SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 646

UPRAVLJANJE MODELIMA GESTAMA RUKU

Adnan Abdagić

Zagreb, veljača 2014.

Zahvala

Prije svega bih se zahvalio obitelji na ljubavi i podršci kroz sve ove godine.

Zahvaljujem se mentorici, prof. dr. sc. Željki Mihajlović na susretljivosti i pruženoj pomoći pri izradi ovog rada.

I na kraju, veliko hvala prijateljima koji su mi svojim društvom olakšali studij i pomogli da postanem ono što sam danas.

Sadržaj

Uvod

U današnjem svijetu računala su integralni dio naših života. S obzirom na to ljudi teže razviti što jednostavnije načine upravljanja s njima. To je naročito slučaj u situacijama koje se mogu poistovjetiti sa situacijama u stvarnom svijetu, ali nismo u mogućnosti da s njima na isti interaktivno način.

Aplikacije virtualne stvarnosti koje se baziraju na 3D virtualnim okruženjima su razvijane u različitim područjima. Obično takvi sustavi primjenjuju miš ili tipkovnicu kao korisničko sučelje za interakciju s objektima. Kako bi se razvila realističnija interakcija između čovjeka i računala javile su se sučelja bazirana na pokretima ruku i gestama. Predstavljaju jako obećavajuću tehnologiju uslijed svojih prirodnih svojstava, tako da je poželjno razviti intuitivnija i prirodnija korisnička sučelja imitirajući interakcije iz stvarnog svijeta u virtualnom okruženju.

U radu je razrađeno jednostavno fleksibilnije sučelje koje omogućuje korisniku da upravlja virtualnim objektima, modelima, u 3D virtualnom okruženju. Korisnik je u mogućnosti manipulirati objektom tako da ga pomiče, rotira, skalira, kao i izaziva određene radnje uporabom gesti. U tu svrhu koristimo uređaj Leap Motion za snimanje i praćenje pokreta ruku u stvarnom svijetu, dok se za veći doživljaj 3D okruženja korisnika koristi Oculus Rift, HMD koji će korisnika „ubaciti“ u virtualno okruženje.

1. Tehnologije praćenja pokreta

Praćenje pokreta je relativno novo područje interakcije čovjeka i računala koje je tek nedavno postalo dostupno širim potrošačima. Vrlo je veliko i brzo područje razvoja. U osnovi se svodi na sposobnost praćenja pokreta objekata tokom vremena. Obično je objekt ljudsko tijelo, međutim ne mora biti.

Postoji velika razlika između detekcije pokreta i praćenja pokreta. Praćenje pokreta u suštini zahtjeva identificiranje određenih točaka, oblika ili objekata i praćenje njihovih pojedinih pokreta tokom vremena. Jednom kad je predmet prepoznat, pokušava se otkriti u kojem smjeru se objekt kreće, obično u 3D prostoru. Nakon identifikacije i praćenja objekta dobiveni podaci se mogu koristiti za izvršavanje određenih akcija, ali i za dubinsku analizu kako bi se realizirale određene geste.

Danas u svijetu postoje različiti uređaji za praćenje pokreta. Neke tehnologije su prikladnije za određene aplikacije od drugih.

Jedan od osnovnih načina praćenja pokreta omogućuju akcelerometri. Akcelerometar je mali senzor koji može detektirati ubrzanje. Kombiniranjem više različitih akcelerometara i žiroskopa mogu se detektirati točni pokreti u svim mogućim smjerovima. Problem s njima je da ga korisnik mora imati na sebi kako bi se detektirao pokret – implicitno identificiraju objekt čiji se pokret prati (on predstavlja korisnika i poznat je).

Čak i obične kamere se mogu koristiti za praćenje pokreta, ali su dosta limitirane i zahtjevaju veliku procesorsku snagu pri obradi slike.

Jedan od novijih načina koji se istražuje za praćenje pokreta je uporabom magnetskih polja kao uređaj Razer Hydra. Još jedan obećavajući način za detekciju pokreta je da se direktno detektiraju električki impulsi koje mozak šalje mišićima. MYO je jedan od takvih uređaja.

1.1. Praćenje pokreta infracrvenim zrakama

Češći način detekcije pokreta danas uključuje korištenje infracrvenih emitera i detektora. Više od pola energije od Sunca dolazi na Zemlju u obliku ljudima nevidljivih infracrvenih zraka. Microsoft Kinect i Leap Motion su najpopularniji primjeri uređaja koji korištenjem infracrvenih zraka mogu triangulirati udaljenost objekta od samog senzora i na osnovu toga ga detektirati u prostoru. Loša strana uređaja koji koriste infracrvene zrake kao primarni mehanizam za praćenje pokreta su lošiji rad i smetnje pod dnevnim sunčanim svjetlom i blizu drugih izvora infracrvenih zraka.

Miscrosoft Kinect i Leap Motion su dosta slični uređaji, međutim izrađeni su za različite uloge. Glavna razlika je u tome da je Leap Motion dizajniran da radi na maloj udaljenosti i za praćenje jako malih pokreta, dok je Microsoft Kinect dizajniran da se koristi iz veće daljine i fokusira se na praćenje individualnih kostura korisnika te samih kretanja točaka na tom kosturu. Dalje, Leap Motion je namijenjen za korištenje pri sjedenju ispred računala i može pratiti s izrazito velikom preciznošću i druge predmete osim same ruke. Microsoft Kinect ne prati precizno poput Leap Motiona, međutim može pratiti kosture više korisnika. Isto tako dolazi opremljen kamerom i mikrofonom za interakciju.

Tablica 1.1 Usporedba Leap Motion i Microsoft Kinect

Leap Motion	**Microsoft Kinect**
Manja udaljenost (1 metar)	Veća udaljenost (1.2 do 3.5 metara)
Preciznost oko 0.01 milimetar	Praćenje kostura tijela
Detaljno praćenje prstiju	Mikrofon
Podrška za prstolike alate	Kamera vidljive svjetlosti

Slika 1.1 Leap Motion i Microsoft Kinect

1.2. Uporaba u praksi

Danas postoji velika zainteresiranost da se ovakav tip tehnologija primjeni u industriji u velikom broju područja. Neka od područja gdje se koristi, eksperimentalno ili produkcijski:

- Zabavna industrija – Najraširenija uporaba danas je u industriji igara, gdje se pomoću ovih tehnologija nastoji povećati uronjenost ljudi u sadržaj koji se prikazuje. Filmska industrija također široko koristi praćenje pokreta, pa bilo to u preslikavanju stvarnih pokreta na animacije ili način na koji prezentiraju sadržaj.
- Medicina – Praćenjem biomehanike pacijenata mogu se dijagnosticirati potencijalne bolesti. U kirurgiji pružaju mogućnost doktoru interakciju računalom bez korištenja ekrana i tipkovnice. Štoviše, ide se tako daleko da će u budućnosti biti mogući cjelokupni operacijski zahvati bez da se doktor nalazi u istom mjestu kao pacijent.
- Industrijski dizajn – Brojne kompanije istražuju mogućnost prirodnije interakcije s tehnologijom kako bi olakšali razvoj kompliciranih industrijskih postrojenja. Jedan od pionira je Elon Musk sa svojim Space-X programom, u koji čak uključuje tehnologije razmotrene u ovom radu.
- Asistivne tehnologije – Široko se već koriste u ovoj grani kako bi pomogle osobama s umanjenim radnim sposobnostima, koje ne mogu koristiti klasične uređaje za unos podataka.

Praćenje pokreta je još mlado područje i postoji cijeli svijet mogućnosti za njihovo istraživanje i korištenje.

2. Uređaj Leap Motion

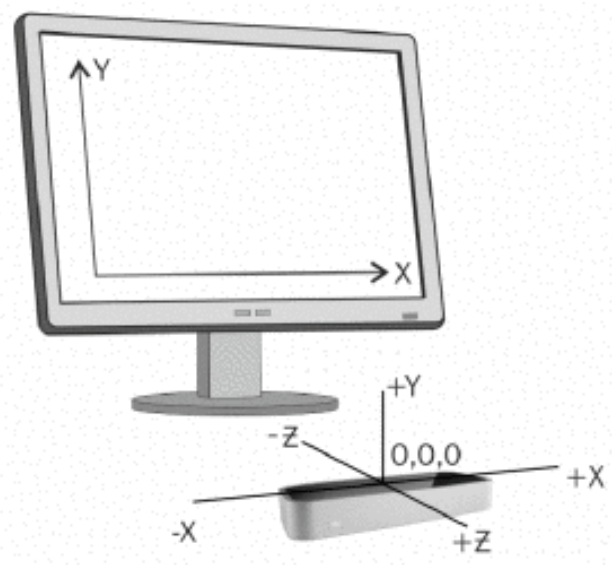
Uređaj Leap Motion omogućuje revolucionaran i uzbudljiv novi način interakcije s tehnologijom. Predstavlja prirodno sučelje zasnovano na gestama, te omogućuje inženjerima programske podrške pristup snažnom skupu alata za njegovo upravljanje. Leap Motion koristi par kamera i infracrvenu projekciju uzorka za stvaranje slike naših ruku pomoću dubinskih informacija. Kako bi se smanjila njihova cijena, sami uređaji vrše najmanji dio procesiranja i obrade dotičnih podataka. To čini Leap Motion ovisnim o snazi računala na kojem se koristi. Slike se post-procesiraju na korištenom računalu kako bi se uklonio šum i izgradio model naših ruku, prstiju ili pokazivačkih alata koje držimo. Vidno polje Leap Motiona je invertirana piramida centrirana na samom uređaju od približno 25 do 600 milimetara iznad uređaja [1].

Leap Motion nudi i moćan API koji omogućuje jednostavnu integraciju gesti u našu aplikaciju. Unutar SDK-a su realizirani mnogi koncepti poput: detekcije ruku unutar okvira, uključujući rotaciju, poziciju, brzinu i pokret u odnosu na raniji okvir; prepoznavanje svih prstiju i pokazivačkih alata (*pointables*) s rotacijom, pozicijom i brzinom; točna lokacija slikovnih elemenata na ekranu; prepoznavanje osnovnih gesti; detekcija promjene pozicije i orijentacije kroz okvire [2].

Slika 2.1 Leap Motion bez vanjskog kućišta

2.1. Koordinatni sustav

Leap Motion koristi desni Kartezijev koordinatni sustav, dok se vrijednosti izražavaju u milimetrima. Ishodište se nalazi u centru Leap Motion uređaja. X i Z osi leže na horizontalnoj ravnini, pri čemu se X os pruža uzduž duljeg ruba uređaja. Y os je vertikalna, s pozitivnim vrijednostima prema gornjoj strani (za razliku od mnogih koordinatnih sustava korištenih u računalnoj grafici). Pozitivne vrijednosti Z osi se povećavaju udaljavajući se od ekrana računala.



Slika 2.2 Leap Motion koordinatni sustav

2.2. Podaci praćenja objekata i pokreta

Dok Leap Motion prati ruke, prste i alate u svom vidnom polju, nudi nam podatke kao niz ili okvir informacija. Svaki okvir sadrži listu osnovnih podataka koji se prate, o rukama, prstima i alatima, kao i prepoznate geste i faktore koji opisuju sveukupne pokrete u sceni.

`Controller` klasa predstavlja glavno sučelje između Leap Motion uređaja i naše aplikacije. `Controller` objekt se povezuje s uređajem i preko Frame objekta prima podatke o praćenju. Njemu pristupamo preko `Controller.Frame()` metode unutar petlje za osvježavanje.

Kada detektira ruku, prst, alat ili gestu, uređaj mu pripiše jedinstveni identifikator. Identifikator se ne mijenja dok dotični entitet ostaje prisutan unutar vidnog polja uređaja. Ako se praćenje izgubi i ponovno uspostavi, uređaj mu može pridijeliti drugi identifikator.

U budućnosti se planira predstaviti kosturni model ruke kako bi se omogućilo bolje praćenje i kontinuitet kroz vrijeme.

2.2.1. Struktura okvira

Objekt `Frame` nudi liste podataka koji se prate, gesti i faktora koji opisuju općenite pokrete očitane u vidnom polju Leap Motiona.

Lista podataka koji se prate

- `Hands` – Sve ruke u vidnom polju.
- `Pointables` – Svi prsti i alati kao `Pointable` objekti.
- `Fingers` – Svi prsti.
- `Tools` – Svi alati.
- `Gestures` – Sve geste koje su počele i završile, ili se osvježile.

Tri pokazivačke liste (`Pointables`, `Fingers` i `Tools`) sadrže svaki pokazivački objekt sadržan unutar okvira. Pokazivačkim objektima asociranim s rukom možemo pristupiti preko `Hand` objekta u listi ruku. S druge strane, prst ili alat ne mora biti asociran s rukom ako je korisnikova ruka samo djelomično unutar vidnog polja Leap Motiona.

Ako pratimo individualni objekt, kao što je prst, kroz okvire, možemo koristiti identifikator asociran s tim objektom tako da ga referenciramo u svakom okviru. Koristimo sljedeće funkcije kako bi došli do specifičnog objekta preko identifikatora:

- `Frame.Hand()`
- `Frame.Finger()`
- `Frame.Tool()`
- `Frame.Pointable()`
- `Frame.Gesture()`

Ove funkcije vraćaju referencu na odgovarajući objekt ako postoji u trenutnom okviru. Ako objekt ne postoji onda se vraća specijalni invalid objekt. Invalid objekti su dobro definirani, ali ne sadrže podatke o praćenju. Ova tehnika pomaže smanjiti količinu `null` provjeravanja kad se pristupa podacima za praćenje Leap Motion uređaja.

2.2.1.1 Pokreti unutar okvira

Leap Motion analizira sveukupne pokrete koji su se dogodili od prethodnog okvira i sintetizira reprezentativne translacijske, rotacijske i faktore skaliranja. Ako pomjerimo obje ruke lijevo u vidnom polju uređaja okvir će sadržavati translacije; ako vrtimo ruke kao da držimo loptu, sadrži rotacije; ako pomjerimo jednu ruku prema ili od druge, okvir sadrži skaliranje. Leap Motion koristi sve objekte unutar vidnog polja dok analizira pokrete. Ako detektira samo jednu ruku onda se faktori pokreta unutar okvira zasnivaju na samo njenim pokretima. Slično, ako su obje ruke u vidnom polju onda ih zasniva na pokretima obje ruke. Također možemo dobiti i neovisne faktore za svaku ruku zasebno preko `Hand` objekta.

Pokreti unutar okvira su izvedeni uspoređujući trenutni okvir s određenim ranijim okvirom. Atributi koji opisuju sintetizirane pokrete uključuju:

- `RotationAxis` – Vektor smjera za os rotacije.
- `RotationAngle` – Kut rotacije oko osi rotacije u smjeru kazaljke na satu.
- `RotationMatrix` – Transformacijska matrica za rotaciju.
- `ScaleFactor` – Faktor skaliranja (ekspanzija i kontrakcija).
- `Translation` – Faktor translacije.

Uporabom ovih faktora možemo manipulirati objektima unutar scene u našoj aplikaciji, bez da pratimo pojedine ruke i prste kroz višestruke okvire.

2.2.2. Model ruke

Model ruke pruža informacije o poziciji, karakteristikama i pokretima detektirane ruke, kao i listu prstiju i alata asociranih s rukom.

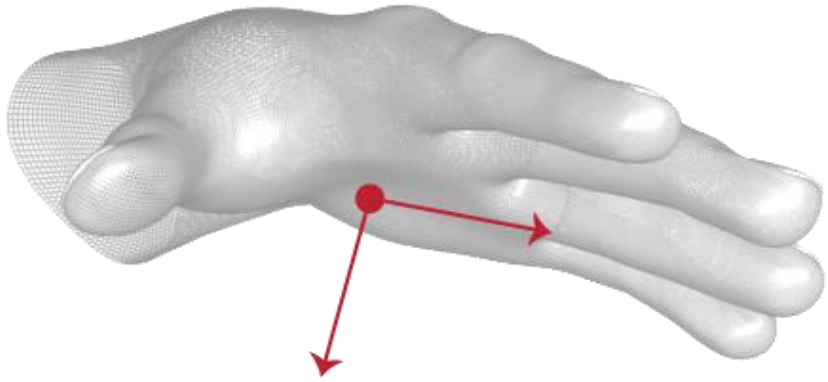
Leap Motion API nudi jako puno informacija o ruci, međutim nije sposoban utvrditi sve atribute ruke u svakom okviru. Kada je ruka stisnuta u šaku njeni prsti nisu vidljivi uređaju pa će lista prstiju biti prazna. To je jedan od slučajeva koji treba uzeti u obzir prilikom oblikovanja programske podrške.

Leap Motion ne utvrđuje da li se radi o lijevoj ili desnoj ruci, ali je moguće napraviti programsko razlikovanje. Više od dvije ruke se mogu pojaviti u listi ruku unutar okvira ako je više ruku ili drugih rukolikih objekata prisutno unutar vidnog polja, međutim preporučuju se najviše dvije ruke za optimalnu kvalitetu praćenja pokreta.

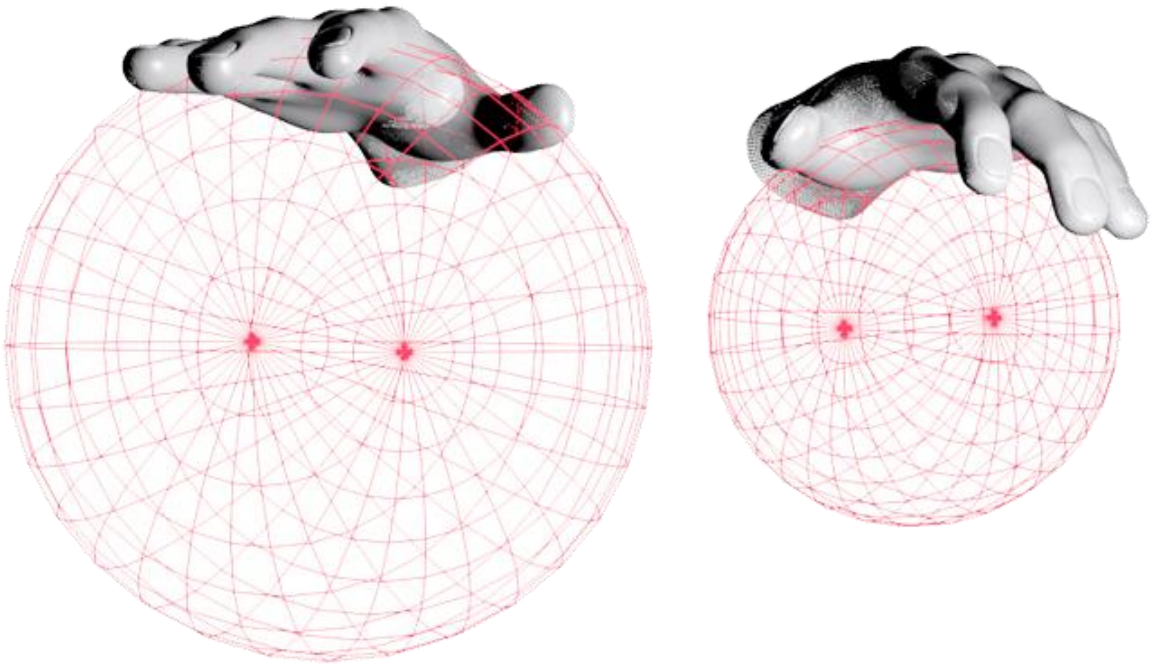
2.2.2.1 Atributi ruke

Objekt `Hand` ima nekoliko atributa kojima se opisuju fizikalne karakteristike detektirane ruke:

- `PalmPosition` – pozicija centra dlana izmjereno u milimetrima od ishodišta Leap Motion uređaja.
- `PalmVelocity` – Brzina dlana u milimetrima po sekundi.
- `PalmNormal` – Jedinični vektor okomit na ravninu koju gradi dlan ruke. Vektor pokazuje u smjeru izvan dlana prema dole.
- `Direction` – Jedinični vektor smjera koji pokazuje od centra dlana prema prstima.
- `SphereCenter` – Centar sfere prilagođene krivini ruke (kao da drži loptu).
- `SphereRadius` – Radijus sfere prilagođene krivini ruke. Radijus se mijenja s oblikom ruke.



Slika 2.3 Vektor normale i smjera



Slika 2.4 Promjena veličine sfere i zakrivljenosti prstiju

2.2.2.2 Pokreti ruke

Objekt `Hand` također pruža nekoliko atributa koji opisuju kretanje detektirane ruke između okvira. Leap Motion analizira kretanje ruke, kao i asociranih prstiju i alata, te obrađuje reprezentativne translacijske, rotacijske i faktore skaliranja. Pokreti rukom unutar vidnog polja uređaja rezultiraju translacijom; okretanje, zakretanje i naginjanje ruke rezultira rotacijom; micanje prstiju ili alata prema i jedno od drugih izaziva skaliranje.

Kao što je slučaj kod okvira, pokreti ruku su izvedeni uspoređujući karakteristike ruke u trenutnom okviru s određenim ranijim okvirom. Atributi su identični kao kod okvira:

- `RotationAxis` – Vektor smjera za os rotacije.
- `RotationAngle` – Kut rotacije oko osi rotacije u smjeru kazaljke na satu.
- `RotationMatrix` – Transformacijska matrica za rotaciju.
- `ScaleFactor` – Faktor skaliranja (ekspanzija i kontrakcija).
- `Translation` – Faktor translacije.

2.2.2.3 Lista prstiju i alata

Prstima i alatima asociranim s rukom pristupamo korištenjem jedne od tri liste:

- `Pointables` – Prsti i alati skupa čine `Pointable` objekt.
- `Fingers` – Samo prsti.
- `Tools` – Samo alati.

Pojedinačni prst ili alat može se naći i korištenjem vrijednosti identifikatora dobivenog u prethodnom okviru korištenjem `Hand.Finger()` i `Hand.Tool()`, ili ako ne trebamo razlikovati prste ili alate, `Hand.Pointable()` metode. Ove metode vraćaju referencu na odgovarajući objekt ako postoji u trenutnom okviru. Ako prst ili alat nije asociran s rukom u tom okviru onda se vraća invalid objekt.

2.2.3. Modeli prstiju i alata

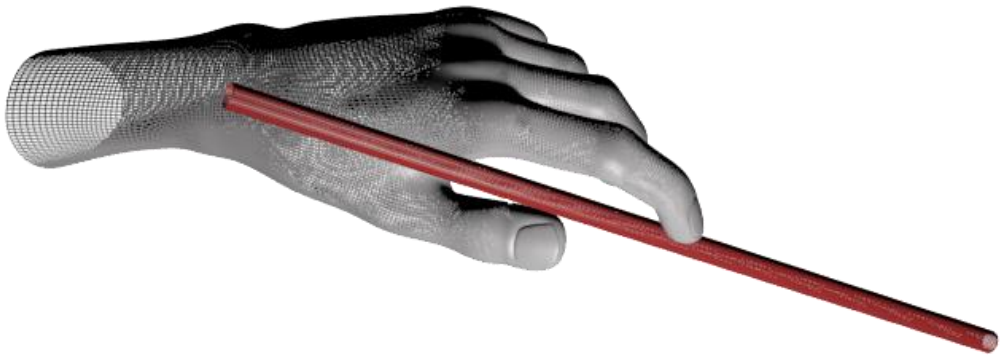
Leap Motion detektira i prati prste i alate unutar svog vidnog polja. Uređaj klasificira prstolike objekte prema njihovom obliku. Razlika između prstiju i alata su logika i kriteriji koji se koristi za njihovo razlikovanje: oblik, duljina i debljina. Prst je kraći, deblji i ima nagib, dok je alat dulji, tanji i ravniji od prsta.

U Leap Motion modelu, fizikalne karakteristike prstiju i alata su izdvojene u `Pointable` objekt i nude jednake informacije u obliku atributa. Prsti i alati su tipovi pokazivačkih objekata. Fizikalne karakteristike pokazivačkih objekata uključuju:

- `Length` – Duljina vidljivog dijela objekta.
- `Width` – Prosječna širina vidljivog dijela objekta.
- `Direction` – Jedinični vektor smjera istog smjera kao i objekt.
- `TipPosition` – Pozicija vrha pokazivača u milimetrima.
- `TipVelocity` – Brzina vrha pokazivača u milimetrima po sekundi.

Slika 2.5 Pozicija vrha prstiju i smjer u kojem pokazuju

Leap Motion klasificira detektirani pokazivački objekt kao prst ili alat. `Pointable.IsTool` atribut se koristi kako bi utvrdili koji od njih dva se koristi kao `Pointable` objekt.



Slika 2.6 Primjer alata

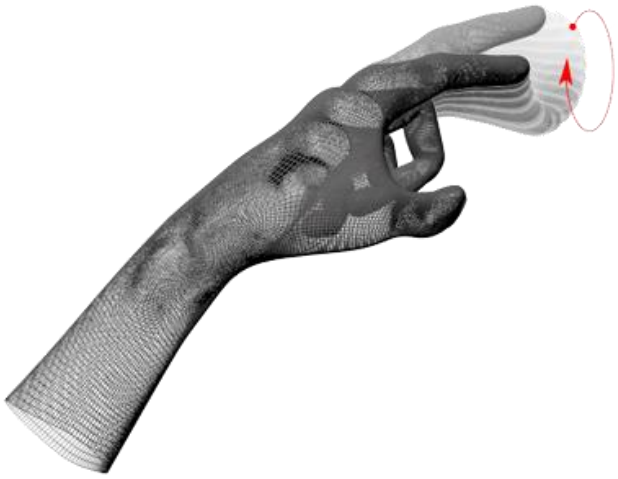
2.2.4. Geste

Ljudi koriste geste, pokrete dijela tijela u određenom obliku, kako bi izrazili ideje, naredbe ili značenje. Raspoznavanje gesti predstavlja sposobnost računala da primjenjuju algoritme kako bi interpretirali govor ljudskog tijela. Geste su u svojoj jednostavnosti samo skup pokreta koji su definirani kao gesta. Razlikujemo dvije grupe gesti: *offline* i *online* geste. *Offline* geste su geste koje se procesiraju nakon korisnikove interakcije s objektom. *Online* geste su geste koje imaju direktan utjecaj na objekt. Odabir izbornika je primjer *offline*, dok je skaliranje ekrana primjer *online* geste. Leap Motion API posjeduje mogućnost raspoznavanja nekih osnovnih gesti, ali je moguća i programska implementacija novih. Treba imati na umu da je interakcija s Leap Motionom treba biti prirodna i ne ovisiti o velikom broju kompliciranih gesti. Geste su često specifične za određenu kulturu i mnoge neće imati smisla, te neće biti prirodne za sve korisnike.

Leap Motion sprema podatke o gestama unutar okvira na isti način kao što to radi s rukama i prstima. Za svaku prepoznatu gestu, dodaje `Gesture` objekt u okvir. `Gesture` objekt se može dohvatiti iz liste gesti unutar okvira (`Frame` objekta).

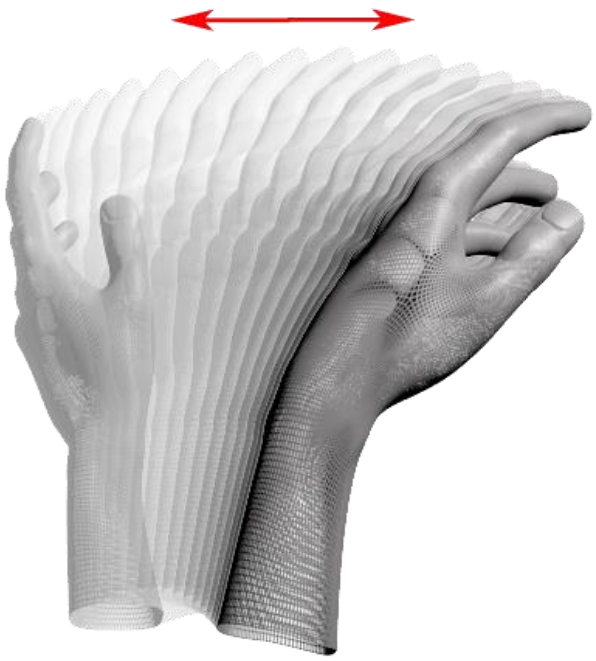
Sljedeći uzorci se prepoznaju kao ugrađene geste:

- Circle – Kružni pokret jednim prstom ili alatom.



Slika 2.7 Kružna gesta

- Swipe – Linearni pokret ruke.

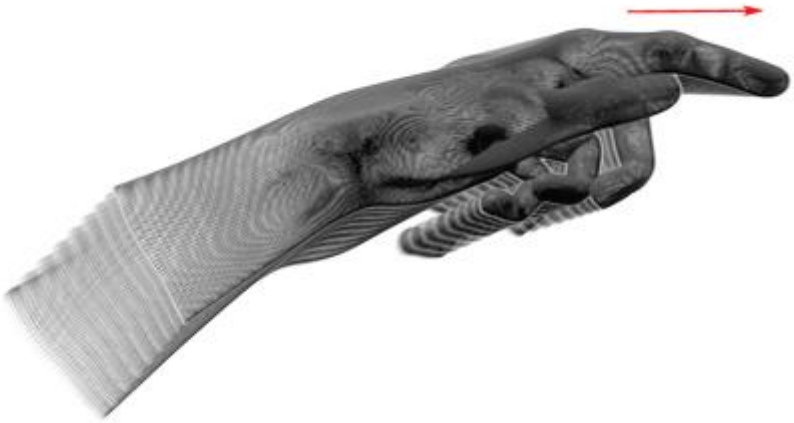


Slika 2.8 Swipe gesta

- Key Tap – Pokret prsta kao da dotaknemo gumb na tipkovnici.

Slika 2.9 Key Tap gesta

- Screen Tap – Pokret prsta kao da dotaknemo gumb na ekranu.



Slika 2.10 Screen Tap gesta

Kada Leap Motion klasificira pokret kao gestu, dodaje `Gesture` objekt u okvir. Ako se dotična gesta nastavlja izvoditi, Leap Motion dodaje osvježenu verziju `Gesture` objekta u buduće okvire. Geste `Circle` i `Swipe` su kontinuirane i Leap Motion osvježava nastavak ovih gesti u svakom okviru. S druge strane, `Key Tap` i `Screen Tap` su diskretne geste, te se obje zapisuju kao jedan jedinstven `Gesture` objekt.

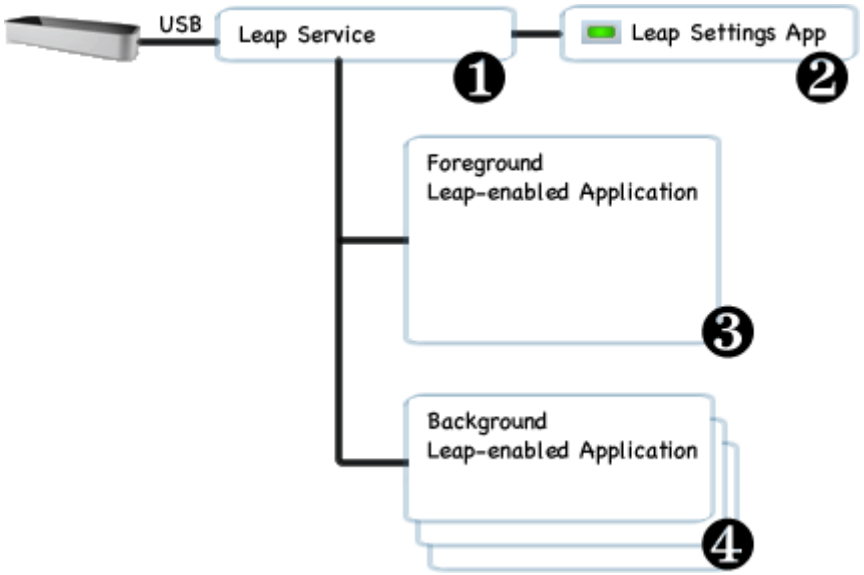
Da bi koristili geste u našoj aplikaciji moramo omogućiti prepoznavanje svake pojedinačne geste koju koristimo. `Controller` klasa sadržava `EnableGesture()` metodu za tu svrhu.

2.3. Leap Motion arhitektura

Leap Motion uređaj podržava sve popularne operacijske sustave (Linux, Windows i Mac). Leap Motion programska podrška se vrti kao servis (Windows) ili *daemon* (Mac i Linux). Programska podrška se spaja na uređaj preko USB priključka i komunicira s aplikacijama korištenjem TCP porta 127.0.0.1:5905. Aplikacijama koje ga koriste se šalju podaci o praćenju pokreta. Leap Motion SDK sadrži dvije vrste API-ja za dohvaćanje podataka: nativno aplikacijsko sučelje i WebSocket sučelje. Oni omogućuju korištenje nekoliko programskih jezika, uključujući JavaScript. (C++, <objective-C, C#, Java, Python)

2.3.1. Aplikacijsko sučelje

Aplikacijsko sučelje [2] je omogućeno kroz dinamično učitanu biblioteku. Ona se povezuje s Leap Motion servisom i dohvaća podatke o praćenju za našu aplikaciju. S bibliotekom se možemo direktno povezati C++ i Objective-C aplikacijama, ili preko poveznica ponuđenih za C#, Javu, Python, ili neki drugi podržani jezik.



Slika 2.11 Rad aplikacije

1. Leap Motion servis prima podatke od uređaja preko USB priključka. On obrađuje primljene informacije i šalje ih pokrenutoj aplikaciji. Servis obično šalje podatke samo površinskoj aplikaciji, međutim i aplikacije u pozadini ih mogu primati po zahtjevu korisnika.
2. Leap Motion aplikacija radi odvojeno od servisa i dozvoljava korisniku konfiguraciju samog Leap Motiona.
3. Površinska Leap Motion aplikacija prima od servisa podatke o praćenju.
4. Ukoliko se površinska aplikacija sruši, Leap Motion servis joj prestaje slati podatke. U tom slučaju aplikacije u podlozi mogu zatražiti dozvolu za primanje tih podataka.

2.3.2. Sučelje WebSocket

Ovo sučelje je primarno namijenjeno za web aplikacije, ali svaka aplikacija koja može uspostaviti vezu WebSocketom ga može koristiti. Poslužitelj zadovoljava RFC6455.

2.4. Programska podrška i alati

Leap Motion dolazi sa skupom alata kojima možemo optimizirati njegov rad, te eventualno stupiti u kontakt s proizvođačem ukoliko dođe do neke nepoznate greške. Na stanje o ispravnosti uređaja nam ukazuje nekoliko stvari: indikatorska svjetala, programski vizualizator, i alati za dijagnostiku kojima radimo dublji uvid u to što sve i kako uređaj mjeri. Također, Leap Motion se posebno kalibrira za svako računalo, te možemo i postaviti i način rada u kojem ga koristimo:

- *Precision* – Daje prednost preciznosti nad brzinom.
- *Balanced* – Balansira brzinu i preciznost.
- *High Speed* – Daje prednost brzini nad preciznosti.

3. Uređaj Oculus Rift

Uređaj Oculus Rift je HMD namijenjen za stvaranje osjećaja virtualne stvarnosti. Iako je primarno zamišljen kao HMD eksplicitno namijenjen igrama, njegova uporaba obećava puno više. Njegova stereoskopska leća su najprirodniji način za gledanje 3D sadržaja. Predstavlja prvi proizvod ove kvalitete koji je dostupan širim potrošačima po niskoj cijeni.

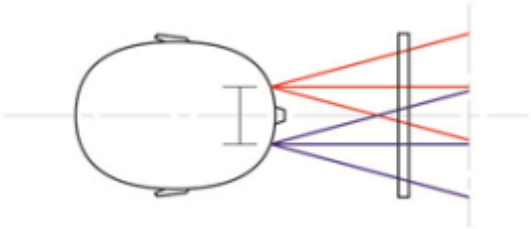
Još uvijek se nalazi u fazi razvoja dok je u radu korištena razvojna verzija (*Developer Kit*). Ova verzija je pojednostavljena verzija finalnog proizvoda i nema sve njegove mogućnosti, što objašnjava nisku rezoluciju prikaza i nedostatak pozicijskog praćenja (prati rotaciju glave, ali ne točnu poziciju).

Komercijalna verzija bi trebala pratiti orijentaciju glave, rotaciju i poziciju u prostoru te prikazati ih zanemarivo velikom brzinom, iako kombinacija širokog vidnog polja s praćenjem pokreta glave i stereoskopskim 3D-om već sada stvara udubljeno virtualno iskustvo [3].

Slika 3.1 Razvojna verzija uređaja Oculus Rift

3.1. Tehničke specifikacije

Oculus Rift prikazuje scenu kao dvije stereo slike, pola ekrana korišteno za svako oko. To znači da će ista slika morati dva puta iscrtati. Iako je prvi prototip uređaja Oculus Rift imao ekran od 5 inča, razvojna verzija koristi panel od 7 inča, što čini uređaj nešto većim. Međutim, izmjena slikovnih elemenata novog panela je znatno veća, smanjujući kašnjenje i *blur* efekt pri brzim pokretima glave. Također ispunjenost slikovnih elemenata je bolja, što smanjuje *screen-door effect*. LCD je svjetliji s dubinom boje od 24 bita po sekundi. Stereoskopski 3D razvojne verzije se ne preklapa i proširuje prostor vida lijevog i desnog oka čime oponaša stvarno ljudsko oko. S tim stereoskopski 3D stvara pogled s izvrsnom dubinom, skaliranjem i paralaksom. Za razliku od 3D iskustva na ekranu ovo se postiže prikazujući jedinstvene i paralelne slike za svako oko [4].



Slika 3.2 Pogled s HMD-om

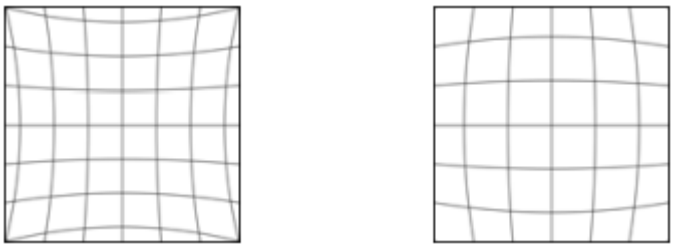
Vidno polje uređaja je više od 90 stupnjeva horizontalno i 110 stupnjeva dijagonalno, što je duplo više od danas dostupnih FOV uređaja. Namijenjeno je da prekrije skoro cijelo vidno polje korisnika i izolirajući ga od vanjskog svijeta stvori jak osjećaj uronjenosti. Fokalna duljina je beskonačna, što znači da je sve u fokusu i vid virtualnog svijeta nije više ograničen ekranom, nego samo onim što naše oči mogu vidjeti. Rezolucija uređaja je 1280x800 (omjer slike 16:10), što je efektivnih 640x800 po oku (omjer slike 4:5).

Kako Oculus Rift nema 100% preklapanje između očiju, kombinirana horizontalna rezolucija je nešto veća od 640. Očekuje se da će u konačnoj verziji rezolucija panela porasti na barem 1920x1080. Slika za svako oko se prikazuje uvećano kako bi se povećalo vidno polje, međutim posljedica toga je jastučna (*pincushion*) distorzija. Ova distorzija se korigira programskom implementacijom bačvaste (*barrel*) distorzije.

$$(r, \theta) \rightarrow (f(r)\, r, \theta) \qquad (1)$$

$$f(r) = k_0 + k_1 r^2 + k_2 r^4 + k_3 r^6 \qquad (2)$$

Ova dva modela su izražena u obliku udaljenosti r od centra. Model distorzije izvodi transformaciju uz zadanu funkciju skaliranja. Smjer θ se ne mijenja, dok koeficijenti k upravljaju distorzijom. Nakon izračuna skaliranja i pomaka distorzije, još je potrebno ispraviti vidno polje



Slika 3.3 *Pincushion* i *barrel* distorzija

Oculus Rift je dizajniran da bude ugodan i lagan za nošnje na glavi. Trenutni razvojni uređaj ima 369 grama, kao teže skijaške naočale.

Uređaj omogućuje modificiranje blizine ekrana očima, kao i izmjenu leća kako bi ga prilagodili profilima različitih ljudi. Oculus Rift ima DVI i HDMI ulaz na upravljačkoj kutiji, dok se podaci o praćenju šalju preko USB sučelja.

3.1.1. Praćenje glave i koordinatni sustav

Oculus Rift koristi *Adjacent Reality Tracker* komponentu za praćenje čime postiže praćenje glave od 360 stupnjeva s jako niskim kašnjenjem. Svaki suptilan pokret glave se prati u stvarnom vremenu pružajući prirodno i intuitivno iskustvo u svim mogućim smjerovima. Kašnjenje na razvojnoj verziji iznosi 40 do 60 ms, s težnjom da u potrošačkoj verziji bude manja od 20 ms. Uz modificiran *firmware* radi na frekvenciji od 1000 Hz. Koristi kombinaciju žiroskopa, akcelerometra i magnetometra kroz 3 osi, čime može pratiti apsolutnu orijentaciju glave bez skretanja i s 6 stupnjeva slobode. Informacije dobivene od

senzora se kombiniraju postupkom fuzije senzora (*sensor fusion*) kako bi dobili orijentaciju glave u stvarnom svijetu i sinkroniziramo ju s virtualnom perspektivom korisnika [4].

Najvažniji dio postupka fuzije senzora je integracija podataka kutne brzine dobivene od žiroskopa. U svakom malom vremenskom intervalu mjeri se kutna brzina (3).

$$\omega = (\omega_x, \omega_y, \omega_z) \tag{3}$$

$$l = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{4}$$

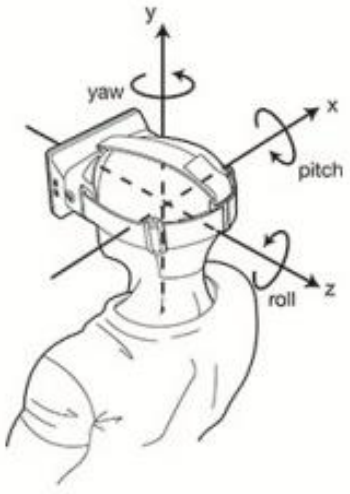
Svaka komponenta je stopa rotacije u radijanima po sekundi oko odgovarajuće osi. Uz duljinu ω (4) možemo iz vektora kružne brzine dobiti trenutnu os rotacije kao normaliziranu vrijednost žiroskopa, kao i stopu rotacije oko te osi. Zbog visoke stope uzorkovanja svih senzora i preciznosti žiroskopa, orijentacija se dobro održava dugo vremena. Međutim, eventualno se pojavi greška pomaka. Srećom, senzori uređaja Oculus Rift nude dovoljno referentnih podataka da kompenziraju tu grešku. Greška pomaka je rotacija koja nastaje iz odnosa stvarne orijentacije i kalkulirane orijentacije. Greška pomaka se može podijeliti na dvije greške:

- Greška nagiba – Promjena orijentacije u *pitch* i *roll* komponenti. Smanjena mjerenjem gravitacijskog vektora akcelerometrom.
- *Yaw* greška – Promjena orijentacije oko Y osi. Korekcije se rade mjerenjima Zemljinog magnetskog polja magnetometrom.

Ove promjene su relativne u odnosu na fiksirani, globalni koordinatni sustav u kojem Y uvijek pokazuje prema gore. Z-os je proizvoljna, dok se X

Orijentacija se sprema kao rotacija u desnom koordinatnom sustavu, što znači: Y je pozitivan prema gore, X je pozitivan prema desno, i Z je pozitivan prema nazad. Rotacija se može spremati i u *yaw-pitch-roll* obliku. Pozitivna rotacija je u smjeru suprotnom od kazaljke na satu kada gledamo iz pozitivnog prema negativnom smjeru svake osi, dok su komponente rotacije sljedeće:

- *Pitch* – Rotacija oko X, pozitivna prema gore.
- *Yaw* – Rotacija oko Y, pozitivna prema lijevo.
- *Roll* – Rotacija oko Z, pozitivna kada naginjemo glavu lijevo u XY ravnini.



Slika 3.4 Oculus Rift koordinatni sustav

3.1.2. Programska podrška

Igre i platforme za igre moraju biti specifično dizajnirani da rade ispravno s uređajem Oculus Rift. Oculus SDK već sad nudi inženjerima programske potpore brojne mogućnosti za integriranje uređaja s njihovim platformama. SDK uključuje kod, primjere i dokumentaciju. Integracija je počela s osobnim računalima i pametnim telefonima, te će se proširiti na igraće konzole.

U međuvremenu mnoge kompanije podržavaju Oculus Rift sa svojim igrama, iako je tek u fazi razvoja. Tako, SDK sadrži integraciju s nekim popularnim okruženjima: *Unreal Development Kit*,i *Unreal Engine*, te *Unity 4*.

Uporabom besplatnih *Dynamic Digital Depth* i *Vireio Perception VR*, ili komercijalnih *VorpX* drivera olakšano je prebacivanje klasičnih programa s 2D ekrana u virtualno 3D okruženje.

Oculus Rift i Oculus SDK trenutno podržavaju Windows, Mac i Linux operacijske sustave.

4. Uporaba i korisnost u oblikovanju aplikacija

Iskustvo korisnika (engl. *user experience*) uključuje ponašanja, stav i emocije osobe koje nastaju korištenjem nekog proizvoda, sustava ili servisa. Uključuje praktične, eksperimentalne, afektivne i značajne aspekte interakcije čovjeka i računala. Dodatno, usko je povezana s upotrebljivosti (engl. *usability*) aspektom koji obuhvaća ljudsku percepciju nekih gledišta sustava, poput korisnost, lakoća, efektivnost i elegantnost korištenja [5]. Subjektivne i dinamične su prirode, ali se njihovi koncepti mogu primijeniti u ovisnosti širokoj uporabi i kontekstu korištenja.

Kako se ovdje radi o uređaju koji je ovisan o vizualnoj programskoj podršci nju uključujemo kao najbitniju komponentu. Kako Oculus Rift predstavlja uređaj za vizualizaciju, treba prilikom modeliranja sustava imati na umu perspektivu u kojoj se koristi – prvo lice. S druge strane, Leap Motion pruža puno veću interaktivnost i tu korisnikova percepcija dolazi do većeg izražaja. Neke smjernice o kojima treba voditi računa su:

- Simbologija gesti može biti teška za naučiti, tako da ne treba pretjerivati s njihovom uporabom.
- Inspiracija za korištenje uređaja bi trebala dolaziti iz fizikalnog svijeta i realnih ponašanja ljudi.
- Pri tome ne treba biti ograničen fizikalnim svijetom, nego ograničenja koje on ima u virtualnom svijetu ukloniti.
- Korisniku davati povratne informacije o njegovim akcijama kako bi preciznije postigao ono što mu je namjera.
- Vizualne reprezentacije ruku, prstiju i alata bi trebale biti jednostavne, funkcionalne i neupadljive.
- Destruktivne akcije i geste se trebaju obraditi s većom pažnjom.
- Navigacija i interakcija u aplikaciji bi trebale imati jasnu granicu kako ne bi zbunile korisnika.
- Općenito treba pretpostaviti da korisnik nema nikakvih uputa za korištenje aplikacije.

5. Korištene tehnologije

Leam Motion i Oculus Rift biblioteke su napisana u programskom jeziku C++.

Leap Motion koristi SWIG, alat otvorenog koda za generiranje poveznica s drugim programskim jezicima (C#, Java, Python). SWIG generirane poveznice prevode pozive iz jednog programskog jezika u C++ pozive Leap Motion biblioteke. Svaka SWIG poveznica koristi dvije dodatne biblioteke karakteristične za pojedini jezik.

S druge strane, Oculus Rift ne pruža takve poveznice s drugim jezicima osim C++, ali nudi integraciju s Unity razvojnim okruženjem, čime otvara mogućnost korištenja svih programskih jezika podržanih od Unity okruženja. Službeno je podržana samo Unity profesionalna verzija.

Kod Leap Motiona definicije klasa za programski jezik C# ponuđene su za .NET 3.5 i .NET4.0 kao posebne biblioteke na koje se referenciramo. Ove biblioteke učitaju odgovarajuću „LeapCSharp“ biblioteku, ovisno o platformi, koja nadalje učitava pripadnu „Leap“ biblioteku. .NET 3.5 biblioteke se koriste s Unity razvojnim okruženjem. Za standardnu, besplatnu, licencu potrebno je ručno napraviti strukturu direktorija kako bi biblioteke bile pronađene, dok za profesionalnu verziju postoji *plug-in*.

Unity i programski jezik C# su odabrani kao najpogodnija zajednička točka ova dva uređaja kako bi se izradio praktični dio rada. Također za potrebu izrade jednostavnog 3D modela je korišten alat za modeliranje Cinema 4D.

5.1. Razvojno okruženje Unity

Unity je ekosustav za razvijanje video igara i ostalog interaktivnog 2D i 3D sadržaja. Uključuje jako moćan grafički pogon (eng. *engine*) i skup alata koji omogućuju intuitivno i brzo stvaranje funkcionalnih svjetova za različite platforme. Trenutno podržava razvijanje za iOS, Android, Windows, BlackBerry 10, OS X, Linux, web preglednike, Flash, PlayStation 3, Xbox 360, Windows Phone 8, i Wii U. Unity povećava produktivnost i znatno smanjuje vrijeme u nastajanju programske podrške, efektivno smanjujući trošak. Koristimo ga za organiziranje resursa koje imamo na raspolaganju, dodavanje svjetlosti, efekata, fizike i animacije, testiranje i uređivanje projekta, kao i za njegovu objavu u izvršnoj verziji. Resursi uključuju, među ostalim, 3D modele, materijale, teksture, zvukove, slike, programske skripte i fontove. Prihvaća jako velik broj formata čineći sustav resursa jako robusnim i inteligentnim. Sve resurse hijerarhijski organiziramo u scenama [6].

Skriptama opisujemo ponašanje naših resursa – programskim kodom činimo našu scenu i resurse interaktivnim. Više skripti možemo vezati za isti objekt, što ohrabruje ponovnu uporabu koda. Unity podržava tri različita programska jezika: C#, JavaScript i Boo. U jednom projektu možemo koristiti više programskih jezika.

Uz jako aktivnu zajednicu, Unity je savršen izbor za male kompanije i neovisne inženjere programske potpore. Dolazi u dvije verzije: besplatna i profesionalna.

5.1.1. Integracija s uređajem Leap Motion

Za izradu rada je korištena profesionalna verzija razvojnog okruženja Unity. Kako bi povezali Leap Motion uređaj s okruženjem dovoljno je prekopirati *Plugins* direktorij iz Leap Motion SDK direktorija u *Assets* direktorij našeg projekta. *Plugins* direktorij sadrži biblioteke i dodatke neophodne za povezivanje Unity okruženja sa svim popularnim operacijskim sustavima.

Struktura Unity skripte za Leap Motion uključuje Leap prostor imena i koristi `Controller()` klasu za upravljanje. U Unity okruženju nije potrebno koristiti Leap.Listener podklasu. Kako Unity aplikacije imaju prirodni *frame-rate* `Update()` metoda dohvaća trenutni okvir s podacima od Leap Motion kontrolera kad se pozove.

5.1.2. Integracija s uređajem Oculus Rift

Oculus Rift pruža jednostavnu integraciju s Unity razvojnim okruženjem u obliku Unity paketa koji uvezemo u naš projekt, čime se za njega vežu sve biblioteke koje su neophodne za jedan Oculus projekt [7].

Struktura direktorija koja nakon toga nastaje uključuje dva tipa resursa:

- OVR – Direktorij najvišeg nivoa za Oculus-Unity integraciju. Sadrži sljedeće komponente:
    - Editor – Skripte koje proširuju funkcionalnost Unity Editora.
    - Materials – Materijali za grafičke komponente integracije.
    - OVRImageEffects – Skripte i konfiguracije za povezivanje slika s lećama.
    - Prefabs – Prefab komponente za povezivanje uređaja s Unity scenom.
    - Resources – Komponente potrebne za rad OVR skripti.
    - Scripts – C# datoteke za povezivanje integracije.
    - Shaders – Programi za sjenčanje korišteni u integraciji.
    - Textures – Slikovne komponente koje koriste skripte.
- *Plugins* – Sadrži biblioteke za komunikaciju uređaja s Unity okruženjem za različite operacijske sustave (u radu je korišten OculusPlugin.dll za Windows).

5.1.2.1 Ugrađene skripte

- `OVRDevice` – Glavno sučelje prema Oculus Rift uređaju. Ima poveznice sa svim napravljenim C++ funkcijama, kao i pomoćne funkcije za postavljanje ponašanja kamere.
- `OVRComponent` – Klasa s osnovnim funkcijama mnogih drugih OVR klasa.
- `OVRCamera` – Koristi se za prikaz u Unity Camera klasi.
- `OVRCameraController` – Jednostavno sučelje za ubacivanje upravljanje Oculus Rift kamere unutar Unity okruženja.
- `OVRPlayerController` – Implementirani kontroler za navigaciju kamere virtualnim okruženjem u prvom licu.
- `OVRMainMenu` – Koristi se za upravljanje učitavanja različitih scena.
- `OVRGamepadController` – Sučelje za *gamepad* kontroler.
- `OVRPresetManager` – Pomoćna klasa za upravljanje nekim Unity varijablama.
- `OVRCrosshair` – Dodaje stereoskopski nišan u scenu.
- `OVRGUI` – Pomoćna klasa koja enkapsulira prikaz teksta u 2D ili 3D okruženju.
- `OVRMagCalibration` – Pomoćna klasa za kalibraciju magnetometra za korekciju pomaka u Y-osi.
- `OVRMessenger` – Omogućuje razmjenu naprednih poruka između različitih klasa.
- `OVRUtils` – Kolekcija statičkih funkcija za korištenje.

6. Implementacija

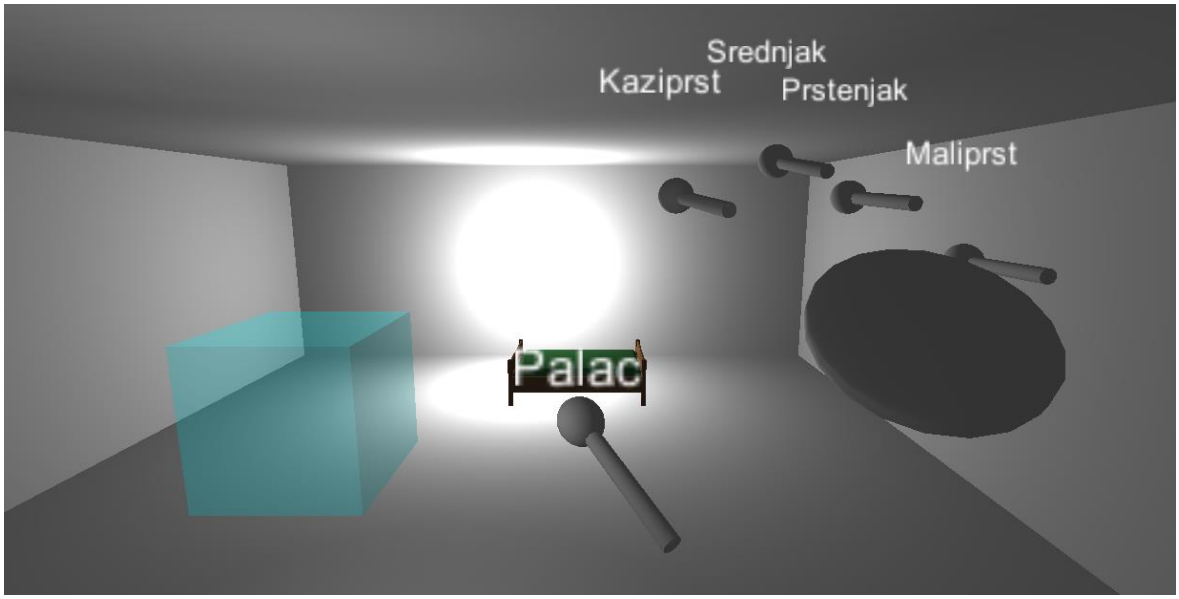
Za potrebe praktičnog dijela diplomskog rada napravljena je Unity aplikacija koja demonstrira rad uređaja Leap Motion u 3D okruženju. Uz to je izvedeno promatranje scene s uređajem Oculus Rift, koji skupa sa senzorom pokreta pruža osjećaj virtualne stvarnosti. U projektu su napravljene 4 glavne:

1. Vizualizacija gesti.
2. Interakcija dodirom s 3D objektom.
3. Interakcija objektom uz kretanje.
4. Demonstracija korištenja sile

Za sve scene je pomoću Unity Editora izgrađena jedna jedinstvena testna soba unutar koje se odvijaju demonstracije. Izbor početne scene radimo prstima ili pokazivačkim predmetima preko uvodnog izbornika označenog kao nulta scena. Izbor je moguće napraviti i pritiskom odgovarajućeg broja na alfanumeričkoj tipkovnici (1, 2, 3, 4, 0) – na taj način možemo i resetirati trenutnu scenu.

6.1. Vizualizacija gesti

U prvoj sceni je na pasivan način demonstrirana interakcija s objektom. Ovdje je više naglasak na mogućnostima Leap Motion uređaja, te načina na koji možemo koristiti neke od ugrađenih gesti.



Slika 6.1 Prva scena

Kružna gesta nam omogućuje rotaciju objekta oko njegove vertikalne ose. Smjer rotacije je određen tipom kružne geste: u smjeru kazaljke na sata ili u smjeru suprotnom od kazaljke na satu. *Swipe* gesta je kalibrirana da prepoznaje geste prema lijevo, desno, gore i dole. S obzirom na smjer geste pomjeri objekt na tu lokaciju u sceni po X i Y osi. Gesta dodira ekrana zamjenjuje objekt s modelom kreveta, dok gesta dodira tipke mijenja boju kocke.

Skripta `GesteController` pokreće prikaz gesti koje inicijaliziramo u `LeapManager` skripti. Geste se pokreću kao `event` svaki put kada se stvori nova instanca.

```
kontroler.EnableGesture(Leap.Gesture.GestureType.TYPECIRCLE);
kontroler.EnableGesture(Leap.Gesture.GestureType.TYPEKEYTAP);
kontroler.EnableGesture(Leap.Gesture.GestureType.TYPESCREENTAP);
```

```
kontroler.EnableGesture(Leap.Gesture.GestureType.TYPESWIPE);
```

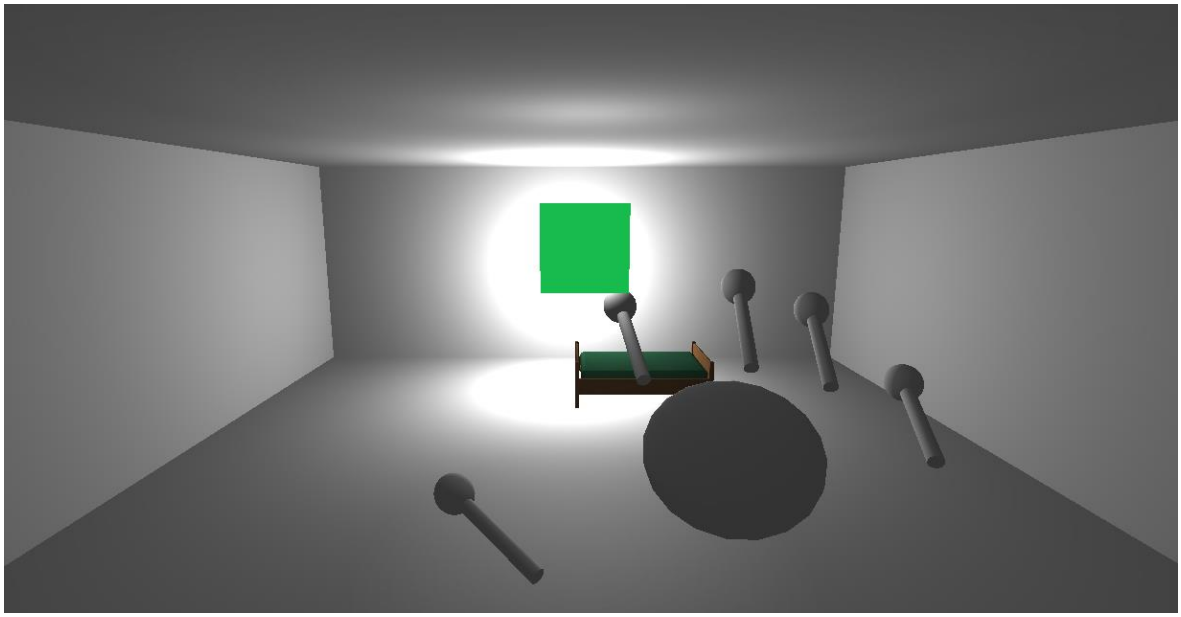
Ona ujedno povezuje gestu s reprezentativnim vizualnim znakom koji će se pokrenuti u slučaju pojave geste. Pripadne klase za vizualizaciju su `GesteCirclePrikaz`, `GesteSwipePrikaz`, `GesteKeyTapPrikaz`, `GesteScreenTapPrikaz`.

Unutar ove scene je napravljen i izračun pojedinih prstiju i na njihove vrhove u virtualnoj sceni zalijepljen natpis imena prstiju. To nam omogućuje da steknemo bolji uvid na koji način bi uređaj mogao razlikovati prste. Izračun je napravljen tako da je prvo pronađena središnja točka pronađenih prstiju, te na osnovu toga udaljenosti najbližeg prsta sredini. Relativno u odnosu na palac kao najudaljeniji prst od srednje je sortiran ostatak.

```
float maxUdaljenost = 0f;
GameObject palac = null;
for (int i = 0; i < list.Count; ++i)
{
      GameObject prst = list[i];
      GameObject vrh = GetTip(prst);
      float udaljenost = Vector3.Distance(sredina,
                          vrh.transform.position);
      if (i == 0)
      {
            maxUdaljenost = udaljenost;
            palac = list[i];
      } else if (udaljenost > maxUdaljenost)
      {
            maxUdaljenost = udaljenost;
            palac = list[i];
      }
}
```

6.2. Interakcija dodirom s 3D objektom

Druga scena ima fokus na upravljanju s objektom direktnim dodirom. Nakon što vrhovi prstiju dodirnu objekt on postaje označen, te ga je moguće proizvoljno translatirati. Korištenjem *pinch* geste (2 prsta) sa selektiranom rukom objekt mijenja veličinu – skaliramo ga. Kako bi manipulacija bila robusnija, rotacija je moguća pomoću dva ili više prstiju samo kad je druga ruka vidljiva u sceni. Objekt se sam deselektira ukoliko izgubi kontakt s pokazivačkim predmetima ili neko vrijeme nije promijenio stanje.



Slika 6.2 Druga scena

`LeapController` skripta je organizirana kao *singleton* [10] oblikovni obrazac kako bi se omogućio lakši pristup od ostalih klasa. Skripta `LeapSelectionController` sadrži logiku za označavanje i translatiranje objekta pomoću Leap Motion uređaja. U slučaju dodira zrake iz vrha prstiju s graničnim okvirom objekta označenog kao „Dodirljiv“ objekt postaje označen te ga možemo manipulirati. Na sam dodir provjeravamo da li je već selektiran, te prste kojima je dodirnut šaljemo na obradu (gdje postavljamo stanje selektiranog objekta).

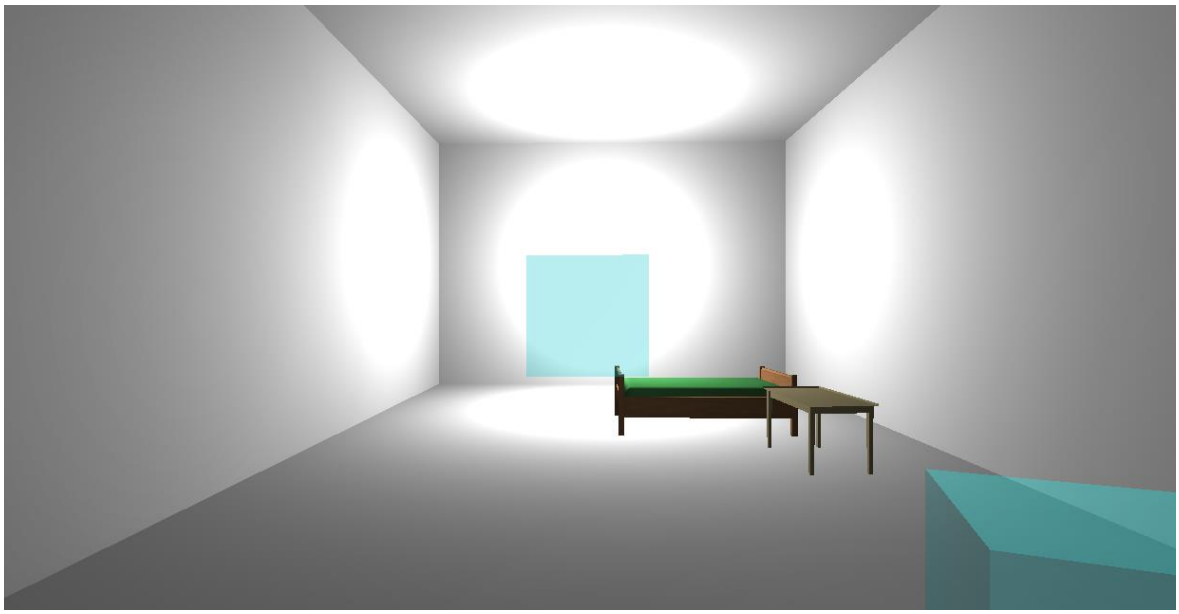
```
If ( !selektiran && granicniOkvir.gameObject != fokusiranObj )
{
      ClearFocus();
      SetFocus(granicniOkvir.gameObject);
}
If ( !dodirniPrsti.Contains(prst) )
{
      dodirniPrsti.Add(prst);
      dodirniPrstiLok.Add(prst.transform.position);
}
```

U svakom okviru se provjerava da li je napravljena manipulacija objekta, te u slučaju da vrijedi uvjet, recimo za rotaciju, rotiramo objekt relativno i pomoću prstiju kojima dodirnut.

```
if (LeapController.enableRotation && dodirniPrsti.Count > 1)
{
      Vector3 zadnjiV = dodirniPrstiLok [1] - dodirniPrstiLok [0];
      Vector3 trenutniV = dodirniPrsti [1].transform.position -
dodirniPrsti [0].transform.position;
      if(zadnjiV!= trenutniV)
      {
            Vector3 os = Vector3.Cross(currVec, lastVec);
            float zadnjaD = zadnjiV.magnitude;
            float trenutnaD = trenutniV.magnitude;
            float osD = os.magnitude;
            float kut = -Mathf.Asin (osD/(zadnjaD * trenutnaD));
            fokusiranObj.transform.RotateAround(os/osD, kut);
      }
}
```

6.3. Interakcije objektom uz kretanje

Kao nadogradnja druge scene napravljena je treća scena, gdje su označavanje i prijenos objekta izvedeni na korisnicima pristupačniji način. Scenu upravljamo jednom rukom. Kamera se kreće naprijed i nazad u ovisnosti da li je ruka bliže ili dalje od imaginarnog ekrana. Objekt podižemo tako da napravimo šaku – pokupimo ga, Jedna ruka služi za podizanje i translatiranje.



Slika 6.3 Treća scena

Nakon što dohvatimo prvu ruku koja se pojavi u vidnom polju uređaja Leap Motion radimo obradu. U slučaju zatvaranja ruke u šaku tražimo objekt za pokupiti – pucamo zrake u obliku sfere, te prvi objekt koji dodirnemo bude selektiran. Selektirani objekt dovučemo do kamere.

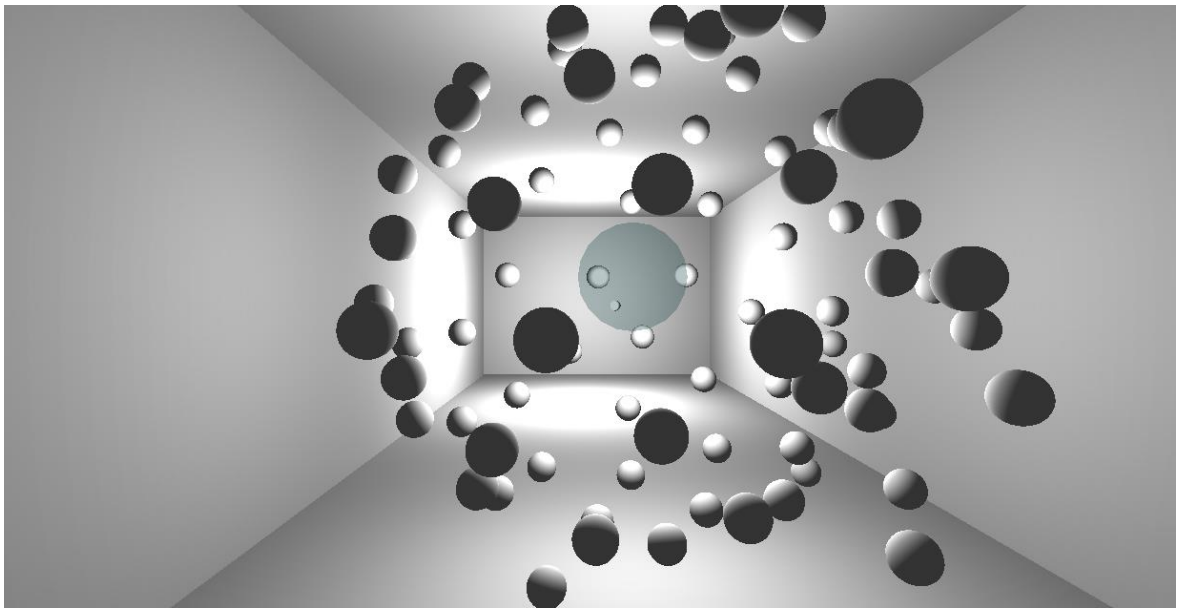
```
RaycastHit hit;
If (Physics.SphereCast(new Ray(transform.position +
transform.forward * 2.0f, transform.forward), 2.0f, out hit))
     oznacenObjekt = hit.collider.gameObject;
```

Nakon toga imamo sposobnost da se krećemo po sceni s objektom, te ga spustimo gdje želimo tako da raširimo ruku. Sve je to izvedeno unutar `Interakcija` klase.

```
Vector3 cilj = transform.position +
new Vector3(transform.forward.x, 0, transform.forward.z) * 5.0f;
Vector3 delta = cilj - oznacenObjekt.transform.position;
if (delta.magnitude > 0.1f)
{
     oznacenObjekt.rigidbody.velocity = (delta) * 10.0f;
} else {
     oznacenObjekt.rigidbody.velocity = Vector3.zero;
}
```

6.4. Demonstracija korištenja sile

U petoj sceni je napravljena ilustracija oponašanja privlačne sile pomoću jedne ruke. Skupljena šaka izaziva da transparentna kugla djeluje na predmete (kuglice) u prostoru tako da ih određenom brzinom privuče na površinu kugle. Kuglice se nastavljaju privlačiti ostaju tu dok ne otvorimo šaku – izazove se suprotna reakcija i dolazi do eksplozije koja razbaca kuglice po prostoru. Testna prostorija je ispunjena sa 100 kuglica, dok je kugla koju upravljamo jednom rukom ilustrirana kao transparentna sfera.



Slika 6.4 Četvrta scena

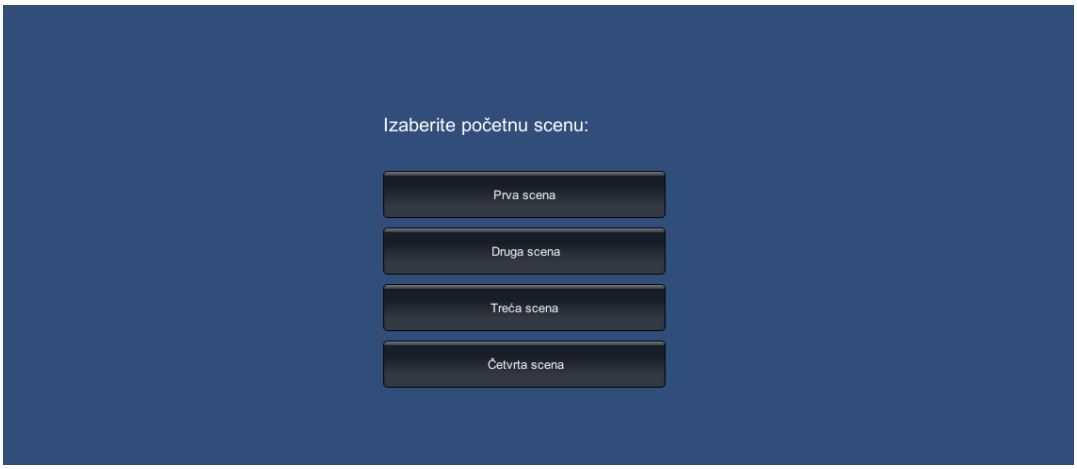
Nakon početne inicijalizacije okvira i kontrolera ispitujemo ruke koje se koriste. Iako je scena napravljena da se koristi s jednom rukom, i druga ruka će biti prepoznata, kako njeno nastupanje u scenu ne bi isključilo kuglu. Na isti način se može objekt prenositi, te slagati na različite druge objekte, međutim cilj je bio pokazati kako otpuštanje ruke može imati izravniji utjecaj na okolinu.

Vektor svake pojedinačne kuglice prema kugli pohranjujemo u varijablu `kuglicaKugla i` kada postavimo stanje ruke počinjemo privlačiti kuglice određenom brzinom.

```
for(int i = 0; i < kuglice.Length; ++i)
{
      Vector3 kuglicaKugla = transform.position –
                              kuglice[i].transform.position;
      kuglice[i].rigidbody.velocity = rigidbody.velocity;
      kuglice[i].rigidbody.velocity += -kuglicaKugla * 9.0f;
}
otvorenaRuka = true;
```

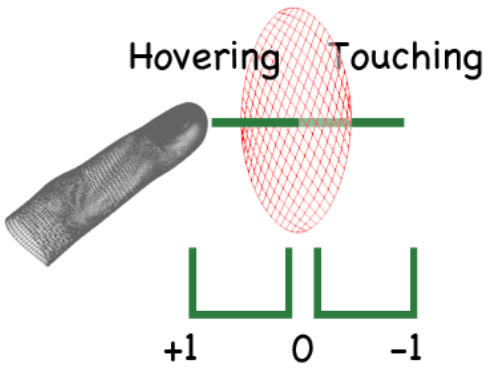
6.5. Izbornik – nulta scena

Kao početni ekran aplikacije napravljen je glavni izbornik na kojem biramo početnu scenu. Izbornik funkcionira pomoću dodira i nudi vizualizaciju prstiju na ekranu. Ponuđene su četiri scene za izbor koje možemo pozvati dodirom odgovarajućeg gumba. Ovo je primjer *offline* geste gdje emuliramo dodir pomoću Leap Motion uređaja. Sve unutar određene dubine vidnog polja se detektira kao dodir.



Slika 6.5Nulta scena

Možemo definirati adaptivnu površinu za dodir kako bi djelovali s 2D elementima u aplikaciji. Ta površina se nalazi paralelno s X-Y ravninom, ali se prilagođava korisnikovim prstima. Kad korisnik pokazivačkim objektom ili prstom se pomakne ispred ravnotežnog položaja to se registrira kao dodir.



Slika 6.6 Emulacija dodira

Iscrtavanje izbornika se radi pomoću `DodirVizualizacija` skripte koja je potpomognuta `LeapGUI` skriptom u kojoj je obrađena logika. Za svaki detektirani pokazivački predmet računamo veličinu oznake prsta, te potrebne parametre za izazivanje elipsoidnog efekta pri kretanju.

```
// Velicina.

float dodirVal = Mathf.Clamp(okvir.Pointables[i].TouchDistance,
-0.05f, okvir.Pointables[i].TouchDistance);

float velicina = 100.0f *
Mathf.Sqrt(Mathf.Min(Mathf.Max(Mathf.Abs(dodirVal), 0.002f),
0.85f));

Vector2 normalaXY = new
Vector2(okvir.Pointables[i].TipVelocity.x,
okvir.Pointables[i].TipVelocity.y);


// Elipsoid.

float yScale = 1.0f + Mathf.Min(0.65f, normalaXY.magnitude /
2000.0f);

float xScale = Mathf.Sqrt(1.0f / yScale);

float kut = Mathf.Rad2Deg * Mathf.Atan(normalaXY.x /
normalaXY.y);

GUIUtility.RotateAroundPivot(kut, new Vector2(x, y));
```

Nakon toga nam jedino ostaje iscrtati oznaku prsta u slučaju dodira.

7. Problemi i moguća poboljšanja

Upravljanje modelom reprezentiranjem grafičkog modela ruke još uvijek predstavlja izazov za tehnologiju poput Leap Motion uređaja, međutim uklanjanjem tog elementa prilično olakšavamo na improviziran način djelovati s određenim modelima.

Detekcija gesti koje smo naveli ima prilično izravnu implementaciju i uređaj uglavnom dobro nosi, ali su primijećene nestabilnosti, jer uređaj ponekad zna pojedine prste stapati jedan, ili pak pogrešno razlikovati pojedine prste. Navedeni problemi se daju zaobići implementacijom gesti koje se ne oslanjaju na te mogućnosti. Novije verzije uređaja Leap Motion bi trebale ponuditi bolje praćenje ruke uvođenjem implementiranog kostura.

Leap Motion se također pokazao kao dobro oruđe za detektiranje uzoraka gesti, ali je to tema koja teži više u raspoznavanje uzoraka pa se nije ulazilo dublje. Budući rad bi mogao dublje istražiti primjenu Ramer-Douglas-Peucker algoritma [9] pomoću uređaja za strojno prepoznavanje iscrtanih oblika.

U praktičnom dijelu rada se pokazala nespretnom interakcija dvije ruke na sceni. Ukoliko uređaj izgubi vezu s glavnom rukom pokušava sporednu detektirati kao glavnu. Uporaba sporedne ruke ili pokazivača bi se koristila za rotaciju objekta u drugoj i trećoj sceni, ali to se ostavlja za sljedeću verziju programske podrške.

Korišteni prototip Oculus Rifta nije stvarao prevelike probleme, iako je imao fizički kvar na kablu. Integracija s Unity okruženjem je prilično jednostavna, te će komercijalna verzija samo nadograditi njegove mogućnosti.

Zaključak

U oblikovanju virtualnih svjetova težimo ka što većoj uronjenosti korisnika, tako da nas je to dovelo do razvijanja interaktivnih 3D sučelja koji će na prirodan način interpretirati reakcije korisnika. Razvijeni su brojni uređaji čija je uporaba proširena na mnoga područja ljudske djelatnosti s ciljem olakšanja života [8].

Sposobnosti praćenja uređaja Leap Motion su se pokazala prilično zadovoljavajuća, iako treba imati na umu na koji način uređaj funkcionira. S obzirom na to, upotrebljivost je velik faktor koji se treba definirati za pojedinu aplikaciju. Unutar virtualnih okruženja detekcija jednostavnih gesti se pokazala korisnom i pružila jako intuitivno sučelje za rad.

Oculus Rift kao uređaj za vizualizaciju predstavlja nešto neviđeno na tržištu i samo je otvorio put za komercijalne HMD-e. Razvojna verzija uz malu rezoluciju prikaza je ponudila uranjanje za korisnika kakvo ni najbolji ekran nema. Veća rezolucija i dodatni senzori samo će poboljšati njegovo praćenje i time ostvariti bolje mogućnosti povezivanja s virtualnim 3D okruženjima.

Ovi uređaji se uglavnom koriste u industriji igara, gdje težimo oživljavanju virtualnih svjetova i činjenici da *holodeck* jednog dana pretvorimo u stvarnost. Postoji dosta prostora za poboljšanje praktičnog dijela u vidu implementacije algoritama za prepoznavanje uzoraka, kao i optimizacija nespretnih kontrola, ali bi to trebalo biti istraženo sa stanovišta UX-a. Radom je svakako pokazano da interakcija različitih interaktivnih 3D sučelja pruža veću uronjenost i bolje snalaženje u prostoru.

Literatura

[1] SPIEGELMOCK, M. *Leap Motion Development Essentials*. Pact Publishing, 2013.

[2] Leap Motion Developer, Web Documentation, https://developer.leapmotion.com/documentation

[3] Oculus Rift Development Kit, Version 1.1. Oculus VR, Inc, 2013.

[4] ANTONOV, M., MITCHELL, N., REISSE, A., COOPER, L., LAVALLE, S., KATSEV, M. SDK Overview. Oculus VR, Inc, 09.10.2013.

[5] TIDWELL, J. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly, 2010.

[6] BLACKMAN, S. *Beginning 3D Game Development with Unity: All-in-One, Multi-Platform Game Development.* Apress, 2013.

[7] GIOKARIS, P., Oculus Unity Integration Guide, Oculus VR, Inc, 27.11.2013.

[8] BOWMAN, D. A., KRUIJFF, E., LAVIOLA JR., J. J., POUPYREV, I. *3D User Interfaces: Theory and Practice*. Addison-Wesley, 2005.

[9] Wikipedia, Ramer–Douglas–Peucker algorithm, http://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm

[10] Wikipedia, Singleton pattern, http://en.wikipedia.org/wiki/Singleton_pattern

Sažetak

Naslov: Upravljanje modelima gestama ruku

Interaktivna 3D korisnička sučelja danas postaju sve većim dijelom tehnološke svakodnevnice. U ovom diplomskom radu su proučene mogućnosti uređaja za uzorkovanje pokreta Leap Motion, te njegova mogućnost prepoznavanja gesti. Kroz nekoliko scena praktički su razrađene tehnike interakcije 3D modelom objekta. Ostvareni 3D kontekst je uspješno prikazan korisniku uređajem za prikaz postavljenim ispred očiju Oculus Rift. Pri tome je napravljena kvalitativna ocjena navedenih uređaja, kao i analizirana njihova uporaba u današnjem društvu. Programska podrška je napravljena pomoću programskog jezika C# i razvojnog okruženja Unity.

Ključne riječi: Leap Motion, Oculus Rift, geste, ruke, raspoznavanje uzoraka, 3D, C#, Unity, virtualna stvarnost, upotrebljivost

Summary

Title: Controlling models with hand gestures

Interactive 3D user interfaces are becoming an increasing part of everyday technology. This thesis explores the possibility of Leap Motion, a movement tracking device, and his ability to recognize gestures. Through several scenes we explored interaction techniques with 3D model objects. Realized 3D context is successfully displayed to a user display device placed in front of the eyes, Oculus Rift. In doing so, we made a qualitative evaluation of the specified devices, and analyzed their use in today's society. The practical software is built using the C# programming language and the Unity development environment.

Keywords: Leap Motion, Oculus Rift, gestures, hands, pattern recognition, 3D, C#, Unity, virtual reality, usability

Skraćenice

HMD	*Head-Mounted Display*	Zaslon za postavljanje na glavu
FOV	*Field Of View*	Vidno polje
SDK	*Software Development Kit*	Alati za razvoj programske potpore
API	*Application Programming Interface*	Aplikacijsko programsko sučelje
TCP	*Transmission Control Protocol*	Protokol transportne razine
LCD	*Liquid-Crystal Display*	Zaslon s tekućim kristalima
UX	*User eXperience*	Iskustvo korisnika

Privitak

Programska potpora s detaljnim uputama za korištenje nalazi se na elektroničkom mediju priloženom uz ovaj rad. .

.

.