Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jan Aleksandrov

Integracija sistema za upodabljanje na podlagi sledenja potem v ogrodje Med3D

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

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

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu preučite delovanje vizualizacijskega ogrodja Med3D in se spoznajte s principi upodabljanja volumetričnih podatkov. V ogrodje integrirajte sistem za upodabljanje na podlagi sledenja potem in s tem omogočite kombinirano upodabljanje s posrednimi in neposrednimi pristopi. Nadgradite uporabniški vmesnik aplikacije z zmožnostmi izbire konfiguracij upodabljanja. Aplikacijo ovrednotite s stališča učinkovitosti.

Hvaležen sem članom Laboratorija za računalniško grafiko in multimedije na FRI. Posebna zahvala so-mentorju as. dr. Cirilu Bohaku za vložen čas in potrpežljivost. Hvaležen sem tudi svoji družini, ki mi je omogočila nemoten študij.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Problem in rešitev 2

1.2 Cilji 3

1.3 Struktura dela 4

2 Pregled področja **5**

2.1 Volumetrični podatki 6

2.2 Pregled metod upodabljanja volumetričnih podatkov 7

2.3 Obstoječa orodja 12

2.4 Med3D 14

2.5 VPT 17

2.6 Uporabljene tehnologije 20

3 Implementacija **25**

3.1 Grafični cevovod 25

3.2 Uporabniški vmesnik 34

3.3 Oddaljeno sodelovanje 40

4 Analiza uspešnosti **43**

4.1 Upodabljanje volumetričnih podatkov s kombiniranim pristopom 43

4.2 Uporabniški vmesnik 51

4.3 Deljenje parametrov v oddaljenemu sodelovanju 54

5 Zaključek **55**

5.1 Nadaljnje delo 56

Literatura **56**

Seznam uporabljenih kratic

kratica	**angleško**	**slovensko**
2D	two dimensional	dvodimenzionalen
3D	three dimensional	tridimenzionalen
GPE	graphics processing unit	grafična procesna enota
CPE	central processing unit	centralno procesna enota
HTML	hyper text markup language	jezik za označevanje nadbesedila
CSS	cascading style sheets	kaskadne stilske podloge
MPUI	multi-level partition of unity implicits algorithm	Algoritem implicitnih funkcij z večnivojsko particijo enote

Povzetek

Naslov: Integracija sistema za upodabljanje na podlagi sledenja potem v ogrodje Med3D

Avtor: Jan Aleksandrov

Diplomsko delo predstavlja nadgradnjo spletne aplikacije Med3D z novim pristopom vizualizacije medicinskih volumetričnih podatkov. Za vizualizacijo podatkov smo razvili način kombiniranega upodabljanja. Ta združuje prednosti klasičnega posrednega pristopa in bolj fizikalno natančnega neposrednega pristopa, pri čemer smo uporabili metodo sledenja potem, razvito v sklopu magistrskega dela *Sledenje poti za neposredno upodabljanje volumetričnih podatkov s spletnimi tehnologijami* [26].

Uporabniški vmesnik smo dopolnili z interaktivnimi funkcijami, katere omogočajo veliko možnih konfiguracij upodabljanja, ki so vključene tudi v sistem oddaljenega sodelovanja uporabnikov. V vrednotenju smo opravili analizo, ki pokaže smiselnost izbranega pristopa s stališča učinkovitosti in uporabniških zmogljivosti.

Rezultat je platformno agnostična aplikacija, ki temelji na najnovejših spletnih tehnologijah, omogoča napredne metode upodabljanja in nudi nov nivo uporabniške izkušnje.

Ključne besede: WebGL, 3D grafika, sledenje potem, upodabljanje volumetričnih podatkov, oddaljeno sodelovanje.

Abstract

Title: Integration of the path tracing rendering system into the Med3D application

Author: Jan Aleksandrov

Thesis presents an upgrade to the existing web framework Med3D with a new approach to medical volumetric data visualization. Developed approach combines benefits of classic indirect mesh based rendering with more complex direct volumetric rendering. For latter we used path-tracing based methods, developed in master's thesis *Path tracing for direct volume rendering with web technologies* [26].

User interface has been extended with new interactive features, that provide vast rendering possibilities, all of which are also available for sharing in online collaboration sessions. Our work is backed up by analysis of rendering performance and user experience.

The result is a platform independent web application that makes use of state-of-the-art web technologies with the latest advancements in volumetric rendering and offers user experience on a new level.

Keywords: WebGL, 3D graphics, path tracing, volumetric data rendering, remote collaboration.

Poglavje 1

Uvod

Področje računalniške grafike se razvija izjemno hitro. Še pred nekaj leti so bile 3D upodobitve na videz bistveno preprostejše in manj realistične kot danes. Razvoj novih metod je odvisen od napredka na več področjih, ki se razvijajo sočasno.

Strojna oprema postaja vse močnejša, z njo mogoče je zajeti več podatkov ter jih hitreje obdelovati. Metode za izris v realnem času so že tako natančne, da je njihov rezultat težko ločiti od resničnosti.

Ključna programska oprema se hitro nadgrajuje z novimi funkcionalnostmi ali pa jo zamenjajo nove, boljše rešitve. Spletne tehnologije nekdaj niso omogočale dobre izrabe strojnih zmogljivosti, zato se je gradilo zahtevnejše aplikacije kot samostojne celostne rešitve. Z novejšimi brskalniki se je bistveno povečal nabor zmogljivosti, moderne spletne tehnologije, kot so WebGL[1], NodeJS[2] in AngularJS[3] pa bistveno poenostavijo razvoj kompleksnih spletnih storitev in aplikacij.

Izdelava oziroma zajem podatkov se je prav tako bistveno spremenila. Še v ne tako daljni preteklosti je bil zajem sveta omejen na dvodimenzionalne slike, z razvojem računalniške grafike [41], optike, računalniške tomografije in

[1] *OpenGL ES for the Web.* Dosegljivo: `https://www.khronos.org/webgl/`. [Dostopano: 25. 2. 2019].

[2] *Node.js.* Dosegljivo: `https://nodejs.org/en/`. [Dostopano: 25. 2. 2019].

[3] *AngularJS.* Dosegljivo: `https://angularjs.org/`. [Dostopano: 25. 2. 2019].

metod obdelave in shranjevanja podatkov, pa je zdaj mogoče posneti tridimenzionalno stanje sveta. Takšne slike lahko temeljijo na zapisu mreže poligonov ali pa kot zapis točk z različnimi vrednostmi v 3D prostoru. Slednjemu obliki pravimo zapis volumetričnih podatkov.

Nove metode upodabljanja so posledica težnje po bolj učinkovitemu ali pa natančnejšemu izrisu scene. V nekaterih primerih metode lahko postanejo aktualne šele, ko strojne zmogljivosti omogočajo uporabno izvedbo. Takšen primer so metode sledenja potem, ki lahko realistično in fizikalno pravilo osvetlijo sceno. Take metode so bile za upodobitve v realnemu času včasih povsem neuporabne, danes pa se z omejeno zmogljivostjo ponekod že uporabljajo.

V računalništvu se meja mogočega hitro premika. Glede na splošno razširjenost uporabe računalništva pri skoraj vseh vsakdanjih opravilih, lahko varno predpostavimo, da bi vsaka novost pritegnila veliko zanimanja pri uporabnikih.

1.1 Problem in rešitev

Vse pogostejša raba volumetričnih podatkov pomeni, da obstaja zanimanje za hitre, interaktivne in realistične pristope njihove vizualizacije. Zaradi kompleksnosti vizualizacijskih metod in visoke izrabe računskih virov, na tem področju še ni enovitih vsestranskih rešitev. Zato obstaja realna priložnost izdelave uporabniške aplikacije, ki bo reševala problematiko izbranega segmenta uporabnikov in bo naslovila pomanjkljivosti obstoječih rešitev:

- *Platformna odvisnost*, omejitev na bodisi določen operacijski sistem, določene strojne zahteve ali pa na obliko podatkov.

- *Omejitev kakovosti izrisa* v primerih, ko sistem ni zmožen predstaviti ključnih informacij. Upodobitev volumetričnih podatkov ni enolična, saj različne metode in parametri prikazujejo različen spekter informacij o slikanemu objektu. Poleg tega morajo biti metode zadosti napredne

in dodelane, da v vizualizacijo ne vnašajo napak. Želimo fizikalno pravilnost. Visoke zahteve lahko vodijo v neizvedljivost upodobitve v realnemu času ali pa potrebo po nadpovprečno zmogljivi strojni podpori, ki ni dosegljiva večini uporabnikov.

- *Uporabniška izkušnja* je ključnega pomena za zadovoljstvo uporabnikov. Ključni vidiki dobre uporabniške izkušnje so, dobra interaktivnost, širok nabor možnosti, intuitivnost in jasnost.

Z omenjenimi problemi so se študenti Fakultete za računalništvo in informatiko Univerze v Ljubljani že spopadli. Ključna sta dva izvedena projekta. Prvi je spletna aplikacija Med3D [25] z naprednim grafičnim cevovodom za upodabljanje objektov poligonskih mrež, visoko interaktivnostjo in pestrim naborom uporabniških funkcij. Drug projekt je ogrodje Volumetric Path Tracing - VPT [26], ki omogoča neposreden izris volumetričnih podatkov z različnimi fizikalno osnovanimi metodami.

1.2 Cilji

Naša naloga je torej združiti dobre lastnosti obeh projektov, bolj natančno vključiti metode iz VPT-ja [26] v ogrodje Med3D [25]. Namen in motivacija je razširitev zmogljivosti Med3D-ja, tako da bo nudil večjo funkcionalnost in širši nabor vizualizacijskih tehnik, s tem pa naslovil problematiko širše skupine uporabnikov.

Končen izdelek mora dosegati sledeče cilje:

- Smiselna vključitev metod VPT v Med3D cevovod.
- Implementacija mora smiselno razpolagati z zmogljivosti, ki so ji namenjene. To pomeni, da kakovost in hitrost upodobitve ne sme zaostajati v primerjavi s predstavitveno implementacijo VPT.
- Aplikacija mora ostati platformno neodvisna.

- Uporabniška izkušnja mora slediti načelom dobre interaktivnosti.
- Uporabniške funkcije morajo biti usmerjene k profesionalni rabi.

1.3 Struktura dela

V sledečemu poglavju je pregled področja. Začne se s splošnim pregledom razvoja računalniške grafike, sledi kratek pregled načinov dela z volumetričnimi podatki. Nato raziščemo zmogljivosti obstoječih rešitev. Ključni sta ogrodji Med3D in VPT, zato zanju podamo natančnejšo analizo delovanja in nabor obstoječih zmogljivosti. Konča se s pregledom uporabljenih spletnih tehnologij, ki jih koristimo pri implementaciji, razloženi v tretjemu poglavju

V četrtem poglavju opravimo vrednotenje rezultatov s performančno analizo in oceno uporabniških zmožnosti in omejitev.

Poglavje 2

Pregled področja

Vizualizacija je vsaka akcija, ki ustvarja slike, diagrame in animacije z namenom prenosa sporočil [58]. Sposobnost vizualizacije je sestavni del človeka. Prve pojavitve, v obliki slik na jamskih stenah, več deset tisoč let nazaj [46], sovpadajo z evolucijskim razvojem v modernega človeka. Skozi čas in širitev kolektivnega človeškega znanja, postaja vizualizacija vse pomembnejša. Surove podatke je bistveno lažje razumeti in pomniti v interpretirani vizualni obliki. Nastanek znanstvenih ved, kot so kartografija in astronomija v antičnih civilizacijah [11] temelji na vizualizaciji takšnih podatkov.

Večji preskok se je zgodil z hitrim razvojem računalništva v 20. stoletju. Metode simuliranja, merjenja in modeliranja so se premaknile v digitalno obliko, količina razpoložljivih podatkov je eksplozivno narasla, posledično se je tudi vizualizacija prestavila v domeno računalništva. Računalniška podpora omogoča nove bolj kompleksne metode slikovne in video predstavitve. Področje vizualizacije z računalniško podporo se imenuje računalniška grafika. Ta pojem se začne uporabljati od leta 1960 dalje [4]. V istemu obdobju se začnejo aktivno razvijati metode 3D grafike. Ideja 3D grafike je preslikava podatkov iz navideznega 3D sveta na 2D platno. Klasični pristop zapisa podatkov je s koordinatami oglišč, skupek oglišč pa predstavlja nek objekt. V sedemdesetih letih so se razvile prve metode senčenja, ki omogočajo predstavitev globine objekta s spreminjanjem barv glede na svetlobno osvetlitev. Prva

znana primera sta Gouraudovo senčenje [14] leta 1971 in Blinn-Phongovo [3] leta 1977. Nadaljnji razvoj metodologije prinese vizualno bolj avtentične upodobitve, ki pa zahtevajo sorazmerno večjo računsko moč. V osemdesetih in devetdesetih letih se pojavijo prvi splošnonamenski strojni 2D in 3D pospeševalniki [56]. Razvoj se usmeri v pohitritev spominsko-intenzivnih ukazov, povezanih s teksturami, in pohitritev izračunov matričnih operacij z dodajanjem grafičnih enot in vpeljavo paralelizacije.

2.1 Volumetrični podatki

Uporaba in zajem volumetričnih podatkov se uveljavi na področju medicine. Od sedemdesetih let 20. stoletja dalje so zdravnikom na voljo tehnologije, ki omogočajo tridimenzionalen posnetek stanja, za posamezne točke objekta, pacienta. Prve takšne tehnologije so bile *računalniška tomografija* (angl. computed tomography) [39, 6], *magnetna resonanca* (angl. magnetic resonance imaging) [40] ter *3D ultrazvok* (angl. 3D ultrasound) [21].

Običajno zajem poteka tako, da se ciljno tkivo pod različnimi koti obseva z različnimi vrstami valovanja, ki se posname ob izhodu iz objekta. Z rekonstrukcijskimi algoritmi [28] se izračunajo 2D slike preseka objekta. Z združitvijo zaporednih presečnih slik pa nastane volumen. Vsak piksel 2D slike predstavlja voksel v volumnu. Tak zapis so t. i. skalarne mreže, kjer so podatki enakomerno vzorčeni vzdolž treh medsebojno ortogonalnih osi. Upodobitev volumetričnih podatkov ima v primerjavi s planarnimi slikami bistveno višjo informacijsko vrednost [10]. Z dodano globino je omogočen prikaz pod različnimi koti. Tako je možno razločiti nove strukture, ki morda niso vidne na 2D slikah in pa obratno, izločiti lažne zaznave kot posledice šuma v posameznih posnetkih. Ravno tako pomembna je tudi natančna ocena velikosti in delovanja najdenih struktur. To so ključne informacije za ocenitev zdravstvenega stanja pacienta in pripravo nadaljnje oskrbe.

Velikost takšnih podatkov je odvisna od natančnosti zajema in naknadne obdelave, običajno pa so to velike datoteke, od nekaj 10 MB do več GB.

Obdelava velike količine podatkov pa je vedno zahtevna. Nekaj znanih metod upodabljanja je opisanih v naslednjemu podpoglavju, splošno pa velja, da je bolj kakovostna upodobitev tudi računsko bolj zahtevna.

2.2 Pregled metod upodabljanja volumetričnih podatkov

V osnovi se volumetrično upodabljanje ne razlikuje bistveno od upodabljanja poligonskih mrež. V obeh primerih so vhodni podatki zapisi točk, s položajem v lastnemu koordinatnemu sistemu, ki jih je potrebno preslikati v koordinatni sistem zaslona. V primeru volumetričnih mrež so točke voksli, položaj je definiran glede na lokacijo v skalarni mreži, vrednost voksla pa predstavlja zajeta intenziteta. Pri poligonskih mrežah so točke oglišča geometrije, vrednost pa določa njihov položaj. V koraku upodabljanja sicer ni smiselno vsakega voksla izrisovati ločeno. Potrebna je metoda, ki določa izbiro, videz in način upodobitve zajetih intenzitet. Skupna lastnost takšnih metod je, da je mogoče njihovo notranje izvajanje paralelizirati. Vsak paralelni klic se izvaja na ločenem odseku podatkov v skalarni mreži. To je pomembna optimizacija, saj so metode na splošno nadpovprečno računsko zahtevne. To še posebej velja za metode neposrednega upodabljanja, kjer je izraba strojne opreme, namenjene paralelnemu računanju, posledično ključna. Preden so postali *cenovno dostopni* (angl. consumer grade) grafični pospeševalniki zadosti zmogljivi, je bilo v ta namen potrebno razvijati namensko strojno opremo za visoko računsko paralelizacijo [59]. Znan tak primer je sistem VolumePro [36] iz leta 1999. Paralelizacija pa je možna tudi v primerih posrednega upodabljanja v koraku priprave poligonskih mrež.

2.2.1 Posredno upodabljanje

Posredno upodabljanje je način upodabljanja volumetričnih podatkov z uporabo poligonskih mrež. Metode posrednega upodabljanja se razlikujejo v

postopku priprave poligonske mreže. Poligonske mreže so za upodobitev manj zahtevne od metod neposrednega upodabljanja. Slabost pa je, da se aktivnost vokslov enolično določi glede na prag intenzitete, ki je vhodni parameter. S tem se lahko izgubi informacija o vsebini objekta. Vsakemu spreminjanju praga pa sledi ponovno računanje aktivnih vokslov in vsaj dela poligonske mreže. Natančnost takšnih mrež je pogojena s številom poligonov, ki pa lahko, glede na vhodni prag, naraste do količine, ki je za upodobitev bolj zahtevna od metod neposrednega upodabljanja. Predstavljene so tri pogosto uporabljene metode:

- *Metoda Marching cubes* [27] je bila predstavljena leta 1987. Potek algoritma je v osnovi obhod celega 3D skalarnega polja. V vsakemu koraku si predstavljamo kocko z oglišči v osmih sosednjih vokslih. Glede na prag intenzitete so oglišča z voksli aktivna ali pa ne. Takšnih konfiguracij je 2^8. Z upoštevanjem simetrij vseh možnih konfiguracij ostane 16 unikatnih primerov. Primer z vsemi aktivnimi in primer z nobenim aktivnim ogliščem se enači, zato je skupno 15 unikatnih konfiguracij. Vsaka konfiguracija enolično določa preslikavo namišljene kocke v skupek poligonov. Poligonski skupki se seštevajo v večjo poligonsko mrežo, položaj v prostoru pa je določen s položajem vokslov v skalarni mreži.

- *Metoda Marching tetrahedra* [7] je bila razvita leta 1991 kot izboljšava metode Marching cubes. Potujoča kocka je tokrat razdeljena na šest tetraedrov. Ta delitev je dosežena z prerezom kocke po eni diagonali za vsak par stranic. Glede na aktivnost vokslov v ogliščih ima vsak tetraeder 16 možnih konfiguracij. Skupno to omogoči boljši rezultat vzorčenja, a za ceno višje računske in spominske zahtevnosti.

- *Algoritem implicitnih funkcij z večnivojsko particijo enote* (angl. Multi-level Partition of Unity Implicits - MPUI) [34] je bil objavljen leta 2003. Ideja algoritma je rekurzivna delitev oblaka orientiranih točk na particije. Particije so hierarhično povezane v osmiških drevesih. Nadaljevanje rekurzivne delitve je odvisno od kompleksnosti oblike, ki

jo predstavljajo podatki particije. V končnih listih drevesa se podatki particij aproksimirajo z lokalnimi implicitnimi funkcijami. Rezultati teh funkcij se združujejo glede na vsote lokalnih utežnih funkcij, vse do korena drevesa.

Prednost algoritma je bolj natančna aproksimacija poligonske mreže, predvsem pri ostrih kompleksnih oblikah. Računska zahtevnost je odvisna od kompleksnosti vhodnih podatkov, povprečno pa je prezahtevna za uporabo v brskalniku.

2.2.2 Neposredno upodabljanje

Bistvo metod neposrednega upodabljanja je neposredna preslikava informacij posameznih vokslov v barvno vrednost na izračunani lokaciji. Na kratko je predstavljenih nekaj znanih pristopov neposrednega upodabljanja:

- *Splatting* [55] je tehnika, opisana leta 1991. Vsak voksel se glede na pogled kamere, v zaporedju po globini, od najbolj oddaljenega do najbližjega, razmaže na platno. Končna barva je odvisna od uteži naloženih 2D zmazkov. Tehnika je hitra, a manj natančna.

- *Upodabljanje na osnovi tekstur* [15] je tehnika opisana leta 1989. Temelji na upodabljanju volumetričnih podatkov v obliki tekstur. Grafični pospeševalniki omogočajo hitro vzorčenje in preslikavo tekstur, kar metodi omogoča hitro delovanje in možnost interaktivnosti v realnem času. Podatki se v senčilnik naložijo kot 3D tekstura. 3D teksture na starejših grafičnih pospeševalnikih niso nujno podprte. V takih primerih se pripravi več zaporedno zloženih 2D tekstur. Vhodne teksture imajo lahko statično rotacijo vezano na volumen. V tem primeru se upodabljajo pod kotom, glede na kamero. Lahko pa imajo rotacijo vezano na ravnino kamere. Slabša lastnost so artefakti, med zloženimi teksturami, pokažejo pa se pri rotaciji volumna.

- *Shear-Warp* [23] je tehnika, razvita leta 1994. Tehnika nadomesti korak rotacije vokslov z *zamikom* (angl. shearing) rezin volumna. Po koraku

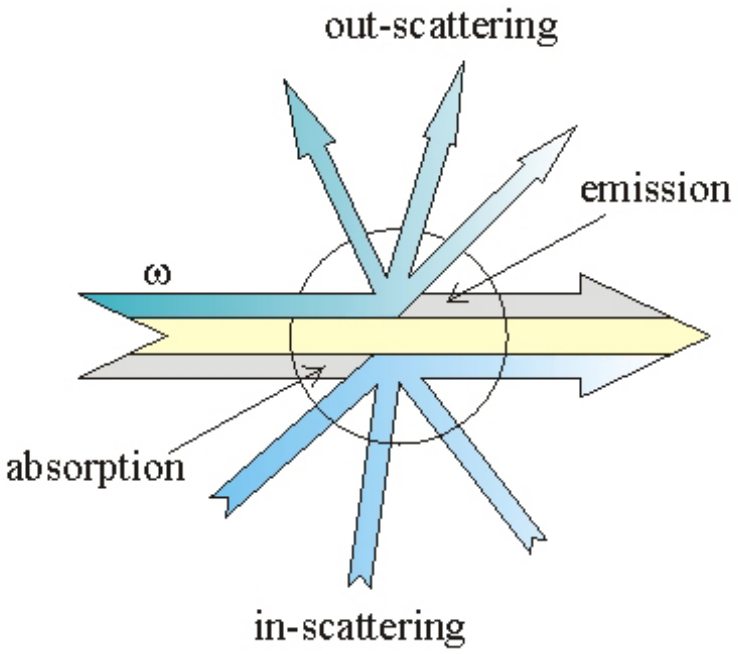
izrisa zamaknjenih rezin v slikovni okvir se popačena upodobitev popravi v koraku *ukrivljanja* (angl. warping). Tehnika je hitra, a manj natančna zaradi dvakratnega vzorčenja.

- *Tehnike žarkov* [20, 32] predstavljajo širok razpon metod z začetkom v letu 1988 z delom [8]. Tehnike poskušajo na različne načine slediti fizikalnim zakonom svetlobe. Zapis pravil je predstavljen leta 1986 z *enačbo upodabljanja* (angl. rendering equation) [19].

  V osnovni različici, *metanja žarkov* (angl. ray casting) gre za pošiljanje žarkov skozi vsak piksel izrisne ravnine. Žarek potuje skozi volumetrične podatke, kjer se na enakomernih ali pa dinamičnih intervalih vzorči. Vzorčenje na vsakemu koraku predstavlja barvni in prosojni (RGBA) prispevek, ki se akumulira do izhoda žarka iz volumna. Izvorni piksel z barvo predstavlja akumulirano vrednost.

  Najbolj napreden pristop je *sledenje potem* (angl. path tracing), ki natančneje sledi enačbi upodabljanja s simuliranjem svetlobnih pojavov *absorbcije* (angl. absorption), *sipanja* (angl. scattering) in *emisij* (angl. emission). Pojavi so vizualizirani na sliki 2.1.

  Postopek je vzorčenje razdalje poti žarka v mediju. Večja razdalja in višja intenziteta medija pomenita večjo verjetnost omenjenih pojavov absorbcije in sipanja za fotone svetlobnega žarka. Simuliranje pojavov povečuje in zmanjšuje količino žarkov, ki jim sledimo, v mediju. To se ponavlja dokler žarek ne zadane vira svetlobe. Generirana pot se uporabi za oceno sevalnosti. Rezultati med iterativnimi simulacijami za isti žarek so precej naključni. Za ocenitev deterministične vrednosti se uporabljajo stohastične metode, na primer Monte Carlo. Ocenjena vrednost se preslika v barvno vrednost izvornega piksla tega žarka.



Slika 2.1: Absorbcija, emisija in sipanje. (Vir: povzeto po [61].)

Skozi čas se razvije nemalo novih izvedb, osnovanih na enakemu principu. Išče se najboljše razmerje med kakovostjo in računsko zahtevnostjo. To so na primer *dvosmerno sledenje poti* (angl. bidirectional path tracing - BDPT) [24] in generiranje novih poti svetlobe z uporabo Metropolis-Hastings algoritma [47].

2.3 Obstoječa orodja

Za medicinske namene je kakovost upodobitve zelo pomembna. Kot smo omenili, so se za namene profesionalne računalniške vizualizacije ločeno razvijali namenski sistemi. Primer takšne rabe so medicinska, vojaška in filmska vizualizacija. Ponudniki takšnih rešitev pa so razvijali lastno strojno opremo, operacijske sisteme in uporabniške programe. Znan primer takšnega ponudnika, ki spada pod vse naštete primere, je Sillicon Graphics[1]. Posledično je bila cena takšnih sistemov zelo visoka. Danes je takšen pristop manj aktualen. Še vedno pa je možno kupiti te stare sisteme[2] kot rezervne dele za organizacije, ki so obstale na takšnih, tudi več deset let starih platformah.

Zdaj je bolj razširjen pristop z uporabo splošno namenske računalniške opreme. Čeprav je ta zdaj bistveno močnejša, kot so bili namenski sistemi v začetku 21. stoletja, je še vedno potreben kompromis med kakovostjo, ceno, hitrostjo izrisa in implementiranimi funkcionalnostmi.

V nadaljevanju predstavimo nekaj modernih odprtokodnih orodij, ki si prizadevajo doseči podobne cilje, kot smo si jih zastavili za Med3D.

- *ImageVis3D* [9] je rešitev za skalabilno, interaktivno vizualizacijo volumetričnih podatkov. Pripravljena je posamično za najbolj pogoste platforme, tudi iOS, temelji pa na OpenGL[3] tehnologiji. V primerjavi z Med3D omogoča vizualizacijo bistveno večjih vhodnih podatkov. Problem omejitve spomina pa rešuje s predprocesiranjem, kjer podatke razkosa na manjše dele. Za upodabljanje uporablja metode, osnovane na delu s teksturami, in metode metanja žarkov s podporo grafičnega pospeševanja. Ne omogoča pa oddaljenega sodelovanja in je platformno odvisna aplikacija.

[1] *High-Performance Computing Solutions.* Dosegljivo: `www.sgi.com`. [Dostopano: 27. 2. 2019], *Indigo2 IMPACT: Medical Imaging.* Dosegljivo: `http://www.futuretech.blinkenlights.nl/medical.html`. [Dostopano: 27. 2. 2019].

[2] *SGI Depot.* Dosegljivo: `http://www.sgidepot.co.uk/sgidepot/`. [Dostopano: 27. 2. 2019].

[3] *OpenGL - The Industry's Foundation for High Performance Graphics.* Dosegljivo: `https://www.opengl.org/`. [Dostopano: 27. 2. 2019].

- *VTK*[4] je podporno orodje, ki se razvija že od leta 1993. Omogoča strojno pospeševanje za manipulacijo in vizualizacijo znanstvenih podatkov, tudi volumetričnih. Namenjeno je samostojni rabi ali pa integraciji v obliki knjižnice. Od leta 2017 se razvija tudi različica za spletne tehnologije, *vtk.js*[5]. Podpira upodabljanje z metodo metanja žarkov.

- *SimVascular*[6] je namizna aplikacija za vizualizacijo medicinskih primerov. Posebej se specializira za simulacijo ožilnih pretokov. Podpira tudi delo z volumetričnimi podatki, za upodabljanje pa uporablja podporo orodja VTK.

- *Exposure Render* [22] je namizna aplikacija, ki temelji na računski platformi CUDA [30] na Windows operacijskih sistemih. Poslužuje se funkcij orodja VTK, ampak implementira svoj upodabljalnik z metodo sledenja žarkov z Monte Carlo stohastičnim vzorčenjem.

- *Ami.js* je precej nova spletna knjižnica iz leta 2017. Namenjena je vizualizaciji medicinskih volumetričnih podatkov. Omogoča pregled prečnih presekov in upodabljanje z metodama projekcije maksimalne intenzitete in sledenju potem. Za 2D prečne preseke omogoča celo postavitev anotacij. V primerjavi z VPT, ki je predstavljen v nadaljevanju, ima ožji nabor konfiguracij upodabljanja.

Predstavitev podobnih orodij kaže na to, da se razvoj počasi premika v smeri platformne neodvinosti in uporabe spletnih tehnologij. V primeru *Ami.js* in *vtk.js* je delovanje na mobilnih napravah zaradi računske zahtevnosti še vedno precej omejeno. Omejen je tudi nabor nastavitev za konfiguracijo upodabljanja. Nobena od predstavljenih rešitev pa ne podpira oddaljenega sodelovanja več uporabnikov.

[4] *VTK - The Visualization toolkit*. Dosegljivo: `https://vtk.org/`. [Dostopano: 27. 2. 2019].

[5] *Visualize Your Data With vtk.js*. Dosegljivo: `https://kitware.github.io/vtk-js/index.html`. [Dostopano: 27. 2. 2019].

[6] *SimVascular*. Dosegljivo: `http://simvascular.github.io/`. [Dostopano: 27. 2. 2019].

Za namen tega dela sta najbolj ključni ogrodje za vizualizacijo Med3D [25] in orodje za Volumetrično sledenje potem [26] - VPT. Med3D je osnova, v katero se v sklopu tega diplomskega dela integrira neposredne upodabljalne metode sledenja poti, ki jih implementira orodje VPT. Projekta sta podrobneje predstavljena v naslednjih podpoglavjih.

2.4 Med3D

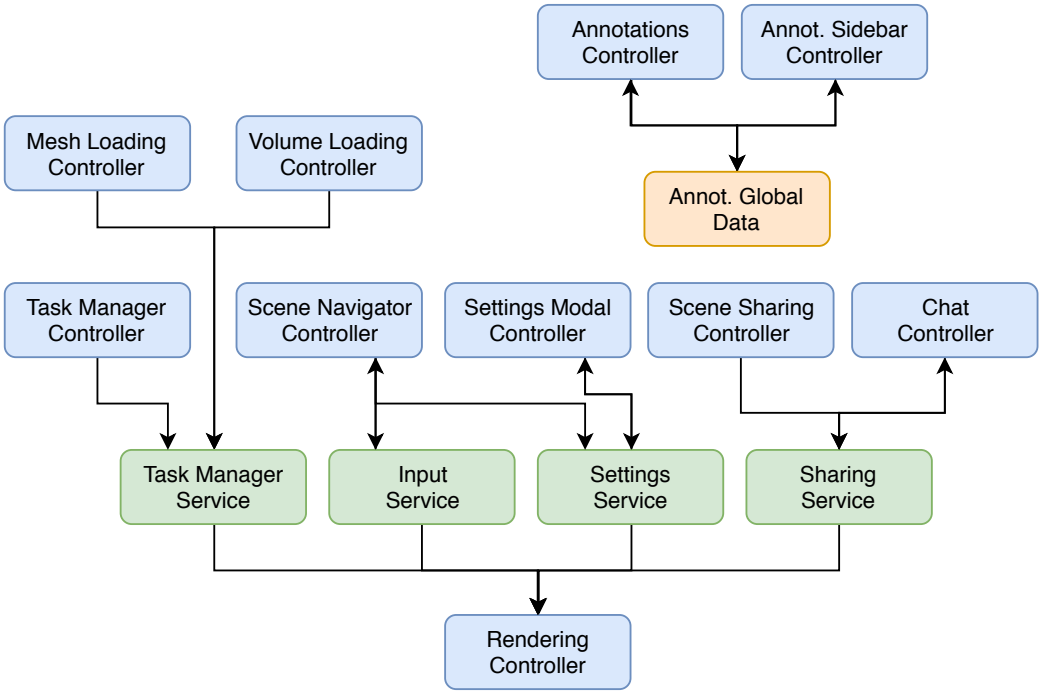
Leta 2016 je na Fakulteti za Računalništvo in Informatiko v sklopu diplomske naloge Primož Lavrič [25] razvil aplikacijo Med3D. Namen aplikacije je interaktivna vizualizacija medicinskih podatkov.

Osrednja funkcija je upodabljanje 3D podatkov. Upodabljanje temelji na spletni tehnologiji WebGL. To omogoča platformno neodvisno uporabo in deluje na skoraj vsaki sodobni pametni napravi.

Druga pomembna funkcija je oddaljeno sodelovanje. Ideja je povezati uporabnike, bolj natančno zdravnike, in jim omogočiti sinhronizirano vizualizacijo istega seta medicinskih podatkov. Vizualizacija omogoča bistveno hitrejši prenos informacij od enega zdravnika do drugega v primerjavi s tekstovno ali pa govorno komunikacijo na daljavo. To je torej lažji, hitrejši in bolj natančen način predstavitve medicinskega problema, posledično pa je izboljšan postopek podaje drugega mnenja. Komunikacija znotraj usklajene vizualizirane scene poteka preko tekstovnega pogovora in anotacij. Omogočene so tekstovne in grafične anotacije. Tekstovne je možno povezati s kazalcem neposredno v prostor 3D scene, grafične pa so v obliki 2D skic poravnane pogledom kamere.

2.4.1 Arhitektura sistema in grafični cevovod

Arhitektura Med3D-ja temelji na gradnikih AngularJS, ki omogočajo razvoj enostranskih aplikacij. Arhitektura je predstavljena na sliki 2.2

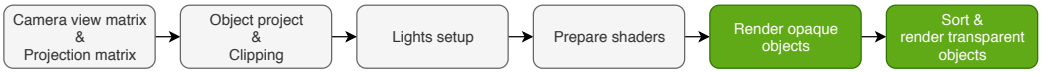


Slika 2.2: Arhitektura gradnikov sistema Med3D, povzeto po [25].

Za upodabljanje je zadolžen *upravljalnik upodabljanja* (angl. rendering controller). Ta vsebuje grafični cevovod. Vsakič ko se cevovod izvede, se na zaslon izriše nova slika. Brskalnik omejuje maksimalno število izvedb cevovoda, običajno do 60 na sekundo, seveda pa le v primeru, da je strojna oprema zadosti zmogljiva.

Cevovod je sestavljen iz več stopenj. Delitev na več stopenj omogoča *naknadno obdelavo* (angl. post-processing) izrisane slike. V temu primeru je to izris grafičnih anotacij na upodobitev 3D scene. Trenutno sta ključni stopnji: (1) izris objektov scene v prvi stopnji in (2) dodatek grafičnih anotacij v eni izmed naslednjih. Ta način omogoča tudi skalabilnost cevovoda za dodatne

funkcije v prihodnosti, pa tudi neodvisnost posameznih stopenj. V temu delu nas zanima predvsem (1) *glavna upodobitvena stopnja* (angl. main render pass). Stopnja je prikazana na sliki 2.3.



Slika 2.3: Stopnja cevovoda, upodobitev objektov scene.

Grafični cevovod Med3D zna upodabljati samo objekte z mrežno geometrijo. V primeru volumetričnih vhodnih podatkov je potrebno izračunati njihovo mrežno predstavitev. V ta namen je uporabljen algoritem Marching cubes.

2.4.2 Uporabniški vmesnik

Osrednji del uporabniškega vmesnika je velika površina platna. Vse uporabniške funkcionalnosti so pospravljene v menije na minimalni navigacijski in stranski vrstici. Navigacijska vrstica je namenjena splošnim nastavitvam, nalaganju podatkov, pripravi nove scene in nastavitvam oddaljenega sodelovanja, stranska vrstica pa hitrim nastavitvam za interakcijo s sceno. To so podmeniji za tekstovne in grafične anotacije ter podmeni za izbiro kamere. Uporabniški vmesnik je zgrajen večinoma iz Bootstrap gradnikov. Logične celote pa so implementirane z AngularJS *upravljalniki* (angl. controllers), kar je vidno na sliki na sliki 2.2.

2.4.3 Oddaljeno sodelovanje

Med3D ima implementiran sistem oddaljenega sodelovanja. Sistem omogoča uporabniku povezavo v sejo, nakar se lokalni podatki sinhronizirajo s stanjem seje. Takšni deljeni podatki so vsi objekti v sceni, kamere posameznih uporabnikov in deljene anotacije. Objekte v sceni lahko spreminja samo uporabnik, ki je odprl sceno. Kamero ima vsak uporabnik svojo. Uporabi

lahko kamere ostalih uporabnikov, ne more jih pa nadzorovati. Isti princip velja tudi za anotacije. Komunikacija poteka preko pogovornega tekstovnega okenca. Omenjeni podatki se delijo in sinhronizirajo med uporabniki seje v realnemu času. Za to poskrbi zaledni strežniški del, s katerim komunicira uporabniški del aplikacije.

Strežniški del uporablja tehnologijo Node.js, za komunikacijo z uporabniškim delom pa se uporablja WebSockets API[54].

2.5 VPT

Leta 2017 je Žiga Lesar v svojemu magistrskem delu [26] predstavil in implementiral metode *sledenja potem* (angl. path tracing) za upodabljanje volumetričnih podatkov v realnemu času s spletnimi tehnologijami. V predstavitveni aplikaciji je implementiral štiri metode neposrednega upodabljanja. Kot smo omenili, v pregledu neposrednih metod, metode simulirajo žarek skozi vsako točko ali piksel slikovnega okvirja, žarek potuje skozi volumen, izvornemu pikslu pa se določi barva, po izbrani metodi, specifični za posamezno implementacijo.

- *Projekcija največje intenzitete* (angl. maximum intensity projection, MIP) je metoda iskanja in projekcije najvišje intenzitete volumna. Žarki se na naključnih globinah volumna vzorčijo. Najvišja intenziteta se zabeleži za posamezen izvorni piksel. Temu se določi barvo glede na zabeleženo intenziteto.

- *Projekcija nivojskih ploskev* (angl. isosurfaces). Metoda upodablja ploskve v volumetričnih podatkih z intenziteto, višjo od vhodne mejne vrednosti. Ker se ploskve izrisuje brez prosojnosti, zadostuje, da se upodobi samo prva vidna ploskev. Tako kot prej se žarki na poti skozi volumen vzorčijo na naključnih točkah. Išče se najmanjša globina, pri kateri je intenziteta še višja od vhodne intenzitetne meje. Za takšne najdene točke v volumnu se izračuna lokalni gradient glede na sosednje

vrednosti. Ta gradient predstavlja normalo na ploskev v tisti točki. Točke najbližjih ploskev se, glede na normalo, upodobijo z modelom lokalnega osvetljevanja.

- *Metoda emisijsko-absorpcijskega modela* (angl. emission absorption model - EAM). Metoda uporabi sevalno enačbo za izračun emisijskega spektra. Žarki se na enakomernih razdaljah znotraj volumna vzorčijo. Vsakemu vzorcu žarka se za prenosno funkcijo in intenziteto po sevalni enačbi z absorpcijo izračuna emisijo. Akumulirana emisija vseh vzorcev določa barvo posameznega piksla.

- *Metoda Monte Carlo za sevalno enačbo z enkratnim sipanjem.* Metoda temelji na iterativnemu izboljševanju slike s stohastično metodo Monte Carlo.

  V vsaki iteraciji se za vsak žarek znotraj medija določi točka sipanja po Woodcockovi metodi [60]. Za nov žarek se oceni prepustnost medija do izvora svetlobe. Izračunana vrednost sevalnosti se vnese v Monte Carlo tekoče povprečje, ki predstavlja barvno vrednost za ta izvorni piksel.

Implementacija upodabljanja s posameznimi metodami uporabljajo enotno obliko cevovoda, prikazano na sliki 2.4.

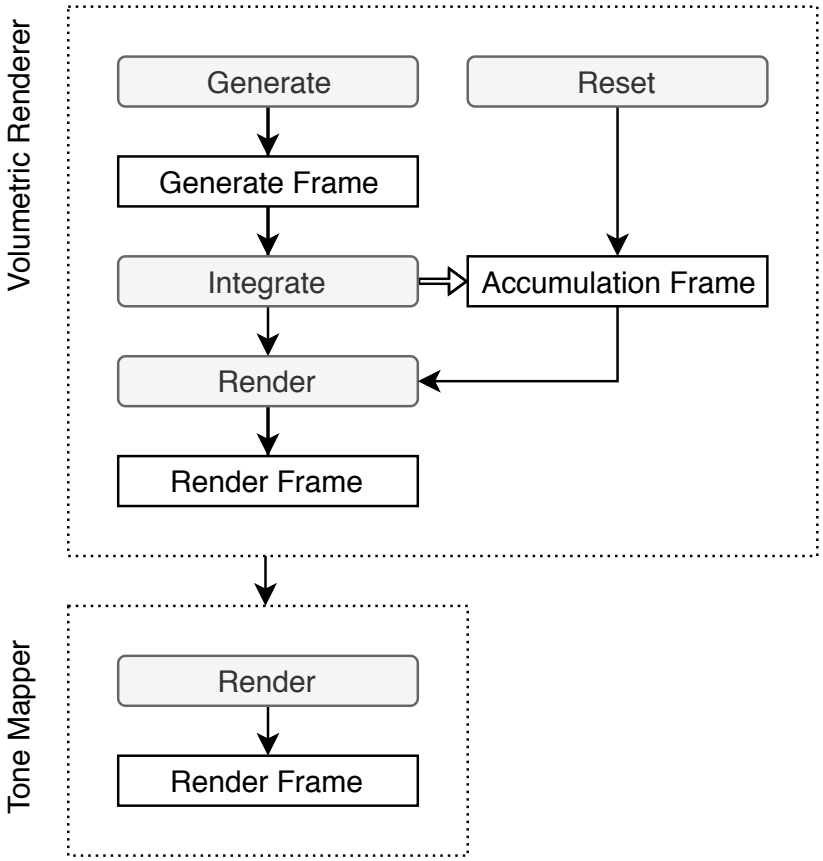
Cevovod je sestavljen iz več stopenj:

- *Generacija* (angl. Generate). Upodobitev aproksimacije končne slike.

- *Integracija* (angl. Integrate). Vključitev generacijskega okvirja v akumulacijskega, ki služi kot medpomnilnik za iterativno izboljšavo.

- *Upodobitev* (angl. Render). Izris slike iz akumulacijskega okvirja. Izrisana slika je v *visokemu dinamičnemu razponu*(angl. high dynamic range - HDR), pred izrisom na zaslon pa jo je potrebno stisniti.

- *Ponastavitev* (angl. Reset). Ponastavi akumulacijski okvir.



Slika 2.4: Splošen grafični cevovod upodabljevalnikov. Povzeto po [26].

Po upodobitvi se barve slike v *tonskemu slikarju* (angl. tone mapper) stisnejo na interval [0, 1], ki ga je možno prikazati na zaslonu. Implementirana sta dva tonska slikarja:

- Tonski slikar intenzitetnega obsega, ki linearno preslika vrednosti iz intervala $[\mathrm{L_{min}}, \mathrm{L_{max}}]$ na interval [0, 1]. Vsaka vhodna vrednost $\mathrm{L_i}$ se preslika v izhodno vrednost $\mathrm{L_o}$ po funkciji,

$$L_o = \frac{L_i - L_{min}}{L_{max} - L_{min}} \tag{2.1}$$

  Problematična je določitev $\mathrm{L_{min}}$ in $\mathrm{L_{max}}$, saj bilo potrebno narediti obhod vseh vrednosti za določitev dejanskega razpona. Vrednosti zunaj

razpona $[\mathrm{L}_{\min}, \mathrm{L}_{\max}]$ se pri prikazu na zaslon enolično preslikajo v 0 ali 1.

- Reinhardov tonski slikar reši omenjen problem z uporabo injektivne, monotono naraščajoče funkcije preslikave

$$L_o = \frac{L_i}{\frac{1}{\varepsilon} + L_i} \tag{2.2}$$

ki preslika vrednosti $[0, \infty)$ v $[0, 1]$, ε pa je ekspozicija kamere.

Izvajanje tonskih slikarjev je hitro, saj se uporabi samo osnovne matematične operacije.

2.6 Uporabljene tehnologije

V tem poglavju predstavimo tehnologije, ki se uporabljajo za delovanje aplikacije Med3D. Uporaba teh tehnologij se nadaljuje v nadgradnji omenjene aplikacije, ki je predstavljena v poglavju implementacije.

2.6.1 Node.js

Običajno strežniki delujejo tako, da se za vsako novo povezavo ustvari novo nit delovanja, ki opravi odziv na prošnjo. V praksi se lahko pojavijo težave. V primeru, kjer je veliko niti, se porablja del zmogljivosti samo za razporejanje opravil in menjanje kontekstov. Node.js je strežnik, ki uporablja drugačen pristop. Uporablja eno nit, ki jo prožijo zunanji dogodki in ne blokira vhodno izhodnih naprav. S skaliranjem količine zahtevkov takšen pristop bolje ohranja nizko latenco odgovorov. To olajša aplikacijske modele, kjer je potrebna komunikacija s strežnikom v realnemu času. Tak primer je sistem oddaljenega sodelovanja v aplikaciji Med3D, kjer lahko veliko uporabnikov ob istemu času proži veliko posodobitev na deljenih podatkih.

Velik del sposobnosti ogrodja Node izhaja iz ogromnega nabora knjižnic in orodij v paketni knjižnici NPM[7].

[7] *npm*. Dosegljivo: `https://www.npmjs.com/`. [Dostopano: 28. 2. 2019].

Node.js poganja V8[8] JavaScript izvajalno okolje. Razvoj strežniškega dela aplikacije torej poteka v programskem jeziku JavaScript, kar ga poenoti z razvojem uporabniškega dela. To olajša tudi kombiniranje s tehnologijami, ki podpirajo podatke v obliki JSON[9], na primer MongoDB[10] in Postgres[11]. MongoDB podatkovna baza se v Med3D aplikaciji uporablja za hiter dostop do nekaterih pogostih objektov.

2.6.2 AngularJS in Bootstrap

Standardi za sodobne spletne aplikacije so dovršen dizajn, hitra odzivnost, dobra funkcionalnost in podpora naprav z zasloni različnih velikosti. To je možno le z veliko kode. Računalnik je zmožen hitre interpretacije velike količine kode, za razvoj pa je to zelo neugodno. V ta namen so se uveljavila spletna ogrodja (angl. frameworks), ki že implementirajo nižjenivojsko delovanje spletne strani, tako da se razvijalec lahko osredotoči na implementacijo višjenivojskih funkcionalnosti.

Ogrodje Bootstrap je namenjeno implementaciji srednjenivojskih gradnikov. To so razni meniji, navigacijske vrstice, namenska vnosna polja in podobno. Za poenoten videz spletne strani morajo imeti gradniki enak stil. Z Bootstrap urejevalniki je možno grafično določiti videz celemu paketu gradnikov in ga izvoziti v obliki slogovnega zapisa CSS (angl. cascading style sheets). Na ta način se implementira tudi logika, ki prikaže Bootstrap gradnike odvisno od vrste naprave, ki jo ima uporabnik.

AngularJS je višjenivojsko ogrodje, katerega glavna ideja je omogočiti visoko dinamičnost dokumentov HTML. Na abstrakten način združuje gradnike uporabniškega vmesnika v hierarhično gnezdene module. Moduli v obliki *direktiv* (angl. directives), *upravljalnikov* (angl. controllers), *tovarn*

[8] *V8 JavaScript engine*. Dosegljivo: `https://v8.dev/`. [Dostopano: 28. 2. 2019].

[9] *JSON*. Dosegljivo: `https://www.json.org/`. [Dostopano: 28. 2. 2019].

[10] *Open Source Document Database — MongoDB*. Dosegljivo: `https://www.mongodb.com/`. [Dostopano: 28. 2. 2019].

[11] *PostgreSQL: The World's Most Advanced Open Source Database*. Dosegljivo: `https://www.postgresql.org/`. [Dostopano: 28. 2. 2019].

(angl. factories) in *servisov* (angl. services) predstavljajo logiko aplikacije. Taka abstrakcija vpeljuje dobre prakse razvoja. Posledično pa je izdelava večjih, kompleksnejših storitev, bistveno lažja. AngularJS omogoča izdelavo monolitnih *aplikacij na eni strani* (angl. single page application). To je posebej uporabno v primerih, kjer ni mogoče deliti ključnih elementov, na primer risalne površine v primeru aplikacije Med3D.

2.6.3 WebGL

Kot smo omenili, grafično procesne enote omogočajo pohitritev nekaterih računskih algoritmov, katere je možno paralelizirati. To so tudi algoritmi upodabljanja. V ta namen se je v vseh brskalnikih, postopoma od leta 2011, standardizirala tehnologija WebGL. Gre za vmesnik med programi na spletni strani, napisanimi v JavaScriptu, in grafičnimi vmesniki operacijskega sistema, ki preko strojnih gonilnikov razpolagajo z grafičnimi procesnimi enotami. Funkcije, ki jih omogoča, so osnovane za delovanje na nižje nivojskemu grafičnemu vmesniku OpenGL[12]. V primeru Windows operacijskih sistemov se uporabi pretvorna plast ANGLE[13], ki namesto na vmesniku OpenGL deluje na bolj domorodnem vmesniku DirectX[14].

Izvajanje algoritmov na GPE je določeno s *senčilniki* (angl. shaders), programi, ki se prevedejo in naložijo na GPE. Senčilniki za WebGL (in OpenGL) uporabljajo standardiziran jezik *GLSL* (angl. Graphics Library Shading Language)[15]. Senčilnik se izvede na določeni stopnji grafičnega cevovoda WebGL, prikazanega na sliki 3.3. Izbira mesta izvajana, znotraj cevovoda, je omejena na koraka senčenja oglišč *vertex shading* in senčenja

[12] *OpenGL - The Industry's Foundation for High Performance Graphics*. Dosegljivo: `https://www.opengl.org/`. [Dostopano: 27. 2. 2019].

[13] *ANGLE*. Dosegljivo: `https://opensource.google.com/projects/angle`. [Dostopano: 1. 3. 2019].

[14] Wikipedia. *DirectX*. Dosegljivo: `https://en.wikipedia.org/wiki/DirectX`. [Dostopano: 1. 3. 2019]. 2019.

[15] *OpenGL Shading Language Overview*. Dosegljivo: `https://www.opengl.org/sdk/docs/tutorials/ClockworkCoders/glsl_overview.php`. [Dostopano: 28. 2. 2019].

fragmentov *fragment shading*.

V Med3D-jevemu grafičnem cevovodu se senčilniki uporabljajo za izris objektov poligonskih mrež v izhodni slikovni okvir, ki je bodisi platno na zaslonu, bodisi shranjena tekstura v pomnilniku GPE. Na sliki 2.3 so mesta, kjer se uporabi GPE, označena z zeleno barvo.

Poglavje 3

Implementacija

V tem poglavju je predstavljena implementacija nadgradnje aplikacije Med3D. Opravljeno delo razdelimo na več logičnih enot: nadgradnja grafičnega cevovoda, uporabniškega vmesnika in sistema oddaljenega sodelovanja. Za vsak del podamo razmislek o izvedbi, potek implementacije in končno stanje.

3.1 Grafični cevovod

V pregledu področja smo predstavili, da se ogrodji VPT in Med3D bistveno razlikujeta v delovanju. Kritične razlike so med sistemoma so:

- Metode VPT uporabljajo pristop neposredne upodobitve. To pomeni, da se neka vrednost iz vhodnih podatkov neposredno preslika v vrednost na zaslonu.

  Med3D-jev grafični cevovod je pripravljen za upodabljanje objektov s poligonskimi mrežami. Za prikaz volumetričnih podatkov uporablja pristop posredne upodobitve. Z algoritmom marching cubes pripravi poligonsko mrežo. Poligonska mreža je samo oblika oziroma geometrija, kateri se v postopku upodabljanja določi videz z izračunom osvetlitve. Pri Med3D je oblika statična in vnaprej znana še pred upodobitvijo. Pri neposrednemu upodabljanju pa je oblika povsem odvisna od parametrov upodabljalnika, dobiti pa jo je možno šele v koraku upodobitve, pa še

to ne vedno. V primerih, kjer metoda neposrednega upodabljanja pri upodobitvi ne projicira informacij o globini, se obliko lahko določi samo kot 2D obris. Pri VPT orodju to velja za metodo projekcije največje intenzitete in metodo emisijsko-absorpcijskega modela.

Omejena informacija o obliki je za nekatere funkcije vizualne interakcije zelo omejujoča. Tak primer je funkcionalnost dodajanja 3D sidranih anotacij neposredno na objekt v sceni. V Med3D se postopek izvede v JavaScriptu, na strani procesorja. Izračuna se presečišče med projicirano lokacijo klika in geometrijo. Na mesto preseka se doda 3D označbo. Tekstovni anotaciji se doda črto, ki kaže na 3D zasidrano označbo.

Pri neposrednemu upodabljanju je taka funkcionalnost težko izvedljiva. Potrebno bi bilo opraviti izračun znotraj senčilnika upodabljanja ali pa izvoziti globinsko sliko, ki predstavlja omejeno informacij o globini objekta. Oba načina sta slaba, saj sta bolj kompleksna in ne upoštevata vseh robnih primerov, v drugačnih splošnih primerih geometrijskih interakcij pa bi bil tak pristop lahko tudi nemogoč. Na primer merjenje dimenzij objekta ali pa zaznavanje kolizij.

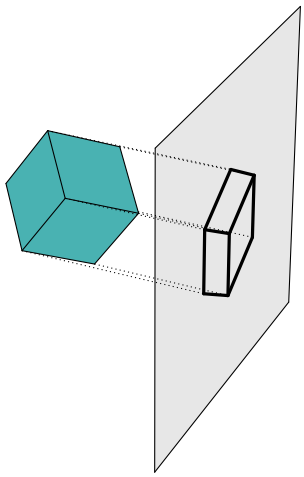
- Med3D podpira izris scene, ki ima poljubno število objektov. VPT lahko upodablja samo en set volumetričnih podatkov naenkrat. Izkaže se, da bi bil izris več setov volumetričnih podatkov v isto sceno napačna odločitev iz več razlogov:

  – Več setov volumetričnih podatkov ni smiselno izrisovati z istimi nastavitvami, torej z enako metodo izrisa in enakimi parametri izrisa. Razlog je, da je potrebno nastavitve izrisa ročno optimizirati tako, da se iz volumetričnih podatkov izlušči največ smiselnih informacij. To pomeni, da bi en deljen set nastavitev verjetno povzročil slabo upodobitev drugega seta volumetričnih podatkov. Če se uporabi več setov nastavitev, pa nastane nov problem, saj neposredni način izrisa določa tudi druge aspekte scene, kot so barva ozadja in barvni razpon. Teh aspektov se ne sme zanemariti,

smiselnega načina združitve pa ni.

– Pri več setih volumetričnih podatkov v sceni je težko določiti pravila o medsebojni interakciji objektov. Če se dva seta podatkov v sceni prekrivata ali pa sekata, je zelo težko pravilno določiti, kako naj se izriše skupno prekrivno območje. Surovih volumetričnih podatkov medsebojno ni smiselno kombinirati, saj so lahko zajeti z različnimi tehnologijami in imajo različne razpone vrednosti. To pomeni, da bi bilo različne sete volumetričnih podatkov bolje upodobiti ločeno, nato pa upodobitvi združiti. Združitev upodobitev je ponovno problematična zaradi negotove narave globinske informacije.

Naša rešitev je združitev neposrednega upodabljanja s posrednim. Z dobro implementacijo lahko ohranimo prednosti obeh načinov in izločimo slabosti. Potrebno pa je minimizirati neželene posledice, kot so dodana računska zahtevnost in izguba kakovosti. Ideja je uporabiti mrežno geometrijo objekta, na katero pri izrisu projiciramo neposredno upodobitev volumetričnih podatkov.



Slika 3.1: Pristop s paralelno projekcijo teksture na geometrijo. (Vir: Povzeto po [16], pod licenco CC BY-SA 4.0 [5].)

To pomeni, da se volumetrični podatki najprej neposredno izrišejo v ločen slikovni okvir, nato pa se ta, kot tekstura, uporabi pri upodobitvi mrežnega objekta. Teksturo je potrebno pravilno mapirati na geometrijo, tako da uporabnik sploh ne opazi, da gre za geometričen objekt namesto neposredne upodobitve. To dosežemo s paralelno projekcijo teksture, pristop pa je prikazan na sliki 3.1.

Tako se lahko uporabi obstoječi grafični cevovod z nadgradnjo za poseben tip objektov. Nov tip objekta bomo zaradi jasnosti imenovali volumetrični objekt.

3.1.1 Implementacija volumetričnega objekta

Volumetrični objekt je implementiran kot razširitev mrežnega objekta. Običajno ima mrežni objekt atributa geometrije in materiala, s katerim se geometrija izriše. Volumetrični objekt ima dodana še dva atributa - surove volumetrične podatke in teksturo z neposredno upodobitvijo.

Volumetrični podatki se uporabijo pri aproksimaciji geometrije in pri neposredni upodobitvi v teksturo.

Tekstura se uporabi kot vhodni podatek za material, ki ga bomo definirali z novim senčilnikom.

Geometrija volumetričnega objekta je pravzaprav poljubna, saj bo vidna le tam, kjer je pripeta projicirana tekstura. Pomembno pa je, da ni manjša od upodobitve na teksturi. V pregledu področja je predstavljen algoritem Marching Cubes, ki ga uporabljamo za aproksimacijo primerne geometrije. Ta algoritem ima višjo računsko zahtevnost, porabi veliko pomnilnika in se izvaja nekaj časa. Tukaj dopuščamo, da se uporabnik sam odloči, kdaj želi bolj reprezentativno geometrijo volumetričnih podatkov. Za začetno geometrijo je potrebna bolj splošna rešitev, ki ni odvisna od volumetričnih podatkov. Vhodni volumetrični podatki so podani v obliki regularne mreže. To so podatki, zajeti z osno-enakomernim vzorčenjem vzdolž treh medsebojno ortogonalnih osi in imajo obliko tridimenzionalnega polja številčnih vrednosti. Takšno tridimenzionalno polje ima torej obliko kvadra, vsak kvader pa je

možno vstaviti v kocko primerne velikosti. Splošna oblika naj bo torej enotska kocka. Kocka ima le 8 oglišč, 6 stranic oziroma 12 trikotnikov v zapisu s poligonsko mrežo. To je odlično, saj je zahtevnost izrisa geometrije sorazmerna s številom trikotnih poligonov, 12 pa je zanemarljivo malo.

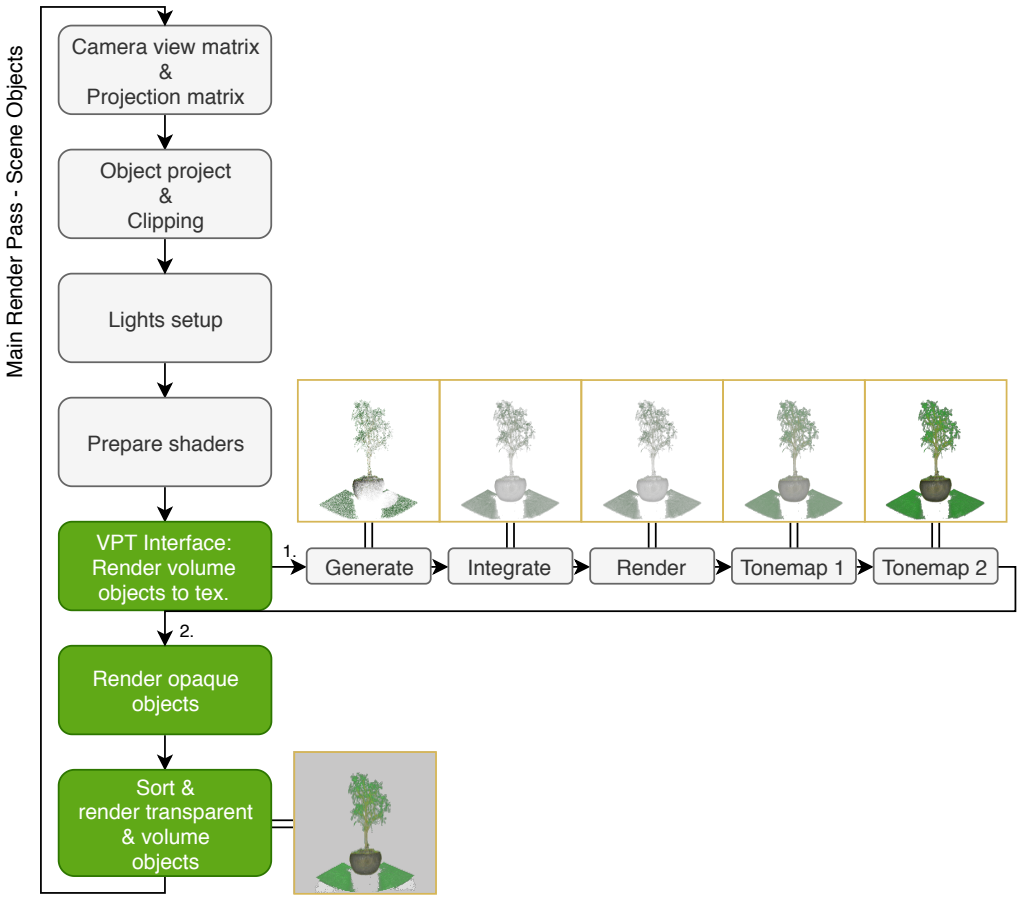
Material predstavlja videz upodobljene geometrije. Splošen primer bi bila neka statična barva, ki se jo v koraku senčenja, za različne točke na geometriji, različno osvetli glede na postavitev luči v sceni. Postopek določitve videza se torej imenuje senčenje in poteka v posebnemu programu, ki se izvede na GPE in se imenuje *senčilnik* (angl. shader). Za volumetričen objekt je material definiran z senčilnikom, ki je predstavljen v nadaljevanju.

3.1.2 Integracija metod VPT

Upodobitev volumetričnega objekt poteka torej v dveh korakih. Neposredna upodobitev podatkov v teksturo volumetričnega objekta z uporabo metod VPT, nato pa še upodobitev geometrije s pripravljeno teksturo. Grafični cevovod aplikacije Med3D smo v ta namen razširili z metodami VPT, ki omogočajo neposredno upodabljanje.

Pri integraciji smo uporabili sistem za verzioniranje Git[1]. Ogrodje VPT je uvoženo v Med3D kot Git modul. Tak pristop omogoča ločen, medsebojno neodvisen razvoj obeh ogrodij. V primeru posodobitev VPT projekta ne bo potrebna ponovna integracija, temveč le manjši popravki v primeru konfliktov. Ker sta projekta ločena, se izdela *vmesnik* (angl. interface), ki predstavlja povezavo z VPT-jevim notranjim cevovodom. Vmesnik je vstavljen v Med3D-jev grafični cevovod, kar prikazuje slika 3.2.

[1] *Git.* Dosegljivo: `https://git-scm.com/`. [Dostopano: 1. 3. 2019].



Slika 3.2: Shema nadgrajenega cevovoda za stopnjo izrisa objektov. Prikazan je primer z uporabo metode MCS, obeh tonskih slikarjev in aproksimirane geometrije.

Potek cevovoda je tako daljši za en dodaten korak izrisa. V primerjavi z neposredno upodobitvijo je potrebno dodatno upodobiti geometrijo in projicirati teksturo. To pa predstavlja neko dodano računsko zahtevo. Reda povečane računske zahtevnosti ni mogoče enostavno izračunati, saj je povsem odvisen od implementacije funkcij v WebGL-u, kot tudi od same implementacije v gonilnikih in strojni opremi. V poglavju vrednotenja je predstavljena performančna analiza, ki pokaže povečanje zahtevnosti za primer volumetričnih podatkov pri različno kompleksni poligonski geometriji in različno zmogljivi

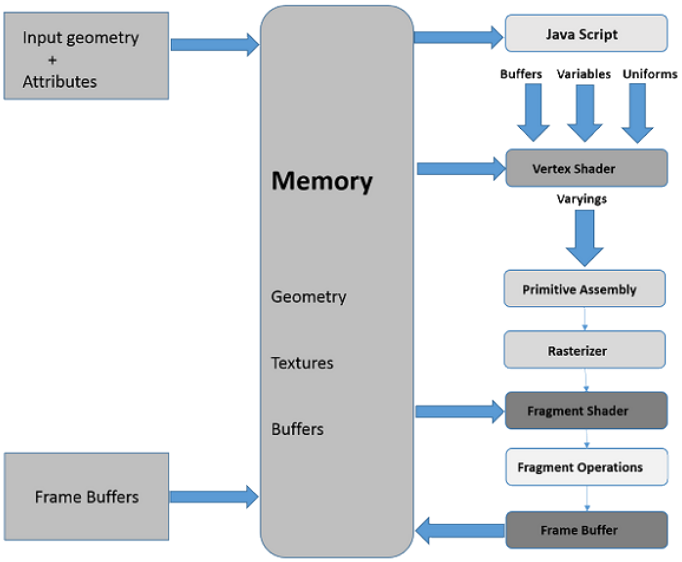
strojni opremi.

3.1.3 Senčilnik volumetričnega objekta

Določiti je potrebno še postopek izrisa geometrije. Implementira se senčilnik, ki kot atribut prejme teksturo upodobitve volumetričnih podatkov iz cevovoda VPT.

Nov senčilnik izhaja iz senčilnika s Phongovim senčenjem, dodana pa je logika paralelne projekcije. Uporabniku želimo omogočiti kombiniranje senčenja geometrije s Phongovim pristopom osvetljevanja in senčenja s projekcijo statične teksture. Tako bo volumetrični objekt prikazan v razponu med posredno upodobitvijo s senčeno geometrijo in neposredno upodobitvijo s fizikalno natančnimi metodami.

Na sliki 3.3 je prikazana shema grafičnega cevovoda aplikacijskega vmesnika WebGL. Senčilnik je sestavljen iz dveh delov: najprej se izvede *senčilnik oglišč* (angl. vertex shader), nato pa se izvede še *senčilnik fragmentov* (angl. fragment shader). To sta edina koraka v cevovodu WebGL-a, ki ju programer lahko določi.



Slika 3.3: WebGL grafični cevovod. (Vir: [53])

Phongovo senčenje je že implementirano, potrebna je še projekcija teksture. Izračunati je potrebno katera točka teksture se preslika na posamezni del geometrije. Oglišča geometrije objekta se iz lokalnih koordinat preslikajo v koordinate pogleda. Velja račun preslikave.

$$vP' = \text{ProjM} \times (\text{ModelViewM} \times vP), \tag{3.1}$$

kjer je vP koordinata oglišča, ModelViewM matrika transformacije objekta, ProjM projekcijska matrika kamere, vP' pa izračunana koordinata oglišča v svetu pogleda.

To se izračuna v senčilniku oglišč in se kot izhodni oz. *variabilni* (angl. varying) atribut pošlje naprej po cevovodu.

V koraku rasterizacije se za vsak poligonski trikotnik, z interpolacijo koordinat oglišč, določijo točke izrisne ravnine. Na isti način se interpolirajo tudi variabilni atributi.

Senčilnik fragmentov vsaki rasterizirani točki določi barvo. Barva naj bo odvisna od projicirane teksture. Ker se pri izrisu teksture uporablja parametre iste kamere in enako transformacijo objekta, je projekcija bistveno poenostavljena. Velja, da se *svet izrisa* (angl. screen space), v kateremu je tekstura, s *svetom pogleda* (angl. clip space), v kateremu se upodabljajo oglišča, razlikuje le v razponu koordinat in številu osi. Določitev mapirnih točk je zato preprosta. Najprej je potrebno normalizirati interpolirane koordinate fragmentov, tako da so v razponu [-1, 1].

$$ivP'' = ivP'/ivP'.w \tag{3.2}$$

$ivP'.w$ označuje homogeno koordinato, oz. četrti element v vektorju. Svet pogleda ima x, y in z osi v razponu [-1, 1], svet izrisa pa osi x in y v razponu [0, 1]. Velja,

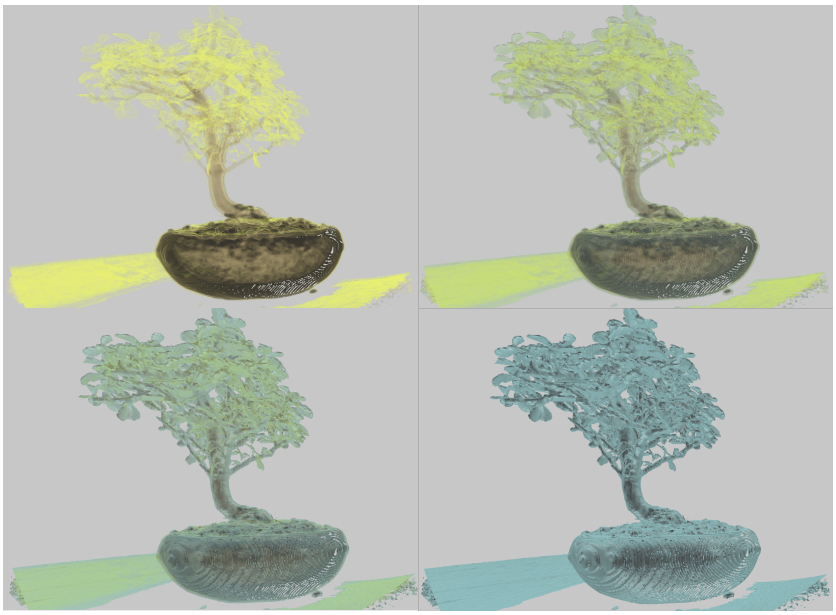
$$tP = (ivP'' \times 0.5 + 0.5).xy, \tag{3.3}$$

kjer je tP koordinata na teksturi.

Napovedan kombiniran izris med geometrijo, senčeno s Phongovim modelom in upodobitvijo volumetričnih podatkov z VPT, je implementiran z *alfa zlivanjem* (angl. alfa blending) obeh izrisov. Uporabnik ima v uporabniškem vmesniku možnost določiti vrednost α v razponu [0, 1]. Barva oglišča se določi kot:

$$vC = \alpha \times \text{vCPhong} + (1 - \alpha) \times tC[tP] \tag{3.4}$$

Kjer je $vC[tP]$ barvna vrednost točke tP na teksturi. vCPhong pa je barva točke ivP'', izračunana s Phongovim modelom osvetljevanja. Pri vrednosti $\alpha == 0$ se sme povsem preskočiti izračun osvetlitve, saj osvetlitev geometrije ne bo vidna, tako pa se prihrani na računski zahtevnosti. Enako velja za $\alpha == 1$, kjer se sme preskočiti celotno neposredno upodobitev v teksturo in njeno projekcijo. Rezultat je viden na sliki 3.4.



Slika 3.4: Prikaz zlivanja Phongovega senčenja aproksimirane geometrije in neposredne upodobitve z metodo emisijsko-absorpcijskega model. Vrednosti α pri 0% (levo zgoraj), 33% (desno zgoraj), 67% (levo spodaj), 100% (desno spodaj).

3.2 Uporabniški vmesnik

Eden izmed zastavljenih ciljev je visoka stopnja interaktivnosti. Interaktivnost je mera informacije v komunikaciji med uporabnikom in programom. Definirati je možno nekaj dobrih praks interaktivnega uporabniškega vmesnika:

- Uporabniku mora biti omogočeno v kar se da malo korakih najti iskane informacije.
- Pri izbiri korakov mora bit uporabnik ustrezno usmerjen. To se lahko opredeli kot smiselno združevanje informacij sorodnega tipa.
- V primerih, ko ni mogoče takoj odgovoriti na uporabnikovo zahtevo, je pomembno obvestiti uporabnika, da ne čaka zaman.
- Nabor funkcionalnosti, s katerimi uporabnik razpolaga, naj bo kar se da širok, a pri tem se ne sme kršiti drugih načel. V praksi se najde veliko primerov, kjer je funkcionalnosti preveč ali pa premalo.

V primeru Med3D-ja se interaktivnost nanaša na kakovost uporabniškega vmesnika. Informacijo se lahko opredeli kot upodobitev objekta. Število korakov naj bo število uporabniških vnosov preko miške, tipkovnice ali dotikov zaslona. Združevanje informacij pomeni združevanje tematsko podobnih opcij v podmenije, ki so ustrezno logično naslovljeni. Odzivnost pa je takojšna povratna informacija na uporabniški vnos. To je lahko kar koli, na primer vidna sprememba na platnu ali pa sporočilno okno, ki obvešča o poteku akcije.

3.2.1 Implementacija novosti v uporabniški vmesnik

V osnovi je Med3D, tako kot ostale spletne aplikacije, definiran s standardiziranim označevalnim jezikom HTML. HTML je jezik, ki ga interpretira brskalnik. Vsebuje informacije o elementih, ki naj jih brskalnik prikaže. Gre za elemente nižjega abstraktnega reda, kot so gumbi stikala, vnosna polja, okvirji, napisi in podobno. Z logičnim in funkcionalnim grupiranjem takšnih elementov je možno narediti elemente višjih abstraktnih razredov (npr. izbirnik barve).

Videz samih HTML elementov je v osnovi takšen, kakršnega definira brskalnik. V namen bolj specifičnega videza elementov je na voljo slogovni jezik CSS. Za logične povezave med elementi pa se uporabi jezik JavaScript. S temi tremi jeziki je možno zgraditi poljubno spletno stran.

Ročno pisanje celotne HTML, CSS in JavaScript kode bi bilo izjemno zamudno in podvrženo napakam. Pri izdelavi večjih spletnih strani se uporabi tehnologije, ki že implementirajo nižjenivojsko delovanje in vpeljujejo dobre prakse razvoja. V pregledu uporabljenih tehnologij so naštete uporabljene tehnologije.

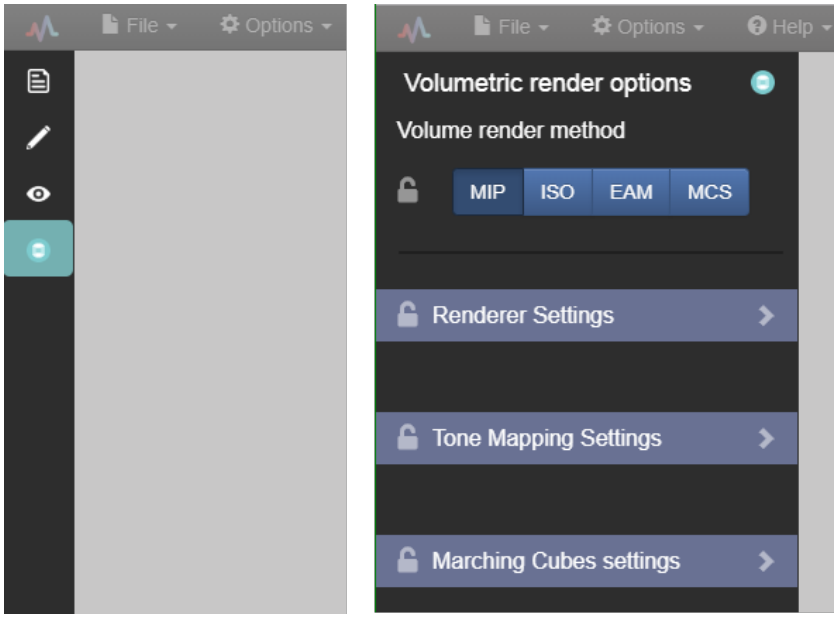
Uporabi se Bootstrap za večino interaktivnih elementov. Videz elementov je usklajen z uporabo lastnega stila Bootstrap gradnikov.

Tehnologija AngularJS združuje gradnike v logične module in omogoča implementacijo celotne aplikacije na eni strani.

Osrednji gradnik je HTML element platna `<canvas>`, na katerega se izrisuje scena. Pomembno je, da je platno veliko in razločno, drugi elementi pa ne smejo bistveno posegati v interakcijo z njim. V ta namen so vse možnosti pospravljene v hierarhične menije v dveh robnih vrsticah. Vrstice so prikazane na sliki 3.5.

Zgornja navigacijska vrstica vsebuje splošne nastavitve, kot so priprava novih scen in nastavitve povezave v skupne seje z ostalimi uporabniki. Stranska vrstica je namenjena hitri interakciji z vsebino scene.

Za volumetrične objekte scene se doda nov podmeni, prikazan na sliki 3.5. Vse nastavitve je možno spreminjati v realnemu času, rezultati pa so nemudoma vidni na platnu.



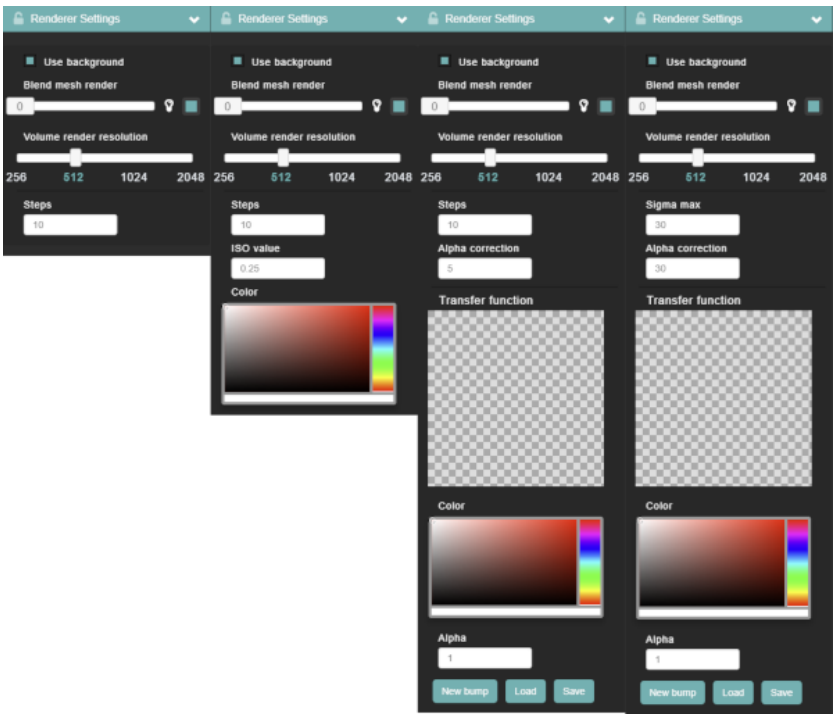
Slika 3.5: Stranska vrstica z zaprtim in odprtim podmenijem za volumetrične objekte.

Izdelan podmeni za volumetrične objekte je razdeljen na 4 področja.

Izbira *metode volumetričnega upodabljanja* (angl. volume rendering method) omogoča hitro preklapljanje med štirimi metodami upodabljanja orodja VPT, razloženimi v pregledu VPT orodja . Glede na izbiro metode se spremenijo tudi ponujene možnosti v ostalih 3 podmenijih.

Drugo področje, *nastavitve upodabljalnika* (angl. renderer settings), odpre nov podmeni, ki se nanaša na parametre izbrane metode. Možnosti za vse štiri metode so prikazane na sliki 3.6. Vsaka metoda ima svojo različico celotnega podmenija, nastavljivi parametri pa veljajo samo zanjo. Prvih nekaj nastavitev je univerzalnih vsem štirim metodam. Te so vključitev in izključitev ozadja, primer na sliki 4.6 , nastavitev vrednosti α zlivanja, vidnega na sliki 3.4, vključitev in izključitev uporabe Phongovega senčenja, izbira osnovne barve geometrije za posredno upodabljanje in pa ločljivost slikovnega okvirja metode VPT. Kakovost izrisa je sorazmerna z nastavljeno ločljivostjo, ki predstavlja število tekslov teksture slikovnega okvirja neposredne upodobitve.

To število pa je enako začetnemu številu svetlobnih žarkov vsake iteracije metod sledenja potem. Ostale nastavitve pa so specifične določeni metodi upodabljanja.



Slika 3.6: Nastavitve parametrov upodabljanja za metode projekcije največje intenzitete - MIP, nivojskih ploskev - ISO, emisijsko-absorbcijskega modela - EAM in Monte Carlo za sevalno enačbo z enkratnim sipanjem - MSC

V tretjemu področju so nastavitve tonskih slikarjev. Namen tonskega slikarja je skalacija visokega dinamičnega razpona barv v razpon [0, 1], ki ga je naprava zmožna prikazati. Delovanje obeh tonskih slikarjev je opisano v pregledu orodja VTP. Podprti sta nastavitvi obeh tonskih slikarjev, ki se izvedeta zaporedno; najprej Reinhardov, ki skrči dinamični razpon na [0, 1], nato pa še tonski slikar intenzitetnega obsega, ki dodatno obreže izbrani intenzitetni razpon in ga linearno preslika v [0, 1]. Ker je preslikava obeh preprosta, to ne predstavlja znatne računske zahtevnosti.

V četrtemu področju ima uporabnik dostop do alternativne geometrije

volumetričnega objekta. Ta geometrija lahko nastane z algoritmom Marching Cubes. Marching cubes algoritem je računsko zahteven in se ne izvaja v realnemu času, zato ima uporabnik na voljo gumb, ki sproži izračun. Naloga izračuna se doda v vrsto opravil *storitve za upravljanje nalog* (angl. task manager service). Upravitelj nalog izračuna ne izvaja v glavni niti, ki skrbi za uporabniški vmesnik, zato uporaba tega ni onemogočena. Uporabniku se prikaže grafični in tekstovni dialog, ki kaže potek izračuna. Po končanemu postopku se volumetričnemu objektu v sceni pripne alternativna dodatna geometrija. Volumetrični objekt obdrži osnovno geometrijo enotske kocke, uporabnik pa ima možnost preklapljanja med uporabo ene ali druge. Geometrija, na katero se volumetrični podatki upodabljajo, je lahko poljubna, zato je smiselno uporabniku ponuditi možnost uporabe neke lastne geometrije. Izračunano geometrijo pa bi se dalo uporabiti ali pa spreminjati tudi v drugih programih. Dodani sta možnosti izvoza in uvoza geometrije. V postopku upodabljanja je geometrija zapisana v binarnemu zapisu s plavajočo vejico z enojno natančnostjo s tri-krat 32 biti za posamezno oglišče. Uvoz in izvoz uporablja datoteke standardiziranega zapisa Wavefront OBJ [51]. Zapis OBJ je najbolj pogosto uporabljen format shranjevanja poligonskih mrež. Pri izvozu je potrebno paziti na velikost, saj se binarni podatki pretvorijo v tekstovne, kar pa zasede veliko več prostora. Pri omejitvi do 35 milijonov oglišč v postopku izračuna geometrije je lahko zapis velik tudi več gigabajtov. Tako velike datoteke ni možno učinkovito prenašati, saj klasične metode na uporabniški strani zadržujejo podatke v pomnilniku sistema. Maksimalno količino porabljenega spomina določa spletni brskalnik, pri presegu te količine pa onemogoči izvajanje spletne strani. Zapis OBJ je potrebno brati ali pa pisati neposredno v datotečni sistem. Ker za uporabniško stran spletne aplikacije ni splošnih rešitev, sta bili uporabljeni knjižnici iz spletnega repozitorija NPM: StreamSaver[2] za pisanje in LineNavigator[3] za branje.

[2]Jimmy Wärting. *StreamSaver.js*. Dosegljivo: `https://github.com/jimmywarting/StreamSaver.js`. [Dostopano: 21. 2. 2019]. 2016.

[3]Anton Purin. *line-navigator*. Dosegljivo: `https://github.com/anpur/line-navigator`. [Dostopano: 21. 2. 2019]. 2015.

StreamSaver uporabi trik, kjer s pojavnim oknom (angl. pop-up site), ki se nikoli ne prikaže, simulira tok podatkov in jih v realnemu času zapisuje v datotečni sistem.

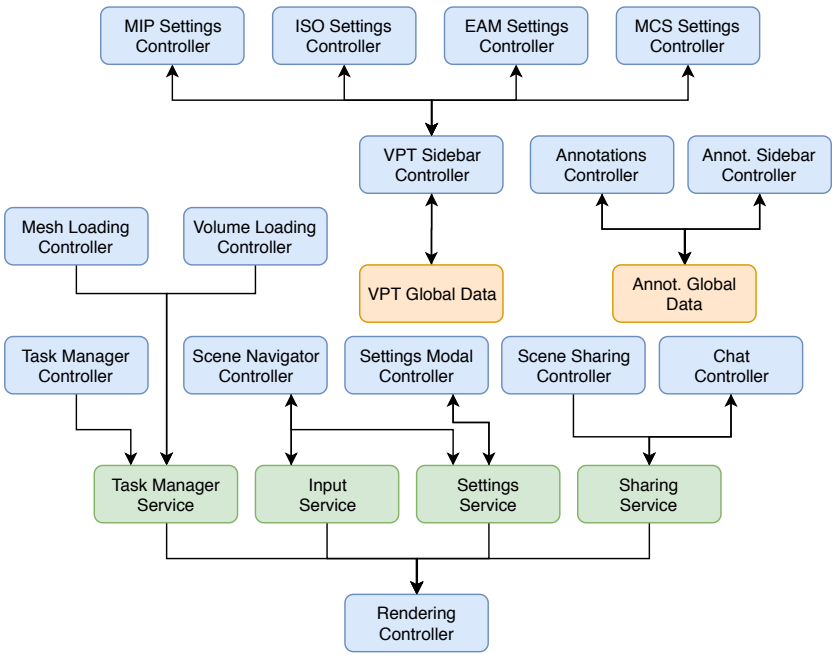
Line navigator pa bere datoteko v manjših kosih in simulira bralni pretok.

Slabost takšnega pristopa je omejena hitrost branja in pisanja. Vzrok za ozko grlo je v sinhroniziranem zaporednem enkodiranju binarnih podatkov v tekstovne oziroma pri dekodiranju manjših tekstovnih blokov podatkov nazaj v binarne. Postopek poteka na enemu jedru CPE. Drugo ozko grlo pa je tudi strojna hitrost prenosa podatkov v in iz trajnega spomina.

Na sistemu s CPE Intel i7 4710HQ in Crucial mx100 SSD diskom je možno doseči hitrosti 30 MB/s izvoza in 10MB/s uvoza. Dodana optimizacija je krajšanje zapisa oglišč na fiksno število decimalnih mest. Implementirana je omejitev na 6 mest, kar prihrani približno 40 odstotkov prostora. Na žalost dodatno upočasni izvoz za približno enak faktor zaradi dodatnih operacij rezanja teksta vsake koordinate vsakega oglišča.

Vsako od štirih področij v podmeniju za volumetrične podatke na stranski vrstici ima tudi stikalo zaklenitve. To je uporabljeno za namene sinhronizacije področij menijev več uporabnikov v seji oddaljenega sodelovanja. Oddaljeno sodelovanje je bolj natančno razloženo v nadaljevanju.

Shema na sliki 3.7 arhitekture prikazuje dodane module. Vsak upodabljalnik ima za svoje nastavitve svoj upravljalnik, ki je gnezden v upravljalnik stranske vrstice. Stanje uporabniškega vmesnika se hrani v globalnemu stanju volumetričnih objektov. Hkrati vsebuje tudi stanja drugih uporabnikov s sistemom oddaljenega sodelovanja.



Slika 3.7: Shema Med3D arhitekture z dodatkom uporabniških funkcij.

3.3 Oddaljeno sodelovanje

V digitalni dobi se pogosto poraja vprašanje, kako spraviti informacije iz ene naprave na drugo. Razvoj interneta omogoči enostaven prenos informacij na podatkovni ravni. Na aplikacijski ravni pa je še vedno dolžnost razvijalcev, da to možnost podprejo. Za aplikacijo Med3D je bil zamišljen preprost sistem replikacije in sinhronizacije podatkov med uporabniki. Ideja je zasnovana za uporabo v medicini. Specialisti bi lahko s sinhronizirano vizualizacijo podatkov na hitro uskladili informacije medicinskega primera in si izmenjali mnenja.

Med3D že ima implementacijo svoje rešitve oddaljenega sodelovanja. Za komunikacijo je bila izbrana tehnologija WebSockets. Preko *aplikacijskega programskega vmesnika* (angl. application programming interface - API) je omogočen dostop do paketa orodij, metod in protokolov za komunikacijo med aplikacijami.

Pozitivne lastnosti te tehnologije so:

- dvosmerna komunikacija med sodelujočimi;
- pošiljanje in prejemanje sporočil sta dva neodvisna procesa in se lahko izvajata istočasno;
- tehnologija temelji na transportnemu protokolu TCP, ki zagotavlja preprečevanje in razreševanje napak pri prenosu.

Načela delovanja sistema oddaljenega sodelovanja aplikacije Med3D smo že opisali v pregledu aplikacije, zato se zdaj posvetimo nadgradnji. Volumetrični objekti se replicirajo po podobnemu postopku kot objekti z mrežno geometrijo. Razlika je v tem, da ni potrebno replicirati geometrije, saj je ta v osnovi vedno enotska kocka. Izračunane geometrije marching cubes se ne pošilja. Poglavitni razlog je, da to ni možno v realnemu času, saj je količina podatkov prevelika in zahteva preveč procesiranja. Če bi vsak uporabnik hranil vse geometrije, ki jih uporabljajo ostali uporabniku, bi hitro zmanjkalo spomina okna brskalnika. Če pa se določi enotna alternativna geometrija, potem so jo vsi uporabniki scene prisiljeni uporabljati. Boljši pristop je, da vsak uporabnik prosto razpolaga z alternativno geometrijo lokalno, uporabniki pa jo lahko še vedno uskladijo z uporabo enakih nastavitev pri uporabi algoritma Marching Cubes.

Potrebno je sinhronizirati še nastavitve izrisa. To so vse nastavitve, s katerimi uporabnik razpolaga v podmeniju za volumetrične podatke na stranski vrstici. Nesmiselno bi bilo, da bi obstajala le ena različica nastavitev, ki jo uporabljajo vsi uporabniki. Zato se omogoči hranjenje več različic nastavitev. Vsaka različica nastavitev naj bo vezana na objekt kamere. V skupni seji lahko kamero nadzoruje samo njen lastnik, vsak udeleženec pa ima vsaj eno lastno kamero. Enako naj velja za pakete nastavitev izrisa volumetričnih podatkov. Dokler je uporabnik na svoji kameri, lahko prosto spreminja nastavitve. Ko uporabnik preklopi na kamero drugega udeleženca, se mu nadzor nad nastavitvami zaklene, nastavitve pa se začnejo, v realnemu času, sinhronizirati z nastavitvami lastnika kamere. Da se uporabniku omogoči

malo več svobode, ima za vsako od štirih področij nastavitev stikalo zaklepa. Z onemogočenim zaklepom lahko uporabnik nadzoruje tudi tuje nastavitve, a spremembe ostanejo lokalne. Ob ponovnemu zaklepu se nastavitve ponovno sinhronizirajo.

Poglavje 4

Analiza uspešnosti

V tem poglavju vrednotimo, delo opravljeno v sklopu diplomskega dela. Vrednotenje je razdeljeno na podpoglavja, za vsako od novih uporabniških funkcij. Vsaki je podan primer uporabe in analiza smiselnosti. Analiza novosti se neposredno nanaša na zastavljene cilje v uvodnemu poglavju.

4.1 Upodabljanje volumetričnih podatkov s kombiniranim pristopom

Osrednja dodana funkcija je možnost izrisa volumetričnih podatkov. Kot je razloženo v poglavju implementacije, upodabljanje volumetričnega objekta poteka v dveh korakih: naprej neposreden izris v teksturo, nato pa upodobitev geometrije. Smiselnost pristopa se lahko oceni z naslednjimi kriteriji.

- Ali je ohranjena uporabniška interakcija?
- Ali trpi kakovost upodobitve?
- Kakšna je dodana računska zahtevnost?

4.1.1 Uporabniška interakcija

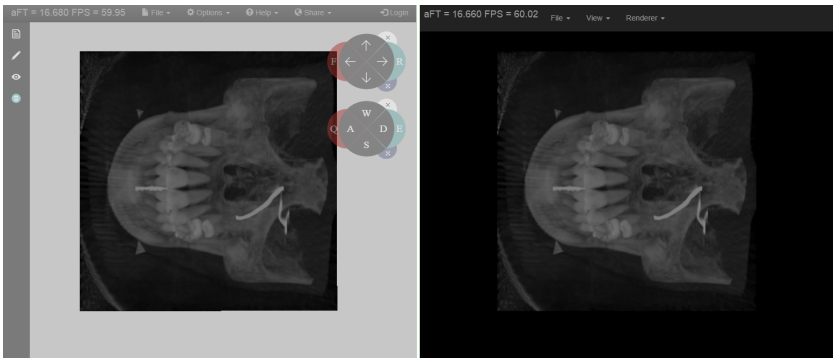
Uporabnik ima možnost vnesti volumetrične podatke, aplikacija pa mu pripravi novo sceno z volumetričnim objektom. Kot velja za običajne objekte poligonskih mrež, ima uporabnik tudi tu enake možnosti interakcije z objektom. Uporabnik z uporabo miške, tipkovnice, grafičnega navigatorja ali pa igralnega ploščka nadzira položaj in pogled kamere. Razlika se pojavi ob spremembi pogleda. Vsakič ko pride do spremembe pogleda, je potrebno volumetrične podatke začeti upodabljati v novemu pogledu. Kot je razloženo v poglavju pregleda orodja VPT-ja, se upodobitev izboljšuje iterativno. Pri spremembi pogleda torej pride do ponastavitve napredka postopka upodobitve. Če upodobitve ni možno ponovno izračunati v zmernemu času, bi bilo to nadležno za uporabnika, saj bi bilo premikanje pogleda nezvezno.

Hitrost postopne izboljšave je odvisna od števila iteracij metode VPT v vsakemu obhodu vsakega grafičnega cevovoda aplikacije. Z drugimi besedami, koliko časa se sme porabiti za neposreden izris z metodo VPT, da preostane še zadosti časa za izris geometrije. Pri previsoki zahtevnosti metode se bo upočasnil celoten Med3D cevovod, uporabnik pa bo opazil manj tekočo animacijo na zaslonu, pri premajhni zahtevnosti pa bo uporabnik moral opazovati postopen izris volumetričnih podatkov. V obeh primerih bi to pomenilo okrnjeno uporabniško izkušnjo. Za dobro uporabniško interakcijo je torej pomembno ravnotežje med hitro upodobitvijo in hitrostjo izvajanja celotnega grafičnega cevovoda. V podpoglavju Dodana računska zahtevnost bomo pokazali, da lahko tudi mobilne naprave, ki se štejejo kot manj zmogljive, uporabljajo veliko iteracij za hitro konvergiranje k absolutni sliki, brez bistvene upočasnitve cevovoda.

4.1.2 Kakovost upodobitve

V koraku projiciranja upodobitve volumetričnih podatkov na geometrijo objekta ne pride do izgube kakovosti, saj gre za surjektivno preslikavo vrednosti točk teksture na fragmente geometrije. Na sliki 4.1 je primerjava

med neposrednim izrisom volumetričnih podatkov v predstavitveni aplikaciji orodja VPT in med posrednim izrisom v aplikacij Med3D. V obeh primerih gre za enake vhodne podatke z enakimi parametri pogleda in izrisne metode, izgub pa ni.



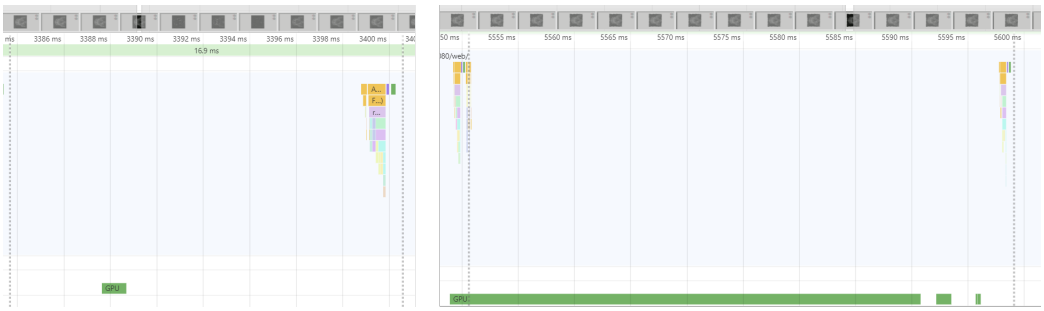
Slika 4.1: Primerjava kakovosti upodobitve z Med3D (levo) in VPT predstavitveno aplikacijo (desno).

4.1.3 Dodana računska zahtevnost

Grafični cevovod aplikacije Med3D je bil za namen upodabljanja volumetričnih objektov nadgrajen z grafičnim cevovodom metod VPT. Smiselnost integracije je potrebno potrditi s stališča dodane računske zahtevnosti. Zanima nas predvsem, kakšnega reda je skupna računska zahtevnost in kako vpliva računska zahtevnost metod VPT na preostanek cevovoda in obratno. Pričakovane računske zahtevnosti v tem primeru ni smiselno izračunavati, saj je izvajanje senčilnikov na GPE močno paralelizirano. Stopnja paralelizacije pa je odvisna od zmogljivosti strojne opreme, nastavitev programske opreme in velikostjo vhodnih podatkov. Smiselnost smo zato raje ocenili na podlagi meritev na treh različno sposobnih konfiguracijah strojne opreme. Za meritve sta bila uporabljena mobilni telefon OnePlus 3T z grafično kartico Adreno 530 in prenosni računalnik Asus g550jk. Prenosni računalnik razpolaga z dvema grafičnima karticama. To sta Intel HD Graphics 4600 in Nvidia GTX 850m. Izbrane konfiguracije smiselno pokrivajo razpon, od nizke zmogljivosti

3D pohitritve do srednje visoke.

Optimalno bi bilo meriti dolžino izvedbe posameznih senčilnikov. Za ta namen je bila zasnovana WebGL razširitev `EXT_disjoint_timer_query_webgl2`, a je v času izdelave diplomskega dela iz varnostnih razlogov ne podpira noben brskalnik. Drugi pristop, ki smo ga preizkusili, je z uporabo vgrajenega *analizatorja* (angl. profiler) v brskalniku Chrome. Ta pristop omogoča vizualno razdelitev dela med CPU in GPE. Vzorčen primer za izris ene zaslonske slike je predstavljen na sliki 4.2.



Slika 4.2: Primerjava profila izrisa ene zaslonske slike s Chromovim *analizatorjem* (angl. profiler) za primer nizke (levo) in visoke zahtevnosti (desno) metode upodabljanja in geometrije. Prekinjene navpične črte predstavljajo vertikalno sinhronizacijo med slikami na zaslonu. Spodnja zelena črta predstavlja zasedenost GPU, srednji stolpič pa trajanje posameznih JavaScript funkcij na CPE.

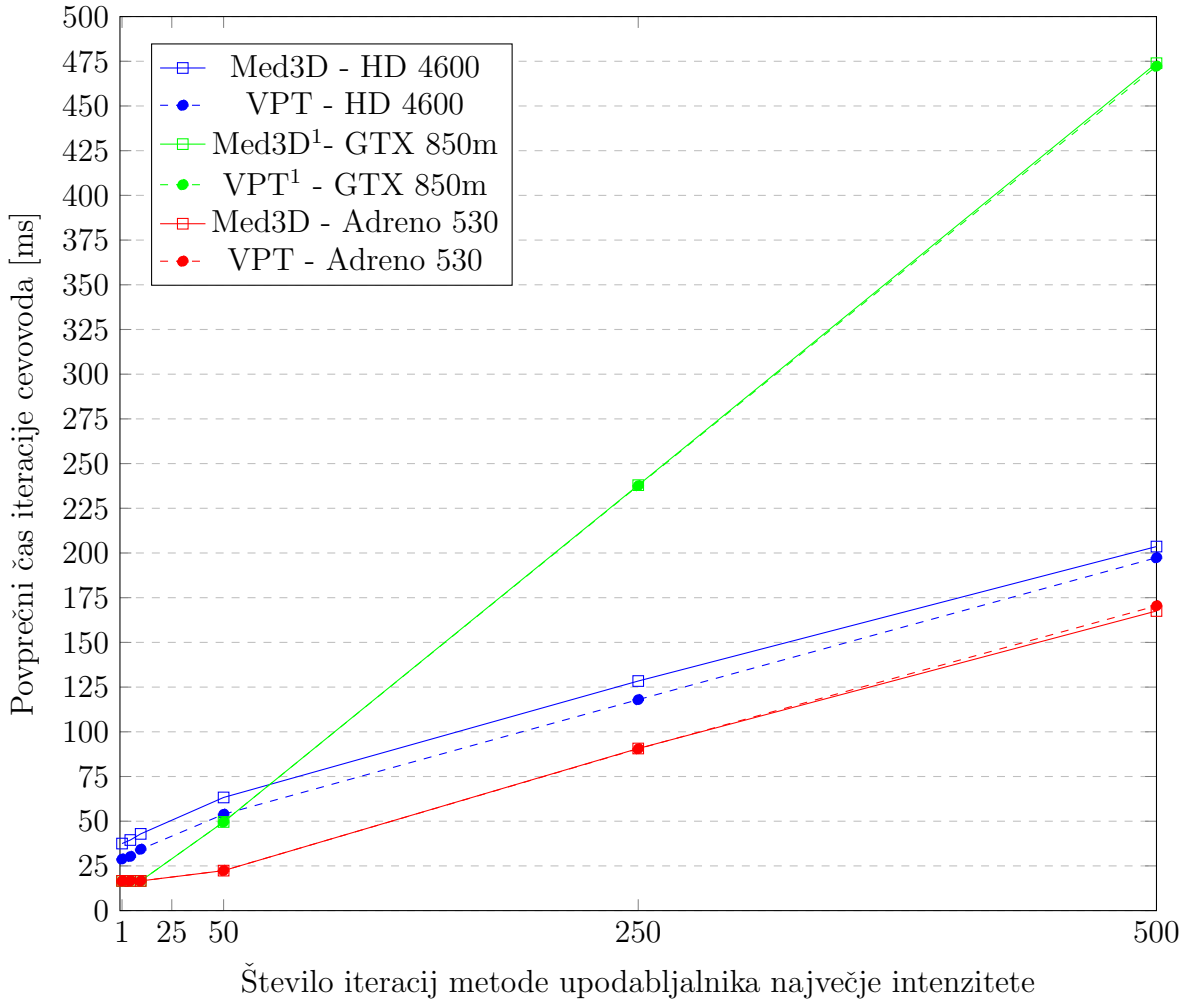
Z uporabo analizatorja je možno pokazati razdelitev dela med CPE in GPE. Na sliki 4.2 sta izmerjena primera z nizko in visoko računsko zahtevnostjo upodabljanja. Ugotovitev je, da je CPE zasedenost konstantna (∼1ms) in neodvisna od zahtevnosti upodabljanja. Zasedenost GPE (od ∼1ms dalje) pa je odvisna zahtevnosti izrisa. Z enako ugotovitvijo na vseh treh strojnih konfiguracijah lahko predpostavimo, da je GPE ozko grlo pri izrisovanju, neposrednih vplivov drugih komponent pa ni. Z uporabo analizatorja ni bilo možno ločiti časov delovanja posameznih senčilnikov, prav tako ni bilo možno oceniti povprečnega izrisnega časa zaradi Chromove notranje logike

razporeditve opravil na GPE in vplivov vertikalne sihronizacije.

Na žalost dobrih alternativ za merjene hitrosti izvajanja WebGL senčilnikov ni. Za metodologijo meritev je uporabljen drug pristop. Meri se čas poteka celotnega grafičnega cevovoda. Bolj natančno, meri se čas med posameznimi iteracijami metode `requestAnimationFrame`, ki jo proži brskalnik. Na tak način izmerjen čas, ne predstavlja nujno samo trajanja delovanja vseh senčilnikov cevovoda, saj vključuje tudi čas priprave in zaključka opravil.

Metoda `requstAnimationFrame` vedno poganja cevovod pri maksimalni zmogljivosti, vse do meje 60 sličic na sekundo oziroma do 16,6 ms za izvedbo ene iteracije celotnega cevovoda. Vse meritve s časom iteracije okoli 16,6ms torej ne predstavljajo maksimalne zasedenosti zmogljivosti strojne opreme in zato niso pomembne za primerjalne namene. Po potrebi je možno povečati zahtevnost metod VPT s povečanjem števila iteracij. Zahtevnost geometrije pa se povečuje s številom trikotnikov trikotnikov geometrije, na katero se volumetrična podoba projicira. Število iteracij metod VPT in število trikotnikov sta torej glavni, domnevno neodvisni spremenljivki. Za namen konsistentnih meritev smo uporabili enake nastavitve in enake vhodne podatke. Izbrali smo volumetričen posnetek lobanje, upodabljalno metodo projekcije največje intenzitete in statičen pogled kamere. Med različnimi strojnimi konfiguracijami pa se spremeni samo velikost platna. Prenosnik ima platno v brskalniku veliko 1536×734 pikslov, mobilni telefon pa 360×560 pikslov. Meritve se izvajajo v brskalniku Chrome z verzijo pogona Chromium 71.0, vzorči pa se povprečni čas 360 zaporednih izrisanih sličic.

Za izhodišče smo izmerili razliko hitrosti cevovoda v aplikaciji Med3D in cevovoda predstavitvene aplikacije orodja VPT. V enakih pogojih se meri povprečne čase 360 izrisov na platno za različno število iteracij metode največje intenzitete. Pri strojni konfiguraciji z namensko grafično kartico GTX 850m se tokrat izjemoma meri pri 100 krat višjih iteracijah, da smo dosegli mejo maksimalne zasedenosti.



Slika 4.3: Primerjava hitrosti povprečnega časa izrisa ene slike na platno, med aplikacijama Med3D in VPT pri različnih strojnih konfiguracijah. Med3D uporabi projekcijo na geometrijo enotske kocke z 12 trikotniki.

Na grafu 4.3 je predstavljena dolžina izvajanja cevovoda Med3D in VPT. Bistvena razlika je, da ima Med3D cevovod dodaten korak izrisa geometrije in projiciranja VPT upodobitve nanjo. Za grafični kartici Nvidia GTX 850m in Adreno 530 ta korak ne predstavlja izmerljive dodane zahtevnosti. V primeru HD 4600 je rezultat dodane zahtevnosti konstanten povprečen

[1]Primer z GTX 850m uporablja 100 krat višje iteracije.

pribitek 8,75ms. Izkaže se, da je grafična kartica HD 4600 nizko zmogljiva za namene 3D pospeševanja s tehnologijo WebGL, saj že čas cevovoda, samo za upodobitve kocke, brez upodobitve volumetričnih podatkov, traja 34,2ms oziroma 27,9ms brez Phongovega senčenja. Zahtevnost izrisne metode iz orodja VPT torej ne vpliva na red računske kompleksnosti koraka projiciranja na geometrijo.

Dokazati je potrebno še obratno: vpliv projekcije VPT upodobitve na zahtevnosti izrisa geometrije za različne težavnosti poligonske mreže.

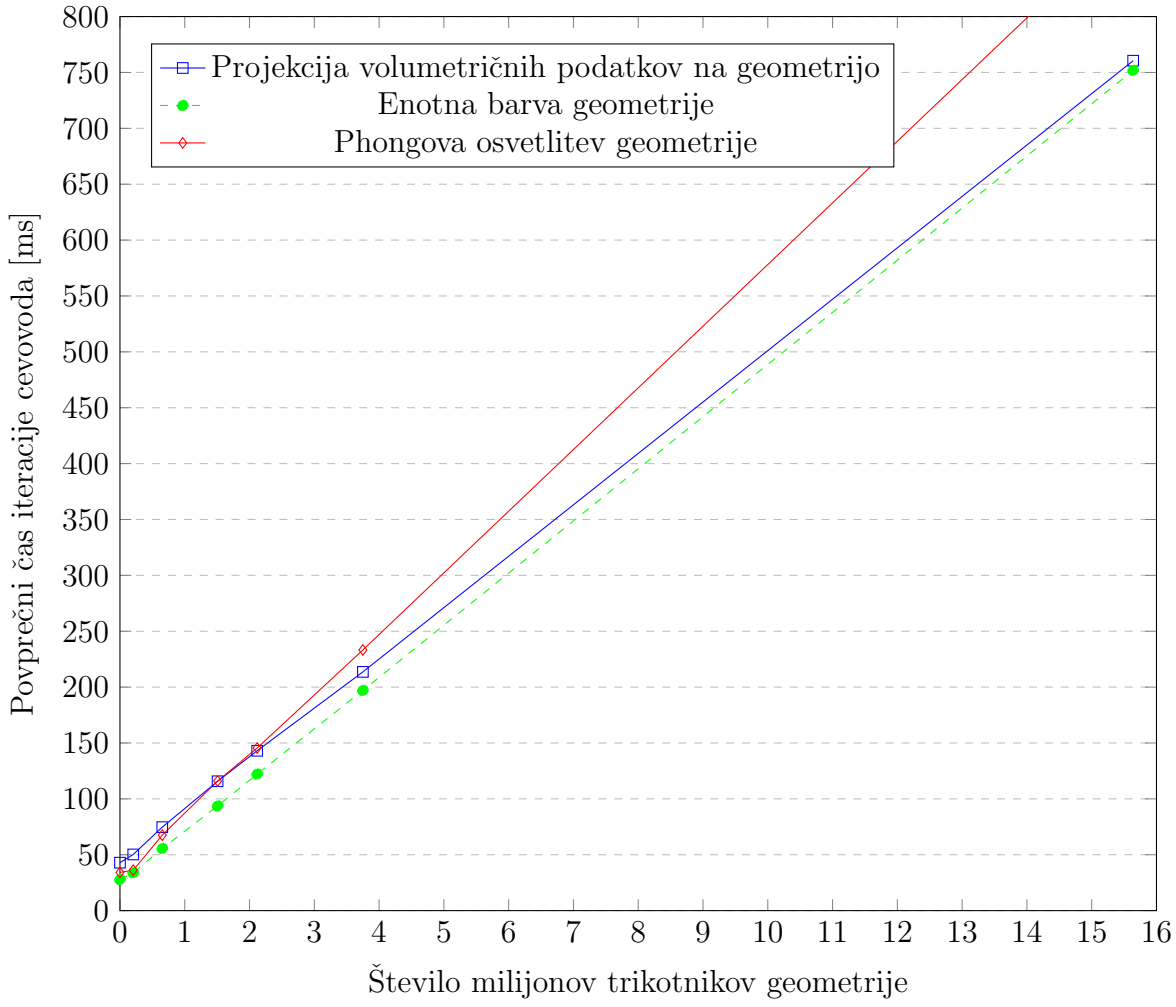
Za drugo meritev se meri čas iteracije celotnega cevovoda Med3D glede na število trikotnikov geometrije, pridobljene z algoritmom Marching cubes za vrednosti ISO $0,4$, $0,3$, $0,2$, $0,15$, $0,10$ in $0,05$. Število trikotnikov za naštete vrednosti ISO je vedno konsistentno, in sicer 204674, 653302, 1507328, 2117929, 3749167, 15636500. Meri pa se tri ločene primere:

1. Primer z upodabljanjem volumetričnih podatkov in s projiciranjem upodobitve na geometrijo. Uporabljena je metoda projekcije največje intenzitete pri konstantnemu številu iteracij 10.

2. Primer brez upodabljanja volumetričnih podatkov in brez projiciranja s fiksno barvo poligonov.

3. Ponovitev drugega, a s Phongovim modelom lokalnega osvetljevanja.

Na grafu 4.4 so predstavljeni rezultati meritev za grafično kartico HD 4600. Drugi dve strojni konfiguraciji sta iz grafa izpuščeni, saj vpliv števila oglišč ni razviden, vse do najvišjih števil trikotnikov. Pri GTX 850m se izrisni čas poveča nad 16,6 ms šele pri 16 milijonih trikotnikov, z enako vrednostjo za vse tri primere merjenja. V primeru Adreno 530 se izrisni čas poveča šele pri dveh milijonih trikotnikov, 16 milijonov trikotnikov pa ni možno upodobiti. Bistveno višjega števila trikotnikov pa tudi ni mogoče meriti, saj pride do omejitev spomina.



Slika 4.4: Primerjava časa trajanja cevovoda glede na število trikotnikov geometrije za projekcijo na geometrijo, za enotno barvo geometrije in za Phongovo osvetljitev geometrije. Velja za GPE HD 4600.

Na grafu 4.4 je razvidno, da se računska zahtevnost za projekcijo teksture in za enotno barvo oglišč povečuje linearno glede na število trikotnikov.

To je možno razložiti z delovanjem WebGL grafičnega cevovoda. Število trikotnikov geometrije je sorazmerno s številom oglišč, ki jih je potrebno obravnavati v koraku senčenja oglišč. Pri tem je vidno linearno naraščanje porabljenega časa, saj je potrebno upoštevati več oglišč. V koraku rasterizacije trikotnikov se to večje število trikotnikov prevede na enako količino

zaslonskih točk, saj je naslovni prostor odvisen od velikosti objekta na zaslonu in ne geometrične kompleksnosti. Posledično je v naslednjemu koraku, v senčenju fragmentov, število vhodnih rasteriziranih fragmentov neodvisno od zahtevnosti geometrije. Zato je cena projekcije teksture prav tako konstantna, ne glede na geometrijo.

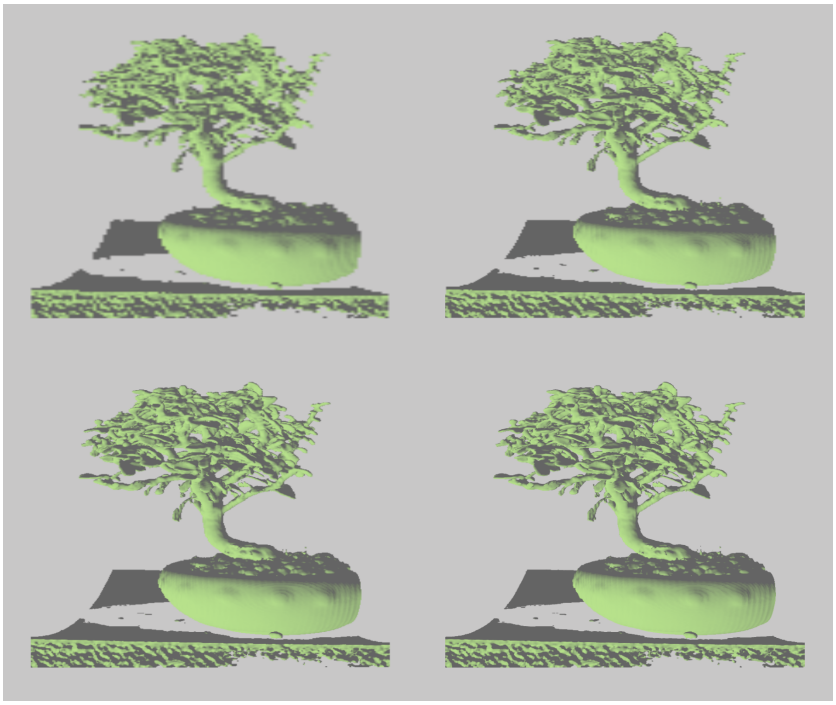
Sklep je, da so VPT senčilniki neposrednega upodabljanja in senčilniki geometrije medsebojno računsko neodvisni. Skupna računska zahtevnost celotnega cevovoda pa je enaka vsoti zahtevnosti enih in drugih senčilnikov. Skupen red računske zahtevnosti je še vedno $O(n)$ glede na število iteracij metod VPT in števila trikotnikov geometrije. Implementacija je z vidika računske kompleksnosti smiselna. Smiselna je tudi z vidika platformne neodvisnosti, kar se odraža v hitrem in kakovostnemu izrisu na mobilnemu telefonu.

4.2 Uporabniški vmesnik

V uporabniškem vmesniku ima uporabnik nov zavihek na stranski vrstici, ki omogoča dostop do menija nastavitev volumetričnih objektov. Meni je viden na sliki 3.5. Nastavitev je na omejeni površini kar veliko, zato so te združene v zavihke glede na funkcionalnost. Tak pristop je smiseln, saj je vsaka nastavitev dosegljiva v največ 2 klikih. Sprememba vsake nastavitve je nemudoma vidna na sliki na platnu, zato je dobra tudi stopnja odzivnosti.

Množica ponujenih nastavitev je unija funkcij, ki so bile že predhodno vsebovane v Med3D ali v VPT. Implementiranih je tudi nekaj novih možnosti.

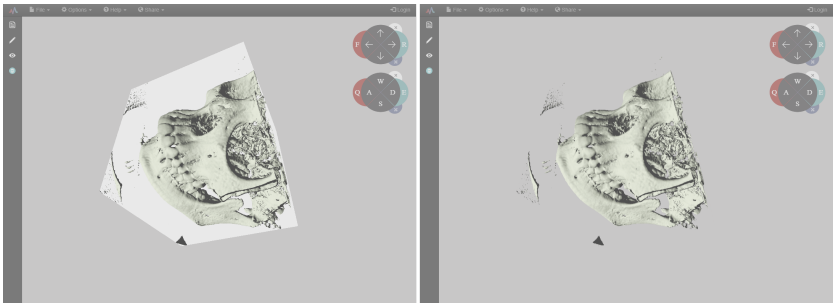
- Za vsako upodabljalno metodo je možno izbrati velikost slikovnih okvirjev. Primerjava izrisa pri različnih velikostih slikovnega okvirja je prikazana na sliki 4.5.



Slika 4.5: Primerjava kakovosti upodobitve pri različnih velikostih slikovnih okvirjev. Velikosti: 256×256 levo zgoraj, 512×512 desno zgoraj, 1024×1024 levo spodaj, 2048×2048 desno spodaj.

Višja velikost oziroma ločljivost slikovnega okvirja pozitivno vpliva na jasnost upodobljene slike, a na račun dodatne porabe računskih zmogljivosti. Za uporabnika je izbira ločljivosti upodobitve smiselna, saj pokriva večji spekter naprav glede na zmogljivost in velikost platna v brskalniku.

- Na sliki 4.1 se opazi, da se tudi ozadje preslika na geometrijo. Uporabnik ima možnost neuporabe ozadja za bolj uniform videz, ki skrije obliko geometrije. Primer je podan na sliki 4.6.



Slika 4.6: Upodobitev človeške lobanje na geometrijo enotske kocke z metodo nivojskih ploskev z in brez ozadja.

- V zadnjemu zavihku je podana še možnost izbire uporabe osnovne geometrije enotske kocke ali pa alternativne geometrije. Primer z aproksimirano geometrijo je prikazan na sliki 3.4. Alternativna geometrija se lahko izračuna z algoritmom Marching cubes ali pa se uvozi. Vsako alternativno geometrijo pa je možno izvoziti za namen uporabe ali modifikacije v drugih aplikacijah. Uporaba dobre aproksimacije geometrije skupaj z uporabo *alpha zlivanja* predstavlja nekakšen kombiniran izris. Kombiniran način ima lahko višjo informacijsko vrednost. V večini primerov si je lažje predstavljati globino objekta na senčeni geometriji, medtem ko ima neposredna upodobitev več informacij o vsebini znotraj geometrije. Pristop omogoča tudi raziskovanje notranjosti volumna s sprehodom kamere čezenj. Ta možnost ni vedno mogoča pri uporabi samo posrednega ali pa neposrednega pristopa.

Skupno je uporabniku podan širok spekter nastavitev, ki omogočajo predstavitev vhodnih volumetričnih podatkov na zelo veliko načinov.

4.3 Deljenje parametrov v oddaljenemu sodelovanju

V skupni seji z več uporabniki je dodana možnost delitev nastavitev izrisa. Vsak nabor nastavitev je vezan na objekt kamere. Nabor nastavitev in pogleda kamere pa nadzoruje lastnik kamere. Vsak uporabnik si lasti vsaj eno kamero. V realnemu času se replicirajo vse spremembe razen alternativnih geometrij volumetričnih objektov v sceni. Nezmožnost posredovanja alternativnih geometrij je sicer omejitev.

Naš sistem oddaljenega sodelovanja omogoča enostavno repliciranje podobe na platnu več uporabnikov. Pri profesionalni rabi se zdi situacija, kjer bi uporabniki nujno potrebovali identično sliko na zaslonu, precej verjetna. To je torej zelo uporabna funkcionalnost, saj bi bilo sicer nemogoče točno uskladiti prikazane slike. Ostale informacije si lahko še vedno izmenjajo preko deljenih anotacij ali pa pogovora v skupnemu tekstovnemu kanalu.

Poglavje 5

Zaključek

V diplomskem delu smo predstavili nadgradnjo aplikacije Med3D. Ta zdaj omogoča bistveno širši nabor zmogljivosti, več kot primerljiv z obstoječimi rešitvami, in je zato korak bližje profesionalni rabi v medicini. Izdelan pristop upodabljanja združuje pozitivne lastnosti preprostosti in interaktivnosti posrednega načina upodabljanja s kompleksnejšim, informacijsko in fizikalno bolj reprezentativnim pristopom neposrednega upodabljanja. Rezultati dokazujejo računsko učinkovitost, kar omogoča več kot zadovoljivo rabo na zmogljivostno šibkejših napravah, hitri osebni računalniki pa imajo možnost povečati stopnjo zahtevnosti in s tem dodatno izboljšati na kakovost izrisane slike.

Razvit uporabniški vmesnik omogoča hiter dostop do skoraj neomejenega števila kombinacij konfiguracij upodabljanja, ki se instantno aplicirajo na podobo na platnu. Ohrani se tudi obstoječe funkcije, risanja in postavljanja 3D anotacij neposredno na površino objekta. Skupno je nivo interaktivnosti visok, hkrati pa nove možnosti podpira tudi sistem oddaljenega sodelovanja, ki je zmožen v realnemu času prenesti uporabnikove vnose na ostale uporabnike.

Zastavljene cilje iz uvodnega poglavja smo dosegli. Med3D zdaj ponuja nekaj unikatnih funkcionalnosti, ki jih nima nobena druga programska rešitev. Vseeno pa je še veliko prostora za izboljšave.

5.1 Nadaljnje delo

Predstavljen je le en korak evolucije aplikacije Med3D. V prihodnosti se bosta projekta Med3D in VPT naprej razvijala neodvisno, naš način integracije pa bo omogočal preprosto integracijo nadaljnjih posodobitev.

Z natančnejšimi informacijami o izvajanju senčilnikov bi bilo možno dinamično spreminjati zahtevnost različnih delov cevovoda, in s tem prihraniti na zmogljivostih, ki se jih uporabi za druge namene. S takšnim sistemom, bi bilo mogoče optimizirati celotno sceno objektov, kjer bi se volumetrični objekti upodabljali s točno tako zahtevnostjo, da uporabnik še ne opazi varčevanja zmogljivosti.

Nove generacije grafičnih pospeševalnikov obljubljajo strojno podporo ekskluzivno za metode sledenja potem, kar bi bilo uporabno tudi za naš primer. Počakati pa bo potrebno, da postanejo na voljo tudi spletnim tehnologijam.

Rešiti moramo tudi problem prenosa velikih podatkov. To so veliki volumetrični podatki in geometrije, ki jih trenutno ni možno prenašati v realnemu času. To bi lahko dosegli z neposredno povezavo med uporabniki. S sistemom, ki temelji na soglasju, bi si uporabniki lahko pošiljali vse vrste podatkov. Tako bi zaobšli nekatere omejitve in omogočili nemoteno uporabo aplikacije tudi med zamudnim prenosom. Seveda pa je potrebno pomisliti tako na varnost kot tudi na anonimnost uporabnikov.

Literatura

[1] *ANGLE*. Dosegljivo: https://opensource.google.com/projects/angle. [Dostopano: 1. 3. 2019].

[2] *AngularJS*. Dosegljivo: https://angularjs.org/. [Dostopano: 25. 2. 2019].

[3] James F. Blinn. "Models of Light Reflection for Computer Synthesized Pictures". V: *SIGGRAPH Comput. Graph.* 11.2 (jul. 1977), str. 192–198. ISSN: 0097-8930. DOI: 10.1145/965141.563893. URL: http://doi.acm.org/10.1145/965141.563893.

[4] Wayne Carlson. *A Critical History of Computer Graphics and Animation*. Dosegljivo: https://web.archive.org/web/20070405172134/http://accad.osu.edu/~waynec/history/lessons.html. [Dostopano: 26. 2. 2019]. 2003.

[5] *Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)*. Dosegljivo: https://creativecommons.org/licenses/by-sa/4.0/deed.en. [Dostopano: 4. 3. 2019].

[6] Carl R Crawford in Kevin F King. "Computed tomography scanning with simultaneous patient translation". V: *Medical physics* 17.6 (1990), str. 967–982.

[7] Akio Doi in Akio Koide. "An efficient method of triangulating equivalued surfaces by using tetrahedral cells". V: *IEICE TRANSACTIONS on Information and Systems* 74.1 (1991), str. 214–224.

[8] Robert A. Drebin, Loren Carpenter in Pat Hanrahan. "Volume Rendering". V: *SIGGRAPH Comput. Graph.* 22.4 (jun. 1988), str. 65–74. ISSN: 0097-8930. DOI: 10.1145/378456.378484. URL: http://doi.acm.org/10.1145/378456.378484.

[9] Thomas Fogal in Jens Krüger. "Tuvok, an Architecture for Large Scale Volume Rendering". V: *Proceedings of the 15th International Workshop on Vision, Modeling, and Visualization*. Nov. 2010. URL: http://www.sci.utah.edu/~tfogal/academic/tuvok/Fogal-Tuvok.pdf.

[10] Andreas Franke in sod. "EAE/ASE Recommendations for Image Acquisition and Display Using Three-Dimensional Echocardiography". V: *European Heart Journal - Cardiovascular Imaging* 13.1 (jan. 2012), str. 1–46. ISSN: 2047-2404. DOI: 10.1093/ehjci/jer316. eprint: http://oup.prod.sis.lan/ehjcimaging/article-pdf/13/1/1/7139463/jer316.pdf. URL: https://dx.doi.org/10.1093/ehjci/jer316.

[11] M. Friendly. "A Brief History of Data Visualization". V: *Handbook of Computational Statistics: Data Visualization*. Ur. C. Chen, W. Härdle in A Unwin. Zv. III. (In press). Heidelberg: Springer-Verlag, 2006, ???–???

[12] *Git*. Dosegljivo: https://git-scm.com/. [Dostopano: 1. 3. 2019].

[13] *OpenGL Shading Language Overview*. Dosegljivo: https://www.opengl.org/sdk/docs/tutorials/ClockworkCoders/glsl_overview.php. [Dostopano: 28. 2. 2019].

[14] H. Gouraud. "Continuous Shading of Curved Surfaces". V: *IEEE Transactions on Computers* C-20.6 (jun. 1971), str. 623–629. ISSN: 0018-9340. DOI: 10.1109/T-C.1971.223313.

[15] William Hibbard in David Santek. "Interactivity is the key". V: *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*. ACM. 1989, str. 39–43.

[16] Michael Horvath. *Axonometric projection of cube above plane*. [Dostopano: 1. 3. 2019], [Licenca: https://creativecommons.org/licenses/by-sa/4.0/deed.en]. URL: https://commons.wikimedia.org/wiki/File:Axonometric_projection_of_cube_above_plane.svg#filelinks.

[17] *Indigo2 IMPACT: Medical Imaging*. Dosegljivo: http://www.futuretech.blinkenlights.nl/medical.html. [Dostopano: 27. 2. 2019].

[18] *JSON*. Dosegljivo: https://www.json.org/. [Dostopano: 28. 2. 2019].

[19] James T. Kajiya. "The Rendering Equation". V: *SIGGRAPH Comput. Graph.* 20.4 (avg. 1986), str. 143–150. ISSN: 0097-8930. DOI: 10.1145/15886.15902. URL: http://doi.acm.org/10.1145/15886.15902.

[20] Arie Kaufman in Klaus Mueller. "7 - Overview of Volume Rendering". V: *Visualization Handbook*. Ur. Charles D. Hansen in Chris R. Johnson. Burlington: Butterworth-Heinemann, 2005, str. 127–174. ISBN: 978-0-12-387582-2. DOI: https://doi.org/10.1016/B978-012387582-2/50009-5. URL: http://www.sciencedirect.com/science/article/pii/B9780123875822500095.

[21] D Krakow in sod. "Use of three-dimensional ultrasound imaging in the diagnosis of prenatal-onset skeletal dysplasias". V: *Ultrasound in Obstetrics and Gynecology: The Official Journal of the International Society of Ultrasound in Obstetrics and Gynecology* 21.5 (2003), str. 467–472.

[22] Thomas Kroes, Frits H. Post in Charl P. Botha. "Exposure Render: An Interactive Photo-Realistic Volume Rendering Framework". V: *PLOS ONE* 7.7 (jul. 2012), str. 1–10. DOI: 10.1371/journal.pone.0038586. URL: https://doi.org/10.1371/journal.pone.0038586.

[23] Philippe Lacroute in Marc Levoy. "Fast Volume Rendering Using a Shear-warp Factorization of the Viewing Transformation". V: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '94. New York, NY, USA: ACM,

1994, str. 451–458. ISBN: 0-89791-667-0. DOI: 10.1145/192161.192283. URL: http://doi.acm.org/10.1145/192161.192283.

[24] Eric P Lafortune in Yves D Willems. "Bi-directional path tracing". V: (1993).

[25] Primož Lavrič. "Spletno ogrodje za vizualizacijo volumetričnih podatkov z možnostjo oddaljenega sodelovanja". V: *repozitorij.uni-lj.si* (2016). URL: https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=slv&id=85027.

[26] Žiga Lesar. "Sledenje poti za neposredno upodabljanje volumetričnih podatkov s spletnimi tehnologijami". V: *repozitorij.uni-lj.si* (2018). URL: https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=slv&id=99766.

[27] William E. Lorensen in Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". V: *SIGGRAPH Comput. Graph.* 21.4 (avg. 1987), str. 163–169. ISSN: 0097-8930. DOI: 10.1145/37402.37422. URL: http://doi.acm.org/10.1145/37402.37422.

[28] R. M. Mersereau in A. V. Oppenheim. "Digital reconstruction of multidimensional signals from their projections". V: *Proceedings of the IEEE* 62.10 (okt. 1974), str. 1319–1338. ISSN: 0018-9219. DOI: 10.1109/PROC.1974.9625.

[29] *Open Source Document Database — MongoDB*. Dosegljivo: https://www.mongodb.com/. [Dostopano: 28. 2. 2019].

[30] John Nickolls in sod. "Scalable Parallel Programming with CUDA". V: *Queue* 6.2 (mar. 2008), str. 40–53. ISSN: 1542-7730. DOI: 10.1145/1365490.1365500. URL: http://doi.acm.org/10.1145/1365490.1365500.

[31] *Node.js*. Dosegljivo: https://nodejs.org/en/. [Dostopano: 25. 2. 2019].

[32] Jan Novák in sod. "Monte Carlo methods for physically based volume rendering". V: *ACM SIGGRAPH 2018 Courses*. ACM. 2018, str. 14.

[33] *npm*. Dosegljivo: `https://www.npmjs.com/`. [Dostopano: 28. 2. 2019].

[34] Yutaka Ohtake in sod. "Multi-level Partition of Unity Implicits". V: ().

[35] *OpenGL - The Industry's Foundation for High Performance Graphics*. Dosegljivo: `https://www.opengl.org/`. [Dostopano: 27. 2. 2019].

[36] Hanspeter Pfister in sod. "The VolumePro Real-time Ray-casting System". V: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH 99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, str. 251–260. ISBN: 0-201-48560-5. DOI: `10.1145/311535.311563`. URL: `http://dx.doi.org/10.1145/311535.311563`.

[37] *PostgreSQL: The World's Most Advanced Open Source Database*. Dosegljivo: `https://www.postgresql.org/`. [Dostopano: 28. 2. 2019].

[38] Anton Purin. *line-navigator*. Dosegljivo: `https://github.com/anpur/line-navigator`. [Dostopano: 21. 2. 2019]. 2015.

[39] Caroline Richmond. "Sir Godfrey Hounsfield". V: *BMJ* 329.7467 (2004), str. 687. ISSN: 0959-8138. DOI: `10.1136/bmj.329.7467.687`. eprint: `https://www.bmj.com/content/329/7467/687.1.full.pdf`. URL: `https://www.bmj.com/content/329/7467/687.1`.

[40] Peter A Rinck. "Magnetic resonance in medicine: the basic textbook of the european magnetic resonance forum". V: *Clinical Radiology-Oxford* 49.11 (1994), str. 842.

[41] Dan Ryan. *History of Computer Graphics: Dlr Associates Series*. Author House, 2011.

[42] *High-Performance Computing Solutions*. Dosegljivo: `www.sgi.com`. [Dostopano: 27. 2. 2019].

[43] *SGI Depot*. Dosegljivo: `http://www.sgidepot.co.uk/sgidepot/`. [Dostopano: 27. 2. 2019].

[44] *SimVascular*. Dosegljivo: http://simvascular.github.io/. [Dostopano: 27. 2. 2019].

[45] *V8 JavaScript engine*. Dosegljivo: https://v8.dev/. [Dostopano: 28. 2. 2019].

[46] H Valladas in sod. "Radiocarbon AMS Dates for Paleolithic Cave Paintings". V: *Radiocarbon* 43.2B (2001), str. 977–986. DOI: 10.1017/S0033822200041643.

[47] Eric Veach in Leonidas J Guibas. "Metropolis light transport". V: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1997, str. 65–76.

[48] *VTK - The Visualization toolkit*. Dosegljivo: https://vtk.org/. [Dostopano: 27. 2. 2019].

[49] *Visualize Your Data With vtk.js*. Dosegljivo: https://kitware.github.io/vtk-js/index.html. [Dostopano: 27. 2. 2019].

[50] Jimmy Wärting. *StreamSaver.js*. Dosegljivo: https://github.com/jimmywarting/StreamSaver.js. [Dostopano: 21. 2. 2019]. 2016.

[51] *Wavefront .obj file*. Dosegljivo: https://en.wikipedia.org/wiki/Wavefront_.obj_file. [Dostopano: 21. 2. 2019].

[52] *OpenGL ES for the Web*. Dosegljivo: https://www.khronos.org/webgl/. [Dostopano: 25. 2. 2019].

[53] *WebGL - Graphics Pipeline*. Dosegljivo: https://www.tutorialspoint.com/webgl/webgl_graphics_pipeline.htm. [Dostopano: 20. 2. 2019].

[54] *The WebSocket API (WebSockets)*. Dosegljivo: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Dostopano: 1. 3. 2019].

[55] Lee A Westover. *SPLATTING: A Parallel, Feed-Forward Volume Rendering Algorithm*. Teh. poročilo. Chapel Hill, NC, USA, 1991.

[56] Wikipedia. *Computer graphics*. Dosegljivo: `https://en.wikipedia.org/wiki/Computer_graphics`. [Dostopano: 26. 2. 2019]. 2019.

[57] Wikipedia. *DirectX*. Dosegljivo: `https://en.wikipedia.org/wiki/DirectX`. [Dostopano: 1. 3. 2019]. 2019.

[58] Wikipedia. *Visualization (graphics)*. Dosegljivo: `https://en.wikipedia.org/wiki/Visualization_(graphics)`. [Dostopano: 21. 2. 2019]. 2013.

[59] Wikipedia. *Volume rendering*. Dosegljivo: `https://en.wikipedia.org/wiki/Volume_rendering`. [Dostopano: 26. 2. 2019]. 2019.

[60] E Woodcock in sod. "Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry". V: *Proc. Conf. Applications of Computing Methods to Reactor Problems*. Zv. 557. 2. 1965.

[61] Dorota Zdrojewska. "Real time rendering of heterogenous fog based on the graphics hardware acceleration". V: (mar. 2019).