SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1374

PROGRAMSKA PODRŠKA ZA VIRTUALNU SJEDNICU

Renato-Zaneto Lukež

Zagreb, lipanj 2017

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 3. ožujka 2017.

DIPLOMSKI ZADATAK br. 1374

Pristupnik: **Renato-Zaneto Lukež (0036458349)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Programska podrška za virtualnu sjednicu**

Opis zadatka:

Napraviti programsku podršku za ostvarivanje virtualne sjednice koja sadrži scenu zajedničkog stola i nekoliko modela osoba koji umreženo sudjeluju na sjednici. Modele osoba ostvariti skeniranjem stvarnih osoba. Pomoću uređaja Kinect pokrete stvarne osobe prenositi na virtualne modele koji sudjeluju na sjednici. U okviru zajedničkog rada na sjednici omogućiti glasovanje sudionika postupkom prepoznavanja gesti dizanja ruke odnosno pritiska virtualne tipke na stolu. Na različitim primjerima prikazati ostvarene rezultate. Načiniti ocjenu rezultata i implementiranih algoritama.
Izraditi odgovarajući programski proizvod. Koristiti programski jezik C#, programski alat Unity i za uređivanje modela objekata programski alat Autodesk 3DS Max. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 29. lipnja 2017.

Mentor:

Prof. dr. sc. Željka Mihajlović

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Zahvala

"Ovim putem javno bi se zahvalio svojim roditeljima koji su mi pružili financijsku i emocionalnu potporu prilikom studiranja na Fakultetu elektrotehnike i računarstva.

Svojoj mentorici prof. dr. sc. Željki Mihajlović na pomoći, savjetima, strpljenju i podršci kroz sve godine mojeg studiranja.

Tvrtki Libusoft Cicom na posudbi računala, te kolegi Ognjenu Bujanu na pomoći pri odabiru teme za izradu diplomskog rada.

Kolegi Marku Nađu na pomoć i savjetima pri izradi modela postupkom 3D skeniranja.

Svojim bliskim prijateljima, rodbini, kolegama i suradnicima koji su mi uvijek pružali pomoć i potporu kada mi je ona bila najpotrebnija.

Svim svojim učiteljima, profesorima, asistentima i demonstratorima uz čiju sam pomoć uspio postati fakultetski obrazovana osoba i što su mi pomogli pri ostvarenju mog životnog sna.

Iz dubine svog srca Vam se zahvaljujem!"

Sadržaj

1. Uvod

Ovaj diplomski rad nastavak je na završni rad [1], gdje se naglasak stavlja na ostvarivanje virtualne sjednice na mreži. Cilj ovog diplomskog rada je proširiti funkcionalnost povezivanja uređaja Oculus Rift i Kinect na mrežu. Ideja je napraviti aplikaciju za više korisnika u kojoj će korisnici moći aktivno sudjelovati u sjednici kao da se nalaze u stvarnom prostoru, kroz praćenje, govor, slušanje i glasovanje, a korisnike će u virtualnom prostoru predstavljati njihov vlastiti model.

Jedan od glavnih razloga implementacije ovakve aplikacije je vidjeti napredak tehnologije virtualne stvarnosti i mogućnost ostvarivanja takvog sustava. Drugi razlog jest pokušati stvoriti aplikaciju na kojoj će korisnici moći održavati poslovne sastanke ili opuštena druženja uz prijatelje. Prednosti ovakve aplikacije su ušteda vremena i novaca na poslovna putovanja, manja potrošnja goriva, lakša interakcija i izražavanje korištenjem govora tijela i gesti, svi članovi unutar jedne organizacije mogu biti prisutni na svakom sastanku bez obzira na koliko je gradova organizacija podijeljena i drugo.

Kako bi se uspostavila virtualna sjednica na mreži potrebno je zadovoljiti nekoliko ključnih komponenti za ostvarivanje ovakvog sustava. Prvo što je potrebno jest ostvariti virtualni prostor na mreži na koji se korisnici takvog sustava mogu spojiti. Ujedno je potrebno implementirati način spajanja na takav sustav, te u takvom virtualnom prostoru stvoriti realističan reprezentativni model korisnika za sudjelovanje u virtualnoj sjednici. Također, potrebno je implementirati sustav za prijenos podataka pokreta, glasovnih i zvučnih podataka, te algoritma za interakciju u virtualnom prostoru. Uz sve to još je potrebno implementirati korištenje uređaja Oculus Rift i Kinect u zajedničkoj sceni preko mreže u stvarnom vremenu. Implementacija ovakvog sustava opisana je u nastavku diplomskog rada.

Za izradu ovog diplomskog rada korišteni su uređaji Oculus Rift DK2 koji služi za prikaz virtualne scene, uređaj Kinect za XBox One koji služi za upravljanje modelom, programski alat Unity verzija v5.6.2 za izradu aplikacije, programski alat Photon Unity Network za uspostavljanje višekorisničke aplikacije na mreži, te dodatak Photon Voice za prijenos glasovnih i zvukovnih podataka. Uređaj za skeniranje 3D modela KScan3D, programi za obradu modela Autodesk 3DS Max i Blender. Cijela aplikacija napravljena je na operacijskom sustavu Windows 10.

2. Tehnološka podrška

2.1 Kinect

Uređaj koji se koristi za implementaciju praćenja ljudskih pokreta u svrhu ovog diplomskog rada je Kinect za XBox One. Slika 1. prikazuje izgled uređaja. Inovacijske tehnologije iza ovog uređaja kombinacija su sklopovlja i programske podrške unutar Kinect senzora koji kroz prirodno sučelje korisniku omogućuje korištenje pokreta, gesti i glasovnih naredbi kao ulazne jedinice. Glavni trojac sklopovlja bez kojeg ovaj uređaj ne bi funkcionirao su:

- VGA video kamera u boji - Video kamera pomaže u raspoznavanju izraza lica i drugih dodataka pri očitanju tri boje: crvena, zelena i plava, bolje poznata kao RGB kamera.
- Dubinski senzor - Infracrveni projektor i monokromatski CMOS senzor rade u paru kako bi "vidjeli" sobu u tri dimenzije, bez utjecaja svjetlosnih komponenti.
- Skup mikrofona - Polje od 4 mikrofona koji mogu izolirati glasove korisnika od buke prostora, što korisniku omogućuje da ostvaruje interakciju nad uređajem sa svega nekoliko metara udaljenost uz korištenje glasovnih naredbi.

Slika 1. - Uređaj Microsoft Kinect

Kroz malo dublji pogled u tehničke specifikacije uređaja može se vidjeti da video kamera i dubinska kamera koriste rezoluciju od 640x480p, te obje kamere imaju vrijeme osvježavanje ekrana od 30 FPS-a (okvira u sekundi). Specifikacije također navode da se za optimalnu upotrebu korisnik treba nalaziti na oko 1,8m udaljenosti od uređaja, međutim to može malo odstupati uzevši u obzir položaj na kojem je uređaj smješten. Kako bi se podatak o udaljenosti korisnika od uređaja očuvao, postolje uređaja je zapravo motor koji pomiče glavu Kinect senzora za +/-30° u vertikalnom smjeru. Isto tako motor je zadužen i za povećanje prostora pogleda kamere u odnosu na blizinu korisnika. Na taj način uređaj se može podesiti na bilo koju razinu i to ga čini univerzalnim i pogodnim za bilo koji prostor.

Razlog zbog čega je ovaj uređaj idealan za korištenje je zbog dodjele i obrade skeleta nad korisnikom u stvarnom vremenu. Uz potporu strojnog učenja uređaj je upoznat s barem 200 uobičajenih poza koje korisnik može napraviti. Uzorkovanje poza rađeno je nad različitom skupinom ljudi koji se razlikuju po dobi, visini, širini, brzini pokreta tijela, brzini pokreta zglobova i slično. Tako uređaj može predvidjeti korisnikove pokrete i upotpuniti praznine kod pokreta koji ometaju pogled kamere na korisnikovu skeletnu strukturu. Isto tako sposoban je prepoznati korisnika neovisno nosi li osoba vrećastu odjeću, prelazi li osobi dužina kose preko ramena ili bilo što, što korisniku može izmijeniti građu tijela.

Način rada ovog uređaja prikazuje Slika 2. Uređaj izgrađuje dubinsko polje izbacivanjem laserske zrake preko cijelog polja kretanja korisnika, te tako razlikuje okolni prostor od samog korisnika. Nakon što se zrake sudare s okolnim prostorom projiciraju se nazad u uređaj, te se pomoću dobivenih podataka, koji su očitani kao sudari okolnog prostora i infracrvene zrake izmjerene u različitim bojama, određuje gdje se nalazi korisnik. Ovisno o udaljenosti korisnika, tako će biti dodijeljene boje pikselima. Tako su korisnici uobičajeno prikazani crvenom bojom, dok je okolni prostor prikazan sivom

bojom. Jednom kada se korisnik odredi, dodjeli mu se skelet i korisnik može koristiti Kinect senzor kao ulaznu jedinicu.

Slika 2. - Način rada Microsoft Kinecta

Kinect senzor koristit će se u svrhu 3D skeniranja i izgradnje stvarnih modela korisnika, te upravljanja izgrađenog modela unutar virtualne sjednice, što je opisano u nastavku ovog diplomskog rada.

2.2 Oculus Rift

Uređaj koji se koristi za prikaz virtualne stvarnosti i vizualizaciju virtualne okoline u svrhu ovog diplomskog rada je Oculus Rift. To je uređaj koji detektira korisnikove pokrete glave i prikazuje virtualnu scenu u stvarnom vremenu. Izgled uređaja prikazuje Slika 3. Stavljanjem radne stanice na glavu (engl. headset) korisnikovo vidno polje zamjeni se digitalnom slikom. Pomakom glave mijenja se korisnikova perspektiva unutar virtualne scene. Magija ovog uređaja, zbog kojeg se korisniku stvara osjećaj znanstvene fantastike, skrivena je u sklopovlju uređaja koji se sastoji od:

- OLED ekran - Ekran na kojem se korisniku prikazuje i
- Uređaj za praćenje stvarnosti (engl. Adjacent Reality Tracker) - Dio sklopovlja sastavljen od žiroskopa, akcelerometra i magnetometra koji u zajedničkom radu prate radnu stanicu kroz sve tri dimenzije.
- Lokator - Uređaj odvojen od same stanice koji prati niz infracrvenih LED dioda ugrađenih po cijeloj stanici kako bi korisniku omogućio perspektivu od 360° u svim smjerovima unutar virtualnog prostora.

Slika 3. - Uređaj Oculus Rift

Pogledom na tehničku specifikaciju OLED ekran ima rezoluciju od 1080x1200p po oku promatrača, vrijeme osvježavanja od 30 FPS-a i vidni kut od 100°. U samom uređaju ugrađeni su zvučnici koji generiraju trodimenzionalni učinak. Prijenos slike ostvaruje se preko HDMI kabla, a prijenos podataka i napajanja preko USB 2.0 priključka. Sama slika koja se prikazuje korisniku na ekranu su zapravo dvije odvojene dvodimenzionalne slike koje su na uglovima djelomično zakrenute. Kada korisnik gleda u ekran, ovim postupkom korisnikove oči se zavaraju i korisnik misli da gleda u jednu veliku trodimenzionalnu sliku, a zapravo gleda u dvije odvojene dvodimenzionalne slike. Korisnik gleda na ekran preko dvije ugrađene leće koje korisniku uvećaju ekran i popune vidno polje. Na taj način korisnikove

oči gledaju drukčije dijelove ekrana i time se stvara trodimenzionalna stereoskopska pojava.

Zajedničkim radom ovakav sustav prenosi veliki količinu podataka između radne stanice i računala, te time ostvaruje vrlo precizno virtualno iskustvo.

Ono što je bitno napomenuti je programska podrška koja se koristi za ovaj sustav. Trenutna javna verzija je SDK 1.6. Osobe koje odluče kupiti uređaj Oculus Rift ili odluče raditi projekt s Oculus Rifotm neka obavezno provjere da li njihovo računalo podržava rad s Oculus Riftom. Na stranici samog uređaja postoji program za provjeru konfiguracije. Pokretanjem programa za provjeru konfiguracije i njegovim izvršavanje dobiva se popis svih komponenti koje zadovoljavaju i ne zadovoljavaju specifikacije za Oculus Rift. Isto tako postoji popis preporučenih i minimalnih specifikacija za ispravan rad Oculus Rifta. Kako bi uređaj radio s optimalnom izvedbom, Tablica 1. prikazuje popis preporučenih specifikacija.

Tablica 1. - Popis preporučenih specifikacija za optimalan rad uređaja

Preporučene specifikacije	
Grafička kartica	NVIDIA GTX 1060 / AMD Radeon RX 480 ili veći
Alternativna grafička kartica	NVIDIA GTX 970 / AMD Radeon R9 290 ili veći
CPU	Intel i5-4590 jednak ili veći
Memorija	8GB+ RAM
Izlazni video	Kompatibilni HDMI 1.3 video izlaz
USB priključak	3X USB 3.0 priključak, 1X USB 2.0 priključak
Operativni sustav	Windows 7 SP1 64bit ili noviji

U slučaju da korisnik nije u stanju nabaviti sve komponente za optimalnu izvedbu, Tablica 2. sadrži popis minimalnih specifikacija potrebnih za rad uređaja.

Tablica 2. - Popis minimalnih specifikacija za rad uređaja

Minimalne specifikacije	
Grafička kartica	NVIDIA GTX 1050 Ti/ AMD Radeon RX 470 ili veći
Alternativna grafička kartica	NVIDIA GTX 960 4GB/ AMD Radeon R9 290 ili veći
CPU	Intel i3-6100 / AMD FX4350 ili veći
Memorija	8GB+ RAM
Izlazni video	Kompatibilni HDMI 1.3 video izlaz
USB priključak	1X USB 3.0 priključak, 2X USB 2.0 priključak
Operativni sustav	Windows 8.1 ili noviji

3. Programska podrška

3.1 Photon Engine

Programski alat Photon Engine je brz, pouzdan i fleksibilan okvir (engl. framework) za razvoj višekorisničkih aplikacija u stvarnom vremenu. Photon Engine sastoji se od poslužitelja i nekoliko klijentskih alata za razvoj programa za sve veće platforme. Ovaj programski alat stvorila je grupa Exit Games i time pružala bezbrižnu uslugu za višekorisničke aplikacije.

Photon Engine nudi razvojne pakete koji uključuje biblioteke, programske isječke i dokumentaciju za različite programske platforme poput programskog alata Unity, operacijskih sustava iOS, Android, Windows Phone, Corona, Marmelade, Cocos2d-x i drugi.

Budući da se sve biblioteke povezuju na istu pozadinu koristeći iste naredbe i istu logiku, moguća je među platformska interakcija, npr. komunikacija između različitih verzija aplikacije pokrenutih na različitim platformama.

Glavne prednosti programskog alata Photon uključuju:

- Mogućnost među platformske interakcije - klijentske razvojne pakete za sve veće platforme za igranje računalnih igara
- Globalno nisku latentnost - Programski alat Photon razmješten je u centrima usluga za poslužitelje diljem SAD-a, Europe i Azije. Uzevši u obzir ovu činjenicu, njegova infrastruktura omogućuje programerima izdavanje računalnih igara u stvarnom vremenu na globalnoj razini bez većih poteškoća s poslužiteljima.
- Stvarno vrijeme - komunikacijski protokoli su aerodinamični i srž poslužitelja je zasnovana na C/C++ programskim jezicima kako bi omogućili najbrže povratno vrijeme za višekorisničke aplikacije u međusobnoj interakciji s globalnim poslužiteljskim centrima.

- Automatska skalabilnost - automatski se skalira sa stvarnim opterećenjem aplikacije, zbog koje može podržavati na desetke tisuća korisnika.
- Uparivanje sučelja programskih aplikacija - brzim i jednostavnim radom povezuje igrače nasumce po odabranim kriterijima ili stvara listu slobodnih soba i omogućuje korisniku da odabere jednu od ponuđenih
- Besplatni plan - programski alata potpuno je besplatan za do 20 istovremenih korisnika po aplikaciji, te se može napraviti neograničen broj aplikacija. Za potrebe više korisnika, može se nadograditi na plan koji se plaća i koji se u bilo koje vrijeme može vratiti na besplatni ili nadograditi bez gubitka korisnika.

Dodaci koje programski alat Photon nudi su:

- Realtime - Potpuno upravljan servis (Program kao servis) s Photon prostornim poslužiteljima pokrenutim u raznim regijama diljem svijeta, koji je spreman podržati nisko latentne višekorisničke aplikacije.
- Photon Unity Networking (PUN) - Unity paket za višekorisničke aplikacije koji pruža opcije ovjere autentičnosti, uparivanja i brze, pouzdane komunikacije unutar aplikacije kroz Photon pozadinu.
- Photon TrueSync - Višekorisnički paralelni sinkronizacijski računarski sustav (engl. lockstep) izgrađen na Photon Unity Networking dodatku.
- Bolt - Moćno umreženo rješenje za programski alat Unity koji se sastoji od apstraktnih kompleksnih mrežnih dodataka koji se kriju iza jednostavnih i lako uporabljivih sučelja.
- Thunder - Dodjeljuje servis za uparivanje, proboj i prijenos višekorisničkih aplikacija.
- Photon Chat - Komunikacijska aplikacija za razmjenu poruka među korisnicima.
- Photon Voice for PUN - Dodatak za Photon Unity Networking koji dodaje prijenos glasa i zvuka kao Photon Unity Networking dodatak.

- Photon Server - Socket poslužitelj u stvarnom vremenu i alat za među platformska višekorisničku aplikaciju, koji je veoma brz i jednostavan za korištenje.

Za korištenje bilo kojeg dodatka, potrebno se registrirati na stranicu [4]. Registracija je potpuno besplatna i prilikom stvaranja bilo koje aplikacije, potrebno je odabrati dodatke koji će se koristiti u aplikaciji. Jednom kada se dodaci odaberu, potrebno im je dodijeliti ime aplikacije i kratki opis same aplikacije kako bi se dobili identifikatori dodatka aplikacije (engl. AppID) koji se kasnije koriste za poziv poslužitelja.

3.2 Photon Unity Networking (PUN)

Photon Unity Networking (PUN) je rješenje visoke razine specificirano za programski alat Unity koje upravlja uparivanjem, jednostavnim povratnim pozivima, komponentama za sinkronizaciju objekata u igri (engl. Game Object), udaljenim pozivima procedura (engl. Remote Procedure Calls, RPC) i sličnim dodacima koji pružaju dobar početak za izradu višekorisničke aplikacije. Izvan ovog opisa Photon Unity Network je kruto, opsežno aplikacijsko programsko sučelje (engl. API - Application programming interface) s većom kontrolom upravljanja mrežnih interakcija. Ovaj dodatak podržava bilo koji tip višekorisničke aplikacije. Kompatibilan je s uslugom Photon oblak (engl. Photon Cloud) koji upravlja Photon poslužiteljima (engl. Photon Sevrer) umjesto samog korisnika.

3.2.1 Usporedba Photon Unity umrežavanja s Unity umrežavanjem

Korištenjem programskog alata Photon s programskim alatom Unity postoje neke prednosti koje se mogu iskoristiti kada se Photon koristi kao posrednički program (engl. middleware) umjesto običnog Unity umrežavanja.

Razlike:

- Model poslužitelja usluge (engl. host model) - Unity Networking temeljen je na odnosu poslužitelj-klijent (engl. server-client). Bitno je naglasiti da to nije komunikacija isti s istim odnosno svaki sa svakim (engl. peer-to-peer). Poslužitelji su pokrenuti preko Unity klijenta (preko jednog od korisnika). Photon Unity Networking također je temeljen na odnosu poslužitelj-klijent, no on ima definiranog određenog poslužitelja (engl. dedicated server). Tako ne dolazi do odbačenih veza zbog odlaska poslužitelja usluge.
- Povezanost (engl. connectivity) - Unity Networking radi s probijanjem prevoditelja mrežne adrese (engl. NAT - Network addess translation) kako bi poboljšao povezanost. Razlog tome je što korisnici poslužuju mrežne poslužitelje, te zbog tog povezanost često zataji zbog vatrozida/umrežitelja i slično. Povezanost nikad ne može biti zagarantirana i ima nisku stopu uspješnosti. Photon Unity Networking ima određenog poslužitelja, tako da nema potrebe za tehnologijom koju koristi Unity Networking ili potrebe za sličnim idejama. Povezanost je 100% zagarantirana i ako (u rijetkim slučajevima) veza zataji, međutim razlog tome može biti stroga umrežena veza poslužitelja i klijenta (npr. poslovni VPN).
- Izvedba - Photon Unity Newtorking pobjeđuje Unity Networking po izvedbi, iako nema podataka koji to trenutno mogu dokazati. Međutim, biblioteke se redovito optimiziraju već godinama. Također, budući da su Unity poslužitelji temeljeni na korisničkim poslužiteljima, latencija je obično lošija zbog toga što ovisi o povezanosti korisnika koji se ponaša kao poslužitelj. Ove veze nikad nisu bolje od povezanosti s određenim Photon poslužiteljem.
- Cijena - Kao i rješenje koje pruža Unity Networking, Photon Unity Networking također nudi besplatni dodatak. Potrebno se prijaviti na njihovu stranicu kako bi se mogao koristiti Photon oblak za usluge posluživanja. Drugi pristup bi bio najam vlastitog poslužitelja na kojem bi se Photon izvršavao. Besplatna licenca dozvoljava do 100

konkurentnih korisnika. Druge licence se plaćaju samo jednom (ako se koristi vlastiti poslužitelj) i podiže maksimalno ograničenje konkurentnih korisnika.

- Dodatci i održavanje - Unity ne pridodaje neku posebnu važnost implementaciji umreženja. Rijetko postoje poboljšanja i ispravke grešaka unutar koda. Photon se aktivno održava i dijelovi su dostupni s izvornim kodom. Također, Photon nudi više dodataka od Unity-a, poput ugrađenog upravljanja opterećenjem i izvan mrežni način rada.
- Glavni poslužitelj - Glavni poslužitelj za Photon Unity Networking malo je drukčiji od glavnog poslužitelja za običan Unity Networking. Photon poslužitelj je onaj koji postavlja imena soba aktivnih igara u takozvanim predvorjima (engl. lobbies). Kao Unity-ev glavni poslužitelj, proslijedit će klijente na poslužitelj igre (engl. game server), gdje se sama igra odvija.

Još neke dodatne usporedbe i karakteristike usporedbe Photon Unity umrežavanja i običnog Unity umrežavanja prikazuje Tablica 3.

Tablica 3. - Pregled razlika Photon Unity umrežavanja i Unity umrežavanja

	PUN	UNet	
Cijena	Besplatno	Besplatno	
Besplatna upravljačka jedinica kamere	20	Beskonačno	
Posluživanje na svjetskoj mreži	Da	Ne	UNet potreban glavni poslužitelj
Kompatibilno s Unity umreženjem	Da	Da	Staro Unity umrežavanje (ne Unet)
Unity FREE: Web, Samostalan	Da	Da	
Unity 4 FREE: iOS, Android	Ne	Da/Ne	UNet - problemi kod probijanja na mobitelima
Unity 5 FREE: iOS, Android	Da	Da/Ne	UNet - potrebne Relay usluge za mobitele
Selidba poslužitelja	Nije ugrađeno	Ne	
Sobe, Podrška za predvorje	Da	Ne	

3.2.2 Izvan mrežni način rada

Izvan mrežni način rada dodatni je način rada koji omogućuje ponovno korištenje višekorisničkog koda u načinu rada pojedinog korisnika. Najčešći dodatak koji se koristi u pojedinačnom načinu rada je slanje proceduralnih poziva (engl. remote procedure call - RPD) i korištenje naredbe za dodavanje novog objekta u scenu. Glavni cilj izvan mrežnog načina rada je onemogućiti NULL reference i druge greške kod korištenja funkcionalnosti Photon umrežavanja dok korisnik nije povezan s mrežom. I dalje je potrebno uzeti u obzir činjenicu da se odvija pojedinačni način rada zbog postavki aplikacije i sličnih razloga. Međutim, dok je aplikacija pokrenuta, cijeli kod treba biti ponovno upotrebljiv. Ručno je potrebno omogućiti izvan mrežni način rada kako bi Photon umreženje razlikovalo normalan rad aplikacije od rad aplikacije uzrokovanog greškom. Postavljanjem opcije izvan mrežnog načina rada ponovno se mogu iskoristiti neke metode koje su namijenjene za višekorisnički način rada bez stvaranja ikakvih veza i grešaka.

3.2.3 Ograničenja vezana za poglede i korisnike

Zbog izvedbe programskog sučelje aplikacije za Photon umreženje podržava se do 1000 Photon gledišta (engl. Photon Views) po korisniku i maksimum od 2,147,483 korisnika. Lako se može nadodati više Photon gledišta po korisniku, ali po cijeni maksimalnog broja korisnika.

Razlog tome je što Photon gledište šalje svoj identifikator gledišta (engl. View ID) za svaku mrežnu poruku. Taj identifikator gledišta je tip podatka *integer*, koji se sastoji od identifikatora korisnika (engl. player ID) i korisnikovog identifikatora gledišta. Maksimalna veličina tipa *integer* je 2.147.483.647, podijeljeno s maksimalnim brojem identifikatora gledišta (koje je 1000) dozvoljava 2.000.000 korisnika unutar jedne aplikacije, gdje svaki ima po 1000 identifikatora gledišta. Kao što se vidi, lako se poveća broj korisnika smanjenjem maksimalnog broja identifikatora gledišta. Suprotno od

toga, korisniku se može dodijeliti više identifikatora gledišta po cijeni smanjenja broja identifikatora korisnika.

Važno je napomenuti da većina aplikacija nikad ne treba više od nekoliko identifikatora gledišta po korisniku (jedan do dva po korisniku). Neka se za primjer uzme akcijska igra u kojoj korisnik ispucava metke na druge korisnike sa svrhom da pobjedi igru. Nije efikasno dodijeliti Photon gledište i identifikator modela svakom metku koje korisnik ispuca za ažuriranje pozicije u sceni, te se umjesto toga preporučuje praćenje pozicije metaka preko igračevog Photon gledišta. Tako se koriste svega jedan do dva Photon gledišta po korisniku, te se tako može osigurati veći broj korisnika u jednoj igri. U sklopu ovog rada bitno je znati da se može dodijeliti više Photon gledišta, a razlog tome detaljno je opisan u nastavku rada.

Također ima mjesta za napredak pri poboljšanju izvedbe propusnog pojasa (engl. bandwidth) i to smanjenjem tipa podataka *integer* na tip podatka *short* (-32.768, 32.768). Podešavanjem maksimalnog broja identifikatora gledišta na 32, još uvijek se može podržati 1023 korisnika unutar jedne aplikacije. Također, programsko sučelje aplikacije trenutno ne koristi *unsigned integer* / *unsigned short* tip podatka, nego isključivo koristi pozitivne vrijednosti. To je tako zbog jednostavnosti tipa podatka i zbog toga što uporaba identifikatora gledišta nije presudan izvedbeni problem u većini slučajeva. U sklopu ovog rada bitno je znati da je to moguće zbog optimizacije, jer broj Photon gledišta koji će se koristiti u ovom radu bit će oko 20 po korisniku, što taman odgovara navedenoj brojci. Smanjenje tipa podataka neće utjecati na smanjenje broja korisnika, a to je bitno u slučaju da se jednog dana želi povećati maksimalni broj mogućih korisnika.

3.2.4 Ograničenja vezana za grupiranje i djelokrug

Dodatak Photon umreženja još ne podržava stvarne mrežne grupe (engl. network groups) i djelokrug (engl. scoping). Iako Unity dodatak

djelokruga još nije implementiran, mrežne grupe implementirane su isključivo na klijentskoj strani. Tako bilo koja vrsta udaljenog proceduralnog poziva bit će ignorirana zbog grupiranja i bit će odbačena nakon što je primljena. Na taj način grupe rade, ali ne štede na propusnosti.

3.3 Dodatak Photon Voice

Photon Voice je dodatak za Photon Unity Network koji donosi glasovni i zvučni prijenos podataka na Photon mrežu. Photon Voice ovisi o dodatku Photon Unity Network i zahtjeva zasebnu Photon Voice aplikaciju. Da bi se Photon Voice mogao koristiti potrebno ga je uvesti u Unity projekt gdje je Photon Unity Network već prisutan i konfiguriran.

Kada je Photon Unity Network klijent u stanju povezanosti unutar jedne Photon Unity Networking sobe, Photon Voice moguće je povezati ili odspojiti aplikaciju na Photon oblak. Osim toga, korisnicima ne trebaju nužno imati interakciju s Photon Voice dodatkom. Photon Voice klijent automatski rukuje s glasovnim i zvučnim podacima tijekom rada veze. Stvorit će glasovnu sobu (engl. voice room) s istim imenom na koju je spojen, tj. ime Photon Unity Networking sobe.

Jednom kada se Photon Voice dodatak uključi u projekt, postavke za Photon Voice bi se trebale pojaviti u postavkama Photon poslužitelja. Tamo je potrebno unijeti identifikator aplikacije za Photon Voice dodatak koja se dobije stvaranjem aplikacije na Photon stranici, slično kao što je napravljeno ranije za Photon Unity Network.

3.3.1 Prepoznavanje glasa

Predefinirani glas dodatna je mogućnost koja će filtrirati snimljeni zvuk i transmitirati ga samo onda kada se prekorači predefinirani prag signalnog nivoa. To znači da glasovni prijenosi automatski zaustavlja kada korisnik

prestane govoriti ili proizvoditi zvuk i ponovno se pokreće kada korisnik ponovno počne govoriti ili proizvoditi zvuk. Ova mogućnost pomaže pri izbjegavanju slanja beskorisnih šumova i time smanjuje zagušenje kod propusnosti mreže.

3.3.2 Glasovna kalibracija

Ako korisnik ima smetnje za vrijeme snimanja vlastitog glasa iako je detekcija glasa uključena, korisniku je onda vrlo vjerojatno potrebna kalibracija glasa. Photon Voice pruža automatsko kalibracijsko stanje koje je ograničeno vremenom. Zadana vrijednost je 2000 milisekundi, no korisnik po potrebi to može promijeniti. Koristeći kalibraciju Photon Voice automatski podešava prag sluha.

3.3.3 Tipka za govor

Photon Voice također podržava opciju tipke za govor (engl. Push-To-Talk) koja se zahvaljujući njemu lako implementira. Korisnik je tada zadužen za ručno upravljanje snimanja i prijenosa glasa pritiskom tipke koja je zadužena za omogućavanje prijenosa glasa i govora. Ova opcija ekvivalentna je akciji uključivanja i isključivanja mikrofona.

3.3.4 Zvučne grupe

Photon Voice ne služi samo za emitiranje. Korisnicima se može omogućiti opcija višestrukog razgovora koji se odvija u isto vrijeme bez međusobnog ometanja. To se može postići korištenjem zvučnih grupa (engl. Audio Groups) koje koriste Photon dodatak interesnih grupa (engl. Interest Group). Zvučna grupa identificira se brojem čiji je tip podatka bajt. Korisnik se može pretplatiti i odjaviti s više različitih grupa, međutim može prenositi zvuk samo jednoj grupi u isto vrijeme.

Po zadanim vrijednostima, svi glasovni korisnici pretplaćeni su na grupu "0". To znači da svaki korisnik može čuti druge korisnike i svi korisnici mogu međusobno komunicirati. Da bi se to promijenilo potrebno je promijeniti postavku globalne zvučne grupe. Jednom kad korisnik to napravi, samo korisnici koji su unutar te zvučne grupe mogu međusobno komunicirati. Baš kao što je ranije navedeno, korisnik može slušati više od jedne grupe istovremeno, ali samo unutar jedne grupe može prenositi glas ili zvuk.

Bitno je uzeti u obzir sljedeće:

- To se može napraviti samo prilikom korisnikovog spajanja u glasovnu sobu
- Korisnik se ne može odjaviti iz zadane zvučne grupe "0".

Isto tako, zvučne grupe mogu se povezati s opcijom tipke za govor, što korisnicima može biti dobar dodatak za igru.

3.4 Paket Kinect with MS-SDK

"Kinect with MS-SDK" je skup skripti i primjera koji služe za lako upravljanje modelima pomoću Kinect v1 uređaja. One demonstriraju kako koristiti modele kojima upravlja Kinect, geste koje detektira Kinectom i druge akcije vezane za rad Kinecta. Ovaj dodatak koristi Kinect alate za razvoj koje pruža Microsoft. Može se skinuti i dodatak imena "KinectExtras" za Kinect v1 s dodatnim primjerima kako ostvariti interakciju s Kinect-om, Kinect prepoznavanje govora, praćenje lica ili uklanjanje pozadine. Ta dva paketa rade samo za Kinect v1 i oba se mogu koristiti u programskim alatima Unity Free i Unity Pro. Za potrebe ovog diplomskog rada koristit će se samo Kinect With MS-SDK paket. Postoji i zaseban paket za Kinect v2 s dodatkom sličnim kao za Kinect v1, međutim ti se dodaci naplaćuju. Svi ovi paketi dostupni su na stranici [6]. Da bi ovaj dodatak uspješno radio potrebno je imati instaliranu verziju 1.8 Kinect alata za razvoj.

3.5 Paket OVR

Paket "OVR" je dodatak za Unity koji se može skinuti na službenoj stranici podrške za Oculus Rift [9]. Za skidanje samog alata potrebno je napraviti registraciju na njihovoj stranici. Ovaj alat sadrži pripremljenu kameru za prikaz scene preko uređaja Oculus Rift uz nekoliko primjera.

4. Implementacija virtualne sjednice

Za implementaciju Virtualne sjednice potreban je virtualni prostor na internetu na koji se korisnici mogu spojiti svojim modelima i međusobno izvršavati interakciju jedni među drugima koristeći govor i glasovanje kao način komunikacije. Spajanje na virtualnu sjednicu implementirat će se korištenjem aplikacije za umrežavanje Photon Engine.

Da bi se mogle koristiti usluge koje nudi Photon Engine potrebno se registrirati na njihovu stranicu kako bi se dobili identifikacijski kodovi [4]. Na taj način omogućuje se pristup njihovim aplikacijama. Za potrebe ovog diplomskog rada korištene su samo dvije: Photon Unity Networking (PUN) i Photon Voice.

Kao što je ranije navedeno Photon oblak je skup računala na kojima je pokrenut Photon poslužitelj. Poslužitelji se dodaju na zahtjev, tako da podržava bilo koji broj igrača.

Photon Unity Networking će za korisnika rukovati Photon oblakom. Pozadina oblaka je optimizirani socket protokol, pouzdan i ne pouzdan protokol izgrađen na vrhu UDP-a (bazirana na ENet što je pouzdana UDP biblioteka za umrežavanje), enkripcija i ostali dodaci.

Jedina mana ovog oblaka jest ta da Photon oblak nije u potpunosti autoritativni poslužitelj, jer aplikacija za Photon poslužitelj nije osviještena o geometrijskoj strani Unity-a, fizici i kolizijskom sustavu, što znači da je te dijelove potrebno obraditi na serverskoj strani. Uz to potrebno je implementirati sinkronizaciju. Prilikom implementacije, očito je tko je vlasnik. Korisnik je vlasnik i korisnik je dužan osvježavati svoju poziciju i obavještavati druge korisnike o svojim koordinatama. Kod drugih korisnika poslužitelj bi trebao biti vlasnik, ali Photon poslužitelj ne može biti vlasnik jer ne zna kako pomaknuti druge korisnike. Stoga je vlasništvo drugih korisnika potrebno

prebaciti na jednog od preostalih korisnika. Poslužitelj dodjeli vlasništvo jednom od igrača koji postane vlasnik drugih igrača i on postane glavni klijent. Uz to potrebno je implementirati dodatnu logiku koja će prebaciti vlasništvo na jednog od preostalih korisnika u slučaju da glavni klijent izgubi vezu ili izađe iz sjednice, što može izazvati velike sinkronizacijske probleme.

4.1 Postupak spajanja na udaljeni virtualni prostor

Svi korisnici prvo se spajaju na "imenovanog poslužitelja". On provjerava koju aplikaciju i koju regiju korisnički klijent želi koristiti. Te informacije prosljeđuju se glavnom poslužitelju (engl. master server)

Glavni poslužitelj je čvorište (engl. hub) za skup regionalnih poslužitelja. On poznaje sve postojeće igre. U bilo kojem trenutku kada se igra (soba) stvori ili joj se neki korisnik pridruži, klijent se prosljeđuje na jednu od drugih uređaja nazvan poslužitelj igre (engl. game server).

4.1.1 Sobe

Photon oblak stvoren je po principu "igra po sobi" (engl. room-based game), što znači da postoji ograničeni broj korisnika po aplikaciji, odvojenih od drugih. U sobi uobičajeno svi primaju informacije što drugi korisnici unutar sobe prosljeđuju. Izvan sobe, korisnici ne mogu komunicirati s drugima, stoga ih je uvijek potrebno držati u sobi.

Najbolji način da se korisnika smjesti u sobu jest da se koristi nasumično uparivanje (engl. random matchmaking). U principu korisnik zatraži da ga poslužitelj smjesti u bilo koju sobu s određenim uvjetima.

Sve sobe imaju ime kao identifikator. Ako je jedna soba pod određenim imenom puna ili zatvorena, korisnik se može pridružiti jednoj od

preostalih soba pod drugim imenom. Prikladno, glavni poslužitelj može pružiti listu soba za korisnikovu aplikaciju.

4.1.2 Predvorje

Predvorje (engl. lobby) za korisnikovu aplikaciju nalazi se na glavnom poslužitelju s listom soba za korisnikovu aplikaciju. U predvorju korisnik može dobiti pregled svih soba u koje se može spojiti. Isto tako, ovisno o aplikaciji, korisnik može dobiti informacije o sobi poput popunjenosti sobe, informacije o drugim korisnicima unutar sobe, te može proizvoljno odabrati želi li stvoriti novu sobu, ući u postojeću ili se pridružiti nasumičnoj sobi.

4.1.3 Identifikator aplikacije i verzija igre

Ako se svi korisnici spoje na istog poslužitelja, mora postojati način kako razlikovati svoj skup korisnika (tim igrača) od drugih korisnika (npr. jedan tim igra protiv drugog tima). Svaka igra (kao aplikacija) dobije svoj identifikator aplikacije (engl. AppID) u oblaku. Korisnici će se uvijek susresti s drugim igračima s istim identifikatorom aplikacije, ali samo na svom klijentu. Isto tako postoji i verzija igre (engl. game version) koja se može koristiti kako bi razdvojila korisnike sa starim klijentima od onih s novim klijentima.

4.1.4 Regije

Photon oblak organizira se u odvojenim regijama širom svijeta kako bi se spriječila loša povezanost s korisnicima koji su potencijalno predaleko. Bitno je razumjeti ovaj koncept pogotovo kada je potrebno raditi s rasprostranjenim udaljenim timom u različitim regijama. Testiranje aplikacija s članovima istog tima može biti onemogućeno zbog regijske podjele. Stoga je bitno namjestiti na prisilu istu regiju svim članovima tima tako da se smjeste u istu regiju kako bi mogli ostvariti interakciju jedni s drugima.

4.1.5 Scena

Scene sadrže objekte sobe. Pri odabiru korisnik može birati kojoj sobi želi pristupiti. Odabirom jedne od ponuđenih soba program pokreće scenu. Svaka soba može biti povezana s istim tipom scene, no isto tako svaka soba može biti povezana s različitim tipom scene, ali to ovisi o aplikaciji i unaprijed definiranim specifikacijama. Scene su unaprijed izgrađene i imaju postavljene objekte nad kojima korisnik može ostvarivati interakciju nakon ulaska u sobu.

4.1.6 Upravitelj igre

Kako bi se korisnici uspješno spojili na scenu po zadanim postavkama, potrebno je definirati skriptu koja će služiti kao upravitelj igre (engl. game manager). Zadaća upravitelja igre jest otvaranje scene i praćenje povezivanja korisnika prema definiranim parametrima, obavještavanje korisnika i slično. Na primjer, ako imamo sobu koja prihvaća samo 4 korisnika, upravitelj igre je onaj koji je dužan voditi brigu o tome. Ako se u sobi nalazi manje od 4 korisnika, upravitelj igre mora dopustiti korisniku da se pridruži sobi, osim ako ne postoje neki dodatni uvjeti, kao što je lozinka za pristup. U tom slučaju prvo mora provjeriti je li lozinka ispravna, te ako je pustiti korisnika u sobu, u suprotnom zabraniti mu ulaz i obavijestiti korisnika o razlozima zabrane ulaska. Isto tako ako se u sobi nalazi više od 4 igrača, korisniku treba zabraniti ulazak i obavijestiti ga da je soba puna, te mu ponuditi alternativna rješenja.

4.2 Model igrača

Kada se omogućilo spajanje na sobu, potrebno je stvoriti korisnika u virtualni prostor. S toga idući bitan korak jest izgradnja modela. Potrebno je napraviti pripremni model (engl. prefab). Dobar pristup je prvo izgraditi ili skinuti model koji nije povezan s Photon Unity Networking dodatkom. Tako se može ispravno testirati model kako bi bili uvjereni da model uistinu radi

bez mrežnih dodataka. Da bi se pripremni model mogao koristiti u mrežnoj aplikaciji potrebno je držati se nekoliko pravila:

- Pripremni model bitno je držati u datoteci s nazivom resursi (engl. Resources). Photon Unity Networking iz te datoteke dohvaća pripremne modela koje potom stvori na mreži.
- Potrebno je paziti na imena pripremnih modela. Bitno je da se imena pripremnih modela razlikuju, jer će Photon Unity Networking dohvatiti prvog kojeg nađe i njega staviti na mrežu.

Kako bi korisnik mogao upravljati modelom, bitno je stvoriti skriptu koja će upravljati pripremnim modelom. Pomoću te skripte određuju se naredbe za bilo koju vrstu interakcije, te se dodjeljuju vrijednosti poput brzina kretnje modela u aplikaciji. Međutim da bi to funkcioniralo, potrebno je da model na sebi ima dodijeljen animator. Animator je dodatak koji ima unaprijed pripremljene sekvence animacija koje model može izvesti. Uz njih definira se sekvenca i prijelazi stanja prikaza animacije modela u aplikaciji.

4.2.1 Kamera

Svaki model treba imati kameru koja služi kao gledište. Potrebno ju je dodijeliti svakom modelu prilikom ulaska u sobu kako bi igrač vidio model i prostor koji okružuje model. Tako korisnik može upravljati modelom i navoditi ga kroz prostor pritom mu zadavati naredbe koje treba izvršiti. Da bi se to ostvarilo potrebno je napraviti skriptu za praćenje modela. Kamera je dužna pratiti model kako bi se korisnik mogao kretati kroz virtualni prostor pomoću modela. Najbolje je podesiti kameru 2-3m iza modela za optimalne rezultate.

4.2.2 Sinkronizacija modela

Kada su pripremni model i kamera podešeni potrebno ih je programirati za spajanje na mrežu. Ključna skripta koja dolazi uz Photon Unity Networking paket je skripta pod nazivom "PhotonView". Ta skripta povezuje razne pojave na svakom računalu, te određuje koje komponente treba promatrati i na koji način. Ovaj postupak zove se sinkronizacija.

Očito je potrebno sinkronizirati poziciju i rotaciju modela tako da se prilikom kretanja modela jednog korisnika unutar virtualnog prostora, drugi korisnici kreću na isti način. Isto tako bitno je dodati i skriptu naziva "PhotonTransformView" koja služi kao posrednik između komponente transformacije i skripte "PhotonView". Ono što ta skripta radi jest olakšava korisniku izradu skripte koja mora paziti na latentnost mreže i sinkronizacije podataka koje model proizvodi.

Ujedno je bitno sinkronizirati komponentu animatora modela. Uz pomoć skripte "PhotonAnimatorView" umrežavanje postaje jednostavno i korisnik može definirati koje težinske slojeve treba sinkronizirati, ako se promjene za vrijeme igre. Preporučeno je pokušati sinkronizirati što manji broj parametara s kojima se korisnik može izvući kako bi se uštedjelo na obradi podataka i sprječavanju zagušenja propusnog pojasa.

Jednom kada su skripte dodijeljene potrebno je napraviti manju izmjenu u skripti za upravljanje modela. Tu sustav mora raspoznati koje podatke treba primiti i ažurirati od drugih korisnika, te koje podatke treba obraditi i dostaviti drugim korisnicima. To se lako osposobi naredbom isMine gdje se određuje da li model pripada određenom korisniku ili nekom drugom. Ovisno o istinitosti te naredbe ažuriraju se podaci za sinkronizaciju. Sustavi razmjene poruka međusobno komuniciraju između korisnika i tako se ostvaruje sinkronizacija među korisnicima i osvježavanje scene na mreži.

Zadnji dodatak koji treba ažurirati je kamera. Kod kamere je bitno ažurirati prikaz scene na temelju dobivenih dodataka. Postupak je sličan kao i kod ažuriranja kretanja modela, samo što se u ovom slučaju ažurira prikaz scene na mreži koju korisnik promatra.

4.2.3 Stvaranje instance modela

Kada su implementirane sve izmjene i model je pripremljen za prikaz na mreži, potrebno ga je stvoriti prilikom spajanja u sobu. Prilikom pokretanja sobe potrebno je pozvati metodu PhotonNetowrk.Instantiate() koja će stvoriti model na mrežu. Pri spajanja u sobu također se moraju navesti koordinate gdje će se model pojaviti na mreži. To mogu biti unaprijed određene koordinate (recimo da postoji teleporter na kojem se stvore svi korisnici koji sudjeluju na mreži) ili nasumične koordinate (unaprijed određena mjesta stvaranja ili ograničeno područje stvaranja modela na potpuno nasumičnim koordinatama). O ovome se brine upravitelj igre.

4.3 Dodavanje komponenti Oculus Rifta i Kinecta u scenu

Do sada je bio opisan proces stvaranja modela u virtualni prostor na mreži, međutim ovo se odnosilo isključivo na unaprijed definirani tip modela čije se gledište prikazuje na standardnom zaslonu, s unaprijed poznatim animacijama, te se modelom upravlja korištenjem ulaznih jedinica kao što su tipkovnica, miš, palica i slično.

No ono što se preko ovog diplomskog rada želi postići je prikaz virtualnog prostora preko Oculus Rifta i upravljanje modelom pomoću uređaja Kinect. Konačni rezultat je pokušaj zavaravanja korisnika i približavanja virtualnog prostora stvarnosti.

4.3.1 Povezivanje uređaja Oculus Rift i Kinect u zajedničku scenu

Način povezivanja uređaja Oculus Rift i Kinect opisan je u završnom radu [1]. Za potrebe izrade diplomskog rada koristit će se sustav povezivanja uređaja opisan u tom radu.

Kako bi se ta dva uređaja povezala potrebno je napraviti pripremni model koji uključuje obje komponente. Za upravljanje modelom pomoću uređaja Kinect potrebno je dodijeliti dvije skripte s nazivima "KinectManager" i "AvatarControllerClassic" koje su uključene u paket "Kinect with MS-SDK" koji je dostupan u Unity Assets web dućanu [6].

Skriptu "Kinect Manager" potrebno je dodijeliti kameri koja mora biti smještena nasuprot modela kojeg korisnik želi kontrolirati. Ova skripta u principu pretvara običnu kameru za prikaz scene u dubinsku kameru. Može se smatrati da ono što se događa u stvarnosti projicira se u virtualan prostor. U stvarnosti korisnik stoji nasuprot uređaja Kinect, dok ga uređaj snima i skuplja podatke koje mu korisnik dodjeljuje pokretima i gestama. Skripta zatim preslikava događanja u stvarnom svijetu u virtualni svijet. Jedina razlika je ta što uređaj Kinect u stvarnom svijetu skuplja i obrađuje podatke, dok kamera u virtualnom svijetu prima podatke i projicira skupljene podatke na skelet modela čime se zapravo ostvaruje kontrola nad modelom. Međutim da bi to bilo moguće na sam model potrebno je dodati skriptu "AvatarControllerClassic". U toj skripti definirana su imenovana polja gdje svako polje određuje jedan zglob čovjeka odnosno modela. Da bi se zglobovi modela povezali s poljima skripte, model mora imati definiran skelet. Skelet je zapravo hijerarhijska struktura kostura modela spojena u jednu cjelinu. Svaki zglob označava dio ljudskog tijela. Obično je korijen skeleta kralježnica, te se od kralježnice račva na gornji i donji dio tijela, pa sve do listova skeleta koji su obično glava, prsti na rukama i nožni prsti. Bez skeleta animacija modela nije moguća, stoga je bitno da model ima definiran skelet. Nakon što se svaki zglob poveže s imenovanim poljem, program se može

pokrenuti i korisnik preko uređaja Kinect može upravljati modelom korištenjem pokreta vlastitog tijela.

Za pregled virtualnog prostora preko uređaja Oculus Rift u scenu je potrebno uključiti pripremnu kameru pod nazivom "OVRCameraRig" koja se nalazi u datoteci "Prefab" unutar paketa "OVR". Potrebno ju je samo dodati na scenu i označiti kao glavnu kameru. To znači da sve što korisnik treba vidjeti u virtualnom prostoru treba se prikazati na projekciji te pripremne kamere. Kameru je potrebno pozicionirati ispred lica modela korisnika, tako da spajanjem u sobu korisnik ima dojam da je model kojim upravlja zapravo njegovo vlastito tijelo. Jednom kada se to napravi i sve je testirano da radi, potrebno je od tog sustava napraviti pripremni model koji će se koristiti kao model za interakciju u virtualnom prostoru na mreži.

4.3.2 Sinkronizacija uređaja

Jednom kada su uređaji spojeni u pripremni model, instanca modela se tada može prikazati na mreži. Kada se model pojavi na mreži korisnik može upravljati modelom pomoću uređaja Kinect i vidjeti okolni prostor kroz uređaj Oculus Rift. No problem nastaje kada se drugi korisnici prijave u sobu. Korisnici se mogu međusobno vidjeti, ali ne vide pokrete koje drugi korisnici proizvode. Mogu vidjeti pokrete koje sami proizvode, no drugi korisnici stoje u T pozi. Razlog tome je što korisnici pokrete vide na lokalnoj razini, ali ne i na globalnoj razini. Iako su modelu dodijeljene skripte "PhotonView", "PhotonTransformView" i "PhotonAnimatorView" pokreti modela na globalnoj razini se ne ažuriraju.

Korijen problema je vrsta modela koji se obično koristi u višekorisničkim aplikacijama naspram onog koji se koristi pri izradi ovog rada. Skipte "PhotonView", "PhotonTransformView" i "PhotonAnimatorView" služe za ažuriranje globalnih koordinata modela drugim korisnicima. "PhotonView" skripta je skripta koja omogućuje umrežavanje, te se kao takva, kod višekorisničkih aplikacija, dodjeljuje samo jedna po korisniku. Za ažuriranje

trenutnih koordinata na mreži na kojima se model nalazi ili prema kojima se model kreće zadužena je skripta "PhotonTransformView". Kada model treba izvršiti neku akciju obično se pokrene animacija akcije koju model izvodi. Za ažuriranje koordinata animacije modela zadužena je skripta "PhotonAnimatorView". I to je u redu za takvu vrstu modela.

Međutim model koji se koristi u ovom radu sastoji se od skupa zglobova kojima korisnik upravlja putem uređaja Kinect. Korisnikove akcije očitavaju se preko dubinske kamere i preslikavaju na model u virtualnom prostoru. Ovakav model obično stoji na mjestu i nema definirane animacije akcija koje će proizvesti. Jedna "PhotonView" skripta nije dovoljna da se ažurira cijeli model i skripte "PhotonTransformView" i "PhotonAnimatorView" u ovom slučaju postaju beskorisne.

Ovaj problem rješava se dodjelom skripte "PhotonView" svakom zglobu modela kojim će korisnik upravljati, slično kao što je opisano u radu [10]. Isto tako svakom zglobu potrebno je dodijeliti i skriptu "PhotonTransformView". Tako svaki zglob moći će komunicirati na mreži s drugim korisnicima, te će se njegove koordinate ažurirati svakim pokretom.

Ovdje se javlja veliki problem što se svakom zglobu dodjeljuje Photon gledište. Međutim problem se može ublažiti tako da se gledišta ne dodjele svim zglobovima, nego samo onim zglobovima kojim će korisnik upravljati. U ovom radu model koji se koristi je u sjedećem položaju i kao takvom zglobovi noge se neće koristiti. Tako se broj Photon gledišta smanjio s 20 na 11 i tako će svaki model kojim se korisnici prijave imati po 11 Photon gledišta.

4.3.3 Uključivanje dodatka Photon Voice

Zadnje što se modelu treba dodijeliti je mogućnost govora, odnosno omogućiti korisniku da komunicira s drugim korisnicima preko mikrofona. Da bi se to ostvarilo, modelu je potrebno dodijeliti skripte "PhotonVoiceSpeaker" i "PhotonVoiceRecorder". Skripta "PhotonVoiceSpeaker" omogućuje da

korisnik čuje što drugi korisnici govore, dok "PhotonVoiceRecorder" omogućuje snimanje i prijenos korisnikovog glasa drugim korisnicima. Prijenos glasa u ovom slučaju je omogućen samo unutar sobe u kojoj se korisnici nalaze.

4.4 Sustav za glasovanje

Konačni dodatak koji je potreban implementirati jest sustav za glasovanje. Glasanjem korisnik aktivno sudjeluje u anketama unutar virtualne sjednice. Korisnik ima pravo birati na koji način želi dodijeliti svoj glas. Ima pravo birati želi li svoj glas dodijeliti pritiskom gumba koji se nalazi na stolu, odnosno gestom podizanja ruke. U daljnjim poglavljima objašnjena su oba postupka.

4.4.1 Glasovanje pritiskom gumba

Prva metoda glasanja je pritiskom gumba. Unutar virtualne sjednice ispred korisnika postavljen je gumb koji može pritisnuti kako bi zabilježio svoj glas.

Da bi korisnik to postigao dovoljno je da gumb pritisne lijevom ili desnom rukom svog modela. Model gumba, model lijeve i desne ruke imaju dodijeljene komponente za detekciju kolizije (engl. Collider). Model gumba napravljen je u programskom alatu Autodesk 3DSMax i sastoji se od dvije komponente, od gornjeg djela koji se pomakne na pritisak, te od donjeg statičnog dijela. Animacija pritiska gumba napravljena je u programskom alatu Unity. U komponenti "Animate" definira se pomak po y osi za određeni dio vremena. Prvo se definira u kojem se vremenskom rasponu gornji dio gumba pomakne iz početne pozicije na poziciju do koje se gumb treba spustiti, a nakon toga se definira u kojem će se vremenskom rasponu vratiti u početnu poziciju. U ovom slučaju gornji dio gumba pomiče se samo po y-osi. Jednom kada se animacija spremi, stvaraju se dvije komponente "Animation"

i "Animator". U komponenti "Animate" spremljena je animacija koja će se prikazati kada se želi pokrenuti animacija za definirani model. Komponenta "Animator" dodijeli se komponenti koju se želi animirati s definiranom animacijom. Kako bi model animirao pritisak gumba i pridodao glas korisnika dodijeljena mu je skripta "AnimateButtonPress". Skripta detektira trenutak kada je model gumba ušao u koliziju s drugim objektom, te tada pokreće animaciju pritiska gumba gdje se gornji dio ponaša kao da se nalazi na opruzi, odnosno gornji dio se spusti i potom se opet podigne gore i zabilježi se korisnikov glas. Mora se pritom paziti da "Animator" radi sa skupom stanja, tako da je bitno ući u prozor gdje se prikazuju stanja i postaviti stanje mirovanja. Ovo je bitno napraviti jer će se u suprotnom prilikom detekcije kolizije pokrenuti stanje animacije i sustav će bez prestanka animirati pritisak gumba iako korisnik ne pritišće gumb. Razlog tome je što se prilikom automatskog stvaranja komponente "Animator" kao zadano stanje dodjeljuje stanje animacije, a sam sustav osvježava stanja i vrti se u beskonačnoj petlji unutar stanja koje je zadano kao zadano. Stoga je potrebno podesiti da se dodjeli zadano stanje koje postavlja model gumba u mirovanje, a komponentu animacije povezati sa stanjem mirovanja tako da se nakon jedne animacije komponenta "Animator" vrati u zadano stanje mirovanja. Tako svaki put kada korisnik pritisne gumb, animacija pritiska pokrenuti će se samo jednom pri pritisku gumba. Bitno je da se pritiskom gumba broj glasova poveća za jedan za svakog korisnika koji je pritisnuo gumb.

4.4.2 Glasovanje podizanjem ruke

Druga metoda glasanja je pritiskom gumba. Unutar virtualne sjednice da bi korisnik zabilježio svoj glas dovoljno je da podigne lijevu ili desnu ruku visoko u zrak.

Unutar paketa "Kinect with MS-SDK" postoji skripta imena KinectGesture. Ta skripta u sebi ima ugrađene već unaprijed definirane geste poput zamaha lijevo, zamaha desno, čučnja, skoka i drugih tipova gesti.

Kada korisnik izvede jednu od navedenih gesti Kinect detektira tu gestu. Jednom kada skripta KinectGesture detektira da li je gesta izvedena ili ne, obaviještava skriptu SimpleGestureListener. Nekoliko metoda koje ta skripta koristi za osluškivanje gesti:

- UserDetected() - Koristi se kao programski početak detekcije geste. Nije potrebno uklanjati geste koje ta metoda eksplicitno dodaje.
- UserLost() - Koristi se za oslobađanje varijabli i oslobađanja dodijeljenih resursa. Geste koje je dodijelila metoda UserDetected() uklanjaju se automatski prije poziva ove metode.
- GestureInProgress() - Metoda se poziva kada je gesta započela ali nije završila ili se prestala izvoditi.
- GestureCompleted() - Metoda se poziva jednom kada je gesta završena. Tu se može dodati vlastiti kod kako bi se odredilo kako rukavati s uspješno izvršenom gestom.
- GestureCancelled() - Metoda se poziva ako je gesta prestala s izvođenjem.

Za stvaranje proizvoljne geste potrebno je otvoriti skriptu KinectGesture i definirati proizvoljnu gestu u Gesture-enum. Zatim je potrebno pronaći metodu CheckForGesture(). Unutar te metode nalazi se *switch* naredba tipa *long* i u slučajevima je potrebno definirati slučaj za proizvoljnu gestu imenom koji je dodijeljen u Gesture-enum. Unutar te naredbe zapisani su svi procesi za detekciju svake pojedine geste.

Kod izrade vlastite geste prvo je potrebno provjeriti da li su zglobovi detektirani ili ne. Isto tako potrebno je odrediti poziciju zglobova ili udaljenost između zglobova za gestu. Unutar metode CheckForGesture() svaki slučaj ima definiranu svoju slučajnu *switch* naredbu. Svaka gesta definirana je slijedom numeričkih stanja. Njeno trenutno stanje pohranjeno je s drugim podacima u internoj strukturi tipa *GestureData*. Ovaj tip definiran je za svaku gestu koja treba biti detektirana u sceni. Početno stanje svake geste je 0. U

tom stanju kod treba detektirati da li je gesta započela ili nije. Da bi se to postiglo provjeravaju se i pohranjuju sve pozicije zglobova lijeve ili desne ruke. Ako je pozicija zgloba prikladna za početak geste, stanje će se pokrenut. U idućem stanju provjerava se da li je zglob postigao potrebnu poziciju (ili udaljenost od prethodnog zgloba), obično unutar vremenskog intervala. Ako je zglob postigao ciljanu poziciju (ili udaljenost) unutar vremenskog intervala gesta se smatra završenom, inače se smatra prekinutom. Tada se stanje detekcije geste vraća na 0 i ponovno započinje procedura očitavanja geste.

Za potrebe ovog rada, unutar same skripte, već su definirane geste za podizanje lijeve i desne ruke. Jedino što treba promijeniti jest ishod završetka geste, a to je da se poveća broj glasa za jedan za svakog korisnika koji je podigao lijevu, odnosno desnu ruku.

4.4.3 Glasovanje na mreži

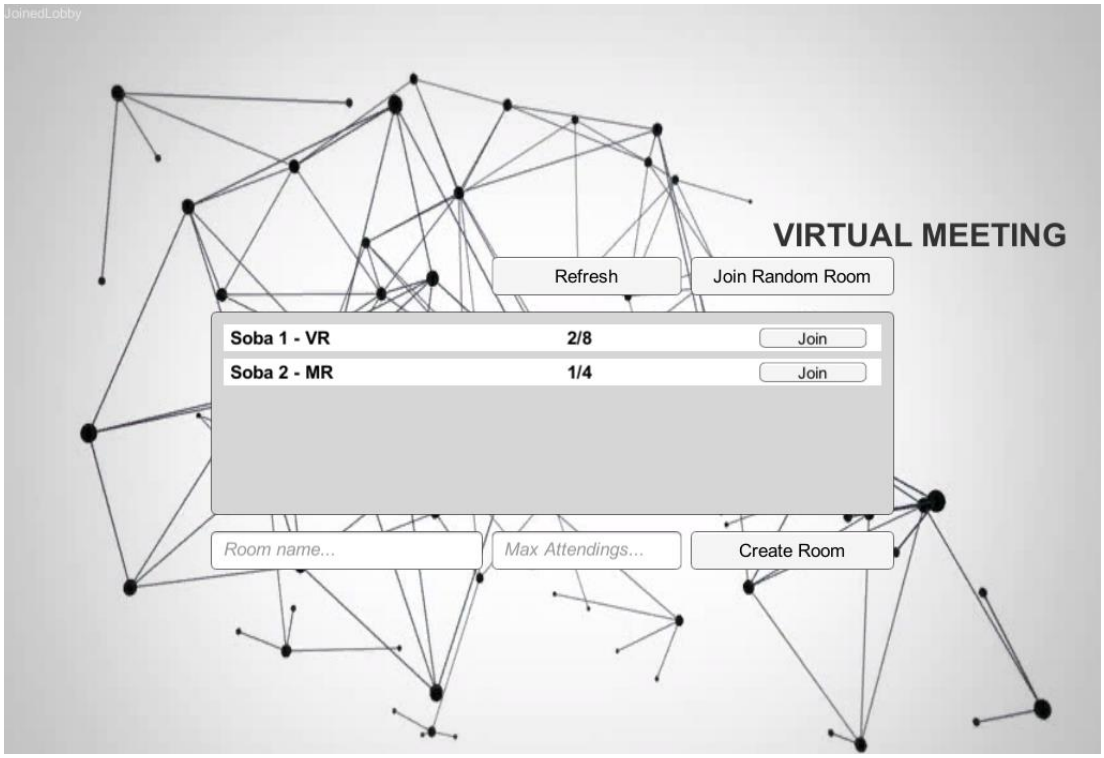
Neovisno koju metodu korisnik odabere njegov rezultat treba se prikazati svim korisnicima u stvarnom vremenu. Najbolji način da se to napravi je da se svakom korisniku u jednom uglu ekrana pojavi prozorčić s trenutnim brojem glasova. Taj prozorčić trebao bi biti definiran kao globalna komponenta. Tako se svim korisnicima odmah ažurira glas, a svakom novom korisniku koji uđe za vrijeme glasovanja očita se trenutni broj glasova spremljen na globalnoj komponenti. Isto se može postići kao da je prozorčić definiran kao lokalna komponenta, međutim tada bi svi korisnici trebali slati poruke drugim korisnicima o trenutnom stanju svojih prozora i informaciju o tome da li je netko u međuvremenu glasovao što u konačnici nije optimalno i bespotrebno može zagušiti mrežu. Ako se komponenta učini globalnom, bilo koji korisnik može glasovati i promijeniti globalnu komponentu. Poslužitelj onda primi informaciju o promjeni, uveća komponentu za jedan i svim korisnicima šalje obavijest o promjeni nakon čega se svakom korisniku ažurira prozorčić za prikaz glasova.

5. Aplikacija "Virtual Meeting"

Aplikacija "Virtual Meeting" implementira sve metode i postupke navedene u prijašnjim poglavljima i sjedinjuje ih u jednu cjelinu.

5.1 Glavni izbornik

Prilikom pokretanja aplikacije korisnik ulazi u glavni izbornik. U principu ono što se dogodi je da se korisnik smješta u predvorje aplikacije. Radi jednostavnijeg upravljanja napravljen je grafički prikaz glavnog izbornika na kojem korisnik može birati što želi poduzeti jednom kada uđe u predvorje. Slika 4. prikazuje izgled izbornika.



Slika 4. - Izgled glavnog izbornika

Kao što se može vidjeti na slici glavni izbornik sastoji se od prikaza stanja povezanosti korisnika na aplikaciju u gornjem lijevom uglu. liste dostupnih soba, njihovog imena i zauzeća, gumba za osvježavanje prikaza,

gumba za nasumičan odabir sobe, te gumba za stvaranje sobe čiji je preduvjet popuniti polja za ime sobe i broj maksimalnog broj osoba koji može pristupiti sobi. Kako bi korisnik uspješno bio spojen u s predvorjem u gornjem lijevom uglu treba pisati "JoinedLobby". Tada se korisniku treba osvježiti lista svih soba koje su trenutno aktivne. Ako soba nije puna, klikom na gumb "Join" korisnik se preusmjerava u odabranu sobu. U slučaju da je soba puna, korisnik će biti obaviješten da mora odabrati drugu sobu. Lista se ne osvježava automatski, stoga ako korisnik želi vidjeti trenutno stanje mora kliknuti gumb "Refresh". Tako se osvježava lista dostupnih soba. Klikom na gumb "Join Random Room" korisnika se preusmjerava u bilo koju od slobodnih soba. U slučaju da nema slobodnih soba, korisnika se obavještava o trenutnom stanju i ostavlja ga u predvorju. Odabirom gumba "Create Room" korisnika se usmjerava u novu sobu gdje on postaje voditelj sobe i tako sve dok ne ode iz sobe kada se uloga prebaci na drugog korisnika ili se soba zatvori u slučaju da nema niti jednog korisnika.

5.2 Model ureda

Nakon ulaska u sobu otvara se scena ureda. Scenu ureda preuzeo sam sa stranice [13]. To je kompleksan prostor koji se sastoji od više prostorija. Slika 5. prikazuje izgled ureda.

Unutar tog paketa nalaze se gotovi modeli uredske opreme kojima se lako da upravljati i može se modelirati drugačiji prostor od zadanog. U tu svrhu, gornji kat preuređen je tako da sadrži jedan veliki stol i nekoliko stolica. Svaka stolica rezervirana je za jednog korisnika. Prilikom ulaska u sobu korisnik će se stvoriti na jednoj od slobodnih stolica. Prilikom ulaska prvog korisnika sjednica započinje. Svaki idući korisnik prilikom ulaska u sobu stvorit će se na jednoj od preostalih stolica i pritom vidjeti druge korisnike koji su se prije njega pridružili sobi. Korisnici će se tako pojavljivati sve dok se soba ne popuni.

Slika 5. - Model ureda

Nakon toga drugi korisnici koji se žele pridružiti sobi moraju pričekati da netko od trenutno prisutnih korisnika napusti prostoriju. Sjednica će trajati sve dok zadnji korisnik ne napusti prostoriju.

5.3 Model korisnika

Model koji je korišten u diplomskom radu napravljen je prema metodama opisanim u završnom radu koji opisuje izradu modela 3D skeniranjem [2]. Slika 6. prikazuje glavu modela nastalu 3D skeniranjem.

Slika 6. - Glava modela nastala 3D skeniranjem

Za izradu modela korišteni su KScan3D, Kinect One, Autodesk 3DSMax i Blender. Uz pomoć kolege Marka Nađa napravljeno je 3D skeniranje ljudskog modela koristeći metode i postupke opisane u tom završnom radu. Prvotno je napravljeno 3D skeniranje cijelog tijela uz pomoć uređaja Kinect One i programskog alata KScan3D, a zatim određenih dijelova radi boljeg izgleda modela. Pod određene dijelove ubrajaju se ruke, šake, noge, stopala i glava, a razlog njihovog ponovnog skeniranja je poboljšano očitanje tih dijelova tijela i bolja rezolucija teksture. Cijelo skeniranje napravljeno je u dobro osvjetljenom prostoru kako bi se eliminirale nepoželjne sjene. Nakon skeniranja, zasebni dijelovi obrađeni su i spojeni u jednu cjelinu preko alata Autodesk 3DSMax i Blender. Zatim se proveo postupak ponovnog umrežavanja (engl. remesh) radi pojednostavljenja mreže modela, te postupak UV preslikavanja i generiranja teksture s polaznog modela na pojednostavljeni model kako bi se dobio realističan prikaz ljudskog modela. Posljednji dio izrade modela jest stvaranje kostura modela kako bi se model mogao pokretati i animirati.

Razlog izrade ovakve vrste modela jest taj da se korisniku što više približi dojam stvarnog svijeta. Iako bi bilo koji humanoidni model poslužio svrsi, realističniji dojam ostvario bi model stvarne osobe s kojim korisnik može komunicirati u virtualnom prostoru.

5.4 Interakcija u virtualnoj sjednici

Jednom kada korisnik odabere sobu, aplikacija ga smješta na jednu stolicu. Korisnik promatra sobu pomoću uređaja Oculus Rift. Okretanjem glave vidi prostor koji ga okružuje. Ispred korisnika nalazi se stol s gumbom za glasanje. Korisnik modelom upravlja svojim gornjim dijelom tijela, dok noge miruju, a njegove pokrete detektira uređaj Kinect. Neovisno da li se unutar sobe nalaze drugi korisnici ili ne, korisnik može govoriti i njegov glas će se prenositi u virtualnu sjednicu, međutim njega samo mogu čuti osobe koje su prisutne u toj sobi. Isto tako korisnik može čuti glasove drugih

korisnika, ali samo onih koji se nalaze u istoj prostoriji kao i on. Nakon što sjednica započne i dođe vrijeme za glasanje korisnik tada može birati želi li pritisnuti gumb ili podići ruku. Korisnikov glas zabilježit će se na malom ekranu pored stola gdje će se zapisati broj očitanih glasova. Jednom kada sjednica završi korisnik mora ugasiti aplikaciju kako bi izašao iz sobe. Primjer jedne virtualne sjednice prikazuje Slika 7.

Slika 7. - Virtualna sjednica u kojoj sudjeluje više različitih modela

5.5 Problemi pri izradi aplikacije

Iako se više korisnika može spojiti u istu sobu i održati virtualnu sjednicu postoji nekoliko problema koje prate ovu aplikaciju.

Jedan od glavnih problema jest smještanje samog korisnika u sobu. Prvi glavni problem je stvaranje instance modela u virtualnu sjednicu. Model korisnika stvori se unutar virtualne sjednice, međutim njegova pozicija je unaprijed određena, što znači da će se svaki korisnik pojaviti na istoj stolici. Ovaj problem privremeno je riješen tako da se napravi više aplikacija na kojima su definirane drukčije pozicije. Tako svaki korisnik ima dodijeljenu jednu poziciju na koju se njegov model stvori. Ovo može biti dobro samo

onda kada znamo da uvijek želimo korisnika smjestiti na istu poziciju odnosno stolicu.

Međutim tu nastaje drugi glavni problem, a to je da se tim pristupom ograničava broj korisnika koji može sudjelovati unutar sjednice. Za potrebe ovog diplomskog rada napravljena je samo jedna soba u koju se korisnici spajaju za jedan stol s četiri stolice. Ovo je dobro za slučaj kada se u virtualnu sjednicu želi smjestiti do 4 korisnika, međutim nije praktično za broj korisnika veći od 4. Iako aplikacija omogućuje spajanje više od 4 korisnika, doći će do preklapanja, te se može dogoditi da se dva korisnika pojave na istoj poziciji odnosno stolici.

Jedno od rješenja jest napraviti više scena s različitim brojem stolica, a broj slobodnih stolica neka označava maksimalni broj korisnika koji može sudjelovati u virtualnoj sjednici. Na taj način točno je određen broj korisnika koji može sudjelovati unutar jedne virtualne sjednice. Isto tako može se definirati da se korisnici nasumično smjeste na jednu od slobodnih stolica ili omogućiti korisnicima da sami odaberu stolicu na koju se žele smjestiti. Drugo rješenje može biti omogućavanje korisnicima koji nisu smješteni za stolicu da se smjeste negdje unutar sobe i prikažu kao u stojećem položaju. Tako mogu sudjelovati u sjednici a glasovati mogu podizanjem ruke.

Idući glavni problem jest sam model. Da bi korisnik mogao sudjelovati u virtualnoj sjednici mora imati dodijeljen model, ali model mora biti unaprijed definiran unutar aplikacije. Svakom modelu dodijeljeno je ime. Skripta zadužena za stvaranje instance modela stvara model u scenu po imenu koje joj je definirano kao ulazni parametar. Ako imamo slučaj da svaki korisnik ima svoju verziju aplikacije, gdje je unaprijed definirano gdje će se korisnik stvoriti, može se dogoditi da se zbog dodijeljenog imena svakom korisniku drugi korisnici prikažu istim modelom. Na primjer, recimo da se korisnik A stvori u virtualnoj sjednici, te se odmah nakon njega, stolicu do, stvori korisnik B. Umjesto da korisnik A vidi model korisnika B, korisnik A će

zapravo vidjeti model korisnika A na korisniku B i baš zbog toga što je unaprijed definirano ime modela koje se treba stvoriti.

Jedno moguće rješenje je da se prilikom ulaska u sobu nudi mogućnost odabira modela koji će korisnika predstavljati u virtualnoj sjednici. Drugo rješenje je da se definira baza podataka čije tablice sadrže skup modela. Svakom korisniku se tada dodijeli identifikacijski broj, te se prilikom ulaska u sobu učita model korisnika iz baze po tom identifikacijskom broju.

Još jedan od problema jest gubitak prijenosa podataka koje očitava dubinska kamera. Za vrijeme očitavanja pokreta korisnika Kinect može izgubiti vezu, te se zbog toga model korisnika prestane kretati. Problem leži unutar skripte "KinectManager", te bi se trebalo doraditi rukovanje prilikom gubitka toka. Problem se lako riješi isključivanjem i ponovnim uključivanjem uređaja Kinect. Ako to ne riješi problem, potrebno je izaći i ponovno uči u aplikaciju.

Postoji i problem pri preslikavanju korisnikovih pokreta na model. Problem je bio što su se nakon obrade podataka preko Kinecta podaci preslikavali na sve modele u sceni umjesto na onog jednog kojim korisnik upravlja, te su se tako podaci preslikavanja međusobno ometali. Problem se riješio uništavanjem kamera za preslikavanje pokreta na drugim modelima pri ulasku u sobu, a ostavila se samo ona koja je bila povezana za model korisnika kojim on upravlja. To se moglo napraviti jer se postupak preslikavanja treba raditi lokalno, a skripte "PhotonTransformView" koje se nalaze na zglobovima modela zadužene su za ažuriranje trenutnih pozicija zglobova u globalnoj sceni.

Jedini problem koji je ostao ne riješen događa se pri spajanju korisnika u sobu, kada se nekoliko korisnika već nalazi u sobi. Dolazi do problema da se novo spojenom korisniku ne prikažu svi prethodno spojeni korisnici. Istraživanjem je otkriveno da se greška javlja pri pozivu metode LoadLevel().

Jedan od razloga je da se prilikom stvaranja instance modela i očitavanja scene ne zaustave procesi dok se ne učita novi igrač, te se zbog toga ne stignu učitati korisnici koji su prethodno spojeni u sceni. No problem se počeo javljati onda kada je modelu dodijeljena skripta "AvatarContollerClassic". Moguće je da se njeni procesi, prilikom stvaranja modela, ometaju kod očitanja već prisutnih korisnika u sceni, stoga bi jedan od mogućih rješenja bio aktivacija skripte nakon što se svi modeli učitaju u scenu.

Zaključak

Ovakva vrsta aplikacije mogla bi se koristiti u bližoj budućnosti. Ponudila bi novi način komunikacije i interakcije među ljudima, te dodala osjećaj stvarnosti u samo virtualno iskustvo. Koristila bi se za susrete s prijateljima koji žive na udaljenim mjestima, poslovne sastanke gdje bi se uštedio novac za prijevoz i smještaj, veće konferencije, pa čak i saborsku sjednicu gdje bi svi mogli prisustvovati iako trenutno nisu fizički prisutni na istoj lokaciji. Izradom ovakve vrste aplikacije dokazano je da se s navedenim tehnologijama može ostvariti virtualna sjednica. Sama aplikacija trebala bi se doraditi, ali bi isto tako mogla proširiti mogućnosti. Mogla bi implementirati spajanje unutar sobe korištenjem sigurnosne lozinke, kako bi spriječio ulazak nepoželjnih gostiju. Također bi se mogli implementirati neki posebni dodaci kao što su prikaz raspored tema unutar jedne sobe za svakog korisnika prije ili prilikom ulaska u aplikaciju, mogućnost snimanja i zabilješki događaja na virtualnoj sjednici, mogućnosti glasanja korištenjem detekcije glasa ili drugih gesti, prikaz statistike glasovanja na kraju sjednice i drugo. Isto tako, korištenjem alata Photon Engine ne mora se nužno ograničiti isključivo na uređaje Oculus Rift i Kinect, nego se može implementirati niz drugih uređaja i dodataka koji ovakvu vrstu aplikacije mogu pretvoriti u među platformsku aplikaciju. Ukratko, pokazano je da ima mjesta za napredak, ali da ima i potencijala razviti se u jedinstvenu socijalnu aplikaciju za interakciju među ljudima u virtualnom svijetu.

Literatura

[1] Lukež R.Z., Povezivanje uređaja Oculus Rift i Kinect u zajedničkoj sceni, Završni rad, Fakultet elektrotehnike i računarstva, 2015

[2] Nađ M., Izrada ljudskih likova pomoću uređaja Kinect, Završni rad Fakultet elektrotehnike i računarstva, 2016

[3] Exit Games, *Exit Games*, https://en.wikipedia.org/wiki/Exit_Games, 08.06.2017

[4] Photon PUN, *Photon PUN*, https://www.photonengine.com/en-US/pun/, 08.06.2017

[5] Photon Voice, *Photon Voice*, https://www.photonengine.com/en-US/Voice, 08.06.2017

[6] Filkov R., Kinect with MS-SDK, 10.08.2015, *Kinect with MS-SDK*, https://www.assetstore.unity3d.com/en/#!/content/7747, 12.06.2017

[7] Filkov R., Kinect with MS-SDK, 16.12.2013, *Kinect with MS-SDK*, https://rfilkov.com/2013/12/16/kinect-with-ms-sdk/, 12.06.2017

[8] Filkov R., KinectExtras with MsSDK, 17.12.2013, *KinectExtras with MsSDK*, https://rfilkov.com/2013/12/17/kinectextras-with-mssdk/, 12.06.2017

[9] Oculus SDK, *Oculus SDK*, https://developer.oculus.com/develop/, 12.06.2017

[10] Iguchi K., VR ネトゲの作り方：Mikulus Kinect Online におけるPhoton Cloudの活, 04.12.2013, *VR ネトゲの作り方：Mikulus Kinect Online におけるPhoton Cloudの活*, http://www.heistak.com/2013/12/04/mikulus-kinect-online-photon-cloud/

[11] Pterneas V., Implementing Kinect Gestures, 27.12.2014, *Implementing Kinect Gestures*, http://pterneas.com/2014/01/27/implementing-kinect-gestures/, 14.06.2017

[12] How to create custom gesture in Kinect v2 with MS SDK, 23.03.2015, *How to create custom gesture in Kinect v2 with MS SDK*, http://www .devindia.biz/how-to-make-custom-gesture-in-kinect-v2-with-ms-sdk/, 14.06.2017

[13] 3D Office Furniture, 14.04.2015, *3D Office Furniture* https://www.assetstore.unity3d.com/en/#!/content/34262, 16.06.2017

Programska podrška za virtualnu sjednicu

Sažetak

Diplomski rad obrađuje temu programske podrške za virtualnu sjednicu. U uvodnom poglavlju navodi se motivacija za izradu diplomskog rada, te ukratko opisuje što će diplomski rad obrađivati. Zatim se opisuje tehnološka podrška za uređaje Kinect i Oculus Rift, te programska podrška za dodatke Photon Engine, Photon Unity Network, Photon Voice, paket Kinect with MS-SDK i paket OVR. Opisuju se načini rada pojedinih komponenti, njihove specifikacije, karakteristike, prednosti i opisi problematike. Također se opisuje njihovo međusobno povezivanje te načini implementacije. Nakon toga opisuje se aplikacija, način rada i izvedbe aplikacije, opis problema s kojima se korisnik može susresti prilikom izrade, te njihova implementirana ili predložena rješenja. Na kraju diplomskog rada radi se osvrt na postignute rezultate, prijedlog mogućih poboljšanja i budućih primjena.

Ključne riječi:

Oculus Rift, Kinect, Photon Engine, Photon Unity Network, Photon Voice, virtualna sjednica, virtualna stvarnost, 3D skeniranje, glasovanje

Software Support for Virtual Meeting

Abstract

The master's thesis covers the topic of software support for virtual meeting. The introduction states the motivation for the making of the master's thesis and in short describes what the master's thesis will cover. Then the technological support for Kinect and Oculus Rift, along with software support for Photon Engine, Photon Unity Network, Photon Voice, package Kinect with MS-SDK and package OVR is described. The way in which each individual components works, their specifications, characteristics, advantage and description of problems is described. Connected interactions between them, along with implementation methods are described. After that the application, mode of operation and implementation are described, description of problems user may come across during the production, their implementations and suggested solutions. At the end of the master's thesis achieved results, suggestions of possible enhancements and future application is reviewed.

Key words:

Oculus Rift, Kinect, Photon Engine, Photon Unity Network, Photon Voice, virtual meeting, virtual reality, 3D scanning, voting