SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4068

POSTUPAK OSVJETLJAVANJA SCENE METODOM ISIJAVANJA

Igor Kramarić

Zagreb, lipanj 2015.

Sadržaj

1. Uvod 1

2. Univerzalna jednadžba iscrtavanja 2

2.1. Fizikalna svojsta svjetlosti 2

2.1.1. Refleksija 3

2.1.2. Refrakcija 4

2.2. Radijancija (engl. *Radiance*) 5

2.3 Funkcija distribucije refleksivnosti (engl. *BRDF*) 6

2.4. Univerzalna jednadžba iscrtavanja 7

3. Lokalni modeli osvjetljenja 8

3.1. Phongov model osvjetljenja 9

3.1.1. Difuzna komponenta 9

3.1.2. Zrcalna komponenta 10

3.1.3. Ambijentna komponenta 10

3.1.4. Ukupan utjecaj 11

3.2. Blinn-Phongov model osvjetljenja 12

4.Globalni modeli osvjetljenja 13

4.1. Algoritam praćenja zrake (engl. *Ray tracing*) 14

4.2. Praćenje puta zrake (engl. *Path tracing*) 17

4.3. Preslikavanje fotona (engl. *Photon mapping*) 18

4.3.1. Prva faza: Emitiranje svjetlosti i raspršivanje fotona 19

4.3.1. Druga faza: Utvrđivanje osvjetljenja i iscrtavanje 19

5. Metoda isijavanja (engl. *Radiosity*) 20

5.1. Osnovna ideja 21

5.2. Matematička formulacija 24

5.3. Faktori vidljivosti 26

5.3.1. Izračunavanje faktora vidljivosti pomoću algoritma polukocke 28

5.4. Rješavanje jednadžbe metode isijavanja 30

5.4.1. Progresivni algoritam rješavanja jednadžbe metode isijavanja 31

5.5. Prednosti i nedostaci 32

6. Implementacija metode isijavanja 32

6.1. Pomoćni razredi 32

6.2. Jezgra metode isijavanja 33

6.3. Grafičko sučelje 37

6.4. Performanse 37

7.Zaključak 40

8.Literatura 41

9. Sažetak 42

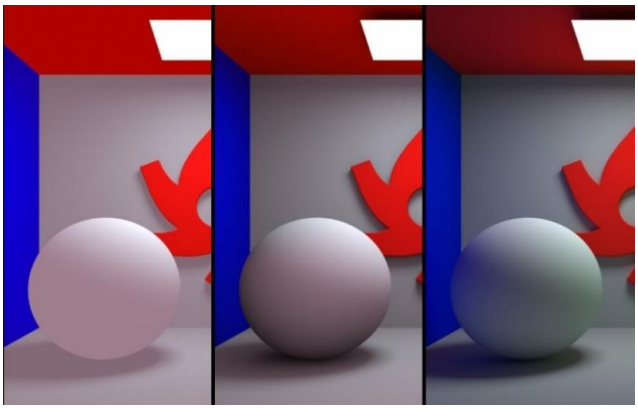
9. Abstract 42

1. Uvod

Od početka nastanka računalne grafike jedan je od glavnih ciljeva proizvesti realistične slike brzo i točno. Međutim, da bi smo mogli izrađivati realistične slike, potrebno je poznavanje interakcije svjetlosti s objektima koji nas okružuju. Interakcija svjetlosti s okolinom u stvarnom svijetu složena je jer nastaju različite optičke i fizičke pojave. Intuitivno se može naslutiti da će tada i proračuni biti veoma zahtjevni (i to je točno) pa ćemo u interaktivnoj računalnoj grafici morati aproksimirati dijelove izračunavanja osvjetljenja. Upravo o aproksimaciji ovisit će omjer realizma naših slika s izvršavanjem u stvarnom vremenu, pa tako nastaju lokalni i globalni modeli osvjetljenja.

Lokalni modeli osvjetljenja ne aproksimiraju samo svjetlost koja dolazi direktno s izvora svjeltosti dok ostale djelove aproksimira s konstantama. Globalni modeli osvjetljenja za razliku od lokalnih teže potpunoj fizikalnoj točnosti te uzimaju u obzir međudjelovanje površina koje nisu izvor svjetlosti, umjesto da ih aproksimiraju konstantama. Upravo jedan od globalnih modela osvjetljenja bit će detaljno opisan u ovom radu, a to je metoda isijavanja (engl. *Radiosity*).

Ovaj rad opisat će osnovnu metodu isijavanja i komentirati različita proširenja koja poboljšavaju osnovni model, ali da bi mogli opisati kako radi metoda isijavanja, prvo moramo točno definirati pojmove vezane uz svjetlost koji su nam važni. Tek tada ćemo moći izvesti univerzalnu jednadžbu iscrtavanja, koja obuhvaća sve fizikalne pojave. „Jačinom “ aproksijmacije te jednadžbe dobit ćemo lokalne i globalne modele, ovisno o našim potrebama.



Slika 1. Aproksimacija svjetlosti. Lijeva slika aproksimira najviše dok desna aproksimira najmanje.

2. Univerzalna jednadžba iscrtavanja

U ovom poglavlju izvest ćemo univerzalnu jednadžbu iscrtavanja koja karakterizira prijenos svjetlosti u sceni. Ograničit ćemo se samo na scene u kojima nema transmisije, odnosno postoji samo refleksija, iako generalna jednadžba uključuje obje pojave. Osim u najjednostavnijim slučajevima, jednadžba se ne može riješiti točno. Aproksimacijske metode zato su neophodno sredstvo. Ali da bi mogli izvesti jednadžbu, prvo se moramo upoznati s nekoliko pojmova koji su neophodni za shvaćanje jednadžbe.

2.1. Fizikalna svojsta svjetlosti

Kroz povijest se razvilo više interpretacija koje opisuju propagaciju svjetlosti kroz medij i interakciju svjetlosti s površinom. Najširа podjela je na klasičnu i kvantnu optiku. Klasična optika podijeljena je u dvije glavne grane: geometrijsku i fizikalnu optiku. Geometrijska optika tretira svjetlost kao skup svjetlosnih zraka koje se šire pravocrtno. Fizikalna optika opširnija je od geometrijske i tretira svjetlost kao elektromagnetsko zračenje te se pomoću nje mogu objasniti pojave kao što su difrakcija i interferencija. Međutim, svjetlost pokazuje i čestična svojstva pa tako kvantna optika proučava svjetlost upravo na toj razini, tretirajući svjetlost kao skup kvantiziranih "paketića", tzv. fotona, od kojih svaki posjeduje energiju $\boldsymbol{h} * \boldsymbol{f}$, no većina svjetlosnih pojava može biti objašnjena pomoću klasične optike na kojoj ćemo se i mi zadržati.

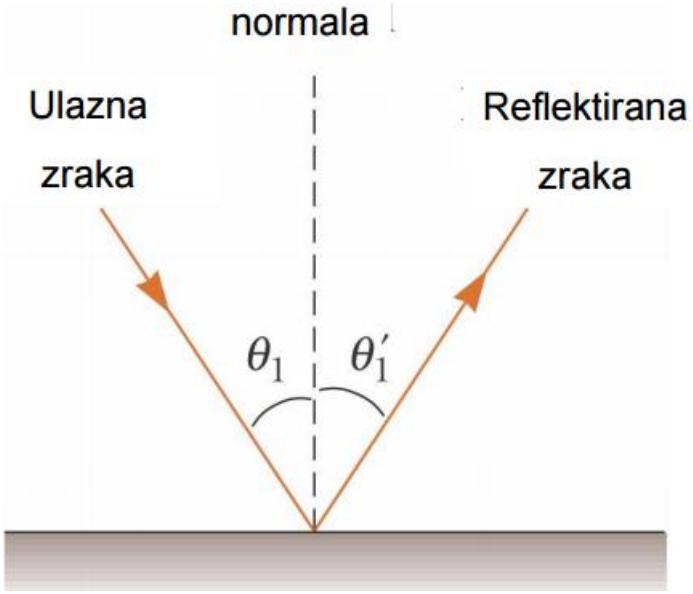
Za naše potrebe trebat ćemo samo svojstva koja se mogu opisati geometrijskom optikom. Geometrijska optika temelji se na 4 osnovna zakona:

1. Zakon o pravocrtnom širenju svjetlosti
2. Neovisnost svjetlosnih snopova
3. Zakon odbijanja (Refleksija)
4. Zakon loma (Refrakcija)

2.1.1. Refleksija

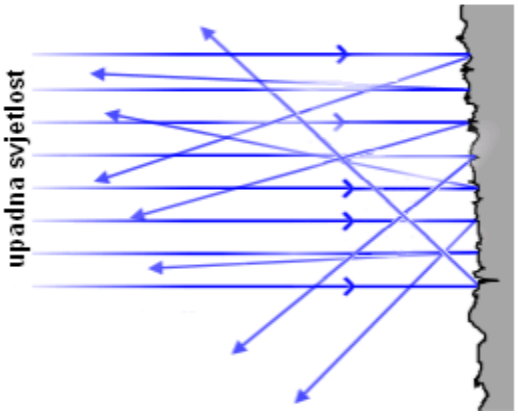
Refleksija je promjena smjera širenja svjetlosne zrake na granici dvaju sredstava. U stvarnosti postoji samo jedan tip refleksije, odnosno mikroskopski gledano refleksija je uvijek ista, no kad se radi o primjeni na realnim površinama postoje različiti tipovi refleksije. Kao i uvijek postoje dvije krajnosti i cijeli niz situacija koje se nalaze između njih. Te dvije krajnosti nazivaju se direktna ili zrcalna refleksija i difuzna ili raspršena refleksija.

Zrcalna refleksija događa se kada se svjetlosna zraka odrazi na glatkim površinama. Tada je kut koji upadna zraka zatvara s normalom površine jednak kutu koji reflektirana zraka zatvara s normalom površine.



Slika 2. Zrcalna refleksija

Za razliku od zrcalne refleksije, difuzna refleksija pojavljuje se na grubim površinama. U difuznoj refleksiji sve se reflektirane zrake i dalje ponašaju u skladu sa zakonom refleksije, ali neravnost površine rezultira različitim normalama kroz cijelu površinu. U tom slučaju, normale susjednih točaka više nisu paralelne. Pošto upadni kut ovisi o normali u svakoj točki površine, reflektirane zrake neće biti paralelne i raspršit će se u različitim smjerovima. Savršeno difuzno tijelo reflektira zrake u svim smjerovima, tako da reflektirane zrake svjetlosti oblikuju polukuglu iznad tijela.



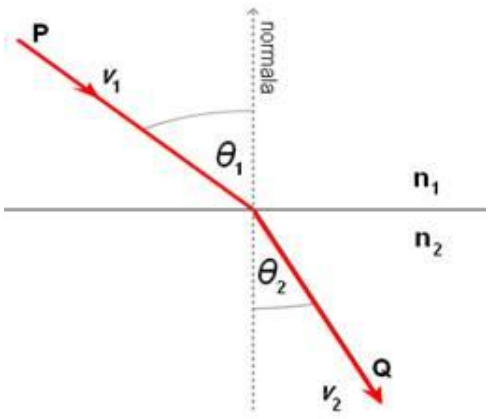
Slika 3. Difuzna refleksija

2.1.2. Refrakcija

Refrakcija je promjena smjera vala do koje dolazi zbog promjene njegove brzine širenja. Kolika će biti promjena smjera ovisi o refrakcijskom indeksu sredstava i kutu između zrake svjetlosti i normale površine koja razdvaja dva sredstva. Svako sredstvo ima različit refrakcijski indeks. Kut između normale i zrake koja napušta sredstvo naziva se upadni kut, a kut između normale i zrake koja ulazi u sredstvo naziva se kut refrakcije. Upadni kut i kut refrakcije povezani su Snellovim zakonom i vrijedi:

$$\frac{\sin(\alpha)}{\sin(\beta)} = \frac{v_1}{v_2} = \frac{n_2}{n_1}$$

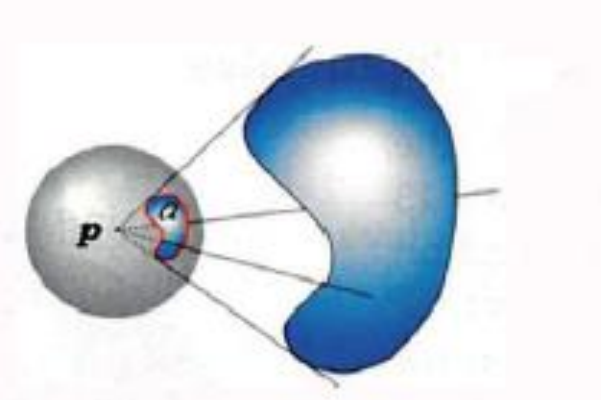
gdje su n_1 i n_2 refrakcijski indeksi pojedinih sredstava, a v_1 i v_2 brzine širenja vala u pojedinim sredstvima. Kako je indeks loma definiran kao $n = \frac{c}{v}$ zakon vrijedi [1].



Slika 4. Refrakcija svjetlosti

2.2. Radijancija (engl. *Radiance*)

Da bi smo mogli definirati „skup smjerova“ za upadnu svjetlost na trodimenzionalnu površinu, moramo prvo definirati što je to prostorni kut. Prostorni kut za 3D prostor isto je što i kut za 2D prostor. Najlakše objašnjenje prostornog kuta je kroz jednostavan primjer. Razmotrimo neki objekt koji je vidljiv iz neke točke **p** kao sto je prikazano na slici 6. Prvo postavimo jediničnu kuglu oko točke tako da se ona nalazi u centru. Zatim iz točke povučemo beskonačno mnogo linija tako da prolaze kroz vanjski rub objekta kojeg promatramo. Pravci koji izviru iz točke **p** i sijeku vanjski rub objekta također sijeku i jediničnu kuglu oko točke i stvaraju krivulju koja je na slici prikazana crvenim rubom. Površina te krivulje naspram površine kugle je prostorni kut. Mjerna jedinica prostornog kuta je steradijan i uobičajeno je označen s $\boldsymbol{\omega}$. Formula za izračunavanje prostornog kuta je $\boldsymbol{\omega} = \frac{\boldsymbol{A} * \mathbf{cos}(\boldsymbol{\theta})}{\boldsymbol{r}^2}$, gdje je A površina promatranog objekta, $\mathbf{cos}(\boldsymbol{\theta})$ kut između zrake iz točke **p** i normale u točki koju sječe na objektu, a $\boldsymbol{r}$ polumjer kugle [2].



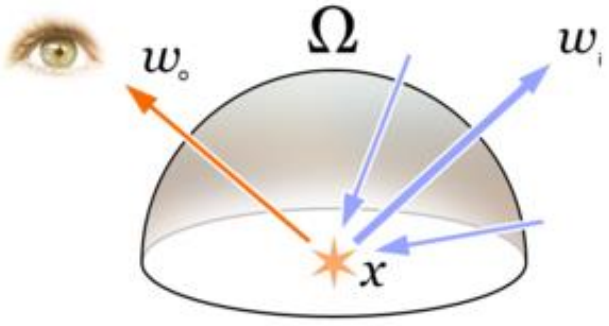
Slika 5. Prostorni kut

Sa znanjem prostornog kuta sad možemo opisati što je to radijancija (zračivost) i kako se mjeri. Razmotrimo funkciju **L** koja se zove radijancija. To je funkcija koja ovisi o vremenu, poziciji i smjeru. Ona bilježi beskonačno male karakteristike prijenosa svjetlosti u smislu da kada je integrirana u određenom vremenskom intervalu i određenom djelu površine okomitom na smjer dolaznih zraka i određenom prostornom kutu smjerova zraka, rezultat je totalna svjetlosna

energija koja stiže do površine iz određenih skupa smjerova u određenom vremenskom intervalu.

$$\boldsymbol{totalna\ ulazna\ energija} = \int_{t0}^{t1} \int_{P \in R} \int_{\omega \in \Omega} \boldsymbol{L(t, P, -\omega)|\omega \cdot n| d\omega dP dt}$$

Formula opisuje totalnu ulaznu energiju na ravnoj površini R s normalom $\boldsymbol{n}$ u određenom vremenskom intervalu. Treba obratiti pažnju na to da je smjer upadne zrake negativan jer prema konvenciji on pokazuje u suprotnom smjeru od onog što mi želimo, a mi želimo izračunati svjetlosnu energiju koja dolazi na površinu.



Slika 6. Prema konvenciji upadna zraka gleda iz površine prema izvoru svjetlosti

2.3 Funkcija distribucije refleksivnosti (engl. *BRDF*)

BRDF je funkcija koja nam govori koliko je svjetlosti reflektirano od neke površine. Ona ovisi o smjeru upadne i izlazne zrake te o poziciji točke za koju mjerimo reflektiranu svjetlost. Ako nam je zadan smjer upadne zrake i smjer izlazne zrake za određenu točku, tada je BRDF izračunat kao omjer svjetlosti koja napušta točku u smjeru izlazne zrake i svjetlosti koja dolazi u točku iz smjera upadne zrake.

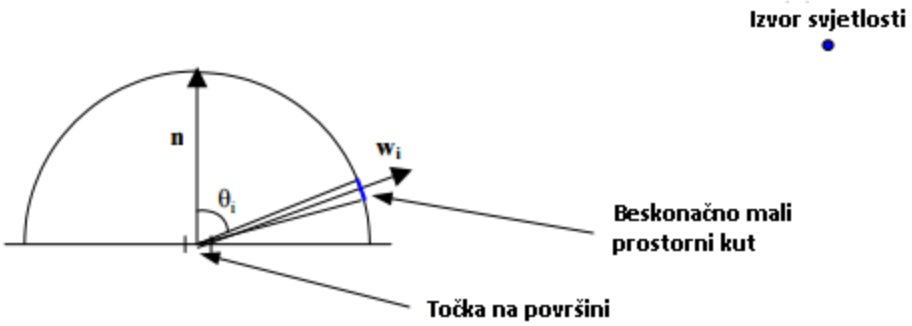
$$\boldsymbol{BRDF} = \frac{\boldsymbol{L_o}}{\boldsymbol{E_i}} \ [\boldsymbol{sr^{-1}}]$$

$\boldsymbol{L_o}$ - svjetlost koja napušta točku

$\boldsymbol{E_i}$ - svjetlost koja stiže u točku

Najlakše je objasniti što je to točno BRDF kroz primjer. Točku osvjetljava točkasto svjetlo. Količina svjetlosti koja dolazi iz smjera $\boldsymbol{\omega_i}$ proporcionalna je količini svjetlosti koja dolazi na prostorni kut. Pretpostavimo da izvor svjetlosti ima intenzitet

L_i. Pošto je prostorni kut beskonačno mali, možemo bez prevelikih gubitaka smatrati da je ravan. Tada je ulazna svjetlost koja stiže jednaka $\boldsymbol{L_i * \omega_i}$. Jedini problem koji nam je ostao je da je to ulazna svjetlost koja je okomita na prostorni kut, a ne na točku koja nas zanima. Da bi dobili uzlanu svjetlost koja je okomita na točku trebamo pomnožiti ulaznu svjetlost s kosinusom kuta između normale točke $\boldsymbol{n}$ i ulaznog smjera zrake $\boldsymbol{\omega_i}$. Time naš $\boldsymbol{E_i}$ postaje jednak $\boldsymbol{L_i * \omega_i * cos\theta}$. Pri čemu je $\boldsymbol{cos\theta = \omega_i * n}$.



Slika 7. Točka osvijetljena izvorom svjetlosti

Time naša konačna formula uz definiciju radijancije iz prošlog poglavlja i formalnog zapisa BRDF funkcije poprima sljedeći oblik:

$$\boldsymbol{f_r(P, \omega_i, \omega_o) = \frac{L(t, P, \omega_o)}{L(t, P, -\omega_i) cos\theta * d\omega_i}}$$

Iz formule je vidljivo da BRDF nije u granicama između [0,1] iako omjer izlazne i ulazne svjetlosti mora biti u tom intervalu, ali zbog kosinusa kuta može biti veći od 1 [3].

2.4. Univerzalna jednadžba iscrtavanja

Nakon što smo se upoznali sa svim potrebnim pojmovima, sada možemo vrlo lako izvesti univerzalnu jednadžbu iscrtavanja koja nam govori da je izlazna svjetlost jednaka zbroju dijela ulazne svjetlosti i emitirane svjetlosti ako je površina izvor svjetlosti.

$$\boldsymbol{L}_{\mathrm{o}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{o}}) = \boldsymbol{L}_{\mathrm{e}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{o}}) + \int_{\boldsymbol{\Omega}} \boldsymbol{f}_{\mathrm{r}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}}) * \boldsymbol{L}(\boldsymbol{P},-\boldsymbol{\omega}_{\mathrm{i}}) * (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}\mathrm{p})\boldsymbol{d\omega}_{\mathrm{i}}$$

$\boldsymbol{L}_{\mathrm{o}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{o}})$ - izlazna svjetlost iz točke $\boldsymbol{P}$ u smjeru $\boldsymbol{\omega_o}$

$\boldsymbol{L}_{\mathrm{e}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{o}})$ - emitirana svjetlost točke $\boldsymbol{P}$ u smjeru $\boldsymbol{\omega_o}$ (nije nula samo ako je točka izvor svjetlosti)

$\boldsymbol{f}_{\mathrm{r}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}})$ - BRDF

$\boldsymbol{L}(\boldsymbol{P},-\boldsymbol{\omega}_{\mathrm{i}})$ - upadna svjetlost u točku $\boldsymbol{P}$ iz smjera $\boldsymbol{-\omega_i}$

$\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_{\boldsymbol{p}}$ - kosinus kuta između smjera upadne zrake i normale točke $\boldsymbol{P}$

Treba još jednom napomenuti da je ovo univerzalna jednadžba iscrtavanja koja u obzir uzima samo reflekcijska svojstva. Postoji i općenitija jednadžba koja u obzir uzima i transmisijska svojstva [1].

3. Lokalni modeli osvjetljenja

Lokalni (direktni) modeli osvjetljenja za razliku od globalnih modela zanemaruju globalni učinak propagacije svjetlosti. U lokalnim modelima ukupna upadna svjetlost jednaka je zbroju svih zraka koje stižu iz izvora svjetlosti. Kao i svi modeli osvjetljenja, temelji se na univerzalnoj jednadžbi iscrtavanja koju smo izveli u prošlom poglavlju. Ako ponovno pogledamo tu jednadžbu vidjet ćemo da je najveći problem u drugom članu, odnosno integralu.

$$\int_{\boldsymbol{\omega}_{\mathrm{i}} \in \boldsymbol{S}^2_+(\boldsymbol{P})} \boldsymbol{f}_{\mathrm{r}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}}) * \boldsymbol{L}(\boldsymbol{P},-\boldsymbol{\omega}_{\mathrm{i}}) * (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_{\boldsymbol{p}})\boldsymbol{d\omega}_{\mathrm{i}}$$

Pošto lokalni modeli osvjetljenja zanemaruju doprinos svjetlosti iz svih smjerova, osim iz smjera izvora svjetlosti, tada se taj integral može svesti na sumu upadnih zraka iz smjera izvora svjetlosti.

$$\sum_{\substack{\boldsymbol{izvori} \\ \boldsymbol{svjetla}}} \boldsymbol{f}_{\mathrm{r}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}}) * \boldsymbol{L}(\boldsymbol{P},-\boldsymbol{\omega}_{\mathrm{i}}) * (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_{\boldsymbol{p}})\boldsymbol{d\omega}_{\mathrm{i}}$$

Upravo na ovoj aproksimaciji temelje se svi lokalni modeli osvjetljenja. Sada smo u mogućnosti opisati neke od najpoznatijih modela lokalnog osvjetljenja.

3.1. Phongov model osvjetljenja

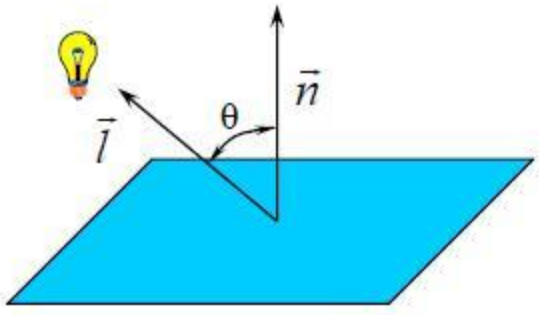
Phongov model empirijski je model osvjetljenja. Opisuje da je izlazna svjetlost bilo koje točke kombinacija difuzne, zrcalne i ambijentalne komponente. Pomoću opisa ovih komponenata moći ćemo izvesti ono što nam jedino nedostaje u našoj gornjoj formuli, a to je BRDF funkcija.

3.1.1. Difuzna komponenta

Kada zraka svjetlosti upada na neku površinu, intenzitet reflektirane svjetlosti uvelike ovisi o kutu između upadne zrake i normale te površine (slika 8) o čemu nam govori i Lambetov zakon:

$$\boldsymbol{I_d = I_i * k_d * \cos(\theta)}$$

gdje $\boldsymbol{I_i}$ predstavlja intenzitet točkastog izvora, a $\boldsymbol{k_d}$ empirijski koeficijent refleksije. $\boldsymbol{k_d}$ se nalazi u intervalu [0,1], a ovisi o valnoj duljini upadne svjetlosti. Kut $\boldsymbol{\theta}$ predstavlja kut između upadne zrake i normale površine. Treba naglasiti da pri proračunu oba vektora trebaju biti normalizirana. U slučaju da $\boldsymbol{\cos(\theta)}$ ispada negativan, za vrijednost ove komponente uzima se nula jer je točka nevidljiva za izvor.



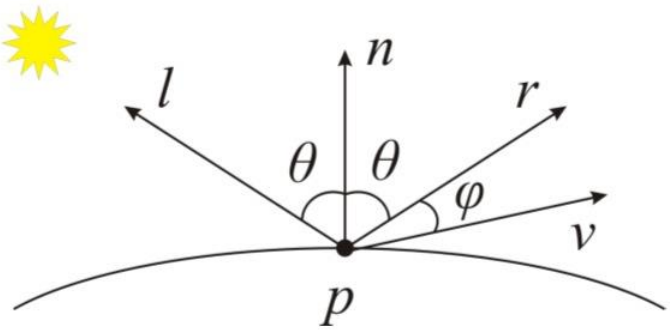
Slika 8. Difuzna komponenta

3.1.2. Zrcalna komponenta

Za razliku od difuzne komponente koja ne ovisi o promatraču, zrcalna komponenta je onaj dio koji dopire upravo do promatrača. Naime, ova komponenta funkcija je kuta između reflektirane zrake i zrake koja dopire do promatračeva oka.

$$\boldsymbol{I_s = I_i * k_s * (\cos(\varphi))^n}$$

Ovdje je $\boldsymbol{I_i}$ intenzitet točkastog izvora, $\boldsymbol{k_s}$ koeficijent refleksije ovisan o materijalu i nalazi se u intervalu [0,1], a kut $\boldsymbol{\varphi}$ kut između reflektirane zrake i zrake prema promatraču. Koeficijent $\boldsymbol{n}$ ovisi o grubоći površine i njenim reflektirajućim svojstvima, što je on veći, to je površina sličnija zrcalu, a od tuda i naziv zrcalna komponenta.



Slika 9. Zrcalna komponenta

3.1.3. Ambijentna komponenta

Ambijentna komponenta simulira interakciju svih točaka scene bez obzira da li su izvor svjetlosti. Ona je zaslužna za to da površine koje nisu vidljive izvoru svjetlosti ipak nisu potpuno crne nego su osvjetljenje. Naime, iako neka točka nije pod izravnim utjecajem svjetlosti, do nje dolaze zrake svijetlosti reflektirane od drugih površina. Radi jednostavnosti za ovu komponentu se uzima konstantna vrijednost:

$$\boldsymbol{I_g = k_a * I_a}$$

gdje je $\boldsymbol{k}_{\mathrm{a}}$ koeficijent u intervalu [0,1] i govori nam koliko će svjetlosti $\boldsymbol{I}_{\mathrm{a}}$ biti reflektirano. U stvarnosti ta vrijednost ima drugačiji iznos za svaku točku, a to možemo vidjeti u jednostavnim primjerima iz svakodnevnog života gdje do različitih mjesta dolaze svjetlosti različitog intenziteta pa su neka mjesta osvijetljena više od drugih.

3.1.4. Ukupan utjecaj

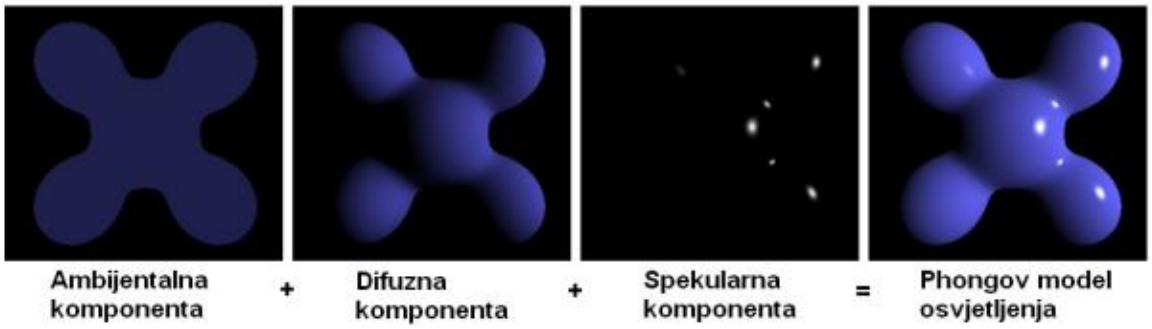
Sad kada imamo ukratko objašnjeno značenje svake komponente, možemo to spojiti u našu početnu jednadžbu za lokalni model osvjetljenja. Kao što je prije rečeno, jedina nepoznanica nam je BRDF funkcija, ali s novim saznanjem lako dolazimo do nje za Phongov model osvjetljenja.

$$\boldsymbol{f}_{\mathrm{r}}(\boldsymbol{P}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = \boldsymbol{k}_{\mathbf{d}} + \boldsymbol{k}_{\mathbf{s}} * \frac{(\vec{r} \cdot \vec{v})^{n}}{\vec{n} \cdot \vec{l}}$$

Sada uz promjenu notacije i dodavanja ambijentalne komponente jednadžbi dobivamo izlaznu svjetlost točke $\boldsymbol{P}$.

$$\boldsymbol{L}(\boldsymbol{P}, \boldsymbol{\omega}_{\mathrm{o}}) = \boldsymbol{I}_{\mathrm{a}} * \boldsymbol{k}_{\mathrm{a}} + \sum_{\substack{\boldsymbol{izvori} \\ \boldsymbol{svjetla}}} (\boldsymbol{k}\mathbf{d} + \boldsymbol{k}\mathbf{s} * \frac{(\vec{r} \cdot \vec{v})^{n}}{\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_{p}}) * \boldsymbol{L}(\boldsymbol{P}, -\boldsymbol{\omega}_{\mathrm{i}}) * (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_{p}) d\boldsymbol{\omega}_{\mathrm{i}}$$

U svemu tome jedino što je ostalo nerazjašnjeno je zašto drugi član u BRDF funkciji dijelimo s kosinusom kuta između upadne zrake i normale u točki $\boldsymbol{P}$. Pošto se u Phongovoj zrcalnoj komponenti nigdje ne spominje vektor između ulazne zrake i normale točke, on nebi smio postojati niti u našoj aproksimiranoj jednadžbi.

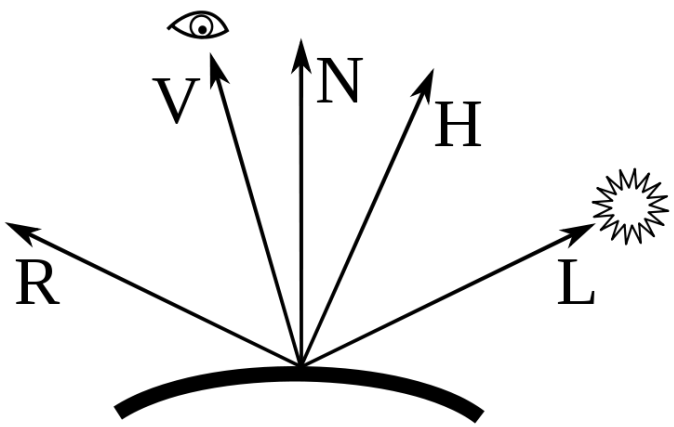


Slika 10. Phongov model osvjetljenja i njegove komponente

3.2. Blinn-Phongov model osvjetljenja

Jedina razlika između Blinn-Phongovog i Phongovo modela osvjetljenja je u zrcalnoj komponenti. U Blinn-Phongovoj zrcalnoj komponenti se umjesto kuta između reflektirane zrake i vektora točka-očiste koristi vektora na pola puta između očista i upadne zrake. Novi vektor označava se slovom $\boldsymbol{H}$, a računa se prema formuli:

$$\boldsymbol{H} = \frac{\boldsymbol{L} + \boldsymbol{V}}{|\boldsymbol{L} + \boldsymbol{V}|}$$



Slika 11. Blinn-Phongov model osvjetljenja

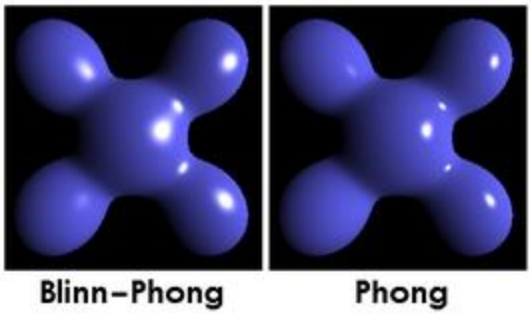
Sada naša BRDF funkcija izgleda malo drugačije, ali je i dalje dosta slična Phongovoj:

$$\boldsymbol{f}_{\mathrm{r}}(\boldsymbol{P}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = \boldsymbol{k}_{\mathbf{d}} + \boldsymbol{k}_{\mathbf{s}} * \frac{(\vec{\boldsymbol{n}} \cdot \vec{\boldsymbol{H}})^{\boldsymbol{n}}}{\vec{\boldsymbol{n}} \cdot \vec{\boldsymbol{l}}}$$

Posljedično naša nova aproksimirana jednadžba za izlaznu svjetlost iz točke $\boldsymbol{P}$ izgleda ovako:

$$\boldsymbol{L}(\boldsymbol{P}, \boldsymbol{\omega}_{\mathrm{o}}) = \boldsymbol{I}_{\mathrm{a}} * \boldsymbol{k}_{\mathrm{a}} + \sum_{\substack{\boldsymbol{izvori} \\ \boldsymbol{svjetla}}} (\boldsymbol{k}\mathbf{d} + \boldsymbol{k}\boldsymbol{s} * \frac{(\vec{\boldsymbol{n}} \cdot \vec{\boldsymbol{H}})^{\boldsymbol{n}}}{\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_{\boldsymbol{p}}}) * \boldsymbol{L}(\boldsymbol{P}, -\boldsymbol{\omega}_{\mathrm{i}}) * (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_{\boldsymbol{p}}) \boldsymbol{d}\boldsymbol{\omega}_{\mathrm{i}}$$

Blinn-Phongova metoda je efikasnija od Phongove jer se izbjegava računanje reflekcijskog vektora $\vec{r}$.



Slika 12. Razlika između Blinn-Phongovog i Phongovog modela osvjetljenja

4.Globalni modeli osvjetljenja

Veliki broj površina u stvarnom svijetu prima većinu ili svu svjetlost od drugih reflektirajućih površina koje nisu izvori svjetlosti. Takvo osvjetljavanje je poznatije pod imenom globalno ili indirektno osvjetljenje.

Slika 13. Primjer globalnog osvjetljenja

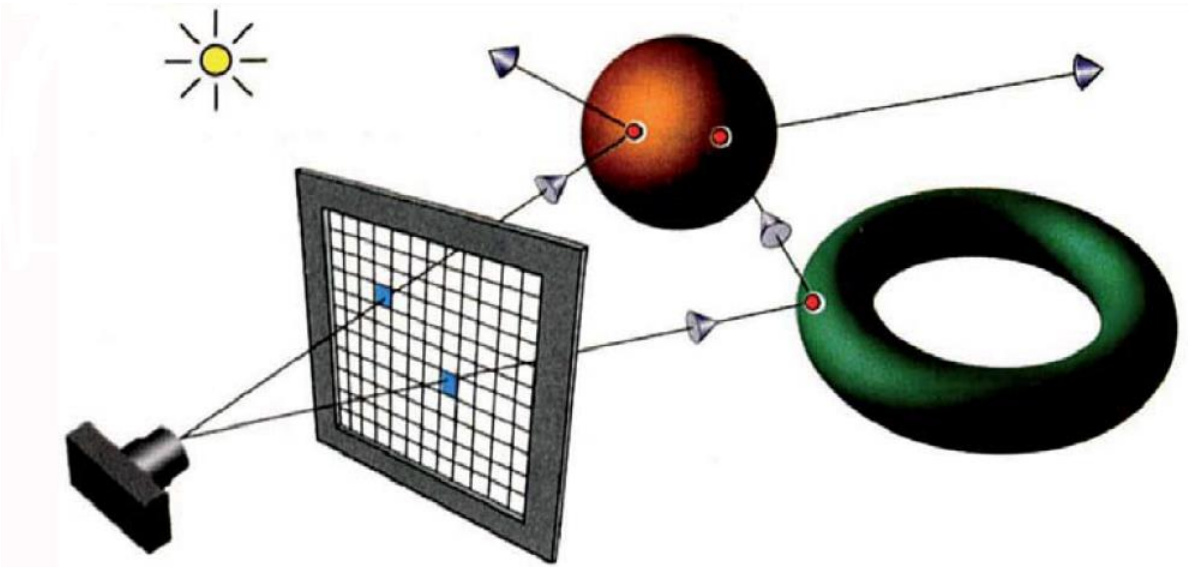
Na slici 13 je dobar primjer globalnog osvjetljenja jer se može vidjeti da je stražnji dio kocke i sve što se nalazi iza nje nevidljivo svjetlosti koja dolazi od prozora, ali ipak je taj dio osvijetljen dovoljno da se jasno vidi što se nalazi u sceni. Valja napomenuti da bi bez globalnog osvjetljenja dio površine iza kocke bio potpuno crn, što se naravno protivi našem stvarnom poimanju.

Sada se već jasno nazire da su modeli globalnog osvjetljenja temeljeni na stvarnom fizikalnom modelu svjetlosti koji smo izveli u 2. poglavlju. Dakako, ni globalni modeli osvjetljenja u računalnoj grafici ne koriste jednadžbu bez aproksimacija, ali rade to u puno manjoj mjeri nego lokalni modeli osvjetljenja. U ovom poglavlju opisat ćemo nekoliko algoritama koji simuliraju globalno osvjetljenje, a koji se koriste u računalnoj grafici. Također, u idućem ćemo poglavlju opisati algoritam koji je centralni dio ovog cijelog rada, a to je metoda isijavanja (engl. *Radiosity*).

4.1. Algoritam praćenja zrake (engl. *Ray tracing*)

U ovom potpoglavlju će biti ukratko objašnjeno što je to algoritam praćenja zrake. Pošto je to vrlo opsežna i zahtjevna tema zadržat ćemo se samo na pojmovima potrebnima za daljnju upotrebu i nećemo se upuštati previše u teoriju.

Algoritam praćenja zrake tehnika je računalne grafike koja kreira slike „ispaljivanjem“ zraka. Kao što je prikazano na slici 14 potrebna nam je kamera, prozor na kojem se nalaze pikseli i scena.

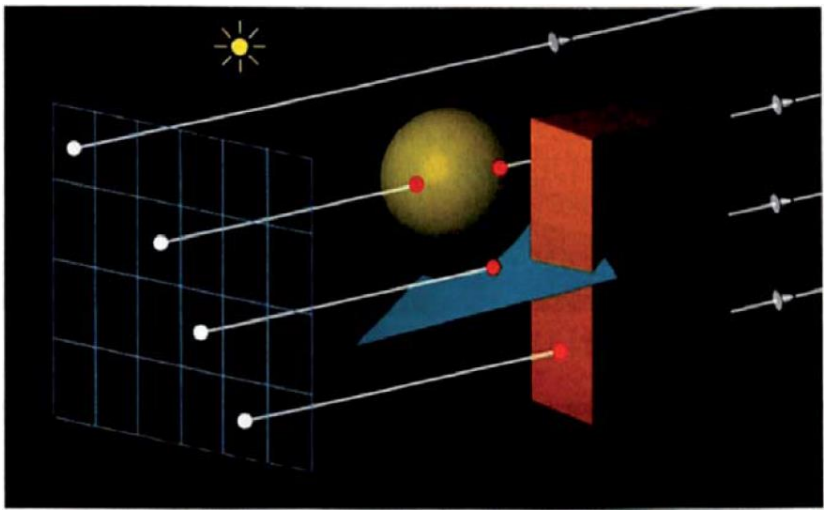


Slika 14. Algoritam praćenja zrake

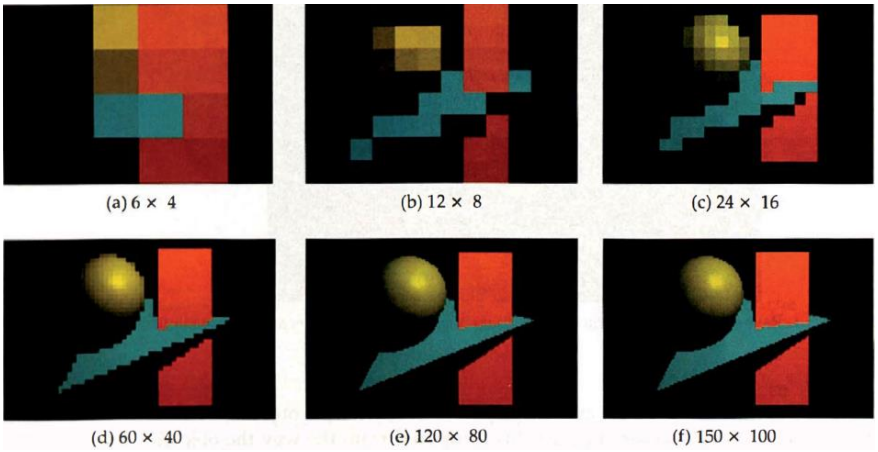
Jednostavan algoritam praćenja zrake mora izvršavati sljedeće operacije, a njih ćemo prikazati pomoću pseudokoda:

```
Definiraj neke objekte;
Specificiraj materijale za pojedine objekte;
Definiraj izvore svjetlosti;
Definiraj prozor koji se sastoji od piksela;
Za svaki piksel
      Ispali zraku iz kamere kroz centar piksela prema sceni;
      Izračunaj najbližu točku koju pogađa zraka u sceni;
      Ako je zraka pogodila neki objekt
            Iskoristi materijal objekta i izvore svjetlosti za izračuna
boje piksela;
      Inače
            Postavi boju piksela na crno;
```

Pošto je algoritam praćenja zrake poznat po svojoj mogućnosti dočaravanja vrlo realnih slika, očito treba napraviti neke preinake u našem algoritmu jer kada bi imali prozor s ukupno 9 piksela, vrlo teško bi mogli dočarati da se u sceni nalazi kugla. Slika 15 i 16 najbolje to dočaravaju. Slika 15 prikazuje scenu i i prozor piksela, a slika 16 nekoliko različitih razlučivosti prozora piksela.

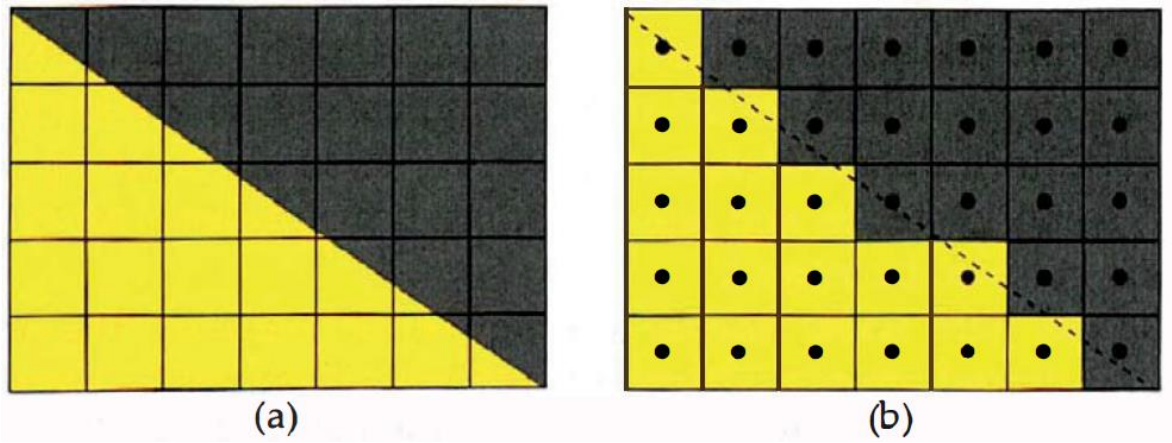


Slika 15. Uzorkovanje scene



Slika 16. Različite razlučivosti

Nakon ovog primjera prirodno je pomisliti da bi povećanjem broj piksela riješili sve naše probleme, ali nažalost to nije slučaj jer je naš algoritam vrlo podložan alias-efektu koji nastaje kada scena nije dovoljno uzorkovana. To se najbolje očituje u pojavi „stepenica“ tamo gdje bi zapravo trebala biti ravna linija. Slika 17 prikazuje pojavu alais-efekta. Slika 17.a pokazuje nam kako bi slika trebala izgledati, a slika 17.b kako izgleda slika uzorkovana sa samo jednim uzorkom po pikselu.

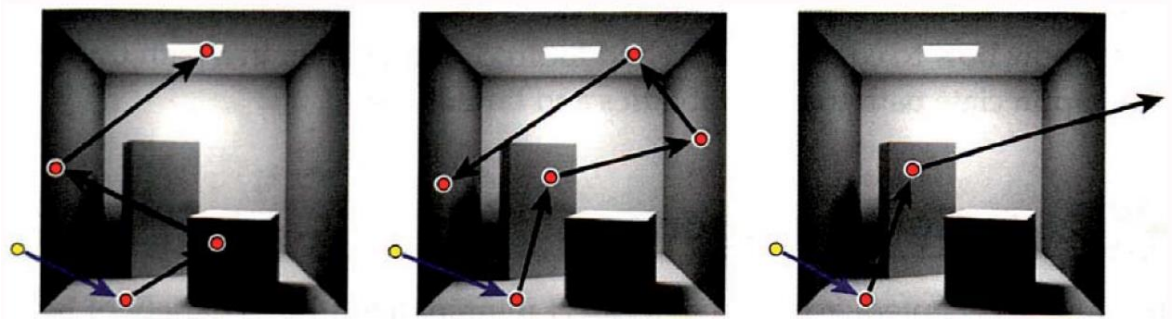


Slika 17. Alias-efekt kod uzorkovanja

Sada je jasno da nam je potrebno vise uzoraka po pikselu koji se tada zbroje i podjele sa brojem zraka po pikselu. Ovaj kratki uvod bio nam je potreban za ostatak ovog poglavlja jer se dalje opisani algoritmi globalnog osvjetljenja više ili manje temelje na algoritmu praćenja zrake [2].

4.2. Praćenje puta zrake (engl. *Path tracing*)

Praćenje puta zrake je konceptualno vrlo jednostavan algoritam i koristi se za izračunavanje direktnog i indirektnog osvjetljenja u sceni. Valjda napomenuti da za praćenje puta zrake nužno u sceni mora postojati površinsko svjetlo, pošto u grafici postoje točkasti i direkcijski izvori svjetla koji nemaju površinu nego su samo grupa aproksimacija realnih izvora svjetlosti. Slika 18 pokazuje nam osnovu ideju praćenje puta zrake.



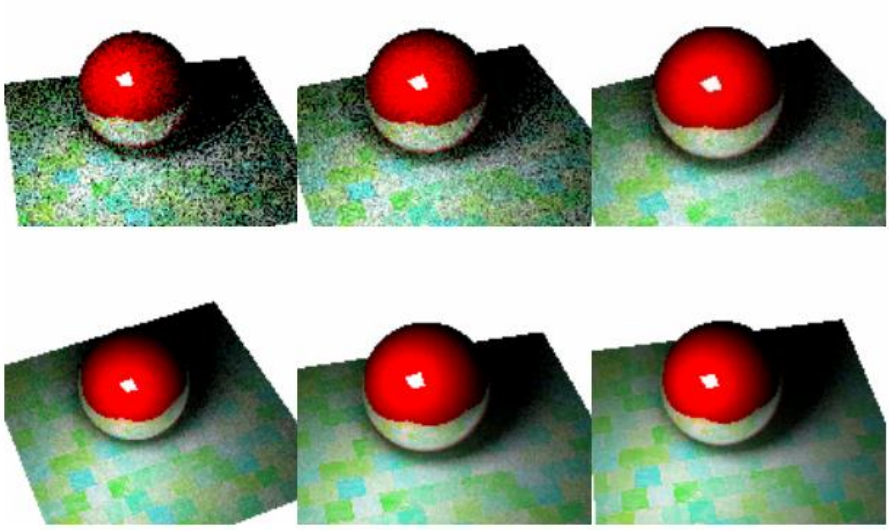
Slika 18. Praćenje puta zrake

Zraka se ispaljuje iz kamere u scenu i prati se sve dok ne pogodi izvor svjetlosti, napusti scenu ili zbog nekih drugih okolnosti. U svakoj točki koju zraka pogodi mora se izračunati koji će biti smjer reflektirane zrake. To se utvrđuje pomoću funkcije gustoće vjerojatnosti od pojedine reflektivne distribucijske funkcije (engl. *BRDF*).

Jako velika prednost algoritma praćenja puta zrake je da nije ograničen na određene materijale kao neki algoritmi globalnog osvjetljenja, što znači da u sceni može postojati potpuno difuzna i potpuno zrcalna površina, uz sve moguće vrste između. To dozvoljava algoritmu da simulira sve vrste prijenosa svjetlosti između površina tretirajući sve materijale kao reflektivne u smislu da svaka točka koju pogodi zraka reflektira novu zraku i prati ju tako da nastavlja put. Iznimka su jedino emitirajući materijali koji u ovom slučaju nemaju reflektivna svojstva. Primarna zraka koja je „ispaljena“ iz kamere i sve reflektirane zrake koje se dalje razvijaju od nje nazivaju se put zraka. Algoritam se uobičajeno implementira pomoću rekurzije, a to je ujedno i još jedan uvjet da se obustavi praćenje određene zrake jer se ne može pratiti zraka u beskonačno.

Jedna napomena koju treba imati u vidu zbog mogućnosti mješanja algoritama praćenja zrake i algoritma praćenja puta zrake. Oba algoritma imaju jako puno zajedničkih stvari, ali s razlikom da algoritam praćenja zrake može reproducirati globalno osvjetljenje. Suštinska razlika je u trenutku kada zraka koja kreće iz kamere udari neki objekt u sceni. U algoritmu praćenja zrake tada se ispaljuje zraka iz te točke prema svakom od izvora svjetlosti i nakon toga određuje izlazna svjetlost te točke, dok algoritam praćenja puta zrake ispaljuje zraku koja ne mora nužno biti usmjerena prema izvoru ili izvorima svjetlosti.

Naravno, postoje i loše strane algoritma. To se može vidjeti u nekim slučajevima kada su izvori svjetlosti u sceni puno manji u usporedbi s ostatkom scene. U tim situacijama algoritam je posebice neefikasan pošto je vjerojatnost pogađanja svjetlosti u tom slučaju vrlo mala. S malim brojem uzoraka po pikselu, u slici mogu nastati takozvani šumovi, a velik broj uzoraka po pikselu može izazvati dugo vrijeme izračunavanja. Za kraj ćemo dati predodžbu koliko izgled slike ovisi o broju uzoraka po pikselu [4].



Slika 19. 100, 250, 1000, 2500, 5000, 10000 uzoraka po pikselu

4.3. Preslikavanje fotona (engl. *Photon mapping*)

Osnovna ideja algoritma preslikavanja fotona vrlo je jednostavna. Ovaj algoritam razdvaja podatke o osvjetljenju od geometrije i ti podaci se spremaju u posebnu strukturu podataka nazvanu foton mapa. Metoda se sastoji od dvije faze.

U prvom djelu izgrađuje se fotonska mapa praćenjem fotona iz svakog izvora svjetlosti. U drugom djelu iscrtava se scena korištenjem informacija pohranjenih u fotonskoj mapi. Prednosti algoritma su da ne zahtjeva mreže trokuta, također metoda može rukovati s nedifuznim površina i kaustikom.

4.3.1. Prva faza: Emitiranje svjetlosti i raspršivanje fotona

Prva faza algoritma sastoji se od dva koraka: emitiranja svjetlosti i raspršivanja fotona. U prvom koraku generiraju se fotoni i „ispaljuju“ u scenu. Izvor svjetlosti s većim intenzitetom proizvest će više fotona, a smjer svakog fotona nasumično je izabran na osnovi vrste izvora svjetlosti. Kada foton udari objekt, on može biti reflektiran, transmitirani ili apsorbirani. Ako foton udari zrcalni objekt (npr. ogledalo), onda će se on reflektirati sa svojim intenzitetom pomnoženim s reflektirajućim indeksom tog objekta. U drugom slučaju, ako foton udari difuznu površinu, tada je on pohranjen u fotonsku mapu i reflektirani. Smjer od tog „difuzno“ reflektiranog fotona je nasumično izabran vektor iznad točke udara s vjerojatnošću proporcionalnom kosinusu kuta s normalnom. To se može implementirati pomoću tehnike „ruskog ruleta“.

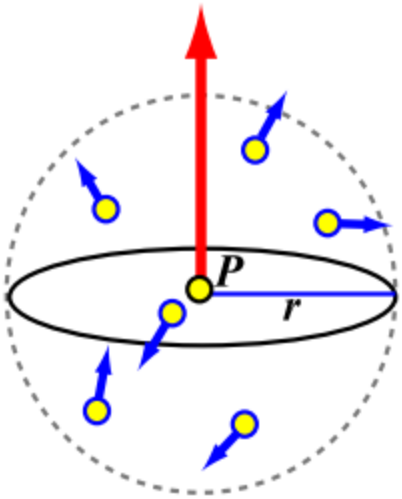
Tehnika „ruskog ruleta“ uklanja nevažne fotone i osigurava istu snagu fotona. U slučaju zrcalnog objekta, nasumični broj je generirani $\boldsymbol{q} \in [0,1]$ i ako je $\boldsymbol{q} \in [0, f]$ gdje je $\boldsymbol{f}$ reflekcijski indeks, tada je foton reflektiran, inače je apsorbiran. Ista stvar vrijedi i za difuznu površinu.

Informacija o točki udara fotona pohranjena je u fotonskoj mapi, koja se najčešće implementira kao kd-stablo. Svaki čvor stabla pohranjuje informaciju koordinate točke udara u sceni, intenzitet boja, ulazni smjer fotona i druge važne informacije.

4.3.1. Druga faza: Utvrđivanje osvjetljenja i iscrtavanje

Druga faza iscrtava scenu uz pomoć fotonske mape izgrađene u prvoj fazi. Tradicionalna tehnika praćenja zrake korištena je „ispaljivanjem“ zraka iz kamere.

Kada zraka pogodi točku $\boldsymbol{P}$ na površini, informacije o osvjetljenju susjednih fotona prikupljenih i pohranjenih u fotonskoj mapi iz prve faze bit će dodani informaciji o osvjetljenju prikupljenom iz algoritma praćenja zrake u točki $\boldsymbol{P}$. Uzmimo da je N normala u točki $\boldsymbol{P}$ i $\boldsymbol{r} > 0$. Uzimamo u obzir sve fotone u kugli $\boldsymbol{S}(\boldsymbol{P}, \boldsymbol{r})$ sa središtem u točki $\boldsymbol{P}$ i radijusom $\boldsymbol{r}$ (slika 20).



Slika 20. Utvrđivanje osvjetljenja

Ne pridonosi svaki foton u kugli osvjetljenju u točki $\boldsymbol{P}$. Zapravo, foton sa smjerom $\boldsymbol{d}$ može doprinjeti osvjetljenju samo ako je $\boldsymbol{d} \cdot \boldsymbol{N} > 0$. Inače, ako je umnožak manji ili jednak nula, njegov smjer ide unutar površine. Ako foton ne doprinosi osvjetljenju, on će biti ignoriran. Osvjetljenje kojem doprinosi jedan foton s ulaznim smjerom je:

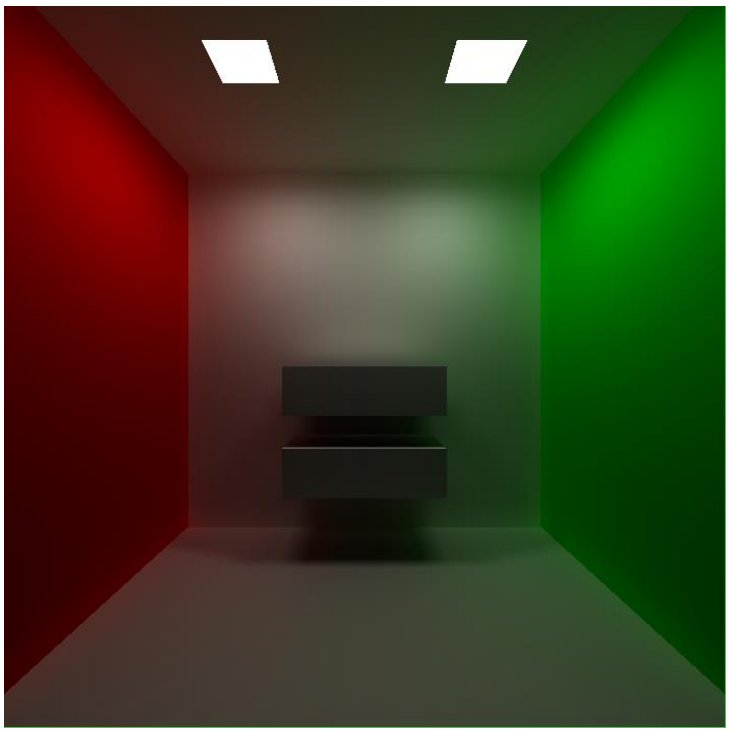
$$\boldsymbol{intenzitet} * (\boldsymbol{d} \cdot \boldsymbol{N}) * \boldsymbol{difuzni\ faktor}$$

Pretpostavimo da je suma svih doprionsa jednaka $\boldsymbol{s}$. Osvjetljenje u točki $\boldsymbol{P}$ je $\frac{\mathbf{s}}{\boldsymbol{\pi} * \mathbf{r}^2}$, gdje je $\boldsymbol{\pi} * \boldsymbol{r}^2$ površina kružnice kugle $\boldsymbol{S}$. Ukratko, boja u točki $\boldsymbol{P}$ je suma toga osvjetljenja i osvjetljenja izračunatog pomoću algoritma praćenja zrake [5].

5. Metoda isijavanja (engl. *Radiosity*)

Za razliku od algoritama navedenih u prijašnjem poglavlju koji se temelje na osnovno principu simulacije zakonitosti geometrijske optike, metoda isijavanja

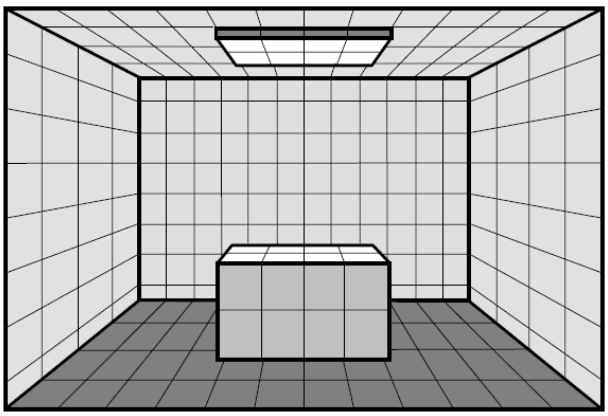
temelji se na simulaciji toplinskog zračenja. Metoda isijavanja je metoda koja osigurava rješenja za difuzne interakcije u zatvorenom prostoru. Metoda nije vezana uz položaj kamere i jednom kada se izračunaju intenziteti svjetlosti u prostoriji za svaku točku ona je neovisna o kameri. Rješenje se temelji na podjeli površina na poligone gdje je intenzitet svjetlosti konstantan. Zbog toga je teško uključiti zrcalne refleksije. Jedna od glavnih prednosti upravo je neovisnost o položaju kamere, ali treba imati na umu da je to i najveći problem pošto je cijena procesiranja vrlo visoka. Također, to je prvi algoritam globalnog osvjetljenja koji je uveo koncept preljeva boja. To znači ako imamo crveni zid i bijeli strop, rub stropa uz crveni zid bit će obojen svijetlocrvenom bojom (slika 21). U ovom poglavlju bit će detaljno opisan rad metode isijavanja. Prikazat ćemo matematički izvod i kako se izračunavaju pojedine komponente.



Slika 21. Preljev boja

5.1. Osnovna ideja

Osnovna ideja kao i kod većine algoritama globalnog osvjetljenja je jednostavna. Prvi korak je stvoriti scenu. Nakon što je scena stvorena, cijela scena dijeli se na poligone (engl. *Patch*).



Slika 22. Scena podijeljena na poligone

Sljedeći je korak izračunati između svakog para poligona faktor vidljivosti koji nam govori koliko dobro jedan poligon vidi drugog. Poligoni koji su daleko jedan od drugog ili se vide pod jako velikim kutom imat će manji faktor vidljivosti. Ako se poligoni ne vide direktno, njihov faktor će biti nula. Nakon što se izračunaju svi faktori, može se krenuti na izračunavanje reflektiranog zračenja za pojedini poligon:

$$\boldsymbol{B}_{\mathrm{i}} = \boldsymbol{E}_{\mathrm{i}} + \boldsymbol{R}_{\mathrm{i}} \int_{\boldsymbol{j}} \boldsymbol{B}_{\boldsymbol{j}} \boldsymbol{F}_{\boldsymbol{ij}}$$

gdje je:

$\boldsymbol{B}_{\mathrm{i}}$ - ukupna izlazna svjetlost poligona $\boldsymbol{i}$

$\boldsymbol{E}_{\mathrm{i}}$ - emitirana svjetlost poligona $\boldsymbol{i}$

$\boldsymbol{R}_{\mathrm{i}}$ - mjera reflektivnosti poligona $\boldsymbol{i}$

$\boldsymbol{B}_{\boldsymbol{j}}$ - ukupna izlazna svjetlost poligona $\boldsymbol{j}$

$\boldsymbol{F}_{\boldsymbol{ij}}$- faktor vidljivosti poligona $\boldsymbol{j}$ s poligona $\boldsymbol{i}$

Uz pretpostavku da su svi poligoni savršeno difuzni, svjetlost se odbija jednoliko u svim smjerovima i lako možemo provesti diskretizaciju. U tom slučaju gornji integral prelazi u sumu:

$$\boldsymbol{B}_{\mathrm{i}} = \boldsymbol{E}_{\mathrm{i}} + \boldsymbol{R}_{\mathrm{i}} \sum_{\boldsymbol{j}=\boldsymbol{1}}^{\boldsymbol{n}} \boldsymbol{B}_{\boldsymbol{j}} \boldsymbol{F}_{\boldsymbol{ij}}$$

gdje je $\boldsymbol{n}$ broj poligona u sceni.

Pretpostavka koju smo napravili da pojednostavimo formulu ujedno je i jedna od temeljnih pretpostavki metode isijavanja. To znači da ona radi samo sa savršeno difuznim poligonima, za razliku od primjerice algoritma praćenja puta zrake koji radi sa bilo kakvim poligonima. Metoda isijavanja primarno je pretpostavljena za rad s površinskim svjetlima koja imaju emitirajuća svojstva i kod njih jedino član $\boldsymbol{E}_i$ neće biti jednak nuli, ovo nam ujedno olakšava stvar jer se i izvori svjetlosti mogu promatrati kao i sve ostale površine samo s određenim emitiranim zračenjem.

Nakon kratkog uvoda možemo napisati pseudokod za osnovni algoritam i dati kratki opis kako točno radi.

```
Kreiraj scenu;
Podjeli površine u približno jednake manje površine(poligone);
Inicijaliziraj poligone;
Za svaki poligon u sceni
      Ako je poligon izvor svjetlosti
            Emitirajuća svjetlost poligona = neka količina svjetlosti
koju emitira;
      Inače
            Emitirajuća svjetlost poligona = 0;
      Izlazno zračenje poligona = emitirajuća svjetlost poligona
Sve dok nije zadovoljen neki uvjet
      Prikupi svjetlost od svih poligona iz scene
            Za svaki poligon
                  Ulazna svjetlost = prikupi svjetlost koja dolazi od
svih ostalih                                    poligona;
      Izračunaj izlaznu svjetlost za poligon
            Za svaki poligon
                 I = ulazna svjetlost sa svih poligona
                  R = mjera refleksivnosti poligona
                  E=emitirajuća svjetlost poligona
                  Ukupna izlazna svjetlost = I*R + E
```

Kod je razumljiv sam po sebi, ali nameće se pitanje koji je to uvjet koji se mora zadovoljiti. Postoje dva uvjeta, prvi je da sami odredimo broj iteracija, a drugi se temelji na konvergenciji koju također mi određujemo. Konvergencija nam govori u kojoj mjeri se izlazna svjetlost poligona povećala u odnosu na prošlu iteraciju. To znači ako je promjena manja od nekog broja kojeg smo odredili, petlja se prekida jer je tada promjena svjetlosti toliko malena da se više ne može primijetiti. To se naravno temelji na stvarno principu svjetlosti koja se odbija od poligona beskonačno puta, ali svaki puta kada udari neki poligon ona se ne reflektira u potpunosti, nego se njezina energija raspršuje na reflektiranu, transmitiranu i apsorbiranu.

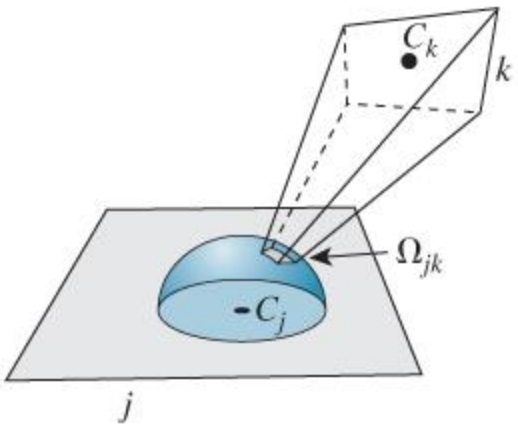
Također, iz koda se može vidjeti da ako postavimo uvjet petlje na samo jednu iteraciju dobit ćemo čisto lokalno osvjetljenje. Pošto su svi poligoni bez početne svjetlosti osim izvora, u prvom prolazu svi će poligoni primiti svjetlost samo od izvora. Tek se kasnijim iteracijama dobiva model globalnog osvjetljenja jer se tada ulazna svjetlost poligona sastoji ne samo od izvora nego i od ostalih poligona.

5.2. Matematička formulacija

U uvodu smo rekli da su modeli globalnog osvjetljenja aproksimacije univerzalne jednadžbe iscrtavanja koju smo izveli u drugom poglavlju. Jednadžba nam govori kolika je izlazna svjetlost koju prima neka točka od svoje okoline. Da bi mogli izvesti jednadžbu koja nam to govori kako se računa izlazna svjetlost za točku metodom isijavanja, moramo krenuti s aproksimacijom univerzalne jednadžbe iscrtavanja koja, prisjetimo se, izgleda ovako:

$$\boldsymbol{L}_{\mathrm{o}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{o}}) = \boldsymbol{L}_{\mathrm{e}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{o}}) + \int_{\boldsymbol{\omega}_{\mathrm{i}}\epsilon\boldsymbol{S}^2_+(\boldsymbol{n}_j)} \boldsymbol{f}_{\mathrm{r}}(\boldsymbol{P},\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}}) * \boldsymbol{L}(\boldsymbol{P},-\boldsymbol{\omega}_{\mathrm{i}}) * (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}_j)\boldsymbol{d}\boldsymbol{\omega}_{\mathrm{i}}$$

Ova jednadžba razlikuje se u odnosu na onu u poglavlju dva samo u notaciji normale točke $\boldsymbol{P}$. Tako je napravljeno jer je normala točke $\boldsymbol{P}$ ujedno i normala cijelog poligona pošto je poligon dio neke ravnine, a kako smo do sada te poligone označavali sa $\boldsymbol{i}$ i $\boldsymbol{j}$, u ovom poglavlju ćemo ih označavati sa $\boldsymbol{j}$ i $\boldsymbol{k}$ tako da se ne pomiješa oznaka za dolazni smjer i normalu poligona.



Slika 23. Poligon k vidljiv je s poligona j. Kad je projiciran na polukuglu oko točke C_k, dobivamo prostorni kut Ω_{jk}

Sada ćemo napraviti supstituciju $B_j = \frac{L(P,\omega_o)}{\pi}$. Pošto je $L(P,\omega_o)$ neovisan o izlaznom smjeru (zbog savršeno difuzne površine koja raspršuje izlazne zrake jednako u svim smjerovima), član B_j nema smjer izlazne zrake kao parametar. Pošto smo cijelu jednadžbu podijelili sa π trebamo napraviti i supstituciju za ostale članove jednadžbe pa tako emitirajući član označavamo s E_j gdje je $E_j = \frac{L_e(P,\omega_o)}{\pi}$, a $f_r(P,\omega_i,\omega_o) = \frac{\rho_j}{\pi}$. Sada naša nova jednadžba izgleda ovako:

$$\pi * B_j = \pi * E_j + \frac{\rho_j}{\pi} \int_{\omega_i \epsilon S^2_+(n_j)} L(P,-\omega_i) * (\omega_i \cdot n_j) d\omega_i$$

Integral preko svih smjerova polukugle može biti diskretiziran u sumu smjerova za svaki prostorni kut Ω_{jk}, pošto svjetlost na poligon j mora stići s nekog poligona k pa tada jednadžba prelazi u sumu integrala:

$$\pi * B_j = \pi * E_j + \frac{\rho_j}{\pi} \sum_k \int_{\omega_i \epsilon \Omega_{jk}} L(P,-\omega_i) * (\omega_i \cdot n_j) d\omega_i$$

Ulazna svjetlost je zapravo izlazna svjetlost poligona k pa prema tome možemo pisati $\pi * B_k$.

Primijenimo sada ovu novu supstituciju i presložimo malo nasu jednadžbu:

$$\pi * B_j = \pi * E_j + \frac{\rho_j}{\pi} \sum_k \int_{\omega_i \epsilon \Omega_{jk}} \pi * B_k * (\omega_i \cdot n_j) d\omega_i$$

$$\boldsymbol{\pi} * \boldsymbol{B_j} = \boldsymbol{\pi} * \boldsymbol{E_j} + \frac{\boldsymbol{\rho_j}}{\boldsymbol{\pi}} * \boldsymbol{\pi} \sum_{\boldsymbol{k}} \boldsymbol{B_k} * \int_{\boldsymbol{\omega_i} \in \boldsymbol{\Omega_{jk}}} (\boldsymbol{\omega}_\mathrm{i} \cdot \boldsymbol{n_j}) \boldsymbol{d\omega}_\mathrm{i}$$

$$\boldsymbol{\pi} * \boldsymbol{B_j} = \boldsymbol{\pi} * \boldsymbol{E_j} + \boldsymbol{\rho_j} * \boldsymbol{\pi} \sum_{\boldsymbol{k}} \boldsymbol{B_k} * \frac{\mathbf{1}}{\boldsymbol{\pi}} * \int_{\boldsymbol{\omega_i} \in \boldsymbol{\Omega_{jk}}} (\boldsymbol{\omega}_\mathrm{i} \cdot \boldsymbol{n_j}) \boldsymbol{d\omega}_\mathrm{i}$$

Dijeljenjem sa $\boldsymbol{\pi}$:

$$\boldsymbol{B_j} = \boldsymbol{E_j} + \boldsymbol{\rho_j} \sum_{\boldsymbol{k}} \boldsymbol{B_k} * \frac{\mathbf{1}}{\boldsymbol{\pi}} * \int_{\boldsymbol{\omega_i} \in \boldsymbol{\Omega_{jk}}} (\boldsymbol{\omega}_\mathrm{i} \cdot \boldsymbol{n_j}) \boldsymbol{d\omega}_\mathrm{i}$$

Koeficijent uz $\boldsymbol{B_k}$ unutar sume naziva se faktor vidljivosti za $\boldsymbol{F_{jk}}$. Sada naša konačna jednadžba za metodu isijavanja izgleda ovako:

$$\boldsymbol{B_j} = \boldsymbol{E_j} + \boldsymbol{\rho_j} \sum_{\boldsymbol{k}} \boldsymbol{F_{jk}} * \boldsymbol{B_k}$$

Prije nego je probamo riješiti, prvo moramo objasniti što su to faktori vidljivosti i kako se oni izračunavaju [1].

5.3. Faktori vidljivosti

Sada ćemo riješiti problem faktora vidljivosti. Faktori vidljivosti su jedan od najvećih problema s kojima se susrećemo prilikom implementacije algoritma. Na sreću, postoje algoritmi koji olakšavaju izračun tih faktora.

Prvo ćemo izvesti formulu za pojednostavljeni izračun faktora vidljivosti. Pogledajmo formulu koju smo izveli u poglavlju 5.2.

$$\frac{\mathbf{1}}{\boldsymbol{\pi}} * \int_{\boldsymbol{\omega_i} \in \boldsymbol{\Omega_{jk}}} (\boldsymbol{\omega}_\mathrm{i} \cdot \boldsymbol{n_j}) \boldsymbol{d\omega}_\mathrm{i}$$

Ako pretpostavimo da su sve zrake s poligona $\boldsymbol{k}$ na poligon $\boldsymbol{j}$ jednake, tada vidimo da se vektor $\boldsymbol{\omega}_\mathrm{i}$ može zamijeniti s vektorom $\boldsymbol{u_{jk}} = \boldsymbol{C_k} - \boldsymbol{C_j}$. To je jedinični vektor iz centra poligona $\boldsymbol{j}$ prema centru poligona $\boldsymbol{k}$. Pošto je skalarni umnožak $\boldsymbol{u_{jk}} \cdot \boldsymbol{n_j}$ konstantan, može se izvući izvan integrala.

Faktori vidljivosti s novom supstitucijom:

$$\frac{1}{\pi} * (u_{jk} * n_j) * \int_{\omega_i \epsilon \Omega_{jk}} 1 d\omega_i$$

Jedini problem koji nam je još ostao je sam integral, ali on nije ništa drugo nego mjera prostornog kuta, a po definiciji znamo da je to površina poligona k, koju ćemo označiti s A_k, podjeljena s kvadratom udaljenosti između centara poligona, a to je $\|C_j - C_k\|^2$. Još u obzir treba uzeti i kut koji zatvaraju zraka i normala n_k.

Sada formula za faktore vidljivosti glasi:

$$F_{jk} = \frac{1}{\pi} * \frac{A_k}{\|C_j - C_k\|^2} * |u_{jk} \cdot n_j| * |u_{jk} \cdot n_k|$$

Ako promijenimo notaciju, dobiva se puno jasnija formula:

$$F_{jk} = \frac{1}{\pi} * \frac{A_k}{r^2} * |\cos\theta_j| * |\cos\theta_k|$$

gdje je:

r – udaljenost između centara poligona

$\cos\theta_j$ – kut između vektora iz centra poligona j prema centru poligona k i normale poligona j

$\cos\theta_k$ - kut između vektora iz centra poligona j prema centru poligona k i normale poligona k

Trebamo načini još jednu preinaku u ovoj formuli, a to je da moramo uvesti funkciju vidljivosti V_{jk} koja nam govori da li se uopće poligoni vide. Pošto je ovo aproksimacijska formula, dovoljno je ispaliti zraku iz jednog poligona prema drugom i ako zraka ne udari niti jedan drugi objekt, tada se poligoni vide. Funkcija je definirana ovako:

$$V_{jk} = \begin{cases} 1, & ako\ su\ poligoni\ vidljivi \\ 0, & inače \end{cases}$$

Konačna formula faktora vidljivosti:

$$F_{jk} = \frac{1}{\pi} * \frac{A_k}{r^2} * |\cos\theta_j| * |\cos\theta_k| * V_{jk}$$

Kao što smo i rekli, ovo je aproksimacija zato jer smo pretpostavili da su sve zrake jednake, što u stvarnosti nije istina. Stoga nam je potreban neki drugi kut gledanja da dobijemo stvarnu formulu. To nas dovodi do toga da promatramo kako se beskonačno mala površina na poligonu $\boldsymbol{j}$ odnosi prema beskonačno maloj površini na poligonu $\boldsymbol{k}$. Sada formula izgleda ovako:

$$\boldsymbol{F_{djdk} = \frac{\cos\theta_j * \cos\theta_k * dA_k}{\pi * r^2} * V_{jk}}$$

Odnos beskonačno male površine na poligonu $\boldsymbol{j}$ s cijelim poligonom $\boldsymbol{k}$ je:

$$\boldsymbol{F_{djdk} = \int_{A_k} \frac{\cos\theta_j * \cos\theta_k * V_{jk}}{\pi * r^2} dA_k}$$

Sada možemo definirati faktor vidljivosti za poligone $\boldsymbol{j}$ i $\boldsymbol{k}$ kao prosječnu površinu promatranog poligona (u našem slučaju poligona $\boldsymbol{j}$) i prošlog izraza:

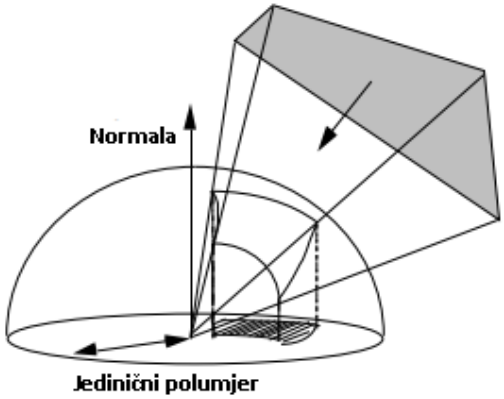
$$\boldsymbol{F_{jk} = \frac{1}{A_j} * \int_{A_j} \int_{A_k} \frac{\cos\theta_j * \cos\theta_k * V_{jk}}{\pi * r^2} dA_k}$$

Ova jednadžba se ne može riješiti direktno, ali postoje algoritmi koji omogućavaju njeno rješavanje.

5.3.1. Izračunavanje faktora vidljivosti pomoću algoritma polukocke

Da bi smo razumjeli algoritam polukocke prvo nam je potrebno znanje o algoritmu polukugle iz kojeg se razvio algoritam polukocke.

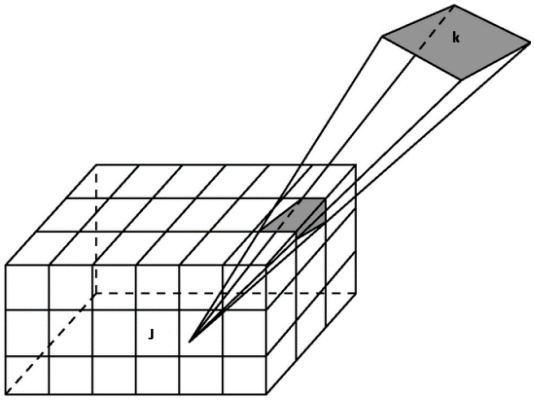
Algoritam polukugle rješava problem unutarnjeg integrala tako da postavlja jediničnu polukuglu oko beskonačno male površine na promatranom poligonu kojeg ćemo zvati $\boldsymbol{j}$ i tu površinu možemo promatrati kao točku (iako se to ne može jer je točka bezdimenzionalna, ali zbog lakšeg predočavanja). Sada se računa faktor vidljivosti za tu točku, suma svih točaka na površini dat će nam konačni vanjski integral. Vratimo se sada na izračunavanje unutarnjeg integrala vidljivog iz točke.



Slika 24. Algoritam polukugle

Unutarnji integral računa se tako da se cijeli poligon kojeg ćemo zvati $\boldsymbol{k}$ projicira na jediničnu polukuglu i ta se projekcija projicira na kružnicu kako je prikazano na slici 24. Omjer te projicirane površine i površine cijele kružnice jednak je unutarnjem integralu za tu točku.

Sada se vratimo na algoritam polukocke. Algoritam polukocke radi tako da postavlja polukocku oko centra poligona $\boldsymbol{j}$ (slika 25) umjesto kugle jer je projekcijski račun lakši nego u slučaju polukugle.



Slika 25. Algoritam polukocke

Nakon što smo postavili polukocku oko centra, trebamo ju podijeliti na manje površine, koje se često nazivaju pikseli (ne treba miješati s pikselima prikaznih jedinica). Nakon toga se izračunavaju faktori vidljivosti tih piskela s centrom poligona $\boldsymbol{j}$. Da bi se razlikovali od stvarnih faktora vidljivosti koje zapravo računamo, faktori vidljivosti tih piksela nazivaju se delta faktori vidljivosti (oznaka $\Delta \boldsymbol{F_{ij}}$), a računaju se prema formuli:

$$\Delta \boldsymbol{F_{ij}} = \frac{\cos \boldsymbol{\theta_i} * \cos \boldsymbol{\theta_j}}{\boldsymbol{\pi} * \boldsymbol{r^2}} * \Delta \boldsymbol{A}$$

gdje je $\Delta \boldsymbol{A}$ površina piksela. Delta faktore vidljivosti izračunamo samo jednom i spremamo ih u tablicu iz koje kasnije samo uzimamo relevantne podatke. Sada jedino što treba napraviti je projicirati poligon $\boldsymbol{k}$ na polukocku i zbrojiti delta faktore vidljivosti koje pokriva projicirana površina [6].

5.4. Rješavanje jednadžbe metode isijavanja

Postoje dva načina kako riješiti jednadžbu metode isijavanja: iterativan(prikupljanje) način i progresivan(ispaljivanje) način.

Iterativan način objašnjen je u poglavlju 5.1. gdje je opisana osnovna ideja i dan je pseudokod iterativnog načina. Ovdje ćemo još jednom samo ukratko objasniti kako ta metoda radi u sedam koraka.

1. Korak: Učitavanje scene
2. Korak: Dijeljenje scene na poligone
3. Korak: Izračunati faktore vidljivosti za svaki par tih poligona
4. Korak: Postaviti početne vrijednosti izlazne svjetlosti za poligone, to znači da će sve površine imati početnu izlaznu svjetlost nula osim izvora svjetlosti
5. Korak: Za svaki poligon prikupiti svjetlost sa svih ostalih poligona u sceni
6. Korak: Izračunati izlaznu svjetlost za svaki poligon
7. Korak: Ako nije zadovoljen uvjet konvergencije ili nismo prešli broj zadanih iteracija vrati se na korak 4

Ovakav način rješavanja jednadžbe isijavanja ima velike poteškoće zbog vremenske i prostorne složenosti izračunavanja faktora vidljivosti koja je $O(n^2)$ gdje je n broj poligona u sceni. U iterativnom načinu izračunavanje faktora vidljivosti troši većinu vremena od cijelog postupka, naravno to ovisi i o broju iteracija. Ako imamo puno iteracija, vrijeme rješavanja jednadžbe metode isijavanja također će uzeti dosta vremena. Iako je veliki problem vrijeme izračunavanja, problem s memorijom je još veći. Uzmimo na primjer da je scena podijeljena na 10000 (10^4) površina, tada u memoriji moramo imati mjesta za 10^8 faktora vidljivosti. Ako uzmemo da jedan faktor vidljivosti u memoriji zauzima 4 bajta, tada je to $4 * 10^8$ bajta memorije, što je

približno oko 0.37 gigabajta. Očito je da trebamo pronaći neko drugo rješenje, a to rješenje naći ćemo u obliku progresivnog algoritma rješavanja jednadžbe metode isijavanja.

5.4.1. Progresivni algoritam rješavanja jednadžbe metode isijavanja

Progresivna metoda radi upravo suprotno od onog što je radila iterativna metoda. Metoda ispaljuje zrake iz svakog poligona dok je u iterativnoj metodi svaki poligon prikupljao zrake sa svih ostalih poligona. U svakom ciklusu, odnosno iteraciji, poligon s potencijalno najvećom izlaznom svjetlosti izabran je za „ispaljivanje“ zraka prema svim ostalim poligonima. Ovaj proces ažurira isijavanje svakog poligona u sceni za razliku od iterativne metode. Prednost ove metode je ta da se u svakoj sljedećoj iteraciji može vidjeti napredak u konvergenciji. Također, izračunavanje faktora vidljivosti računa se samo prilikom svakog ispaljivanja zraka, tako da ne moramo raditi matricu od n^2 elemenata jer neki poligoni nikad niti neće doći u mogućnosti da ispaljuje zrake zbog svoje vrlo male izlazne energije koja neće imati utjecaj na ostatak scene. Najveća prednost ovog algoritma je u njegovoj dobroj vremenskoj i prostornoj složenosti koja je O(n) za oboje te brzini konvergencije koja je zadovoljavajuća već nakon nekoliko iteracija. Za kraj ćemo dati pseudokod progresivnog algoritma.

```
Kreiraj scenu;
Podjeli površine u približno jednake poligone;
Za svaku iteraciju
      Odaberi poligon j;
      Izračunaj faktor vidljivosti F_jk za sve poligone k;
      Za svaki poligon k
            Ažuriraj izlaznu svjetlost za poligon k;
            Ažuriraj emitirajuću svjetlost za poligon k;
      Postavi emitirajuću svjetlost poligona j na nula;
```

5.5. Prednosti i nedostaci

Najveća prednost metode isijavanja je mogućnost generiranja vrlo kvalitetnih realističnih slika. Meke sjene i preljev boja posljedica su te metode. Uz to što se mogu generirati vrlo realistične slike, također je i poprilično točna u smislu kako se energija prenosi između poligona. Korisna stvar ove metode je i to da je neovisna o položaju kamere, što znači da se kamera slobodno može kretati po sceni bez ikakvog dodatnog izračunavanja.

Naravno niti jedan algoritam (zasad) nije savršen pa tako nije niti ovaj. Postoji velika potreba za memorijom, a tako i za vremenom izračunavanja, pogotovo ako se koristi iterativni način. Zahtjeva se da scena bude podijeljena u mrežu poligona pa algoritam ne može raditi sa ne difuznim materijalima.

6. Implementacija metode isijavanja

Za implementaciju metode isijavanja odabran je programski jezik C++ zbog svoje podrške objektno-orijentiranog načina programiranja te mogućnošću upravljanja memorijom. Grafičko sučelje korišteno za izradu aplikacije je OpenGL, GLUT za kreiranje prozora u kojem je prikazana scena i GLUI za podešavanje parametara o kojima će ovisiti izgled scene. U ovom poglavlju bit će opisana jezgra programa koja implementira metodu isijavanja, ali prije nego se krene na objašnjavanje same jezgre programa, prvo treba ukratko objasniti sve pomoćne razrede koji su temelj za izradu jezgre.

6.1. Pomoćni razredi

Point3D je razred koji opisuje točku u prostoru tako da sadrži njezine (x,y,z) koordinate.

Vector3D je razred koji kao što mu i samo ime govori sprema podatke o vektorima. Također razred sadržava sve funkcije važne programu, a koje se mogu izvršavati

nad vektorima, a to su normaliziranje vektora, određivanje norme vektora, skalarni te vektorski produkt.

Color je razred koji omogućuje spremanje RGB boje za pojedine poligone. Uz standardne funkcije za postavljanje boje, sadrži i funkciju mapiranja boje, koja preslikava vrijednosti RGB boja u interval [0,1].

Lambertian je razred koji sadrži podatke vezane uz materijal površine. Kao što je prije rečeno radi se o savršeno difuznim materijalima. Razred sadrži boju i koeficijent reflektivnosti poligona, a najvažnija funkcija je funkcija **rho()** koja vraća refleksivnost koja nam je potrebna u jednadžbi metode isijavanja.

Plane je razred koji sadrži podatke o ravnini na kojoj se poligoni nalaze te pošto se cijeli program temelji na radijus kvadratnim poligonima također sadrži podatke gdje se taj kvadrat nalazi na toj ravnini. Ovaj razred kasnije će nam biti vrlo važan kod utvrđivanja funkcije vidljivosti.

Patch je jedan od najvažnijih razreda jer on sadrži sve podatke važne za svaki poligon. Razred sadrži sva 4 vrha poligona, vektor normale, površinu, centar, podatke o materijalu, kojoj ravnini pripada te da li je poligon emitirajući, odnosno je li taj poligon izvor svjetlosti pošto znamo da, iako metoda isijavanja sve poligone tretira jednako, neki su poligoni zapravo izvor svjetlosti i imaju emitivna svojstva.

AreaLight je razred koji u sebi sadrži podatke o izvoru svjetlosti i treba naglasiti da je to alternativni način spremanja podataka o izvorima svjetlosti pošto postoji mogućnost da se podaci spremaju i u **Patch** razred.

Sada kada imamo sve potrebne temeljne razrede možemo krenuti na implementaciju metode isijavanja koja se nalazi u **World** razredu.

6.2. Jezgra metode isijavanja

Algoritam metode isijavanja koji je implementiran je iterativni algoritam. Ponovit ćemo još jednom svih sedam koraka ovog algoritma pa ćemo dalje tim redom objašnjavati kako je svaki korak implementirani u našem programu.

1. Korak: Učitavanje scene
2. Korak: Dijeljenje scene na poligone

3. Korak: Izračunati faktore vidljivosti za svaki par tih poligona
4. Korak: Postaviti početne vrijednosti izlazne svjetlosti za poligone, to znači da će svi poligoni imati početnu izlaznu svjetlost nula osim izvora svjetlosti
5. Korak: Za svaki poligon prikupiti svjetlost sa svih ostalih poligona u sceni
6. Korak: Izračunati izlaznu svjetlost za svaki poligon
7. Korak: Ako nije zadovoljen uvjet konvergencije ili nismo prešli broj zadanih iteracija vrati se na korak 4

Svaki od tih koraka nalazi se u obliku funkcije u razredu **World** koji sadrži veliki broj varijabli koje će biti objašnjene u procesu. A sada možemo krenuti na objašnjavanje pojedinih koraka.

1. Korak: Učitavanje scene

Izrada scene vrlo je jednostavna iako ne i toliko intuitivna. U tekstualnu datoteku zapisuju se podaci o pravokutnicima. Postoji jedino mala razlika u način označavanja pravokutnika koji su izvori svjetlosti.

$$\boldsymbol{R} < \boldsymbol{num\ of\ patches} >< \boldsymbol{point\ 1} >< \boldsymbol{point\ 2} >< \boldsymbol{point\ 3} >< \boldsymbol{point\ 4} > < \boldsymbol{normala} >< \boldsymbol{material} >$$

gdje je:

$\boldsymbol{R}$ – slovo koje označava da se radi o običnoj površini

$< \boldsymbol{num\ of\ patches} >$ – Broj koji nam govori na koliko će se poligona zadana površina raspodijeliti, zadaje se broj poligona u jednom retku (npr. Ako želimo raspodijeliti pravokutnik na 100 poligona, upisat ćemo broj 10).

$< \boldsymbol{point\ i} >$ – Gdje je $\boldsymbol{i}$ od 1 do 4. To su koordinate točaka zadane u smjeru suprotnom kazaljke na sat. Zadaju se (x,y,z) koordinate točke (npr. (0.0, 0.0,0.0) označava ishodište koordinatnog sustava).

$< \boldsymbol{normala} >$ – Vektor normale, zadaje se također kao i točka (npr. Vektor normale za pravokutnik koji leži u x-z ravnini i usmjeren je prema gore označava se s (0.0, 1.0, 0.0))

$< \boldsymbol{material} >$ – Svojsta materiala, a zadaju se u obliku (r,g,b,kd) (npr. Za bijeli pravokutnik koji ima faktor reflektivnosti 0.5 pisali bismo (1.0, 1.0, 1.0, 0.5))

Isti princip rada je i za izvore svjetlosti samo što se kod njih umjesto slova R koristi slovo E te se umjesto koeficijenta refleksivnosti zadaje intenzitet svjetla koji je također samo određen broj kojeg se može podešavati u svrhu poboljšavanja scene.

Funkcija koja parsira teksutalnu datoteku naziva se **World::build**.

2. Korak: Dijeljenje scene na poligone

Funkcija koja obavlja ovaj zadatak je funkcija **World::meshing**. Funkcija kao argumente prima podatke učitane iz teksutalne datoteke s razlikom da prima samo dvije nasuprotne točke, odnosno prvu i treću jer su one dovoljne za podjelu. Zadatak funkcije je podijeliti površinu zadanu točkama i spremiti te poligone u polje poligona gdje će se nalaziti svi poligoni potrebni za kasniju upotrebu. Ti poligoni se spremaju u polje poligona **vector<Patch> ploha**. Uz to što dijeli površine također podešava sve važne podatke vezane uz svaki poligon, odnosno normalizira normalu u slučaju da ona već prije nije, sprema podatak kojoj ravnini pripada poligon, postavlja svojstva materijala poligona te postavlja podatke o ravnini na kojoj se nalazi površina i te podatke o ravnini sprema u polje ravnina **vector<Plane> ravnine**.

Ideja funkcije je da odredi kolika će biti veličina svakog novonastalog poligona, odnosno koliko će se on protezati u x, y i z smjeru, te tada prema tome dijeli površinu stvarajući nove koordinate za svaki od tih poligona.

Sada slijedi suštinski dio metode isijavanja, a to je izračunavanje faktora vidljivosti i rješavanja jednadžbe metode isijavanja. Sav taj posao obavlja funkcija **World::render**.

3. Korak: Izračunati faktore vidljivosti za svaki par tih poligona

Funkcija koja obavlja posao 3. koraka je funkcija **World::form_factor**. Izračunavanje faktora vidljivosti temelji se ne formuli koju smo izveli u 5. poglavlju pa se prisjetimo kako je izgledala:

$$F_{jk} = \frac{1}{\pi} * \frac{A_k}{r^2} * |\cos\theta_j| * |\cos\theta_k| * V_{jk}$$

Ovo je aproksimacijska formula koja smatra da su sve zrake između poligona j i k jednake. Prvi zadatak funkcije je alocirati memoriju potrebnu za matricu od n^2 elemenata. Sljedeći korak je za svaki poligon odrediti faktor vidljivosti svih ostalih

poligona, a to se radi tako da se izračunaju kosinusi kuteva za oba poligona, taj dio se odnosi na ova dva kosinusa kuta $\cos\theta_j, \cos\theta_k$ te prva provjera vidljvosti je da li je možda koji od ova dva kosinusa manji od nule. Ako je to slučaj, tada se poligoni ne mogu vidjeti i njihov faktor vidljivosti jednak je nuli. Ako se ne može utvrditi da se ne vide tada se provjerava na složeniji način. Složenija provjera zasniva se na tome da se zraka ispaljuje iz centra poligona j prema centru poligona k i prati se da li će ta zraka udariti neku površinu prije nego udari poligon k, ako udari prije neku površinu, tada poligoni nisu vidljivi, a u suprotnom slučaju jesu. Funkcija zadužena za to je funckija **World::vidljivost**. U ovoj funkciji su nam potrebne ravnine koje smo spremali i koje sadržavaju podatke o tome gdje se nalazi poligon na ravnini. Funkcija radi tak da prvo provjeri da li je zraka sjekla neku ravninu prije ravnine na kojoj se nalazi poligon k i ako nije tada znamo da se vide, ako je, tada poziva funkciju **World::pripada_intervalu** koja će provjeriti da li zraka sječe neku od površina scene jer ravnine su beskonačne i zraka može sjeći bilo koji dio ravnine, ali nas zanima samo onaj dio koji je nama važan za scenu.

Treba napomenuti da je ovo jedini dio programa koji se mogao paralelizirati za istovremeni rad na više jezgri procesora, a za to je korišten OpenMP API. Iako se u ovom radu neće ulaziti u objašnjenje što je to paraleliziranje programa, korisno je napomenuti da paralelizacija drastično ubrzava izvođenje programa, naravno pod uvjetom da je dobro odrađena u programu jer bi se inače mogao stvoriti kontraefekt i moglo bi doći do usporavanja programa. Paralelizacija programa vrlo je korisna stvar zbog toga jer se u novije vrijeme jačina procesora ne povećava previše pa u igru ulaze procesori sa više jezgara. O kakvom se poboljšanju radi, jasno govori da se za scenu od 10000 poligona na procesoru Intel Core i5 3210M pomoću paralelizacije postiglo ubrzanje izračunavanja za oko 3.8 puta.

Korak 4 – 7: Rješavanje jednadžbe metode isijavanja i iscrtavanje

Sljedeća 4 koraka spojeni su u jednu funkciju **World::RADIOSITY** pa ćemo ih tretirati zajedno radi lakšeg objašnjavanja.

Prvo što treba funkcija napraviti je alocirati potrebnu memoriju za izlaznu svjetlost svakog poligona (polje **Color RadiositiyFactor**) i postaviti početne vrijednosti izlazne svjetlosti za svaki poligon. To znači da ako je poligon izvor svjetlosti, postavlja mu se emitivna svjetlost, ako nije, onda je ta vrijednost nula.

Sljedeći korak je ući u petlju koja radi na iterativni način tako da joj se zada broj iteracija. Svaki poligon prikuplja svjetlost sa svih ostalih poligona. Nakon što imamo vrijednosti ulaznih svjetlosti, izračunava se izlazna svjetlost za svaki poligon i te vrijednosti spremaju se u polje **RadiosityFactor**. Ako nismo dosegnuli broj iteracija ponavljamo cijeli postupak. Treba istaknuti da se ovaj dio programa ne može paralelizirati ni u kojem slučaju jer je svaki poligon ovisan o vrijednostima drugih poligona i ako se pokuša paralelizirati, dobit će se neki poligoni drugačijih boja nego bi trebali biti (npr. na bijelom zidu može se pojaviti plavi poligon iako u sceni nema plavih izvora svjetlosti niti drugi površina koje su plave).

Nakon što smo dosegnuli broj iteracija izlazimo iz petlje i vrijeme je za crtanje poligona. To je izvedeno tako da se prije iscrtavanja svakog poligona njegova boja interpolira tako da se ne vide prijelazi između susjednih poligona jer, prisjetimo se, rekli smo da svaki poligon ima konstantnu boju kroz cijelu svoju površinu. Interpolacija boja poligona izvodi se u funkciji **World::interpoliraj**.

6.3. Grafičko sučelje

U grafičkom sučelju omogućeno je upravljanje parametrima metode isijavanja odnosno moguće je odabrati broj iteracija te broj poligona u sceni. Također je omogućena je pomična kamera jer, kao što je prije i rečeno, jednom kada se izračunaju intenziteti boja za svaki poligon, oni su neovisni o položaju kamere.

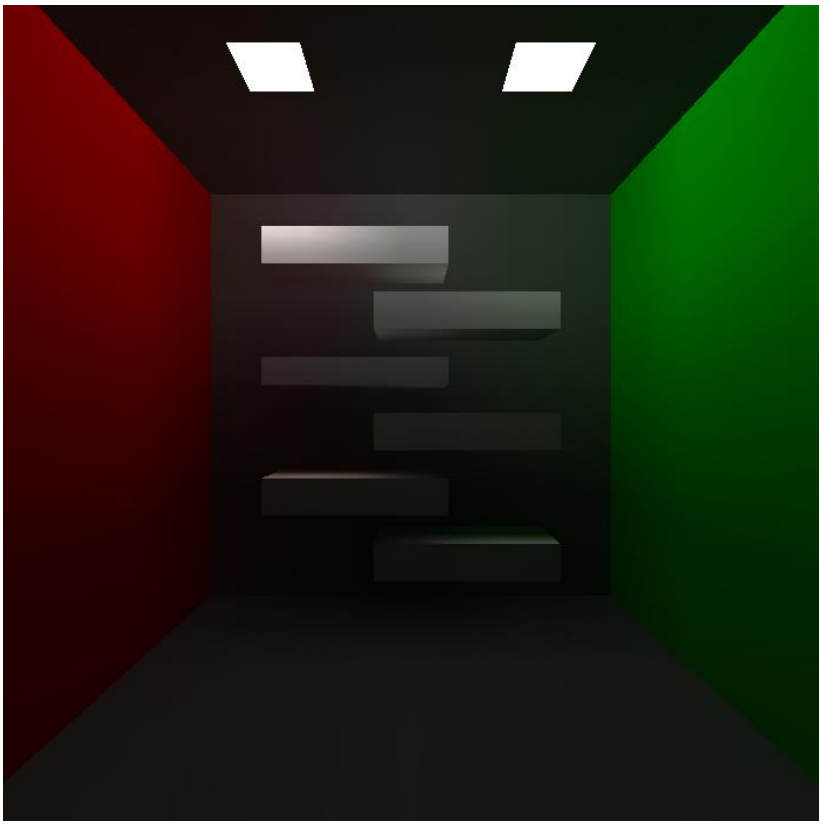
6.4. Performanse

Sada ćemo dati omjer realističnosti slike i vremena koje je potrebno za njen prikaz. Mjerenje se provodi na procesoru Intel Core i5 3210M i na grafičkoj kartici AMD Radeon HD 7600M. U tablici će biti prikazano vrijeme izračunavanja faktora vidljivosti te koliko je ukupno vremena potrebno za izračunavanje cijelog algoritma s tim da je za broj iteracija u iterativnom algoritmu rješavanja jednadžbe metode isijavanja uzeto 200 iteracija. Vrijeme će biti prikazano u sekundama. Nakon toga,

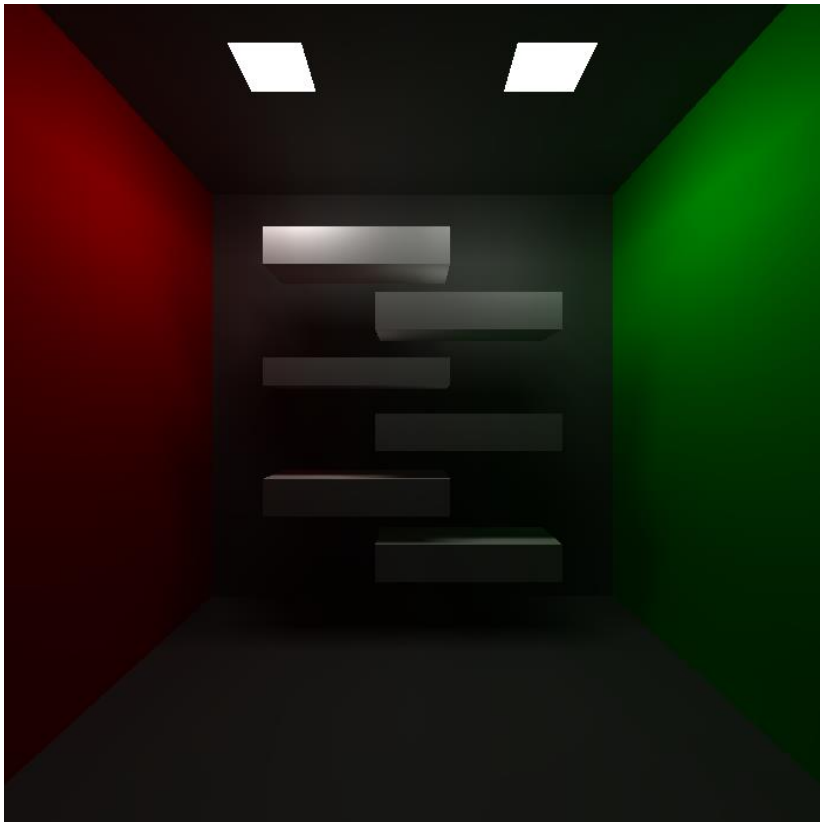
biti će prikazane slike koje se dobiju s 656, 5904 i 16400 poligona u sceni da bi se mogao usporediti napredak u relazimu.

Tablica 1. Ovisnost vremena izračunavanja algoritma o broju poligona

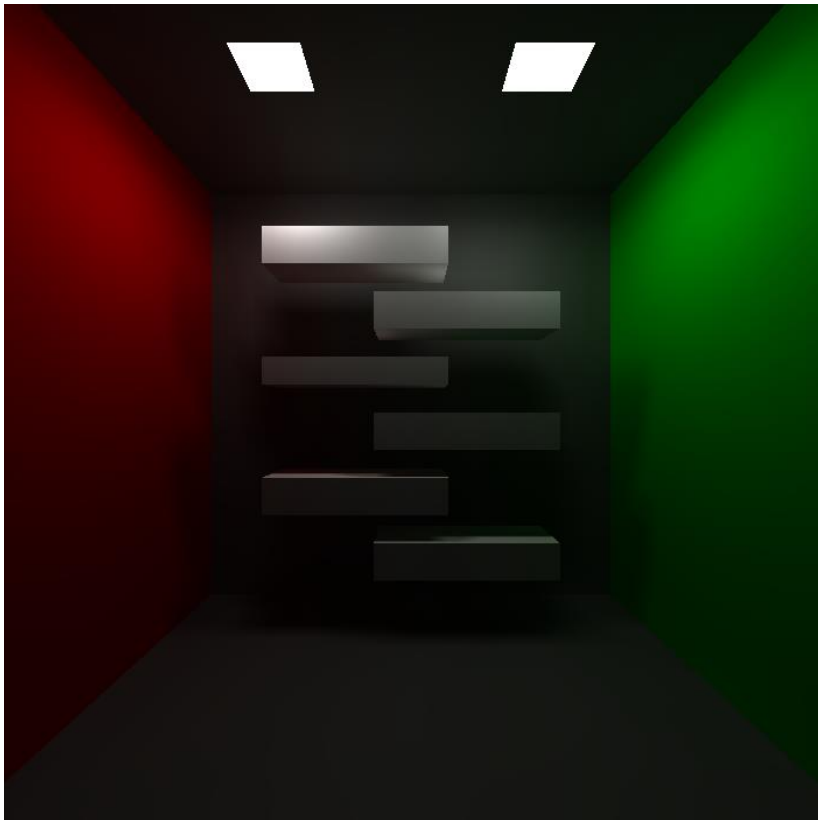
Broj poligona u sceni	Vrijeme izračunavanja faktora vidljivosti (sekundi)	Vrijeme izračunavanja ukupnog algoritma
656	1.375	2.267
2624	20.549	33.163
5904	104.557	167.504
10496	336.688	534.096
16400	878.201	1359.861



Slika 26. 656 poligona



Slika 27. 5904 poligona



Slika 28. 16400 poligona

7.Zaključak

Simulacija propagacije svjetlosti pomoću računalne grafike kompleksan je zadatak iako nam doprinosi razvijanju vrlo realnih slika. Ovaj rad je bio je kratki uvod u to područje računalne grafike. Vidjeli smo nekoliko algoritama lokalnog i globalnog osvjetljenja, ali sve su bile samo aproksimacije, tako da se još uvijek traga za novim metodama koje će još više doprinijeti realnosti slika, a to je ujedno i jedan od glavnih ciljeva računalne grafike.

Danas je najefikasniji model globalnog osvjetljenja kombinacija algoritma praćenja zrake i preslikavanje fotona, ali to je i daleko najsporije rješenje. Iako postoje sklopovske implementacije tog algoritma, računala dostupna prosječnom korisniku još nemaju toliku računalnu moć za rješavanje tih modela efikasno.

Model globalnog osvjetljenja (metoda isijavanja) koji je implementiran u ovom radu vrlo je efikasan što se tiče prikaza realnih slika, ali ima veliki nedostatak, a to je da može raditi samo s difuznim materijalima, a mi se svakodnevno susrećemo sa stvarima poput zrcala koje nije moguće generirati pomoću metode isijavanja. Isto tako, vremenski i memorijski vrlo je zahtjevna, tako da ju je još uvijek jako teško primijeniti u stvarnom vremenu.

Najčešće rješenje problema nemogućnosti rada sa zrcalnim refleksijama u metodi isijavanja je kombiniranje metode isijavanja s metodom praćenja zraka, koja je jako dobra u generiranju zrcalnih objekata.

8.Literatura

[1] John F. Hughes; Andries van Dam; Morgan McGuire; David F. Sklar; James D. Foley; Steven K.Feiner; Kurt Akeley: „*Computer Graphics: Principle and Practice“,*3. izdanje, 2007.

[2] Kevin Suffern: „*Ray Tracing from the Ground Up“*, 2007.

[3] An Introduction to BRDF-Based Lighting, s Interneta, http://www.cs.princeton.edu/courses/archive/fall06/cos526/tmp/wynn.pdf, 25.4.2015.

[4] Peter Shirley; Michael Ashikhmin; Steve Marschner: *„Fundamentals of Computer Graphics“*, 2009.

[5] Photon Mapping Made Easy, http://www.cs.mtu.edu/~shene/PUBLICATIONS/2005/photon.pdf, 25.5.2015.

[6] Michael F. Cohen; John R. Wallace: „*Radiosity and Realistic Image Synthesis“*, 1993.

9. Sažetak

(Postupak osvjetljavanja scene metodom isijavanja)

U ovom radu opisani su modeli osvjetljenja scene u računalnoj grafici s naglaskom na modele globalnog osvjetljenja koji točnije simuliraju propagaciju svjetlosti kroz scenu u odnosu na lokalne modele osvjetljenja. Opisano je nekoliko algoritama globalnog osvjetljenja, a detaljno je razrađen globalni model osvjetljenja pod imenom metoda isijavanja. Također opisana je jednostavna implementacija metode isijavanja te je dan omjer performanse i realizma slike.

Ključne riječi: svjetlost, univerzalna jednadžba iscrtavanja, lokalni modeli osvjetljenja, globalni modeli osvjetljenja, metoda isijavanja.

9. Abstract

(Radiosity illumination model)

This essay contains description of illumination models in computer graphic with main focus on global illumination models which more accurately simulate propagation of light in scene in regard to local illumination models. Several global illumnation algorithms are described and Radiosity global illumination method is described in detail. Also, simple implementation of Radiosity method is described and the ratio of performanse and realism of pictures is given.

Keywords: light, renderig equation, local illumination model, global illumination model, Radiosity.