Riku Marttinen

IOS ARKITIN LISÄTTY TODELLISUUS SOVELLUSKEHITYKSESSÄ

IOS ARKITIN LISÄTTY TODELLISUUS SOVELLUSKEHITYKSESSÄ

Riku Marttinen
Opinnäytetyö
Kevät 2019
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä: Riku Marttinen
Opinnäytetyön nimi suomeksi: iOS ARKitin lisätty todellisuus sovelluskehityksessä
Opinnäytetyön nimi englanniksi: iOS ARKit's augmented reality in software development
Työn ohjaaja: Kari Laitinen
Työn valmistumislukukausi ja -vuosi: Kevät 2019
Sivumäärä: 29

Opinnäytetyössä tutkitaan lisättyä todellisuutta, erityisesti mobiilikehityksessä. Käydään läpi, miten luodaan lisätyn todellisuuden sovellus ja mitä ominaisuuksia mobiililaitteiden valmistajat mahdollistavat laitteillensa. Lisätyn todellisuuden mobiilikehitykseen on kehitetty iOS:lle ja Androidille omat työkalut, ja tässä opinnäytetyössä syvennytään ARKit-nimiseen kirjastoon, jonka Apple on kehittänyt iOS-laitteille.

Opinnäytetyössä käsitellään ensin käsitettä lisätty todellisuus ja sen tarkoitusta. Tästä edetään nykyaikaiseen tapaan luoda lisättyä todellisuutta ja tutkitaan ominaisuuksia, joita ARKit sisältää. Lopuksi esitellään demosovellus, jossa käytetään ARKitin ominaisuuksia hyväksi ja luodaan käytännöllinen mobiilisovellus.

Opinnäytetyön aikana todettiin ARKitin hyödyllisyys mobiilisovellusten kehitystyössä. Sillä voidaan kehittää lisättyä todellisuutta sekä uusiin että jo olemassa oleviin sovelluksiin. Opinnäytetyön lopputulemana todetaan, että ARKit tekee lisätyn todellisuuden lisäämisestä sovelluksiin helpoksi

---

Asiasanat: ARKit, lisätty todellisuus, mobiiliohjelmointi

ABSTRACT

Oulu University of Applied Sciences
Information technology, Software engineering

Author: Riku Marttinen
Title of thesis: iOS ARKit's augmented reality in software development
Supervisor: Kari Laitinen
Term and year when the thesis was submitted: Spring 2019
Pages: 29

This thesis examines augmented reality, especially in mobile development. How to create an augmented reality application and what features mobile-device manufacturers have created for their devices. For augmented reality mobile manufacturers have their own tools. In this thesis, we will be focusing in ARKit developed by Apple for iOS devices.

The thesis first goes through the concept of augmented reality and its purpose. From there we will look into modern way of creating augmented reality and explore the features that ARKit contains. Finally, a demo application is presented with ARKit features to create a practical mobile app.

During the thesis work, the utility of ARKit in modern mobile applications was found for both, ARKit only based applications and in the already existing applications as a drop of new life. Thesis outcome is that ARKit makes it easy to add augmented reality to applications.

Keywords: ARKit, Augmented reality, modern mobile development

ALKULAUSE

Aluksi tahdon kiittää Cubicasa Oy:tä tämän työn mahdollistamisesta ja myös mallintaja Niko Mäntyä, joka suunnitteli ja loi käytettävät 3D-objektit. Tämä työ tuki minun vakituista työtäni mobiilikehittäjänä ja syvensi tietämystäni lisätystä todellisuudesta.

Oulussa 3.4.2019

Riku Marttinen

SISÄLLYS

TIIVISTELMÄ 3
ABSTRACT 4
ALKULAUSE 5
SISÄLLYS 6
SANASTO 7
1 JOHDANTO 8
2 AUGMENTED REALITY – LISÄTTY TODELLISUUS 9
2.1 Historia 9
2.2 Apple ARKit 10
3 ARKITIN OMINAISUUKSIA 11
3.1 ARKitin käyttäjän seuraaminen 3D-tilassa 11
3.2 Laajennetun todellisuuden jakaminen 12
3.3 Lisätyn todellisuuden tallentaminen ja siihen palaaminen 13
4 LISÄTYN TODELLISUUDEN MOBIILISOVELLUKSEN KEHITTÄMINEN 14
4.1 Projektin aloitus 14
4.2 Objektien asettaminen ja vuorovaikuttaminen 16
4.3 Lisätyn todellisuuden tilan tallentaminen ja palauttaminen 17
5 ESIMERKKI-SOVELLUS 19
5.1 Demosovelluksen kuvaus 19
5.2 Sovelluksen lisätyn todellisuuden tilan vaihtaminen 20
5.3 Objektien luonti ja niiden asettaminen 23
5.4 Mittojen saaminen ja niiden tallennus 26
6 YHTEENVETO 28
LÄHTEET 29

SANASTO

ARCNView – Näkymä, joka mahdollistaa lisätyn todellisuuden iOS-laitteilla

ARKit – Applen kehittämä lisätyn todellisuuden työkalu

Main.Storyboard – Xcodessa käyttäjärajapinnan suunnitteluun käytettävä graafinen näkymä

Noodi – Kolmeulotteinen esine

SDK – Software Development Kit on mobiilisovellus kehittämiseen tarkoitettu työkalu

Swift – Applen kehittämä olio-ohjelmointikieli

ViewController – Pääkontrolleri, josta näkymän koodi suoritetaan

Xcode – Applen tekemä sovellus sovelluskehitykseen

1 JOHDANTO

Augmented reality, lisätty todellisuus ja laajennuttu todellisuus tarkoittavat kaikki samaa asiaa: tietokoneen keinotekoisesti lisättyä tietoa todellisessa ympäristössä. Tässä opinnäytetyössä tätä todellisuutta tarkastellaan kameran läpi puhelimen ruudussa. Uusi teknologia kiinnostaa monia, mutta ei ole vielä varmaa, mihin kaikkeen tätä teknologiaa voidaan käyttää. Esimerkiksi suosittu mobiilipeli Pokemon Go! hyödyntää lisätty todellisuusteknologiaa, jolla lisätään todelliseen maailmaan Pokemon-hahmoja, jotka voidaan sitten napata puhelimen avulla pelissä.

Lisätty todellisuus kuulostaa uudelta käsitteeltä, mutta se on ollut olemassa jo yli puoli vuosisataa muodossa tai toisessa. Nykymaailmassa lisätty todellisuus on helposti lähestyttävissä käyttäjälle, jopa omassa mobiililaitteessa. Googlen kehittämä lisätyn todellisuuden kehitystyökalu ARCore on Android-laitteille ja Applen vastine tälle, ARKit on iOS-laitteille ja molemmat yritykset yrittävät rohkaista kehittäjiä luomaan uusia mielenkiintoisia sovelluksia näiden avulla. Molemmissa on mielenkiintoisia ominaisuuksia, mutta lisätty todellisuus ei ole saavuttanut vielä mitään vakiintunutta roolia, joten sitä voi kuvitella muovailuvahana, josta voi luoda mitä tahansa tällä hetkellä.

Tässä opinnäytetyössä tarkastellaan, miten Applen ARKit soveltuu nykyaikaisten sovelluksien kehittämiseen. Käydään läpi myös, mitä ominaisuuksia ARKit sisältää ja miten niitä sovelletaan ja käytetään ohjelmoitaessa lisättyä todellisuutta hyödyntävää sovellusta.

Opinnäytetyön lopussa esitellään demosovellus, jossa demonstroidaan ARKitin ominaisuuksia käytännössä. Sovelluksessa on näkymiä, joissa mainitut ominaisuudet toimivat yhdessä ja erikseen demonstroiden lisätyn todellisuuden mahdollisuuksia.

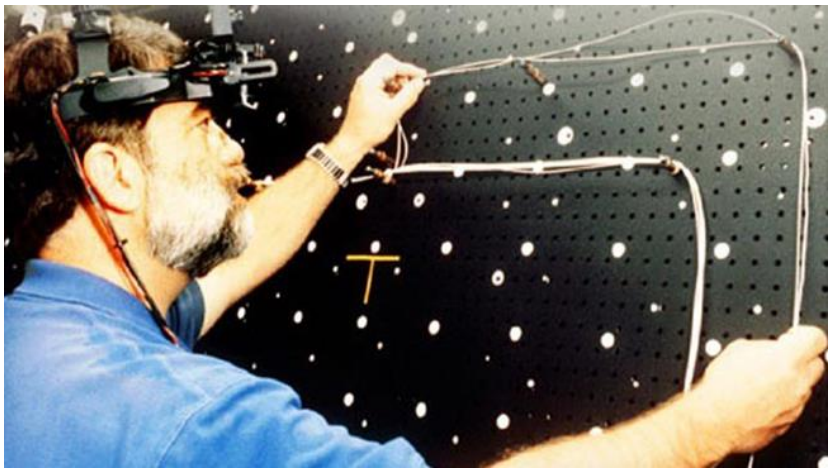
2 AUGMENTED REALITY – LISÄTTY TODELLISUUS

Lisätty todellisuus on tietokoneen keinotekoisesti lisäämää virtuaalista sisältöä oikeaan maailmaan. Lisätty todellisuus ei siis tarkoita virtuaalista todellisuutta. Virtuaalisessa todellisuudessa kaikki, mitä voi nähdä tai kokea, on täysin keinotekoista, kun taas lisätyssä todellisuudessa voidaan nähdä tai kokea virtuaalisia asioita oikeassa maailmassa.

2.1 Historia

Lisätty todellisuus on ollut olemassa jo vuodesta 1968 kun Ivan Sutherland kehitti ensimmäisen päähän asetettavan monitorijärjestelmän. Tällä järjestelmällä hän onnistui näyttämään tietokoneen tekemää grafiikkaa järjestelmän käyttäjille. Tällöin ei kuitenkaan termi lisätty todellisuus ollut vielä olemassa. (1.)

Vuonna 1990 tutkija Tom Caudell määrättiin kehittämään vaihtoehtoinen tapa ohjata työntekijöitä Boeingin tehtaalla. Hän keksi tavan, miten työntekijät voivat nähdä lentokoneen piirustukset päähän laitettavalla laitteella ja uudelleen käytettävillä tasoilla, joihin lisätty todellisuus heijastui. Tämän keksittyään hän toi maailmalle käsitteen lisätty todellisuus. (2.)



KUVA 1. Tom Caudell keksintönsä kanssa (5)

Vuonna 1999 Nara Institute of Science and Technologyn Hirokazu Kato julkaisi ohjelmiston nimeltä ARToolKit avoimen lähteen yhteisölle. Tämä mahdollisti videon jäljentämisen oikeasta maailmasta ja lisäämään siihen virtuaalisia asioita.

Ohjelmiston avulla monet saivat mahdollisuuden kokeilla lisättyä todellisuutta. (2.)

Nykyään lisätyn todellisuuden kokeminen on helpompaa kuin koskaan. Suurin osa nykyaikaisista puhelimista tukee joko Applen kehittämää ARKitiä tai Googlen vastinetta ARCorea. Apple julkaisi ARKit 1.0 -version iOS 11 mukana 19. syyskuuta 2017, ja sitä kehitetään edelleen. (3.)

2.2 Apple ARKit

Applen kehittämä ARKit yksinkertaistaa lisätyn todellisuuden soveltamisen iOS-pohjaisille sovelluksille ja peleille. ARKit mahdollistaa tämän käyttämällä iOS-laitteen liikesensoreita ja kameraa. ARKitistä löytyy monta eri ominaisuutta, joilla saadaan aikaiseksi entistä responsiivisempi kokemus käyttäjälle. (4.)

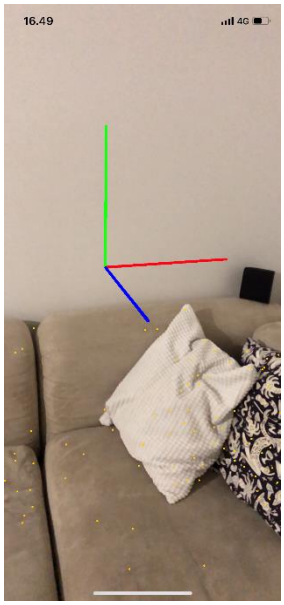
ARKit on pohjimmillaan kehitystyökalu, jonka avulla kehittäjät voivat kehittää lisätyn todellisuuden sovelluksia. ARKit löytyy Xcoden kirjastosta iOS 11 SDK:sta lähtien ja sitä tukevat kaikki iOS-laitteet joissa on A9 tai uudempi prosessori.

Uudemmassa iOS 12 käyttöjärjestelmässä julkaistiin ARKit 2.0, joka mahdollistaa jaetun lisätyn todellisuuden kokemuksen. Se tarkoittaa sitä, että kahdella eri laitteella voidaan nähdä sama lisätty todellisuus. Tämän lisäksi lisätyn todellisuuden tila voidaan tallentaa ja jatkaa myöhemmin. Aikaisemmin tämä ei ollut mahdollista.

ARKitillä kehittäminen on mahdollista kahdella Applen kehittämällä ohjelmointikielellä, Objective-C:llä ja Swiftillä. Tässä opinnäytetyössä käytetään Swiftiä, koska se on Applen suosittelema kieli uusien iOS-applikaatioiden kehitykseen. Swift on oliopohjainen kieli ja sen etuja on koodin luettavuus ja valtavat kirjastot.

3 ARKITIN OMINAISUUKSIA

ARKit mahdollistaa iOS-laitteissa molempien kameroiden käytön lisätyn todellisuuden visualisoimiseen. ARKit kartoittaa ja seuraa oikean maailman tilaa, missä käyttäjä kulkee, ja pystyy lisäämään asioita tähän luotuun 3D-tilaan. ARKit-sovellus luo avautuessaan 3D-koordinaatiston, jossa keskipiste (0,0,0) määräytyy laitteen paikasta. Laitetta seurataan näiden koordinaattien avulla koko ajan. Tämä keskipiste (kuva 2) voidaan tuoda näkymiin käyttämällä virheenkorjausominaisuuksia.



KUVA 2. Keskipiste sovelluksessa

3.1 ARKitin käyttäjän seuraaminen 3D-tilassa

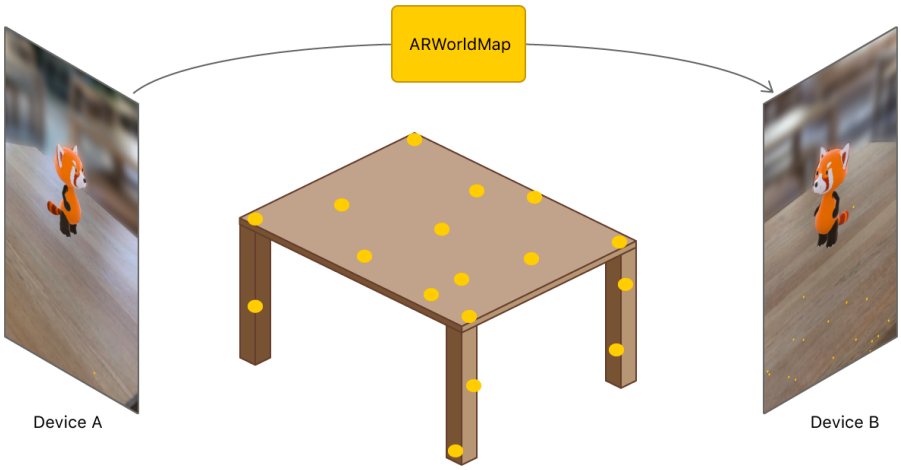
ARKit käyttää ”visual-inertial odometry”-nimistä menetelmää. Tämä menetelmä yhdistää iOS-laitteen liikkeen havainnoinnin ja tietokonenäön analyysin siitä, mitä laitteen kamerat näkevät. ARKit tunnistaa seinät, lattiat ja esineet tilassa kameran avulla, seuraa niiden paikkaa läpi kameran tuottamien kuvaruutujen ja vertailee niiden tuottamaa informaatiota liikesensoreiden dataan. Tätä hyödyntämällä käyttäjä voi vuorovaikuttaa ympäristöönsä esimerkiksi asettamalla virtuaalisen objektin lattialle. Sen paikka pysyy samana käyttäjän havainnoimassa tilassa, vaikka käyttäjä siirtyisi muualle. (4.)

Teknisesti ARKit etsii piirtopisteitä näkyvästä tilasta ja sitoo ne luotuun koordinaatistoon, näin muodostaen eräänlaisen pistepilven esimerkiksi esineistä, lattiasta, seinästä ja jopa katosta. Pisteiden avulla pystytään saamaan selvä 3D-ympäristö käyttäjän ympärille ja luomaan tila, missä käyttäjä on. Kuvassa 3 havainnollistetaan, miten piirtopisteet asettuvat ympäröivään tilaan.

KUVA 3. Keltaiset pisteet ovat ARKitin havaitsemia piirtopisteitä

3.2 Laajennetun todellisuuden jakaminen

ARKit mahdollistaa lisätyn todellisuuden jakamisen useiden laitteiden kanssa. Se toimii sillä periaatteella, että on olemassa vanhempi laite, joka luo 3D-tilan sovelluksessa, johon muut laitteet voivat liittyä. Tämän jälkeen ARKit-sovellus jakaa jatkuvasti informaatiota laitteiden kesken esimerkiksi siitä, missä käyttäjät sijaitsevat ja mitä heidän laitteensa havainnoivat ympäröivästä tilasta, ja lisää 3D-tilan dataa ympäröivästä tilasta. Kuvassa 4 nähdään, miten esimerkiksi kaksi laitetta näkevät saman virtuaalisen objektin pöydän päällä.



KUVA 4. Kaksi eri laitetta tarkastelee samaa objektia eri paikasta (4)

3.3 Lisätyn todellisuuden tallentaminen ja siihen palaaminen

ARKitin yhtenä ominaisuutena on lisättyyn todellisuuteen palaaminen. iOS-sovellus pystyy tallentamaan ja jatkamaan samasta tilasta, mihin se jäi edellisellä kerralla. Tämän saavuttaakseen käyttäjän pitää luoda 3D-tila jossain paikassa, mihin hän voi palata uudestaan. Tätä ominaisuutta käytettäessä ARKit tallentaa tilan sovellukseen ja kuvan, missä käyttäjä oli tallentaessaan tilan. Parhaan tilan palautukseen käyttäjän tulee olla samassa kohdassa, joka on yhteensopiva ARKitin tallentamassa kuvassa. Tätä esimerkiksi voidaan käyttää huoneen sisustuksen havainnollistamisessa.

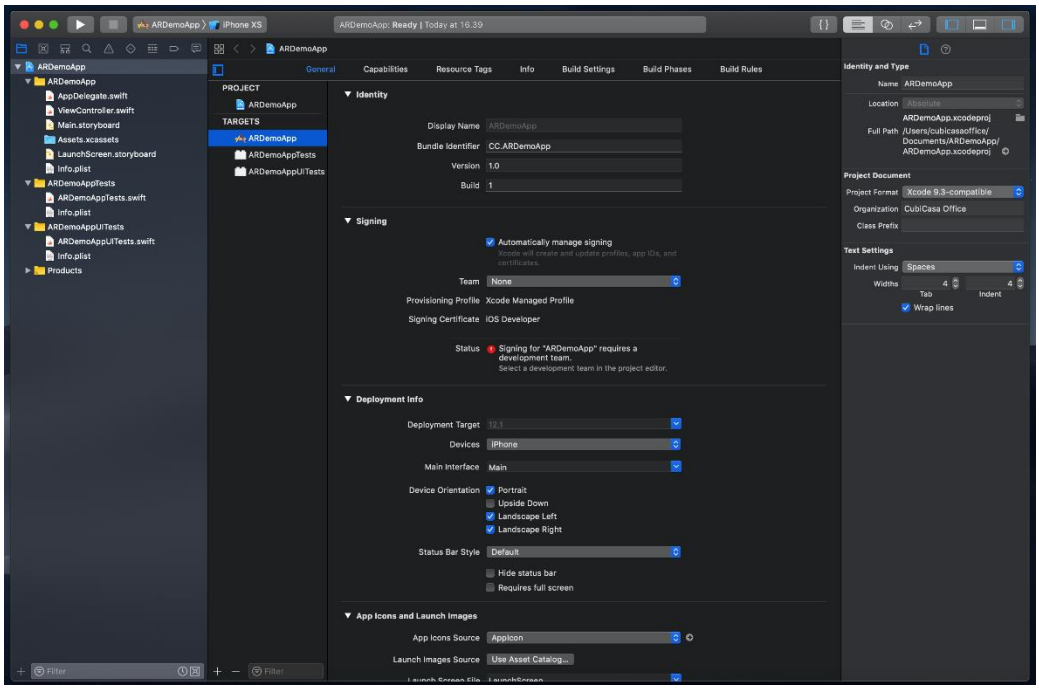
4 LISÄTYN TODELLISUUDEN MOBIILISOVELLUKSEN KEHITTÄMINEN

Lisätyn todellisuuden mobiilisovelluksen kehittämiseen on kaksi vaihtoehtoa. Valmistajien eri laitteisiin käyvät eri työkalut, iOS-laitteiden sovelluksiin soveltuu ARKit ja Android pohjaisille laitteille ARCore. Tässä opinnäytetyössä perehdytään ARKitin käyttöön.

Sovelluskehitykseen tarvitaan iOS-laite, macOS-käyttöjärjestelmällä toimivan tietokone ja Applen luoma kehittämisympäristö, joka on nimeltään Xcode. Xcode pitää olla päivitettynä minimissään iOS 11.0 -versioon ja iOS-laitteen tulee olla myös päivitettynä vähintään samaan versioon. Laitteen prosessorin pitää olla vähintään A9-malli tai uudempi. Kuitenkin suositus on, että kaikki versiot ovat uusimpia ja päivitettyjä.

4.1 Projektin aloitus

Kun tarvittavat vaatimukset on täytetty, voi alkaa kehittämään sovellusta. Ensimmäiseksi avataan Xcode ja aloitetaan uusi projekti. Valitaan projektin asetukset ja tässä tapauksessa valitaan Single-view application eli yhden näkymän sovellus. Nimetään projekti ja Xcode luo tarvittavat tiedostot kuten UI:n graafisen luomiseen tarkoitetun Main.storyboard ja Info.plist, jossa on kaikki tarvittava tieto suoritettavista tehtävistä, jotka menevät kääntäjälle. Kuvassa 5 on näkymä konfiguroidusta projektista Xcodessa.



KUVA 5. Xcode

Siirrytään Main.storyboardiin ja sijoitetaan elementeistä ARKit SceneKit View niminen näkymä sovellukseen. Tämän jälkeen luodaan yhteyden Main.storyboardista itse suoritettavaan koodiin, joka löytyy ViewController nimisestä Swift-tiedostosta. Nimetään näkymä sceneView'ksi ja kun näkymä on yhdistetty kontrolleriin, Xcode osaa yhdistää suoritettavan koodin oikeaan UI-komponenttiin. Kontrollerissa se yhdistyy seuraavalla tavalla.

```
@IBOutlet weak var sceneView: ARSCNView!
```

Tämän jälkeen alustetaan näkymä, että ARKit käynnistyy ja tämä tapahtuu viewDidLoad-funktion sisällä. Lisätään testausmenetelmiä varmistamaan, että näkymä todella toimii. Kuvassa 6 on valmis koodi, jolla voidaan testata toimivuus.

```
import UIKit
import ARKit
class ViewController: UIViewController {
    @IBOutlet weak var sceneView: ARSCNView!

    override func viewDidLoad() {
        super.viewDidLoad()
        let configuration = ARWorldTrackingConfiguration()
        sceneView.debugOptions = [.showWorldOrigin, .showFeaturePoints]
        sceneView.session.run(configuration)
    }
}
```

KUVA 6. Kontrollerin koodi, kun sovellus on alustettu

Kun kaikki toimii, voi alkaa kehittämään lisätyn todellisuuden mobiilisovellusta ARKitin avulla.

4.2 Objektien asettaminen ja vuorovaikuttaminen

Virtuaalisten asioiden lisääminen onnistuu helposti. Ensin pitää luoda ohjelmallisesti objekti, joka halutaan lisätä maailmaan. ARKit mahdollistaa monen geometrisen perusmuodon ja asian luomisen, esimerkiksi voidaan tehdä palloja, neliötä ja sylintereitä. Lisättäessä objekteja näkymään niille voidaan määrittää suunta, väri, koko, painovoima ja monta muuta parametriä, jotka auttavat tekemään laajennetun todellisuuden kokemuksen paremmaksi. Kuvassa 7 nähdään, miten luodaan pallo ja asetetaan se näkyväksi käyttäjälle kutsumalla funktiota addObject.

```
func addObject(){
    let ball = SCNSphere.init(radius: 0.75)
    ball.firstMaterial?.diffuse.contents = UIColor.white
    ball.firstMaterial?.lightingModel = .constant
    ball.firstMaterial?.isDoubleSided = true

    let node = SCNNode.init(geometry: ball)
    node.position = SCNVector3.init(1, 0, 0)
    sceneView.scene.rootNode.addChildNode(node)
}
```

KUVA 7. Pallon luomiseen ja näkymään lisäämiseen tarvittava koodi

Yksi tärkeimpiä ominaisuuksia, joita ARKit mahdollistaa, on käyttäjän ja lisätyn todellisuuden vuorovaikutus. ARKitistä löytyy ominaisuus, jolla pystyy tulkitsemaan iOS-laitteen näytöllä tapahtuneen kosketuksen, missä kohdin se koskee 3D-tilassa olevaa tasoa, pistettä tai virtuaalista objektia. Tämän avulla käyttäjä

voi asettaa, siirtää ja muokata asioita näkymässään vapaasti. Kuvassa 8 on demonstroitu, miten tämä toimii.

```
func realWorldVector(screenPosition: CGPoint) -> SCNVector3? {
    let results = self.hitTest(screenPosition, types: [.existingPlane, .featurePoint])
    guard let result = results.first else { return nil }

    return SCNVector3.positionFromTransform(result.worldTransform)
}
```

KUVA 8. Koodi, jolla 3D-tilan koordinaatti saadaan ruudun kosketuspisteestä

Tätä voidaan käyttää esimerkiksi objektien asettamiseen ARKitin havaitsemien pisteiden tai tasojen kohdalla, ja myös asetettujen objektien tunnistamiseen ja koskemiseen.

4.3 Lisätyn todellisuuden tilan tallentaminen ja palauttaminen

Luodun lisätyn todellisuuden tallentaminen on hyödyllinen ominaisuus sovelluksessa. Voidaan esimerkiksi tallentaa pelin tila tai jonkinlainen demonstraatio, miltä huoneisto voi näyttää sisustuksen kanssa. Yksinkertaisuudessaan voidaan luoda funktio, joka tallentaa nykyisen näkymän kuvan 9 tavalla.

```
func saveState(){
    sceneView.session.getCurrentWorldMap { worldMap, error in
        guard let map = worldMap
            else {
                print("Can't get current world map error: \(error!.localizedDescription)")
                return
        }
        let mapUrl = NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask, true)[0] + "/worldMap"
        let mapURL = URL(string: mapUrl)
        do {
            let data = try NSKeyedArchiver.archivedData(withRootObject: map, requiringSecureCoding: true)
            try data.write(to: mapURL!, options: [.atomic])
        } catch {
            print("Can't save map: \(error.localizedDescription)")
        }
    }
}
```

KUVA 9. Koodi tilan tallentamiseen

Kuvassa 9 lisätyn todellisuuden kartta tallennetaan iOS-laitteen paikalliseen muistiin sovelluksen tiedostot-kansioon. Tilan palauttamiseen tarvitaan vielä kaksi funktiota, yksi datan hakemiseen ja toinen sen purkamiseen. Kuvassa 10 on koodi datan hakemiseen.

```
func retrieveData(url : URL) -> Data?{
    do{
        return try Data(contentsOf: url)
    }catch{
        print("No data found")
    }
    return nil
}
```

Kuva 10. Yksinkertainen datan haku URL:sta

Kuvassa 11 puretaan laajennetun todellisuuden kartta käytettäväksi. Sen jälkeen, kun kartta on käytettävissä, voidaan seuraavalla koodilla palauttaa tallennetun tilan takaisin näkymään.

```
let configuration = self.defaultConfiguration
configuration.initialWorldMap = worldMap
sceneView.session.run(configuration, options: [.resetTracking, .re-
moveExistingAnchors])
```

```
func loadState(data : Data) -> ARWorldMap{
    do {
        guard let worldMap = try NSKeyedUnarchiver.unarchivedObject(ofClass: ARWorldMap.self, from: data)
            else { fatalError("No ARWorldMap in archive.") }
        return worldMap
    } catch {
        fatalError("Can't unarchive ARWorldMap from file data: \(error)")
    }
}
```

KUVA 11. Datan purkamiseen tarvittava koodi, joka palauttaa toimivan kartan

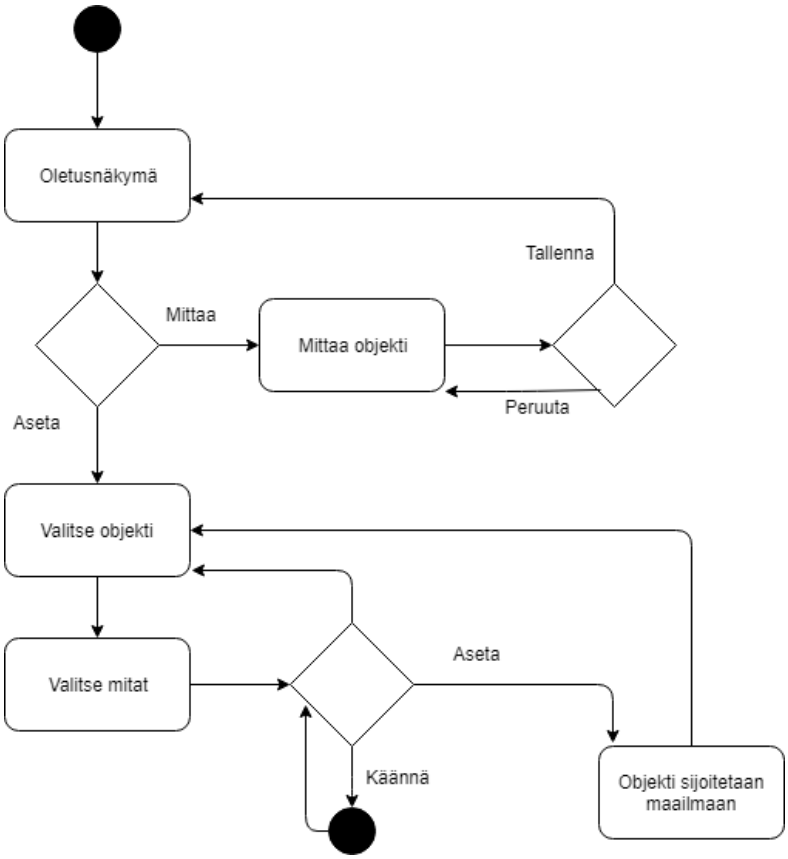
5 ESIMERKKI-SOVELLUS

Opinnäytetyössä demonstroidaan ARKit-kirjaston käyttöä yksinkertaisella lisätyn todellisuuden sovelluksella, jossa käytetään hyväksi esiteltyjä ominaisuuksia. Ideana on toteuttaa sovellus, jolla voi mitata huonekaluja ja sijoittaa niitä vastaavia 3D-objekteja eri tilaan. Tämä toteutus auttaisi esimerkiksi muutettaessa uuteen asuntoon. Kaikki suoritettavat mittaukset tallennetaan puhelimen muistiin, joten sovelluksen voi sammuttaa liikuttaessa eri tilaan.

5.1 Demosovelluksen kuvaus

Sovellus mahdollistaa sen, että käyttäjällä on mahdollisuus mitata esineitä tai asettaa esineitä. Mitattaessa käyttäjä näkee viivan lisätyssä todellisuudessa ja mitatun etäisyyden. Kun käyttäjä on tyytyväinen mittaustuloksiin, hän voi tallentaa nykyiset mittatulokset tai peruuttaa ja mitata uudelleen. Mittaaminen tapahtuu asettamalla tähtäin kohtaan, josta halutaan aloittaa, ja painamalla näyttöä pohjassa niin pitkään, kuin halutaan mitata. Kun leveys, syvyys ja pituus ovat mitattuna, sovellus tuo esille ilmoitusikkunan mittojen tallentamiseksi.

Aseta objekti-tilassa käyttäjä voi valita haluamansa esineen. Tässä vaiheessa sovelluksessa on kolme esinettä: sohva, sänky ja pöytä. Objektin valittuaan käyttäjä valitsee, mitä mittatulosta haluaa käyttää. Tämän jälkeen objekti ilmestyy käyttäjän tähtäimeen, jossa käyttäjä voi valita asettaako sen suoraan, vai haluaako kääntää objektia y-akselin mukaisesti. Käyttäjän valitsee paikan ja objekti jää siihen pisteeseen. Tämän jälkeen käyttäjä päättää, mittaako hän uuden esineen, asettaako toisen objektin tai jatkaa saman objektin asettamista lisättyyn todellisuuteen. Objekti pysyy paikallaan, vaikka käyttäjä liikkuisi. Kuvassa 12 on sovelluksen aktiviteettikaavio, jossa kuvaillaan sovelluksen toimintaa.



KUVA 12. Sovelluksen aktiviteettikaavio

5.2 Sovelluksen lisätyn todellisuuden tilan vaihtaminen

Sovelluksen perusrakenne seuraa kuvan 12 aktiviteettikaavion mallin mukaista toteutusta. Kuvan 6 mukainen session aloitus mahdollistaa lisätyn todellisuuden kanssa vuorovaikuttamisen. Ensimmäiseksi luodaan painike, joka yhdistetään Main.storyboardista, josta voidaan vaihtaa tilaa mittaamisen ja asettamisen välillä. Kuvassa 13 näkyy, miten tiloja vaihdetaan ja miten se vaikuttaa käyttäjärajapintaan. Näkymässä on tärkeä piilottaa ja tuoda esiin elementtejä, joita tarvitaan sovelluksen toimivuuteen.

```
@IBAction func mode(_ sender: Any) {
    if !mode{
        mode = true
        modeButton.setTitle("Measure", for: .normal)
        chooseButton.isHidden = false
        okButtonO.isHidden = false
        rotateB.isHidden = false
        currentObject?.removeFromParentNode()
    }else{
        mode = false
        modeButton.setTitle("Place", for: .normal)
        chooseButton.isHidden = true
        okButtonO.isHidden = true
        rotateB.isHidden = true
    }
}
```

KUVA 13. Painikkeen konfiguraatio

Tilan vaihto on alustettu, ja mode-muuttujan toiminta määritetään. Muuttujan avulla voidaan valita lisätyn todellisuuden session aikana, miten vaikutetaan laitteessa esillä olevaan näkymään. Kahden tilan välillä on helppo tehdä yksinkertainen if-lause. Tämä tapahtuu renderöijän sisällä. Kuvassa 14 nähdään, mitä renderöinti toteutuksen sisällä tapahtuu tilan vaihdon tapahtuessa.

```
func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval) {
    DispatchQueue.main.async { [weak self] in
        if !self!.mode{
            self?.detectObjects()
        }else{

            self?.placeObjects()
        }
    }
}
```

KUVA 14. Tilan muutos renderöintitoteutuksen sisällä

Tässä on kaksi tilaa, joista toisella voidaan asettaa objekteja ja toisella mitata esineitä. Molemmat funktiot toimivat melkein samalla periaatteella. Havaitaan tilasta, mihin voidaan asettaa lisätyn todellisuuden avulla sisältöä. Tässä toteutuksessa käytetään hyväksi tähtäintä, joten voidaan määrittää oikean maailman koordinaatit samasta pisteestä laitteen ruudulla tällä koodilla.

> *guard let worldPosition = sceneView.realWorldVector(screenPosition: view.center) else { return }*

Muuttuja luodaan, jotta vältytään tilanteelta, jossa pistettä ei löydy, esimerkiksi tilanne, jossa ARKit ei ole vielä kalibroinut omaa paikkaansa. Muuttujaa worldPosition haetaan jatkuvasti ja sen arvo lukitaan sovelluksen logiikan niin halutessa. Mitattaessa esineitä detectObjects-funktio asettaa aloituspisteen tähän ja mittauksen päättyessä se asettaa lopetuspisteen tähän arvoon. Kuvassa 15 nähdään, miten mittaus näkyy käyttäjälle.

KUVA 15. Mitattu huonekalu ja mittauksen näkymä

Siniset pisteet kuvassa 15 ovat worldPosition-muuttujan muuttuvia arvoja, jotka on laitettu alku- ja loppupisteiksi, joiden avulla saadaan esineen mitat. Esineitä voidaan nyt mitata.

Seuraavaksi luodaan toinen tila, missä voidaan asettaa esineitä lisättyyn todellisuuteen, eli placeObjects-funktio. Tämä funktio toimii suurin piirtein samalla tavalla samalla tavalla kuin detectObjects-funktio. Haetaan worldPosition-muuttujan avulla piste, missä tähtäin on, mutta tässä tapauksessa ei aseta objektia heti. Objekti jää tähtäimeen kiinni, kunnes käyttäjä on tyytyväinen sen rotaatioon ja sen paikkaan. Kuvassa 16 nähdään asetettu sänky ja objektien asettamiseen tehty näkymä.

KUVA 16. Virtuaalinen objekti asetettu näkymään ja objektin asettamisen näkymä

5.3 Objektien luonti ja niiden asettaminen

Mittaamiseen ja esineiden asettamiseen luodaan omat luokat: MeasureNode ja ObjectNode. MeasureNode-luokassa käsitellään logiikkaa, miten ensimmäinen piste asetetaan, miten viiva kahden pisteen väliin muodostuu ja miten viimeinen piste asetetaan. Tämän lisäksi lasketaan etäisyys näiden pisteiden välillä ja näytetään se käyttäjälle. Tähän tarvitaan neljä SCNNode-muuttujaa, jotka ovat aloituspiste, viiva, teksti ja loppupiste. Aloitus- ja loppupiste luodaan helposti samalla noodilla kuvan 17 tyylisesti, mutta vain aloituspiste node asetetaan aluksi paikalleen.

```
let box = SCNBox.init(width: 0.01, height: 0.01, length: 0.01, chamferRadius: 0.0)
box.firstMaterial?.diffuse.contents = UIColor.blue
box.firstMaterial?.lightingModel = .constant
box.firstMaterial?.isDoubleSided = true
node = SCNNode(geometry: box)
node.position = startVector
sceneView.scene.rootNode.addChildNode(node)

endNode = SCNNode(geometry: box)
```

KUVA 17. Aloitus- ja loppupisteen luominen MeasureNode-luokassa

Seuraavaksi luodaan tekstinoodi, johon lisätään myös logiikka, että noodi on aina käyttäjään päin. Tekstiin lisätään myös muita rakenteita, kuten koko, sijainti luotavalla viivalla ja väri kuvan 18 lailla.

```
text = SCNText(string: "", extrusionDepth: 0.1)
text.font = .systemFont(ofSize: 5)
text.firstMaterial?.diffuse.contents = UIColor.white
text.alignmentMode  = CATextLayerAlignmentMode.center.rawValue
text.truncationMode = CATextLayerTruncationMode.middle.rawValue
text.firstMaterial?.isDoubleSided = true
let textWrapperNode = SCNNode(geometry: text)
textWrapperNode.eulerAngles = SCNVector3Make(0, .pi, 0)
textWrapperNode.scale = SCNVector3(1/500.0, 1/500.0, 1/500.0)
textNode = SCNNode()
textNode.addChildNode(textWrapperNode)
let constraint = SCNLookAtConstraint(target: sceneView.pointOfView)
constraint.isGimbalLockEnabled = true
textNode.constraints = [constraint]
sceneView.scene.rootNode.addChildNode(textNode)
```

KUVA 18. Tekstinoodin luonti ja asettaminen näkymään

Seuraavaksi luodaan luokan sisäisen funktion, jota voidaan kutsua pääkontrollerista, ja se myös päivittää viivanoodin loppupisteen ja viimeisen mitan lisätyn todellisuuden näkymään. Tässä funktiossa määritetään myös tekstinoodin paikan keskellä piirrettävää viivaa. Update funktio luodaan kuvan 19 mukaisesti.

```
func update(to vector: SCNVector3) {
    lineNode.removeFromParentNode()
    lineNode = startVector.line(to: vector, color: UIColor.white)
    sceneView.scene.rootNode.addChildNode(lineNode)

    text.string = distance(to: vector)
    textNode.position = SCNVector3((startVector.x+vector.x)/2.0, (startVector.y+vector.y)/2.0, (startVector.z+vector.z)/2.0)

    endNode.position = vector
    if endNode.parent == nil {
        sceneView.scene.rootNode.addChildNode(endNode)
    }
}
```

KUVA 19. Update funktion määritys

Kuvassa 19 myös nähdään kutsuttavan distance-funktiota, joka päivittää tekstinoodin numeroa. Tämä funktio on luotu yksinkertaisen vektorilaskennan avulla, jonka avulla voidaan laskea kahden kolmiulotteisen pisteen etäisyys lisätyn todellisuuden koordinaatiston avulla. Kuvassa 20 nähdään funktion julistus ja pisteiden etäisyyden laskeva algoritmi.

```
func distance(from vector: SCNVector3) -> Float {
    let distanceX = self.x - vector.x
    let distanceY = self.y - vector.y
    let distanceZ = self.z - vector.z
    return sqrtf((distanceX * distanceX) + (distanceY * distanceY) + (distanceZ * distanceZ))
}
```

KUVA 20. Distance-funktion julistus ja matemaattinen algoritmi

ObjectNode-luokassa objektien lisääminen ja luonti tapahtuu eri tavalla, koska objektia ei luoda itse vaan se saadaan 3D-objektista, jonka pääte on ”.dae”. ARKit tukee montaa erilaista 3D-objektitiedostoa, mutta COLLADA eli ”.dae”-tiedosto-pääte on suositeltu, koska se sisältää hyödyllisiä ominaisuuksia ilman muita tukevia tiedostoja. Tällaisen 3D-objektin luominen tapahtuu kuvan 21 mukaisesti.

```
let objectName = objName + ".dae"
guard let objectScene = SCNScene(named: objectName) else {return}
node = objectScene.rootNode
placementNode = objectScene.rootNode
```

KUVA 21. Uuden 3D-objektin luominen ARKitin käyttöön

Objektia luotaessa se kutsutaan pääkontrollerista parametrillä objName, joka sisältää valitun 3D-objektin.

currentObject = ObjectNode(scene: sceneView, vector: startValue, objName: modelName)

Kutsuttaessa tällä tavoin sovellus osaa valita oikean ”.dae”-mallin ja sijoittaa objektin oikeaan näkymään ja paikkaan. Näin luodaan objekti näkymään lisätyssä todellisuudessa. Objektin rotaatiota voidaan kutsua pääkontrollerista ja se muuttaa noodin ominaista Euler-kulmaa y-akselin suhteen seuraavalla tavalla:

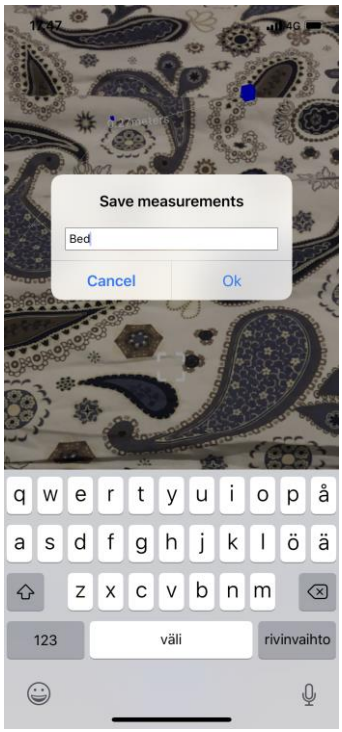
node.eulerAngles = withTrans

Sitä kutsutaan rotateNode-funktiolla, jonka parametriin sisältyy SCNVector3-muuttuja. SCNVector3-muuttuja on kolmiulotteinen vektori ja se rakentuu (x,y,z)-tapaisesti. Lisäksi, toisin kuin MeasureNode-luokassa, jossa päivitetään vain tekstiä ja loppupisteen paikkaa, ObjectNode-luokassa päivitetään yhden noodin paikkaa ja näkymässä painettaessa OK-painiketta sovellus asettaa uuden objektin worldPosition-muuttujan paikkaan.

5.4 Mittojen saaminen ja niiden tallennus

Aikaisemmin kohdassa 3 on käsitelty, miten mittaus tapahtuu lisätyn todellisuuden koordinaatistossa, mutta ei käsitelty, kuinka mitat tallennetaan ja miten ne haetaan tarpeen tullessa.

Kuten aikaisemmin mainittu, mittauksen jälkeen käyttäjälle tulee esiin ponnahdusikkuna, joka on kuvassa 22, johon voi täyttää mittaukselle haluamansa nimen.



KUVA 22. Ponnahdusikkuna

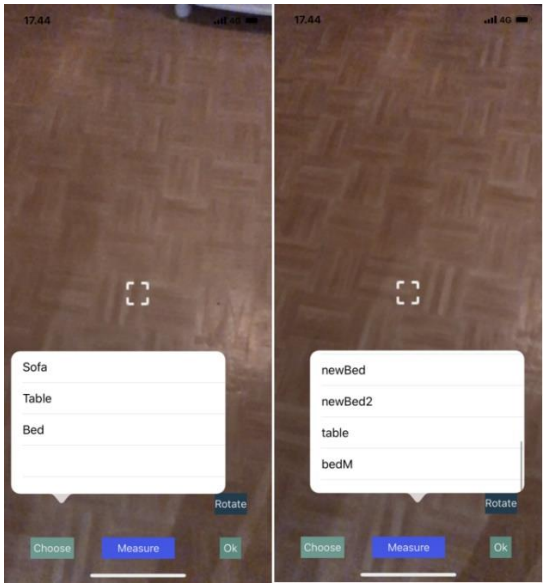
Ikkuna tulee esiin, kun sovellus havaitsee kolmannen mittauksen loppumisen. Mittauksien aikana arvot tallennetaan taulukkomuuttujaan nimeltä measurements. Kolmannen mittauksen päättyessä kutsutaan funktiota nimeltä showFinishAlert, jonka logiikalla ponnahdusikkuna tulee esille, ja painettaessa OK painiketta, mitat ja mittojen nimi tallennetaan. Tässä vaiheessa nimi tallennetaan eri taulukkoon, joka voidaan hakea sovelluksen tallennustilasta yhdellä avainarvolla. Kun haetaan mittauksia, haetaan ensin mittauksen nimi, joka on avainarvo, jolla saadaan mittaukseen kuuluvat arvot. Kuvassa 23 nähdään, miten tämä tapahtuu.

```swift
func writeKeysToMemory(arrayToMemory : [String]){
    let preferences = UserDefaults.standard
    preferences.set(arrayToMemory, forKey: "KEYS")
    print("writing \(arrayToMemory)")
}
func writeMeasurementToMemory(arrayToMemory : [Float], withKey : String){
    let preferences = UserDefaults.standard
    preferences.set(arrayToMemory, forKey: withKey)
}
func getKeys(){
    let preferences = UserDefaults.standard
    if preferences.array(forKey: "KEYS") == nil {
        print("nothing here")
    }
    else{
        keys = preferences.array(forKey: "KEYS") as! [String]
        print(keys)
    }
}

func getMeasurements(key : String) -> [Any]{
    let preferences = UserDefaults.standard
    if preferences.array(forKey: key) == nil {
        print("Error")
    }else{
        return preferences.array(forKey: key)!
    }
    return ["Empty"] //To satisfy function, not going to get here
}
```

KUVA 23. Muistin avainarvojen ja mittauksien kirjoittaminen ja haku

Näin voimme valita sovelluksen sammuttamisenkin jälkeen halutut mitat. Käynnistyessään sovellus hakee ensin mahdolliset avainarvot ja täyttää niillä toisen ponnahdusikkunan listan. Ensimmäinen lista täytetään 3D-objektien nimillä. Kuvassa 24 on ponnahdusikkunat kuvattuna. Listoista valitut asiat hakevat oikeat arvot ja luovat käyttäjän haluaman objektin näkymään lisättyyn todellisuuteen.



KUVA 24. Ponnahdusikkunavalikot objektin ja mittauksien valitsemiseksi

6 YHTEENVETO

Opinnäytetyön tavoite oli selvittää, miten ARKitin avulla voidaan luoda Applen mobiililaitteille nykyaikaisia lisätyn todellisuuden sovelluksia ja mitä lisätty todellisuus on. Tavoite saavutettiin, ARKitin ominaisuuksiin tutustuttiin, niitä testattiin käytännössä ja opinnäytetyön tekijällä on laajempi näkemys ARKitin tuomista mahdollisuuksista nykyaikaisessa sovelluskehityksessä.

Opinnäytetyön demosovellus oli omalla tavallaan haastava ja sen tekeminen syvensi aikaisempaa tietämystä. Siihen sai sisällytettyä monta ARKitin ominaisuutta ja demonstroitua, miten lisättyä todellisuutta voitaisiin käyttää hyödyksi arkisissa asioissa.

ARKit on erinomainen työkalu, josta voidaan luoda täysin siihen pohjautuva sovellus tai tuoda eloa jo olemassa oleviin sovelluksiin. Tästä esimerkkinä toimii nuorison suosima sovellus Snapchat, jossa lisättyä todellisuutta käytetään erilaisiin suodattimiin, joilla voidaan lisätä esimerkiksi aurinkolasit kasvoille.

Opinnäytetyö tehtiin selvitysmielessä ja demosovellus tehtiin vain esittelemään ARKitin ominaisuuksia. Demosovellus löytyy Githubista osoitteesta https://github.com/Ketkukelmi/ARDemoSovellus.

LÄHTEET

1. Infographic: The History of Augmented Reality. 2016. Augment. Saatavissa: https://www.augment.com/blog/infographic-lengthy-history-augmented-reality. Hakupäivä 8.1.2018.

2. Candy, Chris 2013. The History of Augmented Reality. SevenMedia. Saatavissa: http://sevenmediainc.com/the-history-of-augmented-reality/. Hakupäivä 8.1.2018.

3. About iOS 11 Updates. 2018. Apple. Saatavissa: https://support.apple.com/en-us/HT208067#11. Hakupäivä 8.1.2018

4. ARKit. 2018. Apple. Saatavissa: https://developer.apple.com/documentation/arkit. Hakupäivä 8.1.2018

5. Motte, Stefanie. Augmented Reality: A Comprehensive History (Part 1). 2018. Saatavissa: https://blog.vertebrae.com/history-augmented-reality-1. Hakupäivä 8.1.2018