

# Sledenje žarku v neevklidskih prostorih

Blaž Bergant, Martin Jereb, Matjaž Pogačnik

Matematično modeliranje, maj 2024

## Contents

<b>1</b>	<b>Uvod</b>	<b>2</b>
<b>2</b>	<b>Cilji</b>	<b>2</b>
<b>3</b>	<b>Sledenje žarkom</b>	<b>2</b>
<b>4</b>	<b>Zasnova projekta in osnovni algoritem sledenja žarkov</b>	<b>3</b>
4.1	Objekti v sceni . . . . .	3
4.2	Prostori . . . . .	3
4.3	Generiranje žarkov . . . . .	4
4.4	Sledenje žarkom . . . . .	5
4.5	Računanje barve piksla . . . . .	6
<b>5</b>	<b>Evklidski prostor in potovanje žarka do luči</b>	<b>6</b>
<b>6</b>	<b>Sledenje žarkom na flat torusu <math>\mathbb{T}_{pl}^3</math></b>	<b>7</b>
6.1	Kaj je tri dimenzionalni flat torus? . . . . .	7
6.2	Algoritmi za flat torus . . . . .	7
6.3	Interpretacija rezultatov . . . . .	9
<b>7</b>	<b>Sledenje žarkom na dvodimenzionalni sferi <math>\mathbb{S}^2</math></b>	<b>9</b>
7.1	Algoritmi za 2-sphere . . . . .	13
7.2	Interpretacija rezultatov . . . . .	15
<b>8</b>	<b>Primerjava metode sledenje žarku v evklidskem in neevklidskem prostoru</b>	<b>17</b>
<b>9</b>	<b>Pohitritev algoritma</b>	<b>18</b>
9.1	Pohitritev osnovnega algoritma . . . . .	18
9.2	Pohitritev iskanja presečišč za euclidean in flat torus . . . . .	19
9.3	Pohitritev iskanja presečišč za 2Sphere . . . . .	20
<b>10</b>	<b>Zaključek</b>	<b>20</b>
<b>11</b>	<b>Razdelitev dela</b>	<b>21</b>
<b>12</b>	<b>Viri</b>	<b>21</b>
<b>13</b>	<b>Priloga</b>	<b>22</b>

# 1 Uvod

Algoritem sledenja žarku (*ang. Ray Tracing*) je priljubljen algoritem, uporabljen v računalniški grafiki, ki simulira realistično osvetlitev prizorov. Temelji na ideji sledenja žarka svetlobe po prizoru, izračunavanju njegovega preseka z objekti in odbijanju v drugo smer. Običajno ta algoritem uporabljamo v evklidskem prostoru, saj ta predstavlja resnični svet. Vendar pa lahko algoritem implementiramo tudi v drugih (neevklidskih) prostorih, da dobimo zanimive vizualne rezultate. To bo glavna naloga tega projekta.

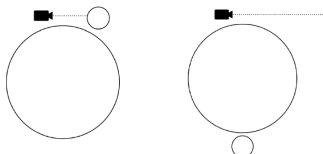
## 2 Cilji

Cilj za projektno nalogu, nam je bil narediti program, v katerem se vsak uporabnik (tudi tak, ki ne pozna izvirne kode) lahko vidi razliko med različnimi prostori ki niso evklidski, in jih primerja z evklidskim. Želeli smo si, da bi bilo dodajanje novih prostorov in novih objektov kar se le da enostavno in da na slike nebi bilo potrebno dolgo čakati.

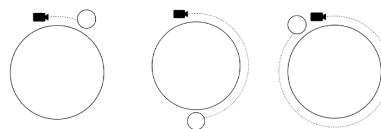
## 3 Sledenje žarkom

Predstavljajte si, da ste na vaši najljubši plaži, ki ima razgled na ocean. Ko se ozrete na barko, ki potuje proti sončnemu zahodu, boste poleg tega da se barka oddaljuje opazili tudi, da se "pogreza" v tla.

To je ravno zaradi potovanja žarkov. Ker smo v evklidskem prostoru, bo žarek potoval naravnost in nas zaradi ukrivljenosti Zemlje po določeni razdalji ne bo več dosegel. Sedaj pa si predstavljajmo, da bi lahko barko kljub oddaljevanju še vedno videli celotno. Ta posebna zmožnost bi nam bila omogočena v sferičnem neevklidskem prostoru.



Slika 1.1: Ilustracija evklidskega prostora.



Slika 1.2: Ilustracija neevklidskega sferičnega prostora.

Figure 1: Primerjava žarka v evklidskem in sferičnem prostoru [Vir: G. Kovač]

Kaj pa pravzaprav je neevklidski prostor? To je prostor, ki ne sledi postulatom evklidske geometrije. Torej prostor ni raven ampak je ukrivljen, vsota kotov v trikotniku je večja ali pa manjša od 180 stopinj. Neevklidski prostor delimo na več različnih geometrij med katerimi so hiperbolična geometrija, eliptična geometrija, sferična geometrija, projektivna geometrija itd.

## 4 Zasnova projekta in osnovni algoritem sledenja žarkov

Želeli smo si načina podajanja objektov in prostorov, ki bi bil dovolj splošen, da lahko brez spremnjanja algoritmov dodamo nove.

### 4.1 Objekti v sceni

Posamičen objekt je podan z enačbo, ki sprejme točko v prostoru in vrne vrednost. Enačba je torej oblike

$$o(x, y, z) : \mathbb{R}^3 \rightarrow \mathbb{R} = a$$

kjer je  $a = 0$  v primeru, da je točka na ploskvi objekta in  $a \neq 0$  v primeru, da je točka v notranjosti ali zunanjosti objekta. Za kroglo s središčem v  $(x_0, y_0, z_0)$  in polmerom  $r$  je to enačba

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0,$$

Za ravnino z normalo  $\vec{n} = [a, b, c]^T$  in točko na ravnini  $(x_0, y_0, z_0)$  pa:

$$ax + by + cz - (ax_0 + by_0 + cz_0) = 0.$$

### 4.2 Prostori

Vsak prostor implementira funkcijo v obliki

$$s(t, T, \vec{v}) : \mathbb{R} \rightarrow \mathbb{R}^3 = [x, y, z]^T$$

ki vrne točko v prostoru, kjer bi bil žarek, ki gre skozi točko  $T$  v smeri  $\vec{v}$ , če ga premaknemo za razdaljo  $t$  po tem prostoru. V prostoru 2-Sphere funkcija računa približek, v primeru evklidskega prostora pa je to preprosto premik po premici za razdaljo  $t$

$$\vec{T}_{n+1} = \vec{T}_n + t \cdot \vec{v}$$

Prostora flat torus in 2-sphere sta podrobnejše opisana kasneje.

### 4.3 Generiranje žarkov

Žarek je definiran z začetno točko  $T_0$  in enotskim vektorjem  $\vec{v}$ . Ker smo si želeli, da širino kadra (ang. Field of View) podamo v stopinjah kota, ki ga kamera vidi po diagonali kadra, smo enotske vektorje  $v$  generirali po postopku:

1. Vsi žarki, imajo začetno točko  $T_0$  na poziciji kamere
2. Iz  $FOV$  izračunamo  $FOV_X$  (širino kadra po horizontali),  $FOV_Y$  (širino kadra po vertikali)
3. Ker so piksli slike štirikotni, lahko zamik v smeri horizontalni in vertikalni smeri izračunamo kot:
 
$$\Delta x = \Delta y = \frac{FOV_X}{RES_X}$$
 kjer je  $RES_X$  širina slike v pikslih.
4. Žarke predstavimo kot točke na enotski sferi, ki je usmerjena v smer v katero kaže kamera. Vsak žarek, ima azimut

$$\theta = -FOV_X/2 + i \cdot \Delta x$$

in elevacijo

$$\phi = FOV_Y/2 - j \cdot \Delta x,$$

kjer je  $i$  stolpec in  $j$  vrstica slike (šteta po računalničarsko, torej začnemo z 0).

5. Korektiramo distorcojo žarkov, ki se pojavi, ker sta točki na enotski sferi, ko je  $\phi$  blizu 0 in rotiramo  $\theta = \theta + \Delta x$  veliko bolj oddaljeni, kot ko je  $\phi$  blizu  $\pi/2$  (kjer bi za vsak  $\theta$  dobili enako točko)

$$\phi_{true} = \phi \cdot \cos(\theta)$$

6. Enotski vektor  $\vec{v}$  dobimo iz azimuta in elevacije kot:

$$\vec{v} = [\cos(\theta) \cos(\phi), \sin(\phi), \cos(\theta) \sin(\phi)]^T$$

7. Vektor  $v$  s pomočjo ortogonalnih matrik rotiramo v smer kamere.

Postopek se zdi kar zapleten, vendar je proizvedel bolj natančne rezultate kot druge metode, ko kamera ne gleda naravnost, npr. mreža točk za piksle nekoliko pred kamero, kjer za vsak piksel generiramo žarek med kamero in točko na mreži. Omogoča nam tudi določanje širine kadra v stopinjah, kar je zelo uporabno.

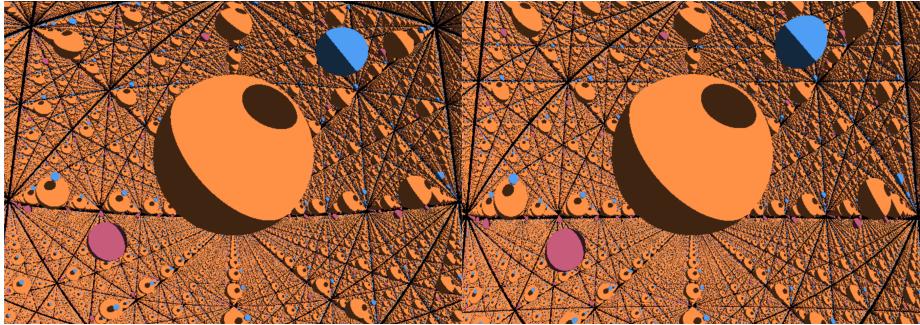


Figure 2: Flat torus z distorzijo

Figure 3: Flat torus brez distorzije

Zgoraj je primerjava slike prostora flat torus, kjer je širina kadra ekvivalentna 14mm objektivu. Na levi je prikazana slika, ki ne upošteva popravka za distorzijo (korak 5), na desni pa je slika, ki ga upošteva.

Opazimo lahko, da so horizontalne črte na desni sliki dejansko horizontalne, medtem ko so na levi sliki ukrivljene. Ker je kader usmerjen za 10 stopinj navzgor, se vertikalne črte na robovih ukrivijo navznoter, kar je pojav, ki se zgodi tudi na pravih kamerah.

#### 4.4 Sledenje žarkom

Osnovni algoritem sledenja žarkov je sledeč:

1. V poljubnem prostoru premikamo žarek s pomočjo enačbe  $s(t) = [x, y, z]^T$  za korak velikosti  $h$ .
2. Za vsak objekt, s pomočjo njegove enačbe  $o(x, y, z) = a$  preverimo, če je žarek prešel skozi ploskev objekta (če je  $a$  spremenil predznak).
3. Če je za nek objekt  $a$  spremenil predznak, razpolovimo korak  $h = h/2$  in ponovimo korak.
4. Postopek razpolavljanja ponavljamo, dokler korak ni manjši od  $\varepsilon$ .

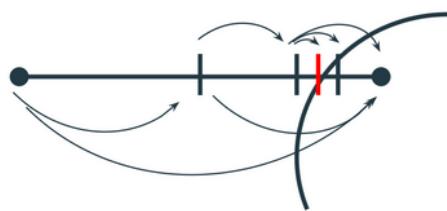


Figure 4: Iskanje natančnejšega presečišča

## 4.5 Računanje barve piksla

Vsak žarek obarvamo z barvo objekta s katerim se je sekal, oziroma črno, če se ni z nobenim. Za senčenje moramo poslati še en žarek od presečišča proti luči.

1. Poiščemo vektor, ki gre v smeri od presečišča do luči:

$$\vec{v} = \frac{\vec{r}_l - \vec{r}_p}{\|\vec{r}_l - \vec{r}_p\|}$$

Kjer je  $\vec{r}_l$  krajevni vektor luči,  $\vec{r}_p$  pa krajevni vektor presečišča.

2. Če žarek, ki se začne v točki presečišča in gre v smeri  $\vec{v}$ , seká kakšen objekt preden prepotuje razdaljo do luči  $d = \|\vec{r}_l - \vec{r}_p\|$ , potem je presečišče v senci.

## 5 Evklidski prostor in potovanje žarka do luči

V evklidskem prostoru je potovanje žarkov preprosto premik po premici:

$$\vec{T}_{n+1} = \vec{T}_n + t \cdot \vec{v}$$

Žarki potujejo naravnost, kot smo navajeni. Algoritem sledenja žarkov je enak kot v primeru ploščatega torusa, le da žarkov ní potrebno preslikati.

Odločili smo se, da bodo žarki od presečišča z objektom do luči potovali kar po evklidskem prostoru. To odločitev smo sprejeli, ker bi bilo v neevklidskih prostorih zelo težko implementirati algoritem, ki žarek pošlje po vseh poteh, ki vodijo do vsaj ene luči. Zadovoljiti bi se morali z algoritmom, ki žarke pošilja v razne smeri in upa, da bo vsaj en žarek prišel do luči, če pot do nje obstaja. Implementacija tega predstavlja izboljšavo, ki bi jo lahko dodali v prihodnosti.

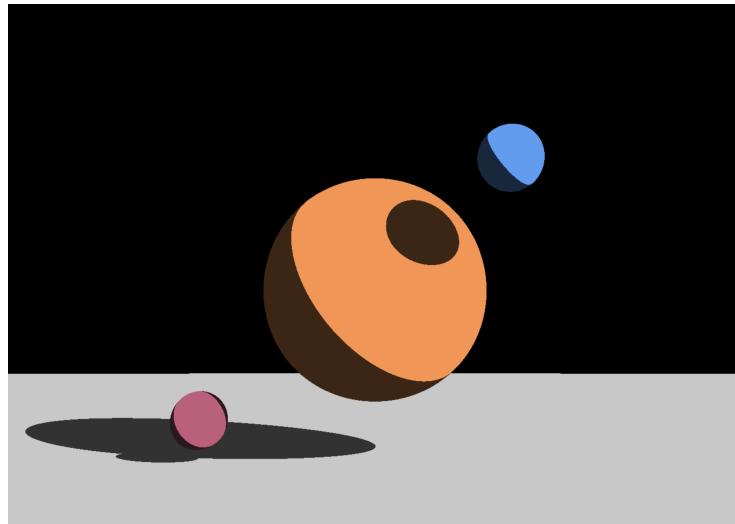


Figure 5: Evklidski prostor

Na sceni kamera gleda v krogle obrnjena za 10 stopinj navzgor. Enaka scena bo uprizorjena tudi v drugih prostorih.

## 6 Sledenje žarkom na flat torusu $\mathbb{T}_{pl}^3$

### 6.1 Kaj je tri dimenzionalni flat torus?

Tri dimenzionalni flat torus  $\mathbb{T}_{pl}^3$  je prostor, ki ga dobimo tako, da "zlepimo" nasprotne ploskve enotske kocke  $[0, 1] \times [0, 1] \times [0, 1] \subset \mathbb{R}^3$ , kjer je  $\mathbb{R}^3$  tridimenzionalni evklidski prostor. Formalno pravimo, da je  $\mathbb{T}_{pl}^3$  kvocient  $\mathbb{R}^3$  z grupo translacij:

$$(x, y, z) \rightarrow (x \pm 1, y, z), \quad (x, y, z) \rightarrow (x, y \pm 1, z), \quad (x, y, z) \rightarrow (x, y, z \pm 1).$$

Običajni torusi so ukrivljeni in se jim ukrivljenost po površini spreminja. Flat torus pa ima konstantno ukrivljenost v vsaki točki. Poimenovanje flat izhaja iz tega, ker je povsod podoben navadni evklidski ravnini. Podobno velja za tridimenzionalni flat torus, kjer pri sledenju žarka lahko uporabimo kar osnovno definicijo žarka. Ključna razlika med algoritmom v  $\mathbb{R}^3$  in  $\mathbb{T}_{pl}^3$  pa je, da bomo tokrat dodali še en korak, ker moramo žarek omejiti na kocko. Dokler se nahajamo znotraj njenih meja, ne naredimo nobenih popravkov. Ko pa ugotovimo presečišče z eno od ploskev, moramo žarek preslikati na nasprotno ploskev. To lahko storimo tako, da preprosto žarek omejimo znotraj domene, izračunamo presežek in ga prištejemo začetku domene.

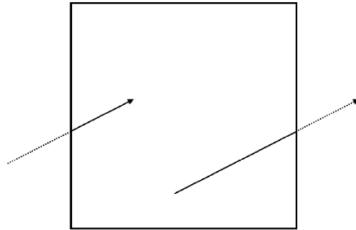


Figure 6: Žarek v dvodimenzionalnem flat torusu

### 6.2 Algoritmi za flat torus

Ključni vpogled, je da žarek omejimo na domeno. V spodnji funkciji map-ToCube je  $T_n$  začetni vektor, a je začetek domene, b konec domene.

---

**Algorithm 1** Map to Cube

---

```
1: function MAPTOCUBE( $T_n$ ,  $a$ ,  $b$ )
2:    $range \leftarrow b - a$ 
3:    $q \leftarrow (T_n - a) / range$ 
4:    $fraction \leftarrow q - \text{floor}(q)$ 
5:    $T_{n+1} \leftarrow a + fraction \cdot range$ 
6:   return  $T_{n+1}$ 
7: end function
```

---

Funkcija **traceRayFlatTorus** sprejme začetno točko  $T_0$ , smer  $\vec{v}$ , korak step, maksimalno število iteracij maxIt, množico objektov objects in meje  $a$  in  $b$  domene na katero bomo preslikovali koordinate žarka. Pomožni funkciji **signs** in **checkIntersect** implementirata funkcije objektov  $o(x, y, z)$ .

---

**Algorithm 2** Sledenje žarku na flat torusu

---

```
function TRACERAYFLATTORUS( $T_0$ ,  $\vec{v}$ , step, maxIt, objects, a, b,  $\varepsilon$ )
   $T_n, T_{n+1} \leftarrow T_0$ 
  signs  $\leftarrow \text{signs}(T_n, \text{objects})$   $\triangleright$  izračunamo začetne predznaake

  t  $\leftarrow 0$ 
  while true do
    if  $t == \text{maxIt}$  then
      return -1
    else if  $step < \varepsilon$  then
      return  $T_n$ 
    end if
     $T_{n+1} \leftarrow T_n + step \cdot \vec{v}$ 
     $T_{n+1} \leftarrow \text{remap}(T_{n+1}, a, b)$   $\triangleright$  naredimo korak, preslikamo na željeno domeno

    signs  $\leftarrow \text{checkIntersects}(T_{n+1}, \text{objects}, \text{signs})$ 
           $\triangleright$  preverimo, če smo sekali objekt
           $\triangleright$  če da, razpolovimo korak

    if object  $\neq \text{NULL}$  then
      step  $\leftarrow step/2$ 
    else
       $T_n \leftarrow T_{n+1}$ 
      t  $\leftarrow t+1$ 
    end if
  end while
end function
```

---

### 6.3 Interpretacija rezultatov

Ko program poženemo, dobimo lep vizualni rezultat.

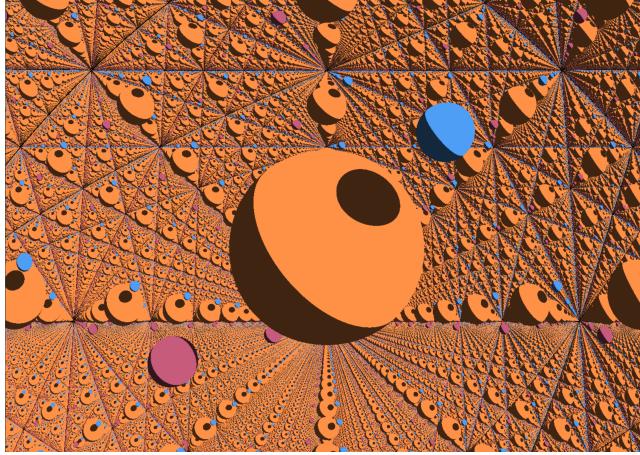


Figure 7: Rezultat "ray tracinga" za tri dimenzionalni flat torus (500 preslikanj), kjer scena nima ravnine.

Na sliki lahko opazimo podoben efekt tistemu v hišah iluzij, ko smo v sobi polni ogledal. Razlika je v tem, da se tu žarki ne odbijejo, temveč "teleportirajo" na drugo stran scene, zato slika ni simetrična. Na levi strani slike lahko opazujemo sceno, kot bi izgledala, če bi jo gledali bolj z desne strani, na desni pa kot da bi jo gledali z leve.

Izrišejo se tudi distinktne črte. Nekateri žarki, ki so kljub velikemu številu preslikavanj (v tem primeru 500) še vseeno črni, se ne bodo nikoli srečali z objektom, ker gredo v smer, ki je povsem vzporedna z objekti v "sosednjih" scenah.

## 7 Sledenje žarkom na dvodimenzionalni sferi $\mathbb{S}^2$

Žarek na dvodimenzionalni sferi bo potoval po najravnejši poti, tj. geodetki, kjer je majhen korak v  $uv$  ravnini opisan s sistemom dveh diferencialnih enačb drugega reda.

$$\begin{aligned} \frac{d^2u}{dt^2} - \cos(u) \sin(u) \frac{dv}{dt} \frac{dv}{dt} &= 0 \\ \frac{d^2v}{dt^2} + 2 \cot(u) \frac{du}{dt} \frac{dv}{dt} &= 0 \end{aligned} \tag{1}$$

Po krivulji se bomo v nadalnjih algoritmih premikali s pomočjo aproksimacijskih metod, zato moramo sistem DE drugega reda preoblikovati v sistem DE

prvega reda. Za to uvedemo 4 nove spremenljivke

$$\begin{aligned} y_1 &= u, & y_2 &= \frac{dy_1}{dt}, \\ y_3 &= v, & y_4 &= \frac{dy_3}{dt} \end{aligned} \tag{2}$$

Dobimo sistem 4 DE prvega reda

$$\begin{aligned} \frac{dy_1}{dt} &= y_2 \\ \frac{dy_2}{dt} &= \cos(y_1) \sin(y_1) y_4^2 \\ \frac{dy_3}{dt} &= y_4 \\ \frac{dy_4}{dt} &= -2 \cot(y_1) y_2 y_4 \end{aligned} \tag{3}$$

Ker se bomo premikali po  $uv$  ravnini, moramo koordinate v kartezičnem koordinatnem sistemu transformirati v parametra  $u$  in  $v$ . Sfera v  $\mathbb{R}^3$  je parametrizirana kot

$$\begin{aligned} X &= R \cos(v) \sin(u) \\ Y &= R \sin(v) \sin(u) \\ Z &= R \cos(u) \end{aligned} \tag{4}$$

$u$  lahko dobimo kot

$$u = \cos^{-1} \left( \frac{Z}{R} \right) \tag{5}$$

za  $v$  pa uporabimo prvi dve enačbi

$$\begin{aligned} \frac{R \sin(u) Y}{R \sin(u) X} &= \frac{\sin(v)}{\cos(v)} \\ v &= \tan^{-1} \left( \frac{Y}{X} \right) \end{aligned} \tag{6}$$

Za sledenje žarku po geodetki, si moramo poleg začetne točke izbrati še začetno smer  $(\frac{du}{dt}, \frac{dv}{dt})$ . Tako kot pri ostalih prostorih v nalogi, bomo uporabili perspektivno projekcijo, kar pomeni da vsi žarki izvirajo iz ene točke in niso paroma vzporedni. Od tu dalje imamo svobodo pri izbiri sfer, po katerih bodo žarki potovali. V tej nalogi smo za vsak žarek izračunali središče sfere z najnižjo  $z$  koordinato in polmerom  $R$ . Žarki bodo vedno "zavijali" navzdol, torej se bodo premikali po poldnevnikih.

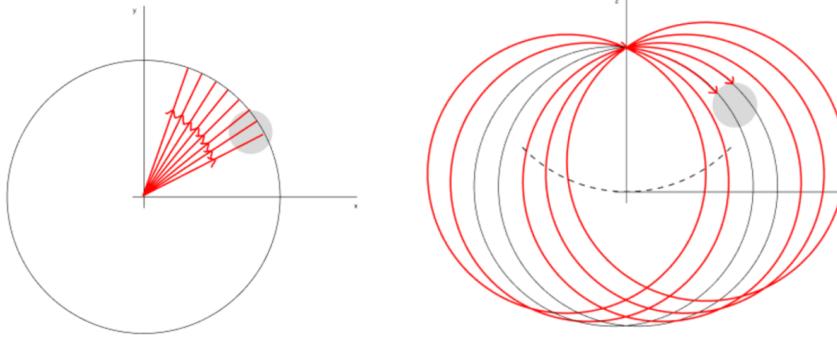


Figure 8: Potovanje žarkov od zgoraj

Figure 9: Potovanje žarkov s strani

Središča sfer žarkov se bodo torej nahajala na sferi s polmerom  $R$  in središem v kameri  $T_0$ . Za različne smeri žarkov  $\vec{v}_i$  je bilo središče  $C_i$  izračunano po sledečem postopku:

Izračunamo vektorski produkt med navpičnim vektorjem in smerjo  $\vec{v}_i$ . Dobljeni vektor je normala na ravnino krožnice z iskanim središčem, ki gre skozi  $T_0$

$$\vec{n}_i = (0, 0, -1) \times \vec{v}_i \quad (7)$$

Vektor, ki od  $T_0$  kaže proti iskanemu središču, bo ležal v tej ravnini, poleg tega pa bo pravokoten na  $\vec{v}_i$ . Ponovno uporabimo vektorski produkt

$$\vec{c}_i = \vec{v}_i \times \vec{n}_i \quad (8)$$

Krajevni vektor središča nato dobimo tako, da se premaknemo za polmer  $R$  v smeri  $\vec{c}_i$

$$\vec{r}_{ci} = \frac{\vec{c}_i}{\|\vec{c}_i\|} \cdot R + T_0 \quad (9)$$

Za korak po geodetki bomo  $T_0$  najprej premaknili za vektor  $\vec{r}_{ci}$ , tako da bo središče sfere v koordinatnem izhodišču. Nato naredimo korak, novo točko  $T_{i1}$  pa premaknemo nazaj za vektor  $\vec{r}_{ci}$ . Ker želimo potovati po poldnevnikih, bomo za začetno smer  $(\frac{du}{dt}, \frac{dv}{dt})$  izbrali  $(\pm 1, 0)$ . Korak po poldnevniku bo v  $xy$  ravnini tako že pravilno obrnjen, določiti pa moramo predznak premika po  $u$  glede na to, ali se premikamo v pravo smer  $\vec{v}_i$ . To preverimo s skalarnim produktom:

Naredimo korak v smeri  $(1, 0)$  in označimo dobljeno točko s  $T_t$ . Vektor v smeri od  $T_0$  do  $T_t$  označimo z  $\vec{v}_t$ . Izračunamo skalarni produkt

$$a = \vec{v}_i \vec{v}_t^T \quad (10)$$

Če se premikamo v pravi smeri, bo  $a > 0$ , drugače moramo za začetno smer izbrati negativni predznak.

V nadaljevanju se premikamo po geodetki z eno izmed aproksimacijskih metod.  
V splošnem lahko za take sisteme uporabimo Eulerjevo metodo

$$\begin{aligned} t_{n+1} &= t_n + h \\ \vec{y}_{n+1} &= \vec{y}_n + h \cdot \vec{f}(t_n, \vec{y}_n) \end{aligned} \tag{11}$$

problem pa se pojavi pri  $\cot(y_1)$  v eni izmed enačb, ki ima pole pri  $k\pi$ ,  $k \in \mathbb{Z}$ . Da se temu izognemo, ne smemo uporabljati fiksne dolžine koraka, temveč moramo uporabiti adaptivne metode, kot je DOPRI5, ki z uporabo večih metod estimira napako približka in temu primerno prilagodi korak.

Algoritmi, ki implementirajo opisane metode so predstavljeni v naslednjem poglavju.

## 7.1 Algoritmi za 2-sphere

Funkcija **traceRay2Sphere** sprejme začetko točko  $T_0$ , smer  $\vec{v}$ , velikost koraka step, maksimalno število iteracij maxIt, množico objektov objects, polmer sfere  $\mathbb{S}^2$  in natančnost presečišča  $\varepsilon$ . Pomožna funkcija **sphereCenter** implementira formule (7), (8), (9), **initializeSphere** formule (3), (5), (6), **uvToVec** pa formulo (4). Funkcija **DOPRI5** predstavlja implementacijo DOPRI5, ki naredi po en korak glede na trenutno velikost koraka, in skupaj s približkom vrne tudi novo velikost koraka. Tu lahko uporabimo tudi katero drugo adaptivno funkcijo. Ostale funkcije so opisane v razdelku

---

**Algorithm 3** Sledenje žarku na sferi  $\mathbb{S}^2$

---

```

function TRACERAY2SPHERE( $T_0$ ,  $\vec{v}$ , step, maxIt, objects, R,  $\varepsilon$ )
     $T_n, T_{n+1} \leftarrow T_0$ 
    signs  $\leftarrow \text{signs}(T_n, \text{objects})$                                  $\triangleright$  izračunamo začetne predznaake
    center, I

    t  $\leftarrow 0$ 
    while true do
        if t == maxIt then
            return -1
        else if step == 0 then
            center  $\leftarrow \text{sphereCenter}(T_0, d, R)$                        $\triangleright$  žarku poiščemo središče sfere
             $T_m \leftarrow T_0 - \text{center}$                                           $\triangleright$  sfero premaknemo v  $(0, 0)$ 
             $\vec{y}_n \leftarrow \text{initializeSphere}(T_0, R)$                           $\triangleright$  pripravimo začetni  $\vec{y}$ 
        end if

         $\vec{y}_{n+1}, \text{step} \leftarrow \text{DOPRI5}(\vec{y}_n, \text{step})$                    $\triangleright$  korak po geodetki
         $T_{n+1} \leftarrow \text{uvToVec}(\vec{y}_{n+1}, R) + \text{center}$ 

        object  $\leftarrow \text{checkIntersects}(T_{n+1}, \text{objects}, \text{signs})$ 
                $\triangleright$  preverimo, če smo sekali objekt
                $\triangleright$  če da, poiščemo natančno presečišče

        if object != NULL then
            I  $\leftarrow \text{findIntersection}(T_n, \text{object}, \text{signs}, \vec{y}_n, \text{step}, R, \text{center}, \varepsilon)$ 
            return I
        else
             $\vec{y}_n \leftarrow \vec{y}_{n+1}$ 
             $T_n \leftarrow T_{n+1}$ 
            t  $\leftarrow t+1$ 
        end if
    end while
end function

```

---

---

**Algorithm 4** Iskanje natančnejšega presečišča

---

```
function FINDINTERSECTION( $T_0$ , object, signs,  $\vec{y}_n$ , step, R, center,  $\varepsilon$ )
     $T_n, T_{n+1} \leftarrow T_0$ 

    while true do
        if step <  $\varepsilon$  then
            return  $T_n$ 
        end if
         $\vec{y}_{n+1}, \text{step} \leftarrow \text{euler}(\vec{y}_n, \text{step})$ 
         $T_{n+1} \leftarrow \text{uvToVec}(\vec{y}_{n+1}, R) + \text{center}$ 
             $\triangleright$  korak po geodetki z Eulerjevo metodo
        object  $\leftarrow \text{checkIntersects}(T_{n+1}, \text{object}, \text{signs})$ 
             $\triangleright$  preverimo, če smo sekali objekt
             $\triangleright$  če da, razpolovimo korak
        if object != NULL then
            step  $\leftarrow \text{step}/2$ 
        else
             $\vec{y}_n \leftarrow \vec{y}_{n+1}$ 
             $T_n \leftarrow T_{n+1}$ 
        end if
    end while
end function
```

---

## 7.2 Interpretacija rezultatov

Za prikaz značilnosti slik generiranih na dvodimenzionalni sferi  $\mathbb{S}^2$  smo generirali sliko scene z dvema različno velikima kroglama, kjer je vir svetlobe "skrit" za eno izmed krogel. Na sliki žarkov vidimo dva snopa žarkov z zaporednimi rdečimi točkami, ki predstavljajo korake. Podobno so z modrimi točkami prikazani žarki proti viru svetlobe. Na generirani sliki so piksli, katerim pripadata snopa žarkov, označeni z rdečo. Opazujemo moder objekt na generirani sliki, ki mu pripada manjša izmed krogel, ki se na generirani sliki pojavi v dveh delih, kjer je en delno osvetljen. Rezultat je pričakovani, saj bodo do neke meje zgornji žarki zaokrožili do krogle in jo tako zadeli iz spodnje strani. Ta stran je tudi edina, ki je delno osvetljena, kar lahko vidimo iz uspešne nadaljnje poti žarkov proti viru svetlobe. Ko se premikamo nižje po stolpcu bodo žarki obhajali kroglo, dokler ne bomo dovolj nizko, da jo spet direktno zadanemo. Tokrat žarki ne bodo uspeli priti do vira svetlobe.

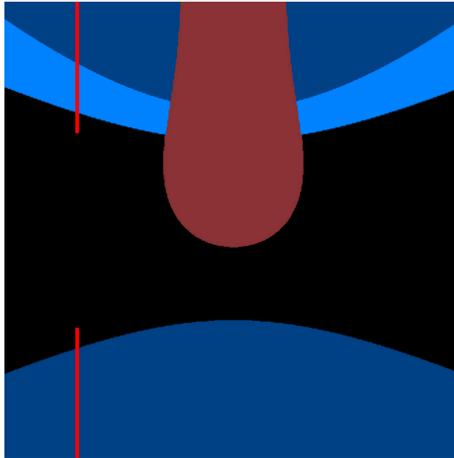


Figure 10: Generirana slika

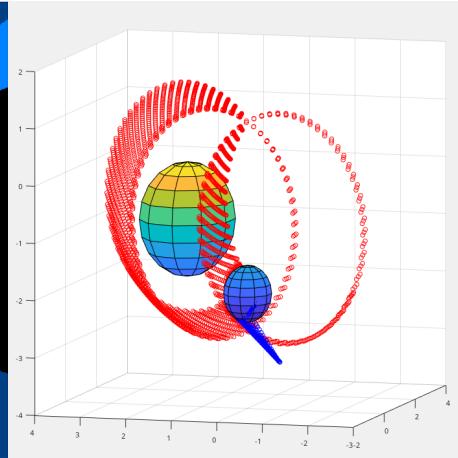


Figure 11: Slika snopov žarkov

Če bi spremljali generirane slike ob premikanju kamere, bi videli, da objekti ki so naravnost pred nami pogosto v sliko vstopajo od zgoraj, kar je prav tako pričakovano. Zgornji žarki bodo imeli namreč najdaljši doseg, saj je središče njihove sfere premaknjeno najdlje stran od kamere v smeri premikanja. Na sekvenci slik se mimo rdeče krogle premikamo proti modri krogli, ki je v celoti osvetljena. V ozadju vidimo oranžno ravnilo.

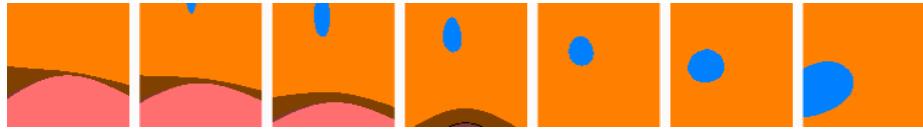


Figure 12: Zaporedje slik s premikanjem kamere

Sceno iz sekvence lahko vidimo na sliki žarkov, kjer je prikazan snop žarkov ene vrstice pikslov.

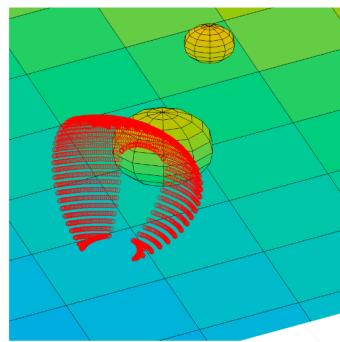


Figure 13: Žarki vrstice pikslov

Pomembno je omeniti tudi, da v primeru velikih radijev sfer, po katerih se premikajo žarki, prostor izgleda manj popačeno 14. Slike scene v neevklidskih prosorih vidimo na 20 in 19. V prostoru 2sphere, kjer je radij sfere 25, izgleda skoraj kot da bi kamero usmerili v tla. To je ponovno posledica dejstva, da objekti pred nami v sliko vstopajo od zgoraj. Opazimo pa tudi, da je krogla na sliki uprizorjena, kot da bi jo gledali z višje točke, kar je posledica ukrivljanja žarkov navzdol.

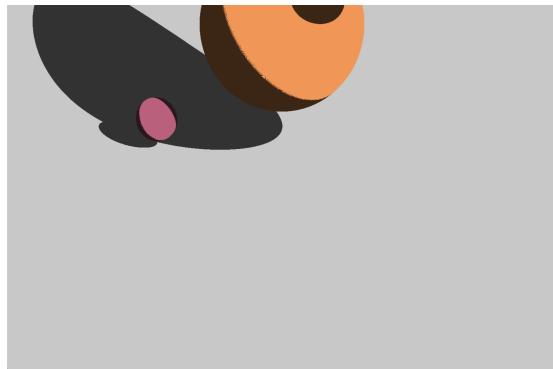


Figure 14: Render, kjer je radij krogle 25

## 8 Primerjava metode sledenje žarku v evklidskem in neevklidskem prostoru

Primerjamo enako pozicijo kamere in enako postavitev objektov v prostoru v evklidskem in dveh neevklidskih prostorih. Postavitev objektov je enaka kot na sliki žarkov. Za razliko od evklidskega prostora pri torusu opazimo podvojene objekte, kjer pričakujemo, da bi z večjim številom korakov bilo čedalje manj črnih piksov. Pri dvodimensionalni sferi, pri dani postavitvi modre krogle še ne vidimo, rdeča pa je precej popačena. Ploskev pod kroglama je nagnjena, zato pri dvodimensionalni sferi opazimo, da je spodnji žarki ne zadanejo več.

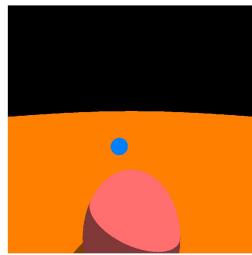


Figure 15: Evklidski prostor

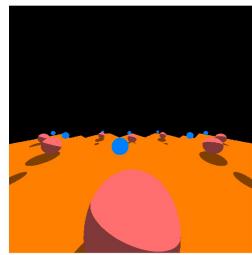


Figure 16: flat torus

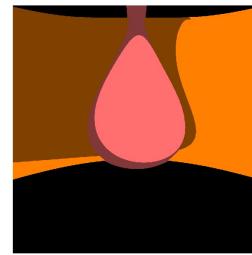


Figure 17: Dvodimensionalna sfera

## 9 Pohitritev algoritma

Da ne bi bilo potrebno na vsako sliko čakati predolgo časa, lahko algoritme pohitrimo.

### 9.1 Pohitritev osnovnega algoritma

Opazimo naslednji težavi:

1. Korakanje žarka po prostoru hitro postane računsko prezahtevno, če je korak majhen.
2. Če je korak prevelik, se lahko zgodi, da v neki točki preskoči presečišče z objektom.

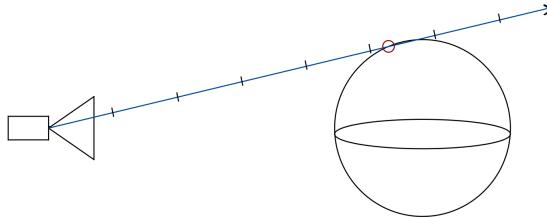


Figure 18: Zaradi prevelikega koraka, zgrešimo presečišče.

Še enkrat se spomnimo, kaj je naš problem. Iščemo najmanjši parameter  $t$ , za katerega velja:

$$f = o(s(t)) = 0$$

$o([x, y, z]) = a$  je funkcija (kateregakoli) objekta v sceni, ki je enaka 0 samo, ko je točka  $[x, y, z]$  na objektu.  $s(t, T, \vec{v}) = [x, y, z]$  je funkcija, ki vrne točko v prostoru, kjer se nahaja žarek z izhodiščem v  $T_0$  in smerjo  $\vec{v}$  ob času  $t$ .

Da bolje razumemmo problem, si poglejmo par grafov, ki prikazujejo kako se "razdalja" do oranžne krogle (vrednost funkcije  $o(t)$ ) spreminja glede na razdaljo žarka od kamere (vrednost  $t$ ) v evklidskem prostoru in prostoru 2sphere. V sredini slike je obkrožen piksel katerega žarek spremljamo.

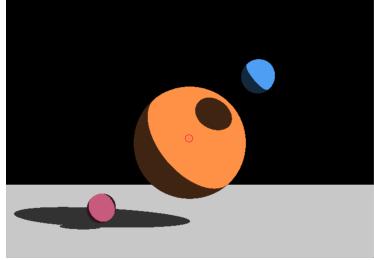


Figure 19: Scena, ki jo bomo pospeševali v evklidskem prostoru

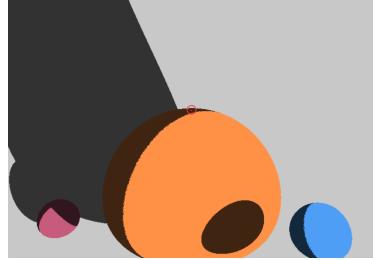


Figure 20: Scena, ki jo bomo pospeševali na 2Sphere

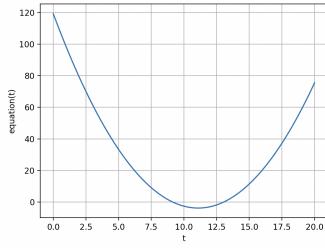


Figure 21: Graf  $o(t)$  za evklidski prostor

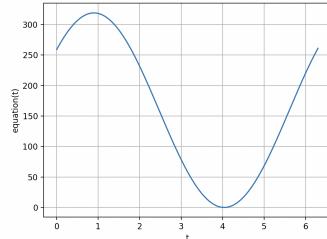


Figure 22: Graf  $o(t)$  za 2Sphere

Hitro nam postane jasno, da razdalja žarka sledi funkciji, ki ima 0, 1, ali 2 ničli (za 2sphere iščemo le na intervalu  $t \in [0, 2\pi]$ ). Imamo le en minimum za katerega velja, da če je večji od 0, se žarek objekta nikoli ne bo dotaknil.

Implementirali smo naslednje izboljšave osnovnega algoritma:

- Ko opazimo, da se funkcija  $o(t)$  začne povečevati, po tem ko je že padala, razpolovimo korak, da preverimo če smo preskočili presečišče
- Dokler se funkcija  $o(t)$  ne povečuje, večamo korak, da hitreje pridemo do presečišča.

## 9.2 Pohitritev iskanja presečišč za euclidean in flat torus

Ideja je preprosta. Zakaj bi delali iterativne korake, če lahko preprosto izpeljemo formulo za izračun parametra  $t$  za katerega velja  $f = 0$ ? Tako presečišče najdemo v času odvisnem le od števila objektov v sceni.

Flat torus uporablja kar formulo za presečišče iz evklidskega prostora, le da žarke še preslika, če pridejo do roba torusa.

### 9.3 Pohitritev iskanja presečišč za 2Sphere

Za 2Sphere smo našli rešitev, ki je hitrejša od iterativnega pristopa aproksimacij premikov po geodetkah.

Vsek žarek, gre namreč čez vrh in dno sfere. Žarek parametriziramo, kot točko na sferi. Za žarek poiščemo center krogla  $[x_0, y_0, z_0]$ , kot smo ga prej in za preslikavo na  $uv$  ravnino uporabimo enačbi 5 in 6. Parametrizacija žarka  $s(t)$  je torej:

$$\begin{aligned} x &= R \cos(v) \sin(t) + x_0 \\ y &= R \sin(v) \sin(t) + y_0 \\ z &= R \cos(t) + z_0 \end{aligned} \tag{12}$$

Parametrizacija za razliko od prejšnje ne računa približkov, ampak točno rešitev za katerokoli velikost koraka. Zato jo lahko uporabimo v algoritmu 9.1.

## 10 Zaključek

Naš cilj je bil narediti program, v katerem se vsak uporabnik lahko igra z različnimi neevklidskimi prostori in jih vizualizira. To nam je uspelo. Uspelo nam je tudi pospešiti program do te mere, da na slike v dokaj visoki resoluciji ni potrebno čakati več kot nekaj deset sekund. Na slike v evklidskem prostoru pa, za 1500 x 1000 sliko, celo manj kot 10 sekund.

Sicer obstaja še marsikatera optimizacija in funkcionalnost, ki bi jo lahko dodali, na primer možnosti dodajanja materialov in boljši algoritem senčenja.

Eden izmed ciljev je bil tudi enostavno dodajanje novih objektov in prostorov:

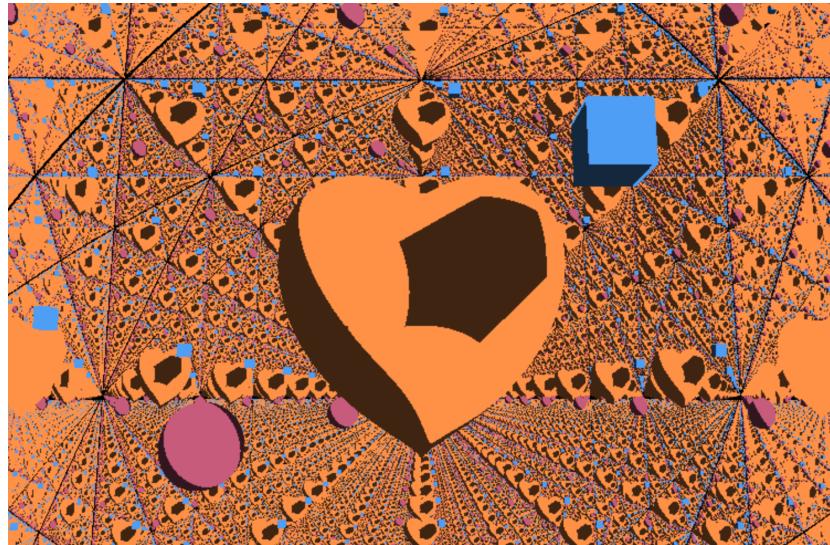


Figure 23: Demonstracija dodajanja objektov srca in kocke

## 11 Razdelitev dela

- Blaž Bergant: Zasnova programa, implementacija osnovnega raytracing algoritma, izpeljava formул za hitrejše delovanje.
- Martin Jereb: Priprava poročila, razлага in implemenacija prostora flat torus, priprava predstavitev.
- Matjaž Pogačnik: Implementacija in izpeljava algoritmov za 2Sphere, priprava poročila in generiranje slik.

## 12 Viri

- Kovač, G. (2023). *Sledenje žarku v neevklidskih prostorih* (Diplomska naloga). Ljubljana: [G. Kovač].
- Kovač, G. (2024). *Ray tracing in Non-euclidean spaces*. Ljubljana: [G. Kovač].
- Zalar, A. *Mathematical Modelling, Lecture Notes* (2024). Ljubljana: [A. Zalar].
- *Non-Euclidean geometry*. (2024). Pridobljeno 24.5.2024 s spletnne strani: [https://en.wikipedia.org/wiki/Non-Euclidean\\_geometry](https://en.wikipedia.org/wiki/Non-Euclidean_geometry).

## **13 Priloga**

- Koda je na voljo na spletnem repozitoriju, na povezavi.
- Spletna predstavitev pa je na voljo tukaj.