Vizualizacija slika računarne tomografije u proširenoj stvarnosti na iOS platformi

Župarić, Adrian

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Računarstvo

VIZUALIZACIJA SLIKA RAČUNALNE TOMOGRAFIJE U PROŠIRENOJ STVARNOSTI NA IOS PLATFORMI

Diplomski rad

Adrian Župarić

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Adrian Župarić
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1345R, 07.10.2022.
JMBAG:	0165079672
Mentor:	prof. dr. sc. Irena Galić
Sumentor:	Marin Benčević, univ. mag. ing. comp.
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	doc. dr. sc. Krešimir Romić
Član Povjerenstva 1:	Marin Benčević, univ. mag. ing. comp.
Član Povjerenstva 2:	doc. dr. sc. Hrvoje Leventić
Naslov diplomskog rada:	Vizualizacija slika računarne tomografije u proširenoj stvarnosti na iOS platformi
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Istražiti i opisati način dobivanja CT slika procesom računalne tomografije i kako se slike spremaju na računalo. Opisati načine vizualizacije CT slika za potrebe dijagnostike i planiranja medicinskih zahvata. Opisati način rada proširene stvarnosti i koncepte računalne grafike relevantne za proširenu stvarnost. Razviti i implementirati rješenje za pretvorbu CT slike u trodimenzionalni model pojedinog tkiva po izboru korisnika. Razviti iOS aplikaciju koja će takav model prikazati u proširenoj stvarnosti uz interakciju korisnika. Opisati razvijene algoritme i ispitati na

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	20.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	04.10.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	10.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 10.10.2024.

Ime i prezime Pristupnika:	Adrian Župarić
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1345R, 07.10.2022.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Vizualizacija slika računarne tomografije u proširenoj stvarnosti na iOS platformi**

izrađen pod vodstvom mentora prof. dr. sc. Irena Galić

i sumentora Marin Benčević, univ. mag. ing. comp.

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

Sadržaj

1. UVOD

U ovom radu istražuje se primjena proširene stvarnosti (AR) u vizualizaciji medicinskih podataka, s naglaskom na računalnu tomografiju (CT). Proširena stvarnost, kao ključna komponenta modernih tehnologija, omogućava nove pristupe interakciji s digitalnim sadržajem u različitim područjima, uključujući medicinu, obrazovanje i inženjering. Razvoj aplikacija koje koriste AR tehnologiju osigurava korisnicima da u stvarnom vremenu pregledavaju složene podatke, što ranije nije bilo moguće. U ovom slučaju, naglasak je na medicinskoj vizualizaciji, gdje se istražuje kako se računalni podaci dobiveni CT skeniranjem mogu učinkovito prikazati unutar AR okruženja na iOS platformi.

Predmet istraživanja u radu je razvoj i implementacija iOS aplikacije koja olakšava vizualizaciju 3D modela dobivenih iz CT snimaka u proširenoj stvarnosti. U tu svrhu, posebna pažnja posvećuje se procesu konverzije DICOM (Digital Imaging and Communications in Medicine) datoteka u RAW format, što omogućava daljnju obradu podataka korištenjem Metal API-ja za renderiranje volumetrijskih podataka. Korištenje proširene stvarnosti pruža intuitivnu interakciju s medicinskim podacima, osiguravajući korisnicima značajan napredak u medicinskom obrazovanju, dijagnostici i planiranju kirurških zahvata. Cilj istraživanja je dizajnirati i implementirati rješenje koje će omogućiti prikaz CT snimaka kao trodimenzionalnih modela u stvarnom okruženju, koristeći proširenu stvarnost na iOS uređajima. U procesu razvoja aplikacije postavljaju se specifični ciljevi kao što su konverzija DICOM datoteka, renderiranje volumetrijskih podataka korištenjem Metal API-ja, integracija ARKit-a za detekciju površina i postavljanje 3D modela unutar AR prostora, te evaluacija performansi aplikacije s obzirom na kvalitetu prikaza i korisničko iskustvo.

Metodologija rada obuhvaća nekoliko ključnih faza. Na početku se provodi istraživanje postojećih tehnologija koje podržavaju vizualizaciju medicinskih podataka u proširenoj stvarnosti, s posebnim naglaskom na DICOM format, ARKit i Metal API, kao i tehnike poput ray marching algoritma za renderiranje volumena. Sljedeći korak uključuje razvoj Python skripte za konverziju DICOM datoteka u RAW format, čime se omogućava daljnja obrada tih podataka u aplikaciji. Implementacija aplikacije izvršava se korištenjem Swift programskog jezika i UIKit biblioteke, integrirajući ARKit i Metal API kako bi se osiguralo renderiranje i prikaz 3D modela CT snimaka unutar AR okruženja. Nakon toga, provode se testiranja

aplikacije na različitim iOS uređajima kako bi se osigurala kvaliteta renderiranja, performanse aplikacije te korisničko iskustvo, a rezultati se evaluiraju kroz analizu performansi aplikacije i povratne informacije korisnika.

Struktura rada osmišljena je kako bi sistematično prikazala sve faze istraživanja. Uvodni dio pruža pregled predmeta i ciljeva rada, metodološkog pristupa te značaja istraživanja. Slijede teorijske osnove, gdje se raspravlja o ključnim tehnologijama i konceptima kao što su računalna tomografija, DICOM format i proširena stvarnost, kao i tehnologije korištene u razvoju aplikacije. U nastavku se detaljno opisuju alati i tehnologije korišteni za razvoj aplikacije, s posebnim naglaskom na iOS razvojne okvire poput Swift-a, Metal API-ja i ARKit-a. Glavni dio rada bavi se razvojem aplikacije, od obrade medicinskih podataka do njihove vizualizacije u AR okruženju, dok završni dio uključuje rezultate, diskusiju i zaključke, gdje se analiziraju postignuti rezultati, ograničenja rada i potencijalni smjerovi za daljnji razvoj.

Značaj ovog istraživanja ogleda se u njegovoj praktičnoj primjeni. Razvoj aplikacije za vizualizaciju medicinskih podataka u proširenoj stvarnosti pruža značajne prednosti u medicinskom obrazovanju, gdje može poslužiti kao edukativni alat za bolje razumijevanje složenih struktura prikazanih na CT snimkama. U medicinskoj praksi, aplikacija omogućava liječnicima da detaljnije analiziraju CT podatke u proširenoj stvarnosti, što može pomoći u preciznijoj dijagnostici i planiranju kirurških zahvata. Također, istraživanje doprinosi tehnološkom razvoju u području računalne grafike i proširene stvarnosti, naglašavajući njezinu praktičnu primjenu u medicini.

2. TEORIJSKA OSNOVA

Ovo poglavlje pruža pregled ključnih tehnoloških koncepata i standarda koji su korišteni u razvoju aplikacije. U nastavku se razmatraju osnovni principi računalne tomografije, struktura i konverzija DICOM formata te osnove proširene stvarnosti (AR) i platforme za razvoj AR aplikacija.

2.1. Računalna tomografija (CT)

Računalna tomografija (CT) je neinvazivna dijagnostička metoda koja koristi X-zrake i računalne algoritme za stvaranje detaljnih poprečnih presjeka tijela. [1] Ova tehnologija je jedan od ključnih alata u modernoj medicinskoj dijagnostici, pružajući precizne informacije koje su neophodne za dijagnosticiranje i liječenje različitih bolesti i stanja.

2.1.1. Princip rada CT skenera

CT skener koristi motorizirani izvor rendgenskog zračenja koji se okreće oko kružnog otvora strukture u obliku krafne koja se naziva portal [2]. CT skeneri koriste rendgenske zrake kako bi kreirali seriju križnih presjeka tijela, koje se kasnije računalno rekonstruiraju u trodimenzionalnu sliku. U osnovi, rendgenska cijev se rotira oko pacijenta, dok detektori smješteni nasuprot cijevi bilježe intenzitet rendgenskih zraka nakon što su prošle kroz tijelo. Omogućava dobivanje slike s visokom razinom detalja, što je neophodno za preciznu dijagnostiku različitih medicinskih stanja. Svaka kriška tijela snimljena CT-om pruža dvodimenzionalni presjek, koji se zatim računalno kombinira u trodimenzionalnu sliku, pružajući liječnicima detaljan prikaz unutarnjih struktura pacijenta.

Princip rada CT skenera temelji se na činjenici da različita tkiva u tijelu apsorbiraju rendgenske zrake u različitoj mjeri, što rezultira varijacijama u intenzitetu signala zabilježenog detektorima. Te varijacije koriste se za stvaranje slike, pri čemu se guste strukture, poput kostiju, prikazuju svijetlim tonovima, dok se meka tkiva prikazuju tamnijim tonovima. Ova tehnologija ne samo da pruža prikaz anatomskih struktura, već i procjenu patoloških promjena, kao što su tumori, krvarenja ili frakture, što je od ključnog značaja u medicinskoj dijagnostici.

2.1.2. Upotreba CT-a u medicini

CT skeneri koriste se u različitim medicinskim područjima, uključujući neurologiju [3], kardiologiju [4], onkologiju [5] i traumatologiju [6]. Pruža neinvazivno vizualiziranje kostiju, krvnih žila i mekih tkiva, te se često koristi za brzo postavljanje dijagnoze u hitnim medicinskim slučajevima. U neurologiji, na primjer, CT se često koristi za detekciju moždanih udara, tumora ili trauma glave, gdje brza i točna dijagnostika može biti presudna za uspješan ishod liječenja. Slično, u onkologiji se CT koristi za detekciju i praćenje tumora, pružajući liječnicima da procijene veličinu, oblik i lokaciju tumora te planiraju odgovarajući tretman.

Osim toga, CT se koristi i u planiranju kirurških zahvata [7], gdje precizne slike pomažu kirurzima da detaljno prouče anatomske strukture pacijenta prije operacije. Ovaj alat pomaže i u praćenju pacijenata nakon zahvata, kako bi se osiguralo da je oporavak u skladu s očekivanjima. Zbog svoje preciznosti i brzine, CT je postao standardna metoda u mnogim medicinskim ustanovama širom svijeta.

2.2. DICOM format

DICOM (Digital Imaging and Communications in Medicine) je standard za pohranu, prijenos i upravljanje medicinskim slikama i pripadajućim podacima. Razvijen je s ciljem postizanja interoperabilnosti između različitih medicinskih uređaja i informacijskih sustava unutar zdravstvenih ustanova [8]. DICOM je postao ključni standard u medicinskoj informatici, osiguravajući da slike i podaci mogu biti dijeljeni i analizirani bez obzira na korištenu opremu ili softver.

2.2.1. Struktura DICOM datoteka

DICOM datoteke sadrže slikovne podatke i metapodatke, koji uključuju informacije o pacijentu, postavkama skenera, vremenu snimanja i drugim relevantnim detaljima. Struktura DICOM datoteke pohranjuje složene medicinske informacije u jedan format, čime se osigurava njihova dostupnost i ispravnost prilikom prijenosa između različitih sustava. Osnovna struktura DICOM datoteke sastoji se od niza elemenata koji uključuju jedinstveni identifikator studije, serije i slike, kao i niz specifičnih tagova koji opisuju tehničke karakteristike snimanja.

Svaka DICOM datoteka uključuje zaglavlje koje sadrži osnovne informacije o slici, poput dimenzija, rezolucije i dubine boje. Uz to, metapodaci u DICOM datotekama pružaju integraciju s bolničkim informacijskim sustavima (HIS) i sustavima za pohranu i distribuciju slika (PACS) [9], što osigurava brzi pristup i analizu slika od strane različitih stručnjaka unutar zdravstvene ustanove.

2.2.2. Konverzija DICOM u druge formate (RAW)

Iako DICOM format omogućuje pohranu detaljnih informacija, ponekad je potrebno konvertirati ove datoteke u jednostavniji format, kao što je RAW, za specifične primjene kao što su obrada i vizualizacija u realnom vremenu. Konverzija DICOM datoteka u RAW format osigurava pristup samim slikovnim podacima bez dodatnih informacija, što pojednostavljuje obradu i pruža bržu i učinkovitiju vizualizaciju unutar AR okruženja. Proces konverzije zahtijeva ekstrakciju relevantnih podataka iz DICOM datoteke, pri čemu se uklanjaju metapodaci koji nisu nužni za prikaz slike, a zadržavaju se bitmape koje predstavljaju samu sliku.

RAW format pruža izravnu manipulaciju pikselima slike, što je korisno u aplikacijama gdje je potrebna visoka razina kontrole nad prikazom i obradom slike, kao što je to slučaj u proširenoj stvarnosti. Ovaj proces konverzije često se provodi pomoću specijaliziranih softverskih alata ili skripti koje automatiziraju obradu velikog broja datoteka, čime se značajno smanjuje vrijeme potrebno za pripremu podataka za vizualizaciju.

2.3. Proširena stvarnost (AR)

Proširena stvarnost (AR) je poboljšana verzija stvarnog svijeta, postignuta korištenjem računalno generiranih digitalnih informacija [10]. AR je postala ključna tehnologija u mnogim industrijama, pružajući korisnicima mogućnost interakcije s digitalnim objektima koji su vizualno integrirani u njihov stvarni okoliš. Ova tehnologija ne samo da obogaćuje stvarni svijet digitalnim sadržajem, već i stvara potpuno nova iskustva koja spajaju stvarne i virtualne elemente.

2.3.1. Definicija i primjena AR tehnologije

AR tehnologija obogaćuje stvarni svijet digitalnim sadržajem, poput slika, zvukova i drugih informacija. Primjena AR-a u različitim industrijama uključuje edukaciju, igru, inženjering, marketing, te u posljednje vrijeme, medicinu. U medicini, AR se koristi za pružanje dodatnih informacija liječnicima tijekom operacija, za edukaciju studenata, te intuitivnu vizualizaciju složenih medicinskih podataka. [11]

Primjeri primjene AR-a u medicini uključuju intraoperativno navođenje, gdje AR pomaže kirurzima pružanjem slojevitih informacija direktno na pacijentovu anatomiju, te u edukaciji, gdje studenti medicine mogu koristiti AR za interaktivno učenje anatomije i drugih medicinskih znanosti. Korištenje AR-a u medicini doprinosi preciznijoj dijagnostici, smanjuje rizik od komplikacija tijekom kirurških zahvata i poboljšava opće razumijevanje složenih medicinskih pojmova.

2.3.2. Platforme za razvoj AR aplikacija

Za razvoj AR aplikacija na mobilnim uređajima koriste se različite platforme, od kojih je ARKit za iOS uređaje jedna od najpopularnijih. ARKit detektira ravne površine, prati pokrete i pozicionira digitalne objekte unutar stvarnog prostora. Ova platforma olakšava razvoj AR aplikacija uz visoku razinu preciznosti i stabilnosti, što je ključno za medicinske primjene gdje su točnost i pouzdanost kritični.

ARKit također podržava razvoj naprednih AR funkcionalnosti, kao što su prepoznavanje lica, stvaranje složenih 3D modela u stvarnom vremenu, te integracija s ostalim Appleovim tehnologijama, poput SceneKit i Metal. Ove mogućnosti čine ARKit izuzetno moćnim alatom za razvoj aplikacija koje zahtijevaju visok stupanj interaktivnosti i realizma. Korištenjem ARKit-a, razvijatelji mogu stvoriti aplikacije koje pružaju korisnicima jedinstveno iskustvo, integrirajući digitalne objekte na način koji je prirodan i intuitivan.

3. ALATI I TEHNOLOGIJE

Razvoj složenih aplikacija za proširenu stvarnost (AR) koje koriste računalnu tomografiju (CT) zahtijeva upotrebu naprednih alata i tehnologija. Ovaj dio rada opisuje ključne tehnologije korištene u razvoju iOS aplikacije, uključujući Swift, Metal, i ARKit. Ove tehnologije omogućuju visoku razinu performansi, detaljnu grafiku i interaktivnost, što je neophodno za medicinske aplikacije koje se oslanjaju na vizualizaciju složenih podataka u AR okruženju.

3.1. Swift i iOS razvojno okruženje

Swift je programski jezik razvijen od strane Applea, a koristi se za razvoj aplikacija na iOS, macOS, watchOS i tvOS platformama. Njegova jednostavnost, brzina i sigurnost učinili su ga jednim od najpopularnijih jezika za razvoj aplikacija na Appleovim platformama.

3.1.1. Uvod u Swift programski jezik

Swift je predstavljen 2014. godine kao nasljednik Objective-C-a, jezika koji je godinama bio osnova za razvoj aplikacija na Apple platformama. Razvijen s naglaskom na sigurnost, Swift uključuje moderne značajke poput automatskog upravljanja memorijom i zaštite od grešaka, što ga čini idealnim za razvoj aplikacija visoke sigurnosti i performansi [12]. Swift koristi sintaksu koja je jednostavna za učenje, ali dovoljno moćna za rješavanje kompleksnih problema, što ga čini prilagodljivim za različite razine programera. Osim toga, Swift je otvorenog koda, što omogućava široku upotrebu i razvoj unutar globalne zajednice. [13]

Jedna od ključnih prednosti Swifta je njegova brzina, koja osigurava učinkovitu obradu podataka u stvarnom vremenu. Za aplikacije koje koriste proširenu stvarnost, poput vizualizacije medicinskih podataka, ova brzina je ključna za osiguravanje fluidnog korisničkog iskustva. Swift također podržava jednostavnu integraciju s postojećim Objective-C kodom, što ga čini vrlo fleksibilnim za projekte koji zahtijevaju kombinaciju različitih tehnologija.

3.1.2. UIKit framework

UIKit je osnovni framework za razvoj korisničkog sučelja na iOS platformi. Pomoću UIKit-a programeri mogu kreirati interaktivna sučelja, upravljati događajima korisnika te animirati elemente sučelja. Ovaj framework pruža gotove komponente, kao što su gumbi, tablice,

navigacijske trake i mnoge druge, koje olakšavaju razvoj vizualno privlačnih i funkcionalnih aplikacija.

Jedna od prednosti UIKit-a je njegova fleksibilnost i mogućnost prilagodbe. Programerima pruža stvaranje jedinstvenih i dinamičnih korisničkih sučelja pomoću storyboarda ili putem programskog koda. U slučaju aplikacija koje koriste proširenu stvarnost, UIKit nudi jednostavan način za integraciju s ARKit-om, omogućujući kreiranje interaktivnih 3D modela i njihovo pozicioniranje unutar stvarnog svijeta.

3.2. Ray Marching

Ray marching je tehnika renderiranja koja se koristi za prikaz složenih trodimenzionalnih scena, posebno volumetrijskih podataka i implicitnih površina. Ova tehnika se temelji na uzorkovanju duž puta zrake (ray) koja se emitira iz pozicije kamere i prolazi kroz scenu. *Ray marching* se najčešće koristi u kontekstu renderiranja volumena, površina definiranih implicitnim funkcijama te vizualizacije složenih polja podataka poput magle, oblaka, tekućina i sličnih fenomena.

U *ray marching* algoritmu, zraka se "maršira" kroz scenu u malim koracima. Na svakom koraku, evaluira se određena vrijednost, najčešće udaljenost do najbliže površine ili gustoća volumena, te se na temelju te informacije odlučuje hoće li zraka nastaviti kretanje ili je naišla na objekt unutar scene. Koraci se ponavljaju dok zraka ne naiđe na površinu ili dok ne izađe iz granica volumena.

Matematički, pozicija p(t) zrake u svakom trenutku t definirana je kao

$$p(t) = o + t * d$$

gdje je o pozicija kamere, d normalizirani vektor smjera zrake, a t udaljenost od kamere do trenutne točke uzorkovanja na zraci.

Ray marching algoritam, koji je prikazan na slici 3.1, započinje emitiranjem zrake iz kamere kroz svaki piksel ekrana, pri čemu se za svaki piksel određuje početna točka zrake o i njezin smjer d. Zraka se zatim kreće kroz volumen, pri čemu algoritam na svakom koraku provjerava

određeni kriterij, poput udaljenosti do najbliže površine, i na temelju te informacije odlučuje hoće li zraka nastaviti kretanje ili će se zaustaviti. U scenama koje sadrže implicitne površine, koristi se funkcija udaljenosti s predznakom (engl. *signed distance function*, SDF), koja za zadanu točku p vraća udaljenost do najbliže površine. Ako je ta udaljenost manja od određenog praga, smatra se da je zraka pogodila površinu. Nakon što zraka naiđe na površinu, izračunava se boja na temelju svojstava materijala i svjetla u sceni, kao i efekata poput sjena ili refleksije. Konačna boja se akumulira uzduž cijelog puta zrake, što omogućava prikaz složenih efekata, uključujući volumetrijsko raspršenje svjetlosti.

```
for svaki piksel na ekranu:
    t = 0
    pozicija = početna_pozicija_zrake
    for i = 0 do max_iteracija:
        udaljenost = evaluiraj_sdf(pozicija)
        if udaljenost < prag:
            break
        pozicija += smjer_zrake * udaljenost
        t += udaljenost
    ako je pogodak:
        izračunaj_boju()
    inače:
        postavi pozadinsku_boju()
```

Slika 3.1 Pseudokod za Ray Marching

Ray marching je izrazito fleksibilna tehnika koja se koristi u situacijama gdje klasične metode rasterizacije ne mogu učinkovito prikazati scene s volumetrijskim efektima, implicitnim površinama ili dinamičkim fenomenima poput oblaka, dima, i magle. Usporedbom s ray tracingom, *ray marching* je efikasniji kada je riječ o renderiranju složenih scena koje nemaju jasno definirane geometrijske granice, ali zahtijeva više iteracija za složenije objekte.

3.3. Metal

Metal je Appleov niskorazinsko grafičko programsko sučelje (API) koje pruža izravan pristup GPU-u (grafičkom procesoru), čime se značajno povećava brzina i učinkovitost obrade složenih grafičkih operacija. Metal API se koristi za stvaranje visokokvalitetnih vizualnih prikaza u stvarnom vremenu, što ga čini ključnim alatom za aplikacije koje zahtijevaju visoku razinu grafičke preciznosti, poput igara, simulacija i medicinskih aplikacija koje uključuju vizualizaciju volumetrijskih podataka. Za razliku od starijih API-ja poput OpenGL-a, Metal ostvaruje mnogo bolje upravljanje resursima GPU-a i optimizaciju performansi, što je ključno za složene zadatke poput renderiranja volumetrijskih podataka iz računalne tomografije (CT).

3.3.1. Uvod u Metal

Metal je prvi put predstavljen 2014. godine kao dio iOS-a 8, a ubrzo je postao standard za obradu grafike na Apple platformama, uključujući iOS, macOS, tvOS i watchOS. Cilj ovog API-ja je omogućiti programerima što bolju kontrolu nad GPU-om, eliminirajući potrebu za posrednicima poput OpenGL-a i Vulkan API-ja, koji su i dalje široko korišteni na drugim platformama.

Metal podržava izvođenje visoko optimiziranih grafičkih i računskih zadataka uz minimalnu latenciju i manju potrošnju resursa procesora. To se postiže izravnim upravljanjem GPU memorijom i paralelizacijom zadataka na tisućama jezgri GPU-a, što aplikacijama omogućuje brzu obradu velikih količina podataka. Za aplikacije koje koriste proširenu stvarnost (AR) i volumetrijsku vizualizaciju medicinskih podataka, poput onih u ovom radu, Metal osigurava prikaz složenih volumena i 3D modela u stvarnom vremenu, uz visoku razinu detalja i preciznosti.

3.3.2. Upotreba Metal API-ja za obradu grafike

Metal API pruža visok stupanj kontrole nad GPU-om, što je ključna prednost u aplikacijama koje zahtijevaju renderiranje složenih scena i volumetrijskih podataka. U kontekstu vizualizacije medicinskih podataka, Metal se koristi za prikaz složenih volumetrijskih skenova, poput onih iz CT-a. Ovaj API omogućuje manipulaciju podacima direktno na GPU-u, što značajno ubrzava procese kao što su filtriranje, segmentacija i renderiranje volumetrijskih podataka.

Renderiranje volumetrijskih podataka zahtijeva posebne algoritme, a jedan od najpoznatijih je *ray marching*. Metal omogućuje izradu prilagođenih *shader* programa koji se izvode direktno na GPU-u, čime se postiže renderiranje složenih 3D volumena u stvarnom vremenu. Metal API nudi skup alata za implementaciju takvih tehnika, uključujući podršku za volumetrijsko osvjetljenje, sjene i simulaciju raspršenja svjetlosti kroz medij. U volumetrijskom prikazu, posebno kada se koristi u medicinske svrhe, važno je osigurati da podaci zadrže visok stupanj točnosti i detalja, što Metal postiže kroz svoju nativnu podršku za visokoprecizne operacije s pomičnim zarezom i manipulaciju teksturama.

3.4. ARKit

ARKit je Appleov framework za proširenu stvarnost prvi put predstavljen 2017. godine s iOS-om 11. Omogućuje iOS aplikacijama da prepoznaju fizičko okruženje, analiziraju ga i u njega integriraju virtualne objekte koji su vizualno uvjerljivi i interaktivni. ARKit koristi senzore uređaja, poput kamere, akcelerometra i žiroskopa, kako bi prikupljao informacije o stvarnom svijetu te precizno pozicionirao digitalne objekte unutar tog okruženja. Ovo je temeljna tehnologija podržava razvoj naprednih AR aplikacija koje mogu reagirati na promjene u stvarnom svijetu u stvarnom vremenu.

ARKit koristi napredne algoritme računalnog vida kako bi detektirao ravne površine, prepoznao lica i omogućio praćenje pokreta, a sve to bez potrebe za dodatnim hardverskim uređajima. Ova funkcionalnost pruža mogućnost razvoja AR aplikacija koje su dostupne širokoj bazi korisnika, jer je potrebna samo kamera iOS uređaja.

3.4.1. Uvod u ARKit

Jedan od ključnih elemenata ARKit-a je detekcija i praćenje ravnih površina, kao što su podovi, stolovi ili zidovi. Ovaj proces koristi vizualne podatke kamere zajedno s podacima iz senzora za procjenu gdje se nalaze te površine u fizičkom prostoru. To omogućuje aplikacijama da precizno pozicioniraju 3D objekte na te površine, stvarajući iluziju da su objekti dio stvarnog svijeta.

Još jedna ključna komponenta ARKit-a je *World Tracking*, koji aplikacijama omogućuje kontinuiran praćenje pokreta uređaja u prostoru. *World Tracking* kombinira podatke iz kamere i senzora kako bi se izračunala precizna pozicija i orijentacija uređaja u stvarnom vremenu. Time digitalni objekti ostaju stabilno postavljeni unutar fizičkog prostora, čak i kada se korisnik kreće oko njih. Ova mogućnost je ključna za stvaranje uvjerljivih AR iskustava jer korisnicima omogućuje da se slobodno kreću oko 3D objekata i promatraju ih iz različitih kutova, baš kao što bi to činili s fizičkim predmetima.

U kasnijim verzijama ARKit-a, dodane su značajke kao što su prepoznavanje lica, detekcija pokreta ruku, te poboljšane mogućnosti interakcije s digitalnim objektima. Prepoznavanje lica pomaže aplikacijama da identificiraju i prate izraze lica korisnika u stvarnom vremenu, dok

detekcija pokreta ruku prepoznaje interaktivne geste koje aplikacije koriste za interakciju s digitalnim sadržajem.

ARKit također koristi razumijevanje scena, funkcionalnost koja pomaže u shvaćanju trodimenzionalne geometrije prostora. Ovo uključuje prepoznavanje i analizu struktura u fizičkom prostoru, poput zidova, vrata i prozora, kako bi interakcija digitalnih objekata s fizičkim svijetom bila što točnija.

3.4.2. ARSCNView i SceneKit

ARSCNView je klasa unutar ARKit-a koja služi za prikazivanje 3D sadržaja unutar proširenog okruženja koristeći SceneKit, Appleov framework za manipulaciju i renderiranje trodimenzionalnih objekata. ARSCNView pruža jednostavno rješenje za integraciju AR funkcionalnosti u aplikacije, omogućujući programerima renderiranje 3D modela i njihovih interakcija sa stvarnim svijetom bez potrebe za implementacijom vlastitog renderera.

SceneKit je alat za stvaranje i manipulaciju 3D objektima. Podržava animacije, fizikalne simulacije, kao i napredne grafičke efekte kao što su osvjetljenje i sjene. U kombinaciji s ARSCNView-om, SceneKit omogućava postavljanje i prikazivanje 3D modela unutar stvarnog svijeta, s mogućnošću interakcije korisnika s tim modelima. Ova kombinacija je posebno korisna za aplikacije koje zahtijevaju detaljnu vizualizaciju složenih podataka.

ARSCNView koristi sve prednosti SceneKit-a, uključujući napredno osvjetljenje, teksture i podršku za fizikalne simulacije. Time aplikacije mogu prikazati visokokvalitetne 3D modele koji realistično reagiraju na promjene u osvjetljenju i interakcije s korisnikom. Na primjer, u medicinskoj aplikaciji koja koristi AR, 3D model CT snimke može biti prikazan unutar stvarnog prostora, a korisnik može rotirati, skalirati i ispitivati model iz različitih kutova kako bi detaljno pregledao unutarnje strukture.

3.4.3. ARKit i integracija s Metal API-jem

Jedna od prednosti ARKit-a jest njegova mogućnost integracije s Metal API-jem, što pruža napredne grafičke efekte i ubrzanje renderiranja. Korištenjem Metal API-ja zajedno s ARKit-om, moguće je prikazati vrlo složene i detaljne 3D scene koje zahtijevaju brzu i optimiziranu

obradu podataka. Ova kombinacija ARKit-a i Metal-a omogućuje stvaranje aplikacija koje mogu brzo i učinkovito renderirati vrlo detaljne 3D modele, dok istovremeno održavaju visoku razinu interakcije s korisnicima. U slučaju volumetrijskih podataka iz CT skenova, Metal ubrzava proces *Ray Marching* algoritma i renderirati složene volumene, dok ARKit osigurava stabilno pozicioniranje modela u stvarnom svijetu.

4. RAZVOJ APLIKACIJE

Aplikacija je osmišljena s ciljem prikaza složenih medicinskih podataka na intuitivan i interaktivan način, koristeći tehnologije poput ARKit-a i Metal API-ja, koje pružaju naprednu vizualizaciju i manipulaciju trodimenzionalnih volumena. Razvoj aplikacije obuhvaća nekoliko ključnih faza: konverzija DICOM datoteka u RAW format, izrada trodimenzionalnog modela, implementacija AR funkcionalnosti te optimizacija performansi za rad u stvarnom vremenu na mobilnim uređajima. Ove faze povezane su s odgovarajućim alatima i tehnologijama, kao što su Python za pripremu podataka, Swift za razvoj aplikacije te Metal API za renderiranje volumena.

4.1. Obrada DICOM podataka

Iako je DICOM izuzetno koristan za medicinske slike, on nije nativno podržan u Swift programskom jeziku ili unutar iOS razvojnih okvira, poput UIKit ili SwiftUI. Zbog toga se programeri koji rade s DICOM datotekama na iOS-u moraju osloniti na vanjske biblioteke ili prethodno obraditi podatke u formate koje iOS može lako interpretirati, poput RAW formata ili PNG-a.

U ovom radu, korištena je Python skripta za konverziju DICOM datoteka u RAW format, koji je prikladan za vizualizaciju unutar aplikacije. Proces započinje čitanjem DICOM datoteka iz direktorija i njihovom dekompresijom, ukoliko su komprimirane. Pomoću biblioteke pydicom, pikselni podaci izdvajaju se iz svake DICOM datoteke i pohranjuju u niz koji predstavlja volumetrijski prikaz. Nakon toga, volumetrijski podaci se spremaju u RAW format, čime se osigurava daljnja obrada i vizualizacija u stvarnom vremenu.

Konverzija DICOM podataka u RAW format posebno je važna jer olakšava i ubrzava rukovanje velikim količinama podataka. U volumetrijskim vizualizacijama, kao što su CT skenovi, ključna je brzina obrade i prikaza podataka kako bi se osiguralo da korisnici mogu nesmetano manipulirati modelima unutar proširene stvarnosti. Nakon konverzije u RAW, aplikacija koristi napredne grafičke API-je poput Metal i ARKit za prikaz volumetrijskih podataka, omogućujući korisnicima da interaktivno pregledaju složene strukture unutar AR okruženja.

4.1.1. Konverziju DICOM u RAW

Za vizualizaciju CT podataka unutar proširene stvarnosti, potrebno je konvertirati DICOM datoteke u RAW format koji omogućava brzu manipulaciju podacima i prikaz u stvarnom vremenu. U tu svrhu razvijena je Python skripta koja koristi biblioteku **pydicom** za čitanje DICOM datoteka i **numpy** za upravljanje slikovnim podacima. Konverzija u RAW format pojednostavljuje rukovanje volumetrijskim podacima prilikom renderiranja u aplikacijama.

Prvi korak skripte sastoji se od čitanja DICOM datoteka iz zadanog direktorija. U tu svrhu koristi se funkcija os.path.join() za formiranje putanje do datoteka, a funkcija pydicom.dcmread() za čitanje podataka iz svake DICOM datoteke. Ako datoteka sadrži slikovne podatke, oni se obrađuju, a ako je komprimirana, dekomprimira se pomoću metode decompress(). Slika 4.1 prikazuje dio koda odgovoran za čitanje DICOM datoteka.

```
def convert_dcm_folder_to_raw(folder_path, raw_filename):
    slices = []

    files = [f"image-{i:06d}.dcm" for i in range(1, 361)]

    for filename in files:
        filepath = os.path.join(folder_path, filename)
        print(filename)
        ds = pydicom.dcmread(filepath)

        if 'PixelData' in ds:
            if ds.file_meta.TransferSyntaxUID.is_compressed:
                ds.decompress()
```

Slika 4.1 Čitanje DICOM datoteka

Nakon što su DICOM datoteke učitane, pikselni podaci se izdvajaju i pohranjuju u listu. Svaka slika se konvertira u 16-bitni integer format, čime se osigurava preciznija pohrana medicinskih podataka. Prikupljeni pikseli pohranjuju se u listu slices za kasniju obradu i stvaranje volumetrijskog prikaza. Slika 4.2 prikazuje ovaj korak.

```
pixel_array = ds.pixel_array
slices.append(pixel_array.astype(np.int16))
```

Slika 4.2 Pohrana piksela u 16-bitnom formatu

Nakon što su svi slojevi prikupljeni, koristi se funkcija numpy.stack() kako bi se slojevi posložili u trodimenzionalni volumen. Ovaj volumen predstavlja cjeloviti CT prikaz, gdje su svi slojevi poredani jedan iznad drugog. Konačni volumen se zatim pohranjuje u RAW format. Slika 4.3 prikazuje kod za stvaranje volumetrijskog prikaza.

```
volume = np.stack(slices, axis=0)
volume.tofile(raw_filename)
```

Slika 4.3 Stvaranje volumena iz niza slojeva

RAW format je odabran zbog svoje jednostavnosti i brzine obrade. Pohrana podataka u RAW format osigurava daljnju obradu i vizualizaciju unutar aplikacija u stvarnom vremenu, bez potrebe za dekompresijom ili konverzijom tijekom izvođenja aplikacije. Ovo značajno ubrzava prikaz i manipulaciju podacima, što je ključno za aplikacije u proširenoj stvarnosti.

Slika 4.3 prikazuje Python skriptu koja pruža efikasno rješenje za pretvorbu DICOM datoteka u RAW format, omogućavajući brzo stvaranje volumetrijskih prikaza medicinskih podataka.

```python
import pydicom
import numpy as np
import os

def convert_dcm_folder_to_raw(folder_path, raw_filename):
    slices = []

    files = [f"image-{i:06d}.dcm" for i in range(1, 361)]

    pixel_spacing = None
    slice_thickness = None
    slice_spacing = None
    slice_number = 0

    np.set_printoptions(threshold=np.inf)

    for filename in files:
        filepath = os.path.join(folder_path, filename)

        ds = pydicom.dcmread(filepath)

        if 'PixelData' in ds:
            if ds.file_meta.TransferSyntaxUID.is_compressed:
                ds.decompress()

        pixel_array = ds.pixel_array

        if 'RescaleSlope' in ds and 'RescaleIntercept' in ds:
            rescale_slope = ds.RescaleSlope
            rescale_intercept = ds.RescaleIntercept
            pixel_array = pixel_array * rescale_slope + rescale_intercept

        if pixel_array.dtype != np.int16:
            pixel_array = pixel_array.astype(np.int16)

        slices.append(pixel_array)
        slice_number += 1
        if pixel_spacing is None:
            pixel_spacing = ds.PixelSpacing
        if slice_thickness is None:
            slice_thickness = ds.SliceThickness
        if slice_spacing is None:
            slice_spacing = ds.SpacingBetweenSlices

    volume = np.stack(slices, axis=0)

    print(f"Volume dimensions (rows, columns, depth): {volume.shape}")

    volume.tofile(raw_filename)

    print(f"PixelSpacing: {pixel_spacing}")
    print(f"SliceThickness: {slice_thickness}")
    print(f"SpacingBetweenSlices: {slice_spacing}")
    print(f"NumberOfSlices: {slice_number}")

folder_path = './dcm_folder'
raw_filename = 'output.raw'

convert_dcm_folder_to_raw(folder_path, raw_filename)
```

Slika 4.3 Python skripta za pretvaranje DICOM u RAW format

4.1.2. Integracija konvertiranih podataka u aplikaciju

Nakon što se DICOM datoteke konvertiraju u RAW format pomoću Python skripte, sljedeći korak je njihova integracija u iOS aplikaciju, gdje će se ti podaci dalje koristiti za prikaz u proširenoj stvarnosti. Integracija ovih podataka ključna je za pravilnu pripremu volumena koji će se prikazati korisniku. Važan aspekt ovog dijela razvoja aplikacije je organizacija i manipulacija sirovim podacima kako bi oni bili prikladni za vizualizaciju koristeći GPU i Metal API.

Osnovni način pohrane volumetrijskih podataka unutar aplikacije obuhvaća korištenje strukture VolumeData. Ova struktura dizajnirana je tako da pohranjuje sirove podatke (RAW), rezoluciju volumena u pikselskim jedinicama, kao i dimenzije volumetrijskog prikaza. Na ovaj način osigurava se pravilno rukovanje velikim količinama podataka, uz optimizaciju za efikasnu manipulaciju na strani GPU-a.

Struktura VolumeData, prikazana na slici 4.4, sadrži tri ključna elementa:

- data: Sirovi podaci u binarnom formatu, koji predstavljaju skenirani volumen
- resoultion: Rezolucija volumena izražena u pikselima po jedinici duljine duž osi X, Y i Z
- dimension: dimenzije volumena, koje predstavljaju broj slojeva duž svake osi. Ove dimenzije određuju ukupnu veličinu volumena i pomažu pri renderiranju svakog sloja pojedinačno

```
struct VolumeData {
    let data: Data
    let resolution: SIMD3<Float>
    var dimension: SIMD3<Int>
}
```

Slika 4.4 Struktura VolumeData

Ova struktura ne samo da omogućava aplikaciji pohranu podataka, već i pristup ključnim informacijama poput veličine volumena i razmaka između slojeva. Na primjer, rezolucija je

definirana kao SIMD3, što osigurava vrlo precizno pohranjivanje gustoće piksela duž svake od tri osi. To je posebno važno kada se vizualizira medicinski volumen, gdje je od ključne važnosti točnost prikaza različitih tkiva i struktura. Dimenzije volumena, definirane kao SIMD3, pružaju informacije o broju slojeva u volumenu, što omogućuje pravilnu interpretaciju podataka tijekom renderiranja.

Nakon što su volumetrijski podaci konvertirani u RAW format, oni se učitavaju u aplikaciju prilikom njenog pokretanja. Ovaj korak omogućuje aplikaciji da preuzme podatke iz prethodno konvertirane arhive (ZIP), raspakira ih i pripremi za vizualizaciju. Funkcija showNextScreen unutar StartViewController klase odgovorna je za učitavanje sirovih podataka, njihovu organizaciju u strukturu VolumeData te prijenos tih podataka u sljedeći ekran aplikacije gdje će biti vizualizirani.

U ovom slučaju, aplikacija učitava datoteku raw.zip, koja sadrži konvertirane volumetrijske podatke. Ovi podaci se raspakiravaju pomoću ZIPFoundation biblioteke, a zatim se prebacuju u objekt VolumeData kako bi bili spremni za daljnje korake obrade. Na slici 4.5 je prikazan relevantan dio koda za učitavanje i raspakiravanje podataka.

```
func continueWithPredefinedButtonTapped() {
    guard let url = Bundle.main.url(forResource: "output", withExtension: "raw.zip") else {
        print("Guard failed: StartViewController – url")
        return
    }

    do {
        let archive = try Archive(url: url, accessMode: .read)
        var data = Data()
        for entry in archive {
            do {
                _ = try archive.extract(entry) { extractedData in
                    data.append(extractedData)
                }
            } catch {
                print("Extraction failed: \(error.localizedDescription)")
            }
        }
        showNextScreen(with: data)
    } catch {
        print("Unzip failed: \(error.localizedDescription)")
    }
}
```

Slika 4.5 Raspakiravanje ZIP arhive i učitavanje podataka

Funkcija continueWithPredefinedButtonTapped učitava ZIP datoteku i koristi Archive objekt za raspakiranje datoteke. Svi podaci iz arhive dodaju se u varijablu data, koja se kasnije prosljeđuje u funkciju showNextScreen radi daljnje obrade.
Jednom kada se sirovi podaci uspješno raspakiraju i učitaju, oni se integriraju u strukturu VolumeData. Ova struktura pohranjuje sve informacije potrebne za vizualizaciju, uključujući sirove podatke, rezoluciju i dimenzije volumena. Funkcija showNextScreen, kao što je prikazano na slici 4.6, inicijalizira objekt VolumeData i prenosi podatke u sljedeći prikaz gdje će biti generirani 3D modeli.

```
func showNextScreen(with data: Data) {
    let volumeData = VolumeData(
        data: data,
        resolution: SIMD3<Float>(0.58984375, 0.58984375, 1),
        dimension: SIMD3<Int>(512, 512, 360)
    )
    let vc = ViewerViewController(data: volumeData)
    navigationController?.pushViewController(vc, animated: true)
}
```

Slika 4.6 Inicijalizacija strukture VolumeData i prikazivanje sljedećeg ekrana

U ovom koraku, funkcija showNextScreen stvara instancu objekta VolumeData, koristeći sirove podatke, zadanu rezoluciju od 0.58984375 mm po pikselskom koraku duž X i Y osi, te dimenzije volumena od 512 x 512 x 360. Ovi podaci zatim se prenose u ViewerViewController, gdje će biti korišteni za generiranje 3D prikaza volumena. Funkcija također pruža prijelaz na sljedeći ekran aplikacije, gdje će se korisniku prikazati volumetrijski prikaz.

Jednom kada su podaci integrirani u strukturu VolumeData, aplikacija koristi funkciju generateTexture kako bi stvorila trodimenzionalne teksture koje će GPU koristiti za renderiranje. Funkcija generateTexture, kao što je prikazano na slici 4.7, koristi Metal API kako bi generirala 3D teksture iz RAW podataka, omogućavajući aplikaciji da brzo i efikasno obrađuje velike količine podataka.

```swift
func generateTexture(with device: MTLDevice) -> MTLTexture? {
    let textureDescriptor = MTLTextureDescriptor()
    textureDescriptor.usage = .shaderRead
    textureDescriptor.textureType = .type3D
    textureDescriptor.pixelFormat = .r16Sint
    textureDescriptor.width = volumeData.dimension.x
    textureDescriptor.height = volumeData.dimension.y
    textureDescriptor.depth = volumeData.dimension.z

    let bytesPerRow = MemoryLayout<Int16>.size * textureDescriptor.width
    let bytesPerImage = bytesPerRow * textureDescriptor.height

    let texture = device.makeTexture(descriptor: textureDescriptor)

    texture?.replace(
        region: MTLRegionMake3D(0, 0, 0, textureDescriptor.width, textureDescriptor.height, textureDescriptor.depth),
        mipmapLevel: 0,
        slice: 0,
        withBytes: (volumeData.data as NSData).bytes,
        bytesPerRow: bytesPerRow,
        bytesPerImage: bytesPerImage
    )

    return texture
}
```

Slika 4.7 Generiranje texture za Metal

Funkcija generateTexture koristi MTLTextureDescriptor kako bi definirala ključne parametre teksture, uključujući njezinu upotrebu (*shader read*), format piksela (16-bitni integeri) i dimenzije volumena. Nakon definiranja teksture, podaci se kopiraju u memorijski prostor na GPU-u koristeći metodu replace, čime su teksture spremne za renderiranje unutar Metal shader-a.

4.2. Vizualizacija 3D modela

Nakon što se volumetrijski podaci uspješno integriraju i obrade u aplikaciji, sljedeći ključni korak je njihova vizualizacija u obliku trodimenzionalnih modela. Vizualizacija osigurava korisnicima pregled i interakciju s volumetrijskim podacima u proširenom stvarnom okruženju, što je posebno korisno u medicinskim primjenama poput dijagnostike i edukacije.

Vizualizacija 3D modela temelji se na pretvaranju sirovih podataka iz RAW formata u teksture koje GPU može brzo obraditi i prikazati. Ovi podaci se zatim prolaze kroz proces renderiranja koristeći Metal API, koji omogućava prikaz volumetrijskih podataka u stvarnom vremenu. Ključna tehnika koja se koristi za vizualizaciju volumena je *Ray Marching* algoritam, koji osigurava precizan prikaz unutarnjih struktura unutar volumena. Ovaj algoritam koristi uzorkovanje piksela duž linija zraka kroz 3D model, stvarajući detaljne prikaze unutarnjih slojeva volumena.

Prvi korak u procesu vizualizacije je kreiranje trodimenzionalnog modela iz sirovih volumetrijskih podataka. Ovi podaci se obrađuju i organiziraju u obliku 3D tekstura koje GPU može dalje koristiti za renderiranje. Teksture pružaju aplikaciji da interpretira podatke poput gustoće (Hounsfieldovih jedinica - HU vrijednosti) unutar volumena. Zatim se koristi *Ray Marching* algoritam kako bi se podaci interpretirali i prikazali, što uključuje kalkulaciju gustoće i osvjetljenja na svakoj točki volumena.

Renderiranje se odvija pomoću Metal API-ja, koji pruža visoku razinu kontrole nad grafičkom obradom podataka. Kroz korištenje specijaliziranih *shader* programa, aplikacija je u stanju prikazati teksture volumetrijskih podataka, zajedno s efektima osvjetljenja i sjena, u realnom vremenu. Ova tehnika omogućava jasno prikazivanje različitih slojeva unutar volumena, što je od velikog značaja za razumijevanje struktura unutar medicinskih podataka.

4.2.1. Obrada volumetrijskih podataka i vizualizacija u 3D modelu

Kreiranje 3D modela iz sirovih volumetrijskih podataka predstavlja ključnu fazu u pretvaranju skenova u vizualni prikaz koji se može interpretirati i analizirati. U ovoj fazi, aplikacija koristi sirove podatke (dobivene iz DICOM formata) i kroz proces manipulacije pomoću TransferFunction klase, *shader* programa i osvjetljenja, transformira ih u trodimenzionalni prikaz. Ovaj prikaz ne samo da omogućava jasan uvid u unutarnje strukture volumena, nego se koristi i u interaktivnim aplikacijama kao što su proširena stvarnost (AR) ili medicinska dijagnostika.

Jedan od ključnih elemenata kreiranja trodimenzionalnog modela je primjena transfer funkcije. S obzirom na to da sirovi podaci koji dolaze iz medicinskih skenova predstavljaju intenzitete (obično Hounsfieldove jedinice, HU), potrebno je te intenzitete mapirati na boje i prozirnost kako bi prikaz bio razumljiv i jasan. To se postiže transfer funkcijom, koja za svaki intenzitet definira specifične boje i stupnjeve prozirnosti.

Transfer funkcija koristi se za mapiranje različitih gustoća tkiva (poput kosti, mekog tkiva i tekućina) na vizualne karakteristike koje olakšavaju diferencijaciju tih struktura. Na primjer, kosti se obično prikazuju kao svijetlo bijele strukture, dok meko tkivo može biti prikazano u nijansama sive ili crvene. Prozirnost (alpha vrijednost) omogućava prikaz dubljih slojeva

unutar volumena, omogućujući korisniku pregled kroz tkiva i stjecanje potpunijeg uvida u unutarnje strukture.

TransferFunction klasa učitava podatke iz vanjske datoteke, obično u formatu .tf, koja sadrži unaprijed definirane boje i prozirnosti za određene raspone HU vrijednosti. Kao što je prikazano na slici 4.8, ova klasa mapira podatke na boje i stvara teksturu koja se zatim koristi u shader programima.

```swift
struct TransferFunction: Codable {
    var colorValue: [ColorValue] = []
    var alphaValue: [AlphaValue] = []

    var min: Float = -1024
    var max: Float = 3071
    var shift: Float = 0

    static func load(from url: URL) -> TransferFunction? {
        do {
            let data = try Data(contentsOf: url)
            let transferFunction = try JSONDecoder().decode(TransferFunction.self, from: data)
            return transferFunction
        } catch {
            print("Failed to load TransferFunction: \(error)")
            return nil
        }
    }

    func getTexture(with device: MTLDevice) -> MTLTexture? {
        let texture_width = 512
        let texture_height = 2

        var transferColors = [RGBAColor].init(repeating: RGBAColor(), count: texture_width * texture_height)

        var colors = colorValue.sorted(by: { $0.dataValue < $1.dataValue })
        var alphas = alphaValue.sorted(by: { $0.dataValue < $1.dataValue })

        colors = colors.map {
            var tempValue = $0
            tempValue.dataValue += shift
            return tempValue
        }

        alphas = alphas.map {
            var tempValue = $0
            tempValue.dataValue += shift
            return tempValue
        }

        if colors.isEmpty {
            colors.append(ColorValue(dataValue: min, colorValue: RGBAColor(r: 1, g: 1, b: 1, a: 1)))
        } else if let last = colors.last, last.dataValue < max {
            colors.append(ColorValue(dataValue: min, colorValue: RGBAColor(r: 1, g: 1, b: 1, a: 1)))
        }

        if let first = colors.first, first.dataValue > min {
            colors.insert(ColorValue(dataValue: max, colorValue: RGBAColor(r: 1, g: 1, b: 1, a: 1)), at: 0)
        }

        if alphas.isEmpty {
            alphas.append(AlphaValue(dataValue: min, alphaValue: 1))
        } else if let last = alphas.last, last.dataValue < max {
            alphas.append(AlphaValue(dataValue: min, alphaValue: 1))
        }

        if let first = alphas.first, first.dataValue > min {
            alphas.insert(AlphaValue(dataValue: max, alphaValue: 0), at: 0)
        }

        var currentColor = 0
        var currentAlpha = 0

        for widthIndex in 0 ..< texture_width {
            let normalizedPosition = Float(widthIndex) / Float(texture_width - 1)

            while currentColor < colors.count - 2,
                    normalize(colors[currentColor + 1].dataValue) < normalizedPosition {
                currentColor += 1
            }

            while currentAlpha < alphas.count - 2,
```

Slika 4.8 Implementacija TransferFunction klase

Učitavanje transfer funkcije iz vanjske datoteke omogućava fleksibilnost u prikazu volumetrijskih podataka jer korisnik može birati različite transfer funkcije ovisno o tipu skenova ili namjeni vizualizacije. Nakon što se podaci učitaju i obrade, generira se tekstura koja se zatim koristi unutar Metal API-ja.

Nakon što je transfer funkcija učitana i mapirana na volumetrijske podatke, postoji mogućnost dinamičkog prilagođavanja prikaza u realnom vremenu. Na primjer, određeni parametri transfer funkcije, poput vrijednosti pomaka, mogu se koristiti za pomicanje raspona HU vrijednosti kako bi se naglasili određeni aspekti volumena (poput kostiju ili mekih tkiva).

Funkcija setShift, prikazana na slici 4.9, koristi se za pomicanje HU raspona koji se prikazuje. Omogućuje korisnicima da "izrežu" određene dijelove volumena ili naglase specifične slojeve unutar volumena. Ova funkcionalnost omogućuje korisnicima precizno fokusiranje na određene dijelove skeniranih podataka, čime se poboljšava interaktivnost i korisničko iskustvo u aplikaciji.

```
func setShift(device: MTLDevice, shift: Float) {
    transferFunction?.shift = shift
    guard let transferFunction else {
        print("Guard failed: Material - transferFunction")
        return
    }
    guard let transferFunctionTexture = transferFunction.getTexture(with: device) else {
        print("Guard failed: Material - transferFunctionTexture")
        return
    }
    let transferFunctionProperty = SCNMaterialProperty(contents: transferFunctionTexture)
    setValue(transferFunctionProperty, forKey: "transferColor")
}
```

Slika 4.9 Funkcija setShift

Osim primjene transfer funkcije, ključan element u prikazu volumetrijskih podataka je implementacija osvjetljenja i sjena. Osvjetljenje daje osjećaj trodimenzionalnosti i dubine, omogućujući korisnicima da bolje interpretiraju strukture unutar volumena. Shader programi unutar Metal API-ja koriste gradijente kako bi izračunali normalne vektore za svaki voxel unutar volumena. Ti vektori zatim omogućuju izračun osvjetljenja i sjena.

Na slici 4.10 prikazana je implementacija fragment shader programa koji koristi gradijente volumetrijskih podataka kako bi izračunao smjer svjetla i generirao sjene unutar volumena. Normalni vektori generiraju se na temelju promjena gustoće (HU vrijednosti) između susjednih

voxela. Osvjetljenje se zatim prilagođava na temelju smjera svjetla i orijentacije volumena, stvarajući efekt dubine i trodimenzionalnosti.

```
static float3 calculateLighting(float3 col, float3 normal, float3 lightDir, float3 eyeDir,
                        float specularIntensity)
{
    float ndotl = max(lerp(0.0, 1.5, dot(normal, lightDir)), 0.5);
    float3 diffuse = ndotl * col;
    float3 v = eyeDir;
    float3 r = ::normalize(reflect(-lightDir, normal));
    float rdotv = max(dot(r, v), 0.0);
    float3 specular = pow(rdotv, 32) * float3(1) * specularIntensity;
    return diffuse + specular;
}
```

Slika 4.10 Izračun osvjetljenja

Ovaj proces omogućuje precizan prikaz unutarnjih struktura unutar volumena, jer osvjetljenje i sjene pomažu korisnicima da bolje razumiju prostorne odnose između različitih dijelova volumena. Korištenjem Metal API-ja, ovi izračuni se mogu vrlo brzo izvršavati na GPU-u, čime se osigurava interaktivno korisničko iskustvo.

4.3. Integracija proširene stvarnosti

Integracija proširene stvarnosti (AR) predstavlja ključni korak u transformaciji 3D modela volumetrijskih podataka u interaktivno iskustvo unutar stvarnog okruženja. Ovaj segment aplikacije omogućava korisnicima da volumetrijske modele, kao što su medicinski CT ili MRI skenovi, smjeste u fizički prostor i interaktivno istražuju složene podatke na intuitivan način. Korištenjem tehnologija kao što su ARKit i SceneKit, aplikacija može detektirati stvarne površine, smjestiti volumetrijski model unutar AR prostora, te omogućiti korisnicima interakciju s modelom, kao što su rotacija, skaliranje i pomicanje.

Korištenje proširene stvarnosti pruža korisnicima pregled trodimenzionalnih podataka u prirodnom kontekstu, gdje se spajaju fizički svijet i digitalne informacije. To je od posebne važnosti u medicinskim aplikacijama, gdje AR omogućava zdravstvenim djelatnicima, studentima medicine i istraživačima da detaljno analiziraju složene anatomske strukture u stvarnom prostoru, što poboljšava njihovo razumijevanje odnosa između različitih tkiva i organa. Na taj način, proširena stvarnost olakšava interpretaciju i analizu volumetrijskih podataka, što bi u tradicionalnom 2D prikazu bilo znatno teže.

Prvi korak u ovoj integraciji je detekcija stvarnog prostora, što uključuje prepoznavanje ravnih površina poput stolova, poda ili zidova u fizičkom okruženju. ARKit koristi kameru uređaja i napredne algoritme računalnog vida za prepoznavanje tih elemenata, čime aplikacija može točno smjestiti 3D model u prostor. Kada su površine prepoznate, aplikacija može pristupiti informacijama o njihovoj orijentaciji i veličini, što osigurava pravilno postavljanje volumetrijskog modela unutar AR scene.

Nakon detekcije prostora, postavljanje volumetrijskog modela ključan je korak kako bi se osiguralo da model bude točno skaliran i pozicioniran unutar prostora. Volumetrijski podaci se mapiraju na prepoznate površine u prostoru, osiguravajući ispravnu orijentaciju i veličinu modela u odnosu na stvarno okruženje. Ova faza uključuje izračune orijentacije i pozicioniranja modela kako bi se osigurao pravilan prikaz, uzimajući u obzir fizičke dimenzije prostora.

Konačno, aplikacija korisnicima pruža interakciju s modelom, što uključuje mogućnosti rotacije, skaliranja i pomicanja volumena unutar AR scene. Ova interaktivnost korisnicima omogućuje detaljnu analizu volumetrijskih podataka iz različitih kutova i perspektiva. Kroz jednostavne gestovne kontrole, korisnik može mijenjati položaj volumetrijskog modela, rotirati ga kako bi pregledao određene dijelove iz različitih uglova, ili promijeniti njegovu veličinu kako bi bolje prikazao unutarnje strukture. Ova razina interaktivnosti čini AR tehnologiju izuzetno korisnom za medicinsku edukaciju, dijagnostiku i istraživanje.

Integracija proširene stvarnosti u aplikaciju omogućava napredan, intuitivan i prirodan način interakcije s volumetrijskim podacima. Na taj način, proširena stvarnost olakšava interpretaciju složenih struktura i osigurava dublje razumijevanje prostornih odnosa unutar volumena. Kombinacija proširene stvarnosti i vizualizacije volumetrijskih podataka može imati široku primjenu, posebno u medicinskim aplikacijama, gdje ova tehnologija nudi značajne prednosti u dijagnostici i edukaciji.

4.3.1. Detekcija ravnina u AR-u

Detekcija ravnina u proširenoj stvarnosti ključan je aspekt AR iskustva jer aplikaciji pomaže da razumije fizički prostor u kojem se nalazi. ARKit koristi kameru i senzore uređaja za prepoznavanje karakterističnih točaka u okruženju, grupirajući ih kako bi identificirao

horizontalne i vertikalne ravnine poput poda, stolova ili zidova. Ove ravnine služe kao osnove na koje se mogu postaviti 3D objekti, uključujući volumetrijske modele.

Prvi korak u detekciji površina jest pravilna konfiguracija AR sesije. To se postiže korištenjem klase ARWorldTrackingConfiguration koja prati položaj uređaja u stvarnom svijetu i detektira horizontalne i vertikalne površine. Kao što je prikazano na slici 4.10, AR sesija se pokreće pomoću odgovarajuće konfiguracije koja osigurava detekciju ravnih površina u prostoru. Ova funkcija postavlja AR sesiju na način da ARKit kontinuirano detektira horizontalne i vertikalne ravnine u stvarnom prostoru. ARWorldTrackingConfiguration koristi kameru i senzore uređaja za praćenje položaja i orijentacije uređaja u odnosu na detektirane površine.

```
func createARConfiguration() -> ARConfiguration {
  let config = ARWorldTrackingConfiguration()
  config.worldAlignment = .gravity
  config.planeDetection = [.horizontal]
  config.isLightEstimationEnabled = true
  return config
}
```

Slika 4.10 Funkcija createARConfiguration

Osim toga, funkcija isLightEstimationEnabled prilagođava osvjetljenje virtualnih objekata stvarnim uvjetima osvjetljenja, čime se postiže prirodniji prikaz unutar AR okruženja. Korištenjem ARWorldTrackingConfiguration, aplikacija je sposobna prepoznati i pratiti ravne površine unutar fizičkog prostora, pripremajući scenu za postavljanje volumetrijskih modela. To stvara temelje za postavljanje modela unutar proširenog stvarnog svijeta, o čemu će detaljnije biti riječi u sljedećem podnaslovu.

4.3.2. Postavljanje modela u AR prostoru

Nakon što je ARKit detektirao ravnine unutar stvarnog okruženja, sljedeći ključni korak je postavljanje volumetrijskog modela na te površine. Ovaj proces omogućuje korisnicima da vizualiziraju volumetrijski model (poput CT ili MRI skenova) u stvarnom svijetu, čime se proširuje iskustvo kroz interaktivnu proširenu stvarnost. Postavljanje modela na detektirane površine postiže se korištenjem SCNNode objekata i informacija iz ARPlaneAnchor. Funkcija addVolume zadužena je za dodavanje modela na površinu unutar AR okruženja, dok

ARViewerContentView koristi te informacije kako bi precizno postavila model na željeno mjesto.

Kada aplikacija prepozna ravninu, koristi se objekt SCNNode kako bi prikazala volumetrijski model u AR sceni. SCNNode je osnovni građevni blok unutar SceneKit-a, odgovoran za definiranje položaja, orijentacije i veličine objekta u prostoru. Pomoću ARPlaneAnchor, aplikacija može dobiti podatke o stvarnim površinama, kao što su pozicija i dimenzije detektirane površine, kako bi točno postavila volumetrijski model na prepoznatu površinu.

Funkcija renderer zadužena je za prepoznavanje kada je nova površina detektirana, te inicijalno postavljanje volumetrijskog modela na tu površinu. Na slici 2.1 prikazana je funkcija renderer, koja detektira kada je nova površina dodana pomoću ARPlaneAnchor. Funkcija zatim poziva addVolume, koja koristi informacije iz planeAnchor kako bi dodala volumetrijski model na detektiranu ravnu površinu. Ova funkcija osigurava da se model prikazuje na pravom mjestu unutar AR okruženja.

```
func renderer(_ renderer: any SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {
    guard let planeAnchor = anchor as? ARPlaneAnchor else { return }
    if !isModelShown {
        addVolume(on: node, for: planeAnchor)
    }
}
```

Slika 4.11 Funkcija renderer

Funkcija addVolume ključna je za postavljanje volumetrijskog modela na detektiranu površinu unutar AR okruženja. Kada je nova površina detektirana, ARPlaneAnchor pruža informacije o njenoj poziciji i dimenzijama, a addVolume koristi te informacije kako bi točno postavila model na središte površine. Funkcija također postavlja orijentaciju modela kako bi odgovarala fizičkoj orijentaciji površine. Na slici 4.12, addVolume koristi podatke iz planeAnchor kako bi točno postavila model na ravnu površinu. Informacije o poziciji detektirane površine pohranjuju se u objektu ARPlaneAnchor, a volumetrijski model se pozicionira na središte te površine. Funkcija također osigurava pravilnu orijentaciju modela, postavljajući ga pod odgovarajućim kutem, te skalira model kako bi odgovarao fizičkim proporcijama detektirane površine.

```swift
func addVolume(on node: SCNNode, for planeAnchor: ARPlaneAnchor) {
    material = Material(device: device!, data: currentShownData)
    material?.setPart(device: device!, data: currentShownData)
    guard let material else { return }

    volume = SCNNode(geometry: SCNBox(width: 1, height: 1, length: 1, chamferRadius: 0))
    volume?.position = SCNVector3(planeAnchor.center.x,
                                  planeAnchor.center.y,
                                  planeAnchor.center.z)

    volume?.eulerAngles = SCNVector3(0, -Double.pi / 2, 0)

    volume?.geometry?.materials = [material]

    guard let scale = material.scale else { return }
    print("Scale value: \(scale)")

    volume?.scale = SCNVector3(x: (scale.x/1000)*3, y: (scale.y/1000)*3, z: (scale.z/1000)*3)

    guard let volume else { return }
    cameraController?.target = volume.boundingSphere.center

    node.addChildNode(volume)
    isModelShown = true
}
```

Slika 4.12 Funkcija addVolume

Klasa ARViewerContentView upravlja prikazom volumetrijskog modela unutar AR okruženja. Ova klasa koristi informacije iz ARKit sesije i upravlja elementima scene, poput SCNNode, kako bi se osiguralo da je model pravilno prikazan unutar proširenog stvarnog svijeta. ARViewerContentView također omogućava korisniku interakciju s modelom, kao što su rotacija i skaliranje, što ćemo detaljnije obraditi u sljedećem dijelu. Na slici 2.3 prikazana je ARViewerContentView klasa koja integrira ARKit i SceneKit kako bi upravljala prikazom volumetrijskih modela unutar AR okruženja. Kroz ovu klasu, volumetrijski model se pravilno prikazuje unutar scene, koristeći informacije iz detektiranih površina.

```swift
class ARViewerContentView: UIView {
    private var sceneView = ARSCNView()
    var material: Material?
    private var device: MTLDevice?
    var currentShownData: VolumeData
    private var volume: SCNNode?
    var isModelShown = false
    private var cameraController: SCNCameraController?

    init(data: VolumeData) {
        self.currentShownData = data
        super.init(frame: .zero)
        initSceneView()
        initARSession()
        setupGestureRecognizers()
    }
```

Slika 4.13 Klasa ARViewerContentView

4.3.3. Interakcija korisnika s modelom

Nakon što je volumetrijski model postavljen unutar proširenog stvarnog svijeta, sljedeći važan korak je omogućiti korisnicima interakciju s modelom. Interaktivnost je ključni aspekt proširene stvarnosti, jer pruža korisnicima da istražuju i prilagođavaju modele na način koji najbolje odgovara njihovim potrebama. U ovom dijelu aplikacije, korisnici mogu rotirati i pomicati volumetrijskog modela, što je važno za detaljno istraživanje unutarnjih struktura modela.

Korištenjem SceneKit i ARKit funkcionalnosti, aplikacija implementira gestovne kontrole koje korisnicima omogućuju da rotiraju i pomiču model unutar AR okruženja. Ove funkcije koriste UIPanGestureRecognizer kako bi prepoznale pokrete korisnika i primijenile ih na volumen.

Interakcija s modelom korisnicima pruža mogućnost rotacije modela u prostoru kako bi pregledali različite kutove i slojeve. Korištenjem UIPanGestureRecognizer, aplikacija prepoznaje horizontalne i vertikalne pokrete prstom i prilagođava orijentaciju volumena u skladu s tim. Kao što je prikazano na slici 3.1, aplikacija koristi UIPanGestureRecognizer za rotaciju modela u AR okruženju. Korisnički pokreti prstom u horizontalnom i vertikalnom smjeru mijenjaju kutove modela, omogućujući pregled iz različitih perspektiva. Funkcija handlePanGesture prati translaciju gesti i ažurira rotaciju modela pomoću eulerAngles.

```
private func setupGestureRecognizers() {
    let panGesture = UIPanGestureRecognizer(target: self, action: #selector(handlePanGesture(_:)))
    sceneView.addGestureRecognizer(panGesture)
}

@objc private func handlePanGesture(_ gesture: UIPanGestureRecognizer) {
    guard let volume = volume else { return }
    let translation = gesture.translation(in: sceneView)
    let newAngleY = Float(translation.x) * (-Float.pi / 180.0)
    let newAngleX = Float(translation.y) * (-Float.pi / 180.0)

    volume.eulerAngles.y -= newAngleY
    volume.eulerAngles.x -= newAngleX

    gesture.setTranslation(.zero, in: sceneView)
}
```

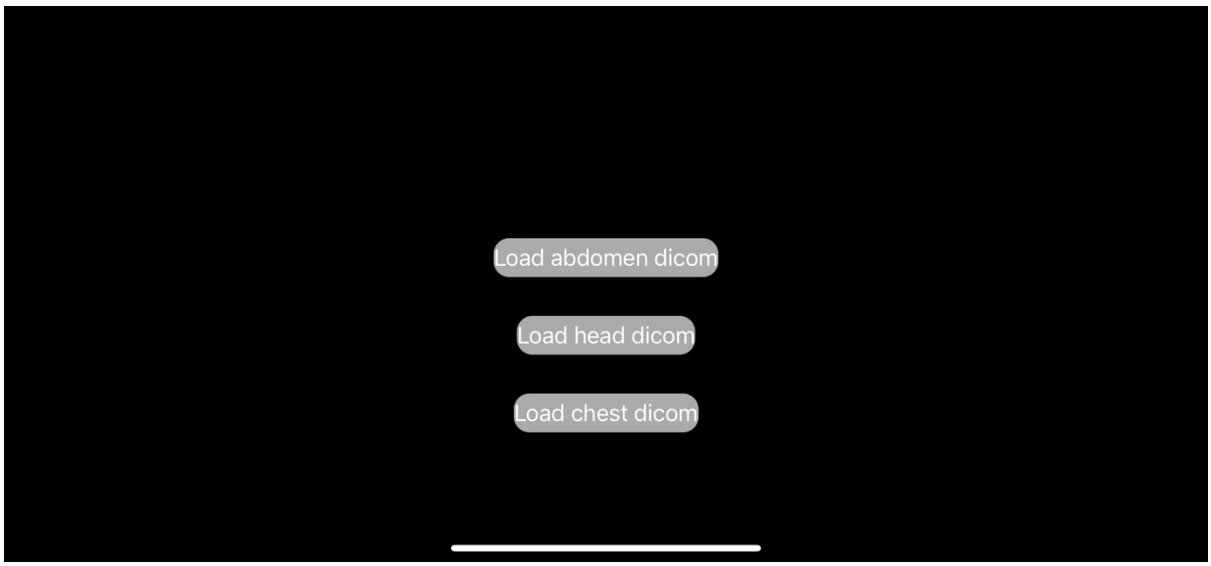
Slika 4.14 Funkcija handlePanGesture

5. REZULTATI

U ovom poglavlju prikazani su rezultati funkcionalnosti i performansi iOS aplikacije za vizualizaciju medicinskih podataka u proširenoj stvarnosti. Aplikacija je testirana na nekoliko ključnih aspekata, uključujući interakciju korisnika s modelom, performanse u realnom vremenu te mogućnosti. U nastavku se detaljnije analiziraju funkcionalnosti aplikacije, performanse u različitim uvjetima te identificirani nedostaci s prijedlozima za poboljšanja.

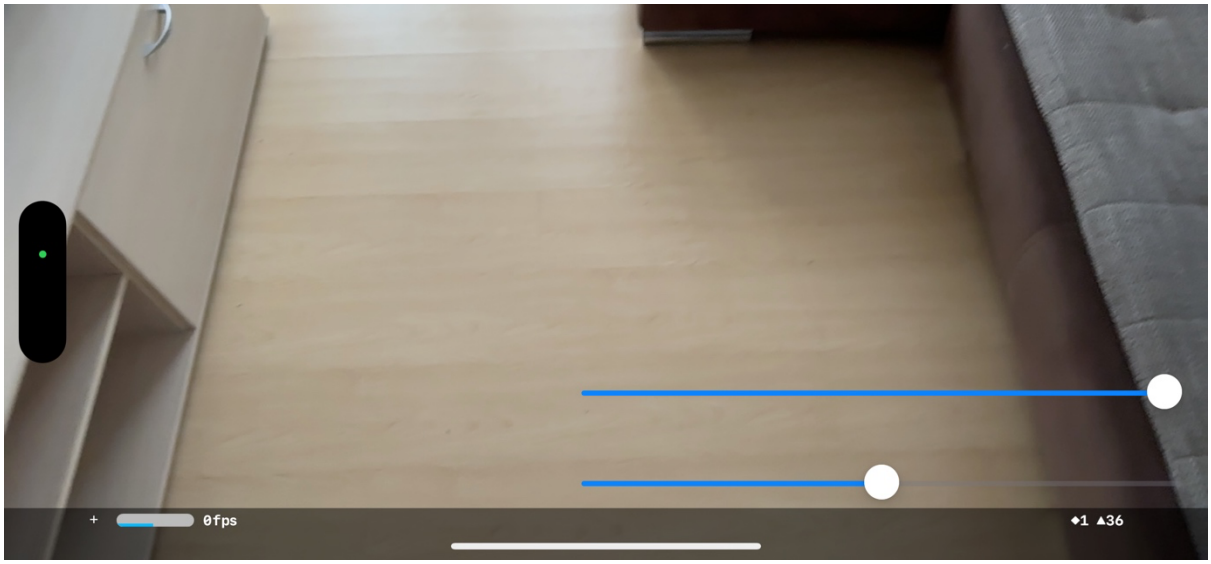
5.1. Pregled funkcionalnosti aplikacije

Aplikacija korisniku pruža intuitivnu vizualizaciju medicinskih podataka pomoću proširene stvarnosti. Nakon pokretanja aplikacije, korisniku se prikazuje glavni izbornik, kao sto je prikazano na slici 5.1, na kojem može odabrati jednu od tri DICOM datoteke, koje će se koristiti za prikaz modela.



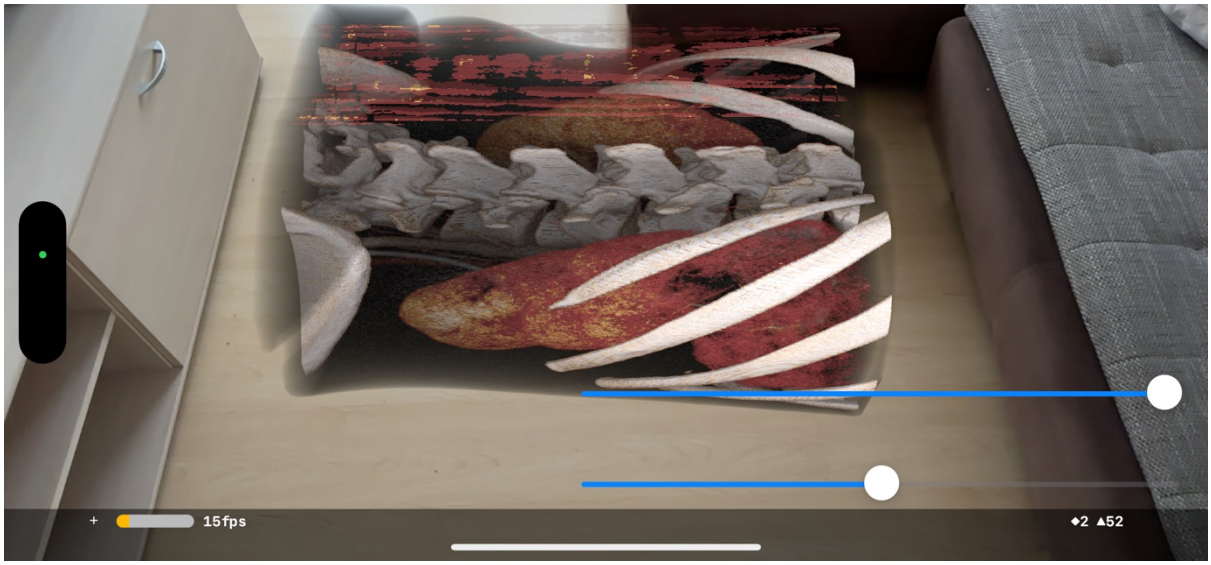
Slika 5.1 Prikaz glavnog izbornika

Nakon što korisnik odabere DICOM datoteku, aktivira se kamera uređaja, kao sto je prikazano na slici 5.2, te započinje proces prepoznavanja ravne površine u stvarnom okruženju. Proširena stvarnost koristi tehnologiju ARKit za detekciju horizontalnih površina, kao što su stolovi ili podovi, na koje se zatim postavlja 3D model generiran iz odabrane DICOM datoteke. Ovaj korak je ključan za stabilno pozicioniranje modela u prostoru, čime se osigurava pravilna interakcija korisnika s modelom.

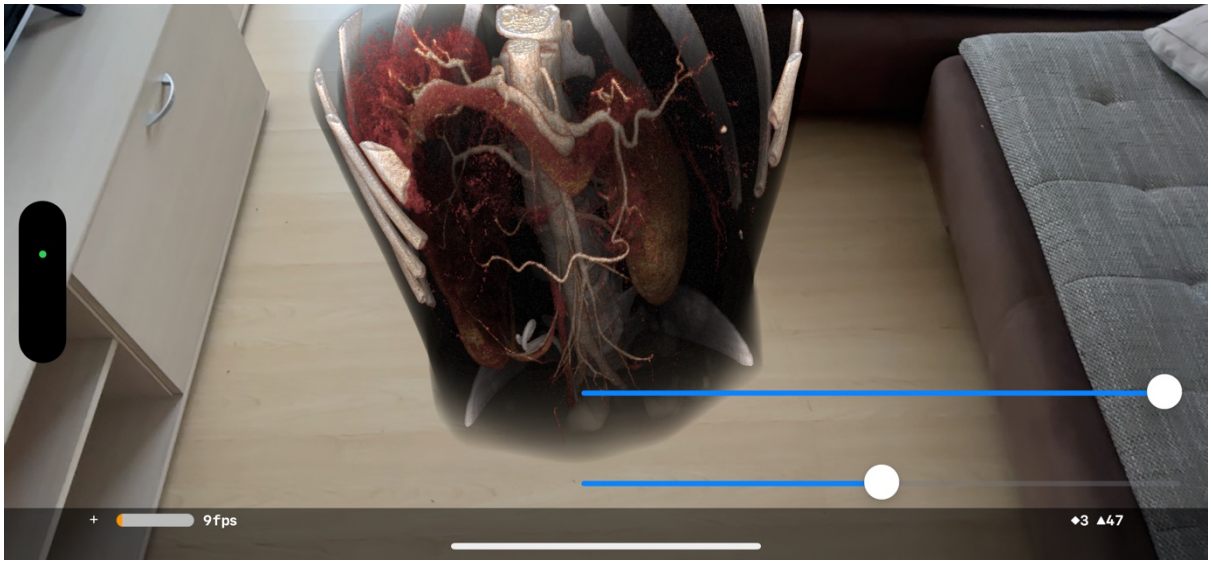


Slika 5.2 Detekcija ravne površine

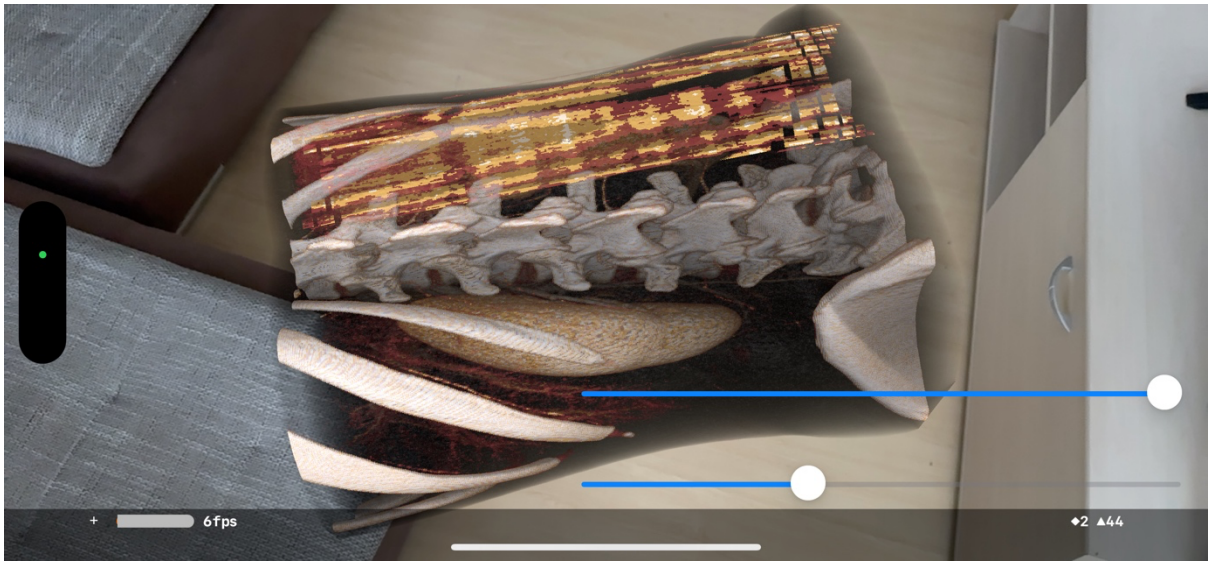
Kada je ravna površina detektirana, 3D model se automatski prikazuje na toj površini što je prikazano na slici 5.3. Korisnik tada može pregledati model iz različitih perspektiva, koristeći uređaj za kretanje oko modela ili rotiranje istog unutar aplikacije kao što je prikazano na slici 5.4 i slici 5.5. Model ostaje postavljen na detektiranu površinu, dok korisnik može slobodno manipulirati prikazom.



Slika 5.3 Prikaz modela

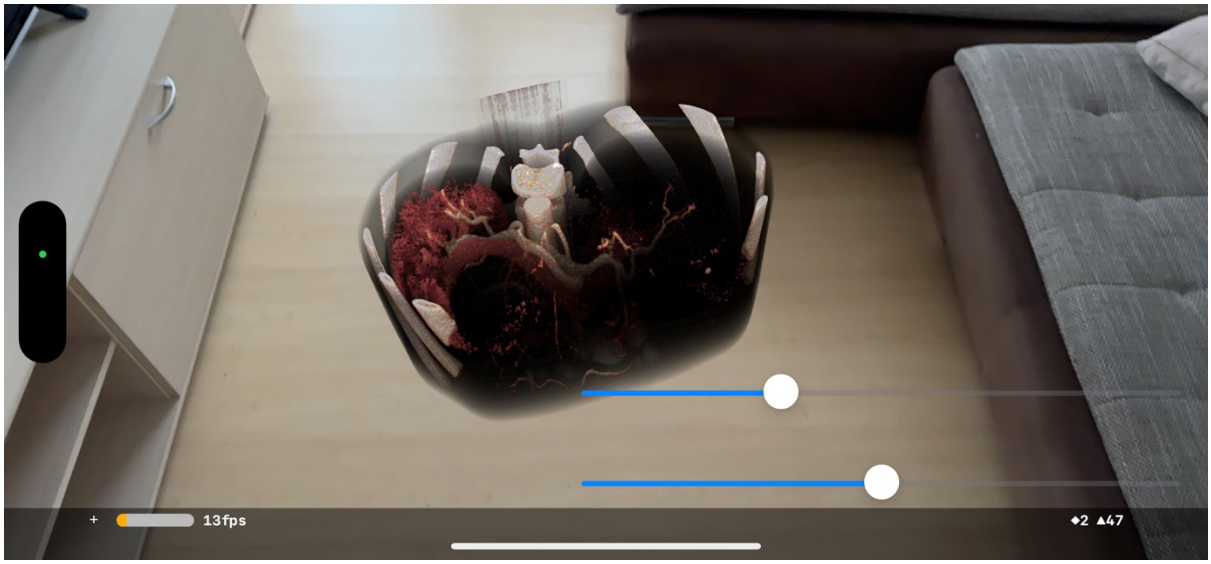


Slika 5.5 Prikaz okretanja modela



Slika 5.6 Prikaz iz drugog kuta prostorije

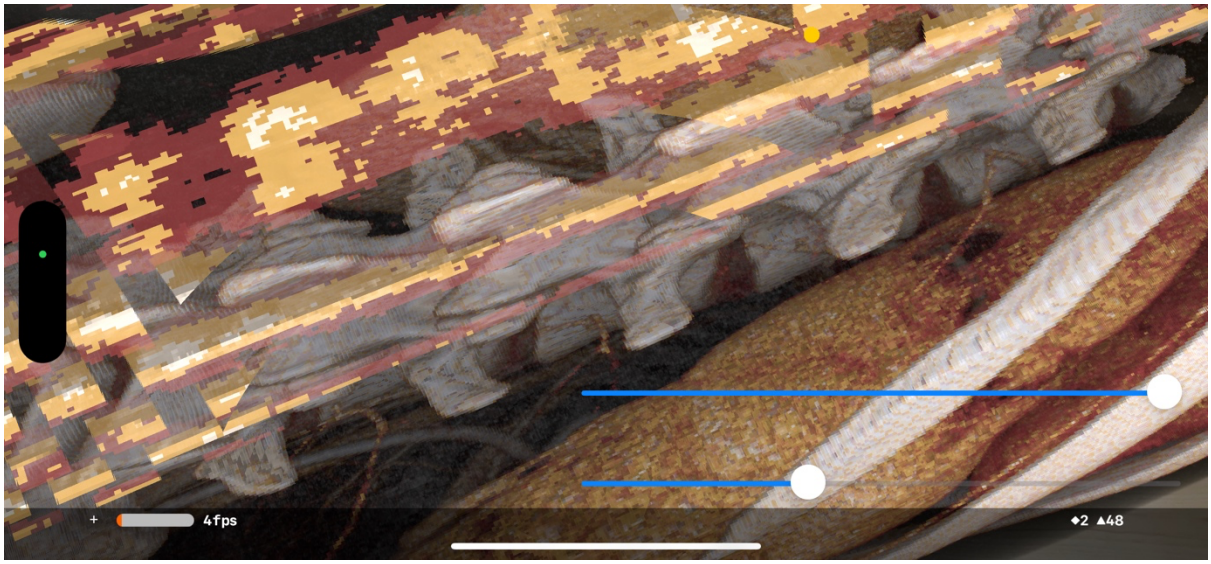
Osim toga, aplikacija nudi dvije dodatne interaktivne kontrole: slider za shift i slider za slice. Slider za shift pomiče raspon prikazanih vrijednosti unutar modela, dok slider za slice omogućuje korisniku da napravi presjek modela i pregleda unutarnje strukture (Slika 3: Interaktivni prikaz modela s dostupnim sliderima).



Slika 5.7 Prikaz s pomaknutim sliderima

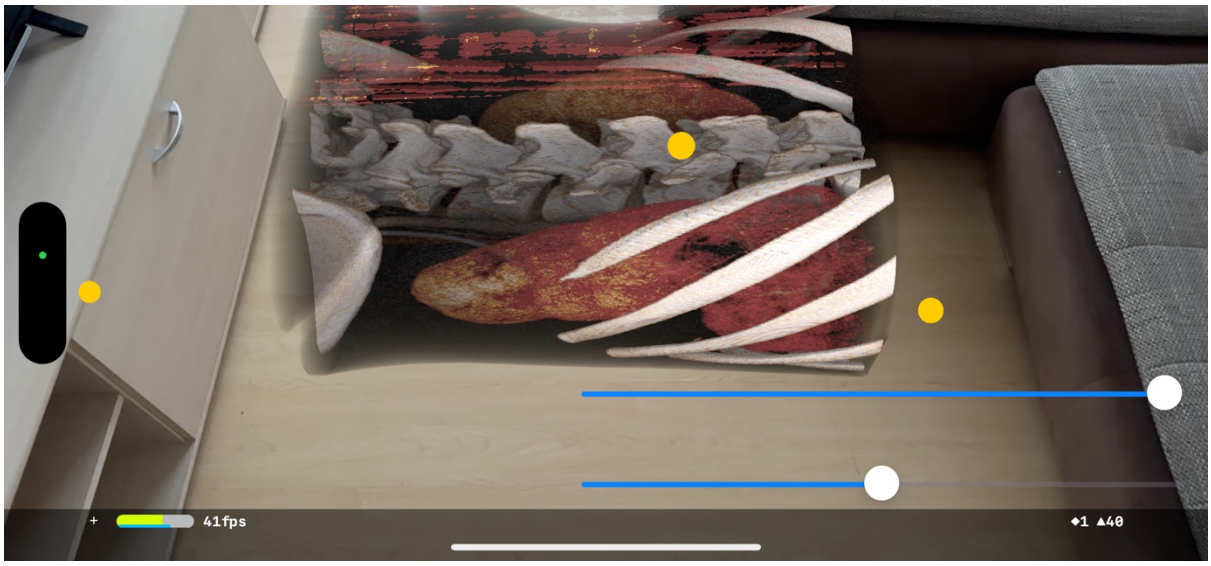
5.2. Performanse aplikacije

Performanse aplikacije testirane su na uređaju iPhone 14 Pro Max, koji nudi visok nivo hardverskih kapaciteta za proširenu stvarnost i grafički intenzivne aplikacije. Tijekom testiranja korišten je Xcode Performance Monitor, koji omogućuje praćenje korištenja resursa poput CPU-a i GPU-a te broj sličica u sekundi (FPS), ključnih za procjenu fluidnosti aplikacije. Prilikom prikaza 3D modela u proširenoj stvarnosti, zabilježena je značajna varijabilnost u FPS-u ovisno o udaljenosti kamere od modela. Kada se kamera približi modelu, dolazi do smanjenja FPS-a, što je posljedica povećanog opterećenja GPU-a zbog renderiranja više detalja na ekranu. U takvim situacijama, kao sto je prikazano na slici 5.8, broj sličica u sekundi može pasti, a korištenje CPU-a doseže gotovo 100%, što ukazuje na intenzivan proces renderiranja.



Slika 5.8 Prikaz pada FPS-a i visokog opterećenja CPU-a pri približenju modelu

S druge strane, kad se korisnik udalji od modela, FPS se povećava jer GPU obrađuje manje složen prikaz modela, smanjujući detalje koje treba renderirati. Ovo rezultira manjim opterećenjem resursa, pri čemu CPU korištenje opada na približno 80%, što je prikazano na slici 5.9.



Slika 5.9 Prikaz povećanja FPS-a i smanjenja opterećenja CPU-a pri udaljenijem prikazu modela

5.3. Analiza nedostataka

Iako aplikacija nudi intuitivnu interakciju i mogućnost vizualizacije medicinskih podataka u proširenoj stvarnosti, tijekom razvoja i testiranja identificirano je nekoliko ključnih nedostataka koji bi mogli biti poboljšani u budućim verzijama aplikacije.

Jedan od glavnih nedostataka aplikacije je izostanak integracije s backend sustavom. Trenutna verzija aplikacije prikazuje unaprijed definirane DICOM datoteke, ali ne podržava učitavanje korisničkih DICOM datoteka. Ova funkcionalnost bi korisnicima omogućila učitavanje vlastitih medicinskih podataka i njihovu vizualizaciju u proširenoj stvarnosti, što bi značajno povećalo fleksibilnost i korisnost aplikacije u kliničkim i edukativnim okruženjima. Implementacija backend sustava, koji bi podržavao prijenos i pohranu korisničkih podataka, predstavlja važan korak u daljnjem razvoju aplikacije.

Drugi uočeni nedostatak odnosi se na performanse aplikacije pri radu s detaljnijim 3D modelima. Kao što je spomenuto u prethodnom dijelu, prilikom približavanja modelu dolazi do pada FPS-a, a CPU i GPU postižu gotovo maksimalno opterećenje. Iako su ove performanse prihvatljive za uređaj poput iPhone 14 Pro Max, stariji ili manje moćni uređaji mogli bi imati poteškoća u održavanju fluidnosti prikaza. Ovo ukazuje na potrebu za dodatnom optimizacijom aplikacije, osobito u dijelu koji se odnosi na renderiranje složenih modela i upravljanje resursima. Moguće rješenje bi uključivalo smanjenje detalja modela ili optimizaciju algoritma za prikaz kako bi se smanjilo opterećenje na GPU i CPU, a zadržala kvaliteta prikaza.

Također, trenutačna verzija aplikacije ne sadrži naprednije opcije interakcije s modelima, poput mogućnosti izmjene ili uređivanja DICOM podataka. Iako je trenutna implementacija dovoljno interaktivna za pregled modela, korisnici bi mogli imati koristi od dodatnih funkcija, kao što su anotacije, mjerenja ili segmentacija specifičnih dijelova modela. Ove funkcionalnosti bi dodatno obogatile iskustvo korisnika i omogućile širu primjenu aplikacije u medicinskoj edukaciji i dijagnostici.

ZAKLJUČAK

U ovom radu istražuje se primjena proširene stvarnosti (AR) u vizualizaciji medicinskih podataka, s naglaskom na računalnu tomografiju (CT). Proširena stvarnost, kao ključna komponenta modernih tehnologija, nudi nove pristupe interakciji s digitalnim sadržajem u različitim područjima, uključujući medicinu, obrazovanje i inženjering. Razvoj aplikacija koje koriste AR tehnologiju pruža korisnicima da u stvarnom vremenu pregledavaju složene podatke, što ranije nije bilo moguće. U ovom slučaju, naglasak je na medicinskoj vizualizaciji, gdje se istražuje kako se računalni podaci dobiveni CT skeniranjem mogu učinkovito prikazati unutar AR okruženja na iOS platformi.

Predmet istraživanja u radu je razvoj i implementacija iOS aplikacije za vizualizaciju 3D modela dobivenih iz CT snimaka u proširenoj stvarnosti. U tu svrhu, posebna pažnja posvećuje se procesu konverzije DICOM (Digital Imaging and Communications in Medicine) datoteka u RAW format, što omogućuje daljnju obradu podataka korištenjem Metal API-ja za renderiranje volumetrijskih podataka. Korištenje proširene stvarnosti pruža intuitivnu interakciju s medicinskim podacima, pružajući korisnicima značajan napredak u medicinskom obrazovanju, dijagnostici i planiranju kirurških zahvata. Cilj istraživanja je dizajnirati i implementirati rješenje za prikaz CT snimaka kao trodimenzionalnih modela u stvarnom okruženju, koristeći proširenu stvarnost na iOS uređajima. U procesu razvoja aplikacije postavljaju se specifični ciljevi kao što su konverzija DICOM datoteka, renderiranje volumetrijskih podataka korištenjem Metal API-ja, integracija ARKit-a za detekciju površina i postavljanje 3D modela unutar AR prostora, te evaluacija performansi aplikacije s obzirom na kvalitetu prikaza i korisničko iskustvo.

Metodologija rada obuhvaća nekoliko ključnih faza. Na početku se provodi istraživanje postojećih tehnologija koje podržavaju vizualizaciju medicinskih podataka u proširenoj stvarnosti, s posebnim naglaskom na DICOM format, ARKit i Metal API, kao i tehnike poput Ray Marching-a za renderiranje volumena. Sljedeći korak uključuje razvoj Python skripte za konverziju DICOM datoteka u RAW format, što omogućuje daljnju obradu tih podataka u aplikaciji. Implementacija aplikacije izvršava se korištenjem Swift programskog jezika i UIKit framework-a, integrirajući ARKit i Metal API za renderiranje i prikaz 3D modela CT snimaka unutar AR okruženja. Nakon toga, provode se testiranja aplikacije na različitim iOS uređajima

kako bi se osigurala kvaliteta renderiranja, performanse aplikacije te korisničko iskustvo, a rezultati se evaluiraju kroz analizu performansi aplikacije i povratne informacije korisnika.

Struktura rada osmišljena je kako bi sistematično prikazala sve faze istraživanja. Uvodni dio pruža pregled predmeta i ciljeva rada, metodološkog pristupa te značaja istraživanja. Slijede teorijske osnove, gdje se raspravlja o ključnim tehnologijama i konceptima kao što su računalna tomografija, DICOM format i proširena stvarnost, kao i tehnologije korištene u razvoju aplikacije. U nastavku se detaljno opisuju alati i tehnologije korišteni za razvoj aplikacije, s posebnim naglaskom na iOS razvojne okvire poput Swift-a, Metal API-ja i ARKit-a. Glavni dio rada bavi se razvojem aplikacije, od obrade medicinskih podataka do njihove vizualizacije u AR okruženju, dok završni dio uključuje rezultate, diskusiju i zaključke, gdje se analiziraju postignuti rezultati, ograničenja rada i potencijalni smjerovi za daljnji razvoj.

Značaj ovog istraživanja ogleda se u njegovoj praktičnoj primjeni. Razvoj aplikacije za vizualizaciju medicinskih podataka u proširenoj stvarnosti pruža značajne prednosti u medicinskom obrazovanju, gdje može poslužiti kao edukativni alat za bolje razumijevanje složenih struktura prikazanih na CT snimkama. U medicinskoj praksi, aplikacija pruža liječnicima mogućnost detaljnije analize CT podataka u proširenoj stvarnosti, što može pomoći u preciznijoj dijagnostici i planiranju kirurških zahvata. Također, istraživanje doprinosi tehnološkom razvoju u području računalne grafike i proširene stvarnosti, naglašavajući njezinu praktičnu primjenu u medicini.

Literatura

[1] "Computed Tomography (CT) | FDA." Accessed: Sep. 06, 2024. [Online]. Available: https://www.fda.gov/radiation-emitting-products/medical-x-ray-imaging/computed-tomography-ct

[2] "Computed Tomography (CT)." Accessed: Sep. 06, 2024. [Online]. Available: https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct

[3] "Computed Tomography - CT Scan - CAT Scan - Neurology - Highland Hospital - University of Rochester Medical Center." Accessed: Sep. 06, 2024. [Online]. Available: https://www.urmc.rochester.edu/highland/departments-centers/neurology/tests-treatments/ct-brain.aspx

[4] "What Does A Cardiac CT Scan Show or Detect and When Do You Need It? - UChicago Medicine." Accessed: Sep. 06, 2024. [Online]. Available: https://www.uchicagomedicine.org/conditions-services/heart-vascular/cardiovascular-imaging/cardiac-ct-angiography

[5] "Computed Tomography (CT) Scans and Cancer Fact Sheet - NCI." Accessed: Sep. 06, 2024. [Online]. Available: https://www.cancer.gov/about-cancer/diagnosis-staging/ct-scans-fact-sheet

[6] Dr Ravin Sharma, "What Is The Role Of CT Scan In Trauma? | Ganesh Diagnostic." Accessed: Sep. 06, 2024. [Online]. Available: https://www.ganeshdiagnostic.com/blog/what-is-the-role-of-ct-scan-in-trauma

[7] "Intraoperative CT | Johns Hopkins Medicine." Accessed: Sep. 06, 2024. [Online]. Available: https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/intraoperative-ct

[8] "About DICOM: Overview." Accessed: Sep. 06, 2024. [Online]. Available: https://www.dicomstandard.org/about-home

[9] "DICOM & PACS - Exploring the Differences in Medical Imaging." Accessed: Sep. 06, 2024. [Online]. Available: https://blog.medicai.io/en/dicom-and-pacs-exploring-the-differences-in-medical-imaging

[10] "Augmented Reality (AR): Definition, Examples, and Uses." Accessed: Sep. 06, 2024. [Online]. Available: https://www.investopedia.com/terms/a/augmented-reality.asp

[11] "Stanford Medicine uses augmented reality for real-time data visualization during surgery | News Center | Stanford Medicine." Accessed: Sep. 06, 2024. [Online].

Available: https://med.stanford.edu/news/all-news/2024/02/augmented-reality-surgery.html

[12] “Swift Pathway - Swift - Apple Developer.” Accessed: Sep. 06, 2024. [Online]. Available: https://developer.apple.com/swift/pathway/

[13] “Swift - Apple Developer.” Accessed: Sep. 06, 2024. [Online]. Available: https://developer.apple.com/swift/

Sažetak

Ovaj diplomski rad bavi se razvojem i implementacijom iOS aplikacije za vizualizaciju medicinskih podataka, konkretno CT snimaka, u proširenoj stvarnosti (AR). Kroz primjenu tehnologija kao što su Metal API, ARKit i Swift programski jezik, razvijena je aplikacija koja korisnicima pruža mogućnost da u stvarnom vremenu pregledavaju i interaktivno manipuliraju trodimenzionalnim modelima dobivenim iz DICOM datoteka. Proces obrade medicinskih podataka obuhvaća konverziju DICOM datoteka u RAW format, čime se omogućava brža i efikasnija obrada volumetrijskih podataka unutar aplikacije. Aplikacija nudi intuitivnu interakciju korisnika s modelima, uključujući rotaciju, skaliranje i pregled unutarnjih struktura modela, čime se poboljšava korisničko iskustvo u edukativnom i kliničkom okruženju. Rad također pruža pregled ključnih tehnologija i algoritama korištenih u procesu razvoja, kao što je Ray Marching za renderiranje volumetrijskih podataka. Osim tehničkih aspekata razvoja, rad se fokusira na primjenu proširene stvarnosti u medicini, naglašavajući njenu korisnost u obrazovanju i dijagnostici, te daje uvid u izazove i potencijalne smjerove daljnjeg razvoja aplikacije. Aplikacija predstavlja značajan doprinos na polju medicinske vizualizacije, omogućavajući liječnicima i studentima medicine da na inovativan način pristupe analiziranju medicinskih podataka.

Ključne riječi: ARKit, DICOM format, Interaktivna medicinska vizualizacija, Metal API, Proširena stvarnost, Računalna tomografija, Vizualizacija volumetrijskih podataka

Abstract

Visualization of computed tomography images in augmented reality on the iOS platform

This thesis focuses on the development and implementation of an iOS application for the visualization of medical data, specifically CT scans, in augmented reality (AR). Through the application of technologies such as Metal API, ARKit, and the Swift programming language, an application has been developed that allows users to view and interactively manipulate three-dimensional models derived from DICOM files in real time. The process of medical data processing involves converting DICOM files into RAW format, enabling faster and more efficient processing of volumetric data within the application. The application offers intuitive user interaction with models, including rotation, scaling, and viewing internal structures, enhancing the user experience in both educational and clinical settings. The thesis also provides an overview of the key technologies and algorithms used in the development process, such as Ray Marching for rendering volumetric data. In addition to the technical aspects of development, the thesis focuses on the application of augmented reality in medicine, highlighting its usefulness in education and diagnostics, and providing insight into the challenges and potential directions for further application development. The application represents a significant contribution to the field of medical visualization, allowing doctors and medical students to approach the analysis of medical data in an innovative way.

Keywords: ARKit, Augmented reality, computed tomography, DICOM format, interactive medical visualization, Metal API, volumetric data visualization

Životopis

Adrian Župarić rođen je u Vinkovcima 11. veljače 2000. godine. Nakon završene osnovne škole Ivana Gorana Kovačića u Vinkovcima, upisuje Tehničku školu Ruđera Boškovića u Vinkovcima, smjer tehnička gimnazija. Po završetku srednje škole 2018. godine upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Preddiplomski studij završava 2022. godine, nakon čega iste godine upisuje diplomski studij računarstva na istom fakultetu. Od rujna 2021. godine zaposlen je u tvrtki Barrage na poziciji iOS developera.