XVIII. Budowanie pokazu świetlnego

Najnowszy postęp technologiczny w grach został dokonany na graficznym końcu rzeczy. Początkowa architektura komputera zaczynała się od zwykłego procesora. Później jednostka FPU (jednostka zmiennoprzecinkowa) pomogła przy przyspieszaniu opóźnionych funkcji zmiennoprzecinkowych. Niedługo potem wymyślono koncepcję posiadania odpowiednika dwóch procesorów w jednym. (To nie są dwa pełne procesory działające równolegle, ale pojedynczy procesor z wbudowanym innym częściowym procesorem, umożliwiający równoległe uruchamianie częściowego zestawu instrukcji Układy Pentium rozpoczęły ten trend, który od tego czasu został odłożony na bok. korzyść z pamięci podręcznej wykonania po wykonaniu zamówienia.) Ostatnim etapem najnowszego procesora jest GPU, w pełni programowalna jednostka przetwarzania graficznego. To marzenie tylu programistów od początku czasu (przynajmniej wynalezienie komputera). Oznacza to kilka rzeczy dla dzisiejszych programistów gier. Oznacza to, że musisz zrównoważyć obciążenie między procesorem i GPU. Procesor graficzny jest wystarczająco potężny, aby ci umożliwić opisać algorytm, który ustawia kolor piksela i / lub położenie tego wierzchołka. Jest to potężne, ponieważ możesz zrobić wiele fantazyjnych rzeczy (produkty dot, produkt krzyżowy) naturalnie i przyspieszone przez GPU. Sam fakt wykonania tego nie jest celem tej książki. Powinieneś rzucić okiem na każdą książkę obejmującą shadery, aby uzyskać więcej informacji na ten temat. Chociaż jest to naprawdę fajna rzecz, jest również bardzo aktualna, więc nie ma żadnych bibliotek pomocniczych, które pomogłyby w implementacji fantazyjnych rzeczy, które można dziś zobaczyć w grach. Aby w pełni wykorzystać zalety procesora graficznego, ważne jest, aby zrozumieć wewnętrzne działanie światła i jego interakcję z otoczeniem. Bez tej wiedzy trudno będzie zdecydować, jaki kolor powinien mieć dany piksel. W szczególności ta część obejmuje:

■Potokowanie rasteryzacji

■ Oświetlenie modelu śledzenia promieni

■ Źródła światła

■ Modele oświetlenia

Badanie potokowania rasteryzacji

W części 13 przyjrzeliśmy się dokładnie fragmentowi potokowaniu 3D, ale w rzeczywistości nie patrzyliśmy na etap teksturowania / rasteryzacji. Ten etap potoku jest odpowiedzialny za próbkowanie tekseli (odpowiednik piksela dla tekstury), zastosowanie zestawu operacji i nałożenie go na ekran. To może wydawać się bardzo prostym zadaniem, ale jest o wiele bardziej zaangażowane, niż początkowo sądzicie. Wybór koloru pojedynczej tekstury, nawet jeśli myślisz tylko o prostej fakturze bez jakiejkolwiek formy światła, nadal wymaga to trochę teorii próbkowania, aby uzyskać ładną fakturę. Teoria próbkowania jest gałęzią matematyki, która analizuje wartości próbkowania (w tym przypadku kolory) ze źródła danych. Ważne jest zrozumienie tego, aby zrozumieć techniki używane w dzisiejszym sprzęcie, a także dać poczucie, jak powolna może być jedna metoda w porównaniu z inną. Na szczęście wiele materiałów tutaj bardzo dobrze pasuje do informacji zawartych w części 13, ponieważ rozdział ten dotyczył także próbkowania. Opisano funkcję, którą można wykorzystać do celów pobierania próbek. Różnica polega na tym, że interesuje Cię tylko konkretna próbka, a nie cała funkcja. Niemniej jednak, kilka następnych rozdziałów powinno uświadomić Ci techniki, które są już wdrożone w większości interfejsów API 3D.

Pobieranie tekstury

Jest to jednostka odpowiedzialna za pobieranie koloru piksela z tekstury. Może to początkowo wydawać się łatwym zadaniem, ale ma swoje zastrzeżenia. Jak określono w większości interfejsów API

3D, dane tekstur przekazane do silnika 3D zawierają współrzędne tekstury. Współrzędne te mogą być 3D, 2D lub 1D. Tekstura 3D, którą można zobaczyć jako opis koloru w 3D. Na przykład może to być przydatne, jeśli chcesz narysować kamień i możesz odłamać jego fragmenty. Możesz utworzyć teksturę 3D opisującą kolor skały w <x, y, z>. Po usunięciu fragmentów skały tekstura może odwzorować wartości bezpośrednio z współrzędnej obiektu w przestrzeni obiektu na współrzędne tekstury. Z drugiej strony tekstura 1D może być teksturą zależną od czegoś takiego jak zakrzywienie obiektu. Jeśli chcesz mieć różne stopnie oświetlenia w zależności od kąta, możesz użyć tekstury 1D, w której wartości tekstury [0, 1] doprowadzą Cię do bardzo podświetlonego koloru tekstury aż do ciemnej czerni, tak jak w przypadku celowania w części 17,

Mipmapping

Mipmapping to sposób na poprawienie jakości próbek tekstur odległych od przeglądarki. Jeśli zajmiesz się szmatką i odsuniesz ją bardzo daleko od widza, pojedynczy piksel na ekranie może reprezentować duży fragment tekstury. Aby być matematycznie poprawnym, zmapuj piksel na teksturę. Innymi słowy, przekształcasz przestrzeń zdefiniowaną przez pojedynczy piksel w przestrzeń tekstur, a ty średniąsz każdy piksel zawarty w tym kształcie. W konsekwencji, jeśli pojedynczy piksel reprezentuje całą teksturę, kolor tego piksela jest idealnie równy średniej każdego teksela tekstury. Jest to niezwykle kosztowne zadanie w czasie wykonywania. W związku z tym wprowadzono pomysł wstępnego obliczania zbioru uśrednionych tekstur. Większość implementacji wybiera generowanie nowej mipmapy, jeśli jeden piksel zawiera więcej niż jeden lub dwa teksle. Innymi słowy, jeśli pominiesz jeden texel dla każdego wybranego teksela, powinieneś wygenerować nową mipmapę. W konsekwencji tekstura o rozdzielczości 512x512 prawdopodobnie będzie zawierała wygenerowany dla niej zestaw tekstur (256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, 1x1). Ten proces nie odbywa się w locie; zamiast tego zwykle robi się to tuż po procesie ładowania tekstury GPU. Liczba generowanych poziomów mipmap zależy oczywiście od implementacji sprzętowej, a wybór mipmap jest bardzo prosty i odbywa się w locie. Aby to zrobić, wystarczy spojrzeć na liczbę tekseli / pikseli. Ponieważ generowanie tekstury zmienia się, gdy stosunek wynosi dwa teksy / piksele, staje się kwestią obliczania stosunku w jednym kierunku. Liczba całkowita tego współczynnika wskaże, którą mipmapę wybrać.

Mapowanie afiniczne tekstury

Istnieją dwa ogólnie akceptowane sposoby zastosowania tekstury do trójkąta. Pierwsza to najbardziej oczywista metoda, którą zwykle określa się jako zwykłe stare mapowanie tekstur. To po prostu przyjmuje wartość <x, y> tekstury dla wierzchołków 2D <x, y> trójkąta i interpoluje próbkowanie liniowo z jednego punktu do drugiego. Daje to dwie współrzędne tekstury na krawędzi trójkąta. Kiedy już to zrobisz, po prostu wypełnisz trójkąt, interpolując liniowo po raz kolejny od prawej współrzędnej tekstury do lewej współrzędnej tekstury. Próbkowanie dla linii jest interpolowane w przyrostach tak, że jeden przyrost zwiększa się o jeden piksel w dla wyrenderowanego trójkąta. Ten proces jest nazywany konwersją skanowania, ponieważ przekształca kształt w zestaw linii skanowania. Innymi słowy, silnik interpretujący oblicza, ile t (dla równania liniowego) wzrasta, gdy y wzrasta o jeden (na ekranie). Wartość t jest następnie zmniejszana do długości próbkowanej tekstury, a następnie pobierana jest próbka. Tak więc, biorąc pod uwagę pozycję na ekranie <x, y>, rasteryzer odwzorowuje tę wartość na skojarzoną współrzędną tekstury <u, v> przez dwie równoczesne interpolacje w x i interpolację między tymi dwiema interpolacjami w y.

Prawidłowe odwzorowanie tekstury w perspektywie

Ale z twojej wiedzy na temat bufora z, wiesz, że to nie jest poprawny sposób radzenia sobie z problemem. Głębokość nie jest liniowa, więc niewłaściwe jest interpolowanie liniowe od jednej wartości koloru do drugiej. Prawidłową funkcją interpolacji jest 1 / z z powodu projekcji. Aby poprawnie

interpolować z jednego texela do innego, musisz interpolować współrzędną texela <u, v> / z liniowo, a także 1 / z liniowo. Gdy chcesz uzyskać współrzędne tekstury w danym momencie, po prostu weź współrzędną texel <u, v> / z i podziel ją przez 1 / z. Może to brzmieć tak, jakby dwa zy anulowały się nawzajem, ale tak nie jest, ponieważ 1 / z jest interpolowane liniowo. Twoja funkcja wygląda następująco:

```
su = U / Z;

sv = V / Z;

sz = 1 / Z;

for (x = startx; x <= endx; x ++) {

u = su / sz;

v = sv / sz;

PutPixel (x, y, tekstura [v] [u]);

su + = deltasu;

sv + = deltasv;

sz + = deltasz;

}
```

To oczywiście nie jest skuteczny sposób radzenia sobie z tym, ale jeśli grasz z równaniem wystarczy, możesz uczynić pętlę rdzeniową tego renderera mapowaniem liniowym.

Najbliższa metoda pobierania próbek

Ponieważ wartości <u, v> są wartościami zmiennoprzecinkowymi, należy je przekonwertować na liczby całkowite, ponieważ tekstura jest wykonana z dyskretnych próbek. Obcinanie współrzędnych tekstury nazywa się najbliższą metodą próbkowania, ponieważ wybiera najbliższą próbkę teksel. Alternatywą jest użycie dodatkowej precyzji, którą posiadasz. Floats dostarcza informacji subpikselowej o pozycji próbki dla tekstury. Wykorzystaj to na swoją korzyść, wykonując kolejną interpolację na poziomie kolorów.

Metoda próbkowania bilinearnego

Te rzeczy powinny być łatwe po przeczytaniu części 13 dotyczącego krzywych interpolacji. Gdy współrzędna tekstury jest liczbą całkowitą, kolor piksela w tej lokalizacji określa kolor całej tej lokalizacji ze 100-procentową dokładnością. Jeśli przesuniesz się w połowie odległości pomiędzy tym pikselem a następnym, jest to odpowiednik połowy masy pierwszego piksela i połowy ciężaru drugiego piksela. Wszystko, co musisz zrobić, to zastosować liniową interpolację koloru z jednego piksela do następnego piksela. Ta interpolacja powinna być mantysą liczby, ponieważ ustaliłeś, że w położeniu całkowitym piksel w pełni opisuje wartość. Mantysa ma wygodny zakres [0, 1], który doskonale nadaje się do interpolacji. Jeśli zastosujesz ten sam pomysł co mapowanie tekstury (interpolując w x dla dwóch wartości, piksel i piksel poniżej) i interpolując te dwie wartości w y, osiągniesz to, co jest powszechnie znane jako interpolacja dwuliniowa. Jeśli masz współrzędne tekstury <u, v>, powinieneś wybrać piksel w miejscu obciętych współrzędnych. Niech reszta tej współrzędnej zmiennoprzecinkowej będzie <s, t>. Obliczanie wartości koloru wymaga jedynie interpolacja koloru, która jest podana za pomocą następującego wzoru:

$$Color1 = (u,v) + s\big((u+1,v) - (u,v)\big)$$
$$Color2 = (u,v+1) + s\big((u+1,v+1) - (u,v+1)\big)$$
$$Color = Color1 + t\big(Color2 - Color1\big)$$

Technika ta jest nazywana próbkowaniem dwuliniowym lub próbkowaniem liniowym. To tylko ważona średnia z czterech najbliższych teksli. W rzeczywistości kończy się tworzenie funkcji zdefiniowanej dla zakresu <[0, 1), [0, 1)>, który opisuje wartość koloru między czterema pikselami. Kształt tej interpolacji jest płaszczyzną, ponieważ zbudowany jest z dwóch liniowo niezależnych interpolacji (lub wektorów).

Metoda pobierania próbek trójliniowych

Innym sposobem próbkowania jest użycie techniki trójliniowej. To po prostu rozszerza próbkowanie dwuliniowe, patrząc na rzeczy z poziomu mipmap. Kiedy pracujesz z mipmapami, widzisz drastyczne różnice, gdy sprzęt zmienia się z jednego mipmap na drugi. Wielokąt zaczyna wyglądać szczegółowo i nagle wygląda na niewyraźny lub niewyraźny. Wynika to z mipmapping. Aby to naprawić, możesz interpolować między mipmapami, które są próbkami trójliniowymi. Aby to zrobić, najpierw przeprowadź dwuliniowe filtrowanie na dwóch najbliższych mipmapach, a następnie interpoluj liniowo między tymi dwiema próbkami. Jest to jedynie dodatkowy poziom interpolacji liniowej, gdzie parametr interpolacji jest podany przez resztę teksli / pikseli, a dwie najbliższe mipmapy są liczbami całkowitymi tej funkcji (podobnie jak to było zrobione z kolorami). Jest to ważona średnia z ośmiu najbliższych tekseli (po cztery z dwóch najbliższych mipmap). Jest to dobry sposób radzenia sobie z próbkowaniem mipmap, ale daje stosunkowo rozmyte wyniki pod małymi kątami.

Metoda próbkowania anizotropowego

Inny sposób pobierania próbek jest znany jako anizotropowy. To zabiera wszystko o krok dalej i bierze pod uwagę ten kąt. Wybierając mipmap, oblicza się stosunek texel / piksel w więcej niż jednym kierunku (na przykład x i y). Jeśli te dwa współczynniki nie pasują, występuje zniekształcenie, które jest wprowadzane przez filtrowanie trójliniowe, które zakłada liniowe obciążenia. Musisz pamiętać, że głębia sceny nie jest liniowa, jest odwrotnie liniowa (1 / z). Aby to naprawić, pobiera się dodatkowe próbki, gdy stosunek między kierunkami jest inny. W skrócie można go zobaczyć jako piksel wyświetlany w przestrzeni tekstury 3D, a zestaw tekseli uśrednia się za pomocą wag. Implementacje anizotropowego pobierania próbek są niestety dość drastycznie różne od jednego dostawcy do drugiego.

Multiteksturowanie

Ten specyficzny obszar jest dość złożony i zajmuje większość tego rozdziału. Jest to część, która łączy ze sobą różne tekstury i dodaje odpowiednie oświetlenie, mgłę itp. Stawia wiązkę tekstur w połączeniu z zestawem parametrów, aby uzyskać ostateczny kolor piksela. Kolor składa się z czerwonego, zielonego i niebieskie kanały, a także kanał alfa (lub przezroczystość). Ogólnie rzecz biorąc, kilka rzeczy opisuje, jak piksel będzie wyglądał: kolor trójkąta, poziom mgły dla każdego wierzchołka, zestaw tekstur używanych do określenia koloru i funkcja, która łączy wszystko razem. W większości implementacji sprzętowych jest to pojęcie określane jako multiteksturowanie. Istnieje wiele sposobów łączenia dwóch tekstur razem, a wybrana funkcja zależy od efektu, który chcesz osiągnąć. Istnieje mnóstwo efektów, które można osiągnąć, które są obsługiwane przez sprzęt, ale dwa z nich są szczególnie przydatne do twoich celów: modulacji i kalkomanii.

Modulacja

Modulacja to typowa kombinacja kolorów między dwoma tekselami. Powiększa kanały razem. W przypadku dwóch podanych kanałów kolorów, a, b, z zakresem [0, 1], wynik wyjściowy jest określony przez następujący wzór:

$$\mathbf{c} = \mathbf{c}_1 \cdot \mathbf{c}_2$$

Na przykład w Części 17, jeśli chciałbyś rzutować teksturę na inną teksturę, mógłbyś użyć funkcji modulacji do połączenia dwóch tekstur. To rzeczywiście rzutuje jedną teksturę na inną teksturę, tak jak rzutowane jest światło. Możesz wybrać poziom przezroczystości, dodając niższy komponent alfa do koloru drugiej tekstury.

Kalkomania

Kalkomania jest podobna do modulacji, ale pozwala zdefiniować połączenie obu tekstur z kanałem alfa jednej z dwóch tekstur. Jest matematycznie zdefiniowany jako interpolacja liniowa między kolorem pierwszej tekstury i drugiej tekstury, gdzie parametr interpolacji jest wartością alfa pierwszej tekstury:

$$\mathbf{c} = \left(1 - c_{1\alpha}\right)\mathbf{c}_2 + c_{1\alpha} \cdot \mathbf{c}_1$$

W pierwszym przypadku, jeśli chcesz wyemitować światło, aby rozjaśnić scenę, a środek jest biały, podczas gdy na zewnątrz jest czarny, scena będzie w rzeczywistości czarna dla czarnych pikseli. Wynika to z tego, że iloczyn 0 i innego koloru nadal daje 0, co jest czarne. To staje się problemem. Z drugiej strony, jeśli użyjesz kalkomanii, aby osiągnąć ten efekt, kolor pikseli nadal może pozostać czarny, ale powinny być całkowicie przezroczyste (to znaczy, że alfa w tym miejscu powinna wynosić 0).

Mgła / zamglenie

Ostatnim elementem, na który patrzę, jest współczynnik mgły. W rzeczywistości mgła jest cieczą / gazem, która unosi się w powietrzu. Powodem, dla którego mgła zasłania widok jest to, że zawieszone w powietrzu cząstki blokują przedostawanie się światła. Po wejściu w obszar mgły obiekty znajdujące się na odległość mogą zostać całkowicie zasłonięte przez mgłę. Grubość mgły jest funkcją odległości i gęstości. Im dalej patrzysz, tym więcej mgły znajduje się pomiędzy tobą a odległymi obiektami, a im bardziej zakryte są obiekty (mniej światła przechodzi). Ze względu na swoją naturę, mgła jest w rzeczywistości wiązką cząstek o wartości alfa, ale jest to zbyt wolne, aby można było wykonać realistyczny model. Zamiast tego sprzęt dokonuje uprzedzenia kolorystycznego na ostatecznym kolorze tuż przed renderowaniem.

Liniowa mgła

Mgła f może być traktowana jako liniowa funkcja odległości:

$$f = \frac{far - z}{far - near}$$

W tym przypadku wartości dalekie i bliskie niekoniecznie muszą odpowiadać płaszczyźnie dalekiego / zbliżonego do stożka ściętego. Jeśli wyliczysz licznik, zobaczysz, że ta funkcja jest w rzeczywistości funkcją liniową (funkcja linii), gdzie nachylenie wynosi -1 / (daleko - blisko). To nachylenie jest zdefiniowane jako gęstość mgły, a przypomnienie lub odchylenie jest początkową gęstością mgły. Zatem f jest używane jako czynnik mieszający (poziom przezroczystości). Kolor mgły odchyla obliczony

kolor (bez mgły). Współczynnik mg f jest konsekwentnie zaciśnięty w zakresie [0, 1], gdzie 0 oznacza, że nic nie jest widoczne, a 1 to przypadek, w którym nie stosuje się mgły.

Wykładnicza mgła

Jeśli myślisz o mgle w kategoriach nieskończenie małych cząstek zawieszonych w powietrzu, które są symulowane z wartością alfa, liniowy model mgły nie jest precyzyjny. Nieskończenie dokładny model polega na tym, że wiązka cząsteczek jest symulowana za pomocą samych pikseli, gdzie alfa decyduje, ile światła przechodzi przez każdą cząstkę. Jeśli zastanowisz się trochę nad tym modelem, na pewno będziesz miał w pewnym momencie kilka cząstek. Jeśli dwie cząstki zachodzą na siebie, operacja mieszania alfa jest stosowana w kolejności. Aby zachować prawidłową kolejność przezroczystości, normalnie musisz renderować od tyłu do przodu, a operacja mieszania alfa zostanie zastosowana dla koloru c i wartości przezroczystości α:

$$\mathbf{c} = (1-\alpha)\mathbf{c}_{dest} + \mathbf{c}_{src}\alpha$$

Jeśli zastosujesz rekurencyjnie równanie mieszania alfa, otrzymasz funkcję wykładniczą (zastosujesz funkcję do ostatniej funkcji). Wymień kolor docelowy kilka razy rekurencyjnie, a zobaczysz co mam na myśli. Jest on zgodny z definicją podaną w części 17. Z tego powodu model wykładniczy jest najlepszym modelem, jaki można zastosować. Jest to najszybszy model. Równanie definiujące wykładniczą mgiełkę definiuje się jako takie:

$$f = e^{-density \cdot z}$$

Ponownie pojawia się współczynnik mgły, który obejmuje [0, 1] i stosuje odchylenie od koloru mgły. Niektóre interfejsy API udostępniają także funkcje wykładnicze o kwadratowym rozkładzie. W takim przypadku wystarczy wyrównać wykładnik. Ten model jest zazwyczaj zbyt agresywny i bardzo szybko wprowadza mgłę.

Wolumetryczna mgła

Jeśli grałeś w gry takie jak Quake 3 i Unreal, zauważyłeś interesującą mgłę, którą zaimplementowali. Ta mgła różni się od mgły z poprzednich metod, ponieważ zakładały, że znajdujesz się w nieskończonej strefie mgły. Z drugiej strony mgła objętościowa bierze pod uwagę więcej właściwości mgły. Na przykład, możesz zauważyć, że mgła ma tendencję do pozostawania na poziomie gruntu. Dzieje się tak tylko dlatego, że gęstość mgły jest niższa niż gęstość powietrza. Podobnie jak w przypadku wody kontra drewniana skrzynia podana w części 8, mgła unosi się nad ziemią. Kiedy wchodzisz do pokoju w grze, nie jest konieczne, aby całe pomieszczenie było wypełnione mgłą. Możesz zamrozić tylko część pomieszczenia. W tym celu większość interfejsów API 3D pozwala kontrolować mgłę za pomocą wielu parametrów. Na przykład możesz określić odległość obiektu od objętości mgły, lub możesz określić współrzędne mgły względem punktu w przestrzeni, który jest środkiem mgły. Dzięki tym rozszerzeniom możesz tworzyć mgły tak, jak byś tego chciał. Jeśli chcesz być naprawdę ciekawy, możesz sprawić, że mgła wyleje się w powietrzu dzięki formule, która po prostu wygląda dobrze. Jeśli grasz z równaniami cosinusa i funkcji sinusoidalnych i tłumaczysz mgłę, możesz stworzyć kroplę mgły, która porusza się mniej więcej tak, jakbyś oczekiwał ruchu mgły. Możesz też zasymulować to za pomocą modelu sprężynowego, aby określić granicę mgły, jeśli używasz kropelki Jell-o utrzymywanej razem przez zestaw sprężyn. Musisz tylko stymulować blob poprzez wiatr wiejący na niego raz na jakiś czas. Spowoduje to efekt kropelki dymu. Im wolniej, tym lepiej dla mgły, ponieważ wygląda bardziej realistycznie. Ułatwienie jest to, aby powiedzieć GPU, jak daleki jest dany wierzchołek od źródła, podając współrzędną lub głębokość. Współrzędne są oczywiście łatwiejsze, ale wymagają większej

przepustowości na GPU. Zapal model Ray Tracing. Do tej pory patrzyłeś na światło w dość abstrakcyjny sposób. Głównie widziałeś to jako komponent, który moduluje kolor obiektów (np. Projekcja reflektorów). Światło samo w sobie jest zjawiskiem, które należy zbadać, ponieważ jest ono kluczem do cieniowania. Światło nie może być mieszane z pigmentem. W przypadku zabarwienia pigmentowego mieszanie wszystkich kolorów razem daje czarny kolor. Możesz wziąć kilka rzeczy w domu, połączyć je ze sobą, a jeśli wynik jest jednorodny, prawdopodobnie jest całkiem ciemno. Światło zachowuje się inaczej, a także ma inne podstawowe kolory. Jak już wspomniano, światło składa się z trzech kolorów rozpoznawanych przez stożki w ludzkim oku. Oko ma trzy kolorowe szyszki, które dostrzegają różnicę między czerwonymi / zielonymi i niebieskimi kolorami. Niektóre rzadkie osoby mają w rzeczywistości cztery czopki i mogą rozróżniać więcej kolorów. Osoby, które nie znają kolorów, nie mają stożka i nie mogą odróżnić niektórych kolorów. Jest tak, ponieważ każdy kolor ma inny poziom intensywność. Na przykład kolor czerwony jest znacznie bardziej intensywny niż niebieski. Jeśli wykonasz czarno-biały obraz całkowicie czerwonego obiektu i innego zdjęcia całkowicie niebieskiego obiektu, czerwony obiekt z tego powodu wydaje się ciemniejszy niż niebieski obiekt. W rzeczywistości, eksperymentalnie, przekształcenie koloru RGB na intensywność (skala szarości) odbywa się za pomocą następującej formuły:

Intensywność = 0,299 Czerwony + 0,587 Zielony + 0,114 Niebieski

W związku z tym, jeśli planujesz scenę czarno-białą, powinieneś wziąć ten fakt pod uwagę. Po prostu wykonanie przeciętnego komponentu koloru nie przyniesie naprawdę imponującego wyniku. W rzeczywistości będzie brakować informacji. Jeśli, na przykład, twój obraz miał kolor czerwony i niebieski, każdą pełną intensywność, kolory wyglądałyby tak samo z obliczeniami uśredniającymi, ale nie powinny być, ponieważ kolor niebieski jest mniej intensywny niż czerwony. A więc, co z samym światłem? Światło jest w rzeczywistości promieniowaniem elektromagnetycznym o długości fali, którą oko może uchwycić i kategoryzować. W zakresie od 4 000 do około 7 700 angstremów, oko dekoduje tę informację w postaci kolorów. Więc jeśli światło jest rzeczywiście promieniowaniem, pojawia się pytanie, jak faktycznie zachowuje się promieniowanie? Cóż, jest wiele dyskusji na temat tego, czy światło jest cząstką, czy częstotliwością, a ja zostawię spekulację fizykom. W twoim przypadku łatwiej jest brać pod uwagę światło jako cząstkę.

Ruch Światła

Jak wspomniano wcześniej, światło jest wychwytywane w trzech komponentach (czerwony, zielony, niebieski). Gdy wykonujesz operację na kolorze, musisz osobno rozważyć te komponenty. Innymi słowy, nie możesz mieszać niebieskiego koloru z kolorem zielonym, jeśli chcesz, aby wynik był kolorem, a nie intensywnością. Zasadniczo nie mieszaj jabłek z pomarańczami. Ponieważ chcesz spojrzeć na światło jako cząstkę, musisz zacząć od lokalizacji źródłowej. Światło jest promieniowaniem, więc źródło wysyła wiązkę cząstek w niemal każdym kierunku. Reflektor, na przykład, jest w rzeczywistości źródłem światła omni wewnątrz stożka lub pudłem śledzącym region, który normalnie by się świecił. Ponieważ światło jest cząstką, podlega tym samym prawom fizyki związanym z ruchem. Einstein prawdopodobnie nie zgodziłby się z tym, ale dla twoich celów jest to rozsądny sposób na symulowanie światła. Co więcej, jeśli światło współdziała z niczym, musi więc mieć prędkość. Prędkość ta została oszacowana jako stała na poziomie 300 000 kilometrów na sekundę lub 186 000 mil na sekundę. Sądzę, że możesz powiedzieć, że to dość szybko. Wróćmy do twojego oryginalnego modelu. Masz źródło światła, które wypluwa praktycznie nieważkie cząstki światła (nazywa je fotonami) we wszystkich kierunkach z niewiarygodną prędkością. Ze względu na tę prędkość światło jest natychmiastowe dla wszystkich praktycznych zastosowań. Oznacza to, że w oświetlonej scenie cząstki światła zderzają się z powierzchniami i napromieniowują je. Jeśli wyizolujesz to, by spojrzeć na pojedynczy foton, ten foton będzie wielokrotnie odbijał się na różnych powierzchniach, zanim w końcu dotrze do twojego oka. Gdy

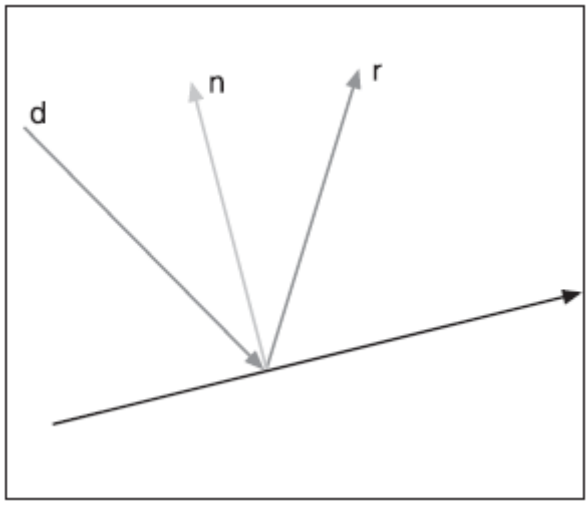
foton dociera do twojego oka, dekodujesz to promieniowanie jako kolor. Ponieważ masa jest tak mała, a prędkość jest tak wysoka, odbicie fotonu można uznać za najprostsze: kolizję czysto elastyczną. W konsekwencji "prawdziwym" sposobem renderowania sceny byłoby rzucenie zestawu promieni we wszystkich kierunkach ze źródła światła i uchwycenie ich w pobliżu płaszczyzny. Jest to jednak całkowicie niepraktyczne, ponieważ obejmuje nieskończoną liczbę kierunkowych wektorów. Zamiast tego, co robi renderers 3D nazywa ray tracing. Proces jest po prostu odwrócony. Zamiast rzucać promienie ze źródła światła, możesz rzucić promień dla każdego piksela na bliskiej płaszczyźnie i określić kolor w zależności od różnych obiektów, które promień uderza. W przypadku gier jest to niepraktyczne, ponieważ działa na poziomie piksela i wymaga wielu wykrycia kolizji. Zamiast rzutować pojedyncze promienie, stosujesz przybliżenia, takie jak rzutowanie tekstury dla plamki świetlnej. Działa to całkiem dobrze, jak widzieliście w poprzednich rozdziałach, ale nie uwzględnia wielu potencjalnych trafień. Na przykład, nie wyglądałoby to realistycznie, gdyby projektować mapę świetlną na szklanym obiekcie, ponieważ szkło robi dwie rzeczy: zniekształca światło i pochłania tylko niewielką ilość intensywności koloru. Zrozumiesz to lepiej w dalszej części rozdziału. Na razie spójrzmy na zachowanie pojedynczego promienia światła, aby zrozumieć, jak wybrać kolor dla danego piksela.

Odblaskowe powierzchnie

Gdy obiekt odbija się od powierzchni, odbija się w tym samym kierunku, z wyjątkiem tego, że kierunek prostopadły do płaszczyzny jest odwrócony. Weź najprostszy przykład: kulkę odbijającą się od podłogi bez grawitacji. Załóżmy, że podłoga jest jedynie płaszczyzną xz. Piłka będzie poruszać się w tym samym kierunku dla elementu x, z. Komponent, którego dotyczy problem, to y, który jest w rzeczywistości odwrócony. Ta sprawa jest dość prosta, ponieważ zajmuje się sytuacją wyrównaną do osi, ale nie musi tak być. Możesz wybrać dowolną płaszczyznę do odbicia obiektu. Musisz trochę zagłębić się w problem i pomyśleć pod względem wektorów. Kierunek odwrócony określa normalna płaszczyzna n. To bardzo łatwe i oczywiste. Jeśli podejmiemy kierunek obiektu d i odejmiemy rzut d na n od d, kończy się na obiekcie, który jest zawsze w samolocie, niezależnie. Jeszcze raz, weź przykład piłki z płaszczyzną xz. Jeśli oddzielisz normalny komponent płaszczyzny od kierunku, y będzie równe zeru, jeśli dopasowane długości. Chcesz, aby obiekt się odbił i nie doznał nieelastycznej kolizji, więc musisz odjąć normalny wektor jeszcze raz. Działa to tylko wtedy, gdy kierunki są równe. Jeśli nie, musisz obliczyć rzutowaną długość kierunku na normalny. Jest to długość wektora, którą należy odjąć. Jak może powiedzieć Regis, oto ostateczna odpowiedź dla wektora odbicia r:

$$\mathbf{r} = \mathbf{d} - 2\mathbf{n}(\mathbf{d} \bullet \mathbf{n})$$

Nie powinno to dziwić, ponieważ zostało już wspomniane w ostatniej części. Poza tym wszystko chyba ma więcej sensu. Jeśli nadal go nie widzisz, spróbuj spojrzeć na rysunek 18.1.



Oczywiście działa to tylko wtedy, gdy n jest znormalizowane, ponieważ tak zdefiniowano projekt i produkt kropkowany. Działa na pojedynczy promień, ale oczywiście nie renderujesz w kategoriach pojedynczych promieni. Jeśli chcesz renderować powierzchnię odbijającą, taką jak lustro, jednym ze sposobów jest użycie bufora szablonu.

Płaszczyzny Zwierciadlane

Technika jest wykonywana w dwóch etapach, tak jak zrobiono cień w poprzednim rozdziale. Pierwszy krok polega na renderowaniu sceny w normalny sposób i renderowaniu lustra tak, aby tylko przełączał bity szablonu. Ogranicza to obszar, przez który obiekty są widoczne. Zacznij od podjęcia najprostszego przypadku - podłogi (płaszczyzny xz), jako powierzchni odbijającej. Jednym z łatwych sposobów na osiągnięcie tego odbicia jest uczynienie sceny tak, aby odzwierciedlało to właśnie tę powierzchnię. Innymi słowy, wyrenderuj scenę z dodatkową transformacją, która po prostu odwraca wartość y:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matryca lustrzana M może być dołączona do matrycy kamery i skutecznie odwraca każdy obiekt wzdłuż płaszczyzny xz. Dzięki buforowi szablonu obiekty znajdujące się nad powierzchnią Xz są renderowane poniżej powierzchni w taki sposób, że są odbijane lustrzanie. Wymaga to dwukrotnego renderowania całej sceny, ale na pewno bije, renderując ją sześć razy, zgodnie z mapowaniem kostki. Jest to nieco zbyt łatwe, ponieważ płaszczyzna jest wyrównana względem osi, ale idea nie jest tak różna dla dowolnej płaszczyzny. Proces otrzymywania tej macierzy transformacji jest dokładnie taki sam jak uzyskanie macierzy transformacji dla obrotu wokół dowolnego wektora. Jest to zbiór przekształceń, które skutkują pojedynczą macierzą transformacji. Nie ma sensu przechodzić przez całą derywację formuły, ponieważ jest ona tak podobna do rotacji wokół dowolnego wektora. Już intuicyjnie wiesz, jak wykonać zwierciadło na osi Z, więc sensownie jest zastosować zestaw transformacji, który da ci płaszczyznę xz, zastosuje transformację lustrzaną i przejdzie przez odwrotność początkową transformacji, aby przywrócić pozycję i obrót obiektu. W związku z tym trzeba zrobić dwie rzeczy. Najpierw musisz przetłumaczyć płaszczyznę na pochodzenie, a następnie obrót, który umieści twój samolot na planie planarnym do płaszczyzny Xz. W tym momencie możesz zastosować M i odwrotność

obrotu / tłumaczenia. Część obrotu została już omówiona w części 5, z obrotami wokół dowolnej osi, więc wszystko, co musisz zrozumieć, to jak uzyskać tłumaczenie, które doprowadzi twój samolot do miejsca pochodzenia. Biorąc pod uwagę dowolny wierzchołek na płaszczyźnie v i normalną płaszczyznę n, poniższa macierz wykonuje trik po zastosowaniu do matrycy kamery M:

$$\mathbf{M} = \begin{bmatrix} 1-2\mathbf{n}_x^2 & -2\mathbf{n}_x\mathbf{n}_y & -2\mathbf{n}_x\mathbf{n}_z & 2(\mathbf{n}\bullet\mathbf{v})\mathbf{n}_x \\ -2\mathbf{n}_x\mathbf{n}_y & 1-2\mathbf{n}_y^2 & -2\mathbf{n}_y\mathbf{n}_z & 2(\mathbf{n}\bullet\mathbf{v})\mathbf{n}_y \\ -2\mathbf{n}_x\mathbf{n}_z & -2\mathbf{n}_y\mathbf{n}_z & 1-2\mathbf{n}_z^2 & 2(\mathbf{n}\bullet\mathbf{v})\mathbf{n}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Oczywiście, już zauważyłeś, że możliwe jest generowanie odbicia poprzez mapowanie środowiska, które jest znacznie wydajniejsze, ponieważ cały świat jest wstępnie skomponowany do tekstury. Jedynym problemem jest to, że zakłada się, że obiekty są nieskończenie oddalone od powierzchni odbijającej, co może doprowadzić do znacznej deformacji obrazu dla płaskiej powierzchni. Twoje oczy są dość dobrze wyszkolone do płaskich powierzchni, więc lekkie odkształcenie może wydawać się czymś znacznie większym. Zaletą tej techniki jest to, że umożliwia ona dowolną formę animacji przed powierzchnią odbijającą. Postać poruszająca się przed lustrem pojawi się także w lustrze, podczas gdy w przypadku mapy środowiskowej nie będzie to możliwe, chyba że obliczymy mapę w każdej klatce, przez co ogólna wydajność będzie gorsza niż ta metoda. Oczywiście, musisz zastosować tę technikę rekurencyjnie, jeśli dwa lustra widzą się nawzajem. Strasznie kosztowne jest wielokrotne rysowanie sceny. Ale jak często widzisz grę, która ma więcej niż jedno sąsiednie lustro? Ogólnie mapy środowiskowe są lepsze, jeśli chcesz renderować naprawdę złożony świat, ponieważ zawierają dużo więcej informacji w kilku wybranych teksturach, co jest znacznie szybsze. Aby przyspieszyć obliczenia, prawdopodobnie warto rozważyć obcinanie sceny do zestawu płaszczyzn zdefiniowanych przez granicę lustra. Kiedy patrzysz w lustro, jest wysoce prawdopodobne, że niektóre fragmenty sceny nie są w ogóle widoczne, więc obliczenie zestawu płaszczyzn dla wypukłej objętości obiektu (jak to było zrobione w przypadku cienia i uśmiercania obiektu) nie jest kiepski pomysł. Ponieważ zwierciadła składają się często z dwóch par ortogonalnych krawędzi, wypukłość obiektu jest już zagwarantowana. Z drugiej strony możesz również wykonać wstępny wypukły wielokąt lustra, jeśli zdecydujesz się nie mieć wypukłego lustra. Oczywiście oznacza to, że zezwalasz na niektóre obiekty, które potencjalnie nie są widoczne, ale przynajmniej będą szybsze. Innym podejściem, które warto wziąć pod uwagę, jest zastosowanie pewnej formy testu widoczności, jak pokazano w poprzednim rozdziale. Na przykład, być może mógłbyś nadać obiektom czysty, jednolity kolor, zamiast barwić je drogim cieniowaniem przez lustro. Powierzchnie refrakcyjne Powierzchnie odbijające przyjmują światło i odbijają je w przeciwnym kierunku niż normalna powierzchnia. W przezroczystym materiale można również zaobserwować właściwości refrakcyjne. Refrakcja jest zniekształceniem światła wywołanym zmianą materiału, przez który przechodzi. Możesz obserwować ten efekt, spoglądając w dół na kałużę wody. Zauważ, że części basenu wyglądają, jakby się chwieją. Gdyby woda była całkowicie przezroczysta, nie zauważyłbyś żadnych zniekształceń. Ale robisz, ponieważ woda jest refrakcyjna. Woda jest również odblaskowa, ponieważ świecące światło na basenie również powoduje chwiejny wzór. Kolejnym przykładem jest szkło. Możesz to przejrzeć, ale to, co widzisz, jest w rzeczywistości lekko zniekształcone. Jest to ponownie efekt załamania. Światło jest procesem fizycznym, który można wydedukować tylko przez obserwację. Jedna taka obserwacja dotycząca powierzchni refrakcyjnych, często nazywana Prawem Snella, obserwuje następującą zależność między dwoma materiałami o odpowiednich współczynnikach załamania η1 i η2:

$$\eta_1 \sin(\theta_1) = \eta_2 \sin(\theta_2)$$

W tym przypadku kąt θ_1 jest określany jako kąt padania, a θ_2 jest kątem załamania. Pierwszy kąt opisuje, w jaki sposób promień światła opuszcza materiał, a drugi opisuje, w jaki sposób światło przenika nowy materiał. W konsekwencji sam materiał nie modyfikuje kierunku światła. To naprawdę przejście z jednego materiału na drugi, które to robi. Na przykład powodem, dla którego twoja pula wygląda na chybotliwą, jest to, że światło zmienia się w czasie podróży z powietrza w wodę. Wektor światła początkowo ma kąt θ_1, a ten kąt jest przekształcany na θ_2 tak, że poprzednie równanie jest spełnione dla dwóch stałych refrakcji. Zachowanie promienia jest podobne do refleksji. Do refleksji na płaszczyźnie Xz, odwołajmy się tylko do komponentu y zmiany. W takim przypadku zmienia się tylko kąt między wektorem a płaszczyzną. W 2D możesz wymyślić tę relację:

$$\theta_2 = \arcsin\left(\frac{\eta_1}{\eta_2}\sin(\theta_1)\right)$$

Niestety ta notacja niewiele pomoże ci w 3D. Prawo w 3D jest zdefiniowane podobnie, ale aby to zrozumieć, powinieneś spojrzeć na problem pod względem wektorów, które są skalowalne do wyższych wymiarów. Jako konwencja notacyjna, bądźmy nadchodzącym promieniem światła, i bądźmy przekształconym promieniem światła. Wektor t może być wyrażony jako kombinacja jednego wektora wzdłuż normalnej powierzchni n i innego wektora m, prostopadłego do n. Tak więc masz następujące:

$$\mathbf{t} = -\mathbf{n} \cdot \cos(\theta_2) + \mathbf{m} \cdot \sin(\theta_2)$$

Masz normalny wektor powierzchni, ale tak naprawdę nie masz m. Jeśli pamiętasz, że kąt wzdłuż powierzchni jest jedyną rzeczą, która się zmienia, a nie cały kierunek, m jest zdefiniowany w następujący sposób:

$$\begin{aligned}\mathbf{m} &= \frac{\text{perp}_{\mathbf{n}}\mathbf{s}}{\sin(\theta_1)} \\ &= \frac{\mathbf{s} - (\mathbf{n} \bullet \mathbf{s})\mathbf{n}}{\sin(\theta_1)}\end{aligned}$$

Teraz złożenie wszystkiego razem daje ostateczną odpowiedź:

$$\begin{aligned}\mathbf{t} &= -\mathbf{n}\cdot\cos(\theta_2) - \frac{\mathbf{s}-(\mathbf{n}\bullet\mathbf{s})\mathbf{n}}{\sin(\theta_1)}\cdot\sin(\theta_2)\\ &= -\mathbf{n}\cdot\sqrt{1-\sin^2(\theta_2)} - \frac{\sin(\theta_2)}{\sin(\theta_1)}\left(\mathbf{s}-(\mathbf{n}\bullet\mathbf{s})\mathbf{n}\right)\\ &= -\mathbf{n}\cdot\sqrt{1-\frac{\eta_1^{\,2}}{\eta_2^{\,2}}\sin(\theta_1)} - \frac{\eta_1}{\eta_2}\left(\mathbf{s}-(\mathbf{n}\bullet\mathbf{s})\mathbf{n}\right)\\ &= -\mathbf{n}\left(\sqrt{1-\frac{\eta_1^{\,2}}{\eta_2^{\,2}}\left(1-(\mathbf{n}\bullet\mathbf{s})^2\right)} + \frac{\eta_1}{\eta_2}(\mathbf{n}\bullet\mathbf{s})\right) - \mathbf{s}\frac{\eta_1}{\eta_2}\end{aligned}$$

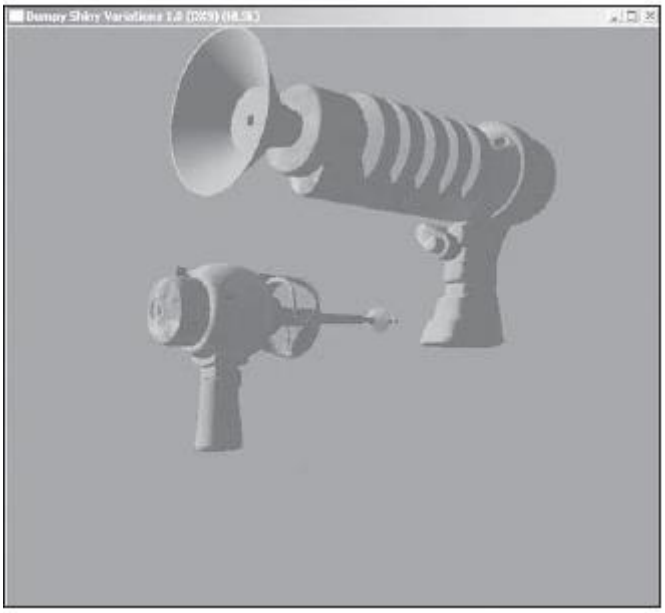
Nie jest to najładniejsze równanie, na które można liczyć, ale nie możesz tak naprawdę zaprzeczać matematyce, prawda? Zauważ, że funkcja nie jest zdefiniowana dla wszystkich wartości. Możliwe jest uzyskanie ujemnego pierwiastka kwadratowego, jeśli wybór współczynnika nie jest zbyt dobry. Zazwyczaj wszystko jest w stosunku do powietrza. Powietrze ma czynnik 1, a woda jest [4/3]. Fizycznie mówiąc, jeśli otrzymasz ujemny pierwiastek kwadratowy, oznacza to, że pracujesz z materiałem, w którym występuje całkowite odbicie wewnętrzne. Światło jest uwięzione w materiale, a dla twoich celów jest całkowicie bezużyteczne.

Źródła światła

Jeśli badasz światło pod kątem pojedynczego promienia odbijającego się na różnych powierzchniach, jest to interesujące, ale nie jest zbyt praktyczne w przypadku gier. Nie można strzelać do nieskończonej liczby promieni świetlnych, aby stworzyć światło. Zamiast tego wolę przyjrzeć się zachowaniom, które podają źródła światła. Modele te opierają się na obserwacji fizycznej, którą możesz wykonać sam. Zazwyczaj istnieją równania, które są zagracone za pomocą stałych, które trzeba dostroić do swoich konkretnych potrzeb. Równania wymuszają ogólne zachowanie światła, ale szczegóły na jego temat są dopracowywane za pomocą stałych.

Światła otoczenia

Światło otoczenia to kumulacja światła niskiej częstotliwości, które pochodzi z wielu źródeł odbicia na scenie. Kiedy włączasz światło w pokoju, jeden konkretny obszar pokoju jest dobrze oświetlony. Reszta pomieszczenia jest również oświetlona, ponieważ promienie światła odbijają się praktycznie od wszystkich powierzchni. To jest światło otoczenia. W związku z tym światło otoczenia może być postrzegane jako podstawowy poziom oświetlenia sceny. To naprawdę nie zależy od kierunku, ponieważ jest to przybliżenie, które uwzględnia światło pochodzące z odbić we wszystkich kierunkach. Zazwyczaj światło z otoczenia jest stałą opisującą najniższy poziom światła, jaki może osiągnąć region, niezależnie od tego, czy jest on bezpośrednio oświetlony czy zacieniony. Spójrz na 18.2.



Światło kierunkowe

Światło kierunkowe jest zbliżone do mapowania otoczenia, ponieważ przybliża ono również światło pochodzące z bardzo odległego (czytaj: nieskończenie odległego) obiektu. Słońce, gwiazdy lub inne światła docierające z daleka emitują kierunkowe światło. Ze wszystkich praktycznych powodów rodzaj światła nie zmniejsza się wraz z odległością. Światło słoneczne nie zmniejsza intensywności, gdy dociera na przykład do Ziemi. Światła kierunkowe nie mają pozycji w przestrzeni. Są one zdefiniowane wyłącznie przez kierunek, jak sama nazwa wskazuje.

Światło punktowe

W przeciwieństwie do światła kierunkowego punktowe światło ma punkt zakotwiczenia. W związku z tym ma również tłumienie. Innymi słowy, natężenie światła jest proporcjonalne, w zależności od odległości między światłem punktowym a oświetlonym obiektem. Włączenie lampy w pokoju daje taki efekt. Im dalej lampa jest od ściany, tym mniej intensywne jest uderzenie światła w ścianę. Jeśli chcesz matematycznie opisać intensywność światła w zależności od odległości od światła punktowego, możesz zapisać je jako takie:

$$i = \frac{1}{a + b \cdot d + c \cdot d^2}$$

Wartości a, b i c są stałymi określonymi dla światła. W rzeczywistości, jeśli chcesz wykreślić to równanie, możesz zobaczyć, jak zmienia się natężenie w zależności od odległości:

$$\frac{1}{a + b \cdot x + c \cdot x^2}$$

Możesz dostroić stałe, aby wygenerować światło, które działa dla twojego świata. Naprawdę nie ma magicznej liczby, która działa we wszystkich przypadkach, po prostu dlatego, że rozmiar świata naprawdę zależy od jednostek i rozmiaru używanych obiektów. Plug and play, i powinieneś być w stanie wymyślić równanie, które ma sens w twoim przypadku. Sama intensywność opisuje, ile światła przechodzi przez medium, takie jak powietrze. W związku z tym, jeśli chcesz obliczyć intensywność

białego światła, musisz pomnożyć intensywność dla każdego kanału koloru białego (<1, 1, 1> i). Jeśli światło jest czerwone, musisz pomnożyć kolor czerwony przez intensywność światła <1, 0, 0> i. Obsługa natężenia światła jest bardzo prosta. Jedynym "groszkiem", którego możesz chcieć uniknąć, jest punkt, w którym x = 0. Musisz upewnić się, że gdy odległość wynosi 0, zakres intensywności nie daje liczby większej niż 1. Jeśli tak, to może uzyskać jasny kolor większy niż 1, co nie ma sensu, jeśli każdy kanał koloru jest związany [0, 1]. Posiadanie a = 1 rozwiązuje ten problem, ale może nie dać pożądanego rodzaju światła, więc bądź ostrożny.

Reflektor

Światło punktowe jest bardzo podobne do światła punktowego, ale jest zamknięte po bokach. Może być okrągły lub kwadratowy lub dowolny dziwny kształt. Przyjmijmy tutaj zaokrąglony kształt. Źródło światła jest rzeczywiście punktowym światłem wewnątrz cylindra z jednym końcem otwartym. Ponieważ wciąż mamy do czynienia ze światłem punktowym, nastąpi osłabienie intensywności światła wraz z odległością. Ale pojawia się również inne zjawisko: promienie światła odbijają się od cylindra. Gdyby to nie było prawdą, reflektor dawałby bardzo trudny cień, ale tak nie jest. Światło stopniowo pogarsza się ze środka reflektora. Wynika to z różnych wiązek światła, które odbijają się na stożku i kończą na powierzchni, tworząc ładny eliptyczny gradient. Im dalej powierzchnia znajduje się od środka światła punktowego, tym mniej belek uderza w nią. Innymi słowy, istnieje znacznie mniej wiązek światła poza regionem, który jest bezpośrednio widoczny ze światła punktowego wewnątrz cylindra. Im dalej od rzutu cylindra, tym mniej światła się zgromadzi. Musisz zdefiniować kilka dodatkowych rzeczy. Światło punktowe ma źródło zdefiniowane przez pozycję p i jest sprzężone z kierunkiem d, którego nie należy mylić z odległością od punktu w przestrzeni d. Równanie określające natężenie światła w punkcie v, z wyżej wymienionymi parametrami, jest podane za pomocą następującego równania:

$$i = \frac{\max\left\{\mathbf{d} \bullet \frac{\mathbf{v}-\mathbf{p}}{\|\mathbf{v}-\mathbf{p}\|_2}, 0\right\}^p}{a + b \cdot d + c \cdot d^2}$$

Jak widać, równanie jest takie samo jak równanie światła punktowego, poza tym, że uwzględnia ono także kąt między wektorem vp (wektor kierunku światła) a wektorem d (wektor punktu względem źródło światła). W konsekwencji im większy jest kąt, tym mniejsza wartość cosinusa dla tego kąta i mniejsza intensywność. To wszystko ma sens, jeśli poświęcisz kilka sekund na przemyślenie tego. Ostatnią tajemniczą zmienną, jaką zauważyłeś, jest wykładnik p. Ten wykładnik określa wrażliwość kąta. Na przykład wartość 100 dałaby bardzo solidne światło, podczas gdy wartość 1 dawałaby znacznie gładsze i bardziej stopniowe światło, które idzie znacznie dalej.

Modele oświetlenia

Skoro już ustaliłeś, że światło odbija się od powierzchni, czy badanie powierzchni odbijających światło nie ma sensu? Gdyby wszystkie powierzchnie były proste, jak wcześniej założono, każdy materiał byłby w 100 procentach odblaskowy, jak lustro. Jednak nawet lustra pochłaniają trochę światła, które odbija się od nich. Jeśli weźmiesz dwa lustra i położysz je naprzeciwko siebie, to, co widzisz w nich odbija się, staje się bardziej zielone i bardziej zielone (kolor samego szkła), aż się ściemni. Możesz symulować to za pomocą lustra, dodając przezroczystą zielonkawą teksturę na scenie widzianej przez lustro. Podobnie, w części 8 dowiedziałeś się, że niektóre przedmioty pozostają w miejscu, zamiast przesuwać się z powodu tarcia. Niektóre obiekty są trudniejsze do przepchnięcia przez niektóre powierzchnie niż

inne powierzchnie (np. Papier ścierny w porównaniu do lodu). Dzieje się tak dlatego, że na poziomie atomowym powierzchnia nie jest tak gładka, jak można się spodziewać. Ten efekt ma również duże konsekwencje dla światła. Ponieważ światło odbija się od powierzchni zdefiniowanej przez równanie widoczne wcześniej, ważna jest orientacja mikropowierzchni materiału. Jeśli na przykład weźmiesz kawałek bibuły, wiele drobnych nici zostanie ściśniętych razem, aby je utworzyć, a obliczenie dokładnego kąta powierzchni, z którego odbija się każdy promień światła, byłoby nierealne. W związku z tym musisz wymyślić modele, które przybliżają sposób, w jaki światło współdziała z konkretnymi materiałami, najlepiej jak to możliwe.

Oświetlenie rozproszone

Rozproszone oświetlenie zostało spopularyzowane przez wczesne gry komputerowe, w których jakość obrazu była naprawdę luksusem, a oprogramowanie rządziło Ziemią. Powierzchnia rozproszona odbija światło w bardziej lub mniej losowy sposób. Oznacza to, że dany obszar jest oświetlony względnie równo. Na przykład włącz najbliższą lampę. Teraz weź mały dywanik i umieść go pod lampą. Powinno to usunąć efekt reflektora, lub przynajmniej go zmniejszyć. Jest to również zgodne z równaniami reflektorowymi, w których odległość jest duża. W tej pozycji dywan wygląda stosunkowo równomiernie. Z drugiej strony, jeśli weźmiesz kawałek plastiku i spojrzysz na niego zza lampy, prawdopodobnie zauważysz w środku środek (który odpowiada bezpośredniemu światłu). Różnica polega na tym, że dywan odbija światło w sposób losowy ze względu na charakter i rozmieszczenie włókien. Z drugiej strony tworzywo sztuczne oferuje o wiele bardziej równomierną powierzchnię i nie jest powierzchnią rozproszoną. Ponieważ odbicie jest stosunkowo jednolite, oświetlenie nie zależy od położenia obiektu w modelu czysto rozproszonym. Intensywność jest naprawdę wartością, która wylicza ile światła trafia na dany obszar. To stosunek oświetlenia do powierzchni. Jeśli spojrzeć na światło, które pochodzi z kierunku d i założyć model oświetlenia rozproszonego, obliczanie intensywności zależy od tego obszaru. Okazuje się, że oświetlony obszar powierzchni naprawdę zależy od nachylenia tej powierzchni. Efekt ten można zobaczyć, jeśli weźmiesz kawałek papieru. Obróć kawałek papieru, aby zmienić kąt padania światła na papier, a zauważysz, jak intensywność światła zmniejsza się dość szybko. To samo dzieje się z naszą planetą. Powodem, dla którego jest cieplej w Kalifornii niż na Islandii, nie jest to, że Islandia jest dalej od słońca (chociaż prawdopodobnie ma niewielką ilość). Prawdziwym powodem jest to, że Islandia znajduje się pod takim kątem, że obszar, który odbiera światło, jest znacznie większy niż obszar Kalifornii, który jest znacznie prostopadły do wektora światła pochodzącego ze słońca. Z tego samego powodu robi się coraz zimniej w nocy. Słońce znajduje się pod mniejszym kątem i jego intensywnością zmniejsza się. W starszych grach wielokąty miały tylko jeden kolor, który zmieniał się wraz ze zmianą kształtu. Ta forma oświetlenia nazywa się cieniowaniem Lamberta. Chodzi po prostu o zmianę intensywności koloru wielokąta w zależności od jego orientacji od źródła światła. Tak naprawdę nie przynoszą one dobrych rezultatów, ale jest to podstawowa zasada światła, którą można zastosować w połączonych modelach oświetlenia. Cosinus można pobrać przez produkt kropkowany, a poniższe równanie wyraża komponent światła dodany przez rozproszone oświetlenie:

$$i_{\text{diff}} = i_{\text{source}} \left(\mathbf{n} \bullet \mathbf{d} \right)$$

Oświetlenie rozproszone ma bezpośredni wpływ na intensywność całego źródła oświetlenia zdefiniowane przez jego źródło. Oczywiście, jeśli produkt z kropką w poprzednim równaniu jest ujemny, oznacza to, że powierzchnia jest ukryta i nie powinna w ogóle otrzymywać światła, powodując, że intensywność gwałtownie spadnie do zera. W tej technice powstaje również wspanialsza forma oświetlenia, która każdego dnia rzuca Lambertowi w cień. Ponieważ już ustaliłem, że możesz obliczyć normalny wektor nie tylko na trójkąt, ale także na wierzchołek, oznacza to, że możesz obliczyć jeden

kolor na jeden wierzchołek. Jeśli obliczysz rozproszony kolor światła na wszystkich trzech wierzchołkach dla danego trójkąta, otrzymasz trzy kolory (po jednym na wierzchołek). Biorąc pod uwagę te kolory, można następnie renderować trójkąt, interpolując kolor z jednego wierzchołka do drugiego w dół po lewej i prawej krawędzi i interpolując ponownie od lewej do prawej. W rzeczywistości jest to mniej więcej ten sam proces co próbkowanie bilinearne, z tym że interpolacja nie jest ortogonalna. Takie trójkąty można renderować za pomocą dowolnego przyzwoitego interfejsu API 3D, po prostu renderując trójkąt z trzema różnymi kolorami dla każdego wierzchołka. To forma cieniowania jest powszechnie nazywana cieniowaniem Gourauda, a to technika w oprogramowaniu, która zapewnia najlepszy stosunek jakości do szybkości. Ale w świecie sprzętu, który naprawdę dba? Nadal się troszczysz, ponieważ koncepcja jest ważna. Powszechnie używane w grach są tekstury zwane rozproszonymi mapami. Te tekstury mają tylko jeden cel, a to daje obraz większego poczucia głębi. Celem tej tekstury jest zmiana intensywności różnych części tekstury. Na przykład, jeśli renderujesz świat ze zwykłymi teksturami, będzie wyglądać bardzo nudno, ponieważ nie będziesz miał pojęcia światła. Jeśli utworzysz rozproszone mapy strategicznie rozmieszczone w scenie, aby przyciemnić rogi pomieszczenia i zastosować podobne oświetlenie, te mapy znacznie zwiększą głębię sceny. Oczywiście można by argumentować, że jeśli chcesz odcień skrzynki z ciemniejszym kolorem, możesz to zrobić w samej fakturze, ale tutaj problemem jest pamięć. Jeśli chcesz zastosować na mapie różne halo przypominające reflektory, lepiej mieć dużo pamięci, aby wygenerować wszystkie możliwe kombinacje. Zamiast tego możesz korzystać z multiteksturowania z trybem modulacji, który wygeneruje to, czego możesz od niego oczekiwać. Spójrz na rysunek 18.3, aby zobaczyć przykład pistoletu renderowanego za pomocą rozproszonego oświetlenia.

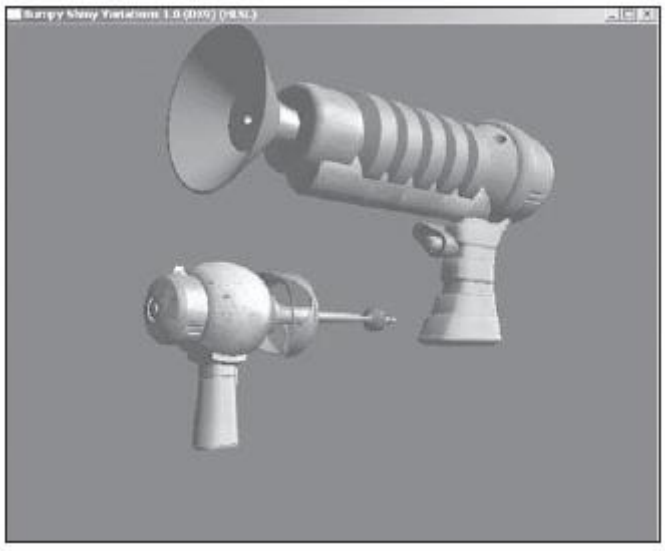
Światło Odblaskowe

Jeśli weźmiesz bardzo błyszczący, nieprzejrzysty obiekt, taki jak malowany metal, zauważysz, że ma on małe halo. W rzeczywistości wygląda mniej więcej jak odbicie żarówki na nieprzejrzystej powierzchni. Ta nieruchomość jest wykazywana przez każdą powierzchnię, która nie jest zbyt gruba. Ciekawostką w tej halo jest to, że jest bardzo jasna tylko w jednym konkretnym regionie na obiekcie, i jest to sekcja, w której promienie światła odbijają się bezpośrednio od źródła do twoich oczu. Z definicji światło zwierciadlane zależy od pozycji widza. Ponownie możesz to sprawdzić samodzielnie, poruszając się po obiekcie i przesuwając światło. Dzięki tej obserwacji można zastosować prostą formułę w celu racjonalnego przybliżenia tego efektu. Prawdę mówiąc, można prawie powiedzieć, że oświetlenie zwierciadlane jest bardzo ostrym światłem reflektora, które zależy od kąta odbicia, a nie od kierunku światła. Ponieważ są one bardzo podobne, następujące równanie zapewnia przyzwoite przybliżenie oświetlenia zwierciadlanego:

$$i_{spec} = \left(\mathbf{r} \bullet \frac{\mathbf{v} - \mathbf{p}}{\|\mathbf{v} - \mathbf{p}\|_2} \right)^m$$

Tutaj v definiuje się jako pozycję widza w przestrzeni, p jest położeniem światła, a r jest wektorem odbijającym, jak zdefiniowano poprzednio. Matrycowy wykładnik m naprawdę wystarczy, aby upewnić się, że intensywność jest naprawdę ostra. Nadanie małej wartości, takiej jak 1 do m, da bardzo dużą atrakcję, praktycznie oświetlając cały obiekt. Z drugiej strony, posiadanie wykładnika 50 da ci bardzo ostry cel (kalambur przeznaczony). Jeśli nie przeszkadza ci to, że samo podświetlenie jest nieco przesunięte (co jest często rozsądnym założeniem), jest to całkiem przyzwoity model. W rzeczywistości przyjrzeć mu się dokładnie, jest bardzo podobny do tego, który został użyty do mapowania środowiska. Z tego powodu można symulować oświetlenie punktowe za pomocą mapy środowiska. Ze względu na swój charakter nie jest konieczne stosowanie niczego więcej niż sferyczną mapę środowiska. Po raz

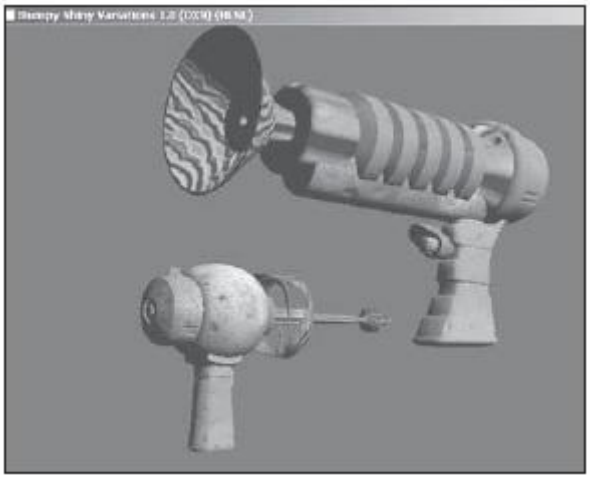
kolejny, w fakturze, po prostu musisz określić intensywność koloru, która ma być obecna dla danego kąta odbicia, a gra będzie odtwarzana. W tym konkretnym przypadku powinieneś patrzeć na obiekt podobny do kuli, gdzie niewielka część w środku kuli jest intensywnie oświetlona. Jeśli chcesz renderować taką teksturę w fantazyjnym narzędziu renderowania 3D, możesz to zrobić, renderując sferę i ustawiając element specular w tym narzędziu artystycznym, aby wygenerować podświetlenie, które spodziewałbyś się zobaczyć na swoich obiektach. Był też bardzo popularny i fajny efekt w dawnych czasach, który był powszechnie znany jako mapowanie Phonga (czasami nieprawidłowo nazywane cieniowaniem Phonga). W cieniu Gourauda kolor jest interpolowany z jednego wierzchołka na drugi. W odwzorowaniu Phonga wektor normalny jest interpolowany z jednego wierzchołka na drugi. Działa to tylko wtedy, gdy twoja tekstura jest sama zakrzywiona, na przykład wygenerowana dla mapy sferycznej. Problem z cieniowaniem Gourauda polega na tym, że nie można umieścić elementu zwierciadlanego w środku trójkąta, ponieważ kolory są interpolowane liniowo. Mapowanie Phonga rozwiązuje ten problem, ponieważ jednostka generująca tekst GPU interpoluje wartości normalne. Jeśli możesz użyć mapy rozproszonej do zmiany koloru obiektu, możesz także użyć mapy do modulacji intensywności dla składnika zwierciadlanego. Takie mapy nazywane są mapami połysku. Ponieważ mapy dyfuzyjne określają intensywność koloru w <x, y>, mapy połysku określają element zwierciadlany w każdym punkcie na powierzchni. Na przykład można zbudować skrzynię z płynem o wysokim połysku płynącym po niej jedynie dzięki połyskowi przyłożonemu do cieczy za pomocą mapy połysku. Jest to świetne, gdy chcesz, aby niektóre części obiektów świeciły bez konieczności cięcia wielokąta na kilka mniejszych elementów, aby zarysować błyszczącą część. Spójrz na rysunek 18.4, aby zobaczyć przykład pistoletu renderowanego z podświetleniem.

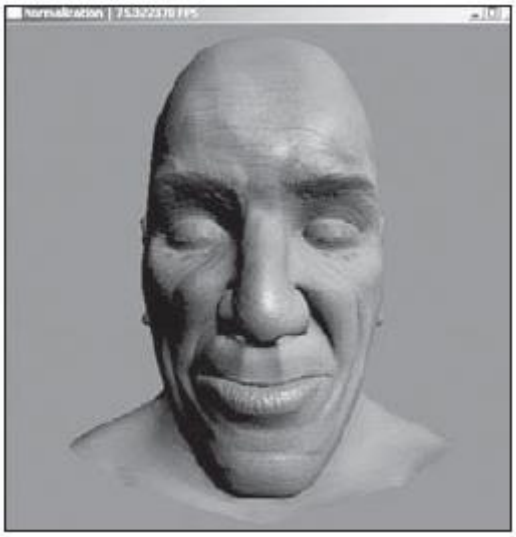


Bumpmapping

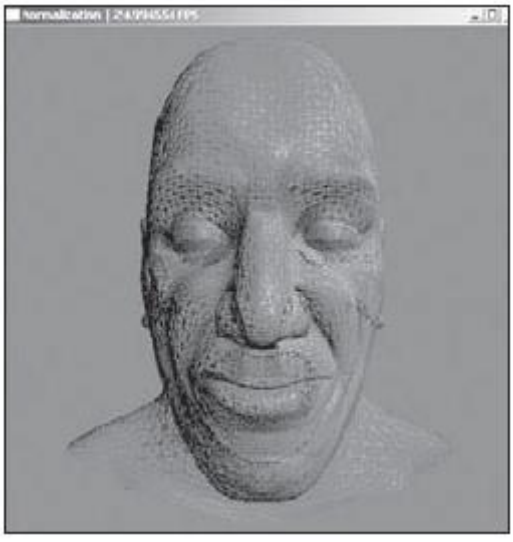
Jeśli dobrze się przyjrzysz obrazowi, możesz stwierdzić, że dom to dom, pies to pies itp. Dokładnie, możesz określić, gdzie zaczyna się kształt psa i gdzie zaczyna się dom. Informacje, które mówią, że to jest kolor. Gdyby pies był tego samego koloru co dom, nie byłbyś w stanie dostrzec różnicy. No, to nie jest do końca prawdą. Z powodu oświetlenia nadal będziesz widział różnicę. Kiedy patrzysz na czarno-białe zdjęcie, jedyną rzeczą, która mówi ci, gdzie kończy się pies, a zaczyna dom, jest ciemność lub intensywność obrazu. Dzięki tej części zrozumiesz, że jeśli sekcją obrazu jest pies, ciemniejsze obszary to najprawdopodobniej te, które są zacienione, lub przynajmniej obszary, gdzie nachylenie może być bardziej strome niż obiekty skierowane bezpośrednio w kamerę. Jeśli więc można ogólnie wywnioskować, że większa intensywność implikuje powierzchnię, która prawdopodobnie stoi przed kamerą, a która ma niższą intensywność, gdy światło nie uderza w powierzchnię prostopadle,

otrzymujemy coś naprawdę interesującego. Spójrz na rysunki 18.5 i 18.6 na przykładach bumpmapping.





Możesz również zobaczyć, jak wygląda rysunek 18.6 na poziomie geometrycznym, patrząc na rysunek 18.7.



Biorąc pod uwagę czarno-białe zdjęcie, możesz spojrzeć na różnicę między sąsiednimi tekselami i zgadnąć, na ile nachylenia między tymi dwoma tekselami. Jeśli kolor pozostaje taki sam, normalna na tej powierzchni jest prawdopodobnie skierowana w stronę światła. Jeśli kolor od lewej do prawej przechodzi od ciemnego do jasnego, normalna powierzchnia jest prawdopodobnie skierowana w lewo. Podobnie w przypadku jaśniejszego / ciemniejszego regionu, gdzie normalny może wskazywać na prawo. Jak więc to może ci pomóc? Jak już wiesz, powierzchnia nie jest idealnie płaska. Składa się z rzeźbionych kawałków i wytłaczanych krawędzi. Jeśli potrafisz wykryć miejsca w teksturach, w których części wycinają się lub rzeźbią, możesz odpowiednio obliczyć światło i rozjaśnić teksturę w zależności od kąta widzenia (pomyśl o cieniowaniu Lamberta z podziałem na piksele). Aby zrobić matematykę na tym, potrzebujesz normalnego wektora dla powierzchni. Jak więc znaleźć normalny wektor z mapy tekstury?

Budowa bumpmap

Nie wszystkie tekstury można wepchnąć do ogólnego algorytmu budowy bumpmap i uzyskać przyzwoite wyniki. Metodą, która daje najlepsze wyniki, jest oczywiście to, w którym możesz samemu przygotować normalne wektory. Innymi słowy, zamiast obliczać normalny wektor dla każdego piksela, edytuj normalne wektory ręcznie i zapisz wynik w fakturze. Zamiast przechowywać teksturę w postaci czerwonych, zielonych i niebieskich komponentów, można przechowywać teksturę jako normalną mapę, gdzie każda z czterech par wartości reprezentuje cztero-wektor normalnej. Jeśli pracujesz z programem do modelowania 3D i możesz generować gładkie obiekty, określenie normalnego dla tekstury generowanej za pomocą takiego narzędzia nie jest zbyt trudne, jeśli obsługuje tę funkcję. Niestety, narzędzia do tego nie są bardzo powszechne, a Ty ". re w zasadzie po to, aby samemu wykonać brudną robotę. Pierwszą rzeczą, którą musisz zrobić, to przekonwertować obraz na mapę intensywności. Mapa intensywności informuje o tym, ile światła dociera do danej pozycji. Dobrym sposobem na odgadnięcie głębokości powierzchni jest spojrzenie na jej intensywność. Pierwszym krokiem jest przekształcenie obrazu kolorowego RGB w czarno-biały obraz o intensywności. Jest to całkiem łatwe i odbywa się za pomocą równania w sekcji "Light a Ray Tracing Model" wcześniej w tym rozdziale. Kiedy już to zrobisz, musisz obliczyć przypuszczenie dla normalności. Weź mapę intensywności jako mapę wysokości. Innymi słowy, rozważ mapę intensywności jako mapę, na której biały piksel znajduje się powyżej czarnego piksela. W ten sposób możesz obliczyć nachylenie w różnych kierunkach. Jeśli obliczysz dwa nachylenia, a więc dwa wektory, możesz obliczyć normalny wektor do obu z nich, a w konsekwencji normalny w tym punkcie. Z matematycznego punktu widzenia:

$$\mathbf{u}(x,y) = \langle 1, 0, h(x+1,y) - h(x-1,y) \rangle$$
$$\mathbf{v}(x,y) = \langle 1, 0, h(x,y+1) - h(x,y-1) \rangle$$
$$\mathbf{n}(x,y) = \langle -\mathbf{u}_z, -\mathbf{v}_z, 1 \rangle$$

Być może zauważyłeś, że dwa przykładowe tekle nie sąsiadują ze sobą. Powodem jest po prostu zachowanie symetrii w modelu. Gdybyś musiał wybrać dwa sąsiednie tekle, musiałbyś wybrać texel powyżej lub poniżej bieżącego, a żaden nie daje wyniku symetrycznego. Oczywiście poprzedni wektor niekoniecznie jest znormalizowany, więc powinieneś go normalizować. Już wcześniej przeszedłeś proces, więc nie powinno być wiadomością, że ten wektor musi zostać przekształcony w zakres [0, 1]. Musisz podzielić każdy komponent przez 2 i dodać 5 aby wykonać normalne mapowanie sześcianu

Przestrzeń styczna

Teraz masz normalny wektor, który możesz wykorzystać do obliczenia ilości światła na powierzchni. Ale wciąż masz jeden problem: wektory, które wymyślisz, odnoszą się do kamery, która spogląda w dół <0, 0, 1>. Tekstura zastosowana do obiektu może znajdować się na tylnej powierzchni, w takim przypadku jakikolwiek wektor powinien być skierowany w tym samym kierunku co kamera. Jednak wektory uzyskane z tego obliczenia są skierowane w stronę kamery. Musisz porównać jabłka z jabłkami. Musisz przekonwertować jedno lub drugie, tak aby znajdowały się w tym samym miejscu. Możesz albo przesuwać wektory do przestrzeni obiektów, albo przesuwać wektory obiektów do przestrzeni stycznej. Te dwie techniki są równoważne; jedyną różnicą jest perspektywa. Przestrzeń styczna może być zdefiniowana jako ortonormalny układ współrzędnych, gdzie norma danego wierzchołka v jest <0, 0, 1> i tym samym pasuje do tego, co masz na normalnej mapie. Na przykład, jeśli wektor miałby przebiegać od światła do wszystkich trzech wierzchołków trójkąta, można przekształcić ten wektor w przestrzeń styczną, a następnie interpolować ten wektor w trójkącie podczas mapowania tekstury. Jedyne prawdziwe pytanie brzmi: jak się nawracacie do takiego układu współrzędnych? Przypuśćmy, że chcesz przekonwertować z przestrzeni stycznej na przestrzeń obiektu. Chcesz przekształcić wektor <0, 0, 1> w przestrzeń obiektu. W tej sytuacji musisz zachować ostrożność, aby zachować kierunki mapowania tekstury. Innymi słowy, musisz upewnić się, że kwadratowe kierunki teksturowania u i v są zachowane. u powinna odpowiadać osi x w przestrzeni stycznej, t powinno odpowiadać osi y w przestrzeni stycznej. Przestrzeń styczna jest często definiowana przez trzy komponenty: normalny n, tangens t i binormal b. Często tworzą one podstawę ortogonalną, ale nie ma takiej gwarancji. To wszystko zależy od kierunku twoich współrzędnych teksturowania. Załóżmy, że masz trójkąt zdefiniowany przez trzy wierzchołki {a, b, c} i odpowiadające im współrzędne tekstury <u {a, b, c}, v {a, b, c}>. Pierwszą i najbardziej oczywistą rzeczą do zrobienia jest obliczenie wszystkiego względem wierzchołka, dla którego chcemy obliczyć przestrzeń styczną. Załóżmy, że ten wierzchołek to a. Tak więc masz następujące nowe definicje:

$$\mathbf{p} = \mathbf{b} - \mathbf{a}$$
$$\mathbf{q} = \mathbf{c} - \mathbf{a}$$
$$\langle u_1, v_1 \rangle = \langle u_b - u_a, v_b - v_a \rangle$$
$$\langle u_2, v_2 \rangle = \langle u_c - u_a, v_c - v_a \rangle$$

Ponieważ chcesz, aby mapowanie tekstury było wykonywane na tej samej ścieżce po transformacji, muszą być spełnione następujące dwa równania:

$$\mathbf{p} = u_1\mathbf{t} + v_1\mathbf{b}$$
$$\mathbf{q} = u_2\mathbf{t} + v_2\mathbf{b}$$

W ten sposób można wyizolować dwa nieznane kierunkowe wektory, które tworzą liniową kombinacją (transformacją) jak dla wektorów wierszowych p, q, t, b:

$$\begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ \mathbf{b} \end{bmatrix}$$
$$\begin{bmatrix} \mathbf{t} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix}$$

Daje to prosty liniowy system, w którym można znaleźć i wyizolować tib. Gdy już to zrobisz, możesz łatwo obliczyć macierz M, która konwertuje z przestrzeni stycznej do przestrzeni obiektu:

$$\mathbf{M} = \begin{bmatrix} \mathbf{t} & \mathbf{b} & \mathbf{n} \end{bmatrix}$$

Tutaj t, b i n są wektorami kolumn. Oczywiście macierz odwrotna wykona odwrotną transformację. Przekształci się z przestrzeni obiektu na przestrzeń styczną. Jest to zgodne z definicją odwrotności podaną w części 3. Jeśli macierz jest matrycą obrotu, odwrotność jest dość łatwa do obliczenia, ponieważ jest równa transpozycji. W ten sposób można uzyskać bardziej zoptymalizowaną odwrotność. Gdy już to zrobisz, możesz użyć możliwości sprzętowych do obliczenia jednego produktu na jeden wierzchołek. Ta funkcja jest zwykle określana jako funkcja bumpmapping dot3, chociaż może być używana do wszystkiego, co wymaga produktu z jedną kropką na piksel.

Emisja

Niektóre obiekty mogą nie tylko otrzymywać światło, ale także emitować światło (bez uwzględnienia refleksji). Na przykład neon może otrzymywać światło, ale emituje również światło. Powoduje to dodanie dodatkowego koloru do całego modelu. Podobnie jak w przypadku poprzednich metod, można również wygenerować mapę emisji, która określa, ile kolorów jest emitowanych na teksel. To nie jest bardzo popularna mapa, ale jest tam, jeśli jej potrzebujesz. dodać .5 do każdego, jako zostało zrobione, aby mapa kostki była normalna.