МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва
Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2025 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ДОДАТКУ ДЛЯ ВИСОКОПРОДУКТИВНОГО ФОРМУВАННЯ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ МЕТОДУ ГУРО

Виконав – студент 4 курсу, групи 41-кн
освітньої програми "Комп’ютерні науки"
спеціальності 122 Комп’ютерні науки
Колесніченко Олександр Віталійович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2025

Зміст

ВСТУП 3

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ 5

1.1. Аналіз методів рендерингу 5

1.2. Аналіз методу Гуро 7

1.3. Аналіз аналогів 11

1.4. Постановка задачі 16

РОЗДІЛ 2 РОЗРОБКА МЕТОДІВ ДЛЯ ФОРМУВАННЯ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ МЕТОДУ ГУРО 18

2.1. Розробка методу зафарбовування тривимірної моделі за Гуро 18

2.2 Адаптації та вдосконалення техніки рендерингу за методом Гуро. 22

2.3. Розробка методів розпаралелення зафарбовування за Гуро..... 26

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИСОКОПРОДУКТИВНОГО ФОРМУВАННЯ ЗОБРАЖЕНЬ 32

3.1. Розробка моделі та алгоритмів роботи програмного забезпечення 32

3.2. Розробка модуля затінення тривимірної моделі за методом Гуро. 37

3.3. Створення модуля для високоефективного генерування зображень 43

3.4. Тестування системи оптимізованого рендерингу..... 47

ВИСНОВКИ..... 55

СПИСОК ЛІТЕРАТУРИ..... 56

ДОДАТКИ..... 59

ВСТУП

Сьогоднішній розвиток технологій віртуальної та доповненої реальності, наукової візуалізації й інтерактивних симуляцій зумовлює суворі вимоги до швидкості та якості формування зображень. Навіть короткі затримки виводу кадру негативно позначаються на сприйнятті користувача, особливо у VR-середовищах, системах дистанційного керування чи високоточних медичних візуалізаціях [1].

На сьогоднішній день існує ряд технічних викликів, пов'язаних з формуванням зображень, що вимагають нових підходів та інструментів для їх вирішення. Велика увага приділяється підвищенню якості, продуктивності та швидкості формування зображень, а також зменшенню обсягу необхідних ресурсів.

Метод згладженого за освітленням за Гуро (Gouraud Shading) залишається одним із базових підходів до реалізації плавних градієнтів на багатокутних поверхнях і водночас добре масштабується на паралельних архітектурах GPU. Однак класичні реалізації потребують оптимізації під сучасні обчислювальні ресурси, щоб забезпечити високу пропускну здатність при мінімальних затратах енергії й пам’яті. Тому розробка спеціалізованого застосунку для високопродуктивного формування зображень із використанням методу Гуро є актуальним завданням прикладної комп’ютерної графіки та програмної інженерії. [2-5].

Мета роботи – створити та експериментально перевірити програмний застосунок, здатний у реальному часі формувати якісні тривимірні зображення методом Гуро.

Об’єкт дослідження – процес комп’ютерного формування зображень у системах реального часу.

Предмет дослідження – алгоритми та програмні засоби реалізації методу згладженого освітлення за Гуро, їх оптимізація під паралельні обчислювальні архітектури й інтеграція у високопродуктивний прикладний програмний продукт.

Розв’язання поставлених задач дозволить підвищити ефективність рендерингу в інтерактивних графічних системах та закладе основу для подальшої модернізації засобів візуалізації з урахуванням новітніх GPU-технологій.

У ході проведення дослідження було використано низку методів, зокрема: положення теорії чисел і чисельного аналізу; засади теорії алгоритмів; методи інтерполяції функцій; підходи тривимірного моделювання; а також засоби комп’ютерного моделювання для перевірки та аналізу теоретичних результатів.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз методів рендерингу

Рендеринг (від англ. rendering – візуалізація) – це процес перетворення цифрової тривимірної сцени на двовимірне зображення, що сприймається людиною як візуальне представлення простору. У комп'ютерній графіці існує кілька основних підходів до рендерингу, які різняться за складністю, точністю відображення фізичних процесів освітлення та обчислювальними затратами. Ефективний вибір методу рендерингу визначає якість кінцевого зображення, продуктивність системи та придатність для застосування в конкретній галузі – від інтерактивних ігор до високоточних наукових візуалізацій.

Основні методи рендерингу можна умовно поділити на дві групи: растрові (rasterization-based) і трасувальні (ray-based).

Растрові методи, зокрема класична растеризація, широко застосовуються в режимах реального часу (наприклад, у відеоіграх або VR/AR-середовищах). Ці методи полягають у проєкції тривимірної геометрії на площину екрана з подальшим обчисленням кольору пікселів на основі освітлення, текстурування та шейдингу. До типових технік освітлення відносяться методи Фонга, Гуро, Блінна–Фонга тощо. Їх головною перевагою є швидкість, але вони часто спрощують фізичні моделі світлової взаємодії, що обмежує фотореалістичність зображень.

Метод Гуро (Gouraud Shading) – один із базових методів інтерполяційного шейдингу, запропонований Анрі Гуро в 1971 році. Він передбачає обчислення освітлення лише у вершинах полігонів, після чого значення яскравості інтерполюються по поверхні трикутника. Це дозволяє досягти плавного переходу між кольорами на поверхні без значних обчислювальних витрат, що робить метод надзвичайно ефективним для GPU-паралелізації.

Трасувальні методи, такі як Ray Tracing та Path Tracing, ґрунтуються на моделюванні шляху світлових променів від джерел до ока спостерігача. Вони дозволяють точно відтворювати ефекти віддзеркалення, заломлення,

глобального освітлення, тіней та каустики. Ці методи забезпечують високу реалістичність, проте вимагають значних обчислювальних ресурсів. З розвитком апаратного прискорення трасування променів (наприклад, NVIDIA RTX) їх починають інтегрувати й у реальний час, але повноцінне використання досі обмежене високопродуктивними системами.

Крім того, останніми роками активно розвиваються гібридні методи рендерингу, які поєднують переваги растеризації та трасування променів. Наприклад, базове зображення формується методом растеризації, а вибіркові оптичні ефекти (тіні, рефлексії) додаються через трасування променів. Це дозволяє досягати балансу між якістю та швидкістю.

З урахуванням потреб у високопродуктивному формуванні зображень у реальному часі, метод Гуро залишається актуальним завдяки своїй ефективності та простоті реалізації. Його потенціал масштабування на сучасних обчислювальних архітектурах (зокрема, на GPU) робить його привабливим для застосування в інтерактивних графічних системах.

Отже, аналіз методів рендерингу свідчить про доцільність використання оптимізованих растрових методів у задачах, де пріоритетом є висока продуктивність і візуальна плавність, що безпосередньо визначає вибір методу Гуро як основи для розробки ефективного рендер-ядра.

У сучасних умовах, коли важливо одночасно досягти високої продуктивності й достатньої якості візуалізації, метод Гуро не втрачає своєї актуальності. Подальша розробка алгоритмів та програмних інструментів, що базуються на ньому, відкриває перспективи для покращення рендерингу в реальному часі. Особливої ефективності можна досягти завдяки поєднанню цього методу з сучасними технологіями – такими як паралельні обчислення, графічні процесори та апаратні прискорювачі, що дає змогу досягати високоякісних зображень без значного зниження швидкості візуалізації.

1.2. Аналіз методу Гуро

У сучасному цифровому світі тривимірна комп’ютерна графіка відіграє ключову роль у візуалізації об'єктів у різноманітних галузях – від бізнесу та реклами до медицини й індустрії розваг. Її застосування забезпечує високу інтерактивність, точність та реалістичність графічного представлення, що особливо важливо для програмних рішень, які потребують наочного відображення складних об’єктів і процесів. Основою побудови тривимірного зображення на двовимірному екрані є рендеринг – процес, у якому визначається колір і яскравість кожного пікселя для створення зображення, що імітує просторову глибину та освітлення.

Одним із ключових принципів сучасної 3D-графіки є використання полігональних моделей, найчастіше у формі трикутників. Такі моделі складаються з вершин і нормалей, які слугують основою для обчислення освітлення поверхні. Для реалістичного візуального подання поверхні цих трикутників застосовують різні методи шейдингу (зафарбовування). Серед них – метод Гуро (Gouraud Shading), який дозволяє створювати плавні градієнтні переходи кольору, надаючи об’єктам м’яке освітлення та більш природний вигляд.

Метод Гуро був запропонований Анрі Гуро в 1971 році як спосіб покращення візуального сприйняття тривимірних об'єктів. Його суть полягає в тому, що освітлення обчислюється лише у вершинах трикутника, а отримані значення інтенсивності кольору інтерполюються по площині полігону. Кожна вершина має своє значення яскравості, яке базується на нормалі в цій точці та орієнтації відносно джерел світла. У результаті інтерполяції значень між вершинами забезпечується плавний перехід кольору всередині трикутника.

Цей підхід забезпечує значне зниження обчислювального навантаження порівняно з методами, які вимагають обчислення освітлення для кожного пікселя (наприклад, метод Фонга). Водночас він зберігає достатню реалістичність для більшості застосунків, що робить метод Гуро дуже

ефективним у контексті задач реального часу, таких як мобільні ігри, AR/VR-середовища та інтерактивні навчальні системи.

Перевагою методу є здатність створювати більш "живе" зображення, де деталі поверхні передаються м’якими переходами. Завдяки цьому зображення виглядає об’ємнішим, навіть при обмеженій кількості полігонів. Крім того, використання тривимірних моделей дає змогу включати різні типи даних – геометричні, текстурні, колірні – що відкриває можливості для глибшого аналізу сцени, включаючи симуляцію тіней, освітлення та взаємодії об’єктів.

Процес зафарбовування за методом Гуро можна описати на прикладі трикутника з вершинами A, B і C, у яких попередньо розраховано інтенсивність світла – відповідно I_A, I_B, I_C. Далі виконується лінійна інтерполяція цих значень уздовж кожного з ребер трикутника, щоб визначити інтенсивність кольору в усіх внутрішніх точках полігону. Основна задача при цьому – обчислення приросту яскравості вздовж відрізків, що з’єднують вершини (AB, BC, AC), після чого аналогічна інтерполяція виконується для кожного рядка пікселів, що проходить крізь трикутник:

$$\Delta I_{AB}=(I_A-I_B)/\text{БП}_{AB} \quad (1.1)$$

$$\Delta I_{AC}=(I_A-I_C)/\text{БП}_{AC} \quad (1.2)$$

$$\Delta I_{BC}=(I_B-I_C)/\text{БП}_{BC} \quad (1.3)$$

де БП - приріст координат Δx, Δy відрізка прямої.

Розглянемо растеризовану лінію, що проходить через точку D (рис. 1.1). Її кінцевими точками є точки L та M. Для подальших обчислень визначимо прирости інтенсивності вздовж відповідних ребер. Щоб обчислити інтенсивність кольору в точках L і M, скористаємося такими формулами:

$$I_M=I_A+\Delta I_{AB}\cdot\text{БП}_{AM} \quad (1.4)$$

$$I_L=I_A+\Delta I_{AC}\cdot\text{БП}_{AL} \quad (1.5)$$

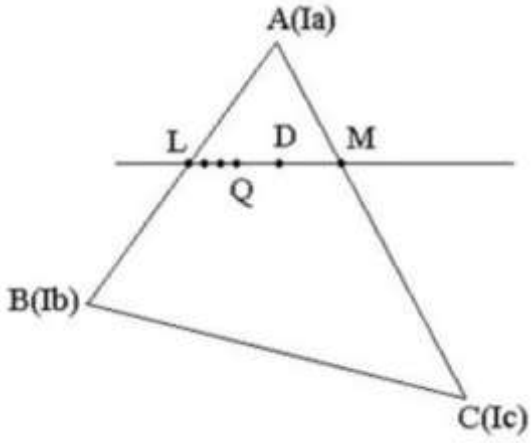


Рисунок 1.1 – Інтерполяційна обробка трикутника

Обчислимо зміну інтенсивності вздовж растеризованої лінії LM. Для цього скористаємося такою формулою:

$$\Delta I_{LM}=(I_L-I_M)/|LM| \tag{1.6}$$

На наступному етапі визначимо інтенсивність кольору для всіх точок, розташованих на лінії растеризації. Як приклад, розраховано інтенсивність у точці D:

$$I_D=I_L+\Delta I_{LM}*|LD| \tag{1.7}$$

Інтенсивність кольору будь-якої іншої точки, що знаходиться всередині заданого полігону, обчислюється за аналогічним принципом. Найменша похибка досягається тоді, коли поверхню розбивають на прямокутні трикутники з катетами, паралельними осям координат, оскільки у такому випадку довжина більшого катета відповідає його приросту.

Під час зафарбовування всього трикутника сканувальна лінія проходить через усі точки полігона поетапно. Приріст як уздовж самої сканувальної лінії, так і між сусідніми скан-лініями становить один піксель. Це дає змогу замінити операції множення на більш ефективне накопичувальне додавання.

$$I_Q=I_L+3*\Delta I_{LM} \tag{1.8}$$

На рисунку 1.2, а представлено рендеринг трикутника I_A=5, I_B=9, I_C=1. Значення яскравостей обчислені за (рис. 1.2 б).

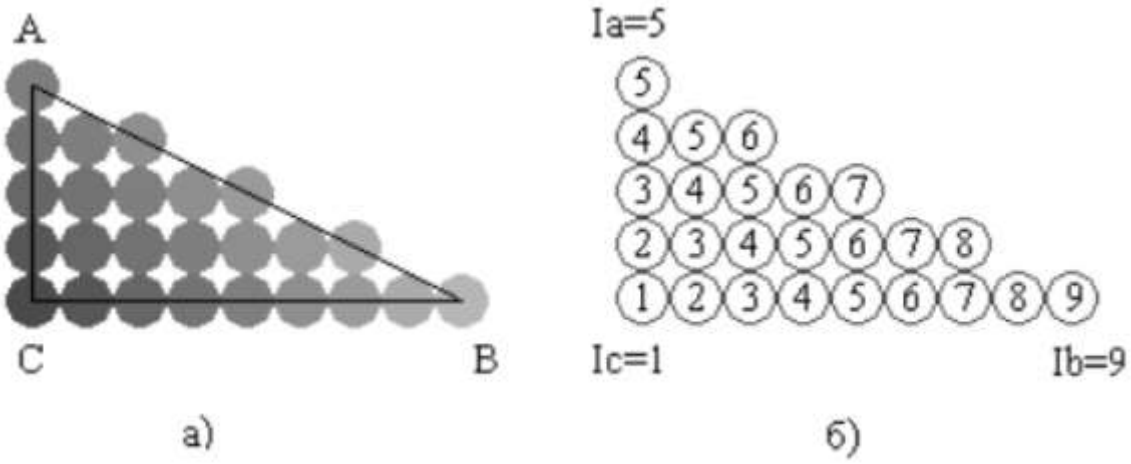


Рисунок 1.2 – Приклад зафарбовування трикутника

Попри численні переваги методу Гуро у формуванні зображень, його застосування супроводжується певними технічними викликами. Одним із ключових є забезпечення високої продуктивності та ефективної обробки графічних об’єктів, особливо в умовах роботи з великими масивами тривимірних даних або в режимі реального часу. Це особливо критично для задач, де важливі як швидкість обробки, так і висока точність відображення.

На сьогодні існує широкий спектр програмного забезпечення та бібліотек для графічної обробки, проте не всі з них підтримують метод Гуро на належному рівні ефективності та точності. Крім того, багато таких інструментів є комерційними, що обмежує їхню доступність для більшості користувачів, зокрема для освітніх або дослідницьких цілей.

Таким чином, аналіз сучасного стану технологій вказує на актуальність створення високопродуктивного алгоритму зафарбовування тривимірних об’єктів на основі методу Гуро. Розробка спеціалізованого програмного забезпечення, яке дозволяє реалізувати цей метод із максимальною продуктивністю, є важливим кроком до підвищення якості та швидкодії процесу рендерингу зображень.

1.3. Аналіз аналогів

У галузі тривимірної графіки та візуалізації існує низка програмних рішень, що використовуються для створення, рендерингу та анімації 3D-сцен. Серед найпоширеніших і функціонально потужних систем слід виділити Blender**,** Autodesk Maya**,** Cinema 4D**,** Unity та Autodesk 3ds Max. Кожен з цих інструментів має свої переваги та недоліки, що впливають на доцільність їх застосування в різних контекстах, зокрема й у задачах, де може бути реалізований метод Гуро.

Blender (рис. 1.3) – це безкоштовне програмне забезпечення з відкритим кодом для 3D-моделювання, анімації, рендерингу, симуляцій, композитингу та постобробки. Підтримує як CPU, так і GPU-рендеринг, включаючи різні методи шейдингу, зокрема й поверхневий шейдинг на кшталт методу Гуро [7].

Сильні сторони:

- Повністю безкоштовний і з відкритим кодом.
- Широкий набір інструментів для моделювання, текстурування, анімації та рендерингу.
- Активна спільнота, регулярні оновлення.
- Підтримка сторонніх плагінів та Python-скриптів.

Слабкі сторони:

- Складний для початківців через велику кількість функцій.
- Інтерфейс може здаватися перевантаженим.
- У деяких випадках поступається комерційним аналогам у спеціалізованих задачах (наприклад, у кінематографічній анімації високого рівня).



Рисунок 1.3 – Приклад роботи застосунку Blender

Autodesk Maya (рис. 1.4) – це професійне середовище для створення 3D-анімації, візуальних ефектів і рендерингу, особливо популярне в кіно- та ігровій індустрії. Забезпечує підтримку різних алгоритмів освітлення та рендерингу, включаючи складні методи обробки поверхонь [8].



Рисунок 1.4 – Приклад роботи застосунку Autodesk Maya

Сильні сторони:

- Потужні інструменти для анімації, скелетної прив'язки та симуляцій.
- Високоякісний рендеринг.

- Широко використовується у професійному середовищі.
- Розширюваність за допомогою MEL- або Python-скриптів.

Слабкі сторони:

- Висока вартість ліцензії.
- Вимагає потужного обладнання.
- Складна система навчання, потребує багато часу на освоєння.

Cinema 4D (рис. 1.5) – це комерційне програмне забезпечення для 3D-моделювання, анімації, візуальних ефектів і рендерингу, яке відоме своєю простотою у використанні та стабільністю. Часто використовується у рекламній графіці, телевізійних заставках та анімаціях [9].



Рисунок 1.5 – Приклад роботи застосунку Cinema 4D

Сильні сторони:

- Інтуїтивно зрозумілий інтерфейс.
- Висока стабільність та зручність роботи.
- Ефективні інструменти для моушн-дизайну.
- Швидкий рендеринг із хорошою якістю результату.

Слабкі сторони:

- Обмежені можливості в порівнянні з Maya у галузі складної анімації та симуляцій.
- Платна модель ліцензування.
- Менш популярна у геймдеві, ніж Maya або Blender.

Unity (рис. 1.6) – це ігровий рушій, орієнтований на розробку інтерактивних додатків, включаючи ігри, AR/VR та симуляції. Підтримує 3D-графіку в реальному часі та має власні засоби для рендерингу (URP, HDRP), що можуть бути адаптовані під різні методи шейдингу [10].

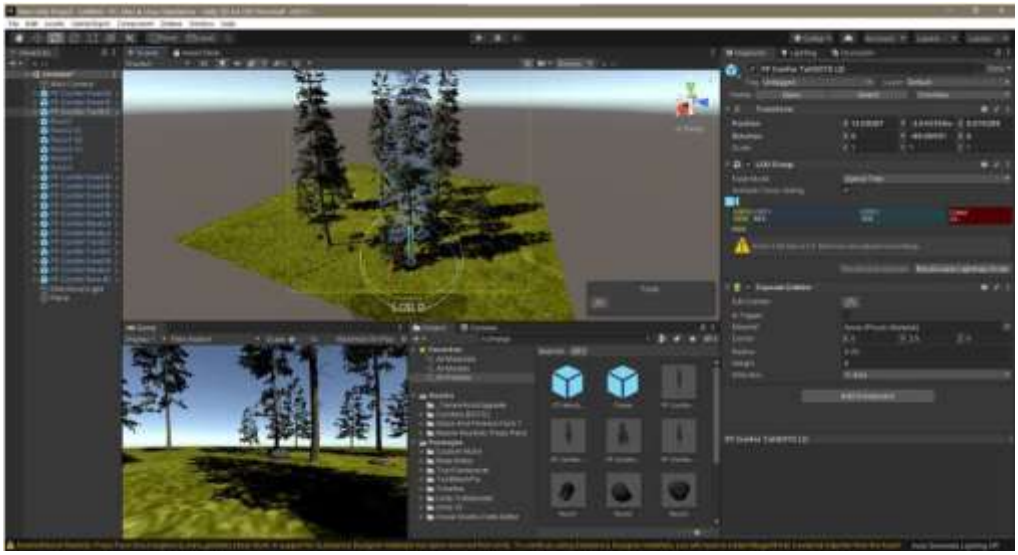


Рисунок 1.6 – Приклад роботи застосунку Unity

Сильні сторони:

- Потужні засоби для розробки в реальному часі.
- Підтримка C#-скриптів для створення кастомних шейдерів.
- Гнучка система рендерингу.
- Можливість кросплатформенного розгортання.

Слабкі сторони:

- Орієнтований більше на ігрову розробку, ніж на кінематографічний рендеринг.
- Для реалізації спеціалізованих методів, як-от Гуро, потребує ручної реалізації шейдерів.
- Високе споживання ресурсів при складних сценах.

3D Max це потужний інструмент для тривимірного моделювання та рендерингу, що використовується в різних галузях, включаючи архітектуру, дизайн та ігрову розробку. Він має велику кількість інструментів та плагінів, що дозволяє реалізувати різноманітні проекти.

Autodesk 3ds Max (рис. 1.7) – [11] це ще один потужний інструмент від Autodesk для 3D-моделювання, візуалізації та анімації, широко застосовуваний у візуалізації архітектурних проєктів та ігрових сцен.

Сильні сторони:

- Високоякісна візуалізація з використанням рендерів Arnold і V-Ray.
- Зручність при роботі з архітектурними та інженерними об'єктами.
- Підтримка великої кількості форматів та плагінів.

Слабкі сторони:

- Висока вартість та комерційна ліцензія.
- Менша гнучкість у створенні складної анімації порівняно з Maya.
- Потребує потужного обладнання для ефективної роботи.

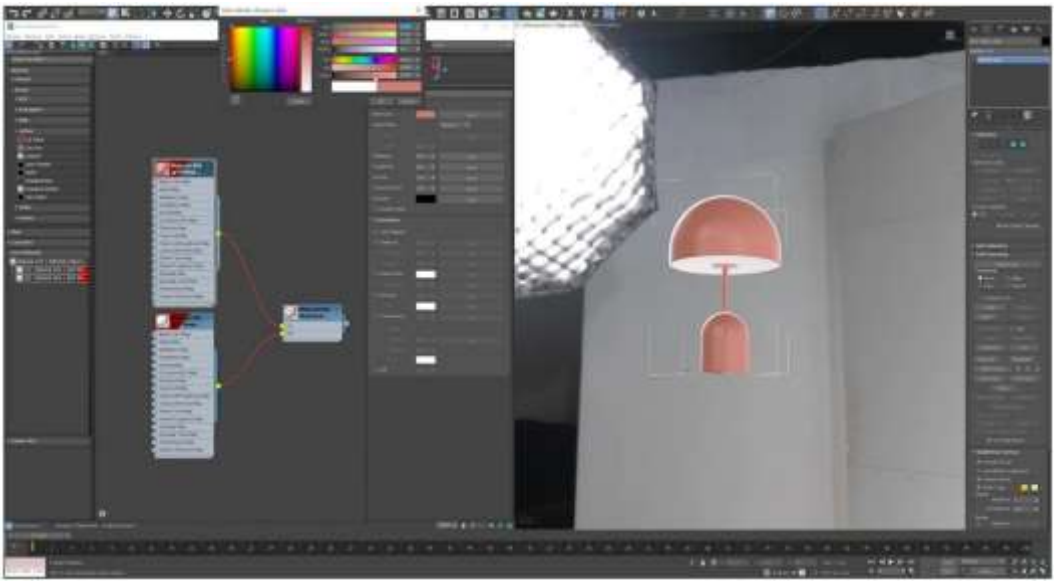


Рисунок 1.7 – Приклад роботи застосунку 3D Max

Кожен з розглянутих програмних продуктів має свою нішу застосування та набір функцій, що визначає його придатність для реалізації методу Гуро. Наприклад, Blender є найкращим варіантом з відкритим кодом для експериментів та досліджень, тоді як Maya – професійний інструмент для кінематографічної графіки. Unity забезпечує гнучкість у реальному часі, але потребує додаткових зусиль для реалізації нестандартних методів рендерингу. З огляду на вартість, доступність та функціональність, постає актуальне завдання – створення спеціалізованого, високопродуктивного рішення для рендерингу

методом Гуро, яке було б водночас ефективним і доступним для широкого кола користувачів.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Blender	Autodesk Maya	Cinema 4D	Unity	3D Max	Власний модуль
Вільне використання	+	-	-	+	-	+
Потреба в додатковому навчанні для ефективної роботи	-	-	-	-	-	+
Підтримка функціоналу рендерингу	+	+	+	-	+	+
Наявність інтерфейсних засобів для зміни параметрів	+	+	+	-	+	+
Низькі вимоги до апаратного забезпечення	-	-	-	-	-	+
Сумарна оцінка ефективності	60%	40%	40%	20%	40%	100%

Згідно з проведеним порівняльним аналізом, створення власного програмного забезпечення для високопродуктивного формування зображень за методом Гуро є обґрунтованим. Такий продукт дозволить усунути обмеження існуючих рішень та забезпечить розширений функціонал для ефективного рендерингу тривимірних моделей із використанням цього методу.

1.4. Постановка задачі

На основі аналізу переваг і недоліків існуючих програмних рішень для високопродуктивного формування зображень із використанням методу Гуро було визначено доцільність створення власного програмного засобу. Такий підхід дозволить усунути наявні обмеження та запропонувати більш

ефективний інструмент для роботи з тривимірною графікою. Для досягнення цієї мети необхідно вирішити наступні задачі:

- розробити алгоритм формування зображень, заснований на методі Гуро, який забезпечуватиме високу точність та деталізацію зображення;
- створити модуль зафарбовування тривимірних моделей, здатний працювати з геометричними даними у форматі полігональних сіток;
- реалізувати модуль рендерингу, що гарантує швидке й ефективне формування зображень у реальному часі;
- розробити зручний графічний інтерфейс, орієнтований на користувача, з підтримкою інтерактивної взаємодії з 3D-моделями;
- забезпечити інтеграцію всіх функціональних модулів у єдину програмну систему;
- створити повноцінний програмний продукт із реалізацією методу Гуро для візуалізації тривимірних об’єктів;
- автоматизувати процес рендерингу з акцентом на високу роздільну здатність, продуктивність і оптимізацію апаратних витрат;
- провести всебічне тестування розробленого програмного забезпечення.

У результаті реалізації зазначених задач буде створено програмний інструмент, що відкриє нові можливості для високоточних і продуктивних візуалізацій. Це особливо актуально для медичних застосунків, де завдяки якісному відтворенню тривимірних моделей людського тіла фахівці зможуть здійснювати індивідуалізовану діагностику та планування лікування з підвищеною точністю.

РОЗДІЛ 2 РОЗРОБКА МЕТОДІВ ДЛЯ ФОРМУВАННЯ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ МЕТОДУ ГУРО

2.1. Розробка методу зафарбовування тривимірної моделі за Гуро

Процес реалізації методу зафарбовування тривимірної моделі за Гуро включає кілька основних етапів, які спрямовані на досягнення реалістичного зображення поверхні об’єкта. На початковому етапі здійснюється підготовка тривимірної моделі, яка передбачає конвертацію геометричних форматів і оптимізацію структури моделі для подальшої обробки.

Одним із ключових кроків є тріангуляція – розбиття поверхні моделі на трикутники, що є необхідною умовою застосування методу Гуро. Такий підхід дозволяє окремо виконувати розрахунки освітлення для кожного трикутника, значно спрощуючи загальний процес рендерингу.

Метод інтерполяції інтенсивності, запропонований Гуро (так зване зафарбовування Гуро), полягає в лінійному поширенні значень інтенсивності по всій площині полігону. Інтенсивності в вершинах полігонів узгоджуються з аналогічними значеннями сусідніх полігонів по спільних ребрах, що дозволяє уникнути помітних переходів і розривів яскравості, властивих плоскому зафарбовуванню.

Візуалізація поверхні кожного полігону за методом Гуро передбачає виконання наступних обчислень:

- обчислення середнього одиничного вектора нормалі в кожній вершині полігону;
- застосування моделі освітлення для визначення інтенсивності світла у вершинах;
- виконання лінійної інтерполяції отриманих інтенсивностей по площині полігону;
- нормаль у кожній вершині отримується шляхом усереднення нормалей усіх прилеглих полігонів, що мають цю спільну вершину (див. рис. 2.1).

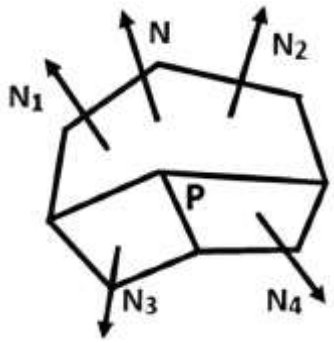


Рисунок 2.1 – Середні значення векторів нормалей

Отже, для будь-якої вершини V тривимірної моделі обчислюється одиничний вектор нормалі до цієї вершини за відповідною формулою:

$$N_v = \frac{\sum_{k=1}^{n} N_k}{|\sum_{k=1}^{n} N_k|} \tag{2.1}$$

Після визначення нормалей у вершинах конкретного полігону, наступним кроком є обчислення інтенсивності освітлення в кожній вершині на основі застосування моделі освітлення:

$$\mathrm{I}_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2 \tag{2.2}$$

Після визначення формули для обчислення інтенсивності в точці наступним важливим етапом є реалізація інтерполяції значень інтенсивності кольору вздовж ребер полігону на тривимірній моделі (див. рис. 2.2). Для цього кожен полігон обробляється окремо шляхом застосування сканування та розрахунку інтенсивності відповідно до обраної моделі освітлення.

Під час проходження кожної сканувальної лінії інтенсивність у точці її перетину з ребром полігону визначається методом лінійної інтерполяції між інтенсивностями в кінцевих точках ребра.

Наприклад, на рис. 2.2 зображено, як сканувальна лінія перетинає ребро полігону з вершинами в позиціях 1 і 2 у точці 4. Для швидкого обчислення інтенсивності в точці 4 застосовується лінійна інтерполяція між значеннями I_1 та I_2, використовуючи лише вертикальне зміщення сканувальної лінії.

Аналогічно, інтенсивність у точці 5 – правому перетині тієї ж лінії з ребром між вершинами 2 і 3 – визначається інтерполяцією між інтенсивностями цих вершин.

.

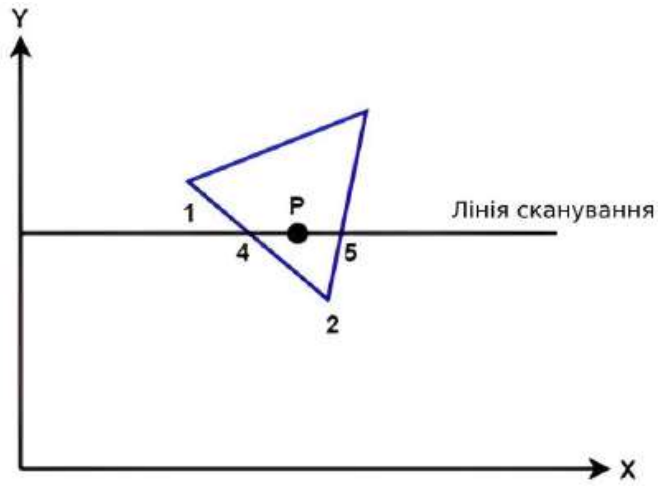


Рисунок 2.2 – Інтерполяції інтенсивностей

Після встановлення граничних значень інтенсивності для конкретної сканувальної лінії, значення інтенсивності у внутрішній точці (наприклад, точці P) обчислюється шляхом лінійної інтерполяції між інтенсивностями у точках 4 та 5 наступним чином:

$$I_P = \frac{x_5 - x_P}{x_5 - x_4} I_4 + \frac{x_P - x_4}{x_5 - x_4} I_5 \tag{2.3}$$

Інкрементний підхід застосовується для обчислення послідовних значень інтенсивності вздовж меж між сусідніми сканувальними лініями, а також для визначення значень інтенсивності вздовж самих ліній сканування, як проілюстровано на рисунку 2.3.

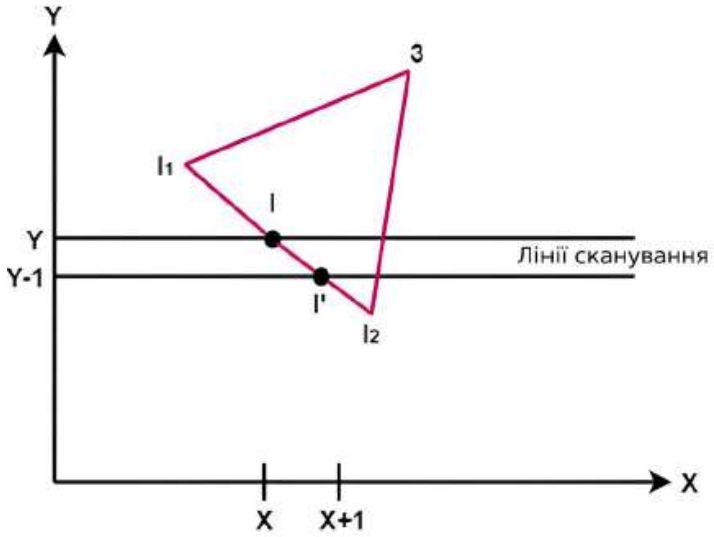


Рисунок 2.3 – Покрокова інтерполяція інтенсивності вздовж ребра полігону для суміжних сканувальних ліній.

На основі цього можна сформулювати вираз для розрахунку кольорової інтенсивності за методом Гуро на краю полігону в точці з координатами (x, y):

$$\mathrm{I} = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2 \tag{2.4}$$

Після виведення формули для обчислення інтенсивності кольору на краю полігона, наступним кроком є визначення значення інтенсивності для краю на наступній сканувальній лінії (Y-1). Для цього застосовується така формула:

$$I^1 = I + \frac{I_2 - I_1}{y_1 - y_2} \tag{2.5}$$

Подібні розрахунки виконуються для визначення інтенсивностей у послідовних горизонтальних позиціях пікселів уздовж кожної сканувальної лінії. У разі, якщо поверхню необхідно відтворити у кольорі, значення інтенсивності для кожного кольорового компонента обчислюються у вершинах. Метод затінення за Гуро можна ефективно поєднувати з алгоритмами усунення прихованих поверхонь для рендерингу лише видимих полігонів на кожній лінії сканування. Приклад моделі, зафарбованої методом Гуро, представлений на наступному зображенні:



Рисунок 2.4 – Застосування методу Гуро для затінення тривимірної моделі

Затінення за методом Гуро дозволяє уникнути розривів інтенсивності, характерних для моделі з постійним затіненням, проте має і певні недоліки. Зокрема, світлові відблиски на поверхні можуть мати неприродну форму, а лінійна інтерполяція інтенсивності іноді призводить до появи яскравих або темних смуг, відомих як смуги збігу (Match bands).

2.2 Адаптації та вдосконалення техніки рендерингу за методом Гуро

Зафарбовування за Гуро – це широко використовуваний метод у комп'ютерній графіці, який допомагає зробити 3D-моделі більш реалістичними, згладжуючи колірні зміни на поверхнях полігонів. Він працює шляхом першого обчислення кольору в кожній вершині (куті) полігону. Потім він інтерполює ці кольори по решті полігону, створюючи плавне змішування.

Цікаво, що дослідження [15] показали: зміни інтенсивності кольору (ΔI_r та ΔI_b) під час зафарбовування за Гуро є постійними, незалежно від того, чи рухаєтеся ви вздовж растрової лінії (горизонтального ряду пікселів), чи вздовж країв полігону. Ця постійність означає, що вам не потрібно перераховувати ці зміни для кожної окремої растрової лінії, що робить процес ефективнішим. Далі ми дослідимо, як ці зміни інтенсивності кольору вздовж країв полігонів співвідносяться зі змінами вздовж растрових ліній.

$$\Delta I_{\Gamma} = \frac{(I_{\text{A}} - I_{\text{C}}) * \Delta Y_{BC} - (I_B - I_{\text{C}}) * \Delta Y_{AC}}{\Delta X_{AC} * \Delta Y_{BC} - \Delta X_{BC} * \Delta Y_{AC}} \quad (2.6)$$

$$\Delta I_{\text{B}} = \frac{(I_B - I_{\text{C}}) * \Delta Y_{BC} - (I_A - I_{\text{C}}) * \Delta Y_{AC}}{\Delta X_{AC} * \Delta Y_{BC} - \Delta X_{BC} * \Delta Y_{AC}} \quad (2.7)$$

$$\Delta I_{\text{B}} = \Delta I_{\Gamma} + \Delta I_{\text{B}} \quad (2.8)$$

Аналіз показав сталість змін інтенсивностей ΔI_H, ΔI_V та ΔI_D у відповідних напрямках: горизонтальному, вертикальному та діагональному. Згідно з ілюстрацією на рисунку 2.5, інтенсивності кольорів у вершинах трикутника мають позначення I_{A}, I_B та I_C.

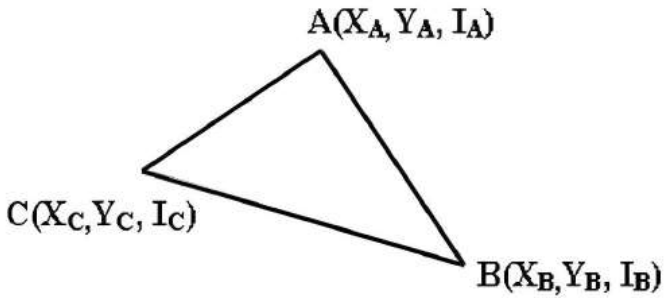


Рисунок 2.5 – Схема паралельного рендерингу інтенсивностей

Описана властивість є основою для розпаралелювання алгоритму рендерингу Гуро із використанням тріангуляції Серпінського першого порядку. Процедура передбачає розбиття початкового трикутника на чотири менші області шляхом сполучення середин його сторін (рис. 2.6). Внаслідок такого поділу формуються три ідентичні за формою та орієнтацією трикутники та один, що є їхнім дзеркальним відображенням.

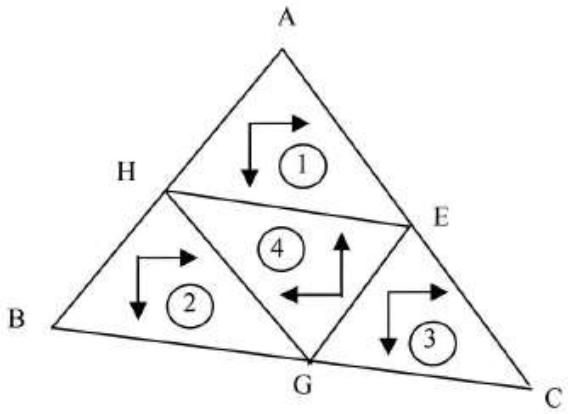


Рисунок 2.6 – Напрямки затінення трикутників

Виходячи з умови сталості градієнта інтенсивності кольору в межах трикутника, обчислення параметрів затінення стає можливим для одного з дочірніх трикутників з подальшою екстраполяцією результатів на решту. Таким чином, достатньо виконати растеризацію лише для одного еталонного трикутника (позначеного як 1), після чого застосувати аналогічні операції до інших внутрішніх трикутників. Слід зауважити, що трикутники 1, 2 і 3 характеризуються однаковим напрямком зафарбовування, натомість центральний трикутник 4 обробляється у протилежному напрямку.

Ключовою передумовою для початку растеризації є попереднє визначення координат усіх вершин новоутворених трикутників. Алгоритм інтерполяції інтенсивності для внутрішніх пікселів еталонного трикутника може бути реплікований для інших трикутників за умови підстановки відповідних вершин. Однак, оскільки тріангуляція базується на поділі ребер навпіл, виникає проблема утворення нерастеризованих пікселів у випадках, коли довжини сторін вихідного трикутника є непарними. Для вирішення цієї задачі було проведено аналіз альтернативних методів розбиття, в результаті якого було обрано оптимальний варіант [16].

Альтернативний підхід до оптимізації продуктивності зафарбовування за методом Гуро ґрунтується на роздільній обробці парних та непарних пікселів у межах кожного рядка сканування. У рамках циклу підготовки даних, для кожної лінії сканування обчислюються значення інтенсивності для початкового (I_1) та наступного (I_2) пікселів. Інтенсивність для першого пікселя рядка ($I_{РБ}$) визначається методом інтерполяції значень уздовж базових ребер трикутника. Натомість інтенсивність I_2 розраховується за допомогою наступної формули:

$$I_2 = I_{РБ} + \Delta I_Г \qquad (2.9)$$

Альтернативний підхід до покращення якості візуалізації, що ґрунтується на методі затінення Гуро, передбачає ідентифікацію та подальшу обробку відблисків, що виникають на ребрах трикутників. Розглянемо трикутник,

визначений вершинами з координатами (x_0,y_0), (x_1,y_1), та (x_2,y_2), для яких задані відповідні вектори нормалей N_0, N_1, та N_2.

У контексті комп'ютерної графіки часто приймається припущення, що джерело світла та спостерігач розташовані на нескінченності. Це припущення спрощує модель освітлення, оскільки вектор напрямку світла L залишається постійним для всіх точок поверхні трикутника. Емпірично встановлено, що за таких умов максимальні значення інтенсивності кольору, що відповідають відблискам, локалізуються у точках, розташованих на ребрах трикутника. Ці точки задовольняють специфічному рівнянню:

$$t_1 = \frac{\cos\gamma\cos\phi - \cos\beta}{(\cos\phi - 1)(\cos\beta + \cos\gamma)}, \tag{2.10}$$

$$t_2 = \frac{\cos\gamma\cos\phi - \cos\lambda}{(\cos\phi - 1)(\cos\lambda + \cos\gamma)}, \tag{2.11}$$

$$t_3 = \frac{\cos\beta\cos\vartheta - \cos\lambda}{(\cos\vartheta - 1)(\cos\beta + \cos\lambda)} \tag{2.12}$$

Візуалізація відповідності кутів між векторами нормалей показано на рисунку 2.7.

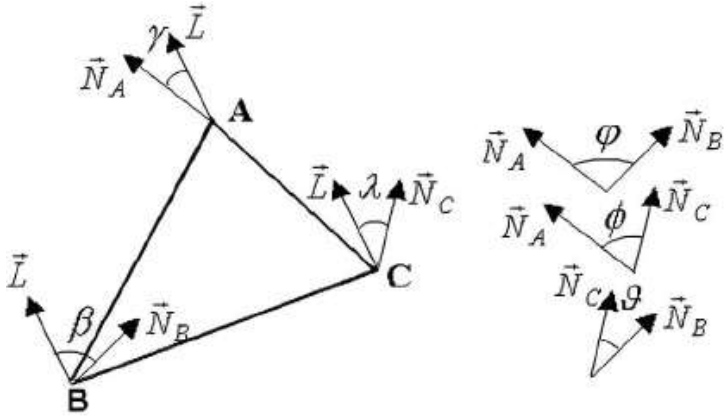


Рисунок 2.7 – Вектори нормалі трикутника ABC

Параметричні змінні t_1,t_2,t_3, що знаходяться в інтервалі [0,1], використовуються для визначення точок на ребрах вихідного трикутника. Залежно від співвідношення між інтенсивністю кольору в цих точках та

заздалегідь визначеними пороговими значеннями, початковий трикутник може бути адаптивно розділений на кілька підтрикутників. Ці новостворені трикутники потім зафарбовуються з використанням методу Гуро. Такий підхід значно підвищує якість візуалізації, оскільки дозволяє виявляти перетин світлових променів з одним або кількома ребрами трикутника. На відміну від класичного методу затінення Гуро, де таке виявлення відсутнє, що призводить до втрати відблисків та появи значних артефактів, запропонований метод дозволяє їх зберегти.

З метою підвищення продуктивності процесу зафарбовування, пропонується комбінований підхід, що інтегрує елементи методу Гуро. Зокрема, лінія розгортки може бути сегментована на відрізки, довжина яких є кратною степеням двійки. Для проміжних пікселів кожного такого відрізка інтенсивність кольору визначається за методом Гуро. Довжина цих відрізків може адаптивно змінюватися, залежно від кривизни поверхні та напрямку вектора H (вектора половинного кута).

При обчисленні як дзеркальної, так і дифузної складових інтенсивності кольору для точок на тривимірній поверхні, критично важливою є нормалізація векторів нормалей. Це зумовлено тим, що косинуси кутів, які використовуються у функціях затінення, легко визначаються за допомогою скалярного добутку одиничних векторів. Беручи до уваги, що нормалізація необхідна для вектора спостерігача, вектора джерела світла та вектора нормалі, виникає проблема оптимізації обчислювальних витрат на цю ресурсомістку процедуру. У зв'язку з цим, запропоновано метод адаптивної нормалізації вектора нормалі. Згідно з цим методом, нормалізація виконується лише тоді, коли спостерігається значна зміна інтенсивності кольору, що дозволяє зменшити загальне обчислювальне навантаження.

2.3. Розробка методів розпаралелення зафарбовування за Гуро

У класичному підході до рендерингу для визначення координат точок, що формують ребра трикутника, використовуються два лінійні інтерполятори. На

основі розрахованих приростів інтенсивності вздовж кожного ребра виконується процес інтерполяції. Для обчислення координат внутрішніх точок трикутника застосовується ще один лінійний інтерполятор, який функціонує по рядках зображення.

Інтенсивність кольору у внутрішніх точках визначається за допомогою відповідного інтерполятора, який може бути об'єднаний із тим, що обчислює інтенсивність вздовж ребер. Таким чином, традиційна реалізація рендерингу потребує щонайменше двох лінійних інтерполяторів: одного для побудови коду інтерполяції та іншого – для інтерполяції по рядках (рис. 2.8).

Одним з варіантів реалізації паралельного процесу зафарбовування трикутного полігону є метод, що базується на використанні зустрічних інтерполяторів (рис. 2.9).

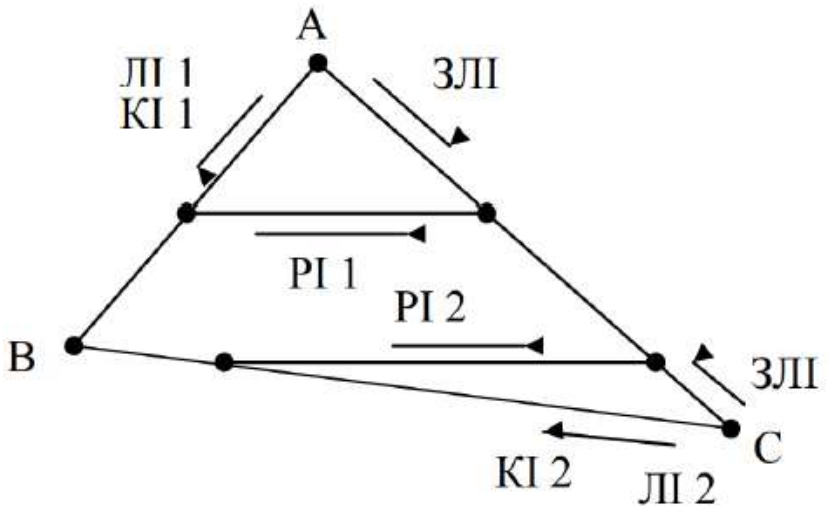


Рисунок 2.8 – Організація обчислювального процесу за допомогою взаємодії інтерполяційних елементів

Суть методу полягає в тому, що значення інтенсивності в кожному рядку трикутника, який обробляється, обчислюються за допомогою двох кодових інтерполяторів, що працюють назустріч один одному – від початкової та кінцевої точок рядка. Лінійна інтерполяція вздовж сторін трикутника виконується аналогічно класичному підходу до рендерингу. Запропонована

схема дозволяє вдвічі спростити процес інтерполяції всередині рядка та рівномірно розподілити похибку рендерингу між обома інтерполяторами.

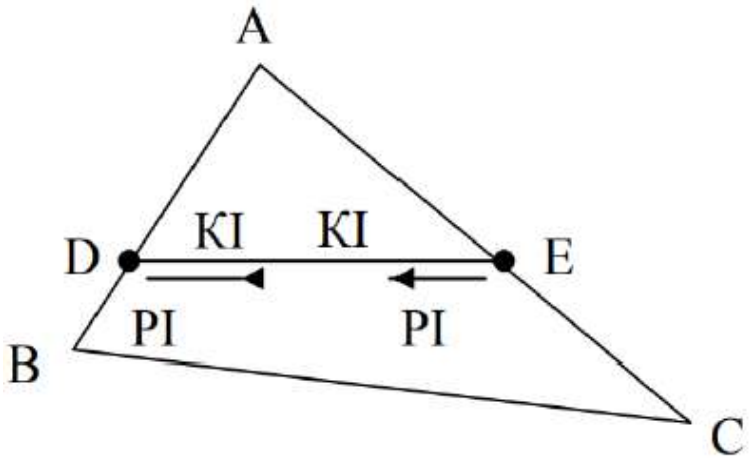


Рисунок 2.9 – Метод затінення на основі зустрічного інтерполювання кодів

Для впровадження цього підходу, на відміну від традиційної методики, достатньо використовувати один кодовий інтерполятор та один інтерполятор для рядка. Додатково, для підвищення швидкості рендерингу пропонується незалежне обчислення інтенсивностей пікселів для парних та непарних точок у сканованому рядку (див. рис. 2.10).

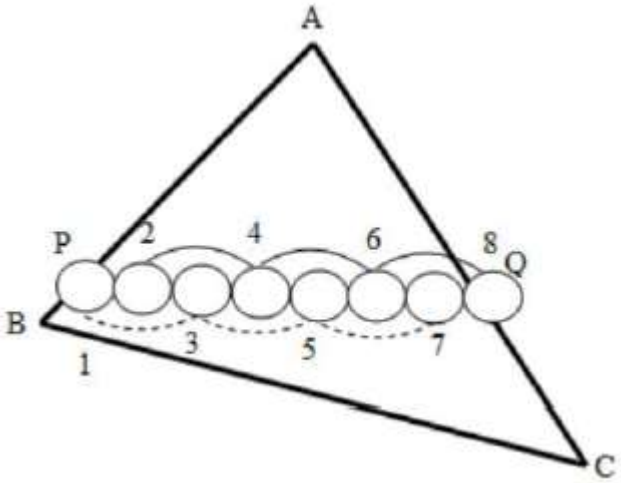


Рисунок 2.10 – Процес знаходження рівнів кольору в парних і непарних точках

Запропонований метод підвищення ефективності інтерполяції інтенсивності полягає у використанні двох кодових інтерполяторів. Перший

інтерполятор відповідає за обчислення інтенсивності виключно для непарних точок у рядку сканування трикутника, при цьому значення приросту інтенсивності подвоюється (тобто використовується 2ΔI). Другий кодовий інтерполятор призначений для визначення інтенсивності парних точок того ж рядка. Такий підхід дозволяє подвоїти швидкість кодової інтерполяції сканованого рядка. Слід зазначити, що реалізація цього методу вимагає додаткових апаратних ресурсів: одного кодового інтерполятора та одного рядкового інтерполятора.

Цей принцип можна масштабувати для більшої кількості кодових інтерполяторів, хоча оптимальна їх кількість зазвичай не перевищує чотирьох. Це пояснюється тим, що значення приросту інтенсивності, кратні 2ΔI, 3ΔI, 4ΔI, можуть бути обчислені за допомогою однієї операції зсуву та одного додавання. Збільшення кількості точок, для яких необхідно визначати приріст інтенсивності, веде до непропорційного зростання обчислювальних витрат, що, як правило, є недоцільним.

Для подальшого підвищення продуктивності рендерингу за методом Гуро, окрім розпаралелювання, пропонується використовувати метод зафарбовування трикутника, що базується на аналізі властивостей теореми про середню лінію трикутника.

Розглянемо трикутник ABC (рис. 2.11). Проводяться три середні лінії: KL, LM, та KM. Середня лінія трикутника визначається як відрізок, що сполучає середини двох його сторін.

Властивості середньої лінії трикутника:

- а) Вона паралельна одній із сторін трикутника.
- б) Її довжина дорівнює половині довжини тієї сторони трикутника, до якої вона паралельна.

Ці геометричні властивості можуть бути використані для оптимізації алгоритмів зафарбовування, зокрема для ефективнішого розподілу обчислень по точках трикутника.

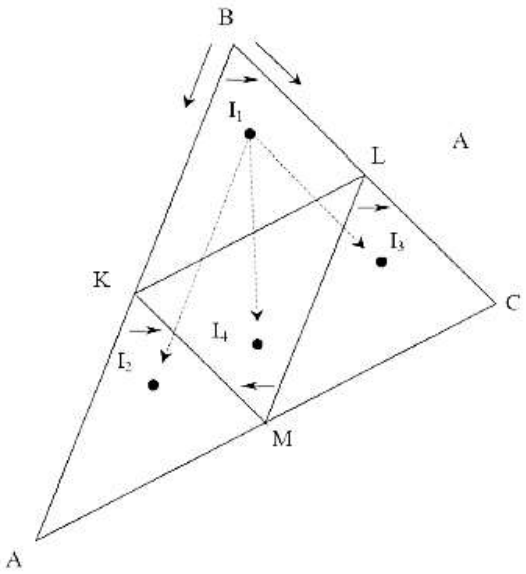


Рисунок 2.11 – Паралельне зафарбовування трикутника

Щоб застосувати цей метод, необхідно спочатку довести, що трикутники AKM і MLC є рівними. Для цього розглянемо й порівняємо відповідні сторони цих трикутників:

$$AM=MC=0.5*AC \quad (2.13)$$

$$AK=ML=0.5*AB \quad (2.14)$$

$$KM=LC=0.5*BC \quad (2.15)$$

З огляду на рівність відповідних сторін розглянутих трикутників, можна стверджувати про їхню конгруентність. Аналогічно, трикутники AKM, MLC, KBL та LMK також є конгруентними.

Одним із ключових етапів для підвищення продуктивності формування тривимірних зображень є прискорення інтерполяції. У процесі інтерполяції трикутника приріст значення вздовж його сторони залишається постійним. Це дозволяє припустити, що після інтерполяції кожна точка одного трикутника буде відрізнятися на певну константу від відповідної точки іншого трикутника.

Таким чином, пропонується зафарбовувати лише один з отриманих трикутників, а інші зафарбовувати шляхом реплікації результату. З метою забезпечення максимальної реалістичності, першим буде зафарбовуватися верхній з отриманих трикутників. Наступним етапом є обчислення констант зсуву для кожної з трьох середніх точок (K, L, M):

:

K:

$$\begin{cases} \Delta X = \dfrac{(X_A - X_B)}{2} \\ \Delta Y = \dfrac{(Y_A - Y_B)}{2} \\ \Delta I = \dfrac{(I_A - I_B)}{2} \end{cases} \quad (2.16)$$

L:

$$\begin{cases} \Delta X = \dfrac{(X_C - X_B)}{2} \\ \Delta Y = \dfrac{(Y_C - Y_B)}{2} \\ \Delta I = \dfrac{(I_C - I_B)}{2} \end{cases} \quad (2.17)$$

M:

$$\begin{cases} \Delta X = \dfrac{(X_A - X_C)}{2} \\ \Delta Y = \dfrac{(Y_A - Y_C)}{2} \\ \Delta I = \dfrac{(I_A - I_C)}{2} \end{cases} \quad (2.18)$$

Застосування представленого методу забезпечує значне прискорення процесу зафарбовування, приблизно в чотири рази. Це пояснюється тим, що площа одного з малих трикутників, який є основою для реплікації, становить лише чверть від площі початкового трикутника. Крім того, паралельне зафарбовування решти частин трикутника дозволяє уникнути надмірних часових витрат, що сприяє загальній оптимізації рендерингу.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИСОКОПРОДУКТИВНОГО ФОРМУВАННЯ ЗОБРАЖЕНЬ

3.1. Розробка моделі та алгоритмів роботи програмного забезпечення

Для забезпечення кращого розуміння функціонування програмних модулів, призначених для високопродуктивного формування зображень з використанням методу Гуро, розроблено модель роботи системи у вигляді UML діаграми діяльності (рис. 3.1).

Згідно з представленою діаграмою, початкова взаємодія користувача з програмним застосунком передбачає вибір одного з двох варіантів: імпорт тривимірної моделі у форматі .obj або вибір базової моделі з доступної бібліотеки.

У разі вибору імпорту моделі, система переходить у стан імпортування тривимірної моделі. В іншому випадку активується стан вибору базової моделі для подальших операцій.

Наступним кроком після імпортування тривимірної моделі у робоче середовище програмних засобів є перевірка імпортованої моделі на відповідність розширенню .obj. У випадку коректного формату розширення, програмний застосунок надає доступ до подальших етапів роботи; в іншому випадку генерується повідомлення про помилку.

Далі виконується аналіз тривимірної моделі, що включає збір інформації про її вершини та полігони. Після цього етапу початкові стани (імпортування та вибору базової моделі) конвертуються в однакові функції.

Після успішного аналізу модель відображається у робочому середовищі програмного застосунку. На цьому етапі програмне забезпечення очікує вибір функції від користувача. Доступні функції включають: відображення каркасу тривимірної моделі (відображення контурів полігонів); зміна кольору зафарбовування; зафарбовування за методом Гуро; зміна параметрів згладжування; зміна параметрів прозорості.

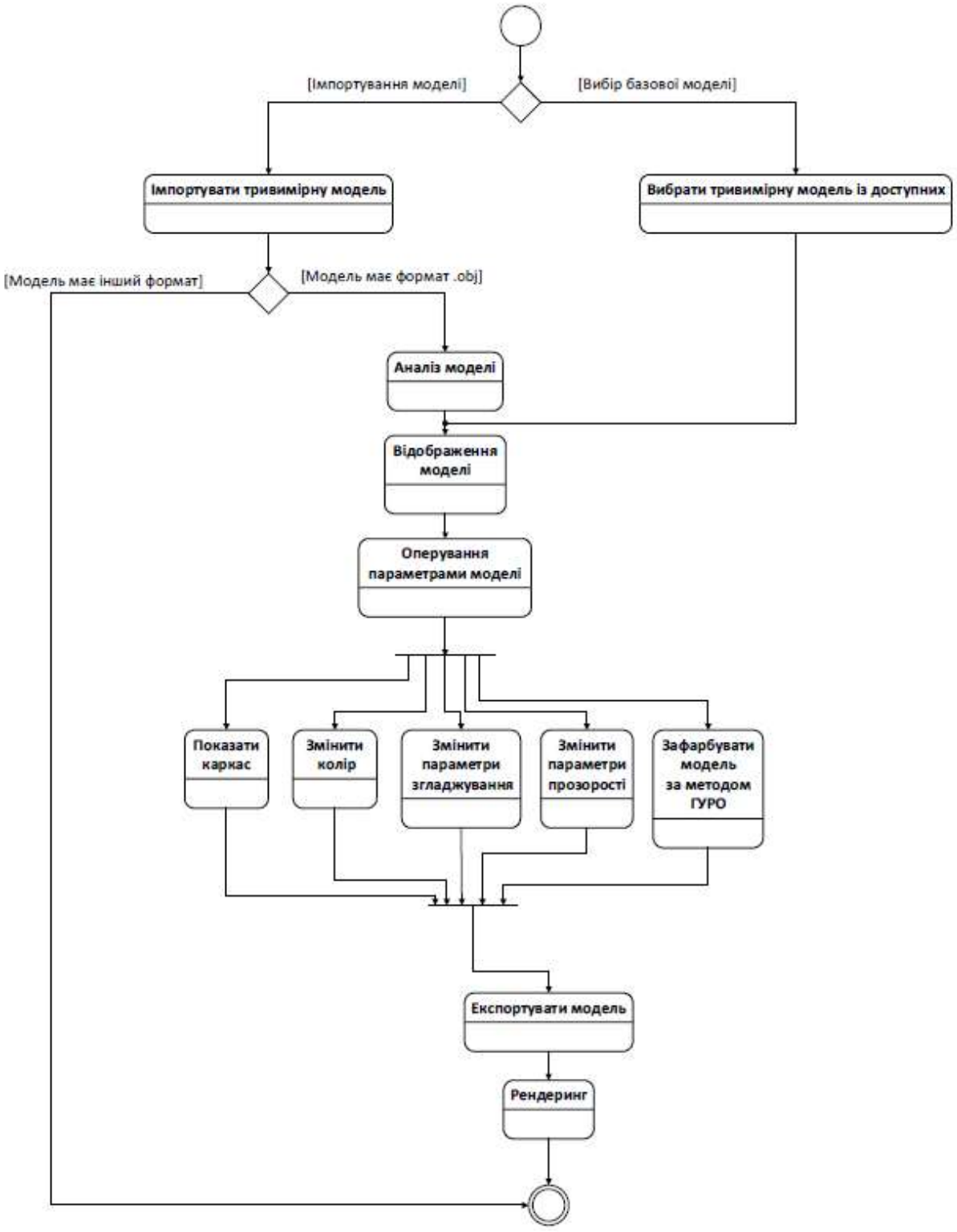


Рисунок 3.1 – Діаграма діяльності програмних засобів

При виборі користувачем певної функції, програмний застосунок автоматично застосовує відповідну функцію до тривимірної моделі, а потім відображає внесені зміни на моделі.

Завершальним етапом є можливість експорту зображення з тривимірної фігури. При виборі цієї функції програмне забезпечення переходить у стан

рендерингу та зберігає створене зображення. Після цього користувач може відкривати, переглядати та використовувати сформоване зображення.

Для створення програмного застосунку, що реалізує функціонал високопродуктивного формування зображень за методом Гуро, необхідно спроектувати загальний алгоритм роботи програмного модуля (див. рис. 3.2).

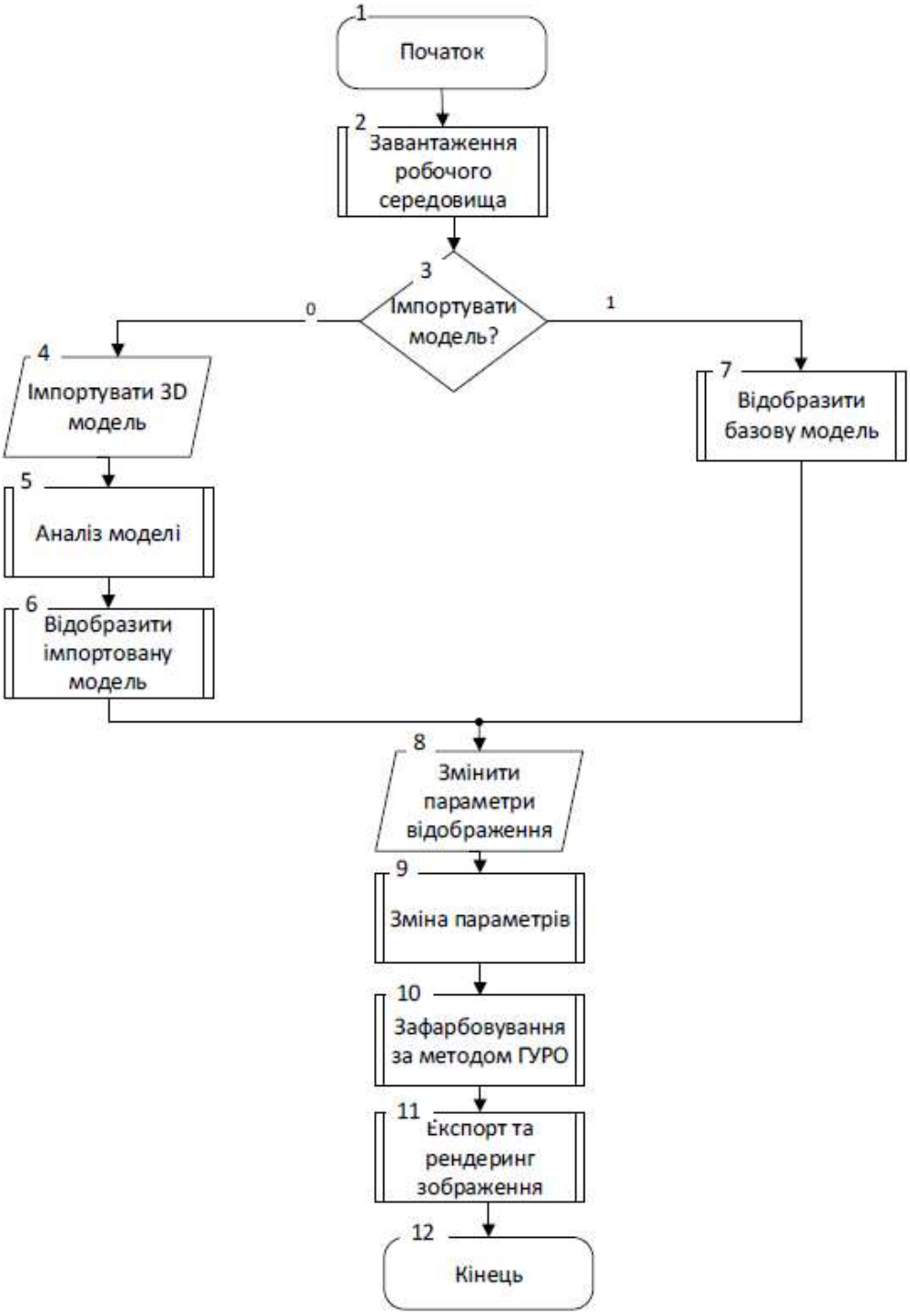


Рисунок 3.2 – Алгоритм роботи додатку

Відповідно до цього алгоритму, першим етапом є завантаження робочого середовища програми, яке надає користувачеві доступ до функціональних можливостей створеного застосунку.

Наступним етапом користувач може імпортувати власну тривимірну модель або вибрати одну з доступних базових моделей, що надаються програмним застосунком. У разі вибору базової моделі, вона миттєво завантажується у робоче середовище програми, після чого система очікує подальших дій користувача, зокрема вибору необхідних функцій.

Якщо ж користувач вирішує імпортувати власну 3D-модель, активується модуль імпорту, який дозволяє обрати відповідний файл з локальної файлової системи. Після імпортування програмне забезпечення виконує аналіз структури моделі та здійснює її тріангуляцію. Потім відбувається сортування вершин полігонів та формування відповідної множини точок. Імпортована модель відображається в інтерфейсі, після чого система переходить у режим очікування подальшого налаштування від користувача.

Користувач може змінити параметри відображення моделі, вибрати функцію зміни кольору та здійснити зафарбовування 3D-фігури за методом Гуро. Після застосування змін та виконання затінення, користувач має можливість експортувати результат. Для цього запускається модуль рендерингу, який обробляє модель і зберігає зображення у вказаній директорії.

Для кращого розуміння логіки роботи модуля зафарбовування за методом Гуро було розроблено окремий алгоритм (рис. 3.3). Відповідно до нього, першим кроком є завантаження даних обраної 3D-моделі. Далі автоматично виконується тріангуляція з наступним розбиттям полігонів на набір трикутників і побудовою відповідної множини тривимірних вершин.

Після формування множини точок виконується їх сортування, необхідне для забезпечення точного зафарбовування з урахуванням положення джерела освітлення. Потім запускається цикл, у межах якого проходиться по всіх вершинах моделі та здійснюється розрахунок параметрів для затінення.

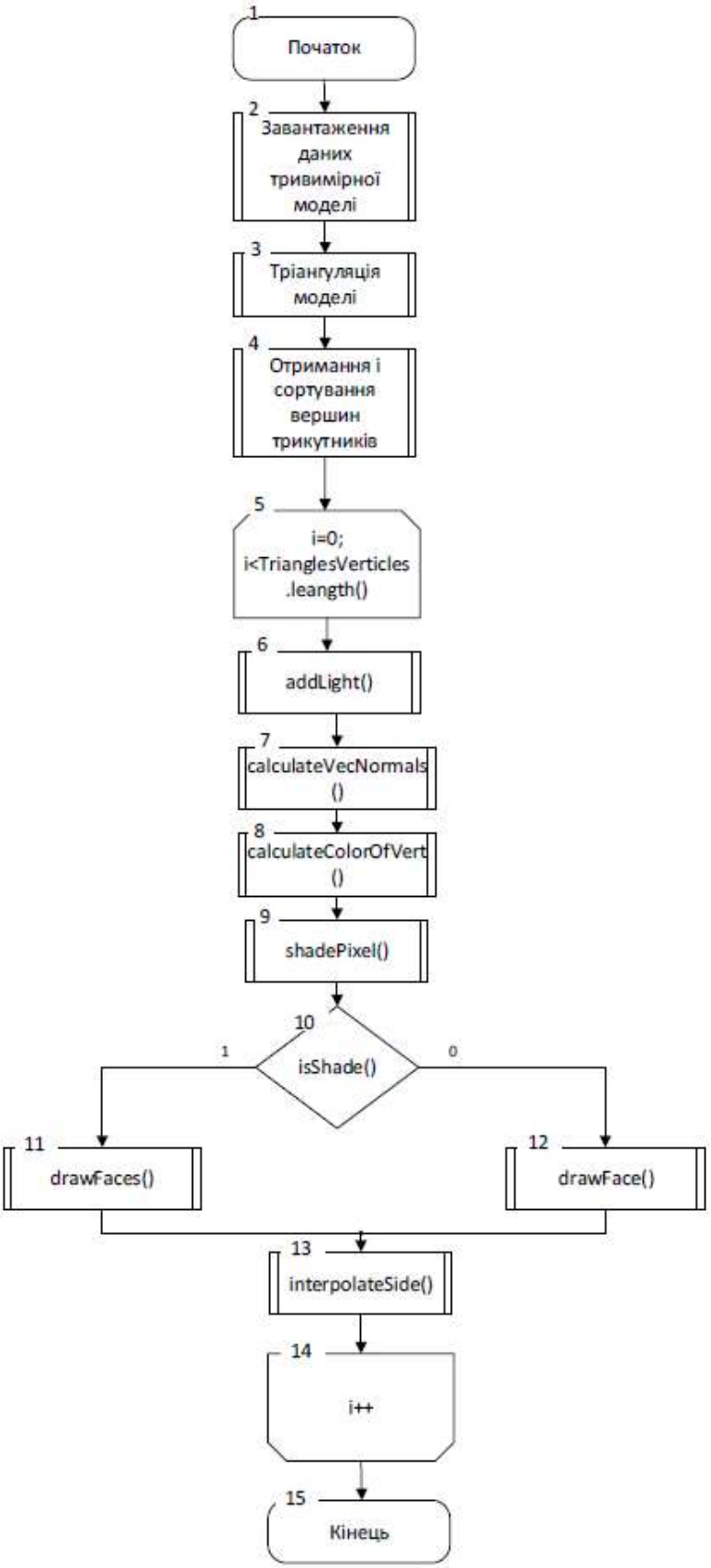


Рисунок 3.3 – Алгоритм зафарбовування тривимірної моделі

На початку кожної ітерації циклу визначається положення джерела світла, розраховується відстань до вершини та інтенсивність його впливу. Далі

відбувається обчислення векторів нормалей для конкретної вершини, що дозволяє досягти рівномірного та природного зафарбовування поверхні.

Далі виконується розрахунок кольору вершини на основі положення джерела світла та відповідного вектора нормалі. Після визначення кольорової компоненти активується модуль, що відповідає за зафарбовування окремого пікселя вершини. Наступним кроком здійснюється перевірка доступності джерела світла відносно вектора нормалі. Якщо напрям світла збігається з орієнтацією нормалі та потрапляє на її поверхню, виконується зафарбовування площини окремого трикутника з урахуванням відбиття світла.

3.2. Розробка модуля затінення тривимірної моделі за методом Гуро.

Під час створення модуля для зафарбовування тривимірної моделі методом Гуро необхідно враховувати низку технічних аспектів. Насамперед слід проаналізувати структуру самої 3D-моделі та виконати її тріангуляцію для точного визначення вершин, що братимуть участь у процесі зафарбовування. Для реалізації цих завдань було створено клас mainDraw, який відповідає за розрахунок позицій усіх вершин, розміщення джерела світла, обробку необхідних параметрів моделі, а також виконує функції зафарбовування та візуалізації.

Зазначений клас, що представлений на рисунку 3.4, забезпечує виконання основних операцій, необхідних для ефективної взаємодії з тривимірною сценою. Він ініціалізує параметри моделі, обробляє координати вершин, враховує положення джерела світла та інші ключові характеристики, що впливають на вигляд зафарбованої моделі. Такий підхід дозволяє забезпечити керованість і гнучкість процесу рендерингу.

Для оптимізації обробки було прийнято рішення замість використання трикутників застосовувати чотирикутні полігони довільної форми. При цьому визначення координат виконується на основі відсортованої множини всіх вершин моделі.

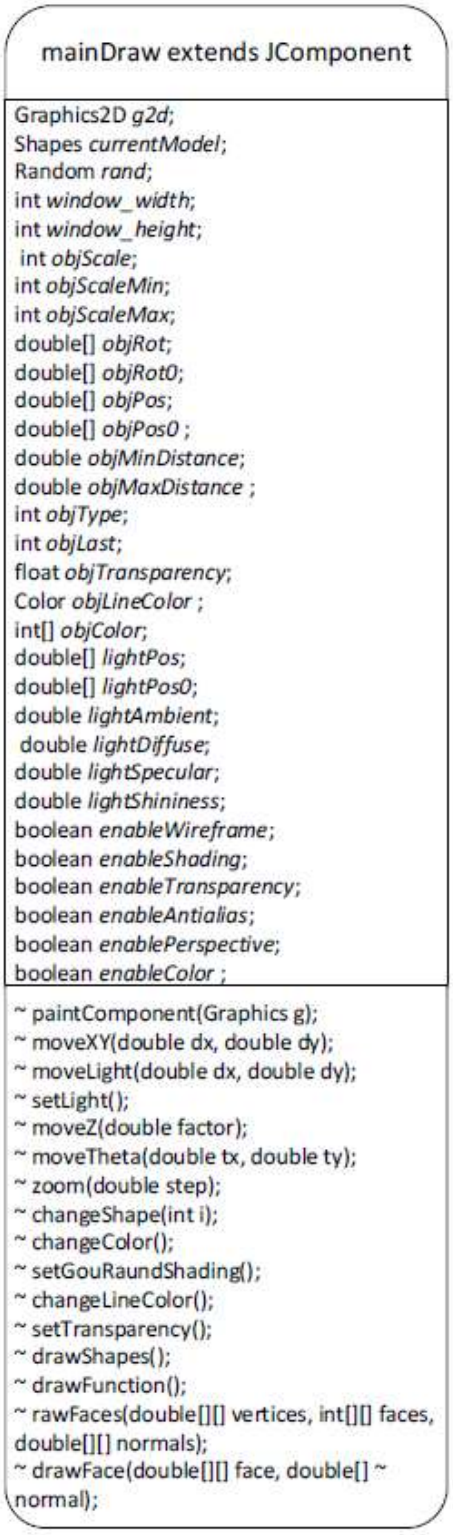


Рисунок 3.4 – Клас mainDraw

Після побудови окремих множин для кожної з координат наступним кроком є обчислення векторів нормалей, що реалізовано у методі drawFunction (див. рис. 3.5). Цей метод визначає положення чотирьох вершин полігона та виконує розрахунок нормального вектора. Після цього викликається функція, яка відповідає за зафарбовування відповідного полігона.

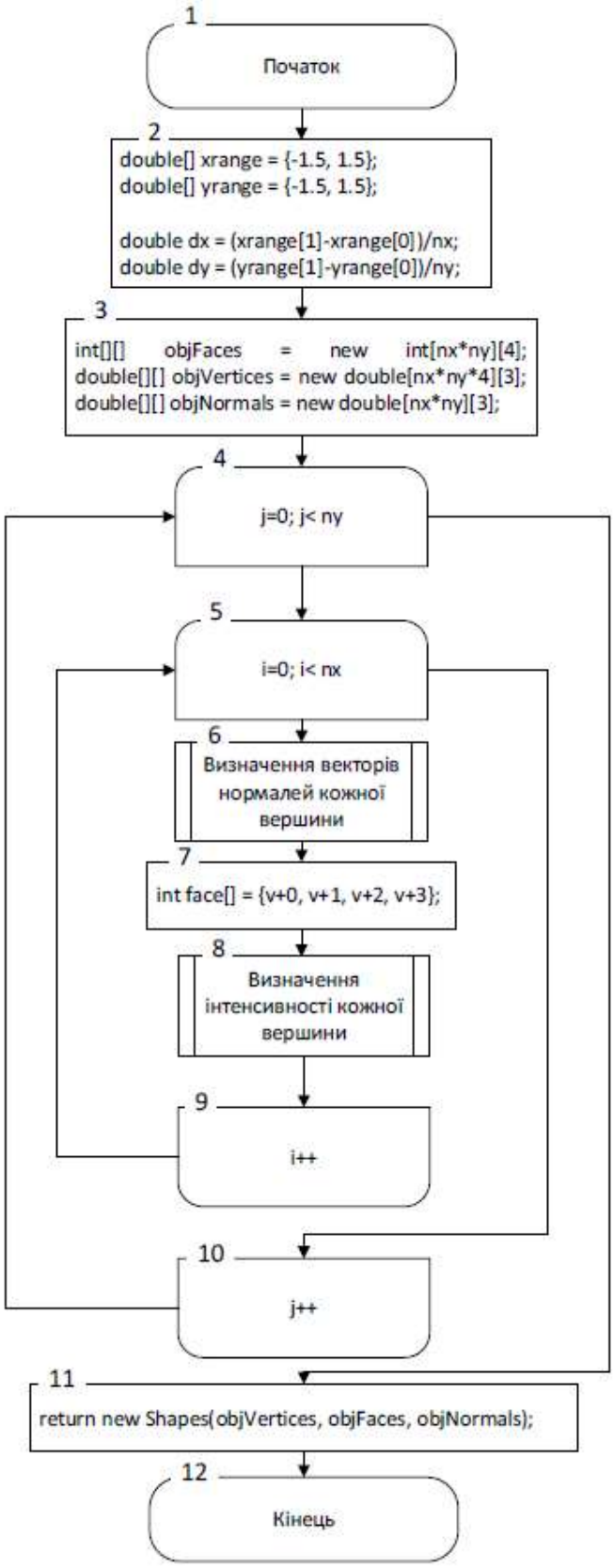


Рисунок 3.5 – Створення полігонів

Наступним етапом є візуалізація сформованих полігонів на тривимірній площині. Це необхідно для наочного відображення процесу зафарбовування, а також для розрахунку рівнів освітлення та затінення кожного полігона. Для

цього у кутах полігонів створюються точки, а інформація про кожну вершину – зокрема її координати, вектори нормалі та набір пікселів, що знаходяться в межах полігона – записується у відповідні масиви. Після цього визначається положення джерела світла та виконується перевірка його впливу на пікселі полігона (див. рис. 3.6).



Рисунок 3.6 – Функція drawSphere

Далі, у випадку відсутності впливу освітлення, викликається метод drawFaces (рис. 3.7), який відповідає за зафарбовування площини відповідним кольором. Метод використовує координати пікселів, що належать полігону, а також обчислені вектори нормалі для моделювання розподілу світла та надання площині більш реалістичного вигляду.

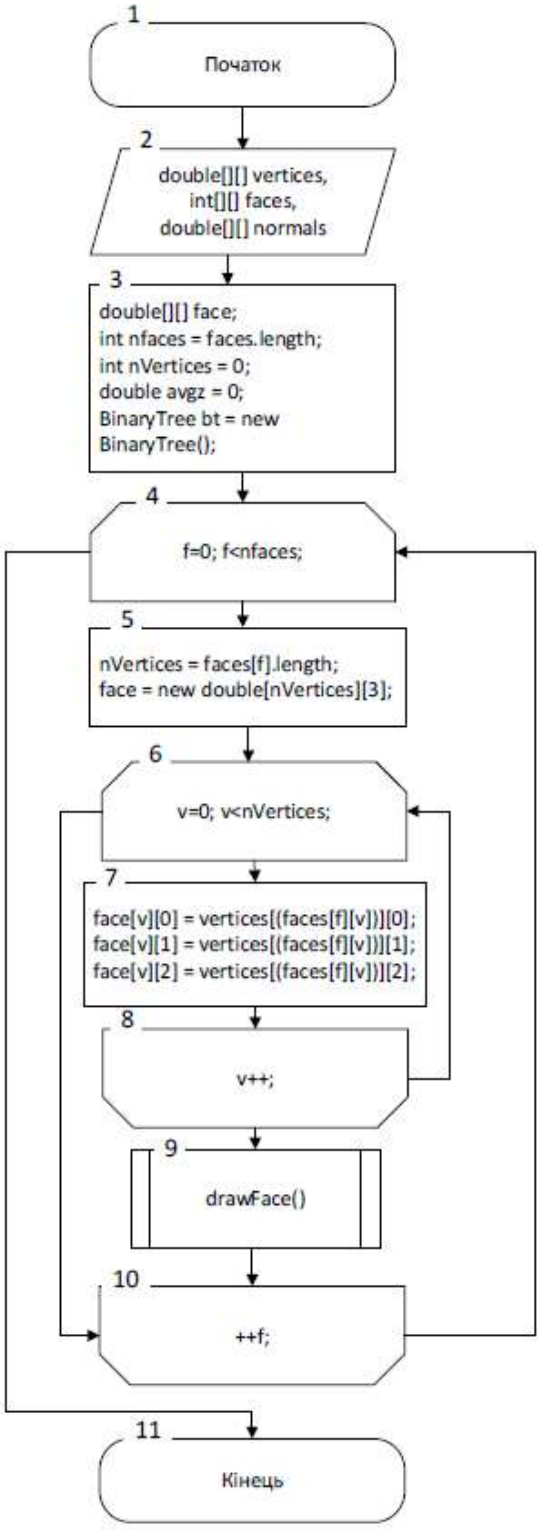


Рисунок 3.7 – Метод drawFaces

У разі наявності освітлення на поверхні полігона викликається метод drawFace (рис. 3.8), який здійснює зафарбовування площини з урахуванням положення джерела світла, дифузної складової та рівня яскравості. Після цього в межах методу проводиться обчислення впливу світла на кожен окремий піксель полігона, що супроводжується викликом функції shade (рис. 3.9). У

цьому методі визначається колір кожного пікселя на основі положення джерела світла, колірної складової та інтенсивності освітлення. Для досягнення плавного переходу освітлення по поверхні полігона використовуються обчислення векторів нормалей та інтерполяція значень освітлення.



Рисунок 3.8 – Метод drawFace

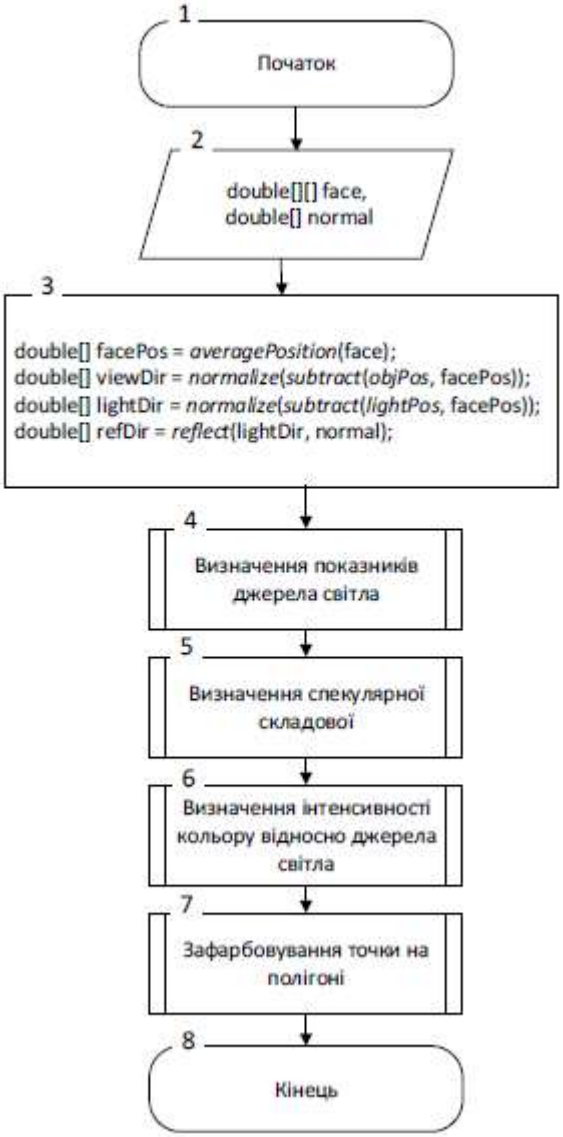


Рисунок 3.9 – Метод shade

3.3. Створення модуля для високоефективного генерування зображень

Основним завданням модуля високопродуктивного формування зображень є забезпечення точного та швидкого рендерингу. Для реалізації цієї мети було створено клас SimpleInterpreter, який під час ініціалізації автоматично визначає положення джерела світла та виконує зафарбовування тривимірної моделі (рис. 3.10).

Окрім цього, було реалізовано окрему функцію, призначену для рендерингу пікселів полігонів з використанням методу Гуро (рис. 3.11). Ця функція забезпечує реалістичне зафарбовування, враховуючи положення

джерела світла, та дозволяє точно передати світлотіньові переходи на поверхні моделі.

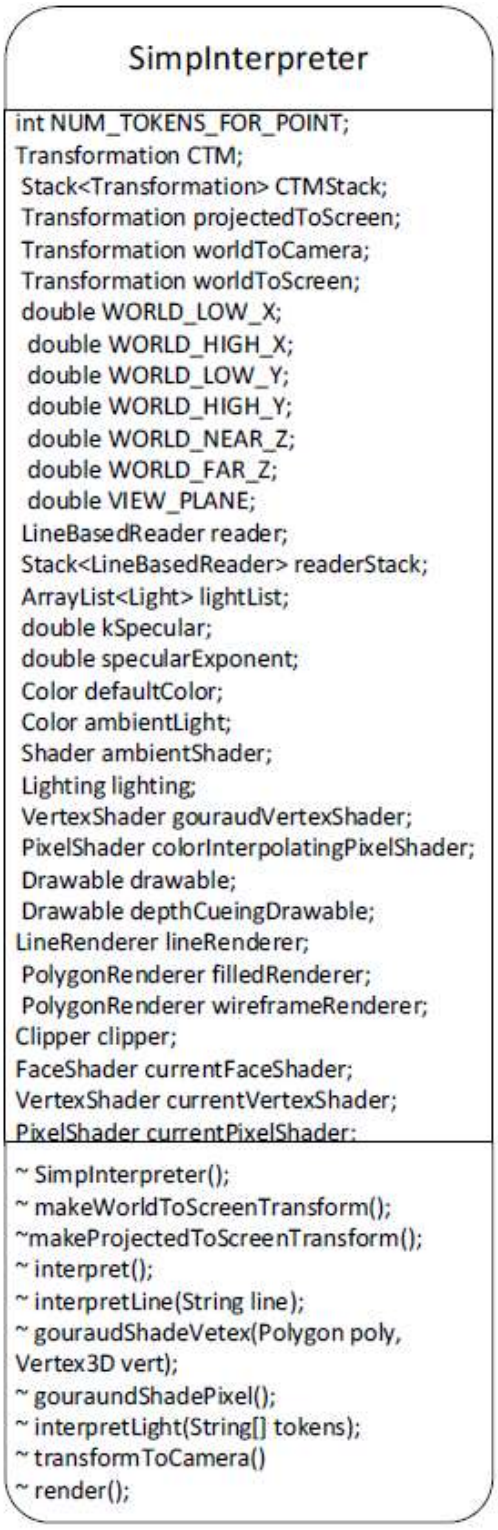


Рисунок 3.10 – Клас SimpInterpreter

Наступним етапом є виконання інтерполяції за вектором нормалі відбитого світла. Для цього застосовується метод класу interpretLight, який визначає напрям відбиття світла з урахуванням розташування камери.

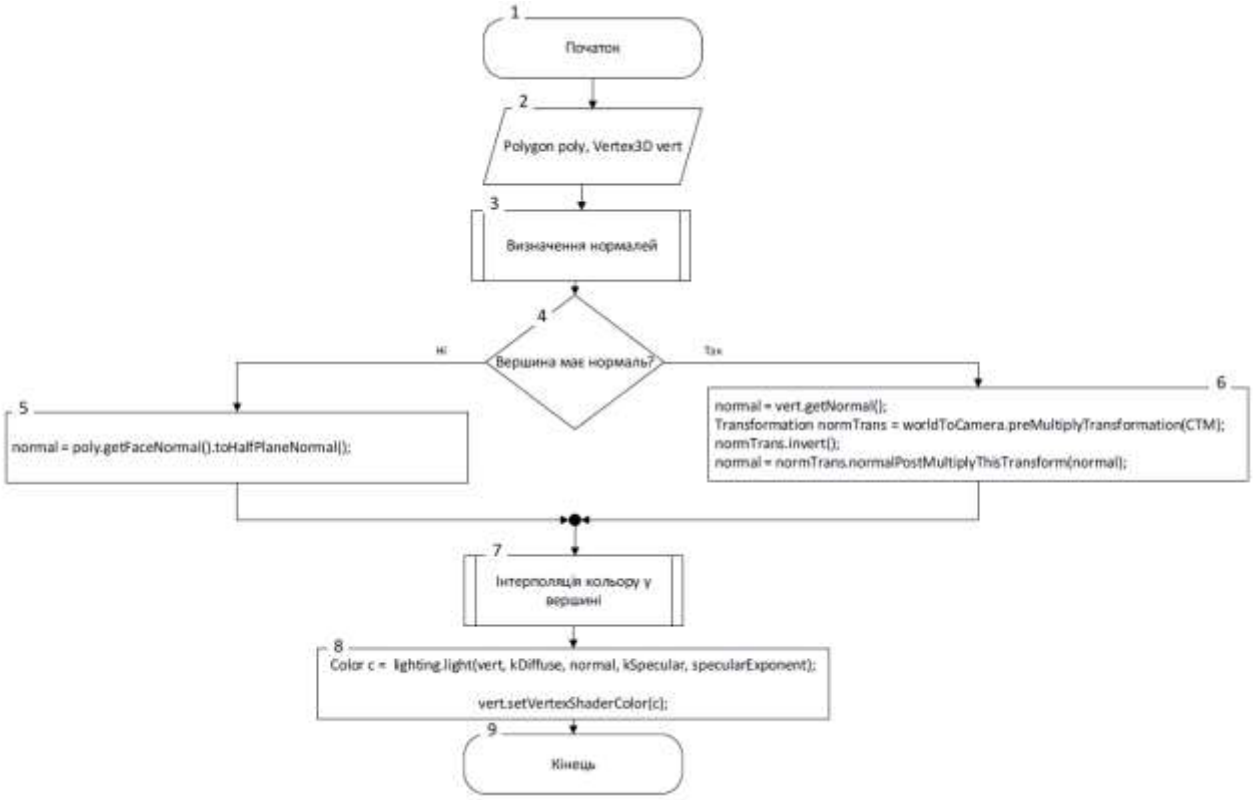


Рисунок 3.11 – Функція представлення пікселів за Гуро

Це дає змогу врахувати положення спостерігача та забезпечити реалістичне відображення світлових ефектів.

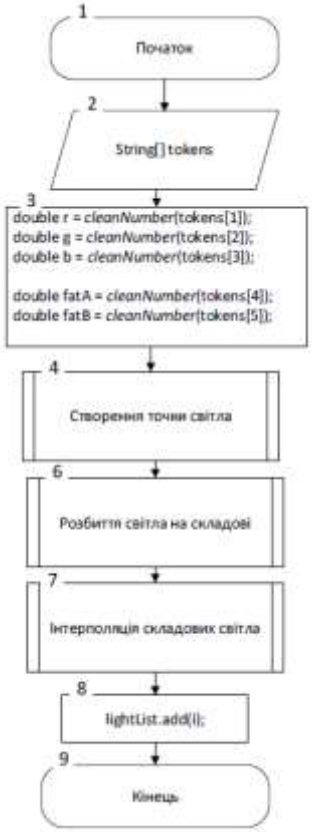


Рисунок 3.12 – Процес розрахунку відбиття світла

Після цього відбувається рендеринг кожного полігона тривимірної моделі (рис. 3.13). У процесі використовуються інтерполяція положення камери, обчислення впливу джерела світла та застосування функції зафарбовування до кожного пікселя 3D-об’єкта.

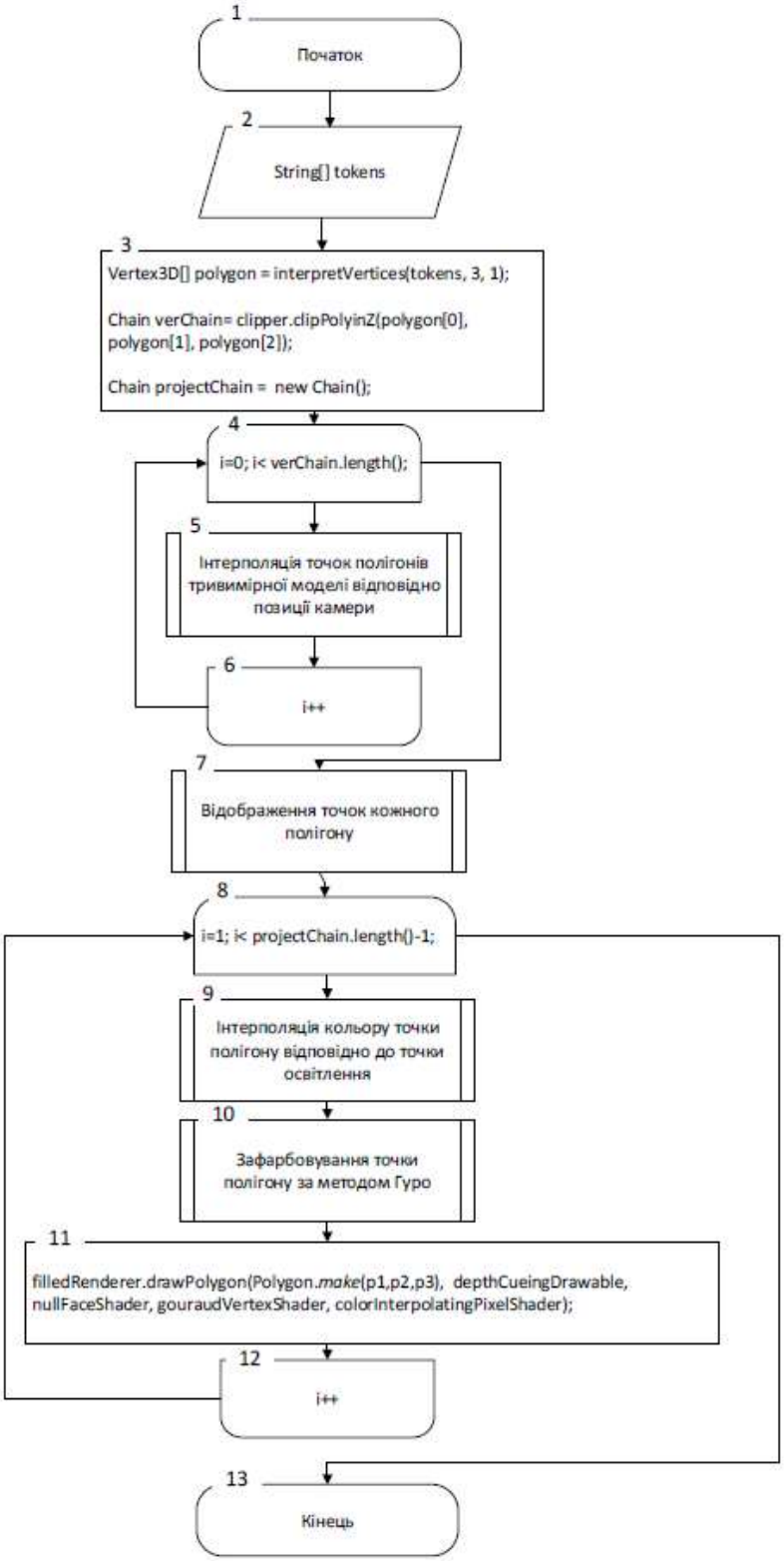


Рисунок 3.13 – Візуалізація полігональних поверхонь 3D-моделі

3.4. Тестування системи оптимізованого рендерингу

Тестування відіграє ключову роль у життєвому циклі розробки програмного забезпечення, оскільки дозволяє виявити та усунути помилки, забезпечуючи відповідність системи заданим вимогам і технічним характеристикам [18]. Його основне призначення – перевірка як функціональних, так і нефункціональних властивостей програми, таких як надійність, продуктивність, зручність для користувача та рівень безпеки. Крім того, тестування охоплює процеси верифікації та валідації, які забезпечують відповідність продукту очікуванням замовника та коректність реалізації функціоналу.

Верифікація полягає в тому, щоб перевірити, наскільки реалізований програмний код відповідає встановленим технічним вимогам. Для цього результати роботи програми порівнюють із попередньо визначеними критеріями. Цей процес може включати як статичні методи (аналіз коду, перегляд реалізації), так і динамічні (виконання тестів окремих компонентів).

Валідація, у свою чергу, спрямована на те, щоб переконатися, що програмне забезпечення дійсно відповідає потребам користувачів і коректно функціонує в умовах реального використання. Це включає тестування зручності інтерфейсу, перевірку основного функціоналу та відповідність очікуванням замовника. Валідація підтверджує, що розроблена система підходить для практичного застосування [19].

Існує декілька підходів до тестування, які відрізняються рівнем обізнаності про внутрішню структуру програми [20]:

1. Метод "чорної скриньки" – передбачає тестування на основі специфікацій без урахування внутрішньої реалізації. Його мета – перевірити правильність виконання функцій, взаємодію з користувачем та обробку вхідних даних.
2. Метод "білої скриньки" – передбачає повну обізнаність тестувальника щодо внутрішньої структури програми. Тестування здійснюється шляхом

аналізу коду, що дозволяє перевірити логіку, контрольні потоки та виявити потенційні помилки або вразливості.

3. Метод "сірої скриньки" – комбінує елементи обох попередніх методів. Тестувальник має часткову інформацію про внутрішню реалізацію та використовує її для створення ефективніших тестів. Це дозволяє поєднати переваги точності "білої скриньки" та практичної орієнтації "чорної скриньки".

У межах проєкту було обрано метод "білої скриньки". Програмне забезпечення було встановлено на ОС Windows 10 для проведення тестування.

На першому етапі перевіряється реакція програми на некоректні вхідні дані. На рисунку 3.14 наведено приклад відповіді системи при спробі імпорту 3D-моделі у невірному форматі – таким чином демонструється здатність ПЗ обробляти помилки.

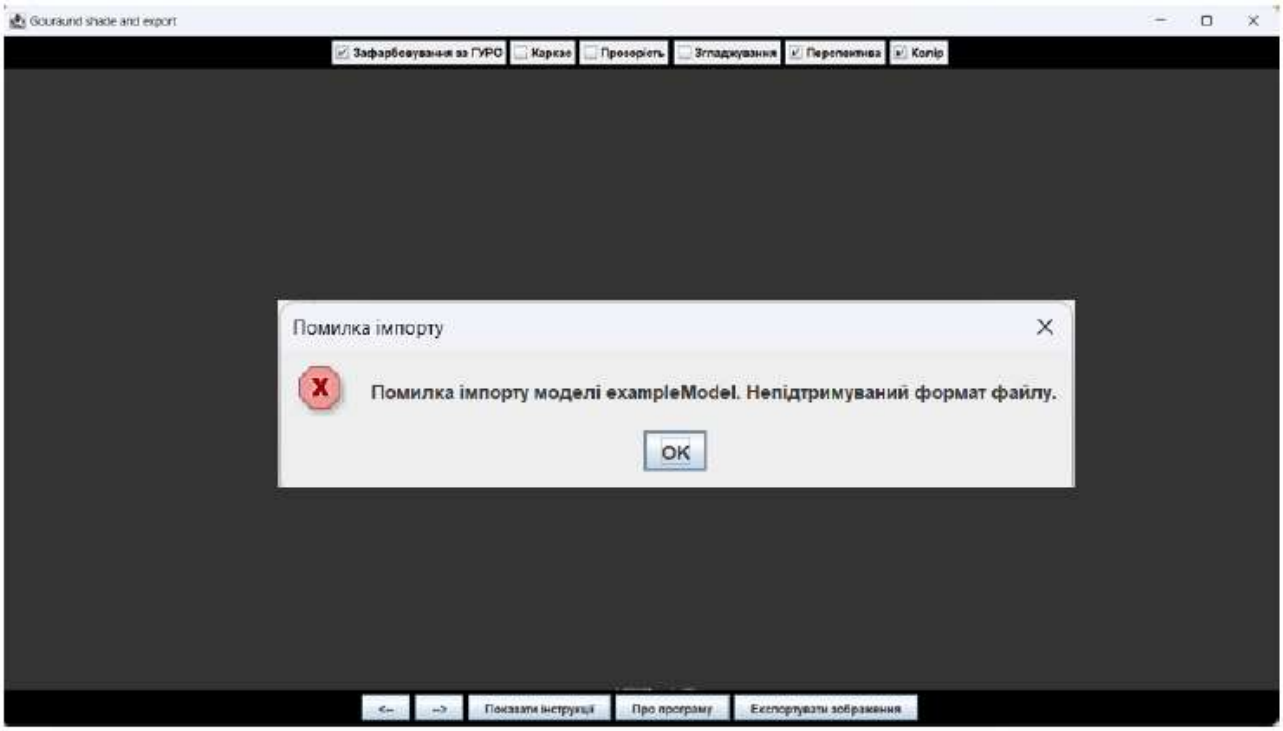


Рисунок 3.14 – Завантаження 3D-моделі у нестандартному форматі

Під час спроби імпорту файлу з непідтримуваним форматом система виводить відповідне повідомлення про помилку, яке повідомляє користувача про неправильний формат файлу та вказує на необхідність використання формату, сумісного з програмним забезпеченням. Це дозволяє уникнути збоїв у

роботі програми та сприяє розумінню користувачем подальших дій для успішного імпорту.

Після завершення тестування на стійкість до помилкових дій було вирішено перевірити функціональність імпорту коректного формату тривимірної моделі. На рисунку 3.15 показано інтерфейс програми в момент успішного імпорту моделі. У цьому випадку система правильно обробила вхідні дані, коректно завантаживши та відобразивши тривимірну модель, що підтверджує працездатність і надійність модуля імпорту.

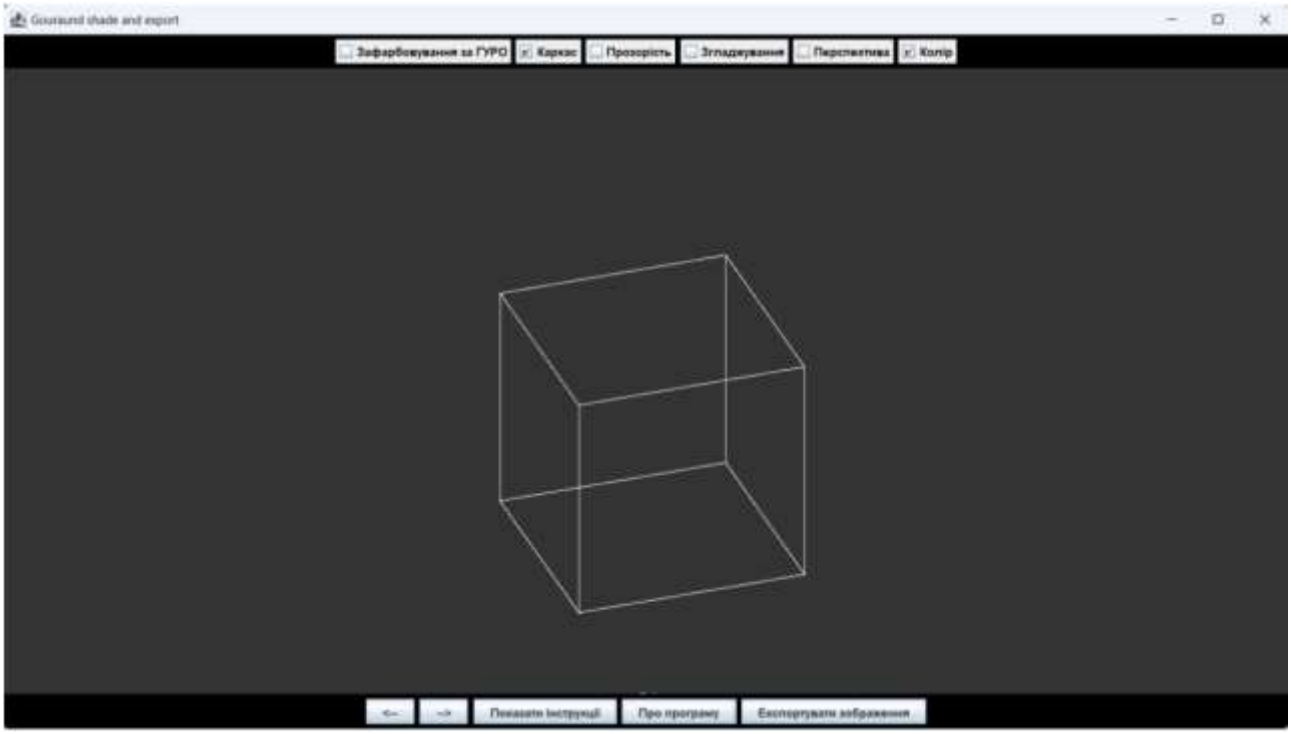


Рисунок 3.15 – Інтерфейс програми в момент успішного імпорту моделі

Наступним кроком у процесі тестування є перевірка працездатності реалізованих функцій програмного забезпечення, тобто проведення функціонального тестування. Цей етап передбачає оцінку правильності роботи ключових можливостей програми, зокрема маніпулювання тривимірною моделлю, зміни масштабу, обертання, а також використання доступних інструментів і опцій у функціональному меню.

Після успішного імпорту 3D-моделі користувач отримує доступ до всіх можливостей керування моделлю через інтерфейс програми, зокрема може змінювати масштаб, кут огляду та орієнтацію моделі. На рисунку 3.16

демонструється взаємодія з тривимірним об'єктом, яка включає обертання, масштабування та інші дії, що дозволяють детально переглядати модель з різних сторін. Така функціональність сприяє зручності користування програмою та забезпечує ефективне виконання поставлених завдань.

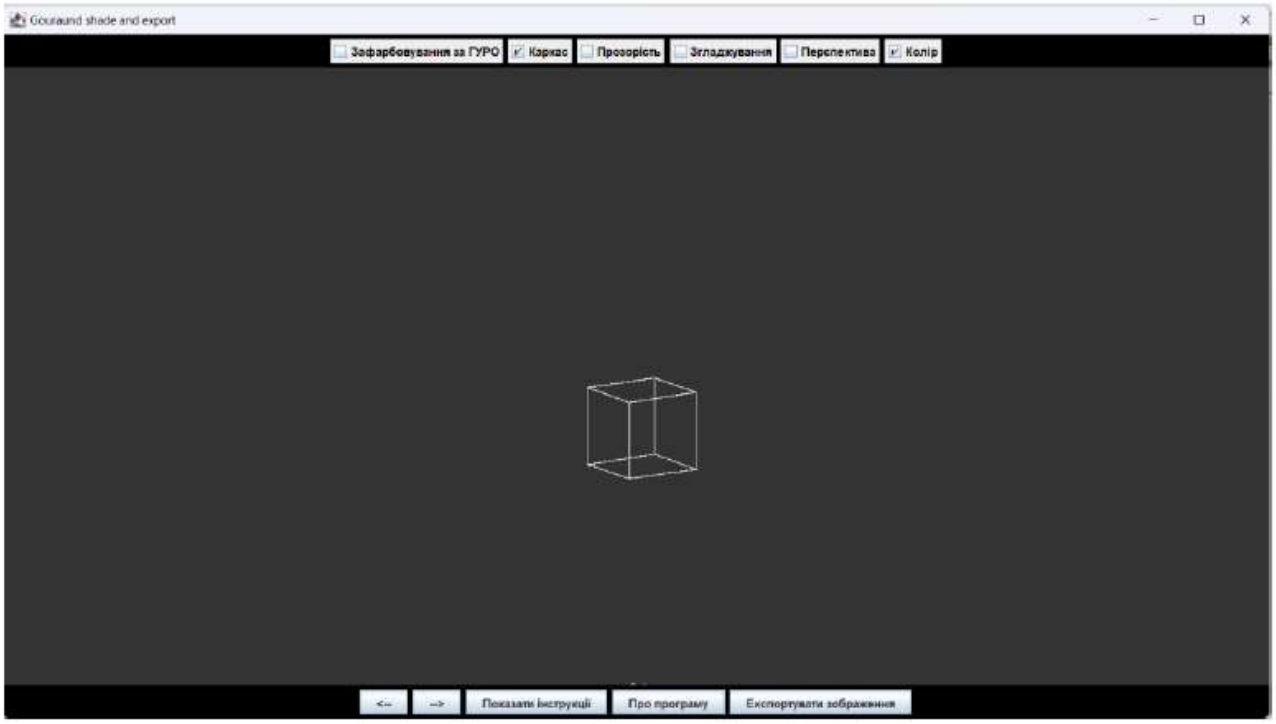


Рисунок 3.16 – Взаємодія з тривимірним об'єктом

Наступним етапом є верифікація коректності реалізації функціональних можливостей розробленого програмного забезпечення. На цьому рівні основну увагу зосереджено на функціональному тестуванні, метою якого є підтвердження відповідності реалізованих функцій встановленим вимогам та специфікаціям. Особливе місце серед функціональних можливостей займає модуль зафарбовування тривимірних об'єктів за методом Гуро, який забезпечує плавне інтерпольоване заповнення полігональних поверхонь для досягнення фотореалістичності відображення.

На рисунку 3.17 представлено результати тестування зазначеної функції. Проведене тестування засвідчило здатність програмного забезпечення виконувати зафарбовування тривимірних моделей з високою продуктивністю та без появи візуальних артефактів, що свідчить про ефективність реалізованого алгоритму та високу якість візуалізації.

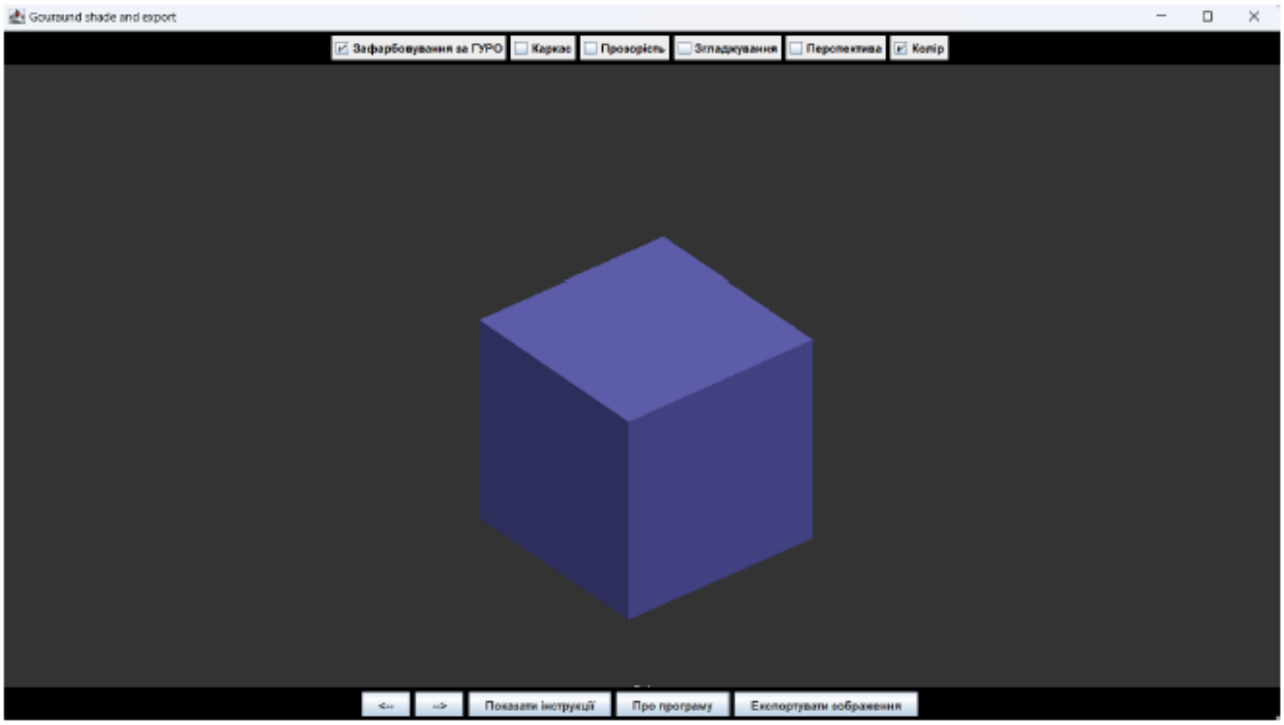


Рисунок 3.17 – Процес рендерингу

У процесі тестування програмного забезпечення також було перевірено коректність функціонування механізму зміни параметрів прозорості, що надає можливість користувачеві здійснювати візуальний аналіз внутрішніх елементів тривимірної моделі під час її зафарбовування. Такий функціонал є особливо корисним для поглибленого дослідження геометричної структури об’єкта. Додатково було протестовано модуль згладжування, який дозволяє усувати візуальні нерівності полігональної сітки та підвищує естетичну якість зображення за рахунок плавного переходу між гранями. Результати відповідних випробувань, представлені на рисунку 3.18, засвідчують правильну та стабільну роботу зазначених функцій.

Окрему увагу в рамках тестування було приділено аналізу сумісності програмного забезпечення з різними версіями операційних систем і мінімальною конфігурацією апаратного забезпечення. Зокрема, програмний продукт було протестовано на платформах Windows 10, Windows 8.1 та MacOS. Перевірка проводилася як на високопродуктивних системах, так і на пристроях із обмеженими ресурсами. Результати підтвердили стабільну роботу додатку в усіх зазначених середовищах, що свідчить про його високий ступінь оптимізації, кросплатформеність і ефективність використання апаратних ресурсів.

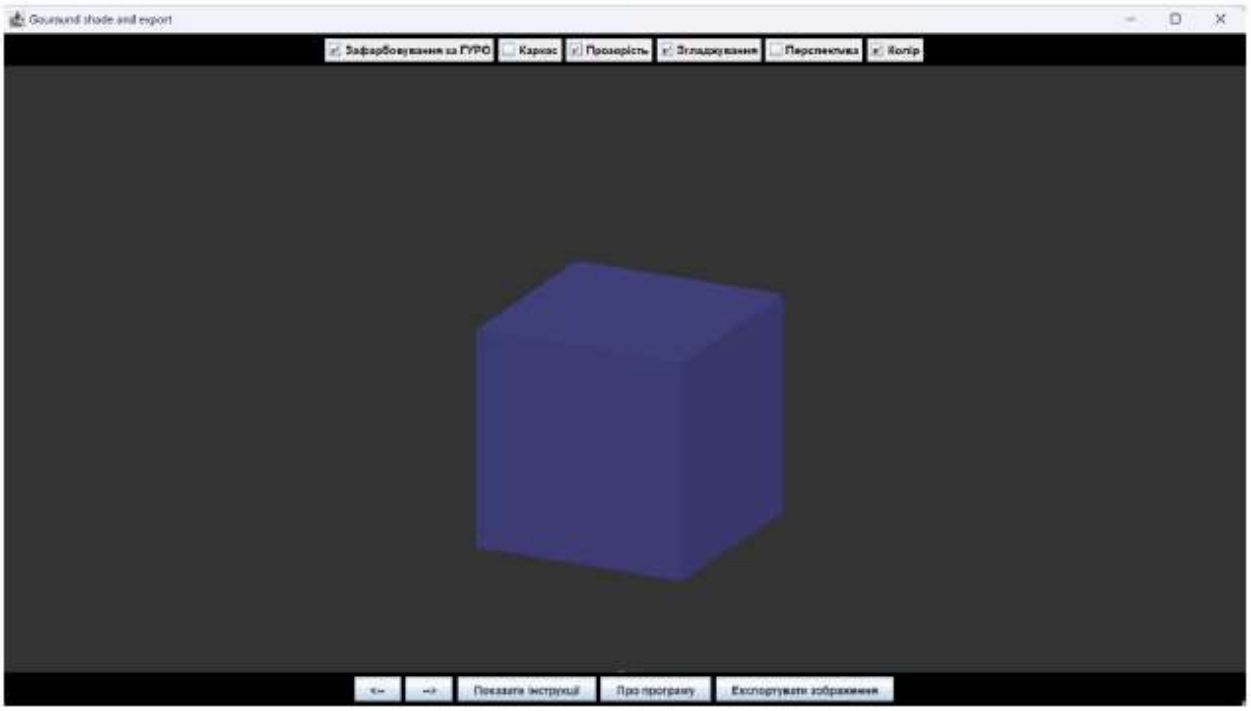


Рисунок 3.18 – Оцінка ефективності зміни параметрів візуалізації

Було також виконано тестування функціональності зміни кольору зафарбовування. Для активації цієї опції користувачеві необхідно натиснути кнопку "Колір", після чого відбувається зміна кольору зафарбовування на обраний варіант. Результати даного тестування, представлені на рисунку 3.19, підтверджують правильну роботу механізму зміни кольорового оформлення, що свідчить про стабільність та функціональну відповідність даного елементу інтерфейсу.

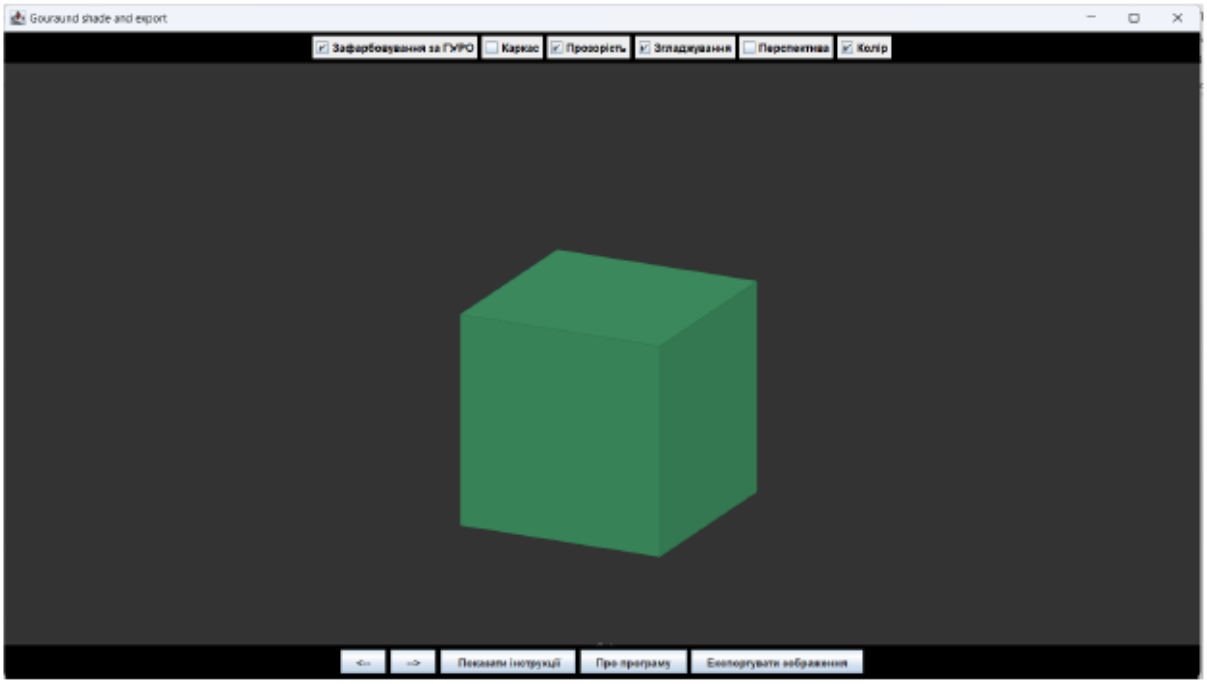


Рисунок 3.19 –Демонстрація механізму зміни кольорового оформлення

З метою підвищення якості візуалізації тривимірної моделі передбачено використання параметра перспективного проєціювання. Даний функціонал забезпечує додавання ефекту перспективи, що суттєво покращує сприйняття глибини сцени та наближає зображення до реалістичного вигляду. На рисунку 3.20 наведено результати тестування цієї функції, які підтверджують її успішну реалізацію та ефективність у покращенні відображення тривимірного об’єкта.

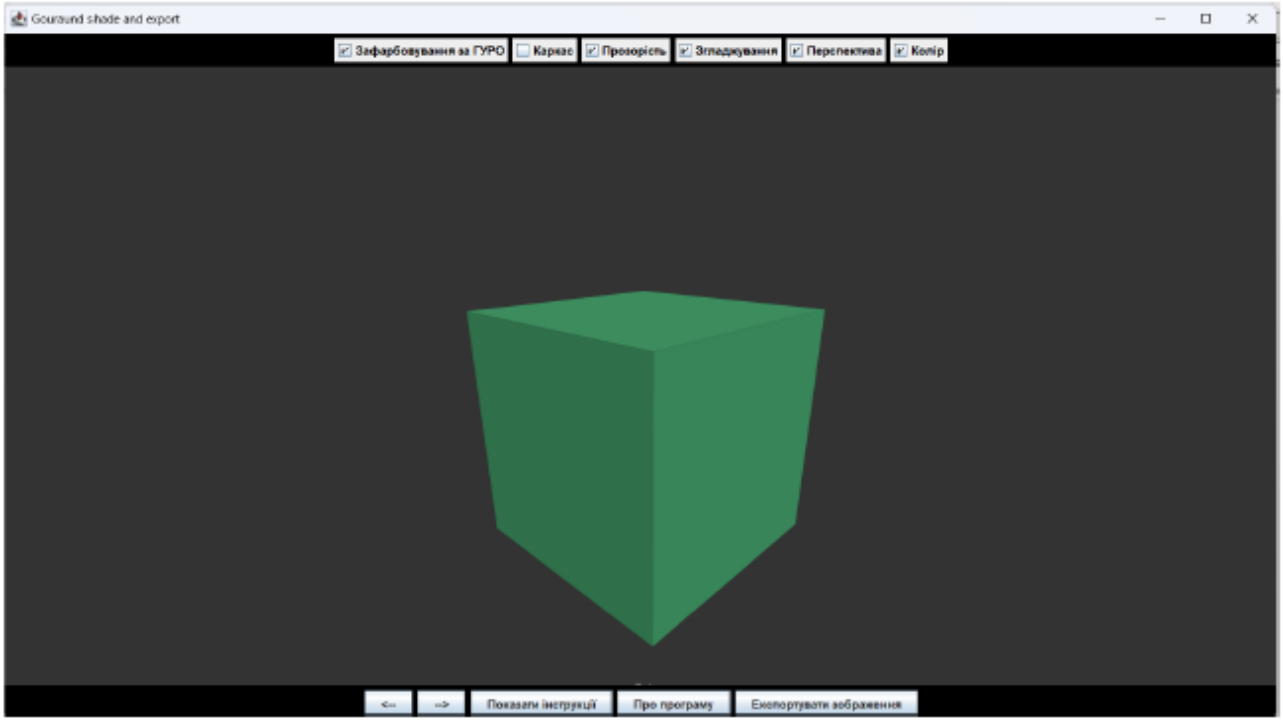


Рисунок 3.20 – Проецес тестування функції перспективного відображення

Програмне забезпечення також забезпечує підтримку базових тривимірних моделей. Для доступу до першої з них користувач може скористатися кнопкою "Назад" у нижній частині інтерфейсу. На рисунку 3.21 представлено результати тестування цієї можливості, що засвідчили коректне завантаження та відображення базової 3D-моделі.

Крім перевірки функціональності, було здійснено аналіз продуктивності зафарбовування моделей методом Гуро, зокрема часу, необхідного для виконання операції та її візуалізації у тривимірному просторі. За результатами тестування встановлено, що програмне забезпечення здатне виконувати зафарбовування тривимірної моделі методом Гуро у межах 0.1–0.5 секунди. Це

свідчить про істотне зростання продуктивності після оптимізації алгоритму, що забезпечує майже миттєвий візуальний зворотний зв’язок для користувача.

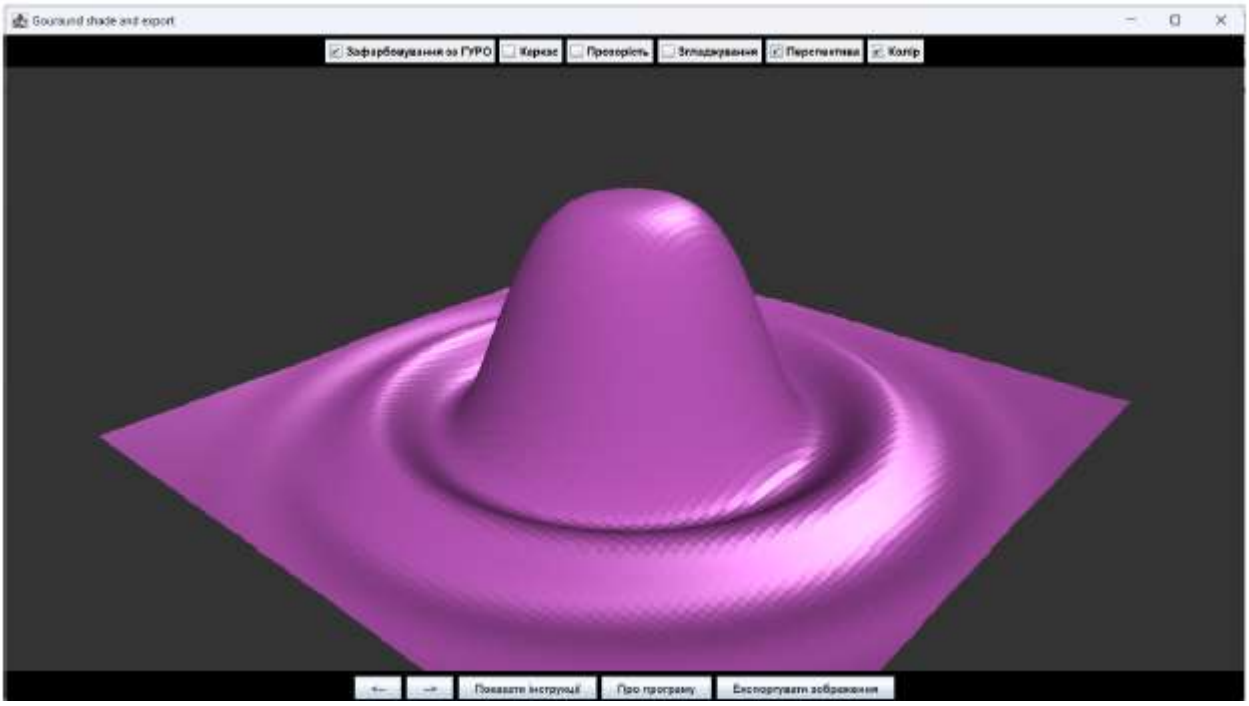


Рисунок 3.21 – Перевірка відображення базової тривимірної моделі

ВИСНОВКИ

У кваліфікаційній роботі бакалавра проведено ґрунтовний аналіз методу Гуро та сучасних підходів до рендерингу тривимірних зображень. Результати дослідження вказують на актуальність розробки нових методів та програмних засобів, здатних забезпечити високопродуктивне формування зображень із використанням методу Гуро.

У ході роботи здійснено огляд існуючих інструментів для рендерингу 3D-графіки, зокрема таких програмних пакетів, як Blender, Autodesk Maya, Cinema 4D, Unity та 3D Max. Для кожного з них було проаналізовано ключові функціональні можливості, переваги та обмеження. На основі цього аналізу сформульовано вимоги до розробки власного програмного забезпечення для реалізації методу Гуро, орієнтованого на підвищення продуктивності та ефективності візуалізації.

У межах дослідження розроблено алгоритм зафарбовування поверхонь тривимірних моделей методом Гуро, обрано та реалізовано методи оптимізації рендерингу, а також спроєктовано високопродуктивний метод візуалізації, який поєднує вдосконалений варіант методу Гуро з техніками розпаралелення обчислень.

Як результат, створено низку алгоритмів та інструментів, призначених для високопродуктивного формування зображень за методом Гуро. Обґрунтовано доцільність розробки власного програмного забезпечення на основі аналізу недоліків існуючих рішень та з метою підвищення ефективності використання обчислювальних ресурсів. Запропоновані методи забезпечують зменшення часу обробки графічної інформації, а також демонструють стабільну роботу за умов обробки вхідних даних різних форматів і в ситуаціях з потенційними помилками користувача.

СПИСОК ЛІТЕРАТУРИ

1. A. Watt та S. Maddock, 3D Computer Graphics. Pearson Educ., Limited, 2021.

2. J. Cardoso, 3D Photorealistic Rendering. Boca Raton : CRC Press, 2016.: K Peters/CRC Press, 2017.

3. Затемнення за ГУРО [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki (дата звернення: 18.03.2025)

4. Gouraud Shading in Computer Graphics [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/gouraud-shading-in-computer-graphics/ (дата звернення: 18.03.2025)

5. Gouraud Shading [Електронний ресурс] – Режим доступу до ресурсу: https://garykeen27.wixsite.com/portfolio/gouraud-shading (дата звернення: 18.03.2025)

6. Blender [Електронний ресурс] – Режим доступу до ресурсу: https://www.blender.org/ (дата звернення: 28.03.2025)

7. Autodesk Maya [Електронний ресурс] – Режим доступу до ресурсу: https://www.autodesk.com/products/maya/overview?term=1- YEAR&tab= subscription (дата звернення: 28.03.2025)

8. Cinema 4D [Електронний ресурс] – Режим доступу до ресурсу: https://www.maxon.net/en (дата звернення: 28.03.2025)

9. Unity [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/ru/releases/editor/whats-new/2022.2.10 (дата звернення: 28.03.2025)

10. 3D MAX [Електронний ресурс] – Режим доступу до ресурсу: https://apps.autodesk.com/3DSMAX/ (дата звернення: 28.03.2025)

11. Gouraud shading [Електронний ресурс] – Режим доступу до ресурсу: https://graphics.fandom.com/wiki/Gouraud_shading (дата звернення: 28.03.2025)

12. How to Gouraud shading with diffuse lighting in WebGl? [Електронний ресурс] – Режим доступу до ресурсу:

https://stackoverflow.com/questions/55634273/how-to-gouraud-shading (дата звернення: 28.03.2025)

13. Hast A. Models that use logarithmic and power functons [Text] / A. Hast, O. N. Romanyuk, O. M. Lyashenko // The Sixth International Conference “INTERNET –EDUCATION - SCIENCE”, Vinnytsia, Ukraine, October 7 –11, 2008 . – 2008. – C. 538-542.

14. Visio [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-365/visio/flowchart-software (дата звернення: 08.04.2025)

15. Що таке тестування програмного забезпечення? [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/ (дата звернення: 20.04.2025)

16. Функціональне тестування [Електронний ресурс] – Режим доступу до ресурсу: https://mate.academy/blog/qa/functional-non-functional-testing (дата звернення: 20.04.2024)

17. Тестування на відмову та поновлення [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/testuvannya-na-vidmovu-ta-ponovlennya (дата звернення: 20.04.2025)

18. Watt A., Maddock S. 3D Computer Graphics. – London : Pearson Education Limited, 2021. – 672 p.

19. Cardoso J. 3D Photorealistic Rendering. – Boca Raton : CRC Press, 2016. – 312 p.

20. Gouraud shading [Електронний ресурс]. – Режим доступу: https://graphics.fandom.com/wiki/Gouraud_shading – Назва з екрана. – Дата звернення: 28.03.2025.

21. How to Gouraud shading with diffuse lighting in WebGl? [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/questions/55634273/how-to-gouraud-shading – Назва з екрана. – Дата звернення: 28.03.2025.

22. Мельник С. В. Основи комп’ютерної графіки. – К. : ВЦ Академія, 2019. – 240 с.

23. Foley J. D., van Dam A., Feiner S. K., Hughes J. F. Computer Graphics: Principles and Practice. – 3rd ed. – Boston : Addison-Wesley, 2013. – 1264 p.

24. Angel E., Shreiner D. Interactive Computer Graphics: A Top-Down Approach with WebGL. – 7th ed. – Boston : Pearson, 2015. – 720 p.

25. Hughes J. F., van Dam A., Foley J. D., Feiner S. K. Computer Graphics: Principles and Practice. – Boston : Addison-Wesley, 2014. – 1170 p.

26. Mortenson M. E. Mathematics for Computer Graphics Applications. – 2nd ed. – New York : Industrial Press Inc., 2002. – 354 p.

27. Salomon D. Curves and Surfaces for Computer Graphics. – New York : Springer, 2006. – 460 p.

28. Shirley P., Marschner S. Fundamentals of Computer Graphics. – 4th ed. – Boca Raton : CRC Press, 2015. – 784 p.

29. Szirmay-Kalos L., Umenhoffer T. Displacement Mapping on the GPU – State of the Art // Computer Graphics Forum. – 2008. – Vol. 27(1). – C. 153–176.

30. Zink J. D., Pettineo M., Hoxley J. Practical Rendering and Computation with Direct3D 11. – Boca Raton : CRC Press, 2011. – 648 p.

31. Hart J. C. Graphics Gems IV. – San Diego : Academic Press, 1994. – 592 p.

32. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering. – 4th ed. – Boca Raton : CRC Press, 2018. – 1172 p.

33. Lengyel E. Mathematics for 3D Game Programming and Computer Graphics. – 3rd ed. – Boston : Cengage Learning, 2011. – 624 p.

34. Eberly D. H. 3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics. – 2nd ed. – Burlington : Morgan Kaufmann, 2006. – 1040 p.

ДОДАТКИ