SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2379

ALGORITAM PRAĆENJA ZRAKE I PRAĆENJA PUTA ZA PRIKAZIVANJE SCENA U STVARNOM VREMENU

Luka Radivoj

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2379

ALGORITAM PRAĆENJA ZRAKE I PRAĆENJA PUTA ZA PRIKAZIVANJE SCENA U STVARNOM VREMENU

Luka Radivoj

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

DIPLOMSKI ZADATAK br. 2379

Pristupnik:	**Luka Radivoj (0036499188)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Željka Mihajlović

Zadatak: **Algoritam praćenja zrake i praćenja puta za prikazivanje scena u stvarnom vremenu**

Opis zadatka:

Proučiti matematičke osnove i generalne koncepte vezane uz algoritam praćenja zrake te praćenja puta poput jednadžbe iscrtavanja i različitih BDRF funkcija. Načiniti pregled stanja područja vezano uz sklopovsku akceleracijsku potporu i programska sučelja vezana uz ovu problematiku. Razmotriti implementacijske algoritme, njihova ograničenja te mogućnosti proširenja te mogućnosti implementacije korištenjem programskog alata Unity. Implementirati scene u alatu koje demonstriraju kako algoritmi poboljšavaju kvalitetu prikaza. Načiniti testiranje na nizu primjera. Analizirati i ocijeniti ostvarene rezultate. Diskutirati upotrebljivost ostvarenih rezultata kao i moguća proširenja. Izraditi odgovarajući programski proizvod. Koristiti programski alat Unity. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2021.

Zahvaljujem se cijeloj svojoj obitelji i svim svojim prijateljima na velikoj podršci tokom cijelog studija i pisanja ovog diplomskog rada. Posebice se zahvaljujem svojoj majci Suzani Radivoj, svome ocu Tomislavu Radivoju, svojoj sestri Lari Radivoj i Ivanu Žuljeviću-Mikasu.

Sadržaj

Uvod

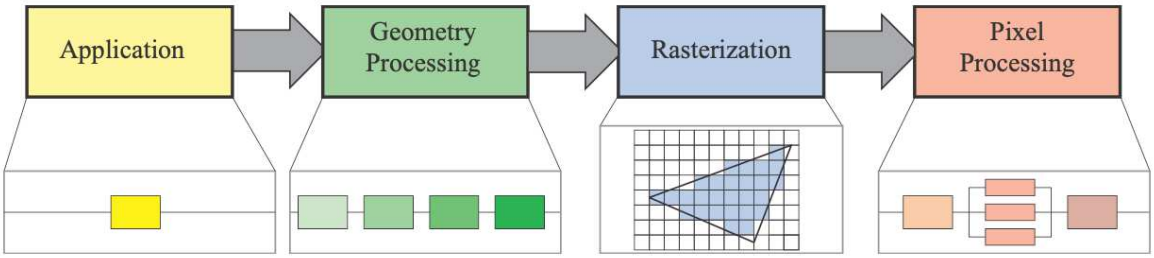
Algoritam praćenja zrake i njegove inačice, koje uključuju i algoritam praćenja puta, kroz dugi niz godina su se u području računalne grafike koristili u svrhu iscrtavanja virtualnih okruženja isključivo u sustavima koji nisu zahtijevali interaktivnost, odnosno u sustavima u kojima se jedan slikovni okvir može generirati i po nekoliko minuta, sati ili čak dana. S druge strane, za sustave koji su zahtijevali interaktivnost, odnosno za sustave u kojima se jedan slikovni okvir smije generirati samo nekoliko desetaka milisekundi, primarno se koristio algoritam rasterizacije. Odnos familije algoritama praćenja zrake i algoritma rasterizacije klasičan je primjer obrnuto proporcionalnog odnosa kvalitete prikaza i brzine iscrtavanja. Familija algoritama praćenja zrake prirodno simulira transport svijetlosti kroz virtualno okruženje i njezinu interakciju s objektima čiji materijali imaju raznolika fizikalna svojstva. No, ova familija algoritama zahtjeva veliku količinu računalnih resursa te je veoma zahtjevna za izvođenje. S druge strane, algoritam rasterizacije nastoji što vjernije simulirati fizikalnu fenomenologiju vezanu uz propagaciju svijetlosti i njezinu interakciju s materijom. Nakon dugog niza godina razvoja rasterizacijskih metoda, moguće je simulirati navedenu fizikalnu fenomenologiju na zavidnoj razini kvalitete. No usprkos iznimno velikom napretku, rasterizacijske metode koje nastoje simulirati navedenu fizikalnu fenomenologiju ograničene su funkcijskim cjevovodom koji se oslanja na algoritam rasterizacije, te zbog toga, bez obzira na to koliko je određena metoda napredna i kvalitetna, jednostavno ne mogu prirodno simulirati navedenu fizikalnu fenomenologiju pa su stoga u njihovim rezultatima vidljive pogreške i razni vizualni artefakti. Također, za određenu fizikalnu fenomenologiju, ne postoje rasterizacijske metode koje ju mogu simulirati. No bez obzira na ove nedostatke, algoritam rasterizacije je izrazito brz i učinkovit algoritam, te je stoga pogodan za primjene u interaktivnim sustavima u kojima je vrijedno žrtvovati dio kvalitete i vjernosti prikaza kako bi se slikovni okviri mogli generirati velikom brzinom. No, paralelno s razvijem familije algoritama praćenja zrake i algoritma rasterizacije i pripadnih metoda, razvijalo se i grafičko sklopovlje. Razvoj grafičkog sklopovlja za interaktivne sustave primarno se fokusirao na napretke u sirovoj snazi sklopovlja, kao i na sklopovske akceleracijske strukture koje su ubrzavale algoritam

rasterizacije i njemu pripadne metode. Usprkos velikim ulaganjima u poboljšanje algoritma rasterizacije, kao i u navedene sklopovske akceleracijske strukture, tokom godina razvoja, familija algoritama praćenja zrake nikada nije zaboravljena te je uvijek predstavljala nedostižan cilj u području iscrtavanja virtualnih okruženja u interaktivnim sustavima u kojima je potrebno generirati slikovne okvire u stvarnom vremenu. O ovome govori i misao koju je izrekao David Kirk o tome kako je algoritam praćenja zrake tehnologija budućnosti i to će zauvijek ostati. Ipak, usprkos generalnom pesimizmu vezanom za primjenu ovih algoritama u interaktivnim sustavima, razvojem grafičkog sklopovlja, te predstavljanjem sklopovskih akceleracijskih struktura u sklopu RTX platforme tvrtke *Nvidia*, kao i generalnom napretku u programskim akceleracijskim strukturama za ovu familiju algoritama, u današnje vrijeme postalo je moguće generirati prikaze virtualnih okruženja u stvarnom vremenu pomoću algoritama iz familije algoritama praćenja zrake. Područje primjene ovih algoritama na iscrtavanje u interaktivnim sustavima izrazito je mlado, te se rapidno razvija. U sklopu ovog diplomskog rada razmotrit će se pomnije familija algoritama praćenja zrake i njezin odnos s klasičnim algoritmom rasterizacije, kao i prednosti i mane obiju vrsta algoritama. Pomnije će se promotriti fizikalni i matematički modeli u pozadini iscrtavanja virtualnih okruženja te njihov odnos s algoritmom rasterizacije i familijom algoritama praćenja zrake. Razmotrit će se sklopovske, kao i programske akceleracijske strukture koje omogućuju primjenu familije algoritama praćenja zrake u interaktivnim sustavima. Dat će se pregled aplikacijskog programskog sučelja DXR i njegove arhitekture koja prati sklopovske akceleracijske strukture prisutne u grafičkom sklopovlju RTX familije. Također će se izraditi i analizirati praktični primjer u alatu *Unity* koji će jasno demonstrirati napretke koje familija algoritama praćenja zrake donosi u područje iscrtavanja virtualnih scena u interaktivnim sustavima. Za kraj će se razmotriti i implikacije koje primjena ove familije algoritama u interaktivnim sustavima ima na cjelokupan proces razvoja programskih proizvoda, a ne samo na kvalitetu dobivenog prikaza.

1. Iscrtavanje virtualnih scena

Na samom početku ovog diplomskog rada razmotrit ćemo proces iscrtavanja virtualnih okruženja u stvarnom vremenu korištenjem algoritma rasterizacije i njemu pripadnih metoda. Razmatranjem ovog procesa dobit ćemo bolji uvid u način na koji slikovni okvir nastaje korištenjem ovog algoritma te koje su prednosti i mane ovakvog pristupa iscrtavanju virtualnih okruženja. Iscrtavanje jednog slikovnog okvira u stvarnom vremenu se intuitivno može predstaviti modelom cjevovoda. Na slici 1.1 prikazan je funkcijski cjevovod na najvišoj razini apstrakcije koji opisuje proces iscrtavanja slikovnog okvira korištenjem algoritma rasterizacije. Na dijagramu sa slike 1.1 je vidljivo kako se proces iscrtavanja slikovnog okvira korištenjem algoritma rasterizacije može podijeliti u 4 funkcijske cjeline [1]:

- Aplikacijska faza (engl. *Application phase*)
- Faza obrade geometrije (engl. *Geometry Processing*)
- Faza rasterizacije (engl. *Rasterization*)
- Faza obrade slikovnih elemenata (engl. *Pixel Processing*)

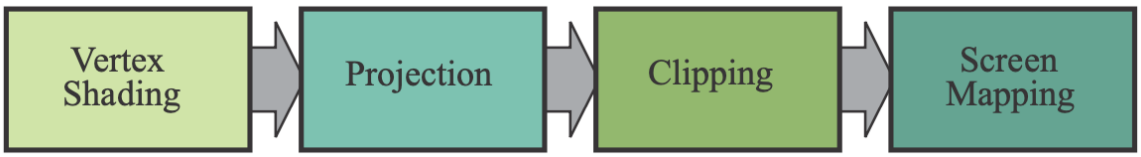


Slika 1.1: *Standardni funkcijski cjevovod za iscrtavanje virtualnih okruženja u interaktivnim sustavima*

Svratimo pozornost na nekoliko važnih činjenica. Prva je ta da su ranije navedene faze, faze funkcijskog cjevovoda. Funkcijski cjevovod je cjevovod visoke razine apstrakcije koji opisuje proces iscrtavanja na temelju općenitih algoritama i postupaka. Zbog toga faze funkcijskog cjevovoda samo propisuju koji su zadaci pojedine faze, koji su njezini ulazi a koji izlazi. Faze funkcijskog cjevovoda ne propisuju točan način implementacije pojedine faze unutar konkretnog sustava za iscrtavanje. Druga činjenica je ta da se svaka od navedenih funkcijskih faza može

promatrati kao zaseban cjevovod jer ju je moguće nadalje podijeliti na manje faze od kojih svaka manja faza ima svoju zadaću, kao i ulaze i izlaze. [1] S druge strane, moguće je spojiti dvije ranije navedene funkcijske faze u jednu veću u kontekstu konkretne implementacije u sklopu sustava za iscrtavanje. U nastavku ćemo opisati svaku od navedenih funkcijskih faza. Proces iscrtavanja slikovnog okvira započinje s aplikacijskom fazom. [1] Aplikacijska faza je jedina od sve četiri funkcijske faze koja se tradicionalno izvodi na procesoru opće namjene (engl. *Central Processing Unit*, *CPU*). Usprkos tome što zadaci ove faze mogu biti proizvoljni, u općenitom slučaju se svode na ažuriranje svojstava objekata u virtualnom okruženju. Tako se unutar aplikacijske faze odrađuju zadaci poput pomaka objekata, detekcije kolizije, animacije i sličnog. Također, unutar aplikacijske faze se obrađuju informacije poput korisničkog unosa te određuje njihov utjecaj na objekte unutar virtualnog okruženja. Ranije je napomenuto kako se ova faza tradicionalno odrađuje pomoću procesora opće namjene. U recentnije vrijeme, zadatke aplikacijske faze moguće je odraditi pomoću grafičkog procesora (engl. *Graphics Processing Unit*, GPU) [1]. Izvođenje zadataka aplikacijske faze na grafičkom procesoru je moguće pomoću procesora za sjenčanje za računanje (engl. *Compute Shader*). Ovi procesori za sjenčanje su korisni za izvođenje općenitih zadataka koji imaju visoki stupanj paralelnosti. Poslije aplikacijske faze slijedi faza obrade geometrije. Ovo je prva faza u funkcijskom cjevovodu koja se izvodi na grafičkom procesoru. Fazu obrade geometrije moguće je podijeliti na manje funkcijske faze. Slika 1.2 prikazuje uobičajenu podjelu faze obrade geometrije. [1] Sa slike 1.2 je vidljivo kako faza obrade geometrije može podijeliti na sljedeće manje faze:

- Faza sjenčanja vrhova (engl. *Vertex Shading*)
- Fazu projekcije (engl. *Projection*)
- Fazu obrezivanja (engl. *Clipping*)
- Fazu preslikavanje na ekran (engl. *Screen Mapping*)



Slika 1.2: *Podjela funkcijske faze obrade geometrije*

Faza sjenčanja vrhova ima dva glavna zadatka [1]. Prvi od njih je izračun koordinata pozicije pojedinog vrha. Drugi zadatak je izračun dodatnih podataka za svaki pojedini vrh. Ovi podaci mogu biti na primjer normale ili koordinate tekstura. Dodatni podaci ovise o konkretnoj implementaciji faze sjenčanja vrhova, dok je konkretna implementacija diktirana operacijama u fazama koje slijede u cjevovodu, a koje koriste dodatne podatke nastale u fazi sjenčanja vrhova. Druga manja faza je faza projekcije. [1] Faza projekcije transformira cijelu scenu na način da se nakon transformacije kamera nalazi u ishodištu koordinatnog sustava, te da gleda u smjeru negativne z-osi, dok y-os pokazuje prema gore, a x-os se nalazi s desna. Smjer pogleda kamere ne mora nužno biti u smjeru negativne z-osi, pa stoga u određenim implementacijama a kamera gleda u smjeru pozitivne z-osi. No transformacija iz pojedine konvencije u onu drugu je trivijalna, te zbog toga ne tvori značajnu razliku o ovom razmatranju. Nakon ove transformacije, moguće je izvršiti projekciju. Standardne projekcije u računalnoj grafici su ortografska i perspektivna projekcija. [1] Nakon projekcije svi se vrhovi, koji su vidljivi kameri, odnosno koji se nalaze u vidljivom volumenu kamere, nalaze u takozvanoj jediničnoj kocki kojoj se ekstremi nalaze u vrhovima (-1,-1,-1) i (1,1,1). Stoga ova kocka predstavlja vidljivi volumen kamere. [1] Kod perspektivne projekcije vidljivi volumen kamere je krnja piramida, dok je kod ortografske projekcije ovaj volumen kvadar. Ako navedena jedinična kocka predstavlja vidljivi volumen kamere, svi vrhovi van ove jedinične kocke nisu vidljivi kameri te ih je moguće odbaciti. Upravo ovo je zadatak faze obrezivanja. [1] Ova faza u odbacuje svu geometriju koja se u potpunosti nalazi van jedinične kocke dobivene projekcijom, dok za geometriju koja siječe jediničnu kocku nalazi nove vrhove koje se nalaze upravo na stranicama navedene jedinične kocke. Nakon što završi faza obrezivanja, sva geometrija, odnosno njezini vrhovi, koja je preživjela fazu obrezivanja se prebacuje u koordinatni sustav ekrana. Ovaj proces izvršava faza preslikavanja na ekran. Završetkom cjelokupne faze obrade geometrije dobivamo obrađene vrhove s pripadajućim dodatnim podacima koji se nalaze u koordinatnom sustavu ekrana. Sljedeće dvije funkcijske faze su odgovorne za određivanje vidljivosti i određivanje konačne boje određenog slikovnog elementa ekrana. Sve faze funkcijskog cjevovoda kojeg trenutno razmatrano usko su vezane uz algoritam rasterizacije. No posljednje dvije funkcijske faze su najviše ovisne o ovom algoritmu.

1.1. Algoritam rasterizacije

Rasterizacija je metoda pomoću koje se određuje koji se slikovni elementi, odnosno pikseli, nalaze unutar određene primitive. Drugim riječima, određujemo je li određena primitiva vidljiva gledajući iz zadanog slikovnog elementa. Zbog toga se proces rasterizacije može opisati pomoću dvostruke petlje [1]:

```
Za svaku primitivu P u skupu svih primitivi:
    Za svaki slikovni element S u skupu slikovnih elemenata:
        Odredi je li S unutar P
```

Funkcijska faza rasterizacije dobila je ime upravo prema ovom algoritmu za određivanje vidljivosti zbog toga što je algoritam rasterizacije posto standardni algoritam za ovaj razrješavanje ovog problema prilikom iscrtavanja slikovnih okvira u interaktivnim sustavima. Funkcijska faza rasterizacije se sastoji od dvije manje funkcijske faze. [1] Ove dvije manje faze su sljedeće:

- Faza inicijalizacije trokuta (engl. *Triangle Setup*)
- Faza prolaska kroz trokute (engl. *Triangle Traversal*)

Ove dvije manje funkcijske faze u svojim nazivima referenciraju primitivu trokuta. Usprkos tome, faza rasterizacije podržava i ostale vrste primitiva, no u području iscrtavanja virtualnih okruženja u stvarnom vremenu trokuti su relativno standardan odabir za baznu primitivu s kojom određeni sustav za iscrtavanje radi.

1.1.1. Faza inicijalizacije trokuta

[1] U ovoj manjoj funkcijskoj fazi izračunavaju se raznorazna svojstva trokuta na temelju podataka koji su nastali kao produkt faze obrade geometrije. Neki primjeri ovakvih podataka koji su vezani uz trokute su diferencijali, rubne jednadžbe i slični podaci koji se koriste u drugoj manjoj funkcijskoj fazi faze rasterizacije. Ovi podaci se također koriste za interpolaciju dodatnih podataka vezanih uz vrhove trokuta koji su nastali u fazi sjenčanja vrhova. Procesi koji se odvijaju u ovoj fazi uobičajeno nisu pod kontrolom programera.

1.1.2. Faza prolaska kroz trokute

U ovoj manjoj funkcijskoj fazi se određuje vidljivost pojedine primitive gledajući iz pojedinog slikovnog elementa [1]. Umjesto vidljivosti možemo govoriti i o prekrivenosti. Za svaki slikovni element donosimo binarnu odluku, odnosno slikovni element se ili nalazi unutar primitive ili se nalazi izvan primitive. Ova binarna odluka unosi pogreške u prikazu zadane primitive na rasteru. Što je raster gušći, odnosno što više slikovnih elemenata raster sadrži, pogreška će biti manja, odnosno dobit ćemo kvalitetniji prikaz primitive na rasteru. Kvaliteta prikaza primitive također ovisi i o odnosu veličine primitive naspram veličine slikovnog elementa. Za svaki slikovni element koji se nalazi unutar primitive generira se fragment koji se prosljeđuje u sljedeću funkcijsku fazu cjevovoda. Ovaj fragment sadrži različite podatke koji se dobivaju interpolacijom podataka koji su vezani uz vrhove primitive. Podaci koji se interpoliraju su primjerice dubina određenog fragmenta, kao i podaci koje je za vrhove primitive generirala faza sjenčanja vrhova. Odluku nalazi li se određeni slikovni element unutar određene primitive možemo donijeti na različite načine. [1] Jedan od načina je da taj da se slikovni element nalazi unutar primitive ako se njegovo središte nalazi unutar primitive. Ovaj način uzorkovanja (engl. *Sampling*) nije jedini te postoje razni načini pomoću kojih možemo odrediti nalazi li se slikovni element unutar primitive. Uz ranije navedenu gustoću rastera, odabir metode uzorkovanja igra vrlo važnu ulogu u kvaliteti konačnog prikaza primitive na rasteru.

Kao što je već ranije napomenuto, faza rasterizacije za svaku primitivu generira fragmente za one slikovne elemente koji se nalaze u primitivi. Ovi fragmenti su izlazni podaci faze rasterizacije Ovaj izlaz predstavlja ulaz u sljedeću i posljednju funkcijsku fazu cjevovoda. Ova funkcijska faza je faza obrade slikovnih elemenata, koja se također često naziva i fazom obrade fragmenata. [1] Ova funkcijska faza se ponovno sastoji od dvije manje funkcijske faze. Manje funkcijske faze su sljedeće:

- Faza sjenčanja slikovnih elemenata (engl. *Pixel Shading*)
- Faza stapanja (engl. *Merging*)

U sklopu funkcijske faze sjenčanja slikovnih elemenata, odnosno sjenčanja fragmenata, obavljaju se operacije na razini slikovnog elementa na temelju

interpoliranih podataka vezanih uz pripadajući fragment koje je generirala faza rasterizacije. Neke od operacija koje se odvijaju u ovoj manjoj fazi su primjerice teksturiranje ili izračunavanje osvjetljenja. Rezultat ove faze je jedna ili više boja koje se prosljeđuju u fazu stapanja. [1] Informacije za svaki pojedinačni slikovni element u konačnici se spremaju u spremnik boja (engl. *Colour Buffer*). Glavni zadatak faze stapanja je odlučiti na koji način kombinirati nove podatke dobivene sjenčanjem slikovnih elemenata koji se nalaze unutar primitive koja se trenutno obrađuje s već postojećim podacima koji se nalaze u spremniku boja. Kombiniranje podataka može biti svakojako. Elementarno kombiniranje je ono u kojem se novi podatak odbacuje ako je novi fragment dalje od kamere od fragmenta koji se trenutno nalazi u spremniku boja. Odluka o tome koji fragment je bliži kameri se donosi na temelju Z-spremnika (engl. *Z-buffer*). Z-spremnik je spremnik jednake veličine kao i spremnik boje, no u njemu se za svaki slikovni element sprema udaljenost do za sada najbližeg fragmenta određene, već obrađene primitive. Time dolazimo do zaključka da je jedan od zadataka faze stapanja i sortiranje primitiva, odnosno briga o tome da se za svaki slikovni element prikaže fragment primitive koja je najbliža kameri. No, spremnik boje i Z-spremnik, kao i pripadne informacije koje oni bilježe, nisu jedini spremnici i informacije koje je moguće identificirati u fazi stapanja. Tako je, primjerice, u sklopu faze stapanja moguće identificirati i informaciju o alfa kanalu (engl. *Alpha Channel*) koji služi za bilježenje prozirnosti određenog slikovnog elementa. Sve operacije koje se odvijaju u fazi stapanja se nazivaju rasterizacijskim operacijama (engl. *Raster Operations, ROP*). [1] Standardna praksa u konkretnim implementacijama sustava za iscrtavanje je da ove operacije nisu programibilne, no nude visok stupanj podesivosti. S druge strane postoje sustavi koji nude programibilne operacije stapanja [1]. Iz navedenog je se lako može uočiti da je faza obrade slikovnih elemenata, usprkos tomu što je odvojena od funkcijske faze rasterizacije, visoko ovisna o njoj te su operacije koje se odvijaju u fazi obrade slikovnih elemenata, a pogotovo u manjoj funkcijskoj fazi stapanja zavisne o rasterizacijskom postupku te njime ograničene. Ovo opažanje će biti važno u danjim razmatranjima prednosti metoda baziranih na familiji algoritama praćenja zrake naspram rasterizacijskih metoda.

U nastavku ćemo razmotriti alternativu rasterizacijskom postupku kao metodi određivanja vidljivosti, odnosno prekrivenosti, te različitim operacijama u fazi obrade slikovnih elemenata koje su zavisne o algoritmu rasterizacije.

1.2. Algoritam praćenja zrake

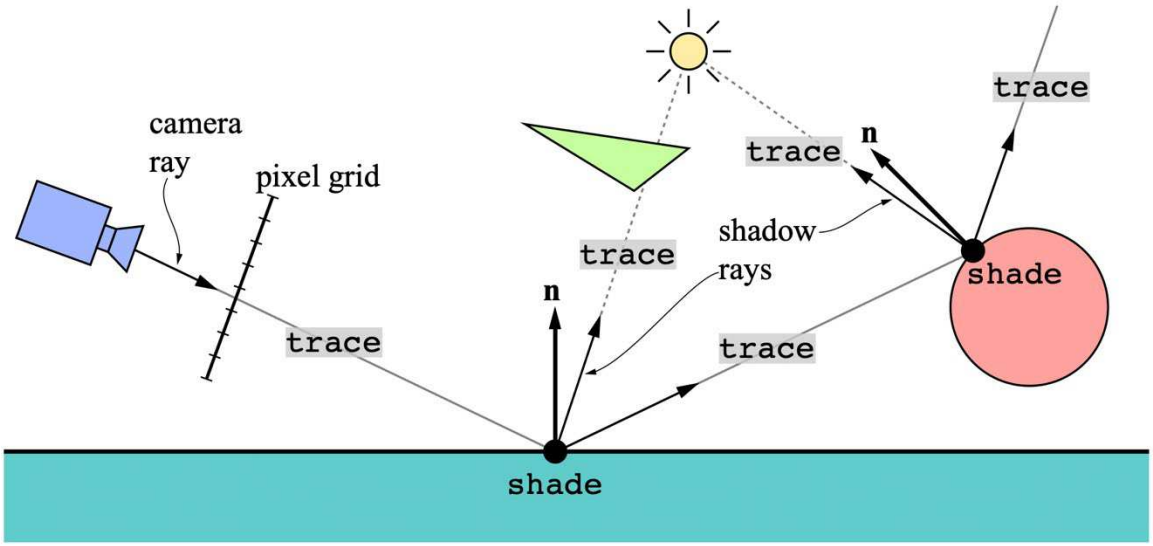
Razmotrimo najprije algoritam praćenja zrake u kontekstu određivanja vidljivosti, odnosno prekrivenosti. U ovom kontekstu, algoritam praćenja zrake možemo opisati pomoću dvije ugniježđene petlje [1]:

```
Za svaki slikovni element S u skupu slikovnih elemenata:
    Za svaku primitivu P u skupu primitiva
        Odredi sječe li zraka kroz S primitivu P
```

Ovaj proces, u kojem ispaljujemo zraku i nalazimo koje se sve primitive nalaze na njezinom putu se često naziva proces bacanja zrake (engl. *Ray Casting*) [2]. No, moć algoritma praćenja zrake ne leži u određivanju vidljivosti, odnosno prekrivenosti, nego u tome da intuitivno prati fizikalnu propagaciju svijetlosti i njezinu interakciju s objektima u sceni. Stoga algoritam praćenja zrake uzima proces bacanja zraka samo kao alat pomoću kojeg se za svaki slikovni element ispaljuje zraka u virtualno okruženje te se rekurzivno prati njezin put kroz scenu. Prije pregleda samog algoritma, potrebno je formalizirati što je zapravo zraka. Zraku možemo analitički zapisati na sljedeći način [1]:

$$q(t) = o + td \tag{1}$$

Ako bolje sagledamo formulu (1) možemo uvidjeti da je zraka određena početnom točkom *o* i vektorom smjera *d*. Parametrom *t* opisujemo pojedinu točku na zraci. Parametar *t* nam govori koliko se moramo pomaknuti u smjeru *d* od početne točke *o* kako bi smo došli do željene točke. Kako bi smo stekli intuiciju o tome kako algoritam praćenja zrake funkcionira, promotrit ćemo put jedne zrake kroz virtualno okruženje. [1] Na slici 1.2.1 se nalazi ilustracija provođenja algoritma za jednu zraku koja se ispaljuje iz kamere, te za koju se prati put u virtualnom okruženju.

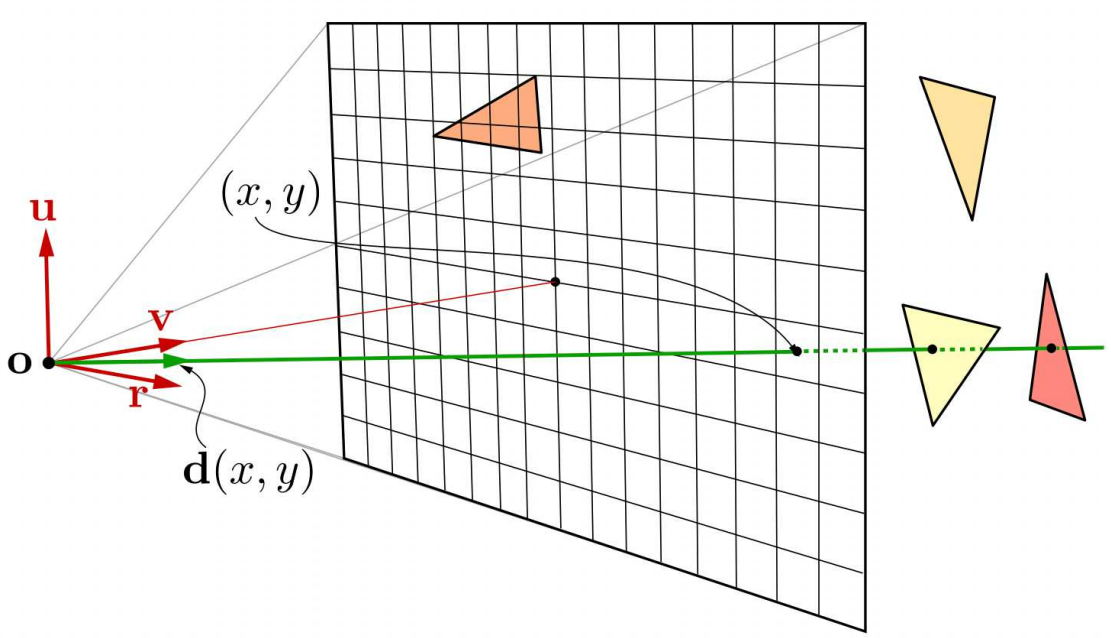


Slika 1.2.1: *Praćenje zadane zrake kamere kroz virtualno okruženje*

No, prije nego li možemo pratiti zraku i njezin put kroz virtualnu scenu, navedenu zraku je potrebno generirati. Zraka koja se generira na samom početku algoritma se naziva zraka kamere, zraka pogleda ili zraka oka (engl. *Camera Ray, View Ray, Eye Ray*). Ova zraka ima istu početnu točku kao i točka u kojoj se nalazi kamera, dok joj je vektor smjera određen tom istom početnom točkom i pozicijom slikovnog elementa na rasteru. Nakon što konstruiramo zraku, puštamo je u virtualno okruženje. [1] Standardne zrake, u koje spadaju i zrake kamere, uobičajeno uz sebe imaju vezan teret (engl. *Payload*). Ovaj teret služi za prijenos informacija o zraci, poput trenutne energije zrake, tokom njezinog puta kroz virtualno okruženje. Sljedeća informacija koja nam je potrebna je informacija o presjeku zrake s nekim od objekata ili točnije primitiva u sceni. Uobičajeno je potrebno način najbliži presjek, odnosno onaj objekt ili primitivu koju zraka sječe, a najbliža je početnoj točki zrake. U kontekstu naivne implementacije, i analogno ranije navedenoj dvostrukoj petlji za određivanje vidljivosti, proces pronalaska presjeka zahtjeva iteraciju po svim primitivama koje sačinjavaju virtualnu scenu te testiranje presjeka zrake i svake od navedenih primitiva. Kada je pronađen odgovarajući presjek, u točki presjeka se odvija proces sjenčanja. Na slici 1.2.1 je vidljivo da je najbliži presjek za zraku kamere onaj s površinom poda. Proces sjenčanja je proizvoljan. Relativno standardna implementacija se sastoji od provjere nalazi li se točka presjeka u sceni, od upijanja dijela svijetlosti i refleksije drugog dijela. Upijanje dijela svijetlosti je moguće modelirati upijanjem dijela energije zrake. No kako bi smo odredili je li točka

presjeka u sceni, ili odredili što se reflektira u točki presjeka, proces sjenčanja može generirati nove zrake. Proces stvaranja novih zraka je također vidljiv na slici 1.2.1. Na slici je vidljivo kako se u točki presjeka zrake s podom stvaraju dvije nove zrake. Razmotrimo najprije zraku koja ide iz točke presjeka prema izvoru svijetlosti u sceni. Ova vrsta zraka se uobičajeno naziva zraka sjene (engl. *Shadow Ray*) i služi za provjeru nalazi li se točka presjeka u sceni. Provjera je relativno jednostavna zbog toga što je potrebno odrediti postoji li presjek zrake s bilo kojim objektom koji se nalazi između točke presjek i izvora svijetlosti. Ako barem jedan takav presjek postoji, to znači da se točka presjeka nalazi u sjeni u odnosu na izvor svijetlosti prema kojem je ispaljena zraka. Zrake sjena se mogu generirati i poslati prema svakom od izvora svijetlosti u sceni. Zrake ove vrste također nemaju nikakav teret vezan uz sebe zbog toga što je potrebno odrediti samo postoji li presjek ili ne. Druga vrsta zrake koja se generira u točki presjeka je zraka refleksije (engl. *Reflection Ray*). Ova zraka se generira pomoću zakona refleksije ovisno o kutu između zrake i normale površine. Ova vrsta zraka služi za određivanje onoga što je vidljivo u refleksiji objekta. Ova vrsta zraka se generira za reflektivne materijale poput metala ili ogledala. Kada stvorimo ovakvu vrstu zraka, pratimo njezin put dalje kroz virtualno okruženje, odnosno ponovno je potrebno odrediti najbliži presjek ove novonastale zrake s objektima u virtualnom okruženju. Kada se pronađe takav presjek, ponovno se u novoj točki presjeka odvija proces sjenčanja koji zauzvrat može generirati svoje zrake sjena i zrake refleksije. Upravo iz ovog opažanja je vidljivo zašto je algoritam praćenja zrake rekurzivan. Kao i sa svakim rekurzivnim algoritmom, moramo odrediti uvjet zaustavljanja. Kod algoritma praćenja zrake tradicionalno se pojavljuju tri uvjeta zaustavljanja koja se koriste zajedno. Prvi uvjet je situacija u kojoj se zraka sječe s izvorom svijetlosti. Drugi uvjet je situacija u kojoj ne postoji presjek zrake s nekim objektom u sceni. U ovom slučaju možemo zraki pripisati neku konstantnu vrijednost ili možemo provesti uzorkovanje primjerice teksture neba. Treći uvjet je najjednostavniji. Ovaj slučaj se odnosi na prekoračenje predviđenog broja odbijanja (engl. *Bounce*). Ovaj broj odbijanja se određuje iskustveno te služi za prekid praćenja u slučaju kada postoji veliki broj međusobnih refleksija između dva predmeta, odnosno kada bi smo došli do prva dva uvjeta nakon jako velikog broja odbijanja. Kada dođemo do jednog od ova tri uvjeta zaustavljanja, rekurzivno sjenčamo sve točke presjeka na putu koji je zraka prošla kroz virtualno okruženje. Ako nadalje sagledamo situaciju na slici 1.2.1, možemo primijetiti kako reflektirana

zraka koju generiramo u točki presjeka zrake kamere i poda sječe sferu u sceni. U točki presjeka reflektirane zrake i sfere ponovno generiramo zraku sjene i reflektiranu zraku. Nakon zaustavljanja postupka praćenja, rekurzivno se sjenča točka presjeka na sferi. Zraka sjene u toj točki presjeka vraća informaciju kako točka presjeka nije u sjeni. Nakon ove točke presjeka, sjenča se točka presjeka zrake kamere i poda. Zraka sjene koja je puštena prema izvoru svjetlosti iz ove točke vraća informaciju da je ova točka presjeka u sjeni trokuta koji se nalazi u sceni, te se sjenčanje odvija u skladu s ovom informacijom. Na samome kraju, boja slikovnog elementa za koji je generirana zraka kamere ekvivalentna je boji prvog presjeka. Kako bi navedeno razmatranje bilo potpuno, razmotrimo još generiranje zraka kamere i fenomenologiju refrakcije. Na slici 1.2.2 se nalazi prikaz procesa generiranja zrake kamere za jedan slikovni element na rasteru [1].



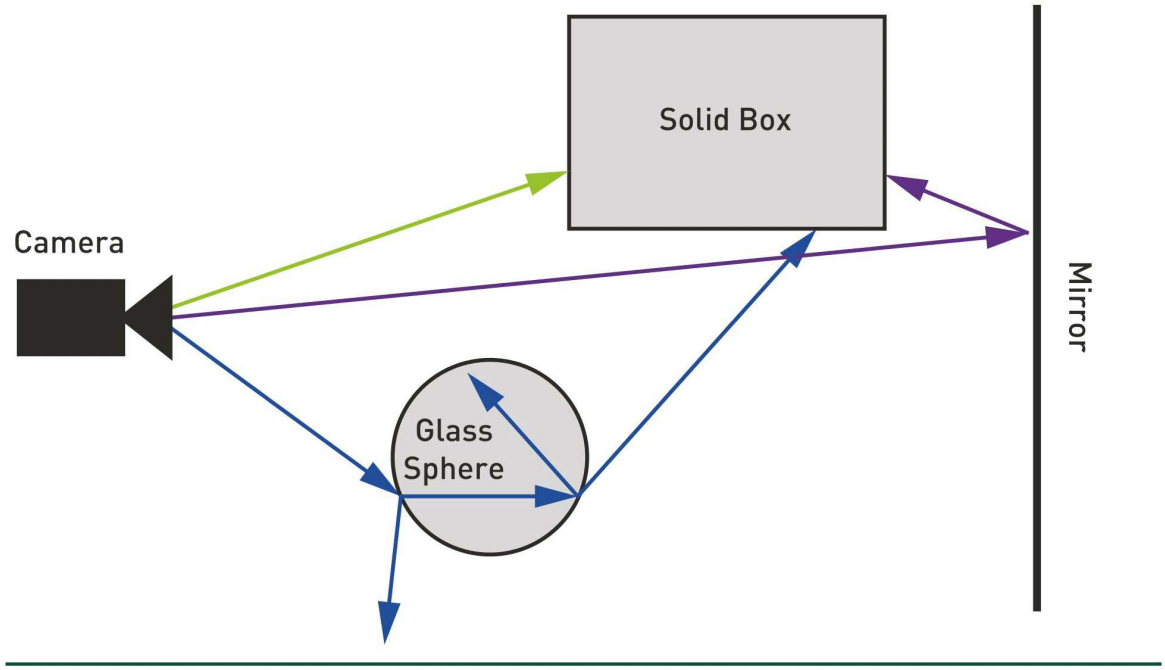
Slika 1.2.2: *Proces generiranja zrake kamere za zadani slikovni element na rasteru*

Prema jednadžbi (1) potrebno je odrediti početnu točku *o* i vektor smjera *d*. Početna točka *o* je trivijalno jednaka točki u kojoj se nalazi kamera. Vektor smjer *d* je moguće izračunati prema sljedećoj formuli [1]:

$$d(x,y) = \frac{w}{h} \tan\frac{\varphi}{2}\left(\frac{2(x+0.5)}{w} - 1\right)r - \tan\frac{\varphi}{2}\left(\frac{2(y+0.5)}{h} - 1\right)u + v \qquad (2)$$

Pretpostavljamo da su *x* i *y* cjelobrojne koordinate rastera kojemu x-os rate prema desno, dok y-os rate prema dolje. Rezolucija rastera je *w x h*, dok kut φ predstavlja

vertikalni kut pogleda (engl. *Field Of View, FOV*) kamere. Vektori *u*, *v* i *r* tvore lijevi koordinatni sustav pomoću kojeg se konstruira vektor smjera zrake *d*. Važno je napomenuti kako formula (2) ne daje normalizirani vektor, pa je stoga vektor d dobiven pomoću formule (2) potrebno normalizirati. Ranije je navedeno kako se u procesu sjenčanja mogu generirati nove zrake. Vrste novih zraka koje su se stvarale u procesu sjenčanja u primjeru sa slike 1.2.1 su bile zrake sjena i zrake refleksije. U procesu sjenčanja je također moguće generirati zrake refrakcije (engl. *Refraction Ray*). Zraka refrakcije služi za simuliranje fenomenologije refrakcije u materijalima poput stakla. Obratimo pozornost na plavu zraku na slici 1.2.3 [2]. Najbliže sjecište ove zrake je sjecište sa staklenom kuglom. U procesu sjenčanja ovog sjecišta stvaraju se dvije nove zrake, jedna zraka refleksije koja se stvara na ranije opisan način, i jedna zraka refrakcije, koja se generira na temelju Snellovog zakona loma. Također, prilikom sjenčanja točke sjecišta zrake refrakcije i staklene sfere, ponovno nastaju po jedna zraka refleksije i jedna zraka refrakcije.



Slika 1.2.3: *Primjer praćenja zrake u virtualnom okruženju s refraktivnim materijalom*

Važno je napomenuti kako se određivanje sjecišta zrake s nekim predmetom može modificirati na razne načine. Primjerice, možemo tražiti najbliži presjek, bilo kakav presjek, presjek samo s pozitivnim, samo negativnim ili bilo kojim vrijednostima parametra *t*, presjek na nekom ograničenom intervalu parametra $t \in [t_{min}, t_{max}]$ i

slično. Ograničavanje i modificiranje parametara pretrage izvodi se kako bi se pomoću algoritma praćenja zrake mogla simulirati razna fizikalna fenomenologija. Ovako opisan algoritam praćenja zrake podržava sljedeću fizikalnu fenomenologiju [2]:

- Tvrde sjene (engl. *Hard Shadows*)
- Zrcalne refleksije (engl. Mirror Reflections)
- Refrakcija (engl. *Refraction*)

Algoritam praćenja zrake u ovom obliku je opisao Turner Whitted. Zbog toga se ova inačica algoritma često naziva Whittedov algoritam praćenja zrake ili klasični algoritam praćenja zrake (engl. *Whitted Ray Tracing, Classical Ray Tracing*). U ovoj varijanti algoritma površine se tretiraju kao savršeno sjajne i glatke, te su izvori svijetlosti točkasti i nalaze se u beskonačnosti [2].

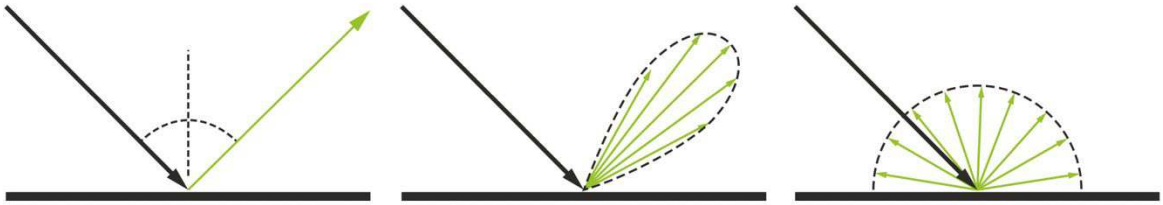
1.3. Algoritam stohastičkog praćenja zrake

Klasična varijanta algoritma praćenja zrake pati od nedostataka koji su direktna posljedica pretpostavki od kojih algoritam polazi. Pretpostavke o karakteristikama izvora svijetlosti, kao i one o svojstvima materijala ograničavaju algoritam na način da nedostaje sljedeća fenomenologija:

- Meke sjene (engl. *Soft Shadows*)
- Sjajne refleksije (engl. *Glossy Reflections*)
- Difuzne refleksije (engl. *Diffuse Reflections*)

Ono što sva navedena fenomenologija ima zajedničko je to da je za njihovu simulaciju potrebna neka vrsta stohastičkog procesa. Zbog toga je Robert L. Cook predložio varijaciju na klasični algoritam praćenja zrake koji se naziva algoritam stohastičkog praćenja zrake ili Cookov algoritam praćenja zrake. Razlika u odnosu na klasični algoritam praćenja zrake očituje se u fazi sjenčanja točke presjeka, odnosno u broju novih zraka koje se stvaraju i prate u točki presjeka. [2] Meke sjene kao fizikalna fenomenologija pojavljuju se zbog toga što izvori u stvarnom svijetu nisu točkasti, već imaju neku vlastitu površinu. Ako se prisjetimo razmatranja klasičnog algoritma praćenja zrake, u točki presjeka se za svaki izvor svijetlosti stvarala jedna zraka sjene i određivalo se postoji li presjek te zrake i nekog objekta

u sceni. [2] Kako bismo dobili meke sjene, umjesto samo jedne zrake po izvoru, stvara se i prati više zraka po izvoru svijetlosti te se promatraju i analiziraju njihova sjecišta s objektima u sceni, odnosno njihov izostanak. Svaka od zraka svijetlosti je usmjerena prema drugačijoj točki na izvoru svijetlosti u odnosu na ostale. Ideja je ekvivalentna za sjajne refleksije. Klasični algoritam praćenja zrake podržava samo savršene zrcalne refleksije zbog toga što se zraka refleksije generira na temelju zakona refleksije. Ovaj zakon je primarno primjeren za zrcala. [2] S druge strane, u stvarnosti postoje materijali koji reflektiraju svijetlost, no ne samo u smjeru zrcalne refleksije, nego s određenom distribucijom oko smjera zrcalne refleksije. Upravo zato umjesto generiranja samo jedne zrake refleksije možemo generirati više zraka u raznim smjerovima oko smjera zrcalne refleksije kako bismo primjereno simulirali sjajne refleksije. Isti princip se odnosi i na difuzne refleksije. [2] Materijali s izraženim difuznim svojstvima reflektiraju svijetlost u svim smjerovima na hemisferi oko normale. Kako bismo simulirali ovu fenomenologiju, u točki presjeka generiramo pripadne zrake i pratimo njihov put kroz virtualno okruženje. Više o svojstvima materijala će biti riječi u nastavku rada, no na slici 1.3.1 [2] vidljive su distribucije potrebnih zraka za pravilno simuliranje interakcije svijetlosti i zadanog materijala.



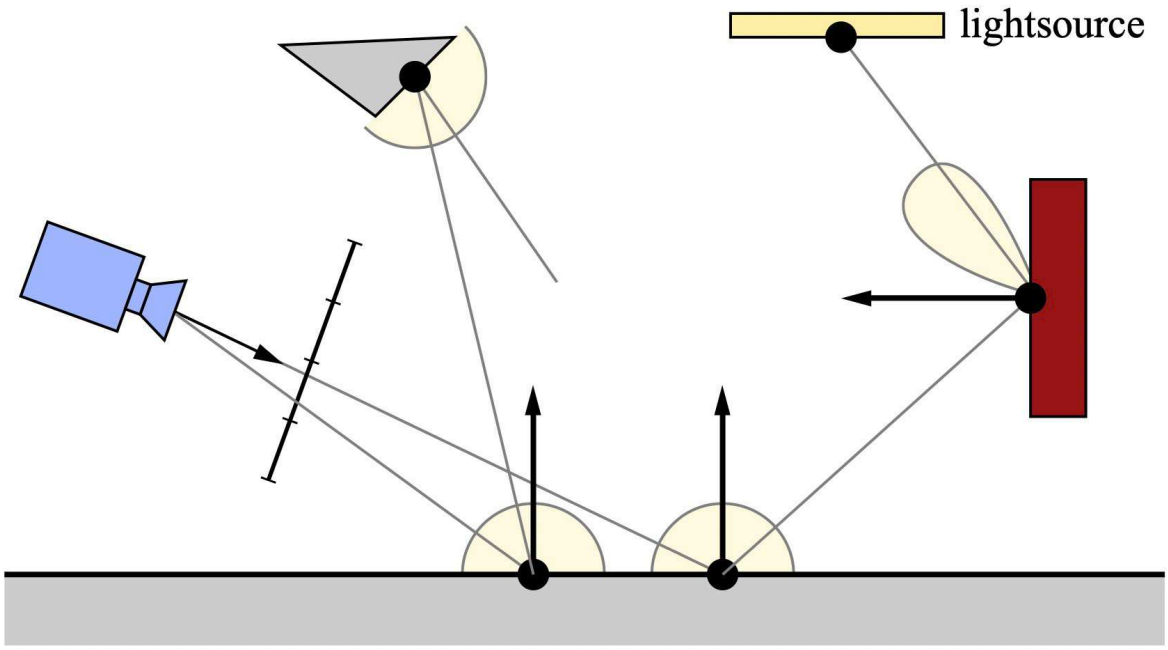
Slika 1.3.1: *Distribucije zraka potrebnih za pravilno simuliranje interakcije svijetlosti i (s lijeva na desno) zrcala, sjajnog materijala i difuznog materijala*

1.4. Algoritam praćenja puta

Algoritam praćenja puta prvi puta je predstavio James Kajiya kao potpuno rješenje problema globalnog osvjetljenja (engl. *Global Illumination*). Ovu varijantu algoritma praćenja zrake možemo promatrati kao poopćenje algoritma stohastičkog praćenja zrake. Ideja algoritma praćenja puta je sljedeća: za svaki slikovni element ne generiramo i pratimo samo jednu zraku, već veći broj zraka. Konačna boja slikovnog elementa je neka vrsta prosjeka svih rezultata koje su vratile zrake koje su generirane i praćene za taj isti slikovni element. No, ideja o većem broju uzoraka ne

staje samo na praćenju većeg broja zraka po slikovnom elementu, nego se proširuje i na zrake koje se generiraju u točkama sjecišta prilikom procesa sjenčanja. Razlika u generiranju zraka u točki sjecišta u odnosu na algoritam stohastičkog praćenja puta je u tome što ne generiramo više zraka za pojedinu vrstu zraka, nego odabiremo samo jednu. Štoviše, gubi se diferencijacija zraka ovisno o tome koju fizikalnu fenomenologiju želimo simulirati pomoću određene vrste zraka. [1] U algoritmu praćenja puta odabiremo samo jedan reflektirani smjer i generiramo novu zraku na temelju tog smjera. Koji ćemo smjer odabrati ovisi o svojstvima materijala u točki presjeka. Naime, ako se radi o ogledalu, generirat ćemo zraku na temelju zakona refleksije. S druge strane, ako se radi o materijalu s izraženim difuznim svojstvima, odabrat ćemo nasumičan smjer na hemisferi oko normale. Na ovaj način prirodno simuliramo svu fizikalnu fenomenologiju vezanu uz širenje svijetlosti u prostoru. [2] Algoritam praćenja puta daje rezultate utemeljene na fizikalnoj stvarnosti. No ove rezultate je moguće dobiti samo za jako veliki broj uzoraka i jako veliki broj odbijanja zrake. U idealnom slučaju zraku je potrebno pratiti sve dok se ne sudari s nekim od izvora svijetlosti u virtualnom okruženju. Ovo može biti problematično ako u virtualnom okruženju imamo dva predmeta koji su veoma blizu te postoji veliki broj međusobnih refleksija. James Kajiya je uz algoritam praćenja puta predstavio i jednadžbu iscrtavanja (engl. *Rendering Equation*) koja predstavlja elegantan matematički model za opisivanje širenja svijetlosti kroz prostor u svrhu iscrtavanja virtualnih scena. O jednadžbi iscrtavanja će biti više riječi u sljedećem poglavlju. Važno je napomenuti kako algoritam praćenja puta ne simulira fenomenologiju koja je vezana uz valnu prirodu svijetlosti poput difrakcije [2]. Također je važno primijetiti kako je algoritam praćenja puta čisti stohastički proces. Upravo zbog ovoga procesu je potreban veliki broj uzoraka za konvergenciju. Ako broj uzoraka nije dovoljno velik, u rezultatu je moguće primijetiti veće ili manje količine šuma. Količina šuma u rezultatima je obrnuto proporcionalna broju uzoraka. Na slici 1.4.1 nalazi se ilustracija algoritma praćenja puta. Sa slike 1.4.1 vidljivo je kako za isti slikovni element na rasteru u virtualnu scenu šaljemo i pratimo dvije zrake. Obje zrake se prvo sudaraju s objektom poda. Prema hemisferama oko normale, koje ilustriraju potencijalne smjerove reflektiranih zraka, može se zaključiti kako se radi o difuznim materijalima. Za lijevu točku presjeka odabiremo smjer reflektirane zrake koji će se sjeći s još jednim difuznim materijalom u sceni. Za desnu točku presjeka odabiremo smjer reflektirane zrake koji će sjeći sjajni materijal, što

možemo zaključiti prema obliku distribucije oko smjera savršene refleksije. Zraka odbijena od ovog materijala završava u izvoru svijetlosti čime završava praćenje njezinog puta. Konačna boja slikovnog elementa bila bi prosjek rezultata dviju ovako praćenih zraka.



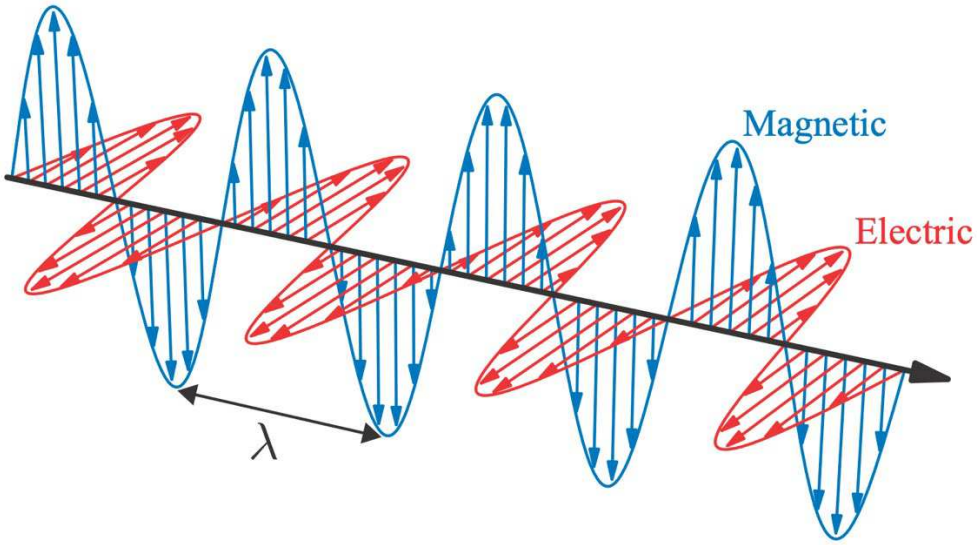
Slika 1.4.1: *Primjer izvođenja algoritma praćenja puta*

2. Fizikalno utemeljeno iscrtavanje

Fizikalno utemeljeno iscrtavanje (engl. *Physically based rendering*) je način iscrtavanja virtualnih okruženja koji nastoji precizno simulirati propagaciju svijetlosti u virtualnoj sceni i njezinu interakciju s materijalima različitih svojstava od kojih su sačinjeni objekti koji tvore virtualno okruženje. Fizikalno utemeljeno iscrtavanje nije jedini mogući način iscrtavanja virtualnih okruženja. Tako, na primjer, postoje različite nefotorealistične tehnike koje generiraju rezultate koji nisu nalik fizikalnoj stvarnosti. No, fotorealistične tehnike, poput fizikalno utemeljenog iscrtavanja, su od samih početaka razvoja računalne grafike u fokusu razvoja te je velik dio istraživačkog proces posvećen upravo njima. U nastavku će se razmotriti opis svijetlosti i povezane fenomenologije u fizikalnom kontekstu, te različiti modeli kojima se nastoji modelirati fizikalna fenomenologija propagacije svijetlosti i njezine interakcije s materijom u svrhu procesa iscrtavanja virtualnih okruženja.

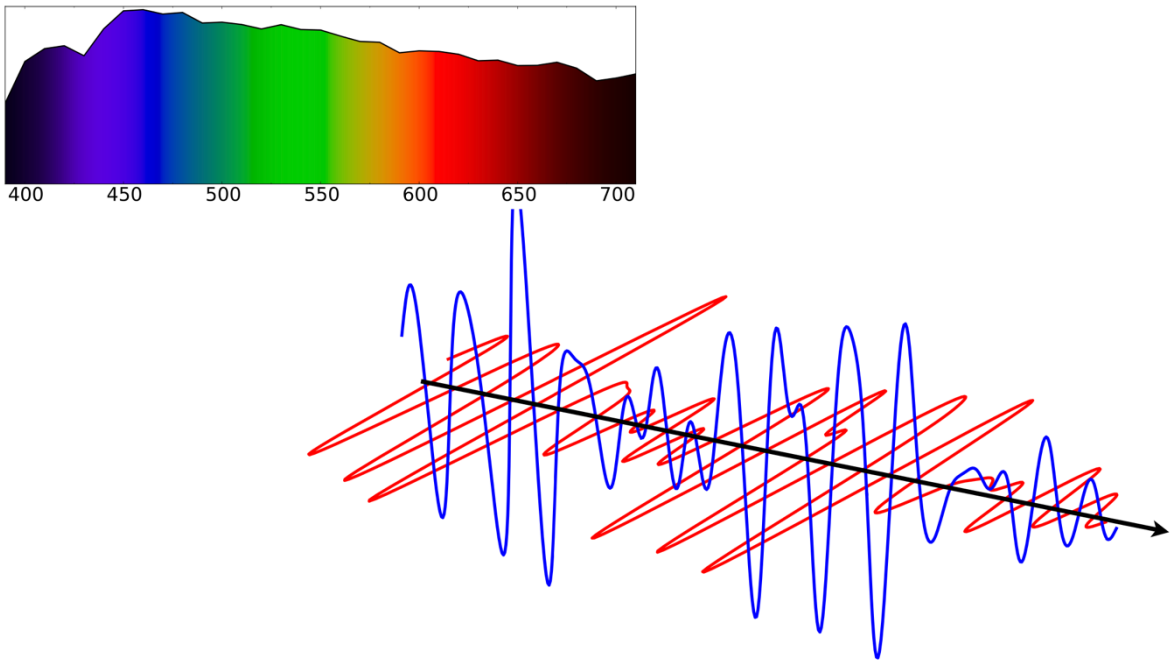
2.1. Propagacija svijetlosti

Fizikalno gledano, svijetlost je elektromagnetski transverzalni val [7]. Drugim riječima, svjetlost je val u kojemu, međusobno okomito, titraju električno i magnetsko polje, te je smjer propagacije vala također okomit na smjer titranja električnog i magnetskog polja. Na slici 2.1.1 nalazi se prikaz linearno polariziranog, monokromatskog vala [1]. Linearno polarizirana svijetlost je ona svijetlost u kojoj električno i magnetsko polje titraju u samo jednom smjeru. Svijetlost koju je moguće pronaći u prirodi, i kojom se bavi proces iscrtavanja u općenitom slučaju, generalno je nepolarizirana. Nepolarizirana svijetlost je svijetlost kojoj električno i magnetsko polje titraju u raznim smjerovima koji su svi okomiti na smjer propagacije svijetlosti.



Slika 2.1.1: *Prikaz linearno polariziranog, monokromatskog svjetlosnog vala s označenom valnom duljinom* λ

Na slici 2.1.1 grčkim slovom λ označena je valna duljina svjetlosnog vala. Ovo je jedno od glavnih obilježja bilo kojeg vala, te je u slučajnu svjetlosti direktno povezana s time je li val vidljiv ljudskom vizualnom sustavu [7]. Valne duljine elektromagnetskih valova su izrazito varijabilne. Na primjer, gama zrake imaju valne duljine manje od 0.01 nanometra, dok valovi ekstremno niske frekvencije (engl. *Extremely Low Frequency, ELF*) imaju valne duljine veće od 10000 kilometara [7]. Ljudski vizualni sustav može percipirati samo jedan, iznimno malen dio spektra elektromagnetskih valova. Točnije, ljudski vizualni sustav percipira samo elektromagnetske valove od otprilike 400-700 nanometara [7]. Ovaj dio spektra elektromagnetskih valova odnosi se na vidljivu svjetlost. Ako svjetlosni val sadrži samo jednu valnu duljinu, za taj val se kaže da je monokromatski. No, svjetlosni valovi koji su u centru problema iscrtavanja rijetko su monokromatski. Naprotiv, svjetlost koja se širi kroz prostor u fizikalnoj stvarnosti, pa tako i kroz virtualno okruženje, u općenitom slučaju sadrži veći broj valnih duljina. Slika 2.1.2 prikazuje svjetlosni val D56 koji predstavlja relativno standardan izvor bijele svijetlosti [7]. Na slici 2.1.2 je također moguće vidjeti i graf spektralne raspodijele snage (engl. *Spectral Power Distribution, SPD*). SPD graf prikazuje udjele pojedine valne duljine u svjetlosnom valu [7]. Za monokromatsku svijetlost SPD graf je delta funkcija na valnoj duljini sadržanoj u monokromatskom svjetlosnom valu.



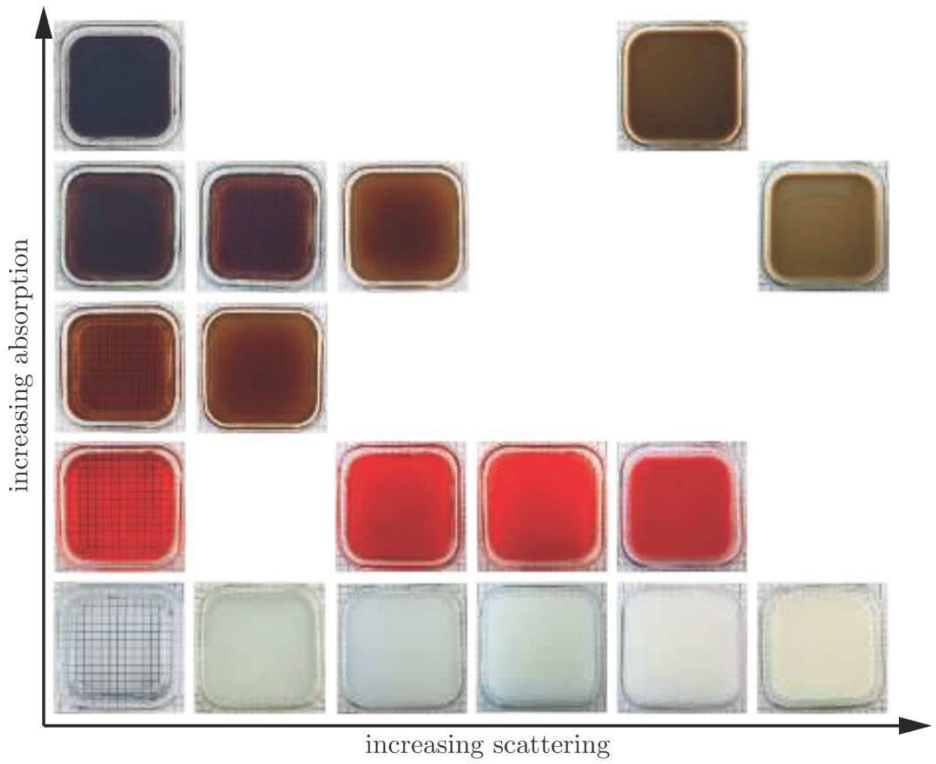
Slika 2.1.2: *SPD graf standardnog izvora bijele svijetlosti D56 (gore lijevo) i skica svjetlosnog vala istog izvora (dolje desno)*

Svjetlosni val se u vakuumu može širiti u beskonačnost jer će se električno i magnetsko polje međusobno podupirati [7]. Ova situacija nije pretjerano zanimljiva u kontekstu iscrtavanja virtualnih okruženja. Prilikom iscrtavanja veliki naglasak se stavlja na interakciju svijetlosti i materije. Materija je sastavljena od atoma. Prilikom interakcije svjetlosnog vala i atoma koji tvore materiju, atomi se polariziraju, odnosno upijaju dio energije svjetlosnog vala prilikom čega se razdvajaju pozitivni i negativni polovi atoma. Time nastaju dipoli [7]. Dio energije vala moguće je izgubiti kao toplinu, dok se energija koju su upili atomi ponovno odašilje u prostor u obliku novih svjetlosnih valova. U rijetkim plinovima moguće je interakciju svjetlosnog vala i atoma promatrati na razini pojedinog atoma, no u drugim medijima interakcije postaju izrazito kompleksne te ih je nemoguće modelirati na ovako elementaran način kao u rijetkim plinovima [7]. Stoga je potrebno osmisliti apstrakcije za navedene interakcije.

2.1.1. Valna optika

Valna optika je grana optike koja se bavi valnim svojstvima svjetlosti. Kao što je ranije navedeno, interakcija svjetlosnog vala i materije koja nije rijedak plin izrazito je kompleksna zbog velikog broja kombinacija dipola i interferencije svjetlosnih

valova [7]. Zbog toga područje valne optike uvodi apstrakcijski koncept homogenog medija. Homogeni medij je medij u kome svjetlosni val putuje po zadanom pravcu. Optička svojstva homogenog medija opisana su indeksom loma. Indeks loma sadrži u sebi dvije vrijednosti. Jedna vrijednost se odnosi na brzinu svjetlosti u mediju, dok se druga odnosi na količinu svjetlosti koju medij apsorbira [7]. Svojstvo medija da raspršuje svijetlost se u modelu homogenog medija modelira česticama. Ove čestice modeliraju lokalne, nehomogene dijelove medija u kojima dolazi do raspršenja svjetlosti. Izgled medija ovisi o njegovim svojstvima apsorpcije i raspršivanja svjetlosti. Slika 2.1.1.1 prikazuje izgled različitih medija u ovisnosti o svojstvima apsorpcije i raspršivanja [1]. Moguće je uočiti kako mediji koji imaju veliku sposobnost raspršivanja svjetlosti imaju svojstven zamućeni izgled. Primjer ovakvog medija je mlijeko. S druge strane, mediji koji nemaju veliku sposobnost raspršivanja svjetlosti imaju proziran izgled. Nadalje, mediji koji imaju izraženu sposobnost apsorpcije svjetlosti poprimaju boju onog dijela spektra kojeg ne apsorbiraju. S druge strane, mediji koji imaju slabu sposobnost apsorpcije imaju karakterističnu bijelu boju ili su bezbojni. U području iscrtavanja virtualnih okruženja ovakvi mediji se nazivaju sudjelujući mediji (engl. *participating media*).

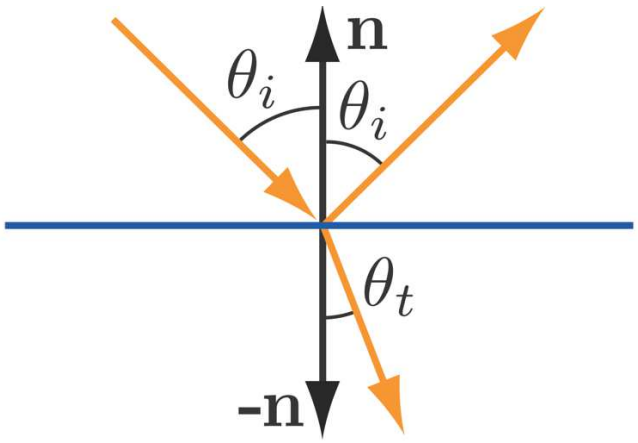


Slika 2.1.1.1: *Izgled različitih medija s različitim kombinacijama sposobnosti apsorpcije i raspršivanja svjetlosti*

U nastavku ovog poglavlja, kao i cijelog diplomskog rada, razmatrat će se samo površine objekata te fenomenologija koja se odvija upravo na površini materije. Jedna od najvažnijih fizikalnih pojava u valnoj optici je ogib odnosno difrakcija. Prilikom razmatranja ogiba na površini nekog objekta važan je pojam nanogeometrije (engl. *Nanogeometry*). Pojam nanogeometrije odnosi se na nepravilnosti koje su istog reda veličine kao i valna duljina svjetlosnog vala ili manje [7]. Upravo su ove nepravilnosti razlog pojave ogiba. Površine kojima sve nepravilnosti spadaju u kategoriju nanogeometrije smatraju se optički glatkima. Pojava ogiba se gotovo nikada ne uzima u obzir prilikom iscrtavanja virtualnih okruženja, te se većina valne optike zanemaruje prilikom modeliranja propagacije svjetlosti u svrhu iscrtavanja virtualnih okruženja.

2.1.2. Geometrijska optika

Geometrijska optika je grana optike koja se bavi pojednostavljenim modelom propagacije svijetlosti i njezine interakcije s materijom. Modeli geometrijske optike su modeli koji se koriste u računalnoj grafici u kontekstu iscrtavanja virtualnih okruženja [7]. Jedno od glavnih pojednostavljenja geometrijske optike je tretiranje optički glatkih površina kao savršeno ravnima. Time se zanemaruju pojave poput ogiba [7]. Moguće je pokazati kako prilikom interakcije zrake svjetlosti i ovakve savršeno ravne površine u općem slučaju nastaju dvije nove zrake: reflektirana zraka i refraktirana zraka.

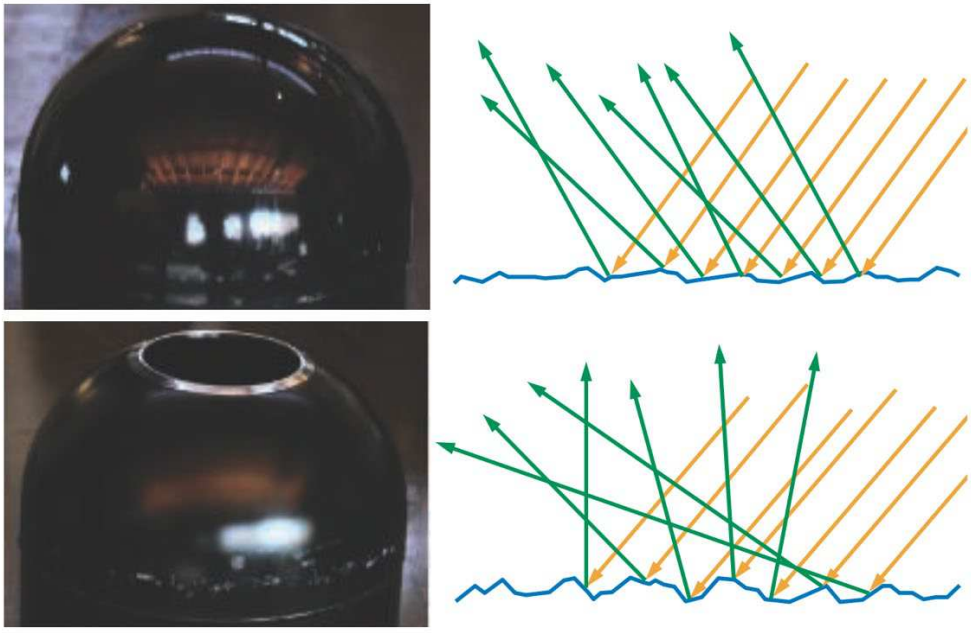


Slika 2.1.2.1: *Prikaz reflektirane i lomljene zrake koje nastaju interakcijom upadne zrake svjetlosti i optički glatke površine*

Reflektiranu zraku moguće je konstruirati pomoću zakona refleksije koji govori kako je kut između normalne na površinu i upadne zrake svjetlosti jednak kutu između normale i reflektirane zrake svjetlosti. Zakon refleksije lako je uočiti na slici 2.1.2.1 [1]. Refraktirana zraka se još često zove i lomljena zraka. Refraktiranu zraku moguće je konstruirati Snellovim zakonom loma koji glasi:

$$\sin\theta_t = \frac{n_1}{n_2}\sin\theta_i \tag{3}$$

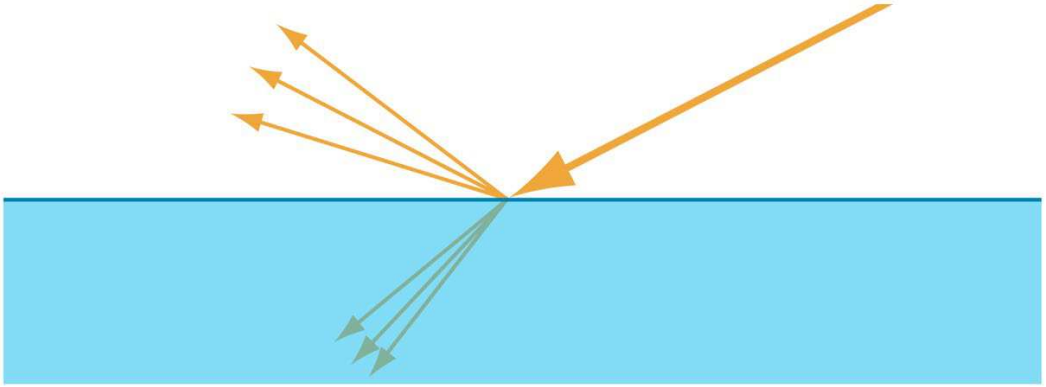
θ_i označava kut između upadne zrake svjetlosti i normalne na površinu, θ_t kut između lomljene zrake i negativne normalne na površinu, n_1 je indeks loma medija u kojem se nalazi upadna zraka, dok je n_2 indeks loma medija u kojem se nalazi refraktirana zraka [1]. Snellov zakon loma je također moguće uočiti na slici 2.1.2.1. No, površine u fizikalnoj stvarnosti su rijetko optički glatke. Naprotiv, većina površina ima određen stupanj nepravilnosti. Ove nepravilnosti, koje su veće od valne duljine svjetlosnog vala, ali manje od slikovnog elementa, spadaju u kategoriju koja se mikrogeometrija (engl. *Microgeometry*) [7]. Slika 2.1.2.2 prikazuje primjere dviju površina s nepravilnostima koje spadaju u kategoriju mikrogeometrije [1].



Slika 2.1.2.2: *Površina s manje nepravilnosti iz kategorije mikrogeometrije s refleksijom sličnoj zrcalnoj refleksiji (gore) i površina s većom količinom nepravilnosti iz kategorije mikrogeometrije s mutnom refleksijom (dolje)*

Sa slike 2.1.2.2 također je moguće uočiti na koji način ove nepravilnosti utječu na izgled površine. Naime, površine s malim nepravilnostima ove kategorije izgledaju

blisko zrcalima, zbog toga što se većina zraka reflektira u sličnim smjerovima, te je time disperzija smjerova reflektiranih zraka mala, dok površine s više nepravilnosti iz ove kategorije imaju mutnije refleksije zbog toga što je disperzija smjerova reflektiranih zraka velika. Makroskopski gledano, moguće je utjecaj mikrogeometrije na izgled površine modelirati na način prikazan na slici 2.1.2.3, odnosno na način da površina reflektira i lomi zrake svijetlosti u više smjerova [1].



Slika 2.1.2.3: *Makroskopski pogled na interakciju svjetlosne zrake s površinom koja ima nepravilnosti iz kategorije mikrogeometrije*

Detalji interakcije zrake svjetlosti i objekta sačinjenog od zadanog materijala ovise o samoj vrsti materijala.

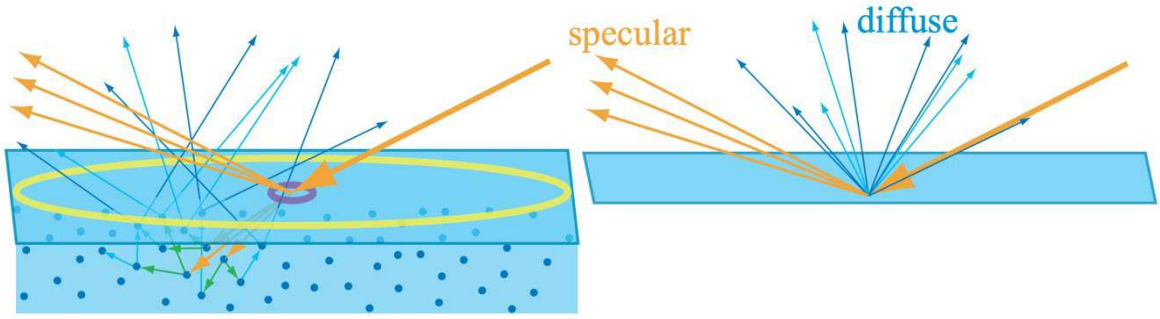
2.1.3. Vrste materijala

Kao što je ranije navedeno, svjetlost je elektromagnetski val. Stoga interakcija svjetlosti s materijom ovisi o električnim svojstvima materije [7]. S obzirom na električna svojstva, materijale je moguće podijeliti u tri skupine [7]:

- Vodiči (engl. *Conductors*) – metali
- Dielektrici (engl. *Dielectrics*) - izolatori
- Poluvodiči (engl. *Semiconductors*)

Vodiči, odnosno metali, su vrsta materijala koja upija sve lomljene zrake. Stoga ovu vrstu materijala primarno karakteriziraju refleksije na površini materijala [7]. Površinske refleksije se uobičajeno apstrahiraju kao spekularna komponenta osvjetljenja. Dielektrici, poput plastike, imaju drukčiju interakciju od vodiča. Na slici 2.1.3.1 moguće je razlučiti interakciju zrake svjetlosti s dielektričnim materijalom [1]. Dio zraka se reflektira, dok se dio zraka lomi. Dio lomljenih zraka se djelomično

upija, dok se dio zraka raspršuje. Dio raspršenih zraka se ponovno odašilje u prostor. Ova pojava se naziva raspršenje pod površinom (engl. *Subsurface Scattering*). Na slici 2.1.3.1 ovaj dio zraka prikazan je plavom bojom. S lijeve slike na slici 2.1.3.1 vidljivo je da plave zrake nastaju većim brojem raspršenja. S iste slike je također vidljivo da su točke u kojima raspršene zrake ponovno izlaze iz materijala različito udaljenje od ulazne točke. Ako su sve izlazne točke unutar područja interesa jednog slikovnog elementa, moguće je uvesti pojednostavljenje u kojemu sve raspršene zrake koje se ponovno šalju u prostor izlaze iz jedne točke [7]. Ovaj slučaj prikazan je na lijevoj slici na slici 2.1.3.1 pomoću žutog kruga. Ovakvo pojednostavljenje se naziva difuzna komponenta osvjetljenja i prikazana je na desnoj slici na slici 2.1.3.1. Ako su pak izlazne točke van područja interesa jednog slikovnog elementa, kao što je prikazano ljubičastim krugom na lijevoj slici slike 2.1.3.1, nije moguće upotrijebiti ovo pojednostavljenje te je potrebno primijeniti drugačije apstrakcijske modele. Obje navedene situacije moguće je reproducirati s istim materijalom, mijenjajući udaljenost objekta od kamere. Ljudska koža je odličan primjer za ovu fenomenologiju. Ako se ljudsku kožu promatra s veće udaljenosti, moguće je modelirati raspršenje pod površinom pojednostavljenim modelom difuzne komponente osvjetljenja. S druge strane, ako kožu promatramo s male udaljenosti, pojednostavljeni model difuzne komponente osvjetljenja nije dovoljno precizan, te je potrebno upotrijebiti prikladniju metodu simulacije raspršenja pod površinom.



Slika 2.1.3.1: *Interakcija zrake svjetlosti i dielektričnog materijala (lijevo) i pojednostavljeni model interakcije (desno)*

Propagacija svijetlosti i njezina interakcija s materijom je do sada opisana u fizikalnom kontekstu. Kako bi se navedeni modeli mogli koristiti prilikom iscrtavanja

virtualnih okruženja, potrebno ih je ukomponirati u prikladan matematički model. Ovakav matematički model predstavio je James Kajiya.

2.2. Jednadžba iscrtavanja

Jednadžbu iscrtavanja predstavio je James Kajiya 1986. godine. Jednadžba iscrtavanja elegantno sažima sve moguće puteve svijetlosti u virtualnom okruženju, kao i sve njezine moguće interakcije s materijom. U nastavku je dan oblik iz knjige *Real-time Rendering Fourth Edition* [1]:

$$L_o(p,v) = L_e(p,v) + \int f(l,v)L_i(p,l)(n \cdot l)^+ dl \quad (4)$$

L_o označava sjajnost koja izlazi iz točke *p* u smjeru *v*. L_e označava sjajnost koju emitira površina iz točke *p* u smjeru *v*. L_i označava ulaznu sjajnost u točku *p* iz smjera *l*. Integral prolazi kroz sve moguće smjerove vektora *l* koji se nalaze na hemisferi oko normale. Faktor L_i moguće je zamijeniti na sljedeći način:

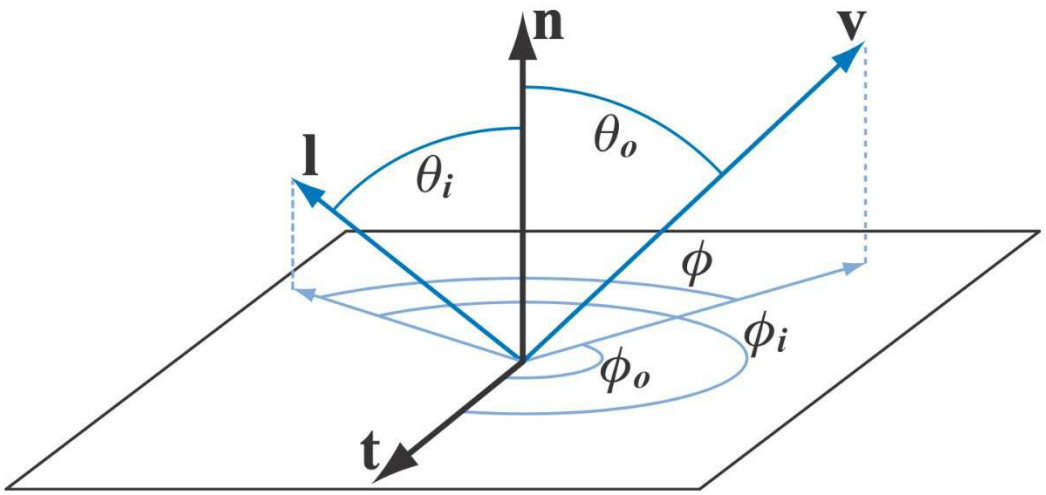
$$L_i(p,l) = L_o(r(p,l), -l) \quad (5)$$

Ova zamjena govori kako je ulazna sjajnost u točku *p* iz smjera *l* jednaka izlaznoj sjajnosti iz neke druge točke u smjeru *-l*. Ova druga točka određena je funkcijom bacanja zrake *r*. Ova funkcija vraća prvi presjek zrake kojoj je početna točka u točki *p* i smjer joj je jednak vektoru *l*. Ispod integrala također se nalazi i skalarni produkt vektora normale na površinu *n* i ulaznog vektora smjera *l*. Notacija + označava da se negativne vrijednosti postavljaju na nulu. Zadnji faktor pod integralom odnosi se na funkciju *f* koja opisuje svojstva materijala od kojeg je sačinjen objekt kojem pripada točka *p*, odnosno funkciju koja modelira interakciju zrake svjetlosti i materijala.

2.2.1. BRDF funkcije

Dvosmjerna distribucijska funkcija reflektivnosti (engl. *Bidirectional Reflectance Distribution Function, BRDF*) je funkcija koja modelira fenomenologiju lokalizirane reflektivnosti. Pojam lokalizirane reflektivnosti odnosi se na površinske refleksije, kao i na lokalizirano raspršenje pod površinom materijala [1]. BRDF funkcije ovise o vektoru smjeru dolazne svjetlosti, koji se označava s *l*, kao i o izlaznom vektoru smjeru pogleda, koji se označava s *v*. Svaki od ova dva vektora smjera ima dva

stupnja slobode [1]. Na slici 2.2.1.1 prikazana je česta parametrizacija navedenih vektora smjera [1].



Slika 2.2.1.1: *Parametrizacija ulaznog vektora svjetlosti l i izlaznog vektora pogleda v kod BRDF funkcije*

Ova parametrizacija uključuje kutove elevacije označene simbolom θ, kao i kutove azimuta, odnosno horizontalne rotacije označene simbolom ϕ. Kutovi elevacije θ određuju se u odnosu na normalu na površinu *n*, dok se kutovi azimuta određuju horizontalnu rotaciju oko normale na površinu *n*. BRDF funkcije iz posebne kategorije BRDF funkcija u kojem relativan odnos između kutova azimuta ostaje isti nazivaju se izotropnim BRDF funkcijama [1]. Važno ja navesti ograničenja BRDF funkcija od kojih su neka sljedeća [1] [7]:

- Ne mogu se koristiti za prozirne materijale
- Ne uzimaju u obzir globalno raspršenje ispod površine materijala
- Ne uzimaju u obzir pojave poput flourescencije (luminiscencije) i fosforescencije
- Definirana samo za vektore *l* i *v* iznad makroskopske površine

Usprkos ograničenjima, BRDF funkcije se izrazito često koriste za modeliranje interakcije svjetlosti i materije u procesu fizikalno utemeljenog iscrtavanja virtualnih okruženja. Na samom početku njihovog razvoja BRDF funkcije su pretpostavljale uniformnost površina. No, ovakav slučaj je izrazito rijedak u fizikalnoj stvarnosti, pa tako i u virtualnim okruženjima. Čak ako je i objekt sačinjen od jednog materijala, površina može sadržavati nepravilnosti koje mijenjaju svojstva interakcije objekta i

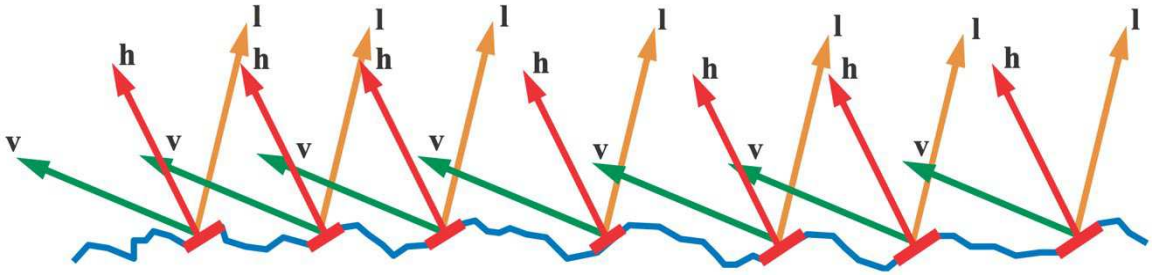
svjetlosti u ovisnosti o poziciji na površini objekta. BRDF funkcije koje uzimaju u obzir ove nepravilnosti nazivaju se prostorno varijabilne BRDF funkcije (engl. *spatially varying BRDF, SVBRDF*) ili jednostavnije prostorne BRDF funkcije (engl. *spatial BRDF, SBRDF*). No, u današnje vrijeme se implicitno podrazumijeva kako svaka BRDF funkcija uzima u obzir ovaj slučaj [1]. Također je važno napomenuti kako u teoriji svaka BRDF funkcija mora zadovoljavati Helmholtzov princip recipročnosti, odnosno da vrijednost funkcije ostaje ista ako se zamjene ulazni i izlazni kut. BRDF funkcije koje se koriste prilikom iscrtavanja virtualnih okruženja često krše ovaj princip bez vidljivih posljedica [1]. Najjednostavnija BRDF funkcija koja se učestalo koristi prilikom iscrtavanja virtualnih okruženja je Lambertova BRDF funkcija. Ova BRDF funkcija kvantificira fenomenologiju lokaliziranog raspršenja svjetlosti pod površinom. Lambertova BRDF funkcija dana je sljedećim izrazom [1]:

$$f(l, v) = \frac{\rho_{ss}}{\pi} \tag{6}$$

Veličina ρ_{ss} označava refleksnu boju materijala (engl. *albedo*). Točnije, ovako označena refleksna boja naglašava da se ova veličina odnosi na boju vezanu uz sposobnost materijala da raspršuje svjetlost ispod površine. U literaturi je moguće pronaći kako se ova veličina označava s c_{diff}, odnosno da se radi od boji difuznog odgovora površina na osvjetljenje [1]. Lambertova BRDF funkcija je konstantna za sve kombinacije vektora *l* i *v*. Lambertova BRDF funkcija jednostavno i elegantno opisuje difuznu komponentu osvjetljenja prikazanu na desnoj dijelu slike 2.1.3.1. Dio jednostavnosti Lambertove BRDF funkcije proizlazi iz opservacije kako su smjerovi zraka koje se ponovno emitiraju kao posljedica raspršenja ispod površine materijala nasumični i jednoliko raspršeni oko normale na površinu u ulaznoj točki. S druge strane, kako bi se opisala spekularna komponenta osvjetljenja prikazana na desnom dijelu slike 2.1.3.1. potrebna je kompleksnija BRDF funkcija. Zrake reflektirane na površini imaju smjerove koji su jednaki smjeru zrcalne zrake ako se radi o optički glatkoj površini ili bliski smjeru zrcalne refleksije ako površina nije optički glatka. Primjer BRDF funkcije koja opisuje površinske refleksije koje uzimaju u obzir nepravilnosti u kategoriji mikrogeometrije dan je sljedećim izrazom [7]:

$$f(l, v) = \frac{F(l,h)G(l,v,h)D(h)}{4(n \cdot l)(n \cdot v)} \tag{7}$$

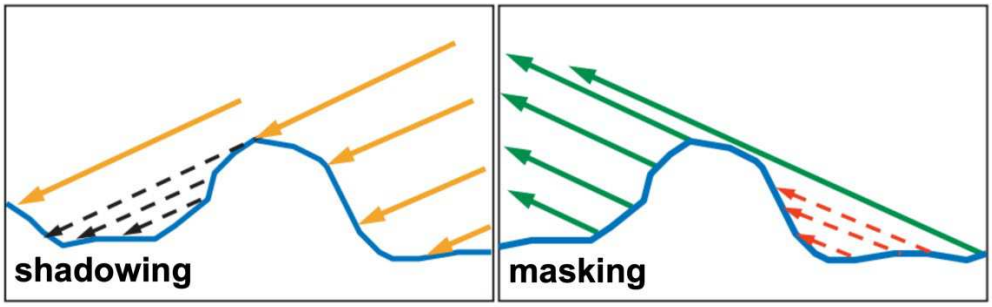
Ovakva BRDF funkcija izvodi se na temelju teorije malih djelića površine (engl. *Microfacets Theory*). Ova teorija promatra malene djeliće površine i lokalno ih tretira kao savršena zrcala [7]. Svaki od ovih djelića površine posjeduje vlastitu normalu koja se uobičajeno označava s *m* [7]. Stoga ovi djelići površine reflektiraju uzlanu svijetlost prema zakonu refleksije u odnosu na vlastitu lokalnu normalu *m*. Podsjetimo se kako BRDF funkcija kao parametre uzima ulazni smjer svjetlosti predstavljen vektorom *l* i izlazni smjer pogleda označen vektorom *v*. Pošto svaki djelić površine ima vlastitu normalu *m*, samo će određeni postotak od ukupnog broja djelića površine reflektirati ulaznu zraku svjetlosti *l* u smjer izlaznog vektora pogleda *v*. Normala koja je potrebna kako bi se ovakva refleksija mogla dogoditi naziva se poluvektor između vektora *l* i *v* i uobičajeno se označava se slovom *h*. Na slici 2.2.1.2 vidljiva je površina koja nije optički glatka te na njoj označene vektore *l*, *v* i *h* za određene djeliće površine [1].



Slika 2.2.1.2: *Površina objekta s optičkim nepravilnostima i označenim vektorima l, v i h za određene djeliće površine*

Postotak djelića površine kojima je normala *m* jednaka poluvektoru *h* kvantificiran je funkcijom *D(h)* u izrazu (7). Ova funkcija je funkcija normalne distribucije (engl. *Normal Distribution Function, NDF*). NDF funkcija daje gustoću djelića površine s normalom jednakom poluvektoru *h* u odnosu na cjelokupnu površinu koja se razmatra [7]. No, neće svi djelići površine kojima je normala jednaka poluvektoru *h* pridonijeti konačnoj refleksiji. Na slici 2.2.1.3 prikazani su primjeri u kojima dolazi do pojave zasjenjivanja (engl. *Shadowing*) gledano u odnosu na upadni smjer svjetlosti ili do pojave maskiranja (engl. *Masking*) gledano u odnosu na izlazni smjer pogleda. Obje pojave onemogućuju određenom postotku djelića površina da doprinosi konačnom rezultatu površinske refleksije [7]. Ovu pojavu kvantificira geometrijska funkcija (engl. *Geometry function*) označena slovom *G* u izrazu (7). Zadnja funkcija koja je preostala u izrazu (7) označena je slovom *F*. Ova funkcija kvantificira pojavu

Fresnelove reflektivnosti. Fresnelova reflektivnost određuje koliki će se postotak ulazne svjetlosti koja pogađa relevantne djeliće površine uopće reflektirati [7]. Nulta vrijednost ove funkcije, odnosno vrijednost za 0°, često označena s F_0, definira spekularna svojstva materijala. [7] Metalima je ova vrijednost uobičajeno različita za svaki od RGB kanala i varira u intervalu od 0.45 do 1 gledajući linearnu skalu. Dielektrični materijali imaju jednaku vrijednost za sve RGB kanale i ta vrijednost se generalno nalazi u intervalu od 0 do 0.2. Područje od 0.2 do 0.45 odnosi se na poluvodiče [7]. Ukupna vrijednost funkcije, koja ovisi o kutu između vektora smjera upadne svjetlosti *l* i poluvektora *h*, može se izraziti pomoću nulte vrijednosti F_0 [7].



Slika 2.2.1.3: *Pojava zasjenjenja djelića površine u odnosu na upadni smjer svjetlosti (lijevo) i pojava maskiranja djelića površine u odnosu na izlazni smjer pogleda (desno)*

Jednadžba iscrtavanja nema analitičko rješenje. Štoviše, integral koji je potrebno izračunati za pronalazak rješenja zadan je rekurzivno. Zbog toga je potrebno odabrati prikladnu metodu za određivanje barem aproksimativnog rješenja jednadžbe iscrtavanja. Ovakav integral pogodan je za rješavanje numeričkim metodama integracije poput *Monte-Carlo* metode integracije.

2.3. Monte-Carlo metoda integracije

Monte-Carlo metoda integracije je numerička metoda integracije koja omogućuje izračunavanje procjene vrijednosti bilo kojeg integrala uzimajući nasumične vrijednosti funkcije pod integralom. Općeniti oblik *Monte-Carlo* metode integracije dan je izrazom (8) [8]:

$$F_N = \frac{1}{N}\sum_{n=0}^{N}\frac{f(x_n)}{p(x_n)} \tag{8}$$

F_N je procijenjena vrijednost integrala funkcije $f(x_n)$. Procijenjena vrijednost integrala dobiva se kao prosjek nasumičnih uzoraka. Vrijednost funkcije za zadani uzorak x_n dijeli se s vjerojatnošću pojavljivanja navedenog uzorka. Time se osigurava da česti uzorci manje doprinose konačnoj procjeni, dok rijetki uzorci doprinose više. Ako se *Monte-Carlo* metoda integracije primjeni na rješavanje jednadžbe iscrtavanja, jednadžbu iscrtavanja moguće je zapisati u sljedećem obliku:

$$L_o(p,v) = L_e(p,v) + \frac{1}{N}\sum_{n=0}^{N} 2\pi f(l,v) L_i(p,l)(n \cdot l)^+ \quad (9)$$

Ovakav zapis pretpostavlja uniformno uzorkovanje smjerova s hemisfere oko normale na površinu u točki *p*. Vjerojatnost pojavljivanja uzorka koji je dobiven na ovaj način iznosi $1/2\pi$ [8]. *Monte-Carlo* metoda ima garantiranu konvergenciju [8]. Procjena vrijednosti integrala postaje sve bolja kako raste broj uzoraka. No, brzina konvergencije je izrazito bitna prilikom korištenja *Monte-Carlo* metode integracije u interaktivnim sustavima. Kako bi se konvergencija ubrzala, uvodi se koncept uzorkovanja po važnosti (engl. *Importance Sampling*), odnosno pristranog uzorkovanja. Ideja uzorkovanja po važnosti odnosi se na pronalazak distribucije uzoraka za koju vrijedi da je jednaka vrijednosti funkcije pod integralom za zadani uzorak [8]. Prilikom ovakvog uzorkovanja uzorci s malenim doprinosom će biti odabrani s proporcionalno malom vjerojatnošću, odnosno rjeđe, dok će se uzorci s velikim doprinosom biti odabrani češće. Ovakve idealne distribucije je teško pronaći za jednadžbu iscrtavanja zbog velike varijabilnosti ulazne svijetlosti. S druge strane, odabirom distribucije koja prati BRDF funkciju ili umnožak BRDF funkcije i skalarnog umnoška pod integralom znatno se ubrzava konvergencija *Monte-Carlo* metode. Na ovome tragu moguće je uniformno uzorkovanje hemisfere zamijeniti kosinusnim uzorkovanjem (engl. *Cosine sampling*). Vjerojatnost odabira smjera *l* prilikom kosinusnog uzorkovanja dana je sljedećim izrazom [8]:

$$p(l) = \frac{\alpha+1}{2\pi}(n \cdot l)^{\alpha} \quad (10)$$

Za vrijednost parametra α=0 kosinusno uzorkovanje degradira u uniformno. Prikladne vrijednosti parametra α ovise o svojstvima materijala. Ako je BRDF funkcija opisuje difuznu komponentu osvjetljenja, prikladna je vrijednost α=1. Za spekularne BRDF funkcije vrijednosti parametra α ovise o količini nepravilnosti na površini materijala.

Za kraj ovog poglavlja je izrazito važno naglasiti kako porodica algoritama praćenja zrake nudi izrazito elegantno rješenje za fizikalno utemeljeno iscrtavanje virtualnih okruženja. Jednadžba iscrtavanja se prirodno može riješiti pomoću porodice algoritama praćenja zrake zbog toga što je i sama jednadžba opisana sličnim konceptima kao i sami algoritmi. Algoritam praćenja puta predstavlja primjenu *Monte-Carlo* metode integracije na rješavanje jednadžbe iscrtavanja u stvarnim sustavima. Sve u svemu, porodica algoritama praćenja zrake postavlja se kao objedinjeno rješenje jednadžbe iscrtavanja.

3. Fizikalna fenomenologija u kontekstu algoritama za iscrtavanje virtualnih okruženja

Kao što je detaljno objašnjeno u sklopu drugog poglavlja, cilj fizikalno utemeljenog iscrtavanja je generiranje prikaza virtualnih okruženja koji nalikuju fizikalnoj stvarnosti. Generiranje ovakvih prikaza postiže se modeliranjem fizikalnog konteksta propagacije svijetlosti i njezine interakcije s materijom matematičkim modelima koje je zatim moguće riješiti pomoću algoritama na računalu. Jedan takav matematički model je i jednadžba iscrtavanja koja nudi općeniti oblik rješenja određivanja boje slikovnog elementa. Porodica algoritama praćenja zrake prirodno je algoritamsko rješenje ovog matematičkog modela. No, kao što je već nekoliko puta izneseno u sklopu ovog diplomskog rada, algoritam rasterizacije i njemu pripadne metode za simulaciju fizikalne fenomenologije dominantna su skupina metoda i algoritama za iscrtavanje virtualnih okruženja u interaktivnim sustavima. Algoritam rasterizacije izrazito je brz i učinkovit algoritam za određivanje vidljivosti. No, algoritam kao takav ne nudi unificirano rješenje jednadžbe iscrtavanja. Zbog toga se javila potreba razdjeljivanja problema iscrtavanja na manje probleme od kojih se svaki odnosi na simulaciju određene fizikalne fenomenologije. Shodno tomu, za svaku vrstu fizikalne fenomenologije, potrebno je razviti prikladne rasterizacijske metode koje simuliraju željenu fenomenologiju unutar ograničenja koja postavlja algoritam rasterizacije. Jedno od najvećih ograničenja koje primarni algoritam rasterizacije postavlja na svoje metode je ograničavanje skupa podataka dostupnih procesu sjenčanja na podatke koji se trenutno nalaze unutar vidnog polja kamere. Kao što će biti vidljivo u nastavku, ovo ograničenje izrazito je ozbiljno te daleko umanjuje sposobnost razvijenih metoda da kvalitetno i vjerno simuliraju različitu fizikalnu fenomenologiju. Stoga se veliki dio istraživačkog procesa veznog uz algoritam rasterizacije odnosi na pokušaj proširenja skupa podataka dostupnih za sjenčanje [1]. Kako bi se proširio skup dostupnih podataka za sjenčanje, sustavi za iscrtavanje temeljeni na algoritmu rasterizacije nastoje dio operacija potrebnih za pravilnu simulaciju fizikalne fenomenologije izračunati prije vremena, te ih pospremiti u prikladnom formatu kojeg je moguće koristiti kasnije. No ni ovakav pristup nije bez svojih ograničenja. Najveće ograničenje ovakvog pristupa je veći ili manji stupanj inertnosti na dinamičke promjene u virtualnom okruženju. Ovo

zapažanje dovodi do zaključka kako prilikom iscrtavanja virtualnih okruženja cjevovodom baziranom na algoritmu rasterizacije veća kvaliteta prikaza dolazi s određenim stupnjem inercije na dinamičke pojave u virtualnom okruženju. Ovaj odnos inercije i kvalitete simulacije određene fizikalne fenomenologije posebno je izražen kod određenih fizikalnih pojava. Ono što je zajedničko svim ovim pojavama je što je za njihovu simulaciju potrebna neka vrsta stohastičkog procesa. Ovo se odnosi na fizikalnu fenomenologiju poput:

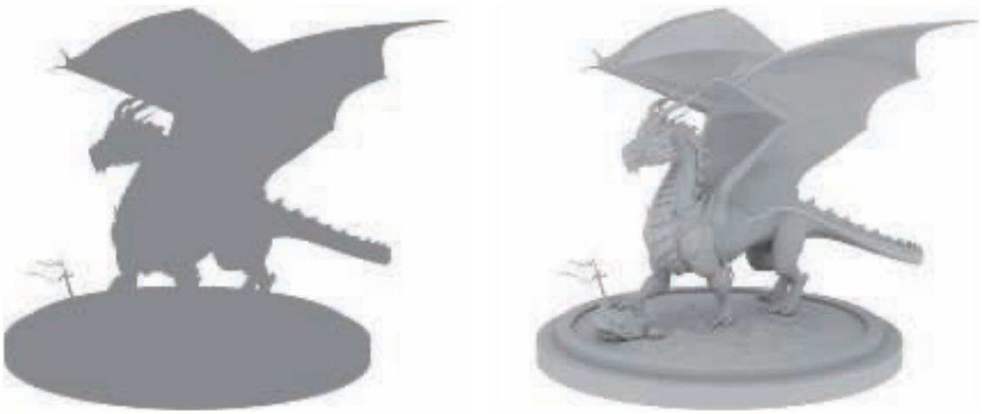
- Ambijentalnog zasjenjenja (engl. *Ambient Occlusion*)
- Mekih sjena (engl. *Soft Shadows*)
- Zrcalnih i sjajnih refleksija (engl. *Mirror and Glossy Reflections*)
- Refrakcija (engl. *Refraction*)
- (Indirektnog) globalnog osvjetljenja (engl. *(Indirect) Global Illumination*)
- Globalnog raspršenja pod površinom (engl. *Global Subsurface Scattering*)

Algoritam rasterizacije za svaku od navedene fenomenologije ima razvijene pripadne metode koje aproksimativno simuliraju pripadnu fenomenologiju. U nastavku će se razmotriti koje su u današnje vrijeme standardne rasterizacijske metode za simulaciju pojedine fizikalne fenomenologije, te koja su ograničenja i nedostaci tih istih metoda.

3.1. Ambijentalno zasjenjenje

Ambijentalno zasjenjenje (engl. *Ambient Occlusion*) jedan je od najjednostavnijih učinaka globalnog osvjetljenja [1]. Ovim učinkom se simulira pojava mekih sjena na mjestima poput pukotina ili između dijelova zadanog objekta koji su međusobno blizu [10]. Najjednostavniji primjer mjesta na kojem bi se generirala meka sjena kao rezultat ovog učinka je na spojnici poda i zida. Pomoću ovog učinka je također moguće naglasiti oblik objekta kada osvjetljenu u sceni nedostaje usmjerenosti [1]. Slika 3.1.1 prikazuje objekt zmaja iscrtanog samo pomoću ambijentalne komponente osvjetljenja na lijevom dijelu slike te isti objekt zmaja iscrtan pomoću ambijentalne komponente osvjetljenja i učinka ambijentalnog zasjenjenja na desnom dijelu slike [1]. Fizikalni model na kojem se temelji učinak ambijentalnog zasjenjenja uključuje mnoga pojednostavljenja, no učinak uvelike doprinosi realističnosti iscrtanog slikovnog okvira [1]. Metode izračunavanja ambijentalnog

zasjenjenja u općem slučaju zasnivaju se na istim konceptima kao i porodica algoritama praćenja zrake. Kako bi se odredio utjecaj učinka ambijentalnog zasjenjenja na određeni slikovni elementa, iz točke presjeka šalje se određeni broj uzoraka u nasumičnim smjerovima na hemisferi oko normale na površinu te se na temelju povratnih informacija i *Monte-Carlo* metode integracije određuje konačni rezultat [1]. Ovom metodom dobiva se točan iznos ovog učinka, no ovakav postupak je u općem slučaju neprikladan za dinamičke sustave. Stoga je ovakav način određivanja učinka ambijentalnog zasjenjenja pogodan za izračunavanje unaprijed. Zbog svega navedenog, razvijene su razne metode koje nude aproksimativan rezultat za učinak ambijentalnog zasjenjenja, a koje se mogu izvoditi u stvarnom vremenu. Jedna od najraširenijih metoda u današnjim dinamičkim sustavima za iscrtavanje virtualnih okruženja je određivanje ambijentalnog zasjenjenja na temelju informacija dostupnih u trenutnom vidnom polju kamere (engl. *Screen-Space Ambient Occlusion, SSAO*). Ovu metodu je razvio razvojni studio *Crytek* za potrebe sustava za iscrtavanja u igri *Crysis* [1]. Metode koje se oslanjaju na podatke dostupne u trenutnom vidnom polju kamere imaju konstantan trošak izračunavanja, za razliku od metoda koje se baziraju na informacijama u prostoru svih objekata, čiji je trošak proporcionalno raste s kompleksnošću virtualne scene [1]. S druge strane, izračunavanje učinka na temelju informacija dostupnih u trenutnom vidom polju kamere pati od istih problema kao i svaka metoda koja se bazira na tim informacijama. Problem je u tome što je skup podataka dostupan u trenutnom vidom polju kamere nepotpun te stoga su moguće situacije u kojim metoda generira netočne rezultate upravo zbog nedostupnosti svih informacija. Iz svega navedenog moguće je zaključiti kako porodica algoritama praćenja zrake predstavlja zlatni standard za izračunavanje učinka ambijentalnog zasjenjenja, no zbog svoje računalne kompleksnosti generalno je rezervirana za izračun statičnog ambijentalnog zasjenjenja.

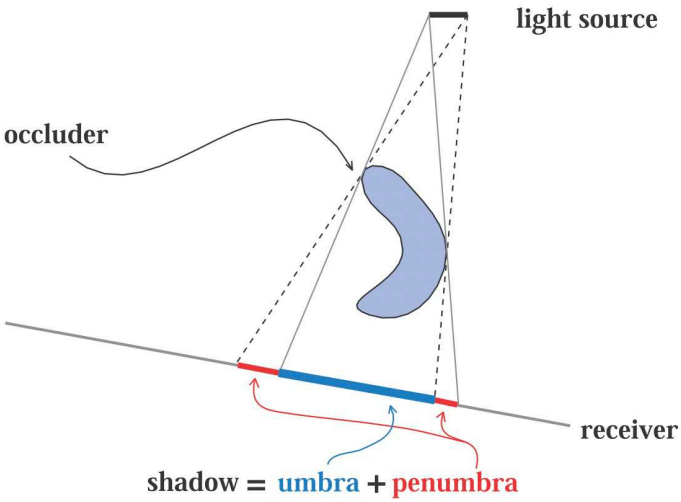


Slika 3.1.1: *Objekt zmaja iscrtan korištenjem samo ambijentalne komponente osvjetljenja (lijevo) te isti objekt iscrtan ambijentalne komponente osvjetljenja i učinka ambijentalnog zasjenjenja*

Kao što je već naglašeno, učinak ambijentalnog zasjenjenja zasnovan je na veoma pojednostavljenom fizikalnom modelu. Upravo zbog toga učinak ambijentalnog zasjenjenja može generirati neprirodne rezultate u mnogim situacijama [1].

3.2. Sjene

Simuliranje fenomenologije sjena izrazito je važna komponenta iscrtavanja virtualnih okruženja zbog toga što sjene ljudskom vizualnom sustavu olakšavaju percepciju odnosa unutar virtualnog okruženja [1]. Slika 3.2.1 prikazuje terminologiju koja se koristi prilikom razmatranja sjena u kontekstu računalne grafike. Sa slike 3.2.1 vidljivo je kako se sjena u općenitom slučaju sastoji od punog dijela sjene (engl. *Umbra*) i polusjene (engl. *Penumbra*). Sjene koje sadrže samo puni dio sjene uobičajeno se nazivaju tvrdim sjenama (engl. *Hard Shadows*), dok se sjene koje imaju i puni dio sjene i polusjenu nazivaju mekim sjenama (engl. *Soft Shadows*).



Slika 3.2.1: *Skica općeg slučaja procesa nastajanja sjena s uobičajenom pripadnom terminologijom*

Tvrde sjene su sjene koje nastaju pomoću točkastih izvora svjetlosti (engl. *Point Light*) [1]. Kako bi nastala meka sjena potreban je izvor svjetlosti koji ima površinu (engl. *Area Light*) [1]. Pošto su savršeni točkasti izvori svjetlosti rijetki u fizikalnoj stvarnosti, tako su u fizikalnoj stvarnosti rijetke i čiste tvrde sjene. No, tvrde sjene su mnogo manje zahtjevne za simulaciju od mekih sjena [1]. Također, usprkos tome što tvrde sjene rijetko kvalitetno opisuju fizikalnu stvarnost, ljudski vizualni sustav preferira sjene manje kvalitete ili vjernosti naspram izostanka sjena u potpunosti [1]. Važno je napomenuti kako se meke sjene ne mogu dobiti zamućenjem ruba tvrdih sjena. Naime, puni dio sjene je manji, ili može u potpunosti izostati, što je površina izvora svijetlosti veća u odnosu na objekt koji baca sjenu ili ako je objekt koji baca sjenu daleko od objekta na kojem se sjena prikazuje [1]. Postoje razne metode za generiranje sjena u sklopu procesa iscrtavanja virtualnih okruženja. No, mape sjena (engl. *Shadow Maps*) jedne su od daleko najrasprostranjenijih. Ideja mapa sjena je generirati Z-spremnik iz perspektive izvora svjetlosti. Ono što izvor svjetlosti „vidi“ ne nalazi se u sjeni, dok se sve drugo nalazi [1]. Osnovne mape sjena mogu generirati samo tvrde sjene. Metode filtriranja postotka bliskosti (engl. *Percentage-Closer Filtering, PCF*) i metoda postotka bliskosti za meke sjene (engl. *Percentage-Closer Soft Shadows, PCSS*) sposobne su generirati rezultate koje odražavaju ponašanje sjena u fizikalnoj stvarnosti. S druge strane, kako bi metoda postotka bliskosti za meke sjene generirala uvjerljive meke sjene, potrebno je primijeniti kvalitetne filtere. Ovi filteri mogu biti računalno zahtjevni. Ako se pak primjeni manje računalno zahtjevan filter, sjene mogu izgledati manje uvjerljivo. S druge strane,

generiranje tvrdih sjena pomoću porodice algoritama praćenja zrake je elementarno i sastoji se od samo jednog bacanja zrake prema izvoru svjetlosti po izvoru svjetlosti. Generiranje mekih sjena je također konceptualno jednostavno. Prema izvoru svjetlosti se ne baca samo jedna zraka, nego veći broj zraka prema različitim točkama na površini izvora svjetlosti. Konačni rezultat je određena vrsta prosjeka svih dobivenih uzoraka. Generiranje sjena pomoću porodice algoritama praćenja zrake izrazito je atraktivan prospekt zbog konceptualne jednostavnosti i preciznosti.

3.3. Globalno osvjetljenje

Pojam globalnog osvjetljenja je pojam u računalnoj grafici koji se koristi kako bi se označilo niz fizikalnih fenomenologija i učinaka. Globalno osvjetljenje se u općem slučaju može promatrati u kontekstu jednadžbe iscrtavanja. Jednadžba iscrtavanja daje unificirani matematički model za rješenje problema globalnog osvjetljenja. No, kao što je već nekoliko puta navedeno, rješavanje jednadžbe iscrtavanja u potpunosti je nemoguć zadatak u interaktivnim sustavima. Zbog toga je području računalne grafike uobičajeno problem osvjetljenja podijeliti na manje probleme koji se odnose na lokalno i globalno osvjetljenje [1]. Lokalno osvjetljenje se bavi problematikom sjenčanja određene točke prostora na temelju informacija o izvorima svjetlosti u sceni, dok se globalno osvjetljenje bavi sjenčanjem točke prostora na temelju svih interakcija svjetlosti u virtualnom okruženju [1]. Ovime se dolazi do još jedne podjele problema osvjetljenja na direktno i indirektno osvjetljenje. Modeli lokalnog osvjetljenja se primarno bave direktnim osvjetljenjem, dok se globalni modeli bave primarno indirektnim osvjetljenjem. Čistokrvni globalni modeli implicitno uključuju i direktno osvjetljenje. Najlakši način za razlučiti razliku između direktnog i indirektnog osvjetljenja je sljedeći: direktno osvjetljenje odgovara na pitanje što se događa u trenutku kada zraka svjetlosti iz izvora pogodi lokaliziranu točku na površini objekta, dok indirektno osvjetljenje odgovara na pitanje što se događa sa svjetlošću koja se odbija i propagira po virtualnom okruženju, te kako taj veliki broj odbijanja doprinosi konačnom izgledu određene točke u prostoru. Primjerice, ako se bijela kocka nalazi pored crvenog zida te scenu obasjamo bijelom svjetlošću, direktno osvjetljenje samo po sebi će odrediti kako je boja kocke bijela. No u stvarnosti, zbog indirektnog osvjetljenja, kocka će na dijelovima biti više ili manje crvena zbog indirektnog osvjetljenja nastalog odbijanjem svjetlosti o crveni zid. Slika

3.3.1 prikazuje prikaz virtualnog okruženja iscrtan korištenjem samo direktnog osvjetljenja, kao i prikaz istog virtualnog okruženja iscrtan kombinacijom direktnog i indirektnog osvjetljenja. Sa slike 3.3.1 vidljivo je koliko je važno indirektno osvjetljenje za vjerno simuliranje i prikazivanje fizikalne realnosti.

Slika 3.3.1: *Prikaz virtualnog okruženja iscrtan samo direktnim osvjetljenjem (lijevo) i kombinacijom direktnog i indirektnog osvjetljenja (desno)*

Kroz godine razvoja, za problem direktnog osvjetljenja razvijeni su razni modeli i pripadne metode rješavanja koje navedenu problematiku kvalitetno rješavaju, čak i u interaktivnim sustavima [1]. S druge strane, problem indirektnog osvjetljenja izrazito je zahtjevan problem za rješavanje u interaktivnim sustavima. Porodica algoritama praćenja zrake nudi unificirano i elegantno rješenje ovog problema. No, rješavanje problema indirektnog globalnog osvjetljenja izrazito je računalno zahtjevno za izvođenje u stvarnom vremenu pomoću ove porodice algoritama. Zbog toga je potrebno potražiti druga rješenja ovog problema, primjerice metodama baziranim na algoritmu rasterizacije. Relativno standardno rješenje u današnjim interaktivnim sustavima sastoji se od korištenja mapi svjetla (engl. *Lightmaps*) i svjetlosnih sondi (engl. *Light Probes*) [1]. Mape svjetla su teksture u kojima su zapisani rezultati prethodno izvedenog procesa izračunavanja indirektnog osvjetljenja. Proces izračunavanja se ne izvodi u stvarnom vremenu te se uobičajeno implementira algoritmima iz porodice algoritama praćenja zrake. Mape svjetla se uobičajeno koriste za objekte u virtualnoj sceni koji su statični te za koje je učinak indirektnog osvjetljenja stalan. S druge strane, svjetlosne sonde također sadrže rezultate već izračunatog indirektnog osvjetljenja. Za razliku od mapi svjetla, koje se primjenjuju na statične objekte, svjetlosne sonde služe za određivanje utjecaja indirektnog osvjetljenja na dinamičke objekte u virtualnom okruženju [1]. Razlika između mapi svjetla i svjetlosnih sondi se može opisati na način da mape

svjetla sadrže informacije o indirektnom osvjetljenju na površinama objekata, dok svjetlosne sonde sadrže informacije o indirektnom osvjetljenju u točkama praznog prostora u virtualnom okruženju [10]. Ovako opisan pristup indirektnom osvjetljenju nije bez svojih mana. Jedna od glavnih mana je sveukupna tromost na dinamičke promjene u sceni. Usprkos tome što se svjetlosnim sondama nastoji smanjiti navedena tromost, koja je posebno izražena kod mapi svjetla, svjetlosne sonde su ograničene učestalošću osvježavanja podataka koji se u njima nalaze. Učestalije osvježavanje drastično povećava potrebu za računalnim resursima, dok svjetlosne sonde koje informacije ne osvježavaju uopće prilikom izvođenja pate od nemogućnosti uračunavanja dinamičkih objekata u izračune indirektnog osvjetljenja za druge dinamičke objekte ili pak ostale statične objekte [10]. Navedeno rješenje se primarno koristi za izračunavanje difuznog indirektnog globalnog osvjetljenja, te nije prikladno za rješavanje problema indirektnog spekularnog globalnog osvjetljenja [1]. Indirektno spekularno globalno osvjetljenje ima drugačija svojstva od difuznog. Na primjer, indirektno spekularno globalno osvjetljenje se na sjajnim površinama manifestira kao pojava refleksija [1]. Kao što je već ranije diskutirano, za izračun refleksija potrebne su zrake iz uskog stošca oko smjera zrcalne refleksije. S druge strane, za izračun difuznog osvjetljenja potrebne su zrake iz svih smjerova na hemisferi oko normale na površinu u točki sjenčanja. Zbog toga mape svjetla i svjetlosne sonde nisu prikladne za izračun indirektnog spekularnog globalnog osvjetljenja. Stoga je uobičajena praska ove dvije komponente osvjetljenja promatrati kao odvojene probleme s različitim pripadnim rješenjima [1]. Jedno od uobičajenih rješenja za izračun indirektnog spekularnog globalnog osvjetljenja su lokalizirane mape okoliša (engl. *Localized Environment Maps*). Lokalizirane mape okoliša poseban su slučaj općenitih mapa okoliša. Ova vrsta mapa okoliša postavlja se na određena mjesta u virtualnom okruženju te se u njoj pohranjuju informacije o indirektnom spekularnom osvjetljenu koje se zatim koristi za izračun utjecaja indirektnog globalnog spekularnog osvjetljenja na objekte u virtualnom okruženju. Problem lokaliziranih mapa okoliša leži u osvježavanju pohranjenih informacija. Kako bi ove mape okoliša pravilno odražavale dinamičke promjene u virtualnom okruženju, potrebno je osvježavati njihov sadržaj. S druge strane, cijena osvježavanja u stvarnom vremenu raste s brojem lokaliziranih mapa okoliša, te je u današnjim dinamičkim sustavima gotove nemoguće sve lokalizirane mape okoliša osvježavati prilikom iscrtavanja svakog slikovnog okvira [1]. Za simulaciju

indirektnog spekularnog osvjetljenja moguće je koristiti i druge metode poput grupe metode koje se zasnivaju na podacima dostupnim u trenutnom vidom polju kamere (engl. *Screen-Space Reflections, SSR*). Nedostatak ovih metoda leži upravo u limitiranom skupu informacija na temelju kojeg se simulira učinak indirektnog spekularnog globalnog osvjetljenja. Posljedica je ta da je nemoguće u refleksijama prikazati objekte koji se nalaze izvan vidnog polja kamere. Vidljivo je kako je porodica algoritama praćenja zrake ponovno najprikladnije rješenje za simulaciju učinka indirektnog osvjetljenja, no velika računalna kompleksnost sprječava njezinu direktnu primjenu u interaktivnim sustavima.

Kod svih navedenih učinaka porodica algoritama praćenja zrake koristi se za izračun kvalitetnih rezultata. S druge strane, velika računalna kompleksnost onemogućuje direktnu primjenu ove porodice algoritama na dinamičko izračunavanje učinaka u stvarnom vremenu, te je rezultate potrebno prethodno izračunati i pohraniti u prikladnom obliku.

4. Primjena algoritama praćenja zrake i praćenja puta u interaktivnim sustavima

Kao što je već spomenuto ranije tokom ovog diplomskog rada, familija algoritama praćenja zrake nije nipošto nova familija algoritama. Ova familija algoritama se razvija još od 70ih godina prošlog stoljeća, te je veoma brzo postala zlatni standard za iscrtavanje virtualnih okruženja u području računalne grafike zbog svoje sposobnosti da izrazito elegantno simulira fizikalnu fenomenologiju širenja svijetlosti. No zbog svoje računalne kompleksnosti, ova familija algoritama generalno je bila rezervirana za iscrtavanje virtualnih okruženja u sustavima koji nisu zahtijevali iscrtavanje u stvarnom vremenu. Izrada specijalnih učinaka u filmskoj industriji, kao i izrada trodimenzionalnih crtanih filmova, primjeri su područja primjene familije algoritama praćenja zrake kroz sve godine njihovog razvoja. S druge strane, ova familija algoritama je bila jednostavno previše računalno kompleksna za iscrtavanje virtualnih okruženja u interaktivnim sustavima, poput onih za izradu i pokretanje računalnih igara. Ovo stanje se održalo sve do 2018. godine, kada je tvrtka *Nvidia* predstavila arhitekturu grafičkog procesora *Turing*. Ova arhitektura grafičkih procesora, koja je ime dobila po znanstveniku Alanu Turingu, prva je arhitektura grafičkih procesora koja u svoju arhitekturu ukomponirala sklopovske akceleracijske strukture koje su namijenjene ubrzavanju kritičnih točaka izvođenja algoritama iz familije algoritama praćenja zrake.

4.1. Programske akceleracijske strukture i postupci

Već je nekoliko puta tokom ovog diplomskog rada iznijeta činjenica kako se porodica algoritama praćenja zrake primarno koristi u sustavima za iscrtavanje koji nisu interaktivni. Primjer ovakvog sustava je sustav za iscrtavanje slikovnih okvira prilikom izrade trodimenzionalnih animiranih filmova. U ovakvim sustavima se na iscrtavanje jednog slikovnog okvira može potrošiti nekoliko redova veličine više vremena nego u sustavima koji zahtijevaju interaktivnost. Primjerice, u sustavima za pokretanje računalnih igara, ako želimo postići razinu izvođenja od 60 slikovnih okvira u sekundi (engl. *Frames Per Second, FPS*), za jedan slikovni okvir imamo budžet od otprilike 16ms. S druge strane, vremensko ograničenje za iscrtavanje

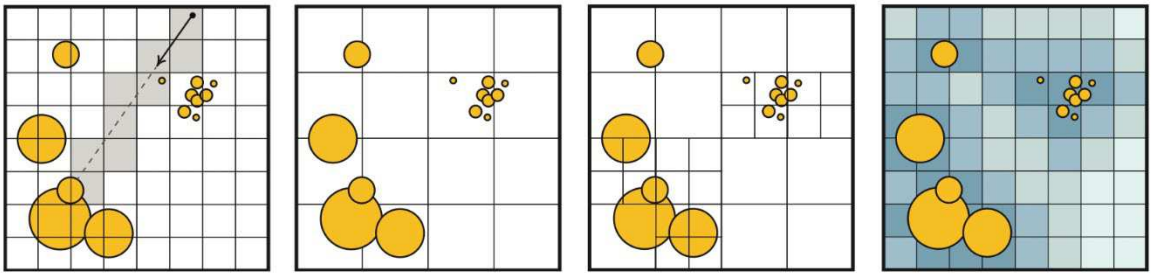
jednog slikovnog okvira animiranog filma je jedino ograničeno ukupnim brojem slikovnih okvira i ukupnim vremenom predviđenim za izradu animiranog filma. Usprkos tome što je vremensko ograničenje u interaktivnim sustavima nekoliko redova veličina manje od onog u sustavima koji nisu interaktivni, to ne znači da ograničenje u sustavima koji nisu interaktivni ne postoji. Stoga je važno konstruirati sustav koji će imati konzistentno vrijeme iscrtavanja jednog slikovnog okvira. Kako bi se slikovni okviri mogli generirati u zadanom ograničenom vremenu, važno je konstruirati sustav koji će iscrtavanje provoditi optimalno. Važnu ulogu u optimalnom izvršavanju porodice algoritama praćenja zrake imaju programske akceleracijske strukture i postupci. Ove strukture i postupci su specifični za ovu porodicu algoritama, no počivaju na univerzalnom konceptu u području računarstva, konkretnije konceptu iskorištavanja koherentnosti operacija prilikom izvođenja kako bi se poboljšale performanse sklopovskog ili programskog sustava. [1] Iskorištavanje koherentnosti operacija prilikom izvođenja očituje se u mogućnosti ponovnog iskorištavanja dijela rezultata operacija koje su već provedene. Kako bi se dobio osjećaj o tome kako se koherencija očituje u algoritmima za iscrtavanja virtualnih okruženja, ponovno će se promotriti algoritam rasterizacije. [1] Jedan od glavnih razloga zašto je algoritam rasterizacije izrazito brz i učinkovit algoritam leži u tome što u njegovim operacijama prirodno postoje visoki stupnjevi koherencije. Prisjetimo se kako je algoritam rasterizacije moguće opisati pomoću dvije ugniježđene petlje. [1] Ako je algoritam moguće opisati ugniježđenom petljom, u općenitom slučaju operacije koje se odvijaju kao dio unutarnje petlje su operacije koje se izvršavaju jedna iza druge te su zbog toga dobri kandidati pronalazak koherentnosti [1]. Unutarnja petlja algoritma rasterizacije izvršava operacije nad slikovnim elementima rastera, te određuje koji se slikovni elementi nalaze unutar poligona koji se trenutno obrađuje. [1] U ovim operacijama moguće je identificirati prostornu koherentnost, odnosno moguće je odraditi određivanje vidljivosti nad malenim grupama slikovnih elemenata umjesto nad pojedinačnim slikovnim elementima te prilikom takvog načina određivanja vidljivosti postoji velika vjerojatnost da je odgovor na pitanje nalaze li se slikovni elementi unutar poligona ili ne jednak za sve slikovne elemente u grupi. No, koherentnost se u algoritmu rasterizacije ne očituje samo u operacijama određivanja vidljivosti. U procesu sjenčanja, koji dolazi poslije procesa određivanja vidljivosti, također je moguće identificirati visoki stupanj koherentnosti. Proces sjenčanja u sklopu rasterizacijskog

funkcijskog cjevovoda je pod direktnim utjecajem samog rasterizacijskog algoritma za određivanje vidljivosti. Fragmenti koji se sjenčaju u određenom trenutku uobičajeno pripadaju istom poligonu, što znači da se sjenčaju istim materijalom, teksturiraju istom teksturom ili čak pristupaju prostorno bliskim elementima teksture. To znači da je i proces sjenčanja u kontekstu rasterizacijskog funkcijskoj cjevovoda prirodno koherentan. S druge strane, porodica algoritama praćenja zrake ne pokazuje ovako prirodno velike stupnjeve koherencije, te se velika većina razvoja optimizacijskih postupaka za ovu porodicu algoritama odnosi na pronalazak koherencije u operacijama ove porodice algoritama. U nastavku ćemo se osvrnuti na nekoliko većih generalnih skupina koherentnosti te kako se one koriste za izgradnju programskih akceleracijskih struktura za porodicu algoritama praćenja zrake.

4.1.1. Prostorna koherencija

Iskorištavanje prostorne koherencije izrazito je važan dio izrade učinkovitog sustava iscrtavanja zasnovanom na porodici algoritama praćenja zrake. Objekti i primitive u virtualnom okruženju tvore prirodne odnose na temelju međusobne prostorne udaljenosti [1]. Ovo nužno ne garantira koherentnost operacija sjenčanja, zbog toga što objekti koji su prostorno međusobno blizu mogu biti sačinjeni od različitih materijala te stoga koristiti različite procesore za sjenčanje u procesu sjenčanja [1]. No, iskorištavanjem prostorne koherencije moguće je ubrzati proces određivanja vidljivosti za pojedinu zraku. Drugim riječima, moguće je ubrzati operacije koji se odvijaju unutar unutarnje petlje algoritma praćenja zrake. Podsjetimo se unutarnja petlja algoritma praćenja zrake iterira po svima primitivama koji tvore zadano virtualno okruženje. Iskorištavanjem prostorne koherencije moguće je preskočiti određeni broj primitiva u ovom procesu. Upravno je ovo razlog zašto velika većina sustava koji proces iscrtavanja baziraju na porodici algoritama praćenja zrake koriste prostorne akceleracijske strukture prilikom određivanja presjeka zrake i primitiva u virtualnom okruženju. Prostorne akceleracijske strukture nastoje grupirati geometriju koja tvori virtualno okruženje u prostorne grupe tako da se u danoj grupi nalazi geometrija koja je međusobno blizu u prostoru. Ovime dobivamo vrstu prostornih akceleracijskih struktura koja se konstruira grupiranjem objekata odnosno primitiva [1]. Također je moguće promatrati prostorne akceleracijske

strukture na način da one dijele prostor virtualnog okruženja po nekoj shemi. Time se dobiva vrsta prostornih akceleracijskih struktura koja se konstruira algoritmima koji dijele prostor po određenim pravilima [1]. Prva slika na slici 4.1.1.1 prikazuje najjednostavniju podjelu virtualnog prostora pomoću uniformne mreže. Ova slika također prikazuje na što se svodi proces pronalaska najbližeg presjeka zrake i geometrije u sceni. Sivi kvadratići predstavljaju ćelije prostora koji se posjećuju prilikom procesa pronalaska presjeka. Sa slike je moguće primijetiti kako je u ovom procesu preskočen veliki broj geometrije koja se nalazi u ćelijama s lijeve i desne strane pravca na kojem leži zraka. Uniformna mreža se rijetko koristi u praksi [1] no služi kao odličan ilustrativan primjer onoga što prostorne akceleracijske strukture rade s prostorom virtualnog okruženja i zašto. Na slici 4.1.1.1 su također prikazane i uniformna mreža u dvije razine (druga slika gledano s desna), kao i uniformna mreža s oblacima blizine (engl. *Proximity Clouds*) (prva slika gledano s desna) [1]. Oblaci blizine sadrže informacije o udaljenosti do najbliže jedinice prostora koja nije prazna. Ovi oblaci blizine omogućavaju efikasno preskakanje praznih ćelija prostorne akceleracijske strukture [1]. Uniforma mreža u dvije razine zapravo je primitivna hijerarhijska struktura. Veće ćelije ne sadrže popis primitiva u ćeliji, nego sadrže novu uniformnu mrežu finije zrnatosti za dio prostora sadržan u toj većoj ćeliji.



Slika 4.1.1.1: *Uniformna mreža finije zrnatosti (prva slika s lijeva), uniformna mreža grublje zrnatosti (druga slika s lijeva), uniformna mreža u dvije razine (druga slika s desna) i uniformna mreža finije zrnatosti s oblacima blizine (prva slika s desna)*

Ovu ideju o hijerarhijski ugniježdenim prostornim akceleracijskim strukturama moguće je razraditi do njezinog limita, u kojoj vrh strukture sadrži podjelu prostora najmanje zrnatosti, a svaka sljedeća razina sadrži rekurzivno podijeljen dio prostora

virtualnog okruženja rekurzivno manje zrnatosti. Primjeri ovakvih struktura su sljedeći [1]:

- Oktalno stablo (engl. *Octree*)
- BSP stablo (engl. *BSP tree*)
- K-d stablo (engl. *K-d tree*)
- BIH stablo (engl. *Bounding Interval Hierarchy Tree, BIH Tree*)
- Hijerarhija obujmica (engl. *Bounding Volume Hierarchy, BVH*)

Oktalno stablo je hijerarhijska struktura koja započinje s ćelijama veličine 2x2x2 na svakoj osi te nadalje rekurzivno dijeli svaku ćeliju koja nije prazna na isti način [1]. BSP stablo i K-d stablo nastaju odabirom ravnine koja dijeli ćeliju određene razine na dva dijela. Razlika je u tome što je ravnina proizvoljna kod BSP stabla, dok kod K-d stabla ravnina mora biti poravnana s koordinatnim osima [1]. Ako se umjesto jedna ravnine poravnate s osima na svakoj razini podijele odaberu dvije, dobiva se BIH stablo [1]. Hijerarhija obujmica je daleko najrašireniji prostorna akceleracijska struktura za ubrzavanje porodice algoritama praćenja zrake [1]. Ova vrsta prostorne akceleracijske strukture koristi obujmice, najčešće one poravnate s koordinatnim osima (engl. *Axis-Aligned Bounding Boxes, AABB*), kao elementarnu jedinicu prostora. Važno je napomenuti kako BVH strukture ne dijele prostor u disjunktne potprostore, odnosno moguće je da se obujmice u hijerarhijskoj strukturi preklapaju. BVH strukture imaju poželjna svojstva poput [1]:

- Maksimalno memorijsko zauzeće je poznato unaprijed
- Potreban je manji broj podjela prilikom izgradnje
- Učinkovite su u preskakanju praznog prostora
- Algoritmi za konstrukciju su brzi i visoke kvalitete
- Pogodne su za animirana virtualna okruženja, pogotovo ako okruženja imaju izraženu vremensku koherenciju

No BVH strukture imaju i određena nepovoljna svojstva u odnosu na neke druge prostorne akceleracijske strukture poput [1]:

- Povećane kompleksnosti izgradnje
- Više potrebne memorije za spremanje jednog čvora

- Nemogućnosti primjene striktnog redoslijeda primitiva od nazad prema naprijed što onemogućuje rano zaustavljanje prilikom procesa pronalaska presjeka

Sumarno, pozitivna svojstva BVH struktura su daleko veća i važnija od onih negativnih. Nadalje, neka negativna svojstva, poput nemogućnosti ranog zaustavljanja, ne predstavljaju veliki problem u praksi i ne manifestiraju se kao usporenje u odnosu na druge prostorne akceleracijske strukture, poput K-d stabla, koje imaju navedenu mogućnost. Inženjeri su kroz godine razvoja ovog područja razvili različite inačice općenite BVH strukture koje nastoje razriješiti navedena negativna svojstva, kao i poboljšati različita navedena pozitivna svojstva. Važno je napomenuti kako se BVH struktura koristi u sklopu RTX platforme [3]. BVH strukturu koriste sklopovske akceleracijske strukture ove porodice grafičkih procesora o kojima će biti riječi kasnije.

4.1.2. Koherencija zraka

Kako bi generirali jedan slikovni okvir, algoritmi iz porodice algoritama praćenja zrake moraju generirati i pratiti veliki broj zraka. Primjerice, ako je potrebno generirati slikovni okvir rezolucije 1920x1080 i ako se generira samo jedna zraka po slikovnom elementu, potrebno je generirati malo više od dva milijuna zraka. Za svaku od ovih zraka moramo odrediti sječe li geometriju koja sačinjava virtualno okruženje. Također, proces sjenčanja će u točkama presjeka vrlo vjerojatno generirati veliki broj novih zraka za svaku od zraka kamere. Kako bi se ubrzao cjelokupan postupak iscrtavanja slikovnog okvira, potrebno je identificirati koje su generirane zrake koherentne i koliki je stupanj koherentnosti. Prvo se identificira koherencija vezana uz samo praćenje zrake, odnosno postupak pronalaska presjeka zraka i geometrije u virtualnoj sceni. Drugim riječima, pokušava se pronaći koherentnost u operacijama određivanja vidljivosti. Procesi generiranja zraka na različitim mjestima u algoritmima iz porodice algoritama praćenja zrake generiraju zrake koje pokazuju različite stupnjeve koherentnosti [1]. Primjerice, proces generiranja zraka kamere generira zrake koje imaju istu početnu točku, koja je ista kao i točka u kojoj se nalazi kamera, dok su smjerovi zraka kamere bliski i ograničeni vidim poljem kamere [1]. Možemo zaključiti da ove zrake pokazuju određenu dozu koherentnosti. Zrake sjena koje se koriste za određivanje je li točka presjeka u sjeni

u odnosu na izvore svijetlosti koji se nalaze u beskonačnosti također pokazuju prirodnu koherentnost [1]. Čak i zrake refleksije sadrže određenu dozu koherentnosti ako se promotri primjer u kojemu se zrake dva susjedna slikovna elementa sijeku s istim objektom u sceni, te se u točkama presjeka generiraju zrake refleksija kojima su početne točke blizu u prostoru te su normale u točki presjeka također slične. S druge strane, zrake koje se generiraju kako bi se evaluirali učinci poput ambijentalnog zasjenjenja i difuznog globalnog osvjetljenja generiraju se nasumično u svim smjerovima na hemisferi oko normale u točki prostora koja se sjenča te su zbog toga prirodno nekoherentne. Rane ideje pomoću kojih se nastojalo iskoristiti koherenciju zraka prožimala je ideja grupiranja zraka u određen geometrijski oblik. Takvi geometrijski konstrukti su na primjer stošci (engl. *Cone*) ili olovke (engl. *Pencil*) [1]. Problem s ovim metodama je to što pretpostavljaju velike kontinuitete na površinama što je rijedak slučaj u virtualnim okruženjima koji modeliraju stvarni svijet. Fleksibilnija varijanta ovakvih algoritama je varijanta praćenja paketa (engl. *Packet Tracing*). Ideja ove varijante iskorištavanja koherencije zraka oslanja se na praćenje određenog broja zraka koji se nalaze u paketu kroz prostornu akceleracijsku strukturu. Paketi u ovoj varijanti nisu geometrijski oblici, nego standardna struktura podataka. Stoga je razrješavanje slučaja u kojem se paket mora razdijeliti, zbog toga što dijelovi zraka moraju pratiti različite dijelove prostorne akceleracijske strukture, jednostavno za razliku od onog kada su paketi geometrijski konstrukti. Operacije potrebne za prolazak kroz prostornu akceleracijsku strukturu mogu se izvršavati paralelno za zrake u paketu te su stoga pogodne za izvršavanje na SIMD procesorima kao što je grafičko sklopovlje. Standardna je praksa da se veličina paketa postavi na vrijednost koja odgovara širini SIMD instrukcija grafičkog procesora. Ova metoda iskorištavanja koherencije zraka korisna je samo kada prilikom prolaska kroz prostornu akceleracijsku strukturu ne dolazi do većeg broja podjela paketa i divergencije zraka. Paketi neće divergirati ako su zrake u paketu prirodno koherentne. No, u idealnom slučaju, poželjno bi bilo pronaći koherenciju i u zrakama koje nisu prirodno koherentne. Praćenje paketa se oslanja na SIMD instrukcije kako bi se odredio presjek više zraka s jednim čvorom prostorne akceleracijske strukture. Ovaj odnos broja zraka i čvorova je moguće obrnuti. Tako je moguće SIMD instrukcije iskoristiti kako bi se odredio presjek jedne zrake i više čvorova iste razine u prostornoj akceleracijskoj strukturi. [1] Ako se na ovaj način nastoji iskoristiti koherencija zraka,

potrebne su plitke i široke prostorne akceleracijske strukture [1]. Primjer ovakve prostorne akceleracijske strukture je posebna vrsta hijerarhije obujmica BVH poznata pod nazivom hijerarhija višestrukih obujmica (engl. *Multi-Bounding Volume Hierarchy, MBVH*). Ova inačica BVH strukture dobiva se djelomičnim izravnavanjem standardne binarne BVH prostorne akceleracijske strukture. Još jedan način za iskorištavanje koherencije zraka je poznat pod nazivom memorijski koherentno praćenje zrake. Ideja ovog postupka je ta da umjesto da se prati zraka sve dok joj se ne pronađe točka presjeka s nekom od primitiva u virtualnom okruženju, zrake se pohranjuju u čvorove prostorne akceleracijske strukture te ih se dinamički sortira tako da se nakon sortiranja dobivaju grupe zraka koje su koherentne te ih se može procesuirati zajedno. Motivacija iza ovog pristupa je ta da, primjerice, algoritam praćenja puta u jednom trenutku obrađuje zrake kojima su svi smjerovi na neki način nasumični. U takvoj skupini zraka teško je pronaći koherentnost na lokalnoj razini, no ako se sagledaju sve zrake koju su stvorene za generiranje rezultata, moguće je pronaći zrake koje pokazuju određen stupanj koherencije. Ovaj pristup rezultira pretraživanjem prostorne akceleracijske strukture u širinu [1].

4.1.3. Koherencija sjenčanja

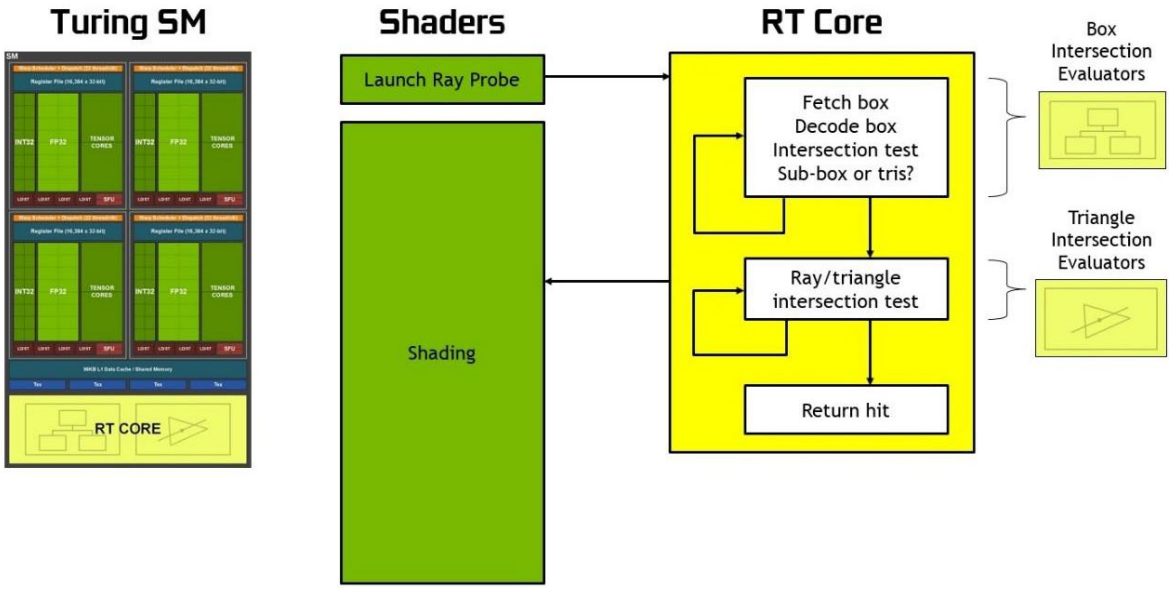
Koherencija zraka se primarno odnosi na pronalazak koherentnosti operacija određivanja vidljivosti. S druge strane, određivanje vidljivosti samo je jedan dio cjelokupnog algoritma praćenja zrake i njegovih inačica. Veliki dio operacija koje se izvode u sklopu algoritma praćenja zrake i njegovih inačica odnose se na sjenčanje točaka presjeka zraka i objekata u virtualnom okruženju. Iako bi se, promatrajući koherenciju zraka, moglo zaključiti kako je koherencija sjenčanja direktna posljedica koherencije zraka, ovo nužno ne mora biti slučaj [1]. Lako je moguće da dvije zrake, čija su sjecišta s objektima u sceni međusobno blizu u prostoru, sijeku objekte s potpuno drugačijim materijalima. Različiti materijali zahtijevaju različite procesore za sjenčanje. U ovom slučaju ne mogu se iskoristiti SIMD instrukcije istog procesora za sjenčanje kako bi se osjenčale obije točke presjeka paralelno. Zbog toga se ideja sortiranja zraka predstavljena u poglavlju 4.1.2 može proširiti i na koherenciju sjenčanja. Ideja je da se zrake dodaju u redove nakon što im je određen presjek s objektom u sceni, te da se tada sortiraju tako da se zrake s točkama presjeka koji su vezani uz isti materijal sjenčaju paralelno pomoću SIMD instrukcija grafičkog

sklopovlja [1]. Ovakvim pristupom se točke presjeka sjenčaju koherentno. Ovime se implicitno odvaja proces određivanja vidljivosti od procesa sjenčanja. Ideja odvajanja ovih dvaju procesa nije novost u području računalne grafike. Naprotiv, upravo je ovo ideja koja stoji iza cjevovoda za iscrtavanje koji se temelji na takozvanom odgođenom sjenčanju i iscrtavanju (engl. *Deferred Rendering*). Ovakav cjevovod je u današnje vrijeme izrazito često korišten u svrhu iscrtavanja u interaktivnim sustavima [1]. Izrazito je važno napomenuti kako treba biti veoma pažljiv prilikom implementiranja sustava za sortiranje zraka u svrhu iskorištavanja koherencije zraka. Naime, potrebno je odrediti je li vremenski trošak sortiranja zraka manji od dobivenog vremenskog smanjenja prilikom procesa sjenčanja u odnosu na sustav bez sortiranja zraka. Sustav sortiranja se već uspješno koristi u sustavima koji nisu interaktivni [1]. Još jedna ideja za iskorištavanje koherencije sjenčanja odnosi se na simuliranje fizikalne fenomenologije, poput ambijentalnog zasjenjenja i difuznog globalnog osvjetljenja, koja zahtjeva praćenje prirodno nekoherentnih zraka. Ideja je ta da za bliske slikovne elemente namjerno odabiremo nekoherentne zrake i iskoristimo filtar za uklanjanje šuma kako bi se generirao krajnji rezultat. Više o procesu uklanjanja šuma i filtrima koji se za taj proces koriste biti će riječi u nastavku ovog poglavlja.

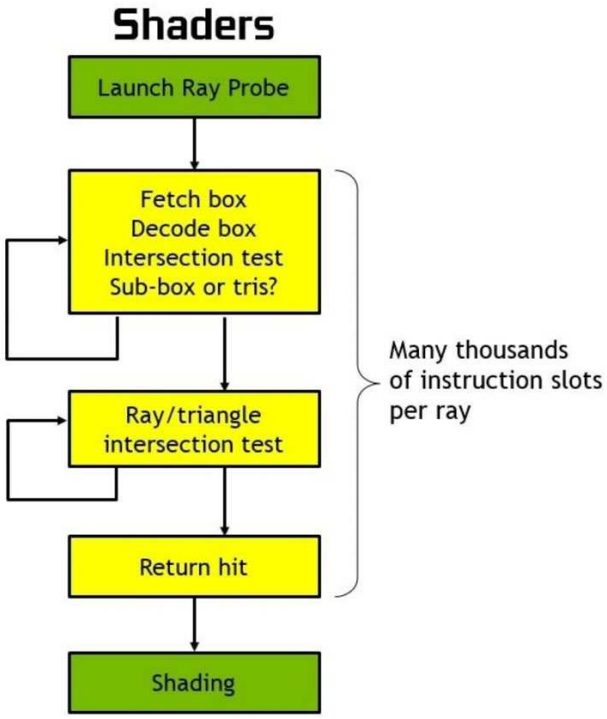
4.2. Sklopovske akceleracijske strukture

Ranije u ovom poglavlju navedeno je kako je jedna od primarnih programskih akceleracijskih struktura koja se koristi za ubrzavanje algoritama praćenja zrake prostorna akceleracijska struktura koja dijeli prostor na način koji ubrzava proces pronalaska presjeka zrake s nekom od primitiva koje tvore virtualnu scenu. Prilikom korištenja prostornih akceleracijskih struktura pronalazak presjeka zrake i primitive svodi se na prolazak kroz navedenu prostornu akceleracijsku strukturu. Bez obzira na to što prostorne akceleracijske strukture bitno ubrzavaju proces pronalaska presjeka same po sebi, ovaj dio algoritma i dalje ostaje izrazito računalno kompleksan i vremenski zahtjevan [3]. Zbog toga je tvrtka *Nvidia* u sklopu svoje Turing arhitekture grafičkih procesora predstavila takozvane RT jezgre (engl. *RT Cores*) koje izvode cijeli proces pronalaska presjeka zrake i primitive u virtualnom okruženju. RT jezgra je jezgra koju sadrži svaka SM jedinica u grafičkom procesoru. Prilikom izvođenja algoritma praćenja zrake, SM jedinica delegira posao pronalaska

presjeka RT jezgri. [3] Desni dio slike 4.2.1 prikazuje funkcionalni izgled RT jezgre. Sa slike 4.2.1 vidljivo je da se RT jezgra sastoji od jedinica za određivanje presjeka s obujmicama koje tvore prostornu akceleracijsku strukturu BVH (engl. *Box Intersection Evaluators*) i jedinicu za određivanje presjeka s primitivom trokuta (engl. *Triangle Intersection Evaluators*). RT jezgra SM jedinici vraća informaciju o pronađenom presjeku, ili o nepostojanju presjeka. [3] Na slici 4.2.2 prikazan je dijagram izvođenja algoritma praćenja zrake na arhitekturi grafičkog procesora Pascal, koja ne sadrži RT jezgru u svojim SM jedinicama. Slike 4.2.2 zorno prikazuje kako posao pronalaska presjeka u tom slučaju mora obaviti SM jedinica. [3] Proces pronalaska presjeka može zauzeti više tisuća instrukcija SM jedinice po jednoj zraci koja se prati. Na slici 4.2.1 je vidljivo kako SM jedinica samo generira zraku i delegira posao pronalaska presjeka RT jezgri. Time SM jedinica može iskoristiti slobodne instrukcije za obavljanje poslova poput sjenčanja vrhova ili sjenčanja slikovnih elemenata. Novija arhitektura grafičkih procesora Ampere nastavila je razvijati RT jezgre koje su postale neizostavan dio u omogućavanju izvođenja algoritama iz porodice algoritama praćenja zrake u stvarnom vremenu.



Slika 4.2.1: *Izgled SM jedinice u porodici grafičkih procesora Turing (lijevo) i dijagram izvođenja algoritma praćenja zrake, kao i funkcionalni izgled RT jezgre, u navedenoj porodici grafičkih procesora (desno)*



Slika 4.2.2: *Dijagram izvođenja algoritma praćenja zrake na arhitekturi grafičkih procesora Pascal bez RT jezgara*

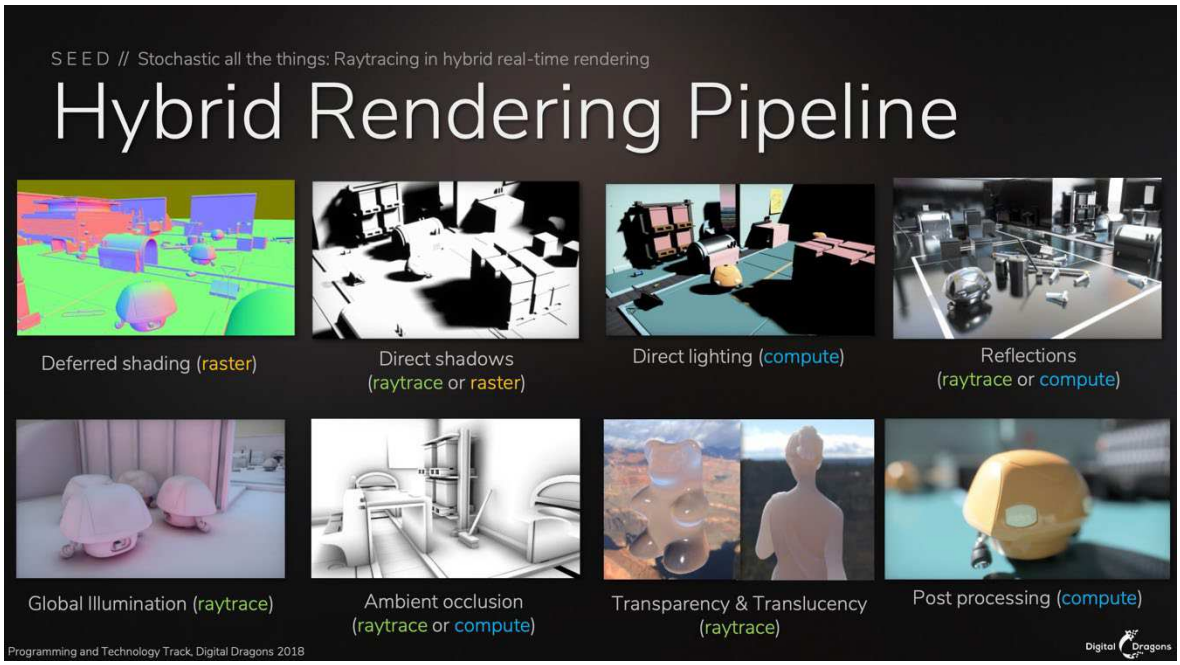
4.3. Hibridni cjevovod za iscrtavanje

Glavni nedostatak algoritama iz porodice algoritama praćenja zrake identificiran je već nekoliko puta tokom ovog rada. Najveći nedostatak ovih algoritama je njihova izrazito velika računalna kompleksnost. Ta velika računalna kompleksnost ih čini izrazito nepogodnima za iscrtavanje slikovnih okvira u interaktivnim sustavima. Nakon dugog niza godina razvoja grafičkog sklopovlja, kao i raznih sklopovskih i programskih akceleracijskih struktura, napokon je moguće primijeniti porodicu algoritama praćenja zrake na iscrtavanje virtualnih okruženja u stvarnom vremenu. No, izvršavanje cijelog procesa iscrtavanja pomoću algoritama iz ove porodice je i dalje izrazito zahtjevno za računalne resurse, pogotovo za kompleksna virtualna okruženja, poput onih koje je moguće pronaći u današnjim računalnim igrama. Također, odbaciti godine razvoja rasterizacijskih algoritama i metoda, kao i pripadajućih sklopovskih i programskih akceleracijskih struktura, bilo bi veoma nesuvislo. Činjenica je da rasterizacijske metode prilikom simulacije određene fizikalne fenomenologije pate od raznih artefakata i pogrešaka, no za određene postupke koji su sastavni dio grafičkog funkcijskog cjevovoda, poput određivanja

vidljivosti, rasterizacijske metode su izrazito precizne, brze i efikasne. Ako se pobliže razmisli o svemu upravo navedenom, prirodno se rađa ideja o kombiniranju porodice algoritama praćenja zrake i rasterizacijskih algoritama i metoda. Time se dolazi do koncepta hibridnog cjevovoda za iscrtavanje. Ideja iza ovog koncepta je izrazito jednostavna. Rasterizacijske metode i algoritmi se i dalje koriste za dijelove grafičkog funkcijskog cjevovoda u kojima su navedene metode i algoritmi precizni i generiraju kvalitetne rezultate, što zbog toga što su te metode i algoritmi izrazito dobro optimizirani kroz godine razvoja, što zbog toga što neki posjeduju i sklopovske akceleracijske strukture koje su također usavršavane tokom godina njihovog postojanja. S druge strane, porodica algoritama praćenja zrake se koristi za generiranje učinaka koji nastoje simulirati kompleksnu fizikalnu fenomenologiju za čije generiranje ne postoji ekvivalentna rasterizacijska metoda koja daje rezultate bez artefakata i pogrešaka. Ovo konceptualno jednostavno i elegantno rješenje omogućuje generiranje prikaza virtualnih scena visoke kvalitete, u kojima je fizikalna fenomenologija korektno simulirana, a koje je usprkos tomu moguće generirati brzom koju zahtijevaju interaktivni sustavi. U nastavku ćemo razmotriti jedan primjer ovakvog hibridnog cjevovoda za iscrtavanje kojeg je osmislila istraživačko odjeljenje *SEED* kompanije *EA*, a koji zorno prikazuje prednosti ovakvog pristupa iskorištavanju porodice algoritama praćenja zrake u interaktivnim sustavima.

4.3.1. Projekt Pica Pica istraživačkog tima SEED

Projekt *Pica Pica* je projekt istraživačkog tima *SEED* koji je dio kompanije *EA*. Cilj ovog projekta bio je istražiti mogućnosti iscrtavanja virtualnih okruženja korištenjem hibridnog cjevovoda bez izračunavanja učinaka prije vremena [4]. To znači da se svi učinci generiraju u stvarnom vremenu. Slika 4.3.1.1 prikazuje hibridni cjevovod kojeg je razvojni tim *SEED* osmislio i implementirao u svrhu iscrtavanja slikovnih okvira u sklopu projekta *Pica Pica*.



Slika 4.3.1.1: *Hibridni cjevovod za iscrtavanje u sklopu projekta Pica Pica*

Promatrajući sliku 4.3.1.1 može se razlučiti koji su dijelovi iscrtavanja implementirani pomoću rasterizacijskih metoda, a koji su implementirani pomoću porodice algoritama praćenja zrake. Na temelju tih informacija konstruirana je tablica 4.3.1.1 za lakši pregled koje se faze cjevovoda implementiraju kojom skupinom metoda. Dijelovi cjevovoda se izvršavaju pomoću procesora za sjenčanje za računanje što je također postao standard za postupke poput postupaka naknadne obrade (engl. *Post processing*).

Tablica 4.3.1.1: *Faze iscrtavanja i metode pomoću kojih se one izvršavaju u projektu Pica-Pica*

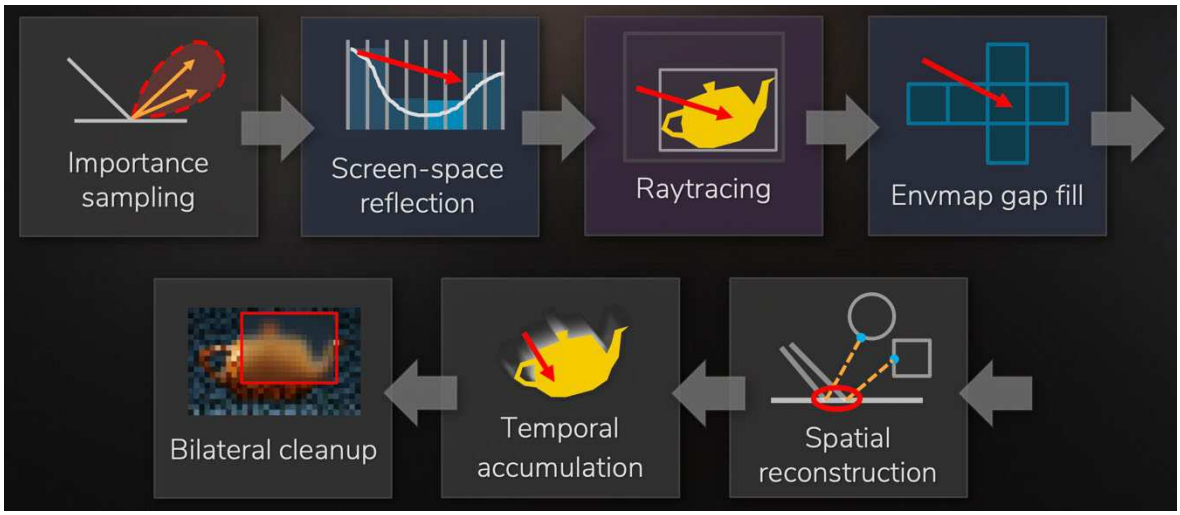
Faza iscrtavanja	**Skupina metoda**
Odgođeno sjenčanje (engl. *Deferred Shading*)	Rasterizacijske metode
Sjene	Rasterizacijske metode ili porodica algoritama praćenja zrake
Izravno osvjetljenje (engl. *Direct Lightning*)	Procesori za sjenčanje za računanje

Refleksije	Porodica algoritama praćenja zrake ili procesori za sjenčanje za računanje
Globalno osvjetljenje (engl. *Global Illumination*)	Porodica algoritama praćenja zrake
Ambijentalno zasjenjenje (engl. *Ambient Occlusion*)	Porodica algoritama praćenja zrake ili procesori za sjenčanje za računanje
Prozirnost i poluprozirnost (engl. *Transparency and translucency*)	Porodica algoritama praćenja zrake
Faza naknadne obrade (engl. *Post processing*)	Procesori za sjenčanje za računanje

Iz tablice 4.3.1.1 je lako vidljivo kako se za proces odgođenog sjenčanja koriste isključivo rasterizacijske metode. Ovoj fazi iscrtavanja pripada i primarno određivanje vidljivosti. Rasterizacijske metode izrazito efikasno odrađuju posao određivanja vidljivosti te zbog toga nema smisla koristiti računalno skuplje metode poput onih iz porodice algoritama praćenja zrake kako bi se dobili rezultate iste razine kvalitete. Također, iz tablice 4.3.1.1 je vidljivo kako je generiranje sjena jedini proces uz proces odgođenog sjenčanja koji nudi opciju korištenja rasterizacijskih metoda. Svi ostali procesi se oslanjaju ili na algoritme iz porodice algoritama praćenja zrake ili procesore za sjenčanje za računanje. U nastavku će se ukratko razmotriti implementacije određenih učinaka. Ove implementacije nisu univerzalne, no nude uvid u način razmišljanja, kao i u određene standardne prakse prilikom osmišljavanja i implementacije sustava iscrtavanja temeljenog na hibridnom cjevovodu za iscrtavanje.

Refleksije

Na slici 4.3.1.2 prikazana je shema procesa pomoću kojeg se generiraju refleksije u projektu *Pica Pica.*



Slika 4.3.1.2: *Shema procesa simulacije refleksija u projektu Pica Pica*

Efektivni budžet zraka za simuliranje fenomenologije refleksija je ½ zrake po slikovnom elementu, od čega ¼ zrake otpada na zraku refleksije, dok ¼ otpada na zraku sjena. Drugim riječima, za grupu od 2x2 slikovna elementa prati se jedna zraka refleksije i jedna zraka sjena. Reflektirana zraka se odabire prema principu važnosti, odnosno prema maksimumu BDRF funkcije, koji je detaljnije opisan u drugom poglavlju. Sirovi rezultat algoritma praćenja zrake u ovom procesu efektivno je na 4 puta manjoj rezoluciji u odnosu na rezoluciju konačnog slikovnog okvira. S druge strane, čak i da je izlaz algoritma praćenja zrake jednake rezolucije kao i konačni slikovni okvir, izlaz bi sadržavao velike količine šuma, kao što je i slučaj s navedenim, 4 puta manjim izlazom. Zbog toga veliki dio procesa simulacije refleksija otpada na proces uklanjanja šuma o kojem će više riječi biti kasnije. Ovime se dolazi do izrazito važnog i gotovo univerzalnog koncepta prilikom izrade sustava za iscrtavanje baziranog na porodici algoritama praćenja zrake. Gotovo nikada se učinci ne generiraju konvergencijom *Monte-Carlo* metode integracije, odnosno s velikim brojem uzoraka po slikovnom elementu, već se učinci generiraju s jednom ili manje od jednim uzorkom po slikovnom elementu za pojedini učinak, te se na temelju takvog rezultata, koji je regularno manje rezolucije od ciljane i pun šuma, procesom uklanjanja šuma generira konačni rezultat simulacije fizikalne fenomenologije.

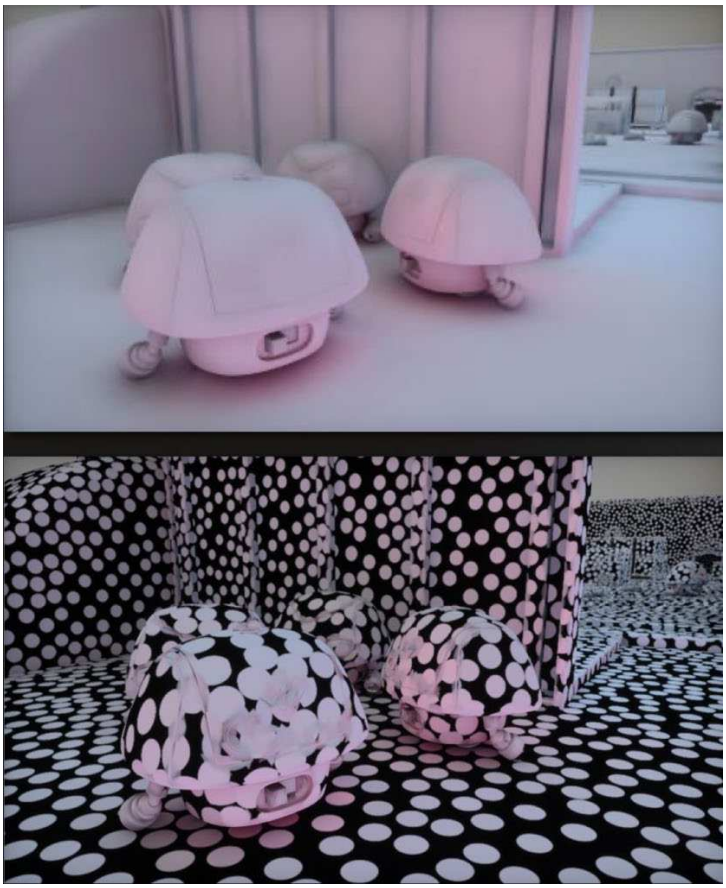
Meke sjene

Meke sjene se u projektu *Pica Pica* generiraju korištenjem jedne zrake po slikovnom elementu [4]. Korištenjem samo jedne zrake se dobiva rezultat u kojemu je vidljiv

šum, te je stoga potrebno nad rezultatom izvršiti proces uklanjanja šuma. Uzorkovanje smjera zrake sjena se izvodi uniformnim odabirom smjera iz stošca koji određuje širinu mekog dijela sjene [4].

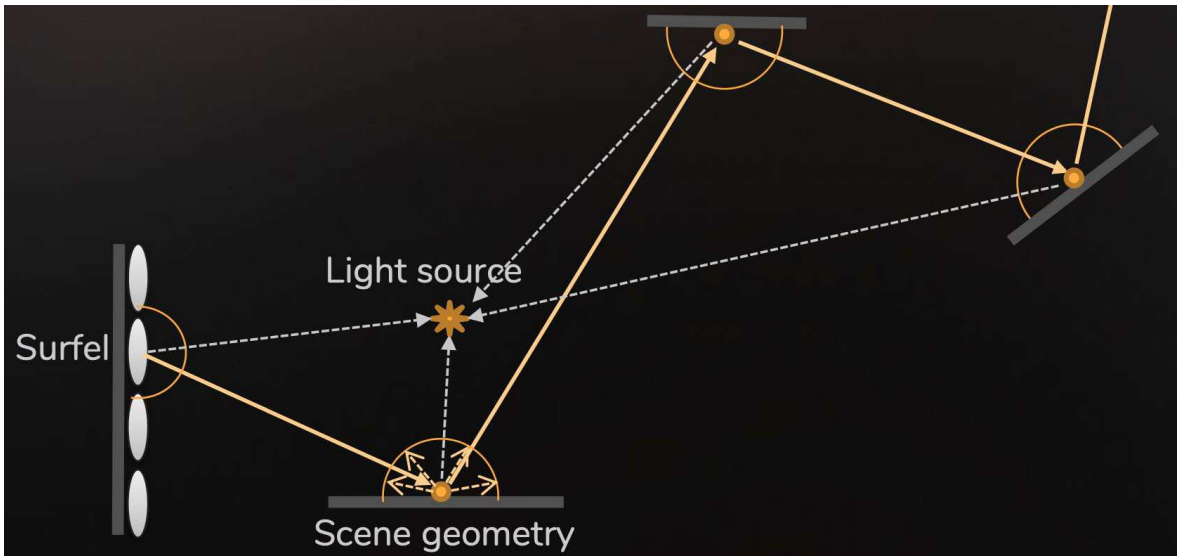
Difuzno globalno osvjetljenje

Prilikom implementacije efekta difuznog globalnog osvjetljenja želja razvojnog tima *SEED* bila je razviti metodu simulacije ovog učinka koja neće zahtijevati prethodni izračun određenih operacija, koja neće zahtijevati parametrizaciju, koja će podržavati i statične i dinamične scene, te koja će imati mogućnost poboljšanja rezultata kroz vrijeme [4]. Budžet zraka je limitiran na ukupno 250 tisuća zraka po jednom slikovnom okviru [4]. Ovo je izrazito ograničen budžet koji efektivno daje manje od jednog uzoraka, odnosno zrake, po slikovnom elementu. Tim *SEED* se odlučio za izradu sustava koji će učinak globalnog difuznog osvjetljenja generirati pomoću jedinica površine (engl. *Surfel*) u globalnom koordinatnom sustavu. Ove jedinice površine se ponašaju kao izvori svjetlosti u cjevovodu s odgođenim sjenčanjem [4]. Jedinice površine su raspoređene u ćelije uniformne mreže pomoću koje je podijeljena virtualna scena, te se za sjenčanje određenog slikovnog elementa koriste jedinice površine one ćelije kojoj slikovni element pripada. Prilikom iscrtavanja prvog slikovnog okvira, broj jedinica površine je ograničen, dok se prilikom iscrtavanja svakog sljedećeg slikovnog okvira stvara nova skupina jedinica površine koji nastoje popuniti prazan prostor koji je ostao primjenom svih prijašnjih jedinica površine [4]. Na slici 4.3.1.3 moguće je vidjeti normalan izgled virtualnog okruženja, kao i izgled pripadnih jedinica površine [4].



Slika 4.3.1.3: *Potpuni izgled virtualnog okruženja (gore) i izgled pripadajućih jedinica površine (dolje)*

Izračunavanje vrijednosti koju će jedinica površine odašiljati kao izvor svijetlosti provodi se pomoću algoritma praćenja puta s direktnim vezama prema izvorima svijetlosti i ostalim bitnim površinama, s time da se za svaku jedinicu površine uzima samo jedna zraka prilikom iscrtavanja jednog slikovnog okvira. Slika 4.3.1.4 prikazuje općenitu shemu izračuna vrijednosti osvjetljenja jedinice površine [4]. Rezultati se akumuliraju kroz vrijeme iscrtavanjem sve većeg broja slikovnih okvira. Prilikom izračunavanja vrijednosti osvjetljenja za jedinicu površine moguće je provesti uzorkovanje jedinice površine iz prošlih slikovnih okvira u točki presjeka zrake i geometrije kako bi se smanjilo potrebno vrijeme za izračun vrijednosti osvjetljenja. Pošto je jedan od zahtjeva podrška za dinamičke, odnosno animirane scene, *Monte-Carlo* metoda integracije je neprimjerena te se tim *SEED* odlučio za vrstu eksponencijalnog prosjeka s pomakom (engl. *Exponential Moving Average*) koji ne konvergira garantirano kao *Monte-Carlo* metoda, no ne mora resetirati rezultat prilikom pomaka objekata ili kamere u virtualnom okruženju [4].



Slika 4.3.1.4: *Izračun vrijednosti osvjetljenja za zadanu jedinicu površine algoritmom praćenja puta*

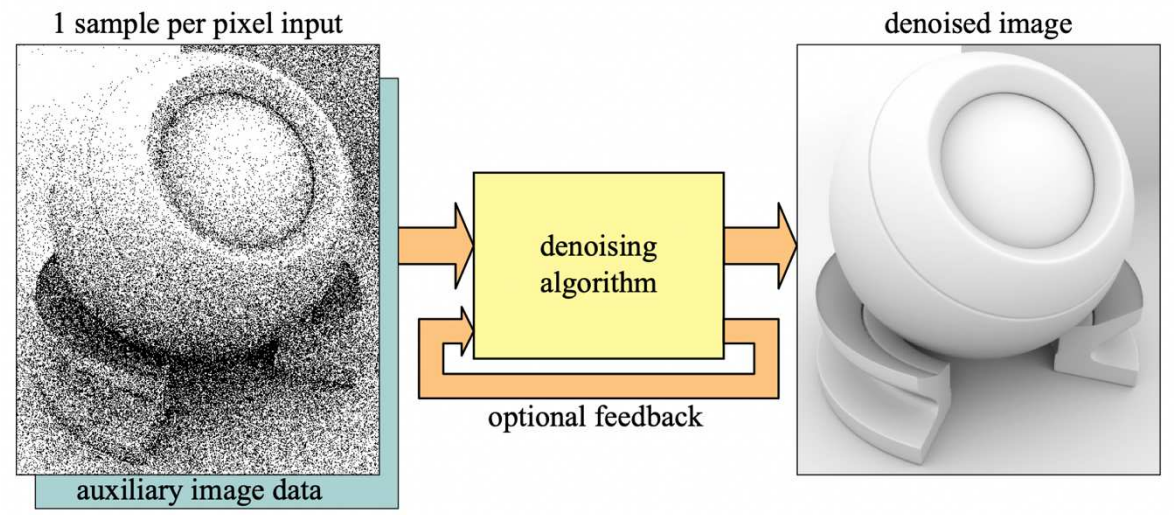
4.4. Proces uklanjanja šuma

Proces uklanjanja šuma (engl. *Denoising*) izrazito je važan proces prilikom korištenja porodice algoritama praćenja zrake za iscrtavanje slikovnih okvira u interaktivnim sustavima. Kao što je implicitno izneseno tokom ovog poglavlja, sustavi za iscrtavanje u interaktivnim sustavima si rijetko mogu priuštiti veliki broj zraka, odnosno uzoraka po slikovnom elementu za zadani učinak. Štoviše, sustav za iscrtavanje u projektu *Pica Pica* ima budžet od otprilike 2.25 zraka po slikovnom elementu za sve učinke koji se generiraju pomoću porodice algoritama praćenja zrake [4]. Podsjetimo kako se za generiranje velike većine učinaka koji simuliraju fizikalnu fenomenologiju, poput ambijentalnog zasjenjenja ili globalnog difuznog osvjetljenja, koriste stohastički procesi koji se oslanjaju na *Monte-Carlo* metodu integracije. Kao što je ranije objašnjeno u sklopu drugog poglavlja, *Monte-Carlo* metoda integracije se, u svom osnovnom obliku, oslanja na nasumično uzorkovanje funkcije pod integralom kako bi dala aproksimaciju vrijednosti integrala. Što je veći broj uzoraka, metoda daje točniju procjenu vrijednosti integrala. Kao što je također objašnjeno u drugom poglavlju, moguće je provoditi uzorkovanje po važnosti kako bi se ubrzala konvergencija *Monte-Carlo* metode. No, u općem slučaju, potreban je određen broj uzoraka kako bi pogreška u procjeni bila manja od neke zadovoljavajuće vrijednosti. Ova pogreška u procijenjenoj vrijednosti integrala na

slikovnim okvirima koji se generiraju stohastičkim inačicama algoritama iz porodice algoritama praćenja zrake očituje se kao zrnatost odnosno šum. Na slici 4.4.1 lijeva slika je slika u kojoj je vidljiva određena količina šuma. Ovakva slika je nastala simuliranjem učinka ambijentalnog zasjenjenja pomoću jednog uzorka po slikovnom elementu [1]. Na slici 4.4.1 desna slika predstavlja rezultat uklanjanja šuma na temelju lijeve slike. Moguće je primijetiti kako je pomoću procesa uklanjanja šuma moguće dobiti rezultat visoke kvalitete na temelju relativno malog broja uzoraka po slikovnom elementu. Ovo je upravo i motivacija za korištenje ovakvih procesa prilikom korištenja porodice algoritama praćenja zrake za iscrtavanje u interaktivnim sustavima. Proces uklanjanja šuma na elementarnoj razini može se shvatiti kao proces koji boju slikovnog elementa određuje kao težinski prosjek slikovnih elemenata koji ga okružuju [1]. Ovaj težinski prosjek možemo izraziti sljedećom formulom [1]:

$$d_p = \frac{1}{n}\sum_{q \in N} c_q w(p, q) \tag{11}$$

U formuli (11) d_p je filtrirana vrijednost slikovnog elementa p. N je skup slikovnih elemenata na temelju kojih se računa težinski prosjek koji sadrži i trenutni slikovni element p. Ovaj skup slikovnih elemenata uobičajeno tvori kvadrat oko slikovnog elementa p. Funkcija w je težinska funkcija, dok je c_q nefiltrirana vrijednost slikovnog elementa q. Težinska funkcija w može koristiti informacije iz prijašnjeg slikovnog okvira. Ovo je proizvoljna povratna veza koja je prikazana na slici 4.4.1. Težinska funkcija, kao i cijeli proces uklanjanja šuma u općenitom slučaju, također može koristiti i proizvoljne pomoćne informacije poput normala.



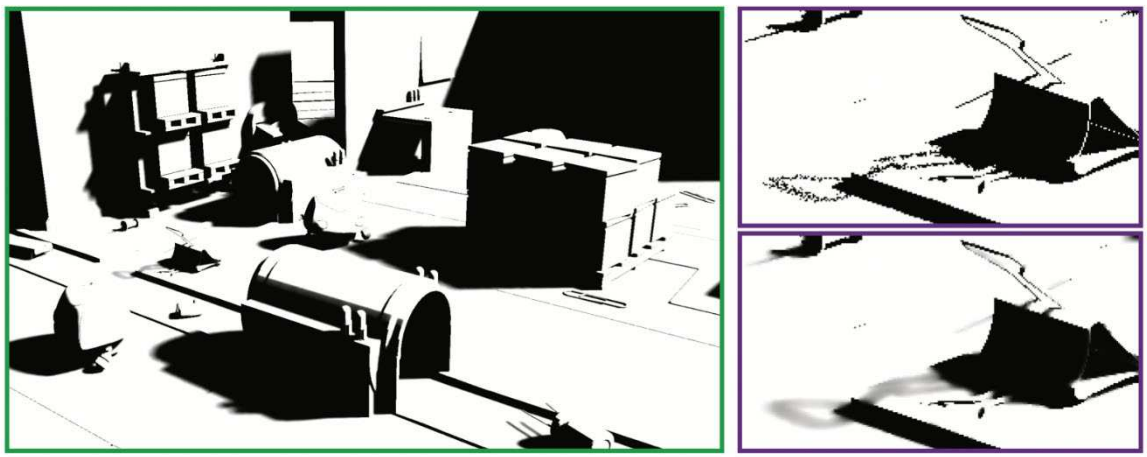
Slika 4.4.1: *Shema procesa uklanjanja šuma s prikazom generiranja učinka ambijentalnog zasjenjenja – učinak generiran s jednom zrakom po slikovnom elementu (slika lijevo) i učinak nakon procesa uklanjanja šuma (slika desno)*

Kao što je već napomenuto, proces uklanjanja šuma iznimno je važna komponenta u omogućavanju korištenja porodice algoritama praćenja zrake u interaktivnim sustavima. Zbog toga su razvijene razne metode i filteri za uklanjanje šuma, kao i razne dobre prakse za njihovo korištenje. Primjeri dobrih praksi su sljedeći [1]:

- Odvojiti izračune direktnog i indirektnog osvjetljenja te ih zasebno provući kroz filtere za uklanjanje šuma
- Odvojiti vrijednosti osvjetljenja od teksture te kroz filter za uklanjanje šuma provući samo komponentu osvjetljenja
- Generirati G-spremnike tako da su bez šuma te ih koristiti kao izvor pomoćnih informacija
- Koristiti metode vremenske akumulacije (engl. *Temporal Accumulation*) kako bi se povećao efektivan broj uzoraka po slikovnom elementu
- Za svaki učinak i za svaku njegovu komponentu koristiti prikladnu metodu, odnosno filter za uklanjanje šuma

Proces uklanjanja šuma se često naziva i filtriranjem te je pojedina metoda uobičajeno opisana pomoću filtera. Primjer izrazito popularne metode uklanjanja šuma je filter SVGF (engl. *Spatio-Temporal Variance-Guided Filtering*). Ova metoda kombinira vremensku akumulaciju i prostorni filter zamućivanja u više prolazaka (engl. *Spatial multi-pass blur*) kako bi uklonila šum [1]. Veličina jezgre (engl. *Kernel*) ovisi o varijanci u podacima na način da je jezgra veća ako je varijanca u podacima

velika [1]. SVGF filter varijancu računa inkrementalno kako dolaze novi uzroci, no u određenim slučajevima ova vrijednost varijance nije korektna te je tada moguće koristiti prostorni izračun varijance. Ova metoda je razvijena za filtriranje šuma na slikovnim okvirima iscrtanim algoritmom praćenja puta koristeći G-spremnike za određivanje primarne vidljivosti i jednu zraku u prvoj i drugoj točki presjeka zajedno s pripadnim zrakama sjena. Ovu metodu koristio je razvojni tim *SEED* za filtriranje mekih sjena u projektu *Pica Pica*. Slika 4.4.2 prikazuje primjenu SVGF filtra prilikom generiranja mekih sjena u projektu *Pica Pica*[1].



Slika 4.4.2: *Primjena SVGF filtra prilikom generiranja mekih sjena – nefiltrirane meke sjene (desno gore) i filtrirane meke sjene (desno dolje)*

Razvojni tim *SEED* je u sklopu svog projekta *Pica Pica* također predstavio nesvakidašnju metodu uklanjanja šuma prilikom generiranja refleksija. Proces uklanjanja šuma prilikom generiranja refleksija može se razložiti na sljedeće faze [4]:

1. Prostorna rekonstrukcija (engl. *Spatial Reconstruction*) i prostorna rekonstrukcija rezolucije (engl. *Spatial Upsampling*)
2. Vremenska akumulacija (engl. *Temporal Accumulation*)
3. Dvostrano čišćenje (engl. *Bilateral Cleanup*) pomoću dvostranih filtera (engl. *Bilateral Filter*)
4. Faza vremenskog procesa uklanjanja aliasa (engl. *Temporal Anti-Aliasing, TAA*)

Slika 4.4.3 prikazuje izgled slikovnog okvira u različitim fazama procesa uklanjanja šuma [1]. Prostorna rekonstrukcija i prostorna rekonstrukcija rezolucije odvijaju se

istovremeno. Ova rekonstrukcija je slična onoj koja se koristi za stohastičku inačicu generiranja refleksija u prostoru ekrana SSR (engl. *Screen-Space reflections*) [4]. Faza dvostranog čišćenja pomoću dvostranih filtera je jednostavniji dvostrani filter u odnosu na onaj koji se koristi pri prostornoj rekonstrukciji te se koristi samo ako je varijanca u podacima nakon prve dvije faze i dalje velika. Ovaj jednostavniji dvostrani filter nastoji smanjiti varijancu unošenjem zamućenja (engl. *Blur*).

Slika 4.4.3: *Izgled dijela slikovnog okvira u različitim fazama proces uklanjanja šuma – izlaz algoritma praćenja zrake (dolje lijevo), varijanca prije faze dvostranog čišćenja (dolje sredina) i konačni izlaz (dolje desno)*

Ovo su samo neke od mogućih metoda i filtera koji se mogu koristiti za uklanjanje šuma. Iz različitih primjera, poput projekta *Pica Pica,* moguće je vidjeti kako jedan filter ne mora nužno biti najbolje rješenje za sve vrste učinaka, te je poželjno za svaki učinak odabrati prikladnu metodu ili filter. Prilikom predstavljanja RTX platforme inženjeri tvrtke *Nvidia* veliki su naglasak stavili na primjenu dubokih neuronskih mreža (engl. *Deep Neural Networks*) u svrhu uklanjanja šuma. DLSS

(engl. *Deep Learning Super Sampling*) primjer je jedne takve duboke neuronske mreže. Primarna svrha ove duboke neuronske mreže je dobivanje konačnog prikaza virtualnog okruženja određene rezolucije na temelju slikovnog okvira manje rezolucije ili kao zamjenu za druge metode uklanjanja aliasa poput metode TAA [4]. DLSS mreža je učena na podacima koji su generirani pomoću metode naduzorkovanja (engl. *Super-Sampling*) na temelju 64 uzorka po slikovnom elementu. No, zbog prirode mreže, koja u svojoj suštini obrađuje slike, zasigurno je moguće mrežu iskoristiti za uklanjanje šuma prilikom generiranja pojedinih učinaka.

5. Arhitektura sustava DXR za primjenu familije algoritama praćenja zrake

Sučelje DXR (engl. *DirectX Raytracing*) je dodatak aplikacijskom programskom sučelju (engl. *Application Programming Interface, API*) *DirectX 12* tvrtke *Microsoft*. *DirectX 12* je sučelje prema grafičkom sklopovlju koje je primarno namijenjeno za iscrtavanje virtualnih okruženja u stvarnom vremenu. Sučelje DXR služi kao proširenje sučelja pomoću kojeg je moguće cijeli proces iscrtavanja, ili samo njegov dio, izvoditi pomoću familije algoritama praćenja zrake. Sučelje DXR je strukturirao na način da je moguće kombinirati klasične rasterizacijske metode i familiju algoritama praćenja zrake kako bi se iscrtala određena virtualna scena. Važnost mogućnosti kombiniranje rasterizacijskih metoda i familije algoritama praćenja zrake istaknuta je u prethodnom poglavlju. U nastavku ćemo razmotriti općenitu strukturu sustava DXR, nove vrste procesora za sjenčanje koje sustav DXR implementira, kao i korisnicima dostupne programske akceleracijske strukture koje sustav koristi kako bi ubrzalo iscrtavanje pomoću familije algoritama praćenja zrake.

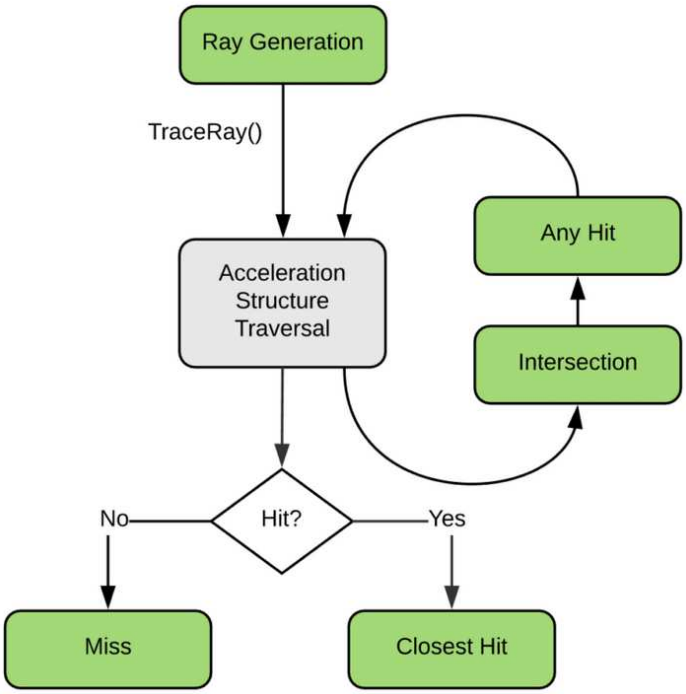
5.1. Procesori za sjenčanje

U sklopu sučelja DXR u aplikacijsko programsko sučelje *DirectX 12* dodane su nove vrste procesora za sjenčanje koje su primarno namijenjene za efikasno izvođenje algoritama iz familije algoritama praćenja zrake. [1] Nove vrste procesora za sjenčanje su sljedeće:

- Procesor za sjenčanje za stvaranje zraka (engl. *Ray Generation Shader*)
- Procesor za sjenčanje najbližeg presjeka (engl. *Closest Hit Shader*)
- Procesor za sjenčanje promašaja (engl. *Miss Shader*)
- Procesor za sjenčanje bilokakvog presjeka (engl. *Any Hit Shader*)
- Procesor za sjenčanje za određivanje presjeka (engl. *Intersection Shader*)

Struktura novih procesora za sjenčanje koji su predstavljeni u sklopu sučelja DXR, kao i njihove zadaće, nisu specifične samo za sučelje DXR. Ovakva struktura procesora za sjenčanje je relativno standardna za sva grafička aplikacijska programska sučelja koja nastoje implementirati familiju algoritama praćenja zrake

za iscrtavanje virtualnih okruženja u stvarnom vremenu. Razlog tomu se može pronaći u tome što je korištenje familije algoritama praćenja zrake za iscrtavanje virtualnih okruženja u stvarnom vremenu izrazito ovisno o sklopovskim akceleracijskim strukturama koje je tvrtka *Nvidia* prva integrirala u vlastito komercijalno grafičko sklopovlje u sklopu svoje RTX platforme. Tvrtka *Nvidia* je također razvila aplikacijsko programsko sučelje *OptiX* koje je još prije razvoja sklopovskih akceleracijskih struktura zamišljeno kao sučelje za efikasno izvođenje familije algoritama praćenja zrake pomoću grafičkog sklopovlja u svrhu industrijske vizualizacije. [6] Ovakvu strukturu procesora za sjenčanje moguće je pronaći i u sučelju *OptiX* te je upravo struktura procesora za sjenčanje sučelja *OptiX* prihvaćena kao svojevrstan standard prilikom izrade aplikacijskih korisničkih sučelja za iscrtavanje virtualnih okruženja pomoću familije algoritama praćenja zrake. [6] Na slici 5.1.1 vidljiva je shema provođenja postupka praćenja zrake u sklopu *OptiX* platforme korištenjem ranije navedenih procesora za sjenčanje. Izvođenje postupka praćenja zrake u sklopu DXR platforme je ekvivalentno onom prikazanom na shemi na slici 5.1.1.



Slika 5.1.1: *Dijagram koji prikazuje tok izvođenja algoritma praćenja zrake i njegovih inačica u kontekstu sučelja OptiX*

Na dijagramu na slici 5.1.1 jasno je vidljivo kako proces praćenja jedne zrake počinje stvaranjem zraka. Nakon što se zraka stvori, prati se njezin put u virtualnom

okruženju pomoću prostorne akceleracijske strukture. Kada pronađemo presjek s geometrijom, moramo odrediti je li to najbliži mogući valjani presjek ili je presjek kojeg želimo zanemariti na temelju nekog kriterija. Također je moguća situacija u kojoj zraka ne sječe ni jedan element virtualne scene. Ova konceptualna shema izvođenja algoritma praćenja zrake vjerno prati postupak opisan u sklopu prvog poglavlja. U nastavku ćemo razmotriti zadaće svakog od navedenih procesora za sjenčanje koji su sastavni dio sučelja DXR.

5.1.1. Procesor za sjenčanje za stvaranje zraka

Procesor za sjenčanje za stvaranje zraka predstavlja ulaznu točku za izvođenje algoritma praćenja zrake i njegovih inačica. [1] Ovaj procesor za sjenčanje se na grafičkom procesoru pokreće na sličan način kao i procesori za sjenčanje za računanje, odnosno pokreće se za određenu mrežu slikovnih elemenata. Ovakav način pokretanja je smislen zbog toga što je zadaća ovog procesora za sjenčanje stvaranje zraka kamere za određenu skupinu slikovnih elemenata. [1] Nakon generiranja zraka, ovaj procesor za sjenčanje poziva funkciju *TraceRay()* čime se pokreće proces praćenja zrake kroz virtualno okruženje. Zrake koje stvara ovaj procesor za sjenčanje se još zbirno nazivaju standardnim zrakama (engl. *Standard Rays*). Standardne zrake koriste najopćenitije postupke prilikom praćenja njihovog puta kroz virtualno okruženje te uobičajeno imaju uz sebe vezan teret pomoću kojeg se razmjenjuju informacije između raznih procesora za sjenčanje.

5.1.2. Procesor za sjenčanje najbližeg presjeka

Procesor za sjenčanje najbližeg presjeka je procesor za sjenčanje koji se izvršava u trenutku kada je pronađen najbliži presjek zrake i određene geometrije u virtualnom okruženju. [1] Unutar ovog procesora za sjenčanje implementira se proces sjenčanja točke presjeka. Proces sjenčanja može sadržavati određivanje je li točka presjeka u sjeni, određivanje fizikalnih fenomenologija poput refleksija ili refrakcija i slično. Zbog toga što se određeni dijelovi sjenčanja, poput testiranja je li točka presjeka u sjeni ili refleksija, izvršavaju rekurzivno algoritmom praćenja zrake, ovaj procesor za sjenčanje može generirati nove zrake i pratiti njihov put u virtualnom okruženju pomoću funkcije *TraceRay()* te na temelju tih rezultata završiti proces sjenčanja inicijalne točke presjeka.

5.1.3. Procesor za sjenčanje promašaja

Procesor za sjenčanje promašaja je procesor za sjenčanje koji se izvršava u onom trenutku kada nije moguće pronaći presjek zrake koju se trenutno prati i geometrije koja tvori virtualno okruženje. [1] Svrha ovog procesora za sjenčanje je pridjeljivanje određene vrijednosti sjajnosti teretu zrake. Vrijednost sjajnosti može odgovarati nekoj fiksnoj boji pozadine ili nekoj fiksnoj boji neba, no može se dobiti i uzorkovanjem teksture neba ili mape okoliša (engl. *Environment Map*).

5.1.4. Procesor za sjenčanje bilokakvog presjeka

Procesor za sjenčanje bilokakvog presjeka je procesor za sjenčanje koji se primarno koristi kada u virtualnom okruženju postoji prozirna geometrija, kao i geometrija na koju se primjenjuje tekstura na način da se tekstura testira na prozirne slikovne elemente. [1] Idealan primjer za korištenje ovog procesora za sjenčanje je vegetacija poput lišća na drveću. Lišće na drveću se rijetko u potpunosti modelira geometrijom. Za prikazivanje lišća na drveću uobičajeno se koristi nekoliko većih poligona na koje se primjenjuje tekstura koja sadrži prikaz lišća. Ovakva tekstura uobičajeno sadrži veliki broj prozirnih slikovnih elemenata. Ako zadana zraka siječe poligon ove vrste, procesor za sjenčanje bilokakvog presjeka može odrediti nalazi li se u točki presjeka proziran ili neproziran slikovni element teksture. Ako je slikovni element neproziran, moguće je odrediti sjenčanje u točki presjeka. S druge strane, ako je slikovni element proziran, možemo odbaciti točku presjeka i nastaviti s praćenjem zrake. Ova povratna veza vidljiva je na slici 5.1.1. [1] Ovaj procesor za sjenčanje se može implementirati za standardne zrake, ali i za posebne vrste zraka poput zrake sjena. Važno je napomenuti kako je poželjno svesti broj izvršavanja ovog procesora za sjenčanje na minimum.

5.1.5. Procesor za sjenčanje za određivanje presjeka

Procesor za sjenčanje za određivanje presjeka je procesor za sjenčanje pomoću kojeg je moguće implementirati procedure određivanja presjeka s nestandardnim oblicima geometrije poput geometrije generirane pomoću fraktala ili analitičkih površina, poput primjerice sfera. [1] Ovaj procesor za sjenčanje se izvršava u onom trenutku kada postoji presjek zrake i obujmice u prostornoj akceleracijskoj strukturi.

Važno je napomenuti kako su svi navedeni procesori za sjenčanje zamišljeni kao potpora izvođenju familije algoritama praćenja zrake. No, to ne znači da se ovi procesori za sjenčanje ne mogu koristiti za izvođenje drugih algoritama. Dodatak DXR je izrazito mlad dodatak aplikacijskom programskom sučelju *DirectX 12* te preostaje vidjeti kakve će sve kreativne primjene za ove procesore za sjenčanje pronaći razni razvojni timovi i programeri u budućnosti. Uz navedene procesore za sjenčanje, dodatak DXR korisniku nudi pristup prostornim akceleracijskim strukturama koje koristi za ubrzavanje postupka određivanja presjeka zrake i geometrije u virtualnom okruženju. [1] Potpuna prostorna akceleracijska struktura nije vidljiva korisniku sučelja, no poznato je kako se radi o nekoj inačici hijerarhije obujmica BVH. U nastavku ćemo pomnije razmotriti dijelove prostorne akceleracijske strukture koje su vidljive korisniku sučelja:

- Akceleracijska struktura visoke razine (engl. *Top Level Acceleration Structure, TLAS*)
- Akceleracijska struktura niske razine (engl. *Bottom Level Acceleration Structure, BLAS*)

5.2. Akceleracijske strukture niske razine – BLAS

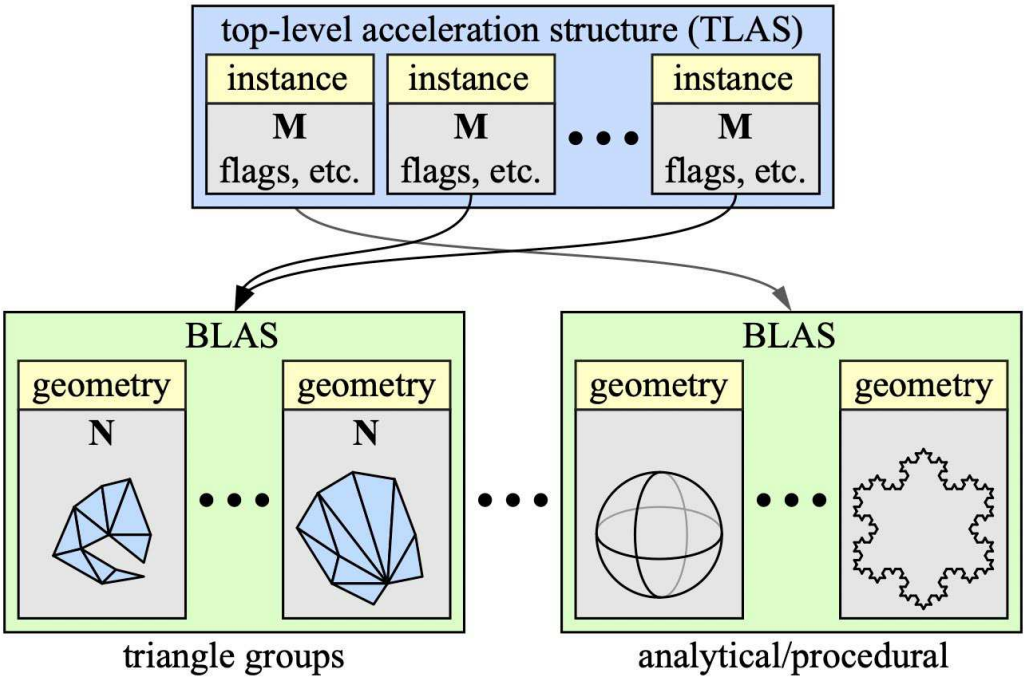
Akceleracijska struktura niske razine BLAS namijenjena je pohranjivanju pojedinačnih geometrijskih elemenata od kojih je sačinjeno određeno virtualno okruženje. [1] Ovi geometrijski elementi mogu biti ili nakupine trokuta ili analitičke geometrijske površine, poput sfere, ili pak proceduralno generirana geometrija. Analitičke površine i proceduralno generirana geometrija moraju u sklopu akceleracijske strukture niske razine imati implementiran pripadni procesor za sjenčanje za određivanje presjeka kako bi se specificirao postupak određivanja presjeka zrake i navedenih vrsta geometrijskih elemenata. [1] Svaki geometrijski element u akceleracijskoj strukturi niske razine ima pridruženu inicijalnu transformacijsku matricu dimenzija 3x4 koja se na geometrijski element primjenjuje samo jednom prije konstruiranja pozadinske prostorne akceleracijske strukture. Ova matrica označena je slovom *N* na slici 5.1. Pošto je općeniti slučaj takav da svaki geometrijski element ima vlastitu akceleracijsku strukturu niske razine,

transformacijska matrica se može promatrati na način da je vezana uz akceleracijsku strukturu niske razine. [5] Moguća je situacija u kojoj jedna akceleracijska struktura niske razine može sadržavati podatke o više geometrijskih elemenata virtualnog okruženja, prilikom čega svaki geometrijski element ima pridruženu vlastitu transformacijsku matricu. [5] Motivacija za ovu mogućnost leži u generalnom pravilu koje govori o tome kako je bolje imati manji broj akceleracijskih struktura niske razine.

5.3. Akceleracijske strukture visoke razine – TLAS

Akceleracijska struktura visoke razine TLAS se sastoji od instanci geometrijskih elemenata koji se nalaze u akceleracijskim strukturama niske razine, odnosno ove instance možemo promatrati kao instance akceleracijskih struktura niske razine. Za instancu geometrijskog elementa u akceleracijskoj strukturi visoke razine se bilježi pokazivač na akceleracijsku strukturu niske razine. [1] Za instancu geometrijskog elementa se također bilježi transformacijska matrica dimenzija 3x4. Ova transformacijska matrica se koristi kako bi se bazični geometrijski element koji se nalazi u akceleracijskoj strukturi niske razine mogao pozicionirati u virtualnom okruženju na drugačiji način u odnosu na druge instance istog bazičnog geometrijskog elementa. Ova matrica je matrica označena slovom *M* na slici 5.1.

Općeniti izgled prostornih akceleracijskih struktura niske i visoke razine prikazan je na slici 5.1 [1]. Na slici 5.1 se također mogu uočiti navedeni odnosi između akceleracijskih struktura niske i visoke razine, kao i podaci vezani uz pojedinu vrstu akceleracijske strukture.



Slika 5.1: *Ilustracija akceleracijskih struktura visoke (TLAS) i niske (BLAS) razine, njihovih odnosa i pripadnih podataka*

Kako bismo bolje razumjeli odnose između akceleracijskih struktura visoke i niske razine, razmotrimo sljedeći primjer virtualne scene. Virtualna scena se sastoji od sljedećih objekata: plavog automobila, crvenog automobila, bijele kuće, zelene kuće, drveta i ceste. U ovoj sceni imamo ukupno 6 objekata koji tvore virtualno okruženje. Pretpostavka je da se auti i kuće razlikuju samo u boji odnosno materijalu, te da su u virtualnom okruženju predstavljeni istom poligonalnom mrežom. Za ovo virtualno okruženje potrebno je konstruirati 4 akceleracijske strukture niske razine: jednu za objekt auta, jednu za objekt kuće i po jednu za objekte drveta i ceste. U akceleracijskoj strukturi visoke razine imamo 6 zapisa o instancama akceleracijskih struktura niske razine, po jednu za svaki objekt. Pomoću iste akceleracijske strukture niske razine za objekt automobila, stvaramo dvije instance u akceleracijskoj strukturi visoke razine koje predstavljaju plavi automobil i crveni automobil. Ekvivalentno se tvore i objekti dviju kuća. Prostorna akceleracijska struktura koja se sastoji od dviju manjih prostornih akceleracijskih struktura tvori takozvano stablo u dvije razine [5] ili hijerarhijsku strukturu u dvije razine [1]. Ovakav oblik prostorne akceleracijske strukture pogodan je za virtualne scene u kojima se objekti animiraju. Animacije koje je moguće opisati elementarnim transformacijama nad cijelim geometrijskim elementom mogu se izvoditi pomoću transformacijskih

matrica vezanih uz pojedinu instancu geometrijskog elementa u akceleracijskoj strukturi visoke razine. Ove matrice je moguće mijenjati između slikovnih okvira bez potrebe za promjenama u akceleracijskim strukturama niske razine. S druge strane, animacije koje modificiraju vrhove pojedinog geometrijskog elementa predstavljaju problem zbog toga što zahtijevaju promjene u akceleracijskoj strukturi niske razine. Ove promjene zahtijevaju ponovnu izgradnju pozadinske prostorne akceleracijske strukture što može biti izrazito računalno zahtjevno. Zbog toga su ovakve promjene neodržive ako bi se pozadinska akceleracijska struktura morala ponovno izgraditi u potpunosti prilikom iscrtavanja svakog slikovnog okvira. [1] Ako su promjene u vrhovima malene, moguće je samo modificirati postojeću pozadinsku akceleracijsku strukturu. Ovo naravno narušava kvalitetu pozadinske prostorne akceleracijske strukture, te također zahtjeva određeno vrijeme, no sveukupno narušavanje kvalitete se može kontrolirati na način da se pozadinska prostorna akceleracijska struktura ponovno izgradi nakon određenog broja slikovnih okvira, dok se u međuvremenu samo modificira. Problem s ovakvim prostornim akceleracijskim strukturama u dvije razine je taj što je njihova kvaliteta generalno lošija od drugih vrsta prostornih akceleracijskih struktura. [1] Primjer nepoželjne situacije je onaj u kojem se mnoge instance akceleracijskih struktura niske razine preklapaju te stoga zraka koja prolazi ovim dijelom prostora mora prolaziti kroz sve instance akceleracijskih struktura niske razine koje se nalaze i preklapaju u tom dijelu prostora virtualnog okruženja.

6. Porodica algoritama praćenja zrake u kontekstu alata *Unity*

Alat *Unity* koristi porodicu algoritama praćenja zrake za implementiranje raznih učinaka u sklopu cjevovoda za iscrtavanje visoke kvalitete (engl. *High-Definition Render Pipeline, HDRP*). HDRP cjevovod je programibilan cjevovod za iscrtavanje koji je primarno namijenjen za korištenje u projektima koji zahtijevaju visoku kvalitetu generiranih slikovnih okvira u kontekstu fotorealističnog iscrtavanja [9]. Učinci za čiju implementaciju alat *Unity* koristi porodicu algoritama praćenje zrake su učinci za koje ne postoji prikladna rasterizacijska metoda koja je istovremeno kvalitetna i dinamička. Ovi učinci su detaljnije opisani u trećem poglavlju. Alat *Unity* porodicu algoritama praćenja zrake implementira pomoću sučelja DXR. Pošto je sučelje DXR dio aplikacijskog programskog sučelja DirectX 12, projekt mora koristiti DirectX 12 kao svoj primarni API za iscrtavanje [9]. Važno je napomenuti kako je implementacija učinaka pomoću porodice algoritama praćenja zrake još uvijek u stadiju razvoja te su moguće određene promjene do konačne inačice implementacije i integracije u alat *Unity* [9].

6.1. Podržana fenomenologija

Kao što je ranije spomenuto, alat *Unity* koristi porodicu algoritama praćenja zrake za generiranje učinaka koje nije moguće dinamički generirati pomoću rasterizacijskih metoda ili za generiranje učinaka za koje odgovarajuće rasterizacijske metode ne mogu ostvariti visoku kvalitetu prikaza u dinamičkim situacijama. U alatu *Unity* moguće je pomoću porodice algoritama praćenja zrake generirati sljedeće učinke:

- Ambijentalno zasjenjenje (engl. *Ambient Occlusion*)
- Refleksije (engl. *Reflections*)
- Sjene (engl. *Shadows*) – uključuje i kontaktne sjene (engl. *Contact Shadows*)
- Indirektno globalno osvjetljenje (engl. *Global Illumination*)
- Raspršenje ispod površine (engl. *Subsurface Scattering*)

Osim konkretnih učinaka, alat *Unity* implementira i algoritam praćenja puta kao unificirani proces za iscrtavanje virtualnih okruženja. Alat *Unity* također implementira i rekurzivno iscrtavanje (engl. *Recursive Rendering*). Ako se za iscrtavanje određenog objekta u virtualnoj sceni koristi rekurzivno iscrtavanje, taj objekt stvara reflektirane i lomljene zrake koje se prate dalje u virtualnoj sceni. Ovaj način iscrtavanje predstavlja implementaciju porodice algoritama praćenja zrake u njezinom prirodnom obliku kako je to opisano u prvom poglavlju ovog diplomskog rada. Rekurzivno iscrtavanje koristi vlastiti odvojeni prolaz kroz grafički protočni sustav. Sve druge implementacije ove porodice algoritama ukomponirane su u određeni oblik hibridnog cjevovoda za iscrtavanje. Zbog toga nije moguće miješanje rekurzivnog iscrtavanja i drugih učinaka implementiranih porodicom algoritama praćenja zrake [9].

6.2. Izrada pokazne scene

Aplikacija izrađena u sklopu ovog diplomskog rada bazira se na pokaznoj sceni koja je sastavni dio predloška HDRP projekta. Ova pokazna scena zamišljena je kao ulazna točka za izradu projekata pomoću HDRP cjevovoda. U sklopu scene moguće je pronaći prirodne izvore svjetlosti s vrijednostima koje odgovaraju fizikalnoj stvarnosti, izvore umjetne svjetlosti s pripadajućim pokaznim vrijednostima, razne fizikalno utemeljene materijale i slično. U scenu su također kvalitetno postavljene reflektivne sonde (engl. *Reflection Probes*). Scenom se nastoje demonstrirati razni alati koji su sastavni dio HDRP cjevovoda, a koji omogućuju izradu virtualnih okruženja koji nalikuju na fizikalnu stvarnost. Čak i uz posjedovanje ovakvih alata, izrada kvalitetne scene koja vjerno reflektira fizikalnu stvarnost izrazito je zahtjevan posao. Za dobivanje kvalitetnih rezultata potrebne su godine iskustva. Cijela odjeljenja razvojnih timova posvećena su izradi virtualnih okruženja te njihovom osvjetljavanju. Ideja iza izrade aplikacije u sklopu ovog diplomskog rada nije pokazati na koji način izraditi virtualnu scenu, niti kako ju osvijetliti da nalikuje na fizikalnu stvarnost, nego demonstrirati na koji način porodica algoritama praćenja zrake poboljšava kvalitetu dobivenog prikaza te koje su njezine prednosti nad ekvivalentnim standardnim rasterizacijskim metodama. Također cilj izrade aplikacije je istraživanje mogućnosti primjene ove porodice algoritama na interaktivne sustave. Stoga pokazna HDRP scena služi kao odlično igralište za demonstraciju

snage porodice algoritama praćenja zrake. Alat *Unity* implementira porodicu algoritama praćenja zrake pomoću sustava volumena (engl. *Volume Framework*) [9]. Volumen u alatu *Unity* određuje dio virtualnog okruženja na kojeg se primjenjuju određeni učinci ili pravila. Volumen je komponenta koja se može pridružiti objektu u virtualnom okruženju. Svaki volumen može biti ili globalan ili lokalan. Globalni volumeni se primjenjuju uvijek bez obzira na to gdje se nalazi kamera u virtualnom okruženju. S druge strane, lokalni volumeni se primjenjuju prilikom iscrtavanja slikovnog okvira kamere koja se nalazi unutar zadanog lokalnog volumena. Alat *Unity* nudi mogućnost kombiniranja više volumena na različite načine kako bi se dobio krajnji izgled slikovnog okvira zadane kamere. Kako bi se omogućilo generiranje određenog učinka porodicom algoritama praćenja zrake, potrebno je u željeni volumen dodati prikladno volumno nadjačanje (engl. *Volume Override*). Lista volumnih nadjačanja tvori profil volumena (engl. *Volume Profile*). Aplikacija izrađena u sklopu ovog diplomskog rada demonstrira implementacije sljedećih učinaka pomoću porodice algoritama praćenja zrake:

- Ambijentalno zasjenjenje
- Refleksije
- Sjene
- (Indirektno difuzno) Globalno osvjetljenje

Aplikacija nudi izbornik pomoću kojeg je moguće mijenjati metode generiranja nekog od navedenih učinaka. Odabirom pojedine metode za određeni učinak mijenjaju se postavke pripadajućeg volumnog nadjačanja u profilu globalnog volumena. U nastavku će se razmotriti koje su ponuđene metode generiranja svakog od navedenih učinaka, te pomoću kojeg se volumnog nadjačanja metode implementiraju.

6.2.1. Ambijentalno zasjenjenje

Učinak ambijentalnog zasjenjenja implementira se pomoću volumnog nadjačanja *Ambient Occlusion*. Navedeno volumno nadjačanje nudi opcije generiranja učinka rasterizacijskom metodom baziranom na podacima u trenutnom vidnom polju kamere ili pomoću porodice algoritama praćenja zrake. Za oba načina ponuđene su različite razine kvalitete generiranja učinka. Za učinak ambijentalnog zasjenjenja aplikacija putem izbornika nudi sljedeće metode generiranja:

- Ambijentalno zasjenjenje u trenutnom vidnom polju kamere (engl. *Screen-Space Ambient Occlusion, SSAO*) – oznaka SSAO
    - Ogovara opciji *High* u izborniku *Quality*
- Ambijentalno zasjenjenje implementirano porodicom algoritama praćenja zrake niske kvalitete – oznaka RTX LOW
    - Odgovara opciji *Medium* u izborniku *Quality*
- Ambijentalno zasjenjenje implementirano porodicom algoritama praćenja zrake srednje kvalitete – oznaka RTX MID
    - Ogovara opciji *High* u izborniku *Quality*
- Ambijentalno zasjenjenje implementirano porodicom algoritama praćenja zrake visoke kvalitete – oznaka RTX HIGH
    - Ogovara opciji *Custom* u izborniku *Quality* s vrijednošću parametra *Ray Length* postavljenoj na 5

Mijenjanjem ponuđenih metoda generiranja učinka moguće je odrediti razliku u razini kvalitete generiranja učinka pomoću standardne rasterizacijske metode temeljene na informacijama dostupnim u trenutnom vidnom polju kamere i kvalitete generiranja učinka pomoću porodice algoritama praćenja zrake. Također, moguće je odrediti koliko je generiranje učinka računalno zahtjevnije kada se za generiranje koristi porodica algoritama praćenja zrake u odnosu na slučaj kada se za generiranje koristi ponuđena rasterizacijska metoda. Također, moguće je odrediti razliku u kvaliteti prikaza između tri ponuđene konfiguracije implementacije porodicom algoritama praćenja zrake, kao i razliku u zahtjevnosti u kontekstu računalnih resursa potrebnih za izvođenje pojedine konfiguracije.

6.2.2. Refleksije

Učinak refleksija implementira se pomoću volumnog nadjačanja *Screen Space Reflections*. Navedeno volumno nadjačanje nudi opcije generiranja učinka refleksija pomoću dvije inačice algoritma baziranog na informacijama u trenutnom vidom polju kamere, kao i pomoću porodice algoritama praćenja zrake. Za generiranje učinka pomoću porodice algoritama praćenja zrake ponuđene su dvije generalne konfiguracije, od kojih je jedna orijentirana na performanse, dok je druga orijentirana na kvalitetu simulacije učinka. Kada je navedeno volumno nadjačanje isključeno, za generiranje učinka refleksija koriste se postojeće reflektivne sonde. Za simulaciju

fenomenologije refleksija, koje uključuju i zrcalne i sjajne refleksije, ponuđene su sljedeće metode generiranja učinka:

- Refleksije generirane korištenjem neosvježenih reflektivnih sondi koje se nalaze u sceni – oznaka REFLECTION PROBES
    - Isključuje volumno nadjačanje *Screen Space Reflections*
- Refleksije generirane rasterizacijskom metodom u trenutnom vidom polju kamere (engl. *Screen-Space Reflections, SSR*) – oznaka SSR
    - Koristi se opcija *PBR Accumulation* u izborniku *Algorithm*
- Refleksije generirane porodicom algoritama praćenja zrake niske razine kvalitete – oznaka RTX LOW
    - Odgovora generalnoj konfiguraciji *Performance* i opciji *Medium* u izborniku *Quality*
- Refleksije generirane porodicom algoritama praćenja zrake srednje razine kvalitete – oznaka RTX MID
    - Odgovora generalnoj konfiguraciji *Performance* i opciji *High* u izborniku *Quality*
- Refleksije generirane porodicom algoritama praćenja zrake visoke razine kvalitete – oznaka RTX HIGH
    - Ogovara generalnoj konfiguraciji *Quality* s parametrom *Bounce Count* postavljenim na vrijednost 3 i s parametrom *Sample Count* postavljenim na vrijednost 1

Mijenjanjem ponuđenih metoda generiranja učinka moguće je odrediti razlike u kvaliteti prikaza fenomenologije refleksija kada se refleksije generiraju standardnim rasterizacijskim metodama poput reflektivnih sondi ili metodama temeljenim na informacijama dostupnim u trenutnom vidom polju kamere i slučaja kada se refleksije generiraju pomoću porodice algoritama praćenja zrake. Također, pomoću zrcala postavljenih na samom početku scene moguće je uočiti nedostatke generiranja učinka pomoću reflektivnih sondi u slučaju kada se pojave novi objekti, koji su u ovom slučaju navedena zrcala, kao i općenite nedostatke druge dostupne rasterizacijske metode. Također, korištenjem različitih konfiguracija implementacije porodicom algoritama praćenja zrake moguće je odrediti razlike u kvaliteti generiranih rezultata unutar navedenih konfiguracija. Također je moguće odrediti koliko su implementacije pomoću porodice algoritama praćenja zrake računalno

zahtjevnije od implementacija rasterizacijskim metodama, kao i koliko raste računalna zahtjevnost promjenom parametara implementacije porodicom algoritama praćenja zrake kako bi se dobili kvalitetniji rezultati.

6.2.3. Indirektno difuzno globalno osvjetljenje

Učinak indirektnog difuznog globalnog osvjetljenja implementira se pomoću volumnog nadjačanja *Screen Space Global Illumination*. Navedeno nadjačanje nudi opciju generiranja učinka rasterizacijskom metodom temeljenom na podacima dostupnim u trenutnom vidnom polju kamere, kao i opciju generiranja porodicom algoritama praćenja zrake. Kada se koristi opcija generiranja pomoću porodice algoritama praćenja zrake, nude se dvije generalne konfiguracije, od kojih je jedna orijentirana na performanse, dok je druga orijentirana na kvalitetu generiranog učinka. Kada je navedeno volumno nadjačanje isključeno, za simulaciju učinka koriste se mape svjetla i svjetlosne sonde. Za simulaciju indirektnog difuznog globalnog osvjetljenja ponuđene su sljedeće metode generiranja navedenog učinka:

- Korištenje već postojeći, neosvježenih mapa svjetla (engl. *Lightmaps*) i svjetlosnih sonda (engl. *Light Probes*) – oznaka BAKED
    - Isključuje volumno nadjačanje *Screen Space Global Illumination*
- Generiranje učinka pomoću rasterizacijske metode u trenutnom vidom polju kamere – oznaka SSGI
    - Odgovara opciji *High* u izborniku *Quality*
- Generiranje indirektnog globalnog osvjetljenja pomoću porodice algoritama praćenja zrake niske razine kvalitete – oznaka RTX LOW
    - Odgovora generalnoj konfiguraciji *Performance* i opciji *Medium* u izborniku *Quality*
- Generiranje indirektnog globalnog osvjetljenja pomoću porodice algoritama praćenja zrake srednje razine kvalitete – oznaka RTX MID
    - Odgovora generalnoj konfiguraciji *Performance* i opciji *High* u izborniku *Quality*
- Generiranje indirektnog globalnog osvjetljenja pomoću porodice algoritama praćenja zrake visoke razine kvalitete – oznaka RTX HIGH

- Ogovara generalnoj konfiguraciji *Quality* s parametrom *Bounce Count* postavljenim na vrijednost 3 i s parametrom *Sample Count* postavljenim na vrijednost 1

Promjenom navedenih metoda generiranja učinka moguće je odrediti razlike u kvaliteti generiranih rezultata. Važna napomena kod promatranja rezultata indirektnog difuznog globalnog osvjetljenja je ta kako se prilikom korištenja opcija koje učinak generiraju ponuđenom rasterizacijskom metodom ili porodicom algoritama praćenja zrake ne koriste postojeće mape svjetla i svjetlosne sonde. Kada se u alatu *Unity* koristi volumno nadjačanje za generiranje učinka indirektnog difuznog globalnog osvjetljenja ili pomoću ponuđene rasterizacijske metode, ili pomoću porodice algoritama praćenja zrake, alat *Unity* zanemaruje sve postojeće mape svjetla i svjetlosne sonde te učinak u potpunosti dinamički računa odabranom metodom. Rezultat dobiven dinamičkim metodama svakako neće biti jednako kvalitetan kao onaj dobiven mapom svjetla, pod uvjetom da nije došlo do dinamičkih promjena, zbog toga što se za stvaranje mapa svjetla koriste algoritmi iz porodice algoritama praćenja zrake s konfiguracijama koje su previše računalno zahtjevne za izvođenje u stvarnom vremenu. No, dinamičke metode reagiraju na promjene u virtualnom okruženju prirodno te je stoga važno promatrati koliko su rezultati dinamičkih metoda bliski rezultatima dobivenim mapama svjetla i svjetlosnim sondama. Važnost sposobnosti odgovora na dinamičke promjene u virtualnom okruženju demonstrira ljubičasti izvor svjetlosti u trećoj prostoriji koji se pomiče. Također, mijenjanjem metoda generiranja učinka moguće je odrediti odnos računalne kompleksnosti između rasterizacijskih metoda i metoda baziranih na porodici algoritama praćenja zrake, te između raznih konfiguracija metoda koje generiraju učinak pomoću porodice algoritama praćenja zrake.

6.2.4. Sjene

Za razliku od ostalih učinaka, učinak sjena se ne implementira pomoću volumnog nadjačanja, nego je potrebno podesiti način generiranja sjena za svaki izvor svjetlosti zasebno. Izvori svjetlosti nude opcije za generiranje sjena pomoću mapi sjena (engl. *Shadow Maps*), pomoću metode bazirane na podacima dostupnim u trenutnom vidnom polju kamere ili pomoću porodice algoritama praćenje zrake. Prilikom korištenja porodice algoritama praćenja zrake, usmjereni izvori svjetlosti

(engl. *Directional Light*) imaju opciju za generiranje obojenih sjena. Za simulaciju sjena izbornik u aplikaciji nudi sljedeće metode za generiranje i simulaciju fenomenologije pojave sjena:

- Generiranje sjena postupkom mapa sjena (engl. *Shadow Maps*) – oznaka PCSS
- Generiranje sjena konfiguracijom algoritma iz porodice algoritama praćenja zrake orijentiranom na performanse – oznaka RTX Performance
    - Onemogućene obojane sjene i parametar *Sample Count* postavljen na 2
- Generiranje sjena konfiguracijom algoritma iz porodice algoritama praćenja zrake orijentiranom na kvalitetu prikaza učinka – oznaka RTX Quality
    - Omogućena obojane sjene i parametar *Sample Count* postavljen na 8

Mijenjanjem metode generiranja sjena moguće je uočiti razlike u kvaliteti rezultata dobivenih pomoću metode PCSS s filterom srednje kvalitete i rezultata dobivenih porodicom algoritama praćenja zrake. Također je moguće odrediti i razlike u zahtjevnosti u kontekstu potrebnih računalnih resursa i razine izvođenja između navedenih metoda generiranja, kao i različitih konfiguracija generiranja pomoću porodice algoritama praćenja zrake.

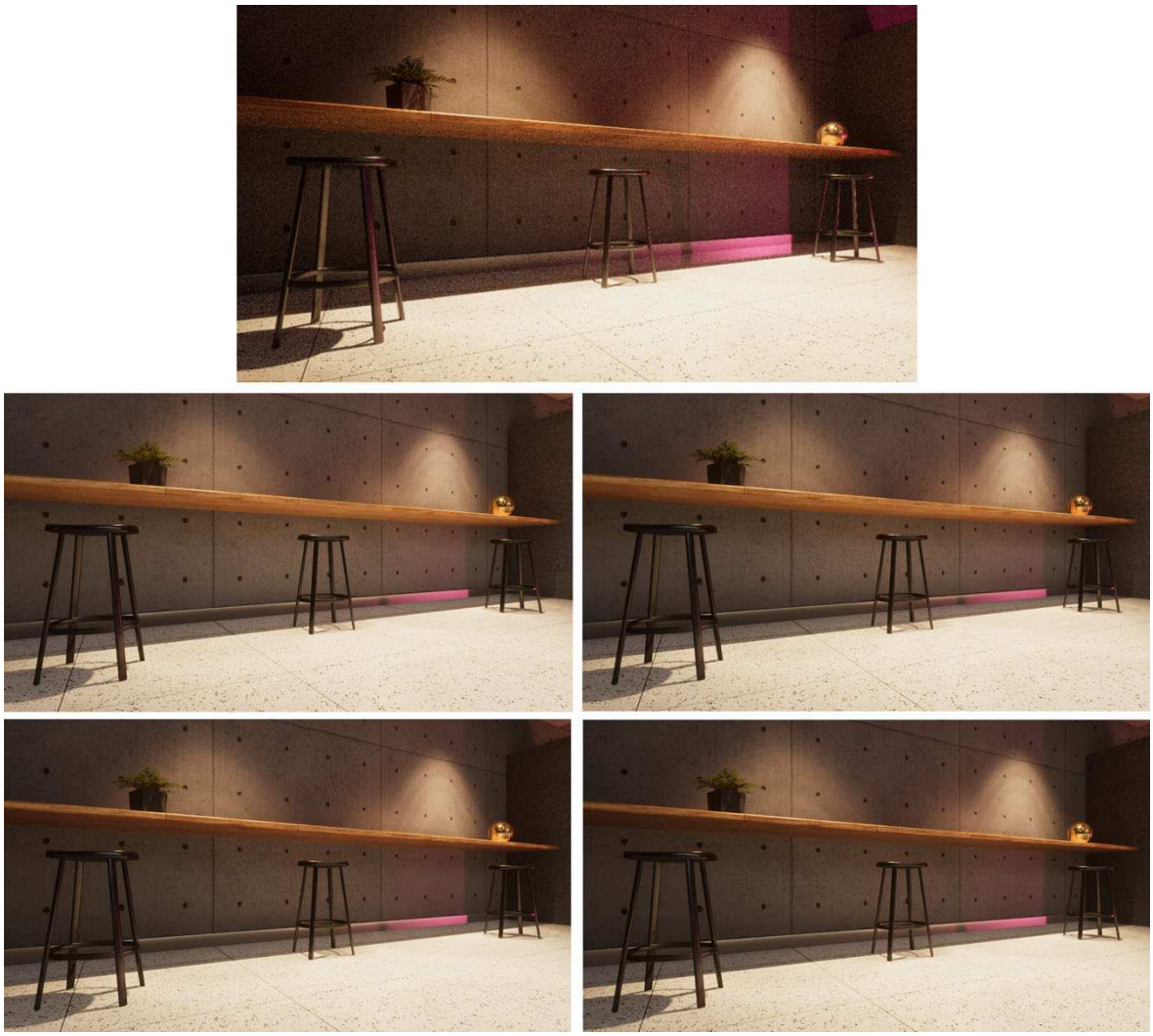
Aplikacija izrađena u sklopu ovog diplomskog rada kroz izbornik nudi opciju iscrtavanja virtualnog okruženja u potpunosti pomoću algoritma praćenja puta. Ovaj postupak nije primjeren za iscrtavanje slikovnih okvira u stvarnom vremenu. S druge strane, postupak nudi unificirano rješenje problema iscrtavanja te je izrazito koristan za generiranje referentnih rezultata. U sklopu ovog diplomskog rada, ovaj postupak se koristi za generiranje referentnih rezultata na temelju kojih je moguće odrediti u kojoj je mjeri koja metoda generiranja i pripadajuća razine kvalitete odraz fizikalne stvarnosti.

7. Rezultati

Rezultate dobivene pomoću aplikacije i projekta izrađenog u sklopu ovog diplomskog rada promatrati će se i analizirati u kontekstu pojedinog učinka koji su implementirani u sklopu navedene aplikacije i projekta. Za svaki od učinaka koji će se analizirati referentni rezultat generirati će se korištenjem algoritma praćenja puta. Također, za svaki učinak biti će priložena tablica u kojoj će biti navedene razine izvođenja prilikom generiranja učinka određenom metodom. Razina izvođenja biti će izražena pomoću jedinice broja slikovnih okvira po sekundi (engl. *Frames Per Second, FPS*). Ova jedinica je obrnuto proporcionalna računalnoj zahtjevnosti pojedine metode.

7.1. Ambijentalno zasjenjenje

Na slici 7.1.1 prikazani su slikovni okviri generirani različitim ponuđenim metodama za generiranja učinka ambijentalnog zasjenjenja. Na slici 7.1.1 se također nalazi referentni rezultat dobiven korištenjem algoritma praćenja puta.



Slika 7.1.1: *Prikazi virtualnog okruženja u kojima je odabran različit način generiranja učinka ambijentalnog zasjenjenja (s lijeva na desno , od gore prema dolje): referentni rezultat dobiven korištenjem algoritma praćenja puta, učinak ambijentalnog zasjenjenja generiran metodom SSAO, rezultat dobiven konfiguracijom porodice algoritama praćenja zrake niske (RTX LOW), srednje (RTX MID) i visoke kvalitete (RTX HIGH)*

U tablici 7.1.1 navedene su razine izvođenja prilikom generiranja učinka određenom metodom.

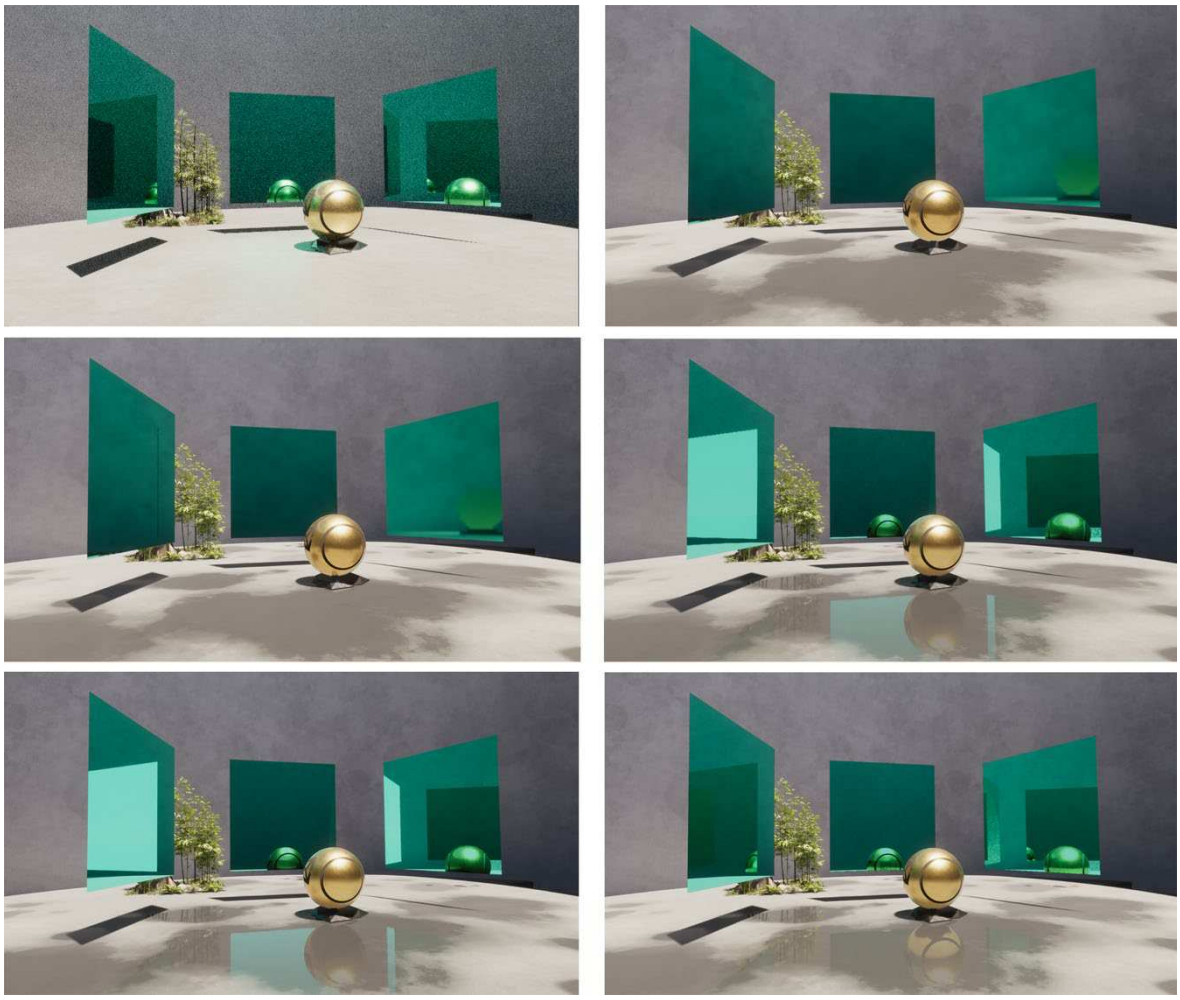
Tablica 7.1.1: *Razine izvođenja prilikom generiranja učinka ambijentalnog zasjenjenja odabranom metodom*

Metoda	SSAO	RTX LOW	RTX MID	RTX HIGH
FPS	114	78	67	41

Sa slike 7.1.1 je vidljivo kako metoda generiranja učinka RTX HIGH generira rezultat najbliži referentnom rezultatu generiranom korištenjem algoritma praćenja puta. Ova metoda je iskoristiva u interaktivnim sustavima, no utjecaj na razinu performansi je velik te je upitno koliko je korisno žrtvovati performanse u tolikoj mjeri za simulaciju relativno suptilnog učinka poput ambijentalnog zasjenjenja. Razina izvođenja RTX MID generira nešto lošiji rezultat od metode RTX HIGH s time da je pripadna razina izvođenja primjerenija kontekstu interaktivnih sustava. Metode generiranja SSAO i RTX LOW generiraju međusobno slične rezultate, s time da je metoda RTX LOW otprilike 30 posto sporija. Razlika u razini izvođenja između metoda RTX LOW i RTX MID nije velika, dok je razlika u rezultatu itekako primjetna. Na temelju dobivenih rezultata moguće je zaključiti kako metode bazirane na porodici algoritama praćenja zrake mogu generirati kvalitetnije rezultate, no razine performansi mogu biti neprikladne za interaktivne sustave.

7.2. Refleksije

Na slici 7.2.1 prikazani su slikovni okviri generirani različitim ponuđenim metodama generiranja i simulacije fenomenologije refleksija. Na slici 7.2.1 također se nalazi i referentan rezultat generiran algoritmom praćenja puta.



Slika 7.2.1: *Prikazi virtualnog okruženja u kojima je odabrana jedna od ponuđenih metoda za generiranje i simulaciju fenomenologije refleksija (s lijeva na desno, od gore prema dolje): referentni rezultat dobiven korištenjem algoritma praćenja puta, rezultat dobiven neosvježenim reflektivnim sondama, rezultat dobiven metodom SSR, rezultati dobiveni različitim konfiguracijama implementacije porodicom algoritama praćenja zrake RTX LOW, RTX MID i RTX HIGH*

Važna napomena prilikom promatranja rezultata prikazanih na slici 7.2.1 odnosi se na referentni rezultat dobiven korištenjem algoritma praćenja puta. Naime, implementacija ovog algoritma u alatu *Unity* ne podržava sustav naljepnica (engl. *Decal*). Pomoću ovog sustava implementirana je lokva vode koja je vidljiva na slikovnim okvirima na slici 7.2.1 te je upravo to razlog zašto navedena lokva vode nedostaje na referentnom rezultatu. U tablici 7.2.1 nalaze se pripadne razine izvođenja za određenu metodu generiranja fenomenologije refleksija.

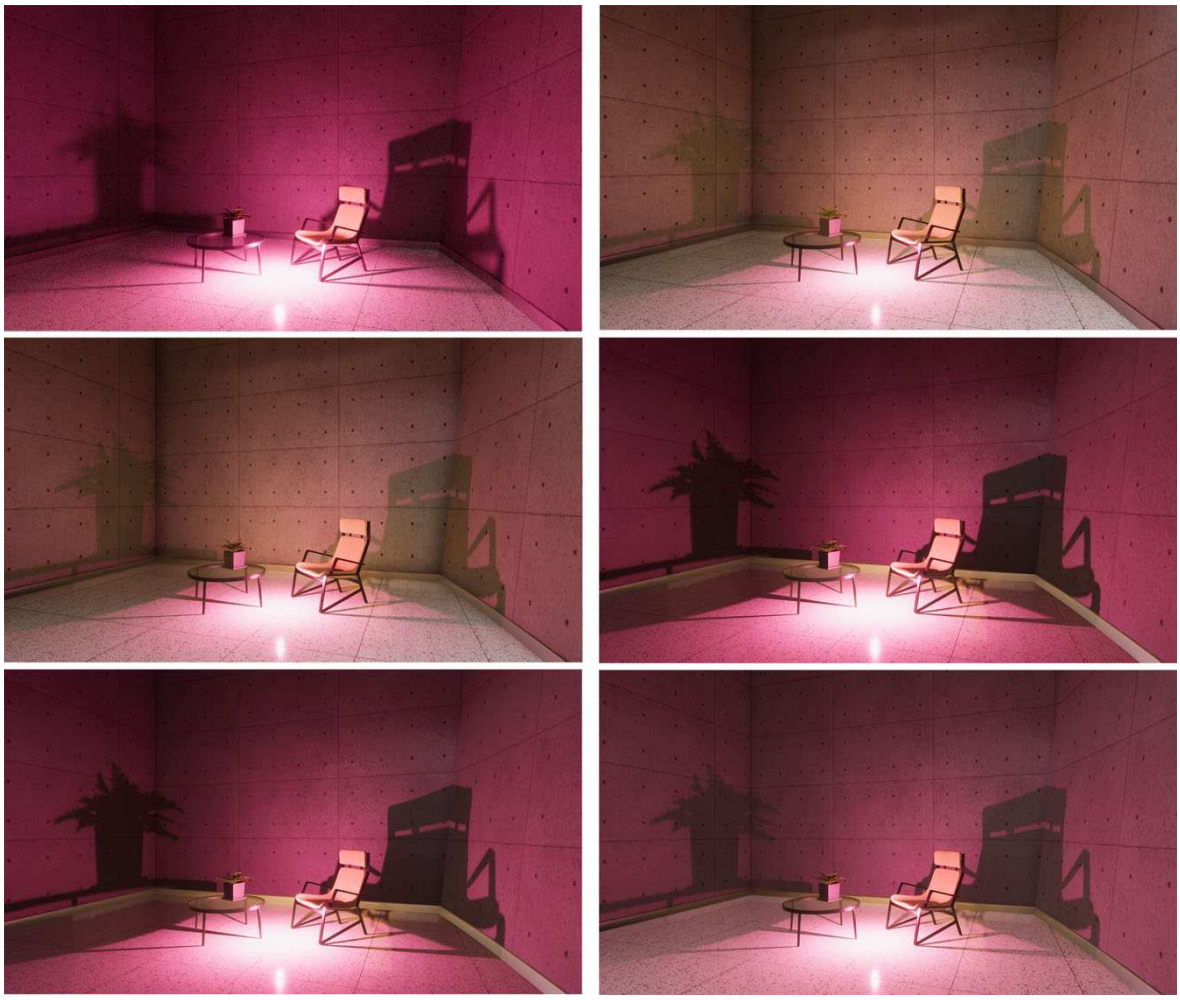
Tablica 7.2.1: *Razine izvođenja prilikom generiranja i simulacije fenomenologije refleksija odabranom metodom*

Metoda	REFLECTION PROBES	SSR	RTX LOW	RTX MID	RTX HIGH
FPS	140	116	82	27	7

Gledajući rezultate prikazane na slici 7.2.1 lako je uočiti kako metoda RTX HIGH generira rezultate koji su drastično kvalitetniji od ostalih ponuđenih metoda. Rezultat ove metode je najbliži referentnom rezultatu. No razina izvođenja je daleko ispod granice iskoristivosti u interaktivnim sustavima. Metode RTX LOW i RTX MID generiraju međusobno slične rezultate, s time da je razine izvođenja metode RTX LOW otprilike 3 puta bolja. Najveći nedostatak ovih dviju metoda proizlazi iz ograničenja broja odbijanja na 1, čime one ne mogu korektno simulirati višestruka odbijanja u prikazanim zrcalima, kao što to može metoda RTX HIGH kojoj su dopuštena tri odbijanja. S druge strane, prikazana situacija je rijetka u virtualnim okuženjima u praksi, te bi rezultat metoda RTX LOW i RTX MID sigurno bio kvalitetniji u kontekstu takvih virtualnih okruženja zbog toga što je u takvim virtualnim okruženjima obično dovoljno jedno odbijanje. S druge strane, metoda SSR daje potpuno nekorektne rezultate te čak prikazuje artefakt na lijevom ogledalu. Metoda reflektivnih sondi također pati od nemogućnosti odgovora na dinamičku pojavu novih objekata zrcala, te je refleksija u zrcalu utemeljena na niskoj rezoluciji rezultata zabilježenog u reflektivnoj sondi. No, razina izvođenja dviju navedenih rasterizacijskih metoda daleko nadilazi metode bazirane na porodici algoritama praćenja zrake.

7.3. Indirektno difuzno globalno osvjetljenje

Na slici 7.3.1 prikazani su slikovni okviri u kojima je učinak indirektnog globalnog osvjetljenja generiran različitim ponuđenim metodama. Na slici 7.3.1 također se nalazi i referentan rezultat generiran korištenjem algoritma praćenja puta.



Slika 7.3.1: *Prikazi virtualnog okruženja u kojem je učinak indirektnog difuznog globalnog osvjetljenja generiran različitim ponuđenim metodama (s lijeva na desno, od gore prema dolje): referentan rezultat dobiven korištenjem algoritma praćenja puta, rezultat dobiven korištenjem neosvježenih mapi svjetla i svjetlosnih sondi, rezultat dobiven metodom SSGI, rezultati dobiveni različitim konfiguracijama implementacije porodicom algoritama praćenja zrake RTX LOW, RTX MID i RTX HIGH*

U tablici 7.3.1 navedene su pripadne razine izvođenja za svaku od ponuđenih metoda.

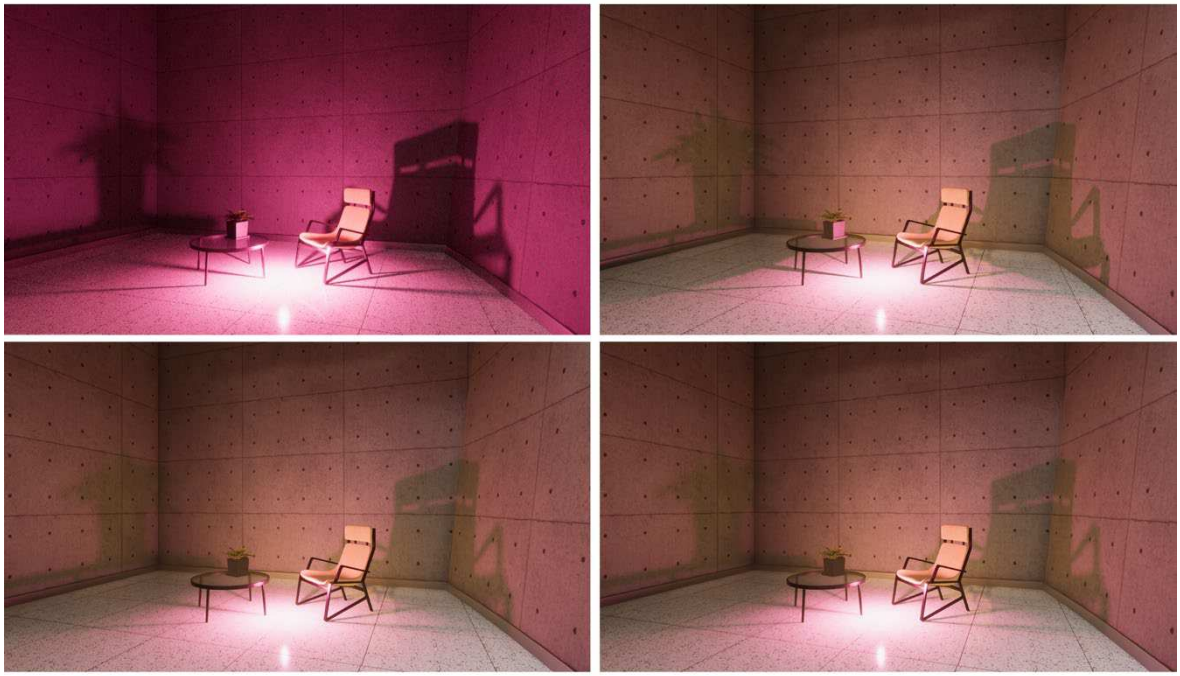
Tablica 7.3.1: *Razine izvođenja prilikom generiranja učinka indirektnog difuznog globalnog osvjetljenja odabranom metodom*

Metoda	BAKED	SSGI	RTX LOW	RTX MID	RTX HIGH
FPS	140	33	45	31	2

Sa slike 7.3.1 vidljiva je drastična razlika između rezultata dobivenih rasterizacijskim metodama i metodama baziranim na porodici algoritama praćenja zrake. Metoda koja koristi neosvježene mape svjetla i svjetlosne sonde nije u mogućnosti dinamički reagirati na novi izvor svjetlosti. Na površinama je vidljivo ponešto roze boje koju daje novi izvor svjetlosti zbog izračuna direktnog osvjetljenja. Za odražavanje pojave novog izvora svjetlosti, potrebno bi bilo osvježiti mape svjetla i svjetlosne sonde koje se nalaze u virtualnom okruženju. Svjetlosne sonde to mogu izvršiti u stvarnom vremenu, no mape svjetla bi bilo potrebno ponovno izračunati i nove rezultate zabilježiti u novim mapama svjetla. S druge strane, sve metode koje se baziraju na porodici algoritama praćenja zrake pravilno reagiraju na novi izvor svjetlosti te površine zidova ispunjavaju bojom koja odgovara boji novog izvora svjetlosti. Sa slike 7.3.1 je vidljivo kako nema veće razlike između metoda RTX LOW, RTX MID i RTX HIGH te su rezultati svih navedenih metoda veoma slični referentnom rezultatu. S druge strane, razina izvođenja metode RTX HIGH se teško može smatrati interaktivnom. Ova metoda ima veći dopušteni broj odbijanja čime je prikladnija za kompleksne interijere. Metode RTX LOW i RTX MID imaju pripadne razine izvođenja koje su prikladnije za interaktivne sustave. Zanimljivo je kako je razina izvođenja metode SSGI, koja ne daje korektne rezultate, manja od metode RTX LOW i bliska razini izvođenja metode RTX MID. Iz dobivenih rezultata je vidljivo koliko implementacije porodicom algoritama praćenja zrake povećavaju razinu sposobnosti odgovora sustava na dinamičke promjene u indirektnog difuznom globalnom osvjetljenju.

7.4. Sjene

Na slici 7.4.1 prikazani su slikovni okviri u kojima je fenomenologija sjena generirana i simulirana različitim ponuđenim metodama. Na slici 7.4.1. također se nalazi i referentni rezultat dobiven korištenjem algoritma praćenja puta.



Slika 7.4.1: *Prikaz virtualnog okruženja u kojem je fenomenologija pojave sjena generirana i simulirana različitim ponuđenim metodama (s lijeva na desno, od gore prema dolje): referentni rezultat dobiven korištenjem algoritma praćenja puta, rezultat dobiven metodom PCSS, rezultat dobiven konfiguracijom implementacije porodicom algoritama praćenja zrake orijentiranom na performanse (RTX Performance) i orijentiranom na kvalitetu (RTX Quality)*

U tablici 7.4.1 navedene su pripadne razine izvođenja prilikom korištenja određene metode.

Tablica 7.4.1: *Pripadne razine izvođenja za odabranu metodu generiranja sjena*

Metoda	PCSS	RTX Performance	RTX Quality
FPS	140	48	24

Važno je napomenuti kako metoda PCSS koristi filter srednje kvalitete. Sa slike 7.4.1 vidljivo je kako metode bazirane na porodici algoritama praćenja zrake generiraju rezultate koji su slični referentnom rezultatu i fizikalnoj stvarnosti. Sjene generirane ovim metodama su oštrije na mjestima na kojima je objekt koji baca sjenu bliži objektu na kojem se prikazuje sjena, dok su mekše na mjestima na kojima su ova dva objekta udaljenija. S druge strane, konfiguracija PCSS metode

ne prikazuje ovakav rezultat. No, razine izvođenja prilikom korištenja metoda RTX Performace i RTX Quality daleko su manje nego prilikom korištenja metode PCSS. Također razlika u kvaliteti između metoda RTX Performance i RTX Quality je izrazito mala, gotovo i neprimjetna, dok je razlika u razini izvođenja otprilike 2 puta.

Na temelju iznesenih rezultata moguće je zaključiti kako metode za simulaciju navedenih učinaka temeljene na porodici algoritama praćenja zrake u većoj ili manjoj mjeri poboljšavaju kvalitetu generiranja navedenih učinaka. No, jednako tako, razina izvođenja prilikom korištenja ovih metoda je znatno manja nego prilikom korištenja ekvivalentnih rasterizacijskih metoda.

Zaključak

Porodica algoritama praćenja zrake kroz godine razvoja područja računalne grafike postavljala se kao zlatni standard za iscrtavanje virtualnih okruženja. Ova porodica algoritama nudi intuitivno rješenje općenitog oblika jednadžbe iscrtavanja, te time vjerno simulira propagaciju svjetlosti kroz virtualno okruženje te njezinu interakciju s materijom u tom istom virtualnom okruženju. Zbog toga je u mogućnosti generirati fotorealistične slikovne okvire koji prikazuju virtualno okruženje. No, ova porodica algoritama izrazito je računalno kompleksna za izvođenje. Iz tog razloga porodica algoritama praćenja zrake primarno se koristi u sustavima za iscrtavanje koji ne zahtijevaju interaktivnost, poput onih u filmskoj industriji. S druge strane, porodica algoritama praćenja zrake uvijek je predstavljala nedostižan cilj za iscrtavanje u interaktivnim sustavima. No, razvojem grafičkog sklopovlja ovaj cilj se više ne doima nedostižnim. Razvoj grafičkog sklopovlja okrenuo se razvoju sklopovskih akceleracijskih struktura koje ubrzavaju algoritme iz porodice algoritama praćenja zrake. Ovi napredci u razvoju napokon omogućuju primjenu ove porodice algoritama na iscrtavanju u interaktivnim sustavima. Bez obzira na ovakav razvoj događaja, nije suvislo odbaciti godine razvoja algoritma rasterizacije i njemu pripadnih metoda zbog toga što taj algoritam za određene probleme, poput problema vidljivosti, daje rezultate jednake kvalitete kao i porodica algoritama praćenja zrake. Nadalje, porodica algoritama praćenja zrake je i dalje previše računalno kompleksna da bi se cijeli proces iscrtavanja izvodio samo pomoću ove porodice algoritama. Stoga su se razvojni timovi okrenuli konceptu hibridnog cjevovoda u kojemu se porodica algoritama praćenja zrake koristi za iscrtavanje učinaka za koje ne postoje u isto vrijeme kvalitetne i dinamičke rasterizacijske metode iscrtavanja. Rezultati dobiveni aplikacijom izrađenom u alatu *Unity*, čija implementacija prati koncept hibridnog cjevovoda, prikazuju drastične razlike u kvaliteti i dinamičnosti generiranih učinaka pomoću porodice algoritama praćenja zrake naspram klasičnih rasterizacijskih metoda. No, razina izvođenja je također manja. Stoga je moguće zaključiti kako je porodica algoritama praćenja zrake napravila dobar prvi korak prema primjeni u interaktivnim sustavima, no područje je još uvijek mlado te zahtjeva daljnji razvoj. Možda još uvijek nije došla budućnost obojana porodicom algoritama praćenja

zrake, no ovakva budućnost se ne čini kao nedostižan cilj kao što je to nekada bio slučaj.

Literatura

[1] Akenine-Möller T., Haines E., Hoffman N., Pesce A., Iwanicki M., Hillaire S. Real-time rendering 4th edition. 4. izdanje. Boca Raton, 2018.

[2] Haines E., Akenine-Möller T., Keller A., McGuire M., Munkberg J., Pharr M., Shirley P., Wald I., Wyman C. Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs, Unofficial, errata corrected version 1.9, 2021.

[3] Kilgariff E., Moreton H., Stam N., Bell B., *NVIDIA Turing Architecture In-Depth*, Nvidia Developer, (2018, rujan). Poveznica: https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/; pristupljeno 26. lipnja 2021.

[4] Stachowiak T., *Stochastic All The Things: Raytracing in Hybrid Real-time Rendering*, EA, (2018). Poveznica: https://www.ea.com/seed/news/seed-dd18-presentation-slides-raytracing; pristupljeno: 26. lipnja 2021.

[5] Lefrancois M. K., Gautron P., *DX12 Raytracing tutorial – Part 1*, Nvidia Developer. Poveznica: https://developer.nvidia.com/rtx/raytracing/dxr/dx12-raytracing-tutorial-part-1; pristupljeno: 26. lipnja 2021.

[6] Morley K., *How to Get Started with OptiX 7*, Nvidia Developer, (2019, studeni). Poveznica: https://developer.nvidia.com/blog/how-to-get-started-with-optix-7/; pristupljeno: 26. lipnja 2021.

[7] Hoffman N., *Physics and Math of Shading*, (2015). Poveznica: https://blog.selfshadow.com/publications/s2015-shading-course/hoffman/s2015_pbs_physics_math_slides.pdf; pristupljeno: 26. lipnja 2021.

[8] Kuri D., *GPU Path Tracing in Unity – Part 2*, Three Eyed Games, (2018, svibanj). Poveznica: http://three-eyed-games.com/2018/05/12/gpu-path-tracing-in-unity-part-2/; pristupljeno: 26. lipnja 2021.

[9] *High Definition Render Pipeline overview*, Unity, (2020). Poveznica: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@11.0/manual/index.html; pristupljeno: 26. lipnja 2021.

[10] *Unity User Manual 2021.1*, Unity, (2021, lipanj). Poveznica: https://docs.unity3d.com/2021.1/Documentation/Manual/index.html; pristupljeno: 26. lipnja 2021.

Sažetak

Algoritam praćenja zrake i praćenja puta za prikazivanje scena u stvarnom vremenu

Na početku rada razmatra se kako porodica algoritama praćenja zrake rješava nedostatke standardnog rasterizacijskog funkcijskog cjevovoda za iscrtavanje. Iznose se fizikalni i matematički modeli na kojima se temelji fizikalno utemeljeno iscrtavanje te način na koji se navedeni modeli rješavaju jednadžbom iscrtavanja i porodicom algoritama praćenja zrake. Iznosi se fizikalna fenomenologija koju je teško simulirati klasičnim rasterizacijskim metodama. Razmatra se implementacija porodice u kontekstu interaktivnih sustava pomoću koncepta hibridnog cjevovoda, sklopovskih i programskih akceleracijskih struktura. Iznosi se primjer implementacije porodice algoritama praćenja zrake u interaktivnom sustavu u kontekstu alata *Unity*. Analiziraju se rezultati dobiveni aplikacijom izrađenom u alatu *Unity* te se razmatra upotrebljivost implementacije u interaktivnim sustavima.

Ključne riječi: *praćenje zrake; praćenje puta; fizikalno utemeljeno iscrtavanje; hibridni cjevovod; Unity; HDRP*

Summary

Real-time Ray and Path Tracing

In the beginning of this paper, it is discussed how raytracing-based algorithms solve problems of the common rasterized function pipeline used for rendering purposes. Physical and mathematical models of physically based rendering are shown and how the rendering equation and raytracing-based algorithms are used to solve them. It is discussed which effects are hard to simulate with raster-based methods. Furthermore, the concept of a hybrid rendering pipeline is presented and it is discussed how this concept, along with hardware and software acceleration structures, enable raytracing-based algorithms to be used in interactive rendering systems. The implementation in the Unity engine is shown as an example. An analysis of the results retrieved form the demo application made with the Unity engine is given. It is also discussed whether these results are plausible for usage in interactive rendering systems.

Keywords: *raytracing; path tracing; physically based rendering; hybrid rendering pipeline; Unity; HDRP*

Skraćenice

DXR	*DirectX Raytracing*	Sučelje za izvođenje porodice algoritama praćenja zrake
CPU	*Central Processing Unit*	Procesor opće namjene
GPU	*Graphics Processing Unit*	Grafički procesor
ROP	*Raster Operations*	Rasterizacijske operacije
SPD	*Spectral Power Distribution*	Graf spektralne raspodjele snage
ELF	*Extremely Low Frequency*	Val ekstremno niske frekvencije
BRDF	*Bidirectional Reflectance Distribution Function*	Dvosmjerna distribucijska funkcija reflektivnosti
SBRDF	*Spatial BRDF*	Prostorna BRDF funkcija
SVBRDF	*Spatially Varying BRDF*	Prostorno varijabilna BRDF funkcija
NDF	*Normal Distribution Function*	Funkcija normalne distribucija
TAA	*Temporal Anti-Aliasing*	Proces vremenskog uklanjanja aliasa
SSAO	*Screen-Space Ambient Occlusion*	Ambijentalno zasjenjenje u vidom polju kamere
PCF	*Percentage-Closer Filtering*	Metoda filtriranja postotka bliskosti
PCSS	*Percentage-Closer Soft Shadows*	Metoda postotka bliskosti za meke sjene
SSR	*Screen-Space Reflections*	Refleksije u vidnom polju kamere
FPS	*Frames Per Second*	Mjerna jedinica slikovnih okvira po sekundi

BSP	*Binary Space Partitioning*	Binarna podjela prostora
BIH	*Bounded Interval Hierarchy*	Hijerarhija ograničenih intervala
BVH	*Bounded Volume Hierarchy*	Hijerarhija obujmica
AABB	*Axis-Aligned Bounding Boxes*	Obujmice poravnate s koordinatnim osima
SIMD	*Single Instruction, Multiple Data*	Jedna instrukcija, više podataka
RT	*Raytracing*	Postupak praćenja zrake
SM	*Streaming Multiprocessors*	Jedinica grafičkog procesora
SVGF	*Spatio-Temporal Variance-Guided Filtering*	Prostorno-vremenski filter vođen varijancom
DLSS	*Deep Learning Super Sampling*	Filter zasnovan na dubokim neuronskim mrežama
BLAS	*Bottom Level Acceleration Structure*	Akceleracijska struktura niske razine
TLAS	*Top Level Acceleration Structure*	Akceleracijska struktura visoke razine
HDRP	*High-Definition Render Pipeline*	Cjevovod za iscrtavanje visoke kvalitete

Privitak

Preuzimanje projekta i aplikacije

Projekt i aplikaciju moguće je klonirati s *GitHub* repozitorija na sljedećoj poveznici:

https://github.com/garderain/Diplomski_rad.git

Projekt je izrađen inačicom alata *Unity* 2021.1.11f1. U direktoriju se također nalazi direktorij *Diplomski rad – Aplikacija* u kojem se nalazi datoteka *Diplomski rad.exe* pomoću koje je moguće pokrenuti aplikaciju. Za pokretanje je potreban operacijski sustav *Windows* te grafička kratica iz porodice RTX.

Korištenje aplikacije

Unutar aplikacije moguće je hodati pomoću tipki WASD na tipkovnici, kao i okretati kameru pomoću miša. Pritiskom na tipku X na tipkovnici otvara se izbornik u kojem je moguće odabrati način generiranja pojedinog učinka. Iz izbornika se izlazi gumbom koji se nalazi u donjem desnom kutu. Izbornik je prikazan na slici u nastavku. Iz aplikacije se izlazi pritiskom na tipku ESC na tipkovnici.