jamk

# Liikkeenkaappaus virtuaalitodellisuus-laitteistoa käyttäen Unity-moottorilla

Joni Laukka

Opinnäytetyö, AMK
Huhtikuu 2022
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), Tieto- ja viestintätekniikan tutkinto-ohjelma

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

**Kuvailulehti**

**Laukka, Joni**

**Liikkeenkaappaus virtuaalitodellisuuslaitteistoa käyttäen Unity-moottorilla**

Jyväskylä: Jyväskylän ammattikorkeakoulu. Huhtikuu 2022, 43 sivua

Tietojenkäsittely ja tietoliikenne. Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Verkkojulkaisulupa myönnetty: kyllä

**Tiivistelmä**

Opinnäytetyössä toteutettiin liikkeenkaappausympäristö, jossa voidaan kaapata käyttäjän liikkeitä reaaliaikaisesti virtuaalitodellisuusjärjestelmän laitteiden avulla. Toteutetun liikkeenkaappausympäristön avulla haluttiin saada selville, onko virtuaalitodellisuusjärjestelmällä mahdollista kaapata käyttökelpoista liikedataa, niin että siitä voidaan luoda animaatioita esimerkiksi kolmiulotteisille pelihahmoille.

Liikkeenkaappausympäristö toteutettiin Unity-ympäristössä. Työssä käytettiin HTC VIVE -virtuaalitodellisuusjärjestelmää kahden Valve Index -virtuaalitodellisuusohjaimen kanssa. Liikkeenkaappauksen apuna käytettiin myös neljää VIVE Tracker -lisäseurantalaitetta, jotta ympäristössä pystyttiin tallentamaan koko ihmiskehon liikkeet.

Lopullisella projektiympäristöllä pystytään tallentamaan käyttäjän liikkeitä Unityn animaatiokäyriksi skriptattavaan objektiin, joiden avulla pelihahmo voidaan animoida käyttäen Unityn Mecanim-järjestelmän käänteiskinematiikkaa. Lopuksi animoidun pelihahmon liikkeet tallennetaan Mecanim-järjestelmään yhteensopivaksi animaatiotiedostoksi.

**Avainsanat (asiasanat)**

Virtuaalitodellisuus, Liikkeenkaappaus, Unity

**Muut tiedot (salassa pidettävät liitteet)**

jamk

**Description**

**Laukka, Joni**

**Motion capture using virtual reality devices in Unity-engine**

Jyväskylä: JAMK University of Applied Sciences, April 2022, 43 pages.

Information and Communications. Degree Programme in Information and Communications Technology. Bachelor´s thesis.

Permission for web publication: Yes

Language of publication: Finnish

**Abstract**

The objectives of the thesis were to implement a motion capture environment, where the user's movements can be recorded in real time using virtual reality devices. The implemented environment was used to find out if virtual reality systems could be used to capture useful movement data from the user's movements so the data could be used to create animations, for example for a three-dimensional game character.

The motion capture environment was implemented using the Unity-platform. The virtual reality system that was used in the thesis was HTC VIVE virtual reality system with two Valve Index virtual reality controllers. Four additional VIVE Tracker virtual reality trackers were used to get user's full body motions while recording.

The final product can be used to record user's movement to Unity's animation curves in a scriptable object, which can be used to animate video game character using Unity's Mecanim system's inverse kinematics. Finally, the animated character's movement will be saved to Mecanim compatible animation file.

**Keywords/tags (subjects)**

Virtual reality, Motion capture, Unity

**Miscellaneous (Confidential information)**

**Sisältö**

**Kuviot**

## Käsitteet

| | |
|---|---|
| Animaatiokäyrä | Unityn luokka, jonka avulla kahden luvun pareista voidaan muodostaa interpoloitava käyrä. |
| Avatar | Unityn käyttämä luurankomalli pelihahmosta. |
| Elinkaarifunktio | Unityn objektien elinkaaren aikana automaattisesti komponenteissa tapahtuva funktiokutsu. |
| Gyroskooppi | Esineen asentoa mittaava laite. |
| IK | Inverse Kinematics, käänteiskinematiikka. |
| IMU | Inertial Measurement Unit, esineen suunnan muutosta mittaava laite. |
| Interpolaatio | Menetelmä, jolla voidaan laskea kahden luvun välinen arvo. |
| Komponentti | Unityn objekteihin kiinnitettävä skripti. |
| Kvaternio | Unityn käyttämä esitysmalli objektin suunnasta. |
| Lapsiobjekti | Unityn peliobjektiin kiinnitetty objekti. |
| Liitännäinen | Unityyn Unityn ulkopuolelta tuotu lisäosa. |
| LTS | Long Term Support, pitkäaikainen tuki. |
| Ohjaintoiminto | SteamVR-rajapinnan toiminto laitteen ja ohjelmiston väliselle toiminnalle. |
| Prefab | Unityn peliobjektista resursseihin tallennettu malliobjekti. |
| Rajapinta | Ohjelmistojärjestelmien välinen kommunikaatioyhteys. |
| Skriptattava objekti | Unityn resurssi, joka sisältää muokattavaa tietoa. |
| Skripti | Unityn komponenttiin ohjelmoitu koodi. |

| | |
|---|---|
| Tilakone | Järjestelmä, jonka tila vaihtelee muuttujien mukaan. |
| Vapausaste | Suunta-akseli, minkä mukaan esineen on mahdollista liikkua tai kääntyä. |
| VR | Virtual Reality, virtuaalitodellisuus. |
| XR | Extended Reality, laajennettu todellisuus. |

# 1 Johdanto

## 1.1 Taustat

Kolmiulotteisten pelihahmojen animoiminen alusta alkaen käsin on hyvin työlästä ja aikaa vievää työtä. Animointia voidaan nopeuttaa ja helpottaa käyttämällä liikkeenkaappausjärjestelmiä, joiden avulla ihmisnäyttelijän liikkeitä voidaan digitalisoida reaaliaikaisesti. Liikkeenkaappausjärjestelmät ovat yleensä kuitenkin kalliita, ja monet järjestelmät vaativat suuren tilan toimiakseen. (Hibbitts 2020.)

Kehittyneet virtuaalitodellisuusjärjestelmät käyttävät liikkeenkaappausjärjestelmiä vastaavia menetelmiä käyttäjän paikantamiseen. Virtuaalitodellisuusjärjestelmät eivät kuitenkaan vaadi suuria tiloja ja ovat liikkeenkaappausjärjestelmiä halvempia (Hibbitts 2020). Opinnäytetyöllä haluttiin selvittää soveltuisiko virtuaalitodellisuusjärjestelmä liikkeenkaappausjärjestelmäksi, mikä mahdollistaisi pienen budjetin liikkeenkaappaamisen harrastelijoille ja pienille yrityksille.

## 1.2 Tavoitteet ja toteutus

Opinnäytetyön tavoitteena oli kehittää projektiympäristö, jossa voidaan kaapata ja tallentaa ihmisen liikettä reaaliaikaisesti virtuaalitodellisuusjärjestelmää apuna käyttäen. Kehitetyn projektiympäristön avulla haluttiin analysoida, soveltuuko käytetty virtuaalitodellisuusjärjestelmä ihmisliikkeiden kaappaamiseen, ja kuinka laadukasta liikedataa järjestelmällä voidaan tuottaa. Projektiympäristön soveltuvuutta analysoitiin käytettävyyden ja kerätyn liikedatan tarkkuuden kannalta.

Opinnäytetyössä tuotettu liikkeenkaappausprojektiympäristö toteutettiin Unity-ympäristössä. Ympäristössä on mahdollista tallentaa ja muokata käyttäjän liikettä virtuaalitodellisuusjärjestelmän avulla. Virtuaalitodellisuusjärjestelmänä käytettiin HTC VIVE -virtuaalitodellisuusjärjestelmää Valve Index -ohjaimilla. Lisäksi järjestelmään lisättiin VIVE Tracker -lisäseurantalaitteita, jotka mahdollistavat koko ihmiskehon liikkeiden seuraamisen reaaliaikaisesti.

# 2 Keskeiset teknologiat

## 2.1 Virtuaalitodellisuus

Woodfordin (2021) ja Lowoodin (n.d.) mukaan virtuaalitodellisuudella tarkoitetaan tietokoneella luotua kolmiulotteista virtuaalista maailmaa, simulaatiota, johon käyttäjän on mahdollista vaikuttaa virtuaalitodellisuusjärjestelmän avulla. Termin virtuaalitodellisuus keksi Jaron Lanier vuonna 1987 (Lowood n.d.). Tarkempaa määritelmää virtuaalitodellisuudelle on vaikea määritellä, koska virtuaalitodellisuusteknologiat ja laitteet kehittyvät nykypäivänä todella nopeasti. Määritelmään vaikuttavat esimerkiksi, miten virtuaalinen maailma näytetään käyttäjälle, tarvitseeko virtuaalitodellisuusjärjestelmän sisältää käyttäjän silmille asetettavia laseja tai kuinka interaktiivinen virtuaalisen maailman tulisi olla, että sitä voidaan kutsua virtuaalitodellisuudeksi. (Vince 2004, 5.)

Nykyään virtuaalitodellisuus mielletään osaksi isompaa kokonaisuutta, laajennettu todellisuus, minkä tarkoituksena on eritellä erilaiset tietokoneella luodut virtuaaliset simulaatiot toisistaan. Laajennettuun todellisuuteen kuuluvat virtuaalitodellisuuden lisäksi lisätty todellisuus sekä yhdistetty todellisuus. Laajennetussa todellisuudessa virtuaalitodellisuudella tarkoitetaan tietokoneella luotua, täysin virtuaalista maailmaa, joka näytetään käyttäjälle virtuaalitodellisuuslasien kautta, kun taas lisätyn todellisuuden järjestelmillä käyttäjän näkökenttään lisätään tietokoneella luotuja elementtejä, kuin ne olisivat todellisia. Yhdistetyllä todellisuudella tarkoitetaan virtuaali- ja laajennetun todellisuuden yhdistelmää, missä todellisuuden ja virtuaalisen maailman elementit vuorovaikuttavat keskenään. (Hooker 2021.)

## 2.2 Virtuaalitodellisuusjärjestelmät

Virtuaalitodellisuusjärjestelmät ovat laitteita tai laitekokonaisuuksia, jotka mahdollistavat virtuaalitodellisuuden käytön. Yleisimmät virtuaalitodellisuusjärjestelmät toimivat käyttäjän silmille asetettavilla virtuaalitodellisuuslaseilla, joilla käyttäjä näkee virtuaalisen maailman stereoskooppisena kuvana, eli kahden lähes samanlaisen kuvan avulla muodostetaan illuusio kolmiulotteisesta maailmasta. Virtuaalitodellisuuslasit sisältävät yleensä myös lasien liikettä seuraavia sensoreita, esimerkiksi gyroskooppeja tai infrapunasensoreita, joiden avulla pystytään seuraamaan järjestelmän laitteiden liikkeitä. Yleisimmät järjestelmät sisältävät myös ohjaimia, joiden avulla käyttäjän kädet voidaan tuoda virtuaaliseen maailmaan. (Woodford 2021; Lowood n.d.)

Modernit virtuaalitodellisuusjärjestelmät voidaan jakaa kahteen ryhmään niiden käytettävyyden perusteella: itsenäisiin järjestelmiin ja tietokoneeseen kytkettäviin järjestelmiin. Tietokoneeseen liitettävät virtuaalitodellisuusjärjestelmät ovat järjestelmiä, jotka vaativat ulkoisen tietokoneen toimiakseen. Järjestelmä voidaan kytkeä tietokoneeseen joko kaapelilla tai langattomasti. Ulkoisen tietokoneen ansiosta kytkettävät järjestelmät mahdollistavat paremman suorituskyvyn itsenäisiin järjestelmiin verrattuna. Itsenäiset virtuaalitodellisuusjärjestelmät sisältävät itsessään kaiken tarvittavan laitteiston, mikä mahdollistaan järjestelmän käyttämisen langattomasti. On olemassa myös järjestelmiä, jotka toimivat virtuaalitodellisuuslasien sisälle kiinnitettävän matkapuhelimen avulla. Puhelinjärjestelmien suorituskyky on yleensä heikko, minkä takia ne soveltuvat vain yksinkertaisiin sovelluksiin, kuten virtuaalitodellisuusvideoiden katselemiseen. (Angelov, Petkov, Shipkovenski & Kalushkov 2020, 1-2.)

Virtuaalitodellisuusjärjestelmät voidaan jakaa myös kahteen ryhmään, riippuen kuinka vapaasti käyttäjä pystyy liikkumaan järjestelmän avulla virtuaalimaailmassa. Varhaiset järjestelmät sisälsivät vain pään kääntymistä seuraavia sensoreita, mikä mahdollisti vain pään ja katseen liikuttamisen virtuaalimaailmassa. Tällaisen järjestelmän sanotaan olevan kolmen vapausasteen järjestelmä. Nykyään yleisimmät virtuaalitodellisuusjärjestelmät toimivat kuudella vapausasteella, mikä mahdollistaa myös käyttäjän pään sijainnin seuraamisen, jolloin käyttäjän näkemä kuva virtuaalimaailmasta liikkuu luonnollisesti käyttäjän pään liikkeiden mukaisesti. (Angelov, Petkov, Shipkovenski & Kalushkov 2020, 3.)

## 2.3 Liikkeenkaappausjärjestelmät

Guerra-Filhon (2005) ja Hibbittsin (2020) mukaan liikkeenkaappauksella tarkoitetaan ihmisen, eläimen tai esineen liikkeen tallentamista digitaaliseen muotoon reaaliaikaisesti liikkeenkaappausjärjestelmän avulla. Liikkeenkaappaustekniikoita käytetään yleisesti esimerkiksi videopelien animaatioiden luonnissa sekä kolmiulotteisissa animaatioelokuvissa. (Guerra-Filho 2005; Hibbitts 2020.)

Yleisimmät liikkeenkaappausjärjestelmät ovat joko optisia järjestelmiä, sensoripohjaisia järjestelmiä tai näitä yhdistäviä hybridijärjestelmiä. On olemassa myös kuvapohjaisia järjestelmiä, mutta niiden tarkkuus liikkeenkaappauksessa on yleensä heikko. (Hibbitts 2020.)

### 2.3.1 Optiset järjestelmät

Optisissa liikkeenkaappausjärjestelmissä näyttelijän kehoon kiinnitetään pieniä seurantamerkkejä, joita seurataan useiden kameroiden avulla. Kamerat asetetaan niin, että näyttelijän liikkuessa vähintään kaksi kameraa näkee saman seurantamerkin. Kameroiden lähettämää tietoa analysoimalla voidaan selvittää seurantamerkkien sijainti reaaliaikaisesti, ja muuntaa sijaintitieto digitaaliseen muotoon. Seurantamerkkejä on kahdenlaisia: aktiivisia merkkejä sekä passiivisia merkkejä. (Guerra-Filho 2005, 3; Hibbitts 2020.)

Aktiiviset seurantamerkit tuottavat valoa, jota voidaan seurata kameroilla. Valo parantaa merkkien näkyvyyttä ja mahdollistaa lisätiedon lähettämisen järjestelmälle, esimerkiksi vilkuttamalla merkin valoa. Toisin kuin aktiiviset merkit, passiiviset merkit eivät tuota itse valoa tai tietoa järjestelmälle, joten passiiviset merkit tarvitsevat aktiivisia merkkejä paremmat valo-olosuhteet erottuakseen ympäristöstä. Heikko valaistus voi aiheuttaa myös tärinää kaapatussa liikedatassa, varsinkin jos liike on hidasta. (Hibbitts 2020.)

Optiset liikkeenkaappausjärjestelmät vaativat suuren kaappausympäristön sekä useita kameroita seuraamaan näyttelijäitä, minkä takia optiset järjestelmät ovat yleensä kalliimpia ja vaikeammin siirrettäviä kuin muut järjestelmät. Optiset järjestelmät ovat kuitenkin yleisimmistä järjestelmistä tarkimpia kaappaamaan liikettä, ja mahdollistavat monen näyttelijän liikkeiden kaappaamisen samanaikaisesti. (Hibbitts 2020; Hindle, Keogh & Lorimer 2021, 2.)

### 2.3.2 Sensoripohjaiset järjestelmät

Sensoripohjaiset järjestelmät käyttävät itsenäisiä, näyttelijöihin kiinnitettäviä inertial measurement unit-sensoreita eli IMU-sensoreita, jotka sisältävät mittauslaitteita liikkeen havainnointiin. IMU-sensorit voivat sisältää esimerkiksi gyroskooppeja, kiihtyvyysmittareita ja magnetometrejä, joiden avulla laite lähettää tietokoneelle tiedon liikkeen muutoksesta. (Hindle, Keogh & Lorimer 2021, 3.)

Sensoripohjaiset järjestelmät eivät vaadi ulkopuolisia kameroita kuvaamaan näyttelijöitä, eikä käytettävien IMU-sensoreiden tarvitse olla näkyvillä liikettä kaapatessa, mikä helpottaa järjestelmän käyttöä ja siirrettävyyttä. Toisaalta sensoripohjaiset järjestelmät ovat epätarkempia kuin optiset

järjestelmät, sillä sensoripohjaiset järjestelmät pohjautuvat IMU-sensoreiden liikkeiden muutoksiin, eikä sensoreiden ulkopuoliseen seurantaan, mikä vaikeuttaa sensoreiden virhemittausten korjaamista reaaliaikaisesti. (Hibbitts 2020.)

### 2.3.3 Hybridijärjestelmät

Hybridijärjestelmät ovat liikkeenkaappausjärjestelmiä, jotka käyttävät sekä optisten järjestelmien, että sensoripohjaisten järjestelmien tekniikoita hyödykseen. Hybridijärjestelmissä voidaan vaihdella käytettäviä kaappausmenetelmiä tilanteen mukaan. Esimerkiksi, näyttelijän kaappausmerkkien peittyessä näyttelijän liikkuessa, voidaan liikettä seurata IMU-sensoreiden avulla. Hybridiliikkeenkaappausjärjestelmiä voi olla kahdenlaisia: sensoripohjaisia hybridijärjestelmiä tai järjestelmäpohjaisia hybridijärjestelmiä. (Hibbitts 2020.)

Sensoripohjaisessa hybridijärjestelmässä käytettävät sensorit sisältävät sekä optisen-, että sensoripohjaisen liikkeenkaappausjärjestelmän ominaisuuksia. Sensorit voivat olla esimerkiksi aktiivisia sensoreita, joita seurataan ulkopuolisten kameroiden avulla, mutta sisältävät IMU-sensoreiden ominaisuuksia parantamaan liikkeenseurannan laatua. Järjestelmäpohjaisessa hybridijärjestelmässä käytetään yhtä aikaa sekä optista järjestelmää, että sensoripohjaista liikkeenkaappausjärjestelmää. Esimerkiksi näyttelijän sijainnin kaappaamiseen voidaan käyttää optista järjestelmää ja liikkeiden kaappaamiseen sensoripohjaista järjestelmää. (Hibbitts 2020.)

## 2.4 Käänteiskinematiikka

Käänteiskinematiikka on robotiikassa käytetty tekniikka, jossa algoritmien avulla lasketaan robotin nivelille asennot, jotta robotti saadaan haluttuun asentoon. Käänteiskinematiikkaa voidaan käyttää, jos tiedetään ketjumaisesti nivelillä kiinnitettyjen osien pituudet sekä haluttu loppupiste, johon ketjun viimeinen osa halutaan siirtää. Käänteiskinematiikan algoritmeissa voidaan ottaa huomioon nivelten liikkeitä rajoittavat tekijät, kuten kääntymisen suunta tai kääntöasteiden rajat. (Inverse Kinematics. N.d)

## 2.5 Unity-kehitysalusta

### 2.5.1 Yleistä

Unity on Unity Technologiesin kehittämä kehitysalusta reaaliaikaisen ja interaktiivisen median kehittämiseen. Unity-kehitysalusta on kehitetty erityisesti käytettäväksi videopelien, teollisuuden tuotteiden visualisoinnin sekä animaatioelokuvien tekemiseen. (Unity Platform N.d.)

Unity-kehitysalustalla voidaan tuottaa reaaliaikaista mediaa monille eri alustoille. Tuettuja alustoja ovat yleisimmät tietokone- ja mobiilikäyttöjärjestelmät, pelikonsolit, virtuaalitodellisuusjärjestelmät sekä verkkoselaimet. (Build once, deploy anywhere n.d.)

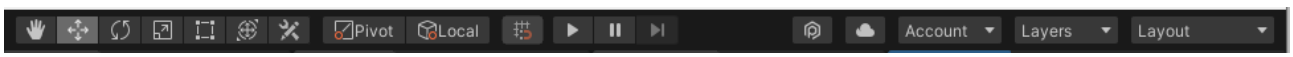
### 2.5.2 Unity Hub

Unity Hub on projektihallintasovellus Unity-projekteille. Hubin avulla käyttäjä voi asentaa tai poistaa eri Unity Editor -versioita, luoda uusia ja avata olemassa olevia projekteja sekä ladata käyttäjän tallentamia projekteja Unityn pilvipalvelusta. (Unity Hub n.d.)

### 2.5.3 Editorin käyttöliittymä

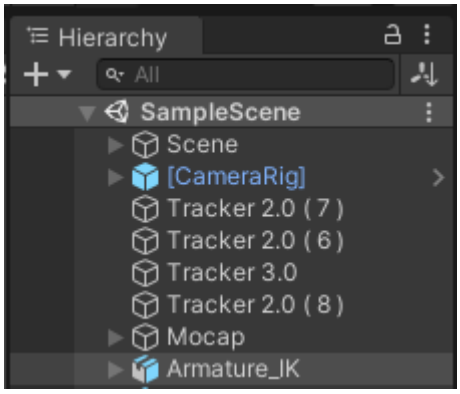
Unityn editorin käyttöliittymä koostuu työkalu- ja tilapalkista sekä käyttäjän määrittämistä työikkunoista. Yleisimmin käytetyt ikkunat ja näkymät ovat hierarkiaikkuna, pelinäkymä, kohtausnäkymä, valitun objektin hallintaikkuna sekä projekti-ikkuna. (Unity’s Interface n.d.)

Työkalupalkilta (ks. Kuvio 1) käyttäjä voi vaihtaa käytössä olevan työkalun. Työkaluilla käyttäjä voi siirtää, kääntää ja muokata editorin objekteja. Työkalupalkin keskellä on toisto-, pysäytys- sekä askelnappi pelitilan hallintaan. Palkin oikeassa laidassa on käyttäjän hallintapaneeli sekä editorin näkymän hallintapaneelit. (The Toolbar n.d.)



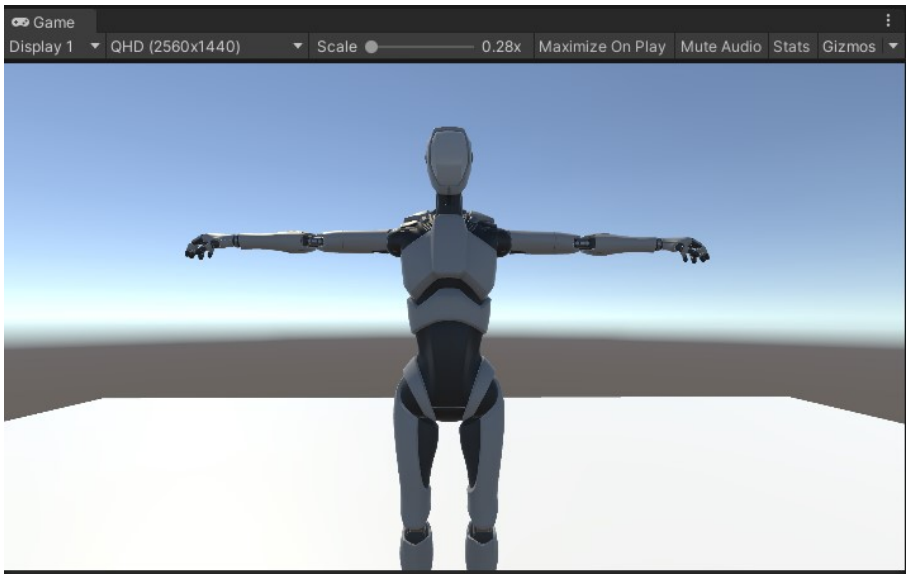

Kuvio 1. Unityn työkalupalkki.

Hierarkiaikkuna (ks. Kuvio 2) sisältää hierarkianäkymän aktiivisen kohtauksen objekteista. Ikkunassa käyttäjä voi luoda, poistaa objekteja sekä muokata niiden suhteita toisiinsa. (The Hierarchy window n.d)



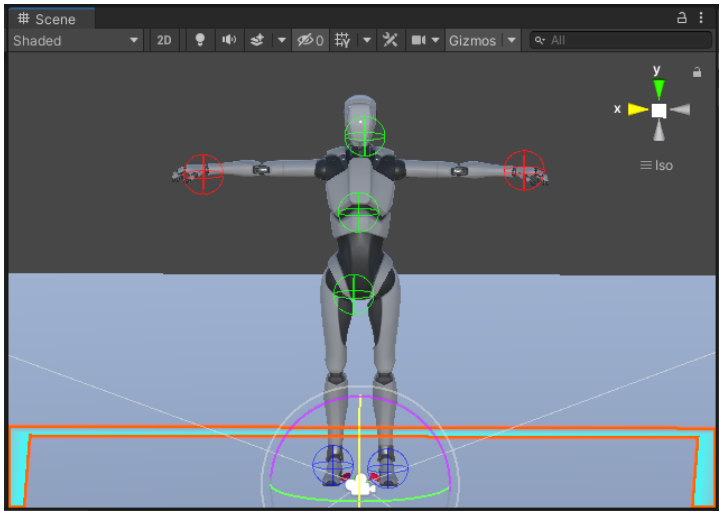

Kuvio 2. Unityn hierarkiaikkuna.

Pelinäkymästä (ks. Kuvio 3) nähdään aktiivisen kameran kuva. (The game view n.d.)



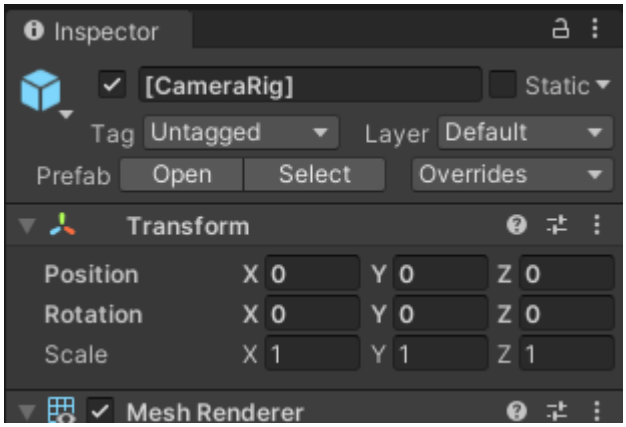

Kuvio 3. Unityn pelinäkymä.

Kohtausnäkymä (ks. Kuvio 4) on interaktiivinen ja visuaalinen näkymä aktiivisesta kohtauksesta. Näkymän avulla voidaan valita, siirtää ja muokata pelimaailman objekteja. (The Scene view n.d.)



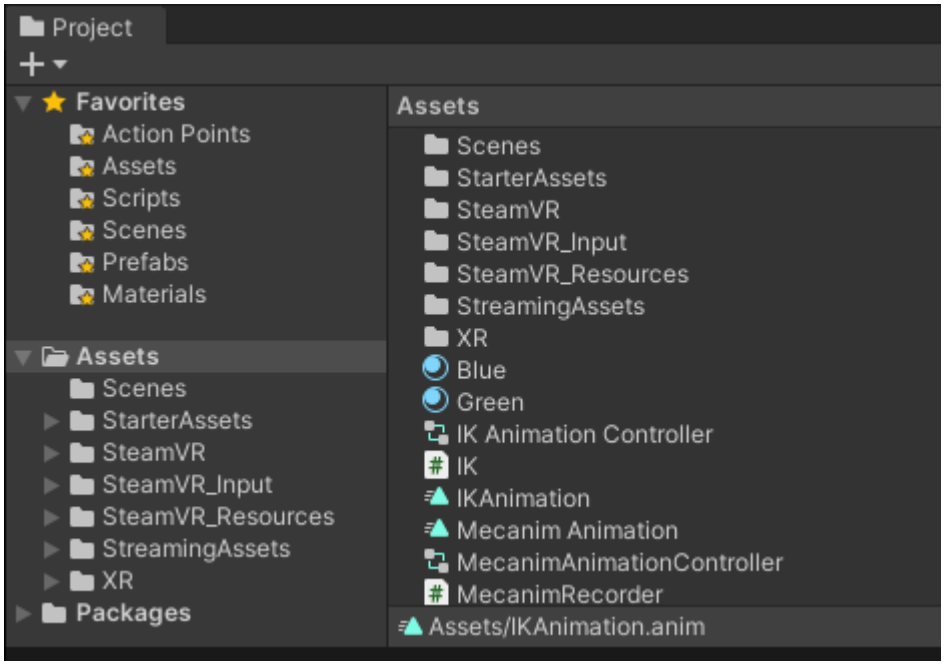

Kuvio 4. Unityn kohtausnäkymä.

Hallintaikkunaa (ks. Kuvio 5) käytetään projektin resurssien ominaisuuksien hallintaan ja tarkasteluun. Resursseja voivat olla esimerkiksi kohtauksen objektit, objekteista tallennetut prefab-objektit sekä ääni- ja materiaalitiedostot. (The Inspector window n.d.)




Kuvio 5. Unityn resurssien hallintaikkuna.

Projekti-ikkunan (ks. Kuvio 6) avulla voidaan valita, organisoida ja hallita projektin kansiorakennetta sekä tiedostoja ja muita resursseja. (The Project window n.d.)




Kuvio 6. Unityn projekti-ikkuna.

### 2.5.4 Komponentit

Unity-moottorissa peliobjektien toiminnallisuus luodaan komponenttien avulla. Komponentit ovat skriptejä, joita kiinnitetään peliobjekteihin, luoden objekteille toiminnallisuutta. Komponenttien avulla voidaan luoda vuorovaikutusta pelimaailman objektien välille. (Introduction to components n.d.)

Komponenttiskriptit ovat C#-ohjelmointikielellä kirjoitettuja skriptejä, jotka sisältävät muuttujia ja funktioita, joiden avulla luodaan Unityn peliobjektien toiminnallisuus. Komponentit periytyvät Unityn MonoBehaviour-perusluokasta, mikä mahdollistaa Unityn elinkaarifunktioiden käyttämisen objektin toiminnallisuuksissa. (Using Components n.d.)

Valitun peliobjektin komponentit näytetään editorin hallintaikkunassa. Komponentin julkisten muuttujien arvoja ja linkityksiä muihin objekteihin voidaan hallita hallintaikkunan näkymästä kyseisen komponentin kohdalta. Komponenttien muuttujien arvoja voidaan muokata, pelitilan ollessa käynnissä, mikä mahdollistaa tuotteen nopean iteroinnin halutun lopputuloksen saavuttamiseksi. (Using Components n.d.)

### 2.5.5 Mecanim-animaatiojärjestelmä

Mecanim on Unityn käyttämä animaatiojärjestelmä, minkä avulla voidaan hallita ja linkittää Unity-projektiin tuotuja animaatioita. Mecanim-järjestelmää käytetään editorin animaattorinäkymällä. Mecanim-järjestelmän toiminta perustuu monipuolisten tilakoneiden rakentamiseen. Mecanim-tilakone voidaan yhdistää Unityn peliobjekteihin, jolloin objektit voidaan animoida objektien tilojen mukaan. Tilakoneeseen voidaan lisätä animaatioita, ja linkittää niitä muihin, saman tilakoneen animaatioihin luoden yhteyksiä animaatioiden välille. Yhteyksille voidaan luoda rajoitteita, joiden mukaan animaatio vaihtuu seuraavaan animaatioon tilakoneen tilan muuttuessa. Tilakoneen tilaa voidaan hallita käyttämällä tilakoneen muuttujia, esimerkiksi lukuarvoja tai liipaisimia, joiden avulla tilakone osaa vaihtaa animaatiosta toiseen. (Animation System Overview n.d.)

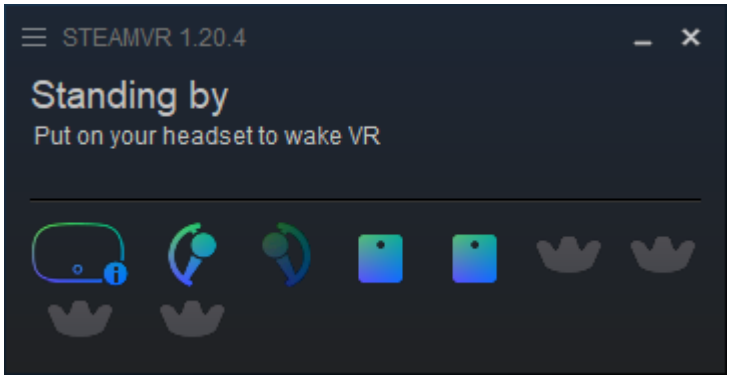
Humanoid-animaatiot käyttävät pelihahmoista luotuja avatar-malleja. Avatarit ovat Unityn käyttämiä malleja pelihahmon luurangosta, jotka sisältävät tiedon, mitkä mallin luut vastaavat avatar-mallin luita. Avatarit mahdollistavat saman humanoid-animaation käyttämisen eri pelihahmojen luurangoilla, sillä Mecanim-animaatiojärjestelmä animoi pelihahmot käyttäen hahmojen avatareihin määriteltyjä lihaksia. Lihaksien avulla hahmon luiden liikeradat voidaan määritellä hahmokohtaisesti, jolloin animaation alkuperäisen luurangon rakenteella ei ole väliä. (Animation System Overview n.d.)

# 3 Liikkeenkaappausympäristön toteutus

## 3.1 Laitteisto ja ohjelmistot

Virtuaalitodellisuuslaitteistona opinnäytetyössä käytettiin HTC VIVE -virtuaalitodellisuuslaseja. Käyttäjän käsien liikkeiden seurantaan käytössä oli kaksi Valve Index -virtuaalitodellisuusohjainta. Käyttäjän jalkojen sekä kehon liikkeiden seurantaan käytettiin kolmea VIVE Tracker 2.0 -lisäseurantalaitetta sekä yhtä VIVE Tracker 3.0 -lisäseurantalaitetta. Kaikkien käytettyjen laitteiden paikantaminen tapahtui ulkoisia tukiasemia käyttäen, joten käytetty liikkeenkaappauslaitteisto toimii sensoripohjaisen hybridijärjestelmän mukaisesti. Työssä oli käytössä kaksi ensimmäisen sukupolven VIVE-tukiasemaa, jotka olivat asennettu käyttöalueen vastakkaisiin yläkulmiin.

Työssä kehitetty projektiympäristö toteutettiin Unity-kehitysalustalla. Unitystä oli käytössä 2020.3.20f1 LTS -versio, mikä oli aloitusajankodan viimeisin Unityn suositeltu versio. Unity-projektin pohjana käytettiin Unity Hubilla luotua 3D-projektipohjaa. Virtuaalitodellisuusjärjestelmän hallintaan käytössä oli SteamVR-rajapinta (ks. Kuvio 7), mikä mahdollistaa rajapintaa tukevien laitteiden käyttämisen ja hallinnoimisen Unity-kehitysalustassa.

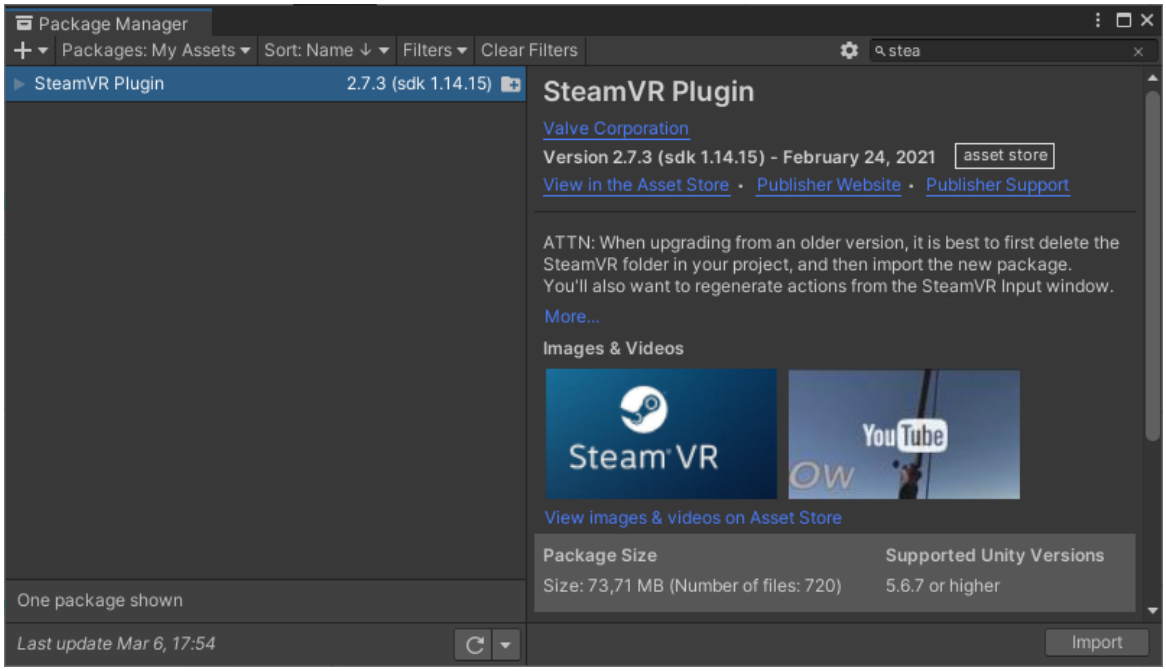



Kuvio 7. SteamVR-rajapinnan käyttöliittymä.

## 3.2 Virtuaalitodellisuusjärjestelmän liittäminen Unity-ympäristöön

### 3.2.1 SteamVR-liitännäinen

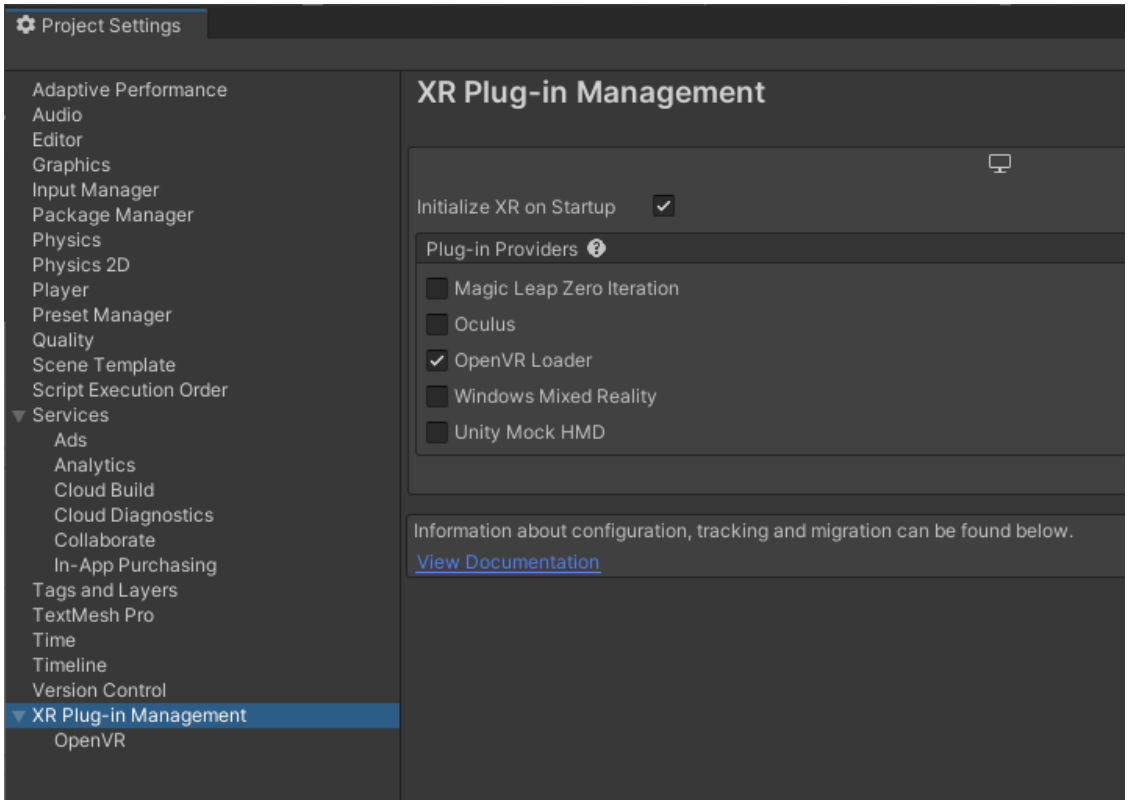
Virtuaalitodellisuusjärjestelmä liitettiin Unity-projektiympäristöön Valve Corporationin kehittämällä SteamVR-liitännäisellä, jonka avulla SteamVR-rajapinnan laitteita voidaan käyttää helposti Unity-projektissa. SteamVR-liitännäinen lisättiin projektiin Unityn pakettienhallintatyökalun avulla (ks. Kuvio 8). Liitännäisestä käytettiin versiota 2.7.3.




Kuvio 8. SteamVR-liitännäinen Unityn pakettien hallintatyökalussa.

SteamVR-liitännäistä tuodessa Unity-projektiin liitännäinen pyytää vaihtamaan projektiasetuksia (ks. Kuvio 9). Työssä käytettiin kaikkia liitännäisen suosittelemia asetuksia. Lisäksi Unityn projektiasetuksista projektiympäristö määritettiin käyttämään OpenVR Loader -liitännäistä sekä alustamaan XR-asetukset Unity-projektin avautuessa (ks. Kuvio 10).

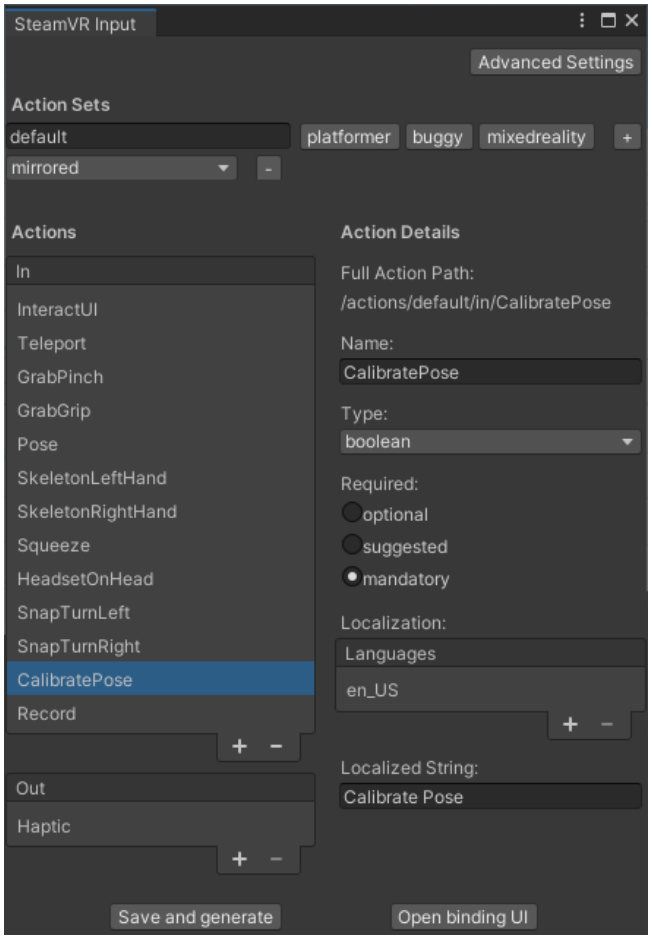

Kuvio 9. SteamVR-liitännäisen suositellut asetukset.




Kuvio 10. Unity-projektin XR-asetusikkuna.

### 3.2.2 SteamVR-ohjaintoiminnot

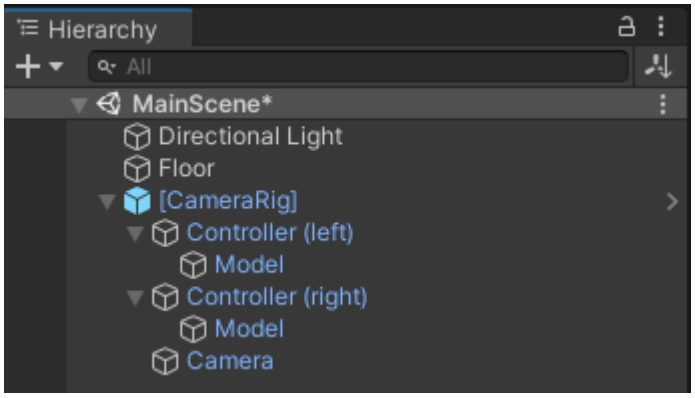
SteamVR-laitteiden käyttöä varten Unity-projektissa tulee määritellä ohjaintoiminnot SteamVR-liitännäisen SteamVR Input -asetuksista. Ohjaintoiminnot ovat SteamVR-rajapinnan laitteiden toimintoihin, kuten painikkeiden painalluksiin tai kosketuspinnan koskettamiseen liitettyjä tapahtumia. Ohjaintoimintojen avulla Unity-projektin toiminnallisuudet ja SteamVR-rajapinnan laitteet voidaan erottaa toisistaan, mikä mahdollistaa esimerkiksi usean erilaisen ohjaimen käyttämisen sekä toiminnallisuuksien painikkeiden vaihtamisen. Jos ohjaintoimintoja ei ole ennestään määritelty, SteamVR-liitännäinen pyytää käyttämään valmiita oletustoimintoja. Projektissa käytettiin ohjaintoimintojen pohjana liitännäisen oletustoimintoja, joiden lisäksi toimintoihin lisättiin Calibrate-Pose- ja Record-toiminnot laitteiden kalibrointia ja liikkeen tallentamista varten (ks. Kuvio 11).




Kuvio 11. SteamVR Input -ikkuna.

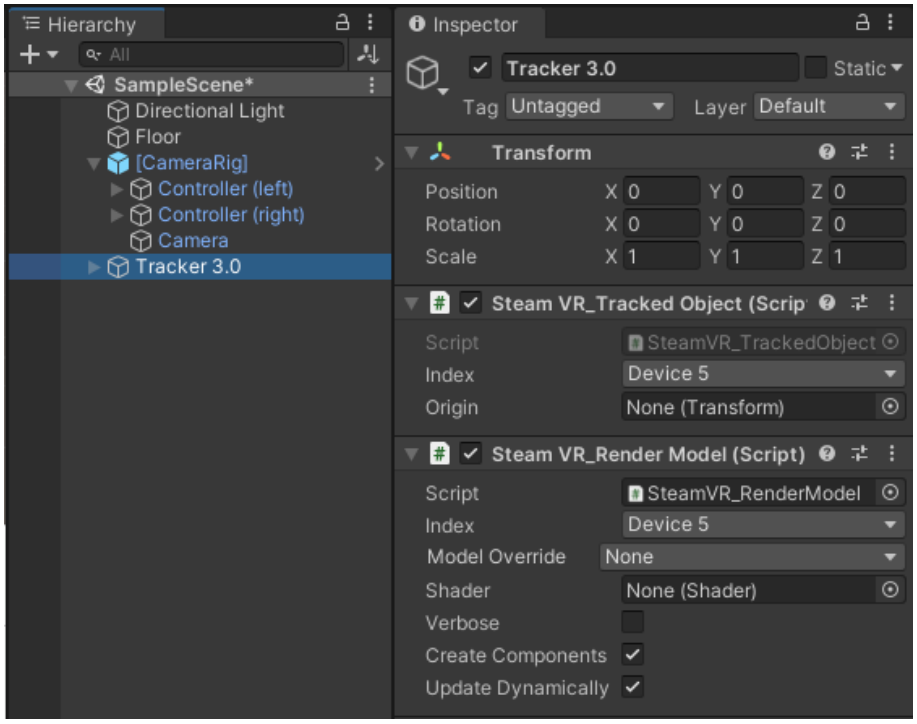
### 3.2.3 SteamVR-komponentit

SteamVR-liitännäinen sisältää monia valmiita komponentteja ja objekteja, jotka helpottavat SteamVR-laitteiden käyttöä ja hallinnointia Unity-projektissa. Unity-projektissa on valmiina kameraobjekti, mutta sitä ei voida käyttää virtuaalitodellisuusjärjestelmän kanssa suoraan. Työn projektiympäristössä oletuskameraobjektin tilalle vaihdettiin SteamVR-liitännäisen [CameraRig]-objekti, mikä mahdollistaa virtuaalitodellisuusjärjestelmän käyttämisen projektiympäristössä. [CameraRig]-prefab objekti sisältää valmiit komponentit, asetukset sekä kolmiulotteiset mallit kahdelle ohjaimelle sekä virtuaalitodellisuuslaseille. Kuvio 12 havainnollistaa objektien hierarkian Unityn hierarkianäkymässä. Kameraobjekti mahdollistaa virtuaalitodellisuuslasien sekä ohjainten paikantamisen projektin ollessa pelitilassa, mutta objekti ei sisällä seurantakomponentteja lisäseurantalaitteille. Lisäseurantalaitteille jouduttiin luomaan omat objektit.




Kuvio 12. SteamVR-liitännäisen kameraobjektin objektit Unityn hierarkianäkymässä.

Lisäseurantalaitteita varten luotiin kullekin laitteelle oma objekti. Objekteihin lisättiin SteamVR-liitännäisen mukana tulleet SteamVR_TrackedObject- sekä SteamVR_RenderModel-komponentit (ks. Kuvio 13). SteamVR_TrackedObject-komponentti mahdollistaa lisäseurantalaitteen paikantamisen ja seuraamisen pelimaailmassa. SteamVR_RenderModel-komponentti ei ole pakollinen seurannan toimivuuden kannalta, mutta komponentti lisää laitteesta tarkan kolmiulotteisen mallin virtuaaliympäristöön, mikä mahdollistaa laitteen visualisoinnin virtuaaliympäristössä. Molempiin komponentteihin tulee valita Index-laitenumero seurattavan laitteen mukaan. Laitenumerot menevät SteamVR-rajapinnan laitteiden järjestyksen mukaisesti, joten ohjaimet sekä lisäseurantalaitteet tulee käynnistää aina samassa järjestyksessä, että laitteen laitenumero pysyy samana. Työssä käytettiin neljää lisäseurantalaitetta, joten objekteja luotiin projektiympäristöön neljä kappaletta.




Kuvio 13. Lisäseurantalaitteen komponentit Unity-ympäristössä.

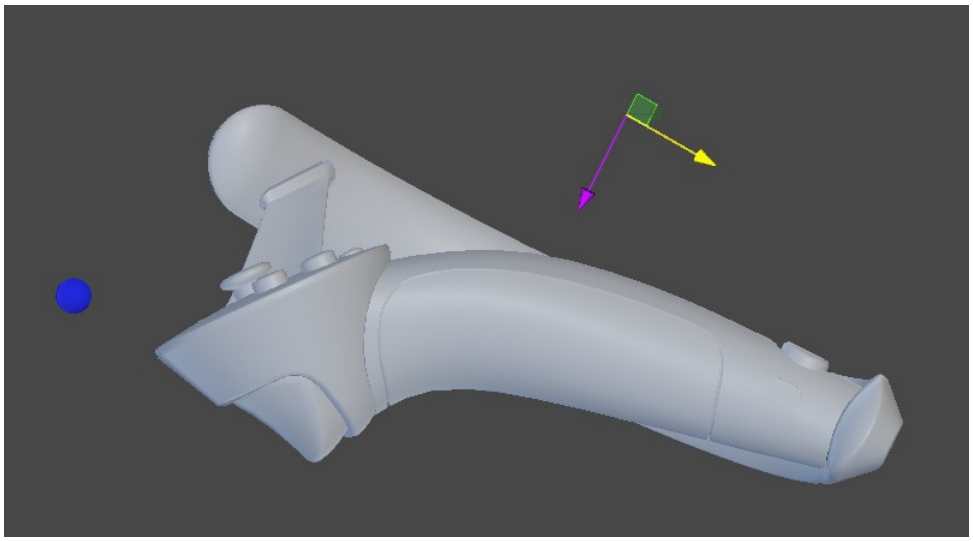
## 3.3 Virtuaalitodellisuuslaitteiden liikkeenkaappaus

### 3.3.1 Untiy-objektit

Virtuaalitodellisuuslaitteiden liikkeenkaappausta varten Unity-projektiin lisättiin ”Mocap” niminen objekti, jonka lapsiobjekteiksi luotiin oma objekti jokaista kaapattavaa laitetta kohden. Virtuaalitodellisuusohjaimia vastaaviin objekteihin lisättiin myös lapsiobjekteiksi ”Offset” nimiset objektit. Kuvio 14 näyttää Mocap-objektin hierarkialistan kokonaisuudessaan.


Kuvio 14. Mocap-objektin objektihierarkia.

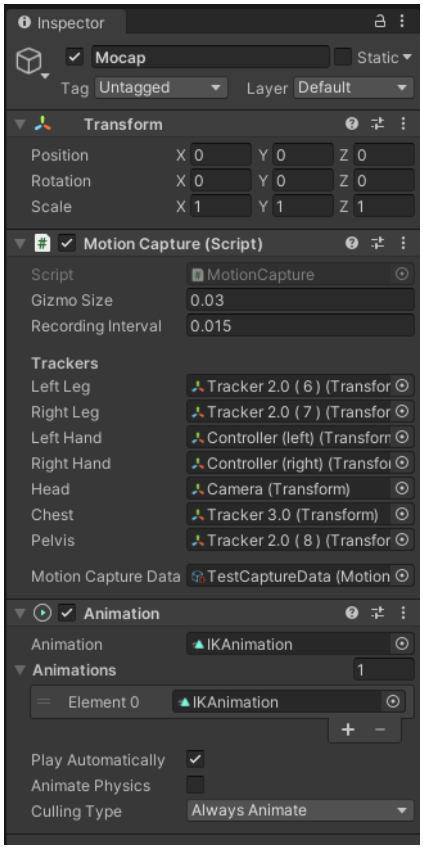
Ohjainobjekteihin lisättyjen Offset-objektien tarkoitus on osoittaa käyttäjän käden sijainti suhteessa ohjainobjektin sijaintiin. Kuvio 15 havainnollistaa kolmiulotteisen mallin avulla, kuinka erillään käden sijainti todellisuudessa on ohjaimen sijaintiin nähden. Kuvion sininen ympyrä esittää ohjaimen objektin keskipistettä, ja nuolet Offset-objektin sijaintia. Punainen nuoli osoittaa objektin Z-akselin suuntaan ja keltainen X-akselin suuntaan. Offset-objektin sijaintia käytettiin määriteltäessä pelihahmon käsien kohdesijaintia käänteiskinematiikkaa varten.



Kuvio 15. Virtuaalitodellisuusohjaimen kalibrointi Unity-projektissa.

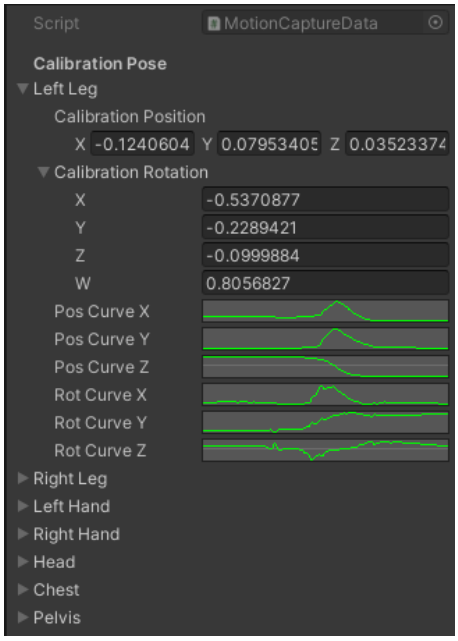
### 3.3.2 Untiy-komponentit

Virtuaalitodellisuuslaitteiden liikkeiden kaappaaminen toteutettiin Mocap-objektiin kiinnitetyn MotionCapture-komponentin avulla (ks. Kuvio 16). Mocap-objektiin lisättiin myös Unityn Animation-komponentti, jolla voitiin animoida Mocap-objektin lapsiobjektit tallennetun liikedatan mukaisesti.




Kuvio 16. Mocap-objektin komponentit.

MotionCapture -komponenttia varten luotiin ”MotionCaptureData” niminen skriptattava objekti, mihin voitiin tallentaa liikedataa virtuaalitodellisuuslaitteista. MotionCaptureData-objektit sisältävät laitteiden kalibrointipaikat ja -suunnan sekä tallennetut paikka- ja suunta-animaatiokäyrät. (ks. Kuvio 17 ja Kuvio 18).




Kuvio 17. MotionCaptureData-objektin hallintanäkymä.

```
public class MotionCaptureData : ScriptableObject
{
    [System.Serializable]
    public class MotionCaptureDataObject
    {
        public Vector3 CalibrationPosition;
        public Quaternion CalibrationRotation;
        public AnimationCurve
            PosCurveX, PosCurveY, PosCurveZ,
            RotCurveX, RotCurveY, RotCurveZ; // Euler

        /// <summary> Adds keyframes to AnimationCurves
        public void AddKeyframe(float time, Vector3 pos, Quaternion rot)...

        /// <summary> Clears AnimationCurves
        public void Clear()...
    }

    [Header("Calibration Pose")]
    public MotionCaptureDataObject LeftLeg;     public MotionCaptureDataObject RightLeg;
    public MotionCaptureDataObject LeftHand;    public MotionCaptureDataObject RightHand;
    public MotionCaptureDataObject Head;        public MotionCaptureDataObject Chest;
    public MotionCaptureDataObject Pelvis;
```

Kuvio 18. MotionCaptureData ja MotionCaptureDataObject luokat.

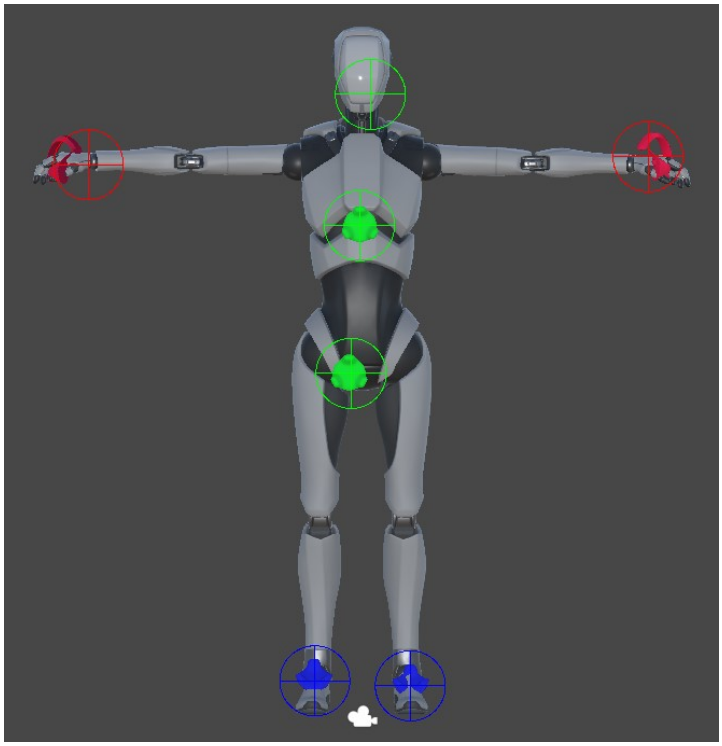
### 3.3.3 Virtuaalitodellisuuslaitteiden kalibrointi

Virtuaalitodellisuuslaitteiden sijaintia ja suuntaa kaapattaessa tulee ottaa huomioon, että laitteet ovat jokaisella käyttökerralla eri asennossa ja kohdassa käyttäjän kehossa. Laitteisto tulee siis kalibroida ennen liikkeen tallentamista, että tallennettu liikedata olisi käyttökelpoista. Työssä kalibrointi toteutettiin MotionCapture-komponentilla. Komponentti tallentaa laitteiden sijainnit ja

suunnat skriptattavaan MotionCaptureData-objektiin, käyttäjän painaessa SteamVR-rajapintaan aiemmin lisätyn CalibratePose-toiminnon painiketta virtuaalitodellisuusohjaimesta Unityn ollessa pelitilassa (ks. Kuvio 19). Laitteita kalibroitaessa käyttäjän tulee seistä T-asennossa ja asettaa seurattavat laitteet Kuvion 20 mukaisesti. Tallennetuilla kalibrointiarvoilla lasketaan myöhemmin, kuinka paljon laitteet ovat liikkuneet kalibrointitilasta.

```
23      private void Update()
24      {
25          // Press (B)-button to calibrate pose
26          if(SteamVR_Input.GetStateDown("CalibratePose", SteamVR_Input_Sources.Any, true))
27          {
28              MotionCaptureData.LeftLeg.CalibrationPosition = LeftLeg != null ? LeftLeg.position : Vector3.zero;
29              MotionCaptureData.RightLeg.CalibrationPosition = RightLeg != null ? RightLeg.position : Vector3.zero;
30              MotionCaptureData.LeftHand.CalibrationPosition = LeftHand != null ? LeftHand.position : Vector3.zero;
31              MotionCaptureData.RightHand.CalibrationPosition = RightHand != null ? RightHand.position : Vector3.zero;
32              MotionCaptureData.Head.CalibrationPosition = Head != null ? Head.position : Vector3.zero;
33              MotionCaptureData.Chest.CalibrationPosition = Chest != null ? Chest.position : Vector3.zero;
34              MotionCaptureData.Pelvis.CalibrationPosition = Pelvis != null ? Pelvis.position : Vector3.zero;
35              MotionCaptureData.LeftLeg.CalibrationRotation = LeftLeg != null ? LeftLeg.rotation : Quaternion.identity;
36              MotionCaptureData.RightLeg.CalibrationRotation = RightLeg != null ? RightLeg.rotation : Quaternion.identity;
37              MotionCaptureData.LeftHand.CalibrationRotation = LeftHand != null ? LeftHand.rotation : Quaternion.identity;
38              MotionCaptureData.RightHand.CalibrationRotation = RightHand != null ? RightHand.rotation : Quaternion.identity;
39              MotionCaptureData.Head.CalibrationRotation = Head != null ? Head.rotation : Quaternion.identity;
40              MotionCaptureData.Chest.CalibrationRotation = Chest != null ? Chest.rotation : Quaternion.identity;
41              MotionCaptureData.Pelvis.CalibrationRotation = Pelvis != null ? Pelvis.rotation : Quaternion.identity;
42
43              // Save scriptable object
44              EditorUtility.SetDirty(MotionCaptureData);
45              AssetDatabase.SaveAssets();
46              Debug.Log("Pose Calibrated");
47          }
```

Kuvio 19. Virtuaalitodellisuuslaitteiden kalibrointi MotionCapture-komponentissa.



Kuvio 20. Virtuaalitodellisuuslaitteston kalibroinnin T-asento.

### 3.3.4 Virtuaalitodellisuuslaitteiden liikkeiden tallentaminen animaatiokäyriksi

Virtuaalitodellisuuslaitteiden paikan ja suunnan tallentaminen toteutettiin MotionCapture-komponentilla. Pelitilan ollessa päällä, MotionCapture-komponentti aloittaa ja pysäyttää laitteiden liikkeiden tallentamisen käyttäjän painaessa virtuaalitodellisuusohjaimeen määriteltyä Record-toiminnon painiketta (ks. Kuvio 21). Käyttäjän käynnistäessä tallentamisen, MotionCapture-komponentti käynnistää RecordingTick-vuorottaisrutiinin (ks. Kuvio 22), joka lisää paikka- ja suuntamerkinnän tallennettaville laitteille MotionCaptureData-objektiin laitteille määritellyille animaatiokäyrille komponenttiin asetetun aikavälin mukaisesti (ks. Kuvio 24).

```
49          // Press (A)-button to start and stop recording
50          if (SteamVR_Input.GetStateDown("Record", SteamVR_Input_Sources.Any, true))
51          {
52              if (!isRecording) { StartCoroutine(RecordingTick()); }
53              else
54              {
55                  // Stop recording
56                  isRecording = false;
57                  StopAllCoroutines();
58
59                  // Save scriptable object
60                  EditorUtility.SetDirty(MotionCaptureData);
61                  AssetDatabase.SaveAssets();
62
63                  // Convert animation curves to animation clip
64                  AnimationClip animationClip = GetComponent<Animation>().clip;
65                  MotionCaptureData.ConvertToAnimationClip(animationClip);
66
67                  // Play recorded animation
68                  Animation animation = GetComponent<Animation>();
69                  animation.AddClip(animationClip, animationClip.name);
70                  animation.Play(animationClip.name);
71
72                  // Set IK active
73                  IK ik = FindObjectOfType<IK>();
74                  ik.SetPosRotOffsets(MotionCaptureData);
75                  ik.enabled = true;
76              }
```

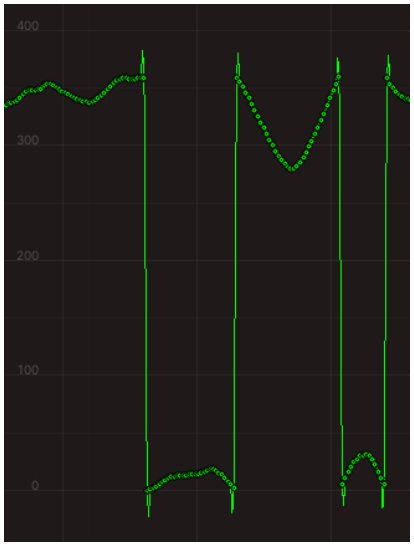
Kuvio 21. Laitteiden liikkeiden tallentamisen aloittaminen ja lopettaminen MotionCapture-komponentilla.

```
80      /// <summary> Coroutine that records IK positions to motion capture data object
83      private IEnumerator RecordingTick()
84      {
85          Debug.Log("REC STARTED");
86          float recordingStartTime = Time.time;
87          MotionCaptureData.Clear();
88          isRecording = true;
89
90          while (isRecording)
91          {
92              float timestamp = Time.time - recordingStartTime;
93              if (LeftLeg) MotionCaptureData.LeftLeg.AddKeyframe(timestamp, LeftLeg.localPosition, LeftLeg.localRotation);
94              if (RightLeg) MotionCaptureData.RightLeg.AddKeyframe(timestamp, RightLeg.localPosition, RightLeg.localRotation);
95              if (LeftHand) MotionCaptureData.LeftHand.AddKeyframe(timestamp, LeftHand.localPosition, LeftHand.localRotation);
96              if (RightHand) MotionCaptureData.RightHand.AddKeyframe(timestamp, RightHand.localPosition, RightHand.localRotation);
97              if (Head) MotionCaptureData.Head.AddKeyframe(timestamp, Head.localPosition, Head.localRotation);
98              if (Chest) MotionCaptureData.Chest.AddKeyframe(timestamp, Chest.localPosition, Chest.localRotation);
99              if (Pelvis) MotionCaptureData.Pelvis.AddKeyframe(timestamp, Pelvis.localPosition, Pelvis.localRotation);
100             yield return new WaitForSecondsRealtime(recordingInterval);
101         }
102         Debug.Log("REC ENDED");
103     }
```

Kuvio 22. RecordingTick-vuorottaisrutiini MotionCapture-komponentissa.

Laitteiden suunta-arvoja tallennettaessa tulee ottaa huomioon, että Unity käyttää kvaternioita muutettaessa kolmiulotteisiksi vektoreiksi asteiden arvoväliä 0 – 360. Kvaternioista muutettuja arvoja ei voida käyttää suoraan animaatiokäyrissä, sillä käyrien arvoja interpoloitaessa joidenkin pis-

teiden välit eivät ole tallenteiden mukaisia. Kuvio 23 näyttää kuinka muutetut suunta-arvot hyppivät arvovälin päädyissä, aiheuttaen suuren vääristymän käyrään, mikä aiheuttaa liikkeen tallenteessa tärinää. Työssä ongelma ratkaistiin lisäämällä tai vähentämällä tallennettavasta arvosta 360 astetta, jos tallennettavan arvon ero edelliseen arvoon oli yli 180 astetta (ks. Kuvio 24).




Kuvio 23. Liikkeen tallenne Unityn animaatiokäyränä.

```
15      /// <summary> Adds keyframes to AnimationCurves
18      public void AddKeyframe(float time, Vector3 pos, Quaternion rot)
19      {
20          float
21              rotX = rot.eulerAngles.x,
22              rotY = rot.eulerAngles.y,
23              rotZ = rot.eulerAngles.z;
24
25          // Move keyframe up or down if it is outside the default 360 degree range.
26          //  (if difference from last frame is over 180 degrees)
27          if(RotCurveX.keys.Length >= 1)
28          {
29              rotX = Mathf.Abs(RotCurveX.keys[RotCurveX.length - 1].value - rotX) > 180 ?
30                  ((RotCurveX.keys[RotCurveX.length - 1].value > rotX) ? rotX + 360 : rotX - 360) : rotX;
31              rotY = Mathf.Abs(RotCurveY.keys[RotCurveY.length - 1].value - rotY) > 180 ?
32                  ((RotCurveY.keys[RotCurveY.length - 1].value > rotY) ? rotY + 360 : rotY - 360) : rotY;
33              rotZ = Mathf.Abs(RotCurveZ.keys[RotCurveZ.length - 1].value - rotZ) > 180 ?
34                  ((RotCurveZ.keys[RotCurveZ.length - 1].value > rotZ) ? rotZ + 360 : rotZ - 360) : rotZ;
35          }
36
37          PosCurveX.AddKey(time, pos.x);  PosCurveY.AddKey(time, pos.y);
38          PosCurveZ.AddKey(time, pos.z);  RotCurveX.AddKey(time, rotX);
39          RotCurveY.AddKey(time, rotY);   RotCurveZ.AddKey(time, rotZ);
40      }
```

Kuvio 24. MotionCaptureDataObject-luokan AddKeyframe-funktio.

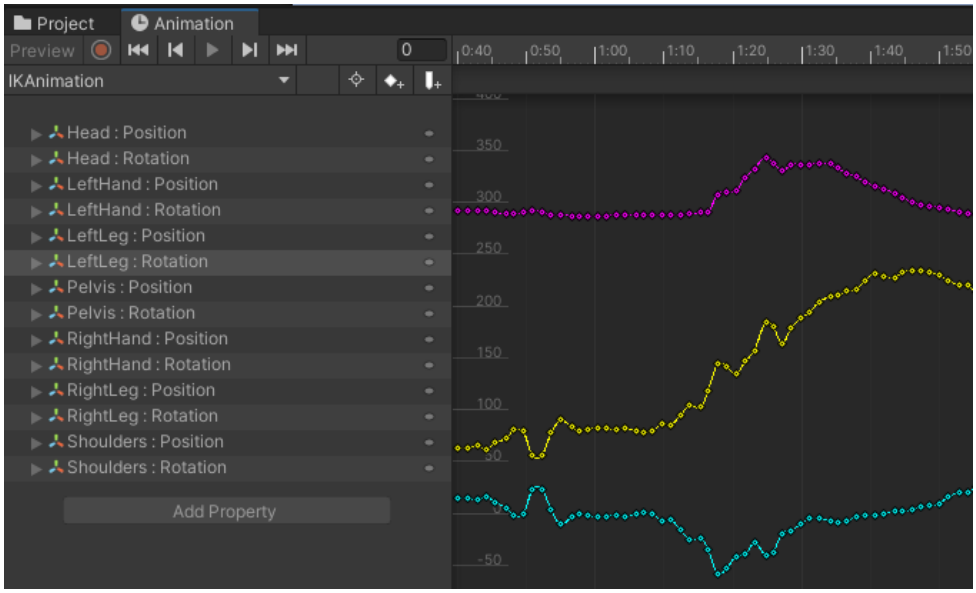
### 3.3.5 Virtuaalitodellisuuslaitteen liikkeen tallentaminen animaatiotiedostoon

Virtuaalitodellisuuslaitteiden liikkeistä tallennetut animaatiokäyrät haluttiin saada toistettavaan muotoon, että käyriä voitaisiin käyttää pelihahmon animoimiseen myöhemmässä vaiheessa. Animaatiokäyristä päätettiin luoda Unityn animaatiotiedosto. MotionCapture-komponentti tallentaa animaatiokäyrät tiedostoon samalla kun käyttäjä pysäyttää laitteiden liikkeiden tallentamisen painamalla Record-toiminnon painiketta (ks. Kuvio 21). Tallentamiseen käytettävä animaatiotiedosto tulee olla asetettuna Mocap-objektin Animation-komponentissa. Tallennetut animaatiokäyrät muunnetaan Unityn animaatioformaattiin MotionCaptureData-luokassa (ks. Kuvio 25). Unityn animaatioformaatista käytetään vanhaa Legacy-mallia, sillä animaatiolla ei ole tarkoitus animoida vielä tässä vaiheessa pelihahmoa, vaan Mocap-objektin lapsiobjektit. Animaatioon lisätään kaikkien laitteiden paikka ja suuntakäyrät käyttäen AnimationClip-luokan SetCurve-funktiota (ks. Kuvio 25).

```
76      /// <summary> Converts MotionCaptureData to AnimationClip
79      public void ConvertToAnimationClip(AnimationClip clip)
80      {
81          clip.legacy = true;
82          clip.ClearCurves();
83          SetAnimationCurveToClip("RightHand", RightHand, clip);
84          SetAnimationCurveToClip("LeftHand", LeftHand, clip);
85          SetAnimationCurveToClip("RightLeg", RightLeg, clip);
86          SetAnimationCurveToClip("LeftLeg", LeftLeg, clip);
87          SetAnimationCurveToClip("Pelvis", Pelvis, clip);
88          SetAnimationCurveToClip("Shoulders", Chest, clip);
89          SetAnimationCurveToClip("Head", Head, clip);
90      }
91
92      /// <summary> Adds position and rotation curves from MotionCaptureDataObject to  ...
95      static void SetAnimationCurveToClip(string name, MotionCaptureDataObject captureDataObject, AnimationClip clip)
96      {
97          clip.SetCurve(name, typeof(Transform), "localPosition.x", captureDataObject.PosCurveX);
98          clip.SetCurve(name, typeof(Transform), "localPosition.y", captureDataObject.PosCurveY);
99          clip.SetCurve(name, typeof(Transform), "localPosition.z", captureDataObject.PosCurveZ);
100         clip.SetCurve(name, typeof(Transform), "localEulerAnglesRaw.x", captureDataObject.RotCurveX);
101         clip.SetCurve(name, typeof(Transform), "localEulerAnglesRaw.y", captureDataObject.RotCurveY);
102         clip.SetCurve(name, typeof(Transform), "localEulerAnglesRaw.z", captureDataObject.RotCurveZ);
103     }
104 }
```

Kuvio 25. ConvertToAnimationClip ja SetAnimationCurveToClip funktiot MotionCaptureData luokassa.

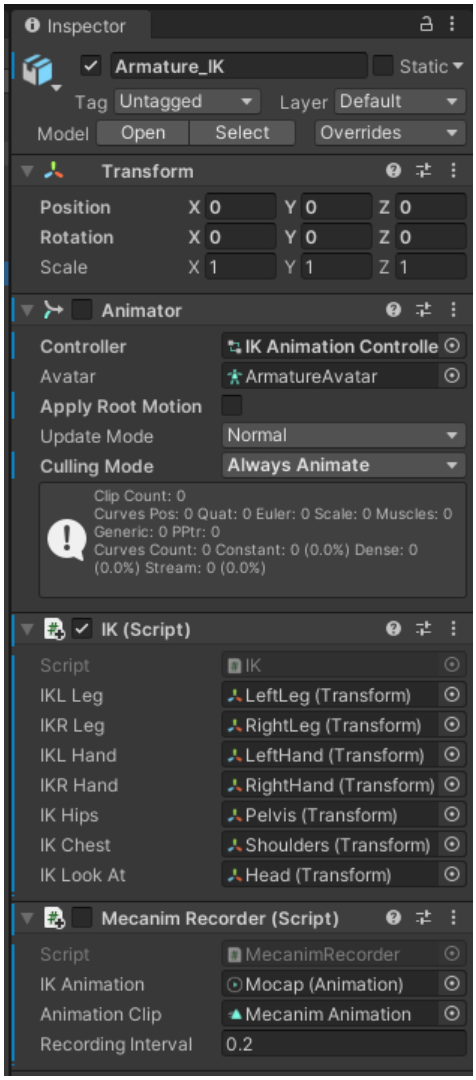
Kuvio 26 näyttää tallennetun animaation virtuaalitodellisuuslaitteiden liikkeistä Unityn animaationäkymässä. Animaatiota toistettaessa Mocap-objektin Animation-komponentilla Mocap-objektin lapsiobjektit liikkuvat animaation mukaisesti, jolloin niiden liikettä voidaan käyttää pelihahmon liikuttamiseen käänteiskinematiikkaa käyttäen.




Kuvio 26. Virtuaalitodellisuuslaitteiden tallenne Unityn animaationäkymässä.

## 3.4 Pelihahmon animointi virtuaalitodellisuuslaitteilla

Pelihahmon animointi virtuaalitodellisuuslaitteilla toteutettiin Unityn Mecanim-animaatiojärjestelmän käänteiskinematiikkaa käyttäen. Käänteiskinematiikan kohteina käytettiin Mocap-objektin lapsiobjekteja, jotka animoitiin liikkumaan virtuaalitodellisuuslaitteilla tallennetun animaation mukaisesti edellisessä luvussa. Pelihahmona työssä käytettiin Unityn julkaisemaa Mecanim-yhteensopivaa robottihahmoa, joka oli saatavilla ilmaiseksi Unityn resurssikaupasta. Pelihahmon objektiin luotiin ”IK” niminen komponentti, jolla toteutettiin hahmon käänteiskinematiikan toiminnallisuus, sekä komponentti ”MecanimRecorder”, jolla toteutettiin pelihahmon liikkeiden tallentaminen Mecanim-järjestelmään yhteensopivaksi animaatioksi (ks. Kuvio 27).




Kuvio 27. Pelihahmon komponentit.

### 3.4.1 Pelihahmon liikuttaminen käänteiskinematiikalla

Pelihahmon käänteiskinematiikka toteutettiin Unityn Mecanim-järjestelmän avulla. Pelihahmon Animator-komponenttiin lisättiin AnimatorController-resurssi, jonka päätasolle sallittiin käänteiskinematiikan käyttö. Käänteiskinematiikkaa hallinoitiin hahmoon kiinnitetyllä IK-komponentilla.

Käänteiskinematiikan kohteiden paikkoja ja suuntia laskiessa tulee ottaa huomioon käytettävän pelihahmon luiden sekä tallennettujen laitteiden liikkeiden väliset paikkojen ja suuntien poikkeamat, sillä pelihahmon luiden sijainnit ja suunnat eivät vastaa käytettyjen laitteiden sijainteja ja suuntia. Poikkeamat laskettiin IK-komponentin SetPosRotOffsets-funktiossa (ks. Kuvio 28). Poik-

keamat saatiin vertailemalla aiemmin tallennettuja laitteiden kalibrointitietoja MotionCaptureData-objektista pelihahmon luihin. Käsien sijaintipoikkeama saatiin vertailemalla käyttäjän ja pelihahmon käsien välistä leveyttä. Käsien sijaintipoikkeamiin jouduttiin ottamaan huomioon myös laitteen virtuaalisen sijainnin ja luvussa 3.3.1 mainitun käyttäjän käden sijainnin poikkeama.

```
/// <summary> Sets position and rotation offsets from a motion capture data obje ...
public void SetPosRotOffsets(MotionCaptureData data)
{
    if(data == null) { return; }

    // From bone to tracker
    hipsPosOffset = data.Pelvis.CalibrationPosition - animator.GetBoneTransform(HumanBodyBones.Hips).position;
    rLegPosOffset = data.RightLeg.CalibrationPosition - animator.GetBoneTransform(HumanBodyBones.RightFoot).position;
    lLegPosOffset = data.LeftLeg.CalibrationPosition - animator.GetBoneTransform(HumanBodyBones.LeftFoot).position;

    // Vector between hand ik points with controller grip offset
    Vector3 ikHandVector = (data.RightHand.CalibrationPosition +
        data.RightHand.CalibrationRotation * IKRHand.GetChild(0).localPosition) -
        (data.LeftHand.CalibrationPosition + data.LeftHand.CalibrationRotation * IKLHand.GetChild(0).localPosition);
    Vector3 charHandVector = animator.GetBoneTransform(HumanBodyBones.RightHand).position -
        animator.GetBoneTransform(HumanBodyBones.LeftHand).position;
    float handLengthDifference = (charHandVector.magnitude - ikHandVector.magnitude) / 2;
    rHandPosOffset = ikHandVector.normalized * handLengthDifference;
    lHandPosOffset = -ikHandVector.normalized * handLengthDifference;

    hipsRotOffset = Quaternion.Inverse(data.Pelvis.CalibrationRotation);
    rLegRotOffset = Quaternion.Inverse(data.RightLeg.CalibrationRotation);
    lLegRotOffset = Quaternion.Inverse(data.LeftLeg.CalibrationRotation);
    rHandRotOffset = Quaternion.Inverse(data.RightHand.CalibrationRotation);
    lHandRotOffset = Quaternion.Inverse(data.LeftHand.CalibrationRotation);
    chestRotOffset = Quaternion.Inverse(data.Chest.CalibrationRotation);
    headRotOffset = Quaternion.Inverse(data.Head.CalibrationRotation);
}
```

Kuvio 28. SetPosRotOffsets-funktio IK-komponentissa.

Käänteiskinematiikka toteutettiin kahdessa osassa. Pelihahmon käsien, jalkojen, selän sekä pään liikuttaminen toteutettiin Mecanim-järjestelmän käyttämässä OnAnimatorIK-funktiossa (ks. Kuvio 29), minkä jälkeen hahmon lantion sijainti asetettiin Unityn LateUpdate-funktiossa (ks. Kuvio 30). Käänteiskinematiikka jouduttiin toteuttamaan kahdessa osassa, koska Unityn käänteiskinematiikka ei tue lantion sijainnin muuttamista OnAnimatorIK-funktiossa.

Hahmon jalkojen ja käsien käänteiskinematiikan kohdesijainnit saatiin yhdistämällä animoidun laiteobjektin sijainti ja laitteiden ja luiden välinen kalibrointipoikkeama. Kohdesijainnissa tulee ottaa myös huomioon lantion liike, sillä Mecanimin käänteiskinematiikkajärjestelmä ei ota huomioon LateUpdate-funktiossa myöhemmin tehtäviä muutoksia. Hahmon selkärangan ja pään kääntämiseen ei käytetty käänteiskinematiikkaa, mutta luiden kääntäminen toteutettiin jalkojen ja käsien

kanssa OnAnimatorIK-funktiossa. Lantion, selän sekä pään käännös saatiin selville yhdistämällä animoidun laiteobjektin suunta luun kalibrointisuuntaan, jolloin pelihahmon luut kääntyvät vain laitteen käännöspoikkeaman verran. Kääntäessä hahmon selkää ja päätä tulee kuitenkin ottaa huomioon, että lantion käännös vaikuttaa myös selän suuntaan, sekä selän käännös pään suuntaan, joten selän käännökseen tulee lisätä lantion käännöksen vastakäännös, sekä pään käännökseen selän vastakäännös. Käänteiskinematiikan kohteet lisättiin pelihahmon Animator-komponentin SetIKPosition- sekä SetIKRotation-funktioilla, ja muiden luiden suunnat asetettiin Animator-komponentin SetBoneLocalRotation-funktiolla.

```
private void OnAnimatorIK(int layerIndex)
{
    Vector3 hipPos = animator.GetBoneTransform(HumanBodyBones.Hips).position;
    Vector3 rotatedHipOffset = IKHips.rotation * hipsRotOffset * hipsPosOffset;
    Vector3 hipMovement = IKHips.position - hipPos;
    Vector3 hipsIKOffset = hipMovement - rotatedHipOffset;

    animator.SetBoneLocalRotation(HumanBodyBones.Hips, IKHips.rotation * hipsRotOffset);
    animator.SetBoneLocalRotation(HumanBodyBones.Spine, Quaternion.Inverse(IKHips.rotation * hipsRotOffset) *
        IKChest.rotation * chestRotOffset);
    animator.SetBoneLocalRotation(HumanBodyBones.Head, Quaternion.Inverse(IKChest.rotation * chestRotOffset) *
        IKLookAt.rotation * headRotOffset);

    Vector3 rLegTrackerOffset = IKRLeg.rotation * rLegRotOffset * rLegPosOffset;
    Vector3 lLegTrackerOffset = IKLLeg.rotation * lLegRotOffset * lLegPosOffset;
    animator.SetIKPosition(AvatarIKGoal.RightFoot, IKRLeg.position - hipsIKOffset - rLegTrackerOffset);
    animator.SetIKRotation(AvatarIKGoal.RightFoot, IKRLeg.rotation * rLegRotOffset);
    animator.SetIKPosition(AvatarIKGoal.LeftFoot, IKLLeg.position - hipsIKOffset - lLegTrackerOffset);
    animator.SetIKRotation(AvatarIKGoal.LeftFoot, IKLLeg.rotation * lLegRotOffset);

    Vector3 rHandTrackerOffset = IKRHand.rotation * rHandRotOffset * rHandPosOffset;
    Vector3 lHandTrackerOffset = IKLHand.rotation * lHandRotOffset * lHandPosOffset;
    animator.SetIKPosition(AvatarIKGoal.RightHand, IKRHand.GetChild(0).position - hipsIKOffset + rHandTrackerOffset);
    animator.SetIKRotation(AvatarIKGoal.RightHand, IKRHand.GetChild(0).rotation);
    animator.SetIKPosition(AvatarIKGoal.LeftHand, IKLHand.GetChild(0).position - hipsIKOffset + lHandTrackerOffset);
    animator.SetIKRotation(AvatarIKGoal.LeftHand, IKLHand.GetChild(0).rotation);

    animator.SetIKPositionWeight(AvatarIKGoal.RightFoot, 1);
    animator.SetIKRotationWeight(AvatarIKGoal.RightFoot, 1);
    animator.SetIKPositionWeight(AvatarIKGoal.LeftFoot, 1);
    animator.SetIKRotationWeight(AvatarIKGoal.LeftFoot, 1);
    animator.SetIKPositionWeight(AvatarIKGoal.RightHand, 1);
    animator.SetIKRotationWeight(AvatarIKGoal.RightHand, 1);
    animator.SetIKPositionWeight(AvatarIKGoal.LeftHand, 1);
    animator.SetIKRotationWeight(AvatarIKGoal.LeftHand, 1);
}
```

Kuvio 29. OnAnimatorIK-funktio IK-komponentissa.

Mecanim-järjestelmä ei tue pelihahmon lantion luun liikuttamista OnAnimatorIK-funktiossa, joten lantion liike jouduttiin lisäämään hahmolle LateUpdate-funktiossa käänteiskinematiikan kohteiden asettamisen jälkeen. Lantion liike saatiin yhdistämällä animoidun laiteobjektin sijainti lantion kalibrointisijaintiin (ks. Kuvio 30).

```
private void LateUpdate()
{
    // Pelvis position
    Vector3 rotatedHipOffset = IKHips.rotation * hipsRotOffset * hipsPosOffset;
    Vector3 hipMovement = IKHips.position - animator.GetBoneTransform(HumanBodyBones.Hips).position;
    animator.GetBoneTransform(HumanBodyBones.Hips).position =
        animator.GetBoneTransform(HumanBodyBones.Hips).position + hipMovement - rotatedHipOffset;
}
```

Kuvio 30. LateUpdate-funktio IK-komponentissa.

### 3.4.2 Pelihahmon liikkeiden tallentaminen animaatioksi

Pelihahmon liike haluttiin muuntaa monimutkaisesta käänteiskinematiikkajärjestelmästä helpommin hallittavaan animaatiomuotoon, jotta animaatiota voitaisiin käyttää muiden Mecanim-animaatioiden kanssa Mecanim-järjestelmässä. Liikkeen tallentamiseen käytettiin pelihahmolle luotua MecanimRecorder-komponenttia, joka käynnistyessään käynnistää RecordingTick-vuorottaisrutiinin (ks. Kuvio 31). Pelihahmon liikkeiden tallentamiseen käytettiin Mecanim-järjestelmän animaatioissa käytettävää hahmojen lihasjärjestelmää.

RecordingTick-vuorottaisrutiini hakee pelihahmon lihastiedot hahmon avatarista käyttäen HumanPoseHandler- sekä Animator-luokkia. Määritellyn aikavälin välein vuorottaisrutiini tallentaa lihasarvot omiin animaatiokäyriin samalla kun pelihahmoa liikutetaan käänteiskinematiikalla edellisen luvun mukaisesti. Animaation loputtua vuorottaisrutiini lopettaa toimintansa ja lisää kerätyt animaatiokäyrät Mecanim-animaatioon SaveAnimation-funktiossa (ks. Kuvio 32). Funktio lisää komponenttiin liitettyyn animaatiotiedostoon animaatiokäyriä Mecanim-järjestelmän lihaksien nimiä käyttäen, jolloin Mecanim-järjestelmä animoin tallennetun animaation käyttäen hahmon lihaksia. Näin animaatiota voidaan käyttää kaikilla Mecanim-yhteensopivilla pelihahmoilla.

```
private IEnumerator RecordingTick()
{
    Debug.Log("Mecanim Recording STARTED");
    float recordingStartTime = Time.time;

    while (true)
    {
        yield return new WaitForEndOfFrame(); // Wait other scripts (IK etc.)

        float timestamp = Time.time - recordingStartTime;
        // Stop recording when the IK animation ends
        if(timestamp > IKAnimation.clip.length) { break; }

        // Get characters HumanPose
        HumanPoseHandler poseHandler = new HumanPoseHandler(GetComponent<Animator>().avatar, transform);
        HumanPose pose = new HumanPose();
        poseHandler.GetHumanPose(ref pose);

        // Add keyframes to curves
        for (int i = 0; i < pose.muscles.Length; i++)
        {
            muscleCurves[i].AddKey(timestamp, pose.muscles[i]);
        }

        motionTCurveX.AddKey(timestamp, pose.bodyPosition.x);
        motionTCurveY.AddKey(timestamp, pose.bodyPosition.y);
        motionTCurveZ.AddKey(timestamp, pose.bodyPosition.z);
        motionQCurveX.AddKey(timestamp, pose.bodyRotation.x);
        motionQCurveY.AddKey(timestamp, pose.bodyRotation.y);
        motionQCurveZ.AddKey(timestamp, pose.bodyRotation.z);
        motionQCurveW.AddKey(timestamp, pose.bodyRotation.w);

        yield return new WaitForSecondsRealtime(recordingInterval);
    }
    Debug.Log("Mecanim Recording ENDED");
    SaveAnimation();
}
```

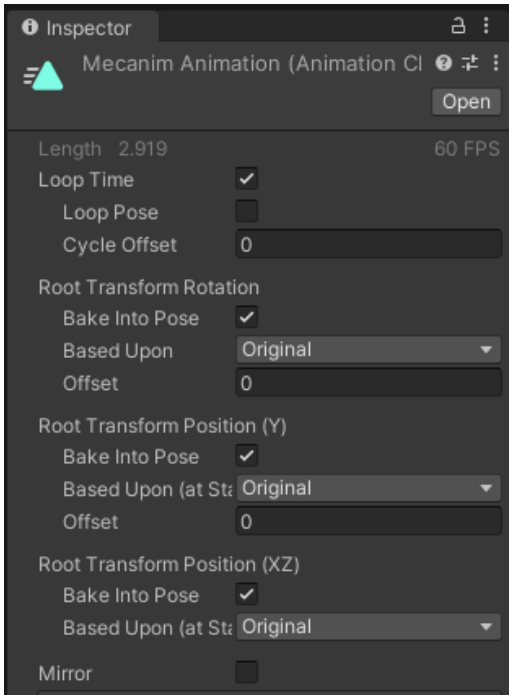
Kuvio 31. RecordingTick-vuorottaisrutiini MecanimRecorder-komponentissa.

```
private void SaveAnimation()
{
    AnimationClip.ClearCurves();

    AnimationClip.SetCurve("", typeof(Animator), "Spine Front-Back", muscleCurves[0]);
    AnimationClip.SetCurve("", typeof(Animator), "Spine Left-Right", muscleCurves[1]);
    AnimationClip.SetCurve("", typeof(Animator), "Spine Twist Left-Right", muscleCurves[2]);
    AnimationClip.SetCurve("", typeof(Animator), "Neck Nod Down-Up", muscleCurves[9]);
    AnimationClip.SetCurve("", typeof(Animator), "Neck Tilt Left-Right", muscleCurves[10]);
    AnimationClip.SetCurve("", typeof(Animator), "Neck Turn Left-Right", muscleCurves[11]);
    AnimationClip.SetCurve("", typeof(Animator), "Head Nod Down-Up", muscleCurves[12]);
    AnimationClip.SetCurve("", typeof(Animator), "Head Tilt Left-Right", muscleCurves[13]);
    AnimationClip.SetCurve("", typeof(Animator), "Head Turn Left-Right", muscleCurves[14]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Arm Down-Up", muscleCurves[39]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Arm Front-Back", muscleCurves[40]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Arm Twist In-Out", muscleCurves[41]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Forearm Stretch", muscleCurves[42]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Forearm Twist In-Out", muscleCurves[43]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Hand Down-Up", muscleCurves[44]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Hand In-Out", muscleCurves[45]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Arm Down-Up", muscleCurves[48]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Arm Front-Back", muscleCurves[49]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Arm Twist In-Out", muscleCurves[50]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Forearm Stretch", muscleCurves[51]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Forearm Twist In-Out", muscleCurves[52]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Hand Down-Up", muscleCurves[53]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Hand In-Out", muscleCurves[54]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Upper Leg Front-Back", muscleCurves[21]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Upper Leg In-Out", muscleCurves[22]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Upper Leg Twist In-Out", muscleCurves[23]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Lower Leg Stretch", muscleCurves[24]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Lower Leg Twist In-Out", muscleCurves[25]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Foot Up-Down", muscleCurves[26]);
    AnimationClip.SetCurve("", typeof(Animator), "Left Foot Twist In-Out", muscleCurves[27]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Upper Leg Front-Back", muscleCurves[29]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Upper Leg In-Out", muscleCurves[30]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Upper Leg Twist In-Out", muscleCurves[31]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Lower Leg Stretch", muscleCurves[32]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Lower Leg Twist In-Out", muscleCurves[33]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Foot Up-Down", muscleCurves[34]);
    AnimationClip.SetCurve("", typeof(Animator), "Right Foot Twist In-Out", muscleCurves[35]);
    AnimationClip.SetCurve("", typeof(Animator), "RootT.x", motionTCurveX);
    AnimationClip.SetCurve("", typeof(Animator), "RootT.y", motionTCurveY);
    AnimationClip.SetCurve("", typeof(Animator), "RootT.z", motionTCurveZ);
    AnimationClip.SetCurve("", typeof(Animator), "RootQ.x", motionQCurveX);
    AnimationClip.SetCurve("", typeof(Animator), "RootQ.y", motionQCurveY);
    AnimationClip.SetCurve("", typeof(Animator), "RootQ.z", motionQCurveZ);
    AnimationClip.SetCurve("", typeof(Animator), "RootQ.w", motionQCurveW);
}
```

Kuvio 32. SaveAnimation-funktio MecanimRecorder-komponentissa.

Tallennettu animaatio voi toistettaessa animoida pelihahmon väärään suuntaan tai paikkaan. Mecanim-yhteensopivien animaatioiden asetuksia vaihtamalla animaatiotiedoston hallintaikkunasta (ks. Kuvio 33) animaatio saadaan toistumaan tallenteiden mukaisesti.




Kuvio 33. Mecanim-animaation asetukset hallintaikkunassa.

# 4 Toteutuksen lopputulokset

Opinnäytetyön tuloksena saatiin kehitettyä Unity-projektiympäristö, missä voidaan seurata SteamVR-rajapintaa tukevia virtuaalitodellisuuslaitteita, sekä tallentaa laitteiden liikkeet Unityn animaatiokäyriksi. Tallennettujen animaatiokäyrien avulla projektiympäristössä voidaan animoida pelihahmo liikkumaan virtuaalitodellisuuslaitteiden liikkeiden mukaisesti käänteiskinematiikkaa käyttäen. Työn toteutuksessa käytettiin seitsemää seurattavaa virtuaalitodellisuuslaitetta, mikä mahdollisti pelihahmon käsien, jalkojen, lantion, selän sekä pään samanaikaisen liikuttamisen pelihahmoa animoitaessa.

Projektiympäristössä voidaan myös tallentaa pelihahmon liike Unityn animaatiotiedostomuotoon. Pelihahmon käänteiskinematiikalla luotu animaatio voidaan tallentaa Unityn Mecanim-animaatiojärjestelmän kanssa yhteensopivaan animaatiomuotoon, jolloin tallennettua animaatiota voidaan käyttää kaikilla Mecanim-järjestelmää tukevilla pelihahmoilla. Tallennettu animaatiotiedosto voidaan myös viedä projektista muihin Unity-projekteihin, ilman että projektiympäristön muita osia tarvitsee siirtää.

# 5 Pohdinta

Opinnäytetyön tavoitteena oli toteuttaa Unity-projektiympäristö, jossa voitaisiin tallentaa virtuaalitodellisuuslaitteiden liikkeet, ja tallentaa liike pelihahmon animaatioksi. Toteutuksen tarkoituksena oli selvittää, soveltuuko virtuaalitodellisuuslaitteet liikkeenkaappausjärjestelmäksi projekteihin, jotka eivät vaadi huipputason liikkeenkaappausta.

Työn projektiympäristö saatiin toteutettua tavoitteiden mukaisesti, niin että projektiympäristöä voidaan laajentaa ja muokata tulevaisuudessa, jos muutoksille on tarvetta. Projektiympäristö toteutettiin sillä periaatteella, että seurattavien virtuaalitodellisuuslaitteiden määrää olisi helppo vaihtaa, jos tulevaisuudessa käytössä olisi enemmän tai vähemmän seurattavia laitteita. Toteutetun projektiympäristön perusteella virtuaalitodellisuusjärjestelmä käyttäminen liikkeenkaappausjärjestelmänä on mahdollista, mutta käytössä tulee ottaa huomioon rajoittavia tekijöitä.

Ihmishahmon liikettä kaapatessa kaapattavia laitteita tulee olla mieluiten enemmän kuin työssä käytetyt seitsemän laitetta. Seurattavia laitteita olisi hyvä olla jalkojen ja käsien lisäksi myös polvissa sekä kyynärpäissä, sillä käänteiskinematiikalla ei voida tarkkaan määritellä, mihin suuntaan pelihahmon kädet ja jalat tulisi osoittaa. Ongelmaa voidaan korjata käyttämällä parempaa käänteiskinematiikka-algoritmia, joka ottaisi huomioon muiden luiden asennot polvien ja kyynärpäiden suuntaa määriteltäessä, mutta tarkimman liikkeenkaappauksen saisi käyttämällä fyysisiä seurantalaitteita. Virtuaalitodellisuuslaitteilla on myös vaikea kaapata pieniä yksityiskohtia, kuten käyttäjän sormien tai olkapäiden liikkeitä. Työssä käytetyillä Index-virtuaalitodellisuusohjaimilla on mahdollista tallentaa yksinkertaisia sormien liikkeitä, mutta virtuaalisten sormien liikeradat täytyisi määritellä ennen tallentamista, ja ohjainten sormitunnistus on hyvin epäluotettava liikkeenkaappausta ajatellen.

Järjestelmien ongelmista huolimatta virtuaalitodellisuusjärjestelmä soveltuu hyvin esimerkiksi mallianimaatioiden tekemiseen lopullisten animaatioiden pohjaksi, tai nopeiden animaatioiden tekemiseen prototyyppiprojekteihin, joissa animaatioiden laadulla ei ole yhtä paljon merkitystä kuin animaatioiden esittävyydellä.

# Lähteet

Angelov, V. Petkov, E. Shipkovenski, G. & Kalushkov, T. 2020. Modern Virtual Reality Headsets. International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). Viitattu 25.11.2021. https://doi.org/10.1109/HORA49412.2020.9152604

Animation System Overview. N.d. Unityn Mecanim-animaatiojärjestelmän dokumentaatio. Viitattu 31.10.2021. https://docs.unity3d.com/Manual/AnimationOverview.html

Build once, deploy anywhere. N.d. Viitattu 25.10.2021. https://unity.com/features/multiplatform

Guerra-Filho, G. 2005. Optical Motion Capture: Theory and Implementation. RITA, 12. Viitattu 7.12.2021.

Hibbitts, D. 2020. Choosing a real-time performance capture system. Viitattu 4.12.2021. https://cdn2.unrealengine.com/Unreal+Engine%2Fperformance-capture-whitepaper%2FLPC_Whitepaper_final-7f4163190d9926a15142eafcca15e8da5f4d0701.pdf

Hindle, B. Keogh, J. & Lorimer, A. 2021. Inertial-Based Human Motion Capture: A Technical Summary of Current Processing Methodologies for Spatiotemporal and Kinematic Measures. Viitattu 8.12.2021. https://doi.org/10.1155/2021/6628320

Hooker, J. 2021. XR, AR, VR, MR: What's the Difference in Reality? Viitattu 24.11.2021. https://www.arm.com/blogs/blueprint/xr-ar-vr-mr-difference

Introduction to components. N.d. Unityn komponenttien dokumentaatio. Viitattu 31.10.2021. https://docs.unity3d.com/Manual/Components.html

Inverse Kinematics. N.d. Käänteiskinematiikka MathWorksin verkkosivuilla. Viitattu 29.3.2022. https://www.mathworks.com/discovery/inverse-kinematics.html

Lowood, H. 2021. Virtual Reality. Encyclopedia Britannica 13.3.2021. Viitattu 26.10.2021. https://www.britannica.com/technology/virtual-reality

The Game view. N.d. Unity editorin pelinäkymän dokumentaatio. Viitattu 25.10.2021. https://docs.unity3d.com/Manual/GameView.html

The Hierarchy window. N.d. Unity editorin hierarkiaikkunan dokumentaatio. Viitattu 25.10.2021. https://docs.unity3d.com/Manual/Hierarchy.html

The Inspector window. N.d. Unity editorin hallintaikkunan dokumentaatio. Viitattu 25.10.2021. https://docs.unity3d.com/Manual/UsingTheInspector.html

The Project window. N.d. Unity editorin projekti-ikkunan dokumentaatio. Viitattu 25.10.2021. https://docs.unity3d.com/Manual/ProjectView.html

The Scene view. N.d. Unity editorin kohtausnäkymän dokumentaatio. Viitattu 25.10.2021. https://docs.unity3d.com/Manual/UsingTheSceneView.html

The Toolbar. N.d. Unity editorin työkalupalkin dokumentaatio. Viitattu 25.10.2021. https://docs.unity3d.com/Manual/Toolbar.html

Unity Hub. N.d. Viitattu 25.10.2021. https://unity.com/unity-hub

Unity Platform. N.d. Unity-ympäristön verkkosivut. Viitattu 25.10.2021. https://unity.com/products/unity-platform

Unity's Interface. N.d. Unity editorin dokumentaatio. Viitattu 25.10.2021. https://docs.unity3d.com/Manual/UsingTheEditor.html

Using Components. N.d. Unityn komponenttien dokumentaatio. Viitattu 31.10.2021. https://docs.unity3d.com/Manual/UsingComponents.html

Vince, J. 2004. Introduction to Virtual Reality. Viitattu 24.11.2021. Springer-Verlag London Limited.

Woodford, C. 2021. Virtual reality. Viitattu 19.11.2021. https://www.explainthatstuff.com/virtual-reality.html