Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів плавного переходу між рівнями деталізації для ефективної візуалізації 3D просторів
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗзм-22-1

Шелємєтьєв Е.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник доц. каф. ПІ Назаров О.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Навчально-науковий центр заочної форми навчання

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми Освітньо-наукова

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
зав. кафедри _____
(підпис)
"__"_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шелємєтьєву Едуарду Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: "Дослідження методів плавного переходу між рівнями деталізації для ефективної візуалізації 3D просторів"
   затверджена наказом університету від "__" _____ 2024 р. № _____
2. Термін подання студентом роботи до екзаменаційної комісії "12" 06 2024 р.
3. Вхідні дані до роботи календарний план роботи, методичні вказівки до оформлення пояснювальної записки, приклади відкритих даних у різних форматах
4. Перелік питань, що потрібно опрацювати в роботі: мета роботи, аналіз предметної галузі і постановка задачі, огляд та аналіз літературних джерел з дослідження, огляд методів плавного переходу між рівнями деталізації, програмна реалізація, аналіз результатів, висновки

Календарний план

№	Назви етапів	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	01.04.2024	виконано
2	Аналіз методів для дослідження	03.04.2024	виконано
3	Огляд методів плавного переходу між рівнями деталізації	04.04.2024	виконано
4	Програмна реалізація	08.04.2024	виконано
5	Аналіз результатів	15.04.2024	виконано
6	Написання пояснювальної записки	22.04.2024	виконано
7	Підготовка доповіді та презентації	13.05.2024	виконано
8	Підготовка роботи для перевірки на антиплагіат та проходження нормоконтролю	28.05.2024	виконано
9	Оцінка роботи рецензентом	03.06.2024	виконано
10	Отримання відзиву керівника	04.06.2024	виконано
11	Попередній захист кваліфікаційної роботи	05.06.2024	виконано
12	Здача роботи в електронних архів	06.06.2024	виконано
13	Отримання допуску до захисту	11.06.2024	виконано
14	Захист кваліфікаційної роботи	12.06.2024	виконано

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. каф. ПІ Назаров О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 62 с., 29 рис., 1 таблиця, 16 джерел.

3D-РЕНДЕРИНГ, АЛЬФА-ЗМІШУВАННЯ, ГЕОМОРФІНГ, ЕФЕКТ ПОППІНГУ, КОМП'ЮТЕРНА ГРАФІКА, ШУМИ, LOD.

Об'єктом дослідження є 3D-візуалізація (рендеринг).

Предметом дослідження є процес забезпечення плавного переходу між LOD моделями.

Метою роботи є підвищення ефективності візуалізації великих 3D сцен із урахуванням плавності переходу між моделями з різним рівнем деталізації.

В ході роботи виконується дослідження існуючих алгоритмів плавного переходу між рівнями деталізації з точки зору обчислювальної складності та візуального вигляду.

Результати дослідження дозволяють прийняти рішення про застосування того чи іншого методу плавного переходу з урахуванням їх сильних та слабких сторін.

3D-RENDERING, ALPHA-BLENDING, COMPUTER GRAPHICS, GEOMORPHING, LOD, NOISE-BLENDING, POPPING EFFECT.

The object of study is 3D-rendering.

The subject of study is providing smooth transitions between LOD models.

The purpose of the work is to increase rendering performance of large 3D-scenes considering smooth transitions between models with different levels of detail.

The study examines existing algorithms for enabling smooth transitions between levels of detail considering computational complexity and visual appeal of said methods.

The results of the work provide the ability to choose a method for smooth transitions based on its strengths and weaknesses.

Умови публікації пояснювальної записки

Я, Шелємєтьєв Едуард Олександрович,
(прізвище, ім'я, по батькові)

студент гр. ІПЗзм-22-1, здобувач вищої освіти на другому (магістерському) рівні

кафедра програмної інженерії
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему Дослідження методів плавного переходу між рівнями деталізації для ефективної візуалізації 3D просторів,
(назва роботи)

що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням "Про протидію академічному плагіату в ХНУРЕ", згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LOD	– (англ.) Level of Detail, рівень деталізації 3D-моделі
Ефект поппінгу	– небажаний ефект в комп'ютерній графіці, який виникає при переході між рівнями деталізації моделей в ситуаціях, коли глядач помічає раптову зміну моделі
Меш	– (англ.) mesh, полігональна сітка – набір вершин, ребер та граней, що описують форму багатогранного об'єкта в тривимірній графіці

ВСТУП

Рівні деталізації (Levels of Detail, LOD) відіграють важливу роль у покращенні продуктивності додатків із тривимірною графікою, забезпечуючи баланс між швидкістю опрацювання кадру зображення та досвідом користувача. Суть цього методу оптимізації полягає в тому, щоб малювати більш прості 3D-моделі, чим далі вони знаходяться від віртуальної камери. Це дозволяє зосередити обчислювальні ресурси на більш важливих для користувача об'єктах.

Оскільки положення камери дуже часто є динамічним, тому деталізацію деяких об'єктів необхідно здійснювати безпосередньо в полі зору глядача. Виникає проблема плавного переходу між різними рівнями деталізації – заміни спрощених моделей на високодеталізовані та навпаки.

Ця робота досліджує методи плавного переходу між рівнями деталізації, які спрямовані на пом’якшення горезвісного "ефекту поппінгу". Цей ефект виникає, коли перехід між LOD призводить до різких і помітних перемикань рівня деталізації, що порушує візуальну узгодженість сцени. Вивчаючи та порівнюючи різні підходи, проведене дослідження має на меті надати цінну інформацію про ефективні стратегії для запобігання або мінімізації ефекту поппінгу, що зрештою сприятиме знаходженню балансу продуктивності та оптимального візуального досвіду користувача.

Отже метою роботи є підвищення ефективності візуалізації великих 3D сцен із урахуванням плавності переходу між моделями з різним рівнем деталізації.

Об'єктом дослідження є 3D-візуалізація (рендеринг).

Предметом дослідження є процес забезпечення плавного переходу між LOD моделями.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати існуючі методи та алгоритми для усунення ефекту поппінгу при переході між рівнями деталізації;
- провести дослідження факторів, що впливають на ефективність малювання віддалених об'єктів;

- реалізувати та порівняти методи плавного переходу між рівнями деталізації;
- зробити висновок про доречність застосування того чи іншого підходу.

На основі отриманих результатів можна буде зробити висновки про те:

- який алгоритм застосовувати для забезпечення найбільшої швидкодії;
- який алгоритм дає найбільшу плавність переходу;
- наскільки взагалі доцільно використовувати плавний перехід між рівнями деталізації у порівнянні з дискретними LOD.

Дана робота є актуальною зараз і буде такою в майбутньому, оскільки 3D візуалізація стає все більш поширеною та має безліч застосувань у сучасному світі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі

Сьогодні 3D візуалізація має багато застосувань: відео-ігри, графічний дизайн, візуальні ефекти, віртуальна реальність, кінематографія, візуалізація, віртуальна інженерія тощо. Перелічені сфери діяльності вимагають від 3D графіки гарного вигляду (реалістичність або стилізованість) та ефективності. Часто ці дві вимоги конфліктують між собою, і прикладним спеціалістам приходиться шукати компроміс між якістю зображення та вартістю його генерації. Особливо складним з цієї точки зору є 3D-рендеринг в реальному часі, коли вартість створення зображення виражається в кількості кадрів в секунду (FPS).

Часто дуже ефективним способом підвищення швидкості візуалізації є використання 3D моделей з різними рівнями деталізації (Level Of Detail). Суть цього методу полягає в малюванні більш простої геометрії та використанні більш швидких шейдерів, чим далі зображуваний об'єкт знаходиться від камери [1].

Наприклад, замість рендерингу кожної деталі дуже далеко розташованого скляного хмарочоса можна намалювати розтягнутий куб із заздалегідь створеною текстурою. При такому підході важливо балансувати кількість деталей та відстань до глядача, оскільки можна або втратити ефект занурення в 3D світ, або отримати дуже великий час генерації кадру.

При роботі з LOD необхідно також приділити увагу плавному переходу між рівнями деталізації при наближенні камери до об'єкту. Наприклад, якщо було прийняте рішення малювати віддалені дерева за допомогою квадрату з натягнутою на нього текстурою (так званий billboard), то коли віртуальна камера дуже швидко наближається до лісу, можна помітити, як 2D зображення дерева раптово стає 3D моделлю (ефект поппінгу). Це явище негативно впливає на досвід користувача, оскільки стає явною фальшивість віртуального світу.

Щоб усунути ефект поппінгу, використовуються різні методи: LOD Blending (плавне змішування двох сусідніх рівнів деталізації за допомогою прозорості або використання шумів) та геоморфінг (створення додаткових

проміжних станів моделі за допомогою наближення геометрії одного рівня деталізації до іншого).

І хоча сьогодні існує певна кількість методів, кожний підхід має свої переваги та недоліки, і часто необхідно знаходити баланс, який задовольняє вимоги до продуктивності та гарного вигляду.

1.2 Актуальність роботи

Дана робота має значну актуальність у контексті 3D-графіки та віртуальних середовищ. Її результати вирішують критичну проблему, яка безпосередньо впливає на досвід користувача та продуктивність прикладних застосунків.

Сучасні додатки та 3D-моделі прагнуть до постійного підвищення рівня деталізації. Якщо раніше обчислювальна техніка була здатна намалювати відносно невелику кількість 3D полігонів, то в останні роки зображення, які згенеровані комп'ютером (CGI), важко відрізнити від фотографій з реального світу. Це досягається зокрема за допомогою використання високодеталізованих моделей.

І хоча сучасне апаратне забезпечення здатне малювати велику кількість полігонів, обчислювальні ресурси не є безлімітними (особливо у контексті 3D-рендерингу в реальному часі та на мобільних платформах), тому розробникам програмного забезпечення все ще необхідно робити компроміси. LOD досі є обов’язковим методом для збереження ілюзії безперервності та оптимальної продуктивності.

Користувачі, які переміщуються у віртуальному просторі, часто стикаються з "ефектом поппінгу", що негативно впливає на занурення. Вибір неправильного методу плавного переходу між рівнями деталізації руйнує досвід користувача, оскільки увага глядача буде спрямована не на те, що задумав дизайнер або розробник, а на 3D-об'єкт, який здійснює перехід на інший рівень якості.

Досліджуючи різні методи запобігання цьому явищу, ця робота має на меті запропонувати практичні рішення, які можна застосувати в широкому спектрі додатків: від відеоігор до архітектурного моделювання та віртуальної реальності.

Підсумовуючи, актуальність дослідження полягає в його можливості покращити взаємодію з користувачем у низці додатків. Незалежно від того, чи йдеться про ігри, симуляції чи платформи з обмеженими ресурсами, результати цього дослідження можуть позитивно вплинути на точність зображення, продуктивність і досвід користувача.

1.3 Постановка задачі

У відповідності до виявлених проблем опишемо задачу кваліфікаційної роботи:

- проаналізувати існуючі методи та алгоритми для усунення ефекту поппінгу при переході між рівнями деталізації;
- провести дослідження факторів, що впливають на ефективність рендерингу віддалених об'єктів;
- виконати програмну реалізацію досліджуваних алгоритмів;
- зробити висновок про доречність застосування того чи іншого підходу.

На основі отриманих результатів можна буде зробити висновки про те:

- який алгоритм застосовувати для забезпечення найбільшої швидкодії;
- який алгоритм дає найбільшу плавність переходу;
- наскільки взагалі доцільно використовувати плавний перехід між рівнями деталізації у порівнянні з дискретними LOD.

Отже, проведене дослідження дозволить обрати той чи інший алгоритм забезпечення плавного переходу між рівнями деталізації з урахуванням їх сильних та слабких сторін.

2 АНАЛІЗ МЕТОДІВ ДЛЯ ДОСЛІДЖЕННЯ

2.1 Обґрунтування методів та етапів дослідження

Об'єктом дослідження є 3D-візуалізація (рендеринг).

Предметом дослідження є процес забезпечення плавного переходу між LOD моделями.

Метою роботи є підвищення ефективності візуалізації великих 3D сцен із урахуванням плавності переходу між моделями з різним рівнем деталізації.

В даному науковому дослідженні використовується емпіричний науковий метод "експеримент": протягом серії дослідів будуть визначені параметри обраних алгоритмів забезпечення плавного переходу між рівнями деталізації (величина ефекту поппінгу та швидкодія).

Потім буде використаний теоретичний метод "аналіз" для розкладання отриманих результатів окремо один від одного. В результаті буде прийнято рішення, який алгоритм більше підходить в конкретній ситуації і з яких причин.

Дане наукове дослідження складається з наступних етапів:

- підготовка до дослідження – визначається мета, обґрунтовується предмет та об'єкт дослідження, формується гіпотеза та методика дослідження;
- експериментальне дослідження та обробка даних – підготовка та проведення дослідів, обробка отриманих даних;
- аналіз результатів дослідження – винесення висновку про доцільність використання певних алгоритмів плавного переходу між рівнями деталізації;
- оцінка застосування результатів дослідження.

2.2 Розробка формальної (математичної) моделі предметної області

В ході дослідження буде проведене порівняння алгоритмів забезпечення плавного переходу між LOD моделями за двома факторами:

- час генерації кадру зображення;
- значимість ефекту поппінгу.

Кожний досліджуваний алгоритм характеризується наступною математичною моделлю сірого ящика (див. рис. 2.1): він є виразом від складності двох сусідніх рівнів деталізації, між якими здійснюється плавний перехід (x_1 та x_2) та відстанню об'єкта до камери (d), а також випадкової похибки ξ, яку ми не можемо контролювати.

Функція $f_1(x_1,x_2,d)$ є контрольованим параметром, що позначає досліджуваний метод плавного переходу між рівнями деталізації. Результатом функції є параметр z, який позначає набір інструкцій до графічного процесора, щоб намалювати 2 сусідні рівні деталізації на заданій відстані від камери.

Функція $f_2(z,\xi)$ позначає середовище, що виконує малювання (операційна система та графічний процесор). Це середовище не є контрольованим і представляє собою чорний ящик. Все, що ми можемо з ним зробити – це передавати вхідні команди та дані, а на виході отримаємо згенероване зображення.

Зауважимо, що для спрощення подальших обчислень оцінки складності рівнів деталізації x_1 та x_2 береться кількість полігонів в моделі (трикутників, що формують грань об'єкта) без урахування положення їх нормалі відносно камери (при застосуванні техніки Back-face culling полігони, що не є видимими з камери, наприклад, задня частина будинку, відсікаються, і не приймають участі в подальших кроках графічного конвеєру).

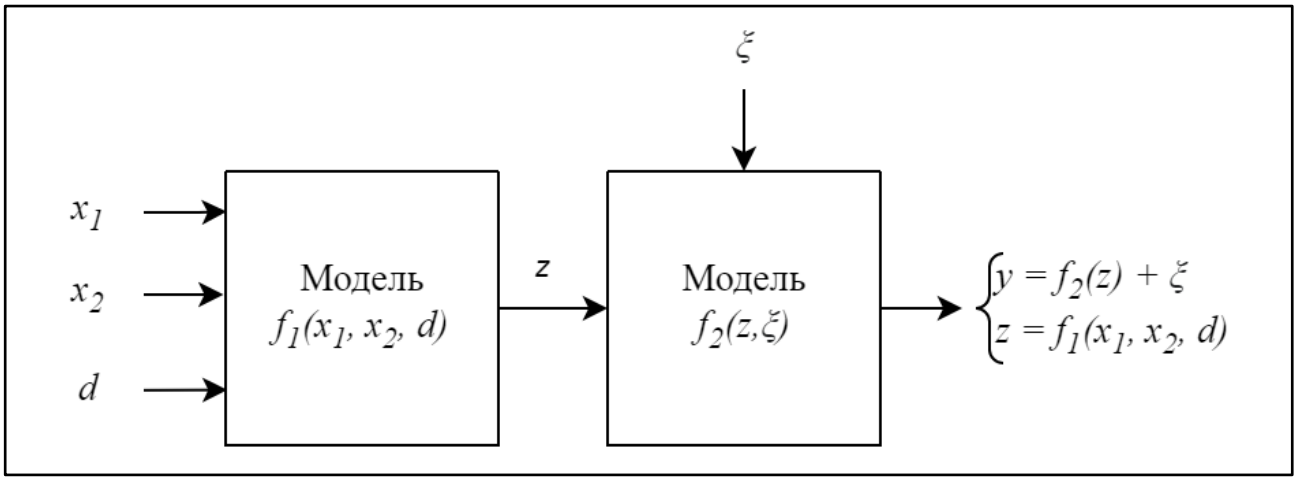

Рисунок 2.1 – Математична модель сірого ящику для алгоритму плавного переходу між рівнями деталізації

При проведенні експерименту щодо порівняння часу генерації зображення y – це час (в мілісекундах) генерації одного кадру, а ξ – це випадкова похибка вимірювання.

В експерименті з оцінки значимості ефекту поппінгу y – це відмінність еталонного зображення (оригінальної високополігональної моделі) та зображення, згенерованого алгоритмом плавного переходу між рівнями деталізації.

В цьому експерименті усі випадкові значення (текстури шумів тощо) попередньо згенеровані, тому результат експерименту є повністю детермінованим (визначається x_1, x_2, d), а значення похибки ξ дорівнює нулю.

Після проведення експериментів необхідно буде зробити висновок про те, наскільки алгоритми відповідають тим чи іншим критеріям та навести приклади, в яких ситуаціях доцільно використовувати певний метод плавного переходу між рівнями деталізації.

2.3 Методологія експерименту

Для проведення експериментів буде створено програмне забезпечення, яке буде складатися з наступних функціональних модулів:

- модуль для вимірювання швидкодії алгоритмів при відображенні великої кількості 3D моделей на різній відстані від камери;
- модуль для вимірювання ефекту поппінгу при порівнянні роботи досліджуваного методу з еталоном (високополігональною моделлю).

Алгоритми плавного переходу між рівнями деталізації, які будуть досліджуватися:

- альфа-змішування;
- змішування за допомогою шумів;
- геоморфінг.

Ці методи будуть порівнюватися з рендерингом дискретних рівнів деталізації без плавного переходу.

В експерименті з вимірювання швидкодії алгоритмів плавного переходу між рівнями деталізації буде виміряно час, необхідний для рендерингу великої

кількості моделей (50, 100, 250, 500, 750, 1.000, 1.500 та 2.000 моделей), які розташовані на різній відстані до камери. Кількість об'єктів повинна бути достатньо великою, оскільки сучасні графічні процесори дуже швидко опрацьовують великі 3D сцени.

Об'єкти розташовуються по осі x в діапазоні від початку видимої області камери до відстані, необхідної для активації найнижчого рівня деталізації з однаковим інтервалом, який дорівнює (2.1):

$$dx = (w - camera_x) / n, \tag{2.1}$$

де dx – інтервал, за яким розміщуються об'єкти на сцені,

n – кількість об'єктів,

$camera_x$ – координата камери,

w – відстань мінімального рівня деталізації.

Відповідно, камера спрямована вздовж позитивної осі x.

Також Occlusion Culling повинен бути вимкнутим, оскільки моделі перекривають одна одну (якщо увімкнути Occlusion Culling, то об'єкти, що розташовані спереду, будуть перекривати задні моделі, і через це останні не будуть малюватися).

Як було зазначено раніше, в даному експерименті наявна похибка ξ, тому вимірювання часу необхідно проводити декілька разів (вважатимемо, що 3600 разів є достатнім) та взяти середнє значення.

Оскільки експеримент буде проводитись на сучасній багатопроцесній операційній системі, то вважатимемо, що похибки вимірювання часу спричинені роботою інших процесів на даному комп'ютері, тому середній час – є саме часом роботи алгоритму.

Величина ефекту поппінгу вимірюється наступним чином:

– досліджувана 3D-модель розміщується на сцені на мінімальній відстані до камери, щоб спочатку був відображений максимальний рівень деталізації;

- об'єкт з постійним інтервалом віддаляється вздовж прямої погляду камери, проходячи усі стани рівнів деталізації;
- після кожного переміщення зберігається згенероване зображення;
- після досягнення найнижчого рівня деталізації об'єкт так само поступово повертається назад;
- кроки 1-4 проводяться для еталонної моделі та заданого алгоритму, і знаходиться абсолютна різниця між значеннями пікселів зображень. Результуюче зображення має лише один канал, який інтерпретується як чорний (значення 0) та білий (1). Оскільки вхідні зображення є багатоканальними (RGB), то різниця є середньою арифметичною між усіма каналами;
- для кожного зображення-різниці знаходиться середньоквадратична помилка (Root-mean-square error, RMSE) [2].

В результаті отримаємо єдине число – наскільки зображення, згенероване певним методом відрізняється від еталону.

Зображення, отримане на кроці 4 може бути корисним для візуальної оцінки різниці роботи алгоритмів.

Оскільки кількість 3D-моделей, як можна підготувати для дослідження, є обмеженою, експеримент для кожної моделі буде проводитися декілька разів: кожного разу під різним кутом погляду. Результат для кожної моделі усереднюється – це зробить його більш надійним та більш репрезентативним (якщо досліджувати модель тільки з однієї сторони, то можна випадково потрапити на ракурс, який має екстремальне значення поппінгу, та невірно зробити висновки).

В результаті експерименту буде знайдено:

- найшвидший та найповільніший алгоритми;
- який алгоритм має найбільш виразний ефект поппінгу.

На основі отриманих результатів можна буде зробити висновки про те:

- який алгоритм застосовувати для забезпечення найбільшої швидкодії;

- який алгоритм дає найбільшу плавність переходу;
- наскільки взагалі доцільно використовувати плавний перехід між рівнями деталізації у порівнянні з дискретними LOD.

2.4 Природа експериментальних помилок і невизначеностей

Помилка вимірювання наявна тільки при проведенні експерименту з оцінки швидкодії алгоритмів плавного переходу між рівнями деталізації. Не зважаючи на те, що програмі передається один й той самий набір даних, час їх опрацювання кожний раз є різним.

Це пов'язано з тим, що застосунок виконується не в окремому ізольованому середовищі: паралельно з ним на тому самому пристрої виконується багато інших процесів та потоків.

Також слід враховувати апаратний і програмний кеш, а також інтерпретацію та JIT-компіляцію коду. Через них, як правило, перший експеримент виконується трохи довше. Тому на практиці перед вимірюванням часу виконання програми здійснюють так званий "прогрів" (бажано з якомога більшою кількістю умовних переходів). Це збільшує шанс того, наступні експерименти будуть мати приблизно однаковий час роботи.

При підрахунку величини ефекту поппінгу необхідно виконувати операції за допомогою цілих чисел (64-бітних) із урахуванням множнику масштабу, а наприкінці перевести отриману суму у число з плаваючою комою. Це треба робити тому, що при складанні великої кількості чисел з плаваючою комою накопичується похибка, яка може бути досить значною.

2.5 Характеристики досліджуваних 3D-моделей

При порівнянні алгоритмів переходу між рівнями деталізації використовуються 3D-моделі в форматі Wavefront (текстові файли з розширенням obj), експортовані за допомогою редактору Blender. Самі моделі беруться з відкритих джерел (з невеликими модифікаціями) або створюються самостійно. Рівні деталізації генеруються за допомогою модифікатору Decimate. Цей

модифікатор виконує згортку найближчих вершин на основі коефіцієнту. Це дозволяє знизити кількість полігонів моделі та створити моделі гіршої якості.

Специфікація досліджуваних моделей наведена в таблиці 2.1.

Всі перелічені моделі мають такі атрибути вершин: положення та нормаль (тривимірні вектори). Окрім них, модель "tree" підтримує координати текстур (двовимірний вектор) та має 2 текстури (стовбура та листя).

Таблиця 2.1 – Характеристика 3D-моделей

№	Назва	Рівень деталізації	Кількість полігонів	Джерело
1	dragon	0	249.881	Stanford Computer Graphics Laboratory
		1	4.163	
		2	1.561	
		3	249	
2	tree	0	2.576	Самостійна робота
		1	1.476	
3	stanford-bunny	0	69.451	Stanford Computer Graphics Laboratory
		1	1.036	
		2	347	
		3	49	
4	car	0	43.612	Модифікоана модель "Beetle" Ейвена Сазерленда
		1	10.903	
		2	5.450	
		3	136	

Рівень деталізації нуль – це модель найвищої якості. Вона служить основою для генерації інших рівнів.

3 ОГЛЯД МЕТОДІВ ПЛАВНОГО ПЕРЕХОДУ МІЖ РІВНЯМИ ДЕТАЛІЗАЦІЇ

3.1 Альфа-змішування

Одним із найбільш простих з точки зору програмної реалізації є метод плавного переходу під назвою "альфа-змішування" (Alpha Blending, LOD Blending).

Суть методу полягає у одночасному відображенні двох рівнів деталізації, які змішуються за допомогою альфа-каналу (каналу прозорості) у відповідності до коефіцієнту переходу. Цей коефіцієнт приймає значення від нуля (початок переходу) до одиниці (кінець). Умовно метод можна представити за допомогою наступної формули (3.1):

$$\alpha_{result} = (1 - k) * \alpha_{prev} + k * a_{next}, \quad (3.1)$$

де α_{result} – це прозорість результуючого пікселя на екрані,

α_{prev} – це прозорість пікселя попереднього рівня деталізації (від якого почався перехід),

a_{next} – прозорість наступного (цільового) рівня деталізації,

k – коефіцієнт переходу.

Значення альфа-каналу одиниця відповідає повністю непрозорому пікселю, значення нуль – повністю прозорому (прихованому).

Коли віртуальна камера рухається і досягає відстані, за якої повинен здійснитися перехід, стара модель плавно затухає, і замість неї поступово з'являється нова модель. Коли прозорість старої моделі досягає нуля, то стару модель не малюють, щоб зекономити ресурси.

Візуально принцип роботи альфа-змішування представлений на рисунку 3.1.

При реалізації альфа-змішування важливо робити перехід протягом невеликого діапазону відстані. Наприклад, якщо необхідно зробити перехід на відстані 1м, то початок та кінець переходу мають бути 0.95м та 1.05м відповідно.

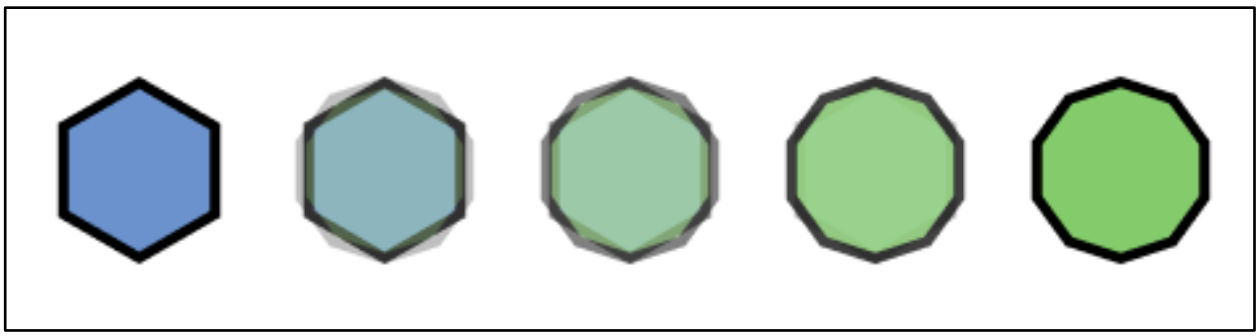

Рисунок 3.1 – Принцип роботи альфа-змішування, перехід від моделі низької якості до високої

Альфа-змішування можливо об'єднати з білбордами, оскільки для цього методу неважлива віртуальна структура моделі.

У контексті LOD-оптимізації білборди зазвичай використовуються для підвищення ефективності візуалізації найбільш віддалених об'єктів (найменший рівень якості).

Білборди – це двовимірні площини або набори з'єднаних між собою багатокутників, які завжди повернуті до камери, імітуючи тривимірний об'єкт. Ця техніка використовується для більш ефективного представлення віддалених або малих об'єктів у сцені, зменшення обчислювального навантаження та підвищення продуктивності.

Перевага білбордів полягає в тому, що візуалізація площини вимагає лише двох трикутників. Часто декілька площин об'єднують разом, щоб створити ілюзію великого об'єкту у просторі (листя дерев, кущі, хмари тощо) [3].

Даний метод плавного переходу має два великих недоліки.

По-перше, візуалізація двох моделей одночасно є дуже складною з точки зору обчислення. Головною причиною для використання рівнів деталізації є зменшення кількості полігонів, які малюються одночасно, щоб пришвидшити створення зображення на екрані, однак перехід в альфа-змішуванні потребує малювання обох моделей. Тому іноді цей метод може навіть нашкодити продуктивності.

Одним із методів боротьби з цим недоліком є обмеження на одночасну кількість об'єктів, які здійснюють перехід між рівнями деталізації. Це допомагає уникнути стрибків у кількості викликів функцій малювання, що гарантує більш стабільну кількість кадрів в секунду. Слід мати на увазі, що відкладення переходу створює затримку, яка може нашкодити досвіду користувача.

По-друге, альфа-змішування є дуже помітним для глядача на близькій відстані: за певних ситуацій модель може виглядати, як напівпрозорий привид.

Альфа-змішування дуже добре працює на великій відстані від камери, коли сусідні рівні деталізації мають невелику кількість полігонів, і візуальний ефект від змішування недостатньо помітний.

3.2 Змішування за допомогою шумів

Плавний перехід між рівнями деталізації за допомогою шумів є дуже схожим на альфа-змішування.

Однак в цьому методі прозорість кожного пікселя є дискретною величиною: або нуль, або один. Іншими словами, цей алгоритм використовує повністю прозорі або повністю непрозорі пікселі для здійснення плавного переходу.

Отже, рішення про те, чи відображувати конкретний піксель з координатами x та y кожної моделі приймається на основі числового значення випадкової величини (шуму): якщо значення коефіцієнту переходу k перевищує поріг, заданий функцією шуму, то використовуємо піксель з наступного рівня деталізації, інакше – попереднього (3.2):

$$\begin{cases} \alpha_{result} = \alpha_{next}, & \text{якщо } k > f_{noise}(x, y) \\ \alpha_{result} = \alpha_{prev}, & \text{інакше} \end{cases}, \tag{3.2}$$

де $f_{noise}(x,y)$ – функція шуму, яка повертає значення від нуля для одиниці.

Шум в комп'ютерній графіці – це псевдовипадкова величина (одновимірна або багатовимірна), яка використовується, щоб додати більше деталей до зображень, згенерованих комп'ютером. Шуми дуже легко обчислити, а їх

застосування є майже безкінечними: від візуалізації хмар і дрібних часток до симуляції морських хвиль та торнадо [4].

Для більшої продуктивності шуми розраховують заздалегідь та зберігають у вигляді текстур. Такий формат є дуже зручним для графічних процесорів, оскільки вони спеціалізовані на тому, щоб робити вибірку даних з текстур.

Отже, для плавного переходу нам необхідно мати двовимірне значення шуму в діапазоні від нуля до одиниці, заздалегідь збереженого у вигляді двовимірної текстури. Її розмір не обов'язково повинен бути великим: максимальний розмір це роздільна здатність зображення, яке генерується. Навіть якщо розрахувати занадто малу текстуру, то її можна повторити, і це мінімально вплине на досвід користувача.

Вибірку з текстури шуму можна виконувати не окремо для кожної моделі, що малюється, а на етапі обробки. Це може пришвидшити алгоритм та уникнути повторюваності шуму, якщо об'єкти знаходяться занадто близько один до одного.

До найбільш популярних функцій шуму можна віднести наступні [5]:

- білий шум – повертає псевдовипадкові значення навіть для вхідних значень, які є дуже близькими один до одного;
- градієнтний шум – виконує інтерполяцію значень білого шуму та повертає близькі значення шуму для близьких параметрів;
- шум Перліна – підвид градієнтних шумів, в якому візуальні деталі мають однаковий розмір; використовується для того, щоб зробити комп'ютерну графіку більш реалістичною;
- багатошаровий шум – використовує комбінацію градієнтних шумів з різним рівнем масштабу та ваги; дозволяє отримати шум, який має як високочастотні деталі, так і низькочастотні;
- шум Вороного – повертає значення шуму, яке виглядає як набір клітин (або відстаней між клітинами).

Окремим видом шуму є дитеринг (dithering) [6]. Його застосовують, щоб надати випадковій величині бажаний стилістичний вигляд. Приклад плавного переходу з використанням дитерингу наведений на рисунку 3.2.

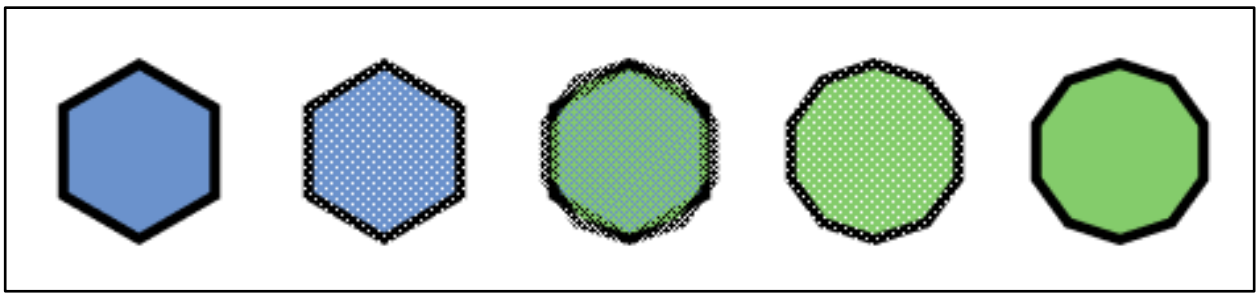

Рисунок 3.2 – Принцип роботи змішування за допомогою шумів, перехід від моделі низької якості до високої

На рисунку відображено корисну властивість змішування на основі шумів – за певного значення порогу плавного переходу можна малювати лише одну модель одночасно. Це допомагає зменшити навантаження на графічний процесор. Очевидно, що ця хитрість підходить не для всіх видів моделей та оточення.

Іншою перевагою цього методу у порівнянні з альфа-змішуванням є те, що дизайнер має повний контроль над виглядом переходу: необхідно лише замінити текстуру шуму, і можна зробити так, щоб певні частини моделі переходили швидше за інші; можна встановити напрямок переходу (знизу-вгору, від країв до центру тощо).

Перехід між рівнями деталізації на основі шумів має такі самі недоліки, що й альфа-змішування: необхідність малювати дві моделі одночасно та очевидність переходу при перегляді зблизька.

Однак перехід на основі шумів на практиці є більш ефективним з точки зору продуктивності [7].

Справа в тому, що найбільш поширений спосіб роботи з напівпрозорими моделями є набагато повільнішим за малювання непрозорих об'єктів. Напівпрозора геометрія часто потребує окремого від непрозорих об'єктів кроку графічного конвеєру, де вимкнений тест глибини та виконується сортування прозорих полігонів з заду на перед [8].

Відсутність тесту глибини викликає повторне малювання кожного пікселя (overdraw), що є дуже шкідливим для продуктивності, оскільки графічний процесор витрачає час на піксель, який все одно не буде видимим користувачу.

Сортування може бути доволі дорогим, якщо кількість трикутників моделі є великою – а це як раз і є випадком використання альфа-змішування (особливо для моделей рівнів деталізації на невеликій відстані до віртуальної камери).

На практиці метод змішування за допомогою шумів дуже часто використовується для рослинності, яка по своїй суті є відображенням випадкових величин у природі [9].

3.3 Геоморфінг

Геоморфінг – є ще одним методом плавного переходу між рівнями деталізації. Його суть полягає у апроксимації 3D моделі для створення проміжних станів переходу.

Головними операціями геоморфінгу є розділ вершин (vertex split; при збільшенні якості моделі до неї додаються нові вершини) та згортка ребр (edge collapse; при зменшенні якості моделі видаляються деякі вершини) [10].

Під час переходу здійснюється не тільки зміна кількості вершин, а й їх положення та інші додаткові атрибути: нормаль, колір, координати текстур тощо (див. рис. 3.3), що забезпечує уникнення поппінгу.

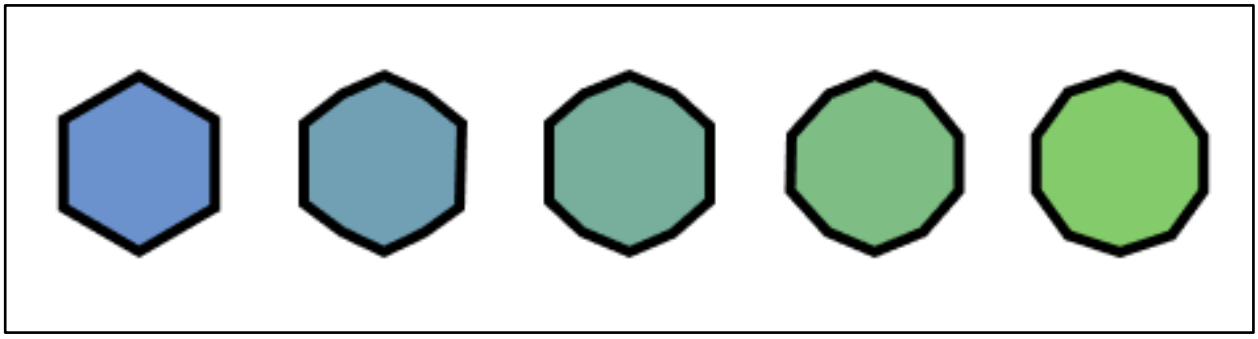

Рисунок 3.3 – Принцип роботи геоморфінгу, перехід від моделі низької якості до високої

Більшість атрибутів вершин інтерполюються лінійно. Нормалі, які є 3-вимірними векторами довжиною в одну одиницю, змінюються за допомогою інтерполяції напрямку, щоб зберегти їх довжину.

При програмній реалізації геоморфінгу важливо врахувати ситуацію, коли розділ вершин або згортка ребр виконуються під час існуючого переходу. Іншими словами, вершини повинні мати змогу виконувати анімації переходів, що перетинаються в часі.

Одним із способів оптимізації методу геоморфінгу є виконання плавного переходу тільки для видимої частини моделі (view-dependent LOD control) [11]. Ця техніка має сенс для великих (і в розмірах, і в кількості вершин) моделей. Увесь меш розподіляється на частини (кластери), які мають власні буфери пам'яті. Рішення про те, який кластер слід малювати на екрані, приймається на основі техніки оптимізації під назвою frustum culling.

Термін "frustum" стосується простору огляду у формі піраміди, який охоплює видиму область 3D-сцени [12]. Ця усічена піраміда отримується за допомогою проектування перспективи камери на ближню та дальню площини [13]. Frustum culling передбачає вибіркове відтворення лише тих об’єктів, які потрапляють у цю усічену область, відкидаючи ті, що знаходяться за межами огляду.

Усуваючи невидимі об’єкти на ранній стадії конвеєра візуалізації, ми значно зменшуємо обчислювальне навантаження та підвищуємо загальну продуктивність.

У процесі видалення об'єктів кожна модель на сцені перевіряється на зрізі, щоб визначити, перетинається вона із полем зору або лежить поза ним. Для швидкого визначення, чи є об'єкт потенційно видимим, використовуються різні алгоритми, наприклад, перевірка за допомогою сфер або Axis-Aligned Bounding Boxes (AABB). Якщо об'єкт повністю знаходиться за межами поля зору, він виключається з процесу візуалізації, запобігаючи непотрібним обчисленням геометрії, освітлення та тіней.

Ця оптимізація стає особливо корисною в сценах із великою кількістю об'єктів, дозволяючи зосередити обчислювальні ресурси на малюванні лише видимих камерою моделей.

Головною складністю при розділенні моделі на кластери є забезпечення послідовного та синхронізованого переходу для вершин, які знаходяться на межі двох кластерів. Якщо прогледіти цю деталь реалізації, то можуть виникнути розломи по краям частин мешу, які в деяких випадках викликають "ефект поппінгу".

Найбільш простим з точки зору реалізації є виконання переходу повністю на центральному процесорі. В оперативній пам'яті може зберігатися список вершин, які виконують плавний перехід, і кожен кадр необхідно виконувати оновлення їх атрибутів. Проте варто мати на увазі, що даний підхід не є оптимальним при великій кількості вершин, оскільки після оновлення даних в пам'яті центрального процесору, ці дані потрібно надіслати графічному процесору, що може викликати деяку затримку [14].

Іншим підходом є зберігання моделі високої та низької якості у пам'яті графічного процесору. Для здійснення плавного переходу використовується параметр ваги (від нуля до одиниці), за яким інтерполюються параметри вершин. Моделі різної якості можна завантажувати заздалегідь із врахуванням пріоритету. Очевидно, що цей метод вимагає більших витрат відеопам'яті, але суттєво зменшується об'єм даних, які центральний процесор відсилає відеокарті кожного кадру.

Перевагою методу геоморфінгу є те, що під час переходу між рівнями деталізації одночасно малюється лише одна модель. Також плавна зміна моделі є менш помітною для глядача на будь-якій відстані.

До недоліків цього методу можна віднести складність реалізації та необхідність особливого ставлення до розміщення 3D-моделі в пам'яті. Таким чином, геоморфінг стає ключовим фактором, що диктує, яким чином відбувається 3D-рендеринг.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

Розробка програми для проведення дослідження виконується за допомогою мови програмування C#, платформи .NET 6 та фреймворку для розробки відеоігор MonoGame (малювання на базі DirectX).

Кожний з досліджуваних методів плавного переходу представлений окремим класом, що реалізує спільний інтерфейс ILodTransition. Кожний клас отримує моделі рівнів деталізації, ступінь переходу, матрицю перетворення об'єкту, а також посилання на графічний конвеєр для малювання.

Плавний перехід між рівнями деталізації здійснюється в залежності від відстані до камери.

Альфа-змішування реалізується за допомогою окремого кроку для напівпрозорих об'єктів та подвійного малювання (double-pass rendering). Через те, що на останньому етапі графічного конвеєру Output-Merger (OM), всі пікселі, які мають значення глибини менше за глибину z-буферу, не малюються зовсім. Через це напівпрозорі об'єкти, як правило, спочатку малюються окремо від непрозорих, а потім додаються до фінального зображення з урахуванням глибини.

Double-pass rendering допомагає зменшити ефект поппінгу для певних моделей. Буфер глибини некоректно працює при малюванні напівпрозорих полігонів однієї моделі, і іноді прозорі трикутники можна побачити крізь один одного – як результат, регіони моделі мають різне значення альфа-каналу. При змішуванні рівнів деталізації нам важливо, щоб всі полігони обох моделей мали однаковий рівень прозорості, якій в сумі дорівнює одиниці. Якщо прозорість менше одиниці, то моделі просвічуються. Якщо ж прозорість більше одиниці, то RGB колір, як правило, відрізняється від оригінального кольору моделі, що дуже помітно користувачу.

Тому кожний рівень деталізації при переході малюється двічі (в сумі 4 виклики функцій малювання). Перше малювання повинно записати лише значення глибини. Будь-які зміни RGB каналів повинні ігноруватися. На цьому етапі використовується дуже спрощений піксельний шейдер, який повертає

простий колір. Це зроблено для того, щоб не витрачати час на розрахунок світла та кольору пікселів. Друге малювання моделі записує значення RGBA каналів на основі фактору змішування (формула 4.1). Попередньо потрібно вимкнути запис в z-буфер, та залишити тільки читання. Перед малюванням наступної моделі буфер глибини необхідно очистити, інакше пікселі другої моделі будуть відкинуті через глибину, яка залишилася в пам'яті:

```
private void DrawTransparentModelDoublePass(
  LodLevel lod, GraphicsDevice graphicsDevice, float alpha) {
  graphicsDevice.Clear(
    ClearOptions.DepthBuffer, Color.Transparent, 1f, 0);
  graphicsDevice.DepthStencilState = DepthStencilState.Default;
  graphicsDevice.BlendState = this.blendDepthOnly;
  this.mainMaterial.AlphaPass.Apply();
  foreach (var part in lod.Mesh.MeshParts) {
    graphicsDevice.SetVertexBuffer(part.VertexBuffer);
    graphicsDevice.Indices = part.IndexBuffer;
    graphicsDevice.DrawIndexedPrimitives(
      PrimitiveType.TriangleList,
      part.VertexOffset, part.StartIndex, part.PrimitiveCount);
  }
  using var bs = new BlendState {
    ColorSourceBlend = Blend.BlendFactor,
    ColorDestinationBlend = Blend.InverseBlendFactor,
    AlphaSourceBlend = Blend.BlendFactor,
    AlphaDestinationBlend = Blend.One,
    BlendFactor = new Color(alpha, alpha, alpha, alpha),
  };
  graphicsDevice.BlendState = bs;
  graphicsDevice.DepthStencilState=DepthStencilState.DepthRead;
  this.mainMaterial.MainPass.Apply();
  foreach (var part in lod.Mesh.MeshParts) {
    graphicsDevice.SetVertexBuffer(part.VertexBuffer);
    graphicsDevice.Indices = part.IndexBuffer;
    graphicsDevice.DrawIndexedPrimitives(
      PrimitiveType.TriangleList, part.VertexOffset,
      part.StartIndex, part.PrimitiveCount);
  }
}
```

Реалізація плавного переходу на основі шумів використовує вбудовану HLSL функцію clip для того, щоб відкинути заданий піксель з подальшої обробки. Такий піксель не записується ані в кольорову текстуру, ані в буфер глибини. Це значно спрощує обчислення зображення, оскільки можна малювати моделі лише один раз, і не потрібен окремий крок для прозорих об'єктів. Правила, за якими

кожний окремий піксель поступово відсікається або з'являється завантажуються у вигляді чорно-білої текстури розміром 16x16.

Плавний перехід на основі геоморфінгу реалізований на графічному процесорі. Це допомагає зменшити навантаження на шину пам'яті, оскільки заздалегідь заготовлений меш плавного переходу завантажується в відео пам'ять лише один раз, і подальша інтерполяція між атрибутами вершини здійснюється у вершинному шейдері за допомогою змінної Progress:

```
MainVertexShaderOutput GeomorphVS(
  in GeomorphVertexShaderInput input) {
      MainVertexShaderOutput output;

      float3 avgPos = lerp(
        input.StartPosition, input.EndPosition, Progress);
      float3 normal = lerp(
        input.StartNormal, input.EndNormal, Progress);
      output.Position = mul(float4(avgPos, 1.0),
                            WorldViewProjection);
      output.Normal = normalize(normal);

      return output;
}
```

Меш для геоморфінгу будується шляхом пошуку найближчих вершин між моделями високої та низької якості. Оскільки 3D моделі часто мають вершини з однаковими позиціями, але різними додатковими атрибутами (нормалями, UV-координатами тощо), то проста відстань між координатами вершин не є достатньою: метрика відстані повинна враховувати також нормальні вектори. Якщо проігнорувати цю властивість, то при переході трикутники можуть мати неправильну освітленість та колір, що негативно впливає на плавність переходу.

Меш для геоморфінгу зберігається в кеші за стратегією Least Recently Used (LRU). Коли перехід генерується на основі двох моделей, він додається до кешу, що дозволяє його повторне використання як при наступному виклику малювання, так і підчас одного й того ж кадру.

Створення мешу є повільною операцією для моделей з великою кількістю вершин. Найповільнішою частиною є розрахунок найближчих вершин. На жаль, навіть з використанням паралелізації затримка в генерації є досить помітною. На

практиці пропонується попередньо розрахувати усі найближчі вершини на етапі компіляції програми, і просто читати їх з файлу. Інший метод – це створення моделі в фоновому потоці, і відображувати її як тільки вона буде готовою. В досліджуваній програмній реалізації затримка не є критичною, але вона враховується при проведенні експериментів.

5 АНАЛІЗ РЕЗУЛЬТАТІВ

5.1 Аналіз метрик плавних переходів

Проведемо вимірювання швидкодії алгоритмів дискретного та плавного переходів. Комп'ютер, на якому виконуються вимірювання має наступні характеристики:

– Windows 10;
– CPU Intel(R) Core(TM) i5-8250U CPU 1.60GHz;
– 8 GB RAM;
– GPU NVIDIA GeForce MX250;
– 2 GB VRAM.

Роздільна здатність вікна малювання 800x600 пікселів.

Середній час малювання 3D простору (в мілісекундах) в залежності від кількості моделей "dragon" відображений на рис. 5.1.

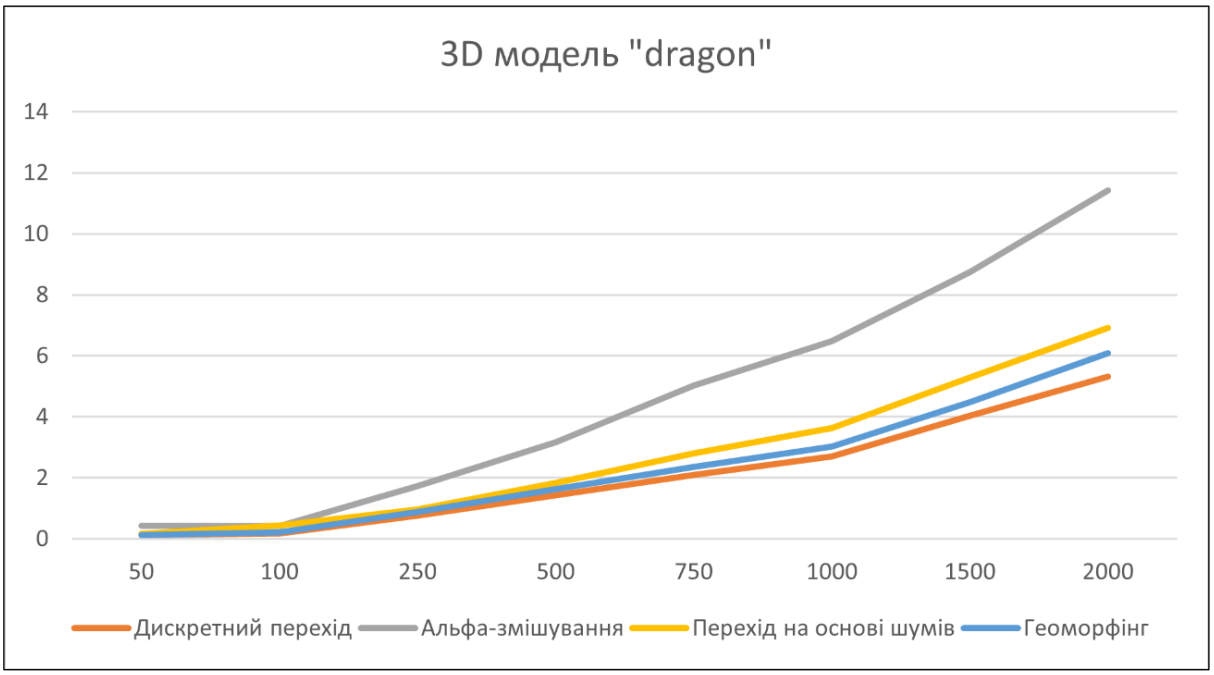

Рисунок 5.1 – Порівняння часу малювання методів плавного переходу для моделі "dragon"

Для невеликої кількості об'єктів (50-100) всі алгоритми генерують кадр більш-менш за однаковий час (менше мілісекунди).

Зі збільшенням навантаження, можна побачити, що альфа-змішування є найповільнішим алгоритмом. Це можна пояснити тим, що алгоритм потребує 4 виклики малювання моделі в окремому від решти сцени буфері.

Змішування за допомогою шумів є набагато швидшим, однак повільнішим за геоморфінг.

Дискретний перехід між рівнями деталізації є найшвидшим, оскільки не потребує ніяких особливих дій.

Така сама поведінка спостерігається і для моделей "stanford-bunny" (див. рис. 5.2) та "car" (рис. 5.3).

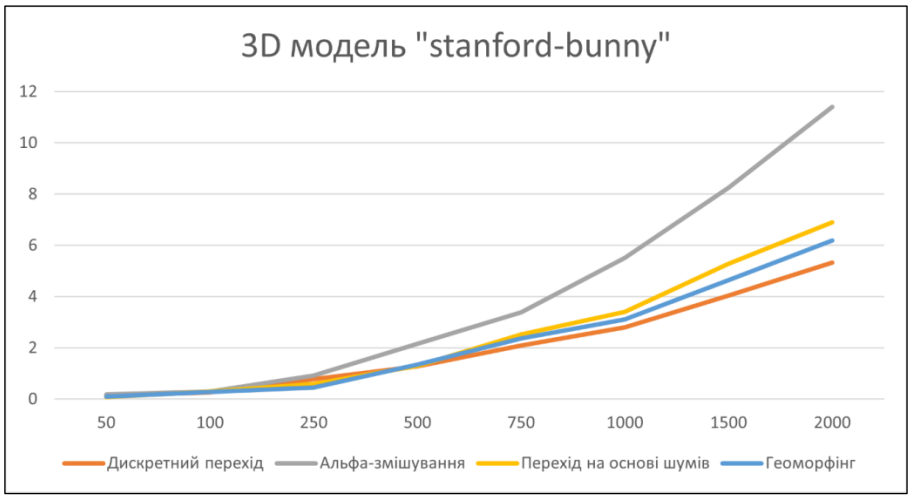

Рисунок 5.2 – Порівняння часу малювання методів плавного переходу для моделі "stanford-bunny"

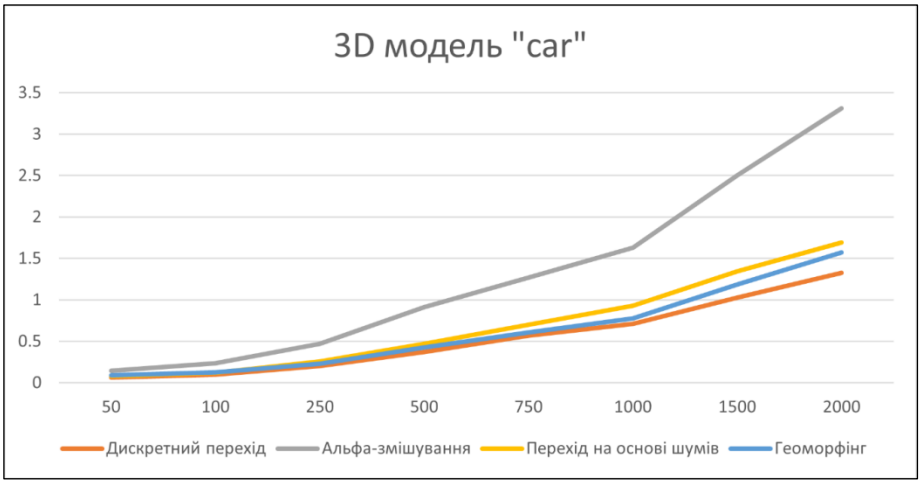

Рисунок 5.3 – Порівняння часу малювання методів плавного переходу для моделі "car"

Час малювання моделей "tree" відображений на рисунку 5.4.

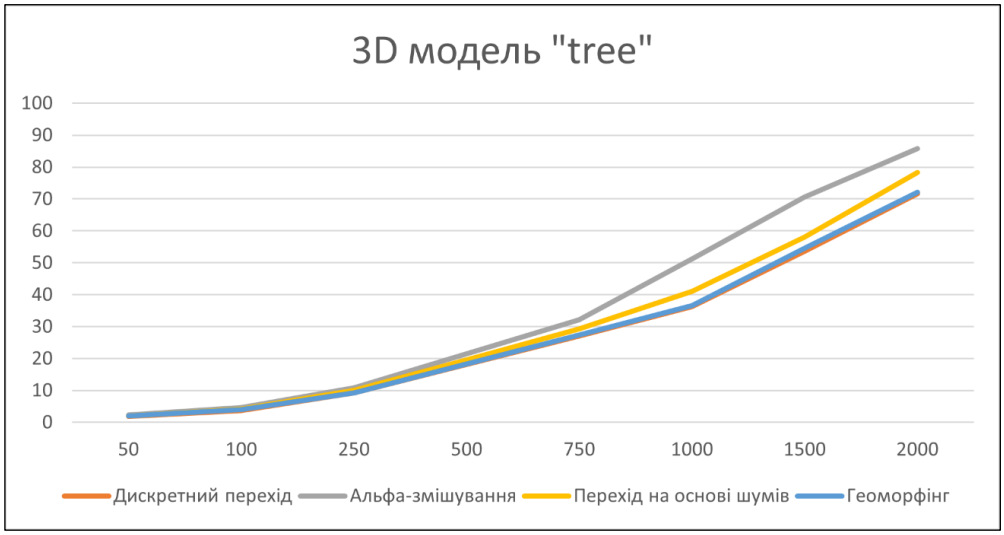

Рисунок 5.4 – Порівняння часу малювання методів плавного переходу для моделі "tree"

Хоча ця модель і має найменшу кільки трикутників, вона містить 54 окремих об'єкти (переважно білборди листви). Через це вона малюється найповільніше, і методи плавного переходу із використанням 1 та 2 викликів функцій малювання (дискретний перехід, геоморфінг та змішування на основі шумів) відрізняються не дуже значно.

Виконаємо аналіз величини ефекту поппінгу для досліджуваних алгоритмів переходу між рівнями деталізації та усереднимо результати для всіх моделей (див. рис 5.5).

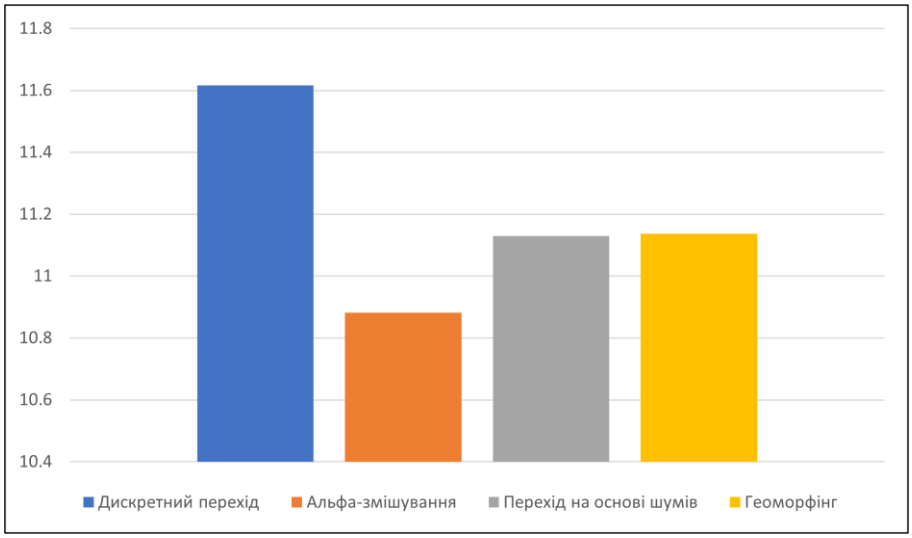

Рисунок 5.5 – Порівняння величини ефекту поппінгу

З рисунку можна побачити, що найбільш плавним методом плавного переходу є альфа-змішування, який має найменше значення поппінгу. Перехід на основі шумів та геоморфінг мають приблизно однакове значення цього показнику.

Порівняємо абсолютні значення поппінгу із дискретним переходом в залежності від відстані до об'єкту для кожної моделі (рис. 5.6, 5.7, 5.8 та 5.9).

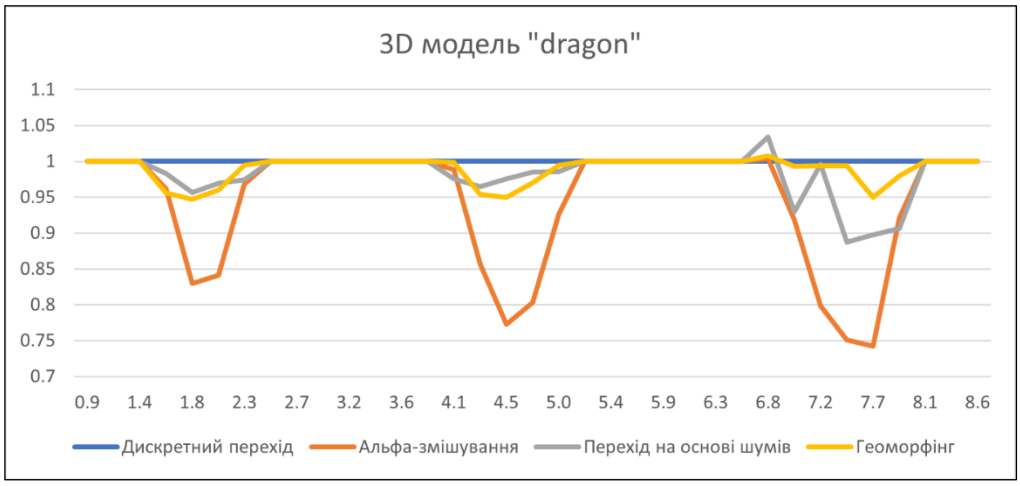

Рисунок 5.6 – Відносні значення поппінгу в залежності від відстані до моделі "dragon"

З рисунку можна чітко побачити 4 рівні деталізації та 3 переходи між ними. Найкращим алгоритмом, що дає найменший "поппінг", є альфа-змішування. Плавність переходу стає більшої з віддалення об'єкту від камери.

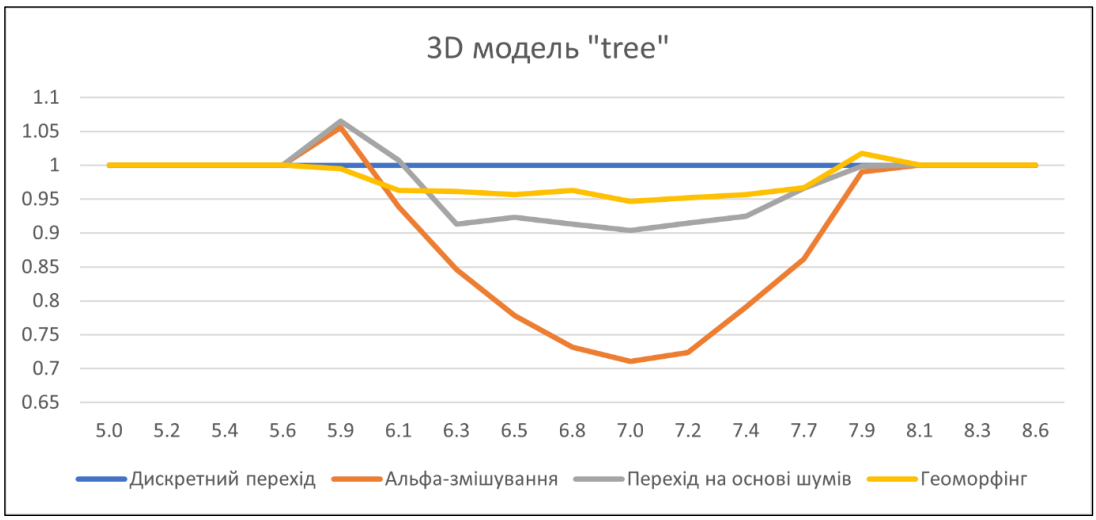

Рисунок 5.7 – Відносні значення поппінгу в залежності від відстані до моделі "tree"

Модель "tree" має лише два рівні деталізації та один перехід. Альфа-змішування все ще є найбільш плавним алгоритмом. На границях переходу можна помітити збільшення поппінгу у порівнянні з дискретним переходом. Цей негативний ефект є менш виразним зі збільшенням відстані.

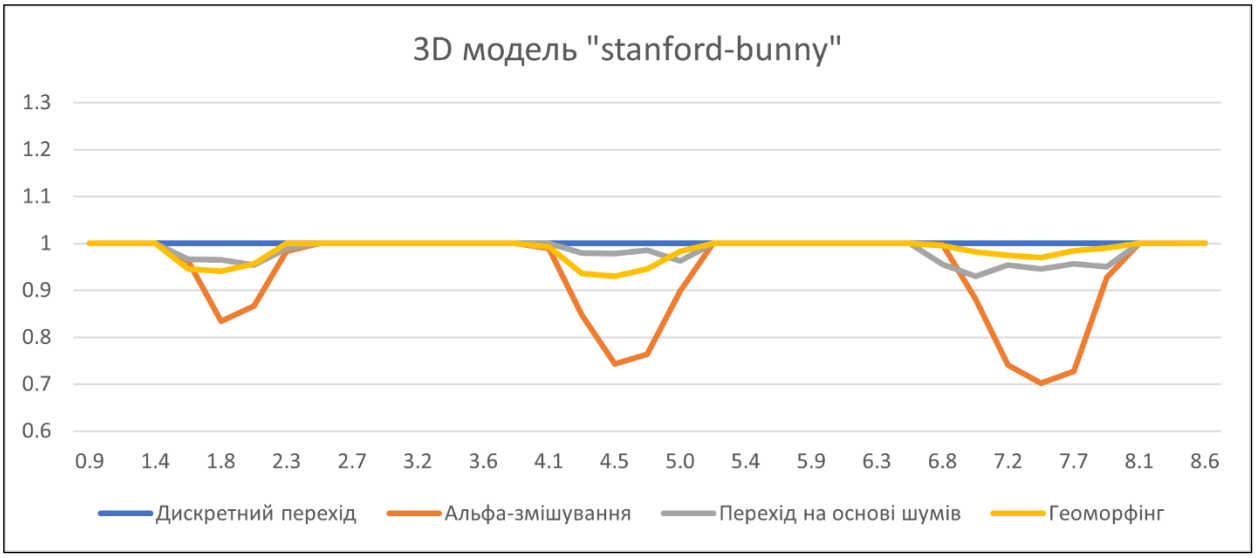

Рисунок 5.8 – Відносні значення поппінгу в залежності від відстані до моделі "stanford-bunny"

Для моделі кролика також найкращим з точку зору плавності є альфа-змішування.

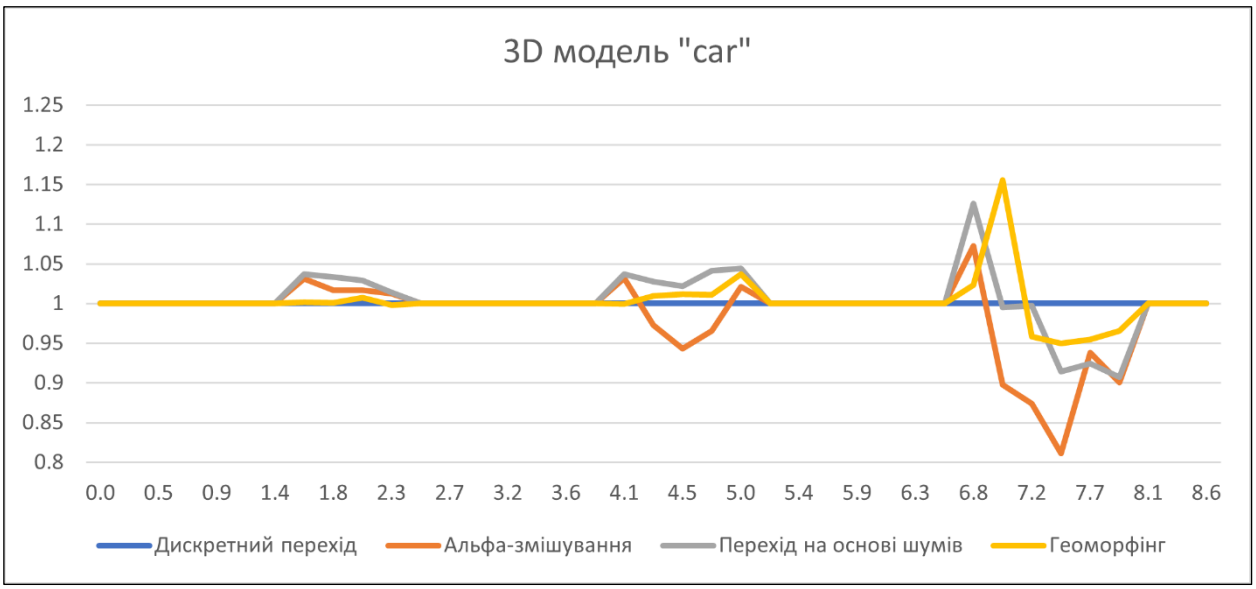

Рисунок 5.9 – Відносні значення поппінгу в залежності від відстані до моделі "car"

Для 3D-моделі автомобіля на ближній дистанції найкращим методом переходу є дискретний, а на середній та дальній – альфа-змішування. Шуми та геоморфінг збільшують значення поппінгу, здійснюючи шкоду послідовності зображення.

Виходячи з отриманих результатів дослідження, можна зробити наступні висновки про доцільність використання методів плавного переходу для досліджуваної 3D моделі:

- альфа-змішування хоч і дає найкращий візуальний вигляд, але цей метод є найповільнішим у 3D просторах з малою та середньою кількістю об'єктів. Плавність переходу, який забезпечує цей метод, збільшується разом із відстанню до об'єкта;
- перехід на основі шумів має середній рівень продуктивності та не потребує великих обчислювальних потужностей для підготовки;
- геоморфінг має приблизно так самі показники, як і змішування на основі шумів, але в ньому є один недолік, що необхідно генерувати 3D меш для плавного переходу.

5.2 Подальше дослідження

Для подальшого дослідження методів плавного переходу можна виділити наступні напрямки діяльності:

- дослідити роботу алгоритмів на більшій кількості 3D моделей різних категорій (архітектура, ландшафт, рослинність, транспорт, предмети інтер'єру, люди тощо);
- розширити метрики для оцінки ефекту поппінгу (відстані в кольоровому просторі CIELAB [15], графічно відобразити поппінг, врахувати статистичну значимість вимірювання);
- дослідити можливість автоматичної рекомендації методів переходу в залежності від типу моделі та кількості об'єктів у 3D просторі для отримання найефективніших значень досліджуваних факторів [16];

– дослідити застосування додаткових візуальних ефектів у комбінації з рівнями деталізації (нормалі, віддзеркалення, відсвічування, трасування променів тощо).

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було виконане дослідження методів плавного переходу між рівнями деталізації (LOD) для тривимірної графіки. Основна увага була приділена знаходженню балансу між продуктивністю та гарним виглядом переходів. Дослідження було зосереджене саме на усуненні "ефекту поппінгу", який порушує візуальну послідовність під час переходів між LOD. Завдяки всебічному аналізу існуючих методів і алгоритмів було отриману цінну інформацію про стратегії запобігання та мінімізації цього ефекту.

Було виявлено, що альфа-змішування, хоч і є найпростішим методом плавного переходу з точки зору реалізації та зменшує ефект поппінгу найсильніше, цей метод є найменш швидким.

Змішування за допомогою шумів має середній рівень продуктивності та достатній рівень візуальної послідовності зображення. Також цей метод дає найбільший рівень творчої свободи при відображенні об'єктів та не потребує попередньої підготовки моделі.

Геоморфінг має схожі властивості продуктивності та послідовності зображення, що і попередній метод, але його відносно важче реалізувати, і він потребує великих розрахунків для підготовки 3D моделі для змішування.

Дискретний перехід все ще має свою сферу застосування: він є найшвидшим та може використовуватися, якщо моделі рівнів деталізації є досить якісними.

Очікується, що на практиці результати цього дослідження допоможуть вибрати алгоритми, які пропонують оптимальний баланс між обчислювальною ефективністю та гарним виглядом із врахуванням різних обставин. Оскільки 3D-візуалізація продовжує набувати популярності в різноманітних програмах, актуальність цієї роботи зберігається, пропонуючи ідеї, які можуть бути використанні в майбутній розробці в цієї галузі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Матвєєв Д.І., Лановий О.Ф. Проблеми оптимізації графіки під пристрої віртуальної реальності. ΛΌГОΣ. ОНЛАЙН, 2020. № 14. URL: https://www.ukrlogos.in.ua/10.11232-2663-4139.14.04.html (дата звернення 16.05.2024).

2. Jim Frost. Root Mean Square Error (RMSE). Statistics by Jim. URL: https://statisticsbyjim.com/regression/root-mean-square-error-rmse/ (дата звернення 03.10.2023).

3. Anton L. Fuhrmann, Eike Umlauf, Stephan Mantler. Extreme Model Simplification for Forest Rendering. ResearchGate. URL: https://www.researchgate.net/publication/221314842_Extreme_Model_Simplification_for_Forest_Rendering (дата звернення 06.10.2023).

4. State of the Art in Procedural Noise Functions. [A. Lagae, S. Lefebvre, R. Cook, DeRose, G. Drettakis, D.S. Ebert, J.P. Lewis, K. Perlin, M. Zwicker]. URL: https://graphics.cs.kuleuven.be/publications/LLCDDELPZ10STARPNF/ (дата звернення 01.12.2023).

5. Noise Functions. Ronja's Shader Tutorials. URL: https://www.ronja-tutorials.com/noise.html (дата звернення 15.01.2024).

6. The Importance of Dithering Technique Revisited with Biomedical Images – A Survey. [Liu Yue, P. Ganesan, B.S. Sathish, C. Manikandan, A. Niranjan, V. Elamaran, Ahmed Faeq Hussein]. ResearchGate. URL: https://www.researchgate.net/publication/329763540_The_Importance_of_Dithering_Technique_Revisited_With_Biomedical_Images-A_Survey (дата звернення 27.11.2023).

7. Nithin Pranesh. Smoother LOD Transitions in Cesium for Unreal with Dithered Opacity Masking. 20.10.2022. Cesium. URL: https://cesium.com/blog/2022/10/20/smoother-lod-transitions-in-cesium-for-unreal/ (дата звернення 04.01.2024).

8. Transparency (or Translucency) Rendering. Nvidia Developer. URL: https://developer.nvidia.com/content/transparency-or-translucency-rendering (дата звернення 05.01.2024).

9. Benny Onrust, Rafael Bidarr, Robert Rooseboom, Johan van de Koppel. Procedural generation and interactive web visualization of natural environments. The 20th International Conference. ResearchGate. URL: https://www.researchgate.net/publication/300490331_Procedural_generation_and_interactive_web_visualization_of_natural_environments (дата звернення 06.01.2024).

10. Hugues Hoppe. Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering. Microsoft Research. URL: https://hhoppe.com/svdlod.pdf (дата звернення 07.01.2024).

11. Pedro V. Sander, Jason L. Mitchell. Progressive Buffers: View-dependent Geometry and Texture LOD Rendering. Advanced Real-Time Rendering in 3D Graphics and Games. URL: https://advances.realtimerendering.com/s2006/Chapter1-Out-of-Core_Rendering_of_Large_Meshes_with_Progressive_Buffers.pdf (дата звернення 04.01.2024).

12. Eun-Seok Lee, Byeong-Seok Shin. Vertex Chunk-Based Object Culling Method for Rreal-Time Rendering in Metaverse. 09.07.2023. MDPI. URL: https://www.mdpi.com/2079-9292/12/12/2601 (дата звернення 11.01.2024).

13. Ревенчук І. Агарков Є. Моделювання доповненої реальності на основі маркерів. Біоніка інтелекту: науково-технічний журнал, 2021. № 96. С. 90–95. URL: https://openarchive.nure.ua/server/api/core/bitstreams/f620de51-d1a2-45f2-adb2-4f3129c7a25d/content (дата звернення 15.05.2024).

14. Data Transfer Matters for GPU Computing. [Yusuke Fujii, Takuya Azumi, Nobuhiko Nishio, Shinpei Kato, Masato Edahiro]. ResearchGate. URL: https://www.researchgate.net/publication/269197419_Data_Transfer_Matters_for_GPU_Computing (дата звернення 14.01.2024).

15. Bao Chau K. Ly, Ethan B. Dyer, Jessica L. Feig, Anna L. Chien, Sandra Del Bino. Research Techniques Made Simple: Cutaneous Colorimetry: A Reliable Technique for Objective Skin Color Measurement. Journal of Investigative

Dermatology. 2020. URL: https://www.researchgate.net/publication/338303610_Research_Techniques_Made_Simple_Cutaneous_Colorimetry_A_Reliable_Technique_for_Objective_Skin_Color_Measurement (дата звернення 14.05.2024).

16. Chalyi S., Leshchynskyi V., Leshchynska I. Method of forming recommendations using temporal constraints in a situation of cyclic cold start of the recommender system. EUREKA: Physics and Engineering. 2019. 4, 34–40. URL: doi: https://doi.org/10.21303/2461-4262.2019.00952 (дата звернення 14.05.2024).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

1. Матвєєв Д.І., Лановий О.Ф. Проблеми оптимізації графіки під пристрої віртуальної реальності. ΛΌГОΣ. ОНЛАЙН, 2020. № 14. URL: https://www.ukrlogos.in.ua/10.11232-2663-4139.14.04.html (дата звернення 16.05.2024).

13. Ревенчук І., Агарков Є. Моделювання доповненої реальності на основі маркерів. Біоніка інтелекту: науково-технічний журнал, 2021. № 96. С. 90–95. URL: https://openarchive.nure.ua/server/api/core/bitstreams/f620de51-d1a2-45f2-adb2-4f3129c7a25d/content (дата звернення 15.05.2024).

16. Chalyi S., Leshchynskyi V., Leshchynska I. Method of forming recommendations using temporal constraints in a situation of cyclic cold start of the recommender system. EUREKA: Physics and Engineering. 2019. 4, 34–40. URL: doi: https://doi.org/10.21303/2461-4262.2019.00952 (дата звернення 14.05.2024).