

# Sadržaj

<b>Popis slika</b>	<b>viii</b>
<b>1 Uvod</b>	<b>1</b>
1.1 Virtualna prisutnost . . . . .	1
1.2 Predmet i cilj rada . . . . .	1
<b>2 Virtualna stvarnost</b>	<b>2</b>
2.1 Primjena virtualne stvarnosti . . . . .	4
2.2 Opasnosti virtualne stvarnosti . . . . .	7
<b>3 Problematika zadatka i slični radovi</b>	<b>8</b>
<b>4 Svojstva predloženog rješenja</b>	<b>11</b>
<b>5 Opis arhitekture sustava</b>	<b>14</b>
5.1 Dubinska kamera . . . . .	15
5.1.1 Princip rada dubinske kamere . . . . .	16
5.1.2 Primjena dubinskih kamera . . . . .	16
5.2 <i>Orbbec Astra</i> kamera . . . . .	21
5.3 <i>Unity game-engine</i> . . . . .	22
5.3.1 <i>Unity editor</i> . . . . .	22

*Sadržaj*

5.4 HTC Vive . . . . .	23
<b>6 Početna razmatranja</b>	<b>25</b>
<b>7 Implementacija softvera za učionicu</b>	<b>27</b>
7.1 OpenGL . . . . .	27
7.2 OpenCV . . . . .	28
7.3 Orbbec Astra razvojni paket . . . . .	28
7.4 TCP Socket . . . . .	30
7.5 Tijek implementacije . . . . .	31
<b>8 Implementacija softvera za udaljenu lokaciju</b>	<b>38</b>
8.1 Arhitektura Unity aplikacije . . . . .	38
8.2 SteamVR . . . . .	43
8.3 Tijek implementacije . . . . .	43
<b>9 Zaključak</b>	<b>49</b>
<b>Bibliografija</b>	<b>51</b>
<b>Sažetak</b>	<b>54</b>

# Popis slika

2.1	Priprema astronauta virtualnom stvarnošću . . . . .	5
2.2	Priprema vojnika virtualnom stvarnošću . . . . .	6
4.1	Prikaz korisnikovog gledišta . . . . .	12
4.2	Pokazivač . . . . .	12
4.3	Dvije osobe u kadru . . . . .	13
5.1	Prikaz konfiguracije cijelog sustava. . . . .	15
5.2	Uzorak točkica infracrvenog projektoru dubinske kamere . . . . .	17
5.3	Komponente dubinske kamere na modernom mobilnom telefonu . . .	18
5.4	Igra za mobilne telefone napravljena kroz izmijenjenu stvarnost koristeći senzor dubine. . . . .	19
5.5	3D skeniranje prostora korištenjem Orbbec Astra kamere. . . . .	20
5.6	Orbbec Astra. . . . .	21
5.7	HTC Vive sa senzorima i upravljačima. . . . .	24
6.1	Prikaz osobe korištenjem trodimenzionalnog modela. . . . .	26
7.1	Arhitektura aplikacije u učionici. . . . .	37
8.1	Preuzimanje modela za izradu učionice. . . . .	40
8.2	Unity Editor. . . . .	41

*Popis slika*

8.3	Komponente GameObject-a upravljača.	42
8.4	Arhitektura aplikacije na udaljenoj lokaciji.	48

# Isječci koda

7.1	Isječak koda koji definira MaskedColorFrameListener klasu . . . . .	31
7.2	Isječak koda koji definira sendData metodu . . . . .	33
8.1	Prijem i pohrana okvira . . . . .	44
8.2	Postavljanje primljene slike za teksturu objekta . . . . .	45

# Poglavlje 1

## Uvod

### 1.1 Virtualna prisutnost

Virtualna prisutnost (eng. virtual attendance) je način virtualne stvarnosti u kojoj se prikazani prostor temelji na nekom stvarnom prostoru i događaju, poželjno u stvarnom vremenu. Omogućuje osjećaj prisutnosti s neke udaljene lokacije.

### 1.2 Predmet i cilj rada

Cilj ovog rada je implementirati sustav virtualne stvarnosti koji omogućava daljinsku prisutnost, to je omogućeno tako da se osoba pomoći uređaja za VR nalazi u virtualnom prostoru u kojem se uživo prikazuje snimka s druge strane. Naprimjer, ako je osoba bolesna i ne može pristupiti predavanju, ona onda može pomoći uređaja za virtualnu stvarnost prisustvovati virtualnoj reprezentaciji predavanja. Osoba bi se našla u virtualnom prostoru koji bi izgledao kao predavaonica, u toj predavaonici prikazivao bi se profesor koji je u stvarnoj predavaonici sniman 3D kamerom iz čije se snimke izrezuje samo tijelo profesora. Za ostvarivanje tog cilja korišten je *Unity* koji omogućuje izradu virtualnog prostora, korištenje VR-a i prikaz snimke unutar toga prostora. Za snimanje korištена je *Orbbec Astra* kamera koja dolazi s podrškom za *Unity*.

## Poglavlje 2

### Virtualna stvarnost

Virtualna stvarnost (eng. virtual reality) je naziv za računalne aplikacije koje nastoje korisniku simulirati iskustvo neke stvarnosti, koristeći pri tome neke tehnologije za pružanje informacija čovjekovim osjetilima putem kojih on doživljava svijet oko sebe. Ideja je da se umjetnim informacijama čovjeku pruža doživljaj da je u nekom drugom prostoru, ulozi, vremenu i slično. Za razliku od tradicionalnih korisničkih sučelja, VR smješta korisnika unutar samoga doživljaja. Umjesto gledanja u udaljeni ekran, on je „potopljen“ i može djelovati na trodimenzionalni svijet. Simuliranjem čim većeg broja dostupnih osjetila, od vida, sluha i dodira pa čak i mirisa, računalo postaje tvorac virtualnog svijeta sa jedinim granicama dostupnosti sadržaja i ograničenosti opreme. Ovaj svijet stvara uvid fizičke i mentalne prisutnosti, okretom pogleda okreće se i prikazani svijet što osigurava da iluzija nikada nije izgubljena. Najprepoznatljivija komponenta VR-a je uređaj kojeg korisnik postavlja na glavu (eng. head mounted display), prikazni uređaj ove posebne namjene, često zvan i *headset*. Ljudi su vizualna stvorenja i prikazna tehnologija je često ono što čini razliku imerzivnih sustava i onih koji to nisu. *Headsetovi* sadrže prikaz putem stereoskopske slike koristeći dva ekrana, po jedan za svako oko. Stereoskopska slika je ona koja prikazuje prostor u tri dimenzije, koristeći dvije dvodimenzionalne slike, vrlo slične, samo prikazane iz gledišta dvije blago udaljene točke, točno onako kako bi ljudske oči vidjele prizor iz stvarnoga svijeta. Takvi uređaji još često uključuju stereo zvučnike i sustav za praćenje pokreta glave i praćenje pokreta u prostoru, koji je bitan radi toga što se na temelju pokreta mijenja slika koju korisnik vidi. Takvi sustavi mogu

## Poglavlje 2. Virtualna stvarnost

sadržavati žiroskope, brzinomjere, magnetometre i slično. Ovakvi su uređaji najčešće na računalo povezani nekim kablom, većinom HDMI i USB priključcima, a u nedavnoj prošlosti započela je proizvodnja bežičnih uređaja. Također, jeftinija popularna verzija VR-a djeluje korištenjem mobilnog telefona za stvaranje stereoskopske slike. Mobilni telefon projicira dvije slike, svaku u svoju polovicu vertikalno podijeljenog ekrana, a korisnik spaja posebni uređaj na njega. Taj uređaj se sastoji od kućišta na koje se telefon spaja, uobičajeno plastično, no često radi uštete troškova proizvodnje može biti i kartonsko. Unutar njega nalaze se dvije leće, slične onima koje posjeduju sami potpuni uređaji za VR. Popularna platforma za razmjenu videozapisa Youtube nudi svojim korisnicima prikaz videa namijenjen za takve uređaje za svaki video sa VR podrškom ili podrškom za prikaz u 360 stupnjeva. [1,2]

VR kakvog poznajemo danas postoji već desetljećima. Prvi popularni *headset* je bio *Oculus Rift* i on je pokrenuo nagli razvoj ove grane. No, mnogo prije njega, u 60-im godinama prošloga stoljeća pojavio se uređaj pod nazivom *Headsight*. On je prvi digitalni uređaj za stvaranje virtualne stvarnosti. Magnetski sustav pratio je položaj glave i na temelju njega je generirao prikaz iz udaljene kamere. Prikaz je koristio po jedan zaslon za svako oko, kao i današnji uređaji. Iskustvo nije bilo virtualno u pravom smislu riječi, zbog toga što računalna simulacija tada još nije bila moguća, no to je bio prvi veliki korak prema dalnjem razvoju tehnologije. Prije njega, postojali su mnogi nedigitalni preci. Najranija slična tehnologija je korištenje panoramskih slika koje obuhvaćaju punih 360 stupnjeva gledišta iz kojih su naslikane. Njihova ideja je da ako popune cijelo vidno polje korisnika, mogu mu predviđiti kako je zaista bilo na tome mjestu u tome vremenu. Primjeri takvih slika dolaze iz samog početka 19. stoljeća. Nedugo nakon (1838. godine), znanstvenik Charles Wheatstone je svojim istraživanjima dokazao koncept stereoskopije kod ljudskih bića, ljudski mozak spaja dvije dvodimenzionalne slike u trodimenzionalnu reprezentaciju objekta, prethodno objašnjeni koncept na kojemu leži i današnja tehnologija u ovoj grani. Tada su proizvedeni i prvi uređaji koji se nazivaju stereoskopi i oni su točno ono što se koristi danas za VR mobilnim telefonom. Nakon toga, ova grana tehnologije je utihnula sve do početka 20. stoljeća kada je ponovno zainskrila poboljšanjem elektronike. Upotreba simulacija započela je stvaranjem simulatora leta. Prvi se pojavio 1929. i bio je u potpunosti elektromehanički. Bio je upravljan koristeći motore povezane na kormilo

## Poglavlje 2. Virtualna stvarnost

i upravljač kako bi simulirali nagibe i okrete kao i motore za simulaciju turbulencija. Bio je korišten za pripremanje američkih pilota sigurnijim načinom i preko 10 tisuća ovakvih uređaja korišteno je za pripremu preko 500 tisuća pilota. Sredinom 1950. razvijen je uređaj zvan *Sensorama*. On je povrh stereoskopskog prikaza i stereo zvučnika još sadržavao i ventilator za simulaciju vjetra, simulatore mirisa i vibrirajuću stolicu. Namijenjen je prikazivanju prilagođenih imerzivnih filmova. Nakon Head-sight uređaja, prvi sljedeći korak bio je takozvani *Sword of Damocles* (1968.) koji nije bio povezan na kameru, već na računalo koje je generiralo grafički prikaz vrlo primitivnih reprezentacija prostorija i objekata. Izraz „virtualna stvarnost“ se prvi put pojavljuje 1987. godine, a najznačajniji uređaji nastali do kraja stoljeća su *Sega VR headset* i *Nintendo Virtual Boy*. Značajni HMD uređaji sadašnjeg doba su *HTC Vive*, *Sony Playstation VR*, *Oculus Rift*, *Samsung Gear VR* i *LG 360 VR*. [3,4]

Često su korišteni i upravljači koji pružaju mogućnost interakcije s objektima unutar virtualnog svijeta, mogućnost kretanja daljeg od udaljenosti koja je dostupna u stvarnom svijetu i drugo.

### 2.1 Primjena virtualne stvarnosti

Virtualna stvarnost ima široko područje primjene. Medicina je jedan od velikih posvojitelja ove tehnologije, time što neke ustanove koriste računalno generirane slike za uspostavljanje dijagnoza i za liječenje. VR simulacije, koristeći stvarne dijagnostičke slike iz ultrazvučnih snimaka ili sličnih, stvaraju 3D modele anatomije pacijenta. Modeli koriste i iskusnim i neiskusnim kirurzima pronaći najsigurniji i najefikasniji način lociranja tumora i smještanja kirurških rezova ili uvježbati teške procedure prije stvarne prilike. Osim zahvata, moguća je i primjena za rehabilitaciju. Žrtve moždanih udara u nekim područjima Europe mogu koristiti terapiju virtualnim okruženjem koje im pomaže povratiti motoričke i kognitivne funkcije brže nego tradicionalnim načinima terapije. Također, virtualna stvarnost može ponuditi pomoći onima koji su oboljeli od nekih uzroka stresa, naprimjer oni koji imaju PTSP, napadajuće panike, fobije i slično. Omogućuje im suočiti se sa svojim strahovima iz sigurnog okruženja. Istraživanje iz područja psihijatrije iz 2014. godine je pokazalo da je ovakvo okruženje dobra zamjena za lijekove pri terapiji ratnih veterana na liječenju od PTSP-ja.

## Poglavlje 2. Virtualna stvarnost

Znanstvenici u području svemirskih istraživanja, koriste zadnje tehnologije virtualne stvarnosti kako bi si olakšali upravljanje udaljenim robotima. Ta primjena se istražuje kako bi se jednog dana moglo kroz virtualnu stvarnost upravljati vozilima na drugim planetima, udaljenima milijunima kilometara. Astronauți također primjenjuju virtualnu stvarnost, time što se njihova obuka i trening olakšavaju prolaskom vježbi u virtualnom okruženju.



Slika 2.1 Priprema astronauta virtualnom stvarnošću.

Od postupka stvaranja nacrta do virtualnih prototipa, automobiliška industrija koristi tehnološki napredne simulacije već desetljećima. *Ford* kompanija je uklopila virtualnu stvarnost kao bitan dio procesa stvaranja novih modela vozila, koristeći *Oculus Rift* uređaj. Zaposlenici mogu koristiti taj sustav da bi pregledali vanjstinu ili unutrašnjost vozila ili da bi mogli sjesti u vozilo prije nego li je ono uopće proizvedeno. Taj prototip omogućuje inženjerima iz raznih odjela da temeljito istraže razne elemente i uoče potencijalne probleme dok nije prekasno. Kompanija *Toyota* koristi virtualne simulacije u drugu svrhu. Nudi simulaciju vožnje namijenjenu mlađim vozačima, stvorenu tako da bi ih educirali o onim distrakcijama pri vožnji koje

## Poglavlje 2. Virtualna stvarnost

su najčešći uzroci nesreća. Također, simulacije vožnje raznih vozila dobar su način za savladavanje osnovnih vještina upravljanja. Učenici letenja mogu svoje osnovne vještine naučiti koristeći sigurnost okruženja simulacije, u kojoj si mogu dozvoliti učiniti pogrešku. U raznim drugim područjima, dizajniranje je olakšano trodimenzionalnim uvidom u kreirani objekt. Primjeri su arhitektonski dizajn, dizajn prometnica i dizajn strojeva. Slično kao za astronaute, virtualni trening ima svoju primjenu i za vojnike. Prilagođene verzije ratnih igrica korištene su za pripremu trupa za borbu. Pružaju zajedničku vježbu u realističnoj replikaciji prije nego to moraju u stvarnoj situaciji.



Slika 2.2 Priprema vojnika virtualnom stvarnošću.

Osim svega toga, postoji i široka primjena u rekreativne svrhe. Sve su češće igrice koje koriste takav koncept. Većina popularnih igračkih konzola pruža tu mogućnost. Također, proizvode se i filmovi koje je moguće gledati iz takve perspektive. Softveri kao *Oculus Cinema* pružaju mogućnost gledanja filma iz okruženja virtualne kino dvorane. Također, postoje i varijante gledanja sportskih događaja virtualnom stvarnošću kroz okruženja virtualnog stadiona, a isto postoji i za gledanje koncerata. Čak je i turistička industrija uključena, time što nude virtualne replike popularnih

## Poglavlje 2. Virtualna stvarnost

turističkih odredišta. [5]

### 2.2 Opasnosti virtualne stvarnosti

U drugu ruku, virtualna stvarnost postavlja i određene zdravstvene i sigurnosne rizike. Neki neželjeni simptomi mogu biti prouzrokovani produljenom upotrebom i oni koče razvoj ove tehnologije. Većina uređaja dolazi s upozorenjima za korisnike vezanim za mogućnost napadaja, mogućnost razvojnih poteškoća kod djece, mogućnost pada pri upotrebi i interferencijom s određenim medicinskim uređajima. Neki korisnici mogu iskusiti trzaje, napadaje ili gubitak svijesti pri upotrebi, čak iako nemaju prethodnih epilepsijskih epizoda. Ovakvi se problemi pojavljuju čak jednoj od 4000 osoba i pošto su to najčešće mlade osobe, preporuča se da djeca ne koriste ovakve uređaje. Druga vrsta opasnosti su one vezane za gubitak svijesti o preprekama u stvarnoj okolini. Korisnici često zaborave na svoj smještaj u stvarnom prostoru i tako se izlažu riziku o spoticanju ili udaranju u neki objekt. Virtualna stvarnost može prouzročiti oftalmološke poteškoće, kao i bilo koje korištenje ekrana. Gledanjem u ekran, osobe manje trepaju što može dovesti do suhih očiju. Bolest virtualne stvarnosti je jedan od rizika korištenja, stvara simptome bolesti putovanja poput morske bolesti. Ženske osobe mnogo su osjetljivije na ovu bolest nego muške osobe i čine 77% slučajeva. Glavni simptomi su općenita nelagoda, glavobolja, nelagoda u trbušu, mučnina, povraćanje, bljedilo, znojenje, umor, usporenost, dezorientacija i ravnodušnost. *Nintendo Virtual Boy* uređaj je primio brojne kritike oko pojavljivanja nekih od tih simptoma. Oni su prouzročeni nepovezanošću između onoga što oči vide i što ostatak tijela osjeća. Kada vestibularni sustav (sustav zadužen za ravnotežu i osjećaj prostorne orijentacije) ne dobiva podražaj pokreta kojeg očekuje radi vizualnog podražaja kojega dobiva, može doći do pojavljivanja ove bolesti. Uzrok, osim toga, može još biti i nedovoljna učestalost slika ili kada postoji zastoj između korisnikovih pokreta i njihove reprezentacije. Pošto čak četvrtina korisnika iskusi nekakve nelagode, proizvođači opreme aktivno pokušavaju pronaći načine za smanjiti pojavu istih. [6]

## Poglavlje 3

# Problematika zadatka i slični radovi

U situaciji kada je pojedinac obavezan prisustvovati nekom događaju, ali je nečime spriječen, razni oblici daljinske prisutnosti su dostupni. Video-sastanci, najčešće korištena metoda za ovakve slučajeve, uglavnom služi dobro ali ne nudi imerzivno iskustvo. Nastojanje da se prilika daljinske prisutnosti učini realističnijom i da korisnik bude uključeniji poboljšava osjećaj prisutnosti, što postavlja bolje uvjete za učenje u slučaju poput ovoga.

Za neke događaje, simulacija je složena od ekrana koji projicira video koji je unutar prilagođenog interijera napravljenog kako bi čim više nalikovao dijelu prostora za sjedenje iz okruženja lokacije događaja u stvarnome životu. Neki primjeri primjene ovoga načina su: prisustvovanje vjenčanju, prisustvovanje sportskom događaju, prisustvovanje predstavi. [7]

Još jedna konfiguracija sustava za rješenje ovoga problema, slična onoj iz ovoga rada je korištenje takozvanih virtualnih događaja. Oni nisu simulacije stvarnih događaja nego su potpuno virtualni događaji. Svi sudionici, korištenjem VR opreme, dijele jedan zajednički virtualni svijet. Jedan rad opisuje integraciju virtualnog okruženja za učenje pri izvođenju nastave. Taj rad, pod nazivom „The global classroom project: learning a second language in a virtual environment“ opisuje napredak projekta kojim se istražuje integracija VR okruženja za suradnju s obukom engleskog jezika na sveučilištu u Hong Kongu. Preko 200 sudionika je bilo uključeno. Veliko netradicionalno postrojenje je razvijeno unutar „Second Life“ virtualnog okruženja u

### Poglavlje 3. Problematika zadatka i slični radovi

obliku američkog restorana iz 1950-ih na pustom otoku, s tradicionalnim stolovima i kabinama te logorskim vatrama izvana kako bi se olakšala kreacija konverzacijskih grupa. I tekstualni i glasovni chat su bili istraženi. [8]

Rješenje korištenjem virtualne stvarnosti s *real-time* replikacijom događaja u pitanju kao metoda daljinske prisutnosti bi bilo vrlo pogodno za mnoge situacije. Nudilo bi ugodan doživljaj i osjećaj imerzivne prisutnosti. Takva bi metoda zahtjevala neki uređaj za snimanje na mjestu događaja, kao i računalo namijenjeno rukovanju i slanju prikupljenih informacija. Također, potrebno bi bilo računalo koje bi koristio udaljeni sudionik, sposobno da prikazuje scenu. Njemu bi trebala biti priključena VR oprema.

Pri implementaciji ovakvog sustava, tehničke specifikacije korištenih računala trebaju biti uzete u obzir. U toku implementacije projekta ovoga rada, oprema je stvarala određene probleme koje je trebalo riješiti. Uloga na strani klijenta traži snažnije računalo. Prvo mora biti sposobno pokretati *game engine*, što zahtjeva grafičku karticu, jaču procesnu jedinicu i veću količinu radne memorije. Brzine operacija pisanja i brisanja na uređaj za pohranu su također bitne, s obzirom na princip rada da se slikovna datoteka konstantno zapisuje i čita pri prijemu podataka i mora držati ritam s učestalošću dobivanja poruka. Pri stvaranju ovoga projekta, određeni *hard disk* je zakazao u tom zadatku i projekt je bio ometen konfliktima između operacija pisanja i brisanja. Problem je riješen izmjenom učestalosti slanja poruka po sekundi. Drugo moguće rješenje je korištenjem radne memorije za pohranu, koja je značajno brža, no neki *game engine*-i to ne podržavaju bez postavljanja cijelog projekta na RAM, što bi vjerojatno zauzelo preveliki dio dostupne memorije na većini računala. Još jedan uvjet je brzina mreže. Ako mrežnim paketima treba previše vremena da bi bili dostavljeni, doživljaj više nije *real-time* ili je kvaliteta prijenosa pokvarena.

Kada se slika koristi za stvaranje iluzije postojanja objekta, neke poteškoće se mogu pojaviti u određenim prigodama. Time što objekt platna nema duljine u smjeru u kojem gleda, gledanje istoga sa boka slama iluziju jer je tada vidljiv tanak, poput lista papira. O tome se treba misliti pri stvaranju ovakvoga okruženja. Najjednostavnije rješenje je sužavanje korisnikovog područja kretanja na ono iz kojega je model dobro vidljiv. Primjer ucionice iz ovoga rada, sasvim je pogodan za nešto ovakvo, zbog toga što je područje kretanja predavača ionako uobičajeno ograničeno

### *Poglavlje 3. Problematika zadatka i slični radovi*

i podijeljeno od prostora za studente. Predavač ima svoje mjesto ispred studenata i taj prostor bi bio postavljen kao nedostupan daljinskom sudioniku. Drugo rješenje je postavljanje sustava s više uključenih kamera, postavljenih na raznim pozicijama, pod raznim kutovima. Tada bi se moglo napraviti da je područje snimanja podijeljeno na dijelove za svaki od kojih bi bila zadužena jedna kamera, što bi riješilo problem ograničenosti prostora za kretanje pri korištenju jedne kamere. Kako bi se predavač kretao, njegov lik bio prikazan kroz odgovarajući snimak. Povećanjem broja kamera povećala bi se vjerodostojnost platna. Odabir pozicije kamere mora se pomno odabrati kako bi čim bolje pokrivala prostor u kojem bi se model mogao naći.

# Poglavlje 4

## Svojstva predloženog rješenja

Predloženo rješenje rješava nedostatke jednostavnosti video prijenosa. Imati ekran koji pokriva cijelo vidno područje zasigurno pruža mnogo bolje iskustvo nego imati klasični ekran koji pokriva samo manji odlomak vidnog polja. Isto tako, mogućnost kretanja kroz prostor i time dostupnost raznih prizora iz različitih kutova gledanja pridonosi ovom osjećaju još više. Korištenje potpuno funkcionalnog rješenja počinje stavljanjem *headset-a* na glavu i paljenjem programa. Korisnik se pojavljuje usred prostorije i ima pogled na stolice, stolove i predavača. Prikaz njegove perspektive može se vidjeti na slici 4.1. Kretanjem u stvarnom prostoru, korisnik se pomiče i u virtualnom. Njegovo je kretanje ograničeno rasponom područja praćenja za korišteni *headset*.

Daljnja kontrola kretanja može se postići korištenjem dodijeljenog ručnog upravljača. Kada je upravljač priključen i nalazi se unutar područja praćenja, sustav projicira pokazni objekt na sjecište površine tla učionice i vektora usmjerenja upravljača. Pokazivač je prikazan u obliku narančaste kugle, vidljive na slici (slika 4.2). On služi kao meta za akcije teleportacije i pomicanja platna. Po tipka na upravljaču je pripisana svakoj njih. Teleportiranje se koristi kada korisnik želi doći do pozicije izvan one do koje može kretanjem u stvarnom svijetu. Nakon što označi željeno mjesto, pritisne određenu tipku i njegova pozicija se premješta tamo. Dodan je efekt zatamnjena ekrana kako bi ova tranzicija bila glađa i ugodnija. Slično, označivanjem i pritiskom tipke, postiže se i promjena pozicije platna. Ono se onda pomiče točno iznad označene lokacije na tlu i postavlja mu se visina analogna visini kamere

*Poglavlje 4. Svojstva predloženog rješenja*



Slika 4.1 Prikaz korisnikovog gledišta

u pravoj učionici, kako bi slika bila čim vjerodostojnija.



Slika 4.2 Pokazivač

Sustav dobro prikazuje i više osoba iz kadra ukoliko njihova udaljenost od kamere

#### *Poglavlje 4. Svojstva predloženog rješenja*

nije previše različita. Ta primjena je korisna u scenariju učionice za situacije kada je student pozvan pred ploču za riješiti zadatak ili slično. Prikaz dvije osobe u kadru vidljiv je na slici 4.3.

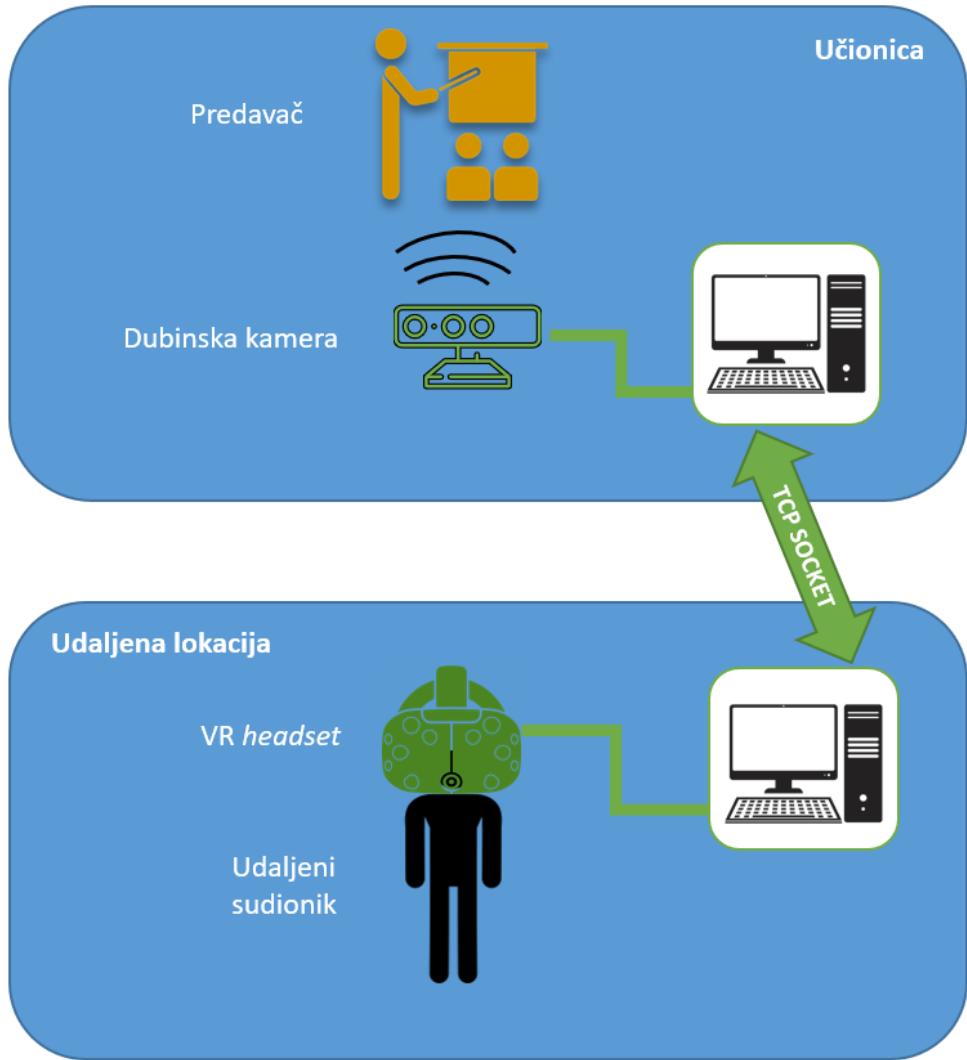


Slika 4.3 Dvije osobe u kadru

# Poglavlje 5

## Opis arhitekture sustava

Konfiguracija aplikacije je podijeljena u dva glavna dijela. Prvi dio nalazi se na lokaciji događaja u učionici. Sadrži računalo na koje je spojena dubinska kamera. Na njemu se vrti *softver* zadužen za izrezivanje i slanje piksela slike. Kamera je postavljena ispred predavača na pomno razmotrenoj poziciji pogodnoj da bi ga snimila u cijelosti. Drugi dio je na udaljenoj lokaciji na kojoj se nalazi daljinski sudionik. Ta strana također sadrži računalo, no na njega nije spojena dubinska kamera nego HMD uređaj. Konfiguracija cijelog sustava je prikazana na slici 5.1.



Slika 5.1 Prikaz konfiguracije cijelog sustava.

## 5.1 Dubinska kamera

Dubinske kamere (eng. depth camera) su kamere koje posjeduju sposobnost prepoznavanja dubine, što im omogućuje bolje shvaćanje prostora oko njih. One koriste infracrvene zrake. Sadrže tri najbitnije komponente:

## *Poglavlje 5. Opis arhitekture sustava*

- običnu RGB kameru
- infracrveni projektor
- infracrvenu kameru.

### **5.1.1 Princip rada dubinske kamere**

Koncept dvije bliske točke gledišta je korišten za izračun udaljenosti svakog objekta unutar pogleda. Poznavanje dva kuta iz kojih je objekt snimljen, korištenjem određenih trigonometrijskih izračuna, pruža mjeru udaljenosti tog određenog objekta, no kada bi se to izvelo samo korištenjem dvije RGB kamere, česte bi bile poteškoće koje bi dovodile do iskrivljenih vrijednosti izračuna. Naime, dva isječka slike (dva različita objekta promatranja) mogu imati sasvim iste ili vrlo slične izglede, koje bi kamera mogla zamijeniti i ne shvatiti da se radi o dva različita. Infracrvene zrake rješavaju ovaj problem. Infracrveni projektor odašilje nepravilni uzorak točaka s valnom duljinom od 700nm do 1mm. Te su točke nevidljive ljudskom oku ali mogu biti vidljive korištenjem kamere s noćnim načinom snimanja. Kako jedan takav uzorak izgleda, prikazano je na slici 5.2. Infracrvena kamera tada snima isječke pjegastog uzorka koji su dizajnirani da se mogu razlikovati jedan od drugoga. Uredaj poznaje kut pod kojim se projicira svaki od njih i može zatim te podatke, zajedno s podatcima RGB kamere iskoristiti za točan izračun udaljenosti i time dovršiti snimak jednog okvira. Nedostatak ove tehnologije je to što funkcioniра jedino u zatvorenom prostoru jer bi previše sunčeve svjetlosti izbjlijedilo infracrveni uzorak i slično, više dubinskih kamera bi se mogle međusobno ometati ukoliko su postavljene preblizu. [9, 10, 11]

### **5.1.2 Primjena dubinskih kamera**

Tehnologija dubinskih kamera najveću će upotrebu zabilježiti kao dio sustava kamera pametnih telefona u skorijoj budućnosti, a na nekima je već upotrijebljena. Do nedaleke prošlosti, osnovna funkcionalnost kamere, pojmiti trodimenzionalni svijet i prenijeti ga u dvodimenzionalni medij bila je dovoljna i gubitku treće dimenzije nije se posvećivala prevelika pažnja. Naglim razvojem tehnologija računalnog vida i strojnog učenja, otvorila su se mnoga vrata za napredak ovog područja. Dovesti

## *Poglavlje 5. Opis arhitekture sustava*



Slika 5.2 Uzorak točkica infracrvenog projektor-a dubinske kamere

prepoznavanje prostora kamera mobilnih telefona do visoke razine omogućilo bi stvaranje vrlo efikasnih aplikacija izmijenjene stvarnosti (eng. augmented reality). Zasada, računalni vid može vrlo uspješno raditi zadatke poput prepoznavanja rukopisa, raspoznavanja nekih objekata. Takve su mogućnosti krucijalne razvoju autonomnih vozila. Računalni vid se iskazao kao vrlo uspješan za dvodimenzionalne zadatke, međutim, pri korištenju računalnog vida u zadatcima koji iziskuju poznavanje treće dimenzije dolazi do mnogih neispravnosti izračuna. Time su raspoznavanje dubine i napredak istoga vrlo ograničeni za slike uhvaćene običnom RGB kamerom. [12, 13]

Ugradnja dubinskih senzora u kamere mobilnih uređaja poboljšava ga u područjima sigurnosti, u kvaliteti slikanih fotografija i doprinosom iskustvima izmijenjene stvarnosti. Na slici 5.3 prikazana je prednja kamera modernog mobilnog uređaja. Kao njezine komponente mogu se zapaziti tri osnovne sastavnice kamere za prepoznavanje dubine. Sigurnost mobilnog uređaja povećava time što nudi prepoznavanje lica vlasnika kao metodu otključavanja uređaja. Koristeći uzorak tisuća infracrvenih točaka, geometrija jedinstvene površine lica vlasnika se snima i sprema kao referenca za buduće otključavanje. Mogućnosti slike kamere mobilnih telefona udaljena je od kvalitete zasebnih kamera time što je veličina njihovih objektiva ograničena

## Poglavlje 5. Opis arhitekture sustava

sve manjim debljinama uređaja. Korištenjem senzora dubine, ova razlika je znatno smanjena. Uređaji mogu na temelju dobivenih informacija dubine zaključiti koji su elementi u okviru slike bliže a koji dalje i time mogu upotrebotom metoda digitalne obrade slika stvoriti privid korištenja određenih načina primjene velikog objektiva. Slikama tako fokus može biti podešen na područje odabrane dubine kako bi željena slika bila stvorena. [12]



Slika 5.3 Komponente dubinske kamere na modernom mobilnom telefonu

Proširenom stvarnošću s kvalitetnim shvaćanjem prostora, bilo koji objekt može biti postavljen kao prividni sadržaj prostora snimljenog kadrom. To nudi mnoge opcije rekreacijskim i korisnim primjenama. Kao rekreacijske primjene postoje mnoge igre koje grade svoje objekte u odabranoj stvarnoj lokaciji (primjer, slika 5.4). Sličnim načinom nudi se i mogućnost stvaranja videozapisa sa nadodanim objektima ili osobama. Mjerenje udaljenosti između dva objekta pokazuje puno precizniju mjeru nego što bi to običnim kamerama bilo moguće. Još neke zanimljive primjene izmijenjene stvarnosti su prikaz zvježđa na snimku neba, prikaz položaja sunca na nebu po satima u danu, prikaz komada namještaja u prostoriji pri korištenju aplikacije za kupnju namještaja.

Osim primjene u sklopu telefona, i samostalni uređaji snimanja dubine imaju široku mogućnost primjene. Područje robotike uvijek je bilo ograničeno osnovnom razlikom u percepciji za razliku od živih bića. Računalni vid robota limitiran je ravnim slikama što razumijevanje okruženja i interakciju čini izazovnim. Pojmiti dubinu

## Poglavlje 5. Opis arhitekture sustava



Slika 5.4 Igra za mobilne telefone napravljena kroz izmijenjenu stvarnost koristeći senzor dubine.

je ključan sastojak za premostiti tu razliku i učiniti robote sposobnima da razumiju i reagiraju na svoje okruženje, kretajući se prostorom i prepoznajući osobe i scene. Trodimenzionalno skeniranje je sve popularnija primjena. *3D scanning* je proces analize prostora ili okruženja s ciljem prikupljanja podataka o obliku i izgledu kako bi se na osnovi njih mogli stvoriti digitalni 3D modeli. Koristeći dubinske kamere, takvi se modeli mogu lako i brzo napraviti. Na slici 5.5 prikazan je proces snimanja prostora koristeći *Orbbec Astra* kameru. Mogućnosti skeniranja mogu se ugraditi na tablet ili neko prijenosno računalo kako bi koristeći kameru mogli stvarati modele u stvarnom vremenu. Ta se primjena se najviše bavi informacijama o obliku mete, a postoje i primjene informacija o pokretima i gestama meta. Praćenje objekata je korisno u mnogim industrijskim, poput rukovođenja skladišta ili maloprodajom. Praćenje gesta i kostura omogućuje interakciju među ljudima i strojevima, a praćenje ljudi gleda na općenitiju primjenu kao praćenje i analiziranje gomila. Povećanjem popularnosti komercijalnih i potrošačkih dronova (malih daljinski upravljenih letjelica s horizontalnim propelerima) rastu i pitanja sigurnosti. Prepoznavanje dubine i prepreka nudi obećanje sigurnosti izbjegavanjem kolizije, ali i strategijama poboljšanja

## *Poglavlje 5. Opis arhitekture sustava*

praćenja i upravljanja na područjima koja nisu pokrivena GPS signalom. Takozvano volumetrijsko snimanje je ustvari snimanje videa u 3D načinu. Slično načinu na koji 3D skeniranje snima nepokretne objekte u visokoj rezoluciji, volumetrijsko snimanje snima ljude ili druge objekte u pokretu. Moguća su razna portabilna rješenja za omogućavanje takvog snimanja. Mjerjenje predmeta također može biti korišteno i samostalnim dubinskim kamerama. Podatci su poprilično precizni i lakoćom se dobivaju za objekte svih oblika i veličina. Također raspoznavanje lica može biti primijenjeno i za otključavanje vrata ili slično. Vrijednost svakog sigurnosnog sustava je onolika koliko je teško zaobići ga. Dodavanjem dubine u jednadžbu raspoznavanja uređaji ne mogu biti prevareni podmetanjem dvodimenzionalne slike, a osim toga omogućuje se i raspoznavanje u svim svjetlosnim uvjetima jer se infracrvene zrake vide dobro i po mraku. Još jedna njihova prednost je to što su sposobni osobu prepoznati iz raznih kutova za razliku od njihovih dvodimenzionalnih suparnika. Sve navedene mogućnosti izmijenjene ili virtualne stvarnosti također se mogu iskoristiti spajanjem dubinske kamere na neko računalo i sve se primjene navedene za mobilne uređaje slično mogu upotrijebiti i u ovoj strukturi. [14]



Slika 5.5 3D skeniranje prostora korištenjem Orbbee Astra kamere.

## 5.2 *Orbbec Astra* kamera

Za ovaj rad, korištena je dubinska kamera modela *Orbbec Astra* (slika 5.6). Kamere iz *Astra* serije dizajnirane su kao daljnja nadogradnja svojstava *Orbbec* 3D kamera za koje tvrde da su bolje od ostalih 3D kamera na tržištu. Astra 3D kamere nude računalni vid koji omogućuje desetke funkcija poput prepoznavanja lica, prepoznavanja pokreta, praćenja ljudskog tijela, trodimenzionalnog mjerjenja, percipiranja okruženja i trodimenzionalne rekonstrukcije mapa. *Astra*, *Astra S* i *Astra Pro* pružaju visokokvalitetnu responzivnost, mjerjenje dubine, glatke gradijente i precizne konture, kao i mogućnost filtriranja niskokvalitetnih piksela dubine. Različite verzije pružaju razvijačima slobodu da se prilagode svojim potrebama koristeći opcije RGB kamera visoke rezolucije pri kratkim i dugim dometima. [15]



Slika 5.6 Orbbec Astra.

## 5.3 *Unity game-engine*

Druga strana je ona gdje je udaljeni sudionik lociran. Njegovo računalo sadrži projekt učionice i spojeni VR uređaj. Ova strana je zadužena za prijem podataka i prikaz slike osobe.

*Unity* je *game engine* za razne platforme korišten za implementaciju ovog dijela projekta. *Unity* je stvoren od strane *Unity Technologies* kompanije, prvi put je predstavljen u lipnju 2005. na *Apple Inc Worldwide Developers* konferenciji kao *game engine* namijenjen isključivo za *Mac OS-X* red operacijskih sustava. Danas *Unity* podržava preko 25 različitih platformi. Može se koristiti za igre u 2D, 3D, virtualnoj stvarnosti, proširenoj stvarnosti, simulacije kao i u druge svrhe. Nudi primarno sučelje za programiranje (API) za *C#* programske jezike, i za *Unity Editor* kroz priključke i za igrice same, kao i mogućnost kreacije povlačenjem komponenti (eng. drag and drop). Unutar dvodimenzionalnih igrica, ponuđen je uvod *sprite*-ova kao i napredni 2D renderer. Za trodimenzionalne igrice, *Unity* dopušta specifikaciju kompresije tekstura, MIP mapa i rezolucijskih postavki za svaku platformu za koju je napravljen i pruža podršku za stvaranje površinskih neravnina, stvaranje refleksija, stvaranje *parallax* efekta, stvaranje sjena, efekte nakon procesuiranja i slično. Prihvaćen je od strane raznih industrija i izvan industrije video igrice. Koristi se u izradi filmova, automobilskoj industriji, arhitekturi, strojarstvu i građevini. Nekoliko velikih verzija je izbačeno. Trenutno najnovija verzija je *2019.2*. Dolazi u tri verzije: *Pro* koji je namijenjen profesionalnim timovima, *Plus* koji daje dodatne resurse za učenje i besplatan sadržaj, *Personal Free* koji je namijenjen početnicima i dolazi bez dodatnih sadržaja. Za ovaj projekt korištena je verzija *Personal Free 2018.3*, za svrhu projekta daljnje nadogradnje nisu bile potrebne. [16]

### 5.3.1 *Unity editor*

Većina razvoja se odvija unutar *Unity Editora*, on nam omogućuje upravljanje svim sadržajima unutar projekta. Unutar *Unityja* možemo igru podijeliti na više razina pomoći scena gdje bi svaka scena bila jedna razina, za ovaj projekt napravljena je samo jedna scena. Možemo mijenjati već definirane objekte unutar *Editora*, pomicati ih, dodavati im skripte koje mogu biti pisane u *C#-u* ili *JavaScriptu*, od više

## *Poglavlje 5. Opis arhitekture sustava*

objekata možemo napraviti predložak (eng. prefab). Unity ima takozvani *Asset store* gdje korisnici mogu kupovati i prodavati gotove 2D/3D objekte, teksture, skripte, proširenja (eng. plugin). Sadržaji mogu biti i besplatni.

### **5.4 HTC Vive**

HTC Vive (slika 5.7) je *headset* za virtualnu stvarnost stvoren od strane HTC-a i Valve-a. Koristi takozvanu *room-scale* tehnologiju praćenja pokreta. To znači da je predviđeno kretanje korisnika u prostoru praćenja, koje se reflektira na umjetno okruženje.

Predstavljen je u ožujku 2015. godine, a razvojna oprema (eng. Development kit) dostupna je od kolovoza iste godine.

S njime u paketu dolaze još i ručni upravljači (za dodatnu interakciju), te bazne stanice (uredaji koji emitiraju infracrvene vibracije s frekvencijom od 60 otkucaja u sekundi koje prepoznaju upravljač i *headset* na temelju čega onda s milimetarskom preciznošću utvrđuju njihove položaje).

Napravljen je kao *hardverski* dio za *Steam VR softver* namijenjen prvenstveno *Windows* operacijskom sustavu. *OpenVR* je razvojna oprema (eng. software development kit) i razvojno sučelje (eng. application programming interface) za stvaranje aplikacija za navedenu kombinaciju tehnologija. [17]

*Poglavlje 5. Opis arhitekture sustava*



Slika 5.7 HTC Vive sa senzorima i upravljačima.

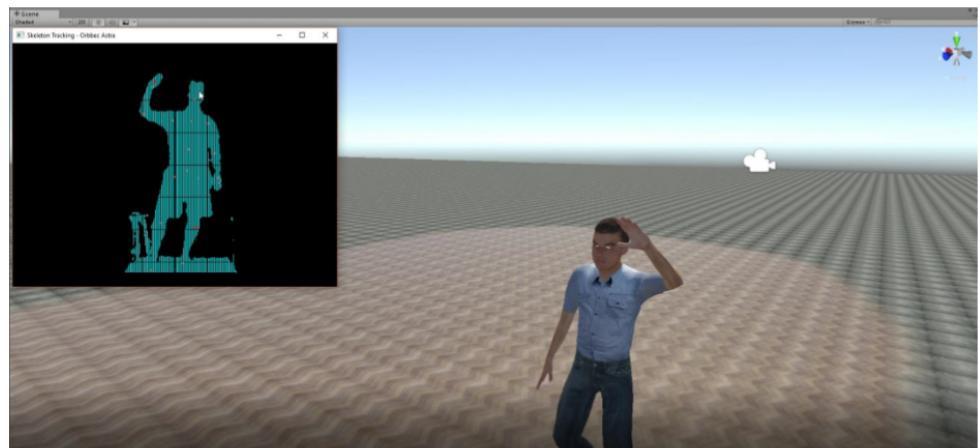
# Poglavlje 6

## Početna razmatranja

Prvotna ideja bazirala se na prijenosu podataka o pozicijama zglobova ljudskog tijela. Implementiran je sustav kojim se lista koordinata za svaki zglob šalje svakim intervalom osvježenja programa. Na drugoj strani, umjesto objekta platna, postavljen je objekt u obliku osobe. Taj je objekt napravljen na temelju dijelova tijela koji su bili praćeni tako da može lako replicirati kretanje snimljene osobe. Prikaz takvog modela korisnika prikazan je na slici 6.1 ispod. Smještaj svakakog dijela bio je ažuriran svakom novom porukom od strane poslužitelja. Bilo je zamišljeno naknadno staviti teksturu modela da odgovara izgledu trenutnog korisnika i čak koristiti sliku lica za postizanje realističnjeg izgleda. Testiranje je pokazalo da je ovaj način u nekoj mjeri nezadovoljavajuć. Dubinske kamere nisu u potpunosti trodimenzionalna metoda snimanja. Često ih se klasificira kao metoda snimanja sa „dvije i pol dimenzije“. Njihovo ograničenje je to što vide samo površine objekata na onoj strani koja je okrenuta prema njima. To je prečesto stvaralo poteškoće. Kada je praćena osoba bila okrenuta prema kameri, model je ispravno pratio sve pokrete. Problemi su se počeli pojavljivati kada je osoba okrenula svoju bočnu ili stražnju stranu prema kameri. Pozicije zglobova modela su se tada prikazale u čudnim neprirodnim pozicijama. Zaključeno je da to previše kvari iskustvo i taj pristup je napušten.

Korištenje izrezanih slika je tada razmotreno kao alternativna ideja. Ideja je implementirana i isprobana i iskazala se kao puno bolje rješenje. Okretanjem bilo koje strane kamери, slika je prikazana iz tog kuta što je dobro imitiralo prisustvo u percipiranom prostoru. Sve poteškoće originalne ideje su tada bile razriješene.

*Poglavlje 6. Početna razmatranja*



Slika 6.1 Prikaz osobe korištenjem trodimenzionalnog modela.

# Poglavlje 7

## Implementacija softvera za učionicu

### 7.1 OpenGL

*OpenGL* (*open-source graphics library*) je okruženje za razvoj prenosivih, interaktivnih 2D i 3D grafičkih aplikacija. Od svojeg prvog pojavljivanja 1992. godine, *OpenGL* je postao najšire korišteno i podržano programsko sučelje za dvodimenzijsku i trodimenzionalnu grafiku, donoseći tisuće primjena širokoj raznovrsnosti računalnih platformi. *OpenGL* cjeni inovaciju i ubrzava razvoj aplikacija uključivši širok paket funkcija za *rendering*, mapiranje tekstura, posebne efekte i druge moćne vizualizacijske funkcije. Razvijači mogu upotrijebiti moć *OpenGL*-a kroz sve popularne *desktop* platforme, osiguravajući široku primjenu.

Svaka vizualna računalna aplikacija kojoj je potreban maksimum u izvedbi, kao 3D animacije, oblikovanje pomoću računala (eng. computer aided design), vizualne simulacije i slično, može iskoristiti *OpenGL*-ove visokokvalitetne mogućnosti visokih performansi. Te mogućnosti dopuštaju upotrebu u raznim tržištima poput oblikovanja, inženjerstva i strojarstva pomoću računala, zabave, medicinskog prikazivanja, virtualne stvarnosti i sličnima za proizvodnju i prikaz nevjerojatno uvjerljivih 2D i 3D grafika.

Za *OpenGL* se tvrdi da je usmjeren prema potrebama razvijača (eng. developer) time što je stabilan, portabilan, skalabilan, pouzdan, lak za upotrebu, dobro dokumentiran, u procesu razvoja i napravljen po standardima industrije. [18]

## 7.2 *OpenCV*

*OpenCV* (*open-source computer vision*) je programska knjižnica (namijenjena jezicima *C* i *C++*) koja služi za funkcije računalnog vida u stvarnom vremenu.

Računalni vid je područje umjetne inteligencije koje se bavi omogućavanjem računalima da postignu razumijevanje slika i videa na visokoj razini i teže automatizaciji zadataka koje ljudski vidni sustav po prirodi izvršava. Osnovna namjena je transformacija slika u informacije, odnosno, prepoznavanje sadržaja slike u pojmovima stvarnog svijeta.

*OpenCV* je prvobitno izašao u lipnju 2000. godine i stvoren je od strane *Intel*-a. Besplatan je za korištenje na temelju BSD *open-source* licence. [19]

## 7.3 *Orbbec Astra* razvojni paket

*Astrin* razvojni paket ili SDK (Software Development Kit) namijenjen je optimizaciji upotrebe *Orbbec Astra* kamera. Da bi se to izvelo, korišteni su koncepti iz mnogih paketa modernih uređaja i senzora, kao i mnoge originalne ideje *Orbbec* tima, s naglaskom na izradu usmjerenu prema kreativnosti razvijača, uklanjanjem bespotrebne kompleksnosti i pružanjem ugodnog i zabavnog iskustva pri razvoju. Dizajneri paketa su i sami iskusni korisnici 3D senzora i time dobro poznaju koje su odlike kamere često korištene. Minimizirana je upotreba ponovljenog koda time što se pružaju tokovi na visokoj razini zajedno s tokovima na niskoj razini u konzistentnom, jednostavnom za razumijevanje programskom sučelju.

Funkcionalnosti su dostupne kroz sučelje za *C* programski jezik kao i za *C++11* verziju programskog jezika. Kroz te temelje, funkcionalnostima se može pristupiti kroz širok broj platformi i jezika. Trenutno su podržani:

- Jezici: *C*, *C++11*, *Java*, *C#*
- Platforme: *Windows*, *OSX*, *Linux*, *Android*
- Okvir *Unity*

Mobilni uređaji postaju sve moćniji, no i dalje su znatno slabiji od stolnih raču-

## Poglavlje 7. Implementacija softvera za učionicu

nala. S time na umu, ovi su paketi građeni kako bi bili manje resursno zahtjevni, minimizirajući pri tome upotrebu procesne jedinice, memorije i baterije.

Bitna značajka ovoga paketa je njegova fleksibilnost, drugim riječima, mogućnost dodavanja priključaka (eng. plugin). Priključci omogućuju naprednim razvijačima mogućnost da prošire mogućnosti razvojnog paketa, a očuvaju konzistentnost i razinu podrške kao i ostatak platforme. Svi su zadani tipovi tokova dostupni i kao zasebni priključci. Ovo omogućuje proširenje aplikacije tako da podržava veći broj različitih uređaja čime se znatno smanjuje vrijeme potrebno za razvoj aplikacija.

Razumijevanje razvoja softvera za *Astra* kamere počinje razumijevanjem triju osnovnih pojmoveva: *Stream*, *StreamSet* i *StreamReader*. *Stream* je sekvensijalni tok okvira od određenog izvora informacija. Kao usporedbu može se spomenuti filmska vrpca starih filmova kojom je video stvoren kao niz pojedinačnih slika u dugoj traci. Ako je jedna takva slika okvir dubinske kamere, onda je *Stream* cijela vrpca. *Streamovi* dolaze u raznolikosti varijanti. *Stream* boje sadrži okvire boje, *Stream* dubine sadrži okvire dubine. *StreamSet* je skupina međusobno povezanih *Streamova*. Za nastavak prethodne analogije, moderni filmovi osim slikovnog niza imaju i zvučni zapis. Spoj videa i zvuka je ono čemu bi u ovoj analogiji odgovarao *StreamSet*. Njima se pristupa koristeći jedinstveni identifikator, kako bi se omogućio pristup većem broju zasebnih skupina ili recimo pristup poslužitelju koji šalje tok visoke razine kao mrežu *StreamSetova*. Naposlijetku, *StreamReader* nudi uvid u okvire toka. Prije nego li ustvari možemo pogledati okvire bilo kojeg tipa, koristeći *StreamSet* klasu kreira se novi *StreamReader* koji nudi pristup svakome od njih. Iz analogije s filmskom vrpcom, on bi predstavljao uređaj za projekciju. Osim što je neophodan za prikaz prikupljenih informacija, on pruža i sinkronizaciju među povezanim dijelovima, predstavljajući pri tome jedno logičko sučelje za pristup svakoj skupini. Svaki ovakav čitač može čitati više različitih tokova odjednom i za svaku skupinu tokova može se stvoriti koliko god čitača je potrebno.

Opširna dokumentacija za rad s *Astra* razvojnim paketom nalazi se u samome paketu i osim objašnjenja osnovnih pojmoveva upotrebe nudi još i osnovne upute korištenja, predloške osnovnih primjena, upute za dobivanje licence, popis svih dostupnih klasa, popis svih datoteka paketa i gotove primjere korištenja osnovnih vrsta tokova.

[20]

## 7.4 TCP *Socket*

*Socket* je jedna krajnja točka dvosmjerne komunikacijske veze između dva programa koja rade na mreži. *Socket* je povezan na broj *porta* kako bi TCP sloj mogao identificirati aplikaciju za koju su primljeni podatci namijenjeni. Uobičajeno, poslužiteljska strana se izvršava na specifičnom računalu i ima *socket* povezan na određeni *port*, kojega sluša čekajući klijenta da napravi zahtjev za spajanjem. Klijent treba znati adresu i taj broj *porta* za uspostavu veze, a kako bi se identificirao, također koristi jedan od svojih *portova*. Adrese i *portovi* dvije strane, zajedno s korištenim protokolom čine 5 glavnih odrednica jednog *socketa*. Portovi su podijeljeni u određene standardizirane raspone, a za određene usluge postoje standardizirane dodjele, kako bi se čim više smanjile greške. Organizacija koja rukuje ovim podjelama naziva se *Internet Assigned Numbers Authority* (IANA). Prvih 1023 su dobro poznati, alocirani su poslužiteljskim uslugama, naprimjer HTTP poslužitelji po pravilu koriste broj 80, a SMTP poslužitelji broj 25. Sljedeća grupa (od 1024 do 49151) su takozvani registrirani portovi. Oni se pridjeljuju specifičnim uslugama i smatrani su polu-rezerviranim. Posljednja skupina (od 49152 do 65535) koristi se za privatne ili prilagođene usluge. Oni su dostupni klijentskim programima. Naprimjer, kada mrežni preglednik uspostavlja nove konekcije alocira si jedan od portova iz ove gomile za svaku od takvih. [21,22, 23]

TCP je jedan od osnovnih protokola *Internet*a i nalazi se na transportnom sloju. Koristi se zajedno sa *Internet Protocol* slojem na *Internet* sloju. TCP pruža pouzdanu, poredanu, provjerenu dostavu niza bajtova između mrežnih aplikacija. Veliki internetski sustavi kao *World Wide Web*, *e-mail*, daljinsko upravljanje, transfer podataka oslanjaju se na TCP. One aplikacije kojima nije potrebna usluga pouzdanog toka podataka koriste UDP (*User Datagram Protocol*) koji pruža prijenos podataka bez prethodne uspostave konekcije, čime poboljšava brzinu prijenosa, a smanjuje pouzdanost. TCP je i dalje pretežno korišten na *webu*, najčešće koristeći HTTP (*Hypertext Transfer Protocol*). [24]

## 7.5 Tijek implementacije

Implementacija cijelog projekta započinje stvaranjem poslužiteljske strane, odnosno strane u učionici događaja. Ona je napravljena u obliku aplikacije u *C++* programskom jeziku. Aplikacija koristi *Astra* SDK, opremu za razvoj softvera za korištenje *Astra* kamera koja sadrži mnoštvo datoteka za korištenje. Kod je implementiran u glavnoj *C++* datoteci imena "main" koja poziva preduvjete iz tog mnoštva. Osim njih, pozivaju se još i prethodno navedeni *OpenGL* i *OpenCV* te nekoliko standardnih knjižica (*iostream*, *stdlib*, *vector*, *fstream*, *algorithm*, *thread*). *OpenGL* je korišten za grafički prikaz snimljenih slojeva. *OpenCV* je korišten za rad s matricama i kompreziji slika.

Glavna skripta sadrži globalnu varijablu naziva „image“ koja je korištena za sačuvanje posljednje prikupljene slike. Klasa *FrameListener* iz *Astrinih* alata služi upravljanju prikupljenih podataka. Njena „on\_frame\_ready“ metoda okida se svakim novim *frame*-om (okvirom) primljenim i uzima objekt klase *Frame* kao parametar. Taj objekt sadrži sve dostupne informacije pojedinog uhvaćenog okvira i može ponuditi sve različite tipove slika dostupne iz mogućnosti uređaja za snimanje. Tip slike koji prikazuje osobu izrezanu iz njene okoline zove se *MaskedColorFrame*. Klasa nazvana *MaskedColorListener* kreirana je kao produžetak (nasljeđivanje) *FrameListener* klase. Njena „on\_frame\_ready“ metoda je nadglasana da izvuče samo *MaskedColorFrame* dio iz *Frame* objekta i pohrani ga u spomenuto „image“ varijablu kako bi mogao biti poslan. Instanca te klase je zatim stvorena kako bi se pokrenule njene funkcionalnosti. Isječak koda 7.1 prikazuje definiciju opisane klase.

Isječak 7.1 Isječak koda koji definira *MaskedColorFrameListener* klasu

```
class MaskedColorFrameListener : public astra::FrameListener
{
public:
    MaskedColorFrameListener( int maxFramesToProcess )
        : maxFramesToProcess_( maxFramesToProcess )
    {
    }
    bool is_finished() const { return isFinished_ ; }
```

Poglavlje 7. Implementacija softvera za učionicu

```
private:
    void on_frame_ready(astra::StreamReader &reader,
    astra::Frame &frame) override
    {
        astra::MaskedColorFrame maskedColorFrame =
            frame.get<astra::MaskedColorFrame>();
        if (maskedColorFrame.is_valid())
        {
            hasNewData = true;
            ++framesProcessed_;
            lastFrame = maskedColorFrame;
            if (true)
            {
                Mat my_frame = Mat(lastFrame.height(),
                    lastFrame.width(), CV_8UC4,
                    (void *)lastFrame.data());
                cv::cvtColor(my_frame, my_frame,
                    cv::COLOR_BGRA2RGBA);
                image = my_frame;
                frameCnt++;
            }
        }
        isFinished_ = framesProcessed_ >= maxFramesToProcess_;
    }
    void print_depth_frame(const astra::MaskedColorFrame
    &depthFrame) const
    {
        const int frameIndex = depthFrame.frame_index();
        const short middleValue = get_middle_value(depthFrame);
        std::cout << "Depth_frameIndex: " << frameIndex << "value: " << middleValue << "\n"
    }
}
```

## Poglavlje 7. Implementacija softvera za učionicu

```
    << middleValue << std :: endl ;
}

short get_middle_value(
const astra :: MaskedColorFrame &depthFrame) const
{
    const int width = depthFrame . width ();
    const int height = depthFrame . height ();

    const size_t middleIndex =
        ((width * (height / 2. f)) + (width / 2. f));
    const short middleValue = 2;

    return middleValue;
}
bool isFinished_ {false};
int framesProcessed_ {0};
int maxFramesToProcess_ {0};
};

MaskedColorFrameListener listener (100);
```

*Main* funkcija je ona koja se prva izvršava u *C++* programu. Ona od ove aplikacije zove funkciju inicijalizacije od *Astrinog* SDK-a i zatim stvara dvije dretve, koje su način istovremenog izvršavanja koda. Prva dretva je za grafički prikaz prepoznatih slojeva. Kroz drugu se izvršava funkcija za slanje podataka klijentu. Zatim funkcija za ažuriranje *Astra* SDK-a se poziva kroz beskonačnu petlju, kako bi se funkcionalnosti *Astre* mogle izvršavati do kraja izvršavanja aplikacije.

Nakon toga, nalazi se blok koda koji definira funkcije povezane na događaje pritiska tipke na tipkovnici. Njima je omogućena kontrola pokretanja i zaustavljanja aplikacije. Spomenuta funkcija za slanje podataka, nazvana „sendData“, prvo inicijalizira *Windows Sockets* API konekciju a zatim TCP *socket* konekciju koristeći *SocketTCP.hpp* knjižicu. Njen kompletan sadržaj prikazan je niže u isječku 7.2.

## Poglavlje 7. Implementacija softvera za učioniku

Isječak 7.2 Isječak koda koji definira sendData metodu

```
void sendData()
{
    string ipAddress = "192.168.1.18";
    // Adresa daljinskog sudionika
    int port = 54000;
    WSADATA data;
    WORD ver = MAKEWORD(2, 2);
    int wsResult = WSAStartup(ver, &data);
    // inicializacija WinSock-a
    if (wsResult != 0)
    {
        cerr << "Greska u Winsock, " << wsResult << endl;
        return;
    }
    SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
    // Kreacija socketa
    if (sock == INVALID_SOCKET)
    {
        cerr << "Greska u socketu, " <<
        WSAGetLastError() << endl;
        return;
    }
    sockaddr_in hint;
    hint.sin_family = AF_INET;
    hint.sin_port = htons(port);
    inet_pton(AF_INET, ipAddress.c_str(), &hint.sin_addr);
    int connResult
    = connect(sock, (sockaddr *)&hint, sizeof(hint));
    // Spajanje socketom
    while (connResult == SOCKET_ERROR)
    {
```

## Poglavlje 7. Implementacija softvera za učionicu

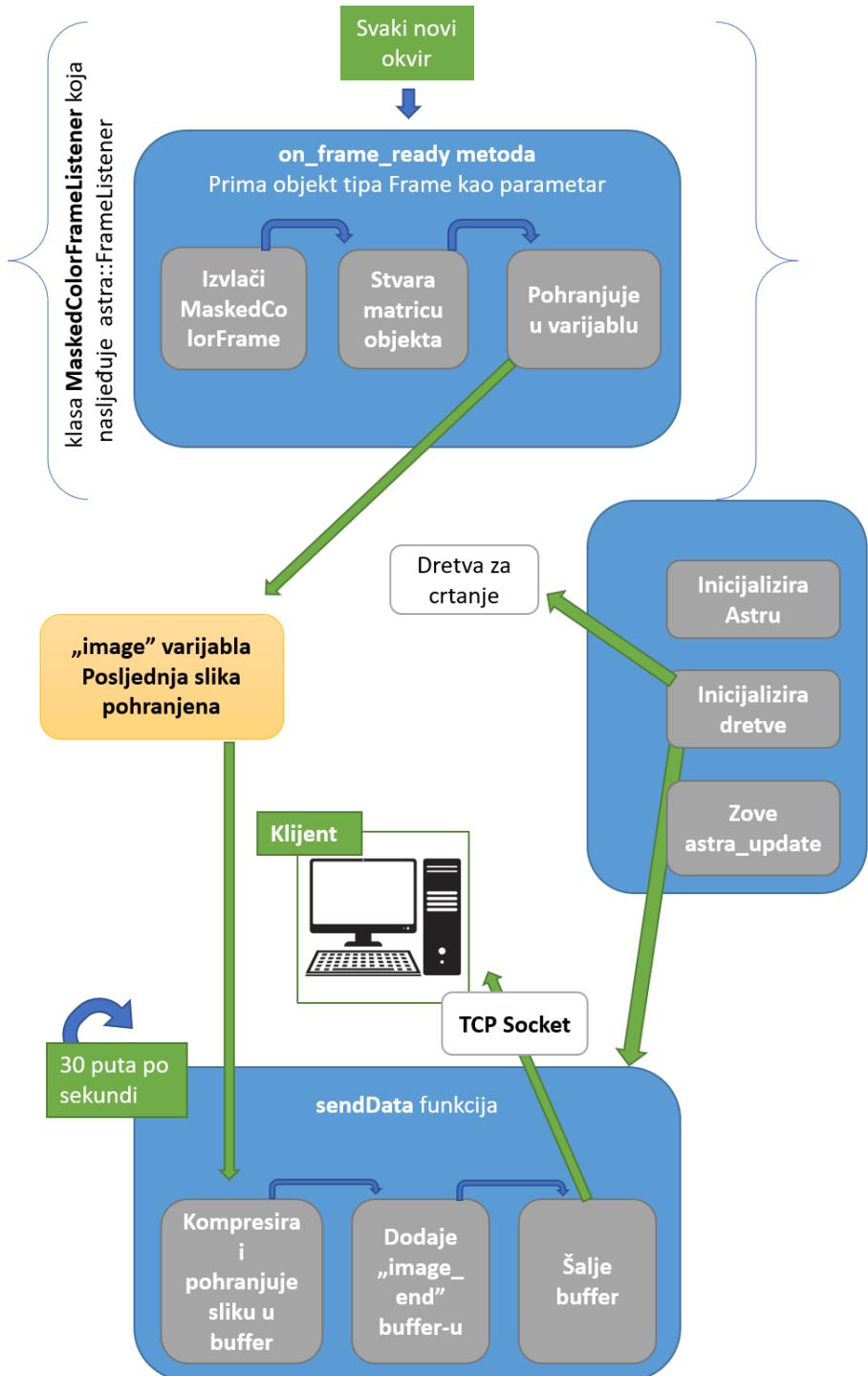
```
cerr << "Greska u povezivanju socketa #"
<< WSAGetLastError() << endl;
Sleep(2000);
connResult
= connect(sock, (sockaddr *)&hint, sizeof(hint));
}
std::vector<BYTE> buffer;
do
{
    Sleep(33); // 30 okvira po sekundi
    try
    {
        cv::imencode(".png", image, buffer);
    }
    catch (const std::exception &)
    {
        printf("Greska u kodiranja");
    }
    char ascii_str[] = "image_end";
    int len = strlen(ascii_str);
    BYTE arr[9];
    int i;
    string2ByteArray(ascii_str, arr);
    for (int i = 0; i <= 8; i++)
        buffer.push_back(arr[i]);
    if (!sendPoint(sock, (char *)buffer.data(),
    buffer.size())))
    {
        std::cout << "Greska u slanja";
        break;
    }
    else
```

## Poglavlje 7. Implementacija softvera za učionicu

```
{  
    printf( "Slika_poslana" );  
}  
buffer . clear ();  
hasNewData = false;  
} while (gameRunning);  
closesocket (sock );  
WSACleanup ();  
}
```

Sljedeće, pri uspjehu TCP *socket* postavljanja, *sendData* pokreće petlju koja traje cijelim preostalim trajanjem životnog vijeka aplikacije. Petlja počinje pozivom funkcije odgode (*sleep* funkcija). Njena svrha je postizanje određene učestalosti slanja. Kada nje ne bi bilo, funkcija bi slala podatke čiji ritam ili mreža ili primajuća strana možda ne bi bili u stanju pratiti. Nakon toga, *buffer* memorije u obliku polja bajtova je inicijaliziran. Vrijednost varijable „image“ je povučena i provučena kroz *encode* funkciju *OpenCV* knjižnice koja ju kompresira i pohranjuje u memorijski *buffer* koji je veličine da odgovara rezultatu. Nakon toga, *bufferu* se dodaje niz sa sadržajem „image\_end“. Taj je tekst namijenjen strani kojoj se šalje kao oznaka završetka prijenosa jednog okvira. Time se sprječava zabuna oko toga koji podatci pripadaju kojemu okviru. *Buffer* je zatim poslan putem *send* funkcije iz knjižnice za TCP socket čime se zaključuje jedna instanca iteracije petlje. Sažeto, aplikacija poslužiteljske strane se sastoji od tri istovremena „procesa“. Prvi donosi podatke svakog novog okvira i sprema *MaskedColorFrame* dio, drugi grafički prikazuje podatke na jednom od prozora i treći oblikuje pohranjenu varijablu u niz bajtova i šalje ju kroz *socket* svaki određeni interval. Tri navedena procesa prikazana su na slici 7.1 u obliku grafa. Prikazani su u oblicima plave boje, njihove logičke akcije su prikazane kroz sive kutije s plavim strelicama unutar njih koje im opisuju redoslijed. Njihove vanjske veze su prikazane strelicama zelene boje.

Poglavlje 7. Implementacija softvera za učionicu



Slika 7.1 Arhitektura aplikacije u učionici.

# Poglavlje 8

## Implementacija softvera za udaljenu lokaciju

### 8.1 Arhitektura *Unity* aplikacije

*Unity Hub* je samostalni alat za korištenje *Unityja* koji pojednostavljuje način pronaleta, preuzimanja i upravljanja *Unity* projektima i instalacijama. Pri paljenju *Unity Editora*, ukoliko *Hub* već nije instaliran, korisniku se nudi instalacija istog. Nakon pokretanja *Huba*, za kreaciju novog projekta potrebno je odabrati predložak od kojeg će se početi. Predlošci pružaju prethodno odabране postavke bazirane na čestim najboljim običajima izrade ovog tipa projekta. Ti su predlošci optimizirani za dvodimenzionalne i trodimenzionalne projekte širom cijelog raspona određenih platformi koje *Unity* podržava. Zadani predložak je 3D predložak. [25]

Imanje (eng. asset) je naziv za bilo koju stavku koja se može koristiti unutar projekta i takve se smještaju u *Assets* direktorij. *Asset* može poteći iz datoteke bilo kojeg podržanog tipa ili datoteke kreirane izvan *Unityja*. Neki česti tipovi su:

- slikovne datoteke (podržani su česti tipovi slika poput BMP, TIF, TGA, JPG i PSD)
- modeli (moguće je uvoz modela bilo kojeg alata za stvaranje 3D modela koji podržavaju FBX format, kao i *SketchUp* datoteke)
- mnogokutne mreže točaka kao kosturi objekata (eng. mesh) i animacije (sadr-

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju

žani u zajedničkoj datoteci)

- audio datoteke
- drugo

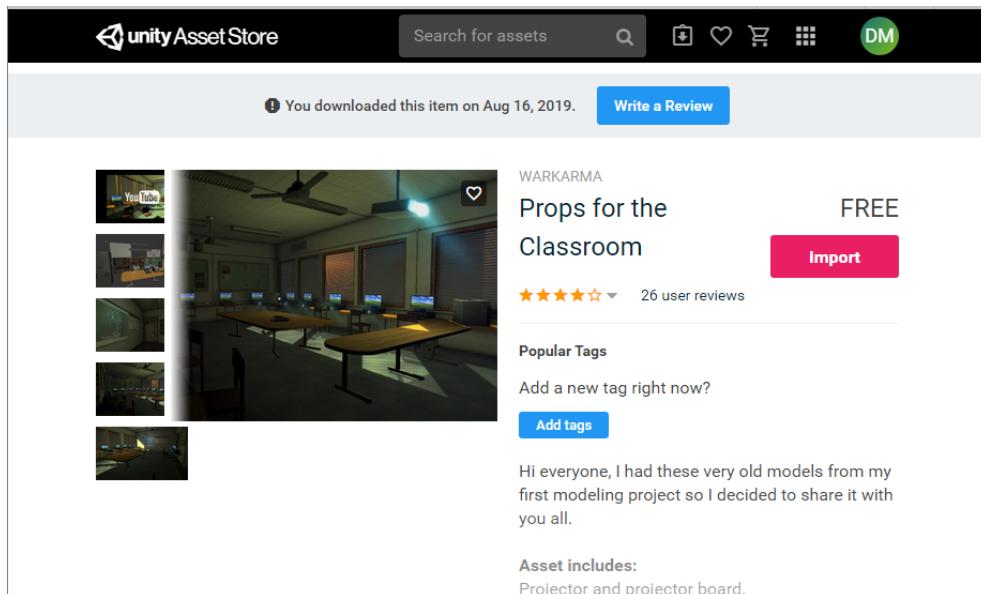
U svim slučajevima, originalne datoteke nisu uređivane od strane *Unityja* bez obzira na mogućnosti odabira metode kompresije, modifikacije ili procesuiranja *Assetsa*. Proces uvoza čita izvornu datoteku i stvara joj reprezentaciju spremnu za pokretanje unutar instance igre, napravljenu po postavkama uvoza. Promjena neke od tih postavki ili promjena samog izvornog fajla, prouzročit će ponovno izvršavanje ovog uvoznog postupka. Širok raspon ovakvih imanja (uključivši priključke, knjižnice i alate) može biti instaliran u *Unity* koristeći *Unity Package Manager* (UPM). *Unity* dolazi s paketom standardnih *Assetsa*. To su kolekcije onih elemenata koji se gotovo uvijek koriste u projektima. Uključuju 2D objekte, kamere, likove, efekte, okruženja, vozila, prototipe, sustave čestica. Za njihov prijenos iz projekta i u projekt, oni se pakiraju u takozvane *Asset* pakete. Oni su dostupni u *Unity Asset Store*-u. [26]

*Asset Store* je *Unityjeva* trgovina koja sadrži mnoge besplatne i komercijalne sadržaje kreirane od tvoraca *Unityja*, a i od članova zajednice. Raznoliki resursi su dostupni, od modela, tekstura i animacija do gotovih projekata i dodataka za uređivač. Na slici 8.1 je prikazano sučelje aplikacije za korištenje trgovine pri implementaciji ovoga projekta. U tome trenutku uvezeni su modeli stolica i stolova primijenjeni za slaganje učionice.

*Manifest* projekta je bitna projektna datoteka koja opisuje njegove zavisnosti. Pri otvaranju nekog projekta UPM čita *manifest* kako bi znao koje pakete učitati u projekt. Zatim šalje zahtjev poslužitelju za svaki paket koji je definiran kao zahtjev. Kontaktirani poslužitelj odgovara informacijama koje su zatim instalirane u projektni direktorij. Ovakav pristup koristi se pri razvoju *softvera* radi pojednostavljenja dijeljenja ili verzioniranja projekata. Nema smisla prenositi javno dostupne pakete ili ih staviti na daljinski repozitorij iz razloga uštede prostora za pohranu. Bitno je samo podijeliti *manifest* i onda za svakog korisnika projekta *Package Manager* brine da je instalirano sve što je potrebno. [27]

*Editor* (slika 8.2) je sastavljen od nekoliko sekcija kroz koje se koriste njegove mogućnosti. U donjem dijelu nalazi se projektni prozor. On prikazuje datotečnu

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju

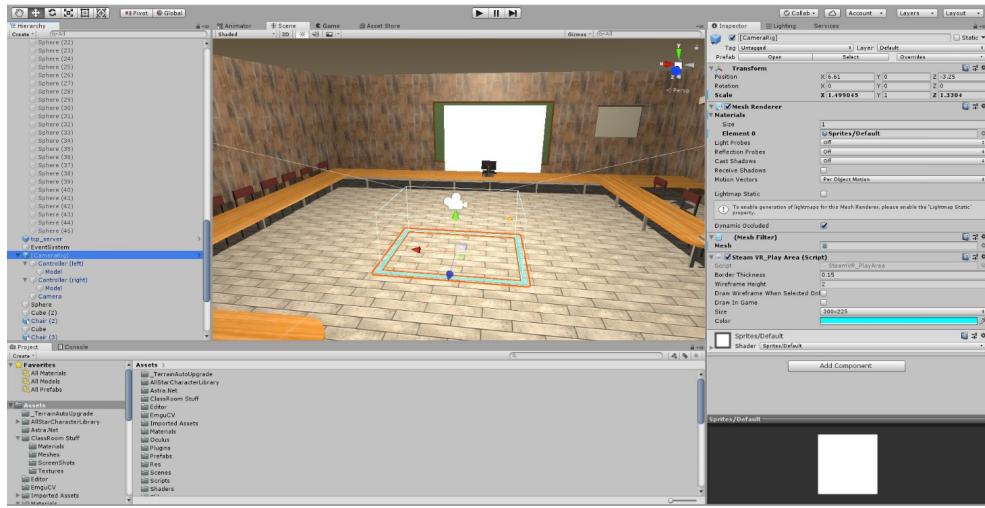


Slika 8.1 Preuzimanje modela za izradu učionice.

strukturu svih sadržaja projekta. Prikaz scene (eng. scene view) interaktivni je prikaz svijeta koji se kreira. Prikaz scene se koristi za odabir i smještaj prizora, likova, kamere, osvjetljenja i svih drugih objekta tipa *GameObject*. Sposobnost odabira, manipulacije i izmjene putem ovoga prikaza je jedna od prvih vještina koje je potrebno savladati pri učenju korištenja *Editor* alata. Prikaz igre nudi uvid u perspektivu kamere igre, odnosno pogled u izvršnu verziju. Za pokretanje ovog pogleda, potrebno je postaviti jednu ili više kamera u scenu. Sve promjene unutar ovoga načina služe samo testiranju i ostaju samo u toj instanci pokretanja te se brišu njenim zaustavljanjem. Hjерархијски prikaz je lista svih objekata trenutne scene. Neki od njih su direktnе instance neke od *Assets* datoteka, druge su instance *GameObject* predložaka koji se nazivaju *Prefabs* i čine većinu igre. Dodavanjem ili brisanjem objekata iz scene (kroz *Editor* ili kroz logiku same igre) objekti se stvaraju i nestaju i sa hijerarhijske liste. Koristi se koncept roditelja i djece što znači da svaki objekt ima svojeg roditelja sve do objekta scene koja je predak njima svima.

Svaki od objekata od kojih se projekt sastoji posjeduje skripte, mrežne kosture, zvukove i druge grafičke elemente poput svjetla. Njih se može vidjeti u sekciji zva-

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju



Slika 8.2 Unity Editor.

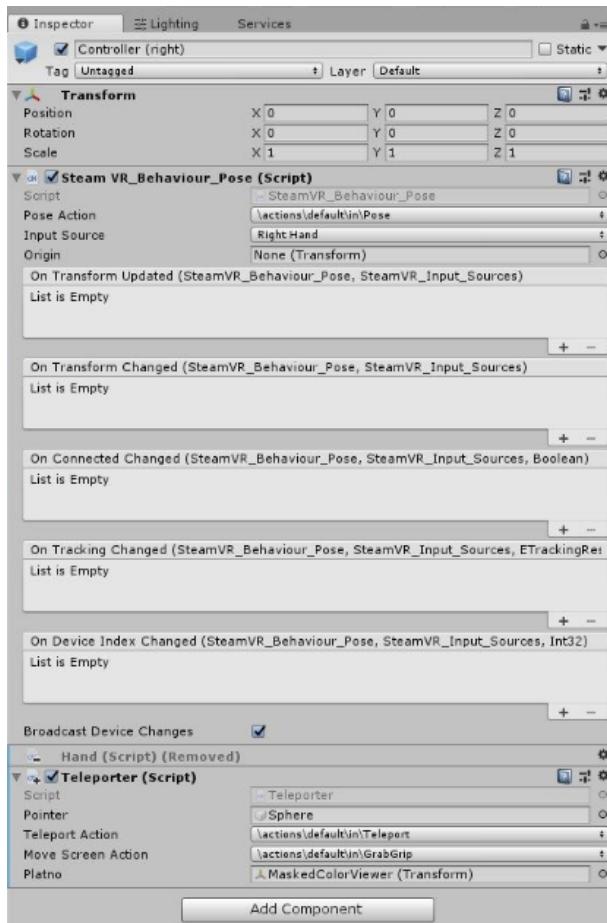
noj *Inspector window*. Ondje su prikazane detaljne informacije trenutno odabranog objekta igre, njegove komponente i parametri i dostupno je uređivanje funkcionalnosti tog objekta u sceni. [28]

Konfiguracija jedne scene se sprema u datoteku *.unity* formata koja je posuda za sve objekte, okruženja, dekoracije, kamere, osvjetljenja i prepreke. Uobičajeno je koristiti svaku scenu kao jednu razinu igre. Pri kreaciji novog projekta, korisnik dobiva novu scenu na raspolaganje koja ne sadrži ništa osim kamere i izvora svjetlosti. [29]

*GameObject* je najbitniji koncept razvoja u ovom *engine-u*. Svaki objekt koji pripada projektu je jedan takav, od likova do skupljivih artikala, kamera i posebnih efekata. Međutim, *GameObject* sam po sebi ne može učiniti ništa. Potrebno mu je dati svojstva koja ga definiraju i čine ga onim što on je. Ovisno o tipu objekta, dodaju mu se komponente. Postoji velik broj ugrađenih komponenti koje se mogu iskoristiti, a komponente se mogu i samostalno izgraditi koristeći *Scripting API*, programsko sučelje za stvaranje skripti koje opisuju ponašanje objekta. Slika 8.3 prikazuje komponente jednoga objekta (objekt upravljača iz stvorene aplikacije). [30]

Obavezna komponenta objekta je *Transform*. Ta osnovna komponenta određuje poziciju, rotaciju i povećanje, određene za sve tri prostorne osi. Sjajan aspekt kom-

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju



Slika 8.3 Komponente GameObject-a upravljača.

ponenti je njihova fleksibilnost koja je postignuta time što svaka komponenta ima neka svojstva koja mogu biti prilagođena, ili u tijeku izrade igre ili logikom skripti u tijeku izvršavanja igre. Postoje dvije vrste svojstva: vrijednosti i reference. Pisanje skripti je alat proširenja izvornih mogućnosti koje *Unity editor* nudi. Povezivanjem jedne na objekt, ona se pojavljuje u listi komponenti. To se događa zato jer se one „prevode“ u tip komponente kako bi mogle biti uključene. Putem *Inspector-a* se definiraju objekti izloženi djelovanju skripte i sve napisane funkcionalnosti tada mogu biti izvršene. [31]

## 8.2 SteamVR

*SteamVR* je softver koji razvijačima omogućuje stvaranje VR aplikacija za osobna računala. Podržava raznovrsnost *headsetova* uključujući *HTC Vive* i *Oculus Rift*. Njegova glavna ideja je to da razvijači mogu koristiti jedno programsko sučelje i stvoriti program koji se može koristiti uz bilo koji od popularnih VR *headsetova* za računalo. *SteamVR* priključak (eng. plugin) upravlja s tri glavna zadatka:

- učitavanje trodimenzionalnih modela za prikaz VR upravljača
- rukovanje ulaznim informacijama upravljača
- prepostavlja kako izgleda ruka koja koristi upravljač

Osim tih osnovnih zadaća, pruža neke primjere primjene da pomogne s početnom fazom izrade VR programa koji implementiraju najčešće oblike interakcije s virtualnim svijetom ili programskim sučeljem. [32]

## 8.3 Tijek implementacije

Za drugu, klijentsku, stranu aplikacije stvoren je *Unity* projekt. Dodana je nova scena u koju su postavljeni objekti stolica, stolova, ploče i zidova kako bi imitirali raspored učionice. *SteamVR* je prepoznat kao koristan alat za stvaranje ovoga projekta pa je tako i instaliran u projektni direktorij.

Osim vidljivih objekta scene, radi TCP komunikacije koja ne spada pod ponašanje nijednoga od njih, stvoren je dodatni objekt nazvan „*TcpClient*“.

Preduvjet ove datoteke, odnosno ono što on uključuje (eng. include), je skripta naziva *TcpSocket*. Ona je odgovarajući suprotni dio *sendData* funkciji iz prvog dijela aplikacije. Obična je, nenaslijedjena *C#* klasa koja je stvorena da sažima funkcionalnosti standardne *C#* knjižnice za TCP komunikaciju, ostavivši samo metode koje su ovdje potrebne. Za to je korištena *Socket* klasa iz *System.net namespace-a*. *System.net* je programski *namespace* koji nudi jednostavna sučelja za upotrebu mnogih protokola korištenih na mreži u današnje dane. Konstruktor *TcpClient-a* prima kao parametre vrijednost IP adrese i vrijednost *porta* i na temelju njih stvara *Socket* instancu. Zatim, metoda za upravljanje događajem TCP prijema, napravljena na

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju

temelju mehanizma delegacije kojega *C#* nudi je ponuđena, tako da ona klasa koja stvori instancu ove može biti obaviještena o dolasku nove poruke.

Klasa povezana s *TcpSocket* objektom naziva se *AvatarController* zbog toga što upravlja teksturom platna koje prikazuje avatar osobe. U programiranju *Unity* skripta, objekti se klasi kao parametri povezuju tako da se za željene objekte parametri postave u *public scope* (javni djelokrug varijable). *Unity* ovo povezivanje omogućuje putem grafičkog sučelja u *Editoru*. Jedina takva javna varijabla za *AvatarController* klasu je takozvani „gameFrame“. On je instanca *GameObject* klase za objekt platna iz scene. U *Awake* metodi je stvorena instanca prethodno opisane *TcpClient* klase, a njen delegirani događaj je povezan za *MessageReceived* metodu koja je definirana kasnije. Također, ovdje je definirana i putanja za direktorij u koji će se pohranjivati dobivene slike. U *Start* metodi, stvara se instanca *Unityjeve Texture2D* klase u BGRA32 formatu i povezana je kao tekstura *gameFrame-a*. BGRA32 je format korišten nekim kamerama. Svaki od plavog, zelenog, crvenog i prozirnog (*blue, green, red, alpha*) kanala pohranjeni su kao 8-bitne vrijednosti u rasponu od 0 do 1 poredani jedan do drugog u navedenom poretku. *MessageReceived* rukuje informacijama iz poruke, dopisuje ih na *buffer* varijablu i provjerava za svaki dio je li mu vrijednost „image\_end“. Ako je, to znači da je prijem jednog okvira završio i da cijeli sadržaj *buffera* može biti zapisan u slikovnu datoteku u naznačenom direktoriju koristeći *StreamWriter*. *StreamWriter* je implementacija *TextWriter-a* za pisanje slova u stream u određenom načinu kodiranja. Sadržaj *MessageReceived* klase prikazan je u isječku 8.1.

Isječak 8.1 Prijem i pohrana okvira

```
private void MessageReceived( byte[] message )
{
    if ( !IsEndOfMessage( message ,
        System . Text . Encoding . ASCII . GetBytes( "image_end" ) ) )
        // Poruka nije u cijelosti primljena
        messageBuffer . AddRange( message );
        return ;
    }
    else {
```

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju

```
// Poruka je u cijelosti primljena
message = RemoveEndOfMessage(message,
System.Text.Encoding.ASCII.GetBytes("image_end"));
messageBuffer.AddRange(message);
message = messageBuffer.ToArray();
messageBuffer.Clear();
}
if (!newTexReady)
{
    try
    {
        StreamWriter streamWriter =
new StreamWriter
("Assets/Slike/capturedImage.png");
recievedImage = new byte[message.Length];
message.CopyTo(recievedImage, 0);
streamWriter.BaseStream.Write(recievedImage,
0, message.Length);
streamWriter.Close();
    }
    catch (Exception e)
    {
        Debug.Log("Greska : " + e);
    }
    newTexReady = true;
}
}
```

Završno, osvježavanje tekture se izvršava u *Update* metodi. Ukoliko je *MessageReceived* koristeći odgovarajuću varijablu *boolean* tipa naznačio da je slika u cijelosti primljena, *Update* metoda pri toj prigodi osvježavanja, učitava i zatim primjenjuje sliku pročitanu s diska, koristeći *StreamReader*, a taj je isječak koda prikazan ispod (isječak 8.2).

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju

Isječak 8.2 Postavljanje primljene slike za teksturu objekta

```
void Update()
{
    if (newTexReady)
    {
        var bytes = default(byte[]);
        using (StreamReader streamReader =
new StreamReader("Assets/Slike/capturedImage.png"))
        {
            var memstream = new MemoryStream();
            var buffer = new byte[1024 * 1024 * 2];
            var bytesRead = default(int);
            while ((bytesRead =
streamReader.BaseStream.Read(buffer, 0,
buffer.Length)) > 0)
            {
                memstream.Write(buffer, 0, bytesRead);
            }
            bytes = memstream.ToArray();
        }
        _texture.LoadImage(bytes);
        _texture.Apply();

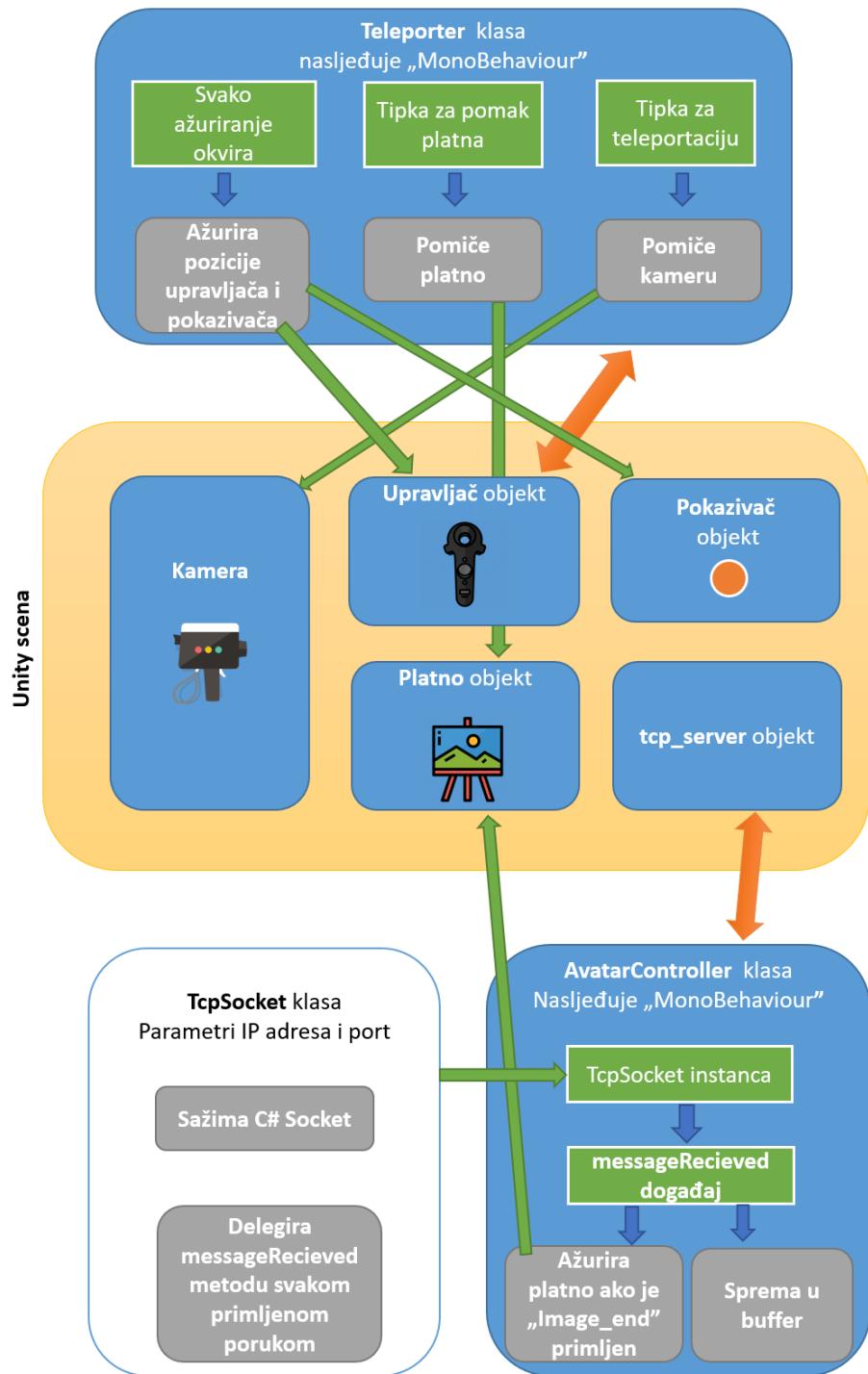
        Array.Clear(recievedImage, 0, 0);
        newTexReady = false;
    }
}
```

Zadnji korak je implementacija koda za upravljače. Objekt pokazivača (spomenuta narančasta kugla), objekt platna i tipke upravljača dodijeljene teleportiranju i pomicanju ekrana su uključeni kao parametri. *Awake* poziva „SteamVR\_Behaviour\_Pose“ od *SteamVR*-a koji pojednostavljuje upotrebu *pose* akcija. *Pose* je u *Unityju* spoj

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju

svojstava pozicije i rotacije u prostoru. Dodavanje tog poziva u *GameObject* će automatski ažurirati poziciju tog objekta, svaki *Update* kako bi *pose* odgovarao. Napredni izračuni brzine dobivaju se koristeći *buffer* koji pamti zadnjih 30 sekundi. Nadalje, *Update* osvježava položaj pokazivača koristeći sjedište zrake određene smjerom upravljača i poziciju prepreke određene tlom i također kontinuirano provjerava je li koja od tipki pritisnuta. Ako je neka pritisnuta, odgovarajuća funkcija se poziva. Te funkcije, *Teleport* i *MoveScreen* zadnji su blokovi ovoga koda. Oni funkcioniraju koristeći osnovne principe vektorske translacije nad vrijednostima pozicije pokazivača i trenutne pozicije kamere i platna. Efekt zatamnjenja za glatkoću tranzicije je efekt iz *SteamVR*-ovih mogućnosti. Kako je aplikacija posložena može se vidjeti u prikazu na slici 8.4 ispod. Značajne klase su prikazane kao plave (klase pridijeljene objektima, nasljednici *MonoBehaviour* klase) i bijele (nenaslijedene obične *C#* klase) kutije s njihovim logičkim operacijama prikazanima u obliku sivih kutija. *Unity* objekti su prikazani kutijama unutar velike narančaste kutije koja predstavlja scenu u kojoj su oni sadržani. Zelene strelice označuju reference iz klasa prema objektima, a narančaste obostrane predstavljaju koja je skripta povezana s kojim od objekata.

## Poglavlje 8. Implementacija softvera za udaljenu lokaciju



Slika 8.4 Arhitektura aplikacije na udaljenoj lokaciji.

# Poglavlje 9

## Zaključak

Ovaj rad pruža rješenje za daljinsku prisutnost korištenjem virtualnog okruženja. Napravljen je sustav koji postiže isto na primjeru jedne učionice, omogućujući time daljinsku prisutnost polazniku nekog kolegija, tečaja ili slično. Iako je osnovna primjena ove aplikacije ovaj specifični slučaj, ovaj projekt može služiti kao dokaz koncepta koji može biti upotrijebljen na bilo kojem slučaju u kojem je potrebna daljinska prisutnost.

Osnovni koncepti na kojima leži ova ideja su princip rada dubinske kamere i princip rada uređaja za virtualnu stvarnost. Ono što čini ovaj doživljaj puno kvalitetnijim od prisutnosti korištenjem video-konferencijskih alata je osjećaj uključenosti koji se virtualnom stvarnošću postiže. Veća uključenost dovodi do kvalitetnijeg praćenja sadržaja doživljaja što je kod slučaja predavanja iznimno bitna značajka. Bolja razina praćenja od strane studenta ili učenika je svakako presudna razlika koja bi trebala privlačiti korisnike da prisvoje ovaj način radije od uobičajenih.

Bez dubinske kamere ovaj koncept ne bi nikako bio moguć na istoj razini kvalitete iskustva. Kada praćena osoba ne bi bila izrezana od svoje pozadine (što bi bez mjere dubine bilo izrazito teško ili nemoguće) iluzija smještaja iste u prostor bi bila narušena i uklopila bi se jedino pri određenom kutu gledanja i pri savršenoj replikaciji okoline. Moguće rješenje pri korištenju obične kamere bi bilo korištenjem zelenoga platna iza predavača, što bi pak narušilo doživljaj „pravim“, fizički prisutnim sudionicima događaja. Dubinske kamere su tehnologija u zamahu i unutar nekoliko

## Poglavlje 9. Zaključak

godina postati će standardna komponenta mobilnih uređaja što će značiti da će uskoro veliki broj ljudi imati jednu pri ruci u svim trenutcima, bili oni toga svjesni ili ne.

Ideja ovog rada u budućnosti ima veoma široko područje za napredak i nadogradnju. Mogućnost dodavanja većeg broja kamera povećava zonu kretanja za metu, a može i ponuditi različite kute prikaza snimke za različita gledišta. Također, koristan sljedeći potez bio bi ugradnja zvučnog prijenosa koji bi se slao paralelno uz prijenos slike. Tada bi doživljaj bio potpun. Smještaj predmeta u prostoru napravljen je samo kao primjer, a za stvarni slučaj primjene to bi bilo poželjno čim više prilagoditi. Ili mjerenjem dimenzija i lokacija predmeta ili korištenjem nekih od metoda 3D *mappinga*. Stvaranje trodimenzionalnih modela na temelju vizualnih snimača je još područje ograničene razvijenosti, no pri upotrebi iskusnog korisnika može imati veoma zadovoljavajuće rezultate. Ugradnja te ideje u projekt značila bi da bi svakom novom lokacijom koju bi trebalo koristiti trebalo i napraviti snimak iste. Snimak bi se tada uvezao u Unity scenu i prilagodili bi se smještaj objekta platna i početni smještaj sudionika nakon čega bi doživljaj mogao početi.

Područje virtualne stvarnosti i virtualne prisutnosti discipline su sa širokim prostorom za napredak. U budućnosti, napretkom korištene opreme napredovat će i sama iskustva pružana korisnicima takvih sustava, a granica je samo razina uvjerljivosti koju čovjeku pruža i stvarni svijet u kojemu se nalazi. Porastom kvalitete znatno će porasti i učestalost primjene tih tehnologija u raznim granama znanosti, industrije, edukacije i rekreacije. Isto tako porasti će i dostupnost što će značiti da će velikom broju ljudi biti moguće iskusiti sve čari i koristi ovoga iskustva.

# Bibliografija

- [1] „Virtual Reality“, s Interneta, [https://en.wikipedia.org/wiki/Virtual\\_reality](https://en.wikipedia.org/wiki/Virtual_reality), 7. studenog 2019.
- [2] Britta O’Boyle i Adrian Willings „What is VR?“, s Interneta, <https://www.pocket-lint.com/ar-vr/news/136540-what-is-vr-virtual-reality-explained>, 16. svibnja 2019.
- [3] „History of Virtual Reality“, s Interneta, <https://www.vrs.org.uk/virtual-reality/history.html>, 2017.
- [4] „History of VR/AR and impact on our future“, s Interneta, <https://www.sutori.com/item/1961-headsight>, 11. studenog 2019.
- [5] „Virtual reality applications“, s Interneta, [https://en.wikipedia.org/wiki/Virtual\\_reality\\_applications](https://en.wikipedia.org/wiki/Virtual_reality_applications), 21. listopada 2019.
- [6] „Virtual reality - health and safety“, s Interneta, [https://en.wikipedia.org/wiki/Virtual\\_reality#Health\\_and\\_safety](https://en.wikipedia.org/wiki/Virtual_reality#Health_and_safety), 7. studenog 2019.
- [7] Charles D. Huston, Chris Coleman: “System and method for creating and sharing a 3D virtual model of an event”
- [8] Knutzen, B., & Kennedy, D.: „The global classroom project: Learning a second language in a virtual environment“. The Electronic Journal of e-Learning, 10(1), 90-106, 2012.
- [9] Vivek Reddy: „Understanding how depth sensing cameras work“, s Interneta, <http://www.vivekc.com/understanding-how-depth-sensing-cameras-work/>, 24. svibnja 2015.
- [10] „How the Kinect Depth Sensor Works“, s Interneta, <https://www.youtube.com/watch?v=uq9SEJxZiUg>, 16. veljače 2013.

## Bibliografija

- [11] Andrew McWilliams: „How depth sensor works“, s Interneta, <https://jahya.net/blog/how-depth-sensor-works-in-5-minutes/>, 6. kolovoza 2013.
- [12] „Multi-camera smartphones: Benefits and challenges“, s Interneta, <https://www.dxomark.com/multi-camera-smartphones-benefits-and-challenges/>, 21. veljače 2017.
- [13] „Depth sensors are the key to unlocking next level computer vision applications.“, s Interneta, <https://blog.cometlabs.io/depth-sensors-are-the-key-to-unlocking-next-level-computer-vision>, 18. srpnja 2017.
- [14] „Use Cases – Intel RealSense Depth and Tracking Cameras“, s Interneta, <https://www.intelrealsense.com/use-cases/>
- [15] „Astra Series – Orbbec“, s Interneta, <https://orbbec3d.com/product-astra-pro/>, 11. studenog 2019.
- [16] „Unity (game engine)“, s Interneta, [https://en.wikipedia.org/wiki/Unity\\_\(game\\_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)), 11. studenog 2019.
- [17] „HTC Vive“, s Interneta, [https://en.wikipedia.org/wiki/HTC\\_Vive](https://en.wikipedia.org/wiki/HTC_Vive), 25. listopada 2019.
- [18] „OpenGL Overview“, s Interneta, <https://www.opengl.org/about/>, 11. studenog 2019.
- [19] „About – OpenCV“, s Interneta, <https://opencv.org/about/>, 11. studenog 2019
- [20] „Astra user guide v.2.0.17.“, dio Orbbec Astra razvojnog paketa dostupnog na <https://orbbec3d.com/develop/>
- [21] „What is a socket?“, s Interneta, <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>, 11. studenog 2019.
- [22] „TCP/IP Ports and Sockets Explained“, s Interneta, <http://www.steves-internet-guide.com/tcpip-ports-sockets>, 12. svibnja 2019.
- [23] „List of TCP and UDP port numbers“, s Interneta, [https://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers), 7. studenog 2019.
- [24] „Transmission Control Protocol“, s Interneta, [https://en.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](https://en.wikipedia.org/wiki/Transmission_Control_Protocol), 3. studenog 2019.

## *Bibliografija*

- [25] „Starting Unity for the first time“, s Interneta, <https://docs.unity3d.com/Manual/GettingStarted.html>, 11. studenog 2019.
- [26] „Asset workflow“, s Interneta, <https://docs.unity3d.com/Manual/AssetWorkflow.html>, 11. studenog 2019.
- [27] „Project manifest“, s Interneta, <https://docs.unity3d.com/Manual/upm-manifestPrj.html>, 11. studenog 2019.
- [28] „The main windows“, s Interneta, <https://docs.unity3d.com/Manual/UsingTheEditor.html>, 11. studenog 2019.
- [29] „Scenes“, s Interneta, <https://docs.unity3d.com/Manual/CreatingScenes.html>, 11. studenog 2019
- [30] „GameObject“, s Interneta, <https://docs.unity3d.com/Manual/class-GameObject.html>, 11. studenog 2019
- [31] „Transform“, s Interneta, <https://docs.unity3d.com/Manual/class-Transform.html>, 11. studenog 2019
- [32] „SteamVR Plugin - Asset store“, s Interneta, <https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647>, 11. studenog 2019