OULUN AMMATTIKORKEAKOULU

Juho Kuoksa

LISÄTYN TODELLISUUDEN MOBIILISOVELLUKSEN KEHITYS UNITY-PELIMOOTTORILLA

LISÄTYN TODELLISUUDEN MOBIILISOVELLUKSEN KEHITYS UNITY-PELIMOOTTORILLA

Juho Kuoksa
Opinnäytetyö
Syksy 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä: Juho Kuoksa
Opinnäytetyön nimi: Lisätyn todellisuuden mobiilisovelluksen kehitys Unity-pelimoottorilla
Työn ohjaaja: Janne Kumpuoja
Työn valmistumislukukausi ja -vuosi: Syksy 2021
Sivumäärä: 35 + 6 liitettä

Opinnäytetyön tarkoituksena oli perehtyä lisättyyn todellisuuteen, sen historiaan ja käyttökohteisiin sekä lisätyn todellisuuden mobiilisovelluksen kehitykseen Unity-pelimoottorilla. Opinnäytetyössä käydään läpi mobiilisovelluksen tekeminen valituilla työkaluilla ja perehdytään siihen, millaisia lisätyn todellisuuden ominaisuuksia mobiililaitteilla voidaan hyödyntää. Tässä opinnäytetyössä keskitytään mobiilisovelluksen tekemiseen Android-pohjaisille laitteille, joille on olemassa omat kehitystyökalut.

Opinnäytetyössä käydään aluksi läpi lisätyn todellisuuden määritelmä ja esitellään erilaisia käyttökohteita, joissa lisättyä todellisuutta on hyödynnetty. Tämän jälkeen siirrytään lisätyn todellisuuden mobiilisovelluksen kehitykseen ja käydään läpi ominaisuuksia, joita Unity ja sen lisäosat mahdollistavat. Lopuksi opinnäytetyössä esitellään esimerkkinä tehty mobiilisovellus, jossa käytetään hyödyksi Unityn ja sen lisäosien tarvittavia ominaisuuksia, jotta lisätyn todellisuuden toiminnallisuudet saadaan toimimaan mobiilisovelluksessa.

Opinnäytetyötä tehdessä huomattiin Unityn helppokäyttöisyys mobiilisovelluksen kehityksessä. Yksinkertaisen lisätyn todellisuuden sovelluksen tekeminen Unitylla ja sen lisäosilla on tehty käyttäjälle helpoksi ja tulevaisuudessa teknologian vielä edelleen kehittyessä siitä tulee todennäköisesti vieläkin helpompaa.

Asiasanat: lisätty todellisuus, mobiiliohjelmointi, Unity

ABSTRACT

Oulu University of Applied Sciences
Information technology, Software engineering

Author: Juho Kuoksa
Title of thesis: Augmented reality mobile app development with Unity game engine
Supervisor: Janne Kumpuoja
Term and year when the thesis was submitted: Fall 2021
Number of pages: 35 + 6 appendices

The object of this thesis was to examine augmented reality, history of augmented reality and different use cases and the development of augmented reality mobile application with Unity game engine. This thesis focuses on creating an augmented reality mobile application to Android devices.

This thesis first explains the concept of augmented reality and different use cases where it is used. After that we will go through the process of creating an augmented reality mobile application with Unity game engine and go through the features that Unity and all the plug-ins allow. Finally, a demo application made with Unity is presented. The demo application uses different augmented reality features.

During the thesis work was discovered that Unity is easy to use in mobile application development. Creating a simple augmented reality mobile application with Unity game engine and its plug-ins is made easy to the user and in the future, it is probably going to get even easier when the technology around augmented reality continues to develop.

Keywords: Augmented reality, mobile development, Unity

SISÄLLYS

1 JOHDANTO

Lisätyn todellisuuden käsite eli tietokoneen luoma virtuaalinen sisältö reaalimaailmassa on ollut olemassa jo pitkän aikaa. Lisättyä todellisuutta voidaan hyödyntää nykyään lähes kaikilla älylaitteilla. Helppokäyttöisyytensä sekä saavutettavuutensa ansiosta lisätyn todellisuuden suosio onkin kasvanut viime vuosina huomattavasti eri toimialoilla.

Tämän opinnäytetyön aihe valikoitui oman kiinnostuksen sekä oppimishalun kautta. Tarkoituksena oli käydä läpi lisätyn todellisuuden erilaisia käyttökohteita sekä tarkastella, miten Unity-pelimoottorilla voidaan tehdä lisätyn todellisuuden mobiilisovelluksia. Oli myös käytävä läpi, mitä kaikkea Unityn projektilta vaaditaan, jotta lisätyn todellisuuden ominaisuudet saadaan käyttöön mobiililaitteessa.

Opinnäytetyössä esitellään aluksi lisätyn todellisuuden historia ja käydään läpi erilaisia esimerkkejä sen mahdollistamista käyttökohteista. Myöhemmin opinnäytetyössä käydään läpi miten luodaan lisätyn todellisuuden Android-mobiilisovellus Unity-pelimoottorin avulla. Lisäksi tarkastellaan lisäosia, jotka Unity-pelimoottoriin täytyi asentaa.

Opinnäytetyön lopussa esitellään esimerkkinä tehty lisätyn todellisuuden mobiilisovellus, jossa käydään läpi Unityn ja sen lisäosien käyttöä mobiilisovelluksen kehityksessä. Sovelluksessa demonstroidaan lisätyn todellisuuden mahdollisuuksia mobiilisovelluksissa.

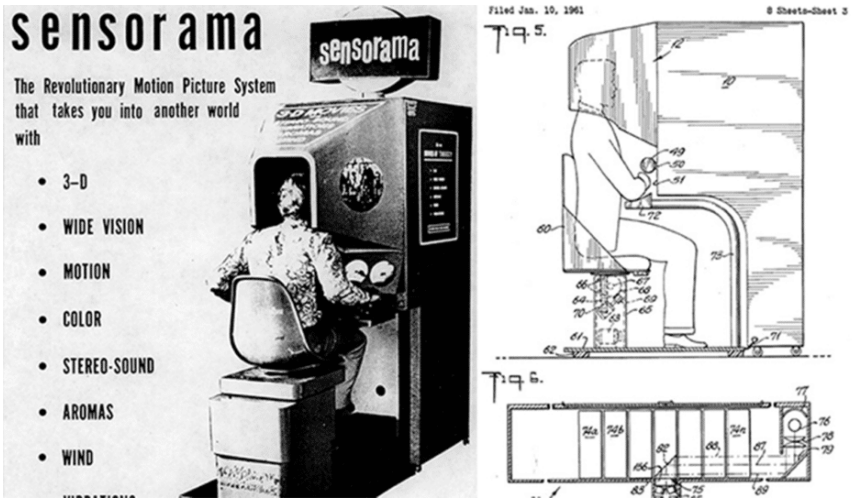
2 LISÄTTY TODELLISUUS

Lisätty todellisuus eli Augmented Reality (lyhenne AR) on interaktiivinen kokemus, jossa yhdistetään oikean maailman kuvaan tietokoneella kehitettyjä elementtejä reaaliajassa. Lisättyä todellisuutta voidaan hyödyntää erilaisilla tavoilla, kuten lisätyn todellisuuden laseilla tai älypuhelimen näytöllä kameran avulla, jotka molemmat yhdistävät omalla tavallaan käyttäjän oikean maailman ympäristöön virtuaalista grafiikkaa. (1.)

Lisätyn todellisuuden ja virtuaalitodellisuuden ero on siinä, että virtuaalitodellisuudessa luodaan kokonaan virtuaalinen maailma esimerkiksi erilaisiin virtuaalilaseihin. Käyttäjä on sijoitettu tähän luotuun ympäristöön ja voi esimerkiksi liikkua ympäriinsä ja vuorovaikuttaa täysin tietokoneella luotujen elementtien kanssa. Lisätyssä todellisuudessa käyttäjä sen sijaan on sijoitettuna oikeaan maailmaan ja tähän näkymään luodaan virtuaalisia elementtejä. Lisätyssä todellisuudessa järjestelmä ymmärtää oikeaa ympäristöä tarpeeksi hyvin, jotta voidaan luoda synteettisiä elementtejä oikean maailman asioiden päälle tai taakse. (1.)

2.1 Historia

Lisätyn todellisuuden käytön aloitus voidaan jäljittää vuoteen 1962, jolloin kuvaaja Morton Heilig teki moottoripyöräsimulaattorin nimeltään Sensorama (kuva 1). Se sisälsi kuvaa, ääntä, liikettä ja hajua. (2.)



KUVA 1. Morton Heiligin keksintö Sensorama (3)

Voidaan kuitenkin sanoa, että lisätty todellisuus ja virtuaalitodellisuus alkoivat todella muotoutua vuonna 1966, kun Harvardin yliopiston professori Ivan Sutherland kehitti ensimmäisen päähän asetettavan monitorijärjestelmän. Järjestelmä simuloi yksinkertaista tietokoneen tekemää grafiikkaa käyttäjälle sen mukaan, mihin tämä katsoi. Laite oli niin painava, että se täytyi laittaa roikkumaan laboratorion katosta. (2.)

Terminä lisätty todellisuus keksittiin vuonna 1990, kun tutkija Tom Caudellia pyydettiin kollegansa David Mizellin kanssa keksimään vaihtoehto Boeingin tehtaalle kalliille kaavioille ja merkintälaitteille, joilla ohjattiin työntekijöitä tehtaan tiloissa. He keksivät päähän asetettavan laitteen, joka heijasti uudelleen käytettäville alustoille lentokoneen tarkkoja kaavoja. Tällä järjestelmällä korvattiin suuret vanerilevyt, joissa jokaisessa oli omat yksilöllisesti suunnitellut ohjeet johdotuksien tekemiseen. Tämä teki tehtaalla työskentelystä nopeampaa, koska jokaista vanerilevyä ei enää tarvitsisi säätää käsin jokaisessa valmistusvaiheessa. (2.)

Ensimmäisenä täysin toimivana lisätyn todellisuuden järjestelmänä pidetään Louis Rosenbergin kehittelemää Virtual Fixtures -nimistä laitetta vuodelta 1992. Virtual Fixtures kehitettiin Yhdysvaltojen ilmavoimien käyttöön ja se helpotti opastuksella käyttäjän tehtäviä. (2.)

Vuoteen 1999 asti lisätty todellisuus oli lähinnä tiedemiesten ja näiden laboratorioiden käytettävissä, eivätkä kuluttajat edes tienneet lisätyn todellisuuden kentän kasvusta. Tämä kuitenkin muuttui, kun Nara Institute of Science and Technologyn Hirokazu Kato julkaisi avoimen lähdekoodin ARToolKitin. Se salli todellisen maailman videokaappauksen yhdistämisen virtuaalisten asioiden vuorovaikutukseen ja tarjosi 3D-grafiikkaa, jota voitiin käyttää millä tahansa käyttöjärjestelmällä. Tämä teki mahdolliseksi internet-yhteydellä ja kameralla varustetun kädessä pidettävän laitteen, joka toi lisätyn todellisuuden mahdolliseksi suurille yleisöille. (2.)

2.2 Lisätyn todellisuuden käyttökohteita ja esimerkkejä

Lisättyä todellisuutta hyödynnetään nykyisin monissa erilaisissa tilanteissa. Lisätyn todellisuuden helppokäyttöisyys ja kustannustehokkuus sekä sen saavutettavuus mahdollistavat erilaisille yrityksille sen hyödyntämisen jokapäiväisessä toiminnassaan. Lisätty todellisuus tarjoaa käyttäjilleen monipuolisemman käyttökokemuksen ja lisää vuorovaikutusta. (4.) Seuraavissa luvuissa perehdytään tarkemmin, miten ja missä kaikkialla lisättyä todellisuutta on hyödynnetty.

2.2.1 Myynti ja markkinointi

Lisätty todellisuus tarjoaa sekä verkossa että kivijalkamyymälöissä tehtyjen ostokokemuksien parantamiseksi suuria mahdollisuuksia. Kätevin tapa ostosten hoitamiseen on ollut kivijalkamyymälässä asiointi, jolloin asiakas on voinut kokeilla tuotetta ennen ostamista. Lisätyn todellisuuden avulla ostokset voidaan nykyisin hoitaa myös kotoa käsin, kun asiakas pääsee kokeilemaan mobiililaitteensa avulla esimerkiksi, miltä tietyt huonekalut näyttävät kodin huoneissa. Ikea on esimerkiksi suunnitellut tällaisen sovelluksen, joka antaa käyttäjän kokeilla virtuaalisesti huonekaluja tiettyyn tilaan. (5.)

Ikean lisäksi on myös muita esimerkkejä lisätyn todellisuuden mahdollisuuksista kodin parantamiseksi. Esimerkiksi yhdysvaltalainen rautakauppaketju Home Depot käyttää lisättyä todellisuutta hyödyksi. Asiakkaat voivat kokeilla eri värejä huoneisiinsa ja keittiökalusteiden sijoittamista virtuaalisesti oikeille paikoilleen. (5.)

Lisättyä todellisuutta käytetään nykyisin paljon hyödyksi etämyynnin kasvattamiseksi. Snapchatin kanssa tehdyssä yhteistyössä Kohl's Virtual Closet -sovelluksessa asiakkaat voivat kokeilla päälleen lisättyä todellisuutta hyödyntäen virtuaalisesti monia eri tuotteita, jotka voidaan sitten ostaa suoraan sovelluksella. Lisätyn todellisuuden teknologialla on suuret mahdollisuudet helpottaa kotoaan ostoksia tekevien ihmisten elämää, kun he voivat kokeilla, miltä esimerkiksi tietynlaiset vaatteet näyttävät heidän päällään. (6.)

2.2.2 Viihde

Monet uskovat, että nykypäivän viihteestä tulee tulevaisuudessa aiempaa realistisempia, mukaansatempaavampia sekä enemmän vuorovaikutusta mahdollistavia. Lisätyn todellisuuden teknologia on yksi suurimmista vaikuttajista tähän muutokseen. (7.)

Viihdekäytössä lisätystä todellisuudesta puhuttaessa monille tulee ensimmäisenä varmasti mieleen pelit, joissa erilaisia virtuaalisia objekteja luodaan reaalimaailmaan ja joiden kanssa käyttäjä voi vuorovaikuttaa. Pokémon GOn julkaisun jälkeen monet pelikehittäjät rohkaistuivat tekemään omia lisätyn todellisuuden pelejään. Esimerkkinä lisättyä todellisuutta hyödyntävistä peleistä voidaan

mainita erilaisia tehtäviä tarjoava Temple Treasure Hunt, ammuskelupeli Real Strike ja kauhupeli Zombie Go. (7.)

Musiikissa lisätty todellisuus ei ole täysin uusi keksintö, vaan eri artistit ovat käyttäneet sitä hyödyksi jo joitakin vuosia. Esimerkiksi eräällä jo edesmenneen poptähti Michael Jacksonin vuonna 2017 julkaistulla levyllä oli mukana juliste, johon kameraa osoittamalla se heräsi eloon ja siitä lensi mustia variksia oikeaan maailmaan samalla, kun taustalla soi bonusmateriaalina Michael Jacksonin kappaleista tehty yhdistelmäkappale. (8.)

Elokuva- ja televisioteollisuuden tulevaisuus saattaa tulla lisätyn todellisuuden myötä viihdyttävämmäksi ja mukaansatempaavammaksi kuin koskaan. Lisättyä todellisuutta voidaan hyödyntää esimerkiksi siten, että katsojan jo näkemän kuvan päälle lisätään erilaista virtuaalista sisältöä. (7.) Kuvassa 2 reaaliaikaiseen videokuvaan on lisätty kuljettajan viereen kierrosluku- ja nopeusmittarit havainnoillistamaan katsojalle auton vauhtia.

KUVA 2. Kuvakaappaus videolta, jossa kuvaan on lisätty auton mittaristo (9)

Viihdekäytöstä puhuttaessa ei sovi unohtaa yhdysvaltalaista viestintäsovellusta Snapchatia, jolla on päivittäin yli 200 miljoonaa käyttäjää. Lisätyn todellisuuden kokemuksia Snapchatiin luodaan Lens Studiolla. Lens Studio on työpöytäkäyttöön tarkoitettu sovellus, jolla kehittäjät voivat luoda lisätyn todellisuuden kokemuksia käytettäväksi Snapchatissa. (10.)

Snapchatin lisätyn todellisuuden kokemuksista voidaan esimerkkinä mainita käyttäjän kasvojen muotoja tunnistavat ja liikkeitä seuraavat erilaiset naamiot, jotka luodaan virtuaalisesti käyttäjän kasvojen päälle. Myös erilaiset käyttäjän liikkeitä seuraavat ja niiden pohjalta esimerkiksi jonkin ihmismäisen hahmon luovat suotimet ovat Snapchatissa yleisiä. (11.)

2.2.3 Koulutus

Lisätyn todellisuuden teknologian kehittyminen ja saavutettavuus tekee helpoksi sen hyödyntämisen myös erilaisissa opetustilanteissa. Opettajat voivat esimerkiksi näyttää virtuaalisia esimerkkejä ja lisätä pelillisiä elementtejä tukeakseen kirjoista löytyvää tietoa. Tämä auttaa oppilaita oppimaan nopeammin ja helpottaa tiedon muistamista. (12.)

Tällaisista koulujen opetuskäyttöön tarkoitetuista sovelluksista mainittakoon esimerkkinä Dinosaur 4D+ ja Google Expeditions. Dinosaur 4D+ antaa käyttäjän skannata erilaisia kortteja mobiililaitteen kameralla. Skannauksen jälkeen oppilaat voivat nähdä tietoja dinosauruksista, seurata dinosauruksen liikkeitä ja käyttää sovellusta niiden kääntämiseen ja zoomaamiseen. Google Expeditions sen sijaan tekee erilaisten kolmiulotteisten objektien näkemisen luokkahuoneessa mahdolliseksi. Esimerkkinä tällaisia objekteja ovat tulivuoret, myrskyt ja jopa DNA. Google Expeditions tarjoaa yli 100 erilaista lisätyn todellisuuden tutkimusretkeä, jotka tekevät oppimisesta vuorovaikutteisempaa ja jännittävämpää kuin ennen. (12.)

Eri ammattialojen opetukseen tarkoitettujen välineiden hinta on erittäin korkea. Lisätty todellisuus tekee koulutuksesta vuorovaikutteisempaa ja antaa tavan kulujen karsimiseen. Esimerkiksi sotilaiden koulutus lisätyn todellisuuden avulla luo turvallisen ja halvemman tavan erilaisten välineiden opetteluun virtuaalisessa ympäristössä. (12.)

2.2.4 Rakentaminen ja suunnittelu

Lisättyä todellisuutta hyödynnetään nykyisin paljon myös erilaisissa suunnittelu- ja rakennustöissä. Päähän asetettavien lisätyn todellisuuden lasien avulla, joilla heijastetaan haluttua tietokoneella luotua kuvaa oikeiden fyysisten objektien päälle, voidaan esimerkiksi helpottaa ja nopeuttaa työntekijän tekemää työtä. Esimerkiksi jonkin tietyn koneen osan päälle voidaan heijastaa seikkaperäiset ohjeet, jotka visuaalisesti ohjaavat työntekijän eri korjausvaiheiden läpi. (13.)

Nopeammasta työnteosta lisätyn todellisuuden lasien avulla voidaan mainita työntekijä, joka teki johdotuksen tuuliturbiinin hallintalaitteeseen ensiksi yhtiön omalla prosessilla ja sen jälkeen teki saman työn käytämällä lisätyn todellisuuden laseja, jotka heijastivat työvaiheet käyttäjän näkökenttään reaaliajassa. Jo ensimmäisellä käyttökerralla lisätyn todellisuuden lasien käyttö nopeutti työprosessia 34 %. (13.)

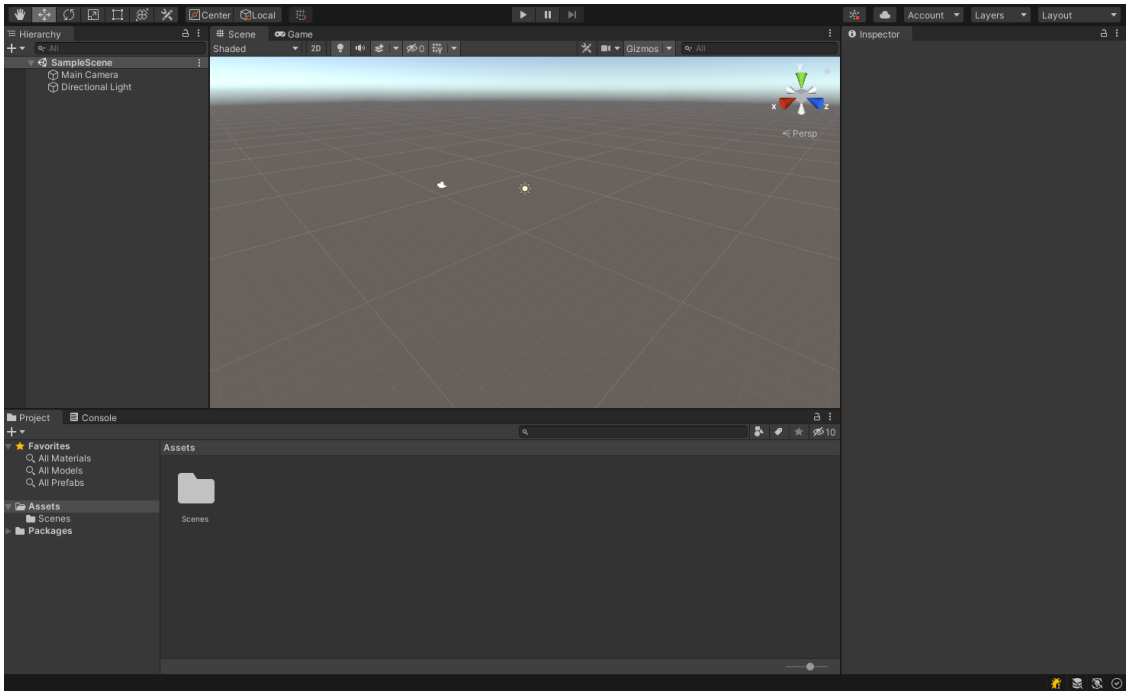
Rakennusalalla erilaisten virtuaalisten suunnitelmien yhdistäminen fyysiseen rakennustyömaahan onnistuu lisätyn todellisuuden avulla. Se lisää tarkkuutta ja tehokkuutta vähentämällä virheitä ajan, rahan ja resurssien käytössä. Esimerkkinä tallaisesta rakennusalalla hyödynnetystä lisätyn todellisuuden sovelluksesta on AugView-niminen sovellus, joka on tarkoitettu helpottamaan työskentelyä maan alle rakennettaessa. Maan alla työskenneltäessä kaivausprosessissa on aina riski osua johonkin rakenteeseen. Googlen karttojen avulla AugView mahdollistaakin maanalaisten asioiden näkemisen, kuten ojat ja haudatut kaapelit. (14.)

3 LAITTEISTOT JA TEKNOLOGIAT

Tässä kappaleessa esitellään tarkemmin opinnäytetyön esimerkkisovelluksen kehityksessä käytetyt laitteistot ja teknologiat. Unitya lisäosineen käytettiin sovelluksen rungon luomiseksi ja Visual Studiolla tehtiin projektissa vaadittavat skriptit.

3.1 Unity

Unity on Unity Technologiesin luoma pelimoottori, joka sisältää myös alustariippumattoman ohjelmointiympäristön (15). Unitylla voidaan tehdä sovelluksia yli 20:lle eri alustalle, joista työpöytäkäyttöön tarkoitettuja käyttöjärjestelmiä ovat Windows, macOS sekä Linux. Mobiililaitteille sovellusten kehitys onnistuu sekä iOS- että Android-pohjaisille käyttöjärjestelmille. Sovelluskehitystä voidaan tehdä myös lisätyn todellisuuden alustoille, kuten erilaisille virtuaalilaseille. (16.) Kuvassa 3 on Unityn käyttöliittymä.



KUVA 3. Unityn käyttöliittymä

Unitylla on tehty monenlaisia pelejä eri alustoille, joista yksi suosituimmista lisättyä todellisuutta hyödyntävistä peleistä viimeisiltä vuosilta on Pokémon GO. Sen kehitti yhdysvaltalainen ohjelmisto-

ja pelialan yritys Niantic ja julkaisi The Pokémon Company vuonna 2016. Pokémon GOssa pelaajan tehtävänä on pyydystää erilaisia Pokémoneja ja taistella muita pelaajia vastaan mobiililaitteellaan. Pyydystettäessä Pokémoneja ne ilmestyvät käyttäjän mobiililaitteen ruudulle lisättynä todelliseen ympäröivään maailmaan. (17.)

3.2 AR Foundation

AR Foundation on Unityn lisäosa, joka tekee mahdolliseksi työskentelyn lisätyn todellisuuden alustoilla. AR Foundation ei itsessään sisällä mitään lisätyn todellisuuden toimintoja, vaan se toimii käyttöliittymänä Unity-kehittäjille. AR Foundationion käyttö kohdelaitteessa vaatii lisäksi Unityn virallisesti tukemat erilliset asennettavat paketit kohdealustalle. (18.)

AR Foundation on kokoelma erilaisia ohjelmointirajapintoja laitteille. Nämä ohjelmointirajapinnat tukevat muun muassa laitteen seurantaa eli laitteen sijainnin ja suunnan seuraamista fyysisessä tilassa sekä vaaka- ja pystysuorassa olevien tasojen tunnistamista. (18.)

3.3 ARCore

ARCore on Googlen luoma alusta lisätyn todellisuuden kokemuksien rakentamiseen. Erilaisia ohjelmointirajapintoja käyttämällä ARCore tekee laitteelle mahdolliseksi ympäristön tunnistamisen ja ymmärtämisen sekä vuorovaikuttamisen ympäristön kanssa. (19.)

ARCore käyttää kolmea avainasemassa olevaa ominaisuutta integroidakseen virtuaalista sisältöä reaalimaailmaan puhelimen kameraa hyödyntämällä. Yksi niistä on liikkeen tunnistaminen, mikä mahdollistaa sen, että laite ymmärtää ja voi seurata sijaintiaan suhteessa ympäröivään maailmaan. Ympäristön ymmärtäminen sallii erilaisten pintojen koon ja sijainnin, kuten lattian, kahvipöydän tai seinien havaitsemisen. Valon arviointi antaa mahdollisuuden laitteen nimensä mukaisesti arvioida ympäristössä voimassa olevaa valaistusta. (19.)

Unitylle ARCore löytyy omana lisäosanaan. Se tekee Unityssa mahdolliseksi lisättyä todellisuutta hyödyntävien sovellusten kehityksen Android-pohjaisille laitteille ja sisältää muun muassa tässä opinnäytetyössä käytettyjä alijärjestelmiä, kuten tasojen tunnistus ja säteen luominen. (20.)

3.4 XR Interaction Toolkit

XR Interaction Toolkit on Unityn lisäosa, joka tarjoaa puitteet kolmiulotteiseen vuorovaikutukseen sekä käyttöliittymän vuorovaikutukseen Unityn syöttötapahtumista. XR Interaction Toolkitin käyttö Unityssa vaatii AR Foundationin asennuksen Unity-projektiin. XR Interaction Toolkit sisältää muun muassa seuraavia lisätyn todellisuuden toiminnallisuuksia, jotka on selitetty taulukossa 1. (21.)

TAULUKKO 1. XR Interaction Toolkitin termit selitettynä (21)

Termi	Selitys
Controller	Komponentti, joka muuntaa XR-ohjaimen syötteen, kuten napin painalluksen vuorovaikutustapahtumaksi, kuten leijunta tai valinta.
Object	Mitä tahansa, mitä käyttäjä näkee tai minkä kanssa voi vuorovaikuttaa virtuaalisessa maailmassa.
Interactor	Objekti kohtauksessa, joka voi valita tai liikuttaa toista objektia samassa kohtauksessa.
Interactable	Kohtauksessa oleva objekti, jonka kanssa käyttäjä voi vuorovaikuttaa (esimerkiksi ottaa kiinni, painaa tai heittää).
Hover	Tila, jossa kohtauksessa oleva valintaobjekti (Interactor) on oikeassa tilassa vuorovaikuttaakseen objektin kanssa.
Select	Tila, jossa valintaobjekti (Interactor) on parhaillaan vuorovaikutuksessa objektin kanssa.
Interaction Manager	Hallintakomponentti, joka vastaa erilaisten objektien välisestä vuorovaikutuksesta (Interactor ja Interactable).
Gesture	Eleiden kokoelma, jotka muuntuvat tapahtumiksi. Eleillä voidaan manipuloida kohtauksessa olevia objekteja.
Annotation	Palanen sisältöä, joka on asetettu lisätyn todellisuuden objektin viereen tai päälle ja antaa käyttäjille tietoja.
Haptic	Sensori tai visuaalinen ärsyke, joka lähetetään käyttäjälle palautteen antamiseksi vuorovaikutukselle.

XR Interaction Toolkit sisältää kokoelman komponenetteja, jotka tukevat muun muassa seuraavanlaisia vuorovaikutustapahtumia, järjestelmäriippumaton syöte, objektin leijunta (Object Hover), valinta ja tarttuminen sekä visuaalinen palaute havainnoimaan mahdollisia ja aktiivisia vuorovaikutuksia (21).

3.5 Visual Studio

Visual Studio on Microsoftin luoma integroitu ohjelmointiympäristö, jota käytetään ohjelmistotuotannossa esimerkiksi tietokoneohjelmien, internet-sivujen, internet-ohjelmien, internet-palveluiden sekä mobiilisovellusten kehitykseen. Se sisältää työkaluja, jotka nopeuttavat ja helpottavat ohjelmistokehitysprosessia. (22.)

Tässä opinnäytetyössä Visual Studiota käytettiin Unityn projektissa vaadittavien skriptien luomiseen ja muokkaamiseen. Skriptit mahdollistavat sovellukselle käyttäjän syötteeseen reagoinnin ja sovelluksen toiminnan halutulla tavalla. Skripteillä sovellukseen voidaan luoda esimerkiksi graafisia efektejä sekä hallita objektien fyysistä käyttäytymistä. (23.) Tässä opinnäytetyössä käytetyistä skripteistä puhutaan tarkemmin luvussa 5.

3.6 Android Studio

Android Studio on yhdysvaltalaisen Google LLC:n kehittämä työkalu, joka tarjoaa helpon ja nopean sovelluskehityksen Android-pohjaisille laitteille. Android Studiolla kehitettyjen sovelluksien asentaminen ja ajaminen tietokoneella virtuaalisissa emulaattoreissa onnistuu ilman fyysistä Android-laitetta. (24.)

Android Studio ei ole pakollinen Unitya käytettäessä, mutta se helpottaa sovelluskoodien testaamista. Sovelluksen testaus onnistuu emuloidulla virtuaalilaitteella ja web-kameran kanssa aivan kuten puhelimellakin. Sovellusta testatessa Android Studiolla voidaan muun muassa asettaa sovelluksen koodeihin erilaisia pysäytyspisteitä, jotka keskeyttävät sovelluksen toiminnan halutulla hetkellä, jotta käyttäjän on helpompaa ratkoa mahdollisia ongelmakohtia koodissaan. (25.)

Tässä opinnäytetyössä Android Studiota ei käytetty sovelluksen testaamiseen. Unityn skriptit olivat sen verran yksinkertaisia ja lyhyitä, että sovelluksen testaaminen oli helppoa myös fyysisellä laitteella.

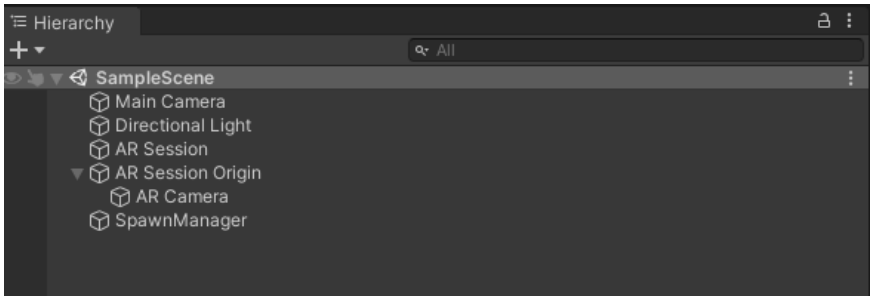
4 UNITYN KÄYTTÖ LISÄTYN TODELLISUUDEN MOBIILISOVELLUKSESSA

Lisätyn todellisuuden mobiilisovelluksen kehittämiseksi Unitylla tarvittiin Unityn lisäosien lisäksi sopiva mobiililaite ja jokin tekstieditori, jolla Unityn skriptejä voitiin kirjoittaa ja muokata. Tässä opinnäytetyössä tekstieditorina oli Visual Studio ja kohdealustana käytettiin puhelinta, jossa oli asennettuna Android 10 -käyttöjärjestelmä. Käytettävässä laitteessa tulee olla vähintään Android 7.0 -käyttöjärjestelmä, jotta lisätyn todellisuuden ominaisuudet saadaan toimimaan.

4.1 Lisätyn todellisuuden projektin aloitus Unitylla

Ensimmäisenä luotiin 3D-projekti Unitylla. Tässä opinnäytetyössä käytettiin Unityn versiota 2020.3.11f1. Projektin luomisen jälkeen Unityn Package Managerista asennettiin ensiksi AR Foundation, jonka jälkeen Android-alustaa varten tarvittiin lisäksi ARCore XR Plugin. Sekä AR Foundationiosta että ARCore XR Pluginista oli tätä opinnäytetyötä tehdessä asennettu uusimmat käytettävissä olevat versiot eli versiot 4.1.7. (26.)

Lisäosien asennusten jälkeen Unityn projektista poistettiin Scenestä eli kohtauksesta Main Camera -peliobjekti, koska projektissa käytettiin AR Cameraa omilla skripteillään. Kohtaukseen lisättiin tämän jälkeen AR Session, joka hallinnoi lisätyn todellisuuden kokemuksen elinkaarta sallimalla tai poistamalla käytöstä lisätyn todellisuuden halutulla alustalla. Lisäksi kohtaukseen tuli lisätä AR Session Origin, jonka tehtävänä on muuntaa seurattavat ominaisuudet Unityn kohtauksessa lopulliseen sijaintiin ja suuntaan sekä skaalaukseen ja kokoon. Nämä muunnetut ominaisuudet mahdollistivat virtuaalisten objektien vuorovaikutuksen ja käsittelyn. (26.) Kuvassa 4 on esitetty yksinkertaisen lisätyn todellisuuden Unity-projektin hierarkia.

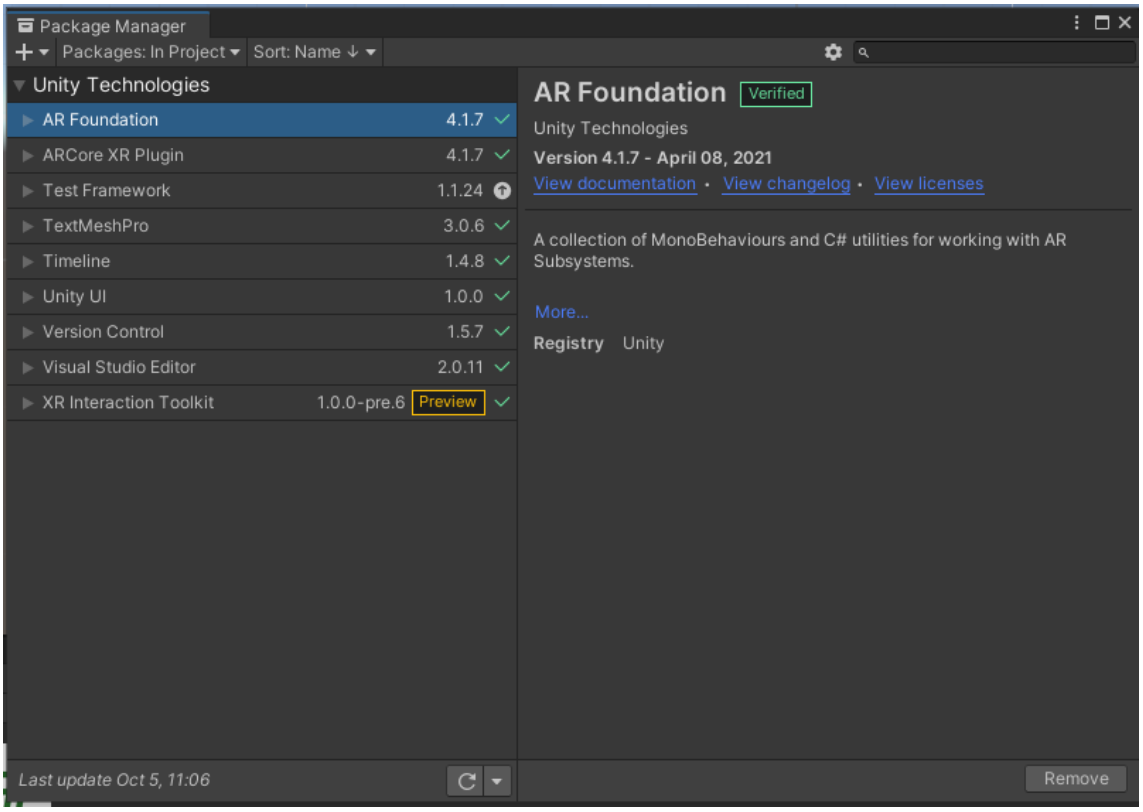


KUVA 4. Unityn projektin hierarkia

Ennen kuin Unityn projektin sai toimimaan kohdelaitteella, tuli projektin asetuksista asettaa minimiversioksi vähintään Android 7.0. Asetuksista tuli myös asettaa ARCore valituksi plug-iniksi eli liitännäiseksi, jotta Unity osasi asentaa ja käyttää projektissa tarvittavia paketteja. (26.)

4.2 Unityn tarvittavat lisäosat

Seuraavissa kappaleissa käydään läpi, mitä lisäosia Unityyn täytyi asentaa, jotta lisätyn todellisuuden kaikki tarvittavat ominaisuudet saatiin toimimaan. Unityssa lisäosien asentaminen hoidettiin Package Managerin avulla, josta haettiin ja asennettiin halutut lisäosat. Lisätyn todellisuuden sovelluksen kehitykseen Unityssa tarvittavat lisäosat ja niiden kirjoitushetkellä saatavilla olevat versiot on esitetty kuvassa 5.



KUVA 5. Unity Package Managerin kautta asennetut lisäosat

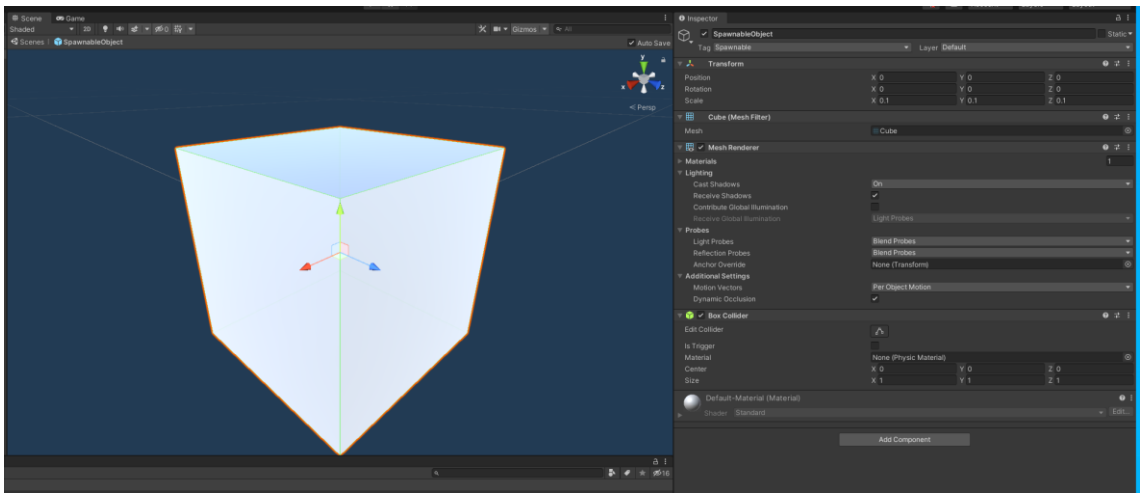
Ensimmäisenä Package Managerin kautta tuli projektiin lisätä AR Foundation, joka sisällytti lisätyn todellisuuden projektiin. Pelkkä AR Foundation ei itsessään vielä riittänyt lisätyn todellisuuden sovellusten luontiin, vaan sen lisäksi tarvittiin vielä alustakohtainen paketti, jonka avulla sovelluksen toimivuus taattiin halutulla alustalla. Tässä opinnäytetyössä käytettiin Android-pohjaista laitetta, joten Unityyn tarvittiin ARCore XR Plugin. (26.)

ARCore XR Plugin mahdollisti Unityssa sovelluskehityksen Android-pohjaisille laitteille, jotka käyttävät Googlen kehittämää ARCorea. ARCore XR Plugin ei itsessään sisällä mitään käyttöliittymää ohjelmoinnille, joten yleensä on helpoin käyttää AR Foundationin valmiita skriptejä perustana mobiililaitteiden lisätyn todellisuuden sovelluksille. (20.)

XR Interaction Toolkit oli opinnäytetyön kirjoitushetkellä keskeneräinen tuote, mutta se on täysin toimiva lisäosa ja sen asentaminen Unityyn oli mahdollista. XR Interaction Toolkit tekee lisätyn todellisuuden toiminnallisuuden Unityssa mahdolliseksi helposti ja nopeasti ilman, että käyttäjän tarvitsee koodata kaikkia toiminnallisuuksia skripteihin itse. Kirjoitushetkellä XR Interaction Toolkitissä tuettuja lisätyn todellisuuden mobiiliprojektissa hyödynnettyjä ominaisuuksia olivat objektien kanssa vuorovaikuttaminen eli objektin valinta, objektin siirtäminen sekä objektin kääntely ja skaalaaminen. (27.)

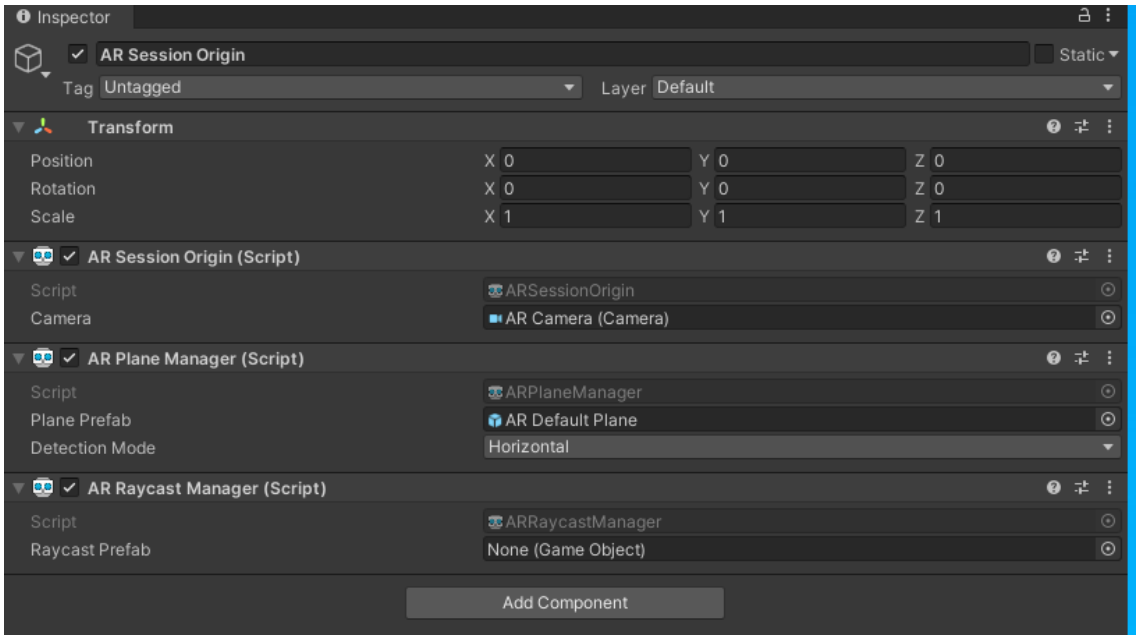
4.3 Objektin luonti ja manipulointi

Objektien luonti ja manipulointi Unityn lisätyn todellisuuden sovellukseen tapahtuu seuraavalla tavalla. Ensiksi tulee tehdä luotavasta objektista Unityssa valmis elementti, jota voidaan myöhemmin käyttää helposti hyödyksi sovelluksessa. Yksinkertaisimmillaan luotava objekti voi olla vaikka kuutio, joka nimetään halutulla tavalla. Kuvassa 6 on näkymä Unityssa luodusta valmiista elementistä, jolle on asetettu nimeksi SpawnableObject.



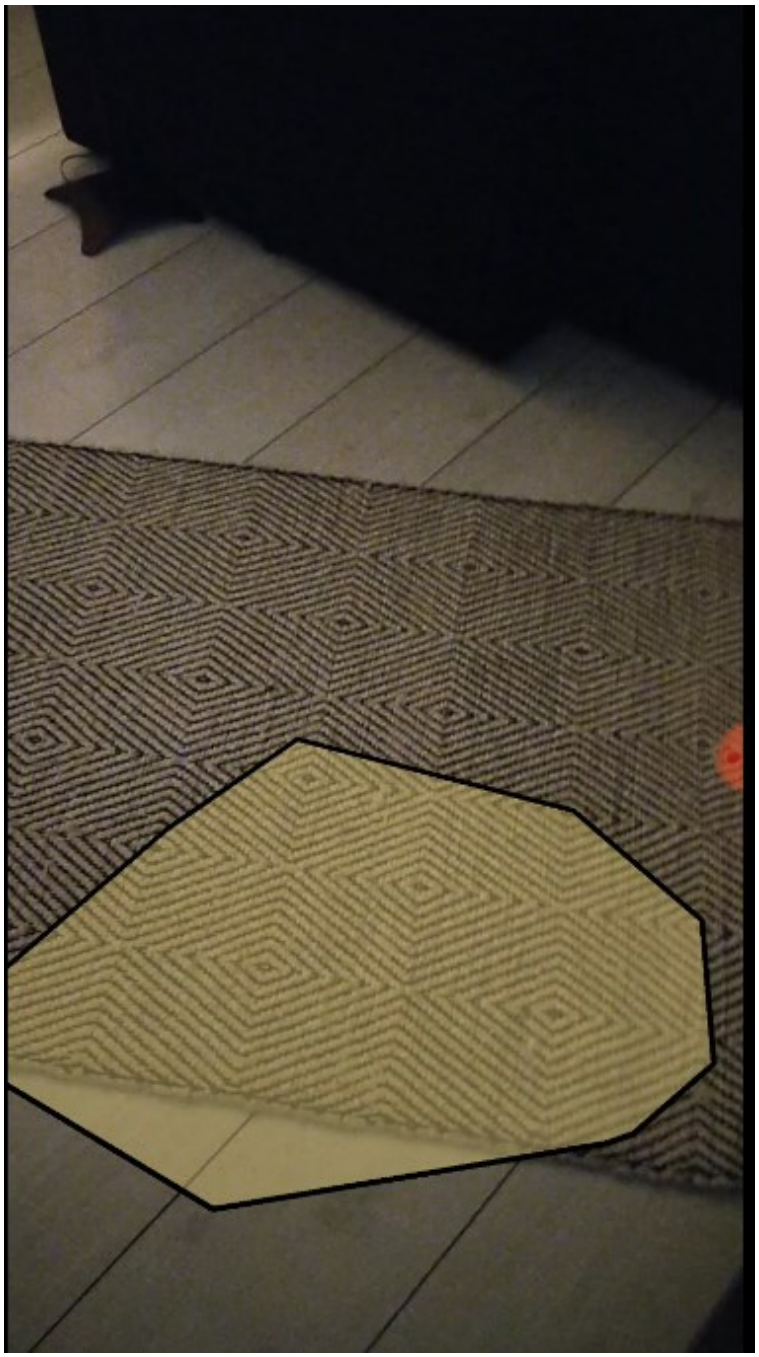
KUVA 6. Unityssa luotu uudelleen käytettävä elementti

AR Foundationin sisältämä luokka ARSessionOrigin tarvitaan jokaiseen lisätyn todellisuuden sovellukseen. Se sisältää kameran ja kaikki peliobjektit, jotka luodaan havaituista toiminnoista kuten tasoista. (28.) Kuvassa 7 on Unityn projektissa käytettävän AR Session Originin sisältämät komponentit sekä AR Plane Managerin sisältämä AR Default Plane-elementti, joka luodaan, kun tunnistetaan vaakatasossa oleva taso.



KUVA 7. Unityn AR Session Originin sisältämät komponentit

Unityssa säteiden luonti haluttua toiminnallisuutta vasten hoidetaan lisätyn todellisuuden sovelluksessa ARRaycastManager-luokan avulla. ARRaycastManager käyttää AR Foundationin sisään rakennettua säteen luonnin toiminnallisuutta. Sitä hyödynnetään Unityssa lisätyn todellisuuden sovelluksessa siten, että sillä luodaan mobiililaitteen kamerasta lähteviä säteitä seurattavia kohteita vasten. ARRaycastManagerin lisäksi sovelluksessa hyödynnetään AR Plane Manageria eli lisätyn todellisuuden tasojen tunnistamiseen tarkoitettua komponenttia. AR Plane Manager-skripti luo peliobjektit jokaiselle säteellä tunnistettavalle tasolle. (29.) Kuvassa 8 on kuvankaappaus mobiilisovelluksesta, jossa vaakatasossa oleva tunnistettu taso näkyy keltaisella värillä.



KUVA 8. Mobiilisovelluksessa tunnistettu taso havainnollistettuna keltaisella värillä

Valitun elementin eli tässä esimerkkitapauksessa kuution luonti tunnistetulle tasolle tapahtuu skriptin avulla. Skriptissä lähetetään sovelluksen ajon aikana mobiililaitteen kamerasta säde kohtaan, johon käyttäjä on sormellaan painanut. Mikäli säde osuu tunnistetulle tasolle kohtaan, jossa ei ole jo luotuna objektia, luodaan siihen kohtaan kuutio. Kuvassa 9 on skripti, jossa luodaan objekti tasolle käyttäjän haluamaan kohtaan.

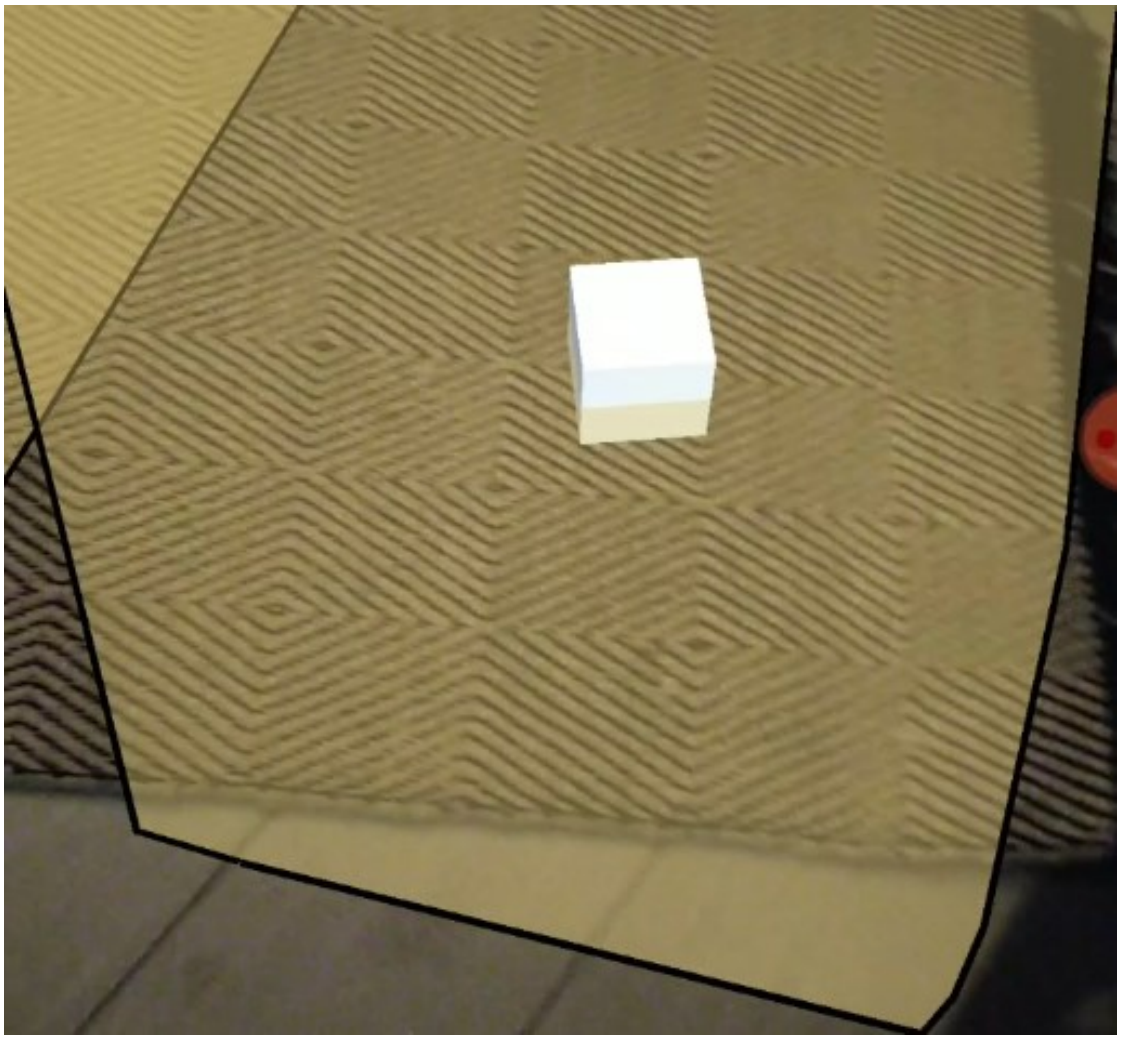
```
void Update()
{
    if (Input.touchCount == 0)
        return;

    RaycastHit hit;
    // Lähetetään mobiililaitteen kamerasta säde kohtaan johon käyttäjä on painanut
    Ray ray = arCam.ScreenPointToRay(Input.GetTouch(0).position);

    if(m_RaycastManager.Raycast(Input.GetTouch(0).position, m_Hits))
    {
        // Mikäli käyttäjä painanut ruudulle kohtaan jossa ei ole jo luotua objektia
        if(Input.GetTouch(0).phase == TouchPhase.Began && spawnedObject == null)
        {
            if(Physics.Raycast(ray, out hit))
            {
                // Mikäli säde osuu kohtaan, jossa on jo objekti, ei luoda kyseiseen kohtaan uutta objektia
                if (hit.collider.gameObject.tag == "Spawnable")
                {
                    spawnedObject = hit.collider.gameObject;
                }
                // Kutsutaan objektin luontifunktiota
                else
                {
                    SpawnPrefab(m_Hits[0].pose.position);
                }
            }
        }
        // Mikäli luotava objekti ei ole tyhjä, niin siirretään jo luotua objektia
        else if (Input.GetTouch(0).phase == TouchPhase.Moved && spawnedObject != null)
        {
            spawnedObject.transform.position = m_Hits[0].pose.position;
        }
        // Poistetaan valinta jo luodulta objektilta
        if(Input.GetTouch(0).phase == TouchPhase.Ended)
        {
            spawnedObject = null;
        }
    }
}
// Luodaan objekti haluttuun kohta
1 reference
private void SpawnPrefab(Vector3 spawnPosition)
{
    spawnedObject = Instantiate(spawnablePrefab, spawnPosition, Quaternion.identity);
}
```

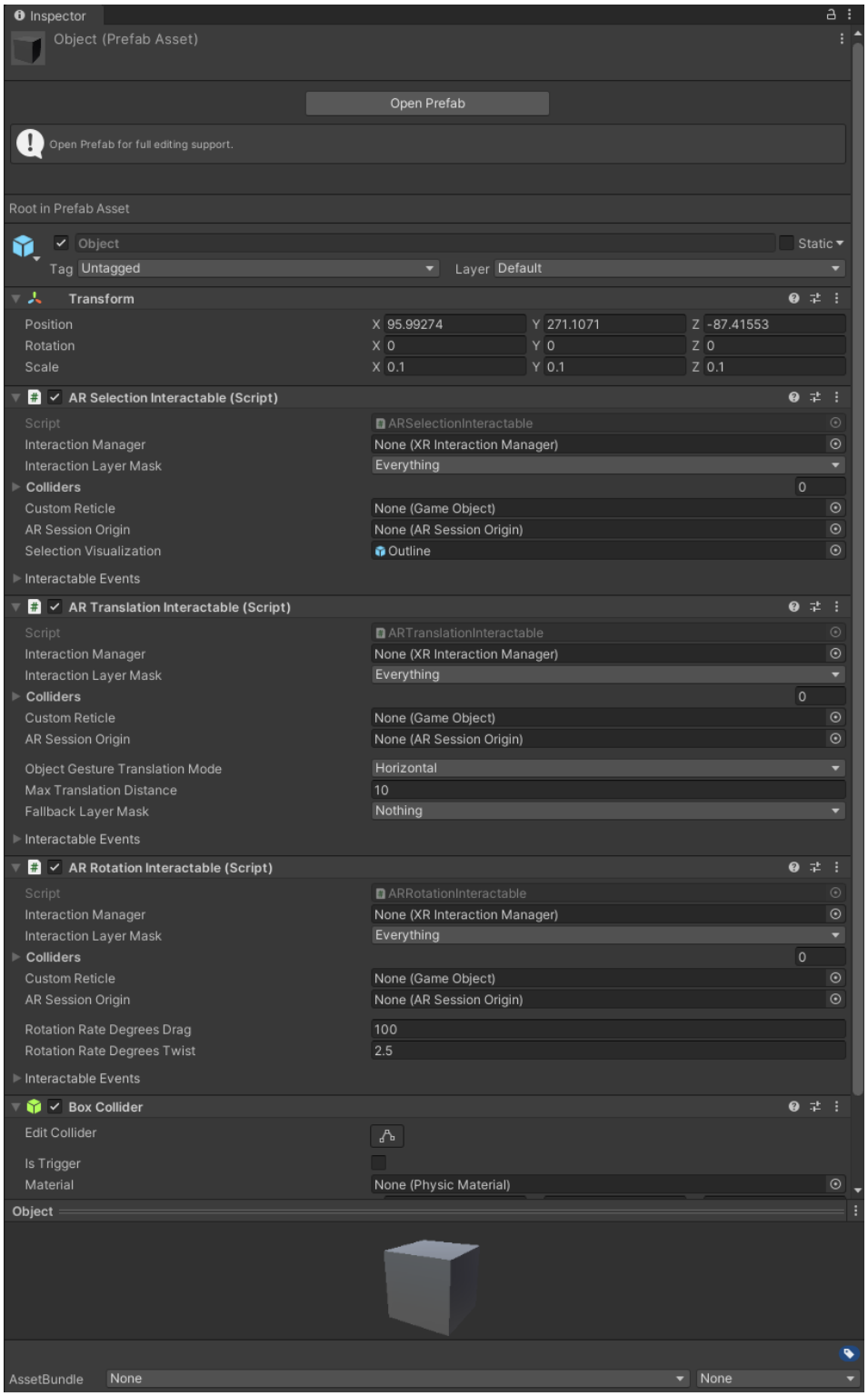
KUVA 9. Koodi, jossa objekti luodaan havaitulle tasolle haluttuun kohtaan

Mikäli käyttäjän painallus osuu kohtaan, jossa on jo valmiiksi luotuna kuutio, asettuu kuutio valituksi käyttäjälle. Valinnan jälkeen kuutiota voidaan siirrellä tasolla sormea mobiililaitteen ruudulla siirtämällä. Kuvassa 10 on havainnollistettuna tunnistetulla tasolla oleva valkoinen kuutio, jota käyttäjä voi sormeaan liikuttamalla siirrellä.



KUVA 10. Kuva sovelluksesta, jossa tunnistetulle tasolle on luotu kuutio

Luodun objektin liikuttamiseksi, skaalaamiseksi ja kääntämiseksi voidaan käyttää hyödyksi XR Interaction Toolkitiä. Unityssa jokaisessa kohtauksessa, jossa XR Interaction Toolkitiä halutaan käyttää, tulee olla hierarkiassa mukana myös XR Interaction Manager. XR Interaction Managerin tehtävänä on ylläpitää eri komponenttien välisiä vuorovaikutuksia. Luodun objektin kanssa vuorovaikuttamiseen löytyvät valmiit skriptit, jotka tulee lisätä objekteihin, jotka sovelluksessa ruudulle halutaan luoda. (21.) Kuvassa 11 on Unityssa luotu objekti, johon on lisätty tarvittavat skriptit valintaa, siirtämistä ja pyörittämistä varten.



KUVA 11. Unityssa luotu elementti, johon on lisätty halutut XR Interaction Toolkitin skriptit

XR Interactionin Toolkitin valmiit skriptit sisältävät edellä mainittujen valinnan, siirtämisen ja pyörittämisen lisäksi myös muita toiminnallisuuksia, kuten skaalauksen. Skriptien toimintaa pystyy myös halutessaan itse laajentamaan. Käyttäjä voi esimerkiksi itse määritellä, millä eleillä objektia voidaan manipuloida. (21.)

5 ESIMERKKISOVELLUS

Tässä opinnäytetyössä luotiin yksinkertainen lisätyn todellisuuden mobiilisovellus Android-pohjaisille laitteille Unity-pelimoottorilla. Ideana oli toteuttaa sovellus, jossa käyttäjä voi asettaa erilaisia objekteja ympäristöönsä puhelimen kameraa hyödyntäen sekä siirrellä, käännellä ja skaalata objekteja lisäyksen jälkeen.

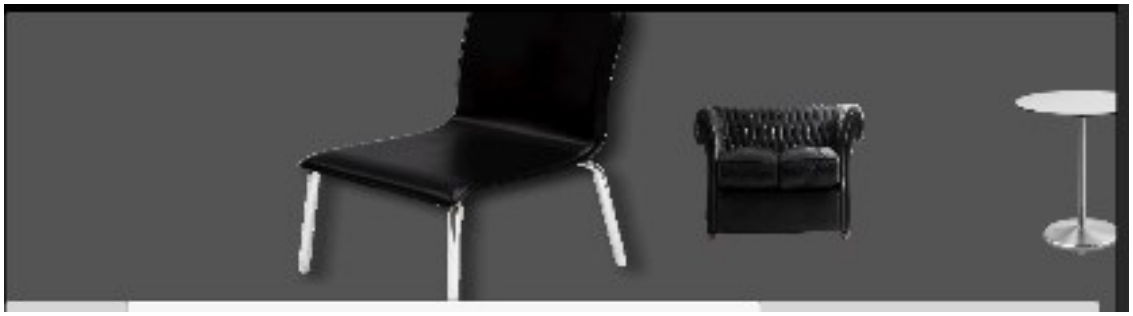
Esimerkkisovelluksessa voidaan puhelimen kameraa hyödyntämällä tunnistaa vaakatasossa olevia tasoja, joille voidaan asettaa erilaisia huonekaluja. Käyttäjä voi painaa ruudulla olevan tähtäimen kohdalle, jolloin siihen luodaan alapalkista valittu huonekalu. Huonekalun voi valita aktiiviseksi painamalla sitä sormella. Valinnan jälkeen käyttäjä voi puhelimen ruudulla kääntää huonekalua kahdella sormella pyöräyttäen, siirtää huonekalua sormea liikuttamalla ja kahdella sormella zoomaamalla skaalata huonekalun halutun kokoiseksi. Sovelluksen näkymä on esitetty kuvassa 12.

KUVA 12. Sovelluskuva, jossa on tasolle luotu ja aktiiviseksi valittu tuoli

Esimerkkisovelluksessa haluttujen toiminnallisuuksien saavuttamiseksi Unityssa täytyi käyttää skriptejä. Skriptit kirjoitettiin C#-ohjelmointikielellä, joka on virallinen Unityn tukema ohjelmointikieli. Sovelluksessa käytettävien skriptien toiminta selitetään seuraavissa luvuissa ja ne löytyvät liitteinä opinnäytetyön lopusta.

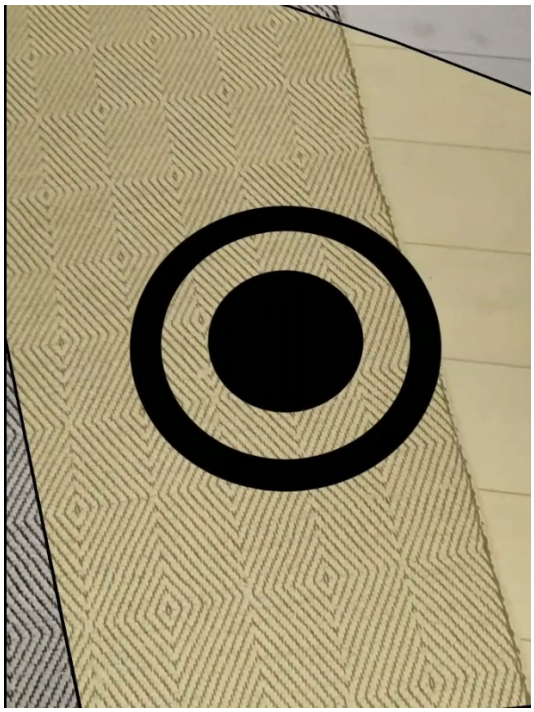
5.1 Objektin luonti ja asettaminen näkymään

Objektit, jotka on mahdollista asettaa näkymään, luodaan sovelluksen alaosan valintapalkkiin Resources-kansiossa olevista huonekalujen asseteista (kuva 13). Item-skriptin avulla kaikille alapalkissa oleville huonekalujen painikkeille asetetaan haluttu kuva sekä yhdistetään siihen Unityssa luotu elementti, jota luonnissa käytetään. (Liite 4)



KUVA 13. Sovelluksen valintapalkki, jossa on valittuna luotavaksi objektiksi tuoli

Sovelluksen alaosan valintapalkin sijainti ja koko määritetään skriptissä, joka löytyy liitteestä 5. Liitteessä 6 olevassa skriptissä määritetään alapalkissa valitun objektin koko hieman isommaksi kuin muut valintapalkin objektit, jotta käyttäjälle hahmottuu helpommin valittuna oleva objekti. Alapalkista valittu objekti voidaan luoda tunnistetulle vaakatasossa olevalle tasolle kohtaan, jossa tähtäin on näkyvillä. Kuvassa 14 näkyy keltaisella värillä taso sekä tähtäin, johon objekti voidaan luoda.



KUVA 14. Sovellusnäkymä, jossa näkyy tunnistettu taso sekä tähtäin objektin luontia varten

Sovelluksen alavalikosta valittu objekti on se, mikä ruutuun painettaessa luodaan. Luotavan objektin valinta tapahtuu ButtonManager-skriptissä (liite 1). Luodun objektin asettaminen valituksi objektiksi tapahtuu Unityn DataHandler-skriptissä olevan SetFurniture-funktion avulla (liite 2). InputManager-skriptin avulla tarkistetaan kamerasta lähtevän säteen osumakohta aiemmin tunnistettuun vaakatasossa olevaan tasoon. Tähän osumakohtaan luodaan valittu objekti. (Liite 3) Kuvassa 15 on näkymään luotu huonekalu.

KUVA 15. Näkymään luotu huonekalu

Luonnin jälkeen objektia voidaan tarkastella eri suunnista liikkumalla ja liikuttamalla kameraa objektin ympärillä. Objektin valinta aktiiviseksi tapahtuu painamalla, jonka jälkeen sitä on mahdollista liikutella, käännellä ja skaalata tason päällä.

5.2 Luodun objektin manipulointi

Objekti voidaan luomisen jälkeen valita aktiiviseksi painamalla sitä ruudulla. Valittu objekti havainnollistetaan käyttäjälle asettamalla sen ympärille Unityssä luotu läpinäkyvä sinertävä laatikko, joka mukailee objektin muotoa ja kokoa. Objektin valinnan jälkeen XR Interaction Toolkitissä olevien valmiiden skriptien avulla sitä voidaan siirrellä, käännellä ja skaalata. Kuvassa 16 on havainnollistettuna valittuna oleva käännetty ja siirretty objekti.

KUVA 16. Sovelluksessa haluttuun suuntaan ja sijaintiin käännetty ja siirretty valittuna oleva tuoli

XR Interaction Toolkitin skriptit toimivat lisäosien asennuksen jälkeen suoraan Unityn projektissa. Unityn projektin hierarkiassa tulee olla hallintakomponentti Interaction Manager, jonka tehtävänä on ylläpitää eri komponenttien välisiä vuorovaikutuksia. Objekteille tulee tämän lisäksi lisätä halutut skriptit komponenteiksi Unityn puolella, jonka jälkeen niitä voidaan käyttää.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli perehtyä lisättyyn todellisuuteen, tutustua sen historiaan ja erilaisiin käyttökohteisiin sekä selvittää, miten Unity-pelimoottorilla ja sen lisäosilla luodaan Android-mobiililaitteella toimiva lisätyn todellisuuden sovellus. Perehdyin lisätyn todellisuuden historiaan ja käyttökohteisiin ja opin kirjoitusprosessin aikana paljon uutta lisätystä todellisuudesta ja sen mahdollisuuksista erilaisissa käyttötilanteissa.

Esimerkkinä luodun lisätyn todellisuuden mobiilisovelluksen tekemisessä opin paremmin käyttämään Unitya ja hyödyntämään sen erilaisia lisäosia ja niiden ominaisuuksia. Opinnäytetyön esimerkkisovellus oli sopivan haastava toteuttaa ja sitä tehdessä sain hyvin demonstroitua Unityn ja sen lisäosien käytön monipuolisuutta lisätyn todellisuuden mobiilisovelluksia suunniteltaessa ja tehdessä.

Sain esimerkkisovelluksessa toimimaan halutut toiminnallisuudet eli objektien luonnin ja manipuloinnin, mutta jatkokehitysmahdollisuuksiakin sovellukseen jäi. Esimerkiksi pystysuorassa olevien tasojen, kuten seinien tunnistaminen ja niille objektien luonti, olisi hyvä lisä tällaiseen sovellukseen. Sen avulla voisi esimerkiksi koristaa seiniä tauluilla ja kuvilla. Pidemmälle kehitettynä sovellusta voisi käyttää apuna esimerkiksi kiinteistövälityksessä, niin että tyhjiin asuntoihin voitaisiin luoda virtuaalisia huonekaluja esittelyä ja tilan havainnollistamista varten.

Unity on tehokas työkalu, jolla voidaan luoda sovelluksia monille eri alustoille. Lisätyn todellisuuden mahdollistavien lisäosien käyttöönotto Unityyn on helppoa ja suoraviivaista ja me mahdollistavat käytännössä kaikenlaisten sovellusten tekemisen. Tulevaisuudessa lisätyn todellisuuden käyttö tulee todennäköisesti lisääntymään entisestään teknologian kehityksen ja lisätyn todellisuuden helpon saavutettavuuden mahdollistaessa entistä monimutkaisempia sovelluskokonaisuuksia.

LÄHTEET

1. Johnson, Dave 2020. What is augmented reality? Here's what you need to know about the 3D technology. Insider. Saatavissa: https://www.businessinsider.com/what-is-augmented-reality?r=US&IR=T. Hakupäivä 3.11.2021.

2. Candy, Chris 2013. The History of Augmented Reality. SevenMedia. Saatavissa: http://sevenmediainc.com/the-history-of-augmented-reality/. Hakupäivä 03.11.2021.

3. Rives, Keith 2018. Valokuva. Augmented Reality: A Comprehensive History (Part 1). Vertebrae. Saatavissa: https://www.vertebrae.com/blog/history-augmented-reality-1/. Hakupäivä 8.11.2021.

4. Engine Creative 2021. Augmented Reality: it's like real life, but better. Saatavissa: https://www.enginecreative.co.uk/blog/unlocking-potential-augmented-reality/. Hakupäivä 04.11.2021.

5. Makarov, Andrew 2021. AR in Retail, Marketing, and Sales in 2021: Practical Yet Innovative. MobiDev. Saatavissa: https://mobidev.biz/blog/augmented-reality-marketing-sales. Hakupäivä 4.11.2021.

6. Kohl's 2021. Reimaginig the Digital Shopping Experience with Snapchat. Saatavissa: https://corporate.kohls.com/news/archive-/2020/august/reimagining-the-digital-shopping-experience-with-snapchat. Hakupäivä 8.11.2021.

7. 3rockAR. Augmented Reality Entertainment: From Games to Sports to Music. Saatavissa: https://www.3rockar.com/augmented-reality-entertainment-games-sports-music/. Hakupäivä 8.11.2021.

8. Carlton, Bobby 2017. Michael Jackson's 'Scream' Features an All-New Track-and Augmented Reality. VRScout. Saatavissa: https://vrscout.com/news/michael-jackson-scream-augmented-reality/#. Hakupäivä 8.11.2021.

9. Arti 2021. Arti AR highlights at SRX -- the first sports augmented reality live from a moving car!. Saatavissa: https://www.youtube.com/watch?v=1jQUkqqnZIc&ab_channel=Arti. Hakupäivä 3.11.2021.

10. Snap Inc 2021. Saatavissa: https://ar.snap.com/. Hakupäivä 9.11.2021.

11. Snap Inc 2021. Saatavissa: https://ar.snap.com/october-lens-drop. Hakupäivä 9.11.2021.

12. Sinha, Shweta 2021. Augmented Reality In Education: A Staggering Insight Into The Future. eLearning Industry. Saatavissa: https://elearningindustry.com/augmented-reality-in-education-staggering-insight-into-future. Hakupäivä 9.11.2021.

13. Abraham, Magid & Annunziata, Marco 2017. Augmented Reality Is Already Improving Worker Performance. Harvard Business Review. Saatavissa: https://hbr.org/2017/03/augmented-reality-is-already-improving-worker-performance. Hakupäivä 9.11.2021.

14. Chinn, Sela 2019. 7 Uses in 2020 for Augmented Reality in Construction. eSUB. Saatavissa: https://esub.com/construction-project-management-software-blog/7-uses-in-2020-for-augmented-reality-in-construction/. Hakupäivä 9.11.2021.

15. Unity Technologies 2021. Saatavissa: https://unity.com/. Hakupäivä 19.10.2021.

16. Unity Technologies 2020. What platforms are supported by Unity. Saatavissa: https://support.unity.com/hc/en-us/articles/206336795-What-platforms-are-supported-by-Unity-. Hakupäivä 19.10.2021.

17. Pokémon 2021. Saatavissa: https://www.pokemon.com/fi/app/pokemon-go/. Hakupäivä 8.11.2021.

18. Unity Technologies 2020. About AR Foundation. Saatavissa: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html. Hakupäivä 19.10.2021.

19. Google 2021. Overview of ARCore and supported development environments. Saatavissa: https://developers.google.com/ar/develop. Hakupäivä 19.10.2021.

20. Unity Technologies 2020. About ARCore XR Plugin. Saatavissa: https://docs.unity3d.com/Packages/com.unity.xr.arcore@4.1/manual/index.html. Hakupäivä 19.10.2021.

21. Unity Technologies 2020. XR Interaction Toolkit. Saatavissa: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@1.0/manual/index.html. Hakupäivä 19.10.2021.

22. Incredibuild 2021. Visual Studio. Saatavissa: https://www.incredibuild.com/integrations/visual-studio. Hakupäivä 19.10.2021.

23. Unity Technologies 2021. Scripting. Saatavissa: https://docs.unity3d.com/Manual/ScriptingSection.html. Hakupäivä 19.10.2021.

24. Google 2021. Android Studio. Saatavissa: https://developer.android.com/studio/. Hakupäivä 19.10.2021.

25. Google 2021. Debug your app. Saatavissa: https://developer.android.com/studio/debug. Hakupäivä 19.10.2021.

26. Unity Technologies 2021. Setting up AR Foundation. Saatavissa: https://learn.unity.com/tutorial/setting-up-ar-foundation#. Hakupäivä 19.10.2021.

27. Dalby, Matt & Fuad, Matt 2019. XR Interaction Toolkit Preview Package is here. Unity. Saatavissa: https://blog.unity.com/technology/xr-interaction-toolkit-preview-package-is-here. Hakupäivä 19.10.2021.

28. Unity Technologies 2020. AR plane manager. Saatavissa: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/plane-manager.html. Hakupäivä 19.10.2021.

29. Unity Technologies 2020. AR Raycast Manager. Saatavissa: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/raycast-manager.html. Hakupäivä 19.10.2021.

LIITTEET

LIITE 1 ButtonManager.cs

LIITE 2 DataHandler.cs

LIITE 3 InputManager.cs

LIITE 4 Item.cs

LIITE 5 UIContentFitter.cs

LIITE 6 UIManager.cs

BUTTONMANAGER.CS LIITE 1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ButtonManager : MonoBehaviour
{
    private Button btn;
    [SerializeField] private RawImage buttonImage;


    private int _itemId;
    private Sprite _buttonTexture;


    public int ItemId
    {
        set { _itemId = value; }
    }

    public Sprite ButtonTexture
    {
        set
        {
            _buttonTexture = value;
            buttonImage.texture = _buttonTexture.texture;
        }
    }
    // Start is called before the first frame update
    void Start()
    {
        btn = GetComponent<Button>();
        btn.onClick.AddListener(SelectObject);
    }

    // Update is called once per frame
    void Update()
    {
        if (UIManager.Instance.OnEntered(gameObject))
        {
            transform.localScale = Vector3.one * 2;
        }
        else
        {
            transform.localScale = Vector3.one;
        }
    }

    void SelectObject()
    {
        DataHandler.Instance.SetFurniture(_itemId);
    }

}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DataHandler : MonoBehaviour
{
    private GameObject furniture;

    [SerializeField] private ButtonManager buttonPrefab;
    [SerializeField] private GameObject buttonContainer;
    [SerializeField] private List<Item> _items;

    private int current_id = 0;

    private static DataHandler instance;
    public static DataHandler Instance
    {
        get
        {
            if (instance == null)
            {
                instance = FindObjectOfType<DataHandler>();
            }
            return instance;
        }
    }

    private void Start()
    {
        LoadItems();
        CreateButtons();
    }

    //Luodaan objektit Resources kansiosta alapalkin valintaa varten
    void LoadItems()
    {
        var items_obj = Resources.LoadAll("Items", typeof(Item));
        foreach (var item in items_obj)
        {
            _items.Add(item as Item);
        }
    }

    //Luodaan alapalkin napit luotaville objekteille
    void CreateButtons()
    {
        foreach (Item i in _items)
        {
            ButtonManager b = Instantiate(buttonPrefab, buttonContainer.trans-
form);
            b.ItemId = current_id;
            b.ButtonTexture = i.itemImage;
            current_id++;
        }
        buttonContainer.GetComponent<UIContentFitter>().Fit();

    }

    public void SetFurniture(int id)
    {
```

```
            furniture = _items[id].itemPrefab;
        }

        public GameObject GetFurniture()
        {
            return furniture;
        }

    }
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.EventSystems;
using UnityEngine.XR.ARSubsystems;
using UnityEngine.XR.Interaction.Toolkit.AR;


public class InputManager : ARBaseGestureInteractable
{

    [SerializeField] private Camera arCam;
    [SerializeField] private ARRaycastManager _raycastManager;
    [SerializeField] private GameObject crosshair;

    private List<ARRaycastHit> _hits = new List<ARRaycastHit>();


    private Pose pose;

    // Start is called before the first frame update
    void Start()
    {

    }

    //Tarkistetaan onko objekti valittuna ja voidaanko aloittaa sen manipulointi
valitulla eleellä
    protected override bool CanStartManipulationForGesture(TapGesture gesture)
    {
        if (gesture.targetObject == null)
        {
            crosshair.SetActive(true);
            return true;
        }
        crosshair.SetActive(false);
        return false;

    }

    // Tätä kutsutaan, kun objektin manipulointi päätetään
    protected override void OnEndManipulation(TapGesture gesture)
    {
        if (gesture.isCanceled)
        {
            return;
        }
        if (gesture.targetObject != null || IsPointerOverUI(gesture))
        {
            return;
        }
        if(GestureTransformationUtility.Raycast(gesture.startPosition, _hits,
TrackableType.PlaneWithinPolygon))
        {
            GameObject placedObj = Instantiate(DataHandler.Instance.GetFurni-
ture(), pose.position, pose.rotation);

            var anchorObject = new GameObject("PlacementAnchor");
            anchorObject.transform.position = pose.position;
```

```
            anchorObject.transform.rotation = pose.rotation;
            placedObj.transform.parent = anchorObject.transform;
        }

    }

    // Update is called once per frame
    void FixedUpdate()
    {
        CrosshairCalculation();
    }

    // Tarkistetaan onko käyttäjän kosketus alapalkin valintaikkunan päällä, jol-
loin ei anneta luoda objektia ruudulle
    bool IsPointerOverUI(TapGesture touch)
    {
        PointerEventData eventData = new PointerEventData(EventSystem.current);
        eventData.position = new Vector2(touch.startPosition.x, touch.startPosi-
tion.y);
        List<RaycastResult> results = new List<RaycastResult>();
        EventSystem.current.RaycastAll(eventData, results);
        return results.Count > 0;
    }

    // Lasketaan "tähtäimen" paikka ruudulla ja päivitetään sitä koko ajan.
    void CrosshairCalculation()
    {
        Vector3 origin = arCam.ViewportToScreenPoint(new Vector3(0.5f, 0.5f, 0));

        if (GestureTransformationUtility.Raycast(origin, _hits, Trackable-
Type.PlaneWithinPolygon))
        {
            pose = _hits[0].pose;
            crosshair.transform.position = pose.position;
            crosshair.transform.eulerAngles = new Vector3(90, 0, 0);
        }
    }

}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[CreateAssetMenu(fileName = "Item1", menuName = "AddItem/Item")]
public class Item : ScriptableObject
{
    public GameObject itemPrefab;
    public Sprite itemImage;
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class UIContentFitter : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    //Luodaan alapalkin valintakenttä ja säädetään sen koko ruudulle sopivaksi
    public void Fit()
    {
        HorizontalLayoutGroup hg = GetComponent<HorizontalLayoutGroup>();
        int childCount = transform.childCount - 1;
        float childWidth = transform.GetChild(0).GetComponent<RectTrans-
form>().rect.width;
        float width = hg.spacing * childCount +
                      childCount * childWidth +
                      hg.padding.left +
                      childWidth;

        Vector2 size = GetComponent<RectTransform>().sizeDelta;
        GetComponent<RectTransform>().sizeDelta = new Vector2(width, 300);
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public class UIManager : MonoBehaviour
{
    private GraphicRaycaster raycaster;
    private PointerEventData pData;
    private EventSystem eventSystem;

    public Transform selectionPoint;

    public static UIManager instance;

    public static UIManager Instance
    {
        get
        {
            if (instance == null)
            {
                instance = FindObjectOfType<UIManager>();
            }
            return instance;
        }
    }

    // Start is called before the first frame update
    void Start()
    {
        raycaster = GetComponent<GraphicRaycaster>();
        eventSystem = GetComponent<EventSystem>();
        pData = new PointerEventData(eventSystem);

        pData.position = selectionPoint.position;
    }

    // Update is called once per frame
    void Update()
    {

    }

    public bool OnEntered(GameObject button)
    {
        List<RaycastResult> results = new List<RaycastResult>();
        raycaster.Raycast(pData, results);

        foreach (RaycastResult result in results)
        {
            if (result.gameObject == button)
            {
                return true;
            }
        }
        return false;
    }
}
```