AR-sovelluksen kehittäminen visuaalista ohjelmointia käyttäen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus

kevät, 2021

Toni Heinonen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Hämeenlinnan korkeakoulukeskus

Tekijä Toni Heinonen Vuosi 2021

Työn nimi AR-sovelluksen kehittäminen visuaalista ohjelmointia käyttäen

Ohjaaja Tommi Saksa

TIIVISTELMÄ

Opinnäytetyössä kehitettiin Android-laitteelle lisätyn todellisuuden sovellus visuaalista ohjelmointia käyttäen. Ohjelmistokehyksenä käytettiin Unreal Engine -pelimoottoria, lisätyn todellisuuden mahdollistivat Google ARCore -ohjelmointirajapinnat, ja sovellus ohjelmoitiin visuaalisella Blueprint-ohjelmointikielellä. Vastaavanlaista teknologiaa tutkivaa opinnäytetyötä ei ole tehty. Opinnäytetyön tavoitteet olivat kuvan tunnistukseen pohjautuvan AR-mobiilisovelluksen kehittäminen ja käytetyn teknologian dokumentoinnin lisääminen.

Opinnäytetyön tietoperustassa käytettiin lähteinä valittujen ohjelmistojen kehittäjien dokumentaatioita, alan tutkimuksia ja internet-julkaisuja. Tietoperusta antaa kattavan yleiskuvauksen opinnäytetyössä käytetystä teknologiasta. Opinnäytetyö on toiminnallinen kehittämistyö, joka kuvantaa sovelluksen kehittämisen suunnittelusta julkaisuun.

Kehittämistyön tuloksena syntyi AR-mobiilisovellus, joka näyttää käyttäjälle tunnistetun tuotteen tuotetietoja. Visuaalisen ohjelmoinnin todettiin olevan sopiva työkalu mobiilisovelluksen kehittämiseen eikä tekstipohjaista ohjelmointia käytetty lainkaan. Kehitystyöprojektista teki ajoittain haasteellisen käytettyjen ohjelmistojen ajantasaisen lähdemateriaalin puute. Koska vastaavanlaista teknologiaa sisältävää opinnäytetyötä ei ole tehty, luo opinnäytetyöraportti oppaana lisäarvoa tuleville kehittäjille.

Avainsanat Google ARCore, Lisätty todellisuus, Unreal Engine Blueprint, Visuaalinen ohjelmointi

Sivut 55 sivua ja liitteitä 1 sivu

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna University Centre

Abstract

Author Toni Heinonen
Year 2021
Subject The Development of an AR application by using visual scripting
Supervisor Tommi Saksa

ABSTRACT

The thesis aimed to develop an AR application by using visual scripting. The subject of the thesis was chosen out of interest in combining visual scripting and mobile development. The augmented reality was chosen to be the theme for the application because of an intriguing challenge to develop an AR-experience.

The theory of thesis starts with an introduction to augmented reality. Next Unreal Engine, visual scripting, and Google ARCore API are discussed. The theoretical framework is based on different research papers, publications, and official documentation of used software.

The type of thesis is practical. The primary goal was to develop a mobile application based on the chosen technology. The practical chapter reviews the journey from setting up the environment to planning the software and publishing it.

As the result of the thesis, an AR mobile application that uses image recognition was developed. The application presents product information for the tracked product. It was found out that the lack of quality documentation for chosen technology was difficult to overcome. In the future thesis can function as source material for developers.

Keywords Augmented Reality, Google ARCore, Unreal Engine Blueprint, Visual Scripting
Pages 55 pages and 1 page of appendices

Sanasto

APK	Android Application Package, Android-sovellusten tiedostotyyppi
AR	Augmented Reality, lisätty todellisuus
ARCORE	Googlen kehittämä lisätyn todellisuuden kehitysalusta Android-käyttöjärjestelmille
BLUEPRINT	Unreal Enginen käyttämä visuaalinen ohjelmointimenetelmä
COLLIDER	Komponentti, jolla määritetään peliobjektin mahdollisuus ottaa kontakti toiseen peliobjektiin
DIRECTX	Microsoftin kehittämä ohjelmistorajapinta tietokoneohjelman ja laitteiston välille 3D-grafiikkaa, ääntä ja ohjauslaitteita varten
FPS	First Person Shooter, pelihahmon näkökulmasta kuvattu ammuntapeli
HDR	High Dynamic Range, kuvan kontrastia parantava tekniikka
MR	Mixed Reality, yhdistelmä virtuaalitodellisuutta ja lisättyä todellisuutta
PELIMOOTTORI	Sovelluskehitysympäristö, jota kehittäjät käyttävät pelien luomiseen
SLAM	Simultaneous Localization and Mapping, samanaikainen paikannus- ja kartoitustekniikka, jolla laite tunnistaa ympäristöään ja sijoittautumista siihen
UNREALSCRIPT	Unreal Enginen käyttämä ohjelmointikieli Unreal Engine 4:een asti
UNREAL ENGINE	Pelimoottori, jonka kehittäjä on Epic Games
VR	Virtual Reality, virtuaalitodellisuus
XR	Extended Reality, yläkäsite kaikille todellisuuden ja virtuaalisen todellisuuden välisille yhdistelmille

Sisällys

Kuvat ja taulukot

Liitteet

1 Johdanto

Tässä opinnäytetyössä yhdistyy kaksi viime vuosikymmenen aikana kovasti kehittynyttä teknologista saavutusta: lisätty todellisuus ja visuaalinen ohjelmointi. Teema on ajankohtainen, sillä sijoittaminen virtuaaliteollisuuteen on moninkertaistunut ja voidaan olettaa, että ala tulee olemaan lähitulevaisuudessa nykyistäkin merkittävämpi työllistäjä.

Lisätty todellisuus eli Augmented Reality, on tekniikka, jossa läpikatseltavalle näytölle lisätään virtuaalisia elementtejä rikastuttaen käyttäjän kokemusta todellisesta maailmasta. Tämän hetken tunnetuimmat lisätyn todellisuuden tuotteet ovat Unitylla kehitetty Pokémon GO -mobiilipeli, Snapchatissa käytettävät tehosteet/suodattimet, sekä IKEA Placen kaltaiset sovellukset, joilla voidaan mallintaa ostettavia tuotteita kodin ympäristöön. Visuaalinen ohjelmointi on ohjelmointimenetelmä, jossa tekstimuotoisen ohjelmakoodin sijaan kehittäjä yhdistelee graafisia lohkoja tai kuvakkeita luoden vastaavat toiminnallisuudet. Visuaalinen ohjelmointi voi helpottaa ohjelmoinnin opiskelua ja sillä voidaan luoda koodia nopeasti.

Tavoitteena tässä työssä on kehittää lisättyä todellisuutta hyödyntävä Android-sovellus Unreal Engine -pelimoottorin visuaalisella ohjelmointimenetelmällä. Mielenkiintoisen haasteen työlle tuo se, että visuaalinen ohjelmointi ei ole tunnetuin tapa tuottaa sovelluksia, eikä ohjelmistokehyksenä Unreal Engine ole yhtä tunnettu virtuaalitodellisuuden kehittämiskäytössä kuin kilpailija Unity. Tutkimuksen rakenne on seuraava: Ensimmäisenä esitellään työssä käytettäviä teknologioita. Seuraavana on vuorossa sovelluksen kehitysprosessin kuvaus aina ohjelmistojen valinnasta ja yhteensopivuuden kartoittamisesta julkaistavaan sovellukseen asti. Lopuksi summataan projektin yhteenveto, jossa analysoidaan, kuinka hyvin keskeisiin tutkimuskysymyksiin on saatu vastaus ja minkälainen arvo työllä on tulevaisuuden kannalta.

Tutkimuskysymykset ovat:

- Miten lisätty todellisuus muuttaa käyttäjän kokemusta todellisesta maailmasta?
- Mitä rajoituksia Unreal Enginellä ja ARCorella kehittäessä tulee vastaan?
- Kuinka hyvin visuaalinen ohjelmointi soveltuu AR-sovelluksen kehittämiseen?

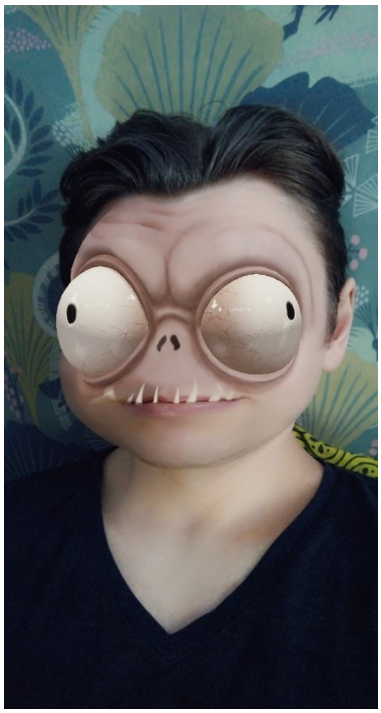
2 Lisätty todellisuus

Lisätty todellisuus on visuaalisen ohjelmoinnin lisäksi toinen tämän opinnäytetyön keskeisistä teknologioista. Tämä luku käsittelee lisättyä todellisuutta ja käy läpi sen historiaa ja kehittymistä tähän päivään.

2.1 Määritelmä

Lisätty todellisuus (Augmented Reality, AR) yhdistää digitaalisia elementtejä todelliseen maailmaan. Yleisin toteutustapa on käyttää älypuhelimen kameraa ja lisätä ruudulle reaaliaikaisesti virtuaalista sisältöä. Tunnetuimmat AR-sovellukset ovat Pokémon Go ja Snapchat. Pokémon Go on mobiilipeli, jossa pelaajan tehtävä on toimia Pokémon-kouluttajana ja kerätä luonnossa esiintyviä Pokémoneja. Kun Pokémon löydetään, se voidaan ottaa kiinni mobiililaitteen kameran kuvaamassa todellisessa ympäristössä. Snapchatin toiminta perustuu kasvojentunnistusalgoritmeihin. Ensin sovellus tunnistaa konenäön avulla kameran kuvaamasta ympäristöstä kasvot, minkä jälkeen voidaan käyttää reaaliajassa erilaisia virtuaalisesti lisättyjä tehostimia ja suodattimia muokkaamaan otettavan kuvan sisältöä (Le, 2018). Kuva 1 sisältää kaksi kuvaa lisätyn todellisuuden sovelluksista, vasemmalla puolella on kuvattuna Pokémon Go -mobiilipelin Pokémonin kiinniottotilanne ja oikealla Snapchat-filtterin käyttö.

Kuva 1. Lisätty todellisuus Pokémon Go ja Snapchat -sovelluksissa



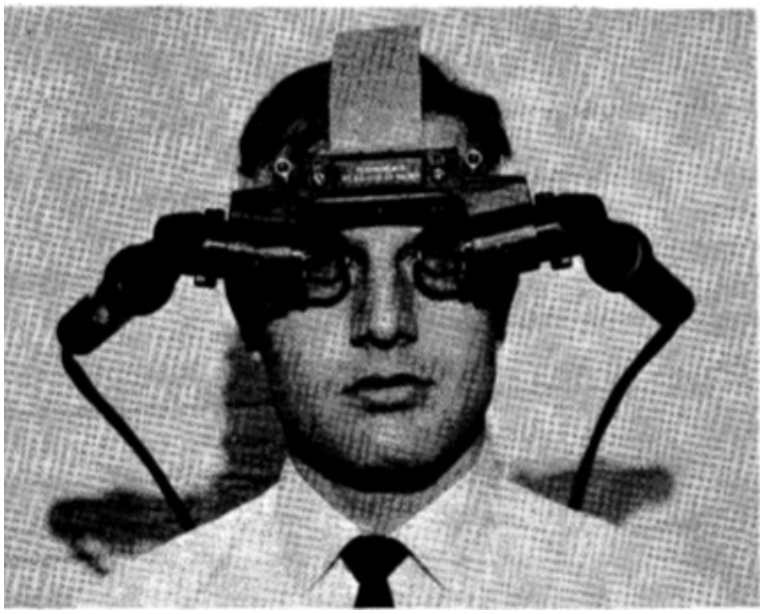
Lisätty todellisuus voidaan määritellä käsitteen jatkettu todellisuus (Extended Reality, XR) alapuolelle. Siihen kuuluvat myös virtuaalitodellisuus (Virtual Reality, VR), sekä lisättyä- ja virtuaalista todellisuutta yhdistävä yhdistetty todellisuus (Mixed Reality, MR) (Gupton & Kiger, 2020). Lisätty todellisuus eroaa virtuaalitodellisuudesta siten, että virtuaalitodellisuus on digitaalisesti kehitetty ja täysin todellisesta maailmasta irrallinen ympäristö. Jotta käyttäjä voisi kokea virtuaalitodellisuuden, tarvitaan tietokoneella mallinnettu 3D-ympäristö ja puettavat VR-lasit. Usein käytössä on myös ohjaimet, joilla voidaan käsitellä virtuaalitodellisuudessa olevia objekteja. Kuva 2 näyttää Euroopan avaruusjärjestön koulutustilanteen, jossa tutkija demonstroi miten astronautit voisivat käyttää virtuaalista todellisuutta hyödyksi harjoitellakseen tehtävän suorittamista avaruuden olosuhteissa. (Johnson, 2020)

Kuva 2. Koulutustilanne Euroopan avaruusjärjestössä (ESA, 2017)

2.2 Historia ja kehittyminen

Ensimmäisenä AR-laitteena pidetään Ivan Sutherlandin vuonna 1968 kehittämää laitetta (Kuva 3). Se oli päähän kiinnitettävä laite, jossa oli kaksi optista näyttöä ja minikatodisädeputkea sekä pään sijaintia mittaavat ultraäänianturit. Käyttäjä näki laitteen linssien läpi sekä todellisen ympäristönsä että hänen eteensä keinotekoisesti heijastetun kolmiulotteisen kuution, minkä vuoksi laitetta pidetään ensimmäisenä lisätyn todellisuuden keksintönä. (Sutherland, 1968)

Kuva 3. Ivan Sutherlandin kehittämä päähän kiinnitettävä laite (Sutherland, 1968)



Lisätty todellisuus terminä ei ollut tunnettu ennen vuotta 1992, jolloin Boeingin tutkijat Thomas Caudell ja David Mizell julkaisivat dokumentin, jossa he mainitsivat sen ensimmäistä kertaa. Modernien lentokoneiden rakentaminen oli alkanut käydä monimutkaisemmaksi ja he tutkivat keinoa helpottaa ja nopeuttaa prosessia. He kehittivät päähän kiinnitettävän laitteen, jossa oli läpinähtävät linssit. Linsseihin voitiin heijastaa esimerkiksi yksinkertainen rautalankamalli, mallin ääriviivat tai ohjeistavaa tekstiä. Heijastettava sisältö ei vaatinut suurta laskennallista tehoa, joten se voitiin toteuttaa tavallisilla mikroprosessoreilla. (Caudell & Mizell, 1992)

Steve Mann on professori Toronton yliopistossa. Hänet tunnetaan useiden puettavien- ja AR-laitteiden keksijänä. Hän on kehittänyt puettavan laitteen, jolla voidaan projisoida interaktiivista lisättyä todellisuutta (Wearcam, n.d.), rannekello-videopuhelimen (2000), näytön ja kameran yhdistelmänä toimivat digitaaliset silmälasit, HDR-tekniikan, sekä useita muita keksintöjä (Mann, n.d.). Laitteiden voidaan sanoa olleen edellä aikaansa, esimerkiksi Mannin keksimät digitaaliset silmälasit ovat vuodelta 1984 ja Googlen kehittämä Google Glass julkaistiin vasta vuonna 2013 (Mannlab, 2020).

2.3 Lisätyn todellisuuden toteutustavat

Lisätty todellisuus voidaan toteuttaa merkkauspohjaista, merkkaamatonta, sijaintiperusteista, päälle laitettavaa, hahmottelevaa tai projisointitekniikkaa käyttäen. Seuraavat kuvaukset perustuvat Collinsin (2018/2020) artikkelissa esitettyihin tietoihin.

Merkkauspohjainen (Marker-based) käyttää pohjana erilaisia ankkureita, kuten QR-koodia tai muuta selkeästi erottuvaa kuvaa tai kuviota. AR-sovellus tunnistaa merkatun kohteen ja toistaa sen päälle digitaalisen sisällön.

Merkkaamaton (Markerless) lisätty todellisuus ei tarvitse pohjakseen kuviota, jonka kautta tunnistaa digitaalisen sisällön toistaminen. Käyttäjä itse voi valita mihin kohtaan AR-sisältö laitetaan, tätä varten sovellus käyttää saatavilla olevia laitteen komponentteja, kuten kameraa, GPS:ää tai kiihtyvyysanturia.

Sijaintiperusteisesta (Location-based) lisätystä todellisuudesta hyvä esimerkki on Pókemon Go. Sijaintiperusteinen digitaalinen sisältö toistetaan, kun käyttäjän sijainti vastaa määriteltyä.

Päälle laitettava (Superimposition) AR käyttää objektin tunnistusta tunnistaakseen fyysisen objektin todellisesta maailmasta, minkä jälkeen digitaalinen sisältö voidaan toistaa tunnistetun objektin tilalla kokonaan tai osittain.

Hahmotteleva (Outlining) tekniikka tunnistaa objekteja vahvistaakseen käyttäjän ympäristön tai näkymän rajoja ja muotoja. Tekniikka on hyödyksi esimerkiksi huonossa ajokelissä vahvistamaan ajotien reunan havaittavuutta.

Projisointi (Projection-based) perustuu projektorilla luodun valon heijastamaan AR-sisältöön. Se voi olla interaktiivista sisältöä, kuten näppäimistö, tai valikko, kuten ravintolamenu, tai se voi olla jokin suurelle yleisölle heijastettu efekti.

2.4 Päätelaitteisto

Lisätyn todellisuuden päätelaitteet voidaan jakaa kolmeen ryhmään: käsin pidettävät laitteet, päässä pidettävät laitteet sekä projisoivat laitteet.

2.4.1 Käsin pideltävät laitteet

Mobiililaitteille kehitetyt sovellukset ovat lisätyn todellisuuden yleisin ja nopeimmin kasvava kehitysalue. Syy siihen ei ole laitteiden tehokkuus tai paremmuus toisiin AR-sisältöä näyttäviin laitteisiin verrattuna, vaan ennemmin syy on käytännöllinen: mobiililaitetta kannetaan aina mukana, joten sisällön tuottaminen mobiililaitteille on markkinoiden kannalta järkevää. (Craig, 2013)

Android-mobiililaitteet ovat vuoden 2016 versiosta 7.0 (Nougat) lähtien tukeneet AR-ominaisuuksia ja Applen iOS-laitteet vuoden 2017 käyttöjärjestelmän versiosta 11 lähtien. AR-yhteensopivia malleja on helmikuussa 2021 tarkastetun listan perusteella Androidilla 505 kappaletta ja Applella 22 kappaletta. (Google-Developers, 2020)

2.4.2 Päässä pidettävät laitteet

Lisätyn todellisuuden älylasit yhdistävät virtuaalista sisältöä todelliseen maailmaan kahdella eri tekniikalla: Optisissa laseissa linssit ovat läpinähtäviä ja niihin heijastetaan digitaalinen kerros. Videonäyttöiset lasit sisältävät kameran ja näytön, jonka läpi käyttäjä näkee todellisen maailman ja virtuaalisesti lisätyn sisällön yhdistelmän. (Rolland;Holloway;& Fuchs, 1994)

Augmented Reality Smart Glass (ARSG) eli AR-älylasit, ovat yleisin päässä pidettävistä AR-tuotteiden kategorioista. Näitä on tarkoitus pitää kuin tavallisia laseja, pois lukien tuotteet, kuten Google Glass (Kuva 4), joka voidaan kiinnittää jo käytettyjen lasien kehyksiin. Älylasit voivat sisältää lisätyn todellisuuden tuottamiseen tekniikkaa kuten kameran, GPS:n tai mikrofonin. Tunnettuja valmistajia Googlen lisäksi ovat Microsoft, Everysight, Magic Leap sekä Vuzix. (Ro;Brem;& Rauschnabel, 2017)

Kuva 4. Google Glass (Mikepanhu, 2014)

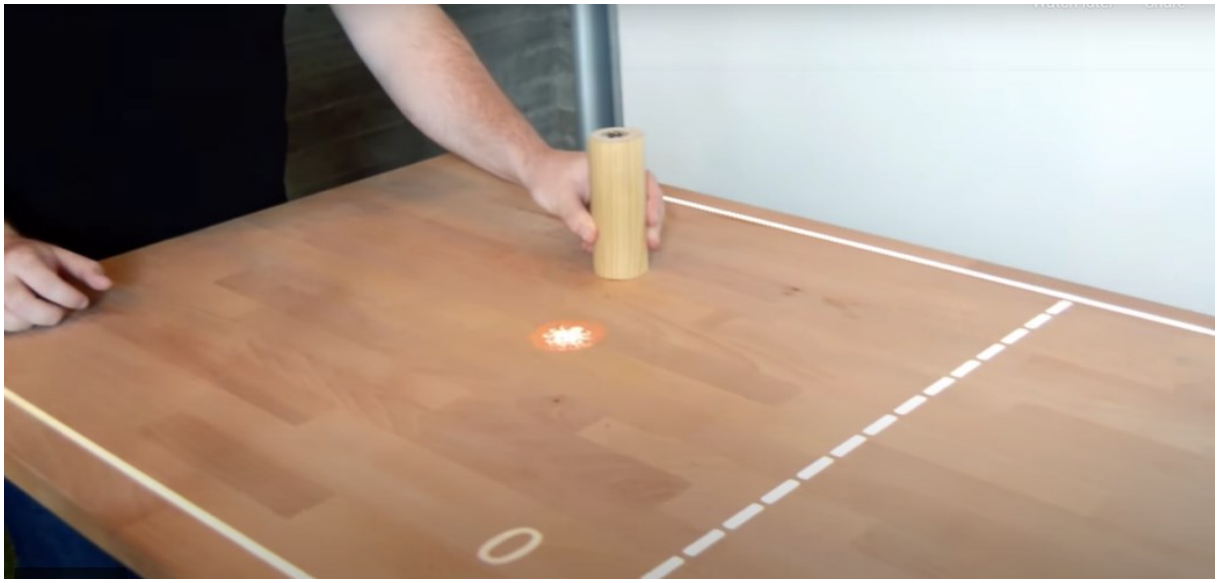
2.4.3 Projisoivat laitteet

Steve Mann kehitti 1990-luvulla Massachusettsin teknillisessä korkeakoulussa (Massachusetts Institute of Technology, MIT) opiskellessaan puettavan laitteen, joka yhdisti eleiden tunnistamista, projektorin ja kameran. Laite tunnettiin myöhemmin vuonna 2009 MIT:n oppilaan Pranav Mistryn jatkokehittämänä nimellä ”SixthSense”. Laitteella voi tunnistaa käyttäjän osoittamia esineitä ja niiden kokoa, ottaa kuvia ja heijastaa niitä halutulle pinnalle, piirtää ilmassa kuvan sormellaan tai esimerkiksi heijastaa tekstiä tai graafista sisältöä kolmiulotteiselle pinnalle, kuten käyttäjän omiin käsiin. (Wearcam, n.d.)

Lightform lähestyy lisätyn todellisuuden teollisuutta uudenlaisesta näkökulmasta, jossa käyttäjien ei tarvitse pitää yllään AR-laitteita, vaan lisätty todellisuus heijastetaan nähtäväksi projektorin avulla. Tekniikka mallintaa ensin kameran ja projektorin avulla heijastettavan pinnan ja laskee pinnan syvyyseroja, minkä jälkeen luotua mallia voidaan muokata tietokoneella ja siihen lisätä valolla tuotettavia efektejä. Lightformin käyttämä tekniikka ei ole reaaliaikaista, sillä heijastettavan pinnan mallintaminen ja efektien lisääminen vie useita minuutteja. Rasitteina tuotteen toimivuudelle ovat huono toistokyky valoisassa ympäristössä, mahdolliset yllättäen ilmestyvät varjot ja prosessointitehon tarve, jos halutaan heijastaa partikkeliefektin kaltaista sisältöä. Tekniikan edut ovat, että lisätty todellisuus on yhtäaikaisesti monen ihmisen nähtävillä ilman henkilökohtaista AR-laitetta, sekä aiemmin hankittujen projektoreiden yhteensopivuus sisällön tuottamiseen. (Turk, 2017)

Tuotesuunnitteluun ja innovointiin perehtynyt yritys Argodesign on kehittänyt projisoitavan lisätyn todellisuuden, jonka kanssa käyttäjä itse voi olla vuorovaikutuksessa. Projekti oli spekulatiivinen ja prototyypin luominen auttoi tekijöitä ymmärtämään, miten valo toimii materiaalina ja kuinka se käyttäytyy heijastettuna ympäristöönsä. He rakensivat projektoria, tietokonetta ja konenäköä hyödyntäen näkyvän interaktiivisen ilmakiekko-pelin (Kuva 5). Peli projisoidaan pöydälle, kiekkona toimii heijastettu valopallo ja lyöntivälineenä voi käyttää mitä tahansa saatavilla olevaa objektia omasta kädestä lähtien. Samaa projisointitekniikkaa voitaisiin soveltaa esimerkiksi interaktiivisen ravintolamenun kehittämiseen. (Chow, 2017)

Kuva 5. Argodesignin AR-projisointia käyttävä ilmakiekkopeli (Argodesign, n.d.)



2.5 Käyttökohteita ja -tarkoituksia

Lisätty todellisuus on tunnettu käsitteenä jo 1990-luvulta, mutta vasta viime vuosikymmenen aikana se on kasvanut ilmiöksi. Mobiililaitteiden yleistyminen ja tekninen kehittyminen on ollut ilmiön mahdollistaja. Seuraavassa listassa esitellään kirjoittajan havaintoihin sekä yleiseen tietämykseen perustuen, miten lisättyä todellisuutta hyödynnetään eri aloilla.

Kaupan ala ja markkinointi

- Mahdollisuus mallintaa ostettavia tuotteita kodin ympäristöön
- Vaatteiden ja meikkituotteiden virtuaalinen sovitus

- Yleisötapahtumien lipunmyynnin yhteydessä näytettävä virtuaalinen katsomonäkymä halutulta istumapaikalta

Lääketiede ja terveysala

- Operaatioiden virtuaalinen opastus, vaativien operaatioiden harjoittelu
- Näkörajoitteisten ihmisten näkökyvyn tehostaminen

Liikenne

- Liikenteen havainnoinnin helpottaminen hahmottelevaa AR-tekniikkaa käyttäen ja pysäköintiavustekameroiden visuaaliset tehosteet
- Mittareiden ja reittiopastusten heijastaminen tuulilasiin

Sotateollisuus

- Pimeänäkö, infrapuna ja muut näköavusteet
- Vaarallisten operaatioiden ja lääkintätehtävien turvallinen koulutussimulointi

Koulutus ja opetusala

- Interaktiivinen opettaminen digitaalisten mallien avulla
- Laitteen osien osoittaminen ja kokoamista helpottavat visuaaliset opasteet

Urheilu ja kulttuuri

- Urheilijan datan, kuten nimen, nopeuden ja kuljetun matkan visualisointi
- Tilastoihin perustuvien ennusteiden näyttäminen urheilulähetyksessä, esimerkkinä jalkapallon rangaistuspotkutilanne: pelaajan aiempien rangaistuspotkujen dataan perustuvan ratkaisun ennustaminen katsojille
- Museoiden ja gallerioiden taideteosten elävöittäminen digitaalisilla elementeillä mobiililaitteen ruudulle

3 Unreal Engine -pelimoottori

Unreal Engine on Epic Games -peliyhtiön kehittämä pelimoottori. Se on tunnettu graafisesti vaikuttavista peleistä ja tarjoaa julkaisutuen usealle eri alustalle. Unreal Enginellä on mahdollista kehittää pelien lisäksi myös viihde- ja virtuaalisisältöä. Unreal Enginen versiota 4 käytetään tässä opinnäytetyössä AR-sovelluksen ohjelmistokehyksenä.

3.1 Yleistä pelimoottoreista

Pelimoottoriksi kutsutaan videopelien kehityksessä käytettävää ohjelmistokehystä. Pelimoottorit voivat erottua toisistaan niiden sisältämien työkalujen, ohjelmointikielen tai käyttöliittymän perusteella, mutta yleisesti pelimoottoreilla on viisi yhteistä elementtiä: pääohjelma, joka sisältää kaiken ohjelmoidun logiikan, renderöintimoottori vastaa animoidusta grafiikasta, fysiikkamoottorilla luodaan objekteille fysikaalinen uskottavuus, tekoäly vastaa pelihahmojen käyttäytymisestä ja äänimoottorilla luodaan pelin äänimaailma. (Interesting Engineering, 2016)

3.2 Unreal Enginen kehityskaari

Vuonna 1998 julkaistiin ensimmäinen Unreal Enginellä kehitetty peli nimeltä Unreal (Kuva 6). Se oli FPS-peli, jonka alustana toimi PC. Unreal Engine -pelimoottori sisälsi useita pelin kehittämiseen hyödyllistä työkalua yhdessä paketissa. Tämä oli vielä vuonna 1998 melko harvinaista. Uniikin pelimoottorista teki sille kehitetty kenttäeditori UnrealEd, sekä oma ohjelmointikieli UnrealScript. (Sutorcen, 2016)

Seuraavan neljän vuoden aikana konsolipelaaminen alkoi yleistyä ja tällöin pelimoottorista julkaistiin uusi versio Unreal Engine 2. Tähän versioon konsolituki kehitettiin ensin XBOX-alustalle ja vuotta myöhemmin myös Nintendolle. (Sutorcen, 2016) Iso kehitysaskel oli myös fysiikkamoottori Karma, joka teki peleistä realistisempia tuottamalla aidompia objektien törmäyksiä ja efektejä (Busby;Parrish;& Wilson, 2009).

Kuva 6. Unrealin Rrajigar Mine -tason latauskuva (me3D31337, 2011)

Vuonna 2004 julkaistiin Unreal Engine 3, jonka konsolituki kattoi kaikki suosituimmat pelikonsolit. Pelimoottorille kehitettiin myös useita graafisia parannuksia DirectX:n, HDR renderöinnin sekä paranneltujen valaistusten ja varjostusten mallintamisen muodoissa. Mobiilikäyttöjärjestelmille iOS ja Android tuki otettiin käyttöön vuonna 2010. Unreal Engine 3:n myötä visuaalinen ohjelmointi alkoi ilmestyä peliohjelmointiin. Järjestelmän nimi oli Kismet ja se oli osa Epic Gamesin julkaisemaa UDK-kehityspakettia. Tämä kehityspaketti mahdollisti ensimmäistä kertaa Unreal Enginen historiassa yksityishenkilöiden myydä itse kehittämiä pelejään ilman, että heidän tarvitsi ostaa kallista lisenssiä. (Sutorcen, 2016)

Kuten aiemmatkin Unreal Engine -versiot, vuonna 2014 julkaistu 4. sukupolven pelimoottori toi myös mukanaan uusia graafisia innovaatioita. Kuva 7 sisältää Final Fantasy VII Remake -pelistä otetun kuvakaappauksen. Pelissä voidaan nähdä pelimoottorin graafisen tason nousu. Suuret muutokset koettiin myös ohjelmointipuolella, kun visuaalinen ohjelmointimenetelmä Kismet korvattiin kehittyneemmällä Blueprint-järjestelmällä (Katso lisää kohdassa 4.1 Blueprint) ja UnrealScriptin tilalle ohjelmointikieleksi otettiin C++. Blueprint-järjestelmällä on mahdollista pelimoottorin käynnissä ollessa testata tai muokata koodia, mikä nopeuttaa ja helpottaa pelin kehitysprosessia. (Sutorcen, 2016)

Kuva 7. Final Fantasy VII Remake (Square Enix, 2021)

Tim Sweeney, Epic Gamesin perustaja ja toimitusjohtaja, esitteli vuonna 2014 pidetyssä Epic GDC -tiedotustilaisuudessa uuden lisensointimallin. Tämä mahdollisti 19 dollarin kuukausimaksulla kehittäjien pääsyn käsiksi Unreal Engine 4 lähdekoodiin, mutta myytävien pelien tuotoista otettaisiin 5-prosentin tekijänoikeusmaksu, jos tuotot ylittäisivät 3000 dollaria vuosineljänneksellä (Nutt, 2014). Tämä lisensointimalli muutti yhtiön ansaintalogiikkaa totaalisesti, mutta toisaalta vapautti pelimoottorin kaikkien kehittäjien saataville. Sittemmin Epic Games on poistanut 19 dollarin kuukausikulun vuonna 2015 (Sweeney, 2015). Lisenssimaksuja on kevennetty useaan otteeseen, joista viimeisin on vuodelta 2020: yksityisissä ja ilmaistuotannoissa lisenssi on ilmainen, mutta kaupallisessa tuotannossa sisältyy 5-prosentin tekijänoikeusmaksu, kun tuotetta on myyty tuotteen eliniän aikana yli miljoonalla dollarilla (Unreal Engine, n.d.-a).

3.3 Liitännäiset ja Marketplace

Liitännäiset ovat tärkeä osa kehitystyötä Unreal Enginessä. Ne ovat data- tai koodikokoelmia, joita kehittäjät voivat liittää projekteihinsa tuomaan mukaan lisäominaisuuksia ja -toiminnallisuuksia. Pelimoottori tarjoaa oletuksena huomattavan määrän liitännäisiä kirjastossaan, mutta kehittäjät voivat myös itse luoda niitä tarpeidensa mukaan. Liitännäiset

ovat helposti liitettävissä projekteihin pelimoottorin käyttöliittymän kautta. Kehittäjän t053kr (2020) GitHubiin jakamassa projektissa ”Ambient LED via BT” tietoliikenteen välittäminen on tehty käyttämällä kolmannen osapuolen kehittämää liitännäistä, jolla voidaan liittää projektiin mukaan COM-porttia käyttävä Bluetooth- tai USB-laite. (Unreal Engine, n.d.-b)

Unreal Engine Marketplace on pelimoottorin verkkokauppatyyppinen lisäosalataamo. Siellä myydään liitännäisiä, ääniraitoja, animaatioita sekä pelihahmoja ja -ympäristöjä. Näiden lisäksi siellä tarjotaan vapaasti ladattavaa ilmaista sisältöä. Sisältöä Marketplaceen voi tuottaa Epic Gamesin lisäksi kolmannen osapuolen kehittäjät. (Denham, n.d.)

3.4 Vertailukohteena Unity3d

Toftedahl ja Engström (2019) julkaisivat tutkimuksen, jossa he vertailivat pelimoottoreiden suosiota vuonna 2018 itch.io ja Steam -videopelinjakelupalveluissa. Tästä tutkimuksesta kävi ilmi, että itch.io-videopelipalvelussa jaossa olevista peleistä 47,3-prosenttia oli kehitetty Unitylla ja ainoastaan 2,8-prosenttia Unreal Enginellä. Steam-palvelussa vastaavat luvut olivat 13,2-prosenttia Unitylla ja 25,6-prosenttia Unreal Enginellä kehitettyjä. Huomionarvoista tilastossa on, että Steamin osalta 86,3-prosentille peleistä ei ole merkitty kehityksessä käytettyä pelimoottoria.

Unity ja Unreal Engine ovat yleisimmät pelimoottorit pelituotannossa, mutta paremmuusvertailun tekeminen näiden kahden välillä on vaikeaa. Molemmilla pelimoottoreilla on pystytty luomaan huippupelejä: Unitylla Pokémon Go ja Unreal Enginellä Fortnite. Taulukko 1 sisältää keskeisimmät Unreal Enginen ja Unityn ominaisuudet. Pelimoottoreiden suurimmat erot löytyvät ohjelmointikielen ja käyttöliittymän muodossa. Siinä missä aloittelijan on helpompi aloittaa Unitylla ohjelmoiminen selkeämmän käyttöliittymän ja helpomman ohjelmointikielen vuoksi, pystyy Unreal Engine tuottamaan parempilaatuisen grafiikan. Molemmilla voidaan ohjelmoida myös visuaalisella ohjelmointikielellä ja tuottaa virtuaalisesti luotua todellisuutta hyödyntäviä sovelluksia. Lisätyn todellisuuden sovelluksissa Unitylla kuitenkin on laajempi lisäosatuki ja dokumentaatio kuin Unreal Enginellä. (Petty, n.d.)

Molemmat pelimoottorit tarjoavat ilmaisen version, mutta tammikuussa 2021 tarkastetun tilanteen perusteella eroja löytyy kaupallisen tuotteen lisensseistä. Unreal Enginellä kehitetyn tuotteen myynnin ylittäessä sen eliniän aikana miljoona dollaria, aletaan maksaa sen ylittävästä osuudesta 5-prosentin suuruista lisenssimaksua. (Unreal Engine, n.d.-a). Unity-kehittäjän on ostettava maksullinen versio, jos tuotteen myynti ylittää 100 000 dollaria. Halvin maksullinen versio maksaa 399 dollaria vuodessa, mutta itse tuotteesta ei jouduta maksamaan tekijänoikeusmaksua. (Unity, n.d.-a)

Taulukko 1. Unreal Engine ja Unity3d -pelimoottoreiden vertailu

	Unreal Engine	**Unity3D**
Ohjelmointikieli	C++	C#
Visuaalinen ohjelmointi	Blueprint	Bolt
Lisätyn todellisuuden tuki	✓	✓
Tuettu käyttöjärjestelmä	Windows, macOS, Linux	Windows, macOS, Linux
Kohdelaitteet	20 tuettua laitetta: yleisimmät konsolit, tietokoneet ja mobiililaitteet sekä AR/VR	28 tuettua laitetta: yleisimmät konsolit, tietokoneet ja mobiililaitteet sekä AR/VR
Ilmainen versio	✓	✓

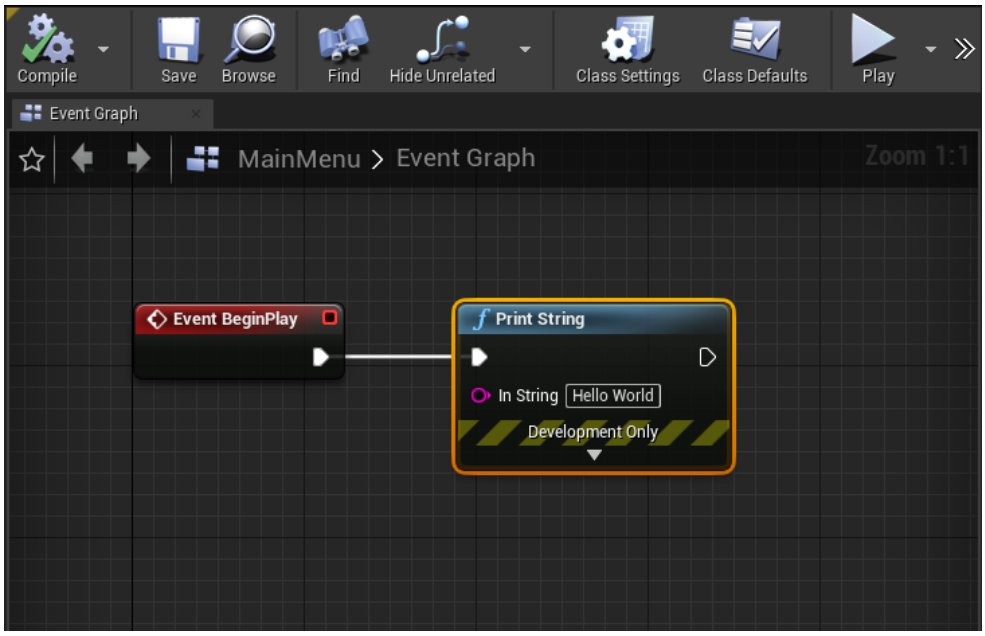
4 Visuaalinen ohjelmointi pelimoottoreilla

Visuaalisessa ohjelmoinnissa ohjelmointilogiikkaa luodaan ikoneita, lohkoja tai muita graafisia muotoja tai kuvioita yhdistellen. Yhteisenä pyrkimyksenä visuaalisilla ohjelmointimenetelmillä on ohjelmointiprosessin yksinkertaistaminen, nopeuttaminen sekä syntaksivirheiden vähentäminen. Visuaalinen ohjelmointi laajentaa pelin kehittämisen mahdolliseksi sellaisillekin kehittäjille, jotka eivät osaa lainkaan tekstipohjaisia ohjelmointikieliä. Tässä luvussa käsitellään visuaalista ohjelmointia pelimoottoreilla, joista keskeisenä on Unreal Engine Blueprint.

4.1 Blueprint

Blueprint on Unreal Engine -pelimoottorissa käytettävä visuaalinen ohjelmointiympäristö. Blueprint-järjestelmä kehitettiin vuonna 2014 julkaistuun Unreal Enginen versioon 4. Se korvasi aiemmin käytössä olleen Kismetin. Käyttöliittymältään Blueprint on samantapainen kuin edeltäjänsä. Ohjelmointi tapahtuu yhdistäen toisiinsa lohkoja (engl. node). Kun kehittäjä lohkoja yhdistelemällä luo ohjelmalle logiikkaa, taustalla ohjelmisto kirjoittaa kaiken C++-ohjelmointikielellä. Unreal Enginessä kehittäjän ei tarvitse osata lainkaan ohjelmointia, sillä peli on mahdollista ohjelmoida kokonaisuudessaan valmiiksi käyttämällä ainoastaan Blueprint-järjestelmää. Unreal Engine mahdollistaa myös C++-kielisen tekstipohjaisen ohjelmoinnin ja visuaalisen ohjelmoinnin yhdistämisen. Kuva 8 näyttää esimerkin Blueprintin käytöstä. Kun taso käynnistyy, ajetaan Event BeginPlay -tapahtuma, joka tulostaa ruutuun Print String -funktiota käyttäen tekstin Hello World. (Game-Ace, 2019)

Kuva 8. Blueprint-ohjelmoinnin Hello World -esimerkki



Blueprint-järjestelmä sisältää viisi erilaista Blueprint-tyyppiä: (Unreal Engine, n.d.-c)

1. **Level Blueprint**, tämä tyyppi vastaa Unreal Engine 3 käyttämää Kismetiä. Level Blueprintiin luodaan tasolle ominainen toiminnallisuus. Tällainen voisi olla tietyn videoklipin käynnistyminen tai tallennuspisteen aktivointi.
2. **Blueprint Class** tuo esiin olio-ohjelmoinnin hyödyt. Tämä on yleisin tapa käyttää Blueprintejä. Uudelle Blueprint-luokalle valitaan aina ensin yläluokka (Parent Class), jonka toiminnallisuudet ja ominaisuudet uusi luokka perii.
3. **Data-Only Blueprint** sisältää vain yläluokalta perityt muuttujat ja komponentit. Näitä perittyjä ominaisuuksia voidaan muokata, mutta uusia elementtejä ei voi lisätä tähän luokkatyyppiin.
4. **Blueprint Interface** on rajapintaluokka, johon määritellään funktioita nimellisesti ilman toteutusta. Tämä rajapinta voidaan lisätä eri Blueprint-luokille, joiden halutaan käyttävän yhteistä rajapintaa ja siihen määriteltyjä funktioita tiedon jakamiseen keskenään.
5. **Blueprint Macro Library**, tänne voidaan luoda projektille yhteisiä Macroja. Macro toimii kuin funktio: sille syötetään tietoa → Macroon luotu toiminnallisuus käyttää tietoa → tuotos on valmis hyödynnettäväksi

jatkokäyttöä varten. Macroa voidaan hyödyntää projektin eri Blueprinteissä, jolloin ei tarvitse useaan kertaan luoda samaa toiminnallisuutta.

4.2 Blueprintin käytön hyvät ja huonot puolet

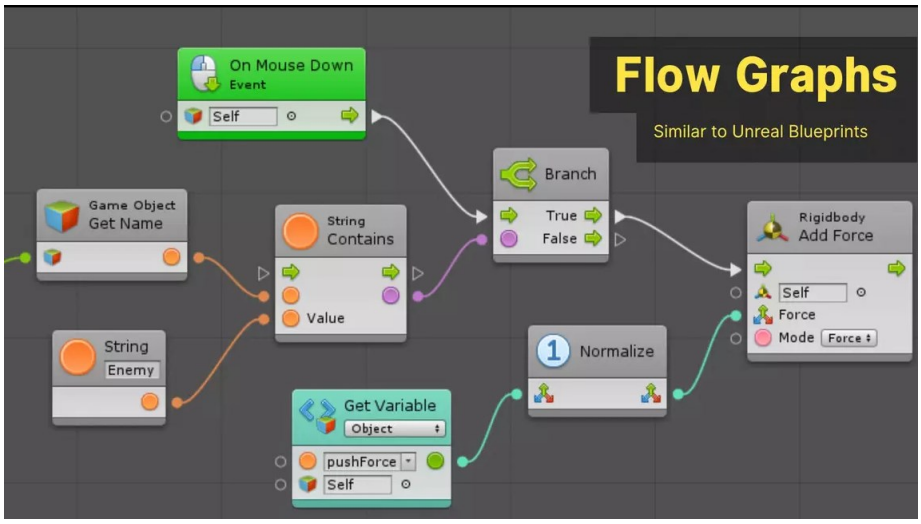
Ohjelmoiminen visuaalisella käyttöliittymällä parhaassa tapauksessa nopeuttaa pelin kehittämistä ja laajentaa sen mahdolliseksi myös henkilöille, jotka eivät osaa kirjoittaa ohjelmakoodia. Blueprintit soveltuvat parhaiten tapahtumakeskeiseen ohjelmointiin. Tapahtumakeskeinen ohjelmointi tarkoittaa sellaisen toiminnallisuuden luomista, joka syntyy jostain tapahtumasta, esimerkiksi hiiren klikkauksesta tai objektiin liitetyn colliderin kosketuksesta toiseen collideriin. Jos peliin ohjelmoidaan useita kertoja sekunnissa kutsuttavia toimintoja ja vaativia laskennallisia tehtäviä, on niiden tuottaminen Blueprint-järjestelmällä hitaampaa kuin tekstipohjaisesti ohjelmoituna. Paras ratkaisu on tässä tapauksessa käyttää pelimoottorin tekstipohjaista ohjelmointikieltä, joka Unreal Enginessä on C++. (Unreal Engine, n.d.-d)

4.3 Muiden pelimoottoreiden graafisia ohjelmointikieliä

Unreal Engine ei ole ainoa pelimoottori, joka mahdollistaa visuaalisen ohjelmoinnin käytön pelikehityksessä. Myös useat muut pelimoottorit, kuten esimerkiksi Unity, Godot ja GameMaker Studio 2, mahdollistavat sisällön ohjelmoimisen visuaalista menetelmää käyttäen.

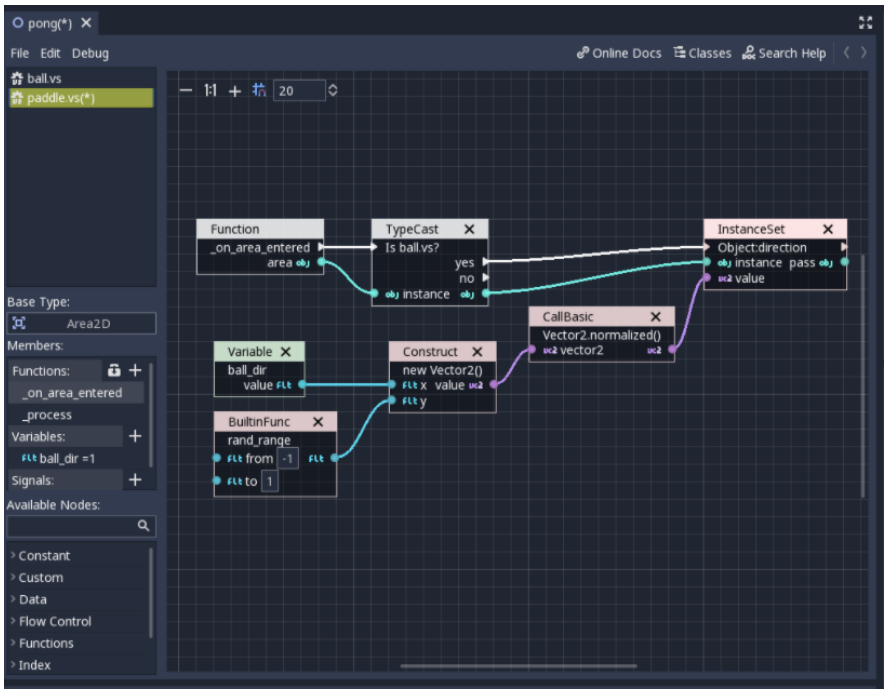
Unityn visuaalinen ohjelmointikieli on Lazlo Boninin kehittämä Bolt. Se julkaistiin vuonna 2017 erikseen ostettavana lisäosana, mutta Unity hankki sen itselleen vuonna 2020, minkä jälkeen se on sisältynyt myös ilmaiseen versioon (Unity Asset Store, n.d.-a). Boltin käyttöliittymässä on samankaltaisuuksia Unreal Enginen Blueprintin kanssa (Kuva 9). Unityyn on saatavilla Boltin lisäksi muitakin visuaalisia ohjelmointityökaluja, kuten PlayMaker ja Adventure Creator. (Unity, n.d.-b)

Kuva 9. Unity Bolt -käyttöliittymä (Unity Asset Store, n.d.-b)



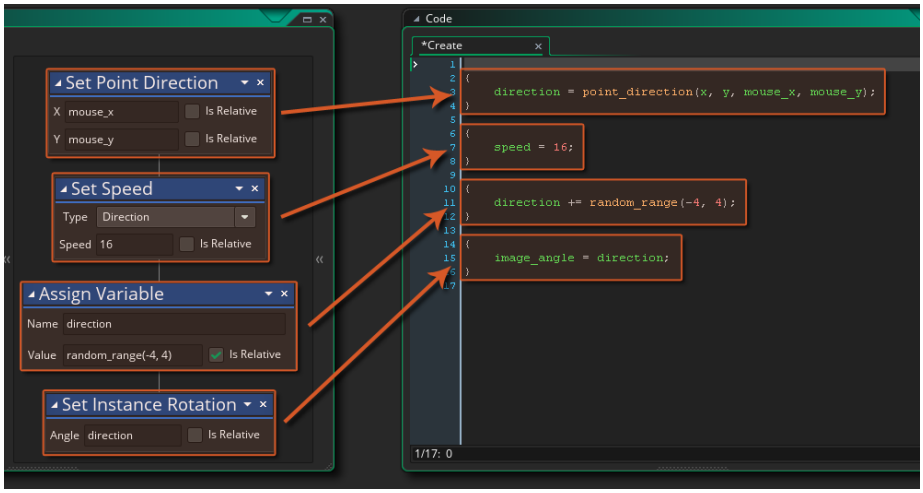
Godot-pelimoottorin kehitys aloitettiin vuonna 2007 ja se julkaistiin lopulta vuonna 2014 avoimella lähdekoodilla (Machado, 2017). Godotilla voidaan tehdä 2D- ja 3D-pelejä ja vuonna 2018 julkaistusta 3.0 versiosta lähtien sillä on voinut ohjelmoida C#:n lisäksi myös visuaalisella ohjelmointikielellä Visual Script (Kuva 10). Godotin visuaalinen ohjelmointi perustuu vahvasti funktioiden käyttöön. Yksi skripti voi sisältää useita funktioita, mutta jokainen funktio kirjoitetaan omaan kanvaasiinsa. (Godot community, 2014/2020)

Kuva 10. Godot-pelimoottorin Visual Script -editori (Godot community, 2014/2020)



YoYo Gamesin kehittämä GameMaker Studio 2 on pääasiassa 2D-pelien kehitykseen tarkoitettu pelimoottori. Se käyttää Drag and Drop (DnD) -nimistä visuaalista ohjelmointityökalua. Tekstipohjainen ohjelmointi on mahdollista käyttämällä GameMaker Languagea, jonka syntaksi on muotoiltu JavaScriptin ja C-ohjelmointikielien tapaiseksi. Ohjelmointia Drag and Dropilla on luonnehdittu helpoksi ja aloittelija pystyy kehittämään sillä kokonaisen pelin. Kuva 11 osoittaa, kuinka editorin sisältämällä Live Preview -tilalla voidaan reaaliajassa nähdä, miten editori luo taustalla toiminnallisuudet tekstipohjaisena ohjelmointikielenä. (GameMaker Studio 2, 2019)

Kuva 11. GameMaker Studion Live Preview -näkymä (GameMaker Studio 2, 2019)



.

5 Google ARCore

ARCore on Googlen kehitysalusta, joka tarjoaa ohjelmointirajapintoja (Application Programming Interface, API) lisätyn todellisuuden sovellusten kehittämiseen mobiililaitteille. Tuettuja mobiililaitteita ovat Android-laitteet 7.0-versiosta alkaen ja iOS-laitteet käyttöjärjestelmän versiosta 11 lähtien.

5.1 Toiminta

ARCore käyttää mobiililaitteen sensoreita, kuten kiihtyvyysanturia ja gyroskooppia havainnoidakseen puhelimen asentoa ja liikettä. Puhelimen kameralla tunnistetaan ympäristöstä visuaalisesti erottuvia kohtia. Yhdistelemällä näitä tietoja laite voi arvioida sijoittautumista ympäristöönsä. Tällaista havainnointitekniikkaa kutsutaan nimellä SLAM (Simultaneous Localization and Mapping) eli samanaikainen paikannus ja kartoitus. ARCore luo kuvaamastaan ympäristöstä samanaikaisesti pistekarttaa, jolla se havainnoi vertikaalisia ja horisontaalisia tasoja. Näille tasoille on mahdollista liittää AR-sovelluksissa digitaalisia objekteja. (Google ARCore, n.d.-a)

Digitaaliset objektit saattavat helposti erottua AR-sovelluksessa avautuvassa näkymässä sinne kuulumattomina, joten ARCore käyttää erilaisia valaistuslaskelmia sulauttaakseen objektit ympäristöönsä paremmin. Valaistuslaskelmat muodostuvat tilan ympäristövalaistuksen, valoheijastusten ja varjojen yhtälöstä. Tämän lisäksi sovellus laskee kameran kuvaamaan RGB-kuvan perusteella syvyyskartan, jotta se osaisi asettaa lisättävät objektit oikealle etäisyydelleen. (Google ARCore, n.d.-b)

5.2 Mahdollisuudet

ARCore tarjoaa ohjelmointirajapintoja Android, iOS, Unity ja Unreal Engine -kehitysympäristöihin. ARCorella voi luoda lisättyä todellisuutta kuvien tai kasvojen tunnistamisen yhteydessä, liittää virtuaalisia malleja havainnoiduille tasoille ja käyttää pilviankkureita. Pilviankkureiden avulla voidaan luoda yhteiskäytettävää AR-sisältöä useiden käyttäjien kesken. Tällöin yksi käyttäjä toimii tapahtuman isäntänä ja muut pääsevät liittymään mukaan sessioon jaetun ID-tunnisteen avulla. (Google ARCore, n.d.-c)

6 AR-sovelluksen kehittämistyön tavoitteet ja tarkoitus

Opinnäytetyön tavoitteena oli lisätyn todellisuuden mobiilisovelluksen kehittäminen. Ohjelmistokehyksenä käytettiin Unreal Engineä ja ohjelmointikielenä visuaalista Blueprint-ohjelmointia. AR-sovellus kehitettiin Android-laitteelle ja ohjelmointirajapintana hyödynnettiin Google ARCore -kehitysalustaa. Käyttöjärjestelmä kehitysympäristössä oli Windows.

6.1 Työn kuvaus

Kehitysprojektin tuotteena syntyi Android-mobiililaitteelle AR-sovellus, joka hyödyntää päälle laitettavaa lisättyä todellisuutta (Superimposition-based AR). Tämä tarkoittaa sitä, että sovelluksen pitää pystyä tunnistamaan ennalta määritelty objekti kameralla kuvatusta ympäristöstä ja toistamaan digitaalinen sisältö tämän tilalla. Tähän tarkoitukseen ARCore sisältää Unreal Enginelle ohjelmointirajapinnan, jota käyttämällä sovellus voi tunnistaa ennalta määritetyn 2D-kuvan kaltaista sisältöä näkemästään ympäristöstä. Sovellukselle osoitetaan kuvien lähdekirjasto tai tietokanta, joista konenäköalgoritmeja käyttämällä ARCore muodostaa harmaasävykuvia. Sovellus vertaa harmaasävykuvia mobiililaitteen kameran kuvaamaan ympäristöön ja tunnistaessaan määritellyt objektit se toistaa digitaalisen sisällön. Kehitysprojektin aikana kerättiin päiväkirjamaisesti muistiinpanoja ylös ja prosessin eri vaiheet kuvataan raportissa niin, että prosessin kulku voidaan hahmottaa ja saadaan tarvittava käsitys AR-sovelluksen kehittämisestä Unreal Enginellä.

6.2 Ohjelmistojen valinta

Unreal Engine on tunnettu graafisesti näyttävistä peleistä. Pelien kehittämisen lisäksi sillä voidaan luoda arkkitehtuurimalleja, videotuotantoja, simulaatioita, virtuaalista sisältöä, sekä kehittää sovelluksia. Tunnetuin pelimoottori virtuaalisen sisällön luomiseen on Unity, mutta Unreal Engine valikoitui tässä työssä käytettäväksi ohjelmistokehykseksi, koska mobiilisovellus ohjelmoitiin visuaalisesti ja opinnäytetyön tekemisen ajankohtana kehittynein saatavilla oleva visuaalinen ohjelmointimenetelmä oli Blueprint. Toinen syy valinnalle oli dokumentoitujen AR-projektien vähyys Unreal Enginellä. Moni kehittäjä valitsee käytettävät ohjelmistot ominaisuuksien ja saatavilla olevan dokumentoinnin

perusteella. Tämän raportin kirjoittaminen antaa lisäarvoa kehittäjille, jotka etsivät vaihtoehtoa Unitylle. Visuaalinen ohjelmointi päätettiin ottaa ohjelmointitekniikaksi kehittäjän mielenkiinnosta testata, kuinka se soveltuu sovelluksen ohjelmoimiseen. Unreal Enginessä käytettävä liitännäinen Google ARCore valikoitui sen Android-kehittämiseen tarjoamien ohjelmointirajapintojen vuoksi.

6.3 Kehitystyön tiedonhaku

Työ aloitettiin etsimällä tietoa käytettävistä tekniikoista. Käytännöllisimmät lähteet olivat kehittäjien ylläpitämät dokumentaatiot ja asennusoppaat. Tämän raportin tietoperustan pyrkimys on koostaa tarvittava pohjatieto käytettävistä tekniikoista näihin lähteisiin perustuen. Tietoperustan avulla saavutetaan yleinen käsitys aihepiiristä ja voidaan ymmärtää kokonaiskuva kehitystyön tavoitteesta.

Aihepiiriin tutustumisen jälkeen kartoitettiin yhteensopivat versiot ja lisenssit käytettävistä ohjelmistoista. Tärkeää on huomata, että kaikista käytettävistä ohjelmistoista ei välttämättä voida käyttää uusinta päivitettyä versiota, vaan tulee löytää versiot, jotka ovat yhteensopivia keskenään ja oleellisilta ominaisuuksiltaan tuettuja. Hyvin tehdyllä versioiden kartoitustyöllä yritettiin välttää yllättäen ilmaantuvat yhteensopivuusongelmat, jotka pahimmillaan johtaisivat projektin uudelleen rakentamiseen.

6.4 Kehitysympäristön asennukset

Kirjottamisen hetkellä valmistajien asennusoppaat olivat ristiriitaisia. ARCoren (n.d.-d) dokumentoinnin perusteella sen käyttöön ottamiseksi täytyi Unreal Enginen lähdekoodi kopioida GitHubista, minkä jälkeen koostaa versio käyttäen Microsoft Visual Studiota. Tämä oli kuitenkin päivittämätöntä tietoa, sillä se koski Unreal Enginen vanhempia versioita. Toimiva ARCore-kehitysympäristö Androidille saatiin asennettua, kun seurattiin Unreal Enginen (n.d.-e) dokumentaatiota Android-kehitysympäristön asentamisesta.

6.4.1 Unreal Enginen asentaminen

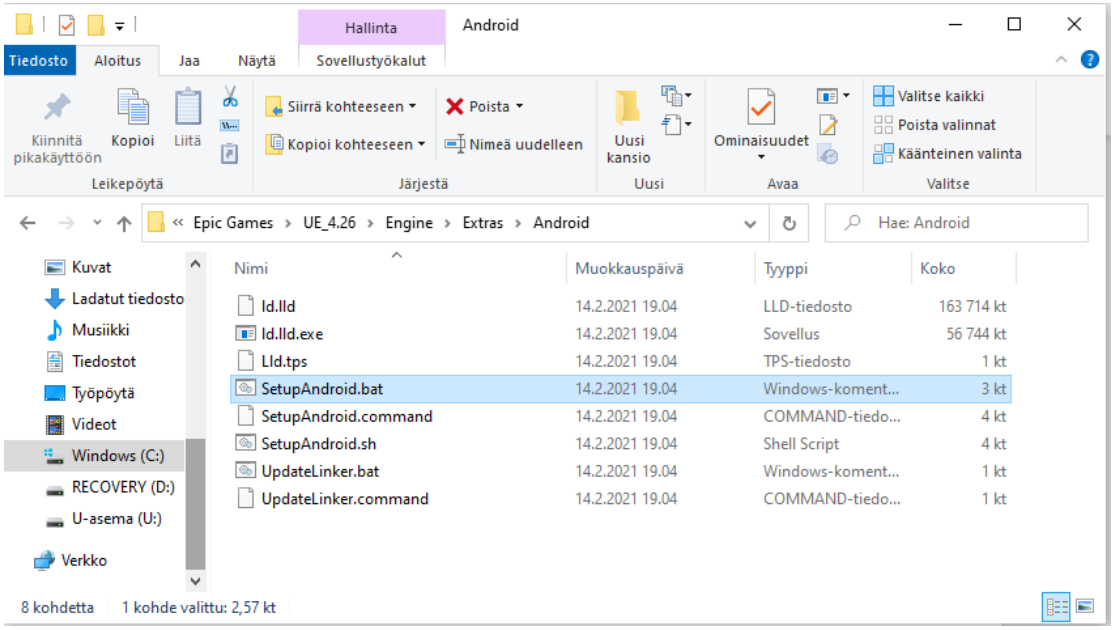
Tässä työssä käytettiin Unreal Enginestä uusinta versiota 4.26.1. Unreal Enginen asennus aloitettiin luomalla tunnukset Epic Games -sivustolle, minkä jälkeen asennettiin sivuston tarjoama Epic Games Launcher. Tämän sovelluksen avulla voitiin asentaa Unreal Engine -pelimoottori. Pelimoottori asennettiin Library-välilehdeltä kohdasta Engine Versions. Asennusta käynnistettäessä oli syytä tarkistaa, että myös Android-kehityksen tuki oli asennuspaketissa valittuna.

6.4.2 Android Studion asentaminen

Android Studio on Android-käyttöjärjestelmän virallinen ohjelmointiympäristö. Android Studiosta käytettiin versiota 3.5.3. Ohjelmistoa asennettaessa tuli Epic Games Launcher ja Unreal Editor olla suljettuina, sillä asennuksen aikana täytyi olla vapaa käyttö sen resursseihin. Myös Windowsin tehtäväpalkki oli huomioitava, sillä ilman erillistä ilmoitusta Epic Games Launcher sijoittautui sinne ohjelmiston sulkemisen jälkeen. Ohjelmisto asennettiin oletussijaintiin, jotta Android NDK (Native Development Kit) asentaminen olisi myöhemmässä vaiheessa helppoa. Kun ohjelmisto oli asentunut, se ehdotti päivityksen asentamista, mutta tätä ei kannattanut suorittaa, sillä toimiakseen Unreal Enginen kanssa oikein, tarvitsi dokumentaation mukaan Android Studio version olla 3.5.3.

Android Studion asentumisen ja järjestelmän uudelleenkäynnistymisen jälkeen asennettiin Android NDK:n versio r21 b. Koska Unreal Engine asennettiin oletussijaintiin, löytyi kansiopolusta ”C:\Program Files\Epic Games\UE_4.26\Engine\Extras\Android” SetupAndroid.bat-tiedosto, jonka ajamalla Android NDK asentui (Kuva 12). Asennuksen alkamiseksi oli hyväksyttävä ehdotus Android-lisenssin käyttöönotosta.

Kuva 12. Android NDK -asennustiedoston sijainti



6.4.3 Kohdelaitteen valmistelu

Kehitystyön kohdelaitteena voi käyttää vain AR-ominaisuuksia tukevaa Android-mobiililaitetta. Laitteeseen ei enää nykyisin tarvitse asentaa erillistä AR-sovellusta Google Play -kaupasta, vaan siihen on puhelimen käyttöönottovaiheessa automaattisesti asentunut Google Play Services for AR, jos laite kuuluu tuen piiriin.

Kohdelaitteesta eli Android-mobiililaitteesta tuli hyväksyä USB-vianetsintä. Tämä asetus löytyi, kun mobiililaitteen asetuksista etsittiin osio ”Kehittäjäasetukset”, ja hyväksyttiin USB-vianetsintä.

Nokia HMD Global: Asetukset → Järjestelmä → Lisäasetukset → Kehittäjäasetukset

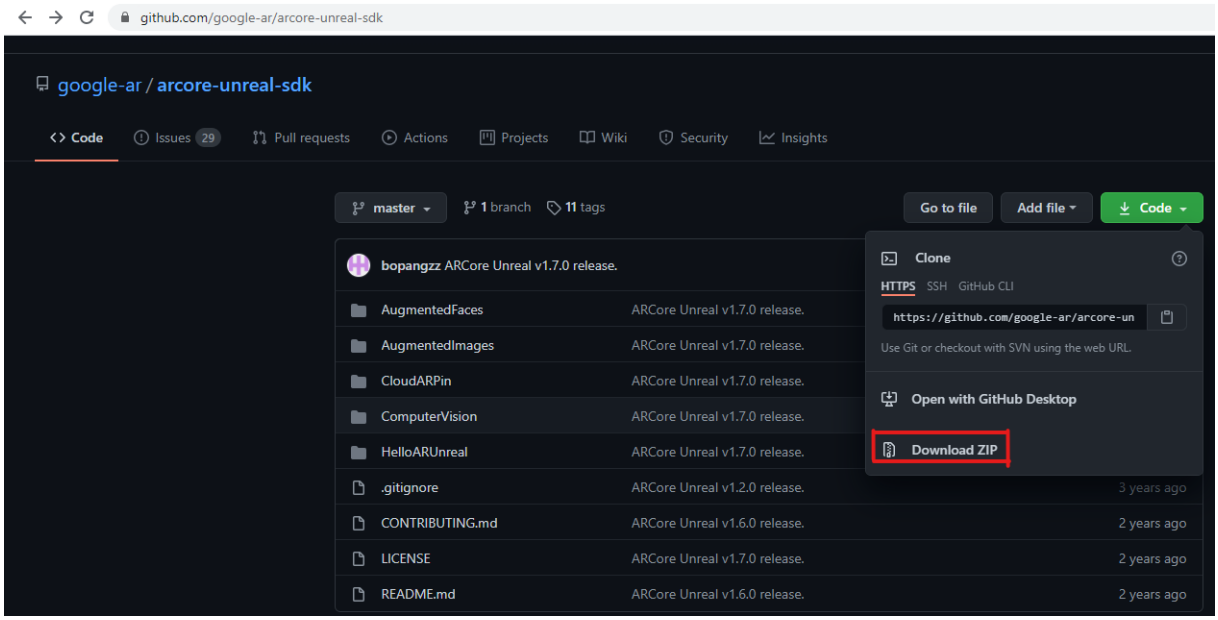
6.4.4 Kehitysympäristön testaus

Kehitysympäristön ohjelmistot olivat nyt asennettuina ja seuraavaksi testattiin niiden toimivuus käyttäen Googlen ARCore-testiprojektia. Google oli julkaissut GitHub-repositorion ”arcore-unreal-sdk” (Google, 2019), joka sisälsi tässä käytettävän HelloARUnreal-testiohjelman. Tällä testiohjelmalla pyrittiin varmistamaan, että ympäristö oli

oikein rakentunut, laitteet olivat keskenään yhteensopivia ja onnistuivat toistamaan AR-sisältöä.

1. Ladattiin Googlen GitHub-sivulta arcore-unreal-sdk-repositorio. Testikäyttöä varten se voitiin ladata zip-tiedostona (Kuva 13). Ladatun tiedoston nimi oli arcore-unreal-sdk-master.zip, tämä tiedosto purettiin samaan kansioon.

Kuva 13. ARCore testiohjelman lataaminen GitHubista



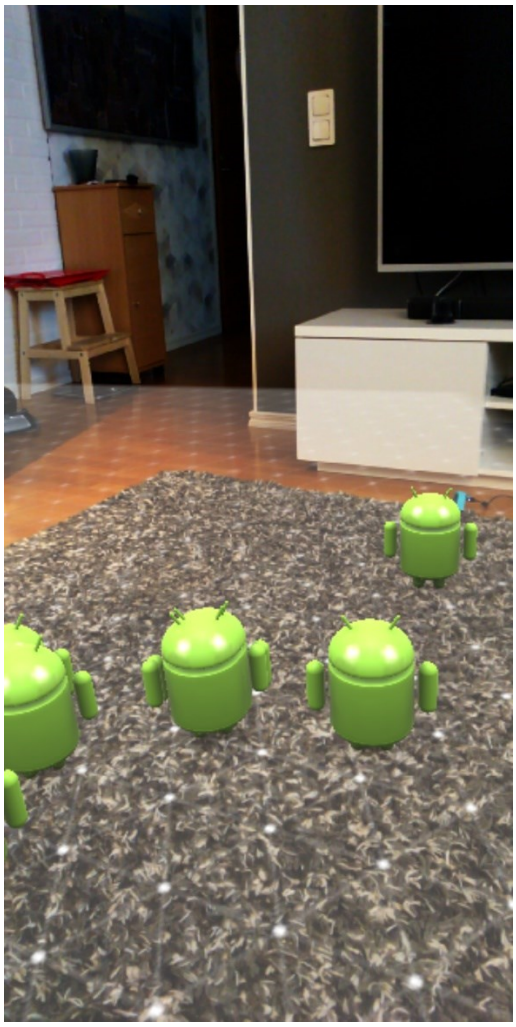
2. Käynnistettiin Unreal Engine ja etsittiin äsken purettu arcore-unreal-sdk-master-kansio. Tämän kansion sisältä avattiin HelloARUnreal-projekti ja valittiin HelloARUnreal.uproject-tiedosto. Koska testiprojekti oli luotu käyttäen vanhempaa ohjelmistoversiota, ruutuun tuli kehotus käynnistää projekti kopiona.

3. Yhdistettiin käytettävä Android-laite USB-kaapelilla tietokoneeseen ja valittiin Unreal Editorin työkaluriviltä oikeasta reunasta nuoli, joka avasi valikon ”Options for launching on a device”. Tästä valikosta valittiin käytettävä Android-laite, minkä jälkeen ohjelmisto alkoi valmistella sovelluksen käynnistämistä Android-laitteella.

4. Projektin koostaminen kesti ensimmäisellä kerralla pitkään. Kun koostaminen oli valmis, testiohjelma käynnistyi Android-laitteella ja sovellukselle oli annettava oikeus

käyttää laitteen kameraa. Sovelluksen käynnistyttyä se alkoi kameran perusteella etsiä tasoja, joille käyttäjä voisi asettaa virtuaalisen Android-mallin. Kuva 14 näyttää miten tunnistettu taso näkyy haaleana valkoisena pisteverkkona ja koskettamalla ruutua voitiin liittää vihreä Android-malli livenäkymään.

Kuva 14. HelloARUnreal-projektin testaus mobiililaitteella



7 AR-sovelluksen kehittämistyön toteutus

Kehitysympäristön asentamisen ja testauksen jälkeen aloitettiin sovelluksen kehittäminen. Tässä luvussa on kuvattuna sovelluksen kehitysprosessi. Kuva 15 havainnollistaa kaavion avulla, kuinka sovelluksen kehitysprosessi voidaan jakaa viiteen osaan: määrittely, suunnittelu, kehitystyö, testaus ja käyttöönotto. Tähän opinnäytetyöhön rinnastettuna määritelmävaihe on kappaleeseen 6.1 kirjoitettu työn kuvaaminen. Kappale 7.1 pitää sisällään sovelluksen suunnittelun Figmalla. Suunnittelun jälkeen aloitettiin sovelluksen kehittäminen Unreal Enginellä, jolla pystyi yhdessä Android-laitteen kanssa suorittamaan myös testaamisen. Viimeinen vaihe oli sovelluksen käyttöönotto, jossa julkaistiin Android-mobiililaitteille jaettavissa oleva paketti. Työ ei ole välttämättä valmis vielä julkaisunkaan jälkeen, sillä sovelluksen käyttäjiltä voidaan saada käyttäjäpalautteen perusteella selville aiemmin testivaiheessa huomaamaton virhe, joka on korjattava. Jos sovellus julkaistaan myöhemmin esimerkiksi Google Play -kaupassa, on julkaisija Googlen kehittäjille suunnattujen käyttöehtojen mukaan velvollinen antamaan käyttäjätukea ja korjaamaan reklamoidut sovelluksen käyttöä estävät tai vaikeuttavat puutteet.

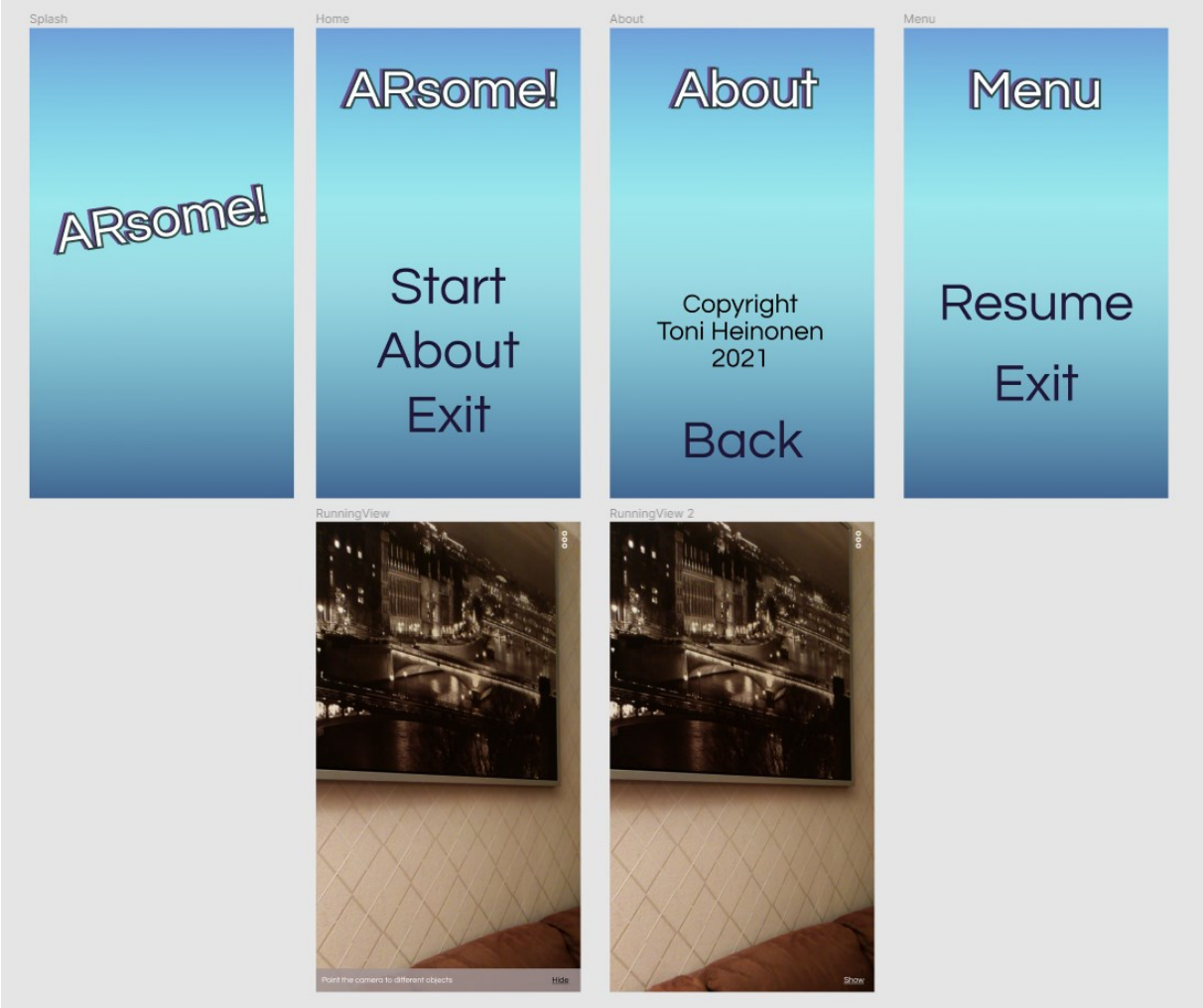
Kuva 15. Sovelluksen kehitysprosessi kaaviona kuvattuna

7.1 Sovelluksen suunnittelu

Sovelluksen ulkoasun suunnitteluun käytettiin web-pohjaista työkalua Figmaa. Figma on pääosin ilmainen, mutta sisältää erikseen ostettavia lisäominaisuuksia. Figmalla on mahdollista tuottaa laadukkaita suunnitelmia käyttöliittymistä sekä suorittaa niiden toiminnan testaamista.

Opinnäytetyössä kehitetyn sovelluksen työnimi oli ARsome ja pääväriksi valittiin sininen. Suunniteltavia käyttöliittymäkuvia olivat splash, aloitusvalikko, tietosivu, menuvalikko sekä kuvat, jotka havainnollistivat sovelluksen toimintaa käytössä (Kuva 16).

Kuva 16. Figmalla luotu käyttöliittymäsuunnitelma



Splash-näyttö: Tämä kuva näkyy ruudulla noin parin sekunnin ajan, kun sovellus käynnistetään. Tavallisesti splash on selkeä taustaltaan ja sisältää sovelluksen nimen ja/tai logon. Splash on yhtenäinen muun sovelluksen värimaailman ja tyylisuunnan kanssa ja se johdattelee käyttäjän kohti sovelluksen käyttöä antamalla visuaalisia vihjeitä tulevasta käyttökokemuksesta.

Aloitus- ja menuvalikko sekä tietosivu rakennettiin käyttäen samaa sapluunaa, jotta lopputuloksesta tulisi yhdenmukainen. Ruudun yläosassa näytetään sivun otsikko ja alaosassa valintapainikkeet, tausta on sama kuin splash-kuvassa. Aloitusvalikko on näkymä,

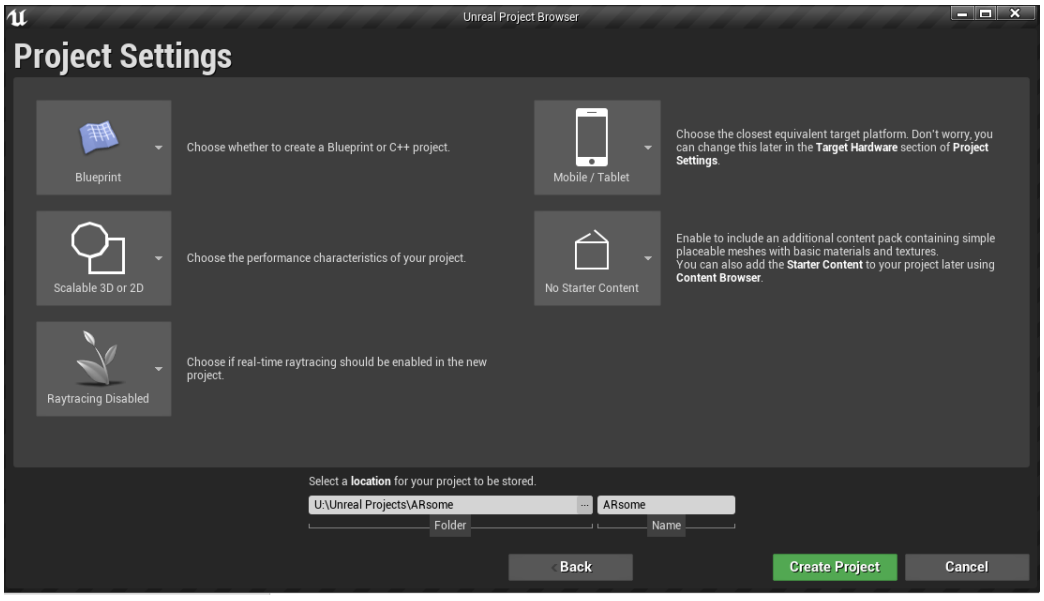
joka näytetään käyttäjälle heti splash-kuvan jälkeen. Aloitusvalikosta tuli pystyä sulkemaan sovellus, näyttämään tietosivu ja aloittamaan sovelluksen toiminta. Menuvalikko näytetään, kun sovelluksen käynnissä ollessa käyttäjä painaa oikeassa yläkulmassa sijaitsevaa kuvaketta. Menuvalikosta käyttäjän täytyi pystyä lopettamaan sovelluksen käyttö, aloittamaan käyttö alusta tai jatkamaan käyttöä samasta tilanteesta.

Toimintaa havainnollistavat kuvat näyttävät miltä sovellus näyttäisi käytössä. Taustalla näkyy kameran kuvaama ympäristö. Ruudun yläreunassa on painike, jolla voi avata menuvalikon. Alareunassa on ohut palkki, jossa näytetään ohjeita käyttäjälle. Palkin oikeassa reunassa on painike, jolla voi piilottaa tai näyttää ohjeet.

7.2 Sovelluksen luominen

Unreal Enginessä sovelluksen kehittäminen aloitettiin valitsemalla uuden projektin määritykset. AR-sovelluksen kehittämisen aloittamiselle Unreal Enginessä oli kaksi vaihtoehtoa: Handheld AR -mallipohja, joka antoi valmiiksi rakennetun pohjan lisätyn todellisuuden sovellukselle tai uusi tyhjä pohja. Mallipohja sisälsi tälle projektille tarpeettomia tiedostoja ja asetuksia, joten pohjaksi valittiin uusi tyhjä projekti käyttäen asetuksia: Blueprint-ohjelmointi, skaalautuva 3D tai 2D, Raytracing pois päältä, kohdelaite mobiili, ja ei aloitussisältöä (Kuva 17).

Kuva 17. Uuden Unreal Engine -projektin aloittaminen

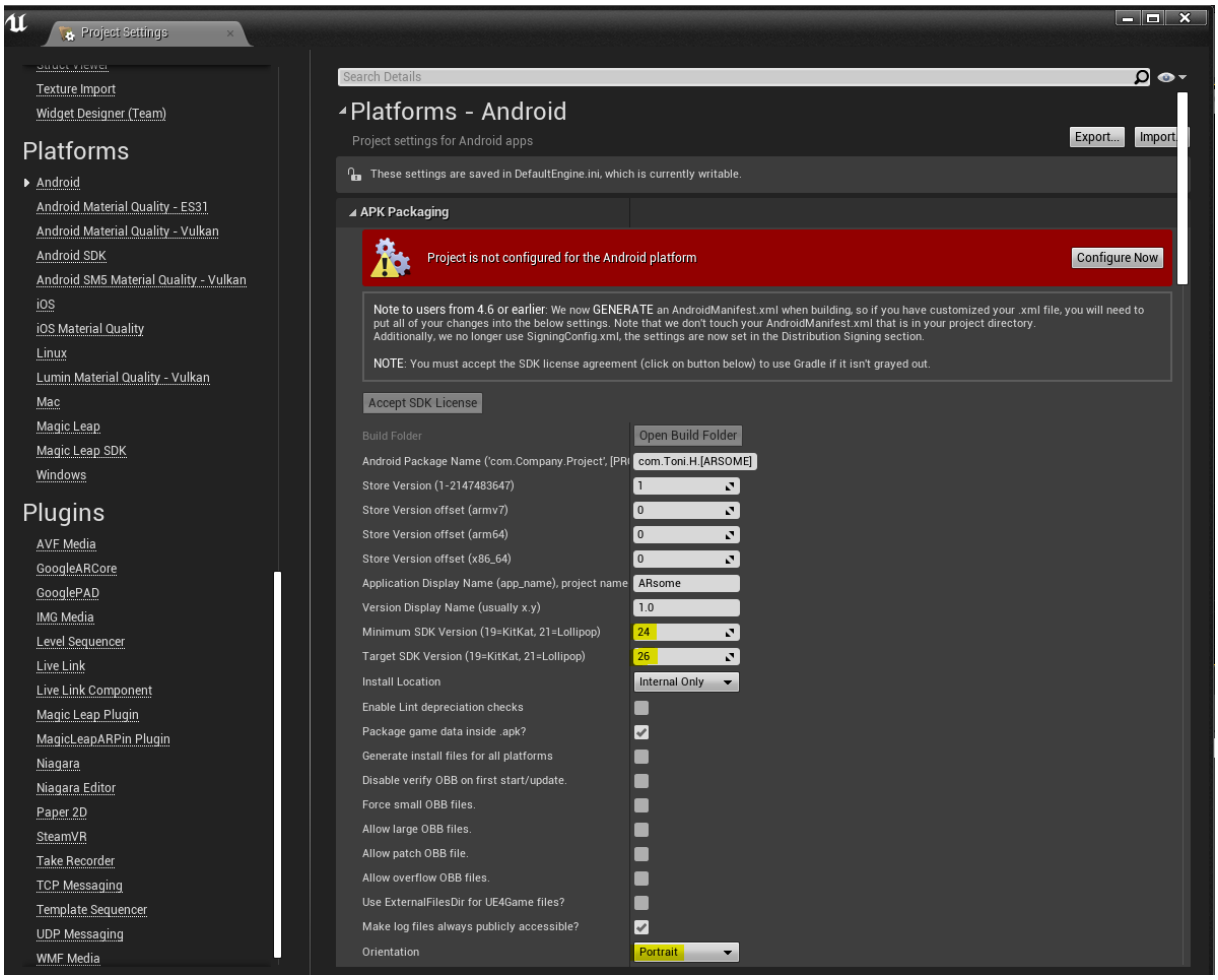


Projektin asennuttua, ensimmäiseksi asetettiin Settings → Plugins-valikosta Google ARCore -liitännäinen käyttöön. Liitännäisen käyttöönotto viimeisteltiin sovelluksen uudelleen käynnistämisellä. Sovelluksen käynnistyttyä uudelleen, luotiin projektiin uusi taso File → New Level, valitaan Empty Level ja tallennettiin se nimellä MainMenu.

7.2.1 Projektiasetukset

Sovelluksen kehitystyön aluksi muutettiin projektin asetuksia valitsemalla Settings → Project Settings. Ensimmäiseksi määriteltiin kohdasta Maps & Modes sovelluksessa oletuksena käynnistettävä taso. Tähän valittiin juuri luotu MainMenu-tiedosto. Seuraavaksi oli muutettava myös Android-kehittämisen asetuksia. Platforms → Android -kohdasta hyväksyttiin Android-konfiguraatio (Configure Now), joka näkyi punaisena ilmoituksena (Kuva 18). Seuraavaksi valittiin sovelluksen näyttötyyppi (Orientation). Sovelluksen näyttötyyppi saattoi olla vaaka, pysty, tai vaihtoehtoisesti voitiin määrittää puhelin tunnistamaan asetus laitteen asennon mukaisesti. Tälle sovellukselle asetettiin näyttötyypiksi Portrait (pystykuva). Android-asetuksissa voi myös nimetä sovelluksen ja määrittää tuetut Android-versionumerot, nämä asetukset koskevat lähinnä julkaisua Play-kauppaan. Jos sovellus julkaistaan sovelluskaupassa, niin versiovalintojen avulla rajataan sovellus pois niiden laitteiden ulottuvilta, jotka eivät tue AR-sisältöä. Asetuksista voi myös vaihtaa sovelluksen käyttämiä kuvia, kuten ikoni ja splash-kuva. Kuvat on mahdollista tuoda Figmalla luoduista malleista .png-tiedostomuodossa.

Kuva 18. Android-kehityksen asetukset



7.2.2 Projektin kansiot, tiedostot ja niiden nimeäminen

Jotta projektin kasvaessa tiedostot pysyivät järjestyksessä sekä löytyivät helposti, sijoitettiin kaikki lisättävät tiedostot tyypin mukaan omiin kansioihinsa. Unreal Enginen dokumentaatiossa suositeltiin käyttämään tiedostojen nimeämisessä tiettyjä käytäntöjä. Ohje käsitteli enemmän pelikehitystä kuin mobiilisovelluskehitystä, joten tässä projektissa selvemmän kuvan antamiseksi tehtiin nimeämiskäytännöissä muutama poikkeus. Yleinen nimeämiskäytäntö suosittelee tiedoston nimeämistä lisäämällä nimen alkuun tiedoston tyypin lyhennys, näin tiedostot on nimetty selkeästi ja ne erottuvat helposti toisistaan kansionäkymässä.

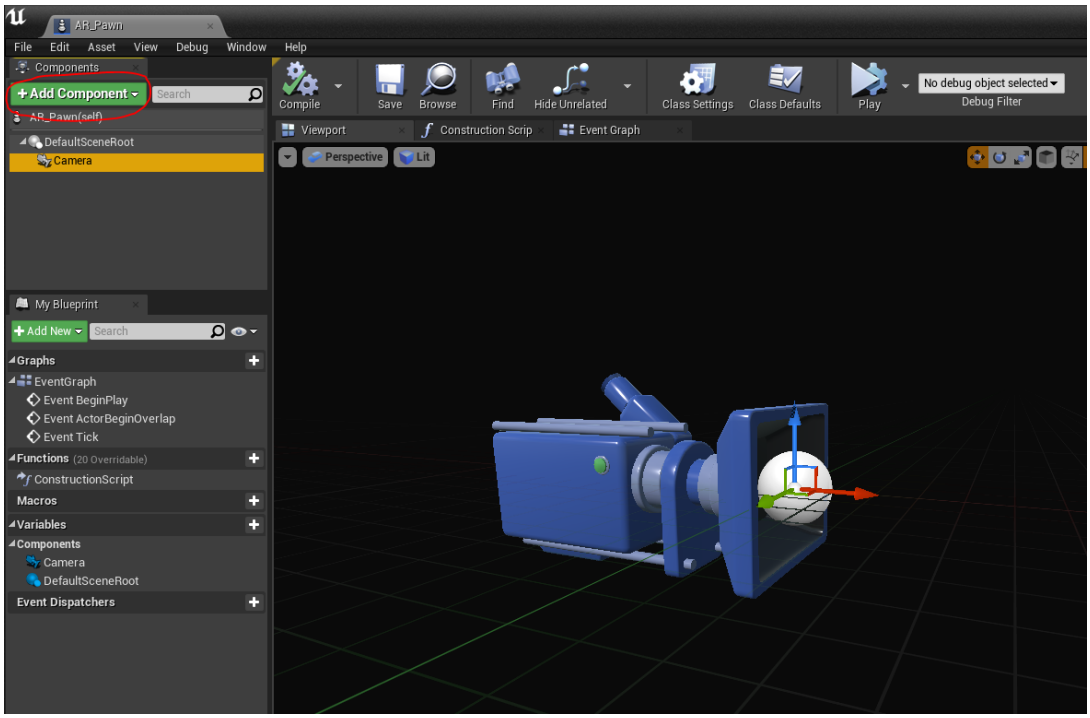
7.2.3 Blueprint-luokat

Projektissa käytettiin tasokohtaisen (Level Blueprint) lisäksi luokkiin Actor, Pawn ja Game Mode perustuvia Blueprintejä. Pawn-luokka on ihmisen tai tietokoneen ohjauksessa olevien aktorien yläluokka ja ohjattavan aktorin fyysinen ilmentymä. Tässä työssä se ilmaisi sovelluksen käyttäjää, tarkemmin sanottuna mobiililaitteen kameraa, sen liikkuvuutta ja näkymää. Game Mode -luokka luodaan jokaiseen Unreal Enginellä luotuun peliin tai sovellukseen. Sen tehtävä on ylläpitää pelin sääntöjä ja asetuksia. Peliin asetettavat säännöt pitävät sisällään tietoa esimerkiksi pelaajien määrästä, voiko peliä tauottaa ja mihin pelihahmot syntyvät pelimaailmassa. Actors-luokka on geneerinen luokka, joka tukee kolmiulotteisia tilamuunnoksia. Actor (Aktori) on objekti, joka voidaan synnyttää tai tuhota ohjelmakoodilla pelin aikana. Sovelluksessa päälle lisättävän todellisuuden malli perustui Actor Blueprint-luokkaan.

Uusi Blueprint-luokka luotiin painamalla projektin kansionäkymässä (Content Browser) hiiren oikeaa painiketta ja valitsemalla Create Basic Asset -kohdasta Blueprint Class. Projektiin oli luotava luokat Actor, Pawn ja Game Mode. Nimeäminen tehtiin liittämällä tiedoston etuliitteeksi BP_ kuvaamaan Blueprint-tyyppiä. Actor nimettiin BP_AR_Placeable, muut tiedostot olivat BP_AR_Pawn ja BP_AR_GameMode.

Koska BP_AR_Pawn ilmaisee sovelluksen käyttäjää, sille lisättiin kamerakomponentti. Avattiin tiedosto ja lisättiin kamerakomponentti valitsemalla Add Component → Camera (Kuva 19). Blueprintiin tehtyjen muutosten jälkeen valittiin työkalupalkista Compile ja sen jälkeen Save, jotta muutokset astuivat voimaan.

Kuva 19. Kamerakomponentin lisääminen Pawn Blueprintiin

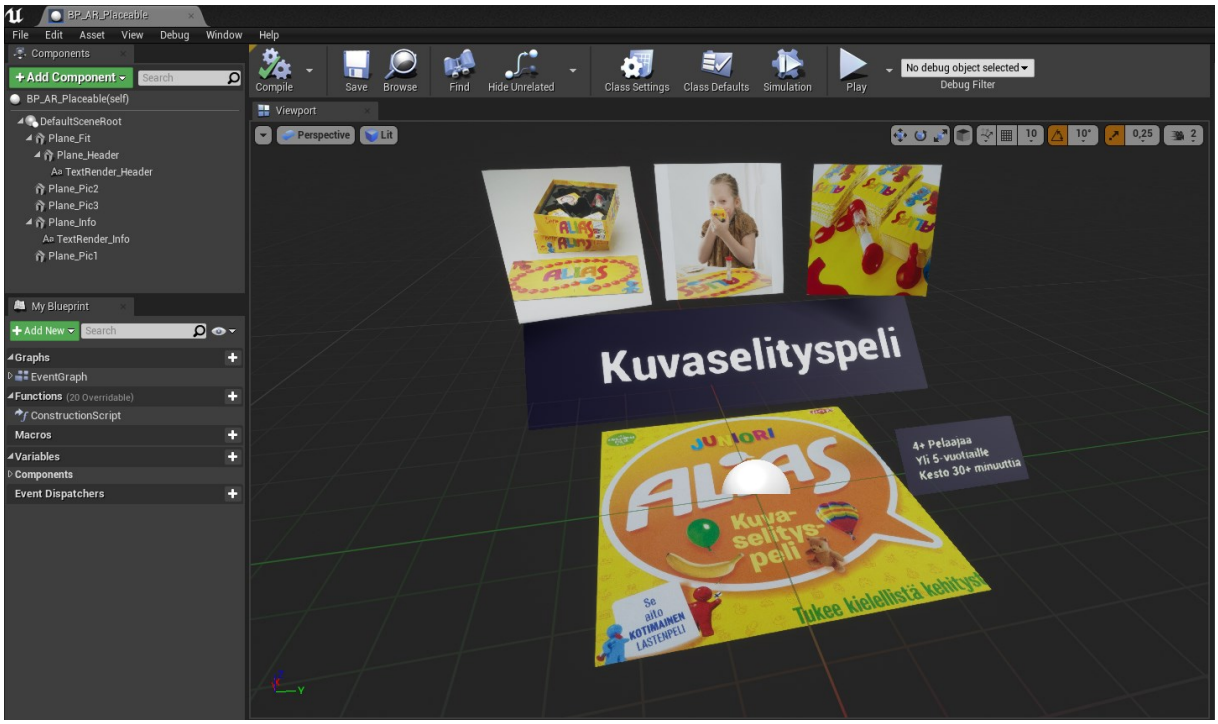


Korvattiin sovelluksessa käytettävä Game Mode avaamalla projektiasetukset ja kohdasta Maps & Modes vaihdettiin oletusvaihtoehdon tilalle BP_AR_GameMode. Avattiin seuraavaksi sen alapuolella oleva alasvetovalikko ja Pawn-luokan tilalle muutettiin BP_AR_Pawn.

BP_ AR_Placeable pitää sisällään päälle laitettavan lisätyn todellisuuden mallin. Tätä mallia oli muokattava sen mukaisesti, millaista sisältöä käyttäjälle oli tarkoitus näyttää oikean kuvan tunnistamisen jälkeen. Tässä sovelluksessa tarkoituksena oli tunnistaa lautapeli sen kansikuvan perusteella ja sen jälkeen näyttää mobiililaitteen ruudulla yleistietoa ja tuotekuvia pelistä. Ensin tuotiin projektiin kuva lautapelin kannesta painamalla kansionäkymässä hiiren oikealla painikkeella ja valitsemalla Import Asset. Nimettiin tiedosto selvyyden vuoksi T_Candidate. Tuotua tiedostoa hyödynnettiin tunnistettavana kuvana, sekä apuna kohdistamaan lisätty todellisuus. Tuodusta kuvasta luotiin materiaali painamalla tiedostoa hiiren oikealla ja valitsemalla Create Material. Luotu materiaali nimettiin M_Placeable. Seuraavaksi avattiin BP_AR_Placeable-Blueprint ja lisättiin sille uusi Plane-komponentti. Komponentti nimettiin Plane_Fit- ja liitettiin Details-ikkunassa sille materiaaliksi äsken luotu M_Placeable. Tämän Plane-komponentin avulla oli mahdollista kohdistaa muu lisätty todellisuus ilmestymään oikeisiin kohtiin. Lopuksi komponentista

Plane_Fit tehtiin näkymätön, Details → Rendering → Visible (false). Kuva 20 näyttää valmiin BP_AR_Placeable-luokan sisällön, siihen on lisätty tekstikomponentteja (Text Render) infotekstille ja otsikolle, sekä erikseen tuotuja tuotekuvia. Näiden pohjana käytettiin Plane-komponentteja.

Kuva 20. Päälle laitettava sisältö BP_AR_Placeable

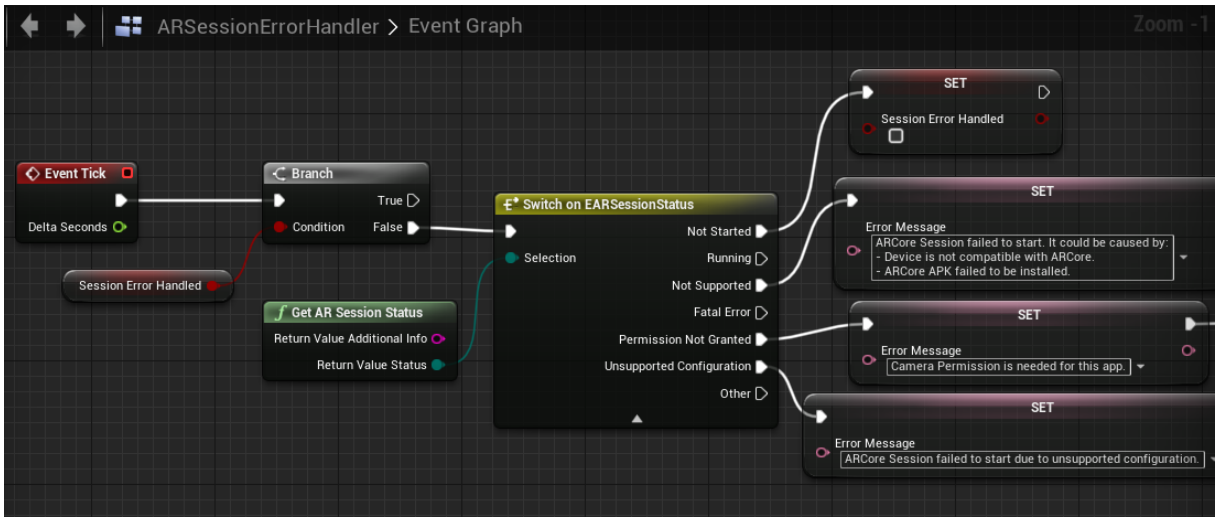


Kappaleessa 6.4.4 GitHubista käyttöympäristön testausvaiheessa ladattu Google ARCore SDK (2019) sisälsi AugmentedImages-esimerkkiprojektin, jossa AR-tapahtuman virheenkäsittelyyn käytettiin Blueprint-luokkaa ARSessionErrorHandler. Assetti oli mahdollista kopioida projektista toiseen avaamalla projekti ja painamalla halutun Blueprintin kohdalla hiiren oikeaa ja valitsemalla Asset Actions kohdasta Migrate. Sitten valittiin siirrettävät assetit, minkä jälkeen aukeavasta hakemistoikkunasta oli etsittävä kohdeprojektin Content-kansio, johon assetit liitettiin. Esimerkkiprojektista siirrettiin samalla myös käyttöliittymävempain MessageDialog, jota tultaisiin käyttämään ilmoituksen näyttämiseen käyttäjälle.

Käynnissä ei voi olla kerrallaan kuin yksi AR-tapahtuma. AR-tapahtuman käynnistäminen ja lopettaminen täytyi tehdä Unreal Enginen Level Blueprintissä, jotta voitiin varmistua, että tapahtuma käynnistyisi ja päättyisi aina oikean tason ollessa käynnissä. AR-tapahtuma

otettiin käyttöön Start AR Session -nodella. Käyttöönoton jälkeen virheentarkistus tapahtui äsken kopioidussa BP_ARSessionErrorHandlerissa. Tämä luokka sisältää jokaisen framen aikana ajettavan Event Tick -tapahtuman (Kuva 21), jonka tehtävä oli havaita mahdollinen virhe AR-tapahtumassa ja näyttää käyttäjälle virheilmoitusikkuna.

Kuva 21. AR-tapahtuman virheentarkistus

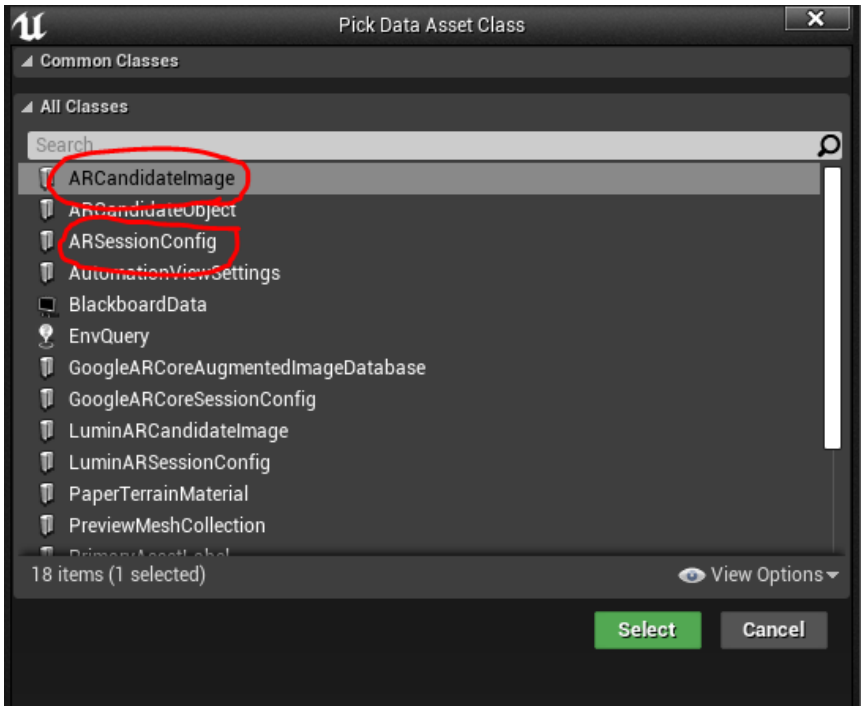


7.2.4 Data assetit

Data-assetit sisältävät nimensä mukaisesti tietoa. Tässä projektissa käytettiin kahta erilaista data-assettia: ARSessionConfig ja ARCandidateImage. ARCandidateImage on tietokanta, jonka tehtävänä on kertoa sovellukselle, minkälaista kuvaa on tarkoitus tunnistaa. Tietoina sille syötettiin tunnistettava tekstuuri, nimi, viitteellinen koko oikeassa maailmassa, sekä kuvan orientaatio. ARSessionConfigin tarkoitus on ylläpitää tietoa käynnissä olevasta AR-tapahtumasta.

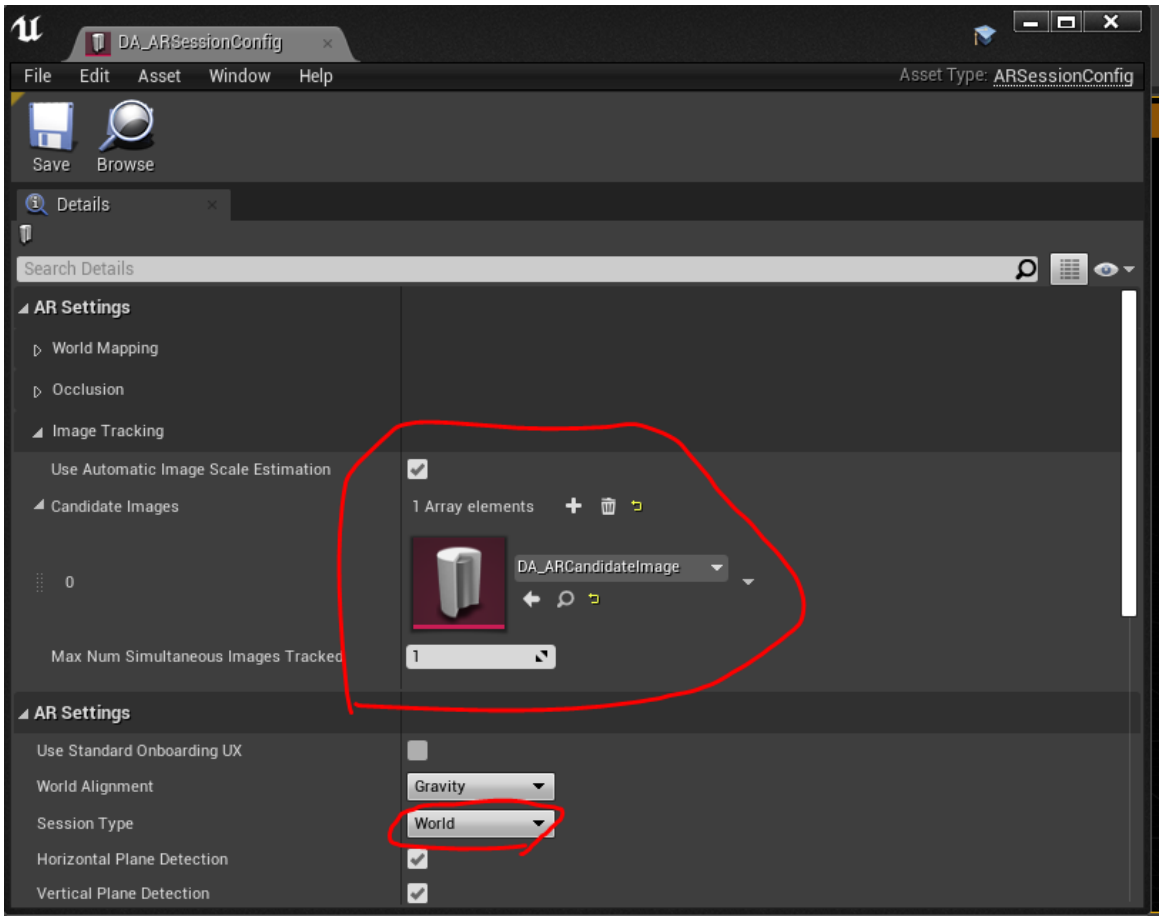
Data-assetti luotiin painamalla kansionäkymässä hiiren oikeaa painiketta ja kohdasta Miscellaneous valitsemalla Data Asset. Kuva 22 osoittaa projektiin liitettävät data-assetit, jotka nimettiin DA_ARSessionConfig ja DA_ARCandidateImage.

Kuva 22. Data-assettien luominen



Kun DA_ARCandidateImage oli saatu asetettua tunnistettavan kuvan (T_Candidate) tiedoilla, avattiin DA_ARSessionConfig ja liitettiin tunnistettava sisältö Candidate Images -kohtaan (Kuva 23). DA_ARSessionConfigin tärkeimmät asetukset ovat tunnistettavien kuvien tietokannan ylläpitäminen ja AR-tapahtumatyypin valinta. Tässä projektissa tapahtumatyyppinä käytettiin World-valintaa.

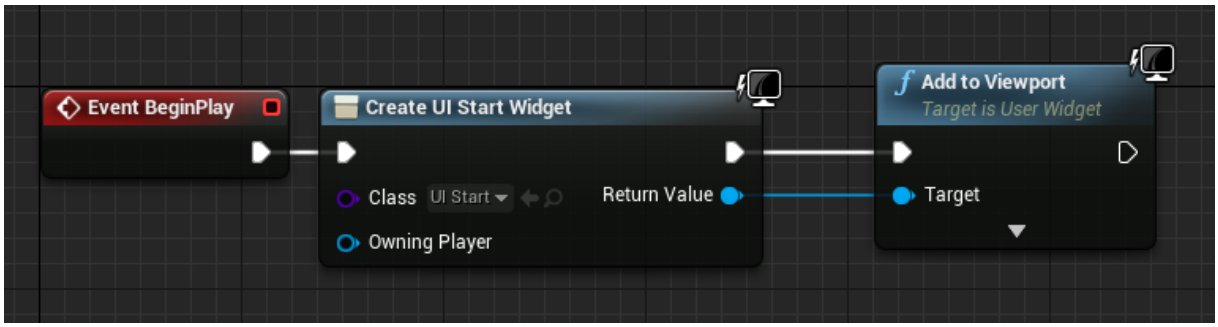
Kuva 23. DA_ARSessionConfigin asetukset



7.2.5 Level-tiedostot ja Level Blueprintit

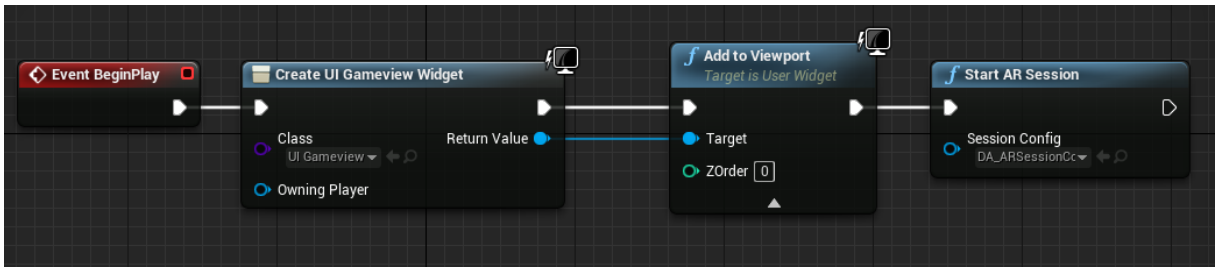
Unreal Enginessä jokaisella tasolla on oma Blueprint (Level Blueprint), johon ohjelmoidaan tasolle ominainen toiminta. Tässä sovelluksessa oli kaksi tasoa: MainMenu (aloitusvalikko) ja ARsome (päätaso). MainMenu on taso, jonka sovellus avaa käyttöön ensimmäisenä. Sen Level Blueprintiin (Kuva 24) aluksi luotiin käyttöliittymävempain (UI Widget) käyttäen lähdeluokkana UI_Startia, mikä sen jälkeen lisättiin käyttäjän näkymään (Add to Viewport).

Kuva 24. MainMenu-tason Level Blueprint



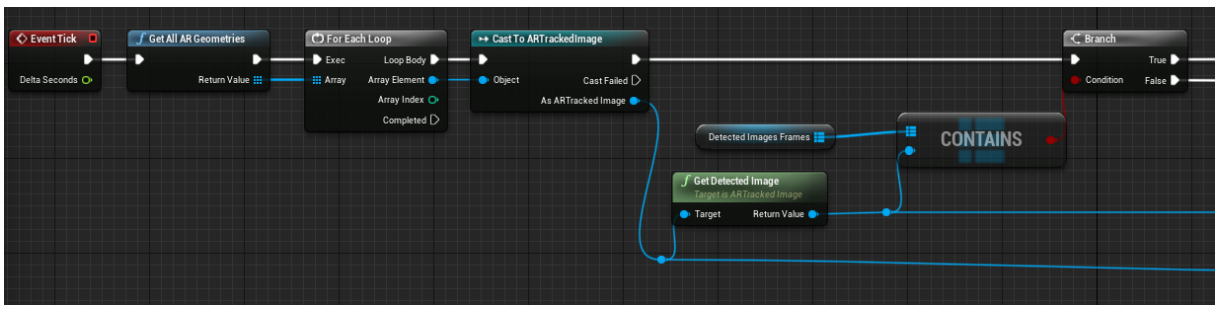
ARsome on pelin päätaso. Se avataan, kun aloitusvalikosta aloitetaan sovelluksen käyttö painamalla Start-painiketta. Kuva 25 sisältää tason aloitustapahtuman toiminnan: Tason käynnistyttyä luodaan käyttöliittymävempain UI_Gameview, joka lisätään käyttäjän näkymään. Tämän jälkeen aloitetaan AR-tapahtuma käyttäen konfiguraatiotiedostona data-assettia DA_ARSessionConfig.

Kuva 25. ARsome-tason aloitustapahtuma



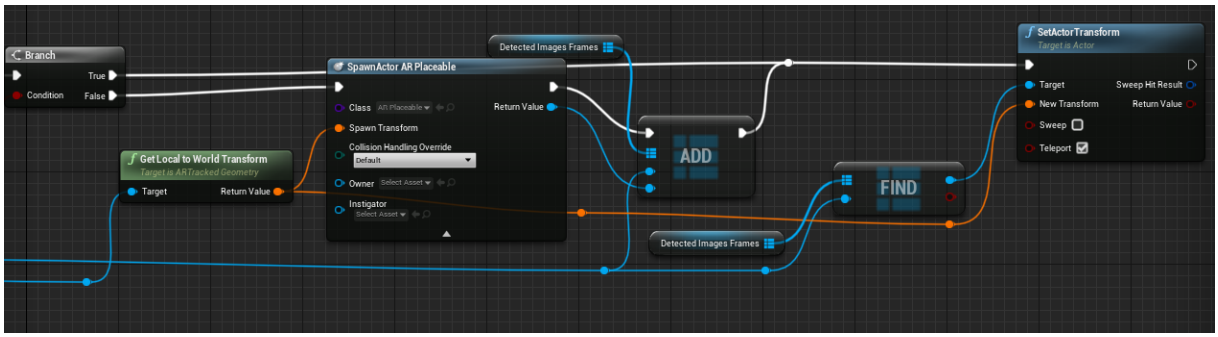
ARsome-tason Blueprintissä suoritetaan myös sovelluksen päätoiminnallisuus. Jokaisella framella ajettavassa Event tick -tapahtumassa ympäristöä tunnistetaan ennalta määritettyjen kuvien (DA_ARCandidateImage) löytämiseksi (Kuva 26).

Kuva 26. ARsome Level Blueprint 1/2: kuvien tunnistaminen



Tunnistettavan kuvan löydyttyä se lisätään tunnistettujen kuvien säilöön (Kuva 27). Kuva pysyy säilössä muistissa, kunnes sovellus suljetaan tai menuvalikon kautta tehdään tapahtuman resetointi. Kun kuva on lisätty säilöön, synnytetään aktori (BP_AR_Placeable) tunnistetun kuvan tilalle.

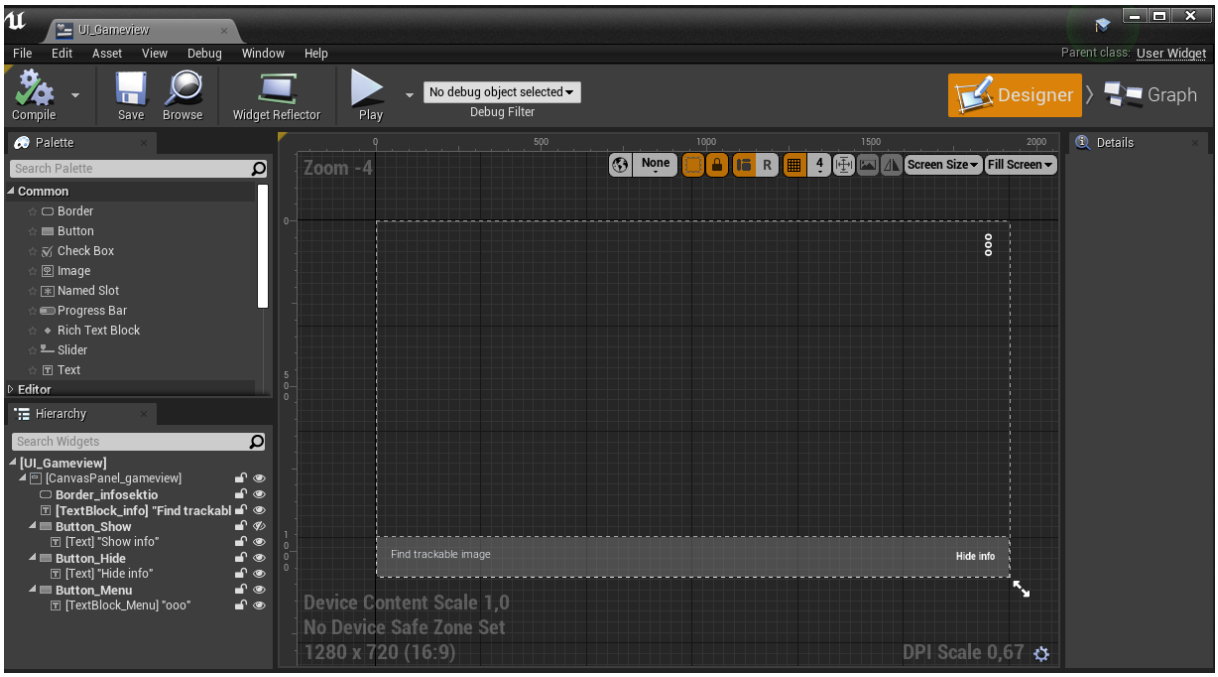
Kuva 27. ARsome Level Blueprint 2/2: lisätyn todellisuuden synnyttäminen



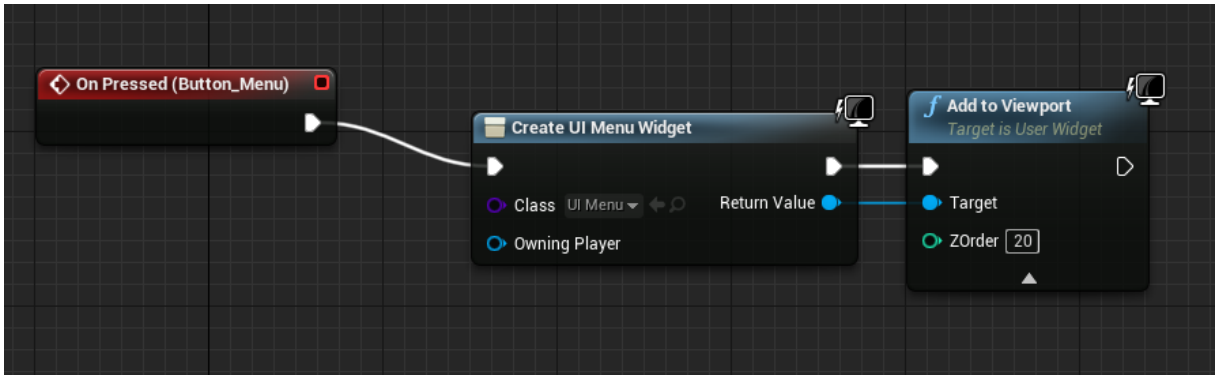
7.2.6 Graafiset käyttöliittymät

Graafiset käyttöliittymät luotiin Unreal Enginessä käyttäen UI Widget Blueprintiä. Se löytyi, kun kansionäkymässä hiiren oikeaa painamalla, kohdasta User Interface valittiin Widget Blueprint. Käyttöliittymät nimettiin tässä projektissa UI_-etuliitteellä. Sovelluksen käyttöliittymävempainten kehittämisessä noudatettiin Figmalla suunniteltuja malleja. Kuva 28 näyttää käyttöliittymän kehitystyökalun Designer-näkymän ja valmiin pääkäyttöliittymän (UI_Gameview). Kuvassa vasemmalla sijaitseva Palette-osio sisältää komponentteja, joita ovat esimerkiksi käyttöliittymän painikkeet ja tekstikentät. Haluttu komponentti vedetään Palette-ikkunan alapuolella olevaan Hierarchy-osioon, minkä jälkeen oikealla Details-ikkunassa annetaan tarkemmat tiedot kuten komponentin nimeäminen, sijainti, mitat ja väri. Kuva 29 sisältää käyttöliittymän kehitystyökalun Graph-näkymän ja tapahtuman, jota kutsutaan käyttäjän painaessa sovelluksen menunäppäintä.

Kuva 28. Sovelluksen pääkäyttöliittymän UI_Gameview Designerin näkymä



Kuva 29. Graph-näkymä tapahtumasta, joka avaa menuvalikon käyttäjälle



7.3 Testaus

Työn edetessä toiminnallisuudet ja sovelluksen käyttöliittymä alkoivat nopeasti olla kompleksisempia ja oli tarpeellista aloittaa säännöllisten testausten suorittaminen. Tämä mahdollisti ratkaisujen yhteensopivuuden ja toimivuuden toteamisen jo aikaisessa vaiheessa kehittämistyötä. Testaus suoritettiin liittämällä Android-kohdelaite USB-virheenkorjaustilassa tietokoneeseen ja ajamalla Unreal Enginessä mobiililaitteille tarkoitettu testaustila. Testaustilassa Unreal Engine pakkasi sovelluksen ja loi automaattisesti Android-pakettitiedoston (APK) mobiililaitteelle sekä suoritti sovelluksen

käynnistämisen. Testaustilaa ajettaessa ensimmäistä kertaa, Unreal Enginen tiedostojen koostaminen kesti noin tunnin ajan, mutta myöhemmillä kerroilla oli huomattavasti nopeampaa. Testausprosessi mobiililaitteella on kuvattu tarkemmin kappaleessa 6.4.4. Kuva 30 kuvantaa vaiheen, jossa testataan sovelluksen käyttöliittymän menupainikkeen sekä alaosion info-osion toimintaa.

Kuva 30. Sovelluksen käyttöliittymän testaus

7.4 Julkaiseminen sovellukseksi

Sovelluksen julkaisupaketin luomiseen Androidille oli kolme vaihtoehtoa: DXT, ETC2 ja ASTC. DXT on Nvidia Tegra -grafiikkaprosessorin tukema, ETC2 sopii kaikille OpenGL 3.x -laitteille ja ASTC on viimeisin tekstuuripakkausformaatti, joka ei ole tuettuna kaikille laitteille, mutta mahdollistaa paremman laadun säätelyn. Pakkaaminen aloitettiin valitsemalla:
File → Package Project → Android → Android ASTC
Pakkaukselle valittiin kohdekansio, minkä jälkeen prosessi eteni automaattisesti. Kun prosessi oli valmis, kytkettiin Andoid-laite USB-johdolla tietokoneeseen ja avattiin äsken luotu Android_ASTC-kansio. Kansio sisälsi tiedoston Install_[sovelluksen_nimi]-armv7.bat, jonka ajamalla suoritettiin sovelluksen asennus mobiililaitteeseen.

8 Tulokset

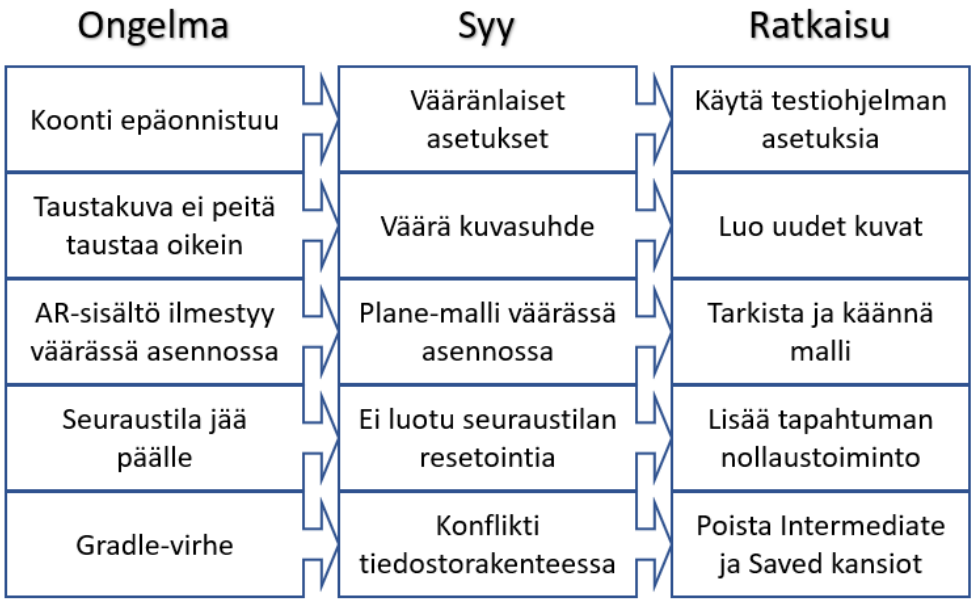
Opinnäytetyön kehitystyön tuloksena syntyi visuaalista ohjelmointia käyttäen kehitetty lisätyn todellisuuden mobiilisovellus. Tässä luvussa kerrotaan ensin sovelluksen testauksen myötä saaduista tuloksista, jonka jälkeen reflektoidaan tutkimusongelmiin löydettyjä vastauksia. Lopuksi esitellään valmiin sovelluksen toiminta.

8.1 Testaushavainnot

Sovellusta kehittäessä todettiin toimivaksi ohjelmoinnin ja testauksen vuorottelu. Kuva 31 ilmentää graafisesti testauksessa havaittujen ongelmien ratkaisuprosessia. Ensimmäiset testausyritykset epäonnistuivat pakettitiedoston koontivaiheessa vääränlaisten asetusten myötä. Tähän ratkaisuna päätettiin testata ARCoren testiohjelmassa käytössä olleita asetuksia, mikä ratkaisi ongelman. Ensimmäinen testaus osoitti myös, että Figmalla mobiililaitteen näyttökoon mukaisesti luodut sovelluksen taustakuvat eivät täsmänneet kohdelaitteen näyttökoon kanssa, eikä Unreal Enginellä pystynyt lennossa muokkaamaan taustakuvien kokoa. Ratkaisuna oli tehdä uudet vastaavat taustakuvat suuremmalla koolla. Muita testauksessa havaittuja huomioita olivat, että päälle laitettava lisätty todellisuus ilmestyi väärässä kulmassa todellisen sisällön tilalle. Lisäksi tunnistettuaan kohteen AR-tapahtuma jäi seuraustilaan, minkä päättämiseksi täytyi sovellus sammuttaa. Ratkaisuna tähän päätettiin menuvalikkoon lisätä valinta seuraustilan nollaamiseksi.

Ajoittain projektin kansiorakenteen eli projektitiedostojen sijaintien vaihtaminen tai uudelleen nimeäminen aiheutti Android-pakettitiedoston koostovaiheessa ilmenneen Gradle-pakkausvirheen, joka esti testausprosessin ajon. Tähän ratkaisuna oli kehitysprojektin tiedostohakemistosta poistaa kansiot Intermediate sekä Saved. Intermediate-kansio sisälsi testausprosessin aikana luotavat koontitiedostot ja Saved-kansio varmuuskopioita projektista, joten peruuttamattoman virheen estämiseksi projekti kannatti varmuuskopioida manuaalisesti ennen näiden kansioiden poistamista.

Kuva 31. Kaavio testauksessa havaittujen ongelmien ratkaisuprosessista



8.2 Ratkaisuja tutkimusongelmiin

Opinnäytetyön alkuvaiheessa asetettiin kolme työn tekemistä ohjaavaa ja tukevaa tutkimuskysymystä. Kysymykset suunniteltiin siten, että niihin vastauksen antaminen olisi mahdollista raportin eri vaiheissa. Ensimmäiseen kysymykseen lisätyn todellisuuden vaikutuksesta käyttäjän kokemukseen vastattiin tietoperustassa kappaleessa 2.1. Ongelmaa lähestyttiin ensin käsitteen määrittelyllä, jossa todettiin lisätyn todellisuuden olevan tekniikka, joka yhdistää digitaalisia elementtejä todelliseen maailmaan, tunnetuimman toteutustavan ollessa mobiililaitteella reaaliaikaisesti kuvatun sisällön päälle liitettävä digitaalinen kerros. Kappaleessa esiteltiin lisäksi tekniikkaa käyttäviä tunnettuja sovelluksia. Luvussa 2 lisättyä todellisuutta käsiteltiin aihetta laajasti tarkoituksena löytää vastaus, miten lisätty todellisuus on ajan kuluessa kehittynyt ja minkälaisia käyttäjäkokemuksia se on kehityspolkunsa aikana tarjonnut. Tuloksena mainittiin useita eri käyttökohteita ja toteutustapoja, joista perinteisten mobiililaitteiden ja älylasien lisäksi mielenkiintoisena uutena havaintona löytyi projisoiva, eli valolla tuotettu lisätty todellisuus.

Kehittämistyön ohjelmistokehykseksi valittiin Unreal Engine -pelimoottori. Pelimoottori esiteltiin luvussa 3. Jotta pelimoottorilla pystyttiin kehittämään lisättyä todellisuutta hyödyntävä Android-mobiilisovellus, tarvittiin erillinen lisäosa. Androidin ollessa Googlen

omistama käyttöjärjestelmä, oli Google ARCore looginen ratkaisu liitännäisen valinnassa. Valintaa painotti myös se, että Unreal Engine tuki ARCorea ohjelmointirajapintana lisätylle todellisuudelle. Google ARCore esiteltiin tarkemmin luvussa 5.

Toinen tutkimuskysymyksistä liittyi Unreal Enginellä ja ARCorella tehtävään kehitystyöhön. Tarkoitus oli löytää mahdollisia rajoitteita tai ongelmia koskien valituilla työkaluilla tehtävää kehitystä. Tämä kysymys oli valittu sitä silmällä pitäen, että vastaus löytyisi kehitystyön aikana. Rajoitteita ja ongelmia kohdattiin kehitystyön aikana muutamia, pääosin ongelmat muodostuivat saatavilla olleiden lähdemateriaalien, kehittäjien käyttöönottodokumenttien ansiosta. Kehitysympäristöä lähdettiin rakentamaan ARCore-dokumentaation perusteella, koska ajateltiin sen olevan määrittelevä tekijä toimivan ympäristön rakentamisessa.

Ensimmäinen kohdattu ongelma on mainittu opinnäytetyön kappaleessa 6.4. Huomattiin, että ARCore-dokumentaatio ei ole ajantasainen ja olisi löydettävä toinen lähde toimivan ympäristön rakentamiselle. Unreal Engine (n.d.-e) -dokumentaatio sisälsi ohjeet mobiilikehitysympäristön asentamiseksi pelimoottorille. Tämän dokumentaation avulla onnistuttiin kehitysympäristön asentamisessa. Kehittämistyön alkaessa kohdattiin uusia ongelmia, jotka liittyivät ohjelmistoversioiden päivitysten myötä vanhentuneisiin ominaisuuksiin. Esimerkkiprojektissa käytössä olleita ARCore-luokkia ja funktioita oli poistettu käytöstä. ARCoren dokumentaation ollessa päivittämätöntä, ei tästä ollut apua selventämään miten näitä luokkia tulisi käyttää. GitHubissa julkaistun ARCore-ohjelmistopaketin yhteydessä olleet julkaisutiedot kertoivat, mitkä luokat olivat poistuneet käytöstä ja miten ne vaihtoehtoisesti tulisi korvata projektissa.

Kohdatut rajoitteet hidastivat työn kehittämistä huomattavasti eikä julkaisutiedoista ollut täydellistä apua visuaalisen ohjelmoinnin läpiviemiseksi. Jos Blueprint-ohjelmoinnin logiikka olisi ollut tekijällä paremmin hallussa, olisi ratkaisu onnistuttu löytämään nopeammin. Lopulta asia ratkaistiin, kun Blueprint-ohjelmointiin tutustuttiin laajemmin erilaisten dokumentaatioiden ja opastusvideoiden avulla. Vastauksen myötä ymmärrettiin, että jotkin luokat ja funktiot täytyi kutsua erikseen esiin käyttämällä ensin luokkaa kutsuvaa nodea. Kehitystyössä käytettiin kuvan tunnistukseen rajattua osaa ARCore-ohjelmointirajapinnoista, joten kokonaiskuvaa mahdollisista rajoitteista ohjelmistojen välillä ei voida luoda. Tehdyn

työn perusteella voidaan kuitenkin olettaa, että versioiden päivitykset saattavat mahdollisesti synnyttää vastaavia ongelmia muidenkin ohjelmistorajapintojen käytössä.

Viimeinen tutkimusongelma käsitteli visuaalisen ohjelmoinnin soveltuvuutta AR-sovelluksen kehittämiseen. Kysymystä suunniteltaessa oletettiin ratkaisun löytyvän tuloksia kirjoittaessa. Opinnäytetyölle lähteitä etsiessä tutustuttiin Unreal Enginellä tuotettuihin peleihin ja todettiin, että visuaalinen ohjelmointi oli pelimoottorissa kehittynyt niin hyväksi, että sillä oli mahdollista luoda kokonaisia pelejä. Tämän myötä voitiin olettaa, että kehittämistyön onnistumiselle olisi edellytykset. Visuaalinen ohjelmointi Unreal Enginessä on monipuolinen työkalu, ja esimerkiksi käyttöliittymien toiminnallisuuksien luominen oli yksinkertaista. Parhaimmillaan Blueprintit ovatkin, kun tehdään tapahtumakeskeistä ohjelmointia. Valikkojen Graph-osioiden lisäksi ohjelmointia tehtiin Level Blueprintiin, johon luodaan tasolle ominainen ohjelmakoodi. Sovelluksessa tasoja on kaksi, joista ensimmäisen on sovelluksen käynnistyttyä ilmestyvä aloitusvalikko ja toinen lisättyä todellisuutta tunnistava ja luova ”pelitaso”. Alkuun ohjelmointilogiikan ymmärtäminen vei aikaa ja oikean noden löytäminen oli hankalaa. Tästä huolimatta visuaalinen ohjelmointi on mahdollista myös kehittäjälle, joka ei osaa ohjelmoida tekstipohjaisilla ohjelmointikielillä. Blueprint-ohjelmointi sisältää paljon valmiita funktioita, mutta niitä on mahdollista myös lisätä itse. Tämä on tarpeen projekteissa, joissa jokin ohjelmakoodi alkaa toistua useaan kertaan. Kehitystyön tuloksena syntynyt sovellus luotiin ainoastaan visuaalista ohjelmointia käyttämällä, joten voidaan todeta sen sopivan pelien kehittämisen lisäksi myös lisätyn todellisuuden Android-sovelluksen kehitykseen.

8.3 Valmis AR-sovellus

Kehitystyön lopputuloksena syntyi AR-sovellus mobiililaitteelle. Sovellus kehitettiin Unreal Enginellä ja ohjelmointi visuaalisella Blueprint-ohjelmointikielellä. AR-tapahtuman hallinta tapahtuu Google ARCore -ohjelmointirajapintojen avulla. Sovelluksen toiminta perustuu kuvan tunnistamiseen. Kuvan tunnistamisen jälkeen käyttäjälle näytetään digitaalisesti lisättyä todellisuutta. Tässä sovelluksessa lisätty todellisuus tarkoittaa tuotetietojen näyttämistä ennalta määritetyn tunnistettavan tuotteen päälle. Visuaalisen ohjelmoinnin lisäksi työ sisälsi käyttöliittymäsuunnittelua Figmalla. Sovelluksen käyttöliittymä toteutettiin suunniteltujen mallien pohjalta.

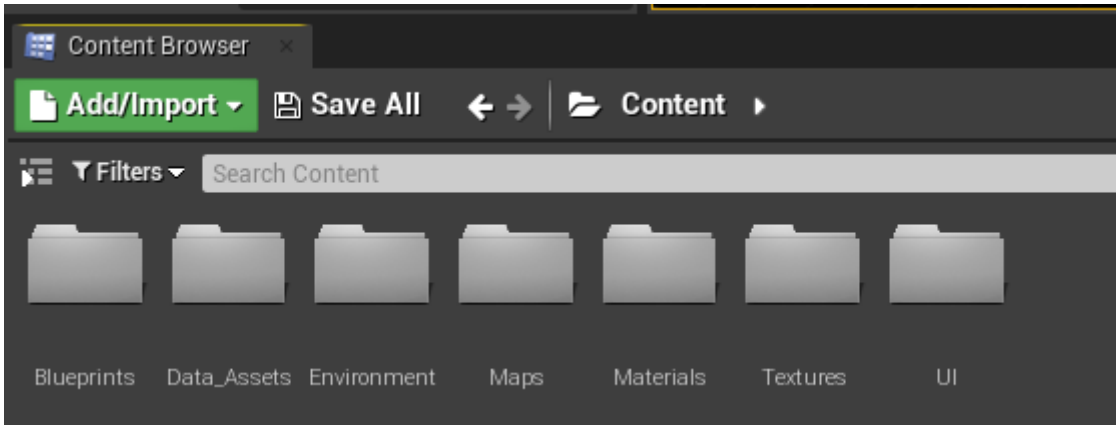
Taulukko 2 sisältää kehitysprojektin sovelluksen oleelliset tiedostot. Taulukon avulla voidaan nähdä kunkin tiedoston tyyppi ja tehtävä. Tiedot on esitetty järjestettynä tyypin mukaan. Taulukossa näkyvien tiedostojen lisäksi projekti sisälsi Figmalla luodut sovelluksen taustakuvat, sekä lisätyssä todellisuudessa hyödynnetyt tuotekuvat.

Taulukko 2. Unreal Engine -kehitysprojektin tiedostot järjestettynä tyypin mukaan

Tiedosto	Tyyppi	Tehtävä
BP_AR_Placeable	Actor Blueprint	Päälle laitettava sisältö
BP_ARSessionErrorHandler	Actor Blueprint	AR-tapahtuman virheentarkistus
BP_AR_GameMode	Game Mode Blueprint	Ylläpitää pelitietoja
BP_AR_Pawn	Pawn Blueprint	Käyttäjän fyysinen ilmentymä
DA_ARCandidateImage	Data Asset	Tunnistettava sisällön datatiedosto
DA_ARSessionConfig	Data Asset	AR-tapahtuman konfigurointi
ARsome	Level-tiedosto	Taso, jossa sovelluksen päätoiminta
MainMenu	Level-tiedosto	Taso, jossa aloitusvalikko
M_Placeable	Material-tiedosto	Päälle laitettavan sisällön materiaali
T_Candidate	Texture-tiedosto	Tunnistettavan sisällön tekstuuri
UI_About	Widget Blueprint	About-sivun graafinen käyttöliittymä
UI_Gameview	Widget Blueprint	Sovelluksen päätoiminnan graafinen käyttöliittymä
UI_Menu	Widget Blueprint	Menuvalikon graafinen käyttöliittymä
UI_MessageDialog	Widget Blueprint	Graafinen käyttöliittymä ilmoitusikkunalle
UI_Start	Widget Blueprint	Aloitusvalikon graafinen käyttöliittymä

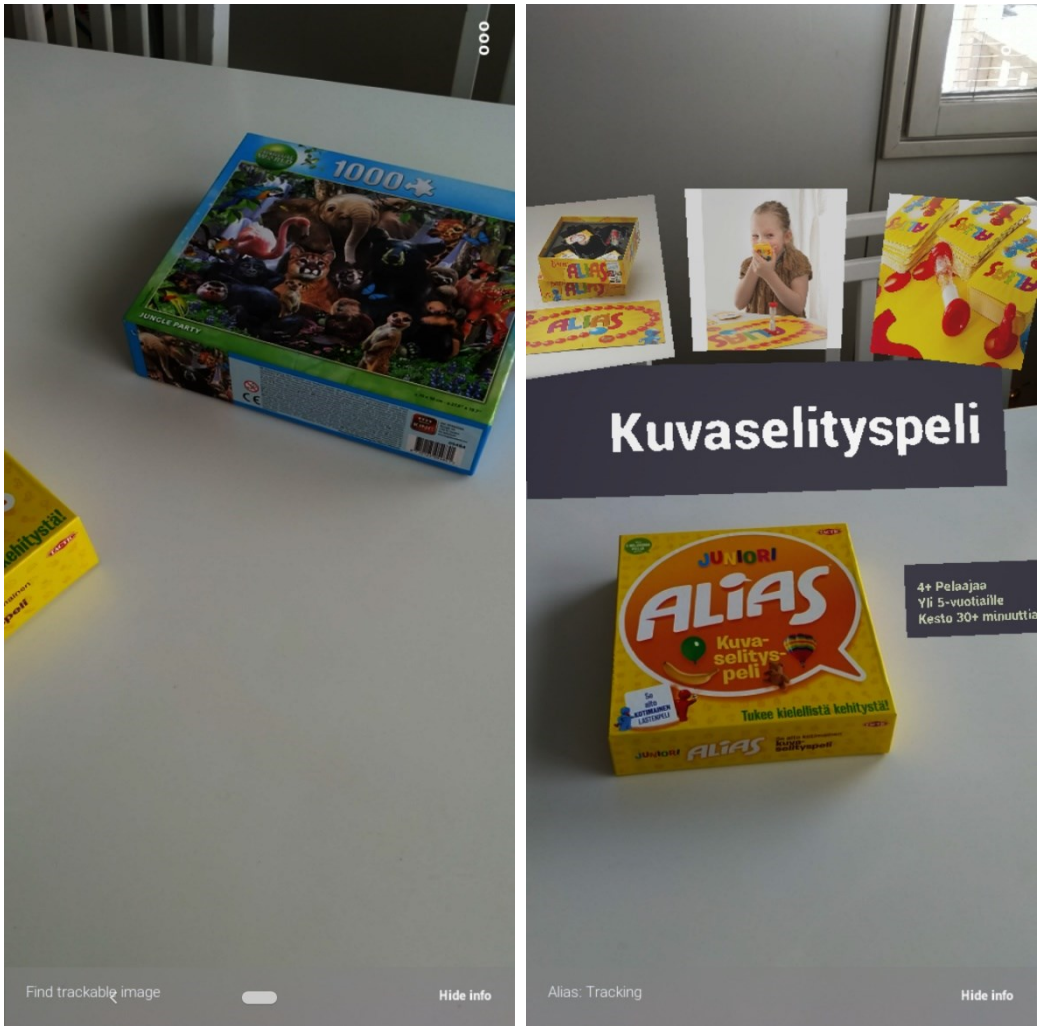
Kuva 32 osoittaa Unreal Engine projektin kansiorakenteen. Projektitiedostot pysyivät hyvin järjestyksessä, kun tiedostot liitettiin tyypin mukaan samaan sijaintiin.

Kuva 32. Unreal Engine -kehitysprojektin kansiot



Lopuksi sovellus julkaistiin Android-pakettina mobiililaitteelle, julkaisuprosessi kuvattiin kappaleessa 7.4. Kuva 33 sisältää kaksi käyttökuvaa valmiista sovelluksesta. Vasemmassa kuvassa näkyy sovellus ennen tunnistettavan kuvan löytymistä. Sovelluksen alareunassa on info-osio, jossa ilmoitetaan tunnistamisen tila käyttäjälle. Koska kuvaa ei ole vielä tunnistettu, osiossa ohjataan käyttäjää etsimään tunnistettavaa kohdetta. Oikeanpuoleisessa kuvassa on nähtävissä tunnistettavan kohteen löydyttyä lisätty todellisuus. Info-osioon on päivitetty tunnistetun kuvan nimi ja tunnistuksen tila. Sovellus säilyttää kuvan tunnistamisen tiedot kuluvan session ajan. Sessio päättyy, kun sovelluksen käyttö lopetetaan tai valitaan tunnistustilan nollaus menuvalikosta.

Kuva 33. Sovelluksen käyttökuvat ja lisätty todellisuus



9 Pohdinta

Päätavoitteena oli tehdä tutkimus, jolla olisi merkitystä tietojenkäsittelyn koulutusalaa opiskelevalle tai käytetyistä tekniikoista tietoa etsivälle henkilölle. Halusin käsiteltävien tekniikoiden olevan ajankohtaisia ja mahdollistavan luonnollisen siirtymän opiskelusta työelämän pariin. En ollut kohdannut ennen opinnäytetyön aloittamista visuaalisella ohjelmoinnilla kehitettyä lisätyn todellisuuden sovellusta mobiililaitteelle, joten uskoin työn olevan osaltaan hyödyksi tuleville kehittäjille ja saattavan valittujen tekniikoiden käyttöä paremmin tunnetuksi.

Opinnäytetyön teknologia valittiin, koska halusin tutkia virtuaalisesti muokatun todellisuuden kehittämistä. Työtä tehdessä maailmaa koetteli pandemia, joka muutti työnkuvaa usealta eri alalta pakottamalla työntekijöitä siirtymään laajalti etätyöskentelyn pariin. Tämä tarkoitti erilaisten digitaalisten ratkaisujen käytön nopeaa kasvamista. Virtuaalisesti tuotettu sisältö ja sen liittäminen todelliseen maailmaan on teknologia, joka mahdollistaisi työtehtävien ja koulutusten jatkuvuutta vallitsevissa erityistilanteissa. Lisätyn todellisuuden avulla voidaan järjestää erityyppisiä koulutustilanteita sijainnista riippumatta ja on siksi vakavasti harkittava ja kustannustehokas vaihtoehto perinteiselle fyysistä läsnäoloa vaativille opetustilanteille. Lisätty todellisuus mahdollistaa tasa-arvoisen koulutustilanteen järjestämisen jokaiselle osanottajalle sekä sellaisten simuloitujen tilanteiden luomisen, joiden kohtaaminen muuten olisi harvinaista tai järjestäminen kallista ja vaikeaa.

HAMK Smart -tutkimusyksikölle hiljattain tehdyn projektin parissa tutustuin visuaaliseen ohjelmointiin Unreal Enginen Blueprint-järjestelmällä. Visuaalinen ohjelmointi mahdollistaa ohjelmointilogiikan lisäämisen sovellukseen tai peliin kehittäjälle, joka ei osaa käyttää tekstipohjaisia ohjelmointikieliä. Mielipiteeni tästä huolimatta on, että ohjelmoinnista olisi hyvä ymmärtää vähintään perusteet, jotta kehittäjä ymmärtäisi ohjelmakoodin rakentumisen visuaalisen ympäristön taustalla ja osaisi muodostaa tehokkaasti ehtolauseita ja tarvittaessa tehdä itse lisää funktioita, koska niiden oikea käyttö selkiyttää ohjelmointiprosessia ja helpottaa mahdollista ongelmanratkaisua. Toisaalta on huomioitava, että isoissa projekteissa visuaalinen kehittäminen ja ohjelmointi ovat toisistaan täysin erilliset työtehtävät, milloin ohjelmoijien tehtävänä on luoda tehokkaat ohjelmakoodit ja

visuaalisen puolen kehittäjät puolestaan käyttävät näitä ratkaisuja tehdessään testausta visuaalisten ratkaisujen valitsemiseksi.

Opinnäytetyön tietoperustan kirjoittamisen vuoksi käytiin läpi suuri määrä alan tutkimuksia ja toimittajien kirjoittamia artikkeleita. Lähdemateriaali oli englanninkielistä ja tutkimusten analysoiminen ja hyödyllisen tiedon seulominen vei huomattavan paljon kirjoitusaikaa. Opinnäytetyössä käytettiin havainnollistavia kuvia ja taulukoita, jotta tieto olisi helpommin sisäistettävissä. Opinnäytetyön painopiste on kehittämistyössä. Toiminnallinen osuus opinnäytetyössä alkaa tavoitteiden esittelyllä ja suoritettavan työn kuvantamisella. Unreal Enginen valitseminen ohjelmistokehykseksi ja Androidin valitseminen kohdelaitteeksi ohjasivat osaltaan työnkulkua. Unreal Enginen tuki Googlen ARCore -ohjelmointirajapinnalle ja Androidin ollessa Googlen omistama käyttöjärjestelmä, muodostui ohjelmistojen rajaus helposti. Unreal Enginestä ja ARCoresta käytettiin ohjelmistojen tuoreimpia versioita. Kehitystyön perusta rakennettiin kehittäjien luoman dokumentaation varaan, senkin vuoksi, että vastaavaa teknologiaa käyttävän lähdemateriaalin määrä oli vähäistä. ARCoren harjoitusprojektit olivat kehitystyölle tärkein lähde niissä esiteltyjen API-esimerkkien vuoksi.

Tutkimuskysymyksiin löydettiin vastaukset. Lisätyn todellisuuden todettiin muuttavan kokemusta todellisesta maailmasta tuomalla käyttäjälle lisäinformaatiota, myös sovellusten viihdearvo noteerattiin. Kappaleessa 2.3 esiteltiin tekniikan toteutustapoja ja kappaleessa 2.5 lisätyn todellisuuden sovelluskohteita. Unreal Enginen ja ARCoren kanssa työskentelyssä rajoitteina huomattiin dokumentaation vähyys ja päivittämättömyys. Uusimpien ohjelmistoversioiden valinnan myötä virallinen dokumentaatio ei ollut ajantasaista ja vaikutti siihen, että ARCoren dokumentaatiota ei voitu suoraan käyttää kehittämistyön apuna. Havaittiin, että Unity-pelimoottorille tehdyn ARCore-dokumentaatio oli ajantasainen ja määrä oli Unreal Engineen verrattuna moninkertainen. ARCoren julkaisuhistoriamuistioiden avulla löydettiin käytöstä poistuneita funktioita ja menetelmiä, sekä näitä korvaavat menetelmät. Yhdistettynä vanhentuneeseen dokumentaatioon ja tutustumalla Blueprintin ohjelmointitutoriaaleihin onnistuin ratkaisemaan kohdatut ongelmat. Vanhentunut dokumentaatio hidasti työn kehitysprosessia ja teki ohjelmoinnista paikoin haastavaa. Sovelluksen kehittämisen edetessä visuaalisen ohjelmoinnin menetelmien ja logiikan ymmärtäminen hioutui paremmaksi. Opinnäytetyön lähteitä tutkiessa huomattiin, että Unreal Enginen visuaalisella ohjelmointikielellä Blueprintillä oli luotu kokonaisia pelejä.

Saatoin sen vuoksi olettaa, että myös AR-sovelluksen ohjelmoiminen olisi mahdollista. Käyttöliittymien toiminnallisuuksien ohjelmointi oli suoraviivaista ja nopeaa. Tasokohtaisia Blueprintejä ohjelmoidessa kohdattiin ongelmia, jotka johtuivat kokemattomuudestani tekniikan käytössä. Suurin kohdattu ongelma visuaalisessa ohjelmoinnissa oli vaikeus löytää haluttuja funktioita. Ongelma ratkesi peliohjelmointitutoriaalin avulla. Löydettiin, että haluttu funktio saatiin käyttöön, kun edellisessä vaiheessa palautuvaa tietoa verrattiin ensin kuuluvaksi kohdefunktion luokkaan. Jos palautunut tieto ja funktio jakoivat saman luokan, niin luokan funktioita voitiin käyttää. Tutkimuksen mukaan visuaalinen ohjelmointi häviää tehokkuudessaan C++-ohjelmointikielelle, mutta on nopeampaa luoda ja AR-sovelluksen ohjelmointi onnistui sitä käyttäen hyvin.

Vaikka opinnäytetyön tulokset saavutettiin ja ne olivat onnistuneita, tulevaisuudessa teknologian vaihto voi tulla kyseeseen. Unity on maailmalla käytetympi teknologia ja yhteisö laajempi, siksi Unityn dokumentaatio on kehittyneempää ja ajantasaista. Jos tulevaisuudessa ohjelmoidaan tekstipohjaisella ohjelmointikielellä, Unityn C#-ohjelmointikieli on minulle entuudestaan tuttu ja saattaa vaikuttaa ohjelmistojen valintaan. Blueprint-ohjelmointikielen todettiin olevan erittäin hyvä työkalu ja sen käyttöä tullaan jatkamaan seuraavissa visuaalista ohjelmointia käyttävissä projekteissa. Kehitysprojektin jatkokehittäminen valmiiksi tuotteeksi asti on mahdollista. Nyt sovellus vain esittelee käytetyn teknologian ja yhden mahdollisen sovellutuksen. Sovelluksessa käytettyä kuvantunnistustekniikkaa voisi hyödyntää esimerkiksi erilaisissa näyttelyissä ja kulttuurikohteissa. AR-teknologian tutkiminen herätti mielenkiinnon virtuaalisen todellisuuden tuotteen kehittämiseen. Unreal Engine soveltuu myös VR-tuotannon ohjelmistokehykseksi.

Lähteet

Argodesign. (n.d.). *Interactive Light.* Noudettu osoitteesta https://www.argodesign.com/work/interactive-light.html

Busby, J.;Parrish, Z.;& Wilson, J. (2009). *Introduction to Unreal Technology.* Sams. Noudettu osoitteesta https://www.informit.com/articles/article.aspx?p=1377834

Caudell, T.;& Mizell, D. (1992). *Augmented Reality: An Application of Heads-Up Display Technology to Manual.* Boeing Computer Services, Research and Technology.

Chow, T. (2017). *Guiding Light: What We Learned from Mixed Reality Prototyping*. Noudettu osoitteesta https://argodesign.medium.com/guiding-light-what-we-learned-from-mixed-reality-prototyping-58a4fbbeb0ae

Collins, A. (2018/2020). *The Ultimate Guide to Augmented Reality*. Noudettu osoitteesta https://blog.hubspot.com/marketing/augmented-reality-ar

Craig, A. (2013). *Understaning Augmented Reality.* Elsevier.

Denham, T. (n.d.). *What is Unreal Engine?* Retrieved from https://conceptartempire.com/what-is-unreal-engine/

ESA. (2017). *Reality check.* Noudettu osoitteesta http://www.esa.int/spaceinimages/Images/2017/07/Reality_check

Game-Ace. (2019). *Understanding Unreal Engine Blueprint*. Noudettu osoitteesta https://game-ace.com/blog/unreal-engine-blueprints/

GameMaker Studio 2. (2019). *Changing DnD™ To Code*. Noudettu osoitteesta https://docs2.yoyogames.com/index.html?page=source%2F_build%2Findex.html

Godot community. (2014/2020). *Getting started with Visual Scripting*. Noudettu osoitteesta https://docs.godotengine.org/fi/stable/getting_started/scripting/visual_script/getting_started.html

Google. (2019). *Google ARCore SDK for Unreal.* Noudettu osoitteesta https://github.com/google-ar/arcore-unreal-sdk

Google ARCore. (n.d.-a). *Fundamental Concepts*. Noudettu osoitteesta https://developers.google.com/ar/discover/concepts

Google ARCore. (n.d.-b). *Using ARCore to light models in a scene*. Noudettu osoitteesta https://developers.google.com/ar/develop/java/light-estimation

Google ARCore. (n.d.-c). *Cloud Anchors overview for Unreal Engine*. Noudettu osoitteesta https://developers.google.com/ar/develop/unreal/cloud-anchors/overview-unreal

Google ARCore. (n.d.-d). *Quickstart for Unreal*. Noudettu osoitteesta https://developers.google.com/ar/develop/unreal/quickstart

Google-Developers. (2020). *ARCore supported devices*. Noudettu osoitteesta https://developers.google.com/ar/discover/supported-devices

Gupton, N.;& Kiger, P. (2020). *What's the difference between AR,VR and MR?* Noudettu osoitteesta https://www.fi.edu/difference-between-ar-vr-and-mr

Interesting Engineering. (2016). *How Do Game Engines Work?* Noudettu osoitteesta https://interestingengineering.com/how-game-engines-work

Johnson, D. (2020). *What is augmented reality? Here's what you need to know about the 3D technology*. Noudettu osoitteesta https://www.businessinsider.com/what-is-augmented-reality?r=US&IR=T

Le, J. (2018). *Snapchat's Filters: How computer vision recognizes your face*. Noudettu osoitteesta https://medium.com/cracking-the-data-science-interview/snapchats-filters-how-computer-vision-recognizes-your-face-9907d6904b91

Machado, D. (2017). *Why we choose Godot Engine*. Noudettu osoitteesta https://medium.com/rock-milk/why-godot-engine-e0d4736d6eb0

Mann, S. (2000). *A GNU/Linux Wristwatch Videophone*. Noudettu osoitteesta https://www.linuxjournal.com/article/3993

Mann, S. (n.d.). *Five significant contributions to research*. Noudettu osoitteesta http://wearcam.org/contributions.pdf

Mannlab. (2020). *EyeTap*. Noudettu osoitteesta https://mannlab.com/eyetap

me3D31337. (2011). *Loading Mission 3 Rrajigar Mine.* Noudettu osoitteesta https://www.mobygames.com/game/macintosh/unreal/screenshots/gameShotId,516601/

Mikepanhu. (2014). *Google Glass with frame for prescription lens.* Noudettu osoitteesta https://commons.wikimedia.org/w/index.php?curid=32348020

Nutt, C. (2014). Epic radically changes licensing model for Unreal Engine. *Gamasutra*. Noudettu osoitteesta https://www.gamasutra.com/view/news/213517/Epic_radically_changes_licensing_model_for_Unreal_Engine.php

Petty, J. (n.d.). *What is Unity 3d & What is it Used For?* Noudettu osoitteesta https://conceptartempire.com/what-is-unity/

Ro, Y.;Brem, A.;& Rauschnabel, P. (2017). *Augmented Reality Smart Glasses: Definition, Concepts and Impact on Firm Value Creation.*

Rolland, J.;Holloway, R.;& Fuchs, H. (1994). *A comparison of optical and video see-through head-mounted displays.* University of North Carolina: Department of Computer Science .

Square Enix. (2021). *FF7RINTERGRADE March Screenshot 05.* Noudettu osoitteesta https://press.na.square-enix.com/FINAL-FANTASY-VII-REMAKE#?tab=screenshots-2

Sutherland, I. (1968). *A head-mounted three dimensional display.* The University of Utah.

Sutorcen. (2016). *This is Unreal Engine*. (Hotgates, Toimittaja) Noudettu osoitteesta https://hotgates.eu/this-is-unreal-engine/

Sweeney, T. (2015). If You Love Something, Set It Free. *Unreal Engine Blog*. Noudettu osoitteesta https://www.unrealengine.com/en-US/blog/ue4-is-free

t053kr. (2020). *Ambient LED via BT*. GitHub. Noudettu osoitteesta https://github.com/t053kr/Portfolio/tree/main/Arduino/Cluster%20Communication%20Port/Ambient%20LED%20via%20BT

Toftedahl, M.;& Engström, H. (2019). *A Taxonomy of Game Engines and the Tools that Drive the Industry*. Noudettu osoitteesta https://www.researchgate.net/publication/337165844_A_Taxonomy_of_Game_Engines_and_the_Tools_that_Drive_the_Industry

Turk, V. (2017). *Video projector creates augmented reality with no bulky headset*. Noudettu osoitteesta https://www.newscientist.com/article/2126430-video-projector-creates-augmented-reality-with-no-bulky-headset/

Unity Asset Store. (n.d.-a). *Ludiq*. Noudettu osoitteesta https://assetstore.unity.com/publishers/11548

Unity Asset Store. (n.d.-b). *Bolt*. Noudettu osoitteesta https://assetstore.unity.com/packages/tools/visual-scripting/bolt-163802

Unity. (n.d.-a). *Plans and pricing*. Noudettu osoitteesta https://store.unity.com/#plans-business

Unity. (n.d.-b). *How to make a video game without any coding experience*. Noudettu osoitteesta https://unity.com/how-to/make-games-without-programming#visual-scripting-playmaker

Unreal Engine. (n.d.-a). *Frequently Aked Questions (FAQ)*. Noudettu osoitteesta https://www.unrealengine.com/en-US/faq

Unreal Engine. (n.d.-b). *Plugins*. Noudettu osoitteesta https://docs.unrealengine.com/en-US/ProductionPipelines/Plugins/index.html

Unreal Engine. (n.d.-c). *Types of Blueprints*. Noudettu osoitteesta https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/index.html

Unreal Engine. (n.d.-d). *Guidelines for Programming for Blueprints*. Noudettu osoitteesta https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/TechnicalGuide/Guidelines/index.html

Unreal Engine. (n.d.-e). *Setting Up Android SDK and NDK for Unreal*. Noudettu osoitteesta https://docs.unrealengine.com/en-US/SharingAndReleasing/Mobile/Android/Setup/AndroidStudio/index.html

Wearcam. (n.d.). *Synthetic Synesthesia of the Sixth Sense™*. Noudettu osoitteesta http://wearcam.org/6ense.htm

Liite 1: Aineistonhallintasuunnitelma

Tämä on opinnäytetyön aineistonhallintasuunnitelma. Tässä liitteessä kuvataan raporttiin kerättävän aineiston luonne sekä säilytystapa. Tekijä sitoutuu säilyttämään kerättyä aineistoa tietoturvallisesti vuoden ajan opinnäytetyön valmistumisesta, jonka jälkeen se voidaan tuhota korkeakoulun ohjeistuksen mukaisesti.

Aineiston luonne:

Kehitysprojektissa syntyvä aineisto on tekijän itsensä kehittämää. Se ei sisällä salassa pidettäviä henkilötietoja tai yrityssalaisuuksia. Tietoperustaan kerättävä aineisto merkitään lähdeluetteloon APA-merkintätapaa käyttäen.

Kehitysprojektin aineiston kerääminen:

Kehitysprojektin aikana kirjoitetaan muistiinpanoja talteen päiväkirjamuotoisesti. Kirjoitetun aineiston pohjalta koostetaan opinnäytetyöraportin toiminnallinen osa.

Aineiston säilytys:

Muistiinpanoja sekä raporttia säilytetään kirjoitusvaiheessa tekijän henkilökohtaisella tietokoneella, joka on suojattu luvattomalta käytöltä. Varmuuskopiot kirjoitetusta aineistosta tallennetaan ajoittain tekijän henkilökohtaiseen pilvipalveluun ja viikoittainen kopio Wihi-palveluun.

Opinnäytetyön julkaisun jälkeen kerättyä aineistoa säilytetään tekijän tietokoneella ja sen varmuuskopiota ulkoisella tallennusmedialla vuoden ajan.

Oikeudet:

Opinnäytetyöraportin tekijä omistaa oikeudet kehitysprojektissa tuotettuun aineistoon ja sen tuloksiin.