DOI: https://doi.org/10.36910/6775-2524-0560-2022-49-11
УДК 004.04
Фещенко Іван Олександрович, магістрант
https://orcid.org/0000-0003-2884-4822
Дадиверін Віталій Валерійович, магістрант
https://orcid.org/0000-0001-5121-2263
Потапова Катерина Романівна, к.т.н., доцент
https://orcid.org/0000-0002-3347-6350
Наливайчук Микола Васильович, к.т.н., ст. викладач
https://orcid.org/0000-0002-8942-9844
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ТЕХНОЛОГІЯ СУПЕРСЕМПЛІНГУ ДЛЯ ПОКРАЩЕННЯ ПРОДУКТИВНОСТІ РЕНДЕРИНГУ ЗОБРАЖЕННЯ МЕТОДОМ ТРАСУВАННЯ ПРОМЕНІВ У РЕАЛЬНОМУ ЧАСІ

Фещенко І. О., Дадиверін В. В., Потапова К.Р., Наливайчук М.В. Адаптування технології суперсемплінгу для збільшення продуктивності рендерингу зображення методом трасування променів у реальному часі. У статті представлено варіант адаптації технології суперсемплінгу, або надлишкової вибірки згладжування, для збільшення продуктивності рендерингу зображення методом трасування променів у реальному часі. Запропоновано модифікований підхід з різною роздільною здатністю. Замість того, щоб дотримуватися жорсткої структури сітки квадродерев, яка властива текстурам і збереженим пікселям, для збору зразків використовується трикутна сітка. Ця трикутна сітка фактично накладається на традиційний формат текстури, який використовується для збору зразків у офсетному візерунку, а потім повертається до традиційної сітки для відображення на екрані. Ми вважаємо, що використання цієї трикутної вибірки потребуватиме меншої кількості вибірок, покращуючи ефективність алгоритму без погіршення якості кінцевої візуалізації. Представлено результати тестування адаптованого алгоритму відносно немодифікованого методу рендерингу.
Ключові слова: суперсемплінг, адаптивний семплінг, трасування променів, рендеринг, світло, роздільна здатність, графічний прискорювач.

Feshchenko I.O., Dadyverin V.V., Potapova K.R., Nalyvaichuk M.V. Adaptation of supersampling technology to increase the performance of real-time ray tracing image rendering. The article presents an option for adapting the supersampling technology, or oversampling smoothing, to increase the performance of image rendering using real-time ray tracing. A modified approach with different resolutions is proposed. Instead of following the rigid quad tree grid structure inherent in textures and stored pixels, a triangular grid is used to collect samples. This triangular mesh is actually superimposed on the traditional texture format used to collect the samples in the offset pattern and then returned to the traditional mesh for display on the screen. We believe that using this triangular sample will require fewer samples, improving the efficiency of the algorithm without compromising the quality of the final rendering. The results of testing the adapted algorithm relative to the unmodified rendering method are presented.
Keywords: supersampling, adaptive sampling, ray tracing, rendering, light, resolution, graphics accelerator.

Постановка проблеми.

Головним видом подання інформації користувачу використовуючи електроні обчислювальні пристрої є графічне відображення на дисплеї. Одною з основних цілей сучасної комп'ютерної графіки є генерування зображення, якість якого близька до фотореалістичної. Метод трасування променів є перспективним варіантом розвитку рендерингу, але достатньо ресурсозатратним. Проблеми, які вирішує даний метод порівняно до інших:

- якісне та реалістичне результуюче зображення;
- відсутність апроксимації при рендерингу гладких об'єктів;
- можливість паралелізації обчислень на рівні алгоритму;
- мала залежність складності алгоритму від складності схеми.

Аналіз існуючих рішень.

Існує кілька підходів, які були розроблені для зменшення або усунення проблем із накладанням спектрів у сценах із трасуванням променів. Більшість, однак, створює високу вартість продуктивності через необхідність великої кількості променів на піксель, як у моделях розподіленого трасування променів. Однією з таких областей дослідження є техніка, відома як адаптивна супервибірка. Завдяки цьому процесу пікселі можуть бути ефективно розділені за

потреби залежно від швидкості зміни між пікселями та наступними субпікселями. Потім можна використовувати промені для визначення конкретного кольору субпікселів, результати яких можна об'єднати разом, щоб відтворити справжній колір повного пікселя. Цей зв'язок також може бути спрямований у зворотному напрямку, у процесі, що називається недостатньою дискретизацією, дозволяючи одному променю вміщувати колір кількох пікселів у регіонах із невеликими змінами. Разом ці методи разом називаються адаптивною вибіркою. Один із способів реалізації адаптивної вибірки передбачає використання кількох сіток різної роздільної здатності, починаючи від більш грубих варіантів цільової роздільної здатності до все більш тонких кратних цільової роздільної здатності. Застосування цього підходу до адаптивної вибірки з різною роздільною здатністю дозволяє покращити згладжування зображень, що відтворюються, із покращеним часом відтворення порівняно з візуалізаціями з простою вибіркою та супервибіркою.

Постановка завдання.

Метою роботи є дослідження можливості оптимізації методу трасування променів для відображення графічної сцени задля підвищення продуктивності.

Для досягнення мети вирішуються наступні наукові завдання:

- розглянути загальний алгоритм трасування променів та виконати його опис;
- розглянути алгоритм адаптивної вибірки;
- розглянути метод почергового рендерингу у текстури з різною роздільною здатністю;
- дослідити можливість використання трикутної сітки для зменшення зразків текстури і оцінити ефективність даного підходу.

Виклад основного матеріалу дослідження.

Трасування променів – це техніка рендерингу, яка відстежує шлях світла у вигляді пікселів на площині зображення та імітує ефект зіткнення з об'єктами в сцені. Цей метод дає змогу досягти більшої реалістичності, ніж звичайні методи рендерингу, але вона вимагає великих обчислювальних витрат і створює велике навантаження на графічний прискорювач. З цієї причини трасування променів найбільше підходить для відносно тривалого нехтувано малого часу рендерингу, наприклад, для нерухомих комп'ютерних зображень і візуальних ефектів у кіно й телебаченні, але, як правило, не підходить для застосунків реального часу, де важлива швидкість покадрового рендерингу, наприклад, для відеоігор.

Однак, останніми роками апаратне прискорення для трасування променів у реальному часі стало стандартним для комерційних відеокарт, і графічні API наслідували цей приклад, даючи змогу додавати цей метод до програмного забезпечення, такого як ігри, де важлива продуктивність у реальному часі.

Принципи алгоритму трасування променів засновані на фізичних законах поширення світла. Таким чином, кожен об'єкт у сцені має матеріал із такими властивостями:

- Випромінювання енергії (англ. emittance) – кількість і довжина хвилі світла, що випромінюється об'єктом.
- Шорсткість (англ. roughness) – наскільки сильно розсіюються промені світла під час потрапляння на об'єкт.
- Відбивна здатність (англ. reflectance) – кількість і довжина хвилі світла, відбитого об'єктом.
- Прозорість (англ. transparency) – відношення кількості світла, що проходить через об'єкт, до кількості відбитого світла.
- Заломлення (англ. refraction) – міра заломлення світла.

Узагальнена схема трасування променів представлена на рисунку 1 і може бути описана наступними етапами:

- Кидання променя з кожного пікселя.
- Перевірка чи було перетинання об'єкта променем. Якщо так – див. пункт 3, якщо ні – див. пункт 5.
- Обчислення впливу кольору об'єкта на колір пікселя.
- Якщо кількість ітерацій не перевищена, то обчислити новий напрямок променя та повернутися до пункту 1, інакше закінчити роботу алгоритму.
- Обчислення впливу кольору фонового зображення на колір пікселя та закінчити роботу алгоритму.

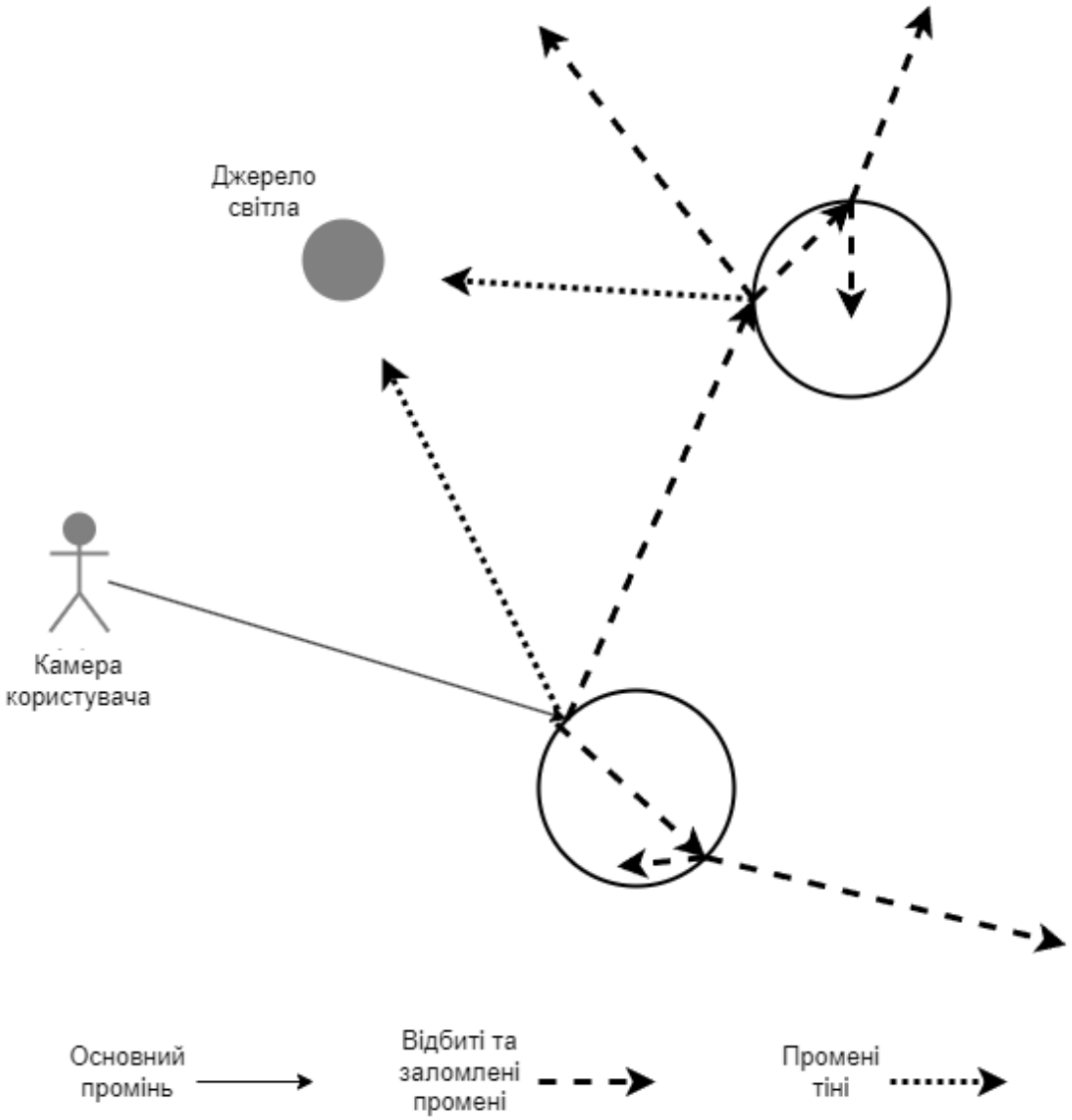

Рис. 1. Схематичне зображення алгоритму рейтрейсингу

Розглянемо метод адаптивної вибірки (англ. adaptive sampling), яку використаємо як один з елементів оптимізації для підвищення продуктивності. Адаптивна вибірка – це алгоритм, що складається з двох процесів, пов'язаних рівнем деталізації необхідним в сцені. На ділянках, де мало деталей і відносно незначні зміни піксель до пікселя, було б достатньо використовувати один промінь для представлення більших блоків пікселів. І навпаки, на ділянках, де спостерігається більша швидкість зміни між пікселями, більше одного променя на піксель може знадобитися для точного визначення правильного кольору пікселя. На рисунку 2 зображено приклад виділення таких ділянок. Хоча геометрія на сцені спрощена, є достатня кількість деталей для перегляду, що також ускладнено шаховим візерунком текстури підлоги. У цій сцені ми можемо виділити всі регіони, де необхідно визначити супервибірку, відповідним зеленим кольором. Примітно, що виділення лежать уздовж країв об'єктів або країв шахової дошки. Це ж зображення можна використовувати, щоб зрозуміти, де можлива недостатня вибірка. Зверніть увагу, що більшість композицій об'єктів, які не відображаються активно інші об'єкти на сцені залишаються еквівалентного або дуже схожого кольору. Зважаючи на це, немає потреби витрачатися велику кількість променів, щоб визначити, що один піксель неба справді такого ж кольору, як сусідній піксель. Натомість один промінь можна використовувати для визначення кольору цілої області пікселів на отриманому зображенні.

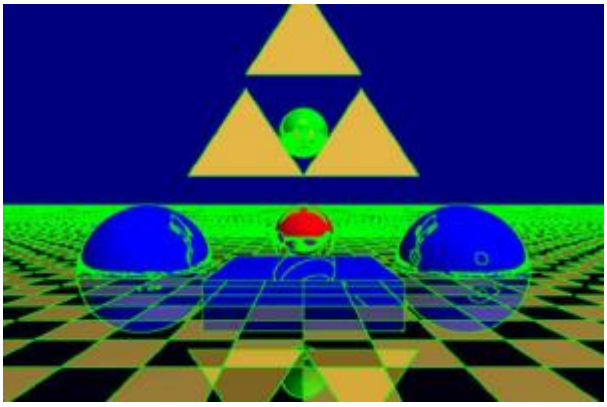

Рис. 2. Виділення зон для суперсемплінгу

Рендеринг у декілька текстур з різною роздільною здатністю

У запропонованому підході використовується підхід із різною роздільною здатністю, щоб визначити, коли кидати більше променів. Основна передумова передбачає спочатку рендеринг сцени на низькій роздільній здатності, що є менше на деяку ступінь двійки за цільову роздільну здатність. Ця роздільна здатність представляє стан із недостатньою дискретизацією візуалізації сцени, де один промінь представляє кілька пікселів візуалізації з цільовою роздільною здатністю. Інформація про сцену збирається з візуалізації з заданою роздільною здатністю, включаючи інформації від того, де і як промінь потрапив на найближчий об'єкт, до кольору отриманого на результуючому зображенні. Ці дані зберігаються в групі окремих текстур, кожна з яких відповідає певній роздільній здатності. Ці текстурні файли потім масштабуються, щоб відповідати наступним сценам з вищою роздільною здатністю. На цьому новому рівні роздільної здатності рендерер тестує текстури попередньої роздільної здатності щоб визначити, чи знімати нові промені з поточною роздільною здатністю. Це виконано шляхом порівняння значень текстур, що відповідають поточному пікселю, зі значеннями сусідніх пікселів. Якщо існує значна різниця між поточним пікселем і сусідніми, то передбачається, що всередині відбувається щось і нові промені мають кидатися у цей локальний блок вже з поточною роздільною здатністю рендерингу. І навпаки, якщо є невелика різниця між поточним пікселем і його сусідами, тоді немає потреби кидати нові промені, а результати візуалізації з нижчою роздільною здатністю можна бути прийняті як валідні і для поточної роздільної здатності. Цей процес триває, доки не буде досягнуто цільової роздільної здатності, після чого буде виконано всю недостатню вибірку. Отримані візуалізації потім можна зменшити до цільової роздільної здатності екрана, щоб зменшити вплив потенціальних дефектів.

Вибір сусідніх пікселів

Загалом, коли ми говоримо про піксельне оточення, ми маємо на увазі діапазон пікселів навколо даного цільового центрального пікселя, який програма «рендерить». Околиці з чотирьох пікселів описують сусідів, які знаходяться безпосередньо над/під і з боків цільового пікселя. А околиці восьми описують усі пікселі, які оточують центр, і так далі. Розмір даного околиці залежить від того, наскільки великі зміни діапазону можуть відбутися, але все ще впливають на вибірку цільового пікселя. Було виявлено, що збільшення розміру околиці пікселів не обов'язково корелює зі збільшенням точність. Навпаки, це може викликати помилкові спрацьовування, які вимагають детального вивчення, навіть якщо джерела змін знаходяться далеко від цільового пікселя. У випадку цієї статті для порівняння було обрано околиці, що складаються з восьми пікселів, що оточують цільовий піксель. Навіть якщо встановлено належний розмір околиці, необхідно визначити управління відмінностями з цільовим пікселем. Мета полягає в тому, щоб отримати нормалізоване значення, яке є результатом порівняння між цільовим пікселем і всіма його сусідніми пікселями.

Рис. 3. Схематичне зображення поступового збільшення роздільної здатності рендерингу

Для початку порівняння між цільовим пікселем і його сусідами описується як різниця між очікуваним значенням для пікселя від інтерполяції між протилежними сусідами та фактичною інформацією всередині пікселя. Цей процес, показаний на рисунку 4, виявився більш ефективним, ніж порівняння даних цільового пікселя з кожним сусідом окремо. Потім настає проблема нормалізації відмінностей між кожним із порівнянь до єдиного корисного значення. Двома поширеними методами є нормалізація L1 і L2. Нормалізація L1 просто бере суму абсолютних різниць, тоді як нормалізація L2 бере суму квадратів кожної різниці, перш ніж брати корінь із цього значення. За допомогою тестування визначено, що взяття середнього значення L1 нормалізації відмінностей дало найбільш розумну метрику для того, чи потрібні нові зразки променів. Таким чином, для кожної пари сусідніх пікселів різницю між їх інтерпольованим значенням і цільовим пікселем додавали до суми, яку потім усереднювали за кількістю сусідів. Потім цю різницю можна порівняти з довільним порогом. Якщо нормалізована різниця більша за порогове значення, це означає, що щось цікаве відбувається в околицях пікселів, і потрібні подальші зразки променів із поточною роздільною здатністю. В іншому випадку дані з текстур з нижчою роздільною здатністю можуть бути взяті та використані для поточної роздільної здатності.

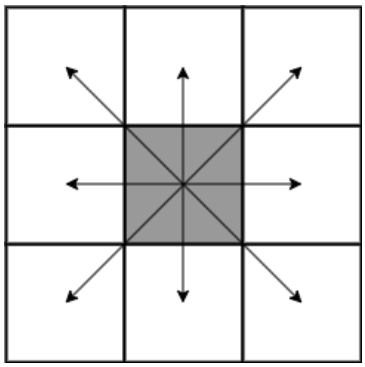

Рис. 4. Порівняння пікселів

Триадаптивна вибірка

Основна мета запропонованого методу полягає в тому, щоб змінити спосіб кидання променів на екран таким чином, щоб для досягнення того самого рівня інформації було необхідно менше зразків текстури. За стандартного підходу з різною роздільною здатністю один промінь потрапляє в центр кожного пікселя. Завдяки зміщенню напрямку променів на половину пікселя для кожного другого рядка створюється трикутна сітка. Ці дані все ще зберігаються у 2D об'єкт текстури, тому концептуально він записується та зчитується з квадратної сітки, як зазвичай. Метою цього процесу є зміна розподілу сусідніх пікселів. Під час аналізу розташування цільового пікселя є шість сусідів, з якими його можна порівняти. З областю розподілу між чотирма та вісьмома сусідами традиційного квадратного «семплера» ми припускаємо, що можна досягти подібного рівня якості порівняно з восьми піксельним околицями. Якщо це так, тоді виключення двох зразків із процесу має отримати незначний приріст продуктивності. Безпосередній недолік цієї системи полягає в тому, що, зміщуючи напрямки променів для кожного другого рядка, створюючи трикутну сітку, виникне очевидне спотворення в остаточному рендері, який відображається на екрані, який читається з квадратної сітки. Однак вирішення цього питання є

простим, оскільки програма повинна просто відобразити трикутну сітку назад у квадратну сітку. Це відображення можна здійснити шляхом інтерполяції відповідного кольору пікселя з навколишніх зразків. Тому для кожного пікселя в рядку зсуву необхідно інтерполювати між двома вибірками з обох боків або додати додаткові вибірки зверху та знизу пікселя, щоб отримати точний колір.

Продуктивність

Тестування кожного варіанту методу рендерингу відбувалося на однакових апаратних застосуваннях та графічних сценах. Для тесту буда взята за основу сцена корнелівської коробки. Як можна бачити на рисунку 5, класичний алгоритм рендерингу має продуктивність в, приблизно, 105 кадрів за секунду в середньому. У той самий час методи з використанням адаптивної вибірки показують приріст у приблизно 40%. Якщо порівнювати квадро- і триадаптивну вибірку, яка була представлена у статті, то маємо приріст у 5 кадрів у середньому. Це не є критичною різницею, але це доводить роботоздатність модифікації.

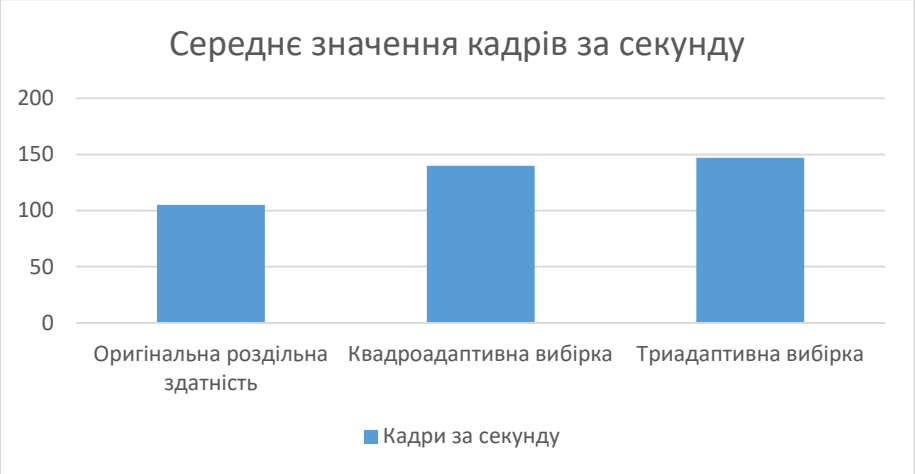

Рис. 5. Середнє значення кадрів за секунду при тестуванні

Висновки

У цій статті досліджено і запропоновано метод адаптування технології суперсемплінгу для збільшення продуктивності рендерингу зображення методом трасування променів у реальному часі. Такий підхід дозволить підвищити продуктивність систем рендерингу, які використовують алгоритм трасування променів, що було показано під час тестування.

Список бібліографічного опису

1. Дженетті, Дж. Трасування променів з адаптивною наддискретизацією в просторі об'єктів / Дженетті, Дж., Д. Гордон., 1993
2. Хачісука, Т. Багатовимірна адаптивна вибірка та реконструкція для трасування променів / Хачісука, Т., В. Ярош, Р. П. Вайстроффер., 2008. – (ACM Trans. Graph.).
3. Метт П. Фізично засноване рендеринг: від теорії до впровадження / П. Метт, Дж. Венцель, Г. Грег., 2016
4. NVIDIA TURING GPU ARCHITECTURE [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.
5. Фещенко І. О., Дадиверін В. В., Потапова К. Р. ОСОБЛИВОСТІ АПАРАТНО-ПРИСКОРЕНОГО АЛГОРИТМУ РЕНДЕРИНГУ НА ОСНОВІ ТРАСУВАННЯ ПРОМЕНІВ. MODERN RESEARCH IN WORLD SCIENCE, Lviv, 4-6 September 2022, 264-269с.

References

1. Genetti, J. Ray Tracing With Adaptive Supersampling in Object Space / Genetti, J., D. Gordon., 1993
2. Hachisuka, T. Multidimensional adaptive sampling and reconstruction for ray tracing / Hachisuka, T., W. Jarosz, R. P. Weistroffer., 2008. – (ACM Trans. Graph.).
3. Matt P. Physically Based Rendering: From Theory to Implementation / P. Matt, J. Wenzel, H. Greg., 2016.
4. NVIDIA TURING GPU ARCHITECTURE [E-source]. – 2018. – Resource access mode: https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.
5. Feshchenko I. O., Dadiverin V. V., Potapova K. R. FEATURES OF THE HARDWARE-ACCELERATED RENDERING ALGORITHM BASED ON RAY TRACING. MODERN RESEARCH IN WORLD SCIENCE, Lviv, September 4-6, 2022, pp. 264-269.