SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 38

POSTUPCI OSTVARIVANJA GLOBALNOG OSVJETLJENJA

Robert Sajko

Zagreb, lipanj 2008.

Sadržaj

1. Uvod

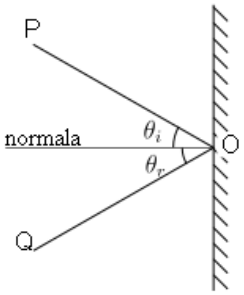
Čovjek je po svojoj prirodi vizualno biće – upravo iz osjetila vida crpimo najviše informacija o svojoj okolini. Međutim, da bismo vidjeli, potrebno je svjetlo. Interakcijama svjetlosti sa raznim predmetima koji nas okružuju nastaju različite fizikalne, optičke pojave. Ljudsko osjetilo vida je osobito 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 (Slika 1). No, takvi proračuni su 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 su se razvijali modeli globalnog osvjetljenja, koji su zasnovani na fizikalno potpuno točnim simulacijama, te modeli lokalnog osvjetljenja, koji točno simuliraju samo uski dio fizikalnih pojava vezanih uz svjetlost, dok ostatak aproksimiraju nekim konstantama, koje se mogu "namjestiti" da daju otprilike zadovoljavajuće rezultate. S vremenom, najpopularniji od tih modela lokalnog osvjetljenja (Blinn-Phong model) je postao 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 ovome radu, krenut ćemo obrnutim putem. Prvo ćemo točno definirati koja su to svojstva i fizikalne pojave vezane uz svjetlost, a zatim ćemo izvesti univerzalnu jednadžbu iscrtavanja koja obuhvaća sve navedene fizikalne pojave. Jednom kada je ta jednadžba zadana, moći ćemo iz nje izvesti i modele lokalnog, i modele globalnog osvjetljenja, aproksimirajući određene njene dijelove. 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.

Slika 1. Na ovoj slici vidite realnu fotografiju čaše mlijeka, pored koje je naknadno dodana računalno isctana čaša. Možete li odrediti koja je čaša stvarna?

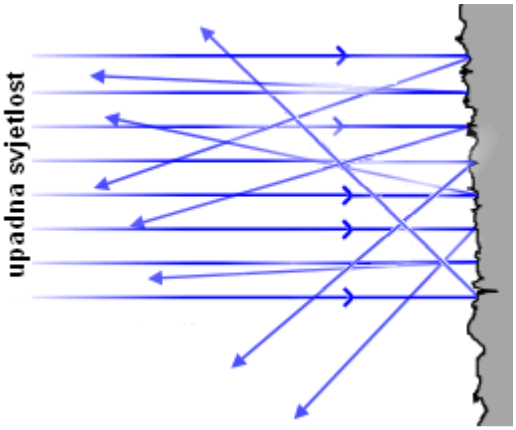
2. Fizikalni model svjetla

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 svjelosti, 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 najučestaliji 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 kutevi definiraju s obzirom na normalu površine na koju svjetlost upada. Zakon refleksije jednostavno kaže: upadni kut je jednak kutu refleksije (Slika 2).



Slika 2. Zrcalna refleksija.



Slika 3. Difuzna refleksija.

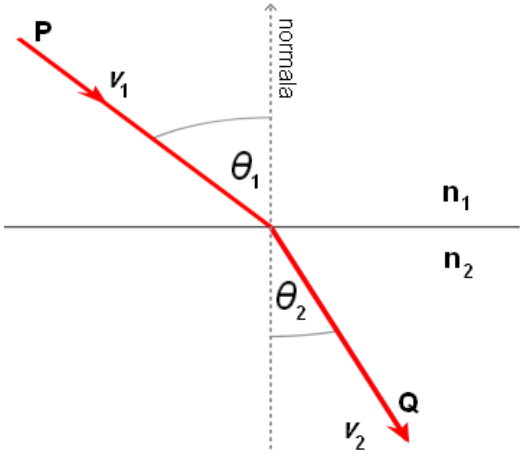
Međutim, valja imati na umu da većina realnih tijela imaju površine koje nisu savršeno glatke. To znači da će normale na površinu biti različite za pojedine upadne zrake svjetlosti, a to nadalje znači i da će kutevi refleksija biti različiti za pojedine zrake, makar sve imaju iste upadne kuteve s obzirom na ravninu površine tijela. Drugim riječima, na grubim površinama će reflektirane zrake biti raspršene u različitim smjerovima (Slika 3). 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.

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 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:

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



Slika 4. Refrakcija zrake svjetlosti.

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:

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

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 n1 indeks loma prvog sredstva, a n2 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.

Slika 5. Primjer kaustike.

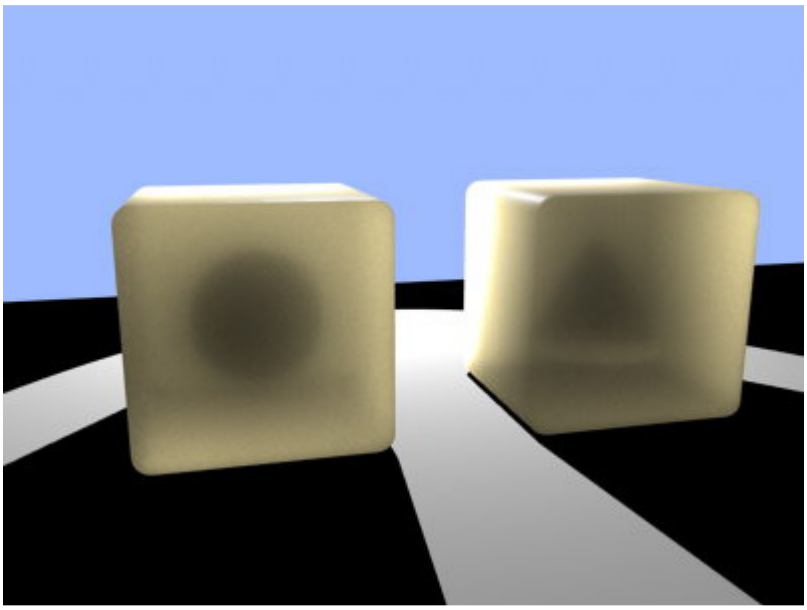
Također, pod kaustikom se može podrazumijevati i projekcija te ovojnice na neku drugu površinu. Dobar primjer kaustike može proizvesti prazna staklena čaša na praznom stolu, na koju upada sunčeva svjetlost. Radi opetovanih refrakcija i refleksija zraka svjetlosti uvjetovanih strukturom čaše, izlazne zrake svjetlosti mogu na stolu projicirati neobičan oblik. Takvi se oblici općenito nazivaju kaustika (Slika 5).

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, te zbog njegove 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 refleksiran i refraktiran, dok će ostatak biti absorbiran. 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 absorbirati foton samo točno odgovarajuće energije (što ovisi o vrsti atoma), očito je da će samo dio fotona iz bijele svijetlosti biti absorbiran, a ostatak će biti odbijen, dakle, promijenit će im se smjer i brzina. Koje valne duljine svjetlosti će biti absorbirane, 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 absorbirani, 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 6 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 6. Primjer ispodpovršinskog raspršivanja.

Ovime smo ukratko pokrili sve bitne učinke i svojstva svjetlosti, promatrane kroz realne fizikalne modele. Budući da je naša primarna zadaća izraditi teorijski model osvjetljenja za koji je ključno da bude pogodan generiranju računalne grafike, već u ovome stadiju možemo uvesti određene aproksimacije. Naime, možemo primijetiti da većina pojava koje se mogu simulirati isključivo izračunom valnih svojstava svjetlosti, su ili ionako gotovo nemoguće za primijetiti (polarizacija svjetlosti), ili su izražajne tek u nekim posebnim slučajevima (primjerice, difrakcija svjetlosti na poleđini CD-a). Iz tog razloga, mnogo je isplativiji pristup simulacije isključivo geometrijske optike (što je računalno mnogo manje zahtjevno), te rješavanja posebnih slučajeva u kojima geometrijska optika nije dostatna raznim trikovima, kao što su pažljivo teksturiranje i slično.

Sad kad smo odredili glavne smjernice našeg modela osvjetljenja, možemo pokušati sažeti načela geometrijske optike u jedinstven matematički zapis. Prema dosad rečenome, možemo ponešto zaključiti o tome koliko bi bilo ukupno osvjetljenje u danoj točki prostora. Naime, 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 drugim izvorom svjetlosti, ili pak rezultat (višestruke) refleksije svjetlosti od objekata na sceni. Također, znamo da se svjetlost reflektira od različitih objekata na drugačiji način, pa možemo definirati funkciju distribucije refleksivnosti (eng. *bidirectional reflectance distribution function, BRDF*), koja govori u kojoj mjeri se svjetlost reflektira od danog materijala. Sve dosad rečeno možemo formalno zapisati na sljedeći način:

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

gdje je:

$L_o(x, \vec{w})$ izlazna svjetlost na poziciji *x*, u smjeru $\vec{w}$,

$L_e(x, \vec{w})$ svjetlost emitirana sa pozicije *x*, u smjeru $\vec{w}$,

$\int_\Omega ...d\vec{w'}$ integral upadnih zraka svjetlosti preko polukugle,

$f_r(x, \vec{w'}, \vec{w})$ mjera svjetlosti reflektirane na poziciji *x*, iz upadnog prema izlaznom smjeru (BRDF)

$L_i(x, \vec{w'})$ upadna svjetlost na poziciji *x*, u smjeru $\vec{w'}$,

$(\vec{w'} \cdot \vec{n})$ atenuacija upadnog svjetla zbog kuta upada, gdje je $\vec{n}$ normala na površinu.

Iako ovu jednadžbu nije teško postaviti, mnogo je veći problem točno je riješiti. Naime, ne samo da je problematično iznaći analitičko rješenje i prevesti ga u jednostavne operacije zbrajanja i množenja, koje računala mogu efikasno izvršavati, već vidimo i da potencijalna kompleksnost izračuna rješenja strmoglavo raste s kompleksnosti i međuodnosima objekata na sceni. Dakle, vidimo da smo suočeni s dva problema – prvi problem jest pronaći način za efikasno numeričko rješavanje dane integralne jednadžbe osvjetljenja. Drugi problem proizlazi iz zahtjeva za interaktivnošću – izvršavanje ovih izračuna mora biti u realnom vremenu. S vremenom se razvilo nekoliko pristupa i algoritama u računalnoj grafici, koji ili favoriziraju kvalitetu prikaza nauštrb interaktivnosti, ili pružaju interaktivnost nauštrb kvalitete. U prvoj skupini se nalaze algoritmi i metode koje pokušavaju raznim numeričkim metodama direktno riješiti jednadžbu iscrtavanja, te se zajednički nazivaju metode globalnog osvjetljenja. Primjerice, tu spadaju direktna simulacija geometrijske optike (ray traycing i srodni algoritmi), razne Monte Carlo metode (path tracing, photon mapping), te metode konačnih elemenata (radiosity). U drugu skupinu spadaju takozvani lokalni modeli osvjetljenja, koji su nastali kao aproksimacija jednadžbe iscrtavanja. Najosnovniji takav model jest Lambertov model refleksije, no mnogo popularnija je njegova proširena verzija poznata kao Phongov model. U sljedećim poglavljima će detaljnije biti objašnjene obje skupine algoritama, te prikazani pokušaji premošćivanja jaza između interaktivnosti i globalnog osvjetljenja.

3. Lokalni modeli osvjetljenja

Osnovna ideja svih lokalnih modela osvjetljenja, po kojoj su i dobili naziv, jest zanemarivanje globalnih učinaka propagacije svjetlosti. Naime, kako je rečeno u prethodnom poglavlju, ukupno osvjetljenje u danoj točki prostora je jednako zbroju doprinosa svih zraka svjetlosti koje upadaju, ili bivaju emitirane s te točke. Odnosno, možemo reći da kako svjetlost iz nekog izvora upada na objekte na sceni, te biva reflektirana, tako ti objekti djeluju kao sekundarni izvori osvjetljenja, i stvaraju ambijentalno osvjetljenje uz razne popratne učinke poput pretapanja boja. Upravo ovoj pojavi odgovara integralni član jednadžbe iscrtavanja:

$$\int_{\Omega} f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) d\vec{w}'$$

Budući da upravo ovaj član jednadžbe predstavlja glavninu poteškoća pri računanju osvjetljenja, prirodno se nameće ideja za njegovom aproksimacijom, ili čak potpunim zanemarivanjem. Naime, analiziranjem velikog broja raznolikih prirodnih uvjeta osvjetljenja, lako se može uočiti da je u većini slučajeva indirektno osvjetljenje, nastalo refleksijom svjetlosti između objekata na sceni, za red veličine manjeg intenziteta od direktnog osvjetljenja. Također, direktno osvjetljenje pokazuje oštre prijelaze u intenzitetu, te stoga iscrtava i oštre sjene. S druge strane, oscilacije u indirektnom osvjetljenju kroz prostor su veoma blage, to jest, indirektno osvjetljenje je približno ravnomjerno rasprostranjeno u prostoru, sa tek blagim gradijentima u intenzitetu, te stoga stvara i blage, meke sjene. Ukoliko bi indirektno, ambijentalno osvjetljenje doista bilo potpuno homogeno, tada bi izraz za intenzitet indirektnog osvjetljenja u bilo kojoj točki prostora bio sveden na jednu jedinu konstantu. Drugim riječima, jednadžba iscrtavanja tada prestaje biti integralna jednadžba, i biva svedena na proračun isključivo direktnog osvjetljenja – što znači jednostavno zbrajanje i množenje:

$$L_o(x, \vec{w}) = \sum_{\substack{\textbf{izvori} \\ \textbf{svjetla}}} f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n})$$

Upravo je ova ideja temelj svih lokalnih modela osvjetljenja, i time interaktivne računalne grafike. No, premda je ovakav oblik jednadžbe iscrtavanja daleko jednostavniji za izračunavanje od početnoga, postoji još jedan trik koji se sve do nedavno redovito i bez iznimke primjenjivao u interaktivnoj računalnoj grafici, a kojim se dodatno smanjuje vrijeme izvršavanja algoritma. Naime, od samog početka smo razmatrali osvjetljenje kao funkciju u tri dimenzije, koja svakoj točki prostora preslikava neku vrijednost osvjetljenja. Međutim, sjetimo se da su objekti u virtualnoj sceni prezentirani mrežom trokuta, a svaki trokut je definiran svojim trima vrhovima. Prema tome, objekt se može promatrati kao skup vrhova, odnosno kao konačni, diskretni skup točaka. Iz ovoga slijedi da je umjesto evaluacije izraza za osvjetljenje za svaku točku površine nekog objekta, odnosno za svaki pojedini slikovni element, dovoljno izračunati osvjetljenje za svaki vrh objekata na sceni. Nakon toga, u fazi rasterizacije konačne slike, vrijednosti izračunate za pojedine vrhove se lako mogu interpolirati nad slikovnim elementima. Dakako, da bi jednadžba iscrtavanja bila potpuna, još se postavlja pitanje kako postaviti i računati funkciju distribucije reflektivnosti. Računski najnezahtjevnija jest difuzna refleksija, kod koje se upadna svjetlost raspršuje u svim smjerovima jednako, tako da je osvjetljenje tijela jednoliko po

cijeloj površini. Već i ovakav model je dovoljno potpun za izračunavanje osvjetljenja koje ne djeluje potpuno nerealno. Tada jednadžba iscrtavanja poprima ovaj oblik:

$$L_o(x,\vec{w}) = \sum_{\substack{\textbf{izvori}\\ \textbf{svjetla}}} L_i(x,\vec{w}')(\vec{w}'\cdot\vec{n})$$

Raspisivanjem gornjeg skalarnog produkta se dobiva Lambertov kosinusni zakon, zbog čega se ovaj model naziva Lambertov model osvjetljenja. Ipak, ovaj se model može lako proširiti da daje mnogo uvjerljivije rezultate, a bez drastičnog povećanja kompleksnosti. Prva ideja koja se nameće jest dodavanje konstantnog člana, za barem vrlo grubu aproksimaciju indirektnog, ambijentalnog osvjetljenja. Druga stvar koju možemo pokušati implementirati za poboljšanje kvalitete prikaza jest zrcalna refleksija, budući da ju većina tijela pokazuje bar u nekoj mjeri. Drugim riječima, zanemarit ćemo pretpostavku da je funkcija distribucije refleksije jednolika, te ćemo je pokušati definirati na prikladan način. Razvijeno je nekoliko takvih načina, čime su i definirani različiti modeli lokalnog osvjetljenja. Najpoznatiji takav jest Phongov model, koji se zasniva na opažanju da vrlo sjajne površine imaju vrlo uzak zrcalni odbljesak, čiji intenzitet opada vrlo naglo. Površine manje sjajnosti imaju širi odbljesak, koji opada sporije. Krajnosti su potpuna sažetost odbljeska u jednoj točki, ili pak rasprostiranje odbljeska preko čitave površine (ogledala). Također, može se primijetiti da zrcalni odbljesak ovisi o kutu gledanja. Shodno svemu navedenome, za svaki objekt u virtualnoj sceni se definira *materijal*, odnosno skup konstanti koje određuju svojstva objekta pri interakciji sa svjetlošću:

- k_d mjera difuzne komponente refleksije
- k_s mjera zrcalne komponente refleksije
- k_a mjera ambijentalne refleksije
- α sjajnost materijala (širina zrcalnog odbljeska)

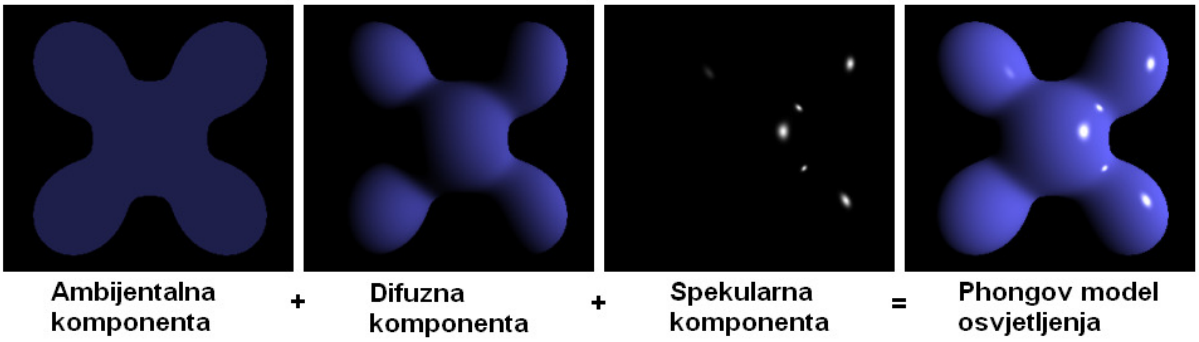
Dakle, materijal objekta određuje parametre funkcije distribucije refleksivnosti. Sama funkcija se definira na sljedeći način:

$$f_r(x,\overrightarrow{w},\overrightarrow{w'}) = k_d\, L_{id}(x,\overrightarrow{w'})\,(\overrightarrow{w'}\cdot\overrightarrow{n}) \;+\; k_s\, L_{is}(x,\overrightarrow{w'})\,(\overrightarrow{w'}\cdot\vec{v})^{\alpha}$$

Prvi član odgovara difuznoj refleksiji, i po svom je obliku jednak Lambertovom modelu osvjetljenja. Drugi član odgovara zrcalnoj refleksiji. Budući da zrcalni odbljesak ovisi o kutu gledanja, potrebno je uvesti vektor $\vec{v}$, koji označava odnos točke promatrača prema danoj točki upada svjetlosti. Skalarni umnožak vektora iz točke na objektu (x) prema očištu i vektora smjera zrcalno reflektirane zrake, potenciran konstantom sjajnosti, daje mjeru zrcalnog osvjetljenja za danu točku. Na kraju, konačna jednadžba iscrtavanja glasi ovako:

$$L_0(x,\overrightarrow{w}) = k_a\, L_{ia}(x,\overrightarrow{w'}) + \sum_{\substack{\textit{izvori}\\ \textit{svjetla}}} \left(k_d\, L_{id}(x,\overrightarrow{w'})\,(\overrightarrow{w'}\cdot\overrightarrow{n}) \;+\; k_s\, L_{is}(x,\overrightarrow{w'})\,(\overrightarrow{w'}\cdot\vec{v})^{\alpha} \right)$$

Početni član označava konstantno ambijentalno osvjetljenje, dok izraz pod sumom odgovara difuzno – zrcalnoj refleksiji, za svako pojedino svjetlo.



Slika 7. Komponente osvjetljenja u Phongovom modelu.

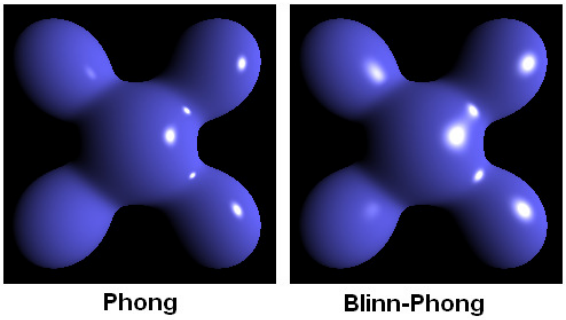
Ovdje valja spomenuti da osim Phongovog, postoji još mnogo drugih lokalnih modela osvjetljenja, koji bolje ili lošije opisuju razne svjetlosne učinke i karakteristike, a koji se razlikuju po definiciji funkcije distribucije reflektivnosti: Gaussov, Beckmannov, Heidrich –Seidelov, Wardov, Cook – Torranceov model, itd. Međutim, daleko najpopularniji od svih jest varijacija gore opisanog Phongovog modela, poznata kao Blinn-Phong model osvjetljenja. Jedina promjena u odnosu na standardni Phongov model jest u vizualno gotovo neprimjetnoj aproksimaciji izraza sa zrcalnu komponentu osvjetljenja. Naime, možemo definirati vektor $\vec{h}$ na sljedeći način:

$$\vec{h} = \frac{\overrightarrow{w'} + \vec{v}}{\left|\overrightarrow{w'} + \vec{v}\right|}$$

Ovaj vektor se naziva poluvektorom između vektora prema izvoru svjetlosti i vektora prema promatraču. Sada možemo u funkciji distribucije refleksivnosti promijeniti izraz za zrcalnu komponentu osvjetljenja na način da skalarni umnožak vektora smjera reflektirane zrcalne zrake i vektora gledišta, $\overrightarrow{w'} \cdot \vec{v}$, zamijenimo skalarnim umnoškom vektora normale na površinu i poluvektora, $\vec{n} \cdot \vec{h}$. No, kut $\sphericalangle\left(\vec{n} \cdot \vec{h}\right)$ će uvijek biti manji od kuta $\sphericalangle\left(\overrightarrow{w'} \cdot \vec{v}\right)$, i to točno upola manji ukoliko $\overrightarrow{w'}$, $\vec{v}$, $\vec{n}$ i $\vec{h}$ leže na istoj ravnini. Da bi krajnji rezultat ostao numerički približno jednak, potrebno je malo podesiti konstantu α, odnosno, pronaći takav α' da vrijedi $\left(\overrightarrow{w'} \cdot \vec{v}\right)^{\alpha} = \left(\vec{n} \cdot \vec{h}\right)^{\alpha'}$. Dakle, jednadžba iscrtavanja će na kraju izgledati ovako:

$$L_0(x, \vec{w}) = k_a \, L_{ia}(x, \overrightarrow{w'}) + \sum_{\substack{izvori \\ svjetla}} \left(k_d \, L_{id}(x, \overrightarrow{w'}) \, (\overrightarrow{w'} \cdot \overrightarrow{n}) \; + \; k_s \, L_{is}(x, \overrightarrow{w'}) \, (\vec{n} \cdot \vec{h})^{\alpha'} \right)$$

Blinn-Phongov model je efikasniji od standardnog Phongovog modela, jer je za usmjerena svjetla poluvektor potrebno izračunati samo jedamput za svako svjetlo, pa je ukupna količina posla koju je potrebno obaviti prilikom svakog iscrtavanja manja. Numerička nepreciznost do koje dolazi uslijed navedenih supstitucija je gotovo neprimjetna (slika 8.), te je ukupni dobitci na performansama potpuno opravdavaju. Zbog svojih odličnih



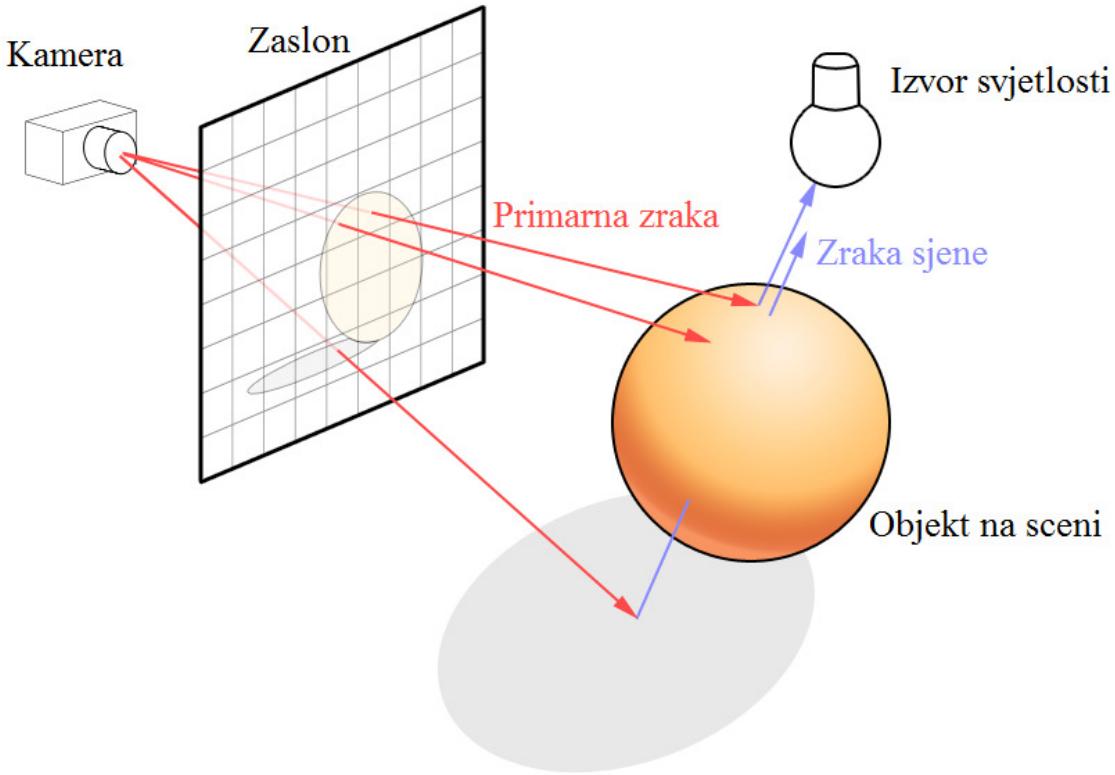
Slika 8. Vizualna razlika između Phong i Blinn-Phong modela osvjetljenja.

svojstava (visoka efikasnost i brzina, realistični rezultati), Blinn-Phong model je postao toliko popularan da je standardiziran kao podrazumijevani model osvjetljenja svih poznatijih grafičkih programskih sučelja (OpenGL, Direct3D), te sklopovski implementiran na svim grafičkim karticama koje pružaju 3D ubrzanje (doduše, u novije vrijeme, grafičke kartice s programirljivim cjevovodima dopuštaju implementaciju vlastitih, proizvoljnih modela osvjetljenja).

4. Globalni modeli osvjetljenja

Jedna od najstarijih skupina algoritama u računalnoj grafici uopće su takozvani algoritmi praćenja zrake. Osnovna ideja je vrlo jednostavna – principi i zakonitosti geometrijske optike se direktno simuliraju računalom. Dakle, pretpostavlja se da izvori svjetlosti emitiraju svjetlost u zrakama, koje se mogu predstaviti pravcima (ili linijskim segmentima). Zatim je potrebno za svaku od tih zraka odrediti da li presijeca neki od objekata na sceni, te po potrebi generirati novu zraku (reflektiranu i/ili refraktiranu). Na kraju, zrake koje upadaju na virtualnu kameru, odnosno, na ravninu projekcije, definiraju slikovni element na koji upadaju, a pritom se za izračun samih doprinosa svjetlosti za pojedinu zraku koristi neki od lokalnih modela osvjetljenja. Na ovaj način se na zaslonu generira 'pogled' u virtualnu scenu. U opisanom postupku je moguće uvesti nekoliko varijacija, pa se tako razlikuje i nekoliko srodnih algoritama. Osnovna varijacija u odnosu na standardni postupak jest u praćenju zrake u obrnutom smjeru; umjesto praćenja pojedine zrake svjetlosti od izvora, preko objekata na sceni do kamere, zraka se prati od kamere, prema objektima na sceni. Naime, možemo primjetiti da u gotovo svim slučajevima, većina zraka svjetlosti koje izlaze iz izvora uopće ne dotiču okolne predmete, ili ne dotiču kameru. Prema tome, ukoliko pratimo zrake iz kamere, sigurno ćemo morati računati samo ono što nam je potrebno za danu scenu, čime drastično povećavamo efikasnost algoritma. Ovdje valja napraviti razliku između algoritma bacanja zraka (*ray casting*) i algoritma praćenja zraka (*ray tracing*). Naime, oba algoritma prate zrake od kamere do objekata na sceni, no algoritam bacanja zraka se zaustavlja kad pronađe sjecište trenutne zrake s najbližim objektom na sceni, dok algoritam praćenja zrake u tom trenutku generira nove zrake (prema zakonima geometrijske optike), te se rekurzivno poziva i za te nove zrake. Očito, prvi algoritam ne može simulirati učinke globalnog osvjetljenja, ali je mnogo brži, te se koristio u počecima interaktivne računalne grafike (primjerice, u igri Wolfenstein 3D). Drugi algoritam je sposoban doći mnogo bliže potpunom rješenju jednadžbe iscrtavanja, ali je mnogo sporiji. Dodatna optimizacija koja se obično koristi u algoritmu praćenja zrake su takozvane zrake sjene (*shadow rays*). Naime, kada se pronađe sjecište trenutne zrake sa najbližim objektom na sceni, prije stvaranja reflektirane i/ili refraktirane zrake (već prema svojstvima materijala), provjerava se da li je presječena površina (poligon) usmjeren prema nekom izvoru svjetlosti. Ukoliko jest, stvara se zamišljena zraka sjene, usmjerena od površine prema izvoru. Ukoliko tu zraku presijeca neki neprozirni objekt (dakle, ako zraka sjene presijeca neku drugu neprozirnu površinu), tada je površina koju ispitujemo u sjeni, te nije potrebno stvarati reflektiranu i/ili refraktiranu zraku, što znači da nije potrebno ulaziti u novu dubinu rekurzije. Na priloženoj ilustraciji (Slika 9) vidimo primjer rada algoritma. Promatramo tri primarne zrake koje pratimo iz kamere. Gornje dvije upadaju na kuglu, te njihove pripadne zrake sjene ne presijecaju nikakve druge površine na putu do izvora svjetlosti. Dakle, u tim bi se slučajevima generirale nove zrake, nastale refleksijom i refrakcijom, te bi se algoritam rekurzivno pozvao. Za treću

primarnu zraku, koja upada na pod ispod objekta, zraka sjene presijeca neprozirnu površinu objekta, te se algoritam zaustavlja.

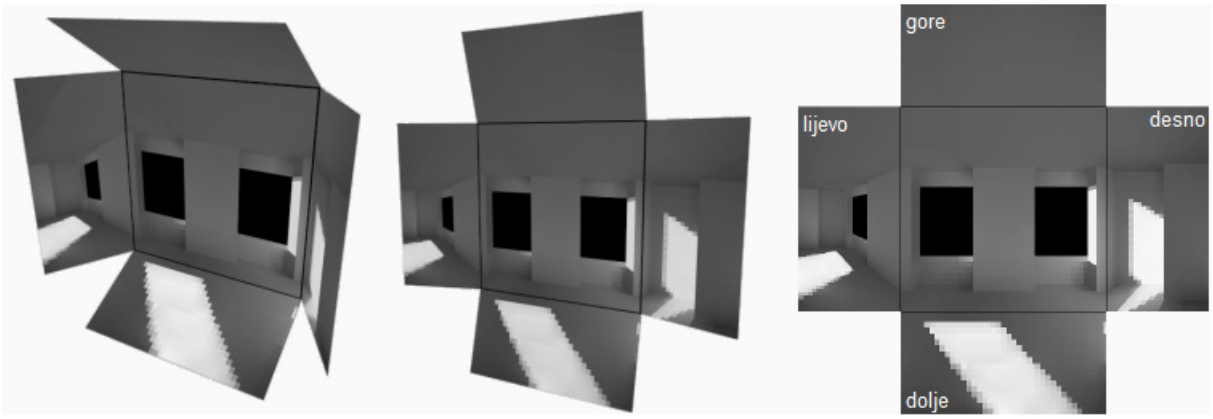


Slika 9. Ilustracija algoritma praćenja zrake.

Postoji nekoliko nadogradnji na ovaj temeljni algoritam koje se baziraju na Monte Carlo (probabilističkim) tehnikama, od kojih je temeljni predstavnik algoritam praćenja puta. Postupak je vrlo sličan standardnom, samo se zrake generiraju na način da prate slučajno odabrani put. Dakle, u svakoj točki presjeka zrake sa objektom na sceni, generira se nova zraka, ali u nekom slučajno odabranom smjeru. Postupak se ponavlja dokle god postoji presjek promatrane zrake sa nekim objektom na sceni, a izračunati svjetlosni doprinosi se zbrajaju po dubinama rekurzije. Mnogo napredniji probabilistički postupak jest algoritam preslikavanja fotona (*photon mapping*), koji zajedno sa standardnim tehnikama praćenja zrake pruža veoma realistične rezultate, s ispravnim efektima kaustike i mekim sjenama. Ovaj se algoritam sastoji od dvije faze. U prvoj fazi se konstruira spremnik fotona, a u drugoj fazi se iscrtava sama scena. Spremnik fotona se popunjava tako da se uzduž zraka svjetlosti odašilju fotoni. Ukoliko zraka presijeca neki objekt, njeni se fotoni pohranjuju u spremnik, te se izvršava neka od sljedećih akcija: generiranje reflektirane i/ili refraktirane zrake (rekurzivno ponavljanje algoritma), apsorpcija (kraj algoritma). Ove se akcije izvršavaju slučajnim odabirom, a vjerojatnost odabira pojedine akcije je definirana materijalom objekta. U drugoj fazi, scena se iscrtava klasičnim postupkom praćenja zrake, no umjesto korištenja nekog modela lokalnog osvjetljenja za izračun svjetlosnih doprinosa, koristi se prethodno generiran spremnik fotona za proračun funkcije distribucije refleksivnosti.

Opisani algoritmi se svi baziraju na osnovnom principu direktne simulacije zakonitosti (geometrijske) optike. No, postoji i alternativni pristup, koji se bazira na simulaciji

toplinskog zračenja (algoritam isijavanja, eng. *radiosity*). Naime, prisjetimo se da kod difuzne refleksije, tijelo isijava svjetlost jednoliko preko čitave svoje površine – baš kao i toplinu! Sam algoritam funkcionira na sljedeći način: za svaki par poligona koji sačinjavaju cjelokupnu scenu, definira se faktor vidljivosti. Taj faktor govori u kojoj mjeri jedan poligon „vidi“ drugoga, a matematički se izražava kao mjera radijacije koja izlazi iz jedne površine, a upada na drugu. Postoji nekoliko načina pomoću kojih se mogu dobiti ti faktori, od kojih je najjednostavnije korištenje polukocaka. U ovom kontekstu, pod pojmom „polukocka“ se podrazumjeva 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 sa odnosom stranica 2:1.



Slika 10. Polukocka i njena pripadna mreža.

Takva polukocka se centrira s obzirom na promatrani poligon, te se cijela scena iscrta pet puta, za svaku stranicu polukocke, i to tako da gledište leži na normali stranice. Na taj način se dobiju projekcije okolnih poligona na promatrani poligon, te se može izračunati mjera preklapanja površina tih poligona – upravo ta veličina jest faktor vidljivosti. Jednom kad izračunamo faktore vidljivosti za sve parove poligona, možemo upotrijebiti formulu za ukupno zračenje površine, koja kaže da je ukupno zračenje jednako zbroju emitiranog zračenja, i reflektiranog zračenja:

$$B_i = E_i + R_i \int_j B_j F_{ij}$$

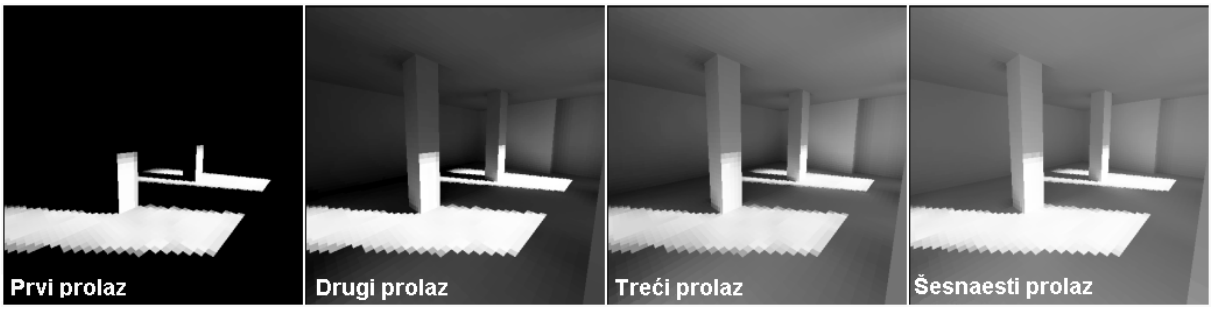
gdje je:

- B_i ukupno zračenje površine *i*
- B_j ukupno zračenje površine *j*
- E_i emitirano zračenje površine *i*
- R_i mjera reflektivnosti površine *i*
- F_{ij} faktor vidljivosti površine *j* sa površine *i*

Ukoliko pretpostavimo savršeno difuznu refleksiju, takvu da se svjetlost odbija savršeno jednoliko u svim smjerovima, lako možemo provesti diskretizaciju i pretvoriti gore napisanu integralnu jednadžbu u sumacijsku:

$$B_i = E_i + R_i \sum_{j=1}^{n} B_j F_{ij}$$

U ovom obliku, jednadžba se lako može izračunati za svaki poligon, te se „zračenje“ nekog poligona može jednostavno interpretirati kao njegova svjetlina, odnosno, boja. Međutim,



Slika 11. Primjer korištenja iterativnog postupka u algoritmu isijavanja.

pošto je svjetlina svakog poligona definirana zračenjima njemu vidljivih poligona, a na početku su zračenja svih poligona koji nisu izvori svjetla jednaka nuli, možemo zaključiti da će algoritam u gore opisanom obliku osvjetljavati samo one poligone na koje upada osvjetljenje direktno iz izvora, te njima susjedne poligone koje algoritam obradi poslije tih direktno osvijetljenih poligona. Dakle, dešava se situacija da poligoni koji bi trebali biti osvijetljeni svjetlošću reflektiranom od nekog poligona koji je direktno osvijetljen, nisu osvijetljeni čisto iz razloga što algoritam u trenutku njihove obrade još nije bio stigao obraditi direktno osvijetljeni poligon, pa je njegovo zračenje ostalo na nuli! Rješenje je vrlo jednostavno – jedna iteracija algoritma nije dovoljna, stoga se provodi dodatna. No, i u drugoj iteraciji se ponovno javlja isti problem, budući da promjenom zračenja indirektno osvijetljenih poligona, oni sami postaju sekundarni izvori svjetla. Također, ti sekundarni izvori svjetlosti mogu opet osvjetljavati neke nove poligone, koji i sami postaju novi izvori svjetlosti, što dovodi do zaključka da je potreban beskonačan broj iteracija algoritma. Međutim, tome ipak nije tako, budući da s brojem iteracija indirektni doprinosi osvjetljenju značajno opadaju, dok napokon ne postanu zanemarivi. Taj prag zanemarivosti ovisi o karakteristikama scene, no za većinu praktičnih primjena, osam do šesnaest iteracija je sasvim zadovoljavajuće. Algoritam, zbog svoje prirode, veoma dobro simulira efekte globalnog osvjetljenja kao što su pretapanje boja i meke sjene, no nažalost, prisutno je fundamentalno ograničenje na isključivo difuzno osvjetljenje. Zanimljiva je komplementarnost algoritma isijavanja i algoritma praćenja zrake; prvi dobro simulira efekte poput difuznih interrefleksija i mekih sjena, ali se ne može koristiti za zrcalnu refleksiju, dok drugi točno simulira zrcalnu refleksiju i oštre sjene, ali difuzna refleksija i meke sjene su teže za izvesti. U obliku u kojem je predstavljen, algoritam isijavanja ima kvadratnu složenost s obzirom na broj poligona od kojih je sačinjena scena. Međutim, postoje razna unaprijeđenja osnovnog algoritma, kojima se smanjuje vremenska kompleksnost – primjerice, korištenjem binarne podjele prostora, problem određivanja vidljivosti poligona se može svesti na logaritamsku kompleksnost. Nadalje, samo računanje faktora vidljivosti pomoću polukocaka je također vremenski dosta zahtjevno, budući da se cjelokupna scena mora iscrtati pet puta. U općenitom slučaju, ti se faktori zbog deformacije objekata na sceni mogu mijenjati prilikom svakog iscrtavanja, pa se svaki put moraju nanovo i izračunavati, no u većini stvarnih situacija, nagle i potpune promjene vidljivosti se dešavaju vrlo rijetko. To znači da između pojedinih iscrtavanja scene postoji značajna kohezija između faktora vidljivosti, a to se svojstvo može iskoristiti za dodatnu optimizaciju algoritma [4].

Međutim, uz sve optimizacije, niti algoritmi isijavanja, niti algoritmi praćenja zrake sve do nedavno nisu bili upotrebljivi u interaktivnoj računalnoj grafici, iz jednostavnog razloga nedostatka računalne moći. No, kako dostupna računalna moć pokazuje tendenciju stabilnog rasta, tako se javlja i ideja interaktivnog globalnog osvjetljenja. U tom kontekstu, postoji nekoliko različitih pristupa, koji se ili razvijaju u sklopu trenutno prevladavajućih

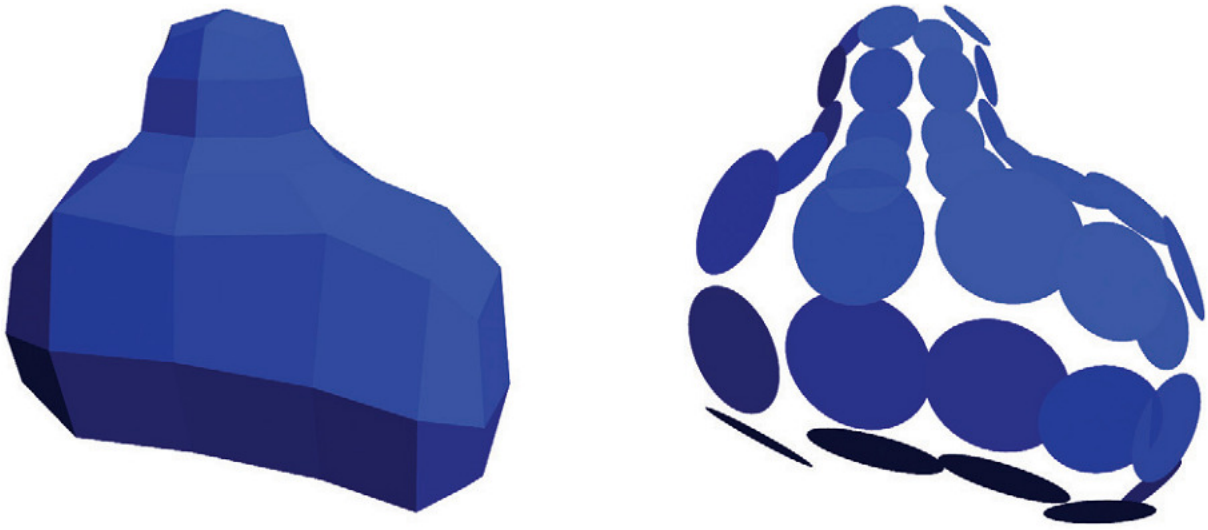
rasterizacijskih tehnika, ili na temelju postupaka praćenja zrake i/ili isijavanja. Algoritam praćenja zrake je posebice zanimljiv u tom pogledu, jer ga je moguće veoma lako paralelizirati. Naime, za svaki slikovni element zaslona, posebno se vrši proračun osvjetljenja, i to neovisno o svim drugim slikovnim elementima. To znači da se svi slikovni elementi mogu obrađivati istovremeno! Ovo svojstvo je izuzetno zanimljivo s obzirom na tendenciju rasta broja jezgri procesora – teoretski, s *n* dostupnih jezgri, iscrtavanje bi trebalo biti *n* puta brže. Prema tome, već i danas dostupnom tehnologijom bi se trebale moći postići gotovo interaktivne performanse, barem na igrama prethodne generacije. Potvrdu ovakvom razmišljanju daje Intelova istraživačka grupa vođena Danielom Pohlom, prilagodivši komercijalnu igru Quake IV, izdanu krajem 2005. godine, za rad s eksperimentalnim paketom za praćenje zrake (razvijenim od strane spomenute grupe) [6]. Rezultati su veoma bliski očekivanima: dvojezgreni sustav je ostvarivao otprilike dvije slike u sekundi, četverojezgreni 3.8, a 16-jezgreni 15.2 (taj je sustav emuliran povezivanjem četiriju četverojezgrenih procesora). Budući da su četverojezgreni procesori već danas široko dostupni, nije nezamislivo da će za nekoliko godina softversko iscrtavanje u stvarnom vremenu postupkom praćenja zrake biti sasvim izvedivo na prosječnim računalima. S druge strane, grafički procesori već danas posjeduju nekoliko stotina procesnih jedinica (ATI Radeon HD3870 X2 ih posjeduje ukupno 640), pa jednostavnom računicom dobivamo da bi praćenje zrake implementirano na grafičkom podsustavu bilo barem 30 do 50 puta brže nego na glavnom procesoru. Dakako, postavlja se pitanje kako izvesti algoritam praćenja zrake na sklopovlju dizajniranom za rasterizaciju. No, grafički procesori već neko vrijeme nisu ograničeni isključivo za obradu grafike. Razvojem jezika za sjenčanje visoke razine (sintakse usporedive sa C jezikom) za upravljanje programirljivim cjevovodom grafičkih procesora, oni postaju gotovo autonomni podsustav opće namjene, što je dovelo do stvaranja posebnog termina „općenamjenski grafički procesor“ (eng. *General Purpose Graphics Processing Unit*, GPGPU). Tvrtka nVidia, jedan od vodećih razvijatelja grafičkih procesora, je otišla korak dalje, te predstavila prevodilac za programski jezik C, koji generira instrukcije izvršive na nVidijinim grafičkim procesorima. Ova se tehnologija naziva *Compute Unified Device Architecture* (CUDA), te iako postoje neka ograničenja (rekurzivne funkcije nisu podržane), većina bitnih mogućnosti koje pruža glavni procesor, poput slobodnog dodjeljivanja memorije, jest podržana. To znači da je teoretski moguće napisati višedretveni algoritam praćenja zrake u C programskom jeziku, te ga prevesti za izvođenje na grafičkom procesoru. Također, i tvrtka ATI posjeduje sličnu tehnologiju koju nazivaju *Close To Metal* (CTM), koja je dostupna na ATI grafičkim karticama. Budući da je riječ o prilično novim tehnologijama čiji je razvoj tek započeo, konkretnih primjera njihove upotrebe još nema, no njihov je potencijal nemjerljiv. Osim klasičnog postupka praćenja zrake, moguće bi bilo implementirati i neki napredniji dodatak, kao što je to preslikavanje fotona. Konkretno, implementacije takvog algoritama su već ostvarene na grafičkim procesorima, koristeći postojeće jezike za sjenčanje [9]. Algoritam isijavanja bi imao tek djelomičnu korist od paralelne arhitekture grafičkih kartica, budući da se njegove iteracije ionako moraju izvršavati sekvencijski, a i unutar pojedine iteracije, vrijednost zračenja pojedinog poligona ovisi o zračenjima svih ostalih poligona. Jedina faza algoritma pogodna za paralelizaciju jest određivanje vidljivosti okolnih poligona za dani poligon.

Drugi značajan smjer razvoja interaktivnog globalnog osvjetljenja su postupci koji se i dalje baziraju na rasterizaciji, no proširuju opisane lokalne modele osvjetljenja tako da zamjenjuju određene aproksimacije fizikalno točnijim proračunima, približavajući se tako pravim globalnim modelima osvjetljenja. Ovo područje računalne grafike se intenzivno razvija, tako da se u ovoj skupini razlikuje veliko mnoštvo algoritama. U nastavku će biti

opisana dva karakteristična postupka u svom osnovnom obliku, a riječ je o zaklanjanju ambijenta (eng. *ambience occlusion*), te predizračunatom prijenosu zračenja (eng. *precomputed radiance transfer*).

5. Zaklanjanje ambijenta

Prisjetimo se, kod lokalnih modela osvjetljenja simulira se samo direktno osvjetljenje, dok se indirektno osvjetljenje i svi popratni učinci poput mekih sjena i pretapanja boja zanemaruju. Da scena ne bi bila premračna, obično se dodaje gruba aproksimacija indirektnog osvjetljenja, u obliku konstantnog člana jednadžbe, a taj se član naziva ambijentalna komponenta. Iako ambijentalno osvjetljenje doista jest gotovo homogeno, s tek blagim prijelazima u intenzitetu, ljudsko oko je osobito osjetljivo i na vrlo suptilne svjetlosne učinke. Tradicionalne metode simulacije globalnog osvjetljenja, opisane u prethodnom poglavlju, dobro rješavaju ovaj problem ali dostupna računalna moć tek sada postaje dostatna za početke njihove primjene u interaktivnoj računalnoj grafici. Iz tog razloga, razvijaju se različiti hibridni algoritmi, koji se baziraju na postojećim lokalnim modelima osvjetljenja i rasterizacijskim tehnikama iscrtavanja, ali uvode elemente kojima nastoje bolje aproksimirati globalne učinke osvjetljenja. Jedan od takvih algoritama jest zaklanjanje ambijenta. Naime, primijetimo da je kod lokalnih modela osvjetljenja, upravo ambijentalna komponenta razlog njihove neuvjerljivosti. Prema tome, ideja je da umjesto osvjetljavanja svakog vrha prikazanih poligona nekom konstantnom vrijednošću, koristimo neku varijabilnu vrijednost, tako da pojedini vrhovi budu tamniji, a pojedini svjetliji. Dakako, htjeli bismo postići da ta vrijednost ovisi o stupnju zaklonjenosti pojedinog vrha okolnom geometrijom, dakle da zaklonjeni vrhovi budu tamniji, a nezaklonjeni svjetliji. Ovakvo postavljanje problema dovodi do algoritma isijavanja, koji jest fizikalno točan, ali ne pruža zadovoljavajuće performanse. Algoritam zaklanjanja ambijenta, makar konceptualno vrlo sličan postupku isijavanja, pokazuje mnogo bolje performanse jer koristi fizikalno ne sasvim točan, ali mnogo efikasniji postupak. Pogledajmo detaljnije kako teče sam algoritam: na početku, stvara se pomoćna struktura na temelju geometrijskih podataka scene (popisa vrhova i poligona koje tvore). Ta pomoćna struktura sadrži takozvane elemente površine (eng. *surfel*, skraćeno od *surface element*). Element površine jest orijentirani krug, definiran svojom točkom središta, radijusom (oplošjem), te normalom.



Slika 12. Lijevo se nalazi mreža poligona, a desno odgovarajuća struktura elemenata površine.

Prednja strana elementa površine se definira kao ona strana na kojoj se nalazi normala. Ova definicija je bitna, jer prema dogovoru, svjetlost se emitira i reflektira sa prednje strane elementa površine, dok se sjene bacaju sa stražnje strane. Za svaki pojedini vrh poligona koji sačinjavaju scenu, stvara se novi element površine, i to tako da je središte tog elementa jednako poziciji njemu pripadnog vrha. Budući da geometrijski podaci obično uključuju i normale na vrhove, te se normale jednostavno mogu preuzeti za elemente površine, a u suprotnom slučaju se lako izračunaju iz poznatih vrhova poligona, koristeći sljedeću formulu:

$$A = (y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1)$$
$$B = -(x_2 - x_1)(z_3 - z_1) + (z_2 - z_1)(x_3 - x_1)$$
$$C = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$$

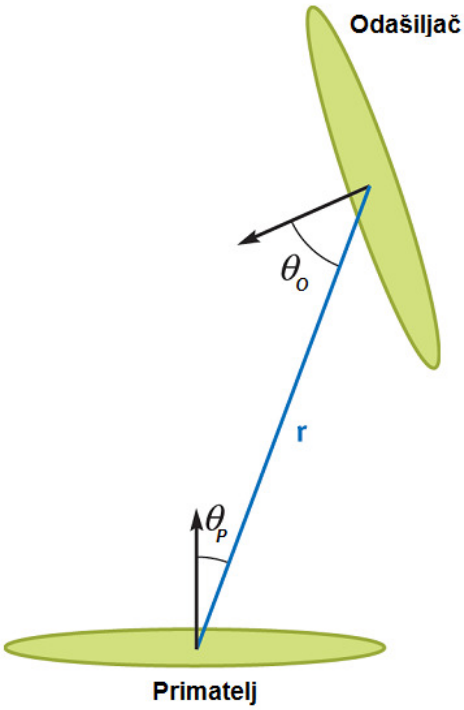
gdje su A, B i C elementi vektora normale. Oplošje pojedinog elementa površine se računa kao zbroj trećina oplošja trokuta koji dijele promatrani vrh. Budući da su poznati svi vrhovi tih trokuta, lako se dobiju duljine njihovih bridova, pa se oplošje pojedinog trokuta može dobiti Heronovom formulom:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

gdje je *s* poluopseg trokuta:

$$s = \frac{a+b+c}{2}$$

Jednom kad je pomoćna struktura izgrađena i popunjena svim podacima, mogu se izračunati faktori vidljivosti. Za razliku od fizikalne definicije kakva se koristi u algoritmu isijavanja, ovdje ćemo koristiti malo drugačiju, aproksimativnu definiciju. Zamislimo iznad elementa površine polukuglu. Ukoliko je taj element površine zaklonjen nekim drugim elementom, određeni dio zamišljene polukugle će biti u sjeni. Faktor vidljivosti definiramo kao postotak polukugle koji *nije* zaklonjen. Drugim riječima, ukoliko zbrojimo



Slika 13. Odnos elemenata površine. Vektor *r* povezuje središta elemenata, a kutevi θ_p i θ_o su definirani s obzirom na vektor *r*.

doprinose zaklonjenosti svih elemenata površine na trenutno promatrani element, onda je faktor vidljivosti tog elementa jednak jedan minus taj zbroj. Prema dogovoru, reći ćemo da je element koji biva zaklonjen *primatelj*, a element koji zaklanja *odašiljač*. Ukoliko zamislimo situaciju sa slike 13, možemo definirati vektor *r* koji povezuje središta odašiljača i primatelja. Kut između normale na element površine, te vektora *r* ćemo označiti sa θ. Sad možemo matematički formulirati definiciju faktora vidljivosti između dvaju elemenata površine:

$$F = 1 - \frac{r \cos\theta_O \max(1,\ 4\cos\theta_P)}{\sqrt{\frac{A_O}{\pi} + r^2}}$$

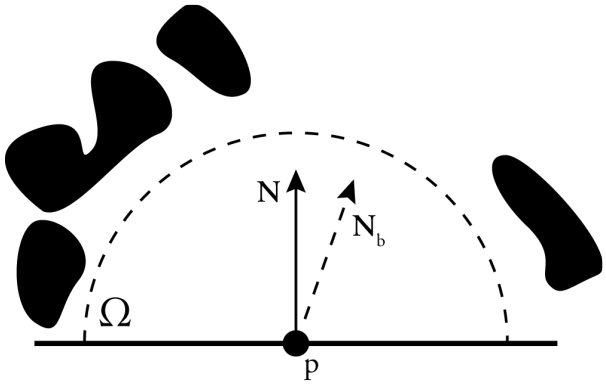
gdje je *r* prethodno definirani vektor, θ_o i θ_p su odgovarajući kutevi odašiljača i primatelja, a A_o je oplošje odašiljača. Funkcija max vraća veći od dvaju argumenata, a dodana je da se ignoriraju odašiljači koji ne leže iznad primatelja (sjetimo se, samo stražnje strane elemenata površine bacaju sjene). Prema slici 13, jasno se vidi da ukoliko bi odašiljač bio ispod primatelja, kut θ_p bi bio veći od PI/2 (ili manji od –PI/2), što znači da bi kosinus tog kuta bio negativan, pa bi funkcija max vratila 1. Dakako, u slučaju da se scena sastoji od više od dvaju elemenata površine, gornji se član izračuna za svaki element u odnosu na promatrani, te se njihov zbroj oduzme od jedinice:

$$F = 1 - \sum_{surfel} \frac{r \cos\theta_O \max(1,\ 4\cos\theta_P)}{\sqrt{\frac{A_O}{\pi} + r^2}}$$

Međutim, gore opisanim postupkom ćemo dolaziti u situacije da promatranom elementu površine pribrojimo doprinos zaklanjanja od nekog elementa koji je i sam potpuno ili djelomično zaklonjen. To znači da će promatrani element biti previše zatamnjen, budući da element koji je i sam u sjeni, ne može bacati novu sjenu. Jedino rješenje jest provedba algoritma u dva prolaza, s time da se u drugom prolazu desni član izraza pomnoži s prethodno izračunatim faktorom vidljivosti tog elementa površine. Drugim riječima, u drugom prolazu se doprinos zatamnjenja odašiljača *smanjuje*, tako da se modulira faktorom vidljivosti samog tog odašiljača. Međutim, što ako je neki element površine u trostrukoj sjeni? Tada će biti potreban treći prolaz, a možda i četvrti, peti itd. Međutim, u velikoj većini praktičnih primjena se pokazalo da su dva prolaza sasvim dostatna za vrlo kvalitetne rezultate [10], kao što to ilustrira sljedeća slika:

Slika 14. Prve tri slike su dobivene postupkom zaklanjanja ambijenta, i to redom s jednim, dva i tri prolaza. Četvrta slika je dobivena postupkom praćenja zrake.

Ovdje još valja napomenuti da se u ovom koraku algoritma obično računa i takozvana savinuta normala (eng. *bent normal*). Savinuta normala je vektor koji dijeli početnu točku sa pravom normalom, ali se ne nalazi pod pravim kutom nad površinom, već pokazuje u smjeru najmanje zaklonjenosti (slika 15). U postupku računanja direktnog osvjetljenja pomoću nekog lokalnog modela, tada se upotrebljava ta savinuta normala umjesto prave, čime se mogu dobiti još realističniji rezultati.



Slika 15. Na slici je prava normala označena s N, dok je N_b savinuta normala.

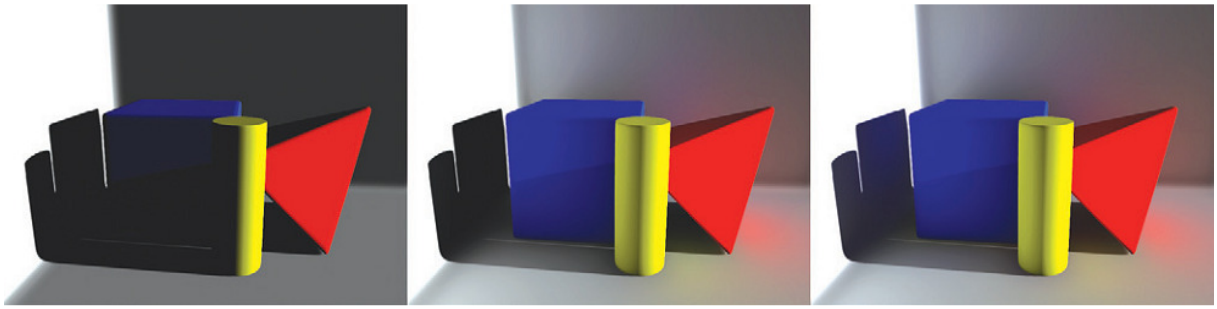
Algoritam zaklanjanja ambijenta u gore opisanom obliku pokazuje kvadratnu složenost, budući da je za svaki element površine (dakle, za svaki vrh poligona) potrebno obraditi sve ostale elemente površine. Međutim, postupak se lako može svesti na $n \log n$ složenost. Naime, algoritam u svome radu ne koristi direktno geometrijske podatke, već posebno generiranu strukturu elemenata površine. Ti se elementi mogu stopiti u veće elemente, a ti veći elementi se mogu stopiti u još veće elemente, itd. Samo „stapanje“ elemenata se vrši tako da središte elementa-roditelja postane aritmetička sredina središta elemenata-djece. Isto tako, normala roditelja se računa kao aritmetička sredina normala djece, dok je oplošje roditelja jednaka zbroju oplošja djece. Dakle, moguće je složiti hijerarhiju elemenata površine, od najgrubljeg prikaza do najfinijeg, što će u memoriji računala biti prikazano kao stablo. Korijeni tog stabla će biti najveći i najgrublji elementi površine, dok će listovi biti najmanji i najfiniji elementi. Pojedini čvorovi će biti negdje između ta dva ekstrema. Ovo si možemo predstaviti kao skup slojeva, takvih da se u najvišem sloju nalazi najmanji broj elemenata, u sljedećem sloju malo veći broj elemenata, u sljedećem još veći, i tako do zadnjeg sloja koji posjeduje najveći broj elemenata. Algoritam tada započinje rad koristeći prvi sloj. Tek ukoliko su dva elementa površine dovoljno blizu jedan drugome, algoritam će prijeći u niži, detaljniji sloj. Rezultat jest taj, da će udaljeni predmeti imati mutne i neprecizne sjene, dok će bliži predmeti pokazivati mnogo detaljnije sjene. Ovaj postupak je opravdan, budući da su ti detalji i blagi prijelazi u sjenama ionako slabo zamjetni osim na vrlo malenim udaljenostima, dok su na većim udaljenostima potpuno nezamjetni.

Upravo opisani postupci sačinjavaju prvu i glavnu fazu ovog algoritma, nakon koje je za svaki vrh objekata koji tvore scenu definirana zasebna ambijentalna komponenta osvjetljenja, umjesto konstantne vrijednosti za čitavu scenu kakvu standardno koriste lokalni modeli osvjetljenja. Međutim, ta je ambijentalna komponenta dobivena isključivo promatranjem zaklonjenosti pojedinih vrhova (od čega potječe i naziv samog algoritma), no ne i indirektnog osvjetljenja koje nastaje refleksijom svjetla između samih objekata. Drugim riječima, dobili smo meke sjene, ali ne i učinke poput pretapanja boja. Ipak, pokazuje se da se algoritam može relativno jednostavno proširiti [10], tako da uključuje i zadovoljavajuću aproksimaciju indirektnog osvjetljenja. Ta se aproksimacija bazira na prijenosu zračenja između pojedinih elemenata površine. Naime, ako se prisjetimo

algoritma isijavanja, računali smo faktore vidljivosti između parova poligona, te koristili te faktore za izračun prijenosa zračenja. Budući da i u algoritmu zaklanjanja ambijenta također koristimo vrlo sličan koncept, moguće je proširiti algoritam tako da dodatno izračunava i prijenos zračenja. Ovo se postiže tako da za svako svjetlo na sceni, obradimo elemente površine (koristeći hijerarhiju elemenata za ubrzanje). Ta obrada se sastoji od toga da se za prednju stranu svakog direktno osvjetljenog elementa površine izračuna maksimalan iznos reflektiranog zračenja s obzirom na njemu susjedne elemente. Pritom možemo koristiti sljedeću aproksimacijsku formulu:

$$B = \frac{A_O \cos\theta_O \cos\theta_P}{\pi r^2 + A_O}$$

gdje je A_o oplošje odašiljača, r vektor između središta odašiljača i primatelja, a θ_o i θ_p kutevi vektora r s obzirom na normale odašiljača i primatelja. Dakako, maksimalan prijenos isijavanja u većini slučajeva neće biti ostvaren, budući da će velik broj vrhova biti barem djelomično zaklonjen. Zbog toga je dobiveni iznos zračenja potrebno modulirati s prethodno izračunatim faktorom vidljivosti. Jednom kad je izlazno isijavanje izračunato, potrebno ga je pomnožiti s bojom odašiljača, te pridodati primatelju. Nakon ovog koraka, može se primijetiti učinak pretapanja boja. Međutim, ovdje se javlja sličan problem kao i kod klasičnog algoritma isijavanja. Naime, samo jednim prolazom, riješili smo problem jednostruko odbijenog indirektnog osvjetljenja. Dodatnim prolazima možemo propagirati zračenje i kroz višestruka odbijanja, no već su i dva prolaza dostatna da globalni učinci osvjetljenja postanu zamjetljivi. Budući da je ovaj algoritam namjenjen za korištenje u interaktivnoj računalnoj grafici, indirektno osvjetljenje sačinjeno od jednostrukih i dvostrukih difuznih interrefleksija predstavlja sasvim prihvatljiv kompromis između kvalitete i performansi, i značajno je unaprijeđenje u odnosu na konstantno ambijentalno osvjetljenje. Dobar primjer pokazuje donja slika. Sasvim lijevo vidimo scenu osvjetljenu standardnim, lokalnim modelom osvjetljenja. U sredini je rezultat proširenog algoritma zaklanjanja ambijenta s jednim prolazom, dok je krajnje desno rezulat dvaju prolaza tog algoritma.



Slika 16. Usporedba direktnog, jednostruko te dvostruko indirektnog osvjetljenja.

6. Predizračunati prijenos zračenja

U prošlom poglavlju predstavljen je efikasan algoritam temeljen na postupku isijavanja, gdje je efikasnost postignuta korištenjem aproksimacijskih, umjesto fizikalno točnih jednadžbi, uvođenjem kompromisa između broja koraka algoritma i kvalitete rezultata, te razvojem posebnih, dobro osmišljenih struktura podataka. Ipak, premda je algoritam dovoljno brz za upotrebu u interaktivnoj grafici na modernim računalima, čak i sa svim navedenim optimizacijskim mjerama, količina izračunavanja koju je potrebno izvršiti za svaku pojedinu generiranu sliku jest poprilična. Ukoliko je generiranje slika jedini zadatak, onda je takvo stanje stvari i prihvatljivo, no, većina interaktivnih aplikacija čija je svrha predočavanje virtualnih okruženja, obično sadrže i razne druge komponente, nevezane direktno uz samu grafiku. Odličan primjer takvih aplikacija su igre, koje predstavljaju jednu od vodećih sila razvoja računalne grafike, a osim same grafike uključuju i raznovrsne druge discipline i tehnologije poput umjetne inteligencije, fizikalnih simulacija, baza podataka, kompleksnih server-klijent arhitektura itd. Također, i raznovrsne specijalizirane poslovne aplikacije osim samog grafičkog prikaza, obično uključuju i dodatnu, specifičnu funkcionalnost vezanu uz njihovu primjenu. Sva ta dodatna funkcionalnost i sama značajno opterećuje računalni sustav, što znači da treba uzeti u obzir da će u praktičnim primjenama za sam postupak iscrtavanja biti dostupan tek dio ukupnih računalnih resursa. Ideja koja se prirodno nameće jest da pokušamo smanjiti količinu posla koji je potrebno obaviti prilikom samog iscrtavanja, tako da što je više moguće posla obavimo prije iscrtavanja, u predfazi, koja se može obaviti odvojeno od glavnog programa. Takva ideja nije nova, te se koristi u interaktivnoj računalnoj grafici već barem desetljeće, u obliku takozvanih mapa svjetlosti (eng. *lightmap*). Princip je vrlo jednostavan – budući su algoritmi poput isijavanja prespori za upotrebu u interaktivnim aplikacijama, takav se algoritam jednostavno izvrši prije pokretanja same aplikacije, te se rezultat pohrani u obliku običnih tekstura, nazvanih mape svjetlosti. Pojedina mapa svjetlosti se sastoji od isključivo sivih tonova, gdje tami dijelovi odgovaraju sjenama, a svijetli osvjetljenim područjima. Očito, ovakvim se pristupom mogu dobro simulirati meke sjene i fini prijelazi u ambijentalnom osvjetljenju, za čiji izračun može poslužiti bilo koji od globalnih algoritama osvjetljenja. Sve što glavna aplikacija tada mora učiniti, jest učitati te dodatne teksture i priložiti ih sceni, na isti način kao i sve obične teksture. Gubitak performansi je u tom slučaju gotovo nezamjetan, no prisutna su bitna ograničenja. Naime, budući da zasjenjenost pojedinih područja scene ovisi o geometrijskom odnosu objekata koji sačinjavaju danu scenu, ti geometrijski odnosi moraju biti poznati prije pokretanja glavne aplikacije. Isto tako, ukoliko bi došlo do geometrijskih promjena scene (deformacija ili pomicanje objekata uslijed animacije), predizračunate mape svjetlosti više ne bi vrijedile, pa bi došlo do vidljivih i očitih artefakata. Osim toga, mape svjetlosti ovise i o položaju i karakteristikama izvora svjetlosti koji obasjavaju scenu, tako da nije moguće imati niti dinamička svjetla. Algoritam predizračunatog prijenosa zračenja, koji će biti opisan u ovom poglavlju, jest konceptualno istovjetan mapama svjetlosti, no koristi matematički mnogo naprednije metode, te stoga pruža još kvalitetnije rezultate. Iz tog razloga, bit će ostvariva i dinamička svjetla, no nažalost, algoritam će i dalje biti ograničen na isključivo geometrijski statičke scene.

Međutim, prije opisa samog algoritma, prisjetimo se na trenutak jednadžbe iscrtavanja. Kako je već rečeno, upravo integralni član te jednadžbe predstavlja suštinski problem kojeg računalna grafika pokušava riješiti. No, može se postaviti naizgled naivno pitanje – budući da su već razvijeni veoma moćni matematički paketi za simboličke izračune kao što su Wolfram Mathematica ili MATLAB, koji su sposobni riješiti gotovo svaku zamislivu

integralnu jednadžbu, zašto se uopće razvijaju razni algoritmi iscrtavanja? Zašto ne bismo jednostavno upotrijebili neki od postojećih modela simboličkog izračunavanja za analitičko rješavanje jednadžbe iscrtavanja? Upravo u „simboličkom računanju“ leži problem! Naime, da bi se neka jednadžba mogla simbolički riješiti, mora biti poznat njen oblik i simbolički zapis. U našem slučaju, to nije moguće. Podintegralna jednadžba je u našem slučaju definirana karakteristikama scene koju želimo iscrtati, a budući da scene mogu biti veoma raznolike, s vrlo kompleksnim međuodnosima objekata, eksplicitan zapis funkcije osvjetljenja će općenito biti prekompleksan za bilo kakvu praktičnu primjenu. Iz tog razloga, općeniti postupak pronalaženja eksplicitnog, simboličkog oblika funkcije osvjetljenja ne postoji. Ipak, taj se postupak može osmisliti i provesti za neke posebne slučajeve, primjerice ako pretpostavimo da je ambijentalno osvjetljenje savršeno homogeno u svakoj točki prostora. U tom posebnom slučaju je problem pronalaženja simboličkog zapisa funkcije osvjetljenja trivijalan, budući da je riječ o tek jednoj konstanti. Upravo je ova pretpostavka temeljna ideja svih lokalnih modela svjetlosti, dok globalni algoritmi predstavljaju raznovrsne pristupe numeričkoj aproksimaciji pravog, analitičkog rješenja jednadžbe iscrtavanja. Svi su ti pristupi razvijeni dobrim poznavanjem domene računalne grafike, i njoj srodnih fizikalnih domena. Međutim, danas su u matematici poznate različite općenite metode numeričkog rješavanja integralnih i diferencijalnih jednadžbi, koje ne ovise o domeni ili kontekstu primjene. Dakako, neku (odgovarajuću) od tih općenitih metoda možemo pokušati upotrijebiti i za rješavanje jednadžbe iscrtavanja. Algoritam predizračunatog prijenosa zračenja se jednim dijelom temelji na upravo takvoj metodi, koja se naziva Montle Carlo integracija. Osnovna ideja te metode jest promatranje podintegralne funkcije kao crne kutije, dakle objekta nepoznatog oblika i interne strukture. Potrebno je poznavati tek skup točaka koje nastaju funkcijskim preslikavanjem iz odabrane domene, gdje je još poželjno da ta domena ima određena korisna svojstva. Sam postupak se izvodi iz teorije vjerojatnosti, stoga je potrebno definirati i objasniti određene temeljne pojmove koje ćemo koristiti. Osnovni koncept u teoriji vjerojatnosti jest slučajna varijabla. Ukoliko zamislimo pokus u kojem bacamo novčić, možemo reći da postoje dva osnovna, međusobno isključiva ishoda – palo je pismo, i pala je glava. Ti ishodi se nazivaju elementarni događaji, a svi postojeći elementarni događaji tvore prostor događaja. Slučajnu varijablu tada možemo definirati kao funkciju čija je domena prostor događaja, a koja preslikava elementarne događaje u realne brojeve. Na primjeru bacanja novčića, možemo definirati slučajnu varijablu X na sljedeći način:

$$X = \begin{cases} 0, & \text{ako je palo pismo} \\ 1, & \text{ako je pala glava} \end{cases}$$

Zatim možemo definirati i zakon razdiobe te slučajne varijable, koji govori kolika je vjerojatnost da varijabla poprimi neku specifičnu vrijednost. U gornjem primjeru, zakon razdiobe bi izgledao ovako:

$$X \sim \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

U gornjem retku se nalaze sve moguće vrijednosti koje varijabla može poprimiti, a u donjem retku su redom zapisane odgovarajuće vjerojatnosti tih vrijednosti. Ovako definirana slučajna varijabla se naziva diskretnom varijablom, budući da postoji konačan skup vrijednosti koje može poprimiti, koje odgovaraju konačnom broju događaja. Međutim, nama će biti zanimljivije kontinuirane varijable, koje se definiraju nad

beskonačnim prostorom događaja, i koje mogu poprimiti beskonačan broj vrijednosti. Kontinuirana slučajna varijabla se primjerice može definirati za pokus u kojem nasumično biramo broj iz intervala [0, 1]. Budući da za realne brojeve vrijedi da u svakom intervalu postoji beskonačan broj različitih brojeva, tako i slučajna varijabla može poprimiti beskonačan broj različitih vrijednosti. Slično kao što se za diskretne varijable definira zakon razdiobe, tako se za kontinuirane varijable definira funkcija razdiobe:

$$F(x) := P(\{X < x\})$$

Naime, budući da je domena beskonačna, općenito će nas zanimati samo vjerojatnost da je varijabla poprimila vrijednost manju od neke zadane, a ne točan iznos. Još jedan izrazito važan pojam vezan uz kontinuirane slučajne varijable jest funkcija gustoće, koja se definira kao derivacija funkcije razdiobe. Drugim riječima, vrijedi sljedeće:

$$F(x) = \int_{-\infty}^{x} g(t)dt$$

gdje je *g(t)* funkcija gustoće. Važno svojstvo funkcije gustoće jest da njen integral po cijelom skupu realnih brojeva mora biti jednak jedan. Također, vrijedi da za svaku slučajnu varijablu postoji neka vrijednost koju ta varijabla najčešće poprima, a ta se vrijednost naziva očekivanje varijable. Očekivanje možemo definirati na sljedeći način:

$$E(X) = \int_{-\infty}^{\infty} x\, g(x)\, dx$$

Budući da su i same slučajne varijable ništa drugo doli funkcije s određenim svojstvima, možemo proširiti gornju definiciju očekivanja na općenite funkcije:

$$E(f(x)) = \int_{-\infty}^{\infty} f(x)\, g(x)\, dx$$

No, očekivanje neke funkcije se može (približno) izračunati i na drugi način, na temelju Zakona velikih brojeva. Taj zakon kaže da ukoliko su poznate vrijednosti funkcije za velik broj različitih točaka, tada je očekivanje te funkcije približno jednako aritmetičkoj sredini svih poznatih uzoraka. Matematički, taj rezultat možemo zapisati ovako:

$$E(f(x)) \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

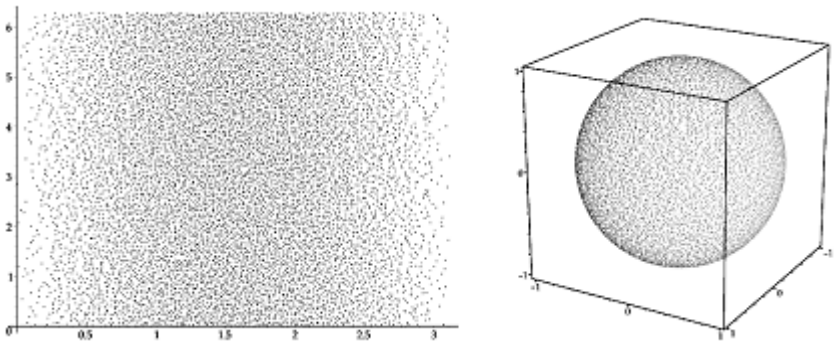
Bitna napomena jest da ovaj zakon vrijedi samo ukoliko su uzorci nasumično, uniformno distribuirani. Dakako, što je veći broj uzoraka, to će rezultat biti točniji. Prethodne dvije relacije sada možemo izjednačiti, te time dobivamo sljedeći rezultat:

$$\int_{-\infty}^{\infty} f(x)\, dx = \int_{-\infty}^{\infty} \frac{f(x)}{g(x)}\, g(x)\, dx \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)}$$

Kao što vidimo, integral neke proizvoljne funkcije *f(x)* možemo aproksimirati aritmetičkom sredinom slučajnih uzoraka te funkcije, skaliranih nekim težinskim faktorom

(koji odgovara vrijednosti funkcije gustoće). Upravo opisani postupak se naziva Monte Carlo integracija. Sad je očito kako se može integrirati funkcija čiji eksplicitan oblik nije poznat – korištenjem ove numeričke metode, dovoljno je poznavati tek određen skup točaka funkcije čiji integral tražimo. Da bismo mogli upotrijebiti ovu metodu za rješavanje jednadžbe iscrtavanja, potrebno je još samo odgovoriti na dva pitanja. Naime, budući da je jednadžba koju rješavamo definirana kao integral preko površine kugle, postavlja se pitanje kako uniformno raspodijeliti uzorke preko sfere. Također, moramo pronaći i težinske faktore, odnosno odgovarajuću funkciju gustoće. Prvi problem ćemo lako riješiti korištenjem takozvane kanonske uniformne varijable, koja poprima vrijedosti u intervalu [0, 1), i to tako da svaka vrijednost ima jednaku vjerojatnost pojavljivanja. Ukoliko uzmemo da kanonske varijable X i Y označavaju pravokutne koordinate točaka, možemo ih projicirati na površinu sfere pomoću sljedeće transformacije:

$$\left(2\cos^{-1}\left(\sqrt{1-X}\right),\ 2\pi Y\right)\rightarrow(\theta,\varphi)$$



Slika 17. Deset tisuća uniformnih uzoraka. Lijevo je prikaz u pravokutnom koordinatnom sustavu, a desno projekcija u sferne koordinate.

Budući da uzorke biramo uniformno, funkcija razdiobe je takva da je njen iznos proporcionalan duljini intervala u kojem se uzorak nalazi. Naime, pošto je funkcija razdiobe definirana kao $P(X < x)$, što je x veći, to je širi interval u kojem se X može nalaziti, a to znači i da je veća vjerojatnost da će se X nalaziti u tom intervalu. Drugim riječima, u našem slučaju funkcija razdiobe jest linearna, a ukoliko se prisjetimo da je funkcija gustoće definirana kao derivacija funkcije razdiobe, lako zaključujemo da funkcija gustoće mora biti tek jedna konstanta. Iz uvjeta da integral funkcije gustoće mora biti jedan, lako dobivamo da je njen iznos jednak recipročnoj vrijednosti oplošja kugle, a pošto radimo s jediničnom kuglom, ta je vrijednost jednaka $1/4\pi$. Sad napokon možemo napisati konačan izraz za numeričku aproksimaciju integrala, koji možemo direktno upotrijebiti za rješavanje jednadžbe iscrtavanja:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,dxdy \approx \frac{4\pi}{N}\sum_{i=1}^{N} f(x_i,y_i)$$

Doista, naprednim matematičkim aparatom uspjeli smo prevesti jednadžbu iscrtavanja iz oblika koji je nerješiv direktno na računalu, u oblik koji to jest, korištenjem isključivo jednostavnih operacija množenja i zbrajanja. Budući da direktno rješavamo jednadžbu iscrtavanja, sasvim sigurno ćemo obuhvatiti apsolutno sve optičke efekte koje svjetlost proizvodi svojom propagacijom (dakako, ne uzimajući u obzir pojave zbog valne prirode svjetlosti, koje jednadžba iscrtavanja ne opisuje). Međutim, ako malo bolje pogledamo gornji izraz, i sjetimo se da ćemo ga morati izračunavati za svaki vrh svih poligona koji sačinjavaju scenu, vidimo da je tu uključeno doista mnogo računanja. Ukoliko

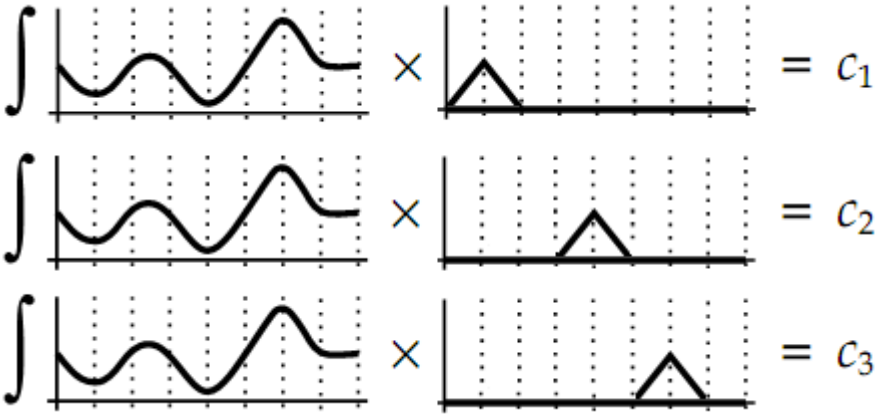
pretpostavimo da se na sceni nalazi stotinu tisuća vrhova (što je uobičajeno za današnje standarde), te da koristimo deset tisuća slučajnih uzoraka, dolazimo do brojke od jedne milijarde operacija množenja i zbrajanja po slici, za cjelokupnu scenu. Dakako, sama ta evaluacija integrala se može provesti tek nakon što su poznati uzorci funkcije osvjetljenja, a da bi oni bili izračunati, potrebno je odrediti karakteristike scene, dakle geometrijske međuodnose objekata na sceni. Algoritam koji efikasno rješava taj problem jest praćenje zrake, koji je i sam po sebi još uvijek vrlo zahtjevan za većinu osobnih računala. Doduše, to određivanje međuodnosa objekata na sceni se može provesti kao predprocesiranje scene, ukoliko koristimo statičku geometriju, takvu da se objekti ne pomiču niti deformiraju za cjelokupno trajanje simulacije. Dakle, u predfazi algoritma iscrtavanja, a po potrebi čak i nekom aplikacijom odvojenom od glavnog programa, može se provesti algoritam praćenja zrake i na temelju toga generirati potrebni uzorci. No, još uvijek ostaje problem evaluacije same jednadžbe iscrtavanja, što je postupak koji se za detaljne scene može sastojati, kako smo prethodno utvrdili, i od milijarde operacija! Očito, opisani postupak je praktički neupotrebljiv, no razlog tomu je što je algoritam u predstavljenom obliku još uvijek nepotpun. U ovom trenutku možemo uvesti još jednu matematičku tehniku koja se često koristi za rješavanje raznih fizikalnih i drugih problema, a koja će na vrlo elegantan način drastično reducirati broj potrebnih operacija za vrijeme samog iscrtavanja. Riječ je o rastavljanju proizvoljnih funkcija u takozvane bazne funkcije, za što se obično odabiru neki jednostavni polinomi ili trigonometrijske funkcije. Ideja je zatim sljedeća: odgovarajućim skaliranjem i zbrajanjem nekog broja tih baznih funkcija se dobiva bolja ili lošija aproksimacija početne funkcije. Ukoliko označimo početnu funkciju sa $f(x)$, i-tu baznu funkciju sa $B_i(x)$, a koeficijent za skaliranje i-te bazne funkcije sa c_i, onda prethodnu rečenicu možemo matematički zapisati ovako:

$$f(x) \approx \sum c_i B_i(x)$$

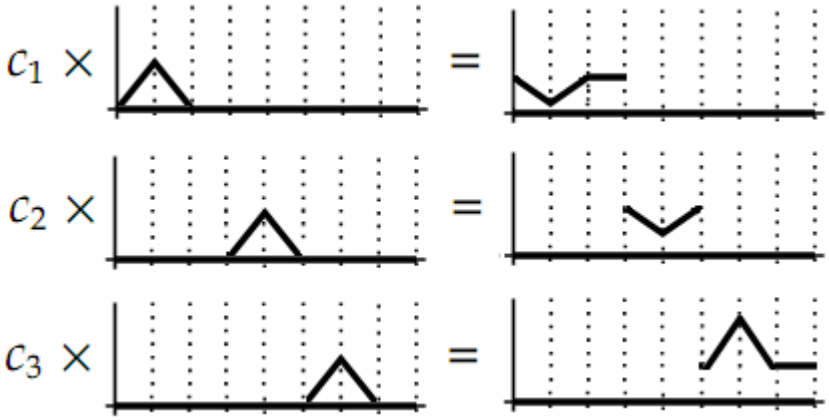
Jedini problem koji moramo riješiti jednom kad smo odabrali bazne funkcije, jest odrediti odgovarajuće koeficijente za danu početnu funkciju, koji govore koliko je ta funkcija „slična“ pojedinim baznim funkcijama. Postupak njihovog izračunavanja se svodi na integriranje umnoška početne funkcije i neke bazne funkcije, po cijeloj domeni početne funkcije, dakle:

$$c_i = \int_{\mathcal{D}_f} f(x) B_i(x) dx$$

Možda je najjednostavnije predočiti si ovaj postupak grafički. Zamislimo da imamo neku krivuljnu početnu funkciju, i tri linearne bazne funkcije. Postupak određivanja koeficijenata bi tada izgledao ovako:

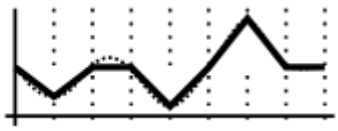


Jednom kad imamo izračunate koeficijente, možemo ih pomnožiti s baznim funkcijama, i time skalirati bazne funkcije na potrebnu amplitudu:



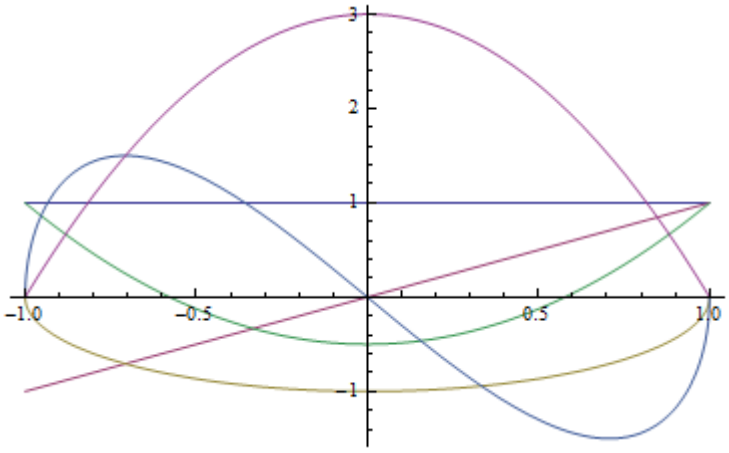
Nakon toga, skalirane bazne funkcije trebamo samo zbrojiti, i dobit ćemo aproksimaciju početne funkcije:

$$\sum c_i B_i =$$



Sam postupak rastavljanja proizvoljne funkcije u sumu baznih funkcija se naziva projekcija, a pojedini članovi te sume se ponekad nazivaju harmonicima. Odabir vrste projekcije (odnosno, oblika baznih funkcija) ovisi o kontekstu primjene i svojstvima funkcije koju želimo aproksimirati. U našem slučaju, budući da je funkcija osvjetljenja definirana preko sfere, odgovarat će nam projekcija u takozvane sferne harmonike. Ova vrsta projekcije se zasniva na određenom obliku polinoma, koji se nazivaju asocirani Legendre polinomi. Ti polinomi se definiraju pomoću dvaju parametara, l i m, te je njihova uobičajena oznaka $P_l^m(x)$. Parametar l se naziva pojasom polinoma, i može biti bilo koji cijeli broj ili nula, dok m mora biti u intervalu [0, l]. Dakle, polinomi su podijeljeni u pojaseve, takve da svaki sljedeći pojas sadrži točno jedan polinom više od prethodnog. Domena asociranih Legendre polinoma su realni brojevi na intervalu [-1, 1]. Prvih nekoliko takvih polinoma izgleda ovako:

$$P_0^0(x) = 1$$
$$P_1^0(x) = x$$
$$P_1^1(x) = -\left(1-x^2\right)^{1/2}$$
$$P_2^0(x) = \frac{1}{2}\left(3x^2-1\right)$$
$$P_2^1(x) = -3x\left(1-x^2\right)^{1/2}$$
$$P_2^2(x) = 3\left(1-x^2\right)$$



Slika 18. Graf prvih pet asociranih Legendre polinoma.

Asocirani Legendre polinomi se definiraju kao rješenja istoimene diferencijalne jednadžbe, no postoje drugi načini kako se oni mogu generirati. Najjednostavniji način, koji je ujedno i najprimjereniji za implementaciju na računalu, jest korištenje rekurzivnih relacija. Osnovna takva relacija jest sljedeća:

$$(l-m)P_l^m = x(2l-1)P_{l-1}^m - (l+m-1)P_{l-2}^m \qquad (1)$$

Primjetimo da je parametar *m* konstantan, te da se mijenja isključivo parametar *l*. To znači da ovom relacijom možemo iz poznavanja dvaju polinoma s istim rednim brojem, ali iz dvaju susjednih pojasa, izračunati treći polinom, koji će se nalaziti u trećem pojasu po redu (počevši od pojasa prvog polinoma), i imati isti redni broj unutar svog pojasa kao i početna dva polinoma. No, osim ove temeljne relacije, može se izvesti i pomoćna, nešto jednostavnija relacija:

$$P_{m+1}^m = x(2m+1)P_m^m \qquad (2)$$

Ova relacija vrijedi samo u slučaju da poznajemo polinom čiji je redni broj unutar pojasa jednak rednom broju pojasa. Tada možemo jednostavno podignuti taj polinom u sljedeći pojas. Osim ovih rekurzivnih relacija, za asocirane Legendre polinome vrijedi i sljedeće zanimljivo svojstvo:

$$P_m^m = (-1)^m(2m-1)!!(1-x^2)^{m/2} \qquad (3)$$

Riječima, ovo svojstvo znači da se polinom čiji je redni broj unutar pojasa jednak rednom broju samog pojasa može dobiti pomoću direktne formule, bez potrebe za postepenim, iterativnim računanjem jednog po jednog polinoma. U gornjoj formuli se javlja operator !!, što je oznaka dvostrukog faktorijela, a koji je definiran na sljedeći način:

$$n!! \equiv \begin{cases} n\cdot(n-2)\cdot\ldots\cdot 5\cdot 3\cdot 1, & n>0 \text{ i paran} \\ n\cdot(n-2)\cdot\ldots\cdot 6\cdot 4\cdot 2, & n>0 \text{ i neparan} \\ 1, & n=-1,0 \end{cases}$$

Promatranjem gornjih triju relacija, lako možemo osmisliti efikasan način za generiranje asociranih Legendre polinoma proizvoljnih parametara *l* i *m*. Očito, najjednostavnije je

evaluirati posljednju relaciju, koja pruža direktno rješenje. Doduše, ona se sastoji od dvostrukog faktorijela koji je i sam rekurzivna relacija, no vremenska kompleksnost računanja faktorijela se može svesti na konstantu upotrebom tablice predizračunatih vrijednosti. Dakle, ideja je sljedeća: za zadane parametre l i m, prvo se izračuna intermedijarni polinom korištenjem relacije (3). Rezultirajući polinom će biti oblika P_m^m, te ukoliko je $l = m$, onda je to konačno rješenje. U suprotnom slučaju, budući da prema definiciji mora vrijediti $m \leq l$, zaključujemo da je m sigurno manji od l. To znači da jednom kad je određen polinom s traženim parametrom m, ukoliko je l različit od m, potrebno je samo podizati pojas dobivenog intermedijarnog polinoma dok se ne dosegne l-ti pojas. Ovo se može postići relacijama (2) i (1). Ukoliko je $l = m + 1$, dovoljna je samo relacija (2), koja odmah daje konačno rješenje. Ukoliko to nije slučaj, onda je još potrebno iskoristiti i relaciju (1). Naime, pomoću relacije (3) smo dobili prvi intermedijarni polinom, a pomoću relacije (2) drugi. No, taj drugi dobiveni polinom očito više nije oblika P_m^m, nego P_{m+1}^m, pa nam je preostala samo relacija (1). Tom relacijom iz P_m^m i P_{m+1}^m dobivamo P_{m+2}^m. Po potrebi, iz P_{m+1}^m i P_{m+2}^m možemo dobiti P_{m+3}^m, i tako dalje, dok ne dosegnemo zadani pojas. Ovime smo pokrili svo znanje o asociranim Legendre polinomima koje je potrebno za korištenje sfernih harmonika, pa je vrijeme da ih matematički definiramo. Uobičajena oznaka jest $Y_l^m(\theta,\varphi)$, gdje parametri l i m imaju isto značenje kao i kod asociranih Legendre polinoma. Jasno, budući da je riječ o funkciji definiranoj preko površine kugle, njeni su parametri dani u obliku sfernih koordinata (θ,φ). Sferni harmonici se općenito definiraju kao kompleksne funkcije, no nas će zanimati samo realni dio:

$$Y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}K_l^m \cos(m\varphi)P_l^m(\cos\theta), & m > 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0 \\ \sqrt{2}K_l^m \sin(-m\varphi)P_l^{-m}(\cos\theta), & m < 0 \end{cases}$$

gdje je $P_l^m(x)$ odgovarajući asocirani Legendre polinom, dok je K_l^m faktor za skaliranje koji se računa na sljedeći način:

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$$

Valja primjetiti da je za uvođenje sfernih harmonika potrebno malo proširiti prethodnu definiciju asociranih Legendre polinoma. Naime, rekli smo da mora vrijediti $l \leq 0$, $0 \leq m \leq$ l. Međutim, za funkcije sfernih harmonika, potrebno je proširiti dozvoljeni interval parametra m tako da uključuje i negativne vrijednosti, dakle da vrijedi $-l \leq m \leq l$. Za ilustraciju, funkcije sfernih harmonika prvih triju pojaseva izgledaju ovako u općem obliku:

$$Y_0^0(\theta,\varphi) = \frac{1}{2}\sqrt{\frac{1}{\pi}}$$

$$Y_1^{-1}(\theta,\varphi) = \frac{1}{2}\sqrt{\frac{3}{2\pi}}\,\sin\theta\, e^{-i\varphi}$$

$$Y_1^0(\theta,\varphi) = \frac{1}{2}\sqrt{\frac{3}{\pi}}\,\cos\theta$$

$$Y_1^1(\theta,\varphi) = \frac{-1}{2}\sqrt{\frac{3}{2\pi}}\,\sin\theta\, e^{i\varphi}$$

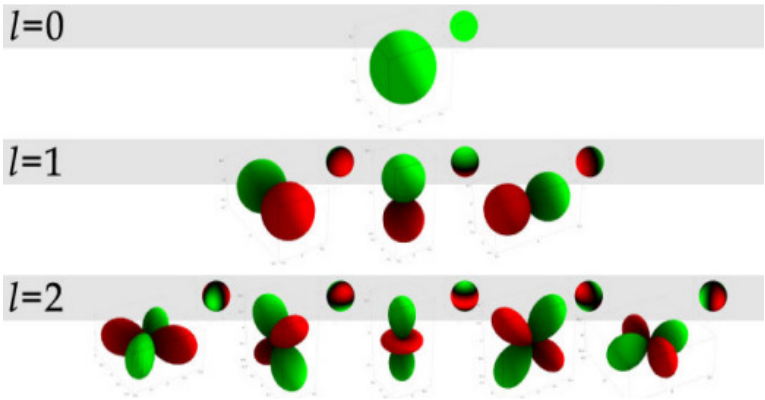
$$Y_2^{-2}(\theta,\varphi) = \frac{1}{4}\sqrt{\frac{15}{2\pi}}\,\sin^2\theta\, e^{-2i\varphi}$$

$$Y_2^{-1}(\theta,\varphi) = \frac{1}{2}\sqrt{\frac{15}{2\pi}}\,\sin\theta\,\cos\theta\, e^{-i\varphi}$$

$$Y_2^0(\theta,\varphi) = \frac{1}{4}\sqrt{\frac{5}{\pi}}\,(3\cos^2\theta - 1)$$

$$Y_2^1(\theta,\varphi) = \frac{-1}{2}\sqrt{\frac{15}{2\pi}}\,\sin\theta\,\cos\theta\, e^{i\varphi}$$

$$Y_2^2(\theta,\varphi) = \frac{1}{4}\sqrt{\frac{15}{2\pi}}\,\sin^2\theta\, e^{2i\varphi}$$



Slika 19. Grafovi realnih sfernih harmonika prvih triju pojaseva.

Kako je već rečeno, općeniti postupak za projekciju neke početne funkcije u bazne funkcije jest izračun odgovarajućih koeficijenata, a koji se računaju kao integral umnoška početne funkcije i baznih funkcija. U našem konkretnom primjeru, može se izvesti sljedeća formula za dobivanje pojedinih koeficijenata:

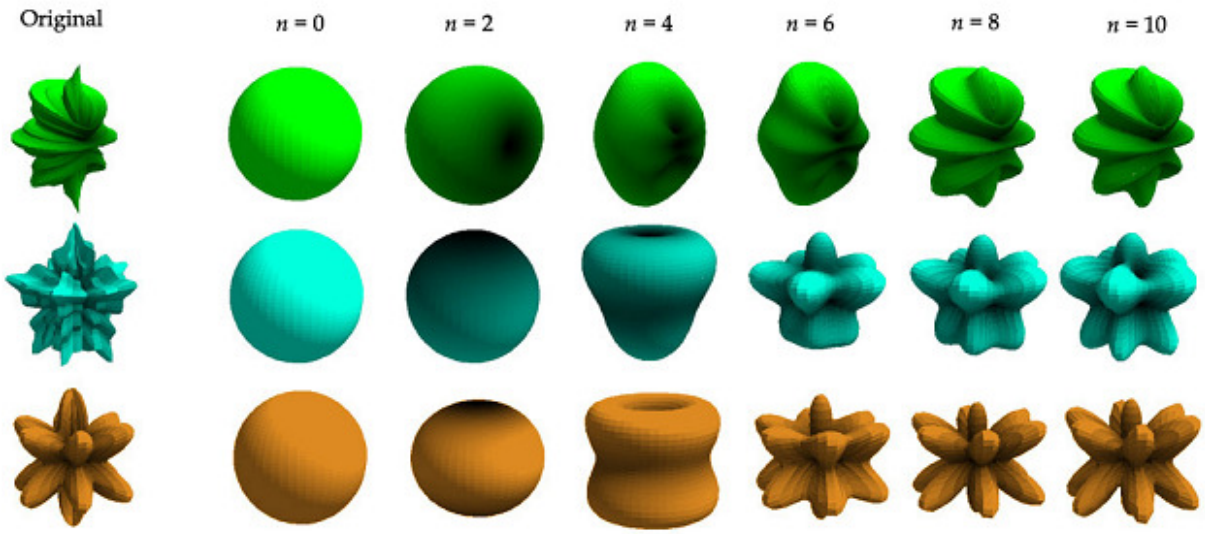
$$c_l^m = \int_0^{2\pi}\int_0^{\pi} f(\theta,\varphi)\, Y_l^m(\theta,\varphi)\sin\theta\, d\theta d\varphi$$

Međutim, funkciju koju ćemo htjeti projicirati – funkciju osvjetljenja – nije moguće, kako smo već utvrdili, zapisati u eksplicitnom, simboličkom obliku, već je moguće samo procijeniti njenu vrijednost u nekom skupu točaka, korištenjem algoritma praćenja zrake ili nekog sličnog. No, vidimo da smo upravo taj problem već riješili, uvođenjem Monte Carlo integracije! Ukoliko se prisjetimo prethodno dobivenog rezultata za numeričku aproksimaciju integrala,

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,dxdy \approx \frac{4\pi}{N}\sum_{i=1}^{N} f(x_i,y_i)$$

možemo povezati taj rezultat sa formulom za koeficijente projekcije, pa dobivamo sljedeće:

$$c_l^m \approx \frac{4\pi}{N}\sum_{i=1}^{N} f(\theta_i,\varphi_i) Y_l^m(\theta_i,\varphi_i)\sin\theta_i$$



Slika 20. Primjeri aproksimacija različitih funkcija sfernim harmonicima, s rastućim redom projekcije.

Prebacivanjem u pravokutne koordinate, dobivamo konačni oblik formule, koji je najprimjereniji za implementaciju na računalu:

$$c_l^m \approx \frac{4\pi}{N} \sum_{i=1}^{N} f(x_i, y_i) y_l^m(x_i, y_i)$$

gdje je y_l^m funkcija sfernog harmonika izražena u pravokutnim koordinatama. Rekonstrukcija početne funkcije se tada vrši jednostavnim sumiranjem umnožaka koeficijenata i sfernih harmonika. Ukoliko se dogovorimo da $\widetilde{f}(x)$ predstavlja oznaku projekcije funkcije $f(x)$ u sferne harmonike, onda se projekcija *n*-tog reda definira ovako:

$$f(x, y) \approx \widetilde{f}(x, y) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} c_l^m y_l^m(x, y)$$

U ovom trenutku možemo opisati neka bitna svojstva sfernih harmonika, koja ih čine idealnim izborom baznih funkcija za projekciju funkcije osvjetljenja. Osnovno svojstvo, koje predstavlja i glavni razlog upotrebe upravo sfernih harmonika, jest ortonormalnost. Za neki skup funkcija se kaže da je ortonormalan ukoliko za svaki par funkcija tog skupa vrijedi da je integral umnoška tih dviju funkcija jednak ili 0 ili 1. U slučaju da su funkcije istovjetne, rezultat je 1, a u suprotnom slučaju 0. Matematički, ovo možemo zapisati na sljedeći način:

$$\int_a^b f_n(x) f_m(x) dx = \begin{cases} 1, & n = m \\ 0, & \text{inače} \end{cases}$$

Bitnost ovog svojstva možemo uočiti ukoliko se prisjetimo jednadžbe iscrtavanja:

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

Radi jasnoće, njen integralni član možemo sažeto zapisati u sljedećem obliku:

$$\int_s L(s)t(s)ds$$

gdje je *L(s)* funkcija upadne svjetlosti, što odgovara izrazu $L_i(x,\overrightarrow{w'})$, a *t(s)* funkcija prijenosa, koja predstavlja sažet prikaz distribucije reflektivnosti zračenja i atenuaciju svjetlosti, dakle, odgovara izrazu $f_r(x,\overrightarrow{w'},\overrightarrow{w})(\overrightarrow{w'}\cdot\vec{n})$. Projekcijom funkcija *L(s)* i *t(s)* u sferne harmonike dobivamo sljedeće:

$$\int_s L(s)t(s)ds \approx \int_s \tilde{L}(s)\tilde{t}(s)ds$$

Ukoliko upotrijebimo definiciju projekcije, gornje podintegralne funkcije možemo ovako zapisati:

$$\tilde{L}(s)=\sum_{l=0}^{n-1}\sum_{m=-l}^{l} c\{L\}_l^m\, y_l^m(s)$$

$$\tilde{t}(s)=\sum_{l=0}^{n-1}\sum_{m=-l}^{l} c\{t\}_l^m\, y_l^m(s)$$

Oznake $c\{L\}_l^m$ i $c\{t\}_l^m$ označavaju redom koeficijente projekcija funkcija *L(s)* i *t(s)*. Ukoliko sada ove izraze uvrstimo u jednadžbu iscrtavanja, dobivamo sljedeće:

$$\int_s L(s)t(s)ds \approx \int_s \left(\sum_{l=0}^{n-1}\sum_{m=-l}^{l} c\{L\}_l^m\, y_l^m(s)\right)\left(\sum_{l=0}^{n-1}\sum_{m=-l}^{l} c\{t\}_l^m\, y_l^m(s)\right)ds$$

Raspišimo sad ove dvije dvostruke sume za prvih nekoliko članova:

$$\left(c\{L\}_0^0\, y_0^0(s)+c\{L\}_1^{-1}\, y_1^{-1}(s)+c\{L\}_1^0\, y_1^0(s)+c\{L\}_1^1\, y_1^1(s)+...\right)$$

$$\left(c\{t\}_0^0\, y_0^0(s)+c\{t\}_1^{-1}\, y_1^{-1}(s)+c\{t\}_1^0\, y_1^0(s)+c\{t\}_1^1\, y_1^1(s)+...\right)$$

Nakon množenja tih izraza, dobivamo:

$$c\{L\}_0^0 c\{t\}_0^0\, y_0^0(s)y_0^0(s)+c\{L\}_0^0 c\{t\}_1^{-1}\, y_1^0(s)y_1^{-1}(s)+c\{L\}_0^0 c\{t\}_1^0\, y_0^0(s)y_1^0(s)+c\{L\}_0^0 c\{t\}_1^1\, y_0^0(s)y_1^1(s)+...$$

Dakle, imamo sumu umnožaka koeficijenata i sfernih harmonika obaju projekcija, što možemo sažeto zapisati:

$$\int_s L(s)t(s)ds \approx \int_s \left(\sum_{l=0}^{n-1}\sum_{m=-l}^{l}\sum_{l'=0}^{n-1}\sum_{m=-l'}^{l'} c\{L\}_l^m c\{t\}_{l'}^{m'}\, y_l^m(s)y_{l'}^{m'}(s)\right)ds$$

Budući da su integracija i sumacija linearni operatori, možemo im promijeniti redosljed:

$$\int_s L(s)t(s)ds \approx \sum_{l=0}^{n-1}\sum_{m=-l}^{l}\sum_{l'=0}^{n-1}\sum_{m=-l'}^{l'}\left(c\{L\}_l^m\, c\{t\}_{l'}^{m'}\int_s y_l^m(s)y_{l'}^{m'}(s)ds\right)$$

Prema svojstvu ortonormalnosti, slijedi da će integral na desnoj strani jednadžbe uvijek biti jednak jedinici za $m = m'$, $l = l'$, i jednak nuli za sve druge slučajeve, pa možemo izbaciti unutarnje dvije sumacije i dobiti ovaj mnogo jednostavniji izraz:

$$\int_s L(s)t(s)ds \approx \sum_{l=0}^{n-1} \sum_{m=-l}^{l} c\{L\}_l^m \, c\{t\}_l^m$$

Ako bolje pogledamo što smo dobili, vidimo da smo uspjeli svesti integral preko površine sfere na jednostavno množenje i zbrajanje koeficijenata! Dakako, istu stvar smo postigli i direktnim numeričkim rješavanjem jednadžbe iscrtavanja, no postoji značajna razlika u efikasnosti. Kod direktnog rješavanja jednadžbe, za svaki vrh poligona koji sačinjavaju scenu potrebno je ponoviti jedno množenje i jedno zbrajanje onoliko puta koliko imamo uzoraka, što u praktičnim primjenama može biti i desetak tisuća. S druge strane, ukoliko provedemo projekciju u sferne harmonike, pokazat će se da je i 16 koeficijenata po vrhu sasvim dovoljno za kvalitetne rezultate, što znači da je dovoljno tek 16 operacija množenja i zbrajanja po vrhu. Drugim riječima, algoritam je sada doslovno postao tisuću puta brži!

Sferni harmonici posjeduju još jedno izrazito korisno svojstvo, a to je rotacijska invarijantnost. Naime, zamislimo da imamo dvije funkcije, *f* i *g*. Funkciju *f* projiciramo u sferne harmonike, a zatim izvršimo rotaciju za neki kut oko neke osi. S druge strane, funkciju *g* prvo rotiramo za isti taj kut oko iste te osi, a zatim projiciramo. Ukoliko su *f* i *g* istovjetne funkcije, onda će i rezultati oba postupka biti istovjetni. Drugim riječima, svejedno je da li prvo rotiramo funkciju pa ju projiciramo, ili prvo projiciramo pa rotiramo. Ovo odlično svojstvo će omogućiti dinamička svjetla, budući da ćemo funkciju upadnog svjetla, u gornjim jednadžbama označenu s *L(s)*, moći rotirati i nakon projekcije. Sama rotacija koeficijenata se jednostavno postiže matričnim množenjem, a jedini je problem konstruirati odgovarajuću matricu. Naime, dimenzije rotacijske matrice očito ovise o broju koeficijenata s kojima računamo, a budući da broj potrebnih koeficijenata ovisi o redu sferne projekcije (za projekciju *n*-tog reda potrebno je n^2 koeficijenata), i dimenzije rotacijske matrice također ovise o redu projekcije. To znači da je potrebno pronaći način za generiranje rotacijskih matrica danog reda. Može se pokazati da će za *n*-ti red projekcije, rotacijska matrica biti dimenzija $n^2 \times n^2$, te se može izgraditi pomoću rekurzivnog postupka. Pritom ćemo pretpostaviti da se rotacija vrši pomoću Eulerovih kuteva, dakle, kao niz od tri rotacije oko koordinatnih osi, pri čemu je potrebno odabrati redosljed komponenti rotacije. Najpovoljniji izbor redosljeda rotacija jest oko ZYZ osi. Naime, tada da su nam dovoljne samo dvije vrste rotacija, oko Z osi , i oko Y osi, pri čemu se rotacija oko Y osi može dodatno rastaviti na rotaciju oko X osi za 90 stupnjeva, zatim rotaciju oko Z osi, te rotaciju oko X osi za -90 stupnjeva. Dakle, izraz za konačnu matricu rotacije sfernih harmonika glasi:

$$R_{SH}(\alpha, \beta, \gamma) = Z_\gamma X_{-90} Z_\beta X_{90} Z_\alpha$$

Dakako, još moramo odrediti i same matrice X i Z. Jedan način kako se to može postići jest upotrebom istog principa koji omogućuje projekciju neke krivulje u bazne funkcije. Naime, prisjetimo se značenja koeficijenata baznih funkcija – oni preslikavaju bazne funkcije u segmente početne krivulje! Prema tome, možemo istu tehniku koristiti za preslikavanje ne-rotiranih koeficijenata u rotirane. Pojedini koeficijenti ove projekcije se

dobivaju na potpuno isti način koji vrijedi za sve projekcije, dakle integracijom umnoška početne i bazne funkcije, što u našem slučaju daje:

$$M_{ij} = \int_0^{2\pi}\int_0^{\pi} Y_i(\theta,\varphi+\alpha)Y_j(\theta,\varphi)\sin\theta d\theta d\varphi$$

gdje su $Y_i(\theta,\varphi+\alpha)$ rotirane funkcije sfernih harmonika, a $Y_j(\theta,\varphi)$ ne-rotirane. Gornji izraz je zapisan u takvom obliku, da se dobiveni koeficijenti M_{ij} lako zapišu u matričnom obliku. Dakako, pravila matričnog množenja će osigurati da množenjem ove matrice s ne-rotiranim koeficijentima projekcije u sferne harmonike dobijemo upravo rotirane koeficijente. Primjenom opisane tehnike, možemo izračunati matricu rotacije oko Z osi za neki kut α, za prva tri pojasa:

$$\mathbf{Z}_\alpha = \left[\begin{array}{c|ccc|ccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & \cos(\alpha) & 0 & \sin(\alpha) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\sin(\alpha) & 0 & \cos(\alpha) & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & \cos(2\alpha) & 0 & 0 & 0 & \sin(2\alpha) \\
0 & 0 & 0 & 0 & 0 & \cos(\alpha) & 0 & \sin(\alpha) & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\
0 & 0 & 0 & 0 & -\sin(2\alpha) & 0 & 0 & 0 & \cos(2\alpha)
\end{array}\right]$$

Odmah uočavamo vrlo zanimljivu pojavu – elementi matrice koji odgovaraju pojedinim pojasevima su odvojeni, i ne utječu jedan na drugoga. Također, nulti pojas se sastoji samo od jedne jedinice, dok se prvi pojas sastoji od jedinice oko koje su raspoređeni sinusi i kosinusi. Nadalje, drugi pojas se sastoji od elemenata nultog pojasa (jedinice u sredini), elemenata prvog pojasa (sinusa i kosinusa raspoređenih oko jedinice u sredini), te sinusa i kosinusa dvostrukog kuta, koji su opet raspoređeni istim redosljedom oko elemenata prethodnih pojaseva. Doista, matematičkom indukcijom se može dokazati da se svaki sljedeći pojas sastoji od svih elemenata prethodnih pojaseva, te dodatnih sinusa i kosinusa kuta $N\alpha$. Također, i matrica rotacije oko X osi se može izgraditi na analogan način. Ovime smo pronašli jednostavan i efikasan rekurzivan postupak za generiranje rotacijskih matrica, koji se lako može implementirati na računalu.

Na kraju, nakon upoznavanja potrebne matematičke pozadine, možemo rezimirati glavne korake algoritma predizračunatog prijenosa zračenja. Predfaza se sastoji od sljedećih koraka:

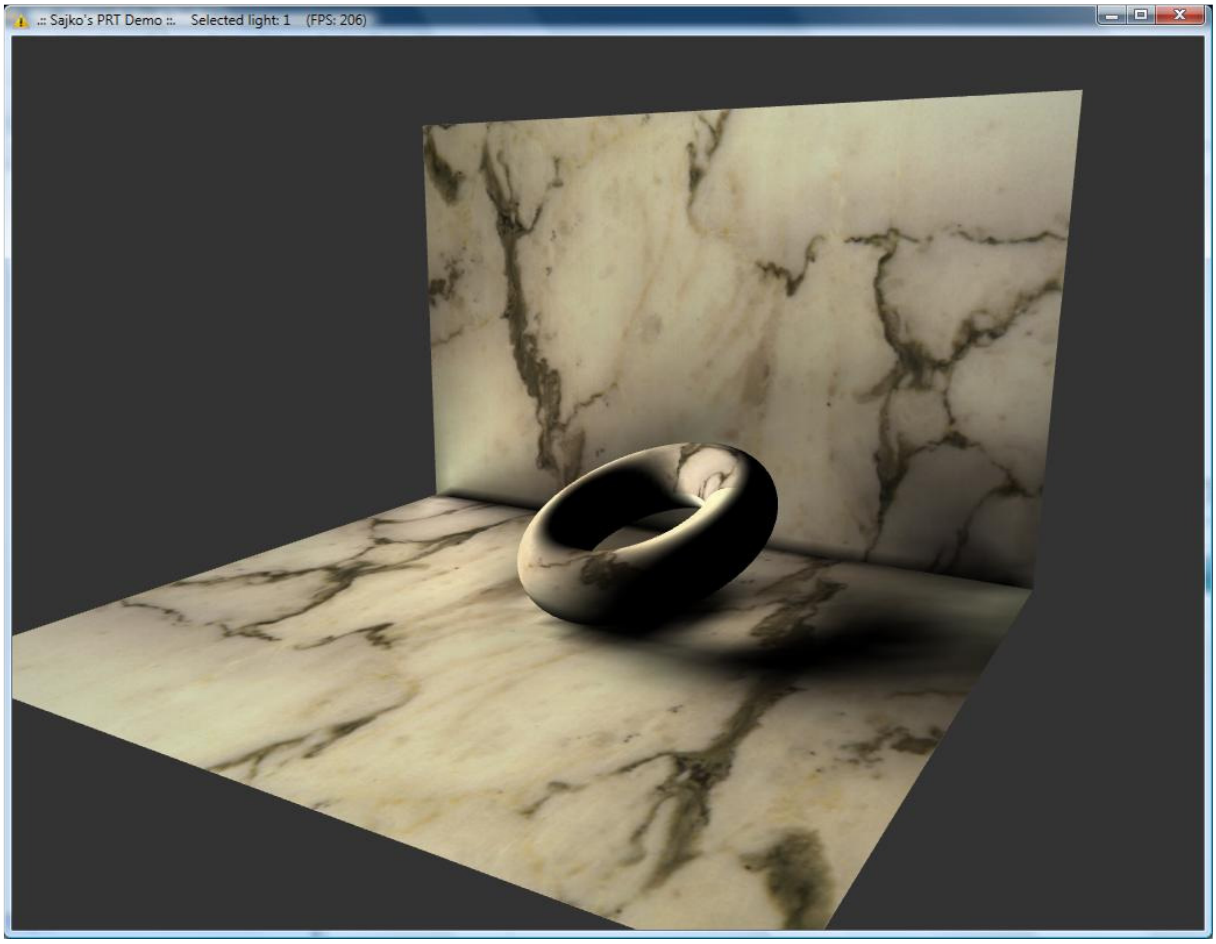
1. Generiranje dovoljne količine uniformnih uzoraka površine sfere.
2. Za svaki uzorak, izračuna se vrijednost funkcije sfernih harmonika, za prvih *n* pojaseva.
3. Za svaki vrh poligona koji sačinjavaju scenu, procijeni se funkcija prijenosa svjetlosti, pomoću algoritma praćenja zrake. Ta funkcija mora uključivati barem

distribuciju reflektivnosti, a može uključivati i detekciju sjena (tehnika zrake sjene, opisana u poglavlju o praćenju zrake), rekurzivnu sumaciju indirektnih doprinosa osvjetljenja, te ispodpovršinsko raspršivanje.

4. Paralelno s računanjem funkcije prijenosa, vrši se Monte Carlo integracija umnoška funkcije prijenosa i prethodno izračunatih vrijednosti funkcija sfernih harmonika. Ovime se dobivaju koeficijenti projekcije funkcije prijenosa za svaki pojedini vrh.
5. Koeficijenti izračunati za svaki vrh se zapisuju u datoteku, za kasniju upotrebu pri iscrtavanju scene.

Jednom kad je predprocesiranje scene dovršeno, može započeti samo iscrtavanje, koje se sastoji od sljedećih koraka:

1. Projekcija funkcije osvjetljenja u sferne harmonike. Ta funkcija može biti vrlo jednostavna, npr. konstanta, ili pak neka od funkcija koje opisuju jačinu Sunčevog zračenja (ovisno o potrebama aplikacije). Ovaj korak je dovoljno izvršiti tek jedamput, za svako svjetlo na sceni.
2. Prilikom svakog iscrtavanja slike, za svaki vrh poligona koji sačinjavaju scenu izračuna se skalarni umnožak vektora koeficijenata (učitanih iz datoteke) sa zbrojenim vektorom koeficijenata svih svjetala na sceni (po potrebi, taj se vektor još rotira, množenjem s rotacijskom matricom). Dobiveni broj predstavlja svjetlinu danog vrha, i može se shvatiti kao (monokromatska) boja vrha. Ovaj korak sjenčanja vrha se može vrlo efikasno implementirati jezikom sjenčanja, tako da se izvršava na grafičkoj kartici, paralelno s transformacijom vrha.



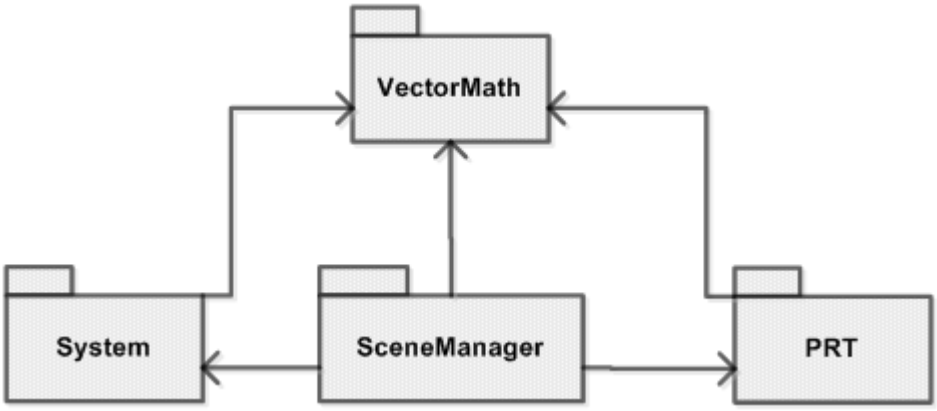
Slika 21. Scena iscrtana algoritmom predizračunatog zračenja.

7. Implementacija

Programski jezik odabran za implementaciju algoritma predizračunatog prijenosa zračenja jest C++, zbog svoje moćne objektne paradigme i efikasnosti. Grafičko programsko sučelje korišteno pri izradi aplikacije jest OpenGL, koji je odabran prvenstveno zbog jednostavnosti uporabe. Zahtjevi nad programskom potporom su sljedeći: mogućnost rada s bilo kojim virtualnim okruženjem priloženim u odgovarajućem formatu; simulacija tog virtualnog okruženja u stvarnom vremenu s naprednim osvjetljenjem, koje pokazuje (barem neke) globalne učinke osvjetljenja koji se ne mogu simulirati postojećim lokalnim modelima. Prvi zahtjev je ispunjen tako da se virtualno okuženje interpretira kao skup objekata koji su raspoređeni u prostoru na neki predodređeni način, i koji posjeduju predodređene karakteristike. U toj interpretaciji, virtualno okruženje možemo nazvati *scenom*, a komponente virtualnog okruženja *objektima scene*. S tim u vidu, za potrebe demonstracije programske potpore, konstruirani su jednostavni 3D objekti korištenjem Autodesk Maya 2008 alata, i zatim eksportirani u jednostavni i općeprihvaćeni .obj format. U svrhu povezivanja tih individualnih objekata u cjelovitu scenu, potrebna je dodatna datoteka koja služi kao opisnik scene. Kao takva, njen je format vrlo jednostavan – riječ je tek o popisu objekata koji se pojavljuju na sceni. Budući da se u Maya alatu objekti mogu već prije eksportiranja translatirati, rotirati i postaviti na scenu u odgovarajuću poziciju, format opisa scene ne treba sadržavati takve podatke. Jedino što je potrebno jest označiti željenu teksturu. Dakle, opis scene se sastoji od stavaka sljedećeg oblika:

```
"NazivDatotekeObjekta.obj" "NazivDatotekeTeksture.tga"
```

Navodnici su potrebni da se omoguće praznine u nazivu datoteka. Sljedeći i glavni zahtjev se može ispuniti korištenjem predizračunatog prijenosa zračenja, što je trenutno ponajbolji izbor za interaktivnu simulaciju globalnog osvjetljenja. Nakon definiranja svih zahtjeva i općih smjernica njihovog ispunjenja, moguće je dizajnirati arhitekturu programske potpore. Prvi korak dizajna jest razlučivanje odgovornosti, i njihovo pridjeljivanje paketima. Shodno tome, možemo razlikovati četiri glavna paketa, kako to prikazuje sljedeći dijagram:

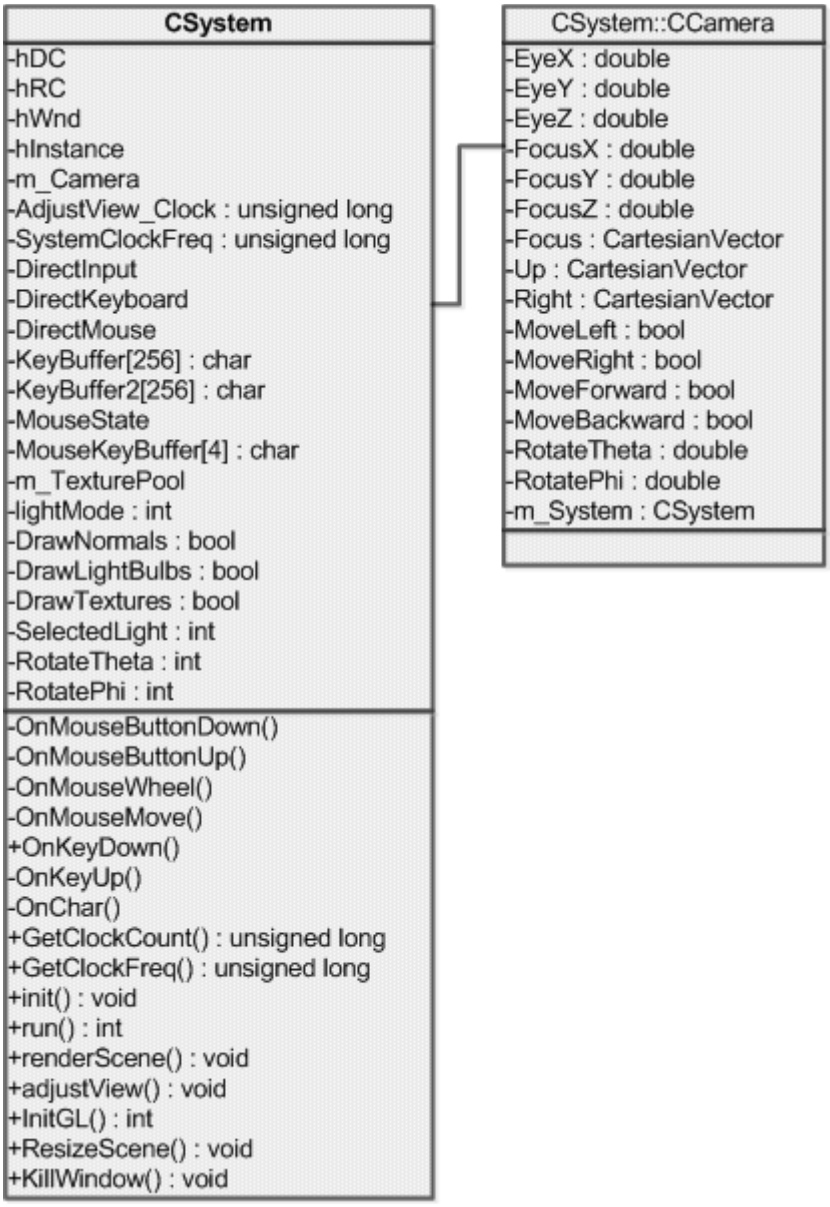


Slika 22. Objektni model implementacije.

Temeljni paket jest `System`, koji je odgovoran za sve sistemske operacije poput stvaranja prozora za iscrtavanje, rukovanje ulaznim jedinicama itd. Zatim, paket `SceneManager` sadrži i upravlja svim podacima koji određuju scenu. Između ostalog, razredi iz tog paketa će biti zaslužni za učitavanje .obj datoteka. Najvažniji paket jest `PRT`, koji sadrži svu

funkcionalnost potrebnu za provođenje algoritma predizračunatog zračenja, dakle i predprocesiranje scene, i samo iscrtavanje. Na kraju, pomoćni paket `VectorMath` sačinjavaju razredi za rad s pravokutnim i sfernim vektorima. Sada slijedi detaljniji pregled svakog od ovih paketa, sa UML dijagramima razreda, te objašnjenjima.

`System`



Slika 23. UML dijagram razreda `CSystem` i `CSystem::CCamera`.

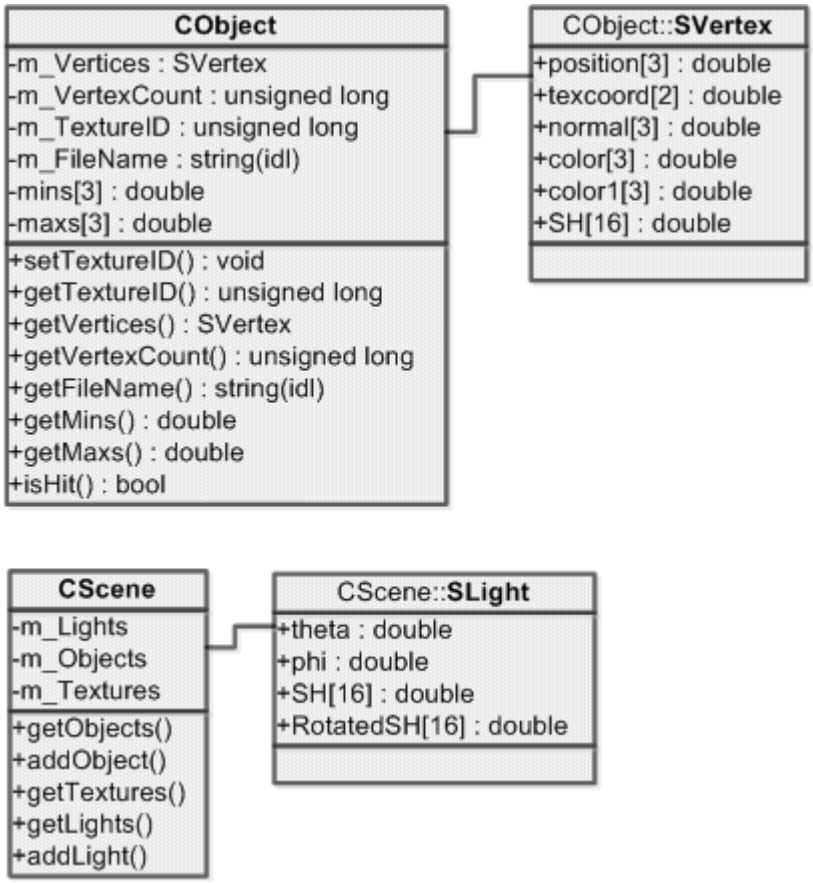
Ovaj paket se sastoji od dvaju razreda, `CSystem` i `CSystem::CCamera`. Prvi razred enkapsulira računalni sustav i njegove ulazne jedinice. Stvaranje prozora za iscrtavanje se vrši direktnim pozivima Windows API-ja, dok se inicijalizacija crtaćeg konteksta ostvaruje OpenGL naredbama. Grafička kartica, predstavljena nekim programskim sučeljem poput OpneGL-a, se također može shvatiti kao jedinica unutar sustava. Razlikuje se faza inicijalizacije (stvaranje crtaćeg konteksta), faza aktivnog rada (opetovano iscrtavanje scene, određeni broj puta u sekundi), te faza deinicijalizacije i prekida rada (otpuštanje

crtaćeg konteksta, oslobađanje memorije). Iz tog razloga, kao što se vidi na gornjem dijagramu, i funkcije upravljanja iscrtavanjem su podijeljene na te faze. Komunikacija s ulaznim jedinicama se ostvaruje pomoću DirectInput tehnologije, što je dio Microsoftovog DirectX skupa tehnologija. Reaktivnost aplikacije na ulazne podražaje (pritisci tipaka, pomaci miša) je ostvarena korištenjem paradigme vođene događajima (*event driven paradigm*). Naime, pojedine korisničke akcije su shvaćene kao događaji, te sustav detektira i identificira različite događaje korištenjem nekog sklopovskog sučelja niže razine (konkretno, DirectInput-a). Zatim, ti se događaji propagiraju kroz sustav posredstvom raspačivača (što je ovdje `CSystem`) tako da se pozivaju odgovarajuće funkcije rukovoditelji (*event handlers*). U općenitom slučaju, bilo koji objekt u aplikaciji može registrirati svoje rukovoditelje kod nekog objekta raspačivača, no u našem slučaju jedini raspačivač jest razred `CSystem`, a jedini rukovoditelj `CSystem::CCamera` (doduše, radi brzine izvođenja, provedena je optimizacija tako da se izbjegne dvostruko indirektni poziv, pa je tehnički gledano `CSystem` i raspačivač, i rukovoditelj). Kako se može naslutiti, razred `CSystem::CCamera` jest apstrakcija koncepta virtualne kamere, i sadrži podatke o položaju očišta (točka u kojoj se kamera nalazi), gledišta (točka fokusa kamere, odnosno „smjer gledanja“), te takozvani vektor prema gore, koji kazuje koji smjer je za kameru „gore“. Dakle, pritiscima tipaka i pomacima miša, moguće je pomicati pogled bilo gdje u sceni. Osim ove osnovne interaktivnosti, dozvoljeno je pomicati i točku izvora svjetlosti, i to tako da se ona rotira oko ishodišta za bilo koja dva kuta (θ, φ), pri čemu možemo reći da (θ, φ) zapravo predstavljaju sferne koordinate izvora svjetlosti. Konkretna izvedba rotacije projicirane funkcije upadnog svjetla je objašnjena u teorijskom izlaganju algoritma predizračunatog prijenosa zračenja, a implementirana je u `CPRT` razredu, koji je objašnjen kasnije u ovom tekstu. U ovom trenutku valja spomenuti da aplikacija podržava i klasično osvjetljenje Blinn-Phong lokalnim modelom, sklopovski dostupnim putem OpenGL-a. Budući da je podržano više od jednog istovremenog izvora svjetla, tipkama 1-8 se bira trenutno aktivno svjetlo (dakle, ono na koje se rotacija odnosi). Radi sklopovskog ograničenja na maksimalno osam istovremenih izvora svjetla, aplikacija ne podržava više od tog broja aktivnih svjetala, iako sam broj svjetala tek neznatno utječe na performanse iscrtavanja algoritmom PRT. Također, registriraju se i neke kontrole za olakšavanje testiranja programa. Sve navedene kontrole su pregledno dane sljedećom tablicom:

Tablica 1. Kontrole programa.

Translacija kamere unaprijed	↑ (strelica prema gore)
Translacija kamere unazad	↓ (strelica prema dolje)
Translacija kamere ulijevo	← (strelica ulijevo)
Translacija kamere udesno	→ (strelica udesno)
Rotacija kamere lijevo-desno	Pomicanje miša lijevo-desno
Rotacija kamere gore-dolje	Pomicanje miša gore-dolje
Odabir trenutno aktivnog svjetla	Tipke 1-8
Povećavanje/smanjivanje θ koordinate izvora	Q / A
Povećavanje/smanjivanje φ koordinate izvora	W / E
Paljenje/gašenje iscrtavanja tekstura	T
Paljenje/gašenje iscrtavanja normala	N
Paljenje/gašenje iscrtavanja vektora izvora	L
Standardno OpenGL osvjetljenje	F1
Osvjetljenje PRT algoritmom	F2

SceneManager

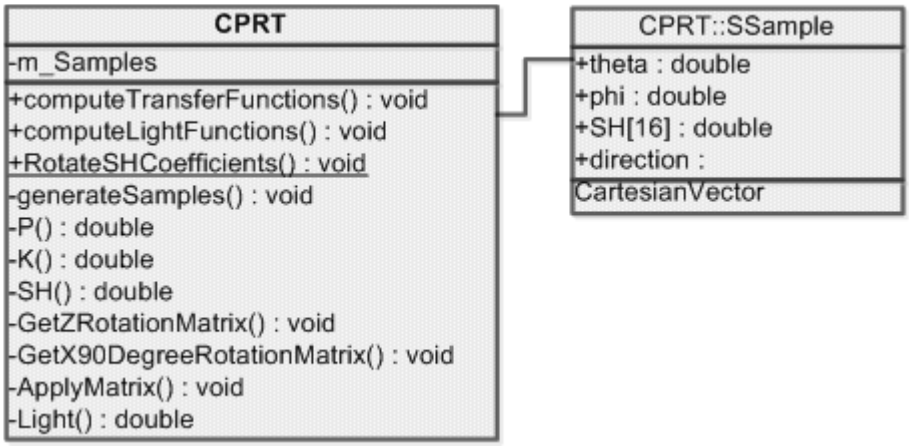


Slika 24. UML dijagram razreda `CScene` i `CObject`.

Sljedeći bitan skup razreda su `CScene` i `CObject`. Ova dva razreda redom predstavljaju enkapsulacije scene, te objekata scene. Iz tog razloga, aplikacija u pravilu treba instancirati samo jedan objekt tipa `CScene`. Konstruktor tog razreda automatski učitava zadanu ulaznu datoteku s opisom scene, te stvara potreban broj objekata, koje pohranjuje u internom spremniku. Sam postupak parsiranja .obj datoteka se delegira na novostvorene `CObject` objekte. Pojedini objekt scene je definiran svojim topološkim i geometrijskim podacima. Geometrijski podaci su jednostavno skup točaka koji definiraju poligone objekta, najčešće trokute, dok su topološki podaci logičke prirode. Naime, oni govore na koji način se prethodno definirane točke interpretiraju, odnosno, kojim redosljedom tvore poligone. Upravo ovakav pristup se koristi u .obj formatu, zato što omogućava da se vrhovi poligona zapišu samo jedanput, a zatim po potrebi dijele između susjednih poligona, čime se ostvaruje značajna ušteda na memoriji. Međutim, radi jednostavnosti kasnije obrade, vrhovi i poligoni se u `CObject` klasi ne pohranjuju odvojeno, već se točke jednostavno nižu onim redosljedom kojim tvore poligone, što znači da se većina njih ponavlja, i do nekoliko puta. Pojedini vrh je predstavljen strukturom `CObject::SVertex`, koja osim pozicije točke u prostoru pohranjuje i neke druge podatke vezane uz taj vrh, a to su koordinate teksture, boja, te koeficijenti projekcije funkcije prijenosa svjetlosti u sferne harmonike, u toj konkretnoj točki. Broj tih koeficijenata je limitiran na 16, budući da se pokazalo da je već i tako malen broj sasvim dovoljan za kvalitetne rezultate. Po potrebi, ovo se može i promijeniti, budući da nema nikakvog utjecaja na izvođenje samog

algoritma (osim produljenja trajanja). Kako se može primijetiti, struktura za pohranu vrhova sadrži dva polja za pohranu boje. Razlog tomu je taj, što aplikacija podržava i klasično, lokalno osvjetljenje, i PRT osvjetljenje. Budući da je krajnji rezultat PRT algoritma boja pojedinog vrha, tu boju je potrebno pohraniti. No, boja utječe i na OpenGL-ov ugrađeni model osvjetljenja, pa je potrebno sačuvati bijelu boju za svaki vrh, i po potrebi alternirati između njih. Sam izvor svjetla je enkapsuliran `CScene::SLight` strukturom, koja pohranjuje sferne koordinate izvora, koeficijente izvorne projekcije funkcije osvjetljenja tog izvora, te rotirane koeficijente projekcije, koji su dobiveni rotacijom izvornih koeficijenata za kutove koji odgovaraju trenutnim sfernim koordinatama izvora. Izvorne koeficijente je potrebno sačuvati zato jer postupak rotacije opisan u prethodnom teorijskom izlaganju pretpostavlja apsolutnu rotaciju, a ne relativnu. Budući da je tokom izvršavanja aplikacije moguće tek dobivati relativne pomake između pojedinih iscrtavanja scene, ti relativni pomaci se moraju zbrajati. Zatim, izvorni koeficijenti projekcije se rotiraju za te zbrojene, apsolutne kutove.

PRT



Slika 25. UML dijagram razreda `CPRT`.

Jezgra cijele aplikacije jest upravo razred `CPRT`. Naime, ovaj razred sadrži svu funkcionalnost potrebnu za provedbu algoritma predizračunatog zračenja. Glavne funkcije, koje ujedno čine javno sučelje razreda, su sljedeće:

```
void computeTransferFunctions(
            CScene& Scene,
            fpProgressCallback ProgressCallback,
            bool FastMode
      )
```

Ulazni argumenti:

`Scene`	referenca na instancu `CScene` klase
`ProgressCallback`	pokazivač na funkciju povratnog poziva
`FastMode`	brzo predprocesiranje (`true`) ili sporo (`false`)

Ova funkcija je zaslužna za predprocesiranje scene. Iz tog razloga, potrebno je prije njenog pozivanja imati izgrađen objekt scene. Sam postupak predprocesiranja može biti vrlo dugotrajan, stoga se omogućava objavljivanje napretka. U tu svrhu, upotrijebljen je mehanizam funkcije povratnog poziva (*callback function*). Naime, korisnik treba napisati vlastitu funkciju odgovarajućeg prototipa, i predati pokazivač na tu funkciju kao argument funkciji `computeTransferFunctions`. Tokom predprocesiranja, funkcija čiji je pokazivač predan će povremeno biti pozivana, točnije, prilikom svakih 1% obavljenog posla. Iz tog razloga, ovaj mehanizam se naziva povratnim pozivom. Generalno govoreći, očekuje se da će ta povratno pozivajuća funkcija obavještavati korisnika o napretku algoritma, primjerice, osvježavanjem nekog elementa grafičkog sučelja aplikacije i/ili tekstualnim ispisom. Upravo je to slučaj u ovoj aplikaciji, kako će biti objašnjeno kasnije. Posljednji argument funkcije specificira način obrade scene. Naime, kako je spomenuto u teorijskom izlaganju, funkcija prijenosa čija se projekcija računa za svaki vrh može biti različitog oblika. Ona može uključivati samo difuznu refleksiju, bez detekcije zaklonjenosti, ili se pomoću zrake sjene može dodatno utvrditi i da li je trenutni vrh u sjeni. Ovaj drugi način je očito sporiji, pa je za potrebe testiranja scene ili same aplikacije mnogo praktičnije preskočiti taj korak i značajno ubrzati obradu scene.

```
void computeLightFunctions(
            CScene& Scene
      )
```

Ulazni argumenti:

`Scene`	referenca na instancu `CScene` klase

Ova funkcija također vrši projekciju u sferne harmonike, no ovaj put za svjetla. Budući da je za funkciju izlaznog isijavanja izvora uzeta konstanta, ovaj je postupak relativno jednostavan i brz. Geometrijske karakteristike scene nisu bitne, no podaci o svjetlima jesu, pa je potrebna referenca na objekt scene, koji sadrži podatke o svjetlima.

```
static void RotateSHCoefficients(
                  int numBands,
                  double* unrotatedCoeffs,
                  double* rotatedCoeffs,
                  double theta, double phi
            )
```

Ulazni argumenti:

`numBands`	Red projekcije (broj pojaseva sfernih harmonika)
`unrotatedCoeffs`	Izvorni koeficijenti projekcije (prije rotacije).
`theta, phi`	Željene sferne koordinate izvora.

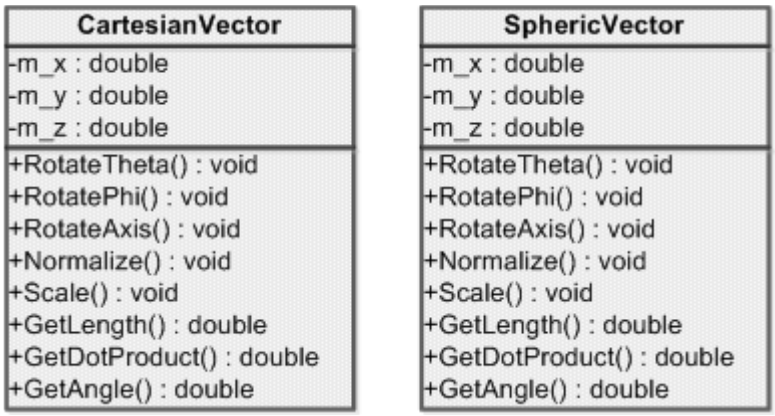
Izlazni argumenti:

`rotatedCoeffs`	Pokazivač na spremnik rotiranih koeficijenata.

Funkcija za rotiranje koeficijenata projekcije vrši prethodno opisan postupak rotacije. Prvo se rekurzivnim postupkom generiraju matrice za rotaciju oko Z i X osi, a zatim se one izmnože i time stvori konačna matrica rotacije. Množenjem izvornih koeficijenata s tom matricom dobiju se rotirani koeficijenti, koji se pohranjuju u spremnik čiji je pokazivač predan kao argument funkciji. Ova funkcija je deklarirana kao statička, iz razloga da se omogući njeno korištenje i bez instanciranja razreda kojemu pripada.

Ostali članovi razreda `CPRT`, koji sačinjavaju njegov privatni dio, su određene pomoćne funkcije koje nemaju svrhu same za sebe, već su potrebne za prethodne tri funkcije. U ovu skupinu spadaju metode za generiranje uniformnih uzoraka sfere, konstruiranje asociranih Legendre polinoma, evaluaciju funkcija sfernih harmonika u danoj točki, generiranje matrica rotacije oko Z i X osi, i slično. Implementacije tih metoda su direktno preuzete iz teorijskog izlaganja algoritma.

`VectorMath`



Slika 26. UML dijagram razreda `CartesianVector` i `SphericVector`.

Posljednji, i najmanji paket se sastoji od dvaju razreda za olakšavanje računanja s vektorima, zadanih sfernim ili kartezijevim koordinatama. Njihovo javno sučelje jest identično, i uključuje često korištene operacije poput rotacije oko proizvoljne osi, normalizacije, skaliranja, skalarnog množenja itd. Zbog česte i raširene potrebe za vektorskom matematikom, ove operacije su enkapsulirane u zasebnom paketu, kojeg koriste svi ostali paketi.

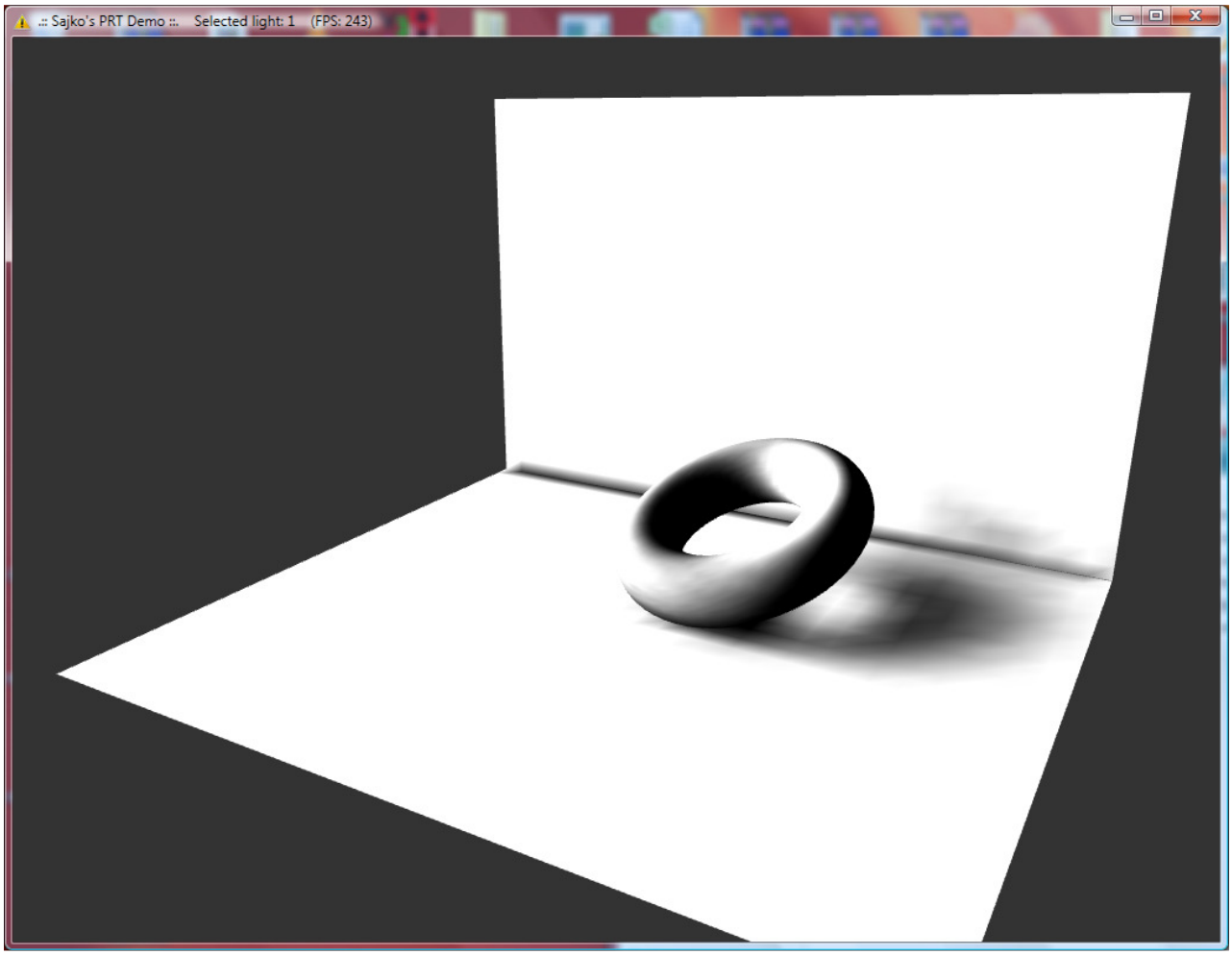
*
* *

Dizajn paketa i razreda programskog rješenja predstavlja njegovu logičku arhitekturu, čiji je pregled upravo dan. Fizički dizajn uključuje raspodjelu aplikacije u izvršne module i podatke. Budući da implementirani algoritam jasno razlikuje dvije odvojene faze – predprocesiranje i samo iscrtavanje – i programska podrška je fizički dizajnirana kao dvije odvojene aplikacije. Prva aplikacija se naziva SHBuilder, i služi za obradu scene i generiranje koeficijenata projekcije funkcije prijenosa. Druga aplikacija, koja se naziva SHDemo, učitava prethodno generirane koeficijente, i na temelju njih vrši iscrtavanje scene u stvarnom vremenu. Pritom obje aplikacije koriste iste, prethodno opisane pakete

razreda. Jedina je razlika u tome što SHBuilder sadrži grafičko korisničko sučelje, za lakšu interakciju s programom. Iz tog razloga, umjesto paketa System, koristi se Microsoftova biblioteka MFC, koja omogućava jednostavniju izgradnju Windows aplikacija s grafičkim korisničkim sučeljem. Sada možemo na konkretnom primjeru pokazati korisnost prethodno objašnjenog mehanizma povratnog poziva – kako predprocesiranje scene napreduje, tako se osvježava *ProgressBar* kontrola (horizontalni stupac, inicijalno prazan, koji progresivno postaje sve ispunjeniji), te se ispisuje očekivano vrijeme trajanja obrade scene (na temelju trajanja obrade dotad obrađenog dijela scene). Budući da vrijeme procesiranja može biti dugotrajno (na Pentium IV 2.0 GHz računalu je trajalo otprilike tri sata za demonstracijsku scenu), u prilogu ovom radu se nalazi aplikacija s demonstracijskom scenom s već predizračunatim koeficijentima. Rezultati implementacije su u skladu s očekivanjima. Iz testiranja na nekoliko računala, osvjetljenje PRT algoritmom je tek oko 15%, do najviše 40% sporije nego osvjetljenje dobiveno ugrađenim OpenGL modelom osvjetljenja (pogledati donju tablicu). Valja imati na umu da je ovdje riječ o naivnoj implementaciji, koja direktno slijedi teorijske rezultate bez ikakvih optimizacija. Prvi korak koji bi svaka ozbiljna implementacija trebala poduzeti jest prebacivanje izračuna boje pojedinog vrha na grafičku karticu. Iako je ta operacija vrlo jednostavna, i sastoji se tek od 16 množenja i zbrajanja, takve operacije grafičko sklopovlje izvršava mnogo brže nego glavni procesor. Također, budući da će sve podatke o vrhovima pohranjivati i obrađivati grafička kartica, ne gubi se vrijeme na komunikaciju glavnog procesora i grafičke kartice. Osim toga, i postupak predprocesiranja scene se može barem djelomično implementirati nekim jezikom sjenčanja, i time značajno ubrzati. Optimizirane implementacije PRT algoritma bi zapravo lako mogle biti i *brže* od lokalnih modela osvjetljenja, ukoliko uzmemo u obzir situacije s više od jednog svjetla na sceni. Naime, konačna boja nekog vrha jest skalarni umnožak vektora koeficijenata tog vrha, i konačnog vektora koeficijenata osvjetljenja. Taj konačni vektor osvjetljenja se dobije kao zbroj vektora koeficijenata svih izvora. Taj se zbroj može izračunati jedanput, a zatim, ukoliko dođe do rotacije pojedinog svjetla, dovoljno je izračunati razliku u odnosu na nerotirano svjetlo, i taj vektor razlike pribrojiti ukupnom zbroju. Drugim riječima, kompleksnost izračuna boje danog vrha time postaje konstantna, uopće ne oviseći o broju svjetala. S druge strane, kod lokalnih modela osvjetljenja, za svako svjetlo se cjelokupan postupak računanja mora ponoviti. Sljedeća tablica ilustrira performanse (izražene u broju iscrtanih slika u sekundi) ove neoptimizirane implementacije, i dobro pokazuje da je upravo centralni procesor usko grlo:

Tablica 2. Performanse implementiranog algoritma.

		1 svjetlo	8 svjetala
Pentium IV 2.0 GHz, GeForce4 Ti4400 (starost 6 godina)	Blinn-Phong	102	66
	PRT	59	16
Intel Core Duo, 1.67 Ghz, Radeon x1600 (starost 2 godine)	Blinn-Phong	184	163
	PRT	131	72
Intel Core 2 Duo 2.6 Ghz, Radeon HD3870 (starost mjesec dana)	Blinn-Phong	268	243
	PRT	209	124



Slika 27. Primjer artefakata nastalih zbog interpolacije osvjetljenja između vrhova.

Promatrajući generirane slike, do izražaja dolazi činjenica da algoritam računa boju *po vrhu*. Naime, prilikom kreiranja konačne slike, u postupku rasterizacije se mora odrediti boja svakog pojedinog slikovnog elementa. Budući da boju specificiramo po vrhovima, slikovni elementi koji upadaju na područja između pojedinih vrhova, nakon projekcije u prostor slike, imaju nedefiniranu boju. Jednostavno rješenje problema jest linearna interpolacija boje između dva susjedna vrha. No, to znači da ukoliko imamo nedostatan broj vrhova, diskretizacija će biti očita, budući da će zamjetno velika područja slike biti obojana istom bojom. To također znači da neće biti finih prijelaza osvjetljenja, već će biti vidljive grube razine. Primjer daje slika 27, nastala isključivanjem tekstura (usporedbe radi, ta je slika nastala u istovjetnim uvjetima kao slika 21).

Na sjeni koju baca torus, uočavaju se dvije razine – najtamnija, središnja sjena, koja se naziva umbra, te okolna, svjetlija sjena koja se naziva penumbra. Prijelaz između umbre i penumbre bi trebao biti gladak, no lako se mogu uočiti diskretne razine osvjetljenja, budući da su pojedini vrhovi koji su unutar umbre gotovo potpuno crni, no dijelovi poligona koji bi također trebali biti unutar umbre su sivi, budući da je boja tih slikovnih elemenata dobivena linearnom interpolacijom između vrha unutar umbre, i vrha unutar penumbre. Na sjeni koju torus baca na samog sebe, takvi artefakti su mnogo teže uočljivi, budući da se torus sastoji od 10800 jedinstvenih vrhova, dok se okolne plohe sastoje od svega 2700 vrhova. Međutim, ukoliko usporedimo ovu sliku sa slikom 21, koja je ista ali s uključenim teksturama, navedeni artefakti više uopće nisu vidljivi, budući da ih tekstura čini nezamjetnima. Također, današnje aplikacije već barataju i s mnogo većim brojem vrhova, tako da ovo zasigurno više nije problem.

8. Zaključak

U ovom radu, dan je pregled svih najvažnijih dostignuća u problematici realističnog, interaktivnog osvjetljenja u računalnoj grafici. Glavni razlog neuvjerljivosti trenutno korištenih tehnika jest ignoriranje globalnih učinaka svjetlosti. Postoji mnogo različitih pristupa simulaciji globalnog osvjetljenja, stoga su razlučeni u dvije glavne skupine: tradicionalne metode prilagođene za potrebe interaktivne grafike, te novorazvijeni algoritmi. U tradicionalne metode spadaju algoritam praćenja zrake i njegova razna proširenja, te algoritmi bazirani na postupku isijavanja. S druge strane, novijih, specifičnih algoritama ima veliko mnoštvo, no za potrebe ovog rada mogu se izdvojiti dva najkarakterističnija postupka. To su algoritam zaklanjanja ambijenta, te algoritam predizračunatog prijenosa zračenja (u tablici PRT, prema engleskom nazivu). Nakon detaljnijeg upoznavanja svih ovih pojedinačnih postupaka, korisno je sagledati cjelokupno stanje stvari, odnosno, načiniti usporedbu i ocjenu izloženih algoritama, i time dobro uočiti njihove jake i slabe strane:

Tablica 3. Pregled značajki svih prethodno predstavljenih algoritama.

	Praćenje zrake	Preslikavanje fotona	Isijavanje	Zaklanjanje ambijenta	PRT
Difuzna refleksija	-	Da	Da	Da	Da
Zrcalna refleksija	Da	Da	-	-	-
Difuzna interrefleksija	-	Da	Da	Da	Da
Zrcalna interrefleksija	Da	Da	-	-	-
Oštre sjene	Da	Da	-	-	-
Meke sjene	-	Da	Da	Da	Da
Samozasjenjivanje	Da	Da	Da	Da	Da
Kaustika	-	Da	-	-	-
Ispodpovršinsko raspršivanje	-	Da	-	-	Da
Dinamička geometrija	Da	Da	Da	Da	-
Dinamička svjetla	Da	Da	Da	Da	Da
Kompleksnost	Visoka	Vrlo visoka	Visoka	Srednja	Niska

Iz gornjeg pregleda, jasno se može uočiti da najkvalitetnije rezultate daje algoritam praćenja zrake proširen postupkom preslikavanja fotona. Nažalost, to je ujedno i najsporije rješenje. Iako postoje implementacije tog postupka u jezicima za sjenčanje, dakle u sklopovskoj izvedbi, računala danas dostupna prosječnim korisnicima još uvijek nemaju dovoljnu računalnu moć za efikasno korištenje ovakvih tradicionalnih metoda u interaktivnoj grafici. Moguće je predviđati da će situacija biti znatno drugačija za desetak godina, no zasad jedino primjereno rješenje predstavljaju algoritmi poput zaklanjanja ambijenta ili predizračunatog prijenosa zračenja. Oba algoritma su dovoljno efikasna za postizanje interaktivnih brzina iscrtavanja, s time da je PRT barem nekoliko puta brži od zaklanjanja ambijenta. Doduše, ta brzina dolazi sa cijenom, a to je nemogućnost deformacije ili translacije geometrije, te potrebitost dugotrajog predprocesiranja scene. Makar sporije rješenje, zaklanjanje ambijenta nema ovakvih ograničenja, stoga je bolje rješenje ukoliko je potrebna dinamička geometrija, ili ako je fazu predprocesiranja nemoguće izvršiti (primjerice, ukoliko se geometrija scene generira proceduralno, prilikom samog iscrtavanja). Međutim, u svim ostalim slučajevima, PRT je zasigurno najbolji mogući izbor, stoga je taj algoritam implementiran u sklopu ovog rada.

9. Literatura

[1] Henč-Bartolić, Višnja. Kulišić, Petar. *Valovi i optika*, izdanje 3, Školska knjiga, 2004.

[2] Haidinger, Wilhelm Karl Ritter von. *Über das directe Erkennen des polarisirten Lichts und der Lage der Polarisationsebene*, Annalen der Physik, svezak 139, broj 9, stranice 29-39, 1844.

[3] Kajiya, James T. *The rendering equation*, SIGGRAPH 1986.

[4] Goral, C. Torrance, K. E. Greenberg, D. P. Battaile, B. *Modeling the interaction of light between diffuse surfaces*, Computer Graphics, svezak 18, broj 3

[5] Waters, Zack. *Photon Mapping*, http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html

[6] Pohl, Daniel. *Ray Tracing and Gaming - One Year Later*, PC Perspective, 17.1.2008., http://www.pcper.com/article.php?aid=506

[7] *CUDA*, http://en.wikipedia.org/wiki/CUDA, 21.4.2008.

[8] *Close to Metal*, http://en.wikipedia.org/wiki/Close_to_Metal, 21.4.2008.

[9] Purcell, Timothy J. Donner, Craig. Cammarano, Mike. Jensen, Henrik Wann. Hanrahan, Pat. *Photon Mapping on Programmable Graphics Hardware*, SIGGRAPH 2003.

[10] Bunnell, Michael. *Dynamic Ambience Occlusion and Indirect Lighting*, 7.3.2005, ftp://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch14.pdf

[11] Green, Robin. *Spherical Harmonic Lighting: The Gritty Details*, 16.1.2003., http://www.research.scea.com/gdc2003/spherical-harmonic-lighting.pdf

[12] Sloan, P.P. Kautz, J. Snyder, J. *Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments*, SIGGRAPH 2002.

[13] Elezović, Neven. *Diskretna vjerojatnost*, 0. izdanje, Element, 2007.

[14] Elezović, Neven. *Slučajne varijable*, 0. izdanje, Element, 2007.

[15] Elezović, Neven. *Statistika i procesi,* 0. izdanje, Element, 2007.

10. Sažetak

(Postupci ostvarivanja globalnog osvjetljenja)

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. U računalnoj znanosti su poznati tradicionalni postupci koji točno rješavaju ovaj problem, no oni su prezahtjevni za praktične primjene u interaktivnim aplikacijama. Ovaj rad daje pregled tih tradicionalnih metoda, te vodi čitatelja kroz različite pokušaje njihove prilagodbe za upotrebu u interaktivnoj grafici. Također, predstavljaju se i neki novi algoritmi, specifično razvijeni za postizanje čim povoljnijeg kompromisa između kvalitete i performansi. Na kraju, dan je pregled i ocjena svih opisanih tehnika, od kojih je najpovoljnija i implementirana. Algoritam odabran za implementaciju jest predizračunati prijenos zračenja, u obliku kompletnog programskog paketa za obradu i simulaciju proizvoljnih virtualnih okruženja.

Ključne riječi: model osvjetljenja, lokalno osvjetljenje, globalno osvjetljenje, tehnike iscrtavanja, praćenje zrake, preslikavanje fotona, isijavanje, zaklanjanje ambijenta, predizračunati prijenos zračenja.

10. Abstract

(Methods for achieving global illumination)

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. Computer science has long known methods which solve this problem accurately, however, those traditional algorithms are computationaly too expensive to be used effectively in interactive applications. This paper outlines the traditional algorithms and leads the reader through various attempts of adapting those algorithms for use in interactive graphics. Also, more recent developments in the field are presented, comprised of algorithms specificaly designed to achieve the most satisfying compromise between quality and performance. Finally, an overview is given, including a comparison of all the algorithms described in the paper, the most favorable of which was implemented. The algorithm chosen for implementation is precomputed radiance transfer, in the form of a complete software package for processing and simulating arbitrary virtual scenes.

Keywords: lighting model, local illumination, global illumination, rendering methods, raytracing, photon mapping, radiosity, ambience occlusion, precomputed radiance transfer.