Algoritmy počítačové grafiky

P.Palát, D.Renát, H.Seifrt, J.Zapletal
SPŠE Brno, SPŠE Brno,
Gymnázium Pobořany, Gymnázium Vídeňská 47 Brno
harry_x@babylon5.cz, xardas.mail@seznam.cz,
janseifrt@seznam.cz, yaplik@atlas.cz

Abstrakt

Shrnutí základních algoritmů počítačové grafiky ve 3D jako je zbuffer,raytracing, trasformace objektů ve 3D, modely osvětlování.

1 Úvod

Počítačová grafika tvoří poměrně rychle se rozvíjející odvětví informatiky, své uplatnění tradičně nachází v různých CAD systémech, ve vědě (např. při vizualizacích fyzikálních či medicínských dat). Mimojiné také nachází v oblasti PC, zejména v počítačových hrách.

2 Algoritmy počítačové grafiky

Počítačová grafika obsahuje velké množství odvětví a tak i škála používaných algoritmů je poměrně široká. My se zaměříme na dvě skupiny algoritmů - vizualizační a osvětlovací.

2.1 Rendering

Rendering neboli vizualizace scény je proces, v němž převádíme zdrojová data, která popisují scénu, na obrazová. Objekty scény mohou být zadány různými způsoby:

- Síť trojúhelníků (či obecně n-úhelníků - v praxi se však používají většinou jen trojúhelníky a čtyřúhelníky)
- Parametrické plochy (NURBS, Beziérovy plochy)
- Implicitně zadané plochy - Metaballs [3]
- Volumetrická data

Nejčastěji používanou metodou zadávání dat je ta první zmíněná - tedy síť trojúhelníků, jejíž velkou výhodou je jednoduchost a vhodnost pro hardwarovou implementaci. Pokud použijeme jiný způsob zadávání dat, tak jej většinou převádíme právě na síť trojúhelníků - teselujeme (u NURBS křivek/ploch). Použití parametrických ploch má velkou výhodu v tom, že za běhu (resp. teselace) můžeme určit jemnost výsledné trojúhelníkové

sítě, což se hodí např. pro různé algoritmy pracující s různými úrovněmi detailnosti objektu (LOD - level of detail). Nevýhodou je právě opět nutnost softwarové teselace (což můžeme samozřejmě částečně řešit předpočítáním).

Důležitým aspektem 3D grafiky je možnost vyjádřit transformace v prostoru pomocí matic (využitím základních prvků lineární algebry). Maticí můžeme vyjádřit každou lineární transformaci - což je např. změna velikosti, rotace, ale i např. perspektivní korekce. V praxi však potřebujeme používat i nelinéární operace jako je např. posun. Díký tomu se v 3D grafice na popis nepoužívá tří dimenzí, ale čtyř. U vstupních dat čtvrtá souřadnice určuje, zda se jedná o bod nebo vektor. Vektor má čtvrtou souřadnici (w) nastavenou na 0 - tudíž se na něj nevztahuje posun. Naopak bod ji má nastavenou na w=1

Samotných algoritmů na renderování je poměrně široká řada. Nejznámějším a asi nejpoužívanějším algoritmem je zbuffer, kdy máme rasterový obraz, do něhož promítáme jednotlivé objekty (rasterizování se provádí v prostoru zařízení - device space - tudíž již je započítána transformace daná kamerou, perspektivní projekcí,etc.). U každého bodu si rasterizer pamatuje nejenom jejich barvu, ale taktéž informace o Z souřadnici. Pak v případě, že na jiném prostoru má rasterizovat jiný bod, tak porovná jejich Z hodnotu a rozhodne se podle ní, zda bod přepíše či nikoliv. Toto je taktéž místo, kde se dá rendering pomocí zbufferu optimalizovat pomocí zsortingu - to má však význam pouze tehdy, když je proces fillování bodu náročnější operací - což je typicky v případě, že používáme pixelové shadery (což jsou krátké prográmky pro grafický procesor, jejímž výstupem je barva a hloubka pixelu). Tento renderovací algoritmus je implementován prakticky na každém 3D grafickém akcelerátoru z důvodů snadné realizace v HW.

Další metodou renderování je ray tracing - či česky sledování paprsku. Tato technika je určena pro fotorealistické renderování, má velmi kvalitní výstup, ale je velmi pomalá. Základem techniky je, že z každého bodu obrazovky vedeme paprsek a sledujeme, jestli narazí do nějakého objektu. V případě, že ano, tak vytvoříme další paprsky, které povedou z daného bodu do ohniska každého světla a otestujeme, zda paprsek koliduje s nějakým dalším objektem. Tím zjistíme, zda se bod nachází ve stínu či nikoliv. V případě, že se nenachází v stínu, tak spočítáme příspěvek daného světla. Pokud je povrch objektu reflektivní, tak vytvoříme další paprsky, které budou simulovat jeho odrazivost.

Výhodou ray tracingu je např. možnost simulování lomu paprsku při přechodu mezi prostředími s jiným indexem lomu. Nevýhodou je extrémně velká náročnost v případě požadavků na reálný výsledek - proto se používá menší množství paprsků - pak ale vzniká problém, že je scéna nedostatečně osvětlena. To se řeší např. jiným vypočítáváním útlumu světla - podle fyzikálních zákonu je útlum světla úměrný čtverci vzdálenosti, v praxi se v 3D grafice používá kombinace linéární a kvadratické závislosti.

Existují i různé modifikace ray tracingu - např. použití fotonových map (photon maps) - kdy z zdrojů světla vrháme fotony, které se zachycují na jednotlivých objektech (čímž se nám zjednodušší výpočet osvětlení při samotném vrhání paprsků).

2.2 Osvětlování

Osvětlování patří mezi nejdůležitější aspekty v renderování 3D grafiky. Je to právě osvětlování, které vytváří realistický dojem scény a vytváří dojem plastičnosti těles. Všechny osvětlovací algoritmy jsou jen zjednodušením fyzikálních modelů (které jsou na výpočet přílis náročné). Liší se kvalitou, ale také rychlostí. Nejdůležitější algoritmy pro osvětlování:

- Phongův osvětlovací model

- Blinnův osvětlovací model
- Radiosita
- Fotonové mapy, Monte Carlo ray tracing [4]

Phongův osvětlovací model je jedním z nejstarších aproximací osvětlování. Jedná se o tzv. lokalní osvětlovací model, tudíž není schopen zahrnout např. odrazy objektů a své výpočty provádí pouze pomocí normály objektu v daném bodě a vektoru světla. Rozlišuje několik prvků v osvětlení:

- Difúzní
- Spekulární
- Ambientní

Ambientní člen popisuje nesměrové osvětlení z vnějších zdrojů, z nichž není možné odkud přicházejí.
Difúzní člen charakterizuje světlo odražené rovnoměrně do všech směrů. V Phongově modelu je jeho výpočet dán vztahem:
$I_d = I_l * r_d * \cos\alpha$
α úhel mezi vektorem normály v daném bodě a směrovým vektorem k světlu. Kosinus úhlu získáme pomocí známého vztahu:
$\cos\alpha = \frac{n.l}{|l||n|}$
V případě, že máme vektory n a l normalizovány (což není problém zajistit), tak se výpočet zjednoduší na pouhý skalárního součin obou vektorů. I_d je difúzní koeficient světla, r_d je dif. koeficient materiálu.

Spekulární člen charakterizuje, jak je objekt lesklý. Je závislý na vektoru odrazu a směrovém vektoru k pozorovateli. Vztah pro výpočet je:
$I_s = I_s * r_s * \cos^h\beta)$
I_s je spek. koeficient světla, r_s je spek. koeficient materiálu. Úhel β získáme opět pomocí skalárního součinu vektorů. Zde je ovšem problém vypočítat vektor odrazu (přesněji řečeno, výpočet vektoru odrazu je výpočetně náročný). Z tohoto důvodu vznikl Blinnův osvětlovací model.

2.3 Blinnův osvětlovací model

Blinnův model je shodný s Phongovým osvětlovacím modelem až na výpočet spekulární složky. Místo vektoru odrazu požívá tzv. halfway vektor - což je normalizovaný vektor mezi vzniklý součtem vektoru k světlu a normály (taktéž normalizovaných). Ten pak používá místo úhlu odrazu. Metoda je méně přesná, ale velmi rychlá, proto se stala populární při implementaci tohoto osvětlovacího modelu v T&L jednotkách grafických karet.

Tyto osvětlovací modely mohou být doplněny o další prvky - např. útlum světla (většinou se používá kombinaci lineární a kvadratické závislosti) či třeba Fresnellův člen.

Velkou nevýhodou lokálních osvětlovacích metod je nesnadné přídávání některých grafických efektů, např. zobrazování stínů.

2.4 Radiositní metoda

Radiosita je metodou globálního osvětlování, tudíž podává nesrovnatelně lepší výsledky než lokální osvětlovací metody. Daní za to je ovšem výpočetní náročnost a z toho též plyne nevhodnost na implementaci a použití v realtime grafice.

Metoda je založena na rozdělení každého objektu na síť čtverců. Na každém čtverci je vypočítáno jeho osvětlení z ostatních čtverců, na něž vidí on sám (samozřejmě je započítáván i útlum světla). Tak je to provedeno pro všechny čtverce v scéně. Poté je proces opakován, většinou několikrát. Díky tomu radiosity velmi dobře zachytí vzájemné odrazy ve scéně a vyrenderovaná scéna vypadá velmi reálně.

Hlavní nevýhodou je právě ona náročnost na výpočet, který není možné většinou provádět v reálném čase. Bylo ovšem vytvořeno několik technik, které použití v reálném čase více či méně umožňují (rozhodně ale ne natolik, aby bylo možno radiositou plně nahradit klasické osvětlovací modely). První takovou technikou je využití grafických akcelerátorů při výpočtu vlivu ostatních části sítě čtverců na právě zpracovávaný čtverec (zde se dá dobře využít i programovatelnosti pipeline grafické karty). Další metodou je částečná aktualizace radiositní mapy v reálném čase.

Radiosita se ovšem nejčastěji používá na výpočet statického osvětlování - tzv. light map.

3 Shrnutí

Počítačová grafika představuje rychlé rozvíjející odvětví aplikované informatiky, které nachází uplatnění v velkém spektru vědeckých a technických činností.

Poděkování

Účastníci miniprojektů by tímto chtěli poděkovat našemu supervisorovi Ing. Tomáši Oberhuberovi,organizátorům fyzikálního týdne a Fakultě jaderné a fyzikálně inženýrské ČVUT a autorův použitého software (Linux,OpenOffice,LaTeX,Povray)

Reference

[1] Dalibor Martišek. *Matematické principy grafických systémů*

[2] Tomas Akenine-Möller, Eric Haines. *Real-time Rendering*

[3] Blinn, James F. *A Generalization of Algebraic Surface Drawing, ACM Transactions on Graphics*

[4] P. Dutre, E. Lafortune, and Y.D. Willems. *Monte carlo light tracing with direct computation of pixel intensities.*