Počítačová grafika III (NPGR010)
12. přednáška: Globální osvětlení ve filmové produkci : Irradiance caching a Bodové globální osvětlení

Jan Tichý
20. prosince, 2012

Ve filmové produkci se nejvíce používají pro globální osvětlení dva algoritmy - Irradiance caching ("úschova ozáření") a Point-based Global Illumination ("bodové globální osvětlení").

1 Irradiance caching

Tento algoritmus je navržen k tomu, aby urychloval právě takové výpočty, jako Final Gathering u Photon Mapping technik. Tedy k nějakému místu na ploše chceme spočítat osvětlení, jehož hodnotu získáme vrháním mnoha sekundárních paprsků. Tento výpočet je ale hodně pomalý a cílem Irradiance caching je ho urychlit.

Metoda Irradiance caching je nekonzistentní, tedy nekonverguje v čase ke správnému výsledku. Dává ovšem velmi uvěřitelné výsledky v mnohem rychlejším čase oproti jiným metodám.

1.1 Základní popis algoritmu

Základní myšlenkou je výpočet neprovádět pro každý bod prostoru zvlášť, ale jenom na několika vybraných místech a ostatní body pouze interpolovat. Lze to dělat díky tomu, že ve scéně, která není modulovaná texturou a odrazem paprsků, pro každý bod zobrazujeme příchozí osvětlení přes celou hemisféru, což se v okolí bodu moc nemění. Pokud je v okolí bodu složitější geometrie, tak se mění rychleji, pokud ne, tak pomaleji. V místech se složitější geometrií tak budou body vybrány hustě, v místech s jednodušší geometrií řídce. Příchozí nepřímé osvětlení se mění velice pomalu i na ploše s texturou, kdy se v cache uschovává irradiance ještě před násobením texturou. Tím pádem caching funguje dobře i povrchu s texturou (viz. obrázek 1).

Algoritmus je navržen pro fungování s difuzní komponentou osvětlení, protože ta je pohledově nezávislá. Odražená radiance je dána jako irradiance * odrazivost povrchu / PI, což je ekvivalentní irradiance * BRDF (na difuzním povrchu nezávislá na směru), tedy bez směrových proměnných, takže je ve všech směrech konstatní.

$$Lo(p, \omega_o) = E(p) * \rho_d(p)/PI = E(p) * f_{r,d}(p)$$

Pokud si budeme pamatovat hodnotu příchozí irradiance ($E(P)$) v nějakém místě (p), tak pro jakýkoli směr (ω_o) pohledu získáme množství odražené radiance pouze pronásobením s texturou (odrazivostí). Když máme potom v cache irradianci pro některé body, tak z nich pro dotazovaný bod vyzvedneme irradianci a vynásobíme hodnotou textury v daném místě.

Algoritmus ma klasické cachovací schéma - líné vyhodnocování. Tedy hodnota osvětlení se počítá, až když je potřeba.

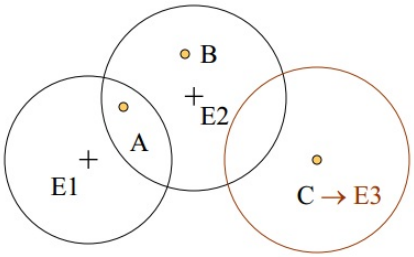
Podíváme se tedy do cache. Pokud jsou v okolí nějaké vhodné body, tak interpoluji, jinak počítám (draze) novou hodnotu a uložím ji do cache.

1.1.1 Příklad

Na obrázku 2 je příklad obsahu cache. Hodnoty E1 a E2 jsou již uloženy. Hodnotu A můžeme rychle interpolovat z E1 a E2. Hodnotu B můžeme rychle extrapolovat z E2. Hodnota C je ale mimo

Obrázek 1: *Nepřímé osvětlení se mění pomalu na plochách*

radius E1 i E2, tedy ji musíme pomalu vyhodnotit a uložit do cache jako novou hodnotu E3.



Obrázek 2: *Příklad obsahu cache*

1.1.2 Pseudokód algoritmu

```
GetIrradiance(p):
        Color E = InterpolateFromCache(p);
        if( E == invalid )
                E = SampleHemisphere(p);
                InsertIntoCache(E, p);
        return E;
```

1.2 Výpočet nepřímé irradiance

Když v cache nejsou záznamy, ze kterých bychom mohli interpolovat, musíme vytvořit nový záznam vrháním sekundárních paprsků.

Z pseudokódu nás tedy teď zajímá řádek:

```
E = SampleHemisphere(p);
```

Vrháme řádově 500 až 5000 paprsků (záleží na uživateli).

Ze scény vyjmeme zdroje světla, jelikož děláme pouze nepřímé osvětlení.

Vrhneme paprsek někam na hemisféře a v průniku nějakým způsobem odhadneme radianci. Buď můžeme dát dotaz do fotonové mapy nebo rekurzivně opakovat irradiance caching (do jiných cache). Pak výsledky z různých paprsků zprůměrujeme.

Pro vrhání sekundárních paprsků typicky používáme Monte Carlo se stratifikací podle θ a ϕ.

Irradianci v bodě p spočítáme podle :

$E(p) = \int L_i(p, \omega_i) \cos\theta_i d\omega_i$

Obecná forma stratifikovaného estimátoru

$E(p) \approx \frac{1}{MN} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} \frac{f(\theta_{j,k}, \phi_{j,k})}{p(\theta_{j,k}, \phi_{j,k})}$

Pro výpočet irradiance je integrand:

$L(\theta, \phi) cos\theta$

Hustota pravděpodobnosti (PDF):

$p(\theta, \phi) = \frac{cos\theta}{\pi}$

Po dosazení je estimátor irradiance:

$E(p) \approx \frac{\pi}{MN} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L_{j,k}$

Kde $L_{j,k}$ je vzorek radiance ze směru $(\theta_{j,k}, \phi_{j,k}) = (arccos\sqrt{1 - \frac{j+\zeta^1_{j,k}}{M}}, 2\pi\frac{k+\zeta^2_{j,k}}{N})$

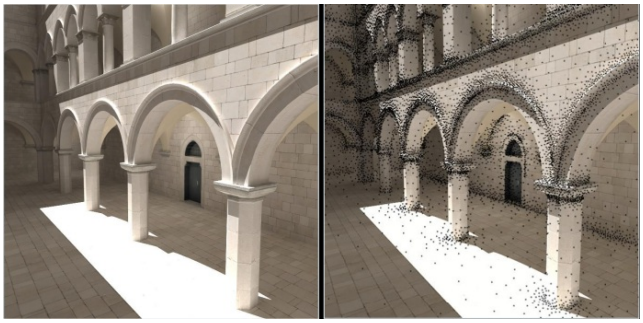
M, N jsou počty dílů podél θ, ϕ

$\zeta^1_{j,k}$, $\zeta^2_{j,k}$ jsou náhodná čísla z R(0,1)

1.3 Adaptivní odhad poloměru platnosti záznamu

Chceme, aby cachování bylo adaptivní. Tedy tam, kde by se osvětlení mohlo měnit rychle, tak tam chceme záznamů hodně. A naopak.

Osvětlení se bude měnit rychle tam, kde se nachází v okolí nějaká geometrie, na rovné ploše se nebude osvětlení rychle měnit (viz. obrázek 1). Jestli je v okolí bodu nějaká geometrie, zjistíme jednoduše - při vzorkování hemisféry u každého vrhnutého paprsku známe i jeho vzdálenost. Spočítáme pak průměrnou vzdálenost, kde se nacházejí objekty v okolí daného místa a to nám dá poloměr (radius), kde se záznam smí ještě použít.



Obrázek 3: *Hustota bodů podle složitosti geometrie v okolí*

1.4 Interpolace z cache

Irradianci v bodě p spočítáme jako vážený průměr irradiancí v záznamech cache, které se nachází v okolí bodu p.

Vážený průměr:

$E(p) = \frac{\sum_{i \in S(p)} E_i(p)\omega_i(p)}{\sum_{i \in S(p)} \omega_i(p)}$

Množina záznamů použitelných pro interpolaci:

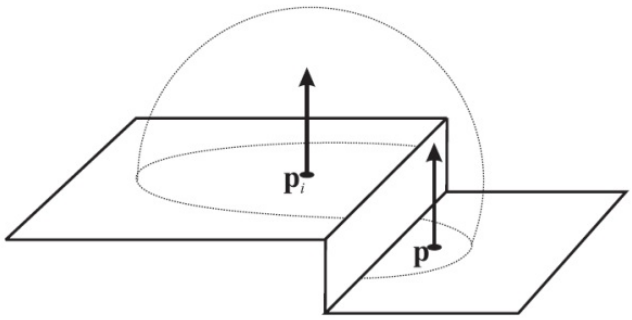
$S(p) = \{i; \omega_i(p) > 0\}$

Váhová funkce:

$\omega_i(p) = 1 - \kappa \max\left\{\frac{\|p - p_i\|}{\text{clamp}(2R_i, R_{min}, R_{max})}, \frac{\sqrt{1 - n \cdot n_i}}{\sqrt{1 - \cos 10^\circ}}\right\}$

Zde p je bod, kde chceme interpolovat, p_i je bod v cache, κ je uživatelem zadaný parametr ovlivňující přesnost interpolace. Funkce clamp omezuje zeshora a zespoda. R_{min} je typicky 1.5x násobek velikosti pixelu promítnutého do bodu p a $R_{max} = 10 * R_{min}$. Člen $\frac{\sqrt{1 - n \cdot n_i}}{\sqrt{1 - \cos 10^\circ}}$ je heuristický. Tento člen zajišťuje penalizaci při interpolaci na zakřiveném povrchu. Jakmile bude rozdíl normál n (normála bodu p) a n_i (normála bodu p_i) větší než 10 stupňů, bude tento člen větší než 1 a záznam se pro interpolaci nepoužije.

Při implementaci se přidávají ještě další heuristické podmínky, které nejsou pokryty ve vzorci. Jedná se například o "behind test", který kontroluje, jestli neinterpolujeme bod, který je za bodem v cache (viz. Obrázek 4). Tento test potřebujeme kvůli tomu, že ačkoli mezi body p a p_i je geometrie (schod), tak o ní nevíme, protože jsme na ni nenarazili při hledání geometrie přes hemisféru nad bodem p_i. Naopak z bodu p bychom geometrii detekovali, tedy není potřeba žádný "infrontof test".



Obrázek 4: *Heuristika "behind test"*

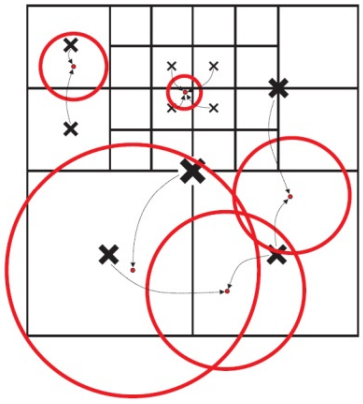
1.5 Datová struktura

Musíme zvolit vhodnou datovou strukturu pro Irradiance cache.

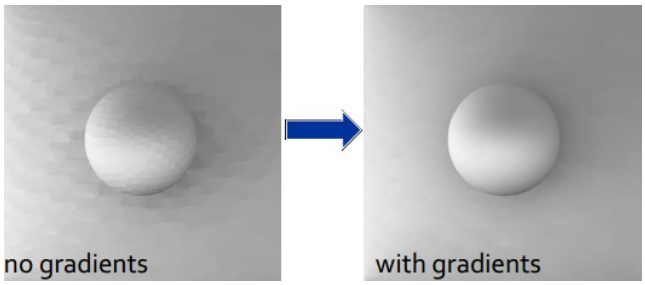
Máme záznam - bod ve scéně, normála, irradiance, poloměr použitelnosti hodnoty irradiance, gradienty (viz. další kapitola).

Do datové struktury budeme inkrementálně přidávat nové záznamy. Dále budeme chtít, aby nám datová struktura vrátila všechny koule (záznamy - bod a poloměr dává kouli), které obsahují dotazovaný bod v prostoru.

Pro tyto potřeby je nejvhodnější datovou strukturou Octree. Do každého uzlu stromu ukládáme všechy koule, které tento uzel pronikají. Máme tak vysokou redundanci záznamů, ale napomáhá to



Obrázek 5: *Octree*



Obrázek 6: *Motivace pro gradienty*

rychlejšímu vyhodnocení dotazu. Větší koule ukládáme ve vyšších patrech, menší v nižších. Při vyhledávání procházíme stromem od kořene do uzlu, kde se dotazovaný bod nachází, a po cestě sbíráme v každém uzlu, který navštívíme, informace o potenciálně proniknutých koulích. Průniky zadaného bodu s nalezenými koulemi vyhodnocujeme.

1.5.1 Pseudokód

```
procedure LookUpRecordsMR(p,n)
  node := root;
  while node != NULL do
   for all records i stored in node do
    if w(i,p) > 0 and (p(i) not in front of p) then
     Include record in S(p)
    end if
   end for
   node := child containing p
  end while
end procedure

w(i,p) je váhová funkce, p(i) je bod v zaznamu
```

1.6 Gradienty

Aby se zlepšila interpolace, tak použijeme gradienty. Na obrázku 6 je dobře vidět, jak dopadne interpolace bez gradientů. Použijeme tedy interpolace vyšších řádů. Celá interpolace funguje tak, že ve chvíli výpočtu osvětlení nějakého místa spočítáme nejen irradianci, ale spočítáme i jaký bude její gradient - tedy jak rychle se bude měnit hodnota irradiance při posouvání bodem. Uděláme tedy Taylorův rozvoj prvního řádu.

Počítají se dva typy gradientů - jeden je vůči rotaci, druhý je vůči translaci.

Rotační gradient

Rotační gradient je vektor, který odpovídá ose rotace povrchu, při které nastává největší změna irradiance. Velikost tohoto vektoru je derivace irradiance podle úhlu rotace podle této osy. Je-li dána libovolná osa rotace v, pak derivace irradiance podle úhlu rotace podél této osy je dána vektorovým součinem v a rotačního gradientu.

Translační gradient

Translační gradient je vektor ve směru největší změny irradiance. Jeho velikost je derivací irradiance v tomto směru. Derivace irradiance při pohybu nějakým daným směrem je dána skalárním součinem translačního gradientu s tímto směrem.

Gradienty můžeme odhadnout pomocí vzorce přímo v rámci vzorkování hemisféry. Když ji vzorkujeme, tak ji rozdělíme na buňky. Informaci, ze které víme, z jakého směru přichází radiance, můžeme využít nejen ke spočítání průměrné hodnoty irradiance v daném bodě, ale můžeme i odhadnout, jak se bude irradiance měnit při rotaci a translaci.

Upravená interpolace radiance s gradienty

$$E(p) = \frac{\sum_{i \in S(p)} E_i(p)\omega_i(p)}{\sum_{i \in S(p)} \omega_i(p)}$$

$$E_i(p) = E_i + (n_i \times n) \cdot \nabla_r E_i + (p - p_i) \cdot \nabla_t E_i$$

1.7 Ambient occlusion

Ambient occlusion je metoda stínování, která pro každý bod na povrchu objektu spočítá, kolik procent hemisféry nad tímto bodem je viditelných z okolního prostředí. Jinými slovy započítává tlumení světla zastíněním. Využívá se ve hrách a ve filmu.

Je efektivnější než globální osvětlení, ale i tak dává dobré výsledky. Definice odpovídá irradiance cachingu, kde příchozí světlo je nahrazeno funkcí viditelnosti V a tudíž můžeme použít caching (ambient occlusion je pohledově nezávislá veličina).

Definice: $A(p) = \frac{1}{\pi} \int_{H^+} V(p,\omega) \cos\theta d\omega$

Tedy "poměr nezastíněné a zastíněné části hemisféry".

1.8 Použití ve filmu

První celovečení film s globálním osvětlením spočítaným pomocí Irradiance caching je Shrek II. Od té doby tento algoritmus použilo studio PDI / Dreamworks na řadu dalších animovaných celovečerních filmů, například Madagaskar a Kung-fu panda, dokud nepřešli na bodové globální osvětlení.

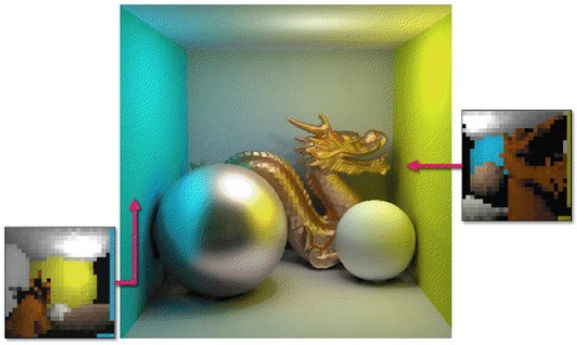
2 Bodové globální osvětlení

2.1 Idea algoritmu

Základní ideu algoritmu bodového globálního osvětlení si můžeme představit tak, že do každého bodu scény dáme virtuální minikameru, skrze níž se podíváme na zbytek scény. Pixely této scény vyrenderované v nízkém rozlišení sečteme.

Tento postup je aproximací integrace světla přicházejícího do daného bodu přes celou hemisféru.

Bohužel realizace této idey by byla velmi pomalá.

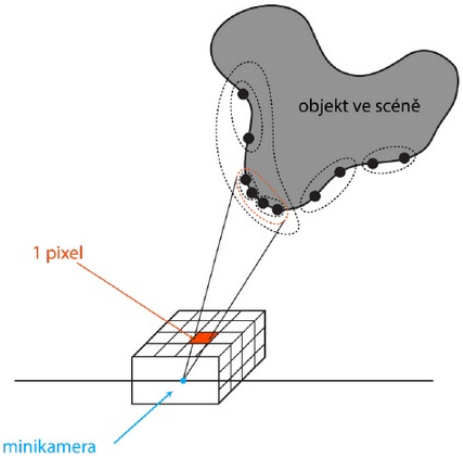


Obrázek 7: *Ukázka principu virtuální kamery*

2.2 Zrychlení algoritmu

Zrychlení bude spočívat v tom, že budeme celou scénu reprezentovat jenom body - tzv. mračny bodů (žádné polygony, jenom body). Při renderování scény pak tyto body budeme promítat na naši minikameru, která je reprezentována jakousi rastrovou polokrabicí.

Vzhledem k tomu, že víme, jak jsou jednotlivé body daleko, využijeme z-buffer k rozlišení viditelnosti.



Obrázek 8

V nejjednodušší verzi tohoto algoritmu bychom všechny body scény promítali jeden po druhém, což je zásadní nevýhodou. Proto než začneme scénu renderovat, vybudujeme nad body reprezentující scénu hierarchii slučující body do uzlů. Potom při zobrazování nějakého objektu ve scéně budeme procházet hierarchii uzlů, dokud nezjistíme, že vůči tomu bodu, kde počítáme přímé osvětlení, je velikost hierarchického uzlu rovna velikosti pixelu na polokrabici. Nemá totiž smysl, aby když se do jednoho pixelu promítá 50 bodů, abychom každý jeden z nich promítali. Místo toho promítneme nějaký výše postavený hierarchický uzel obsahující tyto body a jehož barva je průměrem barev jednotlivých bodů.

2.3 Použití ve filmu

Tento algoritmus byl použit při renderování mnoha celovečeních filmů, například Piráti z Karibiku 2 a 3, Spiderman 3, Harry Potter 5 a 6, Ironman 1 a 2, Quantum of Solace, 2012 a mnoha dalších.