LAB University of Applied Sciences

AR-SOVELLUS MUSEON NÄYTTELYYN

LAB-AMMATTIKORKEAKOULU
Insinööri (AMK)
Tieto- ja viestintätekniikka
Kevät 2021
Aleksi Salonen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Salonen, Aleksi	Opinnäytetyö, AMK	Kevät 2021
	Sivumäärä	
	36	

Työn nimi

AR-Sovellus museon näyttelyyn

Tutkinto

Tieto- ja viestintätekniikka, Ohjelmistotekniikka (AMK)

Tiivistelmä

Opinnäytetyön tavoitteena on AR-sovelluksen toteuttaminen käyttäen Unity-pelimoottorin AR Foundation-kirjastoa. Työ tehdään Suomen Moottoripyörämuseolle.

Opinnäytetyössä käsitellään laajennetun todellisuuden käsitettä. Sen eri alalajeja ovat virtuaalitodellisuus, yhdistetty todellisuus ja päivitetty todellisuus, joista jokainen poikkeaa siinä, miten ne ovat vuorovaikutuksessa todellisen ja virtuaalisen maailman kanssa ja miten käyttäjä voi olla vuorovaikutuksessa eri todellisuuksien kanssa.

Unity on kehitysympäristö sovelluksien tekemiseen lukuisille eri alustoille. Sen pääkonsepteja ovat näkymät, peliobjektit, komponentit ja skriptit. Päivitetyn todellisuuden kehittämistä varten pelimoottoriin on kehitetty AR Foundation-kirjasto, joka mahdollistaa AR-sovelluksien kehittämisen eri alustoille. Kirjastolla on mahdollista tuoda sovellukseen eri ominaisuuksia, jolla tulkita todellisesta maailmasta saatavaa tietoa ja käyttää sitä sovelluksessa.

Sovellus tehdään Suomen Moottoripyörämuseolle tulevaa Jarno Saarisen näyttelyä varten. Sen tarkoituksena on tuoda näyttelyyn lisää sisältöä AR-toiminnallisuuksien avulla, jolloin sovelluksella on mahdollista saada videomateriaalia tunnistetuista kuvista ja tuoda kolmiulotteisia kappaleita niiden todellisessa koossa.

Aikaan saatiin sovelluksen runko, joka näyttää ennalta määritettyjä kolmiulotteisia kappaleita ja pystyy vastaanottamaan verkkopalvelimelta tiedon tunnistettavista kuvista ja niiden kohdalla näytettävistä videoista. Sovellusta olisi kuitenkin mahdollista jatkokehittää lisäämällä palvelimen välityksellä näytettävää sisältöä ja parantaa käyttäjäkokemusta paremmilla kontrolleilla.

Asiasanat

Unity, AR Foundation, Augmented Reality

Abstract

Author(s)	Type of publication	Published
Salonen, Aleksi	Bachelor's thesis	Spring 2021
	Number of pages	
	36	

Title of publication

AR-Application for museum's exhibition

Name of Degree

Bachelor of Information Technology

Abstract

Goal of the thesis was to create an AR-application for museum's exhibition by using the Unity game engine and AR Foundation library. The work is done for Finland's Motorcycle Museum.

Thesis goes through the term of Extended Reality, its subgenres are Virtual Reality, Mixed Reality and Augmented Reality. They all differ with how they interact with actual reality and virtual reality and how user can interact in those realities.

Unity is a development environment which allows application development for multiple platforms. Its primary concepts are Scenes, GameObjects, Components and Scripts. For developing augmented reality applications Unity has AR Foundation-library, which allows bringing multiple features to read information from the real word and use it in the application.

Application is made for Finland's Motorcycle Museum's upcoming exhibition of Jarno Saarinen. Its purpose is to bring more to exhibition by using the AR-features such as playing videos top of recognized images or displaying the 3D-model in its actual scale.

The result was basis of an app, that can display a predetermined 3D-model and can receive the information from webserver to know, what video is played at what recognized image. An application receives all necessary information from the server, so content isn't predetermined. However, application could be developed further by increasing the amount of determined content in the server and improving the usability by better touch controls.

Keywords

Unity, AR Foundation, Augmented Reality

SISÄLLYS

1 JOHDANTO

Teknologian kehittyminen ja mukaantulo elämään on tuonut mielenkiintoisia muutoksia, joista yksi mielenkiintoinen muutos on ohjelmiston hyödyntäminen arkipäivässä. Vielä tarkemmin määriteltynä, voidaan luoda sovellus, joka käyttää hyödyksi ympäristöä ja siitä saatua tietoa, joka näytetään sovelluksen avulla kuin ne olisivat oikeassa maailmassa. Tätä kutsutaan päivitetyksi todellisuudeksi (Augmented Reality, AR).

Päivitetty todellisuus on mielenkiintoinen kehityssuunta. Uusia ideoita AR:n käyttötavoille ja käyttökohteille syntyy jatkuvasti. Esimerkiksi Acura (autovalmistaja Hondan automerkki) kehitti vuonna 2018 autolla ajettavan AR-kokemuksen, jossa käyttäjä pääsee ajamaan eksoottisella radalla ja näkemään sen virtuaalilasien avulla ajaen samaan aikaan oikeaa autoa. (Abramovich. 2020.)

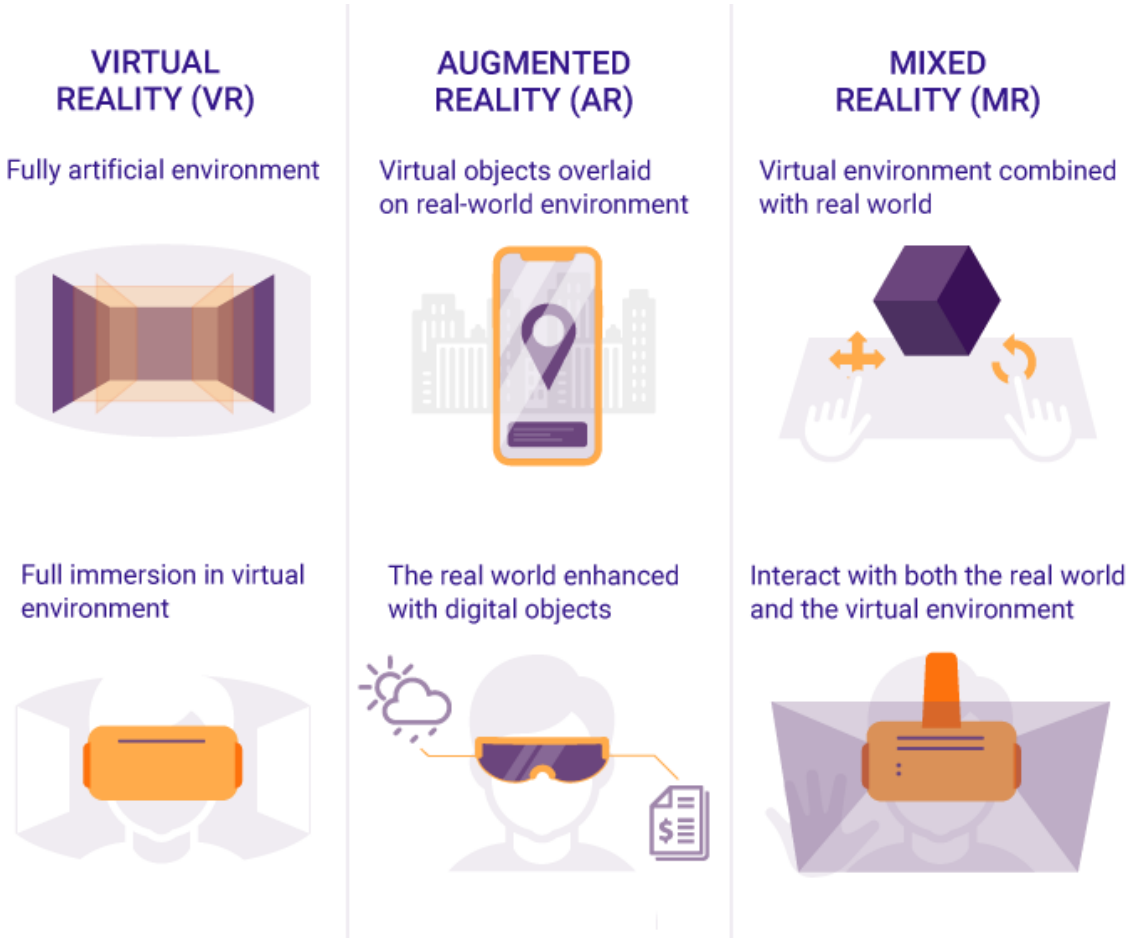
Älypuhelimet ovat ehkä kuitenkin tavanomaisin ja yleisin laite, jossa päivitetyn todellisuuden ideat ja uusien käyttötapojen keksiminen näkyy eniten. Applen iOS- ja Googlen Android-käyttöjärjestelmään tulee koko ajan lisää uusia toimintoja ja ohjelmia, jotka hyödyntävät AR-teknologiaa. Esimerkiksi Android-käyttöjärjestelmään löytyy Google Lens-ohjelma, jolla voi tutkia kamerassa näkyvää tekstiä ja kääntää se toiselle kielelle (Ranieri. 2020). Google Maps-sovellus tarjoaa karttaohjeita reaaliaikaisesti, kun ympäristöä tutkitaan kameran avustuksella (Google 2020). IKEA puolestaan on kehittänyt ohjelman, jolla on mahdollista kokeilla ja sommitella tuotteita esimerkiksi kotiympäristössä sisustamisen avuksi ja ostopäätöksien helpottamiseksi (Ikea 2020).

Työn tavoitteena on luoda Suomen Moottoripyörämuseoon sovellus tulevaan Jarno Saarisen näyttelyyn. Sovelluksen tarkoituksena on kertoa Saarisen henkilöhistoriaa ja tuoda AR-sovelluksen kautta tutkittavaksi näyttelykohteita ja näyttää materiaalia tämän näyttelyn henkilön historiasta, kuten esimerkiksi videomateriaalia ja kolmiulotteisia kappaleita. Projektin tavoitteena on myös luoda palvelin tukemaan itse varsinaista puhelinsovellusta, jota kautta voitaisiin hallita sovelluksessa näytettävää sisältöä. Opinnäytetyö keskittyy puhelinsovelluksen päivitetyn todellisuuden osuuden toteuttamiseen. Opinnäytetyössä esitellään laajennetun todellisuuden käsite ja eri todellisuudet. Lisäksi työ käy läpi Unity-pelimoottorin toimintaperiaatteita ja sen päivitetyn todellisuuden kehittämiseen tarkoitetun AR Foundation-kirjaston toimintaa.

2 LAAJENNETTU TODELLISUUS

2.1 Laajennetun todellisuuden käsite

Lisätty todellisuus on yksi osa laajennetun todellisuuden (Extended Reality, XR) käsitettä. Tämä termi käsittää virtuaalitodellisuuden (Virtual Reality, VR), lisätyn todellisuuden, sekä yhdistetyn todellisuuden (Mixed Reality, MR). Lisätyn todellisuuden käsite käsittää kaikki ympäristöt, missä yhdistyy virtuaalimaailma sekä ihmisen ja koneen välinen interaktio. (North of 41 2018.)



Kuva 1. Havainnollistamisesimerkki laajennetuista todellisuuksista (Gleb. B. 2020)

Lisätyn todellisuuden ympäristön on jaettu karkeasti näihin kolmeen ryhmään. Karkeasti kuvailtuna virtuaalitodellisuus on luodun maailman kanssa toimimista. Yhdistetty todellisuus sekoittaa virtuaalitodellisuuden ja todellisuuden ja mahdollistaa molempien kanssa vuorovaikuttamisen. Lisätty todellisuus lisää virtuaalisia ominaisuuksia ympäristöömme, mutta ei salli interaktiota näiden kanssa (Kuva 1). (North of 41 2018.)

2.1.1 Virtuaalitodellisuus

Virtuaalitodellisuudessa (Virtual Reality, VR) käyttäjä viedään kokonaan omaan maailmaan. Virtuaalitodellisuus yleensä edellyttää erityisiä päätelaitteita, jotta virtuaalimaailman näkeminen ja kokeminen on mahdollista. Tyypillisimpiä laitteita tähän tarkoitukseen on erilaiset virtuaalilasit. Tällaisia laitteita ovat esimerkiksi Facebookin kehittämä Oculus Rift ja Valven Valve Index VR-virtuaalilasit. Molemmat laitteet ovat puettavia laseja, joiden avulla käyttäjä näkee virtuaalitodellisuuden ympärillään. Lisäksi molemmat laitteet hyödyntävät käteen puettavia antureita, joilla on mahdollista tunnistaa käsien liikkeet virtuaalitodellisuudessa ja vuorovaikuttaa erilaisten objektien kanssa, kuten esimerkiksi käyttää virtuaalisia esineitä käsissään.

Yksi suosittu virtuaalitodellisuuden käyttökohde on pelit ja yhtenä esimerkkinä on Valven kehittämä Half Life: Alyx – videopeli. Opetuksessa puolestaan käytetään virtuaalitodellisuutta esittelemään esimerkiksi laitteistojen toimintaa ja niiden todellista mittakaavaa ja maantieteellisiä paikkoja virtuaalitodellisuuden avulla. (Gleb B. 2020.)

2.1.2 Yhdistetty todellisuus

Yhdistetty todellisuus (Mixed Reality, MR) yhdistää virtuaalitodellisuuden ja todellisuuden. MR:ssä tietokoneen luomat objektit näkyvät oikeassa maailmassa. Yhdistetyssä todellisuudessa on mahdollista vuorovaikuttaa sekä todellisten että virtuaalisten kappaleiden kanssa ympäristössä, jonka näemme yhdistetyn todellisuuden avulla. Tämä on tärkeää, jotta käyttäjä kokee maailman aidoksi. Toisin sanoen yhdistetyssä todellisuudessa yhdistyy ihmisen ja tietokoneen välinen interaktio ympäristön avulla. (Microsoft. 2020.)

Yhdistetyn maailman kokemiseen ja näkemiseen tarvitaan esimerkiksi päälle puettavat virtuaalilasit. Tähän käyttötarkoitukseen tarkoitettujen lasien avulla on mahdollista nähdä tietokoneen luomat virtuaaliset kappaleet. Yhdistetyn todellisuuden lasit voivat pitää maailman niin lähellä todellista maailmaa kuin halutaan, tai vaihtoehtoisesti korvata ympäristön lähes kokonaan virtuaalitodellisuudella. Tällaisia laitteita ovat esimerkiksi Microsoftin HoloLens ja Samsungin Odysseus-virtuaalilasit. (Microsoft 2020; Marr 2020.)

Käyttökohteita yhdistyneelle todellisuudelle ovat esimerkiksi suunnittelutyö. Autovalmistaja Ford hyödyntää yhdistettyä todellisuutta autojen ulkonäön ja teknisen toteutuksen suunnitellussa. (Marr 2020.)

2.1.3 Päivitetty todellisuus

Päivitetyssä todellisuudessa olemassa oleva maailma päivitetään tietokoneen avulla, esimerkiksi grafiikan, äänen tai GPS-datan avulla. AR-ympäristö hyödyntää olemassa olevaa maailmaa ja esittää informaatiota tämän avulla. Se ei pyri korvaamaan tai muokkaamaan olemassa olevaa ympäristöä muiden laajennettujen todellisuuksien tavoin. Päivitetty todellisuus lisää todellisuuteemme ominaisuuksia ja informaatiota päätelaitetta hyödyntäen. (North of 41 2020.)

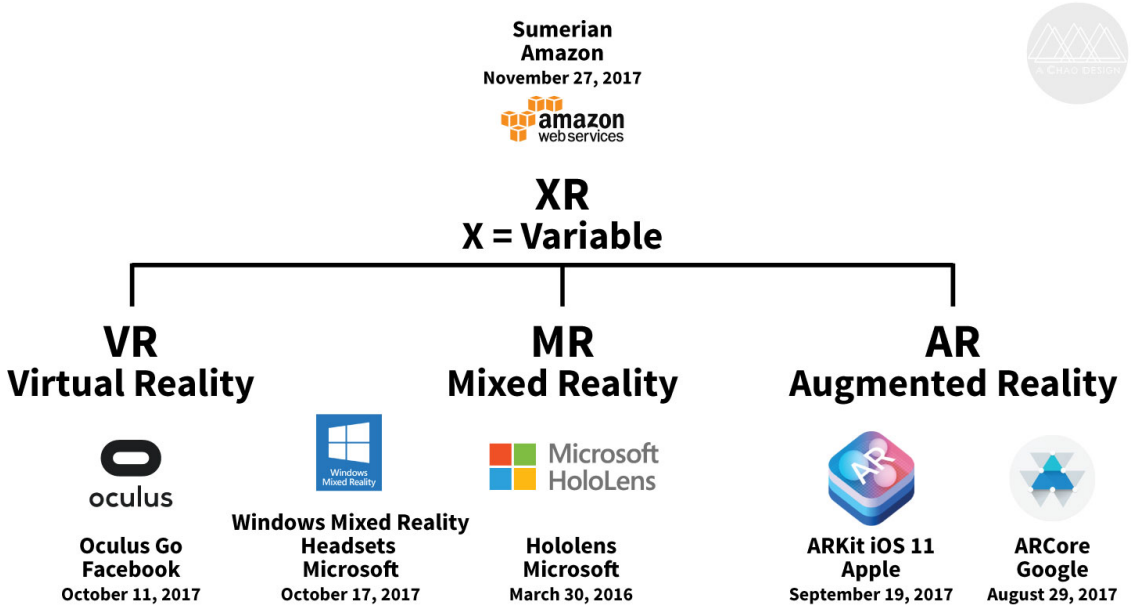
Päivitettyä todellisuutta hyödynnetään erityisten lasien avulla tai VR-päätelaitteilla, jotka näyttävät todellista maailmaa ohjelman tuoman informaation lisäksi. Tällaisia laitteita ovat esimerkiksi Oculus Rift VR-virtuaalilasit. (Noble. 2019.)

On olemassa kuitenkin yksi yleinen päivitetyn todellisuuden laite: älypuhelin. Nykypuhelimien teknologia ja sen sisältämät sensorit mahdollistavat laitteiden hyödyntämisen monipuolisesti AR-sovelluksien kanssa.

2.2 Päivitetty todellisuus puhelimessa

Puhelimista on tullut yleisin laajennetun todellisuuden käyttökohde, ehkä osin jopa huomattamme. Yksi syy on se, miten helposti puhelin kulkee mukanamme, se on kevyt ja helppo käyttää jopa liikkuessamme. Toinen merkittävä syy on nykypuhelimien teknologinen taso. Puhelimista on tullut erittäin tehokkaita näyttämään grafiikkaa ja suorittamaan monimutkaisempiakin tehtäviä. Lisäksi puhelimissa on lukuisia sensoreita, joilla voi lukea oikean ympäristön tilaa. Dataa pystytään hyödyntämään päivitetyn todellisuuden sovelluksissa. Tämän takia moni AR-sovellus ei tarvitse erityisiä päätelaitteita toimiakseen. AR-sovellukset ovat helposti kehittävissä puhelinsovelluksiksi, ja siten jaettavissa helposti. (Craig, A. B. 2013.)

Älypuhelimilla on mahdollisuus hyödyntää lukuisia sisäänrakennettuja sensoreita kuten GPS-antureita, mikrofonia ja liiketunnistusta. Näistä yksi yleinen hyödynnetty sensori (ja samalla ehkä tärkein) on kuitenkin puhelimen kamera. Päivitetyn todellisuuden sovelluksien kehittämisen avuksi on Android- ja iOS-käyttöjärjestelmiin kehitetty ARCore- ja AR-Kit-ohjelmistokehykset vastaavasti (Kuva 2).



Kuva 2. Eri laajennetun todellisuuden laitteita ja teknologioita (Achao Design 2017)

2.2.1 iOS ja ARKit

ARKit on Applen kehittämä ohjelmistokehys AR-sovelluksia varten iOS-käyttöjärjestelmään. Vaatimuksena ovat iOS 11.0 tai uudempi käyttöjärjestelmä ja A9-suoritin tai uudempi. (Apple 2020)

ARKitin tarkoituksena on lisätä kaksi- ja kolmiulotteisia kappaleita puhelimen näkymään kuin ne olisivat osana oikeaa maailmaa. ARKit hyödyntää tähän tarkoitukseen liikkeentunnistusta, kameran avulla kuvattua ympäristöä ja ympäristön prosessointia, jotta näiden elementtien tuominen AR sovellukseen olisi todenmukaisempaa. (Apple 2020.)

2.2.2 Android ja ARCore

ARCore on Android-käyttöjärjestelmää varten kehitetty ohjelmistokehys AR-sovelluksia varten. ARCore on kokoelma rajapintoja, jotka mahdollistavat puhelimen havainnoida ympäristöään, ymmärtämään ja toimimaan ympäristöstä saadun tiedon avulla. ARCore tukee myös iOS käyttöjärjestelmää. ARCore on kuitenkin pääsääntöisesti kehitetty Android käyttöjärjestelmän version 7 (Nougat) ja sitä myöhempiä versioita varten. (Google 2020.)

ARCore:lla on kolme avaintoiminnallisuutta. Liikkeentunnistus, jonka tehtävä on auttaa laitetta ymmärtämään sen suhteellista sijaintia ympäristössämme. Ympäristön ymmärtäminen, joka mahdollistaa vaaka- ja pystysuorien tasojen tunnistamisen ja orientaation. Ja viimeisenä valaistuksen arviointi, jotta laite ymmärtää ympäröivän valaistuksen. (Google 2020.)

3 UNITY JA AR FOUNDATION

3.1 Unity

Unity on 3D kehitysympäristö ohjelmistojen kehittämiseen lukuisille eri alustoille (mm. Windows, MacOS, Android, iOS). Unityn lisenssi on ilmainen, mutta jos Unityllä tehdyt ja julkaistut pelit tuottavat rahaa riittävästi, on maksettava lisenssistä pelien tulojen mukaan. Yrityskäyttö edellyttää myös maksettavaa lisenssiä, jos yrityksen liikevaihto on tarpeeksi suuri.

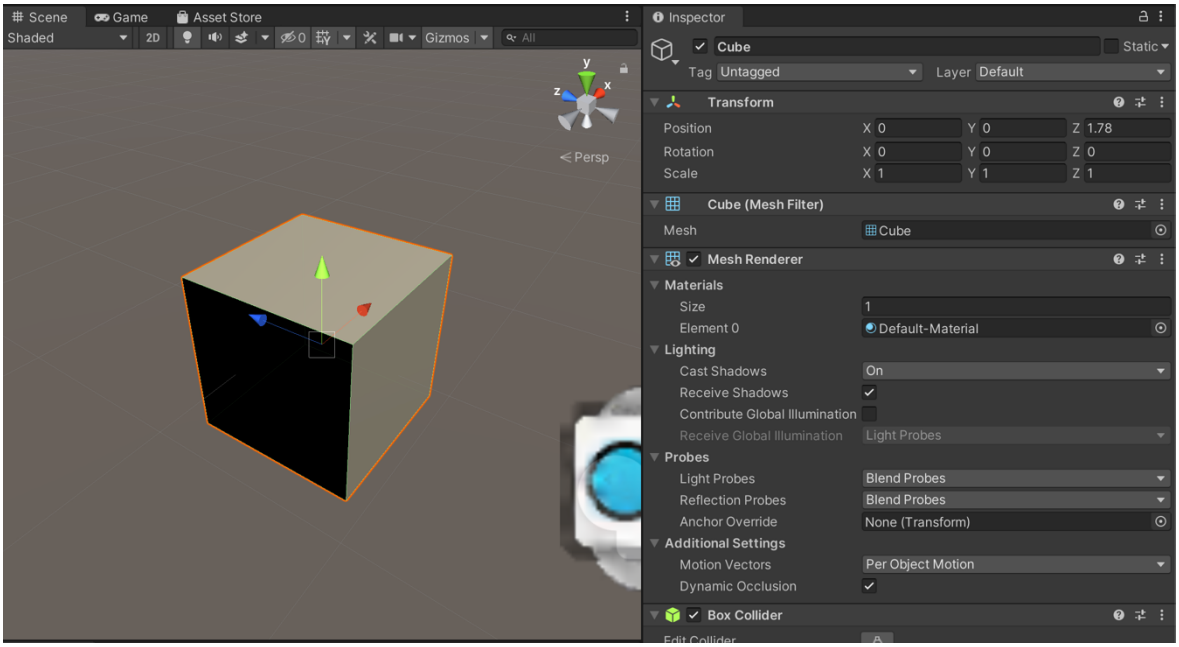
Unityn editorilla työskentely voidaan jakaa kolmeen tärkeimpään ominaisuuteen: Näkymien hallintaan, peliobjekteihin ja skripteihin.

3.1.1 Näkymät

Näkymät (Scenes) ovat Unityn pelimaailmoja ja valikoita. Näkymät sisältävät kokoelman objekteja, jotka sisältävät varsinaisen datan pelistä. Näkymän tehtävänä käytännössä ei siis ole tarkoitus suorittaa mitään toimintoja, vaan ylläpitää informaatiota. (Unity 2020.)

3.1.2 Peliobjektit

Peliobjektit (GameObjects) ovat Unityn tärkein konsepti. Jokainen kappale pelissä on peliobjekti, oli se sitten esine, hahmo, ääni, kamera tai erikoistehoste. Peliobjektit ei sinällään itse sisällä mitään dataa, vaan toimivat säilönä erilaisille komponenteille. Komponentit ovat peliobjektin osa, jotka määrittävät kappaleen toiminnallisuutta (Kuva 3). Kuvassa 3 nähdään valittu peliobjekti, jonka sisältämät komponentit ovat nähtävissä editorin Inspector-välilehdellä. Komponenttien arvoja voidaan muokata vapaasti ja peliobjektille voidaan lisätä uusia komponentteja tässä näkymässä. (Unity 2020.)



Kuva 3. Vasemmalla näkymässä näkyvä peliobjekti ja oikealla peliobjektin komponentit

Editorissa on valmiina lukuisia valmiita komponentteja, mutta mikäli peliobjekti kaipaa lisää toiminnallisuuksia, on mahdollista hallita kappaleen toimintaa skripteillä (Scripts). (Unity 2020.)

3.1.3 Komponentit

Komponentit (components) ovat peliobjektin osia, jotka määrittävät peliobjektin toimintaa. Peliobjekteilla voi lukuisia komponentteja, joita on mahdollista hallinnoida Unityn editorista tai skriptien avulla. Jokaiseen peliobjektiin voi kuulua lähes rajaton määrä komponentteja, mutta jokaisella peliobjektilla on vähintään transform-komponentti, joka käsittelee kappaleen sijaintia, skaalausta ja orientaatiota. Tätä komponenttia ei voi poistaa. (Unity 2020.)

3.1.4 Skriptit

Skriptit (Scripts) ovat komponentteja, joiden tarkoituksena on hallita objektien toimintaa muokkaamalla sen omia, tai muiden komponenttien ominaisuuksia. Skriptit voivat käsitellä mitä tahansa tehtävää kappaleen kiihtyvyyden laskemisesta jonkin erikoistehosteen toistamiseen. (Unity 2020.)

Yksinkertaisuudessaan, skripti suoritetaan peliobjektin käynnistyessä ja se suorittaa sille asetettua tehtävää jokaisen ruudunpäivityksen tai fysiikkamoottorin päivitykseen yhteydessä. Skripti voi toimia vielä myös siihen kuuluvan peliobjektin poistuessa käytöstä.

3.1.5 MonoBehaviour

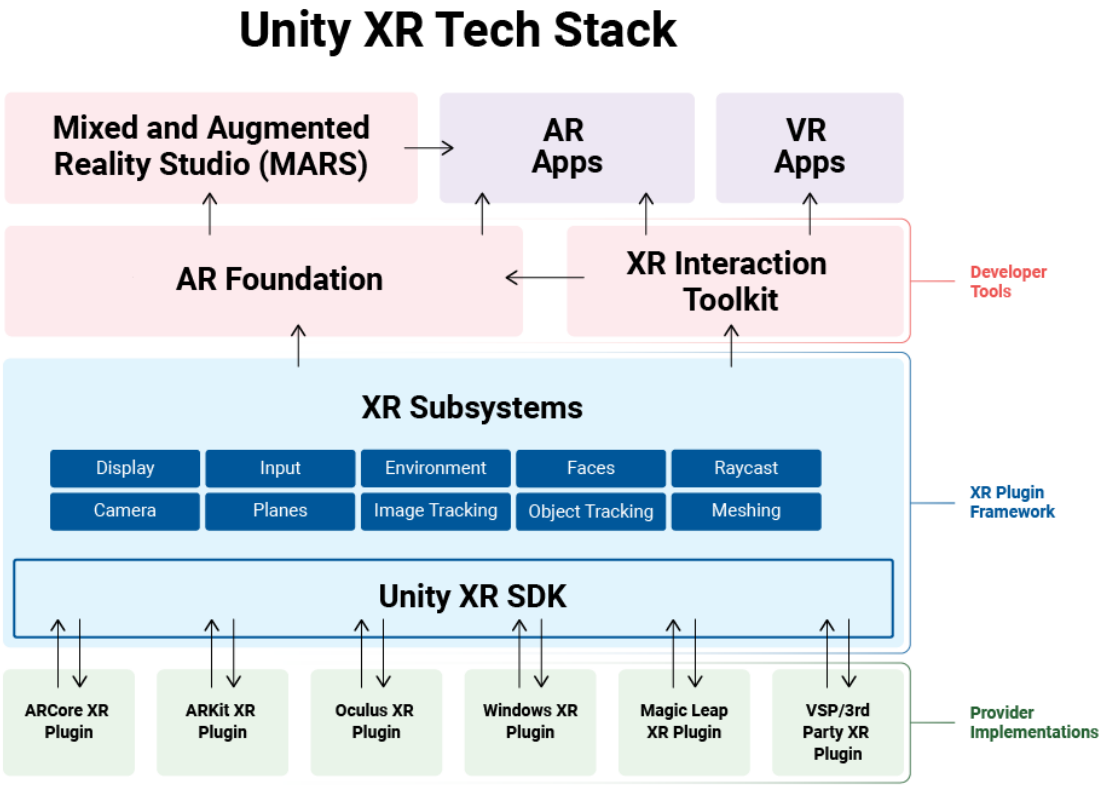
MonoBehaviour on ”pääluokka”, minkä kaikki Unityn skriptit perivät automaattisesti niitä luodessa. Tämän luokan tarkoituksena on tarjota runko skripteille, mikä mahdollistaa skriptin liittämisen peliobjektiin ja sitä kautta mahdollistaa tapahtumankäsittelijöiden käytön, millä hallita peliobjektia. Tällaisia tapahtumia ovat esimerkiksi ruudunpäivitykset ja peliobjektin tullessa kameran näkyviin. (Unity 2020.)

3.2 AR Foundation

AR Foundation on Unityn kirjasto, joka mahdollistaa päivitetyn todellisuuden sovelluksien kehittämisen Unityn pelimoottorilla. AR Foundation ei itsessään sisällä mitään AR toiminnallisuuksia, vaan tarjoaa rajapinnan, millä pelimoottori keskustelee eri laitteiden omien AR-rajapintojen kanssa. Kohdelaitteen todellisesta rajapinnasta ja ominaisuuksista riippuen käytössä on eri määrä ominaisuuksia. (Unity 2020.)

AR Foundation on yhdistelmä Unityn MonoBehaviour-luokan ominaisuuksia ja omia rajapintoja, joita käytetään keskustelemaan eri laitteiden toiminallisuuksien kanssa. Tällaisia toiminallisuuksia ovat

- laitteen seuranta ja orientaatio oikeassa maailmassa
- vaaka- ja pystysuorien tasojen tunnistus
- sijaintipisteiden tunnistus
- referenssipisteiden tunnistus, mielivaltaisesti päätettyjen pisteiden sijainti ja orientaatio, jota laite seuraa
- valon arviointi, värin ja kirkkauden tunnistus oikeassa maailmassa
- ympäristöantureiden käsittely, kuutiokartan luominen oikeasta maailmasta esittämistä varten
- kasvotunnistus
- kuvantunnistus
- kappaleiden tunnistus. (Unity 2020.)



Kuva 4. Unityn laajennetun todellisuuden arkkitehtuuri (Unity 2020)

Subsystems ja Provider Plug-init

AR Foundation perustuu osajärjestelmiin (subsystems). Nämä ovat alustasta riippumattomia rajapintoja, jotka mahdollistavat eri ominaisuuksien käyttämisen kuten esimerkiksi tasojen tunnistuksen. Jokainen osajärjestelmä käsittelee ainoastaan yhtä määritettyä tehtävää (Kuva 4). (Unity 2020.)

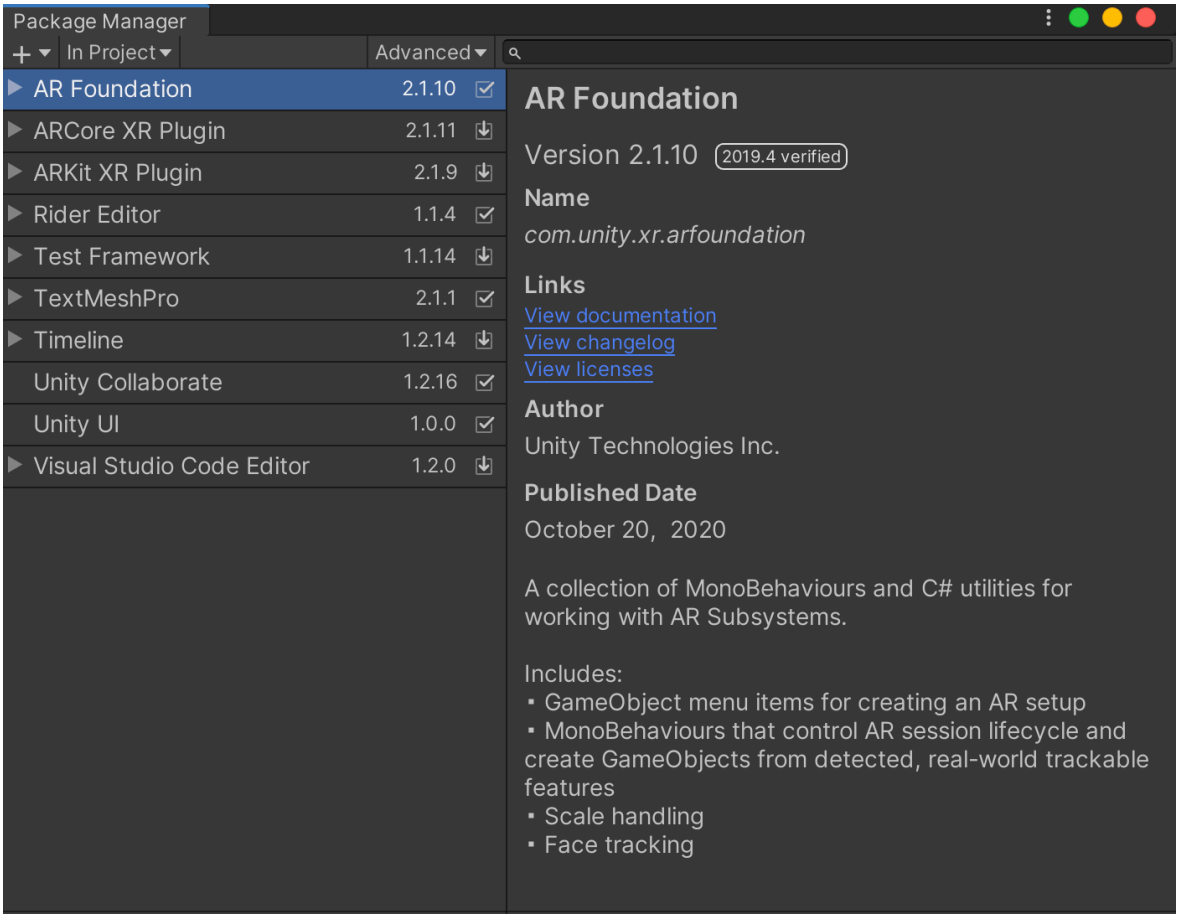
Provider Plug-init ovat todellisia implementaatioita osajärjestelmän osista. Toisin sanoen jokainen Provider-liitännäinen (Provider plug-in) sisältää todellisen implementaation osajärjestelmän toiminnallisuudesta. Nämä liitännäiset on tarkoitettu laajennetun todellisuuden ominaisuuksien hyödyntämiseen eri kohdelaitteissa ja käyttöjärjestelmissä. (Unity 2020.)

3.3 Työskentely AR Foundationilla

3.3.1 Liitännäiset

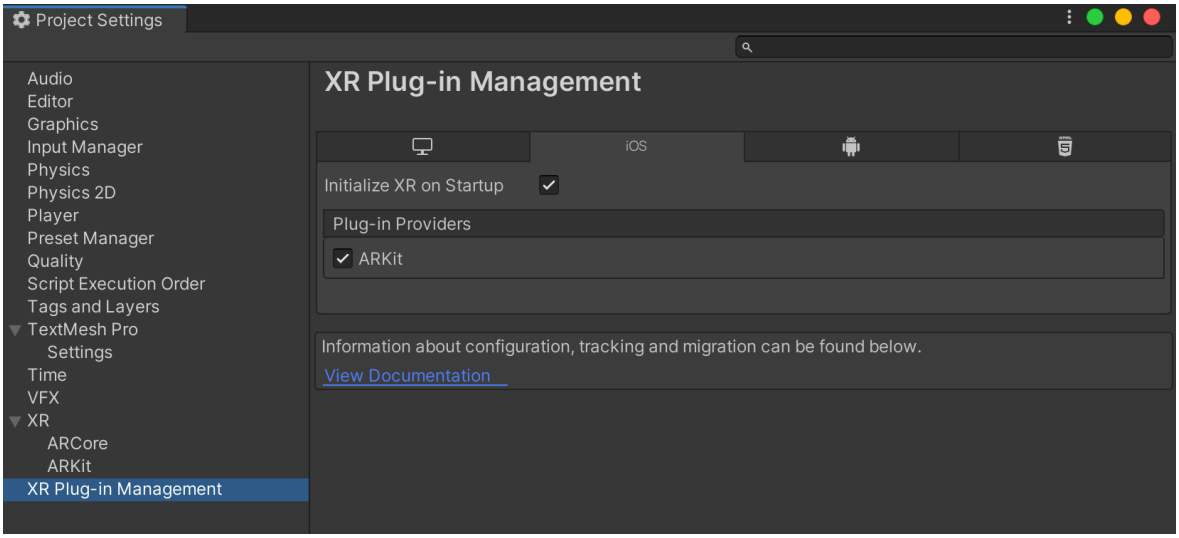
Laajennetun todellisuuden työkalujen kanssa työskentely edellyttää Unityn kehitysympäristössä tarvittavien kirjastojen asentamista Package Manager:sta (Kuva 5). Asennettavat kirjastot ovat AR Foundation ja ainakin yksi kohdelaitteeseen tarvittava liitännäinen.

ARCore XR Plugin on laajennetun todellisuuden liitännäinen Android-käyttöjärjestelmään ja ARKit XR Plugin vastaavasti iOS-käyttöjärjestelmään.



Kuva 5. Unity Package Manager ja AR-liitännäiset

AR Foundationin versiosta 4.0 alkaen on lisäksi valittava Plug-in Provider kohdelaitteisiin. Osaan kohdelaitteista on tarjolla useampi valittava Provider laajennetun todellisuuden toiminallisuuksia varten (Kuva 6).



Kuva 6. Projektin asetuksien Plug-in Provider valintaikkuna AR Foundationin versiolle 4.0 ja eteenpäin

3.3.2 AR Session

Päivitetyn todellisuuden ominaisuuksia varten varatun näkymän täytyy sisältää AR Session komponentti. AR Session-komponentti käsittelee päivitetyn todellisuuden ominaisuuksien elinkaarta ja sen asetuksia. AR Session komponentteja tarvitsee olla vain yksi. (Unity 2020.)

AR Session-komponentin ylläpitämä sessio on aktiivinen niin, kauan kun komponentti itse on aktiivinen. Komponentti voidaan poistaa käytöstä, jolloin sovellus lopettaa päivitetyn todellisuuden ominaisuuksien käyttämisen. Komponenttia asettaessa takaisin aktiiviseksi, sovellus pyrkii palauttamaan kaikki sen aikaisemmin tunnistamat ominaisuudet, kuten esimerkiksi tunnistetut tasot ja niiden sijainnit. Komponentilla voidaan myös määrittää, pyrkiikö AR Session-komponentti arvioimaan ympärillä olevaa valoa valaistuksen tekemisessä ja mahdollisesti tarvittavien päivitetyn todellisuuden ohjelmistojen asennusta laitteeseen. (Unity 2020.)

3.3.3 AR Session Origin

AR Session Origin-komponentin tarkoitus on kääntää tunnistettavat asiat (kuten esimerkiksi kappaleet ja tasomaiset pinnat) niiden ”sijaintiin” Unityn näkymässä. AR Session Originin-komponentin tehtävä on myös kääntää tunnistettujen kohteiden orientaatio ja koko pelimoottorin ymmärtämäksi. AR Session Origin-komponentti ylläpitää suhteellista ”tilaa” ja kääntää kaikki näytettävät kappaleet näkymän omaan suhteelliseen koordinaatistoon. (Unity 2020.)

AR Session Origin-komponentin sisältämän peliobjektin alaisuuteen sisältyy (tai täytyisi sisältyä) AR Camera-objekti, jonka tarkoitus on toimia varsinaisena kamerana näkymässä. Unityn näkymässä yksi skaalan (scale) mittayksikkö vastaa yhtä metriä todellisella maailmassa. Yhden mittayksikön kuutio olisi siis metrin kokoinen kappale oikeassa maailmassa. Kameran transform-komponentin skaalaa muuttamalla on kuitenkin mahdollista muokata näkymään ilmaantuvien kappaleiden kokoa. Isommalla kameran skaalalla, näytettävät kappaleet näkyvät pienempänä ja pienemmällä skaalalla päinvastoin. (Unity 2020.)

3.3.4 Trackables

Trackables-komponentit ovat osa AR Session Origin-komponenttia. Nämä osat ovat mitä tahansa osia, mitä AR Session Origin-komponentin täytyy seurata ja tunnistaa kameran avulla. Näitä ovat esimerkiksi tasot, kasvot ja kappaleet todellisessa maailmassa. Jokainen Trackable-komponentti tunnistaa vain sille määritettyjä kohteita ja jokaisen trackable-komponentin olla samassa peliobjektissa, missä AR Session Origin-komponentti sijaitsee. Trackable-komponentteja on mahdollista ottaa käyttöön, poistaa käytöstä ja lisätä tai poistaa vapaasti session aikana. (Unity 2020.)

3.3.5 Raycasting

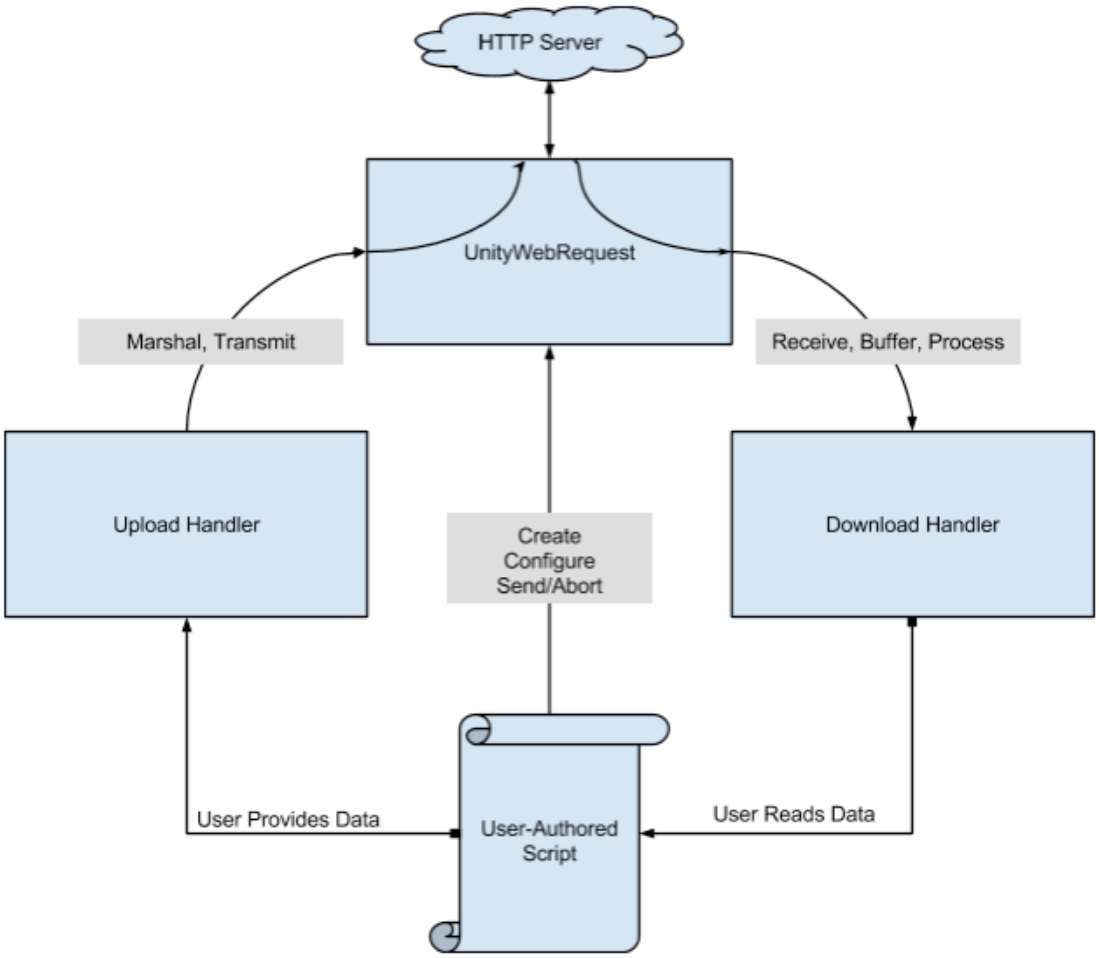
Raycasting tunnetaan myös osumatarkistuksena. Raycasting tarkistaa, missä kuvitteellinen säde kohtaa Trackable-komponentin esittämän peliobjektin. Raycasting-komponentin täytyy kuulua samaan peliobjektiin, missä AR Session Origin-komponentti sijaitsee toimiakseen. Tunnistus tapahtuu todellisesta maailmasta. (Unity 2020.)

AR Foundationin osumatarkistusta käytetään skripteissä tavallisen osumatarkistuksen sijaan. Raycasting palauttaa vastauksena vain AR Session Origin-komponentin tunnistamia kohteita. AR Foundationin Raycasting tukee kirjoitushetkellä tasojen ja pistekarttojen tunnistusta. (Unity 2020.)

3.4 UnityWebRequest

UnityWebRequest on Unityn sisältämä kirjasto, joka mahdollistaa HTTP-pyyntöjen suorittamisen skriptien avulla. Tämä mahdollistaa sovelluksen kommunikoinnin palvelimien kanssa, jotka on kehitetty tavanomaisia verkkopalveluita varten. UnityWebRequest-kirjasto tukee täysin HTTP-otsikkotietojen (headers) määrittämistä ja kaikkia HTTP-operaatioita. Lisäksi kirjasto tukee myös paloiteltuja HTTP-pyyntöjä ja POST/PUT-operaatioiden

suoratoistamista. Ohjelmointikirjasto tarjoaa korkean tason rajapinnan yksinkertaisiin pyyntöihin, sekä matalan tason rajanpinnan vaativampiin operaatioihin. (Unity 2020.)



Kuva 7. UnityWebRequest-kirjaston arkkitehtuuri (Unity 2020)

UnityWebRequest-kirjasto voidaan jakaa kuvan 7 arkkitehtuurikuvauksen mukaan kolmeen osaan. Ensimmäiset kaksi osaa ovat saapuvan ja lähtevän datan lataustenkäsittelijät, jotka suorittavat nimensä mukaisia tehtäviä. Kolmas osa on UnityWebRequest-objekti. Tämä objekti hallinnoi molempia latauskäsittelijöitä ja itse varsinaisen pyynnön asetuksia, kuten URL-osoitetta, HTTP-otsikkotiedostoja ja virhetilanteita (Kuva 7). Käsittelijöiden tarkoitus on tarjota valmiit toiminnot, joita voidaan hallinnoida käyttäjän tekemän skriptin avulla. (Unity 2020.)

UnityWebRequestin käyttäminen

UnityWebRequest-kirjastoa on mahdollista käyttää apuna tiedon hakemisessa ja lähettämisessä skripteissä. Yksinkertaisimmillaan HTTP-pyyntö on mahdollista tehdä valmiilla korkean tason ohjelmointikirjastolla. Yksinkertainen GET-pyyntö tapahtuisi

UnityWebRequest luokan GET-metodilla. Korkean tason ohjelmointirajapinnan kanssa ei tarvitse huolehtia kirjaston latauskäsittelijöiden käyttämisestä. Paluuviestinä voidaan vastaanottaa tekstiä, jonka sisältö voidaan parsia käyttötarpeen mukaan (Kuva 8).

```
22      IEnumerator GetRequest(string uri)
23      {
24          using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))
25          {
26              yield return webRequest.SendWebRequest();
27
28              string[] pages = uri.Split('/');
29              int page = pages.Length - 1;
30
31              if (webRequest.isNetworkError)
32              {
33                  Debug.Log(pages[page] + ": Error: " + webRequest.error);
34              }
35              else
36              {
37                  Debug.Log(pages[page] + ":\nReceived:");
38                  Debug.Log(webRequest.downloadHandler.text);
39              }
40          }
41      }
```

Kuva 8. UnityWebRequest Get-metodin käyttäminen

Monimutkaisempia pyyntöjä varten on mahdollista käyttää matalan tason rajapintaa UnityWebRequest-objektista. Tällöin on mahdollista hallinnoida enemmän tehtävää HTTP-pyyntöä. Esimerkkinä, UnityWebRequest ei sisällä valmista metodia POST-operaation tekemiseen käyttäen JSON-objektia. Tässä tapauksessa on kuitenkin mahdollista alustaa kaikki tarvittava tieto manuaalisesti kuvan 9 esimerkin mukaisesti, jossa luodaan omat latauksenkäsittelijät ja lisätään omat otsikkotiedot pyynnön tekemiseksi (Kuva 9).

```
43      public IEnumerator PostRequest(string url, string postDataJsonString)
44      {
45          var request = new UnityWebRequest(url, "POST");
46          byte[] bodyRaw = Encoding.UTF8.GetBytes(postDataJsonString);
47          request.uploadHandler = (UploadHandler)new UploadHandlerRaw(bodyRaw);
48          request.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
49          request.SetRequestHeader("Content-Type", "application/json");
50          yield return request.SendWebRequest();
51
52          if (request.isNetworkError)
53          {
54              Debug.Log("Error: " + request.error);
55          }
56          else
57          {
58              Debug.Log(request.downloadHandler.text);
59          }
60
61      }
62  }
```

Kuva 9. UnityWebRequest-luokan käyttö matalan tason rajapinnalla

4 TOIMINTAYMPÄRISTÖN ESITTELY

4.1 Suomen Moottoripyörämuseo

Työn asiakkaana toimii Suomen Moottoripyörämuseo. Moottoripyörämuseo avattiin maaliskuussa vuonna 2011 ja se sijaitsee Vesijärven rannalla Lahdessa. Museo on rakennettu aikanaan toimineen vanhan sahan puukuivamoon. Rakennuksen remontointi on rahoitettu eri yrityksien, tahojen ja yksityishenkilöiden tuella. (Moottoripyörämuseo 2020.)

4.2 Työn tavoite

Sovellus tehdään Jarno Saarisen näyttelyä varten. Näyttely käy Saarnisen elämänhistoriaa läpi ja sovelluksen on tarkoituksena toimia näyttelyn tukena tämän elämän läpi käymiseksi. Sovellus esittelee henkilön historiaa, tämän saavutuksia ja sitä, millaisen jäljen Saarinen jätti suomen moottoripyöräurheiluun.

Lisäksi sovelluksessa on päivitetyn todellisuuden osuus, jolla on mahdollista tutkia näyttelyä ja siten saada lisää tietoa ja materiaalia Saarisen elämästä videoiden ja kolmiulotteisten sisällön muodossa. Tutkittavia asioita näyttelyssä ovat näytteille asetettavat kuvat ja päivitettyyn todellisuuteen on mahdollista tuoda Jarno Saarisen aikanaan käyttämä moottoripyörä, jota on mahdollista tutkia todellisessa mittakaavassa. Päivitetyn todellisuuden tarkoituksena on laajentaa näyttelykokemusta tuomalla sinne materiaalia, mikä ei välttämättä fyysisesti mahtuisi kerralla koko näyttelyyn, ja samalla luoda mielenkiintoinen näyttelykokemus.

Sovelluksessa on kaksi osaa; elämänkertaosuus ja päivitetyn todellisuuden osuus. Elämänkertaosuus sisältää tietoa Jarno Saarisen elämästä ja tämän saavutuksista. Tämän osuuden on tarkoitus puolestaan laajentaa näyttelyä itseään tuomalla näkyviin esimerkiksi videoita kuvantunnistuksen avulla ja mahdollistamaan esineiden näyttämistä kolmiulotteisina kappaleina, etenkin sellaisia kappaleita, joita ei enää välttämättä ole olemassa.

5 PROJEKTIN ESITTELY

5.1 Käyttäjän sovellus

Käyttäjän sovellus tehdään Unitylla. Projektiin on jo entuudestaan tehty historiaosuus Jarno Saarisesta, jota hyödynnetään tässä projektissa. Käyttäjän sovelluksen kehitystä jatketaan historiaosuuden puolella muutamalla korjauspäivityksellä, mutta tässä työssä esitellään vain päivitetyn todellisuuden osuus.

Sovellukseen perustetaan myös päivitetyn todellisuuden puoli, jolla näytetään lisää historiallista materiaalia. Tämä puolen sisältö tulisi olla myös hallittavissa palvelimelta saadun tiedon avulla, missä yhteydessä materiaali näkyy päivitetyssä todellisuudessa. Kuten esimerkiksi kiinteässä paikassa tai tunnistetun kuvan päällä. Tämän työn tarkoituksena on tehdä päivitetyn todellisuuden osuus.

Sovelluksessa on tarkoitus hyödyntää Unityn WebRequest-kirjastoa, jolla on mahdollista suorittaa HTTP-pyyntöjä. Tällä on tarkoituksena hakea tarvittavaa tietoa sovellukseen palvelimelta, ja siten tehdä sovelluksen sisällöstä hallittava, ja mukautettava tarpeen mukaan.

5.1.1 AR Foundation

AR Foundation on Unityn laajennetun todellisuuden kirjasto. Tämän kirjaston avulla sovellukseen tuodaan päivitetty todellisuus. Kun Unity-projekti käännetään sovellukseksi, tämä kirjasto toteuttaa automaattisesti tarvittavat toimenpiteet, jotta mobiilisovellus hyödyntää Androidin AR Core-kirjastoa tai iOS:n ARKit-kirjastoa.

Päivitettyä todellisuutta olisi tarkoitus hyödyntää erityisesti kahdella AR Foundationin ominaisuudella; tasontunnistuksella, ja kuvien tunnistuksella. Tasontunnistuksella on tarkoitus tunnistaa lattiatasoja, johon voidaan lisätä kolmiulotteinen kappale.

Kuvantunnistuksella on tarkoitus tuoda lisää sisältöä näyttelyyn käytössä olevien kuvien avulla. Kuvantunnistuksesta ja näytettävästä sisällöstä on tarkoitus tehdä mahdollisimman paljon hallittavia palvelimelta saatavan tiedon avulla.

5.2 Palvelin

Palvelimen tarkoituksena on tallentaa ja ylläpitää kaikkea sovelluksen tarvitsemaa tietoa, jota on mahdollista hallinnoida palvelimen avulla. Käyttöön tuleva palvelin toteutetaan verkkosovelluksille tyypillisellä palvelinratkaisulla, joka välittää tietoja sovelluksen tekemien HTTP-pyyntöjen avulla sisällön näyttämiseksi. Tämän työn tavoitteena ei kuitenkaan

ole palvelimen perustaminen, mutta työssä esitellään palvelimelta vastaanotettavaa tietoa ja sen käyttöä. Palvelin voi itsessään olla melko yksinkertainen. Sovellus ei sisällä käyttäjätietojen hallintaa, eikä sovelluksessa käsitellä mitään muutakaan arkaluontoista tietoa, joten sovelluksen ja palvelimen välille ei tarvitse erityistä käyttäjätunnistusta.

6 PUHELINSOVELLUS

6.1 Unity-projekti

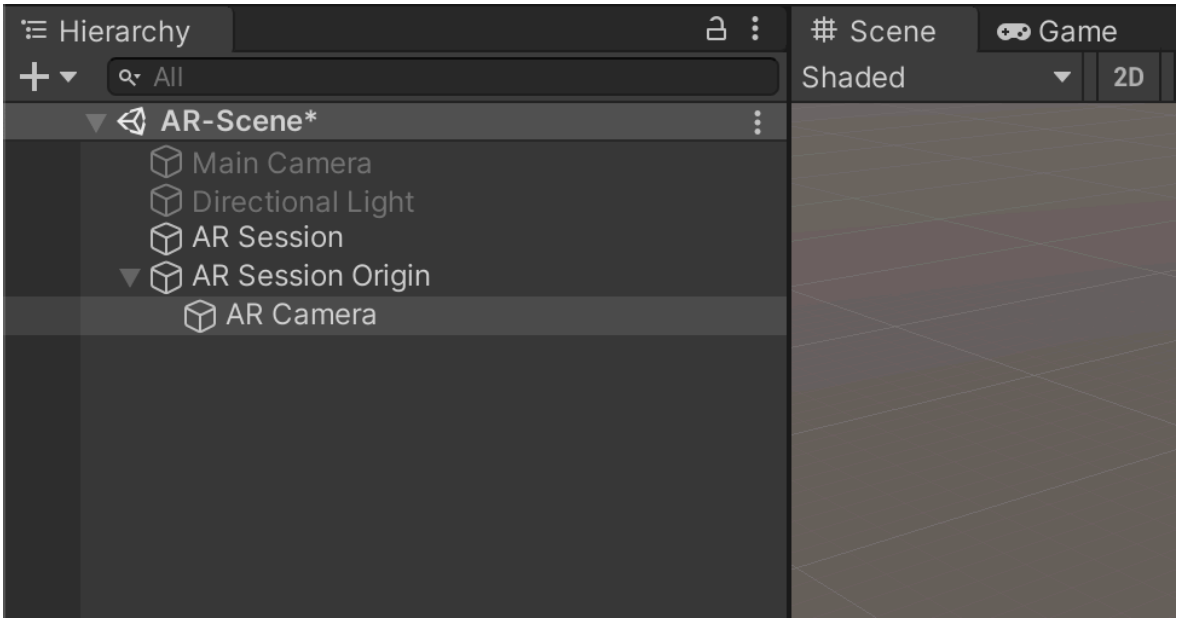
Projektissa käytetään Unityn 2019.4 versiota, joka kuuluu kirjoitushetkellä pitkäaikaisen tuen piiriin (Long-Term Support, LTS). Tämä valinnan syynä on lähinnä projektin aikaisemmin tehty työ, joten päivittäminen uudempaan Unityn versioon ei tässä tilanteessa ole tarpeellista.

Projektissa on kaksi näkymää, historiaosuuden näkymä ja AR-osuuden näkymä, jotka ovat toisistaan riippumattomia. Päivitetyn todellisuuden osuus tehdään omassa näkymässään, jolloin alkuperäisen näkymän muokkaaminen ei ole tarpeellista. Päivitetyn todellisuuden näkymässä käytetään AR-Foundationin 3.1.6 versiota. Androidia varten käytössä on AR Core XR Pluginin versio 3.1.8 ja iOS-käyttöjärjestelmää varten ARKit XR Pluginin 3.1.8 versio. Koska käytössä on vanhempi AR Foundationin versio, ainut asia mitä tarvitsee tehdä, on asentaa tarvittavat liitännäiset Unityn Package Managerista (Kuva 5, sivulla 10). Lisäksi projektille ei tarvitse määrittää projektin asetuksista Provider-Pluginia (Kuva 6, sivulla 11).

6.2 AR Näkymän tekeminen

Päivitettyä todellisuutta käytettäessä näkymään täytyy lisätä kolme peliobjektia. AR Session, AR Session Origin ja AR Camera. Kaksi ensimmäistä peliobjektia ovat itsenäisiä peliobjekteja, mutta kamera täytyy kuulua Session Origin-objektiin kuuluakseen päivitetyn todellisuuden sessioon (Kuva 10). Nämä kolme peliobjektia yhdessä olisivat kaikki tarvittava, kun sovellusta käännettäisiin mobiilisovellukseksi. Sovellukseen tarvitsee seuraavaksi lisätä siinä käytettävät ominaisuudet.

AR Session-peliobjekti sisältää kaksi valmista skriptikomponenttia, joihin ei tässä projektissa ole tarve koskea. Tärkeämpi objekti on kuitenkin AR Session Origin-objekti, jonka sisältämä komponentti käsittelee todelliset päivitetyn todellisuuden toiminnallisuudet. Tässä projektissa tarvitaan ainakin kolmea AR Foundationin ominaisuutta: tasontunnistusta, kuvantunnistusta ja kappaleiden asettamista.

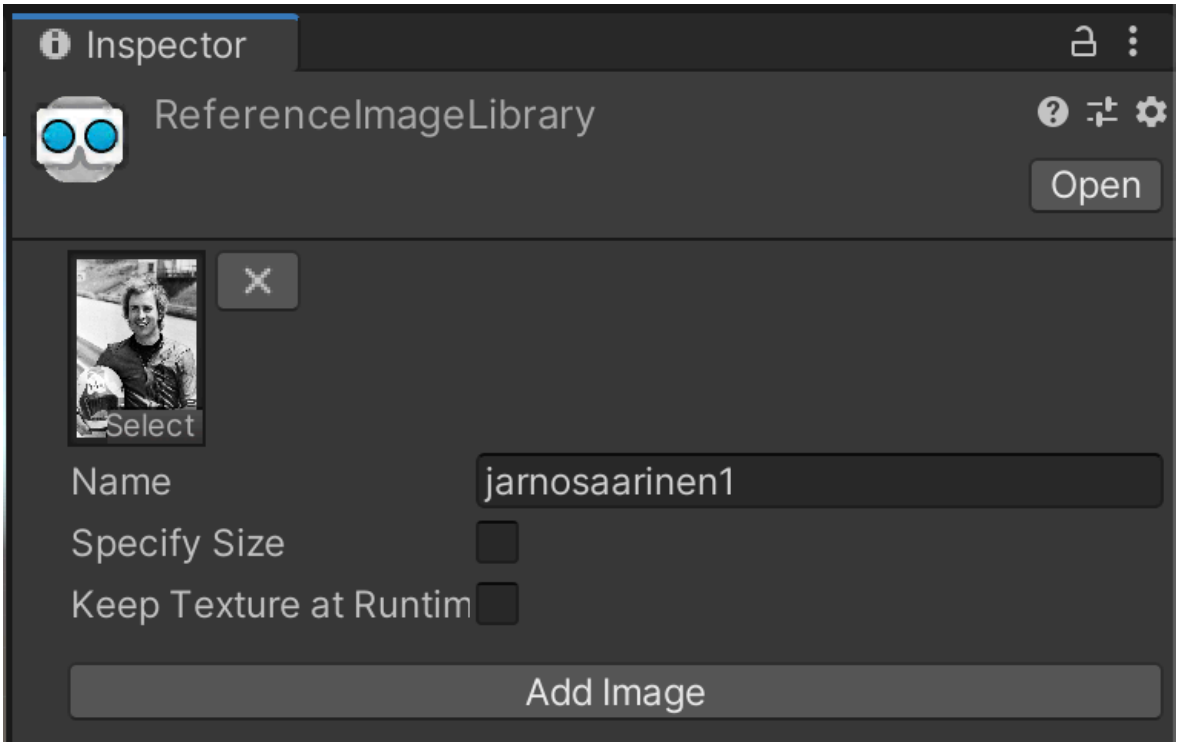


Kuva 10. AR-toiminnallisuuteen tarvittavat objektit ja niiden hierarkia näkymässä

Projektin näkymä on tässä vaiheessa vasta päivitetyn todellisuuden ympäristö. Projekti olisi mahdollista kääntää valmiiksi puhelinsovellukseksi ja asentaa puhelimeen, jolloin on mahdollista kokeilla, mikäli AR Session Origin-objektin kamera näyttää puhelimella kuvattua ympäristöä. Ilman kyseisten objektien olemassaoloa ei kuitenkaan olisi mahdollista tehdä sovellukselle määritettyjä toiminnallisuuksia.

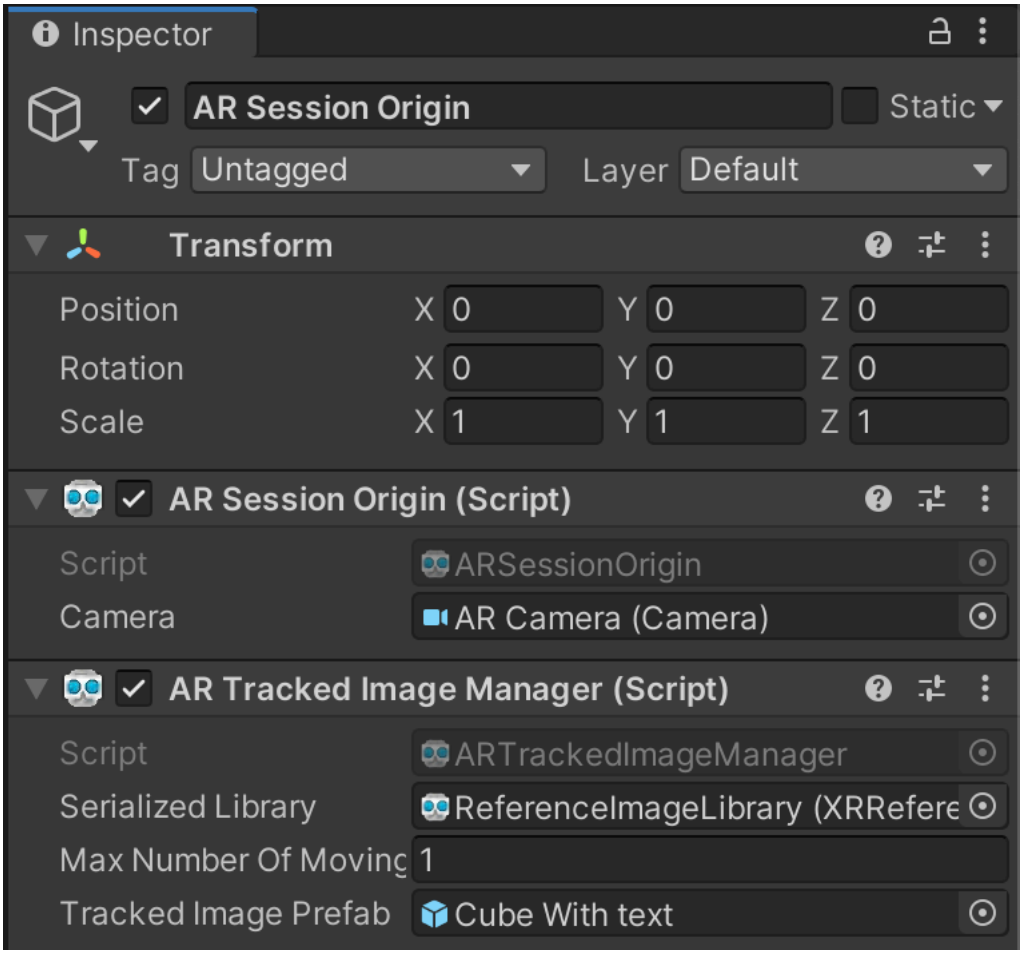
6.3 Kuvantunnistus

Sovelluksen on tarkoitus tunnistaa todellisesta maailmasta löytyviä kuvia, ja sitä kautta tuoda sisältöä näkyviin päivitetyssä todellisuudessa. Kuvan tunnistukseen on olemassa valmis AR Tracked Image Manager-skripti, mutta on myös mahdollista tehdä oma skripti tarpeen mukaan (Kuva 12). Lisäksi kuvan tunnistus tarvitsee Reference Image Library-assetin, joka sisältää kuvantunnistuksessa käytettävät kuvat (Kuva 11).



Kuva 11. Referenssikirjasto ja kuviin kohdistuvat asetukset

Tunnistavissa kuvissa ei sinänsä ole mitään erityisiä rajoituksia, mutta referenssikirjastoon lisättyjen kuvien olisi hyvä olla 300 pikseliä tai suurempi sekä korkeudeltaan, että leveydeltään. Kuvantunnistus onnistuu varmemmin, jos referenssikuva on korkealaatuinen ilman, että kuvaa on pakattu aggressiivisesti.



Kuva 12. Tracked Image Manager-komponentti

Käytettäessä päivitetyn todellisuuden ominaisuuksia, kuvan tullessa vastaan näkyisi referenssikirjaston kuville määritetty PreFab-peliobjekti. Unityn kuvantunnistuksen rajoitteena on se, että yhtä referenssikirjastoa kohti voidaan näyttää vain yhtä peliobjektia. Peliobjektin tai tunnistettavien kuvien vaihtaminen edellyttää skriptiä, joka määrittää kuvantunnistuksen asetuksia uusiksi. Lisäksi referenssikirjasto ei ole muutettavissa ajon aikana, jos se ei ole skriptin avulla perustettu Mutable Reference Library.

Tämän projektin tapauksessa voidaan ylläpitää yhtä referenssikirjastoa, koska näyttelystä tunnistettava kuvamateriaali ei muutu nopeasti ja koska tunnistettavat kuvat ladataan vasta sovelluksen käynnistyessä. Näytettävä sisältö tunnistuksen avulla on videomateriaalia, jonka näyttämiseen referenssikirjastosta saadaan tarpeeksi tietoa siihen, mitä videota näytetään tunnistetun kuvan kohdalla.

6.3.1 Referenssikirjaston perustaminen ja sisällön hakeminen

Yhdellä referenssikirjastolla voidaan seurata vain sille määrättyjä kuvia. Lisäksi on mahdollista käyttää vain yhtä peliobjektia kerrallaan AR Foundationin valmiilla

kuvantunnistuksen komponentilla. Käytettävää peliobjektia ei voida myöskään alustaa tarvittavalla tiedolla oikean sisällön näyttämiseksi.

Tästä syystä ratkaisuna on skripti, jonka tehtävänä on hallinnoida sitä, mitä kappaletta näytetään, tai millä informaatiolla kappaletta näytetään. Käytettävä peliobjekti on yksikertainen tasoelementti, jossa on pelimoottorin valmis Video Player-komponentti. Tähän komponenttiin ladataan tarvittava video näytettäväksi. Ennen sisällön näyttämistä on kuitenkin tarpeellista tietää, mitä sisältöä tarvitsee näyttää jokaisen tunnistettavan kuvan kohdalla.

Skriptin tehtävänä on hakea tarvittava informaatio, jolla luodaan referenssikirjaston sisältö ja saadaan tarvittava tieto sille, mitä videosisältöä näytetään kunkin tunnistetun kuvan kohdalla. Tarvittavat komponentit luodaan skriptin avulla, joten AR Foundationin valmista AR Tracked Image Manager-komponenttia ei käytetä. Skriptissä perustetaan ajonaikainen AR Tracked Image-komponentti ja siinä käytetään jo olemassa olevaa referenssikirjastoa, josta luodaan ajonaikainen, muokattava referenssikirjasto (Kuva 13). Kirjoitushetkellä Unityn AR Foundation ei mahdollistanut ajonaikaisen referenssikirjaston luomista, ellei referenssikirjasto sisältänyt vähintään yhtä kuvaa.

```
19      void Start()
20      {
21          trackImageManager = gameObject.AddComponent<ARTrackedImageManager>();
22          trackImageManager.referenceLibrary = trackImageManager.CreateRuntimeLibrary(runtimeImageLibrary);
23          trackImageManager.maxNumberOfMovingImages = 3;
24          trackImageManager.trackedImagesChanged += OnImageChanged;
25          trackImageManager.enabled = true;
26
27          StartCoroutine(getImageLibraryData());
28      }
```

Kuva 13. Kuvantunnistuskomponentin perustaminen ajonaikaisella referenssikirjastolla

Määritysten jälkeen kuvantunnistus saa käyttöönsä referenssikirjaston, jonne on mahdollista lisätä lisää tunnistettavaa kuvamateriaalia sovelluksen käytön aikana. Tarkoituksena on hakea palvelimelta tieto, joka sisältää listan tunnistettavista kuvista. Skripti tekee käytössä olevalle palvelimelle HTTP-pyynnön, jolla haetaan tarvittava tieto (Kuva 14). Palvelimelta vastaanotetaan JSON-objekti, mikä sisältää taulukon objekteista, jotka käsittävät linkin kuvaan ja sitä vastaavaan videoon (Kuva 15).

```
78      public IEnumerator getImageLibraryData() {
79          UnityWebRequest www = UnityWebRequest.Get("https://mpmuseobackend.herokuapp.com/api/v1/referenceLibrary");
80          yield return www.SendWebRequest();
81
82          if(www.isNetworkError || www.isHttpError) {
83              Debug.Log(www.error);
84          }
85          else {
86
87              ImageData imageData = JsonUtility.FromJson<ImageData>(www.downloadHandler.text);
88              //Stores images to gameobject for further use and referencing
89              ImageSingleton.Instance.imageData = imageData;
90              foreach(ImageInformation imageInformation in ImageSingleton.Instance.imageData.imageInformation) {
91                  Debug.Log(imageInformation.videoUrl);
92                  int imageIndex = imageInformation.id;
93                  StartCoroutine(getImage(imageInformation.imageUrl, imageInformation.id.ToString()));
94              }
95          }
96      }
```

Kuva 14. Referenssikirjaston sisällön hakeminen

```
1   {
2       "imageInformation": [
3           {
4               "id": 0,
5               "imageUrl": "https://mpmuseobackend.herokuapp.com/file/image/0",
6               "videoUrl": "https://mpmuseobackend.herokuapp.com/file/video/0"
7           },
8           {
9               "id": 1,
10              "imageUrl": "https://mpmuseobackend.herokuapp.com/file/image/1",
11              "videoUrl": "https://mpmuseobackend.herokuapp.com/file/video/1"
12          }
13      ]
14  }
```

Kuva 15. Esimerkkisisältö kuvakirjaston tiedoista

Palvelimelta haettu sisältö tallennetaan tyhjään peliobjektiin, joka jatkoa varten ylläpitää tietoa siitä, mikä video ladata kunkin tunnistetun kuvan kohdalla. Sovelluksen ei tarvitse huolehtia siitä, onko jokainen kuva ja video pelimoottorin tukemassa muodossa. Näitä ehtoja voidaan hallinnoidan palvelimen puolella ja siten estetään tilanne, jossa sisällönsyötössä lisättäisiin kuva tai video, jota pelimoottori ei tukisi. Jokainen kuva on kuitenkin lisättävä kuvakirjastoon sovelluksen ajon aikana kuvantunnistusta varten. Tätä varten iteroidaan haetun taulukon sisältö ja ladataan jokainen kuva referenssikirjastoon. Skriptiä jatketaan uudella funktiolla, jonne viedään jokainen kuvalinkki haettavaksi. Tässä funktiossa voidaan käyttää hyödyksi UnityWebRequest-luokan GetTexture-metodia. Tämän metodin avulla HTTP-pyynnöstä palautunut kuva muutetaan suoraan pelimoottorin ymmärtämäksi tekstuuriksi (Kuva 16).

```
46      public IEnumerator getImage(string url, string itemName) {
47          UnityWebRequest www = UnityWebRequestTexture.GetTexture(url);
48          yield return www.SendWebRequest();
49
50          if(www.isNetworkError || www.isHttpError) {
51              Debug.Log(www.error);
52          }
53          else {
54              Texture2D referenceImage = ((DownloadHandlerTexture)www.downloadHandler).texture;
55              StartCoroutine(addImageToLibrary(referenceImage, itemName));
56          }
57      }
```

Kuva 16. UnityWebRequest-luokan GetTexture-metodin käyttö

GetTexture-metodin avulla skripti saa tekstuurit jokaisesta kuvasta, jotka lisätään varsinaiseen referenssikirjastoon. Se myös perustaa tekstuureista referenssikuvat ja lisää ne referenssikirjastoon. Koska referenssikirjasto on perustettu aikaisemmin ajonaikaisena kirjastona, kuvien lisääminen referenssikirjastoon on mahdollista.

6.3.2 Kuvantunnistus ja peliobjektin lisääminen skriptissä

Skriptin on hallittava tässä ratkaisussa kappaleen lisäämistä kuvatunnistuksen yhteydessä. Jokaisen tunnistetun kuvan yhteydessä ladataan eri video eri URL-osoitteesta. Tracked Image Manager-komponentti sisältää ”TrackedImageChanged” tapahtuman, johon on mahdollista lisätä funktio käsittelemään kuvantunnistukseen liittyviä tapahtumia (Kuva 12). Tapahtuman kautta saadaan tieto tunnistetusta kuvasta ja sen tilasta. Kappaleen lisäämisen kannalta tärkein tieto on, kun kuva on tunnistettu ensimmäisen kerran. Asetettujen peliobjektien sijainteja ei tarvitse päivittää tai poistaa ajon aikana, joten näitä tapahtumia ei tarvitse käsitellä (Kuva 17).

```
63      public void OnImageChanged(ARTrackedImagesChangedEventArgs args) {
64          foreach (ARTrackedImage trackedImage in args.added)
65          {
66              updateARImage(trackedImage);
67          }
68          foreach (ARTrackedImage trackedImage in args.updated)
69          {
70              Debug.Log($"Name of the updated object: {trackedImage.referenceImage.name}");
71          }
72          foreach (ARTrackedImage trackedImage in args.removed)
73          {
74              Debug.Log($"Name of the removed object: {trackedImage.referenceImage.name}");
75          }
76      }
```

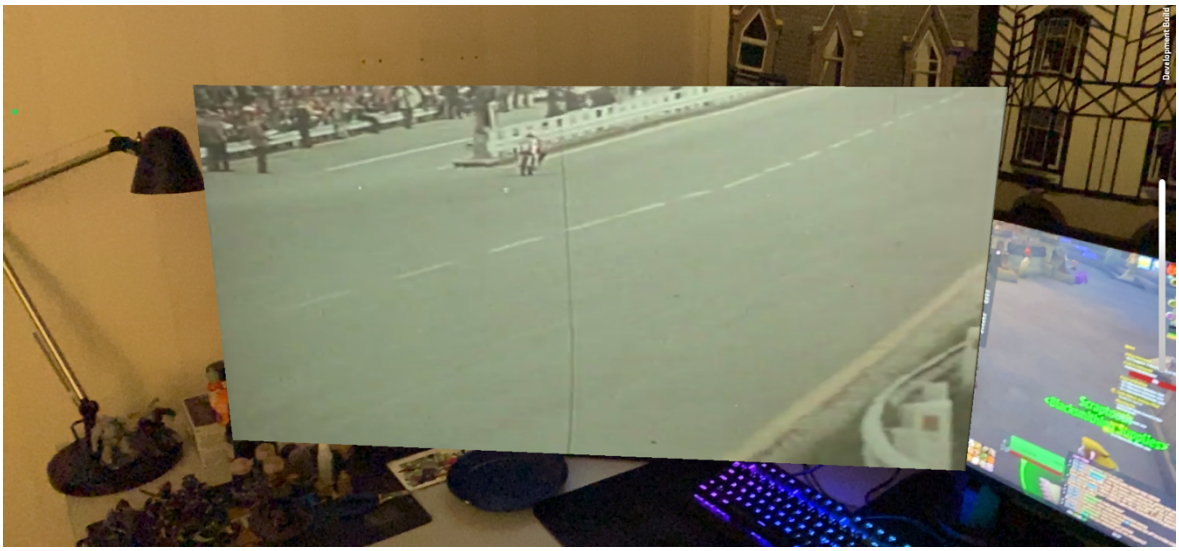
Kuva 17. Kuvantunnistuksen tapahtumankäsittelijä

Mikäli kuva tunnistetaan, kutsutaan varsinaista funktiota, jonka tehtävänä on alustaa peliobjekti näytettäväksi videon kanssa. Kuvan nimenä käytetty tunniste (id) toimii myös tallennetun taulukon indeksinä, josta on mahdollista löytää kyseiselle kuvalle tarkoitettu

video. Tunnistetusta kuvasta on saatavilla kuvan sijainti ja orientaatio pelimoottorin ymmärtämästä ympäristöstä, joita voidaan käyttää suoraan hyödyksi peliobjektia asettaessa. Peliobjekti asettuu tunnistetun kuvan keskelle ja välittömästi alustamisen jälkeen toistaa peliobjektille määritettyä videota (Kuva 18). Tunnistettu kuva on kappaleelle sen ”lattiataso”, joka on hyödyllistä muistaa peliobjektia luodessa. Sovelluksen käytön aikana, mikäli sovellus tunnistaa referenssikirjastoon lisätyn kuvan, kuvan eteen ilmestyy sille määritetty video (Kuva 19).

```
98      private void updateARImage(ARTrackedImage trackedImage) {
99          XRReferenceImage refImage = trackedImage.referenceImage;
100         int refImageName = Convert.ToInt32(refImage.name);
101         Debug.Log(refImageName);
102         GameObject videoPlane = targetPrefab;
103         var videoPlayer = videoPlane.transform.GetChild(0).GetComponent<UnityEngine.Video.VideoPlayer>();
104         videoPlayer.url = ImageSingleton.Instance.imageData.imageInformation[refImageName].videoUrl;
105         Instantiate(videoPlane, trackedImage.transform.position, trackedImage.transform.rotation);
106     }
107 }
```

Kuva 18. Peliobjektin lisääminen tunnistetulle kuvalle



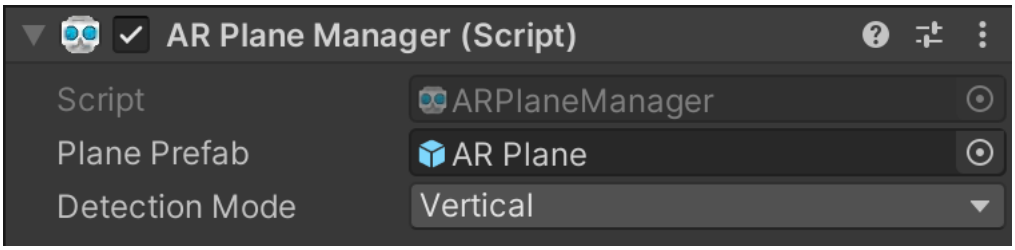
Kuva 19. Tunnistetun kuvan edessä oleva video

6.4 Tasontunnistus ja käyttö

Tasontunnistusta tarvitaan kappaleiden näyttämisessä vaakasuorissa tasoissa (eli lattioissa). Tämä toiminta on käyttäjälle loppujen lopuksi varsin näkymätön, mutta tärkeä, jotta sovelluksen käyttäjä voi tutkia sisältöä ilman, että näytettävät kappaleet leijuisivat ilmassa, tai että tunnistetuille tasoille laitettavat kappaleet menisivät seinistä läpi.

AR Foundation sisältää valmiin AR Plane Manager-komponentin, joka käsittelee tasontunnistusta (Kuva 20). Tämä komponentti täytyy olla muidenkin AR-toiminnallisuuksien tavoin osa AR Session Origin-peliobjektia, joka käsittelee näkymän päivitetyn todellisuuden

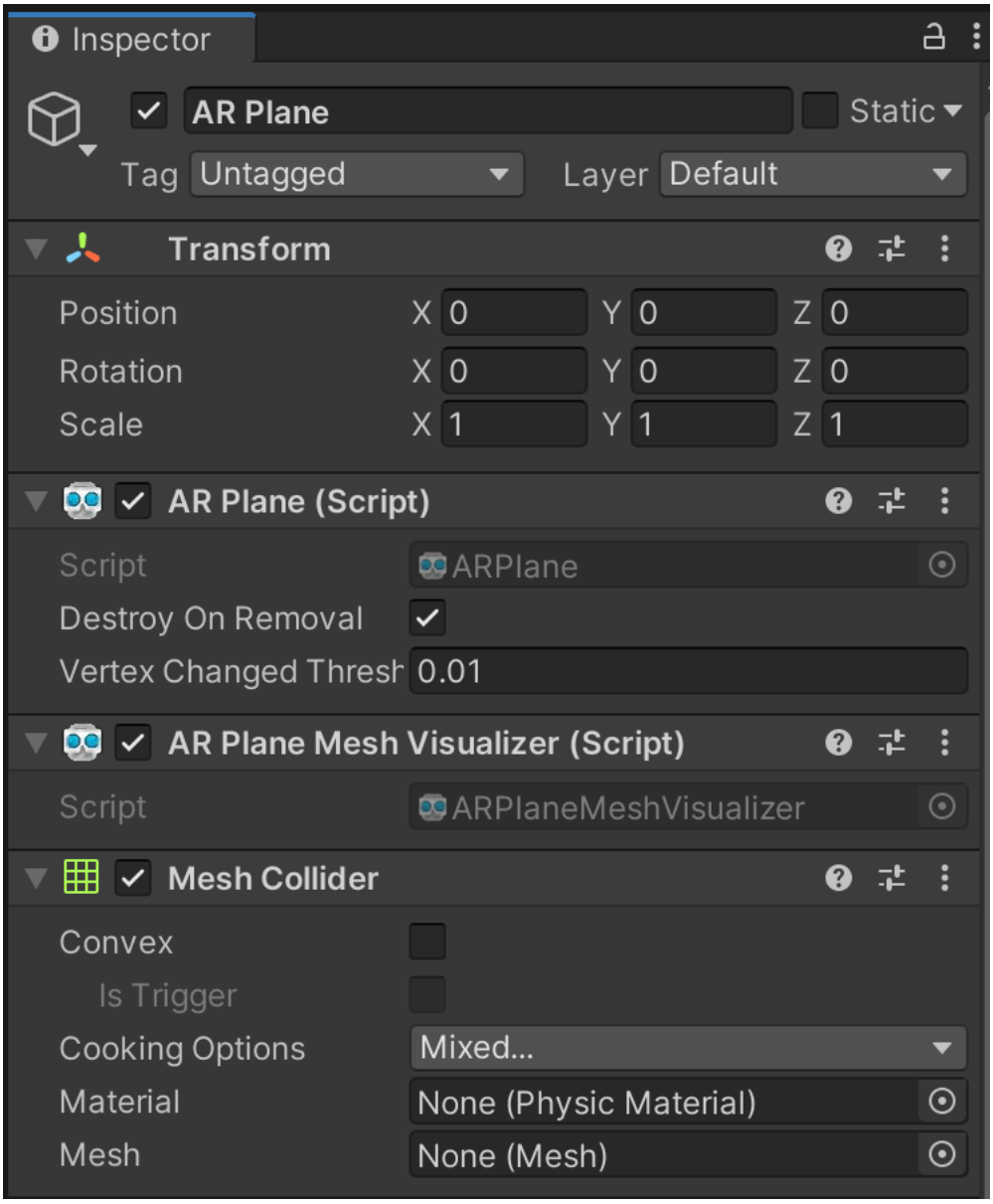
sessiota. Tämä komponentti ei tarvitse muuta kuin määrittelyn sille, mitä PreFab-peliobjektia käytetään tunnistetun tason havainnollistamiseen. Ja lopuksi komponentille määritetään, tunnistaako se vaaka- tai pystysuoria tasoja vai molempia.



Kuva 20. AR Plane Manager-komponentti

6.4.1 Tunnistetun tason havainnollistaminen

AR Plane Manager-komponentti tunnistaa tasoja, mutta tasoja itsessään ei voi käyttää hyödyksi AR-sovelluksen hahmottelemassa kolmiulotteisessa ympäristössä, ellei tunnistetuille tasoille tehdä ”tasoja” pelimoottorin ymmärtämässä ympäristössä. Toisin sanoen tasona toimivaan peliobjektiin on lisättävä Mesh Collider-komponentti, jonka tarkoituksena on toimia ”kiinteänä” fyysisenä tasona kolmiulotteisille kappaleille (Kuva 20). Tämä on erityisen tärkeää, jotta kappaleet pysyvät tunnistetuilla tasoilla ja eivätkä pakene seinien tai lattian läpi. Tämä on vielä tärkeämpää kappaleille, jotka sisältävät fysiikanmallinnusta. Tässä tilanteessa Mesh Collider-tason tehtävänä on kuitenkin vain toimia selkeänä lattianrajana asetettaville kappaleille pelimoottorin ”ymmärtämässä” 3D-ympäristössä.



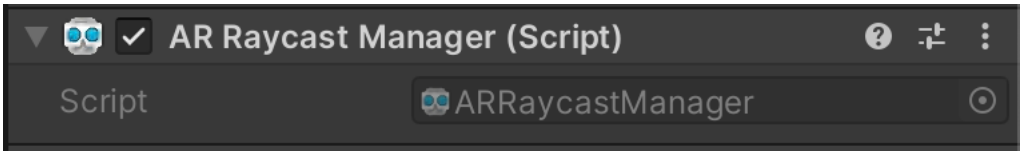
Kuva 21. Taso-objektin komponentit tasontunnistusta varten

Taso on tyhjä peliobjekti, johon lisätään valmiit AR Plane ja AR Plane Mesh Visualizer-skriptit komponenteiksi (Kuva 21). Nämä komponentit ovat AR Foundationin ominaisuuksia, jotka käsittelevät tasonluontia pelimoottorin saamista tiedoista ympäröiviin tasoihin liittyen.

Havainnollistamista varten on mahdollista hyödyntää erilaisia Mesh ja/tai Line Renderer-komponentteja ja niihin määritettyjä tekstuureilla. Tunnistettuja tasoa havainnollistetaan tässä ratkaisussa Line Renderer-komponentilla, joka piirtää rajat tunnistetuille vaakasuorille tasoille.

6.4.2 Kappaleiden asettaminen tasoille

Sovelluksen tarkoituksena on asettaa vaakasuorille tasoille kolmiulotteinen kappale tutkittavaksi. Vaakasuorille tasoille laittaminen edellyttää osumatarkistusta, jonka takia AR Session Origin-peliobjektiin on lisättävä valmis AR Raycast Manager-komponentti, jonka tehtävänä on tunnistaa erilaisia seurattavia kohteita AR-ympäristössä. Komponentin tehtävä on tunnistaa määritettyjä tasoja. Komponentti ei itsessään paljasta mitään toiminnallisuutta, mutta on tarpeellinen skripteja varten (Kuva 22).



Kuva 22. AR Raycast Manager-komponentti

Skriptin tarkoituksena on reagoida, mikäli sovelluksen käyttäjä koskee ruutuun. Skriptin tehtävä on saada kosketuksen koordinaatit ja suorittaa osumatarkistus ympäristössä olevia kappaleita vastaan; tässä tapauksessa tunnistettuja tasoja vastaan. AR Foundationin osumatarkistus ottaa huomioon kaikki muuttujat, kuten tason ja puhelimen välisen kulman tehdessään osumatarkistusta.

```
37      bool getTouchPos(out Vector2 touchPos) {
38          if(Input.touchCount > 0) {
39              touchPos = Input.GetTouch(0).position;
40              return true;
41          }
42          touchPos = default;
43          return false;
44      }
```

Kuva 23. Funktio kosketuksen sijainnin hakemiselle

Kosketuksen sijaintia tarkistetaan jokaisen ruudunpäivityksen yhteydessä (Kuva 23). Mikäli ruudunpäivityksen aikana on syntynyt kosketus, tätä kosketuksen sijaintia vasten suoritetaan osumatarkistus. Osumatarkistuksen täytyy osua tässä tapauksessa tasoon, jonka sovellus on tunnistanut jo aiemmin. Mikäli osumatarkistus tuottaa osuman, skripti luo uuden peliobjektin PreFab objektin avulla paikkaan, missä osuma tapahtui (Kuva 24).

```
20      // Update is called once per frame
21      void Update()
22      {
23          if(!getTouchPos(out Vector2 touchPos))
24              return;
25          if(_arRaycastMngr.Raycast(touchPos, hits, TrackableType.PlaneWithinPolygon)) {
26              var hitPose = hits[0].pose;
27              if(spawnObject == null) {
28                  spawnObject = Instantiate(gameObjectToInstantiate, hitPose.position, hitPose.rotation);
29              }
30
31              else {
32                  spawnObject.transform.position = hitPose.position;
33                  spawnObject.transform.rotation = hitPose.rotation;
34              }
35          }
36      }
```

Kuva 24. Ruudunpäivityksen aikana suoritettava osumatarkistus

Nyt ruutua koskettaessa, mikäli osumatarkistus osuu tunnistettuun tasoon, kappale lisätään tunnistetun tason päälle. Kappale ilmestyy siihen kohtaan tasoa, missä osumatarkistuksen osuma tapahtui. Kappale asettuu aina suhteessa tasoon nähden, eli pystysuora taso olisi asetettavalle kappaleelle sen ”lattia”. Tämä on hyvä ottaa huomioon seiniin asetettavien kappaleiden kanssa ja niiden oletusarvoisessa orientaatiossa pelimoottorin editorissa. Vaakasuoran tason kanssa tätä ei tarvitse huomioida. Jotta kappale saadaan osoittamaan tutkimisen kannalta parhaimpaan mahdollisimpaan suuntaan, kappaleen kääntämiseen luodaan toiminto, jolla voidaan määrittää kappaleen orientaatio asettamisen jälkeen.

6.4.3 Kappaleen valinta ja orientaation kontrollit

AR Session Origin-objekti tarvitsee uuden skriptikomponentin kappaleen valintaa ja orientaation määrittämistä varten. Skriptin tehtävä on hallita kappaleen valitsemista ja sen orientaation määrittämistä. Lisäksi käytettävät PreFab-peliobjektit tarvitsevat skriptin, jonka tehtävänä on hallita peliobjektin valintaa. Skripti lisätään jokaiseen peliobjektiin, jonka valinnan tilaa tarvitsee käsitellä sovelluksessa (Kuva 25). Varsinaisen skriptin tarkoituksena on toteuttaa kappaleen valinta ja sen kääntäminen tasolla.

Kuva 25. Asetettavien kappaleiden valintaskripti

Osumatarkistusta tarvitaan kosketuksen sijainnin tunnistamiseen ja mikäli kosketus kohdistuu kappaleeseen, kappaleesta tulee aktiivinen. Tämä osumatarkistus ei tarvitse AR Foundationin osumatarkistusta vaan pelimoottorin omaa 3D-ympäristöihin tarkoitettua osumatarkistusta. AR Foundationin osumatarkistus toimii ainoastaan AR-toiminnallisuuksia varten, kuten tasontunnistuksessa. Skripti kuitenkin tarvitsee päivitetyn todellisuuden kameran, jotta osumatarkistus tehdään puhelimen sijaintia vasten. Ratkaisuna on funktio, joka tarkistaa ruudunpäivityksen yhteydessä kosketusta ja suorittaa osumatarkistuksen kosketukseen sijaintiin. Mikäli osumatarkistus osuu peliobjektiin, joka sisältää valintakomponentin, osutusta kappaleesta tulee ”aktiivinen” (Kuva 26).

```
33    void Update()
34    {
35        if (Input.touchCount > 0)
36        {
37            print(Input.touchCount);
38            Touch touch = Input.GetTouch(0);
39            touchPosition = touch.position;
40
41            if (touch.phase == TouchPhase.Began)
42            {
43                Ray ray = arCamera.ScreenPointToRay(touch.position);
44                RaycastHit hitObject;
45                if (Physics.Raycast(ray, out hitObject))
46                {
47                    PlacementObject placementObject = hitObject.transform.GetComponent<PlacementObject>();
48                    if (placementObject != null)
49                    {
50                        ChangeSelectedObject(placementObject);
51                    }
52                }
53            }
54        }
55    }
56
```

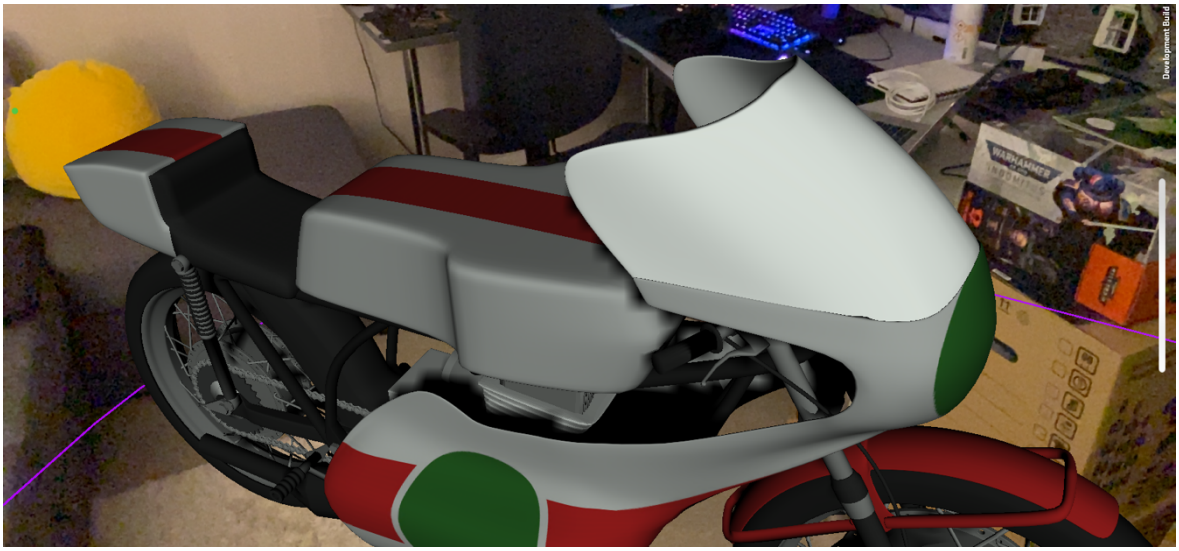
Kuva 26. Kosketuksesta tehtävä osumatarkistus

Yksinkertainen tarkistus ruudunpäivityksen yhteydessä kääntää kappaletta, jos se on valittu käyttäjän kosketuksella (Kuva 27). Sovelluksen käytön aikana kappale voidaan valita kosketuksen avulla. Tämä kappale pyörii akselinsa ympäri tasaiseen tahtiin ollessaan valittuna. Kun kappaletta kosketetaan uudestaan, kappaleen pyörittäminen loppuu.

Kappaletta voidaan myös siirtää kosketuksella. Kappaleen siirtäminen tapahtuu kosketta-malla mitä tahansa vapaata aluetta tunnistetuilla tasoilla (Kuva 28).

```
60          if(objectSelected) {
61              placedPrefab.transform.Rotate(transform.up, rotationSpeed * Time.deltaTime);
62          }
63      }
```

Kuva 27. Kappaleen kääntäminen



Kuva 28. Peliobjekti tunnistetun tason päällä

7 YHTEENVETO

Unity-pelimoottori ja sen AR Foundation-kirjasto ovat mielenkiintoisia työkaluja toteuttaa käytännön sovelluksia. Tällä kertaa moottoripyörämuseon Jarno Saarisen näyttelyyn saadaan käyttöön sovellus, jonka sisältö on mahdollista muokata näyttelyn tarpeisiin ja tuoda haluttua sisältöä näytettäväksi ja vietävissä käyttöön tulevaa näyttelyä varten.

Projektia aloittaessa oli epävarmaa, millaisia rajoitteita kuvantunnistuksessa ja sen avulla näytettävien kappaleiden kanssa tulee vastaan, kun halutaan hyödyntää sisältöä, joka olisi määritettävissä sovelluksen ulkopuolella. Mikäli AR Foundationin toiminnassa olisi ollut merkittäviä rajoitteita, se olisi vaikeuttanut sovelluksen kehittämistä.

Sovelluksen kehittämisessä saavutettiin kuitenkin paljon. Ensimmäinen merkittävä saavutus on päivitetyn todellisuuden sovelluksen tekeminen. AR Foundation-kirjasto on varsin uusi työkalu Unity-pelimoottorissa, joten siihen liittyvä osaaminen on vielä vähäistä, jolloin AR-Foundationin opettelu oli paljon uuden tutkimista ja selvittämistä siitä, mitkä ovat käyttömahdollisuudet sovellusta varten. Toinen merkittävä saavutus oli sisällön hakeminen palvelimelta sovelluksen käytettäväksi päivitetyn todellisuuden osuudessa. HTTP-pyyntöjen suorittaminen mahdollistaa sovelluksen mukautuvuuden jatkoa ajatellen.

Koska sisältö on saatavissa verkkosovelluksille suunnatulla palvelimella, sovelluksen käyttämää sisältöä on yhtä lailla helppo muokata käyttötarpeen selvitessä. Jatkoa ajatellen, sovelluksen kehittämistä olisi mahdollista jatkaa laajentamalla kuvantunnistuksen ominaisuuksia ja lisätä mahdollisuus useamman erilaisen peliobjekti näyttämiseen. Kappaleiden kääntäminen on myös puutteellista ja eikä tällä hetkellä ole mahdollista kääntää asetettuja kappaleita sormen liikkeen mukaan. Tämä parantaisi sovelluksen käytettävyyttä.

8 LÄHTEET

Abramovich, G. 2020. 5 Innovative Examples Of Augmented Reality In Action. [viitattu 5.10.2020]. Saatavissa: https://www.adobe.com/insights/5-realworld-examples-of-augmented-reality-innovation.html

Achao Design. 15.12.2017. What Are All These Realities? VR, MR, AR and XR 101 [viitattu 9.10.2020] Saatavissa: https://www.achao.design/inspire/what-are-all-these-realities-vr-mr-ar-xr-101

Apple. 2020. ARKit [viitattu 22.10.2020] Saatavissa: https://developer.apple.com/documentation/arkit

Craig, A. B. 2013. Understanding augmented reality: Concepts and applications. Amsterdam: Morgan Kaufmann. [viitattu 15.10.2020] Saatavissa: https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/b5ag28/alma991948549006254

Gleb. B. 4.1.2020 VR vs AR vs MR: Differences and Real-Life Applications. [viitattu 9.10.2020] Saatavissa: https://rubygarage.org/blog/difference-between-ar-vr-mr

Google. 2020. ARCore Overview [viitattu 22.10.2020] Saatavissa: https://developers.google.com/ar/discover

Google. 2020. Search what you see [viitattu 5.10.2020] https://lens.google.com/

Ikea. 2020. Say hej to IKEA Place [viitattu 5.10.2020] https://www.ikea.com/au/en/customer-service/mobile-apps/say-hej-to-ikea-place-pub1f8af050

Marr, B. 2020. The Important Difference Between Augmented Reality And Mixed Reality [viitattu 15.10.2020] Saatavissa: The Important Difference Between Augmented Reality And Mixed Reality

Microsoft. 26.8.2020. What is Mixed Reality? [viitattu 13.10.2020] Saatavissa: https://docs.microsoft.com/en-us/windows/mixed-reality/discover/mixed-reality

Moottoripyörämuseo. 2020. Museomme Historiaa [viitattu 16.12.2020] Saatavissa: https://www.moottoripyoramuseo.fi/historia/

Noble. S. 8.2.2019. The ultimate VR, AR, MR guide [viitattu 15.10.2020] Saatavissa: https://www.aniwaa.com/guide/vr-ar/ultimate-vr-ar-mr-guide/ - What_is_Augmented_Reality_AR

North of 41. 20.3.2018. What really is the difference between AR / MR / VR / XR? [viitaattu 8.10.2020]. Saatavissa: https://medium.com/@northof41/what-really-is-the-difference-between-ar-mr-vr-xr-35bed1da1a4e

Ranieri, M. 1.10.2020. A new sense of direction with Live View [viitattu 10.11.2020] Saatavissa: https://blog.google/products/maps/new-sense-direction-live-view/

Unity Technologies. 2020. About AR Foundation [viitattu: 20.10.2020] Saatavissa: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/index.html

Unity Technologies. 2020. GameObjects [viitattu 27.10.2020] Saatavissa: https://docs.unity3d.com/Manual/GameObjects.html

Unity Technologies. 2020. Important Classes – MonoBehaviour [viitattu 10.1.2021] Saatavissa: https://docs.unity3d.com/Manual/class-MonoBehaviour.html

Unity Technologies. 2020. Scripting Concepts [viitattu 28.10.2020] Saatavissa: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

Unity Technologies. 2020. Trackables Manager [viitattu 24.10.2020] Saatavissa: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.1/manual/trackable-managers.html

Unity Technologies. 2020. UnityWebRequest [viitattu 23.11.2020] Saatavissa: https://docs.unity3d.com/Manual/UnityWebRequest.html

Unity Technologies. 2020. XR Plug-in Framework [viitattu 20.10.2020] Saatavissa: https://docs.unity3d.com/Manual/XRPluginArchitecture.html

Unity Techonologies. 2020. Introduction to components [viitattu 10.1.2021] Saatavissa: https://docs.unity3d.com/Manual/Components.html