Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Методи та програмно-апаратні засоби рендерингу тривимірних графічних схем»

Виконав: студент 2-го курсу,
групи 2ПІ-20м
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Озерчук Д. А.
(прізвище та ініціали)

Керівник: д.т.н., професор каф. ПЗ
Романюк О. Н.
(прізвище та ініціали)
«____»_____2021 р.

Опонент: д.т.н., професор каф. КН
Василевський О. М.
(прізвище та ініціали)
«____»_____2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«____»_____2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-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.
« 13 » вересня 2021 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Озерчуку Дмитру Анатолійовичу

1. Тема роботи – методи та програмно-апаратні засоби формування тривимірних графічних схем.

Керівник роботи: Романюк Олександр Никифорович, д.т.н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від « 24 » вересня 2021 р. № 277.

2. Строк подання студентом роботи
1 грудня 2021 р.

3. Вихідні дані до роботи: базові методи зафарбовування – методи Гуро, Фонга; моделі освітлення – моделі Бліна, Фонга; графічний режим – TrueColor; вихідні дані для зафарбовування – напрямки векторів нормалей до вершин трикутників; дані про розташування джерела світла та спостерігача; коефіцієнт спекулярності поверхні; координати вершин трикутників; максимальна розрядність для задання адрес вершин трикутників – 12; вихідні дані – кінцеве зображення, зафарбоване згідно методу Фонга в поєднанні з моделлю освітлення Бліна або Ламберта.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз методів і програмно-апаратних засобів рендерингу тривимірних графічних схем, розробка методів визначення інтенсивності кольору з використанням поліномів, розробка методів прискореного обчислення інтенсивності кольору точок поверхонь тривимірних об'єктів, програмна реалізація методів

формування тривимірних графічних схем, експериментальні дослідження; економічна частина.

5. Перелік графічного матеріалу: обґрунтування вибору теми дослідження; наукова новизна отриманих результатів; методи прискореного зафарбовування тривимірних об’єктів; експериментальні дослідження; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О. Н., д.т.н., завідувач кафедри ПЗ		
5	Бурєннікова Н. В., д.е.н., проф. кафедри ЕПВМ		

7. Дата видачі завдання 14 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів і програмно-апаратних засобів рендерингу тривимірних графічних схем	15.09.2021-26.09.2021	Вик.
2	Розробка методів визначення інтенсивності кольору з використанням поліномів	27.09.2021-15.10.2021	Вик.
3	Розробка методів прискореного обчислення інтенсивності кольору точок поверхонь тривимірних об’єктів	16.10.2021-7.11.2021	Вик.
4	Програмна реалізація методів формування тривимірних графічних схем. Експериментальні дослідження	8.11.2021-21.11.2021	Вик.
5.	Економічна частина	22.11.2021-30.11.2021	Вик.

Студент _____ **Озерчук Д. А.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Романюк О. Н.**
(підпис) (прізвище та ініціали)

Опонент магістерської кваліфікаційної роботи _____ **Василевський О. М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.92

Озерчук Д. А. Методи та програмно-апаратні засоби рендерингу тривимірних графічних схем. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 132 с.

На укр. мові. Бібліогр.: 35 назв; рис.: 39; табл.: 9.

У магістерській кваліфікаційній роботі проведено аналіз методів і засобів рендерингу, обґрунтовано необхідність підвищення його продуктивності та реалістичності. Запропоновано метод зафарбовування, де з обчислювального процесу вилучені нормалізації векторів і розрахунок арккосинуса кута в циклі підготування. Подальшого розвитку отримав метод прискореного визначення дифузної та спекулярної складових кольору, що одночасно й незалежно визначає інтенсивності кольору двох точок рядка растеризації. Побудовано обчислювальний процес, який не містить у циклі розрахунку складових кольору операцій ділення та знаходження квадратного кореня. Запропоновано метод зафарбовування з використанні поверхні другого порядку для визначення інтенсивностей кольору точок поверхні.

Розроблені в роботі методи та засоби формування тривимірних графічних схем можна використати в системах тривимірної комп’ютерної графіки.

Графічна частина складається з 14 плакатів з презентацією МКР.

В економічній частині окреслено комерційний потенціал розробки, витрати на виконання науково-дослідної роботи та конструкторсько-технологічної роботи, комерційні ефекти від реалізації результатів розробки, ефективність вкладених інвестицій та період їхньої окупності.

Ключові слова: тривимірна графіка, рендеринг, зафарбовування, інтенсивність кольору, растеризація.

ABSTRACT

Ozerchuk D. A. Methods, software and hardware for rendering three-dimensional graphic schemes. Master's thesis in specialty 121 – software engineering. Vinnitsa: VNTU, 2021. 132 p.

In Ukrainian language. Bibliographer: 35 titles; fig.: 39; tabl.: 9.

In this master's thesis methods and means of rendering are analyzed, the necessity of its productivity and realism increase is proved. A method of shading is proposed, where normalization of vectors and calculation of angle's arc cosine in the preparation cycle are removed from the computational process. The method of diffuse and specular color components accelerated determination was further developed, which simultaneously and independently determines color intensities of two points on the rasterization line. A computational process is constructed that does not contain operations of division and finding the square root in the color components' calculation cycle. A shading method using a surface of the second order to determine color intensities of surface points is proposed.

Methods and means of forming three-dimensional graphic schemes developed in this work can be used in 3D computer graphics systems.

The graphical part consists of 14 posters with the master's thesis presentation.

In the economic section, issues such as commercial potential, cost of research and development, commercial effects of the implementation, investments' effectiveness and their payback period are outlined.

Keywords: 3D graphics, rendering, shading, color intensity, rasterization.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Засоби графічної візуалізації використовуються практично в усіх сферах діяльності людини, оскільки в цьому випадку найпростіше організувати ефективний інтерфейс між комп'ютером і людиною. Це пояснюється високою інформативністю графічних зображень і оперативністю сприйняття графічної інформації людиною через зоровий інформаційний канал. На даному етапі розвитку комп'ютерної графіки особливу увагу приділяють тривимірним зображенням, що найбільш адекватно відтворюють процеси або об'єкти.

При формуванні тривимірних зображень використовують моделі з високим рівнем деталізації поверхні, для кожної точки якої з урахуванням багатьох факторів визначають інтенсивності кольору та екранні координати. Це призводить до довготривалої візуалізації графічних об'єктів.

Оскільки етап кінцевої візуалізації є найтрудомісткішим етапом формування тривимірних зображень і становить 60-80 % від загального обсягу обчислень [1], доцільною є розробка методів і засобів, які б забезпечили суттєве спрощення процедур рендерингу як на програмному, так і апаратному рівнях, що дасть змогу скоротити тривалість візуалізації складних проектів.

У даний час розширюються сфери застосування комп'ютерної графіки, де вимагається побудова динамічних зображень у реальному часі. При цьому висуваються жорсткі часові обмеження, які ще більше посилюються при збільшенні деталізації поверхонь, роздільної здатності екрана та використанні складних моделей освітлення. Формування в цьому випадку динамічних зображень у реальному масштабі часу наштовхується на можливості сучасних засобів комп'ютерної графіки, що створює протиріччя, для розв'язання якого необхідний розвиток теорії кінцевої візуалізації в напрямку розробки моделей та методів, орієнтованих, насамперед, на апаратну реалізацію.

У зв'язку з цим тема магістерської кваліфікаційної роботи є актуальною і відповідає пріоритетним напрямкам розвитку науки та техніки.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою роботи є підвищення продуктивності та реалістичності формування графічних зображень за рахунок використання нових методів зафарбовування .

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів формування графічних зображень для визначення напрямків підвищення їх продуктивності та реалістичності;
- запропонувати нові:
  - методи підвищення продуктивності формування тривимірних графічних схем;
  - методи підвищення реалістичності зафарбовування;
- розробити алгоритми та програмні компоненти на основі запропонованих методів;
- провести експериментальні дослідження розроблених засобів рендерингу.

Об’єкт дослідження – процес кінцевої візуалізації тривимірних об’єктів у системах комп’ютерної графіки.

Предмет дослідження – методи та засоби зафарбовування тривимірних графічних об’єктів.

Методи дослідження. У процесі досліджень використовувались: теорія чисел та чисельних методів, теорія диференціально-інтегрального числення, лінійна алгебра, методи аналітичної геометрії для розробки моделей і методів зафарбовування тривимірних об’єктів; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Вперше запропоновано метод зафарбовування, особливість якого полягає у вилучені з обчислювального процесу нормалізації векторів і

розрахунку арккосинуса кута в циклі підготування. що дозволило зменшити час зафарбовування середньостатистичного трикутника порівняно з класичною реалізацією в середньому на 25% .

2. Подальшого розвитку отримав метод прискореного визначення дифузної та спекулярної складових кольору, особливість якого полягає в одночасному й незалежному визначенні інтенсивностей кольору відразу двох точок рядка растеризації. Обчислювальний процес, який реалізовано згідно з запропонованим підходу, не містить у циклі розрахунку складових кольору операцій ділення та знаходження квадратного кореня, які мають місце при класичній реалізації Це дозволило підвищити продуктивність та реалістичність рендерингу.

3. Вперше запропоновано метод зафарбовування, особливість якого полягає у використанні поверхні другого прядку для визначення інтенсивностей кольору точок поверхні. Це дозволило підвищити продуктивність рендерингу на 67% порівняно з методом Фонга.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби зафарбовування для комп’ютерних систем високореалістичної візуалізації тривимірних зображень.

Особистий внесок здобувача. Усі наукові результати, викладені в магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: [2] – формули для апроксимації, [3] – формули для визначення інтенсивності кольору сусідніх пікселів, [4] – аналіз методів текстурування, [5] – аналіз методів розпаралелення обчислювального процесу при використанні спарок відеокарт, [6] – формули для оцінювальної функції, [7] – класифікація методів побудови розгорток і поверхонь, [8] – програма для порівняння зображень з використанням перцептивних хеш-методів, [9] – аналіз методів формування тривимірних моделей на основі растрового зображення, [10] – методологія

дослідження пакетів прикладних програм, [11] – аналіз методів формування графу на основі зображення.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2020), XIV міжнародній науково-практичній конференції. «Інформаційні технології та комп’ютерна інженерія»» (Одеса, 2020), Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів. «Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації» (Одеса, 2020), XIV міжнародній науково-практичній конференції «Інформаційні технології і автоматизація». (Одеса, 2021), Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2021).

Публікації. Основні результати досліджень опубліковано в 10 наукових працях, у тому числі 4 статті у фаховому виданні України , 6 – у матеріалах конференцій.

1 АНАЛІЗ МЕТОДІВ І ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ РЕНДЕРИНГУ ТРИВИМІРНИХ ГРАФІЧНИХ СХЕМ

1.1 Особливості формування тривимірних графічних схем

Теорії та методи, пов’язані з відтворенням тривимірних сцен на двовимірній площині, розроблялися впродовж тривалого часу. Спочатку зображення формували вручну на фізичних об’єктах (стіна, дошка, шкіра, тканина, папір). Щоб спроеціювати навколишній або уявний тривимірний світ на двовимірну площину застосовували уяву, емпіричні знання та математику.

Набагато пізніше з’явились фотографії та кіно, в яких проеціювання відбувалось за допомогою лінз, що направляли промені світла на світлочутливу поверхню (плівка, матриця). Нарешті, з розвитком комп’ютерних технологій з’явилась комп’ютерна графіка. Для переносу тривимірних об’єктів на двовимірну площину використовуються геометричні перетворення. Об’єкти розбиваються на геометричні примітиви (точки-вершини, лінії, трикутники) в тривимірному просторі, далі для кожного з об’єктів обчислюється проекція.

З появою комп’ютерів обчислювати таку абстрактну модель реальності стало значно простіше. Зі збільшенням обчислювальних потужностей з’явилась можливість виконувати обчислення проекцій для більшої кількості примітивів, що дозволило створювати високодеталізовані та, навіть, реалістичні зображення. Сьогодні тривимірну комп’ютерну графіку використовують у кіноіндустрії, візуалізації даних та інтерактивних програмних засобах (наприклад, комп’ютерних іграх).

На початку 90-х років минулого століття стрімко набирають популярність комп’ютерні та консольні ігри з програмно візуалізованою тривимірною графікою. У цей період почалась конкуренція в галузі апаратного забезпечення для прискорення візуалізації (графічних процесорів), яку можна було порівняти з перегонами за потужністю центральних процесорів.

За законом Мура, «кількість транзисторів на інтегральній схемі подвоюється приблизно кожні два роки» [5]. Паралельно з розвитком апаратного забезпечення розроблялися й прикладні програмні інтерфейси для тривимірної графіки (OpenGL, Direct3D).

Рендеринг (візуалізація) – процес створення двовимірних зображень на основі тривимірної моделі. Обсяг даних, які потрібно обробити в процесі візуалізації, залежить від складності сцени та об’єктів, які утворюють сцену. Рендеринг можна виконувати або заздалегідь (попередній рендер), або інтерактивно (рендеринг у реальному часі).

Попередній рендер застосовують тоді, коли важливий лише кінцевий результат (висока деталізованість, реалістичність зображення), є вдосталь часу та немає інтерактивності; галузі його застосування – фільми та анімація, візуалізація даних. Рендеринг одного зображення може тривати від кількох годин до кількох днів, основне навантаження при цьому припадає на центральний процесор. Для отримання найкращого результату використовують високоякісні тривимірні моделі та текстури, алгоритми рендерингу без статистичного зміщення та реалістичні фізичні моделі освітлення. Для пришвидшення процесу використовують потужні багатоядерні процесори, або мережу комп’ютерів.

Візуалізація в реальному часі дозволяє створювати зображення зі швидкістю, достатньою для інтерактивної взаємодії. На відміну від попередньо візуалізованих, дані зображення повинні бути сформовані за частку секунди, оскільки вони залежать від інтерактивно введених користувачем даних. Починаючи з 12 кадрів за секунду можна говорити про рендеринг у реальному часі, проте на такій частоті досі можна виділити окремі кадри візуально. Історично склалося, що фільми використовують 24 кадри за секунду для створення плавного руху та непомітності окремих кадрів. Тривалий час ігрові консолі були обмежені частотою 30 кадрів за секунду, проте сучасні представники знаходяться на рівні 60-120 кадрів за секунду.

Комп’ютери можуть формувати й більше (наприклад, 240 кадрів за секунду). Проте, при частоті 72 кадри за секунду вже майже неможливо візуально виділити окремі кадри [12]. Варто відмітити, що кількість кадрів за секунду означає лише скільки в середньому може бути сформовано кадрів за секунду, час на формування кожного кадру залежить від складності сцени та не є константним та однаковим для всіх кадрів, порівняно з частотою оновлення екрану, де частота оновлення екрану в 60 Герц означає, що екран оновлюється 60 раз за одну секунду.

Потреба в створенні більш складних сцен та реалістичних зображень у реальному часі призвела до появи апаратних прискорювачів – графічних процесорів. Архітектура візуалізації побудована у вигляді графічного конвеєра, який містить як незмінні етапи, так і етапи з можливістю програмування – шейдери [13].

Тривимірні графічні схеми – це проекції тривимірних об’єктів на двовимірну екранну площину. Процес формування таких схем можна описати як ланцюг дій; в комплексі всі дії утворюють тривимірний графічний конвеєр. Сукупності елементів ланцюга дій процесу формування тривимірних графічних схем називають кроками, етапами або стадіями тривимірного графічного конвеєра.

Систематизація розрахунків у вигляді конвеєра є на сьогодні найпоширенішим архітектурним рішенням для розробки апаратних (графічні прискорювачі – графічні процесори) та програмних (прикладні програмні інтерфейси, графічні бібліотеки) засобів роботи з графікою.

На рисунку 1.1 представлено етапи еталонного тривимірного графічного конвеєра з групуванням на підсистеми. При впровадженні реального тривимірного графічного конвеєра можливі відмінності реалізації на апаратному та програмному рівнях, проте зміст етапів тривимірного графічного конвеєра залишається без суттєвих змін.

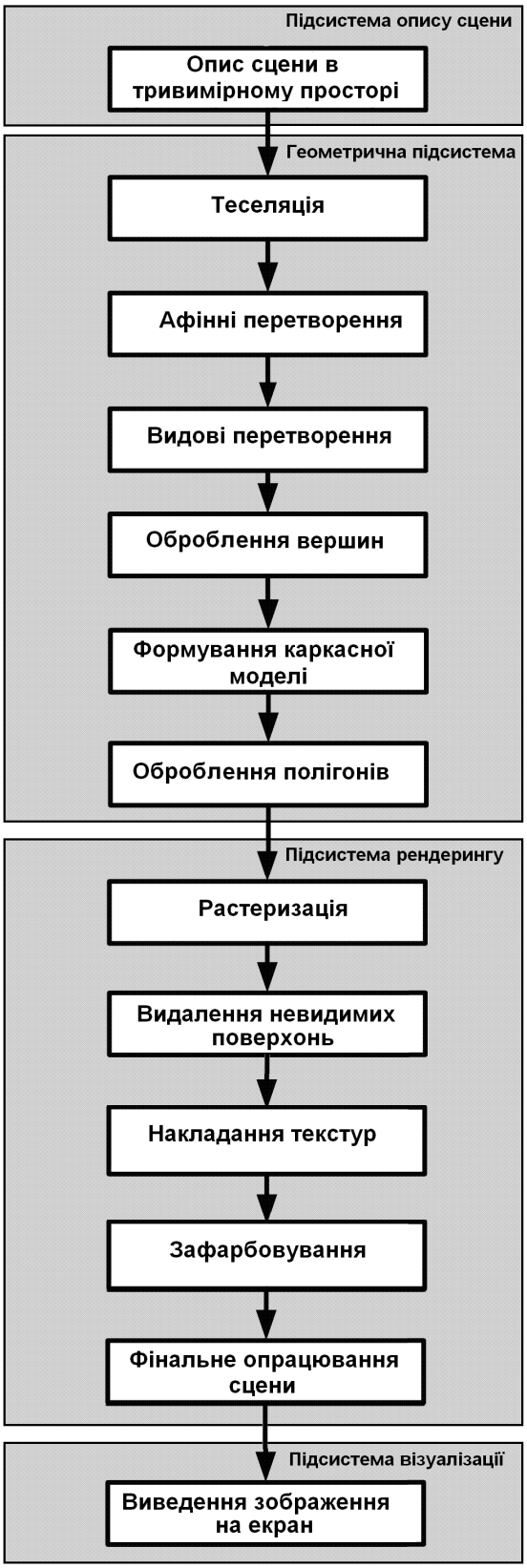


Рис. 1.1. Основні етапи графічного конвеєра

У процесі формування тривимірної графічної схеми можна виділити такі стадії: етап опису тривимірної графічної схеми (сцени в тривимірному просторі), етап геометричних трансформацій, етап візуалізації (рендерингу). На етапі опису сцени встановлюються атрибути об’єктів, з яких складається сцена, їхня позиція в просторі сцени та обирається план виконання операцій над об’єктами.

На етапі геометричних трансформацій графічна сцена декомпонується, виконуються афінні перетворення над отриманими компонентами (об’єктами). Відбувається перенесення з глобального простору сцени в локальний простір спостерігача, виконується відсікання, вилучення невидимих граней. Отримані результати конвертуються в екранний простір. Далі знаходяться параметри вершин тривимірної схеми, включно з їхнім розташуванням в екранній системі координат, вектори нормалей, освітленість, текстурні координати.

Найважливішими на цьому етапі вважаються процедури перетворень і освітлення, тому весь етап оброблення вершин часто називають T&L (Transformation and Lighting) [14]. Також відбувається групування трикутників, формується каркасна модель. Вхідні примітиви обробляються як цілісні об’єкти на етапі обробки багатокутників; за потреби формуються нові примітиви. Цей етап з’явився в тривимірному графічному конвеєрі порівняно нещодавно – з впровадженням стандартів DirectX 10 і OpenGL 2.1 [15].

Комп'ютерна візуалізація (рендеринг) – це сукупність способів створення геометричних образів на основі абстрактних описів об’єктів. Під час візуалізації враховують специфікацію об'єктів, їхні атрибути, взаємне розташування, динаміку та способи взаємодії [13].

Етап кінцевої візуалізації (або етап рендерингу) – це стадія формування пікселів зображення на основі результатів етапу геометричних трансформацій. Для пікселів обчислюються екранні координати та інтенсивності кольору. У тривимірному графічному конвеєрі етап рендерингу є найбільш трудомістким, оскільки він побудований на виконанні дій над окремими пікселями, потребує складних обчислень і використовує моделі, в основі яких – функції багатьох

змінних, тому в процесі враховується багато параметрів. Рендеринг у загальному обсязі обчислень формування тривимірних графічних схем складає 60-80 % [1], тому цей етап визначає продуктивність графічних систем і є одним з «вузьких місць» (bottleneck), які потребують оптимізації.

Останніми роками тривимірний графічний конвеєр почав містити етапи з можливістю програмування, дії в яких виконують шейдери. Шейдер – це програма, що застосовується в тривимірній графіці для обчислення кінцевих параметрів об’єкта чи всього зображення. Тривимірний графічний конвеєр може містити вершинний (для обробки вершин), геометричний (для обробки полігонів) та піксельний (для обробки пікселів) шейдери [16]. Найнавантаженіші (через кількість об’єктів та складність обчислень) – піксельні шейдери.

Фотореалістичність у комп’ютерній графіці досягається точним відтворенням властивостей поверхні та фізично правильним описом ефектів освітлення сцени. Розрахунок освітленості точок поверхні часто розбивається на дві основні задачі.

Перша – визначення способу розрахунку освітленості в заданій точці тривимірного простору; вона вирішується за допомогою побудови математичної моделі освітлення.

Друга задача використовує модель освітлення для розрахунків освітленості тривимірних об'єктів із конкретною геометрією й властивостями поверхні та вирішується за допомогою так званої моделі зафарбовування (Shading model) [14].

Модель освітлення дає можливість обчислити інтенсивність світла, що випромінюється з конкретної точки поверхні у вибраному напрямі. Для цього потрібно задати оптичні властивості поверхонь об’єктів, з яких складається сцена, відносне розташування поверхонь на сцені, орієнтацію площини спостереження, кольори та розташування джерел світла, властивості джерел світла.

Більшість програмних пакетів для оптимізації розрахунків використовує емпіричні моделі [17]. Вони побудовані на спрощених фотометричних

розрахунках. Якщо використовувати глобальні моделі освітлення [14], то тривимірну сцену можна розглядати як єдину систему, опис освітлення в якій виконується з урахуванням взаємного впливу об'єктів. Під час використання даного підходу обчислюють множинне відбиття й заломлення світла. На виході можна одержати високоякісні зображення. Проте, через значний обсяг розрахунків, формування графічної сцени потребує часу (години, навіть дні) [1]. Зрозуміло, що широке застосування подібних моделей у системах візуалізації реального часу можливе лише в перспективі використання більш потужних апаратних засобів.

Використання локальної моделі освітлення [14] для формування зображень дає можливість обмежити взаємодію до однократного відбиття світла від поверхні. Дифузна та спекулярна складові кольору розраховуються без змін, а розсіяне світло апроксимується. Подібні моделі сьогодні набули найбільшого поширення, вони застосовуються в системах візуалізації реального часу.

Реалізм – це об'єктивне, правдиве відтворення дійсності за допомогою спеціальних засобів [12]. Алгоритми для створення тривимірних зображень розробляють на основі математичних моделей реальних фізичних процесів та явищ. Моделі є наближеними, тому зображення не повністю відтворює реальний об’єкт таким, яким його спостерігають.

Фотореалістична графіка [17] є однією з головних цілей досліджень у галузі комп’ютерної графіки. Це сукупність методів і засобів формування реалістичних зображень, які важко або практично неможливо відрізнити від звичайних фотографій, або можна ототожнити з ними.

Реалістичність графіки залежить від точності відтворення як конструктивних рис об'єкта, так і визначеного з урахуванням освітлення кольору його поверхонь. Реалізм тривимірної сцени також залежить від ілюзії об’ємності, яка побудована на зміні кольору об’єктів.

Метод Гуро [18] – один з методів зафарбовування, які дають змогу сформувати достатньо реалістичні зображення. Він найбільш поширений, його

легко реалізувати, проте він має ряд артефактів та його реалістичність залежить від кількості трикутників.

Методи, що враховують локальну кривизну поверхні та використовують три складові кольору (фонову, дифузну та спекулярну) для передачі кольору поверхні дозволяють створювати більш реалістичні зображення.

Метод Фонга [19] – один з таких методів. Даний метод можна обрати як еталон для порівняння з іншими новоствореними методами. Ступінь відмінності сформованих двома методами зображень буде вважатися мірою реалістичності.

Відтворення конструктивних рис об’єкта, тобто ступінь деталізації, залежить від щільності тріангуляційної мережі. Вважається [17], що для екрана з роздільною здатністю 3840×2160 (4К) середня кількість полігонів у сцені до 1000000 забезпечує задовільну за сучасними мірками якість зображення, 5000000 полігонів забезпечують якість, близьку до побутового відео, понад 10000000 полігонів у сцені дозволяє наблизитися до фотореалізму. Сучасні комп'ютерні та консольні ігри використовують у середньому 3000000–5000000 полігонів на сцену [20]. Відеокарта повинна формувати зображення зі швидкістю більш ніж 30 кадрів у секунду (менше 33 мілісекунд на кадр). У найближчі роки очікуються ігри [21], що використовуватимуть до 10000000–30000000 полігонів у сцені. Якість зображень об'єктів залежить від якості апроксимації, це припускає подальше зростання кількості вершин полігонів у моделях, причому темпи цього зростання обганяють темпи росту швидкості графічних прискорювачів [5].

Кожна вершина трикутника описується структурою даних об’ємом мінімум 12 байт. Для координат (x, y, z) вершини в просторі, параметрів нормалей у вершинах координат текстури та параметрів кольору потрібно по 3 байти відповідно. Трикутник як об’єкт, що описується трьома вершина, в цілому потребує 36 байтів. Застосування ефектів графічних бібліотек (прикладом такого ефекту може бути анізотропна фільтрація) збільшує об’єм даних для опису одного трикутника до 768 байт. У середньому опис одного трикутника займає 180 байт [22]. Для сцени з мільйоном трикутників

доведеться виділити більше 180 Мбайт на носії для зберігання або в оперативній пам’яті під час роботи зі сценою. При наявності мільйона трикутників у сцені загальний об'єм даних складе близько 180 Мбайтів. При швидкості зміни кадрів в іграх приблизно 30 разів у секунду загальний потік даних перевищить 5 Гбайтів/с.

Синтез реалістичних зображень передбачає великий обсяг обчислень, який визначається швидкістю зміни кадрів і складністю геометричних сцен. Так, наприклад, для моделі персонажа з близько 200000 полігонів що включають 400000 вершин при частоті зміни кадрів 30 кадрів/с необхідна швидкість обчислень складе близько $1{,}2 \times 10^{11}$ операцій/с із плаваючою крапкою та 2×10^{13} операцій/с з фіксованою крапкою [23]. Причому таких персонажів у сцені може бути декілька, що збільшує кількість обчислень в рази.

Наведені розрахунки свідчать, що формування графічних зображень у реальному часі є надзвичайно вимогливим до ресурсів і потребує застосування апаратного прискорення для реалізації.

1.2 Аналіз моделей освітлення

Розуміння світла та його взаємодії з об’єктами – важливий компонент процесу створення зображень. Біле світло містить усі кольори, які бачить людина [20]. Для прикладу, його можна отримати поєднавши світло червоного, зеленого та синього кольорів на чорному фоні, як показано на рисунку 1.2.

У комп’ютерній графіці використовується колірна модель RGB, побудована за схожим принципом, де усі кольори описуються через значення трьох колірних компонентів – червоного, зеленого та синього. Розглянемо варіант моделі, де ці значення знаходяться в діапазоні від 0 до 1. Опишемо колір як тривимірний вектор (r, g, b).

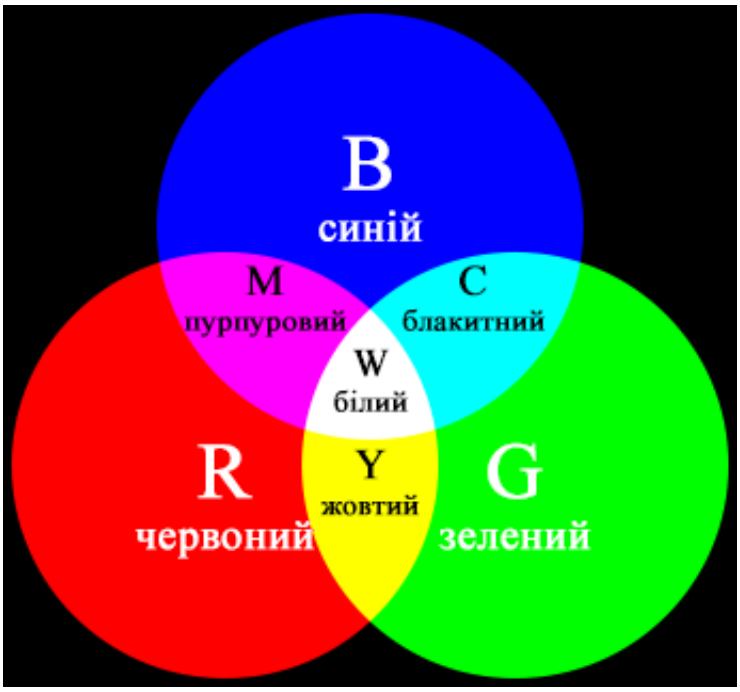


Рисунок 1.2 – Білий колір як результат змішування світла трьох кольорів (червоного, зеленого та синього)

Не лише світло, але й інші об’єкти навколишнього світу мають колір. Розглянемо, що відбувається, коли світло одного кольору потрапляє на поверхню іншого кольору. Якщо світло білого кольору падає на поверхню червоного кольору, то червона поверхня поглинає синій та зелений компоненти білого світла та відбиває червоне світло. Це можна продемонструвати на прикладі системи RGB, виконавши операцію добутку для відповідних компонентів кольору Якщо ж на червону поверхню направити блакитне світло, то поверхня поглине його повністю

Однією з найпростіших моделей освітлення є модель Фонга. Для освітлення зображень вона пропонує використовують три складові – амбієнтну, дифузну та спекулярну.

Амбієнтна складова – найпростіший компонент, базовий колір поверхні об’єкта. Без нього неосвітлені поверхні були б забарвлені в чорний колір, що зазвичай виглядає неприродно. Амбієнтна складова формується шляхом множення кольору поверхні на колір світла, а потім скалярного множення значення відображеного кольору на коефіцієнт навколишнього середовища. У

наведеному прикладі кінцевим результатом є тьмяний червоний колір, на який налаштована поверхня, якщо на неї не потрапляє джерело світла.

Дифузна складова – найважливіший компонент моделі, оскільки визначає основний колір поверхні при попаданні на неї світла. Дифузна складова пропорційна кольору відбитого світла та косинусу кута між вектором нормалі та вектором, спрямованим до джерела світла. Сам кут неважливий, оскільки косинус обчислюється з формули скалярного векторного добутку. У випадку від'ємного значення косинуса його значення приймають рівним нулю.

Спекулярна складова використовується для розрахунку блиску поверхні. «Спекулярний» означає «дзеркальне відображення», він отримав свою назву від реального явища, коли світло відбивається від блискучої поверхні, перш ніж він встигає змішатись із кольоровим шаром поверхні [17]. Ідеальна дзеркальна поверхня схожа на дзеркало, де світло відображається під таким самим кутом, під яким він потрапляє на поверхню. Якщо світло падає на поверхню під кутом 20 градусів – воно буде відбиватися від неї під таким самим кутом [14]. Якщо кут між спостерігачем та відбитим променем світла досить малий, то колір, який бачить людина, буде яскравим відбитим кольором (наприклад, білим). Як і у випадку дифузної складової, вектор нормалі до поверхні важливий для цих розрахунків. Для розрахунку спекулярної складової необхідні вектори від поверхні до світла та від поверхні до камери. Ці вектори показані рисунку 1.3 як вектори L і V [24].

Вектор R це вектор ідеального відображення, тобто, дзеркальна версія напрямку світла. Основна мета тут – розрахувати кут між відбитим вектором R і вектором V камери. Якщо кут малий, світло відбивається в бік камери. Як і в реальному житті, деякі об'єкти блищать сильніше за інші [17]. Тому кут залежить від постійної величини, що відображає блиск матеріалу.

Кожна поверхня має свій коефіцієнт спекулярного кольору. Всі ці вектори повинні бути нормалізовані в одиничні вектори, щоб отримати правильний результат.

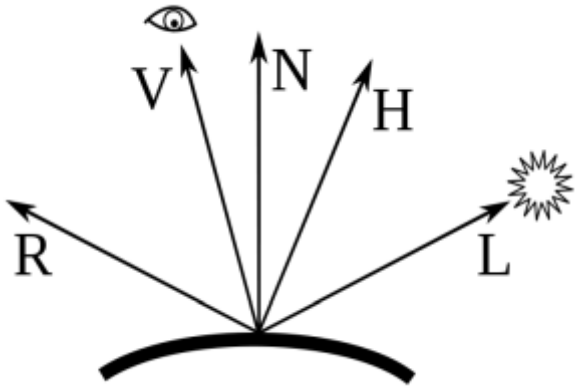


Рисунок 1.3 – Вектори, які використовують у процесі обчислення кольору тривимірної графічної схеми

Існує також покращена версія моделі Фонга, яка називається моделлю відображення Блінна-Фонга. У цій моделі відображення обчислюється вектор половини шляху H між вектором камери V і вектором світла L. Вектор H можна обчислити за допомогою додавання векторів L і V. Знову ж таки, важливо нормалізувати вектори в одиничні вектори [25]. Це ефективніший спосіб, оскільки вектор H економить процес обчислення добутку точок, який інакше використовується для обчислення вектора R [24]. У цьому методі спекулярний коефіцієнт може бути розрахований шляхом взяття скалярного добутку векторів H і N замість R і V [26].

Якість візуалізації тривимірних графічних схем сильно залежить від використаної моделі освітлення.

1.3 Аналіз дистрибутивних функцій відбивної здатності поверхні

Дистрибутивна функція відбивної здатності поверхні (ДФВЗ) дозволяє знайти відбивну здатність поверхні залежно від геометрії освітлення та геометрії огляду, визначає, яку частку випромінювання, що надійшло в точку з напрямку джерела світла, буде відбито в напрямку спостерігача. ДФВЗ є функцією кількох змінних

$$ДФВЗ = f_{\lambda}(\alpha, \mu_1, \varphi, \mu_2, x) = f(\vec{L}, \vec{V}, \vec{N}),$$

де λ – довжина хвилі, $(\alpha, \mu_1), (\varphi, \mu_2)$ – параметри, що визначають відповідно напрямок падаючого світла і напрямок спостереження.

ДФВЗ залежить від довжини хвилі і визначається структурними та оптичними властивостями поверхні, такими як тінеутворення, багаторазове розсіювання, взаємне затінення, пропускання, відображення, поглинання та випромінювання елементами поверхні, розподіл орієнтації граней та щільність граней.

ДФВЗ використовується в дистанційному зондуванні для корекції ефектів кута огляду та освітленості (наприклад, при стандартизації зображень), для отримання альбедо. Дана функція просто описує те, що спостерігаємо щодня: об'єкти виглядають по-різному, якщо дивитися на них під різними кутами та при освітленні з різних напрямків, як показано на рисунку 1.4.

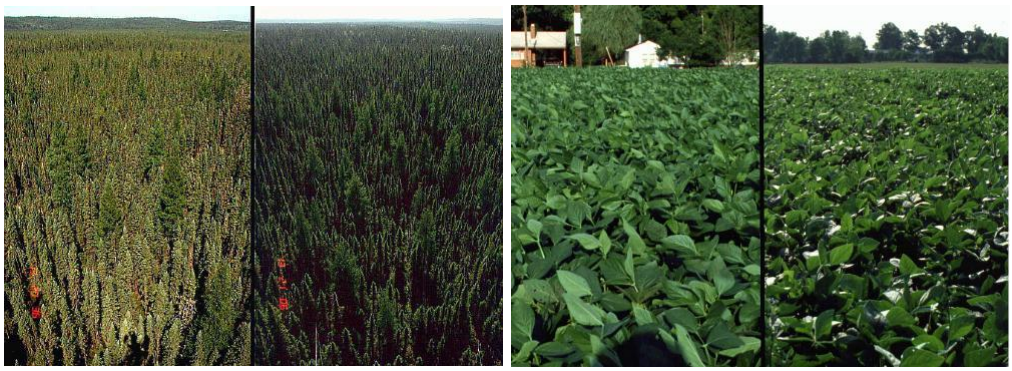


Рисунок 1.4 – Ліс та соєве поле коли джерело світла знаходиться позаду спостерігача та перед спостерігачем

З цієї причини художники та фотографи століттями вивчали зовнішній вигляд дерев та міських територій за різних умов. Як і художники, програмісти тривимірної графіки також повинні обґрунтувати ДФВЗ поверхонь, які вони використовують.

Дистрибутивну функцію відбивної здатності поверхні можна виміряти на реальних об’єктах, проте в комп’ютерній графіці частіше використовуються ДФВЗ, основані на моделях освітлення.

При моделюванні, ДФВЗ поверхні будується з окремих компонентів. Ці компоненти описують стандартний набір можливих типів взаємодії світла та поверхні: розсіювання, глянцеве та дзеркальне відбиття, перехід з одного середовища в інше (рис. 1.5).

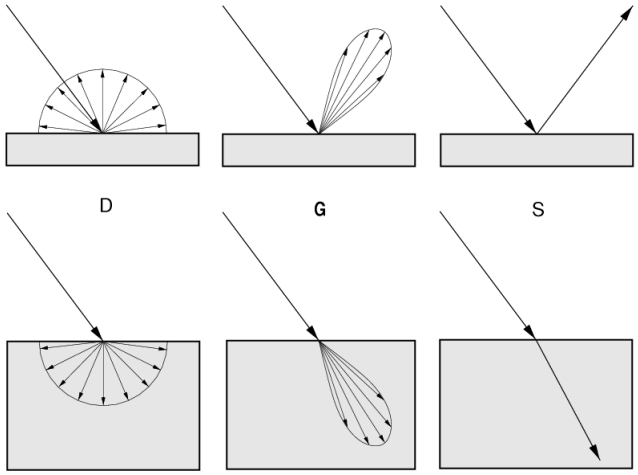


Рисунок 1.5 – Деякі варіанти взаємодії світла з поверхнею

На рисунку 1.6 зображено класифікацію локальних моделей освітлення. Нижче штрихової лінії наведено найпоширеніші на даний час ДФВЗ. Локальні моделі освітлення можна розділити на дві групи.

Перша містить ті ДФВЗ поверхні, які отримано експериментальним шляхом. Вони найбільш поширені, оскільки достатньо прості та забезпечують прийнятну реалістичність відтворення об’єктів, для яких не вимагається точна фізична інформація про освітлення.

До другої групи відносять більш точні моделі, які, як правило, враховують як корпускулярну, так і хвильову природу світлового потоку. В таких моделях поверхня розглядається у вигляді мікроскопічних ділянок.

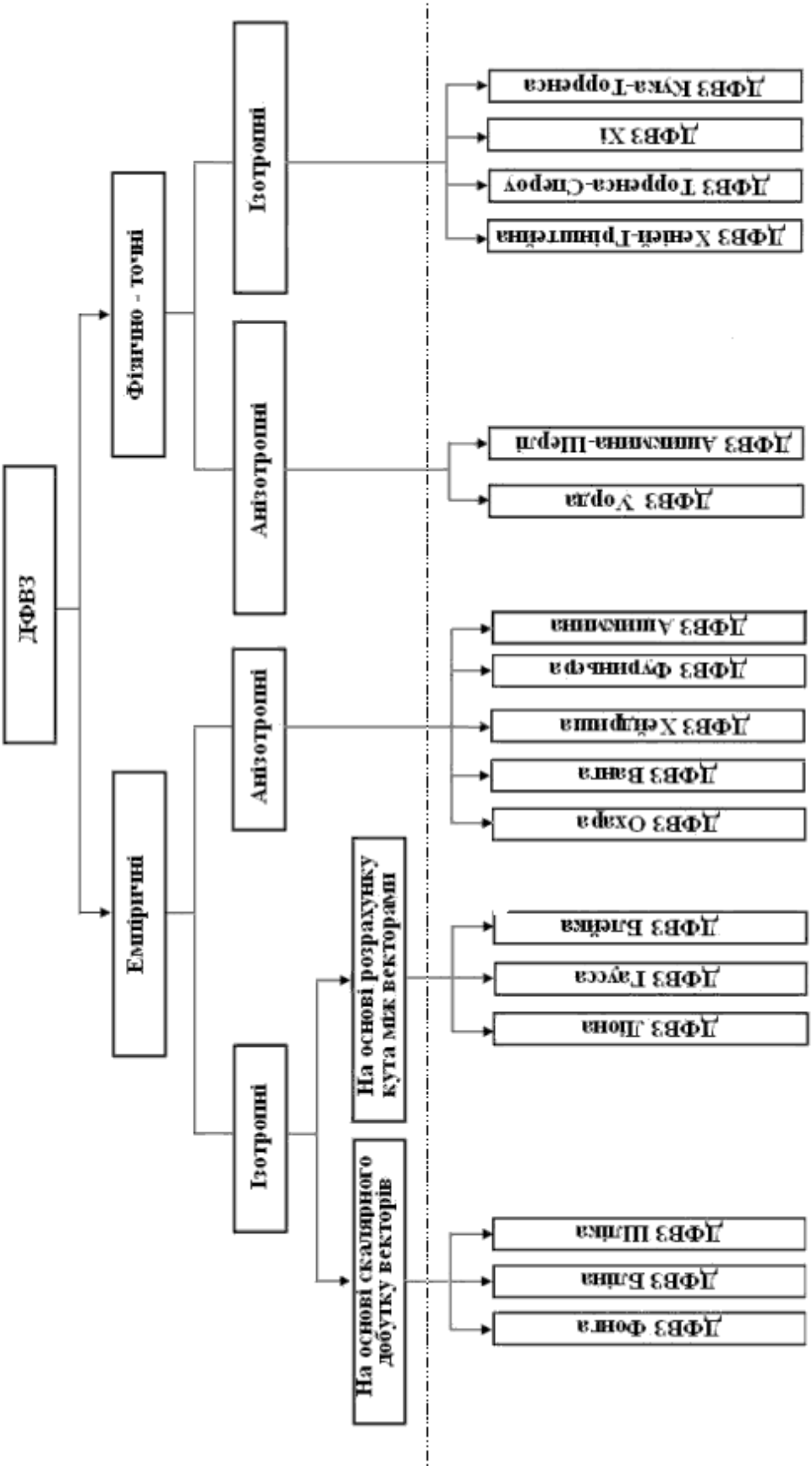


Рисунок 1.6 – Класифікація ДФВЗ

ДФВЗ можна розділити на ізотропні та анізотропні. Ізотропні функції інваріантні відносно повороту навколо вектора нормалі до поверхні. Навпаки, в анізотропних дистрибутивних функціях відбивні властивості матеріалу змінюються при повороті навколо вектора нормалі.

У більшості випадків ДФВЗ визначається через скалярний добуток відповідних векторів [14]. Значно рідше оперують зі значенням кута між векторами, оскільки це передбачає обчислення обернених тригонометричних функцій.

Формули для розрахунку ДФВЗ мають достатньо велику обчислювальну складність. У переважній більшості випадків вони містять трудомісткі операції (ділення, піднесення до степеня і т.п.), а тому орієнтовані на програмну реалізацію. Більшість ДФВЗ передбачають виконання векторних операцій. Оскільки дистрибутивна функція й нормалізовані вектори обчислюються для кожної точки поверхні, то продуктивність формування графічних сцен значною мірою визначається саме реалізацією зазначених процедур.

Теоретичних основ апаратної реалізації ДФВЗ не розроблено. Доцільна розробка ДФВЗ, які б порівняно з існуючими мали меншу степінь, обчислювальну складність і використовували прості з апаратної точки зору операції.

1.4 Аналіз методів зафарбовування тривимірних графічних схем

На ранніх етапах розвитку візуалізації тривимірних графічних схем для створення ілюзії об’єму використовувалися різні підходи до зафарбовування. Один із найочевидніших – просто зафарбовувати багатокутники в різні кольори.

Більш реалістично виглядає метод, де яскравість кольору багатокутника обернено пропорційна відстані до спостерігача, так що об’єкти стають темнішими з віддаленням від глядача. У результаті створюється ілюзія наявності освітлення попереду сцени.

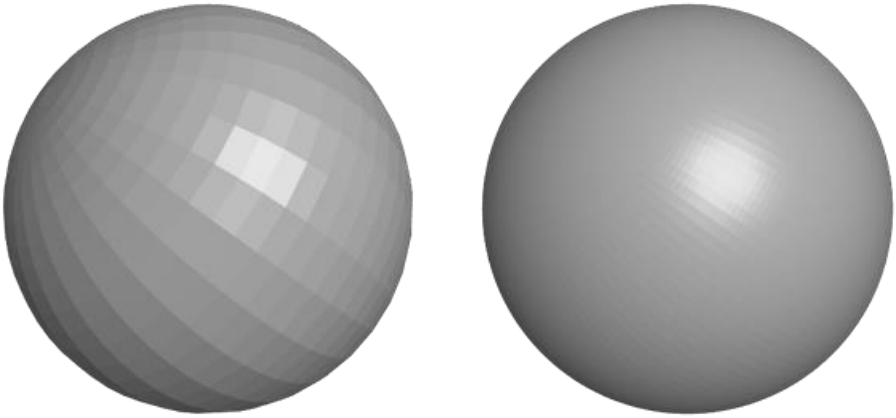
Наступним кроком було введення джерел світла та моделі дифузного відбиття. Якщо джерело світла знаходилося на нескінченній відстані від сцени, то світло, відбите від будь-якого багатокутника в сцені, вважався пропорційним косинусу кута між його нормаллю та напрямком, у якому знаходилося джерело світла.

Це співвідношення, відоме як закон Ламберта [14], гарантує, що полігон буде найбільш яскравим тоді, коли він перпендикулярний до джерела світла, та все менш яскравим у міру того, як він від світла відвертається. Щоб полігон не ставав повністю темним, до інтенсивності кольору зазвичай додається фіксована кількість амбієнтного світла.

Плоске зафарбовування [27] – найбільш простий та ефективний спосіб задання кольору для об'єкта. Він визначає один колір для кожної грані. Його реалізація може бути різною, але основна ідея полягає в тому, що використовується лише одна нормаль поверхні для кожного полігону. Сам колір є однорідним (постійним) для кожного багатокутника (рис 1.7).

Рисунок 1.7 – Модель плоского зафарбовування

На жаль, плоске зафарбовування полігонів дає не реалістичні результати, як можна було б припустити – межі сусідніх багатокутників досить помітні. Навіть суттєве збільшення кількості трикутників не допомагає приховати різкі переходи між гранями, що видно на рисунку 1.8.



2000 трикутників 32000 трикутників

Рисунок 1.8 – Сфери, візуалізовані за методом плоского зафарбовування

Гуро подолав «смуги Маха», обчислюючи модель освітлення у вершинах полігональної сітки, а потім інтерполюючи отримані кольори для кожного пікселя полігону (рис 1.9).

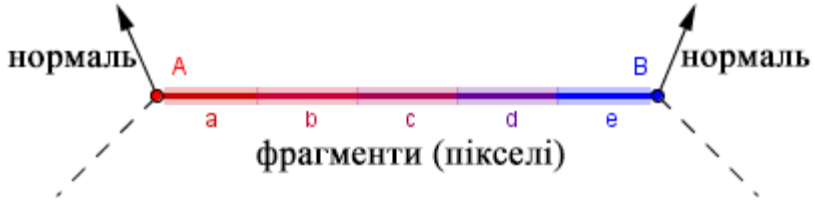
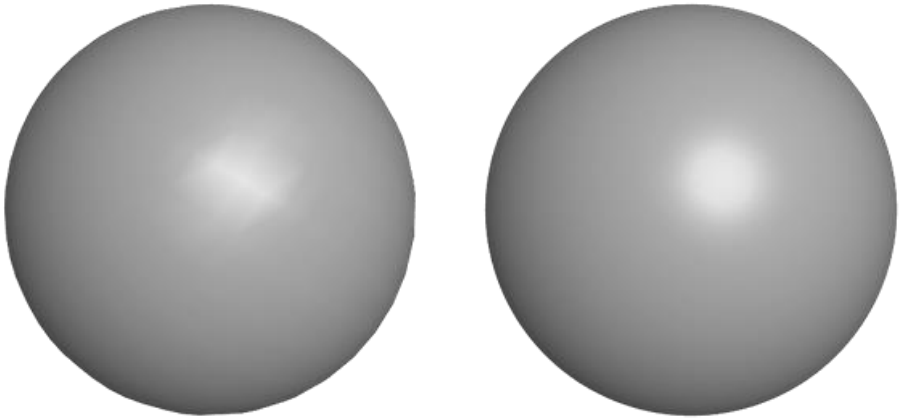


Рисунок 1.9 – Модель зафарбовування Гуро

Основна ідея полягає в тому, що для кожної вершини існує своя нормаль, а колір розраховується у вершинному шейдері. Потім цей колір інтерполюється полігоном. Оскільки вершин менше, ніж фрагментів, то обчислення кольору кожної вершини та його інтерполяція більш ефективні, ніж обчислення кожного фрагмента.

Це дає безперервне зафарбовування усіх багатокутників, що призвело до появи більш реалістичним зображень об'єктів, як на рисунку 1.10.



2000 трикутників 32000 трикутників

Рисунок 1.10 – Сфери, візуалізовані з використанням методу Гуро

Найбільша перевага затінення по Гуро була при візуалізації полігональних апроксимацій криволінійних поверхонь, наприклад сфери. Нормалі поверхні до точок, на які опирається модель освітлення, можна замінити на псевдонормалі, взяті з вихідної криволінійної поверхні.

Оскільки сусідні точки будуть мати ту саму псевдонормаль, отримане зображення створить враження гладкої поверхні. Хоча вони виглядали гладкими, проте достатньо подивитись на контур об’єкта, щоб побачити багатокутники. Також цей підхід погано справляється із матеріалами, що мають спекулярне відбиття (відблиск). Якщо відблиск знаходиться всередині полігону, а не на будь-якій з вершин, то зафарбовування не покаже його, або неправильно його зобразить, що видно на рисунку 1.11.

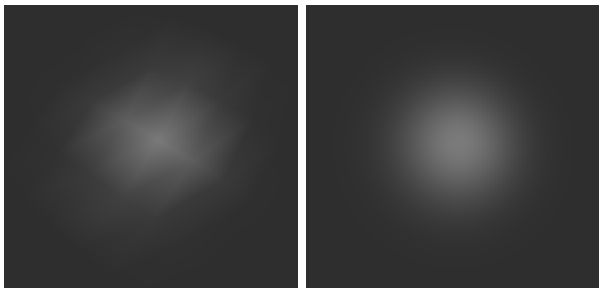


Рисунок 1.11 – Порівняння відблисків, отриманих при візуалізації моделі з малою та великою кількістю трикутників

Для покращення зафарбовування з врахуванням дзеркального (спекулярного) відображення було створено метод зафарбовування Фонга [17]. Основна ідея в тому, що нормаль від вершин інтерполюється. Колір розраховується для кожного фрагмента (пікселя) з урахуванням інтерпольованої нормалі (рис. 1.12).

Для апроксимації сфери це ідеальний варіант, так як інтерпольовані нормалі будуть такими ж, як у ідеальної сфери. Оскільки колір розраховується на основі нормалей, він буде розрахований так, ніби це була ідеальна сфера.

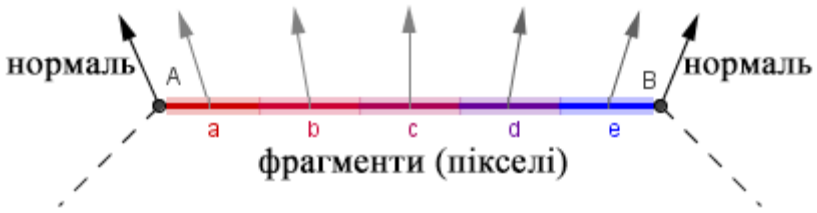
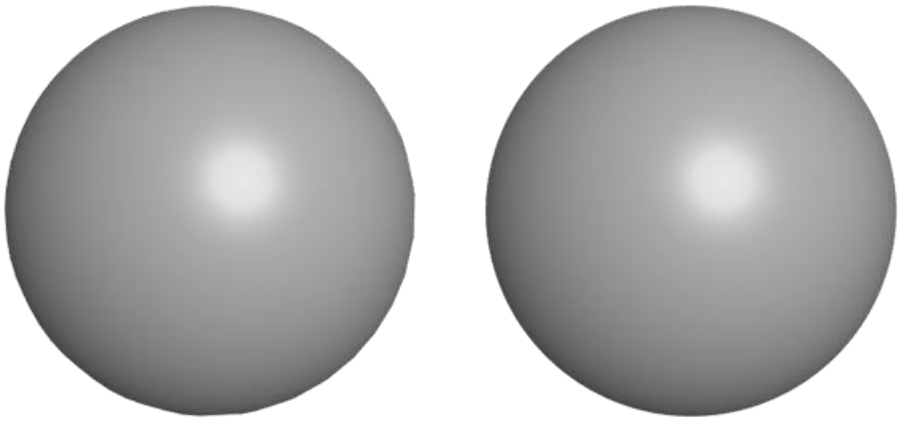


Рисунок 1.12 – Модель зафарбовування Фонга

Навіть при використанні моделей з малою кількістю полігонів результуюча візуалізація виглядає досить реалістично (рис. 1.13).



2000 трикутників 32000 трикутників

Рисунок 1.13 – Сфери, візуалізовані з використанням методу зафарбовування Фонга

Колір кінцевого зображення складається з 3 компонентів – амбієнтного, дифузного та спекулярного. Для знаходження значень всіх цих компонентів необхідні нормалі поверхні [28]. Вплив кожного компонента можна побачити на рисунку 1.14.

Потрібно обчислити три компоненти, які сформують кінцевий колір поверхні, на яку потрапляє джерело світла. Колір формується шляхом додавання значень цих компонентів разом.

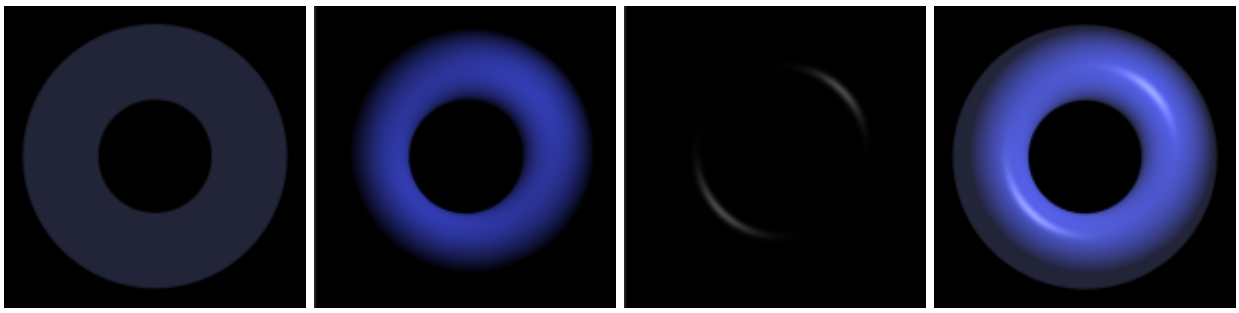


Рисунок 1.14– Результуюче зображення як сума амбієнтної, дифузної та спекулярної складових

Ці компоненти є векторами, тому кінцевий колір можна отримати за допомогою векторного додавання. Якщо сцена потребує альфа-каналу, остаточний колір можна перетворити на 4-вимірний вектор, додавши значення альфа-каналу як останній компонент. Значення 1 є повністю непрозорою поверхнею, а значення нижче цього показника – відсоток прозорості поверхні.

Деякі формати текстур автоматично надають значення альфа-каналу. У цих випадках значення прозорості може бути отримано з альфа-каналу текстури. Крім того, матеріал поверхні може мати значення непрозорості. Остаточне значення непрозорості можна розрахувати, помноживши непрозорість матеріалу на альфа-канал текстури.

Метод зафарбовування Фонга є однією з найпростіших систем відображення в сучасній комп'ютерній графіці. Ця система достатньо легка, щоб її можна було використовувати під час рендерингу в реальному часі. Вона не дає повністю реалістичного освітлення (результат майже завжди виглядає

«пластиковим»), але все ж таки досить хороша для більшості випадків і проста для розуміння.

Аналіз показав, що процес візуалізації тривимірних графічних схем є трудомістким процесом. Для отримання кожної точки результуючого зображення потрібно розрахувати її позицію на екрані та колір. Залежно від складності моделі або сцени та внутрішньої реалізації процесу візуалізації це може потребувати значних обчислювальних ресурсів.

Якість кінцевого результату рендерингу напряму залежить від використаної моделі освітлення та зафарбовування. Найбільш поширеною моделлю освітлення є модель Блінна-Фонга, а найбільш поширеними методами зафарбовування є метод Гуро та метод Фонга.

Для підвищення продуктивності процесу візуалізації потрібно виконувати оптимізацію самого процесу, оптимізацію моделей та сцени, а також використовувати апаратні прискорювачі та паралельні обчислення.

2 РОЗРОБКА МЕТОДІВ ВИЗНАЧЕННЯ ІНТЕНСИВНОСТІ КОЛЬОРУ З ВИКОРИСТАННЯМ ПОЛІНОМІВ

2.1 Модифікований метод зафарбовування з використанням поверхні другого порядку для визначення інтенсивності кольору

Підвищення реалістичності відтворення графічних сцен передбачає не тільки збільшення рівня деталізації поверхонь об'єктів реального світу але й використання більш складних моделей освітлення. Це гостро ставить питання про підвищення продуктивності графічних систем, особливо при формуванні динамічних зображень у реальному часі та в інтерактивному режимі, коли передбачається, що траєкторії руху об'єктів не задано заздалегідь, а визначаються діями користувача в процесі взаємодії із системою.

До високопродуктивних методів зафарбовування відносять метод зафарбовування з використанням поверхні другого порядку для визначення інтенсивностей кольору. Цей метод передбачає розрахунок нормалізованих векторів не для всіх, а тільки для декількох точок поверхні.

Етап зафарбовування [29] – це етап кінцевої візуалізації, на якому згідно з даними, отриманими на етапі геометричних перетворень, формуються пікселі зображення, для яких визначаються екранні координати та інтенсивності кольору. Цей етап вважається найбільш трудомістким у графічному конвеєрі, оскільки пов'язаний із попіксельними діями, складними обчисленнями та передбачає врахування багатьох параметрів. Рендеринг у загальному обсязі обчислень із формування тривимірних сцен складає 60-80 % [1], тому в значній мірі визначає продуктивність графічних систем.

Підвищення реалістичності відтворення графічних сцен передбачає не тільки збільшення рівня деталізації поверхонь для коректної апроксимації об'єктів реального світу але й використання більш складних моделей освітлення. Це гостро ставить питання про підвищення продуктивності графічних систем, особливо при формуванні динамічних зображень у реальному часі та в

інтерактивному режимі, коли передбачається, що траєкторії руху об'єктів не задано заздалегідь, а визначаються діями користувача в процесі взаємодії із системою. Недостатня продуктивність графічних систем є також завадою до моделювання в сценах фізичних процесів і збільшення кількості динамічних об’єктів.

Оскільки традиційні методи, засоби та підходи не задовольняють вимогам по продуктивності для багатьох галузей застосування тривимірної комп’ютерної графіки, то існує важлива науково-прикладна задача – розробка теоретичних основ високопродуктивного рендерингу тривимірних графічних зображень.

У роботі розглянуто питання використання поверхні другого прядку для високопродуктивного зафарбовування поверхонь тривимірних графічних сцен.

Методи, які використовуються для моделювання освітлення, оперують із розсіяним і відбитим світлом [14]. Відбите світло в своєму складі має дві компоненти: дифузну та спекулярну [24]. Розсіяне (фонове) світло – це світло, яке відбивається від навколишнього середовища. Точне моделювання даної складової світла вимагає великих обчислювальних витрат, а тому воно при використанні локальних моделей освітлення апроксимується [30].

Для врахування фонового освітлення можна припустити, що всі поверхні повністю освітлюються розсіяним світлом I_a. При цьому фоновий вклад у дифузне відбиття в будь-якій точці поверхні буде дорівнювати $I_{am} = I_a \cdot k_a$, де k_a – коефіцієнт відбиття розсіяного світла ($0 \le k_a \le 1$) [1].

Дифузне відбиття характерне для матових і шорстких поверхонь із хаотичними нерівностями, розміри яких співставні з довжиною хвилі або перевищують її [17]. Дифузне відбиття визначає видимість навколишніх тіл, оскільки кожна точка освітленої поверхні випромінює відбиті промені в усі сторони.

При наявності в сцені точкового джерела світла [14] інтенсивність дифузного відбиття пропорційна косинусу кута між нормаллю до поверхні й

напрямком на джерело світла $\vec{L}$. У цьому випадку для обчислення інтенсивності дифузного відбиття застосовують закон косинусів Ламберта [14]

$$I_d = I_0 \cdot k_d \cdot \cos\varphi,$$

де I_0 – інтенсивність джерела світла, $k_d \in [0,1]$ – коефіцієнт дифузного відбиття, φ – кут між вектором $\vec{L}$ і нормаллю $\vec{N}$ до поверхні (рис. 2.1).

На рисунку 2.1 відображено також вектор $\vec{V}$ до спостерігача, серединний вектор.

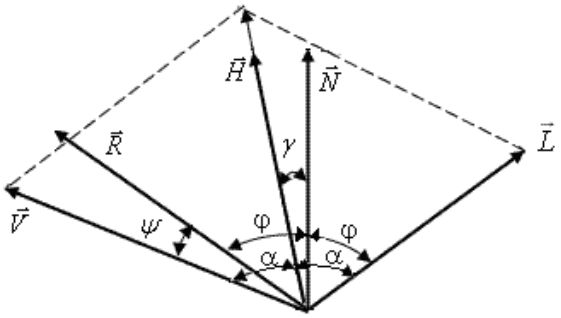


Рисунок 2.1 – Вектори нормалей до точки поверхні

Історично першим методом зафарбовування був метод однотонного зафарбовування [27], згідно з яким для кожного плоского трикутника визначався вектор нормалі, а на його основі – колір. Складові трикутники об’єкта заповнювалися одним кольором без його градації, що призводило до різкої зміни інтенсивності кольору на межах трикутників. Однотонне зафарбовування вимагає найменших обчислювальних затрат. Воно хоча й має низьку якість, але внесено до функцій Direct3D і підтримується тривимірними акселераторами.

До найпоширеніших методів зафарбовування відносять метод Гуро [13], який забезпечує прийнятний компроміс між швидкодією формування тривимірних зображень та їх якістю. Процес зафарбовування має такі стадії [18]: а) розраховують вектори нормалей до кожної грані; б) шляхом

усереднення нормалей усіх граней, яким належить вершина, розраховують нормалі у вершинах трикутника (багатокутника); с) визначають інтенсивності кольору у вершинах багатокутника, використовуючи значення нормалей; д) зафарбовують ділянку, обмежену багатокутником, шляхом лінійної інтерполяції інтенсивностей кольорів вздовж ребер, а потім і між ребрами вздовж кожного рядка растеризації. Останнім часом затінення за Гуро часто використовують як проміжну стадію з інтерактивного формування 3D-зображення, покладаючи побудову повноцінної сцени на етап фінального рендерингу.

До недоліків методу Гуро можна віднести [20]: а) метод використовує для визначення інтенсивностей кольору лінійну інтерполяцію, в той час як дифузна та спекулярна складові кольору мають нелінійний характер зміни; б) не враховується локальна кривизна поверхні, оскільки вектори нормалей визначаються тільки для вершин трикутника; в) відблиски відтворюються тільки в разі, якщо вершини трикутників знаходяться в їх зоні (відблиск, який не має спільних точок із вершинами трикутників, або розташований усередині трикутника, не буде сформовано); г) на межах двох трикутників проявляються смуги Маха [12], які пов’язані з літеральним гальмуванням на сітківці ока; д) має місце зміна інтенсивності кольору зображення від кадру до кадру, що виражається в миготінні, особливо відблисків, оскільки при формуванні динамічних зображень змінюється структура та положення вузлів тріангуляційної мережі; є) наявність артефакту типу “зірка”, який полягає в тому, що відблиск, який повинен мати форму еліпса, має форму зірки. Це пояснюється тим, що ділянки відблиску формуються в різних трикутниках і проявляється ефект смуг Маха; ж) метод не враховує перспективу об’єкта.

При зафарбовуванні за методом Фонга інтенсивність кольору точок визначають за формулою [14]:

$$I = I_a k_a + I_l(k_d \cos\psi + k_s \cos^n \lambda) \qquad (2.1)$$

де I_a, I_l – інтенсивності відповідно розсіяного і направленого джерел світла, k_a, k_d, k_s – коефіцієнти розсіяного, дифузійного і дзеркального світла, ψ – кут між напрямком світла і вектором нормалі, λ – кут між відбитим напрямком світла і спостерігачем, n – коефіцієнт яскравості поверхні, $\cos^n \lambda$ – дистрибутивна функція (BRDF) [1], яка відповідає за оптичні характеристики поверхні.

У більшості випадків при реалізації зафарбовування за Фонгом використовуються моделі освітлення Бліна та Фонга [21].

При використанні моделі освітлення Фонга (рис. 2.2) у формулі (2.1) $\cos\lambda = \vec{R} \cdot \vec{V} = 2(\vec{L} \cdot \vec{V})(\vec{N} \cdot \vec{V}) - \vec{V} \cdot \vec{L}$. Для моделі освітлення Бліна $\cos\lambda$ заміняють на $\cos\gamma = \vec{H} \cdot \vec{L}$, де $\vec{H} = (\vec{L} + \vec{V}) / \left|\vec{L} + \vec{V}\right|$.

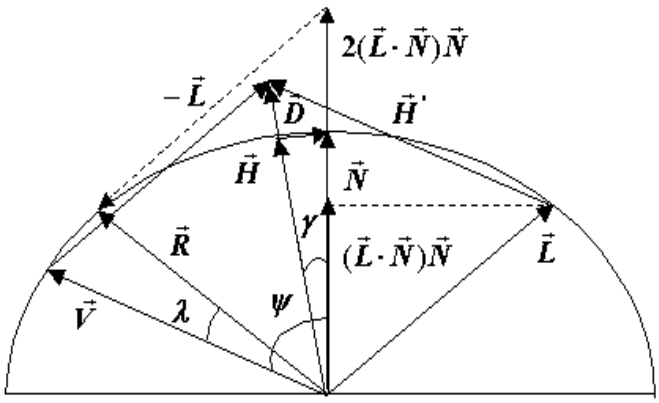


Рисунок 2.2 – Визначення кутів для моделей освітлення Фонга і Бліна

Модель освітлення Фонга вважається точнішою, однак потребує більшого обсягу обчислень. Продуктивність зафарбовування при використанні моделі Бліна значно вища порівняно з моделлю Фонга. Це пояснюється тим, що у випадку, коли джерело світла та спостерігач розташовані на нескінченній відстані від об’єкта (найпоширеніший випадок у комп’ютерній графіці), значення вектора $\vec{H}$ для моделі освітлення Бліна розраховується один раз для кожного кадру зображення. При використанні ж моделі освітлення Фонга для

кожної точки поверхні знаходять вектор $\vec{R}$. У подальшому будемо використовувати модель освітлення Бліна як таку, що має меншу обчислювальну складність.

При зафарбовуванні 3D-об’єктів за методом Фонга визначають нормовані вектори до поверхні об’єкта, джерела світла й спостерігача, а також допоміжні вектори залежно від вибору моделі освітлення.

Нормалізація вектора [28] потребує виконання трьох операцій ділення, трьох операцій множення, двох операцій додавання та операцію знаходження квадратного кореня, виконання яких достатньо трудомістке. У зв’язку з цим актуальним питанням є зменшення при зафарбовуванні кількості використаних нормалізованих векторів.

Один з таких методів зафарбовування передбачає використання для визначення інтенсивності кольору поверхні другого порядку, для якої нормалізовані вектори визначаються тільки в деяких точках. Інтенсивність кольору в точці на поверхні другого порядку задано поліномом [19]

$$I(x, y) = A \cdot x^2 + B \cdot y^2 + C \cdot x \cdot y + D \cdot x + E \cdot y + F.$$

Із рівняння видно, що необхідно визначити шість невідомих A, B, C, D, E, F. Це передбачає формування й розв’язання системи із шести рівнянь. Оскільки для трикутника задаються інтенсивності кольору тільки в трьох його вершинах, то найпростіше довизначити інтенсивності кольору в середніх точках на ребрах трикутника, що дасть можливість скласти систему із шести рівнянь.

Знаходження невідомих A, B, C, D, E, F. наведено в [26]. У подальшому метод, який використовує для визначення інтенсивності кольору наведену формулу, будемо називати «прямим».

Розглянемо рядок растеризації (рис. 2.3), паралельний осі абсцис.

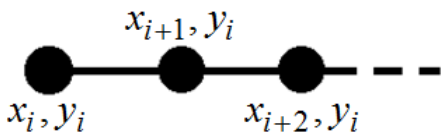


Рисунок 2.3 – Рядок растеризації, паралельний осі абсцис

Для поточної точки (x_i, y_i) інтенсивність кольору визначаємо за формулою

$$I(x_i, y_i) = Ax_i^2 + By_i^2 + Cx_i y_i + Dx_i + Ey_i + F. \quad (2.2)$$

Виконаємо крокове переміщення по осі x і знайдемо інтенсивність кольору в точці (x_{i+1}, y_i)

$$\begin{aligned} I(x_{i+1}, y_i) &= A(x_i + 1)^2 + By_i^2 + C(x_i + 1)y_i + D(x_i + 1) + Ey_i + F = \\ &= Ax_i^2 + 2Ax_i + A + By_i^2 + Cx_i y_i + Cy_i + Dx_i + D + Ey_i + F = \\ &= I(x_i, y_i) + 2Ax_i + A + Cy_i + D. \end{aligned} \quad (2.3)$$

Для рядка растеризації значення Cy_i не змінюється, тому його доцільно обчислювати тільки один раз і в подальшому використовувати як операнд. При зміщенні на дві точки (x_{i+2}, y_i), інтенсивність кольору дорівнює

$$\begin{aligned} I(x_{i+2}, y_i) &= A(x_i + 2)^2 + By_i^2 + C(x_i + 2)y_i + D(x_i + 2) + Ey_i + F = \\ &= Ax_i^2 + 4Ax_i + 4A + By_i^2 + Cx_i y_i + 2Cy_i + Dx_i + 2D + Ey_i + F = \\ &= I(x_i, y_i) + 4Ax_i + 4A + 2Cy_i + 2D. \end{aligned} \quad (2.4)$$

З формул (2.3) і (2.4) видно, що інтенсивність кольору в поточній точці можна визначити через інтенсивності кольору в двох попередніх точках:

$$I(x_{i+2}, y_i) = I(x_i, y_i) + 4Ax_i + 4A + 2Cy_i + 2D =$$
$$= 2[I(x_i, y_i) + 2Ax_i + A + Cy_i + D] - I(x_i, y_i) + 2A =$$
$$= 2I(x_{i+1}, y_i) - I(x_i, y_i) + 2A. \quad (2.5)$$

На рисунку 2.4 наведено структурну схему для визначення $I(x_{i+2}, y_i)$.

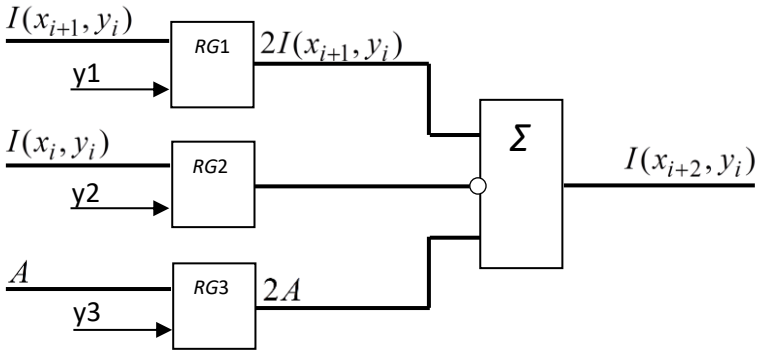


Рисунок 2.4 – Структурна схема блоку для визначення $I(x_{i+2}, y_i)$

Розрахунок за формулою (2.5) для визначення інтенсивності кольору в точці (x_{i+2}, y_i) значно простіший порівняно з формулою (2.2), оскільки передбачається використання тільки двох мікрооперацій зсуву та двох мікрооперацій додавання. При цьому не використовується «довга» операція множення.

Знайдемо значення інтенсивності кольору в точці (x_{i+1}, y_i) через інтенсивності кольорів в двох найближчих сусідніх точках. Для цього використаємо формулу (2.5).

$$I(x_{i+1}, y_i) = \frac{I(x_i, y_i) + I(x_{i+2}, y_i) - 2A}{2}. \quad (2.6)$$

Порівняння формул (2.5) і (2.6) показує, що їхня обчислювальна складність однакова.

При зміщенні на три точки, інтенсивність кольору в точці (x_{i+3}, y_i) дорівнює

$$I(x_{i+3}, y_i) = A(x_i + 3)^2 + By_i^2 + C(x_i + 3)y_i + D(x_i + 3) + Ey_i + F =$$
$$= Ax_i^2 + 6Ax_i + 7A + By_i^2 + Cx_i y_i + 3Cy_i + Dx_i + 3D + Ey_i + F =$$
$$= I(x_i, y_i) + 6Ax_i + 9A + 3Cy_i + 3D.$$

На жаль, в останній формулі необхідно використовувати операції множення, що, безумовно, впливає на продуктивність рендерингу.

Інтенсивність кольору в точці можна виразити через інтенсивність кольору в трьох попередніх точках згідно виразу

$$I(x_{i+3}, y_i) = I(x_i, y_i) + 6Ax_i + 9A + 3Cy_i + 3D =$$
$$= [I(x_i, y_i) + 4Ax_i + 4A + 2Cy_i + 2D] +$$
$$+ [I(x_i, y_i) + 2Ax_i + A + Cy_i + D] - I(x_i, y_i) + 4A =$$
$$= I(x_{i+2}, y_i) + I(x_{i+1}, y_i) - I(x_i, y_i) + 4A. \qquad (2.7)$$

Визначимо інтенсивність кольору $I(x_{i+3}, y_i)$ виключно через значення інтенсивностей кольорів. Для цього знайдемо значення $4A$ через інтенсивність кольору в трьох сусідніх точках, використавши формулу (2.5).

$$4A = 2 \cdot 2A = 2[I(x_{i+2}, y_i) - 2I(x_{i+1}, y_i) + I(x_i, y_i)] =$$
$$= 2I(x_{i+2}, y_i) - 4I(x_{i+1}, y_i) + 2I(x_i, y_i).$$

Підставивши отримане значення у формулу (2.7) знайдемо, що

$$I(x_{i+3}, y_i) = I(x_{i+2}, y_i) + I(x_{i+1}, y_i) - I(x_i, y_i) +$$
$$+ 2I(x_{i+2}, y_i) - 4I(x_{i+1}, y_i) + 2I(x_i, y_i) =$$
$$= 3I(x_{i+2}, y_i) - 3I(x_{i+1}, y_i) + I(x_i, y_i). \qquad (2.8)$$

Множення на 3 можна замінити на операції додавання та зсуву. Наприклад, $3I(x_{i+2}, y_i) = 2I(x_{i+2}, y_i) + I(x_{i+2}, y_i)$.

Тоді $I(x_{i+3}, y_i) = 2I(x_{i+2}, y_i) + I(x_{i+2}, y_i) - 2I(x_{i+1}, y_i) - I(x_{i+1}, y_i) + I(x_i, y_i)$.

У наведеній формулі використовуються виключно операції додавання та зсуву. На рисунку 2.5 наведено структурну схему блоку для визначення інтенсивності кольору $I(x_{i+3}, y_i)$

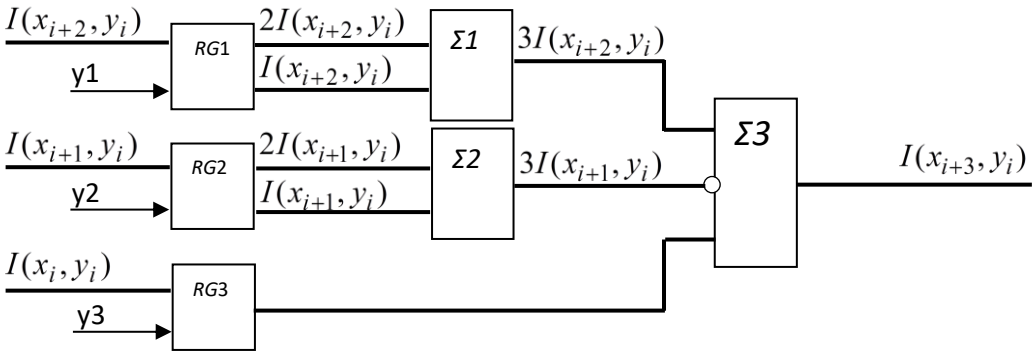


Рисунок 2.5 – Структурна схема блоку для визначення $I(x_{i+3}, y_i)$

У роботі [30] запропоновано розбивати рядок растеризація на цифрові сегменти довжиною 2^k і визначати інтенсивності кольору в його кінцевих точках. Значення інтенсивності кольору в проміжних точках сегменту в подальшому визначають з використанням кодової лінійної інтерполяції.

Розглянемо визначення інтенсивності кольору в кінцевих точках цифрового сегменту довжиною 2^k

$$I(x_{i+2^k}, y_i) = A(x_i + 2^k)^2 + By_i^2 + C(x_i + 2^k)y_i + D(x_i + 2^k) + Ey_i + F =$$
$$= Ax_i^2 + 2 \cdot 2^k Ax_i + (2^k)^2 A + By_i^2 + Cx_i y_i + 2^k Cy_i + Dx_i + 2^k D + Ey_i + F =$$
$$= I(x_i, y_i) + 2^{k+1} Ax_i + 2^{2k} A + 2^k Cy_i + 2^k D.$$

Наведена формула має меншу обчислювальну складність порівняно з використанням формули (2.2) для визначення інтенсивностей кольору кінцевих точок. Дійсно, в цьому випадку використовується десять операцій множення та п‘ять операцій додавання. При використанні отриманої формули для визначення $I(x_{i+2^k}, y_i)$ використовується тільки дві операції множення, чотири операції додавання та $(k+1)+2k+k+k=5k+1$ операцій зсуву.

Порівняємо ресурсні затрати отриманих методів. Як критерій можна використати кількість операцій (множення, додавання/віднімання, зсув). Операція множення є однією з найбільш ресурснозатратних. Множення на числа типу 2^n заміняють зсувом на n розрядів.

Приведемо формулу (2.2) до вигляду

$$I(x_i, y_i) = x_i(Ax_i + Cy_i + D) + y_i(By_i + E) + F. \qquad (2.9)$$

Тоді для обчислення інтенсивності кольору в точці потрібно буде виконати п‘ять операцій множення та п‘ять операцій додавання. Використовуючи «прямий» метод для знаходження інтенсивності кольору в n точках потрібно виконати $5n$ операцій множення та $5n$ операцій додавання.

Незалежно від методу, інтенсивність кольору в першій точці рядка растеризації завжди доведеться знаходити «прямим» методом.

Проаналізуємо знаходження інтенсивності кольору з використанням формули (2.3). Обчислення інтенсивності кольору в кожній точці, починаючи з другої, потребує однієї операції множення, чотирьох операцій додавання та однієї операції зсуву. Тоді для знаходження інтенсивності кольору в n точках потрібно виконати $5+(n-1)=n+4$ операцій множення, $5+4(n-1)=4n+1$ операцій додавання, $n-1$ операцій зсуву.

Проаналізуємо обчислення інтенсивності кольору, використовуючи формулу (2.5). Для кожної точки, починаючи з третьої, знаходження інтенсивності кольору потребує двох операцій зсуву, однієї операції додавання

та однієї операції віднімання. Вважаючи, що для знаходження інтенсивності кольору в другій точці рядка растеризації використано попередній метод, для знаходження інтенсивності кольору в n точках потрібно виконати 5+1=6 операцій множення, $5+4+(n-2)=n+7$ операцій додавання та $n-2$ операцій віднімання (оскільки віднімання реалізується через додавання, то потрібно виконати $(n+7)+(n-2)=2n+5$ операцій додавання), $1+2(n-2)=2n-3$ операцій зсуву.

Якщо обчислювати інтенсивність кольору в точці, використовуючи формулу (2.7), то для кожної точки, починаючи з четвертої, потрібно виконати дві операції додавання та одну операцію віднімання (тобто три операції додавання), а також дві операції зсуву. Якщо інтенсивність кольору в першій точці рядка растеризації знаходити з формули (2.2), у другій точці – з формули (2.3), у третій точці – з формули (2.5), то для обчислення інтенсивності кольору в n точках потрібно виконати 6 операцій множення, $5+4+2+3(n-3)=3n+2$ операцій додавання та $1+2+2(n-3)=2n-3$ операцій зсуву.

Формула (2.8) у неоптимізованому вигляді містить дві операції множення, операцію додавання та операцію віднімання (дві операції додавання). У випадку, коли інтенсивність кольору в перших трьох точках знаходимо аналогічно до попереднього методу, для знаходження інтенсивності кольору в n точках потрібно виконати $5+1+2(n-3)=2n$ операцій множення, $5+4+2+2(n-3)=2n+5$ операцій додавання та 3 операції зсуву.

Оптимізована формула (2.8) містить дві операції додавання, дві операції віднімання (всього – чотири операції додавання), дві операції зсуву. У такому випадку, для визначення інтенсивності кольору в n точках потрібно виконати 6 операцій множення, $5+4+2+4(n-3)=4n-1$ операцій додавання та $1+2+2(n-3)=2n-3$ операцій зсуву.

При використанні процесора Intel Xeon Phi [31] операція множення потребує 8 тактів ЦП, операції додавання, віднімання та зсуву – по одному такту ЦП. Щоб отримати загальну кількість тактів ЦП для кожного методу було

прийнято $n=16$ та домножено отримані значення на «вартість» операції. Результати вимірювань показано в таблиці 2.1.

Таблиця 2.1 – Кількість тактів ЦП, потрібних для обчислення значень інтенсивності кольору в 16 точках

Тип формули	Множення, тактів	Додавання, тактів	Зсув, тактів	Всього
Формула (2.2)	1280	80	0	1360
Формула (2.9)	640	80	0	720
Формула (2.3)	160	65	15	240
Формула (2.5)	48	37	29	114
Формула (2.7)	48	50	29	127
Формула (2.8)	256	37	3	296
Оптимізована формула (2.8)	48	63	29	140

При використанні формули (2.3) досягається підвищення продуктивності на 67% порівняно з формулою (2.9), при використанні формули (2.5) – на 84%.

Отримано нові аналітичні залежності для визначення інтенсивності кольору точки рядка растеризації через інтенсивності кольору сусідніх точок. Формули не використовують довготривалі мікрооперації множення.

Результати проведених досліджень можуть бути використані у високопродуктивних системах тривимірної графіки.

2.2 Метод зафарбовування з використанням кубічного полінома

Методи з використанням лінійної інтерполяції прості для впровадження, не потребують значних обчислювальних ресурсів, проте вони втрачають ефективність зі збільшенням кривизни поверхні. Для досягнення потрібного

наближення до ідеальної кривої доводиться збільшувати кількість інтервалів на рядку растеризації. Це сильно зменшує переваги даного методу над іншими. Також, оскільки диференційовністю кривої нехтують, з'являються побічні ефекти на краях інтервалу.

Квадратична інтерполяція дозволяє позбутися деяких недоліків методів, побудованих на лінійній інтерполяції. Проте використання полінома другого порядку знижує точність розрахунків якщо парабола погано апроксимує ідеальну криву.

Спробуємо застосувати для знаходження інтенсивності кольору на конкретному проміжку рядка растеризації поліном третьої степені

$$I_i(p) = A_i p^3 + B_i p^2 + C_i p + D_i,$$

де p – параметрична змінна, A_i, B_i, C_i, D_i – коефіцієнти, обчислені з чотирьох значень інтенсивності кольору на інтервалі рядка растеризації. Підберемо чотири рівняння для знаходження чотирьох невідомих коефіцієнтів A_i, B_i, C_i, D_i.

Одразу помітно, що при $p = 0$ отримуємо $I_i(0) = D_i$. З іншого боку проміжку $p = 1$, тому $I_i(1) = A_i + B_i + C_i + D_i$. Далі використаємо дві точки, якими можна розділити навпіл інтервал та одну з отриманих половинок ($p = 1/2$ та $p = 1/4$). Позначимо $I_i(0) = I_1, I_i(1) = I_2, I_i(1/2) = I_3, I_i(1/4) = I_4$. Отримуємо систему рівнянь

$$\begin{cases} I_1 = D_i, \\ I_2 = A_i + B_i + C_i + D_i, \\ I_3 = \dfrac{A_i}{8} + \dfrac{B_i}{4} + \dfrac{C_i}{2} + D_i, \\ I_4 = \dfrac{A_i}{64} + \dfrac{B_i}{16} + \dfrac{C_i}{4} + D_i. \end{cases}$$

Після виконання всіх розрахунків знаходимо такі формули для знаходження коефіцієнтів:

$$\begin{cases} A_i = \dfrac{8\cdot(I_2+8\cdot I_4)}{3} - 8\cdot(I_1+2\cdot I_3), \\ B_i = 2\cdot(7\cdot I_1 - I_2 + 10\cdot I_3 - 16\cdot I_4), \\ C_i = \dfrac{I_2+32\cdot I_4}{3} - 7\cdot I_1 - 4\cdot I_3, \\ D_i = I_1. \end{cases}$$

При апаратній реалізації множники, які є степенями двійки, легко замінити операціями зсуву. Інші множники потрібно розглядати як суму двійкових розрядів, наприклад $10\cdot I = 8\cdot I + 2\cdot I$.

Ділення на 3 апроксимуємо таким чином:

$$\frac{1}{3} \approx \frac{1}{2} - \frac{1}{8} - \frac{1}{32} - \frac{1}{128}.$$

Відносна похибка частки з використанням такого наближення не повинна перевищити $0{,}79\%$.

Для розрахунку значень інтенсивності кольору потрібно застосувати прямий метод з використанням полінома третього порядку. Для спрощення процесу та підвищення ефективності порядок виконання операцій зазвичай модифікують так:

$$I_i(p) = A_i p^3 + B_i p^2 + C_i p + D_i = p\cdot(p\cdot(p\cdot A_i + B_i) + C_i) + D_i.$$

На рисунку наведено функціональну схему апаратної реалізації блоку для обчислення інтенсивності кольору як значення полінома третього ступеня.

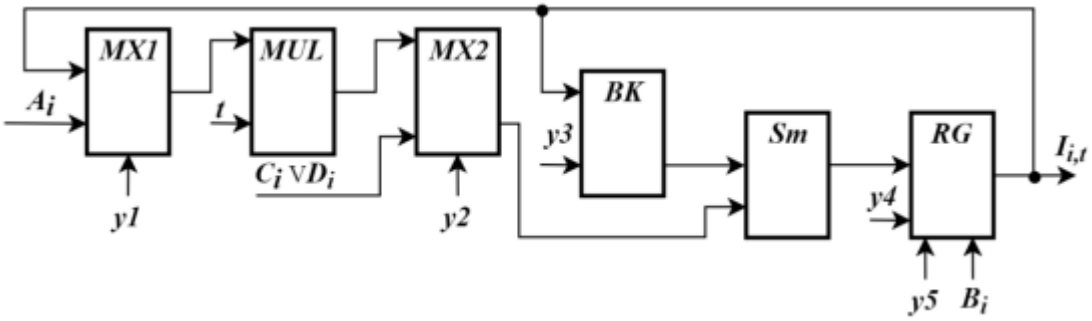


Рисунок 2.6 – Функціональна схема блоку для знаходження інтенсивності кольору з використанням кубічного полінома

Значення B_i записується в регістр RG. Значення A_i передається на вхід мультиплексора $MX1$, звідки воно надходить до перемножувача MUL, де перемножується зі значенням t. Наступний крок – отримання суми $A_i \cdot t + B_i$. Для цього використовується блок суматора Sm, на один з входів якого передається значення B_i за допомогою блоку BK. Даний блок виконує дві функції: передачу значень та їхнє обнулення. Отримане значення відправляється до регістра RG для наступного кроку, виконується обнулення.

Далі виконується множення $(A_i \cdot t + B_i)$ на t, отримане значення проходить через суматор Sm як сума $(A_i \cdot t + B_i) \cdot t + 0$ та записується в регістр RG. Блок BK переходить в стан передачі значень, мультиплексор $MX2$ отримує значення C_i, суматор Sm формує суму $(A_i \cdot t + B_i) \cdot t + C_i$, результат записується до регістра RG. Блок BK виконує обнулення, обчислюється добуток $((A_i \cdot t + B_i) \cdot t + C_i) \cdot t$. Останній етап – обчислення суми $((A_i \cdot t + B_i) \cdot t + C_i) \cdot t + D_i$: значення D_i подається на вхід мультиплексора $MX2$ та додається до значення $((A_i \cdot t + B_i) \cdot t + C_i) \cdot t$ у суматорі Sm. Після виконання усіх потрібних ітерацій робота блоку завершується виведенням кінцевого значення $I_i(t) = ((A_i \cdot t + B_i) \cdot t + C_i) \cdot t + D_i$.

3 РОЗРОБКА МЕТОДІВ ПРИСКОРЕНОГО ОБЧИСЛЕННЯ ІНТЕНСИВНОСТІ КОЛЬОРУ ТОЧОК ПОВЕРХОНЬ ТРИВИМІРНИХ ОБ'ЄКТІВ

3.1 Прискорене обчислення дифузної та спекулярної складових кольору

У даному підрозділі запропоновано новий метод прискореного визначення дифузної та спекулярної складових кольору при зафарбовуванні тривимірних графічних фігур, особливість якого полягає в одночасному й незалежному апаратному визначенні інтенсивностей кольору одразу двох точок РРТ.

Нехай у кінцевих точках a,b рядка растеризації трикутника задано відповідно вектори нормалей $\vec{N}_a$ і $\vec{N}_b$. Розглянемо найбільш поширений при формуванні зображення випадок [32], коли вектор напрямку джерела світла $\vec{L}$ є сталим для трикутника. Дифузну складову кольору знайдемо за виразом

$$I_d = k_d \cdot I \cdot \frac{\vec{N}_x \cdot \vec{L}}{\left|\vec{N}_x\right|},$$

де

$$\vec{N}_x = \vec{N}_a + \frac{\vec{N}_b - \vec{N}_a}{m} \cdot x,$$

а m – довжина рядка растеризації трикутника, x – позиція пікселя в рядку растеризації, k_d – коефіцієнт дифузного відбиття, I – інтенсивність джерела світла; також

$$\left|N_x\right| = \sqrt{\left(\vec{N}_a + \frac{\vec{N}_b - \vec{N}_a}{m} \cdot x\right)^2} = \sqrt{\left(\vec{N}_a\right)^2 + 2 \cdot \left(\frac{\vec{N}_b - \vec{N}_a}{m} \vec{N}_a\right) \cdot x + \left(\frac{\vec{N}_b - \vec{N}_a}{m}\right)^2 \cdot x^2} .$$

Оскільки вектори нормалей на ребрах трикутника нормалізовані, то $\left(\vec{N}_a\right)^2 = 1$. Нехай

$$s = 2 \cdot \frac{\vec{N}_b - \vec{N}_a}{m} \cdot \vec{N}_a,$$

$$c = \left(\frac{\vec{N}_a - \vec{N}_b}{m}\right)^2.$$

За умови таких позначень $\left|\vec{N}_x\right| = \sqrt{cx^2 + sx + 1}$. Знайдемо

$$\vec{N}_x \cdot \vec{L} = \left(\vec{N}_a + \frac{\vec{N}_b - \vec{N}_a}{m} \cdot x\right) \cdot \vec{L} = \vec{N}_a \cdot \vec{L} + \left(\frac{\vec{N}_b - \vec{N}_a}{m} \cdot \vec{L}\right) \cdot x .$$

Позначимо $b = \vec{N}_a \cdot \vec{L}$, $g = \left(\vec{N}_b - \vec{N}_a\right) \cdot \vec{L} / m$. Тоді $\vec{N}_x \cdot \vec{L} = g \cdot x + b$.

З урахуванням позначень

$$\frac{\vec{N}_x \cdot \vec{L}}{\left|\vec{N}_x\right|} = \frac{g \cdot x + b}{\sqrt{c \cdot x^2 + s \cdot x + 1}} = \frac{p}{\sqrt{q}}. \qquad (3.1)$$

В останньому виразі g, b, c, s – скалярні величини.

За умови, що виконано переміщення на один крок уздовж РРТ

$$p_{x+1} = g \cdot (x+1) + b = (g \cdot x + b) + g = p_x + g .$$

Нехай $M_x = c \cdot x^2 + s \cdot x$. Тоді

$$M_{x+1} = c \cdot (x+1)^2 + s \cdot (x+1) = c \cdot x^2 + s \cdot x + 2 \cdot c \cdot x + (c+s) = M_x + 2 \cdot c \cdot x + (c+s).$$

Доведемо, що $q(a) = q \cdot (m-a)$, де a – позиція пікселя з першої половини РРТ.

$$q(a) = \frac{\left(\vec{N}_b - \vec{N}_a\right)^2}{m^2} \cdot a^2 + \frac{2 \cdot \left(\vec{N}_b - \vec{N}_a\right) \cdot \vec{N}_a}{m} \cdot a + 1 = \\ = \frac{\left(\vec{N}_b\right)^2 - 2 \cdot \vec{N}_a \cdot \vec{N}_b + \left(\vec{N}_a\right)^2}{m^2} \cdot a^2 + \frac{2 \cdot \left(\vec{N}_a \cdot \vec{N}_b - \left(\vec{N}_a\right)^2\right)}{m} \cdot a + 1. \quad (3.2)$$

Якщо $x = m - a$, то

$$q(m-a) = \frac{\left(\vec{N}_b - \vec{N}_a\right)^2}{m^2} \cdot (m-a)^2 + \frac{2 \cdot \left(\vec{N}_b - \vec{N}_a\right) \cdot \vec{N}_a}{m} \cdot (m-a) + 1 = \\ = \frac{\left(\vec{N}_b\right)^2 - 2 \cdot \vec{N}_a \cdot \vec{N}_b + \left(\vec{N}_a\right)^2}{m^2} \cdot a^2 + \frac{2 \cdot \left(\vec{N}_a \cdot \vec{N}_b - \left(\vec{N}_a\right)^2\right)}{m} \cdot a + \left(\vec{N}_b\right)^2 - \left(\vec{N}_a\right)^2 + 1. \cdot \quad (3.3)$$

Ураховуючи, що $\left(\vec{N}_b\right)^2 - \left(\vec{N}_a\right)^2 = 0$, приходимо до висновку, що вирази (3.2) і (3.3) ідентичні, тобто $q(a) = q(m-a)$. Це означає, що у виразі (3.1) достатньо розрахувати $\sqrt{q}$ тільки для першої половини рядка растеризації трикутника та використати отримані результати для його другої половини.

Розглянемо питання апроксимації $1/\sqrt{q}$. Для цього дослідимо інтервал зміни $cx^2 + sx$

$$c = \left(\frac{\vec{N}_b - \vec{N}_a}{m} \right)^2 = \frac{\vec{N}_a^2 + \vec{N}_b^2 - 2 \cdot \vec{N}_a \cdot \vec{N}_b}{m^2} = \frac{2 - 2 \cdot \vec{N}_a \cdot \vec{N}_b}{m^2},.$$

$$s = 2 \cdot \frac{\vec{N}_b - \vec{N}_a}{m} \cdot \vec{N}_a = \frac{2 \cdot \vec{N}_a \cdot \vec{N}_b - 2 \cdot \vec{N}_a^{\,2}}{m} = \frac{2 \cdot \vec{N}_a \cdot \vec{N}_b - 2}{m}.$$

$\vec{N}_a \cdot \vec{N}_b$ є скалярним добутком двох нормалізованих векторів нормалей, а тому визначає значення косинуса кута w між ними. Оскільки $\vec{N}_a \cdot \vec{N}_b = \cos w$, то

$$M = cx^2 + sx = \frac{2 - 2 \cdot \cos w}{m^2} \cdot x^2 + \frac{2 \cdot \cos w - 2}{m} \cdot x.$$

Узявши похідну за x від останнього виразу й прирівнявши її до нуля, знаходимо, що екстремум функції буде мати місце при $x = m/2$ і дорівнюватиме $(\cos w - 1)/2$. Тріангуляцію поверхонь, як правило, виконують за умови, що $0 \le \cos w \le 1$[27], а це відповідає нерівності $-0{,}5 \le M \le 0$.

Вираз $1/\sqrt{M+1}$ можна апроксимувати рядом Тейлора. Дослідження [33] показали, що при цьому на інтервалі $-0{,}5 \le M \le 0$ максимальна похибка апроксимації складає 4,98 %.

Максимальну похибку апроксимації виразу $1/\sqrt{M+1}$ можна зменшити майже в 14 разів, використавши поліном Чебишова того ж степеня, що й для ряду Тейлора. При використанні ряду Чебишова для апроксимації функції $1/\sqrt{M+1}$ на інтервалі $-0{,}5 \le M \le 0$ отримано таку функцію

$$\frac{1}{\sqrt{M+1}} \approx 1{,}003 - 0{,}403 \cdot M + 0{,}82 \cdot M^2.$$

Максимальна відносна похибка апроксимації при використані останнього виразу не перевищує 0,357 %. Із метою спрощення апаратної реалізації можна скористатися такою апроксимацією:

$$\frac{1}{\sqrt{M+1}} \approx 1 - \left(\frac{1}{4} + \frac{1}{8} + \frac{1}{32}\right) \cdot M + \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{16}\right) \cdot M^2.$$

У цьому випадку похибка апроксимації зросте всього до 0,51 %. Якщо вираз у перших дужках позначити через l, то можна записати, що

$$\frac{1}{\sqrt{M+1}} \approx 1 - l \cdot M + 2 \cdot l \cdot M^2.$$

Чисельник виразу (3.1) записано для випадку растеризації рядка трикутника зліва направо. При зворотному напрямку растеризації для чисельника виразу (3.1) справедливий такий запис

$$\vec{L} \cdot \left(\vec{N}_b - \frac{\vec{N}_b - \vec{N}_a}{m} \cdot x\right) = h - g \cdot x,$$

де $h = \vec{L} \cdot \vec{N}_b$.

Отримані результати можна використати й для розрахунку спекулярної складової кольору. Відмінність полягає тільки в тому, що чисельник дробу (3.1) необхідно піднести до степеня n, що можна реалізувати апаратно [30].

Характерна особливість запропонованого підходу полягає у визначенні інтенсивностей кольору відразу для двох точок рядка растеризації. Обчислювальний процес, який реалізовано згідно із запропонованим підходом, не містить у циклі визначення складових кольору операцій ділення та знаходження квадратного кореня, які мають місце при класичній реалізації методу Фонга.

Аналіз показав, що у випадку прискореного визначення дифузної складової кольору для поверхні, обмеженої середньостатистичним трикутником, час розрахунків зменшується в 1,8 рази. При визначенні

спекулярної складової кольору для середньостатистичного трикутника досягається підвищення продуктивності в 1,5 рази.

3.2 Зафарбовування з використанням кутової інтерполяції

Розглянемо використання для визначення інтенсивностей кольору точок поверхні кутової інтерполяції для найпоширенішого випадку, коли вектор $\vec{H}$ має стале положення до графічного об'єкта.

Нехай у трикутнику DCF (рис. 3.1) задано рядок растеризації AB, вектори нормалей до точок якого зображено на рисунку 3.2. Вектори $\vec{N}_A, \vec{N}_B, \vec{H}$ нормалізовано.

Для визначення інтенсивності спекулярної складової кольору використаємо формулу $k_d \cdot I \cdot \cos^n v$ [14], де v – кут між вектором $\vec{H}$ і нормалізованим вектором нормалі $\vec{N}$ до поточної точки рядка растеризації трикутника. У випадку визначення дифузної складової кольору замість вектора $\vec{H}$ використовують вектор $\vec{L}$ і функцію Ламберта.

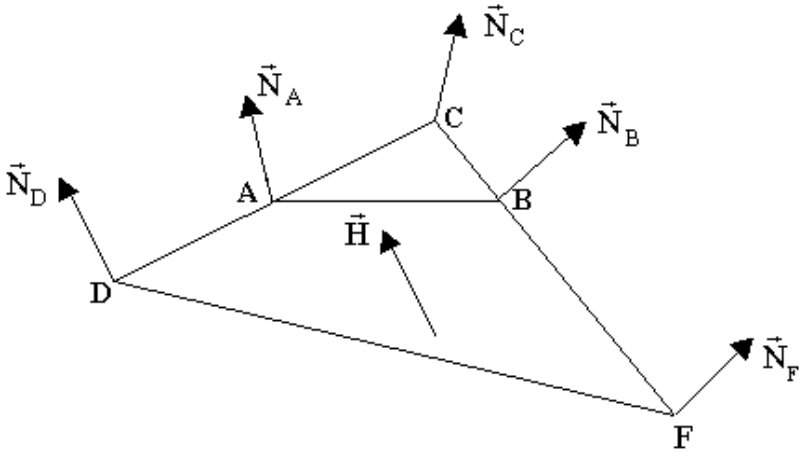


Рисунок 3.1 – Вихідний трикутник

Розділимо кут між векторами нормалей до граничних точок рядка растеризації на його довжину m, отримаємо приріст кута $\Delta\phi = \phi(\vec{N}_A, \vec{N}_B)/m$.

Проведемо через вектор $\vec{H}$ площину, перпендикулярну до площини, яку утворюють вектори нормалей $\vec{N}_A, \vec{N}_B$. Отримуємо прямокутний сферичний трикутник *OHEA*.

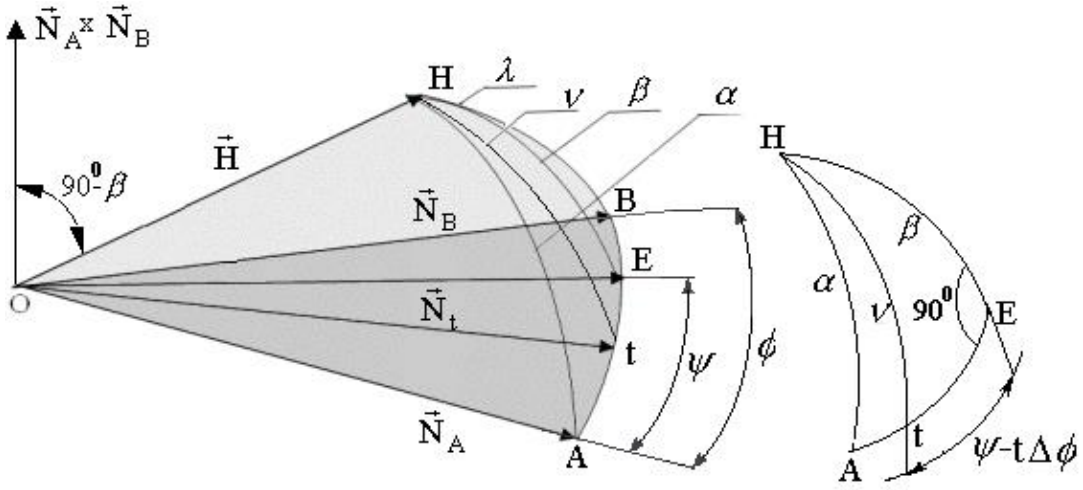


Рисунок 3.2 – Сферичний трикутник, утворений векторами нормалей до рядка растеризації *AB* і вектором $\vec{H}$

Для поточної точки *t* за теоремою Піфагора для сферичного трикутника [28] записуємо

$$\cos\nu = \cos\beta \cdot \cos(\psi - t \cdot \Delta\phi). \qquad (3.4)$$

Із прямокутного сферичного трикутника *OHEA* легко знайти $\cos\psi$. Згідно з теоремою Піфагора для сферичного трикутника $\cos\alpha = \cos\psi \cdot \cos\beta$. Звідси $\cos\psi = \cos\alpha / \cos\beta$.

Знайдемо $\cos\beta$. Проведемо в точці *O* вектор, який перпендикулярний до площини, утвореною векторами нормалей $\vec{N}_A, \vec{N}_B$. Його легко знайти через векторний добуток $\vec{N}_A \times \vec{N}_B$. Нормалізуємо отриманий вектор, ураховуючи що $\left|\vec{N}_A\right| = \left|\vec{N}_B\right| = 1,$

$$\frac{\vec{N}_A \times \vec{N}_B}{\left|\vec{N}_A \times \vec{N}_B\right|} = \frac{\vec{N}_A \times \vec{N}_B}{\left|\vec{N}_A\right| \cdot \left|\vec{N}_B\right| \cdot \sin\phi} = \frac{\vec{N}_A \times \vec{N}_B}{\sin\phi}$$

Отриманий вектор утворює з вектором $\vec{H}$ кут $(90° - \beta)$. Знайдемо синус цього кута через векторний добуток [25] векторів

$$\sin(90° - \beta) = \left|\frac{\vec{H} \times \vec{N}_A \times \vec{N}_B}{\sin\phi}\right|.$$

Використаємо таку властивість подвійного векторного добутку [25]:

$$\vec{H} \times (\vec{N}_A \times \vec{N}_B) = \vec{N}_A \cdot (\vec{H} \cdot \vec{N}_B) - \vec{N}_B \cdot (\vec{H} \cdot \vec{N}_A).$$

З урахування останнього виразу отримуємо

$$\cos\beta = \frac{\sqrt{\vec{N}_A^2 (\vec{H} \cdot \vec{N}_B)^2 - 2(\vec{H} \cdot \vec{N}_B)(\vec{H} \cdot \vec{N}_A)(\vec{N}_A \cdot \vec{N}_B) + \vec{N}_B^2 (\vec{H} \cdot \vec{N}_A)^2}}{\sqrt{1 - \cos^2\phi}}.$$

Ураховуючи, що $\vec{N}_A^2 = \vec{N}_B^2 = 1$, знаходимо, що

$$\cos\beta = \frac{\sqrt{(\vec{H} \cdot \vec{N}_B)^2 - 2(\vec{H} \cdot \vec{N}_B)(\vec{H} \cdot \vec{N}_A)(\vec{N}_A \cdot \vec{N}_B) + (\vec{H} \cdot \vec{N}_A)^2}}{\sqrt{1 - \cos^2\phi}}.$$

Останній вираз запишемо у вигляді

$$\cos\beta = \frac{\sqrt{(\vec{H} \cdot \vec{N}_B)((\vec{H} \cdot \vec{N}_B) - 2(\vec{H} \cdot \vec{N}_A)(\vec{N}_A \cdot \vec{N}_B)) + (\vec{H} \cdot \vec{N}_A)^2}}{\sqrt{1 - \cos^2\phi}}.$$

У чисельнику підкореневого виразу всі множники в дужках є косинусами відповідних кутів між нормованими векторами: $\vec{H} \cdot \vec{N}_A = \cos\alpha$, $\vec{H} \cdot \vec{N}_B = \cos\lambda$, $N_A \cdot N_B = \cos\phi$.

Уведемо позначення $\tau_i = \angle(\vec{N}_C, \vec{N}_A)$, $\mu_i = \angle(\vec{N}_C, \vec{N}_B)$. У подальшому всі параметри, які мають індекс i, відносяться до поточного рядка растеризації AB. Значення τ_i, μ_i легко визначити за формулами:

$$\tau_i = i \cdot \frac{\arccos(\vec{N}_C \cdot \vec{N}_D)}{M_{CD}}, \; \mu_i = i \cdot \frac{\arccos(\vec{N}_C \cdot \vec{N}_F)}{M_{CF}}.$$

Наведені вирази доцільно реалізувати з використанням нагромаджувального додавання:

$$\tau_i = \sum_1^i \frac{\arccos(\vec{N}_C \cdot \vec{N}_D)}{M_{CD}}, \; \mu_i = \sum_1^i \frac{\arccos(\vec{N}_C \cdot \vec{N}_F)}{M_{CF}}.$$

Використовуючи формули косинусів сторін та кутів для сферичного трикутника [28], запишемо

$$\cos\phi_i = \cos\tau_i \cdot \cos\mu_i + \sin\tau_i \cdot \sin\mu_i \cdot \cos\Omega,$$
$$\cos\alpha_i = \cos\tau_i \cdot \cos\angle(\vec{H}, \vec{N}_C) + \sin\tau_i \cdot \sin\angle(\vec{H}, \vec{N}_C) \cdot \cos\Omega_1,$$
$$\cos\lambda_i = \cos\mu_i \cdot \cos\angle(\vec{H}, \vec{N}_C) + \sin\mu_i \cdot \sin\angle(\vec{H}, \vec{N}_C) \cdot \cos\Omega_2,$$

де $\Omega, \Omega_1, \Omega_2$ – відповідно кути між площинами, утвореними парами векторів нормалей $((\vec{N}_A, \vec{N}_C), (\vec{N}_B, \vec{N}_C))$, $((\vec{N}_A, \vec{N}_C), (\vec{H}, \vec{N}_C))$, $((\vec{N}_B, \vec{N}_C), (\vec{H}, \vec{N}_C))$ (рис. 3.3).

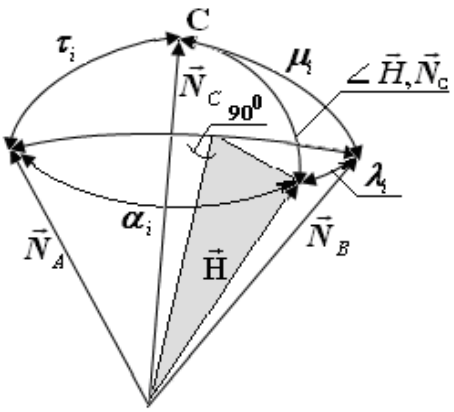


Рисунок 3.3 – Сферичні трикутники, утворені векторами нормалей

Відомо, що при суміщенні початкових точок двох векторів усі проміжні вектори між ними будуть лежати в одній площині. Звідси можна стверджувати, що кут між площинами P_{CA} (утворена векторами нормалей $\vec{N}_A, \vec{N}_C$) і P_{CB} (утворена векторами нормалей $\vec{N}_C, \vec{N}_B$) достатньо розрахувати один раз для трикутника.

Найбільш доцільно це зробити через вектори нормалей $\vec{N}_C, \vec{N}_D, \vec{N}_F$, які задаються для трикутника. Шляхом обчислення векторних добутків $\vec{N}_C \times \vec{N}_D$, $\vec{N}_C \times \vec{N}_F$ отримуємо два вектори $\vec{N}_{CD}$ і $\vec{N}_{CF}$, які є перпендикулярами відповідно до площин P_{CD}, P_{CF}.

Скалярний добуток отриманих векторів за умови їхньої попередньої нормалізації дорівнює косинусу кута Ω між площинами P_{CD} і P_{CF}. Слід зазначити, що $\cos\Omega$ можна знайти і за формулою $\cos\Omega = \cos(\Omega_1 + \Omega_2) = \cos\Omega_1 \cdot \cos\Omega_2 - \sin\Omega_1 \cdot \sin\Omega_2$.

Використовуючи властивості скалярного та векторного добутків векторів, отримуємо такі вирази :

$$\cos\phi_i = \cos\tau_i \cdot \cos\mu_i + \sin\tau_i \cdot \sin\mu_i \cdot \cos\Omega,$$
$$\cos\alpha_i = \cos\tau_i \cdot (\vec{H} \cdot \vec{N}_C) + \sin\tau_i \cdot \left|\vec{H} \times \vec{N}_C\right| \cdot \cos\Omega_1,$$
$$\cos\lambda_i = \cos\mu_i \cdot (\vec{H} \cdot \vec{N}_C) + \sin\mu_i \cdot \left|\vec{H} \times \vec{N}_C\right| \cdot \cos\Omega_2.$$

Запишемо вираз (3.4) у вигляді

$$\cos\nu = \cos\beta \cdot \cos(\psi - t \cdot \Delta\phi) = \cos\beta \cdot (\cos\psi \cdot \cos t \cdot \Delta\phi + \sin\psi \cdot \sin t \cdot \Delta\phi).$$

Такий запис дозволяє уникнути обчислення арккосинуса кута для знаходження ψ, що мало місце при розрахунку за формулою (3.4).

При зафарбовувані растеризацію виконують одночасно в обох напрямках від точки E, де $\cos\nu = \cos\beta$, до того часу, поки поточне значення $\cos\nu$ не досягне меншого із значень $(\cos\alpha, \cos\lambda)$. Оскільки відстань від точки E до точок A і B – різна, то в одному з напрямків растеризацію закінчують раніше шляхом її блокування. При такому підході $\cos\nu = \cos\beta \cdot \cos(t \cdot \Delta\phi)$. Точку E легко знайти, обчисливши $\arccos\psi$.

На рисунку 3.4 зображено два прямокутних сферичних трикутники, які мають дві однакові сторони, що симетрично розміщені відносно точки E. Оскільки в зазначених трикутниках є спільний катет HE, а інші катети – рівні, то гіпотенузи ν_k і ν_{-k} також рівні.

Це означає, що достатньо виконувати розрахунки тільки вздовж дуги EA від точки E (рис. 3.4), використовуючи їх для відповідних симетричних точок уздовж дуги EB. При такому підході в кожному такті косинус кута знаходиться відразу для двох точок рядка растеризації трикутника.

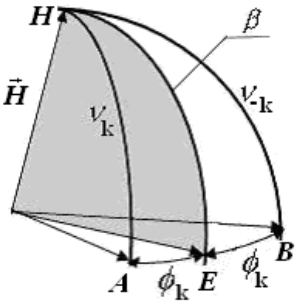


Рисунок 3.4 – Прямокутні сферичні трикутники

У запропонованому методі, хоча і вилучено нормалізацію векторів нормалей, але в циклі підготування до визначення інтенсивностей кольору використовується операція розрахунку арккосинуса кута.

Моделювання з використання графічного конвеєра 3dfx показало, що час зафарбовування середньостатистичного трикутника зменшився порівняно з класичною реалізацією у середньому на 25% .

3.3 Апаратна реалізація методів формування тривимірних графічних схем

У підрозділі 3.2 розроблено метод зафарбовування, особливість якого полягає у вилученні з обчислювального процесу трудомісткої процедури нормалізації векторів за рахунок використання кутової інтерполяції. На рисунку 3.5 зображено основні етапи розрахунку спекулярної (дифузної) складових інтенсивностей кольору на рівні блоків, які можуть бути реалізовані програмним або апаратним шляхом.

На рисунку 3.6 зображено структуру блока для апаратного обчислення $\cos v$ за формулою (3.4). За векторами нормалей $\vec{N}_A$ і $\vec{N}_B$ розраховують кут між ними, а потім $\Delta\phi$. Це передбачає виконання операцій арккосинуса та ділення. За значеннями векторів нормалей $\vec{N}_A, \vec{N}_B, \vec{H}$ визначається $\cos\beta$. Перераховані дії доцільно виконувати програмним шляхом, оскільки вони досить трудомісткі. Подальші дії легко реалізувати апаратним шляхом. Для кожної точки рядка растеризації трикутника визначається косинус кута v, який є вхідним параметром для розрахунку ДФВЗ. За значенням дистрибутивної функції визначають дифузну та спекулярну складові кольору.

На рисунку 3.6 зображено структуру блока для обчислення $\cos v$ за формулою (3.4), який включає нагромаджувальний суматор PSm, регістри $RG1, RG2$, блок постійної пам'яті $Prom$ для зберігання значень $\cos x$, блок множення. У початковий момент часу в регістри $RG1, RG2$ заносяться відповідно значення $\Delta\phi$, $\cos\beta$, а в нагромаджувальний суматор – ψ. У ньому в

кожний тактовий момент часу розраховується значення аргументу $\psi - t \cdot \Delta\phi$, яке надходить на вхід блоку постійної пам'яті для знаходження $\cos(\psi - t \cdot \Delta\phi)$.

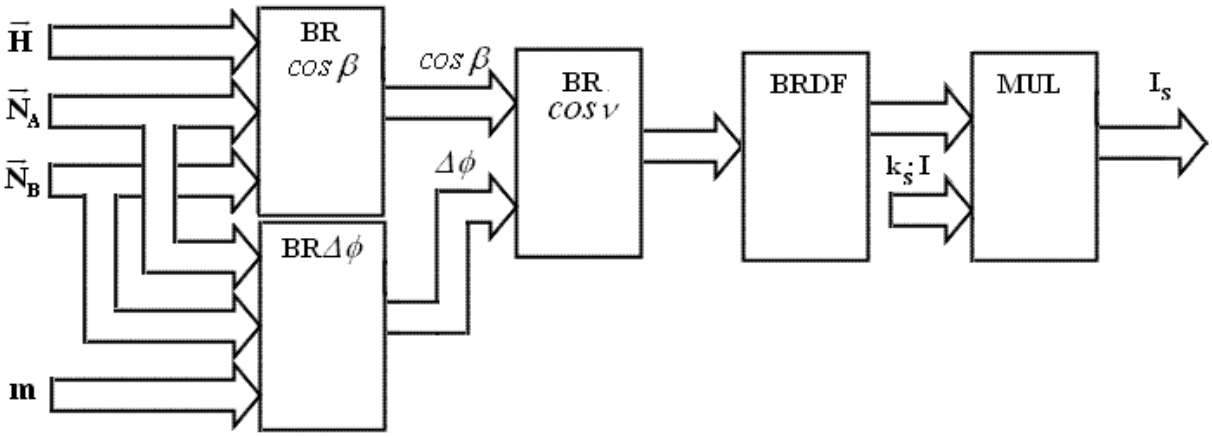


Рисунок 3.5 – Структура пристрою визначення спекулярної складової кольору

Нагромаджувальний суматор працює в режимі віднімання, для чого на його вхід перенесення p_v подають рівень логічної одиниці. На виході блока множення знаходять $\cos\nu = \cos\beta \cdot \cos(\psi - t \cdot \Delta\phi)$.

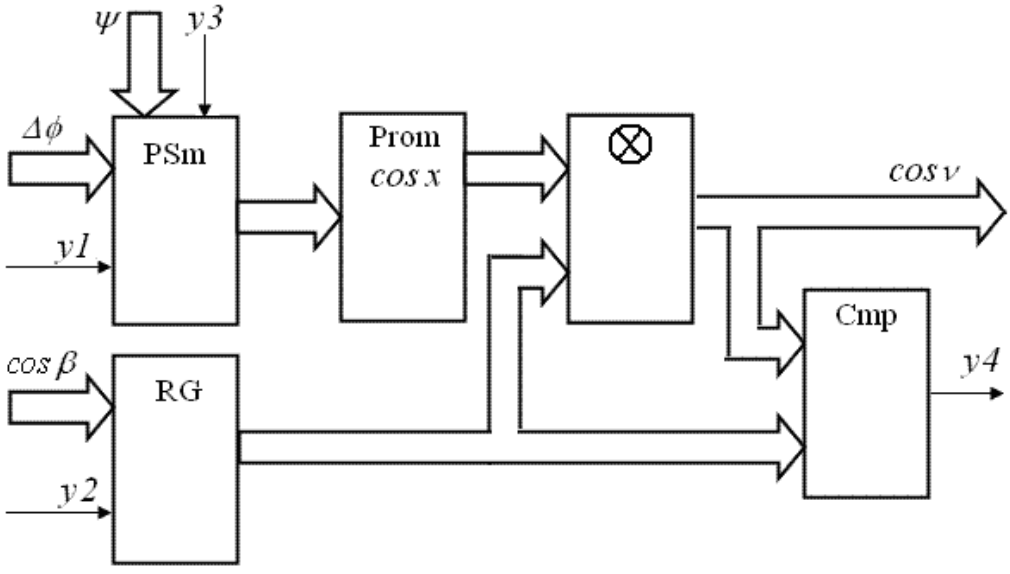


Рисунок 3.6 – Структурна схема блоку визначення $\cos\nu$

Обчислення завершують за умови, що $\psi - t \cdot \Delta\phi < 0$, про що можна судити за знаком вихідного переносу p_{vv}. У подальшому нагромаджувальний

суматор установлюють у стан $\psi := \phi - \psi$ і повторюють дії для правої частини рядка растеризації.

У запропонованому методі необхідно за значенням $\vec{N}_A \cdot \vec{N}_B = \cos\phi$ визначати $\phi = \arccos(\vec{N}_A \cdot \vec{N}_B)$. Розрахунок арккосинуса на програмному рівні вимагає значних затрат часу, тому доцільно його знаходити апаратно. Було проведено розклад для функції arccos$x$ за багатогочленами Чебишова [25]. Порівняно з розкладом у ряд Тейлора в цьому випадку має місце суттєвий виграш у швидкості збігання (розклад функції arccos$x$ у ряд Тейлора, який включає три члени (багатогочлен третього степеня), повільно збігається біля границі ділянки $|x| \le 1$ і вже на проміжку $0 \le x < 0{,}8$ має відносну похибку апроксимації, більшу за 6%). Особливості кусково-лінійної апроксимації функції arccos$x$ розглянуто в [28]. Аналіз показав, що для апаратної реалізації найбільш доцільно використати багатогочлен другого степеня. Це пояснюється тим, що при використанні кусково-лінійної апроксимації має місце відносно велика похибка, для зменшення якої використовують велику кількість інтервалів апроксимації. Це призводить до необхідності зберігання великої кількість коефіцієнтів розкладу. При використанні полінома третього степеня збільшується кількість блоків множення або кількість ітерацій за умови послідовної реалізації.

Наведені оцінки апаратної складності розроблених вузлів для визначення інтенсивностей складових кольору дає можливість вибрати для їх реалізації елементну базу та оцінити їх ефективність, наприклад, за узагальненим критерієм кваліметрії, який визначає відношення отриманого ефекту до затрат на його реалізацію. Наприклад, для реалізації пристроїв для розрахунку спекулярної складової кольору з використанням логарифмічної, квадратичної, косинус-квадратичної ДФВЗ необхідно біля 3000 логічних вентилів, однак найефективнішою є реалізація з косинус-квадратичною дистрибутивною функцією, оскільки вона не вимагає блоку постійної пам’яті.

Розглянемо детальніше адресний блок. На рисунку 3.7 зображено його структурну схему.

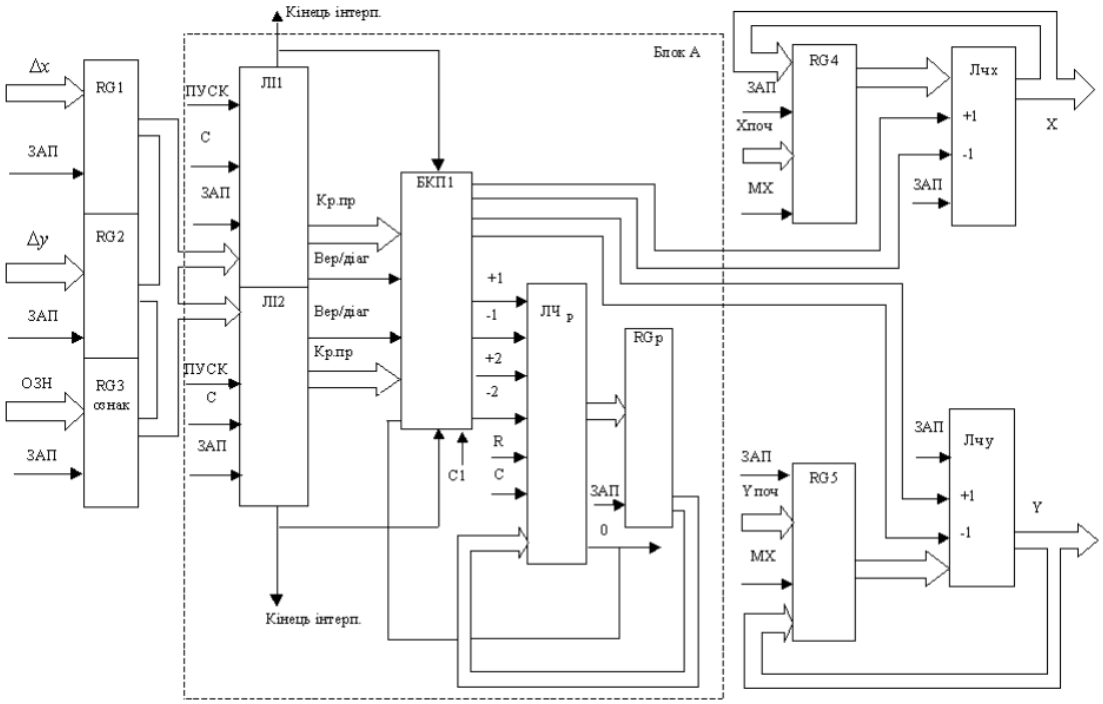


Рисунок 3.7 – Структурна схема адресного блоку

На вході адресного блоку розташовані регістри: $RG1$ – регістр приростів Δx, $RG2$ – регістр приростів Δy, $RG3$ – регістр вектору ознак (атрибутів, властивостей). У внутрішній частині адресного блоку (блок A) вміст регістрів подається до лінійних інтерполяторів $ЛІ1$ та $ЛІ2$. Дані інтерполятори виконують обчислення над значеннями приростів Δx, Δy робочого ($ЛЧ1$) та одного з суміжних з ним ($ЛЧ2$) ребер, а також їхні властивості: знаки приростів Δx, Δy; порівняльна ознака даних приростів. Варто зазначити, що робочим є ребро, від якого виконується процес забарвлювання. Обчислення при зафарбовуванні завжди проводяться зі значеннями властивостей робочого та суміжного з ним ребер.

Регістри $RG5$ та $RG6$ отримують координати x та y на інформаційні входи $x_{поч}$ та $y_{поч}$ відповідно. Це координати вершини трикутника. З регістрів

вони записуються до лічильників $ЛЧ_x$ та $ЛЧ_y$. Також виконується обнулення лічильника рядків $ЛЧ_p$. Коли лінійні інтерполятори $ЛІ1$ та $ЛІ2$ ініціалізовано, можна записувати до регістрів прирости координат та властивості іншого суміжного ребра. На цьому завершується цикл підготовки.

Процес зафарбовування починається зі стартового сигналу та виконується згори донизу. Незалежно один від одного лінійні інтерполятори $ЛІ1$ та $ЛІ2$ розраховують крок та напрям переміщення для робочого та суміжного ребер. Якщо формується діагональний або вертикальний крок, то виникає зупинка у роботі відповідного інтерполятора.

З допомогою блоку крокових приростів $БКП1$ із блоку A перед тим, як обидва інтерполятори зупиняться, лічильник рядків $ЛЧ_p$ генерує значення – відстань від точок робочого ребра у рядку растеризації до точок суміжного ребра. Рядковий лічильник відрізняється від звичайних двійкових лічильників тим, що використовує для зміни внутрішнього значення в ролі приростів сигнали (± 1 та ± 2), отримані на виході блоку крокових приростів у результаті роботи лінійних інтерполяторів.

Зупинка у роботі лінійних інтерполяторів призводить до формування послідовності імпульсів для крокових приростів у рядку растеризації. Кожну ітерацію над значенням рядкового лічильника виконується операція декременту допоки лічильник не обнулиться. Рядковий регістр RG_p зберігає значення довжини наступного рядка растеризації, яке записується до рядкового лічильника після його обнулення. Також для переходу на наступний рядок растеризації значення координат x та y точки на робочій стороні трикутника, з якої починається рядок растеризації, записують до лічильників координат $ЛЧ_x$ та $ЛЧ_y$.

Процес триває доти, доки один з лінійних інтерполяторів не сформує відповідну сторону трикутника. Потім виконується формування ребра, що залишилося. У результаті роботи блоку адрес отримують набір адрес точок трикутника.

Для розрахунку відтінку точки I використовується блок визначення інтенсивності (БВІ). Його можна описати як кодовий лінійний інтерполятор. На рисунку 3.8 представлена структурна схема блоку, що розглядається.

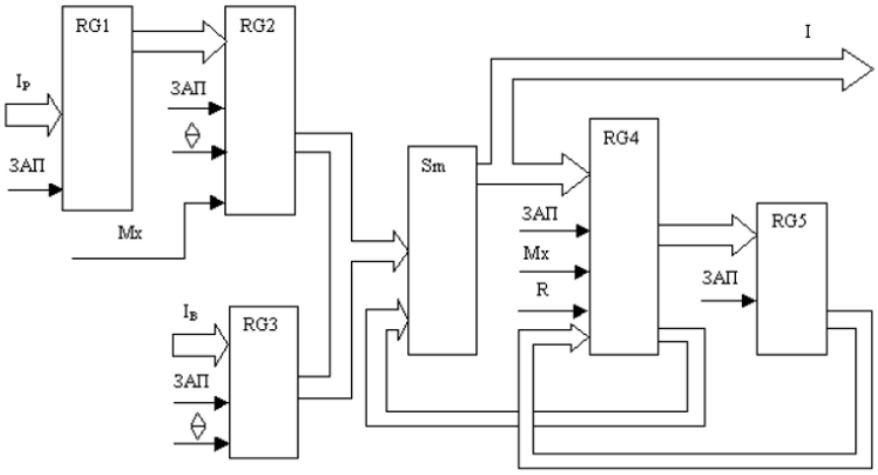


Рисунок 3.8 – Структурна схема блоку визначення інтенсивності

Знаходження відтінку точки виконано через додавання з накопиченням. Структурна схема БВІ складається з п’яти регістрів $RG1$-$RG5$ та комбінаційного суматора Sm.

На етапі ініціалізації регістр $RG2$ отримує з регістра $RG1$ значення різниці відтінків сусідніх точок ΔI по робочій стороні трикутника. Після цього до регістра $RG1$ вноситься значення ΔI по другій робочій або суміжній з робочою стороні. Регістр $RG3$ виконує подвійну функцію. Спершу він містить значення відтінку вершини трикутника, яка розташована першою на шляху растеризації. Паралельно з цим виконується обнулення регістра $RG4$ та блокування регістра $RG2$, тому суматор Sm записує до регістра $RG4$ вміст регістра $RG3$ – значення відтінку верхньої точки трикутника. Етап ініціалізації завершується записом до регістра $RG3$ значення ΔI по стороні растеризації. Основна робота блоку починається з блокування регістра $RG3$. До вмісту регістра $RG4$ додається значення ΔI з регістра $RG2$. Процес виконується доти, доки кроковий приріст по робочій стороні, отриманий з блоку адрес, не стане вертикальним або

діагональним – тоді виконується блокування регістра $RG2$ та запис вмісту регістра $RG4$ до регістра $RG5$.

Після цього виконується інтерполяція відтінків у точках ліній растеризації (окрім останньої). Вміст регістра $RG4$ збільшується щотакту на значення приросту ΔI, яке зберігається в регістрі $RG3$. Остання точка лінії растеризації має визначений відтінок, який зберігається до завершення процесу в регістрі $RG5$. Коли останню точку робочої сторони уже сформовано адресним блоком, БВІ починає обчислювати відтінки точок другої робочої сторони.

Розглянемо блок адрес для іншого методу зафарбовування (рис. 3.9)

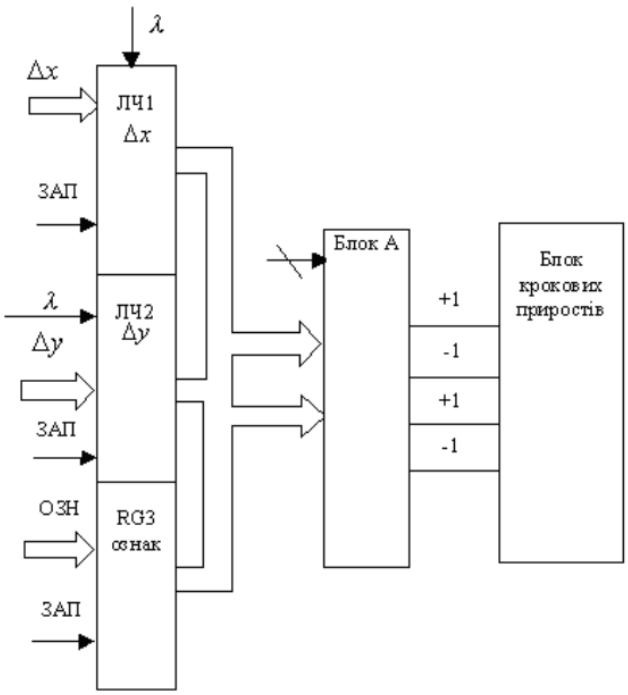


Рисунок 3.9 – Структурна схема модифікованого блоку адрес

Тонування з використанням фрактального розбиття на трикутники використовує аналогічний блок адрес з деякими відмінностями. Лічильники приростів $Лч1\,\Delta x$ та $Лч2\,\Delta y$ замінили відповідні регістри приростів $RG1$ та $RG2$.

До основної пари лічильників адрес $Лі\,x_1$ і $Лі\,x_2$ та відповідних регістрів адрес $RG4$ та $RG5$ блок адрес отримує ще три пари лічильників та регістрів, оскільки даний блок обчислює адреси точок чотирьох трикутників одночасно (рис 3.10).

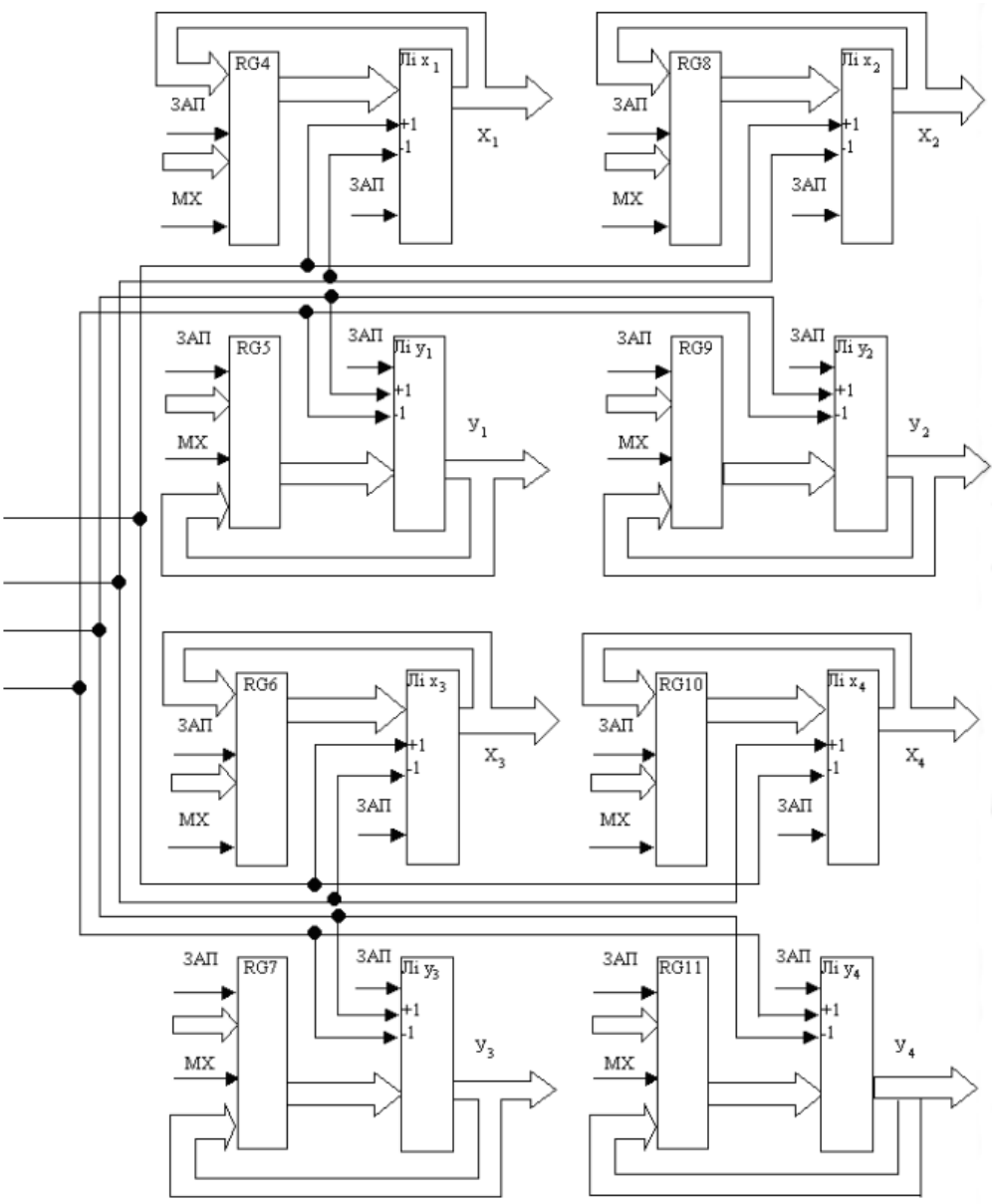


Рисунок 3.10 – Структурна схема блоку адрес для одночасного обчислення значень для чотирьох трикутників

Під час підготовки до роботи координати вершин трикутників вносяться до регістрів, причому три регістри описують верхні точки відповідних трикутників, а регістри *RG*10 та *RG*11 – нижню точку. До лічильника *Лч*1 записується ΔI робочої сторони. Над цим приростом виконується операція інкременту. Блок *A* отримує вміст лічильника *Лч*1 , зсунутий вправо. Аналогічну операцію виконують на лічильнику *Лч*2 . Регістр *RG*3 містить вектор атрибутів приростів.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ФОРМУВАННЯ ТРИВИМІРНИХ ГРАФІЧНИХ СХЕМ. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Обґрунтування вибору засобів реалізації

Потрібно розглянути можливості отриманих способів покращення процесу рендерингу, а також виявити ймовірні проблеми у процесі розробки та використання, знайти можливі способи усунення помилок.

Прототип – це спосіб швидко та недорого перевірити запропоновані ідеї. Він потрібен для того, щоб уникнути проблем на пізніх етапах розробки. Технічний прототип – це чернетка, швидка реалізація системи, яка дозволяє спростити дослідження. Основні вимоги до даної програми – інтерактивність, швидкість розробки та повна демонстрація функцій, що досліджуються.

Технічний прототип створено у вигляді браузерного застосунку. Основною мовою розробки програми обрано мову програмування JavaScript, додатково використано мову розмітки HTML для створення інтерфейсу та мову шейдерів GLSL ES для використання можливостей апаратного прискорення графічним процесором. Програму написано в редакторі коду з можливостями середовища розробки Visual Studio Code. Розглянемо детальніше обрані засоби реалізації.

JavaScript – динамічна багатоплатформна скриптова мова з Сі-подібним синтаксисисом, яка підтримує об’єктно-орієнтоване та функціональне програмування. Це одна з трьох основних мов для веб-розробників: мова розмітки HTML описує вміст сторінок, мова стилів CSS уточнює зовнішній вигляд, а скриптова мова JavaScript керує поведінкою сторінки та її елементів, що дозволяє створювати веб-застосунки, інтерактивні елементи.

На відміну від багатьох інших мов, у ній відсутня концепція введення-виведення. Це мова скриптів, розроблена для запуску в робочому оточенні, яке забезпечує механізми зв’язку з навколишнім світом. Браузер – найпоширеніше

робоче оточення для JavaScript. У браузері JavaScript керує елементами об’єктної моделі документа, змінюючи зовнішній вигляд сторінки, та реагує на дії користувача, як-от: натискання кнопки, зміна вмісту поля форми. Всередині робочого оточення JavaScript під’єднується до об’єктів оточення та забезпечує програмований контроль над ними.

Підтримка JavaScript усіма браузерами робить веб-застосунки багатоплатформними. Серйозних змін з часом стандарт мови не зазнає, тому нові браузери майже завжди підтримують старі програми. Проте завжди варто пам’ятати про сумісність при використанні свіжих можливостей JavaScript.

Мова містить стандартну бібліотеку класів та ядро з набором мовних елементів, а саме операторів, керівних конструкцій та виразів. Ядро можна розширити для конкретної задачі.

JavaScript – популярна мова. Простий синтаксис дозволяє зменшити поріг входу. Поблажливе ставлення до дій користувача (так творці мови намагалися зробити програмування простішим для новачків) також дає певну гнучкість, недоступну іншим мовам.

Елемент <canvas> (частина стандарту HTML5) дозволяє динамічно формувати растрову графіку за допомогою коду мовою JavaScript. Тег створює поле фіксованого розміру, а один або декілька контекстів рендерингу описують вміст цього поля. Контекст рендерингу – це контейнер, який зберігає інфор22мацію про стан сцени.

WebGL API [34] – прикладний програмний інтерфейс мовою JavaScript для швидкого формування інтерактивної дво- та тривимірної графіки, сумісної з браузером, без сторонніх плагінів та з використанням апаратного прискорення. WebGL API відповідає стандарту OpenGL ES 2.0 та використовує відповідний тривимірний контекст. Технологія кросбраузерна та багатоплатформна, проте потребує сумісного апаратного забезпечення. Майже всі сучасні комп’ютерні та мобільні браузери підтримують технологію WebGL (рис. 4.1).

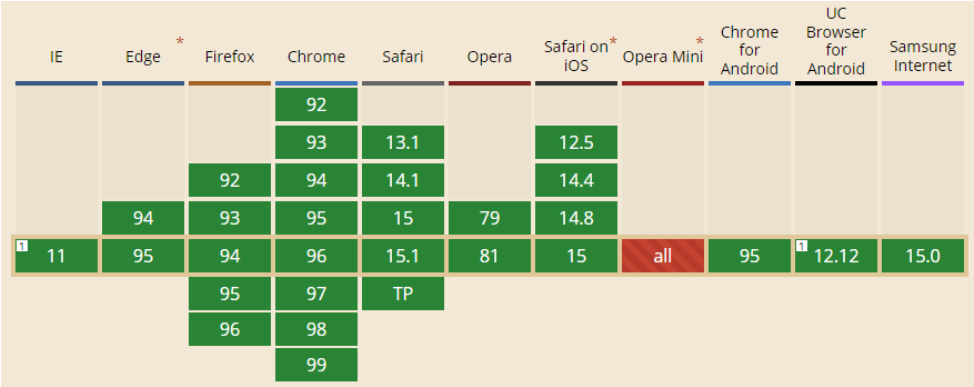


Рисунок 4.1 – Підтримка сучасними браузерами можливості формування тривимірної графіки в елементі <canvas> з використанням технології WebGL

Програма, що використовує WebGL API, складається з керуючого коду мовою JavaScript та коду шейдерів мовою GLSL ES.

Шейдер – програма, яка виконується в графічному процесорі. Мова шейдерів розроблена для роботи з архітектурою апаратного прискорювача SIMD (одна інструкція, багато даних). Графічний процесор підтримує роботу з векторами даних. Тому мова містить вбудовану підтримку векторних типів. У процесі роботи з графікою часто використовують двовимірні вектори (x, y) для представлення векторів площини, тривимірні (x, y, z) – для векторів простору або кольорів у системі *RGB* , чотиривимірні (x, y, z, w) – для кольорів у системі *RGBA*.

Стандартні математичні операції та функції реалізовано як для скалярних, так і для векторних аргументів. Також реалізовано спеціальні функції для роботи з векторами (наприклад, скалярний та векторний добуток векторів) і функції для роботи з матрицями.

Visual Studio Code або VS Code – легкий, проте потужний редактор коду. Він містить вбудовану підтримку мови JavaScript, а також набір інструментів для налагодження та зневадження, форматування та пошуку, валідації та підвищення якості коду, контролю версій. Розширення, створенні користувачами, додають підтримку інших мов та нові можливості (рис 4.2).

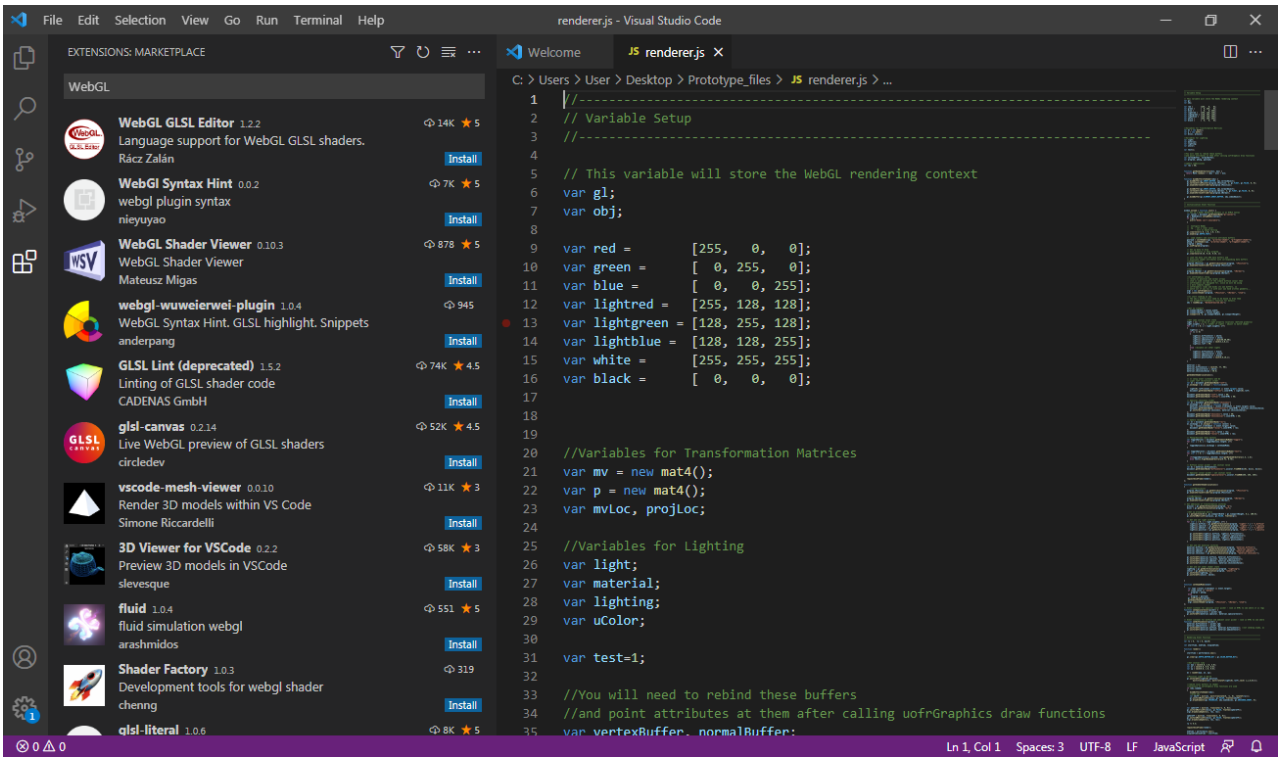


Рисунок 4.2 – Вікно редактора VS Code з відкритим файлом мовою JavaScript та списком доступних розширень для WebGL

4.2 Розробка програмних засобів

У магістерській кваліфікаційній роботі розроблено прототип програмного засобу, який дозволить змоделювати та протестувати методи зафарбовування.

Отриманий прототип дозволяє:

– формувати тривимірні графічні схеми з використанням обраних методів;

– встановлювати для формування різні значення деталізації об’єктів, розміру відблиску, кута падіння світлового променя;

– визначати час формування графічних об’єктів.

Програму написано мовою JavaScript на основі бібліотеки WebGL. Зв’язки між класами, з яких створено прототип, наведено на рисунку 4.3.

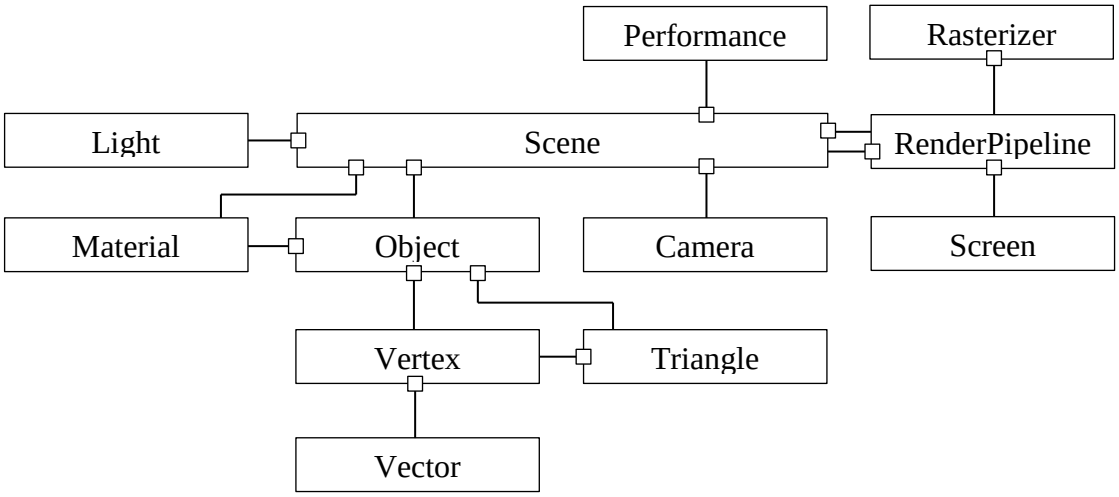


Рисунок 4.3 – Зв’язки між класами прототипу програми

Коротко розглянемо використані класи:

- Vector – вектор як тип даних, а також методи (скалярний та векторний добуток, нормалізація, модуль) та перевантажені операції;

- Vertex – вершини, описані через координати та вектори;

- Triangle – трикутників як набір вершини;

- Object – об’єкти (моделі), подані через вершини та трикутники – трикутна сітка (меш);

- Material – опис властивостей зовнішнього вигляду об’єкта (колір, текстура, шорсткість);

- Light – джерело світла та його властивості;

- Camera – опис точки та поля зору глядача;

- Scene – комплексний об’єкт, що складається з тривимірних моделей та джерел світла, розташованих у просторі певним чином;

- RenderPipeline – графічний конвеєр, що керує процесом рендерингу;

- Rasterizer містить реалізацію методів зафарбовування, включно з тими, які потрібно дослідити;

- Screen – дані про ділянку виведення сформованої тривимірної графічної схеми;

- Performance забезпечує фіксацію часу формування тривимірної графічної схеми.

На рисунку 4.4 зображено логіку роботи прототипу програми.

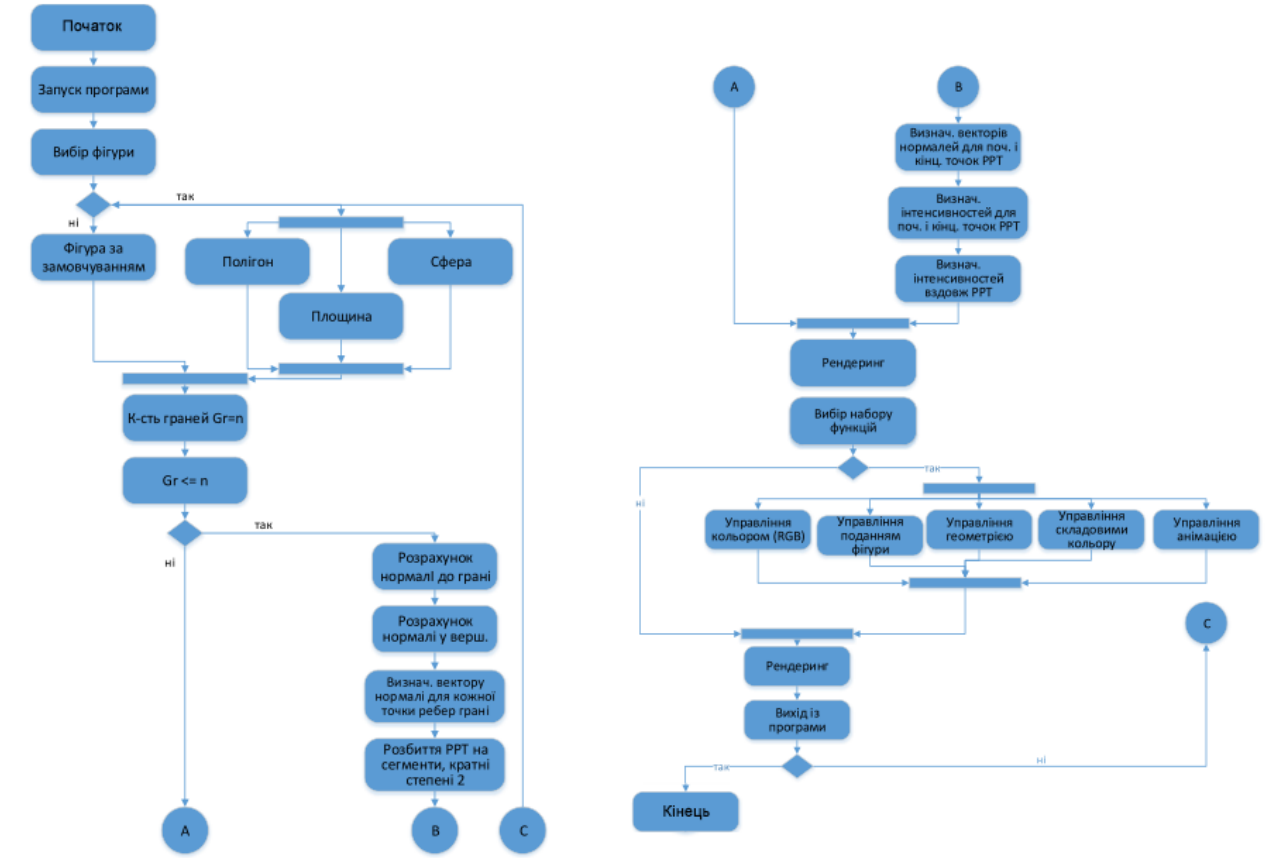


Рисунок 4.4 – Схема роботи прототипу програми

Користувач після запуску та завантаження програми повинен мати змогу обрати модель, на якій буде продемонстровано метод зафарбовування. Після цього доцільно встановити рівень деталізації. Можна також змінювати властивості джерела світла (кут падіння, потужність), матеріалу (дифузна та спекулярна складові кольору).

Графічний інтерфейс прототипу складається з двох основних функціональних частин (рис 4.5).

Перша – ділянка формування графіки, елемент <canvas>. Вона призначена для відображення результатів рендерингу (тривимірних графічних схем).

Друга – панель налаштувань та інформації, яка дозволяє змінювати параметри рендерингу та бачити їхні поточні значення.

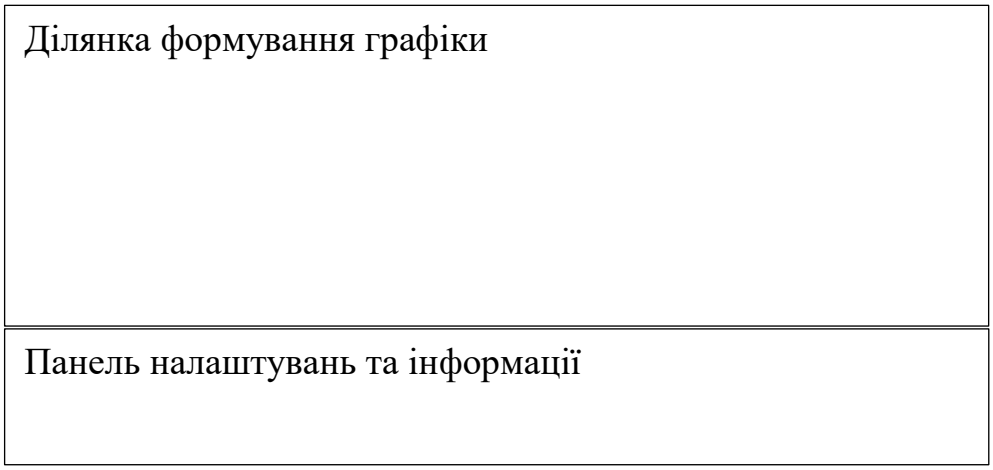


Рисунок 4.5 – План інтерфейсу для прототипу програми

Також тут відображається час формування еталону та зображення, що порівнюється з ним. Користувач може встановити рівень деталізації сфери, розмір відблиску, кут падіння світла, колір дифузної та спекулярної складових, методи зафарбовування зображень (рис 4.6).

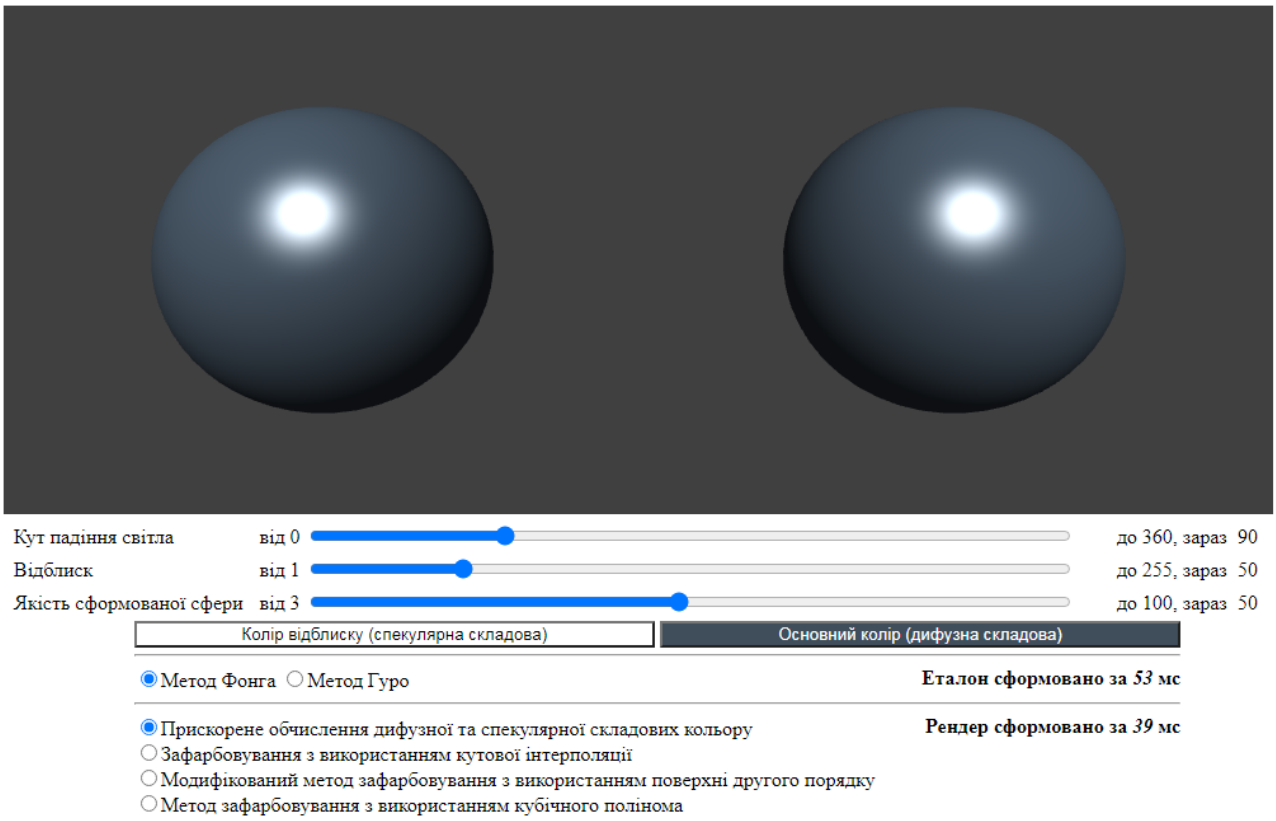


Рисунок 4.6 – Знімок екрану готового прототипу програми

4.3 Експериментальні дослідження

Розглянемо можливості використання розроблених методів покращення процесу рендерингу. Виконані розрахунки та створений прототип дозволили продемонструвати та підтвердити працездатність запропонованих методів. Тепер потрібно визначити їхню ефективність. У цьому підрозділі розглядаються дані – результати експериментальних досліджень.

Для проведення тестування використано портативний персональний комп’ютер, оснащений двоядерним процесором Intel Core i3-2310M (Sandy Bridge) з тактовою частотою 2.10 ГГц, оперативним запам’ятовуючим пристроєм типу DDR3 з частотою 1600МГц та доступним об’ємом пам’яті 3,76 Гб, двома відеокартами – вбудованою Intel GMA HD 3000 та дискретною nVidia GeForce 610M з об’ємом відеопам’яті 1 Гб.

Комп’ютер працює під управлінням операційної системи Windows 7 для 64-х розрядних процесорів. Характеристики апаратного забезпечення посередні, проте вони задовольняють вимоги.

Отримані методи та програмно-апаратні засоби рендерингу тривимірних графічних схем варто порівняти зі стандартом. Критеріями порівняння можуть бути: відповідність сформованих зображень еталону, швидкість формування зображень, продуктивність рендерингу.

Доцільно проводити випробування отриманих методів та засобів як елементів графічного конвеєра. Це дозволить отримати результати для порівняння у контексті взаємодії з іншими процесами.

Візуальна різниця двох зображень кількісно оцінюється з використанням нормованої середньоквадратичної похибки *NMSE*. Для кольорових зображень з пікселями, описаними у форматі *RGB*, дану похибку обчислюють за формулою

$$NMSE = \frac{\sum_i (R_i - r_i)^2 + (G_i - g_i)^2 + (B_i - b_i)^2}{\sum_i R_i^{\,2} + G_i^{\,2} + B_i^{\,2}},$$

де i – число точок зображення, R_i, G_i, B_i – інтенсивність червоної, зеленої та синьої складових кольору точки еталонного зображення; r_i, g_i, b_i – інтенсивність червоної, зеленої та синьої складових кольору точки отриманого зображення, яке порівнюється з еталонним.

У таблиці 4.1 описано, як трактувати значення нормованої середньоквадратичної похибки двох зображень.

Таблиця 4.1 – Значення *NMSE* для порівняння двох зображень

Проміжок значень *NMSE*	Схожість зображень
$\leq 0{,}0001$	Однакові на вигляд
$(0{,}0001;\ 0{,}00025]$	Мають невеликі розбіжності
$(0{,}00025;\ 0{,}001]$	Помітно відрізняються
$> 0{,}001$	Зовсім різні

Для знаходження *NMSE* було розроблено програму з текстовим інтерфейсом, написану мовою C#. На рисунку 4.7 наведено фрагмент лістингу коду – функцію для визначення *NMSE* двох зображень. На вхід подаються два зображення – еталон *imgS* та зображення для порівняння *imgC*. У програмі вони зберігаються та опрацьовуються як два об’єкти типу Bitmap – масиви пікселів. Кожен піксель представлений у форматі RGBA та займає 32 біти.

Загальна кількість ітерацій вкладеного циклу дорівнює числу точок зображення. Накопичувальні суми *standard* та *comp* зберігаються як цілі числа типу long. Результат роботи функції – значення *NMSE* – число з плаваючою крапкою подвійної точності (тип double). Це частка двох цілочисельних накопичувальних сум, тому потрібно виконати явне перетворення типів.

```
for (int i = 0; i < imgS.Width; i++)
{
	for (int j = 0; j < imgS.Height; j++)
	{
		pixel1 = imgS.GetPixel(i, j);
		R = pixel1.R;
		G = pixel1.G;
		B = pixel1.B;
		standard += R * R + G * G + B * B;

		pixel2 = imgC.GetPixel(i, j);
		r_ = R - pixel2.R;
		g_ = G - pixel2.G;
		b_ = B - pixel2.B;
		comp += r_ * r_ + g_ * g_ + b_ * b_;
	}
}
res = (double)comp / standard;
```

Рисунок 4.7 – Функція для визначення *NMSE* двох зображень, реалізована мовою C#

Повний лістинг коду програми наведено в додатку В.2. Результати порівняння рендерів, отриманих з використанням розроблених методів, та еталонного рендера дозволяють стверджувати про візуальну ідентичність сформованих тривимірних графічних схем. Одержані значення *NMSE* подано у таблиці 4.2.

Таблиця 4.2 – Максимальні значення *NMSE* для розроблених методів

Метод зафарбовування	Максимальне значення *NMSE*
Прискорене обчислення дифузної та спекулярної складових кольору	0,00011
Зафарбовування з використанням кутової інтерполяції	0,00018
Модифікований метод зафарбовування з використанням поверхні другого порядку	0,00019
Метод з використанням кубічного полінома	0,00015

Результати тестування показують, що отримані методи формують зображення, схожі на еталонне. Тому робити висновок про доцільність використання нових засобів формування тривимірних графічних схем потрібно після того, як буде порівняно швидкість та продуктивність рендерингу. Тривалість формування тривимірної графічної схеми можна оцінювати емпірично та аналітично.

Емпірична оцінка базується на різниці між виміряним часом початку процесу рендерингу та часом його закінчення. Час виконання програми у комп'ютерній системі може вимірюватись або в одиницях часу СІ та похідних (мілісекунди, секунди, хвилини), або в тактах процесора.

Аналітична оцінка потребує підрахунку всіх операцій, які виконує програма, для подальшого визначення часу виконання як суми добутків кількості операцій кожного виду на їхню «вартість» в одиницях часу. Під час досліджень продуктивність рендерингу розглядалася як кількість операцій, виконаних за відрізок часу. Даний критерій можна розрахувати окремо, або у процесі знаходження тривалості рендерингу аналітичним способом.

Поверхні розбито на середньостатистичні трикутники. Ділянка поверхні, обмежена таким трикутником, включає 100 точок (рис. 4.8). Для блискучих поверхонь коефіцієнт спекулярності дорівнює 256 [27]. Використано трикутник ABC з $\Delta x_{AB} = 7,\ \Delta y_{AB} = 8,\ \Delta x_{AC} = 9,\ \Delta y_{AC} = 12,\ \Delta x_{BC} = 16,\ \Delta y_{BC} = 4.$

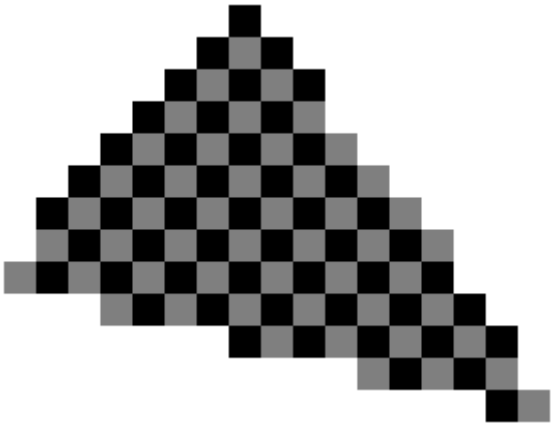


Рисунок 4.8 – 100 точок, які обмежені середньостатистичним трикутником

Для оцінки трудомісткості виконання процедур кінцевої візуалізації використано кількість тактів процесора Pentium M [31] для виконання складових інструкцій. Інструкція задає одну окрему операцію процесора, визначену системою його команд.

Виконання операцій для прискореного визначення дифузної та спекулярної складових кольору для середньостатистичного трикутника, наведених у таблиці 4.3, при використанні процесора Pentium M складає 20823 такти. Порівняно з класичною реалізацією час розрахунку дифузної складової кольору для середньостатистичного трикутника зменшився в 1,83 рази.

Таблиця 4.3 – Склад і кількість інструкцій для прискореного визначення дифузної та спекулярної складових кольору для середньостатистичного трикутника

Процедура	Кількість інструкцій			
	«+»	«х»	«:»	Зсув
Розрахунок векторів $\vec{N}$	396	–	42	–
Розрахунок $b, \tilde{n}, s, g$	165	396	33	66
Розрахунок p_x	258	–	–	–
Розрахунок $\sqrt{q_x}$	342	489	–	114
Визначення інтенсивностей дифузної складової кольору внутрішніх точок поверхні, обмеженої трикутником	–	600	–	–
Разом	1161	1485	75	180

Якщо за середнє значення коефіцієнта спекулярності вибрати 255, то для визначення інтенсивностей спекулярної складової кольору необхідно додатково виконати 800 операцій множення.

В цьому випадку для розрахунку інтенсивностей спекулярної складової кольору необхідно виконати 29623 такти. Порівняно з класичною реалізацією досягається підвищення продуктивності в 1,71 рази.

Виконання операцій при зафарбовуванні середньостатистичного трикутника за методом Фонга (таблиці 4.4) при використанні процесора Pentium M витрачає 146036 тактів.

Таблиця 4.4 – Склад і кількість інструкцій для зафарбовування середньостатистичного трикутника за методом Фонга

Процедура	Кількість інструкцій				
	«+»	«x»	«:»	«$\sqrt{\ }$»	Зсув
Визначення векторів та їхня нормалізація	2684	1200	1524	400	300
Визначення інтенсивностей фонової складової кольору	–	300	–	–	–
Визначення інтенсивностей дифузної складової кольору	200	900	–	–	–
Визначення інтенсивностей спекулярної складової кольору	200	1700	–	–	–
Визначення інтенсивностей інтегральної складової кольору	600	–	–	–	–
Разом	3384	4100	1524	400	300

Під час зафарбовування стандартного трикутника з використанням кубічного полінома процесор Pentium M виконує таку кількість інструкцій, як у таблиці 4.5, за 8484 такти.

У результаті проведених досліджень отримано оцінки продуктивності формування графічних сцен а також оцінки реалістичності (NMSE).

Таблиця 4.5 – Склад і кількість інструкцій для зафарбовування стандартного трикутника з використанням кубічного полінома

Процедура	Кількість інструкцій			
	«+»	«х»	«:»	Зсув
Розрахунок I_1, I_2, I_3, I_4	96	192	–	–
Розрахунок A, B, C, D	108	36	48	264
Визначення інтенсивностей дифузної складової кольору внутрішніх точок поверхні, обмеженої трикутником	300	300	–	–
Разом	404	528	48	264

Експериментальні дослідження підтвердили достовірність отриманих наукових положень, а також показали високу ефективність розроблених засобів.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Для проведення технологічного аудиту з метою оцінювання комерційного потенціалу розробки було залучено 3-х незалежних експертів. Такими експертами будуть Войтко В. В. (Експерт 1), Бабюк Н. П. (Експерт 2) та Майданюк В. П. (Експерт 3). Оцінка комерційного потенціалу розробки проводиться за 12-ма критеріями за 5-ти бальною шкалою [35]. Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Бали, виставлені		
	експертом 1	експертом 2	експертом 3
1	4	4	4
2	3	3	3
3	3	4	3
4	4	3	3
5	3	3	4
6	4	4	3
7	3	4	3
8	4	4	4
9	4	4	4
10	4	4	4
11	4	4	4
12	4	4	4
Сума балів	$СБ_1$=44	$СБ_2$=45	$СБ_3$=43
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{СБ_1 + СБ_2 + СБ_3}{3} = 44$		

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу. Цей рівень досягається завдяки зростанню продуктивності, ефективності нової науково-технічної розробки.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи

Для розробки нового програмного продукту необхідні певні витрати.

Основна заробітна плата розробників визначається за формулою (5.1)

$$З_о = \frac{M}{T_P} \cdot t, \tag{5.1}$$

де M – місячний посадовий оклад конкретного розробника, T_P – кількість робочих днів у місяці (вважаємо, що $T_P = 21$ день), t – кількість днів роботи. Дана формула використовується для знаходження об’єму витрат на оплату праці, результати обчислень наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
Науковий керівник	12000	571,43	10	5714,29
Інженер-програміст	6000	285,71	45	12857,14
Всього				18571,43

Додаткові витрати на оплату праці становлять 10% від основних витрат на оплату праці, тобто

$$З_{дод} = 0{,}1 \cdot 18571{,}43 = 1857{,}14 \text{ (грн.)}.$$

Єдиний соціальний внесок розраховується як 22% від суми основної та додаткової заробітної плати. Сума нарахувань становить

$$Н_{зп} = 0{,}22 \cdot (18571{,}43 + 1857{,}14) = 4494{,}29 \text{ (грн.)}.$$

Витрати на комплектуючі вироби $К_в$, які використовують при дослідженні нового технічного рішення, розраховуються, згідно з їхньою номенклатурою, за формулою

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j,$$

де $Н_j$ – кількість комплектуючих j -го виду, шт.; $Ц_j$ – покупна ціна комплектуючих j -го виду, грн.; $К_j$ – коефіцієнт транспортних витрат, (встановимо однаковим для всіх комплектуючих та рівним 1,3). Отримані розрахунки зведено до таблиці 5.3.

Таблиця 5.3 – Витрати на комплектуючі

Найменування	Кількість, шт.	Ціна за штуку, грн.	Сума, грн.
Флешка	1	100	100
Пачка паперу	2	125	250
Ручка	1	25	25
Всього з урахуванням транспортних витрат			487,50

До статті «Амортизація обладнання» відносять амортизаційні відрахування по кожному виду обладнання для проведення науково-дослідної роботи, за його наявності в дослідній організації або на підприємстві.

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

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12},$$

де $Ц_б$ – балансова вартість обладнання, яке використовувалось для проведення досліджень, грн.; $t_{вик}$ – термін використання обладнання під час досліджень, місяців; $T_в$ – строк корисного використання обладнання, років.

Дослідження проводились на переносному персональному комп'ютері, вихідні дані для якого $Ц_б = 12000$ грн., $T_в = 2$ роки [35], $t_{вик} = 3$ місяці, тому амортизаційні відрахування становлять

$$A_{обл} = \frac{12000}{2} \cdot \frac{3}{12} = 1560{,}50 \text{ (грн.)}.$$

Витрати на силову електроенергію $B_е$ розрахуємо за формулою:

$$B_е = W_у \cdot t \cdot Ц_е \cdot К_{вп},$$

де $W_у$ – встановлена потужність обладнання на певному етапі розробки, кВт; t – тривалість роботи обладнання на етапі дослідження, год; $Ц_е$ – вартість 1 кВт-години електроенергії, грн.; $К_{вп}$ – коефіцієнт, що враховує використання

потужності, $K_{вп} < 1$. Вихідні дані: $W_y = 0{,}065$ кВт; $t = 720$ год; $Ц_е = 1{,}68$ грн./кВт; $K_{вп} = 0{,}8 < 1$. Витрати на силову електроенергію становлять

$$B_е = 0{,}065 \cdot 720 \cdot 1{,}68 \cdot 0{,}8 = 62{,}90 \text{ (грн.)}.$$

Інші витрати $I_в$ – витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Вони розраховуються як 50% від суми основної заробітної плати розробників і становлять

$$I_в = 0.5 \cdot 18571{,}43 = 9285{,}72 \text{ (грн.)}.$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат і становлять

$$B_{заг} = 18571{,}43 + 1857{,}14 + 4494{,}29 + 487{,}50 + 1560{,}50 + 62{,}90 + 9285{,}72 = 36319{,}48$$

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

$$ЗВ = \frac{B_{заг}}{\eta},$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. На стадії впровадження $\eta = 0{,}9$. Загальні витрати становитимуть

$$ЗВ = \frac{36319{,}48}{0{,}9} = 40354{,}98 \text{ (грн.)}.$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Можливе збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П_i = (\Delta П_я \cdot N + П_я \cdot \Delta N)_i,$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження на підприємстві результатів науково-технічної розробки в аналізованому році; N – основний кількісний показник, який визначає обсяг діяльності підприємства у році до впровадження результатів нової науково-технічної розробки; $П_я$ – основний якісний показник, який визначає результати діяльності підприємства у кожному із років після впровадження науково-технічної розробки; ΔN – зміна основного кількісного показника діяльності підприємства в результаті впровадження науково-технічної розробки в аналізованому році [35].

У результаті впровадження результатів наукової розробки витрати на виготовлення програмного продукту зменшаться на 50 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 50 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 500 користувачів, протягом другого року – на 350 користувачів, протягом третього року – на 200 користувачів. Реалізація програмного продукту до впровадження результатів наукової розробки складала 3000 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 600 грн. Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Протягом перших трьох років:

$$\Delta П_1 = 50 \cdot 3000 + (600 + 50) \cdot 500 = 475000,$$
$$\Delta П_2 = 50 \cdot 3000 + (600 + 50) \cdot (500 + 350) = 702500,$$
$$\Delta П_3 = 50 \cdot 3000 + (600 + 50) \cdot (500 + 350 + 200) = 832500.$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їхньої окупності

Розрахуємо приведену вартість збільшення всіх чистих прибутків $ПП$, що їх може отримати розробник (замовник) від можливого впровадження науково-технічної розробки на власному підприємстві:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t},$$

де $\Delta П_i$ – 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 науково-технічної розробки, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні (встановимо $\tau = 0{,}05$); t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання підприємством збільшеної величини чистого прибутку в аналізованому році.

$$ПП = \frac{40354{,}98}{(1+0{,}05)^0} + \frac{475000}{(1+0{,}05)^1} + \frac{702500}{(1+0{,}05)^2} + \frac{832500}{(1+0{,}05)^3} = 1849068{,}94 \text{ (грн.)}.$$

Рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, як на рисунку 5.1.

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Далі розраховують величину початкових інвестицій PV, які розробник (замовник) має вкласти для здійснення науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{розр} \cdot ЗВ,$$

де $k_{розр}$ – коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково-технічної розробки (візьмемо $k_{розр} = 2$); $ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 40354,98 = 80709,96 \text{ (грн.)}.$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для розробника (замовника) від можливого впровадження науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV,$$

де $ПП$ – приведена вартість збільшення всіх чистих прибутків від можливого впровадження науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

$$E_{абс} = 1849068{,}94 - 80709{,}96 = 1768358{,}98 \text{ (грн.)}.$$

Величина $E_{абс}$ має велике додатне значення. Це може свідчити про потенційну доцільність у впровадженні цієї науково-технічної розробки. Але для остаточного прийняття рішення про впровадження цього недостатньо.

Для остаточного прийняття рішення в такому випадку необхідно розрахувати внутрішню економічну дохідність $E_в$ Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені розробником (замовником) у впровадження науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн.; PV – теперішня вартість початкових інвестицій, грн.; $T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[3]{1 + \frac{1768358,98}{80709,96}} - 1 = 1,84 \text{ або } 180\%.$$

Далі визначають бар'єрну ставку дисконтування $\tau_{мін}$. Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках (в 2021 році в Україні $d = 0,9$); f – показник, що характеризує ризикованість вкладення інвестицій (використаємо $f = 0,05$).

$$\tau_{мін} = 0,9 + 0,05 = 0,95 \text{ або } 95\%.$$

Оскільки величина $E_в > \tau_{мін}$, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки.

Далі розраховуємо період окупності інвестицій $T_{ок}$:

$$T_{ок} = \frac{1}{E_в},$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{1,84} = 0,54 \text{ роки}.$$

Оскільки $T_{ок}$ < 3-х років, то це свідчить про економічну ефективність впровадження науково-технічної розробки її розробником (замовником).

ВИСНОВКИ

Проведено аналіз методів і засобів зафарбовування для задач формування тривимірних графічних схем. Обґрунтовано необхідність підвищення продуктивності та реалістичності рендерингу.

Запропоновано метод зафарбовування, особливість якого полягає у вилучені з обчислювального процесу нормалізації векторів і розрахунку арккосинуса кута в циклі підготування. що дозволило значно зменшити час зафарбовування середньостатистичного трикутника порівняно з класичною реалізацією.

Подальшого розвитку отримав метод прискореного визначення дифузної та спекулярної складових кольору, особливість якого полягає в одночасному й незалежному визначенні інтенсивностей кольору відразу двох точок рядка растеризації. Обчислювальний процес, який реалізовано згідно з запропонованим підходу, не містить у циклі розрахунку складових кольору операцій ділення та знаходження квадратного кореня, які мають місце при класичній реалізації Це дозволило підвищити продуктивність та реалістичність рендерингу.

Запропоновано метод зафарбовування, особливість якого полягає у використанні поверхні другого прядку для визначення інтенсивностей кольору точок поверхні. Це дозволило суттєво підвищити продуктивність рендерингу порівняно з методом Фонга.

Розроблено алгоритми та програмні засоби для підвищення продуктивності формування графічних сцен.

Отримані в магістерській роботі наукові та практичні результати можна використати для побудови високопродуктивних засобів рендерингу в комп’ютерних системах візуалізації тривимірних зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pharr M., Jakob W., Humphreys, G. Physically Based Rendering: From Theory to Implementation: 3rd ed. Burlington : Morgan Kaufmann, 2016. 1266 p.

2. 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 – 2021*: матеріали XIV міжнародної науково-практичної конференції (м. Одеса, 21-22 жовтня 2021 р.). Одеса : ОНАХТ, 2021. С. 330-331.

3. Романюк О. Н., Дудник О. О., Озерчук Д. А. Підвищення продуктивності зафарбовування при використанні для визначення інтенсивностей кольору поверхні другого порядку. *Інформаційні технології та комп'ютерна інженерія*. Вінниця : ВНТУ, 2021. №2. С. 51-59.

4. Романюк О. Н., Дудник О. О., Романюк О. В., Озерчук Д. А. Метод перспективно-коректного текстурування. *Інформаційні технології та комп'ютерна інженерія*. Вінниця : ВНТУ, 2021. №1. С. 55-63.

5. Романюк О. Н., Озерчук Д. А., Котлик С. В., Романюк О. В. Розпаралелення обчислювального процесу при використанні спарок відеокарт в комп'ютерних іграх. *Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації*: матеріали І Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів (м. Одеса, 25-26 березня 2021 р.). Одеса : ОНАХТ, 2021. С 65-67.

6. Романюк О. Н., Мельник О. В., Романюк С. О., Озерчук Д. А. Особливості формування еліпсів, повернутих на заданий кут, на гексагональному растрі. *Наукові праці ДонНТУ, серія "Інформатика, кібернетика та обчислювальна техніка"*. Покровськ : ДонНТУ, 2020. № 2(31). С. 23-29.

7. Романюк А. Н., Вяткин С. И., Михайлов П. И., Чехместрук Р. Ю, Озерчук Д. А. Методы построения разверток и поверхностей. *Наукові праці*

ДонНТУ, серія "Інформатика, кібернетика та обчислювальна техніка". Покровськ : ДонНТУ, 2020. № 2(31). С.39-45.

8. Денисюк А. В., Озерчук Д.А., Романюк О.В., Романюк О.Н. Порівняння зображень із використанням перцептивних хеш-методів. *Електронні інформаційні ресурси: створення, використання, доступ*: збірник матеріалів Міжнародної науково-практичної Інтернет конференції (м. Суми, м. Вінниця, 9-10 листопада 2020 р.). Суми/Вінниця : НІКО/ВНТУ, 2020. С. 80-84.

9. Озерчук Д.А., Романюк С.О., Романюк О.Н. Методи формування тривимірних моделей обличчя на основі відповідного растрового зображення. *Електронні інформаційні ресурси: створення, використання, доступ*: збірник матеріалів Міжнародної науково-практичної Інтернет конференції (м. Суми, м. Вінниця, 9-10 листопада 2020 р.). Суми/Вінниця : НІКО/ВНТУ, 2020. С. 198-199.

10. Озерчук Д. А., Романюк О. Н., Ціхановська О. М. Аналіз найпоширеніших пакетів прикладних програм для економістів. *Електронні інформаційні ресурси: створення, використання, доступ*: збірник матеріалів Міжнародної науково-практичної Інтернет конференції (м. Суми, м. Вінниця, 9-10 листопада 2020 р.). Суми/Вінниця : НІКО/ВНТУ, 2020. С. 200-202.

11. Vyatkin, S. I., Ozerchuk, D. A., Romanyuk, O. N., Khoshaba, O. M. A modified method of elastic graph matching based on the gabor wavelets. *Електронні інформаційні ресурси: створення, використання, доступ*: збірник матеріалів Міжнародної науково-практичної Інтернет конференції (м. Суми, м. Вінниця, 9-10 листопада 2020 р.). Суми/Вінниця : НІКО/ВНТУ, 2020. С. 67-70.

12. Журавчак Л. М., Левченко О. М. Програмування комп'ютерної графіки та мультимедійні засоби. Львів : Видавництво Львівської політехніки, 2019. 143 с.

13. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering: 4th ed. Natick : A K Peters, 2018. 1198 p.

14. Birn J. Digital Lighting & Rendering: 3rd ed. Indianapolis : New Riders Pub, 2013. 453 p.

15. Hearn D., Baker M., Carithers W. Computer Graphics with Open GL. London : Pearson, 2010. 888 p.

16. Gomes J., Velho L., Sousa M.-C. Computer Graphics: Theory and Practice. Natick : A K Peters, 2012. 560 p.

17. Dinur E. The Complete Guide to Photorealism for Visual Effects, Visualization and Games. Milton Park : Routledge, 2021. 316 p.

18. Романюк О.Н. Комп'ютерна графіка : навчальний посібник. Вінниця : ВДТУ, 2001. 130 с.

19. Романюк О. Н. Метод прискореного зафарбовування тривимірних поверхонь з урахуванням їх локальної кривизни. *Вісник Східноукраїнського національного університету.* Луганськ : СНУ, 2008. № 12 (130). С. 166-172.

20. Пічугін М.Ф. Комп'ютерна графіка : навчальний посібник. Київ : Центр навчальної літератури, 2019. 346 с.

21. Lengyel E. Foundations of Game Engine Development, Volume 2: Rendering. Lincoln : Terathon Software, 2019. 412 p.

22. Боресков А. В. Программирование компьютерной графики. М. : ДМК Пресс, 2019. 372 с.

23. Buck J. The Ray Tracer Challenge: A Test-Driven Guide to Your First 3D Renderer. Raleigh, North Carolina, USA : Pragmatic Bookshelf, 2019. 292 p.

24. Романюк О. Н. Ефективна модель для відтворення спекулярної складової кольору. *Проблеми інформатизації та управління*: збірник наукових праць. Київ : НАУ, 2007. Випуск 2 (20). С. 115-120.

25. Роджерс Д., Адамс Дж. Математические основы машинной графики : пер. с англ. М. : Мир, 2001. 604 с.

26. Romanyuk O. N., Hast A. A method for accelerated computation of color intensities for shading of three-dimensional graphics objects. *Współczesne problemy informatyki. Algorytmy i modelowanie.* Legnica : Wydawnictwo Wyższej Szkoły Menedżerskiej, 2007. P. 213-227.

27. Marschner S., Shirley P. Fundamentals of Computer Graphics: 5th ed. Natick : A K Peters, 2021. 700 p.

28. Lengyel E. Foundations of Game Engine Development, Volume 1: Mathematics. Lincoln : Terathon Software, 2016. 200 p.

29. Kosarevsky S., Latypov V. 3D Graphics Rendering Cookbook: A comprehensive guide to exploring rendering algorithms in modern OpenGL and Vulkan. Birmingham : Packt Publishing, 2021. 670 p.

30. Романюк О. Н. Високопродуктивні методи та засоби зафарбовування тривимірних графічних об’єктів: монографія. Вінниця : УНІВЕСУМ-Вінниця, 2006. 190 с.

31. Fog A. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs. Kongens Lingby : Technical University of Denmark, 2021. 442 p.

32. Rogers D. F. Procedural Elements For Computer Graphics : 2nd ed. New York : McGraw Hill, 2010. 752 p.

33. Романюк О. Н., Шаманський А. А. Метод зафарбовування тривимірних графічних об’єктів без нормалізації векторів нормалей. *Інформаційні технології та комп'ютерна інженерія*. Вінниця : ВНТУ, 2006. № 2 (6), С. 111-115.

34. Мацуда К., Ли Р. WebGL: программирование трехмерной графики. М. : ДМК Пресс, 2018. 494 с.

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

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
« 13 » вересня 2021 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Методи та програмно-апаратні засоби рендерингу
тривимірних графічних схем»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
д.т.н., проф. О.Н. Романюк
"____"_____2021 р.
Виконав:
студент гр.2ПІ-20м Д. А. Озерчук
"____"_____2021 р.

Вінниця – 2021 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи та програмно-апаратні засоби рендерингу тривимірних графічних схем».

Галузь застосування – системи комп’ютерної графіки.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ ректора по ВНТУ № 277 від « 24 » вересня 2021 р. про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності та реалістичності формування тривимірних графічних зображень за рахунок розробки нових методів і засобів.

Призначення роботи – розробка методів і засобів зафарбовування тривимірних графічних об’єктів.

3 Вихідні дані для проведення МКР

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

1. Романюк О. Н. Високопродуктивні методи та засоби зафарбовування тривимірних графічних об’єктів: монографія. Вінниця : УНІВЕСУМ-Вінниця, 2006. 190 с.

2. Романюк О.Н. Комп'ютерна графіка : навчальний посібник. Вінниця : ВДТУ, 2001. 130 с.

3. Romanyuk O. N. A method for accelerated computation of color intensities for shading of three-dimensional graphics objects. *Współczesne problemy informatyki. Algorytmy i modelowanie.* Legnica : Wydawnictwo Wyższej Szkoły Menedżerskiej, 2007. P. 213-227.

4. Романюк О. Н. Ефективна модель для відтворення спекулярної складової кольору. *Проблеми інформатизації та управління*: збірник наукових праць. Київ : НАУ, 2007. Випуск 2 (20). С. 115-120.

5. Романюк О. Н. Метод прискореного зафарбовування тривимірних поверхонь з урахуванням їх локальної кривизни. *Вісник Східноукраїнського національного університету.* Луганськ : СНУ, 2008. № 12 (130). С. 166-172.

4. Технічні вимоги

Базові методи зафарбовування – методи Гуро, Фонга; режим – TrueColor; вихідні дані для зафарбовування – вектори нормалей до вершин трикутників; серединний вектор $\vec{H}$; дані про розташування джерела світла та спостерігача; коефіцієнт спекулярності поверхні; координати вершин трикутників; максимальна розрядність для задання адрес вершин трикутників – 12; вихідні дані – кінцеве зображення, зафарбоване згідно методів Фонга та Гуро

5. 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.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації та ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз методів та програмно-апаратних засобів рендерингу тривимірних графічних схем	15.09.2021-26.09.2021
2	Розробка методів обчислення інтенсивності кольору в точці з використанням поліномів	27.092021-15.10.2021
3	Розробка методів прискореного обчислення інтенсивності кольору в точці	16.10.2021-7.11.2021
4	Програмна реалізація методів формування тривимірних графічних схем. Експериментальні дослідження	8.11.2021-21.11.2021
5	Економічна частина	22.11.2021-30.11.2021

10. 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 РОБОТИ

Назва роботи: **Методи та програмно-апаратні засоби рендерингу тривимірних графічних схем**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 20м

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

Unicheck	
Оригінальність	**90,9 %**
Схожість	9,1 %

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 «Методи та програмно-апаратні засоби рендерингу тривимірних графічних схем».

Автор _____ Озерчук Дмитро Анатолійович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

ДОДАТОК В

ЛІСТИНГ КОДУ

В.1 Лістинг коду прототипу програми

В.1.1 Файл Prototype.html

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Прототип програми формування тривимірних графічних схем з використанням нових та модифікованих методів зафарбовування</title>
<!-- Опис методу Фонга -->
<script id="p-vertex-shader" type="x-shader/x-vertex">
//шейдер дифузної та амбієнтної складових кольору
//вхідні дані
attribute vec4 vPosition;
attribute vec3 vNormal;
//вихідні дані
varying vec4 mvPosition; // позиція вершини
varying vec3 iN; // нормаль до поверхні
//глобальні константи
uniform mat4 p;     // матриця перспективи
uniform mat4 mv;    // матриця камери
void main()
{
```

```
  iN = normalize((mv*vec4(vNormal,0.0)).xyz); //інтерполяція нормалей
  mvPosition = mv*vPosition;  //інтерполяція позицій
  gl_Position = p*mvPosition; //координати на екрані
}
</script>
<script id="p-fragment-shader" type="x-shader/x-fragment">
precision mediump float;
//структури
struct _light
{
	vec3 diffuse;
	vec3 specular;
	vec3 ambient;
	vec4 position;
};
struct _material
{
	vec3 diffuse;
	vec3 specular;
	vec3 ambient;
	float shininess;
};
//константи
const int nLights = 1; // кількість джерел світла
const float fix = 255.0;
```

```
//вхідні дані
varying vec4 mvPosition; // позиція вершини
varying vec3 iN; // інтерпольована нормаль до поверхні
//глобальні змінні
vec3 N;
//глобальні константи
uniform bool lighting;  // ввімкнути вимкнути освітлення
uniform vec3 uColor;    // колір при вимкненому освітленні
uniform _light light[nLights]; // властивості n джерел світла
uniform _material material; // властивості матеріалів
//прототипи (класи JS)
vec3 lightCalc(in _light light);
varying vec3 color;
void main()
{
 if (lighting == false)
 {
        gl_FragColor.rgb = uColor;
  gl_FragColor.a = 1.0;
 }
 else
 {
        //Нормалізувати вектор нормалі
        N = normalize(iN);
        //Поєднати кольори всіх джерел світла
```

```
        gl_FragColor = vec4(0,0,0,1);
        for (int i = 0; i < nLights; i++)
        {
            gl_FragColor.rgb += lightCalc(light[i]);
        }
  }
}
vec3 lightCalc(in _light light)
{
  //Встановити напрямок вектора до джерела світла
  vec3 L;
  //Якщо позиція джерела вже є вектором – використати його
  if (light.position.w == 0.0)
        L = normalize(light.position.xyz);
  //Інакше встановити вектор від вершини до джерела
  else
        L = normalize(light.position.xyz - mvPosition.xyz);
  //Встановити вектор спостерігача
  vec3 E = -normalize(mvPosition.xyz);
  vec3 H = normalize(L+E);
  float Ks = pow(max(dot(N, H),0.0), material.shininess);
  //Обчислити коефіцієнт розсіювання (дифузний)
  float Kd = max(dot(L,N), 0.0);
  //Обчислити колір світла
  vec3 color =  Ks * material.specular * light.specular
```

```
                                + Kd * material.diffuse * light.diffuse
                                        + material.ambient * light.ambient;
  //привести кольори до промжку [0,255] з проміжку [0,1]
  color = color/fix/fix;
  return color;
}
</script>
<!-- Кінець опису методу Фонга -->
<!-- Опис методу Гуро -->
<script id="g-vertex-shader" type="x-shader/x-vertex">
//шейдер дифузної та амбієнтної складових кольору
//вхідні дані
attribute vec4 vPosition;
attribute vec3 vNormal;
//вихідні дані
varying vec4 color;
//структури
struct _light
{
        vec3 diffuse;
        vec3 specular;
        vec3 ambient;
        vec4 position;
};
struct _material
```

```
{
	vec3 diffuse;
	vec3 specular;
	vec3 ambient;
	float shininess;
};
//константи
const int nLights = 1; // кількість джерел світла
const float fix = 255.0;
//глобальні константи
uniform mat4 p;     // матриця перспективи
uniform mat4 mv;    // матриця камери
uniform bool lighting;  // ввімкнути вимкнути освітлення
uniform vec3 uColor;    // колір при вимкненому освітленні
uniform _light light[nLights]; // властивості n джерел світла
uniform _material material; // властивості матеріалів
//глобальні змінні
vec4 mvPosition; // позиція вершини
vec3 N; // фіксована нормаль поверхні
//прототипи (класи JS)
vec3 lightCalc(in _light light);
void main()
{
  //Перетворити точку
  mvPosition = mv*vPosition;  //mvPosition is використовується часто
```

```
 gl_Position = p*mvPosition;
 if (lighting == false)
 {
        color.rgb = uColor;
  color.a = 1.0;
 }
 else
 {
        //Нормалізувати вектор нормалі,
        // виділити важливі координати
        N = normalize((mv*vec4(vNormal,0.0)).xyz);
        //Combine colors from all lights
        color = vec4(0,0,0,1);
        for (int i = 0; i < nLights; i++)
        {
              color.rgb += lightCalc(light[i]);
        }
 }
}
vec3 lightCalc(in _light light)
{
 //Встановити позиції джерел світла
 vec3 L;
 //Якщо позиція – вектор, то використати його
 if (light.position.w == 0.0)
```

```
        L = normalize(light.position.xyz);
  //Інакше встановити вектор від вершини до джерела
  else
        L = normalize(light.position.xyz - mvPosition.xyz);
  //Встановити вектор спостерігача
  vec3 E = -normalize(mvPosition.xyz);
  vec3 H = normalize(L+E);
  float Ks = pow(max(dot(N, H),0.0), material.shininess);
  //Обчислити коефіцієнт розсіювання (дифузний)
  float Kd = max(dot(L,N), 0.0);
  // Обчислити колір світла
  vec3 color =  Ks * material.specular * light.specular
                              + Kd * material.diffuse * light.diffuse
                                    + material.ambient * light.ambient;
  //привести колір до вигляду [0,255] замість [0,1]
  color = color/fix/fix;
  return color;
}
</script>
<script id="g-fragment-shader" type="x-shader/x-fragment">
precision mediump float;
varying vec4 color;
void main()
{
  gl_FragColor = color;
```

```
        }
        </script>
        <!-- Кінець опису методу Фонга -->
        <!-- Зовнішні файли JS -->
        <script type="text/javascript" src="./Prototype_files/webgl-utils.js"></script>
        <script type="text/javascript" src="./Prototype_files/initShaders.js"></script>
        <script type="text/javascript" src="./Prototype_files/MVnew.js"></script>
        <script type="text/javascript"
src="./Prototype_files/uofrGraphics.js"></script>
        <script type="text/javascript" src="./Prototype_files/j3di.js"></script>
        <script type="text/javascript" src="./Prototype_files/jscolor.js"></script>
        <script type="text/javascript" src="./Prototype_files/renderer.js"></script>
        <style>
        .center {
          margin: auto;
          width: 80%;
        }
        </style>
        </head>
        <body>
                <div class="center">
            <!-- canvas – єдиний елемент, що може формувати графіку з
використанням WebGL. -->
            <canvas id="gl-canvas" width="1000" height="400">
               Вибачте, схоже ваш переглядач не підтримує елемент canvas
            </canvas>
```

```
<table class="center">
    <tbody><tr>
        <td>Light dir:</td>
        <td> 0</td>
        <td><input id="rotY" type="range" min="0" max="360" step="0.1" value="90" style="width: 650px;"></td>
        <td>360|</td>
        <td id="rotYval">90</td>
    </tr>
    <tr>
        <td>Shininess:</td>
        <td> 1</td>
        <td><input id="shininess" type="range" min="1" max="255" step="1" value="50" style="width: 650px;"></td>
        <td>255|</td>
        <td id="shininessval">50</td>
    </tr>
    <tr>
        <td>Sphere Rez:</td>
        <td>5</td>
        <td><input id="rez" type="range" min="3" max="100" step="1" value="50" style="width: 650px;"></td>
        <td>100|</td>
        <td id="rezval">50</td>
    </tr>
</tbody></table>
```

```
<div class="center">

<button id="specularColor" class="jscolor {valueElement:&#39;chosen-value&#39;, onFineChange:&#39;setSpecularColor(this)&#39;}" style="width: 400px; background-image: none; background-color: rgb(255, 255, 255); color: rgb(0, 0, 0);">  Specular Color  </button>

<button id="diffuseColor" class="jscolor {valueElement:&#39;chosen-value&#39;, onFineChange:&#39;setDiffuseColor(this)&#39;}" style="width: 400px; background-image: none; background-color: rgb(64, 77, 90); color: rgb(255, 255, 255);">Diffuse and Ambient Color</button><br>

<input type="radio" name="toggle" value="phong">

Метод Фонга<br>

<input type="radio" name="toggle" value="gouraud" checked="">Метод Гуро<hr>

<input type="radio" name="test" value="fastDS">

Прискорене обчислення дифузної та спекулярної складових кольору<br>

<input type="radio" name="test" value="angleI">Зафарбовування з використанням кутової інтерполяції<br>

<input type="radio" name="test" value="quadric">

Модифікований метод зафарбовування з використанням поверхні другого порядку

<br>

<input type="radio" name="test" value="cubic" checked="">

Метод зафарбовування з використанням кубічного полінома

</div>

</div>

</body>

</html>
```

В.1.2 Файл renderer.js

```
//----------------------------------------------------------------------------
// Підготовка змінних
//----------------------------------------------------------------------------
// Контекст рендерингу WebGL
var gl;
var obj;
var red =        [255,   0,   0];
var green =      [  0, 255,   0];
var blue =       [  0,   0, 255];
var lightred =   [255, 128, 128];
var lightgreen = [128, 255, 128];
var lightblue =  [128, 128, 255];
var white =      [255, 255, 255];
var black =      [  0,   0,   0];
//Змінні для матриць перетворень
var mv = new mat4();
var p = new mat4();
var mvLoc, projLoc;
//Змінні для освітлення
var light;
var material;
var lighting;
var uColor;
```

```
// Потрібно переприв’язати дані
//та направити атрибути до них після виклику функцій малювання
var vertexBuffer, normalBuffer;
var program, phong, gouraud;
//ступінь поділу сфери
var rez = 50;
function bindBuffersToShader(obj) {
  gl.bindBuffer(gl.ARRAY_BUFFER, obj.vertexObject);
  gl.vertexAttribPointer(program.vPosition, 3, gl.FLOAT, gl.FALSE, 0, 0);
  gl.enableVertexAttribArray(program.vPosition);
  gl.bindBuffer(gl.ARRAY_BUFFER, obj.normalObject);
  gl.vertexAttribPointer(program.vNormal, 3, gl.FLOAT, gl.FALSE, 0, 0);
  gl.enableVertexAttribArray(program.vNormal);
  gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, obj.indexObject);
}

//----------------------------------------------------------------------------
// Ініціалізація функцій обробки подій
//----------------------------------------------------------------------------
window.onload = function init() {
  // Set up a WebGL Rendering Context in an HTML5 Canvas
  var canvas = document.getElementById("gl-canvas");
  gl = WebGLUtils.setupWebGL(canvas);
  if (!gl) {
    alert("WebGL isn't available");
```

```
}
//  Налаштувати WebGL
//  встановити колір порожньої області
//   увімкнути тестування глибини
gl.clearColor(1.0, 1.0, 1.0, 1.0);
gl.enable(gl.DEPTH_TEST);
//  Завантажити шейдери та ініціалізувати буфери атрибутів
gouraud = initShaders(gl, "g-vertex-shader", "g-fragment-shader");
phong = initShaders(gl, "p-vertex-shader", "p-fragment-shader");
      program = gouraud;
gl.useProgram(program);
// Встановити дані для малювання
// глобально
gl.clearColor(0.5, 0.5, 0.5, 1);
// Завантажити дані добуферів данихв GPU
// Поєднати атрибути шейдерів та відповідні буфери даних
//***Вершини***
program.vPosition = gl.getAttribLocation(program, "vPosition");
gl.enableVertexAttribArray(program.vPosition);
//***Нормалі***
program.vNormal = gl.getAttribLocation(program, "vNormal");
gl.enableVertexAttribArray(program.vNormal);
//налаштування графіки
// залежить від масивів позицій та нормалей
urgl = new uofrGraphics();
```

```
urgl.connectShader(program, "vPosition", "vNormal", "stub");
//Налаштувати видиму область
gl.viewportWidth = canvas.width;
gl.viewportHeight = canvas.height;
gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
      //Встановити початковий стан
light = []; // глобальний масив розташування властивостей джерел світла
      light.nLights = 1; // кількість джерел – відповідно до шейдера
      for (var i = 0; i < light.nLights; i++)
      {
            light[i] = {};
  if (i == 0)
            {
                  light[i].diffuseColor = white;
                  light[i].specularColor = white;
                  light[i].ambientColor = vec3(50,50,50);
                  light[i].positionVal = vec4(3,3,0,1);
    light[i].rotY = 90;
  }
            else //відключити інші джерела
  {
                  light[i].diffuseColor = black;
                  light[i].specularColor = black;
                  light[i].ambientColor = black;
                  light[i].positionVal = vec4(0,0,10,1);
```

```
                }
            }
        material = {};
        material.diffuseColor = vec3(64, 77, 90);
            material.specularColor = white;
            material.shininessValue = 50.0;
            getAndSetShaderLocations();
        // ** Налаштувати обробники подій
        // Повзунок кута падіння світла через поворот rotY (direction)
        var el = document.getElementById("rotY");
            el.onchange = el.oninput = function(event)
        {
          light[0].rotY=(event.srcElement || event.target).value;
          document.getElementById("rotYval").innerHTML = light[0].rotY;
        };
        document.getElementById("rotY").value = 90;
        document.getElementById("rotYval").innerHTML = 90;
        // Повзунок спекулярної компоненти
        var el = document.getElementById("shininess")
            el.onchange = el.oninput = function (event) {
          material.shininessValue = (event.srcElement || event.target).value;
          document.getElementById("shininessval").innerHTML =
material.shininessValue;
          gl.uniform1f(material.shininess, material.shininessValue);
        };
        document.getElementById("shininess").value = 50;
```

```
        document.getElementById("shininessval").innerHTML = 50;
        // Повзунок рівня деталізації сфери
        var el = document.getElementById("rez");
            el.onchange = el.oninput = function (event) {
          rez = (event.srcElement || event.target).value;
          document.getElementById("rezval").innerHTML = rez;
        };
        document.getElementById("rez").value = rez;
        document.getElementById("rezval").innerHTML = rez;
            // Phong/Gouraud radio buttons
            var toggleOptions = document.getElementsByName("toggle");
            for (var i = 0; i < toggleOptions.length; i++)
            {
                    toggleOptions[i].onchange = setShadeMode;
            }
        // Колір дифузної компоненти – початкове значення
            var dc = material.diffuseColor;
        document.getElementById("diffuseColor").jscolor.fromRGB(dc[0], dc[1],
dc[2]);
        // Колір спекулярної компоненти – початкове значення
        document.getElementById("specularColor").jscolor.fromRGB(255, 255,
255);
        requestAnimFrame(render);
      };
      function getAndSetShaderLocations(){
        //***Вершини***
```

```
        program.vPosition = gl.getAttribLocation(program, "vPosition");
        gl.enableVertexAttribArray(program.vPosition);
        //***Нормалі***
        program.vNormal = gl.getAttribLocation(program, "vNormal");
        gl.enableVertexAttribArray(program.vNormal);
      // Отримати адреси глобальних констант перетворення
        projLoc = gl.getUniformLocation(program, "p");
        mvLoc = gl.getUniformLocation(program, "mv");
        //Set up projection matrix
        p = perspective(45.0, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0);
        gl.uniformMatrix4fv(projLoc, gl.FALSE, flatten(p));
        // Отримати та встановити глобальні параметри освітлення
        for (var i = 0; i < light.nLights; i++) {
          light[i].diffuse = gl.getUniformLocation(program, "light[" + i +
"].diffuse");
          light[i].specular = gl.getUniformLocation(program, "light[" + i +
"].specular");
          light[i].ambient = gl.getUniformLocation(program, "light[" + i +
"].ambient");
          light[i].position = gl.getUniformLocation(program, "light[" + i +
"].position");
                gl.uniform3fv(light[i].diffuse, light[i].diffuseColor);
                gl.uniform3fv(light[i].specular, light[i].specularColor);
                gl.uniform3fv(light[i].ambient, light[i].ambientColor);
                gl.uniform4fv(light[i].position, light[i].positionVal);
        }
        // Отримати та встановити глобальні параметри матеріалів
```

```
  material.diffuse = gl.getUniformLocation(program, "material.diffuse");
  material.specular = gl.getUniformLocation(program, "material.specular");
  material.ambient = gl.getUniformLocation(program, "material.ambient");
  material.shininess = gl.getUniformLocation(program, "material.shininess");
  gl.uniform3fv(material.diffuse, material.diffuseColor);
  gl.uniform3fv(material.specular, material.specularColor);
  gl.uniform3fv(material.ambient, material.diffuseColor);
  gl.uniform1f(material.shininess, material.shininessValue);
  // Отримати та встановити інші стани шейдера
  lighting = gl.getUniformLocation(program, "lighting");
  uColor = gl.getUniformLocation(program, "uColor");
  gl.uniform1i(lighting, 1);
  gl.uniform3fv(uColor, white);
}
function setShadeMode(event)
{
   var type =(event.srcElement || event.target);
   if (type.value == "phong")
            program = phong;
      else
            program = gouraud;
      gl.useProgram(program);
      getAndSetShaderLocations();
      urgl.connectShader(program, "vPosition", "vNormal", "stub");
};
```

```
// Обробник подій для вибору кольору спекулярної компоненти
function setSpecularColor(picker) {
        material.specularColor = picker.rgb;
  gl.uniform3fv(material.specular, material.specularColor);
}
// Обробник подій для вибору кольору дифузної компоненти

function setDiffuseColor(picker) {
        material.diffuseColor = picker.rgb;
        material.ambientColor = picker.rgb;
  gl.uniform3fv(material.diffuse, material.diffuseColor); //not sending alpha, will be ignored by shader anyway
  gl.uniform3fv(material.ambient, material.ambientColor);
}
//----------------------------------------------------------------------------
// Функція обробки події «рендеринг»
//----------------------------------------------------------------------------
var rx = 0,  ry = 0;
function render()
{
  gl.clear(gl.DEPTH_BUFFER_BIT | gl.COLOR_BUFFER_BIT);
  //Встановити початкові параметри огляду
  var eye = vec3(0.0, 1.0, 4.0);
  var at = vec3(0.0, 0.0, 0.0);
  var up = vec3(0.0, 1.0, 0.0);
  mv = lookAt(eye, at, up);
```

```
        //Розташувати джерела світла

        gl.uniform4fv(light[0].position,
            mult(transpose(mv), mult(rotateY(light[0].rotY),vec4( 1,1,0,0))));

        // переприв’язати буфери до шейдера

        if (obj.loaded)
        {
          bindBuffersToShader(obj);
          //draw obj
          var objTF = mult(mv, mult(translate(0, -1, 0), rotateY(ry)));
          gl.uniformMatrix4fv(mvLoc, gl.FALSE, flatten(objTF));
          gl.drawElements(gl.TRIANGLES, obj.numIndices,
gl.UNSIGNED_SHORT, 0);
        }
        var sphereTF = mult(mv, translate(-2, 0, 0));
        gl.uniformMatrix4fv(mvLoc, gl.FALSE, flatten(sphereTF));
        urgl.drawSolidSphere(1, rez, rez);
        sphereTF = mult(mv, translate(2, 0, 0));
        gl.uniformMatrix4fv(mvLoc, gl.FALSE, flatten(sphereTF));
        urgl.drawSolidSphere(1, rez, rez);
        ry += 0.5;
        requestAnimFrame(render);
      }
```

B.1.3 Файл renderer.js

```
function initShaders( gl, vertexShaderId, fragmentShaderId )
{
    var vertShdr;
    var fragShdr;

    var vertElem = document.getElementById( vertexShaderId );
    if ( !vertElem ) {
        alert( "Unable to load vertex shader " + vertexShaderId );
        return -1;
    }
    else {
        vertShdr = gl.createShader( gl.VERTEX_SHADER );
        gl.shaderSource( vertShdr, vertElem.text );
        gl.compileShader( vertShdr );
        if ( !gl.getShaderParameter(vertShdr, gl.COMPILE_STATUS) ) {
            var msg = "Vertex shader failed to compile.  The error log is:"
          + "<pre>" + gl.getShaderInfoLog( vertShdr ) + "</pre>";
            alert( msg );
            return -1;
        }
    }

    var fragElem = document.getElementById( fragmentShaderId );
```

```
if ( !fragElem ) {
   alert( "Unable to load vertex shader " + fragmentShaderId );
   return -1;
}
else {
   fragShdr = gl.createShader( gl.FRAGMENT_SHADER );
   gl.shaderSource( fragShdr, fragElem.text );
   gl.compileShader( fragShdr );
   if ( !gl.getShaderParameter(fragShdr, gl.COMPILE_STATUS) ) {
      var msg = "Fragment shader failed to compile.  The error log is:"
     + "<pre>" + gl.getShaderInfoLog( fragShdr ) + "</pre>";
      alert( msg );
      return -1;
   }
}

var program = gl.createProgram();
gl.attachShader( program, vertShdr );
gl.attachShader( program, fragShdr );
gl.linkProgram( program );

if ( !gl.getProgramParameter(program, gl.LINK_STATUS) ) {
   var msg = "Shader program failed to link.  The error log is:"
      + "<pre>" + gl.getProgramInfoLog( program ) + "</pre>";
   alert( msg );
```

```
            return -1;
        }
        return program;
    }
```

В.2 Лістинг коду програми для знаходження *NMSE*

```
    class Program
    {
    static void Main(string[] args)
    {
    long frequency = Stopwatch.Frequency;
    long nanosecPerTick = (1000000000L) / frequency;
    Console.WriteLine("Timer frequency in ticks per second = {0}, in
nanoseconds per tick = {1}", frequency, nanosecPerTick);
    Bitmap imgS = new Bitmap(@"D:\render_results\spheres\s.png");
    Bitmap imgC;
    Color pixel1, pixel2;
    long standard = 0;
    long comp = 0;
    int r, g, b, r_, g_, b_;
    double res;
    double ares=0;
    double maxres = double.MinValue;
    int loop = 5; // кількість зразків для порівняння
    long ms = 0;
```

```
long t = 0;
for (int k = 0; k < loop; k++)
{
        var watch = System.Diagnostics.Stopwatch.StartNew();
        imgC = new Bitmap("D:\\render_results\\spheres\\"+k+".png");
        for (int i = 0; i < imgS.Width; i++)
        {
                for (int j = 0; j < imgS.Height; j++)
                {
                        pixel1 = imgS.GetPixel(i, j);
                        r = pixel1.R;
                        g = pixel1.G;
                        b = pixel1.B;
                        pixel2 = imgC.GetPixel(i, j);
                        r_ = r - pixel2.R;
                        g_ = g - pixel2.G;
                        b_ = b - pixel2.B;
                        standard += r * r + g * g + b * b;
                        comp += r_ * r_ + g_ * g_ + b_ * b_;
                }
        }
        res = (double)comp / standard;
        ares += res;
        if (res > maxres) maxres = res;
        watch.Stop();
```

```
            var elapsedMs = watch.ElapsedMilliseconds;
            var elapsedT = watch.ElapsedTicks;
            ms += elapsedMs;
            t += elapsedT;
            watch.Reset();
            Console.WriteLine("Time = " + elapsedMs.ToString() +
                              " ms or " + elapsedT.ToString() + " ticks, " +
                              "NMSE = " + res.ToString("F6"));
        }
        Console.WriteLine("Total time = " +
TimeSpan.FromMilliseconds(ms).ToString() + " or " + t.ToString() + " ticks");
        Console.WriteLine("Average time = " + (ms/loop).ToString() +
                          " ms or " + (t/loop).ToString() + " ticks");
        Console.WriteLine("Max NMSE = " + maxres.ToString("F6") +
                          ", Average NMSE = " + (ares / loop).ToString("F6")); }}}
```

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОДИ ТА ПРОГРАМНО-АПАРАТНІ ЗАСОБИ РЕНДЕРИНГУ ТРИВИМІРНИХ ГРАФІЧНИХ СХЕМ

Рисунок Г.1 – Слайд презентації №1

Рисунок Г.2 – Слайд презентації №2

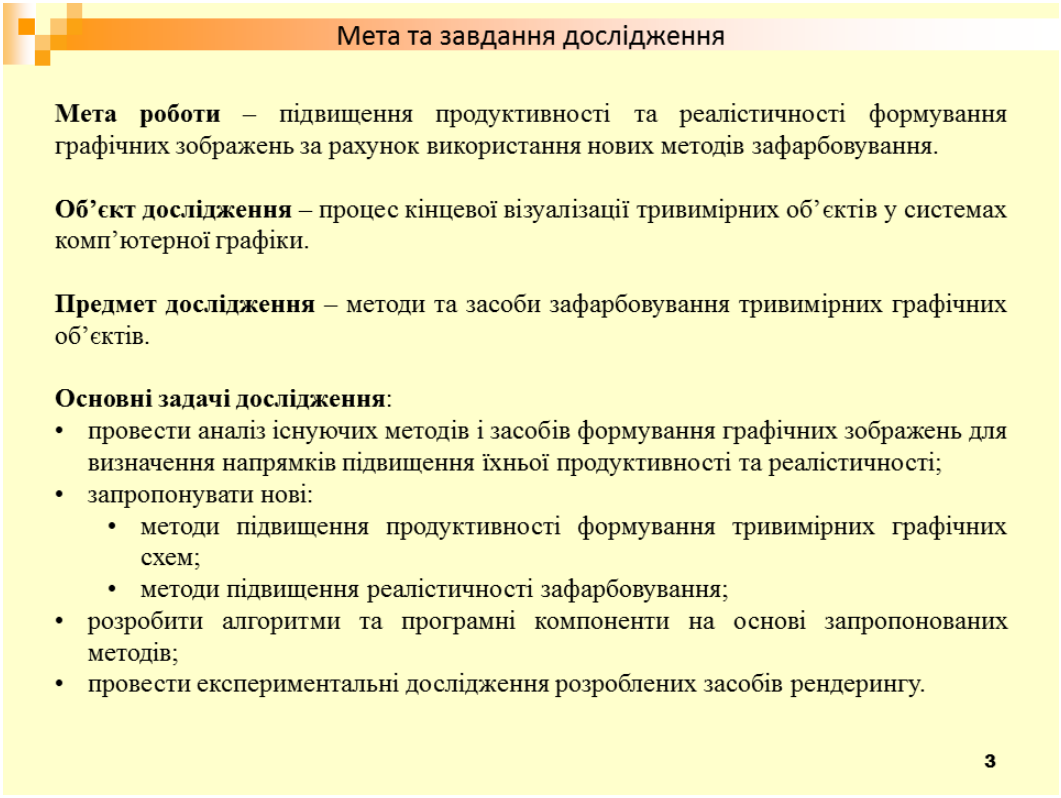


Рисунок Г.3 – Слайд презентації №3

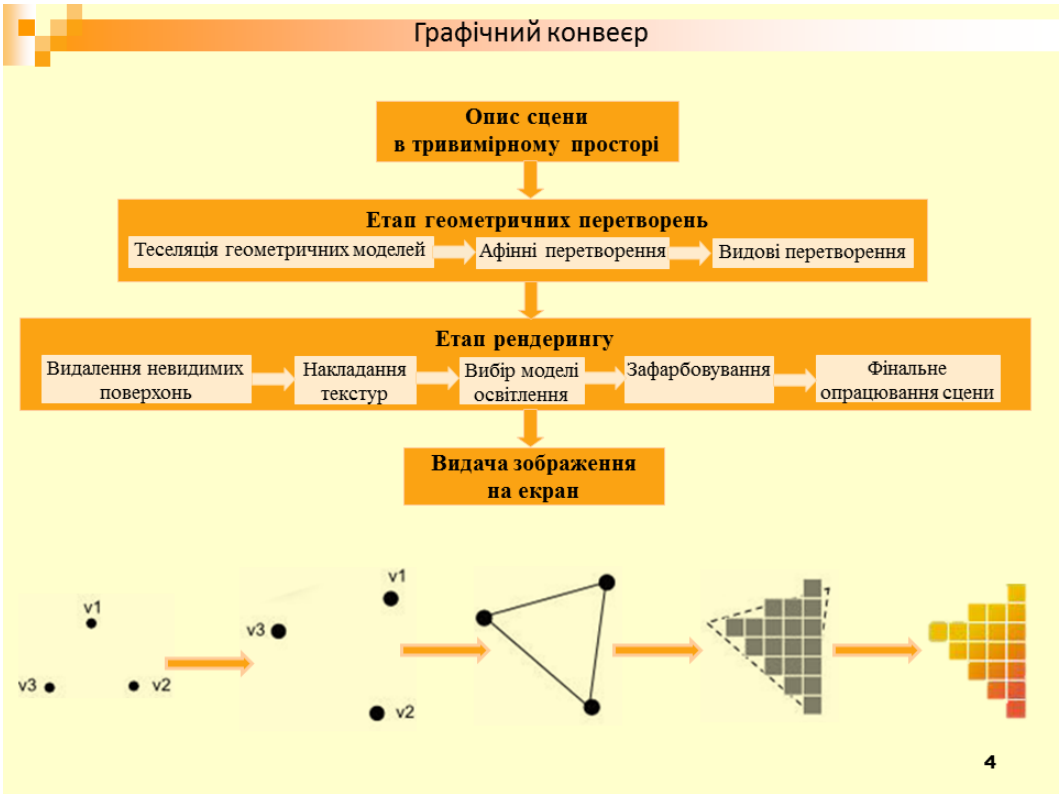


Рисунок Г.4 – Слайд презентації №4

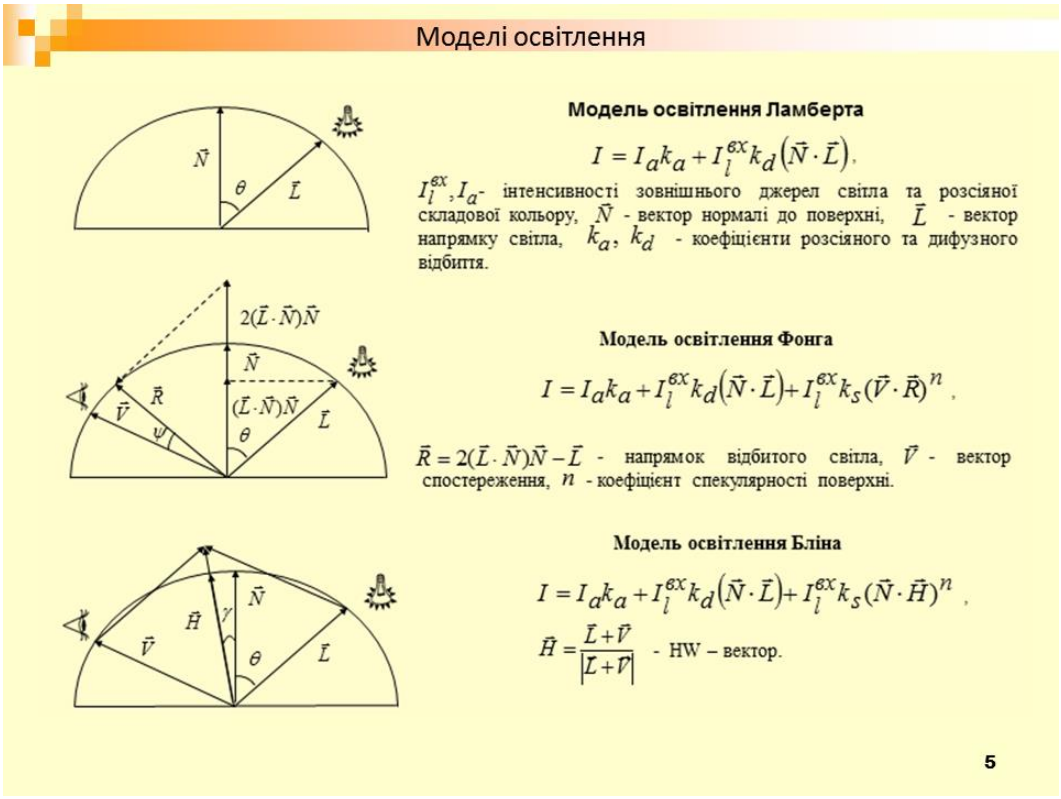


Рисунок Г.5 – Слайд презентації №5

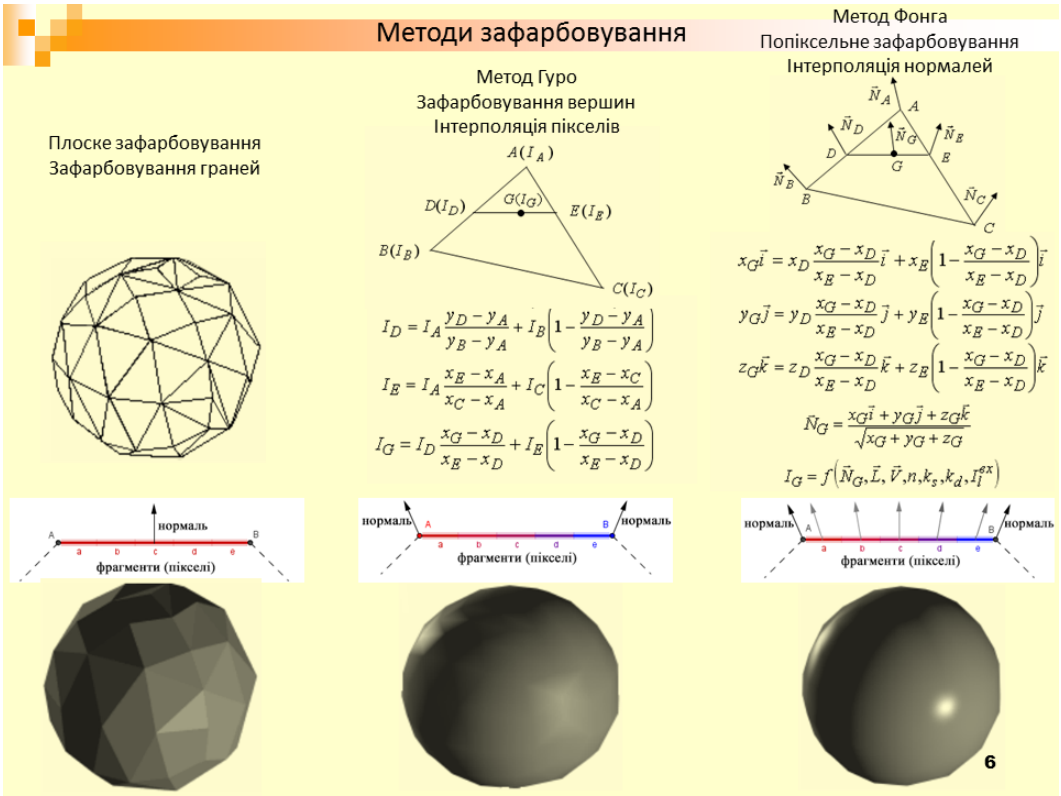
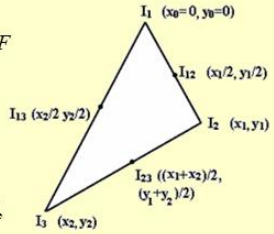


Рисунок Г.6 – Слайд презентації №6

Використання поверхні другого порядку для визначення інтенсивності кольору

$$I(x,y) = Ax^2 + By^2 + Cxy + Dx + Ey + F$$



$$\begin{cases} I_1 = F, \\ I_2 = Ax_1^2 + By_1^2 + Cx_1y_1 + Dx_1 + Ey_1 + F, \\ I_3 = Ax_2^2 + By_2^2 + Cx_2y_2 + Dx_2 + Ey_2 + F, \\ I_{12} = A\frac{x_1^2}{4} + B\frac{y_1^2}{4} + C\frac{x_1y_1}{4} + D\frac{x_1}{2} + E\frac{y_1}{2} + F, \\ I_{13} = A\frac{x_2^2}{4} + B\frac{y_2^2}{4} + C\frac{x_2y_2}{4} + D\frac{x_2}{2} + E\frac{y_2}{2} + F, \\ I_{23} = A\frac{(x_1+x_2)^2}{4} + B\frac{(y_1+y_2)^2}{4} + C\frac{(x_1+x_2)(y_1+y_2)}{4} + D\frac{(x_1+x_2)}{2} + E\frac{(y_1+y_2)}{2} + F. \end{cases}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2y_2 & x_2 & y_2 & 1 \\ x_1^2/4 & y_1^2/4 & x_1y_1/4 & x_1/2 & y_1/2 & 1 \\ x_2^2/4 & y_2^2/4 & x_2y_2/4 & x_2/2 & y_2/2 & 1 \\ (x_1+x_2)^2/4 & (y_1+y_2)^2/4 & (x_1+x_2)(y_1+y_2)/4 & (x_1+x_2)/2 & (y_1+y_2)/2 & 1 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_{12} \\ I_{13} \\ I_{23} \end{bmatrix}$$

7

Рисунок Г.7 – Слайд презентації №7

Використання поверхні другого порядку для визначення інтенсивності кольору

$$I(x_i,y_i) = Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Ey_i + F.$$

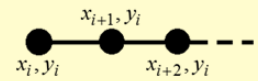
$$I(x_i,y_i) = x_i(Ax_i + Cy_i + D) + y_i(By_i + E) + F.$$

$$I(x_{i+1},y_i) = A(x_i+1)^2 + By_i^2 + C(x_i+1)y_i + D(x_i+1) + Ey_i + F =$$
$$= Ax_i^2 + 2Ax_i + A + By_i^2 + Cx_iy_i + Cy_i + Dx_i + D + Ey_i + F =$$
$$= I(x_i,y_i) + 2Ax_i + A + Cy_i + D.$$

$$I(x_{i+2},y_i) = A(x_i+2)^2 + By_i^2 + C(x_i+2)y_i + D(x_i+2) + Ey_i + F =$$
$$= Ax_i^2 + 4Ax_i + 4A + By_i^2 + Cx_iy_i + 2Cy_i + Dx_i + 2D + Ey_i + F =$$
$$= I(x_i,y_i) + 4Ax_i + 4A + 2Cy_i + 2D.$$

$$I(x_{i+3},y_i) = A(x_i+3)^2 + By_i^2 + C(x_i+3)y_i + D(x_i+3) + Ey_i + F =$$
$$= Ax_i^2 + 6Ax_i + 7A + By_i^2 + Cx_iy_i + 3Cy_i + Dx_i + 3D + Ey_i + F =$$
$$= I(x_i,y_i) + 6Ax_i + 9A + 3Cy_i + 3D.$$

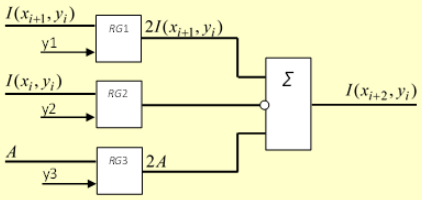
$$I(x_{i+2},y_i) = I(x_i,y_i) + 4Ax_i + 4A + 2Cy_i + 2D =$$
$$= 2[I(x_i,y_i) + 2Ax_i + A + Cy_i + D] - I(x_i,y_i) + 2A =$$
$$= 2I(x_{i+1},y_i) - I(x_i,y_i) + 2A.$$

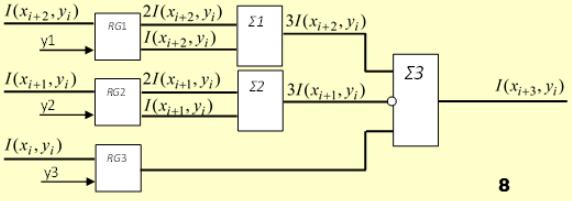


$$I(x_{i+3},y_i) = I(x_i,y_i) + 6Ax_i + 9A + 3Cy_i + 3D =$$
$$= [I(x_i,y_i) + 4Ax_i + 4A + 2Cy_i + 2D] +$$
$$+ [I(x_i,y_i) + 2Ax_i + A + Cy_i + D] - I(x_i,y_i) + 4A =$$
$$= I(x_{i+2},y_i) + I(x_{i+1},y_i) - I(x_i,y_i) + 4A.$$

$$4A = 2 \cdot 2A = 2[I(x_{i+2},y_i) - 2I(x_{i+1},y_i) + I(x_i,y_i)] =$$
$$= 2I(x_{i+2},y_i) - 4I(x_{i+1},y_i) + 2I(x_i,y_i).$$

$$I(x_{i+3},y_i) = I(x_{i+2},y_i) + I(x_{i+1},y_i) - I(x_i,y_i) +$$
$$+ 2I(x_{i+2},y_i) - 4I(x_{i+1},y_i) + 2I(x_i,y_i) =$$
$$= 3I(x_{i+2},y_i) - 3I(x_{i+1},y_i) + I(x_i,y_i) =$$
$$= 2I(x_{i+2},y_i) + I(x_{i+2},y_i) - 2I(x_{i+1},y_i) -$$
$$- I(x_{i+1},y_i) + I(x_i,y_i).$$





8

Рисунок Г.8 – Слайд презентації №8

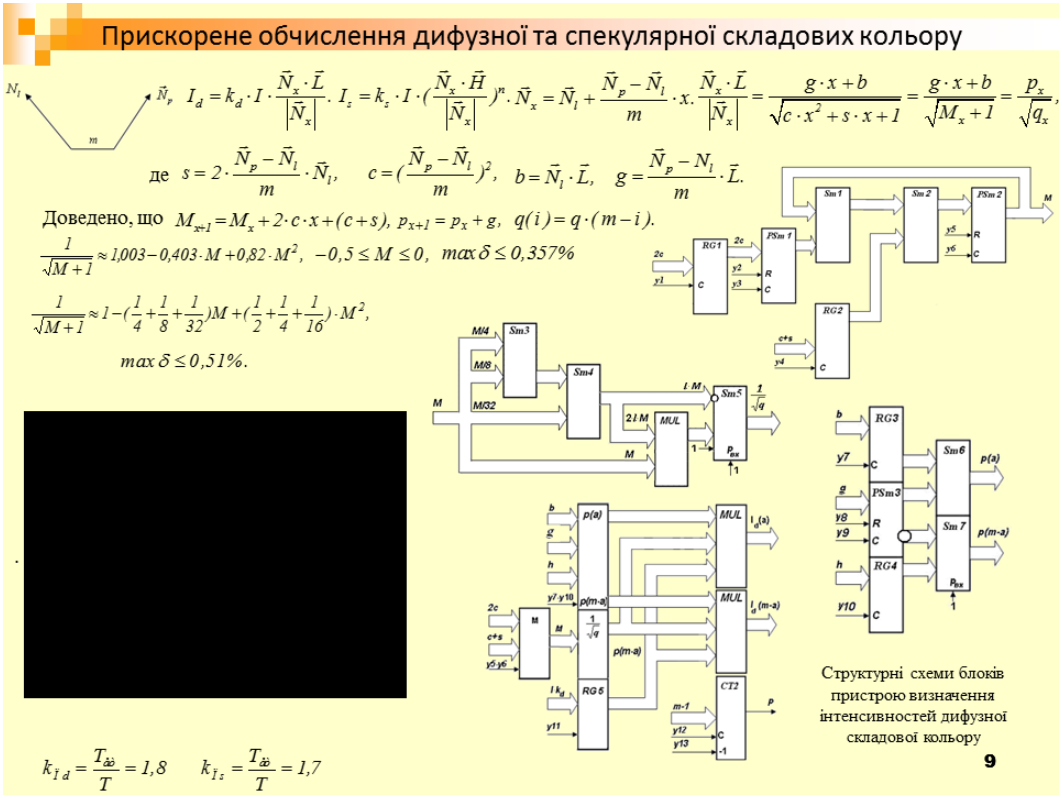


Рисунок Г.9 – Слайд презентації №9

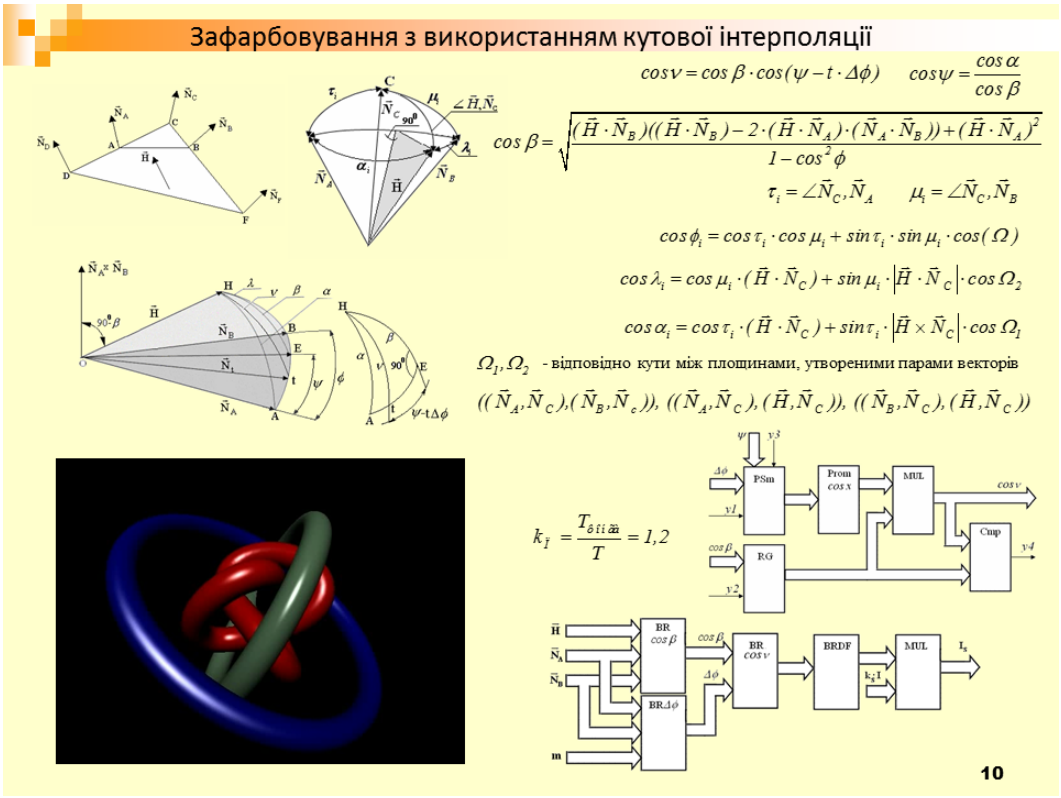


Рисунок Г.10 – Слайд презентації №10

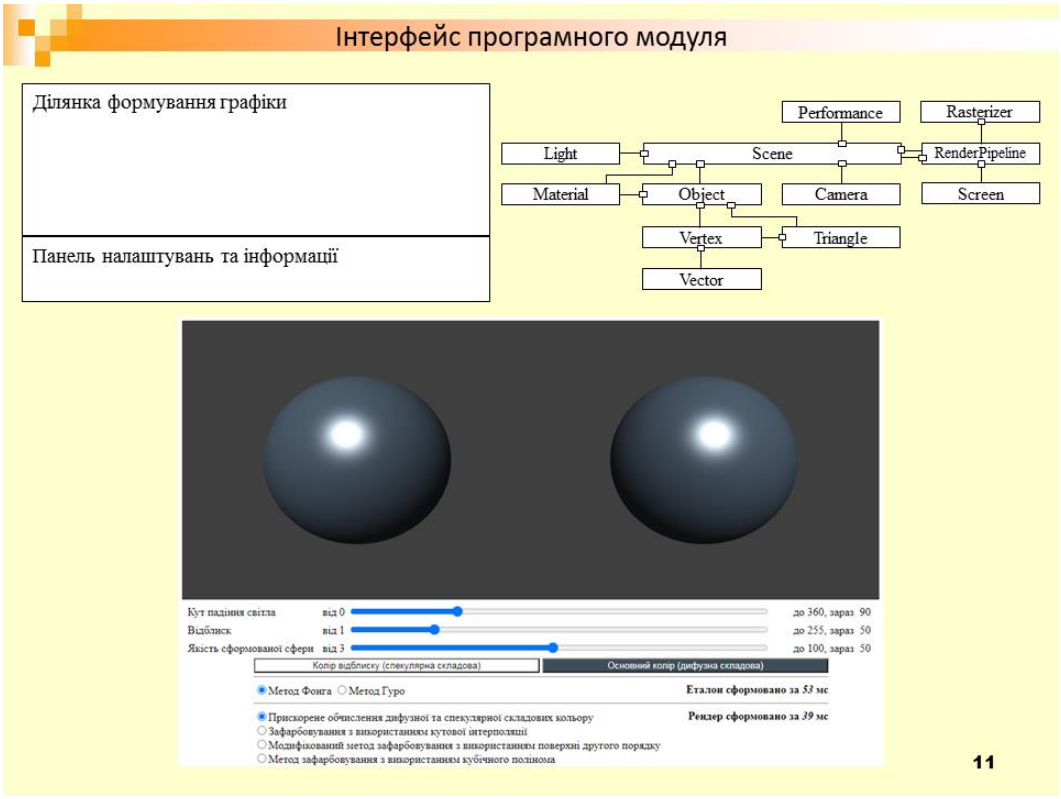


Рисунок Г.11 – Слайд презентації №11

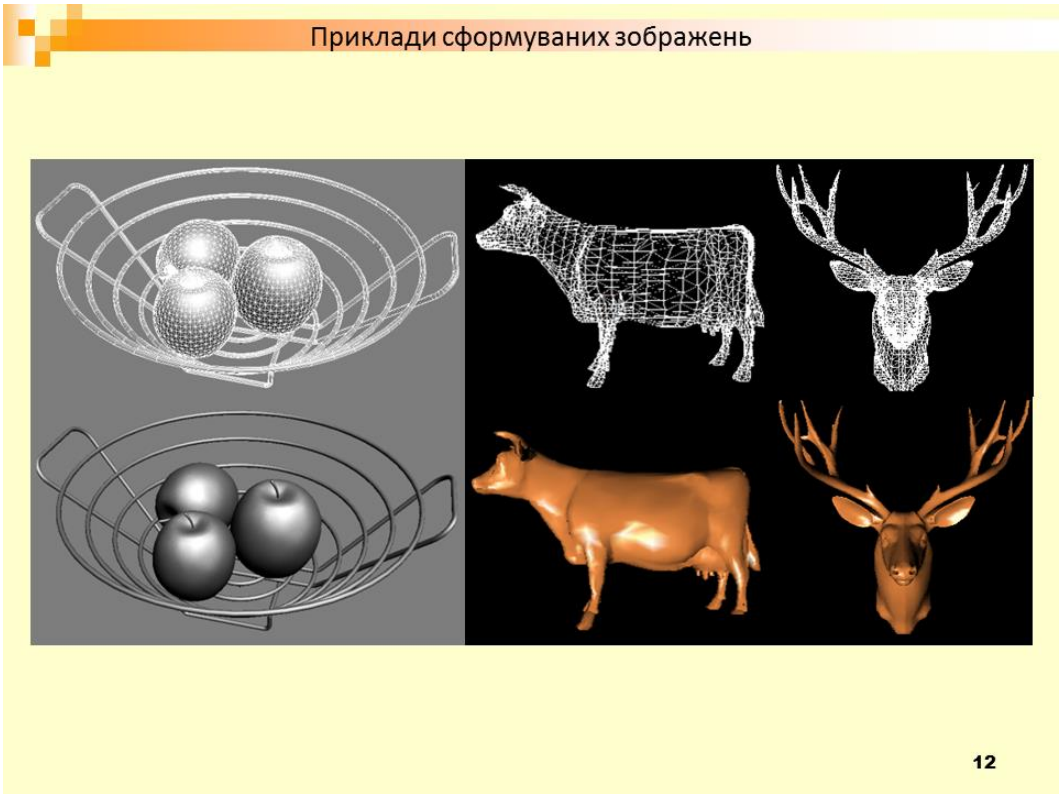


Рисунок Г.12 – Слайд презентації №12

Наукова новизна отриманих результатів

1. Вперше запропоновано метод зафарбовування, особливість якого полягає у вилучені з обчислювального процесу нормалізації векторів і розрахунку арккосинуса кута в циклі підготування. що дозволило зменшити час зафарбовування середньостатистичного трикутника порівняно з класичною реалізацією в середньому на 25% .
2. Подальшого розвитку отримав метод прискореного визначення дифузної та спекулярної складових кольору, особливість якого полягає в одночасному й незалежному визначенні інтенсивностей кольору відразу двох точок рядка растеризації. Обчислювальний процес, який реалізовано згідно з запропонованим підходу, не містить у циклі розрахунку складових кольору операцій ділення та знаходження квадратного кореня, які мають місце при класичній реалізації Це дозволило підвищити продуктивність та реалістичність рендерингу.
3. Вперше запропоновано метод зафарбовування, особливість якого полягає у використанні поверхні другого прядку для визначення інтенсивностей кольору точок поверхні. Це дозволило підвищити продуктивність рендерингу на 67% порівняно з методом Фонга.

Практична цінність отриманих результатів.
На основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби зафарбовування для комп'ютерних систем візуалізації тривимірних зображень.

13

Рисунок Г.13 – Слайд презентації №13

Апробація матеріалів магістерської кваліфікаційної роботи

Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на:

- Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2020),
- XIV міжнародній науково-практичній конференції «Інформаційні технології та комп'ютерна інженерія»» (Одеса, 2020),
- Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів. «Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації» (Одеса, 2020),
- XIV міжнародній науково-практичній конференції «Інформаційні технології і автоматизація». (Одеса, 2021),
- Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2021).

Публікації.
Основні результати досліджень опубліковано в 10 наукових працях, з них 4 статті у фаховому виданні України , 6 – у матеріалах конференцій.

14

Рисунок Г.14 – Слайд презентації №14

ст. гр.: 2ПІ-20м Озерчук Дмитро Анатолійович

ст. гр.: 2ПІ-20м Озерчук Дмитро Анатолійович