LAB University of Applied Sciences

Lisättyä todellisuutta hyödyntävä näyttelysovellus

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja viestintätekniikka
2020
Santtu Lehtinen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Lehtinen, Santtu	Opinnäytetyö, AMK	2020
	Sivumäärä	
	36	

Työn nimi

Lisättyä todellisuutta hyödyntävä näyttelysovellus

Tutkinto

Insinööri (AMK)

Ohjaavan opettajan nimi, titteli ja organisaatio

Henri Koukka, lehtori, Tieto- ja viestintätekniikka

Toimeksiantajan nimi, titteli ja organisaatio

Teemu Parkkinen, toimitusjohtaja, Advison Solutions Oy

Tiivistelmä

Opinnäytetyön päätavoitteena oli luoda tilaajalle iPadilla toimiva, lisätyn todellisuuden näyttelysovellus. Sen kriteereihin kuului mahdollistaa käyttö näyttelytilasta riippumatta. Työn toissijaisena tavoitteena oli perehtyä lisätyn todellisuuden tekniikoihin.

Sovellus toteutettiin Unity-pelimoottorilla ja lisätyn todellisuuden toiminnallisuus mahdollistettiin Unityn ARFoundation-ohjelmistokehyksellä. Ohjelmointiympäristönä toimi Visual Studio for Mac ja lopullisen iPad-sovelluksen kääntämiseen käytettiin Applen Xcode-ohjelmankehitysympäristöä.

Opinnäytetyö koostuu kahdesta osasta. Ensimmäinen on teoriaosuus, jossa perehdytään lisättyyn todellisuuteen, sovelluskehityksessä käytettyihin työkaluihin sekä erilaisiin lisätyn todellisuuden ratkaisuihin. Toisessa osassa käsitellään sovelluksen kehitystä keskittyen lisätyn todellisuuden toiminnallisuuden toteutukseen.

Lopputuloksena syntyi tilasta riippumaton, lisättyä todellisuutta hyödyntävä näyttelysovellus. Sovellus luovutettiin projektista vastanneelle yritykselle ja ladattiin App Storeen valmiina tilaajan käyttöön.

Asiasanat

Lisätty todellisuus, Unity, ARFoundation, ARKit

Abstract

Author(s)	Type of Publication	Published
Lehtinen, Santtu	Thesis, UAS	2020
	Number of Pages	
	36	

Title of Publication

Utilisation of augmented reality in an exhibition application

Name of Degree

Engineer (UAS)

Name, title and organization of the supervising teacher

Henri Koukka, Senior Lecturer, Information and communication technology

Name, title and organization of the client

Teemu Parkkinen, CEO, Advison Solutions Ltd.

Abstract

The primary objective of this thesis was to create an augmented reality (AR) exhibition iPad app to a client. One criterion for the app was to make it usable regardless of the exhibition space. The secondary objective was to familiarise with the AR techniques.

The app was created using Unity game engine and the AR functionality was enabled with Unity's ARFoundation framework. Visual Studio for Mac served as an integrated development environment (IDE) in coding tasks, and the final iPad app build was done with Apple's Xcode IDE.

The thesis consists of two parts. In the first part, we look at the AR, the tools used in the development process, as well as examine different AR solutions. The second part covers the development process of the app, focusing on the implementation of the AR functionality.

As a result of this thesis, a space independent AR exhibition app was created. The app was delivered to the company in charge and uploaded to App Store for the client.

Keywords

Augmented reality, Unity, ARFoundation, ARKit

Sisällys

1 Johdanto

Opinnäytetyön aihe muotoutui erään asiakasprojektin pohjalta, jonka parissa työskentelin ollessani työharjoittelussa Advison Solutions Oy:ssä. Kyseisen projektin tavoitteena oli luoda Unity-pelimoottorilla tilaajalle iPadilla toimiva informatiivinen näyttelysovellus, joka hyödyntää lisättyä todellisuutta.

Opinnäytetyön toimeksiantaja on lahtelainen, vuonna 2019 perustettu Advison Solutions Oy, jonka päätoimialaa on liikkeenjohdon konsultointi. Sen palveluihin kuuluu IT-suunnittelu, -hankinnat sekä -strategia.

Sovelluksen kehitystyö lähti tilaajan tarpeesta saada käyttöön joustava lisätyn todellisuuden ratkaisu osaksi suurempaa näyttelykokonaisuutta, jota pystyttäisiin tarpeen tullen hyödyntämään vaivattomasti tilassa kuin tilassa. Kehitystyön ohessa aukeni oivallinen tilaisuus tutustua tarkemmin lisättyyn todellisuuteen ja analysoida erilaisia ratkaisuja, jotta saataisiin parempi ymmärrys sen tekniikoiden hyödyntämisestä tapauskohtaisesti. Tilaisuus päätettiin käyttää hyväksi ja tuottaa opinnäytetyö sen pohjalta. Tässä opinnäytetyössä perehdytään siihen mitä lisätty todellisuus on, sen erilaisiin toteutustapoihin, sekä käydään tarkemmin läpi sovelluksen kehitystä ja sen yhteydessä käytettyjä ratkaisuja käytännöstä saatujen tulosten kautta.

Opinnäytetyö koostuu kahdesta osasta. Ensimmäinen on teoriaosuus, jossa käydään läpi lisättyä todellisuutta, sen tekniikoita ja projektissa käytettyjä työkaluja. Sen jälkeen perehdytään asiakasprojektin kehitysprosessiin. Kehitysprosessin tarkastelu on rajattu sovelluksen lisätyn todellisuuden toiminnallisuuden toteutukseen. Tarkoituksena on selittää miten erilaiset ratkaisut rakentuvat havainnollistusten ja koodiesimerkkien avustuksella. Tämän lisäksi analysoidaan käytettyjen ratkaisutekniikoiden tuloksia käyttöympäristössä ja määritellään niiden käyttökelpoisuutta kokemuspohjalta. Lopuksi vielä tehdään yhteenveto, jossa pohdiskellaan projektin onnistumista, sovelluksen tuomia hyötyjä, prosessissa törmättyjä ongelmia sekä mahdollisia jatkotutkimuskohteita.

2 Lisätty todellisuus

2.1 Mitä lisätty todellisuus on

Lisätty todellisuus on muunnelma virtuaalitodellisuudesta. Virtuaalitodellisuus upottaa käyttäjänsä täysin keinotekoiseen ympäristöön, eikä käyttäjä näe todellista maailmaa ympärillään. Lisätty todellisuus eroaa tästä siten, että se lisää todelliseen ympäristöön digitaalista materiaalia, kuten kuvia, videoita, ääntä tai tuntoaistimuksia. Täten voidaan sanoa, että lisätty todellisuus täydentää todellisuutta, ei niinkään korvaa sitä. (Kipper & Rampolla 2012, 1.)

Lisätyn todellisuuden tavoitteena on rikastaa havainnointia ja tietämystä todellisesta ympäristöstä. Yleensä lisätyn todellisuuden sovelluksissa, käyttäjälle visualisoidaan digitaalista sisältöä lasien, kuulokkeiden, videoprojektorien tai mobiililaitteiden avulla. Käyttäjä voi esimerkiksi vertailla erilaisia sisustusratkaisuja, asettamalla digitaalisia huonekaluja huoneeseensa tai vaihtamalla seinien väriä. (Arnaldi, Guitton & Moreau 2018, 26.)

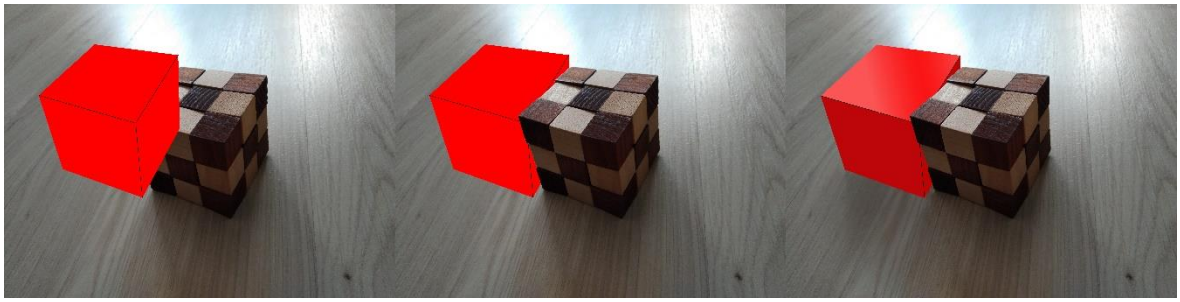
Lisätyn todellisuuden toteutuminen perustuu kahteen perusprosessiin, joiden pitää täyttyä jokaisessa vaiheessa. Ensimmäiseksi sovelluksen täytyy määrittää sekä fyysisen maailman hetkellinen tila, että virtuaalisen maailman hetkellinen tila. Toiseksi sovelluksen pitää näyttää tämä virtuaalisen maailman tila suhteessa todellisen maailman tilaan sellaisella tavalla, että käyttäjä havaitsee virtuaalimaailman elementit osana omaa ympäristöään. (Craig 2013, 39.)

2.2 Lisätyn todellisuuden haasteet

Suurimmat lisätyn todellisuuden haasteet ovat kohteen tunnistus ja sensorien tarkkuus. On tärkeää, että todellisen ympäristön ja virtuaalisen ympäristön elementit kohdistuvat oikein suhteessa toisiinsa. Jotta ympäristöjen kohdistuminen olisi mahdollista, modernit lisätyn todellisuuden järjestelmät hyödyntävät erilaisia sensoreita, jotka keräävät tietoa ympäristöstä. Näihin sensoreihin lukeutuu erityyppisiä laitekomponentteja, kuten kameroita, kiihtyvyysantureita, GPS:ää, gyroskooppeja, kompasseja ym. (Kipper & Rampolla 2012, 23 – 24.) Erilaisten sensorien tarkkuudet vaihtelevat ja saattavat yksinään aiheuttaa ongelmia kohdistuksen kanssa. Tästä syystä olisi hyvä turvautua useamman sensorin tuottamaan dataan.

Lisätyn todellisuuden uskottavuus luo myös omat ongelmansa. Jotta lisätyn todellisuuden tuotos olisi mahdollisimman uskottava, pitää virtuaalisten elementtien seurata todellisen ympäristön fysiikkaa. Esimerkiksi valaistus ja tilan geometria olisi hyvä ottaa huomioon virtuaalisia elementtejä renderöitäessä. Kuvassa 1 havainnollistetaan, kuinka tällä pyritään siihen, että valo osuisi virtuaaliseen kohteeseen oikeassa kulmassa, kohteesta syntyvä

varjo heittyisi oikein ja ettei kohde näkyisi todellisten seinämien läpi. (Arnaldi ym. 2018, 156 - 158.)



Kuva 1. Havainnollistus valaistuksen ja geometrian huomioimisesta (mukailtu Arnaldi ym. 2018, 157.)

2.3 Lisätyn todellisuuden tilanne ja hyödyt

Lisätyn todellisuuden hyödyntäminen on vielä varsin vähäistä ja alkutekijöissään. Tähän mennessä sitä on rajoittanut esimerkiksi laitteiston prosessointivoima, teknologian alkeellisuus ja toteuttamiskustannukset. Näistä syistä monet lisätyn todellisuuden toteutukset on toteutettu pseudoratkaisuilla, joilla pyritään imitoimaan todellista lisättyä todellisuutta, mutta ei päästä täydelliseen ja uskottavaan lopputulokseen.

Arnaldi ym. (2018, 303 – 304) luettelevat teknologian mahdollistamiksi hyödyiksi:

- kustannusten pienentämisen
- paremmat harjoittelumahdollisuudet turvallisessa ympäristössä
- monimutkaisen datan tarkastelun
- tarkkuuta vaativien liikkeiden harjoittelun
- luovuuden lisäämisen
- avustetun ajamisen
- teollisten ja kirurgisten toimenpiteiden suorittamisen
- virtuaalisesti laajennetut paikkavierailut.

Nykytilanne mahdollistaa jo monia edellä luetelluista hyödyistä, mutta teknologian kehittyessä ja tarpeen kasvaessa, tulee lisätyn todellisuuden toteutukset lisääntymään.

3 Taustatiedot

3.1 Projektin toimeksianto

Alkuperäinen toimeksianto, joka saatiin projektista vastanneelta yritykseltä, oli luoda kaksi iPadilla toimivaa lisättyä todellisuutta hyödyntävää näyttelysovellusta. Sovellusten sisällöt poikkeaisivat toisistaan ja niissä käytettäisiin eri ratkaisuja lisätyn todellisuuden toteuttamiseen. Kun ensimmäinen sovellus oli jo valmis ja toinen pitkällä kehityksessä, saatiin tietää, että varsinainen tilaaja oli tilannut ja halusi vain yhden sovelluksen.

Tilaajan varsinainen toimeksianto oli luoda iPadilla toimiva, lisättyä todellisuutta hyödyntävä informatiivinen sovellus osaksi suurempaa näyttelykokonaisuutta. Lähtökohtaisesti haluttiin, että näytteilleasettaja pystyisi asettelemaan virtuaaliset kohteet näyttelytilaan mieleisellään tavalla ja tarvittaessa siirtelemään niitä. Tällä pyrittäisiin siihen, ettei näyttely olisi sidoksissa näyttelytilaan, vaan että sen pystyisi tarvittaessa järjestämään missä tahansa tilassa. Mahdollisesti myös Suomen ulkopuolella. Tästä syystä sovellukseen tulisi myös mielellään jonkinlainen kielivalikko. Lisäksi kohteiden olisi tarkoitus leijua ilmassa tietyssä korkeudessa ja näytettävä tarkempaa tietoa, kun kävijä tähtää iPadin niitä kohti. Toiveena oli, että sovellusta luotaessa käytettäisiin ilmaisia kehitystyökaluja, jos se vain on suinkin mahdollista.

3.2 Työryhmä

Projektin työryhmä koostui pääasiassa projektista vastanneen yrityksen henkilöstöstä. Projektin etenemisestä vastasi yrityksen projektipäällikkö, jonka lisäksi päävastuut sovelluksen graafisesta puolesta ja 3D-malleista kuului yrityksen organisaation sisäisille tiimeille. Projektin loppuvaiheilla työryhmään lisättiin vielä yrityksen puolesta teknologiavastaava, jonka kanssa pyrittiin löytämään sovelluksen lopullinen ratkaisu.

Graafiseen tiimiin kuului pari käyttöliittymäsuunnittelijaa, joiden tehtävänä oli suunnitella sovelluksen käyttöliittymä ja luoda tarvittavat graafiset materiaalit sen toteuttamiseen. 3D-mallintamisesta vastasi pääasiassa kaksi vastaavan yrityksen mallintajaa, mutta osa mallintamisesta ulkoistettiin freelancereille.

Sovelluksen varsinainen kehitystyö ja toteutus oli ulkoistettu Advison Solutions Oy:lle. Advisonin toimitusjohtaja Teemu Parkkinen toimi sovelluskehityksen ohjaajana ja asiantuntijana.

4 Käytetyt työkalut

4.1 Perustelut valituille työkaluille

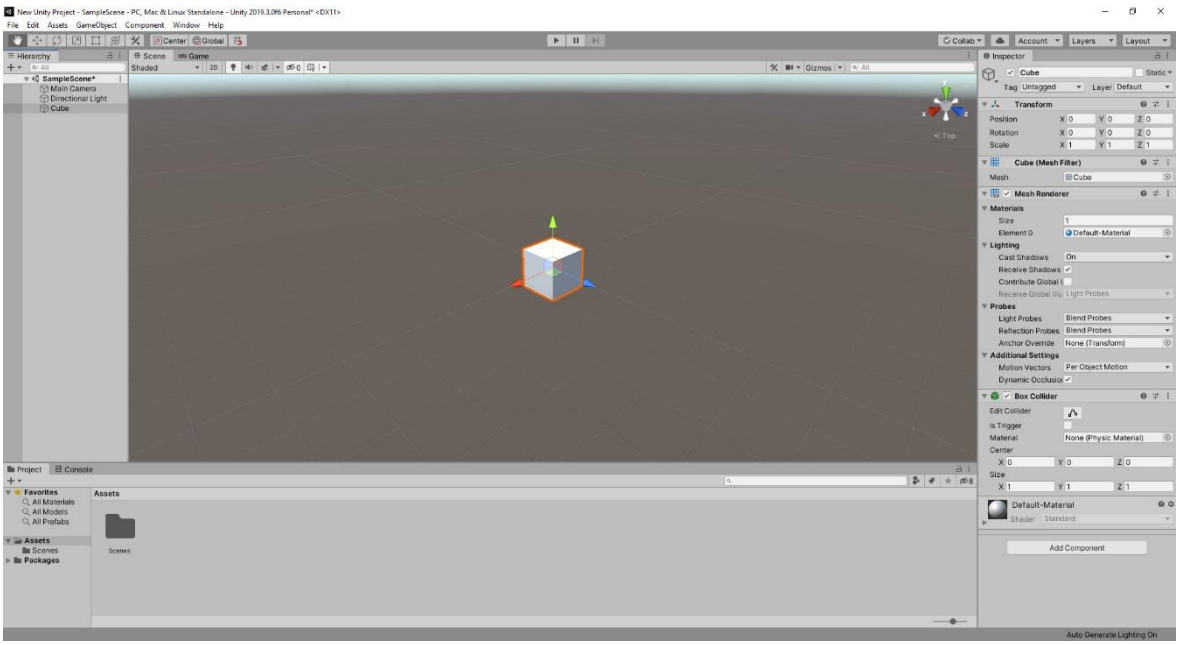
Käytettyjen työkalujen valintaan vaikutti sekä toimeksianto, että aikaisempi kokemus. Sovelluksen lopullinen käyttöalusta eli iPad määritteli sen, että sovelluksen kehitys tulisi tapahtumaan Applen Macbookilla, sillä Applen laitteille kehitys on mahdollista vain Applen tietokoneilla, hyödyntäen Xcode-ohjelmankehitysympäristöä.

Sovelluksen luonteen takia sen kehitys oli järkevää toteuttaa pelimoottoria hyödyntäen. Valmiita pelimoottoreita on olemassa useita ja monet lisätyn todellisuuden ohjelmistokehykset tyypillisesti tukevat etenkin Unity- ja Unreal Engine -pelimoottoreita. Toteutuksessa päädyttiin käyttämään Unity-pelimoottoria aikaisemman kokemuksen vuoksi. Unityn yhteydessä tehtävään koodaukseen ohjelmankehitysympäristöksi valikoitui hyvin luonnollisesti sen tukema Visual Studio. Tai oikeammin Visual Studio for Mac, koska kehitys tapahtui Macbookilla.

Lisätyn todellisuuden ohjelmistokehykseksi mietittiin muutamia eri vaihtoehtoja. Vaihtoehtoina oli esimerkiksi Vuforia ja MAXST, mutta lopulta päädyttiin ARFoundationiin. ARFoundationin valintaa puolsi useampikin syy. Se on Unityn kanssa yhteensopiva ja se mahdollistaa lisätyn todellisuuden toteuttamisen monella eri tekniikalla. Lisäksi sen käyttö on täysin ilmaista.

4.2 Unity

Unity on Unity Technologiesin kehittämä pelimoottori. Pelimoottori on ohjelmisto, joka tarjoaa sovelluskehittäjälle työkalut pelien tai sovellusten nopeaan ja kokonaisvaltaiseen kehittämiseen. Pelimoottorissa yhdistyy pelinkehityksen jokainen vaihe. Se mahdollistaa 2D- ja 3D-grafiikan ja äänien tuonnin ulkoisista lähteistä, animaatiotyöskentelyn, erikoisefektien ja ympäristöjen luonnin, fysiikkamoottorin hyödyntämisen sekä pelilogiikan luonnin ohjelmoinnin avulla. Unityn avulla kehittäjä pystyy kehittämään ja julkaisemaan tuotteensa monille eri alustoille, kuten erilaisille mobiililaitteille, tietokoneiden käyttöjärjestelmille, pelikonsoleille tai web-sivustolle. (Unity Technologies 2020a.) Kuvassa 2 on näyttökaappaus Unityn peruskäyttöliittymästä, jonka keskellä suurimpana näkyy Scene-näkymä, jossa tapahtuu suurin osa pelimaailman luonnista.



Kuva 2. Unityn käyttöliittymä

Unityssa pelilogiikka luodaan skriptikomponenttien avulla. Skriptikomponentit ovat tekstitiedostoja, jotka sisältävät ohjelmistokoodia. Niiden tarkoituksena on laukaista pelissä tapahtumia, muokata ominaisuuksia ja reagoida käyttäjältä saatuun syötteeseen, kuten napin painallukseen. Unityn käyttämät skriptit koodataan sen tukemalla C#-ohjelmointikielellä. (Unity Technologies 2020b.) Itse skriptien koodaus tapahtuu kuitenkin erillisessä, Unityn ulkopuolisessa, integroidussa ohjelmankehitysympäristössä. Ohjelmankehitysympäristöt ovat eräänlaisia tekstinmuokkausohjelmistoja, joihin on lisätty kehitystyötä, kuten koodausta helpottavia ominaisuuksia. Oletusarvoisesti Unityn mukana asentuu Visual Studio -ohjelmankehitysympäristö, mutta myös kevyempi Visual Studio Code sekä JetBrains Rider ovat tuettuja. MacOS-käyttöjärjestelmille oletuksena toimii Visual Studio for Mac. (Unity Technologies 2020c.)

4.3 Visual Studio for Mac

Visual Studio for Mac on avoimen lähdekoodin integroitu ohjelmankehitysympäristö Applen MacOS-käyttöjärjestelmille. Se on uudelleenbrändätty versio MacOS-käyttöjärjestelmille MonoDevelop-ohjelmankehitysympäristöstä. Visual Studio for Mac tarjoaa lisäominaisuuksia iOS- ja Android-sovellusten kehitykseen ja sen ensisijaisena tehtävänä on mahdollistaa avoimen lähdekoodin Mono- ja Microsoftin .NET-ohjelmistokehysten käyttö sovelluskehityksessä (MonoDevelop Project 2020a).

Visual Studio for Macin ominaisuuksiin lukeutuu monia Microsoft Visual Studion ominaisuuksia ja Microsoft toimiikin yhtenä sen kehittäjistä. Sen ensisijaiseen toiminnallisuuteen kuuluu koodin muokkaus, debuggaus eli virheenjäljitys, koodin buildaus eli koodeista toimivan ohjelman kääntäminen ja sovellusten julkaisu. (Microsoft 2019; MonoDevelop Project 2020b.) Projektin kannalta olennaista on, että se tukee alustariippumatonta peli- ja sovelluskehitystä C#-ohjelmointikielellä Unityn yhteydessä sekä iOS-kehitystä.

4.4 Xcode

Xcode on Applen kehittämä integroitu ohjelmankehitysympäristö. Sen tarkoituksena on mahdollistaa sovelluskehitys Applen laitteille. Xcode tarjoaa työkalut sovelluskehityksen alusta loppuun aina sovelluksen luonnista, testaukseen, optimisointiin ja App Storeen julkaisuun asti. (Apple Inc. 2020a.) Xcoden käyttö on ainoa käyttökeino, jos sovellus halutaan julkaista Applen laitteille.

Tässä projektissa Xcoden tehtävänä oli toimia kääntäjänä Unityn luomalle Xcode-projektitiedostolle. Se kääntää projektitiedoston valmiiksi sovellukseksi, jota voidaan käyttää esimerkiksi iPadilla tai iPhonella. Tämän lisäksi Xcodella ladattiin testiversiot Applen App Storen TestFlight-ohjelmaan, jonka kautta sovelluksen eri testiversioita pystyttiin jakamaan projektin osapuolten välillä ja testaamaan. Myös lopullinen sovelluksen julkaisuversio ladattiin Xcodella App Storeen, josta tilaaja sai ladattua sen käyttöönsä.

4.5 ARFoundation

ARFoundation on Unity Technologiesin kehittämä ilmainen ja alustariippumaton lisätyn todellisuuden ohjelmistokehys. Sen tehtävänä on mahdollistaa lisätyn todellisuuden sovelluskehitys Unitylla monille eri alustoille, kuten Androidille, iOS:lle, Magic Leapille ja Microsoft HoloLensille. Se ei itsessään implementoi mitään lisätyn todellisuuden toiminnallisuutta vaan toimii yhteisenä rajapintana eri alustojen omien lisätyn todellisuuden liitännäisille. (Unity Technologies 2020d.) Tämä mahdollistaa saman ratkaisun ja koodin käytön eri alustoilla.

Kohdealustasta tai -alustoista riippuen ARFoundation siis tarvitsee toimiakseen tiettyjä liitännäispaketteja tuekseen. Unityn tarjoamiin virallisiin liitännäispaketteihin kuuluu

- ARCore XR Plugin Android-toteutuksiin
- ARKit XR Plugin iOS-toteutuksiin
- Magic Leap XR Plugin Magic Leap -toteutuksiin

- Windows XR Plugin HoloLens-toteutuksiin.

Näistä liitännäisistä ARKit-liitännäinen tarjoaa laajimmat työvälineet lisätyn todellisuuden toteuttamiseen (Unity Technologies 2020d.) Koska sovellus kehitettiin iOS-laitteelle, vaatii se tuekseen Unityn ARKit-liitännäisen. Tämän avulla lisätty todellisuus voidaan toteuttaa ARFoundationilla, ja se pääsee käsiksi iOS-laitteen natiiviominaisuuksiin (Unity Technologies 2020e). ARFoundation antaa kehittäjälle työkalut luoda lisätyn todellisuuden sovelluksia hyödyntäen monia eri tekniikoita, joita käydään seuraavaksi läpi.

4.5.1 Device tracking

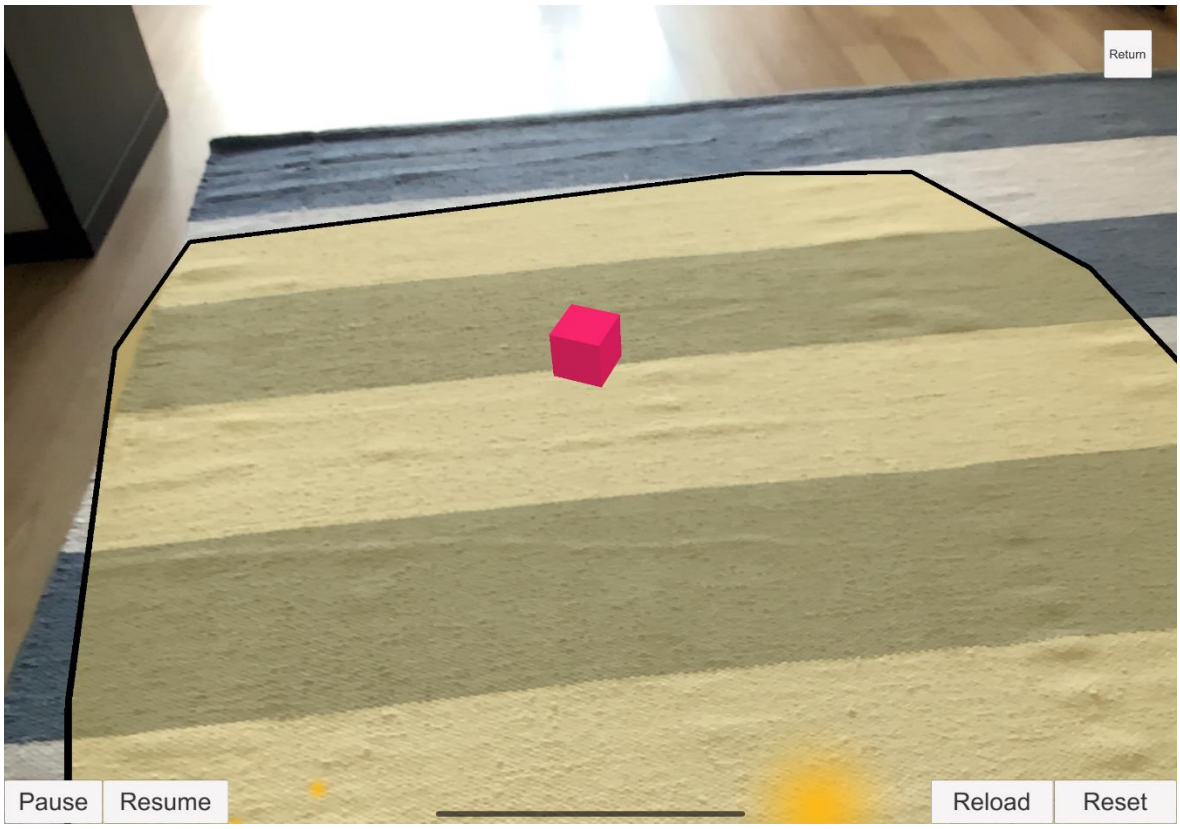
Device tracking eli laitteen seuranta on juuri sitä miltä se kuulostaakin. Sovellus seuraa laitteen sensorien avulla laitteen omaa relatiivista sijaintia ja suuntaa fyysisessä tilassa, jonka pohjalta se määrittelee sijaintinsa virtuaalisessa maailmassa (Unity Technologies 2020d).

Device tracking turvautuu ensisijaisesti laitteen gyroskooppiin, kiihtyvyysantureihin, kompassiin, eikä kameraan. Se on enemmän pohjaratkaisu, jota käytetään muiden ratkaisujen rinnalla, jotta laitteen liike saadaan tuotua tarkemmin virtuaaliseen tilaan. Näin se myös vahvistaa lisätyn todellisuuden vikasietoisuutta tuomalla käyttöön lisää sensoreita. Sillä ei kuitenkaan saada tietoa ympäristön yksityiskohdista, joten lisätyn todellisuuden toteuttaminen laadukkaasti yksinään sitä hyödyntäen on haastavaa. Samasta syystä se voi myös olla ainoa toimiva ratkaisu tilanteissa, joissa ympäristöstä on haasteellista saada informaatiota.

4.5.2 Plane detection

Plane detection on tasojen tunnistusta. Sen tarkoituksena on tunnistaa ympäristöstä tasaisia tasoja, kuten lattia, seiniä tai pöytiä. Tasojen tunnistus voidaan määrittää pelkästään horisontaaliseksi, jolloin se tunnistaa vain horisontaalisia tasoja, kuten lattian. Vaihtoehtoisesti se voidaan määrittää myös pelkästään vertikaaliseksi, esimerkiksi seinämien tunnistamiseksi. Näitä tunnistustapoja on myös tarvittaessa mahdollista hyödyntää samanaikaisesti. (Unity Technologies 2020f.)

Tunnistettuja tasoja pystytään visualisoimaan käyttäjälle luomalla virtuaalinen pinta tunnistetun tason rajapisteiden mukaan. Kuvassa 3 on havainnollistettu, kuinka tälle luodulle pinnalle voidaan sitten asetella virtuaalisia esineitä. Tasoja ei kuitenkaan ole pakko visualisoida, mutta se helpottaa virtuaalisten kohteiden asettelua tilaan, kun käyttäjä näkee missä tunnistetut tasot sijaitsevat. Visualisointi voidaan myös keskeyttää esimerkiksi kohteiden asettelun jälkeen, ettei se vie käyttäjän huomiota varsinaisesta sisällöstä.



Kuva 3. Kohde tunnistetulla tasolla

4.5.3 Point clouds

Point cloud eli pistepilvi muodostuu joukosta ympäristöstä tunnistettuja feature pointeja eli pisteitä. Tyypillisesti nämä pisteet ovat joitain huomattavia yksityiskohtia ympäristössä. Jokaisella pisteellä on oma sijaintinsa, tunnisteensa sekä luottamusarvonsa, joiden perusteella sovellus pystyy määrittelemään ympäristön suhteita ja laitteen sijainnin tilassa. (Unity Technologies 2020g.)

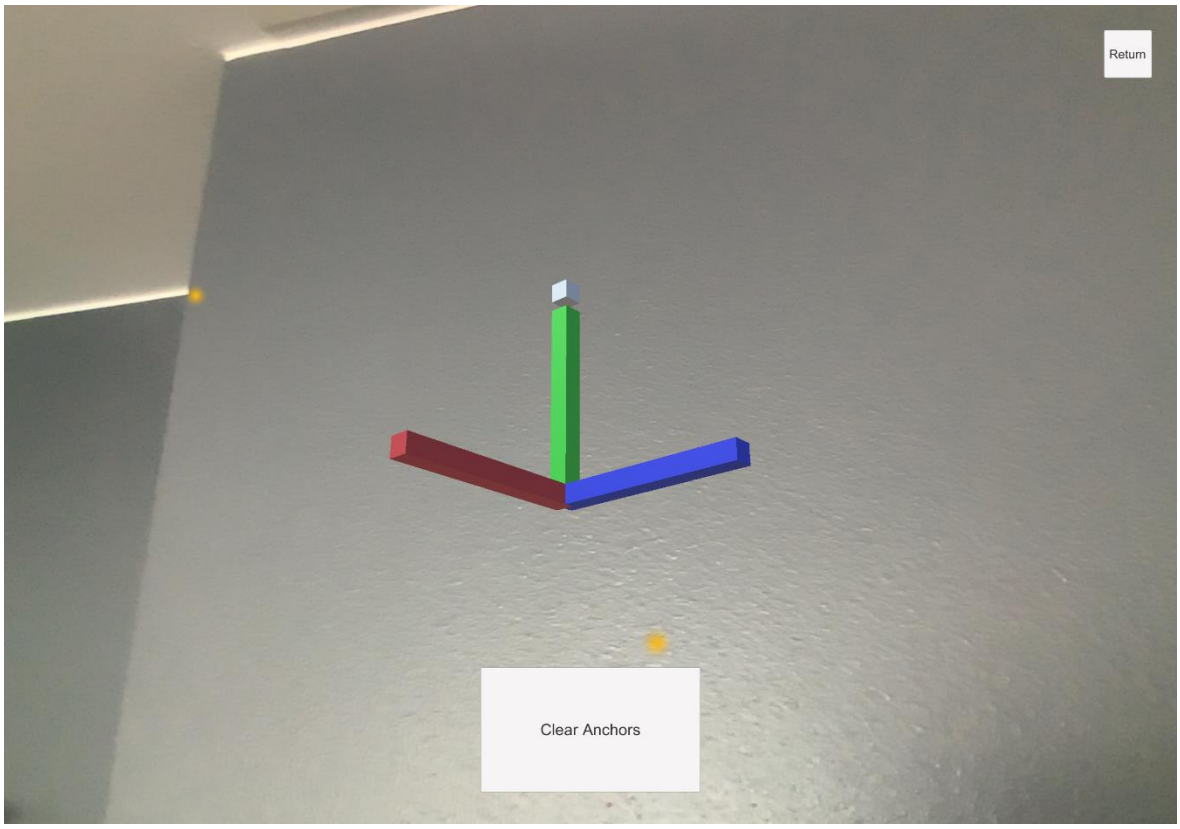
Pistepilviä voidaan hyödyntää, jos tilassa ei ole kunnolla tunnistettavia pintoja, mutta ympäristöstä löytyy joitakin tunnistettavia yksityiskohtia. Niitä voidaan myös käyttää muiden ratkaisujen tukena. Esimerkiksi jos on tunnistettu lattia ja käyttäjä suuntaa laitteen ylöspäin, saadaan tunnistettu lattian taso pysymään paremmin paikoillaan, kun se on kameran kuvan ulkopuolella, jos sovellus muodostaa pistepilven vaikkapa kattovalaisimesta. Kuvassa 4 näkyy, kuinka sovellus on luonut pistepilven kattovalaisimen yksityiskohdista.

Kuva 4. Pistepilvi ympäristön yksityiskohdista

4.5.4 Ankkurit

Ankkurit ovat käyttäjän määrittelemiä pisteitä, joita sovelluksen halutaan seuraavan. Kuvassa 5 on ilmaan asetettu ankkuri, josta näkyy sen x-, y- ja z-koordinaattiakselit. Ankkureita käytettäessä, laite joutuu suorittamaan lisävaiheita pitääkseen ankkurit kohdillaan. Tästä syystä niiden käyttö on resursseja vaativaa, joten niitä olisi syytä käyttää säästeliäästi. (Unity Technologies 2020h.)

Ankkurien käytössä on muistettava, että se tukeutuu aina johonkin jo aikaisemmin läpikäydyistä tekniikoista. Jotta sovellus pystyy määrittelemään ankkurin sijainnin, sen on myös tiedettävä jotain ympäristöstä tai käytettävän laitteen sijainnista. Ankkurin sijainti määrittyy siis suhteessa laitteen sijaintiin tai esimerkiksi tunnistettuun tasoon.



Kuva 5. Ilmaan asetettu ankkuri

4.5.5 Face tracking

Face tracking perustuu kasvojen tunnistukseen. Sen avulla pystytään tunnistamaan yhdet tai useammat ihmiskasvot kameran tuottamasta kuvasta ja seurata niitä. Seuratuille kasvoille voidaan halutessa visualisoida digitaalista sisältöä. (Unity Technologies 2020i.) Face tracking mahdollistaa myös silmien liikkeen ja kasvojen suunnan seurannan (Unity Technologies 2020j). Näin ollen sisältöä voidaan sijoittaa myös sinne, minne henkilön katse osoittaa.

Face tracking -toiminnallisuuden yleisin käyttökohde on erilaisissa kamerasovelluksissa. Sen avulla voidaan luoda erilaisia henkilön kasvonpiirteitä ja ilmeitä mukailevia kamerafiltereitä sekä -efektejä. Kuvassa 6 on esimerkki face tracking -toiminnallisuuden avulla luodusta kameraefektistä, jossa henkilön kasvoille on asetettu laiskiaisen pää.

Kuva 6. Face tracking -toiminnallisuudella luotu kameraefekti

4.5.6 Image tracking

Image tracking -toiminnallisuuden tarkoituksena on tunnistaa ympäristöstä kuvia tai kuvioita, ja näyttää virtuaalista sisältöä suhteessa niihin. Image tracking vaatii toimiakseen lähdekirjaston, joka koostuu joukosta ympäristöstä tunnistettavaksi tarkoitettuja kuvia. Se vertaa kameran tuottamasta kuvasta ympäristön piirteitä lähdekirjastonsa kuviin ja yrittää löytää täsmääviä kuvia. Kun sovellus tunnistaa kuvan, luodaan kuvan kohdalle sille valmiiksi määritetty kohde. (Unity Technologies 2020k; Unity Technologies 2020l.) Kuvassa 7 on havainnollistettu, kuinka sovelluksen tunnistaessa kuvan, kuvan eteen ilmestyy virtuaalista sisältöä.

Kuva 7. Kuvan tunnistus

Tunnistettavana kuvana ei voida käyttää ihan mitä tahansa kuvaa. Kuva ei saa olla yksinkertainen kuvio, kuten yksittäinen viiva tasaisella taustalla. Syy tähän on se, että sovellus saattaisi virheellisesti tunnistaa jokaisen ympäristössä olevan viivan. ARFoundation automaattisesti hylkää kuvan, jos se ei täytä sen kriteereitä. Kuvassa olisi hyvä olla paljon selkeitä piirteitä, suuria värieroja ja mahdollisimman uniikki sisältö. Tällöin se erottuu selkeästi ympäristöstään.

Fyysisen kuvan näkyvyys vaikuttaa mitä virtuaaliselle sisällölle tapahtuu. Kun kuvan näkyvyys on rajallinen tai kuva katoaa kokonaan kameran näköpiiristä, voidaan määritellä niin, että virtuaalinen sisältö jätetään näkyviin tai poistetaan. Jos virtuaalinen sisältö halutaan poistaa, se saadaan ilmestymään takaisin, kun kuva tunnistetaan uudelleen. ARFoundation mahdollistaa kuvien tunnistuksen myös useammasta liikkuvasta kohteesta. (Unity Technologies 2020l.)

4.5.7 Object tracking

Object tracking on toiminnallisuudeltaan hyvin samankaltainen kuin image tracking. Myös se vaatii lähdekirjaston, johon sovellus vertaa ympäristön yksityiskohtia. Tässä tapauksessa erityisesti esineitä. Object tracking -ratkaisun lähdekirjasto ei kuitenkaan koostu image tracking -ratkaisun tapaan kuvista vaan joukosta skannattuja kolmiulotteisia esineitä. (Unity Technologies 2020m.) Ympäristöstä esineen tunnistettuaan, sovellus näyttää määritettyä sisältöä. Kuvassa 8 esineen kohdalle ilmestyy koordinaattiakselisto, kun sovellus on tunnistanut sen vertailtuaan sitä lähdekirjaston kohteisiin.

Object tracking on Applen ARKit-ohjelmistokehyksen tuomaa toiminnallisuutta ja vaatii iOS-laitteen sekä Unityn ARKit-laajennoksen toimiakseen. Tästä syystä myös tunnistettavat

esineet täytyy skannata Applen tarjoamalla iOS-laitteilla toimivalla skannaussovelluksella. Skannauksessa suositellaan seuraamaan seuraavia ohjeita:

- Valaise skannattava esine 250-400 luksin valaistusvoimalla, ja varmista että kohde on hyvin valaistu joka puolelta.
- Käytä valaistukseen päivänvaloa vastaavaa noin 6500 kelvinin värilämpötilaa ja vältä muun värisiä valonlähteitä.
- Aseta tunnistettava esine keskiharmaalle, matalle taustalle.

Näitä ohjeita seuraamalla saadaan paras mahdollinen skannaustulos. Skannaussovellus ohjeistaa käyttäjälle skannauksen vaiheet alusta loppuun. (Apple Inc. 2020b.)

Kuva 8. Esineen tunnistus

4.5.8 Body tracking

Kuten object tracking, on body tracking myös Applen ARKit-ohjelmistokehyksen tuomaa toiminnallisuutta ja vaatii iOS-laitteen sekä Unityn ARKit-laajennoksen toimiakseen. Sen avulla voidaan tunnistaa ihmishahmo kameran tuottamasta kuvasta, ja sitä hyödyntämällä pystytään reaaliaikaisesti seuraamaan ihmiskehon liikkeitä eli toisin sanoen tehdä liikkeen-kaappausta. Kaapattua liikettä voidaan tarpeen mukaan esittää joko kaksiulotteisena tai kolmiulotteisena tuotoksena. (Apple Inc. 2020c; Apple Inc. 2020d.) Kuvassa 9 näkyy kuinka iPadilla sovellus tunnistaa tanssivan ihmisen kehon ja jäljittelee tämän liikkeitä kolmiulotteisella luustorakenteella.

Kuva 9. Ihmiskehon seurantaa

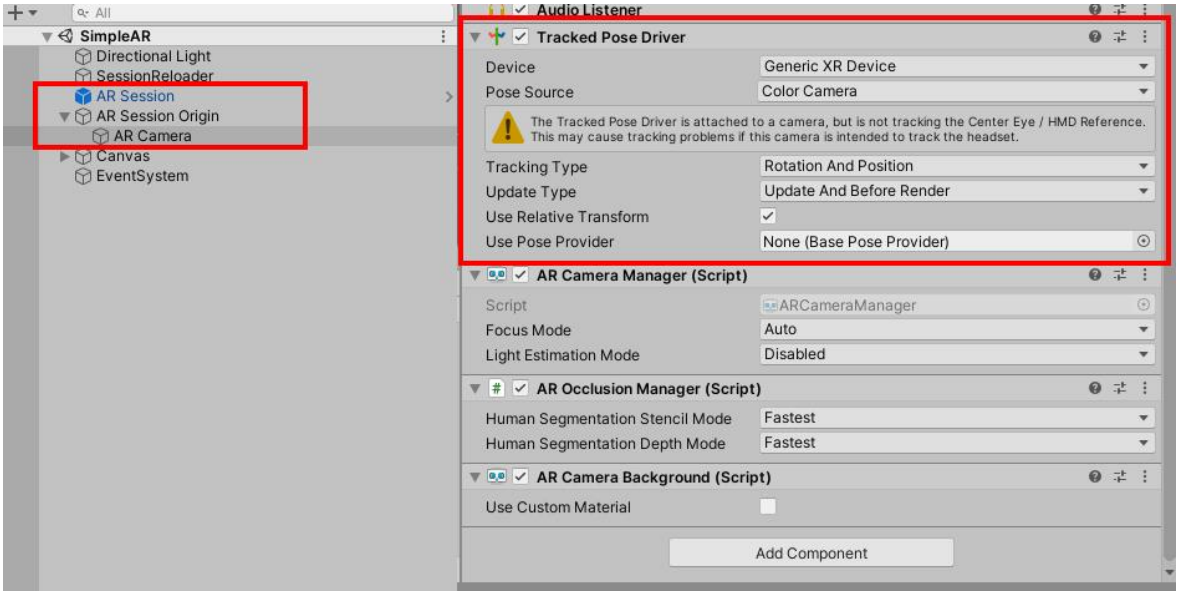
Body tracking -toiminnallisuuden avulla pystytään myös nostamaan lisätyn todellisuuden uskottavuutta. Normaalisti virtuaaliset elementit renderöidään kameran näyttämän kuvan päälle, jolloin ne peittävät todelliset kohteet siitäkin huolimatta, että niiden sijainti on kauempana laitteen kamerasta. Body tracking mahdollistaa virtuaalisten kohteiden edestä kulkevien ihmishahmojen tunnistamisen, jolloin virtuaalista kohdetta ei hahmon kohdalta näytetäkään (Apple Inc. 2020e). Tällöin virtuaaliset kohteet jäävät niiden edessä olevien ihmisten taakse ja peittyy niiltä osin.

5 Sovelluksen kehitys

5.1 ARFoundationin käyttöönotto

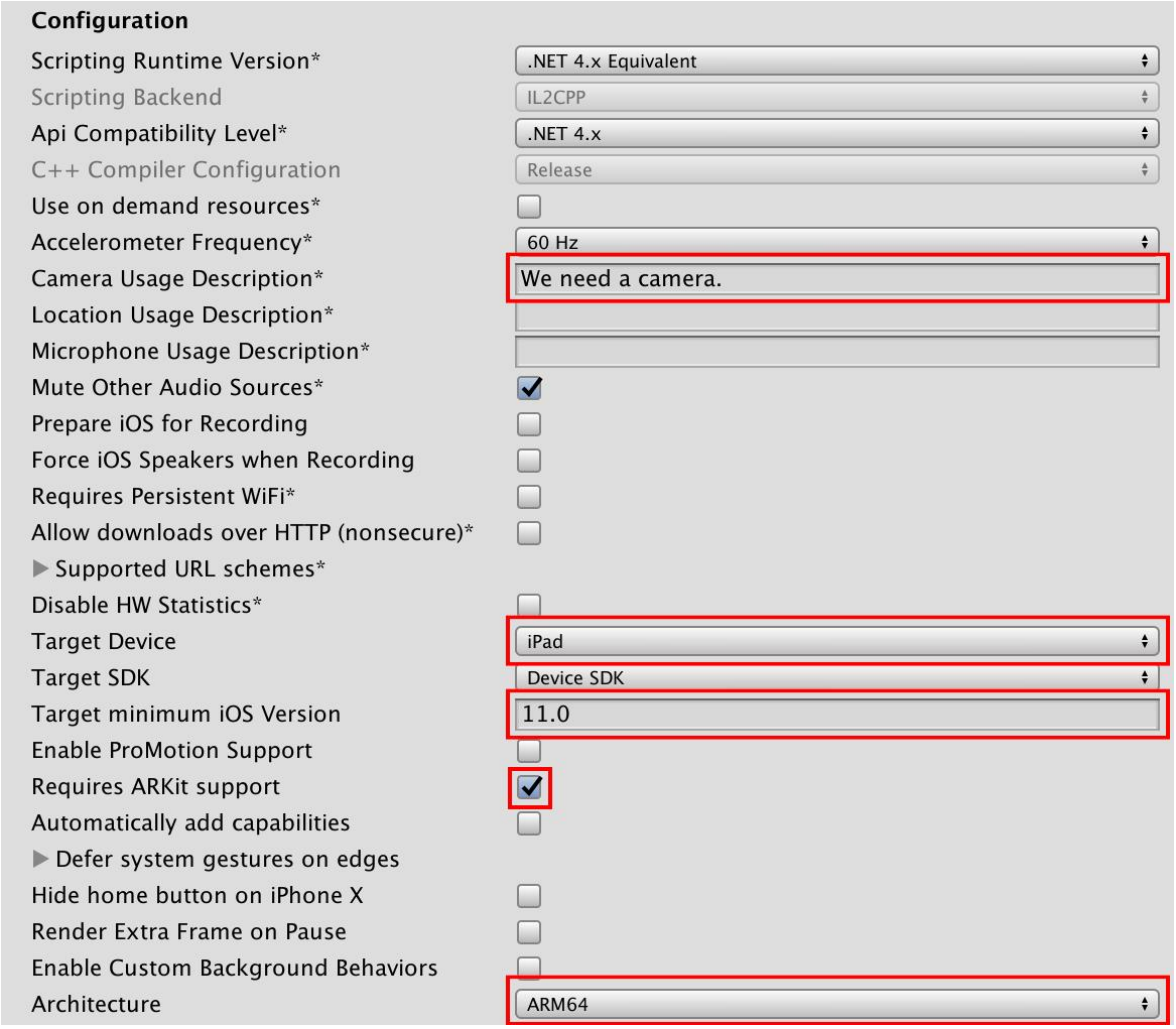
ARFoundationin käyttöönotto tapahtuu Unityssa nopeasti ja vaivattomasti. Ensimmäiseksi mennään Unityn Package Manageriin, josta valitaan ARFoundation ja ladataan se. Package Manager lataa automaattisesti ARFoundationin ja muut sen tarvitsemat paketit. Koska sovellus kehitettiin iPadille, täytyi Package Managerilla lisäksi ladata ARKit XR -liitännäinen. Tämä mahdollistaa ARFoundationin ja ARKitin toiminnallisuuden käytön iOS-laitteille tehdyissä sovelluksissa.

Kun ARFoundation on ladattu, asetetaan Unityn Scene-näkymään ARSession ja ARSession Origin -komponentit. ARSession-komponentti kontrolloi lisätyn todellisuuden istunnon asetuksia ja elinkaarta. ARSession Origin -komponentin tehtävä on toimia emokomponenttina virtuaaliselle kameralle ja ympäristöstä tunnistetuille ja seuratuille yksityiskohdille, kuten tasoille. Se pitää huolen, että virtuaaliset kohteet pysyvät asemissaan suhteessa kameraan. Lisätyn todellisuuden istunnossa, laitteen aloitussijainti on virtuaalitilassa pisteessä (0, 0, 0), johon kaikki virtuaalinen sisältö suhteutetaan. Laitteen seurantaa varten lisätään vielä Tracked Pose Driver tai AR Pose Driver -valmisskripti ARSession Originin alla olevaan AR Camera -komponenttiin. Kuvassa 10 on punaisella ympäröity Scene-näkymään asetetut komponentit sekä AR Cameraan liitetty Tracked Pose Driver -komponentti. Lopuksi ARSession Originiin lisätään ratkaisun mukaan tarvittavat Manager-komponentit.



Kuva 10. ARFoundationin käyttöönottoa

Jotta sovelluksen saa toimimaan iPadilla, täytyy vielä tehdä muutamia muutoksia Unityn Player Settingsin iOS-osion konfiguraatioon. Konfiguraatiosta valitaan kohdelaitteeksi vähintään iPad, käyttöjärjestelmän minimivaatimukseksi asetetaan iOS Version 11.0, arkkitehtuuriksi valitaan ARM64, vaaditaan ARKit-tuki sekä annetaan kameralle käyttökuvaus. Kameran käyttökuvauksen sisällöllä ei ole varsinaisesti mitään merkitystä, kunhan se ei ole tyhjä. Kuvassa 11 on esitettynä Player Settingsin iOS-konfiguraatio, josta tärkeimmät kohdat on ympäröity punaisella.



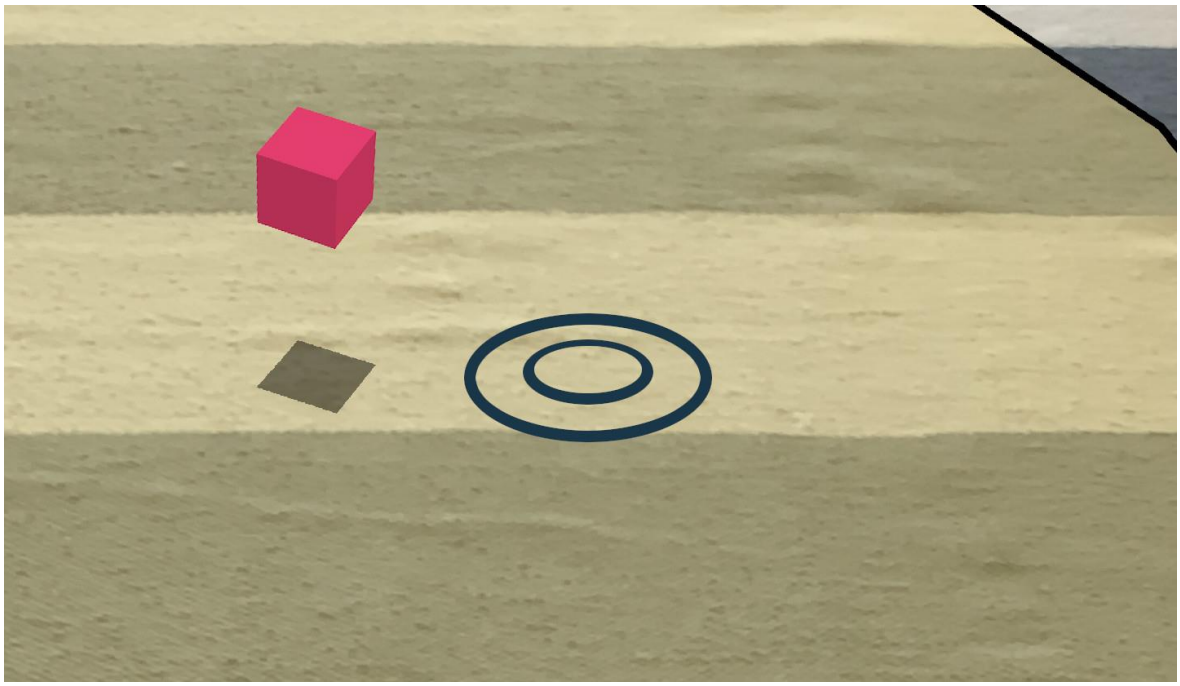
Kuva 11. iOS-konfiguraatio

5.2 Plane detection -ratkaisu

Koska toivottiin että virtuaaliset kohteet leijuisivat ilmassa tietyllä korkeudella lattiasta ja niitä pystyisi asettelemaan tilaan vapaasti, päätettiin sovellus toteuttaa plane detection -

ratkaisulla. Tällä ratkaisulla sovellus tunnistaisi näyttelytilan lattian ja näin ollen kohteet asettuisivat helposti halutulle korkeudelle ja niiden varjot lattian tasolle.

Ratkaisu toteutettiin niin, että näytteilleasettaja tunnistaisi laitteella lattian tason ja asettaisi kohteet tunnistetulle tasolle yksitellen sijoitusindikaattorin osoittamaan kohtaan. Sijoitusindikaattori luotiin säteensuuntauksella eli ikään kuin ampumalla näkymättömällä säteellä laitteen kameran sijainnista tunnistettua tasoa kohti. Sijoitusindikaattori tulee näkyviin tasolle siihen kohtaan, missä säde osuu tasoon. Aseteltavat kohteet koostuvat kolmiulotteisesta mallista ja sen keinotekoisesta varjosta. Kohteiden origot asetettiin varjoihin, jolloin varjo asettuisi suoraan tunnistettuun tasoon. Kuvassa 12 havainnollistetaan, kuinka sijoitusindikaattori ilmestyy tunnistetulle tasolle ja kuinka tasolle asetettu kohde asettuu ilmaan varjoonsa määritetyn origon mukaan.



Kuva 12. Havainnollistus plane detection -ratkaisusta

5.2.1 Plane detection -toiminnallisuuden toteutus

ARFoundationin käyttöönoton jälkeen ARSession Originiin lisätään AR Plane Manager (Script) -komponentti. Komponentin asetuksista määritellään, halutaanko tunnistettuja tasoja visualisoida. Lisäksi valitaan, tunnistetaanko vertikaalisia, horisontaalisia vai kumpiakin tasoja. Tässä vaiheessa sovellus osaa tunnistaa ympäristön tasoja, mutta niiden hyödyntämiseksi tarvitsee vielä luoda omaa ohjelmistokoodia.

Sovelluksen tunnistamien tasojen hyödyntäminen lähtee liikkeelle säteensuuntauksesta. Säteensuuntausta varten ARSession Originiin täytyy lisätä AR Raycast Manager (script) -komponentti. Komponentin tehtävänä on tuoda säteensuuntauksen toiminnallisuus AR-Foundationin käytettäväksi. Säteensuuntaus ohjelmoitiin niin, että säteen alkupiste on keskellä näkymää ja jatkuu siitä suorassa linjassa eteenpäin. Kun säde kohtaa matkallaan tunnistetun tason, tallennetaan tämä osumakohta placementPose nimiseen Pose-tyyppiseen objektiin. Pose-tyyppinen objekti koostuu kolmiulotteisesta sijainti- ja rotaatiodatasta. Tämän jälkeen käytetään hyödyksi kameraobjektin eli laitteen virtuaalisen tilan vastineen asentoa, korvaamalla placementPose-objektin rotaatio laitteen suunnalla. Seuraavaksi placementPosen sisältämää dataa hyödyntämällä luodaan visuaalinen indikaattori kohteiden asettelua helpottamaan. Näin indikaattorin sijainniksi määräytyy säteensuuntauksen osumakohta tasossa ja rotaatioksi laitteesta saatu suunta. Tämä prosessi toistuu jatkuvasti laitteen kuvan päivittyessä ja indikaattorin sijainti tasolla päivittyy sen mukaan mihin säde osuu. Kuvan 13 koodiesimerkissä on metodit placementPosen ja indikaattorin päivittämiseksi. Lopuksi lisättiin nappula, jota painamalla virtuaalinen kohde asettuu tasolle, indikaattorin osoittamaan kohtaan placementPosen datan perusteella. Laitteesta saatu suunta pitää huolen, että virtuaaliset kohteet asettuvat laitetta kohti.

```
118     private void UpdatePlacementPose() {
119         // Get the center point of the apps view
120         Vector3 screenCenter = Camera.main.ViewportToScreenPoint(new Vector3(0.5f, 0.5f));
121         // Create a list to store current raycast hits
122         List<ARRaycastHit> hits = new List<ARRaycastHit>();
123         // Raycast from the center of the screen, store hits to the list, but only if they hit an infinite plane
124         arRaycastManager.Raycast(screenCenter, hits, TrackableType.PlaneWithinInfinity);
125 
126         // Check if there is at least one hit on the list
127         placementPoseIsValid = hits.Count > 0;
128         if (placementPoseIsValid) {
129             // Store the pose value of the first hit to placementPose
130             placementPose = hits[0].pose;
131 
132             // Get camera's (device's) bearing
133             Vector3 cameraForward = Camera.main.transform.forward;
134             Vector3 cameraBearing = new Vector3(cameraForward.x, 0, cameraForward.z).normalized;
135 
136             // Update placementPose's rotation to camera's bearing
137             placementPose.rotation = Quaternion.LookRotation(cameraBearing);
138         }
139     }
140 
141     private void UpdatePlacementIndicator() {
142         // Check if there is a valid placement pose and activate the placement indicator
143         if (placementPoseIsValid) {
144             placementIndicator.SetActive(true);
145             // Update placement indicator's position and rotation according to placementPose
146             placementIndicator.transform.SetPositionAndRotation(placementPose.position, placementPose.rotation);
147         } else {
148             // If there is no valid placement pose, hide the indicator
149             placementIndicator.SetActive(false);
150         }
151     }
```

Kuva 13. Koodiesimerkki

5.2.2 Ratkaisu käytännössä

Plane detection -ratkaisun selkeä vahvuus oli, että asetetut kohteet pysyivät hyvin paikoillaan ja sensorivirheet pysyivät minimaalisina. Pian kuitenkin huomattiin, että näyttelytilassa lattian tunnistus onnistui vain lähietäisyydeltä lattian erottuvuudesta ja tilan hämäryydestä johtuen. Tämä ongelma ratkaistiin niin, että sovelluksen tunnistettua pienenkin alueen tasoa, se olettaa, että kyseinen taso jatkuu loputtomasti joka suuntaan. Tämä mahdollisti virtuaalisten kohteiden asettelun niin kauaksi kuin näytteilleasettaja vain pystyi laitteella osoittamaan. Lisäksi varmuuden vuoksi ohjelmoitiin pieni lisäosa, jonka avulla iPadin taskulamppu syttyy tasojen tunnistuksen ja kohteiden asettelun ajaksi.

Tässä vaiheessa sovellus oli täysin toiminnallinen, mutta ratkaisun luonteen takia kohteet täytyi asettaa virtuaaliseen tilaan uudestaan jokaisella käynnistyksellä. Tämä johtui siitä, että tunnistettuja pintoja ei teknisesti ollut mahdollista tallentaa. Se osoittautui ongelmalliseksi tilaajalle, joten tämän vuoksi päätettiin yrittää toisenlaista ratkaisua.

5.3 Object tracking -ratkaisusta lyhyesti

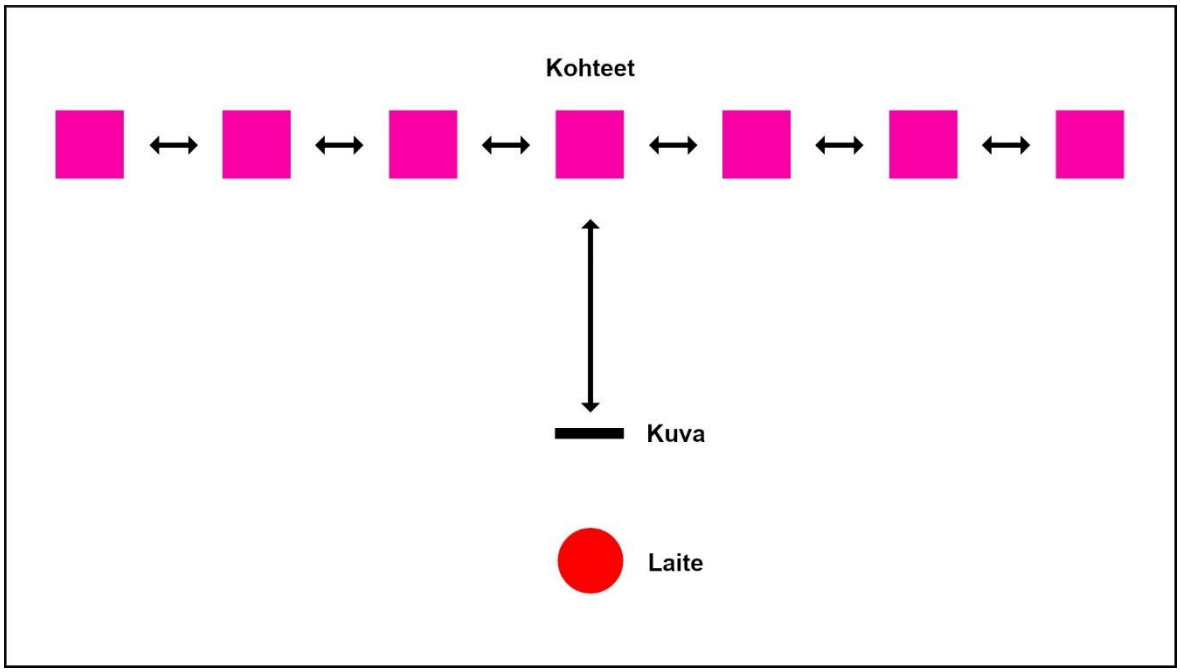
Uutta ratkaisua mietittäessä selvisi, että tilaajalla olisi näyttelytilassa tietynlainen selkeästi erottuva fyysinen elementti. Tämä mahdollistaisi object tracking -toiminnallisuuden käytön, jota päätettiin testata vaihtoehtoisena toteutustapana. Jo hyvin varhain kuitenkin selvisi, että esineiden tunnistaminen ei ole varteenotettava ratkaisu. Sen ongelmana oli, että sovellus tunnisti esineen vain, kun fyysisen tilan ympäristö, valaistus ja etäisyys laitteeseen vastasivat lähdekirjaston skannausta. Lisäksi huomattiin, että myös kohteen katselukulmalla oli vaikutus tunnistamiseen.

Koska object tracking todettiin käyttökelvottomaksi jo ensimmäisissä testeissä, ei sitä edes yritetty toteuttaa varsinaisen sovelluksen ratkaisuna. Jos se olisi toteutettu, toteutus olisi aloitettu luomalla ReferenceObjectLibrary-komponentti, joka toimisi lähdekirjastona tunnistettaville kohteille. Siihen olisi lisätty skannaus näyttelytilan elementistä, johon sovellus vertaisi ympäristöä. ARSession Originiin lisättäisiin AR Tracked Object Manager (Script) -komponentti, johon liitettäisiin luotu lähdekirjasto. Ratkaisussa kohteiden säädöt olisi toteutettu samalla tavalla, kuin seuraavaksi käsiteltävässä image tracking -ratkaisussa.

5.4 Image tracking -ratkaisu

Seuraava testattava ratkaisu löytyi image tracking -toiminnallisuuden käytöstä. Näyttelytilaan sijoitettaisiin kuva, johon suhteessa kohteet ilmestyisivät. Ratkaisun kannalta oli tärkeää tietää kuvan mittasuhteet ja mille korkeudelle kuva sijoittuisi, jotta virtuaalikohteiden

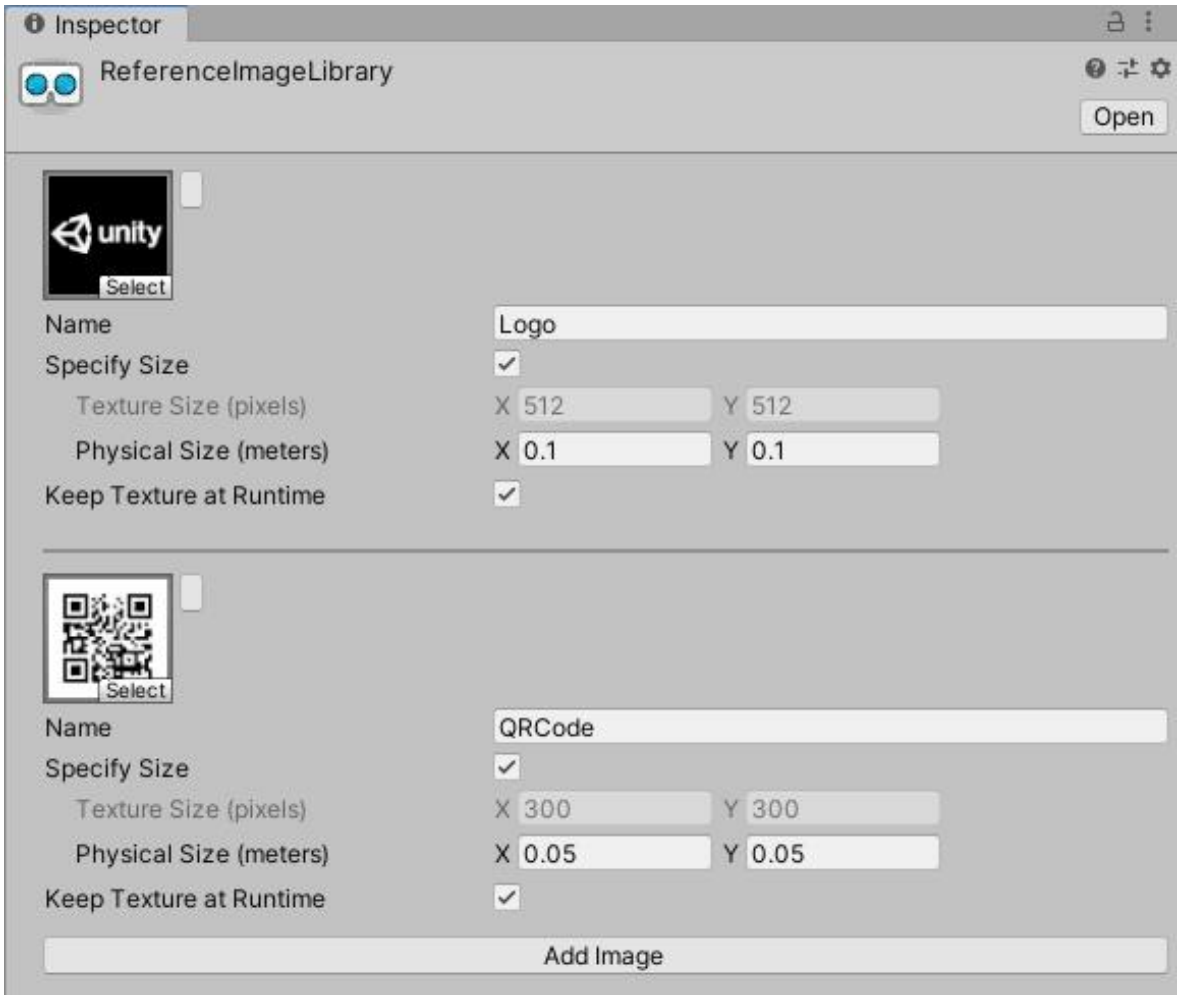
skaala ja sijainti saataisiin suhteessa kuvaan oikein sekä kohteiden varjot aseteltua lattian tasoon. Tässä ratkaisussa jouduttiin kuitenkin luopumaan kohteiden asettelusta vapaasti ympäri tilaa. Ratkaisuna päädyttiin asettelemaan kohteet vierekkäin, tasaisin välimatkoin kuvan kummallekin puolen. Näytteilleasettajalle annettiin myös mahdollisuus suurentaa kohteiden välitystä ja siirtää kohteet syvyyssuunnassa loitommaksi tunnistettavasta kuvasta. Kuviossa 1 on havainnollistettu kohteiden sijainnin määrittelyä ylhäältä käsin katsottuna.



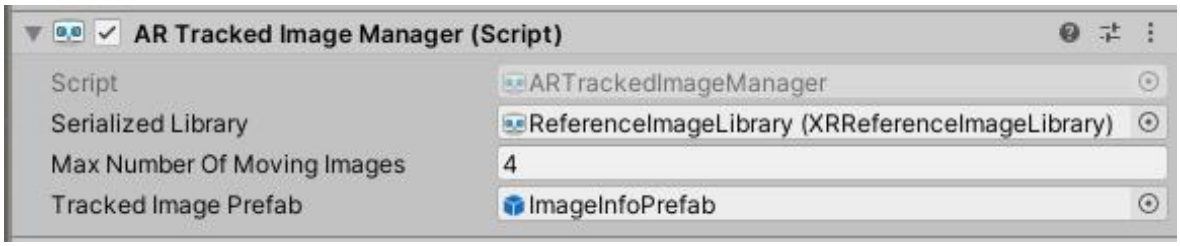
Kuvio 1. Kohteiden sijainnin määrittely

5.4.1 Image tracking -toiminnallisuuden toteutus

ARFoundationin käyttöönoton jälkeen luodaan ReferenceImageLibrary-komponentti, joka on lähdekirjasto tunnistettavista kuvista. Lähdekirjastoon lisätään kuvatiedostot, niiden nimet sekä kuvien fyysiset mittasuhteet metreissä. Kuvassa 14 on esimerkki luodusta lähdekirjastosta, johon on lisätty kaksi tunnistettavaksi tarkoitettua kuvaa. Lähdekirjaston luonnin jälkeen ARSession Originiin lisätään AR Tracked Image Manager (Script) -komponentti. Kuvassa 15 näkyy komponentin sisältö, johon liitetään juuri luotu kuvien lähdekirjasto, määritetään, kuinka montaa kuvaa sovelluksen tahdotaan seuraavan samanaikaisesti sekä kohde, joka ilmestyy tunnistetun kuvan kohdalle.



Kuva 14. Tunnistettavien kuvien lähdekirjasto

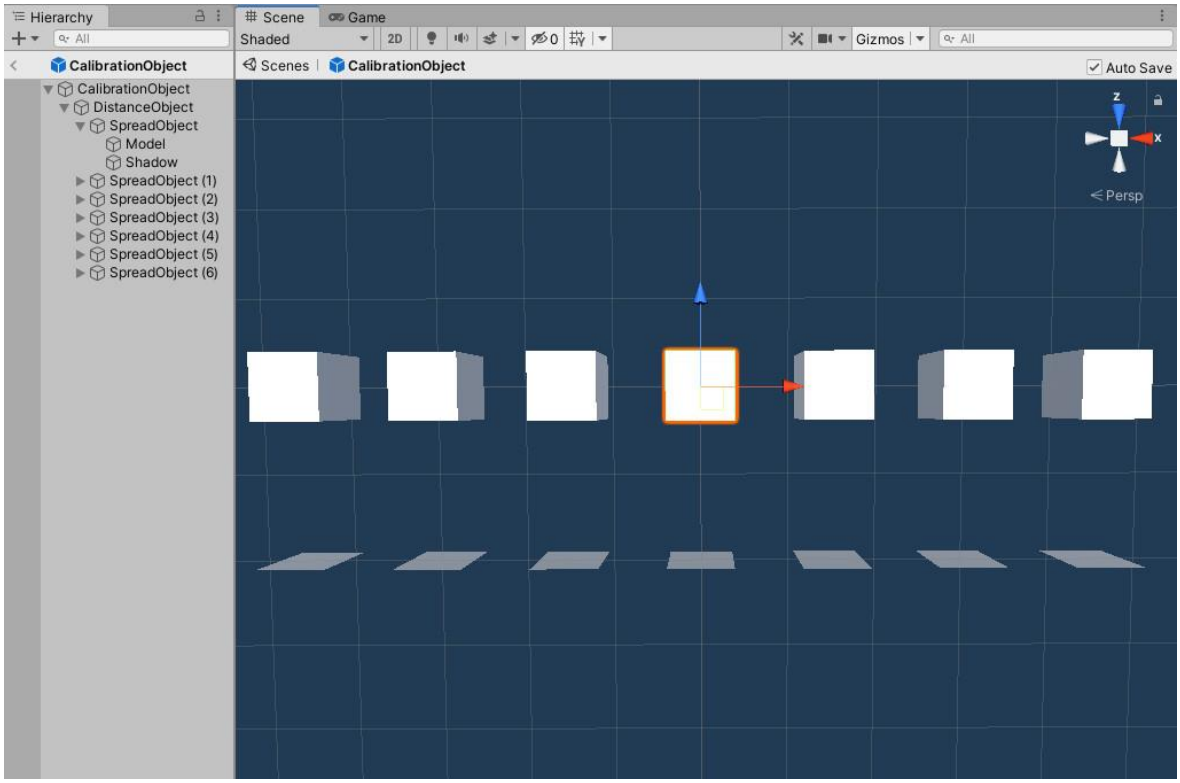


Kuva 15. AR Tracked Image Manager -komponentti

ARFoundationissa tunnistetulla kuvalla on kolme tracking statea eli seurannan tilaa, jotka ovat: tracked, limited ja none. Kun kuva on näkyvissä, seuranta on tracked-tilassa ja virtuaalinen sisältö tulee näkyviin. Jos kuvan näkyvyys on osittaista, seuranta vaihtuu limited-tilaan ja vastaavasti kuvan täysin kadotessa näkyvistä seuranta muuttuu none-tilaan. Kahdessa jälkimmäisessä tilassa ARFoundationin image tracking oletusarvoisesti poistaa kuvaan liittyvän kohteen ja palauttaa sen takaisin, kun kuva tunnistetaan uudelleen.

Sovelluksessa virtuaaliset kohteet eivät kuitenkaan saaneet kadota kesken kaiken, joten koodissa määritettiin, että kohteet pysyvät paikoillaan, vaikka seurannan tila muuttuisi.

Kuvan kohdalle ilmestyvän kohteen origo on samassa pisteessä kuin tunnistettavan kuvan keskipiste. Tämän takia olisi suositeltavaa, että kohde-esineet asetettaisiin hierarkkisesti tyhjän emokohteen alle, ellei erityisesti haluta, että virtuaalinen sisältö ilmestyy kuvan kohdalle. Näin kohde-esineiden sijainnit voidaan määritellä erikseen suhteessa emokohteeseen. Tällöin tyhjän emokohteen ilmestyessä kuvan kohdalle, sen alla olevat kohde-esineet ilmestyvät tilaan suhteessa emokohteeseen. Esimerkiksi metrin korkeuteen emokohteesta asetettu kohde-esine ilmestyisi metrin etäisyydelle tunnistetusta kuvasta. Toteutuksessa käytettiin tyhjää peliobjektia emokohteena, jonka alle asetettiin toinen tyhjä peliobjekti emokohteeksi seitsemälle muulle tyhjälle peliobjektille. Nämä puolestaan toimivat emokohteina niiden alle asetetuille 3D-malleille ja lattian tasoon asettuville varjoille. Kuvassa 16 on havainnollistettu, kuinka tämä kohteiden hierarkia rakentuu. Tähän hierarkkiseen ratkaisuun päädyttiin, koska 3D-mallien etäisyyttä kuvasta ja välitystä toisiinsa haluttiin pystyä säätämään. Tällä tavalla ylimmän tason emokohde pysyy tunnistettavan kuvan päällä, ja pelkästään sen alla olevaa peliobjektia siirtämällä saadaan säädettyä kohteiden etäisyys kuvaan. Vastaavasti välityksen säädön vaikuttaessa pelkästään alimmaisiin emokohteisiin, niihin sidoksissa olevat 3D-mallit ja varjot liikkuvat niiden mukana.



Kuva 16. Kohteen hierarkia

Säätöjä varten luotiin kaksi liukusäädintä. Yksi etäisyyden ja toinen välityksen kontrolloimiseen. Säätimien toiminnallisuus koodattiin seuraavanlaisesti. Etäisyyssäätimen ollessa nollassa, kohteet sijoittuvat kuvan kanssa samaan tasoon ja säätimen maksimiasennossa kymmenen metrin etäisyydelle siitä. Välityssäätimen ollessa minimiasennossaan, keskimmäistä kohdetta lähimpänä olevat kohteet sijoittuvat 1,5 metrin etäisyyteen suhteessa emokohteensa origoon. Maksimiasennossa lähimpien kohteiden etäisyys kasvaa kolmeen metriin. Näistä seuraavat kohteet sijoittuvat suhteessa kaksinkertaisen matkan päähän emokohteeseen ja ulommaiset kohteet kolminkertaisen. Kuvan 17 koodiesimerkissä näkyy liukusäätimien metodit etäisyyden ja välityksen muuttamiseksi. Näytteilleasettajan ollessa tyytyväinen säätöihin, etäisyys sekä lähimmän kohteen etäisyys tallennetaan laitteen muistiin. Sovelluksen käynnistyessä kohteet asettuvat tallennettujen arvojen mukaan.

```
120     private void ChangeDistance(float value) {
121         position.z = value;
122         transformObject.transform.localPosition = position;
123     }
124
125     private void ChangeSpread(float value) {
126         Vector3 pos = new Vector3();
127         foreach (Transform child in children) {
128             pos = child.localPosition;
129             switch (child.tag) {
130                 case "SpreadItem1":
131                     pos.x = value * -3;
132                     break;
133                 case "SpreadItem2":
134                     pos.x = value * -2;
135                     break;
136                 case "SpreadItem3":
137                     pos.x = value * -1;
138                     break;
139                 case "SpreadItem4":
140                     pos.x = value;
141                     break;
142                 case "SpreadItem5":
143                     pos.x = value * 2;
144                     break;
145                 case "SpreadItem6":
146                     pos.x = value * 3;
147                     break;
148                 default:
149                     break;
150             }
151             child.localPosition = pos;
152         }
153     }
```

Kuva 17. Koodiesimerkki

5.4.2 Ratkaisu käytännössä

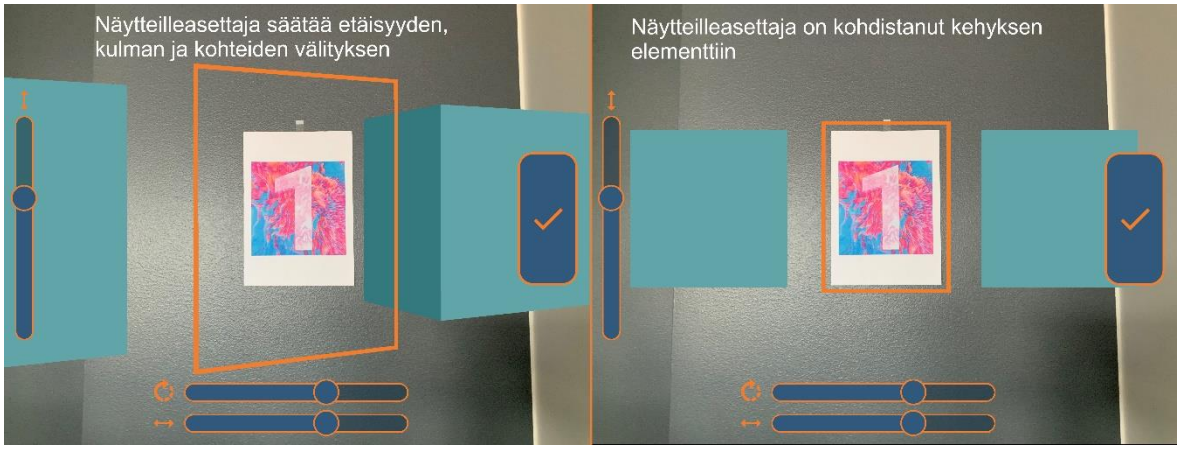
Aluksi ratkaisu vaikutti testeissä toimivalta ja vain pientä satunnaista värinää oli havaittavissa. Varsinaisessa näyttelytilassa kuitenkin huomattiin, että sensorivirheet olivat paljon huomattavampia ja värinä voimakkaampaa. Lisäksi kohteet lähtivät herkästi liikkumaan ympäri tilaa, kun kuva katosi näkyvistä tai sen näkyvyys oli vain osittaista. Vaikka näyttelytilan kuva olikin kooltaan neliömetrin suuruinen, iPadien etäisyys siihen ja tilan hämärä valaistus teki ratkaisusta käyttökelvottoman. Yritimme ratkaista tätä kokeilemalla ARFoundationin sijasta kahta maksullista lisätyn todellisuuden ohjelmistokehystä, Vuforiaa sekä MAXSTia. Testasimme myös monia eri kuvia ja kuvioita, ja lopulta MASXTilla saimmekin kohteet pysymään paikoillaan ja ilman värinää. Siitä huolimatta tilan hämäryys sekä etäisyys kuvaan tuotti yhä liikaa ongelmia kuvan tunnistuksessa. Koska tilaaja ei halunnut lisätä näyttelytilan valaistusta, näyttelytilaan ei ollut enää mahdollista saada suurempaa kuvaa, eikä iPadeja lähemmäksi sijoitettava kuvakaan ollut mahdollinen, jouduttiin jälleen etsimään uutta ratkaisua.

5.5 Device tracking -ratkaisu

Tässä vaiheessa oli käynyt selväksi, etteivät ympäristön tunnistamiseen turvautuvat tekniikat ole käyttökelpoisia näyttelytilassa ja pahimmillaan tuottavat vain lisää ongelmia. Lisäksi projektista vastaavalla yrityksellä oli ennestään toimivaksi todettu sovellus, joka oli toteutettu device tracking -ratkaisun varaan. Näistä syistä päädyttiin turvautua pelkästään device tracking -toiminnallisuuteen ja luottaa sen riittävyyteen. Koska device tracking seuraa vain ja ainoastaan laitteen sijaintia ja suuntaa, eikä näin ollen ole tietoinen ympäristöstä, täytyi kohteiden asettelu luoda lähestulkoon tyhjästä.

Ratkaisu toteutettiin niin, että kohteiden asettelun alussa niiden sijainti olisi sidottuna suoraan laitteen kameran eteen. Kohteiden asettelua helpottamaan lisättiin kohdistimeksi kehys, joka kohdistetaan näyttelytilaan sijoitetun kohdistuselementin ympärille. Kohdistamisen helpottamiseksi lisättiin säätimet kohteiden etäisyyden, välityksen ja rotaation säätämiseksi. Näytteilleasettajan tehtävänä olisi näitä säätimiä käyttämällä asettaa kehys näyttelytilan kohdistuselementin ympärille, jolloin kohteiden sijainti ja rotaatio suhteessa laitteeseen tallentuu. Tätä näytteilleasettajan kohdistustehtävää on havainnollistettu kuvassa 18. Näyttelyn aluksi sovellus antaa vastaavanlaisen kohdistustehtävän näyttelyssä kävijälle. Kuvassa 19 havainnollistetaan, kuinka kävijän tehtävänä on kohdistaa ja lukita laitteen ruudussa näkyvä kehys näyttelytilan kohdistuselementtiin. Kävijälle näkyvä kehys on asetettu suhteessa laitteeseen näytteilleasettajan tallentamien asetusten mukaisesti. Kävijän

lukitessa kehyksen kohdilleen, kehys katoaa ja kohteet tulevat esiin asettuen virtuaaliseen tilaan ankkurin tavoin, eivätkä ne ole enää sidottuna laitteen kameran eteen.



Kuva 18. Näytteilleasettajan kehyksen kohdistaminen

Kuva 19. Kävijän kohdistustehtävä

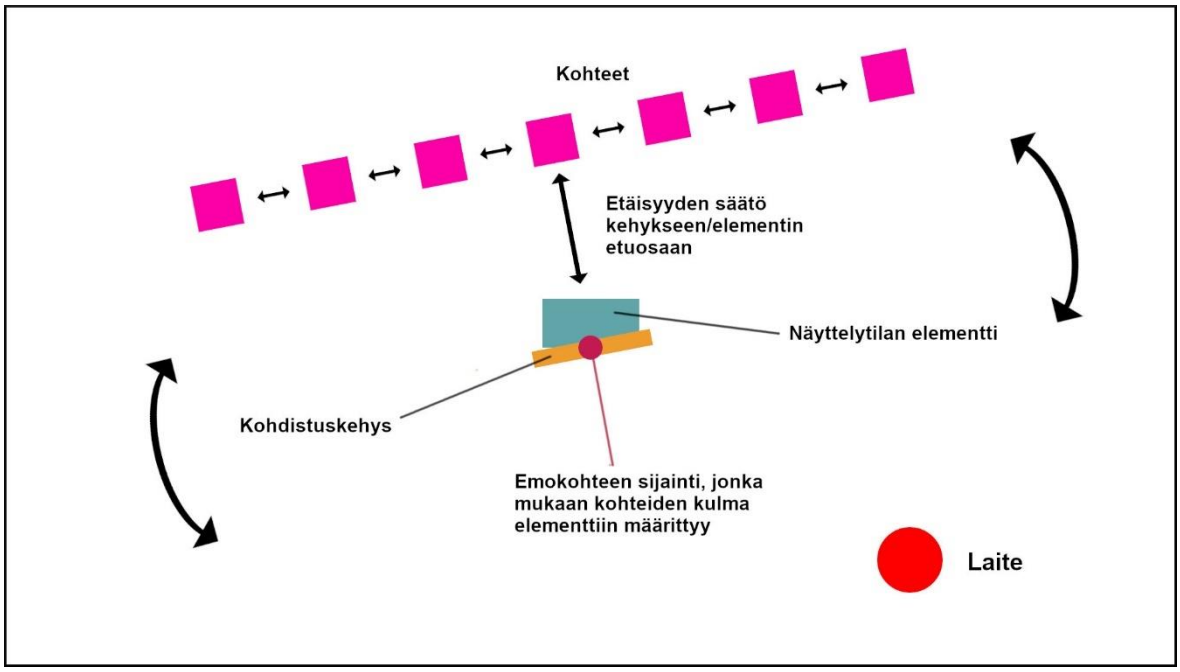
Kuten image tracking -ratkaisun kanssa oli tärkeää tietää kuvan mittasuhteet ja mille korkeudelle kuva näyttelytilassa sijoittuu, myös tässä ratkaisussa tarvittiin tiedot näyttelytilan kohdistuselementin korkeuden sijainnista ja mitoista. Näillä tiedoilla määritettiin virtuaalisten kohteiden skaala ja korkeus suhteessa kohdistuselementtiin ja kohteiden varjojen sijainti lattian tasolle.

5.5.1 Device tracking -toiminnallisuuden toteutus

Device tracking -toiminnallisuuden pohja luotiin jo ARFoundationin käyttöönoton yhteydessä. Sovellus on jo saatu seuraamaan laitteen asentodataa, mutta sitä hyödyntävä varsinainen kohteiden asettelulogiikka täytyy luoda lähestulkoon tyhjästä. Koska aikaisemmin oli jo ohjelmoitu muiden ratkaisujen vaatimaa logiikkaa, pystyttiin niitä hyödyntämään ainakin osittain tai soveltamalla. Plane detection -ratkaisusta saatiin hyödynnettyä placementPosen osuus. Sitä sovellettiin niin, että säteensuuntauksella saadun sijoituspisteen sijasta valittiin piste reilun kahdeksan metrin etäisyydeltä laitteen edestä pienestä alaviistosta. Tähän placementPosen sijaintiin asetettiin pystysuora kehys, jonka avulla virtuaaliset kohteet saataisiin kohdistettua näyttelytilan elementtiin.

Kohteiden kanssa päädyttiin käyttämään samantyyppistä ratkaisua kuin kuvantunnistuksen kanssa. Luotiin samanlainen hierarkkinen rakenne, jossa emokohteeksi otettiin tyhjä peliobjekti, jonka alle asetettiin toinen tyhjä peliobjekti etäisyyden ja rotaation säätämiseen. Tämän alle asetettiin seitsemän tyhjää peliobjektia välityksen säätämistä varten ja niistä kuhunkin liitettiin 3D-malli ja varjo.

Image tracking -ratkaisun avulla kohteet asettuvat automaattisesti tunnistetun kuvan suuntaisesti. Device tracking -ratkaisussa joudutaan kohteiden rotaatio säätämään manuaalisesti näyttelytilan elementin suuntaiseksi. Tätä varten lisättiin kolmas liukusäädin image tracking -ratkaisusta kopioitujen kahden muun säätimen lisäksi. Tällä kolmannella säätimellä säädettäisiin ylimmän tason emokohteen alla olevan tyhjän peliobjektin rotaatiota, jolloin kaikki sisältö saataisiin kerralla asetettua linjaan näyttelytilan elementin kanssa. Kuviossa 2 on havainnollistettu kohteiden asettelun säätömahdollisuuksia ylhäältäpäin katsottuna. Kun näytteilleasettaja on saanut tehtyä tarvittavat säädöt, tallentuu etäisyys, välitys sekä rotaatio laitteen muistiin. Nämä tiedot haetaan muistista ennen jokaista kohdistustehtävää ja kohteet konfiguroidaan niiden mukaan.



Kuvio 2. Kohteiden sijainnin määrittely

Kohteiden kohdistustehtävien aikana niiden sijainti määräytyy placementPosen mukaan. Tämä pitää kohteet lukittuna laitteen edessä, kun laitetta liikutetaan. Kun käyttäjä hyväksyy kohdistuksen, katkaistaan emokohteen linkki placementPoseen, jolloin kohteet jäävät virtuaaliseen tilaan niille sijoilleen. Device tracking jatkaa laitteen liikkeiden seurantaa fyysisessä ympäristössä ja käyttäytyy niiden mukaan virtuaalisessa tilassa, jossa kohteet pysyvät paikoillaan.

5.5.2 Ratkaisu käytännössä

Koska ratkaisu perustui pelkästään laitteen seurantaan ja sovelluksen tietoisuuden ympäristöstä näin ollessa olematonta, suhtauduttiin siihen varauksellisesti. Kaikesta huolimatta sensorivirheitä ei juurikaan ollut havaittavissa ja virtuaalikohteet pysyivät aloillansa paremmin kuin oli osattu odottaa. Näyttelytilan hämäryys ja etäisyys kohdistuselementtiin eivät tuottaneet ongelmia, sillä ratkaisu ei hyödynnä kameran tuottamaa kuvaa tunnistamiseen vaan pelkästään näyttämään ympäristön käyttäjälle.

Ratkaisussa oli kuitenkin yksi ongelmallinen asia. Sovellus itse ei kykene automaattisesti tekemään kohteiden kohdistamista näyttelytilan elementtiin ympäristön tunnistuksen puuttuessa, joten se on jätetty käyttäjien eli näytteilleasettajan ja kävijöiden tehtäväksi. Jos näytteilleasettaja tekee kohdistuksen väärin, tekee se samalla kävijöiden kohdistamistehtävästä täysin mahdottoman. Tämä on kuitenkin hyvin epätodennäköinen skenaario. Paljon todennäköisemmin ongelmia tulisi tuottamaan kävijöiden vastuulla oleva kohdistaminen. Vaikka

sovellus neuvookin kävijää kohdistuksen kanssa, ei kävijän ole pakko totella sitä. Kävijä pystyisi halutessaan kohdistamaan laitteen jonnekin muualle kuin näyttelytilan kohdistus-elementtiin, jolloin virtuaaliset kohteet asettuisivat tilaan väärin. Tällöin lisätty todellisuus ei pääse kunnolla toteutumaan. Käyttäjiin päädyttiin kuitenkin luottamaan ja ratkaisu otettiin käyttöön.

6 Johtopäätöksiä testatuista ratkaisuista

Nämä johtopäätökset perustuvat iPad Prolla saatuihin käytännön tuloksiin. Varsinkin mobiililaitteiden väliset laitteistolliset kuin ohjelmistopohjaiset erot voivat olla hyvinkin suuria ja vaikuttaa merkittävästi toiminnallisuuteen. iPad Pro kuuluu mobiililaitteiden tehokkaimpaan päähän. On siis mahdollista ja hyvinkin todennäköistä, että toisilla alustoilla tulokset olisivat näistä poikkeavia.

Plane detection -ratkaisu osoittautui kokonaisuudessaan vahvaksi ratkaisuksi. Sen toteuttaminen on verrattain yksinkertaista ja se on toiminnallisesti erittäin luotettava. Alkuperäisissä testeissä tasojen tunnistus toimi varsin nopeasti ja kohteiden asettelu tilaan onnistui hyvin tarkasti. Lisäksi sensorivirheet olivat käytännössä olemattomia. Ratkaisun ongelmat tulivat esiin vasta kun päästiin varsinaiseen näyttelytilaan testaamaan. Näyttelytila oli hämärä ja tunnistettavat tasot vaikeasti laitteen kameralla erotettavissa. Näitä olosuhteita voidaan kuvailla ”ääriolosuhteiksi”. Ääriolosuhteissakin ratkaisu oli vielä jossain määrin toimiva. Jos lisätyn todellisuuden sovellusta aiotaan käyttää tilassa, jossa ei olla tekemisissä ääriolosuhteiden kanssa niin plane detection on vahva ja luotettava valinta.

Object tracking osoittautui liikaa tarkkuutta vaativaksi tekniikaksi, sillä testeissä esineiden tunnistus onnistui vain, kun katselukulma, valaistus ja ympäristö täsmäsivät mahdollisimman tarkasti lähdekirjaston skannauksen kanssa. Lisäksi tunnistettavan esineen täytyi olla sellainen, että sen piirteet olivat selkeästi erotettavissa. Ellei nimenomaan haluta, että esine tunnistetaan vain tarkasti määritetyissä olosuhteissa, niin ratkaisu on ainakin toistaiseksi lähestulkoon käyttökelvoton.

Toteutuksen helppoudessa image tracking on ylivoimainen muihin ratkaisuihin verrattuna. Sovelluksen kuvantunnistus saadaan toimintakuntoon todella nopeasti ja virtuaalisten kohteiden sijainnit saadaan määriteltyä tarkasti suhteessa kuviin. Kohteita ei tarvitse sovelluksessa alkaa sijoittelemaan vaan niiden asettuminen tilaan määräytyy sen mukaan mihin kuvat on fyysisessä tilassa aseteltu. Image tracking -toiminnallisuuteen perustuva ratkaisu on melko luotettava, mutta senkin kanssa on otettava olosuhteet huomioon. Testeissä huomattiin, että rajoittavia tekijöitä sen käytölle on kameran tarkkuus, tunnistettavan kuvan etäisyys, yksityiskohdat ja koko sekä ympäristön valaistus. Rajoittavien tekijöiden ilmaantuessa image tracking oli testeissä todella epävakaata. Jos käyttökohteessa tiedetään mihin virtuaaliset kohteet tulee sijoittumaan eikä kuvan tunnistukselle ole rajoittavia tekijöitä niin image tracking on järkevin valinta sovelluksen toteutukseen. Kohteiden sijaintia saadaan myös helposti vaihdettua siirtämällä kuvan tai kuvien sijaintia.

Device tracking on enemmän pohjaratkaisu, jonka päälle lisätään ympäristöä tunnistavia tekniikoita tukemaan ja keräämään tietoa ympäristöstä. Yksinään käytettäessä sen toiminta voi olla kyseenalaista ja työläs toteuttaa, sillä ratkaisu täytyy käytännössä luoda tyhjästä laitteen relatiivisen sijainti- ja suuntadatan pohjalta. Päädyimme sovelluksessa tukeutumaan pelkän device tracking -ratkaisun varaan, mutta se ei ole suositeltavaa, ellei sovelluksen toteutus onnistu jotain muuta tekniikkaa tai tekniikoita hyödyntävällä ratkaisulla.

Ratkaisuja ei testattu voimakkaasti valaistussa ympäristössä, joten ei voida varmuudella sanoa miten se vaikuttaisi ratkaisujen toiminnallisuuteen. On kuitenkin hyvin todennäköistä, että voimakas tai kirkas valaistus aiheuttaisi ongelmia esimerkiksi image tracking -ratkaisun kanssa. Liian voimakas valaistus aiheuttaisi kuvan värien vääristymistä ja heijastuksia, jolloin sen tunnistus vaikeutuisi.

7 Yhteenveto

7.1 Tavoitteeseen pääsy

Projektin alussa henkilökohtaista kokemusta lisätyn todellisuuden käytöstä sovelluskehityksessä ei ollut. Unity-pelimoottori ja sen yhteydessä käytettävä C#-ohjelmointikieli olivat kuitenkin jo entuudestaan tuttuja koulun kursseilta ja aikaisemmista projekteista. Tältä pohjalta sekä tilaajan toiveesta käyttää ilmaisia työkaluja, päädyttiin valitsemaan Unityn oma lisätyn todellisuuden ohjelmistokehys, ARFoundation.

Sovelluksen 3D-mallit, graafinen ulkoasu ja käyttöliittymä tulivat projektia vetävältä yritykseltä. Tehtäväksi jäi varsinainen sovelluksen toteutus, toiminnallisuuden suunnittelu ja toteutus sekä ohjelmointi.

Sovellus valmistui alkuperäisten kriteerien mukaisesti ajallaan, mutta tilaaja halusi vielä tehdä muutoksia. Sovelluksen lopullinen valmistuminen lykkääntyi vielä muutamaan otteeseen teknisten rajoitusten takia sekä tilaajan ja projektista vastanneen yrityksen välisten informaatiokatkosten vuoksi. Kaikesta huolimatta sovellus saatiin tilaajalle sopivasti käyttövalmiiksi koronaepidemiasta johtuvien rajoitusten heltyessä ja näyttelyn avautuessa.

7.2 Ongelmat

Projektin edetessä ilmeni, ettei projektista vastanneen yrityksen ja tilaajan välinen viestintä ollut toiminut odotusten mukaisesti ja jouduimme useampaan otteeseen tekemään asioita uusiksi. Alun perin vastaavan yrityksen puolelta toimeksianto oli luoda kaksi erillistä sovellusta, joissa olisi ollut erilliset sisällöt. Ensimmäisen sovelluksen jo valmistuttua ja toisen ollessa pitkällä kehityksessä, ilmeni ettei tilaaja ollut asiaa hyväksynyt. Lisäksi alkuperäinen plane detection -toiminnallisuuteen perustuva yhden sovelluksen ratkaisu olisi vaatinut tilaajan puolelta kohteiden asettelua virtuaaliseen ympäristöön vähintään jokaisen näyttelypäivän aamuna. Tämä johtui siitä, ettei tunnistettua ympäristöä ollut teknisesti mahdollista tallentaa. Tätäkään ei ollut hyväksytetty tilaajalla, ja kuten myöhemmin kävi ilmi, ei se myöskään tilaajalle sopinut. Huomautuksena voidaan mainita, että tunnistetun ympäristön tallennus on sittemmin tullut mahdolliseksi iOS-laitteille ARKit-toiminnallisuutena. Tämän ansiosta plane detection -toiminnallisuuteen perustuva ratkaisu saattaisi olla nykyään täysin toteutettavissa tilaajan kriteerien mukaan. Lisäksi ARKitin avulla pitäisi onnistua tunnistetun ympäristön jakaminen useamman laitteen välillä. Tällä saataisiin esimerkiksi vähennettyä näytteilleasettajan työmäärää, jos käytössä on useampi laite ja kohteita tarvitsee erikseen asetella. Näiden toiminnallisuuksien luotettavuutta olisi hyvä tutkia tulevaisuudessa.

Näyttelytilan olosuhteet sekä laitteiston ja teknologian rajoitukset toivat myös omat ongelmansa mukanaan. Näyttelytilan valaistus haluttiin pitää hämäränä ja kohteiden etäisyys iPadeihin suurena. iPadien kamerat eivät erota ympäristöä tarpeeksi hyvin hämärissä olosuhteissa, etenkään pitkiltä etäisyyksiltä, joten jouduttiin soveltamaan ja kokeilemaan useita eri ratkaisuja. Lopulta päädyttiin kokonaan sulkemaan erilaiset ympäristön tunnistusta vaativat tekniikat pois, koska totesimme ne täysin toimintakelvottomiksi näyttelytilassa. Keinojen loppuessa kesken, uskaltauduttiin tukeutumaan täysin laitteiden eli iPadien omien sijainti- ja suuntasensorien dataan. Onneksi osoittautui, että kyseinen ratkaisu toimikin paremmin kuin osattiin odottaa. Ratkaisussa on kuitenkin yksi sudenkuoppa. Koska sovellus ei ole tietoinen ympäristöstään ja kohdistaminen on jätetty kävijän tehtäväksi, kävijä voi vastoin sovelluksen ohjeita kohdistaa kohteet jonnekin muualle kuin näyttelytilan kohdistuselementtiin. Tällöin kohteet asettuvat tilaan väärin ja lisätyn todellisuuden illuusio rikkoutuu.

7.3 Hyödyt

Sovelluksen perimmäisenä tarkoituksena oli saada tilaajalle näyttelyyn kohde, joka visualisoi informatiivista dataa uudella, mielenkiintoisella sekä kävijää osallistavalla tavalla, mutta selkeästi ja ymmärrettävästi. Sovellusratkaisulla saatiin aikaiseksi yksityiskohtaisia, animoituja virtuaalimalleja, jotka eivät vie tilaa todellisesta näyttelytilasta ja esittävät datan ymmärrettävällä tavalla.

Toinen päämäärä oli, että näyttely saataisiin tarvittaessa siirrettyä täysin eri sijaintiin, vaikka toiselle puolelle maapalloa. Tämän suhteen sovellus on erittäin onnistunut, vaikka kohteiden asettelussa päädyttiin kompromissiin. Riittää, että kohteessa näytteilleasettaja asettaa näyttelytilaan elementin, johon sovelluksen sisältö kohdistetaan. Tätä päämäärää tukemaan tehtiin myös mahdolliseksi lisätä, muokata ja poistaa kieliä.

Lopulta saatiin tulevaisuutta ajatellen paljon tärkeää tietoa lisätyn todellisuuden erilaisista toteutustavoista, kuinka olosuhteet ja ympäristö vaikuttavat niihin ja niiden vaatimuksista. Saatua tietoa hyödyntämällä pystytään valitsemaan toimiva ratkaisu käyttökohteen perusteella.

Lähteet

Apple Inc. 2020a. Xcode Help: Welcome to Xcode. Viitattu 4.8.2020. Saatavissa https://help.apple.com/xcode/mac/current/#/devc8c2a6be1

Apple Inc. 2020b. Documentation: Scanning and Detecting 3D Objects. Viitattu 12.9.2020. Saatavissa https://developer.apple.com/documentation/arkit/scanning_and_detecting_3d_objects

Apple Inc. 2020c. Documentation: ARBodyAnchor. Viitattu 2.8.2020. Saatavissa https://developer.apple.com/documentation/arkit/arbodyanchor

Apple Inc. 2020d. Documentation: ARBody2D. Viitattu 2.8.2020. Saatavissa https://developer.apple.com/documentation/arkit/arbody2d

Apple Inc. 2020e. Documentation: Occluding Virtual Content with People. Viitattu 13.9.2020. Saatavissa https://developer.apple.com/documentation/arkit/occluding_virtual_content_with_people

Arnaldi, B., Guitton, P. & Moreau, G. 2018. Virtual Reality and Augmented Reality: Myths and Realities. Hoboken, New Jersey: John Wiley & Sons, Inc.

Craig, AB. 2013. Understanding Augmented Reality: Concepts and Applications. Waltham, Massachusetts: Elsevier Inc.

Kipper, G. & Rampolla, J. 2012. Augmented Reality: An Emerging Technologies Guide to AR. Rockland, Massachusetts: Syngress.

Microsoft 2019. Documentation: Visual Studio for Mac tour. Viitattu 4.8.2020. Saatavissa https://docs.microsoft.com/en-us/visualstudio/mac/ide-tour?view=vsmac-2019

MonoDevelop Project 2020a. Documentation. Viitattu 8.9.2020. Saatavissa https://www.monodevelop.com/documentation/

MonoDevelop Project 2020b. Documentation: Feature List. Viitattu 8.9.2020. Saatavissa https://www.monodevelop.com/documentation/feature-list/

Unity Technologies 2020a. Unity: Game engines—how do they work? Viitattu 1.8.2020. Saatavissa https://unity3d.com/what-is-a-game-engine

Unity Technologies 2020b. Unity Manual: Creating and Using Scripts. Viitattu 1.8.2020. Saatavissa https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

Unity Technologies 2020c. Unity Manual: Integrated development environment (IDE) support. Viitattu 1.8.2020. Saatavissa https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html

Unity Technologies 2020d. Unity Manual: About AR Foundation. Viitattu 1.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/index.html

Unity Technologies 2020e. Unity Manual: Using ARKit XR Plugin. Viitattu 1.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arkit@4.0/manual/index.html

Unity Technologies 2020f. Unity Manual: AR plane manager. Viitattu 2.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/plane-manager.html

Unity Technologies 2020g. Unity Manual: AR point cloud manager. Viitattu 2.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/point-cloud-manager.html

Unity Technologies 2020h. Unity Manual: AR anchor manager. Viitattu 2.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/anchor-manager.html

Unity Technologies 2020i. Unity Manual: AR face manager. Viitattu 2.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/face-manager.html

Unity Technologies 2020j. Unity Scripting API: Class ARFace. Viitattu 2.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/api/UnityEngine.XR.ARFoundation.ARFace.html

Unity Technologies 2020k. Unity Manual: XR image tracking subsystem. Viitattu 8.9.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.0/manual/image-tracking.html

Unity Technologies 2020l. Unity Manual: AR tracked image manager. Viitattu 2.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/tracked-image-manager.html

Unity Technologies 2020m. Unity Manual: AR tracked object manager. Viitattu 2.8.2020. Saatavissa https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/tracked-object-manager.html