МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

«РОЗРОБКА АЛГОРИТМУ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ У МУЛЬТИМЕДІЙНІЙ СФЕРІ»

Спеціальність 122 «Комп'ютерні наук»

122 – БКР – 401.21810126

Виконав студент 4-го курсу, групи 401
_____ А.О.Буданов
«22» червня 2022 р.

Керівник:д.п.н., професор
_____О.П.Мєщанінов
«22» червня 2022 р.

Миколаїв – 2022

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема:_____Розробка алгоритму процедурної генерації в мультимедійній сфері

№	Найменування роботи	Початок	Закінчення	Примітки
	Вибір теми дипломної роботи	27.11.2021	30.11.2021	
	Затвердження теми дипломної роботи	08.12.2021	14.01.2022	
	Підготування календарного плану	19.01.2022	19.01.2022	
	Дослідження мови ООП, на якому реалізується алгоритм	01.03.2022	15.03.2022	
	Розробка алгоритму	16.03.2022	.2022	
	Написання анотацій	24.04.2022	24.04.2022	
	Написання І розділу	24.04.2022	25.04.2022	
	Написання ІІ розділу	26.04.2022	28.04.2022	
	Написання ІІІ розділу	05.05.2022	.2022	
	Написання спеціального розділу «Охорона праці»	10.05.2022	17.05.2022	
	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	18.05.2022	20.05.2022	
	Подання доопрацьованого варіанту роботи керівнику	22.05.2022	23.05.2022	
	Подання роботи на кафедру, передзахист дипломної роботи	31.05.2022	31.05.2022	
	Подання переплетеної роботи на кафедру для направлення на зовнішнє рецензування і отримання допуску до захисту	15.06.2022	15.06.2022	
	Захист дипломної роботи	22.06.2022	22.06.2022	

Розробив студент ____Буданов А.О._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи _____д.п.н. професор Мєщанінов О.П___ _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«__22__» _____червня_____ 2022р.

Анотація

До бакалаврської кваліфікаційної роботи роботи студента 401 групи Буданова Андрія Олеговича на тему: «Розробка алгоритму процедурної генерації у мультимедійній сфері».

Метою роботи є розробка власного алгоритму процедурної генерації контенту, теоретичне дослідження проведено шляхом вивчення спеціалізованої літератури, наукових статей та інших публікацій згідно з темою.

У вступі розкрито актуальність обраної теми, наведений опис мети роботи та поставлені цілі, що необхідно досягти під час розробки.

У першому розділі розкривається поняття «процедурної генерації», описуються вже відомі алгоритми, що реалізують методи процедурної генерації контенту. Описується застосування цих алгоритмів у мультимедії, наводяться приклади застосування.

Другий розділ розширює досліджуване поняття шляхом більш детального та поглибленого опису існуючих методів та алгоритмів, із реалізацією програмного коду для більш наглядної демонстрації роботи перелічених алгоритмів.

У третьому розділі описується розробка структури власного алгоритму процедурної генерації, приводиться блок-схема, розробляються методи та формуються бібліотеки. Алгоритм програмно реалізується та пояснюється його робота та застосування.

Висновок узагальнює проведене дослідження теми дипломної роботи, формуються тези про переваги та недоліки методів процедурної генерації. Розроблений, за результатами роботи, алгоритм може бути використаний у подальших проектах, що потребують алгоритмічної генерації контенту.

Дипломна робота містить 68 сторінок друкованого тексту, 33 рисунків, 0 додатків, 21 використаних першоджерел.

Annotation

An annotation of Andrii Budanov Bachelor thesis, which topic is “Development of procedural generation algorithm in multimedia”.

The main goal of the work was to develop own procedural generation algorithm. Theoretical research was achieved by learning specialized literature, science articles, and other publications according to the topic.

An introduction explores topic, its relevance, describes purpose of thesis and it goals, that must be achieved during the development.

A first chapter describes the meaning of procedural generation by presentation of existing algorithms and their usage in multimedia with examples.

A second chapter expands the meaning of procedural generation by delving more into the topic, describing existing methods and algorithms, and showing their practical usage.

A third chapter describes a development of structure student’s own algorithm of procedural generation. That chapter shows development process, algorithm’s flowchart, program realization, describes algorithm work and its usage.

A conclusion of the thesis generalize whole topic of the research that was made. In this chapter was formed benefits and downsides of the procedural network’s methods, The algorithm, developed during the thesis, can be useful for future projects that will demand algorithmically generated content.

Зміст

Анотація 3
Annotation 4
Перелік скорочень 6
Вступ 8
1 Загальні положення процедурної генерації 10
1.1. Поняття процедурної генерації 10
1.2 Опис алгоритмів процедурної генерації 15
1.3 Приклади застосування процедурної генерації у мультимедії 22
Постановка задачі 25
Висновок до розділу 1 27
2 Методи процедурної генерації 28
2.1 Алгоритм Форчуна 28
2.2 Мозаїка Вороного 32
2.3 Синтез текстур методом k-найближчих сусідів та клітинних автоматів 36
Висновки до розділу 2 41
3 Моделювання та проектування інформаційної системи 42
3.1 Структура системи 42
3.2 Розробка алгоритму 49
3.3 Керівництво користувача 53
Висновки до розділу 3 55
ОХОРОНА ПРАЦІ 56
Фактичні та нормативні значення параметрів мікроклімату 60
Висновки 65
Перелік посилань 67

Перелік скорочень

PCG – Procedural Generation

КА – Клітинні автомати

MRF – випадкові поля Маркова

TSVQ – Tree Structure Vector Quantizer

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«Розробка алгоритму процедурної генерації у мультимедійній сфері»

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 401.21810126

Виконав студент 4-го курсу, групи 40Х
_____*А.О.Буданов*
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Керівник: *д.п.н. професор*
(наук. ступінь, вчене звання)
_____*О.П.Мещанінов*
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

Вступ

Метою бакалаврської кваліфікаційної роботи є дослідження методів алгоритмів процедурної генерації контенту, розробка та програмна реалізація власного алгоритму, що застосовує досліджені методи. У результаті роботи буде розроблений програмний код алгоритму, що дозволяє генерувати зображення двовимірних карт із подальшим розвитком у повноцінний генератор мап.

Procedural generation [PCG] – це узагальнена назва групи методів, що користуючись обчислювальними алгоритмами створюють різні графічні моделі, наприклад: будинок, приміщення, ландшафт, рослини, тварини, текстури, тощо.

Методи побудови PCG актуальні у наш час у багатьох сферах діяльності, що пов’язана із комп’ютерною графікою. Використовуючи ці методи, розробники використовують отримані моделі у мультимедійному просторі, до якого можна віднести: мультфільми, фільми, комп’ютерні ігри, рендери об’єктів. Найбільш придатним до використання методи PCG використовують у разі, коли потрібно отримати велику кількість унікальних моделей у невеликий проміжок часу й тоді, коли способи побудови об’єктів вручну вважаються нерентабельними, наприклад методи PCG дозволяють замінити роботу художників та дизайнерів у короткочасній перспективі.

Порівнюючи автоматичні методи створення графічного контенту та більш традиційний спосіб за участі художників, можна виділити наступні переваги процедурної генерації:

- Можливість змінювати параметри генерації у любий момент розробки, що дозволяє уникнути ручного редагування елементів.
- Можливість не втрачати деталізованість зображення при зміні його розмірів.

- Швидка генерація та редагування моделей. Більшість генераторів, що використовують методи PCG, потребують зміни вхідного значення Seed.
- Невеликий розмір пам’яті. Самі алгоритми займають у тисячі разів менше пам’яті, що не залежить від роздільності зображення.

Для побудови власного генератора, у якому використовуються методи PCG, була досліджена наукова література та визначені головні завдання для успішного виконання завдання, а саме: дослідити існуючі алгоритми процедурної генерації та їх можливості, визначити найбільш підходящий алгоритм для вирішення завдання та розробити програму-генератор за допомогою об’єктно орієнтованої мови C#.

1 Загальні положення процедурної генерації

1.1. Поняття процедурної генерації

Методи процедурної генерації (PCG) – це методи створення даних за допомогою потужностей електронно-обчислювальних машин. Зазвичай ці методи протиставлять ручному способу створення контенту, тобто за участі художників, дизайнерів, аніматорів, тощо. Зазвичай використовують поєднання вручну створених асетів та змогу ЕВМ генерувати випадкові значення. Методи PCG використовують у комп'ютерній графіці для створення 3D моделей та текстур. У мультимедії їх використовують, щоб уникнути багатозатратної з фінансового та часового плану роботи художників та аніматорів.

Широкого росповсюдження PCG набуло у роботі з процесом розробки відеоігр. Під терміном «контент» у цій сфері, методи PCG можуть спростити роботу для гейм дизайнерів генеруючи різні моделі, наприклад: окремі рівні, предмети, квести, локації, персонажів та багато іншого. Ключовою властивістю є те, що створений контент є іграбельним, тобто гравець повинен мати змогу переміщуватись по сгенерованим локаціям, використовувати сгенеровані предмети, пройти сгенерований рівень і так далі.

Терміни «процедурний» та «генерація» відносяться до комп'ютерної термінології, так як програмне забезпечення PCG має працювати на комп'ютері у вигляді процедури, тобто підпрограми, що містить у собі певний опис дій – алгоритм, та може бути викликана у процесі роботи програми багато разів, у цьому випадку для графічної генерації чого-небудь.

Процедурну генерацію відносять до одного із напрямків синтетичного медіа простору. Синтетичне медіа – це результат роботи комп'ютера, що завдяки алгоритмам, які маніпулюють та модифікують дані, можуть створювати нові зображення, музику, анімацію. Із поширенням нейронних мереж, синтетичне медіа набуло високої популярності та проявляється у повсякденному житті, наприклад голос, яким розмовляють голосові помічники Siri, Cortana, або помічник від Google.

Створюваний контент повинен задовольняти певним умовам і в такий спосіб вирішувати відповідні проблеми, і практично найчастіше розглядаються такі властивості:

- Різноманітність: створення такого контенту, який був би якомога різним на різних запусках, і при погляді на нього гравець не відчував би одноманітності. Так один генератор може створити багато ітерацій схожого, але водночас різноманітного контенту.

- Здатність контролювати згенерований контент виходячи із ситуації та надання геймдизайнеру відповідної свободи (дана властивість потрібна далеко не завжди); наприклад, генерація гладкого довгастого каменю або створення рівня з певною атмосферою.

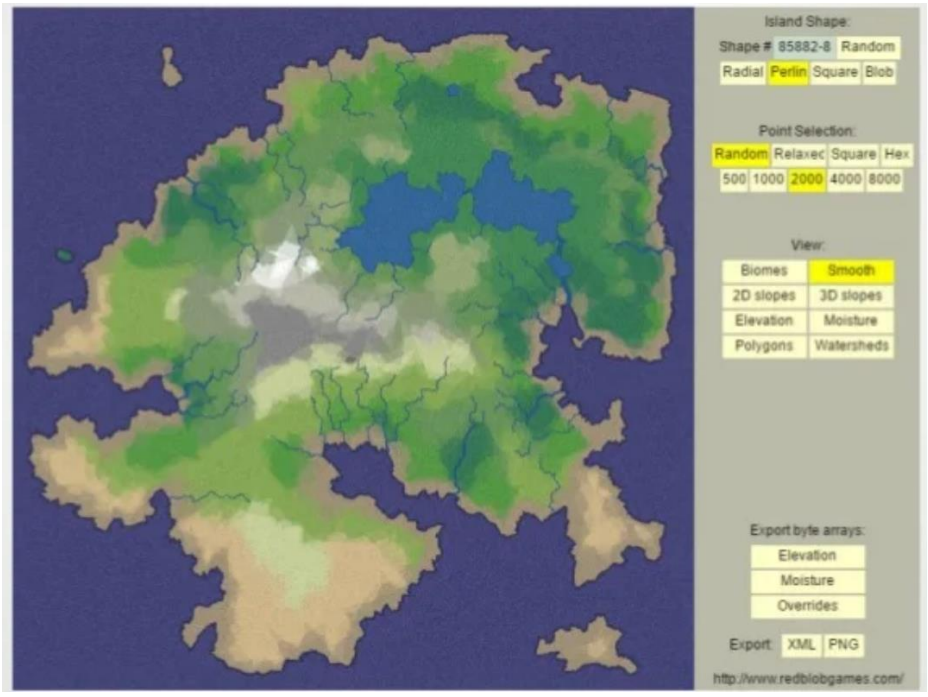

Рисунок 1.1 – створення ландшафту Землі із можливістю керувати параметрами кожної нової генерації

- Швидкість: в залежності від завдання вимоги відрізняються від мілісекунд до місяців, але в загальному випадку контент повинен бути створений вчасно для задоволення потреб ігрового процесу.

- Економія часу: за допомогою PCG можна створювати велику кількість контенту швидше, ніж створювати його вручну. Багато географічно великих.

- Багаторазовий код: генератори, що реалізують алгоритми, можуть бути використані у різних додатках. Наприклад генератор, що створює ландшафт пустелі в одній грі може мати декілька ключових змінних, при редагуванні яких, цей генератор створює льодяні гори в іншій, модуль, що відтворює симуляцію вогню, може відтворювати симуляцію захворювань. Коли розробник працює над проектами один за одним, то така багаторазовість і модульність генераторів є безцінною.

- Виконавчі стандарти: генератори можуть бути виконані за певними виконавчими стандартами, що забезпечують зв’язок між усіма елементами коду під час його роботи. Такий код, якщо він написаний без помилок, буде більш строгим у забезпечені виконавчих стандартів, ніж людина. Архітектори полягають на ці правила, щоб забезпечити виконання стандартів (та виконання законів фізики) у нових проектах.

- Моделювання реальності: зазвичай, техніки симуляції використовуються у генераторах у вигляді симуляції погоди, відтворення покриття ландшафту за правилами природи, тобто уникаючи знаходження хвойного лісу на одній клітині та активного вулкану на іншій. Клітинні генератори відтворюють життєві умови та можуть бути особливо ефективними в умовах розробки деталей, що симулюють безладдя реального життя.

- Подолання технічних обмежень: у ранній історії відеоігор, алгоритми PCG використовували для генерації такої кількості контенту, що було б неможливо зберігати на носіях того часу. Наприклад у Frontier: Elite II була ціла галактика, включаючи зірки та планети, що зберігались на носії

розміром 720 КВ – це менше, ніж розмір однієї простої текстури у наш час. У наш час, методи PCG також долають технічні обмеження, наприклад генерація різноманітних зображень або музики з невеликого за обсягом коду. Ця властивість може здатися безглуздою, враховуючи сучасне комп'ютерне обладнання, але потенціал цієї властивості виражається у мініатюрних пристроях, наприклад електронний годинник, інтегрована електроніка, та інше обладнання, що мають технічні обмеження.

Існують також більш суб'єктивні причини й унікальні властивості використовувати методи PCG у своєму проекті:

- Індивідуальний досвід: PCG має потенціал надати кожному окремому користувачу свій унікальний досвід. Згенерований фрагмент музики, який ніхто до цього не чув та ніколи не почує. Унікальні ігрові ситуації для кожного користувача. Методи PCG генерують шматки нескінченності для кожного окремого користувача, роблячи досвід недосяжним для статичного контенту. Розвинені алгоритми можуть генерувати контент тісно пов'язаний з користувачем, виробляючи персоналізований досвід для кожного. З таким інтерактивним видом медіа, як відеогра, це надзвичайно потужний інструмент, що робить ігровий процес запам'ятовуючим і таким, враженням від якого варто поділитися з іншими.

- Неможливість передбачити: навіть дизайнер не може знати, що очікувати від генератора, який він розробляє. З перспективи QA це може прозвучати лякаюче, але для розробника це може бути захоплюючим. Це дозволяє розробнику насолоджуватися процесом роботи та переживати схожий з кінцевим користувачем досвід.

- Жива симуляція: процедурні алгоритми можуть створювати живі ефекти. Наприклад: воду, вогонь, лаву, забруднення, систему погоди, симуляцію населення, цивілізації, культури, тобто життя у всіх його проявах.

Контент, що створюються вручну, завжди буде здаватися репетативним порівнюючи з непередбачуваним процедурно згенерованим. Деякі алгоритми, як наприклад алгоритм вибору найближчих сусідів, можуть генерувати величезні та заплутані у масштабі ефекти, але такі, що відчуваються для нас реальними, тому що базуються на реально існуючих шаблонах.

- Штучна креативність: як вже було зазначено, контент згенерований методами PCG може бути неймовірним. Вони можуть створювати речі, до яких би ніколи не додумалась людина, від надзвичайних дизайнів до неймовірних анімацій. Можна знайти безліч випадків процедурної анімації у 3D відеоіграх, або невдалі рендери, на яких зображені лякаючі та смішні істоти, що штучно згенеровані алгоритмами PCG, до яких людська уява би ніколи не дійшла. Комп'ютер не мислить у такий спосіб, як мислить людина, що дозволяє отримувати неймовірний, але логічний результат.

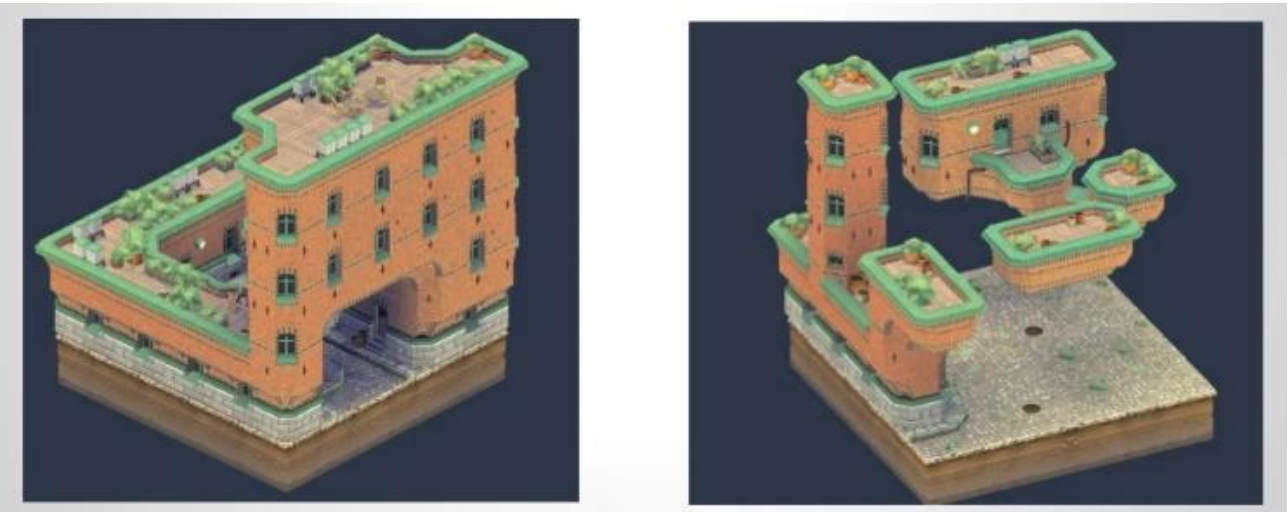

Рисунок 1.2 – процедурно сгенеровані будинки.

- Це просто весело: в цілому, працювати з процедурними генераторами весело. Натискаючи кнопку згенерувати щось нове, потім змінити декілька змінних і згенерувати зовсім інше, що не схоже на попереднє. Тисячоліттями людство розважалось завдяки гральних кубиків та карт, у наш час, можна сказати, що процедурна генерація має у собі мільйони кубиків, якими людина можете маніпулювати, щоб отримати різноманітну кількість персоналізованого контенту. Процедурні генератори – це такий дизайн

дизайнера, що створює більше, ніж людина може досягти сама, а розробка процедурного генератора – це створення творця.

1.2 Опис алгоритмів процедурної генерації

Модульність вважається однією із властивостей коду алгоритмів процедурної генерації. Але це поняття відноситься не тільки до коду, а й безпосередньо до одного із методів PCG. Модульність використовує окремі одиниці, що називають модулями, для того, щоб створювати із них великі структури, що називають гештальтами. Гештальти – це об’єкти, що згенеровані процедурно алгоритмами схрещення іншних об'єктів. Практично це: динамічні головоломки, нові рівні, біоми, дерева діалогів.

У контексті дизайну алгоритму, що виконує процедурну генерацію, модульність повинна задовольнятися двома якостями:

1) Механізм, що виконує сбір гештальтів, включає в себе деякий рівень випадковості.
2) Пам’ять, яку займають гештальти, занадто велика, щоб мати змогу виконати їх вручну.

Отже, проектування алгоритмів модульної генерації мають дві мети. Спроектувати модулі, із яких будуть збиратися гештальти, та спроектувати механізм, що оперує цими модулями.

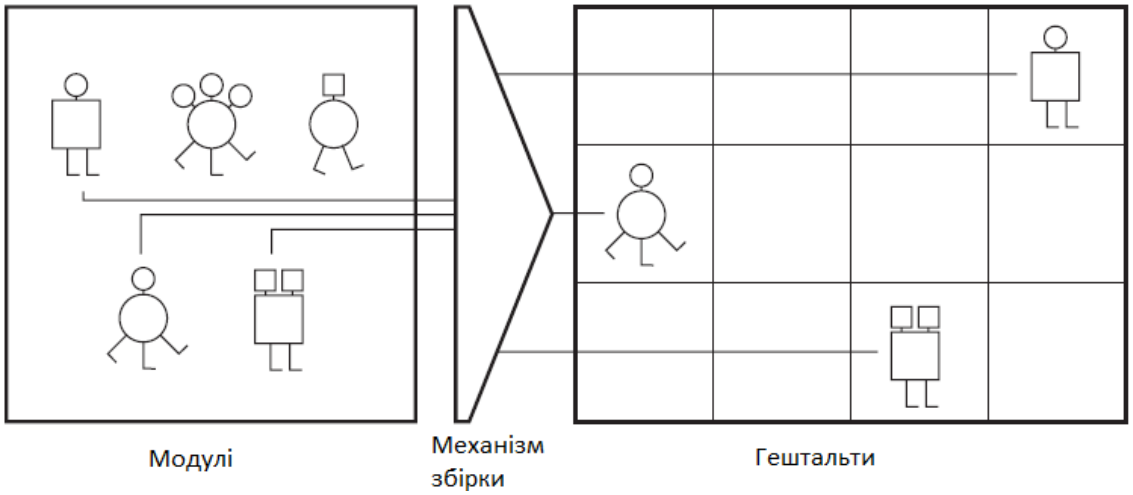

Рисунок 1.3 – приклад модульного алгоритму процедурної генерації популяції

Більшість модульних алгоритмів генерують гештальти з певними обмеженням відповідно до розміру класу. Наприклад, на рисунку 1 зображений алгоритм заповнення кімнати розміром 4 х 3 монстрами, яких 5 типів, кожен може бути обраний повторно. Наведений алгоритм популяції дуже простий: вибрати випадковий об’єкт із модуля, додати його до кімнати, повторяти до заданого обмеження.

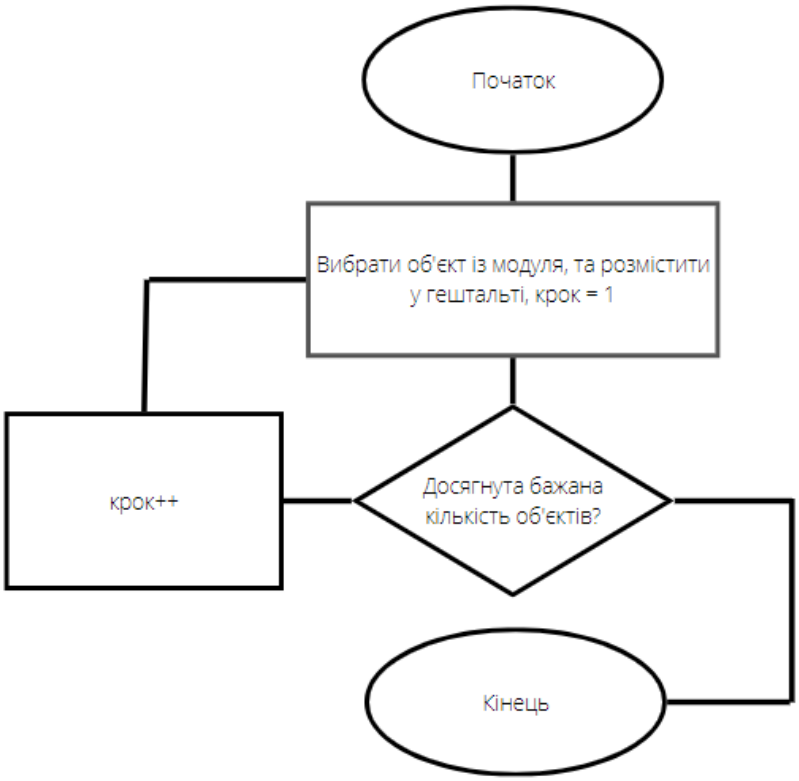

Рисунок 1.4 – блок-схема алгоритму популяції.

У цьому прикладі, тип монстру – це модулі, алгоритм популяції – це механізм сбірки, кімната – це гештальт. Алгоритм може виконати $5 \times 5 \times 5 = 125$ варіантів вибору монстрів, але інше питання постає скільки гештальтів може створити даний алгоритм. У цьому прикладі неважливо у якому порядку вибираються монстри, такий вид гештальту називається комбінацією і кількість можливих гештальтів визначається за наступною формулою:

$$\frac{(r+n-1)!}{r!(n-1)!},$$

де r – кількість можливих варіантів, n = заданий розмір гештальту. Застосувавши формулу до наведеного прикладу, отримається 35 можливих гештальтів.

Вид гештальту, який дотримується правилам певного порядку, називається перестановка.

Можна побачити, що розміри пам’яті, що займає гештальт, збільшується відповідно до збільшення параметрів модуля та механізму збірки.

Таблиця 1.1 – варіанти гештальтів

Модулі	Механізм збірки	Кількість можливих гештальтів
5	3	35
10	3	220
25	3	2925
25	5	118755
25	10	131128140

Ще один популярний метод процедурної генерації використовується для створення випадкового та логічного ландшафту, наприклад генерації приміщень будівлі, архітектури міст, або банального підземелля, приклад якого і буде розглядатись.

Алгоритм генерує дані кімнати у форматі ASCII літер та зберігає їх у окремому текстовому файлі, у цьому прикладі межі генерації визначено прямокутником $x \times y$, але дані кімнати генеруються не в прямокутному форматі.

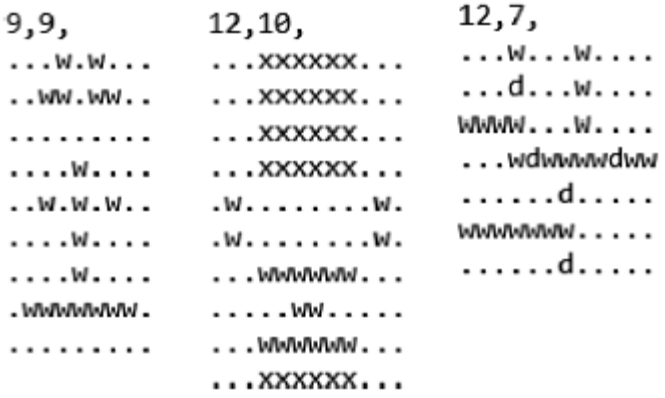

Рисунок 1.5 – результат роботи алгоритму процедурної генерації

На рис.2 зображено три згенеровані кімнати у форматі ASCII таблиці, де перший рядок позначає розміри кімнати, x – невикористане місце, щоб кімнати щільно прилягали один до одної, w – тайли стін, « . » - вільне місце, наприклад для розміщення чогось, або пересування гравця, d – двері, до якої прилягає наступна кімната.

Алгоритм процедурної генерації розміщення згенерованих кімнат полягає у наступному:

1) визначити розміри зони, у яких будуть розміщуватись кімнати, та визначити усі тайли у цій зоні невикористаними, тобто позначити x;
2) розмістити у цій зоні початкову кімнату;
3) із згенерованих кімнати вибрати такі, що відповідають розмірам невикористаного місця та сформувати із них колекцію, визначивши такі об’єкти кандидатами, із якої випадково буде обрано кімнату для розміщення;
4) вибрати кандидата із колекції та розмістити його у довільному напрямку перпендикулярно до початкового кандидата;

5.1) якщо новий кандидат підходить відповідно до старого без явних помилок, та не покидає межі невикористаного простору, то позначити нового кандидата старим та додати двері у позицію старого кандидата;

5.2) якщо виникають помилки, пересікаються стіни, то повернутися до кроку 3;

6) повернутися до кроку 3 та повторювати доки не буде здобуто ліміту.

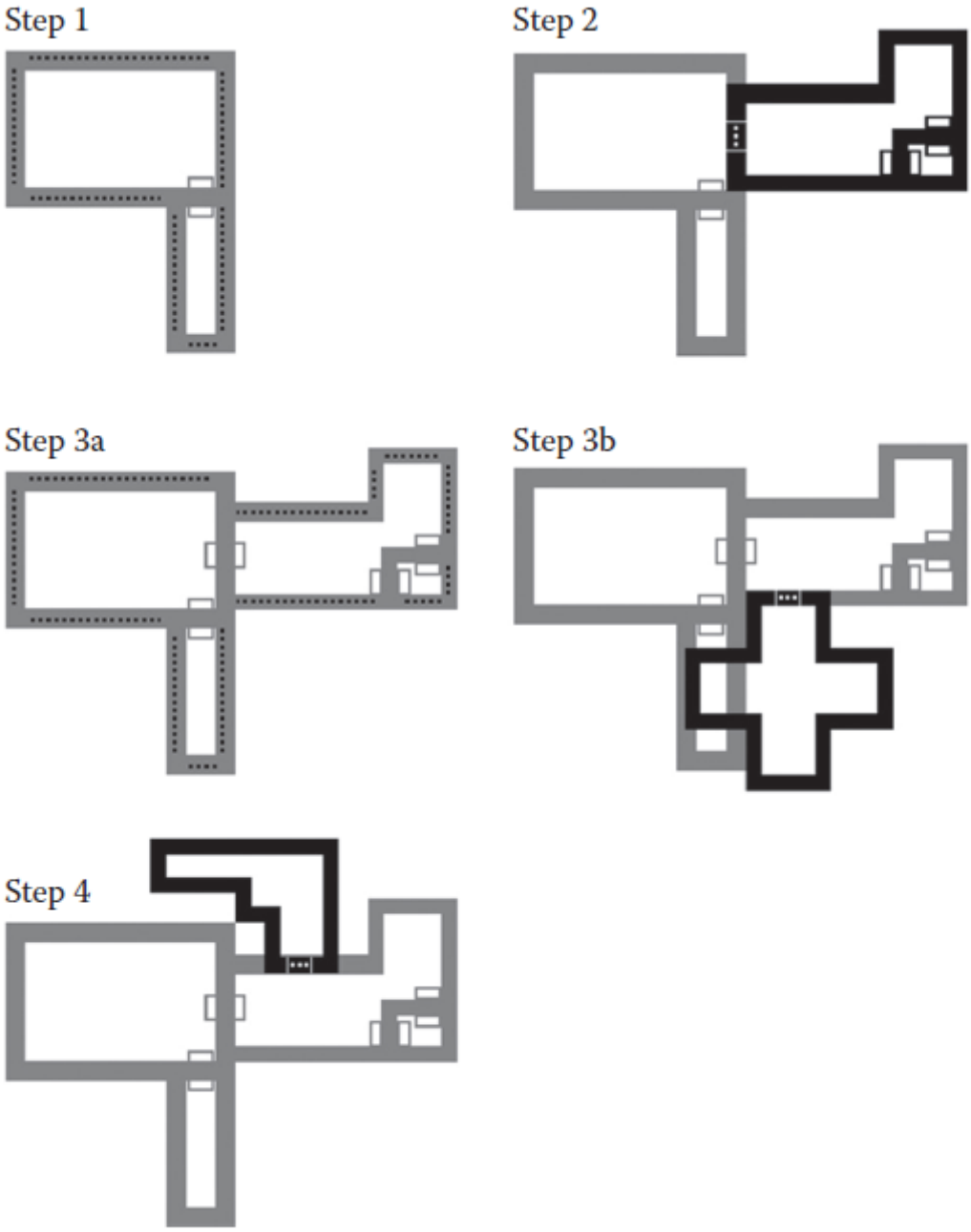

Рисунок 1.6 – процес генерації локації

Псевдокод до поданого алгоритму:

```
{
DetermineDungeonWidthAndHeight();
PlaceStarterRoom();
RoomCount = 0;
GoalCount = 20;
NumberTries = 0;
While(RoomCount < GoalCount && NumberTries < 1000)
{
```

```
NumberTries++;
//every wall tile with two adjacent wall tiles on
the same axis
//and one side exposed to unused space
RecalculateCandidates();
Candidate c = GetRandomCandidateFromJellybeanBag();
TestRoom room = GetCopyOfRandomRoomFromData();
//don' t actually place the room here, just
position it
//and check for overlap with the existing rooms
if( !TryPlaceRoomAdjacentToCandidateInOpenSpace(c,r
oom, out location) )
{
//oops we didn' t succeed
//delete the room if you aren' t in a GC
language ^^
continue;
}
ActuallyPlaceRoom(room, location);
RoomCount++;
}
//now you can fiddle with the layout by introducing doors
//laying out traps and secrets too!
        //then fill it with monsters
```

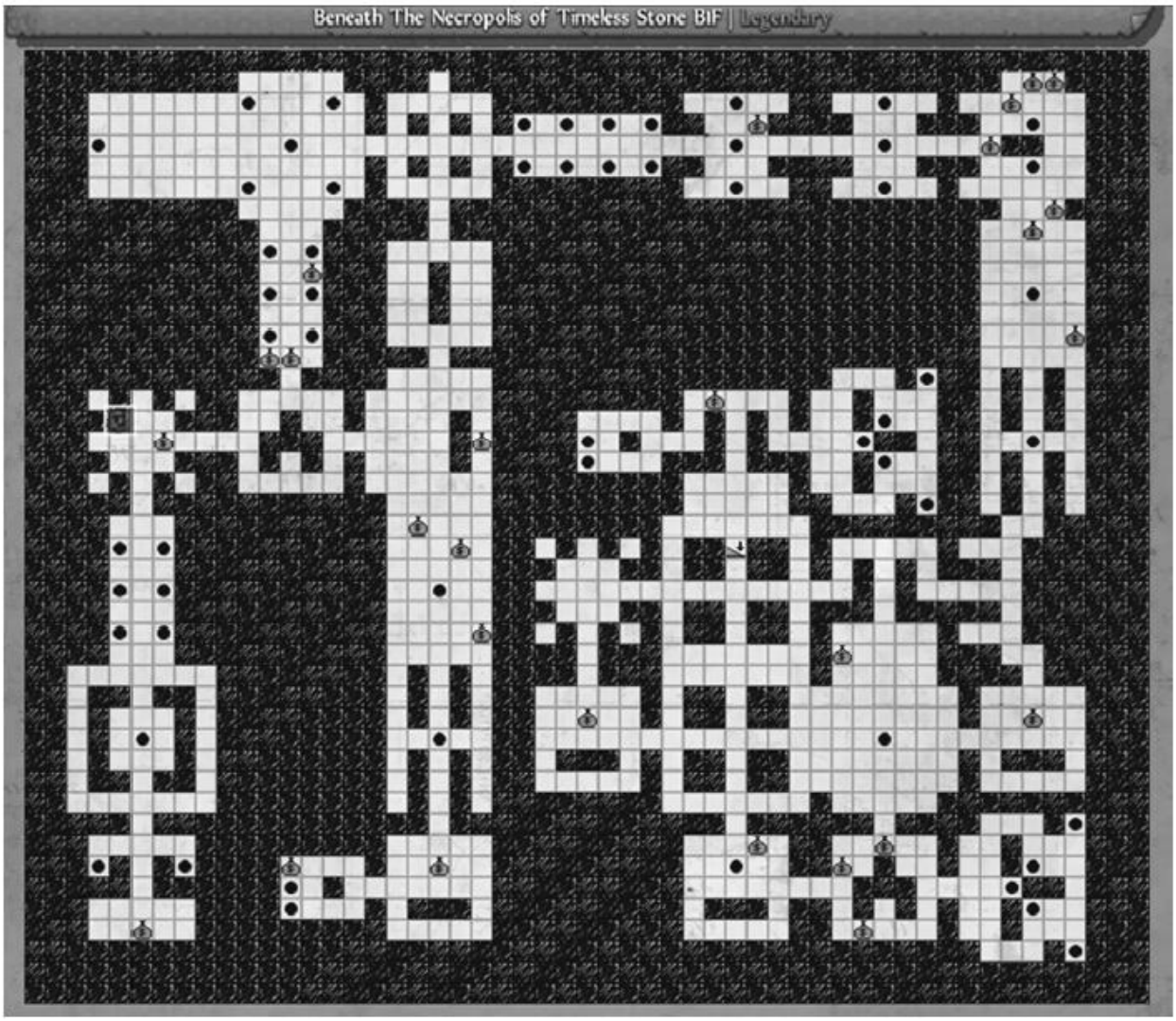

Рисунок 1.7 – результат роботи алгоритму

Ідея використання циклів для генерації рівня виникла під час дослідницького семінару про обчислювальне моделювання в іграх і була натхненена попереднім дослідження в галузі архітектури, містобудування та структури гіпертексту. У реальному світі дерева з гілками зустрічаються рідко. У більшості міст, будівель і парків, можна ходити по колу. Цикли також дуже домінують у ручній роботі проектування рівнів. Існує багато різних способів використання циклу. Обидва шляхи від може запропонувати гравцеві різні проблеми. А може, якщо один шлях набагато коротший, він може бути набагато небезпечнішим або просто набагато важче знайти. Або створення швидкого маршруту назад до початкової точки (наприклад, після того, як гравець знайшов ключ).

1.3 Приклади застосування процедурної генерації у мультимедії

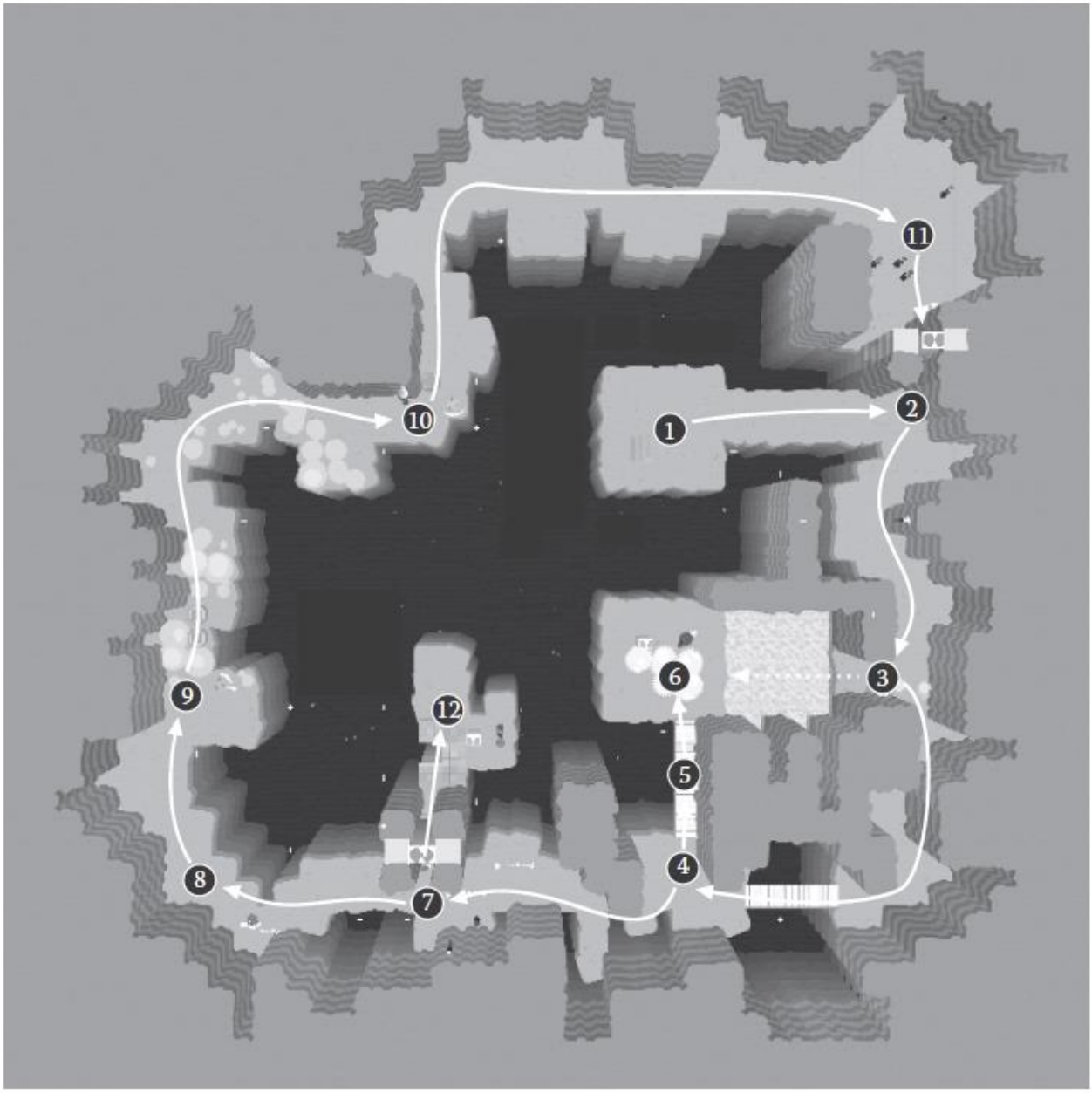

Рисунок 1.8 – Приклад процедурної генерації за допомогою графів у грі Unexplored

З циклами працювати непросто, якщо рівень, наприклад представлений плиткою.

Щоб обійти цю проблему, Unexplored використовує для представлення графи як структуру рівня на початкових етапах процесу генерації.

1) Гравець входить на рівень на платформі в центрі величезної прірви.

2) Двері на півночі закриті. Є тільки один відкритий маршрут - південний.

3) Через лаву сгенерована платформа на 6, але вона недосягаєма.

4) Гравець можете продовжити або перевірити платформу 6, яку можна було побачити раніше.

5) Коли гравець перетинає міст, він руйнується.

6) Якщо гравець добереться до платформи, то потрапляє в пастку перед гігантським щуром у якого є сувій телепортації, який дозволяє гравцю втекти.

7) також закриті двері.

8) вузький шлях охороняє патрульний кобольд-списоносець.

9) секція захищена сплячим гобліном. Гравець може легко пройти повз. Але чи помітить він ті тригери, які спрацьовують тривогу?

10) Подібний до попереднього триггер. Спусковий гачок може розбудити гобліна-лучника.

11) Гігантські мурахи захищають цю територію, але вона також має дві важелі управління дверима на 2 і 7.

12) Після відвідування 11 гравець може дістатися до цієї платформи і втекти сходами на наступний рівень.

Цей процес був натхненний практикою моделювання інженерії.

Коротше кажучи, інженерія, керована моделлю, виступає за автоматизацію процесів. Використовується кілька кроків, щоб отримати кінцевий результат. Вихід кожного окремого кроку — це модель, бажано модель, яка має певний сенс, який передається на наступний крок для подальшої обробки.

Загалом, коли створюється щось настільки складне, як завершене ігровому рівні, має великий сенс розбити процес на кілька кроки. Всі успішні генератори рівнів так роблять. однак, переваги кількох кроків значно покращуються за допомогою різних типи моделей під час процесу. Unexplored спочатку використовує графи для представляють структуру рівня, оскільки

графи набагато краще зберігати структуру інформацію на відстані. Наприклад, рисунок 4 можна зобразити, як наступний граф:

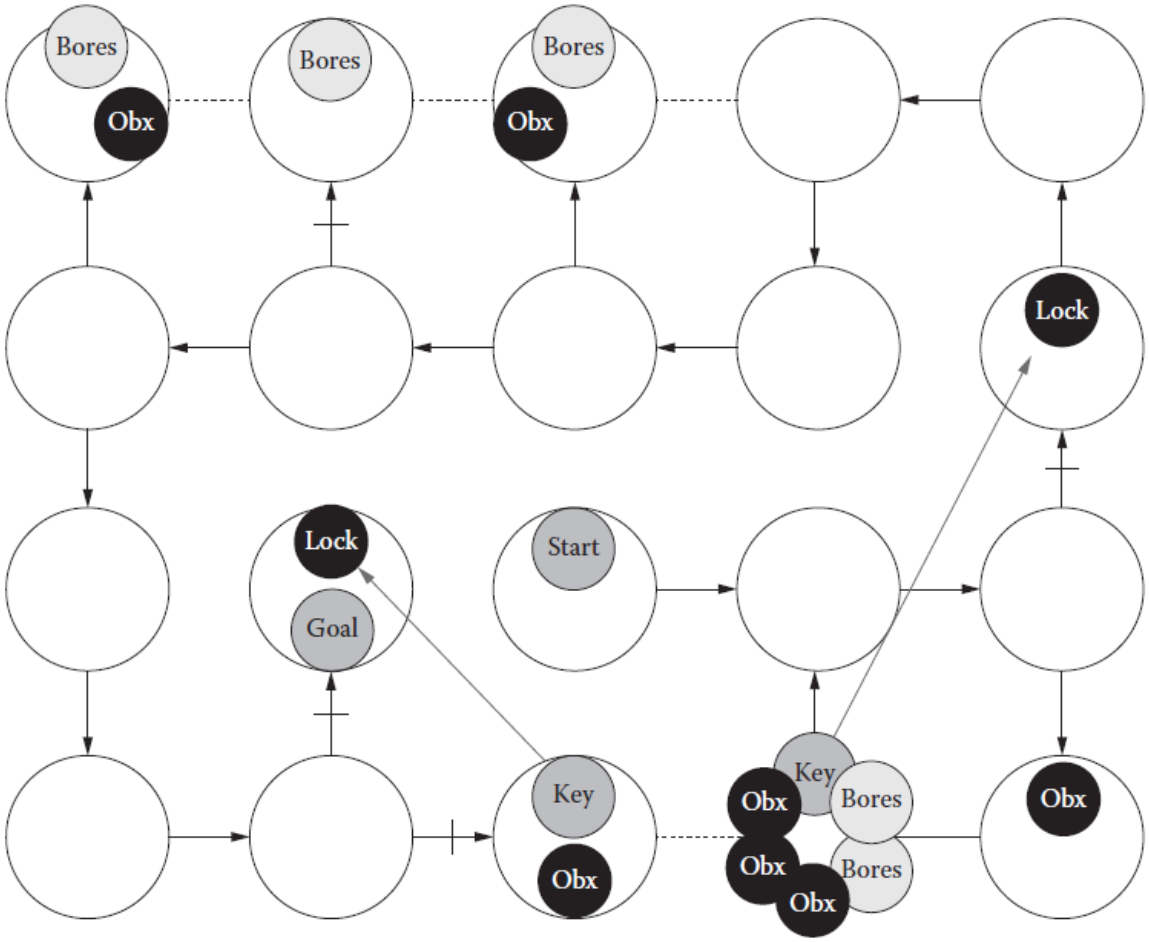

Рисунок 1.9 – Граф рівня в Unexplored

Постановка задачі

Метою дипломної роботи є розробка алгоритму процедурної генерації для створення різноманітних текстур місцевості з різними параметрами. Припускається, що сгенеровані розробленим алгоритмом текстури будуть використовуватись у мультимедійній сфері, наприклад 3D або 2D комп’ютерних іграх.

Так, побудова текстур за допомогою алгоритмів процедурної генерації спрощує задачу для розробників та художників, яким потрібно було витрачати час та ресурси, під час розробки проекту, на реалізацію текстур. Припускається, що подані алгоритми зможуть генерувати унікальні текстури місцевості на основі заздалегіть намальованого людиною зразків і будуть зображати наступні шаблони текстур:

- Ліс.
- Пісок.
- Вода.
- Каміння.
- Лава.
- Піна.
- Печерні утворення.

Так як сгенеровані текстури будуть використовуватись в комп’ютерних іграх, то алгоритм повинен виконувати роботу не ставлячи надмірний реалізм вище за швидкодію та функціональність. Таким чином в основі алгоритмів процедурної генерації повинна бути гнучка структура даних, що забезпечує оптимальний процес генерації текстур.

Щоб досягти поставленої мети необхідно: дослідити теоретичні аспекти процедурної генерації шляхом вивчення наукової літератури, проаналізувати

існуючі алгоритми, що використовують у методах процедурної генерації та знайти їх застосування у вирішені задачі дипломної роботи.

Останніми роками спостерігається динамічне зростання синтетичного медіа, до якого можна віднести: фільми, анімація, відеоігри, музика. В усіх названих видах синтетичного медіа використовуються методи процедурної генерації. Саме емпіричний характер розвитку індустрії, відсутність теоретичного обґрунтування рішень, прийнятих під час їх розробки, значно знижують якість створюваного контенту методами PCG. Складність створення контенту вимагає застосування певних алгоритмів і методів роботи над ним, головним із яких є обдумане проектування. Високоякісне проектування є ключовим моментом розробки алгоритму. Проектування сприяє ясності, чіткості та легкості при проведенні робіт із програмуванням та формуванням алгоритму в цілому.

Синтез текстур важливий для багатьох комп'ютерних програм: графіка, анімація та обробка зображень. Проте, залишається складним розробити алгоритм, який одночасно є ефективним і здатним генерувати високоякісні результати. Метою розробки алгоритму було представити ефективний алгоритм синтезу реалістичних текстур. Алгоритм нескладний використовувати і вимагає лише зразка текстури як вхідних даних. Він генерує текстури з якістю, що відповідає або кращою, ніж створена за попередніми методиками, але працює на два порядки швидше. Це дозволяє нам застосувати синтез текстур до проблем, де він є традиційно вважалося непрактичним. Зокрема, маємо застосував його до обмеженого синтезу для редагування зображень і тимчасових створення текстури. Алгоритм являється похідним від алгоритму Маркова, що буде розглянуто у роботі.

Текстура поля моделює та генерує текстури за допомогою детермінованого процесу пошуку. Цей процес синтезу прискорюється за допомогою деревоструктурного векторного квантування.

Висновок до розділу 1

Підсумовуючи подане у розділу 1 можна зробити наступні висновки:

1. Методи процедурної генерації (PCG) – це методи створення даних за допомогою потужностей електронно-обчислювальних машин. Зазвичай ці методи протиставлять ручному способу створення контенту, тобто за участі художників, дизайнерів, аніматорів, тощо. Зазвичай використовують поєднання вручну створених асетів та змогу ЕВМ генерувати випадкові значення. Методи PCG використовують у комп'ютерній графіці для створення 3D моделей та текстур. У мультимедії їх використовують, щоб уникнути багатозатратної з фінансового та часового плану роботи художників та аніматорів;
2. методи процедурної генерації існують вже давно та переважно використовуються у синтетичних медіа, таких як: анімація, синтезована музика, текстуризація, відеоігри. Та більш традиційних медіа, до яких можна віднести фільми та мультфільми;
3. так, побудова текстур за допомогою алгоритмів процедурної генерації спрощує задачу для розробників та художників, яким потрібно було витрачати час та ресурси, під час розробки проекту, на реалізацію текстур.

2 Методи процедурної генерації

2.1 Алгоритм Форчуна

За допомогою випадкового поля Маркова багато алгоритмів моделюють текстури (або в іншій математичній формі, вибірки Гіббса) і генерують текстури за допомогою випадкової вибірки. Оскільки випадкові поля Маркова довели, що є хорошим наближенням для широкого діапазону текстур, ці алгоритми є загальними, і деякі з них дають хороші результати.

Алгоритм Фортуна відноситься до типу алгоритмів лінійної розгортки (SweepLineAlgorithms). Ідея алгоритмів такого типу полягає в тому, щоб ввести пряму лінію, яка буде рухатись по плоскості, зупиняючись у певних точках. Всі геометричні операції такого алгоритму будуть виконуватися лише над тими об'єктами, які або перетинають пряму, або знаходяться в її непосередній близькості, під час її зупинки. Рішення буде досягнуто, коли пряма пройде по всім об'єктам.

Під час виконання алгоритму Форчуна, пряма буде рухатися зверху вниз по плоскості, на якій точки розташовані випадково, які в останній момент стануть вузлами діаграми Вороного.

Для початку, якщо взяти будь-який вузол P і пряму l (не містить точку P), то кількість таких точок, розташування яких до P буде дорівнювати відстані до l та утворювати параболу.

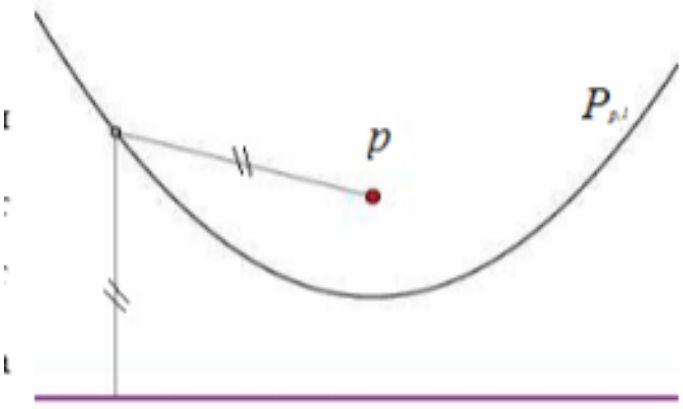

Рисунок 2.10 – Утворена парабола

Цю параболу ми позначимо як$P_{P,l}$, і будемо вважати, що вона розділяє плоскость на дві області: одна складається з точок, які ближче до вузлу P, а друга з точок, які ближче до прямої l.

Це дуже важливий момент, так що розглянемо його детальніше. Розглянемо точку q з координатами $q = (q_x, q_y)$, і з відстанню до вузла P рівним $d(q,p)$. Горизонтальна Пряма l є Замітаючою Прямою. Позначимо її вертикальну координату як l_y.

Таким чином, відстань між точкою q і прямою l рівно $q - l$, а умова того, що точка q лежить на параболі $P_{p,l}$, можна написати як

$$d(q,p) = q_y - \ l_y$$

Отже,

$$d(q,p) < q_y - \ l_y, \text{якщо } q \text{ лежить над } P_{p,l}$$
$$d(q,p) = q_y - \ l_y, \text{якщо } q \text{ лежить на } P_{p,l}$$
$$d(q,p) = q_y - \ l_y, \text{якщо } q \text{ лежить під } P_{p,l}$$

Так, по ходу виконання алгоритму, замітаюча Пряма буде рухатися по плоскості зверху вниз. У кожен момент часу ми будемо розглядати вузли, які лежать над прямою та утворюванними параболами. Наприклад, враховуючи безліч вузлів і положень прямої, зображених на рисунку 2.9, ми будемо розглядати параболи на рисунку 2.10.

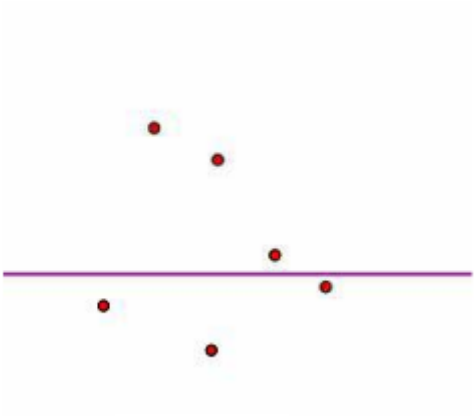

Рисунок 2.11 – Замітаюча пряма

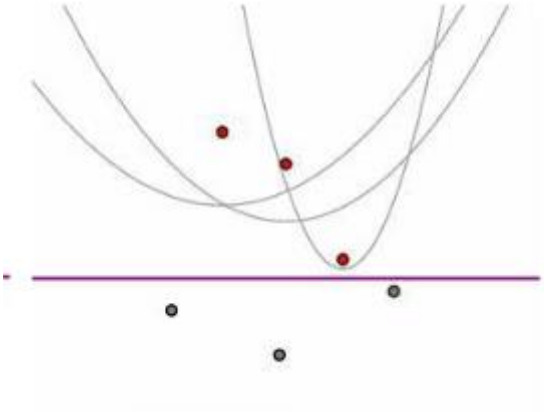

Рисунок 2.12 – Параболічні утворення

Крива (BeachLine). Ця крива складається з нижньо параболічних дуг, кожна з яких відповідає одному з вузлів, що знаходяться над Замітаючою прямою. На рисунку 11 Параболічна Крива виділена зеленим кольором.

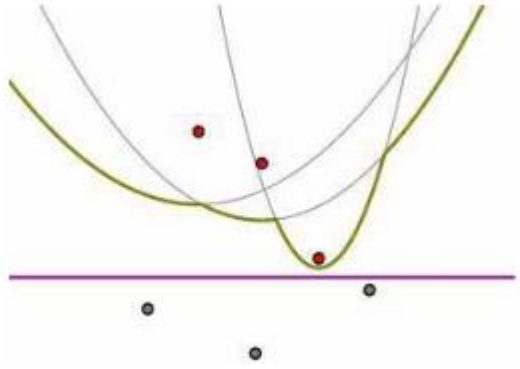

Рисунок 2.13– Параболічна пряма

Параболічна Крива дуже зручна для конструювання мозаїк Вороного. Наприклад, якщо дана точка лежить над кривою, то вона знаходиться ближче до одного з вузлів діаграм над прямою, ніж сама пряма. Це означає, що дана точка лежить всередині одного із плиток мозаїки, через яку вже пройшла помічаюча пряма.

Визначимо тепер, в який момент крива проходить через фіксовану точку. Припустимо, що відстань від точки q до вузла P_l не більше, чим її відстань до будь-якого syijuj узла, отже

$$d(q,p_1) \leq d(q,p_i), \forall i.$$

Умова те, що лежить на параболі виражається так

$$d(q,p) = q_y - l_y$$

А отже:

$$d(q,p) > d(q_i - l_l) = q_y - l_y$$

Це означає, що, коли точка q потрапляє на параболу, вона не може перебувати над якоюсь іншою параболою $P_{pi,l}$. Отже, коли q виявляється на параболі $P_{pi,l}$, вона опиняється на параболічній кривій, а саме, на одній із складових її параболічних дуг, кожна з яких відповідає одному з вузлів діаграми. Таким чином, застосовуючи попереднє міркування, можна зробити висновок, що граничні точки знаходяться на однаковій відстані від двох найближчих вузлів діаграми.

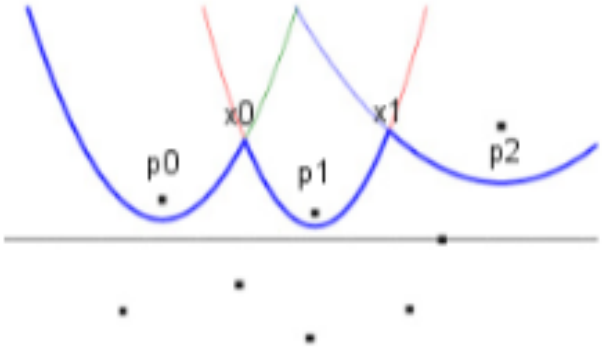

Рисунок 2.14 – Граничні точки

$$d(x_0,p_0) = d(x_0,p_1)$$
$$d(x_1,p_1) = d(x_1,p_2)$$

Таким чином, граничні точки лежать на Гранях мозаїки Вороного, і за їх допомогою, по ходу виконання алгоритму Форуна можна буде позначити межі полігонів.

2.2 Мозаїка Вороного

Діаграма Вороного - це спосіб поділу простору на набір областей (які називається клітинами, cells), заданих набору вхідних точок (які називаються точками, sites), таким чином, що кожна клітина містить рівно 1 ділянку, і точки всередині комірки це саме ті, чия найближча ділянка знаходиться всередині цієї комірки.

Діаграми Вороного мають надзвичайно широкий спектр застосування. Їх можна використовувати для створення текстур для води, створення геометрії для кам'янистого ґрунту, ефективного обчислення найближчого сусіда точки або для керування переміщенням об'єктів AI.

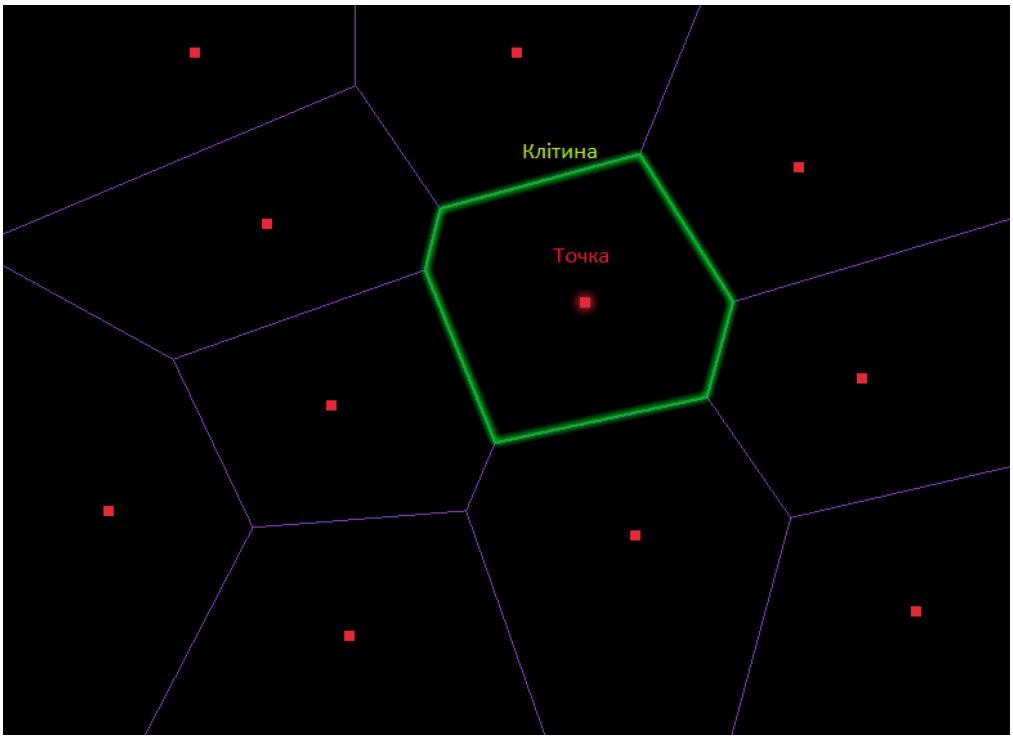

Рисунок 2.15 – Мозаїка Вороного і її основні поняття: клітина та точка

Якщо існує відносно невелика кількість сайтів, то можна просто перерахувати кожен сайт у заданому просторі та обчислити, який сайт є найближчим. Це працює, якщо важливо лише те, до якого сайту належить кожна точка простору, і є кінцева кількість дискретних точок, наприклад, якщо просто потрібно визначити колір для кожного пікселя зображення.

З іншого боку, якщо задана дуже велика кількість вхідних точок, або треба знати де проходять межі між клітинами, потрібен інший підхід. Існує кілька способів обчислення діаграми Вороного для набору вхідних точок, але самий популярний це алгоритм Форчуна.

Мозаїка Вороного має наступні властивості:

- Грані Мозаїки: кожна точка, що знаходиться на межі діаграми Вороного, рівновіддалена від двох найближчих вузлів P_i, P_j. Таким чином, можна намалювати коло з центром у цій точці, так, що вузли виявляться на межі цього кола і жоден інший вузол не виявиться всередині цього кола

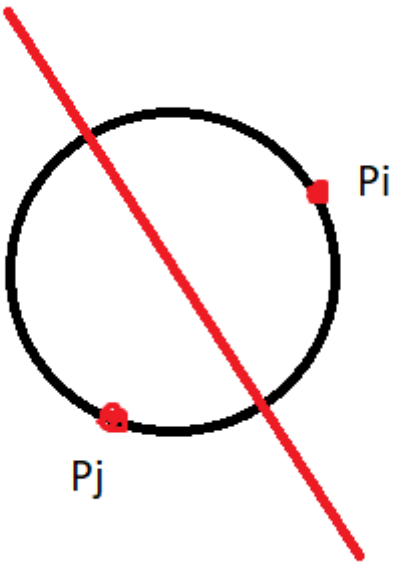

Рисунок 2.16 – Грані мозаїки

- Вершини Мозаїки: точка, в якій перетинаються три полігона діаграми, що відповідають вузлам P_k, P_i, P_j , називається вершиною мозаїки. Така точка також буде рівновіддалена від цих трьох вузлів

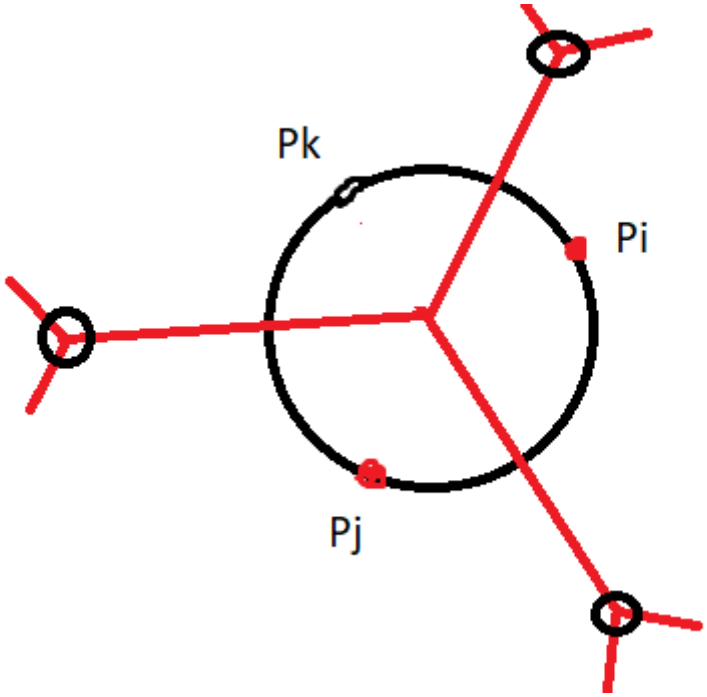

Рисунок 2.17 – Вершини мозаїки

Можна представити діаграму Вороного як неорієнтований граф. Його вершини є перетином трьох або більше клітин, а грані є перетином 2 клітин. Можна додати ще 1 «віртуальну» вершину і з'єднати з нею всі нескінченні ребра, щоб отримати стандартний граф.

Нехай у нас є множина точок P на площині. На них можна створити багато тріангуляцій. Тріангуляція Делоне для множини P точок на площині — це тріангуляція $DT(P)$, така, що жодна точка P не знаходиться всередині описаного кола будь-якого трикутника в $DT(P)$. Можна показати, що тріангуляція Делоне є подвійним графіком до діаграми Вороного (як графік) для будь-якої множини P сайтів.

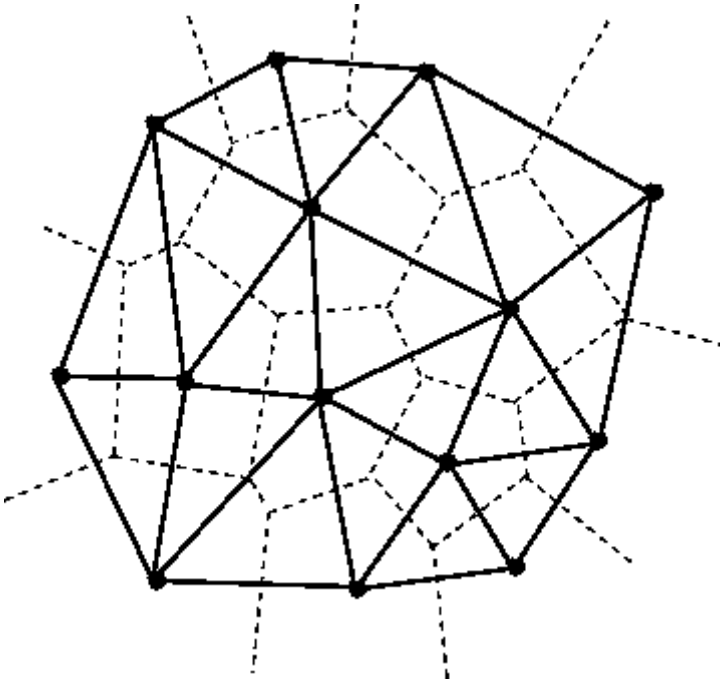

Рисунок 2.18 – Мозаїка Вороного представлена у вигляді графа за допомогою тріангуляції Делоне

2.3 Синтез текстур методом k-найближчих сусідів та клітинних автоматів

Текстура – це повсякденний візуальний досвід. Вона може описати широкі та різноманітні характеристики поверхні, такі як рельєф, рослини, мінерали, хутро і шкіра. Оскільки відтворення візуального реалізму фізичного світу є основною метою комп'ютерної графіки, текстури зазвичай використовуються при відтворенні синтетичних зображень. Ці текстури можна отримати з різних джерел, наприклад, намальовані від руки або відскановані фотографії. Намальовані від руки малюнки можуть бути естетичними, але їх важко зробити фотореалістичними. Більшість відсканованих зображення, однак, мають недостатній розмір і можуть призвести до видимих швів або повторення, якщо вони безпосередньо використовуються для відображення текстури.

Синтез текстур – це альтернативний спосіб створення текстур. Тому що синтетичні текстури можна зробити будь-якого розміру, уникнути візуального повторення. Синтез текстур також може створювати розкладні зображення та правильно обробляти граничні умови. Потенційні застосування синтезу текстур також широкі, наприклад: зменшення шуму зображення, заповнення оклюзії та стиснення. Мету синтезу текстур можна сформулювати так: дано зразок текстури, синтезувати нову текстуру, яка, для спостерігача – людини буде схожа на початкову текстуру.

Основними проблемами є

1) моделювання – як оцінити стохастичний процес із заданого кінцевого зразка текстури;

2) відбір проб – як розробити ефективну процедуру відбору проб та створювати нові текстури з заданої моделі.

І моделювання, і відбір необхідні для успіху синтезу текстури. Візуальна точність створених текстур залежатиме насамперед через точність моделювання, при цьому ефективність вибірки процедури безпосередньо визначить обчислювальну вартість сгенерованної текстури.

Основними перевагами цього алгоритму є якість і швидкість: якість синтезованих текстур або краща за ту, яка генерується попередніми методами, а швидкість обчислень на два порядки швидше. Це дозволяє використати алгоритм в області, де традиційно генерація текстури була занадто дорогою.

Клітинний автомат (КА) — дискретна математична модель, яка визначає сукупність та описується набором клітинок, що утворюють періодичну решітку, та заданими правилами переходу, що визначають стан клітини за теперішнім станом самої клітинки та тих її сусідів, що знаходяться від неї на певній відстані, яка не перевищує максимальну.

Основний напрям дослідження клітинних автоматів — алгоритмічна розв'язність окремих задач. Також розглядаються питання побудови початкових станів, при яких клітинний автомат вирішуватиме задану задачу.

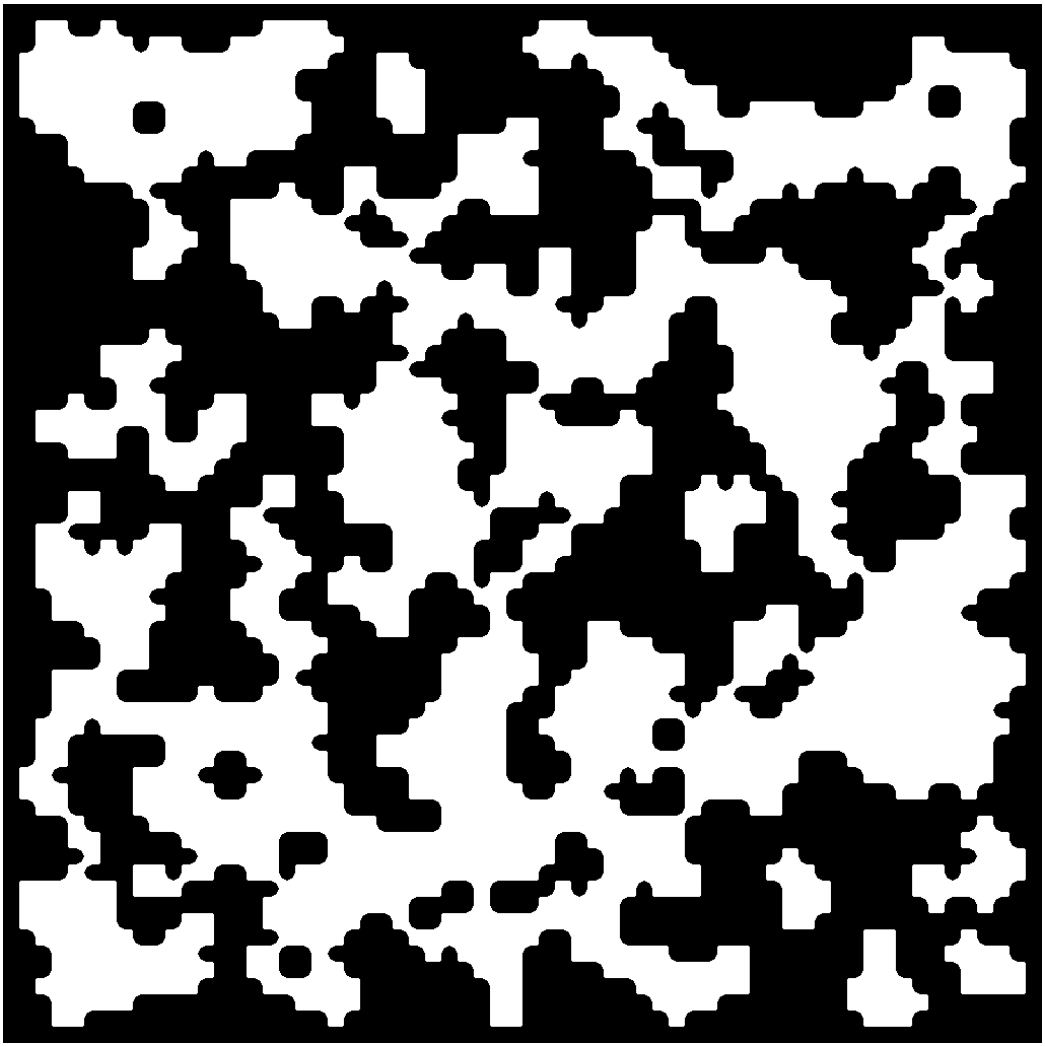

Рисунок 2.19 – Згенерована текстура методом клітинних автоматів

Поняття клітинних автоматів доволі обширне, тому можна знайти доволі багато різних визначень. Найпоширенішими є:

- математичний об'єкт із дискретним простором та часом;
- регулярна структура двійкових скінченних автоматів з однаковими правилами переходів, що виражені у вигляді булевих функцій від станів сусідніх автоматів;
- стилізовані, синтетичні світи, що визначені простими правилами, подібно правилам настільної гри;
- математична ідеалізація фізичної системи, в якій час та простір дискретні, а фізичні величини приймають скінченну множину значень.
- візуальна модель динамічної системи з дискретним часом та простором

Класичні КА в загальному випадку відповідають наступним критеріям:

- зміна значень всіх клітинок відбуваються одночасно після обчислення нового стану кожної клітинки решітки. Інакше порядок перебору клітин решітки при проходженні ітеративного процесу суттєво впливав би на результат;
- решітка однорідна. Неможливо відрізнити жодні два місця на решітці по ландшафту. Однак на практиці решітка виявляється кінцевою множиною клітин (адже неможливо виділити необмежений об'єм даних). В результаті можуть мати місце крайові ефекти: клітини, що стоять на межах решітки будуть відрізнятися за кількістю сусідів. Щоб уникнути цього можна ввести періодичні крайові умови;
- взаємодії локальні. Лише околишні клітинки (як правило, сусідні) здатні вплинути на дану клітинку;
- множина станів клітинки кінцева. Ця умова потрібна, щоб для отримання нового значення стану клітини треба було виконати кінцеву кількість операцій (але це не заважає використовувати клітини для зберігання чисел із плаваючою комою для розв'язку прикладних задач).

Якщо з будь-якого початкового стану можна привести клітинного автомат в будь-яку задану конфігурацію шляхом варіювання значення загального вхідного параметра, такий КА називають повним.

У кожний момент часу кожен елемент КА приймає один стан зі скінченного набору станів. У залежності від цих станів в наступний момент часу набір елементів може прийняти новий стан. Якщо для елементів КА множини можливих станів відрізняються, такий клітинний автомат називається полігенним. Але на практиці використовуються комірки з еквівалентною множинам можливих станів алгебраїчною структурою — лінійні КА.

Елементи можуть бути геометрично розташовані різноманітним чином. Розмірність простору може бути довільною, а число елементів — як безкінечним, так і скінченним. В останньому випадку виникає додаткова міра свободи в граничних умовах. Вони можуть бути різними, але на практиці використовуються постійні у часі або періодичні граничних умовах. У динамічних КА геометрія може змінюватися з часом, а якщо геометрія різна на різних ділянках простору, такі клітинні називають неоднорідними.

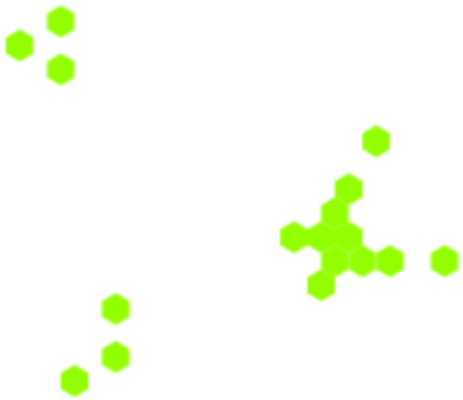

Рисунок 2.20 – Клітинний автомат із комірками, що зображені у вигляді шестикутника

Сусіди — це елементи, від яких залежить елемент КА. Можна назвати поняття сусідства ключовим для КА. При тому сусідство розуміється не в геометричному сенсі, а в інформаційному. Хоча зазвичай інформаційний сенс накладається на геометричний. Сусідство одиничних автоматів встановлюється

постійним для кожного одиничного автомата решітки і визначається спеціальним вектором — індексом сусідства. Як правило, розглядаються *d*-мірні регулярні решітки, в цілочислові точки яких поміщені копії деякого автомата Мура. Стан елемента в наступний момент часу обчислюється зі стану самого елементу і його сусідів. Сусідство більшою мірою визначається геометрією КА. Для різних цілей можлива зміна числа вхідних станів елемента. Якщо для кожного елемента КА число входів і виходів однакове, такий КА називається збалансованим.

Висновки до розділу 2

У розділі були розглянуті відомі методи процедурної генерації. Були проаналізовані можливі варіанти, що використовуються у структурі розробленого алгоритму для синтезу 2D текстур. Розглянені переваги та недоліки кожного з методів.

Отримана база знань під час дослідницької роботи допоможе у подальшій розробці алгоритму, визначені основні напрями та методи розробки подальшого алгоритму.

Основними проблемами є

1) моделювання – як оцінити стохастичний процес із заданого кінцевого зразка текстури;

2) відбір проб – як розробити ефективну процедуру відбору проб та створювати нові текстури з заданої моделі.

І моделювання, і відбір необхідні для успіху синтезу текстури. Візуальна точність створених текстур залежатиме насамперед через точність моделювання, при цьому ефективність вибірки процедури безпосередньо визначить обчислювальну вартість сгенерованної текстури.

Основними перевагами цього алгоритму є якість і швидкість: якість синтезованих текстур або краща за ту, яка генерується попередніми методами, а швидкість обчислень на два порядки швидше. Це дозволяє використати алгоритм в області, де традиційно генерація текстури була занадто дорогою.

3 Моделювання та проектування інформаційної системи

3.1 Структура системи

Було запропоновано безліч підходів для аналізу текстур та їх синтезу, Далі буде коротко розглянуто репрезентативні роботи що лягли в основу розробленого алгоритму.

Фізичне моделювання: можна синтезувати певні текстури поверхні шляхом безпосереднього моделювання їх фізичних процесів генерації. Можуть бути такі біологічні моделі, як хутро, лусочки та шкіра, що моделюється з використанням реакційної дифузії та клітинних автоматів.

Деякі явища вивітрювання та мінералів можна достовірно відтворити шляхом детального моделювання. Ці методи можуть виробляти текстури безпосередньо на тривимірних сітках, тому можна уникнути проблеми викривлення текстурного відображення. Однак різні текстури зазвичай генеруються дуже різними фізичними процесами, тому такі підходи застосовуються лише до обмежених класів текстур.

Випадкове поле Маркова та вибірка Гіббса: багато алгоритмів моделюють текстури за допомогою випадкових полів Маркова (або в іншій математичній формі, вибірки Гіббса) і генерують текстури за допомогою випадкової вибірки. Оскільки випадкові поля Маркова були доведені, як хороший метод для генерації широкого діапазону текстур, ці алгоритми є загальними, і деякі з них дають хороші результати. Недоліки вибірки випадкового поля Маркова вбачаються у тому, що це обчислювально дорого: навіть невелика текстура вимагає тривалої генерації.

Узгодження функцій: деякі алгоритми моделюють текстури як набір функції та створюють нові зображення, зіставляючи функції разом. Ці алгоритми зазвичай більш ефективні, ніж алгоритми випадкового поля Маркова. Алгоритм Хігер і Берген моделюють текстури шляхом збігу граничних гістограм пірамід зображень. Їхня техніка успішна на дуже

стохастичних текстурах, але не вдається на більш структурованих. Де Боне синтезує нові зображення шляхом рандомізації вхідного зразка текстури, зберігаючи міжмасштабні залежності. Цей метод працює краще, ніж алгоритми пірамід структурованих текстур, але він може створювати межі артефактів, якщо вхідна текстура не розкладається. Алгоритм Сімончеллі та Портілья генерують текстури, узгоджуючи спільну статистику пірамід зображень. Їхній метод може успішно захопити глобальні текстурні структури, але зазнає невдачі при збережені початкового виду текстури.

Метою БКР було розробити алгоритм, який поєднує переваги попередніх підходів, щоб він був ефективним, загальним і здатний виробляти високоякісні текстури, які можна пікселізувати. Це також повиннен бути зручний для користувача алгоритм, тобто такий, кількість настроюваних вхідних параметрів якого повинна бути мінімальною. Цього можна досягти шляхом ретельного вибору процедури моделювання та синтезу текстури. Для текстурної моделі використовуються випадкові поля Маркова, оскільки було доведено, що вони охоплюють найширший спектр корисних типів текстур. Щоб уникнути звичайних обчислювальних витрат MRF, була розроблена процедура синтезу текстур, яка дозволяє уникнути явної побудови ймовірності та відбір проб.

Методи випадкового поля Маркова моделюють текстуру як реалізацію локального та стаціонарного випадкового процесу. Тобто кожен піксель текстури зображення характеризується невеликим набором просторово сусідніх пікселів, і ця характеристика однакова для всіх пікселів.

Наприклад: уявимо, що глядачу дається зображення, але дозволено спостерігати за ним лише через маленьке рухоме віконце. Коли вікно переміщується, глядач може спостерігати різні частини зображення. Зображення є нерухомим, якщо при належному розмірі вікна спостережувана частина завжди виглядає схожою. Зображення локальне, якщо кожен піксель генеруеться передбачувано з невеликого набору сусідніх пікселів і не залежить від решти зображення.

На основі цих припущень про локальність і стаціонарність наш алгоритм синтезує нову текстуру, щоб вона локально була схожа на початкову текстуру.

Нова текстура генерується піксель за пікселем, і кожен піксель визначається так, щоб зберігалася локальна подібність між текстурою прикладу та результатом зображення. Цей синтез, на відміну від більшості алгоритмів, заснованих на MRF, повністю детермінований і не створюється явний розподіл ймовірностей. Як в результаті, він ефективний і піддається подальшому прискоренню.

Використання випадкових полів Маркова як моделі текстури, мета розробленого алгоритму синтезу полягає у генерації нової текстури, щоб кожна локальна область була подібна до іншої області початкової текстури.

Спочатку буде описано, як працює алгоритм в одній роздільній здатності, а потім ми розширюємо його, використовуючи піраміду з багатьма роздільними здатностями, щоб отримати покращення ефективності. Для зручності наведемо список символів у таблиці 3.2

Таблиця 3.2 – умовні позначення

Символ	Значення
I_a	Початкова текстура
I_s	Отримана текстура
G_a	Гаусова піраміда утворена з I_a
G_s	Гаусова піраміда утворена зI_s
p_i	Початковий піксель в I_a *або* G_a
p	Отриманий піксель в I_a *або* G_a
$N(p)$	Сусідні пікселі до p, сусідство
$G(L)$	L-рівень піраміди гаусса

Алгоритм починає роботу з вводу початкової текстури I_a, з якої буде процедурно сгенерована нова, створюється пусте зображення із білим шумом I_s з розмірністю як у I_a та змушуємо цей шум виглядати як початкова текстура у порядку растрового сканування зверху до низу, зліва направо. Щоб визначити значення пікселю p у I_s, значення вже визначеного сусіда $N(p)$ порівнюється між усіма можливими сусідами $N(p_i)$ з I_a. Так, початковий піксель p_i з найбільш підходящим $N(p_i)$ визначається як p. Для порівняння схожості між сусідами використовується сума квадратних різниць. Метою цього процесу є забезпечення локальної схожості між I_a та I_s у p. Цей процес повторюється для кожного вихідного пікселя доки не буде визначено кожен піксель. Наприклад у рисунках 3.21 – 3.23 зображено процес синтезу текстури.

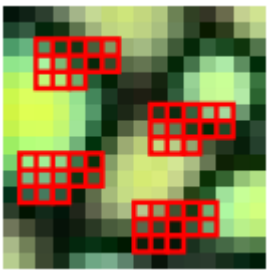

Рисунок 3.21 – Початкова текстура I_a

Рисунок 3.22 – Поняття сусідів p, каретка поточного пікселя

Рисунок 3.23 – Процес синтезу

Оскільки набір локальних сусідів $N(p_i)$ використовується як основна модель для текстур, якість сгенерованих результатів буде залежати від його розміру та форми. Інтуїтивно, розмір сусідів повинен бути в масштабі найбільшої структури регулярної текстури, інакше ця структура може бути втрачена, і зображення результату буде виглядати занадто випадково. Рисунок 3.24 демонструє ефект розміру сусідства на результати синтезу, де послідовно збільшувалось значення сусідства (5x5, 7x7, 9x9).

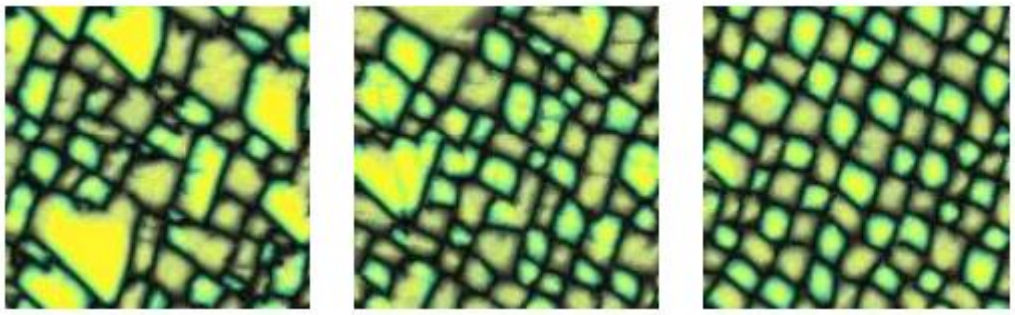

Рисунок 3.24 – Сгенеровані текстури з різними розмірами сусідства.

Алгоритм єдиної роздільної здатності фіксує структури текстури використанням сусідств належного розміру. Однак для текстур, що містять великомасштабні структури, ми повинні використовувати великі сусідства, а великі сусідства вимагають більше обчислень. Цю проблему можна розв’язати за допомогою піраміди зображень із багато-роздільною здатністю. Швидкість обчислення збережено, оскільки можна представляти великомасштабні структури компактніше на кілька пікселів на певному рівні піраміди з нижчою роздільною здатністю.

Алгоритм синтезу з багато-роздільною здатністю працює наступним чином. Дві піраміди Гауса, G_a та G_s, спочатку будуються з I_a та I_s відповідно. Потім алгоритм перетворює G_s з нижчого до вищої роздільної здатності, так що кожен вищий рівень роздільної здатності будується з уже синтезованих нижчих рівнів роздільної здатності. Це подібна схема до послідовності, в якій малюється картина: спочатку наносяться товсті штрихи, а потім додаються деталі. Всередині кожного вихідного рівня піраміди G_s(L) пікселі синтезуються шляхом подібним до випадку з єдиною роздільною здатністю, коли пікселі призначаються в порядку растрового сканування.

Єдина модифікація полягає в тому, що для випадку з багато-роздільною здатністю, кожне сусідство $N(p)$ містить пікселі поточної роздільної здатності, а також такі, що мають меншу роздільну здатність. Подібність між двома сусідствами з багато-роздільною здатністю вимірюється шляхом обчислення суми квадратів відстані всіх пікселів всередині їх. Ці пікселі з нижчою роздільною здатністю обмежують процес синтезу так що додані деталі високої частоти відповідали б вже синтезованим низькочастотним структурам.

Природні текстури часто містять впізнавані структури, а також певну степінь випадковості. Оскільки мета алгоритму полягає у відтворенні реалістичних текстур, важливо, щоб алгоритм фіксував випадкові аспект текстур. Це поняття випадковості іноді може може бути досягнуто шляхом максимізації ентропії, але обчислювальна вартість вважається непомірною.

Замість цього ми ініціалізуємо вихідне зображення I_s як білий випадковий шум і поступово змінюємо цей шум, щоб він виглядав як вхідна текстура I_a. Цей крок ініціалізації заповнює алгоритм з достатньою ентропією, і дозволяє решті процесу синтезу зосередитися на перетворенні (I_s до I_a). Щоб зробити цей випадковий шум кращим початковим припущенням, ми також вирівнюємо пірамідну гістограму G_s щодо G_a.

Початковий шум впливає на процес синтезу наступним чином. Для випадку з одним рішенням сусідства в перших кількох рядка і стовпцях I_s містять пікселі шуму. Ці пікселі шуму вносять невизначеність у процес формування сусідств, через що граничні пікселі призначаються

напівстохастично (Однак процес пошуку все ще детермінований. Випадковість визначається початковим шумом). Решта пікселів шуму перезаписуються безпосередньо під час синтезу. Однак для випадку з багатьма роздільною здатністю більша частина пікселів шуму сприяє процесу синтезу, принаймні опосередковано, оскільки вони визначають початкове значення найнижчого рівнч роздільної здатності G_S.

Сумуючи усе вище написане можна сформувати псевдокод майбутнього алгоритму.

function $I_s \leftarrow$ TextureSynthesis(I_a, $outputSize$)
1 $I_s \leftarrow$ Initialize($outputSize$);
2 $G_a \leftarrow$ BuildPyramid(I_a);
3 $G_s \leftarrow$ BuildPyramid(I_s);
4 **foreach** level L from lower to higher resolutions of G_s
5 **loop** through all pixels (x_s, y_s) of $G_s(L)$
6 $C \leftarrow$ FindBestMatch(G_a, G_s, L, x_s, y_s);
7 $G_s(L, x_s, y_s) \leftarrow C$;
8 $I_s \leftarrow$ ReconPyramid(G_s);
9 **return** I_s;

function $C \leftarrow$ FindBestMatch(G_a, G_s, L, x_s, y_s)
1 $N_s \leftarrow$ BuildNeighborhood(G_s, L, x_s, y_s);
2 $N_a^{best} \leftarrow$ null; $C \leftarrow$ null;
3 **loop** through all pixels (x_a, y_a) of $G_a(L)$
4 $N_a \leftarrow$ BuildNeighborhood(G_a, L, x_a, y_a);
5 **if** Match(N_a, N_s) > Match(N_a^{best}, N_s)
6 $N_a^{best} \leftarrow N_a$; $C \leftarrow G_a(L, x_a, y_a)$;
7 **return** C;

Рисунок 3.25 – Псевдокод алгоритму

Архітектура цього алгоритму гнучка; вона складається з кількох ортогональних компонентів. Нижче будуть перераховані ці компоненти.

Pyramid: піраміди будуються та реконструюються до зображення за допомогою стандартних процедур BuildPyramid та ReconPyramid. Для синтезу текстури можна використовувати різні піраміди; прикладами є піраміди Гауса, піраміди Лапласа, керовані піраміди та піраміди на основі ознак. Гаусова піраміда, наприклад, будується шляхом послідовної фільтрації і операції зниження дискретизації, і кожен рівень піраміди, за винятком найвищої роздільної здатності, є розмитою та розбитою версією оригінального

зображення. Реконструкція пірамід Гаусса тривіальна, оскільки зображення доступне на рівні піраміди з найвищою роздільною здатністю. Ці різні піраміди дають різні компроміси між просторовими і частотними слоями. У розробленому алгоритмі було вирішено використовувати Піраміду Гауса за її простоту та більшу просторову локалізацію.

Neighborhood, сусідство: сусідства можуть мати довільний розмір і форму, єдина вимога полягає в тому, щоб вони містили лише допустимі пікселі.

Упорядкування синтезу синтезу: у рядку 5 псевдокоду використовується упорядкування растрового сканування функція TextureSynthesis(). Однак це також можна розвинути. Наприклад, спіральне впорядкування можна використовувати для обмеженої текстури. Функція упорядкування синтезу повинна співпрацювати з BuildNeighborhood, щоб сусідства містили лише дійсні пікселі.

Searching, пошук: процедура пошуку FindBestMatch() використовується для визначення вихідних значень пікселів. Так як це стандартний процес, то можна використовувати різні алгоритми пошуку точок для прискорення роботи алгоритму.

3.2 Розробка алгоритму

Для розробки програмної реалізації була обрана об’єктно орієнтована мова програмування C#. Такий вибір був зроблений із урахуванням подальшої інтеграції модуля алгоритму у проекти, де необхідна вбудована генерація текстур. Наприклад, мова C# використовується в графічних редакторах, OpenGL та середи розробки комп’ютерних ігор Unity. У якості середи розробки був обраний Visual Studio.

Знання щодо мови розробки C# здобувались протягом навчання та з: наукових статтей, веб-ресурсів, книги «Мова програмування C# 7 і платформи .NET та .NET Core», та офіційної документації на сайті Microsoft.

Алгоритм використовує безкоштовні зразки текстур, у якості вхідної текстури, що були взяті з мережі.

Для налаштування параметрів алгоритму використовується файл samples.xlm який містить наступні параметри:

Name – ім’я файлу початкової текстури;

Width, height – розміри сгенерованої текстури

N – визначає розмірність сусідства

Screenshots – кількість текстур, що необхідно сгенерувати.

Для реалізації самого растрового зображенння використовується класс Bitmap. Растрове зображення складається з піксельних даних для графічного зображення та його атрибутів. Існує багато стандартних форматів для збереження растрового зображення у файлі. GDI+ підтримує такі формати файлів: BMP, GIF, EXIF, JPG, PNG і TIFF. У поточній роботі сгенеровані текстури зберігаються у форматі PNG.

Основні методи, у яких реалізовані алгоритми процедурної генерації:

FullGenerator() – простий алгоритм пошуку найближчих сусідів, результати на рисунках 3.26 – 3.29:

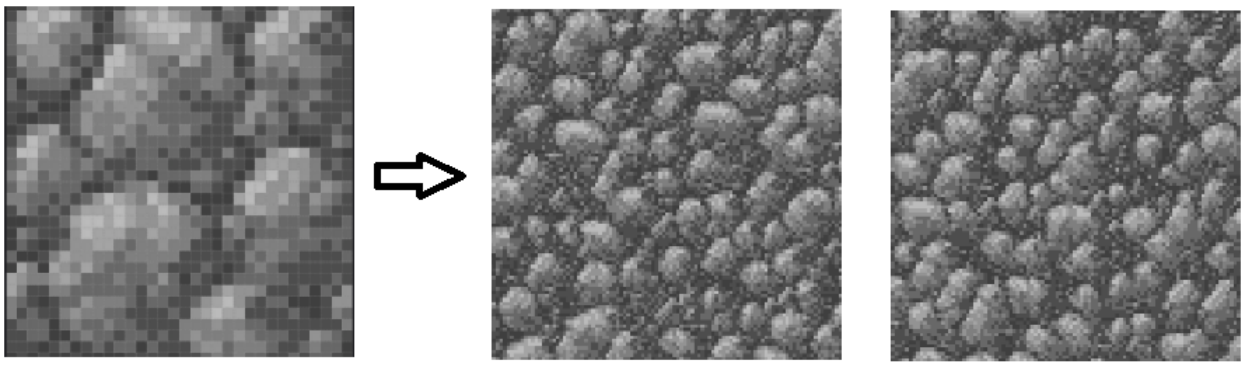

Рисунок 3.26 – Результат роботи алгоритму

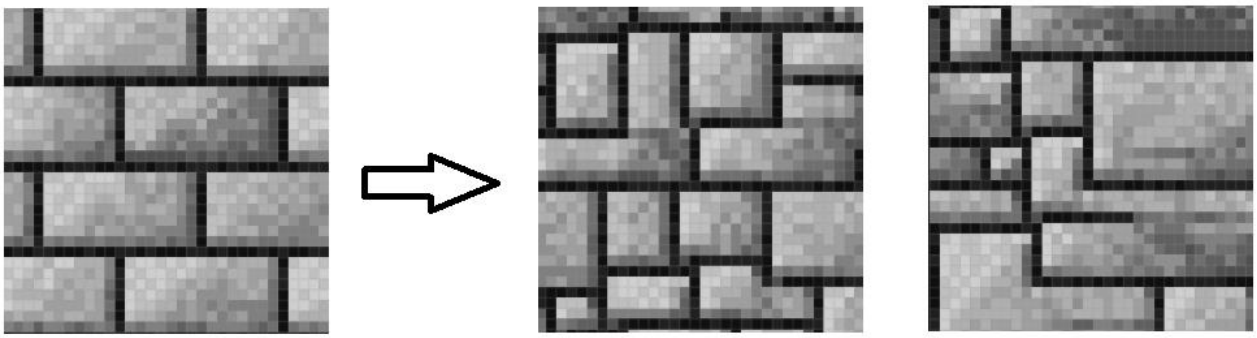

Рисунок 3.27 - Результат роботи алгоритму

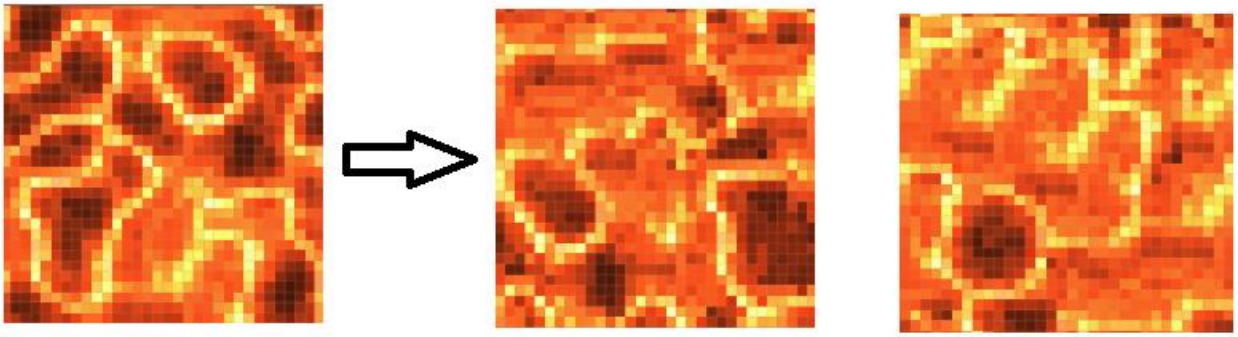

Рисунок 3.28 – Результат роботи алгоритму

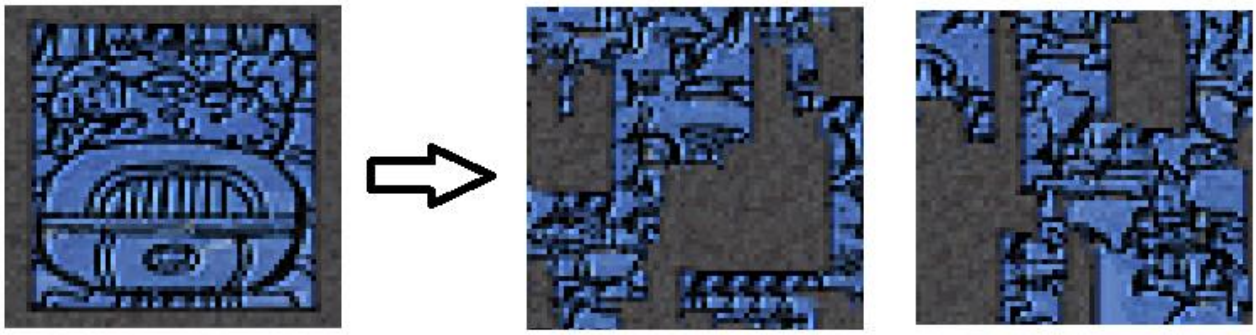

Рисунок 3.29 – Результат роботи алгоритму

KGenerator() – інша імплементація алгоритму, яка реалізує особливість процедурної генерації – масштабування завдяки збільшенню значення радіусу сусідства, результат

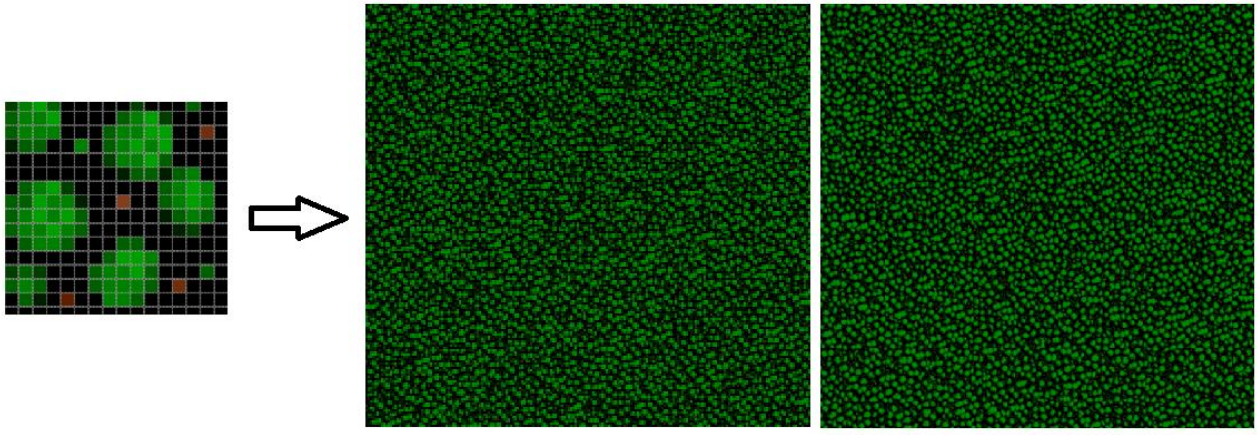

Рисунок 3.30 – Результат роботи алгоритму

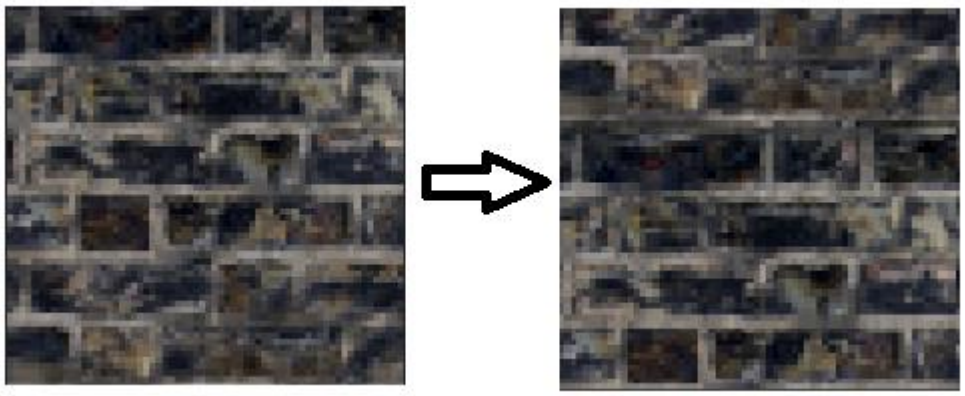

Рисунок 3.31 – Результат роботи алгоритму

ReSynthesis() – імплементація алгоритму Харісона, реалізовано підтримку масштабування текстури без втрати якості.

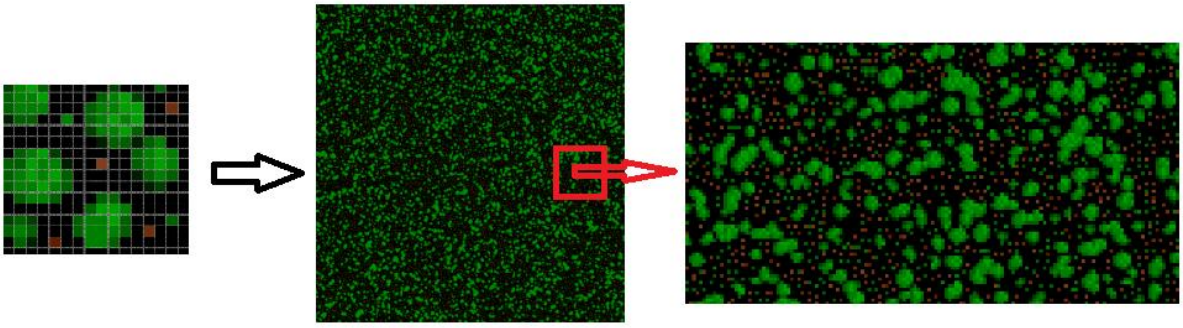

Рисунок 3.32 – Результат роботи програми

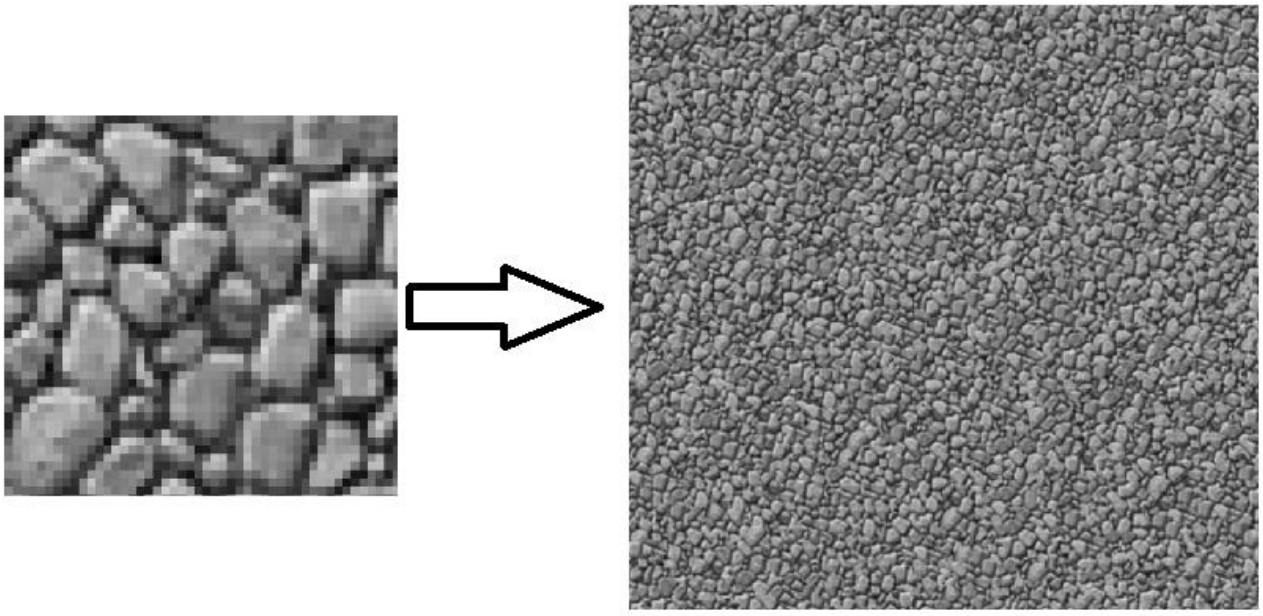

Рисунок 3.33 – Результат роботи алгоритму.

3.3 Керівництво користувача

Робота з додатком оцінюється, як дуже проста, так як додаток представлений у вигляді консольного додатку. Припускається, що додаток буде корисним для розробників та дизайнерів, а отже не вимагає особливих знань та потреб від користувача. У додатку не задіяні жодні сторонні бібліотеки, а отже робота алгоритму покладається лише на стандартні бібліотеки.

Для генерації власних текстур, користувачу необхідно ввести потрібні параметри у файл samples.xlm

```
<sample method="It1" name="lava" N="2" screenshots="2" width="100" height="100" indexed="True"/>
```

Рисунок 3.34 – редагування параметрів

У якому:

method – обраний користувачем генератор;

name – ім’я файлу початкової текстури;

width, height – потрібні розміри сгенерованої текстури;

n – визначає розмірність сусідства, тобто регулює параметр схожості;

screenshots – кількість текстур, що необхідно сгенерувати, збільшеня параметру призведе до зростання роботи алгоритму та більшої кількості сгенерованих текстур;

indexed – параметр, що відповідає за індексацію текстур у назві сгенерованого файлу, не є необхідним.

Генератор можна імпортувати в інші проекти, шляхом імпорту окремих класів, або цілого файлу, отримана бібліотека займає не більше 10кб.

Висновки до розділу 3

Текстури дуже важливі для широкого спектру застосувань у мультимедії, комп'ютерній графіці та обробці зображень. З іншого боку, їх важко генерувати. Мета цього розділу була розробити практичний інструмент для ефективного синтезу широкого спектру текстур. Алгоритм є загальним та натхненним методами випадкового поля Маркова: широкий спектр текстур можна синтезувати без будь-яких знань про процеси їх фізичного формування. Алгоритм також ефективний: завдяки належному прискоренню за допомогою TSVQ типові текстури можуть створюватися за лічені секунди на поточних ПК та робочих станціях. Алгоритм також простий у використанні: потрібен лише приклад початкової текстури.

Замість побудови явних ймовірнісних моделей, наш алгоритм використовує детермінований пошук. Цей підхід використовує сусідство фіксованого розміру, які дозволяють прискорювати TSVQ. Так як такий простий підхід працює на багатьох різних текстурах означає, що в інших методах синтезу текстур можуть бути надлишкові обчислення. Цей лгоритм має деякі з тих обмежень, що й Markov Random Field, зокрема, можуть бути представлені лише локальні та стаціонарні явища. Інші візуальні ознаки, такі як тривимірна форма, глибина, освітлення чи відображення не можна відобразити за допомогою розробленого генератора.

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«Розробка алгоритму процедурної генерації в мультимедійній сфері»

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 401.21810126

Виконав студент 4-го курсу, групи 401
_____*А.О.Буданов*
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Консультант____ *О.В.Макарова*
ст.викладач
(наук. ступінь, вчене звання)
_____
(підпис, ініціали та прізвище)
«___» _____ 2022р.

Миколаїв – 2022

4.1 Вступ

Враховуючи, що законодавством України проголошено пріоритет життя і здоров’я людини, важливими навичками є створення безпечних і здорових умов праці на робочих місцях, в робочих зонах, у виробничих приміщеннях. Ці умови досягаються опрацюванням питань умов праці, гігієни праці і виробничої санітарії, техніки безпеки, пожежної безпеки, цивільного захисту, екологічної безпеки та безпеки життєдіяльності людини.

Метою роботи над розділом є показати вміння вирішувати конкретні технічні, організаційні задачі забезпечення безпечних умов праці, безпеки людини у надзвичайних ситуаціях та дозволяє завершити підготовку з питань безпеки життя та діяльності людини.

Відповідно до мети, виділяються наступні завдання, що спрямовані на створення безпечних і здорових умов праці на робочому місці:

1. Виконати опис приміщення, робочого місця розробника та виробничого обладнання.
2. Оцінити умови праці в офісному приміщенні.
3. Сформулювати та надати рекомендаціїї щодо поліпшення умов праці на робочому місці.

4.2 Основна частина

4.2.1 Опис робочого приміщення

Приміщення для роботи розташовано на сьомому поверсі дев’ятиповерхової будівлі. Розміри робочого приміщення складають:

$$a \times b \times h = 6{,}0 \times 4{,}0 \times 3{,}5\text{м}$$

Де a - ширина приміщення, що дорівнює 6 метрів, b – довжина 4 метра, h - висота стелі, що дорівнює 3.5 метрів. Кількість робочих місць – одне. Приміщення має наступний план:

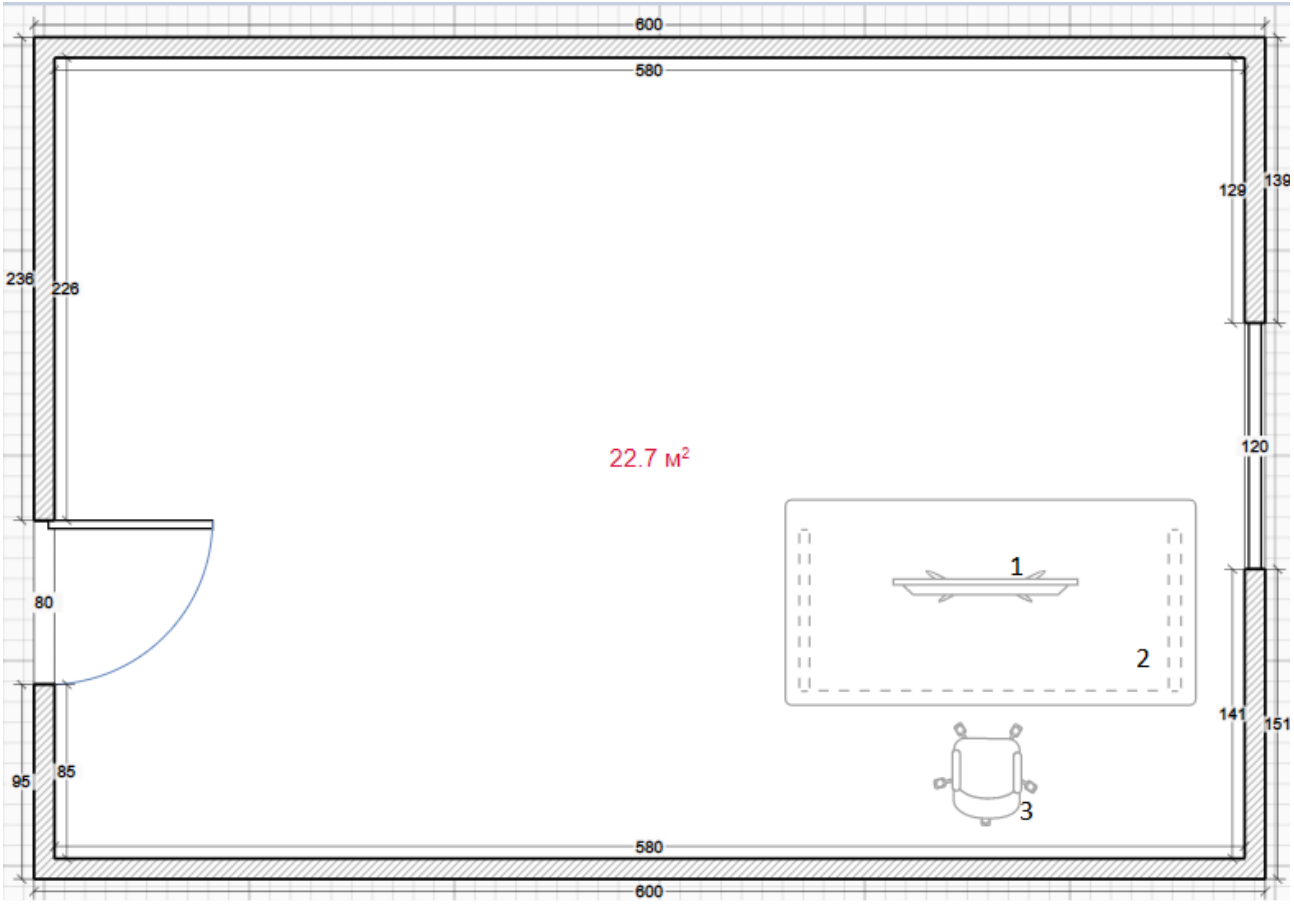

Рисунок 1 – схематичний план робочого приміщення.

Конструкція стін приміщення герметична, при цьому стіни і двері володіють вогнестійкістю не менше 45 хвилин, а міжповерхові перекриття крім цього мають гідроізоляцію. Ширина дверей 800 мм, висота – 2200 мм. Двері відкриваються всередину на 180˚, дверна коробка не має поріжка. В конструкцію дверей вмонтована ущільнювальна прокладка.

Отримані дані приведені у таблиці 1 у порівнянні із нормативними значеннями, згідно з ДСанПІН 3.3.2.007-98 «Державні санітарні правила і

норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [7].

Таблиця 4.1

Фактичні та нормативні значення параметрів приміщення

Параметр	Норма	Фактичні параметри
Площа, S	Не менше, ніж 6 $м^2$	24 $м^2$
Об'єм, V	Не менше ніж 20 $м^3$	84 $м^3$

Із таблиці 1 можна зробити висновки, що фактичні показники розглядає мого приміщення відповідають нормам нормативних документів згідно з [7].

У приміщенні розташовано одне робоче місце, обладнане сучасним персональним комп’ютером з необхідними периферійними пристроями. Для зберігання необхідного інструменту передбачена шафа. Перелік обладнання наведено в табл. 4.2.

Таблиця 4.2

Найменування предметів у серверному приміщенні

№	Назва	Кількість
1	ЕОМ	1
2	Стіл	1
3	Крісло	1
4	Шафа для обладнання	1

Напруга джерела живлення електроспоживної техніки – 380 В. Електромережа виконана у вигляді трипровідної з дотриманням усіх вимог нормативних документів. Згідно НПАОП 40.1-1.32-01 «Правила будови електроустановок» [8] за небезпекою ураження електричним струмом, приміщення відноситься до приміщень без підвищеної небезпеки ураження електричним струмом.

4.2.2 Опис мікрокліматичних умов робочого приміщення

Згідно ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень.» [9] робота відноситься до категорії «легка 1а». Виконавши необхідні вимірювання було порівняно фактичні та оптимальні параметри мікроклімату згідно [9] та наведено у таблиці 4.3

Таблиця 4.3

Фактичні та нормативні значення параметрів мікроклімату

Період року	Параметр	Оптимальний	Фактичний
Теплий	Температура	23 – 25	29
	Вологість	60 – 40	35
	Швидкість повітря	< 0,1 м/с	
Холодний	Температура	22 – 24	23
	Вологість	60 – 40	30
	Швидкість повітря	< 0,1 м/с	

Із таблиці 4.3 можна зробити висновки, що фактичні умови не відповідають межам допустимих норм, згідно [9]. Відносна вологість у приміщенні у теплий та холодний періоди року знаходяться поза межами допустимих значень, для забезпечення вимог безпечного робочого процесу необхідно прийняти заходи, що підвищать вологість. Наприклад, доцільним буде встановлення зволожувачів повітря, кімнатних рослин або встановлення акваріуму.

Температура повітря у теплий період року виходять за межі допустимих значень, для забезпечення вимог [9] необхідно вжити певні заходи, наприклад провести встановлення кондиціонеру.

4.2.3 Освітлення робочого приміщення

Стандарти та норми освітлення зазначені у ДБН В.2.5-28:2006. «Природне і штучне освітлення» [10].

У робочому приміщенні використовується природне, штучне та змішане освітлення.

Джерелом природного освітлення є вікно шириною 1,2 м і висотою 2 м.

Джерелом штучного освітлення є світильник з п’ятьма світлодіодними лампами потужністю 18 Вт, світловий потік 3600лм.

Для визначення освітленості робочої зони скористаємось [11] «Розрахунок штучного освітлення люмінесцентними лампами методом світлового потоку».

Формула світлового потоку має вигляд:

$$Ф = \frac{EkSZ}{N\eta}, \qquad (1)$$

де Ф – світловий потік, Лм;

E – освітленість робочого місця, Лк;

k – коефіцієнт запасу, що враховує зменшення світлового потоку ламп у процесі експлуатації, k = 1,2;

Z – коефіцієнт нерівномірності, Z= 1,1;

S – площа приміщення, м^2;

N – кількість ламп;

η – коефіцієнт використання світлового потоку.

Звідси освітленість на робочому місці дорівнює:

$$E = \frac{Ф N\eta}{kSZ} \qquad (2)$$

Для визначення коефіцієнту використання світлового потоку η потрібно розрахувати індекс приміщення i за формулою 3:

$$i = \frac{S}{h\,(A+B)}\,, \qquad (3)$$

де S – площа приміщення, S = 24 м^2;

h – висота світильників над робочою поверхнею, м;

A - ширина приміщення, A = 6 м;

B - довжина приміщення, B = 4 м.

Висота світильників над робочою поверхнею знаходиться за формулою 4

$$h = H - h_{\text{св}} - h_{\text{рп}}\,, \qquad (4)$$

де Н – висота приміщення, м;

$h_{св}$– висота світильника, м;

$h_{рп}$ – висота робочої поверхні, м.

$$h = 3.5 - 0{,}3 - 0{,}8 = 2{,}4 \text{ м}$$

$$i = \frac{24}{2{,}4\,(6+4)} = 1$$

За індексом приміщення та коефіцієнтами світлового потоку від стелі – 70%, стін – 50%, підлоги – 30 % визначаємо значення коефіцієнту використання світлового потоку η = 0,5.

Підставимо всі значення у формулу 2 для визначення освітленості:

$$E = \frac{3600 \cdot 5 \cdot 0{,}51}{1{,}2 \cdot 24 \cdot 1{,}1} = 289 \text{ Лк}$$

Освітленість на робочому місці становить 289 лк, що відповідає вимогам для IIIв розряду зорових робіт.

4.2.4 Визначення рівню шуму на робочому місці

Технічним джерелом шуму у робочому приміщенні є ЕОМ. Згідно технічних характеристик шум виробляється кулерами, що розташовані у системному блоці комп'ютера. За документацією, шум кулера у блоці живлення становить 10 дБ, шум кулера процессору становить 15 дБ, загальний рівень шуму комп'ютера дорівнює діапазону значень 25-40 дБ у залежності від навантаження системи.

Згідно ДСН 3.3.6.037-99. «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [12] допустимим значенням шуму на робочому місці вважається значення не вище 50 дБ, отже рівень шуму у поточному приміщенні не перевищує зазначені норми.

4.2.5 Випромінювання.

В приміщенні відсутні джерела інфрачервоного, ультрафіолетового та електромагнітного випромінювання. В моніторі комп’ютера використовується рідкокристалічна матриця з світлодіодною підсвіткою, що не створює значного електромагнітного випромінювання.

4.3 Висновк

Виконано аналіз умов роботи в робочому приміщенні. Були проаналізовані умови праці, згідно вимогам наступних документів: [7], [8], [9], [10], [11]. [12]

Розглянуто креслення приміщення, перелік обладнання, проведений аналіз нормативних та фактичних розмірів за документом [7], у результаті якого встановлено, що фактичні показники задовольняють встановленні правовим документом норми.

Виявлені порушення мікрокліматичних умов праці, згідно [9] та надано рекомендації для поліпшення умов праці. Розраховано рівень освітленості приміщення із задовільним висновкам згідно документу [10]. Загалом, робота у розглянутому приміщенні виконується у сприятливих умовах, якщо дотримуватись наданих рекомендацій, щодо покращення мікрокліматичних умов.

Проведено аналіз рівня шуму у приміщенні, у результаті якого доведено, що фактичні значення не перевищують задані норми документом [12].

Висновки

Під час виконання бакалаврської були досліджені методи процедурної генерації у мультимедії. Проаналізовані варіанти основної структури даних для алгоритму, який було розроблено під час роботи. У другому розділі були детально описані методи та алгоритми, які у подальшому стали основою генератора.

Під час розробки були виявлені недоліки та перспективи генератору. Окрім обмеженого синтезу та тимчасових текстур, можливе багато застосувань розробленої моделі. Наприклад.

Багатовимірна текстура: поняття текстури природно поширюється на багатовимірні дані. У реалізації генератора прослідковується головна особливість алгоритмів - послідовності рухів. Ця ж техніка може бути і безпосередньо застосована до твердих текстур або анімованого синтезу твердої текстури. У подальшому стоїть завдання розширити алгоритм для генерації структурованих твердих текстур з 2D-прикладів.

Стиснення/декомпресія текстур: текстури зазвичай містять повторювані шаблони та високочастотну інформацію, тому вони недостатньо стиснуті такими методами на основі JPEG. Проте методи стиснення на основі кодової книги добре працюють з текстурами. Це говорить про те, що текстури можуть бути стиснуті поданою технікою синтезу. Стиснення б складалось зі створення кодової книги, але, на відміну від A. C. Beers, M. Agrawala, and N. Chaddha «Rendering from compressed textures» [5], не мають кодових індексів, які потрібно буде створити;. Декомпресія повина складатися з синтезу текстури. Цей крок декомпресії, якщо прискорити, ще на порядок швидше поточної реалізації програмного забезпечення, подальший розвиток може бути використаний в режимі реального часу текстурного відображення. Перевага цього підходу перед [5] у тому, що виконується більше та швидше стиснення, оскільки передається тільки кодова книга.

Синтез/редагування руху в анімації: деякі анімації можуть ефективно моделюватися як просторово-часові текстури. Інші, такі як тварина або рух людини, занадто високо структуровані для такого прямого підходу. Однак

можна було б закодувати їх рух як суглоб кутів, а потім застосувати текстурний аналіз-синтез до отриманого 1D сигналу руху.

Моделювання геометричних деталей: моделі, відскановані з реального світу часто містять геометричні деталі, схожі на текстуру, моделі, дорогі для зберігання, передачі або маніпулювання. Ці геометричні деталі можуть бути представлені у вигляді карти зміщення над гладкою поверхнею, як представленно у «V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes.» [13]. Отримані карти зміщення мають бути стиснутим/розпакованим як 2D текстури за допомогою розробленного у БКР генератору. Розглядаючи цю ідею далі, було виявлено, що це поширена та актуальна проблема в багатьох ситуаціях сканування великих об’єктів, про що сказано в «M. Levoy, K. Pulli, The Digital Michelangelo Project: 3D scanning of large statues.» [14], може бути вирішена за допомогою синтезу обмежених текстур.

Прямий синтез над сітками(meshes): відображення текстур на нерегулярні тривимірні сітки шляхом проекції часто викликають спотворення, згідно з «M. Segal, C. Korobkin,. Fast shadows and lighting effects using texture mapping. Computer Graphics»[21]. Ці спотворення іноді можна виправити шляхом встановлення відповідної параметризації сітки, але більш прямим підходом було б синтезувати текстуру безпосередньо над сіткою. В принципі, це можна зробити за допомогою поточного генератора. Однак для цього знадобиться продовження звичайного операції обробки сигналу, такі як фільтрація та зниження дискретизації до неправильні 3D-сітки.

У результаті виконання БКР був розроблений додаток, що реалізує генератор процедурної генерації текстур, в основі якого лежить гнучка структура даних, що забезпечує розширення генератору шляхами, що описані вище та можливою інтерграцією генератора в інші проекти, завдяки гнучкості та малого розміру займаного місця у проекті.

Перелік посилань

1. «A. Witkin and M. Kass. Reaction-diffusion textures»

2. «S. P. Worley. A cellular texture basis function»

3. «S. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximun entropy»

4. «P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics»

5. «P. Brodatz. Textures: A Photographic Album for Artists and Designers»

6. «A. Efros and T. Leung. Texture synthesis by non-parametric sampling»

7. «Санітарні норми виробничого шуму, ультразвуку та інфразвуку. Державні санітарні норми. ДСН 3.3.6.037-99. – К., 1999.»

8. «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. – К.: Постанова Головного державного санітарного лікаря України. 1998 - № 7. »

9. «НПАОП 40.1-1.32-01. Правила будови електроустановок. Електрообладнання спеціальних установок»

10. «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – К., 2000. – С.16»

11. «Природне і штучне освітлення: ДБН В.2.5-28:2006. – К. : Міністерство будівництва, архітектури та житлово-комунального господарства України, 2006. – 68 с. – (Національні стандарти України).»

12. «Розрахунок штучного освітлення люмінесцентними лампами методом світлового потоку»

13. «V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes.»[

14. «M. Levoy, K. Pulli, The Digital Michelangelo Project: 3D scanning of large statues.»

15. «M. Szummer and R. W. Picard. Temporal texture modeling»

16. «J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes»

17. «E. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes»

18. «K. Popat and R. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression»

19. «A. C. Popat. Conjoint Probabilistic Subband Modeling»

20. «S. Nene and S. Nayar. A simple algorithm for nearest neighbor search in high dimensions»

21. «M. Segal, C. Korobkin,. Fast shadows and lighting effects using texture mapping. Computer Graphics».