Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра інформатики

ЗАСТОСУВАННЯ ТЕХНОЛОГІЇ ТРАСУВАННЯ ПРОМЕНІВ ДЛЯ 3D ЗОБРАЖЕНЬ

Текстова частина до курсової роботи

за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник курсової роботи

Бучко О. А.
(*прізвище та ініціали*)

_____

(*підпис*)

“____”_____ 2023 р.

Виконав студент

Нечипорчук Д.Є.
(*прізвище та ініціали*)

_____

(*підпис*)

“____”_____ 2023 р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
доцент, к.ф.-м.н.
_____ С. С. Гороховський
(підпис)
“____” _____ 20__ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Нечипорчуку Данилу Євгенійовичу

факультету інформатики 3 курсу

ТЕМА: Застосування технології трасування променів для 3D зображень

Вихідні дані:

Зміст ТЧ до курсової роботи:

Анотація

Вступ

1. Загальний опис технології

2. Характеристика об'єктів

3. Деталі практичної реалізації

Висновки

Список джерел

Дата видачі “___” _____ 20__ р.

Керівник _____ Завдання отримано _____

Тема: Застосування технології трасування променів для 3D зображень

Календарний план виконання роботи:

№	Назва етапу курсової роботи	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу	листопад	
2.	Огляд літератури за темою роботи	листопад-грудень	
3.	Створення практичної частини роботи	січень-березень	
4.	Написання текстової частини роботи	березень-квітень	
6.	Надання роботи керівнику для перевірки	кінець квітня	
8.	Коригування роботи за результатами перевірки	кінець квітня-початок травня	
9	Подання роботи на кафедру для перевірки на плагіат	тиждень до захисту	
10.	Захист курсової роботи	кінець травня	

Студент Нечипорчук Д. Є.

Керівник Бучко О. А.

“____” _____

Зміст

Анотація 5
Вступ 6
Розділ 1. Загальний опис технології 7
1.1 Основні поняття 7
1.2 Рівняння променя та сфери 8
1.3 Камера 10
1.4 Визначення кольору 10
1.5 Згладжування 11
Розділ 2. Характеристика об'єктів 12
2.1 Матеріали та текстури 12
2.2 Матові об'єкти 12
2.3 Металеві об'єкти 13
2.4 Об'єкти, які заломлюють промені 14
2.5 Світлові об’єкти 15
2.6 Текстури 16
Розділ 3. Деталі практичної реалізації 18
3.1 Камера 18
3.2 Сцена 18
3.3 Інформація про перетин променем об'єкта 19
3.4 Матеріали та текстури 19
3.5 Згладжування 20
3.6 Бібліотеки для створення графічного інтерфейсу 20
3.7 Виведення результату 20
3.8 Оптимізація 21
Висновки 22
Список джерел 23

Анотація

У роботі розглянуто технологію трасування променів для створення фотореалістичних 3D зображень. Також описуються основні підходи, які дозволяють створити подібні зображення з використанням сфер, що складаються з різних матеріалів та текстур. Наостанок розглянуто спосіб створення графічного інтерфейсу у вигляді вікна для виведення кінцевого зображення.

Вступ

Зі збільшенням обчислювальних потужностей комп’ютерів, тема створення реалістичних зображень стає дедалі актуальнішою. З кожним роком якість зображень, що створені за допомогою комп’ютерних систем, у відеоіграх чи кіно покращується, і технологія трасування променів є однією з причин цих змін.

Але створення таких зображень – це задача, яка може бути реалізована різними способами. За допомогою деяких з них, як наприклад використання систем DirectX, OpenGL чи Vulkan, можна дуже швидко малювати складні 3D зображення. Хоча з метою створення справді фотореалістичних зображень можна використати підхід трасування променів. В його основі лежить ідея симулювання того, як поводять себе світлові промені у нашому житті. Такий підхід матиме свої недоліки, які будуть стосуватися того, що створення подібних зображень є доволі складним процесом, який вимагає значної кількості ресурсів комп’ютера. Незважаючи на це, подібну систему можна створити, а фінальні зображення будуть в рази кращими за типові приклади зображень, що створені з використанням інших підходів.

Метою роботи є розробка застосунку, який зможе створювати 3D зображення за допомогою технології трасування променів, з подальшим виведенням цих зображень на екран.

Завдання ж роботи будуть:

1. Дослідити основи технології трасування променів.
2. Впровадити цю технологію у вигляді застосунку, який зможе створювати різноманітні 3D зображення.
3. Розробити користувацький інтерфейс, який зможе вивести результат роботи застосунку на екран.

Розділ 1. Загальний опис технології

1.1 Основні поняття

У словосполученні "трасування променів" є 2 слова, які варто розібрати. Почну зі слова промінь, який є доволі простою структурою. За визначенням - це пряма, яка має початок, але не має кінця. У нашому випадку основною характеристикою променя є 2 значення - початок променя і його напрямок. Для роботи з променями досить зручно використати поняття вектору. Можна сказати, що початок і напрямок променя - це 2 вектори, кожен з яких має 3 координати(x, y та z), оскільки ми працюємо у тривимірному просторі. Вектори дуже часто використовуються для вирішення задач з комп'ютерної графіки, оскільки для них існують зручні функції, на зразок скалярного та векторного добутку.

Слово трасування описує те, що ми будемо відслідковувати рух цих променів. Загальна ідея технології полягає у тому, що маючи певну сцену з різними об'єктами у ній, ми би хотіли намалювати її зображення на екрані. Для вирішення цієї задачі ми візьмемо точку, з якої ми дивимося на цю сцену, і запустимо у світ багато променів, для яких ця точка буде початком. Напрямок променя буде залежати від того, яку частину зображення ми малюємо. Загалом ми маємо запустити окремий промінь для кожного пікселя на кінцевому зображенні. Маючи зображення 100 на 100 пікселів, ми маємо запустити 10000 променів. Кажучи запустити я маю на увазі те, що нам потрібно створити цей промінь, а надалі відслідкувати його рух. Цей процес матиме на меті задачу перевірити, чи не перетне промінь якийсь об'єкт у нашій сцені. Якщо промінь натрапить на якийсь об'єкт, то ми маємо замалювати відповідний піксель зображення кольором, який буде залежати від самого об'єкта.

Для отримання справді реалістичних зображень ми хотіли би симулювати реальну поведінку світлових променів. Це означає, що при зіткненні з об'єктом промінь може відбитися і попрямувати у іншу сторону,

поглинутися об'єктом, або ж піддатися заломленню(наприклад при переході через скло). Об'єкти які зустріне відбитий або заломлений промінь на своєму шляху вплинуть на кінцевий колір пікселя. Остаточний колір об'єкта, з яким зіткнувся промінь, буде залежати від текстури об'єкта, а відбитий чи заломлений промінь буде визначати матеріал об'єкта.

Даний підхід розглядає ситуацію, коли ми маємо певну сцену, запускаємо у неї багато променів і рахуємо для кожного променя колір, який ми маємо повернути для відповідного пікселя. Такі обрахунки можуть вимагати від системи значних ресурсів та займати багато часу, але у даній роботі я не буду концентруватися на підходах, які ставлять на меті використання цієї технології для створення зображень у реальному часі.

1.2 Рівняння променя та сфери

Об'єкти у сцені можуть бути абсолютно різними, тому нам необхідно обрати якісь певні фігури, з яких будуть складатися всі ці об'єкти. Загальноприйнятою є ідея використання трикутників, як основи всіх об'єктів. У своїй роботі я обрав описати використання кулі як об'єкту, який ми зможемо намалювати у сцені. Підхід з трикутниками буде дуже схожим, але рівняння сфери треба буде замінити на рівняння площини. Натомість з використанням сфер можна досить просто створити наповнені й красиві зображення та наочно продемонструвати підходи технології трасування променів.

Отже, щоб намалювати кулю, ми маємо перевірити чи перетинає її наш промінь, запущений у світ з точки спостереження. Для цього нам потрібно записати рівняння променя, рівняння сфери і знайти точки перетину променя і сфери, якщо вони існують. Рівняння променя для кожної координати виглядатиме так:

$$a_x + tb_x = x$$
$$a_y + tb_y = y$$
$$a_z + tb_z = z$$

(1.1)

Де a_x, a_y, a_z – координати вектора початку променя, b_x, b_y, b_z – координати вектора напрямку променя, а x, y, z – координати точки, яка належить променю. Рівняння кулі для тривимірного простору виглядатиме наступним чином:

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 - r^2 = 0$$

(1.2)

Де x, y, z – координати точки, яка належить сфері, c_x, c_y, c_z – координати вектора центру сфери, а r – радіус сфери. Далі нам потрібно підставити відповідні значення з рівняння променя у рівняння сфери, щоб знайти точку перетину. Після розкриття дужок і зведення відповідних доданків ми отримаємо таке рівняння:

$$\left(b_x^2 + b_y^2 + b_z^2\right)t^2 + 2\left(b_x(a_x - c_x) + b_y\left(a_y - c_y\right) + b_z(a_z - c_z)\right)t +$$
$$+ (a_x - c_x)^2 + (a_y - c_y)^2 + (a_z - c_z)^2 - r^2 = 0$$

(1.3)

Нам потрібно знайти t, адже тоді ми можемо підставити це значення у рівняння променя та знайти точку перетину. Це рівняння є звичайним квадратним рівнянням, для якого нам потрібно знайти дискримінант. Дискримінант укаже чи існує дві, одна або жодної точки перетину променя і сфери. Промінь, у випадку з двома точками, перетне сферу вперше і

опиниться у ній, а потім перетне її вдруге і вийде з неї. В такому разі з двох точок нам буде доцільно розглядати найближчу. Найближчою точкою буде та, для якої значення t буде меншим.

Для визначення t потрібно знайти значення a, b та c для квадратного рівняння вигляду $at^2 + bt + c = 0$. Якщо поглянути на значення a, то його можна записати як скалярний добуток вектора напрямку променя на самого себе. Значення b можна визначити як добуток двійки та скалярного добутку вектора напрямку променя на вектор різниці початку променя і центру сфери. У свою чергу значення c можна обрахувати як різницю квадрата радіуса сфери та скалярного добутку вектора різниці початку променя і центру сфери на самого себе. Маючи ці значення, залишиться обрахувати дискримінант та знайти t.

1.3 Камера

Для того, щоб реалізувати цей алгоритм, потрібно описати те, як можна створити всі промені для запуску їх у світ. Шляхом використання двох циклів можна перебрати всі індекси x та y для визначення пікселя на вихідному зображенні. Маючи індекси можна отримати відповідний промінь для пікселя. Доцільним рішенням буде визначити клас камери, який зможе надати такий промінь.

1.4 Визначення кольору

Після отримання променя його шлях потрібно відслідкувати і перевірити чи зіткнувся він з якимось об'єктом. Наступним кроком буде знаходження нового напрямку для променя після відбиття або заломлення. Одразу за цим слідуватиме обрахунок кольору об'єкта, з яким зіткнувся промінь, щоб врахувати його для визначення кінцевого кольору пікселя. Щойно колір буде визначено, можна знову почати процес перевірки на

перетин променем об'єкта, при тому що промінь змінив напрям руху і мав би перетнути інший об'єкт. Опрацювавши заздалегідь визначену кількість відбивань чи заломлень, можна перемножити кольори усіх об'єктів для знаходження кінцевого кольору. Кольори будуть визначені у вигляді значень від 0 до 1 для кожного з трьох спектрів: червоного, зеленого та синього.

1.5 Згладжування

Останнім кроком перед виведенням значення пікселя буде процес згладжування. Зображення без згладжування буде мати занадто різкі межі об'єктів та загалом негативно впливати на якість. Хоча питання згладжування та його технік може бути дискусійним, у своїй роботі я буду його використовувати. Ідея яку я обрав полягала у використанні декількох променів для визначення кольору одного пікселя. Якщо відслідкувати декілька променів, отримати окремий колір від кожного з них та знайти середнє значення цих кольорів, то результатом буде зображення зі згладжуванням.

Розділ 2. Характеристика об'єктів

2.1 Матеріали та текстури

Для того, щоб створити фотореалістичні зображення варто описати різні характеристики об'єктів. У реальному світі дерев'яний стіл і сталева куля дуже по-різному відбивають світлові промені. Щоб симулювати ці відмінності у комп’ютерній графіці, я використаю поняття матеріалу та текстури, які матиме кожен об'єкт. Матеріал буде визначати те, як поводить себе промінь після зіткнення з об'єктом. Текстура ж відповідатиме за визначення кольору об’єкта.

Як було сказано у минулому розділі, промені після зіткнення з об'єктом можуть відбиватися, поглинатися чи заломлюватися. Але трасування шляху променя має у певний момент зупинитися, щоб обрахувати кінцевий колір для пікселя. У відкритій сцені промінь може і зовсім не перетнути жодного об'єкту, що буде логічним закінченням процесу трасування для нього. Але якщо ми матимемо сцену у вигляді кімнати з чотирма стінами, підлогою та стелею, то промені у ній можуть відбиватися нескінченно. Щоб зупинити процес трасування у такій ситуації можна обмежити максимальну кількість відбивань або заломлень променя. Коли ця кількість перевищить встановлений поріг, то ми можемо спокійно порахувати фінальний колір, з урахуванням усіх об'єктів, які промінь зустрів на своєму шляху.

2.2 Матові об'єкти

Більшість предметів у реальному світі не відбивають світлові промені рівномірно, тобто таким чином, щоб кут падіння дорівнював куту відбиття. Натомість відбитий промінь може попрямувати у будь-якому напрямку, і така ситуація зумовлена тим, що поверхня подібних об'єктів досить нерівна.

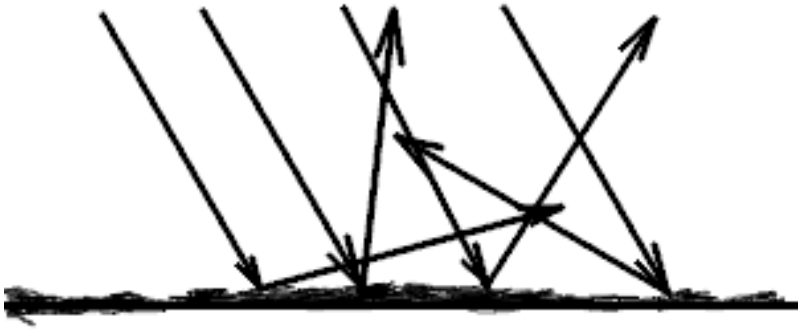

Рис 1. Відбиття променів від матового об’єкту

Щоб симулювати побідну поведінку променів я використав випадкове визначення напрямку відбитого променя. Ідея полягає у тому, щоб спочатку уявити сферу, що знаходиться у точці перетину променя та об'єкта і має діаметр довжиною один. Напрямок променя після відбивання має бути направлений з точки перетину з об'єктом до випадкової точки всередині цієї сфери.

2.3 Металеві об'єкти

Протилежним до матових об'єктів є металеві. Зіткнення з цими об'єктами змушує промінь відбитися ідеально, так щоб кут падіння дорівнював куту відбивання. Фактично, щоб реалізувати цей матеріал потрібно знайти промінь після відбиття.

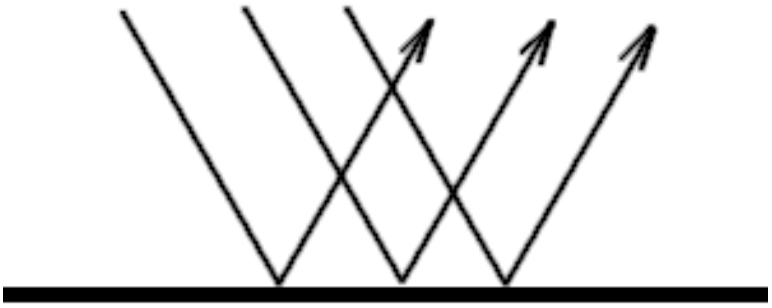

Рис 2. Відбиття променів від металевого об’єкту

Металеві об'єкти поводять себе як дзеркала. Це означає, що у металевих об'єктах ми зможемо побачити відображення об'єктів навколо.

Серед цікавих доповнень варто зазначити, що перетворення дзеркальної металевої поверхні на зовсім нечітку є доволі простим завданням. Можна уявити, що нечітка поверхня буде результатом знову таки нерівностей об'єкту, внаслідок яких промінь буде відхилятися у випадкову сторону після відбиття. Якщо трохи змінити промінь після ідеального відбиття, то можна досить просто досягти необхідного результату. У прикладі я використовую параметр, який визначається при створенні матеріалу, для того, щоб змінити промінь після відбиття, шляхом множення кожної координати цього променя на параметр.

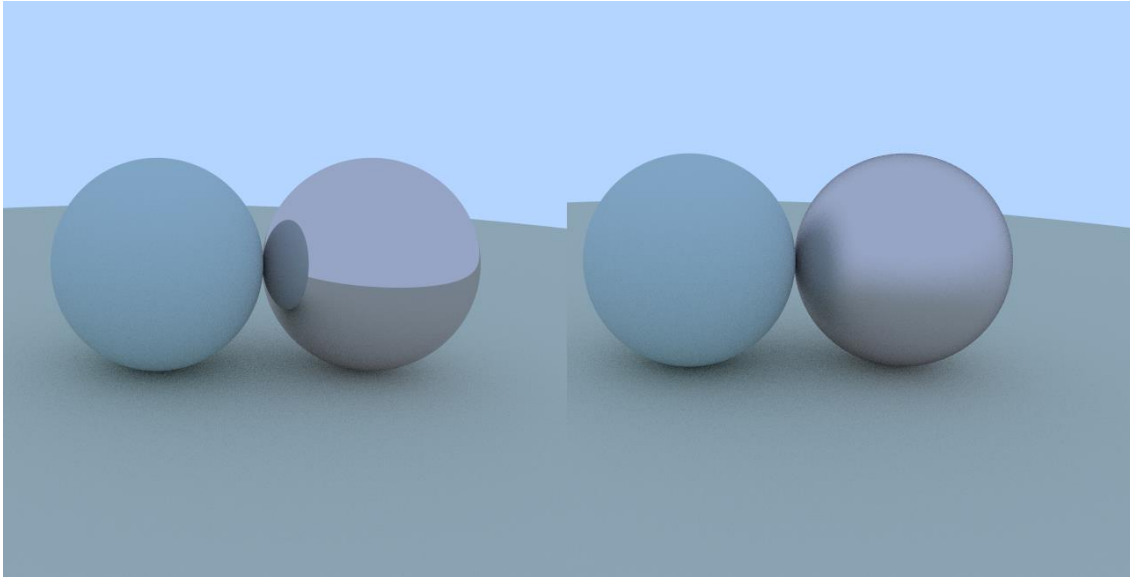

Рис 3. Різниця між дзеркальним та нечітким металевим об’єктом

2.4 Об'єкти, які заломлюють промені

Процес заломлення трохи відрізняється від типового відбиття. Насамперед для реалізації цього матеріалу необхідно знайти промінь після його заломлення. Оскільки кут заломлення буде залежати від індексу заломлення, ми маємо визначати це значення при створенні кожного об’єкту з цим матеріалу. Інша відмінність полягатиме у тому, що при перетині об'єкту, промінь може як заломитися, так і відбитися.

Якщо індекс заломлення для матеріалу потрібно вказувати при його створенні, то індекс заломлення для навколишнього середовища дорівнює 1, адже промені проходять через повітря.

Одразу складається враження, що об'єкт мав би виглядати як типове скло, у якому ми бачимо усі об'єкти позаду нього. Насправді ж об'єкт буде виглядати прозорим, але з перевернутим зображенням. Загалом такий ефект є логічним, якщо самостійно обрахувати як поведуть себе промені після проходженні через сферу.

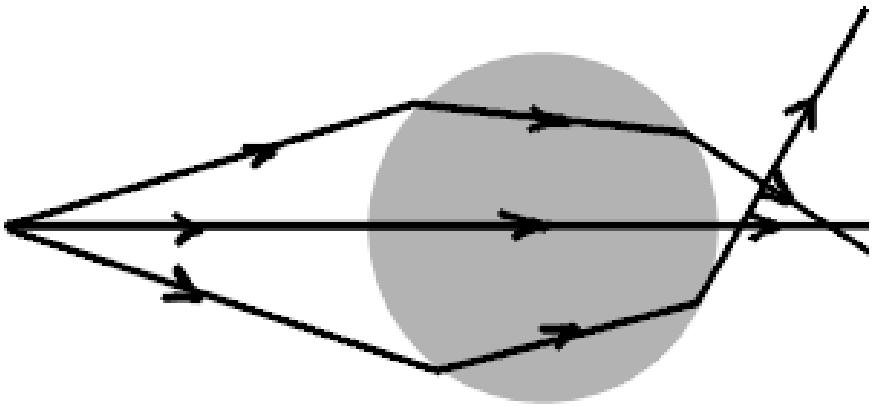

Рис 4. Схематичне зображення руху заломлених променів через сферу

2.5 Світлові об’єкти

Для того, щоб створити світло, можна використати світлові об'єкти. Якщо визначати світло, як матеріал для об'єкту, то ми зможемо створити у сцені світло у дуже простий спосіб. Щоб такий об'єкт впливав на інші об'єкти, він має додавати досить значну яскравість до них. Щоб це реалізувати, можна визначити окреме значення при трасуванні шляху кожного променя, яке буде дорівнювати яскравості світлового об'єкта, якщо промінь натрапить на нього, і нуль у іншому випадку. Щоб це значення справді вплинуло на кінцевий результат, варто додати його, на відміну від

кольорів інших об'єктів на шляху променя, які множаться між собою при визначенні кінцевого кольору пікселя. Промінь не буде відбиватися від світлового об'єкта, а просто зупинить процес відслідковування шляху.

2.6 Текстури

Перший варіант текстури полягає у суцільному кольорі. При перетині об'єкту ми можемо просто взяти його колір, який був визначений при створенні.

Справжня мета текстур - це описати механізм використання якогось зображення для визначення кольору об'єкта. На прикладі сфери можна використати зображення планети Юпітера. Якщо завдання полягає у створенні планети Юпітер у сцені, то необхідно використати розгорнуте зображення планети у прямокутному вигляді.

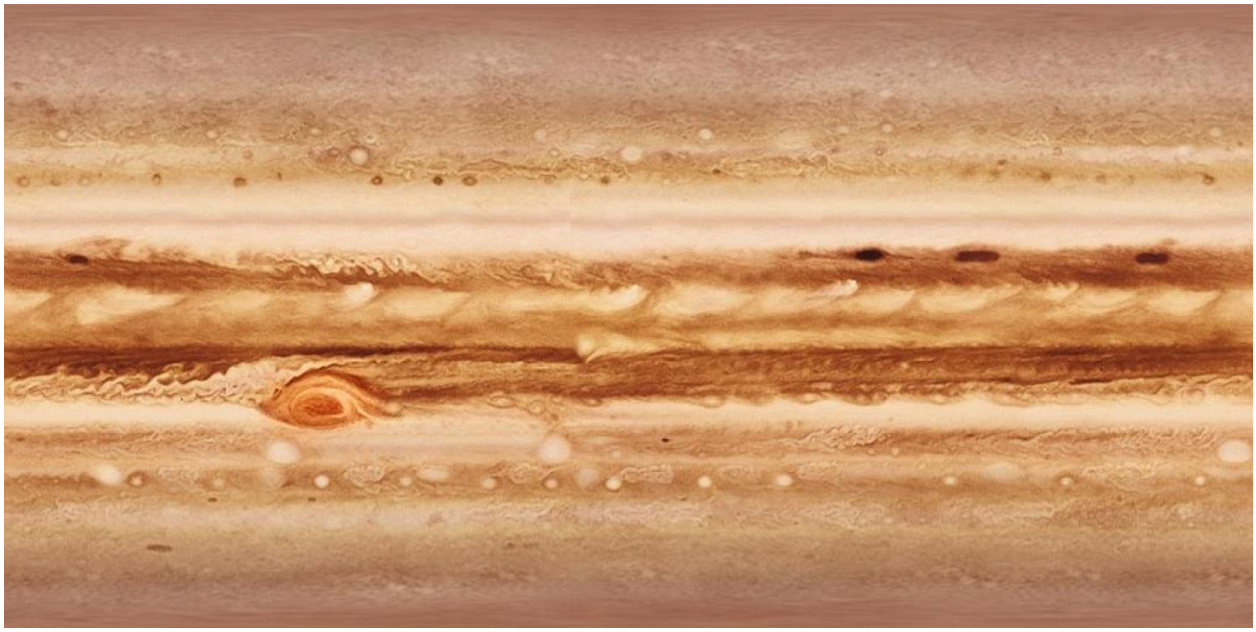

Рис 5. Приклад зображення для створення сфери з текстурою планети Юпітер

Щоб досягти бажаного результату потрібно створити механізм визначення відповідності між пікселем на зображенні текстури і координатою сфери. Ідея полягатиме у тому, щоб визначити координати u та v для сфери,

які будуть набувати значення від 0 до 1 включно. Для координати u ми матимемо усі значення при русі навколо осі Y, зі зміною координат x та z. Для координати v ми матимемо усі значення при русі навколо осі X, зі зміною координати y.

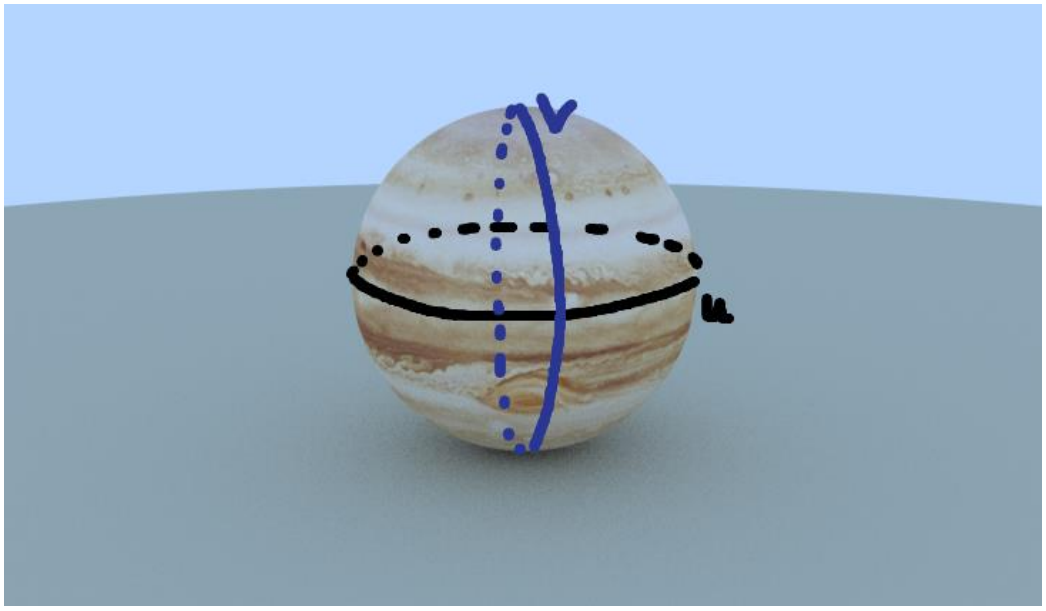

Рис 6. Схематичне зображення різниці між u та v координатами, де усі значення u виділені чорною лінією, а v – синьою

Розділ 3. Деталі практичної реалізації

Застосунок було створено під операційну систему Windows, із використанням мови програмування C++.

3.1 Камера

Для того, щоб створити зображення за допомогою технології трасування променів варто визначити окремий промінь для кожного пікселя зображення. Якщо визначити координати для пікселя у подвійному циклі, то залишиться отримати відповідний промінь за ними. Це завдання має виконати об'єкт класу камери за допомогою відповідного методу, що прийме значення x та y як параметри.

Камера буде створена за допомогою визначення точки, з якої ми дивимося на сцену, та точки, у яку ми дивимося на сцені, вектора направленого вгору, для розуміння положення камери відносно сцени, вертикального поля зору у градусах і співвідношення сторін екрану. Для визначення променя по координатам потрібно зважати на вертикальний і горизонтальний розмір зображення, вектор напрямку, у який дивиться камера, верхній лівий кут зображення, від якого і будуть рахуватися значення переданих координат x та y зображення, і точку, у якій розташована камера.

3.2 Сцена

Після того як відповідний промінь отримано, потрібно виконати процедуру трасування його шляху. Для цього було визначено клас сцена, який містить вектор усіх об'єктів у ній. Щоб знайти який з об'єктів був перетнутий променем, використовується простий метод перебору усіх об'єктів у сцені. У результаті буде зібрано інформацію про перетнутий об'єкт та повернуто значення "істинна" або значення "хиба", що залежить від наявності перетину хоч з одним об'єктом. Якщо жоден об'єкт не було перетнуто, то варто

повернути колір фону. У випадку перетину спочатку потрібно визначити чи є об'єкт світлом, щоб додати його яскравість. Наступним кроком буде визначення кольору об'єкта, який буде помножено на колір наступних об'єктів, які перетне промінь. Процес відслідковування шляху припиниться після досягнення ліміту можливих відбивань, що визначений заздалегідь.

3.3 Інформація про перетин променем об'єкта

Оскільки після перетину променем об'єкту нам необхідно зібрати багато інформації та передати її на наступні етапи виконання програми, її досить зручно описати у вигляді структури. При перетині сфери нам потрібно повернути різні дані: значення t, яке було визначено після вирішення рівняння сфери та прямої, щоб визначити найближчий об'єкт який перетнув промінь, точку перетину променя і сфери та нормаль для цієї точки, при чому ці дані буде використано для визначення відбитого чи заломленого променя, інформацію про сам матеріал та значення u і v для обрахунків текстури. Всі ці дані будуть заповнюватися у відповідному методі класу сфери.

3.4 Матеріали та текстури

Для визначення фінального кольору буде використано кольори усіх об'єктів які перетне промінь на своєму шляху після відбивань та заломлень. Колір кожного об'єкта буде визначено за допомогою його текстури. Якщо текстура буде суцільним кольором, то цей колір і буде використано, а у випадку з текстурою-зображенням буде використано колір зображення після визначення відповідності між координатою перетину променя і сфери та точкою на текстурі-зображенні.

Матеріал буде використано для визначення відбитого чи заломленого променя після його зіткнення з об’єктом.

3.5 Згладжування

Для зображення буде використано згладжування, тому для кожного пікселя буде створено декілька окремих променів, кожен з яких поверне певний колір. Сума усіх кольорів буде поділена на кількість променів для одного пікселя, щоб знайти середнє значення. Наступним кроком буде застосування гамма-корекції, шляхом знаходження кореня від цього значення. Остаточний результат можна виводити.

3.6 Бібліотеки для створення графічного інтерфейсу

Для наочної демонстрації результату було прийнято рішення використати бібліотеку Dear ImGui. Ця бібліотека буде відповідати за створення вікна з зображенням та виведення додаткової інформації, на зразок розміру зображення. Разом з нею було обрано використати графічний API OpenGL та бібліотеку GLFW. Першочергово потрібно створити вікно програми у системі windows, у середині якого уже буде знаходитися вікно, що створено бібліотекою Dear ImGui. За створення вікна програми відповідатиме OpenGL, але оскільки цей API є достатньо низькорівневим, то використання бібліотеки GLFW спростить процес роботи з ним.

3.7 Виведення результату

Програма буде виводити остаточний результат у вигляді 4 значень яскравості для кожного пікселя: червоного, зеленого, синього спектру та значення прозорості, яке дорівнює 255. Усі дані буде записано у вектор, розмір якого дорівнює ширині зображення помноженої на висоту зображення помноженої на 4. Масив цих значень буде завантажено у текстуру з використанням OpenGL та виведено за допомогою бібліотеки Dear ImGui. Зображення створюється поступово та досить повільно, тому було впроваджено механізм виведення проміжних етапів створення зображення.

Таким чином, при кожному оновленні вікна, що має відбуватися 60 разів на секунду, буде оновлено значення текстури OpenGL. Нове значення для текстури буде поточним варіантом зображення, яке буде отримано від програми.

3.8 Оптимізація

Для пришвидшення процесу створення зображення можна використати можливість паралельних обчислень. Найпростіший варіант оптимізація полягатиме у тому, що ми можемо виділити окремий потік для цілого рядка. Таким чином можна досягти одночасного створення декількох рядків різними потоками. Найпростіший спосіб реалізувати таку ідею полягатиме у використанні можливостей паралельного виконання циклу for, яке було додано у 17 версії мови C++. Достатньо визначити структуру, по якій можна пройтися ітеративно, і цикл зможе оброблювати різні елементи цієї структури одночасно. Оскільки основна частина програми полягає у проходженні двох циклів, один з яких вкладений у інший, для визначення координати зображення, то вектор з усіма цілими числами від 0 до значення висоти зображення буде достатнім для паралельного циклу.

Висновки

У цій роботі було розроблено застосунок, який може створювати різноманітні 3D зображення. Для їх створення була використана технологія трасування променів, завдяки якій зображення створюються шляхом відслідковування шляху променів для визначення усіх об’єктів, які він зустріне на своєму шляху. Після зіткнення з об’єктом промінь буде змінювати напрямок свого руху, який буде залежати від матеріалу об’єкта.

Серед матеріалів було визначено матові, які змушують промінь рухатися у випадкову сторону після відбиття, металеві, які відбивають промені на той самий кут, з яким він зіткнувся з об’єктом, і той матеріал, що заломлює промені. Окремо було визначено матеріал для світлових об’єктів.

Для визначення кольору об’єктів було впроваджено систему текстур, які можуть передавати або суцільний колір, або колір з певного зображення, яке можна накласти на об’єкт. Текстура з зображенням визначає відповідність між точкою на об’єкті та пікселем на текстурі-зображенні.

Виведення кінцевого зображення на екран здійснюється за допомогою бібліотек Dear ImGui, OpenGL та GLFW. Після запуску програми буде виведено вікно, на якому буде міститися зображення, що буде поступово створюватися програмою, використовуючи технологію трасування променів.

Список джерел

1. Dear ImGui. URL: https://github.com/ocornut/imgui/wiki.
2. de Greve B. Reflections and Refractions in Ray Tracing. 2006.
3. Driver S. Refraction in a sphere. URL: https://samdriver.xyz/article/refraction-sphere.
4. GLFW. URL: https://www.glfw.org.
5. OpenGL. URL: https://www.opengl.org.
6. Shirley P. Ray tracing in one weekend. URL: https://raytracing.github.io/books/RayTracingInOneWeekend.html.
7. Shirley P. Ray tracing: the next week. URL: https://raytracing.github.io/books/RayTracingTheNextWeek.html.
8. Shirley P. Ray tracing: the rest of your life. URL: https://raytracing.github.io/books/RayTracingTheRestOfYourLife.html.