Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žiga Lesar

Sledenje poti za neposredno upodabljanje volumetričnih podatkov s spletnimi tehnologijami

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matija Marolt

Ljubljana, 2018

Zahvala

Zahvaljujem se mentorju izr. prof. dr. Matiji Maroltu in as. dr. Cirilu Bohaku za tehnično svetovanje med izdelavo te magistrske naloge. Zahvaljujem se tudi družini za podporo in vzpodbudo. Posebna zahvala gre moji najdražji, ki je verjela vame, me motivirala in poslušala od vsega začetka.

Žiga Lesar, 2018

ᔑᒋᓱᓕᓂᒋ ᐊᓂ ᐃᒋᒌ ᔑᓱᐊᓂᓯᒋ ᐊᓂ ᓴᐱᐦᓯᒋᓂᐦ ᐁᓱᓵ,
ᓱᔑᐃᐊᐧᐁᔑᔭ ᐃᒋᓱᐊ ᓱᔑ ᒋᔑᐊᓯᒋᓴᒋᓂᐦᐃ ᓂᐁᒋᔑᒋᓯ ᐧᐃᓱᔑᐃᐧᐁᓂᓂ
ᐃᓯᐧᔭᐧᐃᓱᔑᓴᓴᒋᓯ ᓂᔑᔑ ᓯᐊ ᓱᐊ ᔑᔭᒋ ᐃᔑᐃᔑᐊ ᓂᔑᔑ ᔑᒋ
ᓯᒋᐃᐃᐁᓂᔑᐊᒋᓯᐢ ᔑᐃ ᔑ ᔑᒋᓯᓯᒋᐃᐃ ᓂᔑᔑᐃᐊᒋᔑᔭᒋ
ᐃᓯᐧᔭᐧᐃᓱᔑᓴᓴᒋᓯ ᓂᔑᔑᒋ ᔑᓯᔑᔑᒋᓯ ᔑᐢ ᐃᐃᒋᔑᓯᓂ ᐁᔑᔭᒋᓯᓯ...

ᐁᔑᔭ ᓂᔑᓯᓴᔑᓂᐦ

Kazalo

Povzetek

Naslov: Sledenje poti za neposredno upodabljanje volumetričnih podatkov s spletnimi tehnologijami

Upodabljanje volumetričnih podatkov je temeljnega pomena v znanosti in mnogih industrijskih panogah, a je vse prej kot trivialno. Še težje dosegljivo je interaktivno upodabljanje v realnem času. Dandanes večina aplikacij uporablja primitivne metode, da dosegajo želene hitrosti izvajanja, poleg tega pa so pogosto dostopne le na določenih platformah. Spletna revolucija nam je v zadnjih letih omogočila, da prek spletnih brskalnikov dostopamo do zmogljive grafične strojne opreme in s tem gradimo sodobne grafične aplikacije na platformno agnostičen način. V tem delu zato združujemo najsodobnejšo spletno tehnologijo z najnovejšimi napredki s področja vizualizacije volumetričnih podatkov v predstavitveni spletni aplikaciji za interaktivno, realnočasovno in fizikalno pravilno upodabljanje volumetričnih podatkov poljubnega izvora, ki jo je moč izvajati na široki paleti namiznih in mobilnih naprav. Uporabljene metode so karseda splošne, tako da ne postavljajo omejitev glede interakcije, scene, kamere in osvetljave. Za namene razširljive implementacije predlagamo nov cevovodni model z neposredno podporo stohastičnim metodam, ki nam omogoča enostavno razširjanje obstoječih in hitro preizkušanje novih metod upodabljanja. S tem delom premoščamo trenutni vseprisotni razmak med teorijo in prakso na širokem področju uporabe.

Ključne besede

sledenje poti, WebGL, upodabljanje volumetričnih podatkov

Abstract

Title: Path tracing for direct volume rendering with web technologies

Rendering of volumetric data is of great significance in numerous fields of science and technology, although it is far from being a trivial task. Interactive and real-time rendering is even more difficult to achieve. Majority of applications nowadays employ primitive methods to reach high execution speeds, and furthermore, they are often accessible only on specific platforms. The web revolution in recent years enabled us to use web browsers to access powerful graphics hardware and in turn build modern graphical application in a platform-agnostic manner. Therefore, in this work we combine state-of-the-art web technology with the latest advancements in volume rendering in a proof-of-concept web application for interactive, real-time and physically correct rendering of volumetric data of arbitrary origin that runs on a wide variety of desktop and mobile devices. The methods used are as general as possible so as to not impose any restrictions on interaction, scene, camera and lighting. With extensibility of the implementation in mind we propose a new pipeline model with a direct support for stochastic methods, which allows for simple extension of existing and fast testing of new rendering methods. With this work we bridge the currently ubiquitous gap between theory and practice in a wide range of use cases.

Keywords

path tracing, WebGL, volume rendering

Poglavje 1

Uvod

Človeški vidni sistem je nepogrešljivo orodje za zaznavanje okolice. Omogoča nam, da zaznamo svetlobo, njeno barvo in intenziteto, nato pa te informacije pretvori v nam znane pojme: najprej v ploskve in robove, s tem pa v oblike, predmete, obraze in osebe. V celotnem procesu na eni strani stoji človek, ki zaznava slike in iz njih razpoznava predmete in pomen, na drugi strani pa svetloba, ki s širjenjem po prostoru pridobiva in prenaša informacije o okolici do človeških oči. Človeštvo se je dolgo trudilo razumeti njeno obnašanje in karakteristike, toda še vedno nimamo odgovorov na vsa naša vprašanja. Kljub temu pa smo z dolgotrajnimi raziskavami odkrili ogromno znanja in z njim naše razumevanje dvignili na visoko raven.

S praktičnega vidika so nam nova spoznanja o svetlobi omogočila zajem slik in njihovo dolgotrajno shranjevanje. Omogočila so nam podajanje informacij na hiter, enostaven, predvsem pa razumljiv in intuitiven način. S pojavom računalnikov smo ustvarili nova orodja za digitalno shranjevanje slik in njihovo reproduciranje na zaslonu ali papirju. Računalnike smo začeli uporabljati ne samo za hranjenje slik, temveč tudi za njihovo ustvarjanje. Tako se je pojavilo novo področje računalništva, ki ga imenujemo *računalniška grafika*. Metode, razvite za prikaz točk, črt, ploskev in drugih kompleksnejših geometrijskih objektov so svojo uporabno vrednost izkazale na področjih znanosti, strojništva, medicine, kemije, fizike in še marsikje drugod. Računalniška

grafika kot najbolj temeljno orodje služi metodam za vizualizacijo podatkov, ki jih dobimo kot rezultat simulacij ali pa jih zajamemo iz realnega sveta. V strojništvu se uporablja pri načrtovanju predmetov in procesov, v kemiji in fiziki nam pomaga razumeti kemijske in fizikalne pojave, v medicini pa lahko vidno spremljamo potek operacij ali pa jih lažje načrtujemo vnaprej. Z gotovostjo lahko trdimo, da skoraj vsakršna dejavnost dandanes za svoje delovanje potrebuje podporo računalniške grafike.

Upodabljanje (angl. rendering) je en izmed najbolj temeljnih postopkov na področju računalniške grafike. V najširšem pogledu gre za prikaz podatkov v vizualni obliki, v računalništvu torej najpogosteje s svetlobo na zaslonu. V ožjem pogledu gre za sintezo dvodimenzionalnih slik iz opisa tridimenzionalne scene, torej z objekti in materiali. Zaradi enostavne obravnave in mnogih prikladnih lastnosti so se kot najbolj osnovni geometrijski gradniki uveljavili točka, daljica in trikotnik, ki jih sestavljamo v *poligonske modele*. V sceno nato postavimo izvore svetlobe in kamero ter z izbrano metodo upodabljanja izračunamo sliko, ki jo zajame kamera. To lahko storimo na več načinov, pogosto pa je zaželjeno, da je slika čim bolj fizikalno pravilna in posledično fotorealistična ter da nam poda čim več relevantnih informacij. Klasična fotografija zaradi svoje obče prisotnosti in dolgotrajne tradicije postavlja smernice in ideale na tem področju, zato sem uvrščamo tudi nekatere vizualne efekte, ki sicer predstavljajo odklon od matematičnega ideala, na primer globinsko ostrino (angl. depth of field, DOF), zabrisanost gibanja (angl. motion blur) in kromatično aberacijo.

Poligonski modeli v praksi pogosto niso primerni za predstavitev objektov, zato na mnogih področjih naletimo tudi na *volumetrične podatke*, diskretne opise tridimenzionalnih objektov, ki so v nasprotju s poligonskimi modeli sposobni predstavljati tudi notranje strukture predmetov in ne le ploskev, ki jih obdajajo. Včasih so podani v obliki oblaka točk (angl. point cloud), najpogosteje pa srečamo take v obliki tridimenzionalnih slik, ki jih dobimo kot rezultat simulacij na regularni mreži ali pa jih zajamemo iz realnega sveta s posebnimi napravami, denimo za računalniško tomografijo, magnetno re-

sonanco in tridimenzionalni ultrazvok. Volumetrične podatke lahko najprej pretvorimo v poligonske modele ali pa jih upodobimo brez pretvorbe, pri čemer gre za t. i. *neposredno upodabljanje*. V slednjem primeru ne moremo uporabiti običajnih metod za poligonske modele, temveč moramo te metode razširiti in posplošiti. Neposredno upodabljanje volumetričnih podatkov je zato tudi precej bolj računsko zahtevno kot upodabljanje poligonskih modelov.

Bistvenega pomena za izgled končne slike je osvetlitev scene. Za hitre izrise se običajno zatečemo k enostavnim metodam, ki so pogosto brez fizikalne podlage, izvajajo pa se izredno hitro, zato so primerne za interaktivno vizualizacijo, ne pa tudi za filme ali znanstveno vizualizacijo. Hitrost metod je tesno povezana z obsegom simulacije svetlobe, ki je lahko lokalen ali globalen. Globalna osvetlitev, ki v nasprotju z lokalno upošteva celotno sceno in ne le ploskve, ki jo osvetljujemo, temelji na fizikalni pravilnosti in teoriji širjenja svetlobe, zato je tudi občutno računsko zahtevnejša od lokalne. Interaktivna vizualizacija poligonskih modelov z globalno osvetlitvijo je s hitrim tehnološkim napredkom mogoča le zadnjih nekaj let, metode za volumetrične podatke pa so še v začetku razvoja.

Za izvajanje metod računalniške grafike potrebujemo ogromno računsko moč, ki mnogokrat presega sposobnosti običajnih centralnih procesnih enot. Za potrebe hitrega procesiranja se je razvila posebna *grafična strojna oprema*, ki je posebej namenjena izvajanju operacij v računalniški grafiki. Dandanes je običajno v obliki grafičnih kartic ali pa je integrirana v centralno procesno enoto. V bistvu gre za posebne podatkovno-pretokovne računalnike, ki svojo računsko moč črpajo iz vzporednega procesiranja v zelo širokem obsegu. S starejšo grafično strojno opremo je bilo moč upodabljati le na točno določen način, z močno potrebo in tehnološkim napredkom pa se je razvila tudi taka, ki jo je moč programirati in tako uporabljati tudi za reševanje bolj splošnih problemov, ne nujno upodabljanja.

Pomemben faktor, ki močno vpliva na vsesplošno uporabo metod v praksi, je tudi njihova dostopnost na široki paleti naprav in operacijskih sistemov.

Vseprisotnost spleta in spletnih brskalnikov je izredno poenostavila razvoj spletnih aplikacij, ki se izvajajo na platformno agnostičen način. Tako se lahko ista programska koda izvaja na namiznih ali mobilnih napravah, za pravilno izvajanje pa poskrbi že sam spletni brskalnik. Hiter razvoj mobilnih naprav je v zadnjih letih omogočil tudi vgradnjo grafične strojne opreme. Strojni opremi je sledila še programska podpora v spletnih brskalnikih in s tem možnost izkoriščanja vzporednega procesiranja prek platformno agnostične programske kode. Široka uporabnost in dostopnost te tehnologije je razlog za to, da je sodobna računalniška grafika v času nastajanja tega dela na voljo na večini mobilnih naprav.

1.1 Problem in rešitev

Globalna osvetlitev v vizualizaciji volumetričnih podatkov je v času nastajanja tega dela težko dosegljiv ideal. Obstoječe metode so zaradi velike količine podatkov in zahtevne simulacije svetlobe nedopustno počasne, njihove implementacije pa za svoje hitrejše izvajanje prepogosto izkoriščajo posebnosti sistemov, na katerih se izvajajo. Večina implementacij je tesno vezanih na določen operacijski sistem ali arhitekturo grafične strojne opreme, zato so posledično nedostopne za splošno uporabo, razširitve in izboljšave. Odmevni članki s tega področja svoje uspehe črpajo iz stroge omejitve problema na specifične situacije, splošne rešitve pa ostajajo v senci podpovprečnih rezultatov.

Naš glavni cilj je razvoj rešitve, ki ne izkorišča posebnosti sistema, na katerem se izvaja, temveč je platformno agnostična, dostopna za razširitve in izboljšave ter rešuje splošen problem in ne le ozkega podproblema. Metode za upodabljanje morajo biti predvsem fizikalno pravilne, dovolj splošne za upodabljanje različnih podatkov poljubnega izvora, izvajati pa se morajo v realnem času in s podporo interaktivni vizualizaciji.

V ta namen smo razvili referenčno implementacijo v predstavitveni aplikaciji. Aplikacija temelji na sodobnih spletnih tehnologijah s programskim

vmesnikom WebGL 2.0 za dostop do grafične strojne opreme, zato jo je moč izvajati na široki paleti namiznih in mobilnih naprav. Naše metode za upodabljanje sledijo sodobnim svetovnim trendom v samem vrhu tega področja. Rešitev je splošna in omogoča enostavno razširitev in izboljšavo.

1.2 Cilji

S tem delom želimo doseči sledeče:

- Razviti sodobno spletno aplikacijo za interaktivno in realnočasovno neposredno upodabljanje volumetričnih podatkov z globalno osvetlitvijo. Aplikacijo želimo razviti na platformno agnostičen način za izvajanje na široki paleti namiznih in mobilnih naprav ter s tem razširiti dostopnost sodobnih upodabljalnih algoritmov.
- Najnovejše napredke s področja neposrednega upodabljanja volumetričnih podatkov prilagoditi za delovanje v spletnem brskalniku in s tem pokazati, da so sodobni spletni brskalniki dovolj zmogljivi za izvajanje kompleksnih upodabljalnih algoritmov.
- Razviti enoten, robusten in razširljiv cevovodni model za iterativno upodabljanje ter s tem omogočiti enostavno razširjanje in izboljšavo.

1.3 Struktura dela

V poglavju 2 obravnavamo področje upodabljanja volumetričnih podatkov, razvoj in splošen pregled metod ter trenutno stanje tehnologije. V sledečih poglavjih podamo teoretsko osnovo za naše delo. Za pravilen prikaz slike na zaslonu in pravilno simulacijo svetlobe v navideznem svetu moramo najprej spoznati svetlobo kot pojav, njeno obnašanje in merjenje. To storimo v poglavju 3, kjer neformalno predstavimo problem, ki ga s pričujočim delom rešujemo. Sledijo tri glavna poglavja z uporabljenimi metodami: v poglavju 4 formaliziramo problem in domeno rešitev, v poglavju 5 opišemo algoritmično

realizacijo rešitev za dan problem, na koncu pa v poglavju 6 opišemo še predstavitev lastnosti svetlobe v računalniškem sistemu in implementacijo opisanih rešitev v spletni aplikaciji. V poglavju 7 predstavimo rezultate našega dela, jih ovrednotimo in primerjamo z obstoječimi rešitvami. V poglavju 8 zberemo sklepe, predstavimo doprinos tega dela k razvoju področja upodabljanja volumetričnih podatkov ter navedemo možnosti za nadaljnje izboljšave.

Poglavje 2

Pregled področja

Začetki upodabljanja volumetričnih podatkov segajo v osemdeseta leta 20. stoletja. Danes najbolj razširjeni algoritmi izhajajo iz dela Scotta Rotha [1], ki je z algoritmom *metanja žarkov* upodabljal implicitne ploskve. Nekaj let kasneje so se pojavile prilagoditve za upodabljanje volumetričnih podatkov [2, 3]. Marc Levoy je svojo osnovno metodo za upodabljanje nivojskih ploskev leta 1990 razširil na hkratno upodabljanje volumetričnih podatkov in poligonskih modelov [4]. Približno v istem času se je pojavila še vrsta drugačnih pristopov za upodabljanje volumetričnih podatkov. Med najpomembnejšimi prispevki so uporaba intervalske analize [5], iskanje alternativnih načinov iskanja nivojskih ploskev [6], pretvorba v poligonske modele [7], izkoriščanje računalniške arhitekture za pohitritev [8] in preizkušanje metod, ki izhajajo iz neposredne projekcije [9]. Nekateri pristopi so sčasoma zamrli ali pa so bolj uporabni na drugih področjih upodabljanja, zato se je za neposredno upodabljanje volumetričnih podatkov najbolj uveljavilo prav metanje žarkov in njegove izpeljanke. Različni optični modeli, ki so dandanes v široki uporabi, so zbrani v delu Nelsona Maxa [10]. Njegove metode smo uporabili že v predhodnih delih [11, 12, 13, 14].

Enostavne metode za vizualizacijo volumetričnih podatkov ne modelirajo prave osvetlitve. Veliko količino informacij o prostoru in predmetih v njem dobimo prav iz variacij v svetlosti površin. Dodajanje senc v obstoječe me-

tode je relativno enostavno [15], toda naivne implementacije so lahko počasne in fizikalno nepravilne. Pomembna izboljšava osnovne metode je globoko kartiranje senc (angl. deep shadow mapping) [16], ki je bila prilagojena tudi za izvajanje na grafični strojni opremi [17]. Izboljšana metoda, ki s filtriranjem senc okrepi prostorsko zaznavo, je opisana v [18]. Pregled naprednih osvetlitvenih metod, prilagojenih za izvajanje na grafični strojni opremi, najdemo v [19, 20]. Sodoben in novejši pregled osvetlitvenih modelov je opisan v [21].

Leta 1986 je James Kajiya postavil nov mejnik na področju upodabljanja s svojo *enačbo upodabljanja* [22], ki zelo splošno opiše fizikalno pravilno upodabljanje neprosojnih ploskev. Najbolj razširjen algoritem za njeno reševanje danes poznamo pod imenom *sledenje poti* (angl. path tracing). Sprevidenje visoke računske zahtevnosti je znanstvenike vodilo v raziskovanje alternativnih metod za upodabljanje. Že Marc Levoy je leta 1990 objavil metodo iterativne izboljšave, ki je grobo začetno sliko izrisala hitro, nato pa jo izboljševala skozi čas [23]. Isto idejo izkorišča tudi algoritem sledenja poti, pri katerem gre za uporabo metode Monte Carlo za ovrednotenje integrala znotraj enačbe upodabljanja. V devetdesetih letih je sledilo nemalo izboljšav osnovne ideje, med najpomembnejšimi pa sta dvosmerno sledenje poti (angl. bidirectional path tracing, BDPT) [24] in uporaba algoritma Metropolis-Hastings [25, 26, 27] za generiranje novih poti svetlobe. Podrobnejše opise algoritmov sledenja poti in njihovih izboljšav, ki se dandanes uporabljajo v praksi, najdemo v [28, 29]. Enačba Kajiye je bila kasneje tudi razširjena za upodabljanje volumetričnih podatkov [30, 31], temu pa so sledile izboljšave hitrosti izračunov [32, 33]. Med navedenimi viri izstopa predvsem delo Erica Veacha, na katerem temeljijo praktično vse novejše metode. Veliko metod je prilagojenih za ozko podskupino možnih konfiguracij, denimo podpora le enega vira svetlobe, omejitev heterogenosti podatkov, konstantna konfiguracija osvetlitve, konstantna prenosna funkcija itd. Sodobni prispevki, opisani v [34, 35, 36] skušajo te omejitve odstraniti in s tem predstaviti splošne metode za interaktivno upodabljanje podatkov poljubnega tipa in izvora. Trenutno najsodobnejše metode za upodabljanje volumetričnih podatkov najdemo v

[37].

Vzorčenje je kot temeljni postopek pri upodabljanju z metodami Monte Carlo deležno široke obravnave. Pri reševanju enačbe upodabljanja se pojavlja pri vzorčenju smeri odbitega žarka, pri upodabljanju volumetričnih podatkov pa tudi pri vzorčenju poti fotona. Pri tem nam kot osnovno orodje služi prioritetno vzorčenje [38], s katerim postopek integriranja usmerimo v bolj donosna območja domene. Uporabo prioritetnega vzorčenja smeri odbitega žarka za upodabljanje ploskev obravnava Eric Veach v delu [39], kjer opisuje usmerjanje žarka glede na položaj luči in dvosmerno porazdelitveno funkcijo odbojnosti (angl. bidirectional reflectance distribution function, BRDF) [40]. Pri upodabljanju volumetričnih podatkov podobno vlogo igra fazna funkcija. V praksi se pogosto uporablja Henyey-Greensteinova funkcija [41], katere prioritetno vzorčenje je opisano v [42]. Poleg smeri odbitega žarka moramo vzorčiti tudi dolžino poti, ki jo posamezen žarek opravi brez sipanja. Algoritmično enostavna metoda je t. i. Woodcockovo sledenje [43], ki za vsak žarek vrne razdaljo pred sipanjem. Gre za popolno prioritetno vzorčenje poti, ki je hkrati tudi dosledno in nepristransko. Osnoven algoritem je bil deležen mnogih izboljšav, med katerimi najbolj izstopajo uporaba prostorskih podatkovnih struktur [44, 45] in boljša izraba podatkov, pridobljenih med vzorčenjem [46]. Vzorčimo lahko tudi položaj in smer žarka, ki vstopa v objektiv kamere in s tem simuliramo različne vizualne efekte in svetlobne pojave, ki nastanejo kot posledica nepravilnosti kamere. Primeri različnih modelov, ki se danes uporabljajo v praksi, so opisani v [47, 48].

Metode Monte Carlo v rezultat kot stranski produkt vnesejo nezaželjen šum. Poznavanje specifik upodabljanja je sprožilo tudi razvoj specializiranih filtrov za odstranjevanje šuma iz končne slike in s tem izboljšavo interaktivnosti [49, 50, 51].

Narava metod Monte Carlo za upodabljanje volumetričnih podatkov nam omogoča močno paralelizacijo, saj so tako zaslonske točke kot žarki med seboj neodvisni. Največje pohitritve dosežemo z uporabo grafične strojne opreme. V začetku novega tisočletja je potrošniška grafična strojna oprema postala

dovolj zmogljiva in fleksibilna za izvajanje najenostavnejših upodabljalnih algoritmov [52]. Razvoj boljših implementacij je potekal vzporedno z razvojem boljših metod, hkrati pa so se pojavila številna ogrodja, ki so poenostavila implementacijo in razvoj aplikacij za vizualizacijo, najprej enostavnejša [53], kasneje pa naprednejša, kot so na primer Brigade [54], Nvidia OptiX [55, 56], vizualizacijsko ogrodje VTK [57] in njegova izpeljanka za upodabljanje volumnov rtVTK [58]. VTK sprva ni podpiral grafične strojne opreme, zato je nastalo ogrodje FAST [59], nedavno pa je podporo dobil tudi VTK [60]. Nastala je tudi vrsta posebnih metod, ki so prilagojene za izvajanje na grafični strojni opremi [61, 62, 63].

Glavne težave dosedanjih metod izhajajo iz njihovih implementacij. Večina namreč izrablja zmogljivosti določene strojne opreme, na primer priljubljen programski vmesnik Nvidia CUDA [64, 55], ki je na voljo le na grafični strojni opremi podjetja Nvidia. Poleg tega so praktične vse implementacije dostopne v obliki programskega vmesnika ali knjižnice, ki jo moramo predhodno prevesti za določeno ciljno platformo. Naše predhodno delo je s to problematiko v mislih vključevalo razvoj platformno neodvisne aplikacije z uporabo programskega jezika Java in programskih vmesnikov OpenGL in OpenCL [11]. Izbor tehnologij nam je omogočal izvajanje programa na različnih platformah. Upodabljanje volumetričnih podatkov na mobilnih napravah je dokaj novo področje, saj to še pred dobrimi desetimi leti zaradi omejitev strojne opreme ni bilo mogoče. Večina implementacij je temeljila na modelu strežnik-odjemalec, v katerem je odjemalec zahteval storitve upodabljanja od strežnika, ni pa neposredno upodabljal podatkov. Glede na nedavno obljavljen prispevek [65] ideja še ni zamrla. V delu [66] je opisana aplikacija, ki se v celoti izvaja na mobilni napravi, toda omejena je na upodabljanje nivojskih ploskev. Kasnejši prispevki opisujejo aplikacije, zgrajene na programskem vmesniku OpenGL ES, toda nobena ni platformno agnostična, poleg tega pa metode nimajo pravilne fizikalne podlage [67, 68, 69, 70].

S pojavom tehnologije WebGL, ki je zmogljivosti sodobne grafične strojne opreme prinesla na spletno platformno, so nastale tudi prve prave platformno

agnostične aplikacije za vizualizacijo volumetričnih podatkov. Publikacij, ki omenjajo upodabljanje volumetričnih podatkov s spletnimi tehnologijami in nasploh na mobilnih napravah, je zelo malo, najpomembnejše pa so zbrane v [71]. Dosedanji prispevki na tem področju temeljijo na standardu WebGL 1.0 [72], ki je zgrajen po vzoru programskega vmesnika OpenGL ES 2.0 [73] in je zato dokaj funkcionalno omejen. Poleg tega so metode, opisane v teh prispevkih, zelo osnovne in fizikalno nepravilne, na njih zgrajene aplikacije pa počasne, saj dosegajo le nekaj slik na sekundo [74, 75, 76, 77, 78]. Objave, ki bi uporabljala novejši standard WebGL 2.0 [79], ki temelji na programskem vmesniku OpenGL ES 3.0 [80] ali uporabljala novejše, fizikalno pravilne metode upodabljanja, nismo našli.

Poglavje 3

Svetloba

V tem poglavju opišemo svetlobo, njen nastanek, obnašanje, merjenje, njene različne obravnave, ter zgodovinski potek raziskav in teorij. Pri fizikalnih temah se opiramo na vire [81, 82, 31, 32, 33].

Svetlobo iz različnih zornih kotov obravnavamo na različnih področjih fizike, med najpomembnejšimi pa so optika, elektromagnetizem, radiometrija, fotometrija in kolorimetrija. Optika je veja fizike, ki se ukvarja z značilnostmi in obnašanjem svetlobe ter z interakcijo med svetlobo in snovjo. Svetlobo kot elektromagnetno valovanje obravnavamo na področju elektromagnetizma. Z merjenjem svetlobe in na splošno elektromagnetnega sevanja se ukvarjata fotometrija in radiometrija. Barve in njihovo človeško zaznavanje obravnava kolorimetrija.

Obstaja več fizikalnih teorij, ki opisujejo obnašanje svetlobe, med najpomembnejšimi so *geometrijska optika*, *valovna optika*, *elektromagnetna optika* in *kvantna optika*.

3.1 Zgodovinska obravnava svetlobe

Obnašanje svetlobe je bilo skozi zgodovino deležno veliko obravnave. Do 17. stoletja je bila geometrija glavno orodje za opisovanje svetlobe. Med najbolj znanimi raziskovalci t. i. *geometrijske optike* sta bila Evklid in Isaac

Newton. Newtonov glavni prispevek k optiki je razumevanje svetlobe kot zmesi barvnih komponent. Verjel je v Déscartesove svetlobne delce, ki potujejo v ravnih črtah in se ob stiku s trdnimi snovmi lomijo in odbijajo. S takim geometrijskim pojmovanjem svetlobe ter z odbojnim in lomnim zakonom (Snellov zakon) kot glavnimi orodji geometrijske optike lahko razložimo široko paleto svetlobnih pojavov.

V času Newtonovega življenja je Francesco Grimaldi proučeval uklon svetlobe, ki ga ni znal razložiti. Newton je pojav (nepravilno) pripisal lomu svetlobnih žarkov, Christiaan Huygens pa ga je elegantno razložil s pojmovanjem svetlobe kot valovanja. Newtonova teorija je, delno tudi po zaslugi njegovega ugleda, prevladala, zato geometrijsko optiko pogosto imenujemo tudi *Newtonova optika*. V začetku 19. stoletja sta Thomas Young in Augustin-Jean Fesnel s svojimi eksperimenti (zelo znan je na primer eksperiment z dvojno režo) močno podprla Huygensovo *valovno optiko* in s tem zabila žebelj v krsto Newtonovi optiki. Huygensova teorija je predpostavljala obstoj *etra* - medija, po katerem se širi svetlobno valovanje. Michelson-Morleyjev eksperiment je obstoj takega medija ovrgel in s tem vrgel močan dvom na valovno optiko.

Sredi 19. stoletja je Michael Faraday proučeval polarizacijo svetlobe v povezavi z elektromagnetizmom. Njegovo delo je navdihnilo Jamesa Maxwella, da je bolj natančno začel proučevati elektromagnetizem in svetlobo. V 60. letih 19. stoletja je razvil močno teorijo elektromagnetizma in s svojimi slavnimi enačbami združil pojmovanje svetlobe z elektromagnetizmom. Tako pojmovanje svetlobe imenujemo *elektromagnetna optika*.

Nekaterih pojavov, predvsem fotoelektričnega pojava in diskretnih spektralnih črt, Maxwellov model ni mogel razložiti. Na začetku 20. stoletja je Max Planck raziskoval sevanje črnega telesa in pri tem predpostavil, da se lahko energija svetlobnega valovanja spreminja le v diskretnih intervalih oz. *kvantih*. Idejo t. i. kvantizacije elektromagnetnega polja sta kasneje potrdila še Albert Einstein z razlago fotoelektričnega pojava in Niels Bohr s svojim modelom atoma. S fotoelektričnim pojavom je Einstein obudil pojmovanje

svetlobe kot toka delcev, ki jih je Gilbert Lewis poimenoval *fotoni*. Taka obravnavo svetlobe, imenovana *kvantna optika*, je danes široko priznana teorija, ki enotno razloži elektromagnetno optiko in kvantne pojave.

3.2 Svetlobni pojavi

Štiri zgoraj opisane obravnave svetlobe lahko predvidijo različne svetlobne pojave. Že z geometrijsko optiko lahko pravilno opišemo perspektivo, odboje svetlobe od gladkih površin (angl. reflection), razpršitev svetlobe na grobih površinah (angl. diffusion), lom svetlobe pri prehodu med snovmi z različnimi lomnimi količniki (angl. refraction), dvolomnost (angl. birefringence) in sipanje svetlobe v sodelujočem mediju (angl. scattering). Valovna optika lahko dodatno opiše določene pojave, ki so neposredno povezani s superpozicijo valov. V to kategorijo spadajo na primer interferenca (angl. interference) in z njo povezana iridescenca (angl. iridescence), uklon svetlobe oz. širjenje valovanja v področje sence (angl. diffraction) in Dopplerjev pojav zaradi relativnega gibanja izvora svetlobe in opazovalca (angl. Doppler effect). Elektromagnetna optika dodatno lahko opiše še polarizacijo svetlobe in s tem povezane pojave (angl. polarization) ter razklon svetlobe oz. lom svetlobe v odvisnosti od frekvence (angl. dispersion). Kvantne pojave, kot na primer fluorescenco in fosforescenco, lahko opišemo s kvantno optiko.

Bolj kompleksni modeli svetlobe lahko opišejo širši spekter svetlobnih pojavov, ki pa so vse manj opazni. V računalniški grafiki se zato v veliki meri uporablja le geometrijska optika, za določene ostale pojave pa pogosto uporabljamo le posebne vizualne efekte.

3.3 Nastanek svetlobe

Obravnava svetlobe se začne pri njenem nastanku. Nastanek svetlobe je lahko posledica kemičnih, bioloških ali mehanskih pojavov, ki jih opišemo s skupno besedo *luminiscenca*. Glede na izvor jo ločimo na kemoluminiscenco

(vključno z bioluminiscenco), fotoluminiscenco (vključno s fluorescenco in fosforescenco), elektroluminiscenco, mehanoluminiscenco, radioluminiscenco in termoluminiscenco. V vsakdanjem življenju se najpogosteje srečujemo s *termoluminiscenco*, pri kateri svetloba nastane kot posledica termalnega gibanja delcev snovi. Gibanje delcev in njihove medsebojne interakcije povzročajo oscilacije električnega in magnetnega polja - elektromagnetno valovanje. Del spektra tega valovanja zaznavamo kot vidno svetlobo. Termoluminiscenca je na primer vzrok za izvor sončne svetlobe, na enak princip pa delujejo tudi žarnice z žarilno nitjo.

Vsako telo oddaja termalno sevanje, katerega izsevnost je sorazmerna s četrto potenco temperature. To lastnost opisuje Stefan-Boltzmannov zakon, ki pa navaja le skupno energijo vseh frekvenc, ne pa tudi porazdelitve energije glede na frekvenco. Planckov zakon, generalizacija Stefan-Boltzmannovega zakona, opisuje spektralno sevalnost črnega telesa. Integracija Planckovega zakona po vseh frekvencah nam kot rezultat da Stefan-Boltzmannov zakon. Po Planckovem zakonu je spektralna sevalnost unimodalna funkcija, katere maksimum opisuje Wienov zakon. Ta maksimum se z višanjem temperature premika prek rdeče, oranžne in rumene do modre barve. Wienov zakon lahko s pridom uporabimo za merjenje temperature zvezd prek njihove barve, v vsakdanjem življenju pa ga srečujemo na embalažah žarnic, kjer je njihova barva pogosto predstavljena s temperaturo.

Poleg termoluminiscence sta v vsakdanjem svetu pogosti še elektroluminiscenca, ki jo srečamo pri LED diodah, in fotoluminiscenca, pri kateri gre za sevanje svetlobe določenih valovnih dolžin zaradi prehodov elektronov v nižja energijska stanja. Ostalih oblik luminiscence ne bomo posebej obravnavali, ampak jih navajamo le za referenco.

3.4 Barve

Fizikalno gledano je vsaka fizična barva kombinacija *spektralnih barv*, ki so posledica *monokromatske svetlobe* oz. svetlobe ene same frekvence. Poljubna

fizična barva je vedno kombinacija spektralnih barv. Ljudje ne zaznamo vseh fizičnih barv, temveč le določeno podmnožico. Človeško oko vsebuje fotoreceptorske celice, imenovane *čepki*, ki se odzivajo na frekvenčni spekter vidne svetlobe. Obstajajo tri vrste čepkov z različnimi frekvenčnimi odzivi, zato se je močno uveljavila t. i. *tribarvna teorija* oz. *trikromatizem*, ki pravi, da lahko poljubno vidno fizično barvo predstavimo s kombinacijo treh primarnih barv. Obstoj treh vrst čepkov je predvidel že Thomas Young v začetku 19. stoletja, Hermann von Helmholtz pa je njegovo delo nadaljeval, zato se tribarvna teorija imenuje tudi Young-Helmholtzova teorija.

Ker je frekvenčni spekter neskončen, lahko prostor fizičnih barv obravnavamo kot neskončno-dimenzionalen vektorski prostor oz. Hilbertov prostor nad spektralnimi barvami. Vidne barve so glede na tribarvno teorijo tridimenzionalni evklidski prostor, v katerem je vsaka barva predstavljena s trojico števil, ki opisujejo stimulacijo treh vrst čepkov.

Vsako fizično barvo lahko predstavimo kot funkcijo $C(\lambda)$ nad svetlobnim spektrom Λ. Barva, ki jo zazna človeško oko, je integral zmnožka fizične barve in frekvenčnega odziva čepkov. Frekvenčne odzive treh vrst čepkov tipično označujemo z $l(\lambda)$, $m(\lambda)$ in $s(\lambda)$, kjer simboli teh funkcij namigujejo na dolge, srednje in kratke elektromagnetne valove. Tako dobimo trojček števil (L, M, S) t. i. *barvnega prostora LMS*, ki predstavlja zaznano barvo, in ga izračunamo po sledeči formuli:

$$L = \int_{\Lambda} C(\lambda) l(\lambda)\, \mathrm{d}\lambda, \tag{3.1}$$

$$M = \int_{\Lambda} C(\lambda) m(\lambda)\, \mathrm{d}\lambda, \tag{3.2}$$

$$S = \int_{\Lambda} C(\lambda) s(\lambda)\, \mathrm{d}\lambda. \tag{3.3}$$

Iz zgornjega opisa je očitno predvsem dvoje:

1. Odziv čepkov je *linearen*, kar pomeni, da je odziv večkratnika fizične barve enak večkratniku odziva te fizične barve, odziv seštevka dveh fizičnih barv pa je seštevek odzivov teh dveh fizičnih barv.

2. Različne fizične barve lahko povzročijo enake zaznane barve, kar imenujemo *metamerizem*.

Glede na tribarvno teorijo je Mednarodna komisija za razsvetljavo (CIE) leta 1931 predlagala barvni prostor CIE 1931 RGB, ki s tremi vnaprej določenimi monokromatskimi barvami opiše celoten barvni prostor, ki ga ljudje zaznamo. Iz empiričnih podatkov so določili bazne funkcije $\bar{r}$, $\bar{g}$ in $\bar{b}$ tako, da je veljala preslikava fizične barve v barvni prostor CIE 1931 RGB podobno kot v zgornjih enačbah. Dobljene bazne funkcije niso bile povsod pozitivne, kar je predstavljalo težavo pri reprodukciji barv in računanju, zato so sintetično določili še linearno preslikavo v barvni prostor CIE 1931 XYZ z baznimi funkcijami $\bar{x}$, $\bar{y}$ in $\bar{z}$ tako, da so bile le-te povsod pozitivne. Poleg tega so poskrbeli za to, da je bila funkcija $\bar{y}$ enaka *spektralni občutljivosti očesa*, konveksna kombinacija baznih funkcij pa je vsebovala vso vidno svetlobo.

Dandanes večina naprav uporablja barvni prostor sRGB, ki se je uveljavil precej kasneje. Gre za nelinearno transformacijo barvnega prostora CIE 1931 XYZ, ki zagotavlja večjo barvno enakomernost in zadovoljivo ločljivost pri uporabi 8-bitne kvantizacije. Barvni prostor sRGB je v nasprotju s CIE 1931 RGB in XYZ nelinearen, zato seštevanje barv nima posebnega fizikalnega pomena. Za fizikalno pravilno upodabljanje je bistvena uporaba linearnega barvnega prostora (na primer CIE 1931 XYZ) v internih izračunih, za pravilen prikaz na zaslonu pa jih moramo predtem preslikati v barvni prostor sRGB. To operacijo danes omogoča večina grafične strojne opreme.

3.5 Radiometrija in fotometrija

Bistvo fotorealističnega upodabljanja leži v fizikalno pravilni obravnavi svetlobe in pravilni predstavitvi na zaslonu. Fizikalno in psihofiziološko osnovo za metode, predstavljene v tem delu, najdemo v radiometriji in fotometriji [81, 82]. Medtem ko je radiometrija povsem fizikalna veda, fotometrija obravnava tudi človeško zaznavo. Fotometrija je zaradi neposredne povezave s človeškim zaznavanjem precej bolj pogosto obravnavana in je za naše delo

Simbol	Enota	Veličina
Q	J	sevalna energija
Φ	W	sevalni tok
φ	$\mathrm{W} \cdot \mathrm{m}^{-2}$	gostota sevalnega toka
I	$\mathrm{W} \cdot \mathrm{sr}^{-1}$	sevnost
L	$\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}$	sevalnost
M	$\mathrm{W} \cdot \mathrm{m}^{-2}$	izsevnost
E	$\mathrm{W} \cdot \mathrm{m}^{-2}$	obsevanost
Q	$\mathrm{lm} \cdot \mathrm{s}$	množina svetlobe
Φ	lm	svetlobni tok
φ	$\mathrm{lm} \cdot \mathrm{m}^{-2}$	gostota svetlobnega toka
I	cd	svetilnost
L	$\mathrm{cd} \cdot \mathrm{m}^{-2}$	svetlost
M	lx	svetlobna izsevnost
E	lx	osvetljenost

Tabela 3.1: Radiometrične in fotometrične veličine ter njihove enote.

tudi bolj neposredno uporabna, medtem ko je radiometrija novejša veda z močnejšo fizikalno osnovo. Tabela 3.1 navaja najbolj pogoste radiometrične in fotometrične veličine in enote. V računalniški grafiki s pridom uporabljamo veličine obeh vrst, saj moramo fizikalno pravilno pridobljeno sliko tudi predstaviti uporabniku prek računalniškega zaslona.

Vse uporabljene veličine so odvisne od frekvence svetlobe, česar v tem delu zaradi lažje obravnave eksplicitno ne označujemo. Če želimo poudariti odvisnost določene veličine od frekvence, uporabimo pridevnik *spektralen*, na primer *spektralna sevalnost*. Vse navedene radiometrične veličine imajo svoje fotometrične ekvivalente, ki jih dobimo tako, da radiometrične veličine utežimo s *spektralno občutljivostjo očesa*. Kot primer vzemimo sevalni in svetlobni tok. Spektralni sevalni tok je energija, ki jo vir v enoti časa odda v prostor. Celoten sevalni tok je integral spektralnega sevalnega toka po vseh frekvencah. Spektralni sevalni tok, utežen s spektralno občutljivostjo očesa,

Radiometrična veličina	Fotometrična veličina
sevalna energija	množina svetlobe
sevalni tok	svetlobni tok
gostota sevalnega toka	gostota svetlobnega toka
sevnost	svetilnost
sevalnost	svetlost
(energijska) izsevnost	(svetlobna) izsevnost
obsevanost	osvetljenost

Tabela 3.2: Ekvivalenti med radiometričnimi in fotometričnimi veličinami.

je spektralni svetlobni tok, integral tega pa je celoten svetlobni tok. Tako povezane radiometrične in fotometrične veličine smo navedli v tabeli 3.2.

Iz zgodovinskih razlogov se v fotometriji uporablja širši spekter enot kot v radiometriji. Radiometrične enote temeljijo na *moči*, medtem ko fotometrične temeljijo na *svetilnosti*.

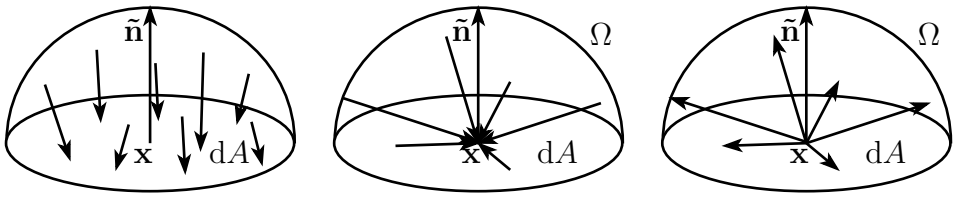
Za potrebe računalniške grafike je najbolj pomembna *svetlost*, saj neposredno odraža človeško zaznavo svetlosti. Svetlost je fotometrični ekvivalent *sevalnosti*, na kateri temelji večina algoritmov za upodabljanje s fizikalno osnovo, zato ta veličina tudi v pričujočem delu igra ključno vlogo.

V sledečih odstavkih so na kratko opisane radiometrične in fotometrične veličine ter njihove medsebojne povezave.

3.5.1 Sevalna energija in sevalni tok

Vsak vir svetlobe oddaja *sevalno energijo* (angl. radiant energy) Q. Količino energije, ki jo vir odda v enoti časa, imenujemo *sevalna moč* (angl. radiant power) ali *sevalni tok* (angl. radiant flux) in jo označimo s Φ.

$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t}. \tag{3.4}$$



Slika 3.1: Sevalni tok (levo), obsevanost (sredina) in izsevnost (desno).

3.5.2 Gostota sevalnega toka, obsevanost in izsevnost

Z dano ploskvijo v prostoru lahko definiramo tudi *gostoto sevalnega toka* (angl. radiant flux density) φ, ki je enaka sevalnemu toku skozi enoto površine.

$$\varphi = \frac{\Phi}{A}. \tag{3.5}$$

Če ploskev v določeni točki $\mathbf{x}$ orientiramo z normalo $\tilde{\mathbf{n}}$, lahko definiramo tudi *obsevanost* (angl. irradiance) E in *izsevnost* (angl. radiant exitance, radiosity) M, če sevalni tok teče v površino ali iz nje:

$$E(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A(\mathbf{x})} \text{ in} \tag{3.6}$$

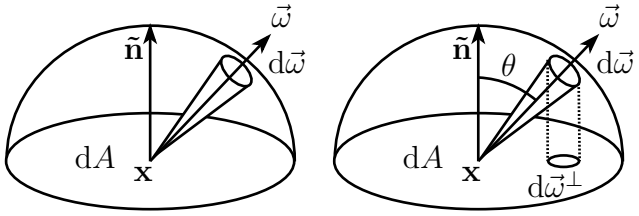
$$M(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A(\mathbf{x})}. \tag{3.7}$$

Zgornje enačbe držijo le, če sevalni tok pada pravokotno na dano ploskev. V splošnem primeru moramo upoštevati še smer toka in količine pomnožiti s kosinusom vpadnega kota, kar opišemo s *sevalnostjo*.

3.5.3 Sevnost in svetilnost

Ker viri sevanja ne sevajo enako močno v vse smeri, lahko sevalni tok v določeni smeri opišemo s *sevnostjo* (angl. radiant intensity) I. Veličina je odvisna od prostorskega kota $\vec{\omega}$:

$$I(\vec{\omega}) = \frac{\mathrm{d}\Phi}{\mathrm{d}\vec{\omega}}. \tag{3.8}$$



Slika 3.2: Sevnost (levo) in sevalnost (desno).

Sevnost posebej omenjamo zato, ker predstavlja osnovo za fotometrijo, kjer jo utežimo s spektralno občutljivostjo očesa in imenujemo *svetilnost* (angl. luminous intensity). Enota svetilnosti je *kandela* (cd), ki je tudi ena izmed sedmih osnovnih enot sistema SI, in je definirana kot svetilnost vira, ki v dani smeri oddaja monokromatsko sevanje frekvence $540 \cdot 10^{12}\,\mathrm{Hz}$ in seva z jakostjo $\frac{1}{683}\,\mathrm{W} \cdot \mathrm{sr}^{-1}$.

3.5.4 Sevalnost

Sedaj lahko združimo obsevanost oz. izsevnost s sevnostjo v določeni smeri in tako definiramo *sevalnost* (angl. radiance) L, ki opisuje količino sevalnega toka v določeni smeri in pod določenim kotom glede na normalo površine:

$$L(\mathbf{x}, \vec{\omega}) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \vec{\omega})}{\mathrm{d}\vec{\omega}\,\mathrm{d}A^{\perp}(\mathbf{x})} \tag{3.9}$$

ali ekvivalentno

$$L(\mathbf{x}, \vec{\omega}) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \vec{\omega})}{\mathrm{d}\vec{\omega}^{\perp}\,\mathrm{d}A(\mathbf{x})}. \tag{3.10}$$

V zgornji enačbi smo z $A^{\perp}$ označili pravokotno projekcijo diferencialne površine na smer svetlobe. Z drugimi besedami, upoštevali smo le delež svetlobnega toka, ki pada v pravokotni smeri $\vec{\omega}^{\perp}$ na površino. Ta delež je enak kosinusu vpadnega kota $\cos\phi$, ki ga lahko izrazimo s skalarnim produktom med normalo površine in smerjo sevalnega toka iz sledečih relacij:

$$\mathrm{d}A^{\perp} = |\tilde{\mathbf{n}} \cdot \vec{\omega}|\,\mathrm{d}A \tag{3.11}$$

in

$$\mathrm{d}\vec{\omega}^{\perp} = |\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}\vec{\omega}. \tag{3.12}$$

Tako lahko sevalnost ekvivalentno zapišemo kot

$$L(\mathbf{x}, \vec{\omega}) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \vec{\omega})}{|\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}\vec{\omega} \, \mathrm{d}A(\mathbf{x})}. \tag{3.13}$$

Poglavje 4

Formalizacija

Za potrebe našega dela bomo obnašanje in lastnosti svetlobe formalizirali s pomočjo sevalnosti, s katero bomo izrazili ostale veličine. Razlog za tem je neposredna povezava med sevalnostjo in svetlostjo neke točke iz dane smeri opazovanja. S tem poglavjem bomo začeli zanemarjati določene lastnosti svetlobe in svetlobne pojave z namenom poenostavitve do oblike, primerne za računanje.

Sevalnost v tem delu uporabljamo kot najbolj osnovno veličino, ki jo lahko uporabimo za izražanje prej opisanih veličin, če dane relacije obrnemo. Tako lahko definiramo sevalni tok kot

$$\Phi = \int_A \int_\Omega L(\mathbf{x}, \vec{\omega}) |\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}\vec{\omega} \, \mathrm{d}A(\mathbf{x}) \tag{4.1}$$

in ostale veličine izpeljemo iz te relacije. Tako dobimo

$$I(\vec{\omega}) = \int_A L(\mathbf{x}, \vec{\omega}) |\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}A(\mathbf{x}), \tag{4.2}$$

$$E(\mathbf{x}) = \int_\Omega L(\mathbf{x}, \vec{\omega}) |\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}\vec{\omega} \text{ in} \tag{4.3}$$

$$M(\mathbf{x}) = \int_\Omega L(\mathbf{x}, \vec{\omega}) |\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}\vec{\omega}. \tag{4.4}$$

V tem delu bomo dodatno upoštevali še trenutek obravnave sevalnosti v določeni točki in smeri, saj lahko tako enostavneje opišemo interakcijo svetlobe s površino. Tik preden svetloba iz določene smeri pade na določeno

točko, je sevalnost v tej točki $L(\mathbf{x} \leftarrow \vec{\omega})$ in jo imenujemo *vstopna sevalnost*, po interakciji s površino pa $L(\mathbf{x} \rightarrow \vec{\omega})$ in jo imenujemo *izstopna sevalnost*. S takim razločevanjem lahko v enačbe enostavno vpeljemo različne svetlobne pojave, kot so na primer emisija, absorpcija, sipanje ipd. Obsevanost in izsevnost sta tako bolj natančno opisani z

$$E(\mathbf{x}) = \int_{\Omega} L(\mathbf{x} \leftarrow \vec{\omega}) |\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}\vec{\omega} \text{ in} \tag{4.5}$$

$$M(\mathbf{x}) = \int_{\Omega} L(\mathbf{x} \rightarrow \vec{\omega}) |\tilde{\mathbf{n}} \cdot \vec{\omega}| \, \mathrm{d}\vec{\omega}. \tag{4.6}$$

Prav zaradi interakcije svetlobe s površino v splošnem te količine niso enake:

$$L(\mathbf{x} \rightarrow \vec{\omega}) \neq L(\mathbf{x} \leftarrow \vec{\omega}), \tag{4.7}$$

toda v vakuumu, kjer interakcij ni, velja, da se energija ohranja:

$$L(\mathbf{x} \rightarrow \vec{\omega}) = L(\mathbf{x} \leftarrow -\vec{\omega}). \tag{4.8}$$

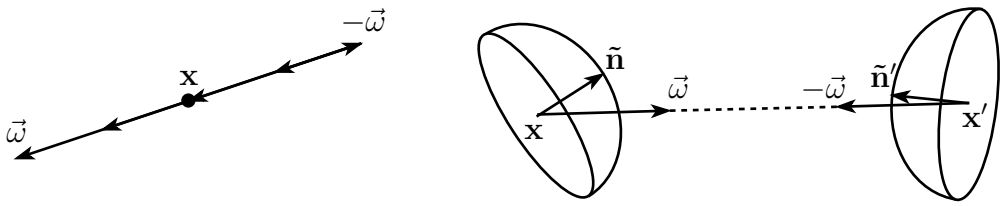
Zgornjo enačbo lahko posplošimo na dve medsebojno vidni točki v vakuumu $\mathbf{x}$ in $\mathbf{x}'$, povezani s smerjo $\vec{\omega}$. V tem primeru se energija ohranja, tako da je izsevnost v eni točki v smeri $\vec{\omega}$ enaka obsevanosti druge točke iz nasprotne smeri in obratno:

$$L(\mathbf{x} \rightarrow \vec{\omega}) = L(\mathbf{x}' \leftarrow -\vec{\omega}) \text{ in} \tag{4.9}$$

$$L(\mathbf{x}' \rightarrow \vec{\omega}) = L(\mathbf{x} \leftarrow -\vec{\omega}). \tag{4.10}$$

4.1 Obnašanje svetlobe ob stiku s površino

Svetloba se v našem poenostavljenem modelu ob stiku s površino odbije ali absorbira. Delež odboja ali absorpcije za dani par vpadne in odbojne smeri lahko opišemo s 6-dimenzionalno funkcijo $f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}')$, ki jo imenujemo *dvosmerna funkcija porazdelitve odbojnosti* (angl. bidirectional reflectance distribution function, BRDF). Funkcija f_r za dano točko $\mathbf{x}$ na površini z normalo



Slika 4.1: Ohranitev sevalne energije: v točki v vakuumu je vstopna sevalnost je enaka izstopni sevalnosti (levo), kar lahko razširimo na žarek, ki neovirano potuje skozi vakuum (desno).

$\tilde{\mathbf{n}}$ ter za vpadno smer $\vec{\omega}'$ in odbojno smer $\vec{\omega}$ vrne razmerje med sevalnostjo v smeri $\vec{\omega}$ in obsevanostjo iz smeri $\vec{\omega}'$:

$$f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \frac{\mathrm{d}L(\mathbf{x} \to \vec{\omega})}{\mathrm{d}E(\mathbf{x} \leftarrow \vec{\omega}')} \tag{4.11}$$

$$= \frac{\mathrm{d}L(\mathbf{x} \to \vec{\omega})}{L(\mathbf{x} \leftarrow \vec{\omega}')|\tilde{\mathbf{n}} \cdot \vec{\omega}'| \,\mathrm{d}\vec{\omega}'}. \tag{4.12}$$

Funkcija f_r mora zadoščati določenim pogojem, da je fizikalno mogoča:

1. *Nenegativnost.* Obsevanost in izsevnost ne moreta biti negativni.

$$f_r \geq 0. \tag{4.13}$$

2. *Ohranitev energije.* Izsevnost ne more biti večja od obsevanosti.

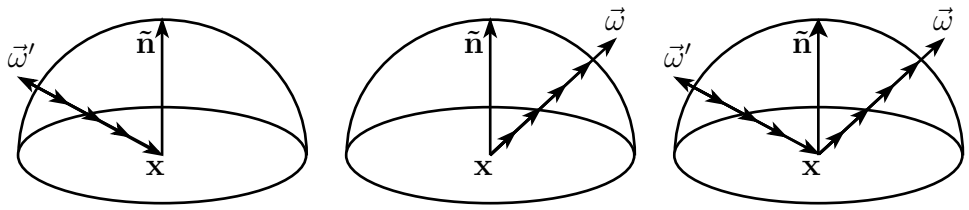
$$\forall \vec{\omega} : \int_\Omega f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}')|\tilde{\mathbf{n}} \cdot \vec{\omega}'| \,\mathrm{d}\vec{\omega}' \leq 1. \tag{4.14}$$

3. *Recipročnost.* Helmholtzov princip recipročnosti pravi, da se enaka količina svetlobe odbije, če njeno smer obrnemo.

$$f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}') = f_r(\mathbf{x}, \vec{\omega}', \vec{\omega}). \tag{4.15}$$

Če enačbo 4.12 pomnožimo z imenovalcem in integriramo po vseh smereh v polsferi Ω okrog normale $\tilde{\mathbf{n}}$ nad točko $\mathbf{x}$, dobimo sevalnost v točki $\mathbf{x}$ v smeri $\vec{\omega}$:

$$L(\mathbf{x} \to \vec{\omega}) = \int_\Omega f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}')L(\mathbf{x} \leftarrow \vec{\omega}')|\tilde{\mathbf{n}} \cdot \vec{\omega}'| \,\mathrm{d}\vec{\omega}'. \tag{4.16}$$



Slika 4.2: Vstopna sevalnost (levo) izstopna sevalnost (sredina) in dvosmerna funkcija porazdelitve odbojnosti (desno).

Zgornja enačba predstavlja osnovo za fizikalno pravilno upodabljanje. Pogosto se uporablja še posplošena inačica, v kateri celotno sevalnost L razbijemo na emisijski del L_e in odbojni del L_r:

$$L(\mathbf{x} \to \vec{\omega}) = L_e(\mathbf{x} \to \vec{\omega}) + L_r(\mathbf{x} \to \vec{\omega}). \tag{4.17}$$

Odboj svetlobe opišemo z zgornjo enačbo in tako dobimo *enačbo upodabljanja*, ki jo je prvi opisal James Kajiya leta 1986:

$$L(\mathbf{x} \to \vec{\omega}) = L_e(\mathbf{x} \to \vec{\omega}) + \int_\Omega f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}') L(\mathbf{x} \leftarrow \vec{\omega}') |\tilde{\mathbf{n}} \cdot \vec{\omega}'| \, \mathrm{d}\vec{\omega}'. \tag{4.18}$$

Pogosto nas ne zanimajo le trdne površine, ampak tudi prosojne. V takih primerih je enačba upodabljanja v zgoraj napisani obliki nezadovoljiva. Odbojni del moramo razširiti in upoštevati tudi nasprotno stran površine, kjer namesto dvosmerne funkcije porazdelitve odbojnosti uporabimo *dvosmerno funkcijo porazdelitve prepustnosti* (angl. bidirectional transmittance distribution function, BTDF) f_t. Funkcija f_t deluje povsem enako kot f_r, le na nasprotni strani površine. Razmerje moči odbojnosti in prepustnosti opisujejo Fresnelove enačbe. Funkciji f_r in f_t lahko skupaj s Fresnelovimi enačbami združimo v skupno obravnavo z *dvosmerno funkcijo porazdelitve sipanja* (angl. bidirectional scattering distribution function, BSDF) f_s. Enačba upodabljanja je v tem primeru podobna, le da integriramo po celotni sferi in upoštevamo absolutno vrednost kosinusnega faktorja:

$$L(\mathbf{x} \to \vec{\omega}) = L_e(\mathbf{x} \to \vec{\omega}) + \int_{S^2} f_s(\mathbf{x}, \vec{\omega}, \vec{\omega}') L(\mathbf{x} \leftarrow \vec{\omega}') |\tilde{\mathbf{n}} \cdot \vec{\omega}'| \, \mathrm{d}\vec{\omega}'. \tag{4.19}$$

4.2 Obnašanje svetlobe v mediju

V prejšnjem poglavju smo predpostavili, da svetloba od ene do druge površine potuje po vakuumu. Za upodabljanje prosojnih materialov moramo to predpostavko odstraniti in upoštevati, da lahko do svetlobnih pojavov pride tudi med samim transportom [83]. Svetloba se tudi med prehodom skozi zrak lomi in odbija, saj ga sestavljajo mikroskopski delci, ki vplivajo na širjenje svetlobe. Predvsem na kratkih razdaljah ima to razmeroma neopazen vpliv, zato je aproksimacija z vakuumom v takih primerih povsem zadovoljiva. Mnoge površine in materiali pa niso povsem trdni, temveč prosojni. Kot tipične primere iz vsakdanjega življenja lahko vzamemo človeško kožo, listje dreves in mleko. Take materiale moramo pravilneje obravnavati kot medij, skozi katerega se širi svetloba. Medij aktivno sodeluje pri širjenju in vpliva na poti fotonov na podoben način kot površine, toda v nasprotju s slednjimi je ta s svetlobo v nenehni interakciji.

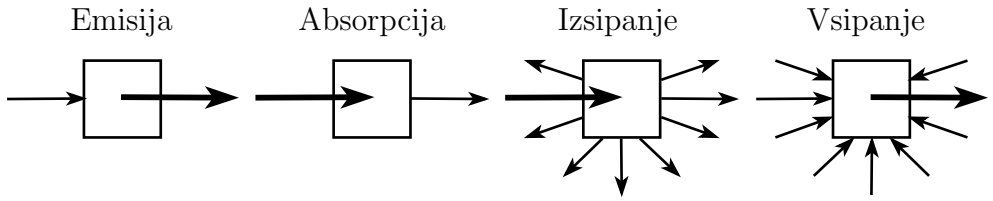
V tem delu predpostavljamo, da medij sestavljajo mikroskopski delci, ki absorbirajo in odbijajo fotone. Ti delci so dovolj majhni, da lahko za njihovo obravnavo uporabljamo statistične metode, trki med fotoni in delci pa statistično neodvisni. Vsak trk med fotonom in delcem medija lahko privede do absorpcije ali odboja, kjer je absorpcija vedno popolna, odboj pa vedno elastičen, tako da foton energije ne izgubi.

Spremembe sevalne energije v mediju povzročajo štirje različni procesi: *emisija* (angl. emission) E, *absorpcija* (angl. absorption) A, *vsipanje* (angl. in-scattering) I in *izsipanje* (angl. out-scattering) O, ki jih opisujejo naslednje lastnosti medija: *emisijska funkcija* L_e, *absorpcijski koeficient* σ_a, *koeficient sipanja* σ_s in *fazna funkcija* p. Shematično so predstavljeni na sliki 4.3.

Emisija in vsipanje povečujeta sevalno energijo, medtem ko jo absorpcija in izsipanje zmanjšujeta, zato lahko nastavimo sledečo *ravnovesno enačbo*:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = E(\mathbf{x} \to \vec{\omega}) + I(\mathbf{x} \to \vec{\omega}) - A(\mathbf{x} \to \vec{\omega}) - O(\mathbf{x} \to \vec{\omega}), \tag{4.20}$$

kjer je $\vec{\omega} \cdot \nabla$ smerni odvod v smeri $\vec{\omega}$.



Slika 4.3: Štirje glavni procesi pri interakciji svetlobe z medijem.

4.2.1 Emisija

Emisijo enostavno opišemo z emisijsko funkcijo $L_e(\mathbf{x} \to \vec{\omega})$, ki za vsako točko v prostoru vrne emisijsko izsevnost v dani smeri:

$$E(\mathbf{x} \to \vec{\omega}) = L_e(\mathbf{x} \to \vec{\omega}). \tag{4.21}$$

V zgolj emisivnem mediju se ravnovesna enačba glasi

$$(\vec{\omega} \cdot \nabla)L(\mathbf{x} \to \vec{\omega}) = E(\mathbf{x} \to \vec{\omega}), \tag{4.22}$$

katere rešitev vzdolž žarka $\mathbf{x} + s\vec{\omega}$ je

$$L((\mathbf{x} + s\vec{\omega}) \to \vec{\omega}) = L(\mathbf{x} \to \vec{\omega}) + \int_0^s L_e(\mathbf{x} \to \vec{\omega}). \tag{4.23}$$

4.2.2 Absorpcija

Ko svetlobni žarek trči v medij se absorbira delež σ_a fotonov na enoto dolžine, zato je absorpcijski del ravnovesne enačbe enak

$$A(\mathbf{x} \to \vec{\omega}) = \sigma_a(\mathbf{x})L(\mathbf{x} \to \vec{\omega}). \tag{4.24}$$

Hitrost absorpcije je sorazmerna s količino fotonov, ki vstopajo v obravnavano točko. Sorazmernostni koeficient, v tem primeru absorpcijski koeficient σ_a, si lahko predstavljamo tudi kot *gostoto materiala*. Če upoštevamo le absorpcijo, dobimo sledečo ravnovesno enačbo:

$$(\vec{\omega} \cdot \nabla)L(\mathbf{x} \to \vec{\omega}) = -\sigma_a(\mathbf{x})L(\mathbf{x} \to \vec{\omega}), \tag{4.25}$$

ki jo lahko integriramo vzdolž žarka $\mathbf{x} + s\vec{\omega}$, da dobimo *absorpcijsko enačbo* ali *Beer-Lambertov zakon*:

$$L((\mathbf{x} + s\vec{\omega}) \to \vec{\omega}) = L(\mathbf{x} \to \vec{\omega}) e^{-\int_0^s \sigma_a(\mathbf{x}+s\vec{\omega})\,\mathrm{d}s}. \tag{4.26}$$

Absorpcijo svetlobe v mediju so med prvimi proučevali Pierre Bouguer, Johann Heinrich Lambert in August Beer v 18. in 19. stoletju, katerih imena nosi zgoraj opisana absorpcijska enačba.

Integral absorptivnosti imenujemo *optična globina* in ga označimo s τ. Optično globino definirajmo med dvema točkama $\mathbf{x}$ in $\mathbf{x}'$, med katerima je vpet enostki vektor $\vec{\omega}_{\mathbf{x}\mathbf{x}'} = \frac{\mathbf{x}'-\mathbf{x}}{||\mathbf{x}'-\mathbf{x}||}$:

$$\tau(\mathbf{x}, \mathbf{x}') = \int_0^{||\vec{\omega}_{\mathbf{x}\mathbf{x}'}||} \sigma_a(\mathbf{x} + s\vec{\omega}_{\mathbf{x}\mathbf{x}'})\,\mathrm{d}s. \tag{4.27}$$

Za optično globino velja princip recipročnosti, zato je

$$\tau(\mathbf{x}, \mathbf{x}') = \tau(\mathbf{x}', \mathbf{x}) = \tau(\mathbf{x} \leftrightarrow \mathbf{x}'). \tag{4.28}$$

Faktor poleg začetne sevalnosti v absorpcijski enačbi imenujemo tudi *prepustnost* in jo označimo s T:

$$T(\mathbf{x} \leftrightarrow \mathbf{x}') = e^{-\tau(\mathbf{x}\leftrightarrow\mathbf{x}')}. \tag{4.29}$$

Tako lahko absorpcijsko enačbo krajše zapišemo kot:

$$L(\mathbf{x}' \to \vec{\omega}_{\mathbf{x}\mathbf{x}'}) = L(\mathbf{x} \to \vec{\omega}_{\mathbf{x}\mathbf{x}'}) T(\mathbf{x} \leftrightarrow \mathbf{x}'). \tag{4.30}$$

Prepustnost je multiplikativna, zato za poljubno točko $\mathbf{x}'$ na žarku med $\mathbf{x}$ in $\mathbf{x}''$ velja

$$T(\mathbf{x} \leftrightarrow \mathbf{x}'') = T(\mathbf{x} \leftrightarrow \mathbf{x}') T(\mathbf{x}' \leftrightarrow \mathbf{x}''). \tag{4.31}$$

4.2.3 Izsipanje

Sipanje smo razdelili na dva ločena dela, saj lahko za dano točko in smer v prostoru izsipanje obravnavamo na enak način kot absorpcijo, saj vsaka

sprememba poti fotona pomeni izgubo sevalne energije v dani smeri. Enačba za izsipanje je enaka enačbi za absorpcijo. Moč izsipanja je prav tako sorazmerna s količino fotonov, ki vstopajo v obravnavano točko, le da tokrat kot sorazmernostni koeficient uporabljamo koeficient sipanja σ_s:

$$O(\mathbf{x} \to \vec{\omega}) = \sigma_s(\mathbf{x}) L(\mathbf{x} \to \vec{\omega}). \tag{4.32}$$

Absorpcijo in izsipanje zaradi enake obravnave pogosto združimo v en faktor s t. i. *koeficientom izginjanja* (angl. extinction coefficient) σ_t:

$$\sigma_t(\mathbf{x}) = \sigma_a(\mathbf{x}) + \sigma_s(\mathbf{x}). \tag{4.33}$$

Za opisovanje lastnosti medija se pogosto uporablja tudi *albedo* ali *koeficient odbojnosti* α, ki ga definiramo kot razmerje med koeficientom sipanja in koeficientom izginjanja:

$$\alpha(\mathbf{x}) = \frac{\sigma_s(\mathbf{x})}{\sigma_t(\mathbf{x})}. \tag{4.34}$$

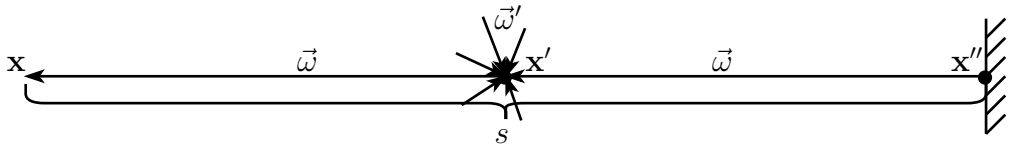
Vrednost $\alpha = 0$ opisuje medij, ki le absorbira svetlobo, medtem ko vrednost $\alpha = 1$ opisuje medij, ki svetlobe ne absorbira, temveč jo le sipa.

4.2.4 Vsipanje

Vsipanje v določeni točki in smeri v prostoru je seštevek vse sevalne energije fotonov iz vseh smeri, ki se odbijejo v dano smer. Podobno kot stik s površino kompaktno opišemo s funkcijo BSDF, lahko sipanje v mediju opišemo s *fazno funkcijo*. Fazna funkcija opisuje porazdelitev sipanja glede na lomni kot. V izračunu vsipanja jo moramo torej upoštevati pri integraciji prek vseh smeri:

$$I(\mathbf{x} \to \vec{\omega}) = \sigma_s(\mathbf{x}) \int_{S^2} p(\mathbf{x}, \vec{\omega}, \vec{\omega}') L(\mathbf{x} \leftarrow \vec{\omega}') \, \mathrm{d}\vec{\omega}'. \tag{4.35}$$

Fazne funkcije so bolj natančno opisane v poglavju 4.2.6.



Slika 4.4: Enačba sevanja združi prispevke izstopne sevalnosti ozadja z emisijo medija in sipanjem v smeri proti kameri.

4.2.5 Enačba sevanja

Ravnovesno enačbo lahko sedaj zapišemo z zgoraj izpeljanimi izrazi:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = L_e(\mathbf{x} \to \vec{\omega}) \tag{4.36}$$

$$- \sigma_t(\mathbf{x}) L(\mathbf{x} \to \vec{\omega}) \tag{4.37}$$

$$+ \sigma_s(\mathbf{x}) \int_{S^2} p(\mathbf{x}, \vec{\omega}, \vec{\omega}') L(\mathbf{x} \leftarrow \vec{\omega}') \, \mathrm{d}\vec{\omega}', \tag{4.38}$$

ki jo lahko integriramo, da dobimo *enačbo sevanja* (angl. radiative transfer equation, RTE):

$$L(\mathbf{x} \to \vec{\omega}) = T(\mathbf{x} \leftrightarrow \mathbf{x}'') L(\mathbf{x}'' \to \vec{\omega}) \tag{4.39}$$

$$+ \int_0^s T(\mathbf{x} \leftrightarrow \mathbf{x}') L_e(\mathbf{x} \to \vec{\omega}) \, \mathrm{d}s \tag{4.40}$$

$$+ \int_0^s T(\mathbf{x} \leftrightarrow \mathbf{x}') \sigma_s(\mathbf{x}') \int_{S^2} p(\mathbf{x}', \vec{\omega}, \vec{\omega}') L(\mathbf{x}' \leftarrow \vec{\omega}') \, \mathrm{d}\vec{\omega}' \, \mathrm{d}s, \tag{4.41}$$

kjer je $\mathbf{x}$ točka, v kateri računamo sevalnost, $\mathbf{x}''$ začetna točka na žarku in $\mathbf{x}'$ vmesna točka na žarku med $\mathbf{x}$ in $\mathbf{x}''$, v kateri pride do spremembe sevalnosti.

4.2.6 Fazne funkcije

Fazna funkcija $p(\mathbf{x}, \vec{\omega}, \vec{\omega}')$ nam pove porazdelitev sipanja v določeni točki glede na vpadno in lomno smer. V tem delu zaradi konsistentne notacije uporabljamo enake parametre tako za funkcijo BRDF kot za fazno funkcijo. Obravnavali bomo le *izotropne* fazne funkcije, ki so rotacijsko invariantne, torej so odvisne le od lomnega kota θ. Anizotropne funkcije se imenujejo tudi *dvosmerne funkcije porazdelitve faze* (angl. bidirectional phase distribution

function, BPDF). V literaturi pogosto zasledimo tudi notacijo s kosinusom lomnega kota, $p(\cos(\theta))$, saj se ta tipično pojavlja v enačbah.

Tudi fazna funkcija mora zadoščati določenim pogojem, da je lahko fizikalno mogoča:

1. *Nenegativnost.* Gre za porazdelitveno funkcijo, ki ne sme biti negativna.

$$p \geq 0. \tag{4.42}$$

2. *Normaliziranost.* Gre za porazdelitveno funkcijo, katere integral po celotni domeni mora biti enak 1.

$$\forall \vec{\omega} : \int_{S^2} p(\mathbf{x}, \vec{\omega}, \vec{\omega}') \, \mathrm{d}\vec{\omega}' = 1, \tag{4.43}$$

   ali ekvivalentno za izotropne funkcije,

$$\int_0^{2\pi} \int_0^{\pi} p(\cos\theta) \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi = 1, \tag{4.44}$$

$$2\pi \int_{-1}^{1} p(\mu) \, \mathrm{d}\mu = 1. \tag{4.45}$$

3. *Recipročnost.* Helmholtzov princip recipročnosti mora veljati tudi za fazne funkcije.

$$p(\mathbf{x}, \vec{\omega}, \vec{\omega}') = p(\mathbf{x}, \vec{\omega}', \vec{\omega}). \tag{4.46}$$

   Izotropne fazne funkcije so vedno recipročne, saj velja $\cos(\theta) = \cos(-\theta)$.

V naslednjih poglavjih bomo obravnavali nekaj faznih funkcij, ki jih najpogosteje zasledimo v literaturi in praksi.

Izotropna fazna funkcija

Najenostavnejša fazna funkcija je *izotropna fazna funkcija*, ki enakomerno sipa v vse smeri:

$$p_I(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \frac{1}{4\pi}. \tag{4.47}$$

Ker je konstantno pozitivna, zadošča pogojem nenegativnosti in recipročnosti. Lahko se tudi prepričamo, da je normalizirana:

$$\int_{S^2} p_I(\mathbf{x}, \vec{\omega}, \vec{\omega}')\,\mathrm{d}\vec{\omega}' = \int_0^{2\pi}\int_0^{\pi} \frac{1}{4\pi}\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi \tag{4.48}$$

$$= \frac{1}{4\pi}\int_0^{2\pi}\Big[-\cos\theta\Big]_0^{\pi}\mathrm{d}\phi \tag{4.49}$$

$$= \frac{1}{4\pi}\int_0^{2\pi} 2\,\mathrm{d}\phi \tag{4.50}$$

$$= 1. \tag{4.51}$$

Povprečen lomni kot $\bar{\theta}$ lahko izračunamo na sledeč način:

$$\bar{\theta} = E(p) = \int_{S^2} (\vec{\omega}\cdot\vec{\omega}')p(\mathbf{x}, \vec{\omega}, \vec{\omega}')\,\mathrm{d}\vec{\omega}' \tag{4.52}$$

$$= \int_0^{2\pi}\int_0^{\pi} \theta p(\cos\theta)\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi. \tag{4.53}$$

$$\bar{\theta}_i = E(p_I) = \frac{1}{4\pi}\int_0^{2\pi}\Big[-\theta\cos\theta + \sin\theta\Big]_0^{\pi}\mathrm{d}\phi \tag{4.54}$$

$$= \frac{1}{4\pi}\int_0^{2\pi}\pi\,\mathrm{d}\phi \tag{4.55}$$

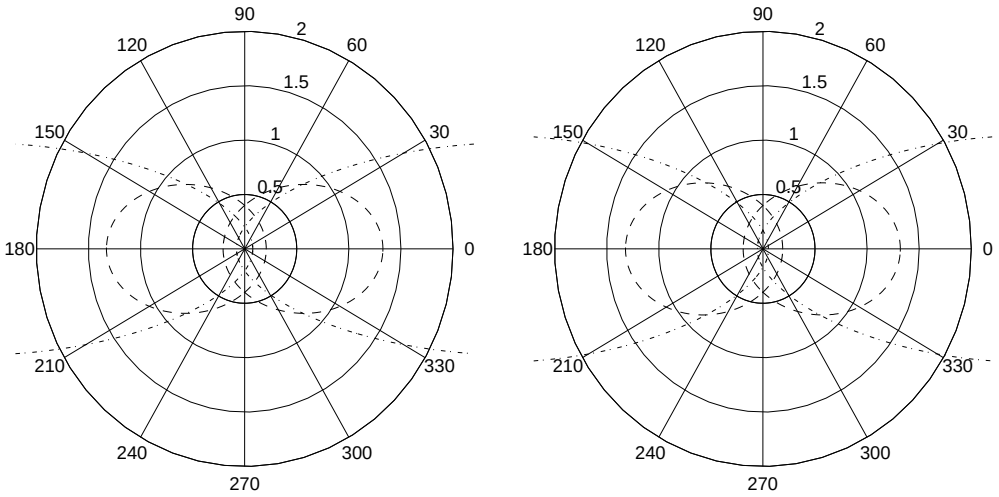
$$= \frac{\pi}{2}. \tag{4.56}$$

V zgornjih enačbah je E pričakovana vrednost.

Pretvorba med funkcijo BSDF in fazno funkcijo

Fazna funkcija je podobna funkciji BSDF in v enačbi sevanja igra enako vlogo kot funkcija BSDF v enačbi upodabljanja, le da operira neposredno s sevnostjo v dani smeri in ne na sevalnosti. S tem se enačba poenostavi, saj lahko iz nje odstranimo kosinusni faktor. Razlog za tem je, da pri sipanju v mediju nimamo površine in posledično normale. Če določimo še normalo, lahko poljubno funkcijo BSDF pretvorimo v fazno funkcijo tako, da jo pomnožimo s kosinusnim faktorjem:

$$p_{BSDF}(\mathbf{x}, \vec{\omega}, \vec{\omega}') = f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}')|\tilde{\mathbf{n}}\cdot\vec{\omega}'|. \tag{4.57}$$



Slika 4.5: Henyey-Greensteinova (levo) in Schlickova (desno) fazna funkcija. Parameter g zavzema vrednosti -0.6, -0.3, 0, 0.3 in 0.6, pri čemer negativne vrednosti pomenijo sipanje nazaj, pozitivne vrednosti sipanje naprej, vrednost $g = 0$ pa pomeni izotropno sipanje. Vrednosti parametra k so izračunane po aproksimacijski formuli.

Henyey-Greensteinova fazna funkcija

Louis George Henyey in Jesse Leonard Greenstein sta leta 1941 [41] predlagala sledečo funkcijo, s katero sta opisala sipanje svetlobe v vesolju zaradi medzvezdnega prahu:

$$p_{HG}(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g(\vec{\omega} \cdot \vec{\omega}'))^{\frac{3}{2}}}. \tag{4.58}$$

Z variiranjem parametra $-1 < g < 1$ lahko opišemo medije, ki sipajo naprej (angl. forward scattering) z $g > 0$, nazaj (angl. backward scattering) z $g < 0$ ali izotropno z $g = 0$. Parameter g ima tudi geometrijski pomen: predstavlja povprečni kosinus lomnega kota.

Schlickova fazna funkcija

Henyey-Greensteinova funkcija je za potrebe računalniške grafike pogosto preveč računsko kompleksna zaradi racionalnega eksponenta, zato se v te

namene običajno uporablja Schlickova aproksimacija [84], ki vsebuje le dve kvadratni potenci:

$$p_S(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \frac{1}{4\pi} \frac{1 - k^2}{(1 + k(\vec{\omega} \cdot \vec{\omega}'))^2}. \tag{4.59}$$

Parameter k lahko aproksimiramo po formuli

$$k \approx 1.55g - 0.55g^3. \tag{4.60}$$

Rayleighova fazna funkcija

Rayleighova fazna funkcija opisuje t. i. *Rayleighovo sipanje* svetlobe v plinastem mediju, kjer je velikost molekul bistveno manjša od valovne dolžine svetlobe. Sipanje naprej in nazaj je nekoliko bolj izrazito kot sipanje vstran, kar lahko opišemo s sledečo funkcijo:

$$p_R(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \frac{3}{16\pi}(1 + (\vec{\omega} \cdot \vec{\omega})^2). \tag{4.61}$$

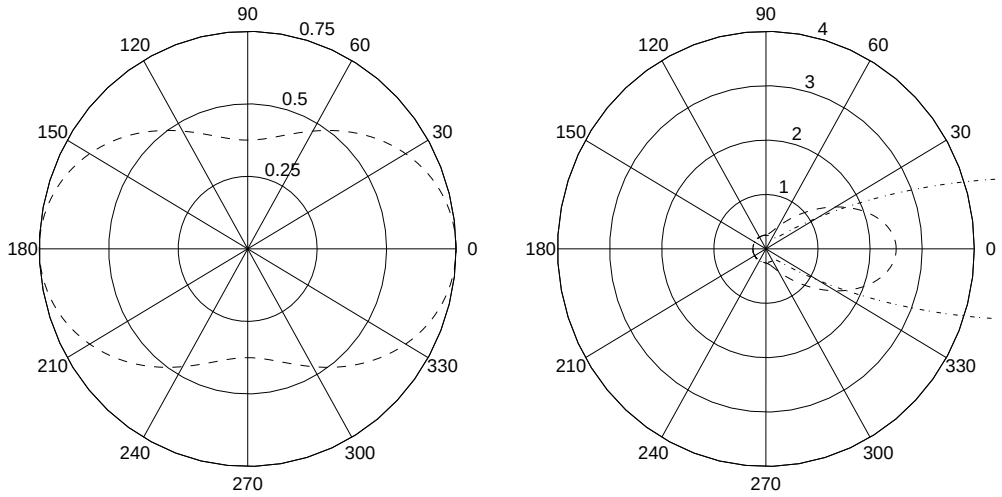
Zgornja funkcija je le aproksimacija kompleksnejše funkcije, ki je močno odvisna od valovne dolžine svetlobe. Tako je stransko sipanje pri svetlobi s kratko valovno dolžino precej močnejše kot pri svetlobi z dolgo valovno dolžino, kar je glavni razlog za modro barvo neba in rdečkasto barvo sončnega zahoda.

Vzorčenje Rayleighove fazne funkcije je računsko preveč zahtevno [85], zato jo pogosto opišemo s kombinacijo dveh Schlickovih faznih funkcij, katerih vzorčenje je precej bolj enostavno.

Mievi fazni funkciji

Zgoraj opisane fazne funkcije so le aproksimacije rešitev, ki jih dobimo z uporabo splošnejše Mieve teorije. Teorija, ki jo je razvil Gustav Mie v začetku 20. stoletja, je izpeljana iz Maxwellovih enačb in opisuje sipanje svetlobe poljubne frekvence v mediju, ki ga sestavljajo sfere določene velikosti.

Obstajata dve fazni funkciji, dobljeni z Mievo teorijo, ki opisujeta sipanje svetlobe v megli - t. i. meglena (angl. hazy) in mračna (angl. murky) fazna



Slika 4.6: Rayleighova fazna funkcija in dva primera Mieve fazne funkcije. Obe Mievi fazni funkciji močno sipata naprej, medtem ko Rayleighova sipa tudi nazaj.

funkcija [86]:

$$p_{MH}(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \frac{1}{4\pi}\left(\frac{1}{2} + \frac{9}{2}\left(\frac{1+\vec{\omega}\cdot\vec{\omega}'}{2}\right)^{8}\right) \text{ in} \tag{4.62}$$

$$p_{MM}(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \frac{1}{4\pi}\left(\frac{1}{2} + \frac{33}{2}\left(\frac{1+\vec{\omega}\cdot\vec{\omega}'}{2}\right)^{32}\right). \tag{4.63}$$

Tudi ti dve funkciji sta preveč računsko zahtevni za hitro vzorčenje, zato ju tako kot Rayleighovo aproksimiramo s kombinacijo dveh Schlickovih faznih funkcij.

Kombinirane fazne funkcije

Naštete fazne funkcije včasih niso dovolj splošne za opis želenega medija, zato jih lahko poljubno kombiniramo med sabo. To je v praksi pomembno predvsem pri stohastičnih simulacijah, ko moramo generirati vzorec žarkov, katerih porazdelitev bo enaka dani fazni funkciji. V tem primeru lahko vzorce generiramo s kombinacijo inverzne metode in metode zavrnitve. Prav tako lahko kompleksnejše funkcije, katerih vzorčenje bi bilo računsko prezahtevno,

aproksimiramo s kombinacijo enostavnejših funkcij, denimo Schlickove ali Henyey-Greensteinove.

Ker gre za porazdelitvene funkcije, je vsaka konveksna kombinacija faznih funkcij tudi fazna funkcija:

$$p(\mathbf{x}, \vec{\omega}, \vec{\omega}') = \sum_{i \in I} w_i p_I(\mathbf{x}, \vec{\omega}, \vec{\omega}'), \ \forall w_i \geq 0, \ \sum_{i \in I} w_i = 1. \tag{4.64}$$

Lahko se prepričamo, da zgoraj našteti pogoji (nenegativnost, normaliziranost in recipročnost) še vedno veljajo.

Poglavje 5

Realizacija

V tem poglavju predstavimo metode za reševanje sevalne enačbe. Naše rešitve temeljijo na stohastičnih metodah, zato najprej podajamo kratke osnove za razumevanje verjetnostne teorije. V nadaljevanju predstavimo metode Monte Carlo in njihove aplikacije v kontekstu upodabljanja volumetričnih podatkov.

5.1 Osnove verjetnostne teorije

Slučajna spremenljivka X je spremenljivka, katere vrednost določa dogodek v slučajnem procesu. *Diskretne slučajne spremenljivke* lahko zavzamejo le štev no mnogo različnih vrednosti (denimo $X \in \mathbb{N}$), medtem ko lahko *zvezne slučajne spremenljivke* zavzamejo neštevno mnogo različnih vrednosti (na primer $X \in \mathbb{R}^2$). V tem delu bomo v glavnem obravnavali zvezne slučajne spremenljivke, zato bomo pridevnik "zvezen" izpustili, če eksplicitna navedba ne bo nujno potrebna.

Verjetnost, da slučajna spremenljivka X zavzame določene vrednosti, določa njena *gostota porazdelitve* (angl. probability density function, PDF) $f_X(x)$, ki je definirana na množici izidov Ω s sledečimi lastnostmi:

1. *Verjetnost.* Gostota porazdelitve opisuje verjetnost, da slučajna spre-

menljivka zavzame vrednost iz množice $A \subseteq \Omega$:

$$\forall A \subseteq \Omega : P(x \in A) = \int_A f_X(x)\,\mathrm{d}A(x). \tag{5.1}$$

2. *Nenegativnost.* Ker gostota porazdelitve opisuje verjetnost, je njena vrednost vedno nenegativna:

$$\forall x \in \Omega : f_X(x) \geq 0. \tag{5.2}$$

3. *Normaliziranost.* Gostota porazdelitve je normalizirana tako, da je njen integral po celotni množici izidov enak 1. Z drugimi besedami, verjetnost gotovega dogodka je enaka 1:

$$P(x \in \Omega) = \int_A f_X(x)\,\mathrm{d}A(x) = 1. \tag{5.3}$$

Za funkcije ene spremenljivke lahko definiramo *porazdelitveno funkcijo* (angl. cumulative distribution function, CDF), ki opisuje verjetnost, da slučajna spremenljivka zavzame vrednost, manjšo od podane vrednosti:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(x)\,\mathrm{d}x. \tag{5.4}$$

Če je porazdelitvena funkcija zvezna, je njen odvod enak gostoti porazdelitve:

$$\frac{\mathrm{d}}{\mathrm{d}x} F_X(x) = f_X(x). \tag{5.5}$$

Definirajmo še *pričakovano vrednost* E, ki opisuje povprečno vrednost slučajnega procesa, če poskus ponovimo neskončno mnogokrat, *varianco* V, ki opisuje pričakovano vrednost kvadrata razlike med vrednostjo slučajne spremenljivke in njeno pričakovano vrednostjo, in *standardni odklon*, ki je kvadratni koren variance:

$$E(X) = \int_\Omega x f_X(x)\,\mathrm{d}A(x), \tag{5.6}$$

$$V(X) = E\left((X - E(X))^2\right) \text{ in} \tag{5.7}$$

$$\sigma(X) = \sqrt{V(X)}. \tag{5.8}$$

Pričakovano vrednost lahko posplošimo na poljubno funkcijo $g(X)$:

$$E(g(X)) = \int_{\Omega} g(x) f_X(x)\, \mathrm{d}A(x). \tag{5.9}$$

Iz zgornjih definicij lahko izpeljemo sledeče lastnosti:

$$E(aX) = aE(X), \tag{5.10}$$

$$E(X+Y) = E(X) + E(Y), \tag{5.11}$$

$$V(X) = E(X^2) - (E(X))^2, \tag{5.12}$$

$$V(aX) = a^2 V(X), \text{ in} \tag{5.13}$$

$$\sigma(aX) = a\sigma(X). \tag{5.14}$$

Porazdelitev lahko tudi *vzorčimo* in dobimo *slučajni vzorec* $X_1, X_2, ..., X_n$, sestavljen iz n članov. Pogosto nas zanimajo vzorci, ki so enako porazdeljeni in neodvisni (angl. independent and identically distributed, IID). Če vzorčimo neznano porazdelitev, lahko z dobljenimi IID vzorci ocenjujemo lastnosti dane porazdelitve. Tako lahko na primer pričakovano vrednost ocenimo s sledečo *cenilko*:

$$\hat{E}(X) = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{5.15}$$

Vsaka cenilka je tudi slučajna spremenljivka, saj je njena vrednost odvisna od slučajnega vzorca. Tako lahko govorimo tudi o pričakovani vrednosti cenilke in njeni varianci. Pogosto nas zanima *pristranskost* cenilke, ki jo izračunamo kot razliko med pričakovano vrednostjo cenilke in vrednostjo, ki jo ocenjujemo. Če razlike ni, gre za *nepristransko cenilko*. Pomembno je tudi, da je cenilka *dosledna*, kar pomeni, da se z večanjem vzorca njena vrednost približuje vrednosti, ki jo ocenjujemo.

Za dano vrednost, ki jo ocenjujemo, lahko obstaja več cenilk. Naletimo lahko denimo na primer, ko obstajata dve dosledni cenilki, ena pristranska in ena nepristranska, ki se razlikujeta v varianci. Varianca je neposredno povezana s hitrostjo konvergence in čeprav je nepristranskost dobrodošla lastnost, je lahko varianca pristranske cenilke manjša.

Za cenilko $\hat{E}$ izračunajmo pričakovano vrednost:

$$E(\hat{E}) = E\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) \tag{5.16}$$

$$= \frac{1}{n}\sum_{i=1}^{n} E(X_i) \tag{5.17}$$

$$= \frac{1}{n}\sum_{i=1}^{n} E(X) \tag{5.18}$$

$$= E(X). \tag{5.19}$$

Ker je pričakovana vrednost cenilke enaka vrednosti, ki jo ocenjujemo, je cenilka nepristranska.

Izračunajmo še varianco cenilke $\hat{E}$:

$$V(\hat{E}) = V\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) \tag{5.20}$$

$$= \frac{1}{n^2}\sum_{i=1}^{n} V(X_i) \tag{5.21}$$

$$= \frac{1}{n^2}\sum_{i=1}^{n} V(X) \tag{5.22}$$

$$= \frac{1}{n}V(X). \tag{5.23}$$

V prvem koraku smo uporabili dejstvo, da je $V(X+Y) = V(X) + V(Y)$ za neodvisni spremenljivki X in Y. Vidimo, da varianca pada z večanjem vzorca s hitrostjo $O(n^{-1})$, posledično pa je red napake $O(n^{-\frac{1}{2}})$.

5.2 Metode Monte Carlo

Običajne metode numerične matematike nam omogočajo, da izračunamo željene vrednosti po točno določenem postopku. Pogosto pa so taki analitični pristopi preveč računsko zahtevni, ali pa celo ne obstajajo. Takrat lahko uporabimo *stohastične metode*, ki z naključnim izvajanjem poskusov

in vrednotenjem njihovih izidov ocenijo željene vrednosti. Stohastične metode za ocenjevanje deterministčnih vrednosti se imenujejo metode Monte Carlo. K njihovemu vzponu in popularnosti v računalništvu sta prispevala predvsem Stanislaw Ulam in John von Neumann med drugo svetovno vojno.

Metode Monte Carlo so družina metod, ki delujejo na podlagi stohastičnih procesov. Izvirajo iz modeliranja sistemov z naključnimi procesi, katerih lastnosti lahko ocenjujemo s pomočjo vzorcev. Veliko matematičnih konceptov lahko formuliramo na podlagi ocenjevanja pričakovane vrednosti s cenilko, opisano v prejšnjem poglavju.

Najbolj pogosta aplikacija metode Monte Carlo je integriranje funkcij po domenah z več dimenzijami. Uporaba klasičnih numeričnih integracijskih metod (na primer Riemannova vsota) v takih situacijah ni primerna, saj je njihova konvergenca močno odvisna dimenzionalnosti domene ter odvedljivosti funkcije. Po drugi strani je konvergenca Monte Carlo integracije vedno enaka $O(n^{-1})$ in neodvisna od odvedljivosti funkcije.

Kot osnoven primer vzemimo integral

$$I = \int_a^b f(x)\,\mathrm{d}x. \tag{5.24}$$

Lahko ga ocenimo s cenilko $\hat{I}$, ki na podlagi enakomerno porazdeljenega vzorca $X_i \in [a,b], i = 1,2,...,n$ z gostoto porazdelitve $p(x) = \frac{1}{b-a}$ oceni vrednost I na sledeč način:

$$\hat{I} = \frac{b-a}{n}\sum_{i=1}^{n} f(X_i). \tag{5.25}$$

Pričakovana vrednost cenilke $\hat{I}$ je enaka

$$E(\hat{I}) = E\left(\frac{b-a}{n}\sum_{i=1}^{n} f(X_i)\right) \tag{5.26}$$
$$= \frac{b-a}{n}\sum_{i=1}^{n} E(f(X_i)) \tag{5.27}$$
$$= (b-a)E(f(X_i)) \tag{5.28}$$
$$= (b-a)\int_a^b f(x)p(x)\,\mathrm{d}x \tag{5.29}$$
$$= (b-a)\int_a^b f(x)\frac{1}{b-a}\,\mathrm{d}x \tag{5.30}$$
$$= \int_a^b f(x)\,\mathrm{d}x \tag{5.31}$$
$$= I, \tag{5.32}$$

zato je ta cenilka nepristranska.

Enakomerna porazdelitev vzorca ni obvezna, izberemo lahko namreč poljubno porazdelitev $p(x)$, dokler je $f(x) \neq 0 \implies p(x) > 0$:

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)}{p(X_i)}, \tag{5.33}$$
$$E(\hat{I}) = E\left(\frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)}{p(X_i)}\right) \tag{5.34}$$
$$= \frac{1}{n}\sum_{i=1}^{n} E\left(\frac{f(X_i)}{p(X_i)}\right) \tag{5.35}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\int_a^b \frac{f(x)}{p(x)}p(x)\,\mathrm{d}x \tag{5.36}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\int_a^b f(x)\,\mathrm{d}x \tag{5.37}$$
$$= \int_a^b f(x)\,\mathrm{d}x \tag{5.38}$$
$$= I. \tag{5.39}$$

Razlike med porazdelitvami $p(x)$ se kažejo v varianci cenilke:

$$V(\hat{I}) = V\left(\frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)}{p(X_i)}\right) \tag{5.40}$$

$$= \frac{1}{n^2}\sum_{i=1}^{n} V\left(\frac{f(X_i)}{p(X_i)}\right) \tag{5.41}$$

$$= \frac{1}{n} V\left(\frac{f(X_i)}{p(X_i)}\right) \tag{5.42}$$

$$= \frac{1}{n} E\left(\frac{f(X_i)}{p(X_i)} - E\left(\frac{f(X_i)}{p(X_i)}\right)\right)^2 \tag{5.43}$$

$$= \frac{1}{n} E\left(\frac{f(X_i)}{p(X_i)} - I\right)^2 \tag{5.44}$$

$$= \frac{1}{n}\int_a^b \left(\frac{f(x)}{p(x)} - I\right)^2 p(x)\,\mathrm{d}x \tag{5.45}$$

$$= \frac{1}{n}\int_a^b \frac{(f(x) - p(x)I)^2}{p(x)}\,\mathrm{d}x. \tag{5.46}$$

Iz zgornje enačbe je razvidno, da je $V(\hat{I}) = 0$ v primeru $p(x) = \frac{f(x)}{I}$, kar pomeni, da mora biti porazdelitev vzorcev $p(x)$ sorazmerna funkciji $f(x)$, ki jo integriramo. Kjer je $f(x) < 0$ lahko uporabimo $p(x) = \frac{|f(x)|}{I}$, kar še vedno privede do optimalne izbire (to trditev navajamo brez dokaza), toda v tem primeru varianca ni več enaka 0. Táko optimalno vzorčenje je v praksi neizvedljivo, saj je sorazmernostni faktor I vrednost, ki jo hočemo izračunati. Kljub temu lahko uporabimo funkcijo $p(x)$, ki je podobna $f(x)$ in tako bistveno zmanjšamo varianco cenilke. Takemu načinu vzorčenja pravimo *prioritetno vzorčenje* [38] in je en izmed najbolj razširjenih pristopov za zmanjševanje variance. Obstaja tudi *večkratno prioritetno vzorčenje* (angl. multiple importance sampling, MIS), ki je uporabno takrat, ko v enačbi sodeluje več porazdelitvenih funkcij. Tipičen primer uporabe je kombinacija prioritetnega vzorčenja glede na fazno funkcijo in glede na porazdelitev luči v prostoru [39].

5.3 Realizacija sevalne enačbe

Metodo Monte Carlo smo v našem delu uporabili za reševanje sevalne enačbe, tako da smo s cenilkami oblike, prikazane v enačbi 5.33, ocenili dva integrala, ki nastopata v enačbi: prvi opisuje prepustnost volumna, drugi pa vsipanje svetlobe.

5.3.1 Vzorčenje poti

Za namen vzorčenja poti modeliramo medij kot zmes mikroskopskih delcev, ki lahko absorbirajo ali sipajo svetlobo. Gostota delcev vpliva na hitrost absorpcije in verjetnost sipanja. Deterministične metode običajno ocenjujejo prepustnost volumna z izračunavanjem absorpcije zaporednih segmentov volumna po absorpcijski enačbi in pod predpostavko, da je gostota posameznega segmenta homogena. Glavna pomanjkljivost teh metod je natančno poznavanje karakteristik volumna in variabilnosti koeficientov sevalne enačbe, saj mora biti frekvenca vzorčenja dovolj visoka. Kljub temu se v praksi pogosto uporablja enakomerna Riemannova vrsta, ki podleže napakam zaradi podvzorčenja.

Drugačen pristop je simuliranje vsakega žarka svetlobe posebej, tako da izračunamo njegovo prosto pot skozi volumen do morebitnega sipanja ob trku z delcem medija. Če uspemo pravilno vzorčiti dolžino poti žarka glede na absorpcijski koeficient, bo za vrednotenje absorpcije poskrbela metoda Monte Carlo.

Gre za prioritetno vzorčenje prepustnosti $T(\mathbf{x} \leftrightarrow \mathbf{x}')$ v enačbi sevanja. Če je absorpcija homogena, se prepustnost poenostavi v absorpcijsko enačbo:

$$T(\mathbf{x} \leftrightarrow \mathbf{x}') = T(s) = e^{-\sigma s}, \tag{5.47}$$

z upoštevanjem $\sigma = \tau(\mathbf{x} \leftrightarrow \mathbf{x}')$ in $s = ||\vec{\omega}_{\mathbf{x}\mathbf{x}'}||$. V tem primeru lahko uporabimo *obratno metodo* oz. *metodo inverzije* (angl. inversion method), ki temelji na dejstvu, da je pri enakomerno porazdeljeni slučajni spremenljivki $U_{[0,1]}$ slučajna spremenljivka $X = F_X^{-1}(U)$, porazdeljena s porazdelitveno funkcijo

F_X. Najprej moramo izračunati gostoto porazdelitve, tako da normaliziramo prepustnost. Za enostavnejšo obravnavo upoštevamo tudi neskončen volumen:

$$p(s) = \frac{T(s)}{\int_0^\infty T(s)\,\mathrm{d}s} = \frac{e^{-\sigma s}}{[-\frac{1}{\sigma}e^{-\sigma s}]_0^\infty} = \sigma e^{-\sigma s}. \tag{5.48}$$

Nato izračunamo porazdelitveno funkcijo, tako da integriramo gostoto porazdelitve:

$$P(s) = \int_0^s p(s)\,\mathrm{d}s = [-e^{-\sigma s}]_0^s = 1 - e^{-\sigma s}. \tag{5.49}$$

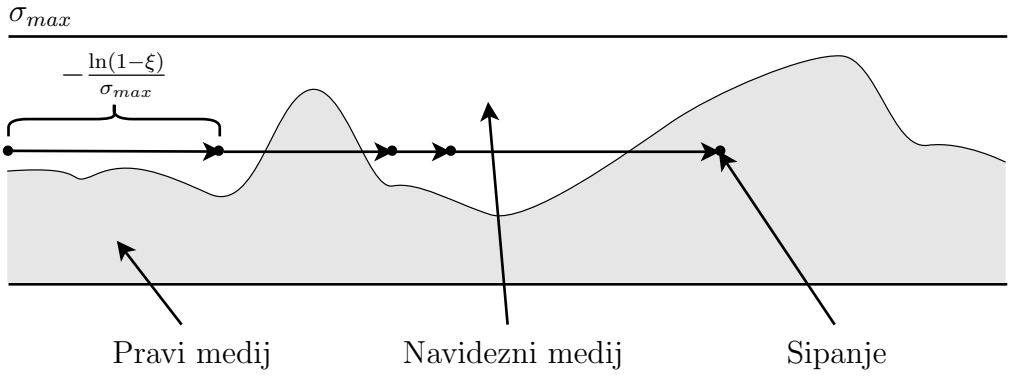
Zadnji korak vključuje generiranje naključnega števila $\xi \sim U_{[0,1]}$ in reševanje enačbe $P(s) = \xi$:

$$\xi = 1 - e^{-\sigma s}, \tag{5.50}$$

$$s = -\frac{\ln(1-\xi)}{\sigma} = -\frac{\ln \xi}{\sigma}. \tag{5.51}$$

Pri implementaciji se moramo zavedati, da zadnja enakost ne velja v primeru $\xi \in [0, 1)$, ki se pogosto pojavlja v generatorjih naključnih števil, saj je v primeru $\xi = 0$ vrednost logaritma nedefinirana.

Homogen primer ni splošen, ampak je dovolj enostaven, da nam bo pomagal pri obravnavi nehomogenega primera. Zgornji postopek lahko interpretiramo tudi kot iskanje dolžine žarka za dano prepustnost. V nehomogenem primeru nam ne preostane drugega, kot da dolžino žarka iščemo s prioritetnim vzorčenjem absorpcije. To lahko storimo z numerično integracijo absorpcije, na primer z enakomerno Riemannovo vrsto, toda tako vzorčenje je počasno, cenilka pa pristranska in nedosledna. V praksi se zato dandanes uporablja *Woodcockovo sledenje* (angl. Woodcock tracking, delta tracking) [43], imenovano po avtorju metode, ki jo je v šestdesetih letih uporabil za vzorčenje poti nevtronov. Prednost metode je neodvisnost od frekvence vzorčenja in s tem posledično nepristranska cenilka. Gre za vnos navideznih delcev v medij, tako da postane homogen. Njihov absorpcijski koeficient je enak 0, sipanje pa izločimo z Diracovo fazno funkcijo, ki vedno sipa v smeri potovanja žarka.



Slika 5.1: Woodcockovo sledenje medij modelira kot zmes pravih delcev, ki svetlobo sipajo, in navideznih delcev, ki svetlobe ne sipajo oz. jo sipajo vedno v smeri potovanja žarka.

V posameznem koraku Woodcockovega sledenja z uporabo inverzne metode podaljšamo pot žarka, nato pa z metodo zavrnitve (angl. rejection sampling) sprejmemo ali zavrnemo vzorec glede na razmerje pravih in navideznih delcev v mediju. To ponavljamo do sprejetja vzorca.

$$s_1 = 0, \tag{5.52}$$

$$s_{k+1} = s_k + \left(-\frac{\ln(1-\xi_i)}{\sigma_{max}}\right). \tag{5.53}$$

$$\text{Sprejmi } s_k, \text{ če } \zeta_k < \frac{\sigma(\mathbf{x} + s_k\vec{\omega})}{\sigma_{max}}. \tag{5.54}$$

Vzorčenje poti je zaradi dela z volumetričnimi podatki najbolj računsko in časovno zahtevna operacija v celotni simulaciji, zato je pomembno, da je čim hitrejše. Glavna pomanjkljivost Woodcockove metode je potreba po poznavanju zgornje meje maksimalne absorpcije σ_{max}, saj mora imeti navidezni homogeni medij vsaj maksimalno absorpcijo prvotnega medija. Lahko je tudi večja, ampak to negativno vpliva na hitrost metode, saj tako naredimo veliko število korakov in vzorcev volumetričnih podatkov, poleg tega pa večino vzorcev zavržemo z metodo zavrnitve. V praksi se zato uporabljajo podatkovne strukture za razdelitev prostora, ki jih uporabljamo za dinamično izbiro maksimalne absorpcije glede na položaj v volumnu [44, 45].

Metodo lahko izboljšamo tudi tako, da vsak dobljen vzorec namesto sprejetja oz. zavrnitve bolje izkoristimo [46].

5.3.2 Vzorčenje fazne funkcije

Z zgoraj opisanim vzorčenjem razdalje smo dobili prepotovano razdaljo svetlobnega žarka do prvega sipanja. Vsipanje je opisano z integralom, ki združi prispevke sevalnosti iz vseh smeri. Tudi tega v splošnem ne moremo izračunati analitično, temveč se moramo zateči k numeričnim metodam. Kvadraturne metode v tem primeru ne pridejo v poštev, saj je domena integrala dvodimenzionalna, integrand pa rekurziven. Izračunu vsipanja se lahko enostavno izognemo s ponovno uporabo metode Monte Carlo, tako da naključno izberemo smer sipanja in pot svetlobnega žarka nadaljujemo v tisti smeri. Pri tem moramo upoštevati še fazno funkcijo, ki opisuje smerno porazdelitev sipanja, kar lahko storimo z utežitvijo enakomerno porazdeljenih vzorcev ali s prioritetnim vzorčenjem fazne funkcije.

Poleg tega se lahko zgodi, da v smeri sipanja ne zadanemo izvora svetlobe in s tem razvrednotimo izračun. Temu se lahko izognemo s prioritetnim vzorčenjem glede na izvor svetlobe in žarek ne glede na fazno funkcijo usmerimo v izvor. Oba načina imata svoje prednosti in slabosti, lahko pa jih združimo z večkratnim prioritetnim vzorčenjem in se znebimo slabosti [39].

Vzorčenje Henyey-Greensteinove fazne funkcije

Prioritetno vzorčenje Henyey-Greensteinove fazne funkcije lahko izvedemo z obratno metodo. Najprej definirajmo

$$\mu = \cos(\vec{\omega} \cdot \vec{\omega}') \tag{5.55}$$

in zapišimo Henyey-Greensteinovo fazno funkcijo kot

$$p_{HG}(\mu) = \frac{1}{2} \frac{1 - g^2}{(1 + g^2 - 2g\mu)^{\frac{3}{2}}}. \tag{5.56}$$

Konstantni faktor je drugačen, saj smo prvotno funkcijo integrirali po vseh smereh, kjer je $\cos(\vec{\omega} \cdot \vec{\omega}') = \mu$. Iz dobljene gostote porazdelitve izračunajmo porazdelitveno funkcijo:

$$P_{HG}(\mu) = \int_{-1}^{\mu} p_{HG}(\mu)\,\mathrm{d}\mu \tag{5.57}$$

$$= \frac{1-g^2}{2g}\left(\frac{1}{\sqrt{1+g^2-2g\mu}} - \frac{1}{1+g}\right). \tag{5.58}$$

Rešitev enačbe $P_{HG} = \xi$ nam da metodo za prioritetno vzorčenje kosinusa lomnega kota:

$$\mu = \frac{1}{2g}\left(1+g^2-\left(\frac{1-g^2}{1+gs}\right)^2\right),\ s = 2\xi - 1, \tag{5.59}$$

kjer je ξ enakomerno porazdeljena slučajna spremenljivka na intervalu $[0, 1)$. Izraz za $g = 0$ ni definiran, toda v tem primeru gre za izotropno fazno funkcijo, katere vzorčenje je precej enostavnejše. Za izračun smeri $\vec{\omega}'$ potrebujemo le še azimut, ki ga vzorčimo enakomerno po intervalu $[0, 2\pi)$.

Vzorčenje Schlickove fazne funkcije

Enako lahko storimo s Schlickovo fazno funkcijo. Zapišimo jo kot funkcijo spremenljivke μ:

$$p_S(\mu) = \frac{1}{2}\frac{1-k^2}{(1+k\mu)^2}. \tag{5.60}$$

Dobljeno gostoto porazdelitve zopet integrirajmo, da dobimo porazdelitveno funkcijo:

$$P_S(\mu) = \int_{-1}^{\mu} p_S(\mu)\,\mathrm{d}\mu \tag{5.61}$$

$$= \frac{k^2-1}{2k}\left(\frac{1}{1+k\mu} - \frac{1}{1-k}\right). \tag{5.62}$$

Rešimo enačbo $P_S = \xi$, da dobimo metodo za prioritetno vzorčenje:

$$\mu = \frac{k-s}{ks-1},\ s = 2\xi - 1. \tag{5.63}$$

Zaradi bistveno manjšega števila aritmetičnih operacij je vzorčenje Schlickove fazne funkcije občutno hitrejše od vzorčenja Henyey-Greensteinove fazne funkcije.

Vzorčenje Rayleighove fazne funkcije

Predstavimo še Rayleighovo fazno funkcijo s spremenljivko μ:

$$p_R(\mu) = \frac{3}{8}(1+\mu^2). \tag{5.64}$$

Izračunajmo porazdelitveno funkcijo z integriranjem gostote porazdelitve:

$$P_R(\mu) = \int_{-1}^{\mu} p_R(\mu)\,\mathrm{d}\mu \tag{5.65}$$

$$= \frac{1}{2} + \frac{1}{8}\mu(3+\mu^2). \tag{5.66}$$

Enačba $P_R = \xi$ je tretje stopnje in ima tri rešitve, od katerih je le ena realna:

$$\mu = \left(s + \sqrt{s^2+1}\right)^{\frac{1}{3}} + \left(s - \sqrt{s^2+1}\right)^{\frac{1}{3}},\ s = 2(2\xi - 1). \tag{5.67}$$

Zaradi mnogih racionalnih eksponentov je prioritetno vzorčenje Rayleighove fazne funkcije preveč računsko zahtevno za uporabo v interaktivnih aplikacijah.

Poglavje 6

Implementacija

Metode, opisane v prejšnjih poglavjih, smo implementirali v predstavitveni aplikaciji. Izdelave aplikacije smo se lotili z mislijo na platformno agnostičnost in podporo mobilnim napravam, zato smo za temelj izbrali spletne tehnologije. Sodobni spletni brskalniki so dovolj zmogljivi, da z njimi lahko implementiramo kompleksne algoritme za upodabljanje, saj je razvijalcem v zadnjih letih omogočen tudi dostop do grafične strojne opreme. Prav tako lahko spletne aplikacije pišemo na platformno agnostičen način in tako dosežemo tudi izvajanje na mobilnih napravah.

Predstavitveno spletno aplikacijo smo razvili s sodobnimi spletnimi tehnologijami (HTML, JavaScript, CSS), za dostop do grafične strojne opreme pa smo uporabili programski vmesnik WebGL 2.0. Razen knjižnice jQuery za lažje delo s strukturo DOM in knjižnice Bootstrap za lažje delo z obrazci nismo uporabili zunanjih knjižnic.

V naslednjih poglavjih predstavimo našo obravnavo volumetričnih podatkov, njihovo predstavitev v računalniškem sistemu, vzorčenje in rekonstrukcijo tridimenzionalnih signalov ter s tem povezane numerične napake. Nadaljujemo z opisom programskega vmesnika WebGL in same predstavitvene aplikacije.

6.1 Volumetrični podatki, vzorčenje signala, rekonstrukcija in prenosna funkcija

Volumetrični podatki v najširšem smislu so funkcije prostora, ki točke v prostoru slikajo v določene vrednosti, denimo optične ali fizične lastnosti. Tako ima denimo lahko vsaka točka v prostoru svojo barvo, gostoto, maso in emisivnost. V ožjem smislu, ki je tudi bližje implementaciji, so volumetrični podatki vzorčene funkcije, ki jih moramo pred uporabo rekonstruirati. Najpogosteje se v praksi uporabljajo oblaki točk ali regularne mreže. V tem delu oblakov točk ne bomo obravnavali, saj so manj primerne za interaktivno vizualizacijo, rekonstrukcija funkcij iz danih podatkov pa precej težja. Omejili se bomo na regularne mreže in njihovo rekonstrukcijo v zvezno funkcijo.

Pri regularnih mrežah gre za podatke, enakomerno vzorčene po treh običajno medsebojno ortogonalnih oseh. Kljub temu, da imamo za dano funkcijo le končno mnogo vzorcev, v praksi pogosto potrebujemo njene vrednosti v poljubnih točkah. To operacijo imenujemo *rekonstrukcija podatkov* ali, nekoliko semantično nepravilno, *filtriranje*. Glede na območje rekonstrukcije gre lahko za *interpolacijo* ali *ekstrapolacijo*. Rekonstrukcija regularnih mrež podatkov v zvezno funkcijo je glede na potrebe lahko zelo enostavna ali pa praktično neizvedljiva. Najenostavnejša rekonstrukcija je le prevzem vrednosti najbližjega vzorca (angl. nearest neighbor filtering), ki kot rezultat vrne Voronojev diagram. Bolj natančna rekonstrukcija vzame več kot le eno točko v obravnavo, denimo osem točk za trilinearno rekonstrukcijo (angl. trilinear filtering). Tako rekonstrukcijo smo uporabili tudi v tem delu, saj je v sodobni grafični strojni opremi pogosto strojno podprta in za naše potrebe dovolj natančna. V splošnem moramo vzorčeno funkcijo obravnavati kot signal, ki ga vzorčimo z neko *frekvenco vzorčenja* ν_s. Rekonstruiramo ga lahko z Whittaker-Shannonovo formulo oz. njeno posplošeno obliko za večdimenzionalne signale [87]. Če je izvorni signal frekvenčno omejen s t. i. Nyquistovo frekvenco $\nu_{max} = \frac{\nu_s}{2}$, ga je na tak način po Nyquist-Shannonovem izreku mogoče rekonstruirati brez napak. Glavna po-

manjkljivost Whittaker-Shannonove formule je to, da moramo za eno samo rekonstruirano vrednost v izračunu upoštevati vse dane vzorce, kar je v praksi neizvedljivo, saj je računsko prezahtevno.

V vsakem primeru v celotnem procesu od zajema podatkov do rekonstrukcije prihaja do napak:

1. *Vzorčenje.* Pri vzorčenju signala moramo poskrbeti, da gre za frekvenčno omejen signal, sicer v vzorce vnašamo šum. Za to lahko poskrbimo na primer z nizkoprepustnim filtrom, kar pa v praksi pogosto ni mogoče.

2. *Kvantizacija.* Podatke shranjujemo v digitalni obliki, zato v signal vnašamo kvantizacijski šum. Glede na format zapisa in stopnjo kvantizacije je šum bolj ali manj opazen. V praksi napake tega tipa običajno ne predstavljajo večjih težav.

3. *Rekonstrukcija.* Pri rekonstrukciji signala pride do napake zaradi rekonstrukcijskega filtra in zaradi numeričnih napak. Glede na želeno hitrost rekonstrukcije lahko to napako zmanjšamo.

V tem delu procesa zajema podatkov ne obravnavamo, saj nas zanima le upodabljanje in s tem rekonstrukcija, zato predpostavljamo, da je bil signal pravilno vzorčen. Kljub temu lahko z izbiro oblike podatkov vplivamo na kvantizacijski šum, z izbiro rekonstrukcijskega filtra pa na rekonstrukcijski šum. Ker so napake teh tipov običajno malo zaznavne, jih v tem delu le omenjamo.

V sevalni enačbi nastopa več optičnih lastnosti medija: absorpcijski koeficient, koeficient sipanja, fazna funkcija in emisivnost. Prostorsko variabilne lastnosti lahko opišemo z volumetričnimi podatki, smerno variabilnih pa zaradi višje dimenzionalnosti ne moremo. Tako na primer ne moremo predstaviti medijev, ki sipajo ali sevajo anizotropno. Kljub temu lahko problem omejimo in z volumetričnimi podatki opišemo le parametre, ki na dane anizotropne lastnosti posredno vplivajo, na primer parameter anizotropije pri Henyey-Greensteinovi fazni funkciji. Tudi tako omejen problem v praksi

še vedno predstavlja težavo zaradi velike količine podatkov - vsaka optična lastnost medija namreč zahteva svoj zapis. Pogosto so podatki, ki jih upodabljamo, le skalarni, poleg tega pa niti ne predstavljajo nobene optične lastnosti, temveč kaj povsem drugega, denimo gostoto snovi ali koncentracijo barvila. Tedaj moramo uporabiti t. i. *prenosno funkcijo* (angl. transfer function), ki preslika dane podatke v optične lastnosti. Prenosna funkcija je lahko tudi večdimenzionalna [88], tako da v obravnavo vzame še velikost gradienta signala, njegovo smer ali celo višje odvode. Večinoma se v praksi uporabljajo dvodimenzionalne prenosne funkcije, saj predstavljajo najboljši kompromis med enostavnostjo manipulacije in ekspresivnostjo. Take smo uporabili tudi v naši predstavitveni aplikaiciji.

6.2 Programski vmesnik WebGL

6.2.1 Standardi in implementacije

Dostop do grafične procesne enote (angl. graphics processing unit, GPU) iz sodobnega spletnega brskalnika je omogočen prek programskega vmesnika (angl. application programming interface, API) WebGL. Gre za standardiziran način za dostop do storitev grafične strojne opreme prek jezika JavaScript in objektnega modela dokumenta (angl. document object model, DOM). Kontekst WebGL namreč ustvarimo s klicem funkcije `getContext` z argumentom `'webgl'` na HTML5 elementu `<canvas>`. Vmesnik temelji na vmesniku OpenGL (odprta grafična knjižnica), bolj natančno na različici za vgrajene sisteme (angl. embedded systems, ES) OpenGL ES 2.0, ki omogoča enoten dostop do grafične strojne opreme na nivoju domorodne aplikacije. Različica za vgrajene sisteme sledi običajni različici, toda v primerjavi s slednjo ji manjka precej funkcionalnosti. Prav tako je programski vmesnik nekoliko poenostavljen, kar odraža njegovo namembnost uporabe. Večina implementacij standarda WebGL prevaja spletno kodo v klice programskega vmesnika OpenGL, obstajajo pa tudi take, ki uporabljajo Microsoftov vmesnik DirectX (prek abstrakcijske plasti ANGLE) ali pa ponujajo povsem pro-

gramsko izvedbo na centralni procesni enoti. Razlog je običajno v strojni podpori in dostopnosti vmesnika OpenGL na nivoju domorodne aplikacije.

Vmesnik WebGL 1.0 [72], ki temelji na vmesniku OpenGL ES 2.0 [73], je začel nastajati leta 2006 kot eksperiment Vladimirja Vukićevića. Projekt je kmalu zatem začel standardizirati konzorcij Khronos Group, ki je tudi glavna pogonska sila za standard OpenGL. Prva različica standarda WebGL 1.0 je bila objavljena leta 2011 in s tem kmalu dobila množično podporo razvijalcev spletnih aplikacij. Vmesnik je podprt v vseh sodobnih spletnih brskalnikih, tako na namiznih kot mobilnih napravah. Kontekst pridobimo s klicem funkcije `getContext` elementa `<canvas>` z argumentom `'webgl'` oz. `'experimental-webgl'` v starejših brskalnikih.

Od leta 2012 je bila v razvoju tudi druga različica standarda, WebGL 2.0 [79], ki temelji na vmesniku OpenGL ES 3.0 [80] in odpravi mnoge pomanjkljivosti prve različice standarda, tako s stališča programskega vmesnika kot tudi zmogljivosti. Končni osnutek standarda je bil objavljen v začetku leta 2017, sredi leta pa so bile privzeto vključene implementacije v številnih sodobnih spletnih brskalnikih (Google Chrome 56 za namizne naprave, Google Chrome 58 za operacijski sistem Android, Android WebView 58, Opera 43, Mozilla Firefox 51). Nova različica zagotavlja enotno ustvarjanje konteksta s klicem funkcije `getContext` elementa `<canvas>` z argumentom `'webgl2'`.

Za oba standarda je na voljo vrsta uradnih in neuradnih razširitev, ki dopolnjujejo ali spreminjajo funkcionalnosti temeljnega standarda. Z njihovo uporabo postane implementacija bolj fleksibilna in zmogljiva. Za skladnost s standardom je nujna le implementacija temeljnih funkcionalnosti, ne pa tudi razširitev. Ker smo predstavitveno aplikacijo razvijali z namenom podpore široki paleti raznovrstnih naprav, smo se uporabi razširitev v splošnem izogibali.

Druga različica standarda WebGL v primerjavi s prvo prinaša vrsto izboljšav, mnoge funkcionalnosti pa so izpostavljene v temeljni obliki standarda in ne kot uradne razširitve, katerih implementacija je povsem opcijska. Za naše delo sta bistveni predvsem dve taki izboljšavi: podpora za tridimen-

zionalne teksture in širši nabor slikovnih formatov, ki nam omogočajo upodabljanje slik z velikim dinamičnim razponom (angl. high dynamic range, HDR). V temeljni obliki standarda je namreč vključena podpora za teksture v formatu s plavajočo vejico, kljub temu pa je pisanje v teksture takih formatov še vedno dostopno le kot uradna razširitev.

Specifikacije standardov WebGL so dostopne na Khronosovi uradni spletni strani `https://www.khronos.org/registry/webgl`, temeljijo pa na specifikacijah standarda OpenGL, dostopnih na naslovu `https://www.khronos.org/registry/OpenGL`.

Podporo standarda v določenem spletnem brskalniku lahko preverimo na več različnih načinov. Sodobni brskalniki omogočajo pregled strojne podpore, na primer Google Chrome prek naslova `chrome://gpu`. Podporo in pravilnost implementacije standardov WebGL lahko preverimo na Khronosovi uradni spletni strani s skladnostnimi testi `https://www.khronos.org/registry/webgl/sdk/tests/webgl-conformance-tests.html`. Lastnosti, parametre in razširitve določene implementacije lahko preverimo s klici določenih funkcij standarda. Za ta namen lahko uporabimo tudi spletno stran `http://webglreport.com`, ki te podatke navede v bolj pregledni obliki. Statistične podatke o podpori, lastnostih, parametrih in dostopnosti razširitev najdemo na spletni strani `https://webglstats.com`.

6.2.2 Programski vmesnik in procesni model

Zaradi svoje vsestranskosti je WebGL izredno zmogljiv programski vmesnik, a hkrati tudi precej kompleksen. Že najbolj osnovne operacije upodabljanja zahtevajo nezanemarljivo količino kode, zato za lažje delo z vmesnikom obstaja nemalo knjižnic, ki že tako abstrakten vmesnik še dodatno abstrahirajo. Posledično je prototipiranje in visokonivojsko delo z grafiko bistveno poenostavljeno, za izrabo celotne zmogljivosti pa moramo kljub temu poznati specifike sistema, na katerem teče aplikacija, in način uporabe vmesnika prilagoditi aplikaciji. Za razumevanje arhitekture naše aplikacije je zato nujno poznavanje standarda WebGL in procesnega modela, na katerem temelji.

WebGL uporablja t. i. *cevovodni procesni model* (angl. pipeline processing model), v katerem je več funkcionalno ločenih stopenj cevovoda povezanih v celoto, ki s t. i. *tokovnim procesiranjem* (angl. stream processing) vhodni *tok podatkov* preoblikuje v izhodni tok podatkov. Vhodni tok podatkov so oglišča geometrijskih objektov (angl. vertices), izhodni tok pa so slikovne točke oz. fragmenti (angl. fragments). Pri programiranju z vmesnikom WebGL gre torej za dvofazni proces: programiranje cevovoda in nato njegovo izvajanje na danih podatkih. Ti dve fazi se pogosto izvajata večkrat na sekundo in tako dobimo animacijo.

Podatke o ogliščih podamo v obliki medpomnilnikov, kamor zapišemo njihove lokacije in morebitne dodatne atribute, na primer barvo in orientacijo. Prva stopnja cevovoda, imenovana *senčilnik oglišč* (angl. vertex shader), je programabilna stopnja, ki se izvaja na vsakem oglišču posebej in vrne lokacijo oglišča skupaj s podatki za naslednje stopnje. Druga stopnja cevovoda oglišča sestavi v osnovne geometrijske objekte (točke, daljice ali trikotnike) glede na nastavitve cevovoda. Sledi naknadna obdelava geometrijskih objektov glede na površino upodabljanja (tretja stopnja) in rasterizacija v tok fragmentov (četrta stopnja). Peta stopnja cevovoda, imenovana *senčilnik fragmentov* (angl. framgnent shader), je programabilna stopnja, ki se izvaja na vsakem fragmentu posebej in vrne barvo in globino fragmenta. Dobljeni fragmenti se po zadnji (šesti) stopnji naknadne obdelave zapišejo v t. i. *medpomnilniški okvir* (angl. framebuffer), ki je lahko tekstura ali zaslon.

Programabilni stopnji cevovoda predstavljata glavnino procesiranja in morata biti karseda učinkoviti. Predvsem se moramo izogibati zaporednim operacijam, pogojnemu izvajanju in prepogostim pomnilniškim dostopom. Pomnilniški dostopi so daleč najbolj časovno potratne operacije, zato moramo njihovo količino minimizirati in čim bolj izrabljati predpomnilnik. V senčilniku oglišč lahko to storimo že z organizacijo podatkov v pomnilniku, tako da atribute oglišč prepletamo in tako preprečimo pretirano razpršene dostope. V senčilniku fragmentov lahko uporabljamo konstanten pomnilnik, ki ga prevajalnik lahko celo optimizira z uporabo registrov, ali pa uporabljamo

teksture, do katerih tudi dostopamo prek posebnega predpomnilnika, hkrati pa lahko izrabljamo morebitno strojno podprto interpolacijo podatkov.

6.3 Struktura aplikacije

Naša aplikacija je zgrajena po cevovodnem modelu z neposredno podporo stohastičnemu upodabljanju. Cevovod je sestavljen iz dveh stopenj, ki se izvajata iterativno in s tem izboljšujeta končno sliko. Prva stopnja je stopnja upodabljanja, v kateri t. i. *upodabljalnik* (angl. renderer) izriše volumetrične podatke v sliko formata HDR, ki vstopi v drugo stopnjo - zaporedje korakov naknadnega procesiranja. t. i. *tonski slikar* (angl. tone mapper) kot zadnji člen zaporedja zmanjša dinamični razpon slike in vrne rezultat v formatu majhnega dinamičnega razpona (angl. low dynamic range, LDR). Tako obdelana slika je primerna za izris na zaslonu.

Vsak konkreten tip upodabljalnika mora implementirati vmesnik abstraktnega upodabljalnika, ki ga sestavljajo sledeči štirje koraki:

1. *Generacija.* Upodabljalnik v tem koraku generira aproksimacijo končne slike. Generirani podatki niso nujno podatki o barvi slikovnih točk, temveč poljubni podatki, ki se kasneje lahko uporabijo za generiranje končne slike. Običajno je to računsko najzahtevnejši del cevovoda.

2. *Integracija.* Generirani podatki se v tem koraku uporabijo za iterativno izboljšavo aproksimacije. Tudi akumulirani podatki so lahko poljubni in ne nujno podatki o barvi slikovnih točk.

3. *Upodabljanje.* Akumulirani podatki se preoblikujejo v barvno sliko v formatu HDR, primerno za nadaljnje procesiranje v sledečih stopnjah cevovoda.

4. *Ponastavitev.* Določeni dogodki lahko sprožijo ponastavitev cevovoda in s tem upodabljalnika v začetno stanje.

Vsak upodabljalnik za podporo navedenim korakom vsebuje še nekaj medpomnilniških okvirjev:

1. *Generacijski okvir*. Pomnilnik, v katerega upodabljalnik v koraku generacije zapiše podatke.

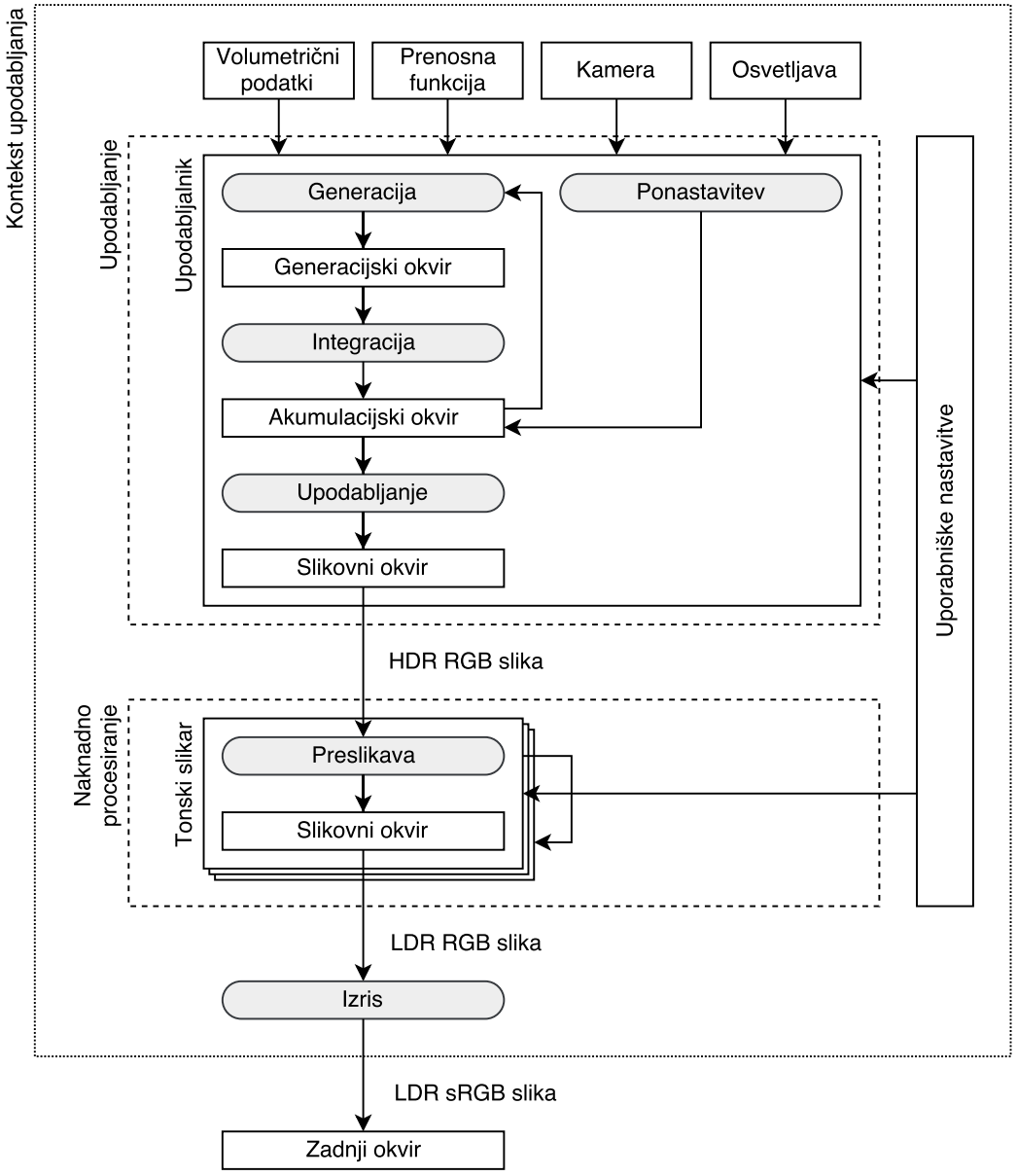
2. *Akumulacijski okvir*. Služi kot medpomnilnik za proces iterativne izboljšave. V koraku integracije ga upodabljalnik posodobi s podatki iz generacijskega okvirja, v koraku ponastavitve pa postavi nazaj v začetno stanje.

3. *Slikovni okvir*. Pomnilnik, v katerega upodabljalnik v koraku upodabljanja z uporabo podatkov iz akumulacijskega okvirja zapiše končno sliko prve stopnje cevovoda, ki se nato pošlje v nadaljnjo obdelavo drugi stopnji. Slikovni okvir mora vsebovati podatke v formatu HDR.

Tonski slikarji sledijo nekoliko manj kompleksnemu vmesniku abstraktnega tonskega slikarja, ki vsebuje le korak preslikave. Za zapis podatkov preslikave vsak tonski slikar vsebuje slikovni okvir v formatu HDR ali LDR, toda zadnji tonski slikar v cevovodu mora vsebovati slikovni okvir v formatu LDR.

Stopnje cevovoda so povezane s *kontekstom upodabljanja*, ki poleg volumetričnih podatkov, prenosne funkcije, podatkov o kameri in podatkov o osvetljavi hrani ter vzdržuje še celoten kontekst vmesnika WebGL. Poleg tega je zadolžen za proženje in izvedbo cevovoda v tesnih časovnih intervalih ter za njegovo ponastavitev po spremembah v sceni. Slikovni okvir zadnjega tonskega slikarja izriše v zadnji okvir (angl. back buffer), ki se preko elementa HTML5 `<canvas>` izriše na zaslon. Na format in velikost zadnjega okvirja ne moremo vplivati, saj je za njegovo upravljanje zadolžen gonilnik grafične strojne opreme. Kontekst upodabljanja je shematično predstavljen na sliki 6.1.

V naslednjih odstavkih opišemo implementacije treh upodabljalnikov in dveh tonskih slikarjev glede na opisane vmesnike.



Slika 6.1: Kontekst upodabljanja, ki vsebuje volumetrične podatke, prenosno funkcijo ter podatke o kameri in osvetljavi. Zadolžen je za izvajanje cevovoda, ki ga sestavljata stopnji upodabljanja in naknadnega precesiranja. Slika prikazuje tudi tok podatkov skozi medpomnilniške okvirje upodabljalnika in tonskih slikarjev.

6.3.1 Upodabljalnik največje intenzitete

Projekcija največje intenzitete (angl. maximum intensity projection, MIP) je en izmed najenostavnejših algoritmov za upodabljanje volumetričnih podatkov. Gre za iskanje in projekcijo največje vrednosti v volumnu vzdolž žarka:

$$I = \mathcal{T}(\max_t \mathcal{V}(t)), \tag{6.1}$$

kjer je $\mathcal{V}$ rekonstruirana vrednost volumna in $\mathcal{T}$ prenosna funkcija. Najdeno največjo intenziteto I s prenosno funkcijo preslikamo v barvo in prikažemo na zaslonu. Težava v tem pristopu je nezmožnost globinskega razlikovanja med deli volumna, saj razen perspektive ne ponuja nobenih podatkov o globini.

V predstavitveni aplikaciji je upodabljalnik največje intenzitete izveden na sledeč način:

1. *Generacijski okvir.* Generacijski okvir hrani prebrano vrednost iz volumna.

2. *Akumulacijski okvir.* Akumulacijski okvir hrani trenutno največjo vrednost med vsemi do tega trenutka prebranimi vrednostmi iz volumna.

3. *Slikovni okvir.* Slikovni okvir hrani podatke o barvi, ki jo dobimo z aplikacijo prenosne funkcije na največji vrednosti iz volumna.

4. *Generacija.* Za vsako slikovno točko generacijskega okvirja izberemo točko na žarku. V tisti točki preberemo vrednost volumna in jo shranimo v generacijski okvir. Velikemu številu prehodov prek cevovoda se izognemo tako, da izberemo več točk na žarku in že znotraj te faze cevovoda izberemo največjo vrednost med temi točkami volumna.

5. *Integracija.* Izberemo večjo vrednost med akumulacijskim in generacijskim okvirjem ter jo zapišemo nazaj v akumulacijski okvir.

6. *Upodabljanje.* Vrednost iz akumulacijskega okvirja spustimo prek prenosne funkcije in dobljen rezultat zapišemo v slikovni okvir.

7. *Ponastavitev.* Vrednosti vseh slikovnih točk akumulacijskega okvirja postavimo na najmanjšo vrednost celotnega volumna.

Format generacijskega in akumulacijskega okvirja je načeloma lahko poljuben, v idealnem primeru pa enak formatu volumetričnih podatkov. V našem primeru smo izbrali 16-biten format s plavajočo vejico, saj je dovolj natančen pri majhnih vrednostih, hkrati pa ima dovolj velik razpon.

6.3.2 Upodabljalnik nivojskih ploskev

Koncept pri upodabljalniku nivojskih ploskev je podoben kot pri projekciji največje intenzitete. Volumen razdelimo na tri dele (notranjega, zunanjega in nivojsko ploskev) z vpeljavo potencialne funkcije:

$$F(\mathbf{x}) = \mathcal{V}(\mathcal{T}(\mathbf{x})) - \mu, \tag{6.2}$$

kjer je $\mathcal{V}$ rekonstruirana vrednost volumna in $\mathcal{T}$ prenosna funkcija.

V predstavitveni aplikaciji kot notranjost upoštevamo pozitiven potencial. Rešitev enačbe $F(\mathbf{x}) = 0$ so nivojske ploskve za potencial μ. Najenostavneje jih izrišemo z lokalnim osvetlitvenim modelom in pod predpostavko, da niso prosojne. Iskanje najbližje nivojske ploskve vzdolž žarka in upodabljanje z lokalnim osvetlitvenim modelom smo implementirali v predstavitveni aplikaciji na sledeč način:

1. *Generacijski okvir.* Generacijski okvir hrani globino vzorca, če je potencialna funkcija pozitivno ovrednotena, sicer pa negativno vrednost.

2. *Akumulacijski okvir.* Akumulacijski okvir hrani najmanjšo globino med vsemi do tega trenutka obravnavanimi vzorci.

3. *Slikovni okvir.* Slikovni okvir hrani podatke o barvi, ki jo izračunamo z lokalnim osvetlitvenim modelom.

4. *Generacija.* Za vsako slikovno točko generacijskega okvirja izberemo točko na žarku. V tisti točki ovrednotimo potencialno funkcijo in v generacijski okvir shranimo globino vzorca (razdaljo od kamere do točke)

ali poljubno negativno vrednost, če je potencialna funkcija negativno ovrednotena.

5. *Integracija.* Obravnavamo globini iz generacijskega in akumulacijskega okvirja. Izberemo manjšo vrednost med dvema globinama ali le tisto, ki je na voljo. Če nobena globina ni na voljo, zapišemo negativno vrednost.

6. *Upodabljanje.* Če je globina v akumulacijskem okvirju na voljo, iz nje rekonstruiramo točko v prostoru in jo izrišemo z lokalnim osvetlitvenim modelom. Kot normalo površine vzamemo gradient potencialne funkcije, ki ga lahko ovrednotimo v tej stopnji cevovoda ali pa že v generacijski stopnji.

7. *Ponastavitev.* V vse slikovne točke akumulacijskega okvirja zapišemo negativno vrednost.

Format generacijskega in akumulacijskega okvirja je v našem primeru 16-biten s plavajočo vejico. Odločili smo se za vrednotenje gradienta že v generacijski stopnji, zato okvirja poleg globine vsebujeta še vrednost gradienta.

6.3.3 Upodabljalnik emisijsko-absorpcijskega modela

Emisijsko-absorpcijski model je izredno razširjen v praksi zaradi njegove enostavnosti in posledično hitrosti izvajanja. Gre za sevalno enačbo, pri kateri zanemarimo sipanje ter uporabimo prenosno funkcijo za izračun emisijskega spektra ter absorpcijske enačbe:

$$L(\mathbf{x} \to \vec{\omega}) = T(\mathbf{x} \leftrightarrow \mathbf{x}'')L(\mathbf{x}'' \to \vec{\omega}) + \int_0^s T(\mathbf{x} \leftrightarrow \mathbf{x}')L_e(\mathbf{x} \to \vec{\omega})\,\mathrm{d}s. \quad (6.3)$$

Zaradi prisotnosti absorpcijske enačbe je možno globinsko razlikovanje, kljub temu pa je globinska zaznava zaradi pomanjkanja pomembnih svetlobnih pojavov (na primer senčenja in senc) dokaj omejena. V predstavitveni aplikaciji je emisijsko-absorpcijski model implementiran z enostavno numerično

metodo, ki ne temelji na stohastičnem modelu in je zato podvržen napakam zaradi podvzorčenja. Tak način je v praksi najbolj pogost zaradi enostavne implementacije. Izpeljava numerične metode je podrobno opisana v [12].

Upodabljalnik emisijsko-absorpcijskega modela je v predstavitveni aplikaciji implementiran na sledeč način:

1. *Generacijski okvir.* Generacijski okvir hrani za vsako slikovno točko njeno končno barvo.

2. *Akumulacijski okvir.* Akumulacijski okvir hrani enake podatke kot generacijski okvir.

3. *Slikovni okvir.* Slikovni okvir hrani enake podatke kot generacijski in akumulacijski okvir.

4. *Generacija.* Za vsako slikovno točko generacijskega okvirja generiramo žarek in se po njem v enakomernih korakih sprehodimo skozi volumen. Na vsakem koraku vzorčimo volumetrične podatke in vzorec preslikamo v emisijo in absorpcijo prek prenosne funkcije. Dobljene podatke uporabimo za ovrednotenje trenutnega koraka emisijsko-absorpcijskega modela.

5. *Integracija.* Vrednost iz generacijskega okvirja prepišemo v akumulacijski okvir.

6. *Upodabljanje.* Vrednost iz akumulacijskega okvirja prepišemo v slikovni okvir.

7. *Ponastavitev.* Ponostavitev v tem primeru nima vpliva, vsekakor pa v vse točke akumulacijskega okvirja zapišemo vrednost 0.

Format okvirjev je v našem primeru `RGBA`, ki je običajno 8-biten. Glede na format prenosne funkcije bi bil lahko tudi 16-biten s plavajočo vejico.

6.3.4 Upodabljalnik sevalne enačbe z enkratnim sipanjem

Upodabljalnik sevalne enačbe implementira opisane metode Monte Carlo z vzorčenjem razdalje potovanja fotonov v mediju in z vzorčenjem fazne funkcije ali točke na izvoru svetlobe. Generirano pot uporabi za oceno sevalnosti, ki konvergira k rešitvi enačbe. V primerjavi z upodabljalnikoma največje intenzitete in nivojskih ploskev je v splošnem dokaj enostaven, saj se vsa kompleksnost skriva v generacijski stopnji. Implementacija sledi abstraktnemu vmesniku:

1. *Generacijski okvir.* Generacijski okvir hrani za vsako slikovno točko ocenjeno sevalnost.

2. *Akumulacijski okvir.* Akumulacijski okvir hrani tekoče povprečje generacijskih okvirjev.

3. *Slikovni okvir.* Slikovni okvir hrani končno sliko upodabljalnika in je v našem primeru enak akumulacijskemu.

4. *Generacija.* Za vsako slikovno točko generacijskega okvirja izberemo točko sipanja na žarku po Woodcockovi metodi. Izberemo smer sipanja in ocenimo prepustnost medija v tej smeri do izvora svetlobe.

5. *Integracija.* Akumulacijski okvir posodobimo z ravnokar izračunano oceno iz generacijske stopnje.

6. *Upodabljanje.* Akumulacijski okvir prepišemo v slikovnega. V bolj napredni implementaciji bi na tem mestu lahko na primer odstranjevali šum.

7. *Ponastavitev.* V vse slikovne točke akumulacijskega okvirja zapišemo vrednost 0.

Format vseh okvirjev je 16-biten `RGBA` s plavajočo vejico.

6.3.5 Tonski slikar intenzitetnega obsega

Tonski slikar intenzitetnega obsega stisne dinamični razpon slike v vrednosti med 0 in 1, primerne za upodobitev na zaslonu. To dosežemo z linearno preslikavo vhodne intenzitete L_i z intervala $[L_{min}, L_{max}]$ na interval $[0, 1]$, da dobimo izhodno intenziteto L_o:

$$L_o = \frac{L_i - L_{min}}{L_{max} - L_{min}}. \tag{6.4}$$

Vrednosti zunaj intervala $[L_{min}, L_{max}]$ preslikamo v 0 ali 1, za kar običajno poskrbi že strojna ali programska oprema, odvisno od implementacije standarda WebGL. Prednost opisanega pristopa je predvsem v uporabi enostavnih aritmetičnih operacij in njihovi hitri izvedbi. Obseg je hitro nastavljiv v uporabniškem vmesniku, njegovo spreminjanje pa ne zahteva kompleksnih operacij. Kljub temu pa je glavna pomanjkljivost metode v tem, da se velik del zastopanih intenzitet lahko preslika v vrednosti 0 ali 1, kar posledično pomeni izgubo informacij na tem delu slike.

6.3.6 Reinhardov tonski slikar

Reinhardov tonski slikar [89] odpravi glavno pomanjkljivost tonskega slikarja intenzitetnega obsega z uporabo injektivne, monotono naraščajoče funkcije, ki slika z intervala $[0, \infty)$ na interval $[0, 1]$:

$$L_o = \frac{L_i}{\frac{1}{\varepsilon} + L_i}. \tag{6.5}$$

Parameter ε v zgornji formuli predstavlja ekspozicijo kamere in je nastavljiv v uporabniškem vmesniku. Tudi ta tonski slikar ne zahteva kompleksnih operacij, temveč le seštevanje in deljenje, zato je izredno hiter. Običajno je dovolj prilagodljiv z ekspozicijo kamere, zato se v praksi pogosto uporablja. V situacijah, ko potrebujemo večjo fleksibilnost, kot jo ponuja ta tonski slikar, se uporabljajo izboljšane in razširjene različice Reinhardove metode, denimo logaritmični tonski slikar [90] ali filmski tonski slikar [91].

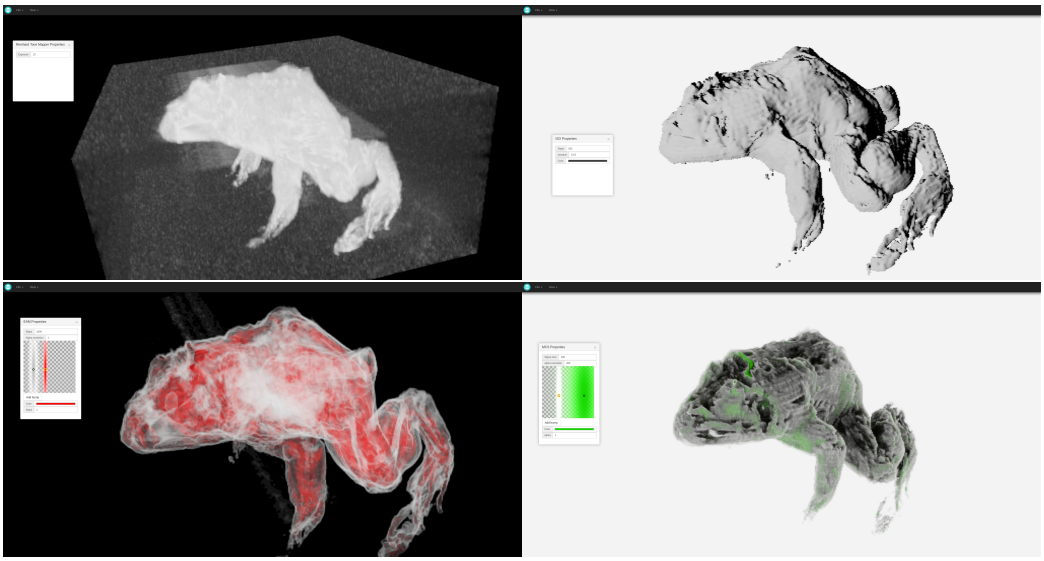
Poglavje 7

Rezultati

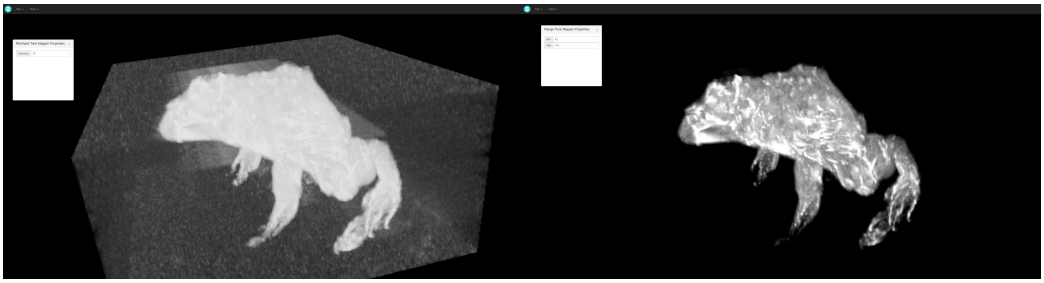
Za preizkus našega dela smo izdelali predstavitveno aplikacijo. S spletnimi tehnologijami, natančneje z jezikom JavaScript in s programskim vmesnikom WebGL 2.0, smo implementirali metode za upodabljanje volumetričnih podatkov v spletnem brskalniku. Predstavitvena aplikacija je platformno agnostična, tako da jo je moč izvajati tako na namiznih kot mobilnih napravah v sodobnem spletnem brskalniku s podporo standarda WebGL 2.0. Po svoji zasnovi je interaktivna in deluje v realnem času, tako da so spremembe volumna, prenosne funkcije, kamere in osvetljave takoj vidne na zaslonu. Narava metod Monte Carlo, ki so bistvenega pomena za interaktivnost, se kaže v šumu, prisotnem v končni sliki, ki hitro konvergira k pravilni upodobitvi.

Zaslonske posnetke različnih upodabljalnikov lahko vidimo na sliki 7.1, razliko med Reinhardovim tonskim slikarjem in tonskim slikarjem intenzitetnega obsega v kontekstu upodabljalnika največje intenzitete pa na sliki 7.2. Vse metode so interaktivne in delujejo v realnem času. Zaradi svoje enostavnosti je upodabljalnik največje intenzitete najhitrejši, najpočasnejši pa upodabljalnik enačbe sevanja.

Prenosna funkcija je interaktivna, tako da so spremembe takoj vidne na končni sliki. Vsaka sprememba prenosne funkcije ponastavi cevovod upodabljanja, kljub temu pa to ne predstavlja večje ovire pri interaktivnem raziskovanju podatkov, saj vse metode konvergirajo dovolj hitro, da je pre-



Slika 7.1: Zaslonski posnetki različnih upodabljalnikov: upodabljalnik največje intenzitete (zgoraj levo), upodabljalnik nivojskih ploskev (zgoraj desno), upodabljalnik emisijsko-absorpcijskega modela (spodaj levo) in upodabljalnik sevalne enačbe (spodaj desno).



Slika 7.2: Zaslonska posnetka dveh različnih tonskih slikarjev, uporabljenih skupaj z upodabljalnikom največje intenzitete: Reinhardov tonski slikar (levo) in tonski slikar intenzitetnega obsega (desno).

Slika 7.3: Različne prenosne funkcije, uporabljene na istih volumetričnih podatkih. Poljubne spremembe so mogoče med samim izvajanjem, saj je celoten sistem prilagojen interaktivnemu raziskovanju volumetričnih podatkov.

poznavna slika vidna na zaslonu že v delčku sekunde. Zaslonski posnetki volumetričnih podatkov, upodobljenih z različnimi prenosnimi funkcijami, so vidni na sliki 7.3.

7.1 Podpora standardov WebGL in njihovih razširitev

V naslednjih odstavkih bomo analizirali podporo standardov WebGL in za našo aplikacijo bistvenih razširitev. Pri tem se zanašamo na spletno stran `https://webglstats.com`, ki zbira podatke z naprav obiskovalcev nekaterih popularnih spletnih strani.

V času pisanja je standard WebGL 1.0 podprt na praktično vseh napravah (97%, od tega 96% na tablicah, 98% na pametnih telefonih in 98% na namiznih napravah), predvsem v brskalniku Chrome (97% vseh naprav) in na pametnih telefonih z operacijskima sistemoma Android (98%) in iOS (99%). Podpora na namiznih napravah je nekoliko manjša, predvsem v brskalniku Firefox (95%) in na operacijskem sistemu OSX (94%).

Podpora standarda WebGL 2.0 je zaradi dolgotrajnega eksperimentalnega stanja še vedno nižja, čeprav so ga najpopularnejši spletni brskalniki privzeto omogočili že junija 2017. Na pametnih telefonih z operacijskim sistemom iOS je še vedno v preizkusni dobi, na Android telefonih v brskalniku

Chrome pa je podpora 60%. Večina novejših naprav standard podpira. Na namiznih napravah je stanje primerljivo (61% vseh naprav) s podporo v vseh novejših brskalnikih Chrome, Firefox in Opera (skupaj 73%). Brskalnik Safari na platformah macOS in OSX standarda ne podpira, zato je večinoma dostopen le prek brskalnikov Chrome in Firefox (skupaj 81%). Brskalnika Internet Explorer in Edge na namiznih napravah z operacijskim sistemom Windows standarda ne podpirata, zato je tudi tam dostopen večinoma le prek brskalnikov Chrome, Firefox in Opera (skupaj 71%). Podpora na vseh napravah se strmo veča, zato lahko v prihodnosti s postopnim sprejemanjem novih posodobitev operacijskih sistemov in spletnih brskalnikov pričakujemo še precej večjo podporo.

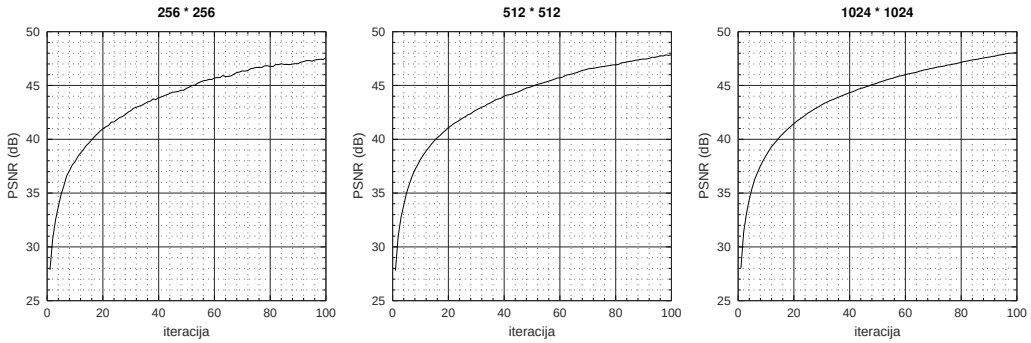
Poleg samega programskega vmesnika WebGL 2.0 se za delovanje naše aplikacije zanašamo še na razširitvi `EXT_color_buffer_float` (za pisanje slik v formatu HDR) in `WEBGL_lose_context`. Slednja je namenjena detekciji izgube konteksta ali programski ukinitvi za namene sprostitve računskih virov grafične strojne opreme. Za delovanje upodabljanja je nujna le prva, druga pa je opcijska in uporabna v primeru, da na napravi izvajamo več instanc brskalnika. Razširitev `WEBGL_lose_context` je dostopna praktično na vseh napravah (100%), medtem ko je podpora razširitve `EXT_color_buffer_float` nekoliko manjša (91% vseh naprav). Na namiznih napravah je podprta praktično povsod (100%), na mobilnih napravah pa v 86% primerov. Presenetljivo vse tablice (100%) z operacijskim sistemom iOS razširitev podpirajo, telefoni z istim operacijskim sistemom pa le v 86% primerov. Za primerjavo navajamo še dejstvo, da je ista razširitev v standardu WebGL 1.0 dostopna le na 30% naprav (17% pametnih telefonov, 10% tablic in 68% namiznih naprav).

Vse interne slike so shranjene v formatu HDR, ki ga standard WebGL 2.0 ponuja v obliki novih teksturnih formatov. Ta funkcionalnost je dostopna v temeljnem standardu WebGL 2.0, medtem ko je bila v standardu WebGL 1.0 na voljo le prek razširitev `OES_texture_float` in `OES_texture_half_float`. Pisanje v takšne okvirje tako v standardu WebGL 1.0 kot WebGL 2.0 ni mogoče v temeljnem profilu, temveč je dostopno v standardu WebGL 1.0 prek

razširitev `WEBGL_color_buffer_float` in `EXT_color_buffer_half_float`, v standardu WebGL 2.0 pa prek razširitve `EXT_color_buffer_float`. Linearna interpolacija tekstur v teh formatih je v obeh standardih dostopna le prek razširitev `OES_texture_float_linear` in `OES_texture_half_float_linear`, a te funkcionalnosti ne potrebujemo, saj se uporablja le pri upodabljanju slikovnega okvirja tonskega slikarja na zaslon, kar pa je omogočeno v temeljnem profilu obeh standardov, saj je slikovni okvir v formatu LDR oz. natančneje `RGBA`.

Merjenje hitrosti izvajanja je zahtevna naloga, saj z naivnimi rešitvami poleg izvajanja operacij vmesnika WebGL merimo še operacije jezika JavaScript in spletnega brskalnika. Uporaba JavaScript funkcij `Date.now()` ali `performance.now()` v kombinaciji z `gl.flush()` ali `gl.finish()` v splošnem ni primerna, saj v teh primerih obnašanja implementacij standard WebGL natančno ne določa. Zanesljiv način za merjenje je dostopen preko razširitev `EXT_disjoint_timer_query` in `EXT_disjoint_timer_query_webgl2`, ki pa na mnogih napravah niso dostopne. Podpora `EXT_disjoint_timer_query` v standardu WebGL 1.0 je skupaj 48%, 61% na namiznih napravah, 45% na pametnih telefonih ter le 16% na tablicah. V standardu WebGL 2.0 je stanje boljše, saj razširitev podpira skupno 81% naprav, 93% namiznih naprav, 73% pametnih telefonov in 58% tablic. V sandardu WebGL 2.0 je razširitev `EXT_disjoint_timer_query_webgl2` podprta na 71% vseh naprav, 71% namiznih naprav, 71% pametnih telefonov in 47% tablic.

Za potrebe neposrednega upodabljanja in odstranjevanja ozkega grla v komunikaciji aplikacija vse podatke predhodno prenese v pomnilnik grafične strojne opreme. Glede na razpoložljiv pomnilnik in ponujenih zmožnosti implementacije WebGL 2.0 lahko upodabljamo volumne različnih dimenzij, s tem da standard OpenGL ES 3.0, na katerem temelji standard WebGL 2.0, zagotavlja vsaj teksture velikosti do $256 \times 256 \times 256$ (iz aplikacije dostopen parameter `MAX_3D_TEXTURE_SIZE`). Statistični podatki kažejo, da praktično vse (100%) implementacije podpirajo teksture do $4096 \times 4096 \times 4096$. Pri tej velikosti je omejitev zaradi implementacije še najmanjša težava, saj se



Slika 7.4: Vrednosti PSNR v odvisnosti od števila iteracij za različne velikosti slikovnega okvirja: 256×256 (levo), 512×512 (sredina) in 1024×1024 (desno) pri volumnu velikosti $128 \times 128 \times 128$.

že veliko prej srečamo s komunikacijskim ozkim grlom in veliko računsko zahtevnostjo metod.

7.2 Konvergenca

Konvergenco metod smo analizirali tako, da smo za različne velikosti slikovnega okvirja izračunali razmerje signal/šum (angl. peak signal-to-noise ratio, PSNR) v odvisnosti od števila iteracij. Odločili smo se za tri velikosti slikovnega okvirja: 256×256, 512×512 in 1024×1024 pri volumnu velikosti $128 \times 128 \times 128$. Za potrebe izračuna vrednosti PSNR smo za signal brez šuma vzeli sliko po 1000 iteracijah. Rezultati so zbrani na sliki 7.4, primer iterativne izboljšave pa je na sliki 7.5. Zaradi velikega števila slikovnih točk so v vseh treh primerih vrednosti PSNR med seboj primerljive. Velikost slikovnega okvirja neposredno vpliva na hitrost posamezne iteracije, toda različnih prostorskih natančnosti slike ne moremo ovrednotiti z vrednostjo PSNR, saj jo računamo neodvisno za vsako slikovno točko. Metoda hitro konvergira, tako da dobimo zelo dobre slike že v 20-30 iteracijah, kot posledica metode Monte Carlo pa so izboljšave kvalitete vedno počasnejše, saj teoretično konvergirajo s hitrostjo $O(n^{-\frac{1}{2}})$.

Slika 7.5: Konvergenca upodabljalnika sevalne enačbe. Prvi zaslonski posnetek je bil zajet v prvi sekundi, zadnji pa deset sekund po začetku upodabljanja.

7.3 Hitrost izvajanja

Izvajanje iteracij v cevovodu upodabljanja znotraj upodabljalnega konteksta temelji na funkciji `requestAnimationFrame`, ki jo spletni brskalniki uskladijo z osveževanjem zaslona in hkrati omejijo njeno frekvenco izvajanja. Običajno to pomeni, da lahko dosežemo le 60 iteracij na sekundo, kar pa izboljšamo z izvajanjem več iteracij znotraj senčilnikov. Ker v sistemu WebGL dlje trajajoče operacije vedno blokirajo glavno nit brskalnika, nam ne pomaga niti izvajanje operacij v spletnem delavcu (angl. web worker), saj moramo kontekst WebGL namreč ustvariti na glavni niti. Kot eksperimentalna funkcionalnost je v novejših brskalnikih na voljo ustvarjanje elementov `<canvas>` v spletnih delavcih (prek vmesnika `OffscreenCanvas`) in s tem ustvarjanje konteksta WebGL, toda vmesnik ni standardiziran, poleg tega pa je večinoma nedostopen na mobilnih napravah. V naši aplikaciji smo poskrbeli za nastavitev hitrosti izvajanja, tako da lahko zmanjšamo število iteracij za generiranje enega okvirja in s tem pridobimo na interaktivnosti. Tako lahko konsistentno dosežemo interaktivne hitrosti (20 iteracij na sekundo in več) tudi na mobilnih napravah.

Hitrost iteracij smo merili na dveh različnih napravah: na prenosnem računalniku Lenovo Thinkpad T460s z integrirano grafično kartico Intel HD graphics 530 in na pametnem telefonu OnePlus 2 z grafično kartico Adreno 430. Merili smo povprečen čas za izračun ene iteracije v odvisnosti od velikosti volumna ($64 \times 64 \times 64$, $128 \times 128 \times 128$ in $256 \times 256 \times 256$ pri velikosti

Naprava	$\mathbf{64^3}$	$\mathbf{128^3}$	$\mathbf{256^3}$	$\mathbf{256^2}$	$\mathbf{512^2}$	$\mathbf{1024^2}$
Pametni telefon	28 ms	65 ms	92 ms	35 ms	65 ms	98 ms
Prenosni računalnik	16 ms	28 ms	45 ms	16 ms	28 ms	55 ms

Tabela 7.1: Povprečen čas izvajanja ene iteracije cevovoda v odvisnosti od velikosti volumna pri slikovnem okvirju velikosti 512×512 (levo) in v odvisnosti od velikosti slikovnega okvirja pri volumnu velikosti $128 \times 128 \times 128$ (desno).

slikovnega okvirja 512×512) in od velikosti slikovnega okvirja (256×256, 512×512 in 1024×1024 pri volumnu velikosti $128 \times 128 \times 128$). Vse meritve smo izvajali v spletnem brskalniku Google Chrome, zaradi nedostopnosti razširitve `EXT_disjoint_timer_query` pa smo se zatekli k uporabi kombinacije funkcij `performance.now()` in `gl.finish()`. Rezultati so zbrani v tabeli 7.1, iz katere je razvidno, da je metoda dovolj hitra za interaktivno uporabo. Pri prenosnem računalniku smo za dovolj majhen volumen in slikovni okvir naleteli na omejitve spletnega brskalnika, zato je najvišja hitrost omejena na 16 ms oz. 60 iteracij na sekundo.

Poglavje 8

Sklepi

V tem delu smo predstavili sodobne metode za hitro, interaktivno in realnočasovno upodabljanje volumetričnih podatkov. Temeljijo na splošnem fizikalnem modelu, kar omogoča izris kvalitetnih fotorealističnih slik. Izdelali smo tudi predstavitveno spletno aplikacijo s prilagojenim cevovodnim modelom za podporo opisanim metodam. Aplikacija je platformno agnostična, zato jo je moč izvajati na široki paleti namiznih in mobilnih naprav, hkrati pa je zaradi svoje strukture enostavno razširljiva, kar omogoča nadaljnje izboljšave.

Metode upodabljanja so izdelane po modelu sevalne enačbe, ki predstavlja fizikalno podlago za naše delo. Rešitev sevalne enačbe v splošnem ni analitična, zato jo rešujemo z numeričnimi metodami. Kvadraturne metode v našem primeru niso dovolj hitre, saj zaradi velike dimenzionalnosti problema trpijo za eksplozijo računske zahtevnosti, zato smo v naši implementaciji uporabili iterativne metode Monte Carlo. Posledično je izris grobega približka končne slike izredno hiter, kar omogoča visoko interaktivnost, hkrati pa zagotavlja konvergenco k pravilni rešitvi problema. Narava metode Monte Carlo nam poleg tega omogoča enostavno vključitev vizualnih efektov in svetlobnih pojavov zaradi nepopolnosti kamere. Hitrost konvergence metod Monte Carlo smo izboljšali z uporabo prioritetnega vzorčenja.

Metode upodabljanja smo testirali na dveh različnih napravah: na preno-

snem računalniku in pametnem telefonu. Rezultati kažejo, da so vse metode primerne za interaktivno upodabljanje v realnem času, saj konsistentno dosegajo hitrosti več kot 10 iteracij na sekundo. Meritve razmerja signal/šum kažejo, da metoda v 20-30 iteracijah konvergira do prepoznavne slike, nato pa so izboljšave kvalitete počasnejše. Posledično dosegamo interaktivne hitrosti, a na kvaliteti ne izgubimo, saj jo povečujemo z nadaljnjimi iteracijami.

Cilje, opisane v uvodnem poglavju tega dela, smo dosegli, s tem pa postavili nov mejnik na področju vizualizacije volumetričnih podatkov s spletnimi tehnologijami. Kljub temu je prostora za izboljšave nemalo. Izboljšati je moč hitrost konvergence metod s posebnimi podatkovnimi strukturami s podporo grafične strojne opreme. Večje količine podatkov lahko upodabljamo s prilagojenim sistemom za upravljanje s pomnilnikom, saj v trenutni implementaciji vse podatke zapišemo v omejen grafični pomnilnik. V aplikacijo lahko integriramo upodabljanje poligonskih modelov in s tem močno razširimo praktično uporabnost aplikacije. Prenove potreben je tudi uporabniški vmesnik, ki sicer v tem delu ni bil bistvenega pomena. Upodabljanje v večjem obsegu (z več uporabniki in večjo količino podatkov) bi lahko podkrepili z gručo porazdeljenih upodabljalnikov ali pa upodabljanje preselili na namenski strežnik. S številnimi idejami v mislih upamo, da bosta znanost in industrija pozitivno sprejeli naše delo in s tem podkrepili nadaljnji razvoj področja.

Literatura

[1] S. Roth, Ray casting for modeling solids, Computer Graphics and Image Processing 18 (2) (1982) 109–144.

[2] R. Drebin, L. Carpenter, P. Hanrahan, Volume rendering, ACM SIGGRAPH Computer Graphics 22 (4) (1988) 65–74.

[3] M. Levoy, Display of surfaces from volume data, IEEE Computer Graphics and Applications 8 (3) (1988) 22–37.

[4] M. Levoy, A hybrid ray tracer for rendering polygon and volume data, IEEE Computer Graphics and Applications 10 (2) (1990) 33–40.

[5] J. C. Hart, Ray tracing implicit surfaces, in: ACM SIGGRAPH, Vol. 1, 1993.

[6] J. Hart, Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces, The Visual Computer 12 (10) (1996) 527–545.

[7] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, ACM SIGGRAPH Computer Graphics 21 (4) (1987) 163–169.

[8] P. Lacroute, M. Levoy, Fast volume rendering using a shear-warp factorization of the viewing transformation, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, 1994, pp. 451–458.

[9] L. A. Westover, Splatting: a parallel, feed-forward volume rendering algorithm, Ph.D. thesis, University of North Carolina (1991).

[10] N. Max, Optical models for direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 1 (2) (1995) 99–108.

[11] Ž. Lesar, Vizualizacija medicinskih volumetričnih podatkov v realnem času, Ph.D. thesis, University of Ljubljana (2014).

[12] Ž. Lesar, Real-time ray casting of volumetric data, in: International Electrotechnical and Computer Science Conference, Portorož, 2014, pp. 199–200.

[13] Ž. Lesar, Real-time ray casting of volumetric data, in: IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON), IEEE, 2015, pp. 1–6.

[14] Ž. Lesar, C. Bohak, M. Marolt, Evaluation of angiogram visualization methods for fast and reliable aneurysm diagnosis, in: C. R. Mello-Thoms, M. A. Kupinski (Eds.), SPIE Medical Imaging, Vol. 9416, 2015, p. 94161D.

[15] U. Behrens, R. Ratering, Adding shadows to a texture-based volume renderer, in: Proceedings of the 1998 IEEE symposium on Volume visualization, ACM Press, New York, New York, USA, 1998, pp. 39–46.

[16] T. Lokovic, E. Veach, Deep shadow maps, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00, ACM Press, New York, New York, USA, 2000, pp. 385–392.

[17] M. Hadwiger, A. Kratz, C. Sigg, K. Bühler, GPU-accelerated deep shadow maps for direct volume rendering, in: Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware - GH '06, ACM Press, New York, New York, USA, 2006, p. 49.

[18] M. Ament, F. Sadlo, C. Dachsbacher, D. Weiskopf, Low-pass filtered volumetric shadows, IEEE Transactions on Visualization and Computer Graphics 20 (12).

[19] A. Kratz, Advanced illumination techniques for GPU-based direct volume rendering, Ph.D. thesis, University of Koblenz-Landau (2006).

[20] M. Hadwiger, P. Ljung, C. Rezk Salama, T. Ropinski, Advanced illumination techniques for GPU-based volume raycasting, in: ACM SIGGRAPH ASIA, 2008, pp. 1–166.

[21] D. Jönsson, E. Sundén, A. Ynnerman, T. Ropinski, A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering, Computer Graphics Forum 33 (1) (2014) 27–51.

[22] J. T. Kajiya, The rendering equation, ACM SIGGRAPH Computer Graphics 20 (4) (1986) 143–150.

[23] M. Levoy, Volume rendering by adaptive refinement, The Visual Computer 6 (1) (1990) 2–7.

[24] E. P. Lafortune, Y. D. Willems, Bi-Directional Path Tracing, in: Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93, 1993, pp. 145–153.

[25] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of State Calculations by Fast Computing Machines, The Journal of Chemical Physics 21 (6) (1953) 1087.

[26] W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1) (1970) 97–109.

[27] E. Veach, L. J. Guibas, Metropolis light transport, in: Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97, ACM Press, New York, New York, USA, 1997, pp. 65–76.

[28] M. Pharr, J. Wenzel, G. Humphreys, Physically based rendering, 3rd Edition, Morgan Kaufmann, 2016.

[29] P. H. Christensen, W. Jarosz, The Path to Path-Traced Movies, Foundations and Trends in Computer Graphics and Vision 10 (2) (2016) 103–175.

[30] E. Lafortune, Y. Willems, Rendering Participating Media with Bidirectional Path Tracing, in: Eurographics, 1996, pp. 91–100.

[31] E. Veach, Robust Monte Carlo methods for light transport simulation, Ph.D. thesis, Stanford university (dec 1997).

[32] M. Pauly, Robust Monte Carlo methods for photorealistic rendering of volumetric effects, Ph.D. thesis, Kaiserslautern university (1999).

[33] W. Jarosz, Efficient Monte Carlo methods for light transport in scattering media, Ph.D. thesis, University of California, San Diego (2008).

[34] J. Křivánek, I. Georgiev, T. Hachisuka, P. Vévoda, Unifying Points, Beams, and Paths in Volumetric Light Transport Simulation, ACM SIGGRAPH 33 (July).

[35] T. Kroes, F. H. Post, C. P. Botha, Exposure Render: An Interactive Photo-Realistic Volume Rendering Framework, PLoS ONE 7 (7) (2012) e38586.

[36] T. Kroes, F. H. Post, C. P. Botha, Interactive direct volume rendering with physically-based lighting, Eurographics vi (2012) 1–10.

[37] J. Fong, M. Wrenninge, C. Kulla, R. Habel, Production volume rendering, in: ACM SIGGRAPH, ACM Press, New York, New York, USA, 2017, pp. 1–79.

[38] J. A. Bucklew, Introduction to Rare Event Simulation, Springer Series in Statistics, Springer, New York, NY, 2004.

[39] E. Veach, L. J. Guibas, Optimally combining sampling techniques for Monte Carlo rendering, in: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95, ACM Press, New York, New York, USA, 1995, pp. 419–428.

[40] F. Nicodemus, Directional Reflectance and Emissivity of an Opaque Surface, Applied Optics 4 (7) (1965) 767–775.

[41] L. C. Henyey, J. L. Greenstein, Diffuse radiation in the Galaxy, The Astrophysical Journal 93 (1941) 70–83.

[42] D. Toublanc, Henyey–Greenstein and Mie phase functions in Monte Carlo radiative transfer computations, Applied Optics 35 (18) (1996) 3270.

[43] E. R. Woodcock, T. Murphy, P. J. Hemmings, T. C. Longworth, Techniques used in the GEM code for Monte Carlo neutronics calculation, Tech. rep. (1965).

[44] Y. Yue, K. Iwasaki, B.-Y. Chen, Y. Dobashi, T. Nishita, Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media, ACM Transactions on Graphics 29 (6) (2010) 1.

[45] L. Szirmay-Kalos, B. Tóth, M. Magdics, Free Path Sampling in High Resolution Inhomogeneous Participating Media, Computer Graphics Forum 30 (1) (2011) 85–97.

[46] J. Novák, A. Selle, W. Jarosz, Residual ratio tracking for estimating attenuation in participating media, ACM Transactions on Graphics 33 (6) (2014) 1–11.

[47] C. Kolb, D. Mitchell, P. Hanrahan, A Realistic Camera Model for Computer Graphics, in: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, 1995, pp. 317–324.

[48] B. A. Barsky, D. R. Horn, S. A. Klein, J. A. Pang, M. Yu, Camera Models and Optical Systems Used in Computer Graphics: Part II, Image-Based Techniques, in: Computational Science and Its Applications, 2003, pp. 256–265.

[49] K. Schwenk, A. Kuijper, J. Behr, D. W. Fellner, Practical Noise Reduction for Progressive Stochastic Ray Tracing with Perceptual Control, IEEE Computer Graphics and Applications 32 (6) (2012) 46–55.

[50] P. Bauszat, M. Eisemann, E. Eisemann, M. Magnor, General and Robust Error Estimation and Reconstruction for Monte Carlo Rendering, Computer Graphics Forum 34 (2) (2015) 597–608.

[51] B. Bitterli, F. Rousselle, B. Moon, J. A. Iglesias-Guiti??n, D. Adler, K. Mitchell, W. Jarosz, J. Nov??k, Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings, Computer Graphics Forum 35 (4).

[52] K. Engel, T. Ertl, Interactive high-quality volume rendering with flexible consumer graphics hardware, in: Eurographics, Eurographics Association, 2002.

[53] P. Bhanirantka, Y. Demange, OpenGL volumizer: a toolkit for high quality volume rendering of large data sets, in: Symposium on Volume Visualization and Graphics, 2002. Proceedings. IEEE / ACM SIGGRAPH, IEEE, 2002, pp. 45–53.

[54] J. Bikker, J. van Schijndel, The Brigade Renderer: A Path Tracer for Real-Time Games, International Journal of Computer Games Technology (2013) 1–14.

[55] S. G. Parker, A. Robison, M. Stich, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, OptiX, in: ACM SIGGRAPH 2010 papers on - SIGGRAPH '10, ACM Press, New York, New York, USA, 2010, p. 1.

[56] S. G. Parker, G. Humphreys, M. McGuire, M. Stich, H. Friedrich, D. Luebke, K. Morley, J. Bigler, J. Hoberock, D. McAllister, A. Robison, A. Dietrich, GPU ray tracing, Communications of the ACM 56 (5) (2013) 93.

[57] W. Schroeder, K. Martin, B. Lorensen, The Visualization Toolkit, 4th Edition, Kitware, 2006.

[58] C. Gribble, J. Fisher, D. Eby, E. Quigley, G. Ludwig, Ray tracing visualization toolkit, in: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games - I3D '12, ACM Press, New York, New York, USA, 2012, p. 71.

[59] E. Smistad, M. Bozorgi, F. Lindseth, FAST: framework for heterogeneous medical image computing and visualization, International Journal of Computer Assisted Radiology and Surgery 10 (11) (2015) 1811–1822.

[60] M. D. Hanwell, K. M. Martin, A. Chaudhary, L. S. Avila, The Visualization Toolkit (VTK): Rewriting the rendering code for modern graphics cards, SoftwareX 1-2 (2015) 9–12.

[61] T. Fogal, A. Schiewe, J. Kruger, An analysis of scalable GPU-based ray-guided volume rendering, in: IEEE Symposium on Large Data Analysis and Visualization, 2013.

[62] T. Davidovič, J. Křivánek, M. Hašan, P. Slusallek, Progressive Light Transport Simulation on the GPU, ACM Transactions on Graphics 33 (3) (2014) 1–19. `arXiv:arXiv:1204.6216v2`.

[63] J. Beyer, M. Hadwiger, H. Pfister, State-of-the-Art in GPU-Based Large-Scale Volume Visualization, Computer Graphics Forum 34 (8) (2015) 13–37.

[64] H.-Y. Peng, K. Wong, K. Williams, Efficient Volume Rendering in CUDA Path Tracer, University of Southern California, 2014.

[65] L. Campoalegre, P. Brunet, I. Navazo, Interactive visualization of medical volume models in mobile devices, Personal and Ubiquitous Computing 17 (7) (2013) 1503–1514.

[66] H. Zhou, H. Qu, Y. Wu, M.-y. Chan, Volume visualization on mobile devices, in: National Taiwan University Press: Taipe, 2006, pp. 76–84.

[67] M. Moser, D. Weiskopf, Interactive Volume Rendering on Mobile Devices, in: Workshop on Vision, Modelling, and Visualization VMV '08, Vol. 41, 2008, pp. 217–226.

[68] M. B. Rodríguez, P. P. V. Alcocer, Practical Volume Rendering in Mobile Devices, 2012, pp. 708–718.

[69] A. Schiewe, M. Anstoots, J. Krüger, State of the Art in Mobile Volume Rendering on iOS Devices, in: E. Bertini, J. Kennedy, E. Puppo (Eds.), Eurographics Conference on Visualization (EuroVis) - Short Papers, The Eurographics Association, 2015.

[70] T. Hachisuka, Implementing a Photorealistic Rendering System using GLSL, arXiv (2015) 1–4arXiv:1505.06022.

[71] J. M. Noguera, J. R. Jimenez, Mobile Volume Rendering: Past, Present and Future, IEEE Transactions on Visualization and Computer Graphics 22 (2) (2016) 1164–1178.

[72] D. Jackson, J. Gilbert, WebGL 1.0 Specification.
URL https://www.khronos.org/registry/webgl/specs/latest/1.0

[73] A. Munshi, J. Leech, OpenGL ES 2.0 Specification.
URL https://www.khronos.org/registry/OpenGL/specs/es/2.0

[74] J. Congote, A. Segura, L. Kabongo, A. Moreno, J. Posada, O. Ruiz, Interactive visualization of volumetric data with WebGL in real-time,

in: International Conference on 3D Web Technology, ACM Press, New York, New York, USA, 2011, p. 137.

[75] J. Noguera, J.-r. Jiménez, Visualization of very large 3D volumes on mobile devices and WebGL, WSCG Communication Proceedings (2012) 105–112.

[76] M. M. Mobeen, W. M. Chiew, L. Feng, On-Site Volume Rendering with GPU-Enabled Devices, Wireless Personal Communications 76 (4) (2014) 795–812.

[77] M. M. Mobeen, L. Feng, High-Performance Volume Rendering on the Ubiquitous WebGL Platform, in: 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, IEEE, 2012, pp. 381–388.

[78] M. M. Mobeen, L. Feng, Ubiquitous Medical Volume Rendering on Mobile Devices, International Conference on Information Society (i-Society 2012) (2012) 93–98.

[79] D. Jackson, J. Gilbert, WebGL 2.0 Specification.
URL `https://www.khronos.org/registry/webgl/specs/latest/2.0`

[80] J. Leech, B. Lipchak, OpenGL ES 3.0 Specification.
URL `https://www.khronos.org/registry/OpenGL/specs/es/3.0`

[81] G. Bizjak, M. B. Kobav, M. Prelovšek, Razsvetljava, Fakulteta za elektrotehniko, Ljubljana, 2013.

[82] J. Drnovšek, J. Bojkovski, G. Geršak, I. Pušnik, D. Hudoklin, Metrologija, Fakulteta za elektrotehniko, Laboratorij za metrologijo in kakovost, Ljubljana, 2012.

[83] S. Chandrasekhar, Radiative transfer, Oxford, 1950.

[84] C. Schlick, An Inexpensive BRDF Model for Physically-based Rendering, Computer Graphics Forum 13 (3) (1994) 233–246.

[85] J. R. Frisvad, Importance sampling the Rayleigh phase function, Journal of the Optical Society of America A 28 (12) (2011) 2436.

[86] T. Nishita, Y. Miyawaki, E. Nakamae, A shading model for atmospheric scattering considering luminous intensity distribution of light sources, ACM SIGGRAPH Computer Graphics 21 (4) (1987) 303–310.

[87] D. Petersen, D. Middleton, Sampling and reconstruction of wave-number-limited functions in N-dimensional euclidean spaces, Information and Control 5 (4) (1962) 279–323.

[88] J. Kniss, G. Kindlmann, C. Hansen, Multidimensional transfer functions for interactive volume rendering, IEEE Transactions on Visualization and Computer Graphics 8 (3) (2002) 270–285.

[89] E. Reinhard, M. Stark, P. Shirley, J. Ferwerda, Photographic tone reproduction for digital images, ACM Transactions on Graphics 21 (3).

[90] F. Drago, K. Myszkowski, T. Annen, N. Chiba, Adaptive Logarithmic Mapping For Displaying High Contrast Scenes, Computer Graphics Forum 22 (3) (2003) 419–426.

[91] J. Hable, Uncharted 2: HDR Lighting, in: Game Developers Conference, 2010, p. 56.