Teoretické základy 3D grafických akcelerátorů

Trocha teorie pro grafické karty

Tento článek navazuje na přehled funkcí grafických karet z minulého Chipu a snaží se vysvětlit teoretické základy, které je k jejich pochopení dobré znát

Phongův osvětlovací model

V tomto textu budeme používat pojmy vektor (popisuje nejen velikost, ale i směr v prostoru, bude označován tučným písmem, např. R) a skalární součin. Skalární součin dvou vektorů je reálné číslo, které říká, jak jsou tyto vektory zorientovány – skalární součin dvou kolmých vektorů je roven nule, zatímco u dvou vektorů rovnoběžných je nenulové číslo. Čím shodnější je směr dvou vektorů, tím větší je jejich skalární součin.

Phongův osvětlovací model vychází z následujících faktů (viz obrázek). Světelný zdroj osvětluje vyšetřovaný bod a směr k němu můžeme vyjádřit jako vektor L. Ten vektor získáme tak, že odečteme souřadnice světelného zdroje a souřadnice vyšetřovaného bodu. Nejdůležitější při výpočtu osvětlování je kolmice k povrchu (normála, normálový vektor) označená jako N. Dalším vektorem je odražený paprsek R, který získáme snadno z vektoru N a L, a V je vektor mířící směrem k pozorovateli.

Phongův osvětlovací model vypočítává intenzitu světla, která se odrazí z bodu do oka pozorovatele, a předpokládá, že toto světlo lze rozdělit na tři základní složky, na složku difuzní, zrcadlovou a ambientní. Nejjednodušší případy odrazu světla jsou zároveň případy mezní. Při úplném difuzním odrazu se světelná energie přicházející z libovolného směru rovnoměrně rozptýlí do všech směrů. Tuto vlastnost mají tzv. lambertovské povrchy, v nichž se proto vůbec nic neodráží. Jejich nejvěrnějším příkladem v realitě je prachobyčejný kbelík z umělé hmoty či křída. Tento model se velice snadno implementuje, a proto většina objektů v počátcích počítačové grafiky vypadala jako z plastiku.

Druhý případ odrazu, který se snadno simuluje, je ideální zrcadlový odraz (specular reflection). Zde se přicházející energie odrazí přesně pod úhlem, pod kterým na povrch dopadla. Pokud zjednodušíme záření na jediný paprsek, pak zrcadlový odraz odpovídá situaci, kdy se paprsek odrazí přesně pod úhlem, pod kterým dopadl. Část světla se může samozřejmě pohltit, a tak je odražený paprsek oslabený. Difuzní odraz odpovídá situaci, kdy se paprsek odrazí do všech stran rovnoměrně – vznikne tedy jakýsi trs paprsků.

Ambientní složka reprezentuje tzv. světelný šum, který je konstantní pro celou scénu, a je chápána jako světlo, které vzniklo mnohonásobným odrazem mezi objekty a na částečkách vzduchu, vodní páry, prachu atd.

Fyzikální modely vypočítávají osvětlení daleko přesněji. Phongův osvětlovací model je empirický a empiričnost tkví právě v onom rozdělení světla na tři nezávislé složky. Ve skutečnosti je samozřejmě světlo jedno jediné a uvedené zjednodušení je v podstatě kombinací mezních případů odrazu. Podívejme se nyní, jak se barva bodu vypočítá.

Označme intenzitu světla, které se odráží k pozorovateli, I, difuzní příspěvek I_D, zrcadlový I_s a ambientní I_A. Intenzita světelného zdroje bude označena jako I_L a ambientní šum v celé scéně bude označen I_{amb}

Difuzní složka osvětlení se vypočítá podle vztahu

$I_D = r_d\, I_L\, (N.L)$,

kde r_d říká, jak značně povrch odráží difuzní složku světla. Je-li tento koeficient roven jedné, odráží povrch úplně, nulová hodnota naopak říká, že neodráží vůbec. Člen v závorce je skalární součin vektorů kolmice k povrchu a směru ke světlu. Skalární součin je nulový pro kolmé vektory a největší pro vektory, které míří stejným směrem. Ze vztahu vyplývá, že se difuzní složka odráží pouze v závislosti na poloze světelného zdroje a není závislá na tom, odkud objekt sledujeme.

Zrcadlová složka se vypočítá jako

$I_Z = r_Z\, I_L\, (V.L)^h$,

kde r_Z je koeficient odrazu zrcadlové složky. Člen v závorce je opět skalární součin dvou vektorů a vidíme, že tato složka je závislá na poloze pozorovatele. Zrcadlový odraz bude největší, když se budeme dívat ve směru odraženého paprsku. Člen v závorce bývá zvykem umocňovat na nějaké reálné číslo h. Zrcadlová složka způsobuje na objektu “prasátko” a h určuje jeho poloměr – čím je větší, tím je “prasátko” menší.

Ambientní vztah je výpočetně nejjednodušší

$I_A = r_A\, I_{amb}$.

I_{amb} popisuje množství světelného šumu ve scéně, a jak jsme řekli, je ve scéně konstantní.

Je důležité si uvědomit, že výpočet se provádí postupně pro jednotlivé barevné kanály (červený, zelený a modrý). Pokud tedy budeme na zcela červený objekt svítit modrým světlem, bude výsledná barva černá.

Stínování

Stínováním (shading) se rozumí výpočet barevných přechodů způsobených nestejným osvětlením různých částí objektu. Nebylo by nijak důležité se stínováním explicitně zabývat, kdyby s ním nebyl spojen

jeden velký problém. Pokud je naším úkolem zobrazit nějaký objekt osvětlený několika světelnými zdroji pomocí Phongova osvětlovacího modelu, musíme nejdříve vypočítat body, které se promítnou do určitých pixelů obrazovky, pro ně najít odpovídající vzory na povrchu objektu, k nim určit normálové vektory a na každý takovýto bod aplikovat Phongův osvětlovací model. Právě poslední zmíněný krok je nejvíce výpočetně náročný. Výpočetní náročnost je vždy důvodem k hledání jednodušších algoritmů a v tomto případě se nejčastěji používá právě stínování.

Podstata stínování spočívá v tom, že se výpočet osvětlení provádí pouze pro některé důležité body objektu a tyto hodnoty se nějakým způsobem distribuují na jeho zbývající části. V počítačové grafice se dnes používají v podstatě čtyři druhy stínování:

* konstantní stínování (flat shading),
* Gouraudovo stínování (Gouraud shading) neboli interpolace barvy,
* Phongovo stínování (Phong shading) neboli interpolace normály,
* Výpočet osvětlení na úrovni pixelu.

Uvažujme dále trojúhelník (pro rychlé zobrazování se geometrie objektů aproximuje sítí trojúhelníků) zadaný vrcholy a předpokládejme, že v nich známe normálové vektory.

Konstantní stínování přiřazuje celému trojúhelníku jedinou barvu na základě hodnoty v jednom z vrcholů, takže se výpočet osvětlení provádí pro každý trojúhelník jen jednou. Tento postup je výpočetně nejméně náročný, ale na přechodech trojúhelníků se objevují výrazné hrany a objekt získává nerealistický ploškový vzhled.

U Gouraudova stínování se barva vypočítává ve všech třech vrcholech a získané hodnoty se lineárně interpolují. Výsledkem je daleko realističtější vzhled s vyhlazenými přechody mezi sousedními trojúhelníky. Gouraudovo stínování má jednu nevýhodu, pro kterou se někdy používá stínování Phongovo. Gouraudovým stínováním vypočítáme barvu ve dvou vrcholech a interpolujeme ji. Podle Phongova modelu by, vzhledem k poloze světla a pozorovatele, uprostřed mezi uzly mělo dojít k odlesku (zrcadlovému odrazu). Vzhledem k tomu, že barvu nepočítáme pro každý bod, ale jen interpolujeme, k odlesku nedojde.

Tuto závadu částečně řeší Phongovo stínování. Protože nejnáročnější bývá výpočet normálového vektoru, pokouší tento problém řešit tak, že interpoluje normálu mezi dvěma uzly a toto přiblížení používá k výpočtu osvětlení. Je nutné podotknout, že Phongovo stínování je výpočetně daleko náročnější nežli stínování Gouraudovo především proto, že se rovnice Phongova osvětlovacího modelu vypočítávají pro všechny body objektu.

Stínování na úrovni pixelu je vlastně výpočet přesného osvětlení pro každý pixel. Nevýhodou je, že se musí někde udržovat normálové vektory pro každý bod.

Komprese textur S3

V originálním algoritmu musí být textura reprezentována v šestnáctibitové reprezentaci RGB565, tj. pět bitů pro červenou složku, šest pro zelenou a pět pro modrou. Zelená složka má k dispozici nejvíce bitů, a je tedy reprezentována nejpřesněji, protože lidské oko je na tuto barvu nejvíce citlivé. Varianty této kompresní metody, například od 3dfx, dovolují i jiné reprezentace, princip je však stejný, i když jméno je jiné.

Prvním krokem komprese (viz obrázek) je rozdělení obrázku na dlaždice skládající se ze 4 x 4 pixelů. Každý pixel je potom zakódován pouze do dvou bitů následujícím způsobem:

V každém čtverci se naleznou dvě nejtypičtější barvy. Obě se pak zakódují jako dvoubitová informace, jedna jako 11 a druhá jako 00. Pro obě tyto barvy se uschová i jejich přesná hodnota RGB v šestnáctibitovém modu RGB565. Ostatní pixely z tohoto čtverce se zakódují jako dvoubitová informace podle své barvy tak, že se barva mezi extrémní 00 a 11 lineárně interpoluje. Právě lineární interpolace zavádí do tohoto algoritmu ztrátu informace. Pouze dva pixely z šestnácti jsou reprezentovány přesně a zbývající jsou odhadnuty. Samozřejmě je zde i podstatná ztráta barevné informace, protože umožňujeme pouze čtyři barvy v každém čtverci. Jelikož však odhadujeme vždy pouze čtrnáct pixelů, je tato ztráta vizuálně přijatelná.

Jak je to s velikostí záznamu obrazu? V původním čtverci bylo 4 x 4 šestnáctibitových pixelů, celkem tedy 4 x 4 x 16 = 256 bitů. Po kompresi tak získáme 2 x 16 bitů pro extrémní hodnoty a 16 x 2 bity pro pixely vypočítávané, celkem tedy 64 bitů. Výsledek je ze zřejmých příčin dělitelný osmi. Výsledný obrázek má tedy pouze neuvěřitelných 25 % velikosti obrázku původního.

Dekomprese je snadná. Pro každý čtverec 4 x 4 pixelů se nejprve přečte tabulka s přesně reprezentovanými barvami a barvy pro všechny dvojice bitů mezi 00 a 11 se z nich určí lineární interpolací. Celou tuto záležitost lze samozřejmě snadno řešit hardwarově. Celá kompresní metoda by samozřejmě nebyla k ničemu, kdyby programátoři neměli možnosti ji využít v existujících standardech pro rychlé zobrazování. Tato technika byla přijata jak do DirectX, tak do OpenGL.

Nutno říci, že kvalita komprimovaných obrázků je velmi vysoká. Na fotografie či na obrázky obsahující poměrně velký šum je vynikající. Nepodařilo se mi nikde najít podstatnou vizuální degradaci, i když jsem obrázky zkoumal dosti pečlivě. Pokud bychom změřili přesnou chybu originálu a komprimovaného obrazu, například RMS, patrně by se jednalo o překvapivě vysoké číslo. Lidské vnímání je však tolerantní k šumu a právě systematické chyby jsou touto metodou zaváděny do obrazu minimálně. Metoda samozřejmě vnáší určitou degradaci do obrazů obsahujících tenké hrany, všechny pravidelné vzory a samozřejmě obrázky syntetické. Interpolací může dojít k tomu, že například čára jdoucí přes různé dlaždice má na nich vypočítánu

i různou barvu a že tak dojde k jejímu poničení. Tento jev demonstruje obrázek. Při kritizování této metody je však nutné mít na paměti, k čemu je určena. Jejím cílem není fungovat jako standard pro reprezentaci obrazů, ale slouží pro mapovaní textur. Na pohybujícím se objektu tento defekt patrně nezaregistrujeme.

Hrbolatá textura (bump texture)

V počítačové grafice hraje klíčovou úlohu kolmice k povrchu objektu (normála, normálový vektor), která má zcela zásadní vliv při výpočtu osvětlení vyšetřovaného bodu. Z polohy pozorovatele, z polohy světel a ze směru kolmice k povrchu v určitém bodě lze vypočítat barvu tohoto bodu, vržené stíny aj. Změnou směru normálového vektoru získáme tzv. hrbolatou texturu (bump texture, bump mapping). Jak je asi zřejmé, čím je povrch objektu hrbolatější, tím komplikovaněji se mění směr normály mezi dvěma sousední body.

Hrbolatou texturu tedy získáme tak, že při výpočtu osvětlení objektu změníme směr normály k povrchu, jako by se jednalo o hrbolatý povrch. Textura tedy v tomto případě neurčuje ani barvu, ani průhlednost, ale členitost povrchu. Hrbolaté textury se prozradí na okrajích objektů, na které jsou naneseny. Například koule, která vypadá jako divoce členitá, nemá obrys odpovídající drsnému povrchu, ale hladce kruhový, jako by byla zcela hladká. To je patrné i na stínu, který je objektem vrhán.

Podkladem pro hrbolatou texturu může být obyčejný obrázek, jehož odstíny chápeme jako popis Z souřadnic (podobně jako vrstevnice na mapě). Někdy se takovýmto texturám říká třírozměrná vytlačená textura (emboss texture). Hrbolaté textury jsou velice populární a pro svou jednoduchost se velmi často používají. Jejich nevýhodou je, že při velikém zvětšení působí nepřirozeně. Jejich výhodou naopak je, že jde o efektivní a jednoduchou metodu, která je v současné době standardní výbavou hardwarových akcelerátorů.

Antialiasing

Počítačová grafika pracuje s diskrétním obrazem, který je složen z pixelů. Dochází tedy k tomu, že se nenulovým ploškám – pixelům – přiřazuje konstantní barva obyčejně na základě hodnoty jediného bodu z reálného světa. Tento proces (vzorkování) zavádí nepřesnosti, způsobující jevy známé jako alias, k nimž dochází zejména u objektů, které jsou velikostí srovnatelné s pixelem. Tyto objekty, pokud se pohybují, jsou jednou vidět, a podruhé ne. Dochází také k “zubatosti” hran či k přerušování tenkých čar apod. Techniky, které se pokoušejí alias odstranit, se jmenují antialiasing a mohou pracovat v podstatě ve dvou režimech – s objekty nebo s celou scénou. Antialiasing objektů – čar, bodů a trojúhelníků – má základní nevýhodu v tom, že musí udržovat informace o scéně. Konkrétně se objekty musí zobrazovat odpředu ve směru pozorovatele. To je mimořádně nepříjemné, a proto se antialiasing objektů používá zcela výjimečně.

Celoobrazovkový antialiasing (Full Screen Antialiasing – FSSA) by se vlastně měl jmenovat antialiasing scény. Na rozdíl od antialiasingu objektů pracuje s celou scénou, se všemi objekty najednou (tedy ne s celou obrazovkou, ke které patří například i rámečky oken).

O co v této metodě jde? Alias vzniká podvzorkováním spojité informace, tedy nedostatkem informace. Jediná cesta, jak ho odstranit či snížit, je dodat informaci do signálu. Toho lze docílit pouze přesnějším zobrazováním, což je samozřejmě časově náročné. Pro FSAA se používají dvě techniky, které jsou založeny na tzv. vzorkování s vyšší frekvencí (supersampling).

První spočívá v zobrazení celé scény ve vyšším rozlišení a v jejím následném zmenšení. Obyčejně se používá rozlišení, které je dvakrát (4 x, 8 x, ...) vyšší, protože se zmenšení provádí jednoduše průměrováním skupiny pixelů. Nevýhodou této techniky je, že se pravidelné vzory – alias – stejně objeví, ale na vyšších frekvencích. Touto cestou jde například NVIDIA.

Jinou technikou je použití náhodného vzorkování s vyšší frekvencí (stochastic/random supersampling). Princip tkví v tom, že se scéna vykreslí z jednoho pohledu v požadovaném rozlišení několikrát, avšak pokaždé se s ní malinko pohne. Posunutí však musí být minimální, typicky uvnitř jednoho pixelu (proto se jí někdy nepřesně říká subsampling). Pro posunutí se používá technika zvaná roztřesení (jittering). Scéna se tedy zobrazí několikrát, ale tato informace se musí někde uchovat. Právě pro tuto techniku se výtečně hodí akumulační buffer (u 3dfx se mu z nepochopitelných příčin říká T-buffer). V této paměti se zobrazovaná informace jednoduše posbírá, spočítá se průměr a nakonec se vše zobrazí. Výsledkem je kvalitnější obraz. Vynikající na této téměř patnáct let staré metodě je, že převádí alias na šum, který je vizuálně mnohem méně patrný než pravidelné vzory. Nevýhodou je, že se scéna musí zobrazit několikrát. Určitě však lze v hardwarových akcelerátorech využít faktu, že data jsou již uvnitř, a tak lze vše výtečně urychlit.

Pokud obě zmíněné metody srovnáme, první z nich je obyčejně rychlejší, i když neposkytuje tak hezké výsledky jako druhá.

Bedřich Beneš
beda@campus.ccm.itesm.mx