Programowanie II R — Projekt Zaliczeniowy

Śledzenie Promieni (Ray Tracing)

Autor: Piotr Dziekan

1. Wstęp

W optyce geometrycznej światło przemieszcza się po liniach prostych (promieniach) ze źródła, ulega odbiciom i załamaniom na granicach ośrodków, a ostatecznie trafia do detektora (np. ludzkiego oka lub matrycy aparatu). Analogiczny, ale odwrócony model rozchodzenia się światła wykorzystuje się do tworzenia grafiki komputerowej metodą śledzenia promieni (*backward ray tracing*). Promienie wypuszczane są z kamery w głąb wirtualnej sceny, a po drodze badane są ich przecięcia z obiektami geometrycznymi.

Celem niniejszego zadania jest napisanie programu do śledzenia promieni i wykorzystanie go do wygenerowania obrazu sceny składającej się z prostych brył.

2. Model

2.1. Promień światła

Promień określany jest przez punkt początkowy $\vec{O}$ i przez znormalizowany wektor kierunku ruchu promienia $\vec{D}$. Rozchodzenie się promienia opisuje następująca funkcja:

$$\vec{R}(t) = \vec{O} + t \cdot \vec{D}, \quad \text{dla } t > 0. \tag{1}$$

Do opisu barwy światła użyj modelu RGB. W tym modelu barwa opisana jest przez wektor $\overrightarrow{C_{RGB}}$ składający się z trzech liczb z przedziału $[0, 1]$. Liczby te kolejno opisują nasycenie kolorami czerwonym, zielonym i niebieskim.

2.2. Widok Sceny

Kamera znajduje się w środku trójwymiarowego układu współrzędnych kartezjańskich, $\vec{E} = (0, 0, 0)$. W odległości d (nazywanej ogniskową, przyjmujemy $d = 1$) od kamery znajduje się płaszczyzna prostopadła do osi z. Na tej płaszczyźnie znajduje się ekran podzielony na piksele (fig. 1). Szerokość ekranu (odległość między środkami skrajnych pikseli) wynosi $h_x = 4$, a jego wysokość (analogicznie) $h_y = 2$. Celem programu jest ustalenie koloru każdego z pikseli na podstawie modelowania promieni przez nie przechodzących. W podstawowej wersji programu modelowany jest jeden promień dla każdego z pikseli, przechodzący od kamery przez jego środek. Kierunek rozchodzenia się promienia od kamery w kierunku danego piksela to:

$$\vec{D} = \frac{\vec{P_{ij}} - \vec{E}}{\|\vec{P_{ij}} - \vec{E}\|}, \tag{2}$$

gdzie $\vec{P_{ij}}$ to położenie środka piksela.

2.3. Obiekty w scenie

Scena powinna zawierać sferyczne obiekty, opisane przez położenie środka $\vec{C}$ i promień r. Promień światła zderzy się ze sferą, jeśli istnieje rzeczywiste rozwiązanie równania:

$$\left(\vec{C} - \vec{R}(t)\right) \cdot \left(\vec{C} - \vec{R}(t)\right) = r^2 \tag{3}$$

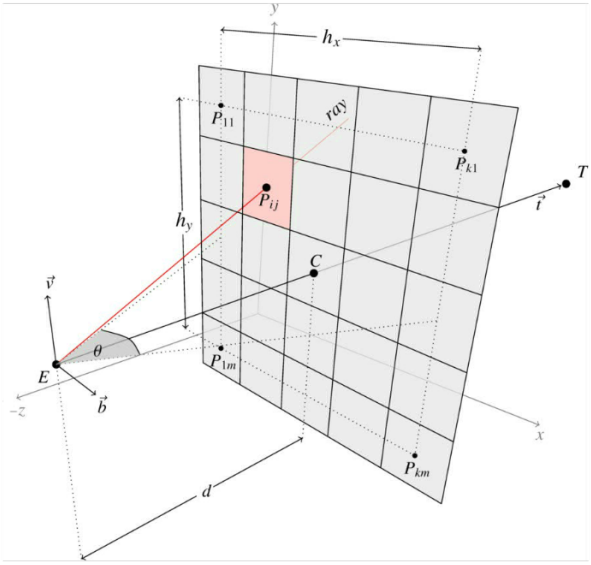

Figure 1: Schemat rozchodzenia się promieni od kamery przez piksele ekranu. Źródło: https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

Punkt przecięcia bliższy źródłu promienia światła odpowiada rozwiązaniu o mniejszej wartości t.

Zderzenie promienia z obiektem powoduje zmianę jego barwy (absorpcja) i zmianę kierunku propagacji. Oba efekty zależne są od rodzaju materiału, z którego zrobiony jest obiekt. Absorpcję opisuje wektor albedo $\overrightarrow{A_{RGB}}$ o trzech składowych z przedziału $[0, 1]$:

$$\overrightarrow{C_{RGB}}^{+} = \overrightarrow{A_{RGB}} \circ \overrightarrow{C_{RGB}}^{-} \,, \tag{4}$$

gdzie $\overrightarrow{C_{RGB}}^{-}$ ($\overrightarrow{C_{RGB}}^{+}$) to barwa przed zderzeniem (po zderzeniu), a $\circ$ oznacza mnożenie wektorów element po elemencie.

Zmiana kierunku propagacji zależy od tego, czy materiał jest lustrzany (odbicie), czy matowy (rozproszenie).

2.4. Odbicie

Kierunek promienia odbitego $\vec{D}^{+}$ wynosi:

$$\vec{D}^{+} = \vec{D}^{-} - 2(\vec{D}^{-} \cdot \vec{N})\vec{N} \,, \tag{5}$$

gdzie $\vec{D}^{-}$ to kierunek promienia przed odbiciem a $\vec{N}$ to jednostkowy wektor prostopadły do płaszczyzny w punkcie odbicia.

2.5. Rozproszenie

Rozproszenie powoduje emisję promieni w wielu kierunkach. Dla uproszczenia symulacji przyjmujemy jednak, że skutkuje ono emisją jednego promienia w losowym kierunku. Kierunek losujemy zgodnie z lambertowskim modelem rozpraszania: prawdopodobieństwo rozproszenia w danym kierunku proporcjonalne jest do $\cos\theta$, gdzie θ to kąt między danym kierunkiem a $\vec{N}$. Kierunek taki można ustalić, losując z jednorodnym prawdopodobieństwem punkt $\vec{S}$ znajdujący się na powierzchni sfery o promieniu 1 stycznej do obiektu rozpraszającego w punkcie odbicia $\vec{P}$ (fig. 2).

Do losowania punktu na powierzchni sfery można użyć algorytmu Monte Carlo:

1. Losujemy trzy współrzędne $\vec{X} = (x, y, z)$ z zakresu $[-1, 1]$.
2. Powtarzamy losowanie, jeśli odległość tego punktu od początku układu współrzędnych jest większa niż 1.
3. Punkt na sferze wskazuje znormalizowany wektor $\frac{\vec{X}}{|\vec{X}|}$.

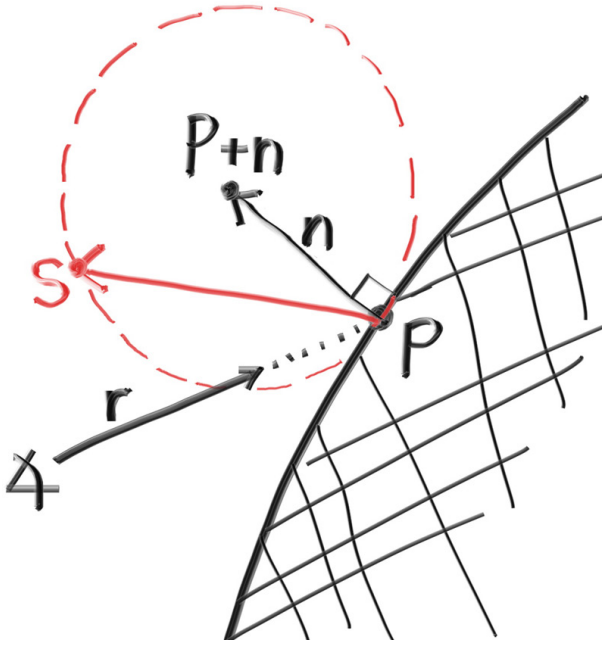

Figure 2: Losowanie kierunku promienia w rozpraszaniu lambertowskim. Źródło: https://raytracing.github.io/books/RayTracingInOneWeekend.html

2.6. Tło

Jeśli promień nie trafia w żaden obiekt, jego barwa równa jest barwie tła. Wybór barwy tła należy do studenta. Może to być jednorodna barwa, gradient barw (np. niebo o odcieniach niebieskiego zależnych od składowej y wektora kierunku), itp.

3. Implementacja

Kod musi zawierać następujące elementy:

- Klasa wektora 3D obsługująca potrzebne operacje matematyczne.
- Klasa reprezentująca promień światła.
- Klasa abstrakcyjna reprezentująca dowolny obiekt na scenie. Wszystkie klasy obiektów (w wersji podstawowej tylko sfera) powinny po niej dziedziczyć.
- Klasa abstrakcyjna reprezentująca dowolny rodzaj materiału, z którego zrobiony jest obiekt. Wszystkie klasy materiałów (lustrzany, matowy) powinny po niej dziedziczyć.
- Funkcja służąca do śledzenia danego promienia, zwracająca jego barwę. W razie zderzenia z obiektem funkcja wywoływana jest rekurencyjnie, aby śledzić promień odbity. Jeśli liczba odbić jest większa niż parametr `max_depth`, zwracany jest kolor czarny $(0, 0, 0)$.

Inne wymagania:

- Obiekty w scenie przechowywane są w kontenerze `std::vector` zawierającym wskaźniki do nich.
- Kod powinien wykorzystywać jedynie inteligentne wskaźniki (`std::unique_ptr`, `std::shared_ptr`).

Modelowana scena powinna być zapisana w formacie `ppm` z 16-bitową głębią kolorów (co oznacza, że barwa określana jest przez trzy liczby całkowite z zakresu $[0, 65535]$ określające nasycenie RGB). Jest to plik tekstowy, który w kolejnych linijkach zawiera:

1. Identyfikator: `P3`
2. Wymiary: `nx ny`, gdzie nx i ny to liczba pikseli w kierunku x i y.
3. Kodowanie kolorów: maksymalną wartość nasycenia koloru (`65535`).
4. Każda kolejna linijka koduje kolor jednego piksela (format `R G B`), zaczynając od lewego górnego rogu i idąc w prawo.

4. Wymagania minimalne symulacji (na 75% punktów)

1. **Rozdzielczość generowanego obrazu:** Minimum 800×400 pikseli (należy zachować proporcje 2:1).
2. **Scena:** Scena musi składać się z minimum czterech lustrzanych sfer o różnym albedo.
3. **Maksymalna liczba odbić:** `max_depth` ≥ 10.

Do projektu należy dołączyć sprawozdanie w formacie PDF zawierające:

- Wyrenderowany obraz (obrazy) wygenerowany przez autorski program.
- Opis struktury kodu, napotkanych problemów i podjętych decyzji architektonicznych.

5. Rozszerzenie (15% punktów)

Do sceny należy dodać matowe sfery. Ze względu na to, że światło rozproszone modelowane jest za pomocą pojedynczego promienia, należy użyć wielu promieni początkowych (min. 10) na każdy piksel obrazu. Kierunek rozchodzenia się każdego z nich należy losowo wybrać wewnątrz piksela. Pozwoli to też wygładzić nienaturalnie ostre krawędzie obiektów (*antyaliasing*). Kolor piksela liczmy uśredniając po promieniach składowe $\overrightarrow{C_{RGB}}$.

6. Zadania dodatkowe (10% punktów)

Wybierz jedno z:

- **Nowe kształty:** Zaimplementowanie klasy `Plane` (płaszczyzna nieskończona) lub `Triangle` (trójkąt w przestrzeni 3D) z analitycznym wyznaczeniem punktu przecięcia.
- **Załamanie światła (refrakcja):** Implementacja klasy materiału przezroczystego realizującego prawo Snella oraz przybliżenie Schlicka dla wyznaczenia współczynnika odbicia Fresnela.
- **Źródło światła:** Dodaj do sceny punktowe źródła światła.

7. Materiały

- https://raytracing.github.io/books/RayTracingInOneWeekend.html#overview
- https://en.wikipedia.org/wiki/Ray_tracing_(graphics)