SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 4795

# MODELIRANJE OBJEKATA U VIRTUALNOM PROSTORU UPORABOM PRIRODNOG KORISNIČKOG SUČELJA

Filip Matijević

Zagreb, lipanj 2017.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 3. ožujka 2017.

# ZAVRŠNI ZADATAK br. 4795

Pristupnik: **Filip Matijević (0036474351)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Modeliranje objekata u virtualnom prostoru uporabom prirodnog korisničkog sučelja**

Opis zadatka:

Proučiti tehnologije koje omogućuju korisnicima prirodne načine zadavanja trodimenzionalnih objekata. Posebice razmotriti zadavanje točke, linije i boje kao temeljnih elemenata objekta u virtualnom prostoru. Razmotriti mogućnosti implementacije aplikacije na mobilnim platformama. Realizirati aplikaciju za modeliranje objekata korištenjem proučenih tehnologija. Na različitim primjerima prezentirati ostvarene rezultate. Načiniti ocjenu rezultata i implementiranih algoritama.
Izraditi odgovarajući programski proizvod. Koristiti potrebne grafičke programske alate. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 9. lipnja 2017.

Mentor:

Prof. dr. sc. Željka Mihajlović

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

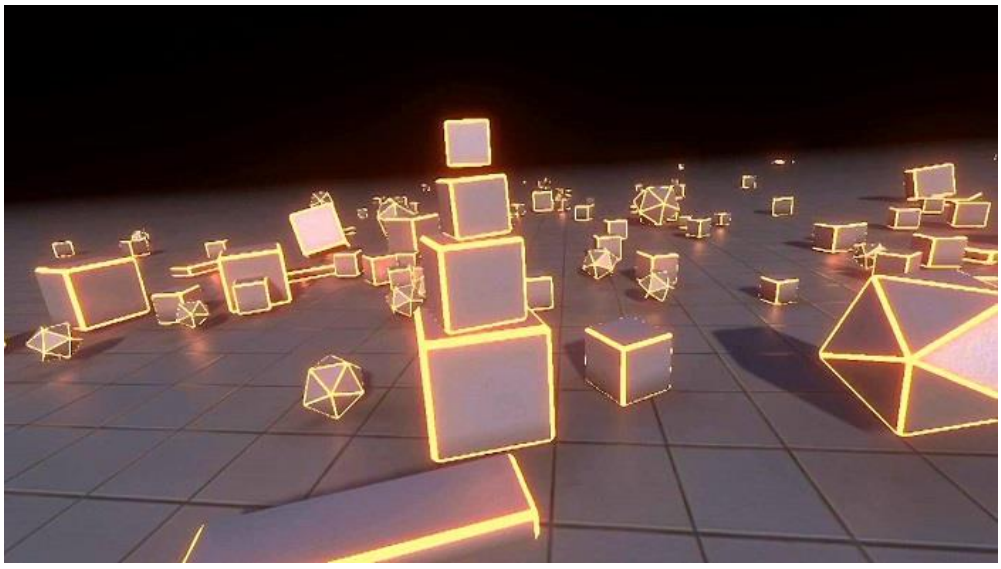
# Sadržaj

# 1. Uvod

U ovome radu objasnit će se način korištenja prirodnog korisničkog sučelja pri upravljanju aplikacije za stvaranje trodimenzionalnih slika koristeći osnovna geometrijska tijela. Ideja je skaliranjem i translacijom tih osnovnih tijela modelirati novi, kompleksan oblik. Glavni fokus ovoga rada je iskorištavanje informacija dobivenih iz LeapMotion uređaja, dok sama aplikacija služi za demonstracijske svrhe kontrolera i korištenje skripti pri modeliranju objekata u stvarnome vremenu.

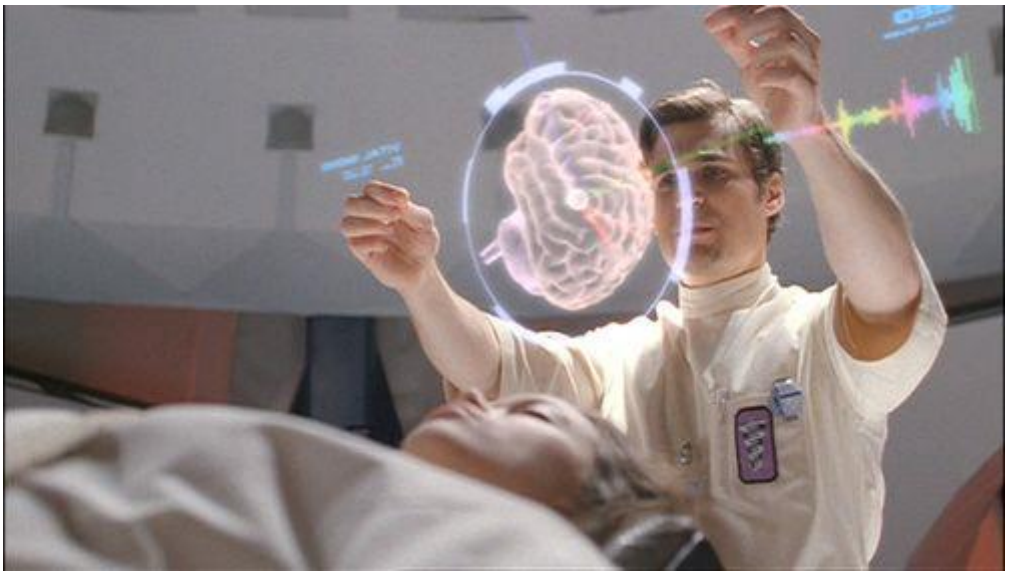
Prednosti korištenja prirodnog korisničkog sučelja jesu mogućnost interakcije sa virtualnim prostorom na intuitivan način, slobodu i preciznost naredbi te jednostavno korištenje.

Rad je inspiriran demonstracijskim primjerom LeapMotion uređaja dostupnom u nadograđenome sustavu „Orion“. Prelaskom na unaprijeđene algoritme praćenja koji su dodani u Orion, otvorila se mogućnost preciznog očitavanja koordinata zglobova u prstima kao i gesti napravljenih rukama.



Slika 1 – Demonstracija LeapMotion – Orion sustava

Inspiracija ovoga rada također su i brojni znanstveno – fantastični filmovi u kojima glumci kao osnovno sučelje preko kojega upravljaju računalima su upravo njihove ruke i geste.



Slika 2 – Prirodno korisničko sučelje

# 2. Sklopovska oprema

Prilikom razvijanja ove aplikacije, kao ulazni uređaj, koristio se Leap Motion kontrolor koji korisniku omogućava upravljanje sučeljem koristeći isključivo vlastite ruke.

## 2.1 Uređaj Leap Motion

Slika 3 - Uređaj LeapMotion

Leap Motion je kontrolor malih dimenzija sposoban za detekciju i praćenje ruku korisnika. To izvodi pomoću dva optička senzora i tri infracrvena svjetla vidljiva na vrhu uređaja. Svrha navedenog kontrolora bila je povezivanje s uređajem Oculus Rift kako bi korisnik mogao upravljati aplikacijama reproduciranim u virtualnoj stvarnosti. Naknadno se podrška proširila i na stolna računala gdje se u potpunosti može izbjeći korištenje konvencionalnih upravljačkih medija te se osloniti na prirodno korisničko sučelje (položaji ruku zajedno s gestama). Ova dva načina rada razlikuju se samo u predodžbi gdje se senzor nalazi u

virtualnom svijetu. U slučaju Desktop verzije, uređaj se obično nalazi ispred monitora gdje korisnik istovremeno vidi vlastite ruke, zajedno s virtualnom reprezentacijom ruku na monitoru.

Slika 4 - Uređaj Oculus zajedno s uređajem Leap Motion

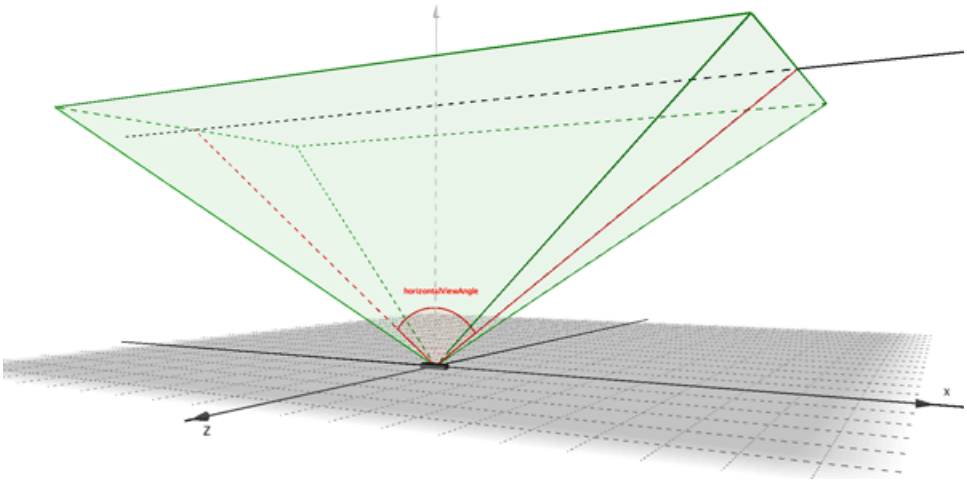
Drugi slučaj, korištenja Leap Motion uređaja s Oculus Rift-om je njegovo montiranje na sami Oculus kao što prikazuje slika 2. Implementacija ovoga projekta temelji se na prvom načinu rada, gdje se uređaj nalazi ispred ekrana te korisnik ruke stavlja u njegovo vidno polje ukoliko želi započeti interakciju s virtualnim svijetom.

Mane ovoga uređaja su vidljive u previše ili premalo osvjetljenim prostorijama. Uređaj ima sposobnost prilagodbe na strane izvore infracrvenog zračenja, ali u većini slučajeva, sinkronizacija virtualnih i stvarnih ruku nestaje.

Najbolje performanse uređaj pokazuje kada se nalazi u umjereno osvjetljenoj prostoriji, gdje izvor svjetla nije u vidnom polju uređaja te se uređaj nalazi u blizini zida (ili bilo koje ravne okomite plohe). Dodatne optimizacije moguće su kroz postavke uređaja ali ja osobno nisam uočio velike promjene.

## 2.2 Veza stvarnog i virtualnog svijeta

Pri korištenju Leap Motion kontrolora važno je obratiti pažnju na njegov koordinatni sustav. U standardnom, neizmijenjenom načinu implementiranja kontrolora u Unity3D prostor, milimetar u stvarnome svijetu preslikava se u jednu jedinicu koordinatnog sustava u virtualnome svijetu. Iako je kontrolor ishodište koordinatnog sustava, pravilnim postavljanjem hijerarhije virtualne kamere i kontrolora omogućuje pomicanje tog ishodišta te interakcijom s prije nedostižnim dijelovima virtualnog svijeta.



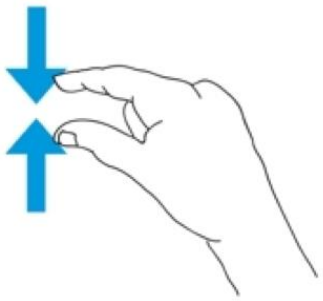

Slika 5 – Koordinatni sustav LeapMotion-a u sceni

# 3. Geste

Osnovni način komuniciranja korisnika s aplikacijom je putem gesti. Leap Motion posjeduje ugrađene dodira zaslona, dodira tipkovnice, pokreta prsta u proizvoljnom smjeru te iscrtavanje kružnice prstom.

Slika 6 – Predodređene geste LeapMotion paketa

U pogledu aplikacije ostvarene u ovom radu, nisu korištene predefinirane geste već su napravljene vlastite radi mogućnosti dodavanja novih funkcionalnosti.



Slika 7 – Gesta štipanja

Osnova gesta kojom se ostvarena aplikacija koristi jest gesta „štipanja“ uz kontinuiranu kontrolu udaljenosti palca i kažiprsta u stvarnome vremenu.

Iako su se mogle koristiti već implementirane geste pri stvaranju alata u aplikaciji, njihovo korištenje bi bilo poprilično neprecizno te bi korisnik se trebao naučiti na način izvršavanja geste prije nego što ju može koristiti u punom potencijalu. Primjer za to bio bi pomicanje kamere. Prirodno bi bilo pomicanje u smjeru zamaha prstom, ali to nam otvara problem za koliko jedinica u koordinatnome sustavu korisnik želi pomaknuti kameru.

## 3.1 Povezivanje Unity 3D I Leap Motion Kontrolora

U početnome stadiju razvoja aplikacije napravio sam grešku gdje sam koristio koordinate virtualnih ruku dostupnih u Leap Motion demonstracijskoj sceni. To mi je stvaralo velike probleme jer se cijelio vidno polje kontrolora skaliralo unutar kocke dimenzija 1x1x1. Kako Unity3D ima poteškoća pri simulacije fizikalnih zakona na malim predmetima, taj model sam skalirao kako bi sustavu i meni bilo lakše upravljati njime. To naime, nije dobar pristup. Ovim načinom ne vrijedi prije navedena korelacija stvarnih milimetara s virtualnim koordinatnim sustavom.

Kako bi se ispravno pristupilo Leap Motion kontroloru, potrebno je stvoriti skriptu sa sljedećim elementima.

Na samome početku skripte, pokraj standardnih Unity3D i C# zavisnosti, potrebno je dodati one odgovorne za upravljanje Leap Motion kontrolerom.

```
using Leap;
using Leap.Unity;
```

Sada stvorena skripta ima mogućnost pristupu kontroleru. Potrebno je deklarirati varijablu tog tipa te u Start metodi referencirati ga na tu varijablu.

```
Controller controller;

controller = new Controller();
```

Rukama pristupamo tako da iz definiranog kontrolera uzima se jedan vremenski okvir (engl. Frame) koji sadrži informacije o rukama i njenim komponentama u kadru. Bitno za naglasiti

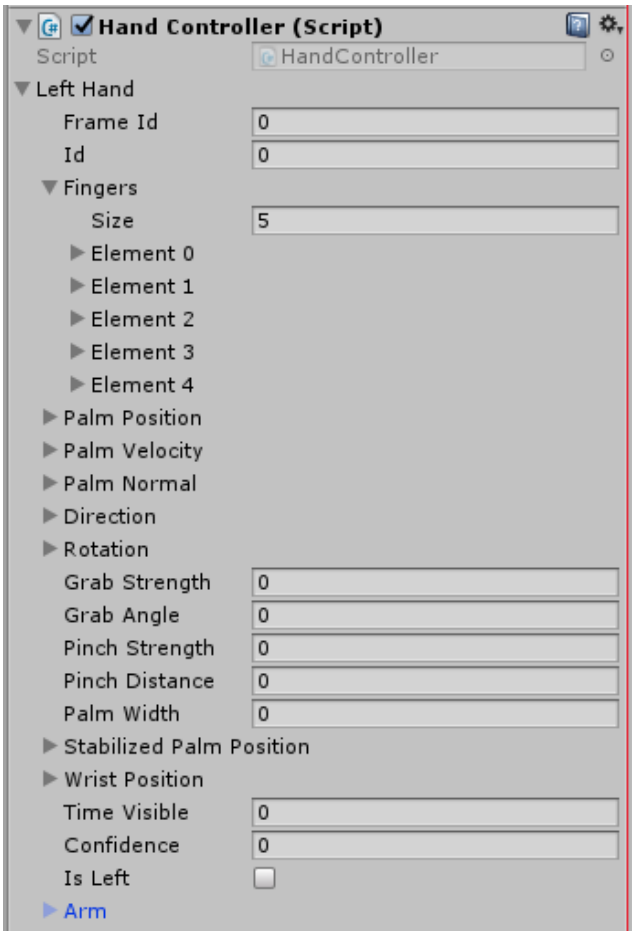
je da Leap Motion posjeduje sposobnost obrade i do 200 slika u sekundi. Kako ovdje koristimo lijevu i desnu ruku, korisno je imati reference na njih.

```
public Hand LeftHand;
public Hand RightHand;

…

…

Frame frame = controller.Frame();

List<Hand> hands = frame.Hands;
foreach (Hand x in hands)
{
    if (x.IsLeft)   LeftHand = x;
    if (x.IsRight)  RightHand = x;
}
```

Ovaj pristup prilagođen je za upravljanje aplikacija u kojima će biti samo jedan korisnik s lijevom i desnom rukom. Pokraj metoda „IsLeft“ i „IsRight“ koje vraćaju logičku istinu ili laž u ovisnosti o tome kako je kontroler klasificirao ruku, postoje metode „leftmost“ i „rightmost“. Koriste se na drugačiji način gdje je povratna vrijednost tipa *Hand* a primjenjuje se na listu. Te metode nisu korištenje u razvoju ove aplikacije ali su korisne pri stvaranju igrica gdje će biti dva korisnika unutar jednog kontrolera (dobacivanje lopticom, na primjer).

Jedina potrebna pretvorba koju je potrebno napraviti je ona iz tipa „Vector“ u „Vector3“. Naime, oba tipa podataka sadrže iste vrijednosti, ali Unity3D ne posjeduje taj tip podataka kojim Leap Motion barata. To se čini pozivanjem metode „ToVector3()“ nad određenim Leap Motion vektorom.

Nakon što je skripta dodijeljena nekome objektu, u inspektoru bi trebale biti dostupne sve informacije o lijevoj i desnoj ruci.




Slika 8 – Svojstva I postavke instancirane varijable tipa “Hand”

Iz razloga što su varijable ruku postavljene kao javne, sada iz bilo koje druge skripte možemo referencirati „HandController“ koji u sebi sadrži sve potrebno za rad s rukama.

```
private HandController Hands;

…

…

Hands.RightHand.Finger(0).TipPosition.ToVector3();
```

### 3.1.1 Opis programske implementacije novonastale geste

Gestu možemo definirati kao oblik neverbalne komunikacije kojom se prenose važne informacije. U ovome slučaju, geste se isključivo izvršavaju rukama. Prvi korak implementacije nove geste jest osmislit ju na način da ju korisnici sustava lako upamte.

Ustanovljeno je da gesta pritiska kažiprsta i palca je daleko najbolja zamjena za pritisak i držanje tipke miša. Većina korisnika već je naučila da pritiskom tipke miša je nešto moguće pomjeriti, promijeniti dimenzije i slično.

Prvi korak stvaranja ove geste jest postavljanje varijabli tipa „Transform“. Varijabla tog tipa u sebi sadrži rotaciju, translaciju i skalu danog objekta. U ovome slučaju, potrebna nam je isključivo translacija.

Kao što je već spomenuto u prijašnjem poglavlju, odvojena skripta brine se o koordinatama dijelova ruku pa je skripti geste dovoljna referenca na položaj palca i kažiprsta.

```
public Transform Thumb_Tip;
public Transform Index_Tip;

public bool Pinched;

…
```

Varijablu imena „Pinched“ koristit će druge skripte koje će izvršavati nastavak geste, ukoliko je napravljen kontakt dva prsta.

Sami prazan objekt koji nosi tu skriptu je pomičan i koristi se za određivanje gdje u prostoru je gesta inicijalizirana. Objekt se uvijek nalazi na srednjoj udaljenosti koordinata kažiprsta i palca

```
void Update () {
	this.transform.position = (Thumb_Tip.position + Index_Tip.position)/2;
	if (Vector3.Distance(Thumb_Tip.position,Index_Tip.position)<0.03f ) {
	Pinched = true; } else { Pinched = false; }
	}
```

Padne li udaljenost dva prsta ispod definirane konstante, sustav to registrira kao početak geste i tu informaciju uzima kao naredbu.

Dok su prsti spojeni, gesta se nastavlja izvoditi sa proširenim mogućnostima, ovisno o tome koji alat je odabran i što želimo napraviti.

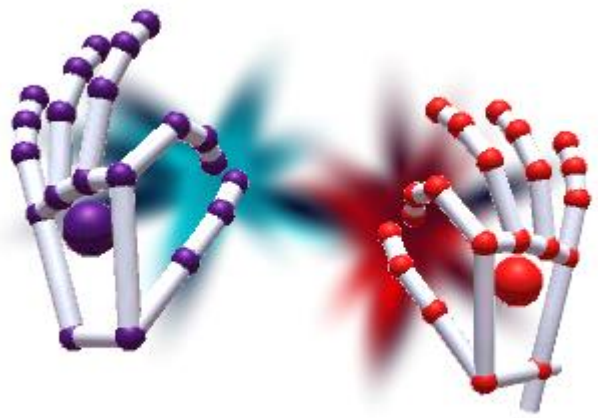
U slučaju translacije i rotacije kamere, gestom pomicanja suprotne ruke od one koja drži pritisak, korisnik sustavu govori u kojem smjeru i kojom brzinom da se kamera pomakne.

```
if (Left_Pinch.Pinched)
 {
	Camera.main.transform.RotateAround(Vector3.up,
	Vector3.Distance(Right_Tip.position, Left_Pinch.Pinch_Location)/10);
 }
```

Navedena metoda „RotateAround“ kao argumente prima vektor oko kojeg se kamera okreće, te kut u stupnjevima. U ovome slučaju, veličina kuta određena je udaljenosti točke pritiska i suprotne ruke.

# 4. Upravljanje aplikacijom

Korisnik aplikacijom upravlja koristeći alate dostupne u izbornicima dostupnim u lijevoj i desnoj ruci te određenim gestama odgovornim za stvaranje i uklanjanje objekata, zajedno sa metodama kretanja po virtualnoj sceni. Osnovna gesta korištena u aplikaciji jest „pinch“, tj. Spajanje palca i kažiprsta.



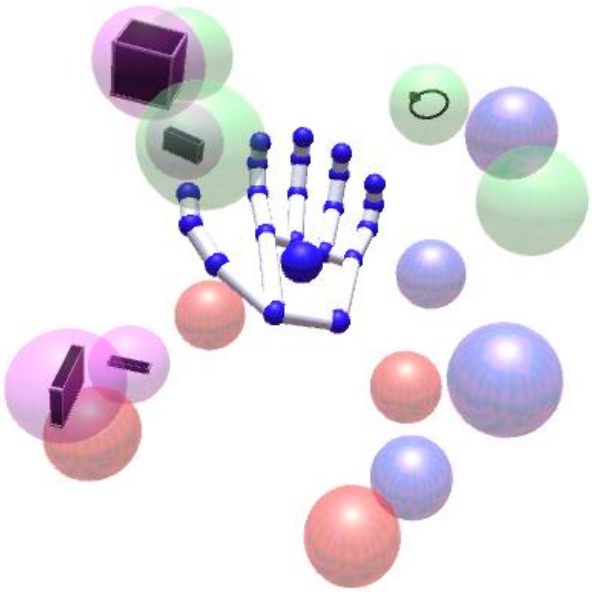
Slika 9 – Povratna informacija o početku geste

Korisnik ima vizualnu reprezentaciju interakcije pomoću svjetlosnih efekata implementiranih koristeći Particle Effects sustav dostupan u Unity 3D alatu.

Izbornicima se pristupa okretanjem dlana prema gore. Dok je izbornih aktivan, niti jedna druga gesta ili akcija se neće registrirati te je korisniku vidljivo isijavanje svjetla iz prsta suprotne ruke, koje reprezentira interakciju s izbornikom. U lijevoj ruci nalaze se svi alati i oblici koje korisnik može koristiti ili instancirati.

## 4.1 Alatni izbornik

Alatni izbornik nalazi se u lijevoj ruci. Sastoji se od četiri podskupine od kojih svaka sadrži po četiri alata. Tri skupine alata temelje se na osnovnim geometrijskim tijelima te njihovom manipulacijom dok je četvrta skupina odgovorna za pomicanje i rotaciju gledišta te uklanjanje i pomicanje odabranog predmeta u prostoru. Kako je toliku količinu alata nezgodno staviti na jedno mjesto, odabran je pristup rotirajućeg izbornika. Elementi izbornika rotiraju se oko korisnikove ruke kako bi sučelje bilo što preglednije. Brzina te rotacije ovisi o udaljenosti desne ruke od izbornika. Naime, što je desna ruka bliže lijevoj, to se izbornik sporije rotira, dok ne pređe granicu gdje u potpunosti staje i pruža korisniku mogućnost odabira alata. Odabir alata popraćen je vizualnim bljeskom nakon dodira prsta sa elementom.

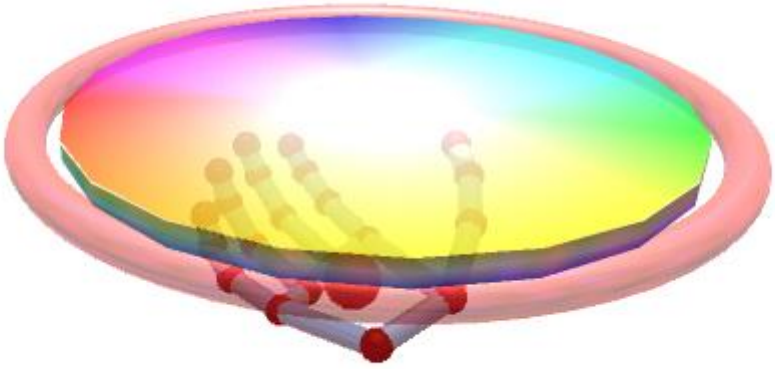


Slika 10 – Izbornik alata na lijevoj ruci

Radi veće preciznosti odabira moguće je dodati zraku koja spaja prst desne ruke s trenutno najbližim elementom izbornika u prostoru.

Okretanjem dlana prema dolje u bilo kojem trenutku uzrokuje nestanak izbornika i nastavak rada aplikacije.

## 4.2 Izbornik Boja



Slika 11 – Izbornik boje na desnoj ruci

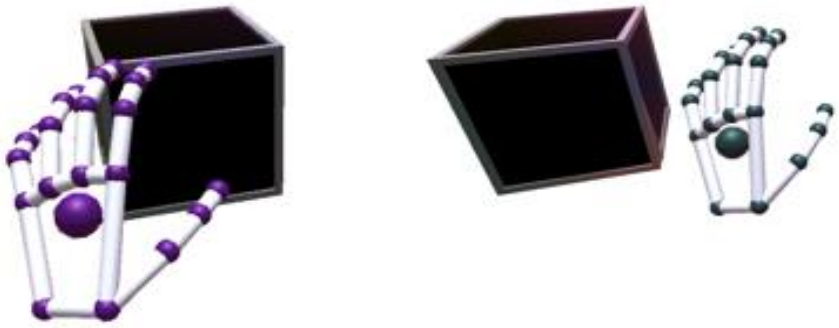
U desnoj ruci korisničkog sučelja nalazi se izbornik boja. Način rada je sličan kao i u alatnom izborniku gdje se kažiprstom lijeve ruke odabire boja. Implementacija uređaja Leap Motion omogućuje mi praćenje koordinate vrha prsta, zajedno s vektorom normale u smjeru gdje prst pokazuje. To zajedno s metodama bacanja zraka i dohvaćanja boje u pogođenoj točki omogućuje pohranu cijelog vidljivog spektra u teksturu iz koje se uzima boja.

Obruč izbornika predstavlja trenutno odabranu boju.

## 4.3 Manipulacija Očišta

### 4.3.1 Translacija Kamere

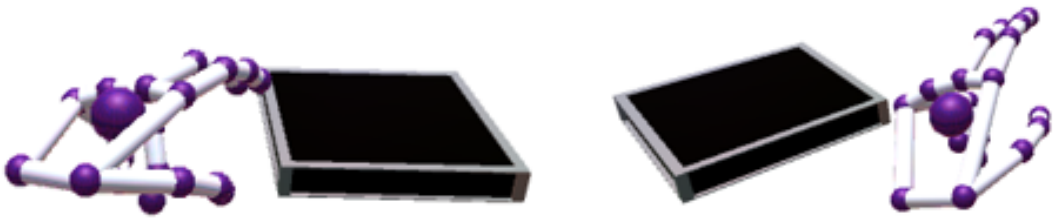
Postupak pomicanja kamere implementiran je u svrhu stvaranja većih nakupina osnovnih modela gdje ne stanu svi u kadar. Ta funkcija implementirana je na način da se ishodište Leap Motion-ovog koordinatnog sustava pomiče zajedno s kamerom (očištem korisnika). Kada se korisnik nalazi u stanju pomicanja kamere, on to radi pritiskom kažiprsta i palca na bilo kojoj ruci. Kamera se pomiče u smjeru vektora od točke pritiska prema kažiprstu druge ruke. Brzina pomicanja je konstantna.



Slika 12 – Translacija kamere

### 4.3.2 Rotacija Kamere

Korisniku je omogućena rotacija kamere isključivo oko Y osi (Vector3.up) globalnog koordinatnog sustava. Ovo je postavljeno da se izbjegne neželjeni nagib kamere do kojeg može doći rotacijom u više smjerova istovremeno. Rotacija se izvršava gestom pritiska kažiprsta i palca, gdje je smjer rotacije suprotan od ruke kojom je napravljena gesta. Brzina rotacije ovisna je o međusobnoj udaljenosti ruku.

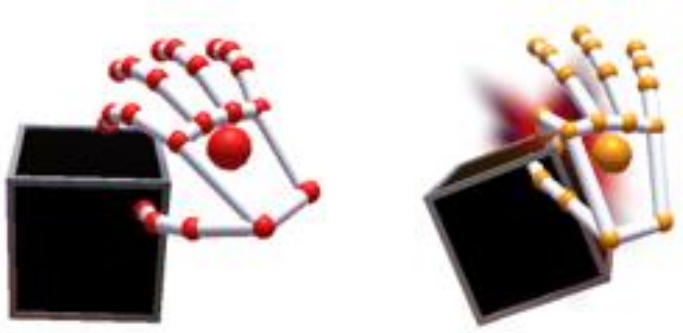


Slika 13 – Rotacija kamere

## 4.4 Manipulacija predmeta

### 4.4.1 Translacija predmeta

Postupak translacije predmeta od korisnika zahtjeva korištenje desne ruke. Pritiskom kažiprsta i palca započinje postupak traženja predmeta. Aktivira se objekt na koordinatama točke pritiska. Kada taj objekt uđe u drugo strano tijelo, on ga odabire i to tijelo počinje pratiti korisnikovu ruku. Nakon što korisnik prestane koristiti gestu, predmet se otpušta. Demonstracijski primjeri Leap Motion kontrolora prikazuju kako je moguće ostvariti realno podizanje predmeta, ali u ovoj aplikaciji nije implementirana slična metoda.



Slika 14 – Translacija predmeta

### 4.4.2 Rotacija predmeta

Slično kao i translacija, rotacija je realizirana usmjeravanju predmeta u smjer točke pritiska bez promjene njegovog vektora pozicije.

Slika 15 – Rotacija predmeta

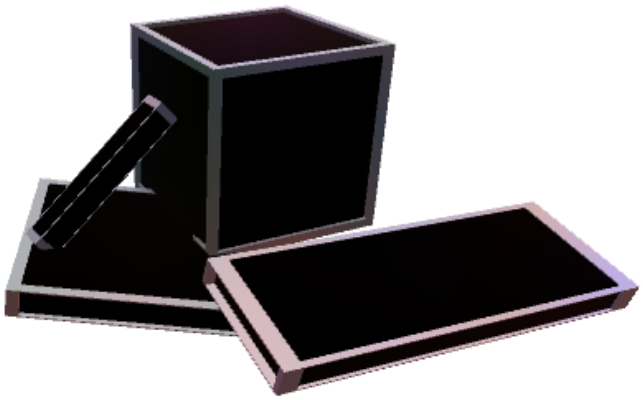
## 4.5 Brisanje predmeta

U sceni postoje točke u prostoru koje neprestano provjeravaju sudaraju li se s nekim od stvorenih predmeta. Njihova uloga je pronalaženje predmeta u virtualnom prostoru sa kojim korisnik želi manipulirati. Kako je prije navedeno za rotaciju i translaciju, također postoji skripta koja uništava prvi dotaknut predmet. Trenutno skripta radi na način da onaj predmet u kojemu se izvede gesta štipanja trajno nestane. Za sada nije implementirana funkcija vraćanja u prijašnje stanje (Eng. „Undo“).

# 5. Stvaranje Predmeta

## 5.1 Predmeti Prve skupine (varijacije kvadra)

U izborniku skupine objekata roze boje nalaze se oni objekti koji se mogu izvesti skaliranjem kombinacija x, y i z osi kocke. To su osnovni modeli koji su dovoljni za izgradnju jednostavnih skulptura kao što su kuće ili zgrade



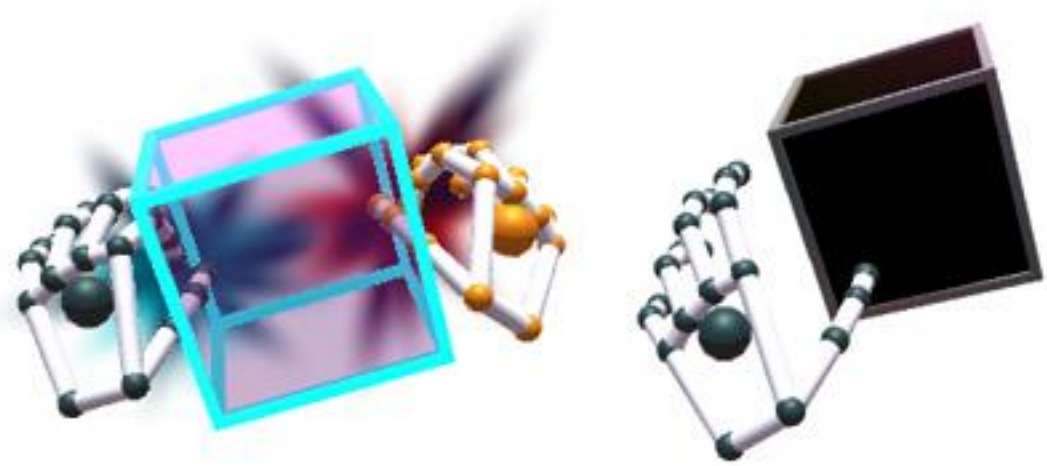
Slika 16 – Objekti prve skupine

Iako postoji više varijacija skaliranja kocke, odabrana su ova četiri prikazana na slici 14 jer svi ostali su izvedivi kombinacijama nekih od njih. Na primjer, u ovome sustavu nije moguće

stvoriti dasku proizvoljne debljine ali je moguće posložiti više njih jedna na drugu kako bi se dobio sličan učinak.

### 5.1.1 Stvaranje Kocke

Postupak stvaranja kocke započinje izvođenjem geste štipanja na objema rukama. U tome trenutku stvara se prozirna hologramska kocka koja je reprezentacija objekta koji će se stvoriti. Sva svojstva hologramskog prikaza objekta preslikavaju se na stvarni objekt, osim tekstura koje se mijenjaju u ovisnosti odabrane boje. Primicanjem i odmicanjem ruku, kocka se skalira jednoliko u sve tri njene osi. Otpuštanjem geste štipanja na bilo kojoj ruci stvara kocku.

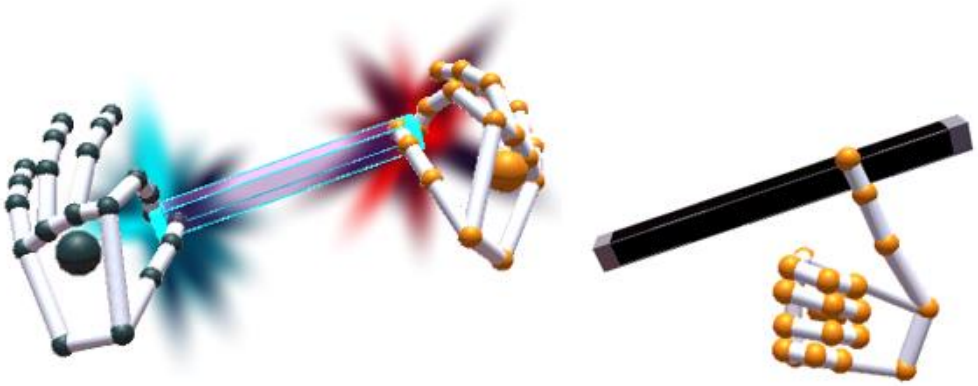


Slika 17 – Stvaranje kocke

### 5.1.2 Stvaranje štapa

Sa strane korisničkih kontrola, stvaranje štapa je potpuno jednako kao i prije naveden proces stvaranja kocke. Ova modifikacija kvadra pri modeliranju može reprezentirati stupove za potporu terase na zgradi, deblo drveta, element za izgradnju ograde…

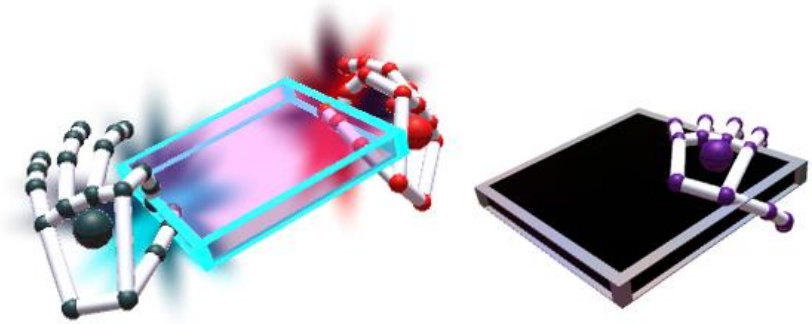
Pri postavljanju predmeta u prostor, njegova visina i dubina su postavljene na konstantu, dok je jedino širina proporcionalna udaljenosti korisnikovih ruku.



Slika 18 – Stvaranje štapa

### 5.1.3 Stvaranje ploha

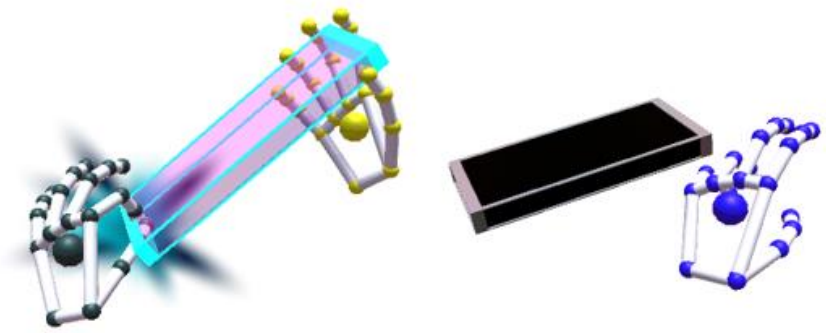
Zamisao plohe kao objekta u ovoj aplikaciji jest da reprezentira teren zgrade ili površinu na kojoj se model izgrađuje ukoliko korisnik ne želi započeti gradnju na predviđenom bijelom podu u aplikaciji. Korisno je da se odabere neka svjetlija boja kako bi pomoću sjena se dobila percepcija trodimenzionalnog prostora. Pri postavljanju predmeta, samo je visina kvadra jednaka konstanti, dok su preostale dvije osi proporcionalne udaljenosti korisnikovih ruku



Slika 19 – Stvaranje plohe

### 5.1.4 Stvaranje daske

U slučaju stvaranja daske moguće je skaliranje kvadra u dvije dimenzije. Za razliku od ostalih predmeta, inicijalizacija stvaranja započinje u trenutku kada se na lijevoj ruci napravi gesta štipanja. Pojavljuje se hologramski objekt čija je širina jednaka udaljenosti korisnikovih ruku, dok je dubina jednaka udaljenosti palca i kažiprsta. Ovaj objekt dodan je kako bi se demonstrirala veća interakcija sa virtualnim prostorom jer korisnikova desna ruka sada određuje nagib i dubinu predmeta. Otpuštanjem geste na lijevoj ruci stvara se objekt.



Slika 20 – Stvaranje daske

## 5.2 Predmeti druge skupine (Potpuno ili djelomično zaobljeni predmeti)

Ova skupina predmeta sastoji se od zaobljenih predmeta koji mogu koristit pri stvaranju oblaka, krošnji drveća i slično. Kako korisnik ima na izbor cijeli spektar boja, uz malo kreativnosti ovih nekoliko predmeta može se iskoristiti za modeliranje mnogo stvari

Slika 21 – Objekti druge skupine

### 5.2.1 Stvaranje kugle

Stvaranje kugle slično je postavljanju kocke u prostoru. Zapravo, nakon malih modifikacija skripte, podešena je  tako da može stvoriti bilo koji predmet koji joj se postavi kao zadani te ga skalirati u sve tri osi. Također je tu i prozirna hologramska kugla koja korisniku prikazuje kako će konačan predmet izgledati.

Slika 22 – Stvaranje kugle

Kao što je vidljivo u daljnjem primjeru, kugle su korištene pri modeliranju dijelova oblaka, Sunca te dijelova krošnje drveta.

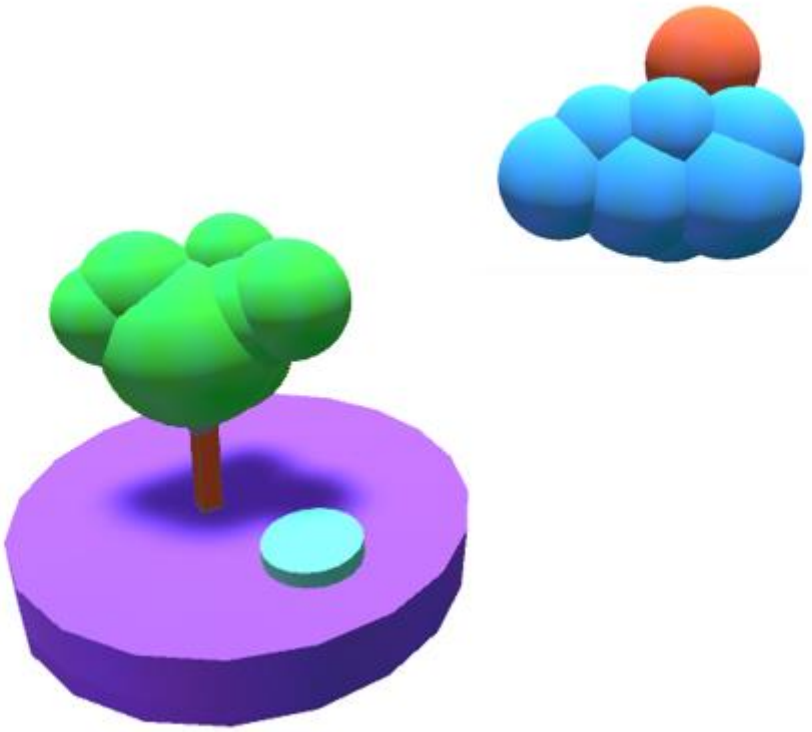
### 5.2.2 Okrugla ploča

Za dizajn ploče također je korištena unaprijeđena skripta odgovorna za stvaranje normalne, pravokutne ploče tako da može u prostor postaviti ovu, zaobljenu ploču. Način stvaranja predmeta je u potpunosti jednak.

Slika 23 – Stvaranje okrugle ploče

Ploču je najbolje koristiti kao podij za izgradnju drugih modela, tako da ona ima reprezentaciju tla. U dolje navedenom primjeru sam ju također koristio kao vodenu površinu.

# 6. Primjeri modela napravljenog aplikacijom



# 7. Korištenje skripti u Unity3D razvojnom okruženju

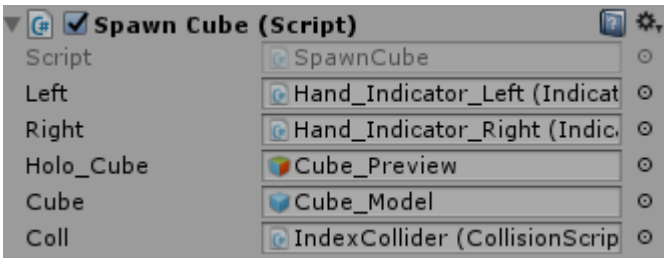
## 7.1 Odabir trenutnog alata

Kao primjer komunikacije nekoliko skripti prikazana je skripta odgovornu za trenutno odabran alat.

Važno za napomenuti je da se skripta ne izvodi ukoliko je predmet za kojeg je ona vezana neaktivan. Koristeći to svojstvo stvorena je hijerarhija praynih objekata u sceni čija je uloga da budu nositelji skripti. Jedina iznimka toga jesu skripte koje u sebi očekuju okidač drugoga predmeta.

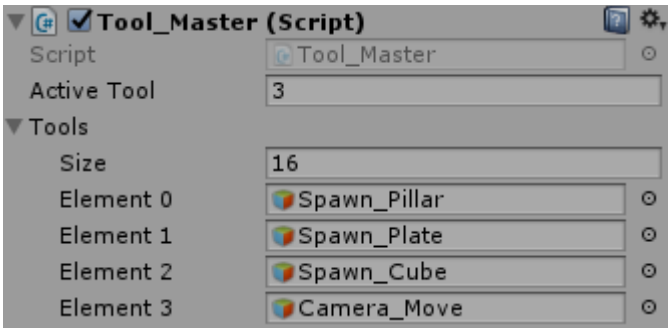

Slika 24 – Hijerarhija nositelja skripti

Iz slika 22 i 23 je vidljivo da u trenutku kada je aktivan objekt imena „Spawn_Cube“, aktivna je i skripta vezana za njega.



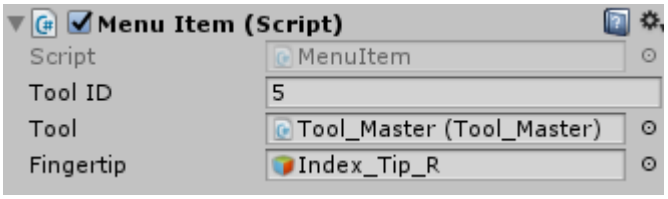

Slika 25 – Javne varijable skripte

Objekt imena „Tool_Master“ uvijek je aktivan i odgovoran da se može koristit samo jedan alat u određenome trenutku. On se sastoji od pokazivača na trenutno aktivan alat te od niza referenci na objekte kojima su dodijeljene skripte.




Slika 26 – Prikaz niza alatnih skripti

Javnu varijabla imena „Active Tool“ mijenjaju skripte dodijeljene elementima glavnog izbornika.




Slika 27 – Univerzalna skripta kao okidač u prostoru

Univerzalna skripta imena „Menu Item“ podesiva je na način da se u liniju „Tool ID“ unese identifikacijski broj alata u nizu. Definiranjem reference na skriptu aktivnog alata i metode koja se izvršava kada Sudarač (eng. Collider) izbornika očita presijecanje sa sudaračem prsta.

```
public Tool_Master Tool;

…

private void OnTriggerEnter(Collider other)
{
	if (other.gameObject == Fingertip) Tool.ActiveTool = ToolID;
}
```

## 7.2 Stvaranje objekta

Kao primjer stvaranja objekta, koristit ću običnu kocku. Kako bi skripta bila što preglednija, u njoj se ne nalaze metode odgovorne za određivanje pozicija ruku, već se to odvija u vanjskim procesima.

Prvo je potrebno odrediti gdje će se kocka pojaviti. Po priloženoj implementaciji, njeno središte bit će na polovini dužine između dvije geste štipanja. Kako bi pristupili koordinatama tih gesti, potrebno je referencirati skripte koje ih neprestano prate. Također kao varijable koriste se skripte kolizije s paletom boje te reference na prozirnu i punu kocku.

```
public IndicatorTool Left, Right;
public GameObject Holo_Cube, Cube;
public CollisionScript Coll;

private bool Spawn_Ready = false;
private float scale;
```

Sve promjene nad referenciranim objektima događaju se unutar Osvježi (end. Update) metode koja se poziva jedanput po okviru. Kada se očita da su obije ruke u gesti štipanja, prikazat će se hologramska kocka koju je moguće pomicati po prostoru, te će se zastavica za stvaranje objekta postaviti u logičku istinu. U ovome stanju postavlja se transformacije hologramskog objekta koje će se kasnije preslikati na onaj stvoreni.

```
…
if (Left.Pinched && Right.Pinched)
 {
       Holo_Cube.SetActive(true);
       Holo_Cube.transform.position = (Right.Pinch_Location +
       Left.Pinch_Location) / 2;
       scale = Vector3.Distance(Right.Pinch_Location, Left.Pinch_Location);
       Holo_Cube.transform.localScale = new Vector3(scale, scale, scale);
       Holo_Cube.transform.rotation =
       Quaternion.LookRotation(Right.Pinch_Location - Left.Pinch_Location,
       Vector3.up);
       Spawn_Ready = true;
 }

…
```

Nakon otpuštanja jedne od gesti (tj. Prestanka ispunjavanja uvjeta funkcije if), objekt se stvara na toj lokaciji.

```
else
 {
     Holo_Cube.SetActive(false);
     if (Spawn_Ready)
     {
             GameObject Clone = GameObject.Instantiate(Cube);
             Clone.transform.position = (Right.Pinch_Location +
             Left.Pinch_Location) / 2;
```

```
                Clone.transform.rotation =
                Quaternion.LookRotation(Right.Pinch_Location -
                Left.Pinch_Location, Vector3.up);
                Clone.transform.localScale = Holo_Cube.transform.localScale =
                new Vector3(scale, scale, scale);
                Clone.GetComponent<Renderer>().material.color =
                Coll.selectedColor;
                Spawn_Ready = false;
        }
    }
```

Zastavica imena „Spawn_Ready“ osigurava da odvoji hologramsku od stvarne kocke te osigura da se uvijek stvori samo jedna stvarna kocka.

## 7.3 Bacanje zrake i odabir boje s teksture

Postupak odabira boja implementiran je bacanjem zrake na plohu s teksturom u trenutku primicanja sudarača plohi s bojama. Zraka koja se baca izlazi iz kažiprsta lijeve ruke te ide u smjeru prsta. Algoritam provjerava jeli predmet kojeg je sudarač registrirao zapravo ploča s bojama, te ukoliko je, nastavlja s ključnim dijelom algoritma.

```
if (other.gameObject == colorPlate)
 {
      RaycastHit hit;
      if (!Physics.Raycast(this.transform.position, smjer.position -
      this.transform.position, out hit))
        return;

      Renderer rend = hit.transform.GetComponent<Renderer>();
      MeshCollider meshCollider = hit.collider as MeshCollider;

      Texture2D tex = rend.material.mainTexture as Texture2D;
      Vector2 pixelUV = hit.textureCoord;
      pixelUV.x *= tex.width;
      pixelUV.y *= tex.height;

      selectedColor = tex.GetPixel((int)pixelUV.x, (int)pixelUV.y);
      colorWheel.material.color = selectedColor;
 }
```

Zraka pogođenom predmetu uzima koordinate teksture te ih pretvara u RGB (eng. Red Green Blue) vrijednosti koju prosljeđuje u javnu varijablu kojoj pristupaju skripte kojima je potrebna trenutno odabrana boja.

# 8. Literatura

[1] Wikipedia, Leap Motion

https://en.wikipedia.org/wiki/Leap_Motion

[2] Wikipedia, Gesture

https://en.wikipedia.org/wiki/Gesture

[3] developer.leapmotion.com, Documentation

https://developer.leapmotion.com/documentation/csharp/index.html

[4] Unity, User Manual (5.6)

https://docs.unity3d.com/Manual/index.html

[5] Leapmotion, VR Setup

https://developer.leapmotion.com/vr-setup/

# 9. Zaključak

Ideja ovog završnog rada bila je istražiti koje sve mogućnosti pruža Leap Motion kontrolor zajedno sa Unity3D razvojnom okruženju te osmisliti aplikaciju kojom se koristeći prirodno korisničko sučelje može u potpunosti manipulirati njome.

U ovome radu oslonio sam se više na podjelu i kontrolu aktivnih skripti kako bi se što manje gesti moralo naučiti korisnika, iako je bilo moguće u potpunosti izbaciti izbornik s alatima na lijevoj ruci te odabir alata prebaciti na predefinirani položaj ruke kao zasebna gesta.

Proces razvoja aplikacije bio je modularan uz činjenicu da kroz paralelno priučavanje Unity3D dokumentacije i popisa referenci Leap Motion paketa, svaka nova skripta mogla je ostvariti bolje rezultate nego ona prijašnja.

Nažalost, Leap Motion kontrolor nije dovoljno rasprostranjen kako bi se za njega mogle razvijati interaktivne igrice namijenjene široj populaciji te je on najbolje primjenjiv u demonstracijskim aplikacijama i u svrhu istraživanja virtualnog okruženja.

Sve u svemu, ovaj rad je znatno razvio moje programerske, kao i vizualne vještine i spreman sam nadograditi ga sa dodatnim, zahtjevnijim funkcionalnostima u nastavku studija.

## Sažetak

Naslov : Modeliranje objekata u virtualnom prostoru uporabom prirodnog korisničkog sučelja

U ovome radu opisuje se postupak korištenja Leap Motion kontrolora zajedno s Unity3D okruženju. Postupak razvoja opisan je modularno na način da se čitatelja uvede u proces dohvaćanja korisnih informacija iz kontrolora. Rad također opisuje način korištenja tih informacija uz primjere kako bi se one mogle koristiti na drugi način. Ključni isječci koda opisuju točan način pisanja skripti odgovornih za ispravan rad aplikacije. Naposljetku dani su napuci o stvaranju hijerarhije objekata i skripti kako bi postupak razvoja aplikacije bio što manje zbunjujući. Kao demonstracija aplikacije, opisani su svi alati koje korisnik može koristiti i način na koji to čini.

**Ključne riječi** : Leap Motion, Unity 3D, Virtualna Stvarnost, Prirodno Korisničko Sučelje, Modeliranje, C#, Praćenje Pokreta

## Abstract

Title : Objects Modelling in Virtual Space by Natural User Interface

This thesis describes the procedure of using the Leap Motion controller together with the Unity3D environment. The development process is described in a modular way by introducing the reader into the process of retrieving useful information from the controller. The thesis also describes how this information was used with examples so that they can be used in other ways. The key snippets of the code show the correct way for writing scripts used in this application. Finally, there are tips on creating hierarchies of objects and scripts to keep the development process understandable. As a demonstration, all tools that a user can use and their functionalities are described aswell.

**Keywords** : Leap Motion, Unity 3D, Virtual Reality, Natural User Interface, Modeling, C#, Motion Tracking