# Dokumentacija iz predmeta Računalna grafika

3. laboratorijska vježba

Nikola Bunjevac (0036485677)

29. siječnja 2019.

## 1 Raymarching

Kao tema treće (samostalne) laboratorijske vježbe odabran je algoritam Raymarching. Glavna ideja algoritma je iskorištavanje paralelizma grafičke kartice (GPU) kako bi se brzo moglo "ispucati" zraku za svaki piksel slike, odrediti njezino sjecište s površinom u sceni te primijeniti osvjetljenje i prikazati rezultat. Možemo reći da je Raymarching algoritam srodan algoritmu praćenja zrake (engl. *raytracing*), ali za razliku od njega je aproksimativan. Provodi se unaprijed određeni maksimalni broj iteracija i određuje se najmanja udaljenost duž ispucane zrake do neke površine. Pseudokod je prikazan ispod.

**Algoritam 1** Funkcija Raymarch

```
1:  function RAYMARCH(vec3 ro, vec3 rd)
2:      float d = 0.0;
3:      for (int i = 0; i < MAX_STEPS; i++) do
4:          vec3 p = ro + d*rd;
5:          float dS = GETDIST(p); // funkcija udaljenosti
6:          d += dS;
7:          if (dS < SURFACE_DIST || d > MAX_DIST) then
8:              break;
9:          end if
10:     end for
11:     return d;
12: end function
```

Funkcija `Raymarch` kao argumente prima točku u kojoj se nalazi kamera i vektor smjera u kojemu će se zraka ispucati. U funkciji se izvodi petlja

koja u najviše `MAX_STEPS` koraka određuje udaljenost najbliže površine (u smjeru zrake) od kamere. Važna komponenta je funkcija `getDist` pomoću koje modeliramo scenu. Ona vraća polumjer kružnice sa središtem u trenutnoj točki duž zrake koja siječe najbližu površinu. To nam daje garanciju da se možemo pomaći duž zrake barem za dužinu polumjera kružnice bez opasnosti da "uđemo" u neki objekt u sceni. Iterativnom primjenom takvih pomaka, ili ćemo se približiti dovoljno blizu neke površine (`SURFACE_DIST`) i proglasiti sjecište, ili ćemo otići u beskonačnost.

Već smo spomenuli kako pomoću funkcije `getDist` modeliramo scenu. Svi objekti u prostoru su modelirani pomoću matematičkih funkcija što malo otežava modeliranje, ali nas ne ograničava u kreativnosti. Tako primjerice možemo jednom funkcijom predstaviti kocku, drugom kuglu, trećom torus itd. Također je funkcije moguće kombinirati raznim operacijama poput presjeka, unije, deformacije i slično. Primjeri funkcija nalaze se na stranici[1].

Korisno je napomenuti kako se sve iscrtava u stvarnom vremenu, uključujući i sjene. Njih dobivamo *gotovo besplatno* tako da u točki sjecišta provjerimo je li izvor svjetlosti izravno vidljiv. Ukoliko nije, taj piksel ćemo zatamniti kako bismo dobili dojam da se točka nalazi u sjeni.

## 1.1 Struktura programa

Program se sastoji od više funkcionalnih dijelova. Možemo ga podijeliti na glavni program koji stvara prozor i uspostavlja OpenGL kontekst, prevodi i povezuje sjenčare te izvršava glavnu petlju programa. Također je zadužen za slanje određenih informacija sjenčarima putem uniformnih varijabli. Informacije koje se prenose su

- vrijeme proteklo od trenutka pokretanja,
- položaj miša unutar prozora,
- rezolucija prozora (visina i širina).

Možemo primijetiti kako je ova struktura slična usluzi Shadertoy[2] što je bio i cilj glavnog programa–omogućiti jednostavnu lokalnu eksperimentaciju sa sjenčarima. Kako bi se omogućilo iscrtavanje na cijelom prozoru, glavni program sjenčaru vrhova (engl. *vertex shader*) prosljeđuje koordinate vrhova dvaju trokuta koji čine pravokutnik i zauzimaju cijelu površinu prozora. Sjenčar vrhova jednostavno prosljeđuje podatke pa ni njega više nećemo razmatrati u nastavku.

[1] https://www.iquilezles.org/www/articles/distfunctions/distfunctions.htm
[2] https://www.shadertoy.com/

Takva podjela omogućava nam da se usredotočimo na sjenčar fragmenata. Kako bismo mogli manipulirati pojedinim pikselima, možemo koristiti dimenzije prozora i UV koordinate. Rezultat je brza povratna veza između promjene u sjenčaru i rezultata na ekranu. Nema potrebe za ponovnim prevođenjem glavnog programa, a prevođenje sjenčara traje vrlo kratko.

## 1.2 Upute za pokretanje

Program je pisan u programskom jeziku C++ i koristi OpenGL te biblioteku SDL2[3]. Kao jedini argument prima putanju do sjenčara fragmenata kojeg izvodi. Program pokrećemo naredbom

```
./raymarching shader.frag
```

nakon čega se otvara novi prozor i započinje iscrtavanje. Pomicanjem miša omogućeno je upravljanje položajem izvora svjetlosti u sceni kako bi se pokazala interaktivnost, bez obzira što se većina programa izvodi u sjenčaru na grafičkoj kartici.

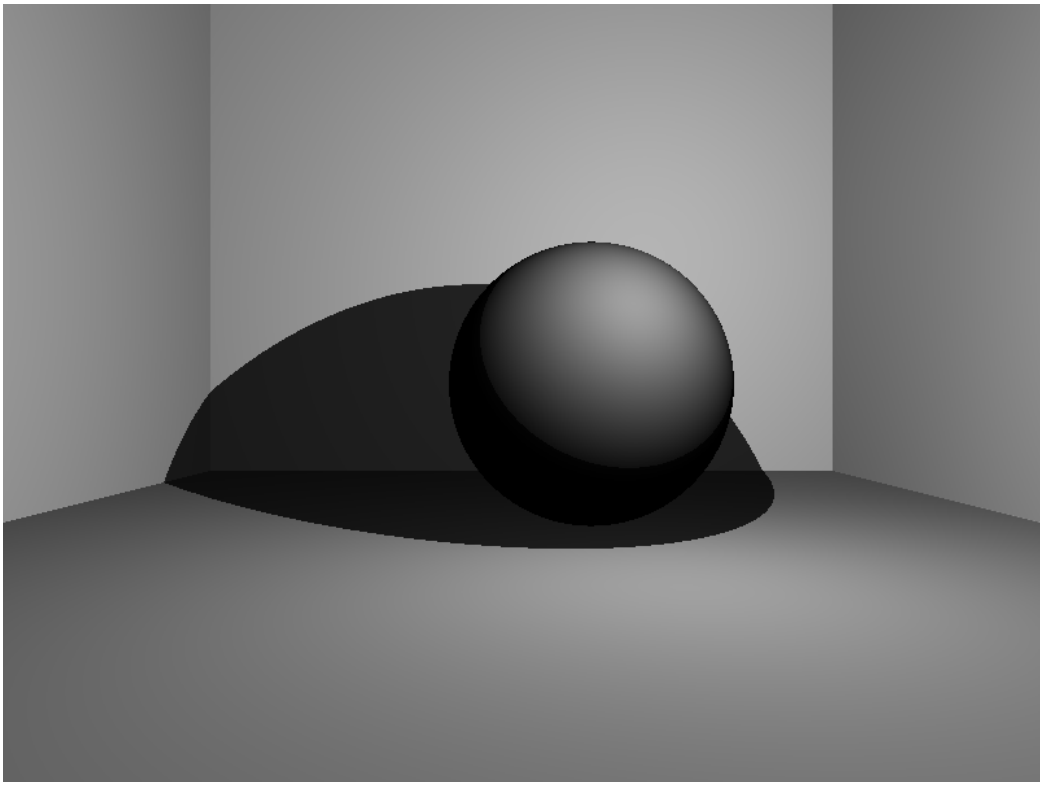
## 1.3 Primjeri

Slika 1 prikazuje jednostavnu scenu u kojoj se nalazi kugla omeđena plohama. Korištenjem miša možemo pomicati izvor svjetlosti i dobiti dozu interaktivnosti. Osim toga, kugla se pomiče u sceni po unaprijed zadanoj putanji (pomoću matematičkih funkcija).

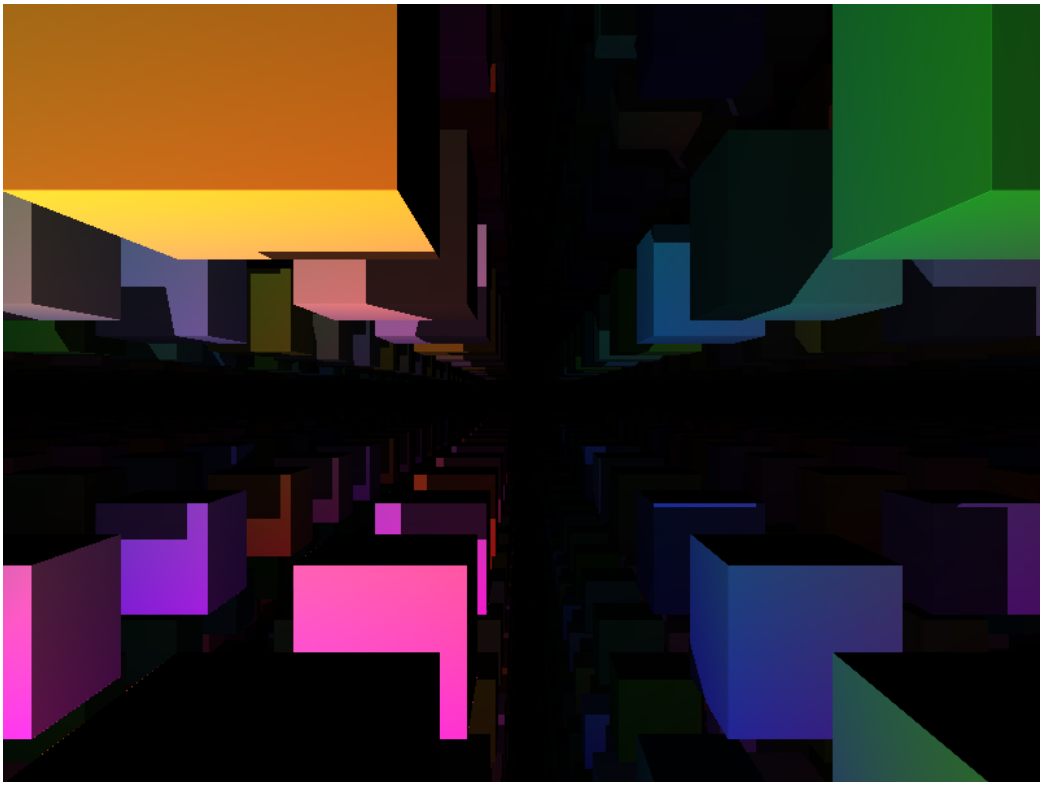
Na slici 2 možemo vidjeti "beskonačno" mnogo objekata/kocki u bojama. Kocke se nalaze na cjelobrojnim koordinatama, a tako su i obojane–boja se primjenjuje ovisno o položaju u prostoru. Slika 3 prikazuje primjer pokretanja gdje se u pozadini vidi kod sjenčara fragmenata za pripadni primjer.

Također je moguće funkcijama modelirati i drugačije, složenije objekte. Primjer s torusima prikazan je na slici 4, dok je apstraktniji primjer prikazan na slici 5.

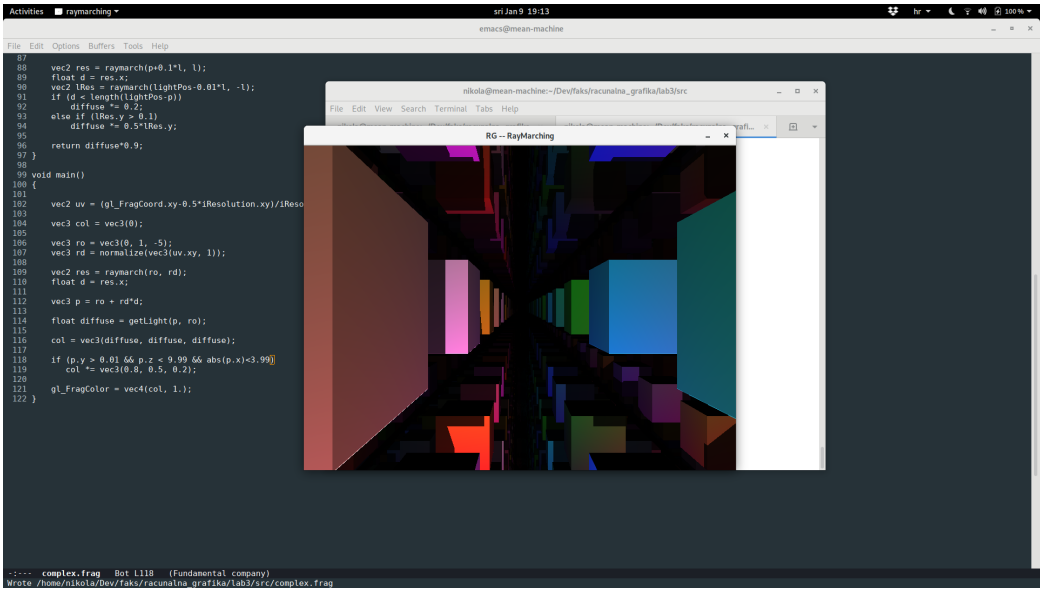
[3] https://www.libsdl.org/download-2.0.php



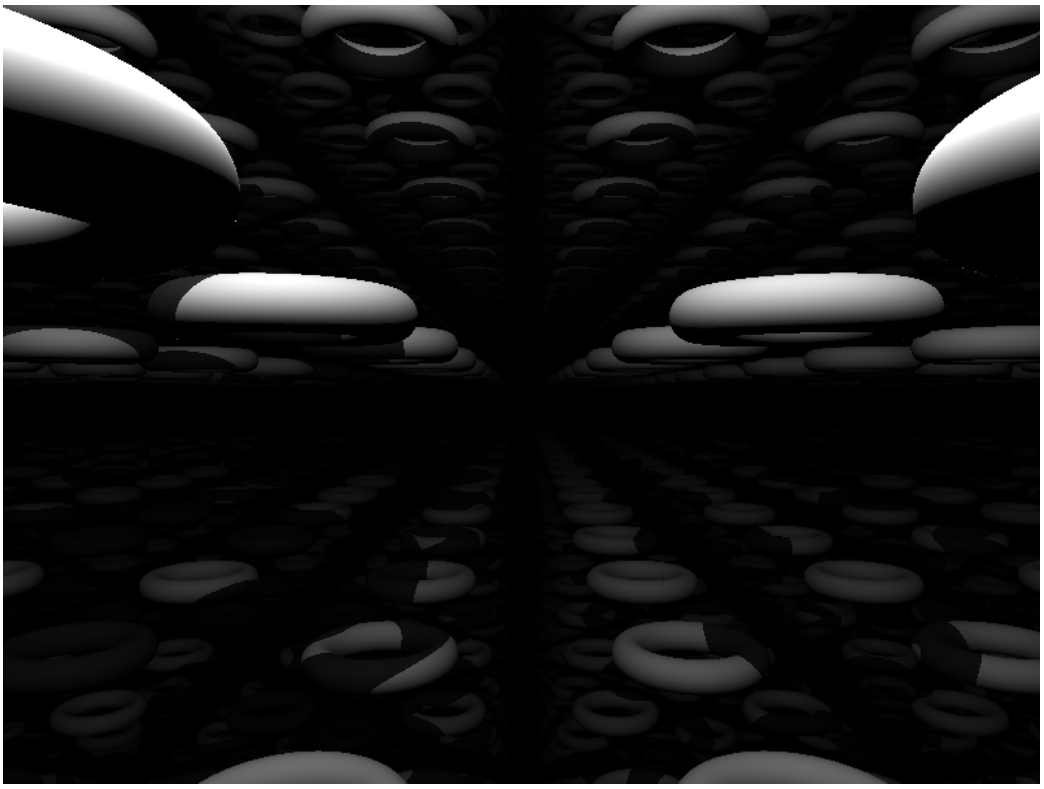
Slika 1: Jednostavna scena



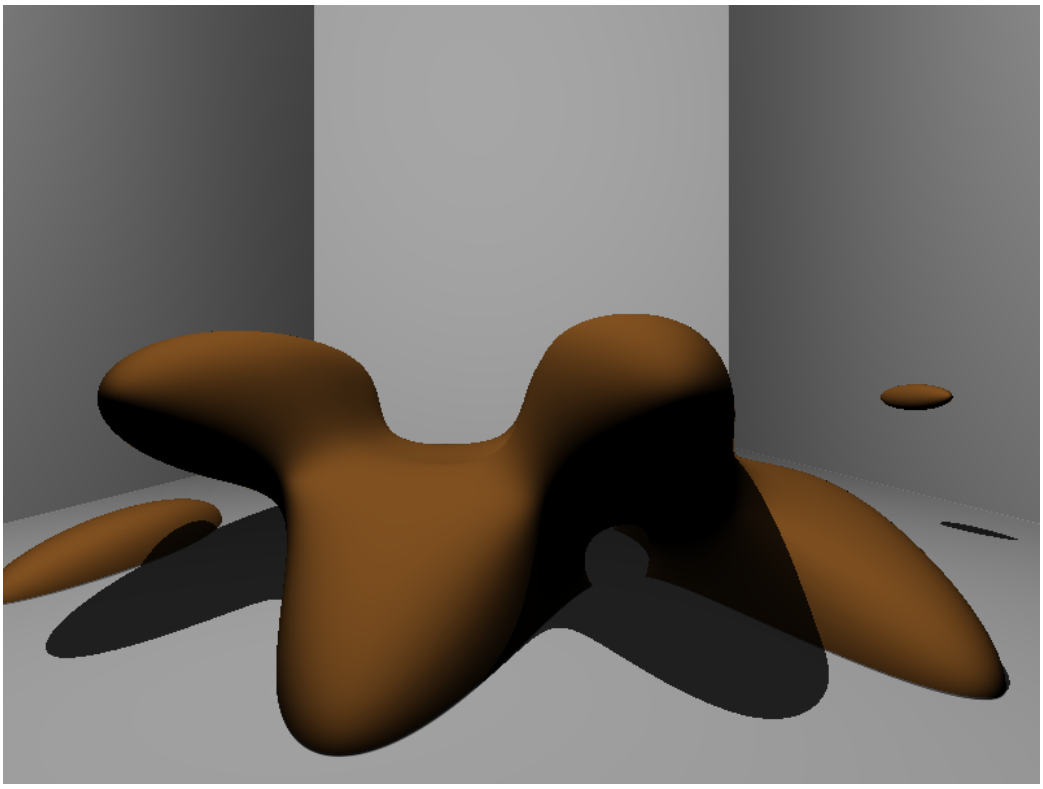
Slika 2: Puno objekata u boji




Slika 3: Primjer pokretanja programa



Slika 4: Primjer torusa



Slika 5: Scena s kompleksnim objektom