Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie

Raytracing jest techniką służącą do generowania foto realistycznych obrazów scen 3D. Na przestrzeni lat technika ta ciągle się rozwijała. Doczekała się wielu modyfikacji, które usprawniają proces generowania realistycznej grafiki. Takimi technikami mogą być między innymi PathTracing, PhotonMapping, Radiostity i wiele innych. Z dnia na dzień wykorzystywanie raytracingu ciągle rośnie. W dzisiejszych czasach w grafice komputerowej oraz w kinematografii do uzyskania realistycznych efektów używana jest metoda śledzenia promieni. Dzięki takim zabiegom jesteśmy w stanie dosłownie zasymulować sceny oraz zjawiska, które nie muszą istnieć w rzeczywistym świecie. Czas generowania pojedynczej klatki/ujęcia takiej sceny niekiedy potrafi być liczony nawet w godzinach. Dlatego technika ta nie doczekała się jeszcze swojej wielkiej chwili w przemyśle rozrywkowym jakim są np. gry komputerowe oraz inne aplikacje generujące grafikę 3D w czasie rzeczywistym.

1.2 Dostępne technologie, pozwalające zrównoleglić obliczenia na kartach graficznych

Poniżej zaprezentowanych zostanie parę wybranych technologii wspomagających zrównoleglenie obliczeń. Niemniej jednak badania przeprowadzone i opisane w dalszej części pracy będą skupiały się na wykorzystaniu jednej z tych metod, a mianowicie technologii NVIDIA CUDA.

1.2.1 Open Computing Language (OpenCL)

Technologia tak zainicjowana została przez firmę Apple. Do inicjatywy i rozwijania tej technologii włączyły się w późniejszym czasie inne firmy takie jak: AMD, IBM, Intel, NVIDIA. W roku 2008 sformowana została grupa Khronos skupiająca powyższe firmy oraz wiele innych należących do branży IT. Grupa ta czuwa nad rozwojem technologii OpenCL. Technologia tak pozwala na pisanie kodu który jest przenośny między wieloma platformami: komputery, urządzenia przenośne, klastry obliczeniowe. OpenCL pozwala rozpraszać obliczenia na jednostki procesorowe CPU oraz na architektury graficzne GPU. Bardzo ważną zaletą OpenCL jest to, że pisanie z użyciem tej technologii nie jest zależne od sprzętu na jakim będzie ona uruchamiana. Model Platformy OpenCL znajduje się poniżej: (screen z platformy OpenCL)

1.2.2 ATI Stream Computing

Technologia ta została stworzona przez firmę AMD. Za pomocą tej platformy jesteśmy w stanie przeprowadzać złożone obliczenia na sprzęcie produkowanym przez AMD. W skład całego pakietu ATI Stream Computing wchodzi autorski język ATI Brook+ i kompilator tegoż języka. Dodatkowo ATI wspiera developerów własną biblioteka matematyczna (AMD Core Math Library) oraz narzędziami do profilowania wydajności kodu (Strem Kernel Analyzer). Model Platformy OpenCL znajduje się poniżej: (screen z platformy ATI Stream Computing)

1.2.3 NVIDIA CUDA

CUDA (Compute Unified Device Architecture) jest technologia opracowaną przez firmę NVIDIA. Swoje początki CUDA miała w 2007 roku i do dziś jest wiodącą technologią strumieniowego przetwarzania danych z wykorzystaniem układów graficznych GPU.

Rozdział 2

Cele pracy

2.1 Opracowanie techniki zrównoleglenia i przyspieszenia metody śledzenia promieni przy użyciu NVIDIA CUDA

Celem niniejszej pracy jest przeniesienie a zarazem zrównoleglenie algorytmu śledzenia promieni na procesory graficzne (GPU) firmy NVIDIA. Celem także jest przyśpieszenie obliczeń standardowego wstecznego Raytracingu w celu jak najszybszego generowania obrazów scen 3D.

2.2 Projekt uniwersalnej aplikacji - benchmark

W ramach projektu napisany został uniwersalny system Raytracingu działający na wielo rdzeniowych procesorach komputerowych (CPU), a także na kartach graficznych (GPU) firmy NVIDIA które obsługują technologie NVIDIA CUDA. Aplikacja testowa jest benchmarkiem, który jest w stanie przetestować zadane sceny 3D na wielu różnych konfiguracjach sprzętowych. Aplikacja ma za zadanie po uruchomieniu na komputerze użytkownika, testować wszelkie sceny z odpowiedniego katalogu. Dodatkowo zbierać potrzebne informacje o sprzęcie użytkownika oraz czasy generowania obrazów z każdej ze scen. Po przeprowadzeniu wszelkich testów aplikacja jest w stanie wyslać na adres e-mail developera (w tym przypadku autora pracy) wszelkie zgromadzone dane.

Rozdział 3

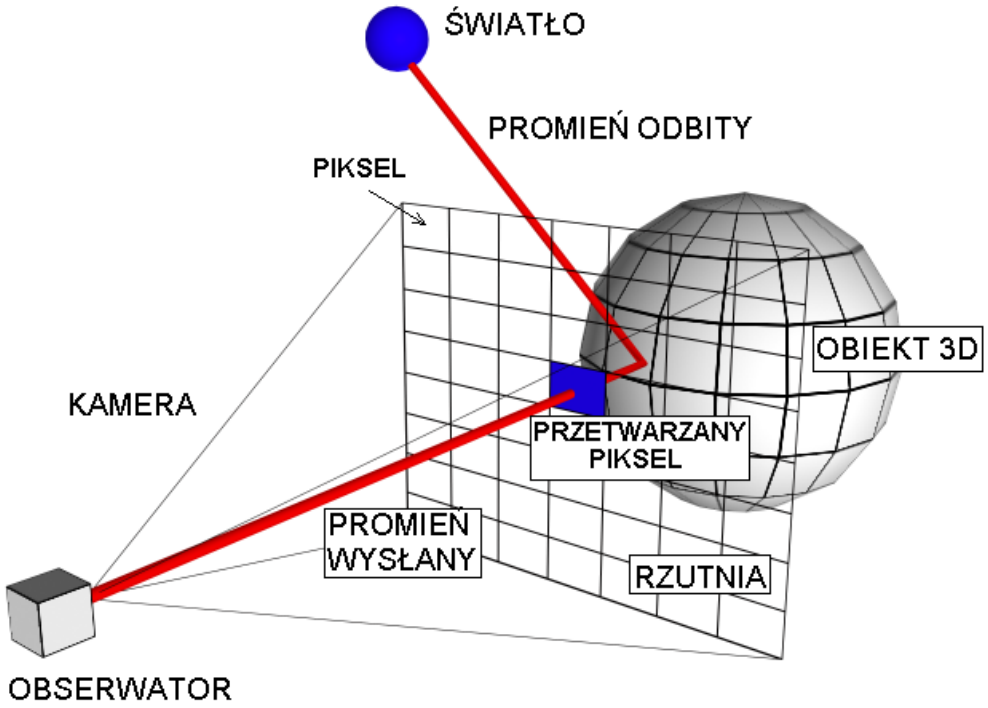
Wprowadzenie do Raytracingu

3.1 Wstępny opis

W rzeczywistym świecie promienie świetlne rozchodzą się od źródła światła do obiektów znajdujących się w świecie. Każde źródło światła wysyła nieskończoną liczbę swoich promieni świetlnych. Następnie te promienie odbijając się od obiektów i trafiają do oczu obserwatora powodując że widzi on określony kolor danego obiektu. Gdyby zaadaptować tą metodę do generowania realistycznej grafiki komputerowej, otrzymalibyśmy nieskończenie dokładny i realistyczny obraz. Z racji jednak na to, że sprzęt komputerowy ma ograniczone możliwości, a metoda ta jest bardzo nie efektywną metodą pod względem obliczeniowym. Najszerzej stosowaną metodą śledzenia promieni jest wsteczne śledzenie promieni (backward raytracing). W odróżnieniu od postępowego algorytmu śledzenia promieni (forward raytrcing), które opiera się na generowaniu jak największej liczby promieni dla każdego źródła światła. Algorytm wstecznego śledzenia promieni zakłada, że promienie śledzone są od obserwatora, poprzez scenę do obiektów z którymi kolidują. Poniżej przedstawiony jest poglądowy rysunek śledzenia pojedynczego promienia od obserwatora poprzez określony pixel na ekranie: 3.1

3.2 Rekursywna metoda śledzenia promieni

Przy omawianiu wstecznej metody śledzenia promieni warto wspomnieć o raytracingu rekursywnym. W zagadnieniu tym bada się rekurencyjnie promienie odbite zwierciadlane oraz załamane, które powstały z kolizji promieni pierwotnych z obiektami na scenie. Tak więc żywotność promienia pierwot-

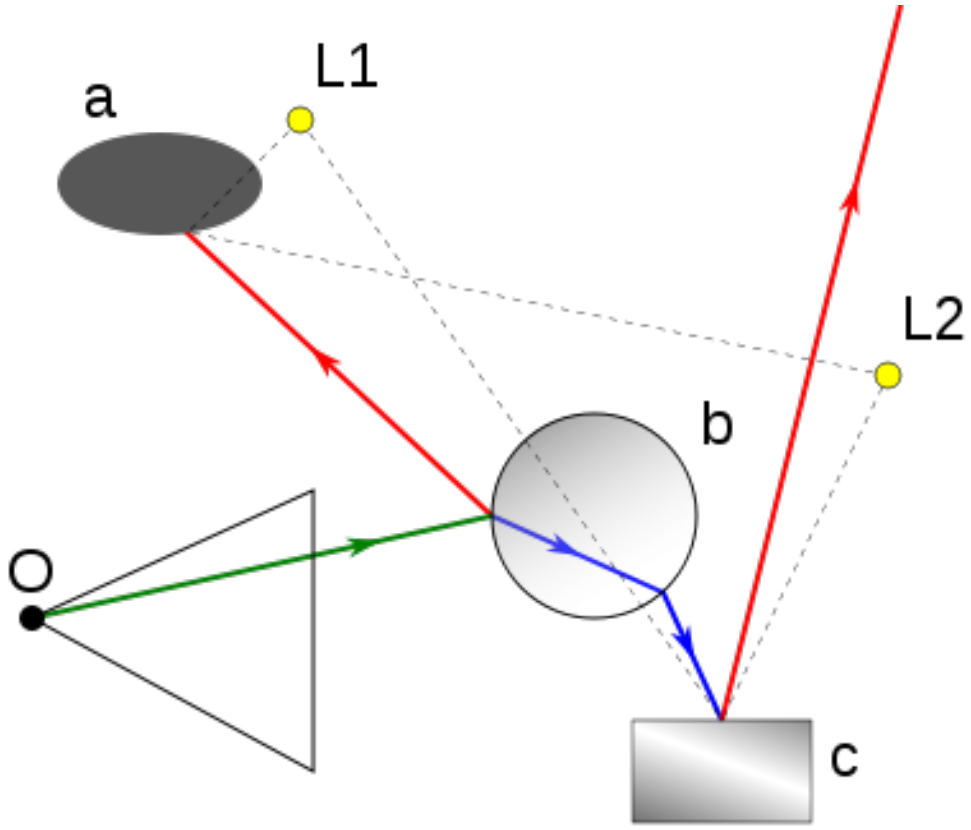


Rysunek 3.1: Sposób określania barwy piksela w raytracigu

nego wcale nie kończy się w momencie kolizji z obiektem sceny. To czy z danego promienia pierwotnego wygenerowane zostaną kolejne promienie w bardzo dużej mierze zależy od materiału jakim pokryty jest dany obiekt sceny. Z pomocą tego rekursywnej metody śledzenia promieni jesteśmy w stanie zasymulować obiekty lustrzane oraz obiekty półprzezroczyste. Rekurencja w tej metodzie trwa do osiągnięcia maksymalnego stopnia zagłębienia. Kolor wynikowy danego pojedynczego Pixela powstaje z sumy kolorów, obiektu w jaki trafił promień pierwotny oraz kolorów obiektów w jakie trafiły promienie pierwotne. Poniżej przedstawiony jest poglądowy schemat zasady działania rekursywnej metody śledzenia promieni: 3.2

3.3 Przedstawienie algorytmu śledzenia promieni

Śledzenie promieni przez scenę rozpoczyna się od obserwatora określanego często jako kamery występującej na scenie. Przez każdy pixel ekranu śledzone są promienie które poruszają się po scenie. Gdy któryś ze śledzonych promieni napotka obiekt i zacznie z nim kolidować.



Rysunek 3.2: Zasada działania rekursywnego algorytmu ray tracingu

3.4 Przykład szósty

Oto przykładowy wydruk:

```
1 /* ta funkcja oblicza a+b */
2 int sum(int a, int b)
3 {
4   int suma=0;
6   suma=a+b;
8   return suma;
9 }
```

Bibliografia

[1] J. Bloch. Effective Java. Addison–Wesley, Stoughton, USA, second edition, 2008. http://www.thinkingblackberry.com/.

[2] E. Ciurana. Developing with Google App Engine. Apress, Berkeley, USA, 2008.

[3] Community. BlackBerry Developer Zone. http://na.blackberry.com/eng/developers/.

[4] S. Hartwig and M. Buchmann. Empty Seats Traveling. http://research.nokia.com/files/NRC-TR-2007-003.pdf.

[5] C. King. Advanced BlackBerry Development. Apress, Berkeley, USA, 2009.

[6] A. Rizk. Thinking BlackBerry. http://www.thinkingblackberry.com/.

[7] D. Sanderson. Programming Google App Engine. O'Reilly Media, Sebastopol, USA, 2009.

[8] Charles Severance. Using Google App Engine. O'Reilly Media, Sebastopol, USA, 2009.

[9] Wikipedia. Stosunek liczby samochodów do zaludnienia. http://pl.wikipedia.org/wiki/Stosunek˙liczby˙samochod%C3%B3w˙do˙zaludnienia.

Listings

Spis rysunków

Spis tabel