

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

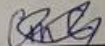
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методів і програмних засобів текстурівання для задач
рендерингу»

Виконав: студент II курсу
групи ІПІ-23м спеціальності
121 – Інженерія програмного забезпечення

 Станіславенко Є.Г.

Керівник: д.т.н., проф., зав. каф. ПЗ

Романюк О. Н.

«06» листопада 2024 р.

Опонент: д.т.н., проф., зав. каф. ЗІ

 Лужецький В.А.

«06» листопада 2024 р.

Допущено до захисту

Зав. кафедри 

«06» листопада 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
18 вересня 2024р.

ЗАВДАННЯ НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Станіславенку Євгенію Григоровичу

1. Тема роботи – «Розробка методів і програмних засобів текстурзування для задач рендерингу»

Керівник роботи: Романюк Олександр Никифорович, д.т.н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від 17 вересня 2024р. № 310.

2. Строк подання студентом роботи 3 грудня 2024р.

3. Вихідні дані до роботи: розмір координатного простору 1280x1024; кольоровий режим – TrueColor; тип тривимірних моделей «.max»; тип полігональної моделі – трикутна; середовище розробки IntelliJ IDEA Community Edition 2022.1; мова розробки Python; операційна система – Windows 10.

4. Зміст текстової частини: вступ; аналіз методів текстурзування; розробка методів текстурзування, які ураховують перспективу; розробка плагіну текстурзування для задач рендерингу; програмна реалізація запропонованих методів текстурзування; економічна частина; висновки; список використаних джерел; додатки; графічна частина.

5. Перелік графічного матеріалу: показ роботи аналогічних програмних продуктів, основні текстурні карти, схема функціонування програмного продукту, ER-діаграма та структурна блок-схема плагіну.

6. Консультанти розділів роботи

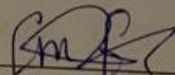
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О. Н., д.т.н., проф., зав. каф. ПЗ	19.09.2024	02.12.2024
5	Причепя І. В., к.е.н., доцент.каф. ЕПВМ	29.10.2024	10.11.2024

7. Дата видачі завдання 19 вересня 2024р.

КАЛЕНДАРНИЙ ПЛАН

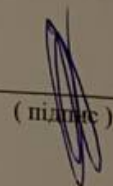
№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів текстуровання	20.09.2024 – 23.09.2024	вик.
2	Розробка методів текстуровання, які ураховують перспективу	23.09.2024 – 13.10.2024	вик.
3	Розробка плагіну текстуровання для задач рендерингу	13.10.2024 – 20.10.2024	вик.
4	Програмна реалізація запропонованих методів текстуровання	20.10.2024 – 29.10.2024	вик.
5	Економічна частина	29.10.2024 – 10.11.2024	вик.
6	Оформлення матеріалів до захисту МКР	28.11.2024 – 30.11.2024	вик.

Студент


(підпис)

Станіславенко Є. Г.
(прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи


(підпис)

Романюк О. Н.
(прізвище та ініціали)

АНОТАЦІЯ

УДК 00492

Станіславенко Є. Г. Розробка методів і програмних засобів текстурзування для задач рендерингу. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 117 с.

На укр. мові. Бібліогр.: 39 назв; рис.: 39; табл.: 9.

Проаналізовано методи та засоби текстурзування тривимірних графічних сцен. Обґрунтовано необхідність підвищення продуктивності накладання текстур для зменшення часу формування зображень.

Доведено, що існуючі методи текстурзування не задовольняють потреби багатьох галузей, що використовують тривимірну комп'ютерну графіку, що обумовило необхідність підвищення продуктивності текстурзування.

Отримано формули для базових перетворень при генерації тривимірних тривимірних об'єктів, що дозволяє їх використовувати для розробки нових методів і засобів. Подальшого розвитку отримав метод текстурзування, відмінність якого від відомих полягає у визначенні координати текселя за картою вагових коефіцієнтів, що дозволяє зчитувати вагові коефіцієнти безпосередньо з текстурної карти, не розраховуючи їх. Встановлено, що застосування вагової карти замість обчислення складних формул підвищує продуктивність до 25%. Вперше запропоновано метод текстурзування, особливість якого полягає у використанні неортогональної растеризації, що дозволяє зменшити $(T-q)$ операцій ділення, де q - кількість горизонтальних рядків, за якими проводиться растеризація трикутника проекції екранного пікселя в текстурний простір, а T - кількість внутрішніх точок трикутника.

Практичне значення отриманих результатів полягає в тому, що на основі отриманих теоретичних положень запропоновано алгоритми та розроблено програмні засоби текстурзування для систем візуалізації тривимірних зображень.

Ключові слова: текстурні карти, 3д-моделі, оптимізація, baking, плагін.

ABSTRACT

UDC: 00492

Stanislavenko E. G. Development of methods and software texturing tools for rendering tasks. Master's qualification work in specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2024. 117 p.

In Ukrainian. Bibliogr.: 39 titles; figures: 39; tables: 9.

In the master's qualification work, a detailed study of the features and methods of texturing 3D models for rendering tasks is carried out. The goal and objectives of increasing the performance and realism of texturing in computer graphics systems are formulated.

The method of analysis for automatic conversion of ready-made material into the “QTexture” format is developed, which simplifies the use of basic textures and provides quick access to editing the parameters of various texture maps, thanks to the developed plug-in. Recommendations for working with texture maps for rendering tasks have been developed.

A method for creating a new texture map “Variety” for rendering tasks has been developed.

A software tool for texturing objects using texture maps that provide a photorealistic look with a small amount of data has been created.

The results obtained in the master's qualification work are used in the development of various 3D graphics tasks. The texturing functions necessary for various 3D models used in different fields have been formed. Such an automated process simplifies the time of model development and its quality and, accordingly, simplifies the time of task execution and improves its performance.

ЗМІСТ

ВСТУП.....	4
1 АНАЛІЗ МЕТОДІВ ТЕКСТУРУВАННЯ.....	10
1.1 Методи рельєфного текстуровання.....	10
1.2 Методи фільтрації текстур.....	16
1.3 Особливості паралакс мепінгу Parallax Mapping.....	19
1.4 Пакети прикладних програм для текстуровання.....	24
1.5 Висновки.....	28
2 РОЗРОБКА МЕТОДІВ ТЕКСТУРУВАННЯ, ЯКІ УРАХОВУЮТЬ ПЕРСПЕКТИВУ	29
2.1 Базові перетворення при генерації тривимірних графічних сцен.....	29
2.2 Метод прискореної анізотропної фільтрації.....	37
2.3 Метод текстуровання з використанням неортогональної растеризації.....	43
2.4 Модифікація методу parallax mapping.....	47
2.5 Висновок.....	51
3 РОЗРОБКА ПЛАГІНУ ТЕКСТУРУВАННЯ ДЛЯ ЗАДАЧ РЕНДЕРИНГУ.....	52
3.1 Призначення та основні функції плагіну для текстуровання під задачі рендерингу.....	52
3.2 Розробка засобів для управління текстурами.....	57
3.3 Розробка процедурних масок та багатошарових текстур.....	61
3.4 Оптимізація та вибікання текстур.....	64
3.5 Висновки.....	78
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНИХ МЕТОДІВ ТЕКСТУРУВАННЯ.....	80
4.1 Програмний модуль для текстуровання тривимірних сцен.....	80
4.2 Реалізація та експериментальні дослідження розроблених методів текстуровання.....	88
4.3 Висновки.....	91
5 ЕКОНОМІЧНА ЧАСТИНА.....	92

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки.....	92
5.2 Розрахунок витрат на проведення науково-дослідної роботи	96
5.2.1 Витрати на оплату праці	96
5.2.2 Відрахування на соціальні заходи	99
5.2.3 Сировина та матеріали.....	99
5.2.4 Розрахунок витрат на комплектуючі.....	100
5.2.5 Спецустаткування для наукових (експериментальних) робіт.....	100
5.2.6 Програмне забезпечення для наукових (експериментальних) робіт..	101
5.2.7 Амортизація обладнання, програмних засобів та приміщень.....	102
5.2.8 Паливо та енергія для науково-виробничих цілей.....	103
5.2.9 Службові відрядження.....	104
5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації.....	105
5.2.11 Інші витрати	105
5.2.12 Накладні (загальновиробничі) витрати	106
5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором.....	107
5.4 Висновки.....	111
ВИСНОВКИ	112
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	113
ДОДАТКИ	118
ДОДАТОК А. Технічне завдання.....	119
ДОДАТОК Б. Протокол перевірки МКР на плагіат	122
ДОДАТОК В. Лістинг програми.....	123
ДОДАТОК Г. Ілюстративна частина	150

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний етап розвитку комп'ютерних систем відзначається інтенсивним застосуванням комп'ютерної візуалізації, що є потужним інструментом для забезпечення ефективної двосторонньої взаємодії між комп'ютером та користувачем.. Оскільки потужність графічних засобів дозволяє досягати високого рівня реалістичності при відтворенні тривимірних об'єктів, вимоги до складності моделей і точності відображення деталізованих поверхонь постійно зростають.

Високу ефективність візуалізації в комп'ютерних системах забезпечують методи та інструменти, які дозволяють оптимально використовувати обчислювальні ресурси графічної системи для досягнення високої реалістичності графічних зображень. Одним із таких методів є текстуркування, що дає змогу враховувати локальні особливості поверхні без значного ускладнення геометрії моделі. Процес накладання текстур ефективно використовується для відтворення кольорів, фактур, нерівностей, імітації відбиття світла та інших візуальних характеристик на поверхні об'єктів.

Текстуркування є одним із найскладніших етапів кінцевої візуалізації, оскільки кожна точка на поверхні об'єкта повинна бути точно відображена в дискретному просторі, а також визначені її кольорові характеристики з урахуванням різних умов освітлення. Це включає в себе детальне визначення того, як текстура буде накладена на поверхню, враховуючи ефекти освітлення, тіні та відбиття. У сучасних методах текстуркування для точного визначення кольору пікселя використовується інформація не тільки з одного текселя, а й з декількох, що дозволяє досягти більш високої реалістичності зображення.

При накладанні текстур для кожного пікселя враховуються декілька текселів, що впливає на час рендерингу. Це створює певні обмеження для застосування цих методів у реальному часі, зокрема в динамічних або інтерактивних режимах, таких як відеоігри або симуляції, де необхідно забезпечити швидку обробку даних для підтримки високої продуктивності.

Таким чином, при розробці графічних систем важливо досягти балансу між якістю візуалізації і вимогами до швидкості обчислень, щоб забезпечити реалістичні, але водночас ефективні графічні сцени.

Для досягнення високої реалістичності при формуванні графічних сцен використовують не тільки основні кольорові текстурні карти, а й додаткові карти, такі як карти висот, нормалей, відстаней та інші, які дозволяють детально відтворювати рельєфні особливості поверхонь. Завдяки таким допоміжним картам можна точніше імітувати складні елементи ландшафтів, поверхневі нерівності, глибину і інші деталі, які значно підвищують рівень візуалізації.

Таким чином, сучасні методи високоточного накладання текстур мають значну обчислювальну складність, що істотно впливає на час рендерингу графічних зображень. З огляду на це, питання підвищення продуктивності методів і засобів текстуровання стають особливо актуальними. Точність визначення кольорів пікселів на етапі текстуровання є ключовим чинником для досягнення високої реалістичності вихідного зображення, що передбачає розробку нових, більш ефективних методів накладання текстур.

Для підвищення реалістичності відтворення зображень важливим є врахування положення спостерігача, що дозволить точніше відтворювати перспективу сцени. Тому розробка методів текстуровання, які б ураховували такі аспекти, є важливим напрямком досліджень.

Особливо складні вимоги до продуктивності виникають при формуванні високореалістичних графічних зображень у реальному часі та в інтерактивному режимі. Для досягнення необхідної якості зображення використання простих методів накладання текстур є неприпустимим.

Таким чином, питання підвищення продуктивності та реалістичності формування графічних сцен є актуальними, оскільки існуючі методи текстуровання не задовольняють потреби багатьох галузей, що використовують тривимірну комп'ютерну графіку. Це ставить завдання розробки нових, більш ефективних методів і засобів текстуровання, які забезпечать високий рівень деталізації при мінімальних витратах обчислювальних ресурсів.

Зв'язок роботи з науковими програмами, планами, темами. Робота проводилася відповідно до затвердженого плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета і завдання дослідження Метою роботи є підвищення продуктивності текстуровання шляхом зменшення обчислювальної складності, а також покращення реалістичності зображень через більш точне визначення кольорів пікселів.

Головними задачами дослідження є:

- провести аналіз існуючих методів і засобів накладання текстур для визначення напрямків підвищення їх продуктивності та реалістичності;
- запропонувати нові методи підвищення продуктивності текстуровання;
- розробити програмні засоби текстуровання на основі запропонованих методів;
- провести експериментальні дослідження розроблених засобів текстуровання.

Об'єкт дослідження – процес текстуровання тривимірних об'єктів у системах комп'ютерної графіки.

Предмет дослідження – методи та засоби текстуровання тривимірних графічних графічних сцен.

Методи дослідження. У процесі дослідження використовувалися різноманітні математичні та комп'ютерні методи. Зокрема, застосовувалися теорія чисел та чисельних методів для розв'язання обчислювальних задач, теорія диференціально-інтегрального числення для моделювання та аналізу динамічних процесів у текстурованні, лінійна алгебра для роботи з матрицями трансформацій та геометричними об'єктами, а також методи аналітичної геометрії для розробки моделей тривимірних об'єктів і візуальних ефектів. Крім того, для аналізу та перевірки отриманих теоретичних результатів було використано комп'ютерне моделювання, що дозволило перевірити практичну реалізацію методів текстуровання та їх ефективність у реальних умовах.

Наукова новизна отриманих результатів:

1. Отримано формули для базових перетворень при генерації тривимірних тривимірних об'єктів, що дозволяє їх використовувати для розробки нових методів і засобів.

2. Подальшого розвитку отримав метод текстурзування, відмінність якого від відомих полягає у визначенні координати текселя за картою вагових коефіцієнтів, що дозволяє зчитувати вагові коефіцієнти безпосередньо з текстурної карти, не розраховуючи їх. Встановлено, що застосування вагової карти замість обчислення складних формул підвищує продуктивність до 25%.

3. Вперше запропоновано метод текстурзування, особливість якого полягає у використанні неортогональної растеризації, що дозволяє зменшити $(T-q)$ операцій ділення, де q - кількість горизонтальних рядків, за якими проводиться растеризація трикутника проекції екранного пікселя в текстурний простір, а T - кількість внутрішніх точок трикутника.

Практичне значення одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби текстурзування для систем візуалізації тривимірних зображень.

Особистий внесок здобувача. Усі наукові результати, викладені у МКР, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: [1] – запропонував адаптивно використовувати методи зафарбування залежно від кривизни поверхні; [2] – розглянув особливості шейдерного рендерингу; [3] – провів аналіз моделей для дзеркального відбиття; [4] – проаналізовано етап текстурзування при розробці персонажів; [5] – провів аналіз новиз BRDF; [6] – проаналізував особливості DirectX 12 для задач рендерингу; [7] – проаналізував нововведення в програмному забезпеченні «Zbrush»; [8] – розглянув особливості використання програмного пакета Substance Painter для розробки 3Д моделей; [9] – проаналізував шейдерну технологію зафарбовування; [10] – проаналізовано процедуру формування зображень з використанням текстурних карт; [11] – розглянуто редактор для веб-

розробників; [12] – проаналізував нововведення в програмному забезпеченні «Blender» для задач текстурювання; [13] – встановив взаємозв’язок між векторами нормалі до поверхні, вектором спостерігача та вектором джерела світла для задач рендерингу; [14] – доведено властивість про сталість приросту інтенсивності; [15] – проаналізував моделі відбивних здатностей поверхні на основі BRDF; [16] – розглянуто особливості використання програмного забезпечення Mari; [17] – розглянуто особливості фізично-коректного рендерингу; [18] – розглянуто використання програмного забезпечення Mari в галузі 3д-моделюванні; [19] – запропоновано алгоритм розбиття тривимірної моделі на матрицю точок; [20] – розглянув особливості використання методу Дисплейсмент-мапінг; [21] – проаналізував методи надання фактурності з використанням векторів; [22] – проаналізував нові методи текстурювання.

Апробація результатів магістерської кваліфікаційної роботи. Основні положення МКР доповідалися та обговорювалися на Міжнародних і Всеукраїнських конференціях: Міжнародної науково-практичної конференції «Використання інформаційних та комунікаційних технологій в сучасному цифровому суспільстві» (м. Херсон, 2020), XII Міжнародної науково-технічної конференції «Інформаційно-комп’ютерні технології – 2021 (ІКТ-2021)» (м. Житомир, 2021), XIV міжнародної науково-практичної конференції «Інформаційні технології і автоматизація» (м. Одеса, 2021), Міжнародної науково-практичної інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» (м. Суми/Вінниця, 2021, 2023, 2024), VI Міжнародної науково-практичної конференції «Сучасна наука: проблеми та перспективи (частина II)» (м. Київ, 2022), Науково-технічні конференції факультету інформаційних технологій та комп’ютерні інженерії (м. Вінниця, 2022, 2024), II Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Комп’ютерні ігри та мультимедіа як інноваційний підхід до комунікації» (м. Одеса, 2022), The 9 th International scientific and practical conference “Modern research in world science «Sci-conf.com.ua» (м. Львів, 2022), Матеріали XXII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів

та студентів «Стан, досягнення та перспективи інформаційних систем і технологій» (м. Одеса, 2022), The 8 th International scientific and practical conference «Topical issues of modern science, society and education» (м. Харків, 2022), V Міжнародної науково-практичної конференції «Актуальні проблеми сучасної науки та освіти (частина I)» (м. Львів, 2022), VI Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених «Сучасні інформаційні системи та технології» за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» (м. Херсон, 2023), XVI міжнародної науково-практичної конференції «Інформаційні технології і автоматизація - 2023» (м. Одеса, 2023), Всеукраїнської науково-практичної інтернет- конференції «Молодь в науці: дослідження, проблеми, перспективи (2023, 2024, м. Вінниця), LIII науково-технічної конференції підрозділів ВНТУ (м. Вінниця, 2024), V Всеукраїнська науково-практична інтернет конференція (м. Миколаїв, 2024).

Публікації. За результатами виконаних досліджень опубліковано 23 наукові праці у збірниках матеріалів Міжнародних і Всеукраїнських конференцій.

1 АНАЛІЗ МЕТОДІВ ТЕКСТУРУВАННЯ

1.1 Методи рельєфного текстуровання

Рельєфне текстуровання [7-10],[16],[20],[21] (рис. 1.1) в комп'ютерній графіці використовується для створення ефекту об'ємності та детальності на поверхнях 3D-об'єктів.



Рисунок 1.1 – Приклад рельєфного текстуровання

Бамп-мапінг [3],[36-38] — це техніка, що надає поверхні ілюзію рельєфу за допомогою текстури, яка модифікує нормалі пікселів. Замість того, щоб змінювати геометрію об'єкта, бамп-мапінг змінює освітлення на поверхні на основі векторів нормалей. Ця техніка легко реалізується і не вимагає великої обчислювальної потужності, дозволяючи додати деталі без значних змін в моделі. Однак вона не забезпечує фізичної зміни поверхні, тому не може коректно відображати тіні та відбиття, підходить для створення поверхонь з дрібними деталями, такими як шкіра, тканина або грубі поверхні (рис. 1.2).

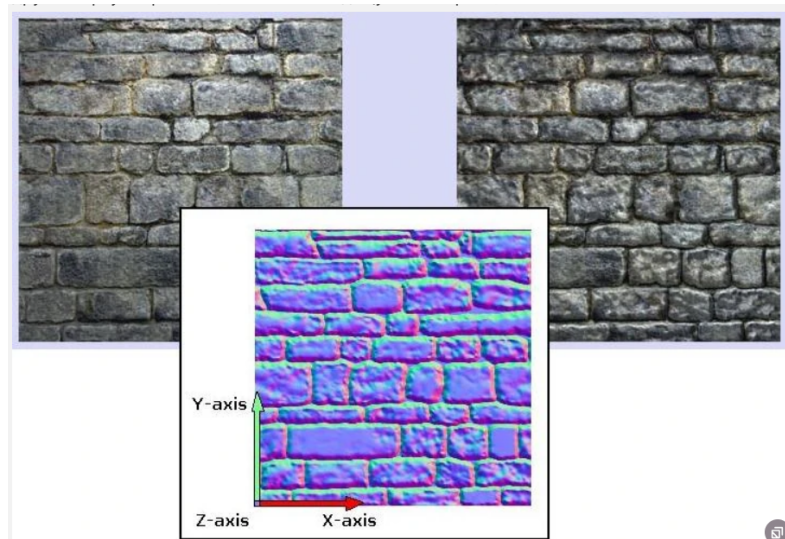


Рисунок 1.2 – Приклад бамп-мепінгу

Нормал-мапінг [3],[7],[8] є удосконаленою формою бамп-мапінгу, що використовує текстуру, яка містить вектори нормалей для кожного пікселя. Це дозволяє створити ще детальніший рельєф, краще передаючи світлотіньові ефекти і роблячи зображення більш реалістичним. Проте він вимагає більше пам'яті для зберігання нормалей і займає більше часу на обчислення в порівнянні з бамп-мапінгом. Нормал-мапінг широко використовується в сучасних іграх для покращення візуального сприйняття поверхонь.

Дисплейсмент-мапінг [20] змінює геометрію об'єкта на основі значень у текстурі, переміщуючи вершини поверхні. Це дозволяє створювати фізично коректні рельєфи, проте вимагає значних обчислювальних ресурсів і може ускладнити геометрію. Дисплейсмент-мапінг використовується в сценах з високим рівнем деталізації, таких як архітектурна візуалізація та анімації. Паралаксове текстуровання поліпшує бамп-мапінг, враховуючи глибину текстури, що збільшує реалістичність зображення, створюючи ефект глибини, що змінюється при русі камери. Це використовується для рельєфних поверхонь, де важлива глибина, наприклад, на дорогах або тріщинах (рис. 1.3).

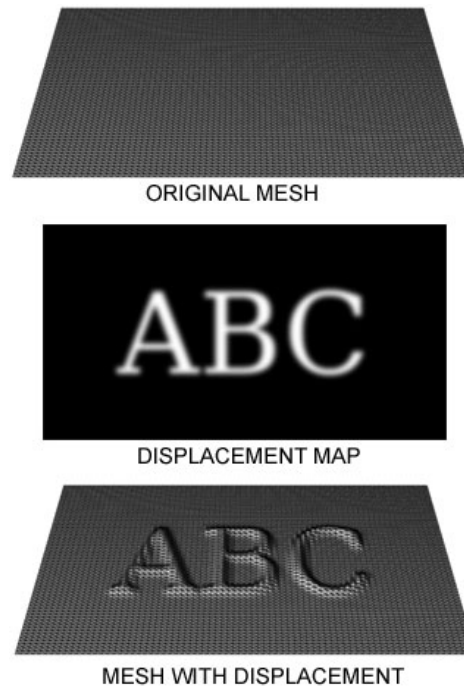


Рисунок 1.3 – Приклад дисплейсмент-мапінг

Статичні тіні [3] створюються заздалегідь і накладаються на об'єкти, тоді як динамічні тіні генеруються в реальному часі і можуть змінюватися в залежності від освітлення та положення об'єкта. Це підвищує реалістичність сцени, підкреслюючи рельєф і деталі поверхні, проте динамічні тіні вимагають значних обчислювальних ресурсів. Фільтрація текстур допомагає згладити текстури при зміні масштабу об'єкта, покращуючи візуальну якість текстур на об'єктах з різними кутами зору. Основні методи фільтрації включають білінійну та трилінійну фільтрацію, які зменшують артефакти при масштабуванні.

Текстури об'ємності використовують об'ємні дані для створення рельєфу, що корисно для ефектів, таких як дим або туман. Вони взаємодіють з освітленням, створюючи динамічні та реалістичні візуалізації, але вимагають великих обсягів пам'яті та обчислювальних ресурсів. Методи рельєфного текстуровання [7-10] відіграють важливу роль у створенні реалістичних 3D-сцен і об'єктів, а вибір методу залежить від специфіки проекту, вимог до якості зображення та доступних ресурсів. Комбінація цих методів може значно

підвищити рівень реалістичності графіки.

Рельєфне текстуровання [10] реалізується у відеокартах за допомогою апаратного прискорення, що дозволяє значно підвищити продуктивність при обробці графіки. Відеокарти використовують спеціалізовані обчислювальні одиниці, відомі як шейдери, які виконують складні обчислення, пов'язані з рельєфним текстурованням. Наприклад, у випадку бамп-мапінгу та нормал-мапінгу, шейдери змінюють нормалі пікселів на основі текстурних координат, що дозволяє створити ефект рельєфу без фізичної зміни геометрії об'єкта. Відеокарти також реалізують алгоритми для фільтрації текстур, такі як білінійна і трилінійна фільтрація, які забезпечують плавний перехід між пікселями при зміні масштабу.

При дисплейсмент-мапінгу [9],[20] відеокарти можуть змінювати вершини геометрії в реальному часі, виконуючи обчислення для кожної вершини на основі значень у дисплейсмент-текстурі. Це потребує великої обчислювальної потужності, тому нові покоління відеокарт оснащені потужнішими GPU та більшою пам'яттю, що дозволяє виконувати ці складні обчислення ефективніше. Додатково, сучасні відеокарти підтримують технології, такі як Tessellation [11], що дозволяє підвищувати рівень деталізації моделей шляхом динамічного розбиття геометрії на більш дрібні елементи.

Відеокарти також використовують буфери, такі як текстурні буфери, для зберігання текстур і нормалей, а також шейдерні програми, які виконують специфічні обчислення для кожного з пікселів або вершин. Це дозволяє створити складні візуальні ефекти, поєднуючи різні методи рельєфного текстуровання для досягнення максимальної реалістичності. Додатково, підтримка API, таких як DirectX і OpenGL [30], забезпечує інтеграцію цих технологій у відеоігри та графічні додатки, дозволяючи розробникам використовувати рельєфне текстуровання у своїх проектах. Таким чином, відеокарти стають важливим інструментом у створенні високоякісної графіки в реальному часі.

Перспективи розвитку рельєфного текстуровання в комп'ютерній графіці обіцяють багато цікавих інновацій, які можуть суттєво підвищити реалістичність

зображень та взаємодію користувачів з віртуальними світами. Однією з основних тенденцій є інтеграція технологій штучного інтелекту та машинного навчання для автоматизації процесів створення текстур і рельєфів, що дозволить знижувати витрати часу на розробку та підвищувати якість фінального продукту. Наприклад, алгоритми глибокого навчання можуть генерувати текстури на основі простих ескізів або образів, створюючи реалістичні поверхні без потреби в ручній роботі.

Іншою перспективою є розвиток технологій віртуальної та доповненої реальності, які вимагають високого рівня деталізації та реалістичності в текстурованні для створення переконливих візуальних ефектів. Це сприятиме вдосконаленню методів рельєфного текстуровання [7-10], таких як дисплейсмент-мапінг, щоб вони могли працювати в реальному часі з мінімальними затратами на обчислення.

Зі збільшенням потужності графічних процесорів та оптимізацією шейдерних технологій можна очікувати, що рельєфне текстуровання стане більш доступним для розробників, дозволяючи використовувати більш складні алгоритми та ефекти без значних втрат продуктивності. Крім того, зростаюча популярність технологій, таких як фізично коректне рендерингування (PBR) [26],[33] спонукатиме до подальшого вдосконалення методів рельєфного текстуровання, щоб забезпечити ще більшу точність в передачі матеріалів та їхнього взаємодії зі світлом.

Важливим напрямком також є розвиток обчислювальної графіки для мобільних пристроїв, оскільки все більше ігор та додатків переходять на мобільні платформи. Це вимагатиме адаптації рельєфного текстуровання для роботи в обмежених умовах, зокрема оптимізації алгоритмів для досягнення високої продуктивності без втрати якості.

Інтеграція рельєфного текстуровання в нові формати контенту, такі як 3D-друк і інтерфейси користувача, відкриває нові можливості для його використання. З появою нових технологій та інструментів, текстуровання продовжуватиме еволюціонувати, залишаючись важливим елементом у

створенні візуально привабливих і реалістичних 3D-сцен.

Адаптивне текстурзування [10] є можливим і стає все більш популярним у комп'ютерній графіці. Ця технологія дозволяє оптимізувати використання текстур залежно від умов рендерингу, таких як відстань до камери, кут зору, освітлення та інші фактори. Основні принципи та можливості адаптивного текстурзування включають:

У ситуаціях, коли об'єкти знаходяться далеко від камери, можна використовувати текстури з нижчою роздільною здатністю, що дозволяє заощаджувати пам'ять і обчислювальні ресурси. Коли об'єкти наближаються, система може автоматично Метод Level of Detail використовує кілька версій однієї й тієї ж текстури з різними рівнями деталізації. На основі відстані до камери або кута зору автоматично вибирається найбільш підходящий рівень деталізації. Це знижує навантаження на графічний Адаптивне текстурзування часто реалізується за допомогою шейдерів, які дозволяють виконувати обчислення в реальному часі. Наприклад, можна реалізувати шейдер, який змінює текстуру залежно від певних умов, таких як освітлення або геометрія сцени.

Текстури можуть адаптуватися до геометрії об'єкта. Наприклад, рельєфні деталі можуть змінюватися в залежно від кута огляду, забезпечуючи більш природне відображення об'єктів у сцені.

Процедурні текстури можуть бути використані в адаптивному текстурзуванні для генерації рельєфів у реальному часі на основі алгоритмів, що реагують на навколишні умови. Це може зменшити потребу в зберіганні великих обсягів текстурних даних.

Адаптивне текстурзування допомагає оптимізувати ресурси при створенні графіки, підвищуючи продуктивність і зменшуючи навантаження на системи. Це особливо важливо в умовах, коли потрібна висока якість зображення, але ресурси обмежені, наприклад, у відеоіграх або віртуальних середовищах.

1.2 Методи фільтрації текстур

Методи фільтрації текстур використовуються у комп'ютерній графіці для покращення візуальної якості текстур на поверхнях, особливо коли текстура відображається під кутом або на великій відстані.

Точкова фільтрація це найпростіший та найшвидший метод, який вибирає тексель, що найближчий до точки вибірки текстури. Часто призводить до зернистого та блокованого зображення, особливо при великих змінах масштабу.

Білінійна фільтрація текстур [33] — це метод інтерполяції, який використовується для згладжування зображень під час текстуровання у 3D-графіці. Цей метод працює шляхом взяття чотирьох найближчих текселів (пікселів текстури) до заданої точки зображення та обчислення середнього значення з ваговими коефіцієнтами, які залежать від відстані від точки вибірки до текселів. Білінійна фільтрація ефективно зменшує зубці та покращує візуальну якість текстур, особливо коли вони переглядаються під кутом або віддалено (рис. 1.4).

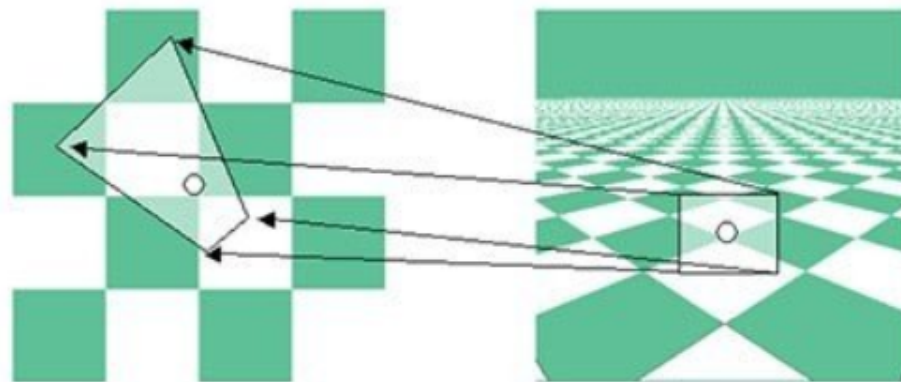


Рисунок 1.4 – Приклад виконання білінійної інтерполяції

Розглянемо основні кроки, які відбуваються під час білінійної фільтрації:

1. Знаходження чотирьох текселів, які знаходяться найближче до точки перегляду на текстурі.

2. Виконання лінійної інтерполяції двічі у горизонтальному напрямку (зазвичай вліво та вправо від точки вибірки).

3. Виконання лінійної інтерполяції результатів горизонтальної інтерполяції у вертикальному напрямку (зазвичай вгору та вниз від точки вибірки).

Білінійна фільтрація [10] є компромісом між швидкістю обчислень та якістю відображення, оскільки вона значно швидша за бікубічну фільтрацію, але забезпечує менш гладке зображення у порівнянні з вищими методами інтерполяції.

Трилінійна фільтрація поєднує дві ключові операції: білінейну фільтрацію в площині (на основі сусідніх пікселів) та лінійну фільтрацію по рівням текстур (по осі глибини). Для кожного пікселя, який потрібно відобразити, беруться до уваги два найближчі рівні текстури, і для кожного з цих рівнів застосовується білінейна фільтрація (рис.1.5).

Трилінійна фільтрація забезпечує плавні переходи між текстурами, що особливо важливо при зміні відстані до об'єктів у сцені.



Рисунок 1.5 – Переходи між текстурами при трилінійній фільтрації

Метод зменшує ефект «блоків» та артефактів, що часто виникають при використанні базових методів.

Хоча трилінійна фільтрація є більш обчислювально інтенсивною порівняно з білінейною фільтрацією, вона забезпечує кращу якість зображення,

що робить її корисною у реальному часі для комп'ютерних ігор та візуалізації. Залежно від апаратних ресурсів, трилінійна фільтрація може вимагати більше пам'яті і процесорної потужності.

Трилінійна фільтрація [9] часто поєднується з іншими техніками обробки текстур, такими як *mipmapping*, для подальшого покращення якості при віддалених об'єктах.

Трилінійна фільтрація в деяких випадках може призводити до втрати деталей, особливо якщо текстура має багато деталей або малюнків. Залежно від контексту, можуть бути альтернативні методи, такі як анізотропна фільтрація, які можуть надати кращі результати для певних типів зображень.

Анізотропна фільтрація [9] — це метод обробки текстур, який використовується для покращення якості зображень, особливо коли об'єкти знаходяться під різними кутами або на різних відстанях від камери. Цей метод є більш складним і обчислювально витратним, ніж трилінійна фільтрація, але він забезпечує значно кращі результати в певних умовах (рис. 1.6).



Рисунок 1.6 – Приклад виконання анізотропної фільтрації

Анізотропна фільтрація аналізує напрямок сприйняття текстури, що дозволяє фільтрувати текстури по-різному в залежності від кута огляду. Вона враховує не тільки сусідні пікселі, але й відстань до них, щоб зберегти деталі текстури при огляді з гострих кутів. Анізотропна фільтрація значно покращує якість зображення, зберігаючи деталі на похилих поверхнях і зменшуючи розмитість текстур. Вона забезпечує більш реалістичний вигляд об'єктів, особливо в 3D-сценах, де присутні віддалені або похилі поверхні.

Хоча анізотропна фільтрація забезпечує кращу якість, вона є більш вимогливою до ресурсів. Вона потребує більше обчислювальної потужності та пам'яті порівняно з білінійною і трилінійною фільтрацією.

Різні рівні анізотропної фільтрації (наприклад, 2x, 4x, 8x, 16x) вказують на те, скільки напрямків текстури буде оброблено, і чим вище значення, тим кращою буде якість зображення.

Використовується в комп'ютерних іграх, візуалізації архітектурних проектів, а також у будь-яких графічних програмах, де важлива деталізація текстур.

Анізотропна фільтрація зазвичай комбінується з іншими методами обробки текстур для досягнення оптимальних результатів.

З-за підвищених вимог до ресурсів, не завжди можливо використовувати анізотропну фільтрацію в реальному часі, особливо на старих або менш потужних пристроях.

Вона може бути надмірною для простих сцен або текстур, де не потрібно зберігати багато деталей.

1.3 Особливості паралакс мепінгу Parallax Mapping

Parallax Mapping [20],[38] – це метод текстуровання в комп'ютерній графіці, який використовується для створення ілюзії глибини та рельєфності на плоских поверхнях без збільшення кількості полігонів. Цей метод ґрунтується на динамічній зміні текстурних координат пікселів залежно від кута огляду, що дає

змогу створити реалістичне враження про тривимірні деталі поверхні. Parallax Mapping, як метод обчислення текстурного рендерингу, має кілька видів, кожен з яких має свої особливості та застосування.

Steep Parallax Mapping – вдосконалена версія стандартного Parallax Mapping, яка враховує більш круті кути огляду. Використовується для обробки випадків, коли поверхня має сильні нерівності. Гостре паралаксне відображення покращує точність візуалізації, зменшуючи артефакти, пов’язані з видимістю.

Parallax Occlusion Mapping [20],[38] – є найбільш досконалим і дозволяє враховувати не лише зміщення текстур, але й ефект оклюзії. Метод здійснює трасування променів через карту висот, що дозволяє більш точно відображати деталі, такі як виїмки або підвищення на поверхні. Relief Mapping є ще одним варіантом, який дозволяє створювати ще більше деталей на поверхнях. Він працює за принципом виявлення геометрії через карту висот і може включати такі функції, як рендеринг нормалей (рис.1.7).

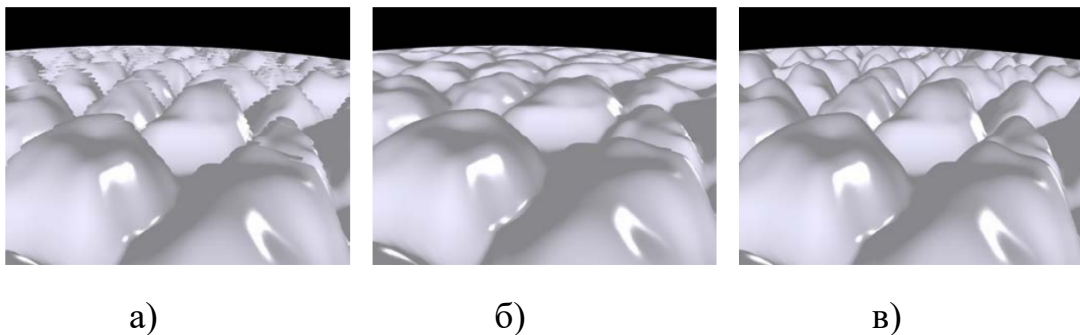


Рисунок 1.7 – Види Parallax Mapping

а) Steep б) Occlusion в) Relief

Принцип роботи Parallax Mapping базується на використанні карти висот для зміщення текстурних координат на поверхні, що дозволяє створити ілюзію глибини та рельєфу на плоских полігональних поверхнях. Основна ідея методу полягає в обчисленні зміщення кожного пікселя поверхні, залежно від кута огляду та висоти на карті висот, що дозволяє точно передавати нерівності

поверхні при різних ракурсах.

Для початку виконується обчислення зміщення паралакса для кожного пікселя. Це зміщення визначається на основі напрямку спостереження та нормалі поверхні, яка є геометричною орієнтацією полігону. Ключовий аспект полягає в тому, що ці обчислення проводяться в дотичному просторі — системі координат, прив'язаній до поверхні, яка забезпечує коректність рендерингу незалежно від кута огляду. Дотичний простір дозволяє ефективно обчислювати зміщення текстур для довільних полігональних поверхонь і знижує обчислювальні витрати завдяки тому, що всі операції виконуються відносно локальної площини поверхні. Вектор зміщення паралакса визначається з урахуванням напрямку погляду та нормалі до поверхні N :

$$P = \left(\frac{h_d}{h_{max}} \right) * \widehat{V_{ts}} \quad (1.1)$$

де h_d — це висота на карті висот у певній точці, а h_{max} — максимальна висота на карті, яка використовується для нормалізації.

Наступним кроком є трасування променя через поле висот для визначення найближчої видимої точки на поверхні. Цей процес виконується за допомогою лінійного або бінарного пошуку, що дозволяє знайти точку перетину променя огляду з профілем висот. Лінійний пошук дозволяє знайти точку нижче перетину променя з поверхнею, тоді як бінарний пошук уточнює це перетинання з більшою точністю. Для кожної вибірки вздовж вектора зміщення паралакса P , ми обчислюємо висоту на карті висот $h(x, y)$ і порівнюємо її з поточним значенням висоти променя h_r :

$$h_r = h_{start} + k * \Delta h \quad (1.2)$$

де h_{start} — початкова висота на карті, Δh — крок висоти вздовж вектора пошуку, а k — це індекс поточної ітерації пошуку.

Коли точка перетину знайдена, обчислюється зміщення координат текстури, яке використовується для вибірки потрібної текстури на екструдованій поверхні. В результаті, кожен піксель поверхні отримує своє індивідуальне зміщення, яке відповідає його рельєфу та глибині на карті висот. Це дозволяє отримати реалістичне зображення поверхонь, що включає в себе як глибокі нерівності, так і плавні зміни рельєфу, що залежить від ракурсу спостереження. Нові координати текстури t_{off} розраховуються наступним чином:

$$t_{off} = t_{orig} + P \quad (1.3)$$

Однією з важливих особливостей Parallax Mapping є можливість обчислення ефектів самозатінення та м'яких тіней. Після обчислення зміщених текстурних координат виконується трасування видимості, щоб визначити наявність загороджувальних елементів на поверхні. Це дозволяє додати ефекти затінення, що підвищують реалістичність рендерингу. Частота дискретизації δ може бути виражена такою функцією:

$$\delta = n_{min} + (n_{max} - n_{min}) * (1 - |N * \widehat{V_{ts}}|) \quad (1.4)$$

де n_{min} і n_{max} – це мінімальні та максимальні значення частоти дискретизації, що контролюються параметрами сцени.

Виконання алгоритму Parallax Mapping відображення графічної сцени можна підсумувати таким чином:

- обчислення напрямку огляду дотичного простору $\widehat{V_{ts}}$ та напрямку світла $\widehat{L_{ts}}$ на вершину з подальшим інтерполюванням та нормалізацією у піксельному шейдері;

- обчислення вектора зміщення паралаксу [13],[21] P для визначення максимального візуального зміщення у текстурному просторі для поточного рівня;
- проведення проміню виду $\widehat{V_{ts}}$ вздовж P для обчислення точки перетину текстури висот з променем та обрахування зміщення координати текстури t_{off} ;
- визначення коефіцієнту світлової видимості v , шляхом проведення променю спрямованого світла $\widehat{L_{ts}}$ і виконавши вибірку текстури висоти для зафарбовування;
- зафарбовування пікселю [1-3], використовуючи коефіцієнт видимості v , $\widehat{L_{ts}}$ та атрибути пікселя (такі як альbedo, карта кольорів, нормалі тощо), що відбираються на рівні зміщення координат текстури t_{off} .

На (рис. 1.8) продемонстровано процес обрахування та зафарбовування пікселя графічним процесором, використовуючи технологію Parallax Mapping.

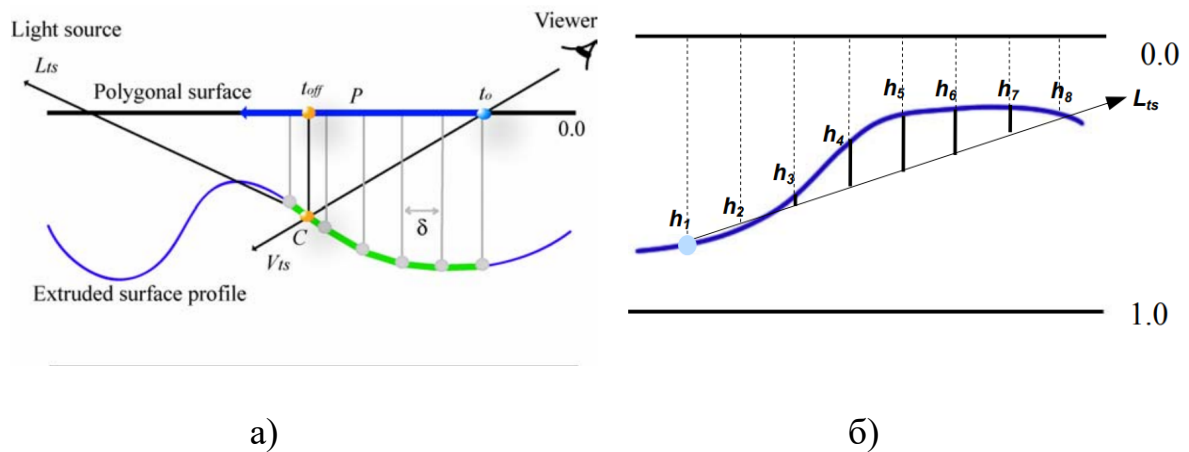


Рисунок 1.8 – Зафарбовування пікселю за методом Parallax Mapping

а) вибірка поля висоти та поточного напрямку погляду б) обрахування світлового променя

У підсумку, Parallax Mapping поєднує в собі теоретичні основи та практичні застосування для створення візуально реалістичних зображень. Технології, які використовують цей метод, постійно вдосконалюються, пропонуючи нові рішення для задач рендерингу. Це сприяє подальшому розвитку графічних двигунів і покращенню графіки в різних іграх і симуляціях.

1.4 Пакети прикладних програм для текстуровання

Substance Painter [8],[22],[32] — одна з найпопулярніших програм для текстуровання. Вона пропонує потужні інструменти для створення та редагування текстур у реальному часі. Програма підтримує роботу з різними типами карт, включаючи карти нормалей, карти висот та карти відбиттів. Substance Painter також пропонує можливість роботи з інтелектуальними матеріалами, які автоматично адаптуються до поверхні моделі, що значно спрощує процес текстуровання.

Substance Painter дозволяє художникам формувати зображення в режимі реального часу, що означає, що зміни, внесені в текстури, одразу ж відображаються на моделі. Це дозволяє швидко та ефективно вносити корективи та поліпшення. Програма також підтримує інтеграцію з іншими популярними 3D-програмами, такими як Maya та 3ds Max, що робить її універсальним інструментом для текстуровання.

Adobe Photoshop [22],[38] також широко використовується для текстуровання, особливо для створення та редагування 2D-текстур. Програма пропонує багато інструментів для малювання, редагування та маніпуляції зображеннями, що робить її незамінною для художників по текстурам. Photoshop дозволяє створювати складні текстури за допомогою шарів, масок та фільтрів, що дає художникам велику гнучкість і контроль над процесом.

Photoshop також підтримує роботу з різними форматами файлів, що робить його універсальним інструментом для текстуровання. Програма пропонує багато плагінів та розширень, які можуть покращити функціональність і спростити

процес створення текстур. Завдяки своїй популярності та широкому поширенню, Photoshop є одним із основних інструментів для багатьох художників по текстурам.

Blender [4], [12],[22],[34] — це безкоштовна та відкрита програма для 3D-моделювання, яка також включає в себе потужні інструменти для текстурювання. Blender підтримує роботу з різними типами текстур та карт, а також пропонує вбудовані інструменти для малювання текстур прямо на 3D-моделях. Програма також підтримує роботу з PBR-матеріалами [26],[33], що дозволяє створювати реалістичні текстури та матеріали (рис. 1.9).



Рисунок 1.9 – Приклад інтерфейсу Blender

Blender пропонує широкий спектр інструментів для текстурювання, включаючи можливість роботи з UV-розгортками, що дозволяє точно контролювати, як текстури будуть відображатися на моделі [17]. Програма також підтримує роботу з вузлами, що дозволяє створювати складні матеріали та текстури за допомогою процедурних методів. Завдяки своїй безкоштовності та відкритості, Blender є відмінним вибором для початківців та досвідчених художників по текстурам.

Mari від Foundry [16],[18] — це професійний інструмент для текстурювання, який використовується у кіноіндустрії (рис. 1.10). Програма пропонує потужні можливості для роботи з текстурами високої роздільної

здатності і підтримує роботу з кількома UV-картами [3]. Mari дозволяє художникам створювати деталізовані текстури для складних моделей, що робить її незамінним інструментом для професіоналів.



Рисунок 1.10 – Приклад інтерфейсу Mari

Mari підтримує роботу з текстурами високої роздільної здатності, що дозволяє створювати деталізовані та реалістичні поверхні. Програма також пропонує можливість роботи з кількома шарами та масками, що надає художникам велику гнучкість і контроль над процесом текстурювання. Завдяки своїм потужним можливостям та професійним інструментам, Mari є одним з основних інструментів для текстурювання у кіноіндустрії.

Проаналізуємо безкоштовні ресурси для текстур та матеріалів.

Textures.com — це один з найпопулярніших сайтів для завантаження безкоштовних текстур. Сайт пропонує велику колекцію текстур різних категорій, включаючи текстури дерева, металу, каменю та багатьох інших. Textures.com також пропонує можливість завантаження текстур у різних роздільних здатностях, що дозволяє використовувати їх у проектах будь-якого масштабу.

На Textures.com можна знайти текстури для різних потреб, включаючи архітектурні візуалізації, ігрові проекти та анімації. Сайт також пропонує преміум-підписку, яка надає доступ до текстур високої роздільної здатності та додаткових ресурсів. Завдяки своїй обширній колекції та зручному інтерфейсу, Textures.com є відмінним ресурсом для художників по текстурам.

Poliigon пропонує високоякісні текстури та матеріали, які можна використовувати в різних 3D-проектах. Хоча більшість ресурсів на сайті платні, є й безкоштовні текстури, які можна використовувати у своїх проектах. Poliigon також пропонує текстури в різних форматах та роздільних здатностях, що робить їх універсальними для використання в різних програмах.

Poliigon спеціалізується на наданні текстур високої якості, які можуть використовуватися у професійних проектах. Сайт пропонує текстури для різних потреб, включаючи архітектурні візуалізації, ігрові проекти та анімації. Завдяки своїй високій якості та різноманітності, Poliigon є відмінним ресурсом для професійних художників по текстурам.

CC0 Textures — це сайт, який пропонує текстури, які можна використовувати без будь-яких обмежень. Усі текстури на сайті розповсюджуються під ліцензією Creative Commons Zero (CC0), що означає, що їх можна використовувати у комерційних і некомерційних проектах без необхідності вказувати авторство. CC0 Textures пропонує текстури різних категорій, включаючи текстури дерева, металу, каменю та багатьох інших.

На CC0 Textures можна знайти текстури для різних потреб, включаючи архітектурні візуалізації, ігрові проекти та анімації. Сайт також пропонує текстури в різних роздільних здатностях, що дозволяє використовувати їх у проектах будь-якого масштабу. Завдяки своїй вільній ліцензії та обширній колекції, CC0 Textures є відмінним ресурсом для художників по текстурам [36].

HDRI Haven пропонує безкоштовні HDRI-карти, які можна використовувати для освітлення 3D-сцен. Ці карти допомагають створити реалістичне освітлення та відображення у ваших проектах. HDRI Haven

пропонує HDRI-карти різних категорій, включаючи денне та нічне освітлення, студійне освітлення та багато іншого.

На HDRI Haven можна знайти HDRI-карти для різних потреб, включаючи архітектурні візуалізації, ігрові проекти та анімації. Сайт також пропонує HDRI-карти в різних роздільних здатностях, що дозволяє використовувати їх у проектах будь-якого масштабу. Завдяки своїй обширній колекції та високій якості, HDRI Haven є відмінним ресурсом для художників по текстурам.

1.5 Висновки

В розділі проведено аналіз предметної галузі. Розглянуто основні методи рельєфного текстуровання. Наведено особливості фільтрації текстур. Проаналізовано паралакс-мепінг. Наведено характеристику пакетів прикладних програм для текстуровання.

2 РОЗРОБКА МЕТОДІВ ТЕКСТУРУВАННЯ, ЯКІ УРАХОВУЮТЬ ПЕРСПЕКТИВУ

2.1 Базові перетворення при генерації тривимірних графічних сцен

Переформулювання координатних систем. Розглянемо систему координат у n вимірах, представлену базисом $(k_1, k_2, \dots, k_n)$, яка визначає розташування точки в просторі через числові значення k_i . У сфері комп'ютерної графіки найбільш розповсюдженим є використання дво- та тривимірних координатних систем.

При визначенні нової N -вимірної системи координат у певному базисі $(m_1, m_2, \dots, m_N)$ і поставленні задачі знаходження координат у цій новій системі, виходячи з відомих координат у попередній, рішення цього завдання, якщо воно можливе, можна виразити у такій формі:

$$\begin{cases} m_1 = f_1(k_1, k_2, \dots, k_n), \\ m_2 = f_2(k_1, k_2, \dots, k_n), \\ \dots \\ m_N = f_N(k_1, k_2, \dots, k_n). \end{cases} \quad (2.1)$$

де f_i - функція перетворення i -ї координати, яка приймає координати в системі як аргументи.

Залежно від характеру функцій перетворення їх класифікують як лінійні та нелінійні. Лінійне перетворення характеризується тим, що функції для кожного $i=1, 2, \dots, N$ є лінійними відносно координат $(k_1, k_2, \dots, k_n)$, а це означає, що

$$f_i = a_{i1}k_1 + a_{i2}k_2 + \dots + a_{in}k_n + a_{i,n+1} \quad (2.2)$$

де a_{ij} є константами. Такі перетворення відомі як лінійні. Коли n дорівнює N , ці перетворення класифікуються як афінні. На площині афінні перетворення можна виразити за допомогою таких формул:

$$\begin{aligned} X &= Ax + By + C, \\ Y &= Dx + Ey + F. \end{aligned} \quad (2.3)$$

де $A, B, \dots, F$ – сталі величини. Ці величини можна розцінювати як координати в новій системі координат. Обернене перетворення також відноситься до афінних перетворень.

$$\begin{aligned} X &= A'x + B'y + C', \\ Y &= D'x + E'y + F'. \end{aligned} \quad (2.4)$$

Афінні перетворення ефективно представляти у вигляді матриць. Константи $A, B, \dots, F$ формують матрицю перетворень. Ця матриця, коли вона множиться на стовпцеву матрицю координат (x, y) , дає нову стовпцеву матрицю $[19]$ (X, Y) . Для включення констант C і F , використовуються однорідні координати, додаючи додатковий рядок до матриці координат:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \quad (2.5)$$

Запис у вигляді матриць дозволяє наочно ілюструвати кілька послідовних перетворень. Особливості афінних перетворень такі:

1. Кожне афінне перетворення може бути представлене як комбінація декількох базових операцій, таких як зсув, масштабування (розтягнення або стиснення) і поворот.

2. Вони зберігають прямолінійність ліній, паралельність прямих ліній, пропорції довжин відрізків на одній прямій, а також співвідношення площ фігур.

Для перетворень в просторі:

$$\begin{aligned} X &= Ax + By + Cz + D, \\ Y &= Ex + Fy + Gz + H, \\ Z &= Kx + Ly + Mz + N. \end{aligned} \quad (2.6)$$

де $A, B, \dots, N$ – константи. Запишемо останню систему у матричній формі

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ K & L & M & N \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (2.7)$$

У тривимірному просторі будь-яке афінне перетворення можна представити як послідовність простих операцій

1. Переміщення осей координат відповідно до заданих параметрів dx, dy, dz

$$\begin{cases} X = x - dx, \\ Y = y - dy, \\ Z = z - dz. \end{cases} \quad \begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.8)$$

2. Розтяг на kx, ky, kz :

$$\begin{cases} X = x/kx, \\ Y = y/ky, \\ Z = z/kz. \end{cases} \quad \begin{bmatrix} 1/kx & 0 & 0 & 0 \\ 0 & 1/ky & 0 & 0 \\ 0 & 0 & 1/kz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.9)$$

3. Поворот точки навколо осі x на кут φ (рис. 2.1).

Поворот. У тривимірному просторі варіантів повороту значно більше, ніж у двовимірному. Розглянемо деякі прикладів поворотів:

Приклад поворот точки навколо осі X (див. рис. 2.1).

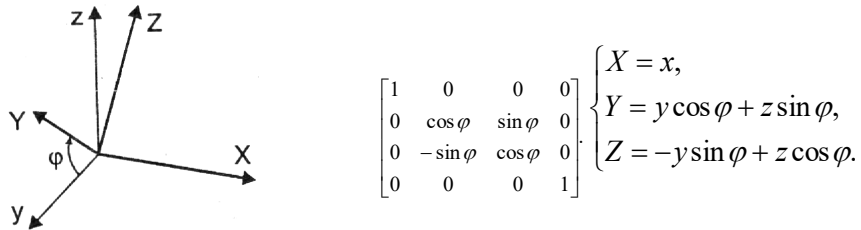


Рисунок 2.1 – Поворот точки навколо осі X в тривимірному просторі

Запишемо формулам для осі y при повороті на кут

Виконаємо поворот точки навколо ψ (рис. 2.2).

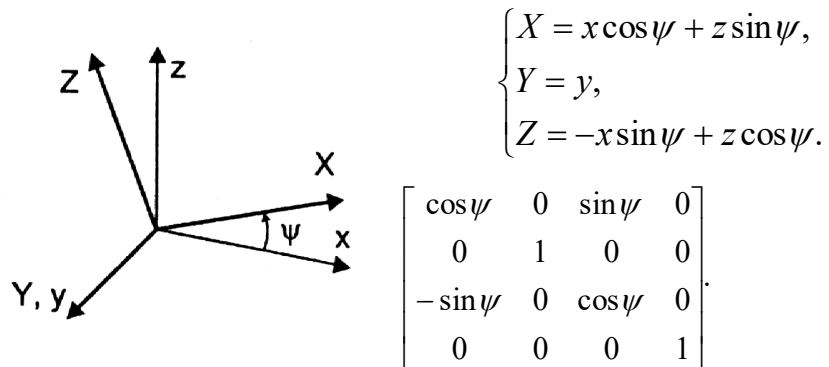


Рисунок 2.2 – Поворот точки навколо осі y

Запишемо аналогічно для осі z при повороті на кут γ (рис. 2.3).

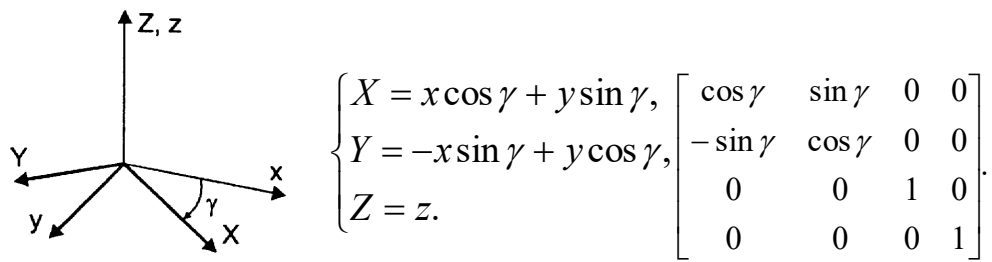


Рисунок 2.3 –. Поворот точки навколо осі z

При проектуванні просторових об'єктів на екрані важливо знати їхні координати. Аналізуємо дві ключові системи координат: світові та екранні координати. Світові координати фіксують реальне положення об'єктів у просторі з високою точністю. Екранні координати, з іншого боку, використовуються для відображення об'єктів у певній проекції на пристрої, як-от моніторі чи аркуші паперу.

Основна задача полягає у перетворенні світових координат в екранні для точного синтезу зображення в двовимірному форматі на площині екрана. Таке перетворення вимагає ретельного обрахунку координат проекції, щоб забезпечити відтворення просторових відносин та масштабування відповідно до заданої проекції.

У комп'ютерній графіці широко використовуються два типи проекцій: паралельна та центральна (часто називана перспективною). При паралельній проекції усі промені проекціювання є паралельними один одному, що забезпечує збереження масштабів об'єктів без зміни їхньої відносної величини в залежності від віддалі. Водночас, при центральній проекції всі промені виходять з однієї точки, що додає глибину і перспективу зображенню, створюючи більш реалістичне відображення просторових об'єктів.

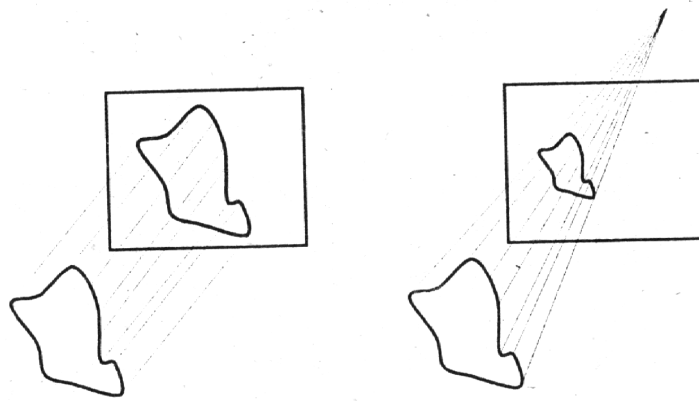


Рисунок 2.4 –. Проекції: зліва - паралельна, справа -центральна

Паралельну проекцію [3] можна розглядати як спеціальний випадок центральної проекції, де точка сходу променів проекціювання знаходиться в нескінченності. Це означає, що промені, що проходять через об'єкт, залишаються паралельними на всьому своєму шляху до площини проекції, відсутня перспективна деформація, яка характерна для центральної проекції [21].

Для коректного застосування методів текстурювання з урахуванням перспективи у програмних продуктах спершу необхідно скласти матрицю перетворення, яка б дозволила конвертувати координати з екранного простору в текстурний простір. Таке перетворення дозволяє адекватно мапити текстури на тривимірні моделі, забезпечуючи натуралістичний вигляд об'єктів на екрані.

Нехай задано полігон, що розташований на площині об'єкта, яка описується рівнянням площини у просторі $AX + BY + CZ + D = 0$. Відстань від центру перспективи до площини екрану позначимо як f . Кожна точка на екранній площині, що визначена $P_{s2D} : (x, y, f)^T$, може бути спроектована на площину об'єкта у тривимірному просторі так:

$$\begin{bmatrix} w_1 X \\ w_1 Y \\ w_1 Z \\ w_1 \end{bmatrix} = \begin{bmatrix} -D & 0 & 0 \\ 0 & -D & 0 \\ 0 & 0 & -D \\ A & B & C \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix}. \quad (2.10)$$

Матриця перетворення 4×4 для тривимірного простору з застосуванням однорідних координат має вигляд:

$$T = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ h & i & j & r \\ l & m & n & s \end{bmatrix}. \quad (2.11)$$

Цю матрицю можна подати так: $\begin{bmatrix} 3 \times 3 & 3 \times 1 \\ 1 \times 3 & 1 \times 1 \end{bmatrix}$

Матриця 3×3 - виконує лінійні перетворення, які включають зміну масштабу, зсув і поворот. Ця матриця забезпечує основну частину афінних трансформацій:

- Зміна масштабу визначається діагональними елементами матриці.
- Поворот визначається елементами матриці, що відповідають за обертання навколо осі.
- Зсув включається шляхом модифікації відповідних елементів у матриці.

- Матриця 1×3 - здійснює перенесення, що дозволяє додати вектор зсуву до координат, зміщуючи точку в просторі. Ця матриця в основному використовується для фінального адаптування положення об'єкта в текстурному просторі.

- Матриця 3×1 - забезпечує перетворення в перспективі, яка використовується для корекції відображення точок відповідно до точки зору або перспективного відхилення, змінюючи глибину перспективи об'єкта.

- Скалярний елемент 1×1 виконує загальну зміну масштабу на всі координати, який може бути використаний для загальної корекції масштабування об'єкта.

Будь-яка точка площини об'єкта може бути перетворена (через поворот і зсув) в простір об'єкта $P_{O3D} : (w_2X_0, w_2Y_0, w_2Z_0, w_2)^T$ таким чином:

$$\begin{bmatrix} w_2X_0 \\ w_2Y_0 \\ w_2Z_0 \\ w_2 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_1X \\ w_1Y \\ w_1Z \\ w_1 \end{bmatrix}. \quad (2.12)$$

Для перетворення точки з світової системи координат в двовимірний простір текстури потрібно застосувати афінні перетворення:

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_2X_0 \\ w_2Y_0 \\ w_2Z_0 \\ w_2 \end{bmatrix} \quad (2.13)$$

Інтегруючи разом зазначені вище матриці, отримуємо кінцеву матрицю перетворення, яка конвертує екранні координати в текстурні. Це дозволяє ефективно накладати текстури на тривимірні об'єкти, забезпечуючи при цьому збереження правильної перспективи зображення.

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -D & 0 & 0 \\ 0 & -D & 0 \\ 0 & 0 & -D \\ A & B & C \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix}. \quad (2.14)$$

Запишемо таке співвідношення :

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix}. \quad (2.15)$$

Отже, перед тим як приступити до текстуровання тривимірного об'єкта, важливо для кожного полігону встановити матрицю перетворення, яка переводить екранні координати в текстурні. Ця матриця визначається відповідно до співвідношення:

$$u(i) = \frac{f_1(i)}{f_2(i)} = \frac{f_1(i-1) + M_1}{f_2(i-1) + M_2}, \quad (2.16)$$

$$v(i) = u(i) \cdot \alpha + \beta.$$

2.2 Метод прискореної анізотропної фільтрації

Сучасні GPU характеризуються великим об'ємом доступної оперативної пам'яті, що дозволяє працювати з великою кількістю текстурних карт. Це обумовлює тенденцію до збільшення продуктивності обчислень шляхом застосування спеціальних текстурних карт із попередньо розрахованими проміжними значеннями (наприклад, карт висот, нормалей, відстаней тощо). Великий обсяг обчислень ускладнює застосування вагових функцій на основі гаусівської моделі пікселя в складних графічних сценах, коли потрібна висока швидкодія текстуровання. Збільшення продуктивності анізотропної фільтрації можна досягти, використовуючи спеціальні текстурні карти для визначення вагових коефіцієнтів [22].

Карта ваг — це кольорова текстурна карта, де значення кожного текселя варіюється в межах від 0 до 1, відображаючи значення певної вагової функції $W(u, v)$ для відповідних координат (рис. 2.5).

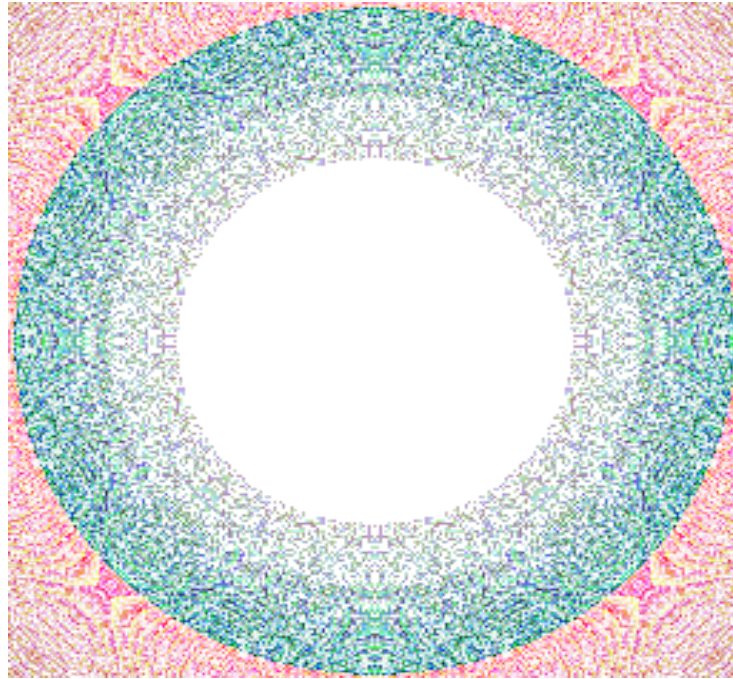


Рисунок 2.5 –Приклад карти вагових коефіцієнтів із роздільною здатністю 128x128

Щоб знайти текстурні координати точки на карті вагових коефіцієнтів, доцільно використовувати білінійну інтерполяцію координат, де за координатні осі беруться осі еліпса. Для розрахунку координат точки необхідно визначити відстань від точки до відповідних осей. Відстань від заданої точки, що розглядається, розраховуємо за виразом:

$$d = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \quad (2.17)$$

де $(x_1, y_1), (x_2, y_2)$ – значення координати двох точок на відрізку прямої, а x_0, y_0 – абсциса та ордината заданої точки.

Для обчислення координат обох осей використаємо координати центру еліпса. Координати другої точки для кожної з осей можна визначити шляхом

додавання до координат векторів осей u_0, v_0 , отриманих зі складання рівняння еліпса.

$$\begin{aligned} [x_2, y_2]_U &= [x_1, y_1] + [U_x, U_y] = [x_1 + U_x, y_1 + U_y] \\ [x_2, y_2]_V &= [x_1, y_1] + [V_x, V_y] = [x_1 + V_x, y_1 + V_y] \end{aligned} \quad (2.18)$$

де $[x_2, y_2]_U, [x_2, y_2]_V$ – координати точок.

Координати точок $[x_2, y_2]_U, [x_2, y_2]_V$, що знаходяться на осях еліпса, визначаються шляхом обчислення положення точок уздовж головних і побічних осей еліпса відносно його центру. Для цього координати центру еліпса збільшують або зменшують на значення довжини головної та побічної осей, що задає розташування точок уздовж цих осей.

Різниці між координатами точок дорівнюють відповідним координатам векторів: Різниці між координатами точок дорівнюють відповідним координатам векторів $x_2 - x_1 = U_x, y_2 - y_1 = U_y$. Підставимо отримані значення у формулу (2.13). Це дає можливість визначити абсолютні координати точки в системі координат осей еліпса. Обчислимо x і y за відповідними виразами:

$$\begin{aligned} x &= \frac{V_y \cdot u - V_x \cdot v + (V_x + u_0) \cdot v_0 - (V_y + v_0) \cdot u_0}{|\sqrt{A}|} = \frac{u_0 \cdot V_y - v_0 \cdot V_x - V_y \cdot u + V \cdot v}{|\sqrt{A}|} \\ y &= \frac{U_y \cdot u - U_x \cdot v + (U_x + u_0) \cdot v_0 - (U_y + v_0) \cdot u_0}{\sqrt{C}} = \frac{u_0 \cdot U_y - v_0 \cdot U_x - U_y \cdot u + U \cdot v}{\sqrt{C}} \end{aligned} \quad (2.19)$$

де u_0, v_0 – абсциси та ординати центру еліпса.

Координати поточного текселя визначають положення цього текселя на текстурній карті. Для їх обчислення використовуються відносні координати в межах текстури, які можна представити як $(u, v)(u, v)(u, v)$, де u і v – нормалізовані

координати текселя в діапазоні від 0 до 1 уздовж ширини та висоти текстури відповідно. Абсолютні координати текселя на текстурі визначаються шляхом множення u та v на розміри текстури в пікселях.

Щоб визначити відносні координати точки в системі координат осей еліпса, потрібно поділити абсолютні координати точки на довжини відповідних півосей еліпса. Довжини півосей відповідають модулям $\vec{U}$ і $\vec{V}$:, які можна обчислити за формулами:

$$|\vec{U}| = \sqrt{U_y^2 + U_x^2} = \sqrt{C}, |\vec{V}| = \sqrt{V_y^2 + V_x^2} = \sqrt{A} \quad (2.20)$$

Відносні координати точки в системі координат осей еліпса можна обчислити за такими виразами:

$$\begin{aligned} x &= u_0 \cdot V_y - v_0 \cdot V_x - V_y \cdot u + V_x \cdot v \cdot \frac{1}{\sqrt{A \cdot C}} \\ y &= u_0 \cdot U_y - v_0 \cdot U_x - U_y \cdot u + U_x \cdot v \cdot \frac{1}{\sqrt{A \cdot C}} \end{aligned} \quad (2.21)$$

З формул (2.14) слідує, що $S = \frac{1}{\sqrt{A \cdot C}}$ спільне для обчислення x і y , тому його достатньо обчислити лише один раз. Сучасні процесори підтримують швидке наближене обчислення оберненого квадратного кореня, що дозволяє зменшити навантаження на обчислювальні ресурси, оминаючи операцію ділення. Для ЦП, що мають обмежені ресурси, швидке обчислення оберненого квадратного кореня можна виконати за допомогою методу Ньютона.

Введемо такі позначення: $K_x = u_0 \cdot V_y - v_0 \cdot V_x$, а $K_y = u_0 \cdot U_y - v_0 \cdot U_x$. Змінні формул (2.22) не корельовано до координат поточного текселя. Зрозуміло, що їх можна обчислити один раз. Таким чином, формули (2.14) набувають такого вигляду:

$$x = K_x - V_y \cdot u + V \cdot v \cdot S, y = K_y - U_y \cdot u + U \cdot v \cdot S \quad (2.22)$$

Абсолютні координати u, v на карті вагових коефіцієнтів визначаються так:

$$u = \frac{w \cdot (x + 1)}{2}, v = \frac{h \cdot (y + 1)}{2} \quad (2.23)$$

де w, h - розміри по осям текстурної карти.

Якщо карта вагових коефіцієнтів створюється на основі функції, яка є симетричною щодо обох осей, наприклад, за допомогою гаусівської кривої, обсяг пам'яті, потрібний для зберігання текстури, можна зменшити в чотири рази. Для цього можна використовувати лише верхній правий сектор текстурної карти. У такій ситуації вирази (2.24) набудуть наступного вигляду:

$$x = |K_x - V_y \cdot u + V \cdot v \cdot S|, y = |K_y - U_y \cdot u + U \cdot v \cdot S| \quad (2.24)$$

$$u = x \cdot w, v = y \cdot h$$

Коли важливо мінімізувати обсяг необхідної пам'яті, і вагова функція задана як тіло обертання, можна застосувати одновимірну текстуру (рис. 2.6).



Рисунок 2.6 – Зразок карти

Для тіла обертання значення функції залишається однаковим для всіх точок, розташованих на однаковій відстані від осі обертання. Відповідно, координати точки в одновимірній текстурі можна розрахувати за виразом:

$$x = \sqrt{(K_x - V_y \cdot u + V \cdot v \cdot S)^2 + (K_y - U_y \cdot u + U \cdot v \cdot S)^2} \quad (2.25)$$

Для векторної форми:

$$x = |\vec{P}|, \quad (2.26)$$

де $\vec{P}$ – значення, отримане за формулами (2.24).

Вектор, координати якого відповідають координатам точки, обчисленим за формулами (2.16).

Формули отримають такий вигляд:

$$u = x \cdot w, v = 0 \quad (2.27)$$

Отже, завдяки отриманим формулам можна визначити координати текселя за картою вагових коефіцієнтів. Це дозволяє зчитувати вагові коефіцієнти безпосередньо з текстурної карти, не розраховуючи їх. Під час тестування було встановлено, що застосування вагової карти замість обчислення складних формул підвищує продуктивність до 25%.

На рис. 2.7 і 2.8 наведено приклади виконання текстурування без та використання вагових коефіцієнтів.

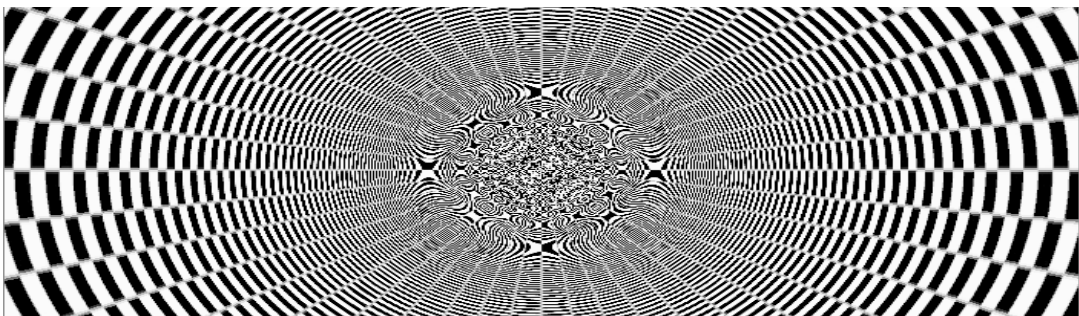


Рисунок 2.7 – Приклад текстурування без виконання анізотропної фільтрації

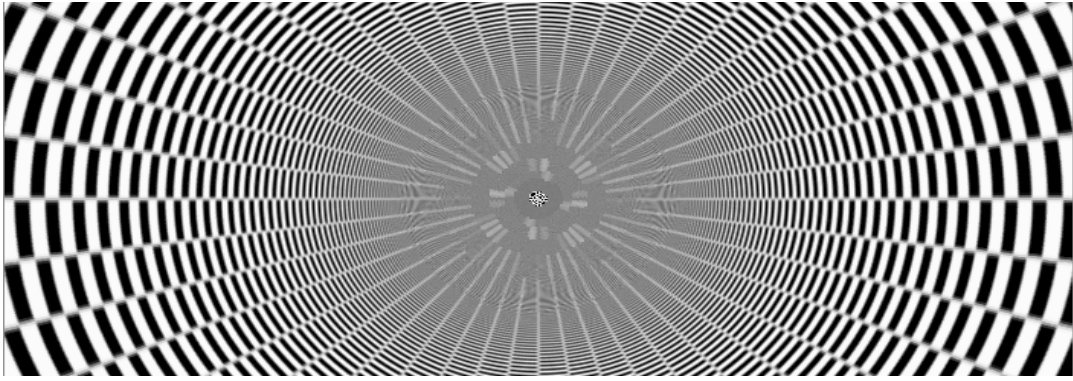


Рисунок 2.8 –Текстурування з використанням вагових функцій

2.3 Метод текстурування з використанням неортогональної растеризації

У методі Хесберга використовуються такі рівняння.

$$u_i = \frac{A_1 \cdot x_i + B_1 \cdot y_i + C_1}{D \cdot x_i + E \cdot y_i + F}, \quad v_i = \frac{A_2 \cdot x_i + B_2 \cdot y_i + C_2}{D \cdot x_i + E \cdot y_i + F} \quad (2.28)$$

За умови, що $x_{i+1} = x_i + 1$, то

$$u_{i+1} = \frac{A_1 \cdot (x_i + 1) + B_1 \cdot y_i + C_1}{D \cdot (x_i + 1) + E \cdot y_i + F} = \frac{(A_1 \cdot x_i + B_1 \cdot y_i + C_1) + A_1}{(D \cdot x_i + E \cdot y_i + F) + D} \quad (2.29)$$

Аналогічну вираз має місце і для v_{i+1} . З поданих рівнянь видно, що для обчислення кожного текселя потрібно виконати 2 мікрооперації ділення та 3 мікрооперації додавання. У запропонованому методі для кожного текселя достатньо 1 операції ділення, 3 операцій додавання та 1 операції множення. Переводячи ці операції у такти процесора (Pentium M), можна стверджувати, що у новому методі продуктивність обчислення текстелів зросла в середньому в 1,5 раза. Підготовчі операції до растеризації рядка в запропонованому методі займають більше часу порівняно з методом Хесберга, але значно менше при розрахунку на полігон. Моделювання показало, що застосування цього підходу

дозволяє зменшити час текстуровання типового трикутника приблизно на 20 % .

Щоб зменшити кількість операцій ділення при накладанні текстур, можна виконати покроковий обхід об'єкта в об'єктному просторі за умови, що рядки растеризації розташовані на постійній відстані від спостерігача. Обчислимо коефіцієнт нахилу лінії в екранній системі координат, що відповідає відрізку лінії в світовій системі координат із фіксованим значенням координати z для РР. Припустимо, що у світовій системі координат задано трикутник, який однозначно визначає площину, рівняння якої можна записати так.

$$AX_w + BY_w + CZ_w = D, \quad (2.30)$$

де A, B, C – координатами вершин трикутника.

З наведеного виразу знаходимо, що

$$Z_w = \frac{A \cdot X_w + B \cdot Y_w + D}{C} \quad (2.31)$$

Між системами координат має місце такі вирази $X_v = \frac{X_w}{Z_w}, Y_v = \frac{Y_w}{Z_w}$.

Подамо отримані рівняння так:

$$X_w = X_v \cdot Z_w, \quad Y_w = Y_v \cdot Z_w. \quad (2.32)$$

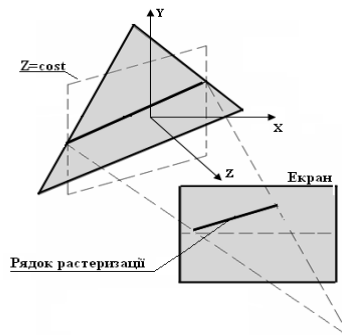


Рисунок 2.9 – РР у об'єктній та екранній системі координат

Знайдемо кут нахилу відрізка прямої $Y_v = k \cdot X_v + h$.

Тому можна записати, що $Y_w = Y_v \cdot Z_w = (k \cdot X_v + h) \cdot Z_w$.

Виконаємо підстановку X_w, Y_w у вираз (2.18)

$$A \cdot X_v \cdot Z_w + B \cdot (k \cdot X_v + h) \cdot Z_w + C \cdot Z_w = D \quad (2.33)$$

Знайдемо Z_w

$$Z_w = \frac{D}{X_v \cdot (A + B \cdot k) + B \cdot h + C} \quad (2.34)$$

За умови, що для РР трикутника в екранній системі координат виконується, для будь-якого значення j справедливе таке рівняння.

$$\frac{D}{X_v \cdot (A + B \cdot k) + B \cdot h + C} = \frac{D}{(X_v + j) \cdot (A + B \cdot k) + B \cdot h + C} \quad (2.35)$$

Наведене рівняння має такий розв'язок $k = -A/B$. Тому нахил сканувального рядка РР залишається незмінним для всього трикутника, який аналізується в екранній системі координат. Значення координати для конкретного рядка растеризації можна легко визначити шляхом підстановки k у вираз (2.19).

$$Z_w = D / (B \cdot h + C) \quad (2.36)$$

Між координатами текстури та координатами об'єктного простору використовують такі вирази:

$$u = \frac{a \cdot X_v + b \cdot Y_v + c}{A \cdot X_v + B \cdot Y_v + C}, \quad v = \frac{d \cdot X_v + e \cdot Y_v + f}{A \cdot X_v + B \cdot Y_v + C} \quad (2.37)$$

Введемо такі позначення:

$$A \cdot X_v + B \cdot Y_v + C = T, \text{ а } I / T = \mathfrak{R}, \text{ тоді}$$

$$u = (a \cdot X_v + b \cdot Y_v + c) \cdot \mathfrak{R}, \quad v = (d \cdot X_v + e \cdot Y_v + f) \cdot \mathfrak{R} \quad (2.38)$$

Для растеризації РР, використовуючи текстурні координати, знаменник залишається незмінним, на відміну від традиційного методу, де він визначається для кожної окремої точки в рядку. Розглянемо детальніше процес обрахунку однієї з координат, наприклад u . Оскільки методика розрахунку однакова для обох координат, виведемо через k і включимо це у вже існуючий вираз. Таким чином, ми отримаємо:

$$u = (a \cdot X_v + b \cdot (k \cdot X_v + h) + c) \cdot \mathfrak{R} = [X_v \cdot (a + b \cdot k) + (b \cdot h + c)] \cdot \mathfrak{R}. \quad (2.39)$$

На початку рядка растеризації $X_v = 0$. Тому $u_0 = \mathfrak{R} \cdot (b \cdot B + c)$.

Збільшимо u на 1:

$$u_{i+1} = [(X_v + 1) \cdot (a + b \cdot k) + (b \cdot h + c)] \cdot \mathfrak{R} = u_i + \mathfrak{R} \cdot (a + b \cdot k). \quad (2.40)$$

Отримане рівняння можна ефективно виконувати за допомогою апаратного обладнання, за умови, що значення $\mathfrak{R}$ відоме. Використання неортогонального напрямку для растеризації ділянки, обмеженої полігоном, сприяє зниженню складності обчислень під час текстуровання поверхні тривимірного графічного об'єкта. У випадку трикутника, який має T внутрішніх точок, унаслідок цього підходу вдається зекономити $(T-q)$ операцій ділення, де q означає кількість горизонтальних рядків, за якими проводиться растеризація трикутника.

2.4 Модифікація методу parallax mapping

Кількість операцій для реалізації алгоритму паралакс-маппінгу [20] визначається максимальною кількістю повторювальних етапів при трасуванні видового вектора. Ця обчислювальна складність визначається довжиною вектора, яка в найгіршому випадку дорівнює довжині діагоналі текстурної карти. Існуючі методи оптимізації продуктивності паралакса маппінгу, які не уточнюють крок трасування, здатні зменшити кількість ітерацій, але часто це призводить до недостатньої точності обчислень і зниження реалістичності зображення. Зростання продуктивності при реалізації паралакса маппінгу з конкретизацією кроку трасування залежить від рельєфу конкретної поверхні, тому може бути неприпустимим у деяких випадках.

Значного підвищення продуктивності паралакса маппінгу можна досягти за рахунок використання спеціально створеної MIP-піраміди для зберігання карти висот. У звичайній MIP-піраміді значення кожного текстурного елемента дорівнює середньому значенню чотирьох сусідніх текселів на попередньому рівні. Натомість у додатковій піраміді буде зберігатися максимальне значення висоти в чотирьох точках замість середнього значення (рис. 2.10).

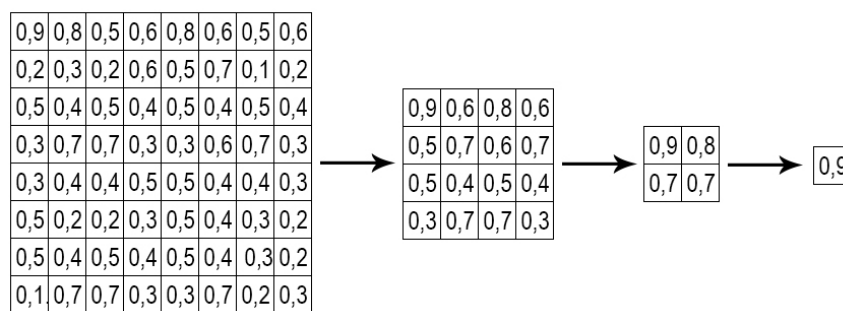


Рисунок 2.10 – Приклад MIP-піраміди максимальних значень

Подальше трасування видового вектора [26] необхідно реалізувати послідовно на всіх рівнях MIP-піраміди максимальних значень, якщо розрахунки

почати з найвищого та закінчуючи рівнем, що відповідає потрібній роздільній здатності (далі – базовий MIP-рівень).

На найвищому MIP-рівні текстура складається лише з одного текстурного елемента, який представляє висоту найвищої точки поверхні. Якщо це значення збігається з висотою, що відповідає початковим текстурним координатам на базовому рівні піраміди середніх значень, це означає, що текстурні координати вказують на найвищу точку. В такому випадку перекриття з жодною висотою не відбувається, і подальше трасування не є необхідним. Якщо ж значення більше, то потрібно переходити до трасування на наступному MIP-рівні.

На другому MIP-рівні площа текстури охоплює чотири текстурні елементи, отже, максимальна довжина видового вектора становить 2 текстури, а максимальна кількість ітерацій трасування дорівнює 2. Якщо значення текстурного елемента менше, ніж висота вектора в поточній точці, це гарантує, що одна з половин вектора не перетинає поверхню. Якщо обидва текстурні елементи мають значення менше висоти вектора, це означає, що точка не перекривається, і подальше трасування не проводиться. Якщо ж значення текстурного елемента перевищує висоту вектора в даній точці, є ймовірність того, що на базовому MIP-рівні може бути точка перетину вектора з поверхнею.

У подальшому необхідно виконувати дії трасувати на слідуючому нижчому MIP-рівні. Оскільки розмір текстури при переході на нижчий рівень збільшується вдвічі, текстурні координати вектора визначаються за такою формулою:

$$\overrightarrow{V_{i+1}} = \overrightarrow{V_i} \times 2, \quad (2.41)$$

де $\overrightarrow{V_{i+1}}$ - значення координати на нижчому рівні піраміди; $\overrightarrow{V_i}$ - значення координати на даному рівні піраміди.

Якщо на нижчому рівні піраміди знайдено точку, значення якої перевищує висоту вектора, необхідно продовжити рекурсивну перевірку на нижчих рівнях

MIP, доки не буде досягнуто базового рівня (див. рис. 2.11). При цьому на кожному рекурсивному рівні число ітерацій не перевищить двох.

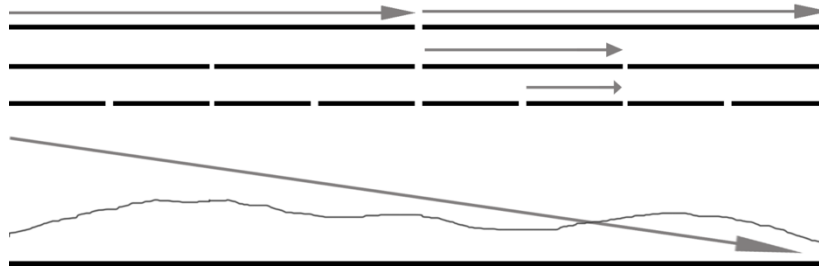


Рисунок 2.11 – Трасування видового вектора на різних MIP-рівнях

Якщо на одному з рівнів MIP не знайдено жодної точки, значення якої перевищує висоту вектора, це свідчить про те, що точка, висота якої перевищує висоту вектора в даній області текстури, існує, але вона не входить до проекції вектора на площину. Отже, перетин вектора з площиною в цій зоні текстури відсутній (рис. 2.12). У такому випадку слід перейти на вищий рівень MIP і продовжити трасування.

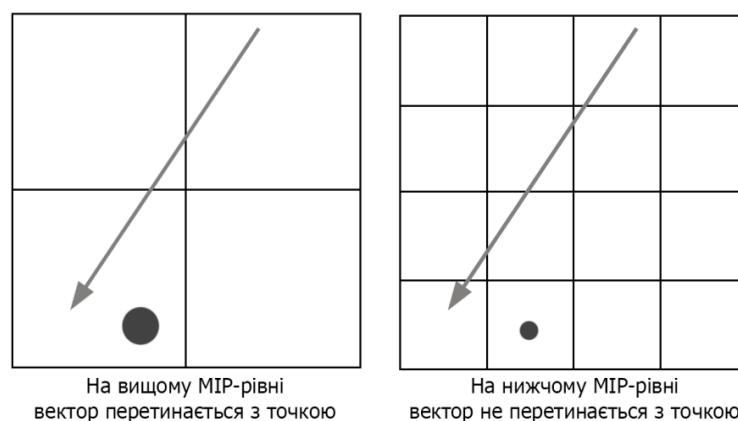


Рисунок 2.12 – Випадок, коли перетин не має місця

Обчислювальний процес завершується, коли рекурсія досягає заданого рівня, і підтверджується наявність точки перетину.

Таким чином, трасування на рівнях МІР, що перевищують базовий, дозволяє уникнути перевірки певних ділянок текстури, де немає точок перетину видового вектора із поверхнею. Ефективність методу залежить від рельєфу поверхні [17], і доцільність його використання повинна встановлюватися експериментально для конкретної карти висот. Найвищі показники продуктивності порівняно зі звичайним методом спостерігаються при моделюванні поверхонь із значними відстанями між перепадами висот, наприклад, цегляних стін. Такі поверхні містять велику кількість ділянок подібної висоти, які можуть бути виключені з обчислення під час трасування на вищих рівнях піраміди.

Можливе поєднане використання піраміди з іншими техніками трасування видового променя. Наприклад, трасуючи на одному з середніх рівнів МІР, можна виявити області, які слід перевірити за допомогою одного з методів з уточненням кроку, а також виключити з обчислювального процесу ті ділянки, перевірка яких не має сенсу.

Зображення, яке ілюструє запропонований метод, представлено на рисунку 2.13.



Рисунок 2.13 – Приклад сформульовано зображення за розробленим методом

2.5 Висновок

1. Отримано вирази для базових перетворень при генерації тривимірних графічних сцен з використанням текстуризації.
2. Розроблено метод прискореної анізотропної фільтрації з використанням вагових коефіцієнтів. Під час тестування було встановлено, що застосування вагової карти замість обчислення складних формул підвищує продуктивність до 25%.
3. Запропоновано для текстурування використовувати не ортогональне растеризацію. У випадку трикутника, який має T внутрішніх точок, унаслідок цього підходу вдається зекономити $(T-q)$ операцій ділення, де q означає кількість горизонтальних рядків, за якими проводиться растеризація трикутника.
4. Модифіковано метод parallax mapping з використанням MIP піраміди, найвищі показники продуктивності порівняно зі звичайним методом спостерігаються при моделюванні поверхонь із значними відстанями між перепадами висот, наприклад, цегляних стін. Такі поверхні містять велику кількість ділянок подібної висоти, які можуть бути виключені з обчислення під час трасування на вищих рівнях піраміди.

3 РОЗРОБКА ПЛАГІНУ ТЕКСТУРУВАННЯ ДЛЯ ЗАДАЧ РЕНДЕРИНГУ

3.1 Призначення та основні функції плагіну для текстурзування під задачі рендерингу

Магістерська кваліфікаційна робота спрямована на розробку програмного забезпечення, призначеного для текстурзування 3д об'єктів під задачі рендерингу. Основною метою є підвищення продуктивності та реалістичності текстурзування в системах комп'ютерної графіки за допомогою розробленого плагіну.

У ході виконання були реалізовані наступні функціональні можливості плагіну:

1. Нормалізація текстурних значень: нормалізація значень кривизни або інших параметрів текстур до діапазону $[0,1]$, що дозволяє ефективно зберігати дані у текстурних каналах.

2. Кодування даних у текстурні канали: кодування нормалізованих значень кривизни, нормалей або відбивної здатності у кольорові канали текстур (RGB) для зменшення обсягу збережених даних.

3. Відновлення даних із нормалізованих значень [4]: відновлення вихідних значень кривизни або інших параметрів із нормалізованих значень під час рендерингу.

4. Обчислення середньої кривизни на сітці: обчислення середньої кривизни для кожної вершини на тривимірній сітці на основі сусідніх вершин.

5. Компресії даних: застосування методів компресії для зменшення розміру текстурних карт без значної втрати якості.

6. Обчислення нормалей з використанням двох компонент: меншення обсягу даних для Normal Map шляхом збереження лише X та Y компонент, з подальшим відновленням Z компоненти у шейдері

7. Функція застосування ефектів з використанням кривизни: використання значень кривизни для модулювання ефектів зносу, пошкоджень або підвищення деталізації поверхні.

8. Інтерактивне меню: надання користувачам швидкого доступу до найбільш важливих і часто використовуваних інструментів для прискорення роботи.

9. Автоматичне управління матеріалами: функція видалення матеріалів з інших об'єктів при активації плагіна для спрощення організації сцени.

10. Оптимізація UV-мапінгу та копіювання матеріалів: швидке перенесення налаштувань матеріалів і UV-мапінгу між об'єктами для зменшення часу на ручне налаштування.

11. Випікання текстур (Baking V2): можливість випікання Ambient Occlusion (AO) та масок порожнин (Cavity) для збереження візуальних ефектів, таких як освітлення і тіні, у вигляді текстурних карт. Це забезпечує ефективну оптимізацію рендерингу та зменшує навантаження на систему.

12. Попередній перегляд AO/Cavity масок: інтерактивний перегляд ефектів AO та Cavity перед випіканням для точного налаштування та корекції.

13. Вимкнення тіней у рендері Cycles: можливість швидкого рендерингу без тіней для тестування та оптимізації текстур і матеріалів.

14. Процедурні маски та багат шарові текстури: впровадження різних масок, таких як Cavity Mask, AO Mask і маски нормалей у глобальному просторі, що дозволяє деталізовано налаштовувати взаємодію текстурних шарів.

15. Карти "Variety" для варіативності текстур: автоматичне додавання випадкових варіацій до текстур для підвищення їх реалістичності.

16. Підтримка багат шарових текстур: можливість застосування кількох текстурних шарів із різними властивостями (альbedo, нормалі, грубість тощо) для створення складних і реалістичних матеріалів.

17. Адаптація інтерфейсу для різних розмірів екранів: забезпечення зручності роботи на моніторах різних розмірів, включаючи високі роздільності.

18. Налаштування чутливості миші для різних рівнів DPI: можливість індивідуального налаштування швидкості миші для підвищення точності та комфорту роботи.

19. Адаптація інтерфейсу для різних розмірів екранів: забезпечення зручності роботи на моніторах різних розмірів, включаючи високі роздільності.

Плагін виконує велику кількість процесів текстурування, що є важливим етапом у розробці сучасних 3D моделей, що дозволяє значно підвищити ефективність і якість роботи. Плагін зменшує кількість ручних операцій та часу, необхідного для налаштування матеріалів і текстур, одночасно забезпечуючи високий рівень деталізації та реалістичності, полегшує роботу художників та дизайнерів, дозволяючи сконцентруватися на творчих аспектах проєкту [25].

Розроблений плагін, пропонує низку функцій, які спрощують процес нанесення текстур на 3D об'єкти, інтегруючи передові інструменти для швидкого створення, налаштування та адаптації матеріалів. До ключових функцій автоматизації можна віднести QuickTexture та QuickDecal, які дозволяють застосовувати базові текстури й декалі (наклейки) до об'єктів буквально кількома кліками. Ці інструменти значно скорочують час, який зазвичай витрачається на ручне налаштування текстур, дозволяючи художникам швидко досягти бажаного результату.

Розроблено метод, який реалізовано в плагіні, автоматичного перетворення вже готового матеріалу в формат "QTexture" спрощує застосування базових текстур і забезпечує швидкий доступ до редагування параметрів. Це значно економить час користувача, дозволяючи перетворювати складні матеріали в прості для використання текстури, що робить процес роботи більш зручним. Плагін підтримує ширший набір найменувань текстурних файлів, що робить його більш універсальною та сумісною з різними робочими процесами й іншими інструментами. Це дозволяє автоматично розпізнавати та сортувати текстури з різними варіантами назв, забезпечуючи ефективнішу інтеграцію

зовнішніх матеріалів і спрощуючи завантаження текстур у проєкт. На (рис. 3.1) наведено блок-схему процесу оптимізації текстурних карт для рендерингу.

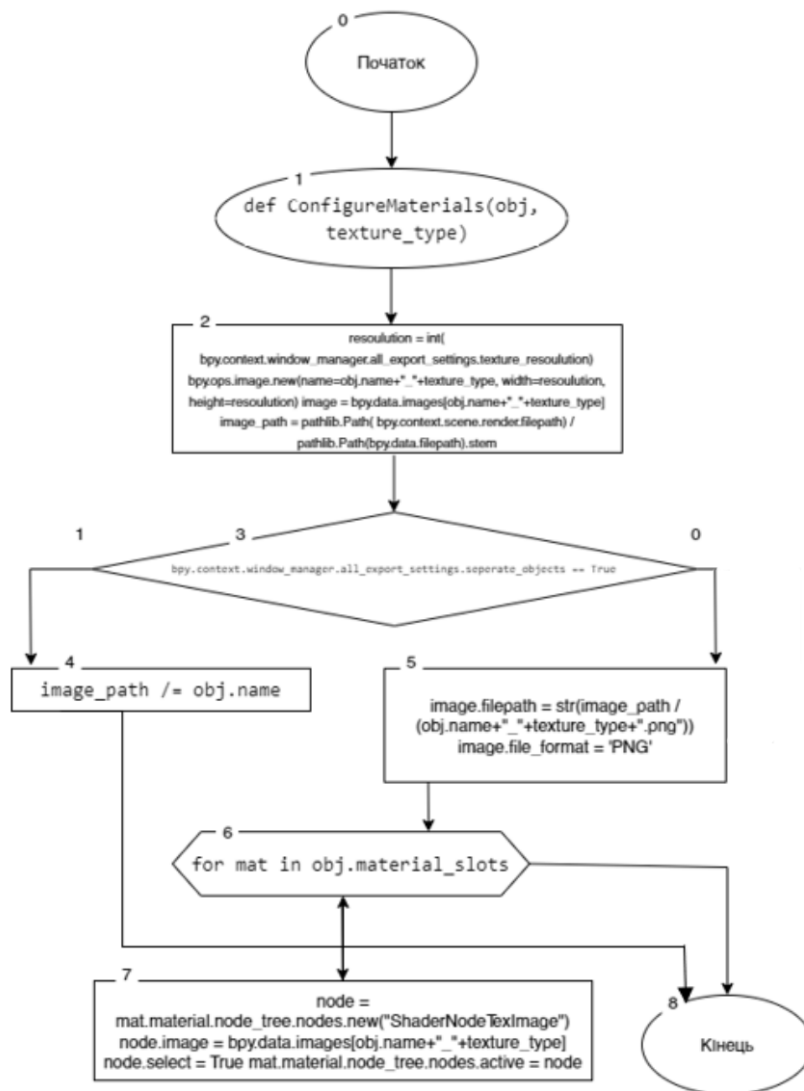


Рисунок 3.1 – Діаграма формування текстур зображень

Розроблено метод автоматизації – QuickDecal, який автоматично аносить наклейки на поверхню об'єкта. Цей інструмент значно спрощує процес додавання графічних елементів, таких як логотипи, пошкодження або інші деталі. QuickDecal дозволяє швидко вибрати та накласти декаль на модель,

використовуючи автоматичну прив'язку до поверхні, а також пропонує інструменти для налаштування її позиції, масштабу та прозорості (рис. 3.2).

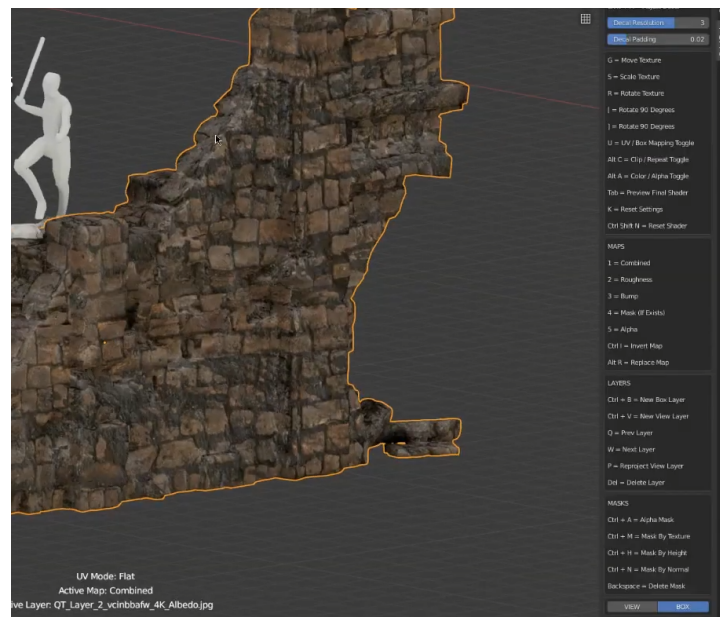


Рисунок 3.2 – Використання QuickDecal для текстури

Функція, яка була додана у плагін для фотомодельювання [15] дозволяє користувачам легко створювати об'єкти на основі зображень, застосовуючи їх на площинах або кубах. Ці інструменти дозволяють швидко перенести зображення на 3D-модель, забезпечуючи точну прив'язку текстур до об'єкта для подальшого налаштування. Це підходить для відтворення реалістичних сцен або об'єктів із фотомоделями, де потрібно досягти високої відповідності з оригінальним зображенням. Функція автоматичного перетворення вже готового матеріалу в формат "QTexture" спрощує застосування базових текстур і забезпечує швидкий доступ до редагування параметрів. Це значно економить час користувача, дозволяючи перетворювати складні матеріали в прості для використання текстури, що робить процес роботи більш зручним.

Завдяки автоматизації процесу текстурування, користувачі можуть зосередитися на творчих аспектах роботи, зменшуючи час, витрачений на технічні налаштування. Програма забезпечує не лише швидке нанесення текстур,

але й дозволяє легко експериментувати з різними варіантами матеріалів, не витрачаючи багато часу на підготовчі роботи. Крім того, автоматизація допомагає уникнути людських помилок, що часто трапляються при ручному налаштуванні матеріалів, забезпечуючи високу точність і узгодженість кінцевого результату.

3.2 Розробка засобів для управління текстурами

У плагіні реалізована можливість для рендера Cycles попереднього перегляду карт Ambient Occlusion (AO) [28-32] та масок порожнин (Cavity) за допомогою простого перемикача. Це дозволяє миттєво оцінити, як саме будуть виглядати тіні та заглиблення на текстурі, що спрощує налаштування ефектів для текстур, які вимагають тонкої деталізації [19]. Користувач може в режимі реального часу бачити, як працюють ефекти затінення та рельєфу, що дозволяє налаштовувати їх з більшою точністю та контролем (рис. 3.3). Ця функція особливо важлива для точного налаштування світлових ефектів і дрібних деталей на текстурах, які можуть бути критичними для реалістичності кінцевої сцени.

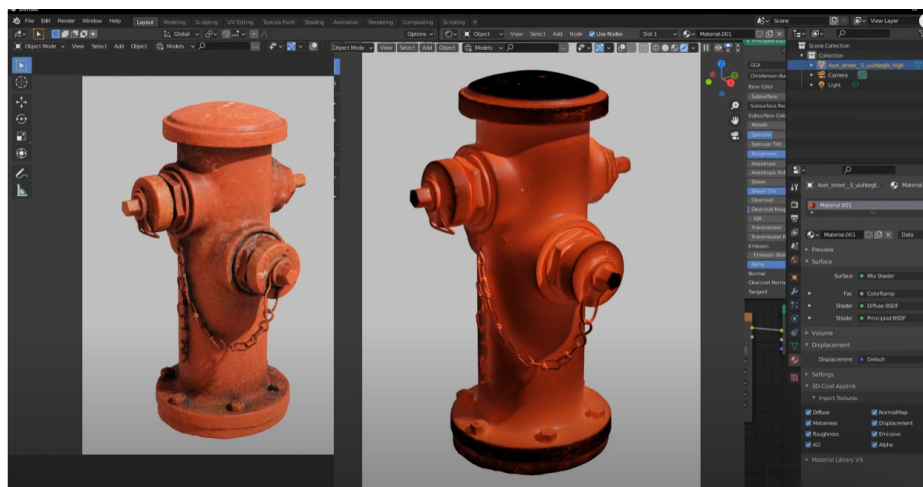


Рисунок 3.3 – Приклад попереднього карту Ambient Occlusion (AO) та масок порожнин (Cavity) в рендері Cycles

Наприклад, завдяки інтерактивному попередньому перегляду можна миттєво оцінити ефективність маскування заглиблень і тіньових зон на складних об'єктах. Швидке перемикання між QT Procedural Mapping та Triplanar Box Mapping [29],[38]. Procedural Mapping — це метод, при якому текстура наноситься за допомогою алгоритмів без використання UV-мапи, що зручно для об'єктів із складною геометрією. Натомість Triplanar Box Mapping проектує текстуру на об'єкт із трьох сторін (по осям X, Y та Z), що ідеально підходить для текстуровання кубічних або інших простих форм (рис. 3.4).

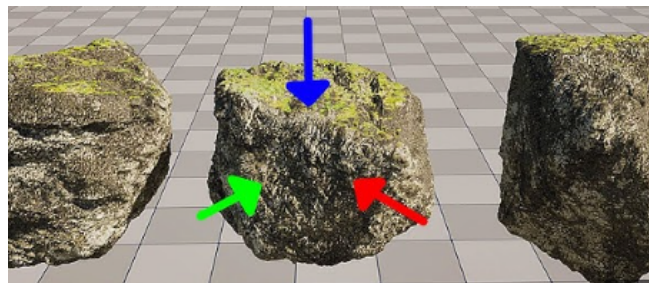


Рисунок 3.4 – Приклад Triplanar Box

Можливість швидкого перемикання дозволяє користувачам швидко тестувати різні варіанти нанесення текстур і вибирати той, який дасть найкращий результат.

Розроблено у плагіні функцію Shift+T, що надає можливість швидко перемикатися між процедурним мапінгом QuickTexture та тривимірним Box Mapping. Це забезпечує більшу гнучкість у виборі методів мапінгу, дозволяючи використовувати найбільш підходящий спосіб нанесення текстури залежно від типу об'єкта та особливостей текстуровання. Програма підтримує карти зміщення в рендері Cycles, що дозволяє додати більше об'ємності й реалізму текстурам [30].

Плагін підтримує карти зміщення в рендері Cycles, що дозволяє додати більше об'ємності й реалізму текстурам. Карти зміщення змінюють фактичну

геометрію об'єкта, створюючи ефект тривимірності поверхні без додаткових полігонів. Інструмент дозволяє створювати карти зміщення для до п'яти шарів, що підвищує деталізацію поверхні та дає можливість створювати реалістичні фактури, як-от тріщини або зморшки (рис. 3.5).



Рисунок 3.5 – Приклад використання карт зміщення через розроблений плагін

Створює складні рельєфи та об'ємні текстури без потреби в повторному рендерингу після кожної зміни. Художник може швидко налаштовувати карти зміщення для різних текстурних шарів, досягаючи реалістичних і детальних результатів. Інструмент дозволяє створювати карти зміщення для до п'яти шарів, що підвищує деталізацію поверхні та дає можливість створювати реалістичні фактури, як-от тріщини або зморшки [31]. Mesh Decals V2 значно покращує можливості накладення декалів (наклейок) на поверхню моделей, надаючи нові інструменти для налаштування країв та перенесення декалів на різні об'єкти. У Mesh Decals V2 є опція процедурного змішування альфа-каналу по краях декалі, що дозволяє створювати більш плавні переходи між декаллю та основною текстурою моделі. Це надає декалям більш природного вигляду, що особливо важливо для створення реалістичних зношених країв чи плавних переходів. Ще однією важливою особливістю Mesh Decals V2 є можливість перепроєктувати наклейку на іншу геометрію, якщо це необхідно. Наприклад, якщо декаль була

створена для одного об'єкта, але згодом виникла потреба застосувати її до іншого, користувач може легко адаптувати наклейку без втрати якості чи потреби її відновлення з нуля. Це значно підвищує гнучкість та ефективність роботи з наклейками в складних сценах. ає можливість інтерактивного управління прозорістю текстур за допомогою альфа-каналу.

Кожен шар текстури можна налаштовувати за рівнем прозорості прямо в режимі перегляду, що дозволяє точно контролювати, як кожна частина текстури взаємодіє з іншими шарами та матеріалами. Це особливо корисно при роботі з багат шаровими текстурами або декалями, де потрібно чітко визначити, які області будуть прозорими, а які матимуть повну непрозорість. Можливість інтерактивного налаштування дозволяє досягти точного рівня деталізації та контролю над кожним шаром текстури [32]. Розглянемо рисунок 3.6, на якому зображено граф-схему алгоритму для покращення текстур.

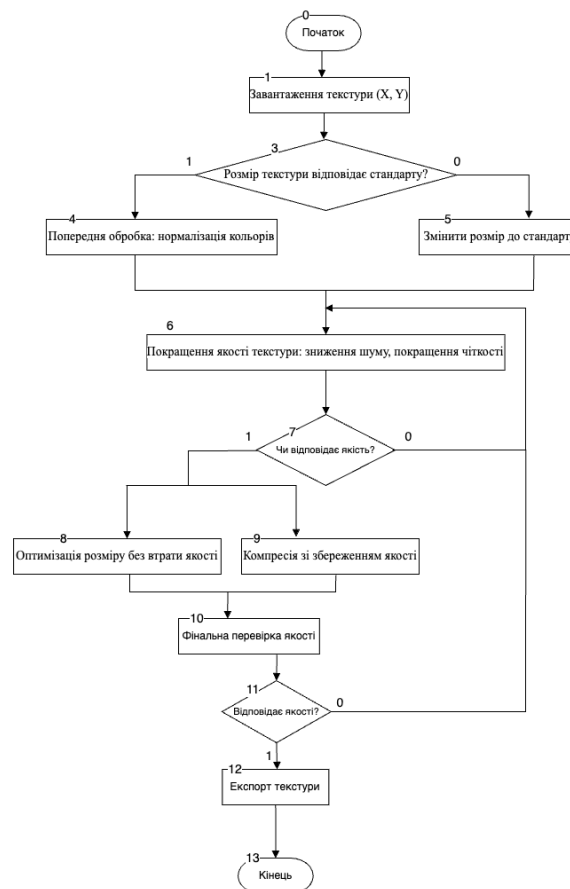


Рисунок 3.6 – Граф-схема алгоритму для покращення текстур

3.3 Розробка процедурних масок та багатошарових текстур

У плагіні розроблено функції масок порожнин. Маска порожнин автоматично визначає порожнини та заглиблення, дозволяючи користувачеві застосовувати різні ефекти текстуровання саме до цих зон. Це особливо корисно для створення реалістичних ефектів, як-от забруднень, зносу чи патьоків, що природно накопичуються в заглибленнях об'єктів. Таким чином, маска порожнин додає глибину та деталізацію до поверхонь, які виглядають більш реалістично та природно. Оновлена маска нормалей тепер функціонує в глобальному просторі, що дозволяє застосовувати ефекти незалежно від обертання об'єкта. Це значно спрощує створення текстур для об'єктів із різною орієнтацією, адже користувач може точно керувати напрямом нормалей, незалежно від локального простору об'єкта. Завдяки цій функції текстуровання стає більш зручним та інтуїтивним, особливо при роботі з комплексними сценами.

Розроблено метод створення нової текстурної карти для задач рендерингу. Карта "Variety" додає можливість додавати випадкові варіації до дифузної текстури [32-37]. Це дозволяє створювати природніші та менш однотипні поверхні, надаючи їм унікальних плям і кольорових відмінностей. Карта "Variety" особливо корисна для органічних матеріалів, як-от камінь, дерево чи тканини, які в природі мають неоднорідну структуру. Додавання цієї карти дозволяє користувачу швидко й легко створювати більш реалістичні текстури без необхідності вносити зміни вручну для кожної окремої ділянки текстури (рис. 3.7).

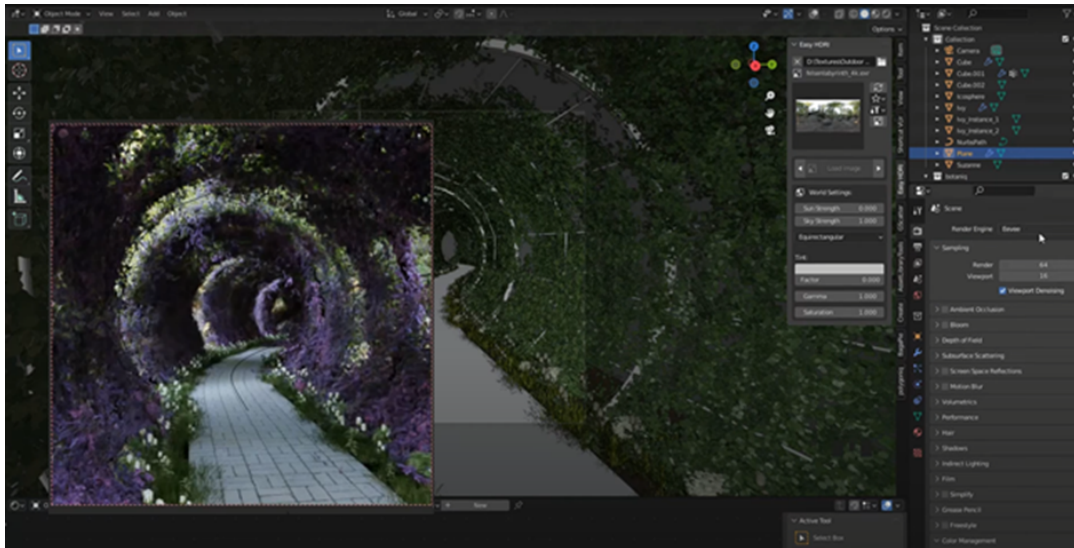


Рисунок 3.7 – Приклад використання карти Variety

У плагіні реалізована маска за кольором вертексів (Vertex Color Attribute Mask) [33],[38] дозволяє використовувати інформацію про кольори вертексів для створення масок, що забезпечує точне налаштування шарів та взаємодію між текстурами на основі кольорових даних об'єкта. Маска оточення (AO Mask) використовує карту Ambient Occlusion (AO) для управління змішуванням шарів на основі тіньових зон. Це дозволяє точно контролювати відображення текстур у темних областях, що додає глибини та об'єму поверхням об'єкта [34]. Це значно розширюють можливості текстурування, додаючи нові інструменти та функції, які покращують управління масками, декалями та варіативністю текстур. Орієнтоване на забезпечення більшої гнучкості, точності та інтерактивності в процесі створення матеріалів для 3D моделей, що особливо корисно для реалістичних та деталізованих проєктів у відеоіграх, анімації та візуалізації.

Розроблено маску оточення, що дозволяє краще контролювати ефекти затемнення на поверхні об'єкта, що створює враження глибокої взаємодії світла та тіні. Це допомагає робити об'єкти більш об'ємними та реалістичними навіть за умови мінімального використання полігонів, оскільки світло й тінь додають моделі додаткового відчуття глибини. Використання багатошарових текстур дозволяє створювати складні матеріали, які складаються з декількох різних

текстурних карт, накладених одна на одну. Користувач може інтерактивно налаштовувати кожен шар, змінюючи його параметри, такі як прозорість, глибина або ступінь змішування з іншими шарами. Функція відключення матеріалу дозволяє користувачеві видалити матеріал з об'єкта, не видаляючи його з бібліотеки. Це забезпечує зручне управління матеріалами, дозволяючи швидко очищати об'єкти від текстур і зберігати їх для подальшого використання.

Плагін покращує систему карт випуклості (bump maps) [3-4], що дозволяє тепер більш точно контролювати глибину кожного шару текстури для досягнення реалістичного багатошарового ефекту. Нова система дозволяє налаштовувати випуклість для кожного шару окремо, створюючи ефекти, які виглядають більш природно, зокрема при застосуванні до складних поверхонь (рис. 3.8).



Рисунок 3.8 – Використання Bump Map через плагін

Багатошарові текстури [5] дозволяють створювати надзвичайно деталізовані та реалістичні матеріали. Наприклад, можна застосувати один шар текстури для базової поверхні каменю, другий — для дрібних тріщин та пошкоджень, третій — для шарів бруду, і, нарешті, четвертий — для додавання

тонких відблисків. Завдяки можливості контролювати порядок і взаємодію цих шарів, користувач може створювати складні поверхні з різними властивостями та ефектами.

3.4 Оптимізація та випікання текстур

Оптимізація текстур [7],[9] для 3D моделей є важливим процесом, який спрямований на зниження використання ресурсів без значної втрати якості візуального сприйняття. Така оптимізація дозволяє покращити продуктивність, особливо у випадках роботи з обмеженими ресурсами, як-от у відеоіграх чи додатках для мобільних пристроїв. У плагіні розроблені різні методи оптимізацій під кожен текстурну карту. Об'єктно-орієнтована модель програмного модуля оптимізації, наведено на рис. 3.9.

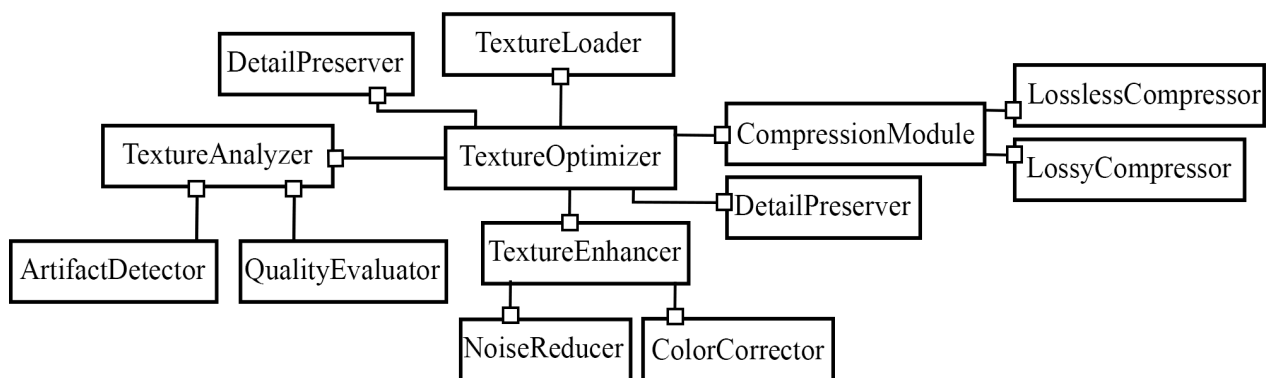


Рисунок 3.9 – Об'єктно-орієнтована модель програмного модуля

Основний підхід полягає у зменшенні розміру текстур, зниженні їхньої роздільної здатності або застосуванні алгоритмів компресії, які ефективно зберігають візуальні характеристики матеріалу. На (рис. 3.10) наведено діаграму структуру класів для роботи з текстурними картами.

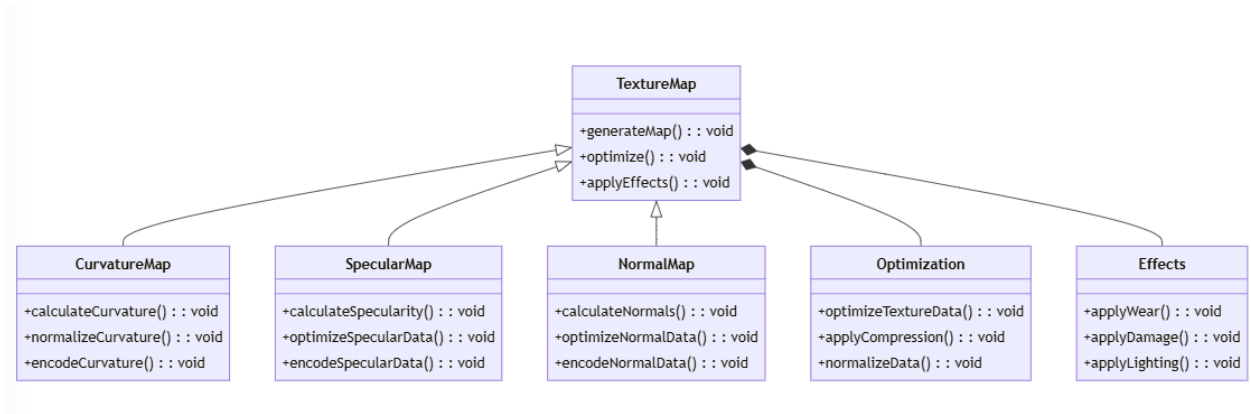


Рисунок 3.10 – Діаграму структуру класів для роботи з текстурними картами.

Ще одним ефективним методом є використання нормалізованих текстур та текстурних карт (як-от нормалі, альbedo та спекулярні карти), які, замість детального промальовування поверхні, допомагають створити ілюзію високої деталізації на моделі з низькою кількістю полігонів [36]. На (рис. 3.11) наведено роботу плагіну з картою Normal Map.



Рисунок 3.11 – Приклад використання Normal Map через плагін

Завдяки розробленому плагіну, можна отримати високоякісне зображення при мінімальному обсязі текстурних даних, що також сприяє оптимізації. Ще одним ефективним методом є використання нормалізованих текстур та

текстурних карт (як-от нормалі, альbedo та спекулярні карти), які, замість детального промальовування поверхні, допомагають створити ілюзію високої деталізації на моделі з низькою кількістю полігонів [36]. Завдяки цьому, можна отримати високоякісне зображення при мінімальному обсязі текстурних даних, що також сприяє оптимізації. Розроблений плагін оптимізує кожен вид карти, що допомагає забезпечити високу якість візуалізації при зниженні навантаження на систему.

Оптимізація текстурних карт для 3D моделей спрямована на покращення продуктивності системи за рахунок зменшення обсягу даних, необхідного для обробки графічним процесором. Основні види текстурних карт, які оптимізуються, включають альbedo (дифузні карти), нормалі, спекулярні карти та іноді карти рельєфу [37]. Кожна з них виконує специфічні функції у візуалізації поверхні, але може бути налаштована для зменшення ресурсоемності без втрати візуальної якості. Альbedo карти містять інформацію про кольорові властивості поверхні і відповідають за відтворення кольору без врахування ефектів освітлення. Їхня оптимізація включає зменшення роздільної здатності та застосування компресії. Розумно підібраний рівень стиснення дозволяє зберегти природність кольорів, одночасно знижуючи обсяг текстурних даних. Формула для оцінки оптимізації альbedo карти може виглядати так:

$$S_{\text{optimized}} = \frac{S_{\text{original}}}{(C \times R)} \quad (3.1)$$

де S — розмір оптимізованої текстури, S_{original} — початковий розмір текстури, C — коефіцієнт компресії, а R — коефіцієнт зниження роздільної здатності.

Спекулярні карти та карти рельєфу

Спекулярні карти, які визначають інтенсивність блиску, та карти рельєфу, що додають додаткову деталізацію на поверхні, також підлягають оптимізації.

Наприклад, спекулярні карти можуть бути знижені в якості для об'єктів, які знаходяться на периферії сцени або вдалині, оскільки деталі їхньої відбивної здатності менш помітні. Формула для розрахунку оптимізації спекулярної карти може бути такою:

$$P_{\text{optimized}} = \frac{P_{\text{original}}}{B \times D} \quad (3.2)$$

де P — розмір оптимізованої карти блиску, B — коефіцієнт стиснення, який залежить від якості блиску, необхідної для даної сцени, а D — дальність.

Програма підтримує роботу з п'ятьма індивідуальними шарами матеріалу для кожного об'єкта, що дозволяє створювати складні матеріали з різними текстурами та властивостями для кожного шару. Кожен шар може мати власні текстури та налаштування, що дає змогу моделювати матеріали різної природи, від металу до тканин, з високим рівнем реалістичності.

Розроблена функція в плагіні випікання (Baking V2) забезпечує вдосконалені можливості для збереження карт і текстур для моделі, що значно спрощує роботу над великими сценами та забезпечує кращу оптимізацію ресурсів. Функція випікання [23] зберігає всю інформацію про тіні, освітлення та кольори в одній текстурі, що дозволяє прискорити рендеринг сцени за рахунок зменшення обчислювальних ресурсів, необхідних для обробки реалістичних матеріалів. У процесі випікання програма автоматично генерує текстурні карти, які зберігають інформацію про освітлення, тіні та деталі поверхні (рис. 3.12).

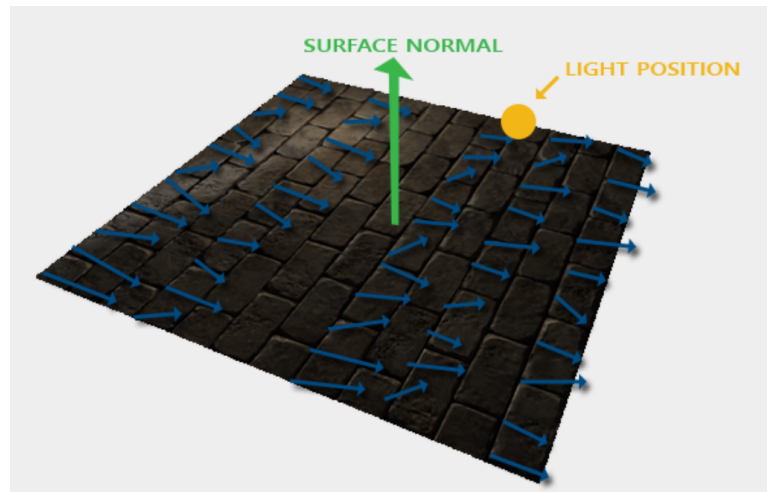


Рисунок 3.12 – Приклад використання карт зміщення

Випечені текстури можна використовувати як маски для подальшого текстурювання, що значно полегшує процес створення складних і реалістичних матеріалів [38]. Це дозволяє зосередитися на налаштуванні загальних параметрів моделі, а не на постійному оновленні динамічних елементів освітлення. Замість того щоб використовувати кілька окремих текстур для кожного ефекту (наприклад, для альбеда, нормалей, грубості, відбивання та інших карт), багатошарові текстури дозволяють об'єднувати кілька параметрів у один файл. Це знижує навантаження на відеокарту та зменшує кількість пам'яті, необхідної для обробки текстур у сцені. Під час випікання багатошарових текстур можна зберігати інформацію про кілька різних текстурних параметрів в одній текстурній карті, що дозволяє застосовувати ці дані в майбутньому для швидкого налаштування матеріалів. Наприклад, випечені карти АО можуть використовуватися для підвищення реалістичності шляхом додавання тіней у важкодоступних зонах, у той час як карти нормалей можуть зберігати деталі рельєфу для створення об'ємних ефектів на поверхні.

Функція копіювання [11], яка реалізована в плагіну, матеріалів і налаштувань UV-мапінгу дозволяє легко переносити текстурні карти та налаштування з одного об'єкта на інший. Це особливо корисно при роботі з великою кількістю об'єктів у сцені, де всі елементи повинні мати однакові

матеріали або текстури.

Плагін автоматично адаптує текстури та UV-мапи до нового об'єкта, що дозволяє заощадити час і зусилля, необхідні для ручного налаштування кожного об'єкта. Ця функція також корисна для оптимізації великих сцен, де важливо зменшити кількість унікальних матеріалів і текстур, що використовуються в проєкті. Завдяки копіюванню матеріалів можна легко підтримувати узгодженість текстур у всій сцені, одночасно зменшуючи навантаження на рендеринг [38].

На (рис. 3.13) наведено інтерфейс плагіну. Інтерфейс Інтуїтивно зрозумілий і гнучкий в налаштуванні, що забезпечує легкий доступ до інструментів та функцій, дозволяючи художникам і дизайнерам працювати швидше та точніше.

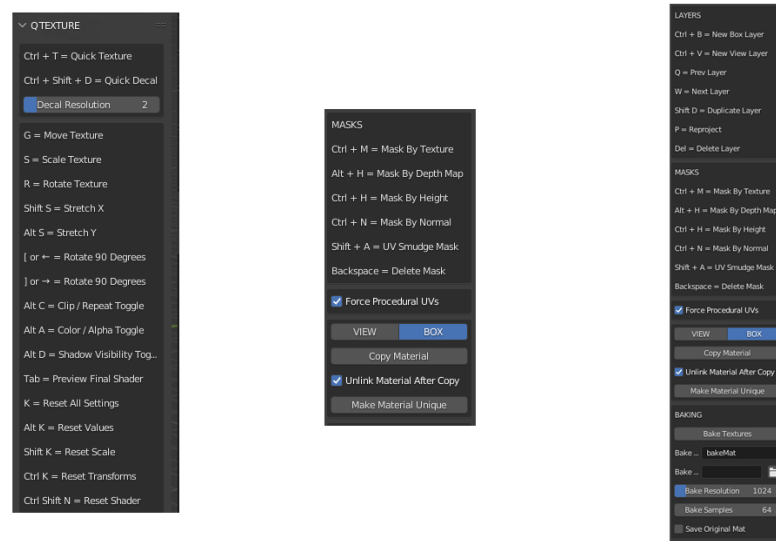


Рисунок 3.13 – Інтерфейс плагіну

Однією з ключових функцій, є меню швидкого доступу, яке відкривається за допомогою комбінації Ctrl+D. Це так зване "pie menu" дозволяє користувачу швидко вибирати необхідні інструменти, не шукаючи їх у численних вкладках інтерфейсу. Меню надає доступ до найбільш часто використовуваних функцій, таких як управління шарами, текстурами та налаштуваннями матеріалів, що значно прискорює робочий процес. Завдяки цьому інструменту користувачі можуть виконувати найважливіші дії в кілька кліків, без необхідності

переключатися між різними панелями та підменю. Це покращує загальну зручність роботи, особливо при виконанні рутинних завдань, що потребують частого доступу до основних інструментів.

Ще однією важливою функцією для розробка налаштувань користувача є можливість індивідуального налаштування швидкості миші залежно від рівня DPI (dots per inch) та розміру монітора. Це дозволяє користувачам із різними конфігураціями обладнання налаштовувати чутливість миші так, щоб вона відповідала їхнім робочим потребам. Для покращення інтуїтивності програми додані текстові підказки (tooltips), що забезпечують користувачу швидкий доступ до інформації про функції, спрощуючи навчання та підвищуючи продуктивність.

Користувачі з високим DPI або великими моніторами можуть зменшити швидкість переміщення курсора, щоб забезпечити точніше налаштування під час роботи з текстурами або об'єктами. Це покращує точність роботи, особливо під час виконання дрібних коригувань або роботи з деталізованими елементами моделі.

Для підвищення зручності використання плагіну, також є підтримка різних розмірів інтерфейсу для різних екранних роздільностей. Це означає, що інтерфейс програми адаптується до розміру екрана користувача, забезпечуючи зручність роботи незалежно від того, чи використовує він монітор стандартного розміру, великий екран або компактний ноутбук.

Ще однією корисною функцією для підвищення продуктивності та зручності роботи є автоматичне видалення матеріалів з інших об'єктів під час активації QuickTexture (QT). Ця функція дозволяє швидко очищати сцени від непотрібних матеріалів, що значно полегшує організацію проекту та зменшує кількість конфліктів між різними текстурами і матеріалами.

Автоматичне видалення матеріалів з інших об'єктів спрощує роботу з великими сценами, де багато моделей використовують різні текстури. Завдяки цьому користувач може уникнути дублювання матеріалів і зберегти ресурси системи, що особливо важливо для оптимізації робочих процесів у великих

проектах.

Плагін, розроблений для автоматизованої обробки текстурних карт у 3D-сценах, що дозволяє мінімізувати обсяг пам'яті, необхідний для зберігання та рендерингу [37], і знизити навантаження на GPU без негативного впливу на візуальне сприйняття. Діаграма класів наведено на (рис. 3.14).

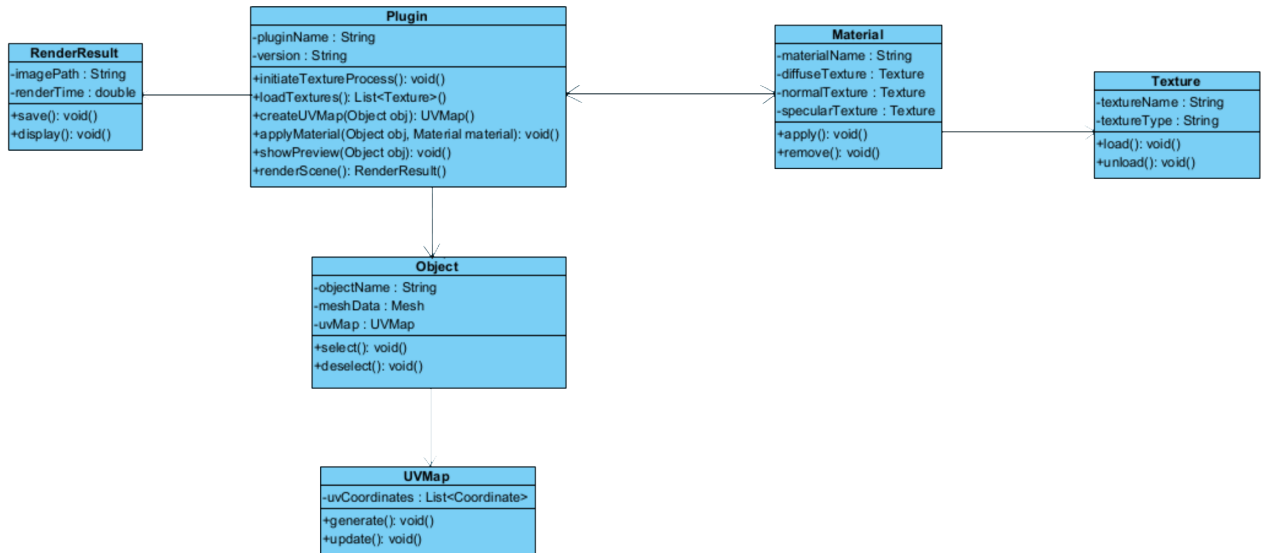


Рисунок 3.14 – Діаграма класів

Плагін використовує низку алгоритмів, які виявляють менш важливі деталі текстур та адаптують їхню роздільну здатність, забезпечуючи точне збереження найважливіших аспектів текстури. У розділі подано опис основних математичних моделей і формул, що використовуються в процесі оптимізації. Описані принципи стиснення даних, адаптивної компресії та динамічної оптимізації текстурних карт.

Оптимізації текстурної карти Normal Map через плагін

Нормаль [1-4] — це вектор, який перпендикулярний до поверхні в певній точці. Він використовується для розрахунку освітлення та відображення світла на поверхні.

Нормалізація нормалі: Для коректної роботи освітлення нормалі повинні бути нормалізовані, тобто мати довжину 1:

Формула:

$$n_z = \sqrt{1 - n_x^2 - n_y^2} \quad (3.3)$$

У стандартних Normal Map зазвичай зберігаються всі три компоненти нормалі (X, Y, Z) в трьох каналах текстури (R, G, B). Для оптимізації ми можемо зберігати лише X та Y компоненти в двох каналах (наприклад, R та G), а Z компоненту відновлювати в шейдері. Розглянемо основні

Відновлення Z компоненти нормалі

Формула:

$$n_z = \sqrt{1 - n_x^2 - n_y^2} \quad (3.4)$$

Теорема Піфагора в тривимірному просторі: Оскільки нормаль має довжину 1, суму квадратів її компонент можна представити як:

Виведення Z компоненти:

$$n_z = \sqrt{1 - n_x^2 - n_y^2} \quad (3.5)$$

а. Діапазон значень в текстурі:

- У текстурі значення компонент зберігаються в діапазоні $[0, 1]$.
- Для перетворення їх у діапазон $[-1, 1]$, використовуємо формулу:

б. Обробка негативних значень Z компоненти:

- Формула дає лише позитивне значення n .
- Якщо нормалі можуть мати негативну Z компоненту, потрібно визначити

знак.

- Один із способів — зберігати знак у третьому каналі текстури або використовувати біт знака в одному з каналів.

с. Запобігання помилок обчислень:

- Через обмежену точність обчислень може статися, що вираз під коренем стане від'ємним.

- Для уникнення цього використовуємо функцію `max`:

Приклад реалізації в шейдері

Отримуємо значення з текстури і перетворюємо їх у діапазон $[-1, 1]$

```
vec2 normalXY = texture(u_NormalMap, v_TexCoords).rg * 2.0 - 1.0;
float normalZ = sqrt(max(0.0, 1.0 - normalXY.x * normalXY.x - normalXY.y *
normalXY.y));
```

Переваги оптимізації:

- Зменшення розміру текстури: Використання двох каналів замість трьох зменшує обсяг текстури приблизно на 33%.

- Покращення продуктивності: Менше даних для зчитування з пам'яті прискорює рендеринг.

- Економія пам'яті: Менший розмір текстур дозволяє використовувати більше різних текстур або збільшити їх роздільну здатність.

Оптимізації текстурної карти Diffuse map

Перетворення RGB в колірний простір Y

Колірний простір Y дозволяє розділити зображення на яскравість та кольорові компоненти, що дає можливість зменшити обсяг кольорової інформації без значної втрати якості.

Формули перетворення з RGB в Y:

$$\begin{cases} Y' = 0.257 \cdot R + 0.504 \cdot G + 0.098 \cdot B + 16 \\ Cb' = -0.148 \cdot R - 0.291 \cdot G + 0.439 \cdot B + 128 \\ Cr' = 0.439 \cdot R - 0.368 \cdot G - 0.071 \cdot B + 128 \end{cases} \quad (3.6)$$

де:

- R — компоненти кольору в діапазоні $[0, 255]$.

- Y — яскравість (luma).

- C — кольорові різниці (chroma).

2. Зменшення роздільної здатності кольорових компонент

Людське око менш чутливе до деталей в кольорових компонентах, ніж до деталей в яскравості. Тому можна зменшити роздільну здатність Cb' та Cr' компонент без помітної втрати якості.

Формула зменшення роздільної здатності (субдискретизація):

$$\begin{aligned} Cb_{\text{sub}}(i, j) &= \frac{1}{4} \sum_{m=0}^1 \sum_{n=0}^1 Cb'(2i + m, 2j + n) \\ Cr_{\text{sub}}(i, j) &= \frac{1}{4} \sum_{m=0}^1 \sum_{n=0}^1 Cr'(2i + m, 2j + n) \end{aligned} \quad (3.7)$$

де:

- Cb_{sub} та Cr_{sub} — субдискретизовані кольорові компоненти.
- i, j — координати пікселів у зменшеній роздільній здатності.

3. Квантування кольорових компонент

Квантування зменшує кількість бітів, необхідних для зберігання кожної компоненти.

Формула квантування:

$$\begin{cases} Y_{\text{quant}} = \left\lfloor \frac{Y' - 16}{\Delta Y} \right\rfloor \\ Cb_{\text{quant}} = \left\lfloor \frac{Cb_{\text{sub}} - 128}{\Delta C} \right\rfloor \\ Cr_{\text{quant}} = \left\lfloor \frac{Cr_{\text{sub}} - 128}{\Delta C} \right\rfloor \end{cases} \quad (3.8)$$

де:

• ΔY та ΔC — кроки квантування для яскравості та кольорових компонент відповідно.

• $\lfloor x \rfloor$ — функція підлоги (floor), що повертає найбільше ціле число, не більше x .

4. Відновлення зображення в шейдері

Під час рендерингу необхідно відновити оригінальні значення компонент та перетворити їх назад в RGB.

Відновлення компонент:

$$\begin{cases} Y'' = Y_{\text{quant}} \cdot \Delta Y + 16 \\ Cb'' = Cb_{\text{quant}} \cdot \Delta C + 128 \\ Cr'' = Cr_{\text{quant}} \cdot \Delta C + 128 \end{cases} \quad (3.9)$$

Інтерполяція кольорових компонент (якщо необхідно):

Для повернення до оригінальної роздільної здатності можна використовувати бікубічну або білінійну інтерполяцію.

Зворотне перетворення з YCbCr в RGB:

$$\begin{cases} R = 1.164 \cdot (Y'' - 16) + 1.596 \cdot (Cr'' - 128) \\ G = 1.164 \cdot (Y'' - 16) - 0.392 \cdot (Cb'' - 128) - 0.813 \cdot (Cr'' - 128) \\ B = 1.164 \cdot (Y'' - 16) + 2.017 \cdot (Cb'' - 128) \end{cases} \quad (3.10)$$

Переваги цього методу

- Зменшення обсягу даних: Завдяки субдискретизації та квантуванню, розмір текстур може бути значно зменшений.
- Покращення продуктивності: Менший обсяг даних для передачі та обробки прискорює рендеринг.
- Збереження якості: Оптимізація враховує особливості сприйняття людиною кольору, що мінімізує помітні втрати якості.

Оптимізації текстурної карти Specular map

Specular Map [4] використовується для контролю над тим, як світло відбивається від поверхні. Вона визначає інтенсивність та колір блиску на різних ділянках моделі. Спекулярна інтенсивність (коефіцієнт відбивання): Визначає, наскільки яскравим буде відбите світло. Спекулярний експонент (шорсткість): Визначає розмиття блиску; більші значення дають гостріший блиск.

В плагіні текстурна карта Specular Map оптимізована, використовуючи математичні моделі, які дозволяють відновити спекулярні властивості поверхні

з меншої кількості даних.

Формула Фонга [1],[14] для спекулярного відбивання:

$$I_{\text{spec}} = k_s \cdot (R \cdot V)^\alpha \quad (3.11)$$

де:

- I_{spec} — інтенсивність спекулярного відбивання.
- k — спекулярний коефіцієнт (інтенсивність блиску).
- R — відбитий вектор світла.
- V — вектор до спостерігача (камери).
- α — спекулярний експонент (шорсткість поверхні).

Заміна текстурної карти функціональною залежністю

Замість зберігання повної Specular Map, можна використовувати математичну функцію для обчислення спекулярних властивостей на основі інших параметрів поверхні або координат.

Використання UV-координат для обчислення спекулярного коефіцієнта:

$$k_s(u, v) = A \cdot \sin(B \cdot u + C \cdot v + D) + E \quad (3.12)$$

де:

- $k_s(u, v)$ — спекулярний коефіцієнт в точці з координатами (u, v) .
- A, B, C, D , — параметри функції, що визначають форму та масштаб спекулярного коефіцієнта.

Кодування спекулярних параметрів у менше число каналів

Якщо Specular Map зберігає кілька параметрів (наприклад, інтенсивність та шорсткість), їх можна упакувати в один колірний канал або використовувати меншу кількість бітів.

Формула упаковки:

$$C_{\text{packed}} = (I_{\text{spec}} \ll 4) | (\alpha_{\text{spec}} \& 0x0F) \quad (3.13)$$

де:

- Cracked — упаковане значення в діапазоні [0, 255].
- Ispec — інтенсивність спекулярного відбивання в діапазоні [0, 15].
- α — спекулярний експонент в діапазоні [0, 15].
- $\ll$ — бітовий зсув вліво.
- $||$ — бітове або.
- $\&$ — бітове і.

Відновлення параметрів в шейдері:

$$I_{\text{spec}} = \frac{(C_{\text{packed}} \gg 4)}{15.0} \quad (3.14)$$

$$\alpha_{\text{spec}} = \frac{(C_{\text{packed}} \& 0x0F)}{15.0} \quad (3.15)$$

Використання моделей освітлення для відновлення деталей.

Замість зберігання детальної Specular Map, можна використовувати фізично обґрунтовані моделі освітлення, такі як модель Кука-Торренса [27] або GGX, та зберігати лише шорсткість поверхні.

Формула GGX для спекулярного відбивання:

Функція розподілу мікрофасет $D(h)$:

$$D(h) = \frac{\alpha^2}{\pi \left((n \cdot h)^2 (\alpha^2 - 1) + 1 \right)^2} \quad (3.16)$$

Функція Френеля $F(l, h)$:

$$F(l, h) = F_0 + (1 - F_0)(1 - l \cdot h)^5 \quad (3.17)$$

Геометрична функція затінення $G(l, v)$:

$$G(l, v) = G_{\text{Smith}}(l) \cdot G_{\text{Smith}}(v) \quad (3.18)$$

Де функція Сміта визначається як:

$$G_{\text{Smith}}(x) = \frac{N \cdot x}{(N \cdot x)(1 - k) + k} \quad (3.19)$$

Перевага цього методу заключається в упакуванні параметрів у меншу кількість каналів, що призводить до зменшення обсягів текстурних даних. Покращення продуктивності. Менша кількість даних для зчитування з пам'яті прискорює рендеринг. Збереження якості, використання фізично обґрунтованих моделей освітлення забезпечує реалістичний вигляд поверхонь. Оптимізація Specular Map шляхом зменшення обсягу збережених даних та використання математичних моделей для відновлення спекулярних властивостей поверхні дозволяє ефективно зменшити розмір текстур та покращити продуктивність рендерингу [38]. Детальне розуміння та правильне застосування наведених формул допоможуть досягти балансу між якістю візуалізації та ефективністю використання ресурсів у графічних додатках.

3.5 Висновки

У даному розділі розглянуто розроблений плагін, що забезпечує ефективну оптимізацію текстурних карт для 3D-графіки, та додає велику кількість функціональності при використанні. Детально розглянуто розроблені методи та функції плагіна, які спрямовані на удосконалення текстурних карт, автоматизувати процес обробки текстурних карт та підвищити продуктивність обробки графічних даних які використовуються для 3d-моделі, інтерфейс та налаштувань користувача у програмному забезпеченні для 3D моделювання. Розроблені функції автоматизованого управління матеріалами, зокрема

видалення матеріалів з інших об'єктів при активації плагіна, спрощують процес організації проєкту та зменшують кількість рутинних дій. Розробка нових функцій та оптимізація стабільності програми забезпечують надійну і безперебійну роботу, що є важливим при створенні складних 3D сцен та текстур.

Важливою складовою цієї розробки є інтеграція методів нормалізації та кодування текстурних даних, що дозволяє зменшити обсяг збережених текстур і підвищити швидкість рендерингу, що є критичним для сучасних графічних систем. Крім того, завдяки додатковим методам для обробки кривизни, плагін може застосовувати ефекти зносу, пошкодження та освітлення, створюючи більш реалістичні візуалізації. Результати реалізації продемонстрували ефективність розробленого плагіна у процесах обробки та оптимізації текстур, що підтверджується зменшенням використання пам'яті та збереженням високої якості зображень.

Завдяки впровадженню цих методів і функцій плагін підвищує ефективність робочого процесу та забезпечує гнучкість у використанні програмного забезпечення. Інтерактивні інструменти дозволяють користувачам прискорити роботу, зменшити час на технічні налаштування та зосередитися на творчій частині проєкту.

Отже, розроблені методи та функції плагіна створюють зручне, продуктивне та інтуїтивне середовище для роботи з 3D моделями, що відповідає сучасним вимогам до швидкості, якості та гнучкості у процесі моделювання і текстурування.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНИХ МЕТОДІВ ТЕКСТУРУВАННЯ

4.1 Програмний модуль для текстуровання тривимірних сцен

У магістерській кваліфікаційній роботі було створено програмний модуль для текстуровання тривимірних сцен та їх подальшого аналізу порівняльного характеру. Цей модуль оптимізовано для роботи з простими низькополігональними тривимірними об'єктами, розташованими на близькій відстані від спостерігача, що забезпечує краще зображення перспективи під час текстуровання.

Основні функції цього програмного модуля включають:

- а) завантаження та архівацію вихідних текстур;
- б) вибір простих низькополігональних об'єктів з представленого переліку;
- в) застосування текстуровання згідно обраного методу;
- г) модифікація розташування тривимірного об'єкта в просторі;
- д) корекція центру перспективи;
- е) обрахунок загальної кількості тактів роботи мікропроцесора для основних операцій у використаному методі текстуровання.

Програмний модуль побудовано на основі бібліотеки `idx3d`, яка має базові функції для роботи з 3D-моделями та тривимірною геометрією.

Бібліотека `idx3d` пропонує Python-розробникам зручний інструмент для створення і маніпулювання тривимірними сценами, що є особливо корисним для невеликих проектів або експериментів, де потрібен швидкий доступ до 3D-графіки. Ця бібліотека підтримує базові 3D-функції, такі як додавання об'єктів, регулювання освітлення, налаштування камери, а також маніпулювання об'єктами для їх переміщення, обертання та масштабування в просторі [38]. Це дає можливість з легкістю втілювати візуальні ідеї без потреби занурюватися в складності більш великих і складних систем, таких як OpenGL.

Інструменти рендерингу, які входять в склад `idx3d`, оптимізовані для ефективної роботи, забезпечуючи здатність до швидкої візуалізації і анімації, що робить її відмінним вибором для розробки прототипів або маленьких ігор. Також, підтримка текстур і різноманітних видів освітлення дозволяє створювати деталізованіші та реалістичніші сцени, не вдаючись до використання важких графічних движків. Хоча `idx3d` не може змагатися з повнофункціональними 3D конвеєрами з точки зору глибини і контролю, вона пропонує достатньо можливостей для багатьох типів застосунків, де основна вимога — простота і швидкість розробки.

Для реалізації програмного модуля було використано мову програмування Python [35]. Це високорівнева мова програмування Python, яка підтримується великою кількістю бібліотек, що в значній мірі полегшує процедуру розробки програм.

Python може бути дуже підходящим інструментом для задач рендерингу з декількох причин. Python відомий своєю високою продуктивністю і оптимізацією. Це особливо важливо в рендерингу, де час обробки і відклик є критичними факторами.

Python дозволяє розробникам оптимально використовувати глибше системні ресурси, такі як пам'ять та обробка даних. Це може суттєво покращити ефективність в реалізації складних алгоритмів рендерингу [38].

Python дозволяє використовувати широкий спектр програмних інтерфейсів і технологій. Наприклад, можна легко інтегрувати DirectX [6],[38],[39] або OpenGL для рендерингу графіки, що є ключовим для багатьох додатків в галузі ігор та симуляцій.

Існує багато бібліотек та інструментів, оптимізованих для Python, які можуть бути використані для формування тривимірних графічних зображень, включаючи математичні бібліотеки, функції для роботи з графікою та ін.

Python добре інтегрується з іншими мовами програмування і платформами, що дозволяє легко використовувати код, написаний на інших мовах, або інтегрувати сторонні плагіни і бібліотеки [39].

Існує велика кількість ресурсів, форумів, та інших джерел знань, які можуть допомогти розробникам у вирішенні проблем та вивченні нових підходів в рендерингу.

У створеній програмі графічний конвеєр має спрощену структуру порівняно з типовими графічними конвеєрами у сфері комп'ютерної графіки. Він включає етап геометричних трансформацій, де проводяться афінні та проєкційні перетворення, що дозволяють змінювати положення об'єкта у просторі, та етап рендерингу. Вормування графічних зображень у програмі значно спрощений і не включає видалення невидимих поверхонь [39] за допомогою методу z-буфера, оскільки передбачається, що невидимі поверхні не будуть відображені.

Програмний модуль надає можливість інтеграції розроблених методів текстуровання та розширення користувацького інтерфейсу для дослідження як наявних, так і інноваційних підходів до текстуровання тривимірних графічних об'єктів.

Графічний інтерфейс розробленої програми розділений на п'ять базових функціональних секцій, що представлені на рисунку 4.1.

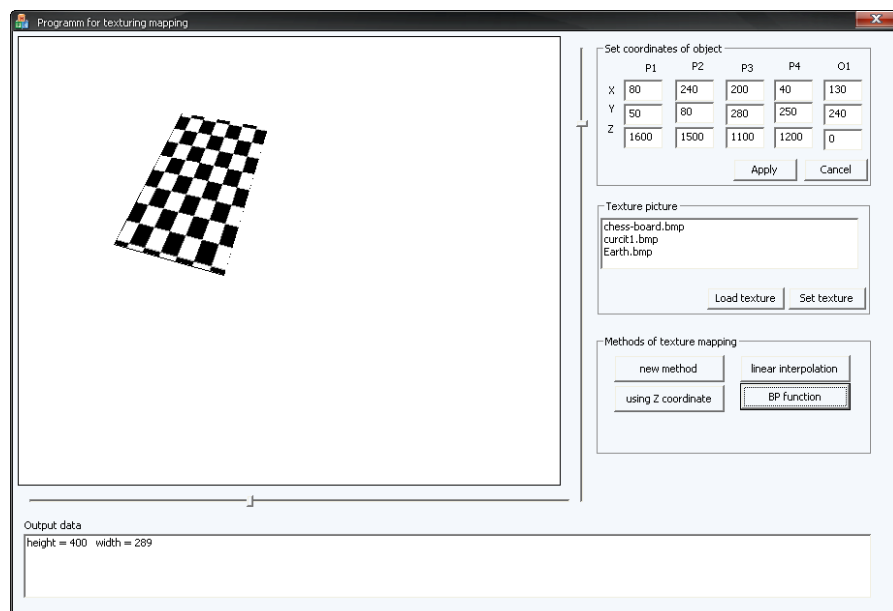


Рисунок 4.1 – Інтерфейс програмного додатку

Перша секція інтерфейсу включає набір полів типу CEdit, які дозволяють змінювати координати об'єкта та центру перспективи у тривимірному просторі, що ілюстровано на рисунку 4.2. На даний момент програмний модуль підтримує відображення лише однієї площини у тривимірному просторі. Зміни координат вершин здійснюються через спеціально призначені для цього текстові поля.

	P1	P2	P3	P4	O1
X	80	240	200	40	130
Y	50	80	280	250	240
Z	1600	1500	1100	1200	0

Рисунок 4.2 – Поля редагування для зміни параметрів розміщення об'єкта

Друга секція користувацького інтерфейсу містить ряд кнопок, які керують процесом текстурування. Ці елементи керування представлені на рисунку 4.3.

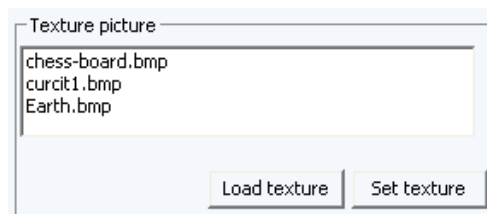


Рисунок 4.3 – Поля керування процесом текстурування

Вибрати текстуру для відображення на тривимірну поверхню можна за допомогою кнопки "Load texture". Після її натискання відкривається діалогове вікно, де користувачу необхідно вибрати файл у форматі *.BMP, як зображено на рисунку 4.4.

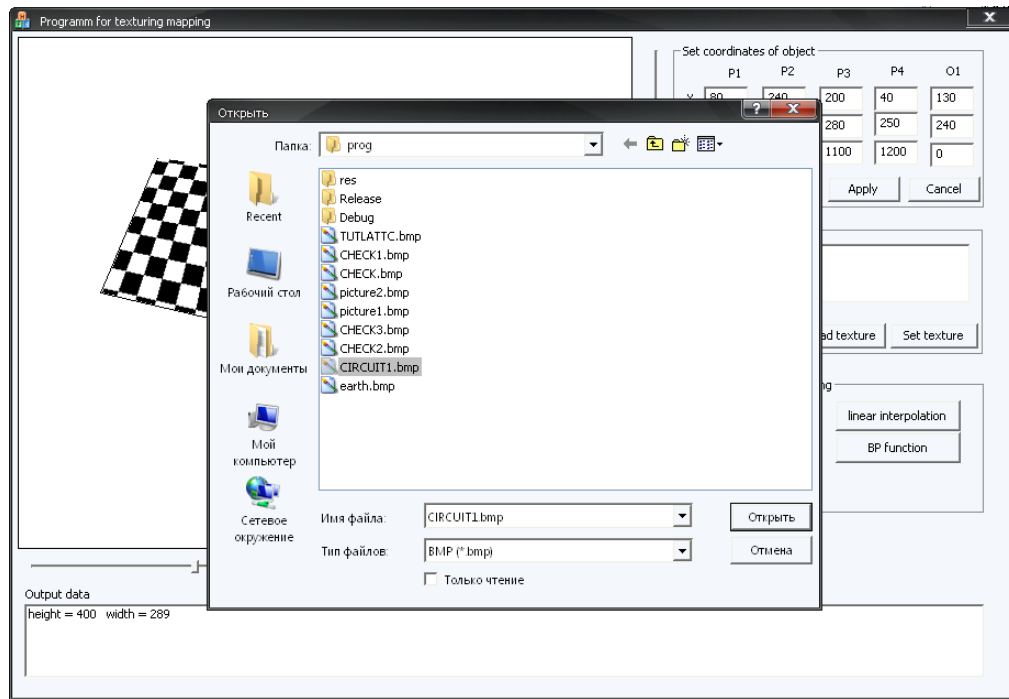


Рисунок 4.4 – Завдання текстури для накладання на тривимірну площину

Четверта частина інтерфейсу відповідає за відображення загальної кількості тактів роботи мікропроцесора для основних операцій, які використовуються у вибраному методі текстурування об'єкта.

П'ята частина інтерфейсу забезпечує можливість вибору методу текстурування для тривимірного об'єкта (рис. 4.5).

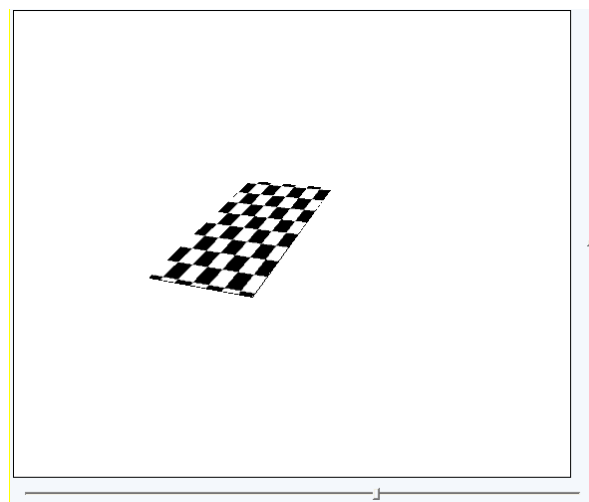


Рисунок 4.5 – Поле для виведення сформованих зображень

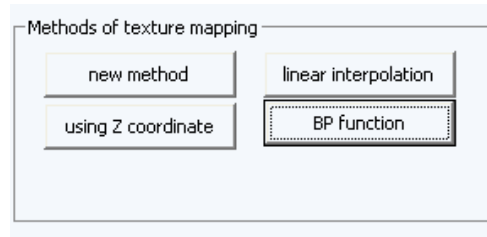


Рисунок 4.6 – Секція для вибору методу накладання текстур

Лістинг основних програмних модулів наведено в додатку А.

Для розробки засобів текстуровання використано такі ресурси:

1. Операційна система: Windows 10
2. Процесор: Intel Core i5
3. Графічний відеоадаптер:
 - NVIDIA GeForce GTX 1660 підтримкою OpenGL і DirectX.
4. Оперативна пам'ять:
 - Мінімум 8 Gb
5. Накопичувач:
 - SSD (твердотільний накопичувач) є обов'язковим для швидшого доступу до даних.

Ця конфігурація дозволить ефективно виконувати завдання моделювання, тестування методів зафарбовування та інших ресурсомістких обчислень.

Розроблений програмний модуль надає можливість тестування різноманітних методів текстуровання.

Програмний модуль створено спеціально для текстуровання тривимірних об'єктів з визначенням спекулярної складової кольору. Його основні функції включають: а) імпорт тривимірних графічних об'єктів з файлів *.3DS; б) введення текстурних файлів у форматі *.JPG; в) налаштування моделі освітлення для вибраного методу текстуровання, що враховує z-координату при перспективному зображенні об'єкта.

Цей модуль побудований з використанням бібліотеки `idx3d`, яка забезпечує базові можливості для роботи з 3D-моделями та тривимірною геометрією. Об'єктно-орієнтована модель програмного модуля показана на рисунку 4.7.

Графічний конвеєр у програмі включає модулі геометричних перетворень, що включають афінні та видові перетворення для переміщення об'єкта в просторі, а також етап рендерингу. На цьому етапі виконується видалення невидимих поверхонь за допомогою методу *z*-буфера та текстуровання об'єкта за обраним методом.

Програмний модуль також передбачає можливість додавання нових дистрибутивних функцій і методів текстуровання, а також розширення інтерфейсу користувача для дослідження як наявних, так і інноваційних підходів до текстуровання тривимірних графічних об'єктів.

Графічний інтерфейс розробленої програми складається з трьох функціональних частин, що представлені на рисунку 4.8.

Перша частина інтерфейсу користувача включає набір кнопок для управління процесом накладання текстур (рис. 4.9). Вибір об'єктів для текстуровання здійснюється з використанням кнопки “Вибрати об'єкт...”. З'являється діалогове вікно, де слід обрати *.3DS файл, в якому міститься необхідний об'єкт.

Кнопка “Пауза” призупиняє процес динамічного відображення об'єктів.

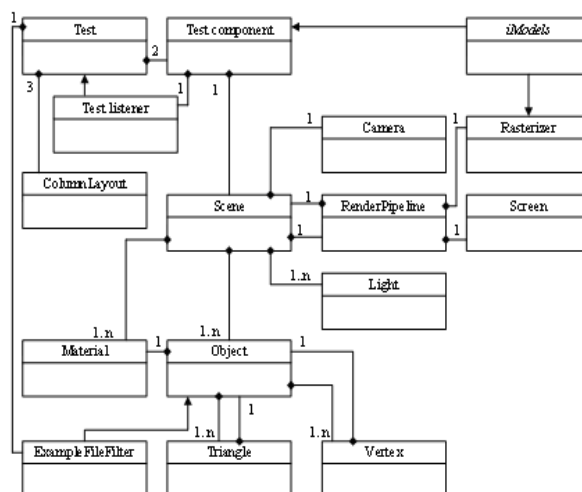


Рисунок 4.7 – Взаємодія класів

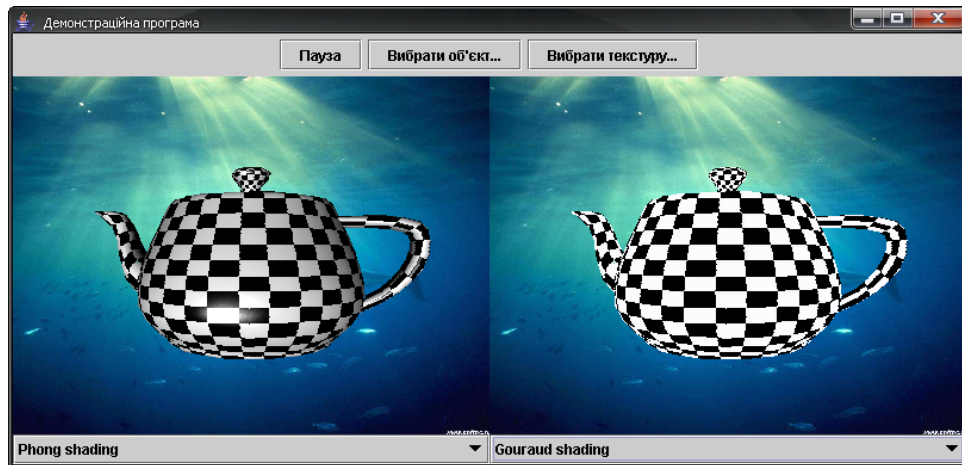


Рисунок 4.8 – Вигляд інтерфейсу

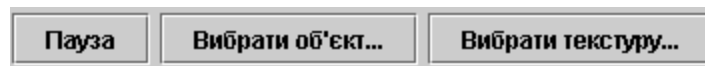


Рисунок 4.9 – Кнопки для управління текстурування

Друга секція інтерфейсу використовується для відображення згенерованих тривимірних об'єктів, як це показано на рисунку 4.10.



Рисунок 4.10 – Кнопки для вибору методу текстурування

У програмі використано багатопотокову модель класів, схематичне зображення якої представлене на рисунку 4.11.

Багатопотокова модель, яка реалізована у програмному додатку, забезпечує ефективну обробку даних шляхом розподілу завдань між кількома потоками виконання. Це дозволяє оптимізувати використання ресурсів обчислювальної системи і підвищити загальну продуктивність програми.

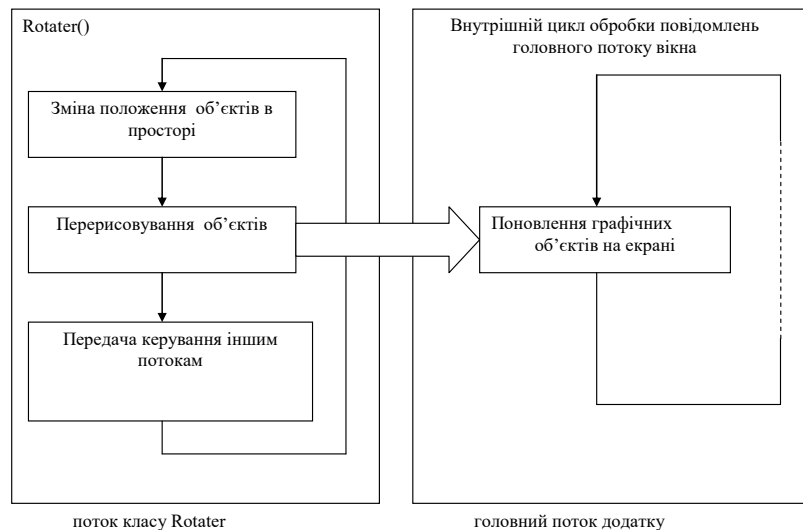


Рисунок 4.11 – Модель управління потоками

Лістинг основних програмних модулів наведено в додатку Б.

Розроблений програмний модуль дозволяє виконати моделювання і тестування різних методів текстуровання та різних моделей освітлення.

4.2 Реалізація та експериментальні дослідження розроблених методів текстуровання

Для покращення точності колірного сприйняття застосовують вагові коефіцієнти під час анізотропної фільтрації, яка має наступні кроки:

1. Встановлення параметрів еліпса з застосуванням текстурних координат ділянки.
2. Визначення координат кутів прямокутника, що охоплює еліпс.
3. Перевірка відповідності кожного текселя в обмежувальному прямокутнику рівнянню еліпса для встановлення текселів, які утворюють проекцію пікселя на текстуру.
4. Розрахунок вагового коефіцієнта для кожного текселя, що відповідає еліпсу, за обраною ваговою функцією.
5. Реалізація множення інтенсивності кольору кожного текселя, що відповідає рівнянню еліпса, на його ваговий коефіцієнт.

6. Підрахунок загальної суми добутків інтенсивностей кольорів текселів, що відповідають еліпсу, помножені на вагові коефіцієнти.

7. Обрахунок кольору ділянки шляхом ділення загальної суми добутків інтенсивностей кольорів текселів на суму вагових коефіцієнтів.

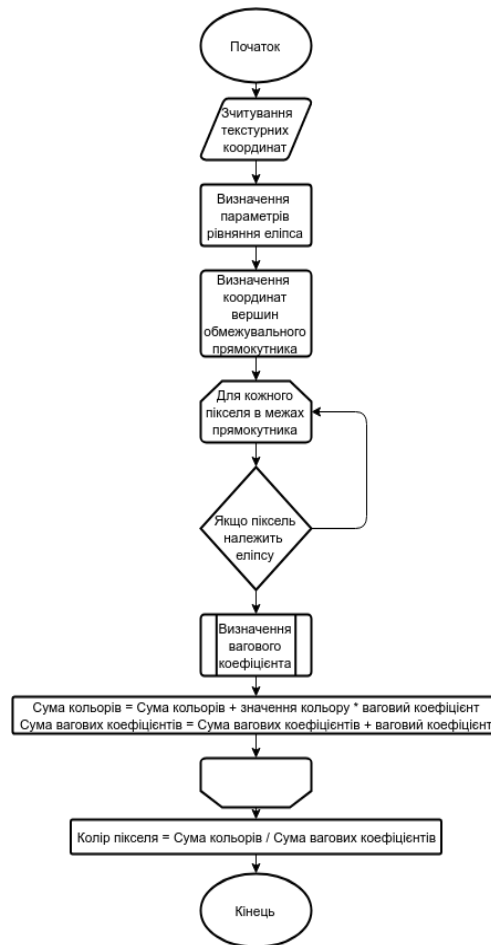


Рисунок 4.12 – Етапи нанесення текстури з використанням вагових коефіцієнтів

У додатку представлено код шейдерної функції, написаної на GLSL, для визначення вагового коефіцієнта. Функція аналізує такі вхідні дані: координати текселя, положення центру еліпса (відповідає текстурним координатам пікселя), а також довжини осей еліпса. В результаті виконання функція забезпечує виведення значення вагового коефіцієнта для кожного текселя.

Вихідний коду вагової функції для шейдерної програми на мові GLSL, яка використовує текстурні карти для визначення вагових коефіцієнтів, подано в додатку. Функція приймає наступні вхідні параметри: координати текселя, текстурні координати центру еліпса, а також координати векторів осей еліпса.

Кожен тексель у текстурі зберігається як 4-байтне ціле значення. Під час введення із текстури, шейдерна програма інтерпретує кожне значення текселя як колір у форматі RGBA, де кожен піксел цього формату представляє собою 4-вимірний вектор, кожний елемент якого має 1 байт. Згідно зі стандартами OpenGL, для перетворення представлення текселя з кольорового формату RGBA у ціле значення використовується формула для перетворення:

$$W = (A \ll 24) \cup (B \ll 16) \cup (G \ll 8) \cup R, \quad (4.1)$$

де, R, G, B, A - складові RGBA; W - ваговий коефіцієнт текселя.

Формати TGA та PNG були обрані для запам'ятовування вагових текстурних карт через їхню здатність до чотирибайтного кодування зображень без втрачання точності вагових значень.

Середньоквадратична похибка для зображення, створеного за допомогою вагових карт, становить лише 0,000708 у порівнянні з зображенням, отриманим за допомогою вагових функцій, розрахованих в реальному часі. Це свідчить про мінімальні візуальні відхилення, які не помітні на око.

Метод анізотропної фільтрації, який використовується у зв'язку з parallax mapping, передбачає такі етапи (рис. 4.13).

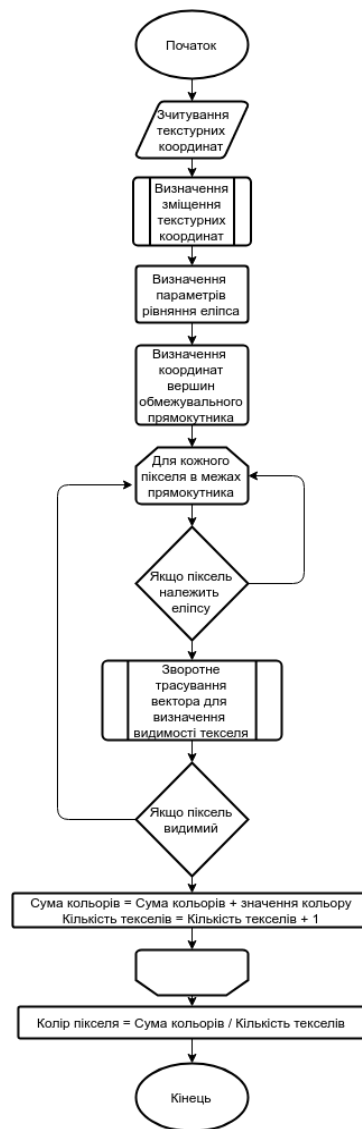


Рисунок 4.13 – Етапи рельєфного текстуровання

4.3 Висновки

1. Розглянуто програмну реалізацію запропонованих методів текстуровання.
2. Розроблено програмний модуль для текстуровання тривимірних сцен.
3. Розглянуто особливості програмної реалізації запропонованих методів текстуровання.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методів і програмних засобів текстурівання для задач рендерингу» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект [24],[25]. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

- 1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;
- 2) розраховано витрати на здійснення науково-технічної розробки;
- 3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів і програмних засобів текстурівання для задач

рендерингу» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [24].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкуренція немає

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витратити значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	3	4
2. Ринкові переваги (наявність аналогів)	4	3	3
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	4	3
6. Ринкові перспективи (розмір ринку)	3	3	4
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	3	3	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	4
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	38	37	41
Сума балів	3	3	4
Середньоарифметична сума балів $СБ_c$	38,7		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [24].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів $СБ_c$, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41...48	Високий
31...40	Вище середнього
21...30	Середній
11...20	Нижче середнього
0...10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів текстурування для задач рендерингу» становить 38,7 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Розробка методів і програмних засобів текстування для задач рендерингу», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

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

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників (Z_o) розраховуємо у відповідності до посадових окладів працівників, за формулою [24]:

$$Z_o = \sum_{i=1}^k \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

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

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, $T_p=21$ день.

$$Z_o = 19100,00 \cdot 21 / 21 = 19100,00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	19100	909,52	21	19100,00
Інженер-розробник програмного забезпечення	15100	719,05	21	15100,00
Консультант	7800	371,43	15	5571,43
Всього				39771,43

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_p$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_p = \sum_{i=1}^n C_i \cdot t_i, \quad (5.2)$$

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

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

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

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), прийmemo $M_M=8000,00$ грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [24];

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

T_p – середнє число робочих днів в місяці, приблизно $T_p = 21$ день;

$t_{зм}$ – тривалість зміни, год.

$$C_1 = 8000,00 \cdot 1,10 \cdot 1,65 / (21 \cdot 8) = 86,43 \text{ грн.}$$

$$З_{р1} = 86,43 \cdot 8,00 = 691,43 \text{ грн.}$$

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Підготовка робочого місця дослідника	8	2	1,1	86,43	691,43
Інсталяція програмного забезпечення	3	5	1,7	133,57	400,71
Тестування	40	5	1,7	133,57	5342,86
Всього					6435,00

Додаткова заробітна плата дослідників та робітників

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

$$З_{дод} = (З_o + З_p) \cdot \frac{H_{дод}}{100\%}, \quad (5.4)$$

де $H_{дод}$ – норма нарахування додаткової заробітної плати. Прийmemo 11%.

$$З_{дод} = (39771,43 + 6435,00) \cdot 11 / 100\% = 5082,71 \text{ грн.}$$

5.2.2 Відрахування на соціальні заходи

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

$$З_n = (З_o + З_p + З_{дод}) \cdot \frac{H_{zn}}{100\%} \quad (5.5)$$

де H_{zn} – норма нарахування на заробітну плату. Приймаємо 22%.

$$З_n = (39771,43 + 6435,00 + 5082,71) \cdot 22 / 100\% = 11283,61 \text{ грн.}$$

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

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

$$M = \sum_{j=1}^n H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^n B_j \cdot Ц_{ej}, \quad (5.6)$$

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

n – кількість видів матеріалів;

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

K_j – коефіцієнт транспортних витрат, ($K_j = 1,1 \dots 1,15$);

B_j – маса відходів j -го найменування, кг;

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

$$M_1 = 2 \cdot 1750,00 \cdot 1,1 - 0,000 \cdot 0,00 = 385,0 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір	175	2	0	0	385
Канцелярське приладдя (набір офісного працівника)	175	2	0	0	385
Flesh-пам'ять	130	1	0	0	143
Всього					913

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі (K_6), які використовують при проведенні НДР на тему «Розробка методів і програмних засобів текстування для задач рендерингу» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

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

Витрати на «Спецустаткування для наукових (експериментальних) робіт» в роботі на тему «Розробка методів і програмних засобів текстування для задач рендерингу» відсутні.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

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

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{npz} = \sum_{i=1}^k \Pi_{inpr} \cdot C_{npz.i} \cdot K_i, \quad (5.7)$$

де Π_{inpr} – ціна придбання одиниці програмного засобу даного виду, грн;

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

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

k – кількість найменувань програмних засобів.

$$B_{npz} = 7299,00 \cdot 2 \cdot 1,1 = 16349,76 \text{ грн.}$$

Отримані результати зведемо до таблиці:

Таблиця 5.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	2	7299	16349,76
Прикладний пакет Microsoft Office 2019	2	5753	12886,72
Комп'ютерна програма для розрахунку оптимізації карт для задач рендерингу	2	4733	10601,92
Всього			39838,4

5.2.7 Амортизація обладнання, програмних засобів та приміщень

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

$$A_{обл} = \frac{Ц_б}{T_г} \cdot \frac{t_{вик}}{12}, \quad (5.8)$$

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

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

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

$$A_{обл} = (30000,00 \cdot 1) / (3 \cdot 12) = 833,33 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук ASUS Vivobook 15	30 000	3	1	833,33
Ноутбук Asus ROG Strix G16 Eclipse	55 000	3	1	1527,78
Робоче місце дослідника	210000	5	1	3500,00
Оргтехніка	6000	4	1	125,00

Мультимедійна проекційна система	200000	5	1	3333,33
ОС Windows 11	16349,76	2	1	681,24
Комп'ютерна програма для розрахунку оптимізації карт для задач рендерингу	10601,92	2	1	441,75
Прикладний пакет Microsoft Office 2019	12886,72	2	1	536,95
Пристрої виводу інформації	6875	4	1	143,23
Всього				11122,61

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^n \frac{W_{yi} \cdot t_i \cdot C_e \cdot K_{\text{внi}}}{\eta_i}, \quad (5.9)$$

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

t_i – тривалість роботи обладнання на етапі дослідження, год;

C_e – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), прийmemo $C_e = 10,50$ грн;

$K_{\text{внi}}$ – коефіцієнт, що враховує використання потужності, $K_{\text{внi}} < 1$;

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

$$B_e = 0,5 \cdot 168,0 \cdot 10,50 \cdot 0,95 / 0,97 = 863,81 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлен а потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук ASUS Vivobook 15	0,5	168	863,81
Ноутбук Asus ROG Strix G16 Eclipse	0,5	168	863,81
Робоче місце дослідника	0,15	168	259,14
Пристрої виводу інформації	0,03	0	0,00
Оргтехніка	0,45	20	92,55
Засоби передачі даних	0,18	20	37,02
Мультимедійна проекційна система	0,11	45	50,90
Всього			2167,25

5.2.9 Службові відрядження

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

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

$$B_{cb} = (Z_o + Z_p) \cdot \frac{H_{cb}}{100\%}, \quad (5.10)$$

де H_{cb} – норма нарахування за статтею «Службові відрядження», прийmemo $H_{cb} = 20\%$.

$$B_{cv} = (39771,43 + 6435,00) \cdot 20 / 100\% = 9241,29 \text{ грн.}$$

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

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

$$B_{cn} = (Z_o + Z_p) \cdot \frac{H_{cn}}{100\%}, \quad (5.11)$$

де H_{cn} – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $H_{cn} = 35\%$.

$$B_{cn} = (39771,43 + 6435,00) \cdot 35 / 100\% = 16172,25 \text{ грн.}$$

5.2.11 Інші витрати

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

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

$$I_s = (Z_o + Z_p) \cdot \frac{H_{is}}{100\%}, \quad (5.12)$$

де H_{is} – норма нарахування за статтею «Інші витрати», приймемо $H_{is} = 50\%$.

$$I_s = (39771,43 + 6435,00) \cdot 50 / 100\% = 23103,21 \text{ грн.}$$

5.2.12 Накладні (загальновиробничі) витрати

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

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

$$B_{нзв} = (3_o + 3_p) \cdot \frac{H_{нзв}}{100\%}, \quad (5.13)$$

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

$$B_{нзв} = (39771,43 + 6435,00) \cdot 120 / 100\% = 55\,447,71 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = 3_o + 3_p + 3_{доо} + 3_n + M + K_v + B_{спец} + B_{прз} + A_{обл} + B_e + B_{св} + B_{сп} + I_v + B_{нзв}. \quad (5.14)$$

$$B_{заг} = 314896,61 \text{ грн.}$$

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

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, прийmemo $\eta = 0,8$.

$$ЗВ = 314896,61 / 0,8 = 393620,76 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 2000 користувачів/рік;

2-й рік – 1800 користувачів/рік;

3-й рік – 1500 користувачів/рік.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, прийmemo 1000 користувачів;

C_6 – вартість програмного продукту у році до впровадження результатів розробки, прийmemo 1800,00 грн /за рік користування;

$\pm \Delta C_6$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, прийmemo 7700,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta \Pi_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [24]:

$$\Delta\Pi_i = (\pm\Delta C_o \cdot N + C_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 30\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$$\Delta\Pi_1 = (1000,00 \cdot 7700,00 + 9500 \cdot 2000) \cdot 0,83 \cdot 0,3 \cdot (1 - 0,18/100\%) = 5471310,6 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta\Pi_2 = (1000,00 \cdot 7700,00 + 9500 \cdot (2000 + 1800)) \cdot 0,83 \cdot 0,3 \cdot (1 - 0,18/100\%) = 8975408,4 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta\Pi_3 = (1000,00 \cdot 7700,00 + 9500 \cdot (2000 + 1800 + 1500)) \cdot 0,83 \cdot 0,3 \cdot (1 - 0,18/100\%) = 11895489,9 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків $\Pi\Pi$, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$\Pi\Pi = \sum_{i=1}^T \frac{\Delta\Pi_i}{(1 + \tau)^i}, \quad (5.17)$$

де $\Delta\Pi_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau=0,25$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 5471310,6/(1+0,25)^1 + 8975408,4/(1+0,25)^2 + 11895489,9/(1+0,25)^3 = 16211800,68 \text{ грн.}$$

Величина початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{инв} \cdot 3B, \quad (5.18)$$

де $k_{инв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{инв}=4$;

$3B$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 393620,76 грн.

$$PV = k_{инв} \cdot 3B = 4 \cdot 393620,76 = 1574483,031.$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.19)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 16211800,68 грн;

PV – теперішня вартість початкових інвестицій, 1574483,031 грн.

$$E_{abc} = PPP - PV = 16211800,68 - 1574483,031 = 14637317,65 \text{ грн.}$$

Внутрішня економічна дохідність інвестицій E_g , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_g = T_{ж} \sqrt[3]{1 + \frac{E_{abc}}{PV}} - 1, \quad (5.20)$$

де E_{abc} – абсолютний економічний ефект вкладених інвестицій, 14637317,65 грн;

PV – теперішня вартість початкових інвестицій, 1574483,031 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_g = T_{ж} \sqrt[3]{1 + \frac{E_{abc}}{PV}} - 1 = (1 + 14637317,65 / 1574483,031)^{1/3} - 1 = 1,18.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій τ_{min} :

$$\tau_{min} = d + f, \quad (5.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні $d = 0,11$;

f – показник, що характеризує ризикованість вкладення інвестицій, прийmemo 0,25.

$\tau_{min} = 0,11 + 0,25 = 0,36 < 1,18$ свідчить про те, що внутрішня економічна дохідність інвестицій E_g , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою

«Розробка методів і програмних засобів текстуровання для задач рендерингу» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_г}, \quad (5.22)$$

де $E_г$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 1,18 = 0,85 \text{ року.}$$

$T_{ок} < 3$ -х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів текстуровання для задач рендерингу» становить 38,7 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,85 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів і програмних засобів текстуровання для задач рендерингу».

ВИСНОВКИ

1. Проаналізовано методи та засоби текстурювання тривимірних графічних сцен. Обґрунтовано необхідність підвищення продуктивності накладання текстур для зменшення часу формування зображень.

Існуючі методи текстурювання не задовольняють потреби багатьох галузей, що використовують тривимірну комп'ютерну графіку. Це ставить завдання розробки нових, більш ефективних методів і засобів текстурювання, які забезпечать високий рівень деталізації при мінімальних витратах обчислювальних ресурсів.

2. Отримано формули для базових перетворень при генерації тривимірних тривимірних об'єктів, що дозволяє їх використовувати для розробки нових методів і засобів.

Подальшого розвитку отримав метод текстурювання, відмінність якого від відомих полягає у визначенні координати текселя за картою вагових коефіцієнтів, що дозволяє зчитувати вагові коефіцієнти безпосередньо з текстурної карти, не розраховуючи їх. Встановлено, що застосування вагової карти замість обчислення складних формул підвищує продуктивність до 25%.

Вперше запропоновано метод текстурювання, особливість якого полягає у використанні неортогональної растеризації, що дозволяє зменшити $(T-q)$ операцій ділення, де q - кількість горизонтальних рядків, за якими проводиться растеризація трикутника проекції екранного пікселя в текстурний простір, а T - кількість внутрішніх точок трикутника.

3. Практичне значення отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби текстурювання для систем візуалізації тривимірних зображень.

4. Основні результати досліджень опубліковано в 23 наукових працях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Романюк О. Н., Озерчук Д. А., Станіславенко Є. Г., Котлик С. В. Адаптивне використання методів Гуро та Фонга для задач рендерингу. Інформаційні технології і автоматизація : матеріали XIV міжнар. наук.-практ. конф., 21–22 жовт. 2021 р., Одеса. Одеса : Видавництво ОНАХТ, 2021. С. 330–332.
2. Романюк О. Н., Хомюк І. В., Вінтонюк В. В., Станіславенко Є. Г. Шейдерна реалізація зафарбовування. Електронні інформаційні ресурси: створення, використання, доступ : зб. матеріалів міжнар. наук.-практ. Інтернет-конф., 9–10 листоп. 2021 р. Суми; Вінниця : НІКО; ВНТУ, 2021. С. 184–185.
3. Романюк О. Н., Станіславенко Є. Г., Романюк С. О. Аналіз моделей відбивних здатностей поверхонь. Сучасна наука: проблеми та перспективи (частина II) : матеріали VI міжнар. наук.-практ. конф., 12–13 січ. 2022 р., Київ. Київ : МЦНіД, 2022. С. 67–69.
4. Романюк О. Н., Станіславенко Є. Г., Денисюк А. В., Рейда О. М., Котлик С. В. Етапи розробки персонажів у відеоіграх. Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації : матеріали II Всеукр. наук.-техн. конф. молодих учених, аспірантів та студентів, 29–30 верес. 2022 р., Одеса. Одеса : Видавництво ОНТУ, 2022. С. 145–149.
5. Завальнюк Є. К., Станіславенко Є. Г., Вінтонюк В. В., Васянович Є. А., Романюк О. Н. Аналіз нових фізично-точних двопроменевих функцій відбивної здатності. Modern research in world science : матеріали 9-ї міжнар. наук.-практ. конф., 28–30 листоп. 2022 р., Львів. Львів : SPC “Sci-conf.com.ua”, 2022. С. 480–484.
6. Завальнюк Є. К., Станіславенко Є. Г., Вінтонюк В. В., Романюк О. Н. Аналіз особливостей DirectX 12. The 9 th International scientific and practical conference “Modern research in world science” (November 28-30, 2022) SPC “Sci-conf.com.ua”, Lviv, Ukraine. 2022. –pp. 484-486.

7. Романюк О. Н., Станіславенко Є. Г., Чехместрук Р. Ю., Романюк О. В., Коваль Л. Г. Нововведення в програмному забезпеченні «Zbrush». Стан, досягнення та перспективи інформаційних систем і технологій : матеріали XXII Всеукр. наук.-техн. конф. молодих учених, аспірантів та студентів, 21–22 квіт. 2022 р., Одеса. Одеса : Видавництво ОНТУ, 2022. С. 181–183.

8. Романюк О. Н., Станіславенко Є. Г., Мельник А. В., Романюк С. О. Використання програмного пакета Substance Painter для розробки 3D моделей. Електронні інформаційні ресурси: створення, використання, доступ : зб. матеріалів міжнар. наук.-практ. Інтернет-конф., 20–21 листоп. 2023 р., Суми; Вінниця. Суми; Вінниця, 2023. С. 227–230.

9. Романюк О. Н., Станіславенко Є. Г., Шевченко О. О. Шейдерна технологія зафарбовування. Сучасні інформаційні системи та технології : матеріали VI Всеукр. наук.-практ. Інтернет-конф. студентів, аспірантів та молодих учених, за тематикою «Сучасні комп'ютерні системи та мережі в управлінні», 2023 р., Херсон. Херсон : Книжкове видавництво ФОП Вишемирський В. С., 2023. С. 46–47.

10. Романюк О. Н., Станіславенко Є. Г., Котлик С. В., Романюк О. В., Стахов О. Я. Текстурування при формуванні тривимірних персонажів. Інформаційні технології і автоматизація – 2023 : матеріали XVI міжнар. наук.-практ. конф., 19–20 жовт. 2023 р., Одеса. Одеса : Видавництво ОНТУ, 2023. С. 448–450.

11. Романюк О. Н., Станіславенко Є. Г. Текстові редактори для веб-розробників. Молодь у світі сучасних технологій : матеріали міжнар. наук.-практ. конф. за тематикою: «Використання інформаційних та комунікаційних технологій в сучасному цифровому суспільстві», 4–5 черв. 2020 р., Херсон. Херсон, 2020. С. 196–197.

12. Романюк О. Н., Станіславенко Є. Г., Романюк О. В. Нововведення в програмному забезпеченні «Blender». Тези доповідей XII Міжнар. наук.-техн. конф. «Інформаційно-комп'ютерні технології – 2021 (ІКТ-2021)», 01–03 квіт. 2021 р., Житомир. Житомир : Житомирська політехніка, 2021. С. 15–16.

13. Романюк О. Н., Чехместрук Р. Ю., Станіславенко Є. Г., Вінтонюк В. В., Захарчук М. Взаємозв'язок між векторами нормалі до поверхні, вектором спостерігача та вектором джерела світла для задач рендерингу. Topical issues of modern science, society and education : матеріали 8-ї міжнар. наук.-практ. конф., 26–28 лют. 2022 р., Харків. Харків : SPC “Sciconf.com.ua”, 2022. С. 300–303.

14. Романюк О. Н., Станіславенко Є. Г., Романюк С. О., Чехместрук Р. Ю., Романюк О. В. Метод спрощення рендерингу Гуро. Актуальні проблеми сучасної науки та освіти (частина I) : матеріали IV міжнар. наук.-практ. конф., 20–21 січ. 2022 р., Львів. Львів : Львівський науковий форум, 2022. С. 68–70.

15. Романюк О. Н., Станіславенко Є. Г. Аналіз дистрибутивних функцій для задач високореалістичного рендерингу [Електронний ресурс] / О. Н. Романюк, О. В. Романюк, Є. Г. Станіславенко. Матеріали LI наук.-техн. конф. підрозділів ВНТУ, 31 трав. 2022 р., Вінниця. 2022. Електрон. текст. дані. Режим доступу: <https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15075>.

16. Романюк О. Н., Ліщинська Л. Б., Станіславенко Є. Г. Використання програмного забезпечення Mari в області 3D-моделювання. Матеріали Всеукраїнської науково-практ. інтернет-конф. «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», 11–20 трав. 2024 р., Вінниця. Електрон. текст. дані. 2024.

17. Романюк О. Н., Чехместрук Р. Ю., Станіславенко Є. Г. Основні положення фізично-коректного рендерингу. Матеріали LIII наук.-техн. конф. підрозділів ВНТУ, 20–22 берез. 2024 р., Вінниця. Електрон. текст. дані. 2024. URI: <https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20288>.

18. Романюк О. Н., Станіславенко Є. Г. Використання програмного забезпечення Mari в галузі 3D-моделювання. Матеріали LIII наук.-техн. конф. підрозділів ВНТУ, 20–22 берез. 2024 р., Вінниця. Електрон. текст. дані. 2024. URI: <https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20347>.

19. Романюк О. Н., Захарчук М. Д., Станіславенко Є. Г. Розробка методів антропометричних вимірювань з використанням тривимірного моделювання. Матеріали Всеукраїнської науково-практ. інтернет-конф. «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», 11–20 трав. 2024 р., Вінниця. Електрон. текст. дані. 2024. URI: <https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/20098>.

20. Романюк О. Н., Станіславенко Є. Г., Новосельцев О. О., Захарчук М. Д. Використання Parallax Mapping для формування зображень.

21. Романюк О. Н., Мельник А. В., Станіславенко Є. Г., Новосельцев О. О., Котлик С. В. Методи надання фактурності зображень поверхонь при формуванні тривимірних графічних сцен. Вінницький національний технічний університет, Україна; Одеський національний технологічний університет, Україна.

22. Романюк О. Н., Новосельцев О. О., Станіславенко С. Г., Майданюк В. П., Романюк С. О. Аналіз нових методів текстурювання. Електронні інформаційні ресурси: створення, використання, доступ та управління : матеріали міжнар. наук.-практ. Інтернет-конф., 20-21 листопада 2024 р. — С. 171–174.

23. Кардосо Дж. V-Ray 5 for 3ds Max 2020: Day & Night Interior Workflows for Parametric Designs. Volume 2 : монографія. — 2020. — 400 с.

24. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

25. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

26. Pharr M., Jakob W., Humphreys G. Physically Based Rendering: From Theory to Implementation / Morgan Kaufmann, 2016. — 500 с.

27. Watt A. 3D Computer Graphics / Addison-Wesley Professional, 2019. — 450 с.

28. Pharr M., Jakob W., Humphreys G. Physically Based Rendering: From Theory to Implementation / Morgan Kaufmann, 2016. — 500 с.

29. Brooker D. Essential CG Lighting Techniques with 3ds Max / Focal Press, 2017. – 320 c.
30. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering / A K Peters/CRC Press, 2018. – 700 c.
31. Rhyne T.-M. Applying Color Theory to Digital Media and Visualization / CRC Press, 2016. – 260 c.
32. McKinley M. The Texture Artist's Guide to Substance Painter / CRC Press, 2018. – 300 c.
33. McDermott W. The PBR Guide: A Handbook for Physically Based Rendering / Allegorithmic, 2018. – 250 c.
34. Brito A. Blender 2.8: The Beginner's Guide / Packt Publishing, 2019. – 420 c.
35. Lengyel E. Foundations of Game Engine Development, Volume 2: Rendering / Terathon Software LLC, 2019. – 280 c.
36. Demers O. Digital Texturing & Painting / New Riders, 2017. – 230 c.
37. Varcholik P. Real-Time 3D Rendering with DirectX and HLSL: A Practical Guide to Graphics Programming / Addison-Wesley Professional, 2014. – 370 c.
38. Dunn F., Parberry I. 3D Math Primer for Graphics and Game Development / A K Peters/CRC Press, 2021. – 400 c.
39. Engel W. Real-Time 3D Terrain Engines Using C++ and DirectX 9 / Charles River Media, 2023. – 450 c.

ДОДАТКИ

ДОДАТОК А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф.

О. Н. Романюк

"19" вересня 2024 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Розробка методів і програмних засобів текстурівання для задач
рендерингу»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

д.т.н., проф. О.Н. Романюк

"19" вересня 2024 р.

Виконав: студент гр.1ПІ-23м

Є. Г. Станіславенко

"19" вересня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: Розробка методів і програмних засобів текстуровання для задач рендерингу.

Галузь застосування – системи комп'ютерної графіки.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи є індивідуальне завдання на МКР і наказ №310 від «17» вересня 2024 р. ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності текстуровання шляхом зменшення обчислювальної складності, а також покращення реалістичності зображень через більш точне визначення кольорів пікселів.

Призначення роботи – розробка методів і програмних засобів текстуровання для задач рендерингу.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. Романюк О.Н. Комп'ютерна графіка. Навчальний посібник / О. Н. Романюк – Вінниця: ВДТУ, 1999. – 130 с.

2. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering / A K Peters/CRC Press, 2018. – 700 с.

3. Pharr M., Jakob W., Humphreys G. Physically Based Rendering: From Theory to Implementation / Morgan Kaufmann, 2016. – 500 с.

5. Технічні вимоги

Методи текстуровання – прискореної анізотропної фільтрації, неортогональної растеризації, parallax mapping; розмір координатного простору 1280x1024; кольоровий режим – TrueColor; тип тривимірних моделей «.max»; тип полігональної моделі – трикутна; середовище розробки IntelliJ IDEA Community Edition 2022.1; мова розробки Python; операційна система – Windows

10.вихідні дані – кінцеве зображення, зафарбоване згідно методів анізотропної фільтрації, неортогональної растеризації та parallax mapping.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз методів текстурування	20.09.2024 – 23.09.2024
2	Розробка методів текстурування, які ураховують перспективу	23.09.2024 – 13.10.2024
3	Розробка плагіну текстурування для задач рендерингу	13.10.2024 – 20.10.2024
4	Програмна реалізація запропонованих методів текстурування	20.10.2024 – 29.10.2024
5	Економічна частина	29.10.2024 – 10.11.2024
6	Оформлення матеріалів до захисту МКР	28.11.2024 – 30.11.2024

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б. Протокол перевірки МКР на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Розробка методів і програмних засобів текстування для задач рендерингу

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ІПІ-23м

Науковий керівник: д.т.н., проф.Романюк О. Н.

Turnitin	
Оригінальність	90%
Схожість	10%

Аналіз звіту подібності

- ☒ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методів і програмних засобів текстування для задач рендерингу».

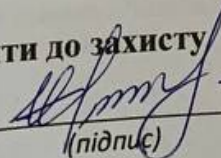
Автор



Станіславенко Євгеній Григорович

Опис прийнятого рішення: допустити до захисту

Особа, відповідальна за перевірку



Г. О. Черноволик
(ініціали, прізвище)

Експерт

(за потреби)

(підпис)

(прізвище, ініціали, посада)

ДОДАТОК В. Лістинг програми

```

#version 430

layout (local_size_x = 1) in;
layout (std430 , binding=3) buffer Vert {
    mat3x4 Position[];
};
layout (std430 , binding=4) buffer TexVert {
    mat3x2 UV[];
};
layout (std430 , binding=5) buffer S {
    int Square[];
};
layout (std430 , binding=6) buffer Norm {
    vec4 Normals[];
};
layout (std430 , binding=7) buffer BackR {
    mat4 Rotates[];
};

uniform mat4 MV;

struct Angles {
    float sinX;
    float cosX;
    float sinY;
    float cosY;
};

vec3 getPlaneNormal(vec4 v1, vec4 v2, vec4 v3){
    float v1x = v2.x-v1.x,
        v1y = v2.y-v1.y,
        v1z = v2.z-v1.z,
        v2x = v3.x-v1.x,
        v2y = v3.y-v1.y,
        v2z = v3.z-v1.z;
    return vec3(v1y*v2z-v1z*v2y, v1z*v2x-v1x*v2z, v1x*v2y-v1y*v2x);
}

Angles getAngles(vec3 normal){
    float d = sqrt(normal.y*normal.y + normal.z*normal.z);
    return Angles(normal.y/d, normal.z/d, normal.x, d);
}

vec3 mul(vec3 v, mat4 m){
    vec4 r = m * vec4(v, 0);
    if (r.w != 0.0)
        return vec3(r.x/r.w, r.y/r.w, r.z/r.w);
    return r.xyz;
}

```

```

}

void turnPoygon(uint idx){
    vec3 normal = getPlaneNormal(Position[idx][0], Position[idx][1], Position[idx][2]);
    normal = normalize(normal);
    Normals[idx] = vec4(normal, 0.0);
    Angles ang = getAngles(normal);
    Normals[idx] = vec4(ang.sinX, ang.cosX, ang.sinY, ang.cosY);
    mat4 mTX = mat4(
        vec4(1,      0,      0, 0),
        vec4(0, ang.cosX, ang.sinX, 0),
        vec4(0, -ang.sinX, ang.cosX, 0),
        vec4(0,      0,      0, 1)
    );
    mat4 mTY = mat4(
        vec4( ang.cosY,      0,      ang.sinY, 0),
        vec4(      0,      1,      0, 0),
        vec4(-ang.sinY,      0,      ang.cosY, 0),
        vec4(      0,      0,      0, 1)
    );
    mat4 mT = mTY * mTX;
    Position[idx] = mT * Position[idx];
    mTX = mat4(
        vec4(1,      0,      0, 0),
        vec4(0, ang.cosX, -ang.sinX, 0),
        vec4(0, ang.sinX, ang.cosX, 0),
        vec4(0,      0,      0, 1)
    );
    mTY = mat4(
        vec4( ang.cosY,      0,      -ang.sinY, 0),
        vec4(      0,      1,      0, 0),
        vec4( ang.sinY,      0,      ang.cosY, 0),
        vec4(      0,      0,      0, 1)
    );
    Rotates[idx] = mTX * mTY;
}

void swapVertexes(uint idx){
    vec4 buff;
    vec2 tbuf;
    if (Position[idx][1].y < Position[idx][0].y) {
        buff = Position[idx][1];
        Position[idx][1] = Position[idx][0];
        Position[idx][0] = buff;

        tbuf = UV[idx][1];
        UV[idx][1] = UV[idx][0];
        UV[idx][0] = tbuf;
    }
    if (Position[idx][2].y < Position[idx][0].y) {
        buff = Position[idx][2];
        Position[idx][2] = Position[idx][0];

```

```

    Position[idx][0] = buff;

    tbuf = UV[idx][2];
    UV[idx][2] = UV[idx][0];
    UV[idx][0] = tbuf;
}
if (Position[idx][1].y > Position[idx][2].y) {
    buff = Position[idx][2];
    Position[idx][2] = Position[idx][1];
    Position[idx][1] = buff;

    tbuf = UV[idx][2];
    UV[idx][2] = UV[idx][1];
    UV[idx][1] = tbuf;
}
}

void roundPositions(uint idx) {
    for (uint i = 0; i < 3; ++i){
        for (uint j = 0; j < 3 ; j ++){
            Position[idx][i][j] = round(Position[idx][i][j]);
        }
    }
}

void getSquare2(uint idx) {
    float dx13 = 0, dx12 = 0, dx23 = 0;
    float x1 = round(Position[idx][0].x);
    float x2 = round(Position[idx][1].x);
    float x3 = round(Position[idx][2].x);

    float y1 = round(Position[idx][0].y);
    float y2 = round(Position[idx][1].y);
    float y3 = round(Position[idx][2].y);
    float b;
    if (y2 < y1) {
        b = y1; y1 = y2; y2 = b;
        b = x1; x1 = x2; x2 = b;
    }
    if (y3 < y1) {
        b = y1; y1 = y3; y3 = b;
        b = x1; x1 = x3; x3 = b;
    }
    if (y2 > y3) {
        b = y2; y2 = y3; y3 = b;
        b = x2; x2 = x3; x3 = b;
    }
    if (y3 != y1) {
        dx13 = x3 - x1;
        dx13 /= y3 - y1;
    }
    if (y2 != y1) {

```

```

        dx12 = x2 - x1;
        dx12 /= y2 - y1;
    }
    if (y3 != y2) {
        dx23 = x3 - x2;
        dx23 /= y3 - y2;
    }

    float wx1 = x1;
    float wx2 = wx1;

    float _dx13 = dx13;

    if (dx13 > dx12) {
        float b = dx13;
        dx13 = dx12;
        dx12 = b;
    }
    int square = 0;
    for (int i = int(y1); i < y2; i++){
        square += int(round(wx2) - round(wx1));

        wx1 += dx13;
        wx2 += dx12;
    }

    if (y1 == y2){
        wx1 = x1;
        wx2 = x2;
    }

    if (_dx13 < dx23) {
        float b = _dx13;
        _dx13 = dx23;
        dx23 = b;
    }

    for (int i = int(y2); i <= y3; i++){
        square += int(round(wx2) - round(wx1));

        wx1 += _dx13;
        wx2 += dx23;
    }
    Square[idx] = square;
}

void getSquare(uint idx){
    float dx13 = 0, dx12 = 0, dx23 = 0;

    if (Position[idx][2].y != Position[idx][0].y) {
        dx13 = Position[idx][2].x - Position[idx][0].x;
        dx13 /= Position[idx][2].y - Position[idx][0].y;
    }

```

```

    }
    if (Position[idx][1].y != Position[idx][0].y) {
        dx12 = Position[idx][1].x - Position[idx][0].x;
        dx12 /= Position[idx][1].y - Position[idx][0].y;
    }
    if (Position[idx][2].y != Position[idx][1].y) {
        dx23 = Position[idx][2].x - Position[idx][1].x;
        dx23 /= Position[idx][2].y - Position[idx][1].y;
    }

    float wx1 = Position[idx][0].x;
    float wx2 = wx1;

    float _dx13 = dx13;

    if (dx13 > dx12) {
        float b = dx13;
        dx13 = dx12;
        dx12 = b;
    }

    int square = 0;
    for (int i = int(Position[idx][0].y); i < Position[idx][1].y; i++){
        square += int(round(wx2) - round(wx1));

        wx1 += dx13;
        wx2 += dx12;
    }

    if (Position[idx][0].y == Position[idx][1].y){
        wx1 = Position[idx][0].x;
        wx2 = Position[idx][1].x;
    }

    if (_dx13 < dx23) {
        float b = _dx13;
        _dx13 = dx23;
        dx23 = b;
    }

    for (int i = int(Position[idx][1].y); i <= Position[idx][2].y; i++){
        square += int(round(wx2) - round(wx1));

        wx1 += _dx13;
        wx2 += dx23;
    }
    Square[idx] = square;
}

void main() {
    uint idx = gl_GlobalInvocationID.x;

```

```

        turnPoygon(idx);
        getSquare2(idx);
    }

#version 430

in vec2 textcoord;

uniform sampler2D Tex1;

float Gaus(float x, float y, float Vx, float Ux, float Vy, float Uy, float U, float V, float A, float C){
    float d = sqrt(A);
    float L = sqrt(C);
    float angle = (Ux*Uy+Vx*Vy) / (d * L);
    angle = sqrt(1.0-angle*angle);
    float dd = min(d, L) * angle;
    if (d < L) {
        d = dd;
    } else L = dd;
    float cos_fi = abs(Vy / sqrt(C));
    float X = pow(U*cos_fi+V*sin_fi, 2.0)/(L*L) + pow(V*cos_fi-U*sin_fi, 2.0)/(d*d);
    float r = pow(1.0 - X / 3, 10) * 1000000000000000.0;
    return r;
}

vec4 ewaFilter(sampler2D tex0, vec2 p0, vec2 du, vec2 dv, float lod, int psize){

    int scale = psize >> int(lod);
    vec4 foo = texture2D(tex0,p0);
    p0 -=vec2(0.5,0.5)/scale;
    vec2 p = scale * p0;

    float ux = FILTER_WIDTH * du.s * scale;
    float vx = FILTER_WIDTH * du.t * scale;
    float uy = FILTER_WIDTH * dv.s * scale;
    float vy = FILTER_WIDTH * dv.t * scale;
    float A = vx*vx+vy*vy+1.0;
    float B = -2*(ux*vx+uy*vy);
    float C = ux*ux+uy*uy+1.0;
    float F = A*C-B*B/4.;
    float bbox_du = 2. / (-B*B+4.0*C*A) * sqrt((-B*B+4.0*C*A)*C*F);
    float bbox_dv = 2. / (-B*B+4.0*C*A) * sqrt(A*(-B*B+4.0*C*A)*F);
    int u0 = int(floor(p.s - bbox_du));
    int u1 = int(ceil (p.s + bbox_du));
    int v0 = int(floor(p.t - bbox_dv));
    int v1 = int(ceil (p.t + bbox_dv));

    vec4 num= vec4(0., 0., 0., 1.);
    float den = 0;
    float ddq = 2 * A;
    float U = u0 - p.s;
    for (int u = u0; u <= u1; ++u) {

```

```

        if (q < F)
        {
            float r2 = q / F;
            float weight = Gaus(u, v, vx, ux, vy, uy, u-p.s, v-p.t, A, C);
            num += weight* textureLod(tex0, vec2(u+0.5,v+0.5)/scale , int(lod));
            den += weight;
        }
        q += dq;
        dq += ddq;
    }
    vec4 color = num*(1./den);
    return color;
}

```

```

vec2 textureQueryLODEWA(sampler2D sampler, vec2 du, vec2 dv, int psize){

```

```

    int scale = psize;

    float ux = du.s * scale;
    float vx = du.t * scale;
    float uy = dv.s * scale;
    float vy = dv.t * scale;
    float A = vx*vx+vy*vy;
    float B = -2*(ux*vx+uy*vy);
    float C = ux*ux+uy*uy;
    float F = A*C-B*B/4.;

    A = A/F;
    B = B/F;
    C = C/F;
    float root=sqrt((A-C)*(A-C)+B*B);
    float majorRadius = sqrt(2./(A+C-root));
    float minorRadius = sqrt(2./(A+C+root));
    float majorLength = majorRadius;
    float minorLength = minorRadius;
    if (minorLength<0.01) minorLength=0.01;

    const float maxEccentricity = MAX_ECCENTRICITY;
    float e = majorLength / minorLength;
    if (e > maxEccentricity) {
        minorLength *= (e / maxEccentricity);
    }
    float lod = log2(minorLength / TEXELS_PER_PIXEL);
    lod = clamp (lod, 0.0, log2(psize));

    return vec2(lod, e);
}
vec4 texture2DEWA(sampler2D sampler, vec2 coords){
    vec2 du = dFdx(coords);
    vec2 dv = dFdy(coords);
    int psize = textureSize(sampler, 0).x;
    float lod;

```

```

        return ewaFilter(sampler, coords, du, dv, lod, psize );
    }
    void main() {
        FragColor = texture2DEWA(Tex1, uv);
    }

#version 420

in vec2 textcoord;

uniform sampler2D Tex1;
readonly layout(r8ui) uniform uimage2D Tex2;

out vec4 FragColor;

float getIntFromTexture(float u, float v){
    uvec4 color = imageLoad(Tex2, ivec2(u, v));
    uvec4 bytes = uvec4(color * 255.0);
    uint i = (bytes.a << 24) | (bytes.b << 16) | (bytes.g << 8) | (bytes.r);
//
    return i;
}

float superGaus(float x, float y, float vx, float ux, float vy, float uy, float U, float V, float A, float C){
    vec2 a = vec2(textcoord.x + ux, textcoord.y + vx);
    vec2 b = textcoord + vec2(uy, vy);
    float X = U / sqrt(C);
    float Y = V / sqrt(A);
    float u = 2000.0*(X)/2.;
    float v = 2000.0*(Y)/2.;
    if (v > 1000) return 7.0;

```



```

float m = getIntFromTexture(u, v) ;
return m;
}

vec4 ewaFilter(sampler2D tex0, vec2 p0, vec2 du, vec2 dv, float lod, int psize){

    int scale = psize >> int(lod);
    vec4 foo = texture2D(tex0,p0);

    //don't bother with elliptical filtering if the scale is very small
    if(scale<2)
        return foo;

    p0 -=vec2(0.5,0.5)/scale;
    vec2 p = scale * p0;

    float ux = FILTER_WIDTH * du.s * scale;
    float vx = FILTER_WIDTH * du.t * scale;
    float uy = FILTER_WIDTH * dv.s * scale;
    float vy = FILTER_WIDTH * dv.t * scale;

    // compute ellipse coefficients
    //  $A*x*x + B*x*y + C*y*y = F$ .
    float A = vx*vx+vy*vy+1.0;
    float B = -2*(ux*vx+uy*vy);
    float C = ux*ux+uy*uy+1.0;
    float F = A*C-B*B/4.;

    // Compute the ellipse's (u,v) bounding box in texture space
    float bbox_du = 2. / (-B*B+4.0*C*A) * sqrt((-B*B+4.0*C*A)*C*F);
    float bbox_dv = 2. / (-B*B+4.0*C*A) * sqrt(A*(-B*B+4.0*C*A)*F);

```

```

#if 1
    //Clamp the ellipse so that the bbox includes at most TEXEL_LIMIT texels.
    //This is necessary in order to bound the run-time, since the ellipse can be arbitrarily large
    //Note that here we are actually clamping the bbox directly instead of the ellipse.
    //Non real-time GPU renderers can skip this step.
    if(bbox_du*bbox_dv>TEXEL_LIMIT){
        float ll = sqrt(bbox_du*bbox_dv / TEXEL_LIMIT);
        bbox_du/=ll;
        bbox_dv/ll;
    }
#endif

    //the ellipse bbox
    int u0 = int(floor(p.s - bbox_du));
    int u1 = int(ceil (p.s + bbox_du));
    int v0 = int(floor(p.t - bbox_dv));
    int v1 = int(ceil (p.t + bbox_dv));

    // Heckbert MS thesis, p. 59; scan over the bounding box of the ellipse
    // and incrementally update the value of  $Ax^2+Bxy+Cy^2$ ; when this
    // value, q, is less than F, we're inside the ellipse so we filter
    // away..
    vec4 num= vec4(0., 0., 0., 1.);
    float den = 0;
    float ddq = 2 * A;
    float U = u0 - p.s;

    #if (FILTERING_MODE!=4)

        for (int v = v0; v <= v1; ++v) {
            float V = v - p.t;
            float dq = A*(2*U+1) + B*V;
            float q = (C*V + B*U)*V + A*U*U;

```

```

#if (FILTERING_MODE==2)
    for (int u = u0; u <= u1; ++u) {
        if (q < F)
        {
            float r2 = q / F;

float weight = superGaus(u, v, vx, ux, vy, uy, u-p.s, v-p.t, A, C);

            num += weight* textureLod(tex0, vec2(u+0.5,v+0.5)/scale , int(lod));
            den += weight;
        }
        q += dq;
        dq += ddq;
    }
#else
    for (int u = u0; u <= u1; u+=2) {
        float w1 = FILTER_FUNC(q / F);
        w1 = (q < F)? w1: 0;
        q += dq;
        dq += ddq;
        float w2 = FILTER_FUNC(q / F);
        w2 = (q < F)? w2: 0;
        float offest= w2/(w1+w2);
        float weight = (w1+w2);

        if(weight>0.0)
        {
            num += weight * textureLod(tex0, vec2(u+0.5+offest, v+0.5)/scale ,
int(lod));

            den += weight;
        }

        q += dq;
        dq += ddq;
    }
#endif

```

```

    }

#else

    for (int v = v0; v <= v1; v+=2) {
        float V = v - p.t;
        float dq = A*(2*U+1) + B*V;
        float q = (C*V + B*U)*V + A*U*U;

        float V2 = v+1 - p.t;
        float dq2 = A*(2*U+1) + B*V2;
        float q2 = (C*V2 + B*U)*V2 + A*U*U;

        for (int u = u0; u <= u1; u+=2) {
            float w1 = FILTER_FUNC(q / F);
            w1 = (q < F)? w1: 0;
            q += dq;
            dq += ddq;
            float w2 = FILTER_FUNC(q / F);
            w2 = (q < F)? w2: 0;

            float w3 = FILTER_FUNC(q2 / F);
            //w3 = (q2 < F)? w3: 0;
            q2 += dq2;
            dq2 += ddq;
            float w4 = FILTER_FUNC(q2 / F);
            //w4 = (q2 < F)? w4: 0;

            q += dq;
            dq += ddq;
            q2 += dq2;
            dq2 += ddq;

            float offest_v=(w3+w4)/(w1+w2+w3+w4);

```

```

        float offest_u;// = (w4+w2)/(w1+w3);
        offest_u= (w4)/(w4+w3);
        float weight =(w1+w2+w3+w4);
        if(weight>0.1)
        {
            num += weight * textureLod(tex0, vec2(u+ offest_u+0.5,
v+offest_v+0.5)/scale , int(lod));
            den += weight;
        }
    }

}

#endif

    vec4 color = num*(1./den);
    return color;
}

void main() {
    FragColor = ewaFilter(Tex1, uv);
}

#include <iostream>
#include <cmath>
#include <png++/png.hpp>

float gaus(double x, double y){
    float q = 0.4;
    auto v = sqrt(x*x+y*y);
    return exp2(-2.0*v);
    return (1 / (2 * 3.141592653589793238462643 * q * q)) * exp(-(v) / (2 * q * q))-0.212934864;
}

std::vector<unsigned char> intToBytes(int paramInt)
{
    std::vector<unsigned char> arrayOfByte(4);

```

```

for (int i = 0; i < 4; i++)
    arrayOfByte[i] = (paramInt >> (i * 8));
return arrayOfByte;
}

int main() {
    int mode = 1;
    png::image< png::basic_rgba_pixel<png::byte> > image(512, 512);
    for (int y = 0; y < image.get_height(); ++y)
    {
        for (int x = 0; x < image.get_width(); ++x)
        {
            int realx, realy;
            float val;
            if (mode == 1) {
                realx = x - image.get_height() / 2;
                realy = y - image.get_height() / 2;
                val = gaus(float(realx)/image.get_width(), float(realy)/image.get_height());
            } else if (mode == 2) {
                realx = x;
                realy = image.get_height() - y;
                val = gaus(float(realx)/image.get_width()/2, float(realy)/image.get_height()/2);
            } else {
                realy = 0;
                realx = x;
                val = gaus(float(realx)/image.get_width()/2, 0);
            }
            auto i_val = static_cast<uint>(roundf(val * 1000000000));
            auto c_val = intToBytes(i_val);
            image[y][x] = png::basic_rgba_pixel<png::byte>(c_val[0], c_val[1], c_val[2], c_val[3]);
            if (y == 128)
                std::cout << y << " - " << x << " : " << val << " : " << realx << " - " << realy << std::endl;
        }
    }
}

```

```

}
image.write("rgb.png");
return 0;
}

#version 330
layout(location = 0) in vec3 pos_i;
layout(location = 1) in vec3 normal_i;
layout(location = 2) in vec2 tex_coord_i;
layout(location = 3) in vec4 tangent_i;
uniform mat4 M;
uniform mat4 V;
uniform mat4 P;
uniform mat3 N;
uniform vec3 light_vector;
uniform vec3 camera_vector;
out vec2 texcoords;
out vec3 tangent_vec_to_light;
out vec3 tangent_vec_to_camera;

void main(void)
{
    vec4 pos = M * vec4(pos_i, 1);
    vec3 normal = normalize(N * normal_i);
    vec3 tangent = normalize(N * tangent_i.xyz);

    vec3 direct_to_light = normalize(light_vector - pos.xyz);
    vec3 direct_to_camera = normalize(camera_vector - pos.xyz);

    vec3 bitangent = cross(normal, tangent) * tangent_i.w;

    tangent_vec_to_light = vec3(
        dot(direct_to_light, tangent),
        dot(direct_to_light, bitangent),
        dot(direct_to_light, normal)
    );

```

```

    );
    tangent_vec_to_camera= vec3(
        dot(direct_to_camera, tangent),
        dot(direct_to_camera, bitangent),
        dot(direct_to_camera, normal)
    );
    texcoords    = tex_coord_i;
    gl_Position  = P * V * pos;}

#version 330

in vec2 texcoords;
in vec3 tangent_vec_to_light;
in vec3 tangent_vec_to_camera;
layout(location = 0) uniform sampler2D t_diffuse;
layout(location = 1) uniform sampler2D t_height;
layout(location = 2) uniform sampler2D t_normal;
out vec4 result;

uniform float PScale; // ~0.1

vec4 lighting(in vec2 T, in vec3 L, in vec3 V, float K_shadow)
{
    vec3 N = normalize(texture(t_normal, T).xyz * 2 - 1);
    vec3 D = texture(t_diffuse, T).rgb;
    float iamb = 0.2;
    float idiff = clamp(dot(N, L), 0, 1);
    float ispec = 0;
    if(dot(N, L) > 0.2)
    {
        vec3 R = reflect(-L, N);
        ispec = pow(dot(R, V), 32) / 1.5;
    }
    vec4 resColor;
    resColor.rgb = D * (ambientLighting + (idiff + ispec) * pow(K_shadow, 4));
    resColor.a = 1;

    return resColor;
}

```



```

void main(void)
{
    vec3 V = normalize(tangent_vec_to_camera);
    vec3 L = normalize(tangent_vec_to_light);
    float h;
    vec2 T = parallaxMapping(V, texcoords, h);
    float K_shadow = pSoftShadowMultiplier(L, T, h - 0.05);
    result = lighting(T, L, V, K_shadow);
}

```

```

import bpy
import bpy_extras
from mathutils import Vector, Matrix
from bpy_extras import view3d_utils
from mathutils.geometry import intersect_line_plane
from bpy_extras.view3d_utils import (location_3d_to_region_2d,
    region_2d_to_vector_3d, region_2d_to_location_3d)
from math import radians
import bmesh
import math
from mathutils.bvhtree import BVHTree
from mathutils.geometry import distance_point_to_plane
from mathutils.geometry import interpolate_bezier
from . import maths

import numpy

def is_front_facing(context, obj, polygon):
    """
    When deciding if a polygon is facing the camera, you need
    only calculate the dot product of the normal vector of
    that polygon, with a vector from the camera to one of the
    polygon's vertices.

```

- If the dot product is less than zero, the polygon is facing the camera.
- If the value is greater than zero, it is facing away from the camera.

```
"""
```

```
region = context.region
```

```
rv3d = context.space_data.region_3d
```

```
# neat eye location code with the help of paleajed
```

```
eye = Vector(rv3d.view_matrix[2][:3])
```

```
eye.length = rv3d.view_distance
```

```
eye_location = rv3d.view_location + eye
```

```
pnormal = obj.matrix_world.to_3x3() @ polygon.normal
```

```
world_coordinate = obj.matrix_world @ obj.data.vertices[0].co
```

```
result_vector = eye_location - world_coordinate
```

```
dot_value = pnormal.dot(result_vector)
```

```
return not (dot_value < 0.0)
```

```
def view_direction(view_dir):
```

```
view_dir_abs = list(map(abs, view_dir))
```

```
i = view_dir_abs.index(max(view_dir_abs))
```

```
results = [['RIGHT', 'LEFT'], ['BACK', 'FRONT'], ['TOP', 'BOTTOM']]
```

```
res = (results[i][view_dir[i] > 0])
```

```
if i == 0:
```

```
    y = (results[1][view_dir[1] > 0])
```

```
    z = (results[2][view_dir[2] > 0])
```

```

if i == 1:
    y = (results[0][view_dir[0] > 0])
    z = (results[2][view_dir[2] > 0])

if i == 2:
    y = (results[0][view_dir[0] > 0])
    z = (results[1][view_dir[1] > 0])

a = res

if y == 'TOP' or y == 'BOTTOM':
    b = z
    c = y
else:
    b = y
    c = z

if res == 'TOP' or res == 'BOTTOM':
    a = c
    c = res

viewlis = [[a, b, c]]

return viewlis

def ortho_view_direction(view_dir):
    view_dir_abs = list(map(abs, view_dir))
    i = view_dir_abs.index(max(view_dir_abs))
    results = [['RIGHT', 'LEFT'], ['BACK', 'FRONT'], ['TOP', 'BOTTOM']]
    res = (results[i][view_dir[i] > 0])

```

```

if res == 'FRONT':
    view = Vector((0,0,1))

elif res == 'BACK':
    view = Vector((0,0,1))

elif res == 'RIGHT':
    view = Vector((0,0,1))

elif res == 'LEFT':
    view = Vector((0,0,1))

elif res == 'BOTTOM':
    view = Vector((1,0,0))

elif res == 'TOP':
    view = Vector((1,0,0))

return view

def ortho_view_matrix(view_dir):
    view_dir_abs = list(map(abs, view_dir))
    i = view_dir_abs.index(max(view_dir_abs))
    results = [['RIGHT', 'LEFT'], ['BACK', 'FRONT'], ['TOP', 'BOTTOM']]
    res = (results[i][view_dir[i] > 0])

    mat = Matrix()

    if res == 'FRONT':
        rot_mat = Matrix.Rotation(radians(90), 4, Vector((1,0,0)))

```

```
elif res == 'BACK':
```

```
    rot_mat = Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
```

```
elif res == 'RIGHT':
```

```
    rot_mat = Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
```

```
elif res == 'LEFT':
```

```
    rot_mat = Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
```

```
elif res == 'BOTTOM':
```

```
    rot_mat = Matrix.Rotation(radians(180), 4, Vector((1,0,0)))
```

```
elif res == 'TOP':
```

```
    rot_mat = Matrix.Rotation(radians(0), 4, Vector((1,0,0)))
```

```
mat @= rot_mat
```

```
return mat
```

```
def ortho_view_vectors(view_dir, axis):
```

```
    view_dir_abs = list(map(abs, view_dir))
```

```
    i = view_dir_abs.index(max(view_dir_abs))
```

```
    results = [['RIGHT', 'LEFT'], ['BACK', 'FRONT'], ['TOP', 'BOTTOM']]
```

```
    res = (results[i][view_dir[i] > 0])
```

```
    if i == 0:
```

```
        y = (results[1][view_dir[1] > 0])
```

```
        z = (results[2][view_dir[2] > 0])
```

```
    if i == 1:
```

```

y = (results[0][view_dir[0] > 0])
z = (results[2][view_dir[2] > 0])

if i == 2:
    y = (results[0][view_dir[0] > 0])
    z = (results[1][view_dir[1] > 0])

a = res

if y == 'TOP' or y == 'BOTTOM':
    b = z
    c = y
else:
    b = y
    c = z

if res == 'TOP' or res == 'BOTTOM':
    a = c
    c = res

viewlis = [[a, b, c]]

mat = Matrix()

if viewlis == [['FRONT', 'RIGHT', 'TOP']]:

    x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
    y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
    z = mat @ Matrix.Rotation(radians(0), 4, Vector((1,0,0)))

elif viewlis == [['FRONT', 'LEFT', 'TOP']]:

```

```

x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(0), 4, Vector((1,0,0)))

```

```

elif viewlis == [['FRONT', 'RIGHT', 'BOTTOM']]:

```

```

x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```

elif viewlis == [['FRONT', 'LEFT', 'BOTTOM']]:

```

```

x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```

elif viewlis == [['BACK', 'RIGHT', 'BOTTOM']]:

```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```

elif viewlis == [['BACK', 'LEFT', 'BOTTOM']]:

```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```

elif viewlis == [['BACK', 'RIGHT', 'TOP']]:

```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(0), 4, Vector((1,0,0)))

```

```
elif viewlis == [['BACK', 'LEFT', 'TOP']]:
```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(0), 4, Vector((1,0,0)))

```

```
elif viewlis == [['RIGHT', 'FRONT', 'TOP']]:
```

```

x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```
elif viewlis == [['RIGHT', 'BACK', 'TOP']]:
```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(0), 4, Vector((1,0,0)))

```

```
elif viewlis == [['RIGHT', 'FRONT', 'BOTTOM']]:
```

```

x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```
elif viewlis == [['RIGHT', 'BACK', 'BOTTOM']]:
```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))

```



```

y = mat @ Matrix.Rotation(radians(90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```

elif viewlis == [['LEFT', 'FRONT', 'TOP']]:

```

```

x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(0), 4, Vector((1,0,0)))

```

```

elif viewlis == [['LEFT', 'BACK', 'TOP']]:

```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(0), 4, Vector((1,0,0)))

```

```

elif viewlis == [['LEFT', 'FRONT', 'BOTTOM']]:

```

```

x = mat @ Matrix.Rotation(radians(90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```

elif viewlis == [['LEFT', 'BACK', 'BOTTOM']]:

```

```

x = mat @ Matrix.Rotation(radians(-90), 4, Vector((1,0,0)))
y = mat @ Matrix.Rotation(radians(-90), 4, Vector((0,1,0)))
z = mat @ Matrix.Rotation(radians(180), 4, Vector((1,0,0)))

```

```

return x, y, z

```

```

def get_perp_vec(view_dir):

```

```

    view_dir_abs = list(map(abs, view_dir))

```

```

i = view_dir_abs.index(max(view_dir_abs))
results = [['RIGHT', 'LEFT'], ['BACK', 'FRONT'], ['TOP', 'BOTTOM']]
res = (results[i][view_dir[i] > 0])

if res == 'FRONT':
    x = Vector((1,0,0))

elif res == 'BACK':
    x = Vector((-1,0,0))

elif res == 'RIGHT':
    x = Vector((0,1,0))

elif res == 'LEFT':
    x = Vector((0,-1,0))

elif res == 'BOTTOM':
    x = Vector((0,0,-1))

elif res == 'TOP':
    x = Vector((0,0,1))

return x

def avg_3d(vertices):
    x = Vector([(sum(col) / float(len(col))) for col in zip(*vertices)])

    return x

def avg_3d_global(vertices, mat):

```

```
x = Vector([(sum(col) / float(len(col)) for col in zip(*vertices))])
```

```
return x
```

```
def coord_on_plane(reg, rv3d, coord, location, normal):
```

```
    #take in 2d coord, give me 3D point on defined plane
```

```
    view_vector = view3d_utils.region_2d_to_vector_3d(reg, rv3d, coord)
```

```
    ray_origin = view3d_utils.region_2d_to_origin_3d(reg, rv3d, coord)
```

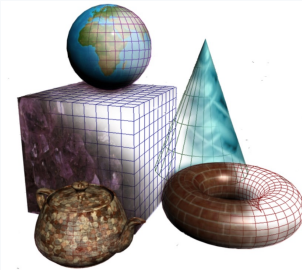
```
    vec = intersect_line_plane(ray_origin, ray_origin + view_vector, location, Vector((normal)))
```

```
    return vec
```

ДОДАТОК Г. Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА **РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ТЕКСТУРУВАННЯ ДЛЯ** **ЗАДАЧ РЕНДЕРИНГУ**

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення



Виконав:
ст. групи 1ПІ-23м
Станіславенко Є.Г.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

Розробка методів і програмних засобів
текстурування для задач рендерингу

Науковий керівник:
д.т.н., професор. каф ПЗ
Романюк О.Н.

Рисунок Г.1 – Титульний слайд

Галузі застосування та перспективи розвитку графічних засобів

Розробка комп'ютерних ігор  У 2024 році геймдев становив близько 40% ринку 3D-графіки, а його обсяг перевищив \$100 мільярдів. Щорічно випускається понад 10 000 ігор із 3D-графікою.	Кіноіндустрія  Понад 90% фільмів-блокбастерів 2023 року (включно з Marvel, DC і Disney) включали 3D-графіку. Очікується, що ринок досягне \$50 мільярдів до 2025 року, зростаючи в середньому на 10% щорічно.	Архітектура та дизайн інтер'єру  Понад 80% архітекторів використовують 3D-візуалізацію у своїй роботі (AutoCAD, Blender, Revit). Сегмент 3D-візуалізації в архітектурі збільшився на 15% за останні 3 роки.
Реклама та маркетинг  Понад 40% сучасної реклами візуалізовано у 3D-форматі. Використання VR та AR-реклами зростає на 20% щороку, особливо у промоції товарів класу "люкс".	Наукові дослідження  3D-моделі органів використовуються в 60% складних хірургічних процедур. У 2024 році понад 60% наукових проектів включали візуалізацію даних у 3D.	Освіта  3D-контент використовується в понад 70% віртуальних навчальних програм. До 2025 року кількість VR-освітніх програм збільшиться на 25%.

Рисунок Г.2 – Галузі застосування та перспективи розвитку графічних засобів



Рисунок Г.3 – Накладання текстури на тривимірний об'єкт

Мета роботи та задачі дослідження

Мета роботи: підвищення продуктивності текстурування шляхом зменшення обчислювальної складності, а також покращення реалістичності зображень через більш точне визначення кольорів пікселів.

Предмет дослідження: методи та засоби текстурування тривимірних графічних сцен.

Об'єкт дослідження: процес текстурування тривимірних об'єктів у системах комп'ютерної графіки.

Задачі дослідження:

- провести аналіз існуючих методів і засобів накладання текстур для визначення напрямків підвищення їх продуктивності та реалістичності;
- запропонувати нові методи підвищення продуктивності текстурування;
- розробити програмні засоби текстурування на основі запропонованих методів;
- провести експериментальні дослідження розроблених засобів текстурування.

Рисунок Г.4 – Мета, об'єкт і предмет роботи



Рисунок Г.5 – Класифікація методів текстурування

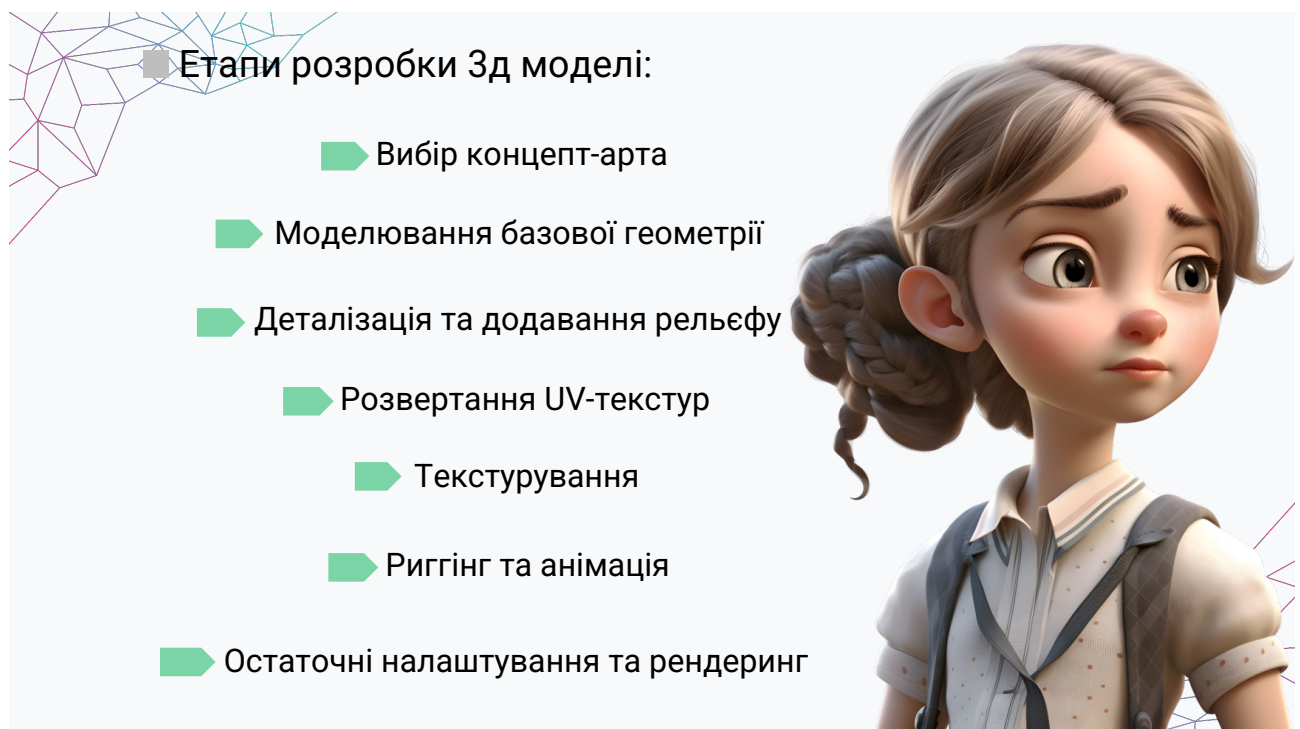


Рисунок Г.6 – Етапи розробки 3д моделі

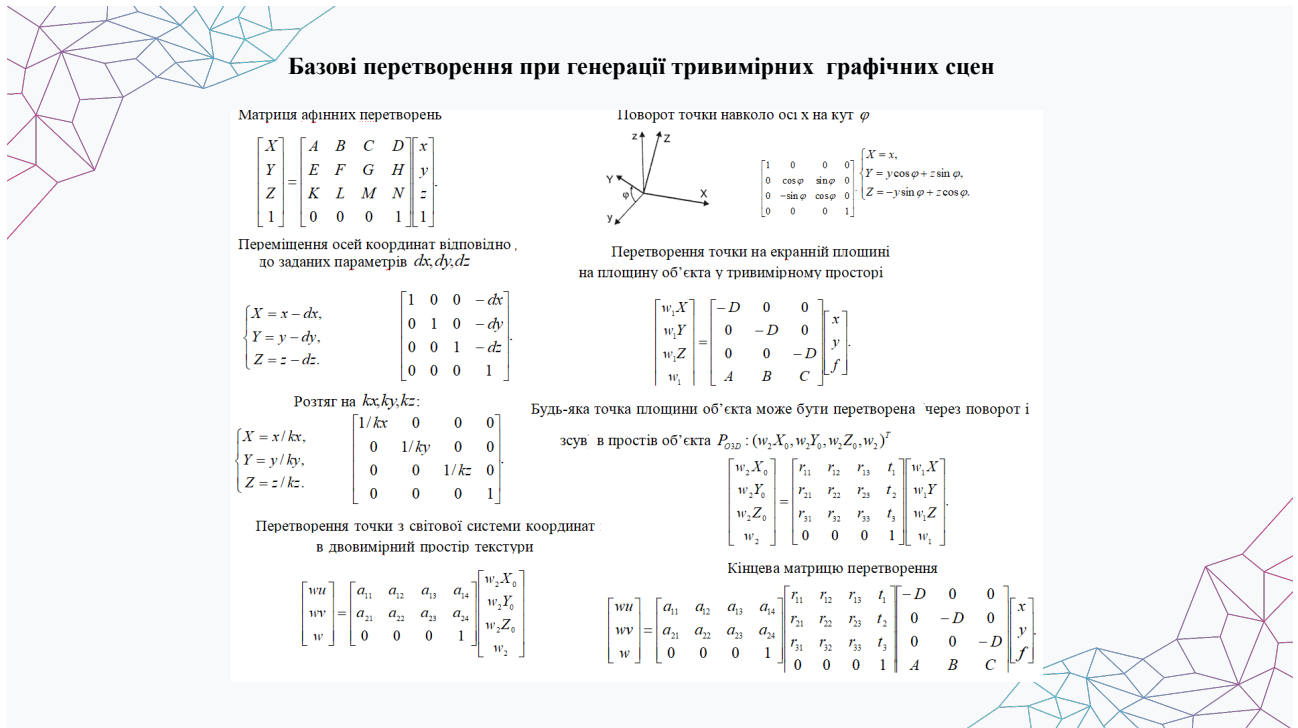


Рисунок Г.7 – Базові перетворення при генерації тривимірних графічних сцен



Рисунок Г.8 – Метод у визначенні координати текселя за картою вагових коефіцієнтів

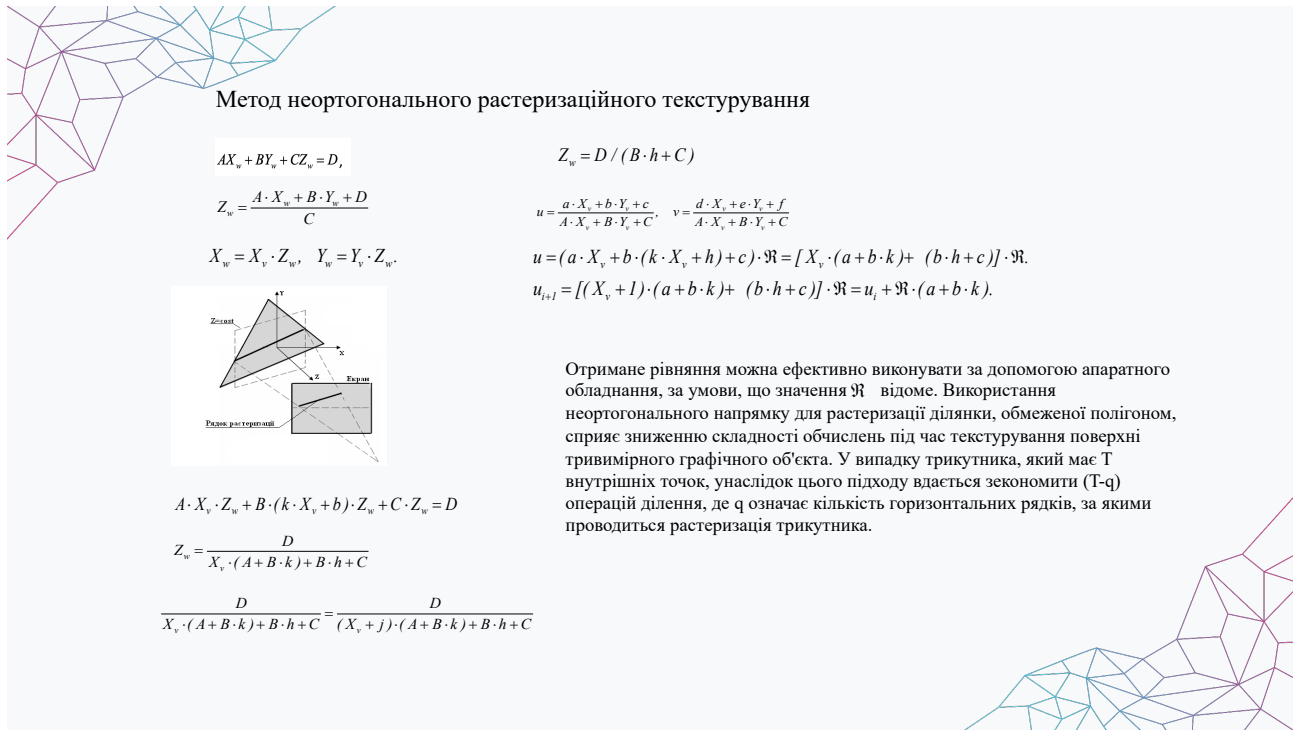


Рисунок Г.9 – Метод неортогонального растеризаційного текстуровання

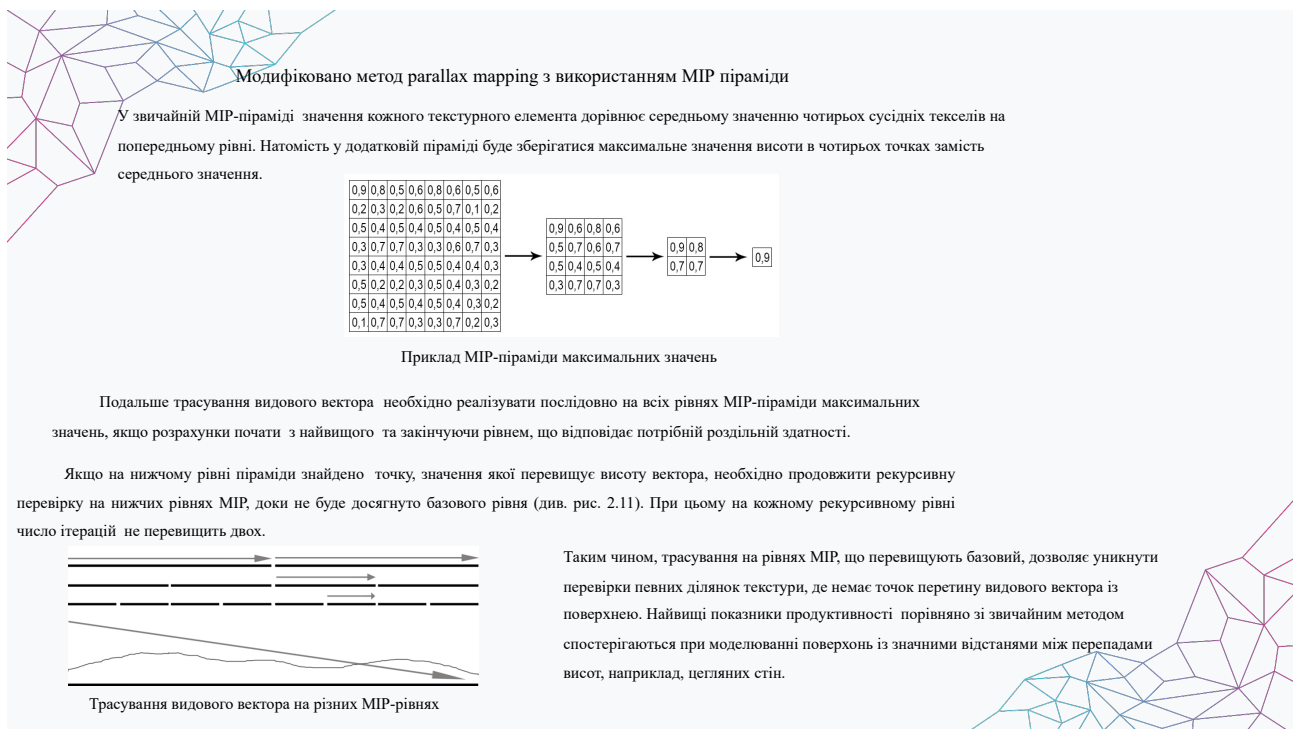


Рисунок Г.10 – Модифіковано метод parallax mapping



Рисунок Г.11 – Розроблені методи



Рисунок Г.12 – Приклад текстурних карт

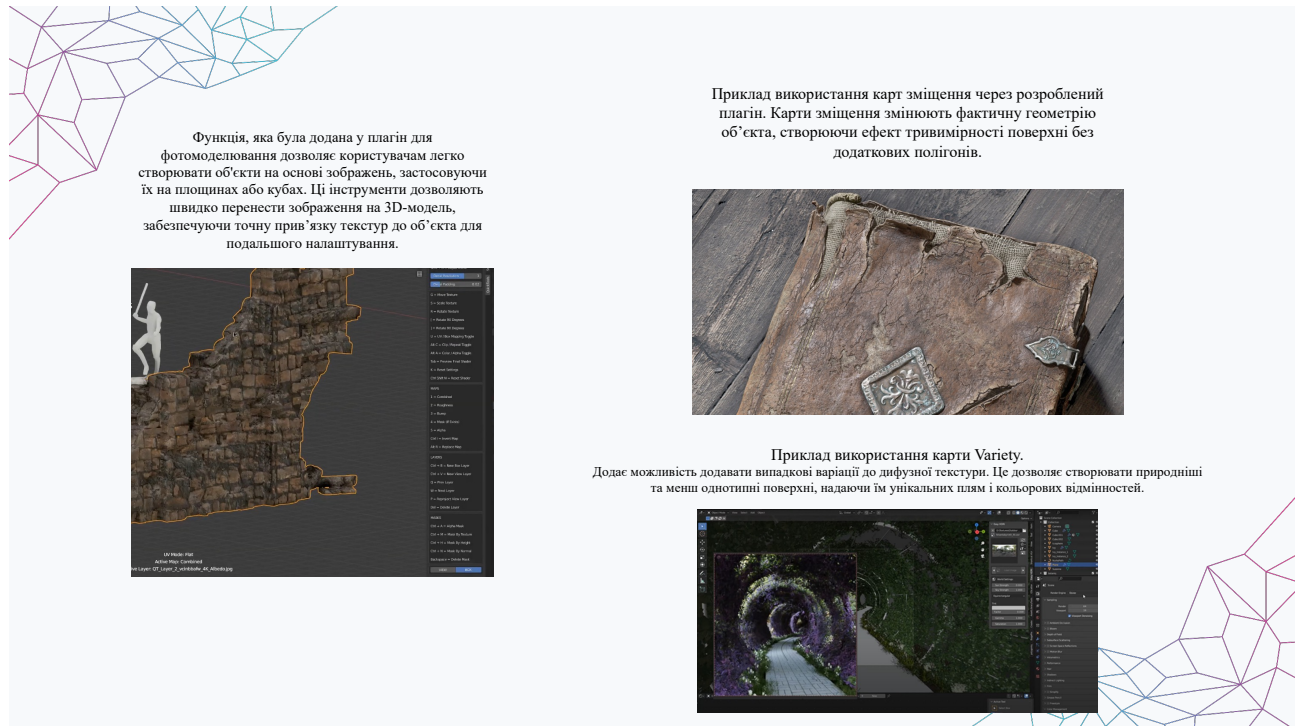


Рисунок Г.13 – Приклад використання карт

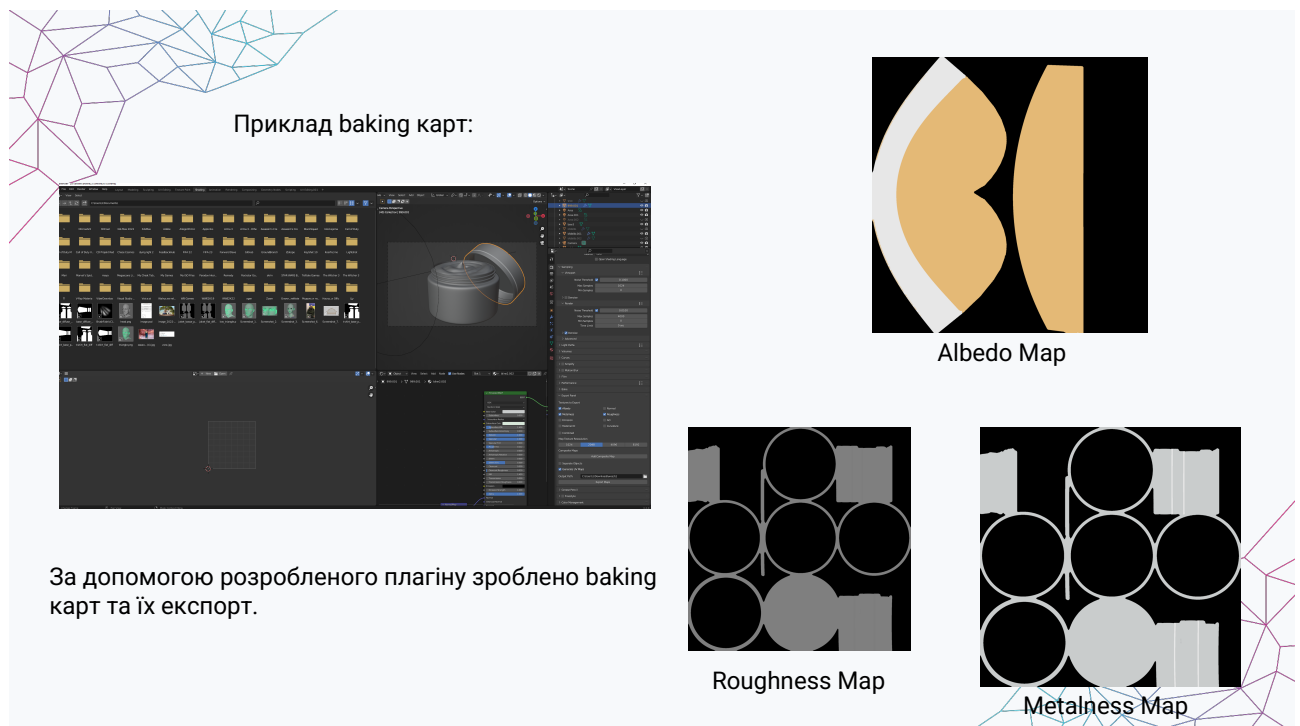


Рисунок Г.14 – Приклад використання baking карт в плагіні

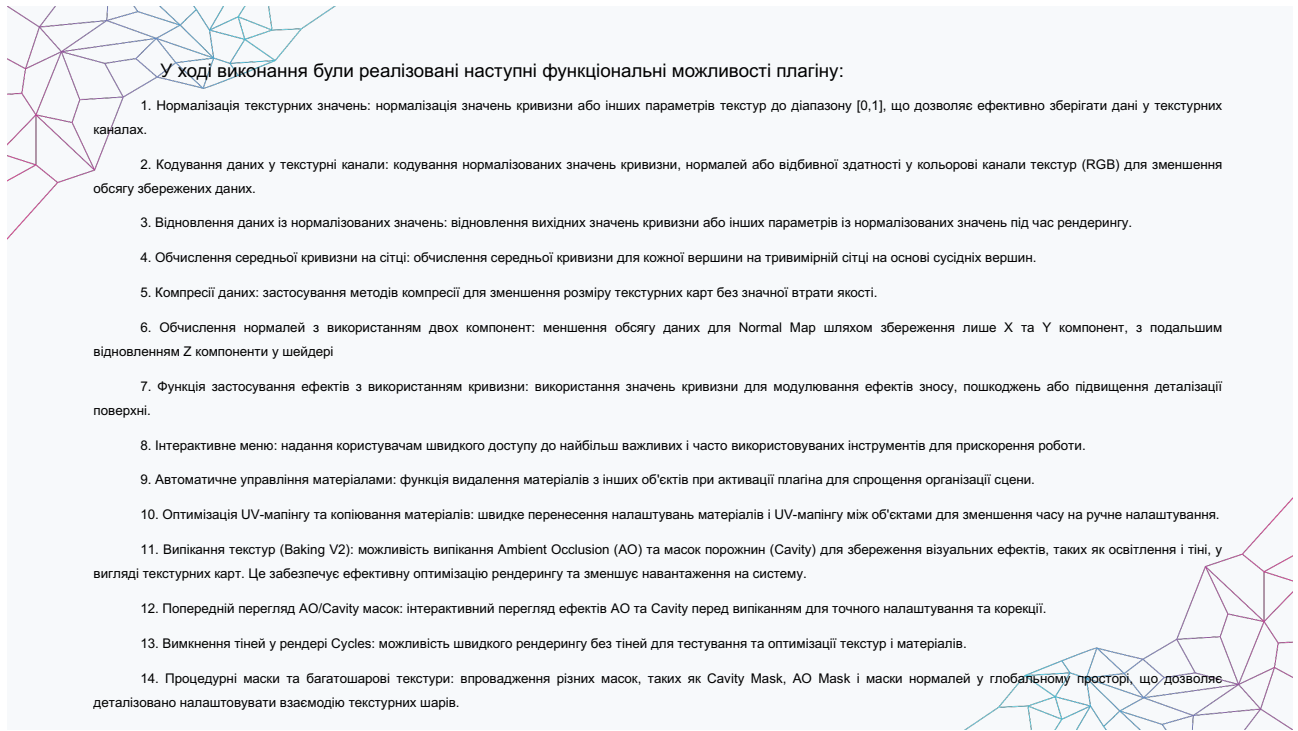


Рисунок Г.15 – Реалізовані наступні функціональні можливості плагіну

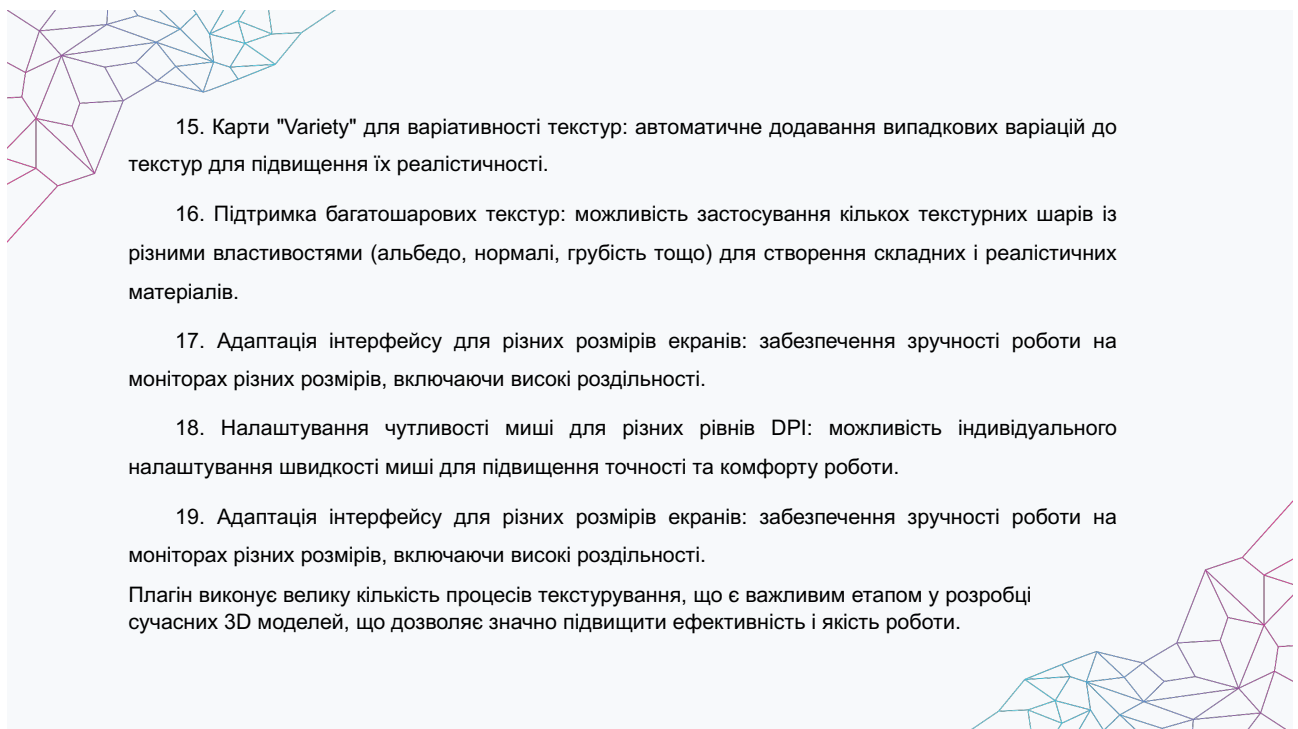


Рисунок Г.16 – Можливості плагіну

Результат використання текстурних карт через плагін:



Рисунок Г.17 – Результат використання текстурних карт через плагін

Граф-схема алгоритму для покращення текстур

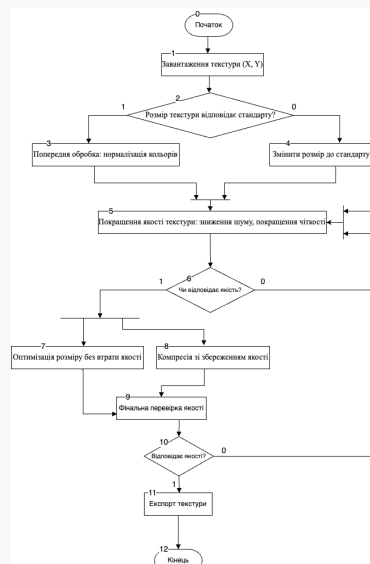


Рисунок Г.18 – Граф-схема алгоритму для покращення текстур

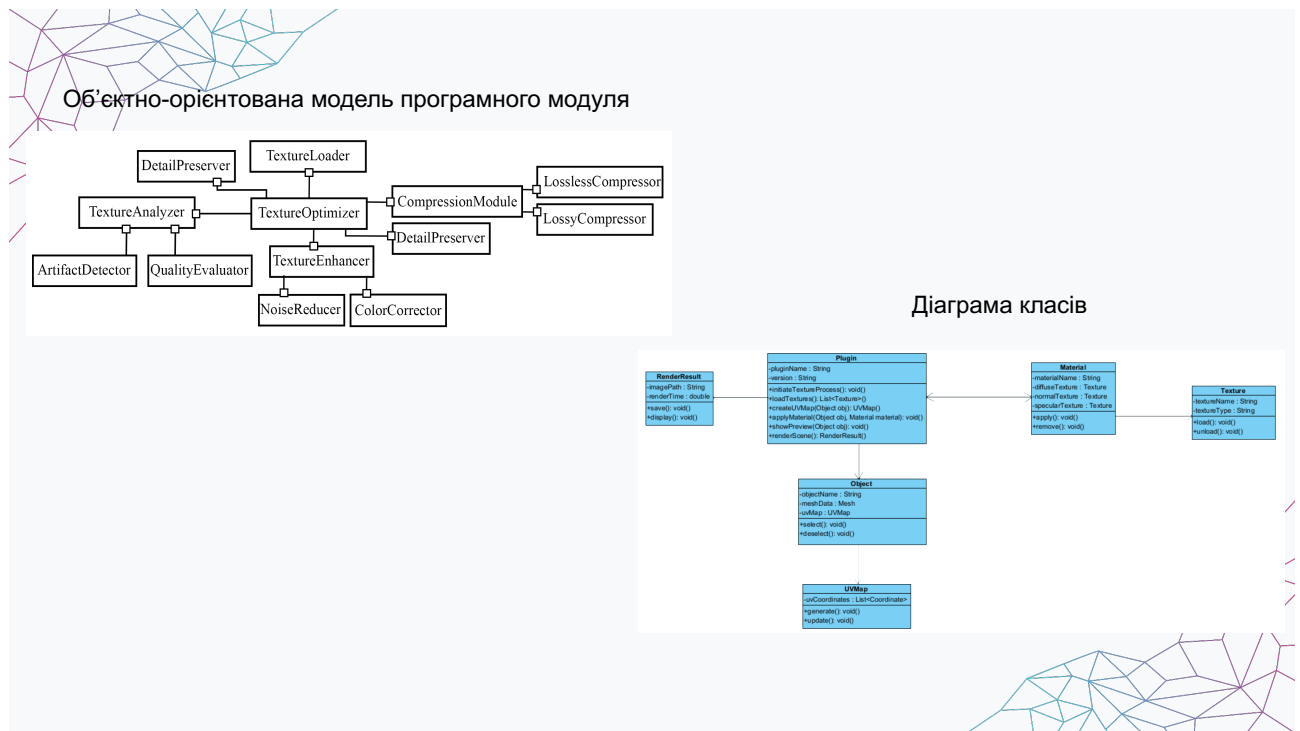


Рисунок Г.19 – Об'єктно-орієнтована модель програмного модуля

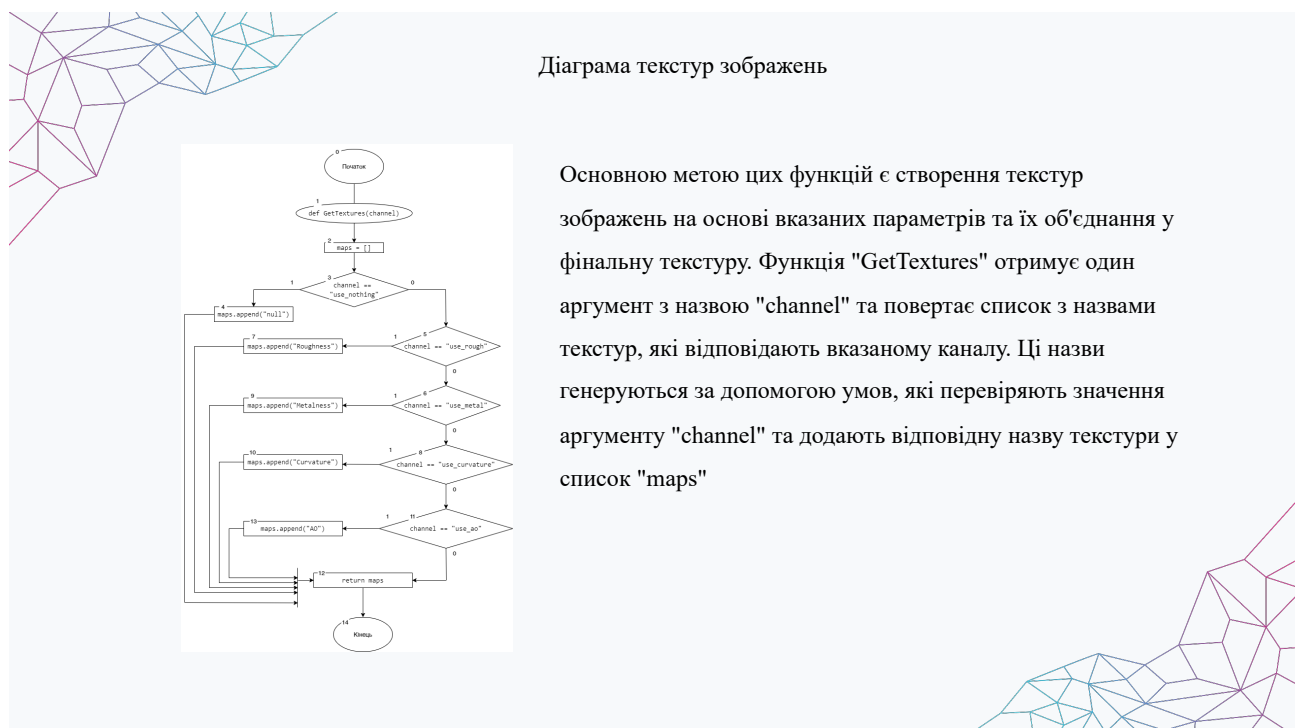
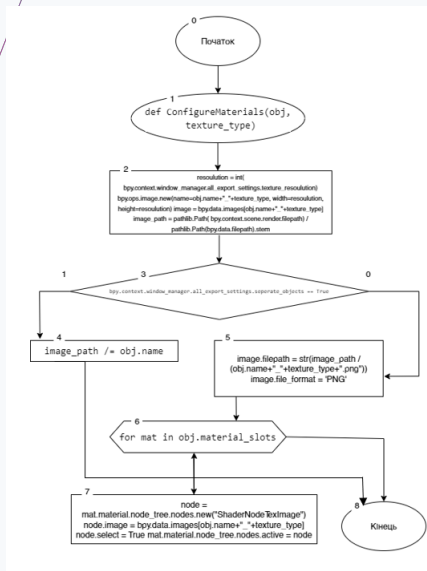


Рисунок Г.20 – Діаграма текстур зображень

Діаграма з формування текстур зображень



Діаграма з формування текстур зображень. Створюється текстура для об'єкта з певним типом текстури (що передається у параметрі texture_type) з заданою роздільною здатністю (що береться з параметру texture_resolution у вікні налаштувань експорту). Далі ця текстура присвоюється як вхідний параметр для матеріалів цього об'єкта. Приймає два параметри: об'єкт і тип текстури. Код спочатку створює новий зображення з ім'ям, що складається з імені об'єкта та типу текстури. Розмір зображення залежить від налаштувань експорту, вказаних у вікні менеджера вікон. Потім він встановлює шлях для зображення, використовуючи поточну директорію експорту та ім'я файлу. Якщо встановлено флажок separate_objects, то шлях також включає ім'я об'єкта. Для кожного матеріалу в об'єкті код створює новий вузол зображення та призначає йому раніше створене зображення як вхідний файл.

Рисунок Г.21 – Діаграма з формування текстур зображень

Порівняння інтегрованих середовищ розробки

Критерій	IntelliJ IDEA Community	JetBrains Rider	MonoDevelop	Sublime Text
Універсальна	1	1	0	1
Кросплатформена	1	1	1	1
Швидкодійна	1	0,5	0,5	0,5
Не висока вимогливість до системи	1	0,5	0,5	0,5
Магазин користувацьких розширень	1	0,5	0	0,5
Автодоповнення коду	1	1	1	1
Підсумковий результат	6	4,5	3	4,5

Рисунок Г.22 – Порівняння інтегрованих середовищ розробки

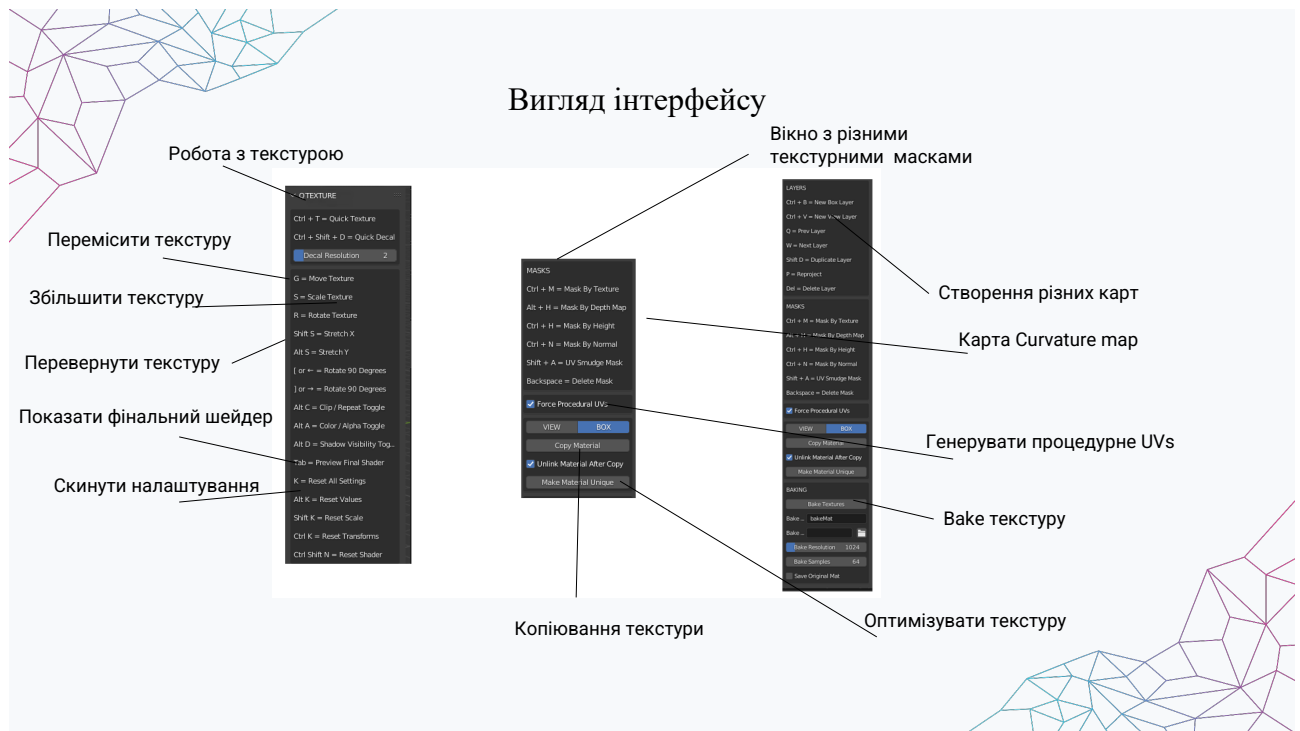


Рисунок Г.23 – Вигляд інтерфейсу

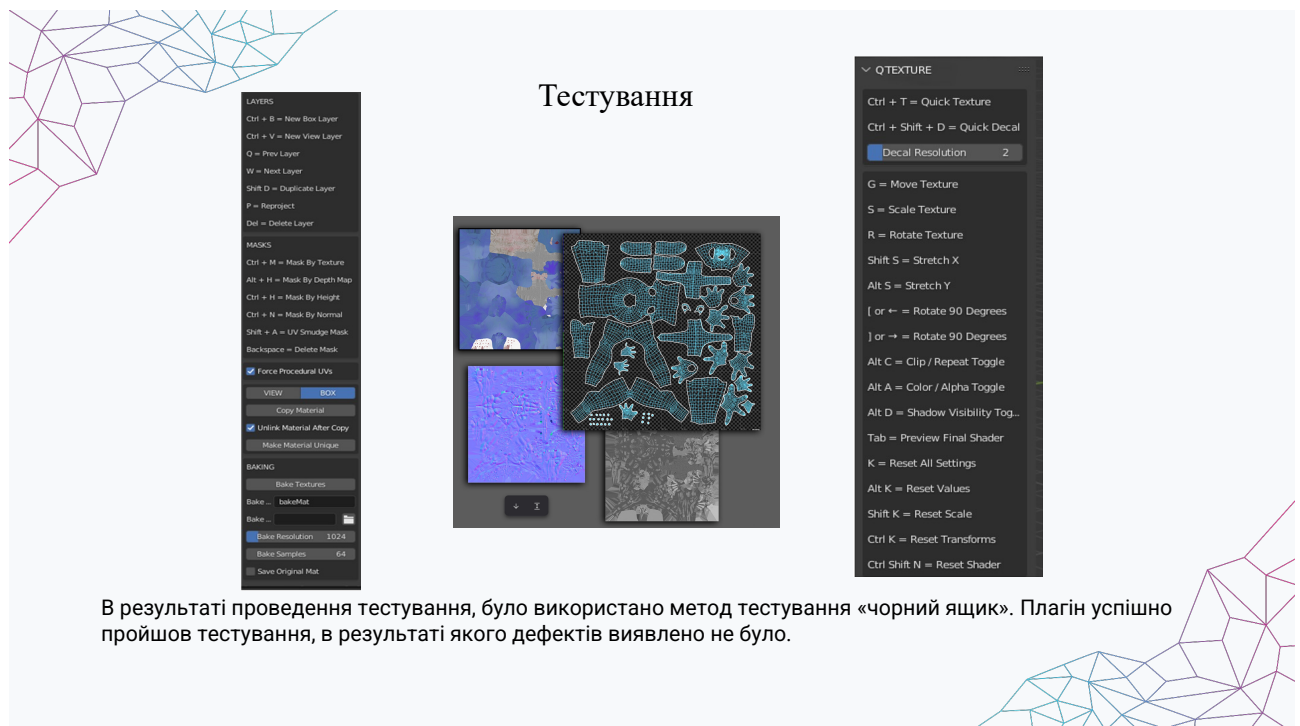


Рисунок Г.24 – Тестування

Наукова новизна отриманих результатів:

1. Отримано формули для базових перетворень при генерації тривимірних тривимірних об'єктів, що дозволяє їх використовувати для розробки нових методів і засобів.
2. Подальшого розвитку отримав метод текстуровання, відмінність якого від відомих полягає у визначенні координати текселя за картою вагових коефіцієнтів, що дозволяє зчитувати вагові коефіцієнти безпосередньо з текстурної карти, не розраховуючи їх. Встановлено, що застосування вагової карти замість обчислення складних формул підвищує продуктивність до 25%.
3. Вперше запропоновано метод текстуровання, особливість якого полягає у використанні неортогональної растеризації, що дозволяє зменшити $(T-q)$ операцій ділення, де q - кількість горизонтальних рядків, за якими проводиться растеризація трикутника проекції екранного пікселя в текстурний простір, а T - кількість внутрішніх точок трикутника.

Рисунок Г.25 – Наукова новизна отриманих результатів

Практична цінність отриманих результатів:

Практична цінність полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби текстуровання для систем візуалізації тривимірних зображень.

Рисунок Г.26 – Практична цінність отриманих результатів



Публікації:

За темою магістерської роботи надруковано 23 праць у матеріалах республіканських і міжнародних конференціях.

Рисунок Г.27 – Публікації



Дякую за увагу!

Рисунок Г.28 – Дякую за увагу