**Osnove virtualnih okruženja**

**Laboratorijske vježbe**

**Vježba 2**

**Principi iscrtavanja: Praćenje zrake**

FER – ZTE – Igor S. Pandžić

Suradnja na pripremi vježbe:

Milenka Gadže

Miran Mošmondor

Vjekoslav Levačić

Marko Pipal

# Uvod

Praćenje zrake (engl. *ray tracing*) klasična je i izuzetno često upotrebljavana metoda iscrtavanja 3D scena. Metoda je zasnovana na zakonima optike (refleksija, refrakcija) uz korištenje pojednostavljenog modela osvjetljenja za izračunavanje intenziteta svjetlosti u određenoj točki.

Metodom praćenja zrake dobivaju se vrlo impresivne slike 3D scena koje uključuju efekte zrcaljenja, prozirnosti i sjena. Pritom je osnovna metoda vrlo jednostavna za implementaciju jer se sastoji od rekurzivnog ponavljanja nekoliko osnovnih operacija.

Cilj ove vježbe je implementirati osnovnu metodu praćenja zrake te kroz to steći osnovne ideje o iscrtavanju slike 3D scene, modelu osvjetljenja i efektima sjene, zrcaljenja i prozirnosti.

# Alati potrebni za izvođenje vježbe

Za izvođenje vježbe potreban je Unity editor v2022.3. Program je napravljen u verziji v2022.3.11f, no bilo koja kasnija 2022 verzija bi trebala funkcionirati. Također je potreban program za uređivanje C# i HLSL koda, najjednostavniji je Visual Studio Code s nastavcima za Unity, C# i HLSL.

# Teorijska podloga

Praćenje zrake obrađeno je na predavanju o iscrtavanju. Postupak praćenja zrake je jednostavan: za svaku točku ekrana, prati se zraka koja kroz tu točku ulazi u scenu. Za zraku se traži presjek s predmetima u sceni. Ako postoji presjek, računa se osvjetljenje u točki presjeka te se računaju reflektirana zraka i refraktirana zraka. Za ove dvije zrake postupak se rekurzivno ponavlja te se zbrajaju doprinosi intenziteta osvjetljenja Iphong, Irefl i Irefr svih nađenih točaka presjeka.

Pseudo-kod za postupak izgleda ovako:

za svaki x,y u prozoru iscrtavanja

izračunaj zraku R kroz x,y

boja I = TraceRay(R, dubina)

obojaj točku x,y bojom I

kraj

funkcija TraceRay(R,dubina)

ako je dubina>max. dubine, prekini funkciju i vrati crnu boju

nađi najbliži presjek zrake R sa scenom

ako nema presjeka, prekini funkciju i vrati kao rezultat boju pozadine

izračunaj boju lokalnog Phongovog osvjetljenja Iphong u točki presjeka

izračunaj odbijenu zraku Rrefl

boja Irefl = TraceRay(Rrefl, dubina+1)

izračunaj refraktiranu zraku Rrefr

boja Irefr = TraceRay(Rrefr, dubina+1)

boja I = suma(Clocal, Crefl,Crefr)

kraj: vrati boju I

## Računanje zrake kroz točku x,y u prozoru iscrtavanja

Zraku promatramo kao pravac kroz dvije točke u trodimenzionalnom xyz-sustavu. Najbolje ju je predstaviti hvatištem i smjerom. Ekran za iscrtavanje postavljamo u xy-ravninu tako da točka (0,0,0) bude u središtu ekrana. Oko promatrača je početna točka koja definira zraku i nalazi se na unaprijed određenoj i fiksiranoj poziciji (0,0,z). Druga točka koja definira zraku je piksel na ekranu za iscrtavanje. Položaj oka se uzima za hvatište zrake, a smjer zrake će biti normalizirani vektor od oka promatrača do piksela na ekranu i računa se kao smjer pravca kroz dvije točke ( oko – točka 1, piksel – točka 2).

 **y**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

piksel

**x**  
 **z (0,0,z)**

oko

**Slika 3.1:** Prikaz zrake koja prolazi kroz piksel ekrana

## Testiranje dubine rekurzije

Dubinu rekurzije kod zbrajanja intenziteta možemo zadati kao konstantu (tj. da se funkcija *TraceRay* rekurzivno zove točno određen broj puta) ili je ograničiti tako da zadamo minimalni intenzitet koji ima smisla uračunati u ukupno lokalno osvjetljenje. Ako je izračunati intenzitet u rekurziji manji od zadanog onda prekidamo postupak rekurzije jer je utjecaj tog intenziteta zanemariv. U ovoj vježbi implementirati će se oba pristupa.

## Nalaženje najbližeg presjeka zrake sa scenom

Jedini način da se odredi najbliži presjek zrake sa scenom je da se ispita i odredi presjek zrake sa svakim od objekata u sceni i uzme onaj čija je udaljenost od početka zrake najmanja.

U ovoj vježbi radit ćemo sa kuglama. Za zraku su poznati hvatište *P* i vektor smjera *Direction*, a za svaku kuglu su poznate koordinate središta *C* i radijus *r* prikazani na slici 3.2.

Algoritam za nalaženje presjeka s kuglom:

funkcija presjekPostoji(zraka)

izračunaj vektor PC

izračunaj kut α između vektora smjera zrake Direction i vektora PC

ako je kut α veći od 90 stupnjeva zraka ne pokazuje u smjeru objekta pa prekini funkciju i vrati false

nađi udaljenost d zrake od središta kugle ( d = |PC| \* sin(α) )

ako je d veće od radijusa kugle r prekini funkciju i vrati false

odredi udaljenost hvatišta zrake P od točke D

|PD| = 

odredi udaljenost hvatišta zrake P od bližeg presjeka

|PBližiPresjek| = |PD| – ()

ako je |PBližiPresjek| <= 0 + Epsilon[[1]](#footnote-1) tada

hvatiste se nalazi unutar kugle i vrijedi

|PBližiPresjek|= |PDaljiPresjek|= |PD| + ()

kraj: vrati true

funkcija vratiPresjek()

kraj: vrati BližiPresjek

r

r

**D**

**C**

DC ┴ PD

BližiPresjek

Smjer

**Hvatište**

DaljiPresjek

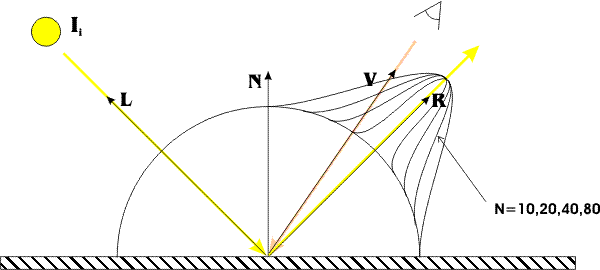
d

α

**Slika 3.2:** Presjek zrake i kugle

## Računanje lokalnog osvjetljenja u točki presjeka

Kad pronađemo presjek zrake i predmeta u sceni potrebno je dotičnom pikselu pripisati intenzitet koji se računa se prema Phongovom modelu odbijanja svjetlosti.



**Slika 3.3:** Phongov model osvjetljenja

Potrebno je razlikovati pojam zrakekao **zraka svjetlosti** (usmjerena suprotno vektoru *L*) i zraku koja **označava smjer našeg pogleda** (usmjerena suprotno vektoru *V*). Formula Phonogovog modela (3.1) za intenzitet glasi:

(3.1)

Svi intenziteti *I* i koeficijenti *k* zapravo su vektori boje koji se sastoje od tri komponente: *R*, *G*, *B* – dakle crvena, zelena i plava. **Sve komponente poprimaju vrijednosti od 0 do 1**. Ukoliko neke od *R*, *G*, *B* vrijednosti premašuju granice zaokružite ih na maksimalnu, odnosno minimalnu vrijednost. Točka predstavlja operaciju skalarnog umnoška između dva vektora.

### Ambijentna komponenta ()

Ambijentno svjetlo grubo aproksimira globalno osvjetljenje I daje minimum osvjetljenja kojim se izbjegava da dijelovi scene na koje svjetlo ne pada direktno budu potpuno crni, no koristi se u minimalno jer daje efekt jednolične boje. Ambijentna svjetlost karakterizirana je intenzitetom *Ia* koji je konstanta za cijelu scenu, te ambijentnim koeficijentom materijala ka, koji određuje u kojoj mjeri materijal reagira na ambijentnu svjetlost.

### Difuzna komponenta ()

Difuzna komponenta vjerno opisuje Lambertov zakon koji opisuje difuzno raspršivanje svjetla na predmetu. Difuzno odbijanje (raspršivanje) ide u svim smjerovima jednakim intenzitetom koji ovisi o ulaznom kutu, te je najveći kada zraka upada okomito na površinu. Ova zakonitost je opisana skalarnim produktom vektora smjera upadne zrake *L* i normale na površinu *N*. Difuzna komponenta je uz to proporcionalna intenzitetu izvora svjetlosti *Ii* i difuznom koeficijentu materijala *kd*.

### Spekularna komponenta ()

Spekularna komponenta aproksimira spekularni odsjaj na predmetu. Spekularni odsjaj je karakteriziran specifičnom krivuljom prikazanom na slici, koja ima oštar maksimum u smjeru zrcaljenja *R* (*R* je simetričan *L* u odnosu na *N*). Što je materijal sjajniji to je ovaj maksimum više izražen, tj. krivulja je strmija. U prirodi se to primjećuje kao oštrina spekularnog odsjaja – što je materijal sjajniji, to je na njemu odsjaj svjetlosti oštriji. Ova karakteristika aproksimirana je prilično vjerno skalarnim produktom smjera gledanja V i smjera zrcaljenja *R*, te potenciranjem tog produkta faktorom *n* koji opisuje sjajnost materijala. Što je *n* veći to je materijal sjajniji. Spekularna komponenta je proporcionalna intenzitetu izvora svjetlosti *Ii* i spekularnom koeficijentu materijala *ks*.

### Utjecaj sjene

U svakoj točki presjeka zrake i predmeta u sceni računa se doprinos osvjetljenja modelom lokalnog osvjetljenja, npr. Phongovim. Model lokalnog osvjetljenja ne računa efekt sjene. Dakle, upotrijebimo li samo Phongov model osvjetljenja, potpuno ćemo zanemariti činjenicu da se primjerice. između izvora svjetlosti i promatrane točke može naći još jedan predmet zbog čega ovaj izvor svjetlosti uopće ne baca svjetlo na našu promatranu točku pa su difuzne i spekularne komponente lokalnog osvjetljenja jednake nuli!

Ovaj problem rješava se zrakama za ispitivanje sjene *(engl shadow ray, shadow feeler).* Te zrake idu od mjesta presjeka do pojedinog izvora svjetlosti i provjeravaju nalazi li im se išta na putu tj. postoji li presjek sa scenom. U slučaju da postoji presjek zrake za ispitivanje sjene sa nekim objektom na njenom putu utjecaj **lokalnog osvjetljenja ne uračunava difuznu i spekularnu komponentu**. Naravno pri takvom postupku zanemarujemo činjenicu da objekti koji stoje na putu između promatrane točke i izvora svjetlosti mogu biti prozirni. Iako u ovoj vježbi to nije potrebno implementirati, ovaj se problem rješava na način da se promatranoj točki pripisuje jakost izvora svjetlosti Ii pomnožena sa faktorima prozirnosti tijela koja se nalaze na putu zrake.

Ovaj postupak je praktičan jer se za traženje presjeka sa zrakom za ispitivanje sjene koriste potpuno iste metode kao i za presjek sa ostalim zrakama u postupku, dakle uvođenjem sjena nismo bitno povećali složenost implementacije (naravno, složenost u smislu vremena izvođenja metode se povećava).

## Računanje odbijene zrake

**N**

**U**

**R**

**U**

**N'**

**2N'**

U – upadni vektor

N – vektor normale

R – reflektirani vektor

**Slika 3.4 i 3.5:** Vektorski prikaz reflektirane zrake

reflektirana zraka

**N**

**U**

**R**

upadna zraka

α

α

Kod računanja odbijene zrake potrebno je najprije odrediti vektor normale *N* na površinu kugle u točki presjeka zrake s kuglom te upadni vektor *U* koji je suprotan vektoru smjera upadne zrake (slika).

Sada se može odrediti reflektirani vektor. Iz Slike 3.4 se vidi da je:

(3.2)

gdje je

(3.3)

a (*U•N*) skalarni produkt upadnog vektora i vektora normale.

Dakle, reflektirani vektor oko normale N na površinu kugle u točki presjeka zrake s kuglom može se izračunati pomoću formule :

(3.4)

## Računanje refraktirane zrake

upadna zraka

**N**

**R**

**U**

α

β

**n1**

**n2**

refraktirana zraka

U – upadni vektor

N – vektor normale

R – refraktirani vektor

**Slika 3.6:** Vektorski prikaz refraktirane zrake

Kut reftraktirane zrake *β* se može jednostavno odrediti pomoću Snellovog zakona :

(3.5)

gdje su *n1* i *n2* indeksi refrakcije sredstva iz kojeg dolazi zraka, odnosno sredstva nad kojim se lomi zraka.

**R**

**a·N**

**b·U**

**Slika 3.7:** Detaljnije prikazan vektorski prikaz

Kod određivanja vektora smjera refraktirane zrake u prostoru polazi se od činjenice da je refraktirani vektor *R* kombinacija upadnog vektora *U* i vektora normale *N* (slika 3.7) :

(3.6)

Važno je da su upadni vektor i vektor normale normalizirani što znači da im je duljina jednaka 1. Sada je potrebno odrediti samo još koeficijente *a* i *b*.

Vrijedi da je :

(3.7)

(3.8)

gdje je

(3.9)

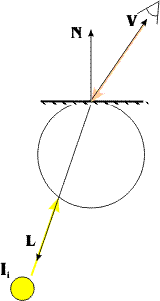
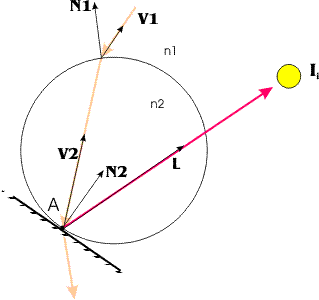
## Kombiniranje lokalnog osvjetljenja i doprinosa odbijene i refraktirane zrake

Nakon što se odredi lokalno osvjetljenje, te doprinosi odbijene i refraktirane zrake potrebno ih je zbrojiti da se dobije konačno osvjetljenje u točki presjeka zrake sa scenom. Međutim, različiti objekti različito kombiniraju te doprinose. Na primjer, kod neprozirnog objekta udio doprinosa refraktirane zrake jednak je 0. Zbog toga se lokalna osvjetljenje (Iillum) i doprinos odbijene (Irefl) i refraktirane zrake (Irefr) kombiniraju na slijedeći način :

(3.10)

gdje su *k1* i *k2* koeficijenti od nula do jedan i predstavljaju udio odbijene, odnosno refraktirane zrake i karakteristike su materijala.

## Dodatne napomene



**Slika 3.9.**

**Slika 3.8**

Pogledajmo zanimljiv slučaj na slici 3.8. kada izvor svjetlosti ne baca svjetlo na promatranu točku. Očito je da tada vrijedi da je skalarni umnožak L i N manji od nule. U ovom slučaju ne računamo utjecaj difuzne i spekularne komponente.

Za bilo koju plohu, pa tako i za plašt kugle vrijedi da je vektor normale određen okomicom na plohu. Odabran smjer normale u općem je slučaju proizvoljan, tj. može pokazivati na jednu ili drugu stranu plohe, ali u našoj primjeni je važan. Potrebno je uvijek odabrati takav smjer normale da normala pokazuje na onu stranu s koje dolazi zraka. Funkcija *getNormal* uvijek vraća normalu koja pokazuje u smjeru od središta prema sjecištu. U slučaju da je skalarni umnožak V i N manji od nule potrebno je normali okrenuti smjer množenjem sa -1. Na slici 3.9 vidi se normala N2 koja je nastala okretanjem smjera. Uočite da se mijenja omjer n1 i n2 pri izlasku zrake iz kugle.

# Opis zadatka

Potrebno je implementirati opisane dijelove metode praćenja zrake, presjek sa scenom, Phongov model osvjetljenja, sjene, reflektirane i refraktirane zrake unutar koda zadanog Unity projekta. Unutar projekta definirane su skripte i objekti nad kojima će se raditi u sceni *RayTracingDemoBalls* koja se otvara klikom na tu scenu unutar *Assets* foldera. Potrebno je implementirati funkcije definirane u sjenčaru *RTComputeShader.compute* unutar direktorija *Assets->Scripts.* U skripti treba dodati vlastiti kod ispod komentara označenih sa TODO.

# Upute za rad

Na stranicama predmeta pronaći ćete *Unity* projekt koji sadrži kostur koji treba dopuniti kodom. Sve funkcionalnosti pišete unutar sjenčara *RTComputeShader.compute* koji je temeljito komentiran u kodu. Uz sjenčar u projektu se nalaze C# skripte *RayTracingMaterialSettings.cs i RayTracingComputeShader.cs* koje konfiguriraju materijale sfera i postavljaju sve potrebne varijable za sjenčar. Ove datoteke ne bi trebalo mijenjati ako nije potrebno. Metode koje je potrebno implementirati su komentirane i opisane su u ovom dokumentu. Rad sjenčara može se vidjeti tijekom *Play* načina rada u *Unityu.*

## Unity scena

Scena koju treba otvoriti u *Unity* editoru je *RayTracingDemoBalls.* Postavljene su tri sfere s tri zasebna materijala i velika sfera za pod. Sjenčar podržava samo sfere, drugi tipovi objekata nisu podržani. Svaka sfera ima svoj definirani materijal unutar *Assets -> Materials*, poziciju u sceni koja se može mijenjati, *Tag Sphere* i *Layer RTDXLayer.* Ako želite dodavati nove sfere morate postaviti točan *Tag* i *Layer* da se sfere prenesu u sjenčar. Najlakše je kopirati postojeću sferu, napraviti novi materijal i pomaknuti ju. Sfere se **ne mogu** micati tijekom pokretanja programa.

U sceni je definirano točkasto svijetlo *PointLight* sa oznakom *Light* koje će služiti kao glavni izvor svjetlosti u praćenju zrake. Pozicija svijetla i boja se može podesiti tijekom rada programa i sjenčar će se ažurirati. Podržavan je samo jedan izvor svijetla.

Kamera služi kao promatrač u sceni. Implementirano je kretanje kamere sa mišem i tipkovnicom unutar skripte *FreeFlyCamera.* Tijekom rada sjenčara kamerom se može gledati iz drugih kutova i ažurirati pozicija.

## Materijali

Svaka sfera ima definiran materijal *.mat* unutar *Unity* projekta. Novi material možete stvoriti desnim klikom unutar prozora s datotekama -> *Create -> Material* i nakon toga ga možete povući na sferu. Na svakoj sferi je postavljena skripta *RayTracingMaterialSettings* koja postavlja varijable bitne za praćenje zrake kao spekularni faktor i koeficijenti refleksije i refrakcije. Varijable se postavljaju u desnoj strain *Unity editora* i postavljaju se prije pokretanja scene. Boju sfere mijenjajte u postavkama materijala na sferi.

## RayTracingComputeShader.cs

Ova skripta postavlja sve potrebne varijable za sjenčar i pokreće (*dispatcha*) sam sjenčar. Kod je komentiran i proučite ga no nije ga potrebno mijenjati.

Skripta je prihvaćena na kameru i u njoj se može uključiti sjenčar tijekom rada programa i konfigurirati sve opcije vezane uz sjenčar. Većinu opcija se može isključiti ili uključiti tijekom rada programa za provjeru doprinosa svake komponente. One koje treba implementirati su opisane u sljedećoj listi. Opcije uključene u skripti su:

* Compute Shader – varijabla koja prima datoteku sjenčara
* Render Texture – slika koja se iscrtava na ekran
* Turn on – uključi/isključi sjenčar
* Black White – testiranje presjeka zraka sa scenom, bijelo ako ima presjeka, crno ako nema – potrebno implementirati
* Ambient – ambijentalno osvjetljenje – potrebno implementirati
* Diffuse – difuzno osvjetljenje– potrebno implementirati
* Specular – spekularno osvjetljenje– potrebno implementirati
* Reflection – doprinos reflektiranih zraka – potrebno implementirati
* Refraction – doprinos refraktiranih zraka – potrebno implementirati
* Shadows – jednostavna provjera sjena – potrebno implementirati
* Blinn Phong – drugačiji način izračuna spekularne komponente – dodatni zadatak
* Black Background – sve zrake koje nemaju presjek vraćaju crno, inače vraćaju *Unity* pozadinu
* Bounce number – dubina rekurzije refleksije i refrakcije – potrebno implementirati
* Minimal intensity – broj između 0 i 1 koji opisuje minimalni doprinos reflektiranih i refraktiranih zraka ispod kojeg će se prekinuti rekurzija
* Ambient color – boja ambijentalnog svijetla

## RTComputeShader.compute

Ovo je skripta sjenčara *Compute* pisana u HLSL jeziku. Sjenčari *Compute* su optimizirani za paralelnu obradu funkcija i korisni su u praćenju zrake. HLSL jezik je jako sličan C++u i trebao bi vam biti poznat. Glavni tip varijable koji ćete koristiti su *float* i *float3* ili *float4* (vektori sa 3 ili 4 člana) koji označavaju intenzitet boje u RGB sustavu. Važno je napomenuti da *Compute* sjenčari **ne podržavaju rekurziju** tako da implementaciju praćenja zrake je potrebno ostvariti u petlji. Kod je podijeljen u funkcije koje je potrebno implementirati. Potrebne funkcije i predlošci su opisani ispod. Promjene koda u sjenčaru se spreme instantno i aplikacija ne smije biti u *Play* načinu rada. Greške se mogu vidjeti u konzoli *Unity* okolinepored prozora *Project.* Tijekom implementacije koristite mogućnost kretanja kamere i micanja pozicije svijetla za provjeru ispravnog iscrtavanja ekrana. Nakon svakog koraka **slikajte zaslon**scene u *Play* modeu i priložite u izvještaj.

Stvaranje zrake od kamere do piksela je već implementirano u funkciji *CreateCameraRay.* Sjenčar koristi pomoćne strukture *RayTracingMaterial* koji opisuje materijal sfere, *Sphere* koji opisuje sferu, *Ray* koji opisuje zraku i *HitInfo* koji opisuje informacije vezane uz presjek zrake sa scenom.

Prvi korak je dopuniti funkciju *RaySphereIntersection* i omogućiti pronalazak presjeka zrake sa scenom. Funkcija se zove za svaku sferu unutar *CalculateRayCollision* funkcije koja je implementirana. Za provjeru presjeka zrake sa sferom koristite formule iz uputa i napunite strukturu *HitInfo* sa svim potrebnim informacija. Na informacije iz strukture *HitInfo* će se oslanjati svi daljnji izračuni. Ispravan presjek možete lagano testirati tako da uključite opciju *BlackWhite* koja će vratiti bijelu boju unutar funkcije *PhongShading* ako je zraka pogodila sferu unutar scene. Uspješnu implementaciju *BlackWhite* moda **slikajte zaslon**i priložite uz izvještaj.

Nakon implementacije presjeka treba implementirati osnovni *Phong* model osvjetljenja. Nadopunite funkciju *PhongShading* tako da implementirate *AmbientLighting, DiffuseLighting i SpecularLighting.* Funkcije ispunite prema uputama i isprobajte svaki dio zasebno i zajedno. Nakon svake uspješne implementacije priložite **snimku zaslona***.* Ako spekularno osvjetljenje nije primjetljivo nemojte zaboraviti podesiti spekularni faktor na materijalima sfera.

Nakon implementacije *Phongovog* modela osvjetljenja treba implementirati jednostavnu provjeru za sjene. Nadopunite funkciju *Shadows* sa istom provjerom presjeka sa scenom, ali za zraku sjene. Ako zraka ima sjenu, doprinos difuzne i spekularne komponente treba biti 0. Utjecaj sjene se lagano provjeri na velikoj kugli koja predstavlja pod i pomoću micanja svijetla u sceni. Utjecaj sjene priložite uz izvještaj.

U CSMain funkciji nakon provjere presjeka sa scenom i računanja *Phongovog* modela osvjetljenja treba implementirati funkciju *TraceRay* koja računa utjecaj reflektiranih i refraktiranih zraka. Funkcija prima zraku, dubinu i tip. Tip 0 označava reflektirane zrake, a 1 refraktirane zrake. U funkciji se prati ukupna boja (*tracedColor)* i *hitInfoPrevious* koji označava *hitInfo* prijašnje zrake u nizu. On se predaje u provjeru presjeka za zrakom i koristi se da se spriječi testiranje presjeka sa istom kuglom. Obje vrste praćenja zrake treba implementirati pomoću petlje. Uvjet za prestanak petlje postavite da bude ili konačan broj (parameter *dubina*) ili ako je energija zrake manja od parametra *minimalIntensity* (zraka je dovoljno slaba da se utjecaj zanemaruje).

Pri implemenatciji refleksije implementirajte funkciju *getReflectedVector* po formulama unutar poglavlja 3.5. Imajte na umu da koeficijent refleksije materijala ostaje isti pošto se računaju zrcaljenja na istoj kugli. Ako zraka ima presjek sa scenom, izračunajte *Phong* osvjetljenje i izračunajte novu zraku pomoću *getReflectedRay*. Nemojte zaboraviti umanjiti energiju zrake kao simulaciju gubljenja snage nakon odbijanja!

Pri implementaciji refrakcije koristite isti pristup samo sa funkcija *getRefractedVector* i *getRefractedRay* sa formulama iz poglavlja 3.6. Kod refrakcije, koeficijent refrakcije materijala se mijenja sa svakim presjekom jer zraka putuje kroz različite kugle. Za obje metode priložite **snimku zaslona***.*

Nakon implementacija svakog dijela algoritma praćenja zraka kombinirajte različite opcije i provjerite da sve ispravno radi. Na kraju priložite jedan ili više **snimki zaslona**koji prikazuju sve opcije uključene istovremeno (osim *BlackWhite*).

# Predavanje rezultata vježbe

Rezultati vježbe se predaju zapakirani u arhivu OVO-V2- Rezultati-<ImePrezime>.zip koja treba sadržavati:

* Izvještaj o izvođenju vježbe koji treba sadržavati:
  + Kratak opis postupka iscrtavanja (max. jedna stranica)
  + Iscrtane slike svih faza iscrtavanja prema uputama za izradu vježbe: (1) slika siluete scene (samo presjek, *BlackWhite* opcija), (2) slika s ambijentnom komponentom, (3) slika s ambijentnom i difuznom komponentom, (4) slika s ambijentnom, difuznom i spekularnom komponentom (puno lokalno osvjetljenje), (5) slika s lokalnim osvjetljenjem i sjenama, (6) slika s lokalnim osvjetljenjem, sjenama i reflektiranim zrakama, te (7) potpuna slika: lokalno osvjetljenje, refleksija, refrakcija i sjene. Svaku sliku treba komentirati, tj. objasniti razlike u odnosu na prethodnu sliku i zašto one nastaju.
* Dobro komentirani izvorni kod zadatka (potrebno je dobro objasniti što se radi u svakom dijelu dokumenta koji ste sami dodavali) te izvršnu datoteku.

Navedena arhiva treba biti predana korištenjem aplikacije Moodle dostupne preko web stranica predmeta.

**Napomena:** Rezultati se šalju isključivo preko gore navedene aplikacije. U slučaju problema, javiti se email-om na adresu vo@fer.hr. Sačuvajte kopiju poslanih rezultata.

# Napredni zadaci (ovaj dio nije obavezan)

## Proširenje algoritma na druge geometrijske oblike

Napraviti algoritam praćenja zrake za bilo koje drugo geometrijsko tijelo osim kugle. Recimo najopćenitije bi bilo napraviti za trokut jer se onda od trokuta može sastaviti velik broj geometrijskih tijela. Ostali primjeri su elipsoid, kocka, torus, stožac itd.

## Implementacija Blinn-Phong modela osvjetljenja

Implementirati izračun spekularnog odsjaja pomoću Blinn-Phong metode.

## Tekstura

Implementirati teksturiranje slike na geometriju.

## Napredniji algoritam zrake za ispitivanje sjena

Implementirati takav algoritam zrake za ispitivanje sjena koji će u obzir uzeti sva tijela kroz koja je prošao i pomnožiti njihove udjele refraktirane zrake. Takav umnožak u programu iskoristiti kao koeficijent kojim množimo iznos spekularne i difuzne komponente u točki promatranja.

1. Zbog ograničenosti računala u prikazu realnih brojeva u nekim nam slučajevima dolazi do toga da je desna strana izraza veća od nule unatoč tome što se hvatište nalazi unutar kugle. Stoga koristimo korekcijski faktor Epsilon koji mora biti dovoljno malen da izraz zadrži smisao, a obuhvati takve slučajeve. [↑](#footnote-ref-1)