SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SAMOSTALNA LABORATORIJSKA VJEŽBA

Algoritam praćenja zraka – Ray Tracing

Filip Anđel

U sklopu kolegija Računalna animacija

Zagreb, siječanj, 2023.

Sadržaj

Uvod

Praćenje zraka (engl. *ray tracing*) metoda je prikazivanja osvjetljenja koja se danas koristi za stvaranje visoko realističnih trodimenzionalnih slika. Ova se metoda već desetljećima koristi u filmskoj industriji, ali zbog tehničkih ograničenja sve do nedavno nije bila praktična za korištenje u interaktivnoj grafici kao što su video igre. No 2018. godine došlo je do velike promjene u svijetu grafike pojavom grafičkih kartica koje su dovoljno snažne da podržavaju praćenje zraka u stvarnom vremenu. Tema ovog seminarskog rada je pružiti uvid u princip metode praćenja zraka i njenu povijest te opisati razvoj koji je doveo do široke upotrebe ove metode.

Seminarski rad

Praćenje zraka kao ideja postoji još od kasnih 1960-ih. Arthur Appel je po prvi put uspio iskoristiti računalo i generirati osjenčanu sliku pomoću praćenja zraka 1968. godine. Praćenje zraka koristio je kako bi odredio površinu najbližu kameri u svakoj točki slike, a sekundarne zrake pratio je od izvora svjetlosti do svake osjenčane točke kako bi odredio je li točka u sjeni ili ne.

1971. Goldstein i Nagel izdaju rad „3-D Visual Simulation“, u kojem koriste praćenje zraka za stvaranje osjenčanih slika objekata tako što simuliraju proces fotografiranja unazad. Slanjem zrake kroz svaki piksel objekta, te prateći kroz koji piksel zraka prvo prođe, moguće je odrediti koji pikseli tvore vidljivu površinu objekta. Ovu tehniku danas nazivamo *ray casting*, i razlikujemo je od *ray tracing*-a jer je znatno manje zahtjevna za računanje i ne omogućava primjerice precizno oslikavanje reflekcija (ogledala), refrakcije ili prirodnog opadanja sjena.

Inženjer Turner Whitted 1979. godine producira film „The Compleat Angler“ koji je prvi primjer korištenja rekurzivnog praćenja zraka za postizanje odraza u ogledalu i refrakcije kroz prozirne objekte, te korištenje *ray tracinga* za *anti-aliasing*. Njegov je algoritam promijenio shvaćanje *renderinga* slika iz jednostavnog zadatka određivanja koje su površine vidljve u stvarni, fizički smislen sustav putovanja svijetlosti.

Svijetlosne zrake u prirodi

U prirodi, izvor svijetlosti emitira zraku svijetlosti u svim smjerovima. U prijevodu, to znači da iz njega izlazi konstantan niz fotona koji putuju niz zrake, odnosno niz istu putanju. U savršenom vakuumu ova zraka je pravac. Zraci svjetlosti mogu se dogoditi sljedeće četiri pojave:

- Apsorpcija – gubi se dio intenziteta originalne zrake
- Reflekcija – dio zrake ili cijela zraka se reflektira u drugom smjeru/smjerovima
- Refrakcija – dio zrake ulazi u prozirno tijelo i mijenja smjer, dok ostatak apsorbira

- Florescencija – tijelo apsorbira dio zrake i ponovo emitira s većom valnom duljinom u nasumičnom smjeru

Promatranjem ove četiri pojave moguće je u potpunosti odrediti kretanje zraka svijetlosti. Svaka zraka putuje dok se ne sudari s nekom tvari, a u tom trenutku odvija se apsorpcija, reflekcija, refrakcija ili florescencija. Zatim, reflektirana ili refraktirana zraka nastavlja u novom smjeru dok se ponovo ne sudari s nekim tijelom, gdje se ponovo odvija neka od navedenih pojava. U konačnici, neke od tih zraka završe u oku ljudi ili leći kamere te tako dobivamo slike.

Rekurzivni ray tracing algoritam

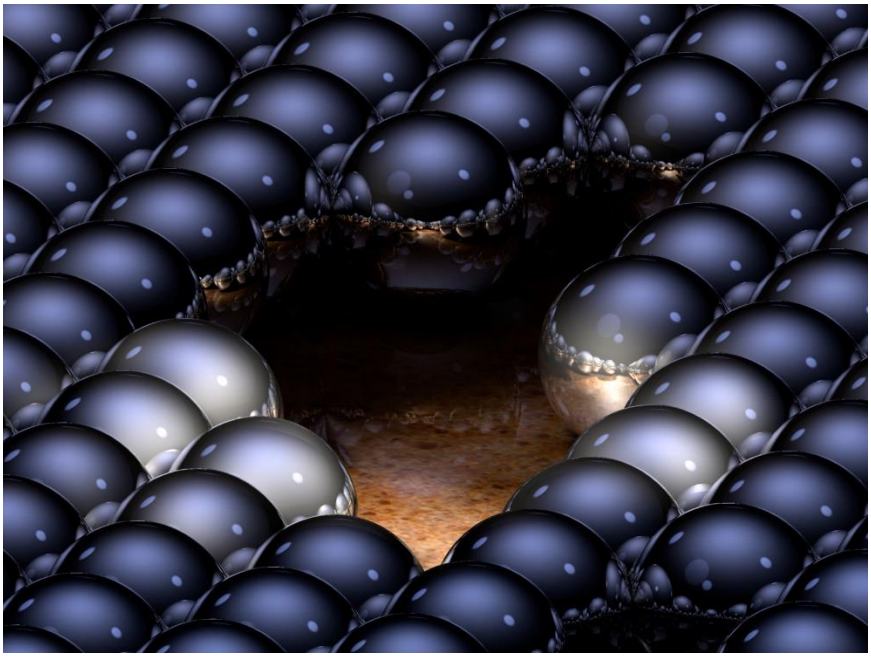
Prethodnici *ray tracing* algoritma prate zrake svjetlosti dok se ne sudare s objektom, ali boju osvjetljenja praktički simuliraju jer ne prate zrake koje se reflektiraju ili refraktiraju od objekta koji je pogođen. Kod rekurzivnog *ray tracing*a, prate se i te zrake, pa tako primjerice imamo sljedeće slučajeve:

- Reflektirana zraka odlazi od objekta i sudara se s novim objektom – taj objekt bit će vidljiv u reflekciji prvog objekta
- Refraktirana zraka radi na sličan način, te će prozirni objekti također prikazivati odraz objekata oko sebe
- „Zraka sjene“ šalje se od površina prema izvorima svjetlosti – ako se sudari s nekim objektom na putu do izvora, površina je u sjeni

Dodavanje ovih pojava povećava realističnost slika. Korištenje ovakve tehnike rezultira smislenim odrazima na reflektirajućim površinama kao što su zrcala i prozori, znatno realističnijim sjenama i općenito oku ugodnijem osvjetljenju scene. Primjerice, prije pojave *ray tracinga* zrcala su se u video igrama u potpunosti simulirala. To je moglo biti ostvareno stvaranjem jednakih objekata na „drugoj strani“ zrcala koji u potpunosti zrcale objekte sa strane s koje gledamo prema zrcalu. Zato se u starijim video igrama zrcala gotovo nikada ne nalaze u velikim prostorijama, već isključivo u malim skučenim prostorima gdje iscrtavanje dvostruke količine poligona nije pretjerano računski zahtjevno. Neki su se programeri video igara pak odlučili jednostavno zanemariti zrcala i iscrtavati ih kao srebrnu površinu koja ništa ne reflektira.

Slika 1. Scena iscrtana uz pomoć ray tracinga

Na slici 1 prikazan je primjer scene iscrtane uz pomoć *ray tracing* algoritma. Uz korištenje starijih algoritama kao što su *scanline* ili *ray-casting*, objekti kao led u čaši ne bi nikada mogli izgledati ovako detaljno i realistično. Teoretski bi se moglo kocku leda nacrtati na ovakav način, no to bi bila samo simulacija koja bi izgubila smisao čim scenu pokušamo preseliti u svijet interaktivne grafike tako što, recimo, promijenimo očište malo ulijevo.



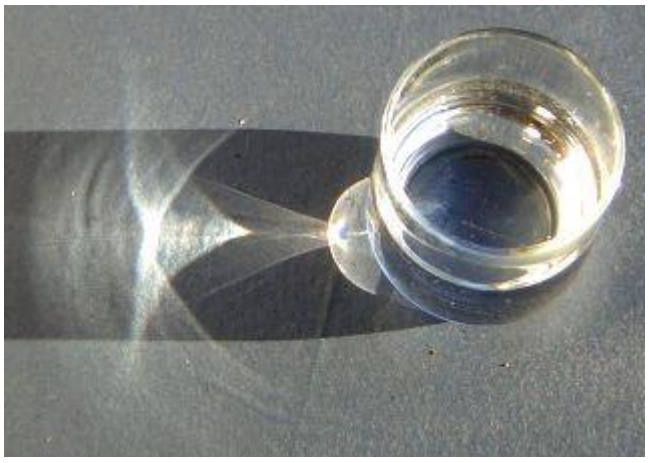
Slika 2. Još jedna scena iscrtana uz pomoć ray tracinga

Ray tracing „unazad“

Algoritam praćenja zraka obično se radi obrnuto od stvarnosti: zrake pratimo od očišta do izvora svijetlosti, dok u stvarnosti fotoni putuju od izvora do oka. Danas se pod pojmom *ray tracing* podrazumijeva ovakav princip rada „unazad“. S obzirom da je to pretpostavljeni način rada, praćenje zraka unazad sada znači praćenje svijetlosti od izvora do očišta, što se protivi intuiciji s obzirom da je to „pravi“ put u prirodi.

Razlog zašto praćenje zraka radimo unazad je većinom zbog količine računanja. Kada bi pratili zrake od izvora, očito je da velika većina zraka neće završiti baš u točki očišta, nego negdje drugdje. Sve te zrake koje će završiti na drugim mjestima su nam potpuno nebitne za crtanje scene, pa nam ovakav pristup uvodi ogromnu količinu nepotrebnog računanja putanja zraka.

Ipak, praćenje zraka unazad ima prednosti u određenim situacijama. Neki indirektni efekti svijetla realističnije su iscrtani ako zrake pratimo od izvora, kao primjerice refraktirana svjetlost koja se projicira na površinu ispod čaše vode (slika 3).

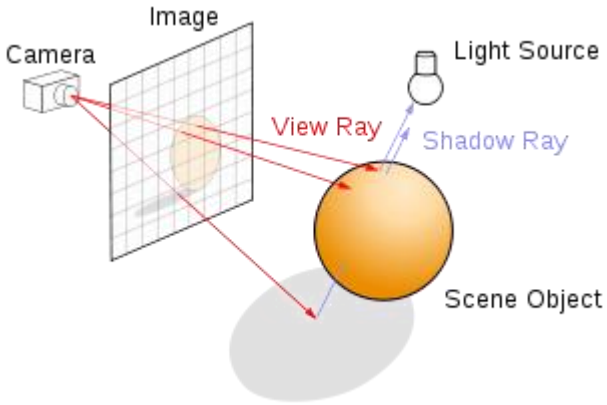


Slika 3. Projekcija refraktirane svijetlosti kroz čašu vode

Za omogućavanje iscrtavanja ovakvih fenomena u interaktivnoj grafici, osmišljene su neke metode praćenja zraka koje kombiniraju oba pristupa. To su primjerice dvosmjerno praćenje zraka i mapiranje fotona.

Princip rada algoritma

Iscrtavanje scene pomoću praćenja zraka radi na slijedeći način. Za svaki piksel koji želimo iscrtati na ekranu šaljemo zraku svjetlosti koju zatim pratimo kroz njen put, te računamo sve kutove refleksije i refrakcije prilikom sudara s objektima da bi rekurzivno dobili konačnu putanju zrake. Obično se dozvoljava ograničen broj odbijanja (deset do dvadeset) prije nego zraku preusmjerimo prema izvoru svjetlosti i izračunamo konačnu boju piksela.



Slika 4. Jednostavan primjer ray tracinga

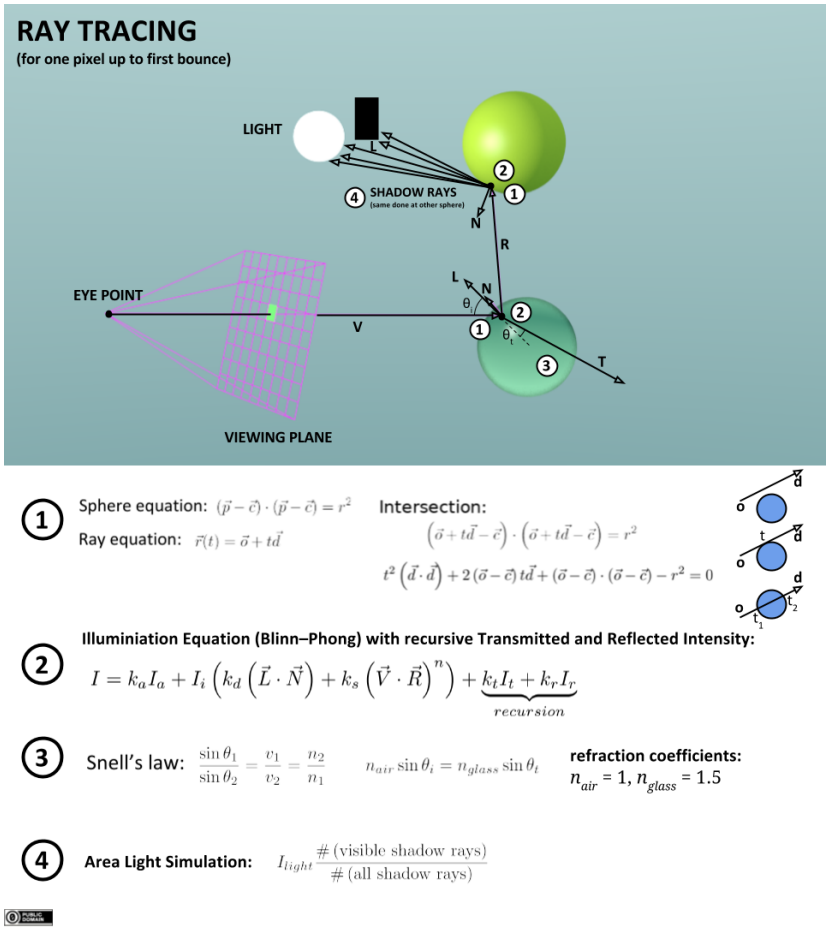
Na slici 4 prikazan je jednostavan primjer praćenja zraka. U sceni imamo samo jednu kuglu tako da ne ulazimo još u dublje rekurzivne zrake. Zrake su u suštini pravci povučeni između očišta i piksela kojeg želimo iscrtati, a taj pravac pratimo do prvog sudara s objektom. Nakon tog sudara, preusmjeravamo zraku prema izvoru svjetlosti. Tu preusmjerenu zraku nazivamo „shadow ray“ ili zraka sjene, jer nam ona pomaže odrediti konačno osvjetljenje piksela. Ako zraka sjene uspije doći do izvora svjetlosti bez sudara s drugim objektom, znamo da piksel nije u sjeni i piksel možemo obojati u boju objekta s kojim se zraka zadnje sudarila. Primjer zrake sjene koja ne uspijeva doći do izvora vidimo na najdonjoj zraci na slici 4, gdje se zraka odbija od „poda“ scene i zatim sudara s objektom na putu do izvora. Zato taj dio scene crtamo u sjeni i tako dobivamo realističan prikaz sjene ispod kugle.

Prošli odlomak objašnjava kako bojamo piksel u slučaju da smo pratili zraku do prvog sudara, poslali zraku sjene i zatim obojali piksel ovisno o tome je li zraka uspjela doći do izvora ili ne. Ovo je situacija koja će se odvijati u većini

slučajeva. Većina scena nema pretjerano puno reflektivnih ili prozirnih površina, pa se iscrtavanje većine površina svodi na sljedeće:

1. Šaljemo zraku kroz piksel koji želimo nacrtati
2. Ako se zraka sudari s površinom objekta koja je neprozirna, šaljemo zraku sjene prema izvoru (ili izvorima) svjetlosti
3. Ako se zraka sjene ne sudari s novim objektom na putu do izvora, bojamo piksel koristeći npr. Phongov model
4. Ako se zraka sjene sudari s novim objektom, bojamo piksel u tamniju boju od boje površine (obično ne skroz crnu, i dalje pretpostavljamo neko ambijentalno osvjetljenje)

Situacija je nešto složenija za slučaj da se zraka u nekom trenutku sudari s reflektivnom ili prozirnom površinom. U tom slučaju moramo pratiti novu putanju reflektirane ili refraktirane zrake sve dok se ne sudari s neprozirnom površinom. Tek kada se to dogodi možemo rekurzivno unazad izračunati boju piksela. Primjer je prikazan na slici 5.



Slika 5. Dijagram i formule za računanje boje piksela pomoću praćenja zraka

Sada smo pokrili sve slučajeve osim jednog: što ako se zraka odbije od reflektirajuće ili prozirne površine (ili više njih), a zatim se nikada ne sudari s konačnim neprozirnim objektom? U interaktivnoj grafici radimo s konačnim, ograničenim prostorima i ograničenim brojem objekata i može se dogoditi da zraka ode u „nebo“ i ne sudari se putem s nijednim čvrstim neprozirnim objektom. Za takve situacije obično se implementira *skybox* odnosno pozadinska slika koja obuhvaća pozadinu cijele scene. *Skybox* nije nikakva nova stvar već se koristi i u starijim vrstama iscrtavanja kao što je rasterizacija, tako da tu nemamo problema. Dakle, zraka će se sigurno sudariti barem sa površinom *skyboxa*, te ćemo ponovo znati izračunati konačnu boju piksela.

Ray tracing u praksi – Nvidia RTX

Kao što je prethodno spomenuto, praćenje zraka nije bilo implementirano u video igrama i interaktivnoj grafici zbog velike količine računanja potrebne za iscrtavanje imalo složenijih scena. Rasterizacija se pokazala kao puno efikasnija metoda koja i dalje pruža zadovoljavajuću razinu realističnosti prikaza.

Međutim, proizvođači hardvera kao ni programeri video igara nikad nisu odbacili ideju praćenja zraka, već je stavljena na čekanje dok se tehnologija grafičkih kartica nije dovoljno razvila da podrži tu količinu računanja i održi razumnu količinu slika po sekundi (barem 60 FPS). Upravo to se dogodilo 2018. godine kada proizvođač grafičkih kartica Nvidia izdaje novu seriju kartica pod imenom RTX. RT u imenu, naravno, znači ray tracing.

Nvidia grafičke kartice iz serije GTX sadržavale su samo CUDA jezgre. CUDA jezgre su „obični“ procesori kojih jedna takva grafička kartica ima po nekoliko tisuća. Ove jezgre koriste se za standardne zadatke koji se očekuju od grafičke kartice, kao što je klasično iscrtavanje slike rasterizacijom ili računanje velikih matričnih operacija ako CUDA jezgre koristimo u svrhu dubokog učenja. Ti su procesori relativno jednostavni ali kada upogonimo velik broj njih u paraleli, možemo dobiti impresivnu razinu realističnosti prikaza. No počevši sa serijom RTX, Nvidia grafičke kartice dolaze s još dvije vrste jezgri: Tenzor jezgre i RT jezgre.

Tenzor jezgre specijalizirane su jezgre koje služe za računanje s tenzorima, koje na nekoj površnoj razini možemo shvatiti kao matrice. Ove jezgre jako su korisne za duboko učenje jer su optimiziranije za rad s tenzorima nego CUDA jezgre. Međutim, osim što ih možemo koristiti na taj način, Nvidia ih koristi i za grafiku – ove jezgre surađuju s RT jezgrama tako da čiste sliku iscrtanu pomoću *ray tracinga*, ali mogu obavljati i inteligentno povećanje rezolucije slike. Nvidia tu tehnologiju naziva DLSS što znači Deep Learning Super Sampling. DLSS omogućava efikasno prikazivanje slike veće rezolucije no što je sama grafička kartica sposobna iscrtati.

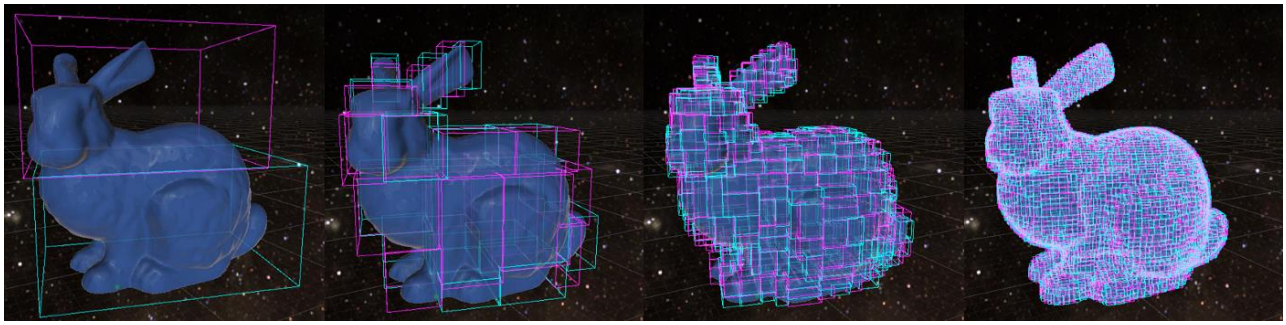
Konačno, najbitniji dodatak RTX seriji su RT jezgre. RT jezgre rade u suradnji s CUDA jezgrama da bi omogućile efikasno izvođenje praćenja zraka.

RT jezgre i BVH

RT jezgre ne obavljaju cijeli posao praćenja zraka kao što je opisano u prethodnim poglavljima jer praćenje po jedne ili više zraka za svaki iscrtani piksel i dalje nije računski moguće uz održavanje zadovoljavajućeg broja okvira po sekundi. Moderni *ray tracing* zato „vara“ korištenjem BVH (*bounding volume hierarchy*) strukture za lakše postizanje praćenja zraka.

Prije objašnjenja principa BVH potrebno je razumjeti zašto je potreban. Pronaći sudara li se zraka svjetlosti s jednostavnim geometrijskim tijelom kao što je sfera je jednostavno, jer postoje matematičke formule kojima možemo pronaći točno mjesto sudara sfere i pravca (zrake). Međutim, u modernoj grafici se gotovo nikad ne pojavljuju sfere i kocke već kompleksni objekti sastavljeni od milijuna poligona. Pronaći mjesto sudara zrake i složenog objekta puno je zahtjevnije, te bi podrazumijevalo iteriranje kroz svaki poligon objekta i provjeru siječe li naša zraka baš taj poligon.

Kako bi se zaobišlo takvo računanje, osmišljen je BVH. Vizualizacija ovog algoritma prikazana je na slici 6.

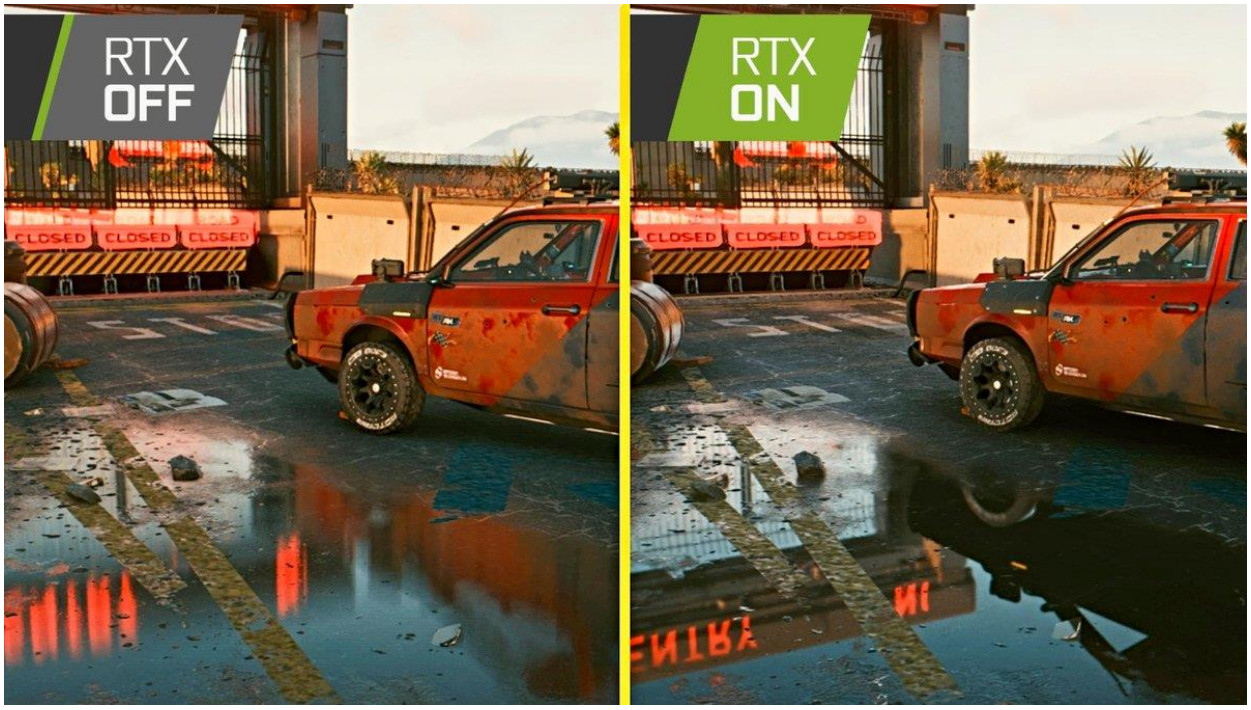


Slika 6. Primjer BVH strukture nad objektom zeca

Složeni objekt možemo „obuhvatiti“ u nekoliko jednostavnih geometrijskih tijela, najčešće kvadra. Zatim, kada pratimo zraku, možemo provjeriti prolazi li kroz jedan takav kvadar. Ako ne prolazi kroz nijedan od najvećih kvadara (skroz lijeva slika), onda znamo da sigurno ne prolazi kroz objekt. No, ako siječe jedan od kvadara, znamo da možda prolazi kroz objekt, te idemo korak dublje i provjeravamo siječe li zraka neki od manjih kvadara na drugoj slici s lijeva. Ako siječe jedan od njih, prelazimo na još manje kvadre sve dok ne pronađemo mjesto dodira zrake i objekta ili se uvjerimo da zraka ipak ne prolazi kroz objekt. Svakom kvadru je poznato koje manje kvadre sadrži, pa na kraju dobivamo hijerarhiju ograničavajućih volumena kako i samo ime strukture govori.

Upravo ovo računanje je posao RT jezgri u RTX seriji grafičkih kartica. RT jezgre su ustvari ASIC-i (*application-specific integrated circuit*) koji su dizajnirani da ogromnom brzinom računaju sijeku li se pravac i BVH struktura. Taj posao RT jezgrama zadaju CUDA jezgre kako bi si znatno ubrzale proces praćenja zraka, a zatim one obavljaju ostatak posla. Međutim, ovaj cijeli proces rezultira slikom koja je zrnasta, jer umjesto slanja zrake kroz svaki piksel na ekranu, CUDA jezgre šalju po zraku za svaku BVH strukturu na ekranu, što je znatno manje zraka od ukupnog broja piksela. Zrnatost slike u konačnici popravljaju Tenzor jezgre koristeći prethodno spomenute metode povećanja rezolucije za koje su optimizirane. Ovaj cijeli „ekosustav“ u konačnici postiže grafiku ostvarenu pomoću praćenja zraka koja nije toliko računski zahtjevna a ostvaruje prilično realistične slike.

Usporedba klasičnog iscrtavanja i ray tracinga



Slika 7. Usporedba scene bez i sa praćenjem zraka u igri Cyberpunk 2077

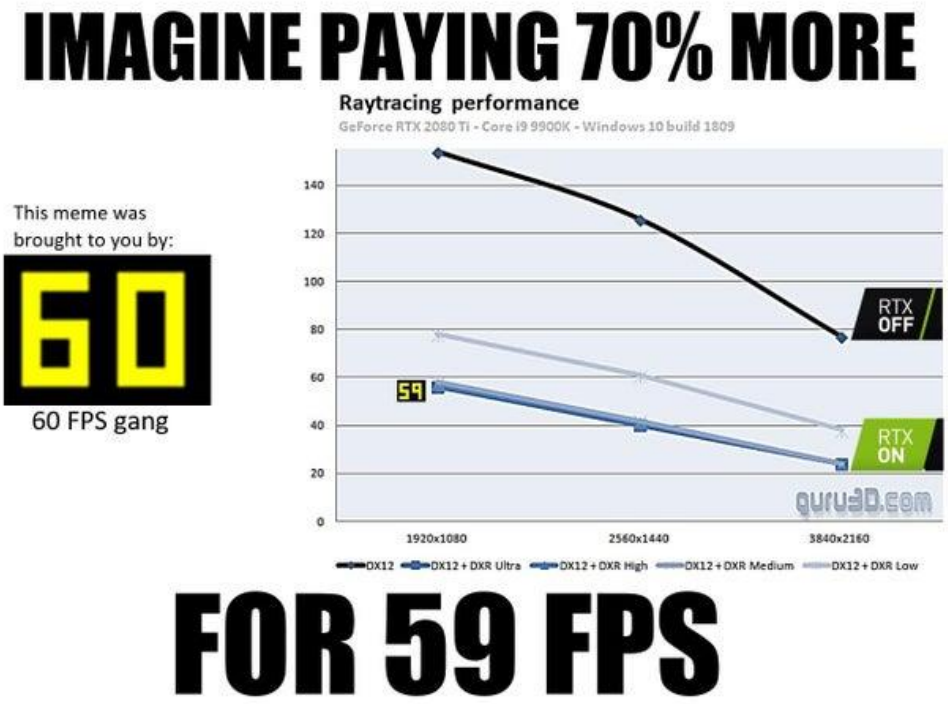
Na slici 7 prilično se jasno vidi razlika u sceni iscrtanoj bez i sa praćenjem zraka. Preciznije, najveća je razlika vidljiva u lokvi s donje lijeve strane scene – s praćenjem zraka odraz u vodi izgleda puno bolje. Osim toga, valja obratiti pažnju na iscrtavanje sjena oko auta i oko zida u stražnjem dijelu scene, gdje se jasno vidi koliko su sjene „nježnije“ i realističnije uz *ray tracing*.

Ograničenja ray tracinga

Ipak, čak i uz svu moguću optimizaciju, praćenje zraka ima značajan utjecaj na ostvareni FPS u igrama. Grafičke kartice serije RTX u jeftinijem rangu kao što je RTX 2060, iako u potpunosti podržavaju praćenje zraka kao i njihove jače varijante, jednostavno imaju premalo CUDA jezgri koje rade na preniskoj frekvenciji da bi omogućile doživljaj *ray tracinga* uz prihvatljiv broj okvira u sekundi.

Još jedno ograničenje ove tehnologije je sama implementacija algoritma u video igrama. S obzirom da je primjena ove metode tek od nedavno moguća, mali je broj video igara koje uopće iskorištavaju hardverske sposobnosti RTX

grafičkih kartica. Što se tiče implementacije, u teoriji je praćenje zraka jednostavniji način iscrtavanja scene nego standardno primijenjena rasterizacija. To je logička posljedica toga što praćenje zraka oponaša prirodu, a rasterizacija ju aproksimira, zaobilazi neke problematične elemente i na razne načine „vara“ kako bi prikazala realistične scene. Unatoč tome, rasterizacija i ostale standardne metode iscrtavanja scene ipak su korištene dugi niz godina i već su dobro poznate programerima video igara, pa će zasigurno trebati barem još nekoliko godina da praćenje zraka postane novi standard u interaktivnoj grafici.



Slika 8. Meme preuzet s Reddita vezan za performanse grafičkih kartica uz ray tracing

Zaključak

Algoritam praćenja zraka nova je metoda iscrtavanja slika u interaktivnoj grafici popularizirana 2018. Nvidia RTX serijom grafičkih kartica koje uz razne oblike optimizacije omogućavaju realističniji prikaz i osvjetljenje scene. Sama metoda osmišljena je još 1970-ih a bazira se na prirodnom kretanju fotona i zraka svjetlosti. Zbog velikog broja računanja potrebnog za ostvarenje ove metode, do nedavno nije korištena u interaktivnoj grafici već isključivo u filmskoj industriji gdje vrijeme iscrtavanja slike nije toliko bitno. Sada postaje sve zastupljenija i u području interaktivne grafike, ali i dalje se suočava s brojnim ograničenjima opisanim u ovom radu.

Literatura

Sve internetske poveznice posjećene 17.1.2023.

1. https://en.wikipedia.org/wiki/Ray_tracing_(graphics)
2. https://www.ppsloan.org/publications/rtrt99.pdf
3. https://www.electronicdesign.com/technologies/displays/article/21801219/whats-the-difference-between-ray-tracing-ray-casting-and-ray-charles
4. https://www.quora.com/Why-do-so-many-video-games-have-an-aversion-to-using-working-mirrors-in-their-environments
5. https://en.wikipedia.org/wiki/Caustic_(optics)
6. https://www.hardwaretimes.com/what-is-ray-tracing-and-how-does-it-work-are-nvidias-rtx-gpus-worth-it/
7. https://math.hws.edu/graphicsbook/c8/s1.html
8. https://blogs.nvidia.com/blog/2022/03/23/what-is-path-tracing/
9. https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-overview/light-transport-ray-tracing-whitted.html
10. https://www.titancomputers.com/What-Are-RT-Cores-in-Nvidia-GPUs-s/1208.htm