SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5363

MODELIRANJE OBJEKATA UPOTREBOM DUBINSKE KAMERE

Nikola Nađ

Zagreb, lipanj 2018.

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

Zagreb, 9. ožujka 2018.

ZAVRŠNI ZADATAK br. 5363

Pristupnik: **Nikola Nađ (0036491265)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Modeliranje objekata upotrebom dubinske kamere**

Opis zadatka:

Proučiti mogućnosti dubinske kamere kao što je na primjer Leapmotion. Proučiti mogućnosti modeliranja trodimenzionalnih objekata u prostoru upotrebom ovakvih uređaja. Proučiti mogućnosti povezivanja navedenog uređaja s 3D naočalama. Razraditi aplikaciju koja će omogućiti modeliranje jednostavnih objekata korištenjem kombinacije dubinske kamere i 3D naočala. Na različitim primjerima prikazati ostvarene rezultate. Načiniti ocjenu rezultata i implementiranih postupaka.
Izraditi odgovarajući programski proizvod. Koristiti grafički pogon Unity. 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: 16. ožujka 2018.
Rok za predaju rada: 15. lipnja 2018.

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

Trodimenzionalni modeli jedan su od ključnih pojmova u računalnoj grafici, te je njihova primjena široko rasprostranjena – od računalnih igara, preko animiranih filmova, do raznih realističnih simulacija. Danas postoji velik broj programa za modeliranje 3D objekata poput Blendera, 3D's Maxa, Maye, itd., a u zadnje vrijeme počeli su se pojavljivati i programi za modeliranje u virtualnoj stvarnosti, kao što su Oculus Medium, Google Tilt Brush i još nekoliko drugih. Tema ovog završnog rada upravo je kombiniranje programa za modeliranje 3D objekata s virtualnom stvarnošću uporabom Oculus Rifta i Leap Motiona. Najprije će biti opisana korištena oprema. U sljedećem poglavlju bit će opisan program za modeliranje razvijen u sklopu ovog završnog rada. Zatim će biti riječi o ključnim dijelovima implementacije dotičnog programa, a u posljednjem poglavlju bit će navedena ograničenja ovakvog pristupa modeliranju.

Slika 1 - Primjer uporabe 3D modela u animiranom filmu Cars

2. Korištena oprema

U ovom radu proučena je mogućnost kombiniranja 3D naočala Oculus Rift te uređaja s dubinskom kamerom Leap Motion. U nastavku poglavlja bit će opisani navedeni uređaji.

2.1. Uređaj za prikazivanje slike Oculus Rift

Oculus Rift jedan je od uređaja za prikazivanje virtualne stvarnosti (engl. Virtual Reality, VR), uz HTC Vive, Samsung Gear VR, Playstation VR... Virtualna stvarnost je računalom generiran virtualni svijet (može biti simulacija realnog ili u potpunosti fiktivan) koji simulira korisnikov doživljaj preko njegovih osjetila i percepcije uporabom posebnih uređaja i senzora.

Slika 2- Uređaj Oculus Rift Development Kit 2

Tehnika koju koriste svi VR uređaji za prikaz treće dimenzije pomoću dvodimenzionalnih zaslona naziva se stereoskopija, a ta tehnika poznata je još od 19. stoljeća. 1838. godine Sir Charles Wheatstone konstruirao je prvi stereoskop, uređaj koji prikazuje dvije dvodimenzionalne fotografije – obje fotografije prikazuju istu scenu, međutim perspektiva je malo pomaknuta na jednoj u odnosu na drugu. Svaka slika namijenjena je jednom oku, a pomaknuta perspektiva, koja simulira razmak između očiju, stvara iluziju dubine.

Slika 3 - Jednostavni stereoskop [1]

2.1.1. Građa i princip rada

Oculus Rift je tzv. Head Mounted Display, odnosno zaslon koji se montira na glavu. Za prikaz slike koriste se dva malena stereoskopska OLED zaslona (po jedan za svako oko) čime se dobiva prikaz 3D prostora, a u kombinaciji s lećama postignuto je široko vidno polje od 110 stupnjeva. Svaki zaslon ima rezoluciju od 1080x1200 piksela te ima mogućnost prikazivanja do 90 sličica u sekundi (engl. frames per second, FPS). Iako je moguće, nije preporučljivo prikazivanje s manje od 90 FPS-a jer isprekidanost u prikazu uzrokovana premalenom frekvencijom osvježavanja slike može izazvati mučninu. OLED (engl. Organic Light Emitting Diode) zasloni imaju sloj organskog poluvodičkog materijala sa svojstvom elektro-luminiscencije – materijal emitira svjetlost pod utjecajem električne struje. Prednost OLED zaslona nad LCD zaslonima je činjenica da ne koriste pozadinsko osvjetljenje, što omogućava prikaz tamnijih crnih nijansi. Također ima manje vrijeme kašnjenja, postiže veći omjer kontrasta i ima veći kut gledanja, pa boje ostaju korektne čak i pri kutovima gledanja bliskim 90 stupnjeva od normale zaslona.

Slika 4 – Sastavni dijelovi uređaja Oculus Rift [2]

Zasloni i leće montiraju se na glavu zajedno s ugrađenim senzorima poput žiroskopa, magnetomjera i akcelerometra, i slušalicama[1]. Pokreti glave registriraju se statičnim infracrvenim senzorom (obično se stavi na stol ispred korisnika) koji detektira infracrvenu svjetlost emitiranu iz LED dioda montiranih na glavi. Senzor registrira pokrete unutar određenog 3D područja, a ono se može dodatno proširiti uporabom više infracrvenih senzora. Registrirani pokreti uzrokuju promjenu virtualnog položaja, što se prikazuje na zaslonu i to uz minimalno kašnjenje od svega 2 milisekunde. Senzori također pružaju visoku preciznost reda veličine milimetar.

[1] Slušalice su ugrađene u samo komercijalnu verziju Oculus Rifta, dok razvojne verzije (Development Kit 1 i 2) nemaju slušalice.

2.2. Senzor dubine Leap Motion

Leap Motion controller periferni je USB uređaj koji korisnicima pruža vrlo interesantno i intuitivno sučelje prema računalu – geste ruku. Uređaj se može postaviti na stol ispred korisnika, ili se može montirati na VR *headset*, poput Oculus Rifta. Leap Motion pruža veliko proširenje virtualne stvarnosti, omogućavajući korisniku da doslovno vlastitim rukama interagira s računalom.

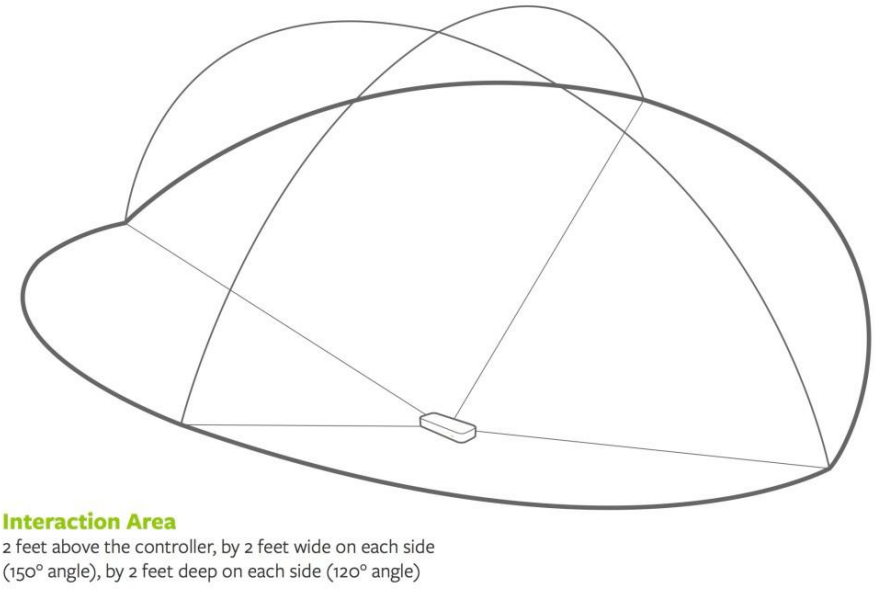
Slika 5 - Uređaj Leap Motion [3]

2.2.1 Građa i princip rada

U svom radu Leap Motion koristi 3 infracrvene svjetleće diode (engl. infrared LED) kojima osvjetljava prostor, te 2 monokromatske infracrvene kamere koje detektiraju svjetlost valne duljine 850nm, dakle izvan vidljivog spektra svjetlosti. Podatke koje kamere snime obrađuje softver Leap Motion Service. Pri obradi se nastoji kompenzirati pozadinsko infracrveno osvjetljenje, poput sunčevog svjetla. Udaljenost detektiranog objekta od uređaja računa se na temelju 2 slike pribavljenih od 2 kamere. Leap Motion pruža sub-milimetarsku preciznost, te ima mogućnost praćenja objekata do 200 slika u sekundi. Vrlo je malenih dimenzija, samo 80mm x 30mm x 11.25mm i teži 45g, što ga čini vrlo portabilnim.

2.2.2. Interaktivni prostor

Leap Motion može detektirati objekte na udaljenosti do 80cm, i to pod kutem od 150 stupnjeva u širinu, odnosno 120 stupnjeva u dubinu, čime je definiran interaktivni prostor uređaja u obliku invertirane piramide. Za prikaz koordinata koristi se desni koordinatni sustav čije se ishodište nalazi na središtu prednje strane uređaja. Y os sustava definirana je u smjeru okomitom na uređaj, X os definirana je u smjeru paralelnom s uređajem u širinu (vidno polje od 150 stupnjeva po X osi), dok je Z os definirana u smjeru dubine (120 stupnjeva). Pritom su mjerne jedinice po sve tri koordinate milimetri.



Slika 6 - Vizualizacija interaktivnog prostora Leap Motiona [3]

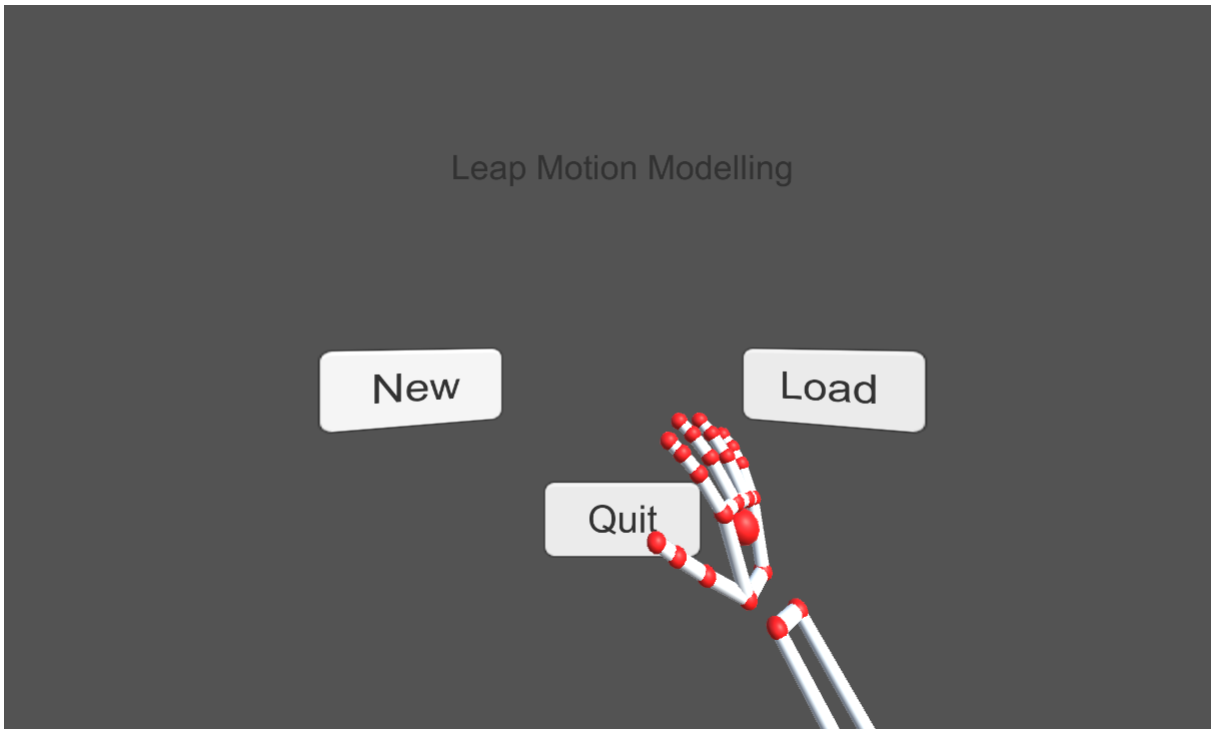
2.2.3. Ograničenja detektiranja objekata

S obzirom da Leap Motion ima samo dvije kamere koje su relativno blizu jedna drugoj, može se reći da uređaj detektira objekte iz samo jedne perspektive, međutim ponekad je potrebno pogledati objekt iz više različitih perspektiva kako bi se u potpunosti mogao odrediti njegov položaj. Uzrok nepravilnog detektiranja objekata koji se u ovakvom pristupu javlja je samo-zaklanjanje (engl. self-occlusion), odnosno situacija u kojoj kamere ne mogu snimiti sve važne dijelove objekta jer su

oni zaklonjeni samim objektom. Tada softver za obradu slika može jedino aproksimirati gdje se ti ključni dijelovi objekta nalaze, što nerijetko rezultira pogrešnom aproksimacijom. Primjerice, ako se ruka okrene na način da kamere ne mogu „vidjeti“ sve prste, njihov položaj će se morati aproksimirati. Zbog ovog ograničenja preporučljivo je uvijek okrenuti ruke tako da su kamerama vidljivi svi prsti ruku, kako bi njihova pozicija bila što preciznije određena.

3. Razvijeni program

U sklopu ovog završnog rada implementiran je program za modeliranje 3D objekata uporabom Oculus Rifta i Leap Motiona u razvojnoj okolini Unity. Ideja programa je omogućiti korisnicima intuitivan pristup modeliranju da vlastitim rukama oblikuju objekte u trodimenzionalnom prostoru. Pri pokretanju programa otvara se glavni izbornik, gdje korisnik može odabrati kreiranje novog objekta (pritiskom na gumb „New“), može odabrati učitavanje već postojeće .obj datoteke (gumb „Load“) ili može prekinuti rad programa (gumb „Quit“).

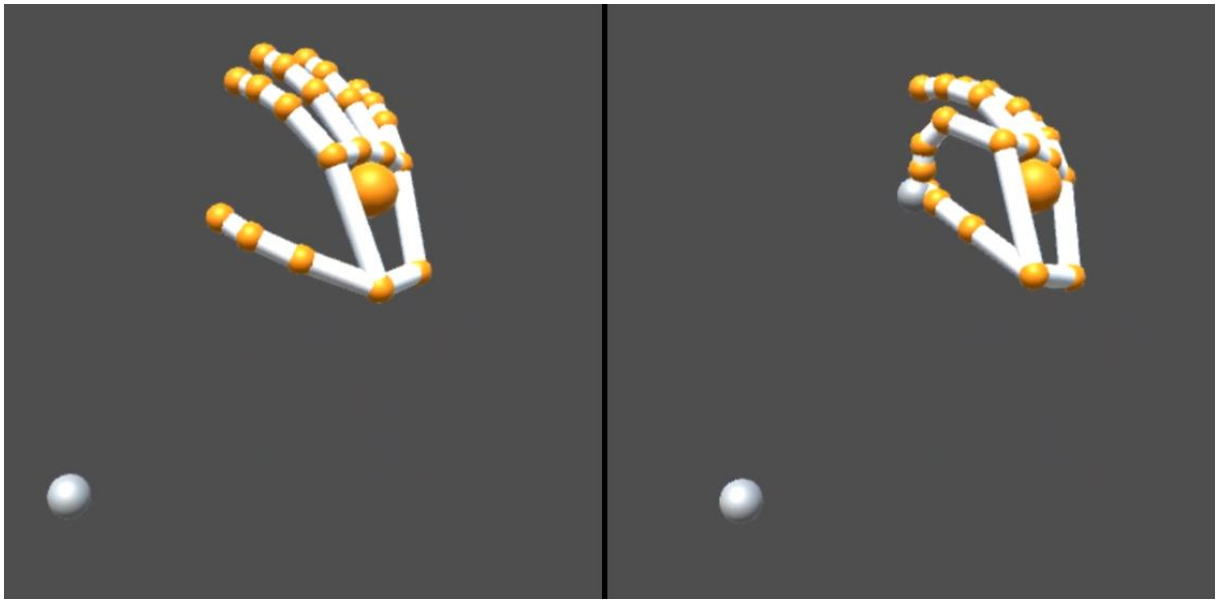


Slika 7 - Glavni izbornik

Nakon učitavanja datoteke (ili odabira kreiranja novog objekta) počinje modeliranje objekta. Pritom na raspolaganju korisnik ima sljedeće akcije:

- **Stvaranje vrhova objekta**

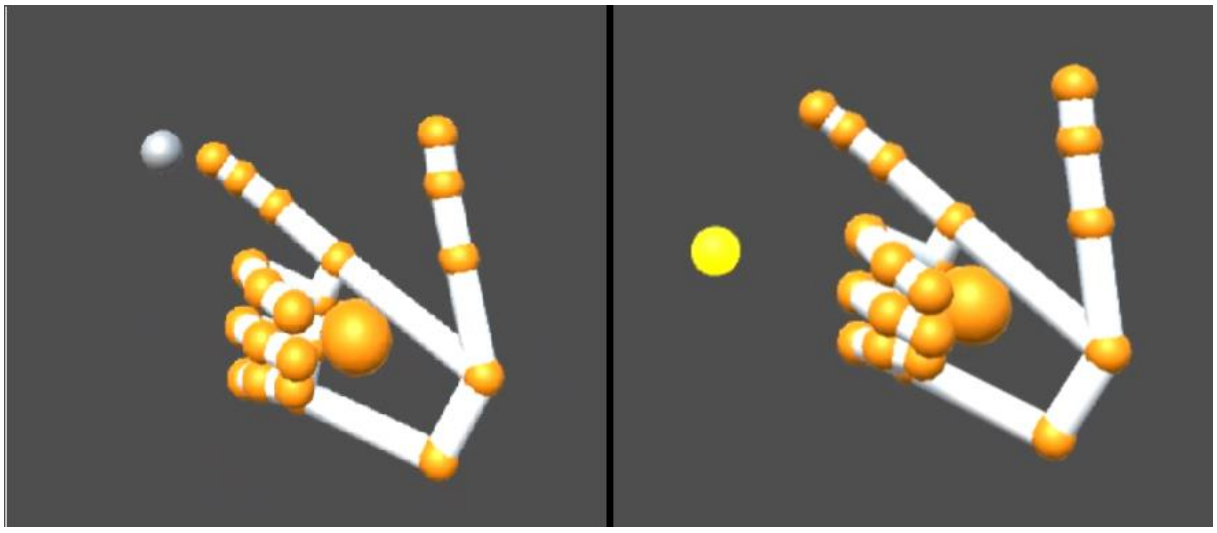
Približavanjem kažiprsta i palca bilo koje ruke stvara se vrh objekta (engl. Vertex). Sve dok su prsti na dovoljno malenoj udaljenosti, vrh će pratiti položaj ruke, a udaljavanjem prstiju vrh prestaje pratiti ruku te se fiksira u prostoru.



Slika 8 - Stanje prije (lijevo) i nakon (desno) detektiranja geste za stvaranje vrhova

- **Selekcija vrhova objekta**

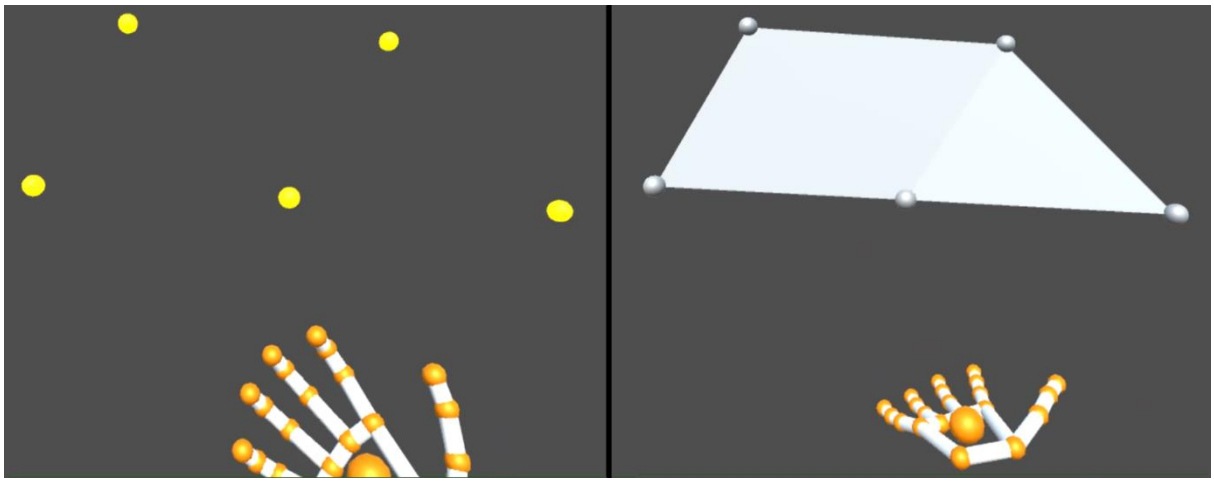
Gestom pokazivanja kažiprstom, odnosno ispruživanjem kažiprsta i sklapanjem ostalih prstiju ruke omogućava se selekcija već stvorenih vrhova objekta. Pritom desna ruka služi za selekciju vrhova, dok lijeva ruka služi za deselekciju. Selektirani vrhovi se označavaju svjetlećim žućkasto-narančastim materijalom, dok su neselektirani vrhovi prikazani bijelim materijalom.



Slika 9 - Stanje prije (lijevo) i nakon (desno) detektiranja geste za selekciju vrhova

- **Kreiranje oplošja objekta trokutima**

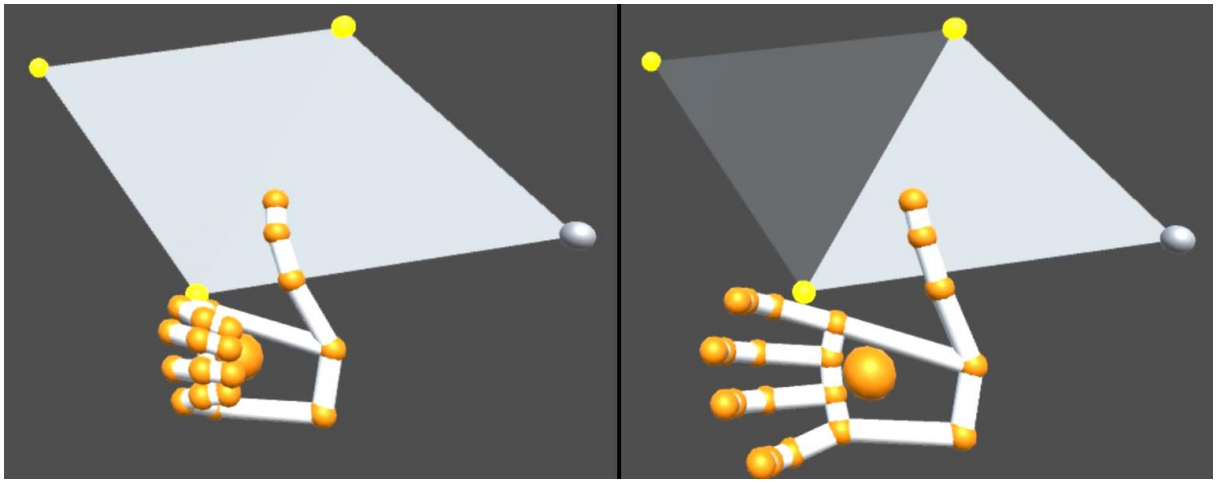
Ispruživanjem svih prstiju ruke te usmjeravanjem normale dlana prema gore (u lokalnom koordinatnom sustavu kamere) stvaraju se trokuti, koji predstavljaju oplošje objekta. Trokuti se stvaraju samo od selektiranih vrhova, i to nalik na OpenGL-ov primitiv TRIANGLE_STRIP. Ovaj način kreiranja trokuta uzima u obzir redoslijed selekcije vrhova, gdje se za n vrhova stvaraju n-2 trokuta, na način da se spajaju vrhovi s indeksima (1,2,3), zatim (2,3,4), (3,4,5) … (n-3, n-2, n-1), (n-2, n-1, n). Ukoliko je broj selektiranih vrhova manji od 3, ovom gestom se ništa neće dogoditi. Prednje strane trokuta prikazane su bijelim neprozirnim materijalom, dok su stražnje strane trokuta prikazane tamno sivim materijalom.



Slika 10 - Stanje prije (lijevo) i nakon (desno) detektiranja geste za stvaranje trokuta

- **Invertiranje normale trokuta**

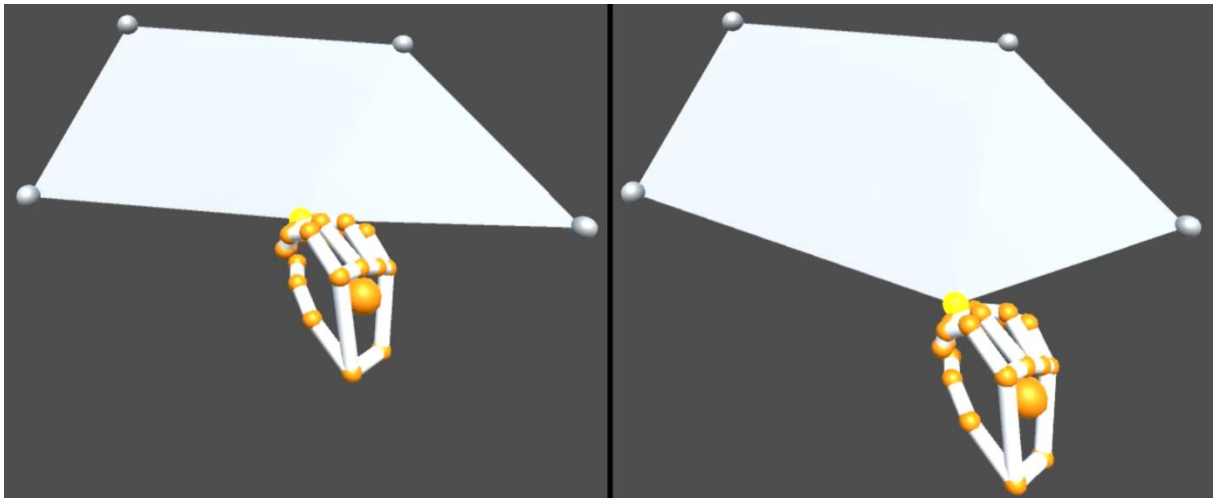
Gestom „thumbs up“ bilo koje ruke invertiraju se normale selektiranih trokuta (to su trokuti čija su sva 3 vrha selektirana). Na taj način dotad prednja strana trokuta postaje stražnja, dok stražnja strana postaje prednja. Na slici (Slika 11) može se uočiti kako je lijevi trokut nakon detekcije geste postao tamniji – tamni materijal predstavlja stražnju stranu, dok svijetli predstavlja prednju.



Slika 11 - Stanje prije (lijevo) i nakon (desno) detektiranja geste za invertiranje normala

- **Translacija selektiranih vrhova**

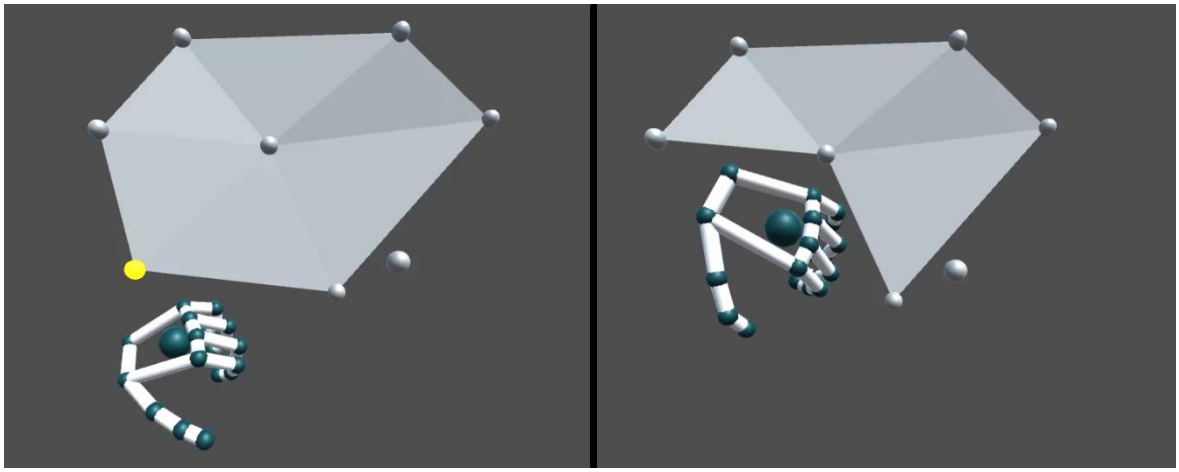
Sklapanjem šake desne ruke pokreće se akcija translacije selektiranih vrhova. Sve dok je šaka sklopljena, vrhovi će se pomicati zajedno s rukom. Intuitivno se ova gesta može protumačiti kao da je korisnik šakom primio vrhove te ih pomiče u prostoru. Pritom se pomiču/deformiraju i trokuti koji su sastavljeni od selektiranih vrhova.



Slika 12 - Stanje prije (lijevo) i nakon (desno) detekcije geste za translaciju selektiranih vrhova

- **Brisanje selektiranih vrhova**

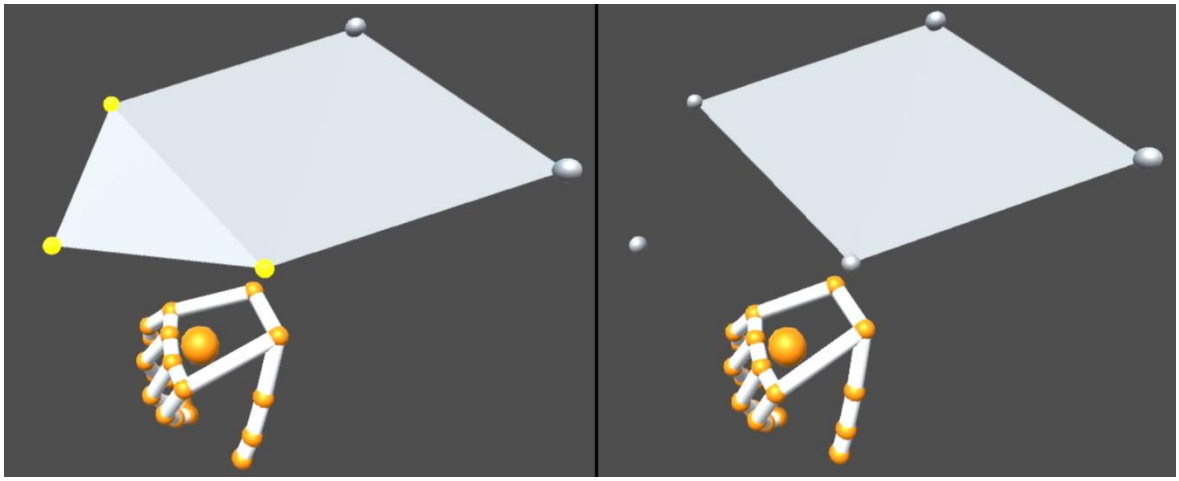
Gestom „thumbs down“ lijeve ruke brišu se trenutno selektirani vrhovi. Pritom se brišu i svi trokuti koji sadrže barem jedan od selektiranih vrhova.



Slika 13 - Stanje prije (lijevo) i nakon (desno) detektiranja geste za brisanje vrhova

- **Brisanje selektiranih trokuta**

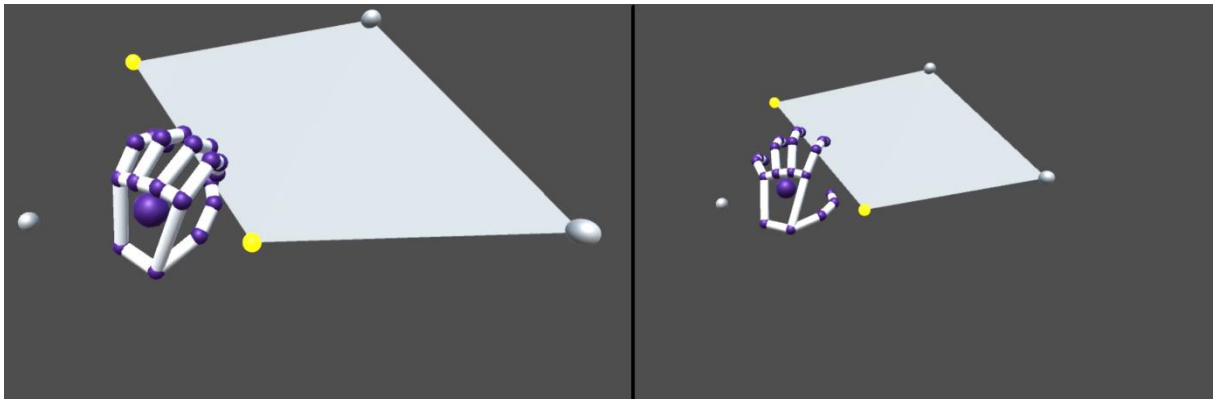
Gestom „*thumbs down*“ desne ruke brišu se svi trokuti kojima su sva tri vrha selektirana. Na slici (Slika 13) može se uočiti kako je obrisan samo jedan trokut, čija su sva tri vrha selektirana, dok njegovi vrhovi nisu obrisani.



Slika 14 - Stanje prije (lijevo) i nakon (desno) detektiranja geste za brisanje trokuta

- **Pomicanje kamere**

Sklapanjem šake lijeve ruke pokreće se akcija translacije kamere u prostoru. Intuitivno se ova akcija može opisati kao da je korisnik primio točku u prostoru, te se pokretima ruke približava, odnosno udaljava od dotične točke. Ovom akcijom omogućava se korisniku da pomakne pogled kako bi pogledao objekt iz drugačije perspektive, da modelira proizvoljno velike objekte i slično.



Slika 15 - Stanje prije (lijevo) i nakon (desno) detektiranja geste za pomicanje kamere

U bilo kojem trenutku tijekom modeliranja pritiskom na tipku escape na tipkovnici prikazat će se izbornik u kojem korisnik može odabrati spremanje dosad napravljenog modela (pritiskom na gumb „Save). Otvorit će se datotečni preglednik u kojem korisnik može specificirati ime i mjesto spremanja objekta. Objekt se zapisuje u .obj formatu, pri čemu se zapisuju samo prednji poligoni. Stražnji poligoni se u programu prikazuju isključivo kao olakšanje prilikom modeliranja. Druga opcija koju korisnik ima na raspolaganju u otvorenom izborniku je gumb „Quit“ te se pritiskom na taj gumb korisnik vraća na glavni izbornik opisan na početku ovog poglavlja.

4. Implementacija programa

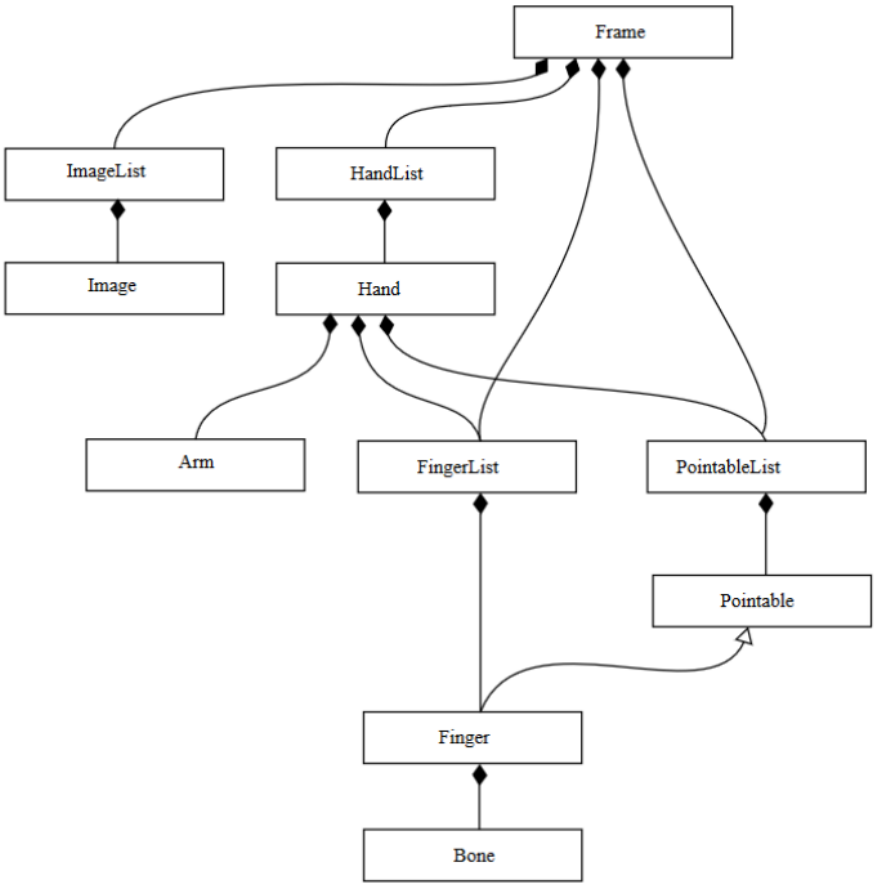
U ovom poglavlju bit će opisani bitni dijelovi implementacije programa. Najprije će biti riječi o integraciji uređaja Leap Motion s razvojnom okolinom Unity. Zatim će biti objašnjena implementacija gesti. Nakon toga slijedi opis podatkovne strukture mreže objekta i manipuliranje tim podacima. Na posljetku će biti riječi o učitavanju i spremanju mreže objekta u .obj formatu.

4.1 Leap Motion i razvojna okolina Unity

Program je u cijelosti implementiran u razvojnoj okolini Unity, a sve skripte pisane su u programskom jeziku C#. Leap Motion nudi aplikacijsko programsko sučelje (engl. API – Application Programming Interface) za Unity, koje omogućava detekciju i pozicioniranje ruku u virtualnoj sceni unutar Unity-ja. Također je moguće saznati razne podatke o položaju ruku i prstiju, primjerice brzina i smjer pokreta dlana, položaj vrha kažiprsta i slično.

4.1.1. Programsko sučelje Leap Motion

Leap Motion API organiziran je u nekoliko razreda, a pristupna mu je točka vršni razred Controller. Preko Controller-a moguće je dobiti okvire (engl. Frame), koji sadrže sve podatke o praćenim objektima. Preko Frame-a moguće je dobiti listu detektiranih ruku, listu usmjerenih objekata (engl. pointable), kao što su prsti, te čak listu neobrađenih slika dviju kamera na temelju kojih su detektirani objekti i određen njihov položaj. Također je moguće dobiti i listu detektiranih alata i gesti, međutim ti razredi su *deprecated* te se ne preporuča njihovo korištenje.



Slika 16 - Pojednostavljeni dijagram razreda Leap Motion API-ja

Za potrebe implementacije programa korišteni su sljedeći razredi:

Controller – korišten je za dobivanje trenutnog okvira.

Frame – korišten je kako bi se dobila lista detektiranih ruku u određenom trenutku.

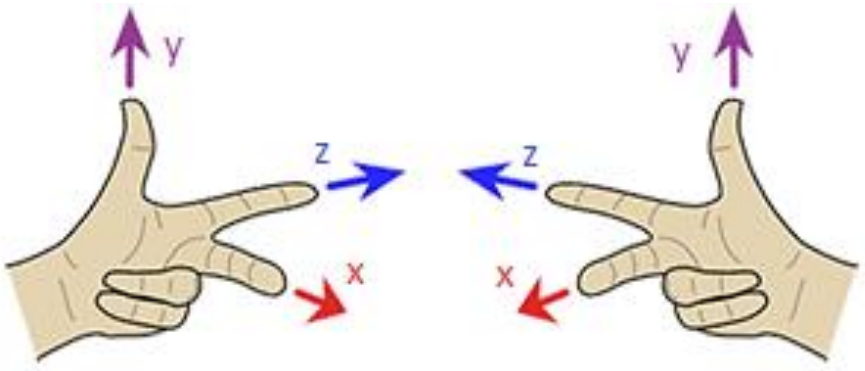
Hand – ovaj razred predstavlja dlan ruke te nudi korisne informacije, primjerice je li ruka lijeva ili desna, u kojem smjeru gleda normala dlana, kojom brzinom se dlan kreće u odnosu na trenutni i prošli okvir itd. Također, ovaj razred ima listu prstiju koji su dio te ruke.

Finger – predstavlja prst ruke, te implementira sučelje Pointable. Sadrži informacije kao što su pozicija vrha prsta, duljina i širina prsta, brzina kretnje prsta, vektor u smjeru kojega prst pokazuje, informacija o tome je li prst ispružen itd.

Uporabom gore navedenih razreda implementirane su geste kojima se u programu modeliraju objekti. O implementaciji dotičnih gesti bit će riječi u sljedećem poglavlju.

4.1.2. Koordinatni sustavi u Unity-ju i Leap Motionu

Koordinatni sustav koji koristi Leap Motion je desni koordinatni sustav te se koriste milimetri kao osnovna mjerna jedinica. Koordinatni sustav u Unity-ju je lijevi, dok je osnovna mjerna jedinica metar. Razlika između lijevog i desnog koordinatnog sustava može se vidjeti na slici (Slika 17). Očito je da ovi koordinatni sustavi nisu usklađeni, te je potrebno primijeniti određenu transformaciju kako bi se točka iz jednog sustava preslikala u drugi. U tu svrhu napisan je pomoćni razred LeapUtil koji nudi statičku metodu Vector3 LeapVectorToScaledVector3(Vector v), kojom se za vektor iz sustava Leap Motiona dobije ekvivalentni vektor u koordinatnom sustavu Unity-ja. Transformacija koju provodi metoda je prilično jednostavna – potrebno je podijeliti sve koordinate vektora s 1000 (kako bi se iz milimetara dobili metri) te je potrebno obrnuti predznak z koordinate kako bi se iz desnog sustava dobila koordinata u lijevom.



Slika 17 - Usporedba lijevog koordinatnog sustava (lijevo) i desnog koordinatnog sustava (desno)

4.2. Geste

Leap Motion API nudi 4 osnovne geste: CircleGesture (kružni pokret prsta), SwipeGesture (kontinuirani pokret otvorenog dlana u jednom smjeru), KeyTapGesture (pokret pritiska tipke na tipkovnici, prema dolje) te ScreenTapGesture (pokret pritiska tipke na ekranu, prema naprijed), međutim ti razredi su deprecated te se ne preporuča njihovo korištenje. Stoga su za potrebe programa implementirani sljedeći razredi: MyGesture, PointGesture, PinchGesture,

GrabGesture, FlatHandGesture, ThumbUpGesture te ThumbDownGesture, kojima su modelirane sve potrebne geste za rad programa. Svaka gesta za praćenu ruku može biti u jednom od četiri stanja:

START – predstavlja prvi trenutak u kojem je detektirana gesta na ruci,

ONGOING – svi trenutci dok gesta traje (nakon prvoga),

STOP – svi trenutci u kojima gesta nije prepoznata na praćenoj ruci, te

INVALID – ukoliko ne postoji praćena ruka

Stanja su modelirana enumeracijom GestureState.

4.2.1. Apstraktni razred MyGesture

Ovaj apstraktni razred predstavlja zajedničku implementaciju svim gestama. Sadrži rječnik koji preslikava praćenu ruku na stanje geste za tu ruku. Na taj način je istu gestu moguće raspoznati na većem broju ruku. Metodom GestureState getState(Hand hand) dohvaća se trenutno stanje geste za dotičnu ruku. U metodi Update() (koja se poziva u svakom okviru) iz trenutnog okvira dohvaća se lista detektiranih ruku te se za svaku ruku određuje stanje geste metodom GestureState isDetected(Hand hand). Ta metoda je apstraktna te njena implementacija ovisi o željenoj gesti. Sve geste nasljeđuju ovaj razred te im je jedina zadaća implementirati ovu metodu.

```
private Dictionary<int, GestureState> handsToStates = new Dictionary<int, GestureState>();

public abstract GestureState isDetected(Hand hand);

public GestureState getState(Hand hand) {
    if (!handsToStates.ContainsKey(hand.Id)) {
        return GestureState.INVALID;
    }

    return handsToStates[hand.Id];
}

void Update() {
    List<Hand> hands = c.Frame().Hands;

    foreach(Hand hand in hands) {
        if(!handsToStates.ContainsKey(hand.Id)) {
            handsToStates.Add(hand.Id, GestureState.STOP);
        }
        handsToStates[hand.Id] = isDetected(hand);
    }
}
```

Isječak koda 1 - Bitan dio koda razreda MyGesture

4.2.2. Konkretni razredi

S obzirom da se geste razlikuju isključivo po implementaciji metode isDetected(Hand hand), u nastavku će biti opisana implementacija jedne konkretne geste, dok su ostale geste implementirane na sličan način.

Gesta Pinch Gesture

Ovaj razred predstavlja gestu „štipanja“, odnosno približavanje kažiprsta i palca. Konstruktor prima 2 parametra – activationDist i maintainDist. Parametar activationDist je maksimalna udaljenost između kažiprsta i palca kako bi se gesta detektirala, dok je maintainDist minimalna udaljenost između kažiprsta i palca da bi se gesta prestala detektirati. Na ovaj način se pokušava doskočiti problemu povremene nepreciznosti detekcije položaja prstiju, koji bi inače mogli uzrokovati neželjeno ponašanje programa.

Metoda isDetected implementirana je na način da se dohvaćaju palac i kažiprst ruke, te se pomoću thumb.TipPosition.DistanceTo(index.TipPosition) mjeri udaljenost između njihovih vrhova. Ukoliko je prethodno stanje geste STOP te se ustanovi da je udaljenost prstiju manja od udaljenosti potrebne za aktivaciju geste (activationDist), trenutno stanje geste postaje START. Ukoliko prethodno stanje nije bilo STOP (dakle bilo je START ili ONGOING), te se ustanovi da je udaljenost prstiju manja od maintainDist, zaključuje se da je gesta još uvijek traje te se trenutno stanje postavlja na ONGOING. Ukoliko niti jedan od prethodno navedenih uvjeta nije ispunjen, gesta nije detektirana te trenutno stanje geste za praćenu ruku postaje STOP.

```
public float activationDist;
public float maintainDist;

public PinchGesture(float activationDist, float maintainDist) {
    this.activationDist = activationDist;
    this.maintainDist = maintainDist;
}
public override GestureState isDetected(Hand hand) {
    List<Finger> fingers = hand.Fingers;

    Finger thumb = fingers[0];
    Finger index = fingers[1];

    float distance = thumb.TipPosition.DistanceTo(index.TipPosition);

    if (getState(hand) == GestureState.STOP && distance <= activationDist) {
        return GestureState.START;
    }

    if (getState(hand) != GestureState.STOP && distance <= maintainDist) {
        return GestureState.ONGOING;
    }

    return GestureState.STOP;
}
```

Isječak koda 2 - Razred PinchGesture

4.3. Mreža objekta u razvojnoj okolini Unity

Unity za prikaz objekata koristi komponente Mesh Filter i Mesh Renderer. Mesh Filter sadrži podatke o mreži objekta u obliku razreda Mesh, dok je komponenta Mesh Renderer zadužena za iscrtavanje mreže na ekranu. Da bi se objekti mogli oblikovati, potrebno je mijenjati podatke u objektu tipa Mesh, stoga će struktura ovog razreda biti opisana u nastavku.

4.3.1. Razred Mesh

Mreža objekta sastoji se od niza trokuta raspoređenih u trodimenzionalnom prostoru kako bi se dobila impresija čvrstoga tijela. Svaki trokut definiran je svojim trima vrhovima, te normalom koja određuje prednju stranu trokuta. Unity koristi algoritam odbacivanja stražnjih poligona (engl. backface culling), tako da je samo

prednja strana trokuta vidljiva. U razredu Mesh svi su vrhovi objekta pohranjeni u jednom polju vertices tipa Vector3, koji predstavlja poziciju vrha u 3D prostoru. Svaki trokut definiran je trima indeksima koji referenciraju polje vertices, dok je normala određena redoslijedom tih vrhova pravilom lijeve ruke – ako se lijeva ruka okrene tako da prsti ruke pokazuju u smjeru navođenja vrhova trokuta, tada će palac pokazivati u smjeru normale. Drugim riječima, ako se kamera pozicionira tako da se navedeni vrhovi obilaze u smjeru kazaljke na satu, tada će normala tog trokuta gledati prema kameri. Indeksi kojima se definiraju trokuti pohranjeni su u jedno polje tipa int, te se grupiraju po tri – prva 3 indeksa referenciraju vrhove koji tvore prvi trokut, sljedeća 3 tvore drugi trokut i tako dalje.

Svaki vrh mreže uz poziciju može sadržavati još neke podatke kao što su normala, UV koordinata, tangenta, boja… Svako svojstvo definirano je u dodatnom polju, pa tako primjerice ako mreža ima 10 vrhova, postojat će još i polje normals veličine 10 koje će sadržavati normale vrhova. Pritom normala s indeksom 0 odgovara vrhu iz polja vertices s indeksom 0 i tako dalje. Ovakva struktura podataka za sobom povlači i određene posljedice, pa ako neki vrh objekta treba imati 2 normale (kako bi se prikazali oštri rubovi), jedina opcija je definirati 2 vrha u polju vertices koji će imati istu poziciju, ali će se normale u polju normals koje korespondiraju dotičnim vrhovima razlikovati. Tako primjerice kocka u Unity-ju nema 8 vrhova, već čak 24, jer je za svaki matematički vrh kocke potrebno definirati 3 normale – po jednu za svaku stranicu koju vrh dotiče.

Razred Mesh nudi još i opciju podjele trokuta u takozvane podmreže (engl. submesh) te je svakoj podmreži moguće pridružiti jedan materijal – svi trokuti iz prve podmreže bit će prikazani jednim materijalom, svi trokuti iz druge mreže bit će prikazani drugim materijalom i tako dalje. Ova opcija je od velike koristi ukoliko jedan objekt treba imati više materijala.

4.3.2. Razred MeshWrapper

S obzirom da razred Mesh ne pruža nikakvo sučelje kojim bi se lako modificirala mreža objekta, napisan je razred MeshWrapper koji kao člansku varijablu sadrži primjerak razreda Mesh, te nudi metode čijim pozivima je modificiranje mreže u velikoj mjeri olakšano, pa tako postoje metode za dodavanje

vrhova mreže, selektiranje i deselektiranje vrhova mreže, kreiranje trokuta od selektiranih vrhova, translaciju vrhova, invertiranje normala trokuta… Također, s obzirom da Unity ne prikazuje stražnje poligone, a oni uvelike pomažu pri modeliranju objekta, razred MeshWrapper se brine i za dodavanje stražnjih poligona na način da članska varijabla mesh ima dvije podmreže – jednu za prednje, a jednu za stražnje poligone. Svaki put kada se kreira trokuta, zapravo se stvore dva trokuta, prednji i stražnji. Prednji trokut ima redoslijed vrhova definiran redoslijedom selekcije vrhova te se dodaje u prvu podmrežu, a stražnji trokut ima obrnut redoslijed vrhova te se dodaje u drugu podmrežu. Na sljedećoj slici (Isječak koda 3) dan je isječak koda metode createTrianglesFromSelected, koja od selektiranih vrhova mreže stvara trokute po uzoru na OpenGL-ov primitiv TRIANGLE_STRIP. Iterira se po listi selektiranih vrhova te se u listu vrhova mreže dodaju po tri vrha (na indeksima i, i+1 i i+2). Zatim se u listu frontTriangles dodaju tri indeksa koji čine trokut, a to su upravo ova tri prethodno dodana vrha koji se u listi vertices nalaze na indeksima offset, offset+1 te offset+2. Pritom se indeksi dodaju u obrnutom redoslijedu svaki put kad je brojač i neparan, kako bi novonastali trokuti imali normalu okrenutu u istom smjeru. Nakon toga se ponovno dodaju isti vrhovi listi vertices kako bi se mogao stvoriti i stražnji poligon. Međutim, ti vrhovi se dodaju u obrnutom redoslijedu kako bi stražnji trokut imao suprotnu normalu u odnosu na prednji trokut.

```
public void createTrianglesFromSelected() {
    if (selectedVerts.Count < 3) {
        return;
    }

    List<Vector3> vertices = new List<Vector3>(mesh.vertices);
    List<int> frontTriangles = new List<int>(mesh.GetTriangles(0));
    List<int> backTriangles = new List<int>(mesh.GetTriangles(1));

    for (int i = 0; i < selectedVerts.Count - 2; i++) {
        int offset = vertices.Count;

        vertices.Add(selectedPositions[i]);
        vertices.Add(selectedPositions[i + 1]);
        vertices.Add(selectedPositions[i + 2]);

        frontTriangles.Add(offset);
        if(i % 2 == 0) {
            frontTriangles.Add(offset + 1);
            frontTriangles.Add(offset + 2);
        } else {
            frontTriangles.Add(offset + 2);
            frontTriangles.Add(offset + 1);
        }
        vertices.Add(selectedPositions[i]);
        vertices.Add(selectedPositions[i + 2]);
        vertices.Add(selectedPositions[i + 1]);

        backTriangles.Add(offset + 3);

        if(i % 2 == 0) {
            backTriangles.Add(offset + 4);
            backTriangles.Add(offset + 5);
        } else {
            backTriangles.Add(offset + 5);
            backTriangles.Add(offset + 4);
        }
    }

    mesh.vertices = vertices.ToArray();
    mesh.SetTriangles(frontTriangles.ToArray(), 0);
    mesh.SetTriangles(backTriangles.ToArray(), 1);
    mesh.RecalculateNormals();

    notify();
}
```

Isječak koda 3 - Metoda za stvaranje trokuta u razredu MeshWrapper

4.4. Akcije pokrenute gestama – razred GestureRecognizer

Svaki put kad se prepozna neka gesta, potrebno je izvršiti određenu akciju nad mrežom modeliranog objekta. Ovo ponašanje modelirano je razredom GestureRecognizer, koji ima 2 rječnika kao članske varijable: leftHandDictionary te rightHandDictionary. Ovi rječnici su po tipu OrderedDictionary, što znači da čuvaju poredak dodavanja ključeva te se ključ-vrijednost parovi (engl. key-value pair) pri iteraciji po rječniku dohvaćaju u redoslijedu u kojem su dodani u rječnik. Pri inicijalizaciji GestureRecognizer-a ovi rječnici se popune ključ-vrijednost parovima gdje su ključevi primjerci razreda MyGesture, dok su vrijednosti implementacije sučelja GestureAction, o kojem će biti riječi kasnije. Kôd metode Update(), koja je ključni dio ovog razreda, može se vidjeti na sljedećoj slici (Isječak koda 4). U metodi Update(), koja se poziva svaki okvir, iz trenutnog okvira se dohvaća lista detektiranih ruku. Ta lista se obilazi te se, ovisno o tome je li ruka desna ili lijeva, prolazi kroz odgovarajući rječnik i svaka akcija se pokušava izvršiti na temelju stanja pridružene geste. Ukoliko se akcija uspješno izvrši, obilazak rječnika se prekida. Na taj način ostvaren je prioritet među gestama – gesta koja je prije dodana u rječnik prije će se ispitati, te ako se njoj pridružena akcija izvrši, ostale geste će se ignorirati. Prioritetom se izbjegava istodobno prepoznavanje međusobno sličnih gesti, koje bi inače moglo uzrokovati nepoželjno ponašanje programa.

```
void Update () {
    Frame frame = controller.Frame();
    List<Hand> hands = frame.Hands;

    foreach(Hand hand in hands) {
        if(hand.IsLeft) {
            foreach(DictionaryEntry pair in leftHandMap) {
                GestureState state = ((MyGesture)pair.Key).getState(hand);
                bool detected = ((GestureAction)pair.Value).perform(hand, state);
                if(detected) {
                    break;
                }
            }
        } else {
            foreach (DictionaryEntry pair in rightHandMap) {
                GestureState state = ((MyGesture)pair.Key).getState(hand);
                bool detected = ((GestureAction)pair.Value).perform(hand, state);
                if (detected) {
                    break;
                }
            }
        }
    }
}
```

Isječak koda 4 - Metoda Update razreda GestureRecognizer

4.5. Implementacija akcija

Sve akcije modelirane su sučeljem GestureAction, koje ima jednu metodu – bool perform(Hand hand, GestureState state). Na temelju stanja geste i ruke na kojoj je detektirana dotična gesta izvodi se određena akcija. Implementirane su sve akcije opisane u 3. poglavlju. U nastavku će biti analizirana implementacija jedne od konkretnih akcija, dok su preostale akcije implementirane na sličan način.

4.5.1. Razred AddVertexAction

Ova akcija zadužena je za dodavanje novih vrhova objektu. Na slici (Isječak koda 5) priložen je kôd ovog razreda. Razred ima dvije članske varijable, referencu na MeshWrapper te indeks trenutnog vrha, vertexIndex. Ukoliko je gesta u stanju STOP ili INVALID, akcija se neće izvršiti; trenutni indeks se postavlja na -1 te se vraća false kao oznaka da je izvršenje akcije neuspješno. Ako to nije slučaj, uzima se pozicija vrha palca (hand.Fingers[0].StabilizedTipPosition) koja se pomoću statičke metode razreda LeapUtil iz koordinatnog sustava Leap Motiona transformira u koordinatni sustav Unity-ja. Ako je stanje u ovom trenutku START, znači da je detektiranje geste tek započelo te se mreži dodaje novi vrh na poziciji vrha palca, a vertexIndex se postavlja na indeks upravo dodanog vrha. Ako je pak stanje ONGOING, znači da je mreži već dodan novi vrh. U ovom slučaju se neće dodati još jedan vrh, već će se pomicati prethodno zapamćeni vrh na poziciju vrha palca sve dok se gesta ne prestane detektirati.

```
public class AddVertexAction : GestureAction {

    private MeshWrapper mesh;
    private int vertexIndex = -1;

    public AddVertexAction(MeshWrapper mesh) {
        this.mesh = mesh;
    }

    public bool perform(Hand hand, GestureState state) {
        if(state == GestureState.STOP || state == GestureState.INVALID) {
            vertexIndex = -1;
            return false;
        }

        Vector3 position = LeapUtil.LeapVectorToScaledVector3(hand.Fingers[0].StabilizedTipPosition);

        if (state == GestureState.START) {
            vertexIndex = mesh.addVertex(position);
        } else if (state == GestureState.ONGOING) {
            mesh.moveVertex(vertexIndex, position);
        }
        return true;
    }
}
```

Isječak koda 5 - Razred AddVertexAction

4.6. Učitavanje i spremanje .obj datoteka

Program podržava učitavanje .obj datoteka te spremanje modeliranog objekta u dotičnom formatu. U tu svrhu razred MeshWrapper proširen je dvjema sljedećim metodama:

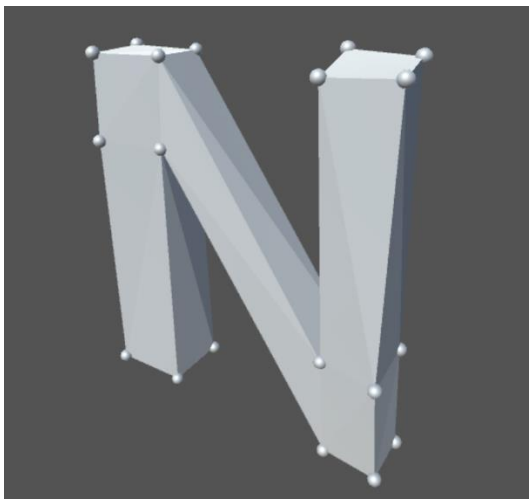
1. string meshToObj()
    - na temelju mreže objekta stvara string prema specifikaciji formata. Za svaki vrh objekta kreira se zapis oblika „v x y z“, gdje su x, y i z koordinate vrha. Za svaki trokut stvara se zapis oblika „f v_1 v_2 v_3“, gdje su v_1, v_2 i v_3 indeksi vrhova koji čine trokut. Bitno je napomenuti kako ova metoda zapisuje samo prednje trokute, iako se mreža objekta sastoji i od stražnjih trokuta – već je ranije rečeno kako su stražnji trokuti dodani objektu samo kako bi se olakšao proces modeliranja.

2. void loadFromObj(string obj)
    - na temelju parametra, stringa koji treba biti napisan prema specifikaciji formata .obj, učitava mrežu objekta. Za svaki zapis oblika „v x y z“ u mrežu se dodaje novi vrh s koordinatama x, y i z, dok se za svaki zapis oblika „f v_1 v_2 v_3“ dodaju 2 trokuta, jedan prednji kojem je redoslijed vrhova identičan redoslijedu u zapisu, te jedan stražnji kojem su vrhovi u obrnutom redoslijedu u odnosu na zapis.

Ove metode koristi razred ObjLoader, koji je zadužen za učitavanje, odnosno spremanje objekta kada korisnik pritisne odgovarajuće gumbe na izborniku. Pritom je moguće specificirati bilo koju putanju na disku.

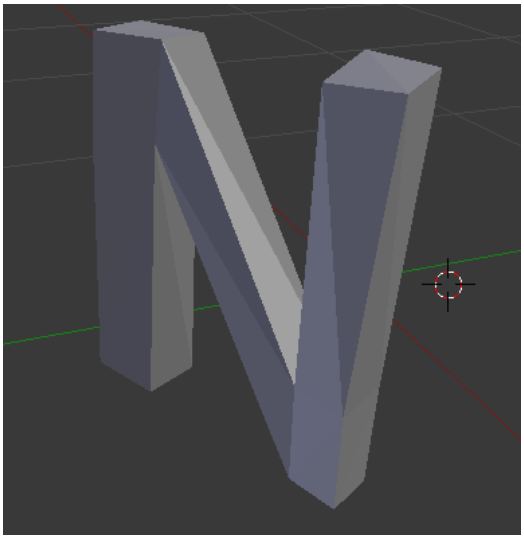
5. Rezultati

Kao primjer objekta u potpunosti modeliranog uporabom razvijenog programa napravljen je 3D model slova N, koji se može vidjeti na slici 18.



Slika 18 - 3D model slova N u programu

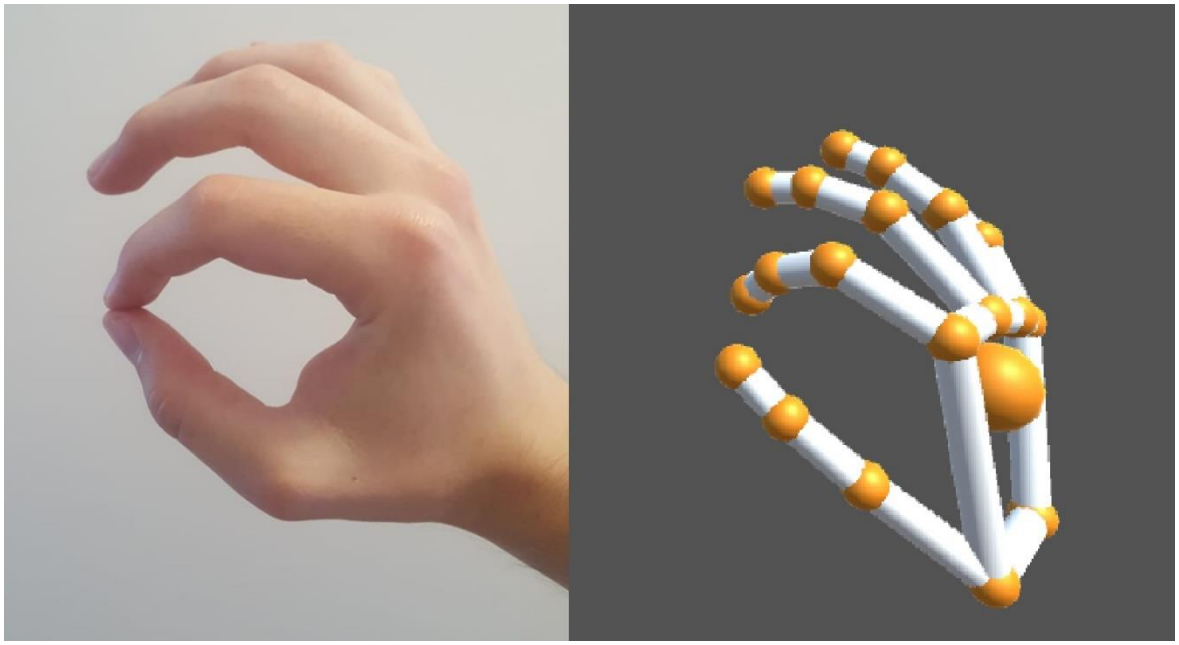
Nakon modeliranja, objekt je spremljen u obliku .obj datoteke, koju je dalje moguće učitati u neki drugi program za 3D modeliranje, primjerice Blender. Na slici 19 može se vidjeti prikaz modela nakon učitavanja u Blender.



Slika 19 - Model slova N učitan u Blender

5.1 Robusnost detektiranja gesti

Jedan od problema koji su se javljali prilikom razvijanja programa bilo je pogrešno aproksimiranje položaja virtualnih ruku u odnosu na pravi položaj ruku. Posljedica toga je da ponekad korisnik rukama točno oponaša gestu, međutim virtualne ruke nisu u sasvim jednakom položaju te se gesta ne detektira. Da bi se utjecaj ovog problema u što većoj mjeri smanjio te da bi se povećala robusnost detektiranja gesti i da bi korisničko iskustvo bilo što bolje, sve geste su oblikovane na način da imaju podesivi stupanj slobode. Primjer te slobode već je spomenut u opisu konkretne geste PinchGesture u poglavlju 4.2.2, gdje su definirana dva parametra – activationDist i maintainDist. Ti parametri predstavljaju minimalnu udaljenost između palca i kažiprsta kako bi se gesta počela detektirati, odnosno maksimalnu udaljenost između prstiju da bi se gesta nastavila detektirati. Bez tih parametara gesta bi se detektirala isključivo u slučajevima kada se vrhovi kažiprsta i palca virtualne ruke dotaknu, što ponekad zbog nesavršenog detektiranja položaja ruku može biti teško postići. Na slici 20 može se uočiti ta nesavršenost detektiranja – vrh kažiprsta i palca prave ruke se dotiču, dok se prsti virtualne ruke ne dotiču.



Slika 20 - Usporedba pravog položaja ruke (lijevo) i detektiranog položaja ruke (desno)

Uvođenjem stupnja slobode dozvoljava se odstupanje od savršenog oblika geste. Podešavanjem parametara ta se sloboda može povećati ili smanjiti, što ima

svoje prednosti, ali i mane – ako se parametri previše smanje, korisnik mora točnije oponašati gestu i tu problemi aproksimacije mogu doći do izražaja. S druge strane, ako se parametri previše povećaju, dozvoljavat će se veliko odstupanje od izvorne geste te se može dogoditi da se gesta detektira čak i kada se ne bi trebala detektirati. Stoga je bilo potrebno pronaći prave vrijednosti parametara kako bi se minimizirao ukupni negativni utjecaj oba ekstremna slučaja.

5.2 Ograničenja modeliranja gestama

Modeliranje 3D objekata može biti vrlo zahtjevan posao, pogotovo kada je potrebno modelirati kompleksne objekte s velikim brojem vrhova. Da bi se proces u što većoj mjeri olakšao i ubrzao, programi za 3D modeliranje obično raspolažu širokom lepezom raznih alata za modifikaciju objekta. Međutim, u slučaju programa razvijenog u sklopu ovog završnog rada možda je bolje zadržati se na skupu elementarnih akcija. U suprotnom bi bilo potrebno definirati velik broj različitih gesti kako bi se mogao podržati velik broj akcija, što bi moglo biti kontraproduktivno – korisnicima bi se moglo dogoditi da slučajno aktiviraju gestu koju nisu namjeravali napraviti. Također, broj različitih gesti koje je uopće moguće definirati ograničen je brojem ruku i prstiju. Bilo bi potrebno pronaći alternativni način preslikavanja gesti na akcije. Jedna ideja mogla bi biti da se definira nekoliko stanja programa te da se definiraju geste kojima se prelazi između tih stanja. Nadalje, u različitim stanjima bi se mogle javljati identične geste, ali u kontekstu trenutnog stanja one bi izvodile različite akcije.

6. Zaključak

U sklopu ovog završnog rada proučena je mogućnost kombiniranja virtualne stvarnosti i 3D modeliranja uporabom Oculus Rifta i Leap Motiona, te se kombinacija pokazala vrlo interesantnom i zabavnom za korištenje. Prednost ovog pristupa modeliranju je vizualizacija i doživljaj modeliranog objekta – stječe se vrlo uvjerljiv dojam da se objekt nalazi pred korisnikovim očima. Također, modeliranje gestama je poprilično intuitivno, a činjenica da se rukama radi u tri dimenzije znatno olakšava definiranje točke u prostoru u odnosu na miš i tipkovnicu, gdje se trodimenzionalna točka definira preko nekoliko različitih perspektiva u dvije dimenzije.

Literatura

[1] Wikipedia, stereoscopy

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

Datum pristupanja: travanj 2018.

[2] ifixit, Oculus Rift teardown

https://www.ifixit.com/Teardown/Oculus+Rift+CV1+Teardown/60612

Datum pristupanja: travanj 2018.

[3] Leap Motion official

https://www.leapmotion.com/

Datum pristupanja: svibanj 2018.

[4] Unity Documentation

https://docs.unity3d.com/

Datum pristupanja: travanj – lipanj 2018.

[5] Leap Motion Developer Documentation

https://developer.leapmotion.com/documentation/

Datum pristupanja: travanj – lipanj 2018.

Sažetak

U ovom radu obrađena je mogućnost uporabe Leap Motion dubinske kamere te 3D naočala Oculus Rift u kontekstu izrade trodimenzionalnih objekata. U uvodnom poglavlju dani su primjeri uporabe trodimenzionalnih objekata te programa za njihovo kreiranje. U sljedećem poglavlju opisane su tehničke značajke uređaja Oculus Rift i Leap Motion. U trećem poglavlju opisan je program razvijen u sklopu ovog rada. U četvrtom poglavlju opisani su ključni implementacijski detalji programa, dok su u posljednjem poglavlju navedena ograničenja ovog postupka modeliranja.

Ključne riječi: 3D modeliranje, raspoznavanje gesti, Oculus Rift, Leap Motion, Unity

Abstract

This paper explores the possibility of combining Leap Motion depth camera with Oculus Rift HMD (Head Mounted Display) in the context of 3D modelling. Examples of 3D model usages are given in the introductory chapter along with examples of 3D modelling software. Next chapter gives an overview of the technical specifications of Oculus Rift and Leap Motion. The third chapter describes the program developed as a part of this thesis. The fourth chapter deals with the key implementation details of the program, while the last chapter enlists the limitations of this approach to modelling.

Key words: 3D modelling, gesture recognition, Oculus Rift, Leap Motion, Unity