

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**  
на тему:  
**«Розробка методів і програмних засобів  
рендерингу для систем комп'ютерної графіки»**

Виконав: студент II-го курсу,  
групи 1ПІ-21м

спеціальності 121 – Інженерія  
програмного забезпечення

(шифр і назва напряму підготовки, спеціальності)

Завальнюк Є. К.  
(прізвище та ініціали)

Керівник: д.т.н., професор каф. ПЗ

Романюк О. Н.  
« 14 » 12 2022 р.

Опонент: Савицька Л. А.  
к.т.н., доцент каф. ОТ Савицька Л. А.

« 14 » 12 2022 р.

Допущено до захисту

Завідувач кафедри ПЗ

д.т.н., проф. Романюк О. Н.

(прізвище та ініціали)

« 14 » 12 » 2022 р.

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Рівень вищої освіти II-й (магістерський)

Галузь знань 12 – Інформаційні технології

Спеціальність 121 – Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О. Н.

«16» вересня 2022 р.

### ЗАВДАННЯ НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Завальнюку Євгену Костянтиновичу

1. Тема роботи – розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки.

Керівник роботи: Романюк Олександр Никифорович, д.т.н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від «15» вересня 2022 р. № 205-А.

2. Строк подання студентом роботи

9 грудня 2022 р.

3. Вихідні дані до роботи: модель освітлення – на основі двопроменевої функції відбивної здатності поверхні; кольоровий режим – TrueColor; вихідні дані для зафарбовування – вектори нормалей до спостерігача  $\vec{V}$ , до джерела світла  $\vec{L}$ , серединний вектор  $\vec{N}$ ; діапазон зміни коефіцієнта спекулярності поверхні  $1 - 1000$ , мова програмування – C#; розмір дискретного координатного простору –  $4096*4096$ .

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання та постановка задач дослідження; розробка нових дистрибутивних функцій відбивної здатності поверхні; модифікація косинус-квадратичної та квадратичної моделей освітлення; розробка програмного засобу для порівняння моделей відбиття та тестування розроблених функцій відбивної здатності; економічна частина.

5. Перелік графічного матеріалу: галузі застосування та перспективи розвитку графічних засобів, розроблені моделі відбиття світла, тестування розроблених моделей відбиття.

#### 6. Консультанти розділів роботи

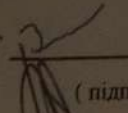
Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О. Н., д.т.н., завідувач кафедри ПЗ	16.09.22	15.12.22
5	Глущенко Л. Д., к.е.н., доц. кафедри ЕПВМ	21.11.22	08.12.22

7. Дата видачі завдання 16 вересня 2022 р.

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задач дослідження	17.09.2022- 25.09.2022	Вик.
2	Розробка нових дистрибутивних функцій відбивної здатності поверхні	26.09.2022- 16.10.2022	Вик.
3	Модифікація косинус-квадратичної та квадратичної моделей освітлення	17.10.2022- 06.11.2022	Вик.
4	Розробка програмного засобу для порівняння моделей відбиття та тестування розроблених функцій відбивної здатності	07.11.2022- 20.11.2022	Вик.
5	Економічна частина	21.11.2022- 08.12.2022	Вик.

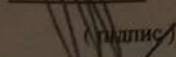
Студент

  
( підпис )

Завальнюк Є. К.

( прізвище та ініціали )

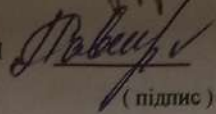
Керівник магістерської кваліфікаційної роботи

  
( підпис )

Романюк О. Н.

( прізвище та ініціали )

Опонент магістерської кваліфікаційної роботи

  
( підпис )

Савицька Л. А.

( прізвище та ініціали )

## АНОТАЦІЯ

УДК 004.92

Завальнюк Є. К. Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2022. 144 с.

На укр. мові. Бібліогр.: 40 назв; рис.: 72; табл.: 9.

У магістерській кваліфікаційній роботі проведено аналіз основних етапів графічного конвеєра, методів зафарбовування поверхонь, двопроменевих функцій відбивної здатності, графічних конвеєрів. Запропоновано розроблені моделі відбиття світла: модифікацію моделі Шліка з розрахованими степенями, коефіцієнтами, модель на основі двох утворюючих функцій. Запропоновано модифікацію косинус-квадратичної моделі відбиття. Запропоновані розраховані нормуючі коефіцієнти для модифікованої моделі Шліка, квадратичної моделі.

Розроблено програмний засіб рендерингу для порівняння моделей відбиття.

Розроблені методи та програмні засоби рендерингу призначаються для використання у системах комп'ютерної графіки.

В економічній частині розраховано рівень комерційного потенціалу наукової розробки, витрати на здійснення розробки, зміну чистого прибутку, період окупності інвестицій.

Ключові слова: рендеринг, графічний конвеєр, двопроменева функція відбивної здатності, фізично коректні моделі, модель зафарбовування.

## ABSTRACT

Zavalniuk Y. K. The development of computer graphics systems rendering methods and applications. Master's thesis in specialization 121 – software engineering, educational program – software engineering. Vinnytsia: VNTU, 2022. 144 p.

In Ukrainian language. Bibliographer: 40 titles; fig.: 72; tabl.: 9.

In this master's thesis graphics pipeline main steps, shading methods, bidirectional reflectance distribution functions, graphics engines were analyzed. Developed reflectance models were proposed: Schlick model modification with calculated powers, coefficients, the model based on two formative functions. The modification of cosine-quadratic model was proposed. The calculated normalizing coefficients for the Schlick modified model and quadratic model were proposed.

Software rendering application for reflectance models comparison was developed.

Developed rendering methods and applications are appointed for computer graphics systems usage.

The level of commercial potential, development spendings, net gain, investment payback period were calculated in ecomics section.

Keywords: rendering, graphics engine, bidirectional reflectance distribution function, physically correct models, shading model.

# ЗМІСТ

<b>ВСТУП</b> .....	9
<b>1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ</b> .....	12
1.1 Аналіз основних етапів графічного конвеєра.....	12
1.2 Аналіз методів зафарбовування поверхонь і двопробових функцій відбивної здатності.....	15
1.3 Аналіз професійних графічних конвеєрів .....	22
1.4 Постановка задач для розробки методів і програмного забезпечення рендерингу .....	28
1.5 Висновки .....	28
<b>2 РОЗРОБКА НОВИХ ДИСТРИБУТИВНИХ ФУНКЦІЙ ВІДБИВНОЇ ЗДАТНОСТІ ПОВЕРХНІ</b> .....	30
2.1 Розробка модифікованої моделі Шліка.....	30
2.2 Знаходження нормуючого коефіцієнта для моделі відбиття.....	40
2.3 Розробка ДФВЗ на основі двох утворюючих функцій.....	45
2.4 Висновки .....	49
<b>3 МОДИФІКАЦІЯ КОСИНУС-КВАДРАТИЧНОЇ ТА КВАДРАТИЧНОЇ МОДЕЛЕЙ ОСВІТЛЕННЯ</b> .....	50
3.1 Модифікація косинус-квадратичної моделі освітлення.....	50
3.2 Визначення значень степенів косинус-квадратичної функції.....	56
3.3 Знаходження нормуючого коефіцієнта для квадратичної моделі освітлення	60
3.4 Висновки .....	65
<b>4 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ПОРІВНЯННЯ МОДЕЛЕЙ ВІДБИТТЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНИХ ФУНКЦІЙ ВІДБИВНОЇ ЗДАТНОСТІ</b> .....	66
4.1 Обґрунтування засобів розробки.....	66
4.2 Розробка інтерфейсу програмного засобу .....	68
4.3 Розробка програмного засобу .....	69
4.4 Тестування розроблених ДФВЗ у програмних засобах .....	82
4.5 Висновки .....	88
<b>5 ЕКОНОМІЧНА ЧАСТИНА</b> .....	90
5.1 Оцінювання комерційного потенціалу розробки.....	90

5.2 Розрахунок витрат на здійснення науково-дослідної роботи.....	94
5.3 Прогнозування комерційних ефектів від реалізації результатів розробки .....	98
5.4 Розрахунок ефективності вкладених інвестицій та періоду їхньої окупності.....	100
5.5 Висновки .....	102
<b>ВИСНОВКИ .....</b>	<b>104</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b>	<b>105</b>
<b>ДОДАТКИ.....</b>	<b>110</b>
ДОДАТОК А Технічне завдання .....	111
ДОДАТОК Б Протокол перевірки роботи .....	115
ДОДАТОК В Лістинг програми обчислення оптимальних параметрів моделі Шліка .....	116
ДОДАТОК Г Лістинг програмного засобу.....	120
ДОДАТОК Д Ілюстративна частина .....	133



## ВСТУП

**Актуальність теми.** У системах комп'ютерної графіки значна увага приділяється формуванню тривимірних зображень, які найбільш реалістично подають особливості реальних об'єктів.

Етапи формування зображень утворюють графічний конвеєр [1]. Перша підсистема конвеєра включає формування математичних моделей. Друга підсистема конвеєра (рендерингу) включає формування елементів зображень об'єктів.

Однією з процедур рендерингу є зафарбовування об'єкта.

Зафарбовування об'єктів характеризується значною обчислювальною складністю [2], оскільки для визначення інтенсивності кольору використовуються координати джерела світла, спостерігача, точки поверхні, значення інтенсивності джерела, особливості відбивної здатності поверхні.

Обчислювальна складність процедури зафарбовування залежить від складності двопроменевої функції відбивної здатності (ДФВЗ), що подає оптичні властивості об'єктів [3]. Тому актуальною є розробка обчислювально простих ДФВЗ.

Серед бажаних особливостей ДФВЗ: точне відтворення зон епіцентру відблиску та блюмінгу, дотримання принципу симетричності (при зміні напрямків значення ДФВЗ зберігається) [3], дотримання закону збереження енергії (розсіяна над поверхнею кількість енергії не перевищує кількість, що надійшла) [3], додатні значення ДФВЗ на проміжку (для відсутності необхідності відсікання частини кривої), відсутність залежності складності обчислення значення ДФВЗ від зростання коефіцієнта спекулярності поверхні, відсутність великих степенів, використання значень косинусів кутів (що легко обчислюються), монотонна зміна значень функції [4].

Наявні двопроменеві функції відбивної здатності не завжди відповідають особливостям завдань рендерингу. Тому актуальною є розробка нових ДФВЗ.

**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 реалістичності відтворення спекулярної складової кольору за рахунок розробки та впровадження нових двопробових функцій відбивної здатності поверхні.

Основними задачами дослідження є:

- розробка нових двопробових функцій відбивної здатності;
- модифікація двопробових функцій відбивної здатності поверхні;
- дослідження моделей відбивних здатностей поверхонь;
- розробка програмних засобів для реалізації отриманих ДФВЗ;
- тестування отриманих функцій у розробленому засобі та аналогах.

**Об'єкт дослідження** – процес формування зображень тривимірних об'єктів у системах комп'ютерної графіки.

**Предмет дослідження** – методи та засоби рендерингу у системах комп'ютерної графіки.

**Методи дослідження.** У процесі досліджень використовувались: інтегральне та диференціальне числення, теорія чисел, теорії інтерполювання для розробки нових моделей відбиття світла від поверхні, комп'ютерне моделювання для аналізу та перевірки достовірності отриманих теоретичних положень.

**Наукова новизна отриманих результатів.**

1. Запропоновано модифікацію функції Шліка, яка відрізняється від відомої використанням другої степені та поправочних коефіцієнтів, що дає можливість підвищити точність визначення спекулярної складової кольору та реалістичного відтворення як епіцентру відблиску, так і його блюмінгу.

2. Вперше запропоновано для визначення двопробової функції відбивної здатності використання не однієї, а суми двох утворюючих функцій, що дає можливість підвищення точності визначення спекулярної складової

кольору і, як наслідок, більш реалістично відтворити відблиски.

3. Запропоновано модифікації двопробових функцій Шліка, квадратичної функції, які відрізняються від відомих введенням поправочних коефіцієнтів, що дало можливість фізично коректно відтворювати відблиски на поверхнях тривимірних об'єктів.

**Практична цінність отриманих результатів.** Практичне значення полягає у розробці на основі проведених теоретичних досліджень нових алгоритмів і програм для задач рендерингу.

**Особистий внесок здобувача.** Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У наукових працях, опублікованих у співавторстві, автору належать такі результати: [5] – аналіз нових моделей відбивної здатності, [6] – формула модифікованої моделі Шліка, [7] – аналіз рендерів для систем автоматизованого проектування, [8] – аналіз нових фізично точних моделей відбивної здатності, [9] – аналіз особливостей стандарту DirectX 12.

**Апробація матеріалів магістерської кваліфікаційної роботи.** Матеріали магістерської кваліфікаційної роботи доповідалися на конференціях: Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів «Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації» (Одеса, 2022), Міжнародна науково-практична конференція «Інформаційні технології і автоматизація» (Одеса, 2022), Міжнародна науково-практична дистанційна конференція Modern Research in World Science (Львів, 2022), Міжнародна науково-практична Інтернет-конференція «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2022).

**Публікації.** Результати досліджень опубліковано у 6 наукових працях, з них стаття у фаховому журналі, 5 тез доповідей на міжнародних конференціях.

# 1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

## 1.1 Аналіз основних етапів графічного конвеєра

Формування тривимірних зображень складається з послідовності визначених етапів, що утворюють графічний конвеєр (рисунок 1.1) [1].

Першим здійснюється опис сцени. Встановлюється, які є об'єкти, як вони розташовані один відносно одного.

Наступною є система геометричних перетворень.

Виконується тесеція об'єкта – представлення у формі сукупності багатокутників без перекриття та проміжків. Найбільш часто об'єкти діляться на трикутники [1] (процес триангуляції) через простоту даного підходу.

Далі здійснюються модельні перетворення об'єктів. До об'єктів застосовуються афінні перетворення для маніпуляції ними. Серед перетворень виділяють переміщення, масштабування, поворот об'єкта. Потім моделюється освітлення об'єктів сцени – встановлюються тип джерела, оптичні властивості поверхні об'єкта. Здійснюються видові перетворення [1], що полягають у встановленні координат вершин об'єкта відносно спостерігача (або камери).

На завершени етапу об'єкт представляється каркасною сіткою.

Етап рендерингу полягає у визначенні адрес і кольорів пікселів. Характеризується найбільшими обчислювальними витратами, оскільки здійснюються складні обчислення навколо кожного пікселя [10]. Є останнім етапом перед виведенням зображення на екран.

Растрезація включає перетворення геометричного представлення у піксельне. Для підвищення реалістичності видаляються невидимі поверхні об'єктів. Серед алгоритмів видалення невидимих поверхонь об'єктів найпоширенішими є методи Варнока, Галімберті і Монтанарі, Уоткінса, трасування променів, Z-буфера.

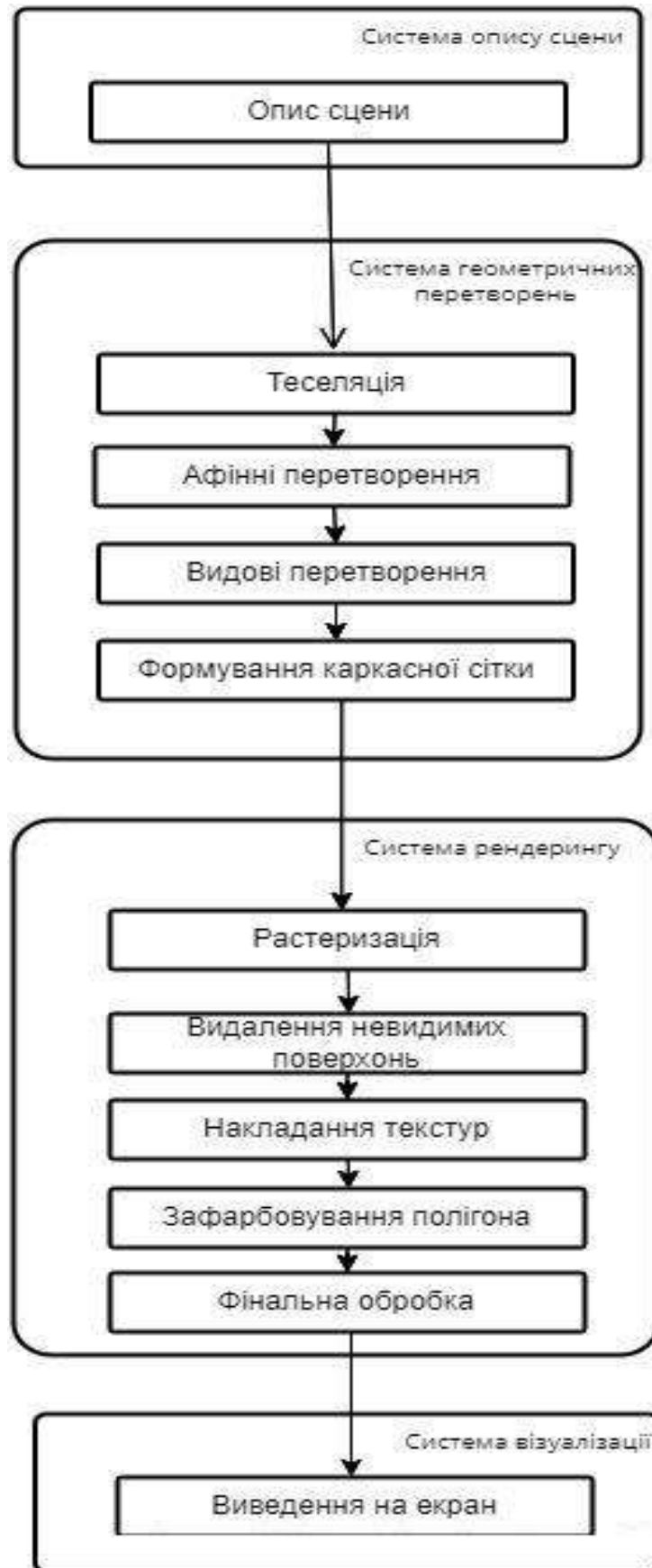


Рисунок 1.1 – Основні етапи графічного конвеєра

Алгоритм Z-буфера характеризується простотою та високою швидкістю. У спеціальному буфері зберігається відстань кожної точки зображення до окремої площини. При порівнянні відстаней визначається, чи точка перекрита іншою й чи потрібно змінювати колір пікселя [1].

Наступним є накладання текстури на об'єкт – зображення, що представляє особливості поверхні.

Для визначених геометричних примітивів здійснюється поточкове зафарбовування об'єкта. Для зафарбовування об'єктів застосовуються такі методи: плоский, Гуро, Фонга. В основі даних методів використовується інформація у вершинах полігонів [11]. Метод Гуро включає визначення інтенсивностей у точках полігону відносно інтенсивностей вершин. Методи зафарбовування розглянуто у підрозділі 1.2.

При визначенні інтенсивності кольору для точки враховуються компоненти: фоновий (відбите світло від зовнішнього середовища), дифузний (світло, відбите під багатьма кутами), спекулярний (відбите світло у напрямку до спостерігача). Для даних складових підбираються відповідні коефіцієнти. Відбиття світла визначається через значення двопрменевої функції відбивної здатності (ДФВЗ). ДФВЗ показують відбиту частку світла у напрямку  $\vec{V}$ . Основними двома видами ДФВЗ є емпіричні (прості, менш точні) та фізично точні (більш точні, фасетне представлення поверхні). Найбільш часто використовуються функції Фонга, Блінна через простоту їхнього розрахунку. ДФВЗ розглянуто у підрозділі 1.2.

Останнім підетапом рендерингу є фінальна обробка. Здійснюються згладжування та антиаліайзинг [1].

Оскільки етап рендерингу використовує найбільше обчислювальних ресурсів, то актуальною є розробка нових методів і програмних засобів рендерингу, що забезпечать підвищену продуктивність і реалістичність візуалізації.

## 1.2 Аналіз методів зафарбовування поверхонь і двопробених функцій відбивної здатності

До основних методів зафарбовування поверхні відносяться [11]: плоске зафарбовування, зафарбовування методом Гуро, зафарбовування методом Фонга.

В основі плоского зафарбовування (рисунок 1.2) має місце незмінність кольору в межах полігону. Значення інтенсивності кольору кожної точки полігону є середнім значенням інтенсивностей вершин [11]. Зазвичай використовується, коли необхідно швидко зафарбовування поверхні. Основний недолік – видно різкі границі між полігонами.

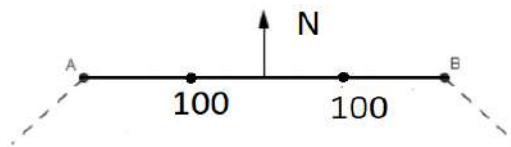


Рисунок 1.2 – Зафарбовування середніми значеннями інтенсивностей вершин при плоскому методі

Зафарбовування методом Гуро – метод, розроблений Анрі Гуро у 1971 році. Є найбільш поширеним методом із розглянутих. Спершу обчислюються інтенсивності кольору у вершинах полігону, далі значення інтерполюються між вершинами (рисунок 1.3) [11].

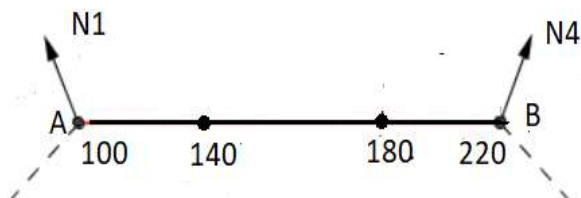


Рисунок 1.3 – Інтерполяція інтенсивностей кольору між вершинами при зафарбовуванні методом Гуро

Для методу характерними є смуги Маха (збільшення контрасту між відтінками) на ребрах полігонів (рисунок 1.4) [1].



Рисунок 1.4 – Смуги Маха

Модель зафарбовування Фонга – метод, розроблений у 1973 році Буї Туонг Фонгом. В основі методу використовується інтерполяція нормалей відносно нормалей вершин полігону (рисунок 1.5) [11]. Кожна з обчислених нормалей застосовується для обчислення інтенсивності кольору у відповідній точці поверхні.

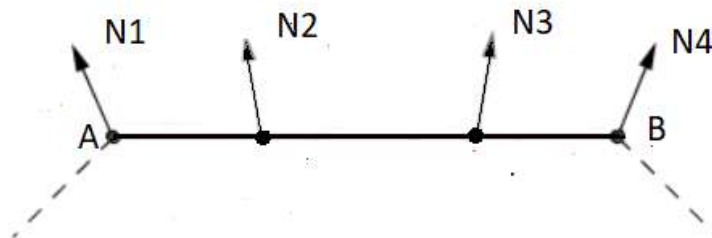


Рисунок 1.5 – Інтерполяція нормалей при зафарбовуванні методом Фонга

Метод Фонга найточніший з розглянутих, однак найбільш обчислювально складний.

Для обчислення інтенсивності кольору в окремій точці використовується функція зафарбовування, що розраховується згідно з формулою [3]



$$I = I_a k_a + \frac{I}{k+d} (k_d \cos \theta + k_s \cos^n \gamma), \quad (1.1)$$

де  $I_a$  – інтенсивність розсіяного світла,  $I$  – інтенсивність джерела,  $k_a$  – коефіцієнт відбиття розсіяного світла,  $k_d$  – коефіцієнт дифузного відбиття,  $k_s$  – коефіцієнт спекулярного відбиття,  $\theta$  – кут між нормаллю та падаючим променем,  $d$  – відстань точки від джерела,  $k$  – стала,  $\cos^n \gamma$  - представляє ДФВЗ (у даному випадку Фонга-Блінна).

Перший доданок функції відповідає складовій розсіяного світла, другий дифузній складовій, третій спекулярній складовій.

Складова розсіяного світла включає освітлення від зовнішнього середовища. Наприклад, світло відбивається від віддаленого об'єкта та впливає на предмет, що розглядається. Для обчислення складової інтенсивності розсіяного світла множиться на коефіцієнт відбиття розсіяного світла [12].

Дифузна складова (рисунок 1.6) включає світло, що відбивається від точки у різних напрямках. Для обчислення складової перемножуються інтенсивність джерела, коефіцієнт дифузного відбиття, косинус кута між вектором від точки до джерела світла та нормаллю [3].

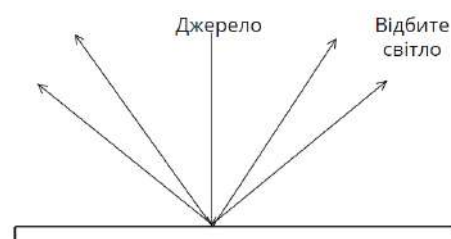


Рисунок 1.6 – Дифузне відбиття світла

Спекулярна складова (рисунок 1.7) включає світло, відбите в напрямку під визначеним кутом. Складова пов'язана із відбивною здатністю поверхні. Для обчислення спекулярної складової перемножуються інтенсивність джерела

світла, коефіцієнт спекулярного відбиття та значення двопроменевої функції відбивної здатності (ДФВЗ).

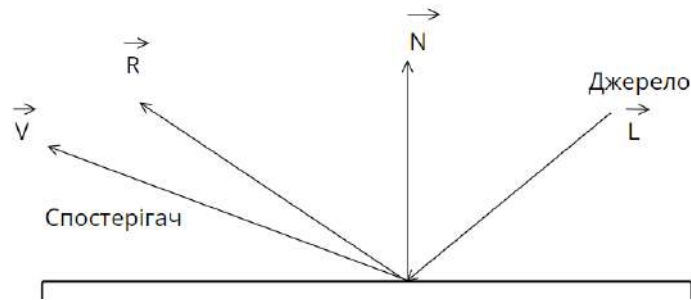


Рисунок 1.7 – Спекулярне відбиття світла

Двопроменева функція відбивної здатності, що визначає частку відбитого від точки випромінювання, обчислюється за формулою

$$\frac{d\Phi}{ds \perp d\omega}, \quad (1.2)$$

де  $d\Phi$  – променевий потік,  $ds$  – площа поверхні, на яку падає потік,  $d\omega$  – тілесний кут, з якого падає потік.

Серед отриманих експериментальним шляхом ДФВЗ основними є моделі Фонга, Блінна, Шліка, Гаусса. Моделі даного типу є більш простими, підтримують формування достатньо реалістичних зображень [3].

Модель Фонга поєднує вектор до спостерігача  $\vec{V}$  і вектор віддзеркаленого променя  $\vec{R}$ , що обчислюється за формулою [13]

$$2(\vec{L}\vec{N})\vec{N} - \vec{L}, \quad (1.3)$$

де  $\vec{L}$  – вектор падіння,  $\vec{N}$  – вектор-нормаль.

ДФВЗ Фонга розраховується за формулою

$$\cos(\alpha)^n, \quad (1.4)$$

де  $\alpha$  – кут між  $\vec{V}$  і  $\vec{R}$ ,  $n$  – коефіцієнт спекулярності для поверхні (залежить від особливостей виду поверхні, 1..1000).

ДФВЗ Блінна (Фонга-Блінна) є схожою моделлю, однак використовується інший кут – між нормаллю та сумою векторів  $\vec{V}$ ,  $\vec{L}$ . Заміна кута надає перевагу у швидкості [14], якщо спостерігач і джерело значно віддалені. Обчислюється згідно з формулою

$$\cos(\gamma)^n, \quad (1.5)$$

де  $\gamma$  – кут між  $\vec{N}$  і  $\vec{H}$  (сума векторів до джерела  $\vec{L}$  й спостерігача  $\vec{V}$ ).

Недоліком ДФВЗ Фонга та Блінна є суттєве зростання обчислень зі збільшенням  $n$ .

Для зменшення кількості обчислень моделі Блінна й Фонга апроксимовані ДФВЗ Шліка, де відсутнє піднесення у степінь. ДФВЗ Шліка розраховується за формулою

$$\frac{\cos(\gamma)}{n - n * \cos(\gamma) + \cos(\gamma)} \quad (1.6)$$

Апроксимація Шліка характеризується нереалістичним формуванням відблисків у зоні затухання, оскільки функція повільно спадає [3]. Тому доцільним є вдосконалення апроксимації.

ДФВЗ Фонга, Блінна та Шліка є емпіричними функціями на основі скалярного добутку векторів. Тобто, косинус кута можна представити через

скалярний добуток векторів променів, що полегшує обчислення. Існують також емпіричні функції на основі визначення кута.

Серед емпіричних функцій на основі визначення кута ДФВЗ Гаусса, що розраховується згідно з формулою

$$e^{-\frac{n^*(\angle(\vec{H}, \vec{L}))^2}{2}} \quad (1.7)$$

Характеризується порівняно високою реалістичністю, однак обчислення кута займає багато обчислювальних ресурсів, як і в усіх ДФВЗ підгрупи [3].

Складніші фізично реалістичні ДФВЗ використовуються, коли необхідне точне відтворення поверхні об'єкта. Для більш детального представлення поверхня розбивається на дрібні частини – фасети.

Серед ізотропних фізично реалістичних ДФВЗ виділяються моделі: дзеркальна Уорда, Торренса-Сперроу та Кука-Торренса. Ізотропні моделі не залежать від повороту об'єкта.

У дзеркальній ДФВЗ Уорда за рахунок розділення поверхні на фасети забезпечується висока точність візуалізації. Знаходиться за формулою [15]

$$\frac{1}{\sqrt{(\vec{N}\vec{L})(\vec{N}\vec{V})}} \frac{1}{4\pi m^2} e^{\frac{(NH)^2-1}{m^2(NH)^2}}, \quad (1.8)$$

де  $m$  – середнє відхилення фасетної орієнтації.

Для моделі Торренса-Сперроу використовується мікрофасетний розподіл Гаусса згідно формули [16]

$$c e^{\left(\frac{-\gamma}{m}\right)^2}, \quad (1.9)$$

де  $c$  – довільна константа.

Модель обчислюється за формулою

$$\frac{F}{\pi} \frac{DG}{(\vec{NL})(\vec{NV})}, \quad (1.10)$$

де  $F$  – коефіцієнт Френеля (представляє відбиття на межі середовищ),  $D$  – розподіл мікрофасетів Гаусса,  $G$  – коефіцієнт ослаблення світла.

Моделі Кука-Торренса також відповідає формула (1.10), однак використовується розподіл Бекмана.

Це забезпечує більшу точність, однак зростає обчислювальна складність функції [16].

Розрахунок розподілу здійснюється за формулою

$$\frac{1}{m^2 (NH)^4} e^{\frac{(NH)^2 - 1}{m^2 (NH)^2}} \quad (1.11)$$

Серед анізотропних моделей (де властивості об'єкта змінюються під час повороту) можна виділити моделі Уорда, Ашикмина-Ширлі.

Модель Уорда є видозміненою дзеркальною ДФВЗ Уорда для випадку поворотів об'єкта. Обчислюється за формулою [15]

$$\frac{1}{\sqrt{(\vec{NL})(\vec{NV})}} \frac{\exp(-\tan^2 \delta (\cos^2 \phi / \alpha_x^2 + \sin^2 \phi / \alpha_y^2))}{4\pi a_x a_y}, \quad (1.12)$$

де  $\phi$  - азимутний кут проекції напіввектора,  $\delta$  - кут між напіввектором і нормаллю,  $\alpha_x$  - стандартне відхилення фасети по  $x$ ,  $\alpha_y$  - стандартне відхилення фасети по  $y$ .

Модель Ашикмина-Ширлі заснована на моделях Уорда, Шліка, Неймана [17].

Обчислюється за формулою

$$\frac{\sqrt{(n_u + 1)(n_v + 1)}}{8\pi} \frac{(\vec{N}\vec{H})^{n_u \cos^2 \phi + n_v \sin^2 \phi}}{(\vec{N}\vec{L}) \max((\vec{N}\vec{V}), (\vec{N}\vec{L}))} F(\vec{N}\vec{L}), \quad (1.13)$$

де  $n_u, n_v$  - експоненти для зміни форми фасетів.

Отже, ДФВЗ вибирається залежно від необхідного рівня точності та допустимого рівня обчислювальних затрат. Для високопродуктивної візуалізації застосовуються прості емпіричні моделі, для точного відтворення фізично реалістичні.

Для засобів динамічної графіки необхідна розробка двопроменевої функції відбивної здатності, яка має відносно високу реалістичність і низьку обчислювальну складність.

### 1.3 Аналіз професійних графічних конвеєрів

Для реалізації алгоритмів рендерингу й виведення зображення на екран використовуються програми-рендери (графічні конвеєри).

Основними двома видами рендерів є: з допущеннями та без допущень. Рендери з допущеннями забезпечують наближену реалістичність, мають місце відхилення від фізичних аспектів. Рендери без допущень [18] відповідають фізичним аспектам, надають більш реалістичні результати, однак потребують більше обчислень.

Також рендери поділяються на основі процесора, що використовується: CPU, GPU, гібридні. CPU-рендери працюють на основі центрального процесора. Є більш надійними та забезпечують більш якісні зображення [19]. GPU-рендери працюють на основі графічного процесора. Значною перевагою є швидкість (50-100 разів), однак характерною є нестабільність роботи [20].

Гібридні рендери поєднують можливості використання центрального й графічного процесорів.

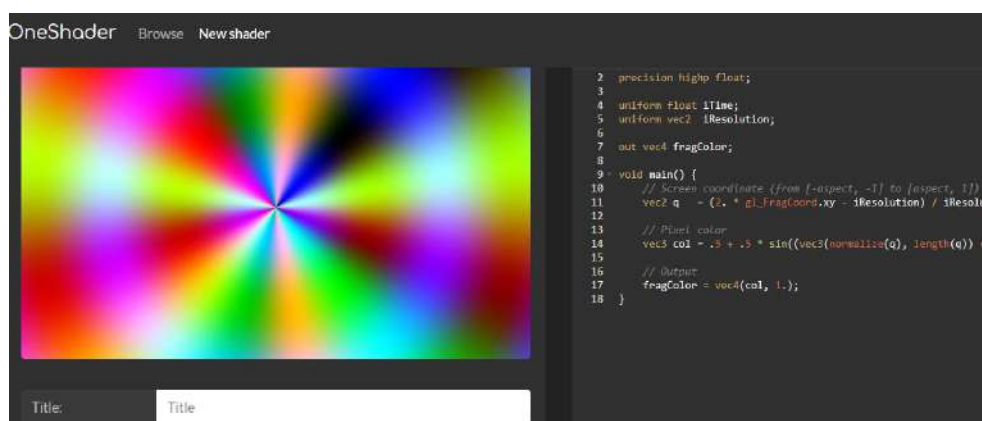
Для пришвидшення процесу візуалізації (зазвичай при роботі з фільмами, анімацією, архітектурними проектами) можливе використання груп вузлів рендерингу – рендер-ферм. Вузли рендер-ферм містять рендери, засоби 3D-моделювання. Тоді можлива візуалізація різних кадрів водночас. Рендер-ферми [21] можуть бути самоствореними або базуватись на хмарних технологіях.

Розглянемо основні графічні конвеєри.

OneShader [22] – безкоштовний веб-засіб створення фрагментних шейдерів (засобів визначення кольорів пікселів всередині полігону), розроблений Рейндером Ніджхоффом. Мовою написання шейдерів є GLSL (OpenGL Shading Language), підтримуються версії мови 1.0 і 3.0. Використовується для швидкої та «легкої» візуалізації об'єктів.

Основні пункти меню: New shader («новий шейдер», розробка коду для шейдера (рисунок 1.8)), Browse (сховище розроблених користувачами лістингів для шейдерів).

При створенні шейдера можливо додати код у відкрите сховище, зберегти як чорновик, зберегти у приватному режимі.



```

1 precision Highp float;
2
3
4 uniform float iTime;
5 uniform vec2 iResolution;
6
7 out vec4 fragColor;
8
9 void main() {
10     // Screen coordinate (from [-aspect, -1] to [aspect, 1])
11     vec2 q = (2. * gl_FragCoord.xy - iResolution) / iResolu
12
13     // Pixel color
14     vec3 col = .5 + .5 * sin((vec3(normalize(q), length(q)) *
15
16     // Output
17     fragColor = vec4(col, 1.);
18 }

```

Рисунок 1.8 – Приклад шейдера в OneShader

Перевагами OneShader є: простота, вузькоспеціалізований функціонал, безкоштовність, можливість швидкого опанування початківцями, робота без встановлення програмного забезпечення на комп'ютер.

До недоліків OneShader відносяться: відсутність багатовіконного представлення для порівняння, відсутність повноцінної документації.

RenderMan (рисунок 1.9) [23] – пропрієтарний рендер з допущеннями від Pixar. Основне застосування – розробка спецефектів для фільмів і комп'ютерних ігор. Характеризується широким набором функцій і характеристик: фотореалістичний рендеринг, отримання статистики, гібридний рендеринг на основі центрального й графічного процесорів, симулятор денного світла, використання мови шейдерів OSL (відкрита мова затінення), багат шарових матеріалів. Інтегрується з програмами 3D-моделювання Maya, Katana, Houdini, Blender. Pixar RenderMan безкоштовний за умови некомерційного використання [24].

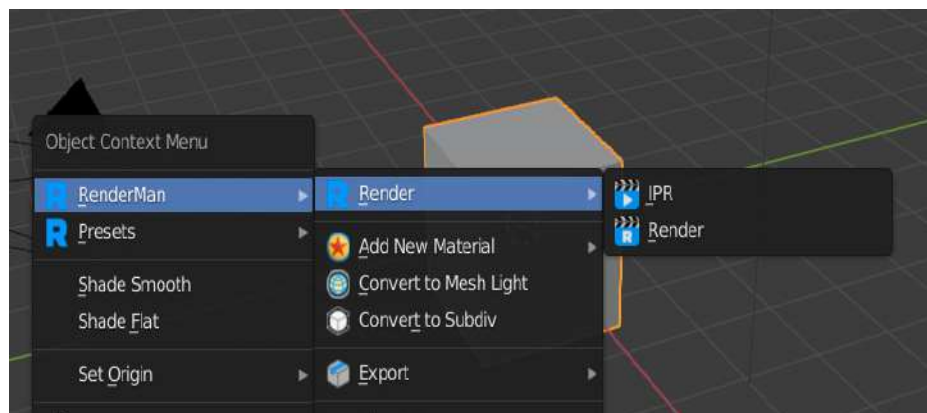


Рисунок 1.9 – Використання RenderMan у Blender

Перевагами Pixar RenderMan є: підтримка фотореалістичності, гнучкість, широкий функціонал, інтеграція з основними засобами 3D-моделювання, умовна безкоштовність, підтримка гібридного рендерингу.

Недоліками Pixar RenderMan є [25]: важкість опанування початківцями, використання вимагає наявності базових знань.



Cycles (рисунок 1.10) – безкоштовний рендер без допущень, вбудований у Blender.

Графічний конвеєр підтримує [26]: рендеринг на основі центрального й графічного процесорів, використання мови OSL, фізично коректний рендеринг, використання основних ДФВЗ, шейдери на основі вузлів, модель неба, шари рендерингу для декомпозиції, текстури картинок, відображення середовища на текстурі.

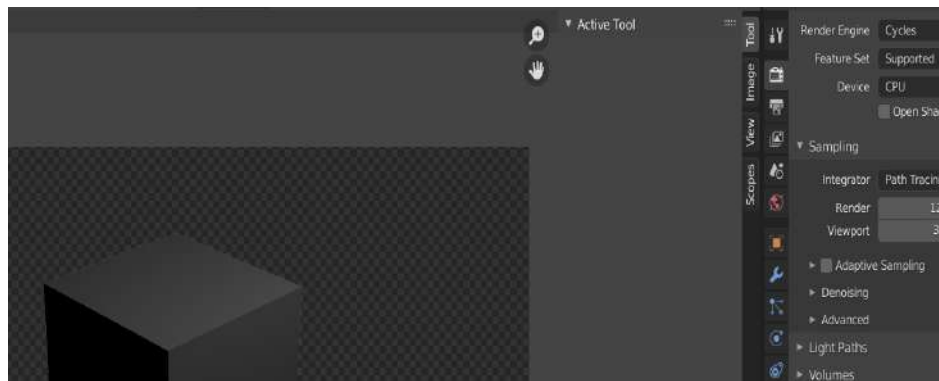


Рисунок 1.10 – Використання Cycles в Blender

Серед переваг конвеєра Blender Cycles виділяють [27]: можливість отримання фотореалістичних результатів, широкий функціонал, безкоштовність, потужний редактор вузлів, велика кількість ресурсів для вивчення, вбудованість у Blender.

Основний недолік – повільне функціонування (через використання трасування променів і кращу якість зображення).

Redshift Render (рисунок 1.11) – пропрієтарний GPU графічний конвеєр з допущеннями від німецької компанії Maxon. Є одним з найшвидших рендерів. Реалістичність сформованих зображень є порівняною з рендерами без допущень. Можлива інтеграція із засобами [28]: Cinema4D, 3DsMax, Blender, Maya, Katana, Houdini.

Використовується для розробки візуальних ефектів і дизайну руху.

Рендеринг здійснюється позаядерно (якщо пам'ять графічного процесора завершується, використовується пам'ять системи). Серед функціоналу підтримуються [29]: об'ємний рендеринг, багатокрокові перетворення розмиття, рендеринг волосся, підтримка моделі глобального освітлення, теселяція (розділення на фігури), згладження країв, рендеринг у режимі командного рядка.

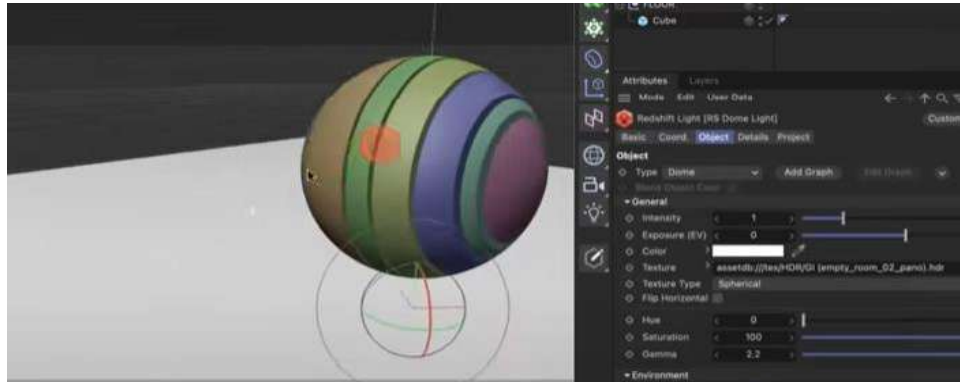


Рисунок 1.11 – Використання Redshift разом з Cinema4D

Перевагами Redshift є: висока швидкість, забезпечення фотореалізму, висока інтерактивність, підтримка рендер-ферм. Серед недоліків Redshift [30] виділяють: здобуття значного виграшу у продуктивності вимагає витрат часу для здійснення налаштувань, використання вузлового підходу для рендерингу може відлякати початківців.

Ostane (рисунок 1.12) – пропрієтарний GPU рендер від компанії OTOY Inc. Є одним із найшвидших рендерів без допущень (у 10-50 разів швидший, ніж рендери без допущень на основі центрального процесора). Через фізичну реалістичність і швидкість часто використовується для створення візуальних ефектів [28]. Наявні окрема програма та інтеграція із засобами [31]: Cinema4D, 3dsMax, Maya. Підтримуються об'ємний рендеринг (хмар, диму), мова OSL для створення шейдерів, вибікання текстури (збереження інформації про 3D-сітку у

текстурний файл), позаядерна геометрія, вузловий редактор для створення комплексних матеріалів, робота з HDR (розширеним динамічним діапазоном).

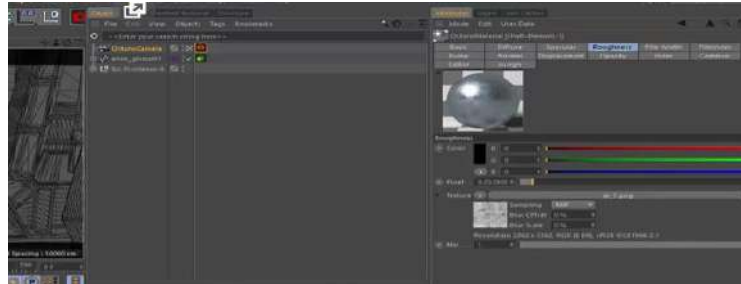


Рисунок 1.12 – Використання Octane разом з Сінема4D

Перевагами Octane є: фізична реалістичність, висока швидкість для рендера без допущень, простота налаштувань, широкий функціонал. Недоліками Octane є [27]: нестабільність роботи, обмежений вибір матеріалів, непослідовні оновлення.

Shadertoy (рисунок 1.13) [32] – безкоштовний веб-засіб створення шейдерів, розроблений Ініго Квілезом і Полом Джереміасом. Можливе створення шейдерів зображень, звуку, віртуальної реальності [33]. Мовою написання шейдерів є GLSL. Перевагами засобу є: простота, підтримка різних типів шейдерів, наявність документації, навчальних матеріалів. До недоліків засобу відносяться: одновіконне представлення, відсутність повного набору функціоналу графічних конвеєрів.

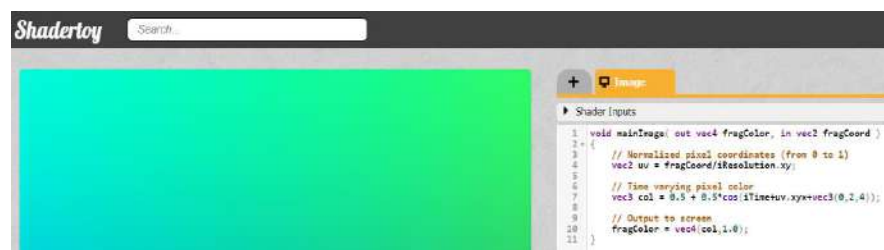


Рисунок 1.13 – Реалізація нового шейдера у Shadertoy

Аналіз показав:

1. Вибір типу графічного конвеєра залежить від особливостей завдання. Якщо найважливіша швидкість, то доцільними є GPU рендери з допущеннями. Якщо немає часових обмежень для виконання завдання, то важливі висока реалістичність й надійність, доцільні CPU рендери без допущень.

2. Для візуалізації складних сцен необхідно використовувати потужні рендери, як Pixar RenderMan, з 3D-засобами. Якщо необхідно швидко сформувати зображення простого об'єкта, доцільні прості засоби без зайвого функціоналу, як OneShader і Shadertoy.

3. Більшість рендерів є продукто-орієнтованими та застосовуються для ігор, фільмів. Тому наявна потреба у розробці засобів, орієнтованих на порівняння методів і моделей рендерингу, зокрема ДФВЗ.

#### **1.4 Постановка задач для розробки методів і програмного забезпечення рендерингу**

Провівши аналіз графічного конвеєра, методів зафарбовування, моделей відбиття, програмних засобів рендерингу, було визначено набір задач для магістерської кваліфікаційної роботи:

- розробка нових двопроменевих функцій відбивної здатності;
- модифікація двопроменевих функцій відбивної здатності поверхні;
- дослідження моделей відбивних здатностей поверхонь;
- розробка програмних засобів для реалізації отриманих ДФВЗ;
- тестування отриманих функцій у розробленому засобі та аналогах.

#### **1.5 Висновки**

Виконано аналіз етапів графічного конвеєра, методів зафарбовування, моделей відбивної здатності, професійних графічних конвеєрів.

Проаналізовано стадії графічного конвеєра: опис сцени, геометричні перетворення, рендеринг, візуалізацію. Обґрунтовано розробку нових методів і програмних засобів рендерингу.

Проаналізовано переваги й недоліки методів зафаровування (Гуро, Фонга, плоского), двопроменевих функцій відбивної здатності (Фонга, Блінна, Шліка, Гаусса, Торренса-Сперроу, Кука-Торренса, дзеркальна Уорда, Уорда, Ашикмина-Ширлі).

Обґрунтовано необхідність вдосконалення та розробки нових ДФВЗ. Розглянуто особливості основних рендерів (OneShader, RenderMan, Octane, Cycles, Redshift, Shadertoy), їх типи. Обґрунтовано розробку засобу для порівняння моделей ДФВЗ.

Здійснено постановку задач до магістерської кваліфікаційної роботи.

## 2 РОЗРОБКА НОВИХ ДИСТРИБУТИВНИХ ФУНКЦІЙ ВІДБИВНОЇ ЗДАТНОСТІ ПОВЕРХНІ

### 2.1 Розробка модифікованої моделі Шліка

Розглянемо модифікацію ДФВЗ Шліка. Знайдемо нову дистрибутивну функцію відбивної здатності поверхні у вигляді формули

$$\frac{a_1 \cos(x)^{b_1}}{a_2(a_3 n^{b_3} - a_4(n * \cos(x))^{b_4} + a_5 \cos(x)^{b_5})^{b_2}} \quad (2.1)$$

Знайдемо невідомі змінні  $b_n$  (степені) та  $a_n$  (множники).

Оскільки при  $x=0$   $\cos(x)$  дорівнює 1, то у ДФВЗ Шліка  $n - n \cos(x) = 0$ . Звідси  $b_4 = b_3, a_4 = a_3$ . З урахуванням останнього отримуємо такий вигляд запропонованої ДФВЗ

$$\frac{a_1 \cos(x)^{b_1}}{a_2(a_3 n^{b_3} - a_3(n * \cos(x))^{b_3} + a_5 \cos(x)^{b_5})^{b_2}} \quad (2.2)$$

Позначимо запропоновану ДФВЗ як  $F_{MSH}$ .

Авторами розроблено програму для підбору коефіцієнтів  $a_n$  й степенів  $b_n$ , граф-схему якої наведено на рисунку 2.1. Лістинг коду наведено у додатку В.

Для спрощення визначення невідомих в  $F_{MSH}$  розділимо процедури підбору коефіцієнтів і степенів.

Підберемо степені  $b_1 - b_5$ . На даному етапі значення коефіцієнтів  $a_1 - a_5$  приймемо рівними одиниці.

Враховуючи швидке зростання обчислювальної складності перебору та необхідність низької обчислювальної складності функції, значення масиву можливих степенів `rows` виберемо з інтервалу  $[-3,3]$ .

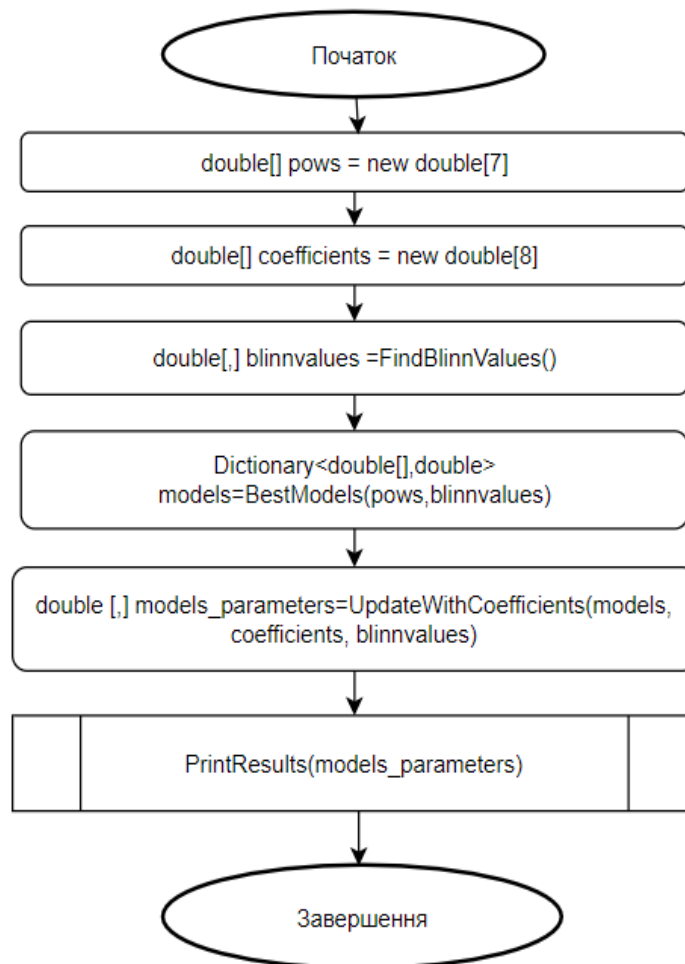


Рисунок 2.1 – Блок-схема комп’ютерного перебору параметрів  $F_{MSH}$

Масив можливих коефіцієнтів `coefficients` містить значення: 0.25, 0.5, 1, 1.25, 1.5, 2, 2.5, 3.

Далі у подвійному циклі для кутів  $[0,90]$  і коефіцієнтів спекулярності поверхні  $n$   $[1,1000]$  (всередині методу `FindBlinnValues`) викликається функція обчислення значень ДФВЗ Блінна. Отримані значення формують двовимірний масив `blinnvalues`.

У чотиривимірному циклі всередині функції `BestModels` (відповідно до 4 невідомих степенів) для  $n$  і кутів заповнюється двовимірний масив значень  $F_{MSH}$  з випадковими степенями. Обчислюється середнє з абсолютних відхилень масиву від `blinnvalues`. В циклах формується список п’яти наборів степенів з

найменшим середнім абсолютним відхиленням, наприкінці список записується у словник `models` (ключ словника – набір степенів, значення – відхилення).

У чотиривимірному (відповідно до 4 невідомих коефіцієнтів) циклі функції `UpdateWithCoefficients` випадкові набори коефіцієнтів поєднуються зі знайденими п'ятьма наборами степенів. Для кожного набору степенів визначається набір коефіцієнтів, що забезпечує найменше середнє абсолютне відхилення від `blinnvalues`. Функція повертає двовимірний масив, де рядки представляють інформацію сукупності параметрів  $F_{MSH}$ : середнє абсолютне відхилення від ДФВЗ Блінна, п'ять значень степенів, п'ять значень коефіцієнтів. Здійснюється присвоєння результату функції масиву `models_parameters`.

Функція `PrintResults` здійснює виведення інформації про отримані моделі (набори шуканих параметрів). Проаналізуємо отримані результати (таблиця 2.1). Найвищу точність має модель 1. Однак модель вимагає 5 піднесень до вищих степенів. Модель 0 є другою за точністю, необхідно 3 піднесення до вищих степенів. Модель 2 є співставною за точністю та потребує лише одне піднесення. Тому, враховуючи точність й простоту моделей, обрано модель 2.

Таблиця 2.1. Отримані моделі параметрів для  $F_{MSH}$

№	Степені					Коефіцієнти					Середнє абсолютне відхилення від ДФВЗ Блінна
	2	2	1	1	2	2	1.25	1	1	1.25	
0	2	2	1	1	2	2	1.25	1	1	1.25	0.0063
1	3	2	1	1	3	2	1.25	1	1	1.25	0.0062
2	1	2	1	1	1	2	1.25	1	1	1.25	0.0066
3	2	2	1	1	3	2	1.25	1	1	1.25	0.0066
4	1	2	1	1	2	2	1.25	1	1	1.25	0.0070



Для розрахунку спекулярної складової найбільш оптимальним є вид  $F_{MSH}$ , що представляється формулою [34]

$$\frac{2\cos x}{\left(1 + \frac{1}{4}\right)\left(n - n^* \cos x + \left(1 + \frac{1}{4}\right)\cos x\right)^2} \cdot \quad (2.3)$$

Проаналізуємо запропоновану ДФВЗ. Для здійснення порівнянь позначимо ДФВЗ Шліка як  $F_{SH}$ , ДФВЗ Блінна як  $F_B$ .

На рисунку 2.2 зображено графіки  $F_{MSH}$ ,  $F_{SH}$ ,  $F_B$ . З наведених графіків видно, що запропонована  $F_{MSH}$  забезпечує краще відтворення блюмінгу відблиску порівняно з  $F_{SH}$ , що дає можливість усунути її суттєвий недолік — неприродне відтворення зони затухання відблиску й надлишкові обчислення.

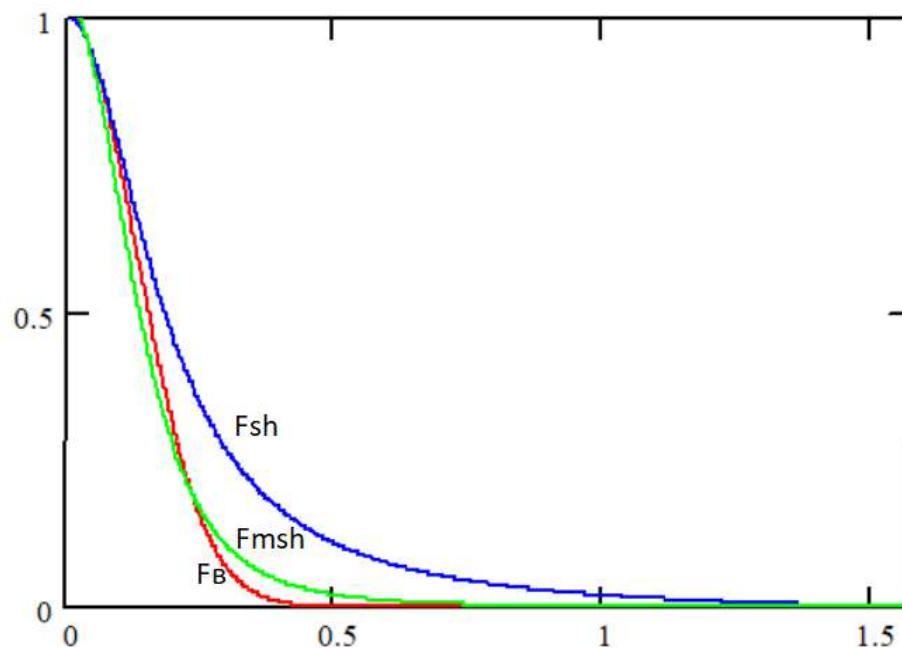


Рисунок 2.2 – Графіки  $F_{MSH}$ ,  $F_{SH}$ ,  $F_B$  ( $n=60$ )

Порівняємо максимальні відносні похибки  $F_{MSH}$  і  $F_{SH}$  ( $\delta_{msh}, \delta_{sh}$  відповідно) при апроксимації  $F_B$  у точці, де значення  $F_B = 0.3$ . Цей рівень відповідає за епіцентр відблиску.

Знайдемо абсцису цієї точки ( $x_{ep}$ ) для еталонної функції  $F_B$ , яка визначається за виразом

$$a \cos\left(\exp\left(\frac{-1.2}{n}\right)\right) \quad (2.4)$$

При  $n$  на проміжку  $[2, 1000]$   $\delta_{msh}$  менше  $\delta_{sh}$  (не більше 16% проти не більше 52%) (рисунок 2.3).

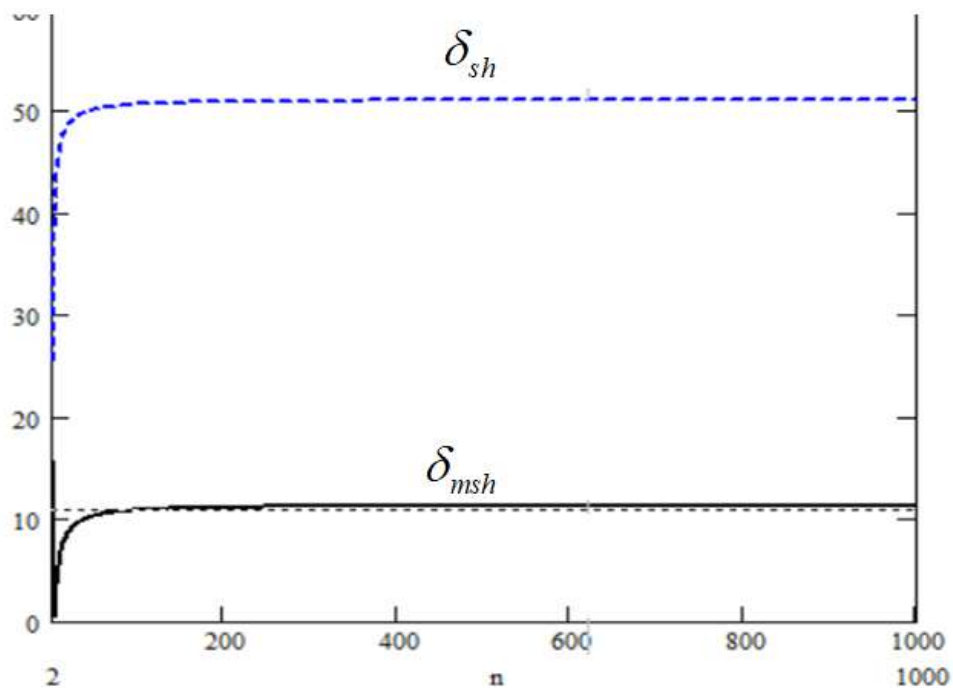


Рисунок 2.3 –  $\delta_{msh}$  і  $\delta_{sh}$  у точці  $x_{ep}$

Формула для обчислення абсолютної похибки при апроксимації  $F_B$  має вигляд

$$|F_{apr} - \cos(x)^n|, \quad (2.5)$$

де  $F_{apr}$  – апроксимація  $F_B$ .

Побудуємо тривимірні графіки (рисунок 2.4) абсолютних похибок  $F_{MSH}$ ,  $F_{SH}$  від  $F_B$ . ( $\Delta_{F_{MSH}}$ ,  $\Delta_{F_{SH}}$  відповідно) залежно від  $n$  і кута.

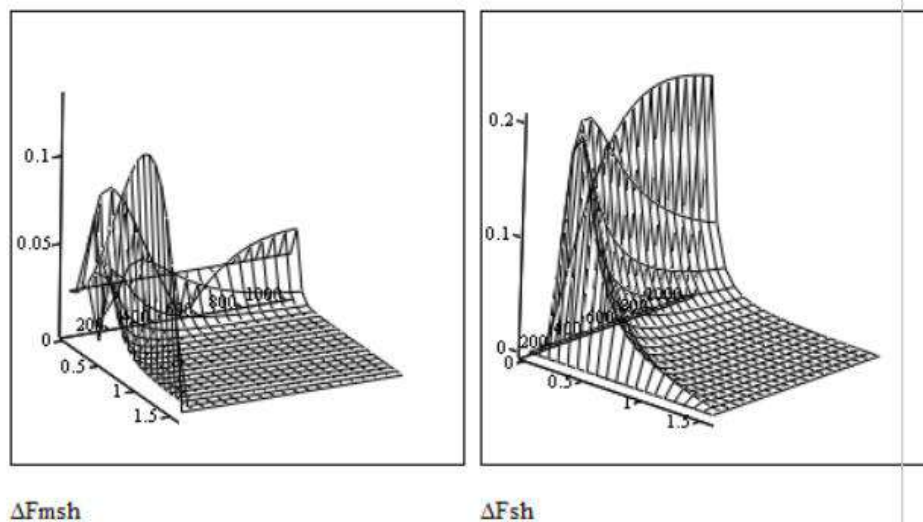


Рисунок 2.4 – Графіки  $\Delta_{F_{MSH}}$ ,  $\Delta_{F_{SH}}$  відносно  $n$  і кута

Отже, з графіків встановлюємо, що найбільша  $\Delta_{F_{MSH}}$  менша за найбільшу  $\Delta_{F_{SH}}$  приблизно удвічі.

Знайдемо значення аргументів, за яких  $F_{MSH}$  досягає нульового значення. Для цього прирівнюємо  $F_{MSH}$  до нуля. Знаходимо, що  $F_{MSH}$  досягає нуля при  $x = 1.560796$ , що наближено дорівнює  $\frac{\pi}{2}$ .

Доведемо, що функція  $F_{MSH}$  монотонно спадає. Для цього знайдемо похідну, яка дорівнює виразу

$$\frac{-1.6 * \sin(x)}{(n - n * \cos(x) + 1.25 \cos(x))^2} - \frac{3.2 * \cos(x)}{(n - n * \cos(x) + 1.25 * \cos(x))^3} (n \sin(x) - 1.25 \sin(x)) \quad (2.6)$$

На рисунку 2.5 наведено графік зміни похідної від  $n$  і кута  $x$ .

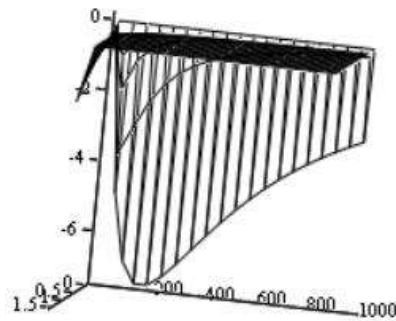


Рисунок 2.5 – Графік похідної функції

Функція монотонно спадає, оскільки її похідна від'ємна.

Обчислимо співвідношення розмірів відблисків  $F_{MSH}$  і  $F_{SH}$  зр відносно  $F_B$ . Введемо граничне значення  $2^{-q}$ , яке визначає аргумент, за межами якого розрахунок ДФВЗ припиняється.

Спершу запишемо нерівність для  $F_{SH}$ , яка визначає інтервал розрахунку аргументу для визначення спекулярної складової кольору

$$0 \leq \frac{\cos x}{n - n \cos(x) + \cos(x)} \leq 2^{-q} \quad (2.7)$$

В результаті розв'язання нерівності отримуємо вираз

$$0 \leq x \leq a \cos\left(\frac{n}{2^q + n - 1}\right) \quad (2.8)$$

Аналогічно запишемо нерівність для визначення інтервалу зміни значення для  $F_{MSH}$

$$0 \leq \frac{2 \cos x}{1.25(n - n \cos(x) + 1.25 \cos(x))^2} \leq 2^{-q} \quad (2.9)$$

Знаходимо значення кута  $x$ , що визначається нерівністю

$$0 \leq x \leq a \cos\left(\frac{4(16 * 2^q + 20n^2 - 25n - 4 * 2^{q+0.5}(8 + 20 * 2^{-q} n^2 - 25 * 2^{-q} n)^{\frac{1}{2}})}{5(16n^2 - 40n + 25)}\right) \quad (2.10)$$

Для  $F_B$  нерівність визначається

$$0 \leq \cos(x)^n \leq 2^{-q} \quad (2.11)$$

Результатом розв'язку є значення  $x$  відносно нерівності

$$0 \leq x \leq a \cos(2^{-\frac{q}{n}}) \quad (2.12)$$

Обчислимо  $\varkappa$ . Для  $F_{SH}$  відношення  $\varkappa$  розраховується

$$\frac{a \cos\left(\frac{n}{2^q + n - 1}\right)}{a \cos(2^{-\frac{q}{n}})} \quad (2.13)$$

Для  $F_{MSH}$  відношення  $\varkappa$  обчислюється

$$\frac{a \cos\left(\frac{4(16 * 2^q + 20n^2 - 25n - 4 * 2^{q+0.5}(8 + 20 * 2^{-q} n^2 - 25 * 2^{-q} n)^{\frac{1}{2}})}{5(16n^2 - 40n + 25)}\right)}{a \cos(2^{-\frac{q}{n}})} \quad (2.14)$$

На рисунку 2.6 наведено графіки відношень  $\varpi$  для  $F_{MSH}$  і  $F_B$  (А),  $F_{SH}$  і  $F_B$  (В).

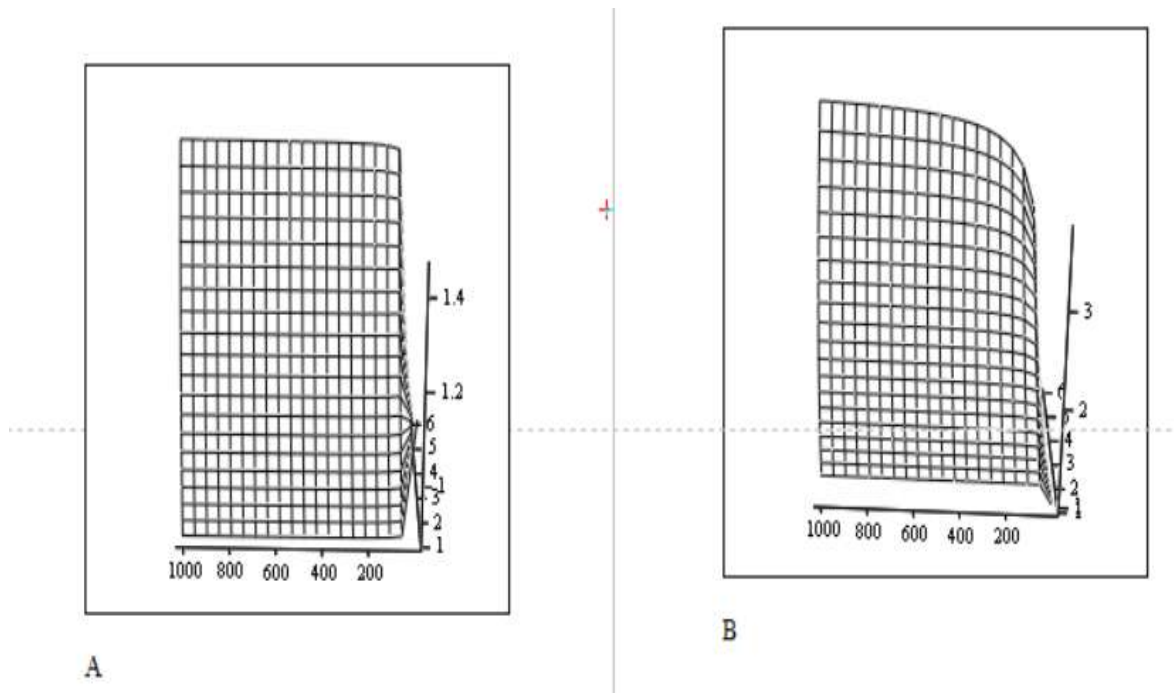


Рисунок 2.6 – Графіки відношень  $\varpi$  для  $F_{MSH}$  і  $F_B$  (А),  $F_{SH}$  і  $F_B$  (В)

Отже, для розробленої функції блюмінг відповідає меншому проміжку чисел.

Обчислені граничні значення аргументів ДФВЗ використаємо при побудові графіків максимальних відносних похибок ( $\delta$ )  $F_{MSH}$ ,  $F_{SH}$  залежно від  $n$ .

Вираз для визначення  $\delta$  апроксимації ДФВЗ відносно  $F_B$  має такий вигляд

$$\left| \frac{100(F_{apr} - \cos(x)^n)}{\cos(x)^n} \right| \quad (2.15)$$

Для обчислення максимального  $\delta$  апроксимації ДФВЗ по кожному  $n$  з врахуванням граничних значень у Mathcad 2001 було розроблено програму, граф-схема програми наведена на рисунку 2.7.

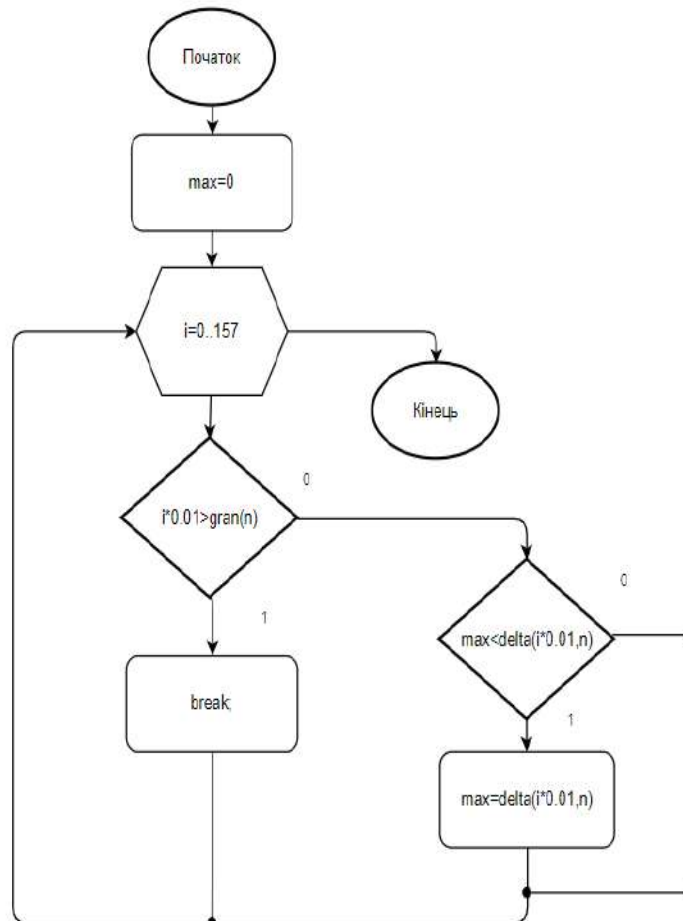


Рисунок 2.7 – Граф-схема алгоритму розрахунку максимальної  $\delta$  для визначеного  $n$

У циклі множник  $i*0.01$  відповідає значенню кута в радіанах  $i$  порівнюється із обчисленим граничним значенням розрахунку для ДФВЗ  $gran(n)$ . Якщо кут більший, здійснюється вихід з циклу оператором `break`. Інакше тимчасове максимальне значення  $\delta$  (змінна `max`) порівнюється з обчисленим  $\delta$  для нового кута (результат функції `delta`). Якщо значення `delta` більше, змінна `max` оновлюється.

На рисунку 2.8 зображено побудовані графіки максимальних  $\delta F_{MSH}$ ,  $F_{SH}$  відносно  $n$ . Значення для кожного  $n$  обчислені через розроблену програму. Граничним значенням кута при обчисленні є результат виразу (2.10).

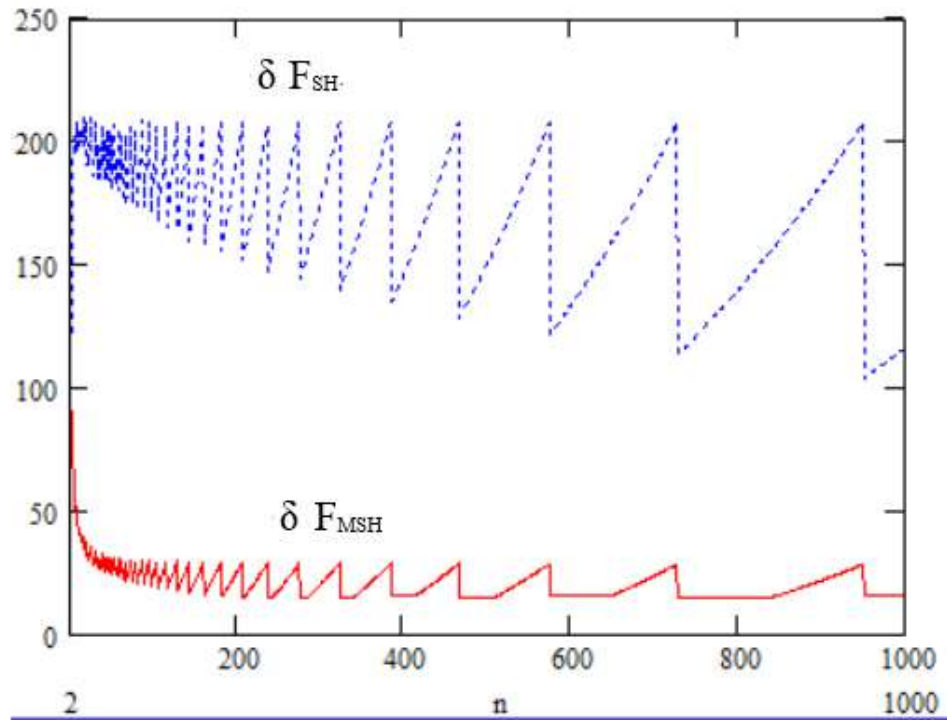


Рисунок 2.8 – Залежність  $\delta F_{MSH}$  і  $\delta F_{SH}$  від  $n$

З побудовано графіка встановлюємо, що максимальні  $\delta F_{MSH}$  є меншими, ніж максимальні  $\delta F_{SH}$  у 1,5 – 7 разів для послідовності  $n$ .

## 2.2 Знаходження нормуючого коефіцієнта для моделі відбиття

Для фізичної реалістичності ДФВЗ має відповідати закону збереження енергії. Частка випромінювання, відбита від точки поверхні у півсфері не повинна перевищувати 1 [35].

Умову можна виразити через вираз інтегралу



$$\int_{\Omega} f_r(\omega, \omega_r) * \cos(\theta) d\omega \leq 1, \quad (2.16)$$

де  $d\omega$  – тілесний кут, обчислюється:  $\sin\theta d\theta d\varphi$ .

Тому здійснимо розрахунок нормуючих коефіцієнтів для вдосконаленої моделі Шліка.

Позначимо шуканий коефіцієнт як  $coef(n)$  і формуємо рівняння для розв'язку

$$coef(n) \int_{\Omega} f_r(\omega, \omega_r) * \cos(\theta) d\omega = 1 \quad (2.17)$$

Підставляємо у виразі замість  $f_r(\omega, \omega_r)$  формулу вдосконаленої ДФВЗ Шліка, отримуємо

$$coef(n) * \int_{\Omega} \frac{2 \cos(\theta)}{1.25(n - n * \cos(\theta) + 1.25 \cos(\theta))^2} \cos(\theta) d\omega = 1 \quad (2.18)$$

Перейдемо до виразу інтегралу у сферичній системі координат. За нуль ДФВЗ взято приблизне значення  $\frac{\pi}{2}$ . Отримуємо

$$\int_0^{2\pi} \int_0^{\pi/2} \frac{2 \cos(\theta)}{1.25(n - n * \cos(\theta) + 1.25 \cos(\theta))^2} \cos(\theta) \sin(\theta) d\theta d\varphi \quad (2.19)$$

Знайдемо значення визначеного інтегралу по  $d\varphi$ . Інтеграл набуває вигляду згідно виразу

$$2\pi \int_0^{\pi/2} \frac{2 \cos(\theta)}{1.25(n - n * \cos(\theta) + 1.25 \cos(\theta))^2} \cos(\theta) \sin(\theta) d\theta \quad (2.20)$$

Для інтегралу введемо позначення  $Int(n)$ . Знайдемо вираз за формулою

$$Int(n) = \int_0^{\pi/2} \frac{2 \cos(\theta)}{1.25(n - n * \cos(\theta) + 1.25 \cos(\theta))^2} \cos(\theta) \sin(\theta) d\theta \quad (2.21)$$

Тоді рівняння для знаходження нормуючого коефіцієнта має такий вигляд

$$coef(n) * 2\pi * Int(n) = 1 \quad (2.22)$$

З даного виразу визначаємо нормуючий коефіцієнт  $coef(n)$  для одного  $n$  за формулою

$$coef(n) = \frac{1}{2\pi * Int(n)} \quad (2.23)$$

Обчислюємо значення визначених інтегралів  $Int(n)$  для коефіцієнтів спекулярності поверхні  $n \in [1, 1000]$ . Розраховуємо значення  $coef(n)$  відповідно до кожного  $n$  на проміжку. Вибірку значень  $coef(n)$  розміром 50 збережемо у текстовому файлі (рисунок 2.9). На основі розрахованих даних визначимо формулу залежності значення коефіцієнта від  $n$ .

```
n, coef(n)
1, 0.419
2, 0.6
3, 0.765
5, 1.073
7, 1.367
9, 1.653
10, 1.793
30, 4.476
40, 5.779
50, 7.071
70, 9.634
```

Рисунок 2.9 – Частина розрахованих значень коефіцієнтів для  $n$

Для знаходження нормуючого коефіцієнта застосовано програмний засіб TuringBot. Як вхідні дані розрахунку використано текстовий файл з вибіркою даних, метрикою обрано середнє квадратичне (RMS) (рисунок 2.10).

The screenshot shows the 'Input' and 'Search options' sections of the TuringBot interface. In the 'Input' section, the 'Input file' is 'знач5.txt', the 'Target variable' is 'x', and the 'Input variables' are 'n' (checked) and 'Row number' (unchecked). In the 'Search options' section, the 'Search metric' is 'RMS error', 'Train/test split' is 'No cross-validation', and 'Test sample' is 'Chosen randomly'.

Рисунок 2.10 – Вибір налаштувань для обчислення формули нормуючого коефіцієнта у TuringBot

Для вхідних даних у програмі отримано графік залежності ідеальних значень нормуючого коефіцієнта відносно  $n$  (рисунок 2.11, вісь абсцис – номер у вибірці, вісь ординат – значення коефіцієнта).

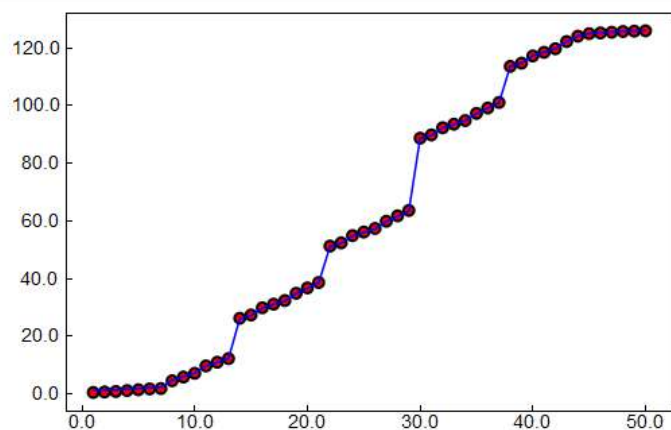


Рисунок 2.11 – Графік залежності розрахованих значень  $coef(n)$  від  $n$

У результаті отримано набір формул нормуючого коефіцієнта різних рівнів точності та складності (рисунок 2.12).

Solutions		
Size	Error	Function
1	46,102308	62.2421
3	0,504588	0.126348*n
5	0,188049	0.12533*(6.22623+n)
8	0,111746	0.122027*(sqrt(n)+n)
10	0,063278	(0.124905-(1.12537/(-9.64838-n)))*n
12	0,020065	0.124723*n-(45.76/(n+46.5778))+1.31686

Рисунок 2.12 – Отримані варіанти формул нормуючих коефіцієнтів для вдосконаленої ДФВЗ Шліка

Було обрано останню формулу зі списку, враховуючи точність і прийнятну складність обчислення. Отже, отримана формула нормуючого коефіцієнта розраховується

$$0.125n - \left( \frac{45.76}{n + 46.578} \right) + 1.317 \quad (2.24)$$

На рисунку 2.13 зображено графік абсолютної похибки виразу  $coef(n) * 2\pi * Int(n)$  від 1. Максимальна похибка на проміжку  $n \in [2, 1000]$  не перевищує  $42 * 10^{-3}$ .

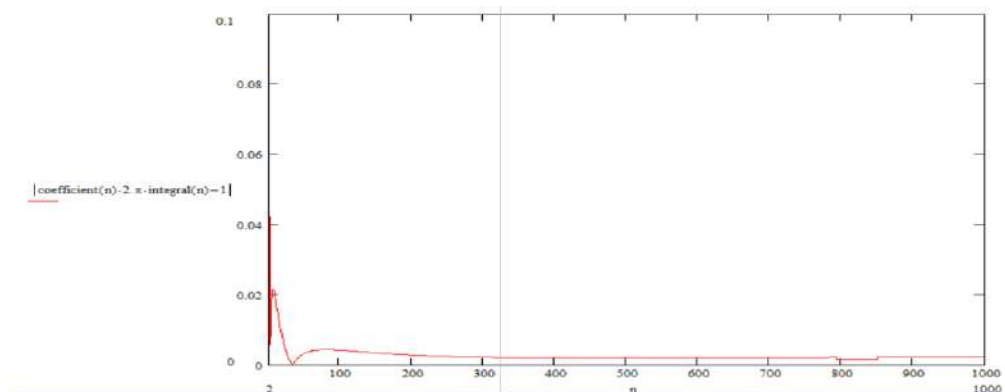


Рисунок 2.13 – Залежність абсолютної похибки виразу від  $n$

### 2.3 Розробка ДФВЗ на основі двох утворюючих функцій

Розробимо ДФВЗ на основі двох утворюючих функцій: функції Шліка та модифікованої функції Шліка. Позначимо функцію як  $F_{dsh}$ .

$F_{dsh}$  обчислюється за формулою

$$\frac{\cos(x)}{n - n * \cos(x) + \cos(x)} + \left( \left( \frac{a_1 (\cos(x) + c_1)^{b_1}}{a_2 (a_3 n^{b_3} - a_4 (n * \cos(x))^{b_4} + a_5 \cos(x)^{b_5})^{b_2} + c_2} \right)^{b_6} - c_3 \right), \quad (2.25)$$

де  $a_n, b_n, c_n$  – шукані множники, степені, константи.

Для визначення значень  $a_n, b_n, c_n$  використано комп'ютерну програму перебору значень (див. рис. 2.1). До програми був доданий третій етап підбору констант  $c_n$ .

На рисунку 2.14 зображено 5 найбільш точних (відносно абсолютного відхилення значень  $F_{dsh}$  від значень функції Блінна-Фонга) сукупностей множників, степенів, констант для формули (2.25). Значення невідомих розташовані у порядку, що відповідає порядку індексів  $a_n, b_n, c_n$ .

```

модель 0 степені 16 2 1 1 16 1 коефіцієнти 2 1,25 1,25 1,25 1,25 доданки 0 0 0,1 точність 0,007421892
модель 1 степені 4 1 1 1 8 2 коефіцієнти 1,25 0,5 2,5 2,5 2,5 доданки 0,2 1,1 0,25 точність 0,0034289
модель 2 степені 8 1 1 1 16 2 коефіцієнти 1,5 0,5 3 3 3 доданки 0,15 2,25 0,5 точність 0,003304745403
модель 3 степені 8 2 1 1 8 1 коефіцієнти 2 1,25 1,25 1,25 1,25 доданки 0,1 1,5 0,25 точність 0,002505
модель 4 степені 2 1 1 1 4 2 коефіцієнти 1,5 0,5 3 3 3 доданки 0,25 0,5 0,25 точність 0,0029209517294

```

Рисунок 2.14 – Підібрані програмою сукупності коефіцієнтів для  $F_{dsh}$

Для  $F_{dsh}$  взято значення змінних моделі 3, враховуючи найбільшу точність.

Тоді  $F_{dsh}$  розраховується за формулою

$$\frac{\cos(x)}{n - n \cdot \cos(x) + \cos(x)} + \left( \frac{2(\cos(x) + 0.1)^8}{1.25(1.25n - 1.25(n \cdot \cos(x)) + 1.25 \cos(x)^8)^2 + 1.5} - 0.25 \right) \quad (2.26)$$

На рисунку 2.15 зображено графіки  $F_{dsh}$ ,  $F_{sh}$ ,  $F_B$  відносно  $n$  20, 200. Графік  $F_{dsh}$  характеризується меншим відхиленням від  $F_B$ , ніж  $F_{sh}$ .

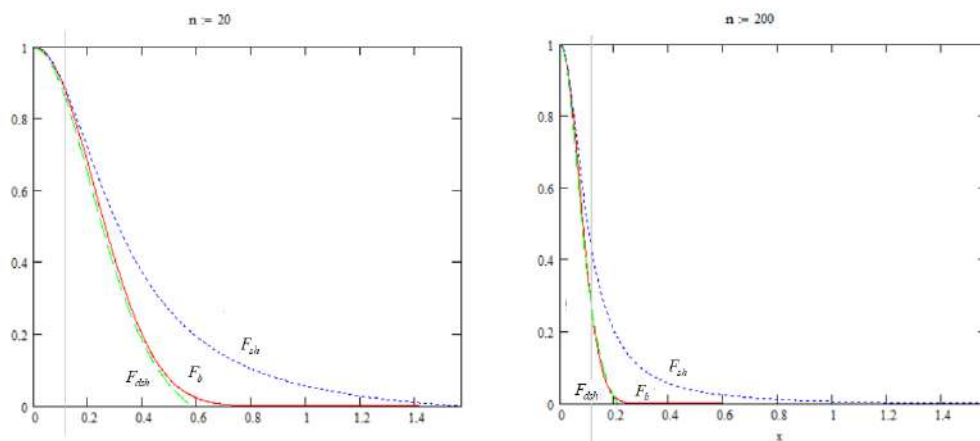


Рисунок 2.15 – Графіки  $F_{dsh}$ ,  $F_{sh}$ ,  $F_b$

Порівняємо відносні похибки  $F_{dsh}$ ,  $F_{sh}$  від  $F_B$  (відповідно  $\delta_{dsh}$ ,  $\delta_{sh}$ ) у точці, де значення  $F_B = 0.3$  (формула (2.4)). Графіки  $\delta_{dsh}$ ,  $\delta_{sh}$  у даній точці зображено на рисунку 2.16.

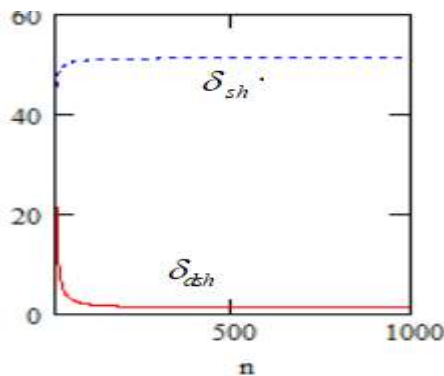


Рисунок 2.16 – Графіки  $\delta_{dsh}$ ,  $\delta_{sh}$  у точці рівня епіцентру для  $n \in [9, 1000]$

$\delta_{sh}$  не перевищує 52%,  $\delta_{dsh}$  не перевищує 22% на проміжку  $n \in [9, 1000]$ .

На рисунку 2.17 зображено тривимірний графік абсолютних похибок  $F_{dsh}$ ,  $F_{SH}$  від  $F_B(\Delta_{F_{dsh}}, \Delta_{F_{SH}})$  залежно від  $n$  і значення кута.

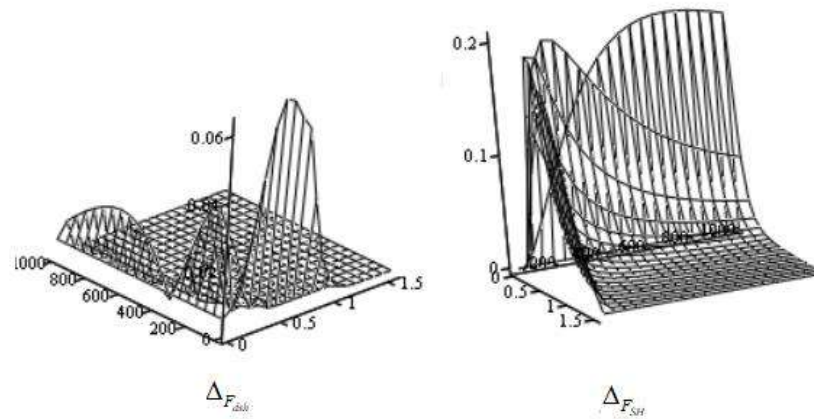


Рисунок 2.17 – Графіки зміни  $\Delta_{F_{dsh}}, \Delta_{F_{SH}}$

$\Delta_{F_{SH}}$  не перевищує 0.2,  $\Delta_{F_{dsh}}$  не перевищує 0.06.

Знайдемо значення аргументу, до якого доцільно обчислювати  $F_{dsh}$ .

Отримуємо нерівність

$$0 \leq \frac{\cos(x)}{n - n^* \cos(x) + \cos(x)} + \left( \frac{2(\cos(x) + 0.1)^8}{1.25(1.25n - 1.25(n^* \cos(x)) + 1.25 \cos(x)^8)^2 + 1.5} - 0.25 \right) \leq 2^{-q} \quad (2.27)$$

Встановлюємо, що граничне значення  $x$  наближено обчислюється за формулою

$$\frac{1.134}{\sqrt{\left(\frac{1.295 + n}{q}\right) + 0.439}} \quad (2.28)$$

Знайдемо відношення розмірів плям  $F_{dsh}$  і  $F_B$   $\varpi$ , що обчислюється за формулою

$$\frac{1.134}{\sqrt{\left(\frac{1.295+n}{q}\right)+0.439}} \cdot \frac{-q}{a \cos(2^n)} \quad (2.29)$$

На рисунку 2.18 наведено графік відношення  $\varpi$  для  $F_{dsh}$  і  $F_B$  (В).

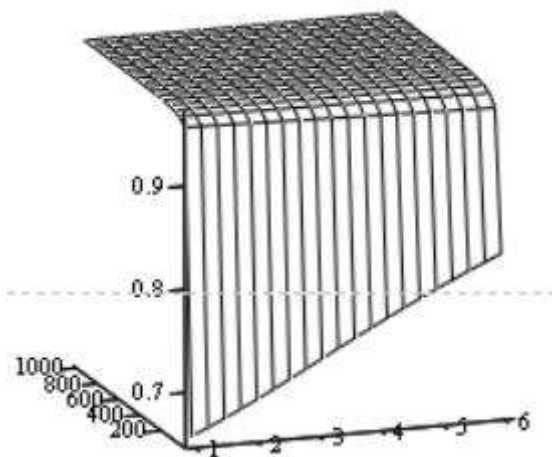


Рисунок 2.18 – Графік  $\varpi$  для  $F_{dsh}$  і  $F_B$  відносно  $n, q$

Отже,  $F_{dsh}$  характеризується значно меншим інтервалом розрахунку значення, ніж  $F_{SH}$  (див. рис. 2.6 (В)).

Порівняємо максимальні  $\delta$   $F_{dsh}$ ,  $F_{SH}$  відносно  $n$ . Для обчислення максимальної  $\delta$  відносно кожного  $n$  використано програму у MathCad (див. рис. 2.7).  $\delta$  для  $F_{SH}$ ,  $F_{dsh}$  обчислюються до граничного значення аргументу  $F_{dsh}$  при  $q=3$ . Графік максимальних  $\delta$   $F_{dsh}$ ,  $F_{SH}$  відносно  $n$  зображено на рисунку 2.19.



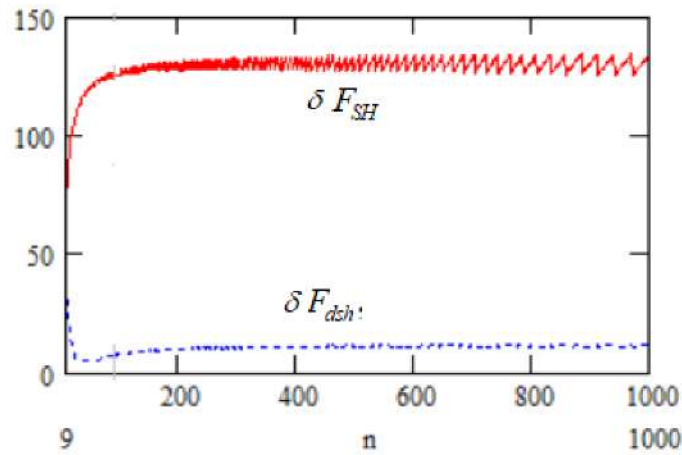


Рисунок 2.19 – Залежність максимальних  $\delta F_{dsh}$  і  $\delta F_{SH}$  від  $n$

Максимальне значення на графіку  $\delta F_{dsh}$  становить 31%,  $\delta F_{SH}$  – 134%.

Таким чином, у результаті проведення досліджень отримано нову модифіковану ДФВЗ.

## 2.4 Висновки

На основі ДФВЗ Шліка розроблено нову дистрибутивну функцію відбивної здатності. Для підбору оптимальних коефіцієнтів і степенів ДФВЗ розроблено комп'ютерну програму перебору. Досліджено розроблену ДФВЗ. Показано більш швидке падіння до нуля значень розробленої ДФВЗ, ніж значень ДФВЗ Шліка. Розроблена ДФВЗ характеризується меншою відносною похибкою у точці, що відповідає епіцентру відблиску, меншими максимальними відносними похибками залежно від  $n$ , меншою максимальною абсолютною похибкою. Для розробленої ДФВЗ обчислено нормуючий коефіцієнт, що застосовується для забезпечення фізичної коректності функції.

Розроблено нову ДФВЗ, що поєднує функцію Шліка та модифіковану функцію Шліка. Аналогічно досліджено математичні особливості отриманої функції. ДФВЗ на основі двох утворюючих функцій характеризується більш точним наближенням значень ДФВЗ Блінна, ніж ДФВЗ Шліка.

### 3 МОДИФІКАЦІЯ КОСИНУС-КВАДРАТИЧНОЇ ТА КВАДРАТИЧНОЇ МОДЕЛЕЙ ОСВІТЛЕННЯ

#### 3.1 Модифікація косинус-квадратичної моделі освітлення

Однією з апроксимацій ДФВЗ Блінна є косинус-квадратична функція, що розраховується за формулою [36]

$$\left( \frac{n}{2} (\cos(x) - 1) + 1 \right)^2 \quad (3.1)$$

Позначимо дану функцію як  $F_{SCOS}$ . На рисунку 3.1 зображено графіки ДФВЗ Блінна ( $F_B$ ), ДФВЗ Шліка ( $F_{SH}$ ),  $F_{SCOS}$  для  $n = 50$ .

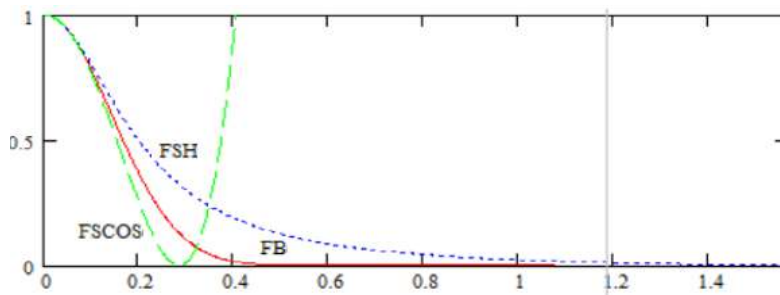


Рисунок 3.1 – Графік  $F_{SCOS}$ ,  $F_B$ ,  $F_{SH}$  при  $n = 50$

Значення графіку  $F_{SCOS}$  недостатньо точно наближають значення графіку  $F_B$ . Отже, необхідним є підвищення точності апроксимації  $F_{SCOS}$ .

$F_{SCOS}$  розраховується до точки, де її значення досягає нуля. Дана точка розраховується за формулою

$$a \cos\left(\frac{n-2}{n}\right) \quad (3.2)$$

Здійснимо модифікацію  $F_{scos}$ . Замінімо  $\frac{1}{2}$  змінною  $coef$ , отримуємо вираз

$$(coefn(\cos(x) - 1) + 1)^2 \quad (3.3)$$

Прирівняємо формулу (3.3) до формули  $F_B$ . Отримуємо рівняння

$$(coefn(\cos x - 1) + 1)^2 = \cos(x)^n \quad (3.4)$$

Розв'язуємо дане рівняння відносно  $coef$ . Отримуємо формулу для визначення  $coef$

$$\frac{-1 \pm \cos(x)^{\frac{n}{2}}}{(\cos(x) - 1)n} \quad (3.5)$$

У складовій виразу  $\cos(x)^{\frac{n}{2}}$  наявне зростання обчислень зі збільшенням  $n$ . Тому замінімо дану складову на  $F_{SH}$ , де  $n$  замінюємо на  $\frac{n}{2}$ . Формула для визначення  $coef$  набуває вигляду

$$\frac{-1 \pm \left( \frac{\cos(x)}{\frac{n}{2} - \frac{n}{2} * \cos(x) + \cos(x)} \right)}{(\cos(x) - 1)n} \quad (3.6)$$

Спростимо даний вираз. При виборі знаку  $-$  у верхній частині виразу отримуємо формулу визначення  $coef$

$$\frac{-(-n + n \cdot \cos(x) - 4 \cdot \cos(x))}{n(-2n \cos(x) + n + n \cdot \cos(x)^2 - 2 \cos(x)^2 + 2 \cos(x))} \quad (3.7)$$

При виборі знаку + формула визначення *coef* має вигляд

$$\frac{-1}{(-n + n \cos(x) - 2 \cos(x))} \quad (3.8)$$

Вираз (3.8) є простішим, ніж (3.7), тому для розрахунку *coef* обираємо вираз (3.8).

Підставляємо знайдений вираз замість *coef* у (3.3). Отримуємо вираз

$$\left( \frac{-1}{(-n + n \cos(x) - 2 \cos(x))} n(\cos(x) - 1) + 1 \right)^2 \quad (3.9)$$

Позначимо модифіковану косинус-квадратичну ДФВЗ як  $F_{SCOS2}$ . На рисунку 3.2 зображено графіки  $F_{SCOS2}$ ,  $F_{SCOS}$ ,  $F_B$  для  $n = 50$ .

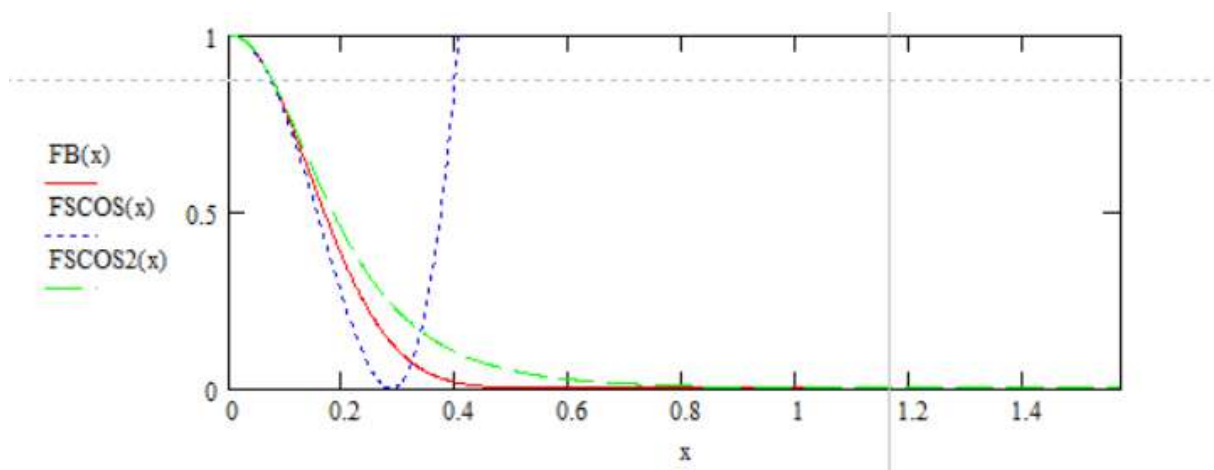


Рисунок 3.2 – Графік  $F_{SCOS}$ ,  $F_B$ ,  $F_{SCOS2}$  при  $n = 50$

Графік  $F_{SCOS2}$  занадто повільно падає, тому замінимо чисельник  $-1$  у знаменнику коефіцієнта на  $-1.1$ . Тоді  $F_{SCOS2}$  обчислюється за формулою

$$\left( \frac{-1.1}{(-n + n \cos(x) - 2 \cos(x))} n(\cos(x) - 1) + 1 \right)^2 \quad (3.10)$$

На рисунку 3.3 зображено оновлені графіки  $F_{SCOS2}$ ,  $F_{SCOS}$ ,  $F_B$  для  $n = 50$ .

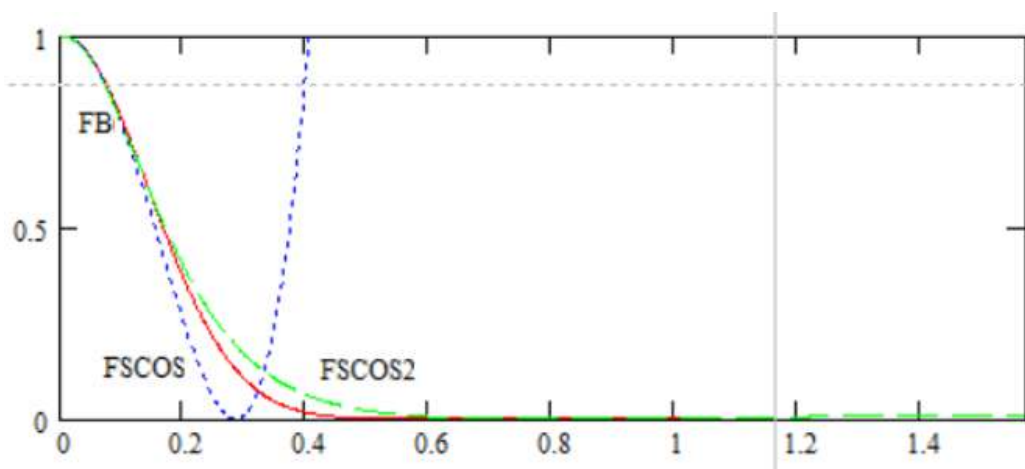


Рисунок 3.3 – Графік  $F_{SCOS}$ ,  $F_B$ ,  $F_{SCOS2}$  при  $n = 50$  після оновлення  $F_{SCOS2}$

$F_{SCOS2}$  повільніше досягає нульового значення, ніж  $F_{SCOS}$ , однак більш точно наближає  $F_B$ .

Порівняємо відносні похибки  $F_{SCOS}$ ,  $F_{SCOS2}$  від  $F_B$  (відповідно  $\delta_{SCOS}, \delta_{SCOS2}$ ) у точці перегину  $F_B$  (границі епіцентру відблиску). Точка перегину  $F_B$  розраховується за формулою

$$\arctg\left(\frac{1}{\sqrt{n-1}}\right) \quad (3.11)$$

На рисунку 3.4 зображено  $\delta_{scos}, \delta_{scos2}$  для точки перегину  $F_B$  на проміжку  $n \in [3, 1000]$ .

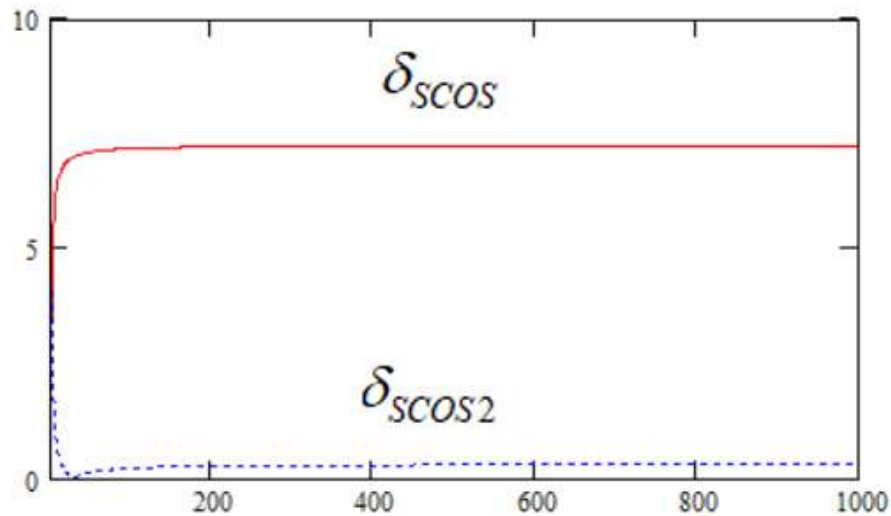


Рисунок 3.4 –  $\delta_{scos}, \delta_{scos2}$  у точці перегину  $F_B$

$\delta_{scos2}$  не перевищує 4% (після  $n = 30$  не більше 0.3%),  $\delta_{scos}$  не перевищує 7%.

Прирівнюємо вираз (3.10) до 0. Знаходимо значення аргументу, при якому  $F_{scos2}$  дорівнює 0. Тоді значення аргументу розраховується за формулою

$$a \cos\left(\frac{n}{n+20}\right) \quad (3.12)$$

За допомогою розробленої програми у Mathcad (див. рис. 2.7) обчислимо максимальні абсолютні похибки для  $F_{scos}, F_{scos2}$  від  $F_B$  (відповідно  $\Delta_{scos}, \Delta_{scos2}$ ) відносно  $n \in [2, 1000]$ .  $\Delta_{scos}, \Delta_{scos2}$  розраховувалась до нулів функцій (відповідно формули (3.2), (3.12)). Програму було модифіковано - замість підфункції

знаходження відносної похибки використано підфункцію обчислення абсолютної похибки.

На рисунку 3.5 зображено максимальні  $\Delta_{scos}, \Delta_{scos2}$  для  $n \in [2, 1000]$ .

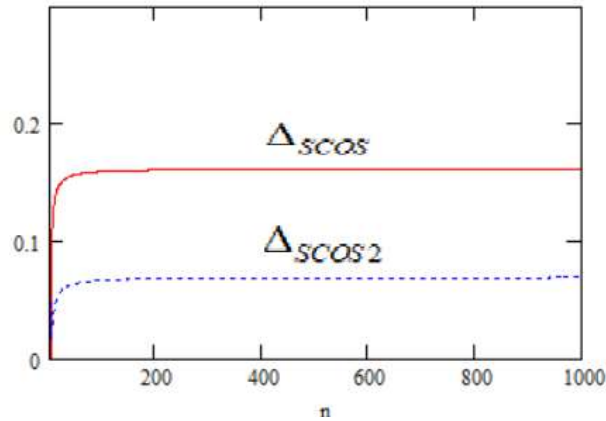


Рисунок 3.5 –Графік максимальних  $\Delta_{scos}, \Delta_{scos2}$  для  $n \in [2, 1000]$

Максимальна  $\Delta_{scos2}$  не перевищує 0.07,  $\Delta_{scos}$  не перевищує 0.16.

Визначимо максимальні  $\delta_{scos}, \delta_{scos2}$  від  $F_B$  для  $n \in [2, 1000]$ . Для обчислення максимального  $\delta$  відносно одного  $n$  використано програму (див. рис. 2.7).

Розрахунок для  $\delta_{scos}$  здійснюється до нуля  $F_{scos}$  (формула (3.2)).

Для розрахунку  $\delta_{scos2}$  знайдемо граничний аргумент обчислення  $F_{scos2}$ . Для цього розв'яжемо нерівність

$$0 \leq \left( \frac{-1.1}{(-n + n \cos(x) - 2 \cos(x))} n (\cos(x) - 1) + 1 \right) \leq 2^{-q} \quad (3.13)$$

Отримуємо межі для аргумента  $F_{scos2}$ , що визначаються нерівністю

$$0 \leq x \leq a \cos \left( \frac{n * (20 * 2^q + n * 2^q + 200 - 100n + 220 * 2^q (\frac{1}{2^q})^{\frac{1}{2}})}{n^2 * 2^q + 40n * 2^q + 400 * 2^q - 100 * n^2 + 400n - 400} \right) \quad (3.14)$$

На рисунку 3.6 зображено графіки максимальних  $\delta_{SCOS}, \delta_{SCOS2}$  для  $n \in [2, 1000]$ .

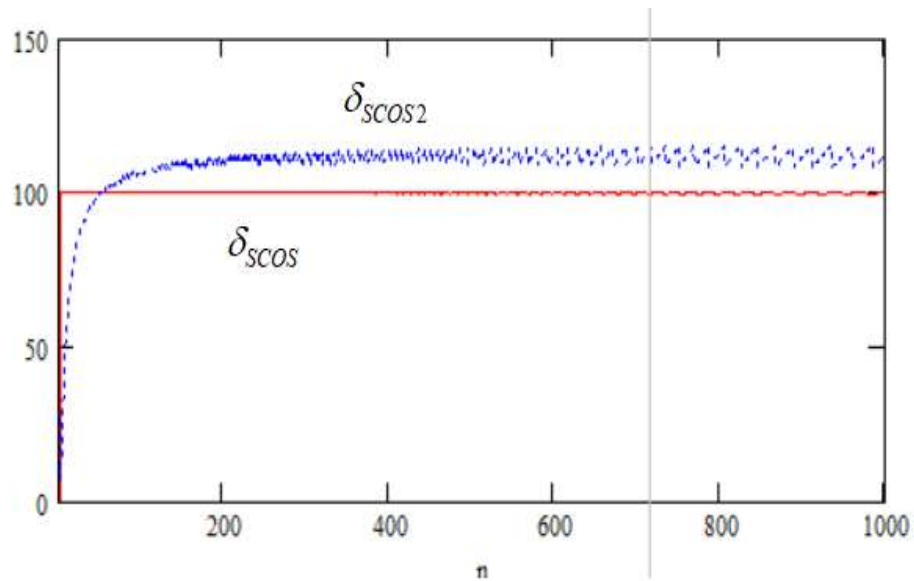


Рисунок 3.6 – Залежність максимальних  $\delta_{SCOS}, \delta_{SCOS2}$  від  $n$

Максимальна  $\delta_{SCOS2}$  не перевищує 115%, максимальна  $\delta_{SCOS}$  не більша 100%.

Отже, модифікована  $F_{SCOS2}$  характеризується дещо більшою максимальною відносною похибкою, ніж  $F_{SCOS}$ . Однак різниця у значеннях  $\delta_{SCOS}, \delta_{SCOS2}$  не є значною, також для  $F_{SCOS2}$  характерна значно менша відносна похибка у зоні епіцентру відблиску, що більш важливо.

### 3.2 Визначення значень степенів косинус-квадратичної функції

Замінімо степінь 2 та множник  $\frac{1}{2}$  у косинус-квадратичній функції на змінні  $a, \frac{1}{a}$  відповідно. Отримуємо формулу ДФВЗ



$$\left( \frac{n}{a} (\cos(x) - 1) + 1 \right)^a \quad (3.15)$$

Позначимо дану ДФВЗ як  $F_{ST}$ .

Знайдемо значення змінної  $a$ , що забезпечить найменшу абсолютну похибку значень  $F_{ST}$  від  $F_B$ .

Для обчислення значення  $a$  було розроблено комп'ютерну програму, блок-схема якої зображена на рисунку 3.7.

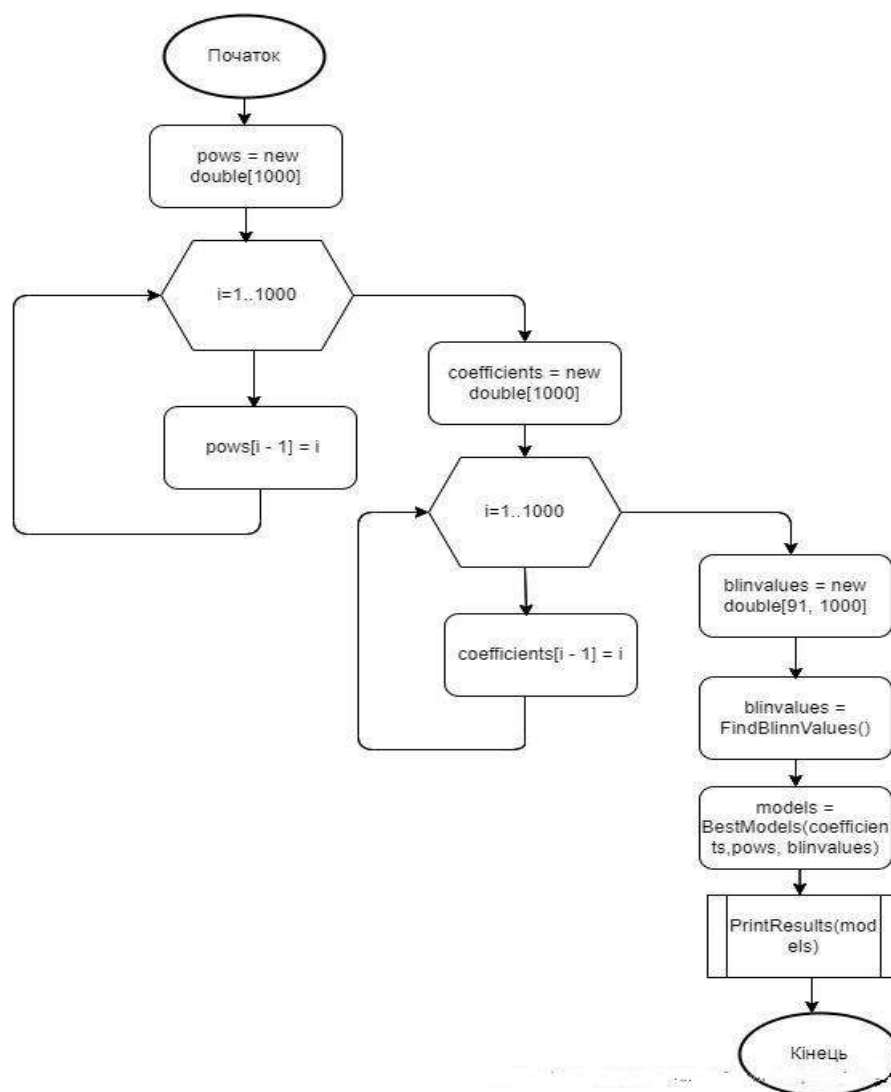


Рисунок 3.7 – Блок-схема програми визначення значень  $a$

Дана програма є оновленням програми для підбору оптимальних коефіцієнтів до модифікованої моделі Шліка (див. рис. 2.1).

Масив rows вміщує можливі значення степенів. Масив ініціалізується у циклі значеннями проміжку [1,1000]. Масив coefficients вміщує можливі значення коефіцієнтів та аналогічно ініціалізується значеннями проміжку [1,1000]. Функція FindBlinnValues повертає масив з обчисленими значеннями ДФВЗ Блінна для  $n \in [1,1000]$ , кутів  $[0, \frac{\pi}{2}]$ . Значення результату функції FindBlinnValues зберігаються у масиві blinvalues. Функція BestModels приймає як параметри масиви rows, coefficients, blinvalues. У циклі (відносно змінної  $a$ ) для  $n \in [1,1000]$ , кутів  $[0, \frac{\pi}{2}]$  підбираються значення  $a$ , що забезпечать найменшу абсолютну різницю значень  $F_{ST}$  від значень масиву blinvalues. Якщо значення  $F_{ST}$  стає менше нуля, значення враховується як 0, якщо значення  $F_{ST}$  починає зростати, береться попереднє значення. Функція PrintResults забезпечує виведення 10 найточніших значень  $a$ .

На рисунку 3.8 зображено результат роботи програми – 10 значень змінної  $a$  та середнє абсолютне відхилення масиву значень  $F_{ST}$  від масиву blinvalues.

```

модель 0 коефіцієнт 268 степiнь 268 точність 0,0004738259334582657
модель 1 коефіцієнт 267 степiнь 267 точність 0,00047382616505358354
модель 2 коефіцієнт 269 степiнь 269 точність 0,0004738262041485802
модель 3 коефіцієнт 266 степiнь 266 точність 0,00047382690743475456
модель 4 коефіцієнт 270 степiнь 270 точність 0,00047382696877599616
модель 5 коефіцієнт 265 степiнь 265 точність 0,00047382816925697544
модель 6 коефіцієнт 271 степiнь 271 точність 0,00047382821914058077
модель 7 коефіцієнт 272 степiнь 272 точність 0,00047382994718795594
модель 8 коефіцієнт 264 степiнь 264 точність 0,000473829959333625
модель 9 коефіцієнт 273 степiнь 273 точність 0,00047383214500628263

```

Рисунок 3.8 – Результат роботи програми підбору значень  $a$

Запишемо формулу  $F_{ST}$  для  $a = 268$ :

$$\left( \frac{n}{268} (\cos(x) - 1) + 1 \right)^{268} \quad (3.16)$$

Позначимо даний варіант  $F_{ST}$  як  $F_{ST268}$ .

На рисунку 3.9 зображено графіки  $F_{SCOS}$ ,  $F_{ST268}$  ( $a = 268$ ),  $F_{ST16}$  ( $a = 16$ ),  $F_B$ .

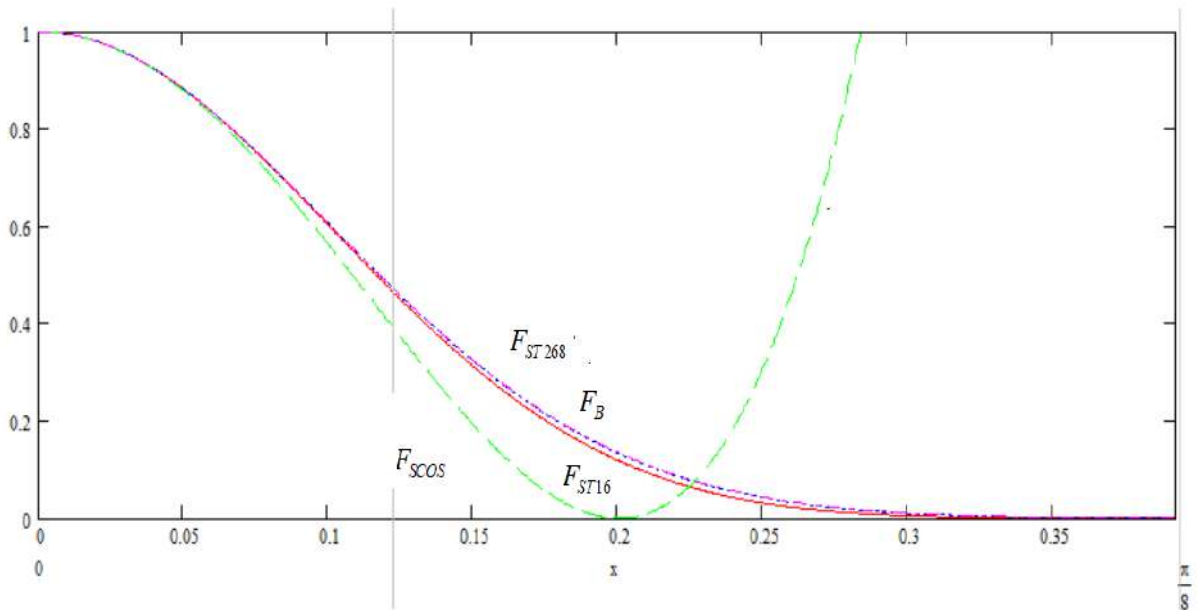


Рисунок 3.9 —  $F_{SCOS}$ ,  $F_{ST268}$ ,  $F_{ST16}$ ,  $F_B$  при  $n = 100$

Отже, встановлено що найменше середнє абсолютне відхилення значень  $F_{ST}$  від  $F_B$  має місце при  $a = 268$ .

У даному випадку обчислювальна складність є непринятною, тому варто використовувати форму  $F_{ST}$  при  $a = 16$  [35], що забезпечує майже ідеальну апроксимацію  $F_B$  з порівняно невеликою обчислювальною складністю.

### 3.3 Знаходження нормуючого коефіцієнта для квадратичної моделі освітлення

Апроксимація ДФВЗ Блінна квадратичною функцією розраховується за формулою [36]:

$$a * \cos(x)^2 + b * \cos(x) + c, \quad (3.17)$$

де  $a$ ,  $b$ ,  $c$  – коефіцієнти.

Квадратична модель повинна відповідати закону збереження енергії. Тому знайдемо нормуючий коефіцієнт для квадратичної моделі освітлення.

Встановлено [36], що  $c = 0$ ,  $b = 1 - a$ ,  $a$  наближено визначається як  $0.786n$ .

Тоді вираз визначається за формулою

$$0.786n * \cos(x)^2 + (1 - 0.786n) * \cos(x), \quad (3.18)$$

ДФВЗ досягає нуля при  $x = a \cos(2.5 * 10^{-3} * (\frac{393n - 500}{n}))$ . Позначимо даний вираз як  $null(n)$ .

Підставляємо формулу ДФВЗ у рівняння для визначення нормуючого коефіцієнта  $coef(n)$

$$coef(n) * \int_{\Omega} (0.786n * \cos(\theta)^2 + (1 - 0.786n) \cos(\theta)) \cos(\theta) d\omega = 1 \quad (3.19)$$

Подвійний інтеграл у сферичній системі координат розраховується за формулою

$$\int_0^{2\pi} \int_0^{null(n)} (0.786n * \cos(\theta)^2 + (1 - 0.786n) \cos(\theta)) \cos(\theta) \sin(\theta) d\theta d\varphi \quad (3.20)$$

Перейдемо до форми одиничного інтегралу, здійснивши інтегрування по  $d\varphi$ . Отримуємо вираз

$$2\pi \int_0^{null(n)} (0.786n * \cos(\theta)^2 + (1 - 0.786n) \cos(\theta)) \cos(\theta) \sin(\theta) d\theta \quad (3.21)$$

Позначимо вираз інтегралу як  $Integral(n)$

$$Integral(n) = \int_0^{null(n)} (0.786n * \cos(\theta)^2 + (1 - 0.786n) \cos(\theta)) \cos(\theta) \sin(\theta) d\theta \quad (3.22)$$

Підставляємо  $Integral(n)$  у рівняння (3.19), що набуває вигляду

$$coef(n) * 2\pi * Integral(n) = 1 \quad (3.23)$$

Тоді значення нормуючих коефіцієнтів для  $n$  знаходяться за формулою

$$coef(n) = \frac{1}{2\pi * Integral(n)} \quad (3.24)$$

Обчислюємо значення  $coef(n)$  для проміжку коефіцієнтів спекулярності поверхні  $n \in [1, 1000]$ . Отримуємо набір залежностей розрахованих значень  $coef(n)$  від  $n$ .

З допомогою програмного засобу TuringBot знайдемо формулу залежності  $coef(n)$  від  $n$ , що забезпечить невелике абсолютне відхилення лівої частини рівняння (3.23) від 1.

Для цього у текстовому файлі збережено 50 значень  $n$  і відповідних їм значень  $coef(n)$  (рисунок 3.10).

```

n,x
1,0.593
20,5.222
40,10.223
60,15.227
80,20.232
120,30.245
140,35.254
160,40.265
180,45.278
200,50.293
220,55.31
240,60.331
260,65.354
280,70.381
300,75.412
320,80.466
340,85.484

```

Рисунок 3.10 – Файл залежностей  $coef(n)$  від  $n$

Вміст текстового файлу використовується TuringBot як початкові дані для підбору формули нормуючого коефіцієнта (рисунок 3.11). Точність підбраної формули нормуючого коефіцієнта визначається в програмі через середнє квадратичне значення (RMS).



Input file: знач453.txt

Find a formula that satisfies: [Help]

x = f(n)

**Search options**

Search metric: RMS error

Train/test split: No cross-validation

Test sample: Chosen randomly

Рисунок 3.11 – Вибір вхідних даних і метрики помилки у TuringBot

Задаємо максимальний розмір формули нормуючого коефіцієнта як 30 (рисунок 3.12).

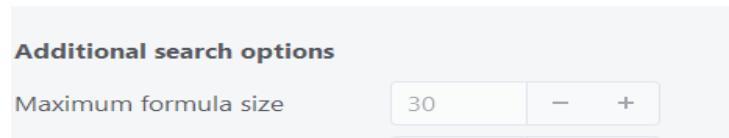


Рисунок 3.12 – Задання максимального розміру шуканої формули

Отримуємо набір можливих виразів нормуючого коефіцієнта, обираємо останнє найточніше у списку значення (рисунок 3.13).

Size	Error	Function
1	74,423765	128.708
3	0,780123	0.253783*n
5	0,644715	(-0.255088)*(3.45762-n)
8	0,196643	n/(4.01875+(-0.000103303*n))
10	0,103370	(n+1.9417)/(-0.000127589*n+4.0485)
14	0,099731	(n+1.97194)/(tan(0.785272*n)+4.04818)
17	0,094893	(n+((-40.1786)/(n+35.2391))+2.25549)/(-0.000127573*n+4.04984)
18	0,072357	(0.256147+(0.0106754*(-1.01638-(-0.000958606*n))*n/(175.176+n)))*n
19	0,027158	0.211677*(1.17488*(n+(1.26544+(n*(-0.0229706*tan(-0.000871643*n))))))

Рисунок 3.13 – Можливі формули нормуючого коефіцієнта, отримані у TuringBot

Отже, формула для нормуючого коефіцієнта має такий вигляд

$$0.212(1.175(n+(1.265+(n(-0.023tg(-0.0008n)))))) \quad (3.25)$$

Побудуємо графік значень нормуючого коефіцієнта відносно  $n$ . Визначаємо, що формулу нормуючого коефіцієнта можна апроксимувати лінійним виразом (рисунок 3.14).

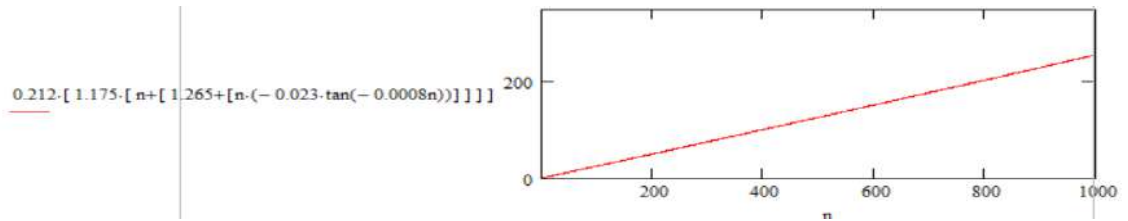


Рисунок 3.14 – Залежність значення нормуючого коефіцієнта від  $n$

Ділимо максимальне значення нормуючого коефіцієнта на максимальне значення  $n$ . Отримуємо лінійний вираз

$$0.255n \quad (3.26)$$

Апроксимована формула (3.26) нормуючого коефіцієнта доцільна для використання при  $n \in [21, 1000]$ . Максимальна абсолютна похибка  $\Delta$  між  $1$  і  $coef(n) * 2\pi * Int(n)$  на проміжку становить  $21 * 10^{-3}$  (рисунок 3.15).

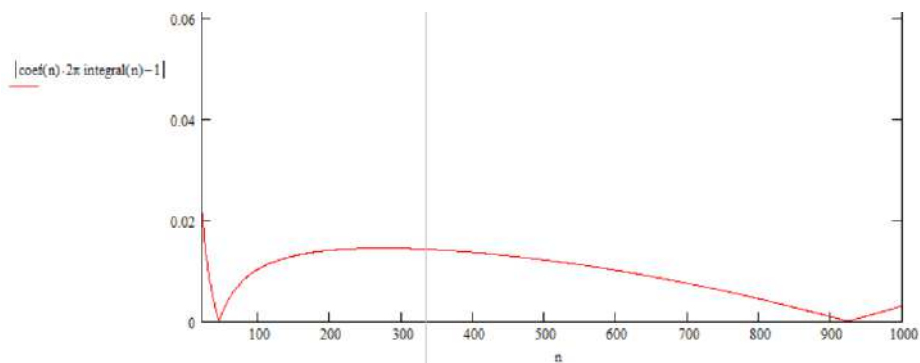


Рисунок 3.15 – Значення  $\Delta$  відносно  $n \in [21, 1000]$

Для  $n \in [1, 20]$  використовується обчислена формула (3.25) нормуючого коефіцієнта. Максимальна  $\Delta$  для проміжку дорівнює  $58 * 10^{-3}$  (рисунок 3.16).



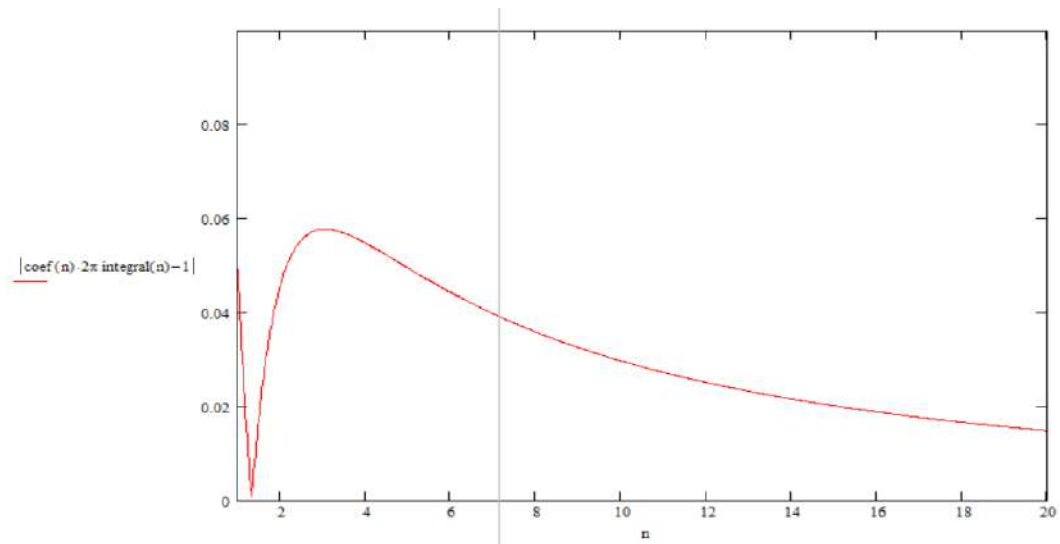


Рисунок 3.16 – Значення  $\Delta$  відносно  $n \in [1, 20]$

Отже, знайдено вираз нормуючого коефіцієнта для квадратичної моделі освітлення на проміжках  $n \in [1, 20]$ ,  $n \in [21, 1000]$ .

### 3.4 Висновки

Модифіковано косинус-квадратичну модель освітлення. Отримана модифікація більш точно наближає ДФВЗ Блінна, характеризується меншою відносною похибкою у точці перегину, меншими максимальними абсолютними похибками.

Досліджено можливі значення степенів косинус-квадратичної функції. Найбільш точне наближення ДФВЗ Блінна має місце при значенні степеня 268, однак дане значення є неприпустимим відносно обчислювальної складності.

Знайдено нормуючий коефіцієнт для квадратичної моделі освітлення.

## 4 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ПОРІВНЯННЯ МОДЕЛЕЙ ВІДБИТТЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНИХ ФУНКЦІЙ ВІДБИВНОЇ ЗДАТНОСТІ

### 4.1 Обґрунтування засобів розробки

Особливістю програмних засобів візуалізації об'єктів є підвищене використання обчислювальних ресурсів. Для кожної точки зображення необхідно здійснити складні розрахунки. Тому засоби реалізації програми порівняння моделей відбиття повинні забезпечувати швидке виконання коду.

Іншим аспектом візуалізації об'єктів є значне використання ресурсів пам'яті. Тому важливо зменшити число помилок програміста при використанні пам'яті. Це можливо за рахунок автоматизації процесу роботи з пам'яттю, наявності більш зручного синтаксису програмної мови.

Розглянемо переваги та недоліки програмних мов C#, C++, Python.

C# – компільована високорівнева мова програмування від Microsoft, спроектована Андерсом Гейлсбергом. До переваг C# належать [37]: швидке виконання коду, підтримка об'єктно-орієнтованої парадигми, автоматичне управління пам'яттю, легкість вивчення, підвищена безпека, підтримка технології Windows Forms, забезпечення швидшої розробки, простота синтаксису, наявність докладних онлайн-ресурсів (Microsoft Learn). Основним недоліком є відносно менша гнучкість.

C++ – компільована середньорівнева мова програмування, створена Б'ярном Страstrupом. До переваг мови відносяться [38]: висока швидкість виконання коду, підтримка об'єктно-орієнтованого програмування, можливість ручного управління пам'яттю досвідченим програмістом, можливість роботи з кодом на низькому рівні. До недоліків мови належать: відсутність засобу автоматичного збору сміття, більш складний синтаксис серед розглянутих мов, проблеми з безпекою.

Python – інтерпретована високорівнева мова програмування, створена Гвідо ван Россумом. До переваг відносяться [39]: легкість вивчення й використання, гнучкість, велика кількість бібліотек. До недоліків відносяться: низька швидкість виконання коду, проблеми з дизайном синтаксису, несумісність версій, велике використання пам'яті.

Порівнюємо мови програмування у таблиці 4.1 за критеріями: швидке виконання коду, наявність засобу автоматичного управління збіркою сміття, наявність докладної документації, підтримка форм графічного інтерфейсу.

Таблиця 4.1 –Порівняння мов програмування

Характеристика /Мова	C#	C++	Python
Швидке виконання коду	+	+	-
Наявність докладної документації	+	+	+
Підтримка форм графічного інтерфейсу	+	+	+
Наявність засобу автоматичного управління збіркою сміття	+	-	+
Сума	4	3	3

Для розробки програмного засобу порівняння ДФВЗ обрано мову C#, враховуючи відповідність усім розглянутим критеріям порівняння.

Середовищем розробки програми обрано MS Visual Studio, оскільки при виборі програмної мови, що підтримується Microsoft, доцільне використання середовища від Microsoft.

Отже, засобами розробки засобу порівняння ДФВЗ обрано C# і MS Visual Studio.

#### 4.2 Розробка інтерфейсу програмного засобу

Інтерфейс програмного засобу для порівняння ДФВЗ повинен забезпечувати зручне управління візуалізацією об'єктів на основі обраних користувачем моделей відбиття.

Модель інтерфейсу зображена на рисунку 4.1.

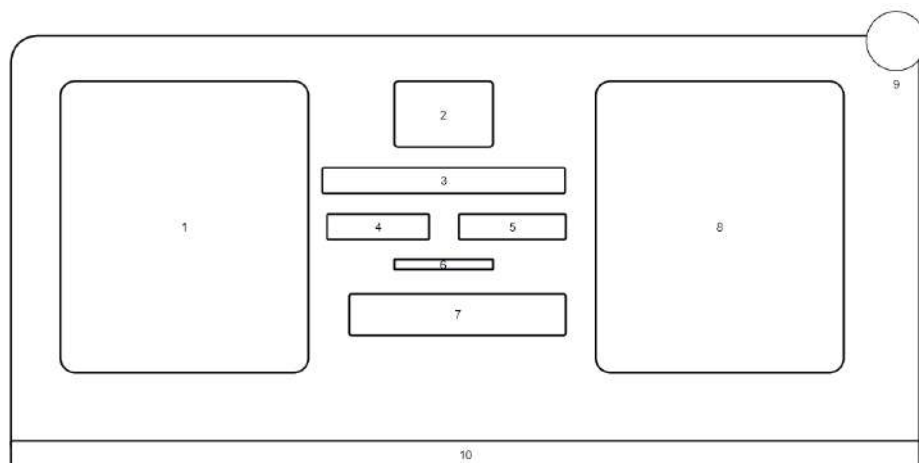


Рисунок 4.1 – Схематична модель інтерфейсу програмного засобу порівняння ДФВЗ

Позначення на рисунку 4.1 означають:

1. Вікно візуалізації об'єкта для першої обраної зі списку користувачем ДФВЗ. У вікні візуалізації виводиться зображення кулі з відблиском.

2. Поля для вибору значень коефіцієнтів: відбиття розсіяного світла, спекулярного відбиття, дифузного відбиття, спекулярності поверхні.

3. Поле для введення формули ДФВЗ.
4. Список для вибору першої моделі ДФВЗ.
5. Список для вибору другої моделі ДФВЗ.
6. Поле, де виводиться манхеттенська відстань між двома зображеннями.
7. Кнопка запуску процесу візуалізації об'єктів.
8. Вікно візуалізації об'єкта для другої обраної зі списку користувачем ДФВЗ.
9. Кнопка закриття програми.
10. Рядок стану програми.

Наявність в інтерфейсі програми двох вікон візуалізації з полем відстані між зображеннями забезпечує полегшення порівняння ДФВЗ.

### 4.3 Розробка програмного засобу

Основні класи програмного засобу, що відображають різні ДФВЗ, представлено на рисунку 4.2.

UML моделі розділу розроблені у Visual Paradigm .

Клас BRDF є загальним класом-предком для класів конкретних ДФВЗ. Вміщує поля: масив типу `double` `sposterigach` представляє координати спостерігача об'єкта, змінні (тип `double`) `KoefSpec`, `KoefDiff`, `KoefAmb` представляють коефіцієнти спекулярного й дифузного відбиття, відбиття розсіяного світла відповідно, змінна (тип `int`) `n_val` – коефіцієнт спекулярності поверхні, змінна (тип `string`) `formula` – вираз для заданої користувачем ДФВЗ, `Dzhereło` (кортеж з масиву `double` `Dzpos` і змінної `Color IntPoint`) – координати джерела світла та інтенсивність джерела світла, `Tochka` (масив `double`) – координати точки, на яку падає світло, `Normal` (масив `double`) – нормаль до точки поверхні.

Методами класу є: BRDF – конструктор класу (до параметрів належать koefSpec, koefDiff, koefAmb, n\_val\_znach), Specular відповідає за обчислення значення ДФВЗ (параметри – n\_val і kut (значення косинуса кута типу double)), GetLVector відповідає за розрахунок вектора між джерелом і точкою (параметри – масив координат точки tochka та Dzpos), GetVVector – за розрахунок вектора між спостерігачем і точкою (параметри – sposterigach, tochka), GetHVector (параметри – масиви значень векторів L, V) – за розрахунок суми векторів падіння й відбиття, PixelIntensity – за розрахунок інтенсивності кольору у точці (параметри – tochka, масив double нормалі normal, змінна для представлення формули типу string customeq), KorygKolor – за коригування значень інтенсивності кольору при виході за можливі значення (параметр d – значення інтенсивності).

Класами-нащадками класу BRDF є PhongBRDF, SchlickBRDF, NewBRDF, CustomBRDF, що відповідно подають ДФВЗ Фонга, Шліка, модифікованої ДФВЗ Шліка, ДФВЗ на основі формули користувача. Класи-нащадки містять конструктори та власні реалізації методу Specular.

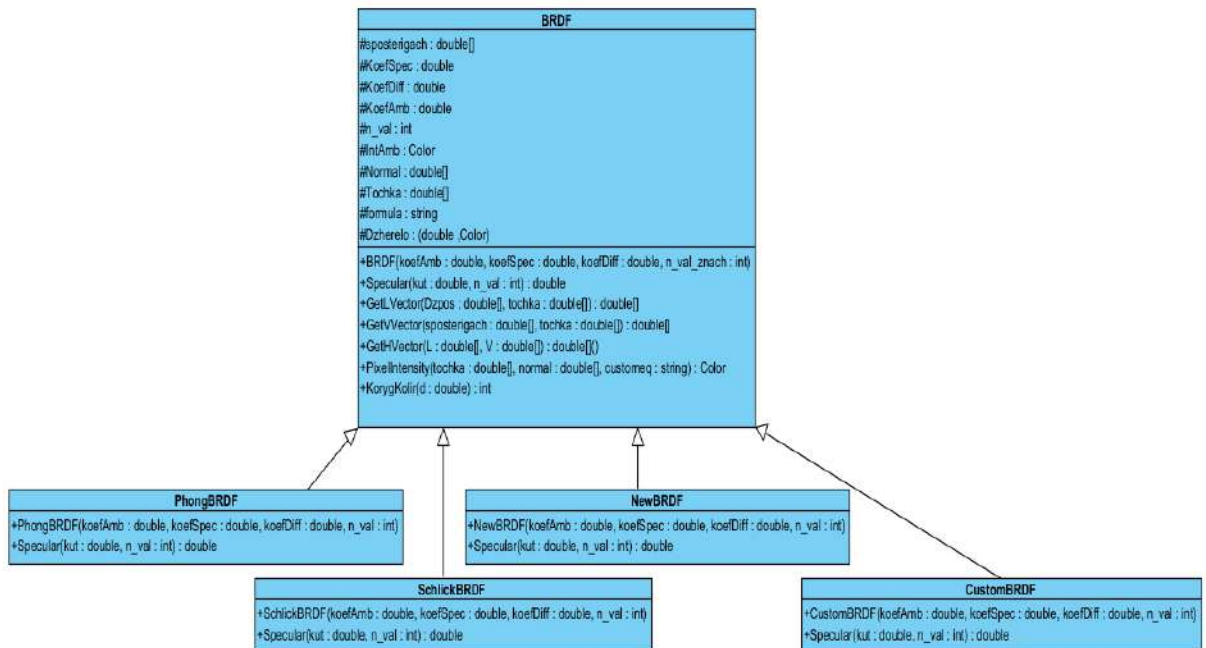


Рисунок 4.2 – Діаграма основних класів засобу порівняння ДФВЗ

Для представлення точок об'єктів використовується клас *Tochka* (рисунок 4.3).

Полями класу є: *d3spherecoord* (масив типів *double*) представляє координати точки відносно центра кулі (сфери), *d2imagecoord* (масив *double*) представляє координати точки відносно користувача програми (у 2 вимірах відповідає координатам вікна візуалізації), *d3scenecoord* (масив *double*) – світові координати точки у сцені, *Normal* (масив *double*) – нормаль до точки, *tempmatrix* (двовимірний масив *double*) – матрицю для обчислення значення нормалі.

Також наявний конструктор класу *Tochka*, що приймає як аргументи змінні типу *double* *x*, *y*, *z* і присвоює їх масиву *d3spherecoord*.

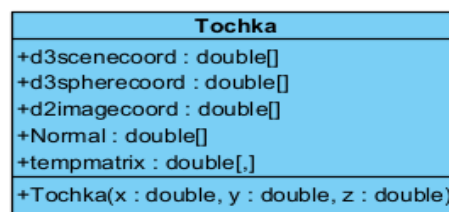


Рисунок 4.3 – Діаграма класу *Tochka*

Клас *RGBval* використовується для роботи з трьома каналами колірної моделі *RGB* у вигляді одної сутності (рисунок 4.4). Полями класу є змінні типу *double* *red*, *green*, *blue*, що відповідно представляють червоний, зелений, синій кольори.

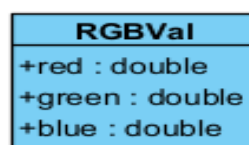


Рисунок 4.4 – Діаграма класу *RGBval*

Для реалізації математичних операцій, що використовуються для формування зображення, розроблено клас MathOper (рисунок 4.5).

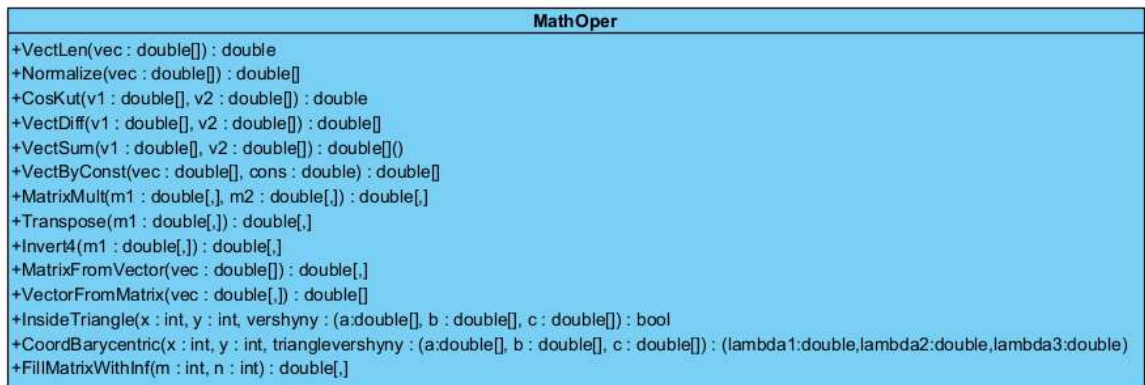


Рисунок 4.5 – Діаграма класу MathOper

Клас містить методи: VectLen (до параметрів належить масив значень вектора *vec* типу *double*) відповідає за знаходження довжини вектора, Normalize (параметр – масив *vec*) відповідає за нормалізацію вектора, CosKut (параметри – масиви значень векторів типу *double* *v1*, *v2*) – за обчислення косинуса кута між векторами, VectDiff (параметри – масиви *v1*, *v2*) – за обчислення різниці між векторами, VectSum (параметри – масиви *v1*, *v2*) – за обчислення суми векторів, VectByConst (параметри – константа типу *double* *cons*, масив *vec*) – за множення вектора на число, MatrixFromVector (параметр – масив *vec*) – за перетворення одновимірного масиву у матрицю, VectorFromMatrix (параметр – двовимірний масив *vec*) – за перетворення двовимірної матриці (де одна зі сторін 1) у одновимірний масив, MatrixMult (параметри – двовимірні масиви типу *double* *m1*, *m2*) – за множення двох матриць, Invert4 (параметр – масив *m1*) – за формування оберненої матриці розміром 4\*4, FillMatrixWithInf (параметри – розміри матриці *m*, *n* типу *int*) – за ініціалізацію матриці нескінченними значеннями, InsideTriangle (параметри – змінні типу *int* *x*, *y* (координати у вікні візуалізації), кортеж *vershynu* з координатами вершин трикутника) – за визначення належності точки до трикутника, CoordBarycentric (параметри –



змінні  $x$ ,  $y$ , кортеж координат вершин  $\text{trianglevershyny}$ ) – за обчислення барицентричних координат точки у трикутнику.

Загальний алгоритм розробленої програми полягає у формуванні зображення у першому вікні (функція  $\text{DrawWindow}$ ), формуванні зображення у другому вікні (функція  $\text{DrawWindow2}$ ), обчисленні похибки між двома зображеннями (функція  $\text{PoHybka}$ ) (рисунк 4.6).

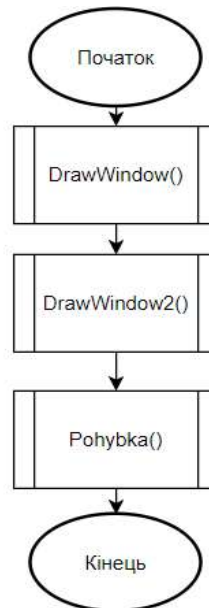


Рисунок 4.6 – Загальна блок-схема роботи програми

Для порівняння ДФВЗ у вікнах програми формуються зображення куль. Координати сфери у програмі обчислюються за формулами

$$x = (r * \sin(j * \Delta y) * \cos(i * \Delta x)) \quad (4.1)$$

$$y = (r * \cos(j * \Delta y)) \quad (4.2)$$

$$z = (r * \sin(j * \Delta y) * \sin(i * \Delta x)), \quad (4.3)$$

де  $r = \sqrt{(x^2 + y^2 + z^2)}$ ,  $i$ ,  $j$  – номери ітерацій в подвійному циклі,  $\Delta y$  та  $\Delta x$  -

$\pi / 20$ .

Обчислені координати  $x$ ,  $y$ ,  $z$  використовуються при ініціалізації екземпляру класу `Tochka`. Через функції `ToSceneCoord`, `To2DCoord` обчислюються координати у сцені та відносно спостерігача для точки. Екземпляр класу `Tochka` записується у масив `mas`. Блок-схему формування точок кулі зображено на рисунку 4.7.

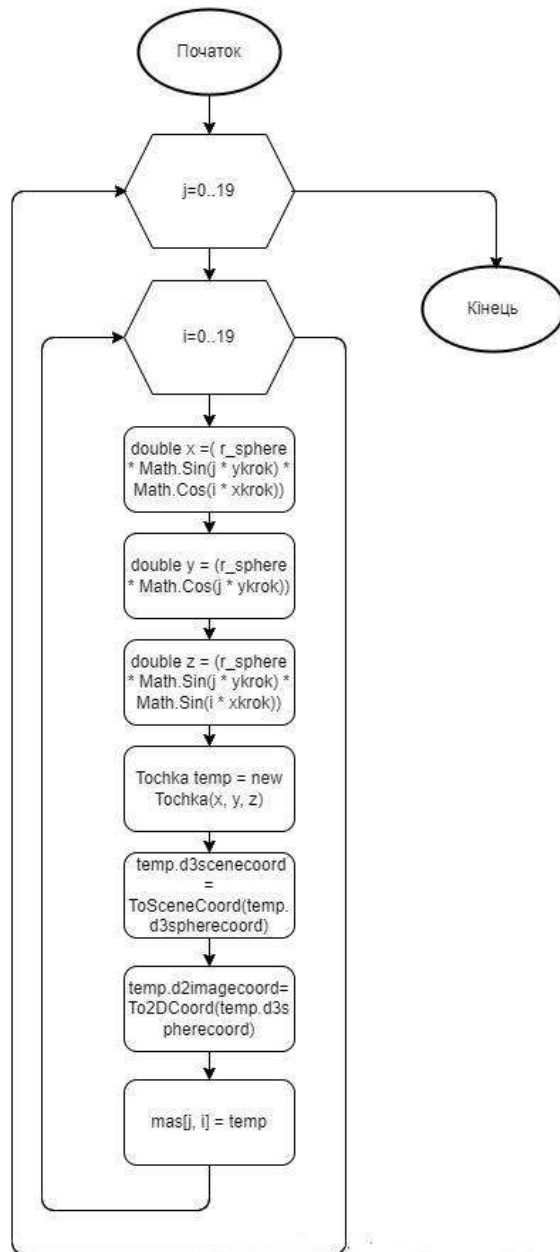


Рисунок 4.7 – Блок-схема обчислення точок кулі

Розіб'ємо поверхню сформованої кулі на трикутники. Формування набору вершин трикутників здійснюється у подвійному циклі. Точки вершин кожного трикутника вибираються з масиву точок кулі `mas` відповідно до значень змінних ітерування `i`, `j`. На кожній ітерації трикутники формуються ліворуч і праворуч. Сформовані трикутники на кулі зображено на рисунку 4.8. Розраховані точки додаються у список масивів `Трикутнику` типу `Точка`. Блок-схема розрахунку вершин трикутників зображена на рисунку 4.9.

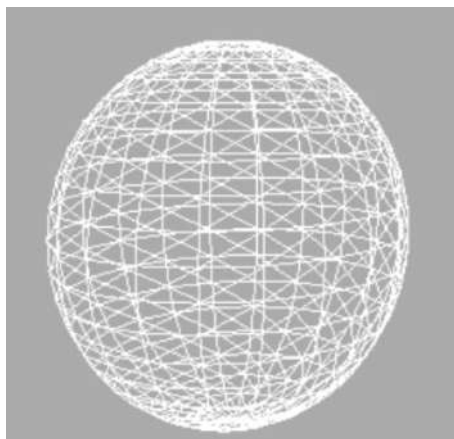


Рисунок 4.8 – Сформовані трикутники на поверхні кулі

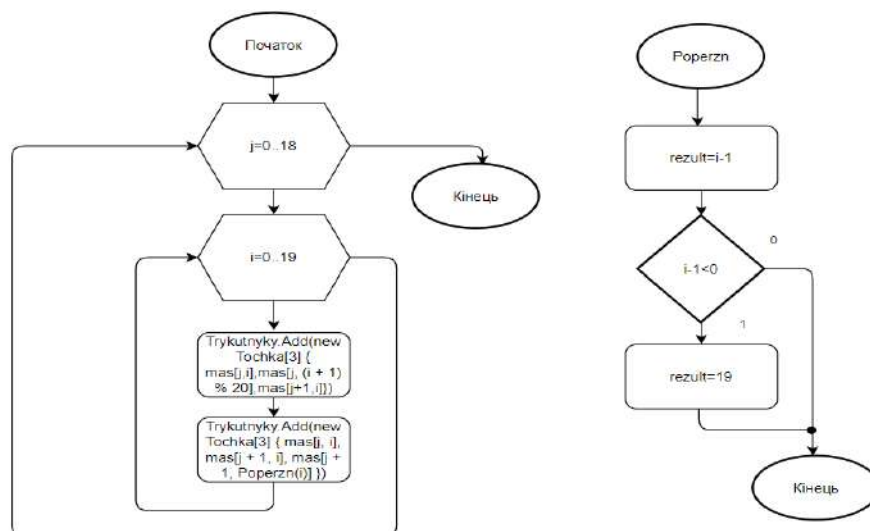


Рисунок 4.9 – Блок-схема функції розрахунку вершин трикутників, підпрограми `Popern`

Для кожного отриманого трикутника на поверхні кулі здійснюється зафарбовування точок.

На вхід функції зафарбовування трикутника передається масив вершин `tochku`. Використовуються координати відносно користувача.

У список `x` додаються значення координат вершин трикутника по осі `x`, `y` – по `y`. Змінна `xма` зберігає максимальне значення у списку `x`, `xмі` – мінімальне. Аналогічно змінна `ума` зберігає максимальне значення у списку `y`, `умі` – мінімальне.

За допомогою двовимірного циклу перебираються точки в межах `xмі`, `xма`, `умі`, `ума`. Через функцію `InsideTriangle` перевіряється, чи належить точка трикутнику.

Через функцію `CoordBarycentric` обчислюються барицентричні координати точки відносно вершин трикутника згідно з формулами

$$\lambda_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)} \quad (4.4)$$

$$\lambda_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)} \quad (4.5)$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 \quad (4.6)$$

Через функцію `ZBuffer` методом `Z`-буфера перевіряється, чи не є дана точка невидимою та чи потрібно її зафарбовувати.

Викликається функція `DotInPhongShading`, у якій спершу інтерполюється нормаль точки відповідно до барицентричних координат, далі нормаль використовується для обчислення інтенсивності кольору точки.

Функція `SetColor` встановлює значення пікселя для координати відносно користувача.

Блок-схема зафарбовування всередині трикутника зображена на рисунку 4.10.

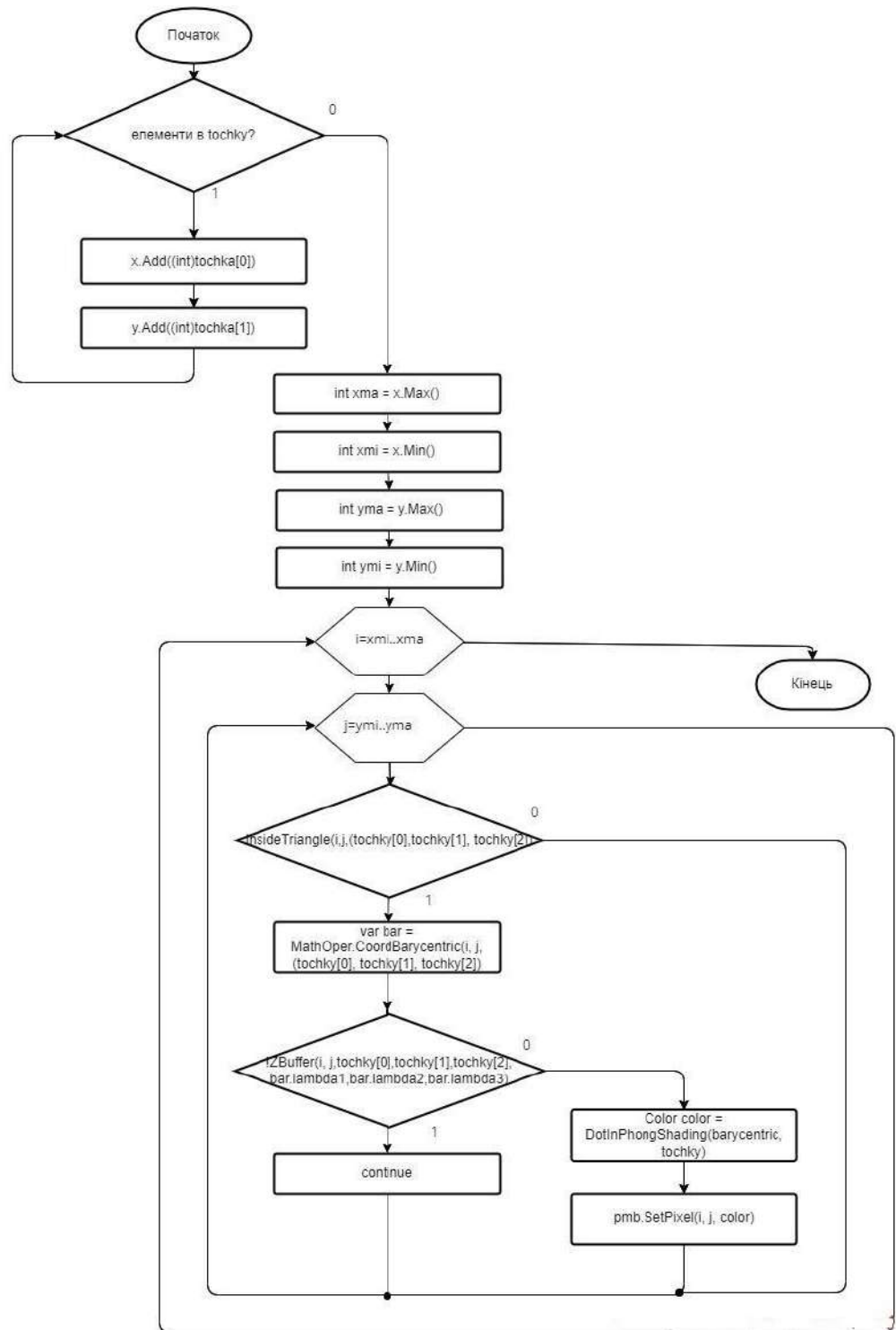


Рисунок 4.10 – Блок-схема зафарбовування всередині трикутника

Змінна `values` типу `RGBval` використовується для визначення інтенсивності кольору по каналах: червоний (поле змінної `red`), зелений (поле `green`), блакитний (поле `blue`). Для кожного каналу коефіцієнт відбиття

розсіяного світла ( $KoefAmb$ ) множиться на поканальне значення інтенсивності розсіяного світла.

Через функції `GetVVector`, `GetLVector` отримуються координати векторів відбиття й падіння відповідно.

Через функцію `CosKut` обчислюється косинус кута між нормаллю та вектором падіння. Отримане значення множиться на коефіцієнт дифузного відбиття `CoefDiff`. Отримуємо значення дифузної складової (змінна `difpart`).

У змінній `temppar` зберігається інформація для обчислення вектора дзеркального відбиття, що представляється масивом `R1`.

Через функцію `CosKut` обчислюється косинус кута між векторами відбиття й дзеркального відбиття.

У змінній `specpart` зберігається значення спекулярної складової відбиття. Функція `Specular` повертає значення емпіричної двопроменевої функції відбивної здатності на основі косинуса (залежно від вибору користувача). Дане значення множиться на коефіцієнт спекулярного відбиття ( $KoefSpec$ ).

Інтенсивність джерела світла поканально множиться на значення дифузної і спекулярної складової, добутки додаються.

В результаті, при поєднанні розсіяної, дифузної, спекулярної складової формуються кінцеві значення інтенсивності кольору по каналах.

Блок-схему алгоритму знаходження інтенсивності кольору точки зображено на рисунку 4.11.

Після зафарбовування куль на двох зображеннях здійснюється порівняння зображень через середню манхеттенську відстань. Манхеттенська відстань між зображеннями полягає у знаходженні суми абсолютних різниць між двома наборами значень.

Створюємо екземпляри типу `Bitmap bit1`, `bit2`, куди записуємо інформацію з вікон відображення куль. Для запису інформації використовується функція `DrawToBitmap`.

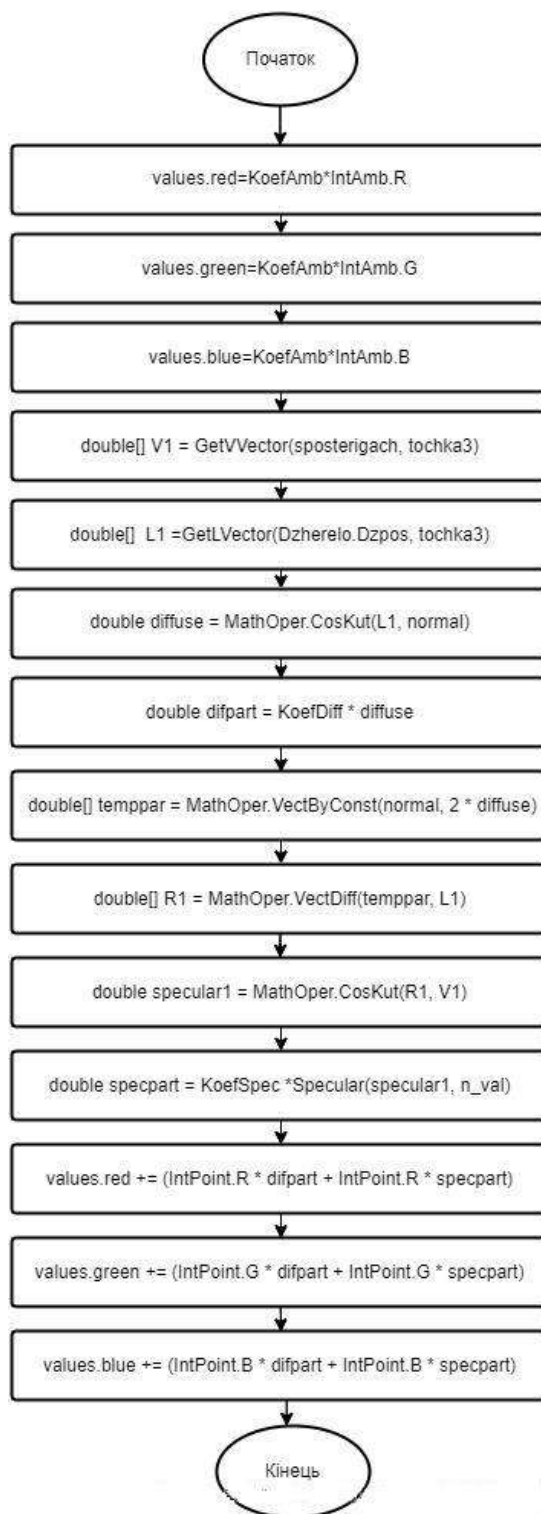


Рисунок 4.11 – Блок-схема обчислення інтенсивності кольору точки

У масиви mas1, mas2 записуються інтенсивності точок двох зображень. Інтенсивності кольору представляються одним каналом сірих відтінків, обчисленим функцією ImageValues.

Між масивами `mas1`, `mas2` обчислюється манхеттенська відстань. Для знаходження модуля різниці використовується функція `Math.Abs`.

Далі сумарна відстань ділиться на число точок у зображенні. Блок-схема порівняння зображень об'єктів наведена на рисунку 4.12.

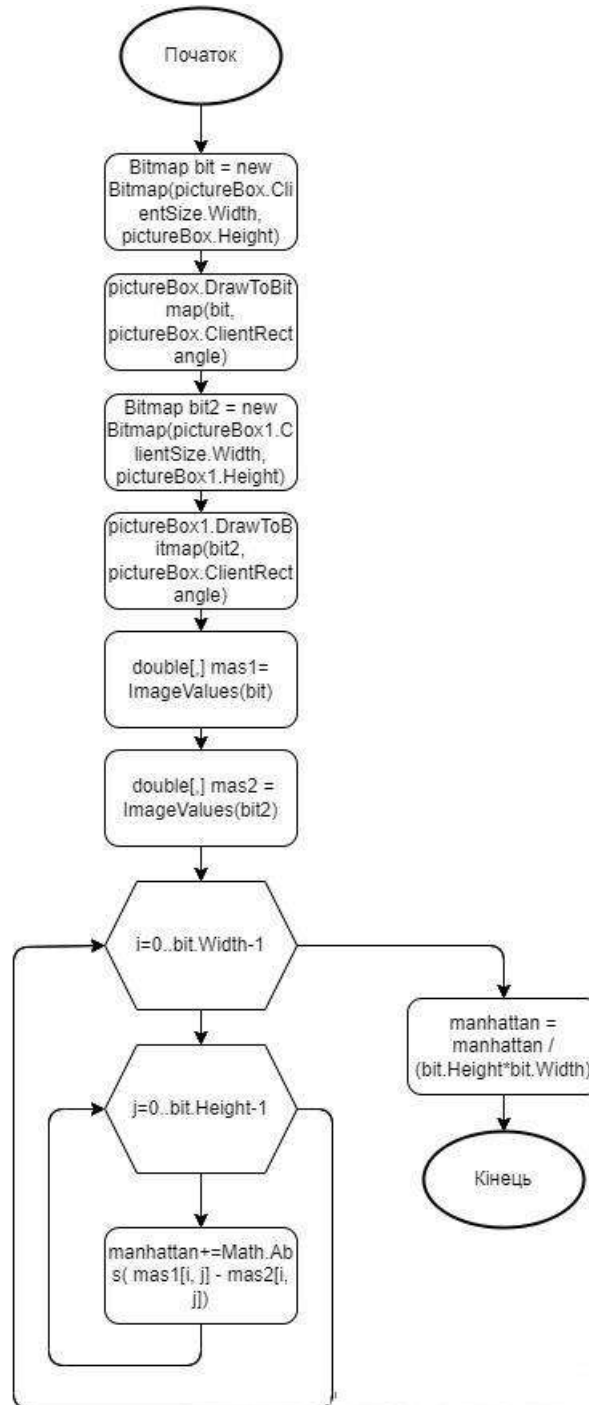


Рисунок 4.12 – Блок-схема порівняння зображень об'єктів



Для порівняння ДФВЗ наявна можливість вводу власної формули, окрім використання вбудованих моделей Фонга, Шліка, модифікованої моделі Шліка.

Користувацька формула обмежується списком можливих операндів: коефіцієнт спекулярності, косинуси кута між векторами відбиття й дзеркального відбиття, нормаллю та сумою векторів відбиття, падіння, нормаллю та вектором падіння, нормаллю та вектором відбиття.

Можливі операнди для формули зберігаються у словнику `slovnuk`.

Для оцінювання вмісту введеної формули використовується бібліотека `NCalc`. Вираз `NCalc` ініціалізується. Для оцінки вмісту введеного виразу відносно можливих операндів використовується функція `EvaluateParameter`. Значення виразу обчислюється функцією `Evaluate`.

Алгоритм оцінювання вмісту введеної користувачем функції зображено на рисунку 4.13.

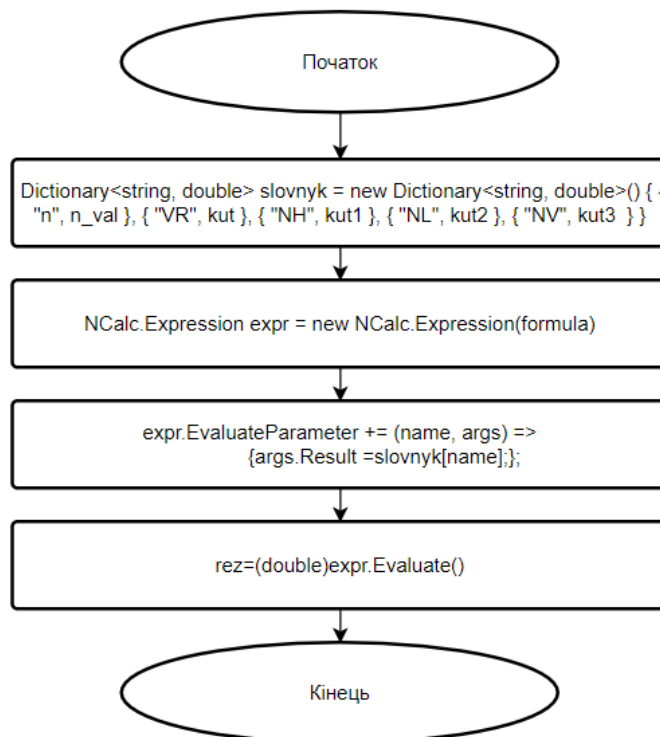


Рисунок 4.13 – Блок-схема оцінювання вмісту формули

Отже, у даному підрозділі розглянуто моделі та алгоритми розробленого програмного засобу порівняння ДФВЗ. Лістинг наведено у додатку Г.

#### 4.4 Тестування розроблених ДФВЗ у програмних засобах

Здійснимо тестування модифікованої ДФВЗ Шліка (див. формулу (2.3)) у розробленому засобі. У лівому списку для вибору вибираємо модель Фонга, у правому модель Шліка. Встановлюємо коефіцієнт спекулярності поверхні 50. Отримуємо порівняльну візуалізацію у двох вікнах, обчислена середня манхеттенська відстань – 0.41 (рисунок 4.14).

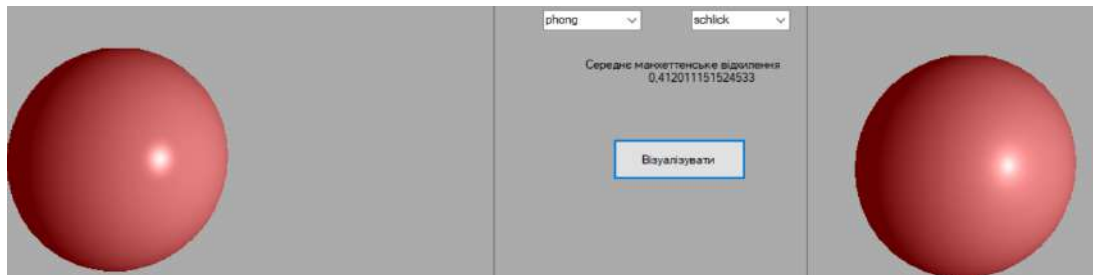


Рисунок 4.14 – Порівняння застосування ДФВЗ Фонга, Шліка у розробленому засобі

У правому списку обираємо «new», що відповідає модифікованій ДФВЗ Шліка. Середня манхеттенська відстань – 0.07 (рисунок 4.15).

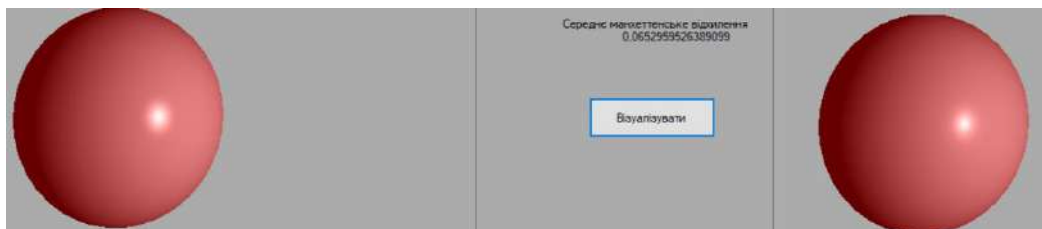


Рисунок 4.15 – Порівняння застосування ДФВЗ Фонга та модифікованої ДФВЗ Шліка

Протестуємо можливість вводу користувацької формули. Введемо у відповідне поле програми формулу ДФВЗ Фонга у виді «Pow(VR,n)». У лівому списку вибираємо «custom», що відповідає режиму вводу формули. Отримуємо (рисунок 4.16) аналогічний результат (див. рис. 4.15).



Рисунок 4.16 – Порівняння застосування ДФВЗ Фонга та модифікованої ДФВЗ Шліка у режимі вводу формули

У таблиці 4.2 зображено таблицю отриманих значень середньої манхеттенської відстані між зображеннями на основі ДФВЗ Фонга та ДФВЗ Шліка, модифікованою ДФВЗ Шліка залежно від коефіцієнта спекулярності поверхні.

Таблиця 4.2 – Середні манхеттенські відстані між зображеннями

Коефіцієнт спекулярності поверхні	Манхеттенська відстань між зображеннями на основі ДФВЗ Фонга, ДФВЗ Шліка	Манхеттенська відстань між зображеннями на основі ДФВЗ Фонга, модифікованої ДФВЗ Шліка
5	1.22	0.65
10	1.24	0.52

## Продовження таблиці 4.2

Коефіцієнт спекулярності поверхні	Манхеттенська відстань між зображеннями на основі ДФВЗ Фонга, ДФВЗ Шліка	Манхеттенська відстань між зображеннями на основі ДФВЗ Фонга, модифікованої ДФВЗ Шліка
15	0.77	0.19
20	0.71	0.17
50	0.41	0.07
100	0.26	0.04
250	0.14	0.02
500	0.08	0.01

Здійснимо тестування модифікованої ДФВЗ Шліка у онлайн-шейдері OneShader.

Використаємо готову візуалізацію використання ДФВЗ Блінна. Допишемо мовою GLSL код для ДФВЗ Шліка, модифікованої ДФВЗ Шліка (рисунок 4.17). Результат обчислення ДФВЗ Блінна зберігається у змінній `spec`, ДФВЗ Шліка – `specschlick`, модифікованої ДФВЗ Шліка – `specnew`.

```
float spec = pow(max(dot(N, N), 0.0), shininess);
float specschlick= (max(dot(N, H), 0.0)) / (n-n*max(dot(N, H), 0.0)+max(dot(N, H), 0.0));
float specnew= (2.0*max(dot(N, H), 0.0)) /
(1.25*pow((n-n*max(dot(N, H), 0.0)+1.25*max(dot(N, H), 0.0)), 2.0));
```

Рисунок 4.17 – Дописаний код для представлення ДФВЗ Блінна, Шліка, модифікованої ДФВЗ Шліка у OneShader

Здійснимо візуалізацію кулі для коефіцієнтів спекулярності поверхні  $n$  10, 20, 50, 200, 500. Порівняємо отримані зображення на основі ДФВЗ Блінна, Шліка та модифікованої ДФВЗ Шліка. На рисунку 4.18 зображено колаж візуалізованих куль відповідно до ДФВЗ і  $n$ .

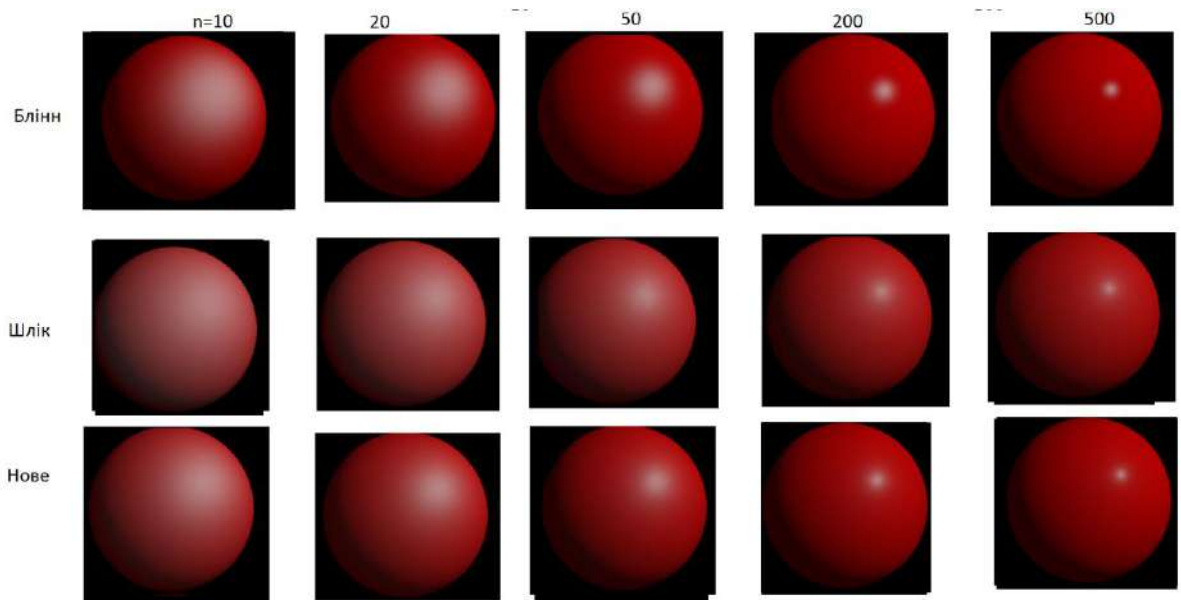


Рисунок 4.18 – Порівняння візуалізації куль залежно від ДФВЗ,  $n$  у OneShader

Отже, модифікована ДФВЗ Шліка забезпечує більш реалістичне відображення відблисків відносно ДФВЗ Блінна.

Здійснимо тестування у онлайн-шейдері Shadertoy.

Використаємо готове рішення для візуалізації лампи. Допишемо код (рисунок 4.19).

При тестуванні у Shadertoy назви змінних результатів обчислень ДФВЗ аналогічні відносно тестування у OneShader.

```
float nval=200.0,
//float spec = pow ( hn, nval );
//float specschlick=hn/(nval-nval*hn+hn);
float specnew=2.0*hn/(1.25*pow(nval-nval*hn+1.25*hn,2.0));
```

Рисунок 4.19 – Дописаний код для представлення ДФВЗ Блінна, Шліка, модифікованої ДФВЗ Шліка у Shadertoy (до зображення лампи)

На рисунку 4.20 зображено порівняльний колаж зображень лампи для  $n$  25, 50, 200.

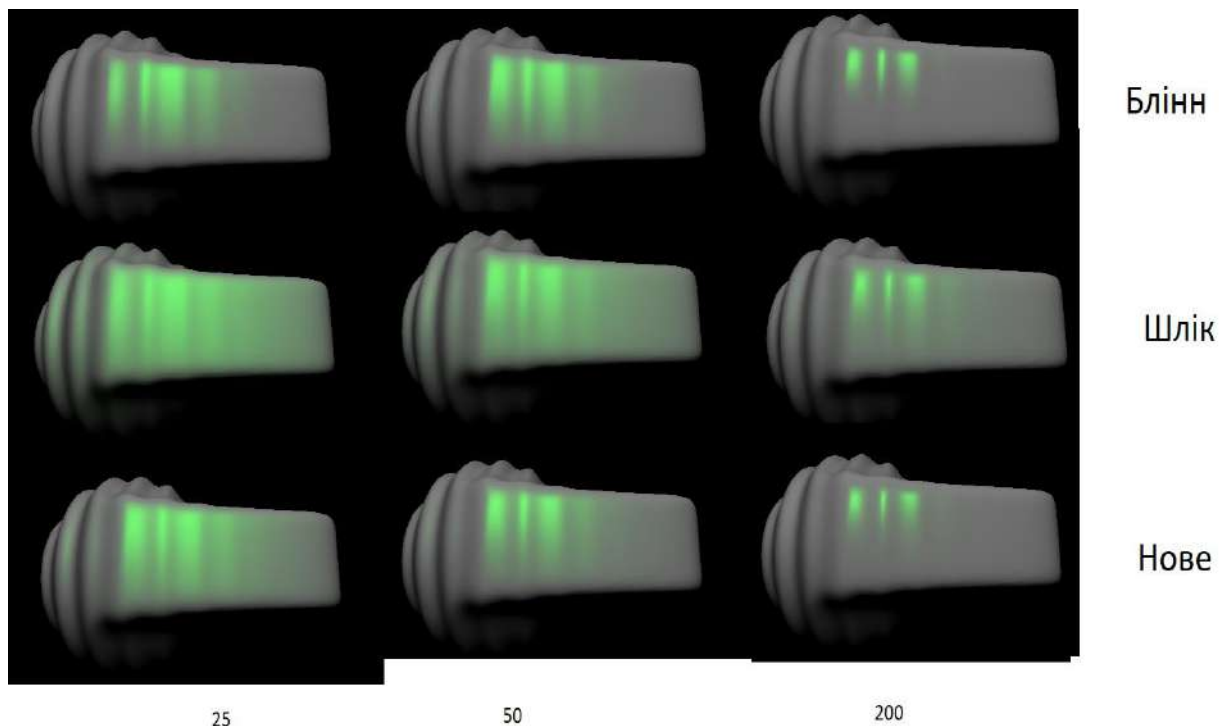


Рисунок 4.20 – Порівняння візуалізації ламп залежно від ДФВЗ,  $n$  у Shadertoy

Оберемо для тестування візуалізацію тора. Допишемо код для ДФВЗ, що порівнюються (рисунок 4.21).

```
//vec3 spec = 0.9*vec3(1)*pow(dot(H,N),200.0);
//vec3 specschlick = 0.9*vec3(1)*(dot(H,N)/(200.0-dot(H,N)*200.0+dot(H,N)));
vec3 specnew = 0.9*vec3(1)*(2.0*dot(H,N)/(1.25*pow(200.0-dot(H,N)*200.0+1.25*dot(H,N),2.0) ));
```

Рисунок 4.21 – Дописаний код для представлення ДФВЗ Блінна, Шліка, модифікованої ДФВЗ Шліка у Shadertoy (до зображення тора)

На рисунку 4.22 зображено колаж зображень тора відносно ДФВЗ і  $n$ .

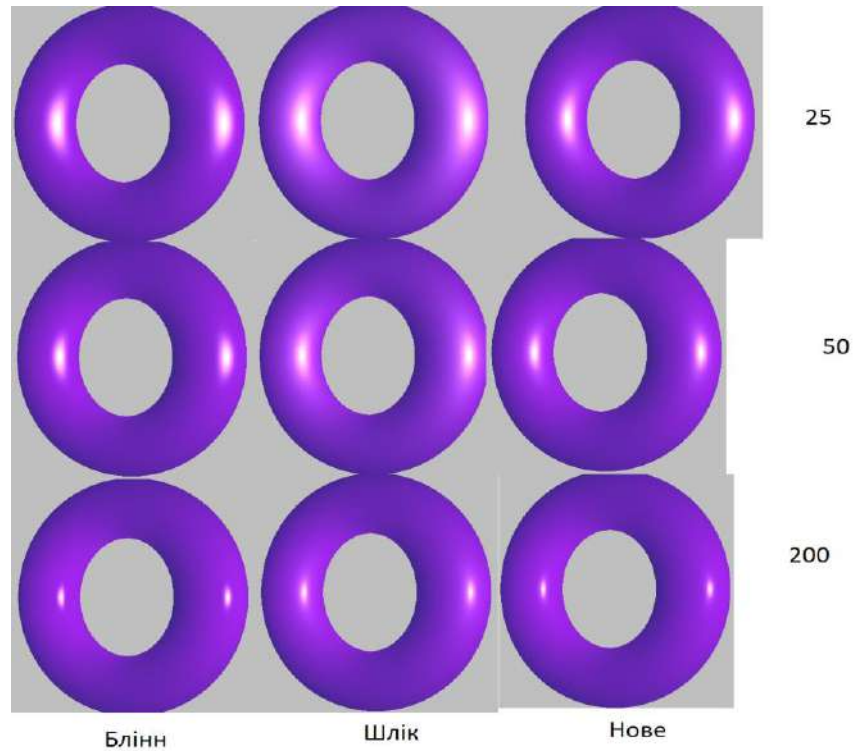


Рисунок 4.22 - Порівняння візуалізації тора залежно від ДФВЗ,  $n$  у Shadertoy

Здійснимо тестування ДФВЗ на основі двох утворюючих функцій (див. формулу (2.26)) у розробленому засобі. Коефіцієнт спекулярності поверхні встановлюємо як 50. Середня манхеттенська відстань— 0.009 (рисунок 4.23):

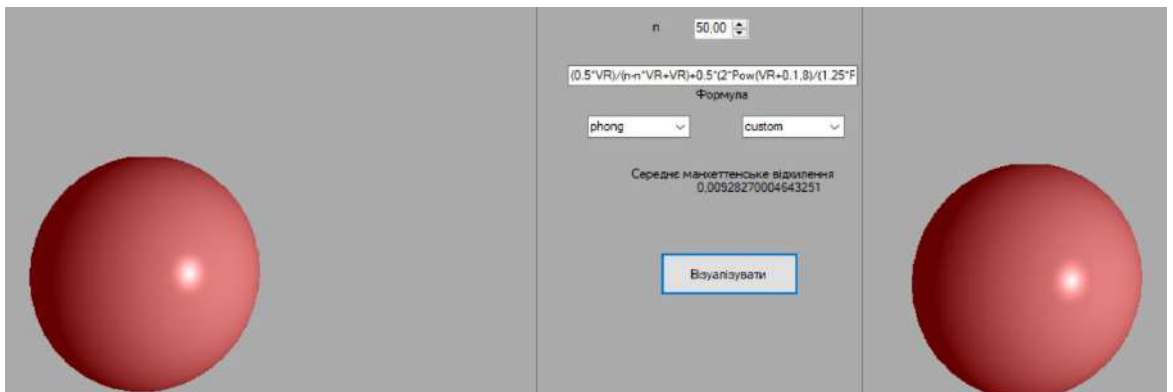


Рисунок 4.23 – Порівняння застосування ДФВЗ Фонга та ДФВЗ на основі утворюючих функцій

Здійснимо тестування модифікованої косинус-квадратичної функції (див. формулу (3.10)). Коефіцієнт спекулярності поверхні встановлюємо як 50. Отримуємо значення середньої манхеттенської відстані між зображеннями – 0.19 (рисунок 4.24).

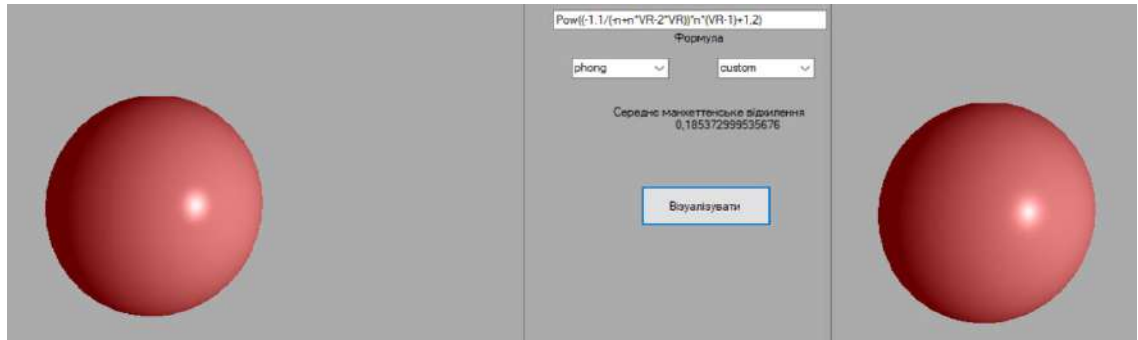


Рисунок 4.24 – Порівняння застосування ДФВЗ Фонга та модифікованої косинус-квадратичної ДФВЗ

Отже, показано переваги у точності модифікованої ДФВЗ Шліка над оригінальною ДФВЗ у зонах формування блюмінгу та затухання. Показано високу точність модифікованої косинус-квадратичної ДФВЗ, ДФВЗ на основі функції Шліка й оновленої функції Шліка при відтворенні відблисків зображення.

Розроблений засіб забезпечує виведення зображень об'єктів у двох вікнах поряд, що полегшує візуальне порівняння точності представлення відблисків різними ДФВЗ.

#### 4.5 Висновки

Описано розробку програмного засобу порівняння моделей ДФВЗ, тестування розроблених ДФВЗ.

Обґрунтовано розробку програмного засобу за допомогою мови C#.



Розроблено схему інтерфейсу програмного засобу для порівняння ДФВЗ.

Для розробки програмного засобу описано діаграми класів, алгоритми обчислення точок кулі та трикутників, зафарбовування всередині трикутника, обчислення інтенсивності кольору у точці, порівняння зображень, оцінки введеної формули.

Проведено тестування розроблених ДФВЗ у розробленому засобі та OneShader, Shadertoy. Показано переваги розроблених ДФВЗ у точності відтворення об'єктів. Перевагами розробленого засобу над OneShader, Shadertoy є двовіконне формування зображень для можливості порівняння моделей, виведення манхеттенської відстані між зображеннями.

## 5 ЕКОНОМІЧНА ЧАСТИНА

### 5.1 Оцінювання комерційного потенціалу розробки

Для визначення можливостей та перспектив комерціалізації наукової розробки за темою «Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки» необхідне здійснення оцінювання її комерційного потенціалу.

Для оцінки комерційного потенціалу доцільним є здійснення комерційно-технологічного аудиту, що полягає в опитуванні групи експертів щодо списку критеріїв.

При оцінюванні аспектів комерційного потенціалу наукової розробки були залучені експерти з кафедри програмного забезпечення Вінницького національного технічного університету: експерт 1 (к.т.н., доц. каф. ПЗ Майданюк В.П.), експерт 2 (к.т.н., доц. каф. ПЗ Войтко В.В.), експерт 3 (к.т.н., доц. каф. ПЗ Романюк О.В.).

У таблиці 5.1 надано результати опитування експертів щодо комерційного потенціалу наукової розробки.

Критеріями оцінювання є: технічна здійсненність концепції, ринкові переваги (наявність аналогів), ринкові переваги (ціна продукту), ринкові переваги (технічні властивості), ринкові переваги (експлуатаційні витрати), ринкові перспективи (розмір ринку), ринкові перспективи (конкуренція), практична здійсненність (наявність фахівців), практична здійсненність (наявність фінансів), практична здійсненність (потреба нових матеріалів), практична здійсненність (термін реалізації), практична здійсненність (розробка документів) [40].

Кожен параметр оцінений експертами за п'ятибальною шкалою – від 0 до 4.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу наукової розробки

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	4	4	3
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	4	4	4
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	4	3	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	3
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	42	41	40
Середньоарифметична сума балів $CB_c$	$CB_c = \frac{\sum_{i=1}^3 CB_i}{3} = 41$		

Результат оцінювання відповідає проміжку 41...48 та означає [40], що для наукової розробки за темою «Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки» рівень комерційного потенціалу високий.

До аналогів науково-технічної розробки належать: OneShader, Shadertoy, Redshift, Octane, RenderMan, Cycles (розглянуто у розділі 1).

Аналогом для порівняння було обрано OneShader.

Основними недоліками аналога є: відсутність обчислення похибки між зображеннями на основі різних двопроменевих функцій відбивної здатності, необхідність створення колажу зображень для порівняння результатів візуалізацій об'єктів, необхідність знання користувачем мови програмування шейдерів GLSL.

Перевагами нової розробки відносно аналогу є: наявність двох вікон візуалізації для порівняння зображень на основі двопроменевих функцій відбивної здатності, виведення значення манхеттенської відстані між двома зображеннями, більш високорівневий інтерфейс, швидше освоєння новими користувачами, більша кількість вбудованих моделей відбивної здатності світла.

У таблиці 5.2 наведено порівняння технічних параметрів аналога та науково-технічної розробки.

Параметр «функціональність» означає кількість наявних у розробки функцій з переліку (двовіконна візуалізація зображень, обчислення відстані між зображеннями, збереження зображень у бібліотеку, можливість налаштування коефіцієнтів моделей, вводу формул). Коефіцієнт вагомості параметра становить 0.4.

Параметр «простота використання» означає кількість властивостей з переліку (відсутність необхідності знання програмних мов, простота інтерфейсу, відсутність необхідності встановлення додаткових програм, браузерів). Коефіцієнт вагомості параметра становить 0.3.

Параметр «кількість мовних версій інтерфейсу» означає число мов, на яких доступний контент у засобі. Коефіцієнт вагомості параметра – 0.1.

Параметр «кількість вбудованих моделей відбиття» означає число моделей відбивної здатності світла, на основі яких у засобі можна візуалізувати об'єкт без введення коду та формул. Коефіцієнт вагомості параметра – 0.2.

Таблиця 5.2 – Основні технічні показники аналога і нового програмного продукту

Параметр	Аналог	Нова розробка	Індекс зміни значення параметра	Коефіцієнт вагомості
Функціональність	3	4	1.33	0.4
Простота використання	1	3	3	0.3
Кількість мовних версій інтерфейсу	1	1	1	0.1
Кількість вбудованих моделей відбиття	2	3	1.5	0.2

Груповий параметричний індекс за технічними показниками розраховується за формулою [40]

$$I_{TP} = \sum_{i=1}^n \alpha_i \cdot q_i, \quad (5.1)$$

де  $q_i$  – одиничний параметричний показник,  $\alpha_i$  - вагомість параметричного показника.

Розраховуємо значення групового індекса за технічними показниками

$$I_{TP} = 0.4 \cdot 1.33 + 0.3 \cdot 3 + 0.1 \cdot 1 + 0.2 \cdot 1.5 = 1.8.$$

За технічними показниками розробка переважає аналог у 1.8 разів.

Отже, наукова розробка характеризується високим комерційним потенціалом за рахунок розширення функціональних можливостей (можливість двовіконної візуалізації об'єктів, обчислення відстані між зображеннями) та

підвищення простоти використання (не вимагається знання мов програмування, встановлення сторонніх програм).

Нова розробка може бути використана учнями, вчителями у школах, студентами й викладачами в університетах при вивченні моделей відбиття. Реалізовані у засобі нові моделі можуть бути використані інженерами в системах комп'ютерної графіки для підвищення точності візуалізації. Технічна готовність розробки становить 70%, наявний готовий промисловий зразок.

## 5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Розрахунок витрат для науково-дослідної роботи включає здійснення обчислень по статтях калькуляції [40].

Витрати на основну заробітну плату розраховуються за формулою

$$Z_o = \sum_{i=1}^k \frac{M}{T_p} \cdot t, \quad (5.2)$$

де  $M$  – місячний посадовий оклад дослідника,  $T_p$  – кількість робочих днів у місяці (21),  $t$  – кількість днів роботи дослідника,  $k$  – кількість посад.

Основна заробітна плата дослідників розрахована у таблиці 5.3.

Таблиця 5.3 – Розрахунки основної заробітної плати дослідників

Працівник	Оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
Науковий керівник	10000	476.2	3	1428.6
Інженер- програміст	12000	571.4	63	35998.2
Всього				37426.8

Витрати на додаткову заробітну плату дослідників розраховуються за формулою

$$Z_{\text{доп}} = Z_o \cdot \frac{10\%}{100} \quad (5.3)$$

Додаткова заробітна плата дослідників становить

$$Z_{\text{доп}} = 37426.8 \cdot 0.1 = 3742.7 \text{ грн.}$$

Відрахування на соціальні заходи обчислюються за формулою

$$Z_n = (Z_o + Z_{\text{доп}}) \cdot \frac{22\%}{100} \quad (5.4)$$

Відрахування на соціальні заходи становлять

$$Z_n = (37426.8 + 3742.7) \cdot 0.22 = 9057.3 \text{ грн.}$$

Витрати на комплектуючі, що використовуються при дослідженні науково-технічної розробки, розраховуються за формулою

$$K_B = \sum_{j=1}^n H_j C_j K_j \quad (5.5)$$

де  $H_j$  – кількість комплектуючих виду (шт.),  $C_j$  – покупна ціна комплектуючих виду (грн.),  $K_j$  – коефіцієнт транспортних витрат (1.1).

Витрати на комплектуючі розраховано у таблиці 5.4.

Таблиця 5.4 – Розрахунок витрат на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн.	Сума, грн.
Папір офісний А4 80 г / м <sup>2</sup> 500 аркушів	1	329	361.9
Набір олівців чорнографітних 12 шт.	1	85	93.5
Всього			455.4

Витрати на програмне забезпечення розраховуються за формулою

$$B_{\text{прг}} = \sum_{i=1}^k C_{\text{инрг}} \cdot C_{\text{прг.}i} \cdot K_i, \quad (5.6)$$

де  $C_{\text{инрг}}$  – ціна одиниці програмного засобу (грн),  $C_{\text{прг.}i}$  – кількість придбаних одиниць,  $K_i$  – коефіцієнт врахування інсталяції (1.1).

У таблиці 5.5 розраховано витрати на програмне забезпечення

Таблиця 5.5 – Витрати на програмне забезпечення

Назва	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
MS Office 2021	1	4789	5267.9
Програма набору формул MathType	1	2132.4	2345.6
Засіб моделювання Visual Paradigm	1	1446.1	1590.7
Засіб для дослідження моделей рендерингу Mathcad	1	2516.9	2768.6
Всього			11972.8



Амортизаційні відрахування для обладнання розраховуються через прямолінійний метод згідно з формулою

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.7)$$

де  $Ц_б$  – балансова вартість обладнання, програм, приміщень (грн),  $t_{вик}$  – термін використання обладнання, програм, приміщень (місяців),  $T_в$  – термін корисного використання обладнання, програм, приміщень (років).

Обчислення амортизаційних відрахувань представлені у таблиці 5.6.

Таблиця 5.6 – Амортизаційні відрахування по кожному виду обладнання

Найменування	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук ASUS	24173	5	3	1208.7
Принтер Canon	23845	5	3	1192.3
Всього				2401

Витрати, які не ввійшли до попередніх статей, враховуються в інших витратах. Інші витрати становлять 50% ( $H_{ie}$ ) від основної заробітної плати дослідників, обчислюються за формулою

$$I_в = (3_о) \cdot \frac{H_{ie}}{100\%} \quad (5.8)$$

Інші витрати становлять

$$I_e = 0.5 \cdot 37426.8 = 18713.4 \text{ (грн)}.$$

Сумарні витрати на проведення науково-дослідної роботи обчислюються за формулою [40]

$$B_{\text{заг}} = Z_o + Z_{\text{дод}} + Z_n + K_e + B_{\text{прг}} + A_{\text{обл}} + I_e \quad (5.9)$$

Отже, сумарні витрати на проведення науково-дослідної роботи

$$B_{\text{заг}} = 37426.8 + 3742.7 + 9057.3 + 455.4 + 11972.8 + 2401 + 18713.4 = 83769.4 \text{ (грн)}.$$

Загальні витрати на завершення науково-технічної роботи обчислюються

$$ЗВ = \frac{B_{\text{заг}}}{\eta}, \quad (5.10)$$

де  $\eta$  – коефіцієнт виконання наукової роботи (0.7).

Загальні витрати на завершення розробки становлять

$$ЗВ = \frac{83769.4}{0.7} = 119670.6 \text{ (грн)}.$$

### **5.3 Прогнозування комерційних ефектів від реалізації результатів розробки**

Чистий прибуток є прибутком, що залишається після врахування витрат на виробництво, амортизації, сплати податків.

Річне збільшення чистого прибутку від реалізації розробки обчислюється за формулою [40]

$$\Delta\Pi_i = (\pm\Delta C_o \cdot N + C_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (5.11)$$

де  $N$  – основний кількісний показник,  $\Delta N$  – зміна  $N$  після впровадження розробки у розглянутому році,  $C_o$  – основний якісний показник (ціна реалізації після впровадження розробки),  $\Delta C_o$  – зміна якісного показника від впровадження розробки у розглянутому році,  $\lambda$  – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість (0.8333),  $\rho$  – коефіцієнт, який враховує рентабельність інноваційного продукту (0.5),  $\vartheta$  – ставка податку на прибуток, який має сплачувати потенційний інвестор (18%).

У даному випадку  $N$  є кількістю користувачів,  $\Delta N$  – зміною кількості користувачів,  $\Delta C_o$  – зміна вартості після впровадження результатів розробки (взято 400 грн),  $C_o$  – вартість програми після впровадження науково-технічної розробки, прийmemo як 1000 грн +  $\Delta C_o$ .

Прогнозується, що протягом 3 років отримуються позитивні результати від впровадження розробки.

Передбачається, що  $\Delta N$  у першому році становить 400, у другому 850, у третьому 1050.  $N$  до реалізації становить 3000.

Обчислимо зміну чистого прибутку (грн) у кожному з цих років згідно з виразами

$$\Delta\Pi_1 = (400 \cdot 3000 + (1000 + 400) \cdot 400) \cdot 0.8333 \cdot 0.5 \cdot (1 - 0.18) = 601309.3 \text{ грн.} \quad (5.12)$$

$$\Delta\Pi_2 = (400 \cdot 3000 + (1000 + 400) \cdot 850) \cdot 0.8333 \cdot 0.5 \cdot (1 - 0.18) = 816550.7 \text{ грн.} \quad (5.13)$$

$$\Delta\Pi_3 = (400 \cdot 3000 + (1000 + 400) \cdot 1050) \cdot 0.8333 \cdot 0.5 \cdot (1 - 0.18) = 912213.5 \text{ грн.} \quad (5.14)$$

#### 5.4 Розрахунок ефективності вкладених інвестицій та періоду їхньої окупності

Приведена вартість збільшення всіх чистих прибутків від впровадження розробки обчислюється за формулою

$$ПП = \sum_{i=1}^T \frac{\Delta\Pi_i}{(1+\tau)^i}, \quad (5.15)$$

де  $T$  – роки, протягом яких очікується отримання позитивних результатів від впровадження та комерціалізації розробки,  $\tau$  – ставка дисконтування (0.15),  $t$  – період від впровадження розробки до отримання інвестором додаткових чистих прибутків у році.

Приведена вартість збільшення чистих прибутків розраховується за формулою

$$ПП = \frac{601309.3}{(1+0,15)^1} + \frac{816550.7}{(1+0,15)^2} + \frac{912213.5}{(1+0,15)^3} = 1740102.5 \text{ (грн)}. \quad (5.16)$$

Початкові інвестиції для вкладення інвестором обчислюються за формулою

$$PV = k_{розр} \cdot 3B, \quad (5.17)$$

де  $k_{розр}$  – коефіцієнт витрат на впровадження розробки (2).

Обчислимо значення початкових інвестицій для вкладання інвестором

$$PV = 2 \cdot 119670.6 = 239341.2 \text{ (грн.)}$$

Абсолютний економічний ефект від впровадження розробки є чистим приведеним доходом, обчислюється за формулою

$$E_{abc} = III - PV \quad (5.18)$$

В результаті абсолютний економічний ефект від впровадження розробки

$$E_{abc} = 1740102.5 - 239341.2 = 1500761.3 \text{ (грн).}$$

Отже, можлива доцільність впровадження розробки.

Внутрішня економічна дохідність допомагає остаточно визначити, чи доцільно впроваджувати розробку, розраховується за формулою

$$E_e = T_{ж} \sqrt[3]{1 + \frac{E_{abc}}{PV}} - 1, \quad (5.19)$$

де  $T_{ж}$  – життєвий цикл розробки.

$E_e$  становить

$$\sqrt[3]{1 + \frac{1500761.3}{239341.2}} - 1 = 0.94 = 94\% .$$

Мінімальна внутрішня економічна дохідність інвестицій  $\tau_{min}$  показує, чи буде інвестор зацікавлений в інвестиціях, обчислюється за формулою

$$\tau_{min} = d + f, \quad (5.20)$$

де  $d$  – середньозважена ставка за депозитними операціями в комерційних банках (обрано 0.09),  $f$  –ризикованість вкладення інвестицій (обрано 0.1).

$$\tau_{min} = 0.19 = 19\%.$$

$E_g > \tau_{min}$ , тому інвестори повинні бути зацікавлені у фінансуванні розробки.

Період окупності обчислюється за формулою

$$T_{ок} = \frac{1}{E_g} \quad (5.21)$$

Отримуємо значення  $1/0.94 = 1.06$  року. Отриманий період окупності менший за 3 роки, тому наукова розробка за темою «Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки» є комерційно привабливою.

## 5.5 Висновки

Отже, у даному розділі досліджено економічні аспекти науково-технічної розробки за темою «Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки».

Здійснено оцінювання комерційного потенціалу науково-технічної розробки шляхом систематизації результатів опитування експертів. Проведено порівняння технічних параметрів розробки та аналогу OneShader. Обчислено витрати за статтями: витрати на основну заробітну плату дослідників, на додаткову заробітну плату дослідників, на соціальні заходи, на комплектуючі, на програмне забезпечення, амортизаційні відрахування для обладнання, витрати, які не ввійшли до попередніх статей. Обчислено загальні витрати на завершення розробки. Розраховано річне збільшення чистого прибутку від реалізації для трьох років, на основі розрахунків знайдено приведену вартість збільшення всіх чистих прибутків. Здійснено порівняння внутрішньої економічної дохідності та мінімальної внутрішньої економічної дохідності для

прогнозування інтересу інвестора у фінансуванні. Розраховано період окупності розробки.

В результаті аналізу опитування експертів показано високий комерційний потенціал науково-технічної розробки. Обчислені загальні затрати для завершення розробки становлять 119670.6 грн. Зміна чистого прибутку у першому році дорівнює 601309.3 грн., у другому році 816550.7 грн., у третьому році 912213.5 грн. Встановлено, що внутрішня економічна дохідність перевищує мінімальну внутрішню економічну дохідність інвестицій. Період окупності дорівнює 1.06 року. Отже, показано, що розробка є комерційно привабливою.

## ВИСНОВКИ

Проаналізовано основні методи й програмні засоби рендерингу. Обґрунтовано необхідність розробки нових двопробених функцій відбивної здатності, програмних засобів рендерингу.

Запропоновано модифікацію функції Шліка, яка відрізняється від відомої використанням другої степені та поправочних коефіцієнтів, що дає можливість підвищити точність визначення спекулярної складової кольору та реалістичного відтворення як епіцентру відблиску, так і його блюмінгу.

Вперше запропоновано для визначення двопробеної функції відбивної здатності використання не однієї, а суми двох утворюючих функцій, що дає можливість підвищення точності визначення спекулярної складової кольору і, як наслідок, більш реалістично відтворити відблиски.

Запропоновано модифікації двопробених функцій Шліка, квадратичної функції, які відрізняються від відомих введенням поправочних коефіцієнтів, що дало можливість фізично коректно відтворювати відблиски на поверхнях тривимірних об'єктів.

Здійснено дослідження отриманихДФВЗ.

Описано розробку програмного засобу для порівняння моделей відбиття світла. Перевагами засобу над аналогами є простота функціоналу, двовіконне представлення, розрахунок відстані між сформованими зображеннями. Проведено тестування отриманих функцій відбивної здатності у розробленому засобі та аналогах Shadertoy, OneShader.

Розроблені методи та програмні засоби можуть бути використані у графічних системах і при вивченні комп'ютерної графіки.



## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. О. Н. Романюк, *Комп'ютерна графіка: Навчальний посібник*. Вінниця, Україна : ВНТУ, 1999.
2. О. Н. Романюк, «Альтернативна реалізація дистрибутивної двопрменевої функції для моделей освітлення Бліна та Фонга», *Наукові праці Донецького національного технічного університету*, вип. 106, с. 219 – 228, 2006.
3. О. Н. Романюк, «Класифікація дистрибутивних функцій відбивної здатності поверхні», *Наукові праці Донецького національного технічного університету*, №9, с. 145–151, 2008.
4. О. Н. Романюк, О. В. Романюк, С. В. Котлик, А. В. Снігур та Л. Г. Коваль, «Розробка моделі відбивної здатності поверхні з використанням поліномів Чебишева», *Інформаційні технології та комп'ютерна інженерія*, №2, с. 45–54, 2022.
5. Є. К. Завальнюк, О. Н. Романюк, О. В. Романюк, А. В. Денисюк та С. В. Котлик, «Аналіз нових моделей відбивної здатності поверхні для задач комп'ютерної графіки», у *Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації*, Одеса, Україна, 29 – 30 вересня 2022, с. 115 – 117.
6. Є. К. Завальнюк, О. Н. Романюк, О. В. Романюк, О. М. Рейда та С. В. Котлик, «Модифікація моделі Шліка для підвищення реалістичності формування зображень», у *Інформаційні технології і автоматизація – 2022*, Одеса, Україна, 20 – 21 жовтня 2022, с. 25 – 27.
7. Є. К. Завальнюк, О. Н. Романюк, С. В. Котлик, О. В. Романюк та А. В. Денисюк, «Аналіз рендерів для САПР», у *Інформаційні технології і автоматизація – 2022*, Одеса, Україна, 20 – 21 жовтня 2022, с. 74 – 76.
8. Є. К. Завальнюк, Є. Г. Станіславенко, В. В. Вінтонюк та О. Н. Романюк, «Аналіз нових фізично точних двопрменевих функцій відбивної здатності», у *Modern research in world science. Proceedings of the 9th*

*International scientific and practical conference*, Львів, Україна, 28 – 30 листопада 2022, с. 480 – 483.

9. Є. К. Завальнюк, Є. Г. Станіславенко, В. В. Вінтонюк та О. Н. Романюк, «Аналіз особливостей DirectX 12», у *Modern research in world science. Proceedings of the 9th International scientific and practical conference*, Львів, Україна, 28 – 30 листопада 2022, с. 484 – 486.

10. О. Н. Романюк, Т. М. Павлик та І. Г. Бабій, «Особливості формування тривимірних графічних зображень», у *Сучасні напрямки теоретичних і практичних досліджень*, Одеса, Україна, 2011.

11. «Моделі затенення. Плоская модель. Затенение по Гуро и Фонгу». CompGraphics. [http://compgraphics.info/3D/lighting/shading\\_model.php](http://compgraphics.info/3D/lighting/shading_model.php) (дата звернення 21 лист. 2022).

12. «Basic Lighting». LearnOpenGL. <https://learnopengl.com/Lighting/Basic-Lighting> (accessed Nov. 21, 2022).

13. В. Т. Phong, «Illumination for Computer Generated Pictures», *Communications of the ACM*, №6, p. 311–317, 1975.

14. «Blinn–Phong reflection model». Wikipedia. [https://en.wikipedia.org/wiki/Blinn–Phong\\_reflection\\_model](https://en.wikipedia.org/wiki/Blinn–Phong_reflection_model) (accessed Nov. 21, 2022).

15. G. Ward, «Measuring and Modeling Anisotropic Reflection», *ACM SIGGRAPH Computer Graphics*, №2, p. 265–272, 1992.

16. R. Cook, «A Reflectance Model for Computer», *ACM Transactions on Graphics*, №1, p. 7–24, 1982.

17. M. Ashikhmin, «An Anisotropic Phong BRDF Model», *Journal of Graphics Tools*, №2, p. 25–32, 2001.

18. «Рендеринг». Вікіпедія. <https://uk.wikipedia.org/wiki/Рендеринг> (дата звернення 21 лист. 2022).

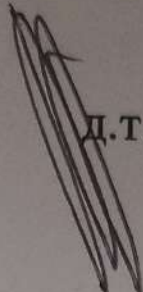
19. «CPU vs. GPU Rendering: Which Should You Choose and Why?». IncrediBuild. <https://www.incredibuild.com/blog/cpu-vs-gpu-rendering-which-should-you-choose-and-why> (accessed Nov. 21, 2022).
20. «CPU vs. GPU Renderer: Which is Better?». AcademyOfAnimatedArt. <https://academyofanimatedart.com/cpu-vs-gpu-renderer-which-is-better/#:~:text=Modern%20GPUs%20offer%20superior%20processing,times%20faster%20than%20CPU%20rendering> (accessed Nov. 21, 2022).
21. «What is a Render Farm? Everything you need to know about Render Farms». RebusFarm. <https://us.rebusfarm.net/en/blog/2990-what-is-a-render-farm-everything-you-need-to-know-about-render-farms> (accessed Nov. 21, 2022).
22. «About». OneShader. <https://oneshader.net/about> (accessed Nov. 21, 2022).
23. «Modern Physically-Based Rendering». RenderMan. <https://renderman.pixar.com/tech-specs> (accessed Nov. 21, 2022).
24. «Download & Installation». RenderMan. [https://renderman.pixar.com/intro\\_logged\\_in#:~:text=RenderMan%20is%20free%20for%20all,%2C%20research%2C%20and%20personal%20projects](https://renderman.pixar.com/intro_logged_in#:~:text=RenderMan%20is%20free%20for%20all,%2C%20research%2C%20and%20personal%20projects) (accessed Nov. 21, 2022).
25. «Renderman Review». Slant. <https://www.slant.co/options/24860/~renderman-review> (accessed Nov. 21, 2022).
26. «Rendering». Blender. <https://www.blender.org/features/rendering/> (accessed Nov. 21, 2022).
27. «Winning the render wars with Chad Ashley». MotionGrapher. <https://motionographer.com/2017/08/14/winning-the-render-wars-with-chad-ashley/> (accessed Nov. 21, 2022).
28. A. Glawion. «What's the Best 3D Render Engine (GPU & CPU) for your Needs?». CGDirector. <https://www.cgdirector.com/best-3d-render-engines/> (accessed Nov. 21, 2022).

29. «Redshift Render: A complete overview». VFXRendering. <https://vfxrendering.com/redshift-render-a-complete-overview/> (accessed Nov. 21, 2022).
30. «An Overview of Redshift in Cinema 4D». SchoolOfMotion. <https://www.schoolofmotion.com/blog/redshift-in-cinema-4d> (accessed Nov. 21, 2022).
31. «OctaneRender Support Guides». Otoy. <https://help.otoy.com/hc/en-us/categories/201718003-OctaneRender-Support-Guides> (accessed Nov. 21, 2022).
32. «About Shadertoy». Shadertoy. <https://www.shadertoy.com/about> (accessed Nov. 21, 2022).
33. «Documentation». Shadertoy. <https://www.shadertoy.com/howto> (accessed Nov. 21, 2022).
34. Є. К. Завальнюк, О. Н. Романюк, В. В. Войтко, О. В. Романюк та А. В. Снігур, «Розробка модифікованої моделі Шліка для визначення спекулярної складової кольору», *Інформаційні технології та комп'ютерна інженерія*, вип. 55, №3, с. 4 – 12, 2022.
35. О. Н. Романюк, І. В. Абрамчук, С. А. Кирилащук та С. О. Романюк, «Моделювання спекулярного складника кольору з використанням енергетично-коректних моделей відбивних здатностей поверхонь», *Вчені записки ТНУ імені В.І. Вернадського. Серія «Технічні науки»*, №3, с. 153–157, 2019.
36. О. Н. Романюк та А. В. Чорний, *Високопродуктивні методи та засоби зафарбовування тривимірних графічних об'єктів*. Вінниця, Україна : УНІВЕСУМ-Вінниця, 2006.
37. «Pros and cons of programming languages». ChubbyDeveloper. <https://www.chubbydeveloper.com/pros-and-cons-of-programming-languages/> (accessed Nov. 21, 2022).
38. «Advantages and Disadvantages of C++». Data-Flair. <https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/> (accessed Nov. 21, 2022).

39. «Advantages and Disadvantages of Python | Python Language Advantages, Disadvantages and Its Applications». APlusTopper. <https://www.aplustopper.com/advantages-and-disadvantages-of-python/> (accessed Nov. 21, 2022).

40. В. О. Козловський, О. Й. Лесько та В. В. Кавецький, *Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт*. Вінниця, Україна : ВНТУ, 2021.

**ДОДАТКИ**



ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

«16» вересня 2022 р.

**Технічне завдання**

**на магістерську кваліфікаційну роботу**

**«Розробка методів і програмних засобів рендерингу для систем  
комп'ютерної графіки»**

**за спеціальністю**

**121 – Інженерія програмного забезпечення**

Керівник магістерської кваліфікаційної роботи:

д.т.н., проф. О.Н. Романюк



" 16 " 09 2022 р.

Виконав:



студент гр.1ПІ-21м Є. К. Завальнюк

" 16 " 09 2022 р.

## **1. Найменування та галузь застосування**

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки».

Галузь застосування – системи комп'ютерної графіки.

## **2. Підстава для розробки.**

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР і наказ ректора по ВНТУ № 205-А від «15» вересня 2022 р. про закріплення тем МКР.

## **3. Мета та призначення розробки.**

Метою роботи є підвищення реалістичності відтворення спекулярної складової кольору за рахунок розробки та впровадження нових двопробових функцій відбивної здатності поверхні.

Призначення роботи – розробка методів і програмних засобів рендерингу.

## **4 Вихідні дані для проведення МКР**

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

1. О. Н. Романюк, *Комп'ютерна графіка: Навчальний посібник*. Вінниця, Україна : ВНТУ, 1999.

2. О. Н. Романюк, «Альтернативна реалізація дистрибутивної двопробової функції для моделей освітлення Бліна та Фонга», *Наукові праці Донецького національного технічного університету*, вип. 106, с. 219 – 228, 2006.

3. О. Н. Романюк, «Класифікація дистрибутивних функцій відбивної здатності поверхні», *Наукові праці Донецького національного технічного університету*, №9, с. 145–151, 2008.



4. О. Н. Романюк, О. В. Романюк, С. В. Котлик, А. В. Снігур та Л. Г. Коваль, «Розробка моделі відбивної здатності поверхні з використанням поліномів Чебишева», *Інформаційні технології та комп'ютерна інженерія*, №2, с. 45–54, 2022.

5. О. Н. Романюк, Т. М. Павлик та І. Г. Бабій, «Особливості формування тривимірних графічних зображень», у *Сучасні напрямки теоретичних і практичних досліджень*, Одеса, Україна, 2011.

6. О. Н. Романюк, І. В. Абрамчук, С. А. Кирилащук та С. О. Романюк, «Моделювання спекулярного складника кольору з використанням енергетично-коректних моделей відбивних здатностей поверхонь», *Вчені записки ТНУ імені В.І. Вернадського. Серія «Технічні науки»*, №3, с. 153–157, 2019.

7. О. Н. Романюк та А. В. Чорний, *Високопродуктивні методи та засоби зафарбовування тривимірних графічних об'єктів*. Вінниця, Україна: УНІВЕСУМ-Вінниця, 2006.

## **5. Технічні вимоги**

Модель освітлення – на основі двопроменевої функції відбивної здатності поверхні; кольоровий режим – TrueColor; вихідні дані для зафарбовування – вектори нормалей до спостерігача  $\vec{V}$ , до джерела світла  $\vec{L}$ , серединний вектор  $\vec{H}$ ; діапазон зміни коефіцієнта спекулярності поверхні 1 – 1000, мова програмування – C#; розмір дискретного координатного простору – 4096\*4096.

## **6. Конструктивні вимоги.**

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

Графічна та текстова документація повинна відповідати діючим стандартам України.

## 7. 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 стану питання та постановка задач дослідження	17.09.2022- 25.09.2022
2	Розробка нових дистрибутивних функцій відбивної здатності поверхні	26.09.2022- 16.10.2022
3	Модифікація косинус-квадратичної та квадратичної моделей освітлення	17.10.2022- 06.11.2022
4	Розробка програмного засобу для порівняння моделей відбиття та тестування розроблених функцій відбивної здатності	07.11.2022- 20.11.2022
5	Економічна частина	21.11.2022- 08.12.2022

## 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

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ)  
РОБОТИ

Назва роботи: Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ІПІ – 21м

Науковий керівник: д.т.н., професор каф. ПЗ Романюк О.Н.

Unicheck	
Оригінальність	98.13
Схожість	1.87

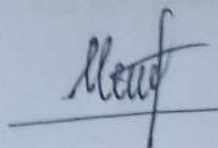
## Аналіз звіту подібності

■ 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.

Особа, відповідальна за перевірку



Черноволик Г. О.

Опис прийнятого рішення: допустити до захисту

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Завальнюк Є.К.

Автор роботи

3

Романюк О.Н.

Керівник роботи



## ДОДАТОК В

## Лістинг програми обчислення оптимальних параметрів моделі Шліка

```

using System;
using System.Collections.Generic;
using System.Linq;

namespace coefficients3
{
    class Program
    {
        static void Main(string[] args)
        {
            double[] pows = new double[7] { -3,-2, -1, 0, 1, 2,3 };
            double[] coefficients = new double[8] {0.25,0.5,1,1.25,1.5,2,2.5,3};
            double[,] blinvalues = new double[91, 1000];
            blinvalues = FindBlinnValues();
            Dictionary<double[], double> models = new Dictionary<double[], double>();
            models = BestModels(pows, blinvalues);
            double [,] models_parameters = UpdateWithCoefficients(models, coefficients,
            blinvalues);
            PrintResults(models_parameters);
            Console.ReadLine();
        }
        public static double [,] FindBlinnValues()
        {
            double[,] blinvalues = new double[91, 1000];
            for (int deg = 0; deg <= 90; deg++)
            {
                for (int n = 1; n <= 1000; n++)
                {
                    blinvalues[deg, n - 1] = Math.Pow(Math.Cos(torad(deg)), n);
                }
            }
            return blinvalues;
        }
        public static Dictionary<double[], double> BestModels(double [] pows, double [,]
            blinvalues)
        {
            double[] bestval = new double[5];
            Dictionary<double[], double> models = new Dictionary<double[], double>();
            for (int i = 0; i < 7; i++)
            {
                for (int i2 = 0; i2 < 7; i2++)
                {
                    for (int i3 = 0; i3 < 7; i3++)
                    {
                        for (int i5 = 0; i5 < 7; i5++)
                        {
                            double[,] ShlickValues = new double[91, 1000];
                            for (int deg = 0; deg <= 90; deg++)
                            {
                                for (int n = 1; n <= 1000; n++)
                                {
                                    ShlickValues[deg, n - 1] = Shlick(pows[i], pows[i2],
                                    pows[i3], pows[i3], pows[i5], deg, n);
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

double rzn = Riznytsya(blinvalues, ShlickValues);
if (models.Count < 5)
{
    bestval = new double[5];
    bestval[0] = pows[i];
    bestval[1] = pows[i2];
    bestval[2] = pows[i3];
    bestval[3] = pows[i3];
    bestval[4] = pows[i5];
    models.Add(bestval, rzn);
}
else
{
    double max = models.Values.Max();
    if (rzn < max)
    {
        models.Remove(models.Where(a => a.Value ==
max).FirstOrDefault().Key);
        bestval = new double[5];
        bestval[0] = pows[i];
        bestval[1] = pows[i2];
        bestval[2] = pows[i3];
        bestval[3] = pows[i3];
        bestval[4] = pows[i5];
        models.Add(bestval, rzn);
    }
}
}
}
}
}
return models;
}
public static double [,] UpdateWithCoefficients(Dictionary<double[,], double> models,
double [,] coefficients, double [,] blinvalues)
{
    double min;
    double[, ] bestval2 = new double[5, 11];
    for (int k = 0; k < 5; k++)
    {
        min = 91000;
        var onemodelstepins = models.ElementAt(k).Key;

        for (int i = 0; i < 8; i++)
        {
            for (int i2 = 0; i2 < 8; i2++)
            {
                for (int i3 = 0; i3 < 8; i3++)
                {
                    for (int i5 = 0; i5 < 8; i5++)
                    {
                        double[, ] ShlickValues = new double[91, 1000];
                        for (int deg = 0; deg <= 90; deg++)
                        {
                            for (int n = 1; n <= 1000; n++)
                            {
                                ShlickValues[deg, n - 1] =
Shlick2(onemodelstepins[0], onemodelstepins[1], onemodelstepins[2], onemodelstepins[3],
onemodelstepins[4], coefficients[i], coefficients[i2], coefficients[i3], coefficients[i3],
coefficients[i5], deg, n);
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        double rzn = Riznytsya(blinvalues, ShlickValues);
        if (rzn < min)
        {
            min = rzn;
            bestval2[k,0] = (double)(min) / 91000;
            bestval2[k,1] = onemodelstepins[0];
            bestval2[k,2] = onemodelstepins[1];
            bestval2[k,3] = onemodelstepins[2];
            bestval2[k,4] = onemodelstepins[3];
            bestval2[k,5] = onemodelstepins[4];
            bestval2[k,6] = coefficients[i];
            bestval2[k,7] = coefficients[i2];
            bestval2[k,8] = coefficients[i3];
            bestval2[k,9] = coefficients[i3];
            bestval2[k,10] = coefficients[i5];
        }
    }
}

    }
}

    }
    return bestval2;
}
public static double torad(double degrees)
{
    return degrees * Math.PI / 180.0;
}
public static double Shlick(double pow1, double pow2, double pow3, double pow4,
double pow5, double deg, double n)
{
    double rez = Math.Pow(Math.Cos(torad(deg)), pow1) / Math.Pow(Math.Pow(n, pow3) -
Math.Pow(n * Math.Cos(torad(deg)), pow4) + Math.Pow(Math.Cos(torad(deg)), pow5), pow2);
    return rez;
}
public static double Shlick2(double pow1, double pow2, double pow3, double pow4,
double pow5, double co1, double co2, double co3, double co4, double co5, double deg, double n)
{
    double rez = (co1 * pow(Math.Cos(torad(deg)), pow1) ) / (co2 * (Math.Pow((co3 *
pow(n, pow3) -
co4 * pow(n * Math.Cos(torad(deg)), pow4) + co5 * pow(Math.Cos(torad(deg)), pow5)), pow2));
    return rez;
}
public static double pow(double number, double stepin)
{
    return Math.Pow(number, stepin);
}
public static double Riznytsya(double[,] blinvalues, double[,] newvalues)
{
    double rez = 0;
    for (int i = 0; i < 91; i++)
    {
        for (int j = 0; j < 1000; j++)
        {
            rez += Math.Abs(((double)(blinvalues[i, j] - newvalues[i, j])));
        }
    }
    return rez;
}
public static void PrintResults(double [,] models_parameters)
{
    for(int i=0;i< 5;i++)
    {

```

```
        Console.WriteLine("модель " + i + " степені " + models_parameters[i,1] + " "
+ models_parameters[i,2] + " " + models_parameters[i,3] + " " + models_parameters[i,4] + " "
+ models_parameters[i,5] + " коефіцієнти " + models_parameters[i,6] + " " +
models_parameters[i,7] + " " + models_parameters[i,8] + " " + models_parameters[i,9] + " " +
models_parameters[i,10] + " точність " + models_parameters[i,0]);
    }
}
}
```

## ДОДАТОК Г

## Лістинг програмного засобу

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BRDFViz
{
    public static class MathOper
    {
        public static double VectLen(double [] vec)
        {
            double rez = 0;
            foreach (var v in vec)
            {
                rez += Math.Pow(v, 2);
            }
            rez = Math.Sqrt(rez);
            return rez;
        }
        public static double [] Normalize(double [] vec)
        {
            double [] vec2 = new double[vec.Length];
            double znamennyk = VectLen(vec);
            for(int i=0;i<vec.Length;i++)
            {
                vec2[i] = vec[i] / znamennyk;
            }
            return vec2;
        }
        public static double CosKut(double []v1,double []v2)
        {
            double znamennyk1 = VectLen(v1);
            double znamennyk2 = VectLen(v2);
            double scaldob = 0;
            for (int i = 0; i < v1.Length; i++)
            {
                scaldob+= v1[i]*v2[i];
            }
            double coskut = scaldob / (znamennyk1 * znamennyk2);
            return coskut;
        }
        public static double[] VectDiff(double[] v1, double[] v2)
        {
            double[] v3 = new double[v1.Length];
            for (int i = 0; i < v1.Length; i++)
            {
                v3[i] = v1[i] -v2[i];
            }

            return v3;
        }
        public static double[] VectSum(double[] v1, double[] v2)
        {
            double[] v3 = new double[v1.Length];
            for (int i = 0; i < v1.Length; i++)
            {

```



```

        v3[i] = v1[i] + v2[i];
    }

    return v3;
}
public static double[] VectByConst(double[] vec, double cons)
{
    double[] v2 = new double[vec.Length];
    for (int i = 0; i < vec.Length; i++)
    {
        v2[i] = cons*vec[i];
    }

    return v2;
}

public static double[,] MatrixMult(double [,] m1, double[,] m2)
{
    int rows1 = m1.GetLength(0);
    int columns1 = m1.GetLength(1);
    int rows2 = m2.GetLength(0);
    int columns2 = m2.GetLength(1);
    if (columns1 != rows2)
    {
        Console.WriteLine("Неправильні розміри");
        return null;
    }
    double[,] retmas = new double[rows1, columns2];
    for (int i = 0; i < rows1; i++)
    {
        for (int j = 0; j < columns2; j++)
        {
            int k = 0;
            while (k < rows2)
            {
                retmas[i, j] += m1[i, k] * m2[k, j];
                k++;
            }
        }
    }
    return retmas;
}

public static double[,] Transpose(double [,] m1)
{
    int rows1 = m1.GetLength(0);
    int columns1 = m1.GetLength(1);
    double[,] m2 = new double[rows1, columns1];
    for (int i = 0; i < rows1; i++)
    {
        for (int j = 0; j < columns1; j++)
        {
            m2[j, i] = m1[i, j];
        }
    }
    return m2;
}

public static double[,] CopyMatr(double[,] m1, int rows1, int columns1)
{
    double [,] m2 = new double[rows1, columns1];
    for (int i = 0; i < rows1; i++)
    {
        for (int j = 0; j < columns1; j++)

```

```

        {
            m2[i, j] = m1[i,j];
        }
    }
    return m2;
}
public static double[,] Invert4(double[,] m1)
{
    int rows1 = m1.GetLength(0);
    int columns1 = m1.GetLength(1);
    double[,] m2 = new double[rows1, columns1];
    m2 = CopyMatr(m1,rows1,columns1);
    double mnozhn = 1/(m2[0,0]*m2[1,1]-m2[0,1]*m2[1,0]);
    double temp = m2[0, 0];
    m2[0, 0] = m2[1, 1];
    m2[1, 1] = temp;
    temp = m2[0, 1];
    m2[0, 1] = (-1)*m2[1, 0];
    m2[1, 0] = (-1) * temp;
    for (int i = 0; i < rows1; i++)
    {
        for (int j = 0; j < columns1; j++)
        {
            m2[i, j] *= mnozhn;
        }
    }
    return m2;
}
public static double[,] MatrixFromVector(double [] vec)
{
    double[,] vec2 = new double[vec.Length,1];
    for (int i = 0; i < vec.Length; i++)
    {
        vec2[i, 0] = vec[i];
    }
    return vec2;
}
public static double[] VectorFromMatrix(double[,] vec)
{
    double[] vec2 = new double[vec.GetLength(0)];
    for (int i = 0; i < vec.GetLength(0); i++)
    {
        vec2[i] = vec[i,0];
    }
    return vec2;
}
public static double[,] FillMatrixWithInf(int m,int n)
{
    double[,] mat = new double[m,n];
    for (int i = 0; i < m; i++)
    {
        for (int j = 0;j < n; j++)
        {
            mat[i, j] = double.PositiveInfinity;
        }
    }
    return mat;
}
public static bool InsideTriangle(int x, int y, (double[] a, double[] b, double[] c)
vershyny)
{
    var rez= CoordBarycentric(x,y,vershyny);
}

```

```

        if(rez.lambda1>=0&&rez.lambda2>=0&&rez.lambda3 >=0)
            return true;
        return false;
    }
    public static (double lambda1, double lambda2, double lambda3) CoordBarycentric(int
x, int y, (double[] a, double[] b, double[] c) trianglevershyny)
    {
        var ver1 = trianglevershyny.a;
        var ver2 = trianglevershyny.b;
        var ver3 = trianglevershyny.c;
        double determinant = (ver2[1] - ver3[1]) * (ver1[0] - ver3[0]) + (ver3[0] -
ver2[0]) * (ver1[1] - ver3[1]);
        double lambda1 = (ver2[1] - ver3[1]) * (x - ver3[0]) + (ver3[0] - ver2[0]) * (y
- ver3[1]);
        lambda1 /= determinant;
        double lambda2 = (ver3[1] - ver1[1]) * (x - ver3[0]) + (ver1[0] - ver3[0]) * (y
- ver3[1]);
        lambda2 /= determinant;
        double lambda3 = 1 - lambda1 - lambda2;
        return (lambda1, lambda2, lambda3);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BRDFViz
{
    public class RGBval
    {
        public double red { get; set; }
        public double green { get; set; }
        public double blue { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;

namespace BRDFViz
{
    class PhongBRDF:BRDF
    {
        public PhongBRDF(double KoefAmb, double KoefSpec, double KoefDiff, int
n_val):base(KoefAmb, KoefSpec, KoefDiff, n_val)
        {
        }

        public override double Specular(double kut, int n_val)

```

```

        {
            double rez = Math.Pow(kut, n_val);
            return rez;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;

namespace BRDFViz
{
    public class SchlickBRDF: BRDF
    {
        public SchlickBRDF(double KoefAmb, double KoefSpec, double KoefDiff, int n_val) :
base(KoefAmb, KoefSpec, KoefDiff, n_val)
        {
        }
        public override double Specular(double kut, int n_val)
        {
            double rez = ((1 * kut) / (Math.Pow(n_val - n_val * kut + kut, 1)));
            return rez;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BRDFViz
{
    public class NewBRDF:BRDF
    {
        public NewBRDF(double KoefAmb, double KoefSpec, double KoefDiff, int n_val) :
base(KoefAmb, KoefSpec, KoefDiff, n_val)
        {
        }
        public override double Specular(double kut, int n_val)
        {
            double rez = ((2 * kut) / (1.25 * Math.Pow(n_val - n_val * kut + 1.25 * kut,
2)));
            return rez;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

using System.Windows.Forms;

namespace BRDFViz
{
    public class CustomBRDF: BRDF
    {
        public CustomBRDF(double KoefAmb, double KoefSpec, double KoefDiff, int n_val) :
base(KoefAmb, KoefSpec, KoefDiff, n_val)
        {
        }

        public override double Specular(double kut, int n_val)
        {
            double [] L=GetLVector(DzhereIo.Dzpos, Tochka);
            double[] V = GetVVector(sposterigach, new double[3] { Tochka[0], Tochka[1],
Tochka[2] });
            double[] H = GetHVector(L,V);
            double kut1 = MathOper.CosKut(Normal, H);
            double kut2 = MathOper.CosKut(Normal, L);
            double kut3 = MathOper.CosKut(Normal, V);
            Dictionary<string, double> slovnyk = new Dictionary<string, double>() { { "n",
n_val }, { "VR", kut }, { "NH", kut1 }, { "NL", kut2 }, { "NV", kut3 } };
            double rez = 0;
            try
            {
                NCalc.Expression expr = new NCalc.Expression(formula);
                expr.EvaluateParameter += (name, args) =>
                {
                    args.Result =slovnyk[name];
                };

                rez=Math.Max((double)expr.Evaluate(),0);
            }
            catch
            {
                MessageBox.Show("Неправильний ввід");
            }
            return rez;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BRDFViz
{
    public class Tochka
    {
        public double[] d3scenecoord;

        public double[] d3spherecoord;

        public double[] d2imagecoord;

        public double[] Normal;

        public double[,] tempmatrix= new double[4, 4] { { 1, 0, 0, 0 }, { 0, 1, 0, 0 }, { 0,
0, 1, 0 }, { 1, -1, 0, 1 } };
        public Tochka(double x, double y, double z)
    }
}

```

```

    {
        d3spherecoord= new double[4] { x, y, z, 1 };
        Normal = MathOper.Normalize(d3spherecoord);
        Normal = MathOper.VectorFromMatrix(MathOper.MatrixMult(new
MathOper.Matrix(tempmatrix),new MathOper.Matrix(MathOper.MatrixFromVector(Normal))));
        Normal = MathOper.Normalize(Normal);
    }
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;

namespace BRDFViz
{
    public class BRDF
    {
        public virtual double Specular(double kut, int n_val)
        {
            return 0;
        }
        public double [] GetWVector(double[] sposterigach,double []tochka)
        {
            double[] V1 = new double[3];
            V1 = MathOper.VectDiff(sposterigach, tochka);
            V1 = MathOper.Normalize(V1);
            return V1;
        }
        public double[] GetLVector(double[] Dzpos, double[] tochka)
        {
            double[] L1 = new double[3];
            L1 = MathOper.VectDiff(Dzpos, tochka);
            L1 = MathOper.Normalize(L1);
            return L1;
        }
        public double[] GetHVector(double[] L, double[] V)
        {
            double[] H = new double[3];
            H = MathOper.VectSum(L, V);
            H= MathOper.Normalize(H);
            return H;
        }

        protected string formula;
        protected double[] sposterigach=new double[3] { 0.0, 0.0, 6.0 };
        protected double KoefSpec;
        protected double KoefDiff;
        protected double KoefAmb=5.0;
        protected int n_val;
        protected Color IntAmb=Color.Red;
        protected (double [] Dzpos, Color IntPoint) DzhereIo=(new double[3] { 10.0, 0.0,
10.0 }, Color.White);
        protected double[] Normal;
        protected double[] Tochka;

        public BRDF(double koefAmb, double koefSpec, double koefDiff, int n_val_znach)
        {

```

```

        KoefAmb = koefAmb;
        KoefSpec = koefSpec;
        KoefDiff = koefDiff;
        n_val = n_val_znach;
    }

    public Color PixelIntensity(double[] tochka, double [] normal,string customeq)
    {
        formula = customeq;
        Normal = normal;
        Tochka = tochka;
        RGBval values = new RGBval();
        values.red = KoefAmb * IntAmb.R;
        values.green = KoefAmb * IntAmb.G;
        values.blue = KoefAmb * IntAmb.B;
        double [] tochka3= new double[3] { tochka[0], tochka[1], tochka[2] };
        double[] V1 = GetVVector(sposterigach, tochka3);
        double[] L1 =GetLVector(Dzherelo.Dzpos, tochka3);
        var IntPoint = Dzherelo.IntPoint;
        double kut1 = MathOper.CosKut(L1, normal) ;
        double diffuse = Math.Max(kut1, 0.0);
        double[] temppar = MathOper.VectByConst(normal, 2 * diffuse);
        double[] R1 = MathOper.VectDiff(temppar, L1);
        double specular1 = MathOper.CosKut(R1, V1);
        double difpart = KoefDiff * diffuse;
        double specpart = KoefSpec *Specular(specular1, n_val);
        values.red += (IntPoint.R * difpart + IntPoint.R * specpart);
        values.green += (IntPoint.G * difpart + IntPoint.G * specpart);
        values.blue += (IntPoint.B * difpart + IntPoint.B * specpart);
        var red2 = KorygColir(values.red);
        var green2 = KorygColir(values.green);
        var blue2 = KorygColir(values.blue);
        return Color.FromArgb(red2,green2,blue2);
    }
    protected int KorygColir(double d)
    {
        if(d<0)
        {
            d = 0;
        }
        if(d>255)
        {
            d = 255;
        }
        return (int)d;
    }
}
}
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;using System.Windows.Forms;

namespace BRDFViz
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            koefamb.Value = 0.4M;
            koefspec.Value = 0.5M;
        }
    }
}

```

```

    koefdif.Value = 0.5M;
    nval.Value = 50M;
    VizulizeNatysk.Click += VizualizeNatysk_Click_1;
}

private double [,] ImageValues(Bitmap image)
{
    double[,] rez = new double[image.Width, image.Height];
    for (int i = 0; i < image.Width; i++)
    {
        for (int j = 0; j < image.Height; j++)
        {
            double r = image.GetPixel(i, j).R;
            double g = image.GetPixel(i, j).G;
            double b = image.GetPixel(i, j).B;
            double grays = 0.299 * r + 0.587 * g + 0.114 * b;
            rez[i, j] = grays;
        }
    }
    return rez;
}

private void Pohybka()
{
    Bitmap bit = new Bitmap(pictureBox.ClientSize.Width, pictureBox.Height);
    pictureBox.DrawToBitmap(bit, pictureBox.ClientRectangle);
    Bitmap bit2 = new Bitmap(pictureBox1.ClientSize.Width, pictureBox1.Height);
    pictureBox1.DrawToBitmap(bit2, pictureBox1.ClientRectangle);
    double[,] mas1 = new double[bit.Width, bit.Height];
    double[,] mas2 = new double[bit.Width, bit.Height];
    mas1 = ImageValues(bit);
    mas2 = ImageValues(bit2);
    double manhattan = 0;
    for (int i = 0; i < bit.Width; i++)
    {
        for (int j = 0; j < bit.Height; j++)
        {
            manhattan+=Math.Abs( mas1[i, j] - mas2[i, j]);
        }
    }
    manhattan = manhattan / (bit.Height*bit.Width);
    label2.Text = manhattan.ToString();
}

Bitmap pmb;
double[,] matrix = new double[4, 4];
double[,] matrix4 = new double[4, 4];
double[,] matrix5 = new double[4, 4];
List<Tochka[]> Trykutnyky = new List<Tochka[]>();
Tochka[,] mas = new Tochka[20, 20];
BRDF bRDF;
private double[,] zBuffer=MathOper.FillMatrixWithInf(499, 520);

private void DrawWindow()
{
    int vybir = comboBox1.SelectedIndex;
    InstantiateBRDF(vybir);
    zBuffer = MathOper.FillMatrixWithInf(pictureBox.Width, pictureBox.Height);
    pmb = new Bitmap(pictureBox.Width, pictureBox.Height);
    Graphics g = Graphics.FromImage(pmb);
    StvorytySphery();
    GetSpeherePoints(1 * 3.14 / 20, 3.14/20,1);
    GetTrykutnyky();
    foreach (var t in Trykutnyky)

```



```

    {
        ColorTriangle(t);
    }
    pictureBox.BackgroundImage = pmb;
    g.Dispose();
}
private void DrawWindow2()
{
    int vybir = comboBox2.SelectedIndex;
    InstantiateBRDF(vybir);
    zBuffer = MathOper.FillMatrixWithInf(pictureBox1.Width, pictureBox.Height);
    pmb = new Bitmap(pictureBox1.Width, pictureBox1.Height);
    Graphics g = Graphics.FromImage(pmb);
    StvorytySphery();
    GetSpeherePoints(1 * 3.14 / 20, 3.14 / 20, 1);
    GetTrykutnyky();
    foreach (var t in Trykutnyky)
    {
        ColorTriangle(t);
    }
    pictureBox1.BackgroundImage = pmb;
    g.Dispose();
}
private void ChooseBRDF(int vyb, double KoefAmb, double KoefSpec, double KoefDiff,
int n_value)
{
    if (vyb == 0)
    {
        bRDF = new PhongBRDF(KoefAmb, KoefSpec, KoefDiff, n_value);
    }
    if (vyb == 1)
    {
        bRDF = new SchlickBRDF(KoefAmb, KoefSpec, KoefDiff, n_value);
    }
    if (vyb == 2)
    {
        bRDF = new NewBRDF(KoefAmb, KoefSpec, KoefDiff, n_value);
    }
    if (vyb == 3)
    {
        bRDF = new CustomBRDF(KoefAmb, KoefSpec, KoefDiff, n_value);
    }
}
private void InstantiateBRDF(int vyb)
{
    double KoefAmb = (double)koefamb.Value; double KoefSpec =
(double)koefspec.Value; double KoefDiff = (double)koefdif.Value;
    int n_value = (int)nval.Value;
    ChooseBRDF(vyb, KoefAmb, KoefSpec, KoefDiff, n_value);
}
private void StvorytySphery()
{
    float centerSphereX = -1;
    float centerSphereY = 1;
    float centerSphereZ = 0;
    matrix = new double[4,4] { { 1, 0, 0, 0 }, { 0, 1, 0, 0 }, { 0, 0, 1, 0 }, { 0,
0, 0, 1 } };
    double [,] matrix2 = new double[4, 4] { { 1, 0, 0, centerSphereX }, { 0, 1, 0,
centerSphereY }, { 0, 0, 1, centerSphereZ }, { 0, 0, 0, 1 } };
    matrix = MathOper.MatrixMult2(matrix, matrix2);
}

```

```

        matrix4 = new double[4, 4] { { 1, 0, 0, 0 }, { 0, 1, 0, 0 }, { 0, 0, 1, -6 }, {
0, 0, 0, 1 } };
        matrix5 = new double[4, 4] { { 2.41, 0, 0, 0 }, { 0, 2.51, 0, 0 }, { 0, 0, -1, -
0.2 }, { 0, 0, -1, 0 } };
    }
    public double[] ToSceneCoord(double[] coord)
    {
        double[,] coord2 = new double[4, 1];
        for(int i=0;i<coord2.Length;i++)
        {
            coord2[i, 0] = coord[i];
        }
        double[,] rez = MathOper.MatrixMult2(matrix,coord2);
        double[] rez2 = new double[4];
        for (int i = 0; i < coord.Length; i++)
        {
            rez2[i] = rez[i,0];
        }
        return rez2;
    }
    public double[] To2DCoord(double[] coord)
    {
        double[,] coord2 = MathOper.MatrixFromVector(coord);

        coord2 = MathOper.MatrixMult2(matrix,coord2);
        coord2 = MathOper.MatrixMult2(matrix4, coord2);
        coord2 = MathOper.MatrixMult2((matrix5), coord2);
        double[] rezvek= MathOper.VectorFromMatrix(coord2);
        rezvek[0] = (rezvek[0]/rezvek[3]+1)* pictureBox.Width/2.0;
        rezvek[1] = (rezvek[1] / rezvek[3] + 1) * pictureBox.Height / 2.0;
        rezvek[2] = (rezvek[2] / rezvek[3]);
        rezvek[3] = 1;
        return rezvek;
    }
    public void GetSpeherePoints(double xkrok, double ykrok, double r_sphere)
    {
        for (int j = 0; j < 20; ++j)
        {
            for (int i = 0; i < 20; ++i)
            {
                double x =( r_sphere * Math.Sin(j * ykrok) * Math.Cos(i * xkrok));
                double y = (r_sphere * Math.Cos(j * ykrok));
                double z = (r_sphere * Math.Sin(j * ykrok) * Math.Sin(i * xkrok));
                Tochka temp = new Tochka(x, y, z);
                temp.d3scenecoord = ToSceneCoord(temp.d3spherecoord);
                temp.d2imagecoord= To2DCoord(temp.d3spherecoord);

                mas[j, i] = temp;
            }
        }
    }

    private int Poperzn(int i)
    {
        int rezultat = i - 1;
        if (i - 1 < 0) {rezultat = 19; }
        return rezultat;
    }
    public void GetTrykutnyky()
    {
        for (int j = 0; j < 19; j++)
        {

```

```

        for (int i = 0; i < 20; i++)
        {
            Trykutnyky.Add(new Tochka[3] { mas[j,i],mas[j, (i + 1) %
20],mas[j+1,i]});
            Trykutnyky.Add(new Tochka[3] { mas[j, i], mas[j + 1, i], mas[j + 1,
Poperzn(i)] });
        }
    }
}

private void ColorTriangle(Tochka[] tochky)
{
    List<int> x = new List<int>();
    List<int> y = new List<int>();
    foreach (var v in tochky)
    {
        x.Add((int)v.d2imagecoord[0]);
        y.Add((int)v.d2imagecoord[1]);
    }
    int xma = x.Max();
    int xmi = x.Min();
    int yma = y.Max();
    int ymi = y.Min();
    for(int i = xmi; i <= xma; i++)
    {
        for (int j = ymi; j <= yma; j++)
        {
            Point temp = new Point(i, j);
            if (MathOper.InsideTriangle(i, j, (tochky[0].d2imagecoord,
tochky[1].d2imagecoord, tochky[2].d2imagecoord)))
            {
                var barycentric = MathOper.CoordBarycentric(i, j,
(tochky[0].d2imagecoord, tochky[1].d2imagecoord, tochky[2].d2imagecoord));
                if (ZBuffer(i, j, tochky[0], tochky[1], tochky[2],
barycentric.lambda1, barycentric.lambda2, barycentric.lambda3)==false)
                {
                    continue;
                }
                Color color = DotInPhongShading(barycentric, tochky);
                pmb.SetPixel(i, j, color);
            }
        }
    }
}

private bool ZBuffer(int i, int j, Tochka A, Tochka B, Tochka C, double lambda1,
double lambda2, double lambda3)
{
    bool rez = false;
    double thirdDim = lambda1 * A.d2imagecoord[2] + lambda2 * B.d2imagecoord[2] +
lambda3 * C.d2imagecoord[2];
    if (thirdDim<zBuffer[i,j])
    {
        zBuffer[i,j] = thirdDim;
        rez= true;
    }
    return rez;
}

public Color DotInPhongShading((double lambda1, double lambda2, double lambda3)
barycentric, Tochka[] tochky)

```

```

    {
        double[] normal1 = tochky[0].Normal;
        double[] normal2 = tochky[1].Normal;
        double[] normal3 = tochky[2].Normal;
        double normx = barycentric.lambda1 * normal1[0] + barycentric.lambda2 *
normal2[0] + barycentric.lambda3 * normal3[0]; //барицентричні координати для отримання
інтерпольованих значень нормалей
        double normy = barycentric.lambda1 * normal1[1] + barycentric.lambda2 * normal2[1]
+ barycentric.lambda3 * normal3[1];
        double normz = barycentric.lambda1 * normal1[2] + barycentric.lambda2 *
normal2[2] + barycentric.lambda3 * normal3[2];
        double [] x = tochky[0].d3scenecoord;
        double [] y = tochky[1].d3scenecoord;
        double [] z = tochky[2].d3scenecoord;
        double wcx = barycentric.lambda1 * x[0] + barycentric.lambda2 * y[0] +
barycentric.lambda3 * z[0]; //з барицентричними при множенні дає 3д координати сцени
        double wcy = barycentric.lambda1 * x[1] + barycentric.lambda2 * y[1] +
barycentric.lambda3 * z[1];
        double wcz = barycentric.lambda1 * x[2] + barycentric.lambda2 * y[2] +
barycentric.lambda3 * z[2];
        double[] norm = MathOper.Normalize(new double[3] { normx, normy, normz });
        double[] tochka = new double[4] { wcx, wcy, wcz, 1 };
        Color color = bRDF.PixelIntensity(tochka, norm, textBox1.Text);
        return color;
    }
    private void Form1_Load(object sender, EventArgs e)
    {

    }

    private void VizualizeNatysk_Click_1(object sender, EventArgs e)
    {
        progressBar1.Value = 0;
        progressBar1.Minimum = 0;
        progressBar1.Maximum = 3;
        progressBar1.Value += 1;
        DrawWindow();
        progressBar1.Value += 1;
        DrawWindow2();
        progressBar1.Value += 1;
        Pohybka();
    }
}

```


**ДОДАТОК Д****ІЛЮСТРАТИВНА ЧАСТИНА**

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ РЕНДЕРИНГУ ДЛЯ

СИСТЕМ КОМП'ЮТЕРНОЇ ГРАФІКИ

*(Назва магістерської кваліфікаційної роботи)*

## Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки



Студент гр. 1ПІ-21м **Завальнюк Є.К.**  
 Науковий керівник - д.т.н. проф.  
**Романюк О.Н.**

Вінниця – 2022

1

Рисунок Д.1 – Слайд презентації №1

### Галузі застосування та перспективи розвитку графічних засобів

 <p>У 2021 році реалізовано понад 480 млн. графічних процесорів.</p>	 <p>У 2021 року світовий ринок смартфонів становив 1.43 млрд. шт. Прибуток від реалізації мобільних ігор досяг 90 млрд. дол.</p>	 <p>У 2021 р. продаж хромбуків становив 37 млн. одиниць</p>
 <p>У 2021 році реалізовано понад 3.5 млн. графічних станцій</p>	 <p>Оборот у галузі ігрових консолей склав у 2021 році 65 млрд. дол</p>	 <p>Відеокарти</p>
 <p>У 2021 році у світі реалізовано 276 млн. ноутбуків</p>	 <p>Архітектури AMD Fusion, Larrabee</p>	 <p>У бортових системах прогнозується широке використання комп'ютерної графіки</p>

2

Рисунок Д.2 – Слайд презентації №2

## Мета і завдання дослідження

- **Мета і завдання дослідження.** Підвищення реалістичності відтворення спекулярної складової кольору за рахунок розробки та впровадження нових двопробових функцій відбивної здатності поверхні.
- **Об'єкт дослідження** – процес формування зображень тривимірних об'єктів у системах комп'ютерної графіки.
- **Предмет дослідження** – методи та засоби рендерингу у системах комп'ютерної графіки.
- **Задачі дослідження:**
  - розробка нових двопробових функцій відбивної здатності;
  - модифікація двопробових функцій відбивної здатності поверхні;
  - дослідження моделей відбивних здатностей поверхонь;
  - розробка програмних засобів для реалізації отриманих ДФВЗ;
  - тестування отриманих функцій у розробленому засобі та аналогах.

3

Рисунок Д.3 – Слайд презентації №3

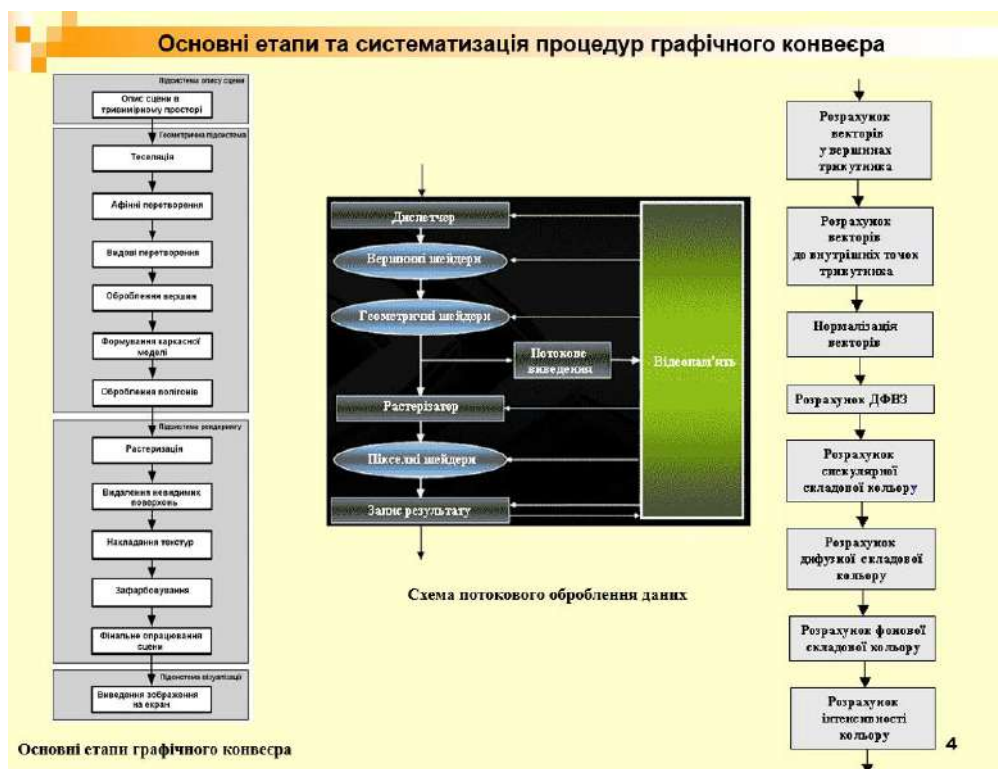
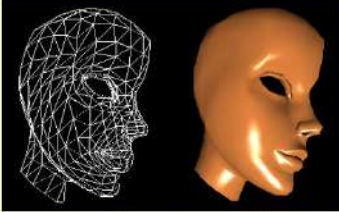
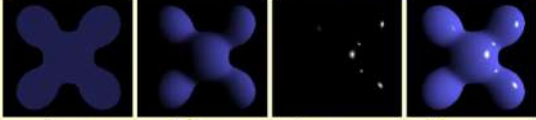
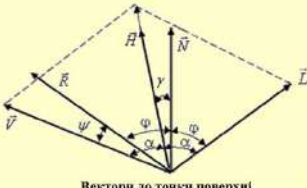


Рисунок Д.4 – Слайд презентації №4

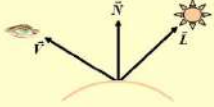
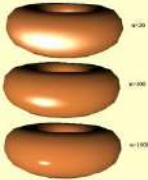
### Складові інтенсивності кольору

Фоновіа + Дифузна + Спекулярна = Інтегральна



Вектори до точки поверхні

$$I = k_a I_a + I_l (k_d (\vec{N} \cdot \vec{L}) + k_s (\vec{N} \cdot \vec{H})^n)$$



$$(\vec{V} \cdot \vec{R})^n = \cos^n \psi \quad \text{- ДФВЗ Фонга,}$$

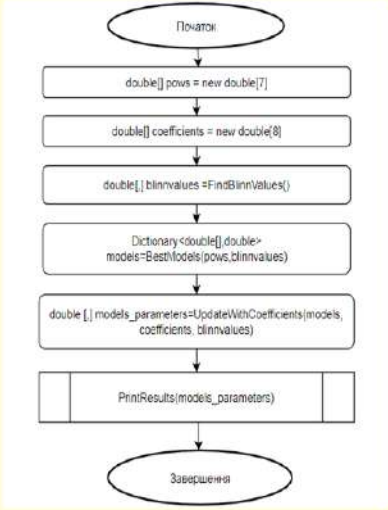
$$(\vec{N} \cdot \vec{H})^n = \cos^n \gamma \quad \text{- ДФВЗ Бліна,}$$

$$\vec{H} = \frac{\vec{L} + \vec{V}}{|\vec{L} + \vec{V}|}, \quad n \in [1, 1000].$$

$1 - (\vec{N} \cdot \vec{V})^2 - (\vec{L} \cdot \vec{N})^2 - (\vec{L} \cdot \vec{V})^2 + 2(\vec{L} \cdot \vec{N}) \cdot (\vec{N} \cdot \vec{V}) (\vec{L} \cdot \vec{V}) = (\vec{L} \times \vec{N} \cdot \vec{V})^2$

Рисунок Д.5 – Слайд презентації №5

### Розробка модифікованої моделі Шліка



Програма підбору коефіцієнтів та степенів

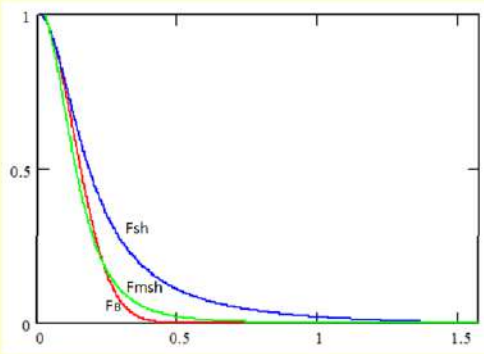
ДФВЗ Бліна,

$$F_B = \cos(x)^n$$

ДФВЗ Шліка

$$F_{SH} = \frac{\cos(x)}{n - n * \cos(x) + \cos(x)}$$

Модифікована ДФВЗ Шліка

$$F_{MSH} = \frac{2 \cos(x)}{(1 + \frac{1}{4})(n - n * \cos(x) + (1 + \frac{1}{4}) \cos(x))^2}$$


Графік ДФВЗ Бліна, ДФВЗ Шліка, модифікованої ДФВЗ Шліка (n=60)

Рисунок Д.6 – Слайд презентації №6



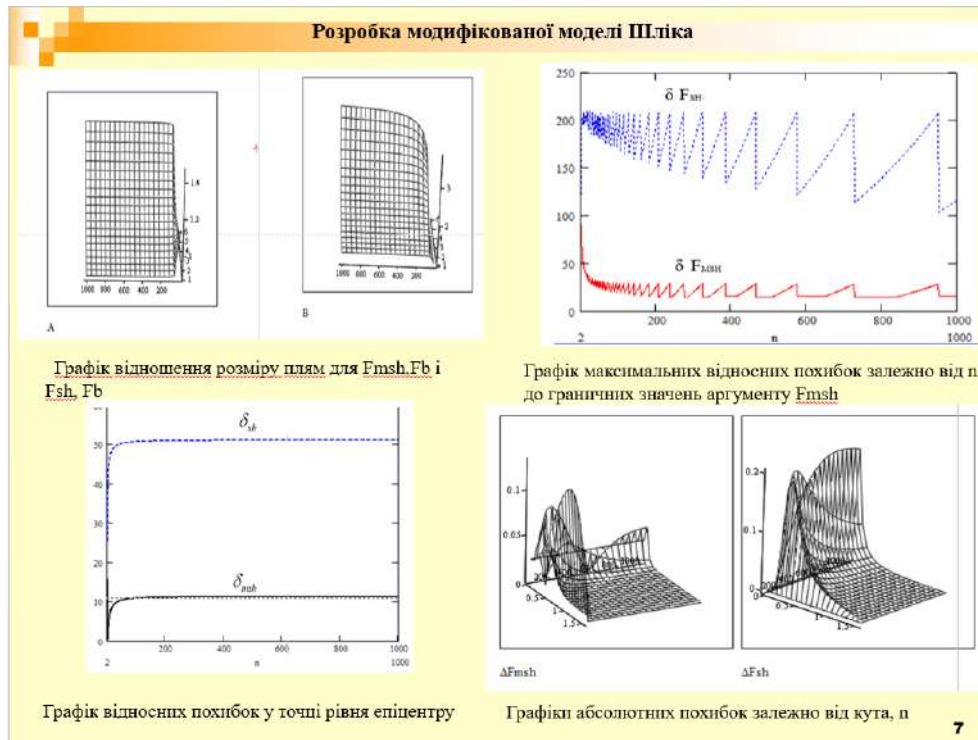


Рисунок Д.7 – Слайд презентації №7

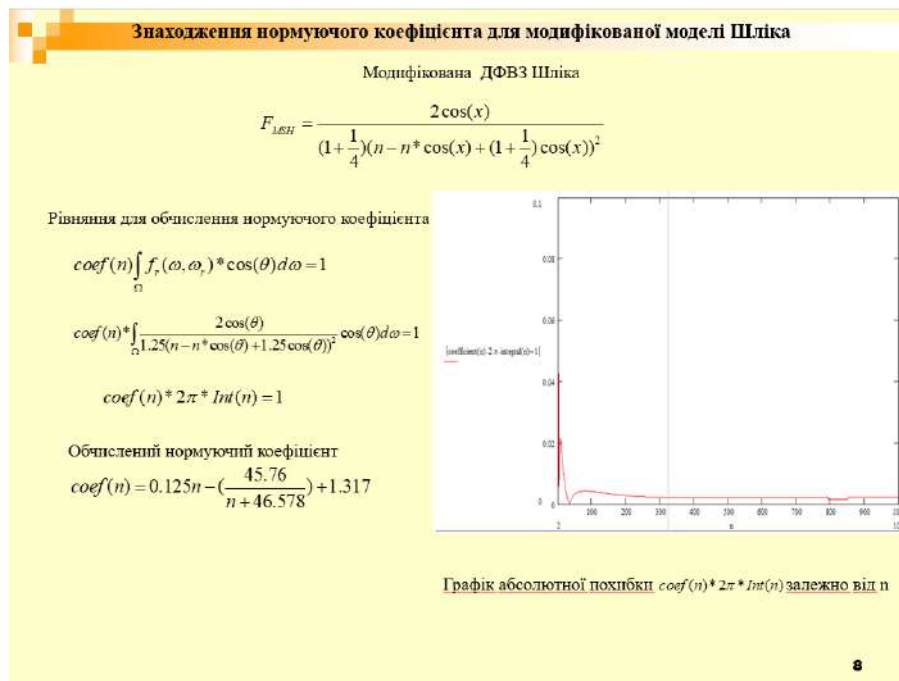


Рисунок Д.8 – Слайд презентації №8

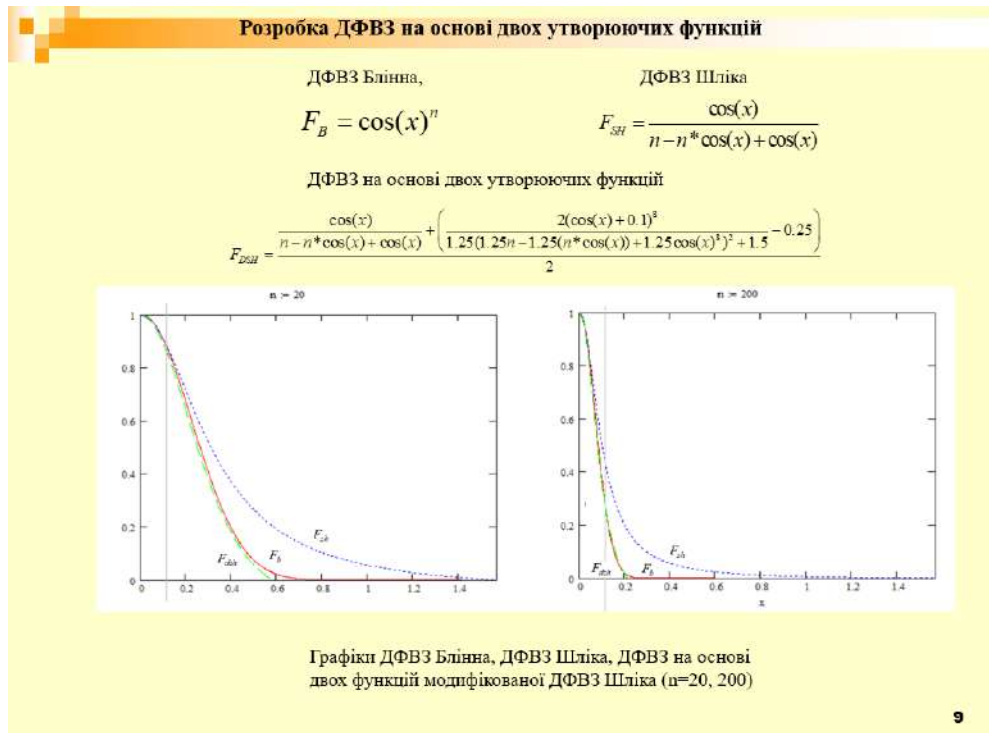


Рисунок Д.9 – Слайд презентації №9

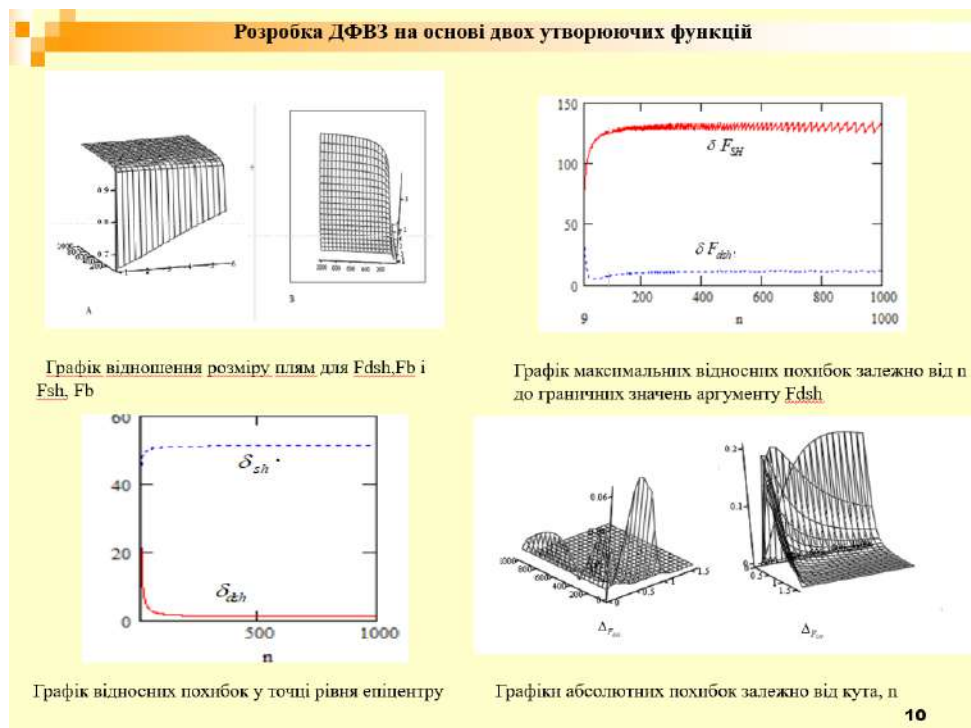


Рисунок Д.10 – Слайд презентації №10

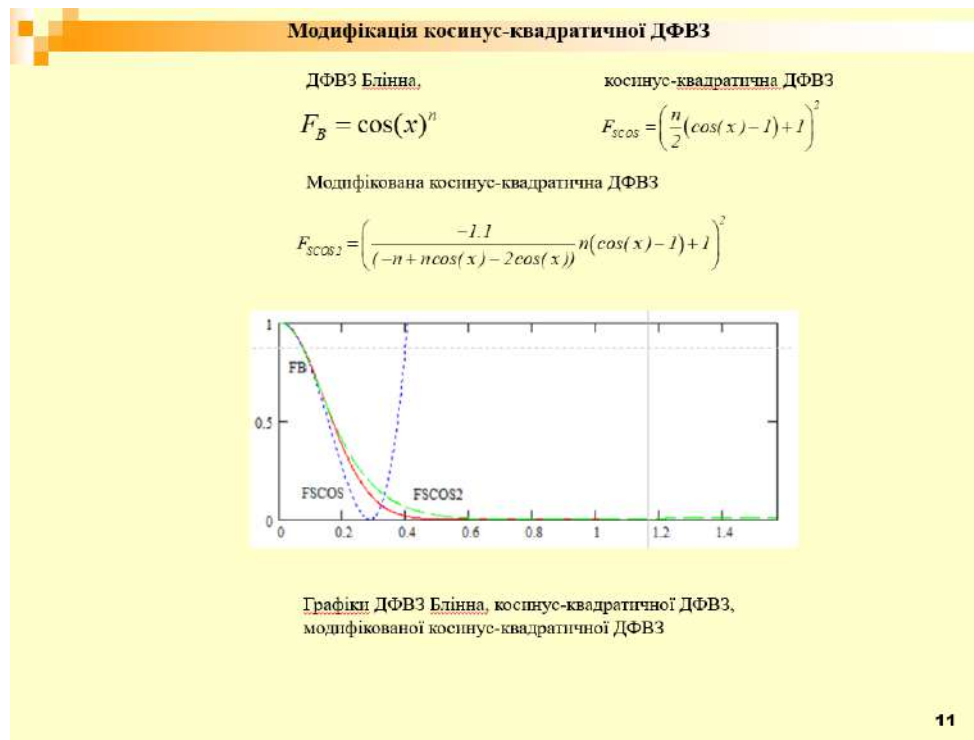


Рисунок Д.11 – Слайд презентації №11

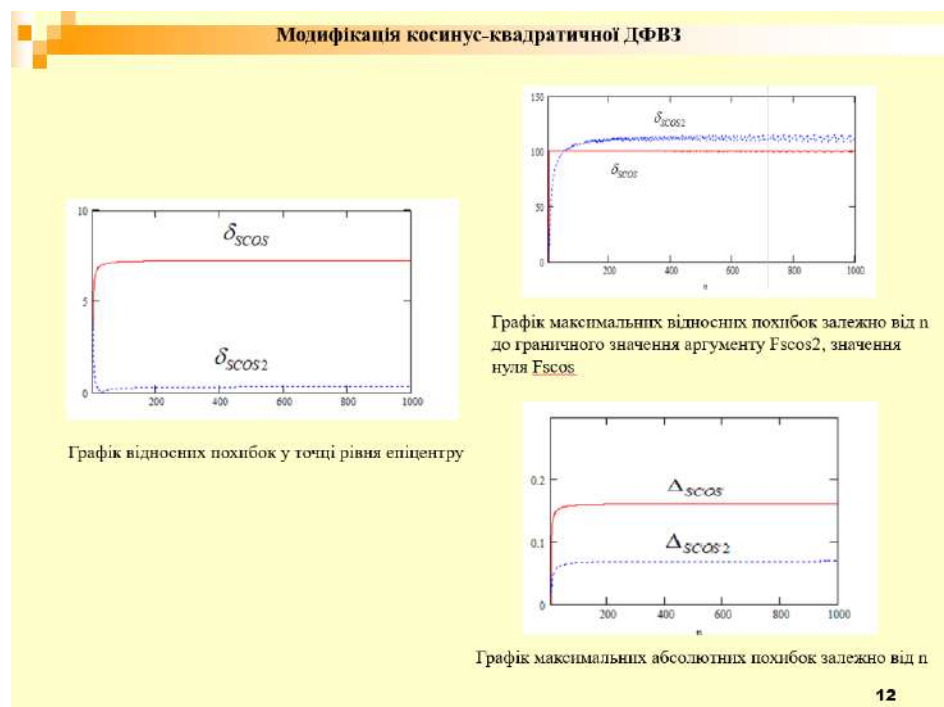


Рисунок Д.12 – Слайд презентації №12

### Знаходження нормуючого коефіцієнта для квадратичної моделі

Квадратична ДФВЗ

$$F_{QR} = 0.786n * \cos(x)^2 + (1 - 0.786n) * \cos(x)$$

Рівняння для обчислення нормуючого коефіцієнта

$$coef(n) \int_{\Omega} f_r(\omega, \omega_r) * \cos(\theta) d\omega = 1$$

$$coef(n) * \int_{\Omega} (0.786n * \cos(\theta)^2 + (1 - 0.786n) \cos(\theta)) \cos(\theta) d\omega = 1$$

$$coef(n) * 2\pi * Int(n) = 1$$

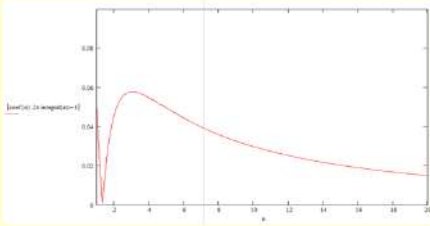
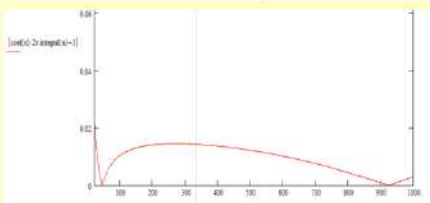
Обчислений нормуючий коефіцієнт

$n \in [1, 20]$

$$coef(n) = 0.212(1.175(n + (1.265 + (n(-0.023 \operatorname{tg}(-0.0008n))))))$$

$n \in [21, 1000]$

$$coef(n) = 0.255n$$

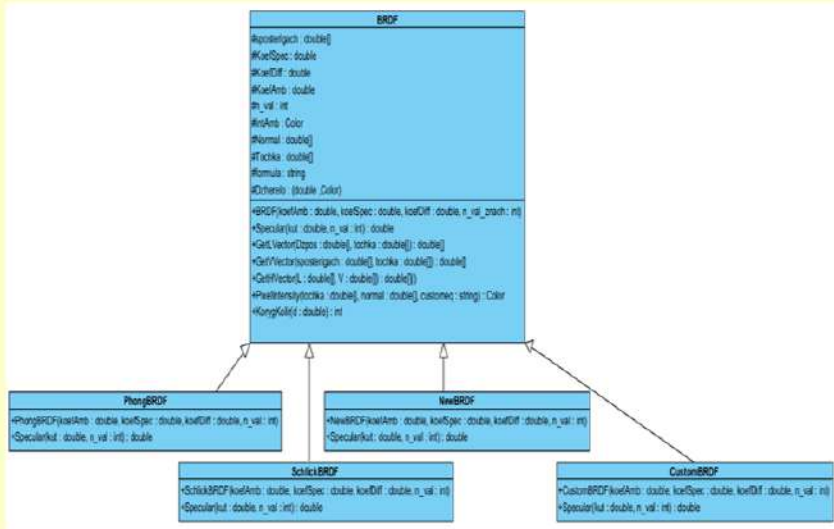



Графіки абсолютної похибки  $coef(n) * 2\pi * Int(n)$  залежно n

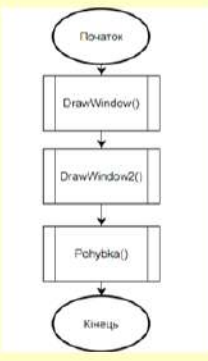
13

Рисунок Д.13 – Слайд презентації №13

### Програмний засіб для порівняння моделей відбиття



**Основні класи програмного засобу**

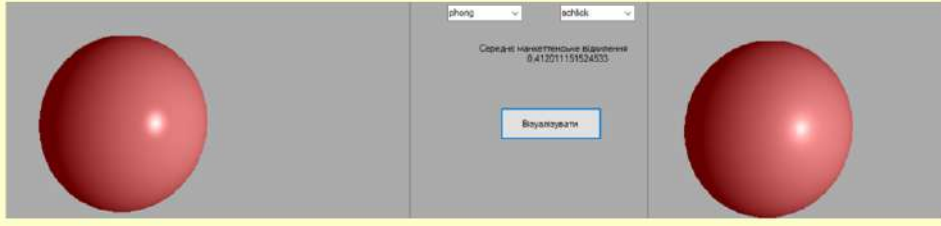


**Загальна схема роботи засобу**

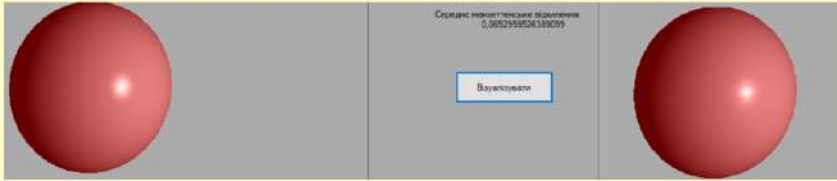
14

Рисунок Д.14 – Слайд презентації №14

**Тестування розроблених функцій**



Порівняння застосування ДФВЗ Фонга, Шліва у розробленому засобі (n=50)



Порівняння застосування ДФВЗ Фонга, модифікованої ДФВЗ Шліва у розробленому засобі (n=50)

**15**

Рисунок Д.15 – Слайд презентації №15

**Тестування розроблених функцій**



Порівняння застосування ДФВЗ Фонга, ДФВЗ на основі утворюючих функцій (режим вводу формули, n=50)



Порівняння застосування ДФВЗ Фонга, косинус-квадратичної ДФВЗ (режим вводу формули, n=50)

**16**

Рисунок Д.16 – Слайд презентації №16

**Тестування розроблених функцій**

```

float spec = pow(max(dot(N, H), 0.0), shininess);
float specschlick = (max(dot(N, H), 0.0) / (n + n*max(dot(N, H), 0.0)*max(dot(N, H), 0.0)));
float rspec = (1.0*max(dot(N, H), 0.0)) / (1.25*pow((n + n*max(dot(N, H), 0.0) + 1.0), 2.0));

```

Код у OneShader для порівняння ДФВЗ Блінна, ДФВЗ Шліка, модифікованої ДФВЗ Шліка

	n=10	20	50	200	500
Блінн					
Шлік					
Нове					

Порівняння візуалізації куль залежно від ДФВЗ, n у OneShader (колаж)

17

Рисунок Д.17 – Слайд презентації №17

**Тестування розроблених функцій**

```

float nval=200.0;
//float spec = pow ( hn, nval );
//float specschlick=hn/(nval-nval*hn+hn);
float specnev=2.0*hn/(1.25*pow(nval-nval*hn+1.25*hn, 2.0));

```

Код у ShaderToy для порівняння ДФВЗ Блінна, ДФВЗ Шліка, модифікованої ДФВЗ Шліка

			Блінн
			Шлік
			Нове

Порівняння візуалізації ламп залежно від ДФВЗ, n у ShaderToy (колаж)

18

Рисунок Д.18 – Слайд презентації №18

**Тестування розроблених функцій**

```
float nval=200.0;
//float spec = pow ( hn, nval );
//float specslick=hn/(nval-nval*hn+hn);
float specnew=2.0*hn/(1.25*pow(nval-nval*hn+1.25*hn,2.0));
```

Код у ShaderToy для порівняння ДФВЗ Бліна, ДФВЗ Шліка, модифікованої ДФВЗ Шліка

Порівняння візуалізації торів залежно від ДФВЗ, n у ShaderToy (колаж)

19

Рисунок Д.19 – Слайд презентації №19

**Наукова новизна результатів. Практичне значення одержаних результатів**

- **Наукова новизна результатів**, отриманих у магістерській кваліфікаційній роботі:
- Запропоновано модифікацію функції Шліка, яка відрізняється від відомої використанням другої степені та поправочних коефіцієнтів, що дає можливість підвищити точність визначення спекулярної складової кольору та реалістичного відтворення як епіцентру відблиску, так і його блюмінгу.
- Вперше запропоновано для визначення двопроменевої функції відбивної здатності використання не однієї, а суми двох утворюючих функцій, що дає можливість підвищення точності визначення спекулярної складової кольору і, як наслідок, більш реалістично відтворити відблиски.
- Запропоновано модифікації двопромених функцій Шліка, квадратичної функції, які відрізняються від відомих введенням поправочних коефіцієнтів, що дало можливість фізично коректно відтворювати відблиски на поверхнях тривимірних об'єктів.
- **Практичне значення одержаних результатів**. Практичне значення полягає у розробці на основі проведених теоретичних досліджень нових алгоритмів і програм для задач рендерингу.
- **Публікації** Результати досліджень опубліковано у 5 наукових працях, з них стаття у фаховому журналі, 4 тези доповідей на міжнародних конференціях.

20

Рисунок Д.20 – Слайд презентації №20

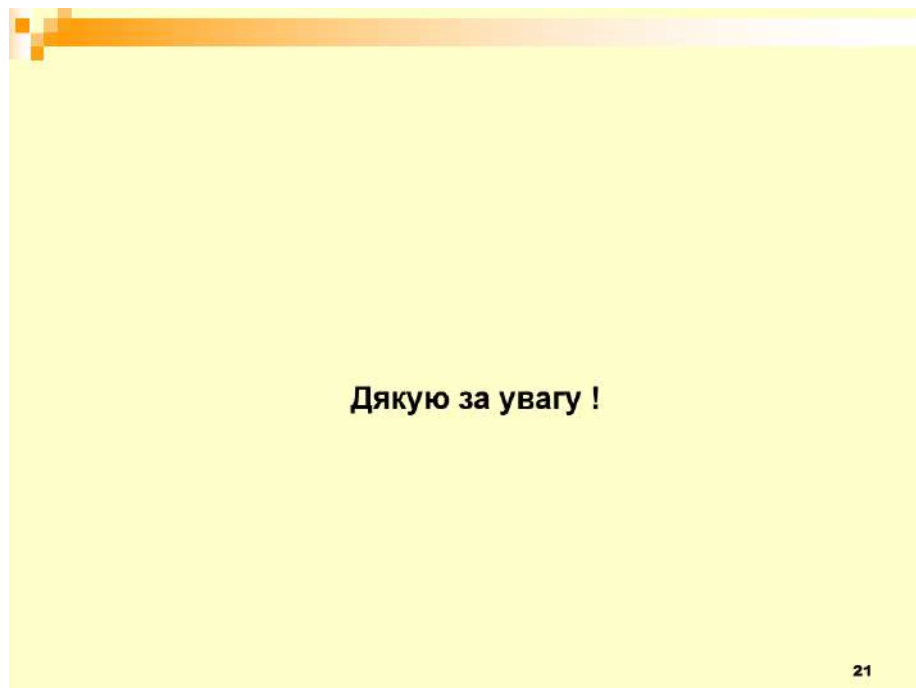


Рисунок Д.21 – Слайд презентації №21



## ВІДГУК НАУКОВОГО КЕРІВНИКА

на магістерську кваліфікаційну роботу студента гр. ІІІ-21м Завальнюка Є. К.

«Розробка методів і програмних засобів рендерингу

для систем комп'ютерної графіки»

Вимоги до продуктивності графічних систем і пристроїв неухильно зростають у зв'язку з необхідністю вирішувати все більш складні та обчислювально ємні задачі формування реалістичних зображень у реальному часі. Формування просторових зображень є складним обчислювальним процесом, який складається з багатьох етапів, основним серед яких є етап рендерингу, на якому моделюються світлові ефекти. Врахування великої кількості параметрів при визначенні інтенсивностей кольору робить цей етап найтрудомісткішим серед інших етапів формування просторових сцен, що обумовлює необхідність розробки нових методів та засобів. У зв'язку з цим тема магістерської кваліфікаційної роботи Завальнюка Є. К., в якій розглядаються питання підвищення продуктивності та реалістичності засобів формування тривимірних зображень, є актуальною.

В роботі розроблено низку моделей відбивної здатності поверхонь.

Розроблено нову модифікацію функції Шліка, яка відрізняється від відомої використанням другої степені, що дає можливість підвищити реалістичність відтворення як епіцентру відблиску, так і його блюмінгу.

Запропоновано нову двопроменеву функцію відбивної здатності, в якій використовується сума двох утворюючих функцій. Це дало можливість підвищення точності визначення спекулярної складової кольору

Запропоновано фізично-коректні моделі відбивної здатності поверхонь. Розроблено алгоритми та програмні засоби для реалізації розроблених в роботі методів.

Пояснювальна записка відображає всі етапи проведення досліджень.

За тематикою МКР опубліковано 6 наукових праць, у тому числі: 1 – у фаховому виданні, 5 – у матеріалах конференцій. Робота має достатню апробацію.

Вважаю, що магістерська кваліфікаційна робота виконана на високому науково-технічному рівні, з дотриманням встановлених вимог. Робота заслуговує оцінки «А», кількість балів - 95 а її автор Завальнюк Є. К. – присвоєння ступеня вищої освіти магістр, спеціальність 121 – «Інженерія програмного забезпечення», освітня програма «Інженерія програмного забезпечення».

Науковий керівник

д.т.н., проф. кафедри ІІЗ



Романюк О.Н.,

## ВІДГУК ОПОНЕНТА

на магістерську кваліфікаційну роботу студента гр. ІІІ-21м Завальнюка Є. К.  
«Розробка методів і програмних засобів рендерингу для систем комп'ютерної графіки»

Системи формування реалістичних зображень повинні забезпечувати передачу усієї сукупності властивостей об'єктів: розташування об'єктів в сцені, колір, об'ємність, текстура поверхні об'єкта і т.д., що в свою чергу висуває підвищені вимоги до їх продуктивності та реалістичності. Виникає задача вдосконалення існуючих і розробки нових методів і засобів, які б забезпечували формування реалістичних зображень у реальному часі. У зв'язку з цим актуальність роботи Завальнюка Є.К. не викликає сумнівів.

У роботі проаналізовано сучасний стан проблеми та визначено основні напрямки дослідження.

Запропоновано модифікацію функції Шліка, яка відрізняється від відомої використанням другої степені та поправочних коефіцієнтів, що дає можливість підвищити точність визначення спекулярної складової кольору та реалістичного відтворення як епіцентру відблиску, так і його блюмінгу. Вперше запропоновано для визначення двопроменевої функції відбивної здатності використання не однієї, а суми двох утворюючих функцій, що дає можливість підвищення точності визначення спекулярної складової кольору і, як наслідок, більш реалістично відтворити відблиски. Запропоновано модифікації двопромених функцій Шліка, квадратичної функції, які відрізняються від відомих введенням поправочних коефіцієнтів, що дало можливість фізично коректно відтворювати відблиски на поверхнях тривимірних об'єктів.

Розроблено алгоритми та програмні засоби для реалізації розроблених в роботі методів.

Пояснювальну записку написано технічно грамотно, відповідно до встановлених вимог. Графічна частина гармонійно доповнює теоретичний матеріал.

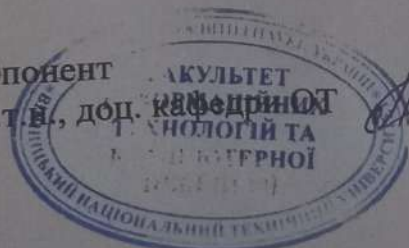
За тематикою дослідження опубліковано 6 наукових праць, у тому числі: 1 – у фаховому виданні, 5 – у матеріалах конференції. Робота має достатню апробацію.

Зауваження до магістерської кваліфікаційної роботи: не обґрунтовано вибір тестових фігур.

Зазначене зауваження не зменшує цінність роботи та не стосуються основних її положень.

Вважаю, що магістерська кваліфікаційна робота виконана відповідно до завдання із дотриманням встановлених вимог. Робота заслуговує оцінки «А», кількість балів - 90 а її автор Завальнюк Є. К. – присвоєння ступеня вищої освіти магістр, спеціальність 121 – «Інженерія програмного забезпечення», освітня програма «Інженерія програмного забезпечення».

Опонент  
К.Т.П., доц. кафедри  
ТЕХНОЛОГІЙ ТА  
ПРОГРАМНОЇ



Савицька Л.А.