SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 5

GLOBALNO DIFUZNO OSVJETLJENJE

Robert Sajko

Zagreb, lipanj 2010.

Sadržaj

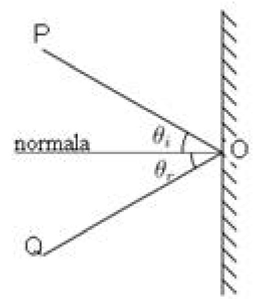
1. Uvod

Interakcijom svjetlosti s okolinom nastaju različite optičke pojave. Ljudsko osjetilo vida osobito je osjetljivo čak i na vrlo suptilne svjetlosne efekte, te uz pomoć tih detaljnih informacija mozak slaže sliku obrisa okolnog prostora. Iz tog razloga, da bi računalno generirane slike bile uvjerljive, od iznimne su važnosti točni proračuni osvjetljenja. No, takvi su proračuni veoma zahtjevni, pa su sve do nedavno bili rezervirani isključivo za skupe radne stanice kojima su se koristili filmski studiji pri izradi specijalnih efekata i računalno animiranih sekvenci. U području interaktivne računalne grafike, izračun osvjetljenja se redovito vrlo grubo aproksimirao, što je bilo potrebno da se postigne izvršavanje u stvarnom vremenu. Takvim razvojem modela osvjetljenja, došlo je do njihove profilacije. S jedne strane, razvijali su se modeli i tehnike globalnog osvjetljenja, koji su zasnovani na fizikalno točnim simulacijama, te modeli i tehnike lokalnog osvjetljenja, koji točno simuliraju samo uski dio fizikalnih pojava vezanih uz svjetlost, dok ostatak aproksimiraju proizvoljnim, empirijski utvrđenim, jednostavnim funkcijama, ili čak konstantama. S vremenom, najpopularniji od tih modela lokalnog osvjetljenja (Blinn-Phong model) postao je općeprihvaćen kao standardni model osvjetljenja u interaktivnoj računalnoj grafici. U velikoj količini literature, kada se govori o osvjetljenju u kontekstu računalne grafike, pretpostavlja se upravo taj model, te se objašnjava kako se njime modeliraju fizikalne pojave. Međutim, u ovom radu, krenut ćemo od samog početka. Prvo ćemo točno definirati koja su to svojstva i fizikalne pojave vezane uz svjetlost, a zatim izvesti lokalni model osvjetljenja, te prikazati tehnike evaluacije takvih modela. Proširit ćemo definiciju lokalnog modela osvjetljenja, i izvesti potpunu jednadžbu iscrtavanja. No, umjesto direktnog i potpunog rješenja ove jednadžbe, pokušat ćemo pronaći parcijalna rješenja, koja aproksimiraju samo poneke globalne učinke, poput ambijentalnog zaklanjanja ili difuzne interrefleksije s jednim odbijanjem. Glavno pitanje jest, koliko toga možemo aproksimirati na današnjim računalima a da dobijemo maksimalnu kvalitetu prikaza u stvarnom vremenu? Predložit ćemo nekoliko kompromisa, i iz njih izvesti nekoliko tehnika, te odrediti prednosti i nedostatke svake od njih. Na kraju, najpogodniji teorijski model će biti implementiran.

2. Fizikalni modeli svjetla

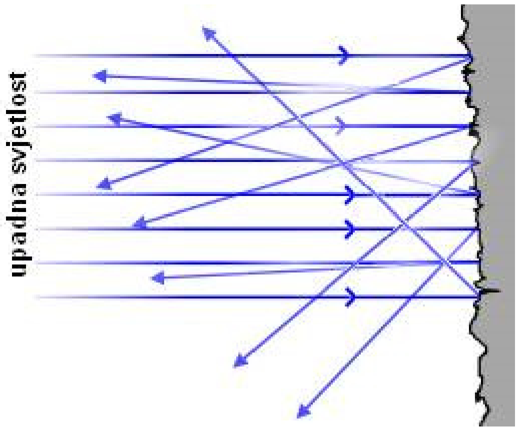
Da bismo mogli formalno opisati propagaciju svjetlosti kroz prostor, i time izgraditi teorijski model osvjetljenja pogodan za računalnu simulaciju, potrebno je prvo sagledati i točno definirati pojam svjetlosti, način propagacije svjetlosti, te sve pojave i učinke koji se pritom javljaju. Postoji više interpretacija, razvijanih tokom stoljeća. Najjednostavnije shvaćanje svjetlosti jest u okviru geometrijske optike. Svjetlost se predstavlja skupom svjetlosnih zraka – diskretnih, apstraktnih objekata bez mase i dimenzija. S druge strane, u okviru fizikalne optike, svjetlost je shvaćena kao elektromagnetsko zračenje, te se sukladno tome, širenje svjetlosti shvaća kao širenje vala. Svjetlosne 'zrake' su tada ništa drugo doli aproksimacija valnih fronta svjetlosti. Treće shvaćanje svjetlosti dolazi iz kvantne fizike, a ono kaže da je svjetlost skup elementarnih čestica, 'paketića' energije, zvanih fotoni. Trenutno najpotpuniji model svjetlosti jest kombinacija valne i čestične interpretacije prirode svjetlosti. Neovisno o modelu interpretacije, svjetlost pokazuje određena svojstva i učinke koji se javljaju prilikom njena gibanja. Najosnovnija svojstva su refleksija i refrakcija.

Refleksija se definira kao promjena smjera širenja svjetlosne zrake (tj. vala), na granici dvaju sredstava. Ovo je najčestaliji primjer optičkih efekata – svjetlost putuje zrakom, doseže površinu nekog objekta, te se odbija, mijenjajući smjer svog širenja. Matematički, ta se promjena smjera opisuje pomoću dviju veličina – upadnog kuta, i kuta refleksije. Ti se kutovi definiraju s obzirom na normalu površine na koju svjetlost upada. Zakon refleksije jednostavno kaže: upadni kut jednak je kutu refleksije (slika 1).



Slika 1. Zrcalna refleksija.

Međutim, valja imati na umu da većina realnih tijela imaju površine koje nisu savršeno glatke. To znači da će normale na površinu biti različite za pojedine upadne zrake svjetlosti, a to nadalje znači i da će kutovi refleksija biti različiti za pojedine zrake, makar sve imaju iste upadne kutove s obzirom na ravninu površine tijela. Drugim riječima, na grubim površinama će reflektirane zrake biti raspršene u različitim smjerovima (slika 2). Iz tog razloga, možemo reći da postoje dvije različite vrste refleksija: zrcalna refleksija, i difuzna refleksija. Kod idealno zrcalne refleksije, savršeno glatko tijelo reflektira sve zrake u istom smjeru, dok kod idealno difuzne refleksije, savršeno difuzno tijelo reflektira zrake svjetlosti jednoliko u svim smjerovima, čineći da se svjetlost širi iznad tijela u obliku polukugle. Primjer gotovo idealne zrcalne refleksije su ogledala, dok bi primjer difuzne refleksije bilo neobrađeno drvo. Dakako, svi realni predmeti pokazuju obje komponente refleksije, ali u različitim omjerima.



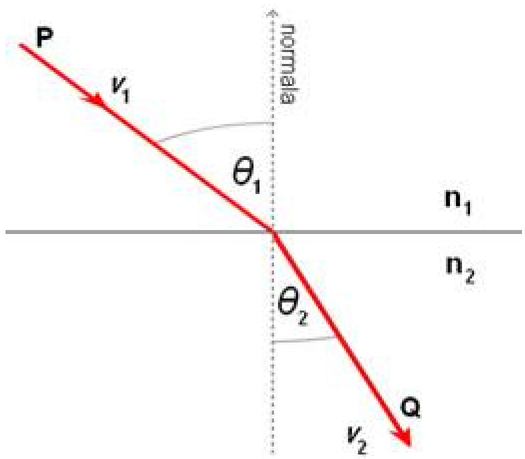
Slika 2. Difuzna refleksija.

Vezano uz refleksiju, valja spomenuti još i učinak pretapanja boja. Naime, ukoliko se dva predmeta nalaze blizu jedan drugoga, očito je da se može dogoditi da zrake svjetlosti reflektirane od jednog predmeta upadnu na drugi. U slučaju zrcalne refleksije, cjelokupna slika predmeta će biti reflektirana prema drugom predmetu (primjerice, ako postavimo dva ogledala jedno pored drugog, vidjet ćemo u njima sliku u slici u slici, u beskonačnost). No, u slučaju difuzne refleksije, efekt je mnogo suptilniji. Budući da nema jasne, usmjerene slike, nego samo širok prostor razasutih zraka, ti će predmeti utjecati jedan na drugoga samo blagom promjenom percipirane boje. Primjerice, ako pored izvora bijele svjetlosti stavimo bijeli papir, te mu približimo jarko crveni predmet (poput crvenog flomastera), papir će na jednom dijelu poprimiti blago crvenkastu boju. Ovaj se efekt, koji nastaje kao direktan rezultat difuzne interrefleksije, naziva pretapanje boja.

Sljedeće osnovno svojstvo koje pokazuje svjetlost jest refrakcija. To je pojava koja se javlja kad svjetlost prelazi iz jednog propagacijskog sredstva u drugo, i time mijenja brzinu. Naime, za svako sredstvo se može definirati tzv. indeks loma, koji govori koliko će dano sredstvo usporiti gibanje svjetlosti. Dakle, indeks loma se definira na sljedeći način:

$$n = \frac{c}{v}$$

Zakon refrakcije kaže da će se zbog promjene brzine gibanja svjetlosti, promijeniti i njen smjer širenja. Ukoliko definiramo upadni kut i kut refleksije s obzirom na normalu granice između dvaju sredstava, onda će vrijediti da se sinus upadnog kuta prema sinusu kuta refrakcije odnosi kao upadna brzina svjetlosti prema izlaznoj brzini. Ukoliko upadnu i izlaznu brzinu izrazimo pomoću indeksa loma, zakon refrakcije možemo formulirati ovako:



Slika 3. Refrakcija svjetlosti.

$$\frac{sin\theta_1}{sin\theta_2} = \frac{v_1}{v_2} = \frac{n_2}{n_1}$$

Međutim, treba imati na umu da u većini realnih situacija, upadna zraka svjetlosti neće biti isključivo reflektirana, niti isključivo refraktirana, već će se dogoditi obje pojave. Totalna refleksija i totalna refrakcija su moguće samo u nekim posebnim slučajevima, kad su ispunjeni određeni uvjeti. Ti su uvjeti za totalnu refleksiju sljedeći: upadni kut svjetlosti mora biti veći od kritičnog kuta, te indeks loma prvog sredstva mora biti veći od indeksa loma drugog sredstva. Pritom je kritični kut dan sljedećim izrazom:

$$\theta_c = \arcsin\left(\frac{n_2}{n_1}\right)$$

gdje je n_1 indeks loma prvog sredstva, a n_2 indeks loma drugog sredstva. Za totalnu refrakciju postoji samo jedan uvjet, a taj je da jedno od sredstava propagacije svjetlosti između kojih svjetlost prolazi ima negativni indeks loma. U prirodni ne postoje tvari s takvim svojstvom, no, laboratorijski je moguće proizvesti određene materijale koji imaju negativni indeks loma.

Zanimljiv optički fenomen direktno vezan uz refrakciju (i refleksiju) jest kaustika. Kaustika je ovojnica zraka svjetlosti refraktiranih, ili reflektiranih, od zaobljene površine. Također, pod kaustikom se može podrazumijevati i projekcija te ovojnice na neku drugu površinu. Dobar primjer kaustike može proizvesti prazna staklena čaša na praznom stolu, na koju upada sunčeva svjetlost. Radi opetovanih refrakcija i refleksija zraka svjetlosti uvjetovanih strukturom čaše, izlazne zrake svjetlosti mogu na stolu projicirati neobičan oblik. Takvi se oblici općenito nazivaju kaustika (slika 4).

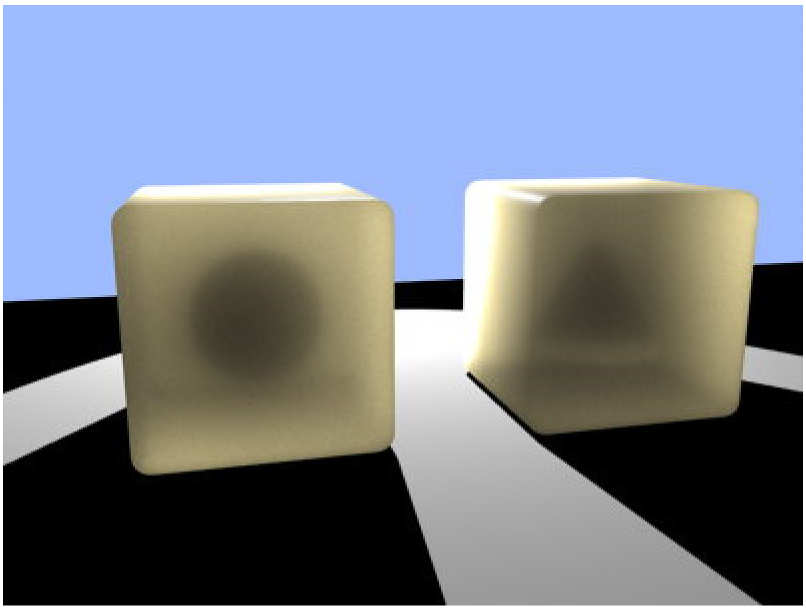
Slika 4. Primjer kaustike.

Osim refleksije i refrakcije, dvaju najosnovnijih pojava koje dobro opisuju i geometrijska i fizikalna optika, svjetlost pokazuje i svojstva difrakcije i interferencije, koja se ne mogu objasniti geometrijskom optikom. Naime, difrakcija je pojava savijanja vala zbog neke prepreke. Premda je difrakcija uvijek prisutna, njeni su učinci primjetljivi tek ako je red veličine dimenzija prepreke jednak redu veličine valne duljine vala. Najpoznatiji primjer učinaka difrakcije bi bila stražnja strana CD-a, koja promatrana ispod izvora bijele svjetlosti stvara efekt duginih boja. Značajna primjena difrakcije su i hologrami, koji se primjerice koriste za zaštitu e-indeksa. Učinci difrakcije su uzrokovani interferencijom valova svjetlosti. Interferencija je superpozicija (zbrajanje) dvaju valova koji su međusobno koherentni (imaju istu frekvenciju). Ukoliko dva vala iste valne duljine, odaslana iz istog izvora, poradi neke prepreke budu savinuti (difrakcija), te se tako desi da propagiraju kroz isti prostor, s istim amplitudama i frekvencijama (jer imaju i iste valne duljine), ali međusobno pomaknuti u fazi, bit će ispunjeni uvjeti za interferenciju, pa će se interferencija i dogoditi. Očito, ovisno o pomaku u fazi, na nekim mjestima će i prvi i drugi val oba imati pozitivnu ili negativnu amplitudu, dok će na drugim mjestima jedan imati pozitivnu, a drugi negativnu amplitudu. To znači da će na nekim mjestima rezultirajući val imati povećanu amplitudu, a na nekim mjestima umanjenu amplitudu. Na primjeru CD-a, upadni valovi svjetlosti dopiru do premaza na stražnjoj strani CD-a, zbog čije mikroskopske strukture bivaju savinuti, dakle dolazi do difrakcije, što rezultira interferencijom. Upravo te valove koji nastaju kao rezultat interferencije percipiramo kao dugine boje. Dakako, defrakcijom i interferencijom se mogu objasniti i zakoni refleksije i refrakcije, koji vrijede ako je prepreka mnogo većih dimenzija od valne duljine vala.

Još jedna pojava vezana uz propagaciju svjetlosti, a koja se može objasniti isključivo valnom prirodom svjetlosti, jest polarizacija. Naime, općenito govoreći, valovi se mogu podijeliti na progresivne (putujuće) i stacionarne, a među progresivnima razlikujemo transverzalne i longitudinalne. Svjetlost jest progresivni, transverzalni val, što znači da se njegovo električno i magnetsko polje mijenjaju periodički u smjerovima okomitim na smjer gibanja vala. Kod nepolarizirane svjetlosti, vektori električnog i magnetskog polja zauzimaju s jednakom vjerojatnošću bilo koji smjer okomit na vektor širenja vala. To znači da se prirodna, nepolarizirana svjetlost sastoji od valova koji titraju u svim ravninama. S druge strane, polarizirana svjetlost je takva čiji valovi titraju u samo jednoj ravnini. Do polarizacije svjetlosti dolazi refleksijama i refrakcijama. Primjerice, odbljesak Sunca na površini mora jest polarizirana svjetlost. Iz tog razloga, kvalitetnije sunčane naočale su zapravo polarizacijski filtri, čime se ublažuju neugodni odbljesci. Inače, golim ljudskim okom je gotovo nemoguće razlikovati polariziranu od nepolarizirane svjetlosti – potrebne su posebne vježbe da bi osoba naučila zapažati veoma suptilni efekt pri promatranju polarizirane svjetlosti, nazvan Haidingerova četka.

Potrebno je spomenuti da u većini situacija, samo će dio upadne svjetlosti biti reflektiran i refraktiran, dok će ostatak biti apsorbiran. Objašnjenje ove pojave daje čestična teorija svjetlosti – fotoni prilikom sudaranja s atomima mogu predati svoju energiju elektronu i time nestati, ili samo promijeniti smjer i brzinu. Naime, možemo zamisliti da se elektroni nalaze raspoređeni oko jezgre atoma ne na proizvoljnim udaljenostima, već u diskretnim razinama. Da bi elektron prešao iz niže u višu razinu, potrebno je da foton koji se sudari s njim ima točno odgovarajuću količinu energije, koja odgovara razlici više i niže energetske razine elektrona. Prijelazom elektrona iz niže u višu energetsku razinu, povećava se ukupna unutarnja energija tijela, odnosno, povećava se njegova toplina. U obrnutom slučaju, ukoliko elektron prelazi iz više u nižu razinu, višak energije se oslobađa u obliku fotona, čija je energija točno jednaka razlici energija više i niže razine. U ovim činjenicama leži razlog zašto različiti predmeti imaju različitu boju. Naime, bijela svjetlost sadrži sve vidljive valne duljine, što znači fotone svih frekvencija. Budući da je frekvencija fotona proporcionalna njegovoj energiji, a atom može apsorbirati foton samo točno odgovarajuće energije (što ovisi o vrsti atoma), očito je da će samo dio fotona iz bijele svijetlosti biti apsorbiran, a ostatak će biti odbijen, dakle, promijenit će im se smjer i brzina. Koje valne duljine svjetlosti će biti apsorbirane, a koje reflektirane i refraktirane, ovisi o vrsti materijala, odnosno, o vrsti atoma od kojih je materijal sačinjen.

Sada možemo objasniti i pojavu ispodpovršinskog raspršivanja svjetlosti. Naime, kako fotoni upadaju na tijelo, neki bivaju apsorbirani, dok se drugi odbijaju od atoma. Moguće je da neki od tih fotona nastave putovati unutar samog tijela, dakle ispod njegove površine. Drugim riječima, svjetlost kroz neka tijela može barem djelomično prolaziti, te se takva tijela nazivaju (djelomično) prozirna. U stvarnosti, sva su tijela prozirna bar u nekoj malenoj mjeri. Očitovanje djelomične prozirnosti jest upravo u tome što se dio upadnih fotona raspršuje ispod površine tijela, te nakon opetovanih sudara napušta tijelo. Ova je pojava sveprisutna, iako je u većini slučajeva slabo zamjetna. Ispodpovršinsko raspršivanje se najčešće asocira s organskim materijalima, poput kože ili voska, ali i tvarima poput gume, mramora i slično. Na slici 5 možemo vidjeti dobar primjer homogenog raspršivanja svjetlosti kroz volumene dviju kocaka, unutar kojih naziremo obrise sadržanih neprozirnih tijela (u jednoj kugla, u drugoj stožac). Te unutarnje obrise raspoznajemo upravo radi efekta ispodpovršinskog raspršivanja svjetlosti.



Slika 5. Primjer ispodpovršinskog raspršivanja.

3. Modeli direktnog osvjetljenja

U prethodnom poglavlju predstavljeni su različiti fizikalni modeli svjetla. Za potrebe računalne grafike (posebice u stvarnom vremenu), pojave koje je moguće prikazati valno-čestičnim modelom svjetlosti su zanemarive, budući da se događaju u rijetkim slučajevima, a znatno pridonose kompleksnosti problema. Iz tog razloga, u pravilu se pretpostavlja geometrijski model svjetla kao idealni, fotorealistični model. No, i takav ograničeni model potrebno je aproksimirati i pretočiti u oblik pogodan za računalnu implementaciju. Tradicionalno, u interaktivnoj računalnoj grafici promatra se samo direktni doprinos izvora svjetla na danu površinu. Drugim riječima, efekti poput difuzne interrefleksije ili ispodpovršinskog raspršivanja se zanemaruju, budući da se time problem izračuna osvjetljenja daleko pojednostavljuje (i u smislu jednostavnosti implementacije, i, još važnije, u smislu računalne zahtjevnosti). Dakle, potrebno je formulirati model koji opisuje osvijetljenost neke površine, u ovisnosti o intenzitetu i kutu upadne i reflektirane zrake svjetlosti. U općem slučaju, definira se funkcija koja opisuje reflektivnost površine. U literaturi, takva se funkcija naziva *bidirekcionalna funkcija distribucije reflektivnosti* (eng. *bidirectional reflectance distribution function, BRDF*). Formalno, opći oblik BRDF funkcije izgleda ovako:

$$f_r(\omega_i, \omega_o) = \frac{\mathrm{dL_r}(\omega_o)}{\mathrm{dE_i}(\omega_i)} = \frac{\mathrm{dL_r}(\omega_o)}{\mathrm{L_i}(\omega_i)\cos\theta_i\,\mathrm{d}\omega_i} \quad (1)$$

Pritom je ω_i vektor upadne zrake svjetlosti, a ω_o vektor reflektirane zrake svjetlosti, dok je θ_i kut između vektora upadne zrake svjetlosti i normale na površinu u točki upada. Dakako, svi su spomenuti vektori jedinični. Iz tog razloga, mogu se parametrizirati dvjema sfernim koordinatama, kutom azimuta φ i kutom elevacije θ, dok je radijus po definiciji jednak jedan (iz ovoga slijedi da je konačna funkcija zapravo četverodimenzionalna). Nadalje, L_r je intenzitet reflektiranog svjetla (zračenje po steradijanu po jedinici površine, eng. *radiance*), dok je E_i upadno zračenje po steradijanu (eng. *irradiance*). Dakle, BRDF funkcija za dani par upadnog i reflektiranog vektora definira odnos intenziteta reflektiranog svjetla i upadnog zračenja po steradijanu.

Na temelju općeg oblika BRDF funkcije, razvijeno je nekoliko različitih, konkretnih modela osvjetljenja koji se koriste u računalnoj grafici. Takvi modeli se dijele u dvije glavne skupine: *analitički modeli* (temelje se na aproksimacijama općeg oblika BRDF-a, pretpostavljajući određene parametre ili ograničenja, ili na heurističkim i numeričkim aproksimacijama), te *empirijski modeli* (zasnivaju se na mjerenjima reflektivnosti stvarnih materijala u laboratorijskim uvjetima).

Najjednostavniji analitički BRDF model može se izvesti pretpostavljajući direkcionalni izvor svjetlosti, te idealno difuznu površinu koja je ravna ploha. Kod direkcionalnog svjetla, upadne zrake svjetlosti su paralelne, te u ovom slučaju, sve upadaju pod istim kutem na površinu (normala na površinu je u svakoj točki upada jednaka). Također, budući da je površina savršeno difuzna, svaka upadna zraka se reflektira jednoliko u svim smjerovima, u hemisferi nad površinom. Uz navedene pretpostavke, BRDF funkcija se reducira na konstantnu vrijednost. Naime, kako je rečeno, vrijednost BRDF funkcije je omjer intenziteta reflektiranog svjetla i upadnog zračenja po steradijanu. Budući da se upadne zrake svjetlosti reflektiraju jednoliko po cijeloj hemisferi, bez obzira na kut upada ili poziciju promatrača, to znači da je traženi omjer jednak za bilo koji par upadnog i reflektiranog vektora zrake svjetlosti. Dakako, iz iskustva već znamo da intenzitet svjetlosti (zračenje po steradijanu po jedinici površine) opada s povećanjem upadnog kuta, a to je upravo zato što se jednak iznos zračenja rasprostire po većoj površini, pa intenzitet svjetlosti opada. Bez gubitka općenitosti, možemo uzeti da je navedeni omjer točno jednak jedan, pa dobivamo sljedeće:

$$\frac{\mathrm{dL_r}}{\mathrm{L_i}\cos\theta\,\mathrm{d}\omega_\mathrm{i}} = 1$$

$$\int \mathrm{dL_r} = \int \mathrm{L_i}\cos\theta\,\mathrm{d}\omega_\mathrm{i}$$

S obzirom da L_i, L_r i θ_i više nisu funkcije upadnog ili reflektiranog vektora, ili normale na površinu, već konstante, možemo pisati:

$$\mathrm{L_r} = \mathrm{L_i}\cos\theta \int \mathrm{d}\omega_\mathrm{i}$$

Budući da je vektor ω_i parametriziran sfernim koordinatama (θ, φ) uz konstantan radijus, gornji integral zapravo opisuje površinu jedinične hemisfere, za koju možemo uzeti da je jednaka nekoj konstanti C, iz čega slijedi da je intenzitet reflektiranog svjetla proporcionalan intenzitetu upadnog svjetla pomnoženom s kosinusom upadnog kuta:

$$L_r \propto L_i \cos\theta \quad (2)$$

Valja primijetiti da smo ovim putem zapravo izveli Lambertov kosinusni zakon, koji predstavlja model difuzne refleksije tradicionalno korišten u računalnoj grafici, a čiji je standardni oblik:

$$I_d = k_d \,(L \cdot N)\, i_d \quad (3)$$

gdje je I_d intenzitet difuznog osvjetljenja, i_d intenzitet difuzne komponente upadnog svjetla, L vektor upadne zrake svjetla, N vektor normale na površinu u točki upada, a k_d vrijednost iz intervala [0, 1] koja definira stupanj difuzne reflektivnosti površine (gdje $k_d = 1$ predstavlja idealno difuzni materijal).

Međutim, jasno je da je dobiveni model tek jedan specijalni slučaj, koji vrijedi samo ako su zadovoljene početne (vrlo ograničavajuće!) pretpostavke. Primjerice, za mnoge stvarne materijale *ne vrijedi* pretpostavka da je omjer intenziteta reflektiranog svjetla i upadnog zračenja po steradijanu jednak za svaki par upadnog i reflektiranog vektora zrake svjetlosti. Štoviše, kod idealnih ravnih zrcala, taj omjer je jednak nuli za sve takve parove vektora, osim za slučaj kada je upadni kut točno jednak kutu refleksije, kada je spomenuti omjer jednak jedan (gdje je upadni kut definiran kao kut između upadnog vektora i normale na površinu u točki upada, a kut refleksije kao kut između reflektiranog vektora i normale). Drugim riječima, idealna zrcala reflektiraju upadnu svjetlost u samo jednom, točno određenom smjeru, što rezultira neizobličenom, simetričnom slikom. Prema tome, BRDF funkcija se za idealna ravna zrcala može zapisati u sljedećem obliku:

$$f_r(\omega_i, \omega_o) = \begin{cases} 1, & \theta_i = \theta_r \\ 0, & \theta_i \neq \theta_r \end{cases}$$

Odnosno,

$$L_r(\omega_o) = \begin{cases} \iint L_i(\omega_i) \cos\theta_i \, d\omega_i \,, & \theta_i = \theta_r \\ 0, & \theta_i \neq \theta_r \end{cases}$$

S druge strane, brojni materijali pokazuju nesavršenu zrcalnu reflektivnost, što znači da BRDF funkcija nije binarna, već može poprimiti bilo koji realni broj u intervalu [0, 1]. Kod nesavršeno zrcalnih objekata proizvoljnog oblika, dolazi do pojave *zrcalnog odsjaja* (eng. *specular highlight*), budući da se upadna svjetlost reflektira u uskom stošcu oko idealno reflektirane zrake. Dakle, svo upadno zračenje će biti reflektirano na uskom dijelu površine, što znači da će to područje imati znatno veći intenzitet osvjetljenja, a otuda i naziv pojave. Širina zrcalnog odsjaja ovisi, dakako, o karakteristikama specifičnog materijala. Formalno, opisani se problem može zapisati na sljedeći način:

$$f_r(\omega_i, \omega_o) = f_s(\delta)$$

$$\mathrm{L_r}(\omega_\mathrm{o}) = \int \mathrm{L_i}(\omega_\mathrm{i}) \cos\theta_\mathrm{i} \, f_s(\delta) \, \mathrm{d}\omega_\mathrm{i}$$

gdje je δ kut između vektora ω_o i idealno reflektirane zrake (valja imati na umu da za zrcala koja nisu ravna, smjer idealno reflektirane zrake općenito ovisi o položaju promatrača), a $f_s(\delta)$ funkcija koja definira širinu zrcalnog odsjaja (dakle, opada s porastom δ).

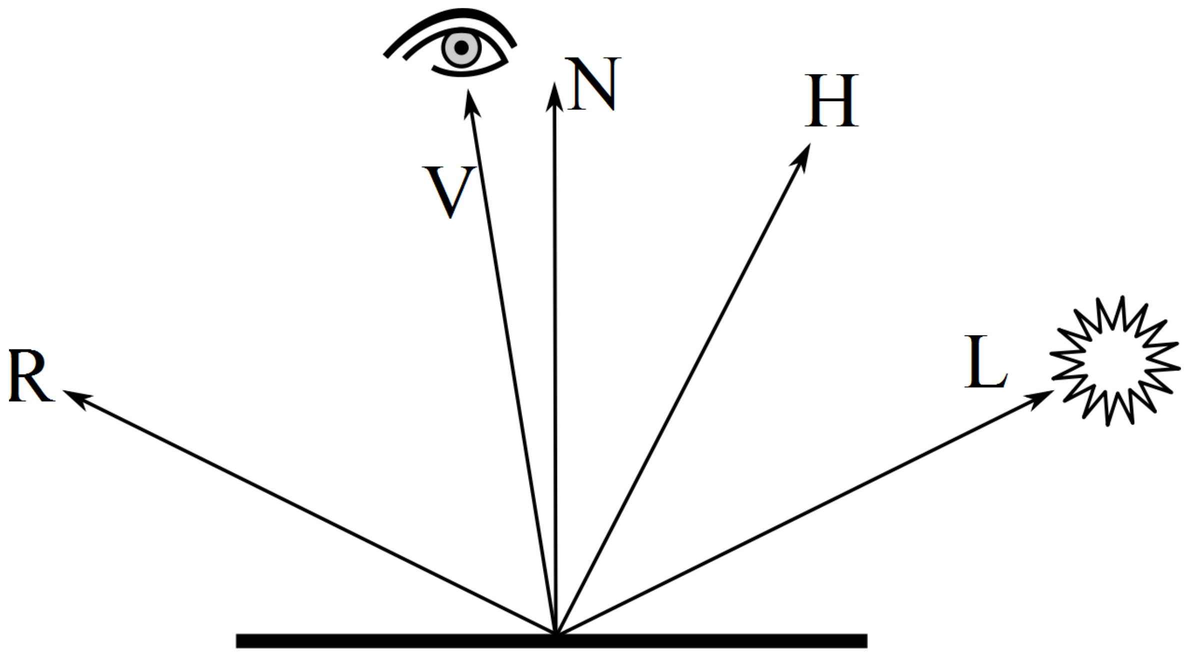
Analitičko rješenje gornjeg izraza bi zahtijevalo izračunavanje integrala po hemisferi za svaki slikovni element, pri svakom osvježavanju slike, što nije primjenjivo za potrebe interaktivne računalne grafike. Iz tog razloga, svi postojeći modeli zrcalnog odsjaja se baziraju na određenim heuristikama i opažanjima. Primjerice, vjerojatno najpoznatiji je Phongov model, koji se bazira na opažanju da funkcija intenziteta odsjaja odgovara eksponencijalnoj:

$$I_s = k_s(R \cdot V)^\alpha i_s \quad (4)$$

gdje je I_s intenzitet zrcalnog odsjaja, i_s intenzitet zrcalne komponente upadnog svjetla, R vektor idealno reflektirane zrake svjetla, V vektor promatrača, α faktor sjajnosti (određuje širinu i intenzitet zrcalnog odsjaja), a k_s vrijednost iz intervala [0, 1] koja definira stupanj zrcalne reflektivnosti površine (gdje $k_s = 1$ predstavlja idealno zrcalni materijal).

U praktičnim primjenama, pokazalo se da Phongov model pruža prihvatljivu aproksimaciju zrcalnog odsjaja, uz nisku računalnu složenost. No, češće se koristi varijanta nazvana Blinn-Phong model, koja u formuli (4) zamjenjuje izraz $R \cdot V$ izrazom $N \cdot H$, gdje je N vektor normale na površinu, a H je tzv. *poluvektor*:

$$I_s = k_s(N \cdot H)^\alpha i_s \quad (5)$$

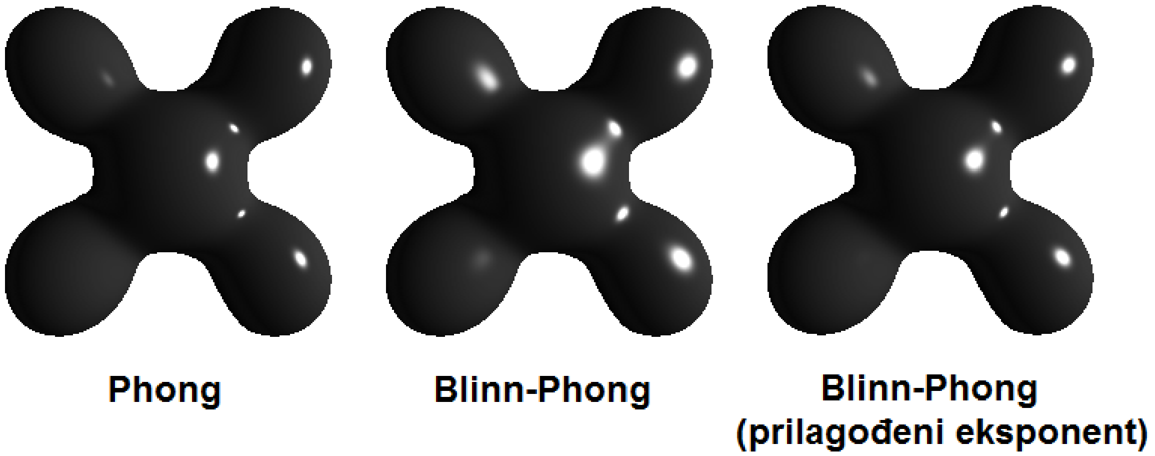


Slika 6. Vektori korišteni pri izračunu osvjetljenja Phong i Blinn-Phong modelom.

Poluvektor H se definira na sljedeći način:

$$H = \frac{L + V}{|L + V|} \quad (6)$$

Dakle, kao što je ilustrirano na slici 6, poluvektor H se nalazi "na pola puta" između vektora promatrača V, i vektora zrake svjetlosti L, pa otuda i naziv. Rezultati koji se dobiju Blinn-Phong modelom su vrlo slični rezultatima običnog Phong modela, iako se mogu primijetiti razlike u obliku zrcalnog odsjaja. Također, bitno je uočiti da skalarni umnožak $R \cdot V$ odgovara kosinusu kuta koji je dvostruko veći od kuta čijem kosinusu odgovara skalarni umnožak $N \cdot H$, iz čega možemo zaključiti da bismo za dobivanje približno jednakih rezultata trebali prilagoditi eksponent α. Na slici 7 mogu se vidjeti zrcalne komponente osvjetljenja dobivene Blinn-Phong i Phong modelima. Prva slika slijeva prikazuje objekt osvijetljen Phong modelom, dok slika u sredini prikazuje isti objekt osvijetljen Blinn-Phong modelom uz identične parametre. Jasno je primjetljiv pojačan intenzitet i drugačiji oblik zrcalnog odsjaja. Uz prilagodbu eksponenta α, moguće je ostvariti odsjaj vrlo sličnog ili gotovo jednakog intenziteta, kako pokazuje zadnja slika slijeva - no, i dalje se mogu primijetiti suptilne razlike u obliku odsjaja.



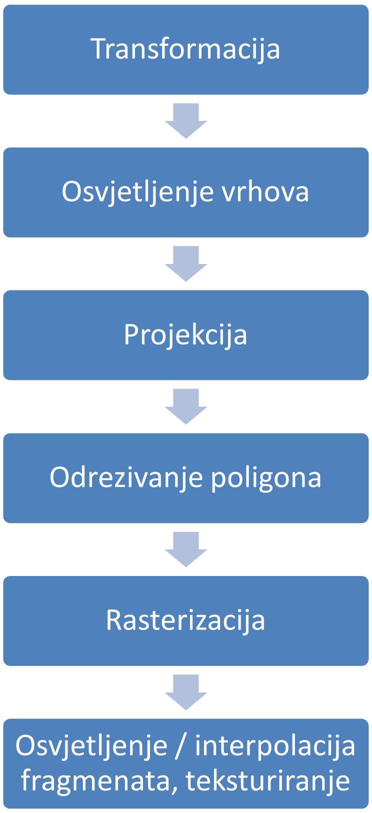
Slika 7. Usporedba zrcalne komponente osvjetljenja dobivene različitim modelima.

Zanimljivo je istaknuti kako Blinn-Phong model, premda je nastao kao aproksimacija Phong modela, zapravo daje rezultate koji su bliži empirijski utvrđenim BRDF modelima, za brojne vrste stvarnih materijala. Ova kombinacija dobrih svojstava (rezultati vjerni stvarnima, niska računalna zahtjevnost) rezultirala je iznimnom popularnošću Blinn-Phong modela, te su rani grafički akceleratori, koji nisu pružali mogućnost programirljivih cjevovoda, sklopovski implementirali upravo Blinn-Phong model. Dakako, moderne programirljive grafičke kartice omogućuju implementaciju proizvoljnog modela osvjetljenja. Za specifične primjene, razvijeni su različiti, računalno zahtjevniji modeli koji pružaju rezultate vjernije stvarnosti, primjerice, u slučaju iscrtavanja ljudskih lica (kože), ili raznih drugih specifičnih materijala.

4. Tehnike evaluacije modela osvjetljenja

Neovisno o odabranom modelu osvjetljenja, potrebno je osmisliti i implementirati odgovarajući postupak evaluacije traženog modela. Danas su u široj primjeni dva glavna pristupa: **unaprijedno iscrtavanje** (eng. *forward rendering*), te **iscrtavanje s odgodom** (eng. *deferred rendering*).

Unaprijedno iscrtavanje je starija metoda, koja je tradicionalno bila izvedena sklopovljem na grafičkim karticama. Osnovni tok operacija pri iscrtavanju neke scene ovim postupkom dan je sljedećom slikom (scena je, dakako, definirana kao skup poligona):



Slika 8. Postupak unaprijednog iscrtavanja.

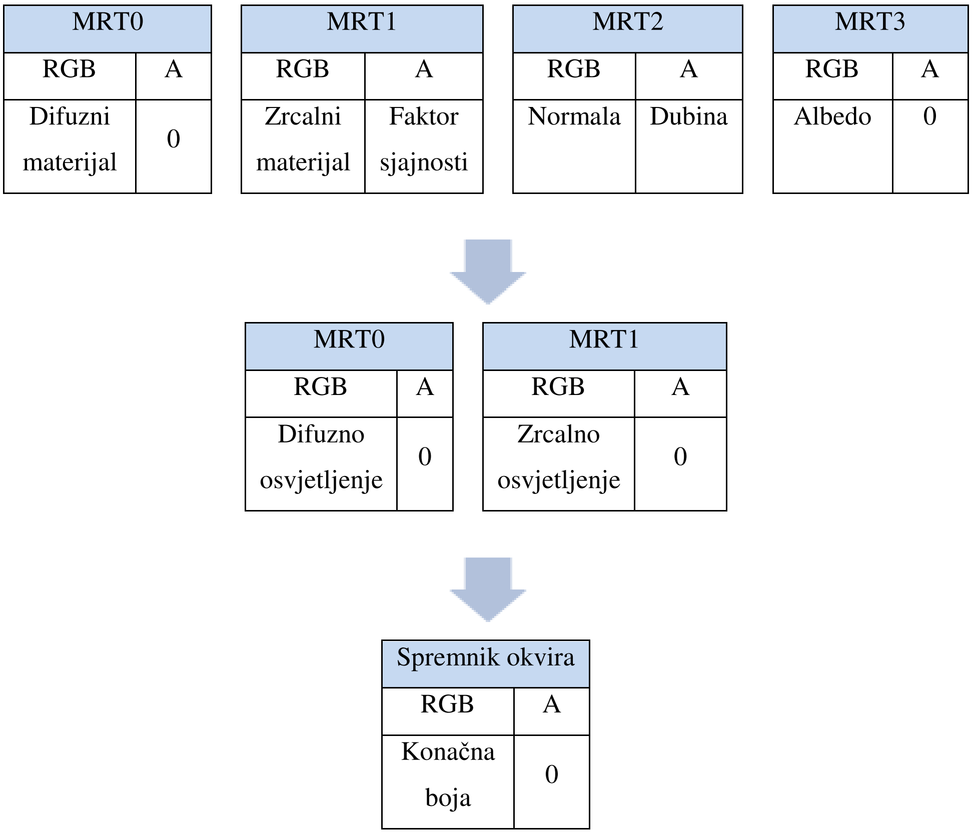
U prvoj fazi, nad ulaznim podacima (vrhovima poligona) obavlja se transformacija, jednostavnim matričnim množenjem. Možemo razlikovati dvije glavne matrice: matricu modela (eng. *world matrix*, odnosno, *model matrix*), te matricu pogleda (eng. *view matrix*). Matrica modela omogućuje transformaciju vrhova objekta iz svog lokalnog koordinatnog sustava (prostor modela), u globalni koordinatni sustav scene (prostor scene, tj. "svijeta"). Matrica pogleda služi za transformaciju iz prostora scene u prostor kamere, dakle omogućava koncept virtualne kamere koju je moguće slobodno pozicionirati i orijentirati u prostoru scene. Vezano uz koncept kamere, bitna je još i projekcijska matrica. Naime, za prikaz 3D objekata na 2D zaslonu, potrebno je izvršiti odgovarajuću projekciju vrhova poligona. Vrsta i parametri projekcije su prirodno enkapsulirani unutar koncepta kamere, te ti podaci tvore projekcijsku matricu. Ova cjelokupna faza se naziva geometrijska faza, budući da se direktno vrši obrada ulazne geometrije. Tradicionalno, u ovoj se fazi izračunava osvjetljenje za pojedini vrh, upotrebom Blinn-Phong modela osvjetljenja, i time dobiva boja vrha kao četvero-komponentni vektor (crveni, zeleni, plavi i alfa kanal). Pojedinačne komponente tog vektora su širine 8 bita, što je dostatno za pohranu boja za prikaz na računalnim zaslonima. Dakako, ukoliko se definira vlastiti program za sjenčanje vrhova (eng. *vertex shader*), moguće je obraditi vrhove na proizvoljan način.

U sljedećem koraku, vrši se odrezivanje poligona koji nisu vidljivi (eng. *clipping*). Naime, nakon projekcije, vrhovi poligona leže na 2D ravnini, koja je po definiciji beskonačna. No, rasterizirana slika koju želimo dobiti ima konačne dimenzije, što znači da je vrhove koji leže izvan tih granica potrebno odbaciti, i time odrezati poligone ili dijelove poligona koji nisu vidljivi.

Nakon odrezivanja poligona, moguće je provesti postupak rasterizacije, odnosno generiranja skupa diskretnih slikovnih elemenata, koje ćemo nazvati fragmenti. Za svaki pojedini fragment, potrebno je odrediti konačnu boju. Jednostavan pristup je interpolacija boje izračunate po vrhu, što je funkcionalnost koju pruža grafičko sklopovlje. No, uz upotrebu vlastitih programa za sjenčanje fragmenata (eng. *fragment shader*) moguće je računati osvjetljenje zasebno za svaki fragment, korištenjem proizvoljnog modela osvjetljenja. Konačni rezultat za dani fragment se zapisuje u spremnik okvira (eng. *framebuffer*), uz mogućnost miješanja sa prethodno zapisanom vrijednosti (eng. *blending*). Alfa-miješanjem (miješanje boja modulirano vrijednostima alfa kanala) može se postići efekt prozirnosti ili poluprozirnosti, no moguće je koristiti miješanje i za druge efekte.

Najvažnija činjenica za istaknuti jest, da kod unaprijednog iscrtavanja, cjelokupni opisani postupak se izvršava za svaki vrh ulazne geometrije zasebno, od prvog koraka do zadnjeg. Budući da se vrhovi obrađuju potpuno neovisno jedan o drugome, ovaj je postupak moguće trivijalno paralelizirati. Moderne grafičke kartice upravo imaju masivno paralelnu arhitekturu, te se sastoje od više stotina procesorskih jezgri. Međutim, ovo također znači da prilikom obrade nekog vrha ili fragmenta, nije moguć pristup podacima nekog drugog vrha ili fragmenta.

Upravo ovaj problem rješava postupak iscrtavanja s odgodom. Naime, umjesto direktnog generiranja konačne slike iz ulazne geometrije, kod iscrtavanja s odgodom na temelju ulazne geometrije generira se samo skup nekih podataka (u obliku 2D slike), a koji su potrebni za računanje osvjetljenja. Dakle, svi prethodno navedeni koraci iscrtavanja se izvode nad ulaznom geometrijom, uz korištenje odgovarajućih programa za sjenčanje vrhova i fragmenata, i dobije se "slika" scene koja nije konačna, osvijetljena slika, već samo skup određenih podataka o sceni. Zatim, iscrtava se nova geometrija - pravokutnik koji prekriva cijeli zaslon - a boja pojedinih slikovnih elemenata tog pravokutnika određuje se računanjem osvjetljenja početne scene, na temelju prethodno dobivenih podataka. Dakle, za iscrtavanje s odgodom, potrebna su barem dva prolaza. Prvi prolaz služi isključivo za ekstrakciju potrebnih podataka iz ulazne geometrije scene, dok se samo osvjetljenje računa dodatnim prolazom - otuda i naziv postupka. Podaci generirani prvim prolazom se pohranjuju u teksture, koje se nazivaju G-spremnici (eng. *G-buffer*, skraćeno od *geometry buffer*), a njihova priroda ovisi o željenom modelu osvjetljenja. Primjerice, za često korišteni Blinn-Phong model, potrebni su barem difuzni i zrcalni materijal, parametar sjajnosti, te normale (pogledati poglavlje 3). Nadalje, većina stvarnih aplikacija koristi teksturirane modele, što znači da je potreban dodatan spremnik za pohranu neosvijetljene boje modela učitane iz teksture, a koja se naziva albedo. Navedene spremnike je moguće odjednom generirati u jednom prolazu, korištenjem tehnike višeodredišnog iscrtavanja (eng. *multiple render targets, MRT*). Sve dosad rečeno može se pregledno prikazati sljedećom slikom:



Slika 9. Primjer konfiguracije spremnika i prolaza pri iscrtavanju s odgodom.

Drugi i treći prolaz je moguće sažeti u jedan prolaz, pogotovo za jednostavnije aplikacije, no ukoliko je potrebno implementirati neke dodatne algoritme opisane u kasnijim poglavljima, pogodnije je razložiti postupak na više prolaza.

Obje opisane tehnike iscrtavanja imaju svoje prednosti i mane. Unaprijedno iscrtavanje je moguće obaviti u jednom prolazu, bez korištenja dodatnih spremnika, no glavni nedostatak je nemogućnost dijeljenja podataka o vrhovima odnosno fragmentima, što može biti preduvjet za implementaciju određenih naprednih algoritama. Iscrtavanje s odgodom rješava ovaj problem, ali uz cijenu dodatnih prolaza i utroška memorije. Međutim, možda i najznačajnija prednost iscrtavanja s odgodom se očituje kod korištenja većeg broja svjetala na sceni. Naime, kod tradicionalnog, unaprijednog iscrtavanja, cjelokupnu (vidljivu) geometriju scene je potrebno ponovno obraditi za svako svjetlo - od transformacije vrhova, do rasterizacije i određivanja boje fragmenata. S druge strane, kod iscrtavanja s odgodom, geometrija scene se obrađuje samo jedanput - jednom kad su G-spremnici izgrađeni, moguće ih je upotrijebiti proizvoljan broj puta, kroz proizvoljan broj prolaza. Dakle, svako dodatno svjetlo je zapravo jedan dodatan prolaz nad G-spremnicima. Budući da se osvjetljenje računa u prostoru slike, geometrijska kompleksnost više nema utjecaja na postupak osvjetljavanja, te glavni utjecaj na performanse ima rezolucija iscrtavanja. Rezultati svih prolaza osvjetljenja se akumuliraju u spremniku okvira, korištenjem aditivnog miješanja (boji fragmenta već upisanog u spremnik jednostavno se zbroji boja novo-izračunatog fragmenta). Nadalje, moguće su još poneke optimizacije kod iscrtavanja s odgodom. Naime, većina svjetala koja se javljaju u različitim scenama su lokalna, tj. svjetla ograničena u prostoru. Primjeri takvih svjetala su točkasta svjetla (eng. *point light*), te reflektorska svjetla (eng. *spot light*). Točkasti izvor svjetlosti isijava svjetlost iz neke točke prostora, jednoliko u svim smjerovima, i to na način da intenzitet osvjetljenja opada s udaljenosti od izvorišne točke; stoga, volumen kojeg obasjava točkasto svjetlo možemo zamisliti kao kuglu. Reflektorski izvor svjetlosti je sličan točkastom, no s tom razlikom da se svjetlost isijava samo u određenim smjerovima, i to tako da je opisan volumen stošca, čiji je vrh, naravno, u točki izvora svjetla. Bitno za uočiti jest činjenica da lokalna svjetla gotovo nikada ne osvjetljavaju cjelokupnu vidljivu scenu na zaslonu. Čak štoviše, u većini slučajeva, svako pojedino svjetlo na sceni će nakon iscrtavanja zauzimati možda 10% do 20% površine zaslona, odnosno slikovnih elemenata. Iz ovog opažanja slijedi da kod iscrtavanja s odgodom, nije čak niti potrebno za svako svjetlo provesti prolaz nad cjelokupnim G-spremnicima, nego samo nad onim dijelom nad kojim dano svjetlo ima utjecaja. Određivanje regije utjecaja danog svjetla moguće je obaviti korištenjem spremnika šablone (eng. *stencil buffer*), što je zapravo bitovna maska. Naime, budući da je volumen kojeg opisuje neko lokalno svjetlo unaprijed poznat, te odgovara nekom primitivnom geometrijskom tijelu, moguće je to geometrijsko tijelo iscrtati u spremnik šablone i tako dobiti bitovnu masku, gdje su svi elementi jednaki nuli osim onih koji odgovaraju fragmentima na koje dano svjetlo ima utjecaja. Zatim, dovoljno je jednostavno primijeniti tu bitovnu masku pri računanju prolaza osvjetljenja - grafičko sklopovlje će obraditi samo one fragmente koji su označeni maskom. Premda generiranje spremnika šablone traje određeno vrijeme, to vrijeme je gotovo zanemarivo (budući da je riječ o iscrtavanju jednostavnog geometrijskog tijela, bez teksturiranja, osvjetljavanja ili bilo kakvih drugih efekata - i to u spremnik širine jednog bita). S druge strane, primjenom šablone moguće je u većini slučajeva preskočiti relativno skupo izračunavanje osvjetljenja za 80-90% fragmenata, što je izvanredna ušteda. Za primjer, uzmimo iscrtavanje pri rezoluciji 1920x1080, što znači da je potrebno obraditi otprilike dva milijuna slikovnih elemenata. Bez upotrebe spremnika šablone, za svako pojedino svjetlo bilo bi potrebno obraditi svih dva milijuna fragmenata, dok je u protivnom za svako svjetlo u prosjeku potrebno obraditi samo dvjesto tisuća fragmenata. Uz ovu optimizaciju, iscrtavanje s odgodom omogućava upotrebu gotovo neograničenog broja lokalnih svjetala. Božićno drvce okićeno mnoštvom šarenih žaruljica, ili gradska ulica noću, osvijetljena nizom uličnih lampi i farova automobila u prolazu, samo su primjeri scena vrlo bogatih dinamičkim, lokalnim izvorima svjetla, koje bi bilo gotovo nemoguće izvesti unaprijednim iscrtavanjem. Dakako, navedena razmatranja vrijede samo za lokalna svjetla - globalni, direkcionalni izvori svjetla (kao što je Sunce), ionako obuhvaćaju cijelu scenu, pa optimizacija spremnikom šablone nije moguća - uostalom, za većinu scena, dovoljan je samo jedan direkcionalni izvor svjetla, i to najčešće Sunce.

Sljedeća zanimljiva činjenica koju valja razmotriti jest, da kod unaprijednog iscrtavanja, nije unaprijed poznato da li će pojedini vrh rezultirati fragmentom koji će biti prekriven nekim drugim fragmentom ili ne, što znači da će vrlo vjerojatno doći do precrtavanja (eng. *overdraw*). Da bi se osiguralo pravilno iscrtavanje, potrebno je ili iscrtavati objekte pravilnim redoslijedom (od najdaljih, do najbližih), ili koristiti spremnik dubine (eng. *depth buffer*, odnosno *z-buffer*), što je vrlo često korišteno rješenje. (Kao mala digresija, kod iscrtavanja poluprozirnih objekata, spremnik dubine ne pomaže, te je ipak potrebno iscrtavati objekte od najdaljeg do najbližeg, radi pravilnog miješanja boja.) Budući da se kod unaprijednog iscrtavanja osvjetljenje odmah računa u prvom prolazu, jasno je da će neki od tih (relativno skupih) izračuna osvjetljenja biti potraćeni, budući da će dobiveni fragmenti možda biti precrtani fragmentima manje dubine. S druge strane, kod iscrtavanja s odgodom, do precrtavanja može doći jedino u prvom prolazu, kad se generiraju G-spremnici, no u toj fazi se ionako ne vrše nikakvi izračuni, već samo izvlače potrebni podaci. To znači da možemo garantirati da osvjetljenje nikada nećemo računati za fragment koji će biti odbačen, što, ovisno o sceni, može biti ili marginalna ili značajna ušteda, ali ušteda u svakom slučaju. Nažalost, to također znači da više nije moguće pravilno iscrtati poluprozirne objekte - budući da pri računanju osvjetljenja imamo informaciju samo o jednom, najbližem fragmentu, nije moguće izvesti miješanje s dubljim fragmentima (koji su bili odbačeni u prvom prolazu). Jedna mogućnost za rješavanje ovog problema jest korištenje dodatnih spremnika za pohranu dodatnih dubina fragmenata (u literaturi se ova tehnika naziva guljenje dubine, eng. *depth peeling*), no jasno je da time znatno povećavamo i vrijeme iscrtavanja i utrošak memorije. Druga mogućnost jest jednostavno korištenje unaprijednog iscrtavanja za ovaj poseban slučaj poluprozirnih objekata.

Dodatno pitanje koje je potrebno posebno razmotriti, jest pitanje izglađivanja nazubljenih rubova poligona (eng. *anti-aliasing*). Naime, postojeće grafičko sklopovlje posjeduje ugrađene mehanizme za izglađivanje koji su osmišljeni za upotrebu s jednoprolaznim, unaprijednim iscrtavanjem, te nisu direktno upotrebljivi pri iscrtavanju s odgodom. Naime, navedeni mehanizmi se baziraju na ideji višestrukog uzorkovanja spremnika dubine, primjerice dvostrukog ili četverostrukog. Prilikom obrade vrhova i generiranja fragmenata, ukoliko se uzorci spremnika dubine razlikuju za dobiveni fragment (dakle, promatrani fragment se nalazi na granici poligona), odredi se boja za svaki od uzoraka i zatim obavi jednostavno miješanje, čime se rubovi poligona izglade. Međutim, problem s ovim postupkom jest taj, što kod iscrtavanja s odgodom prilikom iscrtavanja geometrije zapravo ne dobivamo konačnu boju fragmenata, već samo određene podatke za kasniju obradu. Prema tome, miješanje i "izglađivanje" tih podataka će rezultirati pogrešnim izračunom osvjetljenja (izračunati osvjetljenje nad izglađenim materijalima nije isto kao i izgladiti već osvijetljene fragmente). Ovaj ozbiljan problem je srećom riješen na modernim grafičkim karticama koje podržavaju DirectX 10 tehnologiju, koja omogućava višestruko uzorkovanje proizvoljnih spremnika i tekstura, te direktan pristup tim poduzorcima unutar programa za sjenčanje fragmenata. Na taj način, moguće je višestruko uzorkovati G-spremnike, ali odgoditi miješanje poduzoraka do završetka prolaza za osvjetljenje. Pritom program za sjenčanje fragmenata koji računa osvjetljenje mora ručno učitati poduzorke G-spremnika za dani fragment, izračunati osvjetljenje za svaki poduzorak zasebno, te kombinirati dobivene rezultate u konačnu boju. Dakako, ovaj postupak nije potrebno provoditi za baš svaki fragment, već samo za one koji se nalaze na rubovima poligona. Detekciju rubnih fragmenata je također potrebno ručno napraviti, unutar programa za sjenčanje. Primjerice, dobar način kako to izvesti jest korištenjem spremnika šablone; u pretprolazu, posebnim programom za sjenčanje fragmenata određuju se rubni fragmenti u G-spremnicima, usporedbom dubina poduzoraka. Ukoliko su zadovoljeni odgovarajući uvjeti, fragment se proglašava rubnim, te se u spremnik šablone zapisuje vrijednost jedan, a u suprotnom, zapisuje se vrijednost nula. Prema tome, izračun doprinosa pojedinog svjetla se razdvaja na dva pod-prolaza. Prvi pod-prolaz izračunava osvjetljenje koristeći samo jedan poduzorak G-spremnika (budući da su svi isti), a drugi pod-prolaz izračunava osvjetljenje onoliko puta koliko postoji poduzoraka G-spremnika (budući da se neki ili svi razlikuju), te kombinira te rezultate u konačnu boju. Uz upotrebu spremnika šablone, možemo osigurati da će grafičko sklopovlje izvesti prvi pod-prolaz samo nad ne-rubnim fragmentima, a drugi, puno skuplji pod-prolaz samo nad rubnim fragmentima, za koje je doista i potrebno izglađivanje. Dakle, premda je izglađivanje nazubljenih rubova poligona višestrukim uzorkovanjem moguće i kod iscrtavanja s odgodom, potrebna je relativno moderna grafička kartica (DirectX 10 ili novija), a čak i tada, potrebno je mnogo dodatnog posla, što nije slučaj kod unaprijednog iscrtavanja.

U konačnici, najbolji rezultati se postižu kombiniranjem obje tehnike, na način da se iscrtavanje s odgodom koristi za neprozirne objekte, a unaprijedno za (polu)prozirne. Također, i direkcionalna svjetla je moguće riješiti unaprijednim iscrtavanjem, budući da bi njihovo rješavanje iscrtavanjem s odgodom samo značilo trošak dodatnog prolaza, a bez mogućnosti optimizacije kao kod lokalnih svjetala. Prednosti i nedostaci pojedinih tehnika sažeti su tablicom 1. Analizom kompleksnosti, možemo zaključiti da su oba algoritma linearne složenosti - svako dodatno svjetlo zahtjeva jednak broj dodatnih koraka. Međutim, ovo je zavaravajući rezultat, budući da ova dva postupka zapravo ne djeluju nad istim skupom podataka, te se ne mogu na tako jednostavan način direktno usporediti. Unaprijedno iscrtavanje za svako svjetlo obrađuje svaki vrh, te svaki generirani fragment. Iscrtavanje s odgodom vrhove obrađuje samo jedanput, a zatim za svako svjetlo obrađuje minimalni podskup fragmenata. Ukoliko s L označimo ukupan broj svjetala, s V ukupan broj vrhova, s F ukupan broj fragmenata, a s F' minimalni skup fragmenata (u prosjeku 10% ukupnog broja), možemo kompleksnost formulirati izrazima navedenima u tablici 1:

Tablica 1. Usporedba unaprijednog iscrtavanja, i iscrtavanja s odgodom.

	Unaprijedno iscrtavanje	**Iscrtavanje s odgodom**
Kompleksnost	$L \cdot (V+F)$	$V+L \cdot F'$
Minimalan broj prolaza	1	2
Minimalan dodatan utrošak memorije (prosjek)	0 MB	~100 MB
Maksimalan broj simultano aktivnih svjetala	8	Neograničeno
Stopa precrtavanja	Niska do srednja	Nema precrtavanja
Poluprozirni objekti	Da	Ne
Izglađivanje rubova poligona	Da	Da (DX10 ili noviji)

5. Preslikavanje i prikaz osvjetljenja

U prethodnim poglavljima, promatrani su modeli lokalnog osvjetljenja, te postupci izračunavanja tih modela. Međutim, ono što nije bilo posebno promotreno jest problem ograničene preciznosti grafičkog sklopovlja i računalnih zaslona. Naime, u stvarnom svijetu, omjer najvećeg i najmanjeg intenziteta osvjetljenja koje se javlja u svakodnevnom životu je vrlo visok, i iznosi otprilike $10^{12} : 1$. Ovaj omjer se naziva dinamički raspon, te u kontekstu računalne grafike, definira potrebnu preciznost za točan prikaz svih razina osvjetljenja. Za dani primjer, ukupno bi bilo potrebno barem $\lfloor \log_2(10^{12}) \rfloor + 1 = 40$ bitova za očuvanje dinamičkog raspona. Međutim, postojeći moderni računalni zasloni su u prosjeku sposobni za prikaz dinamičkog raspona od svega $10^2 : 1$, za čiji prikaz je dovoljno $\lfloor \log_2(10^2) \rfloor + 1 = 7$ bitova. Budući da 8-bitni cijeli brojevi pružaju dostatnu preciznost za ovaj raspon, a operacije nad njima su vrlo brze, upravo je takav tip podataka definiran kao standardni format za zapis boje slikovnih elemenata (dakle, 8 bitova po R, G, B, A kanalima, ukupno 32 bita po slikovnom elementu). Rane grafičke kartice su podržavale pohranu i operacije nad fragmentima isključivo u navedenom formatu, što je dostatno za zapis slike prikazane na zaslonu. Međutim, jasno je da pri računanju osvjetljenja u ograničenoj preciznosti dolazi do značajnog gubitka informacija - odličan primjer scene kod koje se javlja ovaj problem jest zatvoreni, tamni prostor u kojeg upada jarka, vanjska svjetlost kroz neki otvor, tako da je dinamički raspon scene vrlo visok. Dakako, čak i kada bismo računali osvjetljenje u dovoljnoj preciznosti za očuvanje cjelokupnog dinamičkog raspona, te rezultate ne bismo mogli direktno prikazati na zaslonu. Područja koja bi bila presvijetla bi bila odrezana na maksimalnu vrijednost, a ostatak zaokružen na najbližu od 256 diskretnih razina osvjetljenja po kanalu, gubeći pritom mnoge detalje. Upravo ovaj problem rješava postupak tonskog preslikavanja. No, prije objašnjenja tog postupka, bilo bi korisno upoznati se s određenim osnovnim svojstvima ljudskog vida. Naime, niti ljudsko oko nije sposobno jednoliko zapažati detalje u cjelokupnom dinamičkom rasponu kakav se javlja u stvarnom svijetu. Štoviše, sposobnost opažanja ljudskog oka je ograničena na uzak raspon od otprilike $10^3 : 1$. Ipak, čovjek je sposoban opažati detalje i u relativno mračnom, i u jarko osvijetljenom prostoru - ali ne istovremeno u oba. Ključ ove sposobnosti leži u mogućnosti adaptacije oka, tako da opažajni raspon oka pokriva onaj dio dinamičkog raspona osvjetljenja, koji ima najveći utjecaj na scenu, odnosno, u kojem se nalazi najviše detalja. Dakle, ljudsko oko se zapravo prilagođava srednjoj vrijednosti intenziteta upadne svjetlosti. Ovaj princip se koristi i u fotografiji, gdje se definira pojam ekspozicije, kao ukupne dozvoljene količine svjetlosti koja upada na fotografski medij. Idealno, cilj je da ekspozicija fotografije odgovara srednjoj vrijednosti intenziteta osvjetljenja fotografirane scene. U protivnom, moguća su dva pogrešna slučaja. U prvom slučaju, ekspozicija je previsoka, slika je presvijetla, te su u najsvjetlijim područjima detalji izgubljeni (potpuno bijela područja); u drugom slučaju, ekspozicija je preniska, slika je pretamna, te su u najtamnijim područjima detalji izgubljeni (potpuno crna područja). Ove pojave se najlakše uočavaju naglim prijelazom iz vrlo svijetlog u vrlo tamni prostor (ili obrnuto) - budući da adaptacija ljudskog oka traje određeno vrijeme, slika koju ćemo isprva vidjeti će biti preniske ili previsoke ekspozicije.

Upravo opisani princip se koristi u postupku tonskog preslikavanja. Naime, osvjetljenje izračunato u visokoj preciznosti, potrebno je preslikati u nizak dinamički raspon kakav se može prikazati na zaslonu. Dakle, problem je ekvivalentan adaptaciji ljudskog oka, stoga možemo koristiti isto rješenje - u prvom koraku, izračunat ćemo srednji intenzitet osvjetljenja scene, i na temelju toga potrebnu ekspoziciju. U drugom koraku, primijenit ćemo određeni ne-linearni operator nad fragmentima visokog raspona, te dobiti fragmente niskog raspona, takve da tvore sliku tražene ekspozicije. Konačni rezultat jest očuvanje detalja koji bi inače bili izgubljeni, da su svi izračuni osvjetljenja bili izvršeni u niskoj preciznosti. Primjer scene za koju iscrtavanje HDR osvjetljenjem (eng. *high dynamic range*) znatno poboljšava kvalitetu u odnosu na iscrtavanje LDR osvjetljenjem (eng. *low dynamic range*) jest dan slikom 10.

Slika 10. Usporedba LDR (lijevo) i HDR (desno) načina iscrtavanja.

Premda je algoritam tonskog preslikavanja konceptualno vrlo jednostavan, pravilna implementacija nije tako jednostavna za izvesti. Dakako, prvi korak je odabir odgovarajućeg formata za pohranu fragmenata. Moderne grafičke kartice nude 16- i 32-bitnu preciznost po kanalu boje, dakle ukupno 64 ili 128 bita po fragmentu, uz korištenje standardizirane aritmetike pomičnog zareza. Veća preciznost povlači lošije performanse i dvostruko veću potrošnju memorije, ali niža preciznost može uzrokovati pojavu artefakata. Jednom kad je odabran zadovoljavajući format, svi G-spremnici moraju biti kreirani istim formatom, osim konačnog spremnika okvira, koji mora imati 8-bitnu preciznost po kanalu boje (radi prikaza na zaslonu). Zatim, potrebno je definirati dodatan prolaz koji će se izvoditi nakon što je generirana konačna osvijetljena slika (dakle, nakon svih prolaza osvjetljenja) - budući da će ta slika biti visokog dinamičkog raspona (HDR), potrebno je implementirati program za sjenčanje fragmenata koji će izvršiti tonsko preslikavanje, i generirati ekvivalentnu sliku niskog dinamičkog raspona (LDR). Kako je rečeno, prvi korak ovog postupka se sastoji od izračuna srednje vrijednosti intenziteta osvjetljenja, za što možemo koristiti sljedeću formulu:

$$I_{sr} = \exp\left(\frac{1}{N}\sum_{x,y} \log\left(\delta + I(x,y)\right)\right) \quad (1)$$

gdje je I_{sr} srednji intenzitet osvjetljenja, N ukupan broj fragmenata, δ neka malena vrijednost, a $I(x, y)$ boja fragmenta na poziciji (x, y). Broj δ je potreban za slučaj potpuno crnih fragmenata, odnosno $I(x, y) = 0$. Međutim, postavlja se pitanje - kako evaluirati gornju formulu nad spremnikom u obliku teksture u memoriji grafičke kartice? Srećom, postoji jednostavan trik kojeg možemo koristiti. Naime, dovoljno je predprolazom izračunati logaritam svakog fragmenta HDR slike i te vrijednosti zapisati u novu teksturu, što je trivijalno izvedivo vrlo jednostavnim programom za sjenčanje fragmenata. Zatim, upotrebom ugrađene funkcionalnosti grafičke kartice, potrebno je generirati kompletan mip lanac dobivene logaritamske teksture, za što je dovoljan tek jedan funkcijski poziv. Naime, mip preslikavanjem se početna tekstura sažimlje u teksturu dvostruko manjih dimenzija, na način da je svaki fragment umanjene teksture dobiven kao srednja vrijednost četiriju odgovarajućih susjednih fragmenata početne teksture. Ovaj se postupak može ponavljati sve dok se ne dobije tekstura dimenzija 1x1, čime se dobije niz uzastopno umanjenih tekstura koji se naziva mip lanac. Dakle, u konačnoj teksturi mip lanca, zapravo je sadržana srednja vrijednost ukupne početne slike. Prema tome, generiranjem mip lanca logaritamske teksture dobivamo u završnoj teksturi konačnu vrijednost izraza:

$$\frac{1}{N}\sum_{x,y}\log\left(\delta + I(x,y)\right) \quad (2)$$

Za izračun konačne vrijednosti I_{sr}, te provedbu samog tonskog preslikavanja originalne HDR slike scene, potreban je još jedan prolaz, sa zasebnim programom za sjenčanje fragmenata. U ovom prolazu, prvi korak je dakako učitavanje vrijednosti završne teksture prethodno generiranog mip lanca, te izračun srednje vrijednosti intenziteta osvjetljenja, jednostavnim antilogaritmiranjem. Nakon toga, potrebno je izvršiti linearno skaliranje HDR fragmenta na temelju zadane, tražene ekspozicije, pomoću izraza:

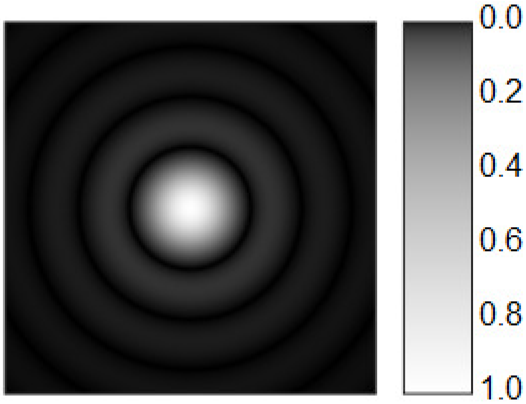
$$I_s(x,y) = \frac{\alpha}{I_{sr}} I(x,y) \quad (3)$$

gdje je α tražena ekspozicija, a $I_s(x, y)$ skalirana boja fragmenta (x, y). Valja uočiti da smo ovime linearno preslikali boju fragmenta u interval centriran oko tražene ekspozicije, no taj interval nije nužno [0, 1], kakav mora biti za prikaz na zaslonu, već općenito [0, ∞). Prema tome, potrebno je još odabrati odgovarajući nelinearni operator koji će izvršiti preslikavanje [0, ∞) → [0, 1]. Najjednostavniji takav operator koji se često koristi dan je sljedećim izrazom:

$$I(x,y) = \frac{I_s(x,y)}{1 + I_s(x,y)} \quad (4)$$

Sada je još jedino preostalo pitanje kako odabrati odgovarajuću ekspoziciju. Jasno, moguće je jednostavno uzeti da ekspozicija odgovara izračunatoj srednjoj vrijednosti intenziteta osvjetljenja scene, što je zapravo idealni slučaj u fotografiji. No, moguće je i simulirati adaptaciju ljudskog oka na nagle promjene u osvijetljenosti, na način da promjena ekspozicije kasni za promjenom srednjeg intenziteta osvjetljenja, što se može postići postepenom promjenom ekspozicije kroz nekoliko slikovnih okvira (eng. *frame*).

Dodatan efekt koji se javlja u stvarnim lećama, a kojeg je jednostavno implementirati uz HDR osvjetljenje, jest pojava prelijevanja svjetla. Naime, čak i kod idealnih leća, javlja se artefakt nazvan Airyev uzorak, koji nastaje zbog difrakcije svjetla koje upada na okruglu leću, a rezultira pojavom središnjeg svijetlog kruga i alternirajućih tamnih i svijetlih pojaseva, oko jarko osvijetljenih objekata. Primjer ovog artefakta dan je slikom 11:



Slika 11. Airyev uzorak.

Međutim, realne leće nikada nisu savršeno fokusirane, tako da je i ovaj artefakt zamućen, te prijelazi između pojedinih svijetlih pojaseva nisu primjetni, kako prikazuje sljedeća, stvarna fotografija (slika 12):

Slika 12. Primjer prelijevanja svjetla.

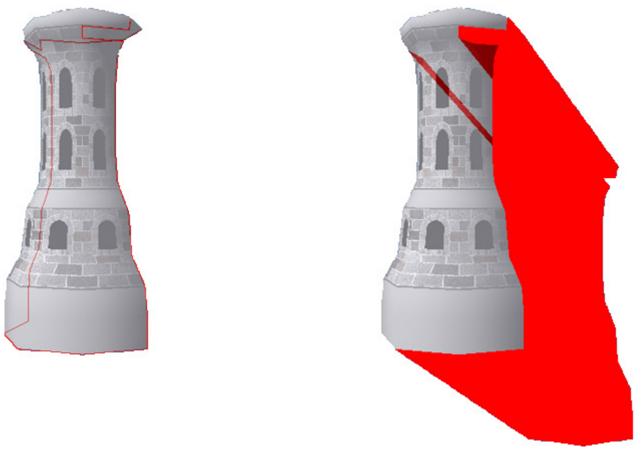
Iz gornjih opažanja, proizlazi ideja praktične implementacije pojave prelijevanja svjetla. Dakle, budući da već računamo osvjetljenje fragmenata u visokom dinamičkom rasponu, moguće je dodati još jedan prolaz kojim ćemo odbaciti tamne fragmente, a zadržati samo vrlo svijetle; drugim riječima, primijenit ćemo visoko-propusni filtar nad HDR slikom. Rezultirajuću sliku je zatim potrebno nekoliko puta uzastopno zamutiti, primjenom Gaussovog zamućivanja. Konačni rezultat je aproksimacija zamućenog Airyevog uzorka kakav se javlja u stvarnim lećama kod jarko osvijetljenih objekata. Na kraju, dobivenu zamućenu sliku je potrebno jednostavno zbrojiti s LDR slikom (dobivenom tonskim preslikavanjem), i rezultat je slika s efektom prelijevanja svjetla. Za uštedu vremena obrade i memorije, dovoljno je gornje korake izvršiti nad HDR slikom smanjenih dimenzija (dvostrukim ili trostrukim mip preslikavanjem), budući da ionako vršimo zamućivanje.

Na kraju, potrebno je još razmotriti problem izglađivanja nazubljenih rubova poligona. Naime, HDR osvjetljenje je moguće implementirati i s unaprijednim iscrtavanjem, i s iscrtavanjem s odgodom. Međutim, čak i kod unaprijednog iscrtavanja, nije moguće koristiti ugrađene, sklopovske mehanizme izglađivanja višestrukim uzorkovanjem, budući da su opet potrebna dva prolaza za računanje konačnih, osvijetljenih fragmenata. Dakle, u prvom prolazu bila bi obrađena geometrija scene i izračunati HDR fragmenti, a u drugom prolazu izvedeno tonsko preslikavanje (što je obrada slike), i tako dobiveni LDR fragmenti. Budući da se sklopovsko izglađivanje automatski provodi u prolazu u kojem se obrađuje geometrija, dobili bismo "izglađene" HDR fragmente. Problem je u tome što nad tim fragmentima moramo provesti tonsko preslikavanje, što *nije* linearna operacija - dakle, rezultati neće biti isti kao kada bismo izglađivanje proveli *nakon* tonskog mapiranja. Drugim riječima, dobili bismo pogrešne rezultate, što znači da opet moramo koristiti mogućnost ručnog pristupa pojedinim višestrukim uzorcima koju nudi DirectX 10 (vidjeti poglavlje 4).

6. Određivanje zaklonjenosti direktnog svjetla

Pitanje koje je dosad ostalo neodgovoreno, a koje ćemo razmotriti u ovom poglavlju, jest pitanje sjena. Naime, iz iskustva je već poznato da osvijetljeni objekti zaklanjaju određeni dio prostora, tako da taj prostor leži u sjeni. Međutim, opisanim postupcima osvjetljenja, računali bismo osvjetljenje i u prostoru koji bi trebao ležati u sjeni. Potreban je, dakle, poseban postupak kojim ćemo odrediti zaklonjenost. U tu svrhu, postoje dva glavna algoritma, od kojih svaki ima određene prednosti i nedostatke, a to su algoritam volumena sjene (eng. *shadow volume*), te algoritam preslikavanja sjena (eng. *shadow mapping*).

Algoritam volumena sjene se zasniva na ideji da se zaklonjeni prostor također može geometrijski opisati, kao i pravi objekti na sceni. Prema tome, scena se dijeli na dva dijela - početna geometrija scene, te geometrija sjena. Tu dodatnu geometriju, koju možemo nazvati volumen sjene (otuda i naziv algoritma) je potrebno generirati za svaki osvijetljeni objekt, za svako svjetlo zasebno. Postupak generiranja volumena sjene se sastoji od dva koraka: pronalaženje siluete objekta, te ekstruzije siluete u volumen. Silueta se definira kao skup bridova koji povezuju prednje i stražnje poligone objekta. Prednji poligoni su oni koji su orijentirani prema izvoru svjetla, a stražnji oni koji su orijentirani u suprotnom smjeru. Jednostavan test za određivanje koji poligoni su prednji, a koji stražnji, jest izračun skalarnog umnoška vektora normale poligona, te vektora smjera svjetla. Ukoliko je rezultat pozitivan broj, poligon je prednji, inače je stražnji. Jednom kad je određena silueta objekta, potrebno ju je proširiti u volumen, na način da se svakom vrhu siluete pridoda i bridom poveže novi vrh, pomaknut u smjeru suprotnom smjeru svjetla za dani vrh, i to na neku (beskonačno) veliku udaljenost. Opisani postupak predočava slika 13.



Slika 13. Prikaz siluete objekta, te proširenja siluete u volumen.

Konačno, sam postupak iscrtavanja scene sastoji se od sljedećih koraka:

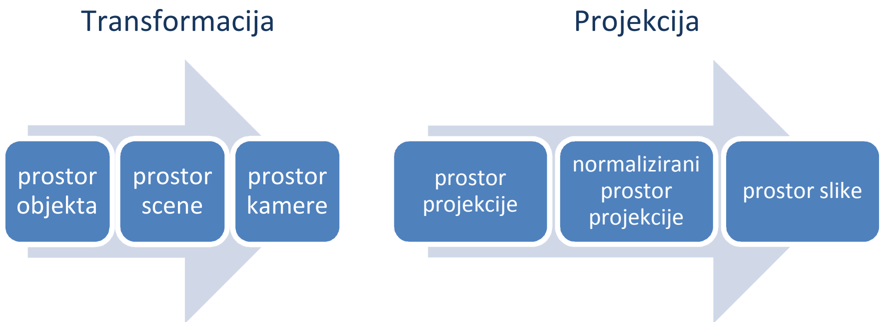
1. Iscrtaj scenu u spremnik okvira, kao da je u potpunosti u sjeni
2. Generiraj volumene sjene
3. Za svako svjetlo, čini sljedeće:
   3.1. Iscrtaj volumene sjene u spremnik šablone
   3.2. Iscrtaj scenu s osvjetljenjem, primjenjujući dobivenu masku
   3.3. Dodaj dobiveno osvjetljenje u spremnik okvira, aditivnim miješanjem

Prednost ovog postupka je što su sjene vrlo točne i precizne, bez obzira na poziciju kamere ili kut gledanja. Dapače, sjene mogu čak izgledati i preoštro, tj. neprirodno, stoga je moguće koristiti zamućivanje. Glavni nedostatak je potreba za generiranjem i iscrtavanjem dodatne geometrije sjena. Tradicionalno, određivanje silueta i proširivanje u volumene se vršilo na glavnom procesoru, no moderne grafičke kartice (DirectX 10 ili novije), nude mogućnost pisanja programa za sjenčanje geometrije (eng. *geometry shader*), a koji omogućuju generiranje volumena sjene na grafičkoj kartici. Zbog masivno paralelne arhitekture grafičkih procesora, ovime se postiže znatno ubrzanje algoritma. Međutim, i dalje ostaje problem iscrtavanja, odnosno rasterizacije volumena sjene pri generiranju spremnika šablone. Naime, budući da se volumeni sjene stvaraju na temelju geometrije scene, i to za svako svjetlo posebno, geometrijska kompleksnost scene ima direktan i značajan utjecaj na performanse. Već i sa samo tri ili četiri izvora svjetla na sceni, govorimo o trostrukom ili četverostrukom povećanju broja vrhova koje je potrebno obraditi. Međutim, poseban je problem što pojedini volumen sjene zauzima mnogostruko veći prostor od objekta koji ga opisuje. Teoretski, volumen sjene je beskonačan, na u praktičnim primjenama odrezuje se na neku dovoljnu dubinu, tako da je pokrivena kompletna scena. Zbog svoje veličine, volumeni sjene zahtijevaju znatan utrošak vremena na rasterizaciju.

S druge strane, algoritam preslikavanja sjena ima bitno drugačija svojstva, budući da se zasniva na drugačijem principu. Naime, ideja je sljedeća: scenu možemo iscrtati kao da je promatramo iz pozicije svjetla, i pritom za svaki fragment odrediti samo dubinu, bez računanja boje, odnosno osvjetljenja. Tako nastali spremnik dubine naziva se mapa sjene (otuda i naziv algoritma). Dakle, u mapi sjene zapravo imamo prikaz reljefa scene, u prostoru svjetla. U sljedećem koraku, scena se iscrtava na standardni način, ali prilikom izračunavanja osvjetljenja potrebna je dodatna provjera zaklonjenosti danog fragmenta, pomoću mape sjene. Tek ukoliko fragment nije zaklonjen, računa se osvjetljenje - u suprotnom slučaju, fragment se ne osvjetljava. Dakako, preostalo je pitanje kako iskoristiti mapu sjene, odnosno, izvršiti provjeru leži li dani fragment u sjeni ili ne. Postoji nekoliko različitih varijanti algoritma koje drugačije rješavaju ovo pitanje. U svom osnovnom, najjednostavnijem obliku, postupak se sastoji od tri koraka:

1. Rekonstrukcija pozicije fragmenta u prostoru kamere
2. Transformacija pozicije fragmenta iz prostora kamere u prostor svjetla
3. Usporedba z komponente pozicije fragmenta s vrijednosti u mapi sjene

Prvi korak je trivijalan u slučaju korištenja unaprijednog iscrtavanja, budući da se vrhovi i fragmenti obrađuju u istom prolazu, pa je moguće jednostavno prenijeti podatak o poziciji vrha iz programa za sjenčanje vrhova u program za sjenčanje fragmenata. U slučaju iscrtavanja s odgodom, potrebno je rekonstruirati poziciju u prostoru kamere iz položaja fragmenta u prostoru slike, te spremnika dubine. Kao kratki podsjetnik, transformacija vrhova geometrije se provodi na sljedeći način:



Slika 14. Tok transformacije i projekcije vrhova u fragmente.

Početno, zadane su koordinate vrhova u lokalnom koordinatnom sustavu objekta kojem ti vrhovi pripadaju. Postavljanjem objekta na scenu, definira se transformacija iz prostora objekta u prostor scene. Konačno, postavljanjem kamere na scenu, definira se transformacija iz prostora scene u prostor kamere, čime je faza transformacije završena. Zatim, slijedi projekcija iz prostora kamere u neki projekcijski volumen, primjerice kvadar za ortogonalnu projekciju, ili (krnju) piramidu za perspektivnu projekciju. Budući da se koriste homogene koordinate *(x, y, z, w)*, potrebno je izvršiti normalizaciju, dijeljenjem s *w* koordinatom, čime se prelazi u normalizirani prostor projekcije, gdje su sve koordinate u intervalu [0, 1]. Napokon, skaliranjem i translacijom pomoću parametara zaslona, prelazi se u prostor slike. Primjerice, za rezoluciju zaslona 800x600, x koordinata u prostoru slike je u intervalu [0, 800], a y koordinata u intervalu [0, 600]. Dakle, za rekonstrukciju pozicije fragmenta u prostoru kamere, potrebno je izvršiti inverznu projekciju, što je najjednostavnije učiniti množenjem koordinata fragmenta u prostoru slike, i dubine, s inverznom matricom projekcije. Nakon transformacije rekonstruirane pozicije fragmenta u prostor svjetla, moguće je provesti sam test zaklonjenosti. Ukoliko je dubina danog fragmenta u prostoru svjetla veća nego vrijednost u mapi sjene, zaključujemo da je fragment zaklonjen, budući da je na danoj poziciji izvoru svjetla bio bliži neki drugi objekt (čija je dubina zapisana u mapi sjene).

Dakle, za razliku od volumena sjena, kod preslikavanja sjena nije potrebno generirati i iscrtavati dodatnu geometriju za svaki izvor svjetla, već je dovoljno iscrtati samo početnu geometriju scene, i pritom generirati samo spremnik dubine, kao zasebnu teksturu za svako svjetlo. Iz tog razloga, preslikavanje sjena općenito pruža bolje performanse od volumena sjena, te se češće koristi u komercijalnim igrama i aplikacijama. Međutim, u svom osnovnom obliku, algoritam pokazuje određene neprihvatljive nedostatke. Naime, budući da je mapa sjena točno određene, konačne rezolucije (primjerice, 512x512 slikovnih elemenata), na sjenama se često pojavljuju nazubljeni rubovi. Problem je dodatno naglašen činjenicom da sjene nije moguće prikazati jednolikom rezolucijom u mapi sjena - nakon transformacije pozicije fragmenta u prostor svjetla, moguće je da veći broj fragmenata (šira regija u prostoru slike) upada na istu (užu) regiju u mapi sjene, što dakako, ovisi o poziciji i upadnom kutu svjetla. Ovo je poseban pod-problem, za kojeg je razvijeno nekoliko rješenja. Primjerice, kod trapezoidalnih mapa sjena (eng. *trapezoidal shadow maps*), algoritam pronalazi optimalnu projekciju scene na mapu sjena, iz pogleda izvora svjetla, i time osigurava bolju iskoristivost razlučivosti mape sjena. Drugačiji pristup, koji je posebice pogodan za vrlo velike, otvorene scene, su kaskadirane mape sjena (eng. *cascaded shadow maps*), a koji se sastoji od particioniranja scene na više regija, i to tako da je svaka regija prikazana zasebnom mapom sjena. Sve mape sjena unutar kaskade su jednake rezolucije, ali regije zauzimaju progresivno veći prostor što su dalje od promatrača, tako da je u neposrednoj blizini promatrača broj slikovnih elemenata mape sjene po jedinici prostora najveći, te s udaljenosti taj broj opada. Dakle, ideja je slična tehnici razine detalja (eng. *level of detail*, *LoD*) kod iscrtavanja udaljene geometrije.

Međutim, čak i uz dovoljnu veličinu i razlučivost mape sjena, rubovi sjena će i dalje biti oštri, što izgleda neprirodno. Prema tome, potrebno je zamutiti rubove sjena i tako dobiti mekše, prirodnije sjene - drugim riječima, potrebno je izvršiti filtriranje mape sjena. Upravo u ovom dijelu razlikuju se različite varijante osnovnog algoritma. Tradicionalan pristup, koji je bio sklopovski implementiran na grafičkim karticama (na sličan način kao sklopovsko izglađivanje nazubljenih rubova poligona), bazira se na uzorkovanju i testiranju nekoliko susjednih uzoraka u mapi sjena (eng. *percentage closer filtering*, *PCF*). Dakle, PCF algoritam proširuje osnovno preslikavanje sjena tako da provjeru zaklonjenosti fragmenta izvrši nekoliko puta, i to s različitim, susjednim elementima mape sjena. Uzorci se obično uzimaju u pravilnoj rešetki, primjerice 4x4, odnosno, ukupno 16 uzoraka po fragmentu. Rezultati svih usporedbi se pomnože odgovarajućim težinskim faktorom te zbroje, i time se dobije konačan rezultat koji više nije binarna vrijednost (zaklonjen / nije zaklonjen), već decimalan broj u intervalu [0, 1], gdje nula predstavlja potpunu sjenu, a jedinica potpunu osvijetljenost. Dakle, ono što zapravo dobijemo kao rezultat PCF algoritma jest postotak uzoraka unutar regije uzorkovanja čije je dubina veća od one u mapi sjena. Prema tome, jasno je da rubovi sjena više neće biti oštri (niti nazubljeni!), već blago zamućeni. Dakako, cijena ovog algoritma je potreba za dodatnim, višestrukim uzorkovanjem, koje je potrebno provoditi pri svakoj provjeri.

No, postoji još jedna, relativno nova varijanta algoritma preslikavanja sjena, koja pruža bolje performanse uz jednaku ili čak bolju kvalitetu. Riječ je o preslikavanju sjena temeljenom na izračunu varijance dubine scene zapisane u mapi sjena (eng. *variance shadow maps*, *VSM*). Naime, umjesto jednostavne usporedbe dubine fragmenta s vrijednosti u mapi sjena, vrši se analiza distribucije vrijednosti dubina u mapi sjena. Ideja se zasniva na opažanju da kod PCF algoritma, ono što želimo dobiti kao rezultat jest udio zaklonjenih uzoraka u skupu svih uzoraka neke regije. Drugim riječima, uspoređujemo vrijednost dubine danog fragmenta, sa skupom vrijednosti dubina uzoraka. Ovako postavljen problem veoma podsjeća na definiciju funkcije distribucije vjerojatnosti, koja se matematički formulira na sljedeći način:

$$x \to F_X(x) = P(X \leq x) \quad (1)$$

Ukoliko bi distribucija vrijednosti dubina u mapi sjena bila poznata, onda usporedbe pojedinih uzoraka ne bi bile potrebne - dovoljno bi bilo tek evaluirati funkciju distribucije vjerojatnosti nad danim fragmentom. Jasno, problem je što je distribucija nepoznata, pa je to nemoguće - mogli bismo pretpostaviti neku funkciju distribucije, no to vrlo vjerojatno ne bi proizvelo zadovoljavajuće rezultate za proizvoljne scene. Međutim, sada se valja prisjetiti Čebiševljeve nejednakosti, koja kaže da za bilo koju distribuciju vjerojatnosti, većina vrijednosti će biti blizu očekivane vrijednosti. Točnije, ne više od $\frac{1}{k^2}$ vrijednosti će biti udaljeno više od k standardnih devijacija od očekivane vrijednosti. Dakle, ovim teoremom možemo odrediti gornju granicu vrijednosti funkcije distribucije vjerojatnosti $F_X(x) = P(X \leq x)$ za danu vrijednost *x*. Drugim riječima, možemo odrediti gornju granicu udjela zaklonjenih uzoraka u nekom skupu. Korisnost Čebiševljeve nejednakosti leži u činjenici da je primjenjiva na proizvoljnu, nepoznatu distribuciju, za koju je poznato barem očekivanje i varijanca. U našem slučaju, očekivanje se može jednostavno definirati kao aritmetička sredina svih uzoraka. Varijanca se može trivijalno izvesti iz očekivanja:

$$\sigma^2 = E(x^2) - E(x)^2 \quad (2)$$

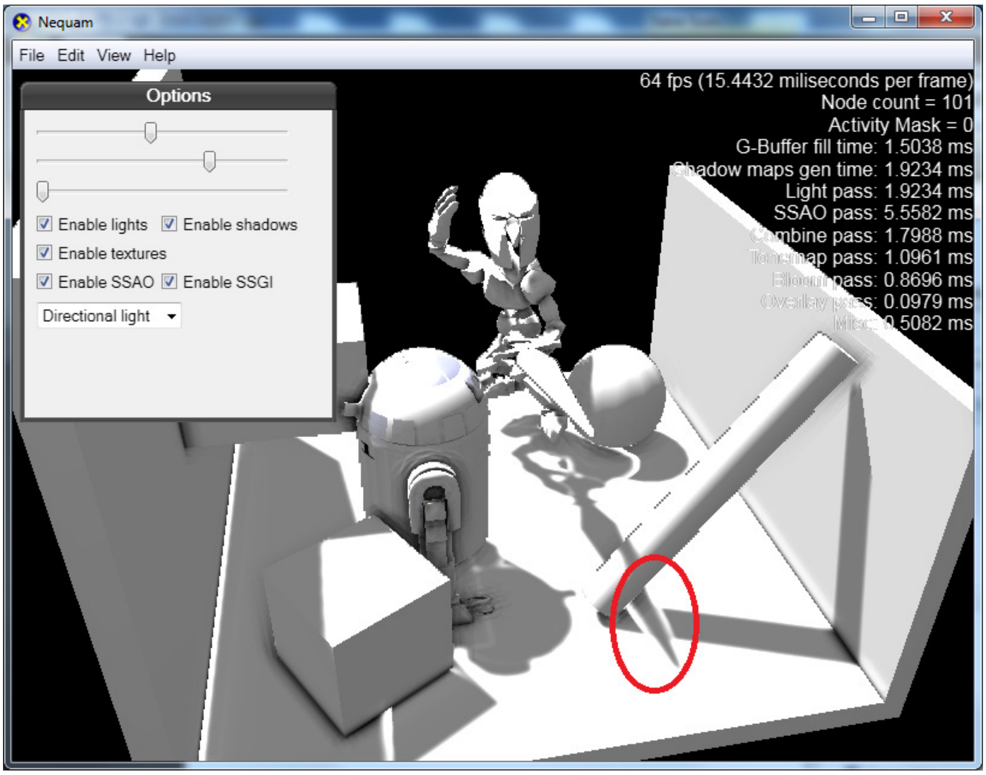
Konačno, Čebiševljevu nejednakost možemo formulirati na sljedeći način:

$$P(x \geq t) = \frac{\sigma^2}{\sigma^2 + (t - E(x))^2}, \qquad \text{kada } t > E(x) \quad (3)$$

U našem slučaju, t će biti dubina promatranog fragmenta, $E(x)$ aritmetička sredina vrijednosti dubina neke regije uzorkovanja, a σ^2 varijanca te regije. Prema tome, skupo višestruko uzorkovanje PCF algoritma se zamjenjuje evaluacijom izraza (3). Dakako, jedino preostalo pitanje je kako točno izračunati i pohraniti očekivanje i varijancu. Ovaj korak je jednostavan - dovoljno je proširiti teksturu mape sjena tako da sadrži dva kanala, umjesto samo jednog. Prvi kanal, kao i dosad, sadrži dubinu scene dobivenu iscrtavanjem iz pogleda svjetla. U drugi kanal, zapisuje se kvadrat te dubine. Zatim, potreban je podprolaz zamućivanja mape sjena, jednostavnim zbrajanjem uzoraka i podjelom s njihovim brojem, koristeći primjerice rešetku uzorkovanja dimenzija 15x15 elemenata. Jasno, ovim zamućivanjem zapravo dobivamo očekivanje, te budući da u teksturi pohranjujemo i dubinu, i kvadrat dubine, odmah dobivamo $E(x)$ u prvom kanalu, te $E(x^2)$ u drugom kanalu teksture. Zatim, prilikom izračuna osvjetljenja, $E(x)$ i $E(x^2)$ se jednostavno očitaju iz mape sjena te se trivijalno odredi varijanca prema izrazu (2), nakon čega je moguće evaluirati formulu (3), ukoliko je zadovoljen uvjet da je dubina fragmenta veća od očekivanja. U suprotnom, jednostavno se uzima $P(x \geq t) = 1$.

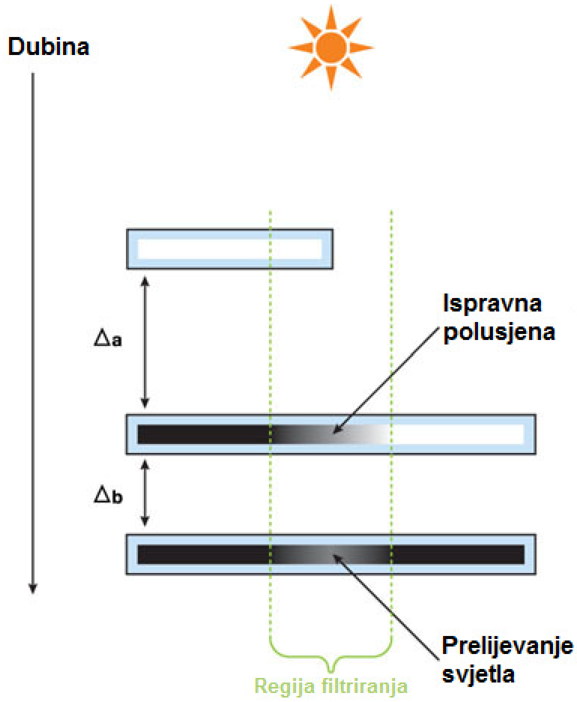
Usporedbom PCF i VSM algoritma, možemo zaključiti da oba teže k istom rezultatu, no drugačijim postavljanjem problema kod VSM algoritma, omogućeno je obavljanje filtriranja mape sjena u zasebnom pred-koraku. Prava prednost ovog pristupa jest činjenica da je sada moguće primijeniti separabilnu jezgru zamućivanja. Naime, kod PCF algoritma, budući da se filtriranje obavlja u istom prolazu kao i izračun osvjetljenja, dakle u jednom prolazu, kompleksnost je kvadratna - za 4x4 rešetku, potrebno je obaviti 16 uzorkovanja. Međutim, ukoliko filtriranje obavljamo kao zaseban prolaz, moguće ga je razbiti u dva podprolaza, tako da u prvom podprolazu zamućujemo originalnu teksturu samo u vertikalnom smjeru, a u drugom podprolazu zamućujemo vertikalno zamućenu teksturu u horizontalnom smjeru. Na taj način, kompleksnost filtriranja postaje linearna, budući da je ukupan broj koraka jednak zbroju vertikalne i horizontalne dimenzije rešetke uzorkovanja. Primjerice, za 15x15 rešetku, separabilnim filtriranjem kod VSM algoritma dovoljno je 30 uzorkovanja po fragmentu, dok bi jednoprolaznim filtriranjem kod PCF algoritma bilo potrebno čak 225 uzorkovanja, što nije praktički izvedivo ni na modernim grafičkim karticama. Prema tome, VSM algoritmom možemo uz jednake performanse postići mnogo kvalitetnije filtriranje sjena - ili, uz jednaku kvalitetu, mnogo bolje performanse nego kod PCF algoritma.

Međutim, u ovom trenutku treba spomenuti i određene artefakte koji se javljaju primjerice kod dvostrukih sjena, tj. dijela prostora koji je zaklonjen dvjema različitim objektima.



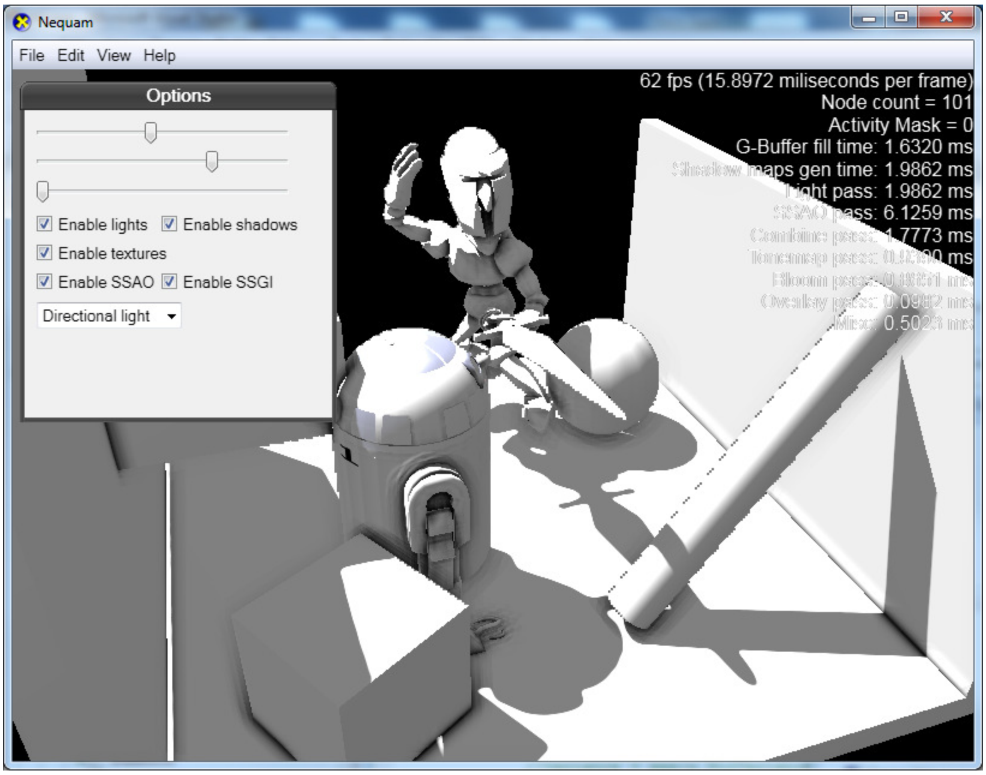
Slika 15. Primjer artefakta u VSM-u kod dvostrukih sjena.

Kao što se vidi na Slici 15, u takvim situacijama dolazi do prelijevanja svjetla (eng. *light bleeding*), budući da se meki rub sjene preslikava dijelom na prvi zaklonjeni objekt (sjena mača na položenoj cijevi), a dijelom na drugi zaklonjeni objekt (sjena mača na podu, na mjestu gdje sječe sjenu cijevi). Razlog zašto se ovo događa, jest taj što unutar iste regije filtriranja upadaju elementi mape sjena generirani i prvim i drugim objektom, pa zbog zamućivanja oba postaju sivi (polusjena). Ova se pojava najlakše objašnjava sljedećom slikom:



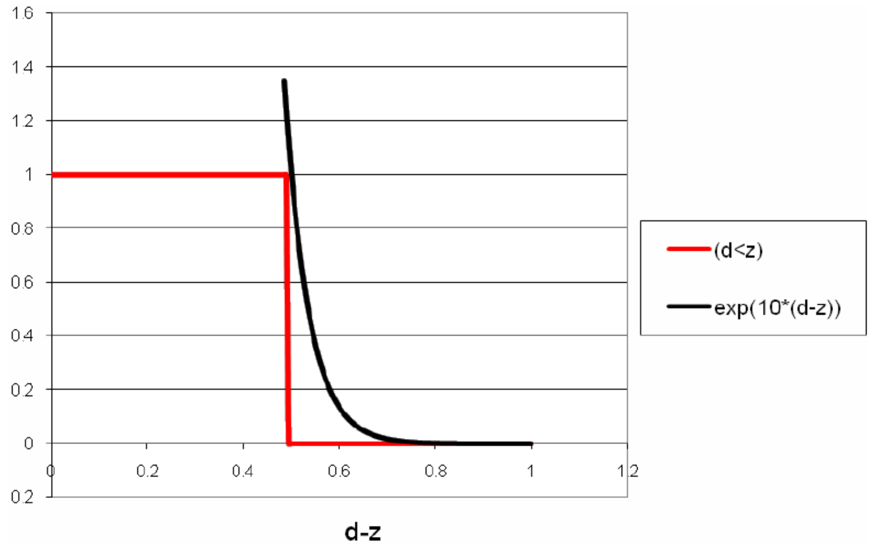
Slika 16. Pojava pogrešnog preslikavanja polusjene kod VSM algoritma.

Srećom, postoji jednostavno rješenje ovog problema koje je gotovo trivijalno (i u smislu implementacije, i u smislu računalne zahtjevnosti), a pruža dovoljno dobre rezultate za većinu stvarnih primjena. Naime, ukoliko ponovno razmotrimo Čebiševljevu nejednakost, možemo uočiti da ukoliko je dani fragment u potpunosti zaklonjen, vrijedit će $t > E(x)$. Prema tome, vrijedi $(t - E(x))^2 > 0$, a što znači da je konačno $P(x \geq t) < 1$. Drugim riječima, polusjene (točne ili netočne) nikada neće doseći punu zaklonjenost $P(x \geq t) = 1$ dok je zadovoljen uvjet $t > E(x)$, dakle na potpuno zaklonjenim fragmentima. Iz navedenog slijedi jednostavna ideja - možemo se riješiti netočnih osvijetljenih područja gdje je $P(x \geq t)$ blizak nuli, tako da definiramo prag P_{min}, kao korisničku varijablu. Nakon evaluacije izraza (3), odrezat ćemo dobivenu vrijednost ukoliko upada u interval [0, $Pmin$], te skalirati s intervala ($Pmin$, 1] natrag na [0, 1]. Rezultat jest taj, da će sve polusjene općenito postati tamnije, što znači da će doći do gubitka detalja. No, pažljivim namještanjem parametra $Pmin$ za danu scenu, moguće je postići da artefakti gotovo u potpunosti nestanu, uz tek neznatan gubitak detalja kod polusjena. Primjer identične scene kao u slici 15, ali uz navedenu korekciju polusjena dan je slikom 17.



Slika 17. VSM algoritam s ispravnim dvostrukim sjenama.

Na kraju, valja istaknuti kako je koncept preslikavanja sjena predstavljen još 1978 godine (Lance Williams), no ovo područje računalne grafike se još uvijek aktivno istražuje. Primjerice, jedan od novijih pokušaja [Annen07] su konvolucijske mape sjena (eng. *convolution shadow maps, CSM*), koje se bazira na opažanju da je funkcija testa zaklonjenosti zapravo step funkcija $f(z) = \begin{cases} 1, (z-d) > 0 \\ 0, (z-d) \leq 0 \end{cases}$, gdje je z dubina fragmenta, a d dubina u mapi sjena. Ideja algoritma jest, da je ovu funkciju moguće aproksimirati konačnim Furierovim redom. Dakle, CSM algoritam zahtjeva n-komponentnu mapu sjena (gdje je n broj članova Furierovog reda), tako da je u svakoj komponenti kodiran pojedini član reda. Na taj način, opet je omogućeno separabilno predfiltriranje mape sjena. Premda CSM algoritam ne pokazuje problem prelijevanja svjetla, dolazi do drugih artefakata zbog aproksimacije nepotpunim Furierovim redom, koje je mnogo teže maskirati. Dakako, takve artefakte je moguće umanjiti uzimanjem većeg broja članova, no to povlači i povećanu potrošnju memorije (svaki dodatni član zahtjeva dodatni teksturni kanal), te zahtjevniji korak predfiltriranja (budući da je potrebno obraditi veći broj teksturnih kanala). Slična, novija ideja [Salvi08] jest aproksimacija funkcije zaklonjenosti (step funkcije) pomoću jednostavne eksponencijalne funkcije (eng. *exponential shadow maps, ESM*). Naime, sljedeći graf zorno prikazuje ovu ideju:



Slika 18. Numerička aproksimacija step funkcije eksponencijalnom funkcijom.

Kao što gornja slika prikazuje, funkcija oblika $e^{\alpha(d-z)}$ može biti dovoljno "slična" step funkciji za odgovarajuću konstantu α. Dakle, slično kao kod CSM algoritma, ESM zahtjeva pohranu dubine scene prikazane aproksimacijskom funkcijom u mapi sjena. No, ovdje je razlika što je dovoljan tek jedan teksturni kanal, što znači smanjeni utrošak memorije, te brže filtriranje. I kod ESM algoritma se mogu javiti artefakti, primjerice pojava svijetlih regija - na slici 17, step funkcija opada trenutačno u nulu nakon prelaska određenog praga, no eksponencijalna funkcija postepeno opada, te je u određenom intervalu iza praga još uvijek iznad nule. Ovaj se problem može riješiti pažljivim namještanjem parametra α za danu scenu, dakle slično kao kod VSM algoritma.

Zbog svojih odličnih svojstava, preslikavanje sjena se nametnulo kao standardna metoda za rješavanje direktnih sjena. Danas možda najpopularnija inačica algoritma jest VSM, budući da je to prva metoda koja je omogućila separabilno pred-filtriranje, što je rezultiralo znatnim poboljšanjem i kvalitete i performansi, te je algoritam vrlo dobro dokumentiran i prihvaćen od strane razvijatelja. No, ovo se područje i dalje intenzivno razvija, te već postoje nova, obećavajuća poboljšanja kao što je ESM algoritam.

7. Model globalnog osvjetljenja

U prethodnim poglavljima, dan je pregled fizikalnih modela svjetlosti (geometrijskog i valno-čestičnog), te je odabran geometrijski model kao standard fotorealističnosti za računalnu grafiku. Predstavljeni su lokalni modeli osvjetljenja koji se tradicionalno koriste u interaktivnoj računalnoj grafici, kao što je Blinn-Phong model. Takvi, lokalni modeli su grube aproksimacije kompletne jednadžbe iscrtavanja, budući da ignoriraju odbijanje svjetlosti između objekata, i time zanemaruju razne suptilne efekte indirektnog osvjetljenja poput pretapanja boja uslijed difuzne interrefleksije. Ambijentalna komponenta osvjetljenja se u lokalnim modelima predstavlja tek jednom konstantom, te se računaju jedino direktni doprinosi osvjetljenja. Pokušajmo sada proširiti lokalni model osvjetljenja formulirajući kompletnu jednadžbu iscrtavanja. Možemo razmišljati ovako: znamo da je ukupna izlazna svjetlost u bilo kojoj točki prostora jednaka zbroju emitirane svjetlosti (što može biti nula, ukoliko nije riječ o izvoru svjetla), te doprinosima svih svjetlosnih zraka koje upadaju na tu točku, a mogu biti rezultat direktnog osvjetljenja nekim izvorom svjetlosti, ili pak rezultat (višestruke) refleksije svjetlosti od drugih objekata na sceni. Također, znamo da se svjetlost reflektira od različitih objekata na drugačiji način, pa možemo definirati funkciju distribucije reflektivnosti (BRDF funkcija, vidjeti poglavlje 3). Sve dosad rečeno možemo formalno zapisati na sljedeći način:

$$L_o(x,\omega) = L_e(x,\omega) + \int_{\Omega} f_r(x,\omega',\omega) L_i(x,\omega')(\omega' \cdot n)\, d\omega' \quad (1)$$

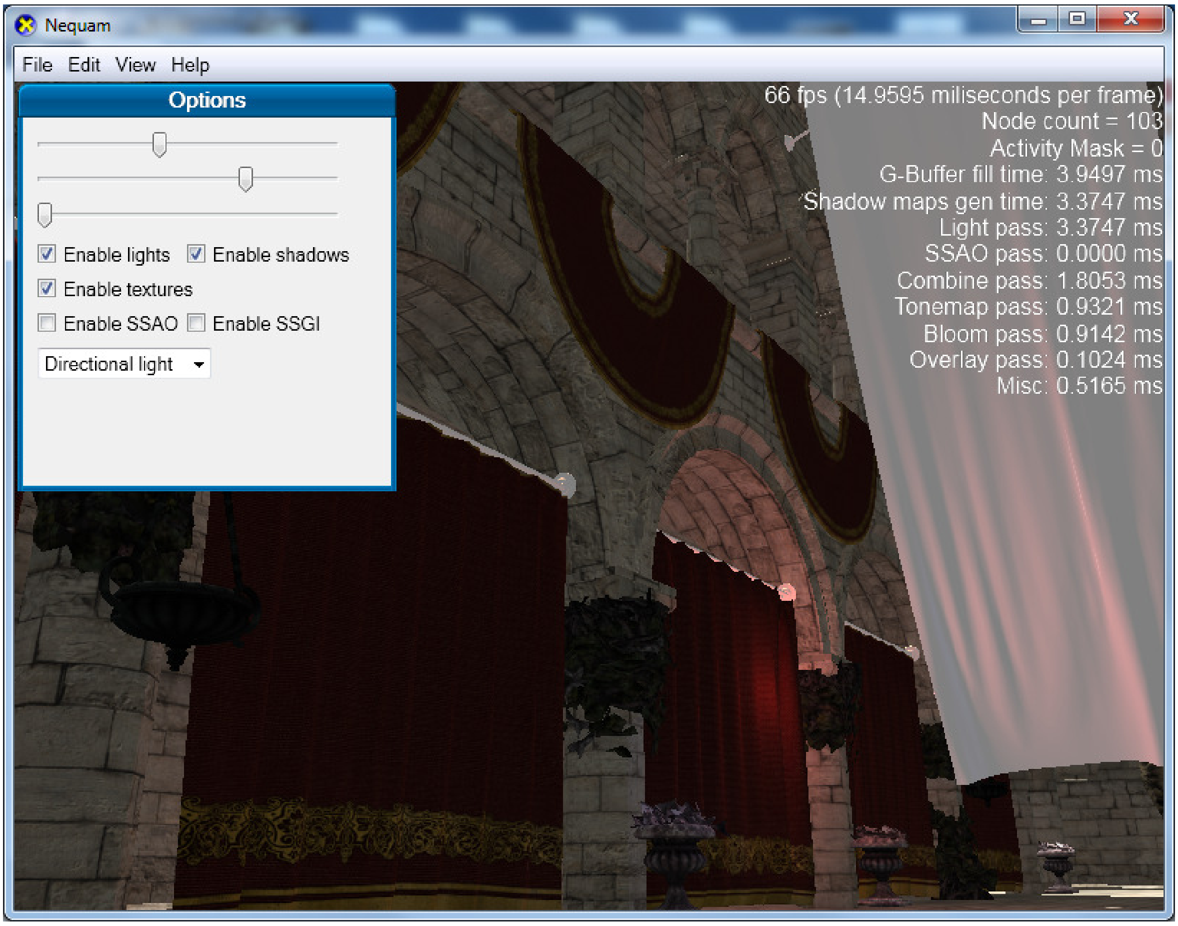
gdje je:

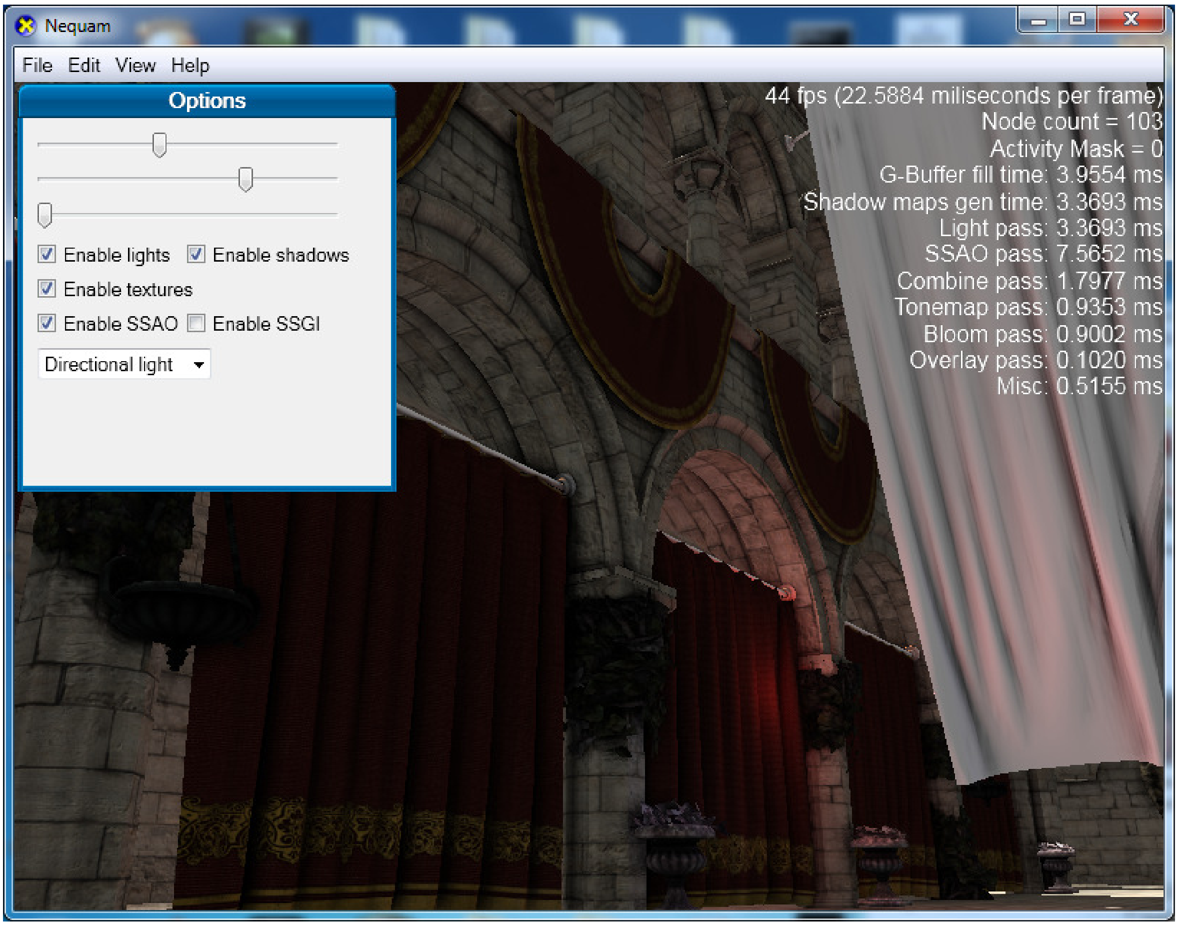
$L_o(x,\omega)$	izlazna svjetlost na poziciji x, u smjeru ω,
$L_e(x,\omega)$	svjetlost emitirana sa pozicije x, u smjeru ω,
$\int_{\Omega} \dots d\omega'$	integral upadnih zraka svjetlosti preko polukugle,
$f_r(x,\omega',\omega)$	mjera svjetlosti reflektirane na poziciji x, iz upadnog prema izlaznom smjeru (BRDF funkcija)
$L_i(x,\omega')$	upadna svjetlost na poziciji x, u smjeru ω',
$\omega' \cdot n$	atenuacija upadnog svjetla zbog kuta upada, gdje je n normala na površinu.

Dakako, gornju je jednadžbu nemoguće analitički riješiti, budući da funkcija upadne svjetlosti ovisi o geometrijskim međuodnosima objekata na sceni, te bi u simboličkom obliku bila prekompleksna za praktičnu uporabu, osim u vrlo jednostavnim, specijalnim slučajevima. Primjerice, jedan takav slučaj dobijemo pretpostavkom difuznog osvjetljenja bez interrefleksije - time se jednadžba iscrtavanja reducira na Lambertov kosinusni zakon, koji predstavlja osnovu lokalnih modela osvjetljenja. Iz navedenog razloga, svi algoritmi kojima je cilj doći do kompletnog rješenja jednadžbe iscrtavanja se zasnivaju na nekim numeričkim metodama i heurističkim aproksimacijama. Pritom možemo razlikovati dvije skupine algoritama - tradicionalni pristupi poput praćenja zrake, preslikavanja fotona i algoritma isijavanja (za detaljniji pregled ovih tehnika pogledati [Sajko08]), te noviji pristupi poput difuzne interrefleksije u prostoru slike, predizračunatog prijenosa zračenja te propagacijskih volumena svjetla. Tradicionalne metode su razvijene s ciljem čim točnijeg rješenja problema osvjetljenja, te zbog svoje zahtjevnosti sve do nedavno nisu bile primjenjive u interaktivnoj računalnoj grafici. No, rastom moći računala i specifično, grafičkih kartica, koje postaju gotovo autonomni, masivno paralelni sustavi primjenjivi za rješavanje općenitih problema, otvaraju se i mogućnosti primjene postupaka poput praćenja zrake u interaktivnoj računalnoj grafici. Međutim, premda su takve implementacije već ostvarene uz interaktivne brzine iscrtavanja, ovo se područje još uvijek smatra eksperimentalnim, te većina komercijalnih grafičkih aplikacija i igara koristi metode prilagođene radu s rasteriziranim podacima, što je još uvijek dominantan pristup u interaktivnoj računalnoj grafici.

8. Ambijentalno zaklanjanje u prostoru slike

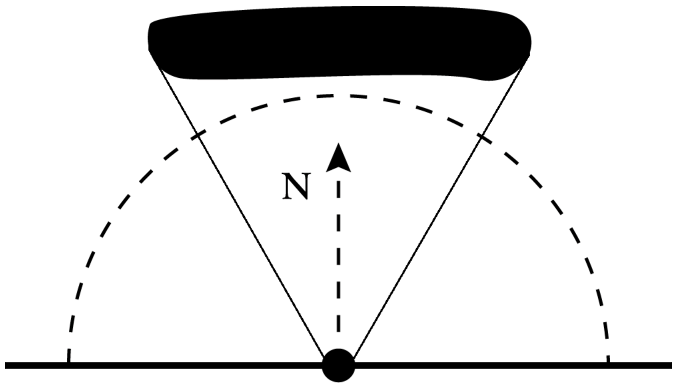
U poglavlju 6 predstavljene su metode za određivanje direktnih sjena, koje nastaju kao rezultat zaklanjanja direktnog izvora svjetla. Međutim, jedan od učinaka globalnog osvjetljenja, kakvo se javlja u stvarnom svijetu, jest indirektno, ambijentalno osvjetljenje, koje nastaje difuznom interrefleksijom. Dakako, i indirektno osvjetljenje može biti zaklonjeno, pa govorimo o ambijentalnom zaklanjanju, čiji su rezultat suptilne, meke sjene koje naglašavaju primjerice uglove soba, pukotine u objektima, područja dodira dvaju objekata itd. Ovaj je efekt dovoljno suptilan da ga u stvarnom svijetu svjesno ne primjećujemo, no njegov nedostatak kod računalne grafike je primjetan te odaje umjetno generiranu sliku. Kao primjer ovog učinka, dana je slika 18.





Slika 18. Gornja slika je iscrtana bez ambijentalnog zaklanjanja, a donja sa.

Konkretno, implementirani algoritam jest ambijentalno zaklanjanje u prostoru slike (eng. *screen space ambient occlusion, SSAO*). Ova relativno nova tehnika prvi je puta predstavljena na SIGGRAPH-u 2007. godine, a zanimljivost algoritma jest što se zaklonjenost izračunava isključivo u prostoru slike, slično kao računanje osvjetljenja pri iscrtavanju s odgodom (vidjeti poglavlje 4). U osnovi, ideja ambijentalnog zaklanjanja je jednostavna. Naime, indirektno difuzno osvjetljenje se od svake točke površine objekta širi jednoliko u svim smjerovima, dakle u hemisferi. Prema tome, ambijentalnu zaklonjenost neke točke možemo definirati kao udio površine hemisfere prekriven nekim drugim objektom, kako je prikazano slikom 19.



Slika 19. Ilustracija ambijentalnog zaklanjanja.

Formalno, možemo prikazati ambijentalnu zaklonjenost A_p kao integral funkcije vidljivosti $V_{p,\omega}$ po hemisferi Ω:

$$A_p = \frac{1}{\pi}\int_{\Omega} V_{p,\omega}\,(N\cdot\omega)\,d\omega \quad (1)$$

gdje je, dakako, N normala na površinu, a ω kut integracije. Funkcija vidljivosti jest binarna funkcija čija je vrijednost nula ukoliko je u danoj točki p za dani kut ω vidljiv neki drugi objekt, a jedan inače. Početna ideja SSAO algoritma, kakav je opisan u radovima [Mittring07] i [Filion08], jest numerička aproksimacija integrala danog izrazom (1). Naime, možemo aproksimirati integral sa sumacijom:

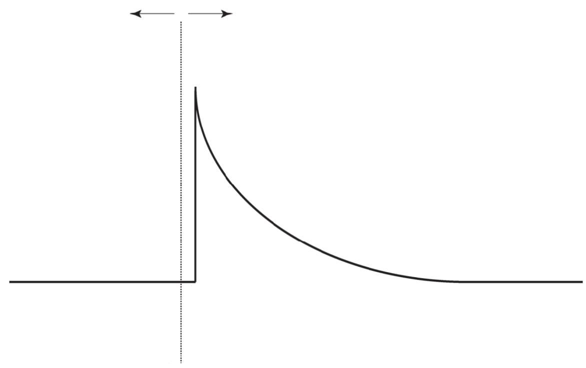
$$A_p \approx \frac{1}{C}\sum\nolimits_{\Omega} V_{p,\omega}\,(N\cdot\Delta\omega) \quad (2)$$

Sada je dovoljno uzorkovati funkciju vidljivosti na neki odgovarajući način, te zbrojiti doprinose svih uzoraka. Valja primijetiti kako je ovaj problem moguće riješiti na vrlo elegantan i prirodan način, u okviru postupka praćenja zrake. Naime, funkciju vidljivosti lako odredimo testiranjem presijecanja zrake s objektima na sceni. Za danu točku, ukoliko generirana zraka ne presijeca nikakve druge objekte, izračuna se doprinos ambijentalnog osvjetljenja zrake (primjerice u okviru Blinn-Phong modela). U suprotnom, ukoliko zraka presijeca neki objekt, znači da je promatrana točka zaklonjena, pa se doprinos ambijentalnog osvjetljenja zrake postavlja na nulu. Doprinosi svih zraka generiranih za pojedinu točku se zbroje, i konačni rezultat jest taj da su zaklonjena područja tamnija, a nezaklonjena svjetlija. Nažalost, sam postupak praćenja zrake još uvijek ne pruža zadovoljavajuće performanse za primjenu u kompleksnim, interaktivnim aplikacijama, te je stoga potreban pristup primjereniji rasterizacijskom cjevovodu modernih grafičkih kartica.

Jedan takav pristup zasniva se na proširenju tehnike iscrtavanja s odgodom. Naime, kako je objašnjeno u poglavlju 4, kod iscrtavanja s odgodom, u prvom se prolazu samo izvlače potrebni podaci iz geometrijskog opisa scene, i time generiraju G-spremnici. Podaci koji će nas zanimati u okviru SSAO algoritma su normala i dubina po fragmentu, što u predloženoj konfiguraciji (slika 9) odgovara spremniku MRT2. Za izbjegavanje raznih artefakata koji bi mogli nastati uslijed nedovoljne preciznosti spremnika normala i dubina, potrebno je koristiti 32-bitnu preciznost po kanalu, što je još jedan razlog korištenja HDR osvjetljenja (vidjeti poglavlje 5). Međutim, prirodno se postavlja pitanje - kako uzorkovati funkciju vidljivosti u hemisferi nad površinom objekata, ukoliko samo imamo pristup podacima scene u prostoru slike? Zapravo, ono što je potrebno napraviti jest transformirati poziciju fragmenta iz prostora slike u prostor kamere. Istovjetan problem se javlja i kod algoritma preslikavanja sjena, te je njegovo rješenje opisano u poglavlju 6. Sada se već nazire osnovni princip algoritma - za dani fragment, rekonstruirat ćemo poziciju u prostoru kamere, te odabrati određeni broj uzoraka u hemisferi oko te rekonstruirane pozicije. Dakako, još je preostalo pitanje kako točno odrediti funkciju vidljivosti, odnosno, kako utvrditi da li pojedini dobiveni uzorak zaklanja dani fragment. Prihvatljivu aproksimaciju rješenja ovog problema možemo dobiti analizom spremnika dubine. Naime, ukoliko većina uzoraka ima manju dubinu od promatranog fragmenta, to znači da se ti uzorci nalaze bliže promatraču, odnosno, zaključujemo da se dani fragment nalazi u udubini ili pukotini nekog objekta, te stoga očekujemo da je u većoj mjeri zaklonjen nego okolni uzorci. U suprotnom slučaju, možemo očekivati da je promatrani fragment slabo ili nikako zaklonjen. Dakle, jednom kad odaberemo odgovarajući broj uzoraka u prostoru kamere, projicirat ćemo ih natrag u prostor slike, te očitati njihovu dubinu iz spremnika. Zatim, definirat ćemo neku funkciju zaklonjenosti kojom ćemo moći izraziti doprinos pojedinih uzoraka. Ta funkcija može biti proizvoljnog oblika, ali bi trebala zadovoljavati sljedeća svojstva:

- Negativne razlike dubina ne pridonose zaklonjenosti.
- Manje razlike dubina daju veću zaklonjenost.
- Doprinos zaklonjenosti mora pasti na nulu nakon nekog praga.

Oblik funkcije s navedenim svojstvima prikazan je slikom 20.

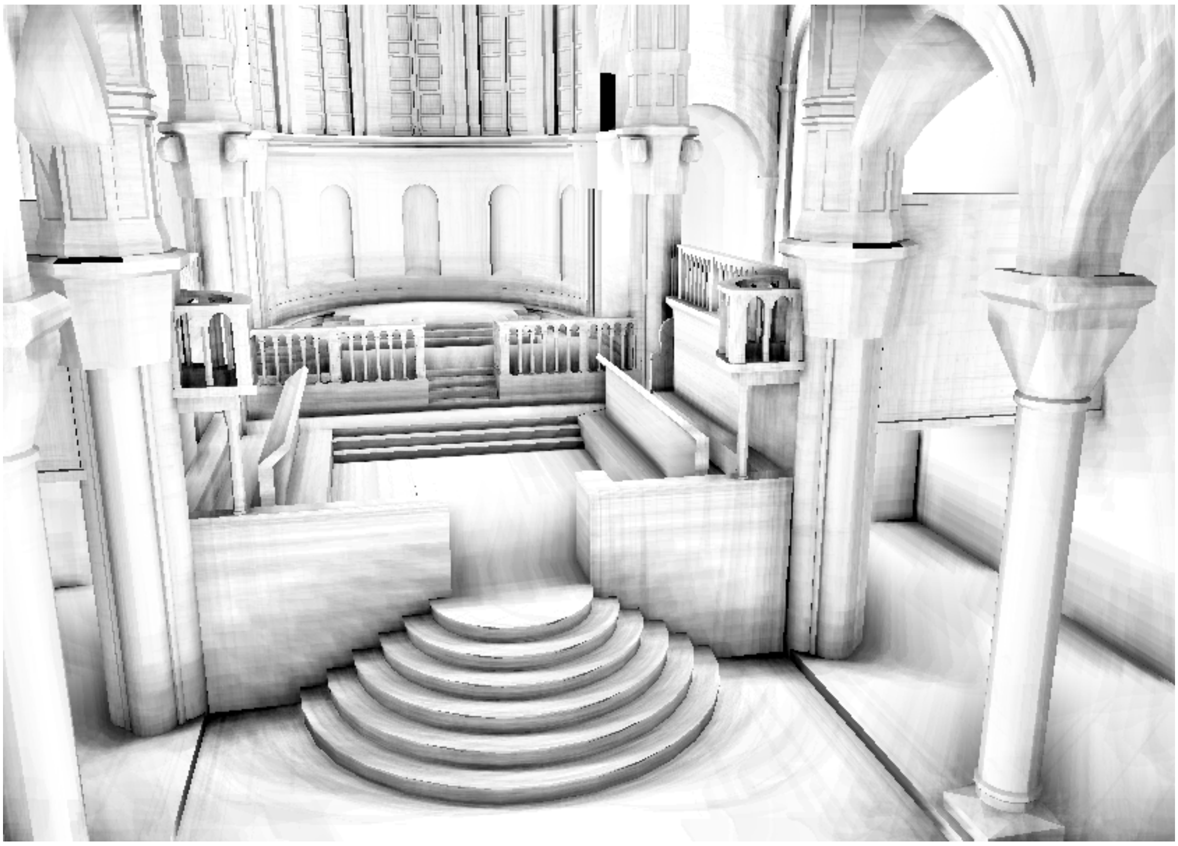


Slika 20. Općeniti oblik funkcije zaklanjanja.

Upravo opisani postupak predstavlja originalni SSAO algoritam. Da rezimiramo, konačni postupak je sljedeći:

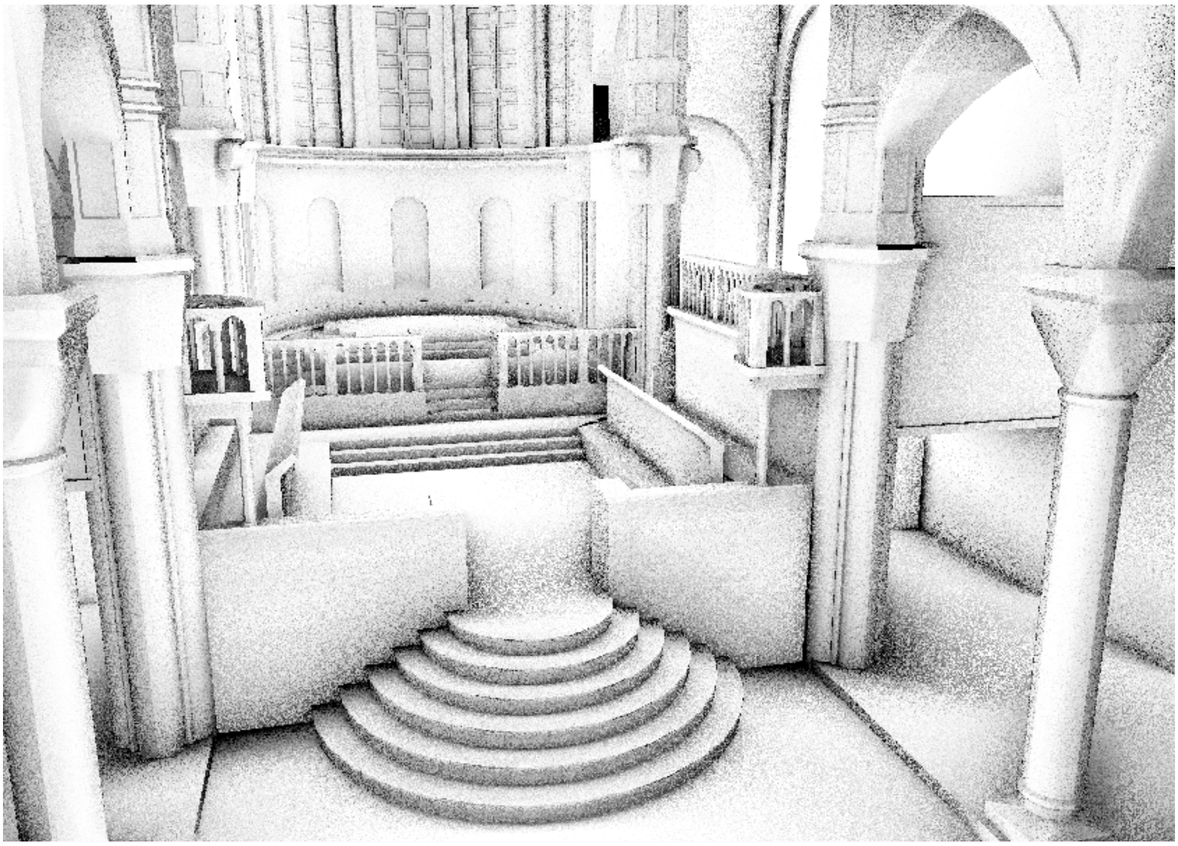
1. Rekonstruiraj 3D poziciju fragmenta (u prostoru kamere).
2. Odaberi 8-32 slučajna uzorka u 3D prostoru, u hemisferi oko fragmenta.
3. Projiciraj uzorke natrag u prostor slike.
4. Odredi dubine uzoraka.
5. Izračunaj funkciju zaklonjenosti za pojedine uzorke i pribroji doprinose.

Premda je SSAO algoritam konceptualno jednostavan, postoji mnogo detalja o kojima je potrebno voditi računa prilikom implementacije. Primjerice, spomenuto je da je uzorke potrebno odabirati na slučajan način. Naime, ukoliko bismo uzorkovali hemisferu oko danog fragmenta na neki predodređeni način, istovjetno za svaki fragment, došlo bi do pojave prekrivanja uslijed poduzorkovanja (aliasing), te vrlo uočljivih artefakata, kako pokazuje slika 21:



Slika 21. Primjer artefakata koji nastaju kod uzorkovanja pravilnim redoslijedom.

Idealno, bilo bi potrebno za svaki fragment generirati jedinstvene uzorke, bez ikakvog ponavljanja. Za 32 uzorka po fragmentu, uz rezoluciju 1920x1080, bilo bi potrebno generirati ukupno 66355200 različitih 3-komponentnih vektora, što bi uz preciznost od samo 8 bitova po komponenti zahtijevalo otprilike 190 MB. Jasno, ovakav pristup nije primjenjiv za stvarne aplikacije, no postoji drugačiji način. Naime, nastavljajući se na gornji primjer, umjesto generiranja svih 66 milijuna različitih slučajnih vektora, dovoljno je generirati samo 32 (tj. onoliko koliko uzimamo uzoraka po fragmentu), i te vektore prenijeti kao konstante u program za sjenčanje fragmenata. Zatim, potrebno je na neki način pripremiti teksturu šuma, dakle teksturu čiji je svaki slikovni element neke slučajne boje (moguće je upotrijebiti neki grafički alat, ili programski generirati odgovarajuću teksturu). Tekstura šuma treba biti malenih dimenzija, primjerice 64x64 slikovnih elemenata. Trik se zatim sastoji u reflektiranju konstantnog slučajnog vektora od vektora u teksturi šuma, za svaki uzorak, što zapravo rezultira rotacijom uzoraka u slučajnom smjeru. Dakle, rezultat je taj, da vektore uzoraka rotiramo za svaki fragment u nekom slučajnom smjeru, čime se razbija pravilnost u uzorkovanju između fragmenata.

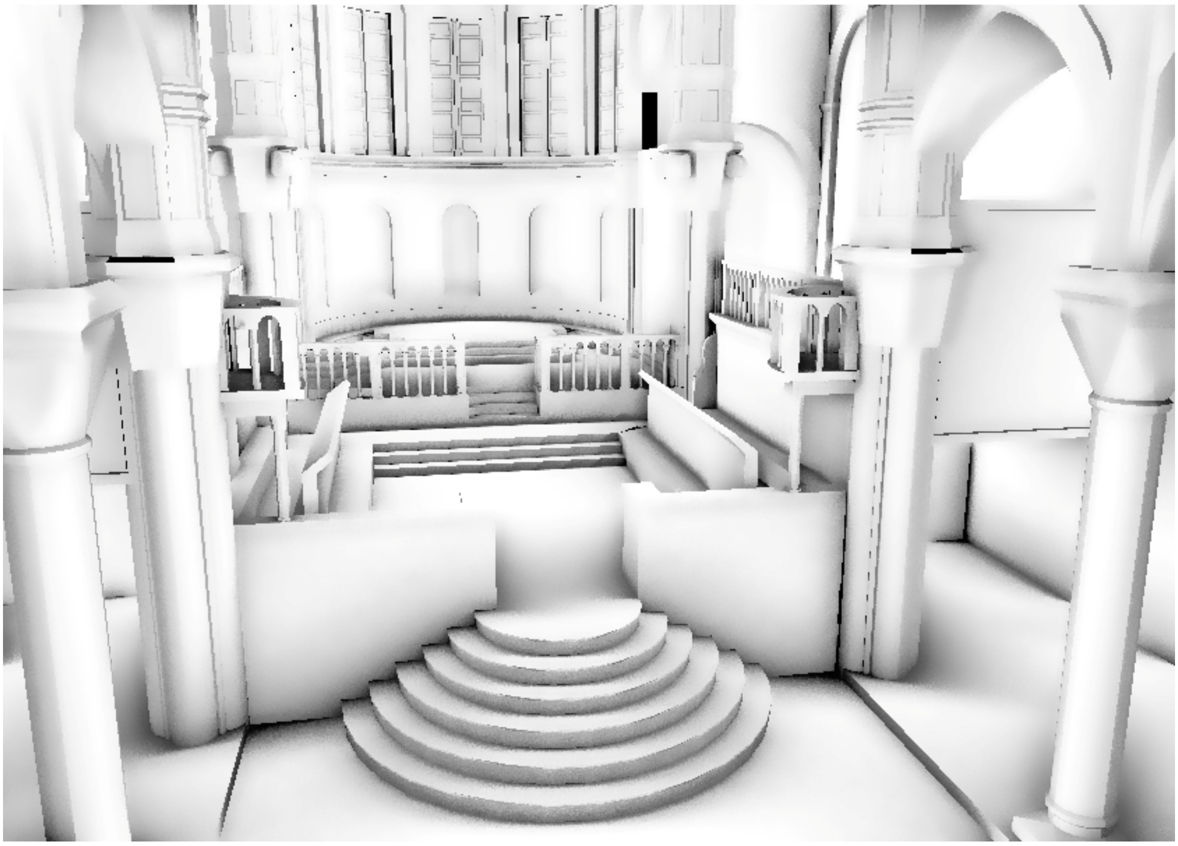


Slika 22. Ista scena kao na slici 21, ali uz slučajno uzorkovanje.

Ovako postavljen algoritam je posebice primjeren implementaciji na grafičkim karticama, budući da je reflektiranje vektora ostvarivo jednim funkcijskim pozivom u programu za sjenčanje. Kao što se vidi na slici 22, slučajnim uzorkovanjem uspjeli smo eliminirati artefakte nastale poduzorkovanjem, no istovremeno smo dodali neugodan šum na sliku. Jedan jednostavan način za ublažavanje šuma temelji se na principu minimalne energije, koji glasi: za zatvoreni sustav s konstantnom entropijom, ukupna energija će biti minimalna u stanju ravnoteže. Kako se ova općenita tvrdnja može primijeniti na naš konkretan problem šuma? Dakle, početnih n slučajnih vektora (gdje je n broj uzoraka po fragmentu) distribuirani su uniformno po sferi, odnosno generiranjem sfernih koordinata s uniformnom distribucijom. Ovaj skup vektora možemo interpretirati kao zatvoreni sustav. Pretpostavimo da je entropija (mjera kaotičnosti) sustava konstantna. Razina šuma u konačnoj slici će biti direktno proporcionalna energiji sustava. Dakle, ono što želimo postići, jest da energija tog sustava bude minimalna, a prema principu minimalne energije, to će se dogoditi u stanju ravnoteže. Kako definirati stanje ravnoteže ovog sustava? Možemo zamisliti sljedeću konstrukciju: traženi vektori zapravo predstavljaju pozicije točaka povezanih oprugama. Prema tome, potrebno je numerički riješiti ovaj sustav.

Postupak je jednostavan - za svaki par vektora, odredi se sila opruge. U ravnotežnom stanju, sila mora biti jednaka nuli, te ako je dobivena neka druga vrijednost, pozicija se pomakne za određeni iznos. Kroz određeni broj iteracija (stotinjak), novih pomaka više neće biti, što znači da je pronađeno ravnotežno stanje sustava. Budući da vektore generiramo unutar sfere, njihova je distribucija neovisna o rotaciji (koja će uslijediti prilikom samog uzorkovanja), tako da je ovaj postupak dovoljno provesti samo jednom, primjerice prilikom pokretanja programa, a zatim opet možemo jednostavno prenijeti dobivene vektore kao konstante u program za sjenčanje.

Premda princip minimalne energije pomaže pri ublažavanju šuma, taj trik sam po sebi nije dovoljan. Dodatan postupak kojeg možemo provesti zasniva se na ideji zamućivanja. Naime, jasno je da zamućivanjem slike gubimo detalje, što uključuje i šum. Budući da je ambijentalno zaklanjanje općenito nisko-frekvencijska pojava, određeni gubitak detalja je prihvatljiv. Međutim, zamućivanjem bismo "premazali" meke sjene preko površina koje nisu zaklonjene. Prema tome, pitanje je da li je moguće implementirati takav postupak zamućivanja, kojim bismo očuvali oštrima granice između objekata. Odgovor je potvrdan, te leži u tzv. bilateralnom filtriranju. Naime, standardno zamućivanje, primjerice Gaussovim filtrom, jednostavno zamjenjuje vrijednost svakog slikovnog elementa nekom linearnom kombinacijom susjednih elemenata, gdje su pojedini koeficijenti konstantni, tj. ne ovise o vrijednostima slikovnih elemenata. S druge strane, bilateralno filtriranje u obzir uzima i vrijednosti slikovnih elemenata, te kombinira one elemente koji su istovremeno blizu, tj. susjedni (geometrijska lokalnost), te slični bojom (fotometrijska lokalnost). Budući da je ambijentalna komponenta koju pruža SSAO algoritam uglavnom crno-bijela slika sa sivim tonovima, potrebno je samo definirati prag sličnosti, tj. granicu između zaklonjenosti i nezaklonjenosti. Ovaj prag se može ostaviti kao korisnička varijabla, budući da donekle ovisi o karakteristikama specifične scene, a donekle o broju korištenih uzoraka po fragmentu. Konačan rezultat svih opisanih tehnika, koje ćemo nazvati klasičnim SSAO algoritmom, prikazan je slikom 23:



Slika 23. Konačan rezultat klasičnog SSAO algoritma.

Premda je dobivena slika oku ugodna, performanse su nažalost nezadovoljavajuće. Na testnoj konfiguraciji (Intel Core 2 Duo 2.66 Ghz, ATI Radeon HD3870), iscrtavanje scene na slici 23 traje otprilike 58 milisekundi po slici, što daje 17 slika u sekundi, a što je sasvim neprihvatljivo. U postojećim radovima, spominje se nekoliko pristupa kako poboljšati performanse. Najjednostavniji način jest koristiti smanjenu rezoluciju G-spremnika za potrebe SSAO algoritma, primjerice četvrtinu ili čak osminu rezolucije zaslona. Budući da ionako provodimo zamućivanje, dodatan gubitak detalja neće biti toliko primjetan, pogotovo u konačnoj slici sa osvjetljenjem, teksturama itd. Premda stvaranje dodatnih G-spremnika umanjene rezolucije također zahtjeva određeno vrijeme, dobit zbog znatno bržeg SSAO prolaza opravdava ovaj trošak. Naime, ne samo da je smanjen broj fragmenata koje je potrebno obraditi, već je i poboljšana iskoristivost brze lokalne memorije za teksture (eng. *texture cache*). Veliki je problem što slučajnim uzorkovanjem u (širokoj) hemisferi oko fragmenta poništavamo korisnost brze lokalne memorije dostupne grafičkim procesorima. Uslijed zahtjeva za čitanjem podatka iz teksture, grafička kartica učitava čitavi blok, te pohranjuje taj blok u brzu lokalnu memoriju. Potencijalni dodatni zahtjevi za čitanjem teksture će po pretpostavci zahvaćati susjedne slikovne elemente, koji će biti sadržani u već učitanom bloku. Za većinu primjena, ova je pretpostavka ispunjena, što donosi značajno ubrzanje. Nažalost, kod klasičnog SSAO algoritma, zbog prirode uzorkovanja, dolazit će do velikog broja promašenih zahtjeva, tj. zahtjeva za podacima koji se nalaze izvan trenutnog teksturnog bloka. Smanjivanjem rezolucije G-spremnika, zapravo postižemo da pojedini blok obuhvaća veću površinu konačne slike (nakon skaliranja na punu veličinu). Budući da uzorkovanje ne vršimo u prostoru slike, već u 3D prostoru - dakle, neovisno o rezoluciji zaslona - dobivamo veću vjerojatnost da će pojedini uzorci nakon projekcije upadati u isti blok, od kojih sada svaki obuhvaća veći udio zaslona. Drugim riječima, u prosjeku ćemo ostvariti manji broj promašenih zahtjeva za čitanjem teksture. Jasno, iz danog objašnjenja je očito da bismo problemu mogli pristupiti i sa suprotne strane - osim povećavanja površine teksturnih blokova, možemo smanjiti radijus hemisfere po kojoj odabiremo uzorke, i na taj način dodatno smanjiti vjerojatnost promašenih zahtjeva. Međutim, širina uzorkovanja također ima znatan utjecaj na konačan izgled ambijentalnog zaklanjanja. Naime, uzak radijus uzorkovanja daje oštrije, tamnije sjene, koje daju jači naglasak pukotinama i oštrim rubovima objekata. S druge strane, širok radijus uzorkovanja daje mekše, svjetlije sjene, te omogućava blagu zaklonjenost i među objektima koji nisu direktno u kontaktu. Konkretno, u prikazanoj implementaciji klasičnog SSAO algoritma (slike 21-23), korišten je dvojni pristup uzorkovanja. Dakle, umjesto jednog skupa slučajnih uzoraka, generiraju se dva skupa, takvih da je prvi skup unutar hemisfere užeg radijusa, a drugi skup unutar hemisfere šireg radijusa. Dakako, sami iznosi tih radijusa ovise o skali scene, no njihov međusobni odnos je neovisan, te se pokazalo da omjer 1:5 daje dobre rezultate. Nažalost, premda se ovime kvaliteta može znatno podići, dodatnim uzorkovanjem u širokom radijusu također se znatno narušavaju performanse, zbog već spomenutog problema promašenih teksturnih zahtjeva. U ovom trenutku, također valja istaknuti dodatnu poteškoću. Naime, kako je već rečeno, uzorke biramo u 3D prostoru te projiciramo u prostor slike, što znači da površina zaslona koju pokriva hemisfera za neki dani fragment može znatno varirati od fragmenta do fragmenta, u ovisnosti o njihovoj dubini. Fragment koji pripada nekom vrlo udaljenom objektu će rezultirati uzorcima koji upadaju u vrlo malenu okolnu površinu, što daje gotovo sto postotnu iskoristivost brze lokalne memorije. S druge strane, hemisfera fragmenta nekog objekta koji se nalazi vrlo blizu kamere će se preslikati u vrlo veliku površinu zaslona, a što znači da će gotovo sigurno svaki uzorak značiti promašeni zahtjev za teksturom, doslovno uništavajući performanse. Iz tog razloga, potrebna je dodatna provjera dubine svakog fragmenta - ukoliko je dubina ispod određenog praga (što, opet, ovisi o skali scene), taj se fragment naprosto odbacuje. Međutim, čak i s postavljanjem praga na dubinu, performanse će donekle varirati pomicanjem kamere po sceni - približavanjem kamere objektima, performanse će opadati, a udaljavanjem će rasti. Konkretno, u razvijenoj implementaciji, performanse u prosjeku variraju od otprilike 10 do 25 slika u sekundi za danu scenu.

U konačnici, premda je klasični SSAO algoritam po prvi puta omogućio praktičnu aproksimaciju ambijentalnog zaklanjanja za dinamičke scene u stvarnom vremenu, postoji i niz poteškoća pri njegovoj izvedbi i korištenju:

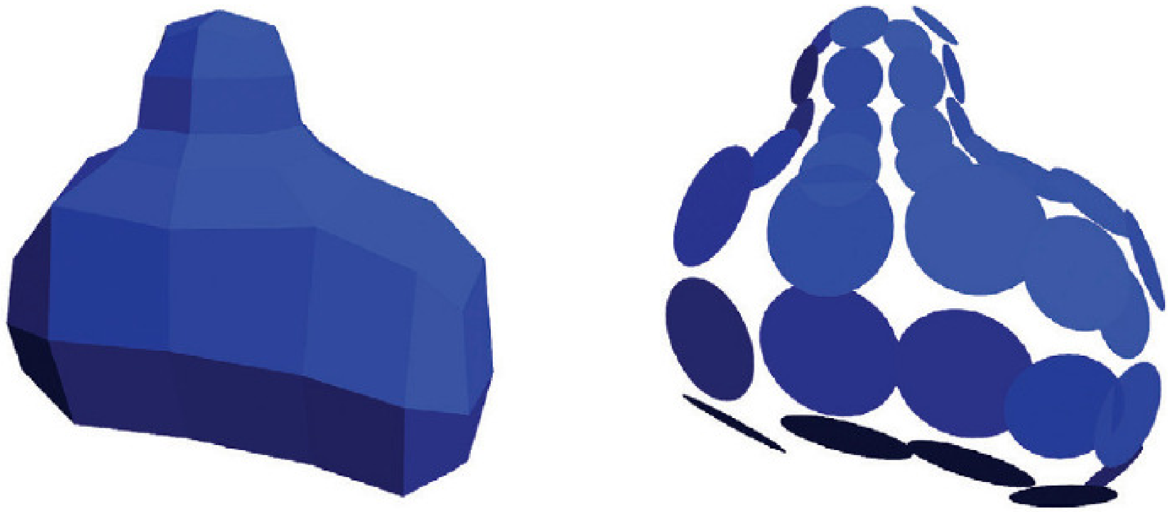
- Potreba za slučajnim uzorkovanjem - stvaranje šuma
- Potreba za dodatnim prolazom za filtriranje šuma
- Slaba iskoristivost brze lokalne memorije GPU-a
- Variranje performansi s promjenom pozicije kamere

Glavni uzročnik svih ovih poteškoća jest nepredvidljivost uzorkovanja. Naime, ne samo da je potrebno odabirati uzorke unutar hemisfere oko fragmenta na slučajan način, već i sama površina zaslona koju ta hemisfera zauzima varira od fragmenta do fragmenta. Prethodno su opisani poneki trikovi kako ublažiti probleme koji nastaju zbog ovakvog uzorkovanja, no pravo rješenje bi bila promjena same prirode uzorkovanja, na način da osiguramo bolju kontrolu i konzistentnost.

Prva ideja koje se možemo dosjetiti, jest da se zapitamo da li je uopće potrebno odabirati uzorke u 3D prostoru, budući da ih ionako projiciramo natrag u prostor slike. Drugim riječima, mogli bismo definirati regiju uzorkovanja u prostoru slike, koja bi zauzimala neku odgovarajuću površinu zaslona u okolini promatranog fragmenta. Ovom jednostavnom promjenom bismo naizgled doista dobili ono što smo željeli. Regiju uzorkovanja, u prostoru slike, bismo sada mogli direktno postaviti na željenu veličinu, i time osigurati dobro pokrivanje s teksturnim blokovima, eliminirajući problem promašenih teksturnih zahtjeva. Također, ta regija bi mogla biti konstantne površine, čime bismo riješili problem varirajućih performansi. Međutim, premda ova ideja djeluje obećavajuće, naivnom promjenom prostora uzorkovanja iz 3D hemisfere u 2D regiju nećemo dobiti zadovoljavajuće rezultate. Dobivena slika će izgledati "plošno", bez pravog osjećaja dubine, budući da će algoritam degenerirati u običan 2D filtar. Također, budući da je regija uzorkovanja konstantna, unutar iste regije će biti obuhvaćena veća ili manja površina objekta, u ovisnosti o blizini objekta kameri. Kao rezultat, približavanjem ili udaljavanjem kamere, sjene će postajati svjetlije ili tamnije. Jasno, mogli bismo riješiti ovaj problem tako da prilagodimo veličinu regije uzorkovanja dubini fragmenta, no time bismo samo došli do neke refaktorirane verzije klasičnog SSAO algoritma. Postoje pristupi poput [Bavoil, Sanz 08], gdje se sama hemisfera projicira u prostor slike, i time dobije varijabilna regija uzorkovanja u obliku diska. Premda i ovakav pristup pruža određena poboljšanja, kako je pokazano u navedenom radu, glavni problemi originalnog algoritma su i dalje prisutni, te performanse i dalje nisu sasvim zadovoljavajuće. Dakle, potrebna je neka nova spoznaja koja bi dovela do pravog pomaka. Ključ te spoznaje je već natuknut rečenicom: "algoritam degenerira u običan 2D filtar". Doista, glavni problem je što je funkcija zaklonjenosti, koja predstavlja jezgru samog algoritma, u osnovi jednodimenzionalna, budući da je definirana isključivo nad dubinom. No, budući da samo uzorkovanje vršimo u 3D prostoru, to znači da je informacija o 3D poziciji implicitno sadržana u odabiru uzoraka nad kojima evaluiramo funkciju zaklanjanja. Upravo ova činjenica omogućava da klasični SSAO algoritam, koji evaluira 1D funkciju u prostoru slike, ipak daje osjećaj dubine i prostora, za razliku od običnog 2D filtra. Uzorkovanjem u 2D regiji gubimo implicitnu informaciju o poziciji u prostoru kamere, što znači da ju moramo eksplicitno izračunati, te redefinirati funkciju zaklanjanja tako da djeluje nad vektorom pozicije, umjesto nad samo dubinom. Drugim riječima, algoritam sada glasi ovako:

1. Odaberi 8-32 uzorka u 2D regiji oko danog fragmenta.
2. Rekonstruiraj pozicije danog fragmenta i odabranih uzoraka u prostoru kamere.
3. Evaluiraj funkciju zaklanjanja nad dobivenim 3D uzorcima.
4. Pribroji pojedinačne doprinose uzoraka i odredi konačnu zaklonjenost.

Za početak, jednostavnosti radi, neka uzorkovanje bude predodređeno i jednako za svaki fragment, odnosno, unutar neke pravilne rešetke. Glavno pitanje na koje sada treba pronaći odgovor jest kako definirati funkciju zaklanjanja. Općenita svojstva i oblik te funkcije trebaju ostati jednaki već opisanima, no s tom promjenom da umjesto razlike dubina, promatramo primjerice euklidsku udaljenost uzoraka. Kao dobar izbor funkcije zaklanjanja, pokazala se metoda korištena u algoritmu dinamičkog zaklanjanja ambijenta (eng. *dynamic ambient occlusion, DAO*) [Bunnell 06]. Detaljan pregled ovog algoritma dan je u [Sajko 08], te ćemo ovdje dati samo kratak uvid u osnove. Dakle, ideja je vrlo slična kao kod klasičnog SSAO algoritma, no postupak se provodi u prostoru scene, te ne zahtjeva generiranje G-spremnika. Umjesto toga, potrebno je generirati dodatnu strukturu podataka na temelju geometrije scene, u kojoj će svaki vrh biti predstavljen diskom. Svaki takav disk treba biti orijentiran u prostoru tako da je okomit na normalu u danom vrhu. Primjer prikaza nekog objekta ovom strukturom dan je slikom 24:

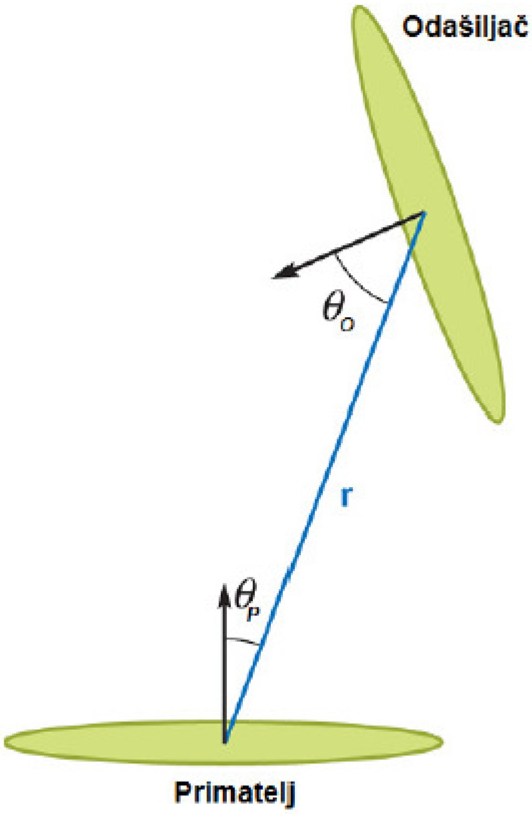


Slika 24. Objekt prikazan mrežom poligona (lijevo), te strukturom diskova (desno).

Jednom kad je odgovarajuća struktura diskova izgrađena, moguće je za svaki disk testirati vidljivost svih ostalih diskova, te evaluirati mjeru zaklonjenosti promatranog diska. Dakako, dobivene se vrijednosti zatim mogu jednostavno upotrijebiti pri izračunu ambijentalnog osvjetljenja po vrhu. Premda opisani algoritam ima značajne nedostatke (računanje zaklonjenosti po vrhu umjesto po fragmentu, potreba za dodatnom obradom scene), ono što je zanimljivo jest sama funkcija zaklanjanja. Naime, po definiciji, ambijentalna zaklonjenost neke točke jest udio površine hemisfere nad tom točkom, koji je zaklonjen nekim drugim objektom. Neka se objekt koji biva zaklonjen naziva primatelj, a objekt koji zaklanja odašiljač. Ukoliko pretpostavimo da su i primatelj i odašiljač u obliku diska (projekcija hemisfere na ravnu plohu), onda se pokazalo da dobru aproksimaciju mjere zaklonjenosti daje sljedeća formula:

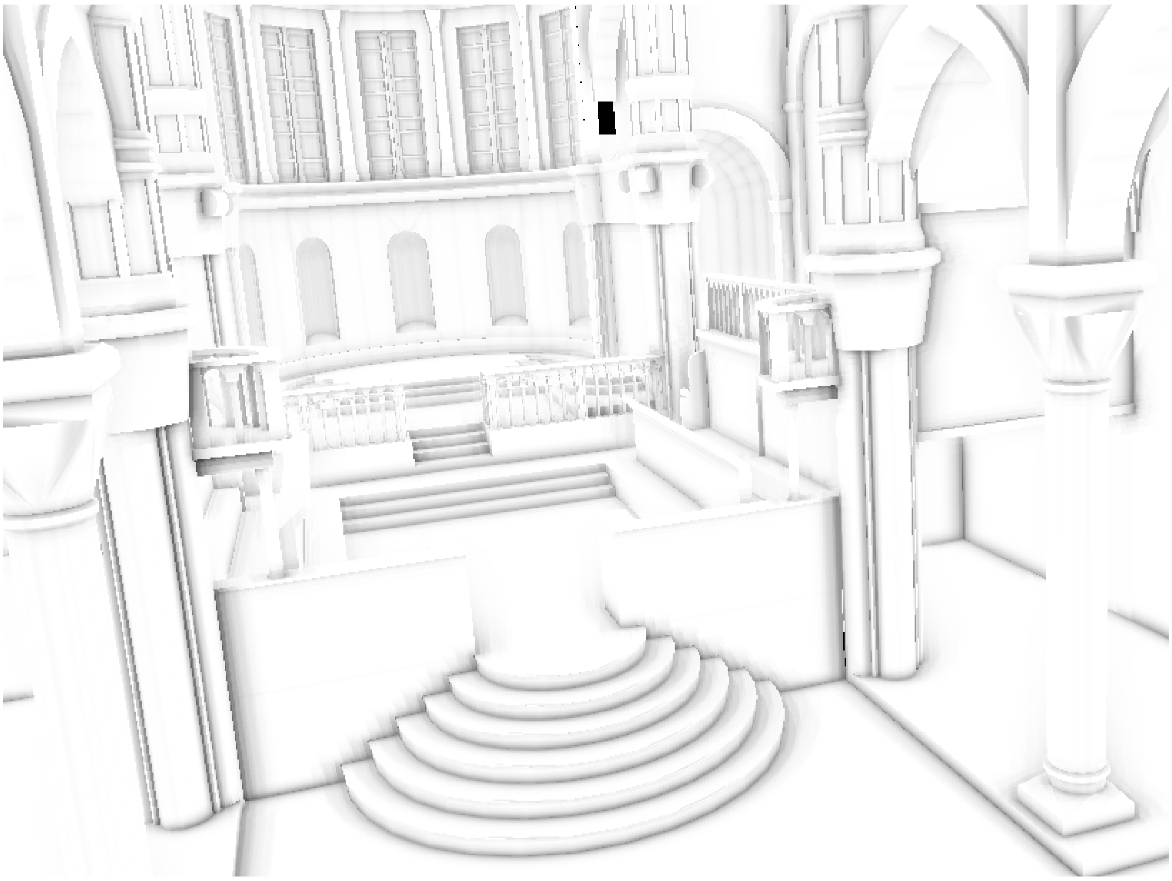
$$1 - \frac{r \cos \theta_O \max(1, 4 \cos \theta_P)}{\sqrt{\frac{A}{\pi} + r^2}} \quad (3)$$

gdje je r vektor koji spaja središta diskova, θ_O kut između normale odašiljača i vektora r, θ_P kut između normale primatelja i vektora r, te je A oplošje odašiljača. Funkcija *max()* vraća veći od dvaju argumenata, a služi zato da ograničimo računanje zaklonjenosti samo za objekte iznad gornje hemisfere. Navedene veličine grafički su prikazane slikom 25:



Slika 25. Ilustracija zaklanjanja diskova.

Vidimo da ovako definirana funkcija ovisi o međusobnom položaju uzoraka u 3D prostoru, a ne o njihovoj dubini - čime upravo zadovoljava svojstva potrebna za upotrebu u SSAO algoritmu uz 2D uzorkovanje. Direktnom ugradnjom navedene funkcije zaklanjanja u SSAO algoritam, zapravo implicitno aproksimiramo površinu objekata skupom diskova, baš kao što to eksplicitno čini DAO algoritam. Iz tog razloga, uslijed poduzorkovanja možemo očekivati pojavu artefakata u obliku pojaseva diskretnih razina intenziteta sjena, umjesto glatkih, kontinuiranih polusjena. No, budući da su takvi artefakti također oblik aliasinga, moći ćemo primijeniti već razmotrenu tehniku slučajnog uzorkovanja.

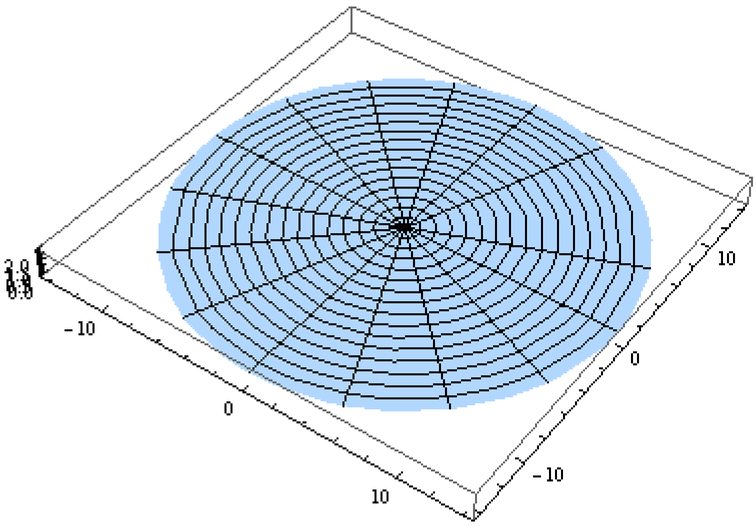


Slika 26. Rezultat SSAO algoritma uz 2D uzorkovanje.

Međutim, pokazalo se da su dobiveni rezultati čak iznad očekivanja. Na slici 26, prikazana je istovjetna scena kao u prethodnim primjerima, no iscrtana SSAO algoritmom uz pravilno, predodređeno uzorkovanje u 2D regiji oko fragmenata. Sam oblik ove regije nije presudan, no možda je najprirodnije pretpostaviti oblik diska. Redoslijed odabira uzoraka može biti u obliku koncentričnih kružnica, spirale ili slično. Pozicije traženih uzoraka, odnosno, oblik same regije uzorkovanja je najpraktičnije prikazati parametarskom jednadžbom, koju je moguće jednostavno evaluirati kroz niz iteracija. Jedan jednostavan i prikladan pristup prikazan je sljedećim grafom (slika generirana Wolfram Mathematica programskim paketom), gdje parametar *a* definira širinu regije uzorkovanja (slika 27).

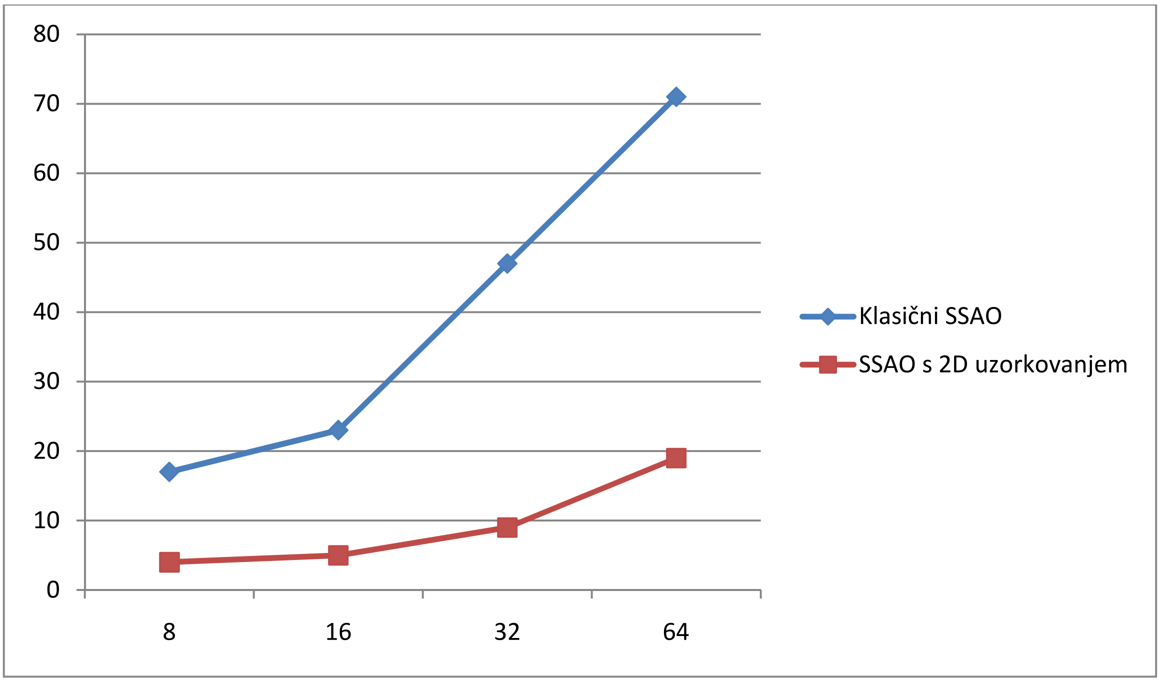
```
In[18]:= ParametricPlot3D[{a*Cos[u] - a*Sin[u], a*Sin[u] + a*Cos[u], 1},
           {u, 0, 2π}, {a, 0, 10}]
```

Out[18]=



Slika 27. Parametrizacija regije uzorkovanja.

Premda su na slici 26 prisutni očekivani artefakti, najuočljivije kod kružnih stuba u donjem dijelu slike, pojavljuju se u drastično manjoj mjeri nego kod klasičnog SSAO algoritma, uz jednak broj uzoraka po fragmentu. Razlog tomu djelom leži u samoj promjeni prostora uzorkovanja, a djelom u smanjenoj širini regije uzorkovanja. Kao što se vidi usporedbom slika 23 i 26, izgubljene su prostrane meke sjene dobivene širokim uzorkovanjem, a zadržani su samo naglasci oštrih prijelaza u reljefu scene. Premda je moguće proširiti regiju uzorkovanja, time bismo ponovno narušili iskoristivost brze lokalne memorije, pa je potrebno odvagnuti širinu pretraživanja i performanse. Također, budući da je pravilno uzorkovanje u uskoj 2D regiji vrlo pogodno arhitekturi modernih grafičkih kartica, možemo lako povećati broj uzoraka na primjerice 64 ili čak 128 po fragmentu, a i dalje zadržati sasvim prihvatljive performanse. Kod ovako visokog broja uzoraka, problem poduzorkovanja u potpunosti nestaje, premda i sa standardnih 32 uzorka, artefakti poduzorkovanja su slabo primjetni. Iz ovoga slijedi da slučajno uzorkovanje nije niti potrebno, što znači da neugodan, primjetan šum zamjenjujemo slabo primjetnim diskretizacijskim artefaktima. Nadalje, više nije potreban niti dodatan prolaz zamućivanja, čime se postupak implementacije te računalna zahtjevnost dodatno pojednostavljuju. Sljedeći graf pokazuje ovisnost performansi o broju uzoraka, za klasični SSAO, te za SSAO s 2D uzorkovanjem. Mjera performansi jest vrijeme utrošeno po slici (u milisekundama), uz konstantnu rezoluciju (1280x720) i punu veličinu G-spremnika, mjereno na konfiguraciji: Intel Core 2 Duo 2.66 Ghz, ATI Radeon HD3870.

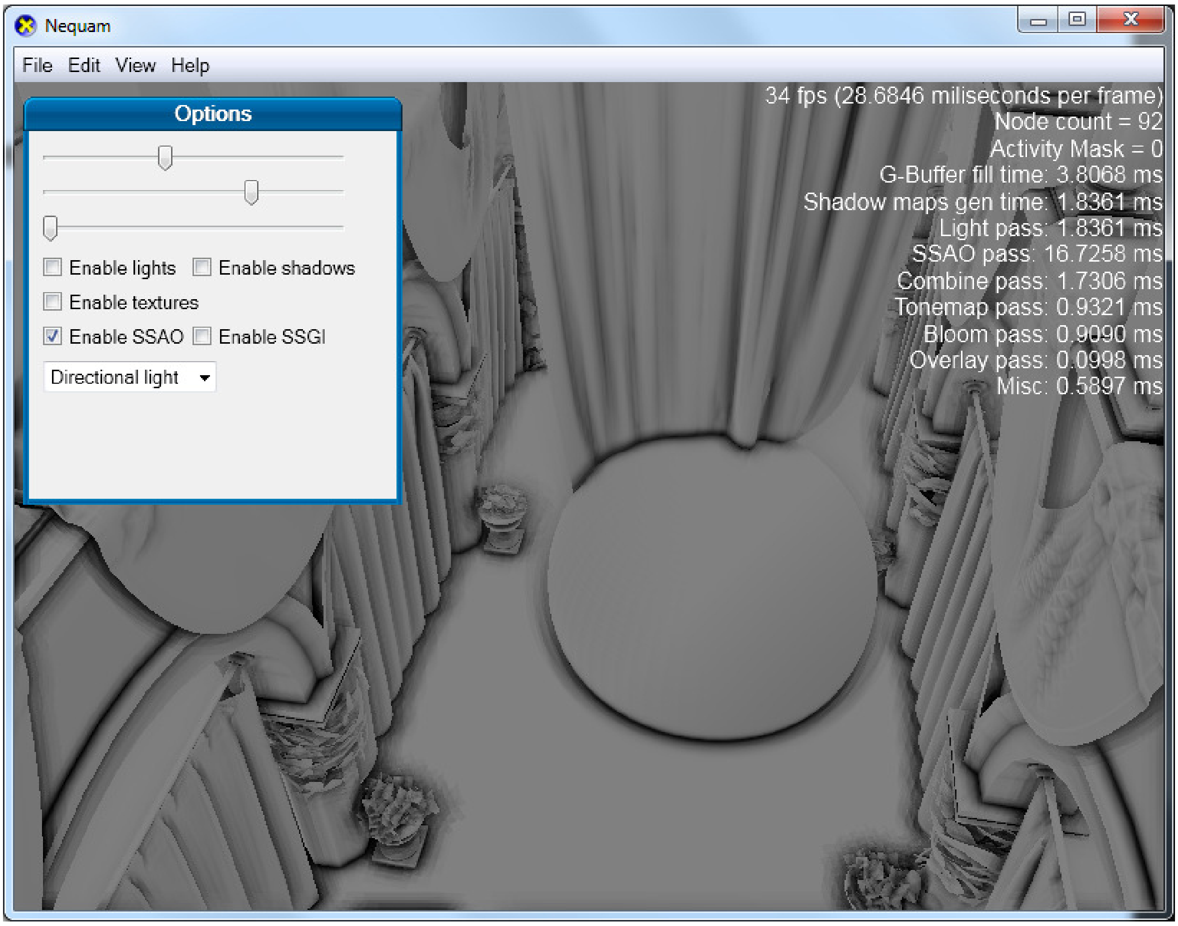


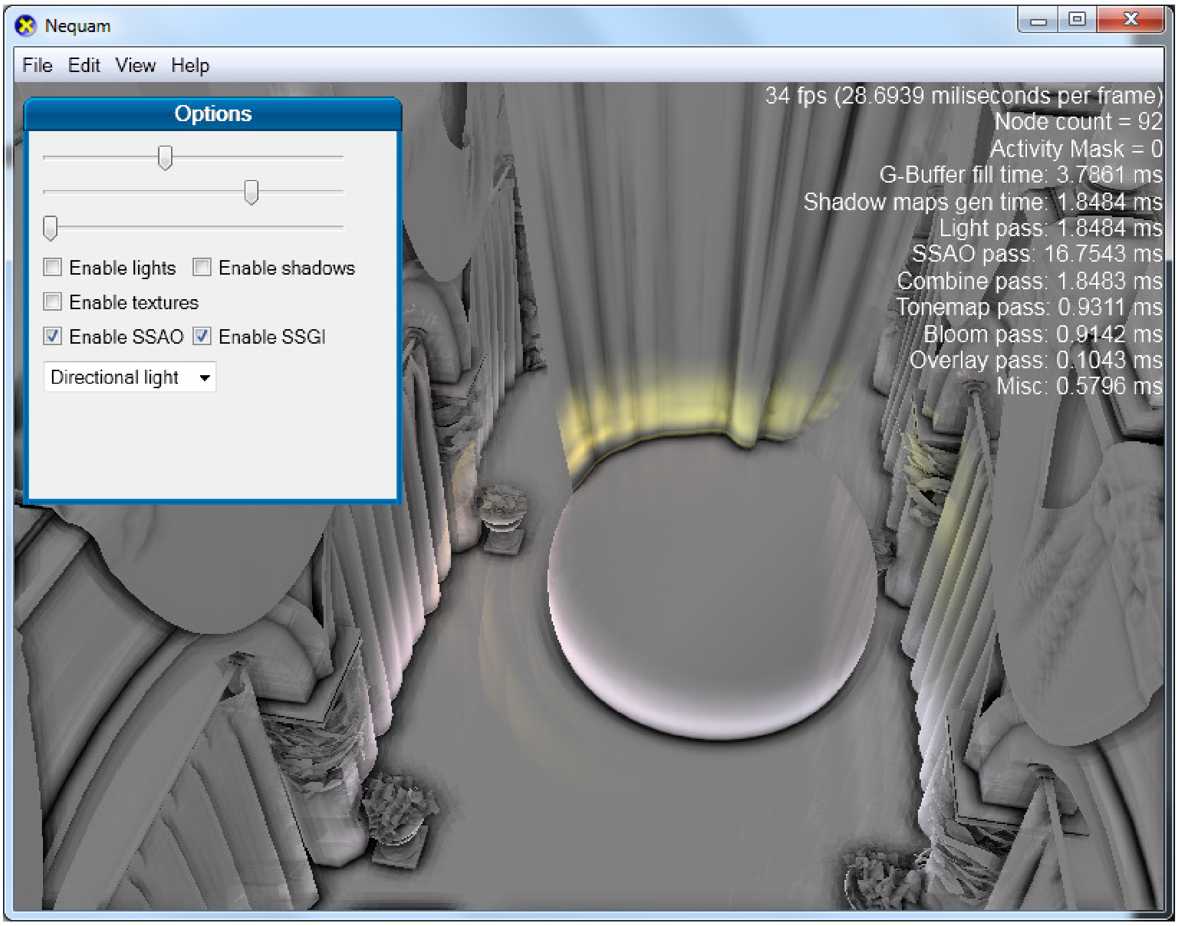
Slika 28. Usporedba performansi dviju varijanti SSAO algoritma.

Ovime smo pokazali kako, uz relativno jednostavne izmjene, SSAO algoritam može poprimiti znatno bolja svojstva. Redefinicijom funkcije zaklanjanja, omogućeno je jednostavno uzorkovanje u prostoru slike, umjesto dosadašnjeg uzorkovanja u 3D prostoru. Time je algoritam postao jednostavniji za implementaciju, te mnogo prikladniji arhitekturi modernih grafičkih kartica. Kroz bolju iskoristivost brze lokalne memorije, značajno su poboljšane performanse, koje su također mnogo konzistentnije. Daljnji razvoj algoritma je moguć u obliku pronalaska jednostavnije funkcije zaklanjanja, čime bi se smanjilo vrijeme obrade pojedinačnih uzoraka, ili boljeg načina uzorkovanja (primjerice, uzimanje varijabilnog broja uzoraka, u ovisnosti o dubini, ili neki poluslučajni pristup), čime bi se u prosjeku smanjio broj potrebnih uzoraka za postizanje iste kvalitete.

9. Difuzna interrefleksija u prostoru slike

Računanjem ambijentalnog zaklanjanja, proširili smo standardni lokalni model osvjetljenja, dodavši meke, ambijentalne sjene na scenu. Međutim, ambijentalne sjene su tek jedan od efekata globalnog osvjetljenja, koji nastaje uslijed difuzne interrefleksije. Još jedan efekt vezan uz višestruko odbijanje svjetlosti među objektima na sceni jest pretapanje boja. Naime, interrefleksijom difuznog osvjetljenja između dva objekta, doći će do promjene percipirane boje tih objekata, budući da će jedan utjecati na drugoga svojom refleksijom, tj. obojanom svjetlošću (vidjeti poglavlje 2). Opisana situacija jest rezultat jednostrukog odbijanja svjetla, no jasno, moguće je i višestruko odbijanje. Ukupan rezultat svog indirektnog osvjetljenja ćemo nazvati ambijentalnom komponentom osvjetljenja. Na slici 29, prikazano je konstantno ambijentalno osvjetljenje izračunato standardnim lokalnim modelom (Blinn-Phong), no obogaćeno ambijentalnim sjenama (prva slika), te jednostrukim indirektnim osvjetljenjem (druga slika).





Slika 29. Usporedba ambijentalne komponente osvjetljenja. Gore - SSAO, dolje - SSGI.

Algoritam korišten za ambijentalne sjene jest prethodno opisani postupak zaklanjanja ambijenta u prostoru slike, s 2D uzorkovanjem (SSAO). Uz relativno jednostavno proširenje ovog postupka, moguće je izračunati i jednostruko odbijeno indirektno osvjetljenje, rezultat čega je prikazan na gornjoj slici. Ovaj prošireni algoritam se naziva globalno osvjetljenje u prostoru slike (eng. *screen space global illumination, SSGI*). Naziv je pomalo zavaravajući, budući da umjesto punog globalnog osvjetljenja (s efektima poput ispodpovršinskog raspršivanja, kaustike, itd.) zapravo dobivamo samo jednostruko indirektno difuzno osvjetljenje; točniji naziv algoritma bi mogao biti difuzna interrefleksija u prostoru slike. Dakle, ideja je sljedeća - prema jednadžbi iscrtavanja izvedenoj u poglavlju 7, indirektno osvjetljenje u nekoj točki prostora jednako je integralu upadnog zračenja po hemisferi nad tom točkom (uz izostavljanje direktne komponente):

$$L_I(x) = \int_{\Omega} L_i(x, \omega) d\omega \quad (1)$$

Ovo je veoma slično definiciji ambijentalnog zaklanjanja, koja uključuje integral funkcije vidljivosti po hemisferi. Iz tog razloga, možemo opet aproksimirati rješenje integrala sumacijom uzoraka podintegralne funkcije:

$$L_I \approx \sum_{\Omega} L_i(x, \Delta\omega) \quad (2)$$

Budući da je riječ o istovjetnom problemu, možemo upotrijebiti isti postupak rješavanja kao kod ambijentalnog zaklanjanja. Jasno, razlika je sada u tome što je umjesto funkcije vidljivosti potrebna funkcija upadne indirektne svjetlosti. Pretpostavimo da je moguće samo jednostruko odbijanje svjetlosti. U tom slučaju, dovoljno je uzorkovati boju difuznog osvjetljenja fragmenata unutar hemisfere (odnosno, u nekoj regiji). Svaki uzorak pridonosi osvijetljenosti danog fragmenta, na temelju svoje boje te udaljenosti od danog fragmenta. Dakle, bit će potrebno definirati i neku funkciju prijenosa između uzoraka, tj. fragmenata. Ova funkcija može biti proizvoljna, ali kao i funkcija zaklanjanja kod SSAO algoritma, mora posjedovati određena svojstva:

- Bliži uzorci imaju veći utjecaj nego udaljeniji uzorci.
- Funkcija opada u nulu nakon nekog praga.

Također, "udaljenost" uzorka ćemo definirati kao euklidsku udaljenost od danog fragmenta, u prostoru kamere. Preostalo pitanje jest - kako zapravo odrediti boju difuznog osvjetljenja uzoraka? Ukoliko se prisjetimo predložene konfiguracije G-spremnika (slika 9), vidimo da spremnik MRT0 sadrži upravo difuznu komponentu osvjetljenja. Dakle, kao što kod SSAO algoritma uzorkujemo spremnik dubine i evaluiramo funkciju zaklanjanja, kod SSGI algoritma ćemo uzorkovati spremnik difuznog osvjetljenja te evaluirati funkciju prijenosa. Kompletan algoritam glasi ovako:

1. Odaberi 8-32 uzorka u 2D regiji oko danog fragmenta.
2. Odredi vrijednosti difuznog osvjetljenja odabranih uzoraka.
3. Rekonstruiraj pozicije danog fragmenta i odabranih uzoraka u prostoru kamere.

4. Evaluiraj funkciju prijenosa nad dobivenim 3D uzorcima i pripadajućim vrijednostima difuznog osvjetljenja.

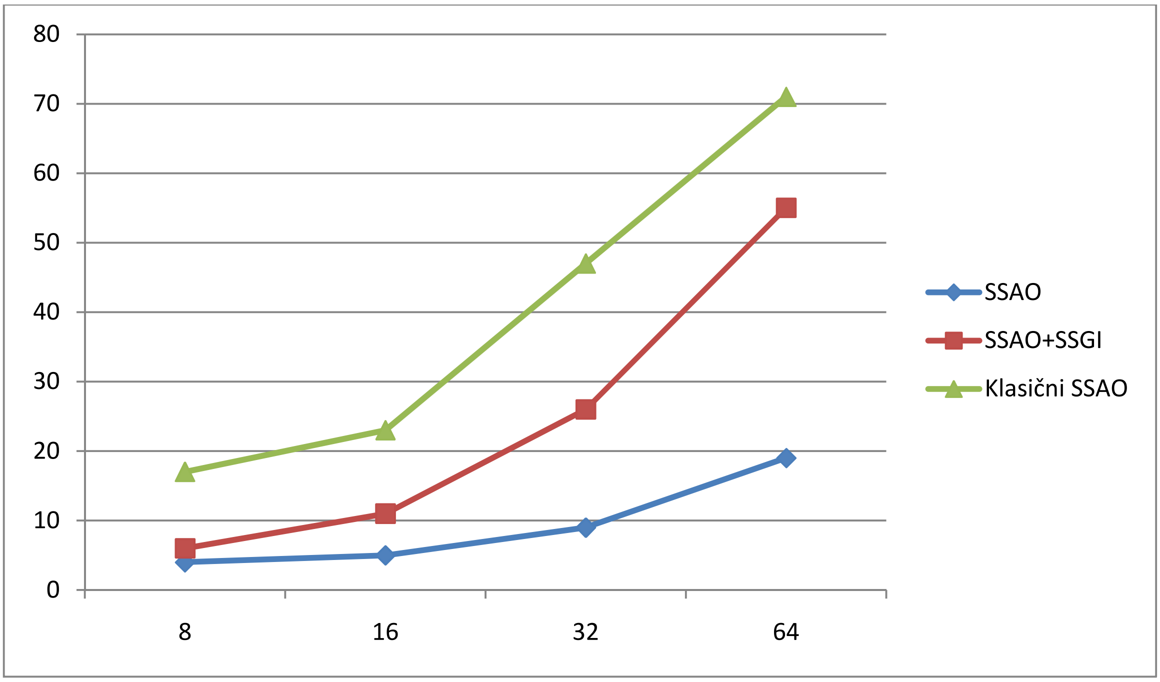
5. Pribroji pojedinačne doprinose uzoraka i odredi konačnu osvijetljenost.

Doista, algoritam veoma podsjeća na prethodno opisani postupak ambijentalnog zaklanjanja, te je ova dva algoritma moguće i integrirati, te izvršiti unutar istog prolaza. No, baš kao i kod funkcije zaklanjanja, i funkciju prijenosa je potrebno pažljivo formulirati, budući da predstavlja jezgru samog algoritma. Kao dobro i prikladno rješenje, možemo preuzeti funkciju prijenosa korištenu u već spomenutom DAO algoritmu. Naime, već u [Bunnell 06] razvija se ideja proširenja ambijentalnog zaklanjanja u indirektno osvjetljenje, zbog sličnosti problema. Premda predloženo rješenje djeluje u prostoru scene i time zadržava nedostatke osnovnog DAO algoritma (opisano u prethodnom poglavlju), sama funkcija prijenosa se pokazala korisnom. Dakle, definirat ćemo funkciju prijenosa sljedećim izrazom:

$$\frac{A\cos\theta_O\cos\theta_P}{\pi r^2 + A} \quad (3)$$

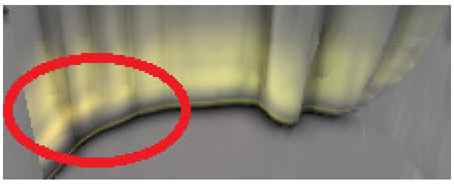
gdje je *r* vektor koji spaja središta diskova, θ_O kut između normale odašiljača i vektora *r*, θ_P kut između normale primatelja i vektora r, te je *A* oplošje odašiljača (vidjeti sliku 25).

Sljedeće bitno pitanje jest širina regije uzorkovanja. Naime, kod SSAO algoritma, dovoljna je uska regija uzorkovanja, takva da se postigne dobro poklapanje s teksturnim blokovima, i time poveća efikasnost. No, indirektno osvjetljenje emitirano nekim objektom može imati vrlo širok prostor utjecaja, što ovisi o intenzitetu osvijetljenosti tog objekta. Iz tog razloga, potrebno je koristiti širu regiju uzorkovanja, i time obuhvatiti veći broj (potencijalno) utjecajnih uzoraka. U integriranoj implementaciji SSAO i SSGI algoritama, ova potreba rezultira dvojnim načinom uzorkovanja (slično dvoprolaznom klasičnom SSAO pristupu), gdje se usko uzorkovanje koristi za ambijentalno zaklanjanje, a široko uzorkovanje za indirektno osvjetljenje. Kao omjer širina ovih regija, odabran je 1:5. Nažalost, proširenjem regije uzorkovanja, ponovno narušavamo iskoristivost brze lokalne memorije za teksture, što pogoršava performanse. Međutim, postupak je i dalje efikasniji od klasičnog SSAO algoritma s uzorkovanjem u 3D prostoru, kako pokazuje sljedeći graf:



Slika 30. Usporedba performansi SSAO, SSAO+SSGI, te klasične SSAO implementacije.

Kao i u prethodnom grafu, performanse su izražene u milisekundama po slici, u ovisnosti o broju uzoraka po fragmentu (mjereno na istoj, prethodno navedenoj testnoj konfiguraciji uz jednake parametre i istu scenu). Ovi rezultati još jednom potvrđuju važnost smanjivanja broja promašenih zahtjeva. Kod 32 uzorka, integrirana SSAO+SSGI implementacija je na granici upotrebljivosti, što znači da bi u stvarnim primjenama bilo poželjno koristiti umanjene G-spremnike, za poboljšanje efikasnosti. Također, proširivanjem regije uzorkovanja, diskretizacijski artefakti postaju donekle uočljiviji, iz jednostavnog razloga što jednakim brojem uzoraka pokrivamo veći prostor (vidjeti prethodno poglavlje, slika 27). Na slici 31, prikazan je detalj sa slike 29, gdje je označen artefakt u obliku tamnog pojasa na dijelu zastave pod utjecajem kugle, koji bi trebao biti jednoliko žute boje, a što je rezultat preskakanja pojaseva fragmenata uslijed poduzorkovanja.

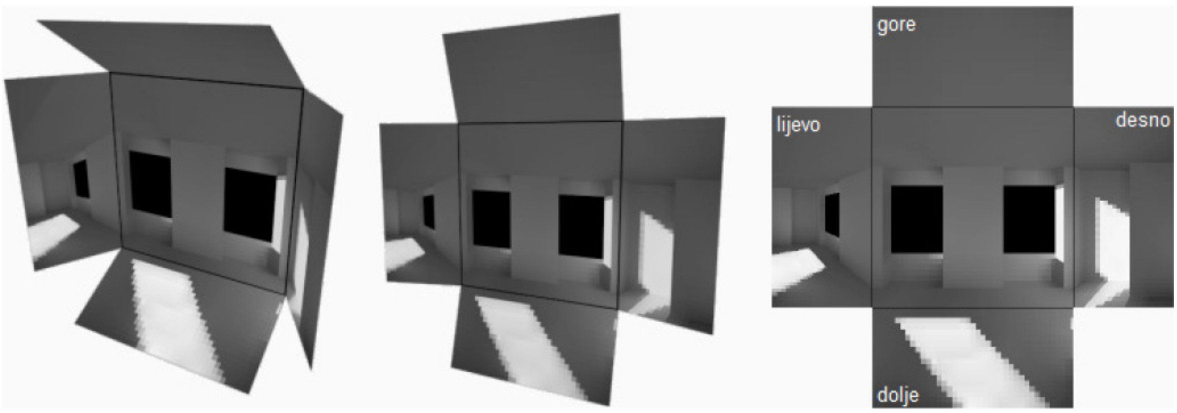


Slika 31. Artefakt poduzorkovanja kod SSGI algoritma.

Nažalost, ovaj se problem ne može u potpunosti riješiti bez povećanja broja uzoraka, no moguće ga je ublažiti boljim načinom uzorkovanja (primjerice, spiralno, ili polunasumično). No, pri konačnom iscrtavanju scene, takvi artefakti neće biti vrlo primjetni (ako uopće), budući da je i sam učinak indirektnog osvjetljenja suptilan.

Na kraju, osvrnimo se na ograničenja SSGI algoritma. Jedna od početnih pretpostavki jest ograničenje na tek jednostruko odbijanje svjetla. Budući da je problem višestruke interrefleksije po prirodi rekurzivan, moguće je izračunati doprinose dodatnih odbijanja jednostavnim dodavanjem prolaza istovjetnog algoritma. Međutim, doprinos već i drugog prolaza bi u većini stvarnih scena bio vrlo suptilan, a trećeg prolaza gotovo neprimjetan. S druge strane, vrijeme obrade svakog novog prolaza bi bilo otprilike jednako, što znači da za većinu praktičnih primjena dodatni prolazi ne opravdavaju svoju cijenu.

Vrlo značajno ograničenje algoritma proizlazi iz same prirode djelovanja u prostoru slike. Naime, u bilo kojem trenutku, algoritmu su dostupni isključivo podaci o objektima koji su trenutno vidljivi. Međutim, moguća je situacija da poneki objekt i dalje ima utjecaj na vidljivi dio scene, makar sam trenutno nije vidljiv. U tom slučaju, utjecaj tog objekta će biti izgubljen, što nažalost nije moguće izbjeći na elegantan način. Jedno rješenje bi bilo iscrtavanje scene iz nekoliko pogleda, primjerice unutar polukocke. U ovom kontekstu, pod pojmom "polukocka" podrazumijeva se takva kocka, koja je presječena nekom ravninom paralelnom s nekom stranicom te kocke, i to tako da se mreža dobivene polukocke sastoji od jednog kvadrata, te četiri pravokutnika s odnosom stranica 2:1.



Slika 32. Polukocka i njena pripadna mreža.

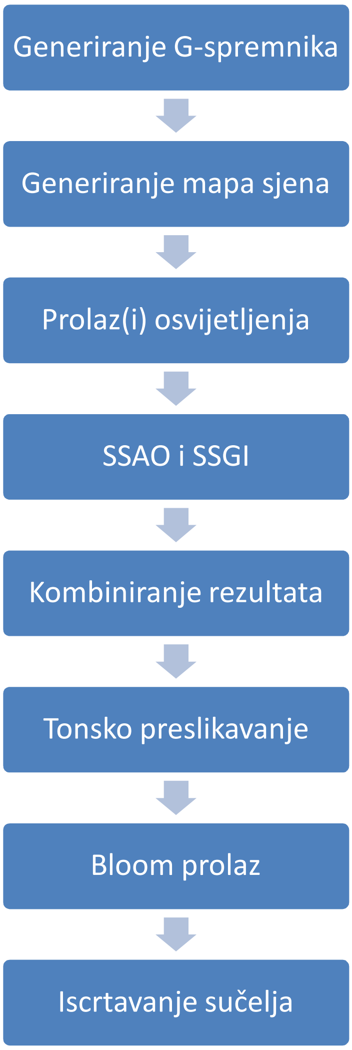
Takva polukocka se centrira s obzirom na očište, te se cijela scena iscrta pet puta, za svaku stranicu polukocke, i to tako da normala stranice leži na pravcu gledišta. Jasno, ovime bismo pokrili veći dio prostora nego što je potrebno iscrtati, što bi rezultiralo mnogo većom potrošnjom memorije (zbog povećanja G-spremnika), te mnogo duljim vremenom iscrtavanja (zbog većeg broja fragmenata koje je potrebno obraditi). Iz tog razloga, ograničenje na računanje utjecaja isključivo trenutno vidljivih objekata je u praktičnim implementacijama nažalost neizbježno.

Dodatan problem koji se javlja iz istog razloga jest ograničenje na računanje utjecaja samo prednjih poligona. Naime, definirat ćemo prednje poligone kao one koji su okrenuti prema kameri, a stražnje kao one koji su okrenuti u suprotnom smjeru. Dakle, za bilo koji objekt, dovoljno je iscrtati samo prednje poligone, budući da stražnji poligoni ionako nisu vidljivi. Iz tog razloga, u G-spremnicima posjedujemo informacije dobivene isključivo na temelju prednjih poligona. Međutim, jasno je da dani objekt utječe na okolne objekte i svojom stražnjom stranom. Budući da su informacije o stražnjim poligonima izgubljene, nije moguće izračunati njihov doprinos indirektnom osvjetljenju. Jednostavno rješenje ovog problema bi bilo iscrtavanje scene dva puta, i generiranje dva skupa G-spremnika. U prvom prolazu, bili bi obrađeni prednji poligoni, a u drugom stražnji. Rezultat oba prolaza bismo mogli akumulirati u spremniku difuznog osvjetljenja, pomoću aditivnog miješanja. Premda bismo ovime poboljšali točnost i kvalitetu rezultata, udvostručili bismo utrošak memorije te vrijeme izvođenja algoritma. Drugim riječima, ovo postaje pitanje odnosa kvalitete i performansi, što može biti ostavljeno i kao korisnička opcija.

U konačnici, SSGI algoritam omogućava na relativno jednostavan način računati indirektno osvjetljenje za dinamičke scene, u stvarnom vremenu. Premda postoje određena ograničenja algoritma, zbog same prirode djelovanja u prostoru slike, algoritam je vrlo efikasan, te nudi dobru aproksimaciju globalnog ambijentalnog osvjetljenja uz vrlo dobre performanse, što ga čini odličnim kandidatom za upotrebu u kompleksnim interaktivnim aplikacijama poput računalnih igara. Daljnja poboljšanja algoritma su direktno vezana uz poboljšanja SSAO algoritma, budući da se oba zasnivaju na istom principu.

10.Implementacija

Kao trenutno najbolje rješenje za aproksimiranje globalnog osvjetljenja, odabrana je kombinacija zaklanjanja ambijenta te difuzne interrefleksije u prostoru slike, dakle, SSAO te SSGI algoritam. Premda ovo rješenje ima određena ograničenja (kako je opisano u prethodnim poglavljima), rezultati su zanimljivi, uz vrlo dobre performanse te jednostavnost implementacije. Međutim, budući da je riječ o efektima u prostoru slike, koji se baziraju na dostupnosti G-spremnika, potrebno je implementirati kompletan sustav iscrtavanja s odgodom. Također, za jednostavno testiranje učinaka raznih algoritama i efekata, potreban je i sustav prozora, kojim bi bilo moguće izgraditi jednostavno korisničko sučelje. Nadalje, potrebna je i funkcionalnost učitavanja proizvoljnih testnih scena, te njihov učinkovit prikaz u memoriji računala. Iz tog razloga, odlučeno je temeljiti programsku izvedbu na sustavu grafa scene. Jedna mogućnost jest koristiti gotovo rješenje poput OpenSceneGraph paketa, no takvi paketi nude mnoštvo mogućnosti, od čega bi velika većina bila nepotrebna u ovom konkretnom slučaju. Iz tog razloga, dizajniran je vlastiti, maleni i efikasni sustav grafa scene. Također, umjesto direktne izvedbe korisničkog sučelja putem Windows API-ja (odnosno, neke intermedijarne biblioteke poput wxWidgets-a) i kombiniranja tog sučelja sa slikom iscrtanom kroz Direct3D ili OpenGL grafičko sučelje, odabrana je opcija direktne implementacije sustava prozora kao dijela grafa scene. Prema tome, umjesto korištenja dva odvojena sustava iscrtavanja (jedan za korisničko sučelje, što omogućava sam operacijski sustav, te drugi za scenu), koristi se samo jedan, vlastito implementirani sustav iscrtavanja temeljen na Direct3D grafičkom sučelju. Prozori koji sačinjavaju korisničko sučelje su tek objekti na sceni, ni po čemu drugačiji od ostalih objekata unutar grafa scene (osim što su dvodimenzionalni), te ih sustav za iscrtavanje niti ne raspoznaje. Ovime su izbjegnuti mnogi problemi i poteškoće koji mogu nastati prilikom sinkronizacije dvaju odvojenih sustava iscrtavanja, te je sama programska izvedba uvelike pojednostavljena. Modularnost ovakvog dizajna je omogućila razvoj pojedinih podsustava u obliku neovisnih komponenti, koje su sve povezane te komuniciraju putem centralne strukture grafa scene. Konkretno, postoje tri takve komponente - iscrtavatelj (eng. *renderer*), upravljač prozorima (eng. *window manager*), te upravljač resursima (eng. *resource manager*). Jasno, sve su komponente reaktivne, te se svaka odvija u zasebnoj dretvi. Iscrtavatelj pristupa grafu scene kao skupu ulaznih podataka, te iscrtava sve objekte obuhvaćene grafom. Upravljač prozorima manipulira grafom prozora, što je podgraf cjelokupnog grafa scene, i to na temelju korisničkih akcija (pomaci miša, pritisci tipaka, itd). Rezultati ovih promjena se zatim očituju u iscrtanoj slici, bez da iscrtavatelj bude svjestan na koji način su se ove promjene dogodile, ili da se uopće jesu dogodile. Riječ je o sličnom ustroju kao kod Model-Pogled-Upravljač obrasca, no s tom razlikom da nema potrebe za korištenjem obrasca Promatrača za komunikaciju između komponenti, budući da su sve reaktivne (izvršavaju se neprekidno i opetovano, za vrijeme trajanja programa). Dakako, umjesto toga, potrebno je koristiti odgovarajuću tehniku sinkronizacije između dretvi, kao što su *mutex* objekti. Također, na jednostavan je način moguće dodati i komponentu za fiziku. Proračunom sila i drugih fizikalnih veličina, ova bi komponenta manipulirala objektima u grafu scene, što bi bilo transparentno za ostatak sustava. Konačno, upravljač resursima koristi graf scene i kao ulaz i kao izlaz. Dodavanjem novog objekta u graf scene (ili putem korisničkog sučelja, dakle, putem upravljača prozora, ili programski iz bilo kojeg dijela programa), ukoliko taj objekt sadrži referencu na geometrijski model ili teksturu koja nije još učitana, iscrtavatelj će preskočiti iscrtavanje takvog objekta (budući da ne postoje potrebni resursi), te doznačiti upravljaču resursa da je potrebno učitavanje. Dretva upravljača resursa je u stanju mirovanja, sve dok ne pristigne zahtjev za učitavanjem. Implementirane su dvije vrste zahtjeva - blokirajući, te neblokirajući. Kod prvog tipa zahtjeva, pozivajuća dretva se zaustavlja dok učitavanje nije dovršeno, dok kod drugog tipa, pozivajuća dretva može nastaviti s radom. Dakako, iscrtavatelj koristi upravo neblokirajuću vrstu zahtjeva za učitavanjem, tako da se iscrtavanje može nastaviti i dok se u pozadini vrši učitavanje dodatnih resursa. Svi resursi se održavaju u pričuvnoj memoriji (u obliku hash-tablice), sve dok su u upotrebi, odnosno, dok na njih postoji barem jedna referenca unutar grafa scene. Jednom kad broj referenci nekog resursa padne na nulu, upravljač resursima oslobađa taj resurs. Na taj način, omogućeno je konstantno dodavanje novog sadržaja u graf scene uz oslobađanje nepotrebnih resursa, bez prekida za učitavanje. Ovo svojstvo je posebice korisno, primjerice, za prostrana, otvorena virtualna okruženja, koja zbog svoje veličine nikako ne bi mogla biti učitana u memoriju odjednom i u cijelosti. Za samo učitavanje modela i tekstura u raznim formatima, korištena je malena ali moćna pomoćna biblioteka Assimp (skraćeno od *Asset Import Library*). Ovime su ukratko pokriveni temelji grafičkog pogona razvijenog za potrebe testiranja raznih algoritama opisanih u okviru ovog diplomskog rada. Posebice je zanimljiva komponenta iscrtavatelja, budući da su unutar ove komponente implementirani svi opisani postupci, poput osvjetljenja s visokim dinamičkim rasponom, preslikavanja sjena, te zaklanjanja ambijenta i difuzne interrefleksije. Za čim jednostavnije dodavanje raznih efekata pri iscrtavanju, razvijen je objektni model prikladan domeni. Koncepti kao što su G-spremnici su enkapsulirani odgovarajućim razredima. Također, budući da je sve navedene algoritme potrebno implementirati na grafičkoj kartici, i koncept programa za sjenčanje je enkapsuliran zasebnim razredom. Na taj način, omogućeno je vrlo jednostavno dodavanje novih efekata i povezivanje njihovih ulaza i izlaza na odgovarajuće spremnike. Kompletan postupak iscrtavanja, te međusobni ustroj pojedinih prolaza i faza, grafički je prikazan slikom 33.



Slika 33. Prikaz redoslijeda prolaza pri iscrtavanju.

11.Zaključak

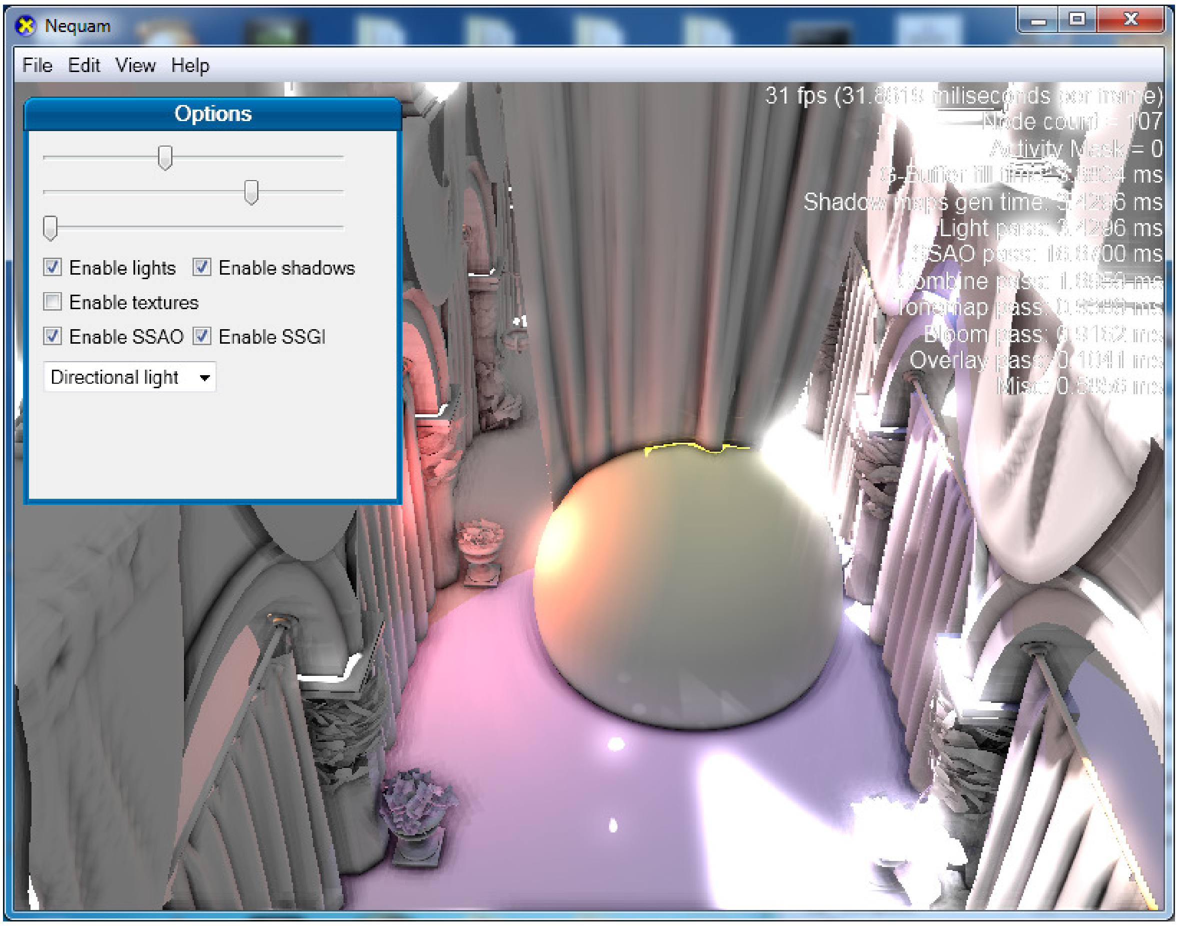
Kroz prethodna poglavlja, opisani su razni postupci ostvarivanja direktnog i indirektnog osvjetljenja, te sjena. Kroz unaprijeđenje tehničkih karakteristika grafičkih kartica, omogućeno je korištenje aritmetike s pomičnim zarezom, u visokoj preciznosti. Jedan od prvih vidljivih rezultata jest iscrtavanje s visokim dinamičkim rasponom, što već samo po sebi rezultira mnogo uvjerljivijom i prirodnijom slikom. Međutim, dostupnost visoke preciznosti, uz rastuću količinu memorije i brzine grafičkih procesora, omogućuje i razvoj novih algoritama za različite specijalne efekte, poput aproksimacije učinaka globalnog osvijetljenja. Posebice su zanimljivi algoritmi bazirani u prostoru slike, budući da je zapravo riječ o slikovnim filtrima, koji vrlo dobro odgovaraju masivno paralelnoj arhitekturi grafičkih procesora. Neka od odličnih svojstava ovakog tipa algoritama su:

- Neovisnost o geometrijskoj kompleksnosti scene.
- Podrška i za statičke, i za dinamičke scene.
- Izvedba isključivo na grafičkoj kartici - nema opterećenja glavnog procesora i radne memorije.

Dakako, postoje i određena ograničenja:

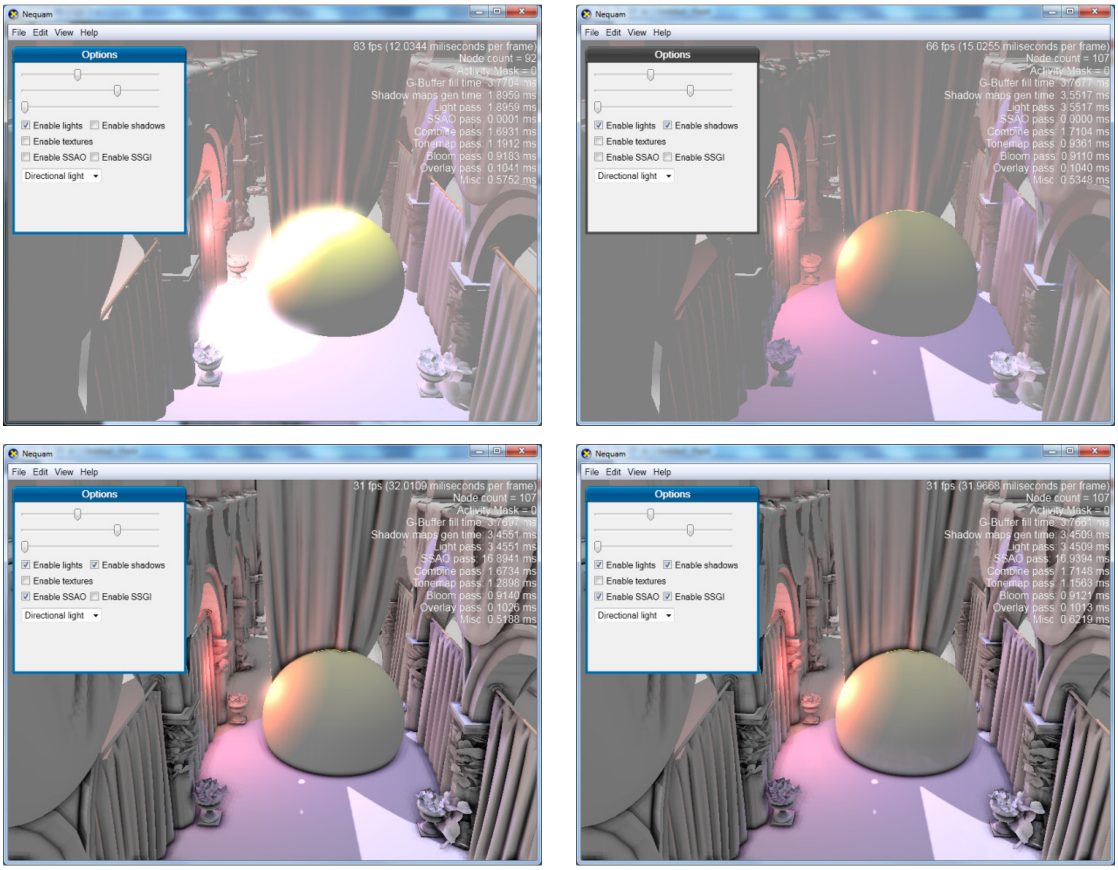
- Nedostupni podaci o stražnjim poligonima.
- Nedostupni podaci o poligonima izvan vidljivog dijela prostora.

Premda algoritmi bazirani u prostoru slike pokazuju određena ograničenja, njihov potencijal je izuzetan, te se u bliskoj budućnosti mogu očekivati zanimljivi novi pomaci u ovom području. Iz tog razloga, za rješenje problema indirektnih sjena, te indirektnog osvjetljenja, odabrani su upravo SSAO, te SSGI algoritam. Kombinacijom njihovih rezultata, s direktnim osvjetljenjem u visokom dinamičkom rasponu, uz preslikavanje sjena, dobiju se oku ugodni rezultati, kao što prikazuje slika 34.



Slika 34. Prikaz konačnog osvjetljenja scene, bez tekstura.

Kroz upotrebu tonskog preslikavanja, omogućen je visoki kontrast između osvjetljenog dijela scene, te dijela scene u sjeni, uz očuvanje detalja. Također, filtriranjem i zamućivanjem svijetlih regija, postignut je efekt prelijevanja svjetlosti, kakav se javlja kod stvarnih leća. Kroz ambijentalno zaklanjanje SSAO algoritmom, istaknuti su prijelazi i rubovi objekata, primjerice kod tkanina. U konačnici, SSGI algoritam na scenu dodaje suptilno indirektno osvjetljenje kao rezultat difuzne interrefleksije. Na slici 35, dan je usporedni prikaz pojedinih komponenti osvjetljenja. Gornja lijeva slika pokazuje samo direktno osvjetljenje, uz konstantnu ambijentalnu komponentu. Gornja desna slika dodaje direktne sjene, kroz algoritam preslikavanja sjena. Na donjoj lijevoj slici, umjesto konstantnog ambijentalnog osvjetljenja, koristi se rezultat SSAO algoritma, što uvelike smanjuje plošni izgled, naglašavajući pukotine i utore u objektima. Napokon, donja desna slika uključuje i indirektno osvjetljenje dobiveno SSGI algoritmom.



Slika 35. Usporedba učinaka pojedinih komponenti osvjetljenja.

12.Literatura

1. Henč-Bartolić, Višnja. Kulišic, Petar. Valovi i optika, izdanje 3, Školska knjiga, 2004.
2. Kajiya, James T. The rendering equation, SIGGRAPH 1986.
3. Goral, C. Torrance, K. E. Greenberg, D. P. Battaile, B. Modeling the interaction of light between diffuse surfaces, Computer Graphics, svezak 18, broj 3.
4. Bunnell, Michael. Dynamic Ambient Occlusion and Indirect Lighting, GPU Gems 2, poglavlje 14, 2006.
5. Shanmugam, Perumaal. Arikan, Okan. Hardware Accelerated Ambient Occlusion Techniques on GPUs. University of Texas, Austin.
6. Filion, Dominic. McNaughton, Rob. Starcraft II: Effects and Techniques, SIGGRAPH 2008.
7. Bavoil, Louis. Sainz, Miguel. Screen Space Ambient Occlusion, SIGGRAPH 2008.
8. Ritschel, Tobias. Grosch, Thorsten. Seidel, Hans-Peter. Approximating Dynamic Global Illumination in Image Space. MPI Informatik.
9. Annen, Thomas. Mertens, Tom. Bekaert, Philippe. Seidel, Hans-Peter. Kautz, Jan. Convolution Shadow Maps. MPI Informatik, Hasselt University, University College London.
10. Annen, Thomas. Mertens, Tom. Seidel, Hans-Peter. Flerackers, Eddy. Kautz, Jan. Exponential Shadow Maps. MPI Informatik, Hasselt University, University College London.
11. Donnelly, William. Lauritzen, Andrew. Variance Shadow Maps. Computer Graphics Lab, School of Computer Science, University of Waterloo.
12. Myers, Kevin. Variance Shadow Mapping. NVIDIA Corporation.
13. Bavoil, Louis. Advanced Soft Shadow Mapping Techniques, GDC 2008.
14. Kawase, Masaki. Real-Time High Dynamic Range Image-Based Lighting.
15. Reinhard, Erik. Stark, Mike. Shirley, Peter. Ferwerda, James. Photographic Tone Reproduction for Digital Images. ACM Transactions on Graphics (TOG), SIGGRAPH 2002.

13.Sažetak

(Globalno difuzno osvjetljenje)

Modeli osvjetljenja koji se danas koriste u interaktivnoj računalnoj grafici su nedostatni, budući da ne obuhvaćaju globalne učinke gibanja svjetla, poput pretapanja boja, mekih sjena, kaustike, ispodpovršinskog raspršivanja itd. Tradicionalni postupci poput praćenja zrake točno rješavaju ovaj problem, no njihova primjena u interaktivnim aplikacijama je tek eksperimentalna, zbog još uvijek vrlo visokih računalnih zahtjeva. Iz tog razloga, problem globalnog osvjetljenja razbijamo u manje podprobleme, uz razumne pretpostavke i ograničenja. Predstavljaju se razne tehnike i postupci za rješavanje pojedinačnih problema, poput direktnih sjena, ambijentalnih sjena, indirektnog osvjetljenja i drugih efekata. Najuočljiviji učinak globalnog osvjetljenja jest difuzna interrefleksija, koja uzrokuje indirektno osvjetljenje, te ambijentalno zaklanjanje. Za ostvarivanje ovih efekata, istraženi su postupci u prostoru slike, zbog svojih odličnih svojstava.

Ključne riječi: globalno osvjetljenje, tehnike iscrtavanja, zaklanjanje ambijenta, indirektno osvjetljenje, difuzna interrefleksija.

14.Abstract

(Global diffuse lighting)

Lighting models currently used in interactive computer graphics are insufficient, as they do not encompass the global effects of light propagation, such as color bleeding, soft shadows, caustics, subsurface scattering, etc. Traditional algorithms such as ray tracing solve this problem correctly, however, their use in interactive applications is still experimental, due to high computational costs. Therefore, we split the global illumination problem into smaller sub-problems, with reasonable assumptions and limitations. We present various techniques and algorithms for solving individual problems such as direct shadows, ambient shadows, indirect lighting, etc. The most notable effect of global illumination is diffuse interreflection, which causes indirect lighting and ambient occlusion. To achieve these effects, we investigate algorithms based in screen space, due to their excellent properties.

Key words: global illumination, rendering techniques, ambient occlusion, indirect lighting, diffuse interreflection.

15.Zahvale

Ovim putem, zahvaljujem sljedećim osobama:

Luka Piljek - suradnja pri dizajnu integriranog sustava grafa scene i sustava prozora, korištenog kao temelj implementacije grafičkog pogona.

José María Méndez González - ideja korištenja Bunnellove funkcije zaklanjanja, te korisna diskusija o SSAO algoritmu.

Željka Mihajlović - konstruktivne kritike i pomoć pri pisanju rada.