Uzupełnienia do książki

Jacek Matulewski, Tomasz Dziubak, Marcin Sylwestrzak, Radosław Płoszajczak
Grafika. Fizyka. Metody numeryczne nowość
Symulacje fizyczne z wizualizacją 3D
Wydawnictwo Naukowe PWN, Warszawa 2010

Strona 124:

Podstawowe prawo optyki sformułowane przez holendra Willebrorda Snella van Royena na początku XVII wieku mówi, że światło padające na płaszczyznę odbija się od niej pod tym samym kątem. Dzięki temu droga pokonana przez promień światła jest najkrótsza. W praktyce prawo to ma jednak ograniczone zastosowanie – dobrze opisuje na przykład odbicie światła od lustra, dobrze wyszlifowanego metalu lub powierzchni pokrytej lakierem. W pozostałych przypadkach powierzchnia, od której odbija się promień nie jest gładka (w skali mikroskopowej), a tym samym bieg obitego promienia zależy w sposób dość przypadkowy od miejsca, od którego promień się odbił i w efekcie światło ulega rozproszeniu. Jeżeli założymy, że światło rozpraszane jest w sposób równomierny we wszystkich kierunkach, to jasność przedmiotu nie zależy od położenia obserwatora, a jedynie od orientacji płaszczyzny względem źródła światła. Określa to prawo sformułowane w drugiej połowie XVIII wieku przez szwajcara Johanna Heinricha Lamberta (o tym nieco niżej).
W praktyce żaden obiekt nie jest ani doskonale matowy ani doskonale błyszczący, dlatego w swoim modelu oświetlenia Phong uwzględnił jednocześnie oba komponenty światła odbitego, przy czym światło odbite zgodnie z prawem Snella zazwyczaj zachowuje swój kolor (jak w obraz widziany w lustrze), a światło rozproszone uwzględnia zjawisko absorpcji, co możemy kontrolować za pomocą współczynników odbicia ustalanych osobno dla poszczególnych składowych koloru.
Światło rozproszone na płaszczyznach obiektu oświetla inne przedmioty w pobliżu. Wyobraźmy sobie pokój, w którym zasłonięte są wszystkie okna, poza jednym niewielkim przez które wpada dzienne światło. Promienie słoneczne mogą bezpośrednio oświetlić tylko fragment ściany naprzeciw okna, a mimo to w pokoju wcale nie jest ciemno. Z łatwością możemy dostrzec nawet te miejsca, które są niedostępne dla promieni biegnących bezpośrednio z odsłoniętego okna. Dzieje się tak dlatego, że światło rozproszone na ścianach i meblach obecnych w pokoju oświetla ten pokój (stanowią wtórne źródła światła). W dużym przybliżeniu można przyjąć, że dzięki wielokrotnym rozproszeniom to oświetlenie tła jest równomierne (o jednakowym natężeniu w całym pokoju) i izotropowe (niezależne od kierunku). Niestety próba odtworzenia tej sytuacji w komputerze, nawet przy ograniczeniu promieni świetlnych tylko do tych, które docierają do kamery (por. ang. *ray tracing*) powoduje, że renderowanie jednej klatki sceny trwa znacznie dłużej niż jej wyświetlenie w grafice czasu rzeczywistego. Może trwać nawet kilka godzin. Rozwiązanie to stosuje się więc w animacji komputerowej (w filmach animowanych), a nie w grach. W zamian w modelu Phonga efekt ten imitować ma omawiane już światło otoczenia, które jednorodnie i izotropowo oświetla całą scenę. Skutkiem ubocznym pominięcia wtórnych źródeł światła w modelu Phonga jest konieczność osobnego generowania cieniów rzucanych.

Strona 157:

Umówmy się, że tekstura złożona z 16 pikseli to nie jest poważne potraktowanie sprawy, ale chciałbym na jej przykładzie wskazać pewien ogólny problem wynikający z konieczności pogodzenia dwóch światów: wektorowego modelu sześcianu i rastrowego obrazu, którym chcemy go obkleić. **W tym celu koniecznych jest kilka słów o tym, w jaki sposób karta graficzna umieszcza tekstury na trójkątach i czworokątach. Wbrew ciągle używanym przeze mnie określeniom „nałożenia tekstury”**

lub wręcz „obklejenia powierzchni teksturą", teksturowanie nie ma nic wspólnego z manipulacją całymi obrazami. Na pewno nie jest tak, że karta graficzna zmniejsza i pochyla obraz, aby dopasować go do kształtu ściany widocznej na ekranie. W zamian stosuje metodę wydajniejszą i przede wszystkim lepiej dopasowaną do potoku przekształceń w karcie graficznej. Każdemu werteksowi, poza położeniem, kolorem i wektorem normalnym można przyporządkować także współrzędne teksturowania (*s*,*t*). Więcej powiem o nich w następnym podrozdziale, a na razie przyjmijmy, że na przykładowym czworokącie obie współrzędne są równe zero w lewym górnym rogu i rosną aż do jedności; pierwsza w miarę przesuwania w prawo, druga – w dół.
W przypadku czworokąta do karty graficznej wysyłane są zatem cztery werteksy razem z współrzędnymi teksturowania. Werteksy przechodzą przez jednostkę werteksów, a następnie wszystkie jej własności (położenie, kolor, normalna i współrzędne teksturowania) są interpolowane tak, aby obliczyć ich pośrednie wartości dla wszystkich pikseli składających się na obraz owego czworokąta na ekranie. Przy czym czworokąt może być obrócony, pochylony lub przeskalowany – z punktu widzenia przeprowadzenia interpolacji, która odbywa się we wszystkich trzech zmiennych położenia to nie stanowi większego problemu. Następnie w jednostce zajmującej się już poszczególnymi pikselami (tu interpolowane współrzędne teksturowania mają ju z wartości ułamkowe) obliczany jest ostateczny kolor każdego piksela, jaki zobaczymy na ekranie (tj. uwzględniający na przykład oświetlenie). W przypadku nakładania tekstur kolor ten obliczany jest przez interpolację koloru tekseli z pobliża miejsca, na które wskazuje w teksturze zinterpolowana współrzędna teksturowania. Kwestię kształtu ostatecznego obrazu rozwiązuje zatem interpolacja współrzędnych teksturowania, kwestię koloru piksela – interpolacja koloru tekseli dla konkretnej wartości współrzędnych teksturowania tego piksela. Jeżeli obszar na ekranie odpowiadający teksturze jest mniejszy od rozmiaru tekstury, to kolor piksela jest wynikiem uśrednienia kilku tekseli. W przeciwnym przypadku konieczna jest interpolacja koloru. Metoda wykorzystana do przeprowadzenia tego uśrednienia lub interpolacji znacząco wpływa na efekt końcowy, nie tylko ze względu na jakość uzyskanego obrazu, ale również szybkość jego tworzenia. W naszym przykładzie obraz budowany jest w taki sposób, że teksele, które muszą być wstawiane między te odczytane z tablicy `tekstura`, **mają kolor równy** najbliższemu tekselowi **z tablicy**. Nie jest przeprowadzana żadna interpolacja. To szybki sposób, który sprawdza się, gdy chcemy, jak w naszym przypadku, utrzymać geometryczny wzór tekstury (rysunek 6.1, lewy). Odpowiadają za niego dwie funkcje z listingu 6.2, a mianowicie:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

OpenGL oferuje także interpolację liniową (stałą `GL_NEAREST` powinna zastąpić **wówczas** stała `GL_LINEAR`), w której każdy uzupełniany teksel jest obliczany z proporcji odległości między najbliższymi zadanymi przez tablicę `tekstura` (rysunek 6.1, prawy).

Strona 583:

W powyższych rozważaniach pominąłem przypadki szczególne. Ich przykładem jest takie zderzenie prostopadłościanów, w którym uderzane boki są równoległe do siebie. Wówczas żadna z dwóch opisanych metod nie zwróci prawidłowego punktu zderzenia.