Prepoznavanje gesta praćenjem ruku u proširenoj stvarnosti

Fumić, Matija

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:079826

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

PREPOZNAVANJE GESTA PRAĆENJEM RUKU U PROŠIRENOJ STVARNOSTI

Diplomski rad

Matija Fumić

Osijek, 2022.

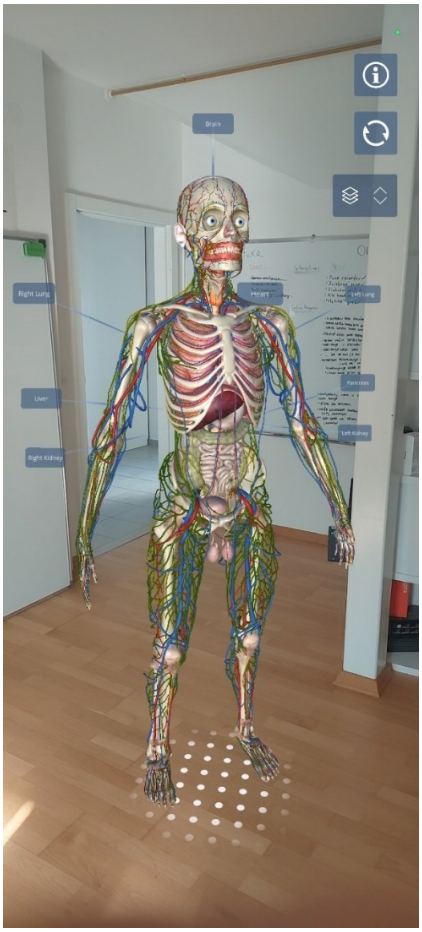
SADRŽAJ

1. UVOD 1
2. PREGLED PODRUČJA PREPOZNAVANJA GESTI U TEHNOLOGIJAMA PROŠIRENE STVARNOSTI 5
3. KORIŠTENE TEHNOLOGIJE 7
3.1. UNITY GAME ENGINE 7
3.2. VISUAL STUDIO 2022 8
3.3. OCULUS QUEST 2 8
3.4. OCULUS SDK 9
3.5. OCTOXR 10
4. PRAĆENJE RUKU – OCULUS QUEST 2 11
5. IMPLEMENTACIJA ALGORITMA 12
6. PRIKAZ I PRIMJENA IMPLEMENTACIJE 20
6.1. KAMEN, ŠKARE, PAPIR 20
6.2. GESTURE HERO 23
7. ZAKLJUČAK 26
LITERATURA 28
SAŽETAK 29
ABSTRACT 30
ŽIVOTOPIS 31

1. UVOD

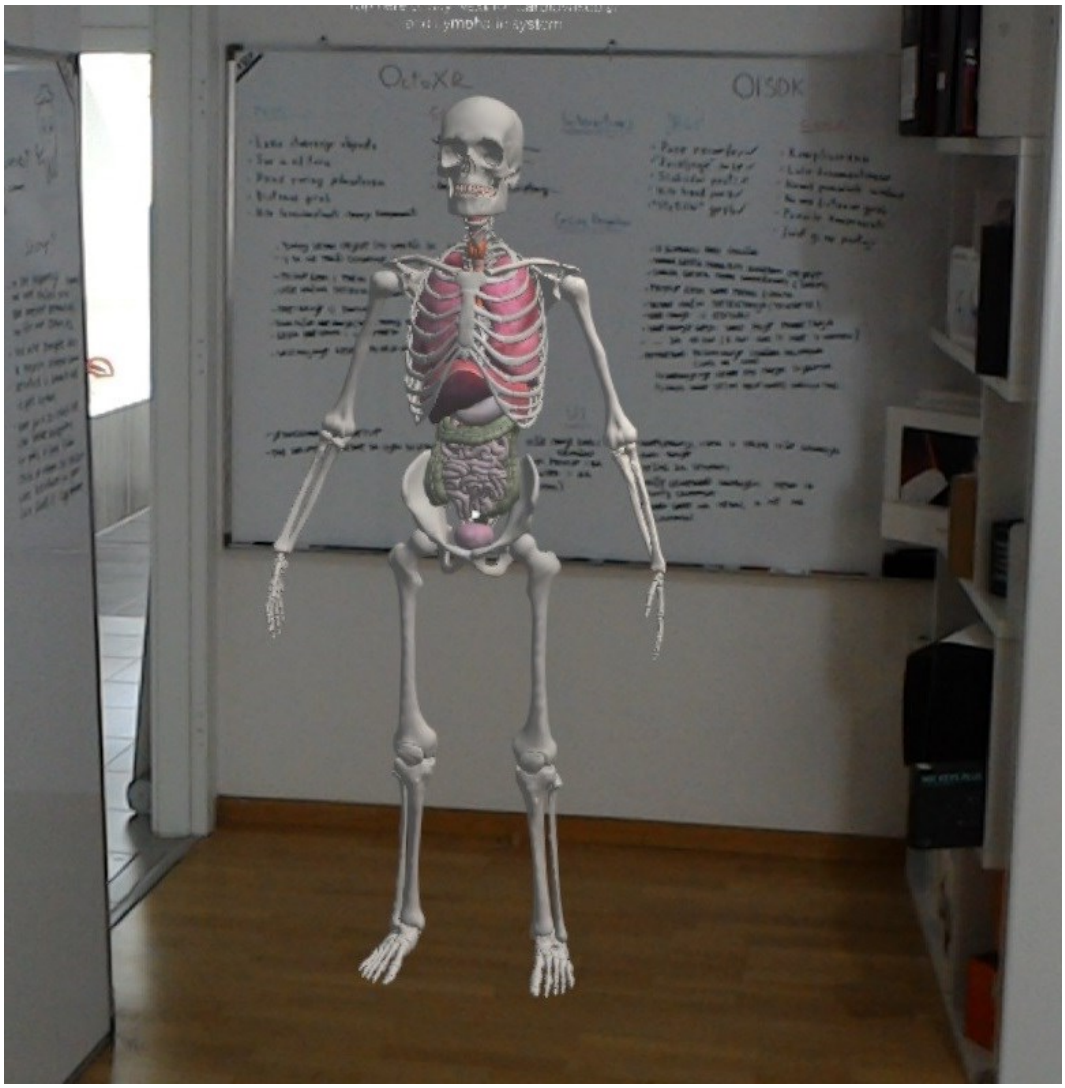
Akronim *XR* otvara vrata širokom spektru različitih tehnologija i mogućnosti. Proširena stvarnost (engl. *Extended Reality*) označava skup od tri različite, ali slične tehnologije.

Prvu od njih, a ujedno i najzastupljeniju, označava proširena stvarnost, akronim *AR* (engl. *Augmented Reality*), predstavlja uvećavanje stvarnosti za još jedan sloj. To znači da se preko sloja stvarnosti koju korisnik vidi kamerom, dodaje još jedan digitalni sloj medije. Najčešće su to neki modeli, slike, videozapisi i ostali multimedijski sadržaji. Takav se tip sadržaja najviše predstavlja kroz mobilne i web aplikacije. Iako postoje *AR* naočale, još uvijek nisu toliko zastupljene, što zbog cijene, a što zbog činjenice da skoro svaki čovjek već ima svoj *AR* uređaju u obliku svog mobitela. Najčešće se koristi u edukativne svrhe jer olakšava učenicima vizualizirati neke nedostupne stvari. Primjer *AR* tehnologije nalazi se na slici 1.1.



Slika 1.1. Prikaz *AR* tehnologije.

Sljedeća tehnologija poznata je pod nazivom pomiješana stvarnost, akronima *MR* (engl. *Mixed Reality*), predstavlja najmanje zastupljeni tip proširene stvarnosti u kojem se gore spomenuta dva sloja (stvarnosti i digitalni) spajaju u jedan. Ove dvije tehnologije ljudi najčešće miješaju te ih je teško razlikovati. Ključna je razlika u tome što je *AR* sloj digitalnog preko stvarnog te takvi digitalni objekti nemaju mogućnost interakcije sa stvarnim svijetom. U *MR*-u su ta dva sloja pomiješana u jedan što otvara mogućnost interakcije digitalnih objekata sa stvarnim svijetom. Ova tehnologija nije toliko zastupljena zato što je nedostupna i jako skupa. Trenutno jedini istinski *MR* uređaji su *Microsoft Hololens* i *Microsoft Hololens 2*. Najlakša je razumjeti ovu tehnologiju ljudima kada se povuče paralela s hologramima. Prikaz holograma u pomiješanoj stvarnosti na slici 1.2.



Slika 1.2. Snimak zaslona iz *Microsoft Hololensa*.

Posljednja od tri tehnologije koja pripada ovoj skupini, a ujedno i ona koja je odabrana za prezentaciju ovoga rada, je virtualna stvarnost, predstavlja se pod akronimom *VR* (engl. *Virtual Reality*) i označava tehnologiju potpunog uranjanja (engl. *Immersion*) u digitalni svijet. Za razliku od dvije gore spomenute tehnologije, virtualna nema sloj stvarnoga svijeta već se sve odvija u potpuno digitalnom svijetu. Korisnik u potpunosti uranja u virtualni svijet i ima potpunu moć upravljanja virtualnim predmetima. Primjer tehnologije virtualne stvarnosti nalazi se na slici 1.3.

Slika 1.3. Prikaz *VR* tehnologije iz *Oculus* početnog izbornika

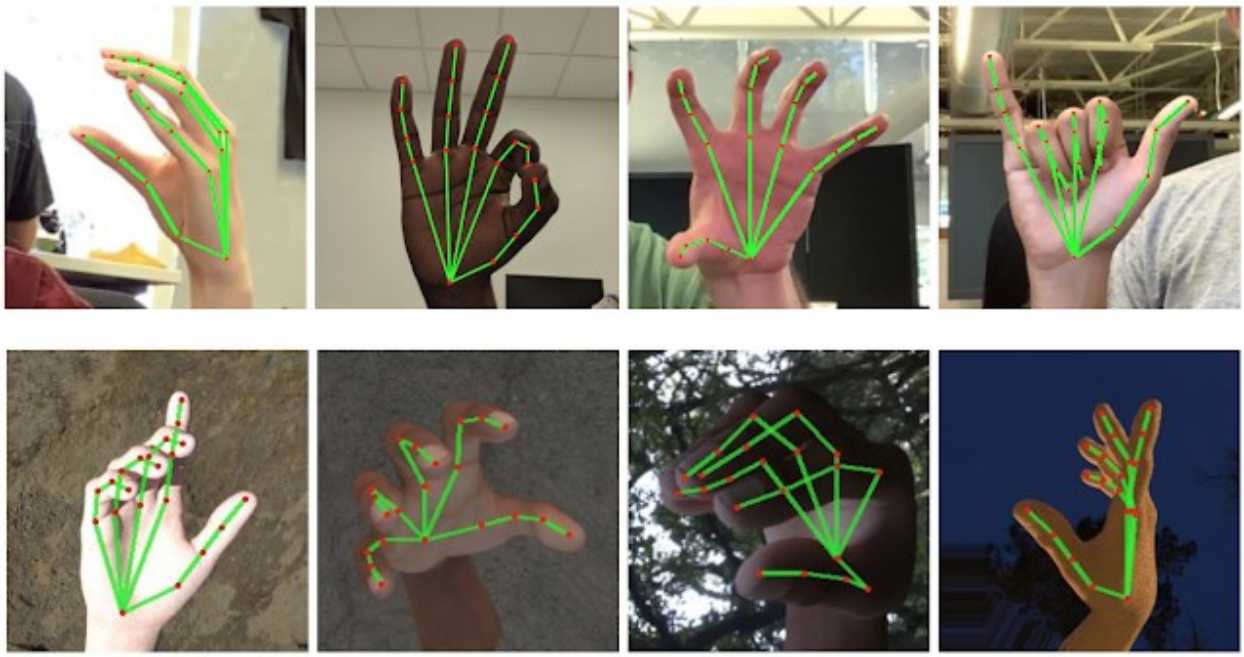
Najveću primjenu ova tehnologija pronalazi u industriji video igara, ali je također zastupljena i u turizmu, obrazovanju i virtualnim treninzima. Svoje začetke pronalazi ponajviše u takozvanim virtualnim šetnjama gdje korisnici mogu doživjeti neko područje u drugačijem ruhu, kako je ono izgledalo prije dvjestotinjak godina ili posjetit željeno područje kao što su Piramide u Gizi. S

napretkom tehnologije i samih uređaja za virtualnu stvarnost, rasle su i mogućnosti takvih aplikacija. Kako bi se lakše razumjela vremenska crta ove tehnologije, u ovom primjeru će se iskoristit android uređaji marke *Oculus*. Prvi je takav bio *Oculus Go*. Uređaj koji ima karakteristike tri stupnja slobode (rotacija oko x, y i z osi). Dolazi s jednim kontrolorom i nije dovoljno jak kako bi mogao prikazivati kompleksnije aplikacije. Korišten je upravo za gore spomenute virtualne šetnje i prikazivanje videozapisa od 360 stupnjeva. Nakon njega prekretnicu u *VR* svijetu donio je *Oculus Quest*, odnosno njegov nasljednik *Oculus Quest 2*, trenutno uvjerljivo najzastupljeniji android *VR* uređaj. Dolazi sa šest stupnjeva slobode. Uz gore spomenute tri, *Quest* dolazi s dodatne tri prostorne koordinate što znači da se korisnik može kretati u stvarnom svijetu. Ovaj je uređaj dovoljno jak da pokreće i zahtjevne sadržaje zbog kojih dolazi i velika primjena u igraćoj industriji. Još jedna velika primjena koju su omogućili ovi uređaji je treniranje zaposlenika za situacije visokog rizika ili opasnosti po ljudski život. Teško je zaposlenike odmah ubaciti na upravljanje zahtjevnim strojevima koje ih mogu koštati ozlijede ili života pa ih se zato prvo pripremi kroz aplikaciju koja vjerodostojno prikazuje što ih sve čeka kada krenu upravljati spomenutim strojem. Naravno, treba napomenuti kako postoji i uređaji koji služe samo za prikaz slike dok se sva logika obrađuje na računalu. Takvi uređaji nemaju vlastiti operacijski sustav i moraju biti povezani kablom s računalom.

Uređaji sami po sebi uranjaju korisnika u digitalni svijet, ali se još uvijek pokušava taj osjećaj podići na višu razinu. Tu u priču ulazi praćenje ruku. Do nedavno jedini način upravljanja sadržajem u virtualnom svijetu bio je korištenjem kontrolera. Ugrađivanje infracrvenih kamera na uređaje koji prepoznaju ljudske ruke otvorio je novi svijet mogućnosti. Interakcije rukama znatno su intuitivnije i prirodnije. Korisnici se lakše snalaze u aplikacijama, ne moraju prolaziti treninge za upravljanje aplikacijom i slično. U svakodnevnoj su komunikaciji ljudi navikli nesvjesno (ili svjesno) gestikulirati rukama. Pa tako geste, kao što su one otvorenog dlana, šake, palca dolje ili palca gore, pronalaze svoje mjesto u svakodnevnim životima. Zadatak ovoga rada je implementirati sustav prepoznavanja gesta u takvim aplikacijama te dopustiti korisnicima da na lak način povežu svoje funkcionalnosti s određenom gestom i učine svoje aplikacije intuitivnije (na primjer: istakni dijelove u koje korisnik upire prstom i slično).

2. PREGLED PODRUČJA PREPOZNAVANJA GESTI U TEHNOLOGIJAMA PROŠIRENE STVARNOSTI

Prepoznavanje ruku pomoću računalnog vida nešto je što postoji već nekoliko godina. Dostupan je velik broj biblioteka koje obavljaju posao, a najkorištenija trenutno je ipak Googleov *MediaPipe*. To je biblioteka otvorenog koda za strojno učenje za *live* i *streaming* medije (slike koje dolaze u stvarnom vremenu i već gotove slike). Najčešće se koristi uz *OpenCV* okruženje i *Tensorflow* biblioteku za strojno učenje. *MediaPipe* dolazi s već naučenim algoritmom prepoznavanja ruku. *OpenCV* se koristi za prepoznavanje slike u stvarnom vremenu dok se *Tensorflow* koristi kako bi se prepoznale geste ruku (palac gore, palac dolje…). Prikaz ruke koristeći *MediaPipe* biblioteku nalazi se na slici 2.1.



Slika 2.1. Ruke i kosti ruke koristeći *MediaPipe* biblioteku [2]

Handtrack.js je javascript biblioteka za prepoznavanje ruku u stvarnom vremenu u web pregledniku. Sadrži već naučen model korištenjem *Tensorflowa* i konvolucijske neuronske mreže. Izabrani skup podataka je *Egohands* sa sveučilišta u Indijani. Ova biblioteka koristi za prepoznavanje osnovnih gesta kao što su otvoreni dlan, zatvoreni dlan i štipanje [3].

ManoMotion MobileAR je komplet alata za razvoj softvera (*SDK, Software Development Kit*) za razvoj *Unity AR* aplikacije korištenjem *ARFoundation* nadogradnje. Nudi prepoznavanje ruku i

osnovnih gesta pomoću kamere na android i iOS uređajima. Radi se o novom projektu o kojemu još nema puno informacija, ali je vrijedno spomena kao potencijalna tehnologija u ovom području [4].

Gore se navedene biblioteke koriste u različitim tehnologijama, ali niti jedna od njih nije primjenjiva u tehnologiji virtualne stvarnosti. *Oculus* (sada *Meta*) predstavio je korištenje ruku za upravljanje uređajem tek u prosincu 2019. godine. U to je vrijeme praćenje ruku bilo dostupno samo u njihovom izborniku, a tek krajem 2020. godine su objavljene i prve aplikacije koje podržavaju praćenje ruku. Danas je pristup praćenja ruku dostupan svima u Oculusovom *SDK*-u. Prvi je koncept prepoznavanja gesta u ovoj tehnologije prikazao profesor Quentin Valembois sa sveučilišta u Liègeu, Belgija u svom YouTube-u videu. Osnovna ideja za ovaj rad preuzeta je od njega te još nadograđena u cijeli sustav prepoznavanja gesta. Dvije godine nakon objavljivanja Quentinovog rada, a za vrijeme razvijanja ovog rada, *Oculus* je u svoj *SDK* dodao sekciju pod nazivom *Pose Detection*. Korištenjem njihovog *SDK*-a korisnik može definirati status svakog prsta (ispružen, djelomično savijen, u potpunosti savijen) te takovom kombinacijom kreirati gestu pa bi tako palac gore imao palac ispružen te sve ostale prste u potpunosti savijene [6].

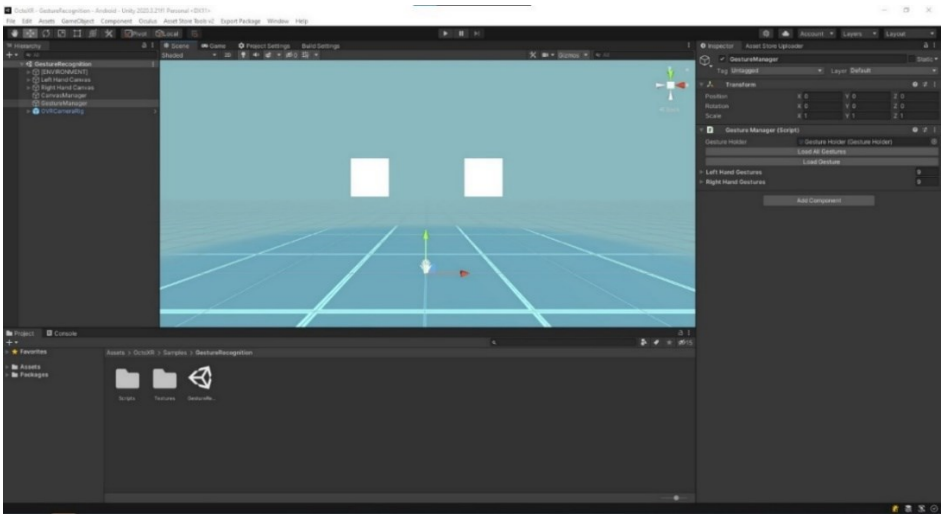
3. KORIŠTENE TEHNOLOGIJE

U ovom će poglavlju biti opisane tehnologije i alati korišteni pri izradi ovog diplomskog rada.

3.1. UNITY GAME ENGINE

Kako je teško smisleno prevesti stručnu terminologiju, u ovom će poglavlju prvo biti navedena engleska definicija te objašnjenje riječ po riječ. *Unity is a cross-platform realtime 3D/2D game engine*. *Cross-platform* označava da se aplikacije napravljene u Unityju mogu izvesti na različite platforme (windows, android, iOS, PlayStation, server, web…). Ovo je moguće zato što je glavni skripti jezik u Unityju C#, a konkretna implementacija za *Unity* bazirana je na Monu i IL2CPP-u.

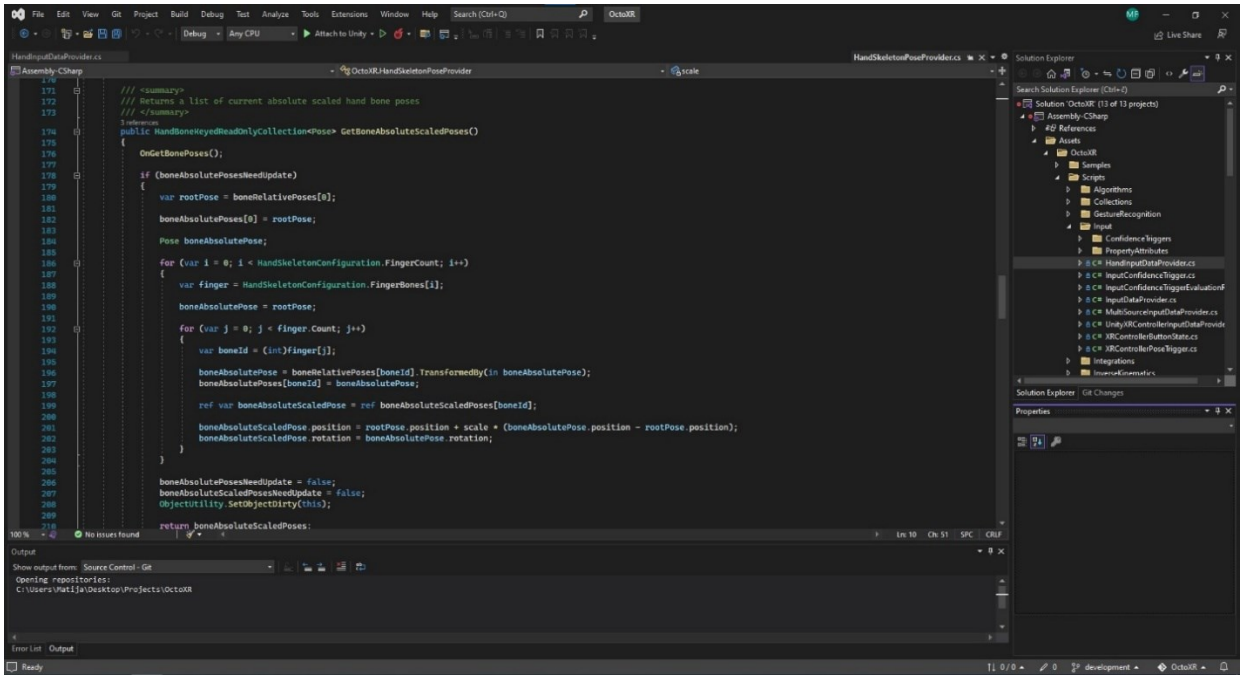
Realtime označava da se aplikacija prikazuje u stvarnom vremenu. Filmovi (općenito videozapisi) su napravljeni i generirani te njihov prikaz nikada nije u stvarnom vremenu. *Unity* aplikacije se prikazuju u stvarnom vremenu što otvara mogućnost različitih interakcija te reagiranja na iste. *Unity* nudi mogućnost izrađivanja aplikacija u tri te dvije dimenzije. Diplomski će rad biti implementiran na LTS verziji Unityja 2020. Korišten je IL2CPP *backend* i .NET standard 2.0 i *Roslyn* prevoditelj. Projekt je prilagođen za izvoz aplikacije na android jer je *Oculus Quest* baziran na androidu. Unityjevo sučelje olakšava developerima razvoj na intuitivan način, a sam pogonski sklop igre (engl. *game engine*) je koncipiran tako da se developer može koncentrirati više na samu mehaniku igre nego implementacije sustava fizike, kolizije i slično. Prikaz grafičkog sučelja Unityja nalazi se na slici 3.1



Slika 3.1. Grafičko sučelje Unityja

3.2. VISUAL STUDIO 2022

Microsoft Visual studio razvojno je okruženje za različite vrste desktop, web, mobilnih i igraćih aplikacija. U ovom konkretnom slučaju korišten je paralelno s Unityjem za razvoj aplikacije koja će biti testirana na android uređaju. Kako bi navedeno radilo, potrebno je instalirati modul za razvoj računalnih igara i *Visual studio* alate za *Unity*. Prikaz *Visual studio* razvojnog okruženja na slici 3.2.



Slika 3.2. Prikaz *Visual studio* okruženja sa C# izvornim kodom

3.3. OCULUS QUEST 2

Oculus Quest 2 treći je u nizu *Oculusovih* android uređaja, drugi u seriji *Quest*. Pušten je na tržište krajem 2020. godine. Najveći napredak u odnosu na prethodnika je revolucionarni Qualcommov procesor *Snapdragon XR2* napravljen isključivo za uređaje virtualne stvarnosti. Po procesorskoj moći je najsličniji Samsungovom *Galaxy* S20 uređaju, ali s jednom velikom razlikom. *Quest 2* mora prikazivati sliku na dva 1832x1920 zaslona po dva oka što zahtjeva prilagođeniji rad grafičkog procesora. Dolazi sa dva kontrolera koja su s uređajem povezana bluetoothom. Kako je već spomenuto, kontrolere se može zamijeniti i konkretnim interakcijama rukama koje uređaj prepoznaje koristeći četiri infracrvene kamere koje se nalaze na prednjoj strani

uređaja. Ovaj je uređaj pristupačne cijene i zastupljen je značajno na tržištu *VR* uređaja, a njegove napredne mogućnosti interakcije rukama upravo ga čine idealnim kandidatom za demonstraciju ovog diplomskog rada.

Slika 3.3. *Oculus Quest 2*

3.4. OCULUS SDK

Kako bi olakšao developerima razvoj aplikacija za svoje uređaje, *Oculus* je napravio vlastiti SDK. Budući da je moguće aplikacije praviti na više platforma, pa tako postoje i četiri vrste SDK-a za četiri platforme. Konkretno, radi se o SDK-u za web platforme, izvoran android razvoj, *Unreal game engine* te *Unity game engine*. Ovaj je diplomski rad napravljen u Unityju pa je tako korišten i *Oculus integration* paket preuzet iz *Unity* trgovine. U vrijeme pisanja ovoga rada aktualna je verzija 42 te je ista korištena u izradi.

3.5. OCTOXR

OctoXR je alat dostupan za kupnju na *Unity* trgovini. Dolazi s gotovim alatima za izradu *VR* aplikacije. Ovaj je diplomski rad rađen u sklopu *OctoXR* i sav izvorni kod dostupan je na trgovini. Uz prepoznavanje gesta, ovaj alat sadrži i osnovne *VR* interakcije kao što su hvatanje stvari, kretanje, animacije avatara, interakcija s korisničkim sučeljem i slično. Njegov je cilj olakšati korisnicima izradu takvih aplikacija. Maknuti koncentraciju developera s implementiranja osnovnih funkcionalnosti na kvalitetu same aplikacije i mehaniku.

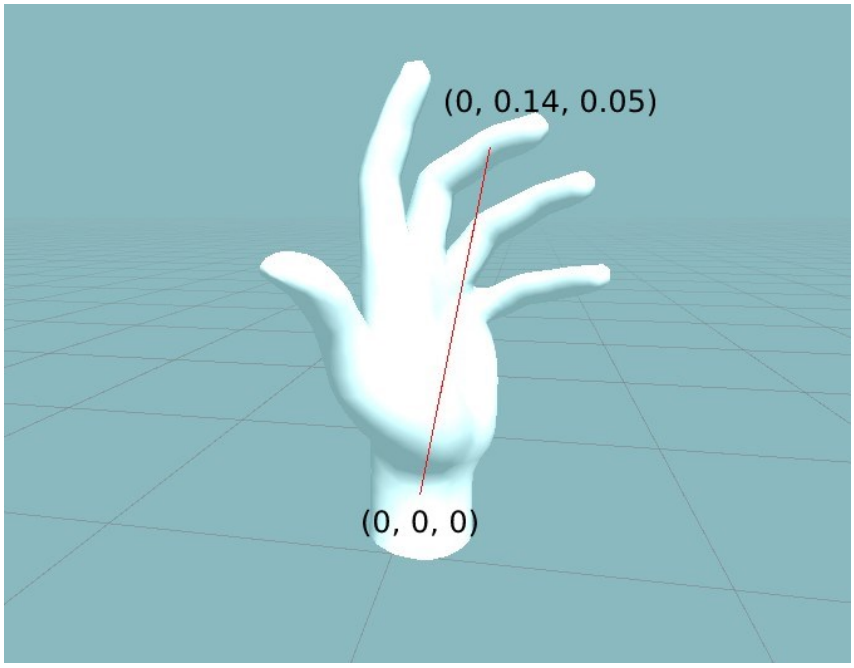
4. PRAĆENJE RUKU – OCULUS QUEST 2

Kako je već gore spomenuto, *Oculus Quest* ima četiri infracrvene dubinske kamere kojima prati stanje svoje okoline. Pomoću te četiri kamere i inercijalnih mjernih jedinica koristeći VISLAM (*Visual – Inertial Simultaneous Localization and Mapping*) tehniku, uređaj određuje svoj točan položaj u prostoru i njegov položaj u odnosu na druge objekte. Uređaj tako u stvarnom vremenu frekvencijom od 30 slika po sekundi kreira oblake točaka svoje okoline. Pomoću tih oblaka točaka i algoritama strojnog učenja uspješno se prepoznaju korisnikove ruke. *Oculus* kreira kostur ruke kao što je prikazan na slici 2.1. Koristeći njihov *API* u Unityju može se pristupiti lokalnim rotacijama svake kosti (napomena kako se radi o ugniježdenoj strukturi kostiju kod svakog prsta te je distalna falanga u hijerarhiji dijete medijalne falange koja je opet u hijerarhiji dijete proksimalne falange i tako dalje) te ih zapisati u svoj kostur i prilagođeni *mesh* ruke (3D zapis koji se sastoji od poligona sadržanih u x, y i z referentnim točkama koje opisuju visinu, širinu i dubinu modela). Za potrebe je ovog rada napisan adapter za ulazne podatke koji daje standardiziran izlaz podataka. To znači da se izvorni kod ovog rada može koristiti i na drugim uređajima kao što su *HTC Vive Focus* ili *Leap Motion*. Potrebno je samo adapteru dati točan set ulaznih podatka.

5. IMPLEMENTACIJA ALGORITMA

Sustav prepoznavanja gesti sastoji se od nekoliko ključnih komponenti kao što su kreiranje geste, spremanje geste, uvoženje konkretnih gesta u scenu te prepoznavanje same geste. U ovom će poglavlju biti opisana implementacija i korištenje algoritma korak po korak.

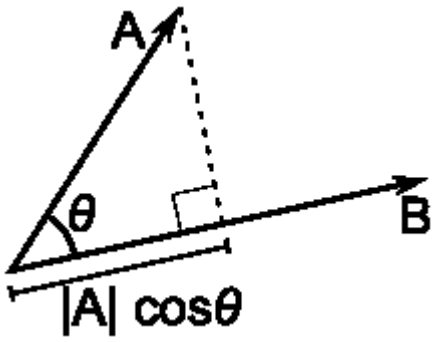
Kreiranje geste radi se kroz skriptu *GestureCreator.cs*. Istu je moguće kreirat klikom na gumb *Create Gesture* dok korisnik rukom ispred kamere pokazuje željenu gestu. Nakon klika na gumb, na ekranu se pojavi prozor sa spremanje geste te korisnik sprema gestu negdje u hijerarhiju samog projekta. Format spremanja geste bit će naknadno objašnjen. Osim što je moguće kreirati gestu u editoru, ista se može kreirati i tijekom trajanja aplikacije, ali se onda drugačije mora spremati. Oba načina pozivaju metodu *CreateGesture*. Važno je da se komponenta *GestureCreator.cs* nalazi na istom objektu kao korijenski element kostura ruke kako bi ista imala pristup cijelom kosturu. Korijenski element sadrži listu svih kostiju ruke. Kada se pozove metoda, ona prolazi kroz listu svih kostiju i računa relativnu poziciju svake kosti u odnosu na korijensku kost i sprema ju u novu listu. Ovo se može predočiti kao da se korijenska kost stavi u koordinatu (0, 0, 0) te se računa na kojoj se lokalnoj poziciji nalazi svaka kost u odnosu na ishodište novog koordinatnog sustava. Na slici 5.1 prikazana je relativna pozicija distalne falange srednjeg prsta lijeve ruke i korijene kosti (ručni zglob).



Slika 5.1. Prikaz relativne pozicije distalne falange srednjeg prsta.

Unity game engine koristi lijevi koordinatni sustav s y osi uperenom prema gore. Svaki objekt u *Unityju* ima također svoj lokalni koordinatni sustav. Ako se taj objekt nalazi na rotaciji 0 oko x osi, 0 oko y osi i 0 oko z osi, onda se njegov lokalni koordinatni sustav poklapa s globalnim. Kada se izračunaju lokalne pozicije kostiju u odnosu na korijensku kost, računa se skalarni produkt svake koordinatne osi lokalnog koordinatnog sustava korijenske kosti s jediničnim vektorom okomitog vektora prema gore (0, 1, 0). To se radi kako bi se izbjegla kriva detekcija geste. Kada bi se geste prepoznavale samo po lokalnim pozicijama kostiju u odnosu na korijensku kost, palac gore i palac dolje bi bili jedna te ista gesta. Zato se u obzir mora uzeti i orijentacija same ruke, odnosno korijenske kosti. Kao referentni vektor se uzima jedinični okomiti vektor prema gore jer je on u svakom trenutku jednak. Jedinični vektor prema desno može, a i ne mora biti jednak pri kreiranju geste i u nekom trenutku detektiranja, točnije njihov skalarni produkt ne mora biti jednak ili sličan. Skalarni produkt dva vektora može se predočiti kao mjera poklapanja (vrijednost od 0 do 1) dva vektora. Točnije, projekcija jednog vektora na drugi. Prikaz skalarnog produkta nalazi se na slici 5.2, a formula (5-1) prikazuje kako se računa skalarni produkt.

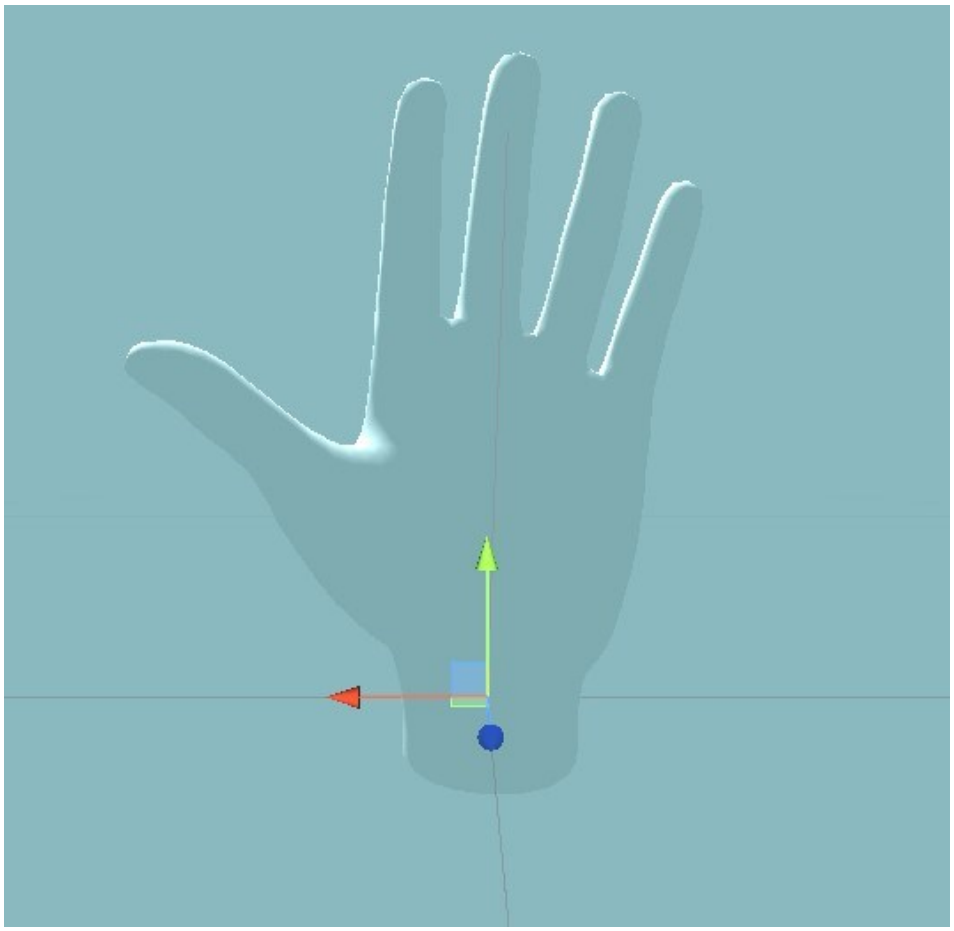
$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\gamma \qquad (5-1)$$



Slika 5.2. Skalarni produkt dva vektora

Neka x os lokalnog koordinatnog sustava korijenske kosti (ručnog zgloba) ide u smjeru palca (crvena os na slici 5.3). Pri pokazivanju geste palac gore skalarni produkt lokalne x osi i globalnog jediničnog vektora prema gore daju vrijednost približno 1 (imaju približno isti smjer i orijentaciju). Pri pokazivanju geste palac prema dolje skalarni produkt lokalne x osi i globalnog jediničnog

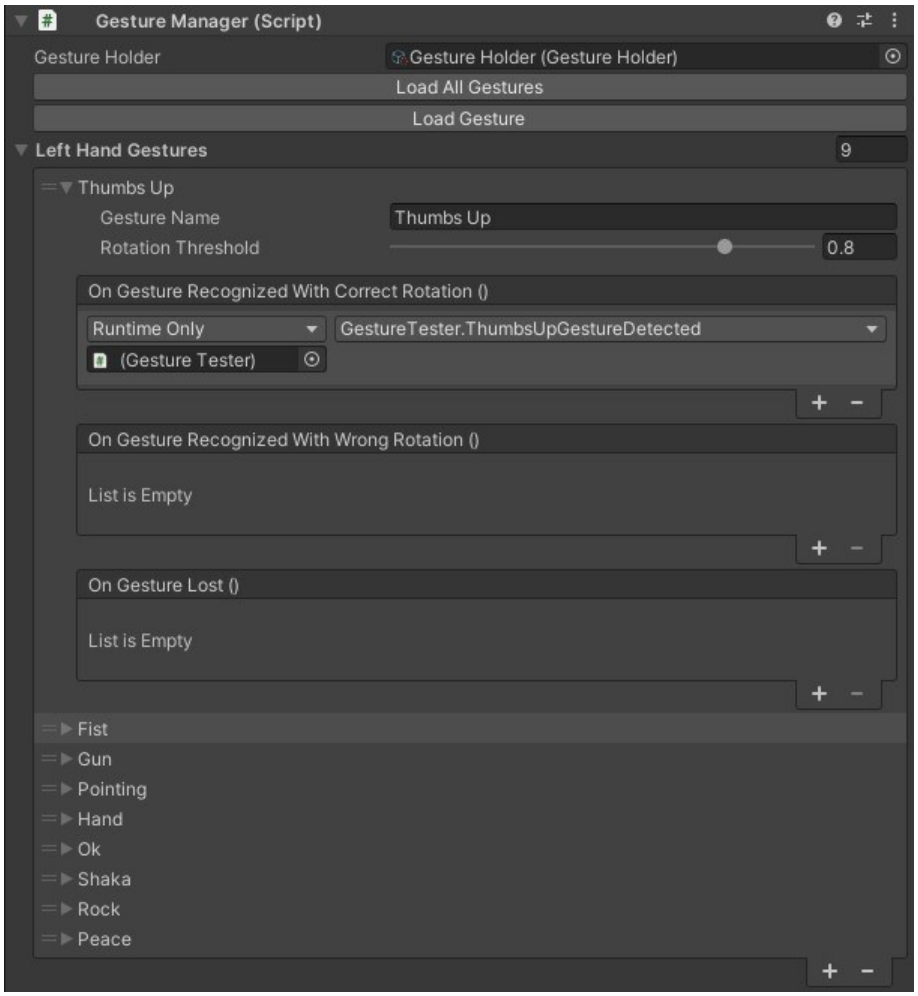
vektora prema gora daju vrijednost približno -1 (imaju približno isti smjer, ali suprotne orijentacije). Iz tog se razloga prati skalarni produkt sve tri lokalne koordinatne osi i iz ta se tri podatka može odrediti globalna orijentacija ruke u prostoru.



Slika 5.3. Lokalne koordinatne osi korijenske kosti ruke

Nakon što se izračunaju lokalne pozicije kostiju i skalarni produkti, kreira se skriptni objekt u koji se te vrijednosti zapisuju skupa s ostalim podacima kao što su naziv geste, ruka i slično. Takav se skriptni objekt može zatim spremiti te koristiti u drugim scenama/projektima/aplikacijama. To je moguće zato što *GestureManager.cs* komponenata može kroz izbornik uvesti više različitih gesta. Ta je komponenta zadužena za objedinjavanje svih gesta za jednu scenu i poveže ih s njihovim funkcionalnostima. Skriptni objekti prevode se u strukturu koja se može serijalizirati (proces transformacije struktura u format koji *Unity* može spremiti i rekonstruirati ponovo kasnije u izvođenju) i prikazati u editoru te se za svaku gestu vežu tri događaja: gesta prepoznata, gesta

prepoznata, ali s krivom rotacijom, geste izgubljena. Kako to izgleda u *Unity* izborniku prikazano je na slici 5.4.



Slika 5.4. Prikaz povezivanja funkcionalnosti sa specifičnim gestama

GestureDetector.cs komponenta je također usko vezana za korijenski element kostura ruke jer treba pristup svim kostima. Ona pri pokretanju dohvaća sve geste i događaje vezane za tu ruku te svaki *frame* aplikacije uspoređuje trenutno stanje ruke s kreiranim gestama. To može činiti na dva načina. Prvi način je kontinuirana detekcija. Svaki *frame* traži novu gestu te ako je pronađena, pozove određene funkcionalnosti koje su pretplaćene na taj događaj. To znači da ako korisnik drži palac gore, svaki *frame* će biti pozvane funkcionalnosti vezane za gestu palac gore. Drugi je način diskretna detekcija. U ovom obliku detekcije svaki *frame* se detektiraju geste, ali funkcionalnosti vezane za određenu gestu bit će pozvane samo jednom, onog trena kada je ona prvi puta prepoznata. To znači da kada korisnik pokaže palac gore, funkcionalnosti vezane uz tu gestu bit

će pozvane samo kada je ona prvi puta prepoznata i neće se pozivati više. Sljedeće funkcionalnosti pozvat će se tek kada je prepoznata neka druga gesta, različita od prethodne. Svaki *frame* prolazi se kroz listu kreiranih gesta. Za svaku se gestu prolazi opet kroz sve kosti. Računa se trenutna lokalna pozicija kosti u odnosu na korijensku kost. Ako je trenutna lokalna pozicija kosti približno jednaka onoj spremljenoj za tu kost pri kreiranju te geste, sprema se razlika njihovih pozicija (varijabla x) i prelazi se na sljedeću kost. Ako je razlika u pozicijama kosti prevelika, ova gesta se odbacuje i prelazi se na sljedeću. Kada završi provjeravanje svih kostiju za jednu gestu, to znači da su sve kosti na približno potrebnim pozicijama za određenu gestu. Zbrajaju se sve razlike pozicija njihovih kostiju (suma svih varijabla x za svaku kost) i spremaju (varijabla y). Kada se prođe kroz sve kreirane geste, prepoznata je ona gesta koja ima najmanju sumu razlika njihovih pozicija (najmanju varijablu y). Ako nije prepoznata niti jedna gesta, algoritam vraća praznu gestu. Ta se jedna gesta (prepoznata ili prazna) zatim koristi u daljnjem računanju. Prvo se provjerava je li vraćena gesta jednaka praznoj gesti. Ako je prepoznata prazna gesta, a prošli *frame* aplikacije je bila prepoznata neka konkretna gesta, to je znak da se na prošloj gesti pozove sva funkcionalnost povezana na događaj gesta izgubljena te se završi s iteracijom algoritma u ovom *frameu*. Ako je ipak prepoznata gesta različita od prazne, iteracija se nastavlja. Zatim se provjerava ako je odabran diskretan način rada, a trenutna gesta jednaka je gesti prepoznatoj prošli *frame*, prekida se iteracija za ovaj *frame*. Ako se radi o kontinuiranom načinu rada, ili pak diskretnom, ali je prepoznata gesta različita od prethodne, algoritam nastavlja. Računa se trenutni skalarni produkt lokalnih koordinatnih osi korijenske kosti s globalnim jediničnim vektorom prema gore. Ako su ta tri skalarna produkta približno jednaka onim spremljenim za tu gestu, smatra se da prepoznata gesta ima dobru orijentaciju. Ako skalarni produkti nisu približno jednaki spremljenima, onda se smatra da je gesta prepoznata, ali s krivom rotacijom. Sukladnoj toj odluci pozivaju se određene funkcionalnosti pretplaćene na jedan od ta dva događaja. Nije moguće prepoznati oba događaja. Slika 5.5. prikazuje linije koje se pozivaju svaki *frame* u prepoznavanju gesta.

```
Unity Message | 0 references
private void Update()
{
    if (!handSkeleton.IsComplete) return;

    if (handBones == null) handBones = new List<HandBone>(handSkeleton.Bones);

    Gesture currentGesture = DetectGesture();

    hasRecognized = !currentGesture.Equals(defaultGesture);

    if (!hasRecognized)
    {
        if (!previousGesture.Equals(defaultGesture))
        {
            previousGesture.OnGestureLost?.Invoke();
            previousGesture = defaultGesture;
        }

        return;
    }

    if (GestureDetectionMode == GestureDetectionMode.Discrete && currentGesture.Equals(previousGesture)) return;

    if (CheckIfGestureIsInRotationThreshold(currentGesture))
    {
        currentGesture.OnGestureRecognizedWithCorrectRotation?.Invoke();
    }
    else
    {
        currentGesture.OnGestureRecognizedWithWrongRotation?.Invoke();
    }

    previousGesture = currentGesture;
}
```

Slika 5.5. Kôd *Update* funkcije

Slika 5.6. prikazuje funkciju *DetectGesture* koja sadrži algoritam prepoznavanja relativnih pozicija kostiju.

```
1 reference
public Gesture DetectGesture()
{
    Gesture currentGesture = new Gesture();
    float currentMinimalDistanceSquared = Mathf.Infinity;

    for (var i = 0; i < handGestures.Count; i++)
    {
        var sumDistanceSquared = 0f;
        var isAboveThreshold = false;

        for (var j = 0; j < handBones.Count; j++)
        {
            var boneLocalPosition = transform.InverseTransformPoint(handBones[j].Transform.position);
            float distanceSquared = (handGestures[i].HandBonePositions[j] - boneLocalPosition).sqrMagnitude;

            if (distanceSquared > thresholdSquared)
            {
                isAboveThreshold = true;
                break;
            }

            sumDistanceSquared += distanceSquared;
        }

        if (!isAboveThreshold && sumDistanceSquared < currentMinimalDistanceSquared)
        {
            currentMinimalDistanceSquared = sumDistanceSquared;
            currentGesture = handGestures[i];
        }
    }

    return currentGesture;
}
```

Slika 5.6 Kôd *DetectGesture* funkcije

Slika 5.7. prikazuje funkciju *CheckIfGestureIsInRotationThreshold* koja provjerava orijentaciju ruke kako bi se izbjegao gore spomenuti problem krivog detektiranja palac dolje geste kao palac gore.

```
1 reference
public bool CheckIfGestureIsInRotationThreshold(Gesture gesture)
{
    var dotValueUp = Vector3.Dot(transform.up, Vector3.up);
    var dotValueRight = Vector3.Dot(transform.right, Vector3.up);
    var dotValueForward = Vector3.Dot(transform.forward, Vector3.up);

    var deltaUp = dotValueUp - gesture.DotValueUp;
    var deltaRight = dotValueRight - gesture.DotValueRight;
    var deltaForward = dotValueForward - gesture.DotValueForward;

    var rotationThreshold = gesture.RotationThreshold;

    var isInThresholdUp = Mathf.Abs(deltaUp) < rotationThreshold;
    var isInThresholdRight = Mathf.Abs(deltaRight) < rotationThreshold;
    var isInThresholdForward = Mathf.Abs(deltaForward) < rotationThreshold;

    return (isInThresholdUp && isInThresholdRight && isInThresholdForward);
}
```

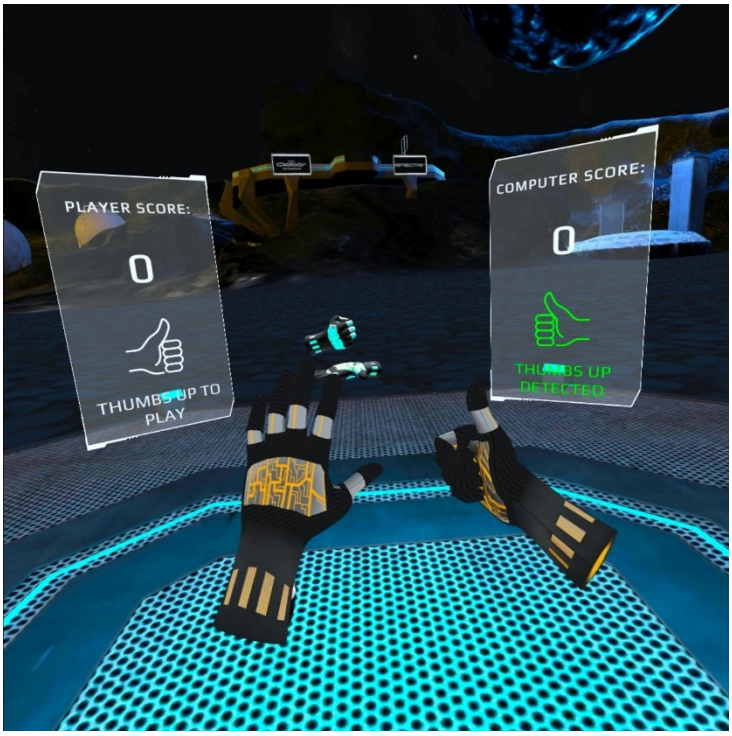
Slika 5.7. Kôd *CheckIfGestureIsInRotationThreshold* funkcije

6. PRIKAZ I PRIMJENA IMPLEMENTACIJE

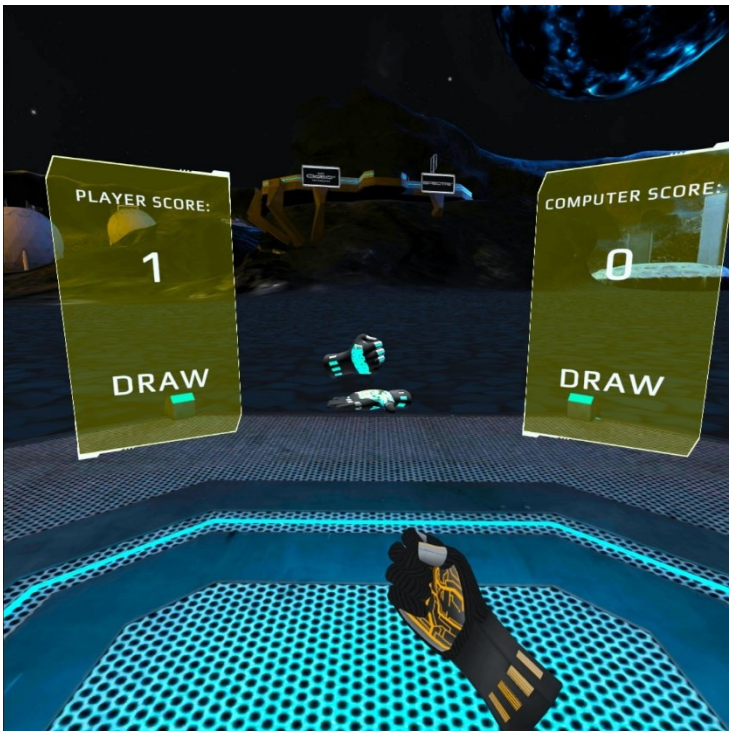
U ovom će poglavlju biti opisano nekoliko primjera implementacije algoritma. Napravljene dvije aplikacije dostupne su za preuzimanje na platformi *SideQuest* pod nazivom *SpaceArcadeVR*.

6.1. KAMEN, ŠKARE, PAPIR

Kamen, škare, papir poznata je stara igra koja koristi tri različite geste. Kamen predstavlja gestu u kojoj su sva tri zgloba na svih pet prstiju ruke u potpunoj fleksiji (stisnuta šaka). Škare je gesta u kojoj su sva tri zgloba malog prsta, prstenjaka i palca u potpunoj fleksiji, a zglobovi srednjeg prsta i kažiprsta u ekstenziji. Papir je geste u kojoj su svih pet prstiju u ekstenziji (ispruženi). Kamen pobjeđuje škare, škare pobjeđuju papir, a papir pobjeđuje kamen. U ovoj aplikaciji korisnik igra protiv računala. Računalo nasumično generira jednu od tri geste, a korisnik mora po isteku vremena pokazati svoju. Pobjednik je onaj tko osvoji tri igre. Prikaz igre i različitih ishoda prikazan je od slike 6.1 do slike 6.5.



Slika 6.1. Prikaz izbornika za početak igre



Slika 6.2. Neodlučena igra. Igrač i računalo pokazali su kamen

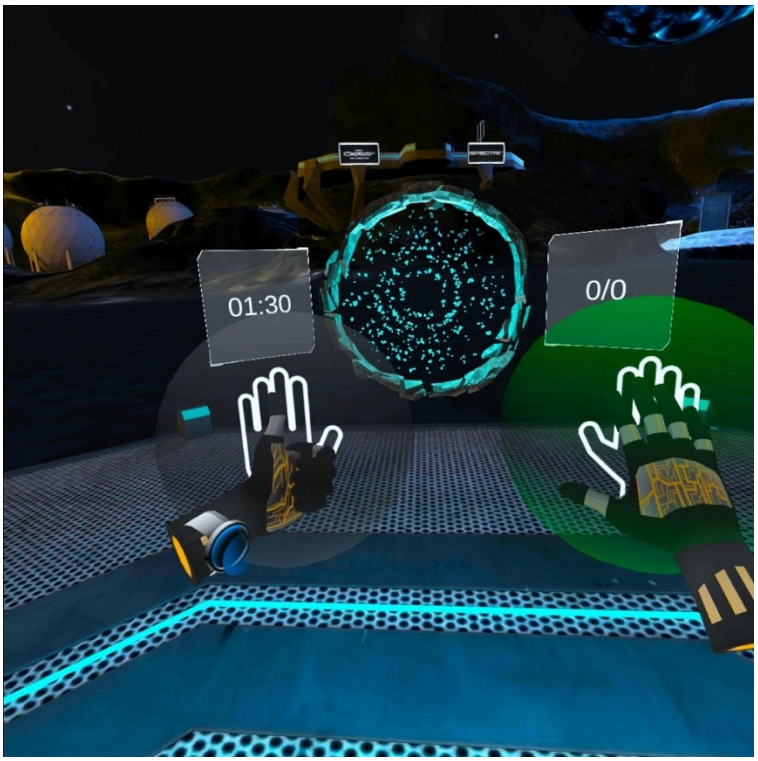
Slika 6.3. Igrač je izgubio pokazavši kamen, a računalo papir

Slika 6.4. Igrač je pobijedio pokazavši papir, a računalo kamen

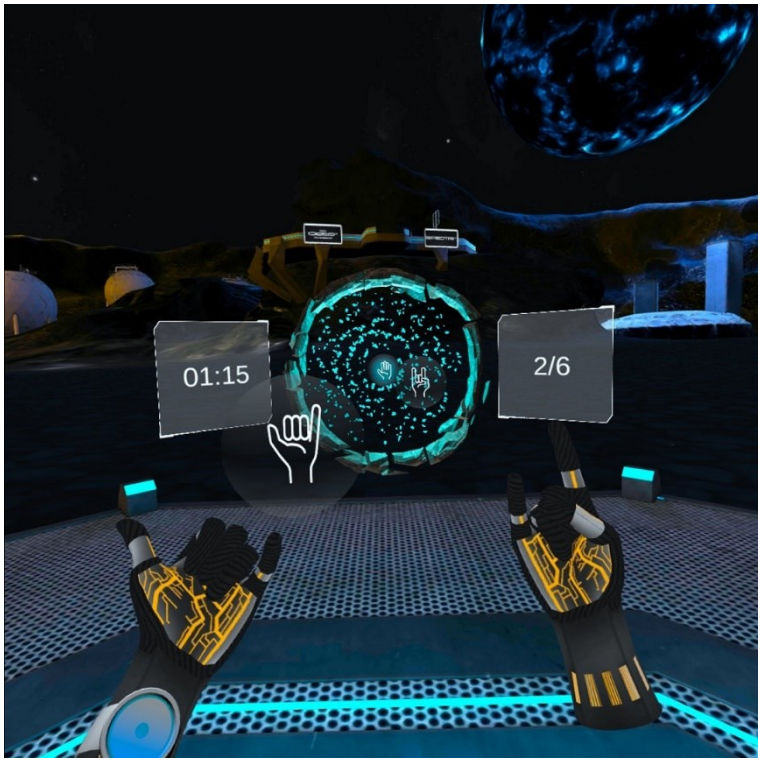
Slika 6.5. Igrač je pobijedio pokazavši škare, a igrač papir

6.2. GESTURE HERO

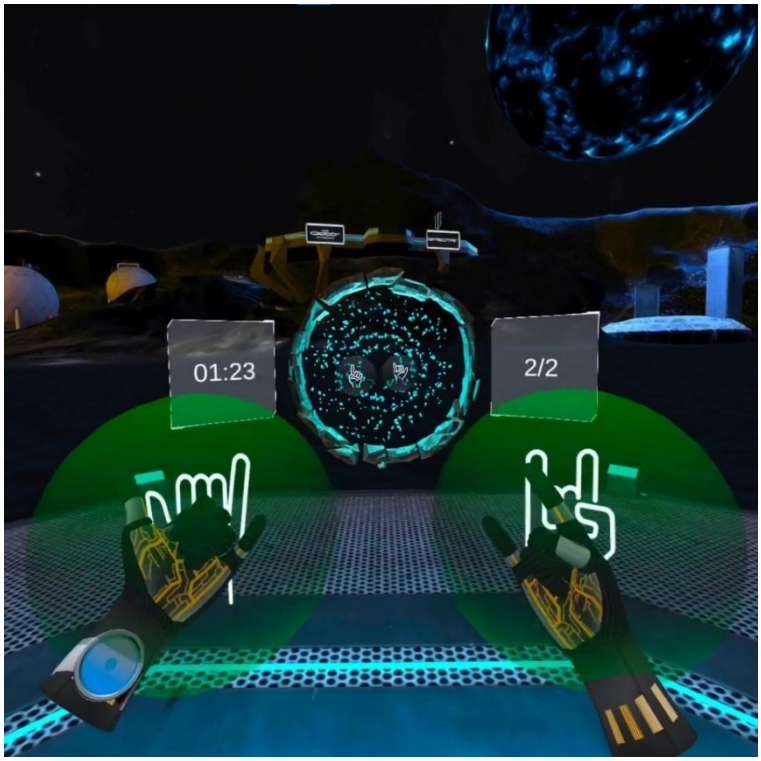
Gesture hero aplikacija je sa sličnom mehanikom kao proslavljena igra *Guitar Hero*. Korisnik se nalazi u prostoru i prema njemu dolaze krugovi u dvije linije. Jedan prema njegovoj lijevoj, a drugi prema njegovoj desnoj ruci. Svaki krug sadrži jednu gestu. Korisnik mora staviti svoju ruku u svaki krug i pokazati gestu prikazanu na njemu. Kako vrijeme igre odmiče, geste idu prema korisniku sve brže i brže te ih je sve više i više. Ovisno o postotku pogođenih gesta, korisnik na kraju igre dobije primjerenu medalju. Mehanika igre prikazana je od slike 6.5. do slike 6.10.



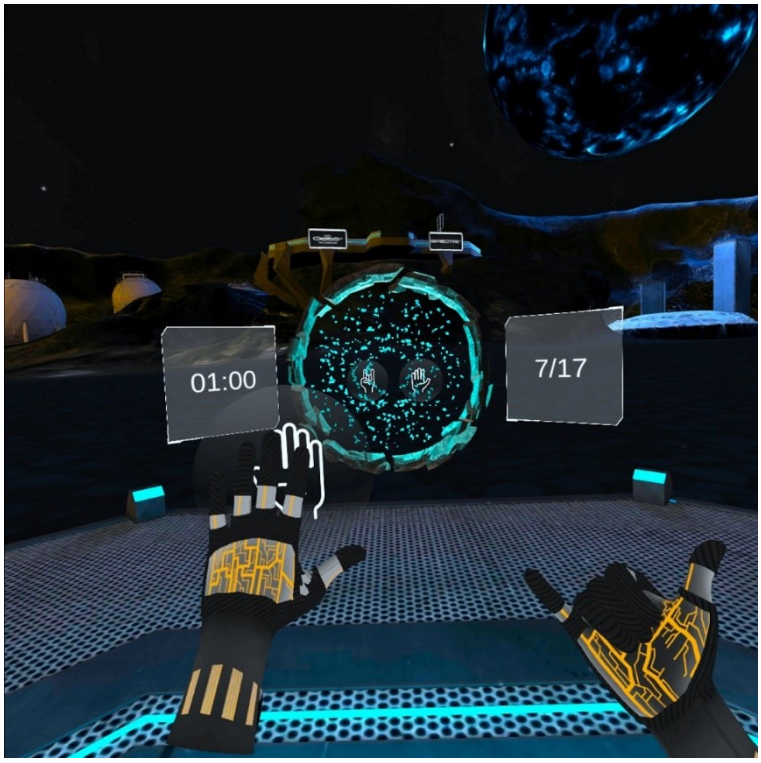
Slika 6.6. Početak igre detekcijom dvije geste otvorenog dlana



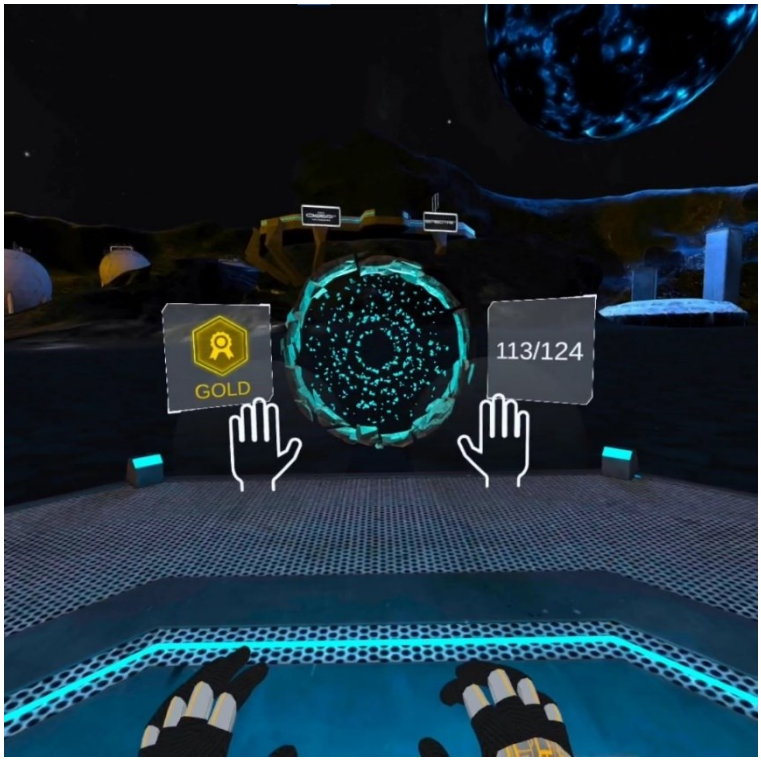
Slika 6.7. Geste dolaze u susret korisniku



Slika 6.8. Korisnik je točno pokazao obje geste



Slika 6.9. Prikaz nadolazećih gesta



Slika 6.10. Završetak jedne igre

7. ZAKLJUČAK

Tehnologije proširene stvarnosti još su uvijek relativno mlade i nisu doživjele svoj puni procvat. Velikom većinom zato što su takvi uređaji skupi i teško dostupni široj masi. Revoluciju na tržištu takvih uređaja napravio je Meta (u to vrijeme Facebook) sa svojim *Oculus Quest 2* uređajem. Njegova cijena se trenutno kreće kao i PlayStation 5, a broj aplikacija koje podržava je sve veći i veći. Budući da se radi o android uređaju, podržava veliku većinu običnih android aplikacija uz aplikacije proširene stvarnosti. U srpnju 2021. godine *Oculus* je uveo *passthrough* koji je nadogradio ovaj uređaj sa podržavanja isključivo VR sadržaja na MR i AR sadržaj.

Kako uređaji postaju sve dostupniji, važno je da korisnicima budu što ugodniji i jednostavniji za korištenje. Ovdje veliku prednost ima korištenje ruku. Gotovo svaki ovakav uređaj dolazi sa svojim pripadajućim kontrolerima. Istima se može upravljat i u početnom izborniku, aplikacijama, igrama i slično. Ono što je zajedničko svima je da pri pokretanju svake aplikacije mora biti objašnjeno na neki način kako se ta aplikacija koristi. Kako je koja tipka kontrolera povezana sa različitim akcijama i mogućnostima aplikacije. Korištenjem ruku taj se dio praktički može izbaciti. Ruke su intuitivne. Čine interakcije lakšima i prirodnijima. Za hvatanje stvari se ne mora stisnuti tipka, treba se samo napraviti gesta hvatanja. Korisnici koji prvi puta stave naočale na glavu se teško snalaze u virtualnom svijetu. Prirodne interakcije rukama su upravo ono što im pomaže da se brže i lakše prilagode virtualnim okruženjima i interakcijama s istima.

Ljudi u svojoj svakodnevnici često gestikuliraju rukama. Bilo to nesvjesno odmahivanje, upozoravanje na opasnost, neverbalna komunikacija, znakovni jezik, ovaj način komunikacije se učestalio. Prepoznavanje tih gesta još je jedan dodatan sloj u približavanju virtualnog svijeta ljudima te ih čini još jednostavnijima i bližim korisnicima. Korištenjem ovog sustava prepoznavanja gesta, korisnik može vrlo lako povezati neke funkcionalnosti s određenom gestom i ubrzati si razvoj aplikacije.

Ovaj sustav može se još nadograditi. Cijeli rad baziran je na statičkim gestikulacijama ruku koje su neovisne jedna od druge. Ostaje dalje implementirati sustav dinamičkog prepoznavanje gesti kao što su odmahivanje pokazivanje da netko priđe bliže i slične. Takva gesta kombinacija je statički geste i brzine kretanja same ruke u različitim smjerovima. Nakon toga preostaje implementirati kombinaciju dvije dinamičke geste na dvije različite ruke. Najbolji primjer jedne takve geste bi bio pokazivanje koraka u košarci ili rukometu. Radi se o gesti koja ima prste na oba

dlana ruke u potpunoj fleksiji, a ruke rade pokret rotacije oko druge ruke. Gesta je prikazana na slici 7.1.

Slika 7.1. Dinamička gesta koraka u košarci [11]

Osim nadograđivanja sustava gestikulacije, treba još poboljšati efikasnost algoritma i pronaći načine da se izbjegnu teške matematičke operacije (jer se ipak ovaj algoritam izvodi svaki *frame* aplikacije).

Ovaj implementirani sustav ima vrlo slično sučelje kao Oculusov *PoseDetecion* opisan u poglavlju 2. Oboje prema korisniku pružaju popis gesta te vezivanje određenih akcija za te pojedine geste, ali „ispod haube“ to prepoznaju na drugačije načine. U trenutku pisanja ovog rada Oculusov *PoseDetection* je također ograničen samo na statičke gestikulacije.

LITERATURA

[1] https://google.github.io/mediapipe/solutions/hands - kolovoz 2022.

[2] https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html - kolovoz 2022.

[3] https://blog.tensorflow.org/2019/11/handtrackjs-tracking-hand-interactions.html - kolovoz 2022.

[4] https://www.manomotion.com/ - kolovoz 2022

[5] https://developer.oculus.com/documentation/ - kolovoz 2022.

[6] https://developer.oculus.com/documentation/unity/unity-isdk-hand-pose-detection/ - kolovoz 2022.

[7] https://youtu.be/lBzwUKQ3tbw - kolovoz 2022.

[8] https://github.com/andypotato/fingerpose - kolovoz 2022.

[9] https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/ - kolovoz 2022.

[10] https://ai.facebook.com/blog/powered-by-ai-oculus-insight/ - rujan 2022.

[11] https://searchresearch1.blogspot.com/2015/03/traveling.html - rujan 2022.

SAŽETAK

U ovom je diplomskom radu opisana implementacija sustava prepoznavanja gesta u tehnologijama proširene stvarnosti te zašto je isti koristan pri korištenju praćenja ruku. Ovaj sustav je implementirat na uređaju *Oculus Quest 2* koristeći njihov *Oculus SDK* i *Unity game engine*, ali radi i na ostalim uređajima koji imaju mogućnost praćenja ruku. Kreirana gesta zatim je prenosiva na sve ostale platforme koji koriste kostur istog tipa. Neki modeli ruku napravljeni su drugačije te se palac i mali prst sastoje od četiriju kostiju. Za primjer u ovom diplomskom radu koristio se palac i mali prst od triju kostiju. Kreirane geste na modelu s četiriju kostiju rade na svim modelima ruku koje sadrže četiri kosti. Osim implementacije samog sustava, implementiran je i primjer korištenja algoritma kroz nekoliko različitih aplikacija.

Ključne riječi: Pomiješana stvarnost, praćenje ruku, prepoznavanje gesta, proširena stvarnost, virtualna stvarnost.

ABSTRACT

Title: Hand tracking gesture recognition in extended reality

This thesis describes the implementation of a gesture recognition system in extended reality technologies and why it is useful when using hand tracking. This system was implemented on the Oculus Quest 2 device using their Oculus SDK in the Unity game engine, but it also works on other devices that have hand tracking capabilities. Created gesture is then compatible with other platforms that use the same skeleton type. Some hand models are made differently, and the thumb and little finger consist of four bones. For the example in this thesis, the three-bone thumb and little finger were used. Gestures created on a four-bone model are compatible with all hand models that contain four bones. In addition to the implementation of the system itself, an example of using the algorithm through several different applications was shown.

Keywords: Augmented reality, extended reality, gesture recognition, hand tracking, mixed reality, virtual reality.

ŽIVOTOPIS

Matija Fumić rođen je 31. ožujka 1998. godine u Osijeku. Odrastao je u Belišću gdje završava Osnovnu školu Ivana Kukuljevića. Srednjoškolsko obrazovanje nastavlja u Valpovu, smjer Opća gimnazija. 2016. godine upisuje sveučilišni preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Po završetku preddiplomskog studija upisuje diplomski sveučilišni studij smjer Programsko inženjerstvo. Osim što je student, zaposlenik je tvrtke SpectreXR uz čiju pomoć je izrađen ovaj diplomski rad. Također je i licencirani rukometni trener od strane Hrvatskog olimpijskog odbora te trenira mlađe kategorije u Rukometnom klubu Valpovka iz Valpova.