SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1720

SIMULACIJE RASPRŠIVANJA ZRAKA SVJETLOSTI

Marko Lujo

Zagreb, lipanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1720

SIMULACIJE RASPRŠIVANJA ZRAKA SVJETLOSTI

Marko Lujo

Zagreb, lipanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 3. ožujka 2025.

ZAVRŠNI ZADATAK br. 1720

Pristupnik:	**Marko Lujo (0036549373)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	prof. dr. sc. Željka Mihajlović

Zadatak: **Simulacije raspršivanja zraka svjetlosti**

Opis zadatka:

Realistično atmosfersko raspršenje svjetlosti u računalnoj grafici uvijek je bio izazovan problem. Proučiti modele raspršivanja zraka svjetlosti i mogućnosti rješavanja postavljenih jednadžbi. Posebice obratiti pažnju na modele raspršivanja svjetlosti Rayleigha i Mie. Razraditi proučene modele i ostvariti programsku implementaciju. Načiniti testiranje i usporedbu na nizu primjera. Analizirati i ocijeniti ostvarene rezultate. Diskutirati upotrebljivost ostvarenih rezultata kao i moguća proširenja. Izraditi odgovarajući programski proizvod. Koristiti programski jezik C++ i grafičko programsko sučelje Vulkan. 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: 23. lipnja 2025.

Sadržaj

Uvod ... 1

1. Teorija ... 2

1.1. Osnove raspršivanja čestica ... 2

1.2. Modeli atmosferskih raspršivanja ... 2

1.2.1. Rayleighova aproksimacija raspršivanja ... 3

1.2.2. Mie teorija ... 4

1.3. Teorijski opis simulacije ... 5

2. Korištena tehnologija ... 9

2.1. Programsko sučelje Vulkan ... 9

2.1.1. Računalni sjenčari ... 9

2.2. Ostale biblioteke ... 10

2.2.1. SDL ... 10

2.2.2. GLM ... 10

2.2.3. VkBootstrap ... 11

2.2.4. Vulkan Memory Allocator ... 11

2.2.5. Dear ImGui ... 11

3. Opis implementacije ... 12

3.1. Organizacija ulaznih podataka ... 12

3.2. Sustav prikazivanja ... 13

3.2.1. Klasa RenderEngine ... 13

3.3. Glavna simulacija ... 14

3.3.1. Algoritam raspršivanja ... 15

4. Rezultati, analiza i usporedba ... 19

4.1. Analiza vidljivih rezultata ... 19

4.2. Mjerenje performanse ... 25

4.3. Usporedba sa sličnim radovima.....31
Zaključak.....34
Literatura.....35
Sažetak.....36
Summary.....37
Privitak.....38

Uvod

Nebo je stalno prisutan dio vanjskog svijeta koji često zauzima više od trećine vidokruga, i pojave koje se u njemu događaju mogu biti vrlo zanimljive. Većina 3D aplikacija koje pokušavaju prikazati ili simulirati neku vrstu okoliša (koji nije u potpuno zatvorenom prostoru) trebaju prikazati nebo na neki način. Međutim, nebo sadrži vrlo mnogo atmosferskih pojava koje mu daju njegove karakteristične izglede, te u većini aplikacija simulacija tih pojava jednostavno nije vrijedna za izvesti pa se koriste zamjene kao gradijenti ili teksture neba koje su prikazane u pozadini.

Cilj ovog rada je proučiti modele raspršivanja zraka svjetlosti (s posebnom pažnjom na osnovne Rayleighove jednadžbe i aproksimacije Mie teorije) te napraviti programsku implementaciju koja može nacrtati relativno realističnu sliku neba bez oblaka u interaktivnom vremenu, te proučiti rezultate i usporediti ih s drugim sličnim implementacijama.

Implementacija projekta bit će napisana u jeziku C++ i koristit će programsko sučelje Vulkan zbog mogućnosti upotrebe računalnih sjenčara - malih programa koji se mogu masovno izvoditi na grafičkom procesoru računala, time omogućujući vrlo brzo izvođenje simulacijskih algoritama i direktan prikaz njihovih rezultata na zaslon.

1. Teorija

1.1. Osnove raspršivanja čestica

U malim količinama, plinovi propuštaju svjetlost. No kada svjetlost prolazi kroz velike količine plinova, počinju se događati pojave raspršivanja poput refleksije i refrakcije - putanje zraka svjetlosti mijenjaju se ovisno o njihovim svojstvima i okolnim uvjetima.

Glavni uzrok karakterističnih boja neba upravo je raspršivanje čestica kroz atmosferu. Neke od odbijenih zraka dospiju do oka promatrača iz različitih smjerova te pridonose percipiranoj boji neba u tom smjeru.

Pri prolasku kroz atmosferu događa se velik broj različitih vrsta raspršivanja, poput elektromagnetske apsorpcije i reemisije fotona [1] ili refrakcije zraka svjetlosti kroz koloidne sustave (mješavine čestica veličina između 1 nm i 1 μm u otopini, npr. magla)

1.2. Modeli atmosferskih raspršivanja

Zbog ogromne količine ulaznih zraka svjetlosti, dalekog puta kroz atmosferu i velikog broja događaja, simulacija točnih pojava raspršivanja na čestičnoj razini potpuno je nepraktično za bilo kakvu primjenu na većoj razini od mikroskopske - broj potrebnih simulacija u jedinici vremena jednostavno je prevelik. Zbog toga, nastalo je nekoliko modela i teorija koje opisuju raspršivanje zraka svjetlosti na makroskopskoj razini. Ovaj rad usredotočit će se na dva najčešće primijenjena modela - Rayleighovu aproksimaciju i Mie teoriju.

1.2.1. Rayleighova aproksimacija raspršivanja

Rayleighov model raspršivanja opisuje „najčešću" vrstu raspršivanja u atmosferi - slučaj kada je radijus čestice od koje se svjetlost raspršuje puno manji od valne duljine svjetlosti.

Ovakvo raspršivanje događa se pri sudaru zrake s najčešćim osnovnim molekulama zraka (dušikom i kisikom) te je glavni uzrok plave boje neba po danu.

Teoriju ove vrste raspršivanja opisao je engleski znanstvenik John Strutt (poznat i kao Lord Rayleigh) u raznim radovima 1871. - 1899. - važno otkriće za ovaj rad bile su formule koja opisuju intenzitet dolazne svjetlosti u odnosu na ulaznu. Postoji nekoliko načina izražavanja Rayleighovih formula, a u ovom radu kao osnova koristit će se sljedeća:

$$I = I_0 * F(\theta) * N * \sigma(\lambda) \tag{1}$$

Gdje je I_0 ulazna svjetlost, $F(\theta)$ funkcija faze (ovisnosti raspršivanja o kutu ulazne i izlazne zrake), N broj raspršujućih čestica na putu zrake, a $\sigma(\lambda)$ optički presjek plina i zrake.

$$F(\theta)_{Rayleigh} = \frac{3}{4} * (1 + \cos^2(\theta)) \tag{2}$$

Zbog izrazito malih veličina čestica, sudari Rayleighovog raspršivanja zapravo nisu mehanički sudari već elektromagnetske apsorpcije i emisije, te se raspršuju simetrično (ali ne i uniformno).

$$\sigma(\lambda) = \frac{8\pi}{3} * \left(\frac{2\pi}{\lambda}\right)^4 \left(\frac{n^2 - 1}{n^2 + 2}\right)^2 * r^6 \tag{3}$$

Gdje su λ valna duljina svjetlosti, n indeks refraktivnosti zraka, i r radijus raspršujuće čestice.

Kako dolazna svjetlost ovisi o negativnoj četvrtoj potenciji valne duljine svjetlosti, zrake s manjom valnom duljinom (tirkizne, plave i ultraljubičaste) raspršit će se mnogo više od onih s manjom (crvenih i narančastih).

Rezultat ovoga jest Tyndallov efekt - nakon prolaska kroz plin, zrake koje su zadržale originalni smjer bit će više crveno obojane dok zrake promijenjenog smjera sadrže više plave valne duljine.

1.2.2. Mie teorija

Kada se zraka svjetlosti sudara s česticom veće ili približno jednake veličine njezine valne duljine, Rayleighova aproksimacija prestaje dobro predviđati raspršivanje svjetlosti.

Kako se nezanemariva količina zraka sastoji od aerosolnih čestica (suspenzije tekućih ili čvrstih tvari u plinu, kao magla ili dim), i kako one raspršuju zrake u mnogo većem omjeru po količini nego molekularne čestice, bitno je uključiti i njih u simulaciju.

Nakon Rayleighovog modela, nastalo je mnogo teorija i pokušaja rješenja općenitog čestičnog raspršivanja elektromagnetskog zračenja. Njemački fizičar Gustav Mie uspio je riješiti jednadžbe raspršivanja te ih je opisao kao niz valova. Postoji puno formula i aproksimacija izvedenih iz tog rješenja, često nazvanih "Mie teorija".

Nažalost, većina rješenja sadrže sumacije i integrale koji su nedovoljno praktični za interaktivnu računalnu implementaciju, te ne postoji jedinstvena "čarobna" formula koja bi približno točno opisala sve potrebno.

Međutim, vrlo koristan dio Mie teorije upravo je funkcija faze, koja opisuje intenzitet svjetlosti ovisno o ulaznom i izlaznom kutu. U svome radu [2] W.M.Cornette i J.Shanks opisali su vrlo koristan izraz za generalnu funkciju faze raspršivanja:

$$F(\theta)_{Mie} = \frac{3*(1-g^2)}{2*(2-g^2)} * \frac{1+\cos^2(\theta)}{(1+g^2-2g*\cos(\theta))^{\frac{3}{2}}} \tag{4}$$

Gdje je g faktor asimetrije i veći je što su pretpostavljene koloidne čestice veće.

Ova formula zapravo je originalni oblik funkcije faze za Rayleighovo raspršivanje (i reducira se u nju kada je g=0).

Dovoljno dobra aproksimacija koloidnog raspršivanja može se dobiti primjenom Rayleighovog modela, pritom zamjenjujući faznu funkciju s ovom i postavljajući valnu duljinu na dogovorenu konstantu (budući da koloidno raspršivanje ne ovisi o valnoj duljini zrake).

1.3. Teorijski opis simulacije

Glavni način izvođenja simulacije bit će pomoću računalnih sjenčara, tj. za svaki piksel na ekranu simulirat će se jedna dolazna zraka.

Za svaku zraku potrebno je simulirati dvije komponente - količina svjetla koju zraka izgubi zbog raspršivanja, te količina svjetla koju zraka dobije od drugih zraka raspršivanjem.

Gubitak svjetla nije teško izračunati primjenom jednadžbe Rayleighove aproksimacije. Potrebno je odrediti ukupan optički presjek cijelog puta kojeg zraka prođe, te ga uvrstiti u formulu.

Formule korištene u implementaciji izvedene su iz osnovne Rayleighove formule (uz par dodatnih aproksimacija) prema radu Nishite i ostalih [3] i praktičnije su za računalnu izvedbu.

$$t(AB,\ \lambda) = 4\pi * \rho_{AB} * \frac{K}{\lambda^4} \tag{5}$$

Funkcija t određuje omjer optičke dubine, tj. količinu raspršivanja zrâka određene valne duljine λ. ρ_{AB} ukupan je omjer gustoće atmosfere između točke A i B (omjer znači da je ρ=1 na morskoj razini a 0 na rubu atmosfere i dalje). Kako atmosfera nema jednaku gustoću svugdje već opada eksponencijalno ovisno o visini, potrebno je integrirati aproksimativnu funkciju gustoće (u ovom slučaju e^h) po putu.

$$\rho_{AB} = \int_A^B \left(e^{-\frac{h}{H_0}}\right) dh \tag{6}$$

gdje je h visina, a H_0 postavljena visina na kojoj je gustoća atmosfere jednaka njezinoj srednjoj gustoći.

Ovakav integral nije moguće riješiti analitički, no moguće ga je numerički aproksimirati: putanja zrake podijelit će se na dijelove, te će se na svakom dijelu izračunati gustoća atmosfere i pridodati rezultatu.

Komponenta K jest konstanta atmosfere. Budući da se tijekom simulacije atmosfera neće često mijenjati (samo kod ručnih mijenjanja parametara), korisno je unaprijed izračunati sve konstante atmosfere u jednu. Izračun konstante je:

$$K = \frac{2\pi^2 * (n^2 - 1)^2}{3 * N} \tag{7}$$

gdje je n indeks loma zraka (približno 1) a N gustoća molekula na površini atmosfere (broj čestica molekula po 1 kubnom metru).

Ove varijable moguće je ili direktno zadati ili ih izračunati primjenom plinskih jednadžbi ako su druge konstante zadane. U ovoj simulaciji, zadani su površinski tlak zraka i temperatura (a budući da utječu samo na jednu stvar, samo se tlak može mijenjati kao parametar), preko kojih se računa broj mola a time i N.

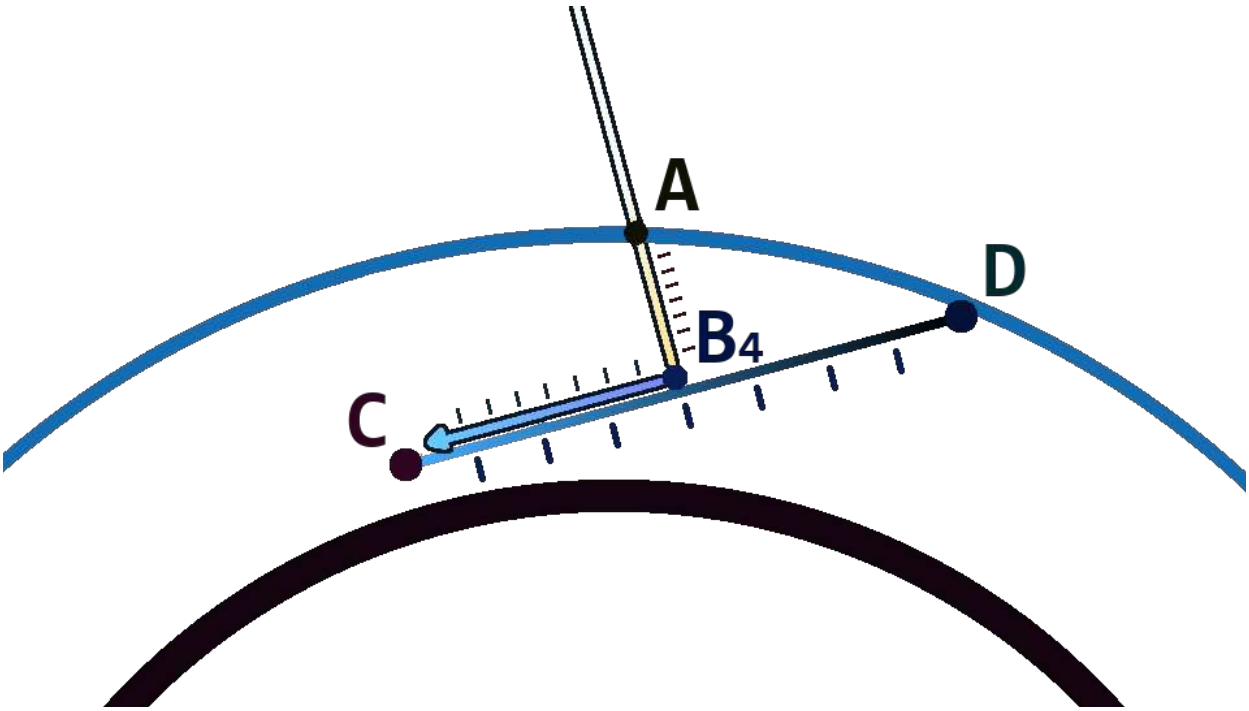
Nakon određivanja varijable optičke dubine moguće je odrediti izlazno raspršivanje (količinu svjetlosti koja se izgubi putovanjem zrake kroz atmosferu):

$$I = I_0 * e^{-t(AB,\ \lambda)} \tag{8}$$

Optička dubina veća je što je valna duljina manja (te zrake raspršuju se više) tako da je potrebno uzeti inverzno proporcionalnu vrijednost kada se gleda koje zrake ostaju na početnoj putanji, što odgovara Lambert-Beerovom zakonu eksponencijalnog pada [4].

Računanje dobitka svjetla raspršivanjem nešto je teže. Zrake mogu doći iz bilo kojeg smjera i raspršiti se više puta prije dolaska do očišta. Potrebno je uvesti još jednu aproksimaciju, gdje će se dobivanje svjetlosti računati samo za zrake koje dolaze direktno iz Sunčevog smjera. Osim što i te zrake gube svjetlost putujući do zrake pogleda, potrebno je izvesti još jedan numerički integral koji dodaje svjetlost glavnoj zraci po njezinom putu.

Slika (1.1) vizualizira bitne dijelove algoritma.



Sl. 1.1: Vizualizacija algoritma simulacije

Glavna zraka bit će podijeljena na točke uzorka, te će se na svakoj izračunati doprinos Sunčevog svjetla sljedećom jednadžbom:

$$I_{Bi} = I_{Bu} * F(\theta) * \rho_B * \frac{K}{\lambda^4} \tag{9}$$

Jednadžba je slična računanju optičke dubine (5) samo što se ovaj put gleda izlazna svjetlost kao rezultat i također je dodana fazna funkcija. I_{Bu} je ulazna svjetlost u točku uzorka (za koju se pretpostavlja da je već izgubila nešto svjetlosti putujući do točke), a ρ_B gustoća atmosfere u točci (koja se računa samo za tu točku pa nije potreban integral).

Putujući do točke očišta zraka će ponovno izgubiti nešto svjetlosti raspršivanjem tako da će ukupan pridonos (raspisujući i ulaznu zraku) glasiti:

$$I_{Cn} = \left(\left(I_0 * e^{-t(ABn,\ \lambda)} \right) * F(\theta) * \rho_{Bn} * \frac{K}{\lambda^4} \right) * e^{-t(BnC,\ \lambda)} \tag{10}$$

Gdje je Bn točka uzorka, A točka ulaza Sunčeve zrake u atmosferu a C točka očišta (ili izlaza odbijene zrake iz atmosfere, ovisi što je bliže)

Napokon, zbrajajući sve zrake dobije se konačan integral (koji će se naravno riješiti numerički, podjelom na točke uzorka):

$$I_C = \int_C^D \left(\left(I_0 * e^{-t(ABn,\ \lambda)} \right) * F(\theta) * \rho_{Bn} * \frac{K}{\lambda^4} \right) * e^{-t(BnC,\ \lambda)} dn \tag{11}$$

Sažeti opis rada algoritma simulacije:

Kako bi algoritam izračunao ukupnu svjetlost zrake CD na poziciji očišta (u ovom slučaju C), prvo bi podijelio dužinu CD (gdje je C ili ulazni presjek zrake s rubom atmosfere ili očište ako je unutar atmosfere, a D izlazni presjek zrake s rubom atmosfere) na unaprijed određeni broj točaka uzorkovanja. Na svakoj točci uzorka (na primjeru točka B_4), odredila bi se nova, upadna zraka AB koja ulazi direktno od smjera Sunca. Zraka gubi svjetlost putujući do točke B (gubitak svjetlosti računa se spomenutom funkcijom izlaznog raspršivanja), te se na točci B računa doprinos te zrake novoj, odbijenoj zraci. Ta odbijena zraka putuje do kamere te također gubi nešto svjetlosti na svom putu, nakon kojeg doprinosi ukupnom zbroju svjetlosti.

2. Korištena tehnologija

2.1. Programsko sučelje Vulkan

Za izvođenje računalnih sjenčara i prikazivanje njihovih rezultata korišteno je programsko sučelje Vulkan.

Vulkan je relativno novo (službeno je stvoren 2015.) sučelje niske razine koje omogućuje komunikaciju i korištenje grafičkih procesora prisutnih na računalu. Održava ga grupa Khronos.

Sučelje omogućuje vrlo veliku kontrolu nad grafičkim procesorima i njihovim protočnim sustavima neovisno o operacijskom sustavu i arhitekturi računala, zbog čega je široko korišteno u programima raznih funkcija, npr, modeliranja (Blender), razvoja 3D aplikacija (Unity Engine, Unreal Engine), kao i u filmskoj industriji i sustavima koji zahtijevaju veliku paralelizaciju.

U ovom projektu, Vulkan se koristiti kako bi omogućio izvođenje računalnih sjenčara u velikoj paralelnoj količini (što omogućuje prikaz u interaktivnom vremenu) te stvorio grafičku protočnu strukturu koja može izravno prikazati njihov izlaz na zaslon.

Bitno je spomenuti da se ovaj projekt oslanja na Vulkan SDK (razvojne alate za Vulkan koje je stvorila kompanija LunarG), koji sadrže sve potrebne alate za izgradnju Vulkan aplikacije, kao i neke dodatne korištene biblioteke (GLM i SDL2).

2.1.1. Računalni sjenčari

Prednost sučelja Vulkan nad nekim ostalima jest što podržava pokretanje proizvoljnih programa na grafičkom procesu (za razliku od standardnih vertex/fragment sjenčara koje se najčešće koriste za prikazivanje) - ovo omogućuje rješavanje problema direktno simulirajući zrake i jednostavno kopirajući njihove rezultate na zaslon bez dodatnog računanja ploha ili projiciranja.

Računalni sjenčari maleni su programi koji se izvode u velikim paralelnim količinama na grafičkom procesoru. Rade na principu SIMD (single instruction, multiple data), te uobičajeno dobivaju ulazne podatke iz ulaznih nizova kopiranih na memoriju grafičkog procesora ili uniformnih varijabli, pa ih također spremaju u svoju poziciju na izlaznom nizu.

Vulkan koristi računalne sjenčare napisane u GLSL-u (isto kao i OpenGL), s dodatnim zahtjevom da se sjenčari trebaju prvo prevesti u SPIR-V bajtkod prije učitavanja i izvođenja. Zbog toga ovaj projekt također sadrži skriptu za pozivanje prevoditelja sjenčara iz Vulkanovog SDK-a.

2.2. Ostale biblioteke

2.2.1. SDL

Sučelje Vulkan omogućuje korištenje grafičke kartice i slanje podataka na zaslon, no nema mogućnost stvoriti prozor koji bi prikazao te podatke, ni pratiti korisnički ulaz. Zbog toga Vulkan aplikacije najčešće koriste još neku biblioteku za stvaranje prozora i procesiranje ulaza s tipkovnice ili miša.

Biblioteka Simple Directmedia Layer omogućuje jednostavno stvaranje i upravljanje prozorima, te čitanje korisničkih ulaza. Implementirana je u C++u i ima jako dobru podršku za Vulkan (toliko dobru da je uključena u Vulkan SDK), što je učinilo njezin izbor laganom odlukom.

Projekt koristi 2. verziju biblioteke (SDL2).

2.2.2. GLM

OpenGL Mathematics je C++ biblioteka koja implementira vektorske strukture i funkcije prisutne u jeziku GLSL.

Korištenje vektorskih struktura direktno u C++ kodu jako je korisno za bilo kakve prostorne operacije, i također omogućuje lagano prenošenje istih vektorskih struktura u ulazne parametre računalnih sjenčara.

Ova biblioteka također je jako često korištena i uključena je u SDK, zbog čega ju nije bilo potrebno ručno dodavati već se samo referencira unutar koda.

2.2.3. VkBootstrap

Velika kontrola koju Vulkan omogućuje također znači da postoji vrlo malo predefiniranih struktura i funkcija, te je potrebno ručno izvesti sve korake uspostavljanja komunikacije s grafičkim procesorom u kodu, što zauzvrat znači pisanje jako puno koda. Mnogo ovih koraka (stvaranje Vulkan instance, pronalaženje odgovarajućeg grafičkog uređaja, dohvaćanje prikaznog sustava računala) zapravo su isti ili slični u većini aplikacija (uključujući ovu) te ih nije potrebno mijenjati. Biblioteka VkBootstrap dodaje nekoliko funkcija koje olakšavaju ove korake i smanjuju potrebnu količinu pisanja koda.

2.2.4. Vulkan Memory Allocator

Iz sličnih razloga kao i VkBootstrap, ova biblioteka korištena je kako bi olakšala prijenos podataka iz memorije glavnog računala na memoriju grafičkog procesora. Jedina razlika u projektnoj implementaciji jest što je VMA uključen u Vulkanove razvojne alate te ga nije bilo potrebno dodavati u projektne datoteke.

2.2.5. Dear ImGui

Ispis kontrola na konzolu te mijenjanje svih parametara s pomoću posebnih fizičkih tipki nije baš intuitivan dizajn, zbog čega je također naknadno dodana GUI biblioteka.

Dear ImGui široko je korištena biblioteka za prikaz elemenata grafičkog sučelja. Često je korištena u razvojnim okruženjima, no može se koristiti i u komercijalnim aplikacijama. Integracija biblioteke u postojeću aplikaciju nije prezahtjevna zbog izvrsne podrške za razne sustave prikaza (uključivo Vulkan i SDL2). Jednom kada je integrirana, pisanje koda za stvaranje novih GUI elemenata unutar aplikacije može se izvesti vrlo brzo, što je idealno za ovaj projekt budući da je biblioteka bila dodana tek nakon osnovne funkcionalnosti.

3. Opis implementacije

Projekt se sastoji od:

- struktura koje predstavljaju ulazne parametre planeta, atmosfere i izvore svjetla
- sustava prikazivanja koji povezuje sva sučelja potrebna za izvođenje sjenčara te prikaz simulacije i korisničkog sučelja
- glavnog sjenčara koji prima navedene strukture te izvodi opisanu simulaciju na njima

3.1. Organizacija ulaznih podataka

Program prikazuje sve strukture u parametarskom obliku, što omogućuje da se veliki, vrlo kompleksni objekti poput planeta jednostavno aproksimiraju kao sfera s položajem i radijusom.

U datoteci atmosphere.h definirane su strukture `Planet` i `Atmosphere`, koje sadrže sve potrebne parametre za prikaz atmosfere. Budući da je simulacija ograničena na prikazivanje jednog planeta, zapisivanje pozicije nije potrebno te se mnogo jednadžbi pojednostavi ako se pretpostavi geocentričan sustav te se središte planeta postavi u središte koordinatnog sustava. Planet je onda opisan samo svojim radijusom i atmosferom.

Parametri atmosfere već su bili opisani u teorijskom dijelu te su opisani u kodu, no ukratko, atmosfera je opisana:

- *parametrima za računanje konstante K*: tlak na morskoj razini, temperatura i refraktivnost
- *opisnicima visine*: gornja granica atmosfere i razina na kojoj se nalazi prosječna gustoća zraka te aerosola (zasebno budući da su uobičajeno različite)
- *podacima specifičnih za aerosolno raspršivanje*: relativna količina aerosola te faktor asimetrije

Osim planeta, kao parametarska kugla opisano je i simulirano Sunce (klasa `Sun` u datoteci sun.h) koje može kružiti oko planeta. Parametri potrebni za to su udaljenost i trenutačni kut (iz kojih se može odrediti pozicija), te veličina.

Također je moguće promijeniti intenzitet Sunčeve svjetlosti te jačinu i valnu duljinu pojedinih komponenata svjetlosti koje ono emitira.

Na početku programa, većina vrijednosti pretpostavljene su na one koje odgovaraju izmjerenim parametrima Zemlje i Sunca u pravom svijetu.

3.2. Sustav prikazivanja

Glavni zadatak implementacije protočnog sustava jest omogućavanje pokretanja računalnog sjenčara, te prikaz njegovih rezultata na zaslon.

Potrebno je postaviti protočnu strukturu koja će moći svaki prikaz (poželjno 30 ili više puta u sekundi) dohvatiti informacije iz programskih struktura glavnog programa, kopirati im sadržaje na grafičku karticu, izvršiti glavni simulacijski sjenčar, dohvatiti rezultate, te prikazati izlazne podatke (zajedno s grafičkim sučeljem) na zaslon.

3.2.1. Klasa `RenderEngine`

Glavna (i jedina) klasa u projektu koja koristi Vulkanovo sučelje jest `RenderEngine`. Prije pokretanja, u nju je potrebno unijeti strukture navedene u 3.1, te par dodatnih parametara simulacije kao veličina prozora, broj iteracija za računanje integrala i brzina kamere.

Klasa sadrži 3 javne metode:

`init()` - stvori korisnički prozor i pokuša naći prikladni grafički procesor, te, ako uspije, postavi parametre protočne strukture i pokuša učitati glavni simulacijski sjenčar iz datoteke /shaders/main_shader.spv .

`run()` - pokrene simulaciju te počinje prikazivati njezine rezultate na prozor (kao i korisničko sučelje). Program će ostati u ovoj metodi sve dok se aplikacija ne prekine pritiskom na gumb zatvaranja prozora ili naredbom Alt+F4.

`cleanup()` - počisti i dealocira sve strukture alocirane za vrijeme `init()`-a.

Uobičajeno bi se sve tri funkcije pozvale ovim redoslijedom u kodu.

3.3. Glavna simulacija

Nakon postavljanja svih parametara, pokreće se glavni simulacijski sjenčar. Kao i opisano, on sadrži program koji se izvodi jedanput za svaki piksel na ekranu, jednom po crtanju zaslona, time simulirajući jednu zraku svjetlosti.

Svaki paralelan program pokretanja sjenčara sadrži jedinstven ID preko čega je moguće izračunati kojem pikselu na zaslonu on odgovara te u kojem smjeru se kreće simulirana zraka.

Za svaku zraku prvo se provjerava s čime ima presjek. Kako su jedini vidljivi objekti u simulaciji sfernog oblika, program implementira provjeru presjeka zrake i kugle:

```
struct ray_sphere_result {
    bool intersect;
    float t_min;
    float t_max;
};

// Provjera pogađa li zraka sferu i, ako da, u kojim
parametarskim točkama
ray_sphere_result ray_sphere_intersect(vec3 ray_pos, vec3
ray_dir, vec3 sphere_pos, float sphere_r);
```

Funkcija u sebi sadrži rješavanje kvadratne jednadžbe opisane krugom i parametarskim pravcem, te vraća podatke ima li presjeka te gdje su parametarske točke presjeka (ako postoji samo 1 presjek točke će biti iste, ako nema presjeka nisu definirane). Nakon izvođenja funkcije nad planetom, Suncem i atmosferom, potrebno je razdijeliti program u slučajeve:

- Ako nema presjeka ni s jednim (varijabla `intersect` svakog rezultata je false ili su obje točke negativne (presjek se događa iza smjera zrake)), crta se prazna boja (crna)

- Ako ima presjeka s planetom i zraka je unutar planeta (`t_min` je manja od 0 a `t_max` veća od 0), crta se samo boja planeta

- Ako ima presjeka s atmosferom, algoritam simulacije započinje.

3.3.1. Algoritam raspršivanja

Glavni dio algoritma u sjenčaru direktna je implementacija koraka opisanih u teorijskom dijelu. Program prvo definira dolaznu svjetlost zrake do očišta,

```
// Ukupno svjetlo koje kamera dobiva
vec3 total_light = vec3(0,0,0);
```

te zatim započne petlju koja dijeli zraku na točke uzorka i na svakoj računa svjetlost koju ona pridonosi.

Za svaku točku prvo će izračunati postoji li uopće upadna zraka Sunca. Kada je noć, nema raspršivanja svjetla jer je atmosfera u sjeni planeta. Za ovo se izvodi još jedna provjera presjeka:

```
// Provjera presjeka s planetom - ako ga ima, ne doprinosi
svjetlo
ray_sphere_result sample_planet_intersect =
ray_sphere_intersect(t_pos, normalize(ray_sun_vector),
planet_pos, atmosphere_info.planet_radius);
```

Zatim se računa ulazno raspršivanje - ulazno svjetlo koje dođe do točke uzorka i udio tog svjetla koje se odbije prema smjeru očišta.

Nakon što se odredi presjek ulazne zrake s atmosferom, izvodi se numeričko računanje integrala relativne gustoće zraka.

```
float outScatter_partial(vec3 start, vec3 end, float
average_distance){
    float result = 0;

    for(int j = 0; j < camera_info.sampleAmount_out; j++){
        // Pozicija točke uzorka na liniji
        vec3 pos = (start *
(1-(float(j)/(camera_info.sampleAmount_out-1))) + end *
(float(j)/(camera_info.sampleAmount_out-1))); //
Interpolacija

        float distance_from_center = length(pos);
        float distance_from_surface = distance_from_center -
atmosphere_info.planet_radius;

        result +=
exp(-distance_from_surface/average_distance) * length(end -
start)/camera_info.sampleAmount_out;
    }

    return result;
}
```

Gustoća se određuje zasebno za Rayleighove i Mie aerosolne komponente zraka.

Slijedi izvođenje formule (9) za izračun koliko se svjetla odbije u smjeru očišta. Ovaj dio se također izvodi nezavisno za Rayleighovu i Mie simulaciju, te također za svaku komponentu svjetlosti u slučaju Rayleighove jednadžbe.

Isječak koraka:

Određivanje ulaznog svjetla (B_u u formuli (9)):

(`t_pos` je pozicija točke uzorka (B na slici 1.1), dok izraz `(t_pos + normalize(ray_sun_vector) * t_max_2)` odgovara presjeku s rubom atmosfere (A))

```
vec3 ray_light = starting_ray_light;


float average_density_ratio = outScatter_partial(t_pos,
t_pos + normalize(ray_sun_vector) * t_max_2,
atmosphere_info.atmosphere_average_distance);


float rayleigh_part_1_red   = 4 * pi * average_density_ratio *
(float(atmosphere_info.K) / (red_wavelenght * red_wavelenght *
red_wavelenght * red_wavelenght));
```

Određivanje izlaznog svjetla (B_u u formuli (9)):

Ako zraka gledanja ima presjek sa Suncem ili planetom, ovaj korak se preskače te se zraci direktno dodaje ulazno svjetlo ili difuzno odbijanje ulaznog svjetla od površine planeta.

```
// Rayleighova fazna funkcija
float angle_const_rayleigh = float(3.0/(4.0) * (1 + cos_sun_angle *
cos_sun_angle));


// Određivanje gustoće zraka oko točke uzorka - jednako onom integralu
ali računa se samo jedanput
float density_ratio = exp(-(length(t_pos) -
atmosphere_info.planet_radius)/atmosphere_info.atmosphere_average_distanc
e);


rayleigh_part_2_red   = ray_light.r  * angle_const_rayleigh *
density_ratio * (float(atmosphere_info.K) / (red_wavelenght *
red_wavelenght * red_wavelenght * red_wavelenght)) * exp(-
rayleigh_part_1_red);


in_scatter_light += vec3(rayleigh_part_2_red, rayleigh_part_2_green,
rayleigh_part_2_blue) + vec3(mie_part_2, mie_part_2, mie_part_2);
```

Preostalo je još samo izračunati izlazno raspršivanje zrake od točke uzorka do očišta, što je jednako 1. dijelu računanja ulaznog raspršivanja, samo s drugim parametrima.

```
float average_density_ratio_2 = outScatter_partial(initPos +
normalize(velocity) * t_min, initPos + normalize(velocity) * t_smpl,
atmosphere_info.atmosphere_average_distance);


rayleigh_part_1_red   = 4 * pi * average_density_ratio_2 *
(float(atmosphere_info.K) / (red_wavelenght * red_wavelenght *
red_wavelenght * red_wavelenght));


total_ray_light.r = in_scatter_light.r * exp(-rayleigh_part_1_red   -
mie_part_1);
```

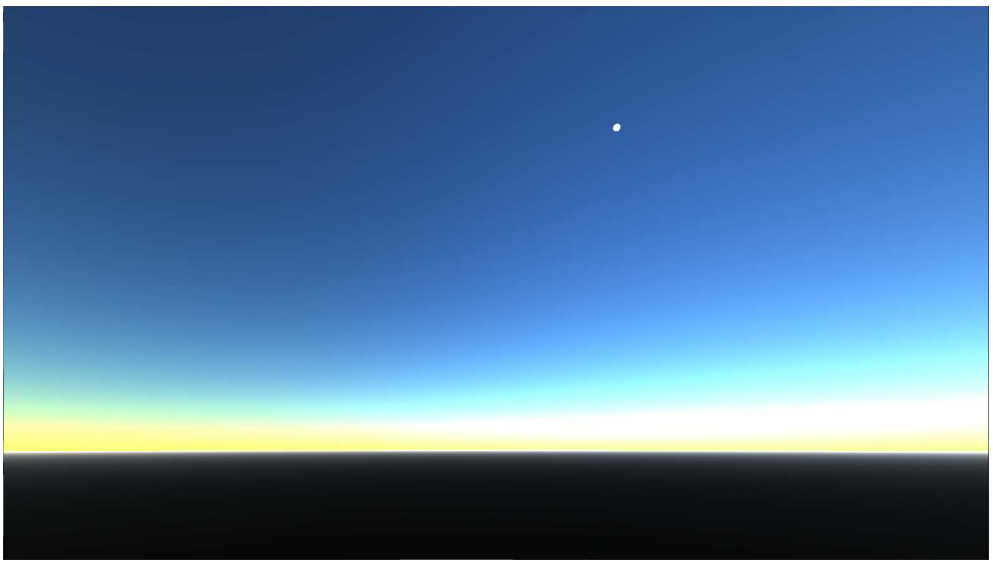
Nakon izvršavanja petlje, izlazna boja `total_ray_light` sprema se u teksturu na jedinstveno mjesto ove iteracije sjenčara (tj. na piksel kojem odgovara ova zraka).

Na kraju izvršavanja svih komponenti sjenčara slika se kopira natrag na glavnu memoriju centralnog procesora gdje se onda može preslikati na zaslon.

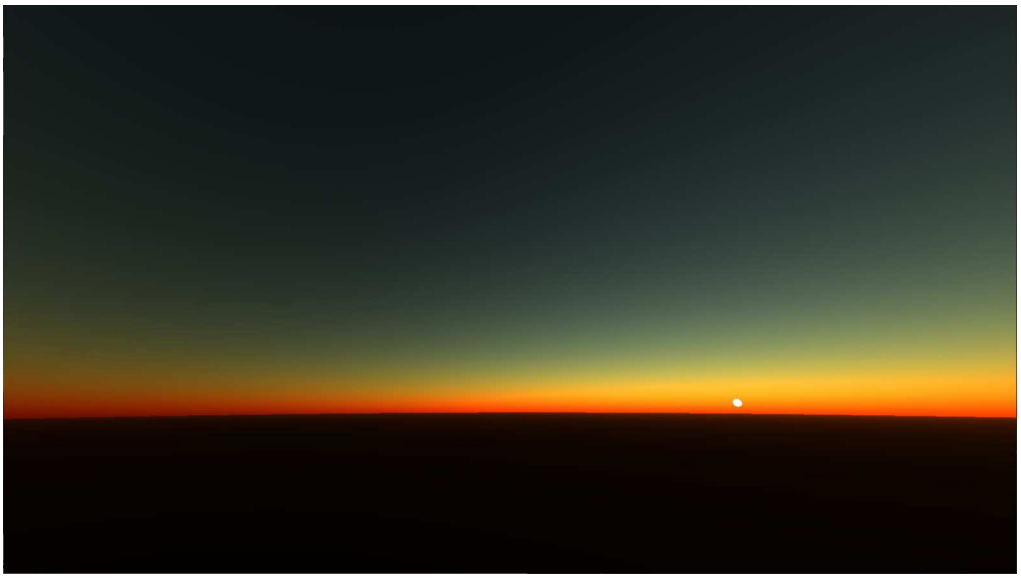
4. Rezultati, analiza i usporedba

4.1. Analiza vidljivih rezultata

Pokretanjem programa moguće je dobiti sljedeće izlaze (Sl. 4.1), (Sl. 4.2):



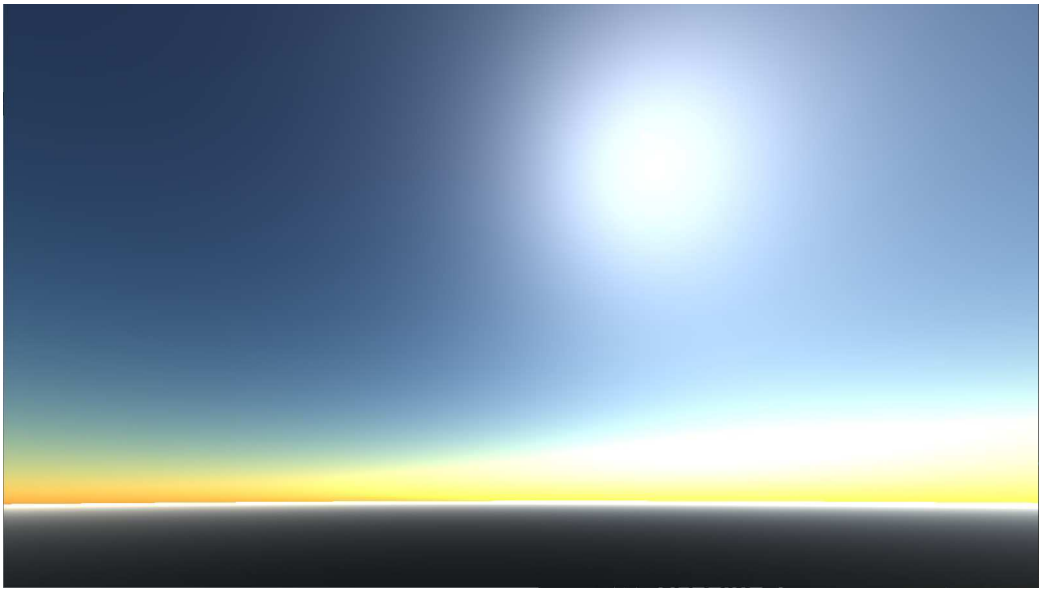
Sl. 4.1: Prikaz simulacije s Rayleighovom komponentom po danu



Sl. 4.2: Zalazak Sunca uz Rayleighovu simulaciju

Jedan od glavnih ciljeva implementiranja simulacije Rayleighovog raspršivanja bilo je upravo postizanje plavih boja atmosfere po danu/visokim kutevima Sunca, a crvenih boja pri njegovom zalasku.

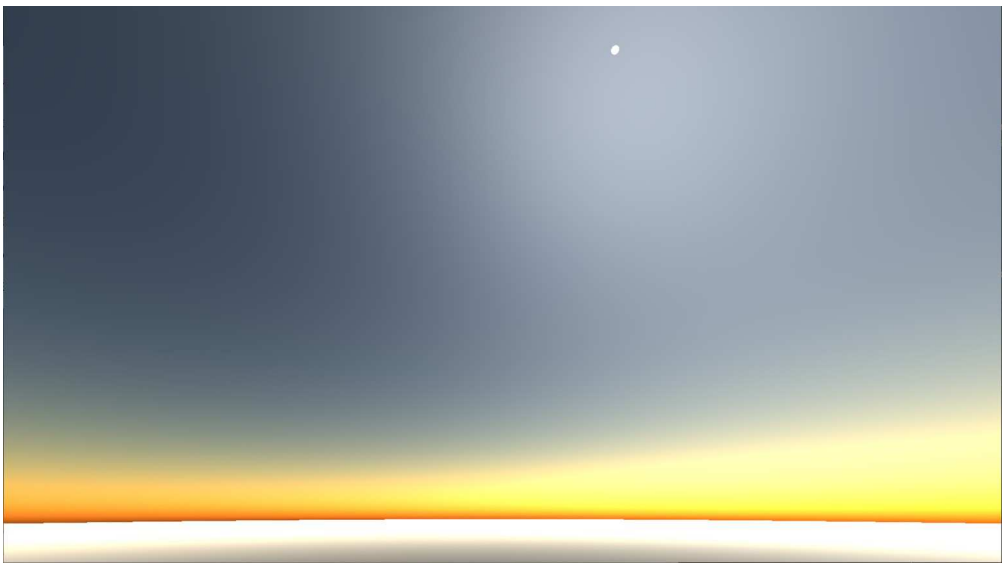
Uključivanjem simulacije Mie raspršivanja primjećuje se nekoliko promjena (Sl. 4.3).



Sl. 4.3: Dodavanje Mie raspršivanja uz veliku količinu aerosola

Najuočljivija pojava jest jak sjaj oko Sunčevog kuta, koji također lijepo odgovara opažanjima u pravom svijetu. Okolna boja neba manje je zasićena plavom bojom zbog jednolikog aerosolnog raspršivanja neovisno o valnoj duljini svjetlosti. Ova pojava može biti još uočljivija povećavanjem količine aerosola i smanjivanjem faktora Mie asimetrije koji lokalizira raspršivanje na područje oko kuta Sunca (Sl. 4.4).

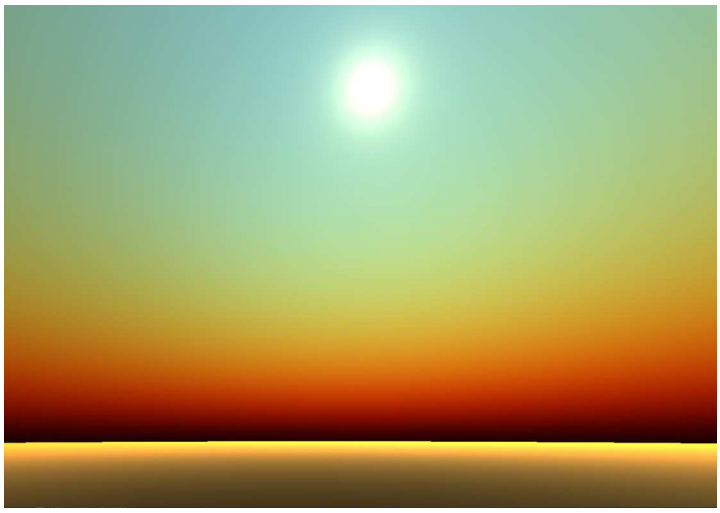
Zadnja promjena uočljiva je na dnu slika i produkt je nepreciznosti simulacije - boja neba pri horizontu žuto-narančasta je čak i kada je kut Sunca visok zbog akumuliranih pogrešaka numeričkog računanja integrala na područjima velike gustoće.



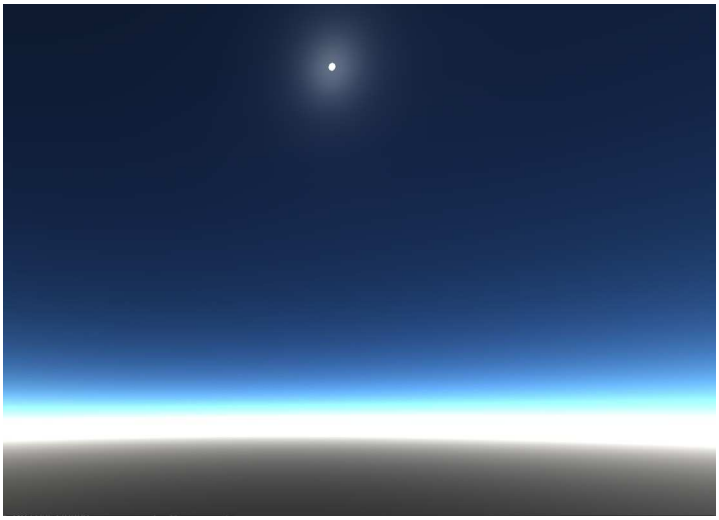
Sl. 4.4: Sivo nebo zbog vrlo velike količine aerosola

Navedeni efekt može se smanjiti povećavanjem broja iteracija korištenih za računanje integrala tlaka ili smanjenjem maksimalne visine atmosfere za računanje, ali također i povećavanjem površinskog tlaka, što je iduća primijećena zanimljivost simulacije.

Mijenjanjem parametra površinskog tlaka atmosfera se ponaša obrnuto intuiciji - pri niskom tlaku prisutno je vrlo jako raspršivanje zraka svjetlosti do te mjere da se plave boje potpuno izgube raspršivanjem te ostaju samo tople boje (Sl. 4.5) - efekt koji bi bio očekivan kod izrazito guste atmosfere. Obrnuto, pri visokom tlaku raspršivanje se događa u puno manjoj mjeri te su jedino vidljive plave boje (Sl. 4.6).



Sl. 4.5: Atmosfera s niskim tlakom



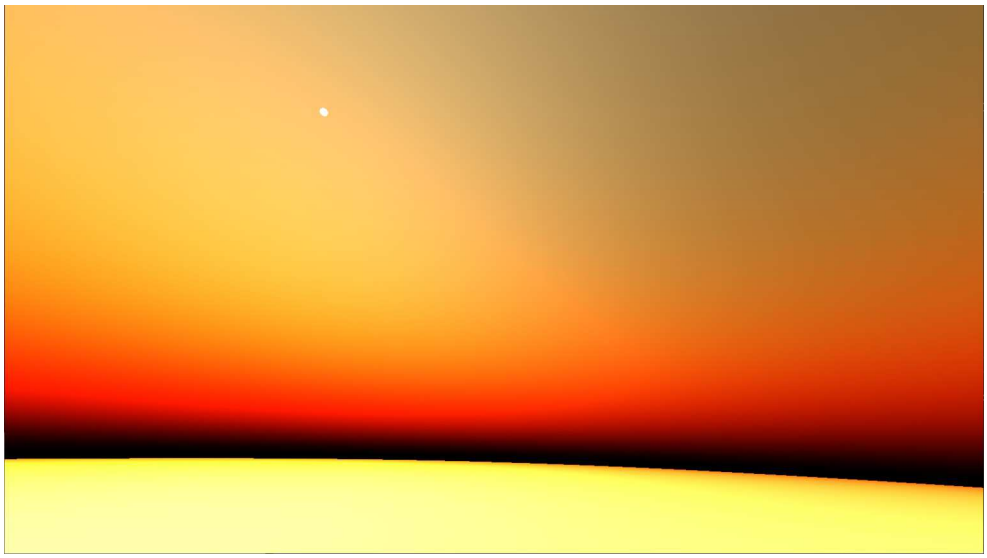
Sl. 4.6: Atmosfera s visokim tlakom

Ova netočnost može se pratiti do izračuna konstante atmosfere K navedenog u formuli (7). Konstanta je obrnuto proporcionalna gustoći čestica na površini planeta, koja je proporcionalna tlaku prema formulama plinske jednadžbe i molarnosti ($pV = nRT$ i $N = nN_A$). Provjera nekonzistentnosti može se izvesti postavljajući gustoću na 0 - optička dubina bila bi beskonačno velika te bi se svjetlost potpuno izgubila raspršivanjem u svakom koraku, iako nema čestica koje bi ju raspršile.

Kako originalni rad iz kojeg je jednadžba preuzeta [3] i drugi pronađeni derivirani radovi [5][6][7][8][9] nisu mijenjali parametre atmosfere koji bi utjecali na K (ili nisu direktno implementirali njezin račun već je samo spominju kao konstantnu vrijednost), mogući zaključak jest da je ta formula rezultat aproksimacija koje uvedu disproporcionalnost, ali zauzvrat svedu rješenje na potreban red veličine za ostale izračune (te nisu namijenjene za proizvoljnu parametrizaciju).

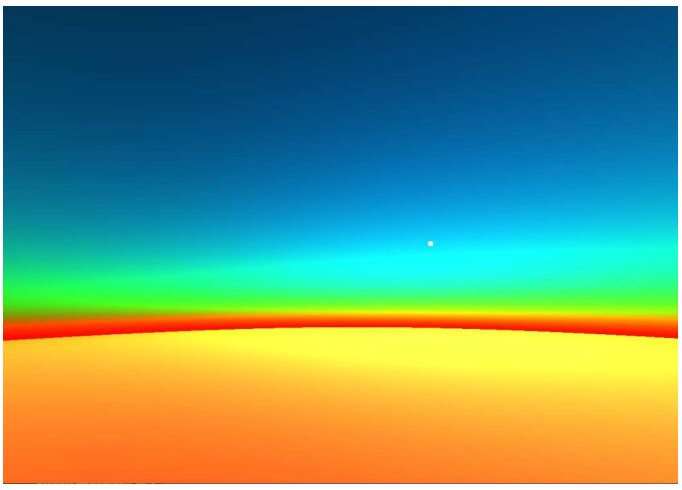
Moguće poboljšanje ovog rada bilo bi otkrivanje izvora ovih aproksimacija, te direktan izračun Rayleighovog optičkog presjeka s točnom proporcionalnošću bez njih.

Još jedan od zanimljivih slučajeva jest kada se tlak jako "smanji" (tj. količina raspršivanja poveća) te se aerosolna komponenta postavi na veliku vrijednost (Sl. 4.7) - moguće je izolirati Tyndallov efekt gdje su jedine zrake koje dolaze do promatrača one crvene koje se ne izgube po putu raspršivanjem (što je također razlog zašto su zalasci Sunca većinom crveni).

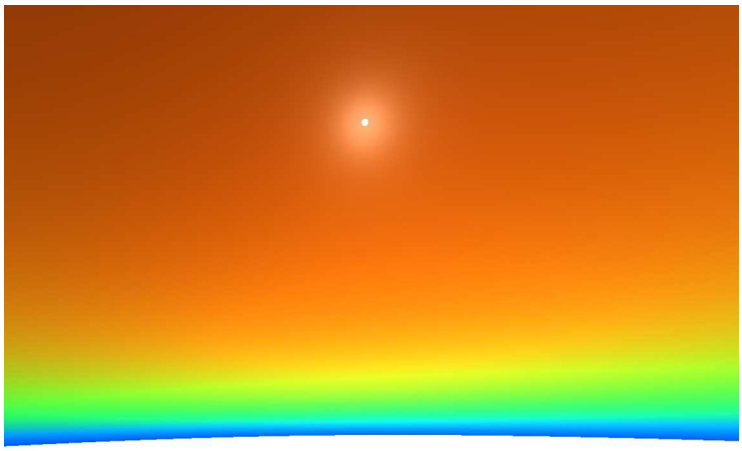


Sl. 4.7: Raspršivanje svjetla pri vrlo velikoj gustoći atmosfere

U simulaciji je također moguće mijenjati valne duljine crvene, zelene i plave komponente svjetlosti. Njihovim razdvajanjem može se uočiti da svaka valna duljina ima određenu visinu na kojoj je ona najviše vidljiva - ovim je moguće dobiti atmosferu gdje je zelena boja također vidljiva jer nije izgubljena u miješanju s ostalima (Sl. 4.8), (Sl. 4.9).

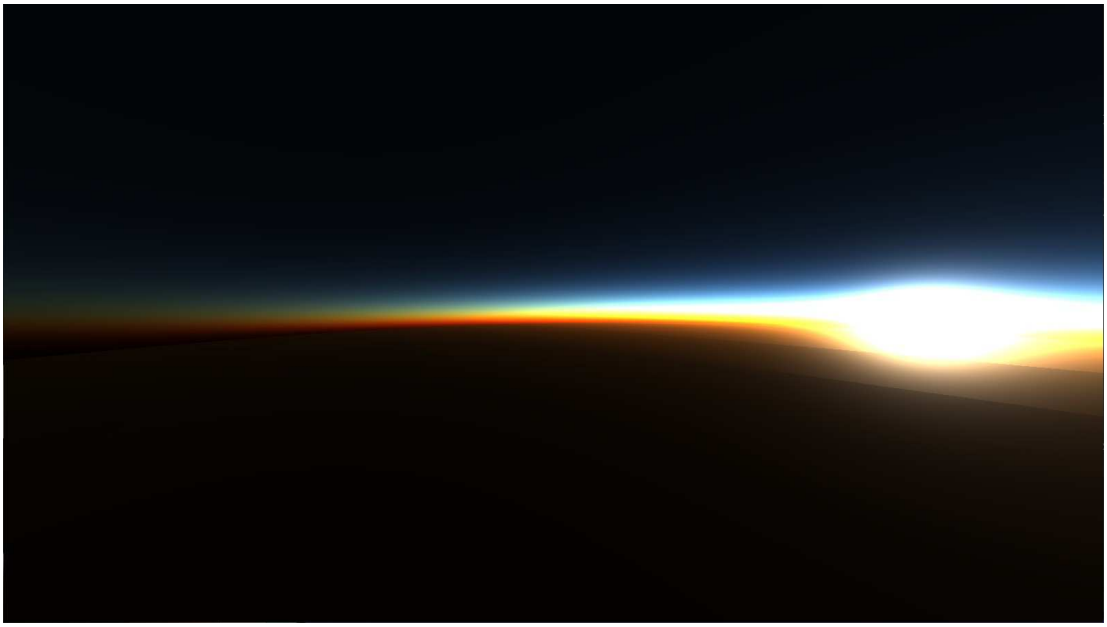


Sl. 4.8: Atmosfera s izraženim valnim duljinama (crvena je veća nego inače a plava manja)

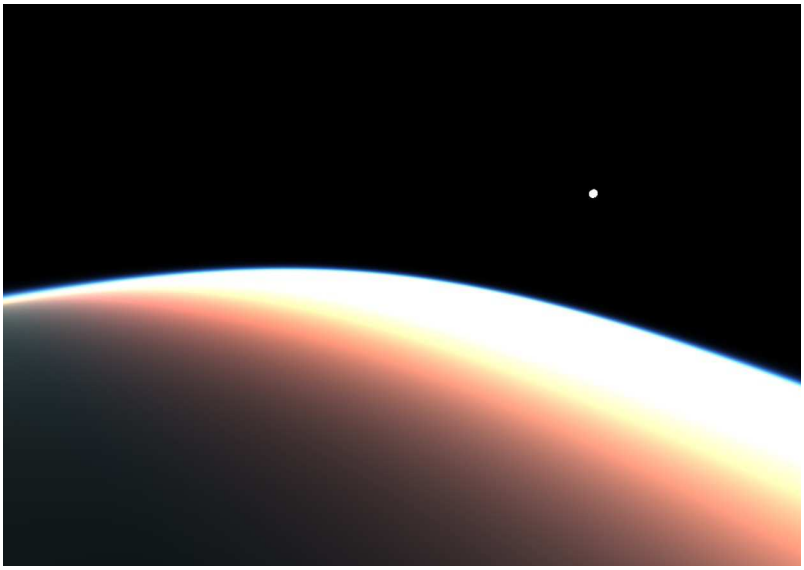


Sl. 4.9: Atmosfera s izraženim i obrnutim valnim duljinama komponenata

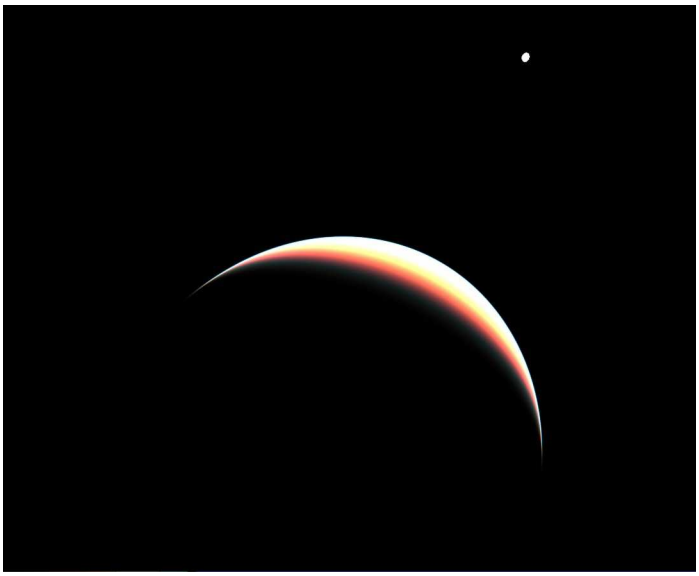
Iako vizualizacija samog planeta nije bila dio cilja ovog rada, također je moguće promotriti atmosferu iz vanjskog pogleda, odbijanje svjetlosti od površinu te višebojne zalaske Sunca pri udaljavanju od planeta (Sl. 4.10), (Sl. 4.11), (Sl. 4.12).



Sl. 4.10: Višebojan izgled atmosfere pri velikoj udaljenosti od površine

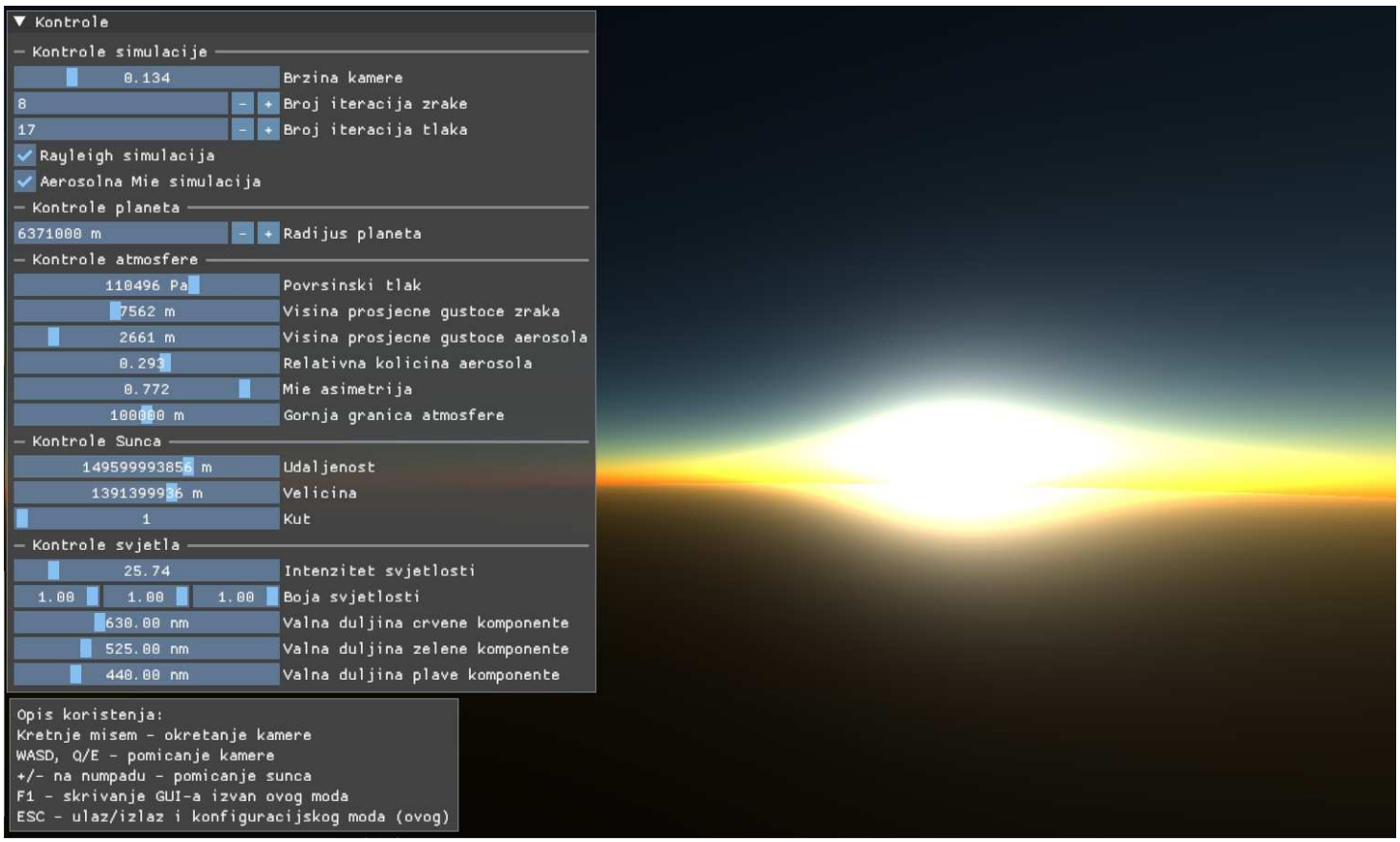


Sl. 4.11: Udaljen pogled na planet

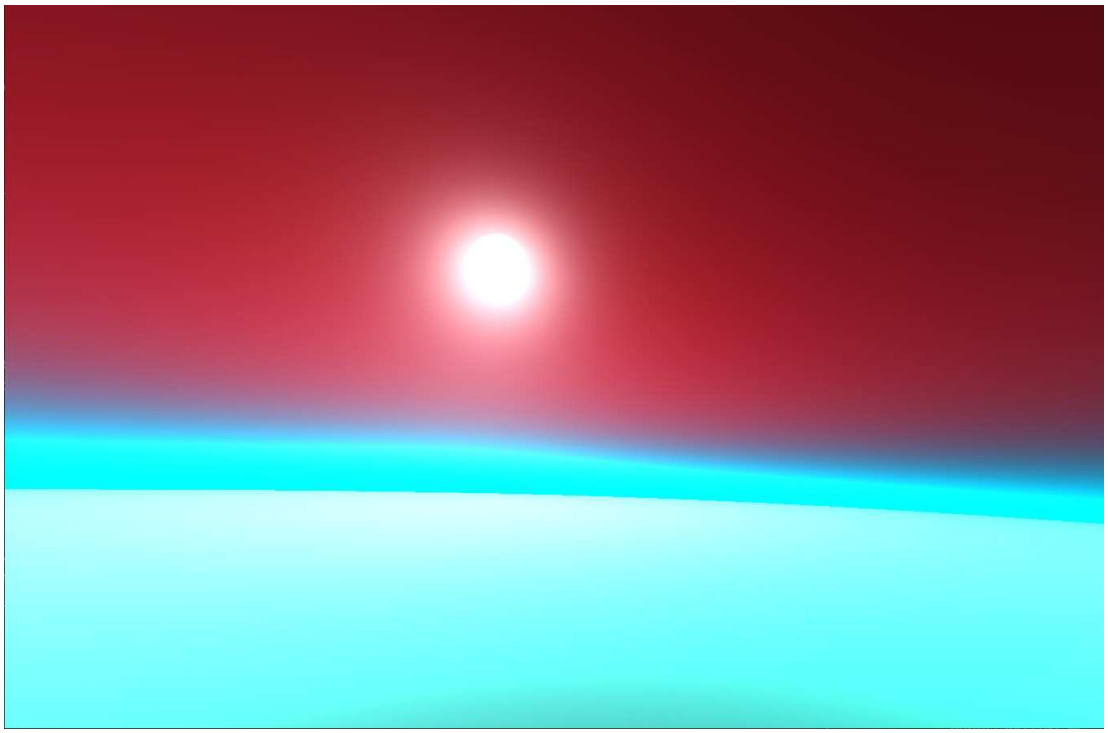


Sl. 4.12: Još udaljeniji pogled

Na sljedećim slikama prikazane su sve opcije korisničkog sučelja (Sl. 4.13) i jedan od rezultata igranja njihovim parametrima (Sl. 4.14):



Sl. 4.13: Korisničko sučelje



Sl. 4.14: Parametri postavljeni na vrlo nerealistične vrijednosti

4.2. Mjerenje performanse

Jedan od ciljeva rada bio je postići simulaciju u interaktivno prihvatljivom vremenu, zbog čega je provedeno mjerenje dvije komponente performanse programa: brzina crtanja slika i računalna zahtjevnost izvođenja programa.

Način na koji se provelo mjerenje brzine jest pomoću standardne C++ biblioteke `std::chrono`. Unutar koda, postavljena je točka mjerenja vremena na početku izvođenja potprograma crtanja te bi se svako izvedeno crtanje inkrementirala brojačka varijabla. Na kraju jednog crtanja provjerila bi se nova vremenska točka, te ako je razlika te točke i točke prošlog ispisa veća od jedne sekunde, program bi ispisao broj crtanja u sekundi te ponovno postavio točke i brojače.

Dodatak funkciji `run` koji omogućuje ispis vremena:

```
void RenderEngine::run(){

	std::chrono::high_resolution_clock frame_clock;
	auto time_point_1 = frame_clock.now();

	int frame_counter = 0;
	double time_counter = 0;

	while (!should_quit){

		// Procesiranje korisničkog inputa
		handle_input();
		process_movement();

		... // Dio koda izostavljen jer zauzima puno mjesta a nije
relevantan za prikaz

		ImGui::Render();
		compute();

		frame_counter++;
		auto time_point_2 = frame_clock.now();

		auto diff =
std::chrono::duration_cast<std::chrono::milliseconds>(time_point_2 -
time_point_1);

		if (diff.count() >= 1000){ // Ispis i novo postavljanje
	std::cout << "Broj crtanja u sekundi: " << frame_counter << "\n";
			frame_counter = 0;
			time_point_1 = frame_clock.now();
		}
	}
}
```

Za mjerenje zahtjevnosti izvođenja korišten je program "Task Manager" (upravitelj zadataka) koji je ugrađen u operacijski sustav Windows. Task manager ima sposobnost čitati postotak opterećenja procesora računala (uključujući i grafičke) u stvarnom vremenu, što može ukazati na tijesne točke i slučaje koji mogu usporiti program.

Kako grafička kartica sadrži nekoliko procesorskih jedinica, za mjerenje opterećenja bit će uzeta ona čije se izmjereno opterećenje vidljivo mijenja pri mijenjanju parametara.

Mjerenja su izvedena na dva računala: Prvo je stolno računalo s odvojenom grafičkom karticom na kojem se također izvodio razvoj programa, a drugo je prijenosno računalo sa slabijim integriranim grafičkim procesorom.

Detalji:

Računalo 1

Grafički procesor: Nvidia GeForce GTX 1650
Količina video RAM-a: 4 GB
Centralni procesor: Intel Core i3-6300 s 4 jedinice brzine 3.70 GHz
Količina RAM-a: 16 GB
Maksimalna brzina osvježavanja monitora: 60 Hz

Računalo 2

Grafički procesor: Intel UHD Graphics 620
Količina video RAM-a: 128 MB
Centralni procesor: Intel Core i7-8550 s 8 jedinica brzine 1.80 GHz
Količina RAM-a: 16 GB
Maksimalna brzina osvježavanja monitora: 60 Hz

Bitno je napomenuti da arhitektura protočnog sustava ne omogućuje pokretanje sjenčara ni crtanje slike brže od one brzine kojom se slike mogu prikazati na zaslon računala. Npr. na stolnom računalu broj crtanja u sekundi imat će gornju granicu na 60, zbog čega je bitno također pratiti srednju opterećenost grafičkog procesora jer će u slučaju čekanja na osvježavanje zaslona biti manja.

Konstante mjerenja:

Stvoreni prozor u svim slučajevima bit će veličine 1920*1080 piksela. Kako količina piksela direktno utječe na broj potrebnih izvođenja simulacije u jednom koraku, korelaciju nije potrebno dodatno istraživati, a izabrana veličina dovoljno je velika za razumnu upotrebu.

U pozadini računala neće biti pokrenuti dodatni programi koji bi mogli bitno utjecati na rezultate.

Mijenjat će se samo brojevi iteracija u sjenčaru i pozicija kamere, a ostali parametri ostat će netaknuti.

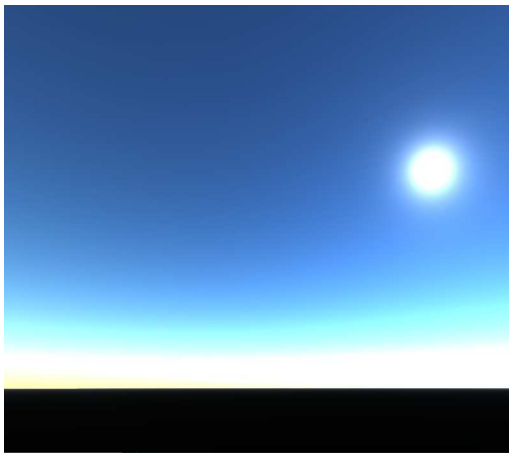
Rezultati su vidljivi u sljedećoj tablici (4.1):

Tablica 4.1: Rezultati mjerenja

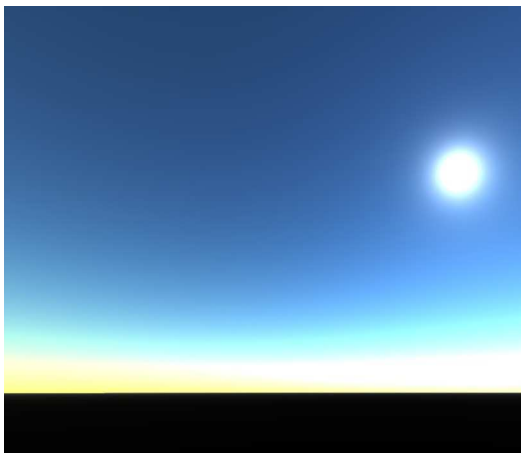
Računalo	Broj iteracija glavne zrake	Broj iteracija računanja tlaka	Pozicija i smjer kamere	Prosječan broj prikaza u sekundi	Prosječno opterećenje GPU-a
1	10	10	U atmosferi, prema horizontu	60	63%
1	25	10	U atmosferi, prema horizontu	48	99%
1	10	25	U atmosferi, prema horizontu	42	99%
1	10	10	Van atmosfere, gledajući rub planeta	60	40%
1	25	25	U atmosferi, prema horizontu	19	99%
1	5	8	U atmosferi, prema horizontu	60	40%
2	10	10	U atmosferi, prema horizontu	10	99%
2	5	5	U atmosferi, prema horizontu	31	99%
2	12	4	U atmosferi, prema horizontu	24	99%
2	4	12	U atmosferi, prema horizontu	14	99%

Iako se simulacija može pokrenuti u interaktivnim vremenima, uočljivo je kako je vrlo zahtjevna za grafički procesor računala.

Smanjenjem broja iteracija olakšava se opterećenje grafičke kartice, no to također utječe na točnost rezultantne slike. Iteracije računanja tlaka utječu više na zahtjevnost programa i također puno više na kvalitetu slike. Na slikama (4.15), (4.16) i (4.17) mogu se primijetiti razlike (brojevi označuju ("broj iteracija računanja zrake, broj iteracija računanja tlaka")).



Sl. 4.15: Simulacija s velikim brojem iteracija (17,17)



Sl. 4.16: Smanjen samo broj iteracija zrake (5,17)

Sl. 4.17: Smanjen samo broj iteracija tlaka (17,5)

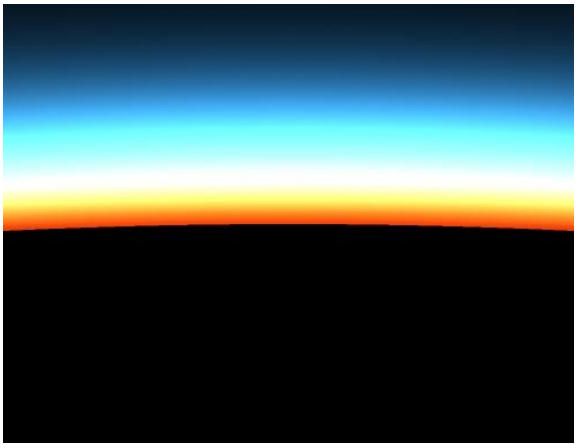
Jasno je vidljiv gubitak kvalitete kod smanjenja broja iteracija računanja tlaka.

Mnogo sličnih radova implementiraju neku vrstu optimizacije u simulaciji kako bi smanjili opterećenje grafičkog procesora, te je bi jedno od mogućih poboljšanja ovog rada bila optimizacija brzine simulacije.

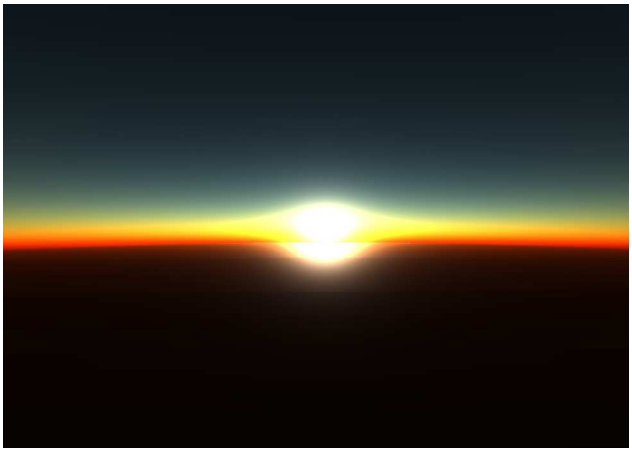
4.3. Usporedba sa sličnim radovima

Ovo potpoglavlje sadržavat će razne dobivene slike iz drugih radova atmosferskog raspršivanja, uz pokušaje njihove rekreacije unutar programa ovog rada (sa što manjom konfiguracijom parametara) te analize razlika.

Prva usporedba bit će na Nishitinom radu [3]. Iako su metode prikazivanja rezultata simulacije vrlo vjerojatno potpuno različite, osnovne korištene formule i algoritmi isti su. Slike (4.18) i (4.19) prikazuju usporedbu pri pretpostavljenom zalasku Sunca u originalnoj slici, a (4.20) i (4.21) prikazuju udaljen pogled na planet (bez teksturiranja u originalnom radu)

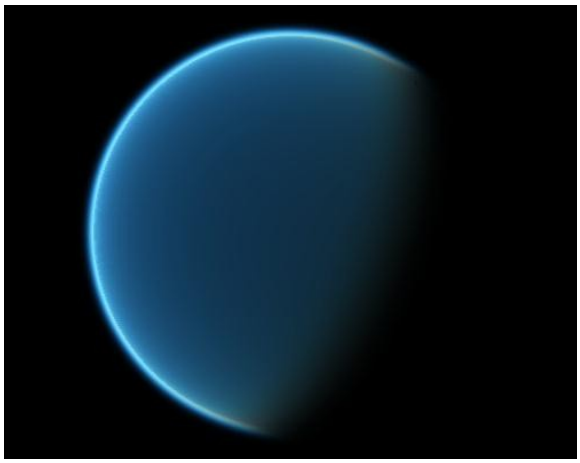


Sl. 4.18: Atmosfera iz Nishitinog rada. Slika je preuzeta s [10]

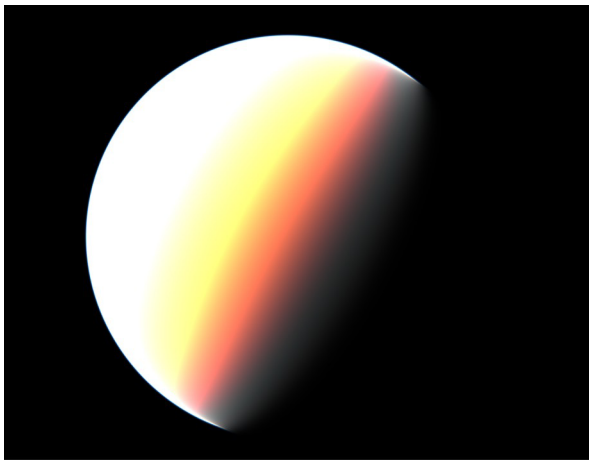


Sl. 4.19: Pokušaj rekreacije unutar programa

Vidi se sličnost rezultata pri obzoru, uz razlike da raspršivanje po atmosferi ovog rada puno brže opada s visinom, te je doprinos aerosola puno veći čak i s niskim postavkama - ovo vjerojatno proizlazi iz direktnog zbrajanja Rayleighovog i aerosolnog doprinosa.



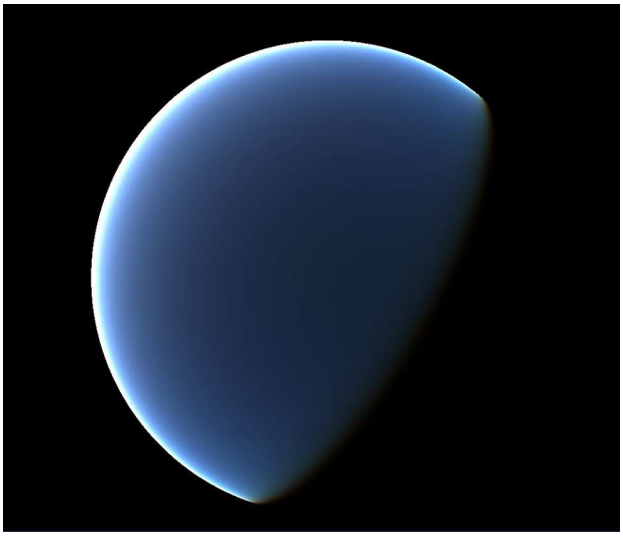
Sl. 4.20: Planet iz Nishitinog rada. Također preuzeto s [10]



Sl. 4.21: Prikaz planeta u programu s vidljivom greškom

Razlike u prikazu površine planeta očekivane su budući da točan prikaz površine nije bio cilj ovog rada. Primjećuje se kako je površina planeta u ovom radu skoro potpuno bijela pri direktnoj svjetlosti Sunca, što se može pripisati nedostatku metode točaka uzorkovanja. Početna svjetlost zrake u simulaciji postavljena je na `vec3{50,50,50}`, dok se sve vrijednosti veće od 1 na zaslonu prikazuju kao bijela boja. Velika količina Sunčeve svjetlosti potrebna je kako bi atmosfera poprimila boju (budući da je i u pravom svijetu Sunčeva svjetlost toliko jaka da je opasno gledati direktno u nj), no kako direktno difuzno odbijanje često očuva većinu svjetlosti rezultat lagano prijeđe preko granice, rezultirajući bijelom bojom. U idealnom slučaju, ta svjetlost bi se svejedno izgubila raspršivanjem putujući od površine do promatrača, no zbog (relativno) malog broja točaka uzorkovanja to se ne dogodi jer većina njih promaši nisko područje atmosfere s visokom gustoćom.

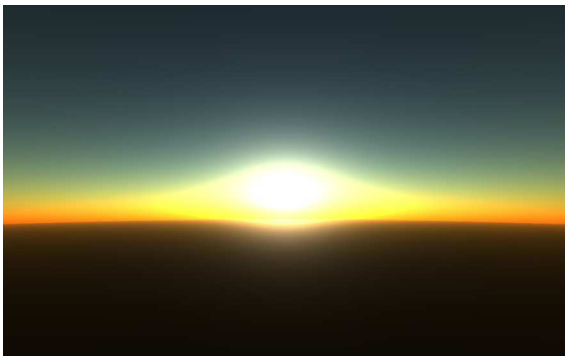
Povećavajući broj iteracija uzorkovanja tlaka i smanjujući maksimalnu razinu atmosfere (tako da je većina točaka u području velike gustoće) ovaj efekt se popravi i planet čak dobije lagano plavo obojenje (osnovna difuzna boja površine je tamnosiva) (Slika 4.22).



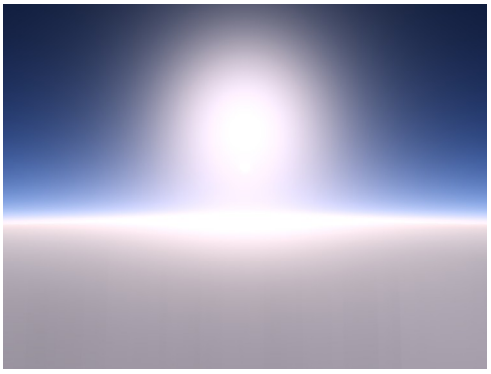
Sl. 4.22: Prikaz planeta uz popravljene parametre

Slijedi još par usporedbi s drugim radovima (koji također implementiraju formule iz Nishitinog rada): Slike (4.23), (4.24), (4.25) i (4.26) prikazuju usporedbu s implementacijom Bartholomeja Paleologua [7]. Ta aplikacija može se interaktivno pokretati na internet pregledniku i također omogućuje mijenjanje parametara, neki od kojih su prisutni i u ovom radu, što omogućuje laganu usporedbu u "sličnim" uvjetima. Lijeve slike uzete su kao isječci zaslona iz aplikacije.

Sl. 4.23: Original



Sl. 4.24: Pokušaj rekreacije

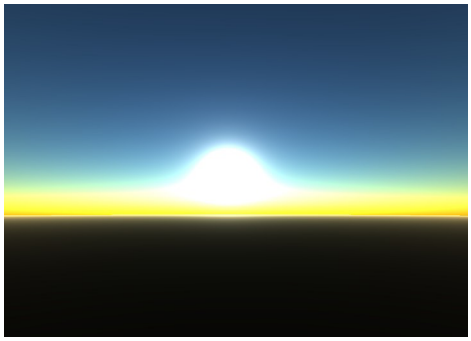


Sl. 4.25: Original

Sl. 4.26: Pokušaj rekreacije

Prikazi (4.27) i (4.28) usporedbe su s radom Seana O'Neila [8], koji je također bio jedna od inspiracija ovog projekta. Na obama se primjećuje prelijevanje bijele boje zbog već opisanog problema velikih komponenti Sunčevog svjetla. O'Neilovo rješenje implementira algoritam HDR (Sl. 4.29) koji eksponencijalno skalira boje s konstantom obično postavljenu na najveću komponentu, koji bi također mogao biti potencijalno poboljšanje ovog rada.

Sl. 4.27: Izvorni rezultat. Preuzet s [8]



Sl. 4.28: Rekreacija slike u ovom radu

Sl. 4.29: Izvorni rezultat s HDR-om (također preuzet s [8])

Zaključak

Postignuta je implementacija simulacije raspršivanja zraka svjetlosti s pomoću Rayleighovog modela i aproksimacije aerosolnog Mie raspršivanja. Vidljivi rezultati simulacije većinom odgovaraju prirodnim pojavama, uz poneke objašnjive artefakte.

Zbog korištenja sjenčara na grafičkom procesoru moguće je izvoditi simulaciju u interaktivnim vremenima (iako još uvijek zahtjevnim za računalo) te mijenjati mnoge parametre simulirane atmosfere uz rezultate koji su odmah vidljivi.

Usporedbom s drugim, sličnim modelima simulacije primjećuju se vrlo slični rezultati što potvrđuje da ova implementacija odgovara pozadinskoj teoriji raspršivanja.

Analizom artefakata i korištenih formula pronađena je mana u teoriji koja pretpostavlja da se gustoća čestica na površini atmosfere neće mijenjati. Ako se ona ipak promijeni, atmosfera se ponaša nekonzistentno s logikom u pravom svijetu, označavajući da je potrebno implementirati drugačije formule od onih u radu Nishite i ostalih [3] ukoliko je cilj mijenjati parametre koji utječu na gustoću atmosferskih čestica na površini (tlak i temperatura), ili ostaviti površinsku gustoću kao konstantu i implementirati skaliranje drugdje.

Literatura

[1] Chance, K., *Earth and Planetary Sciences 238*, poglavlje 8, (2012). Poveznica: https://lweb.cfa.harvard.edu/~kchance/EPS238-2012/class_notes/07-EPS-238-2012.pdf;

[2] Cornette, W.M., Shanks, J.G. *Physically reasonable analytic expression for the single-scattering phase function*, Applied Optics, 31, 16 (1992), str. 3152-3160. Poveznica na stranicu pristupa: https://www.researchgate.net/scientific-contributions/William-M-Cornette-50963985

[3] Nishita, T., Sirai, T., Tadamura, K., Nakamae, E. *Display of the Earth Taking into Account Atmospheric Scattering.* Proceedings of the 20th annual conference on Computer graphics and interactive techniques. (1993), str. 175.-182. Poveznica: http://nishitalab.org/user/nis/cdrom/sig93_nis.pdf

[4] Stull, R. *Practical Meteorology*, poglavlje 22.4.1., mrežna verzija. Poveznica: https://geo.libretexts.org/Bookshelves/Meteorology_and_Climate_Science/Practical_Meteorology_(Stull)/22%3A_Atmospheric_Optics/22.03%3A_New_Page

[5] Simmons, B., implementacija Nishitinog modela neba "*NishitaSky*". Poveznica: https://github.com/BenSimonds/NishitaSky

[6] Kim, D., implementacija Nishitinog modela neba "*Atmospheric Scattering*". Poveznica: https://github.com/newpolaris/Atmospheric-Scattering

[7] Paleologue, B. implementacija atmosferskog raspršivanja "*Volumetric Atmospheric Scattering*". Poveznica: https://github.com/BarthPaleologue/volumetric-atmospheric-scattering

[8] Nvidia, *GPU Gems* 2, poglavlje 16 (2005): O'Neil, S. *Accurate Atmospheric Scattering.* 2. izdanje. Poveznica: https://developer.nvidia.com/gpugems/gpugems2/part-ii-shading-lighting-and-shadows/chapter-16-accurate-atmospheric-scattering

[9] 阳光真强烈, *Atmospheric Scattering*. (2015) Poveznica: https://gwb.tencent.com/community/detail/103633
Napomena: formule, slike i kod na stranici nisu mogli biti učitani no informacije su svejedno korisne za problem

[10] Lerios, A. *CS 348C Readings,* Stanford Computer Graphics Laboratory (1995) (repozitorij slika s rada [3]). Poveznica: https://graphics.stanford.edu/courses/cs348c-95-fall/reader/display/

Sve stranice pristupljene su 19. 6. 2025.

Sažetak

Simulacije raspršivanja zraka svjetlosti

U ovom radu opisani su osnovni modeli svjetlosnog raspršivanja u atmosferi te njihove formule i aproksimacije. Objašnjena je moguća implementacija aproksimacije Rayleighovog i Mie modela raspršivanja svjetla u atmosfere u računalnoj simulaciji.

Opisan je algoritam za izvršavanje simulacije, te metoda numeričkog računanja integrala potrebnog za dobivanje intenziteta ulazne i izlazne svjetlosti nakon raspršivanja.

Navedeni su (i ukratko opisani) alati potrebni za programsku implementaciju simulacije u jeziku C++ i sučelju Vulkan te osnovni koraci implementacije algoritma.

Rezultati dobiveni pomoću simulacije analizirani su s obzirom na prirodne pojave, te su analizirani utjecaji nekih parametara simulacije. Rezultati su također uspoređeni sa sličnim projektima.

Izmjerene su performanse programa u nekoliko različitih slučajeva.

Ključne riječi: atmosfersko raspršivanje, svjetlosne zrake, Rayleighov model, Mie model, simulacija atmosfere, prikaz atmosfere, računalna grafika, C++, Vulkan

Summary

Simulation of atmospheric scattering

In this paper, basic models of light ray scattering in the atmosphere have been described, alongside their formulae and approximations. A possible, approximative implementation of Rayleigh and Mie scattering in a computer simulation has been explained.

The algorithm used for the simulation has been described, as well as a method of numeric integral solving needed for intensity calculation of in and out-scattered light.

Tools used for software implementation of said simulation in the language C++ and the Vulkan interface have been listed (and briefly described) alongside some basic steps of the used algorithm.

The simulation's results have been analyzed in regards to natural phenomenons, and the effects of the simulation's parameters have been analyzed. The results were also compared with similar projects.

The performance of the simulation was measured for various cases.

Keywords: atmospheric scattering, light rays, Rayleigh model, Mie model, atmosphere simulation, atmosphere rendering, computer graphics, C++, Vulkan

Privitak

Izvorni kod projekta nalazi se na sljedećoj stranici:

https://github.com/MarkoLujo/Simulacije-rasprsivanja-svjetlosti

Instalacija programa

Za pokretanje programa moguće je preuzeti već izgrađenu izvršnu datoteku koja se nalazi u polju "Releases". Datoteka je izgrađena za operacijski sustav Windows 64-bitne arhitekture.

Izgradnja projekta

Za ručnu izgradnju izvršnog programa potrebno je na računalo instalirati sustav izgradnje CMake (https://cmake.org/) i razvojne alate za Vulkan (https://vulkan.lunarg.com/), te preuzeti izvorni kod ovog projekta s GitHuba.

Pokretanjem CMake aplikacije nad glavnom datotekom repozitorija trebao bi se izgraditi projekt koji onda može prevesti cijeli izvorni kod u izvršnu datoteku. Izgradnja projekta neće uspjeti ako lokacija instalacije Vulkan razvojnih alata nije zapisana u okolišnu varijablu (environment variable) sustava.

Ako se mijenja kod u sjenčaru, potrebno ga je ručno prevesti i SPIR-V datoteku. Direktorij "shaders" sadrži skriptu koja može pozvat prevoditelja iz Vulkan SDK-a i kratke upute za korištenje.

Upute za korištenje programa

Kretnje mišem okreću kameru, dok se njezina pozicija mijenja s tipkama WASD i QE. Tipke "+" i "-" na brojčanom dijelu tipkovnice mijenjaju poziciju Sunca.

Pritiskom na gumb "ESC" prikazat će se korisničko sučelje s opcijama za mijenjanje parametara i dodatnim uputama, a gumb "F1" skriva tekst koji daje osnovne upute.

Zatvaranje aplikacije moguće je pritiskom na gumb "x" prozora, ili kombinacijom tipki "Alt" i "F4".

Poveznice na druge korištene alate:

VkBootstrap: https://github.com/charles-lunarg/vk-bootstrap

Dear Imgui: https://github.com/ocornut/imgui