Anna TOMASZEWSKA, Krzysztof STEFANOWSKI
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY, WYDZIAŁ INFORMATYKI,
ul. Żołnierska 52, 71-210 Szczecin

Efekt rozpraszania podpowierzchniowego z wykorzystaniem programowalnego procesora graficznego

Dr inż. Anna TOMASZEWSKA

Ukończyła studia na Wydziale Informatyki Politechniki Szczecińskiej w 2000 r. W 2003 r. obroniła pracę doktorską na tym samym wydziale. Obecnie pracuje jako adiunkt w Zakładzie Grafiki Komputerowej Instytutu Grafiki Komputerowej i Systemów Multimedialnych. Jej zainteresowania naukowe to przetwarzanie obrazów HDR, analiza percepcyjna oraz systemy graficzne czasu rzeczywistego.

e-mail: atomaszewska@wi.zut.edu.pl

Inż. Krzysztof STEFANOWSKI

Absolwent Zachodniopomorskiego Uniwersytetu Technologicznego na Wydziale Informatyki. Interesuje się grafiką czasu rzeczywistego i efektami wizualnymi w grach. W swojej pracy dyplomowej poruszył temat podpowierzchniowego rozpraszania światła. Obecnie rozpoczął pracę jako programista gier.

e-mail: kstefanowski@wi.zut.edu.pl

Streszczenie

W artykule zaprezentowano sposób obliczania w czasie rzeczywistym efektu podpowierzchniowego rozpraszania światła w obiektach częściowo przeźroczystych przy zwróceniu szczególnej uwagi na wydajność obliczeniową algorytmu. Algorytm zaprojektowanego pod kątem implementacji sprzętowej realizowanej na programowalnym procesorze graficznym. Dane przekazywane są do GPU w postaci zmiennych (uniform i attribute), gdzie wykorzystywane są do dalszych obliczeń. Porównanie wydajności prezentowanego podejścia z innymi algorytmami przedstawiono w podsumowaniu artykułu.

Słowa kluczowe: podpowierzchniowe rozpraszanie światła, percepcja, programowalny procesor graficzny, grafika komputerowa.

GPU based subsurface scattering effect

Abstract

In the paper there is presented the spherical harmonics (SH) based method for subsurface scattering and its GPU-based implementation. The described approach is modification of the Green's algorithm [1]. The 3D model thickness was encoded for each vertex in every possible direction. The algorithm is divided into two parts: the preprocessing executed on CPU and the visualization stage designed for GPU. The tests were carried out and described. They revealed the effectiveness of the obtained results. To verify the results, they were compared with those obtained from other algorithms. The results show efficiency benefits of the authors' algorithm in comparison with the comparable quality approaches. Moreover, the modification of the Green`s algorithm improves the quality of the subsurface scattering effect, as the unnatural effect of sharp curves visible on the final images is reduced. It is possible because in this approach the way the light goes through an object depends on the model thickness. The paper is organized as follows. In Section 2 the previous works are discussed. In Section 3 the application of subsurface scattering based on the spherical harmonics and its hardware implementation are presented. Section 5 shows the obtained results. At the end of the paper there are given some concluding remarks.

Keywords: subsurface scattering, perception, graphics processing unit, computer graphics.

1. Wstęp

Dostępne obecnie gry komputerowe oferują coraz lepszą grafikę zawierają efekty niemal do złudzenia odzwierciedlające rzeczywistość. Jeden z takich efektów można zauważyć w realnym świecie obserwując np. promienie światła przechodzące przez bryłę wosku, kiść winogron, czy prześwietloną ludzką skórę. Zjawisko podpowierzchniowego rozpraszania światła (rys. 1) stanowi jedno z wielu zagadnień, którymi zajmuje się grafika komputerowa. Symulacja takiego efektu w grach komputerowych wymaga niezwykle dużej szybkości generowania obrazów. Przy najczęściej stosowanym modelu oświetlenia upraszczającym w dużym stopniu rzeczywistość i uwzględniającym odbicie promienia światła w punkcie jego padania, realistyczna wizualizacja efektu jest niemożliwa. Bardziej rozbudowane modele zakładają wnikanie w materiał załamującego się promienia światła, jego rozpraszanie oraz opuszczenie w wielu miejscach jednocześnie.

Rys. 1. Efekt podpowierzchniowego rozpraszania światła. Światło przenika i rozprasza się w obiektach częściowo przeźroczystych
Fig. 1. Subsurface scattering effect. The light goes through and diffuses in semi-translucent objects

Wszelkie obliczenia wykonywane przez komputer związane z grafiką są na tyle czasochłonne, że nawet najszybsza karta graficzna ani procesor nie zapewniają uzyskania wyników w zadowalającym czasie jeżeli algorytm jest nieoptymalny. Istniejące metody umożliwiają symulację podpowierzchniowego rozpraszania światła w czasie rzeczywistym - jednak uzyskane wyniki w dalszym ciągu nie są satysfakcjonujące. Jakość otrzymywanych obrazów jest niezadowalająca bądź czas ich generowania jest zbyt długi by sprostać wciąż rosnącym oczekiwaniom.

W artykule zaprezentowano wydajny algorytm symulacji efektu, stanowiący syntezę zalet istniejących rozwiązań oraz jego sprzętową implementację. Przy projektowaniu algorytmu symulacji efektu podpowierzchniowego rozpraszania światła szczególną uwagę zwrócono na wydajność obliczeniową otrzymywanych obrazów.

W rozdziale 2 zaprezentowano przegląd istniejących algorytmów podpowierzchniowego rozpraszania światła. Prezentowane podejście, oraz jego implementację sprzętową przedstawiono w rozdziale 3. Rezultaty działania metody oraz przeprowadzone testy wydajnościowe zaprezentowano w rozdziale 4.

2. Przegląd technik podpowierzchniowego rozpraszania światła

Jednym z podstawowych elementów symulacji półprzezroczystych materiałów jest absorpcja światła. Aby wyliczyć jej wartość należy zmierzyć drogę jaką przebyło światło pod powierzchnia materiału. Jedną z metod szacowania odległości

jest metoda Green'a [1] bazująca na algorytmie rzucania cienia. W pierwszym przebiegu algorytmu scena jest renderowana z pozycji źródła światła. Odległość od każdego punktu sceny do źródła światła zapisywana jest do tekstury. W następnym przebiegu dla każdego punktu na scenie do tekstury zapisywana jest jego odległość od kamery. Ostatecznie kolor obiektu wyznaczany jest na podstawie różnicy pomiedzy głębokościami zapisanymi w teksturach obliczonymych w kolejnych przebiegach. W ten sposób wyznaczana jest droga, którą przebył promień światła wewnątrz obiektu. Algorytm Green'a dobrze symuluje zjawisko absorpcji światła oraz może być renderowany w czasie rzeczywistym. Jednak ostre przejścia kolorów w wynikowych obrazach stanowią nieprawidłowy i sztuczny efekt. Rozwiązanie można ulepszyć rozmywając mapę głębokości np. uśredniającym filtrem splotowym Gaussa [1]. Efektem tej operacji jest wygładzenie ostrych przejść kolorów. Wciąż jednak stopień rozproszenia nie uwzględnia grubości obiektu.

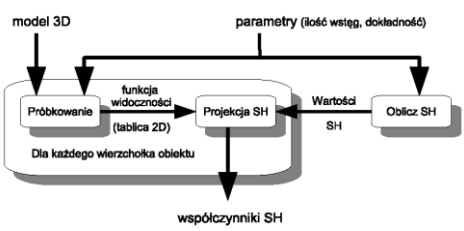
Kolejny algorytm, przeliczony transfer radiancji (PRT), jest metodą obliczania oświetlenia z uwzględnieniem powierzchniowych źródeł światła [2, 3]. Opisuje sposób interakcji światła na powierzchni oraz wnętrzu obiektu. Zakodowanie wartości funkcji transferu odbywa się najczęściej przy użyciu funkcji sferycznych harmonicznych. Rozwiązanie umożliwia symulację efektu w czasie rzeczywistym.

Model Patro, uwzględnia wyłącznie lokalny model oświetlenia, dostosowując się do niego i nie wprowadzając żadnych dodatkowych obliczeń do procesu wizualizacji obiektu [4]. Ingeruje on w samą strukturę modelu, zmieniając wektory normalne wierzchołków. Algorytm Patro daje wyniki nieporównywalnie gorsze od pozostałych metod, jest za to bardzo prosty i nie wprowadza dodatkowych obliczeń w czasie renderowania.

W kolejnych rozdziałach przedstawiono sprzetową implementację hybrydowego podejścia łączącego zalety przedstawionych algorytmów.

3. Algorytm

Prezentowany algorytm do symulacji efektu podpowierzchniowego rozpraszania światła (SHSSS - ang. *Spherical Harmonics SubSurface Scattering*) wykorzystuje podstawy fizyczne, zawiera jednak uproszczenia pozwalające na wykonanie obliczeń w czasie rzeczywistym. Punktem wyjściowym koncepcji jest algorytm Green'a, w którym grubość obiektu obliczana jest za pomocą algorytmu rzucania cienia. Zmiana sposobu wyznaczania grubości obiektu umożliwiła wzrost jakości oraz wydajności renderowanych obrazów. Dokonuje się wstępnego przeliczenia grubości obiektu. W celu jej zakodowania w każdym możliwym kierunku dla każdego wierzchołka modelu graficznego zastosowano funkcje sferyczne harmoniczne (SH) [5]. Ze względu na charakter dokonanej modyfikacji algorytm podzielono na dwa etapy: inicjalizacji oraz wizualizacji.



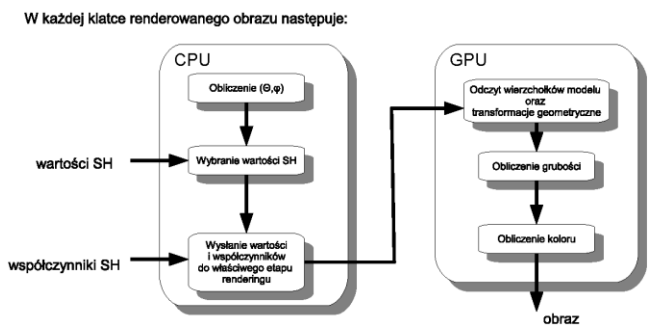
Rys. 2. Schemat procesu inicjalizacji. Dane wejściowe stanowią: model 3D oraz ilość wstęg oraz dokładność. Implementacji tego etapu dokonano na CPU
Fig. 2. An initialization schema. Input data: 3D model and parameters: number of spherical harmonics and precision

Na etapie inicjalizacji (rys. 2) realizowanym na CPU grubość obiektu obliczana i kodowana jest za pomocą współczynników SH. W tym celu:

- funkcja odległości modelu 3D próbkowana jest z punktu widzenia każdego wierzchołka,
- wyznaczane są wartości funkcji SH dla pełnej dziedziny kątów w układzie współrzędnych sferycznych,
- wykonywana jest projekcja funkcji SH wyznaczająca współczynniki SH.

Ponieważ jest to najbardziej kosztowny etap algorytmu, wskazane jest aby był on wcześniej przeliczony. Obliczone raz wartości współczynników SH nie zmieniają się później podczas renderingu.

W celu przyśpieszenia obliczeń drugi etap został podzielony na dwa podrzędne podetapy (rys. 3). Pierwszy wykonywany jest przez główny procesor, drugi przez procesor karty graficznej, wyspecjalizowanej do zadań graficznych. W pierwszym wyznaczane są indeksy współczynników, wykorzystywanych w kolejnych obliczeniach. Zależą one od położenia obiektu względem kamery oraz od lokalizacji źródeł światła. Na drugim podetapie, współczynniki SH są wykorzystywane do odtworzenia w czasie rzeczywistym przybliżonej grubości obiektu w kierunku światła. Obliczenie grubości obiektu polega na zsumowaniu kolejnych iloczynów współczynników i wartości SH na podstawie równania 1. Ilość iloczynów i sum rośnie z kwadratem ilości wstęg. Po wyznaczeniu grubości jest ona uwzględniana w modelu oświetlenia w celu obliczenia koloru.



Rys. 3. Schemat generowania efektu rozpraszania podpowierzchniowego światła. Zestaw operacji wykonywany jest w każdej klatce renderowanego obrazu
Fig. 3. A schema of the algorithm of a subsurface scattering effect. All operations are rendered in every image frame

$$\tilde{f}(s)=\sum_{l=0}^{n-1}\sum_{m=-l}^{l} c_l^m y_l^m(s)=\sum_{i=0}^{n^2} c_i y_i(s), \qquad (1)$$

gdzie: s – współrzędne sferyczne, f – funkcja odległości (grubości), c – współczynnik SH, y – wartość funkcji SH, n - ilość wstęg, m - numer wstęgi, l – numer pasma, i – indeks zastępczy.

4. Sprzętowa implementacja algorytmu

Do implementacji kodu wykorzystano język C++, do implementacji wizualizacji grafiki bibliotekę OpenGL 2.0 (z wykorzystaniem GLSL 1.1 – języka programowania GPU). W celu zredukowania wymiany danych między pamięcią RAM a procesorem graficznym, każdy wierzchołek modelu wraz z maksymalnie 36 współczynnikami SH przechowywany jest w pamięci karty graficznej w postaci struktury VBO (zmienne attribute). 36 wartości funkcji SH zależnych od kierunku światła jest obliczanych przez CPU i przesyłanych do GPU w każdej klatce animacji (zmienne uniform).

Na etapie uruchomienia programu realizowanego przez procesor graficzny, do pamięci GPU przesłano cały zestaw danych o wierzchołkach modelu oraz współczynnikach. Umieszczenie modelu i współczynników w pamięci karty graficznej odbywa się poprzez szereg funkcji udostępnionych przez OpenGL. Dane przechowywane są w VBO (ang. vertex buffer object). Jest to rozszerzenie OpenGL definiujące interfejs pozwalający na przechowywanie różnego rodzaju danych dla każdego wierzchołka w pamięci GPU o wysokiej wydajności. Dzięki temu nie jest konieczny każdorazowy transfer danych do GPU

z pamięci głównej RAM. Zaraz po załadowaniu i skompilowaniu programów cieniujących (vertex program i fragment program) pobierane są adresy zmiennych wykorzystywanych w programach. Dzięki temu możliwe jest załadowanie w odpowiednie miejsce danych do pamięci GPU. Przeliczone wartości SH (36 floatów) przesyłane są w każdej klatce animacji. Ponieważ ilość danych jest mała, nie pogarsza się wydajności systemu względem rozwiązania polegającego na przechowywaniu informacji w postaci tekstury.

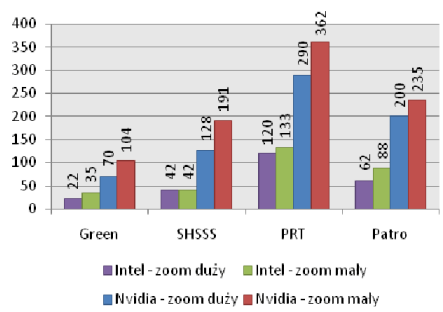
Najważniejsza część implementacji, odwrotna projekcja sferycznych harmonicznych, realizowana jest za pomocą programu vertex shadera odpowiedzialnego za wszystkie obliczenia geometryczne związane niezależnie z każdym z wierzchołków. W tym miejscu wyznaczana jest przybliżona grubość obiektu w kierunku światła. Operacja polega na zsumowaniu odpowiednich iloczynów (równanie 1):

```
// Suma iloczynow dla wstegi 1
   w1 += v1.x*c1.x;

// Suma iloczynow dla wstegi 2
   w2 += v1.y*c1.y + v1.z*c1.z + v1.w*c1.w;

(...)

// Suma iloczynow dla wstegi 6
   w6 += v7.y*c7.y + v7.z*c7.z + v7.w*c7.w + v8.x*c8.x
       + v8.y*c8.y + v8.z*c8.z + v8.w*c8.w + v9.x*c9.x
       + v9.y*c9.y + v9.z*c9.z + v9.w*c9.w;

   sh_dist = w1 + w2 + w3 + w4 + w5 + w6
```

Gdzie *w* oznacza kolejne wstęgi, *v* to 4-elementowe wektory kolejnych wartości funkcji SH dla kolejnych pasm, *c* wektory współczynniki SH, *sh_dist* wyjściowa wartość grubości obiektu w kierunku światła. Wartość *sh_dist* przekazywana jest do programu fragment shadera (zmienna varying). Następnie jest interpolowana liniowo dla odpowiednich pikseli względem trzech wierzchołków tworzących trójkąt (którego częścią w obrazie rastrowym jest dany piksel).

5. Rezultaty

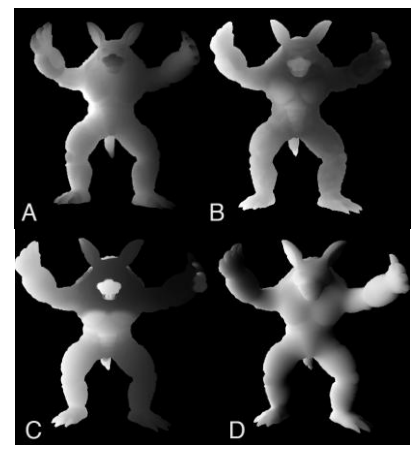
Oprogramowanie przetestowano na 2 platformach: komputerze PC wyposażonym w kartę z procesorem graficznym Intel GMA X3100 128+256MB, procesor Intel Core 2 Duo T7250 2GHz, 2GB pamięci RAM oraz komputerze PC z procesorem graficznym NVIDIA GeForce 8400 GS 512MB oraz procesorem CPU: 2 x Intel Pentium 4 2.8GHz, system operacyjny Windows 7.



Rys. 4. Wydajność algorytmu zmierzona w FPS dla dwóch różnych kart graficznych przy różnych ustawieniach kamery. Przedstawione wyniki otrzymano dla modelu świeczki (179112 wierzchołków)
Fig. 4. The algorithm effectiveness in FPS measured for two different graphic cards and two different camera settings. The results are obtained for the model "candle" (179112 vertices)

W ramach przeprowadzonych testów zbadano wydajność algorytmów: prezentowanego: SHSSS, Patro, Green'a oraz PRT. Przed przeprowadzeniem eksperymentu, aby zapewnić miarodajne wyniki zwrócono uwagę na jakość implementacji algorytmów. Ponieważ nie wyszytkie dostępne implementacje wykorzystują indeksowanie wierzchołków, do badań przygotowano modele, w których każdy z trójkątów zdefiniowany jest osobnymi trzema wierzchołkami. Ze względu na sposób pracy kart graficznych potok można podzielić logicznie na dwa etapy: przetwarzanie geometrii i rasteryzację. W zależności od złożoności modelu graficznego większą lub mniejszą pracę wykonywał vertex shader. Dla wszystkich modeli i algorytmów ustalono rozmiar ramki 1280 x 800 pikseli. Wydajność zbadano również dla kadru 5x5.

Rysunek 4 przedstawia szybkość działania algorytmów. Uwzględnienie wpływu wydajności karty graficznej oraz rozmiaru renderowanego obiektu na ekranie pozwala na obiektywne porównanie wszystkich algorytmów. W zbadanej implementacji PRT zastosowano kompresję współczynników metodą CPCA, co znacznie poprawia jej wydajność [3]. Na rysunku 5 zaprezentowano graficzne wyniki działania algorytmów.



Rys. 5. Graficzny wynik działania algorytmów: (A) PRT, (B) SHSSS, (C) Green, (D) Patro
Fig. 5. Graphical output of the algorithms: (A) PRT, (B) SHSSS, (C) Green, (D) Patro

6. Podsumowanie

W artykule zaprezentowano algorytm rozpraszania podpowierzchniowego światła z wykorzystaniem funkcji sherycznych harmonicznych. Przeprowadzone testy dowiodły effektywności pod kątem wydajnościowym algorytmu i jego implementacji. Algorytm porównano z innymi technikami generowania efektu uzyskując nawet dwukrotne przyspieszenie, przy porównywalnej jakości uzyskiwanego efektu (PRT). Dodatkowo zauważono, że do wyrenderowania bardzo dobrej jakości efektu przy świetle znajdującym się z przodu wystarczą tylko 4 współczynniki na jeden wierzchołek. Umożliwia to szybkie działanie algorytmu oraz jego wdrożenie do rzeczywistych projektów gier komputerowych.

Praca została wykonana w ramach realizacji grantu MNiSW nr N516 193537 (Polska).

7. Literatura

[1] Green Simon. GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics, chapter Real-Time Approximations to Subsurface Scattering. Addison-Wesley, 2004.
[2] Sloan Peter-Pike, Hall Jesse, Hart John and Snyder John: Clustered principal components for precomputed radiance transfer, 2003.
[3] Sloan Peter-Pike, Kautz Jan and Snyder John: Precomputed radiance transfer for real-time rendering, 2002.
[4] Patro Robert: Real-time approximate subsurface scattering, 2007.
[5] Green Robin: Spherical harmonic lighting. The gritty details, 2003.

otrzymano / received: 13.05.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*