ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра інформаційних систем
(повна назва кафедри)

КУРСОВА РОБОТА

на тему:

КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ СЦЕН З УРАХУВАННЯМ ЗАЛОМЛЕННЯ ПРОМЕНІВ СВІТЛА

Студентки 5 курсу, групи ПМІм-12с ,
напряму підготовки Комп'ютерні науки
Ковальчук С. А.
(прізвище та ініціали)

Керівник асист. каф. ІС, канд. фіз.-мат. наук
Стельмащук В. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____Оцінка: ECTS _____

Львів – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CG – Computer Graphics

RT – Ray Tracing

RC – Ray Casting

ВСТУП

На сьогодні комп’ютерне моделювання вважається однією з провідних галузей, яка використовується авіа-, машинобудівними фірмами під час конструювання та виготовлення виробів, які потребують високої точності, а основною задачею комп’ютерного моделювання є візуалізація, тобто створення зображення на основі опису (моделі) того, що потрібно зобразити. Поступово ця галузь увійшла у різні сфери повсякденного життя. До основних сфер застосування її технологій можна віднести такі:

– графічний інтерфейс користувача;

– спецефекти, візуальні ефекти, цифрова кінематографія;

– цифрове телебачення, відеоконференції;

– цифрова фотографія;

– цифровий живопис;

– візуалізація наукових та ділових даних;

–комп’ютерні ігри, системи віртуальної реальності (наприклад, тренажери з керування автомобілем, літаком тощо);

– системи автоматизованого проектування;

– комп’ютерна томографія.

Цей перелік постійно розширюється, бо використання комп’ютерного моделювання у різних сферах знань дає поштовх новим напрямкам.

Моделювання графічних об’єктів використовує математичні методи, досягнення у галузях фізики, природничих наук. Для реалістичного зображення сцен необхідно враховувати як геометрію складових об’єктів сцени, так і фізичні властивості матеріалів, з яких виготовлені об’єкти, біологічну будову живих об’єктів тощо.

Очевидно, що середовище, яке нас оточує, ми бачимо завдяки тому, що всі предмети відбивають світло, пропускають його через себе, випромінюють. Для передачі та збереження кольору в комп’ютерній графіці використовуються різні форми його представлення. У загальному випадку колір – набір чисел, які представляють собою координати у певній колірній моделі.

Для моделювання тривимірних об’єктів необхідно враховувати закони поширення світла у середовищах, властивості об’єктів відбивати, заломлювати, випромінювати світло.

У своїй роботі я займалась дослідженнями, які пов’язані з принципами побудови та з використанням законів поширення світла для моделювання просторових сцен.

1. 3D-МОДЕЛЮВАННЯ. ЕТАПИ СТВОРЕННЯ 3D-ОБ'ЄКТІВ

3D-моделювання – це процес створення моделі об'єкту у тривимірному просторі. Розробка 3D-об'єкту здійснюється в декілька етапів [11, 12]:

1. Створення ескізу.

2. Побудова (або моделювання) моделі.

3. Текстурування.

4. Розташування джерел освітлення та шейдинг.

Розглянемо детальніше кожний з етапів.

1.1 СТВОРЕННЯ ЕСКІЗУ

Перш ніж починати моделювання на комп'ютері, деякі художники створюють ескіз на папері, обирають анатомію моделі, кольори, текстури. Якщо проектується об'єкт, який існує у реальному житті, ескіз створюють за фотографіями або відео. Після того, як художник-постановник та режисер затвердять ескізи персонажів, їх передають спеціалістам з моделювання.

У випадку створення мультфільму група художників має розробити розкадровку, що надалі стає у нагоді як для надання зовнішнього вигляду персонажу, так і для анімації його рухів та міміки.

1.2 ПОБУДОВА МОДЕЛІ

Існує доволі багато видів моделювання, проте для створення моделей, що згодом підлягають анімації, використовують метод полігонального моделювання.

Полігональне моделювання – низькорівневе моделювання, яке дозволяє візуалізувати об'єкт за допомогою полігональної сітки [13].

Полігональна сітка складається з простих фігур (полігонів). Полігон являє собою трикутник або чотирикутник, який має вершини, ребра та грані. Модель, створену методом полігонального моделювання, можна назвати фігурою, яка складається з полігонів з різним ступенем перспективного спотворення, за рахунок чого об'єкт має певну форму.

Регулюючи кількість полігонів, з якої складається об’єкт, можна регулювати ступінь гладкості об’єкта. Наприклад, сфера з більшої кількості полігонів виглядає більш гладкою, ніж та, яка складається меншої кількості полігонів (рис. 1.1).

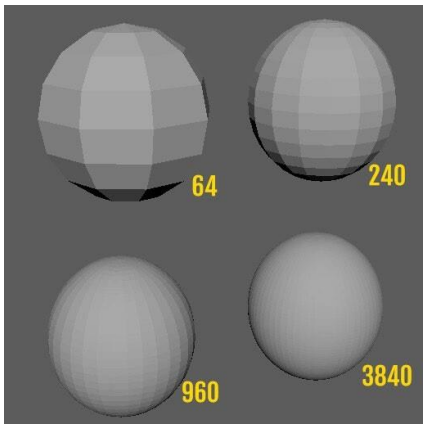

Рисунок 1.1 – Сфери, що складаються з різної кількості полігонів [14]

Також, чим більшою є кількість полігонів у моделі, тим більш деталізованою є модель. За таким критерієм моделі ділять на [15]:

- високополігональні (high-poly) – з великою кількістю полігонів;
- середньополігональні (mid-poly) – з середньою кількістю полігонів;
- низькополігональні (low-poly) – з малою кількістю полігонів.

Високополігональні моделі є більш фотореалістичними, ніж середньо- та низькополігональні, тому можуть використовуватися для створення фільмів та мультфільмів. Проте даний вид моделей потребує багато програмних та апаратних ресурсів. Рендеринг таких моделей займає багато часу, адже кожний кадр підлягає попіксельному прорахуванню, тому на створення мультфільмів може бути витрачено навіть декілька років.

Для комп’ютерних ігор високополігональні моделі не використовуються, тому що кадри б провантажувались доволі повільно. Щоб уникнути перевантаження програми та відеокарти комп’ютера, розробники виконують оптимізацію візуальної частини гри під можливості комп’ютерів користувачів.

За подібністю до високополігональної моделі створюють низькополігональну з полігонів-трикутників. Для побудови низькополігональної моделі залишають лише ті полігони, які впливають на

силует та форму. Елементи, які не видно, видаляють [15]. До моделі додають джерело світла, регулюють світлі та темні ділянки (шейдинг). Далі обидві моделі аналізують на напрямки нормалей. Нормалі – це вектори, які використовуються для визначення того, як світло відбивається від поверхні. Програма будує промені за напрямками нормалей низькополігональної моделі. Коли ці промені стикаються з високополігональною моделлю, програма обчислює, як відобразити ці промені, щоб вони слідували у напрямку нормалей високополігональної моделі. Інформацію про напрямки нормалей програма зберігає у текстуру під назвою «карта нормалей» [16]. Процес аналізу та зберігання називають «запікання» (рис. 1.2).

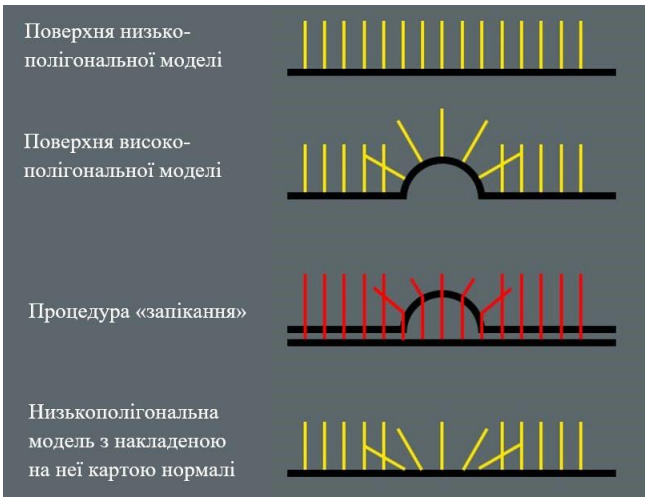

Рисунок 1.2 – Процедура «запікання» [17]

Далі карту нормалей накладають на низькополігональну модель. Таким чином виникає оптична ілюзія: низькополігональна модель відбиває світло так само, як і високополігональна (рис. 1.3).

а б в

Рисунок 1.3 – Приклад оптимізації моделі: а – високополігональна модель; б – низькополігональна модель; в – низькополігональна модель з накладеною на неї картою нормалей [18]

Полігональне моделювання не має прив'язки до реальних одиниць вимірювання, тобто даний метод моделювання не підходить для створення моделей з точними розмірами, наприклад, з кресленика або плану будівлі. Проте для створення відеоконтенту точність не має великого значення, більшу роль відіграє художня візуалізація об'єктів.

Якщо для створення відеоряду необхідне моделювання сцен, наприклад вулиці або місцевості, використовують метод автоматичної 3D-реконструкції за відеоматеріалами. Даний метод дозволяє отримати 3D-модель міської сцени у режимі реального часу.

Відеоматеріали знімаються багатокамерною системою в поєднанні з INS (інерціальною навігаційною системою) і GPS-вимірами. Для досягнення роботи у реальному часі виконують реконструкцію карт глибини з наборів зображень з подальшим злиттям зображень з картами глибини. Алгоритм є простим і швидким, може бути реалізований на графічних процесорах. В результаті виходить компактне і геометрично узгоджене представлення тривимірної сцени [19].

1.3 ТЕКСТУРУВАННЯ

Текстурування об'єкту – це відтворення фізичних якостей текстур та матеріалів, з яких виготовлено об'єкт для надання зображенню більшої реалістичності.

Текстура – це растрове зображення, що застосовується до полігональної моделі шляхом накладення з метою надання моделі фактурності, рельєфності і потрібного колірного забарвлення.

Якість текстурування об'єкта визначається такими одиницями як тексель. Тексель – це сукупність пікселів, що припадають на одиницю текстури.

Формат і роздільна здатність картинки текстури, що використовується, безпосередньо визначають якість підсумкового результату.

Розрізняють декілька видів текстурування [20]:

- Рельєфне текстурування;
- MIP-текстурування;

Рельєфне текстурування – технологія роботи з 3D-графікою, що дозволяє створити поверхню об’єкта, що моделюється, реалістичною [20]. Рельєфне текстурування нагадує процес накладення текстури на полігон. Відмінність полягає в тому, що під час звичайного накладення текстури виконується робота з кольором і змінюється лише колірне сприйняття полігона, а під час рельєфного текстурування додається відчуття рельєфу, об’ємності плоскому полігону. Ця техніка може додати деталізацію сцені без створення додаткових полігонів, що також є частиною оптимізації моделей для комп’ютерних ігор. Існують наступні види рельєфного текстурування [20]:

1. Створення рельєфної структури (bump mapping) – технологія, що дозволяє надати поверхні об’єкта, що моделюється, ефект рельєфу і ретельно її деталізувати. Даний ефект створюється шляхом віртуального зміщення пікселів за допомогою одноканальної карти висот (рельєф поверхні відображається у градаціях сірого кольору) і джерела світла. В результаті отримують ділянки з різним ступенем освітленості [20]. Bump mapping застосовується під час створення нерівних поверхонь, таких як виступи і западини [21].

2. Normal mapping – це метод зміни нормалі пікселя на базі кольорової карти нормалей. При цьому інформація про зміни зберігається у текселях. Даний метод є найбільш точним завдяки застосуванню трьох каналів текстур в карті нормалей [20].

3. Parallax occlusion mapping – метод локального трасування променів, який використовується з метою визначення висот і видимості текселя. Завдяки

цьому методу створюються більш сильні глибини рельєфу. Однак він не дає можливості ретельної деталізації об'єктів [20].

MIP-текстурування – метод, за якого під час накладення текстур застосовуються копії однієї і тієї ж ілюстрації текстури з різним ступенем промальовування деталей [20].

1.4 РОЗТАШУВАННЯ ДЖЕРЕЛ ОСВІТЛЕННЯ ТА ШЕЙДИНГ

Незважаючи на оформленість та текстуру моделі, без освітлених та затемнених ділянок кадри виглядатимуть ненатурально та без об’єму. Складність освітлення кадру полягає у достовірній імітації фізики розповсюдження світла, тому що промені мають багато властивостей, які важко візуалізувати. Наприклад, взаємодія світла з різними видами поверхонь, відбиття, заломлення, тіні, розсіювання, огинання об’єктів, зміна інтенсивності, яскравості, насиченості.

На ранніх етапах розвитку комп’ютерної графіки освітлення та його властивості виставляли вручну, тобто доступними були лише джерела прямого світла, які визначали лише напрям розповсюдження світла та на які ділянки об’єкта буде освітлено. Решту ефектів змінювали та доповнювали вручну. Згодом стало доступним глобальне освітлення, коли усі розрахунки розповсюдження світла у середовищі робить комп’ютер, причому для кожного окремого променя. Мова йде про технологію трасування променів.

Трасування променів (ray tracing) – технологія побудови зображення тривимірних моделей в комп’ютерних програмах, при яких відстежується зворотна траєкторія поширення променя. Спеціальний алгоритм відстежує шлях променя від об’єкта освітлення, а потім створює симуляцію того, як він взаємодіє з об’єктами: відбивається, заломлюється і так далі [22]. Спеціалісту залишається лише розставити джерела світла, їх напрям, віддаленість, а також налаштувати температуру світла та інтенсивність, а все інше візуалізує програма.

Шейдинг – процес, що здійснюють за допомогою шейдера – програми, що застововується у тривимірній графіці для визначення остаточних параметрів об’єкту або зображення. Шейдер може включати опис поглинання та розсіювання світла довільної складності, накладання текстури, відбиття та заломлення, затемнення, зміщення поверхні та ефекти пост-обробки. Шейдери, що програмуються, є дуже ефективними і дозволяють за допомого простих геометричних форм візуалізувати складні з вигляду поверхні [23].

1.5 ВИСНОВКИ ДО РОЗДІЛУ

В результаті огляду технологій створення 3D-об’єктів проаналізовано метод створення 3D-моделі – на основі полігонального моделювання. Виявлено, що в залежності від сфери використання та типу рендерингу (повільний процес візуалізації для кінофільмів або рендеринг у реальному часі для комп’ютерних ігор) модель потребує оптимізації. Для полегшення рендерингу моделі у реальному часі виконують перетворення з високополігональної у низькополігональну модель. Відсутність деталізації компенсується текстуруванням (накладанням карт нормалей, рельєфу та ін.) та шейдингом

2. АНАЛІЗ ОСНОВНИХ МЕТОДІВ РЕНДЕРИНГУ ЗОБРАЖЕНЬ

2.1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ПОБУДОВИ ЗОБРАЖЕННЯ.

Для рендерингу зазвичай використовують 4 основні обчислювальні методи. Кожна техніка візуалізації має свій власний набір переваг і недоліків. Серед них існують:

- Scanline rendering and rasterization
- Ray casting
- Ray tracing
- Radiosity
- Z-buffer

Scanline rendering – це алгоритм визначення видимої поверхні в комп'ютерній графіці, працюючий на основі рядків за рядками замість полігонів та пікселів. Кожен багатокутник, котрий підлягається візуалізації, спочатку сортують за верхніми координатами, у яких вони створюються спочатку, а потім кожен рядок сканування зображення обчислюється з використанням перетину лінії сканування з полігонами на передній частині відсортованого списку. Далі цей список оновлюється, щоб відкинути невидимі полігони, оскільки активна лінія сканування просувається вниз. [1]

Переваги:

- Сортування вершин вздовж нормальної площини сканування зменшує кількість порівнянь між ребрами;
- Відсутність потреби переводу координат всіх вершин з основної пам'яті в робочу;

Можливість інтегрування алгоритму з багатьма іншими графічними методами, такими як модель відображення Фонга або алгоритм Z- буфера; Недоліки:

- За допомогою даного методу немає можливості отримати реальні відображення та заломлення;
- Застарілість алгоритму;
- Поступається альтернативним методам у реалістичному рендерингу зображення;
- Складна реалізація, оскільки система працює з об’єктним кодом;

Ray casting – це алгоритм прокладання променів, у якому змодельована геометрія аналізує кожен рядок та кожен піксель з точки зору назовні об’єкта. У місці, де перетинається об’єкт, значення кольору в точці може бути оцінено за допомогою декількох методів. У найпростішому випадку значення кольору об’єкта в точці перетину стає значення цього пікселя. Також колір може бути визначений з текстурної карти. Більш складним методом є зміна значення кольору за допомогою коефіцієнта освітленості, але без розрахунку відношення до імітованого джерела світла. Інший метод робить розрахунок на кут падіння світлових променів від джерела світла, а також виходячи із зазначених інтенсивностей джерел світла, обчислює значення пікселя. [2]

Переваги:

- Легко реалізувати;
- Інтуїтивно створює алгоритм Line of Sight;

Недоліки:

- Повільний порівняно з іншими методами;

- При відливанні лише декількох променів, квадрати поблизу джерела будуть відвідуватися багато разів;

- Багато артефактів, навіть у звичайних ситуаціях;

Radiosity – метод радіосигналу синтезу зображень у середині 1980-х років. Система дивиться виключно на баланс світла або енергії в такому закритому середовищі, в якому вся енергія, що випромінюється або відбивається від даної поверхні, враховується шляхом відображення або поглинанням іншими поверхнями. За допомогою цього методу можна визначити величину поверхневого радіозв'язку та швидкість з якою енергія виходить з поверхні. Для обчислення радіозв'язку для кожної поверхні використовуються значення кількості взаємодій енергії між ними. Зазвичай, за допомогою відповідних маніпуляцій, радіосигнал може генерувати зображення на льоту на відстані 15-20 кадрів в секунду. Розрахунки радіозв'язку не залежать від точки зору але збільшують обчислення, що корисні для будь-яких точок зору. Якщо відбувається невелика перестановка об'єктів радіозв'язку в сцені, однакові дані радіосигналу можуть бути повторно використані для ряду кадрів, що робить радіозв'язок ефективним способом поліпшення відливання від променів, без серйозного впливу на загальний час візуалізації. Через це радіозв'язок є основним компонентом провідних методів візуалізації в реальному часі і використовується від початку до кінця для створення великої кількості відомих останніх анімаційних 3D-мультфільмів. [3]

Переваги:

- обчислює дифузні взаємозв'язки між поверхнями;
- забезпечує перегляд незалежних рішень для швидкого відображення довільних переглядів;
- відтворює відносно реалістичні зображення;

Недоліки:

- 3D сітка вимагає більше пам’яті, ніж оригінальні поверхні;

алгоритм відбору проб поверхні є більш сприятливим до артефактів зображення, ніж трасування променів;

- не враховує дзеркальних відображень або ефектів прозорості

Z-buffer – алгоритм, який використовується для визначення видимої поверхні та є основою процесу візуалізації сканованої ліній. Головна ідея використання Z-buffer полягає в тому, щоб перевірити відстань від спостерігача кожної поверхні задля розробки найближчої поверхні кожного об’єкта. Якщо два об’єкти мають різні значення z-глибини вздовж однієї проектованої лінії, то вище значення знаходиться попереду, а позаду залишається ближча поверхня або об’єкт. Застосування цього підходу дозволяє нам відображати сцени за допомогою візуалізації сканової лінії. [4]

Переваги:

- простий у використанні;
- може бути легко реалізований в об’єкті або зображенні;
- може виконуватися швидко, навіть з багатьма полігонами;

Недоліки:

- займає багато пам’яті;
- неможливо створити прозорі поверхні без додаткового коду;

2.2. ОБҐРУНТУВАННЯ ОБРАНОГО МЕТОДУ

Темою курсової роботи було обрано створення системи побудови зображення 3D моделей з відстеженням зворотної траєкторії променя, що має такі переваги:

- Елегантність алгоритму;

- Гарна апроксимація відображень;
- Здатність працювати з великою кількістю явищ;
- Найреалістичніший і бажаний режим візуалізації на сьогоднішній день;

Імітування руху світла в реальному світі;

- Точність надання прямого освітлення, тіней, дзеркального відображення та ефекту прозорості. Ефективна пам’ять;

У методі трасування променя, кожен піксель сцени, один або більше променів світла відстежуються від камери до найближчого 3D-об’єкста. Світловий промінь проходить через задане число “відскоків”, що може включати відбиття або заломлення в залежності від матеріалу у 3D-сцені. Кожен колір пікселів обчислюється алгоритмічно на основі взаємодії світлового променя з об’єктами на його трасі. Ray tracing здатний більшого фотореалізму ніж інші методи рендерингу зображення, але його єдиним недоліком є складна обчислювальна спроможність.

2.3 ВИСНОВКИ ДО РОЗДІЛУ

В цьому розділі були розібрані основні існуючі методи побудови зображення та їх переваги та недоліки. Був обгрунтований вибір теми дипломного проекту, та пояснено домінування обраного методу рендерингу зображення відносно альтернативних методів. Також детально розібрали причини вибору конкретного середовища розробки.

3 ТЕХНІКА ВІЗУАЛІЗАЦІЇ ВІДСТЕЖЕННЯ ПРОМЕНІВ

3.1. ОГЛЯД ТЕХНІКИ ВІЗУАЛІЗАЦІЇ ВІДСТЕЖЕННЯ ПРОМЕНІВ

Трасуванням променів називають метод обчислення видимості між точками. Транспортні алгоритми призначені для імітації способу поширення світла через простір при взаємодії з об’єктами. Їх використовують для обчислення кольору точки сцени. Трасування променів не є легким транспортним алгоритмом, це лише техніка обчислення видимості між точками. [4]

Растрове зображення зроблено з пікселів. Один із способів відтворення 3D-сцени полягає в тому, щоб якось рухати це растове зображення вздовж площини зображення віртуальної камери, і знімати промені (Рис 3.1).

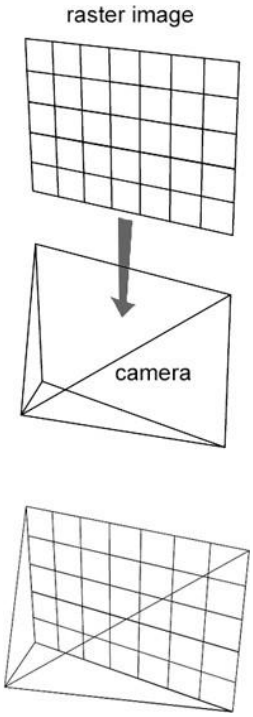

Рис. 3.1 – рух растового зображення перед площиною зображення камери [5]

За допомогою закидання променів, що походять від ока (положення камери), та які проходять через центр кожного пікселя ми знаходимо об’єкт зі сцени на якому ці промені перетинаються.

Якщо піксель щось і «бачить», то він бачить саме об’єкт, що знаходиться прямо по напрямку, вказаного променем. Напрямок променя можна побудувати, якщо простежити лінію від походження камери до центру пікселя (Рис 3.2).

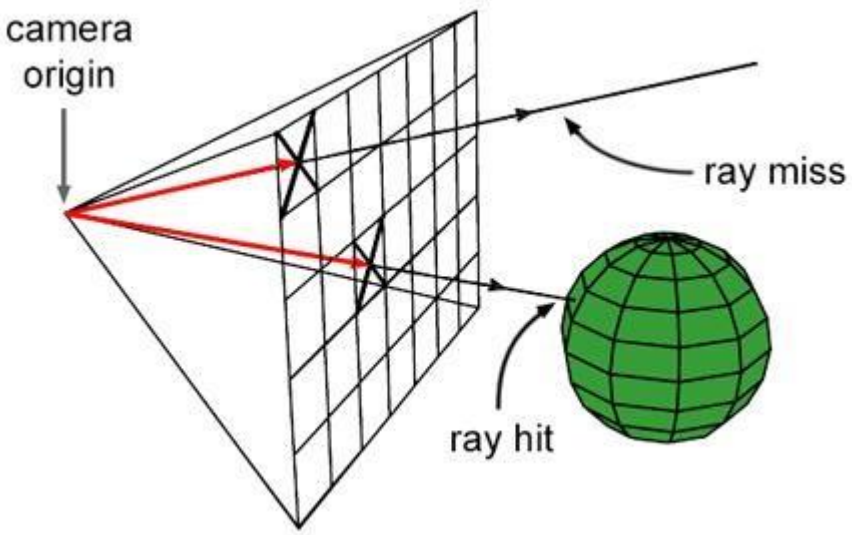

Рис. 3.2 – промінь може потрапити в геометрію сцени або пропустити її [5]

Тепер, коли ми знаємо, що бачить піксель, треба лише повторити цей процес для кожного пікселя зображення. Налаштовуючи колір пікселя відповідно до кольору об’єкта, в котрому кожен промінь проходить через центр кожного пікселя, ми можемо сформувати зображення сцени. Цей метод вимагає циклічного перегляду всіх пікселів у зображенні і і перетворення променя на сцену для кожного з цих пікселів. На другому етапі, етапі перетину, потрібно виконати цикл над усіма об’єктами сцени, щоб перевірити, чи перетинає промінь будь-яких з цих об’єктів.

Слід зауважити, що деякі промені можуть взагалі не перетинати будь-яку геометрію. Наприклад, як показано на малюнку 2.2, один з променів не перетинає сферу. У цьому випадку, як правило, треба залишити чорний колір пікселя, або встановити його на довільний інший колір. В наведеному вище коді ми ми задаємо колір пікселя кольором об’єкта в точці перетину.

Об’єкти в реальному світі виглядають дуже складно. Їх яскравість змінюється залежно від кількості світла, яке вони отримують, деякі з них блискучі, інші матові тощо. Мета фото-реалістичної візуалізації полягає не лише у точному зображенні геометрії, але й в імітуванні зовнішнього вигляду об’єктів переконливо (Рис 3.3).

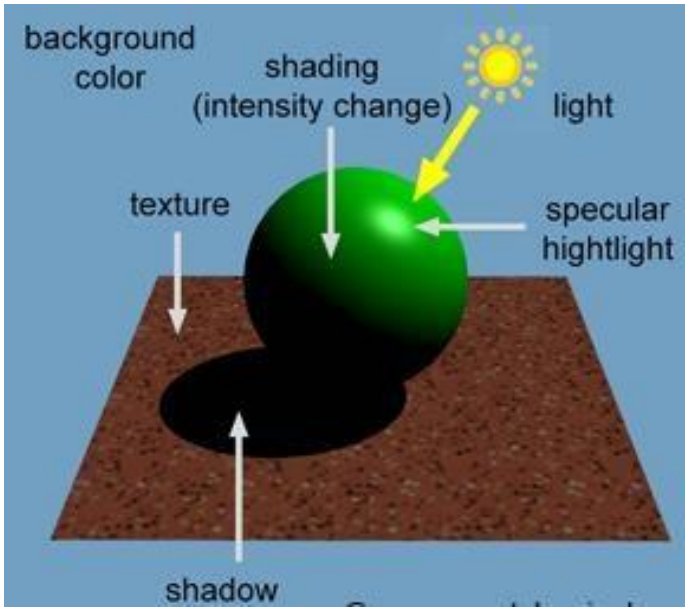

Рис. 3.3 – ілюстрація, що показує, як важко відтворити реальність [6]

Цей процес передбачає щось більш складне, ніж просто повернення постійного кольору. У комп’ютерній графіці, завдання визначення фактичного кольору об’єкта в будь-якій заданій точці на його поверхні, знаючи про ймовірні зовнішні (кількість світла, що отримує об’єкт) та внутрішні параметри (матерія об’єкта, блискучі кольори), називається затіненням.

Трасування променів вважається орієнтованим на зображення. Зовнішні петлі повторюють всі пікселі зображення і внутрішній цикл перебирає об’єкти сцени. Алгоритм растеризації є відносно орієнтованим на об’єкт. Він вимагає циклічного перегляду всіх геометричних примітивів сцени та проектування цих примітивів на екран.

Трасування променів слід використовувати для двох етапів візуалізації: видимість та затінення. Це пов’язано з тим, що растеризація

підходить тільки для видимості, і краща ніж трасування променів лише у швидкодії.

Зазвичай рендеринг займається обчисленням видимості між точкою в просторі і першою видимою поверхнею в заданому напрямку, або видимістю між двома точками. Перша задача призначена для вирішення проблеми видимості, друга - для вирішення таких проблем, як затінення. Растеризація дуже добре підходить для пошуку першої видимої поверхні, але неефективна для вирішення задачі видимості між двома точками. Трасування променів дозволяє ефективно обробляти обидва випадки. Пошук першої видимої поверхні корисний для вирішення проблеми видимості. Таким чином, і трасування променів, і растерізація підходять для вирішення цієї задачі. Затінення вимагає вирішення проблеми видимості між поверхнями, яке використовується для обчислення тіней,коли використовуються світлові області, і глобальні ефекти освітлення, такі як відображення, заломлення, непрямі відображення та непрямі дифузії. Таким чином, для цієї конкретної частини процесу візуалізації трасування променів є більш ефективним, ніж растеризація. Але майте на увазі, що будь-яка техніка, яка обчислює видимість між точками, може бути використана для затінення та вирішення проблеми видимості.

3.2 ЕТАПИ ОБЧИСЛЕННЯ ФОТОРЕАЛІСТИЧНИХ 3D-ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ТРАСУВАННЯ ПРОМЕНІВ.

Використання трасування променів для обчислення фотореалістичних зображень може бути поділено на 3 етапи:

- Відливання променів на сцену;
- Тестування перехресть променевої геометрії;
- Затінення;

Перше, що потрібно зробити, щоб створити зображення за допомогою трасуванням променів, це подати промінь для кожного пікселя зображення. Ці промені називаються камерами або первинними променями. Вторинні

використовуються для пошуку точок сцени, що знаходяться в тіні, або для обчислення ефектів затінення, таких як відображення або заломлення. Коли первинний промінь відкидається в сцену, наступний крок полягає в тому, щоб знайти його перетин з будь-яким об’єктом у сцені.

Тестування променя на перетин з будь-яким об’єктом у сцені вимагає циклічного перегляду всіх об’єктів сцени і перевірки поточного об’єкта на промінь

Геометрія в 3D може бути визначена різними способами. Прості форми, такі сфера, диски, площини, можуть бути визначені математично або параметрично. Форми більш складних об’єктів можуть бути описані лише за допомогою полігонових сіток, поверхонь підрозділів поверхонь NURBS. Основна проблема з другою категорією об’єктів полягає в тому, що для кожної підтримуваної поверхні необхідно реалізувати методи перетину променевої геометрії. Наприклад, поверхні NURBS можна промалювати прямо, хоча рішення для цього дуже відрізняється від того, що використовується для тестування перетину між променем і багатокутною сіткою. Таким чином, для того щоб підтримувати всі типи геометрії, треба написати одну з перехресних процедур променевої геометрії для кожного підтримуваного типу, але це потенційно збільшує складність коду програми. Альтернативне рішення полягає в перетворенні кожного типу геометрії в одне і те ж внутрішнє уявлення, яке майже завжди буде триангульованою сіткою багатокутника (Рис 3.4).

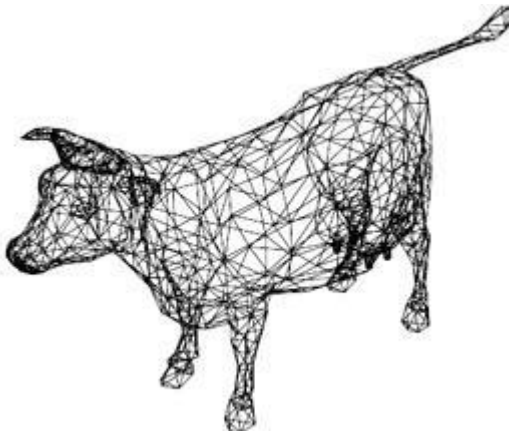

Рис. 3.4 – триангульовані сіткі зручніші для трасування променів, ніж інші типи геометрії [8]

Трапляється, що перетворення майже будь-якого типу поверхні на сітку багатокутника дуже просте. Звідси перетворення полігональної сіткі в триангульовану також досить просте. Крім цього, трикутники є вигідними з той точки зору, що вони можуть бути використані як базовий геометричний примітив для трасування променів та растеризаації. Обидва алгоритми люблять трикутники через те, що вони мають цікаві геометричні властивості, яких не мають інші типи.

Таким чином, полігональні сітки або інші типи поверхонь перетворюються на трикутники. Це можна зробити або перед завантаженням геометрії в пам’ять програми, або під час візуалізації. Тепер не тільки кожен промінь камери повинен бути протестований на кожному об’єкті сцени, але і на кожному окремому трикутнику, який складається з кожного багатокутника в сцені.

Це означає, що час, необхідний для відтворення сцени трасування променів, прямо пропорційний кількості трикутників, які містить сцена. Всі трикутники повинні бути збережені в пам’яті, і кожен з них повинен бути перевірений на додавання променя в сцену. Висока обчислювальна вартість трасування променів є головним недоліком алгоритму.

3.3.ОПИС АЛГОРИТМУ ТРАСУВАННЯ

Для реалізації алгоритму трасування променів і отримання результуючого зображення, яке складається 3 геометричних примітивів у просторі R^3, знадобляться наступні програмні моделі:

- Модель променю у тривимірному декартовому просторі;
- Модель променевого емітера - камера;
- Математичні моделі графічних примітивів;

- Модель точкового джерела світла.

Для побудови первинного променю і визначення перших перетинань 3 об'єктами сцени вводяться поняття джерела променів і картинної площини. В основі моделі променевого емітера лежить механізм спрощеної камери-обскури (рис. 3.5) з безкінечно малим отвором, крізь який світло потрапляє на область огляду.

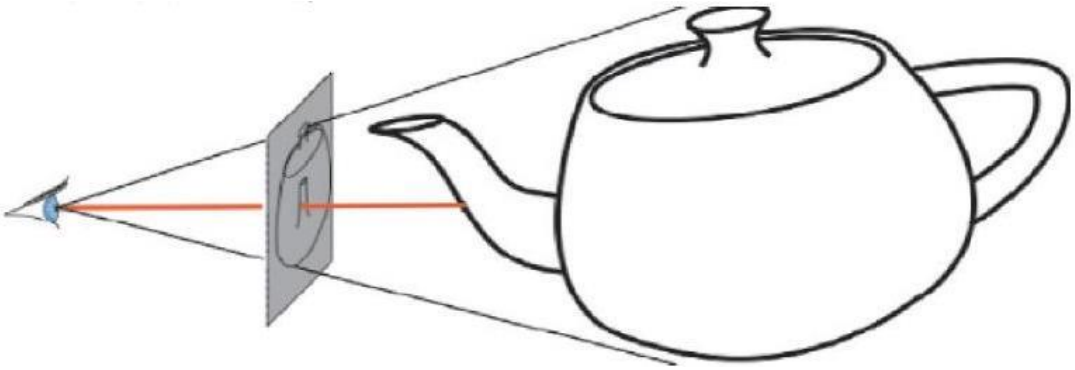

Рис. 3.5. Принцип побудови зображення. Механізм спрощеної камериобскури [3]

Визначимо плоску область огляду як масив пікселей $\{n \times m\}$, де n і m - кількість пікселей в ширину і висоту відповідно, використовуючи заготовлені структури даних.

Встановимо початок координат у точку $O(x_0, y_0, z_0)$, центр камери у точку $C(x_c, y_c, z_c)$. Визначимо фокусний центр камери точкою $F(x_F, y_F, z_F)$. Вектор $\vec{\omega} = \overrightarrow{CF}$ визначає напрямок камери.

Для коректного визначення напрямку розповсюдження променю від позиції камери до центру кожного пікселя плоскої області огляду треба сформувати ортогональний базис $\{\vec{u}, \vec{v}, \vec{\omega}\}$ за допомогою операції векторного перемноження:

$$\vec{u} = \vec{\omega} \times \vec{y}_{(0,1,0)}$$
$$\vec{v} = \vec{u} \times \vec{\omega}$$

Напрямок первинного променю $\vec{R}$ (рис. 3.6) визначається наступними формулами:

$$\alpha = \tan\left(\frac{fovx}{2}\right) \cdot \left(\frac{i - \left(\frac{n}{2}\right)}{\frac{n}{2}}\right)$$

$$\beta = \tan\left(\frac{fovy}{2}\right) \cdot \left(\frac{\left(\frac{m}{2}\right) - j}{\frac{m}{2}}\right)$$

$$\vec{R} = \overrightarrow{OC} + \frac{\vec{\omega} + \alpha\vec{u} + \beta\vec{v}}{|\vec{\omega} + \alpha\vec{u} + \beta\vec{v}|}$$

де α, β - величини зміщення променю по осям X, Y області огляду; i, j - цілочисельні координати пікселя, для якого генерується промінь; fovx, fovy - горизонтальний і вертикальний кути обзору камери;

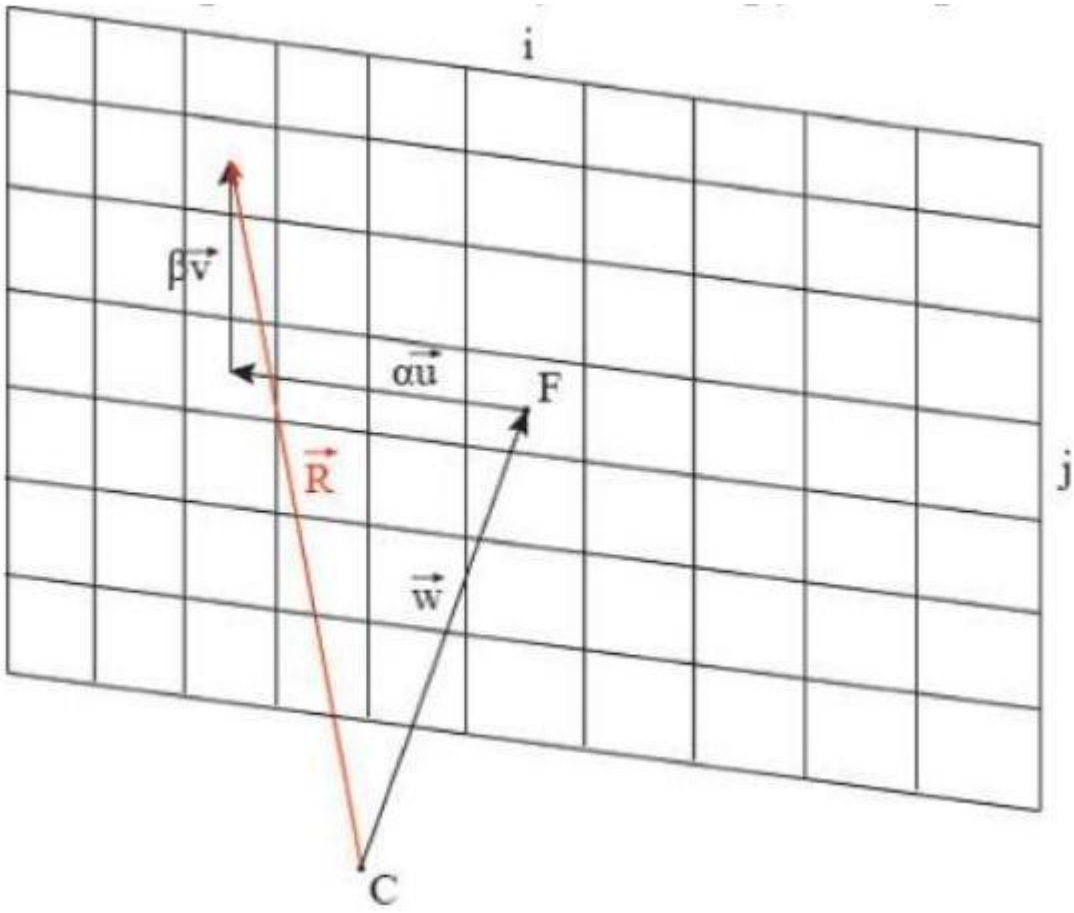

Рис. 3.6. Обчислення напрямку розповсюдження променю від позиції камери до центру кожного пікселя координатної площини

Нехай

$$\vec{d} = \frac{\vec{\omega} + \alpha\vec{u} + \beta\vec{v}}{|\vec{\omega} + \alpha\vec{u} + \beta\vec{v}|^2}.$$

Таким чином можна записати векторне рівняння для пучка променів, які виходять 3 точки O з направляючими векторами $\vec{d}$:

$$\vec{R}_{ij}(t) = \overrightarrow{OC} + t\vec{d},\ t \geq 0,$$

де t - відстань від початку променю до будь-якої точки на ньому; $\vec{d}$ - вектор напрямку розповсюдження променю.

Нижче наведена послідовність рівнянь для визначення перетинів 3 наступними об'єктами, які використовуються у демонстраційній програмі: площина та сфера.

Розглянемо векторні рівняння площини, яка проходить через фіксовану точку Q перпендикулярно до вектору нормалі $\vec{n}$:

$$\vec{n} \cdot \overrightarrow{QP} = 0,$$

де $\mathrm{P}(\mathrm{x_p}, \mathrm{y_p}, \mathrm{z_p})$ – довільна точка площини.

Вирішення системи рівнянь (6), (7)

$\begin{cases} \overrightarrow{OC} + t\vec{d} \\ \vec{n} \cdot \overrightarrow{QP} = 0 \end{cases}$ визначає значення параметра t, який відповідає точці перетинання променю площиною:

$$t = \frac{\vec{n} \cdot \overrightarrow{OQ}}{\vec{n} \cdot \vec{d}}.$$

Якщо $t < 0$, площина знаходиться позаду камери і промінь її не перетинає. Якщо $t \geq 0$, точка перетину знаходиться на відстані $|\overrightarrow{OC} + t\vec{d}|$ від центру розміщення променевого емітера. Якщо $\vec{n} \cdot \vec{d} = 0$, то промінь проходить паралельно площині і не перетинає її.

Аналогічним чином вирішується задача про перетин променю і сфери з радіусом R та центром у точці $Q_d(x_s, y_s, z_s)$:

$$(x - x_s^2) + (y - y_s^2) + (z - z_s^2) = R^2.$$

Значення параметра t, при якому промінь перетинає сферу, визначаються коренями квадратного рівняння:

$$at^2 + bt + e = 0$$

де $a = |\vec{Q}_s|^2; b = 2\left((\overrightarrow{Q_sC}) \cdot \vec{d}\right); e = |\overrightarrow{Q_sC}|^2 - R^2.$

В залежності від значень a, b, e можливі наступні варіанти вирішення рівняння (9) відносно t :

- Р Рівняння має два додатних дійсних корені. Промінь перетинає сферу у двох точках (входить і виходить 3 неї). Відповідно далі необхідно працювати 3 перетинами 3 найменшою дистанцією.

- Рівняння має два однакових додатних або від'ємних дійсних корені. Промінь проходить по дотичній до сфери.

- Р Рівняння має додатний і від'ємний дійсні корені. Променевий емітер знаходить у самій сфері і промінь тільки виходить 3 неї.

- Рівняння має комплексні корені. Промінь не перетинає сферу.

Встановивши мінімальне і максимальне значення параметра t, визначимо найближчу і найвіддаленішу площини відтинання тривимірної сцени, тим самим завершивши конфігурацію усіченої піраміди огляду камери.

Реалізувавши алгоритм «кидання променів», описаний на попередньому етапі, сформуємо базову сцену, використовуючи підготовлені графічні примітиви. Зробимо наступні налаштування: площина: точка $(0, -1, 0)$, вектор нормалі $(0,1,0)$; сфера: точка $(0,0,0)$, одиничний радіус; джерело світла: точка $(-7{,}10, -10)$. Встановимо відповідно перерахованим об'єктам коричневий, зелений та білий кольори.

Представлене вихідне зображення є результатом перевірки перетинання променів 3 об'єктами сцени. Для досягнення ефекту об'єму можна використати відомі моделі освітлення і затінення, зокрема: - Ізотропні моделі дифузного освітлення Ламберта [4] і Фонга [5], остання 3 яких доповнює розсіяне освітлення поверхні блискучою складовою;

- Моделі затінення поверхонь, яка візуалізує тіньовий об'єм.

У найпростішій своїй реалізації технологія тіньового об'єму у трасуванні променів полягає в генерації «тіньового променю» 3 точки перетину сцени первинним променем у напрямку джерела світла. Без врахування коефіцієнта затухання світла, якщо «тіньовий промінь» досягнув джерела світла, не перетинаючи при цьому інших об'єктів сцени, то точка освітлюється. В іншому випадку, якщо знайдено хоча б один перетин - точка знаходиться у тіні. При правильній програмній реалізації моделей освітлення і затінення результуючі зображення наближаються до реалістичних (рис. 3.7).

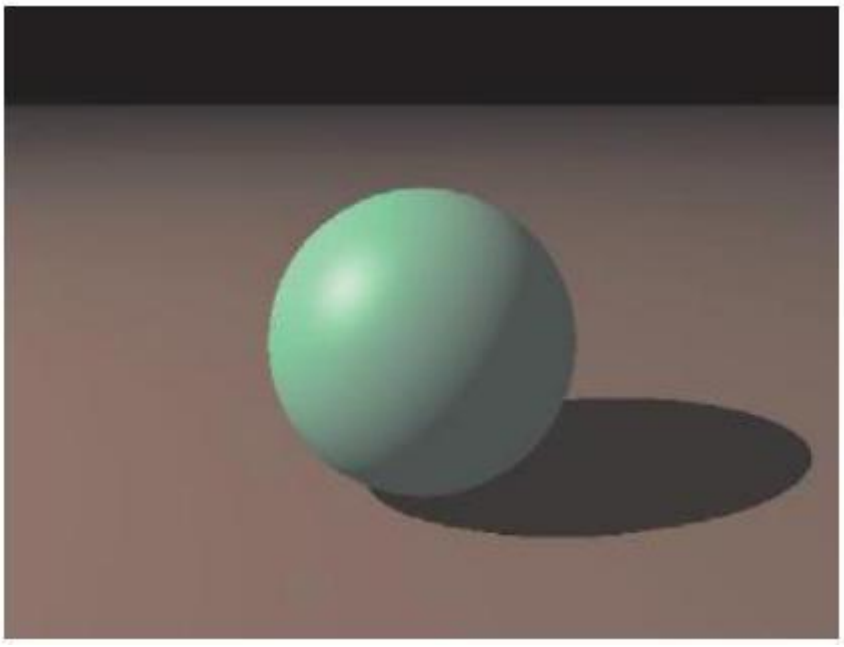

Рис. 3.7. Результат генерації тіней і роботи моделі освітлення БлінаФонга. Результат роботи демонстраційної програми

Для розрахунку характеристик кольорової компоненти відбиття (віддзеркалення) у точці перетину первинного променю і об'єкту необхідно

створити новий промінь у цій точці і направити його по нормалі, що відновлена 3 цієї точки по відношенню до поверхні об’єкта, потім рекурсивно повторити загальну процедуру отримання кольору для первинного променю. У даній роботі використовується модель ідеальних віддзеркалень. Саме тому для виключення нескінченного формування променів для взаємного розрахунку кольору відбиття об'єктів, поверхні яких будуть наділені такою властивістю, необхідно ввести обмежувальне правило: поверхня повинна відбивати не всю енергію променю, а тільки певний відсоток.

Після імплементації алгоритму генерації додаткових променів для формування віддзеркалень ускладнимо сцену, додавши примітив (сферу) і присвоївши шаховий шаблон площини для надання наглядності роботи нового алгоритму, отримаємо наступний результат (рис. 3.8).

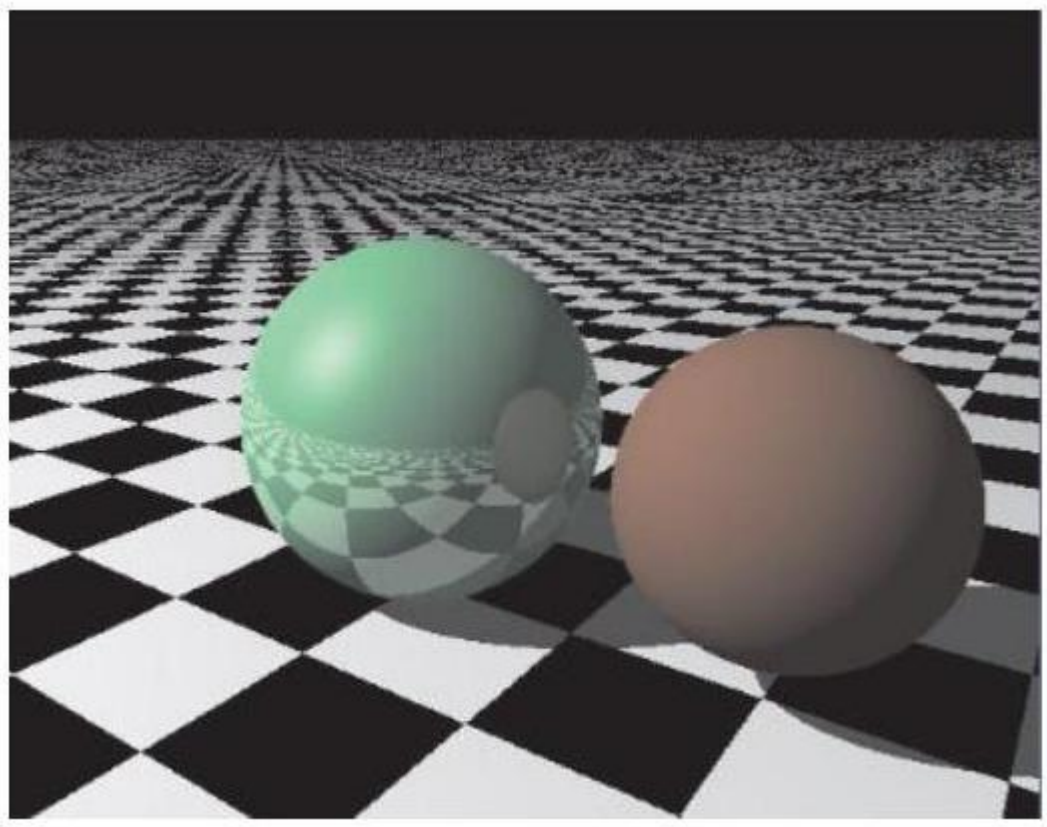

Рис. 3.8. Вихідне зображення після генерації дзеркальних відбиттів на поверхні сфери. Результат роботи демонстраційної програми Серйозним недоліком методу є невисока продуктивність. Метод растеризації і сканування рядків використовує когерентність даних, щоб розподілити обчислення між пікселями. В той час як метод трасування променів кожного разу починає процес визначення кольору пікселів по-новому, розглядаючи кожного разу трасований промінь окремо.

Якщо через кожен піксель області огляду проводити по одному променю, то гладкі лінії, в тривимірному просторі, після побудови проекції стануть східчастими на екрані. Щоб уникнути цього, необхідно генерувати декілька променів на кожен піксель, рахувати для кожного колір і знаходити його середнє значення.

До переваг методу трасування променів можна віднести можливість рендерингу гладких об’єктів без апроксимації їх полігональними поверхнями.

3.4. ВИСНОВКИ ДО РОЗДІЛУ

У комп’ютерній графіці концепція зйомки променів або світла, або від очей називається трасуванням траси. Термін трасування променів також може бути використаний, але концепція трасування траси дозволяє припустити, що цей спосіб створення зображень, що генеруються комп’ютером, спирається на шлях від світла до камери, або навпаки. Роблячи це фізично реалістичним чином, ми можемо легко моделювати оптичні ефекти, такі як каустика або відображення світла іншою поверхнею в сцені. Підводячи підсумок, важливо пам’ятати, що процедуру рендерингу можна розглядати як два окремі процеси. Перший крок визначає, чи видно точку на певному пікселі, а другий – тінь. Алгоритм елегантний і потужний, але змушує нас торгувати часом візуалізації для точності і навпаки.

4. ТЕСТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ

4.1. МЕТОДИ ОПТИМІЗАЦІЇ СИСТЕМИ

Як було проаналізовано, метод трасування променів – це повністю функціональний алгоритм, але потребує значних обчислювальних витрат. Тому були реалізовані деякі ідеї для прискорення роботи трасування.

Найбільш очевидний спосіб пришвидшення роботи трасування променів полягає в тому, щоб трасувати декілька променів одночасно. Оскільки кожен промінь, який виходить з камери, незалежний від усіх інших, та більшість структур потрібні лише для перегляду, можна трасувати по одному променю на кожне ядро центрального процесора без яких-небудь складностей через проблем з синхронізацією.

Паралелізація. Насправді, алгоритм трасування променів вважається надзвичайно паралельним, тому що саме їх суть дозволяє дуже просто їх розпаралелювати.

Кешування даних. Зазвичай, трасування променів більше всього часу витрачає на обрахунок значень для променів сфери, але деякі значення розташування сфери, після їх виявлення, залишаються незмінними. Можна їх обрахувати один раз під час завантаження сцени та зберігати їх в самих сферах.

Оптимізація затінення. Якщо точка об’єкту в тіні знаходиться відносно джерела світла, тому що на її шляху знаходиться інший об’єкт, то ми отримуємо велику ймовірність того, що сусідня з нею точка, через той самий об’єкт, також знаходиться в тіні відносно джерела світла (Рис 4.1). [15]

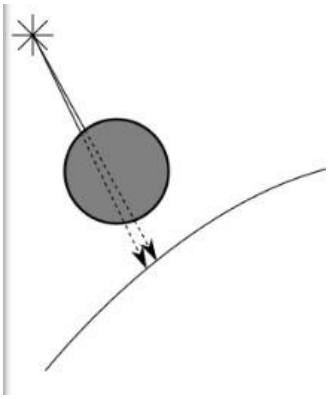

Рис. 4.1 – дві затіненні точки відносно одного джерела світла та об’єкту затінення [15]

Тобто коли ми шукаємо об’єкти між джерелом світла та заданою точкою, ми повинні спочатку перевірити, чи не накладається на поточну точку тінь від найближчого до неї об’єкту, що наклав тінь на попередню точку, відносно того ж самого джерела світла. У випадку, коли це так, ми можемо завершити, а якщо ні, треба продовжити звичайним методом перевіряти інші об’єкти.

Аналогічно, при обчисленнях перетину між променем світла та об’єктами нам не потрібно знати найближчий перетин, достатньо лише знати, що існує хоча б один перетин. Отже, використовується спеціальна версія алгоритму з урахуванням методу Монте-Карло, яка повертає результат, коли знаходиться перший перетин, і для цього треба повертати лише булеве значення.

Просторові структури. Обрахування перетину кожного променя з кожною сферою досить велика втрата часу. Існує велика кількість структур, яка дозволяє легко відкидати цілі групи об’єктів, що не потребують обчислень перетинів.

Уявімо, що є декілька сфер, які знаходяться досить близько один до одного. Можна обрахувати центр та радіус найменшої сфери, яка містить в собі всі ці сфери. Якщо промінь не перетинає цю граничну сферу, то можна бути впевненим, що він не перетинає сфери, які містяться в головній. Це можна зробити за допомогою одної перевірки перетину, але якщо промінь перетинає сферу, то все одно треба перевіряти наступні сфери на перетин з ним.

Субдискретизація. Існує доволі простий спосіб зробити трасування променів в N разів швидше, тобто обраховувати в N разів менше пікселів.

Уявімо, що трасуються промені для пікселів (10, 100) та (12, 100), та вони падають на один об’єкт. Можна логічно припустити, що промінь для пікселя (11, 100) також буде припадати на той же об’єкт, пропустити початковий пошук перетинів з усією сценою, та перейти до обрахування кольору в цій точці. [12]

Якщо зробити так в горизонтальному та вертикальному напрямках, то можна виконувати максимум на 75% менше початкових обрахувань перетинів

променів зі сценою.

Але в даному методі оптимізації існує один суттєвий недолік. Можна дуже просто пропустити дуже тонкий об’єкт, та результати використання оптимізації не будуть схожі на ті, що отримуються без неї. Головна ідея полягає в тому, щоб здогадатися, коли слід використовувати цей метод правильно, для отримання задовільних результатів.

Суперсемпінг. Суперсемпінг – це протилежність методу субдискретизації, якщо ми віддаємо перевагу точності замість шкидкості. Уявімо, що промені, які відповідають двом сусіднім пікселям, припадають на два різних об’єкта. Нам потрібно розмалювати кожен піксель у відповідний колір.

Кожен промінь повинен задавати колір для кожного квадрату сітки, через яку ми дивимося. Використовуючи по одному променю на піксель, ми умовно вирішуємо, що колір променя світла, який проходить через середину квадрату, визначає цілий квадрат, але це не завжди так.

Для вирішення цієї проблеми можна трасувати декілька променів на піксель – 5, 10, 17, і так далі, а потім брати середнє значення, для отримання кольору пікселя. Це призводить до того, що трасування променів становиться повільніше в 5, 10 або 17 разів, завдяки тій же самій причині, по якій субдискретизація робить його в N разів швидше. Але існує компроміс. Можна припустити, що властивості об’єкта на протязі його поверхні змінюються плавно, тобто трасування 5 променів на піксель, які припадають на один об’єкт в сусідніх точках, не дуже сильно поліпшить вид сцени. Тому можна почати з одного променя на піксель та порівнювати сусідні промені: якщо вони припадають на інші об’єкти, або їх колір відрізняється більше ніж на порогове значення, то застосовуємо до них підрозділ пікселів. [13]

4.2. Методи тестування

Для базового тестування системи побудови зображень 3D моделей з використанням технології відстеження зворотної траєкторії променя було обрано 3 початкових сцени. Всі тестування розробленої системи відбувались на

комп’ютері за наступними технічними характеристиками:

- Процесор Intel Core i5-6198DU, 2 ядра, 2.4 GHz;
- Оперативна пам’ять DDR 3, 8 gb;
- Відеокарта GeForce 960;

У результаті досліджень було розроблено програмний модуль мовою програмування cPython з інтеграцією у веб-оболонку з застосуванням WEBJL та базового WEB UI KIT для наочного відображення роботи удосконаленої версії алгоритму.

4.3. РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Для тестування результатів роботи розробленої системи побудови зображення 3D моделей з відстеженням зворотної траєкторії променя були побудовані зображення з наступними властивостями:

- Базова растеризація;
- Перетворення об’єктів за допомогою матриць (переміщення, обертання камери та джерела світла);
- Затінення;
- Білінійна інтерполяція;
- Відображення променів;
- Навколишнє освітлення сцени
- Відображення;
- Заломлення світла;

Отже, наступні приклади демонструють функціональну роботу системи:

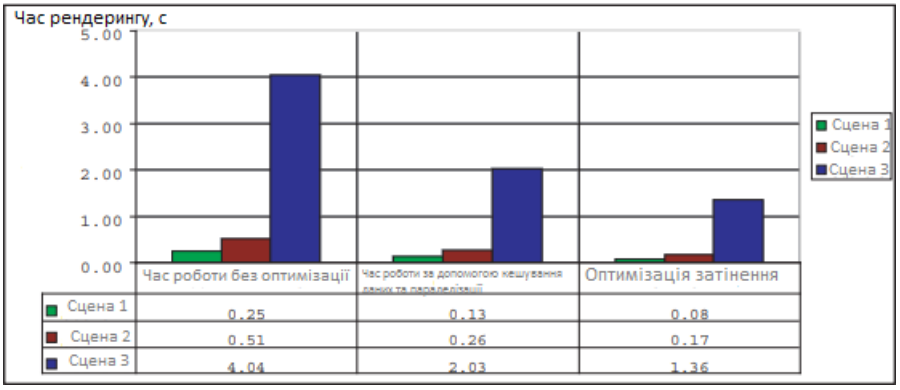

Рис. 4.2 – час рендерингу для різних методів оптимізації

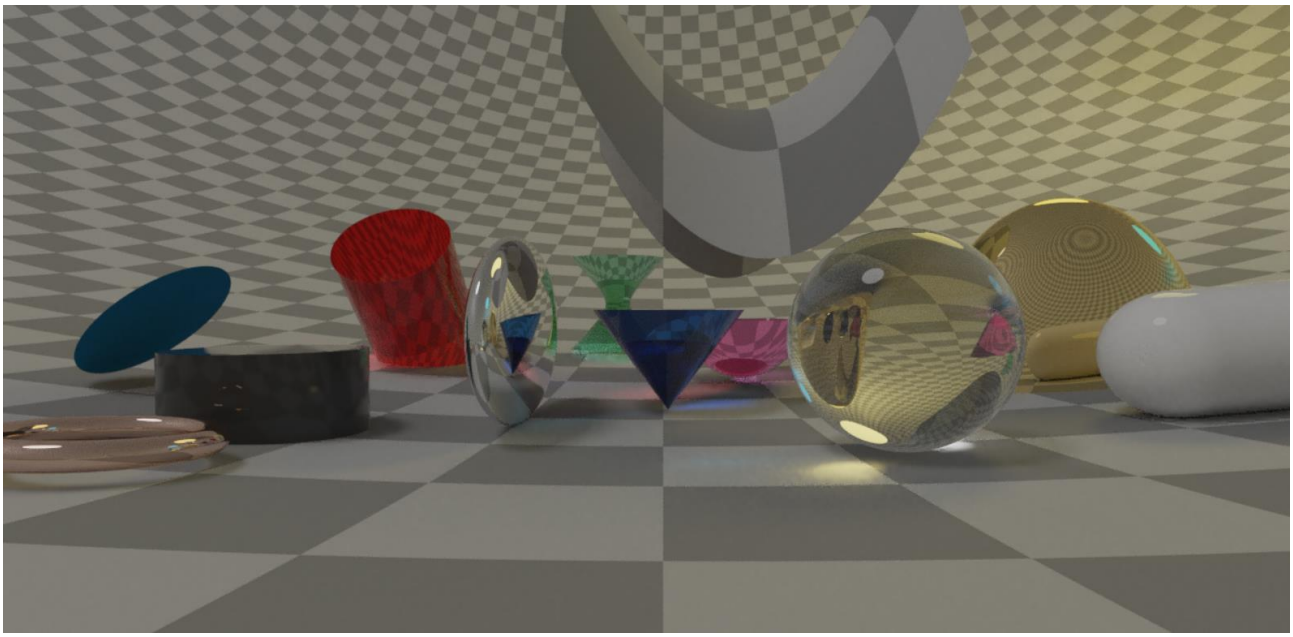

Рис. 4.3 – сцена 1 з застосуванням технології зворотного відстеження променя

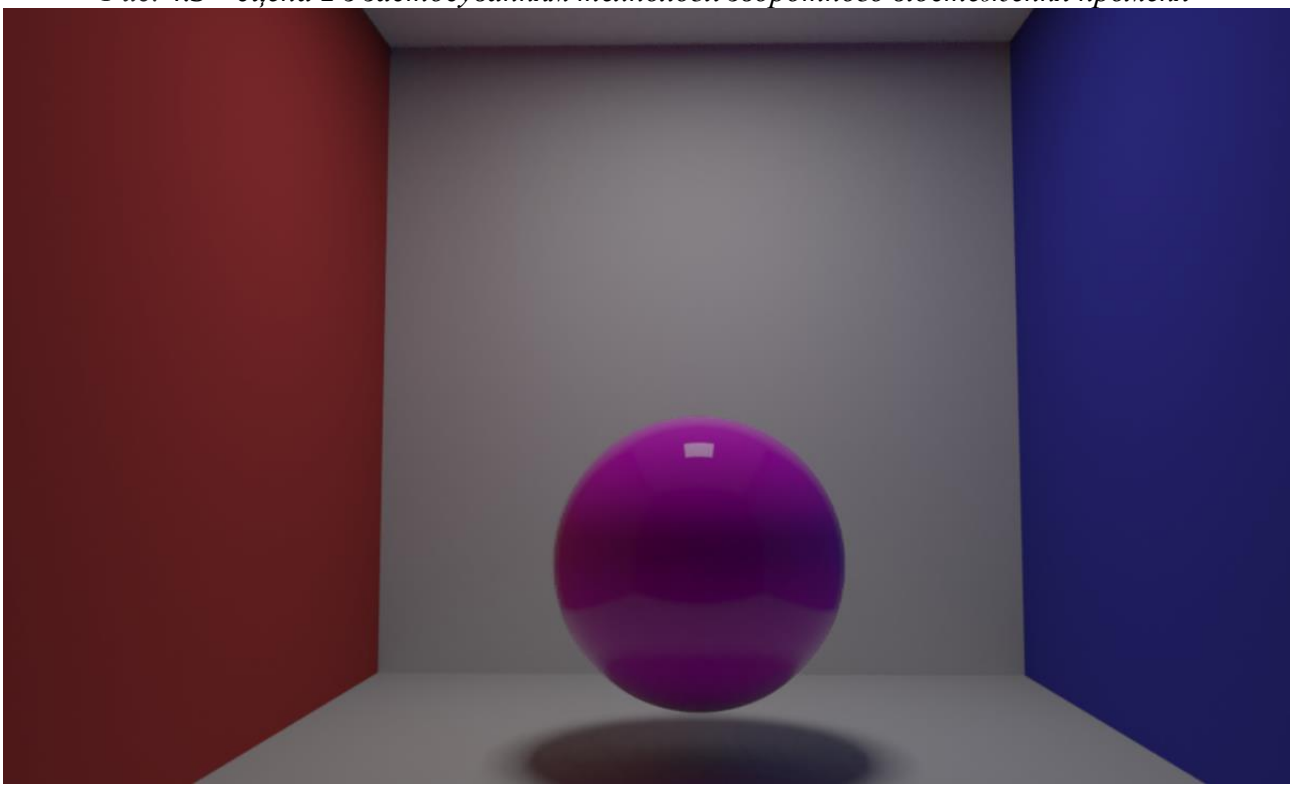

Рис. 4.4 – сцена 2 з застосуванням технології зворотного відстеження променя

Рис. 4.5 - сцена 3 з застосуванням технології зворотного відстеження променя

Кількість субпікселів на піксель	Час генерації кадру зображення, мс
0	594
4	2501
16	11448
25	20444

Табл. 4.1. Залежність часу генерації кадру растрового зображення від кількості надлишкових дискретних вибірок на один піксель

4.4. ВИСНОВКИ ДО РОЗДІЛУ

В даному розділі були обґрунтовані методи оптимізації системи, такі як: кешування даних, розпаралелювання, оптимізація затінення,

субдискретизація та суперсемпінг. Продемонстровано на малюнку функціонування методів оптимізації та доведена їх коректність. Крім цього, візуально показана робото-спроможність та фото-реалістичність системи побудови зображення 3D моделей з відстеженням зворотної траєкторії променя на багатьох прикладах. Також було протестована система на слабших комп’ютерах, і виявлено, що вона може не працювати коректо, у зв’язку з тим, що система потребує доволі потужну обчислювальну спроможність.

ВИСНОВКИ

В ході першого етапу досліджень я зробила висновок, що розрахунок освітлення є алгоритмічно складною задачею. Навіть з урахуванням усіх спрощень та компромісів, що допускаються у ряді методів моделювання освітлення, результуюча складність усе одно може бути завеликою і затрати часу на візуалізацію окремого кадру можуть перевищувати визначений максимальний ліміт.

Відповідно до цього було досліджено існуючі аналоги математичних моделей для побудови різних видів сцен та розроблено певні додаткові оптимізації щодо них для можливості зменшення використання часу на процес візуалізації, а також спроби використання більш складних та якісних методів моделювання освітлення, щоб, за рахунок цього, за загальним часом візуалізації залишитись у встановлених вимогами межах та зі збереженням якості сцен.

В ході дослідження та опрацювання даної теми мною було створено програмний модуль з використанням алгоритму на основі методу трасування променів для збереження якості сцен з урахуванням заломлення променів світла відповідно до заданих матеріалів при зменшенні часу на обробку та формування даної сцени, а також ваги вихідного файлу та мови програмування Python завдяки гнучкості та простоті синтаксису даної мови.

Саме метод трасування променів увійшов в основу за рахунок можливості рендерингу гладеньких об’єктів без апроксимації їх полігональними поверхнями, відсічення невидимих поверхонь, перспективи і коректних змін поля зору, які є логічним наслідком алгоритму та низки переваг над іншими методами, які були розглянуті в ході досліджень, за рахунок яких було отримано зменшення часу на генерацію одного кадру результуючої сцени з мінімальними втратами по якості для користувача.

На даний момент основним призначенням даного модулю є його використання, як бібліотеки для використання різними програмами, які створені як системи віртуальної реальності, так і для професійних цілей,

зокрема перетворювачів для зменшення розмірів вхідних файлів для подальшого опрацювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тернер Уіттіс. «Покращена модель освітлення для затіненого дисплея», 1980 [Text].
2. Роберт Кук, Томас Портер, Лорен Карпентер. «Розподілене трасування променів», 1984 [Text].
3. J. Avro, and D. Kirk. A survey of ray tracing acceleration techniques. In An Introduction to Ray Tracing, A. Glassner, Ed., pages 201–262. Academic Press, San Diego, CA, 1989 [Text].
4. Артур Аппель. «Деякі методики відтінку твердих тіл», 1969 [Text].
5. Кевін Сафферн. «Ray Tracing from the Ground Up», 2007 [Text].
6. HEKPIKS.ORG [Electronic resource] https://helpiks.org/3-2158.html - Last access: 2019.
7. NVidia Corporation, NVIDIA GPU Programming Guide Version 2.2.0, 2004 [Text].
8. John Amanatides, Andrew Woo. A Fast Voxel Traversal Algorithm for Ray Tracing, 1987 [Text].
9. Michael McCool. http://libsh.sourceforge.net/, 2004 [Electronic resource].
10. Интерактивная трассировка лучей с использованием SIMD инструкций INTEL [Electronic resource]. / Intel, 2009. - Режим доступа: http://software.intel.com/ru-ru/articles/ interactive-ray-tracing.
11. Yunfan Zhang. FPGA Ray Tracer: [Electronic resource]. – Режим доступа: http://www.eeweb. com/project/yunfan_zhang/fpga-ray-tracer.
12. Юніс Мохаммад. Залучення блочної міжпіксельної інтерполяціх для прискорення алгоритму трасування променів / Мохаммад Юніс, Р.В. Мальчева, С.О. Ковалев // Машинобудування та техносфера XXI століття. / Сбірник XVIII міжнародної научної конференції. - Донецьк: ДонНТУ, 2011.

13.- Т.4. - С.189-191 [Text].

14. Лисиця В.Т. Колірні моделі та закони поширення світла / В. Т. Лисиця. – Х. : ХНУ імені В.Н. Каразіна, 2012. – 82 с

15. Шикін Є. В. Комп'ютерна графіка. Полігональні моделі (рос.) / Є. В. Шикін, О. В. Борєсков. – М. : Діалог МІФІ,2001. – 462 с

16. Han-Wei Shen. Ray tracing basics [Електронний ресурс]. – Режим доступу: http://web.cse.ohiostate.edu/~hwshen/681/Site/Slidesfiles/basicalgo.pdf (дата звернення: 19.10.2021).

17. Roth S.D. Ray casting for modeling solids // Computer Graphics and Image Processing. -1982. - № 18. - P. 109-144.

18. Appel A. Some techniques for shading machine renderings of solids // AFIPS spring joint computer conference. IBM Research Center, Yorktown Heights, N.Y. - 1968. - P. 37-45.

19. Han-Wei Shen. Ray tracing basics [Електронний ресурс]. - Режим доступу: http:// web.cse.ohiostate.edu/ hwshen/681/Site/Slidesfiles/basicalgo.pdf (дата звернення: 07.09.2021).

20. Foley J.D., van Dam A. Feiner S.K., Hughes J.F. Computer Graphics: Principles and Practice // Addison-Wesley Publishing Company. - 1996. - P. 718-739.

21. Phong B.T. Illumination for computer generated pictures// Communications of ACM. - 1975. – № 6. – P. 311-317.

22. Foley J.D., van Dam A. Feiner S.K., Hughes J.F. Computer Graphics: Principles and Practice // Addison-Wesley Publishing Company. – 1996. – P. 718–739.

23. Phong B.T. Illumination for computer generated pictures// Communications of ACM. 1975. – № 6. – P. 311–317.