Uniwersytet Mikołaja Kopernika
Wydział Matematyki i Informatyki

Daniel Nadolny
nr albumu: 312887

Praca inżynierska
na kierunku informatyka

Ray Tracing w czasie rzeczywistym

Opiekun pracy dyplomowej
doktor Jakub Narębski
Wydział Matematyki i Informatyki

Toruń 2026

Pracę przyjmuję i akceptuję

. . . . . .

data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

. . . . . .

data i podpis pracownika dziekanatu

Spis treści

Wstęp

W 2018 roku NVIDIA przedstawiła światu nową generację kart graficznych nazwanych RTX. Od tego roku każda kolejna seria począwszy od serii 20 do teraz (seria 50) jest wyposażona w tzw. RT Cores. Rdzenie RT są specjalnie stworzone do przyspieszania obliczeń związanych ze śledzeniem promieni (dalej będę używał nazwy ray tracing), w szczególności przy testowaniu przecięcia promienia z trójkątem i przechodzenia przez strukturę danych zwaną BVH (ang. bounding volume hierarchy). Odpowiednikiem RT Cores w kartach graficznych od AMD są ”Ray accelerators”. Ray tracing jest bardzo wymagającym algorytmem pod względem obliczeniowym. Dodanie powyższych rozwiązań sprzętowych do GPU pozwoliły programistom implementowanie ray tracingu w czasie rzeczywistym np. w grach, gdzie obecnie w jednej scenie może pojawić się kilka milionów trójkątów (dotychczas ray tracing wykorzystywany był głownie w filmach).

W ramach pracy inżynierskiej stworzony został silnik graficzny przedstawiający ray tracing w czasie rzeczywistym, napisany jest w języku C++, wykorzystując bibliotekę DirectX 11 i win32. Interfejs użytkownika stworzony został za pomocą biblioteki ImGui.

Pierwszy rozdział tej pracy będzie poświęcony przedstawieniu podstaw ray tracingu, od opisania idei algorytmu do wyprowadzenia dwóch podstawowych procedur badania przecięć promienia z obiektami w scenie (promień-sfera i promień-trójkąt). W tym rozdziale poruszone będzie również zagadnienie optymalizacji silnika używając struktury danych BVH. W następnym rozdziale zostanie opisane zagadnienie materiałów. Materiały są jednym z najważniejszych tematów w ray tracingu, jak i ogólnie w grafice komputerowej, jeśli chodzi o aspekty wizualne. W trzecim rozdziale opisa-

ny będzie stworzony silnik i implementacje przedstawionych wcześniej algorytmów. W końcowej części pracy przedstawione zostaną wyniki testów wydajnościowych programu. Testy były przeprowadzone przed optymalizacją silnika i po optymalizacji, aby wykazać różnice wydajności.

Rozdział 1

Podstawy Ray Tracingu

1.1 Definicje i oznaczenia

W dalszej części będę posługiwać się takimi oznaczeniami:

- a - wartość skalarna.
- $\boldsymbol{v}$ - wektor, w większości przypadków $\boldsymbol{v} = (x, y, z)$.
- $\boldsymbol{n}$ - wektor normalny. Wektor prostopadły do danej powierzchni.
- $\boldsymbol{a} \cdot \boldsymbol{b}$ - iloczyn skalarny.
- $\boldsymbol{a} \times \boldsymbol{b}$ - iloczyn wektorowy.
- Promień - promień (ang. ray) jest podstawową strukturą w ray tracingu. Składa się on z punktu początkowego (ang. origin) i kierunku (ang. direction). Oba elementy zdefiniowane są jako wektory trójwymiarowe, z czego kierunek jest wektorem znormalizowanym (długość równa 1).

1.2 Algorytm ray tracingu

Algorytm ray tracingu znany jest już od 1979 roku, kiedy John Turner Whitted opublikował artykuł ”An improved illumination model for shaded

display” opisujący rekurencyjny ray tracing [PRZYPIS].

Ideą algorytmu jest naśladowanie światła. W opisie zachowania się światła i jego oddziaływania z materią korzysta się z teorii falowej i optyki geometrycznej. W rzeczywistości światło porusza się po liniach prostych, od źródła np. Słońca. Ray tracing działa odwrotnie, tzn. źródłem promieni jest ”oko” kamery i od niego wychodzi światło w generowaną scenę, ponieważ jak w rzeczywistości mózg człowieka ”renderuje” obraz korzystając tylko z tych promieni, które padają na siatkówkę w oku [PRZYPIS]. [TUTAJ WSTAWIĆ OBRAZEK PRZEDSTAWIAJĄCY ALGORYTM].

Symulowanie światła pozwala programistom na uzyskanie szczegółowych i poprawnych fizycznie efektów takich jak: odbicie, refrakcja itd. Wcześniej, przed erą ray tracingu w czasie rzeczywistym, programiści musieli korzystać z różnych sztuczek aby zaimplementować te efekty, dla przykładu odbicia tworzone były poprzez zrenderowanie danego obiektu drugi raz ale odwrotnie w np. kałurzy, lustrze. Korzystając z ray tracingu efekt odbicia jest dużo prostszy w implementacji.

Minusem algorytmu jest jego złożoność obliczeniowa. Dla przykładu, bez optymalizacji BVH mając model złożony z 100000 trójkątów, w rozdzielczości 2560x1440, mając ustawione 2 próbki na piksel i 5 odbiciach, program musiałby dla każdego piksela wykonać (w tej rozdzielczości mamy 3686400 pikseli) 1000000 testów.