Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів трасування променів для досягнення мети покращення швидкодії візуалізації 3D-простору у галузі комп'ютерної графіки
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-1____
_____Засименко Г.Р._____
(прізвище, ініціали)

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

Освітньо-наукової програми
(тип програми)

__Інженерія програмного забезпечення____
(повна назва освітньої програми)

Керівник ___к.т.н. доц. Назаров О.С._____
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри, проф _____ З.В.Дудар
2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет_Комп'ютерних наук_____
Кафедра_Програмної інженерії_____
Рівень вищої освіти – другий (магістерський)

Спеціальність_121 – Інженерія програмного забезпечення_____
(код і повна назва)
Тип програми_освітньо-наукова програма_____

Освітня програма_Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Засименко Георгію Романовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів трасування променів для досягнення мети покращення швидкодії візуалізації 3D-простору у галузі комп'ютерної графіки
затверджена наказом університету від "___" _____20 ___ р. № _____
2. Термін подання студентом роботи до екзаменаційної комісії
__ червня 2020 р.
3. Вихідні дані до роботи алгоритми візуалізації тривимірної геометрії, алгоритми генерації зображення у реальному часі, пояснювальна записка. Використовувати ОС Windows, середовище проектування._____
4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів трасування променів для генерації тіней, відображень, прозорості та глобального освітлення.

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Назаров О.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Огляд літератури у галузі	02.03.2020	виконано
2	Вибір технології	15.03.2020	виконано
3	Розробка підходу експериментів	12.03.2020	виконано
4	Підготовка зразків даних	23.03.2020	виконано
5	Проведення експериментів	29.03.2020	виконано
6	Підготовка пояснювальної записки	12.05.2020	виконано
7	Підготовка презентації та доповіді	17.05.2020	виконано
8	Попередній захист	18.05.2020	виконано
9	Нормоконтроль, рецензування	20.05.2020	виконано
10	Занесення диплома в електронний архів	21.05.2020	виконано
11	Допуск до захисту у зав. кафедри	21.05.2020	виконано

Дата видачі завдання _____ _____ 2020 р.

Студент _____
(підпис)

Керівник роботи _____ __доц. Назаров О.С._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Атестаційна робота магістра: 99 с., 41 рис., 9 табл., 22 джерел,

ТРИВИМІРНА ВІЗУАЛІЗАЦІЯ, ТРАСУВАННЯ ПРОМЕНІВ, РАСТЕРИЗАЦІЯ, КАДР, ВІДОБРАЖЕННЯ, ПРОЗОРІСТЬ, TIHI, DXR, UE4

Метою роботи є методи трасування променів, алгоритми генерації зображення, проектування та розробка програмної системи генерації зображення для збору даних щодо швидкодії.

Методи дослідження базуються на відкладеному підході до візуалізації тривимірної геометрії, графічному API D3D12 та фреймворку UE4, який дозволяє апаратне прискорення методу трасування променів.

У результаті роботи розглянуто методи трасування променів, алгоритми генерації зображення та розроблено програмна система генерації зображення для аналізу даних щодо якості отриманого зображення та швидкодії методу.

ABSTRACT

3D RENDERING, RAY TRACING, RASTERIZATION, FRAME, REFLECTIONS, TRANSLUCENCY, SHADOWS, DXR, UE4

The purpose of the work is ray tracing methods, image generation algorithms, design and development of an image generation software system for data collection on performance.

The research methods are based on a deferred approach to three-dimensional geometry visualization, graphical API D3D12 and UE4 framework, which allows hardware acceleration of the ray tracing method.

As a result, the methods of ray tracing, image generation algorithms are considered and a software system of image generation for data analysis on the quality of the obtained image and the speed of the method is developed.

ЗМІСТ

Перелік умовних скорочень 6
Вступ 7
1 Огляд і аналіз наукової і патентної літератури 9
1.1 Методи візуалізації тривимірної геометрії 9
1.2 Методи та проблеми генерації зображення у реальному часі 19
1.3 Принципи роботи апаратного прискорення 28
1.3.1 Растеризація 28
1.3.2 Трасування променів 29
1.4 Науково-технічна задача 31
2 Опис проведених теоретичних та експериментальних досліджень 32
2.1 Принципи роботи модифікованого алгоритму генерації зображення 32
2.2 Методи освітлення 35
3 Аналіз результатів досліджень 40
4. Опис розробленої програмної системи 50
4.1 Вибір середовища реалізації програмного засобу 50
4.2 Інтерфейс розробленого програмного засобу 51
Висновки 55
Перелік джерел посилання 56
Додаток А Лістинг програмного коду 58
Додаток Б Апробація результатів роботи 75
Додаток В Слайди презентації 83

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

BDRF – Bidirectional reflectance distribution function

API – Application programming interface

SSLR – Screen space local reflections

PBR – Physics based rendering

BVH – Bounding volume hierarchies

HDR – High dynamic range

LDR – Low dynamic range

UE4 – Unreal Engine 4

ВСТУП

Розвиток комп'ютерної індустрії визначає необхідність ефективного забезпечення обчислювальних процесів графічної візуалізації, що пропонують для користувачів найвищий рівень інформативності та наочності відображення процесів, що відбуваються у різних системах автоматики. Основні напрямки досліджень у цій галузі пов'язані з радикальним підвищенням продуктивності та реалістичності візуалізації. Протягом останнього десятиліття завдання реалістичної візуалізації і генерації статичних та динамічних зображень об'єктів і сцен навколишнього оточення комп'ютерними системами як і раніше зберігає свою актуальність.

Сферами застосувань візуалізації є як традиційні для комп'ютерною графіки галузі, так і практично будь-які інші задачі, що виникають у науковій, дослідницькій та практичній діяльності людини, що потенційно вимагають на сьогоднішній день якісно нового рівня відображення результатів комп'ютерної симуляції і моделювання, ступеня занурення у віртуальну реальність і добротності візуальних почуттів від взаємодії з нею [1]. Сюди можна віднести:

– системи комп'ютерної графіки, синтезу та обробки зображень;

– системи тривимірного комп'ютерного моделювання а різних галузях;

– системи віртуальної і розширеної реальності;

– багатоцільові симулятори та імітатори;

– системи медичної діагностики та автоматизації виконання операцій;

– системи візуалізації багатовимірних і просторових наукових даних в астрономії, метеорології, географії, геології та інших наукових галузях;

– системи моделювання та візуалізації в хімії, фізиці, кристалографії;

– індустрія розваг.

Метод трасування променів був першим кроком до загальної тривимірної візуалізації, тому як є дуже простим та водночас дає найкращі результати якості зображення. Але для того, щоб згенерувати один кадр навіть сучасному «залізу»

необхідно від секунд до декількох днів, що є неприпустимим для систем реального часу, які мусять генерувати кадр до 33.3 мс.

З другої сторони є метод растреризації, який використовується у всіх системах генерації зображення реального часу, тому як дозволяє генерувати кадр, в якому міститься велика кількість геометрії та джерел освітлення за допустимий час, тому як має апаратну підтримку на відеокартах, що буквально є їх основною функцією. Але з розвитком графіки росте потреба в фото реалістичності зображення. Деяких ефектів, таких як відображення, прозорість та непряме освітлення методом растеризації добитися вкрай важко. Те, с чим метод трасування променів немає проблем.

Нещодавно компанія NVIDIA зробила перший крок у привнесенні методу трасування променів до систем візуалізації у реальному часі додавши апаратну підтримку його до своїх відеокарт. На час написання роботи вже було анонсовано ігрові консолі нового покоління з апаратною підтримкою трасування променів, що безумовно призведе до появи великої кількості систем візуалізації реального часу з підтримкою цього методу.

Метод трасування променів навить з апаратною підтримкою все ще складний в обчислюванні, але його можна впровадити у існуючі алгоритми генерації зображення в ті міста, де метод растеризації має складнощі. Необхідно зробити дослідження, як метод трасування променів впливає на сучасні алгоритми генерації зображення, які з етапів привносять більший вклад в якість зображення меншою ціною, щоб зрозуміти.

1 ОГЛЯД І АНАЛІЗ НАУКОВОЇ І ПАТЕНТНОЇ ЛІТЕРАТУРИ

1.1 Методи візуалізації тривимірної геометрії

Растеризація є основним методом отримання зображення з геометрії, яка складається з полігонів у сучасній комп’ютерній графіці. За допомогою цього методу знаходяться фрагменти(пікселі) зображення, які перекриваються полігоном (див. рис. 1.1). В якості полігонів будуть розглянуті трикутники, так як вони являються найпоширенішим форматом в індустрії, а також тому що апаратне прискорення растеризації працює лише з трикутниками [2].

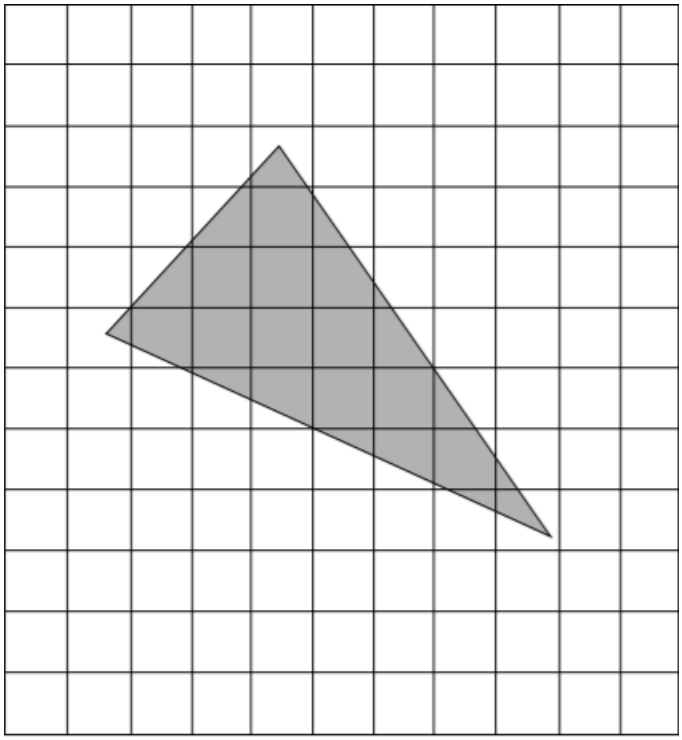

Рисунок 1.1 – Перетин трикутника з фрагментами

Для того щоб знайти фрагменти, які покриті трикутником необхідно знайти чи перетинається прямокутник фрагменту з трикутником, також можна визначити чи лежить центр фрагменту у трикутнику. Частина фрагментів буде лежати всередині трикутника, частина буде поза трикутником, а частина буде

перетинатись з трикутником частково (див. рис. 1.2). Якщо декілька трикутників перетинають одні й ті ж самі фрагменти це може призвести до погіршення швидкодії.

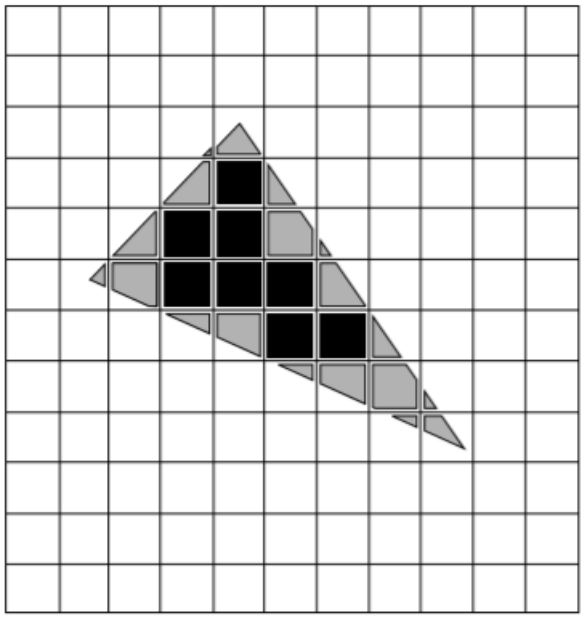

Рисунок 1.2 – Перетин трикутника з фрагментами

У серці методу растеризації лежить алгоритм сканування рядків. Трикутники опуклі, тому трикутник перетинає горизонтальний рядок фрагментів найбільш в один безперервний сегмент. Тобто для кожного рядка, що перетинає трикутник можна представити перетин тільки як мінімальне та максимальне значення x[3]. Тобто представлення трикутника буде складатись з набору мінімальних та максимальних значень x для кожного рядку, який перетнув трикутник. Опуклість трикутника так же означає, що набір рядків, які перетинають трикутник неперервні по y. Так для трикутника буде мінімальне та максимальне значення y. Приклад набору рядків, які перетинають трикутник представлено на рисунку 1.3.

Крайні ліві та праві фрагменти можуть перетинати трикутник частково, так як грань трикутника може бути не на границі фрагменту. Так само й для крайніх верхніх та нижніх. Останні фрагменти для трикутника будуть лежати повністю у трикутнику. Щодо роботи з частковими фрагментами – це системне рішення.

Можна взагалі їх відкинути, а можна залишати лише ті фрагменти, центр яких лежить у трикутнику, як на показано на рисунку 1.4.

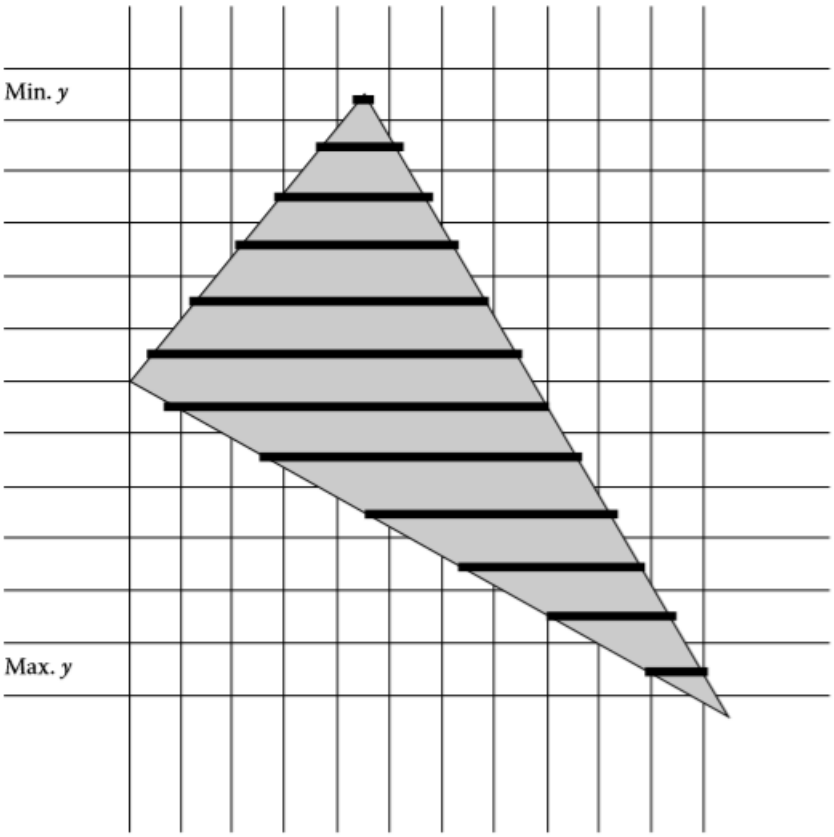

Рисунок 1.3 – Алгоритм сканування рядків

Але ми працюємо з тривимірною графікою, тобто спочатку до растризації необхідно з перейти від набору тривимірних трикутників до двовимірного. Ключовим компонентом для того, щоб кінцеве двовимірне зображення почувалось як тривимірна сцена з глибинною є перетворення координат [4].

Всього є чотири важливих систем координат:

– система локальних координат об’єкта;

– світова система координат;

– система координат камери;

– система координат зображення.

Координати кожної моделі сцени спочатку знаходяться в унікальній для кожного об’єкта системі координат. Для того, щоб на сцені можна було розмістити

багато різних об’єктів необхідно усі ці об’єкти трансформувати до єдиної системи координат – мирової (хоча можна представити сцену як один об’єкт у своєму локальному просторі).

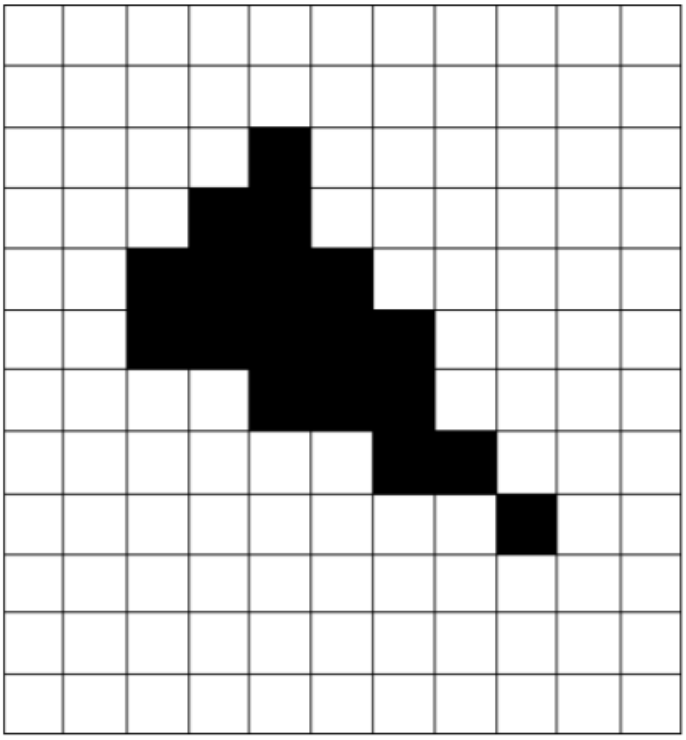

Рисунок 1.4 – Результат роботи растеризації трикутника

Для того, щоб можна було контролювати які об’єкти будуть знаходитися на зображенні, необхідно перевести мирові координати сцени у систему координат відносно камери [5].

Далі є останній етап перспективного перетворення тривимірних вершин у системі координат камери до двовимірних координат зображення.

Як було сказано вище багато трикутників можуть бути растризовані в один фрагмент. Для того, щоб сцена була реалістичної необхідно, щоб віддалені об’єкти були перекриті ближчими об’єктами. Для цього доречно використовувати буфер глибини. На етапі растеризації заповнюється буфер того ж самого розміру, що й зображення, але замість кольору кожен фрагмент зберігає у собі значення глибини трикутника.

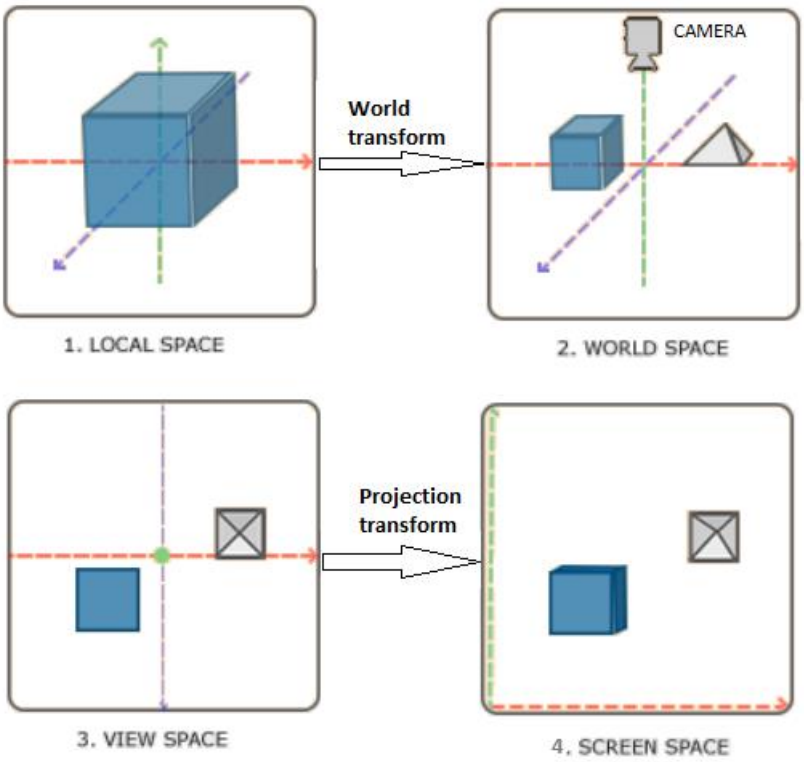

Рисунок 1.5 – Трансформації систем координат

Тобто кожен раз, коли трикутник растеризується у один з фрагментів відповідне значення глибини з буфера порівнюється з поточним значенням та якщо поточне значення менше, то це значення записується у буфер глибини та трикутник успішно растеризується у цей фрагмент, якщо значення більше, то фрагмент відкидається, та в буфер нічого не записується (див. рис. 1.6).

Цей метод використовується у всіх системах рендерингу реального часу завдяки його великій швидкодії, апаратної підтримки графічними прискорювачами [6]. Цей метод накладає обмеження до моделей геометричних об’єктів, адже працює лише с трикутниками.

Немає сенсу растеризувати геометрію, яка не входить до поля зору, тому її можна відкинути (frustrum culling)[7]. Геометрія, яка лежить на межі зору відрізається (frustrum clipping)[7].

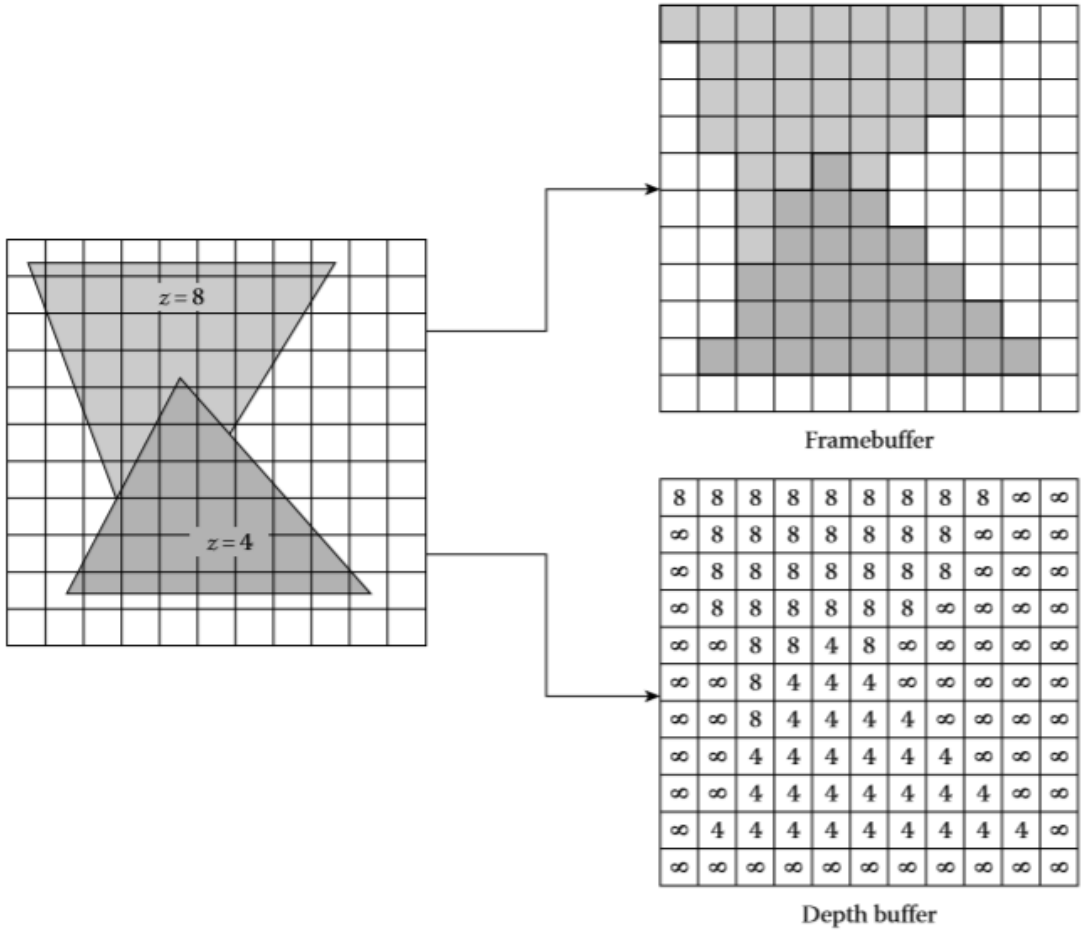

Рисунок 1.6 – Робота методу буферизації глибини

Демонстрація роботи цієї оптимізації представлена на рисунку 1.7.

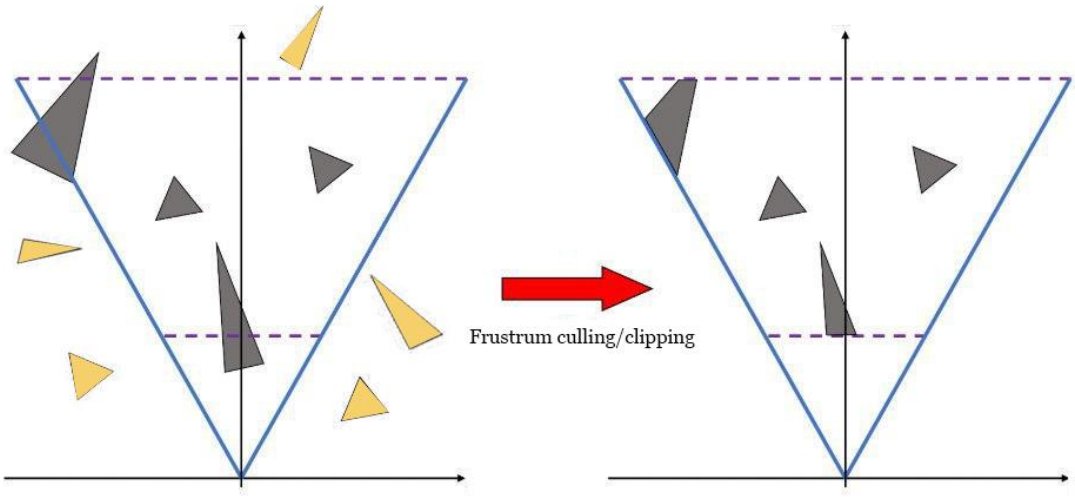

Рисунок 1.7 – Демонстрація роботи frustrum culling та frustrum clipping

Геометрія, яка лежить поза зоною видимості не растеризується зовсім, а та, що лежить на границі – відрізається по границі.

Наступним методом отримання зображення є трасування променів.

Концептуально метод трасування променів легший за растеризацію. З позиції віртуальної камери у мировій системі координат посилаються промені через пікселі віртуального зображення (див. рис. 1.8). Позиція зображення визначається параметрами камери, тобто, якщо кут обзору менший, то зображення буде далі від камери та навпаки [8]. Таким образом перспективна є базовою частиною алгоритму та немає потреби робити додаткове перспективне перетворення кожного об’єкту.

У серці цього алгоритму лежить параметричне рівняння променю:

$$P(t) = (1 - t)A + tB \tag{1.1}$$

де A та B задані точки у тривимірному просторі, t – параметр лінійної інтерполяції між двома точками, P(t) –точка, яка лежить на промені.

Частіше використовуються таке представлення (див. рис. 1.9):

$$P(t) = O + td \tag{1.2}$$

де O – точка у тривимірному просторі, яка характеризує початок променю, d – тривимірний вектор, який характеризує направлення променю.

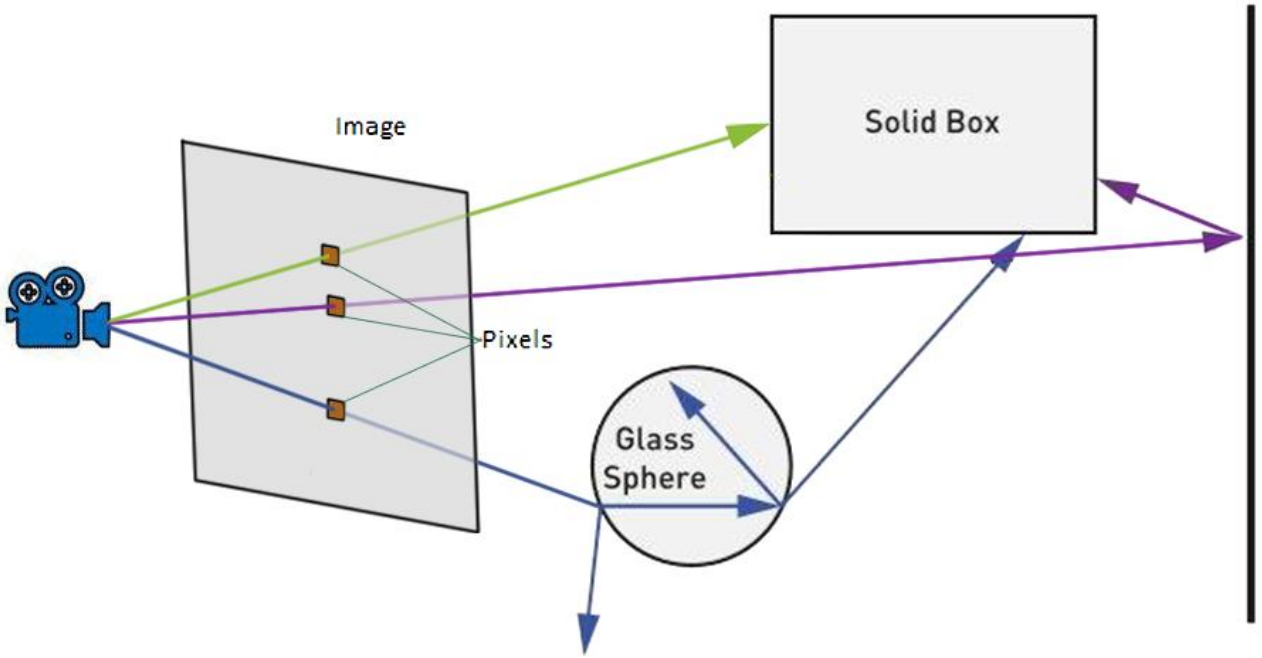

Рисунок 1.8 – Метод трасування променів

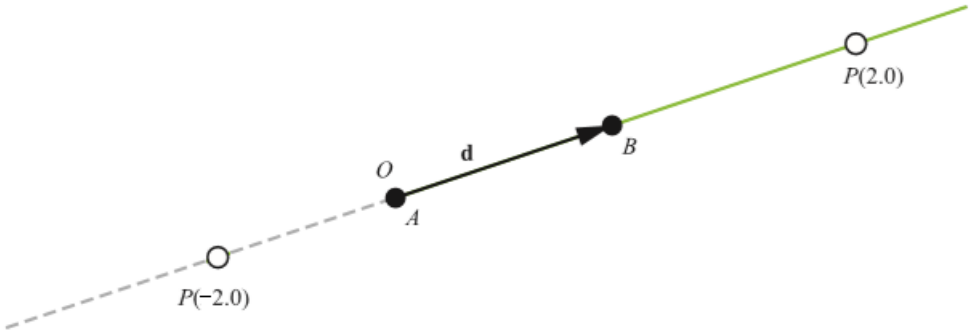

Рисунок 1.9 – Представлення променю через початок та направлення

Вектор d може бути не нормалізованим, але набагато зручніше мати його нормалізованим, тому що нерідко необхідно розрахувати косинус кута між векторами за допомогою скалярного добутку векторів та якщо d нормалізований, то параметр t характеризує відстань до точки, наприклад перетину, у логічній величині системи, наприклад t = 1 є відстань в 1 см.

Найважливіша операція методу трасування променів – це знаходження точки перетину з геометричними поверхнями. Як було сказано вище трикутники – найбільш розповсюджений примітив з яких складається поверхня [9].

Трикутник описується положенням у тривимірному просторі його трьох точок P0, P1 та P2. Нормаль N поверхні у якій лежить трикутник за допомогою формули:

$$N = (P_1 - P_0) \times (P_2 - P_0) \tag{1.3}$$

Поверхня L представляється, як чотиривимірний вектор:

$$L = (N, -N \cdot P_0) \tag{1.4}$$

Точка P перетину визначається за формулою (1.2), де

$$t = \frac{L \cdot S}{L \cdot V} \tag{1.5}$$

Тепер є проблема вирішення чи є точка P у середині трикутника. Розрахуємо барицентричні координати P:

$$P = \omega_0 P_0 + \omega_1 P_1 + \omega_2 P_2 \tag{1.6}$$

де ω_0, ω_1 та ω_2 – коефіцієнти барицентричних координат та $\omega_0 + \omega_1 + \omega_2 = 1$.

Якщо замінити ω_0 на $1 - \omega_0 - \omega_0$ отримаємо:

$$\begin{gathered} P = (1 - \omega_1 - \omega_2)P_0 + \omega_1 P_1 + \omega_2 P_2, \\ R = P - P_0, \\ Q_1 = P_1 - P_0, \\ Q_2 = P_2 - P_0 \end{gathered} \tag{1.7}$$

Вирішивши систему для ω_1 та ω_2 можна визначити положення точки відносно трикутника

$$\begin{bmatrix} Q_1^2 & Q_1Q_2 \\ Q_1Q_2 & Q_2^2 \end{bmatrix} [\omega_1 \ \omega_2] = [R \cdot Q_1 \ R \cdot Q_2] \quad (1.8)$$

Точка R знаходиться всередині трикутника, якщо всі три ваги ω_0, ω_1 та ω_2 є не негативні. Оскільки $\omega_0 = 1 - \omega_1 - \omega_2$, то $\omega_1 + \omega 2 <= 1$.

Сучасні геометричні моделі можуть складатися з тисяч трикутників, а сцена візуалізації с сотень об'єктів. Для того, щоб поліпшити швидкодію алгоритму використовуються BVH [10]. Ключова ідея алгоритму в тому, щоб заключити геометричні об'єкти в обсяг, який описується найпростішої фігурою з якою можна розрахувати перетин. Спрощена схема алгоритму представлена на рисунку 1.10. Тобто тут об'єкти розподіляються на дві групи та ці дві групи включається у ще одну групу. Якщо промінь не влучив в найвищий обсяг, то він точно не влучить нікуди, якщо влучив то перевіряється нижчі обсяги та нарешті, якщо влучив в один з обсягів, то вже знаходиться точка перетину з геометричними об'єктами.

Деякі об'єкти можуть міститися у декількох об'ємах, які можуть перетинатися і вони не впорядковані – вони просто обоє всередині. Отже це дерево не має поняття впорядкування, тобто не має значення від якого об'єму промінь дійшов до об'єкту.

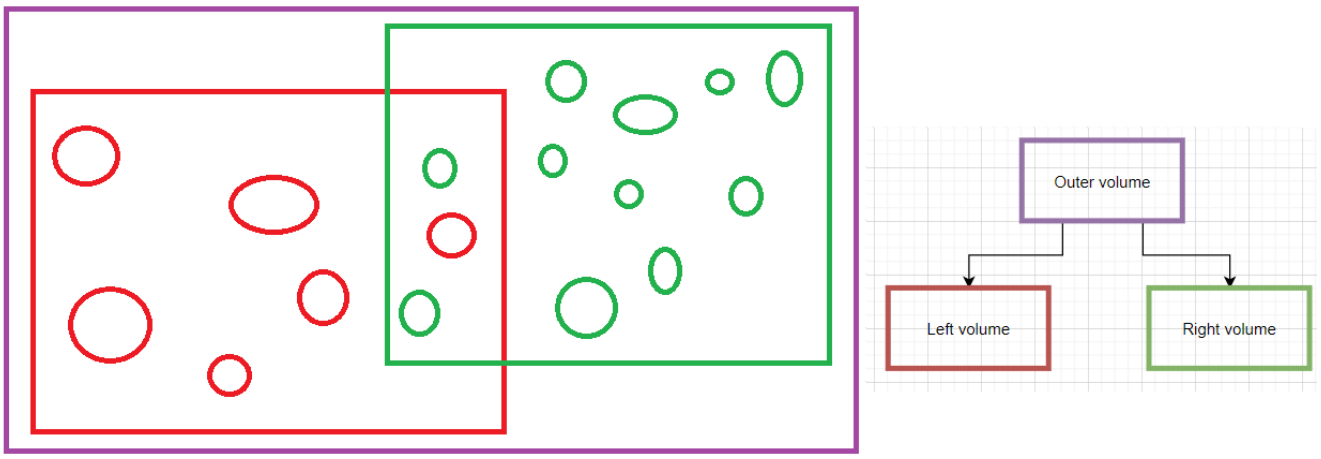

Рисунок 1.10 – Базова схема роботи BVH

Але об’єкти таки складаються з тисяч трикутників. Розвиваючи попередню ідею далі, то можна розділити сам об’єкт на декілька зон, як показано на рисунку 1.11.

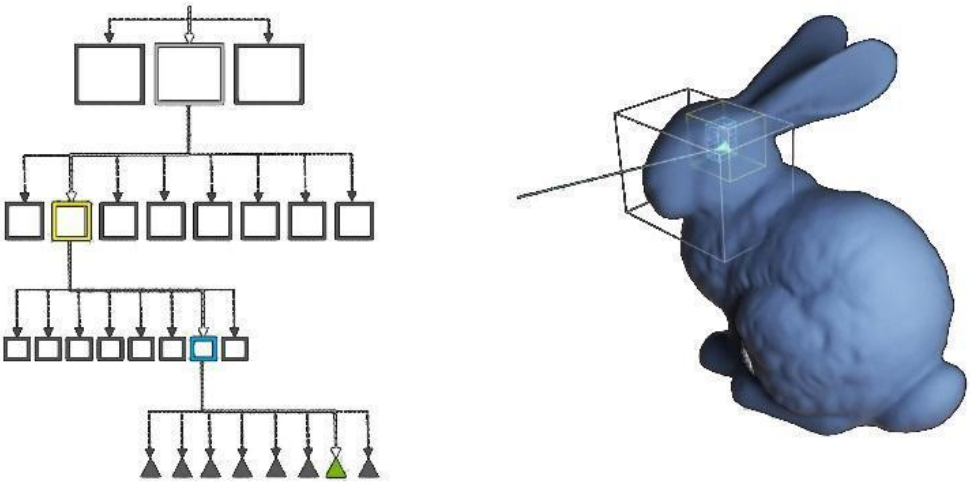

Рисунок 1.11 – BVH для геометричної моделі

Тобто перший обсяг буде включати всю модель, наступні певну її частину. Глибина таких обсягів залежить від складності певної моделі та має тестуватися у конкретній системі візуалізації.

1.2 Методи та проблеми генерації зображення у реальному часі

Для отримання зображення початкові дані (геометричні об’єкти, джерела освітлення та камера) мають бути оброблені так, щоб фінальне двомірне зображення мало глибину.

Самим простим методом отримання зображення у реальному часі [11] є алгоритм прямого освітлення (див. рис. 1.12). Його можна назвати алгоритмом грубої сили, тому як при обробці кожного об’єкту геометрії одразу розраховується його колір залежно від кожного джерела освітлення у сцені.

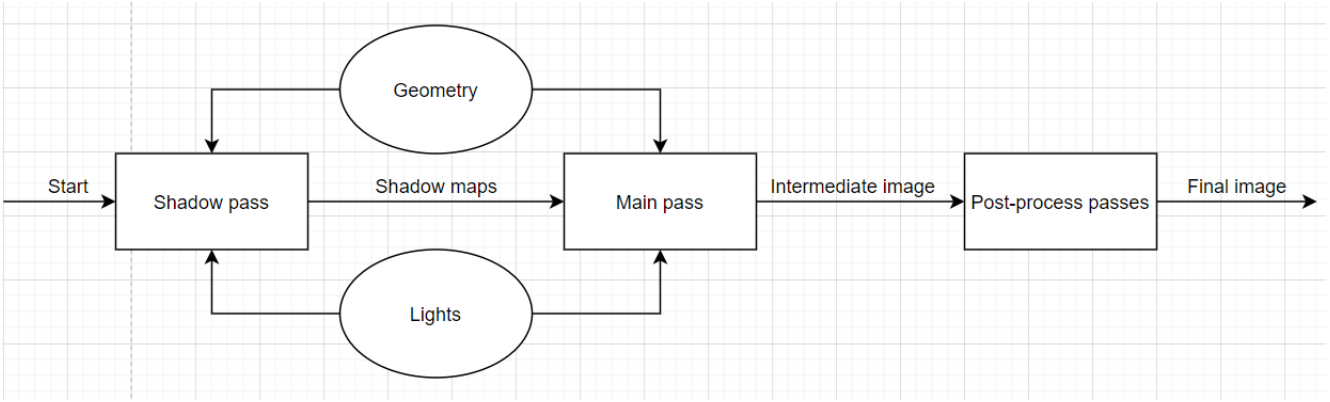

Рисунок 1.12 – Схема алгоритму прямого освітлення

Першим є генерація карт тіней. Тіни – це результат відсутності світла в зв'язку з тим, що промінь світла не достигає поверхні (див. рис. 1.13). Для розрахунку цього ефекту з положення джерела освітлення розраховується глибина кожного пікселя.

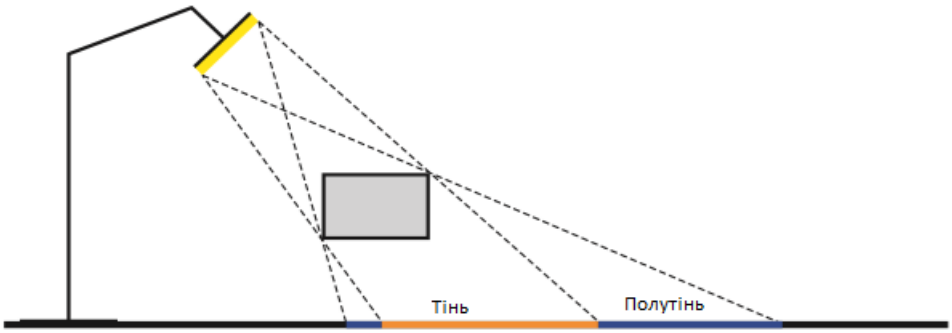

Рисунок 1.13 – Тіні

Це досягається шляхом постановки віртуальної камери у положення джерела освітлення [12] та рендерингу сцени з цього положення та запису у буфер інформації щодо глибини (див. рис. 1.14). Від розміру буферу залежить швидкодія цього етапу.

Цей етап в великій кількості систем виконується за допомоги методу растеризації, так як цей метод має найкращу швидкодію, що є ключовим параметром для систем реального часу. Але цим методом можна генерувати карти тіней лише для нескінченно малих джерел, або необхідно використовувати хитрощі для отримання, наприклад, полу тіней від прямокутного джерела. Для цього можна

отримати карти тіней з кожного кута, а потім дивитись якщо фрагмент у тіні для всіх кутів – він у тіні, якщо у декількох – він у полу тіні.

Рисунок 1.14 – Візуалізація буферу глибини для джерела освітлення

Карти тіней можна також генерувати за допомогою методу трасування променів. Одним з варіантів є також поставити віртуальну камеру у положення джерела освітлення та запису інформації, щодо глибини у буфер. Отримання карт тіней від джерел освітлення, які мають площину таким шляхом легше, адже можна трасувати промені з випадкової точки цієї площини.

Наступним етапом є головний прохід. У цьому етапі трапляється отримання інформації щодо кольору геометричних об’єктів з точки зору з точки зору віртуальної камери. На цьому етапі використовується інформація, щодо положення геометрії, положення джерел освітлення та їх карти тіней, тобто для кожного геометричного об’єкту, що потрапляє у набір фрагментів (пікселів) трапляється розрахунок з кожним джерелом освітлення. Складність такого алгоритму становить O(n + m), де n – кількість геометричних об’єктів сцени, m – кількість джерел освітлення.

Я не бачив жодної статті, де цей етап робився би не за допомоги растеризації у системах візуалізації реального часу, але виконати його за допомоги трасування реально, тому що обидва методи дозволяють отримати інформацію щодо відповідності фрагменту (пікселя) геометричному об’єкту. Так насправді в цьому

немає сенсу, тому що на даний чар апаратне прискорення растеризації набагато розвинуте (апаратне прискорення растризації з 1981 року, апаратне прискорення трасування променів з 2018 року [13]), а зайняти ці блоки під час виконання даного етапу немає чим.

На етапі пост обробки виконується різні ефекти кінцевого зображення та приведення зображення до формату, який підходить до виводу, наприклад, приведення зображення до LDR з HDR [14].

Розглянемо обернене освітлення.

Головною відмінністю оберненого освітлення від прямого є розділ обробки геометрії від обробки освітлення. Схема методу оберненого освітлення представлена на рисунку 1.15.

На геометричному етапі відбувається генерація геометричного буферу, який є сукупністю буферів параметрів сцени. Наприклад для PBR рендеру цей буфер буде складатися з п’яти буферів (див. рис. 1.16):

– альбедо буфер;

– буфер нормалей у мировому просторі;

– буфер шорсткості;

– буфер металевості;

– буфер мирової позиції (для оптимізації пам’яті можна використовувати буфер глибини, та перетворювати його до буферу позиції за потреби, але більш наглядно представляти його як буфер позицій).

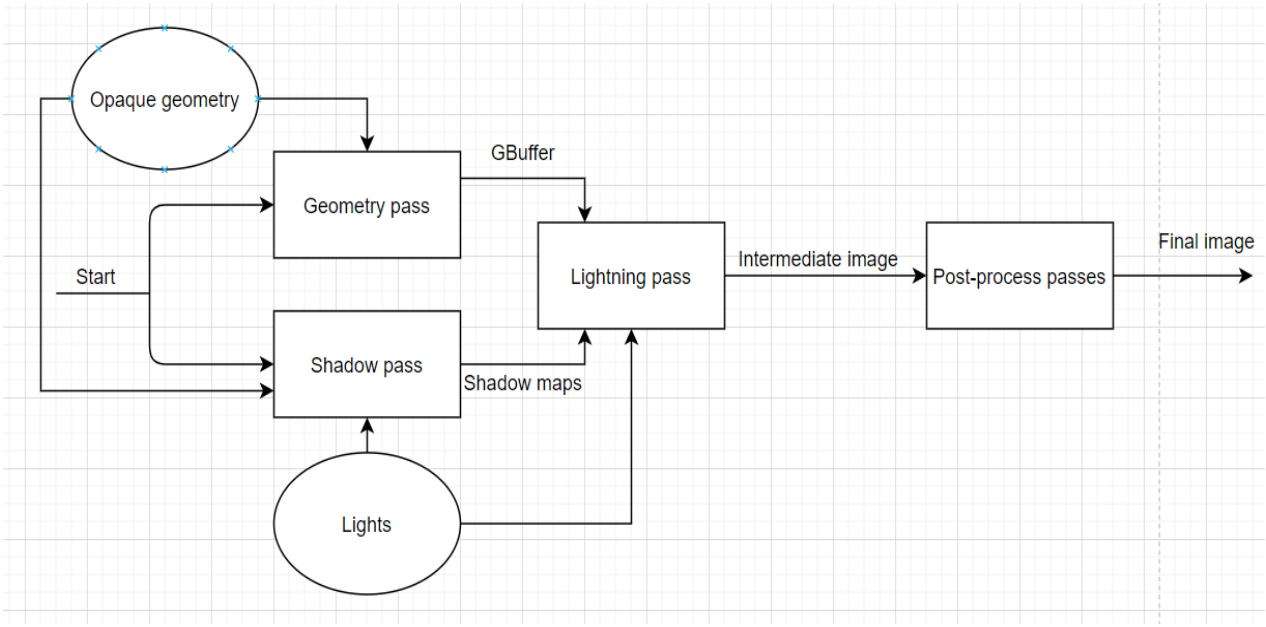

Рисунок 1.15 – Схема алгоритму відкладеного освітлення

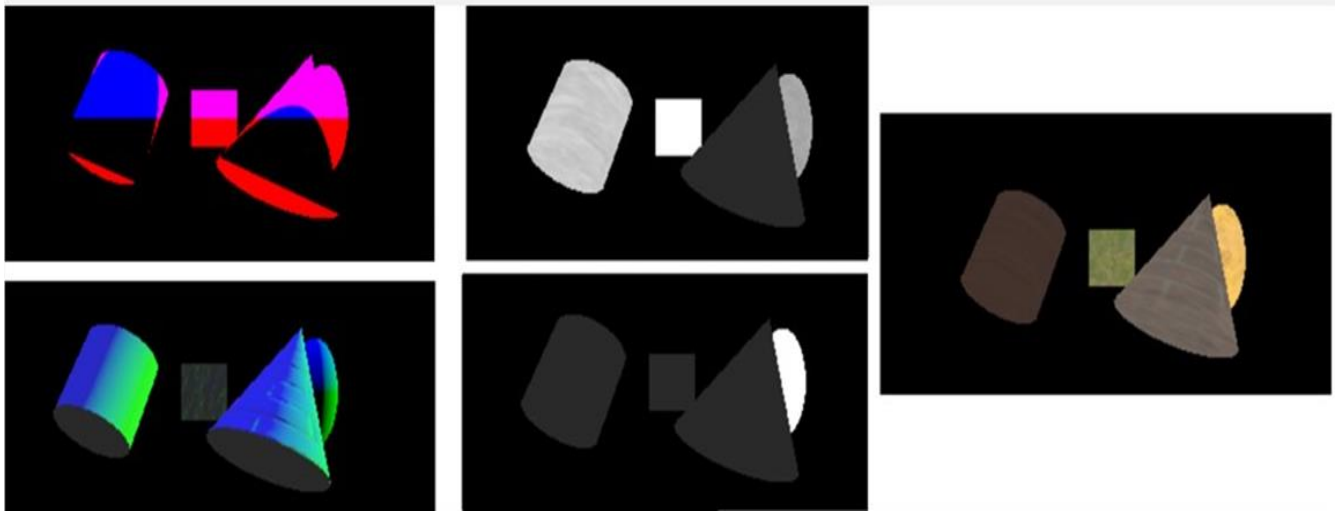

Рисунок 1.16 – Візуалізація геометричного буферу

Геометричний буфер є результатом обробки всієї геометрі, тобто на вхід наступного етапу подається лише цей буфер та жодного окремого геометричного об’єкту. Як і з головним етапом прямого освітлення немає сенсу роботи його методом трасування променів.

Наступним етапом є освітлення. Для розрахунку освітлення кожного фрагменту для кожного джерела освітлення знаходиться місце на геометричному буфері, де джерело взаємодіє з геометрією, та маючи усі параметри геометрії у

буфері, карти нормалей та параметри джерела легко розрахувати колір фрагменту. Наприклад, точкове джерело освітлення може бути представлене (див. рис. 1.17).

Можна сказати, що такий підхід – це пост обробка світлом зображення. Такий підхід дозволяє мати складність як O(n + m). Тобто, за наявністю великої кількості геометричних об’єктів та джерел освітлення швидкодія оберненого освітлення такого алгоритму буде вища ніж у прямого. Але при такому алгоритмі не має можливості візуалізувати прозору геометрію, що є дуже великим недоліком.

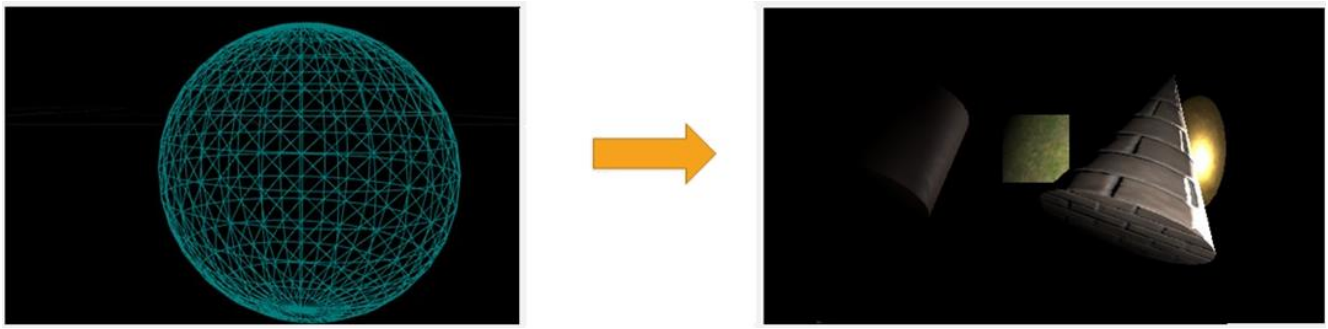

Рисунок 1.17 – Точкове джерело освітлення

Одна з можливостей візуалізувати прозору геометрію – це додати ще один етап прямої відрисовки прозорої геометрії (див. рис. 1.18).

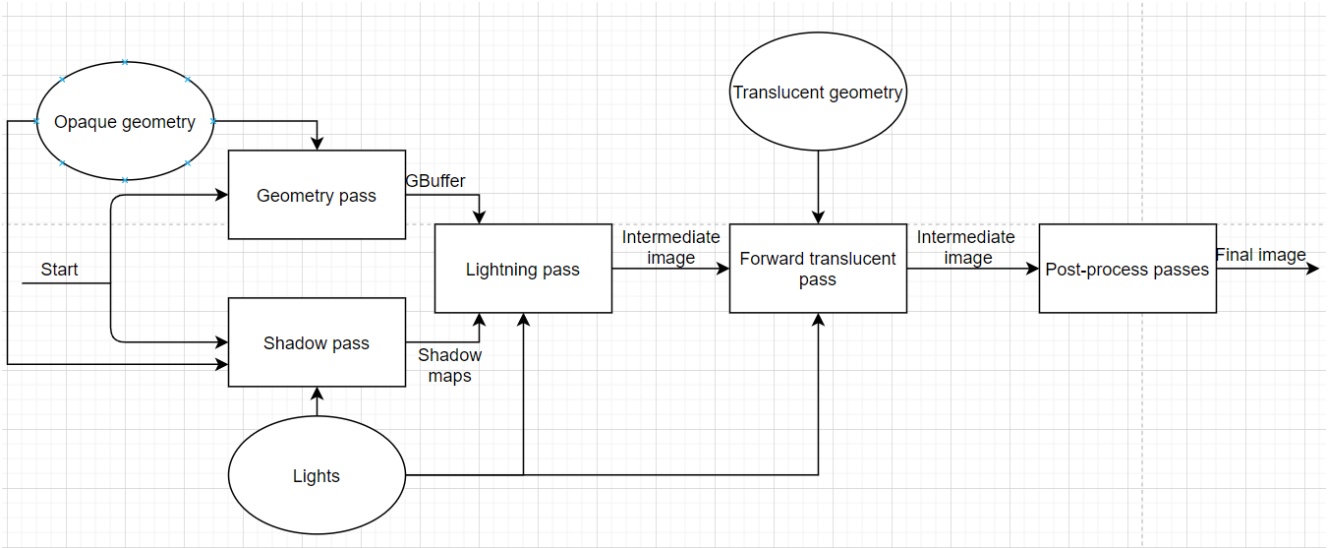

Рисунок 1.18 – Схема алгоритму оберненого освітлення з додаванням алгоритму прямого для відрисовки прозорої геометрії

Так складність всього алгоритму складатиме O(n+m) + O(nk), де k – кількість прозорої геометрії.

Прозора геометрія накладається зверху непрозорої, тобто інформація щодо непрозорої має бути отримана насамперед [15]. Прозоро геометрія має переломляти світло відповідно закону переломлення світла на границі розподілу середовищ [16]. Інформація, щодо геометрії куди переломиться промінь не завжди присутня у геометричному буфері, тому що промінь може влучити у геометрію поза зображенням.

Ще одна проблема виникає тоді, коли необхідно растеризувати, наприклад, прозору сферу, або прозору порожню сферу. Яку сторону сфери необхідно растеризувати у першу чергу та можемо ми взагалі на це вплинути? Як буде показано далі – ми не маємо доступу до етапу растеризації на графічному прискорювачі. Цю проблему можна вирішити маючи дві півсфери для передньої частини та задньою, але сфера – це ідеальний приклад. На рисунку 1.19 видно артефакти, що виникають при змозі візуалізувати лампу.

Рисунок 1.19– Приклад візуалізації прозорої геометрії методом растеризації

Метод трасування променів не має цих проблем, адже ми можемо переломляти промінь при контакті з поверхнею (див. рис. 1.8). Результат візуалізації прозорої геометрії трасуванням променів видно рисунку 1.20.

Рисунок 1.20 – Результат візуалізації прозорої геометрії трасуванням променів

З рисунку видно, що прозора геометрія немає тих артефактів, що були отримані методом растеризації.

Великою проблемою є відображення. В класичному оберненому методі це робиться одним з етапів пост процесу. Методом растеризації немає способу отримати коректні відображення. У великій кількості систем використовується SSLR [18]. У геометричному буфері є інформація, щодо нормалей та глибини сцени, тому можна апроксимувати на поверхні з малою шорсткістю те, що відображається. З назви алгоритму ясно, що цей ефект працює лише на геометрію, яку видно в поточному полі зору, тому за допомогою цього алгоритму нереально зробити щось схоже на дзеркало [19]. Він підходить, коли на сцені багато геометричних об’єктів неправильної форми та відображення розмиті, коли таке відображення попадає на рівний об’єкт біля до камери, то артефакт важко не помітити (рисунок 1.21).

З рисунку видно, що на метолевому стакані відображається лише та частина книг, яка входить в поле зору, а більша частина стакана взагалі немає відображень, тому що те що могло би там відобразитись не входить у поле зору.

Відображення методом трасування променів випливає з роботи самого цього методу та в ніяких допущеннях, таких як SSLR не має потреби.

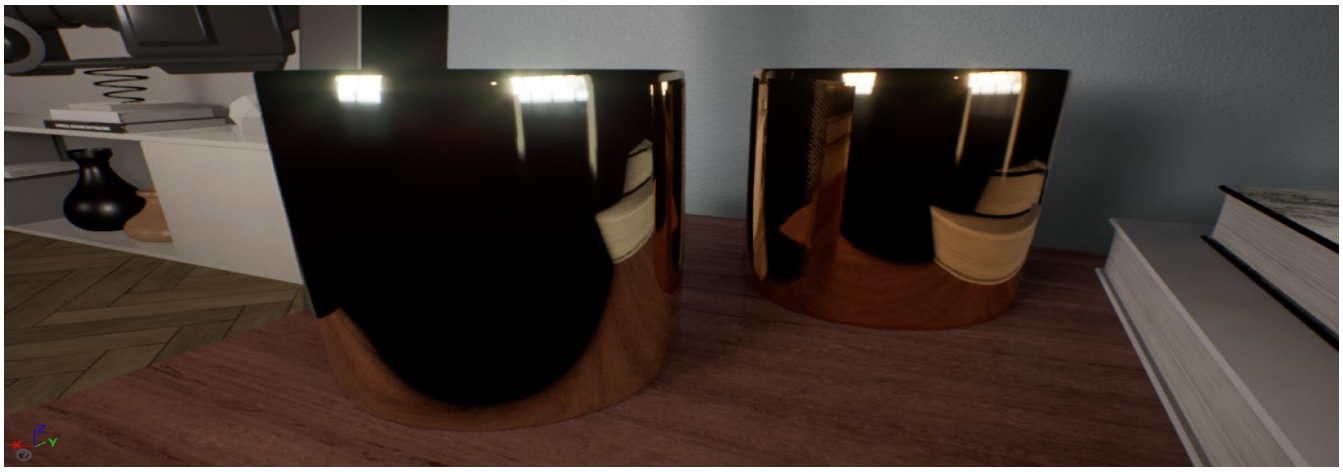

Рисунок 1.21 – Результат роботи SSLR

З рисунку 1.2 видно, як легко можна отримати відображення трасуванням. Порівняти результат можна на рисунку 1.22.

Рисунок 1.22 – Відображення трасуванням променів

Чим більше відскоків кожного променю – тим більше деталей буде у відображені, але швидкодія буде нижчою.

1.3 Принципи роботи апаратного прискорення

1.3.1 Растеризація

У сучасних графічних API, як DirectX12 та Vulkan генерація зображення відбувається за допомогою конвеєру растеризації (див. рис. 1.23). Ці API підтримуються усіма сучасними відеокартами. DirectX12 є лише на платформі Windows коли як Vulkan має підтримку багатьох платформ таких як Windows та Android[19].

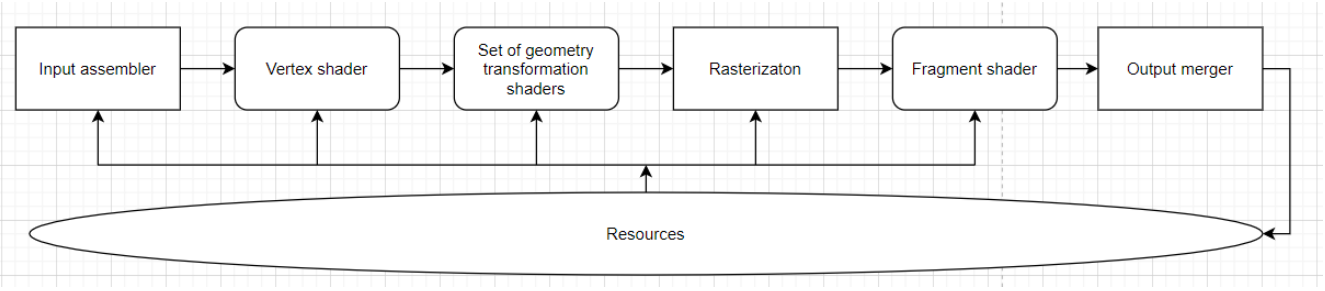

Рисунок 1.23 – Конвеєр візуалізації сучасних графічних API

Хоча він називається конвеєром растеризації, цей метод є лише одним з етапів візуалізації.

Першим етапом є вибір примітивів. На цьому етапі інформація щодо вершини (положення, нормаль, колір тощо) групується та передається далі у вершинний шейдер. Цей етап є непрограмованим, його робота відбувається за вхідних параметрів таких як зміщення певних атрибутів геометричної моделі в пам’яті, як це трапляється – відповідність драйверу ГПУ.

Інформація щодо вершини потрапляє до вершинного шейдеру. Тобто кожна окрема вершина геометричної моделі обробляється вершинним шейдером. Цей етап є доступним до програмування. У першу чергу тут робиться перетворення координат з локальних координат об’єкту до мирових координат, координат камери та координат зображення. Також тут може бути розрахована додаткова інформація, щодо різноманітних ефектів трансформації геометрії.

Набір шейдерів трансформації геометрії трохи різний для кожного з API, але всі вони направлені на генерації та зміщення геометрії. Вони є програмованими.

Перед растеризацією відбувається збірка примітивів у трикутники та на вхід етапу растеризації йдуть трикутники (можуть лінії, але вони найчастіше використовується для відладки). На виході маємо фрагменти(пікселі) зображення, куда був растреізован трикутник. Цей етап є непрограмованим.

Для кожного фрагменту трикутника з етапу растеризації виконується програмований фрагментарний (піксельний) шейдер. Саме тут розраховується фінальний колір, або як з геометричним буфером різні параметри такі як глибина, позиція, нормалі, шорсткість та металевість кожного пікселю[21].

На етапі збірки результату відбувається тестування глибини та запис в буфер відповідного фрагменту.

Тобто для кожного геометричного об’єкту виконується цей конвеєр та результат запустується до асоційованого до етапу буферу, наприклад, геометричного буферу оберненого освітлення. Кожен наступний етап методу генерації зображення у реальному часі може використовувати буфер з попереднього етапу та писати в той самий чи інший буфер. Для кожного об’єкта можна задати різний набір шейдерів, так як вони можуть мати різні властивості поверхонь, наприклад, метал або скло.

1.3.2 Трасування променів

Апаратну підтримку трасування променів може забезпечити лише API DirectX 12 з його розширенням DXR. Це розширення підтримують лише відеокарти NVIDIA покоління Pascal та Turing [20]. Але тільки покоління Turing має виділену апаратну частину для обробки трасування променів.

На час написання цієї роботи API Vulkan має тестову підтримку трасування променів. Також були представлені ігрові консолі Sony PS5 та Microsoft Xbox X,

для яких підтверджено апаратну підтримку трасування променів, тож в недалекому майбутньому використання трасування променів у системах візуалізації реального часу не визиває питань.

Конвеєр трасування променів для DXR представлено на рисунку 1.24.

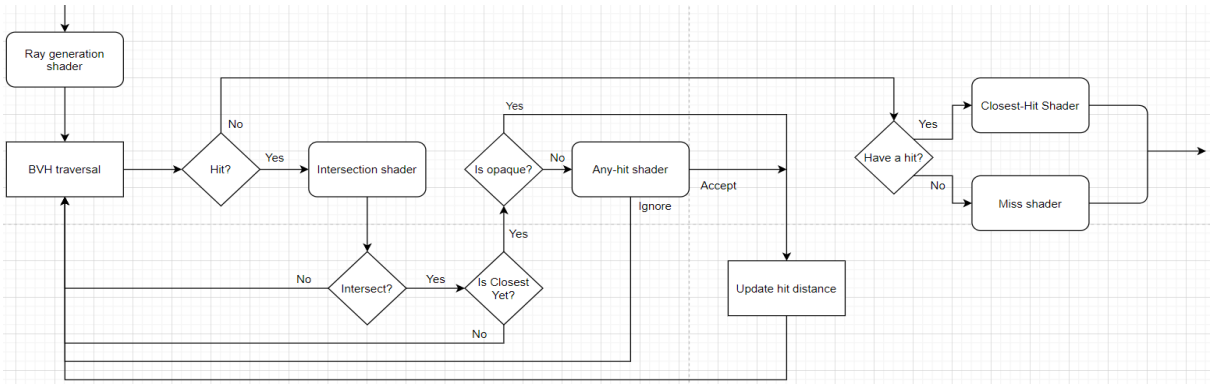

Рисунок 1.24 – Конвеєр трасування променів DXR

Шейдер генерації променів стартує конвеєр. Він є програмованим та дозволяє обумовлювати, які промені необхідно трасувати.

Далі відбувається обхід BVH, де визначається відповідний примітив.

Якщо є перетин с BVH виконується програмований шейдер перетину, в якому можна реалізувати алгоритм перетину з певною геометрією. Частіше за все з трикутником.

Шейдер влучення з прозорою геометрією є програмованим та дозволяє не враховувати, або враховувати прозору геометрію, як найближчу.

Шейдер найближчого влучення є програмованим та виконується один раз для кожного променю. Зазвичай визначає колір у точці перетину подібно фрагментному(піксельному) шейдеру у конвеєрі растеризації.

Шейдер промаху виконується кожен раз, коли промінь не влучив у жодну геометрії у сцені. Це дозволяє розрахувати колір, наприклад, неба. У конвеєрі растризації для цього необхідно мати нескінченно віддалену сферу або куб, щоб він растеризувався у фрагменти, куди не растеризувалась інша геометрія.

Кількість запущених променів шейдером генерації залежить від розміру буферу. Розрахунок камери та напряму променів для кожного пікселя відбувається у шейдері генерації променів.

1.4 Науково-технічна задача

У недалекому майбутньому усі системи візуалізації у реальному часі будуть використовувати апаратно прискорене трасування променів. На даному етапі метод трасування променів не може повністю замінити метод растеризації, тому що має дуже великі вимоги до обчислювальних здібностей, яке сучасне “залізо” не може задовольнити, але почалися перші кроки у цьому напрямку та деякі етапи генерації зображення можна замінити трасуванням променів. Такий комбінований підхід дозволяє наблизити зображення у сторону фото реалістичності та що найважливіше для систем візуалізації у реальному часі не втратити швидкодію та на деяких етапах її збільшити завдяки тому, що апаратне прискорення трасування променів відбувається на виділеній лише під це частині відеокарти. З цієї причини було прийнято рішення дослідити можливість використання методу трасування променів у методах візуалізації у реальному часі.

Дана дослідницька робота повинна демонструвати наступні використання методу трасування променів:

– генерації карт тіней;

– візуалізація прозорої геометрії;

– візуалізація відображень;

– глобальне освітлення.

Таким чином вихідна програма повинна демонструвати застосування у системі візуалізації методу трасування променів для різних етапів генерації зображення та показувати на якому етапі, або їх комбінації доцільніше його використовувати чи доцільно взагалі.

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ТА ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

2.1 Принципи роботи модифікованого алгоритму генерації зображення

Як було показано у розділі 1.2 генерація зображення складається з кількох етапів (див. рис. 2.1). Їх можна представити, як послідовність модифікаторів, яка використовує дані з попереднього етапу починаючи з інформації, щодо положення об’єктів закінчуючи фінальним зображенням.

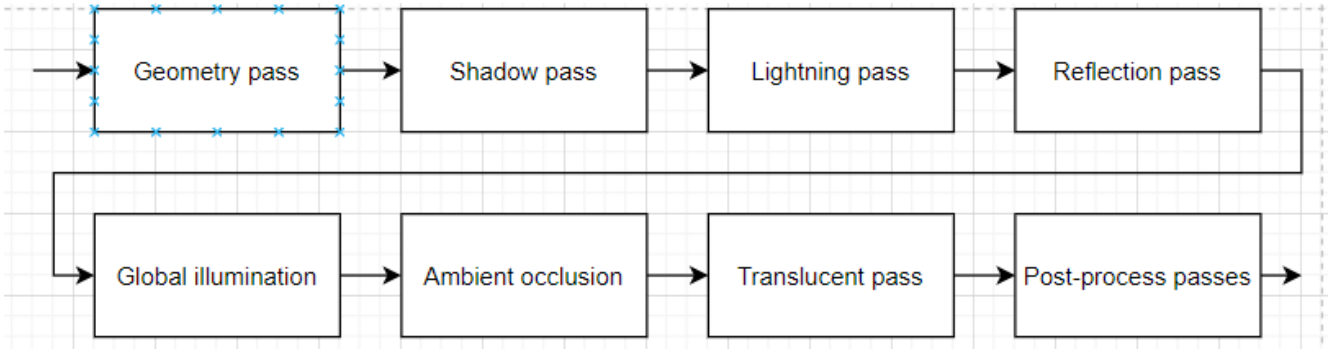

Рисунок 2.1 – Алгоритм генерації зображення

Довгий час кожен з цих етапів був зроблений методом растеризації. Для деяких етапів цього методу було достатньо для отримання фізично коректної інформації. Наприклад для генерації геометричного буферу растеризація підходить найбільш. Маючи геометричні об’єкти сцен у своєму локальному просторі дуже легко отримати їх властивості як один буфер, зробивши перетворення координат за допомоги набору матриць [18]. Множення матриць – це основна задача графічного прискорювача, яку він може робити паралельно для кожного пікселя.

Розрахунок прямого світла також добре піддається растеризації, тому як було показано у розділі 1.2 – це практично теж саме, що і обробка геометрії.

Але обробка динамічного глобального освітлення не можливо зробити методом растеризації не вдаючись до припущень. Точна динамічна обробка

відображень також неможлива, тому як необхідно б було візуалізувати сцену з кожної точки простору та у кожному направленні, важко уявити це для кількох, не говорячи о десятках та більше. Найближче, що схоже на динамічні відображення – це SSLR, які виглядають схожими на відображення на віддалених нерівних об’єктах, але не можуть бути використані, як метод для візуалізації, наприклад, дзеркал. Схожа проблема з прозорою геометрією. Часто прозорість в таких системах використовується для локальних ефектів пост процесу, як, наприклад, ефект зміни повітря від вогню, та ріже для прозорої геометрії.

З іншого боку є метод трасування променів, який використовується у системах візуалізації, де час генерації не є основним параметром. Навидь з появою апаратного прискорення трасування променів час генерації повного зображення з його допомогою дуже довге.

Але нічого не заважає використати комбінований підхід до генерації, де деякі етапи можна отримати методом растеризації, а деякі трасуванням променів. Такий алгоритм має бути не набагато повільнішим, а може й швидшим за стандартний, так як єдина архітектура відеокарт з апаратною підтримкою трасування має окремі блоки для цього [19].

В цій роботі досліджено наступні методи трасування променів у алгоритмі генерації зображення:

– тіні (shadow pass);

– відображення (reflections);

– глобальне освітлення (global illumination);

– затінення навколишнього світла (ambient occlusion);

– прозора геометрія (translucent pass).

Для відображень існує два параметри – максимальна кількість відскоків одного променю та максимальна шорсткість відображених об’єктів. Максимальна кількість відскоків – це потенційна кількість інформації, яку отримає промінь, тобто для підвищення якості треба підвищувати максимальну кількість. Другим параметром є максимальна шорсткість поверхні. Тобто шорсткість до якої промінь буде відображатись. Якщо промінь зустріне поверхню, шорсткість якої перевищує

це значення – збір інформації припиниться. Також шорсткість впливає на відхилення променю, як буде показано у розділі 2.2. Тобто обидві ці параметри впливають на кількість кінцевої інформації. Найкращий результат буде отримано за максимізації цих параметрів, але навіть за мінімальних результатів результат набагато кращий ніж методом SSLR.

Як і для відображень для прозорості таки існує два параметри. Один також відповідає за максимальну шорсткість, а другий за кількість променів, але на відміну від відскоків навпаки відповідає за кількість можливих заломлювальних променів. Якщо встановити цей параметр низьким, то промінь не зможе вирватися з складного об’єкту. Наприклад, одного променю недостатньо для того, що вибратись зі сфери (див. рис. 2.2).

Рисунок 2.2 – Промінь не може вибратись зі сфери

Параметром глобального освітлення є кількість ітерацій на піксель. Тобто скільки раз буде зібрана інформація для пікселя з деякими випадковими зміщеннями [21]. Підвищення цього параметру дозволяє зменшувати шум. Тобто якщо промінь сусіднього пікселя відскочив до світлої геометрії, а поточний до темної. Підвищенням кількості променів на піксель можна отримати середнє значення. Затінення навколишнього світла працює так само, але трасується лише

дальність до ближчого сусіднього об’єкту, коли як в глобальному освітленні ще трасується колір.

Параметри тіні – це кількість джерел освітлення на сцені. Параметри сцен вказані в таблиці 2.1.

Таблиця 2.1 – Параметри досліджуваних сцен

	Сцена 1	Сцена 2	Сцена 3
Кількість трикутників	547800	101000	268200
Кількість джерел освітлення	15	5	10

Ці параметри були зібрані з учбового проекту UE4, тому в цілому вони відображають середню складність сцен у сучасних системах.

2.2 Методи освітлення

Для цілей затінення прийнято розглядати промені світла як відбитий і заломлений в різних напрямках в кожній точці поверхні (див. рис. 2.3).

У разі металів енергія переломлення світла миттєво поглинається вільними електронами, в разі діелектриків промені поводяться як на границі розподілу середовищ. Заломлене світло може бути випромінене тієї ж самою поверхнею.

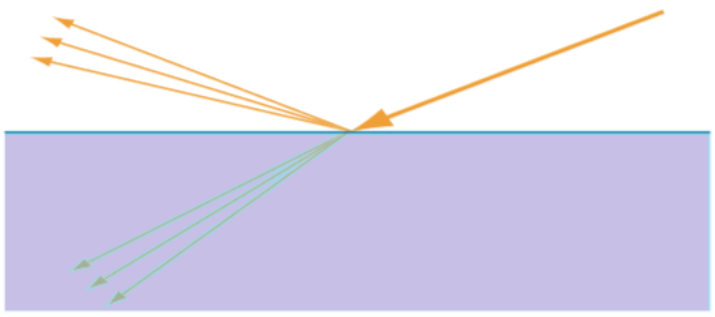

Рисунок 2.3 – Розбивка променю світла на кордоні середовища

Прийнято, що поверхня, яка буде перевипромінювати світло менше одного пікселя, це дозволить розраховувати затінення як локальний процес. В цьому випадку реакція точки поверхні на світло повністю залежить тільки від вхідного (світло) і вихідного (огляд) напрямків [16].

Для затінення для кожної точки поверхні (пікселя) необхідно вирішити рівняння візуалізації (2.1). Це означає, що необхідно знайти всю енергетичну яскравість в точці, яка прийшла по півсфері і знайти енергетичну яскравість, відображену в сторону спостерігача [17]:

$$L_0(p, \omega_0) = \int_{\Omega} \quad f_r(p, \omega_i, \omega_0) L_i(p, \omega_i) n \cdot \omega_i d\omega_i, \tag{2.1}$$

де $L_0(p, \omega_0)$ – енергетична яскравість в точці p в напрямку ω_0, $L_i(p, \omega_i)$ – енергетична яскравість в точці p в напрямку ω_i, $f_r(p, \omega_i, \omega_0)$ – BDRF.

BDRF – функція, яка приймає в якості аргументу напрямок світла, напрямок огляду, нормаль поверхні і параметр, що відповідає за шорсткість поверхні. BDRF апроксимує наскільки кожен вхідний промінь у напрямку ω_i впливає на результат. Наприклад, для повністю гладкої поверхні функція поверне нуль для всіх вхідних променів у напрямку ω_i, крім одного променю, який має такий самий напрямок відображення, що і напрямок огляду ω_0.

Для задоволення вимог в збереженні енергії підходить BDRF Кука-Торренса [20].

BDRF Кука-Торренса містить розсіяну і відображену частини:

$$f_r = k_d f_{lambertian} + k_s f_{cook-torrance}, \tag{2.2}$$

де k_d – коефіцієнт світла, який був розсіяний, $f_{lambertian}$ – розсіяна складова Ламберта, k_s – коефіцієнт відбитого світла, $f_{cook\text{-}torrance}$ – відбиваюча складова Кука-Торренса.

Розсіяна складова Ламберта апроксимує розсіяне світло і є константою, залежить від альбедо поверхні C:

$$f_{lambertian} = \frac{C}{\pi} \tag{2.3}$$

Відображена складова базується на теорії мікро поверхонь, яка описує відображення від не оптично-рівних поверхонь[6]:

$$f_{cook-torrance} = \frac{DFG}{4(\omega_0 * n)(\omega_i * n)} \tag{2.4}$$

де D – це функція нормального розподілу (NDF), апроксимуюча кількість мікроповерхонь поверхні, чиї нормалі вирівняні до медіанного вектора, що залежить від параметра шорсткості поверхні, G – це геометрична функція, яка описує властивість самозатінення мікроповерхнонь, F – рівняння Френеля. Визначає ставлення відображення під різними кутами огляду/падіння.

Функція нормального розподілу Тровбрідж-Рейтца [17]:

$$D_{GGXTR}(n, h, \alpha) = \frac{\alpha^2}{\pi((n * h)^2(\alpha^2 - 1) + 1)^2} \tag{2.5}$$

де α – коефіцієнт шорсткості поверхні від 0 до 1. Чим менше шорсткість, тим статистично більше мікро поверхонь вирівняно до медіанного вектора (див. рис. 3.11), n – нормаль поверхні, h – медіанний вектор.

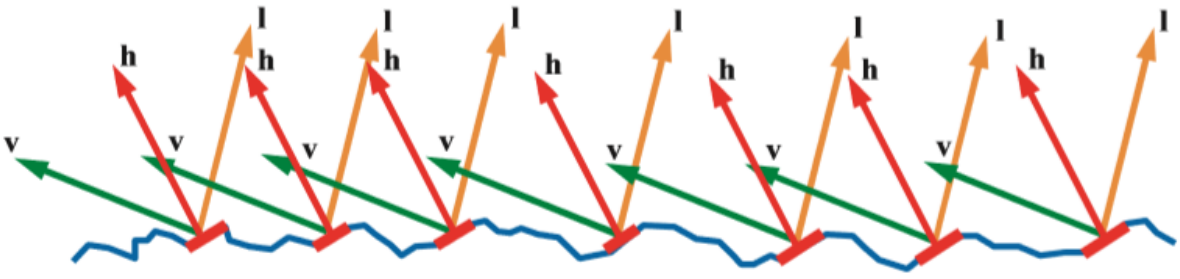

Рисунок 2.4 – Мікро поверхні, нормалі яких вирівняні с медіанним вектором

Геометрична функція G так само, як і D базуючись на шорсткості апроксимує кількість світла, який перекривається власної геометрією поверхні (див. рис. 2.5).

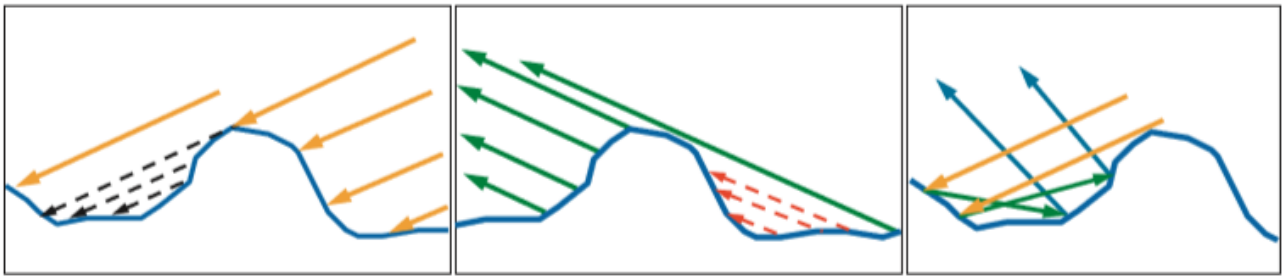

Рисунок 2.5 – Променів світла, що перекриваються геометрією поверхні

Використовувана геометрична функція є комбінацією GGX і апроксимація Шліка-Бекмана – Schlick-GGX [15]:

$$G_{schlickGGX}(n, v, k) = \frac{n * v}{(n * v)(1 - k) + k} \quad (2.6)$$

де k – масштабування α для різних моделей джерел освітлення, v – нормалізований вектор від поверхні.

Функція Френеля описує частку світла, яке було відбите від поверхні. Залежить від кута падіння променя світла і від індексу рефракції матеріалу. Так як індекс рефракції може бути різним по всьому видимому спектру, відображення Френеля – спектральна величина, закодована в RGB триплет. Рівняння Френеля досить складні, щоб обчислювати їх для кожного пікселя з допустимим часом, можна використовувати апроксимацію Френеля-Шліка [19]:

$$F_{Schlick}(h, v, F_0) = F_0 + (1 - F_0)(1 - (h * v)^5 \quad (2.7)$$

де F_0 – заздалегідь розраховане значення базової відбивної здатності для матеріалу поверхні [18].

Рівняння 2.1 при підстановці набирає вигляду:

$$L_0(p,\omega_0) = \int_{\Omega} \left(k_d \frac{c}{\pi} + \frac{DFG}{4(\omega_0 \cdot n)(\omega_i \cdot n)}\right) L_i(p,\omega_i) n \cdot \omega_i d\omega_i \quad (2.8)$$

Функція Френеля F в даному випадку є коефіцієнтом відбитого світла ks. Так як кількість розсіяного і відбитого світла не може бути більшою за кількість світла, отриманого від джерела освітлення, то коефіцієнт розсіяного світла описується формулою:

$$k_d = (1 - F_0)(1 - metal) \quad (2.9)$$

де metal – бінарне значення, що характеризує матеріал як метал (1) або діелектрик (0).

Тобто метод трасування променів для отримання відображень та прозорої геометрії враховує α поверхні матеріалу та відсікає усі значення, які перевищують заданий параметр. Насправді, як було показано вище поверхня з великою шорсткістю буде частіше поглинати світло.

3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

У ході проведення досліджень було порівняно швидкодію методів трасування променів на різних етапах генерації зображення з базовою швидкодією, тобто без використання трасування променів. Базова швидкодія для першої сцени – 10.9 мс, другої сцени – 6.5 мс, третьої – 9.1 мс.

Результати замірів для генерації тіней наведено у таблиці 3.1, а графік різниці часу кадру з базовою швидкодією наведено на рисунку 3.1

Таблиця 3.1 – Час генерації кадру для тіней

	Сцена 1	Сцена 2	Сцена 3
Час кадру, мс	34.5	18.1	31.6

З графіку видно, що використання таких тіней коштує дуже багато, навіть для простої сцени. Найгірший час кадру, який є допустимим для систем візуалізації реального часу – 33.3 мс, а бажане – 16.6 мс.

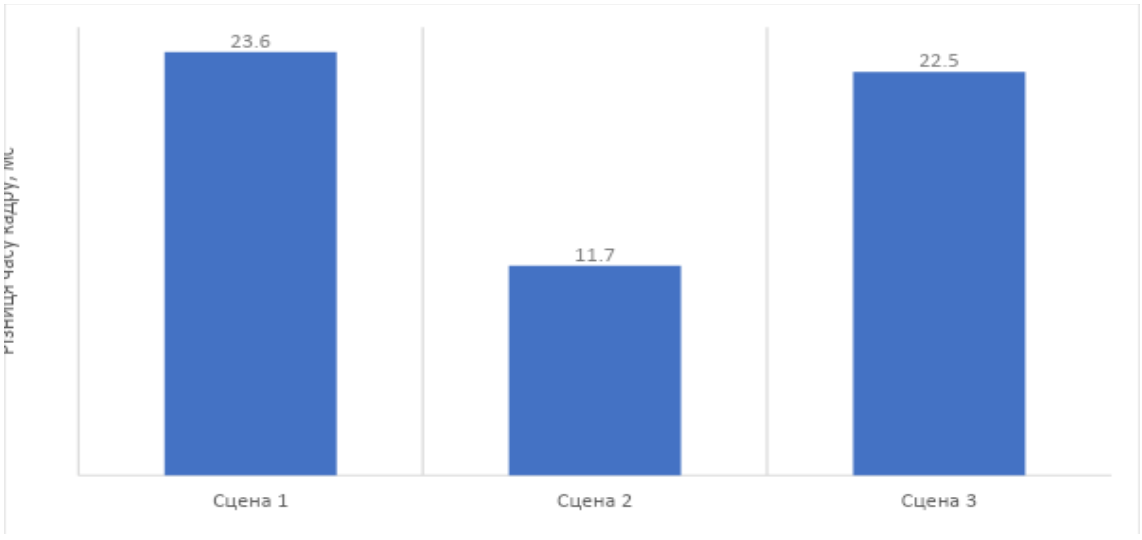

Рисунок 3.1 – Різниця часу кадру від базової для тіней

Час кадру для двох з трьох сцен є допустимим, тобто за необхідністю в, наприклад, малих іграх можна використовувати цей метод.

Для прозорої геометрії були виміряні комбінації двох параметрів. Результати замірів для першої сцени наведено в таблиці 3.2 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.2.

Таблиця 3.2 – Час генерації кадру для прозорої геометрії для першої сцени

Кількість променів	Максимальна шорсткість			
	0.01	0.1	0.5	1
3	14.9	15.3	25.8	26.2
10	15.2	15.4	26.8	27.2
20	15.5	15.8	28	28.2
30	15.8	16.1	29	29.2
40	16.1	16.3	30	30.2
50	16.4	16.7	31	31.2

Результат для параметрів, які забезпечують максимальну якість не виходить за рамки допустимого часу.

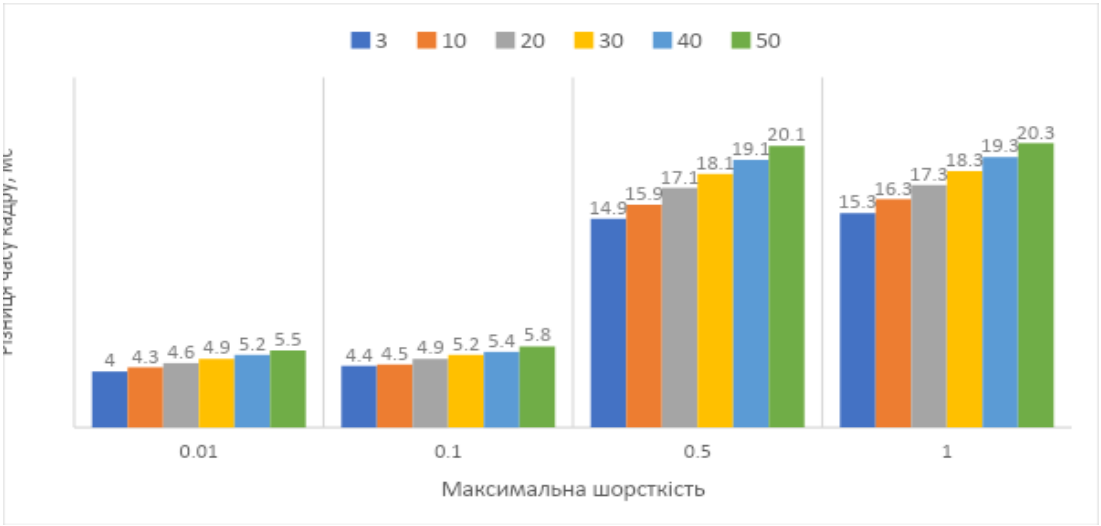

Рисунок 3.2 – Різниця часу кадру від базової для прозорості для першої сцени

Результати замірів для другої сцени наведено в таблиці 3.3 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.3.

Таблиця 3.3 – Час генерації кадру для прозорої геометрії для другої сцени

Кількість променів	Максимальна шорсткість			
	0.01	0.1	0.5	1
3	9.9	10	20.3	20.3
10	9.9	10.4	21.5	22.3
20	10.6	10.7	22.8	23
30	10.7	10.8	22.9	23
40	10.7	10.8	22.9	23
50	10.7	10.8	22.9	23

Для другої сцени результати також знаходяться у рамках допустимого.

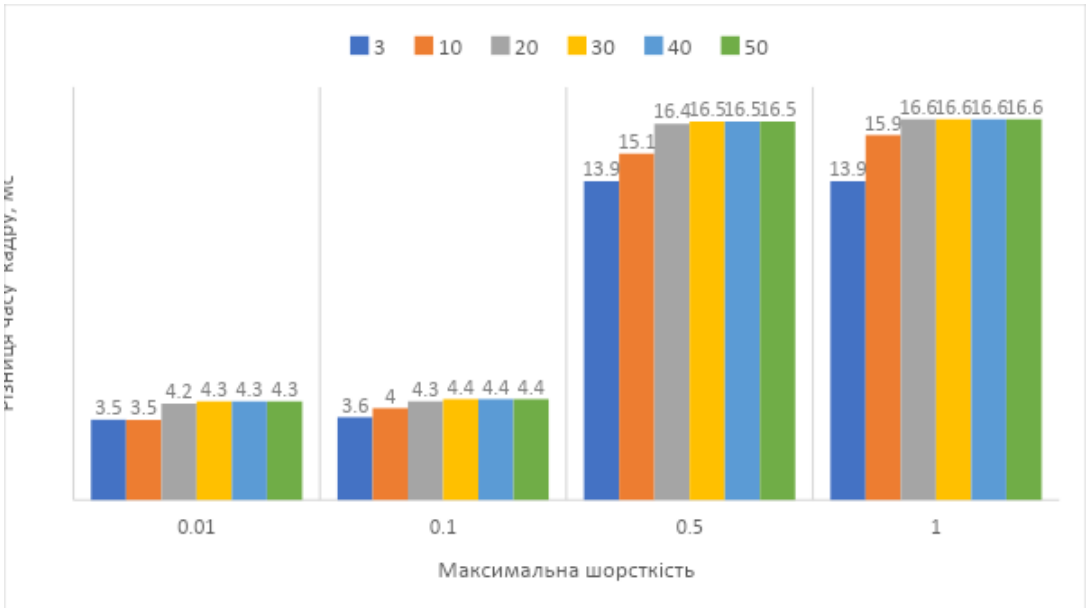

Рисунок 3.3 – Різниця часу кадру від базової для прозорості для першої сцени

Також видно, що підвищення кількості променів впливає на швидкодію не так сильно, як максимальна шорсткість.

Результати замірів для другої сцени наведено в таблиці 3.4 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.4.

Таблиця 3.4 – Час генерації кадру для прозорої геометрії для третьої сцени

Кількість променів	Максимальна шорсткість			
	0.01	0.1	0.5	1
3	14	14.2	30.5	31.6
10	14	14.2	31.6	32.6
20	14	14.4	32.1	33.4
30	14.1	14.7	32.7	33.5
40	14.4	14.9	33.1	33.7
50	14.7	15	33.4	34.5

Для третьої сцени швидкодія від підвищення шорсткості погіршується швидше незалежно від того, що сцена має менше трикутників у цілому. На відміну від першої сцени третя має вид на океан, який являється прозорої геометрію, тобто прозора геометрія займає більшу частину зображення.

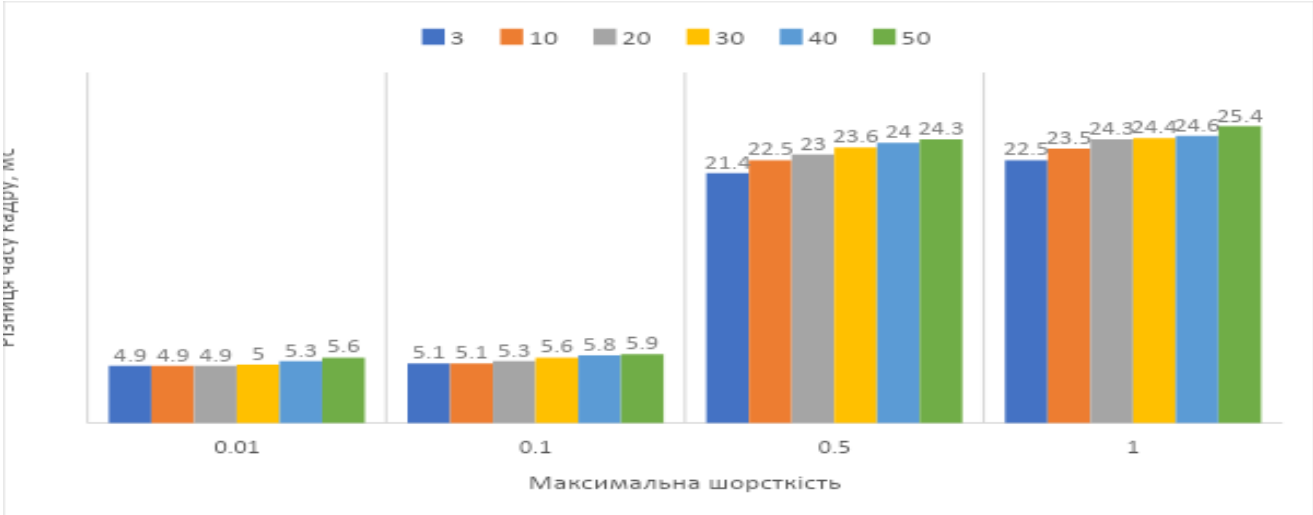

Рисунок 3.4 – Різниця часу кадру від базової для прозорості для першої сцени

Результати замірів для відображень для першої сцени наведено в таблиці 3.5 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.5.

Таблиця 3.5 – Час генерації кадру для відображень для першої сцени

Кількість променів	Максимальна шорсткість			
	0.01	0.1	0.5	1
1	13.7	14.2	30.7	34.1
2	13.8	14.9	64.3	84.7
4	13.9	15.4	105.3	232
8	13.9	15.6	112	285
16	13.9	15.6	130	295

Результати досліджень для відображень сходні з результатами для прозорої геометрії. Підвищення кількості променів для маленької шорсткості практично не впливає на швидкодію, але якщо підвищити шорсткість , то максимальна кількість променів починає значно впливати.

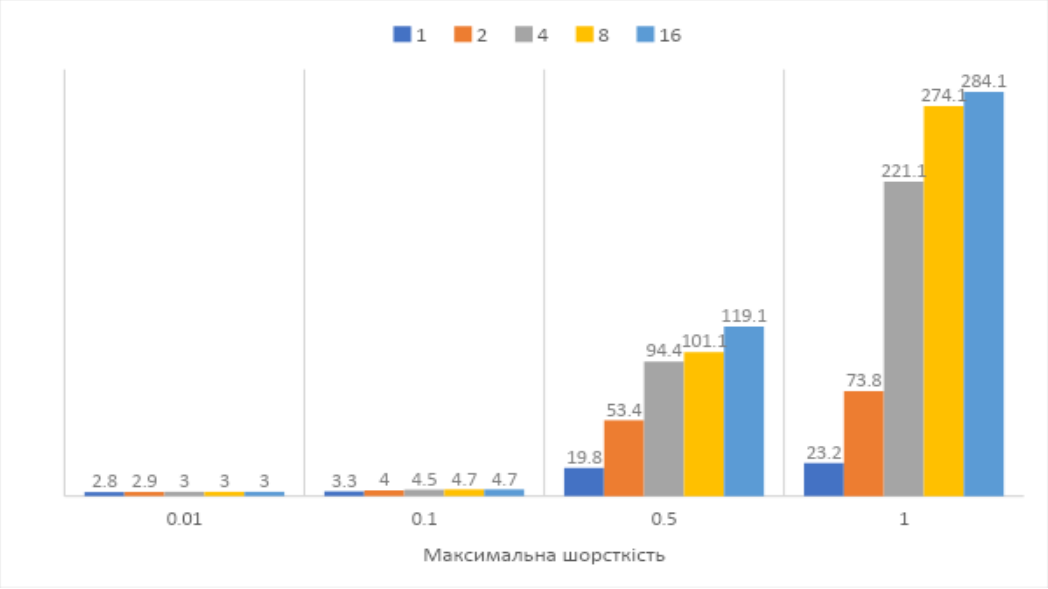

Рисунок 3.5 – Різниця часу кадру від базової для відображень для першої сцени

Результати замірів для відображень для другої сцени наведено в таблиці 3.6 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.6.

Таблиця 3.6 – Час генерації кадру для відображень для другої сцени

Кількість променів	Максимальна шорсткість			
	0.01	0.1	0.5	1
1	8.1	8.7	21.6	22
2	8.1	8.8	45.3	51.8
4	8.1	8.8	67.8	134
8	8.1	8.8	69	168
16	8.1	8.9	69	168

Результати для другої сцени сходні з результатами для першої. Кількість трикутників. Для невеликої шорсткості результати можна назвати відмінними, так як вони не перевищують 16.6 мс. Тобто можна доповнити сцену більшою кількістю ефектів.

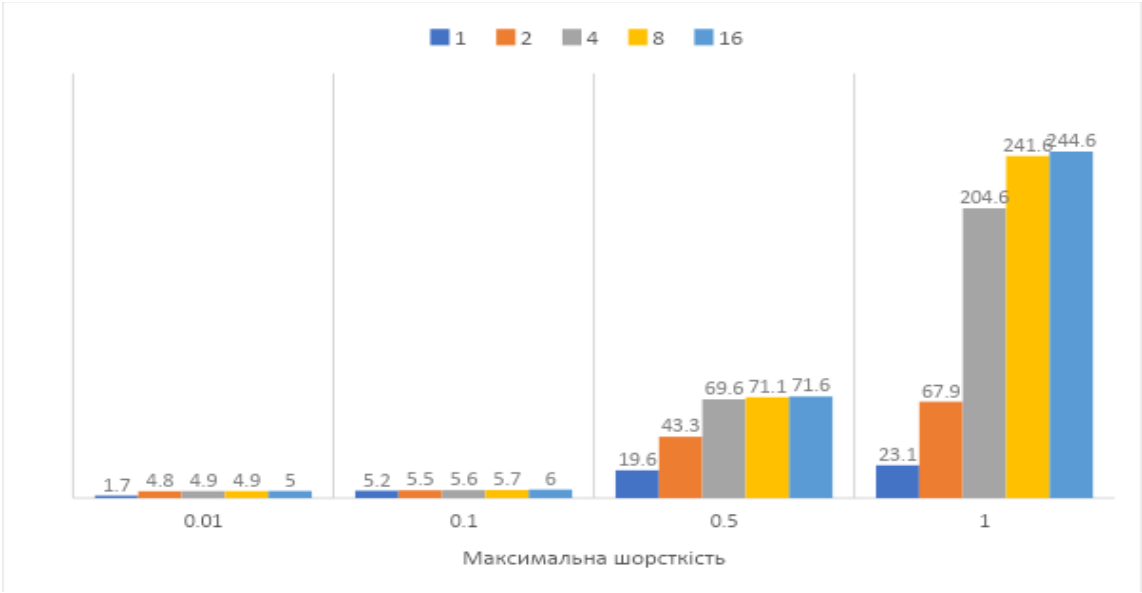

Рисунок 3.6 – Різниця часу кадру від базової для відображень для другої сцени

Результати замірів для відображень для другої сцени наведено в таблиці 3.7 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.7.

Таблиця 3.7 – Час генерації кадру для відображень для третьої сцени

Кількість променів	Максимальна шорсткість			
	0.01	0.1	0.5	1
1	11.1	11.6	26	29.5
2	11.2	11.9	49.7	74.3
4	11.3	12	76	211
8	11.3	12.1	77.5	248
16	11.4	12.4	78	251

Мінімальна максимальна шорсткість дозволяє мати якість відображення краще ніж методом SSLR та погіршення швидкодії не виводить час генерації зображення до рівня, коли такий метод буде неприємним. Підвищення максимальної шорсткості підвищує рівень деталізації відображеного.

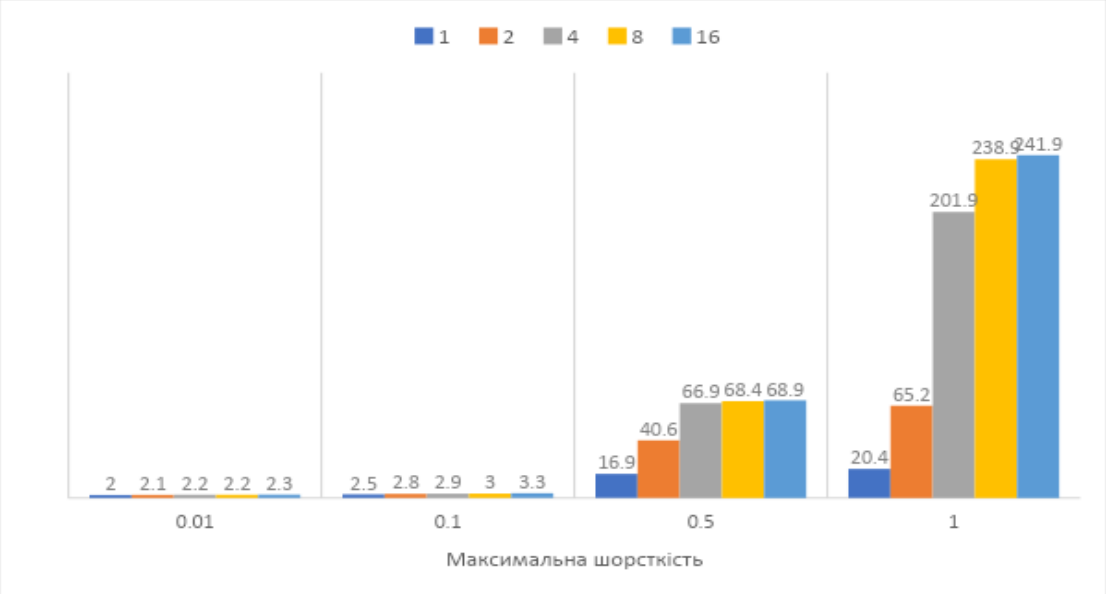

Рисунок 3.7 – Різниця часу кадру від базової для відображень для третьої сцени

Таке погіршення часу здається розумним, так як при маленькій максимальній шорсткості з великою ймовірністю робота алгоритму припиниться на першому ж

промені. Тобто за максимальним значенням максимальною шорсткості буде оброблено усі відбиття променю.

Результати замірів для глобального освітлення наведено в таблиці 3.8 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.8.

Таблиця 3.8 – Час генерації кадру для глобального освітлення

	Кількість проб			
	4	8	16	32
Сцена 1	21.7	23.1	26	31.9
Сцена 2	17.3	18.4	20.8	25.6
Сцена 3	21	22.6	25.8	32.2

На всіх сценах результат допустимий та не виходить за рамки 33.3 мс.

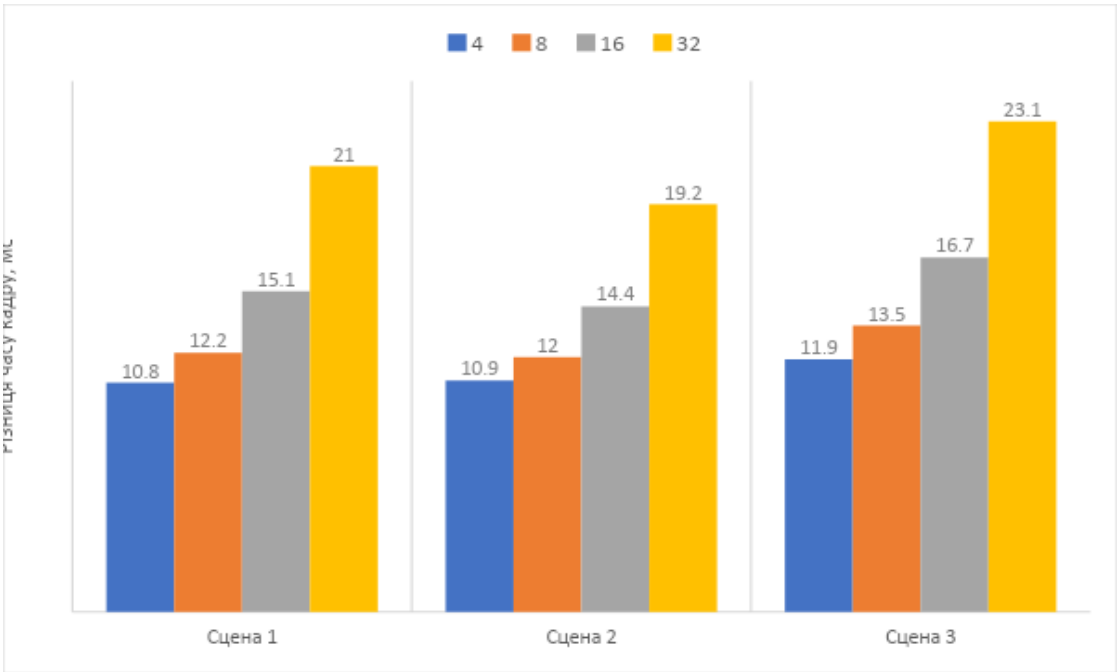

Рисунок 3.8 – Різниця часу кадру від базової для глобального освітлення

Результати замірів для затінення навколишнього світла наведено в таблиці 3.9 та графік різниці часу кадру з базовим результатом наведено на рисунку 3.9.

Таблиця 3.9 – Час генерації кадру для затінення навколишнього світла

	Кількість проб			
	1	2	4	16
Сцена 1	12.1	12.9	14.6	18.4
Сцена 2	7.9	9.4	10.3	13.2
Сцена 3	10.7	11.3	13.1	16.7

Генерація кадру методом трасування променів для затінення перевищує 16.6 мс тільки на сцені з найбільшою кількістю трикутників та з максимальним параметром якості. Для кількості проб в 1 та 2 помітні артефакти, тобто їх використання недоцільне.

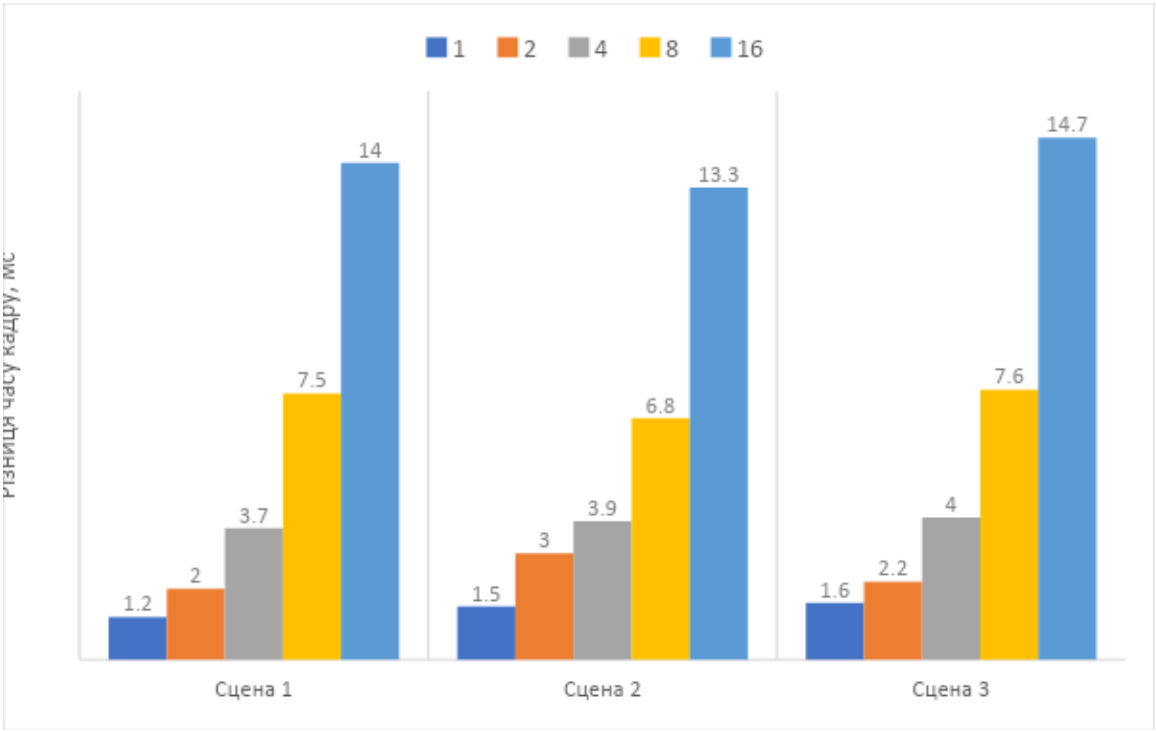

Рисунок 3.8 – Різниця часу кадру від базової для затінення навколишнього світла

Дослідження проводилось для розміру зображення 1920*1080 на такому апаратному забезпеченні:

– CPU: Ryzen 1700x 3.4 GHz;

– GPU: NVIDIA RTX 2070 1830 MHz;

– RAM: 32 гб 2133 MHz.

З результатів дослідження видно, які методи трасування променів скільки додають часу до генерації кадру в порівнянні зі стандартними. Нажаль жоден з методів не підвищив швидкодію алгоритму генерації кадру, але деякі методи «коштують» не набагато більше в той час як мають набагато більшу якість цільового ефекту. Це можна сказати про відображення та прозору геометрію та затінення навколишнього світла.

Ефекти, як тіні та глобальне освітлення додають великій час до генерації кадру, але для сцен зі схожими параметрами можуть бути використані не привішуючи 33.3 мс на думку розробника, якщо якість зображення буде важливішим за більшу швидкодію.

4 ОПИС РОЗРОБЛЕНОЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Вибір середовища реалізації програмного засобу

Для розробки програмного засобу обрано Unreal Engine 4 (див. рис. 4.1) версії 4.25, оскільки він має підтримку DirectX 12 та зручний інтерфейс роботи з трасуванням променів. Ще великим плюсами є підтримка візуальної скриптової мови[21] та редактор шейдерів, що дозволяє заощадити час розробки. Також платформа нативно підтримує мову програмування C++, тобто код компілюється разом з середою, а не інтерпретується нею.

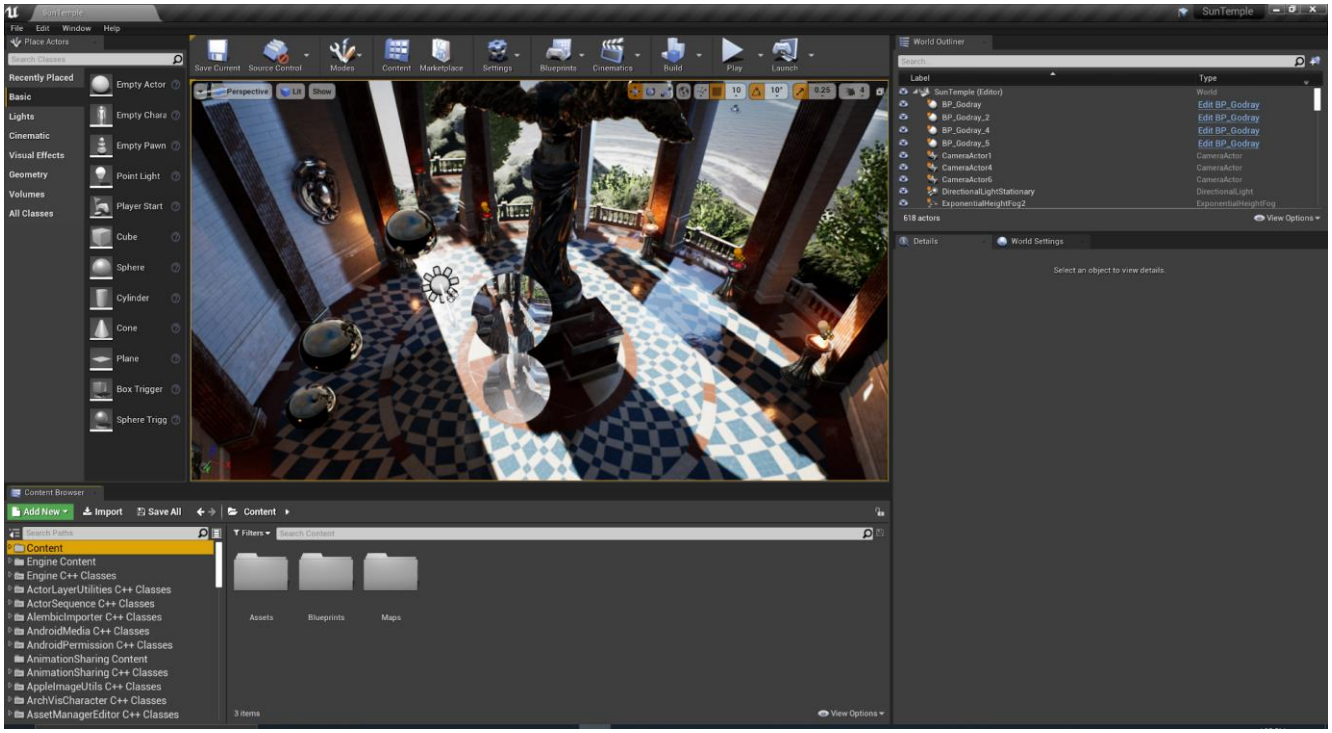

Рисунок 4.1 – Інтерфейс редактору UE4 v4.25

Також UE4 має відкритий вихідний код, що буде корисно при необхідності внутрішньої зміни роботи чи деякої специфічної для проекту оптимізації.

4.2 Інтерфейс розробленого програмного засобу

Даний програмний засіб має один робочий інтерфейс, який дозволяє змінювати параметри алгоритму генерації зображення, тестові сцени та також дозволяє одразу дивитись за залежністю швидкодії від змінюваних параметрів у вигляді графіків та числових показників (див. рис. 4.2).

Рисунок 4.2 – Робоче вікно першої сцени

Для проведення дослідження усі три тестові сцени одразу присутні у засобі, для їх відображення треба скористатись кнопками зміни камери, які знаходяться у верхній правій частині. Перша сцена видно на рисунку 4.2. Останні дві сцени можна бачити на рисунку 4.3 та 4.4 відповідно.

За основу сцен було взято учбовий проект UE4 “Sun template” та модифіковано його геометричними об’єктами та матеріалами для більш точного розуміння впливу методів на геометрію.

Рисунок 4.3 – Друга сцена

Рисунок 4.4 – Третя сцена

Для того, щоб змінити методи генерації зображення використовується кнопки, що знаходяться у лівій верхній частині. Поточний метод відображається

справа від кнопки. Якщо активовано метод трасування променів для етапу, то знизу відображається слайдер зміни параметрів відповідного етапу (див. рис. 4.5).

Рисунок 4.5 – Робоче вікно програми з розгорнутими слайдерами параметрів

Таким чином можна легко переключатися між сценами та досліджувати різні комбінації методів з різними параметрами.

На рисунку 4.6 показано приклад скрипту візуальної мови програмування для роботи з кнопками. При натисканні кожної кнопки відбувається зміна тексту, відображення слайдерів параметрів та зміна поточного методу генерації зображення. Наприклад, для зміни тіней на сцені знаходяться усі джерела освітлення та визивається функція зміни методу генерації тіней. Для зміни сцени трапляється зміна поточного активного «гравця», які були розставлені насамперед. Вони не мають контролю, щоб кожен раз відображався один й той самий ракурс з незмінною кількістю трикутників.

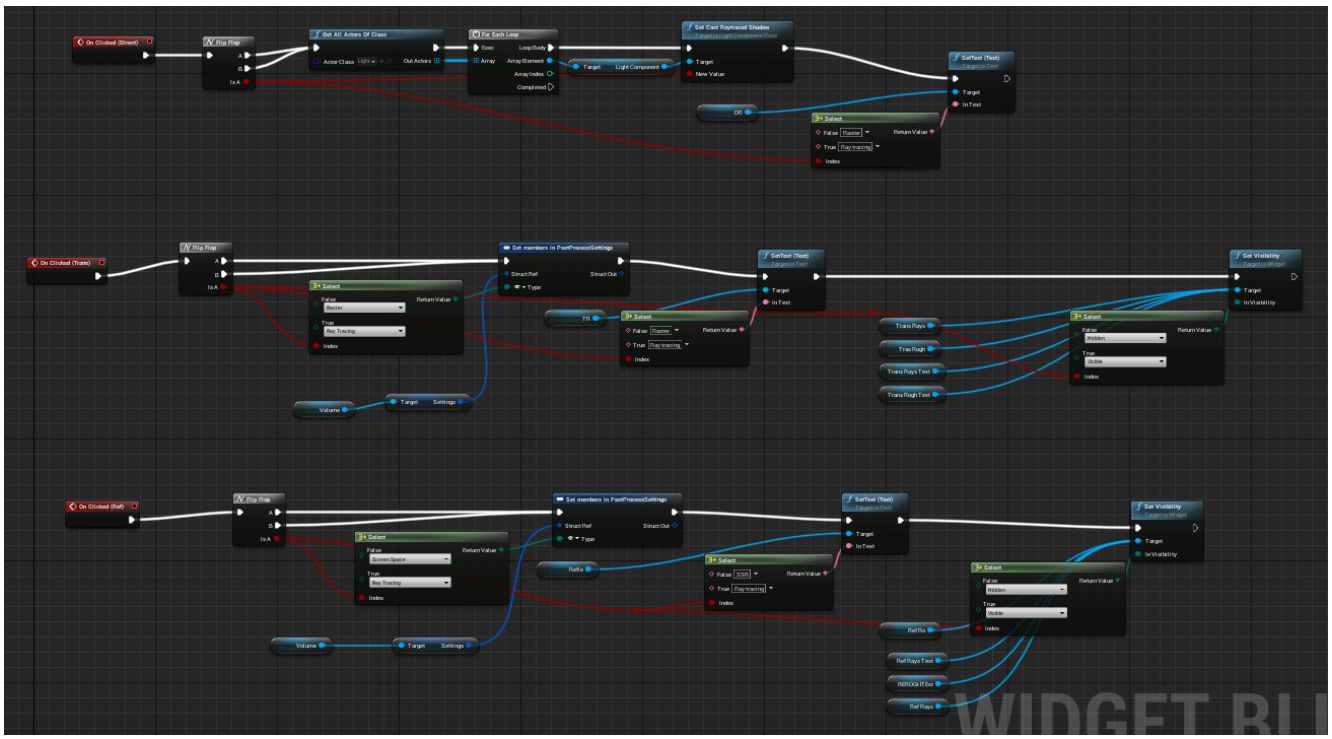

Рисунок 4.6 – Робота з параметрами методу

На даному етапі програмний засіб є досвідченою моделлю. З цієї причини він в першу чергу призначений для демонстрації роботи алгоритму і проведення дослідів на фіксованих сценах. Вся необхідна інформація знаходиться у самому програмному засобі та не вимагає ніякого імпорту зовнішніх ресурсів.

ВИСНОВКИ

В ході проведених досліджень була отримана модель алгоритму та програмного забезпечення для аналізу результатів дослідницької роботи і визначення подальших перспектив розробки.

Проаналізовано можливість використання методу трасування променів на різних етапах генерації зображення.

Дослідження генерації тіней методом трасування променів дало неоднозначний результат. З одної сторони є підвищення якості завдяки об’ємності джерела освітлення та час генерації кадру на більшості сцен не перевищив нижньої границі, а с другої сторони різницю часу з методом растеризації можна витратити на щось інше, тому як він більший в рази.

Відображення дали цікавий результат. При мінімальній якості вони забезпечують результат, який метод SSLR не може надати навить близько, при цьому швидкодія не набагато гірша (приблизно 1-2 мс). Підвищення параметрів якості хоча і приближає до повного ефекту дзеркала, але дуже швидко виводять час генерації кадру за межі допустимого.

Прозорість трасуванням променів впливає на швидкодію більше ніж відображення та за мінімальними показниками якості промінь може не «вибратися» з геометрії, що зробить її чорною. Також прозорість методом растеризації надає можливість для різних ефектів, таких як «гаряче повітря» біля вогню, тобто є локальним пост процесом. В майбутньому треба дослідити можливість генерації прозорої геометрії двома цими засобами.

Глобальне освітлення вирішує не виводить час генерації кадру за межі 33.3 мс, хоча впливає на нього набагато більше за відображення та не надає такої видимої переваги.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сушинський И., Новиков Ю., Technology of studying with using vr// SCIENCE, RESEARCH, DEVELOPMENT №17. С 8.

2. Van Verth J., Bishop. L. Essential mathematic for games and interactive applications: навч. посіб. – Boca Raton: CRC Press, 2013. – 599 с.

3. Suffern, K. Ray tracing from the ground up: навч. посіб. – Boca Raton: CRC Press, 2007. – 784 с.

4. Shirley P. Ray tracing in one weekend [Електронний ресурс]: Дата оновлення: 10.04.2020. URL: https://raytracing.github.io/books/RayTracingInOneWeekend.html (дата звернення: 25.04.2020).

5. Shirley P. Ray tracing: next week [Електронний ресурс]: Дата оновлення: 10.04.2020. URL: https://raytracing.github.io/books/RayTracingTheNextWeek.html (дата звернення: 25.04.2020).

6. Shirley P. Ray tracing: the rest of your life [Електронний ресурс]: Дата оновлення: 10.04.2020. URL: https://raytracing.github.io/books/RayTracingTheRestOfYourLife.html (дата звернення: 25.04.2020).

7. Vries J. Learn OpenGL [Електронний ресурс]: URL: https://learnopengl.com/ (дата звернення 20.04.2020).

8. Pharr, M. Physically based rendering from theory to implementation: навч. посіб. – MA: Elsevier, 2017. – 1225 с.

9. Hoffman N. Background: Physics and math of shading [Електронний ресурс]: URL: http://blog.selfshadow.com/publications/s2013-shading-course/hoffman/s2013_pbs_physics_math_notes.pdf (дата звернення: 29.04.2020).

10. Russel J. Basic theory of physically-based rendering [Електронний ресурс] URL: https://www.marmoset.co/posts/basic-theory-of-physically-based-rendering/ (дата звернення: 5.05.2020)

11. Kessenich J., Seller G., Shreiner D. OpenGL programming guide: навч. посіб. – Pearson education, 2017. – 1192 с.

12. Real shading in Unreal Engine 4 [Електронний ресурс] URL: http://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf (дата звернення: 05.05.2020).

13. Cook R., Torrance K., A Reflectance Model for Computer Graphics//SIGGRAPH '81 Proceedings of the 12th annual conference on Computer graphics and interactive techniques. 1981 г. New York – 307 с.

14. A Reflectance Model for Computer Graphics [Електронний ресурс] URL: http://graphics.pixar.com/library/ReflectanceModel/paper.pdf (дата звернення: 06.05.2020).

15. An Inexpensive BRDF Model for Physically-based Rendering [Електронный ресурс] URL: http://dept-info.labri.u-bordeaux.fr/ ~schlick/DOC/eur2.html (дата звернення: 06.05.2020).

16. Peatross J. Ware M. Physics of light and optics: навч. посіб. – Provo: Brigham Young University, 2015. – 340 с

17. Microfacet Models for Refraction through Rough Surfaces [Електронний ресурс] URL: https://www.cs.cornell.edu/~srm/publications/EGSR07-btdf.html (дата звернення: 06.05.2020).

18. Moving Frostbyte to physically based rendering 3.0 [Електронний ресурс] - Режим URL: https://seblagarde.files.wordpress.com/2015/07/course_notes_moving_frostbite_to_pbr_v32.pdf (дата звернення: 06.05.2020).

19. Akanine-Moller T., Haines E., Hoffman N. Real-time rendering: навчальний посбник. – Boca Raton: CRC Press, 2008. – 1045 с.

20. Nvidia Turing GPU architecture [Електронний ресурс] URL: https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf (дата звернення: 07.05.2020).

21. Hammerley points on the hemisphere [Електронний ресурс] URL: http://holger.dammertz.org/stuff/notes_HammersleyOnHemisphere.html (дата звернення: 07.05.2020).

22. Sampling with Hammersley and Halton Points [Електронний ресурс] URL: http://www.cse.cuhk.edu.hk/~ttwong/papers/udpoint/udpoint.pdf (дата звернення: 07.05.2020).

23. Sampling with Hammersley and Halton Points [Електронний ресурс] URL: http://www.cse.cuhk.edu.hk/~ttwong/papers/udpoint/udpoint.pdf (дата звернення: 07.05.2020).

22. Brenden S. Blueprints Visual Scripting for Unreal Engine: навч. посіб. – Birmingham: Packt Publishing, 2015. – 118 с.