Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žiga Kokelj

Implementacija navidezne resničnosti v spletno vizualizacijsko ogrodje Med3D

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matija Marolt
Somentor: as. dr. Ciril Bohak

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu preučite načine za implementacijo navidezne resničnosti v spletne aplikacije in implementirajte razširitev spletnega vizualizacijskega ogrodja Med3D s prikazom scene v navidezni resničnosti. Pri tem poskrbite tako za prikaz kot za navigiranje po sceni z ustreznimi upravljalniki. Poleg tega v ogrodje implementirajte tudi vizualizacijski algoritem metanja žarka in ga evalvirajte.

Najprej bi se zahvalil mentorju izr. prof. dr. Matiji Maroltu in somentorju as. dr. Cirilu Bohaku za usmerjanje ter veliko podporo pri izdelavi diplomske naloge. Zahvalil bi se tudi Primožu Lavriču za pomoč pri spoznavanju vizualizacijskega ogrodja in Anji Ajdovec za odpravo pravopisnih napak. Poleg zgoraj omenjenih bi se zahvalil tudi vsem prijateljem, sošolcem, sostanovalcem v študentskem domu in ostalim, ki so mi stali ob strani tekom študija. Posebej pa bi se zahvalil družini, ki me je vedno podpirala, spodbujala in mi omogočila študij.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
2D	two-dimensional	dvodimenzionalen
3D	three-dimensional	tridimenzionalen
FPS	frames per second	število slik na sekundo
GLSL	OpenGL shading language	jezik za pisanje OpenGL senčilnikov
MRI	magnetic resonance imaging	slikanje z magnetno resonanco
RGBA	red, green, blue, alpha	rdeča, zelena, modra, prosojnost
VR	virtual reality	navidezna resničnost

Povzetek

Naslov: Implementacija navidezne resničnosti v spletno vizualizacijsko ogrodje Med3D

Avtor: Žiga Kokelj

V tem diplomskem delu predstavljamo razširitev spletnega vizualizacijskega ogrodja Med3D, ki uporabniku omogoča ogled vizualizacij v navidezni resničnosti. Poleg pogleda v navidezni resničnosti smo poskrbeli tudi za upravljanje scene s pomočjo upravljalnikov, priloženih sistemu HTC Vive. S temi upravljalniki lahko objekte premikamo in rotiramo v vseh treh smereh ter si tako zagotovimo ogled tistega dela podatkov, ki nas najbolj zanima. Poleg omenjenih dveh razširitev smo implementirali tudi metodo neposrednega upodabljanja volumetričnih podatkov z metanjem žarkov. Uporabnik pa ima ob tem možnost prilagajanja parametrov, ki določajo barvo, število vzorčenj in korekcijski faktor izrisa, saj se vhodni podatki lahko med seboj precej razlikujejo. Na ta način lahko zagotovimo njihovo ustrezno vizualizacijo. Za konec smo izvedli še teste, v katerih smo ugotavljali vpliv števila vzorčenj in korekcijskega faktorja na hitrost izrisa.

Ključne besede: navidezna resničnost, vizualizacija medicinskih podatkov, metoda metanja žarkov, spletno vizualizacijsko ogrodje

Abstract

Title: Implementation of virtual reality into the Med3D web visualisation framework

Author: Žiga Kokelj

In this diploma thesis, we present the extension of web-based visualization framework Med3D, which allows users to view visualizations in virtual reality. We also implemented controlling visualized objects in virtual reality with HTC Vive controllers. It allows us to translate and rotate objects in all three directions, to ensure that the user gets the angle of view that he wants. In addition to extensions described above, we also implemented a volume ray casting method for visualizing volumetric data. Users can set a color, number of samples per ray and correction factor. Those settings are important for visualization results since input data can vary greatly and hardcoded parameters cannot provide the desired visualization results. We also tested the speed of our implementation of the algorithm (in FPS - frames per second) for different values of the number of samples and correction factor.

Keywords: virtual reality, visualization of medical data, volume ray casting, web based visualization framework

Poglavje 1

Uvod

Naprave za zajem 3D medicinskih slik so v zadnjih desetletjih doživele velik napredek [20, 12, 3]. Z njimi lahko zajemamo vse bolj natančne posnetke notranjosti teles, ki lahko zdravnikom pomagajo do lažjih in pravilnejših odločitev glede morebitnega nadaljnjega zdravljenja. Za sprejemanje odločitev na podlagi podatkov pa ni dovolj le kakovosten zajem, ampak je bistvena predvsem vizualizacija zajetih podatkov. Medicinske naprave za skeniranje tkiv v večini primerov zajamejo podatke v obliki tridimenzionalnega skalarnega polja. Takšni podatki pa sami po sebi človeku niso enostavno berljivi. Ravno zaradi tega sta ključnega pomena njihova obdelava in prikaz, saj le tako lahko izkoristimo celoten potencial, ki ga prinašajo sodobne tehnologije za zajem. Poleg vizualizacije na klasičnih dvodimenzionalnih zaslonih se v zadnjih letih vse bolj uveljavlja tudi vizualizacija v navidezni resničnosti, ki nam daje občutek, da so predmeti fizično prisotni okoli nas in jih vidimo v treh dimenzijah [42]. K temu je pripomogel pospešen tehnološki razvoj, saj strojna oprema, ki omogoča vizualizacijo v navidezni resničnosti, postaja vse bolj zmogljiva in vedno bolj cenovno dostopna širši populaciji.

Eden od programov, ki omogočajo vizualizacijo medicinskih podatkov, je tudi odprtokodno spletno vizualizacijsko ogrodje Med3D, razvito na fakulteti za računalništvo in informatiko. Do sedaj je ogrodje ponujalo vizualizacijo medicinskih podatkov prek spleta na klasičnem računalniškem zaslonu. To

smo želeli izboljšati in uporabnikom ponuditi možnost ogleda podatkov v navidezni resničnosti. Tako jim ponudimo priložnost za večjo poglobitev v vizualizirane podatke in posledično boljše razumevanje njihovega pomena.

1.1 Cilji

Cilji diplomskega dela so:

- integracija navidezne resničnosti v vizualizacijsko ogrodje Med3D[1];
- integracija interakcije z upravljalniki, ki omogočajo interakcijo z objekti v navidezni resničnosti;
- integracija preprostega prehoda med običajnim prikazom in prikazom v navidezni resničnosti;
- implementacija tehnike neposrednega upodabljanja z metanjem žarkov.

1.2 Struktura dela

Delo je razdeljeno na šest poglavij. Uvodu sledi pregled področja v poglavju 2, kjer smo bralcu predstavili osnove vizualizacije in navidezne resničnosti, sisteme za prikaz navidezne resničnosti, tehniko metanja žarkov ter nekatere uporabljene tehnologije. V 3. poglavju sledi predstavitev vizualizacijskega ogrodja Med3D. 4. poglavje opisuje našo implementacijo navidezne resničnosti, implementacijo uporabe upravljalnikov za interakcijo in tehnike metanja žarkov v samo vizualizacijsko ogrodje. V 5. poglavju smo ovrednotili opravljeno delo in predstavili rezultate. Delo se zaključi s 6. poglavjem, ki vsebuje zaključek in opis možnosti nadaljnjega razvoja.

[1] `https://github.com/UL-FRI-LGM/Med3D`

Poglavje 2

Pregled področja

Vizualizacija podatkov ima zelo dolgo zgodovino, ki sega daleč pred čas prvih računalnikov. Prve vizualizacije so bile narejene ročno in so služile za prikaz položajev nebesnih teles ter njihovega gibanja [14]. Od takrat so načini in natančnost prikaza podatkov krepko napredovali. Pravi razcvet pa zaradi razvoja računalništva doživljajo v zadnjih desetletjih, saj jim ta omogoča prikaz zelo kompleksnih in obsežnih podatkov na načine, ki prej niso bili mogoči [10]. Prav razvoj vizualizacije – podpodročja računalniške grafike – pa je bistveno pripomogel k razvoju navidezne resničnosti (angl. virtual reality - VR), saj navidezno okolje v večini primerov simuliramo s pomočjo računalniške grafike.

2.1 Uvod v vizualizacijo

Vizualizacija podatkov je postala nepogrešljivo orodje v znanosti, medicini in na mnogih področjih inženirstva. Njena uporabna vrednost se povečuje z razvojem tehnologij, ki omogočajo zajem velikih količin podatkov. Ti zaradi svoje obsežnosti postanejo uporabni takrat, ko jih uspemo ustrezno vizualizirati in jih posledično lažje interpretiramo [24]. V mnogih znanstvenih vedah znanstvenikom iz ogromnih količin surovih podatkov brez ustrezne vizualizacije ne bi uspelo odkriti pomembnih vzorcev, ki se v njih pojavljalo in lahko

vodijo do novih odkritij. Vizualizacija je izjemno koristna tudi pri izvajanju raznih simulacij, saj lahko določene stvari simuliramo navidezno, kar zmanjša stroške fizičnih poizkusov in pomaga pri boljšem načrtovanju produktov ter urjenju ljudi [32]. Vedno večjo vlogo ima tudi v filmski in multimedijski industriji, saj lahko z njeno pomočjo ustvarijo različne efekte, ki so bili pred tem zelo dragi ali pa celo nemogoči za izvedbo [1].

V računalniški grafiki lahko 3D predmete predstavimo na različne načine. Predstavitve predmetov v grobem delimo na ploskve, polna telesa, predstavitev z neposredno zajetimi podatki in predstavitev z višjenivojskimi strukturami [48]. Za vsakega od naštetih načinov predstavitve 3D objektov obstajajo različni načini vizualizacije, ki pa imajo svoje prednosti in slabosti.

Najbolj se je uveljavila tehnika upodabljanja 3D objektov, sestavljenih iz poligonskih mrež. V tem načinu so objekti v večini primerov sestavljeni iz trikotnikov, saj imajo le-ti lastnost, da so vedno ravninski in konveksni. Sodobna računalniška grafika se je zato usmerila v razvoj hitrega izrisovanja velikega števila majhnih trikotnikov. To nam omogočajo grafične kartice, ki so narejene prav s tem namenom. Hkrati nam omogočajo tudi vzporedno računanje, kar postopek še dodatno pohitri. Poleg 3D objektov pa za končno sliko potrebujemo tudi luči, ki nam dajejo informacije o osvetlitvi, in kamero, ki določa od kod, v kateri smeri ter v kakšnem formatu bomo zajemali sliko iz scene. Za pretvorbo 3D scene iz računalniškega zapisa v bitno sliko, ki jo vidimo na zaslonu, skrbi grafični cevovod [30]. Sestavljen je iz več faz in je tipično (zaradi hitrosti) implementiran v strojni opremi (na grafični kartici). Določene faze so implementirane strojno, druge pa programsko s pomočjo posebnih programov, imenovanih senčilniki [4].

Čeprav je upodabljanje objektov s poligonskimi mrežami najbolj razširjeno in ima zaradi razvoja grafičnih kartic zelo dobro podporo strojne opreme, pa obstajajo primeri uporabe, ko tak način prikaza ni optimalen in je potrebno izbrati enega izmed ostalih. Lep primer je vizualizacija volumetričnih podatkov, ki je opisana v naslednjem poglavju. Tu predstavitev s poligonskimi mrežami običajno ni najboljša izbira.

2.1.1 Vizualizacija volumetričnih podatkov

Eden od tipov predstavitve predmetov v računalniku je tudi predstavitev z neposredno zajetimi volumetričnimi podatki [48]. Ta način je zelo primeren za prikaz medicinskih podatkov, saj s sodobnimi napravami običajno zajemamo podatke, ki jih je nato treba še ustrezno vizualizirati. Pristope za vizualizacijo takih podatkov delimo na pristope posrednega upodabljanja in pristope neposrednega upodabljanja, ki so podrobneje predstavljeni v naslednjih dveh podpoglavjih.

2.1.2 Pristopi posrednega upodabljanja

Metode posredne vizualizacije volumetričnih podatkov le-te pretvorijo v drugo obliko predstavitve. Najpogosteje se uporablja pretvorba v model poligonskih mrež. V tem primeru volumetrične podatke najprej pretvorimo v poligonske mreže, slednje pa potem vizualiziramo z že predstavljenim pristopom. Za pretvorbo iz volumetričnih podatkov v poligonske mreže imamo na voljo več algoritmov. Med najbolj znanimi sta metodi marching cubes [29] in Bloomenthalov poligonizator [5]. Metoda marching cubes je v praksi najbolj uporabna, saj zanjo obstajajo zelo hitre implementacije. To metodo je v sklopu svoje diplomske naloge opisal Primož Lavrič in jo tudi vgradil v spletno vizualizacijsko ogrodje Med3D [27].

2.1.3 Pristopi neposrednega upodabljanja

Posredno upodabljanje ima kljub številnim dobrim lastnostim tudi svoje slabosti. Natančnost takega upodabljanja je odvisna od števila uporabljenih poligonov (najpogosteje trikotnikov), ki pri zelo kompleksnih modelih hitro narastejo preko obvladljivih meja. Poleg tega s tem pristopom vizualiziramo le površino objekta, kar pa pri volumetričnih podatkih ni optimalno, saj s tem izgubimo del informacij, vsebovanih v zajetih podatkih. V takih primerih nam koristijo algoritmi za neposredno upodabljanje, ki volumetričnih podatkov pred vizualizacijo ne pretvarjajo v geometrijsko mrežo, ampak iz-

vajajo vizualizacijo nad zajetimi podatki v njihovi originalni obliki (tridimenzionalno skalarno polje). Osnovna metoda neposrednega upodabljanja je tehnika metanja žarkov (angl. Ray casting) [13], ki sem jo implementiral v sklopu te diplomske naloge. Podrobneje sem jo predstavil v podpoglavju 2.1.5. Med metodami neposrednega upodabljanja poznamo tudi metode: splatting [46], algoritem shear-warp [25] in vizualizacijo s tridimenzionalnimi teksturami [43].

2.1.4 Volumetrični medicinski podatki

Razvoj naprav za zajem medicinskih podatkov in razvoj računalniške grafike sta pripomogla k temu, da vizualizacija medicinskih podatkov postaja vse močnejše orodje zdravnikov. Natančen zajem podatkov in dobra vizualizacija postajata vse pomembnejša pri hitrejšem in natančnejšem postavljanju diagnoz. Prav tako pa omogočata boljše načrtovanje in izvedbo kirurških posegov [11].

Za zajemanje notranje strukture telesa se uporabljajo različne tehnike radiološkega slikanja. Med najbolj pogoste sodijo računska tomografija (angl. computed comography - CT) [12], magnetna resonanca (angl. magnetic resonance imaging - MRI) [3], pozitronska emisijska tomografija (ang. positron emission tomography - PET) [35] in 3D ultrazvok (angl. ultrasound - US) [22]. Vsem zgoraj omenjenim metodam je skupno to, da v večini primerov zajamejo podatke v obliki 3D skalarnega polja. Z razvojem naprav in večanjem natančnosti se zelo hitro povečuje tudi velikost zajetih podatkov, ki v nekaterih primerih znaša tudi več GB. Za njihovo vizualizacijo potrebujemo zmogljive sisteme, ki so sposobni hitrega procesiranja tovrstnih podatkov. Težava takšnih sistemov je, da so večinoma stacionarni in razmeroma dragi. To zdravnikom otežuje izmenjavo mnenj na daljavo, kar negativno vpliva na obravnavo pacientov. Za njihovo vizualizacijo imamo na voljo posredne in neposredne pristope vizualizacije, ki so opisani v prejšnjih dveh podpoglavjih.

2.1.5 Tehnika metanja žarkov

Tehnika metanja žarkov (angl. volume ray casting) je zaradi svoje preprostosti in natančnosti ena najpogosteje uporabljenih tehnik neposrednega upodabljanja skalarnih polj. Metoda temelji na *metanju* navideznih žarkov iz izhodišča (kamere oz. pogleda) skozi skalarno polje. Koraki algoritma so prikazani na sliki 2.1. Osnovni algoritem je leta 1982 predstavil Scott Roth [37]. Algoritem so v naslednjih letih še dodatno izboljšali in razširili možnosti uporabe [28].

Osnovna oblika algoritma je sestavljena iz naslednjih štirih stopenj:

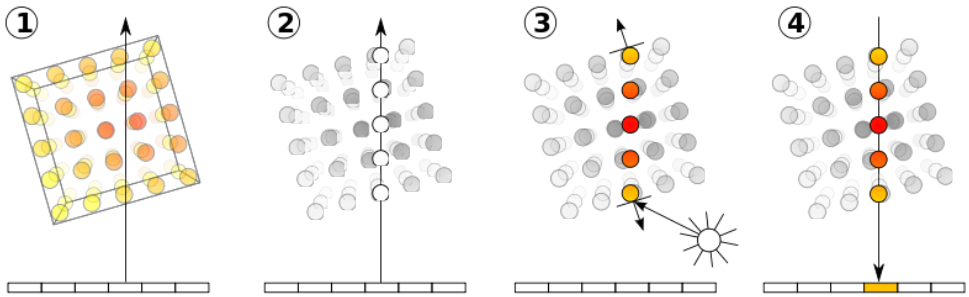
- *Metanje žarka (angl. Ray casting)* - V prvem koraku za vsako slikovno točko (angl. pixel) na končni sliki iz izhodišča kamere pošljemo žarek skozi skalarno polje. Slikovna točka predstavlja najmanjšo enoto slike, ki se jo lahko izriše ali pa prebere. Taka točka nima definirane velikosti ali oblike, temveč le barvo in intenziteto. Objekt, predstavljen v taki obliki, običajno obdamo z enostavnim geometrijskim objektom, kar nam koristi pri ugotavljanju tega, kdaj moramo nehati slediti žarku oz. kdaj se le ta zaključi.

- *Vzorčenje (angl. Sampling)* - Vzdolž dela žarka, ki leži znotraj skalarnega polja, naredimo vzorčenje na enakomernih razdaljah. Ker točke, kjer vzorčimo, lahko ležijo med voksli, moramo za izračun vrednosti v posamezni točki uporabiti interpolacijo (pogosto v ta namen uporabljamo tri-linearno interpolacijo). Voksel v tridimenzionalni grafiki predstavlja najmanjši del, ki ga lahko prikažemo. V tem pogledu je zelo podoben slikovnemu elementu - pikslu, le da slednji obstaja v dveh dimenzijah.

- *Senčenje (angl. Shading)* - Za vsako točko vzorčenja s pomočjo prenosne funkcije (angl. transfer function) pridobimo barvno vrednost in vrednost osvetlitve. Prenosna funkcija se uporablja za določanje RGBA vrednosti posameznim točkam. S spreminjanjem vrednosti prenosne

funkcije lahko vizualiziramo le določena tkiva, druga pa postanejo prosojna.

- *Akumulacija (angl. Compositing)* - Ko zaključimo s prejšnjim korakom, vrednosti vzdolž žarka združimo in tako dobimo končno barvo piksla, ki ga s tem žarkom obravnavamo.



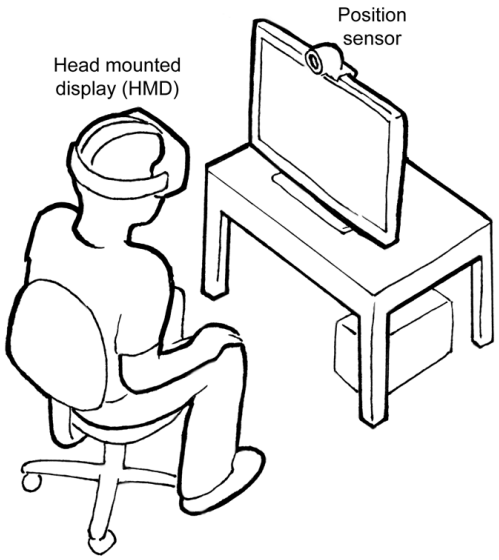
Slika 2.1: Koraki tehnike metanja žarkov[1]

2.2 Navidezna resničnost

Pojem navidezna resničnost (angl. virtual reality - VR) predstavlja interaktivno računalniško generirano uporabniško izkušnjo, s katero dajemo uporabniku občutek prisotnosti v navideznem okolju. Za ustvarjanje občutka prisotnosti v umetnem okolju poskušamo vplivati na različne čute, kot so vid, sluh, vonj in dotik. Sodobni sistemi za navidezno resničnost najpogosteje temeljijo na uporabi posebnih očal za navidezno resničnost (angl. head mounted displays - HMD) [39], ki si jih uporabnik nadene na glavo. Ta običajno vsebujejo dva zaslona (po enega za vsako oko). Zaslona pokrivata velik del vidnega polja in tako oči sprejmejo dve veliki sliki, ki ju možgani združijo v eno. Na ta način nam dajejo vtis, da se fizično nahajamo v računalniško poustvarjenem prostoru. Na sliki 2.2 je prikazan uporabnik, ki nosi očala

[1] `https://upload.wikimedia.org/wikipedia/commons/thumb/7/76/Volume_ray_casting.svg/800px-Volume_ray_casting.svg.png`

za prikaz navidezne resničnosti. Taka očala so običajno opremljena tudi s številnimi senzorji, ki beležijo vse premike in rotacije ter jih posredujejo programski opremi, ki skrbi za izris pogleda. Potek komunikacije, ki skrbi za ustrezen prikaz slike v navidezni resničnosti, si lahko pogledate na sliki 2.3.

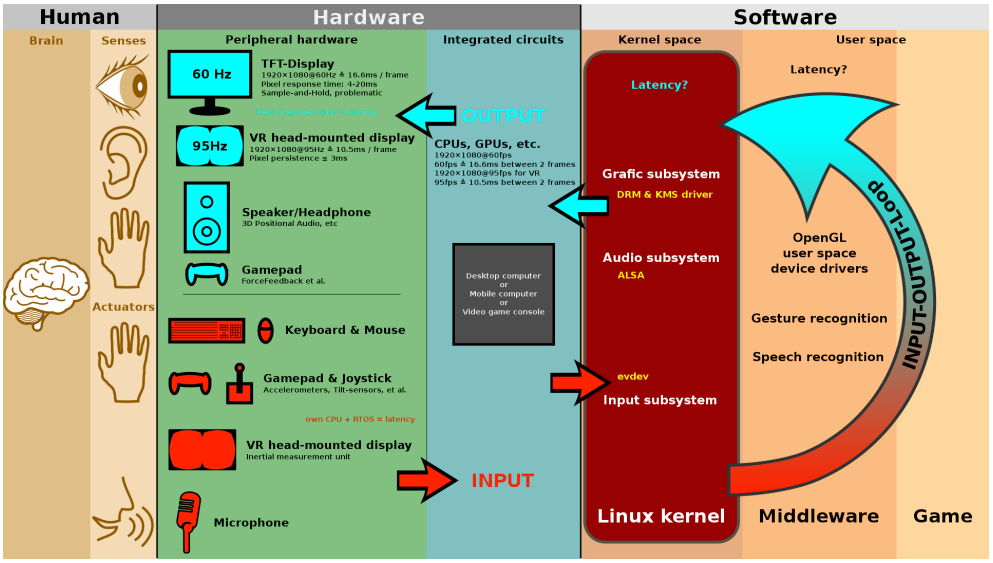


Slika 2.2: Sistem za prikazovanje navidezne resničnosti, ki vključuje očala za prikaz navidezne resničnosti[2]

Navidezna resničnost je zaradi svojih prednosti uporabna na mnogih področjih [9]:

- *Arhitektura* - V arhitekturi je možnost uporabe zelo velika, saj si lahko z njeno pomočjo načrtovane stavbe ogledamo, še preden so zgrajene.
- *Zabava* - Navidezna resničnost je vedno bolj prisotna v igričarski industriji, saj uporabniško izkušnjo dvigne na popolnoma novo raven. Poleg igričarske pa je na tem področju zelo dejavna tudi filmska industrija [8].
- *Vojska* - Mnoge prednosti uporabe navidezne resničnosti so prepoznali tudi v vojski. Tam se uporablja predvsem v simulatorjih letenja. Vedno

[2] `https://mdn.mozillademos.org/files/11035/hw-setup.png`



Slika 2.3: Prikaz povezanosti človeka, programske in strojne opreme[3]

bolj pa se razširja tudi na druga področja treninga in izobraževanja vojakov [33].

- *Izobraževanje* - Vse več idej in produktov pa v povezavi z navidezno resničnostjo nastaja tudi na področju izobraževanja. Tam nam razvite tehnologije lahko pomagajo do lažjega razumevanja različnih 3D modelov, ki si jih je težje predstavljati [21]. Prav tako pa so lahko zelo koristne za otroke s težavami v razvoju, saj lahko v navideznem svetu trenirajo določene spretnosti, ki jih v resničnem življenju ne bi mogli [31].

2.2.1 Razvoj navidezne resničnosti skozi zgodovino

Eno izmed prvih pomembnih odkritij na področju navidezne resničnosti sega v leto 1838, ko je Charles Wheatstone odkril, da možgani s procesiranjem dveh dvodimenzionalnih slik (po eno iz vsakega očesa) ustvarijo tridimenzi-

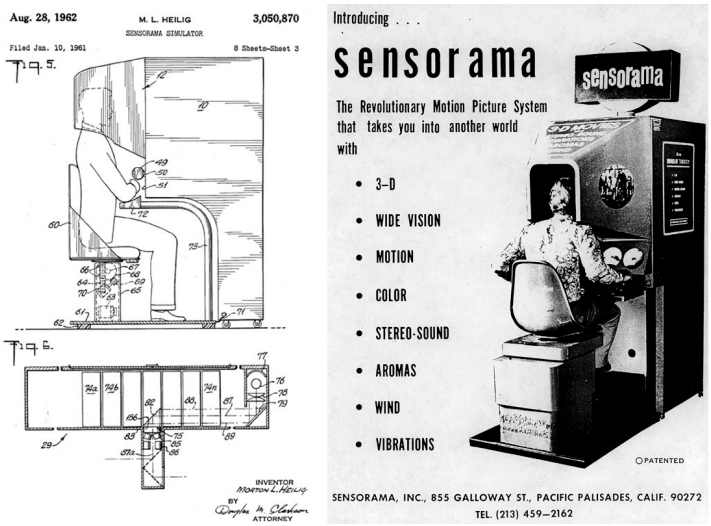
[3] https://en.wikipedia.org/wiki/Virtual_reality#/media/File:Linux_kernel_and_gaming_input-output_latency.svg

onalno predstavo sveta, v katerem živimo [47]. Razlika v položaju oči nam omogoča, da na ta način lažje pridobimo občutek za globino [16]. Na osnovi teh ugotovitev je predstavil napravo, imenovano stereoskop, ki je s pomočjo risb dajala občutek, da si objekt ogledujemo v treh dimenzijah [45]. Izgled stereoskopa je prikazan na sliki 2.4. Današnje naprave za prikaz navidezne resničnosti temeljijo prav na osnovi te ugotovitve o delovanju naših možganov.

Slika 2.4: Stereoskop[4]

Skozi leta so znanstveniki in inovatorji nadgrajevali ideje, uporabljene pri izdelavi stereoskopa. V 60. letih 20. stoletja je Morton Heilig razvil napravo, imenovano Sensorama (prikazana na sliki 2.5). To je bila prva naprava, ki je uporabniku v veliki meri dajala vtis, da se nahaja v navideznem svetu. Poleg prikaza slike v treh dimenzijah je naprava izvajala tudi vibracije in spuščala različne vonjave.

[4] https://spotlight.kcl.ac.uk/wp-content/uploads/sites/14/2016/10/Stereoscope-3.jpg



Slika 2.5: Sensorama[5]

Za naslednjo prelomnico velja izdelava prvih očal za prikaz navidezne resničnosti (angl. head mounted display - HMD), za katero sta zaslužna Ivan Sutherland in Davied Evans. Napravo sta poimenovala *The Sword of Damocles*. Bolj razširjeno uporabo so omejevale preslaba računalniška grafika, teža in kopica žic, potrebnih za pravilno delovanje naprave. Tehnologija in principi, ki sta jih uporabila, so postali temelj za nadaljnji razvoj na tem področju. Izgled prvih očal si lahko pogledate na sliki 2.6.

Velik del k razvoju navidezne resničnosti je prispevala tudi ameriška vesoljska agencija (angl. National Aeronautics and Space Administration - NASA). Za razvoj navidezne resničnosti je bil zelo pomemben projekt VIEW (angl. The virtual interface environment workstation), prikazan na sliki 2.7. S pomočjo tega sistema lahko uporabnik vstopi v računalniško generiran svet in s pomočjo premikanja glave ter posebnih rokavic s številnimi senzorji z njim tudi upravlja [36]. Tehnologije, razvite v tem obdobju, so še danes prisotne v številnih napravah za prikaz navidezne resničnosti.

[5] `https://www.avadirect.com/blog/wp-content/uploads/2015/08/Sensorama-History-of-VR.png`

Slika 2.6: Prva očala za prikaz navidezne resničnosti[6]

Slika 2.7: Sistem VIEW, Nasa[7]

[6] http://www.dsource.in/sites/default/files/course/virtual-reality-introduction/evolution-vr/sword-damocles-head-mounted-display/images/17.jpg

[7] https://www.nasa.gov/sites/default/files/styles/946xvariable_height/public/1990_a_new_continent.jpg

2.3 Navidezna resničnost v medicini

Med eno najbolj perspektivnih področij uporabe navidezne resničnosti spada tudi medicina. Podobno kot na ostalih področjih obstaja tudi tu več načinov uporabe [42]. V grobem lahko možnosti uporabe razdelimo na naslednja področja:

- *Diagnosticiranje* - Razvoj tehnologij za zajem medicinskih slik (angl. medical imaging - MI) je omogočil, da imamo danes na voljo posnetke delov teles v zelo visoki ločljivosti. Za učinkovito diagnosticiranje jih je potrebno tudi ustrezno vizualizirati. Vizualizacija v navidezni resničnosti tu lahko predstavlja dodano vrednost, saj si specialist poljuben objekt lahko pogleda v treh dimenzijah in ga tako lažje prepozna.

- *Načrtovanje operativnih posegov* - Zahtevnejše operacije zahtevajo temeljito pripravo in načrtovanje. Navidezna resničnost je kirurgom lahko v veliko pomoč, saj si lažje predstavljajo morebitne težave, s katerimi bi se med posegom lahko soočili.

- *Izobraževanje in usposabljanje specialistov* - Navidezna resničnost prinaša velike obete tudi v samo izobraževanje specialistov. Tu sta zelo pomembna predvsem učenje anatomije in trening operativnih posegov.

- *Lajšanje in zdravljenje fobij* - Zaradi občutka prisotnosti v nekem novem okolju se navidezna resničnost uporablja tudi za lajšanje in zdravljenje različnih strahov in fobij. Pacienti preko navidezne resničnosti doživljajo podobne situacije kot v resničnem življenju. Razlika pa je v tem, da so na ta način izpostavljeni doživljanju teh fobij v kontroliranem okolju. Znani so primeri zdravljenja vojakov za posledicami post-travmatske stresne motnje (angl. posttraumatic stress disorder - PSTD), strahu pred letenjem, pajki, klavstrofobijo in še mnogimi drugimi fobijami [15] [38].

- *Rehabilitacija* - Navidezna resničnost ponuja tudi nove možnosti zdravljenja nekaterih zdravstvenih težav, povezanih s poškodbami glave in

možganske kapi [18]. Ključna prednost je v tem, da lahko prek navidezne resničnosti vplivamo na pacientovo dojemanje sveta okoli sebe in ga prilagajamo glede na njegovo zdravstveno stanje.

2.4 Sistemi za prikaz navidezne resničnosti

Z razvojem in komercializacijo navidezne resničnosti postajajo sistemi vse bolj dostopni uporabnikom. V tem poglavju bomo opisali sisteme, ki so trenutno najbolj popularni na trgu. Sistemi se delijo na tiste, ki delujejo s pomočjo mobilnega telefona, in na namenske sisteme.

2.4.1 Sistemi za prikaz navidezne resničnosti, ki uporabljajo pametni telefon

V zadnjih letih so se na trgu pojavili sistemi za prikaz navidezne resničnosti, ki za njen prikaz potrebujejo zgolj dovolj zmogljiv pametni telefon. Veliki prednosti takih sistemov sta njihova dostopnost in razširjenost. Zaradi enostavnosti svojega delovanja pa se po kvaliteti uporabniške izkušnje ne morejo primerjati z bolj specializiranimi sistemi. Najbolj znani predstavniki na tem področju so:

- *Google Cardboard* - Gre za zelo enostaven sistem, ki ga je Google predstavil leta 2014. Google Cardboard so enostavna kartonasta očala, kamor vstavimo pametni telefon, ki nato s svojim zaslonom, senzorji in namenskimi aplikacijami skrbi za ustrezen prikaz.

- *Google Daydream* - Temelji na istih principih kot Google Cardboard. Razlika je v tem, da so očala za navidezno resničnost, kamor vstavimo telefon, bolje izdelana in nam dajejo boljši občutek prisotnosti v navideznem prostoru. Poleg očal nam je na voljo tudi upravljalnik (angl. controller), ki se preko Bluetootha poveže z našim telefonom in nam omogoča dodatno interakcijo z navideznim svetom.

- *Samsung Gear VR* - Je zelo podoben sistemu Google Daydream, a je namenjen izključno telefonom podjetja Samsung.

2.4.2 Namenski sistemi

Najbolj tipični sistemi za prikaz navidezne resničnosti delujejo v povezavi z osebnim računalnikom. Običajno vsebujejo očala za prikaz navidezne resničnosti s številnimi senzorji in dvema zaslonoma. Nekateri vsebujejo tudi posebne upravljalnike, s katerimi lažje manipuliramo z objekti v navideznem svetu. Za svoje delovanje pa potrebujejo povezavo z dovolj zmogljivim računalnikom, na katerem teče namenska programska oprema. Najbolj priljubljeni sistemi, ki delujejo na opisan način so:

- *Oculus Rift* - Je sistem za prikaz navidezne resničnosti podjetja Oculus. Na trg je prispel leta 2016 in s tem postal eden najbolj prepoznavnih produktov na trgu. Sistem omogoča lokacijsko sledenje s šestimi prostostnimi stopnjami (tri ustrezajo premiku v treh neodvisnih smereh tridimenzionalnega prostora, druge tri pa vrtenju okrog treh glavnih osi). Očala vsebujejo dva kakovostna zaslona z visoko frekvenco osveževanja in slušalke, ki uporabniku pričarajo tridimenzionalne zvočne učinke. Na samih očalih je tudi več senzorjev, ki skrbijo za natančno spremljanje uporabnikovih premikov. Poleg senzorjev na samih očalih pa so na voljo tudi senzorji, ki jih postavimo v prostor in omogočajo še bolj natančno spremljanje uporabnikovih gest. Za lažjo interakcijo z navideznim svetom so vključeni tudi posebni upravljalniki, ki jih držimo v rokah.

- *HTC Vive* - Smo uporabljali za testiranje delovanja naše aplikacije v navidezni resničnosti. Izgled očal za prikaz navidezne resničnosti in upravljalniki so prikazani na sliki 2.8. Produkt ponuja zelo podobne senzorje kot prej opisani Oculus Rift. Za svoje delovanje potrebuje večji prostor, a vsebuje tudi boljše zunanje sledilne postaje, ki omogočajo natančnejše sledenje uporabnikovim gibom v prostoru. Slaba stran

sistema je, da ne vključuje slušalk. To slabost pa odpravlja naslednja generacija sistema, imenovana HTC Vive Pro. Poleg vgrajenih slušalk vsebuje tudi kamero, kar omogoča prikaz obogatene resničnosti.

Slika 2.8: Sistem za prikaz navidezne resničnosti HTC Vive[8]

- *Sony Playstation VR* - Je sistem, ki uporabniku omogoča izkušnjo navidezne resničnosti z uporabo igralne konzole Playstation 4. Za sledenje lokaciji v prostoru pa namesto posebnih senzorjev uporablja posebno kamero, ki jo priključimo na Playstation 4.

2.5 Spletne tehnologije

Deljenje podatkov in vizualizacij še nikoli ni bilo tako preprosto, kot je danes. Izmenjavo informacij z ljudmi s celega sveta nam omogoča uporaba svetovnega spleta. Ta se je v zadnjih desetletjih zelo razširil in je danes dostopen že več kot 54 odstotkom vseh ljudi [17]. Zaradi svoje razširjenosti postaja zelo priljubljena platforma za razvoj raznih aplikacij, ki tečejo v spletnih brskalnikih. V namen poenostavitve implementacij spletnih aplikacij obstaja mnogo knjižnic in različnih orodij, ki ta razvoj pospešijo in olajšajo. V naslednjih

[8] `http://www.affinityvr.com/wp-content/uploads/2016/07/tweedievr-702x336.png`

podpoglavjih so opisane knjižnice in orodja, ki so bila najbolj uporabna pri razvoju vizualizacijskega ogrodja in razširitev, implementiranih v sklopu te diplomske naloge.

2.5.1 Knjižnica WebGL

Za razvoj vizualizacijskega ogrodja Med3D je bila najpomembnejša knjižnica WebGL (angl. Web Graphics Library)[9], ki nam omogoča strojno pospešeno grafično upodabljanje znotraj brskalnika. Aplikacije, ki uporabljajo knjižnico WebGL, so sestavljene iz dveh delov. Prvi del je spisan v jeziku JavaScript. Njegova naloga je prenašanje podatkov med grafično kartico in računalnikom. Poleg prenašanja podatkov pa omogoča tudi izbiro nastavitev, ki vplivajo na končni izris. Drugi del predstavljajo programi, napisani v namenskem programskem jeziku, imenovanem GLSL (angl. Graphics Library Shading Language)[10]. Jezik je v veliki meri podoben programskemu jeziku C in je bil leta 2004 definiran s standardom OpenGL 2.0. Ker je knjižnica WebGL precej nizkonivojska, so skozi leta nastale mnoge knjižnice, ki abstrahirajo njeno delovanje. Dandanes sta najbolj popularni knjižnici Three.js[11] in BabylonJS[12]. Kljub obstoju naštetih knjižnic se v ogrodju Med3D v veliki meri zanašamo na neposredno uporabo knjižnice WebGL, saj nam omogoča večjo prilagodljivost. GLSL ima tako kot večina ostalih programskih jezikov več verzij, saj tehnologija hitro napreduje in izraža vedno nove potrebe. Tak primer je uporaba 3D tekstur, ki je omogočena v verziji 3.00. Več o njih si lahko preberete na povezavi [40].

2.5.2 Ogrodji Angular JS, Bootstrap

Zaradi potrebe da imamo pri razvoju grafične aplikacije s pomočjo knjižnice WebGL v aplikaciji eno platno, na katero izrisujemo želene stvari, je bilo

[9] `https://www.khronos.org/webgl/`
[10] `https://www.khronos.org/opengl/wiki/OpenGL_Shading_Language`
[11] `https://threejs.org/`
[12] `https://www.babylonjs.com/`

potrebno ogrodje razviti v okviru ene strani (angl. single page application - SPA). Za razvoj je bilo uporabljeno ogrodje AngularJS[13]. Za izgled aplikacije v veliki meri skrbi ogrodje Bootstrap[14], ki vsebuje mnoge med seboj usklajene elemente uporabniškega vmesnika. Z uporabo jezika CSS (angl. cascading style sheets) pa jih je mogoče po potrebi tudi spreminjati in tako oblikovati željeno grafično podobo.

2.5.3 Programski vmesnik WebVR API

Za pomoč pri implementaciji navidezne resničnosti v ogrodje Med3D smo uporabili WebVR API (angl. application programming interface). Gre za eksperimentalen programski vmesnik, ki omogoča uporabo v programskem jeziku JavaScript. Nudi nam podporo za naprave, ki prikazujejo navidezno resničnost. Njegova prednost je v tem, da nudi podporo za več različnih sistemov za prikaz navidezne resničnosti in tako omogoča uporabo iste kode, kar zmanjša možnosti za napake ter pohitri razvoj programske opreme. Druga velika prednost pa je v tem, da teče v internetnem brskalniku. Leta 2014 ga je zasnoval Vladimir Vukćević iz podjetja Mozzila. Prvega marca leta 2016 pa je izšla prva verzija (1.0). Zadnja izdana verzija nosi številko 1.1 in je izšla 5. aprila 2017. Njegova podpora je vezana na brskalnik in operacijski sistem in žal še ni popolna. Podprtost v brskalnikih si lahko ogledate na povezavi[15].

Tekom pisanja diplomske naloge je izšel opis novega standarda, imenovanega WebXR Device API [7], ki bo nadomestil WebVR API. To priča o aktualnosti in hitrem razvoju na tem področju.

2.5.4 Platfroma Node.js

Večina spletnih aplikacij poleg odjemalčevega dela aplikacije potrebuje tudi strežniški del. To nam omogoča shranjevanje podatkov, interakcijo med uporabniki in mnoge druge priljubljene akcije uporabnikov. Za implementacijo

[13] `https://angularjs.org/`
[14] `https://www.getbootstrap.com`
[15] `https://webvr.rocks/#browsers`

strežnika v aplikaciji Med3D je uporabljena platforma Node.js[16].

2.5.5 Programska vmesnika Gamepad API in Gamepad Extensions API

Gamepad API[17] je programski vmesnik, ki nam nudi možnost dostopa do upravljalnikov, priključenih na računalnik. Dostop do upravljalnikov nam omogoča prek vmesnika (angl. interface) *Gamepad*, ki predstavlja posamezen upravljalnik. Na ta način lahko poizvedujemo o stanju posameznih gumbov in o drugih informacijah upravljalnika. Gamepad Extensions API pa, kot že ime pove, prinaša razširitve Gamepad API-ja. Njegove glavne prednosti so predvsem to, da nam ponuja napredne podatke, kot sta pozicija in orientacija upravljalnika, ter možnost haptične povratne informacije na upravljalnikih, ki to omogočajo. Opozoriti pa je potrebno, da gre za precej nov programski vmesnik in je njegova uporaba zato omejena na določene brskalnike[18].

[16] `https://nodejs.org/`

[17] `https://www.w3.org/TR/gamepad/`

[18] `https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API`

Poglavje 3

Vizualizacijsko ogrodje na spletu

V nadaljevanju bomo predstavili spletno vizualizacijsko ogrodje Med3D, razvito na Fakulteti za računalništvo in informatiko. Predstavili bomo razloge za razvoj takega ogrodja in se spustili v arhitekturo samega ogrodja. Tako bomo spoznali module, ki ga sestavljajo. Na kratko bomo predstavili tudi uporabniški vmesnik, ki je ključnega pomena za dobro uporabniško izkušnjo.

3.1 Predstavitev ogrodja

3.1.1 Obstoječa ogrodja

Vizualizacija (medicinskih) volumetričnih podatkov ni povsem novo področje, zato je bilo na tem področju razvitih že več aplikacij in ogrodij. Med popularnejše sodijo SimVascular [26], Exposure Renderer [23] in ParaView [2]. Našteta orodja so namenjena uporabi na osebnih računalnikih. Prvi dve sta namenjeni lokalni vizualizaciji volumetričnih podatkov in vključujeta tudi nekatere napredne osvetlitvene tehnike, zadnji pa je namenjen paralelni obdelavi obsežnih podatkov s pomočjo namenskih strežnikov in vizualizaciji rezultatov. Ogrodje ParaViewWeb [19] je ogrodje, ki omogoča delovanje na mnogo platformah, saj teče v spletnem brskalniku. Nobeno od naštetih

orodij pa ne omogoča oddaljenega sodelovanja med uporabniki in prikaza vizualiziranih podatkov v navidezni resničnosti. Kombinacijo vizualizacije medicinskih volumetričnih podatkov in navidezne resničnosti pa predstavlja aplikacija SpectroVive [41]. Ta je bila razvija na univerzi v Baslu kot del projekta MIRACLE [34].

3.1.2 Predstavitev ogrodja Med3D

Ogrodje Med3D temelji na predhodno razvitem ogrodju Neck Veins [6]. Razvito je bilo z namenom vizualizacije medicinskih volumetričnih podatkov neposredno v spletnem brskalniku, saj tako omogočimo široko dostopnost. Ogrodje podpira tudi možnost oddaljenega sodelovanja med uporabniki in prikaz vizualiziranih podatkov v navidezni resničnosti, za katerega pa ne potrebujemo posebne programske opreme. Implementiran je namreč s pomočjo standarda WebVR in omogoča tak prikaz s pomočjo brskalnika. Oddaljeno sodelovanje pa omogoča klepet med uporabniki, deljenje anotacij in deljenje pogleda, kar je zelo koristno pri sodelovanju med uporabniki, ki se nahajajo na različnih koncih sveta. Zaledni del sistema omogoča tudi hranjenje podatkov in omogoča številne nadgradnje v prihodnosti.

3.1.3 Arhitektura sistema

Ogrodje Med3D je razvito s pomočjo ogrodja AngularJS. Ta omogoča razvoj aplikacij SPA (angl. single page application). Aplikacije, razite z AngularJS, so običajno sestavljene iz več modulov. Ti moduli kasneje implementirajo še druge dele aplikacije, kot so komponente, storitve ipd. V nadaljevanju poglavja bomo predstavili dele ogrodja Med3D, ki so najbolj pomembni za njegovo delovanje in sem se z njimi ob razvoju razširitev bolje spoznal:

- *Kontroler upodabljanja (angl. rengering controller)* - Gre za najbrž najpomembnejši del ogrodja, saj skrbi za upodabljanje, posodabljanje scene in izris na platno, kjer vidimo željeni prikaz.

- *Kontroler nalaganja mrež (angl. mesh loading controller)* - Da lahko s kontrolerjem upodabljanja kaj prikažemo, moramo ogrodju najprej zagotoviti podatke. Pri tem je dejaven eden od dveh kontrolerjev, napisanih v ta namen. Prvi je kontroler nalaganja mrež, ki skrbi, da kontrolerju upodabljanja zagotovi podatke, če so le-ti naloženi v obliki poligonskih mrež.

- *Kontroler nalaganja volumetričnih podatkov (angl. volume loading controller)* - Ta kontroler ima podobno nalogo kot prej omenjeni kontroler nalaganja mrež, le da ga uporabimo, ko uporabnik hoče vizualizirati volumetrične podatke. Poleg nalaganja datotek iz diska na osebnem računalniku nam ogrodje omogoča hranjenje teh podatkov na strežniku. Tam se ti podatki hranijo v podatkovni bazi MongoDB.

- *Storitev za upravljanje nalog (angl. task management service)* - Omogoča asinhrono delovanje ogrodja, kar izboljša uporabniško izkušnjo pri časovno zahtevnih operacijah (npr. nalaganju velikih datotek).

- *Kontroler za upravljanje nalog (angl. task manager controller)* - Skrbi za prikaz napredka operacij na uporabniškem vmesniku. Na ta način zagotavlja boljšo uporabniško izkušnjo, saj uporabnik takoj dobi povratno informacijo v zvezi z njegovo akcijo.

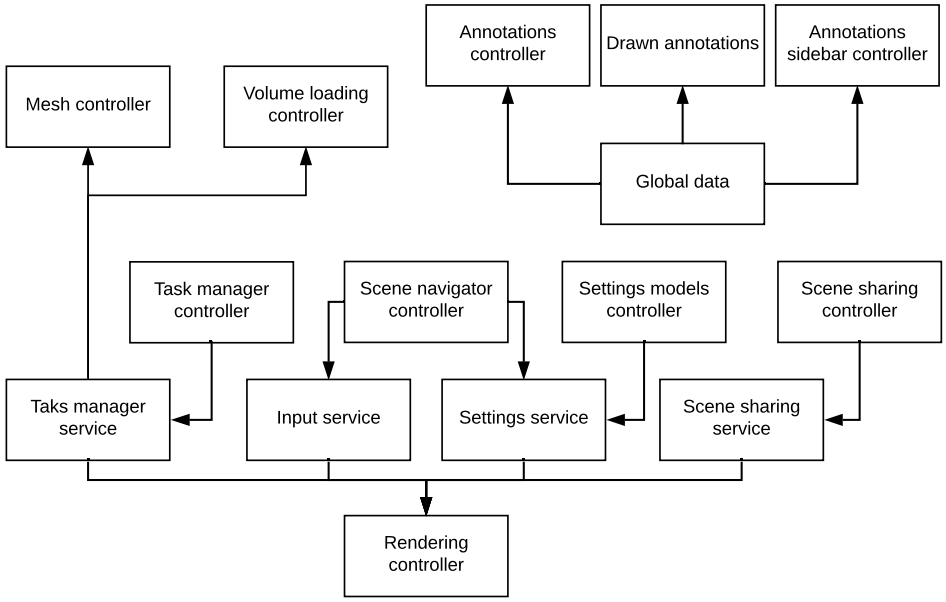
- *Kontroler scenskih navigatorjev (angl. scele navigation controller)* - Ta kontroler skrbi za interakcijo uporabnika z objektom na sceni preko grafičnih navigatorjev na sceni.

- *Storitev za nadzor vhodnih enot (angl. input service)* - Skrbi za pravilno interpretacijo vnesenih uporabniških ukazov o premikih scene prek grafičnih navigatorjev ter vnosov s pomočjo tipkovnice in miške.

- *Storitev in kontroler za nastavitve (angl. settings controller)* - Vsem prej omenjnenim kontrolerjem lahko do neke mere konfiguriramo nastavitve. To omogočata storitev in kontroler za nastavitve. Ob spre-

membah nastavitev se podatki dinamično posodobijo s pomočjo dvosmernega povezovanja podatkov (angl. two way data binding). Storitev za nastavitve je globalna in je na voljo vsem ostalim kontrolerjem, ki potem delujejo skladno z nastavitvami, zapisanimi v njej.

Vse zgoraj opisane komponente so skupaj s preostalimi podrobneje razložene v diplomski nalogi Primoža Lavriča [27].



Slika 3.1: Arhitektura ogrodja Med3D

3.1.4 Uporabniški vmesnik

Uporabniški vmesnik predstavlja zelo pomemben del aplikacije, saj povezuje uporabnika s funkcionalnostmi, ki jih aplikacija omogoča. Uporabnik lahko s pomočjo vhodno-izhodnih naprav nadzoruje delovanje aplikacije, ki teče na napravi. Za boljšo uporabniško izkušnjo je potreben enostaven uporabniški vmesnik, ki pa uporabniku hkrati omogoča uporabo vseh implementiranih funkcionalnosti ogrodja. Prav tako pomembna pa sta tudi njegov videz in konsistentnost, saj tako ogrodje naredimo bolj prijazno za uporabo.

V primeru ogrodja Med3D je uporabniški vmesnik (ogledate si ga lahko na sliki 3.2) zasnovan s pomočjo elementov, ki jih definira standardni označevalni jezik za izdelavo spletnih strani HTML (angl. Hyper Text Markup Language)[1]. Uporaba jezika HTML je logična posledica tega, da je ogrodje realizirano znotraj spletnega brskalnika. Jezik HTML v osnovi definira hierarhijo gradnikov. Za dodajanje želenih funkcionalnosti in videza se običajno uporablja v kombinaciji s slogovnim jezikom CSS (angl. Cascading Style Sheets)[2]. Kot smo omenili v poglavju 2.5, sta za razvoj uporabljeni tudi ogrodji Bootsrap in Angular.js. V nadaljevanju bomo predstavili bistvene elemente uporabniškega vmesnika:

- *Navigacijska vrstica* - Navigacijska vrstica se nahaja na zgornjem robu zavihka, v katerem teče aplikacija Med3D. Z njeno pomočjo lahko dostopamo do različnih oken, ki nam nudijo dostop do raznih funkcionalnosti ogrodja. Temelji na Bootstrapovi predlogi *Navbar*, a ima spremenjen videz, da se lepše sklada z ostalimi elementi aplikacije.

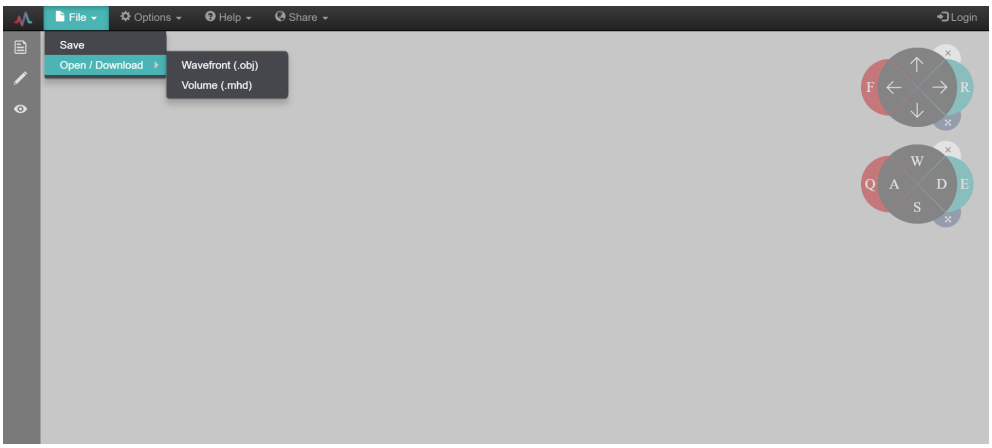
- *Okna* - Zaradi potrebe da je ogrodje implementirano kot SPA, je prikaz vsebine uporabniškega vmesnika narejen na dinamičen način, saj bi nam preusmerjanje na novo stran porušilo zasnovo enostranske spletne aplikacije. V ta namen so uporabljeni Bootstrapovi objekti *Modal*, ki implementirajo modalna okna. Ta je možno premikati po zaslonu znotraj zavihka, kjer teče ogrodje. Bistveno pri tem pa je, da omogočajo dinamičen prikaz podatkov, saj ne želimo preusmerjanja in nalaganja novih strani.

- *Stranska vrstica* - V stranski vrstici se nahajajo orodja, namenjena interakciji z vizualizacijami. Ta se nahaja na levi strani zaslona in je uporabniku dostopna ves čas, saj je njena uporaba precej pogosta in je zato ključno, da je dostop preprost in hiter.

[1] `https://www.w3.org/html/`
[2] `https://www.w3.org/Style/CSS/Overview.en.html`

- *Grafični navigatorji in dodatni gumbi* - Na osnovnem zaslonu sta prisotna tudi dva navigatorja, ki ju je mogoče tudi odstraniti. Prvi omogoča pomikanje, drugi pa rotiranje kamere, s katero gledamo na objekt. Navigatorji predstavljajo alternativo tipkam, določenim za upravljanje pomikov in rotacij, in so za novega uporabnika bolj preprosti za uporabo. Ko brskalnik zazna, da so na voljo očala za prikaz navidezne resničnosti, se v spodnjem desnem kotu pojavi poseben gumb. Ob kliku na gumb iz običajnega pogleda pridemo v pogled prek navidezne resničnosti. V tem pogledu se nam začasno skrijejo tudi stranska vrstica in grafični navigatorji, ker tam izgubijo svojo uporabnost. Za izhod iz načina prikaza navidezne resničnosti nam je prav tako na voljo gumb v spodnjem desnem delu zaslona, ki skrbi za vrnitev na običajen pogled na trenutno vizualizacijo.

- *Podpora uporabniškim anotacijam* - Uporabniške anotacije so najbolj kompleksen del uporabniškega vmesnika ogrodja Med3D. H kompleksnosti pripomore dejstvo, da je treba del anotacij prikazati na platnu, kjer je vizualiziran objekt, del pa v okviru spletne strani kot del HTML dokumenta. Ogrodje hrani vse anotacije v *singleton* objektih, ki so dostopni vsem delom ogrodja. To nam omogoča, da lahko anotacije ustvarimo s pomočjo različnih delov ogrodja in tam do njih tudi dostopamo. Podatki so povezani s pomočjo dvosmernega podatkovnega povezovanja (angl. two way data binding), kar nam zagotavlja, da se spremembe anotacije takoj odražajo povsod, kjer je spremenjena anotacija uporabljena. Izrisanim objektom anotacije dodajamo z uporabo miške, tako da kliknemo na del vizualiziranega objekta, kamor želimo dodati anotacijo. S klikom se na objektu prikaže krog in se s črto poveže na okno anotacije. Točko na objektu, kamor smo kliknili, dobimo s pomočjo tehnike iskanja preseka z žarkom (angl. ray casting).



Slika 3.2: Uporabniški vmesnik ogrodja Med3D

Poglavje 4

Implementacija

4.1 Prikaz v navidezni resničnosti

Implementacije navidezne resničnosti v ogrodju Med3D smo se lotili s pomočjo uporabe knjižnice WebVR. Ta nam omogoča, da na očalih za ogled navidezne resničnosti le-to prikazujemo kar prek spletnega brskalnika. Več podrobnosti o aplikacijskem programskem vmesniku smo predstavili v poglavju 2.5.3.

Za prikaz vizualizacije v navidezni resničnosti moramo najprej pridobiti sezam trenutno priključenih naprav, ki so jo sposobne predvajati, ter izmed njih izbrati tisto, na kateri želimo prikazovati. Do informacij o izbrani napravi dostopamo prek objekta[1], ki predstavlja to napravo. Preden vstopimo v zanko, v kateri bomo izrisovali izbrano vizualizacijo, moramo pravilno nastaviti velikost platna (angl. canvas), da bo le-ta ustrezala resoluciji naprave, kjer želimo prikazovati navidezno resničnost.

Za preklop iz običajnega pogleda v pogled navidezne resničnosti moramo od izbrane naprave zahtevati predvajanje. To lahko zaradi varnostnih razlogov storimo le kot odziv na uporabnikovo akcijo (npr. klik gumba). Ob zahtevi za predvajanje moramo zagotoviti tudi dovolj visoko frekvenco osveževanja slike (angl. refresh rate). Ta ob običajni rabi v brskalniku tipično znaša 60 Hz (vsebina se vsako sekundo izriše šestdesetkrat). Za prikaz navi-

[1] `https://developer.mozilla.org/en-US/docs/Web/API/VRDisplay`

dezne resničnosti pa ta frekvenca ni dovolj visoka in lahko povzroča glavobole ter slabost [44]. Priporočljiva frekvenca za navidezno resničnost znaša 90 Hz in jo omogočata tudi obe najbolj popularni namenski napravi na trgu (HTC Vive in Oculus Rift).

V spodnjem izseku kode je predstavljen enostaven primer, kako ob kliku na gumb zahtevati predvajanje na očalih za navidezno resničnost.

```
btn.addEventListener('click', function() {
  if(btn.textContent === 'Start VR display') {
    vrDisplay.requestPresent([{source:c}]).then(function() {
      console.log('Presenting to WebVR display');
      displayVRMode();
    }
  }
}
```

Znotraj zanke izrisovanja za vsak izris iz objekta, ki predstavlja napravo, pridobimo podatke o projekcijski matriki (angl. projection matrix) in matriki pogleda (angl. view matrix) za trenutni izris. Ker prikazujemo sliko posebej za levo in desno oko, moramo za vsak izris pridobiti obe matriki, saj le tako lahko prikažemo ustrezno sliko vizualiziranega objekta. S pomočjo pridobljenih matrik izrišemo sceno. Iz projekcijske matrike moramo izračunati ustrezne parametre kamere. Pomen posameznih številk v projekcijski matriki je prikazan na sliki 4.1. Na levi del platna upodobimo pogled levega očesa, na desni del platna pa pogled desnega očesa. Po zaključenem izrisu sliko s pomočjo namenske funkcije pošljemo na napravo, kjer jo nato vidi uporabnik sistema.

Celotno opisano funkcionalnost smo v sklopu tega diplomskega dela vgradili v spletno vizualizacijsko ogrodje Med3D. Največ sprememb je bilo potrebnih v kontrolerju izrisa (angl. rendering controller), saj smo tam morali ogrodje prilagoditi, da smo lahko omogočili izris navidezne resničnosti. Poleg celotne inicializacije smo morali uporabniku s pomočjo gumba ponuditi možnost vstopa v pogled navidezne resničnosti. Ob uporabnikovem kliku na

```
2*near/(right-left)          0                    0                  -near*(right+left)/(right-left)
        0            2*near/(top-bottom)          0                  -near*(top+bottom)/(top-bottom)
        0                    0          -(far+near)/(far-near)          2*far*near/(near-far)
        0                    0                   -1                             0
```

Slika 4.1: Projekcijska matrika

gumb je potrebno prekiniti aktualno zanko izrisovanja in sprožiti novo, ki zagotavlja hitrejše osveževanje ter prikazuje vizualizirane objekte v načinu navidezne resničnosti. Za ta izris smo ustvarili dva upodobitvena prehoda (angl. render pass), ki izrišeta vsak svojo sliko. Sliki smo nato naknadno združili v eno.

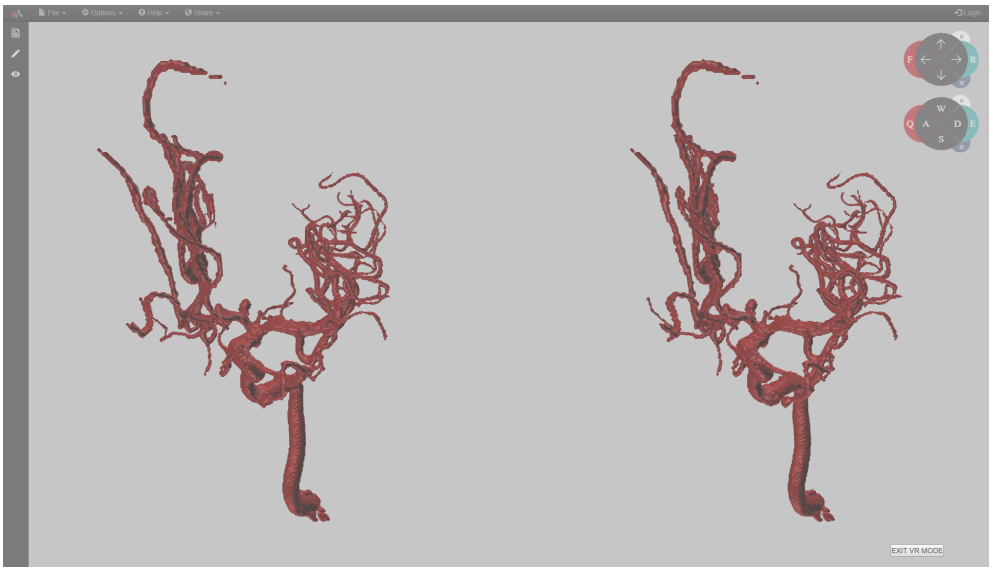
Poleg tega smo ustvarili tudi nov razred, v katerem smo definirali kamero za prikaz navidezne resničnosti. Ta vsebuje dve perspektivni kameri (angl. perspective camera), ki predstavljata levo in desno oko. Ob prehajanju med obema pogledoma je potrebno skrbeti, da vzdržujemo pravilno velikost slike in razmerje stranic (angl. aspect ratio) glede na napravo, kjer sliko prikazujemo.

Primer izrisa slike v načinu navidezne resničnosti si lahko pogledate na sliki 4.2. Potek komunikacije, da do takega prikaza pride, pa je opisan na sliki 4.3.

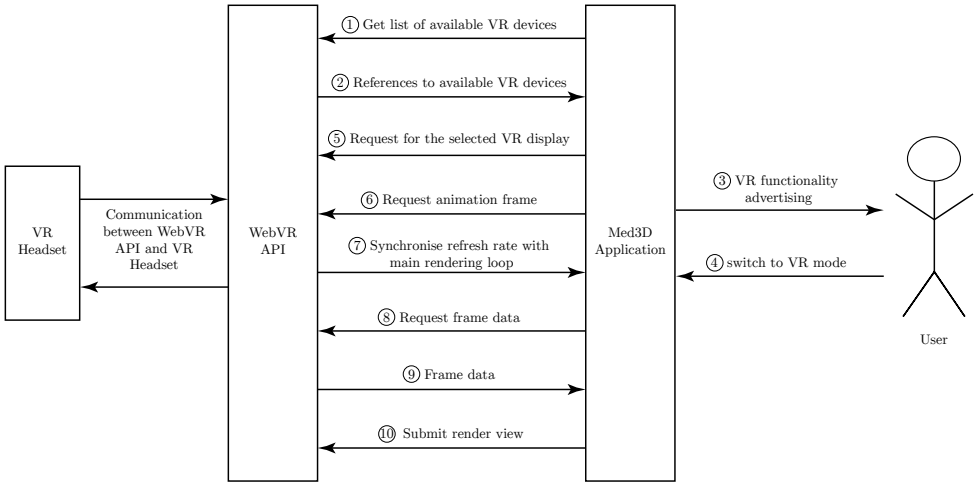
4.2 Upravljalniki za interakcijo v navidezni resničnosti

Za dodatno interakcijo s sceno (poleg spreminjanja pogleda ob spremembah položaja očal za prikaz navidezne resničnosti) smo poskrbeli z implementacijo upravljalnikov za navidezno resničnost. Pri tem smo si pomagali s programskim vmesnikom Gamepad API[2]. Ta nam nudi možnost za branje in odzivanje na signale upravljalnikov. V našem primeru smo se osredotočili na

[2]`https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API`

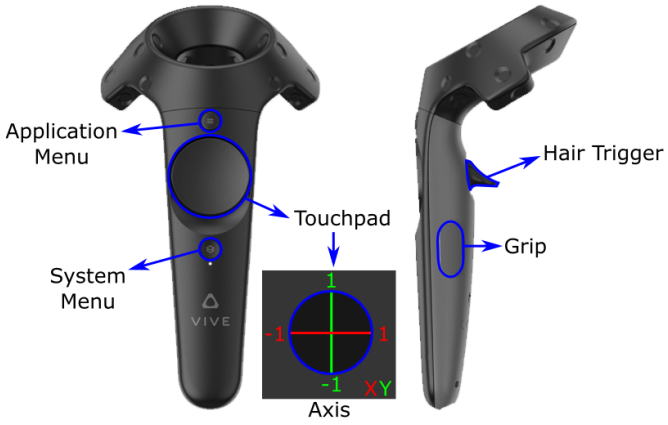


Slika 4.2: Izris objekta v načinu navidezne resničnosti



Slika 4.3: Delovanje WebVR v ogrodju Med3D.

upravljalnike, ki so del sistema HTC Vive in so prikazani na sliki 4.4.



Slika 4.4: Upravljalniki sistema HTC Vive[3]

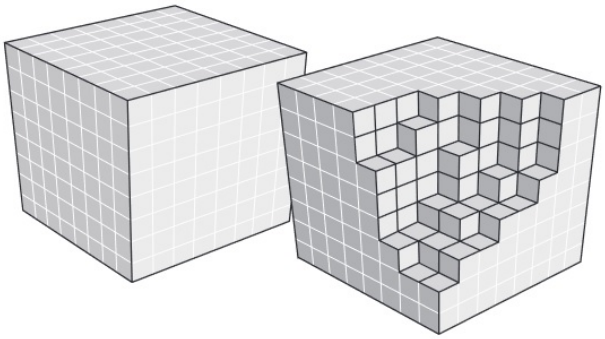
Pri implementaciji upravljalnikov v ogrodje Med3D smo iz seznama vseh priključenih upravljalnikov izbrali tiste, ki so del sistema za prikaz navidezne resničnosti. Na pritiske gumbov levega upravljalnika in na drsenje po njegovi površini, občutljivi na dotik, se odzovemo s premikanjem scene po vseh treh glavnih smereh. Pri tem površina občutljiva na dotik skrbi za premikanje v dveh smereh. Za pomik v tretji smeri pa poskrbimo z gumbi. Podobni logiki sledimo tudi pri drugem upravljalniku, le da tam namesto premikanja scene le-to rotiramo.

4.3 Tehnika metanja žarkov

Eden izmed ciljev diplomske naloge je bila tudi implementacija vizualizacije s tehniko metanja žarkov. Algoritem smo predstavili v podpoglavju 2.1.5, zato se bomo tu posvetili njegovi implementaciji. Algoritem se izvaja na podatkih, ki jih želimo vizualizirati. Ti so predstavljeni v obliki tridimenzionalnega skalarnega polja, kjer vsak skalar predstavlja intenziteto vrednosti v določeni točki v prostoru - vokslu. Za lažjo predstavo si na sliki 4.5 lahko predstavljate,

[3] https://koenig-media.raywenderlich.com/uploads/2016/12/ViveControllerButtons.png

da ima vsaka mala kocka svojo številsko vrednost, ta pa predstavlja lastnost skeniranega materiala (oz. tkiva) na njenem mestu.



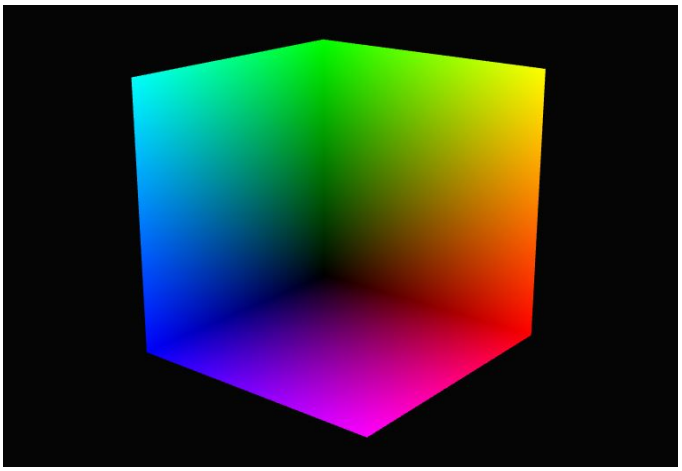
Slika 4.5: Slika kocke, sestavljene iz manjših kock, ki predstavljajo posamezne voksle s številskimi vrednostmi[5]

Taki podatki so običajno na voljo v obliki para datotek s končnicama *.mhd* in *.raw*. V datoteki s končnico *.mhd* se nahajajo metapodatki, ki povedo dimenzije slike, njeno orientacijo, pozicijo in druge podatke, ki jih potrebujemo za branje datoteke s končnico *.raw*, ki dejansko vsebuje te podatke. Za branje tovrstnih datotek se zanašamo na obstoječo rešitev, ki nam jo ponuja ogrodje Med3D. Ta nam preko razreda *MHDReader* ponuja možnost branja tovrstnih datotek na način, da nam vrne številsko polje z vsebovanimi podatki, zapisanimi v vhodni *.raw* datoteki. To polje moramo nato shraniti v teksturo in s pomočjo knjižnice WebGL poslati na grafično kartico, kjer se bo algoritem metanja žarkov izvajal.

Ko smo ustrezno poskrbeli za pripravo podatkov, smo se lotili implementacije samega algoritma. Ker vizualizacija teče na grafični kartici, moramo algoritem napisati v programskem jeziku GLSL, ki sem ga predstavil v podpoglavju 2.5.1. Ti programi so sestavljeni iz senčilnika oglišč in senčilnika fragmentov, ki vedno delujeta v paru. Za potrebe naše implementacije smo morali napisati dva taka para senčilnikov. V prvem paru senčilnikov prido-

[5] `http://lebarbacom.azurewebsites.net/wp-content/uploads/2014/11/voxels.jpg`

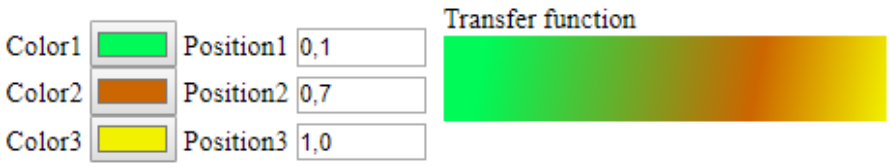
bimo koordinate končnih točk žarka, ki ga bomo kasneje navidezno pošiljali skozi teksturo, ki vsebuje prebrane podatke. Podatke o koordinatah zakodiramo v RGB vrednosti barve posameznih slikovnih točk, tako da vsako od smeri X, Y in Z zapišemo v svoj barvni kanal RGB. Izhodno sliko zapišemo v teksturo, ki bo služila kot eden od vhodov v drugi par senčilnikov. V kolikor izrišemo izhod prvega para senčilnikov, dobimo kocko, ki je prikazana na sliki 4.6.



Slika 4.6: Izhod prvega para senčilnikov - kocka, kjer so koordinate konca žarkov kodirane v posameznem kanalu RGB barve

Namesto izrisa pa ta izhod, kot rečeno, shranimo v teksturo in kot enega od vhodnih podatkov podamo v naslednji par senčilnikov, kjer dejansko implementiramo metanje žarkov. Poleg tega moramo podati tudi prenosno funkcijo, teksturo s prebranimi volumetričnimi podatki in parametre za izris. Prenosno funkcijo lahko uporabnik sam poljubno nastavlja z izbiro treh barv in njihovih pozicij na intervalu od 0 do 1. Barve na preostalem delu prenosne funkcije se izračunajo z linearno interpolacijo. Primer prenosne funkcije je predstavljen na sliki 4.7

V tem senčilniku oglišč ustvarimo dva izhoda, ki predstavljata koordinato točke na zaslonu in koordinato točke v prostoru, ki ju potem v pripadajočem senčilniku fragmentov uporabimo za pridobivanje začetne in končne lokacije



Slika 4.7: Prikaz prenosne funkcije

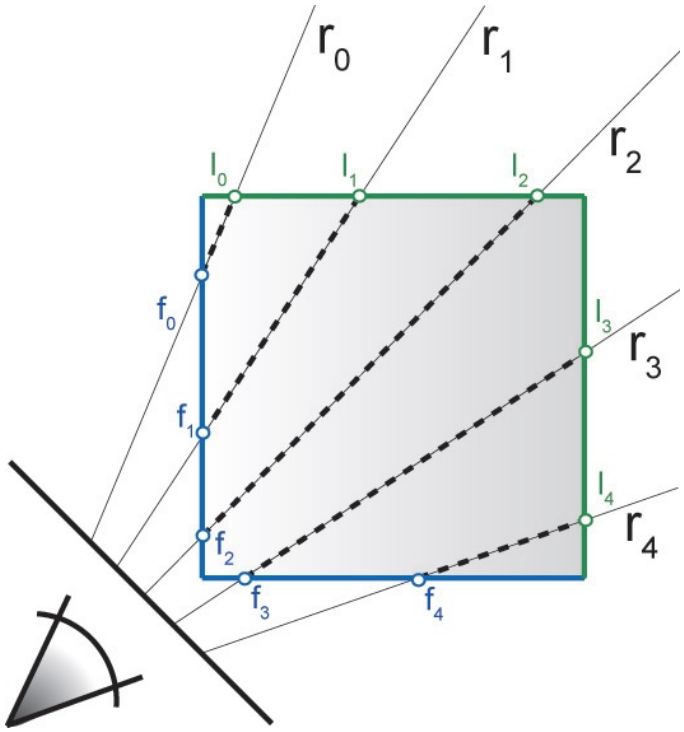
žarka. Na sliki 4.8 lahko vidite dvodimenzionalno predstavitev žarkov, ki sekajo kvadrat. Podobno se dogaja tudi v algoritmu za metanje žarkov, le da tam žarek potuje po tridimenzionalnem prostoru.

Pred začetkom zanke, v kateri bomo posamezen navidezni žarek pomikali po prostoru, moramo inicializirati spremenljivke, ki bodo hranile akumulirano barvo, vrednost alfa kanala in trenutno lokacijo žarka. V sami zanki žarek premikamo s konstantnim korakom, ki smo ga izračunali pred vstopom v zanko. V vsakem koraku iz teksture, ki vsebuje podatke, pridobimo skalarno vrednost v tej točki in jo s pomočjo prenosne funkcije pretvorimo v barvo. To barvo nato zaradi lepšega izrisa popravimo s korekcijskim faktorjem, ki ga lahko nastavimo in se lahko od primera do primera razlikuje, odvisno od vrednosti, ki jih vhodni podatki vsebujejo. Poleg barve za vsak žarek izračunamo in akumuliramo tudi njegovo alfa vrednost, ki določa prozornost v barvnem zapisu RGBA. Zanka, ki navidezno premika žarek po prostoru, se izvaja, dokler ni izpolnjen eden od treh pogojev:

- Prepotovana razdalja žarka je večja ali enaka razdalji od začetka do konca žarka.

- Akumulirana vrednost alfa doseže vrednost 1. Po tem nadaljevanje žarka ni več smiselno, saj imamo že v tej točki popolnoma neprosojno barvo.

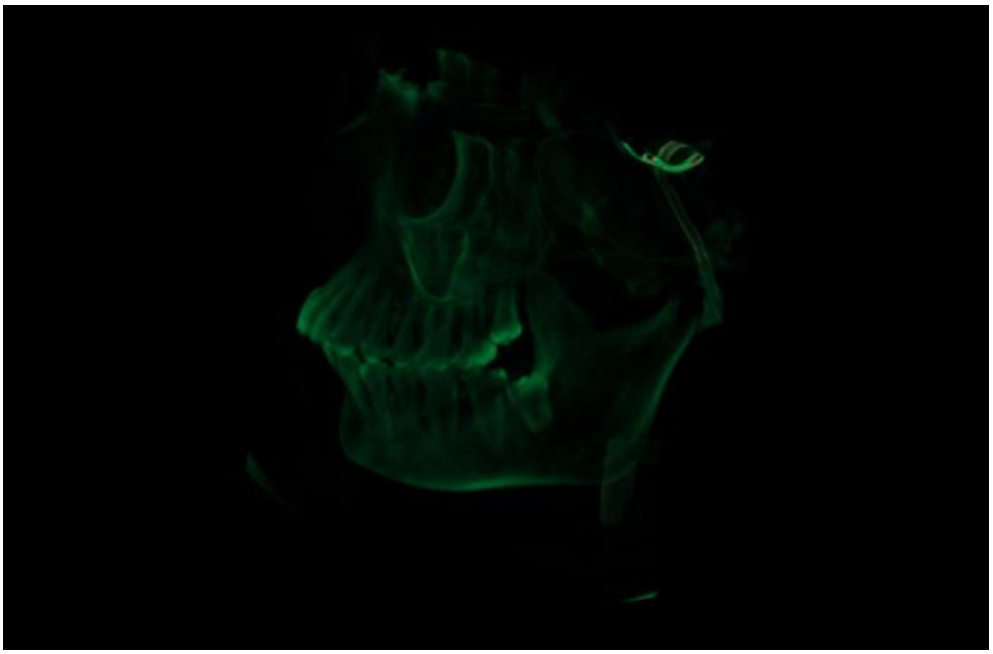
[7] `http://lebarbacom.azurewebsites.net/wp-content/uploads/2014/11/rays.jpg`



Slika 4.8: Izhod prvega para senčilnikov - kocka, kjer so koordinate konca žarkov kodirane v posameznem kanalu RGB barve[7]

- Število iteracij doseže maksimalno število iteracij. To varovalko smo vgradili v implementacijo zato, da ne more priti do situacije, ko je razdalja med posameznimi vzorčenji zelo majhna in bi bilo potrebno preveliko število vzorčenj za vsak žarek, saj bi to bistveno upočasnilo izvajanje programa.

Ko je vsaj eden od treh pogojev izpolnjen, imamo v spremenljivki, ki akumulira barvo, izračunano končno barvo slikovne točke, ki jo ta žarek predstavlja. Ko se opisan postopek izvede za vse slikovne točke, na zaslonu dobimo končno sliko. Primer take vizualizacije je prikazan na sliki 4.9.



Slika 4.9: Primer vizualizacije volumetričnih podatkov z metodo metanja žarkov

Poglavje 5

Evalvacija in rezultati

Glavni rezultat diplomskega dela je nadgradnja spletnega vizualizacijskega ogrodja Med3D za uporabo navidezne resničnosti. Nadgradnja omogoča, da lahko objekte, ki smo jih pred tem s pomočjo ogrodja vizualizirali na računalniškem zaslonu, prikažemo tudi v navidezni resničnosti. Zaradi zasnove samega ogrodja pa bo nadgradnja omogočala tak prikaz tudi novim metodam vizualizacije, ki bodo ogrodju dodana v prihodnosti. Prednost naše implementacije je v tem, da uporabnikom omogoča razmeroma enostaven vstop v pogled navidezne resničnosti, saj za to poleg (prave verzije[1]) brskalnika in operacijskega sistema ne potrebujejo dodatne programske opreme. Tako kot vsak pristop pa ima tudi ta svoje slabosti, saj uporabljene tehnologije ne sodijo med tiste, ki bi omogočale najhitrejše procesiranje podatkov. Zato se lahko pojavijo težave s hitrostjo. Pojav in obseg teh težav pa sta odvisna od zmogljivosti računalnika, ki poganja vizualizacijsko ogrodje, ter od kompleksnosti podatkov, ki jih vizualiziramo.

Poleg implementacije načina pogleda v navidezni resničnosti smo v ogrodje implementirali tudi upravljanje s sceno v tem pogledu. Zaenkrat smo se osredotočili na upravljalnike sistema HTC Vive. V prihodnosti pa bi bilo to podporo dobro razširiti tudi na upravljalnike drugih popularnih sistemov za prikaz navidezne resničnosti.

[1] `https://research.mozilla.org/webvr-compatibility/`

V sklopu diplomskega dela smo poleg zgoraj opisanih nadgradenj ogrodja Med3D implementirali tudi metodo neposrednega upodabljanja z metanjem žarkov. Metoda je primerna za vizualizacijo medicinskih podatkov in tako predstavlja dodatno možnost vizualizacije volumetričnih podatkov. Uporabnik ima pri uporabi te metode na voljo tudi nekaj fleksibilnosti, saj lahko poljubno določa prenosno funkcijo, korekcijski faktor in število vzorčenj žarkov.

Želeli smo preveriti, kako korekcijski faktor in število korakov oz. vzorčenj vplivata na hitrost izrisovanja. Za testiranje smo uporabili prenosni računalnik Lenovo Thinkpad T580 z grafično kartico Nvidia GeForce MX150, 8 GB RAM-a in procesorjem Intel i5. V vseh testih smo uporabili isti vir vhodnih podatkov. Prav tako med samim testiranjem nismo spreminjali orientacije objekta in njegove oddaljenosti od kamere, saj bi to lahko vplivalo na rezultate.

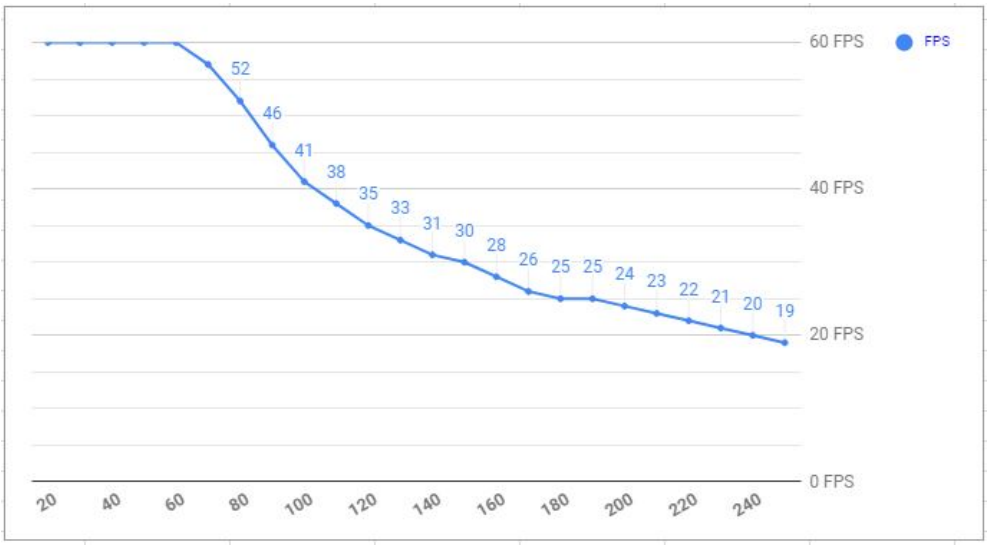
Najprej smo želeli testirati odvisnost hitrosti izrisovanja od števila korakov vzorčenja v algoritmu metanja žarkov. Korekcijsko vrednost alfa smo ves čas testiranja imeli nastavljeno na vrednost 1. Testirati smo začeli pri desetih korakih vzorčenja, pri vsaki naslednji meritvi pa smo število korakov vzorčenja povečali za 10. Testiranje smo izvajali do vrednosti 250 korakov vzorčenja za posamezen žarek. Ob spremembi števila korakov smo vedno počakali, da se je merjena vrednost nehala spreminjati zaradi spremembe v številu korakov. Vsako izmerjeno vrednost smo merili 10 sekund in njeno vrednost zaokrožili na celo število.

Na sliki 5.1 lahko vidite graf, ki prikazuje število izrisov slike na sekundo (angl. FPS - frames per second). Z grafa je lepo razvidno, da število izrisov z naraščanjem števila vzorčenj pada, kar je tudi pričakovan rezultat, saj se z večjim številom vzorčenj poveča računska zahtevnost za izračun končne slike. Pri relativno nizkem številu vzorčenj (manj kot 60) opažamo, da je število izrisov konstantno na 60-ih slikah na sekundo. Razlog tiči v tem, da je ta omejitev umetno določena, saj ob običajni uporabi računalnika ne potrebujemo hitrejšega izrisovanja. Ob ogledu strmega padanja števila izrisanih slik na sekundo lahko sklepamo, da je uporaba te metode primerna za vizua-

lizacije, kjer se pogled ne spreminja večino časa, saj v nasprotnem primeru lahko doživimo slabšo uporabniško izkušnjo zaradi prepočasnega osveževanja slike. Na račun manjšega števila vzorčenj pa je tudi kakovost vizualizacije slabša. Na tej točki mora vsak uporabnik sprejeti kompromis med hitrostjo in kvaliteto izrisovanja slike na zaslon za konkreten primer vizualizacije.

Ob testiranju smo spremljali tudi obremenjenost grafične kartice, procesorja in zasedenost bralno-pisalnega pomnilnika (RAM). Iz opazovanja je jasno vidno, da je hitrost izrisa najbolj odvisna od hitrosti grafične kartice, saj je bila le-ta v vseh primerih, kjer je bilo število vzorčenj večje od 40, obremenjena praktično 100-odstotno. Ostale računalniške komponente med testiranjem niso nikoli dosegle take ravni obremenjenosti.

V splošnem pa iz rezultatov lahko sklepamo, da vizualizacija z večjim številom vzorčenj, ki daje tudi lepše rezultate, za visoko število izrisov na sekundo potrebuje zmogljivejšo grafično kartico. To je pomembno predvsem za predvajanje v navidezni resničnosti, saj tam za dobro uporabniško izkušnjo potrebujemo še višje hitrosti izrisa.



Slika 5.1: Graf hitrosti izrisovanja z algoritmom metanja žarkov v odvisnosti od števila vzorčenj

Poleg vpliva števila vzorčenj vzdolž žarka na čas izrisovanja smo v te-

stih želeli preveriti tudi vpliv spreminjanja korekcijske vrednosti alfa. To vrednost smo spreminjali v območju med 0.1 in 5.0. Teste smo izvedli pri naslednjih vrednostih števila vzorčenj vzdolž posameznega žarka: 50, 100, 150, 200, 250. Rezultate smo zaokrožili na celo število povprečnega števila izrisa slik na sekundo. V nobenem testnem primeru nismo zaznali razlik v rezultatih ob spreminjanju korekcijskega faktorja. Iz tega lahko zaključimo, da ta parameter (v nekem razumnem obsegu, kjer običajno daje dobre rezultate vizualizacije) bistveno ne vpliva na hitrost izrisa volumetričnih podatkov z algoritmom metanja žarkov.

Poglavje 6

Zaključki in nadaljnje delo

V diplomskem delu smo vam predstavili osnove navidezne resničnosti, spletno vizualizacijsko ogrodje Med3D in algoritem metanja žarkov, ki se uporablja za neposredno vizualizacijo volumetričnih podatkov. Vsem opisanim temam je skupna vizualizacija (predvsem medicinskih) podatkov. Ogrodje je bilo že pred tem začetkom mojega dela razvito z namenom širše dostopnosti vizualizacije 3D podatkov, temu pa smo dodali možnost prikaza vizualizacije v navidezni resničnosti prek (ustrezne verzije) spletnega brskalnika. Poleg ogrodja in uporabljenih metod ter tehnologij smo vam poskušali čim bolj predstaviti tudi nekoliko bolj splošen kontekst vizualizacije in razvoja ter trenutnega stanja navidezne resničnosti.

Ta diplomska naloga ne predstavlja začetka razvoja vizualizacijskega ogrodja, vsekakor pa tudi ne njegovega konca. Pri razvoju je še prostor za optimizacijo uporabniške izkušnje uporabe ogrodja v navidezni resničnosti. Poleg optimizacije pa bo v prihodnje potrebno razmisliti tudi o nadgradnji na novi standard WebXR[1], katerega opis je izšel tekom pisanja diplomske naloge. Poleg optimizacije in dokončne integracije že razvitih rešitev obstaja še veliko možnosti za nadgradnjo ogrodja. V podpoglavju 2.1.1 smo našteli nekatere izmed metod, ki se uporabljajo za vizualizacijo podatkov. Implementacija preostalih metod, ki še niso vgrajene v ogrodje, bi predstavljala dobrodošlo

[1] `https://immersive-web.github.io/webxr/`

razširitev ogrodja in povečala njegovo uporabnost. Za zagotavljanje zahtevnejših vizualizacij na manj zmogljivih napravah (mobilni telefoni, tablice, manj zmogljivi računalniki) pa bi bila dobrodošla razširitev obdelave in uporabljanje podatkov na strani strežnika. To bi že tako široko uporabno ogrodje naredilo še bolj dostopno povprečnemu uporabniku.

Literatura

[1] Stacey Abbott. Final frontiers: Computer-generated imagery and the science fiction film. *Science Fiction Studies*, pages 89–108, 2006.

[2] James Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. *The visualization handbook*, 717, 2005.

[3] Peter Armstrong. Magnetic resonance in medicine: The basic textbook of the european magnetic resonance forum, pa rinck (ed.). blackwell scientific publications, uk (1993), 1994.

[4] Mike Bailey and Steve Cunningham. *Graphics shaders: theory and practice*. CRC Press, 2016.

[5] Jules Bloomenthal. Olv. 8. *Graphics gems IV*, 4:324, 1994.

[6] Ciril Bohak, Simon Žagar, Anže Sodja, Peter Škrlj, Uroš Mitrović, Franjo Pernuš, and Matija Marolt. Neck veins : an interactive 3d visualization of head veins. In *Proceedings of the 4th International Conference World Usability Day Slovenia 2013, Ljubljana, Slovenia, 25th November 2013*, pages 64–66, 2013.

[7] Nell Waliczek Brandon Jones. Webxr device api. Dosegljivo: `https://immersive-web.github.io/webxr/`, 2018. [Dostopano 27. 8. 2018].

[8] Luke Buckmaster. If virtual reality is film's next big thing, how long will it take to get right? Dosegljivo: `https://www.theguardian.com/film/2017/oct/06/if-virtual-reality-is-films-next-big-`

thing-how-long-will-it-take-to-get-right, 2017. [Dostopano 7. 7. 2018].

[9] C Burdea Grigore and P Coiffet. *Virtual reality technology*. London: Wiley-Interscience, 1994.

[10] Chaomei Chen. Information visualization, 2002.

[11] Volker A Coenen, Timo Krings, Lothar Mayfrank, Richard S Polin, Marcus HT Reinges, Armin Thron, and Joachim M Gilsbach. Three-dimensional visualization of the pyramidal tract in a neuronavigation system during brain tumor surgery: first experiences and technical note. *Neurosurgery*, 49(1):86–93, 2001.

[12] Carl R Crawford and Kevin F King. Computed tomography scanning with simultaneous patient translation. *Medical Physics*, 17(6):967–982, 1990.

[13] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *ACM Siggraph Computer Graphics*, volume 22, pages 65–74. ACM, 1988.

[14] Michael Friendly. A brief history of data visualization. In *Handbook of data visualization*, pages 15–56. Springer, 2008.

[15] Azucena Garcia-Palacios, Hunter Hoffman, Albert Carlin, TA Furness Iii, and Cristina Botella. Virtual reality in the treatment of spider phobia: a controlled study. *Behaviour research and therapy*, 40(9):983–993, 2002.

[16] Ian P Howard and Brian J Rogers. *Binocular vision and stereopsis*. Oxford University Press, USA, 1995.

[17] Internet world stats. Dosegljivo: https://www.internetworldstats.com/stats.htm, 2018. [Dostopano: 9. 7. 2018].

[18] David Jack, Rares Boian, Alma S Merians, Marilyn Tremaine, Grigore C Burdea, Sergei V Adamovich, Michael Recce, and Howard Poizner. Virtual reality-enhanced stroke rehabilitation. *IEEE transactions on neural systems and rehabilitation engineering*, 9(3):308–318, 2001.

[19] Sebastien Jourdain, Utkarsh Ayachit, and Berk Geveci. Paraviewweb, a web framework for 3d visualization and data processing. In *IADIS international conference on web virtual reality and three-dimensional worlds*, volume 7, page 1, 2010.

[20] Willi A Kalender, Wolfgang Seissler, Ernst Klotz, and Peter Vock. Spiral volumetric ct with single-breath-hold technique, continuous transport, and continuous scanner rotation. *Radiology*, 176(1):181–183, 1990.

[21] Hannes Kaufmann, Dieter Schmalstieg, and Michael Wagner. Construct3d: a virtual reality application for mathematics and geometry education. *Education and information technologies*, 5(4):263–276, 2000.

[22] D Krakow, J Williams, M Poehl, DL Rimoin, and LD Platt. Use of three-dimensional ultrasound imaging in the diagnosis of prenatal-onset skeletal dysplasias. *Ultrasound in obstetrics & gynecology*, 21(5):467–472, 2003.

[23] Thomas Kroes, Frits H Post, and Charl P Botha. Exposure render: An interactive photo-realistic volume rendering framework. *PloS one*, 7(7):e38586, 2012.

[24] Sun-Yuan Kung. Visualization of big data. In *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015 IEEE 14th International Conference on*, pages 447–448. IEEE, 2015.

[25] Philippe Lacroute and Marc 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*, pages 451–458. ACM, 1994.

[26] Hongzhi Lan, Jameson Merkow, Adam Updegrove, Daniele Schiavazzi, Nathan Wilson, Shawn Shadden, and Alison Marsden. Simvascular 2.0: an integrated open source pipeline for image-based cardiovascular modeling and simulation. In *APS Division of Fluid Dynamics Meeting Abstracts*, 2015.

[27] Primož Lavrič. Spletno ogrodje za vizualizacijo volumetričnih podatkov z možnostjo oddaljenega sodelovanja. Diplomska naloga, Fakulteta za računalništvo in informatiko, 2016.

[28] Marc Levoy. Display of surfaces from volume data. *IEEE Computer graphics and Applications*, (3):29–30, 1988.

[29] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[30] Matija Marolt. RGTI predavanja, Grafični cevovod. Dosegljivo: `https://ucilnica1617.fri.uni-lj.si/pluginfile.php/49478/mod_resource/content/0/17%20Graficni%20cevovod.pdf`, 2016. [Dostopano 28. 6. 2018].

[31] Joan McComas, P Pivik, and Marc Laflamme. Current uses of virtual reality for children with disabilities. *Studies in health technology and informatics*, pages 161–169, 1998.

[32] John McHale. Training in a virtual world is cost effective. Dosegljivo: `http://mil-embedded.com/articles/training-a-virtual-world-cost-effective/`, 2012. [Dostopano 28. 6. 2016].

[33] Maj. Loren Mer. Virtual reality used to train soldiers in new training simulator. Dosegljivo: `https://www.army.mil/article/84453/virtual_reality_used_to_train_soldiers_in_new_training_simulator`, 2012. [Dostopano 7. 7. 2018].

[34] Flagship Project MIRACLE. Dosegljivo: https://www.dbe.unibas.ch/en/research/flagship-project-miracle/, 2018. [Dostopano: 20. 7. 2016].

[35] John M Ollinger and Jeffrey A Fessler. Positron-emission tomography. *IEEE Signal Processing Magazine*, 14(1):43–55, 1997.

[36] Lois Rosson. The Virtual Interface Environment Workstation. Dosegljivo: https://www.nasa.gov/ames/spinoff/new_continent_of_ideas/, 2014. [Dostopano 7. 7. 2018].

[37] Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

[38] Barbara Olasov Rothbaum, Larry Hodges, Renato Alarcon, David Ready, Fran Shahar, Ken Graap, Jarrel Pair, Philip Hebert, Dave Gotz, Brian Wills, et al. Virtual reality exposure therapy for ptsd vietnam veterans: A case study. *Journal of Traumatic Stress: Official Publication of The International Society for Traumatic Stress Studies*, 12(2):263–271, 1999.

[39] Takashi Shibata. Head mounted display. *Displays*, 23(1-2):57–64, 2002.

[40] Robert J. Simpson. The opengl es® shading language. Dosegljivo: https://www.khronos.org/registry/OpenGL/specs/es/3.0/GLSL_ES_Specification_3.00.pdf, 2016. [Dostopano: 25. 8. 2018].

[41] Virtual Reality in Medicine: New Opportunities for Diagnostics and Surgical Planning. Dosegljivo: https://www.unibas.ch/en/News-Events/News/Uni-Research/Virtual-Reality-in-Medicine.html, 2018. [Dostopano: 20. 7. 2018].

[42] Gábor Székely and Richard M Satava. Virtual reality in medicine. *BMJ: British Medical Journal*, 319(7220):1305, 1999.

[43] Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. In *Volume Visualization, 1996. Proceedings., 1996 Symposium on*, pages 23–30. IEEE, 1996.

[44] Virtual reality sickness. Dosegljivo: `https://en.wikipedia.org/wiki/Virtual_reality_sickness`, 2018. [Dostopano: 26. 8. 2018].

[45] Nicholas J Wade. On the late invention of the stereoscope. *Perception*, 16(6):785–818, 1987.

[46] Lee Alan Westover. *Splatting: a parallel, feed-forward volume rendering algorithm*. PhD thesis, University of North Carolina at Chapel Hill Chapel Hill, NC, 1991.

[47] Charles Wheatstone. Xviii. contributions to the physiology of vision.—part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical transactions of the Royal Society of London*, 128:371–394, 1838.

[48] Helena Wong. CS3162 Introduction to Computer Graphics - 9. 3D Object Representations. Dosegljivo: `http://www.cs.cityu.edu.hk/~helena/cs31622000B/Chap9Notes.pdf`, 2001. [Dostopano 28. 6. 2016].