SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

ARCORE-SOVELLUS YMPÄRISTÖN VISUALISOINTIIN

TE-KIJÄ/T: Tuomas Honkonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala			
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma			
Työn tekijä(t) Tuomas Honkonen			
Työn nimi ARCore-sovellus ympäristön visualisointiin			
Päiväys	13.Toukokuu 2018	Sivumäärä/Liitteet	29
Ohjaaja(t) Mikko Pääkkönen, TKI-asiantuntija, Mikko Laasanen, TKI-asiantuntija			
Toimeksiantaja/Yhteistyökumppani(t) 3DTalo			

Tiivistelmä

Opinnäytteen varsinaisen työ osion tarkoituksena oli kehittää 3DTalolle demosovellus ympäristön visualisoimisesta, jonka avulla he voisivat myydä uudenlaisia lisätyn todellisuuden palveluita. Työn käyttötarkoituksesta johtuen varsinainen työ on kuitenkin salattava suurimmilta osin. Tästä johtuen opinnäytetyön raportti osio keskittyy työssä keskeiseen tekniikkaan ARCoreen.

Opinnäytetyön teko alkoi opiskelujaksolla, jonka aikana perehdyttiin ARCoreen tekniikkana. Tämän jälkeen varsinaisen työn kehittäminen päästiin aloittamaan. Työ tehtiin käyttäen Unity pelimoottoria kehitysalustana. Ohjelmointi demosovellukseen hoidettiin C# -ohjelmointi kielellä, jota käytettiin kommunikoimaan ARCoren kanssa. Opinnäytetyön raportti on kirjoitettu pääasiassa perustuen opiskelujaksoon ja tutkimustyöhön ARCoreen liittyen.

Varsinaisen demosovelluksen teko 3DTalolle onnistui, vaikkakin tehdessä ilmeni muutamia ongelmia. Yksi suurimmista ongelmista oli sovelluksen tarkkuuden vaihtelu käyttökerran pidetessä. Sovellusta on mahdollista lähteä vielä joskus kehittämään eteenpäin, varsinkin jos Google tekee merkittäviä muutoksia tekniikkaan, jotka saattaisivat ratkaista joitakin ongelma kohtia.

Avainsanat
ARCore, lisätty todellisuus.

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Tuomas Honkonen			
Title of Thesis ARCore application for surroundings visualization			
Date	13 May 2018	Pages/Appendices	29
Supervisor(s) Mr Mikko Pääkkönen, RDI Specialist, Mr Mikko Laasanen, RDI Specialist			
Client Organisation /Partners 3DTalo			

Abstract

The goal of this Thesis was to develop a demo application about visualizing surroundings for 3DTalo. The purpose was that they could use this demo to sell new type of augmented reality services to their clients. Because of the type of usage that the demo would be in, most of it is confidential. For this reason, the report part of the Thesis focuses on the key technique of the demo ARCore.

The work was started with a practical training, where the needed knowledge of the technology was gained. After this the development of the demo started. The demo was made by using a Unity game engine as a development platform. Programming was done with C#, which was used to communicate with ARCore. The report of the Thesis was written mostly based on what was learned during the practical training and research on ARCore.

As a result of this thesis, a demo application was created for 3DTalo, even though few problems came up in development. One of the biggest problems was that the application started to lose accuracy over time. There is still a possibility to develop the demo even further, especially if Google makes some major changes in the technique that might solve some of the problems there were.

Keywords
ARCore, augmented reality.

ESIPUHE

Opinnäytettä varten tehty varsinainen sovellus on tehty 3DTalolle kaupallista käyttöä varten. Tästä johtuen varsinaisen työn koodia ei voida näyttää eikä sitä voida tarkemmin avata kuin mitä raportin loppupuolella on tehty. Näin ollen raportti käy läpi sovelluksen keskeisintä tekniikkaa ARCorea. Raporttia kirjoitettaessa on tehty oletus, että lukija osaa asentaa Unityn ja tietää joitakin Unityn käytön perusteita.

SISÄLTÖ

1 JOHDANTO

1.1 Mitä ovat VR ja AR?

VR ja AR ovat molemmat lyhenteitä kahdesta suuresta tulevaisuuden teknologiasta (VR = Virtual reality & AR = Augmented reality). Molemmat ovat suhteellisen uusia teknologioita ja niitä kehitetään vielä paljon. Lisäksi niille kehitetään koko ajan uusia tekniikoita, joilla niitä pyritään hyödyntämään.

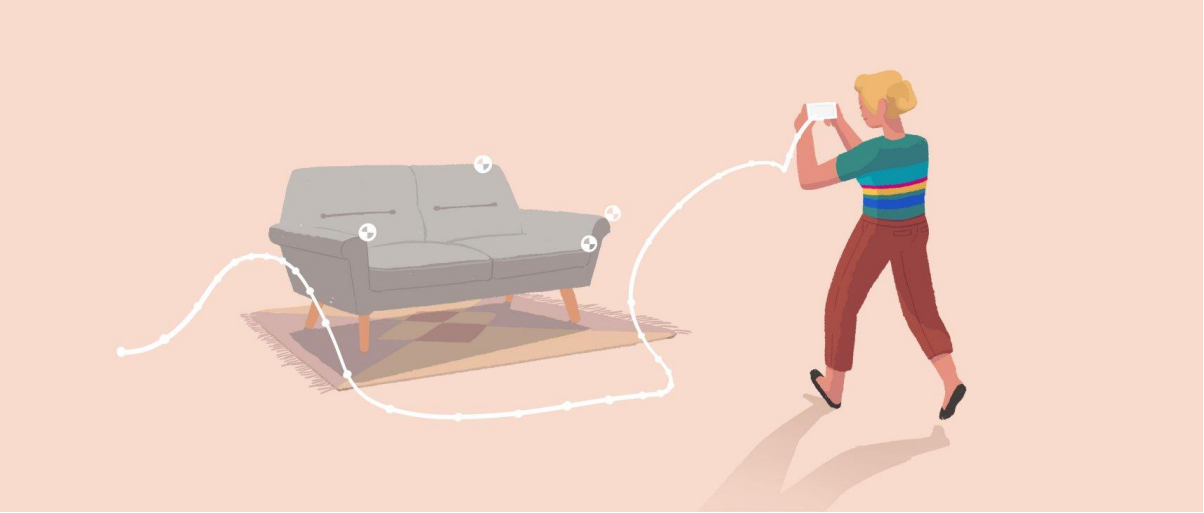
Virtual reality eli virtuaalitodellisuus on sitä, kun käyttäjän näkökenttään luodaan tietokoneella näkymä ja siten tunne toisesta kolmiulotteisesta maailmasta. Virtuaalitodellisuus yleensä luodaan käyttäen virtuaalilaseja, joita on tällä hetkellä myynnissä muutamia merkkejä. Suurin osa niistä on vielä toistaiseksi suhteellisen hintavia ja siten liian kalliita suurelle yleisölle. Useissa virtuaalitodellisuuspaketeissa on myös käsiohjaimet, jotka mahdollistavat vuorovaikutuksen käyttäjän ja virtuaalitodellisuuden välillä.

Augmented reality eli lisätty todellisuus on sitä, kun käyttäjän esimerkiksi kännykän kameran läpi nähtävään todellisuuteen lisätään jotakin, mitä siinä ei alkuperin ole. Lisättyä todellisuutta käytetään yleensä kännyköillä tai tablettietokoneilla, koska ne ovat helposti liikuteltavissa ja useimmat ihmiset omistavat joko toisen tai molemmat laitteet jo valmiiksi. Huomioitavaa on kuitenkin, että puhelimen tai tabletin tulee olla suhteellisen hyvin ajan tasalla, koska lisätyn todellisuuden tekniikat vaativat yleensä melko uuden Android tai iOS version sekä vasta viime aikoina kännyköhin tullutta teknologiaa.

1.2 Mikä on ARCore?

Googlen omien kehittäjäsivujen (Google Developers, 2018) mukaan ARCore on alusta lisätyn todellisuuden sovellusten rakentamista varten Androidille. ARCore on uusin keksintö lisätyn todellisuuden maailmassa ja luo uusia käyttömahdollisuuksia lisätylle todellisuudelle. ARCoren toiminta perustuu kolmeen keskeiseen teknologiaan: liikeentunnistukseen (ks. Kuva 1), ymmärrykseen ympäristöstä ja valon määrän arviointiin.

ARCore pyrkii luomaan käsityksen ympäristöstään samalla, kun se seuraa omaa liikettään siinä. Tämä mahdollistaa erilaisten asioiden sijoittamisen ”oikeaan maailmaan”. ARCore myös pystyy ”ankkuroimaan” käyttäjän tai sovelluksen sijoittaman objektin ympäristössä oleviin pisteisiin tai tasoihin. Tällöin objektit pysyvät paikallaan, vaikka käyttäjä liikkuisi ympärillä. Koska ARCore käyttää pisteitä luodakseen ymmärryksen ympäristön tasoista ja omista liikkeistään, se tarvitsee ympäristöstä selkeitä kiintopisteitä. Siksi mikäli ympäristöstä ei löydy riittävästi tekstuureja, joiden avulla ARCore saa luotua kyseisiä kiinnepisteitä, voi ARCoren tarkkuudessa olla ongelmia vielä teknologian nykyisessä tilassa. (Google Developers, 2018.)



Kuva 1. Googlen havainnollistuskuva ARCoren liikkeen tunnistuksesta (Google Developers, 2018)

1.2.1 ARCore vs Vuforia

Toisin kuin ARCore Vuforia on ollut työkaluna AR kehittäjillä jo huomattavasti pitemmän aikaa ja on jo menossa versiossa v7.0.50 (tarkistettu 12.03.2018). Tästä johtuen määrä laitteita, jotka pystyvät Vuforiaa käyttämään on huomattavasti suurempi kuin mitä ARCorella. Vuforia toimii suurimmalla osalla Android laitteista, sekä myös iOS sekä Windows 10 laitteilla, mikä laajentaa käyttäjäkuntaa entisestään ARCoreen verrattuna. Vuforia myös tukee muutamia AR -lasisettejä.

Vuforian toiminta perustuu pääasiassa erilaisten esineiden/kuvien tunnistamiseen. Esimerkiksi jos oletetaan että sanomalehdessä olisi Vuforia sovellukselle ennalta määritelty kuva. Pystyy sovellus tunnistamaan lehdessä olevan kuvan, jonka jälkeen pystyttäisiin avaamaan esimerkiksi aiheeseen liittyvä YouTube -linkki käyttäjälle. Samanlaista esineiden ja kuvien tunnistusta ei ARCoressa ole ainakaan toistaiseksi eikä vielä ole tietoa, onko sellaista tu-

lossa jatkossakaan. Koska Vuforian toiminta perustuu pääasiallisesti asioiden tunnistamiseen, on sovelluksen käyttöympäristön valonmäärä huomattavasti suuremmassa merkityksessä kuin mitä ARCorella. Toki mikäli ARCorella tehty sovellus vaatii tunnistettuja pintoja niin ympäristön valoisuudella on merkitystä, mutta se ei ole yhtä isossa roolissa kuin mitä Vuforialla. Vaikkakin teoriassa ARCoren tulisi toimia säkkipimeässä, ennalta asetettujen esineiden osalta, kärsii sen toiminta, jos sovellus ei löydä itselleen ympäristöstä kiinnepisteitä. Kaiken kaikkiaan koska ARCore ja Vuforia toimivat eri periaatteilla soveltuvat ne hieman erilaisiin sovelluksiin. Näin ollen, ellei ARCoreen lisätä esineiden ja kuvien tunnistusta, ei ole todennäköistä, että Vuforia olisi katoamassa tekniikkana minnekään varsinkaan ihan lähitulevaisuudessa ainakaan ARCoren toimesta. Tähän mielipiteeseen päädytään myös, kun huomioidaan Vuforiaa tukevien laitteiden huomattavan suuri määrä. (Vuforia, 2018.) (Google Developers, 2018.)

1.2.2 ARCore vs ARKit

ARKit on iOS 11 esittelemä AR -teknologia. ARKitin käyttämä virtual-inertial odometry lyhyesti VIO tekniikka pyrkii luomaan yhteyden oikean ja virtuaalimaailman välille. VIO seuraa maailmaa ympärillään ja yhdistää kameran sensoreiden dataa CoreMotion datan kanssa. CoreMotion antaa iOS laitteelle dataa liittyen laitteen liikkeisiin ja ympäristöön. ARCoren vuorostaan käyttää concurrent odometry and mapping lyhyesti COM tekniikkaa liikkumisensa seuraamiseen. Tämän tarkoituksena on tunnistaa yksityiskohtia kameran kuvasta ja niiden avulla sijoittaa laite oikeaan paikkaan ympäröivään mailmaan nähden. Molemmat tekniikat myös pyrkivät arvioimaan valonmäärää, jotta virtuaaliobjekteille tulisi ympäristön mukaisesti oikean verran valoa.

Yksi etu ARKitillä on verrattuna ARCoreen, nimittäin suurin osa iOS käyttäjistä käyttää jo viimeisintä versiota käyttöjärjestelmästä ja tämä trendi luultavasti jatkuu myös tulevaisuudessa, kun taas vasta pieni osa Android käyttäjistä omaa laitteen, joka käyttää ARCorea tukevaa Android versiota (7.0 ->). (HiddenBrains, 2017.)

1.2.3 ARCore v1.0

ARCore v1.0 myötä Google siirsi ARCoren ennakko (preview) tilasta viralliseen julkaistuun versinoon. Julkaisun myötä sovellusten kehittäjät voivat julkaista omia töitään Googlen Play Storeen. Julkaisu toi mukanaan myös tuen muutamille uusille laitteille ja siten suuremman asiakaskunnan ARCorelle. Google myös kertoi tekevänsä yhteistyötä monien puhelinvalmistajien kanssa, kuten esimerkiksi Samsung, Huawei, Motorola, jotta heidän tuleviin malleihin saataisiin tuki ARCorelle. Tämä tuo tulevaisuudessa ison määrän uusia asiakkaita ARCore

sovelluksille, varsinkin kun laitteiden hinnat alkavat laskea. Kuitenkin Android käyttöliittymien versioiden hajanaisuus aiheuttaa ongelmia vielä pitkälle tulevaisuuten ARCore sovellusten saamiselle suuren yleisön haltuun. ARCore 1.0 toi mukanaan paremman käsityksen ympäristöstä ja antaa mahdollisuuden käyttäjälle lisätä objekteja pinnoille, joissa on tekstuuria, kuten esimerkiksi julisteet, huonekalut ja lelut, mikä onnistui ennen vain tasojen kanssa. (Gosalia, Announcing ARCore 1.0 and new updates to Google Lens, 2018.)

2 ARCORE UNITYSSÄ

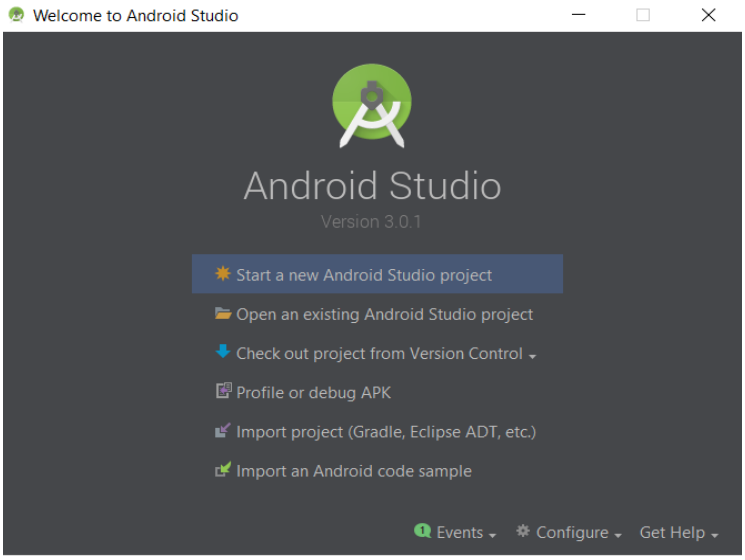
Jotta mahdollisimman moni kehittäjä pääsisi työskentelemään ARCoren kanssa, on Google julkaissut ARCore SDK:n usealle eri kehitysalustalle, kuten myös Unitylle. Muita kehitys alustoja ovat muun muassa Android Studio ja Unreal Engine. Pystyäkseen rakentamaan ARCore sovelluksen Unityssä, tulee kehittäjän kuitenkin tehdä useita vaiheita, jotka tässä osiossa käydään läpi. Tulee myös huomioida, että Unitystä vaaditaa suhteellisen uusi versio. Alhaisin toimiva Unity versio on ”Unity 2017.3.0f2”. Mikäli Unityä ei löydy kehityskoneelta tai versio on edellä mainittua vanhempi, tulee käyttäjän hankkia viimeisin versio Unityn omilta nettisivuilta (unity3d.com). Sopivan version pystyy lataamaan myös Unityn arkistoista (unity3d.com/get-unity/download/archive). (Google Developers, 2018.)

2.1 Android SDK ja sen asennus

Yksi tärkeimmistä asioista, että Unity pystyy rakentamaan ARCore sovelluksen tai minkä tahansa muunkaan Android laitteelle tulevan sovelluksen on se, että oikean Android versio on asennettuna kehityskoneella. Kun lähtee työskentelemään uuden SDK:n tai ylipäätään itselleen uuden teknologian kanssa tulee ottaa selvää, mikäli kyseinen teknologia vaatii jotain tiettyä Android versiota. ARCoren tapauksessa alhaisin toimiva versio Androidista on 7.0.

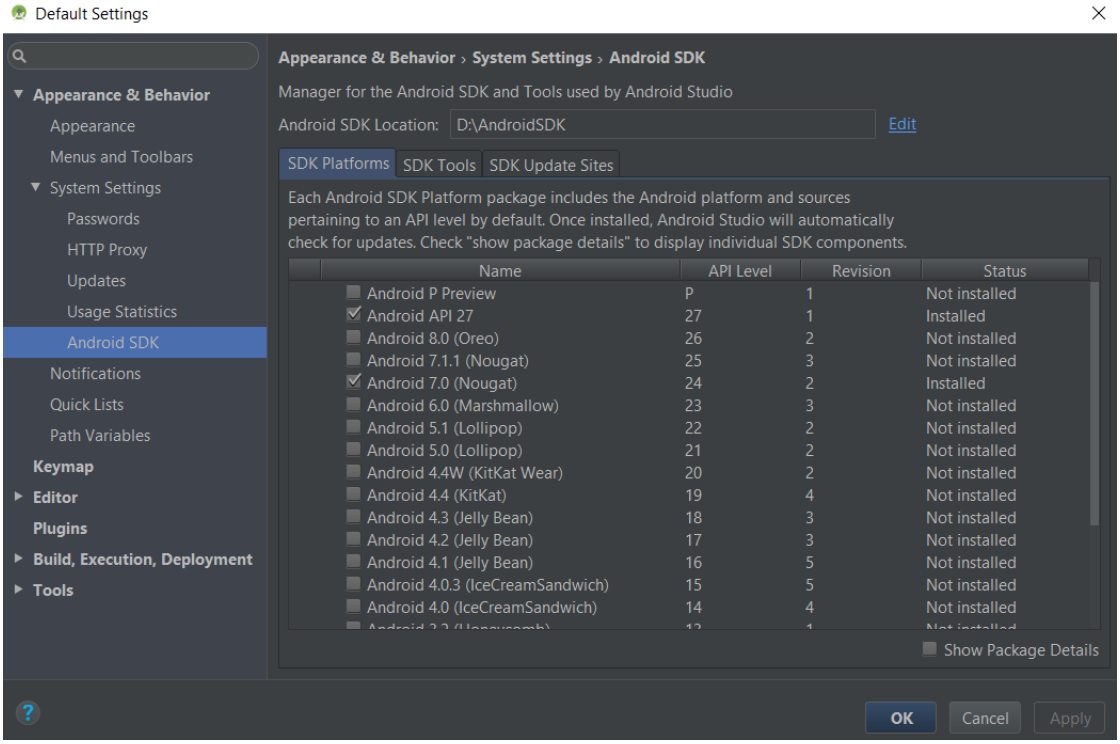
Oikean Android SDK:n asentaminen vaatii sen, että koneelta löytyy Andoid Studio, jonka kautta kehittäjä pääsee asentamaan uusia SDK-paketteja. Mikäli koneelta ei löydy Android Studiota pystyy sen asentamaan Androidin kehittäjäsivuilta (developer.android.com/studio/index.html) ilmaiseksi. Huomioitavaa on että, Android Studiota asentaessa on myös jo mahdollista asentaa haluamiaan SDK-paketteja. Muuten asentaminen menee aivan muiden ohjelmien asennusten kaavaa noudattaen, mutta loppupuolella asennusta ohjelma kysyy, mikäli käyttäjä haluaa asentaa tässä vaiheessa jo joitakin tiettyjä SDK-paketteja.

Mikäli Android Studion asennuksen yhteydessä ei tarvittavaa SDK-pakettia tullut asennettua, voidaan se asentaa jälkikäteen SDK Managerin avulla. SDK Managerin saadaan auki, kun ensin avataan Android studio ja aukeaa ”Welcome to Android Studio” -ikkuna (ks Kuva 2).



Kuva 2 ”Welcome to Android Studio” -ikkuna

”Welcome to Android Studio” -ikkunasta käyttäjä valitsee ”Configure”, josta avautuu valikko, jonka ylimmäisenä on ”SDK Manager”.



Kuva 3. Android Settings

Yllä olevan kuvan (Kuva 3) kaltainen ikkuna tulisi avautua, mikäli Android SDK kohta ei aukea automaattisesti löytyy se ”System Settings” -kohdan alta. Tästä näkymästä käyttäjä pystyy valitsemaan haluamansa SDK paketit ja sen jälkeen pitää valita Apply, minkä jälkeen sovellus hoitaa SDK-pakettien latauksen ja asentamisen tietokoneelle. Tämän jälkeen kyseisellä koneella pystyy kehittämään sovelluksia valituille Android versioille. Kannattaa myös huomioida, mihin kansioon Androidin versiot ja SDK toolsit on asennettuna, mikäli niihin

täytyy päästä käsiksi myöhemmässä vaiheessa. Yhden syyn tällaiselle tilanteelle käyn läpi myöhemmässä vaiheessa ”Omat haasteet ARCoren ja Unityn kanssa” -kohdassa.

2.2 Puhelimen valmistelu

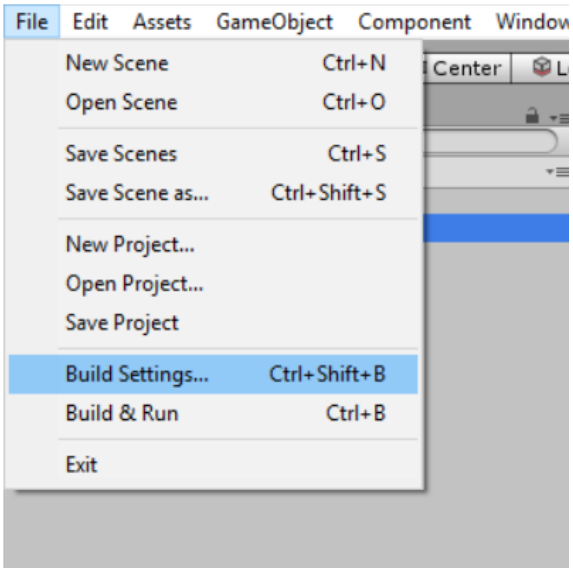
Rakentaakseen Android sovelluksen Unity vaatii avukseen Android laitteen. Laitteen tulee omata vähintään sama Android versio, jonka kehittäjä on valinnut sovelluksen minimi Android versioksi. Eli ARCoren tapauksessa laitteen tulee omata vähintään Android version 7.0. Jotta laitetta pystyisi käyttämään sovelluskehityksessä tulee siinä olla kehittäjä asetukset ja USB viankorjaus päällä. Mikäli kehittäjä asetukset eivät ole päällä, saa ne päälle menemällä laitteen asetuksiin, josta pohjalta löytyy ”Tietoja puhelimesta” -kohta. ”Tietoja puhelimesta”-kohdan pohjalta löytyy laitteen mallinumero. Mallinumeroa tulee painaa 7 kertaa, minkä jälkeen taaksepäin mentäessä pitäisi ”Kehittäjäasetukset”-osio tulla näkyviin. Kehittäjäasetuksissa kun menee vähän alaspäin virheenkorjaus kohtaan, löytyy sieltä USB viankorjaus, joka tulee laittaa päälle. Kun ARCore oli vielä Preview2 tilassa, nykyisen version 1.0 sijaan, tarvitsi kehittäjän vielä ladata Googlen ARCore sivuilta Preview2 APK, joka antoi laitteelle tuen Preview2:lle. Samanlaista tilannetta ei näytä olevan enää ensimmäisen julkaistun version kanssa. Näin ollen voidaan olettaa, että mikäli tuki ei lataudu itsestään ohjelmisto- tai järjestelmäpäivityksen mukana, se on helposti ladattavissa Google Play -kaupasta.

2.3 ARCore ja Unity

Kuten aikaisemmin mainittiin, Unity vaatii vähintään version ”Unity 2017.3.0f2”. Kyseinen versio on ensimmäinen versio, johon on lisätty tuki ARCorea varten. Versio, jota työssä käytettiin, ARCoren kanssa on ”Unity 2017.3.1f1”, joten kaikki kokemukset ja kuvat liittyvät kyseiseen versioon. Huomioitavaa on, kun tekee uutta ARCore projektia Unityssä, että 3D on varmasti valittuna. Vaikka Unity versio onkin ARCoren kanssa yhteensopiva ei se itsestään riitä vaan kehittäjän tulee ladata Googlen ARCore kehittäjä sivuilta Unitylle sopiva SDK-paketti (developers.google.com/ar/develop/unity/quickstart).

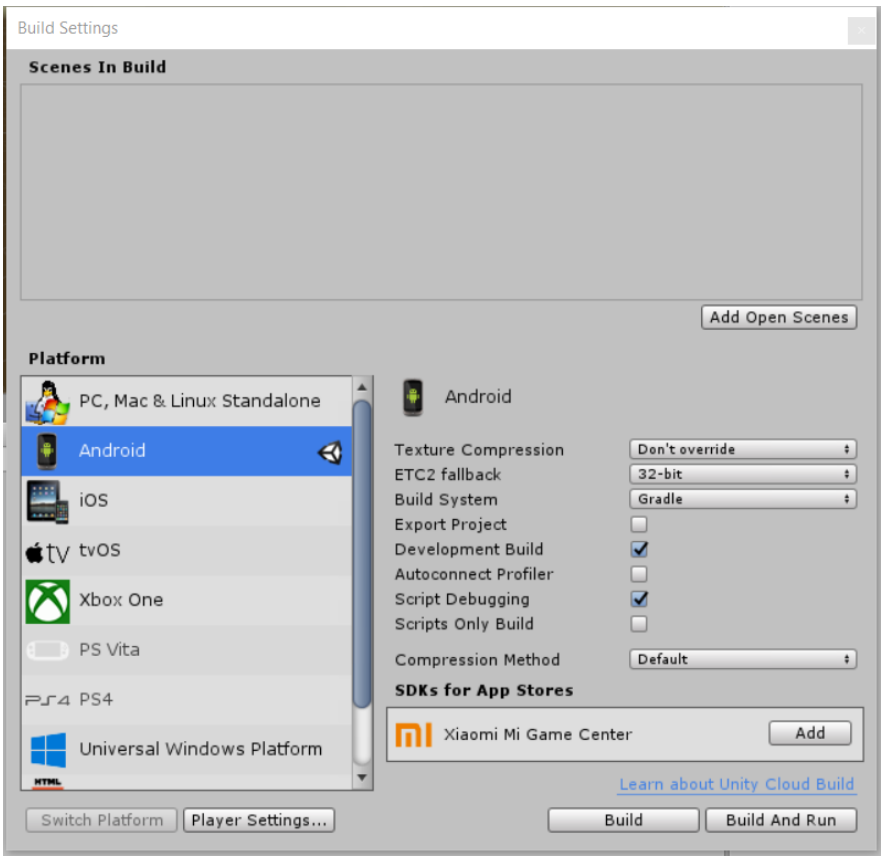
2.3.1 Build Settings

Ennen kuin tuodaan ladattu SDK-paketti Unity projektiin, kannattaa muuttaa kohdealusta PC:stä Androidiin. Tämä saadaan tehtyä siten että mennään File->Build Settings tai Ctrl+Shift+B, kuten alla olevassa kuvassa on näytetty (ks. Kuva 4).



Kuva 4. ”Build Settings” -ikkunan avaaminen

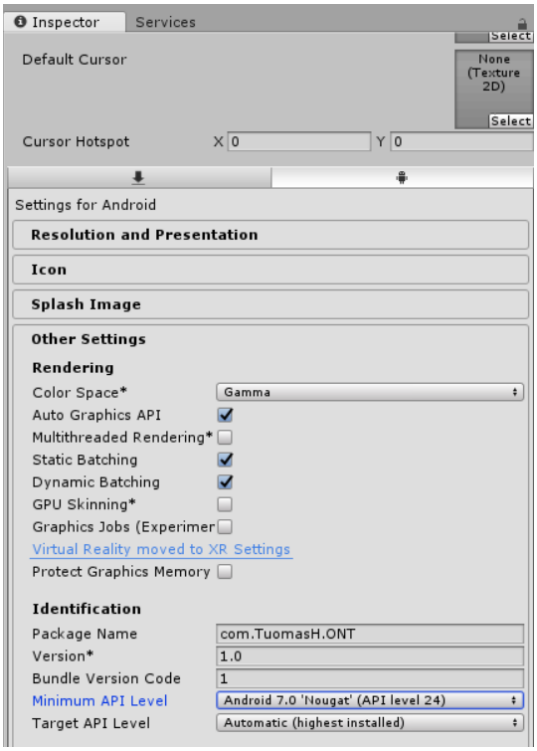
Tämän jälkeen aukeaa ”Build Settings” -ikkuna (Kuva 5). Sieltä tulee valita Android ja painaa ”Switch Platform” -nappia. Tämän jälkeen Unity vaihtaa kohdealustan Androidiin ja käy läpi jo olemassa olevat asset -tiedostot sekä tarvitseeko niihin tehdä muutoksia. Kohdealustan vaihdon voi tehdä myöhemmässäkin vaiheessa, mutta se tulee silloin kestämään pitempään, koska Unityllä on enemmän läpikäytäviä tiedostoja. Myöhemmässä vaiheessa tehty kohdealustan vaihto voi myös aiheuttaa ongelmia, joilta olisi voitu välttyä, jos vaihto olisi tehty heti alussa. Kun vaihto on tehty voi halutessaan tehdä lisävalintoja, kuten esimerkiksi ”Development Build”, joka antaa lisää mahdollisuuksia esimerkiksi vianetsinnän osalta. Kyseinen tilanne on esitetty alla olevassa kuvassa.



Kuva 5. Unity Build Settings

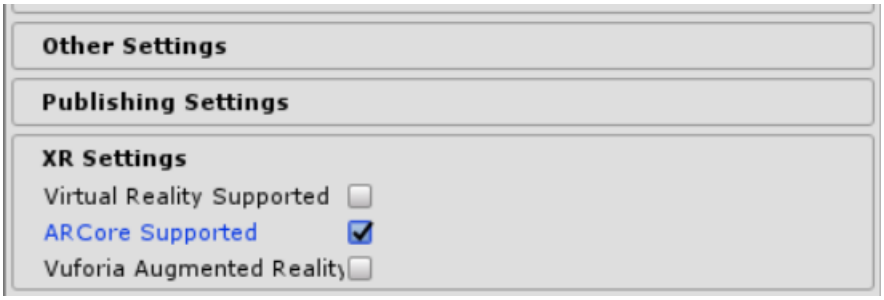
2.3.2 Player Settings

”Build Settings” -ikkunasta päästään myös muokkaamaan muita asetuksia klikkaamalla ”Player Settings…” -nappia. Kun nappia on painettu aukeaa ”Inspector” -välilehdelle pelaaja-asetuksia, joita täytyy muokata, jotta ARCore toimisi.



Kuva 6. Other Settings

Ensimmäisenä tulee mennä ”Other Settings” -kohtaan (ks. Kuva 6), josta pitää poistaa valinta ”Multithreaded Rendering” -kohdasta. Paketin nimi kohtaa tulee kehittäjän muokata omanlaisekseen. Alun perin nimi on muotoa: ”com.CompanyName.ProductName”. Myös minimi API taso tulee muuttaa ARCoren vaatimaksi ”Android 7.0 ’Nougat’ (API level 24)”. Kohde API tason voi jättää oletukseksi. Tässä tapauksessa se on sama kuin minimi API taso, koska koneelle on asennettu vain Android versio 7.0. Ennen kuin poistuu pelaaja-asetuksista, tulee vielä käydä lisäämässä ”XR Settings” -osioon ”ARCore Supported” kohtaan valinta (ks. Kuva 7).

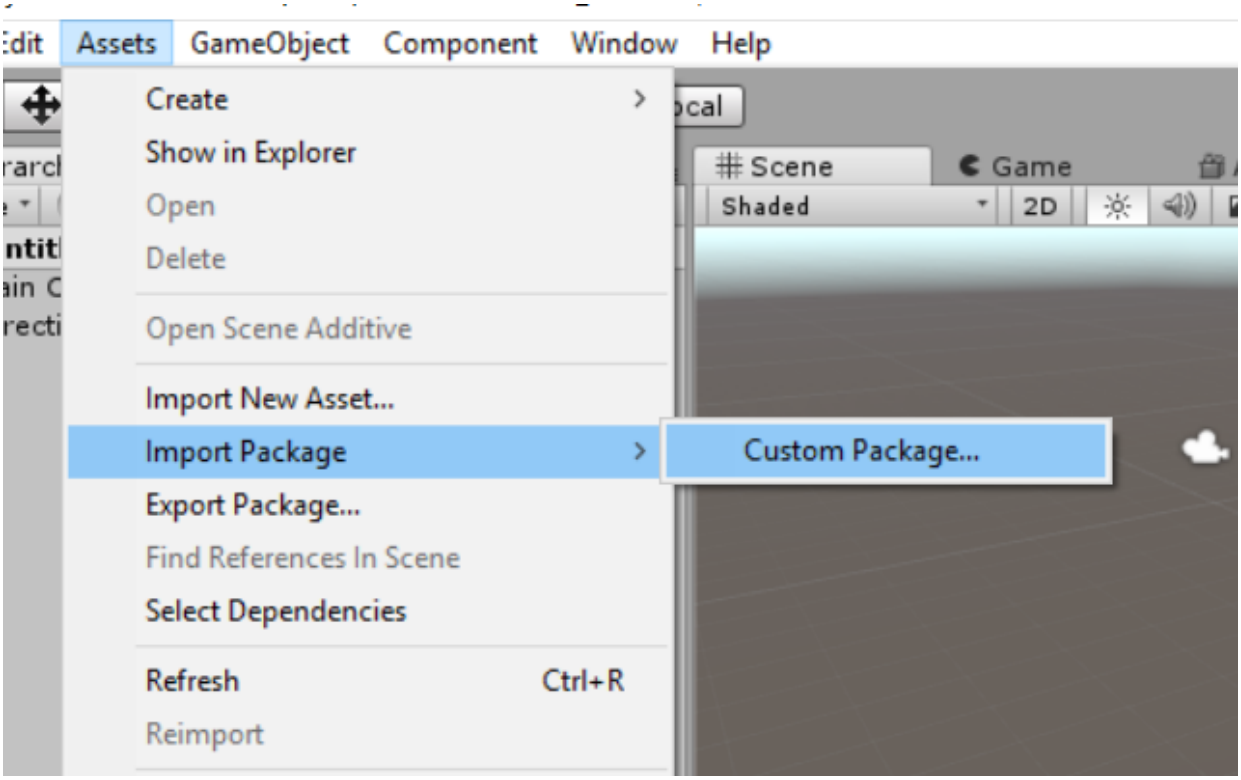


Kuva 7. XR Settings

Tämän jälkeen voidaan sulkea ”Build Settings” -ikkuna ja poistua pelaaja-asetuksista.

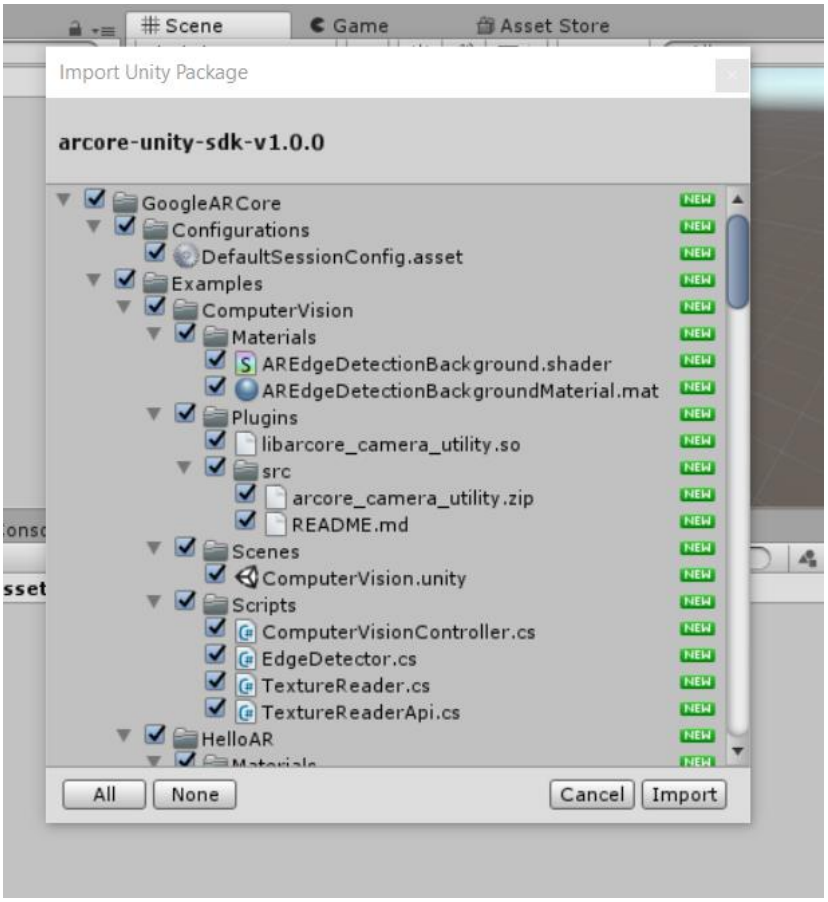
2.3.3 Import Package

Kun asetukset ovat kunnossa voidaan siirtyä tuomaan ladattu ARCore SDK paketti Unity projektiin.



Kuva 8. Pakettien tuominen

Unityyn saadaan tuotua uusia paketteja menemällä Assets -> Import Package -> Custom Package, kuten yllä olevassa kuvassa (ks. Kuva 8) on näytettynä. Kun Custom Package nappia on painettu aukeaa normaali tiedoston valinta ikkuna, josta kehittäjä voi valita tässä tapauksessa *arcore-unity-sdk-v1.0.0.unitypackage* nimisen tiedoston. Kun tiedosto on valittu aukeaa alla olevan näköinen ”Import Unity Package” -ikkuna (Kuva 9).



Kuva 9. Tuotavien tiedostojen valinta

”Import Unity Package” -ikkunassa kehittäjä pystyy valitsemaan, mitkä tiedostot tuo paketista projektiinsa. ARCoren tapauksessa kaikki muut kansiot paitsi ”Example” ovat tarpeellisia, mutta alussa sieltä löytyvistä koodeista on paljon apua. ARCoren oma esimerkki käydään tarkemmin läpi myöhemmässä vaiheessa. Kun ARCore SDK on tuotu projektiin niin kannattaa ensimmäisenä poistaa Unityn valmiiksi luomat Main Camera ja Directional Light, koska nämä korvataan ARCoren omilla vastineilla. Korvatakseen puuttuva kamera ja valonlähde, tulee kehittäjän vetää ARCoren ”Prefabs” kansiosta ”ARCore Device” ja ”Environmental Light” objektit hierarchy -osioon. Tämän jälkeen voidaan lähteä rakentamaan projektiin oikeaa sisältöä.

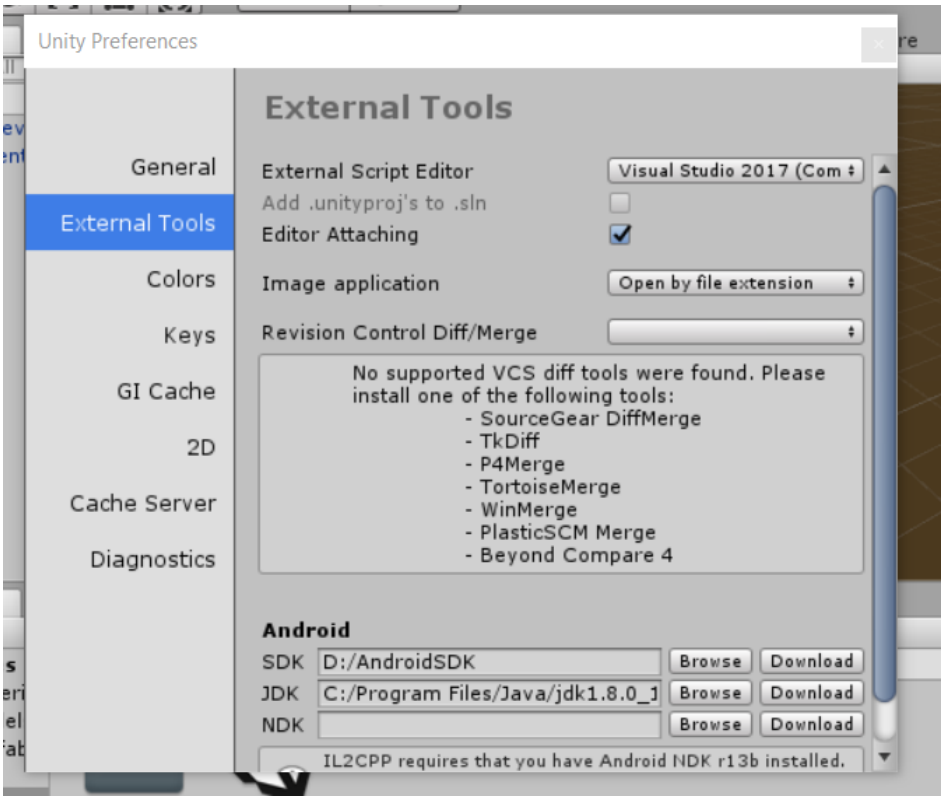
2.3.4 Haasteet ARCoren ja Unityn kanssa

Ensimmäinen vastaan tullut haaste tuli heti alussa, kun ensimmäistä kertaa ARCoren esimerkkiprojektia pyrittiin kokoamaan. Projektin kokoaminen kaatui tällöin *failed to read key* error viestiin. Tarkemmin ongelmaa tutkittua ja erilaisia ohjeita kokeiltua tullaan tulokseen, että joku oli mennyt vikaan Android Studiota asennettaessa. Tätä teoriaa vahvisti se, että kone, joka oli aluksi kehitysympäristönä, oli vanha ja huonossa kunnossa ja sen C-levyn muisti oli aivan täynnä. Vaikka asennettaisiin sovellukset D-levylle niin virhe pysyi samana, koska osa tiedoista edelleen oli C-levyllä tai sitten asennusohjelma väkisin asensi osan C:lle. Kehitystyötä tehtäessä toisella koneella, jossa oli paljon vapaata tilaa molemmilla levyillä ja Android Studio sai vapaasti asentaa itsensä C:lle niin virheilmoitusta ei enää tullut. Eli kun lähtee Android Studiota ja SDK-paketteja asentamaan kannattaa varmistaa, että levyillä on tarpeeksi tilaa, koska hetkellisesti asennus vaatii enemmän tilaa kuin, mitä lopullinen asennetun paketin koko on.

Seuraava haaste tuli vastaan, kun pyrittiin ensimmäistä kertaa kokoamaan ARCoren esimerkkiprojektia. Kone jota silloin käytin oli vanha ja hidas, mutta levyillä oli tarpeeksi tilaa Android Studion ja Android SDK asennusta varten. Hakiessani apua virheviestiin Googlesta tuli vastaan Reddit viestiketju (reddit.com/r/Unity3D/comments/77azfb/i_cant_get_unity_to_build_run_my_game/?st=jeqsploc&sh=9e676f93), jossa yksi vastaajista (alexspoettel) antoi toimivan ratkaisun ongelmaan. Tarkempaa tietoa ei ole onko kyseessä alkuperäisesti hänen ratkaisunsa. Tämä oli kuitenkin ensimmäinen toimiva sekä myös suhteellisen helppo toteuttaa.
Ratkaisu:

1. Poistaa Android SDK ”tools” kansio, joka löytyy Android SDK:n juuresta. Omassa tapauksessani se oli D:\AndroidSDK, mikä näkyi myös SDK:n asennuskuvassa (ks. Kuva 3).
2. Ladata uusi SDK Tools osoitteesta: dl-ssl.google.com/android/repository/tools_r25.2.5-windows.zip.
3. Purkaa ladattu zip-tiedosto Android SDK juureen.
4. Mennä Java arkistoihin osoitteessa: oracle.com/technetwork/java/javase/downloads/java-archive-javase8-2177648.html.
5. Etsiä julkaisu jdk-8u131.
6. Ladata ja asentaa kyseinen julkaisu. Huom! Tulee tehdä tunnus ja hyväksyä lisenssi sopimus.
7. Varmistaa että Unityssä on oikeat polut Android SDK:lle, sekä JDK:lle.

Polkuja SDK:lle ja JDK:lle pääsee asettamaan, kun menee Unityssä Edit-> Preferences. Tämän jälkeen tulisi aueta ”Unity preferences” -ikkuna, josta tulee mennä ”External Tools”-välilehdelle, josta kuva alla (ks. Kuva 10). Sieltä pystyy määrittämään polut SDK:lle ja JDK:lle, minkä jälkeen Unityn pitäisi koota projekti ongelmitta.



Kuva 10. Unity Preferences

Vaikka myöhemmässä vaiheessa kehittämisessä oli käytössä huomattavasti uudempi ja tehokkaampi kone, tuli silti tehdä tämä sama uudestaan ensimmäisellä kerralla, kun pyrittiin kokoamaana projektia.

2.4 Muuta huomioitavaa

Ensimmäinen huomioitava asia on, että joka kerta, kun lähtee rakentamaan uutta ARCore projektia Unityssä, kehittäjän tulee tehdä lukujen 2.3.1 – 2.3.3 vaiheet. Tämä johtuu siitä, että Unity ei tallenna asetuksia projektien välillä, mikä toisaalta on hyvä, koska se vaikeuttaisi työskentelyä muiden kuin ARCore projektien kanssa.

Toinen merkittävä asia ARCoressa ja Unityssä on, että ARCore katsoo Unityn mittayksiköiden olevan metreissä. Eli toisin sanottuna, jos esimerkiksi Unityn 3D koordinaatistossa kulkee yhden mittayksikön X-akselin suuntaisesti, olettaa ARCore -sovellus, että on liikuttu metrin verran oikealle.

Skriptejä kirjoitettaessa on huomioitava, että pitää muistaa laittaa jokaisen skriptin alkuun *using GoogleARCore;*, jolloin pääsee käsiksi ARCoren ominaisuuksiin skripteissäkin.

3 HELLOAR

HelloAR on Googlen tekemä demosovellus ARCorelle. Kyseisen sovelluksen tarkoituksena on tunnistaa tasaisia pintoja sekä visualisoida tunnistettua aluetta. Kun sovellus on tunnistanut tasaisia pintoja, pystyy käyttäjä koskettamaan visualisoidulta alueelta kohtaa, jolloin sovellus asettaa siihen kohtaan Androidin maskotin.

3.1 Yleiset osat

Kun HelloAR esimerkkiohjelmaa lähtee käymään läpi huomaa, että esimerkissä on kolme skriptitiedostoa. Näistä kolmesta kaksi on skriptejä, jotka käsittelevät visualisointeja. Tässä osiossa keskitytään käymään läpi sitä koodia (”HelloAR Controller”), mikä hoitaa varsinaisen tasojen etsimisen ja objektin lisäyksen tasoon. Skriptin alussa nimetään muuttujia, jotka ovat jo Googlen toimesta erittäin hyvin kommentoituja. Skripti ei sisällä *Start()* -funktiota, joten siirrytään suoraan *Update()* -funktioon. Heti *Updaten* alussa on alla näkyvä koodi (ks Kuva 11).

```
if (Input.GetKey(KeyCode.Escape))
{
    Application.Quit();
}
```

Kuva 11. Sovelluksen sulkemis koodi

Tämän tarkoituksena on vain antaa käyttäjälle helppo tapa sulkea sovellus ilman, että tarvitsee lisätä erillistä nappia UI:seen tätä varten. Tämän koodin lisäämisen jälkeen käyttäjä pystyy poistumaan sovelluksesta painamalla takaisinpäin nappia.

Seuraavana koodissa on ensimmäinen ARCoreen liittyvä osio, kun *Update* kutsuu *_QuitOnConnectionErrors()* -funktiota (ks. Kuva 12).

```
/// <summary>
/// Quit the application if there was a connection error for the ARCore session.
/// </summary>
private void _QuitOnConnectionErrors()
{
    if (m_IsQuitting)
    {
        return;
    }

    // Quit if ARCore was unable to connect and give Unity some time for the toast to appear.
    if (Session.Status == SessionStatus.ErrorPermissionNotGranted)
    {
        _ShowAndroidToastMessage("Camera permission is needed to run this application.");
        m_IsQuitting = true;
        Invoke("_DoQuit", 0.5f);
    }
    else if (Session.Status.IsError())
    {
        _ShowAndroidToastMessage("ARCore encountered a problem connecting.  Please start the app again.");
        m_IsQuitting = true;
        Invoke("_DoQuit", 0.5f);
    }
}
```

Kuva 12. _QuitOnConnectionErrors()

Googlen kommentin mukaan funktion tarkoituksena on sulkea sovellus, jos tulee yhteysvirhe ARCore sessioon. Ensimmäinen *if* -lause tarkistaa, onko muuttuja m_IsQuitting (muuttuja joka on määritelty skriptin alussa ja on tosi vain, jos on tullut virhe aikaisemmin), mikäli muuttuja on tosi niin katkeaa funktion läpikäynti ja palaa takaisin *Updateen*. Seuraava *if* -lause tarkistaa, onko käyttäjä antanut sovellukselle luvan käyttää kameraa, mikä on erittäin tärkeätä, koska kyseessä on AR -sovellus ja kameran läpi nähtävä maailma on olennainen osa sitä. *Else if* -lause vuorostaan tarkistaa, mikäli ARCore sessio on kohdannut jonkun ongelman, joka estää sovelluksen toiminnan. Molemmat kahdesta viimeisestä lauseesta kutsuvat *_ShowAndroidToastMessage()* -funktiota ja lähettävät sille itselleen ominaisen virheviestin. Kyseisen funktion näyttäminen ei ole tarpeellista, koska saman pystyisi hoitamaan UI tekstielementillä, jonka tekstin vaihtaisi vaan vastaamaan haluamaansa virheviestiä. Tällöin tulee vaan muuttaa alimmassa *Invoke()* -kohdassa olevaa numeroarvoa siten että käyttäjä kerkeää lukea virheviestin ennen kuin sovellus sulkeutuu. Molemmat myös muokkaavat *m_IsQuitting* -muuttujan arvon todeksi, jotta sovellus ei suotta käy virheentarkastusta heti seuraavalla *Updaten* kutsulla uudestaan vaan katkeaa ensimmäiseen *if* -lauseeseen.

Seuraava kohta *Updatesta* tarkistaa, onko liikkeen seuranta päällä (ks. Kuva 13).

```
// Check that motion tracking is tracking.
if (Session.Status != SessionStatus.Tracking)
{
    const int lostTrackingSleepTimeout = 15;
    Screen.sleepTimeout = lostTrackingSleepTimeout;
    if (!m_IsQuitting && Session.Status.IsValid())
    {
        SearchingForPlaneUI.SetActive(true);
    }

    return;
}
```

Kuva 13. Liikkeen seurannan tarkistus

Mikäli liikkeen seuranta ei ole päällä, asettaa sovellus näytön sammumiselle tietyn ajan. Tämän jälkeen, mikäli edellisessä kohdassa ei ole tullut mitään virhettä, asettaa funktio *SearchingForPlaneUI* -osion näkyviin, jonka jälkeen poistuu *Updatesta*. Funktion jälkeen, mikäli siis liikkeen tunnistus on päällä, asetetaan *Screen.sleepTimeout*:n arvoksi *SleepTimeout.NeverSleep*, jolloin näyttö pysyy automaattisesti päällä, ellei käyttäjä sitä itse sammuta.

3.2 Tunnistettujen pintojen visualisointi

```
// Iterate over planes found in this frame and instantiate corresponding GameObjects to visualize them.
Session.GetTrackables<TrackedPlane>(m_NewPlanes, TrackableQueryFilter.New);
for (int i = 0; i < m_NewPlanes.Count; i++)
{
    // Instantiate a plane visualization prefab and set it to track the new plane. The transform is set to
    // the origin with an identity rotation since the mesh for our prefab is updated in Unity World
    // coordinates.
    GameObject planeObject = Instantiate(TrackedPlanePrefab, Vector3.zero, Quaternion.identity,
        transform);
    planeObject.GetComponent<TrackedPlaneVisualizer>().Initialize(m_NewPlanes[i]);
}
```

Kuva 14. Tasojen visualisointi

Yllä kuvattu osio (ks. Kuva 14) hakee ARCoren Sessio tiedoista, kyseisen framen aikana löydetyt tasot ja asettaa ne *m_newPlanes* -listaan. Tämän jälkeen käydään *for* -loopissa läpi jokainen listan arvo ja lisätään sille visualisointia. Visualisointi tehdään lisäämällä esimerkki projektista löytyvä *TrackedPlanePrefab* koordinaatiston origoon. Sitten lisätystä objektista kutsutaan toista visualisointi skripteistä, tässä tapauksessa *TrackedPlaneVisualizer*, joka alustaa lisätyn objektin, sillä hetkellä vuorossa olevan listan arvon mukaan.

```
// Disable the snackbar UI when no planes are valid.
Session.GetTrackables<TrackedPlane>(m_AllPlanes);
bool showSearchingUI = true;
for (int i = 0; i < m_AllPlanes.Count; i++)
{
    if (m_AllPlanes[i].TrackingState == TrackingState.Tracking)
    {
        showSearchingUI = false;
        break;
    }
}

SearchingForPlaneUI.SetActive(showSearchingUI);
```

Kuva 15. Tason etsintä UI:n näkyviin asettaminen

Seuraavaksi skripti tarkistaa onko kaikki tasot voimassa olevia ja sen jälkeen asettaa *SearchingForPlaneUI* -elementin aktiiviseksi tai ei (ks. Kuva 15).

3.3 Kosketus ja Objektin lisääminen haluttuun kohtaan.

Kun tämän hetkiset tasot ovat visualisoitu voidaan niihin lisätä objekteja, kuten esimerkki projektin Android maskotti. Ensimmäisenä skripti tarkistaa onko kosketuksia vähemmän kuin 1 eli onko näyttöä kosketettu ollenkaan tai onko näyttöä kosketettu edellisen framen aikana. Mikäli jompikumpi ehdoista täyttyy, poistuu skripti *Update()* -funktiosta (ks. Kuva 16).

```
// If the player has not touched the screen, we are done with this update.
Touch touch;
if (Input.touchCount < 1 || (touch = Input.GetTouch(0)).phase != TouchPhase.Began)
{
    return;
}
```

Kuva 16. Näytön kosketuksen tunnistaminen

Seuraava osuus koodissa tarkistaa, että mihin käyttäjän kosketus osuu ja onko se kohde, joka aiheuttaa reaktion koodissa. Sen sijaan että alla olevassa koodissa olisi käytetty normaalia *RaycastHit* -mehtodia käsitelläkseen osumaa niin koodi käyttää *TrackableHit*, joka sisältää tietoja objektista, jota ARCore seuraa. Tässä tapauksessa halutaan, että osuma kohdistuisi tasoihin, joten käytetään *TrackableHitFlags*, joka toimii filtterinä osumalle. Osumista filtteröidään ne, jotka osuvat tasoihin näiden rajojensa sisällä tai paikkapisteeseen, joista niiden kulma tiedetään.

```
// Raycast against the location the player touched to search for planes.
TrackableHit hit;
TrackableHitFlags raycastFilter = TrackableHitFlags.PlaneWithinPolygon |
    TrackableHitFlags.FeaturePointWithSurfaceNormal;

if (Frame.Raycast(touch.position.x, touch.position.y, raycastFilter, out hit))
```

Kuva 17. Raycast ja sen vaatimat muuttujat

Mikäli yllä oleva *if*-lause (ks. Kuva 17) on tosi ja käyttäjän kosketus osuu, joko seurattuun tasoon tai halutun laiseen paikkapisteeseen niin voidaan lisätä objekti siihen kohtaan. Tämä tapahtuu siten että luodaan uusi objekti, joka käyttää esimerkki projektista löytyvää *AndyAndroidPrefab* -objektia ja sijoitetaan luotu objekti kohtaan, jota käyttäjä kosketti ja käännetään objekti vastaamaan kosketetun kohdan kulmaa. Kun Andy on lisätty, täytyy sille lisätä ankkuri, jota ARCore pystyy seuraamaan, vaikka sen käsitys maailmasta laajenisi. Kuten alla näkyy (ks. Kuva 18), ankkurin luonti tapahtuu myös samaan kohtaan, jota käyttäjä kosketti.

```
var andyObject = Instantiate(AndyAndroidPrefab, hit.Pose.position, hit.Pose.rotation);

// Create an anchor to allow ARCore to track the hitpoint as understanding of the physical
// world evolves.
var anchor = hit.Trackable.CreateAnchor(hit.Pose);
```

Kuva 18. Andyn ja ankkurin luominen

Tämän jälkeen voidaan kääntää Andy katsomaan käyttäjää päin. Tämä tapahtuu käyttämällä *transform.LookAt()* -methodia, joka kääntää tässä tapauksessa Andyn katsomaan kameraan, joka on asetettu kohteeksi. Andyn tulee myös olla suorassa tasoon, jolla se seisoo, nähden. Tämä saadaan aikaiseksi siten, että kameran Y-arvon sijaan käytetään Andylla jo olevaa Y-arvoa, koska yleensä kameran Y-arvo on korkeammalla kuin mitä Andyn on. Tekemällä tämän pienen muutoksen, kuten koodissa on näytetty (ks. Kuva 19), saadaan Andy katsomaan suoraan, mutta käyttäjää päin.

```
// Andy should look at the camera but still be flush with the plane.
if ((hit.Flags & TrackableHitFlags.PlaneWithinPolygon) != TrackableHitFlags.None)
{
    // Get the camera position and match the y-component with the hit position.
    Vector3 cameraPositionSameY = FirstPersonCamera.transform.position;
    cameraPositionSameY.y = hit.Pose.position.y;

    // Have Andy look toward the camera respecting his "up" perspective, which may be from ceiling.
    andyObject.transform.LookAt(cameraPositionSameY, andyObject.transform.up);
}

// Make Andy model a child of the anchor.
andyObject.transform.parent = anchor.transform;
```

Kuva 19. Andyn kääntäminen käyttäjän suuntaan

4 ARCORE YMPÄRISTÖN VISUALISOIMISESSA

Sovellus, johon opinnäytetyö toimii demona, tulee todennäköisimmiten olemaan tehtaan tai muun vastaavan isomman alueen visualisoinnin apuväline. Tämä tarkoittaa sitä, että sovelluksen kuten myös demon näin ollen tulee toimia myös isommalla kuin vain huoneen mittakaavalla. Koska ARCore rakentaa käsitystä ympäristöstä koko käyttökerran ajan, niin sovellus alkaa toimia hitaammin, mikäli kaikki pisteet, jotka näyttävät dataa ovat päällä saman aikaisesti. Myös jos visualisoitavia kohteita on paljon ja kaikki kiinnitetään ARCoren Anchor -ominaisuudella yhtäaikaisesti, niin rasittaa se laitetta huomattavasti ja siten hidastaa sovelluksen toimintaa.

Tähän ratkaisuna päädyttiin siihen, että aina kun laite on liikkunut tietyn määrän se ajaa tarkistus silmukan läpi, joka käy kaikki visualisoitavat kohteet ja tarkistaa kuinka lähellä ne ovat laitetta. Mikäli kohde on halutulla etäisyydellä laitteesta, luodaan kohde näkyviin ja ankkuroidaan se, jotta se pysyy paikallaan laitteen liikkuessa sen lähiympäristössä. Ja myös päinvastoin eli jos laite meni liian kauaksi kohteesta, poistettiin se näkyvistä ja samalla myös poistettiin tarpeeton ankkuri, jolloin laite ei alkanut kärsiä niin paljoa käytöstä isossa tilassa.

Vaikkakin lopullinen käyttökohde olisikin todennäköisimmiten laaja tehdasympäristö, tuli myös tutkia pienen alueen visualisointia. Tällöin monta kohdetta olisi lähellä toisiaan, mikä aiheuttaa epäselkeyttä varsinkin, jos tavoitteena on saada visualisoitua dataa. Ongelmaksi nousi myös se, että jos kohteita on paljon ja kaikki halutaan kiinnittää ankkureilla ympäristöön, hidastaa tämä laitetta huomattavasti, kun laite koittaa laskea koko ajan jokaisen ankkurin sijaintia.

Yksi esille noussut asia tutkittaessa työn toimivuutta erilaisissa olosuhteissa on, että ARCorella on vaikea visualisoida asioita, joiden tulee olla esim. senttimetrin tarkkuudella oikeassa kohdassa. Tämä tuli esille muun muassa, kun testattiin mitenkä sovellus reagoi, kun mennään tarpeeksi kauaksi visualisoitavasta kohteesta, jolloin se poistuu näkyvistä ja sitten palataan tarpeeksi lähelle. Tällöin kun laitetta oli tarpeeksi paljon käytetty ja laite oli käytöstä hieman lämmennyt, saattoi visualisoitavan kohteen alkuperäisen ja nykyisen sijainnin välillä olla, jopa kymmenen centtimetriä eroa.

Koska visualisoitavat kohteet sijoitetaan sovelluksen käynnistyspisteeseen nähden voi käynnistys pisteestä riippuen voi visualisoitavat kohteet olla vaikka kuinka kaukana oikeista kohteista, joita halutaan visualisoida. Tähän yksi ensimmäisistä esille tulleista ratkaisuehdotuksista oli puhelinteline, jolloin sovellus voitaisiin käynnistää aina täsmälleen samassa kohtaa. Puhelintelineen käytön estää kuitenkin se, kun tätä mahdollisuutta tutkittaessa tuli ilmi se, että puhelimen kamera ei saa olla peitettynä, vaan puhelimen pitää nähdä eteenpäin, jotta se voisi löytää kiinnepisteitä ja siten aloittaa laitteen sijainnin seuraamisen. Telineestä riippuen kamera saattaa olla peitettynä, joka johtaa siihen, että sovellus ei lähde toimimaan käynnistettäessä tällaisessa telineessä.

Myös ongelma mikä tuli ilmi testausvaiheessa oli, että mikäli kamera suunnataan esimerkiksi valkoista seinää päin, jolloin laite ei löydä yhtään kiinnepistettä, kadottaa se ymmärryksen koko ympäristöstä. Ymmärrys ympäristöstä ei palaa enää takaisin sen jälkeen, jos laite sen on kadottanut, vaan tällä hetkellä ainut mahdollisuus on käynnistää sovellus uudestaan. Ja jotta visualisoitavat kohteet olisivat mahdollisimman lähellä oikeita kohteitaan, tulee käyttäjän palata takaisin aloituspisteeseen, jossa vasta sovellus tulee pistää päälle.

5 ARCOREN TULEVAISUUS

On mahdotonta ennustaa, minkälainen tulevaisuus ARCorella tekniikkana on, koska tekniikat kehittyvät jatkuvasti ja uusia parempia tekniikoita saattaa ilmestyä. Koska ARCore tarjoaa erilaisia mahdollisuuksia lisätyntodellisuuden sovelluksille kuin esimerkiksi Vuforia, niin voidaan olettaa, että ARCore ei ole heti kuolemassa pois vaan Googlella on aikaa kehittää teknologiaansa. ARCoren tulevaisuus on myös pitkälti riippuvainen siitä miten laajan tuen Android laitteista se saa. Mikäli jatkossa jokainen Android laite tukee ARCorea laitteen tehokkuuden puolesta, eikä vain Android version, niin tulee ARCoren loppukäyttäjä määrä kasvamaan jatkossa sitä mukaa kun vanhan Android version ja huonomman raudan omaavat laitteet poistuvat käytöstä.

Iso osa ARCoren tulevaisuudesta on kiinni sen omasta kehittäjästä Googlesta. Google voi päivityksillään vaikuttaa ARCoren suosioon, joko kehittämällä jo olemassa olevaa tekniikkaa siten, että jotkut nykyiset ongelmat pienenevät tai häviävät kokonaan tai sitten kehittämällä tekniikkaan kokonaan uusia ominaisuuksia. Tästä on hyvänä esimerkkinä äskettäin julkaistu ARCoren versio 1.2, jossa Google muun muassa lisää ARCoreen ominaisuuden, joka sallii jaetun lisätyntodellisuuden kokemuksen huomattavasti helpommin. Kyseisen päivityksen mukana tuli myös ominaisuus nimeltä ”Augmented Images”. Tarkemmin ei ole vielä tietoa tuleeko tämä nostamaan ARCoren huomattavasti muiden tekniikoiden yläpuolelle tai jopa syrjäyttämään kokonaan esimerkiksi Vuforian. On selvää kuitenkin, että Google pyrkii paikkaamaan aukkoja, joita ARCoressa on ollut verrattuna muihin tekniikoihin. Tämän kaltaiset päivitykset voivat tulevaisuudessa nostaa ARCoren pääasialliseksi tekniikaksi, kun tehdään lisätyntodellisuuden ohjelmistoa Android laitteille. (Gosalia, Experience augmented reality together with new updates to ARCore, 2018.)

6 LÄHDELUETTELO

Google Developers. (2018). *ARCore Overview*. Haettu 1. Maaliskuu 2018 osoitteesta Developers Google: https://developers.google.com/ar/discover/

Google Developers. (2018). *Fundamental Concepts*. Haettu 12. Maaliskuu 2018 osoitteesta Google Developers: https://developers.google.com/ar/discover/concepts

Gosalia, A. (23. Helmikuu 2018). *Announcing ARCore 1.0 and new updates to Google Lens*. Haettu 12. Maaliskuu 2018 osoitteesta Google Blog: https://www.blog.google/products/google-vr/announcing-arcore-10-and-new-updates-google-lens/

Gosalia, A. (8. Toukokuu 2018). *Experience augmented reality together with new updates to ARCore*. Haettu 13. Toukokuu 2018 osoitteesta Google Blog: https://www.blog.google/products/google-vr/experience-augmented-reality-together-new-updates-arcore/

HiddenBrains. (2017). *Google ARCore vs Apple ARKit Powerplay continues in Augmented Reality*. Haettu 12. Maaliskuu 2018 osoitteesta Medium: https://medium.com/@hiddenbrains/google-arcode-vs-apple-arkit-powerplay-continues-in-augmented-reality-d9aa3b95c906

Unity3D Reddit. (2017). Haettu 18. Maaliskuu 2018 osoitteesta Unity3D Reddit: https://www.reddit.com/r/Unity3D/comments/77azfb/i_cant_get_unity_to_build_run_my_game/?st=jeqsploc&sh=9e676f93

Vuforia. (2018). *Vuforia*. Noudettu osoitteesta https://www.vuforia.com/

Kuva1 : Google Developers. Fundamental Concepts [verkkojulkaisu]. [Viitattu 2018-03-13] Saatavissa: https://developers.google.com/ar/discover/concepts