

Izgradnja mobilnih aplikacija interaktivne proširene stvarnosti za platformu iOS

Markotić, Antonio

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:168:045899>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-15**



Repository / Repozitorij:

[FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repository](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 349

**IZGRADNJA MOBILNIH APLIKACIJA INTERAKTIVNE
PROŠIRENE STVARNOSTI ZA PLATFORMU IOS**

Antonio Markotić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 349

**IZGRADNJA MOBILNIH APLIKACIJA INTERAKTIVNE
PROŠIRENE STVARNOSTI ZA PLATFORMU IOS**

Antonio Markotić

Zagreb, lipanj 2024.

DIPLOMSKI ZADATAK br. 349

Pristupnik: **Antonio Markotić (0036526466)**

Studij: Računarstvo

Profil: Znanost o mrežama

Mentor: izv. prof. dr. sc. Igor Čavrak

Zadatak: **Izgradnja mobilnih aplikacija interaktivne proširene stvarnosti za platformu iOS**

Opis zadatka:

Korištenje proširene stvarnosti (engl. Augmented Reality, AR) postaje sve uobičajenije u suvremenom svijetu, osobito zbog napretka u mobilnim tehnologijama, osjetilnim sustavima poput LiDARa, te široko rasprostranjenoj primjeni umjetne inteligencije. Jedno od brzorastućih područja primjene tehnologije AR u mobilnom okruženju je transformacija interakcije korisnika s njihovim fizičkim okruženjem, obogaćivanjem stvarnog okruženja korisnika interaktivnim virtualnim elementima. Ovaj diplomski rad fokusirat će se na istraživanje tehnologije potrebne za stvaranje proširene stvarnosti na mobilnim uređajima, s posebnim osvrtom na upotrebu LiDAR senzora. Rad će objasniti osnovne principe izgradnje AR aplikacija i ulogu LiDAR tehnologije u njihovom razvoju. Također, u okviru rada, razvijat će se aplikacija za iOS platformu kao praktični primjer primjene tih principa, koja će omogućiti korisnicima pozicioniranje virtualnih interaktivnih objekata u stvarnom prostoru. Aplikacija će se oslanjati na skeniranje prostora i stvaranje 3D modela prostora, dodavanje predefiniranih virtualnih objekata u stvarni prostor i interakciju s njima. Poseban naglasak bit će na analizi performansi aplikacije, uključujući testiranje preciznosti modela prostora, brzine odziva aplikacije i korisničkog iskustva.

Rok za predaju rada: 28. lipnja 2024.

*Zahvaljujem se mentoru izv.prof.dr.sc. Igoru Čavraku
na stručnim savjetima i strpljenju koje je uloženo u prijenos znanja
i u usmjerenje oko izvedbe ovog završnog rada.*

Sadržaj

Skraćenice.....	1
Uvod	2
1. Proširena stvarnost.....	3
1.1. Definicija i povijest	3
1.2. Vrste virtualnih okruženja	4
1.3. Primjena u različitim industrijama	6
2. Senzori za detekciju svjetla i dometa	8
2.1. Princip rada.....	8
2.2. Primjena u različitim industrijama	9
2.3. Integracija s Apple uređajima.....	11
2.3.1. Povijest	11
2.3.2. Tehničke karakteristike.....	12
2.3.3. Measure aplikacija – primjer	14
2.4. Prednosti i ograničenja	16
3. Razvoj aplikacija za operacijski sustav iOS	17
3.1. Uvod u iOS	17
3.2. Razvojno okruženje Xcode	18
3.3. Programski jezik Swift	18
4. Proširena stvarnost za operacijski sustav iOS	23
4.1. Uvod u ekosustav.....	23
4.1.1. SceneKit.....	23
4.1.2. ARKit.....	24
4.1.3. RealityKit.....	26
4.2. Ikea aplikacija - primjer.....	27
4.3. Ostale tehnologije i njihov utjecaj na proširenu stvarnost.....	29

4.3.1.	Umjetna inteligencija.....	29
4.3.2.	Strojno učenje	29
4.3.3.	Utjecaj na proširenu stvarnost	30
5.	Izrada mobilne aplikacije AR Scanner	31
5.1.	Kreiranje projekta	31
5.2.	Provjera podrške senzora za detekciju svjetla i dometa	32
5.3.	Zahtjev za pristup kameri	33
5.4.	Arhitektura Model Pogled Prezenter – Koordinator.....	34
5.5.	Početni zaslon	36
6.	Integracija senzora za detekciju svjetla i dometa u AR Scanner	39
6.1.	Skeniranje okoline i objekata	40
6.2.	Skeniranje prostorija koristeći RoomPlan	43
6.3.	Usporedba pristupa	47
7.	Dodavanje interaktivnih virtualnih objekata u AR Scanner	49
7.1.	Dodavanje virtualnog objekta.....	50
7.2.	Interakcija s virtualnim objektom.....	51
	Zaključak	52
	Literatura	53
	Sažetak.....	55
	Summary.....	56
	Privitak.....	57

Skraćenice

AR	<i>augmented reality</i>	proširena stvarnost
VR	<i>virtual reality</i>	virtualna stvarnost
MR	<i>mixed reality</i>	mješovita stvarnost
LiDAR	<i>light detection and ranging</i>	detekcija svjetla i dometa
iOS	<i>iPhone operation system</i>	operacijski sustav iPhone
AI	<i>artificial intelligence</i>	umjetna inteligencija
ML	<i>machine learning</i>	duboko učenje
LIFO	<i>last in first out</i>	zadnji unutra prvi van

Uvod

Proširena stvarnost (engl. augmented reality, AR) postaje sve prisutnija u svakodnevnom životu, zahvaljujući razvoju mobilnih tehnologija i sofisticiranim sustavima poput senzora za detekciju svjetla i dometa (engl. light detection and ranging, LiDAR). Tehnologija proširene stvarnosti omogućava obogaćivanje stvarnog svijeta virtualnim elementima, stvarajući iskustvo koje briše granice između stvarnog i virtualnog svijeta. Takva tehnologija mijenja način na koji komuniciramo, učimo, igramo se i obavljamo razne svakodnevne aktivnosti, te se koristi u širokom spektru industrija, uključujući obrazovanje, zabavu, zdravstvo, marketing i mnoge druge.

U kontekstu mobilnih uređaja, AR je posebno značajan zbog svoje dostupnosti i praktičnosti. Milijuni korisnika diljem svijeta svakodnevno koriste pametne telefone, što ga čini lako dostupnim velikom broju ljudi. Osim toga, napredak u tehnologijama poput umjetne inteligencije i senzora omogućava kreiranje aplikacija koje rješavaju postojeće probleme na dotad nezamislive načine. Jedan od ključnih napredaka u ovom području je uvođenje senzora za detekciju svjetla i dometa u mobilne uređaje, što omogućava nikad preciznije skeniranje i mapiranje prostora te stvaranje trodimenzionalnih modela stvarnog okruženja visoke kvalitete.

Ovaj diplomski rad fokusirat će se na istraživanje tehnologija potrebnih za razvoj mobilnih aplikacija proširene stvarnosti, s posebnim naglaskom na uređaje koji koriste operacijski sustav iPhone (engl. iPhone operation system, iOS). U okviru rada, analizirat će se osnovni principi razvoja aplikacija proširene stvarnosti, tehnologije koje omogućavaju njihovu realizaciju, te konkretne metode i alati koji se koriste. Ovaj rad će paralelno analizirati izazove na koje se nailazi prilikom korištenja navedenih tehnologija. U praktičnom dijelu rada, razvijat će se aplikacija proširene stvarnosti za iOS koja će omogućiti skeniranje i mapiranje prostora koristeći LiDAR senzore, te dodavanje predefiniranih virtualnih objekata u stvarno okruženje.

1. Proširena stvarnost

1.1. Definicija i povijest

Proširena stvarnost je tehnologija koja proširuje stvarni svijet virtualnim objektima, pružajući korisnicima interaktivna iskustva koja kombiniraju fizičko i digitalno okruženje. Ova tehnologija koristi kamere i senzore za detekciju stvarnog prostora, a zatim vrši kompleksne kalkulacije kako bi stvarni svijet dopunila računalno generiranim slikama, zvukovima i drugim virtualnim elementima. Jedna od definicija proširene stvarnosti glasi – Proširena stvarnost je tehnologija koja kombinira stvarne i virtualne elemente, omogućujući interakciju u stvarnom vremenu i stvarajući trodimenzionalne prikaze [1].

Povijest ove tehnologije seže nekoliko desetljeća unatrag. Prvi koncepti proširene stvarnosti pojavili su se 1960-ih godina, no tada još nisu bili poznati niti dostupni široj javnosti i bili su isključivo konceptualni. Damoklov mač (engl. Sword of Damocles) jedan je od prvih praktičnih sustava proširene stvarnosti. Razvio ga je Ivan Sutherland 1968. godine. Ovaj sustav koristio je kacigu s ekranom koja je prikazivala jednostavne virtualne oblike, stvarajući prvi korak ka integraciji virtualnih elemenata u stvarni svijet. smatra se pretečom moderne proširene stvarnosti. Zanimljivo je da je naziv dobio zbog toga što je cjelokupni sustav kacige i pratećih dijelova podsjećao na mač [2].

Desetljeće kasnije, 2003. godine, američka vojska integrirala je SmartCam3D sustav proširene stvarnosti u Shadow bespilotni zračni sustav kako bi pomogla operaterima u centrali pri lociranju osoba ili točaka interesa [3]. Ovaj sustav koristio je fiksne geografske informacije, kao što su nazivi ulica, zračne luke, bitne građevine i iz tih informacija kreirao virtualne elemente. Ti elementi proširivali su snimke snimane kamerom iz Shadow bespilotne letjelice koji su prenošene uživo operaterima. Glavni problem koji je sustav rješavao bio je problem uskog vidnog polja. Koristeći snimku iz letjelice, sustav je u stvarnom vremenu, na osnovu informacija koje je prethodno prikupio, generirao proširenje te snimke u obliku virtualnog proširenja okvira. Sustav je također prikazivao markere lokacija suradnika, neprijatelja i neutralnih subjekata u stvarnom vremenu, čime je operaterima znatno olakšavao posao. SmartCam3D jedan je od prvih sustava koji je koristeći tehnologiju proširene stvarnosti ostvario nešto što je u to vrijeme smatrano neostvarivim.

1.2. Vrste virtualnih okruženja

Proširena stvarnost pojam je koji se često miješa s pojmovima virtualna stvarnost (engl. virtual reality, VR) i pojmom mješovita stvarnost (engl. mixed reality, MR), no ove tehnologije imaju različite karakteristike i primjene. Prije detaljnije analize navedenih pojmova, bitno je spomenuti i proširenu virtualnost.

Proširena virtualnost (engl. augmented virtuality, AV) nešto je rjeđe korištena tehnologija od ostalih spomenutih. To je tehnologija koja je obrnuta od proširene stvarnosti, tj. dodaje u virtualna okruženja scene i objekte iz stvarnog okruženja. Jedan od primjera su studenti medicine koji koriste virtualne simulatore koji integriraju stvarne snimke pacijenata (npr. CT snimka) u virtualne modele tijela. To omogućava studentima da vježbaju složene kirurške zahvate u kontroliranom virtualnom okruženju, koristeći stvarne podatke iz medicinske prakse. Proširena virtualnost pomaže u stvaranju realističnijih i edukativno bogatijih simulacija, čime se poboljšava kvaliteta medicinske edukacije i priprema studenata za rad u stvarnim uvjetima.

Virtualna stvarnost potpuno je računalno generirano okruženje koje korisnika u potpunosti izolira od stvarnog svijeta. Korisnici virtualne stvarnosti obično koriste specijalizirane uređaje poput VR naočala, koje im omogućuju da se kreću i komuniciraju unutar virtualnog svijeta. Takvi sustavi najčešće imaju podršku senzora koji registriraju korisnikove fizičke pokrete i preslikavaju ih u virtualnu stvarnost. Industrija igara prednjači u doprinosu razvoja virtualne stvarnosti. Prve komercijalne igre virtualne stvarnosti nastale su 1980-ih i imale su fiksne zaslone koji su bili direktno spojeni na zaslon računala. Nakon velikog interesa javnosti, razvijeni su znatno napredniji sustavi koji su omogućili puno bolja iskustva za VR igre, primjerice naočale PlayStation VR. Još jedan od takvih uređaja je Kinect – uređaj koji funkcionira kao sklopovlje kamera i senzora koji proširuje funkcionalnosti Xbox konzole i omogućava igranje VR igara na njoj. Kinect treba postaviti tako da je korisnik sustava ispred njega i tada je moguće registrirati korisnikove pokrete i preslikati ih u virtualnu stvarnost u kojoj se odvija igra. Tako, svaki korisnikov pokret u stvarnom svijetu se preslikava u virtualni svijet i postaje sudionik tog virtualnog svijeta. Zanimljivost je da taj sustav može prepoznati lice korisnika i automatski ga ulogirati u njegov prethodno kreiran profil i time mu poboljšati korisničko iskustvo [4].

Mješovita stvarnost kombinira elemente proširene i virtualne stvarnosti i omogućuje interakciju između stvarnog i virtualnog svijeta u stvarnom vremenu. Ova tehnologija koristi kombinaciju senzora i kamera kako bi omogućila nove funkcionalnosti virtualnih elemenata. Mješovita stvarnost nadilazi granice proširene stvarnosti, dopuštajući virtualnim objektima da integriraju sa stvarnim svijetom u stvarnom vremenu, stvarajući novi oblik hibridnog okruženja [1]. Slično kao i sa sustavima virtualne stvarnosti, i mješovita stvarnost je pronašla svoju primjenu u industriji igara. Primjer jedne interakcije u igri mješovite stvarnosti bi mogao biti da je korisniku kroz VR naočale prikazana virtualna stvarnost u kojoj je on šef kuhinje. U toj virtualnoj stvarnosti korisnik vidi nož, lonac i tavu prikazane kao virtualne objekte. U običnom sustavu virtualne stvarnosti bi korisnik mogao imati interakciju s navedenim objektima bez da ti objekti postoje u stvarnosti, ali u sustavu mješovite stvarnosti korisnik će morati imati pripremljen svoj fizički stvarni nož, lonac i tavu i staviti ih na stol. Ti stvarni objekti će biti prepoznati koristeći napredne senzore, kamere i algoritme prepoznavanja te će korisnik morati fizički uzeti stvarni nož u desnu ruku kako bi mu virtualni nož u mješovitoj stvarnosti bio prikazan u virtualnoj desnoj ruci. Tako, mješovita stvarnost pruža znatno „stvarniji“ osjećaj interakcije u virtualnoj stvarnosti zato što uključuje i čulo dodira.

Kako se navedeni sustavi pozicioniraju u usporedbi s drugima prikazano je na Slika 1.1. Iz ilustracije je zanimljivo vidjeti koliki dio sustava pokriva mješovita stvarnost, što govori i o njezinom ogromnom potencijalu. U istraživanju iz 2020. godine, koje je proveo Microsoft, utvrđeno je da 68% tvrtki smatra da mješovita stvarnost može biti ključan alat za postizanje strateških poslovnih ciljeva. Nastavno, u navedenom istraživanju je oko 90% tvrtki potvrdilo da već eksperimentiraju s mješovitom stvarnošću [5].



Slika 1.1 Spektar od stvarnosti do potpune virtualnosti (preuzeto iz [5])

1.3. Primjena u različitim industrijama

Zbog brojnih mogućnosti koje nudi, proširena stvarnost ima široku primjenu u različitim industrijama, od kojih je neke u potpunosti promijenila. Neke od tih industrija su:

- **Obrazovanje:** Omogućava interaktivno učenje kroz vizualizacije koje olakšavaju razumijevanje kompleksnih pojmova i procesa. Na primjer, studenti mogu koristiti sustave proširene stvarnosti u obliku mobilnih aplikacija koji stvarne kartice za ponavljanje proširuju s virtualnim objektima kada se te kartice snime kamerom.
- **Zdravstvo:** U medicini se koristi za pomoć liječnicima tijekom operacija, omogućujući im da vide unutarnje strukture pacijenata bez potrebe za invazivnim postupcima. Također pomaže u obuci medicinskog osoblja putem simulacija.
- **Igre:** Našla je široku primjenu u industriji igara i zabave, pružajući igračima jedinstvena iskustva koja kombiniraju stvarni svijet s virtualnim elementima. Popularne igre poput Pokémon Go koriste ovu tehnologiju za stvaranje interaktivnih iskustava.
- **Marketing i maloprodaja:** Tvrtke koriste proširenu stvarnost za stvaranje interaktivnih marketinških kampanja i poboljšanje korisničkog iskustva. Na primjer, kupci mogu koristiti aplikacije kako bi isprobali proizvode poput namještaja ili odjeće prije kupnje.
- **Graditeljstvo i arhitektura:** AR pomaže arhitektima i građevinskim radnicima kroz vizualizaciju projekata u stvarnom okruženju. Na primjer, arhitekti mogu koristiti AR za prikaz budućih zgrada na gradilištu, dok građevinski radnici mogu dobiti upute za postavljanje strukturalnih elemenata u stvarnom vremenu.
- **Industrijska primjena:** Pomaže radnicima u industrijskim okruženjima kroz prikazivanje uputa i podataka u stvarnom vremenu što smanjuje mogućnost pogrešaka tijekom složenih operacija. Primjer takvog sustava bi bila proširena stvarnost za radnike koji razvrstavaju ispravne od oštećenih proizvoda, gdje im proširena stvarnost dodatno pomaže prepoznati oštećenja koja ljudsko oko ne primjećuje.

- **Turizam:** Proširena stvarnost omogućuje turistima interaktivna iskustva tijekom posjeta povijesnim lokalitetima ili muzejima. Na primjer, posjetitelji mogu koristiti AR aplikacije za prikaz rekonstrukcija drevnih građevina ili za dodatne informacije o izlošcima u muzejima.
- **Automobilska industrija:** Proizvođači automobila koriste proširenu stvarnost za dizajn i razvoj vozila. Na primjer, inženjeri mogu pregledavati virtualne modele automobila u stvarnom prostoru, a kupci mogu koristiti AR aplikacije za pregled unutrašnjosti automobila i personalizaciju prije kupnje.

Proširena stvarnost nastavlja se razvijati i pronalaziti nove primjene u različitim sektorima, mijenjajući način na koji se obavljaju svakodnevne aktivnosti. Sve učestalije ćemo susretati osobe s uređajima koji im omogućuju proširenje stvarnosti s virtualnim elementima kao na Slika 1.2. Iako je moguće koristiti sustave proširene stvarnosti koristeći jednostavne uređaje, primjerice običnu kameru mobilnog uređaja, napredniji sustavi koji zahtijevaju veću preciznost, koriste senzore za detekciju svjetla i dometa koji su predstavljeni u idućem poglavlju.

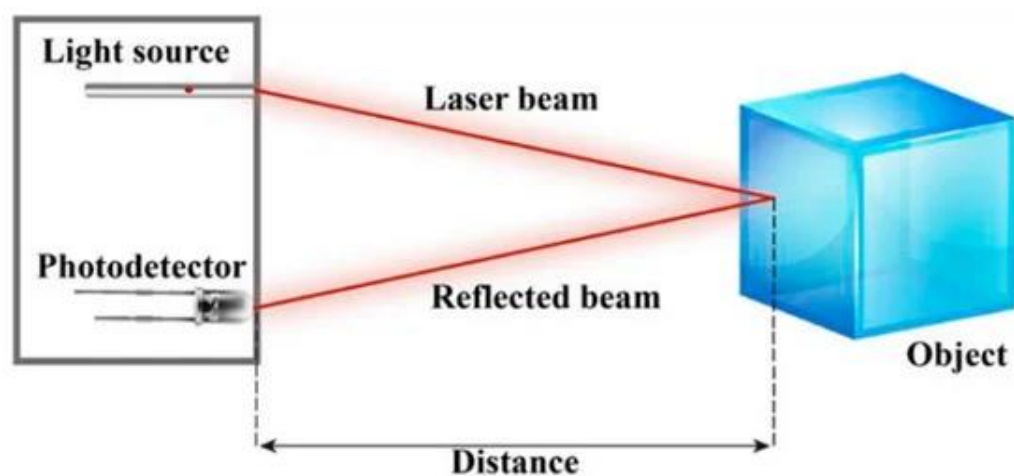


Slika 1.2 Ilustracija osobe s VR naočalama (preuzeto iz [6])

2. Senzori za detekciju svjetla i dometa

2.1. Princip rada

Senzori za detekciju svjetla i dometa, poznatiji kao LiDAR senzori, predstavljaju naprednu tehnologiju koja koristi pulsirajuće lasere za mjerenje udaljenosti do objekata. Takvi senzori emitiraju laserske zrake koje se odbijaju od površina i vraćaju do senzora, stvarajući precizne trodimenzionalne mape okruženja. Primjer odbijanja prikazan je na Slika 2.1



Slika 2.1 Odbijanje laserske zrake od objekta (preuzeto iz [7])

Koristeći takve sustave, moguće je dobiti razne informacije o okolini i objektima koji se u njoj nalaze. Neke od tih informacija i načini dobivanja su:

- brzina i smjer kretanja objekta koristeći princip Dopplerovog efekta,
- tekstura površine objekta slanjem brojnih signala na malen dio površine,
- razlike materijala objekta na osnovu prethodno definiranih tekstura površina,
- oblik i veličinu objekta preciznim slanjem zraka od ruba do ruba tog objekta,
- stanje atmosfere registrirajući odbijene zrake od sitnih čestica u atmosferi,
- podzemne strukture koristeći posebne penetrirajuće zrake i
- udaljenost objekta od izvora laserskih zraka.

Udaljenost objekta (d) računa se mjerenjem vremena koje je potrebno laserskoj zruci da se vrati do senzora nakon što se odbije od objekta prema Jednadžba 1. gdje je c konstanta brzine svjetlosti, a t vrijeme između izlaska zrake do njezina povratka.

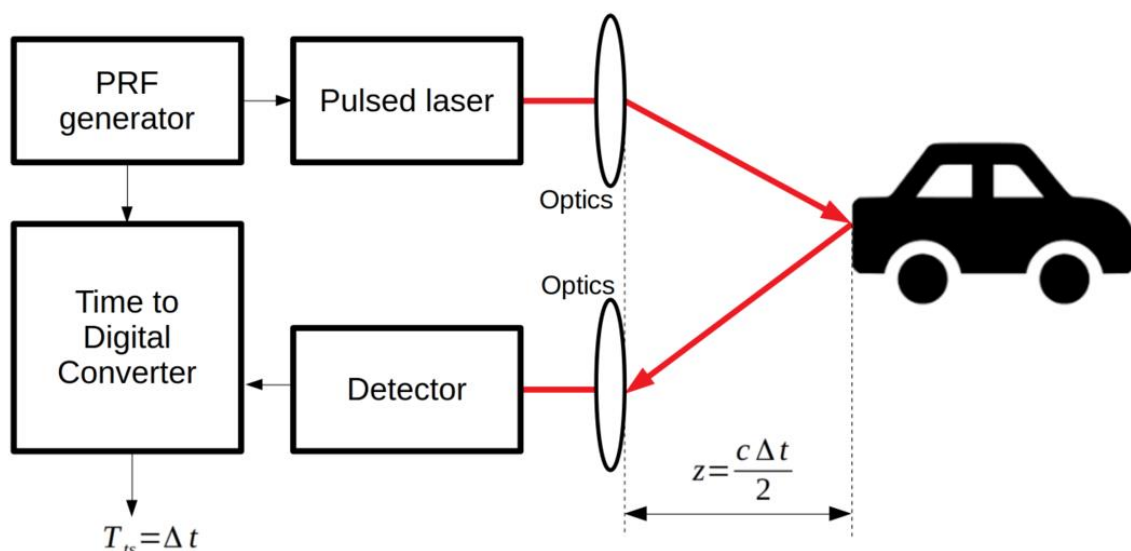
$$d = \frac{c * t}{2} \quad (1)$$

Naveden je opis kako funkcionira jedna zraka, ali sustavi za detekciju svjetla i dometa koriste senzore koji istovremeno prate odbijanje mnoštva laserskih zraka. Takvi senzori opremljeni su rotirajućim mehanizmom koji omogućava skeniranje velikih površina i stvaranje detaljnih 3D modela. Tehničke karakteristike uključuju visoku preciznost i točnost mjerenja, što omogućava detekciju objekata s izuzetno malim odstupanjima. Razlučivost senzora varira ovisno o modelu, ali većina komercijalnih LiDAR senzora ima sposobnost mjerenja udaljenosti s preciznošću do nekoliko centimetara. Osim toga, takvi senzori mogu raditi u različitim uvjetima osvjetljenja, uključujući noćne uvjete, što ih čini izuzetno korisnim za širok spektar primjena.

2.2. Primjena u različitim industrijama

- **Podmornice** su jedno od prvih područja u kojima su LiDAR senzori upotrijebljeni za računanje udaljenosti okolnih objekata, ostalih podmornica i potencijalnih prepreka. S obzirom na to da ljudsko oko ne može raspoznati okolne objekte u tami u kojoj se podmornice kreću, pojavila se potreba za drugačijim načinom orijentacije [8].
- U **šumarstvu**, LiDAR senzori koriste se za ispitivanje krošnji šuma i stvaranje preciznih 3D modela drveća. Ovi modeli pomažu u procjeni biomase, praćenju zdravlja šuma i planiranju održivog gospodarenja šumama. Na primjer, NASA koristi LiDAR tehnologiju za proučavanje učinaka suša i drugih klimatskih promjena na amazonske šume, što omogućava detaljno praćenje i analizu stanja šumskih ekosustava [9].
- U **arheologiji**, LiDAR senzori pomažu u otkrivanju i mapiranju arheoloških nalazišta koja su skrivena pod vegetacijom ili erodirana tijekom vremena. To omogućava arheolozima identificiranje struktura i artefakata bez potrebe za iskopavanjem, što čini istraživanje bržim i manje invazivnim. Na primjer, LiDAR je omogućio nove uvide u povijest i kulturu drevnih gradova u Srednjoj Americi gdje je korišten za skeniranje drevnih ruševina [10].

• **Automobilska industrija** jedan je od najznačajnijih primjera upotrebe senzora detekcije svjetla i dometa, posebno u razvoju autonomnih vozila. LiDAR senzori omogućuju autonomnim vozilima da skeniraju svoju okolinu i pretvore ju u 3D model, te uz pomoć umjetne inteligencije prepoznaju prepreke, pješake i druge sudionike u prometu te na temelju tih informacija donose odluke u stvarnom vremenu. Proizvođači autonomnih vozila poput firmi Tesla i Waymo koriste LiDAR senzore za unapređenje svojih sustava autonomne vožnje, čime značajno povećavaju sigurnost i pouzdanost svojih vozila [11]. Na Slika 2.2 prikazano je kako izgleda računanje udaljenosti do obližnjeg vozila. Uređaj za generiranje frekvencije pulsiranja laserskih zraka (engl. puls-repetition frequency generator, PRF) dojavljuje izvoru lasera kada da lansira lasersku zraku. Trenutak odašiljanja te zrake je registriran kada ona prođe kroz optičku barijeru. Tada ta zraka putuje sve dok ne naiđe na objekt u okolini i od njega se odbija. Ako se odbije i vrati kroz optičku barijeru, registrira se vrijeme njezina povratka i s pomoću njega i vremena odašiljanja računa koliko dugo je laserska zraka putovala. Odašiljanjem ogromnog broja laserskih zraka u kratkom vremenskom roku i prateći kut njihovog upada i još mnogo faktora, određuje se oblik površine predmeta od kojeg se odbijaju. Ako sustav prepozna da je površina od koje se zrake odbijaju stražnji dio nekog drugog automobila, informacije o vremenu putovanja zrake se zatim šalje u centralnu jedinicu koja računa udaljenost vozila od koje se zraka odbila.



Slika 2.2 Izračun udaljenosti drugog vozila koristeći LiDAR (preuzeto iz [12])

2.3. Integracija s Apple uređajima

2.3.1. Povijest

Apple je prvi put integrirao LiDAR senzor u svoje uređaje s iPad Pro modelom u ožujku 2020. godine. To je bio iPad Pro četvrte generacije, a LiDAR senzor je bio smješten u modul kamere, s kamerom i sensorima za prepoznavanje dubine. Nakon toga, u listopadu 2020., LiDAR senzor je ugrađen i u iPhone 12 Pro i iPhone 12 Pro Max (Slika 2.3), što je proširilo njegove mogućnosti na širu korisničku bazu.



Slika 2.3 LiDAR senzor na iPhone uređaju (preuzeto iz [13])

Apple je razvijao svoju verziju LiDAR senzora za precizne AR aplikacije i poboljšanje performansi kamera. Tim za razvoj hardvera iOS uređaja radio je integraciji kako bi osigurao da je senzor u mogućnosti pružiti brza i točna mjerenja udaljenosti, što je ključno za AR aplikacije, poboljšanje fotografske funkcije i mjerenje prostora. Do sada su izašle 4 generacije LiDAR senzora u iOS uređajima:

1. generacija:

- **Uređaji:** iPad Pro (2020.), iPhone 12 Pro, iPhone 12 Pro Max
- **Glavne značajke:** Osnovno mapiranje prostora, poboljšanje autofokusa u uvjetima slabog osvjetljenja.

2. generacija:

- **Uređaji:** iPhone 13 Pro, iPhone 13 Pro Max, iPad Pro (2021.)
- **Glavne značajke:** Poboljšana točnost i brzina, bolja integracija s ARKit-om, unaprijeđeno prepoznavanje objekata i prostora.

3. generacija:

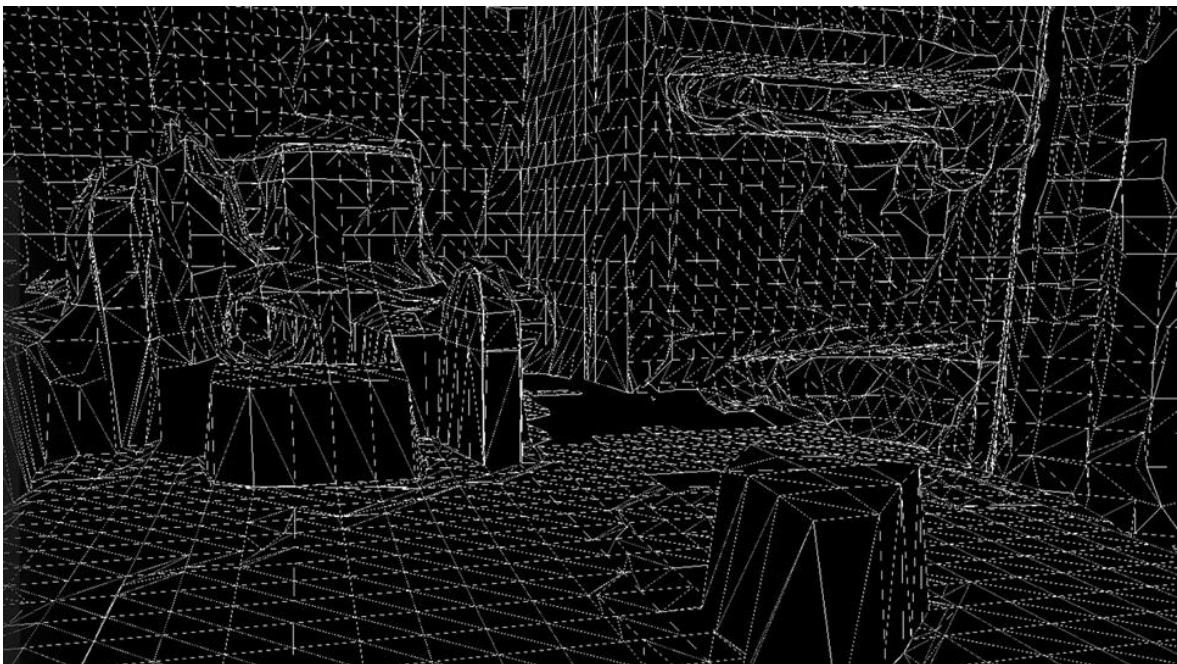
- **Uređaji:** iPhone 14 Pro, iPhone 14 Pro Max, iPad Pro (2022.)
- **Glavne značajke:** Dodatno smanjenje potrošnje energije, još veća preciznost i brzina skeniranja, napredne mogućnosti za AR i fotografske aplikacije.

4. generacija:

- **Uređaji:** iPhone 15 Pro, iPhone 15 Pro Max (2023.)
- **Glavne značajke:** Povećana preciznost i domet, bolji automatski fokus u uvjetima slabog osvjetljenja i preciznije mjerenje dubine za portretni način rada što omogućavajući detaljnije i profesionalnije fotografije.

2.3.2. Tehničke karakteristike

LiDAR senzori na iOS uređajima koriste laserske zrake za mjerenje udaljenosti do objekata. Ovi senzori omogućuju stvaranje preciznih 3D mapa okoline na mobilnim uređajima. Na Slika 2.4 prikazano je kako izgleda takva 3D mapa.



Slika 2.4 Mapa okruženja skeniranog uređajem koji koristi LiDAR senzor (preuzeto iz [14])

Ključne tehničke karakteristike LiDAR senzora u Apple uređajima su:

- **Raspon udaljenosti** - mogu precizno mjeriti udaljenosti do pet metara, što je pogodno za detaljno mapiranje unutarnjih prostora. Ovaj raspon omogućuje korisnicima da lako mapiraju sobe, hodnike i druge unutarnje prostore s visokom preciznošću, ali zbog ograničenja od 5 metara, skeniranja većih prostora s mobilnim uređajem ipak nisu moguća.
- **Točnost** - pružaju visoku razinu točnosti, s mogućnošću detekcije objekata s milimetarskom preciznošću. Ova razina preciznosti ključna je za aplikacije koje zahtijevaju detaljna mjerenja, kao što su arhitektonske i građevinske inspekcije, unutarnji dizajn, te razne industrijske primjene.
- **Brzina skeniranja** - mogu brzo skenirati okolinu, što omogućava stvaranje 3D modela u stvarnom vremenu. Brzina skeniranja važna je za aplikacije koje zahtijevaju trenutne povratne informacije, poput aplikacija proširene stvarnosti gdje je potrebno brzo prilagoditi virtualne objekte stvarnom okruženju.
- **Integracija s ARKit** – omogućava napredne funkcionalnosti poput poboljšanog prepoznavanja površina i točnijeg pozicioniranja virtualnih objekata. ARKit koristi podatke iz LiDAR senzora za stabilnije i realističnije postavljanje virtualnih objekata u stvarnom prostoru.

Korištenje LiDAR senzora na iOS uređajima omogućuje razvoj aplikacija koje mijenjaju način na koji obavljamo svakodnevne zadatke. Na primjer, aplikacije za unutarnje dizajniranje omogućuju korisnicima skeniranje svojih prostorija i precizno postavljanje virtualnog namještaja, što olakšava proces donošenja odluka pri kupnji namještaja ili preuređenju prostora. Uz to, LiDAR senzori omogućuju napredne fotografske funkcionalnosti. Na primjer, precizno mjerenje dubine omogućuje profesionalni portretni način rada s boljim efektima zamućenja pozadine, čak i u uvjetima slabog osvjetljenja. Ova tehnologija također poboljšava fokus kamere, čineći ga bržim i preciznijim.

Jedan od konkretnih primjera aplikacije za uređaje koji koriste iOS u kojima je uvođenje LiDAR tehnologije znatno poboljšalo funkcionalnost je Measure aplikacija koja je automatski instalirana na uređaje s iOS operacijskim sustavom.

2.3.3. Measure aplikacija – primjer

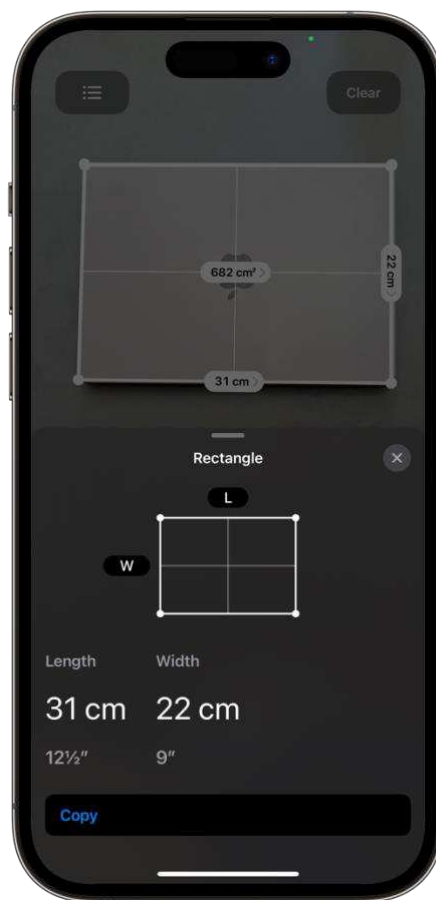
Measure aplikacija je odavno dostupna za uređaje s operacijskim sustavom iOS. Ona omogućuje mjerenja stvarnih udaljenosti i površina koristeći samo mobilni uređaj. Prije podrške LiDAR tehnologije, ta aplikacija se zasnivala na funkcionalnosti kamere i alata proširene stvarnosti. Od uvođenja senzora za detekciju svjetla i dometa, aplikacija Measure je doživjela procvat. Mjerenja napravljena s njome, na uređajima koji imaju LiDAR podršku su centimetarske točnosti.

Upotreba aplikacije je jednostavna, ako se želi izmjeriti udaljenost između točke A i točke B, jednostavno je potrebno obje točke obilježiti koristeći glavni gumb aplikacije na dnu ekrana u sredini sa znakom plus. Provedeno mjerenje prikazano je na Slika 2.5. Druga funkcionalnost koju ova aplikacija nudi jest mogućnost automatskog prepoznavanja oblika objekata, prepoznavanja njegovih rubnih točki i računanje njegove površine. Na

Slika 2.6 prikazano je kako ta funkcionalnost izgleda u primjeni.



Slika 2.5 Mjerenje udaljenosti



Slika 2.6 Mjerenje površine

Nakon toga provedeno je mjerenje fizičkim metrom, što je prikazano na Slika 2.7. Mjerenjem je utvrđena širina od 31,2 cm, dok aplikacija Measure prikazuje širinu prijenosnog računala kao 31 centimetar. To potvrđuje visoku preciznost rezultata aplikacije. Dimenzije širine i visine prijenosnog računala također se poklapaju sa službenim specifikacijama navedenim na stranici proizvođača, što je prikazano na slici Slika 2.8



Slika 2.7 Ručno mjerenje duljine pokazuje 31.2cm

Height: 0.61 inch (1.55 cm)

Width: 12.31 inches (31.26 cm)

Slika 2.8 Službene dimenzije računala (preuzeto iz [15])

2.4. Prednosti i ograničenja

Prednosti LiDAR senzora su visoka preciznost i točnost mjerenja. LiDAR senzori mogu detektirati objekte s izuzetno malim odstupanjima, što omogućava detaljno mapiranje okruženja i preciznu navigaciju autonomnih sustava. Treba imati na umu da ova tehnologija ne zahtijeva da teren koji skenira bude osvjetljen, s obzirom na to da laserske zrake koje LiDAR odašilja ponašaju se jednako i po danu i po noći. Zbog toga može raditi u različitim uvjetima osvjetljenja, uključujući potpunu tamu, što je izuzetno korisno za aplikacije koje zahtijevaju kontinuirano praćenje i detekciju. Druga značajna prednost je sposobnost izrazito brzog skeniranja velikih površina i generiranja trodimenzionalnih modela u stvarnom vremenu. To omogućava brže donošenje odluka i učinkovitije planiranje u industrijama poput građevinarstva i autonomnih vozila. Još jedna prednost jest da već postoje LiDAR sustavi koji mogu u određenim uvjetima raditi i podvodna skeniranja [16].

S druge strane, jedno od glavnih **ograničenja** je visoka cijena senzora i složenost integracije u postojeće sustave. Troškovi razvoja i implementacije LiDAR sustava mogu biti značajni, što može predstavljati prepreku za manje tvrtke ili projekte s ograničenim budžetom. Drugo ograničenje je osjetljivost na vremenske uvjete. Iako LiDAR senzori mogu raditi u različitim uvjetima osvjetljenja, njihova učinkovitost može biti smanjena u prisutnosti gustih oblaka prašine, magle ili kiše. Takvi uvjeti mogu ometati laserske zrake i smanjiti preciznost mjerenja zato što se zrake mogu odbijati od čestica u atmosferi i time smanjivati vjerodostojnosti mjerenja.

Unatoč ovim ograničenjima, tehnologija se nastavlja razvijati, unapređivati i pružati sve bolje performanse i šire mogućnosti primjene. Sa sve većom prisutnošću u različitim industrijama, LiDAR senzori postaju ključan alat za poboljšanje učinkovitosti, sigurnosti i preciznosti u mnogim aspektima našeg svakodnevnog života. Od posebne važnosti za ovaj rad je činjenica da su LiDAR senzori u posljednjih nekoliko godina postali dostupni široj javnosti u sklopu kamera pametnih uređaja iPhone. Time je otvorena mogućnost za razvoj rješenja brojnih problema i distribucija njih krajnjim korisnicima u obliku mobilnih aplikacija. Kako izgleda izrada mobilne aplikacije za uređaj iPhone i koji su izazovi implementacije proširene stvarnosti i korištenja senzora za detekciju svjetla i dometa bit će objašnjeno u narednim poglavljima.

3. Razvoj aplikacija za operacijski sustav iOS

3.1. Uvod u iOS

Razvoj aplikacija za operacijski sustav iPhone (iOS) predstavlja izazovan proces koji uključuje korištenje specifičnih alata, programskih jezika i razvojnih okruženja. iOS je Appleov operacijski sustav koji pokreće uređaje kao što su iPhone, iPad i iPod Touch. Njegova popularnost i široka primjena čine ga izuzetno privlačnim za razvijatelje diljem svijeta. U ovom poglavlju, detaljno ćemo obraditi ključne aspekte razvoja iOS aplikacija, uključujući uvod u iOS, razvojno okruženje Xcode i programski jezik Swift.

Operacijski sustav iOS predstavlja jedan je od najpopularnijih mobilnih operacijskih sustava na svijetu. Apple ga je lansirao 2007. godine s prvim iPhone uređajem i tada je postavio novi standard u pogledu dizajna, funkcionalnosti i sigurnosti mobilnih uređaja. Od tada, svake godine bez iznimke izlazi nova verzija iOS-a kao što je prikazano u Tablica 3.1 i Tablica 3.2. Zahvaljujući redovitim ažuriranjima, iOS je postao platforma koja nudi širok spektar mogućnosti za razvoj aplikacija, uključujući napredne značajke kao što su proširena stvarnost, umjetna inteligencija (engl. artificial intelligence, AI) i strojno učenje (engl. machine learning, ML).

Poznat je po svom intuitivnom korisničkom sučelju, visokom stupnju sigurnosti i stabilnosti te bogatom ekosustavu aplikacija dostupnih kroz App Store. Ova platforma omogućuje razvoj aplikacija koje mogu koristiti sve prednosti fizičkog sklopovlja Apple uređaja.

Tablica 3.1 iOS verzije od 2007. - 2015.

2007	2008	2009	2010	2011	2012	2013	2014	2015
iOS 1	iOS 2	iOS 3	iOS 4	iOS 5	iOS 6	iOS 7	iOS 8	iOS 9

Tablica 3.2 iOS verzije od 2016. - 2024.

2016	2017	2018	2019	2020	2021	2022	2023	2024
iOS 10	iOS 11	iOS 12	iOS 13	iOS 14	iOS 15	iOS 16	iOS 17	iOS 18

3.2. Razvojno okruženje Xcode

Xcode je integrirano razvojno okruženje (engl. integrated development environment, IDE) koje je Apple razvio za kreiranje aplikacija za sve svoje platforme, uključujući iOS, macOS, watchOS i tvOS. Xcode pruža sve alate potrebne za razvoj, testiranje i distribuciju aplikacija, čineći ga neizostavnim alatom za sve iOS razvijatelje. Neke od glavnih značajki koje Xcode nudi prikazane su u Tablica 3.3.

Tablica 3.3 Xcode alati

Uređivač koda	Napredni uređivač koda s podrškom za sintaksu i formatiranje, automatsko dovršavanje koda, te integracijom s alatima za upravljanje verzijama poput Gita.
Graditelj sučelja	Vizualni alat za dizajn korisničkog sučelja koji omogućuje kreiranje složenih prikaza bez potrebe za pisanjem koda.
Simulator	Alat za testiranje aplikacija na različitim modelima iOS uređaja, omogućujući razvijateljima da provjere funkcionalnost i izgled aplikacija na različitim zaslonima i rezolucijama.
Instrumenti	Alat za analizu performansi aplikacija, koji pomaže u optimizaciji koda i identifikaciji potencijalnih problema s memorijom, procesorom i drugim resursima.
TestFlight	Platforma za beta testiranje aplikacija, koja omogućuje developerima da distribuiraju svoje aplikacije testerima i prikupljaju povratne informacije prije konačnog objavljivanja.

3.3. Programski jezik Swift

Za razumijevanje razvoja mobilnih aplikacija za platformu iOS, neophodno je poznavanje programskog jezika Swift. To je moderan programski jezik kojeg je Apple predstavio 2014. godine, s ciljem da zamijeni Objective-C kao glavni jezik za razvoj aplikacija za Apple platforme. Swift je dizajniran da bude brz, siguran i jednostavan za upotrebu, s naglaskom na performanse i razvojnu produktivnost. Razvoj aplikacija za iOS uz korištenje Xcode-a i Swift-a omogućuje razvijateljima da iskoriste sve prednosti modernih tehnologija, pružajući korisnicima aplikacije visoke kvalitete s naprednim funkcionalnostima.

Neke od glavnih značajki Swifta su:

- **Klase i strukture (engl. classes and structures)**

Klase i strukture su osnovni građevni elementi u Swiftu koji omogućuju definiranje vlastitih tipova podataka. Glavna razlika između njih je u tome što su klase referentni tipovi, dok su strukture vrijednosni tipovi. To znači da se klase prosljeđuju po referenci, a strukture po vrijednosti. Primjer klase i strukture prikazan je na Slika 3.1.

```
class Osoba {
    var ime: String
    var prezime: String

    init(ime: String, prezime: String) {
        self.ime = ime
        self.prezime = prezime
    }
}

struct Tocka {
    var x: Int
    var y: Int
}
```

Slika 3.1 Klase i strukture

- **Slučajevi (engl. enumerations)**

Slučajevi omogućuju vezanje nekoliko specifičnih slučajeva za jedan objekt. Primjerice, objektu primljenom s vanjskog servisa mogu se pridijeliti slučajevi uspjeh, neuspjeh, učitavanje. To su tri slučaja u kojima se taj objekt može naći. Mogu imati pridružene vrijednosti koje omogućuju spremanje dodatnih informacija uz svaki slučaj. Primjer slučajeva dana u tjednu prikazan je na Slika 3.2.

```
enum DanUTjednu {
    case ponedjeljak, utorak, srijeda, cetvrtak, petak, subota, nedjelja
}

let danasnjiDan = DanUTjednu.ponedjeljak
```

Slika 3.2 Slučajevi

- **Opcionalni tipovi (engl. optional types)**

Opcionalni tipovi omogućuju varijablama da imaju vrijednost ili da ih nema i budu obilježeni kao nil. Ovo je korisno kada varijabla može biti prazna, a razvijatelj jasno mora naznačiti da se to može dogoditi. Primjer opcionalnog tipa prikazan je na Slika 3.3.



```
var ime: String? = "Ana"
ime = nil
```

Slika 3.3 Opcionalni tip

- **Precizno hvatanje grešaka (engl. precise error handling)**

Swift pruža napredne mogućnosti za rukovanje pogreškama pomoću **do**, **try** i **catch** blokova. Razvijatelji mogu definirati vlastite tipove pogrešaka i precizno hvatati i rukovati različitim pogreškama. Primjer hvatanja greške prikazan je na Slika 3.4.



```
enum DatotekaGreska: Error {
    case nemaDatoteke
    case neispravanFormat
}

func ucitajDatoteku(ime: String) throws -> String {
    guard ime == "postoji.txt" else {
        throw DatotekaGreska.nemaDatoteke
    }
    return "Sadržaj datoteke"
}

do {
    let sadrzaj = try ucitajDatoteku(ime: "ne postoji.txt")
    print(sadrzaj)
} catch {
    print("Greska pri ucitavanju datoteke: \(error)")
}
```

Slika 3.4 Hvatanje greške

- **Protokoli (engl. protocols)**

Protokoli definiraju skup metoda i svojstava koje klasa ili struktura mora implementirati. Oni omogućuju uspostavljanje zajedničkog skupa funkcionalnosti između različitih tipova. Primjer protokola prikazan je na Slika 3.5.

```
protocol Vozilo {
    var brzina: Int { get }

    func vozi()
}

class Automobil: Vozilo {
    var brzina: Int = 100

    func vozi() {
        print("Automobil vozi brzinom \{(brzina) km/h")
    }
}
```

Slika 3.5 Protokol

- **Proširenja (engl. extensions)**

Proširenja omogućuju dodavanje nove funkcionalnosti postojećim tipovima, bez potrebe za mijenjanjem izvornog koda tih tipova. Proširenja omogućuju grupiranje koda po zasebnim funkcionalnostima što olakšava čitljivost. Primer proširenja prikazan je na Slika 3.6

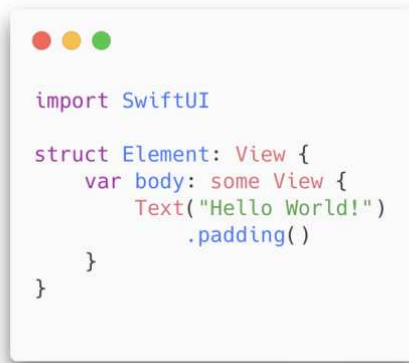
```
extension String {
    func obrnut() -> String {
        return String(self.reversed())
    }
}

let ime = "Swift"
print(ime.obrnut()) // tfiwS
```

Slika 3.6 Proširenje

- **SwiftUI**

SwiftUI je biblioteka za izradu korisničkog sučelja koja omogućuje deklarativni pristup dizajniranju aplikacija. Pomoću SwiftUI-ja, razvijatelji definiraju korisničko sučelje kao funkciju trenutnog stanja aplikacije, što omogućuje jednostavnije i intuitivnije građenje aplikacije. Primjer izgradnje komponente koristeći SwiftUI prikazan je na Slika 3.7

A screenshot of a code editor window showing SwiftUI code. The code defines a struct named 'Element' that inherits from 'View'. It contains a variable 'body' of type 'some View' which contains a 'Text' view with the text 'Hello World!' and a padding modifier. The code is as follows:

```
import SwiftUI

struct Element: View {
    var body: some View {
        Text("Hello World!")
            .padding()
    }
}
```

Slika 3.7 SwiftUI

Swift je postao ključni alat za razvoj aplikacija na Apple platformama, nudeći moderne i efikasne značajke koje olakšavaju rad razvijateljima. Njegova fleksibilnost, sigurnost i jednostavnost korištenja čine ga idealnim izborom za gradnju aplikacija. Kroz razumijevanje osnovnih koncepata kao što su klase, strukture, protokoli, opcionalni tipovi i sl., moguće je početi graditi aplikacije za Apple uređaje.

Sljedeće poglavlje će se fokusirati na proširenu stvarnost na iOS platformi. Biti će moguće saznati nešto više o iOS ekosustavu proširene stvarnosti, proučiti ogledni primjer aplikacije Ikea, te istražiti ulogu i značaj LiDAR senzora u Apple uređajima. Također će biti razmotrene ostale relevantne tehnologije kao što su umjetna inteligencija i strojno učenje, koje igraju važnu ulogu u razvoju AR aplikacija.

4. Proširena stvarnost za operacijski sustav iOS

4.1. Uvod u ekosustav

iOS platforma nudi napredne alate i biblioteke za razvoj aplikacija proširene stvarnosti, od kojih su najznačajniji ARKit, SceneKit i RealityKit. Ove biblioteke omogućuju razvijateljima da koriste napredne značajke poput prepoznavanja površina, praćenja pokreta, modeliranja i iscrtavanja složenih 3D scena i integracije s drugim iOS tehnologijama. U nastavku će detaljno biti predstavljene sve spomenute biblioteke sa Slika 4.1.



Slika 4.1 Apple biblioteke s podrškom za proširenu stvarnost

4.1.1. SceneKit

SceneKit je biblioteka za 3D grafiku koji olakšava stvaranje složenih 3D scena i objekata [17]. Bitno je naglasiti kako je SceneKit izvorno nastao kao biblioteka isključivo za virtualnu stvarnost, ali danas se koristi u kombinaciji s ARKit-om, što omogućava iscrtavanje 3D modela u proširenoj stvarnosti. Glavne značajke SceneKit-a su navedene ispod.

- **Jednostavno stvaranje 3D sadržaja** pruža način za stvaranje i manipuliranje 3D objektima, koristeći pristup sličan tradicionalnim grafičkim bibliotekama što smanjuje vrijeme potrebno za upoznavanje s bibliotekom iskusnim razvijateljima koji koristeći SceneKit mogu definirati geometriju, materijale i svjetlosne efekte za svoje 3D modele.

- **Fizikalni simulacijski motori** omogućuju simulaciju stvarnih fizičkih interakcija između objekata. Ovo je korisno za stvaranje realističnih animacija i interaktivnih iskustava. Na primjer, objekti se mogu ponašati kao da imaju masu, trenje i elastičnost, stvarajući autentične reakcije na sile.
- **Integracija s ARKit-om** omogućava razvijateljima da koriste 3D modele i scene u svojim aplikacijama. Ovo uključuje mogućnost iscrtavanja složenih 3D objekata koji se točno uklapaju u stvarni prostor prepoznat od strane ARKit-a.
- **Animacija i interakcija** omogućuju napredne animacijske mogućnosti, uključujući ključne okvire i proceduralne animacije. Programeri mogu stvoriti interaktivne animacije koje reagiraju na korisnikove pokrete i akcije.

4.1.2. ARKit

ARKit je Appleova biblioteka za proširenu stvarnost koja omogućuje razvijateljima da stvore impresivne aplikacije za iOS uređaje [18]. Prvi put predstavljen 2017. godine, ARKit kontinuirano evoluirao, dodajući nove značajke i poboljšavajući performanse. Neke od ključnih značajki su:

- **Prepoznavanje površina i okoline** poput podova, stolova i zidova omogućuje točno pozicioniranje virtualnih objekata u stvarnom prostoru. Također može analizirati okolinu i prilagoditi osvjetljenje virtualnih objekata kako bi se bolje uklopili u stvarni svijet.
- **Praćenje pokreta i pozicije** koristeći podatke iz senzora i kamere uređaja kako bi se precizno pratio položaj i pokreti korisnika. To omogućuje stvaranje interaktivnih iskustava gdje se virtualni objekti mijenjaju ovisno o korisnikovom kretanju. ARKit može pratiti i složenije pokrete, poput gestikulacija ruku, za još bogatiju interakciju.
- **Praćenje i prepoznavanje lica** omogućuje stvaranje maski i filtera koji se dinamički prilagođavaju izrazima lica. Ovo je posebno popularno u aplikacijama za društvene mreže i zabavu.

- **Podrška za višekorisničke AR aplikacije** gdje više korisnika može vidjeti i interaktivno djelovati s istim virtualnim objektima u stvarnom prostoru. Ova funkcionalnost otvara nove mogućnosti za društvene igre i kolaborativne aplikacije, gdje korisnici mogu dijeliti i zajednički manipulirati objektima.
- **Integracija s LiDAR sensorima** u najnovijim verzijama ARKit-a, omogućava još preciznije mapiranje prostora i prepoznavanje objekata. Ovo značajno poboljšava kvalitetu AR iskustava na uređajima poput iPhone-a i iPad-a koji su opremljeni s navedenim sensorima.

Na Sliku 4.2 prikazan je primjer aplikacije koja koristi ARKit za pozicioniranje virtualnih objekata u stvarnost.

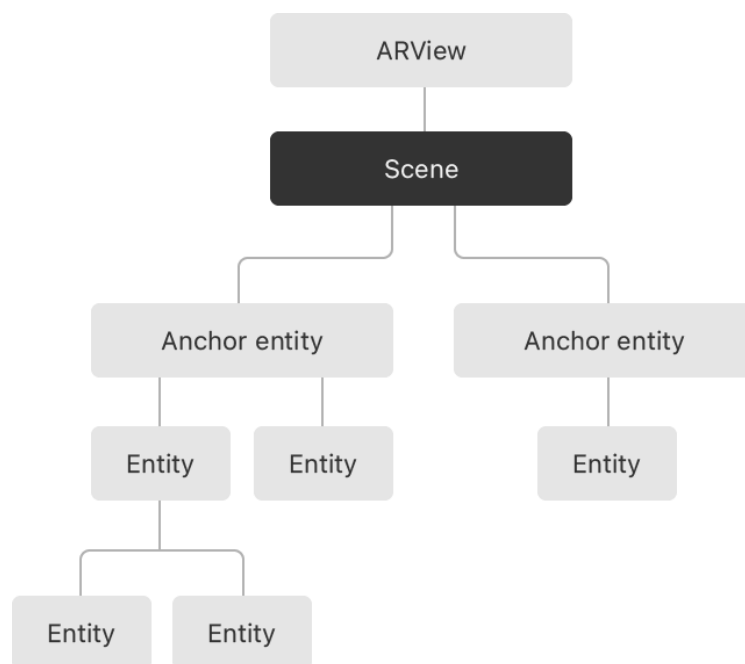


Slika 4.2 Dodavanje virtualnih objekata u stvarno okruženje u obliku igre

4.1.3. RealityKit

RealityKit je biblioteka za stvaranje AR iskustava koji kombinira mogućnosti ARKit-a i SceneKit-a, te dodaje nove alate za još bogatija AR iskustva [19]. Dizajniran je da pojednostavi razvoj AR aplikacija i uključuje mnoge unaprijed izgrađene komponente koje olakšavaju rad programerima. Sve navedene značajke SceneKit-a i RealityKit-a se odnose i na RealityKit u kojem su još naprednije, brže, efikasnije i jednostavnije za implementaciju.

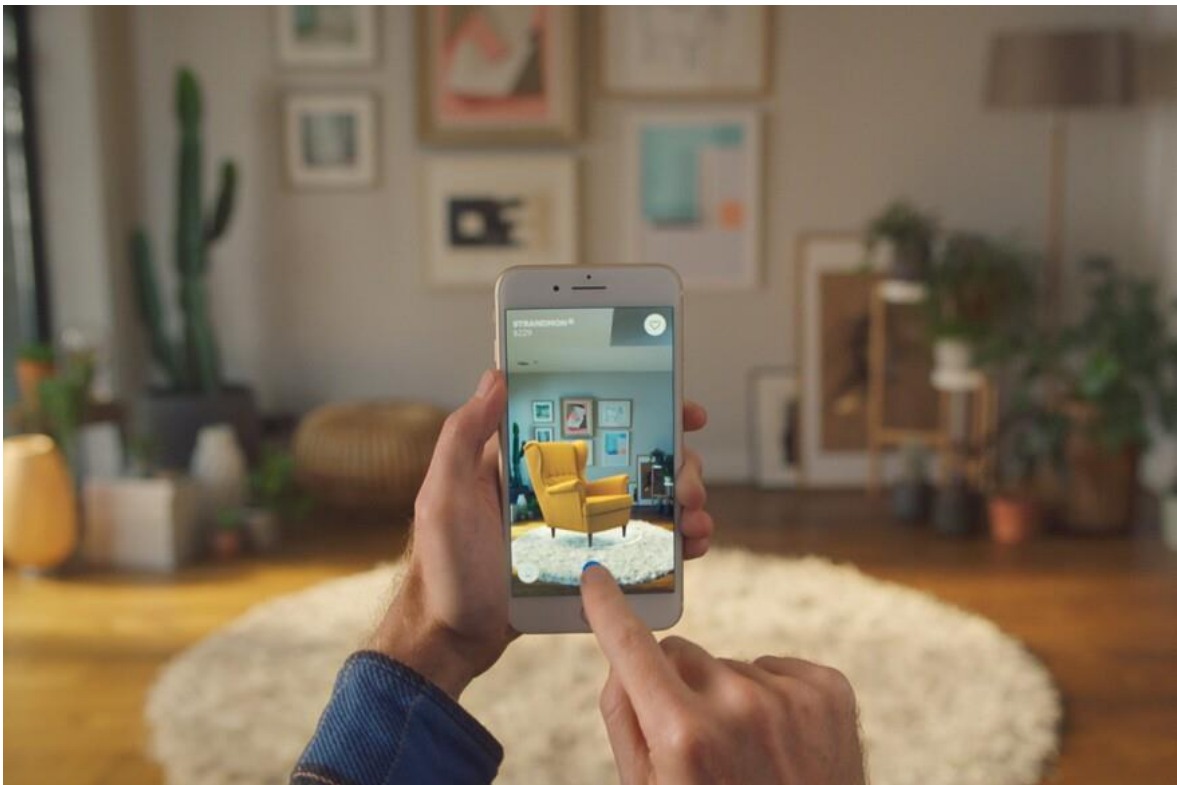
Glavne komponente RealityKita prikazane su na Slika 4.3. Prva komponenta je ARView, koji je glavna točka interakcije između korisnika i AR aplikacije. U ARView prikazuju se sve AR scene i objekti. Scena predstavlja skup svih entiteta koji se prikazuju u ARView, a može sadržavati jedan ili više sidrišnih entiteta (engl. anchor entity), koji su temeljni elementi za pozicioniranje drugih entiteta u prostoru. Entiteti su osnovni gradivni blokovi koji mogu predstavljati različite 3D objekte ili skupove objekata. Svaki entitet može sadržavati različite komponente poput geometrije, materijala, animacija, fizikalnih svojstava i slično. Entiteti se mogu hijerarhijski organizirati, pri čemu mogu sadržavati druge entitete kao podentitete. Bitno je naglasiti kako je ovo stablo strukture RealityKit komponenti, ali kako je u pozadini RealityKit-a ARKit, to znači da prikazano stablo vrijedi i za njegovu strukturu.



Slika 4.3 Stablo gradivnih elemenata RealityKit-a (preuzeto iz [19])

4.2. Ikea aplikacija - primjer

Jedan od najpoznatijih primjera korištenja AR tehnologije na iOS platformi je aplikacija Ikea. Ova aplikacija omogućuje korisnicima da virtualno postave namještaj iz Ikea kataloga u svoj dom koristeći kameru svog uređaja. Korisnici mogu mijenjati položaj, veličinu i vrstu namještaja, te tako dobiti realističan dojam kako će izgledati njihov prostor s odabranim proizvodima kao što je vidljivo na Slika 4.4.



Slika 4.4 Prikaz virtualne fotelje u stvarnom prostoru koristeći aplikaciju Ikea (preuzeto iz [20])

Ova aplikacija ujedno je i glavna motivacija za izradu ovog diplomskog rada i istraživanje svijeta proširene stvarnosti. Korisnici mogu birati iz širokog spektra Ikea proizvoda, vidjeti kako bi izgledali u njihovom domu i prilagoditi poziciju i kut gledanja. Ova tehnologija ne samo da pomaže korisnicima u donošenju odluka o kupnji, već i smanjuje vjerojatnost pogrešnih kupnji i vraćanja proizvoda.

Jedna od ključnih prednosti korištenja AR tehnologije u Ikea Place aplikaciji je mogućnost da točno procijeni veličinu namještaja u odnosu na stvarni prostor i time ispravno skalira virtualni objekt. ARKit koristi kamere i senzore iOS uređaja za mapiranje okoline, što

omogućava virtualnim objektima da izgledaju prirodno i realno u stvarnom svijetu. Koliko je aplikacija precizna, vidljivo je na Slika 4.5, gdje je moguće vidjeti kako izgleda stvarna fotelja u stvarnom prostoru koja je na prethodnoj slici predočena u gotovom savršenom omjeru i veličini koristeći ARKit.



Slika 4.5 Stvarna fotelja u stvarnom prostoru

Dodatno, aplikacija omogućuje korisnicima da dijele svoje kreacije s drugima putem društvenih mreža, čime se potiče interakcija i povećava angažman korisnika. Ovo predstavlja ne samo tehničko unapređenje, već i marketinški alat koji privlači nove korisnike i zadržava postojeće.

ARKit biblioteka postala je dostupna javnosti u lipnju 2017. godine, a Ikea je već u rujnu iste godine implementirala ovu tehnologiju u svoju aplikaciju. Ovaj brzi razvoj pokazuje kako je Ikea prepoznala vrijednost AR tehnologije i njen potencijal za poboljšanje korisničkog iskustva.

4.3. Ostale tehnologije i njihov utjecaj na proširenu stvarnost

Pored ARKit-a i LiDAR senzora, iOS platforma koristi i druge tehnologije koje značajno unapređuju AR aplikacije. Neke od tih tehnologija su umjetna inteligencija i strojno učenje. One igraju ključnu ulogu u prepoznavanju objekata, analizi okoline i interakciji s korisnicima. U nastavku ćemo detaljnije opisati kako se ove tehnologije koriste u kombinaciji s proširenom stvarnosti u aplikacijama za Apple uređaje.

4.3.1. Umjetna inteligencija

Umjetna inteligencija omogućava aplikacijama da analiziraju i razumiju korisnikovo okruženje na način koji je ranije bio nezamisliv. Korištenjem tehnologije umjetne inteligencije, aplikacije mogu prepoznati i interpretirati različite objekte u stvarnom svijetu te pružiti korisnicima prilagođene informacije i interakcije. Umjetna inteligencija može se koristiti za:

- **Prepoznavanje objekata i scena** u stvarnom vremenu i pružanje dodatnih informacija o njima korisnicima. Na primjer, AR aplikacija može prepoznati vrstu namještaja i sugerirati odgovarajuće dodatke iz kataloga.
- **Interaktivne edukativne aplikacije** mogu prepoznati obrazovne materijale i pružiti korisnicima interaktivne lekcije. Na primjer, AR aplikacija može prepoznati kemijske elemente na tablici i prikazati njihove 3D modele te informacije o njihovim svojstvima.

4.3.2. Strojno učenje

Strojno učenje omogućuje aplikacijama proširene stvarnosti da uče iz podataka i prethodnih iteracija te postaju preciznije i učinkovitije u prepoznavanju i interpretaciji okoline. Apple-ova Core ML biblioteka omogućava jednostavnu integraciju modela strojnog učenja u aplikacije proširene stvarnosti.

Neke od primjena dubokog učenja u mobilnim aplikacijama su:

- **Personalizacija korisničkog iskustva** omogućava analizu korisničkih postavki i ponašanja te prilagodbu AR iskustva njihovim potrebama. Na primjer, aplikacija za dizajn interijera može naučiti korisnikov stil i predlagati relevantne komade namještaja.
- **Analiza pokreta** omogućuje korištenje strojnog učenja za praćenje i analizu korisnikovih pokreta, omogućujući preciznije interakcije. Algoritmi za analizu pokreta mogu prepoznati složene geste i radnje.

4.3.3. Utjecaj na proširenu stvarnost

Integracija tehnologije proširene stvarnosti s naprednim alatima poput umjetne inteligencije i strojnog učenja značajno unapređuje mogućnosti i korisničko iskustvo aplikacija za uređaje koji koriste iOS. Ove tehnologije omogućuju preciznije prepoznavanje objekata, bolje prilagodbe korisnicima i stvaranje interaktivnih i dinamičnih iskustava. Kroz daljnji razvoj i integraciju ovih tehnologija, aplikacije proširene stvarnosti nastavit će pružati sve sofisticiranije mogućnosti, transformirajući način na koji korisnici komuniciraju s digitalnim sadržajem u stvarnom svijetu.

5. Izrada mobilne aplikacije AR Scanner

Detaljno su analizirani primjeri Ikea aplikacije i Measure aplikacije te su objašnjeni principi koje te aplikacije koriste za implementaciju proširene stvarnosti. U ovom poglavlju bit će opisani koraci za izradu mobilne aplikacije proširene stvarnosti za iOS uređaje. Aplikacija koristi ARKit i LiDAR tehnologiju za precizno skeniranje i mapiranje prostora, dodavanje virtualnih objekata te omogućava interakciju s njima. Uz to, koristi SceneKit za poboljšanu kvalitetu iscrtavanja i prikaza virtualnih objekata u stvarnosti, kako bi oni izgledali što realističnije.

5.1. Kreiranje projekta

1. Preuzimanje i instalacija Xcode-a:

- Za instalaciju Xcode-a potrebno je imati operacijski sustav macOS. To je moguće postići na Apple računalima ili koristeći virtualni operacijski sustav.
- Xcode je moguće preuzeti s App Store trgovine aplikacija. Nakon preuzimanja, potrebno je instalirati i pokrenuti ga.

2. Kreiranje novog Xcode projekta:

- Otvara se Xcode i odabire "Create a new Xcode project".
- Odabire se predložak "App" pod sekcijom "iOS".
- Unosi se ime projekta, organizacija i druge osnovne informacije.

3. Dodavanje potrebnih biblioteka:

- Uvoze se ARKit i SceneKit biblioteke u projekt. S obzirom da su to Appleove biblioteke, nije potrebno uvoženje istih u projekt pomoću alata za organizaciju i postavljanje vanjskih biblioteka, već se to može jednostavno učiniti dodavanjem naredbi `import ARKit` i `import SceneKit` na sami vrh datoteka u kojima se te biblioteke koriste. Napravivši navedeno, razvijatelj ima pristup svim funkcionalnostima ARKit i SceneKit biblioteka u dokumentima u kojima ih je uvezao.

5.2. Provjera podrške senzora za detekciju svjetla i dometa

Aplikacija će sadržavati funkcionalnosti koje zahtijevaju da uređaj na kojem je pokrenuta ima senzore za detekciju svjetla i dometa. Na uređajima koji nemaju te senzore, može doći do prisilnog gašenja aplikacije ili neželjenog ponašanja kada dođu na zaslon koji zahtjeva podršku spomenutih senzora. Iz tog razloga, kako bi se osiguralo kvalitetno korisničko iskustvo svim korisnicima, pri pokretanju aplikacije potrebno je utvrditi ima li uređaj podršku za navedene senzore.

Kod koji želimo da se pokrene prilikom paljenja aplikacije treba smjestiti u funkciju koja u svom potpisu sadrži `didFinishLaunchingWithOptions`. Ta funkcija dolazi predefiniрана u datoteci `AppDelegate.swift`. Na vrhu te datoteke potrebno je navesti da će se koristiti funkcionalnosti ARKit-a, dodavanjem `import ARKit`. Zatim se poziva statička funkcija koja u potpisu sadrži `supportsSceneReconstruction` s parametrom `.mesh` nad objektom `ARWorldTrackingConfiguration`, koja vraća `bool` vrijednost. Ta vrijednost će označavati ima li uređaj na kojem je aplikacija pokrenuta LiDAR senzor. Zadnji korak jest tu informaciju spremiti u memoriju uređaja kako bi joj se kasnije moglo pristupiti. Navedena provjera i spremanje u memoriju prikazani su na Slika 5.1.

```
func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
) -> Bool {

    let hasLidarSupport = ARWorldTrackingConfiguration.supportsSceneReconstruction(.mesh)
    UserDefaults.standard.setValue(supportLiDAR, forKey: "hasLidarSupport")

    return true
}
```

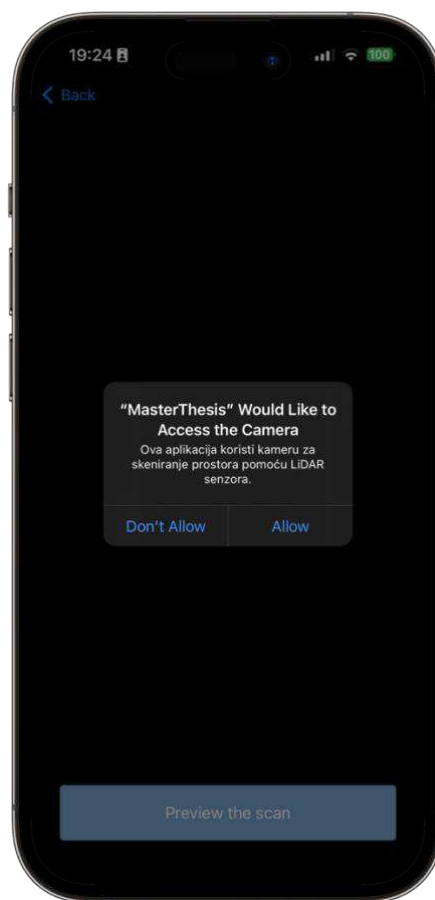
Slika 5.1 Provjera ima li uređaj LiDAR senzor

Iako postoje brojni drugi alati za baratanjem memorijom prilikom razvoja aplikacija za iOS poput Core Data, Keychain, Realm, SwiftData, za ovu implementaciju je odabran `UserDefaults`. Motivacija iza toga jest da Apple preporuča navedenu komponentu za upis jednostavnih, neosjetljivih informacija u memoriju jer je tim podacima moguće lako

pristupiti. S obzirom da informacija o podršci LiDAR senzora ne predstavlja sigurnosnu prijetnju, uzeta je u obzir Appleova preporuka i iskorišten je UserDefaults.

5.3. Zahtjev za pristup kameri

Kako bi funkcionalnost aplikacije bila potpuna, ključan je pristup kameri uređaja. Korisnik će biti zatražen da dozvoli pristup kameri i njezinim funkcionalnostima kao što je prikazano na Slika 5.2.



Slika 5.2 Traženje dozvole korisnika za pristup kameri

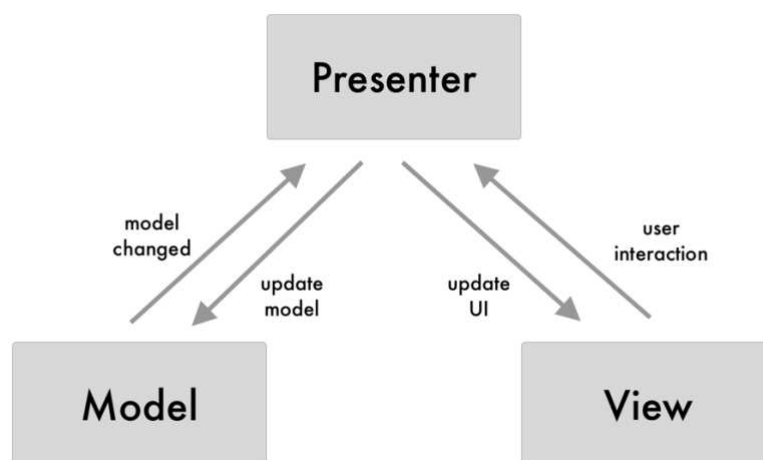
To se postiže novim parom ključ-vrijednost u Info.plist datoteci koja sadrži informacije o konfiguraciji aplikacije. Pod ključ „Privacy - Camera Usage Description“ potrebno je dodati vrijednost „The app requires camera access to scan the room“. iOS će se pobrinuti da korisnika zatraži za pristup kameri prvi puta kada on uđe u zaslon koji to zahtijeva.

5.4. Arhitektura Model Pogled Prezenter – Koordinator

Iako se neće ulaziti u dubinu implementacije arhitekture, zbog važnosti njezina odabira, bit će objašnjena glavna obilježja arhitekture iskorištene za ovaj projekt. Prije izgradnje svake aplikacije valja voditi računa o načinu na koji će njezine komponente međusobno komunicirati te kako će kod biti strukturiran. Za ovaj projekt odabrana je arhitektura Model Pogled Prezenter (engl. Model View Presenter, u daljnjem tekstu MVP) u kombinaciji s Koordinator arhitekturnim obrascem.

- **Model Pogled Prezenter**

MVP arhitektura sastoji se od 3 glavne komponente. Prva je model – to je dio aplikacije koji se brine za dohvat i spremanje podataka. Druga komponenta je pogled koji se brine za iscrtavanje podataka na zaslon, te primanje korisnikovih interakcija na zaslon, primjerice pritiska na određeni gumb ili kliznog pokreta prstom po ekranu. Zadaća pogleda je registrirati takve interakcije i proslijedi informacije o njima sloju prezenter. Prezenter je međusloj koji je pozicioniran između modela i pogleda. Njegova zadaća je primiti interakcije od pogleda i na osnovu njih odlučiti koje podatke iz modela treba proslijediti pogledu na iscrtavanje. Komunikacije je dvosmjerna – prezenter šalje zahtjeve modelu za određenim podacima na osnovu interakcija s pogledom, a zatim šalje podatke iz modela prema pogledu, kao što je prikazano na Slika 5.3. Svrha ovakve arhitekture je imati kod razdvojen po logičkim cjelinama te tako povećati čitljivost, olakšati dugoročno održavanje i pronalazak potencijalnih grešaka u kodu s obzirom na to da je lakše izolirati logičke cjeline i utvrditi odakle problem dolazi.

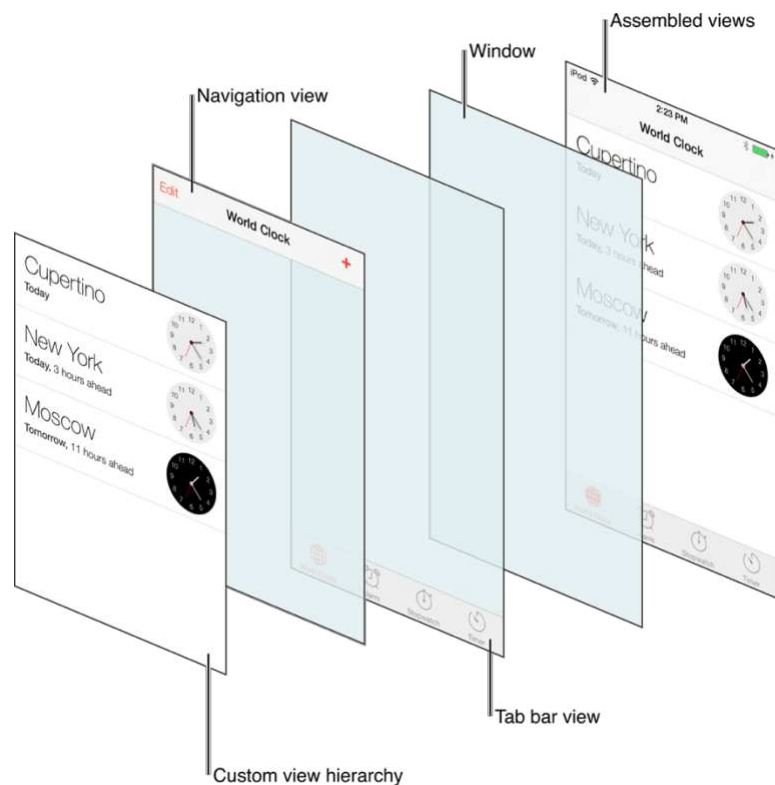


Slika 5.3 Arhitektura Model Pogled Prezenter

- **Koordinator**

U kombinaciji s navedenom arhitekturom, iskorišten je i dodatni arhitekturni obrazac koordinator koji dodatno doprinosi svim navedenim prednostima MVP arhitekture tako da dodatno odvaja kod za navigaciju između različitih zaslona u zasebnu cjelinu. Obrazac koordinator uklapa se u navedenu arhitekturu tako da svakom presenteru daje referenca na središnji dijeljeni koordinator unutar kojeg se nalaze predefinirane funkcije za navigaciju. Prezenteri koji primaju interakcije sa sloja pogled tako mogu registrirati da je primjerice pritisnut gumb koji treba otvoriti novi ekran te tu informaciju proslijediti koordinatoru koji će se zatim pobrinuti za prikaz novog zaslona.

Koordinator komponenta koristi Appleovu gradivnu komponentu za navigaciju, UINavigationController. Ova komponenta funkcionira po principu stoga (engl. stack) i koristi „zadnji unutra, prvi van“ (engl. last in, first out, LIFO) princip. To znači da će ekran koji posljednji bude postavljen na stog UINavigationController-a biti prvi ekran koji će biti uklonjen s tog stoga pritiskom na gumb „Nazad“. Komponente koje sačinjavaju sustav UINavigationController-a prikazane su na Slika 5.4.



Slika 5.4 Komponente UINavigationController-a (preuzeto iz [21])

Takvim pristupom stvara se još jedna logička cjelina koordinator koja je zadužena isključivo za navigaciju između zaslona aplikacije, što dodatno olakšava utvrđivanje uzroka pogreški. Primjerice, ako prilikom razvoja aplikacije na pritisak određenog gumba koji treba otvoriti novi zaslon aplikacija ne reagira, razvijatelji će prvo pogledati u koordinator i provjeriti postoji li neka greška tamo. Nakon što je odluka o arhitekturi donesena, vrijeme je za kreiranje početnog zaslona.

5.5. Početni zaslon

Početni zaslon ove aplikacije zove se MenuViewController. Njega prikazujemo pri pokretanju aplikacije tako da pozovemo funkciju start prikazanu na Slika 5.5. U njoj inicijaliziramo objekte presenter i pogled te sloj pogled postavimo na vrh stoga UINavigationControllerera.



```
func start() {
    let menuPresenter = MenuPresenter(coordinator: self)
    let menuViewController = MenuViewController(presenter: menuPresenter)
    navigationController.pushViewController(menuViewController, animated: false)
}
```

Slika 5.5 Funkcija start koja se nalazi unutar koordinator komponente

Za iscertavanje i dizajn komponenti ovog projekta, korištena je vanjska biblioteka SnapKit koja pruža funkcionalnost davanja naredbi komponentama kako da se pozicioniraju u odnosu jedna na drugu. Početni zaslon bit će iskorišten za objašnjenje kako ta biblioteka funkcionira. Na početnom zaslonu nalaze se tri gumba od kojih svaki vodi na jedan od preostalih zaslona aplikacije. Gumbi su dodani u komponentu po imenu UIStackView čija implementacija trenutno nije od velike važnosti. Način na koji je UIStackView iscertan na početnom zaslonu, definiran je koristeći SnapKit biblioteku. Najčešći pristup davanja naredbi kako da se komponenta iscerta jest da se promatra njezina roditelj-komponenta unutar koje se nalazi, te da se dijete-komponenta pozicionira unutar nje. Kako je roditelj-komponenta UIStackView-u u ovom primjeru sami zaslon, ograničenja za iscertavanja prikazana na Slika 5.6 joj govore da se s lijeve i desne strane udalji od rubova zaslona za duplu vrijednost varijable defaultPadding, te da se pozicionira vertikalno i horizontalno u centar svoje roditelj-komponente.

```
stackView.snp.makeConstraints {
    $0.leading.trailing.equalToSuperview().inset(defaultPadding * 2)
    $0.center.equalToSuperview()
}
```

Slika 5.6 Pozicioniranje stackView komponente

Visinu UIStackView komponente određuju njezina konfiguracije te visine komponenti kojima je ona roditelj. Iz tog razloga, za sva 3 gumba, koja se u hijerarhiji ponašaju kao dijete-komponenta od UIStackView-a na ovom zaslonu, postavljamo visinu na konstantnu vrijednost definiranu u buttonText parametru na način prikazan na Slika 5.7.

```
virtualObjectButton.snp.makeConstraints {
    $0.height.equalTo(buttonHeight)
}
```

Slika 5.7 Postavljanje visine virtualObjectButton komponente

Nakon inicijalizacije osnovnih komponenti zaslona i definiranja pravila kako će se te komponente na zaslonu iscrtati, preostaje ih samo stilizirati. Uvijek je dobra praksa imati predefinirane stilove za komponente koje će se koristiti više puta unutar istog projekta kako bi se zadržala konzistentnost i uklonila potencijalna neočekivana ponašanja. Funkcija koja je korištena unutar ovog projekta za stiliziranje svih gumba prikazana je na Slika 5.8.

```
func applyMainButtonStyle() {
    setTitleColor(.white, for: .normal)
    titleLabel?.font = UIFont.futura()
    backgroundColor = UIColor.appBlue
    layer.borderColor = UIColor.gray.cgColor
    layer.borderWidth = 2
    layer.cornerRadius = 16
    clipsToBounds = true
}
```

Slika 5.8 Generična funkcija za stiliziranje komponenti

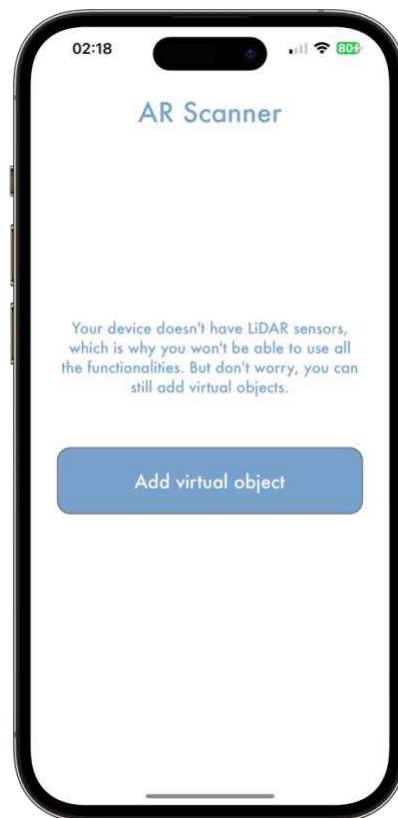
Pozivanjem ove funkcije nad bilo kojim gumbom koji je UIButton tip, pridjeljuju mu se sljedeća svojstva – boja teksta je postavljena na bijelu, font je Futura, pozadinska boja je pastelno plava te mu je dodan obrub širine 2 piksela sive boje. Na kraju, komponenti su zaobljeni rubovi s radijusom od 16 piksela.

Nakon uspješnog prikaza početnog zaslona preostaje razmisliti postoje li potencijalna ograničenja i greške koje je potrebno pokriti i obavijestiti korisnika o njima. Pokrivanje grešaka i neočekivanih slučajeva znatno povećava kvalitetu aplikacije i osigurava da korisnici budu informirani ako aplikacija završi u neočekivanom stanju ili ne funkcionira ispravno.

Za provjeru postoji li greška na početnom zaslonu AR Scannera, korištena je informacija koja je dohvaćena i spremljena u memoriju prema uputama u poglavlju Provjera podrške senzora za detekciju svjetla i dometa, a to je ima li uređaj LiDAR senzor. Ako ima, korisniku aplikacije bit će dostupne sve njezine funkcionalnosti kao što je prikazano na Slika 5.9, dok na uređajima koji nemaju LiDAR senzore, pisat će obavijest i biti onemogućen pristup funkcionalnostima koje zahtijevaju podršku navedenih senzora – prikazano na Slika 5.10.



Slika 5.9 Uređaj ima LiDAR senzore.



Slika 5.10 Uređaj nema LiDAR senzore

6. Integracija senzora za detekciju svjetla i dometa u AR Scanner

Nakon odabira arhitekture, alata za iscrtavanje komponenti i kreacije početnog zaslona, preostaje implementirati funkcionalnosti proširene stvarnosti. Zaslone do kojih se dolazi dodiranjem prvih dva gumba sa Slike 5.9 bit će objašnjeni u ovom poglavlju, a zaslon na koji se dolazi pritiskom na posljednji gumb bit će objašnjen u sljedećem poglavlju. Koristeći arhitekturu MVP s koordinator obrascem, prikaz novog zaslona pritiskom na gumb *Scan anything* izgleda ovako:

1. Sloj Pogled registrira interakciju dodira na gumb i javlja prezenteru

```
@objc private func scanAnythingTapped() {
    presenter.showScanAnything()
}
```

2. Sloj prezenter prosljeđuje tu informaciju koordinatoru

```
func showScanAnything() {
    coordinator.navigateToScanAnything()
}
```

3. Koordinator se brine za prikaz novog zaslona

```
func navigateToScanAnything() {
    let scanPresenter = ScanPresenter()
    let scanViewController = ScanViewController(presenter: scanPresenter)
    navigationController.pushViewController(scanViewController, animated: true)
}
```


6.1. Skeniranje okoline i objekata

Jedan od pristupa za skeniranje okoline koristeći senzore za detekciju svjetla i dometa jest koristeći ARKit i SceneKit biblioteke. Implementaciju tog pristupa moguće je pronaći u praktičnom dijelu ovog rada datotekama `ScanViewController.swift` i `ScanPresenter.swift` gdje su navedene biblioteke uvezene.

Za skeniranje okoline koristeći senzore za detekciju svjetla i dometa, korištena je komponenta `ARSCNView`. To je komponenta koja spaja najbolje iz dva svijeta – virtualne i proširene stvarnosti. Ona je dio SceneKit biblioteke koja se koristi za prikaz virtualne stvarnosti, ali omogućuje integraciju s komponentama iz ARKit biblioteke poput `ARSession` i `ARCamera` [22]. Kako bi ova implementacija radila, `ScanViewController` mora implementirati 2 delegatske metode iz `ARSCNViewDelegate` protokola. Delegatske metode omogućuju aplikaciji da odgovori na događaje u AR okruženju i ažurira scenu u skladu s novim informacijama koje dolaze iz senzora.

- Prva metoda poziva se kada ARKit dodaje sidro (engl. anchor) na scenu i njezina implementacija je prikazana na Slika 6.1. Ova metoda vraća novi `SCNNode` koji predstavlja sidro. Sidro u ARKitu nije prikazano vizualno korisniku, već je referentna točka koja se koristi za orijentaciju u 3D prostoru. U primjeru ispod, radi se specifična provjera je li sidro tipa `ARMeshAnchor` zato što taj tip sidra pruže najdetaljnije podatke o skeniranim objektima [23]. Postoje različite vrste sidara za različite upotrebe (npr. `ARImageAnchor` se koristi prilikom skeniranja slika). Scena može imati više sidara i za svako koje je tipa `ARMeshAnchor`, određuje se geometrijska konfiguracija i ona se upisuje u sami čvor sidra. Konačno, ova funkcija se brine da sva sidra u sceni trenutne sesije budu ispravno konfigurirana kako bi `ARSCNView` mogao ispravno raditi.

```
public func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
    let node = SCNNode()

    if let meshAnchor = anchor as? ARMeshAnchor {
        let geometry = presenter.createGeometry(from: meshAnchor)
        node.geometry = geometry
    }

    return node
}
```

Slika 6.1 Implementacija prve delegatske metode `ARSCNViewDelegate-a`

- Druga metoda poziva se na svakom ažuriranju trenutno vidljive scene kroz kameru uređaja, tj. i najmanji pomak mobilnog uređaja s aktivnim ARSession-om koji koristi ARSCNView komponentu će uzrokovati pozivanje ove metode. Ona tada kontinuirano iterira kroz sva sidra u trenutno vidljivom okviru (engl. frame) i, ako je sidro tipa ARMeshAnchor, ponovno kreira geometriju i ažurira odgovarajući SCNNode. To osigurava konstantno ažuriranje sidara što rezultira maksimalnom preciznošću finalnog modela. Implementacija je prikazana na Slika 6.2.

```
public func renderer(_ renderer: SCNSceneRenderer, updateTime time: TimeInterval) {
    guard let frame = sceneView.session.currentFrame else { return }

    for anchor in frame.anchors {
        if let meshAnchor = anchor as? ARMeshAnchor {
            let geometry = presenter.createGeometry(from: meshAnchor)
            let node = sceneView.node(for: anchor)
            node?.geometry = geometry
        }
    }
}
```

Slika 6.2 Implementacija druge delegatske metode ARSCNViewDelegate-a

Uz implementaciju navedenih metoda koje osiguravaju ispravno postavljanje i ažuriranje sidra koja omogućuju orijentaciju u prostoru ARKit komponentama, potrebno je konfigurirati i pokreni sesiju proširene stvarnosti kao što je prikazano na Slika 6.3. Ključno je postaviti parametar sceneReconstruction na .mesh, jer upravo ta jedna linija daje naredbu sesiji da koristi LiDAR senzore.

```
func startSession(_ session: ARSession) {
    guard !isSessionRunning else { return }

    let configuration = ARWorldTrackingConfiguration()
    configuration.sceneReconstruction = .mesh
    session.run(configuration)

    ...
}
```

Slika 6.3 Konfiguracija i pokretanje sesije

Uz implementaciju navedene 2 metode, ispravno konfiguriranje i pokretanje ARSession sesije, korisnik na zaslonu ScanViewController vidi kameru, koja nakon prepoznavanja dovoljno sidara, počinje iscrtavati čvorove prateći upute za iscrtavanje dane u funkciji createGeometry kao što je prikazano na Slika 6.5. Ključni parametar koji odlučuje o načinu prikaza u funkciji je SCNGeometryPrimitiveType koji je u postojećoj implementaciji postavljen na liniju. Taj parametar odlučuje kojim gradivnim elementom će susjedni čvorovi biti povezani. Preostali tipovi su točka, poligon, trokut ili trokutasta traka.

Nakon što korisnik skenira željeni objekt ili prostor, može pritisnuti gumb *Preview the scan* na dnu zaslona kako bi vidio generirani 3D model. Za njegov prikaz i interakciju s njime koristi se komponenta SCNView. To je komponenta iz biblioteke SceneView koja omogućuje prikaz objekata virtualne stvarnosti. Prije iscrtavanja, objekt mora biti stvoren koristeći funkciju createPreviewNode prikazanu na Slika 6.4. U njoj koristimo postojeća sidra zabilježena u trenutnoj ARSession sesiji, te i iz njih kreiramo SCNNode objekte koji će predstavljati čvorove skeniranog modela. Treba obratiti pažnju kako se ovdje za postavljanje geometrijske konfiguracije ponovno koristi ista funkcija createGeometry. To rezultira s ekvivalentnim modelom prikazanim dok ARSession sesija traje i dok ju gledamo kao virtualni objekt. Konačno, koordinate novokreiranih čvorova transformiraju se aplicirajući SCN matricu. To je zadnji korak koji omogućuje da pregled našeg skeniranog model ne bude statičan, već dio koordinatnog sustava nad kojim je moguće vršiti interakciju poput rotacije. Na Slika 6.6 je prikaz skeniranog modela nad kojim je moguće vršiti rotaciju kliznim interakcijama prsta.

```
private func createPreviewNode(_ session: ARSession) {
    let node = SCNNode()

    guard let frame = session.currentFrame else { return }

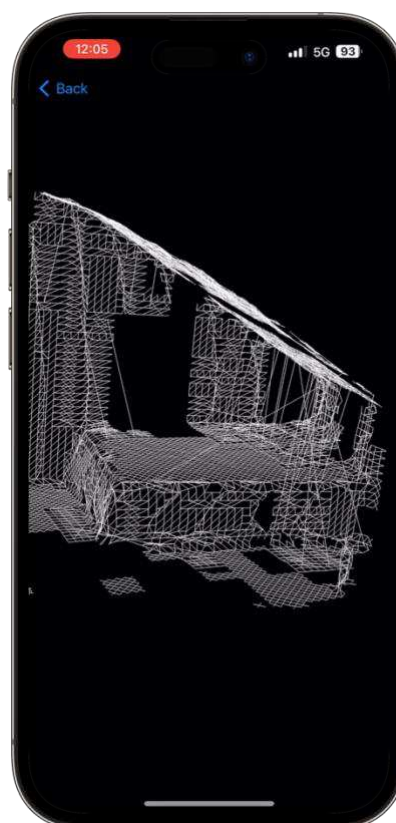
    for anchor in frame.anchors {
        if let meshAnchor = anchor as? ARMeshAnchor {
            let geometry = createGeometry(from: meshAnchor)
            let meshNode = SCNNode(geometry: geometry)
            meshNode.transform = SCNMatrix4(meshAnchor.transform)
            node.addChildNode(meshNode)
        }
    }

    scannedNode = node
}
```

Slika 6.4 Funkcija createPreviewNode



Slika 6.5 Skeniranje okoline



Slika 6.6 Pregled skenirane okoline

6.2. Skeniranje prostorija koristeći RoomPlan

U prethodnom primjeru objašnjeno je kako koristiti moć senzora za detekciju svjetla i dometa, interpretirati dobivene podatke, te konačno prikazati skenirane modele. Iako takva implementacija nosi brojne prednosti poput fleksibilnosti i odgovornosti za upravljanje i modificiranje dobivenih podataka i modela, u nekim slučajevima razvijatelji nemaju potrebu za pristupom svemu već im je za implementaciju potrebno da imaju kontrolu nad pokretanjem, pauziranjem i zaustavljanjem sesije te mogućnostima pregleda i izvozom generiranih modela.

RoomPlan je primjer Appleove biblioteke koji pruža upravo takvu mogućnost i ograđuje razvijatelje od brojnih mapiranja podataka, konfiguriranja čvorova ili ručnog iscrtavanja kako bi se prikazao pregled skeniranog modela [24]. Modeli generirani koristeći RoomPlan biblioteku su prethodno konfigurirani za podršku skeniranja prostorija i unutrašnjosti. To znači da skeniranje koristeći RoomPlan neće biti moguće za bilo kakvu okolinu poput skeniranja iz prošlog poglavlja, već će se specifično odnositi na skeniranje prostorija.

Uz mogućnost skeniranja, RoomPlan koristi modele dubokog učenja za prepoznavanje tipa objekta koji je skeniran. U Tablica 6.1 prikazano je koje objekte ova biblioteka prepoznaje.

Tablica 6.1 Popis objekata koje RoomPlan raspoznaje koristeći duboko učenje

2D Objekt	3D Objekt:
Zid	Prostor za skladištenje
Pod	Hladnjak
Prozor	Štednjak
Otvorena vrata	Krevet
Zatvorena vrata	Sudoper
Otvor	Sušilica/Perilica
	WC/Toalet
	Kada
	Pećnica
	Perilica posuđa
	Kauč
	Stolac
	Kamin
	Televizor

Implementacija je prilično jednostavna i moguće ju je pronaći u datotekama `InteriorViewController.swift` i `InteriorPresenter.swift`. Prvi korak je uvoz biblioteke koristeći naredbu `Import RoomPlan`. Kako bi ova implementacija radila, `InteriorViewController` ne mora implementirati delegatske metode iz `RoomCaptureSessionDelegate` protokola kao što je to bilo neophodno u prethodnom primjeru, ali s obzirom na to da će jedna takva metoda biti korisna za funkcionalnost, ona će biti implementirana kao što je prikazano na Slika 6.7.

- Implementirana metoda poziva se svaki put kad RoomPlan detektira novost u sceni i pruža tu novost kroz objekt CapturedRoom, koji se odmah sprema u presenter što konstantno omogućuje pristup najsvježijem modelu [25].

```
public func captureSession(_ session: RoomCaptureSession, didUpdate room: CapturedRoom) {
    presenter.capturedRoom = room
}
```

Slika 6.7 Delegatska funkcija RoomCaptureSessionDelegate -a

Uz to, potrebno je stvoriti konfiguraciju sesije i pokrenuti ju nad objektom tipa RoomCaptureView što je prikazano na Slika 6.8.

```
private func startSession() {
    let config = RoomCaptureSession.Configuration()
    roomCaptureView.captureSession.delegate = self
    roomCaptureView.captureSession.run(configuration: config)
}
```

Slika 6.8 Konfiguracija i pokretanje sesije

Time osnovna implementacija završava i pritiskom na *Interior scan* gumb sa Slika 5.9, otvara se novi zaslon koji omogućava skeniranje prostora i prepoznavanje objekata kao što je prikazano na Slika 6.9. Rubovi objekata animirano se obilježavaju bijelom linijom kako bivaju prepoznati što korisniku daje do znanja koji dio prostorije je do sad uspješno skenirao. Uz to, RoomCapture model koji se automatski osvježava svakim novim skeniranim objektom je prikazan na dnu ekrana. Da bi korisničko iskustvo bilo dodatno poboljšano, korisniku se daju jasne upute poput koje mu govore kako da uspješno obavi skeniranje kao što je vidljivo na e Slika 6.10.



Slika 6.9 Skeniranje prostorije



Slika 6.10 Upute za skeniranje

Kako bi se sesija skeniranja završila, dovoljno je pozvati sljedeću metodu:

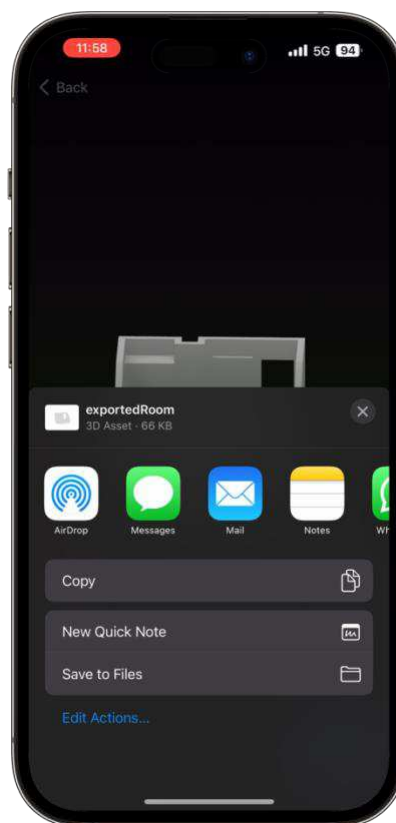
```
roomCaptureView.captureSession.stop()
```

Nakon toga, korisniku se animirano uveća RoomCapture model koji postaje interaktivan. Korisnik ga može rotirati, ali i uvećavati/smanjivati kako bi promotrio detalje kao što je prikazano na Slika 6.11. Taj model, korisnik može izvesti u .usdz formatu jednostavnim pristupom koristeći sljedeću naredbu:

```
capturedRoom.export(to: exportURL)
```



Slika 6.11 Pregled skeniranog modela



Slika 6.12 Izvoz skeniranog modela

6.3. Usporedba pristupa

Kroz prethodna poglavlja istražena su dva različita pristupa skeniranja i generiranja 3D modela koristeći ARKit i RoomPlan biblioteke. Ovi pristupi imaju specifične prednosti i nedostatke te se razlikuju u načinu upravljanja procesom skeniranja, generiranja i prikazivanja 3D modela. U ovom poglavlju ta 2 pristupa će biti uspoređena kako bi njihove svrhe bile jasnije.

6.3.1 Fleksibilnost i kontrola

ARKit u kombinaciji sa SceneKit-om pruža visoku razinu fleksibilnosti i kontrole, što omogućava razvijateljima izravno upravljanje procesom skeniranja, konfiguriranja sidara i čvorova te izrade prilagođene geometrije. Ovo je korisno za aplikacije koje zahtijevaju specifične prilagodbe. Međutim, visoka razina kontrole dolazi s većom složnošću jer razvijatelji moraju ručno upravljati mnogim aspektima, što zahtijeva više vremena za implementaciju i povećava mogućnost grešaka.

S druge strane, RoomPlan pruža jednostavniji pristup s manje potrebne konfiguracije. Razvijatelji mogu brzo pokrenuti i zaustaviti sesije skeniranja te automatski dobiti generirane 3D modele, što je idealno za skeniranje prostorija i unutrašnjosti. Ovaj pristup je ograničen jer razvijatelji nemaju pristup svim aspektima prilagodbe, ali je vrlo koristan za brze implementacije.

6.3.2 Točnost i preciznost

ARKit omogućuje visok stupanj točnosti i preciznosti u skeniranju objekata, koristeći LiDAR senzore za precizno mapiranje okoline. Razvijatelji mogu kontinuirano ažurirati geometriju kako bi osigurali maksimalnu točnost, što je ključno za aplikacije koje zahtijevaju precizno modeliranje.

RoomPlan koristi ugrađene modele dubokog učenja za prepoznavanje objekata, što može poboljšati točnost u prepoznavanju specifičnih tipova objekata unutar prostorija. Međutim, njegova točnost može varirati ovisno o uvjetima skeniranja i složenosti prostora.

6.3.3 Prikladnost za različite primjene

ARKit je prikladan za aplikacije koje zahtijevaju visoku razinu prilagodbe, točnost i preciznost, kao što su aplikacije za detaljno modeliranje objekata ili industrijske primjene. Kompleksnost implementacije može biti izazov za manje iskusne razvijatelje.

RoomPlan je idealan za aplikacije koje zahtijevaju brzo i jednostavno skeniranje unutrašnjosti prostorija s prepoznavanjem specifičnih objekata, poput dizajna interijera ili procjene nekretnina. Ograničena fleksibilnost i kontrola mogu biti prepreka za aplikacije koje zahtijevaju specifične prilagodbe.

Odabir između ARKit-a i RoomPlan-a ovisi o specifičnim potrebama projekta i razini kontrole koju razvijatelj želi imati nad procesom skeniranja i generiranja 3D modela. ARKit pruža visoku razinu fleksibilnosti i točnosti, ali dolazi s većom složenošću implementacije. RoomPlan nudi jednostavniji i brži način skeniranja prostorija s prepoznavanjem objekata, ali s ograničenom fleksibilnošću i kontrolom. Razumijevanje ovih prednosti i ograničenja pomoći će razvijateljima da odaberu pravi alat za svoje specifične potrebe.

7. Dodavanje interaktivnih virtualnih objekata u AR Scanner

U prethodnom poglavlju prikazano je u dva primjera kako integracija senzora za detekciju svjetla i dometa u aplikaciju za uređaje koji koriste operacijski sustav iOS može doprinijeti sustavima proširene stvarnosti. U ovom poglavlju bit će obrađena treća funkcionalnost aplikacije AR Scanner koja također koristi proširenu stvarnost, ali bez senzora za detekciju svjetla i dometa. Do tog zaslona će korisnik doći pritiskom na posljednji gumb u aplikaciji s naslovom *Add virtual object* i on će mu omogućiti dodavanje virtualnog objekta u stvarni prostor i interakciju s njime. Implementaciju je moguće pronaći u praktičnom dijelu ovog rada u datotekama `VirtualObjectViewController.swift` i `VirtualObjectPresenter.swift`.

Ponovno je potrebno uvesti biblioteke ARKit i SceneKit kako bi njihove funkcionalnosti bile dostupne u navedenim datotekama. Glavna komponenta za prikaz stvarnosti i dodavanje virtualnih objekata je `ARSCNView`, koji je prethodno spomenut i analiziran u poglavlju Skeniranje okoline i objekata. Stvaranje konfiguracije i pokretanje sesije bit će slično, uz glavnu razliku da parametar `sceneReconstruction` neće biti postavljen na tip `mesh` što znači da senzori za detekciju svjetla i dometa neće biti korišteni. Pristup koji je korišten u ovom primjeru, prikazan je na Slika 7.1.



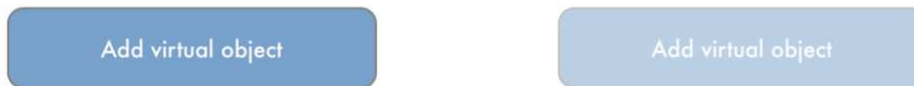
```
func startSession(_ session: ARSession) {
    let configuration = ARWorldTrackingConfiguration()
    configuration.planeDetection = .horizontal

    session.run(configuration, options: [.resetTracking, .removeExistingAnchors])
}
```

Slika 7.1 Konfiguracija i pokretanje sesije

Kako bi prepoznavanje horizontalnih površina bilo omogućeno, parametar `planeDetection` postavljen je na tip `horizontal`. To je ograničenje uvedeno kako bi virtualni objekt bilo moguće dodati tek kada je središte zaslona usmjereno na ravnu plohu. Tako je poboljšano korisničko iskustvo zato što dodavanje virtualnih objekata na neravne površine često rezultira s neočekivanim ponašanjem, neuspjelim dodavanjem i sličnim pogreškama. Kako

bi korisnik znao da dodavanje može izvršiti samo na ravnu plohu, u delegatskoj metodi renderer koja se poziva kad dođe do osvježavanja scene, izvršava se provjera cilja li središte ekrana na horizontalnu plohu u stvarnom okruženju. Gumb za dodavanje virtualnog objekta na osnovu te informacije odlučuje hoće li biti aktivan kao što je prikazano na Slika 7.2.



Slika 7.2 Aktivan i neaktivan gumb

7.1. Dodavanje virtualnog objekta

U trenutku kada je gumb za dodavanje objekta aktivan, u pozadini aplikacija prepoznaje plohu, te ju dodaje kao sidro tipa ARPlaneAnchor u trenutnu scenu. To je ključno za dodavanje virtualnog objekta, koji se kao čvor jednostavno nadoveže na registrirano sidro horizontalne površine. Virtualni objekt se tada stvara kao SCNNode i postavlja na tu površinu. Funkcija addVirtualObject(screenCenter:scene:) sa Slika 7.3 koristi informacije iz trenutnog okvira AR sesije kako bi precizno postavila objekt na odgovarajuću lokaciju.

```
func addVirtualObject(screenCenter: CGPoint?, scene: ARSCNView) -> SCNNode? {
    guard
        let center = screenCenter,
        let query = scene.raycastQuery(from: center, allowing: .estimatedPlane, alignment: .horizontal),
        let transform = scene.session.raycast(query).first?.worldTransform
    else { return nil }

    guard
        let virtualObject = fetchObject()
    else {
        print("Error while fetching virtual object")
        return nil
    }

    let position = SCNVector3(transform.columns.3.x, transform.columns.3.y, transform.columns.3.z)
    virtualObject.scale = SCNVector3(0.01, 0.01, 0.01)
    virtualObject.position = position

    virtualObject.enumerateChildNodes { (node, item) in
        node.castsShadow = true
        node.geometry?.firstMaterial?.lightingModel = .physicallyBased
    }

    scene.autoenablesDefaultLighting = true
    scene.scene.rootNode.castsShadow = true
    scene.automaticallyUpdatesLighting = true
    scene.scene.rootNode.addChildNode(virtualObject)

    return virtualObject
}
```

Slika 7.3 Funkcija za dodavanje virtualno objekta

Virtualni objekt se učitava iz datoteke formata .USDZ. Nakon što je objekt postavljen, njegova geometrija se prilagođava kako bi se osigurala pravilna osvjetljenost i sjene, čime je poboljšana realističnost prikaza kao što je prikazano na Slika 7.4.



Slika 7.4 Virtualni objekt u stvarnom prostoru

7.2. Interakcija s virtualnim objektom

Dodani virtualni objekt postaje interaktivan, što omogućava korisniku da ga dodirne i ukloni iz scene. Dodavanjem UITapGestureRecognizer na ARSCNView, aplikacija može detektirati dodir na objekt. Kada korisnik dodirne objekt, on se uklanja iz scene i gumb za dodavanje postaje ponovno vidljiv, omogućujući korisniku da doda novi objekt. Implementacija te funkcionalnosti prikazana je na Slika 7.5 Slika 7.5.

```
@objc private func handleTap(_ gesture: UITapGestureRecognizer) {
    let location = gesture.location(in: sceneView)
    let hitTestResults = sceneView.hitTest(location, options: nil)

    if let tappedNode = hitTestResults.first?.node {
        tappedNode.removeFromParentNode()
        addButton.isHidden = false
    }
}
```

Slika 7.5 Funkcija koja registrira interakciju s objektom

Zaključak

Proširena stvarnost značajno je napredovala i postala sve prisutnija u svakodnevnom životu zahvaljujući razvoju mobilnih tehnologija i sofisticiranim sustavima poput LiDAR senzora. U ovom radu istražene su ključne tehnologije i metode koje omogućavaju razvoj AR aplikacija za iOS uređaje, s posebnim naglaskom na korištenje ARKit i SceneKit biblioteka te integraciju funkcionalnosti koje pružaju senzori za detekciju svjetla i dometa.

Kroz analizu i praktičnu primjenu, pokazano je kako se ove tehnologije mogu iskoristiti za stvaranje aplikacija koje obogaćuju stvarni svijet virtualnim elementima, omogućujući korisnicima interaktivna iskustva koja kombiniraju fizičko i digitalno okruženje. Razvijena je aplikacija AR Scanner koja koristi ARKit i LiDAR senzore za precizno skeniranje i mapiranje prostora, dodavanje virtualnih objekata i interakciju s njima.

Korištenjem MVP arhitekture u kombinaciji s koordinator obrascem, osigurana je modularnost i lakoća održavanja koda, što je ključno za dugoročni razvoj i skalabilnost aplikacije. Implementacija dodatnih funkcionalnosti, poput prepoznavanja horizontalnih površina i interakcije s virtualnim objektima, dodatno je poboljšala korisničko iskustvo i demonstrirala fleksibilnost i mogućnosti ARKit i SceneKit biblioteka.

Kroz rad su istražene i druge tehnologije koje unapređuju AR iskustva, poput umjetne inteligencije i strojnog učenja, koje igraju ključnu ulogu u prepoznavanju objekata i analizi okoline. Proučavanje primjera poput Ikea aplikacije i Measure aplikacije pružilo je vrijedne uvide u primjenu AR tehnologije u stvarnim scenarijima.

Zaključno, ovaj rad pokazuje da integracija tehnologije proširene stvarnosti u mobilne aplikacije za iOS ne samo da poboljšava interaktivnost i korisničko iskustvo, već otvara nove mogućnosti za inovacije u raznim industrijama. Budući razvoj i daljnja istraživanja u ovom području zasigurno će donijeti još impresivnija rješenja koja će mijenjati način na koji komuniciramo s digitalnim svijetom.

Literatura

- [1] Alan B. Craig., *Understanding Augmented Reality: Concepts and Applications*. USA: Morgan Kauffman, 2013.
- [2] Van Krevelen, Rick, *Augmented Reality: Technologies, Applications, and Limitations*, VU University Amsterdam, 2007.
- [3] Rapid Imaging Tech, *SmartCam3D*, Poveznica: <https://www.rapidimagingtech.com/smartcam3d/>; pristupljeno 4. travnja 2024.
- [4] Cong R., Winters R., *How Does the Xbox Kinect Work*, Jameco Electronics, Poveznica: <https://www.jameco.com/Jameco/workshop/Howitworks/xboxkinect.html>; pristupljeno 17. travnja 2024
- [5] Kunić D., *Mješovita stvarnost (MR) – kratak pregled mješovite stvarnosti*, Virtualna Stvarnost, Kolovoz, 2022 Poveznica: <https://virtualnastvarnost.net/mjesovita-stvarnost-mr-kratak-pregled-mjesovite-stvarnosti/>; pristupljeno 15. travnja 2024.
- [6] Freepik, *Continious one line drawing Man in glasses device virtual reailty and VR*, poveznica: https://www.freepik.com/premium-vector/continuous-one-line-drawing-man-glasses-device-virtual-reality-vr_39789253.html, pristupljeno 19. travnja 2024.
- [7] Gusmão G.F., Barbosa C.R.H., Raposo A.B., *Development and Validation of LiDAR Sensor Simulators Based on Parallel Raycasting*, Sensors (2020)
- [8] Massot-Campos M, Oliver-Codina G. *Optical Sensors and Methods for Underwater 3D Reconstruction*. Sensors. 2015, str. 31525-31557.
- [9] Öhman, N. *Simulation of LiDAR Data for Forestry Applications*. Diplomski rad. Umeå University, Umeå, Sweden, 2018.
- [10] Nuttens, T.; De Maeyer, P.; De Wulf, A.; Goossens, R.; Stal, C. *Comparison of 3D accuracy of terrestrial laser scanning and digital photogrammetry: An archaeological case study*. Proceedings of the 31st EARSeL Symposium: Remote Sensing and Geoinformation Not Only for Scientific Cooperation, Prague, (2011), str. 2-8.
- [11] Hanke, T.; Schaermann, A.; Geiger, M.; Rauch, A.; Schneider, S.-A.; Biebl, E. *Generation and validation of virtual point cloud data for automated driving systems*. Proceedings of the IEEE 20th International Conference on Intelligent Transportation Systems, Yokohama, Japan, (2017), str. 1-10.
- [12] Carrara L, Fiergolski A. *An Optical Interference Suppression Scheme for TCSPC Flash LiDAR Imagers*. Applied Sciences. (2019), str. 1-6.
- [13] Zawacki E., *iPhone lidar with applications for geoscience*, OpenTopography (2022, siječanj). Poveznica: <https://opentopography.org/blog/iphone-lidar-applications-geosciences>; pristupljeno 2. svibnja 2024
- [14] Brookes T., *Your iPhone Pro Has LiDAR: 7 Cool Things You Can Do With It*, How To Geek (2022, svibanj). Poveznica: <https://www.howtogeek.com/759121/your-iphone-pro-has-lidar-7-cool-things-you-can-do-with-it/>; pristupljeno 2. svibnja 2024

- [15] *Macbook Pro*, Apple. Poveznica: <https://www.apple.com/macbook-pro/specs/>; pristupljeno 30. svibnja 2024
- [16] *Does LiDAR work underwater*, YellowScan (2024, veljača) Poveznica: <https://www.yellowscan.com/knowledge/does-lidar-work-underwater/>, pristupljeno 2. lipnja 2024
- [17] Apple Developer Documentation, SceneKit, Poveznica: <https://developer.apple.com/documentation/scenekit>; pristupljeno 2. Svibnja 2024.
- [18] Apple Developer Documentation, ARKit, Poveznica: <https://developer.apple.com/augmented-reality/arkit/>; pristupljeno 2. Svibnja 2024.
- [19] Apple Developer Documentation, RealityKit, Poveznica: <https://developer.apple.com/documentation/realitykit>; pristupljeno 2. Svibnja 2024.
- [20] Ayoubi A., *IKEA Launches Augmented Reality Application*, <https://www.architectmagazine.com/technology/ikea-launches-augmented-reality-application>; pristupljeno 6. svibnja .2024
- [21] Apple Developer Documentation, UINavigationController, Poveznica: <https://developer.apple.com/documentation/uikit/uINavigationController>; pristupljeno 15. svibnja 2024.
- [22] Apple Developer Documentation, ARSCNView, Poveznica: <https://developer.apple.com/documentation/arkit/arscnview>; pristupljeno 22. svibnja 2024
- [23] Apple Developer Documentation, ARMeshAnchor. Poveznica: <https://developer.apple.com/documentation/arkit/armeshanchor>; pristupljeno 22. svibnja 2024
- [24] Apple Developer Documentation, RoomPlan, Poveznica: <https://developer.apple.com/augmented-reality/roomplan/>; pristupljeno 23. svibnja 2024.
- [25] Apple Developer Documentation, CapturedRoom, Poveznica: <https://developer.apple.com/documentation/roomplan/capturedroom>; pristupljeno 27. svibnja 2024

Sažetak

Naslov: Izrada mobilnih aplikacija interaktivne proširene stvarnosti za platformu iOS

Sažetak: Ovaj rad istražuje tehnologije i metode potrebne za razvoj mobilnih aplikacija proširene stvarnosti za iOS uređaje. Fokus je na korištenju ARKit i SceneKit biblioteka, kao i integraciji LiDAR funkcionalnosti za precizno skeniranje i mapiranje prostora. Kroz praktični dio rada, razvijena je aplikacija AR Scanner koja omogućava dodavanje i interakciju s virtualnim objektima u stvarnom prostoru. Rad također analizira primjenu umjetne inteligencije i strojnog učenja u AR aplikacijama te pruža uvid u primjere poput Ikea i Measure aplikacija. Zaključno, integracija AR tehnologija u mobilne aplikacije poboljšava interaktivnost i otvara nove mogućnosti za inovacije u raznim industrijama.

Ključne riječi: proširena stvarnost, iOS, ARKit, SceneKit, LiDAR, mobilne aplikacije, umjetna inteligencija, strojno učenje.

Summary

Title: Building interactive augmented reality mobile applications for the iOS platform

Summary: This thesis explores the technologies and methods required for developing augmented reality mobile applications for iOS devices. The focus is on using ARKit and SceneKit libraries, as well as integrating LiDAR functionalities for precise scanning and mapping of spaces. In the practical part of the thesis, an AR Scanner application was developed, enabling the addition and interaction with virtual objects in the real world. The thesis also examines the application of artificial intelligence and machine learning in AR applications and provides insights into examples such as the Ikea and Measure applications. In conclusion, the integration of AR technologies into mobile applications enhances interactivity and opens new possibilities for innovation in various industries.

Keywords: augmented reality, iOS, ARKit, SceneKit, LiDAR, mobile applications, artificial intelligence, machine learning.

Privitak

Poveznica na izvorni kod aplikacije AR Scanner:

<https://github.com/amarkotic/AR-Scanner>