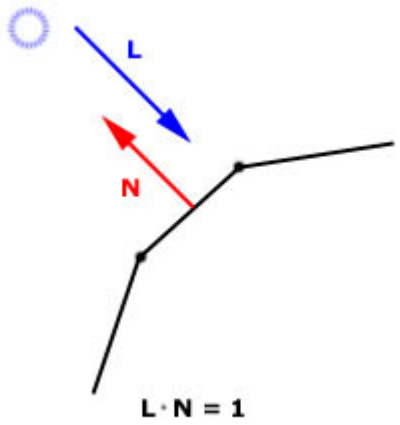
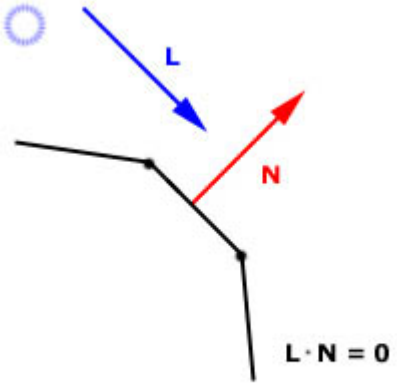
Od samego początku komputerowa grafika 3D czasu rzeczywistego starała i stara się nadal naśladować rzeczywistość. Rosnące możliwości obliczeniowe sprzętu pozwalają implementować coraz bardziej skomplikowane algorytmy, które dają w końcowym efekcie coraz bardziej realny obraz na ekranach naszych komputerów. Pojawienie się shaderów w wiecie grafiki 3D czasu rzeczywistego zupełnie zmieniło podejście do tworzenia efektów i dało do ręki programistom nowe, wspaniałe możliwości, których oczywiście nie omieszkali wypróbować. I paradoksalnie - mając nowe możliwości przystąpiono do "wydziwiania" i uzyskiwania efektów nie mających nic wspólnego z rzeczywistością.
Jednym z takich dziwactw okazała się tzw. technika cieniowania kreskówkowego. Wielu z nas na pewno w przeszłości a niemała część i dzisiaj czyta komiksy - rysowane historyjki często gęsto ilustrowane tekstem w tzw. chmurkach. Przygody Thorgala, Kajko i Kokosza, Funky Kovala czy innych bohaterów są na pewno większości znane. Bajki rysunkowe też na pewno każdy widział i wie jak wyglądają - w końcu każdy był kiedyś dzieckiem ;). To, co charakteryzuje wymienione przeze mnie rodzaje rozrywki to na pewno wygląd tego, co oglądamy - wynikający z takiego a nie innego sposobu tworzenia ich wizualnej części. Kiedyś ręcznie, dzisiaj wspomagane w większości wypadków komputerem rysowanie postaci i otaczającego ich świata charakterystycznym sposobem - przedmioty otoczone ciemną kreską, tzw. konturem i wypełnione odpowiednie fragmenty odpowiednim kolorem lub kilkoma, dającymi wrażenie przestrzenności. Im lepszy rysownik tym lepiej potrafi oddać dany przedmiot lub postać i za pomocą w sumie niewielu kolorów i ich odcieni przedstawić jego głębię, o którą jak wiemy na ekranie telewizora czy kartce papieru na pewno niełatwo. Sztuka ta, podobnie jak sposób rysowania na monitorze komputera w 3D nazwana została cieniowaniem, choć może trochę te pojęcia różnią się między sobą. Płaską postać na kartce odpowiednio malujemy i w odpowiednich miejscach po prostu domalowujemy jej cienie, ciemniejsze lub jaśniejsze, aby uzyskać odpowiedni efekt.

O Pytaniu czy my, w grafice 3D możemy uzyskać takie efekty w czasie rzeczywistym nie powinniśmy nawet wspominać, bo to po prostu wstyd. Zamiast tego powinniśmy już się zacząć zastanawiać, w jaki sposób to osiągnąć, bo dla nas nie ma "niemożliwe". Weźmy sobie raz na zawsze do serca, że grafika 3D w komputerze to coś, co jest ograniczone tylko naszą wyobraźnią ;). No ale zacznijmy może od początku. Wyobraźmy sobie, że robimy bardzo prostą animację z prostym oświetleniem i zwróćmy uwagę właśnie na nie. Doskonale wiemy jak liczy się jasność poszczególnych pikseli na ekranie - wystarczy iloczyn skalarny wektora padającego światła i normalnej do powierzchni. Wiemy po przykładach, że możliwe teraz jest już robienie tego zarówno dla normalnych wierzchołków (oświetlenie per vertex) jak i dla poszczególnych pikseli (per piksel). Iloczyn ten może przyjmować dosyć charakterystyczne wartości - jeśli powierzchnia jest prostopadła do wektora padającego światła a jej normalna jest skierowana w stronę światła to natężenie wyliczonego światła na tej powierzchni będzie maksymalne, równe 1 (wartość bezwzględna iloczynu skalarnego pomiędzy wektorami o kącie 0 pomiędzy nimi wynosi właśnie 1):

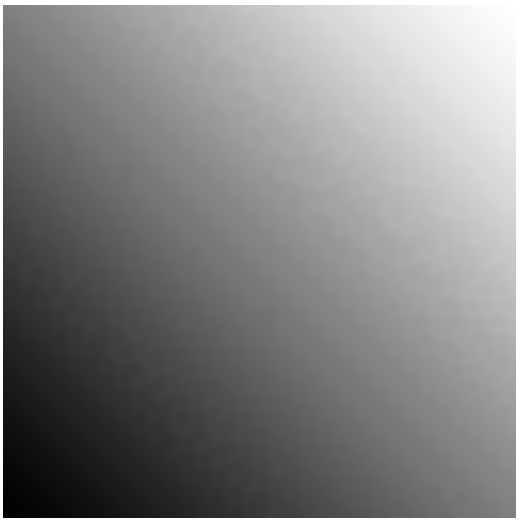


Jeśli powierzchnia będzie równoległa do padającego światła to jej normalna będzie prostopadła i iloczyn pomiędzy wektorami będzie wynosił 0 (światła nie będzie). Poniższe rysunki powinny wyjaśnić sprawę:



Tak działa oświetlenie w każdym API grafiki 3D. Wyliczane wartości oświetlenia są aproksymowane po całej bryle i na ekranie otrzymujemy ładne cieniowanie gładkie (Gourauda), albo płaskie. Ale zastanówmy się nad pewną kwestią - gdybyśmy tak nie mieli możliwości oświetlania pikseli bezpośrednio po wyliczeniach iloczynu skalarnego - tzn. nie mielibyśmy możliwości pokolorowania pikseli bezpośrednio przez urządzenie, to czy stoimy na straconej pozycji? Z podstawowych argumentów w walce o lepszy efekt pozostają nam... no właśnie - tekstury! Zakładamy więc, co jest bardzo ważne dla naszego artykułu, że tekstury możemy na bryłę nakładać. Pytanie więc, czy jesteśmy w stanie jakoś zastąpić cieniowanie przez nakładanie tekstury? Spójrzmy na wartości wyliczane przez iloczyn skalarny - dostajemy je w zakresie od 0 do 1 a to bardzo przypomina w najogólniejszym przypadku współrzędne mapowania tekstur. Jeśliby więc stworzyć taką teksturę, która zawierałaby odpowiednie wartości natężenia oświetlenia i wyznaczać z niej piksele na podstawie wyniku iloczynu wektorowego to można by kolorować obiekty tak samo, jak to robi urządzenie standardowymi metodami. Jak będzie wyglądała taka tekstura nie trudno się domyśleć z jednym małym szczegółem. Otóż wynik iloczynu skalarnego jest jedną liczbą a współrzędne tekstur to co najmniej dwie dane. Ale na to jest rada a w zasadzie dwie:

1. Pierwszym sposobem będzie stworzenie tekstury 2D, która będzie zawierać i we współrzędnych x i y odpowiednie wartości oświetlenia a potem poprzez shader wybierzemy odpowiednie (takie same) wartości x i y aby z tekstury wybrać odpowiedni teksel. Poniższy rysunek przedstawia możliwy wygląd takiej tekstury:

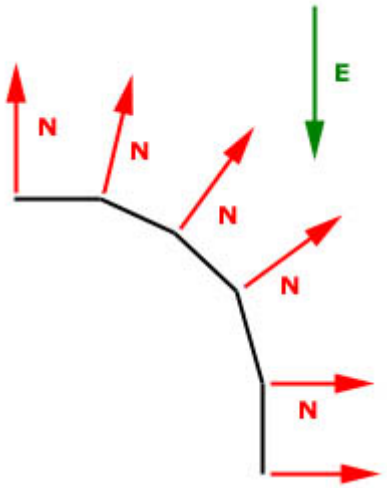


2. Tekstura nadal będzie 2D, z tym, że natężenie koloru będzie się zmieniać tylko wzdłuż jednej osi a druga będzie stała, tzn. nie będą się zmieniać wartości kolorów. W shaderze jedna ze zmiennych będzie się zmieniać wraz z iloczynem skalarnym a druga pozostanie stała, bo i tak nie będzie mieć znaczenia. No i poniżej rysunek poglądowy:

Który sposób wybierzemy to już nasza sprawa. Wydawać by się mogło, że ten drugi jest wygodniejszy: łatwiej będzie dobierać współrzędne (wystarczy w zasadzie jedną), tekstura będzie mniejsza. Jak widać, współrzędne tekstury wyliczone jako iloczyn skalarny normalnej i wektora światła pokrywają się idealnie z naszymi wymaganiami. Czyli jeśli iloczyn skalarny będzie bliski 0 to tekstura będzie ciemniejsza (niższa wartość współrzędnej mapowania) a zatem ciemniejszy będzie obiekt, w miejscu, w którym ten fragment tekstury zostanie nałożony. Od razu także widać jak wspaniałe możliwości

dostajemy do ręki mogąc cieniować obiekt za pomocą tekstur. Wystarczy zmienić teraz teksturę cieniującą w jakikolwiek sposób i uzyskamy arcyciekawe efekty na ekranie w dziedzinie oświetlenia.
Pozostaje pytanie, co to ma wspólnego z cieniowaniem znanym z kreskówek czy komiksów. Rysownik malując przedmioty lub postacie na pewno nie jest tak precyzyjny jak komputer wyliczając cieniowanie na podstawie normalnych. Przeważnie więc zadowala się tym, że aby oddać przestrzenność obiektu nakłada dwie lub trzy wartości głębi cienia na pokolorowany obiekt. Daje to właśnie charakterystyczny efekt kreskówkowy który znany jest od bardzo dawna. Mają do dyspozycji cieniowanie za pomocą tekstur możemy łatwo ten efekt osiągnąć idąc tropem myślenia rysownika. Skoro jemu wystarczy raptem kilka poziomów cieniowania to czemu nie nam? Wystarczy aby tekstura cieniująca nie była płynna ale aby miała na przykład trzy lub cztery poziomy koloru - resztę załatwi za nas shader cieniujący i mechanizmy teksturowania. Taka tekstura może na przykład wyglądać tak:

Efekt będzie oczywisty - po nałożeniu takiej tekstury cieniującej na obiekt na podstawie iloczynu skalarnego na pewno nie uzyskamy płynnego cieniowania Gourauda, ale na przykład piękną, kreskówkową postać tylko z kilkoma poziomami cienia!
Do rozwiązania pozostanie jeszcze jedna kwestia. Otóż wspomniałem na początku, przy opisie rysowania postaci, że rysownik najpierw rysuje ciemny, przeważnie czarny kontur postaci a następnie wypełnia go odpowiednim cieniowaniem. My zaczęliśmy trochę od drugiej strony, bo najpierw obliczyliśmy efekt wypełnienia konturu a teraz zajmiemy się nim samym. Ponieważ rysownik przeważnie zaczyna od tego elementu, więc jasne wydaje się być to, że nie jest on związany z oświetleniem a definiuje jedynie wygląd ogólny obiektu. W naszym przypadku w obliczeniach nie będziemy więc brali pod uwagę wektora światła. Skoro kontur będzie określał widok obiektu, więc na pewno będzie on zależał od położenia obserwatora. Aby uzmysłowić problem na pewno przyda nam się rysunek:



Wektor E to wektor oka skierowany od oka w kierunku wierzchołków obiektu a wektory N to oczywiście normalne wierzchołków obiektu. A jak narysujemy kontur to już chyba oczywiste - wystarczy powrócić do metody cieniowania za pomocą tekstury i wszystko powinno być już jasne, choć pewnego omówienia wymagać na pewno będzie tekstura, która posłuży do narysowania konturu. Będziemy oczywiście wyliczać iloczyn skalarny wyżej pokazanych wektorów i rezultatu tego obliczenia używać będziemy do samplowania tekstury. Ponieważ kontur jest widoczny tylko na krawędziach obiektu i jest on przeważnie ciemny (czarny) a reszta pozostaje niezmieniona więc tekstura, z której będziemy korzystać przy rysowaniu konturu będzie dwukolorowa (no chyba, że wykombinujemy jaki ciekawy efekt związany z konturem). Dodatkowo, ponieważ chcemy kolorować tylko kontury (zaciemniać je) więc tylko dla bardzo niewielkich wartości iloczynu skalarnego (a co za tym idzie i współrzędnych mapowania) tekstura powinna być ciemna (lub w kolorze konturu). Aby osiągnąć tutaj dobry efekt należy odpowiednio przygotować taką teksturę. Jak widać z rysunku poniżej, dla widocznych krawędzi iloczyn skalarny będzie bliski zera (odpowiednie wektory są prostopadłe), więc na teksturze we współrzędnych bliskich zera powinien być kolor ciemny a reszta biała (aby nie zmieniać zawartości wnętrza konturu). W zależności od tego, jak mocny chcemy mieć kontur (jak szeroki), w okolicach mniejszych wartości mapowania po prostu zaciemniamy mniej lub więcej tekstury. Przykładowa tekstura służąca do narysowania konturu może wyglądać następująco:

Jak widać tylko niewielki element tekstury jest zaczerniony, aby mógł skutecznie oznaczać krawędź obiektu. Pewien problem, na który trzeba koniecznie tutaj zwrócić uwagę może wystąpić przy używaniu mipmap dla tekstury konturu, ponieważ przy filtrowaniu i niższych poziomach mipmapy mogą się zlewać piksele i wystąpią zakłócenia w kształcie konturu. Należy więc zawczasu przygotować ręcznie wszystkie poziomy mipmap i wczytać je ręcznie dla tekstury konturu, aby osiągnąć dobre efekty. Trzeba to niestety robić trochę na wyczucie i kontrolować, jak wyglądać będzie obiekt bo uniwersalnej metody na to nie ma a gdybyśmy tak dla przykładu zastosowali tylko taką teksturę jak przedstawiłem powyżej

to kontur może i będzie, ale niezbyt widowiskowy. Trzeba więc tutaj trochę poeksperymentować.
No i to w zasadzie wszystko jeśli chodzi o cieniowanie kreskówkowe w grafice czasu rzeczywistego. Teraz jak komu się nudzi może przerobić jaką znaną gierkę na mapowanie znane z komiksu - na przykład starego Quake-a - musiałby naprawdę ciekawie wyglądać. Możliwości jak widać przy tym cieniowaniu jest wiele i można stosować różne warianty - cieniowanie wraz z konturami, same kontury czy samo cieniowanie. Polecam także zainteresowanie się bliżej wykorzystaniem tekstur przy cieniowaniu obiektów, bo ta technika daje naprawdę wielkie możliwości i całą gamę efektów można stworzyć. Mamy przecież do dyspozycji tekstury z przeźroczystocią na przykład, które można by wykorzystać w ciekawy sposób, jest do dyspozycji Pixel Shader, który daje wręcz nieograniczone możliwości jeśli chodzi o manipulację tekstur.