Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/02SA01Vasiljevic

TEORIJSKE OSNOVE I IMPLEMENTACIJA *RAYTRACER* ALGORITMA ZA RENDEROVANjE

THEORETICAL BASICS OF THE RAYTRACER ALGORITHM FOR RENDERING

Ivana Vasiljević, Lidija Krstanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSKA GRAFIKA

Kratak sadržaj – *Raytracer je algoritam globalnog osvetljenja koji se u današnje vreme koristi za kreiranje (renderovanje, eng.* rendering*) slika visokog stepena realističnosti. U osnovi, ideja je imitirati način realnog prostiranja svetlosti kroz prostor. Ako je prostor kroz koji se svetlost kreće homogen i izotropan, putanje svetlosti su prave linije. Dakle, problemu bi mogli pristupiti ovako: krenuvši od nekog izvora svetlosti pratili bismo kretanje jednog od zraka. Prilikom kretanja kroz prostor u jednom trenutku zrak bi došao do dioptrijskih površina i pritom se jedan njegov deo reflektuje, drugi prelama, a treći apsorbuje. U slučaju da površina nije dioptrijska jednostavno bismo zanemarili prelamanje svetlosti. Sada bi pratili novonastale zrake i ponavljali postupak za svako presecanje zraka sa površinom dok zrak ne bi došao do sočiva kamere ili oka promatrača. Samo takve zrake koje u konačnici prolaze kroz povšinu sočiva ili oka doprinose generisanju slike. Postupak bi trebalo ponoviti za svaki zrak poslat od odgovarajućeg izvora svetlosti, i isto uraditi za sve ostale izvore svetlosti (ukoliko postoje) na sceni.*

Ključne reči: *Raytracer, rendering,* algoritam, *zraci, slike*

Abstract – *Raytracer is a global lighting algorithm which is used for rendering images with a high degree of realism. Basically, the idea is to imitate the propagation of light rays through the space. If the space through which light moves is homogeneous and isotropic, the path of light ray is the straight line. So, the problem could be accessed in the following way: by moving from a source of light we would follow the movement of one of the ray. When moving through the space at one time the ray would come to the dioptic surfaces, and one part of it would reflect, the other refracts, and the third absorbed. In case the surface is not dioptric, we would simply ignore the reflection of light. Now we will follow the generated rays and repeat the procedure for each ray intersection with the surface until the ray would reach the lens of the camera or the eye of the observer. Only rays which ultimately pass through the lens surface or eye contribute to the generation of an image. The procedure should be repeated for every ray sent from the appropriate light source, and it should be done the same for all other light sources (if any) on the scene.*

Keywords: *Raytracer, rendering, algorithm, rays, images*

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Lidija Krstanović, docent.

1. UVOD

Jednostavan *raytracer* svodi se na izvršavanje sledećih operacija:

- definiši nekoliko objekata
- odredi materijal za svaki objekat
- definiši nekoliko izvora osvetljenja
- definiši prozor čija je površina pokrivena pikselima
- za svaki piksel
  - usmeri zrak prema objektima iz centra piksela
  - odredi najbližu presečnu tačku zraka sa objektom (ili bilo koju)
  - ukoliko zrak pogodi objekat
    - uz pomoć materijala piksela i osvetljenja odredi boju piksela

ili

  - ukoliko zrak ne pogodi objekat
    - boju piksela postavi da bude crna

Ovo je poznato kao *ray casting*.

U ovom dokumentu biće pojašnjen *C++* izvorni kôd koji demonstrira rad *raytracer* algoritma. Kao, što je već rečeno, *raytracing* matematički simulira ponašanje svetlosti i njene interakcije sa fizičkim objektima kao što su refleksije, refrakcije, pojava senki. *Raytracing* omogućava kreiranje realističnih specijalnih efekata za filmove i igre, kao i vizualizaciju organskih molekula za biologe.

2. OPŠTI PRISTUP ZA ODREĐIVANjE PRESEKA ZRAKA SA OBJEKTIMA

Postavlja se pitanje kako pronaći presek zraka sa objektom na sceni? Ovo je poseban problem algebre koji mora biti rešen za sfere, ali i za sve druge objekte. Svaki objekat je napisan u *C++* kôdu kao zasebna klasa izvedena iz klase *SolidObject*. Svaka izvedena klasa uključuje matematičko rešenje za presek zraka sa datim oblikom/objektom u metodi *AppendAllIntersections*. U ovom radu biće objašnjeno određivanje preseka zraka sa sferom, kao najjednostavnijim objektom. Na isti način se vrši određivanje preseka zraka i sa kompleksnijim objektima. Određivanje preseka zraka sa bilo kojim objektom se svodi na izvršenje sledećih koraka:

- Napisati parametarsku jednačinu za bilo koju tačku na datom zraku. To znači da će jednačina

zraka koristiti parametar $u > 0$ kako bi mogla biti odabrana bilo koja tačka duž zraka.

- Napisati jednu ili više jednačina koje definišu dati objekat.
- Algebarski rešiti sistem jednačina iz prethodnih koraka. Rešenja jednačina daće nam tačke koje se nalaze na površini objekta duž zraka, odnosno tačke preseka zraka sa objektom.
- Za svaki presek zraka sa objektom potrebno je odrediti jedinični vektor koji je normalan na površinu objekta i pokazuje ka spoljašnosti.
- Strukturu *Intersection* popuniti sa lokacijama tačaka preseka, kvadratnog rastojanja između tačke preseka i očne tačke, jediničnog vektora normale u datoj tački. Svaki član metode *AppendAllIntersections* mora kopirati pokazivač objekta sa kojim je zrak ostvario presek u član strukture *solid* što se koduje kao

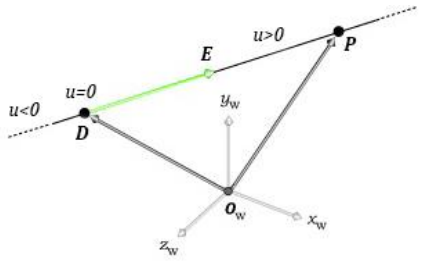
  intersection.solid = this;
- Dodati strukturu *Intersection* na listu izlaznih parametara *intersectionList.*

2.1. Parametarska jednačina zraka

Zrak je prava koja je definisana sa tačkom $\boldsymbol{D}$, koja se naziva početak, i jediničnim vektorom $\boldsymbol{E}$, koji se naziva pravac, odnosno vektor pravca. Zrak je parametrizovan parametrom u koji se naziva parametar zraka, gde je $u = 0$ na početku zraka, tako da tačka $\boldsymbol{P}$ na zraku može biti izražena kao

$$\boldsymbol{P} = \boldsymbol{D} + u\boldsymbol{E} \tag{1}$$

Na slici 1. dat je vizuelni prikaz zraka. Pravac $\boldsymbol{E}$ definiše pozitivan smer zraka duž linije, gde se parametar u povećava sa porastom parametra $\boldsymbol{E}$. Vektor $\boldsymbol{E}$ je jedinični, dok u određuje rastojanje zraka od njegovog početka [1]. Iako zrak ima svoj početak, u ima vrednost u intervalu $(-\infty, +\infty)$, tako da prethodna jednačina definiše pravu, a ne duž. Početak i vektor pravca izraženi su u globalnim koordinatama pre nego što zrak „naiđe“ na objekat.



Slika 1. *Zrak u globalnom koordinatnom sistemu*

2.2. Jednačina sfere

Ukoliko se određuje presek zraka sa sferom, jednačinu sfere zapisaćemo u vektorskoj formi:

$$(\boldsymbol{P_x} - \boldsymbol{C_x})^2 + (\boldsymbol{P_y} - \boldsymbol{C_y})^2 + (\boldsymbol{P_z} - \boldsymbol{C_z})^2 = R^2 \tag{2}$$

gde je $\boldsymbol{P} = (\boldsymbol{P_x}, \boldsymbol{P_y}, \boldsymbol{P_z})$ bilo koja tačka na sferi (namerno isto $\boldsymbol{P}$ kao u parametarskoj jednačini zraka iznad), $\boldsymbol{C} = (\boldsymbol{C_x}, \boldsymbol{C_y}, \boldsymbol{C_z})$ je centar sfere izražen kao vektor položaja , a R je radijus sfere.

2.3. Presek zraka sa sferom

Za određivanje preseka zraka sa sferom potrebno je $\boldsymbol{P}$ zameniti jednačinom (1) u jednačini u (2):

$$((\boldsymbol{D_x} + u\boldsymbol{E_x}) - C_x)^2 + \left((\boldsymbol{D_y} + u\boldsymbol{E_y}) - C_y\right)^2 + \left((\boldsymbol{D_z} + u\boldsymbol{E_z}) - C_z\right)^2 = R^2 \tag{3}$$

Kvadriranjem određenih izraza, dobija se

$$\boldsymbol{E_x}^2u^2 + 2\boldsymbol{E_x}(\boldsymbol{D_x} - \boldsymbol{C_x})u + (\boldsymbol{D_x} - \boldsymbol{C_x})^2 + \boldsymbol{E_y}^2u^2 + 2\boldsymbol{E_y}(\boldsymbol{D_y} - \boldsymbol{C_y})u + (\boldsymbol{D_y} - \boldsymbol{C_y})^2 + \boldsymbol{E_z}^2u^2 + 2\boldsymbol{E_z}(\boldsymbol{D_z} - \boldsymbol{C_z})u + (\boldsymbol{D_z} - \boldsymbol{C_z})^2 = R^2 \tag{4}$$

Ovaj izraz može biti napisan kao standardna kvadratna jednačina:

$$(\boldsymbol{E_x} + \boldsymbol{E_y} + \boldsymbol{E_z})u^2 + 2\left(\boldsymbol{E_x}(\boldsymbol{D_x} - \boldsymbol{C_x}) + \boldsymbol{E_y}(\boldsymbol{D_y} - \boldsymbol{C_y}) + \boldsymbol{E_z}(\boldsymbol{D_z} - \boldsymbol{C_z})\right) + (\boldsymbol{D_x} - \boldsymbol{C_x})^2 + (\boldsymbol{D_y} - \boldsymbol{C_y})^2 + (\boldsymbol{D_z} - \boldsymbol{C_z})^2 - R^2 = 0 \tag{5}$$

Jednačina (5) je kvadratna jednačina za u koja može biti napisana kao

$$au^2 + bu + c = 0 \tag{6}$$

Kvadratna jednačina rešava se uz pomoć standardnih izraza:

$$u = \frac{-b \pm \sqrt[2]{(b^2 - 4ac)}}{2a} \tag{7}$$

U zavisnosti od vrednosti parametara a, b i c, kvadratna jednačina može imati nijedno, jedno ili dva realna rešenja za u. Ovo je u vezi sa činjenicom da zrak može pogoditi sferu nijednom, jednom (tangencijalno) ili dva puta (ulazi u objekat u jednoj tački, a izlazi u drugoj). Vrednost izraza unutar kvadratnog korena rešenja kvadratne jednačine naziva se diskriminanta, određuje koliko preseka zrak ima sa sferom:

- $b^2 - 4ac < 0 \Rightarrow$ nema rešenja
- $b^2 - 4ac = 0 \Rightarrow$ jedno rešenje
- $b^2 - 4ac > 0 \Rightarrow$ dva rešenja .

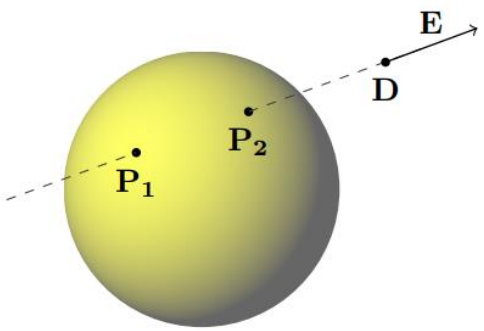
Vrednosti parametara a , b i c mogu se izraziti preko skalarnog proizvoda i intenziteta vektora (što je korisno za pisanje kôda):

$$a = |\boldsymbol{E}|^2$$
$$b = 2\boldsymbol{E}(\boldsymbol{D} - \boldsymbol{C})$$
$$c = |\boldsymbol{D} - \boldsymbol{C}|^2 - R^2$$

Prilikom izračunavanja diskriminante najpre se proverava da li je njena vrednost manja od nule. Ako jeste, presek zraka sa sferom ne postoji. U suprotnom, računaju se dve vrednosti parametra u (pošto se retko dešava da je vrednost diskriminante jednaka nuli ne vrše se posebne provere). Bilo da je diskriminanta nula ili pozitivna računaju se odvojene vrednosti za u korišćenjem predznaka $\pm$ u kvadratnoj formuli rešenja [2].

Brže rešenje je da pretpostavimo da postoje dva preseka, čak i ako je to jedna tačka u prostoru.

Nije dovoljno proveriti samo da li su rešenja realna; moramo proveriti svaku vrednost parametra u da vidimo da li je pozitivna. Kao što je ranije rečeno, negativna vrednost parametra u govori da je presečna tačka na polupravoj koja se ne razmatra (uzimamo u obzir samo tačke koje su na polupravoj određenom sa početkom $\boldsymbol{D}$ i smerom $\boldsymbol{E}$, kao što je prikazano na slici 2.).



Slika 2. *Kada je* $u < 0$*, to znači da se presečna tačka nalazi suprotno od željenog smera vektora* $\boldsymbol{E}$ *od tačke* $\boldsymbol{D}$*. U ovom primeru postoje dve presečne tačke koja leže na delu poluprave koji ne razmatramo. Za* P_1 $u = -2.74$*, a za* $P_2, u = -1.5$ [3]

Ukratko, ako je $u = 0$, presek je u očnoj tački, ako je $u<0$ presek je iza očne tačke, a ako je $u>0$, presek je ispred očne tačke. Dakle, samo pozitivne vrednosti u određuju validne preseke zraka sa sferom.

Kada je parametar $u > 0$, veća je vrednost u, što je presečna tačka udaljenija od očne tačke. Zbog grešaka zaokruživanja decimalnih vrednosti, ustvari se proverava da li je vrednost od u veća od male konstantne vrednosti $EPSILON$, a ne nule [4]. Ovo sprečava pronalazak pogrešnih presečnih tačaka koje su previše blizu očnoj tački, što bi uzrokovalo probleme pri izračunavanju senki i osvjetljenja. $EPSILON$ je definisano kao 10^{-6} u fajlu *imager.h*:

```
const double EPSILON = 1.0e-6;
```

Ako su nađena pozitivna realna rešenja za u, te vrednosti mogu se uvrstiti u kvadratnu jednačinu $\boldsymbol{P} = \boldsymbol{D} + u\boldsymbol{E}$ kako bi se dobile tačne lokacije presečnih tačaka.

3. OPTIČKI PRORAČUNI

Već je spomenuto kako *Scene::SaveImage* poziva *Scene::TraceRay* da traži zrake koji dolaze iz kamere na scenu. *TraceRay* određuje da li se zrak preseca sa površinom bilo kojeg objekta na sceni. Ako *TraceRay* pronađe jedan ili više preseka duž određenog pravca, onda odabere najbliži presek i poziva *Scene::CalculateLighting* da odredi boju piksela.

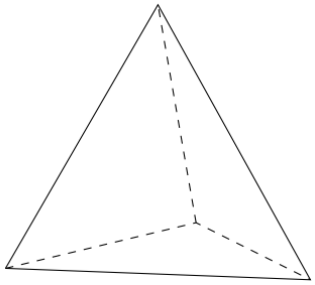
CalculateLighting predstavlja "srce" *raytracer* algoritma. Na osnovu jednačina fizike određuje gde se zrak odbija, prelama i rasipa na više zraka. Ova funkcija vrši skoro sve proračune u ovom radu. Na istom principu se vrše proračuni osvetljenja za sve objekte na sceni, tako da primeri iz *SolidObject* pozivaju ovu funkciju. Jedina razlika je u jednačini površine objekta.

CalculateLighting prolazi listu objekata kod kojih je registrovan presek zraka sa njegovom površinom i to u tački koja je najbliža kameri. Njeno osnovno zaduženje je da uzme u obzir doprinose svih zraka poslatih ka kameri od tačke preseka i da kao rezultat vrati *Color* strukturu koji u sebi sadrži zasebne crvene, zelene i plave vrednosti boje. Poziva sledeće pomoćne funkcije za određivanje boje piksela zasnovanih na različitim fizičkim procesima:

- *CalculateMatte* određuje mat refleksiju u tački preseka, koja nastaje mekim rasipanjem svetlosti kod materijala poput papira ili krede. Mat refleksija raspršuje incidentno svetlo u svim pravcima jednako.
- *CalculateReflection* izračunava refleksiju ogledala u određenoj tački, kod materijala poput stakla ili srebra. Refleksija ogledala dovodi do reflektovanj svetlosnog zraka od objekta pri čemu reflektovani zrak zaklapa jednak ugao sa normalom na površinu objekta, kao i upadni samo što je sa druge strane normale. Odbijanje zraka ne vrši se u svim pravcima kao što je slučaj kod mat refleksije. Refleksija ogledala se sastoji od sjajnog odsjaja zbog imaginarnog prelaza na objektu koji može imati bilo koju željenu boju. Oba tipa refleksije (sjajna i mat) se sabiraju kako bi se kreirala kompozitna reflektovana slika.
- *CalculateRefraction* vrši proračune vezane za pelamanje svetlosti pri nailasku na transparentne materijale. Prozirni objekti sa zakrivljenim površinama uzrokuju da drugi objekti koji se vide kroz njih izgledaju izobličeni po veličini i obliku.

4. KREIRANjE OBJEKATA UZ POMOĆ TROUGLOVA

C++ klasa *TriangleMesh* potiče od klase *SolidObject*. Ona predstavlja objekat čije su sve površine trouglovi. Jednostavan primer takvog objekta je četvorostrani pravilni poliedron zvan *tetrahedron* (vidi sliku 3.).

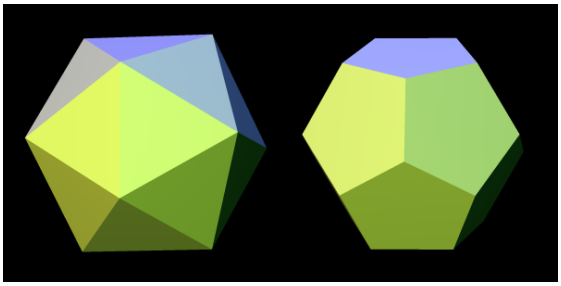


Slika 3. *Pravilni četvorostrani* tetrahedron

Kreiranje *tetrahedorna*, vrši se pozivanjem kôda koji prati sledeće korake:

- Dinamički dodeljuje instancu *TriangleMesh* korišćenjem operatora *new*.
- Poziva instancu *AddPoint* metode definiše tačke (vrhove) objekta. Svaka od tih tačaka zajednička je za tri površine (tri trougla). Svaki poziv *AddPoint* mora sadržati celobrojnu vrednost indeksa kao parametra. Prvim pozivom dodeljuje se vrednost nula, drugim pozivom dodeljuje se vrednost jedan, itd. Ovaj parametar naziva se *pointIndex* i služi programerima za proveru. Ako se pogrešna vrednost prenese, *AddPoint* će je posmatrati kao izuzetak.
- Poziva *AddTriangle* kako bi bila dodata svaka od površina objekta. Prolazi kroz tri različita indeksa tačaka kako bi odredio između koje tri tačke kreira površinu.

- Dodaje *TriangleMesh* instancu u *Scene* kako bi bila renderovana na slici.

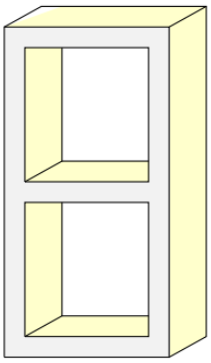


Slika 4. *Generisane slike* icosahedron-*a (levo) i* dodecahedron-*a (desno) (generisana korišćenjem prikazanog programa)*

5. OPERATORI NAD SKUPOVIMA

Iako je moguće kreirati prilagođenu klasu za svaki željeni oblik koji se može zamisliti, to uključuje niz matematičkih proračuna.
Neki oblici mogu biti kombinacija oblika koje su već implementirali ili deo postojećeg oblika. Za ovakve oblike, često je moguće izbeći puno truda korišćenjem nekih *C++* klasa definisanih u *imager.h* koje nam omogućavaju da kombinujemo oblike koje imamo da bismo kreirali nove. Ove klase nazivaju se operatori. Spomenute klase operatori su:

- *SetUnion*
- *SetIntersection*
- *SetComplement*
- *SetDifference*



Slika 5. *Oblik predstavljen klasom ConcreteBlock (generisana korišćenjem prikazanog programa)*

Kombinovanjem navedenih operacija na jednostavan način moguće je kreirati kompleksnije objekte, kao što je prikazano na slici 5.

6. ZAKLjUČAK

U ovom dokumentu prikazane su osnove rada *raytracer* algoritma, takođe je i opisan načine implementacije jednog takvog algoritma. Za razumevanje rada algoritma potrebno je razumevanje algebre i analitičke geometrije, kao i osnova *C++* programiranja. U radu je uključeno i poglavlje o vektorima jer su oni od primarnog značaja za rad *raytracer* algoritma. Takođe, prikazani su ključni koncepti računarske grafike transformisani u slike koje su dobijene kao rezultat.

Samom implementacijom ovog algoritma pokušano je imitiranje realnih optičkih osobina kretanja svetlosti, a sve to uz "podršku"zakona fizike.

7. LITERATURA

[1] P. Shirley, R.K. Morley, *Realistic ray tracing.* AK Peters, Ltd.,2008.
[2] G. Humphreys, M. Pharr, *Physically Based Rendering.* Morgan Kaufmann, 2004
[3] http://cosinekitty.com/raytrace/ (pristupljeno u julu 2018.)
[4] K. Suffern, *Ray Tracing from the Ground up*. AK Peters/CRC Press, 2016.

Kratka biografija:

Ivana Vasiljević rođena je u Vlasenici 1993. god. Osnovne studije završila je 2016. godine na Fakultetu tehničkih nauka, oblasti Računarska grafika. Master rad na Fakultetu tehničkih nauka iz oblasti Računarske grafike odbranila je 2018.god. Trenutno je zaposlena kao saradnik u nastavi na istom fakultetu, na katedri za Animaciju u inženjerstvu.
kontakt: mail.ivanav@uns.ac.rs

Lidija Krstanović završila je osnovne i master studije na Prirodno-matematičkom fakultetu u Novom Sadu, smer profesor matematike. Doktorirala je 2017. godine na Fakultetu tehničkih nauka, smer Matematika u tehnici. Godine 2018. izabrana je u zvanje docenta na istom fakultetu na katedri za Animaciju u inženjerstvu.