	1500
Інженер-програміст	5500	250	50	12500
Всього:				14000

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0{,}1 \cdot 14000 = 1400 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 22% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп} = (З_{осн} + З_{ДОД}) \cdot \frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп} = (14000 + 1400) \cdot \frac{22}{100} = 3388 \text{ (грн.)}$$

Розрахунок амортизаційних витрат виконується за такою формулою:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, (12000 грн.);

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – термін використання (Т=3міс.).

Розрахунок амортизаційних відрахувань наведений в таблиці 4.3

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування	Балансова вартість	Норма амортизації %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
Персональний комп’ютер	12000	25%	3	750
Всього				750

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$К = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i, \quad (4.4)$$

де n – кількість комплектуючих;

$Н_і$ – кількість комплектуючих і-го виду;

$Ц_і$ – покупна ціна комплектуючих і-го виду грн.;

$К_і$ – коефіцієнт транспортних витрат (приймемо $К_і$ = 1,1).

Витрати на комплектуючі наведені в таблиці 4.4

Таблиця 4.4 – Витрати на комплектуючі, що були використані для розробки ПЗ

Найменування	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	Шт,	175	1	175
Папір	Упаковка.	150	1	150
Ручка	Шт.	25	1	25
Всього з урахуванням транспортних витрат				385

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.5)$$

де В – вартість 1кВт години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера(П=0,6 кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=300 год.);

$К_п$ – коефіцієнт використання потужності ($К_п$<1, $К_п$=0,7).

$$В_е = 1,7 \cdot 0,6 \cdot 300 \cdot 0,7 = 214,2 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $В_{ін}$ можна прийняти як (100...300)% від суми основної та додаткової заробітної плати розробників та робітників, які виконували дану роботу, тобто

$$В_{ін} = (1..3) \cdot (З_о + З_{дод}), \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 \cdot (14000 + 1400) = 15400 \text{ (грн.)}$$

Сума всіх попередніх витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_{дод} + Н_{зп} + А + К + В_е + В_{ін}$$

$$В = 14000 + 1400 + 3388 + 750 + 385 + 214{,}2 + 15400 = 35537{,}2 \text{ (грн.)}$$

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою:

$$В_{заг} = \frac{В}{\alpha}, \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях =1.

$$В_{заг} = \frac{35537{,}2}{1} = 35537{,}2 \text{ (грн.)}$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (4.8)$$

де β - коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{35537{,}2}{0{,}9} = 39485{,}78 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових засобів, які дозволять покращити фінансові результати діяльності.

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_{i=1}^{n} (\Delta П_я \cdot N + П_я \Delta N)_i \qquad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки в і-му році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_i$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшиться на 35 грн. (що автоматично спричинить збільшення чистого прибутку підприємства на 35 грн.), а кількість користувачів, які будуть користуватися, збільшиться:

протягом першого року – на 120 користувачів, протягом другого року – на 100 користувачів, протягом третього року – на 80 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 600 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 565 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже збільшення чистого продукту $\Delta П_i$ протягом першого року складатиме:

$$\Delta П_1 = 35 \cdot 600 + (565 + 35) \cdot 120 = 93000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 35 \cdot 600 + (565 + 35) \cdot (120 + 100) = 153000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 35 \cdot 600 + (565 + 35) \cdot (120 + 100 + 80) = 201000 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$ вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (4.10)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;

PV – теперішня вартість інвестицій PV=ЗВ, грн.

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_{i=1}^{т} \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн.

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3, 4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

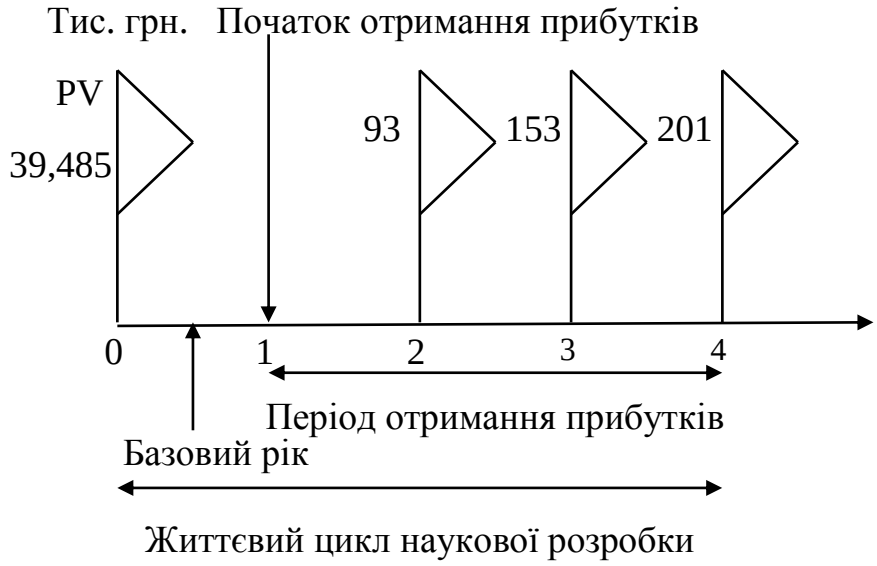

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{93000}{(1,1)^2} + \frac{153000}{(1,1)^3} + \frac{201000}{(1,1)^4} = 329096,37(грн.)$$

Тоді розрахуємо $Е_{абс}$:

$$Е_{абс} = 329096,37 - 39485,78 = 289610,59 \text{ грн.}$$

Оскільки $Е_{абс}>0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (4.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн.;

PV – теперішня вартість інвестицій PV=ЗВ, грн.;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{289610,59}{39485,78}} - 1 = 1,03$$

Далі, розраховану величину $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування τ_{min}, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ_{min} визначається за формулою:

$$\tau = d + f, \tag{4.13}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d=-0,1;

f – показник, що характеризує ризикованість вкладень, величина f=0,1;

$$\tau = 0{,}1 + 0{,}1 = 0{,}2$$

Оскільки $E_в$=103%>τ_{min} =0,2=20%, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Розрахуємо термін окупності інвестицій.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{4.14}$$

$$T_{ок} = \frac{1}{1{,}03} = 0{,}97 \text{ року.}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено та програмно реалізовано інформаційну технологію створення аморфних об’єктів для фотореалістичних зображень. Під час аналізу предметної області було відзначено, що створення аморфних об’єктів для фотореалістичних зображень може знайти своє застосування в кіновиробництві, ігровій індустрії та створенні різних анімованих зображень. Проте сьогодні не існує досконалого засобу, який є зручним в використані, та дозволяє з достатньою швидкістю створювати аморфні об’єкти для фотореалістичних зображень.

В даній магістерській кваліфікаційній роботі було розглянуто та проаналізовано існуючі програмні реалізації розв’язання задачі створення аморфних об’єктів для фотореалістичних зображень; запропоновано математичну модель на основі алгоритму «Diamond-Square» для розробки інформаційної технології створення аморфних об’єктів для фотореалістичних зображень; визначено складові інформаційної технології та на їх основі розроблено структуру та алгоритм роботи програмного засобу; реалізовано розроблено інформаційну технологію створення аморфних об’єктів для фотореалістичних зображень на мові програмування Pascal в середовищі PascalABC.Net; було проведено тестування функціоналу розробленого програмного засобу , яке показало коректність його роботи. Також виконано порівняння швидкості роботи вдосконаленого алгоритму Diamond-Square зі звичайним алгоритмом Diamond-Square, що показало зростання швидкодії на 9,8%. Це свідчить про досягнення мети магістерської кваліфікаційної роботи, а саме підвищення швидкодії створення аморфних об’єктів для фотореалістичних зображень.

В результаті виконання даної кваліфікаційної роботи поставлені задачі були виконані в повній мірі

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Матеріали XII міжнародної науково-практичній конференції «ІОН-2020» [Електронний ресурс] – Режим доступу: http://ies.vntu.edu.ua/ru/ies2020/report/proceedings2020

2. Стрижалов О. І. Аналіз методів й алгоритмів генерації фрактальних шумів для задач генерації аморфних об’єктів.. НТКП ВНТУ. Стрижалов О. І [Електронний ресурс] – режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9275

3. Мандельброт Б. Фрактальная геометрия природы. — Москва: Институт компьютерных исследований, 2002, 656 стр.

4. Сайт Типи комп’ютерної графіки. [Електронний ресурс] – режим доступу: https://studopedia.com.ua/1_57942_tipi-kompyuternoyi-grafIki.html

5. Сайт Введение в компьютерную графику. Визуализация природных явлений. [Електронний ресурс] – режим доступу: http://graphicon.ru/oldgr/courses/cg02b/assigns/hw-5/hw5_cld.htm

6. Сайт Алгоритм «diamond-square» для построения фрактальных ландшафтов. [Електронний ресурс] – режим доступу: https://habr.com/ru/post/111538/

7. Сайт graphics. pixar.com libraryWaveletNoise [Електронний ресурс] – режим доступу:https://webcache.googleusercontent.com/search?q=cache:Ck1vkQTekqUJ:https://graphics.pixar.com/library/WaveletNoise/paper.pdf+&cd=1&hl=ru&ct=clnk&gl=pl&client=opera

8. Сайт progipro [Електронний ресурс] – режим доступу: https://progi.pro/simplex-noise-t28708

9. Сайт sidefx.com [Електронний ресурс] – режим доступу: https://www.sidefx.com/docs/houdini/nodes/vop/worleynoise.html

10. Сайт neurohive [Електронний ресурс] – режим доступу: https://neurohive.io/ru/tag/gradientnyj-shum/

11. Сайт Цвет в компьютерной графике. Цветовые модели. [Електронний ресурс] – режим доступу: http://eor.dgu.ru/lectures_f/Курс_леций_Компьютерная_геометрия_и_графика_Гаджиев_А_М/лекция_4.htm

12. Буч, Г. UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон. − М.: ДМК Пресс; Издание 2-е, стер., 2014. − 432 с.

13. Сайт Справка PascalABC.NET [Електронний ресурс] – режим доступу: http://pascalabc.net/downloads/pabcnethelp/index.htm