LAPIN AMK
Lapland University of Applied Sciences

# MIGAEL AR -SOVELLUS

Nissi Petri

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2021

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

---

| | | | |
|---|---|---|---|
| **Tekijä** | Petri Nissi | Vuosi | 2021 |
| **Ohjaajat** | Maisa Mielikäinen, Toni Westerlund | | |
| **Toimeksiantaja** | Frostbit Software Lab | | |
| **Työn nimi** | MiGaEL AR -sovellus | | |
| **Sivumäärä** | 20 | | |

---


Opinnäytetyössä käsitellään Unity 3D -pelimoottorilla toteutettua ja lisätyn todellisuuden (Augmented Reality, AR) teknologiaa hyödyntävää sovellusta, jonka avulla voidaan harjoitella ajolähtötarkistusta kaivosteollisuudessa käytettävälle työkoneelle. Opinnäytetyön tavoitteena oli selvittää, kuinka virtuaaliselle todellisuudelle suunniteltu sovellus toimii lisättyä todellisuutta hyödyntävässä versiossa sekä tuottaa visuaalisesti autenttisen tuntuinen sovellus, jonka avulla käyttäjä pystyy turvallisesti perehtymään työkoneeseen ja suorittamaan sille ajolähtötarkistuksen. Sovellus kehitettiin Android-käyttöjärjestelmälle ja ohjelmointikielenä toimi C#.

Opinnäytetyössä tutustutaan lisätyn todellisuuden teknologiaan sekä käydään läpi sovelluksen tavoitteet ja perehdytään myös ohjelmasuunnittelun ja toteutusprosessin eri vaiheisiin.

Opinnäytetyön toimeksiantajana toimii Lapin ammattikorkeakoulun Frostbit Software Lab, ja sovellus on toteutettu laboratorion tiloissa. Opinnäyte toteutettiin MiGaEL-hankkeessa, jonka on rahoittanut Euroopan sosiaalirahasto (ESR). Rahoituksen hankkeelle on myöntänyt Pohjois-Pohjanmaan elinkeino-, liikenne- ja ympäristökeskus.

Opinnäytetyön lopputuloksena syntyi prototyyppi mobiilisovelluksesta, jonka avulla käyttäjä pystyy suorittamaan ajolähtötarkistuksen virtuaaliselle työkoneelle käyttäjän sijainnista riippumatta. Sovelluksen avulla erityisesti kaivosalan opiskelijat pystyvät perehtymään työkoneeseen ja hahmottamaan paremmin työkoneen suuren koon.



Avainsanat AR-mobiilisovellus, lisätty todellisuus, Unity

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Degree Programme in Information
and Communication Technology
Bachelor of Engineering

| | | | |
|---|---|---|---|
| **Author** | Petri Nissi | Year | 2021 |
| **Supervisor** | Maisa Mielikäinen, Toni Westerlund | | |
| **Commissioned by** | Frostbit Software Lab | | |
| **Subject of thesis** | MiGaEL AR-Application | | |
| **Number of pages** | 20 | | |

The main objective of this thesis was to develop an application which allows the user to perform a pre-departure check to a virtual haul truck by utilizing features of Augmented Reality. In addition, the application was made to test how well an application, which was originally designed for Virtual Reality, could work with utilizing features of Augmented Reality.

The application was designed for Android based devices which have a support for Augmented Reality. The difference between Augmented Reality and Virtual Reality was clarified. The technology utilizing Augmented Reality and how it has already been used in modern marketing and educational purposes was reviewed.

As a result, a prototype of the application was developed which allows the user to safely perform a pre-departure check to haul truck regardless the position of the user. The application can also be used for educational purposes, for example to help the user to realize the huge size of the haul truck.

Key words AR, application, augmented reality, unity

SISÄLLYS

# KÄYTETYT LYHENTEET JA TERMIT

| | |
|---|---|
| AR | Augmented Reality, lisätty todellisuus |
| Asset | sovelluksessa käytettäviä 3D-malleja, tekstuureita, scriptejä ja materiaaleja |
| UI | User Interface, käyttöliittymä |
| VR | Virtual Reality, virtuaalinen todellisuus |

# 1 JOHDANTO

Opinnäytetyö käsittelee lisätyn todellisuuden (Augmented Reality, AR) käyttämistä oppimistarkoitukseen käyttäen apuna mobiililaitetta, tässä tapauksessa puhelinta. Sovellus toteutettiin käyttämällä Unity3D-pelimoottoria.

Opinnäytetyön tavoitteena oli tuottaa prototyyppi sovelluksesta, minkä avulla voidaan suorittaa autenttisen tuntuinen ja turvallinen harjoitus työkoneen ajolähtötarkistuksesta. Vastaavanlaisia sovelluksia ei ole toistaiseksi vielä julkisesti saatavilla. Perinteinen ajolähtötarkistuksen suorittaminen voi olla monelle asiasta kiinnostuneelle haastavaa, sillä työkoneita löytyy vain yrityksistä. Myös kaivosalan opiskelijoille se voi olla haastavaa, koska oppilaitoksilla ei usein ole omia työkoneita, minkä vuoksi opiskelijoiden tulee siirtyä työmaalle suorittamaan harjoitukset.

Sovelluksen avulla käyttäjä pystyy harjoittelemaan ajolähtötarkistusta mobiililaitteella paikasta riippumatta, esimerkiksi luokkahuoneessa. Tämä helpottaa varsinkin oppilaitoksia perehdyttämään opiskelijat turvallisesti työkoneeseen ennen siirtymistä varsinaisen työkoneen luokse. Myös teoriapainotteisessa opiskelussa lisätyn todellisuuden käyttö helpottaa asioiden hahmottamista, sillä esimerkiksi työkoneen koko voi olla vaikea hahmottaa kuvien ja videoiden perusteella. Sovelluksen avulla käyttäjä pystyy sijoittamaan itsensä työkoneen viereen, mikä helpottaa mittasuhteiden hahmottamista ja ymmärtämistä.

Opinnäyte toteutettiin MiGaEL-hankkeessa, jonka on rahoittanut Euroopan sosiaalirahasto (ESR). Rahoituksen hankkeelle on myöntänyt Pohjois-Pohjanmaan elinkeino-, liikenne- ja ympäristökeskus. Hankkeen tavoitteena on kaivosalan koulutuksen pelillisen oppimisympäristön kehittäminen. Lisäksi hankkeen tavoitteena on nostaa kaivosalalla työskentelevien osaamista, sekä parantaa kaivosteollisuuden työturvallisuutta. Tavoitteena on myös lisätä kaivosalan houkuttelevuutta, minkä avulla voidaan parantaa työvoiman saatavuutta, edistää opiskelijoiden hakeutumista alalle sekä nostaa alan imagoa. Lisäksi pelillisen oppimisympäristön avulla voidaan edistää kaivosteollisuuteen hakeutuvien henkilöiden osaamista ja valmiuksia toimia alalla. (MiGaEL 2021.)

Sovelluksen kehitys toteutettiin käyttämällä Unity 3D -pelimoottoria. Myös Android-käyttöjärjestelmälle kehitettyä ARCore-laajennusta hyödynnettiin sovelluksen kehityksessä. Sovelluksen toiminnot ohjelmoitiin käyttämällä Visual Studio Code -ohjelmistokehitysympäristöä.

# 2 LISÄTTY TODELLISUUS

Lisätty todellisuus (Augmented Reality, AR) tarkoittaa sitä, että todellisen maailman päälle lisätään virtuaalista sisältöä, jota pystytään tarkastelemaan erilaisilla laitteilla, esimerkiksi älypuhelimella. Pelkästään älypuhelimella AR-teknologiaa hyödyntäviä sovelluksia käytti vuonna 2020 lähes 600 miljoonaa aktiivista käyttäjää. Yksi tutuimpia sovelluksia, joka hyödyntää lisätyn todellisuuden teknologiaa on sosiaalisen median palvelu Snapchat. (AR Insider 2020.)

Tulevaisuudessa lisättyä todellisuutta hyödyntävien älylasien odotetaan olevan seuraava suuri läpimurto päälle puettavassa teknologiassa, jonka käyttö sulautuu jokapäiväiseen elämään. Useat suuret teknologiayritykset ovat jo aloittaneet omien lisättyä todellisuutta hyödyntävien lasien kehityksen. Esimerkiksi Facebook on vahvistanut kehittävänsä lisätyn todellisuuden laseja ja julkaisevansa ne markkinoille vuoden 2021 aikana. (Wareable 2021.)

Lisättyä todellisuutta on käytetty pitkään videopelialalla, mutta teknologian kehityttyä sitä on alettu hyödyntämään laajemmin, kuten markkinoinnissa. Esimerkiksi huonekaluvalmistaja Ikea on kehittänyt oman sovelluksen, jonka avulla käyttäjä pystyy virtuaalisesti sijoittamaan huonekaluja huoneisiin ja tarkastelemaan niitä lisättyä todellisuutta tukevalla laitteella. (Kotimikro 2018.)

Myös selaimella käytettävä WebAR on yksi laajasti yleistyvä lisätyn todellisuuden teknologiaa hyödyntävä ominaisuus, jonka avulla esimerkiksi yritykset pystyvät esittelemään tuotteittaan ja palveluitaan lisätyn todellisuuden avulla ilman, että käyttäjän tarvitsee ladata sovelluksia laitteelleen nähdäkseen yrityksen tarjoaman sisällön. WebAR:n tuottaman sisällön tarkasteluun käyttäjä tarvitsee vain AR-teknologiaa tukevan laitteen siirtyäkseen sisällöntarjoajan nettisivulle. Koska WebAR-toteutuksessa ohjelmakoodi ja sisältö on tallennettu verkkosivuston palvelimelle, vähentää se käyttäjän päätelaitteen rasitusta ja mahdollistaa kevyemmät laitteistovaatimukset. Myös sisällöntarjoajan kannalta palvelimen kautta toimiva sisältö on tehokkaampaa. Jos sisältöön halutaan muutoksia, voidaan ne ladata suoraan palvelimelle. Tällöin päivitetty sisältö on heti kaikkien käyttäjien käytettävissä, sen sijaan että jokainen käyttäjä joutuisi erikseen lataamaan päivityspaketin. (Hurja 2020.)

Viime vuosina lisättyä todellisuutta on alettu hyödyntämään laajemmin myös opetuskäytössä. Opetuskäytössä lisätyn todellisuuden vahvuuksia on pelillinen lähestymistapa opiskeltavaan aiheeseen, mikä tuo perinteisen opiskelun tueksi interaktiivista sekä visuaalista opetusmateriaalia, mikä helpottaa opiskeltavan aiheen hahmottamista ja parantaa opiskelun kiinnostavuutta. Myös ammatillisessa opiskelussa lisättyä todellisuutta voidaan käyttää esimerkiksi luomalla harjoitussimulaatio, missä käyttäjä pystyy suorittamaan tehtäviä turvallisesti ilman loukkaantumisvaaraa. (Sinha 2021.)

# 3 SOVELLUKSEN TOTEUTTAMINEN

## 3.1 Määrittely ja suunnittelu

Sovelluksen suunnittelun alkuvaiheessa määriteltiin toimeksiantajan kanssa sovelluksen tavoitteet sekä aikataulun puitteissa toteutettavat toiminnallisuudet. Koko opinnäytetyöhön käytettävä aika oli kolme kuukautta, johon sisältyivät sovelluksen prototyypin tuottaminen ja opinnäytetyön kirjoittamisvaihe. Sovelluksen kehittämisessä käytettävät työkalut olivat Unity 3D -pelimoottori ja Visual Studio Code -ohjelmakehitysympäristö.

Sovelluksen minimitavoitteiksi asetettiin, että sitä pystytään käyttämään Android Nougat 7.0 -käyttöjärjestelmää tukevalla mobiililaitteella. Lisäksi työkoneen tulee näkyä sovelluksessa realistisen kokoisena ja lukittuna paikalleen sovelluksessa siten, että sitä pystytään kiertämään. Myös turvallisuuspoikkeamien sattumanvaraisuuden ohjelmointi oli yksi minimitavoitteista. Turvallisuuspoikkeaminen sattumanvaraisuus tarkoittaa sitä, että työkoneen tarkastettavien kohteiden kunto vaihtelee jokaisella käyttökerralla, mikä lisää tarkistuksen haastavuutta.

Sovelluksen kohderyhmänä ovat kaivosteollisuuden opiskelijat ja muut siihen perehtyneet henkilöt, mutta sitä pystyvät käyttämään myös kaivosteollisuuteen perehtymättömät käyttäjät. Täysin kaivosteollisuuden työkoneista tietämätön käyttäjä voi aluksi kokea haastavaksi löytää kaikki tarkastuslistalla olevat kohteet ja tulkita oikein kohteiden kunnon.

Koska sovelluksen aihe on osa MiGaEL-hanketta ja hankkeeseen on tehty jo perinteinen mobiilisovellus, valmista mallia työkoneesta pystyttiin hyödyntämään myös sovelluksen AR-versiossa. Myös muita aineistoja, kuten tekstuureita pystyttiin hyödyntämään sovelluksen AR-versiossa.

Sovelluksesta on tehty myös virtuaalista todellisuutta (Virtual Reality, VR) hyödyntävä versio. Siinä missä lisätyn todellisuuden avulla oikeaan maailmaan lisätään virtuaalista sisältöä, jota voidaan tarkastella käyttämällä esimerkiksi älypuhelinta, virtuaalinen todellisuus hyödyntää virtuaalilaseja, jotka luovat käyttäjän näkökentälle kokonaan virtuaalisen maailman.

## 3.2 Toteutusprosessin vaiheet

Projekti toteutettiin käyttämällä SCRUM-menetelmää, joka on yksi ketterän projektihallinnan viitekehyksistä. SCRUM:n pääpiirteisiin kuuluvat määrämittaiset kehitysjaksot, joita kutsutaan sprinteiksi. Sprintti kestää yleensä 1–4 viikkoa, jonka aikana projektiryhmä toteuttaa ennalta määrätyt tehtävät. SCRUM-menetelmä on erityisesti suunniteltu pienten (3–9 hlöä) ohjelmistokehitystiimien käyttöön, mutta sitä voidaan käyttää myös yksin työskennellessä. Sprinttien sisällä käydään päivittäin noin 15 minuutin palavereja, joiden aikana projektiryhmä käy läpi tapahtuneen kehityksen ja tarvittaessa työn uudelleen organisointi, jos se nähdään tarpeelliseksi. (Projektihallinta 2021.)

Taulukossa 1 esitetään opinnäytetyöprojektin backlog eli kehitysjono. Sprint 1 aloitettiin projektisuunnitelman tekemisellä sekä opinnäytetyöhön liittyvien dokumenttien jakamisella ohjaajien kanssa. Kehitysjonon työvaiheiden priorisointi määriteltiin sovelluksen kehitysvaiheissa tarvittavien ominaisuuksien mukaan, esimerkiksi työkoneen lukitsemista paikalleen ei voi tehdä ilman käyttöliittymää. Projektisuunnitelman tekeminen ennen varsinaista sovelluksen kehitystyön aloittamista on tärkeää, sillä siinä määritellään sovellukseen halutut ominaisuudet ja laatutavoitteet sekä koko projektin aikataulu. Sovelluksen kehitysvaiheessa sprinttien määräksi määriteltiin viisi ja sprintin pituudeksi viikko.

Taulukko 1. Projektissa käytetty tuotteen kehitysjono

| Kuvaus | Työmäärä arvio (h) | Sprint | Prioriteetti |
|---|---|---|---|
| Projektisuunnittelu | 20 | 1 | 1 |
| Sovelluksen käyttöliittymän suunnittelu | 20 | 2 | 2 |
| Sovelluksen käyttöliittymän ohjelmointi | 40 | 2-3 | 2 |
| Tutkittavan esineen lisääminen sovellukseen | 16 | 3 | 3 |
| Tutkittavan esineen lukitseminen sovellukseen | 16 | 3 | 3 |
| Sattumanvaraisten turvallisuuspoikkeamien ohjelmointi | 30 | 4 | 4 |
| Tarkistuslistan ohjelmointi | 80 | 5 | 4 |
| Ohjelmakoodin virhetestaus | 40 | 2-5 | 5 |

### 3.3 Sovelluksen käyttöliittymän suunnittelu ja ohjelmointi

Sovelluksen käyttöliittymää suunniteltiin siten, että käyttöliittymä pidetään mahdollisimman yksinkertaisena ja selkeänä, jotta itse työkoneen tarkastelu olisi mahdollisimman vaivatonta ja sujuvaa. Sovelluksen käyttökieleksi valittiin englanti, jotta myös kansainväliset käyttäjät voivat käyttää sovellusta vaivattomasti. Sovelluksen päänäkymässä on yhteensä kolme sovelluksen toimintaa ohjaavaa näppäintä, joilla ohjataan työkoneen lukitsemista maailmaan, tarkistuslistan avaamista sekä tarkistuksen uudelleen aloittamista. Kuviossa 2 näkyy, kuinka käyttöliittymän näppäimet on sijoitettu näytön nurkkiin, jotta käyttäjä pystyy vaivattomasti tarkastelemaan työkonetta ilman, että käyttöliittymä tulisi tielle.

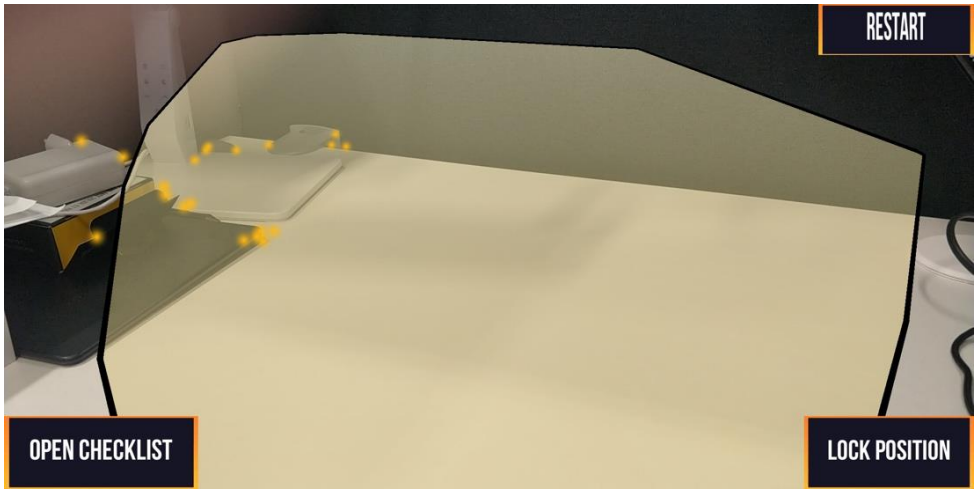

Kuvio 2. Sovelluksen ulkoasu työkonetta tarkastaessa

### 3.4 Tutkittavan esineen lisääminen sovellukseen

Projektin alkuperäinen suunnitelma oli tuottaa sovellus, jossa työkone näkyy pelkästään alkuperäisessä mittakaavassa, mutta sovelluksen kehittämisen kannalta sen testaaminen olisi ollut työlästä, sillä alkuperäinen työkone ei olisi mahtunut työskentelytilaan, joten siitä kehitettiin myös sisätiloissa käytettävä versio.

Tutkittavan esineen lisäämiseksi, sovelluksen täytyy ensin tunnistaa ympäristö. Pintojen tunnistamiseen sovellus käyttää laitteen kameraa. Kuviossa 3 näkyvä vaalennettu alue on sovelluksen tunnistamaa pintaa, johon työkone pystytään lisäämään näpäyttämällä sormella aluetta. Samassa kuviossa näkyvät keltaiset

pallot kertovat käyttäjälle, että sovellus on tunnistanut pinnan, mutta ei ole vielä piirtänyt aluetta loppuun.




Kuvio 3. Sovelluksen tunnistama ja piirtämä vaalennettu alue

Ohjelmakoodi käyttää Unityn AR Raycast Manager -ominaisuutta maaston tunnistamista varten. AR Raycast Managerin avulla sovellus lähettää virtuaalisia säteitä hyödyntäen älypuhelimen kameraa. Säteen kohdatessa tunnistettavan pinnan se tallentaa datan muistiin, minkä avulla Unityn AR Plane Manager piirtää tunnistetun maaston näkyviin sovellukseen kuvion 3 esimerkin tapaan.

## 3.5 Tutkittavan esineen tarkastelu sovelluksessa

Tutkittavan esineen monipuolisempaa tarkastelemista varten työkone lukitaan sovelluksessa paikalleen, mikä mahdollistaa sen, että käyttäjä voi halutessaan itse kiertää työkonetta, ilman että työkone vaihtaisi paikkaa puhelimen liikkuessa. Kuviossa 4 tarkasteltava työkone on lisätty maailmaan.


Kuvio 4. Työkone lisätty maailmaan

Kuviossa 5 käydään läpi vaiheet, jotka ohjelma käy läpi, kun tarkasteltavaa kohdetta yritetään lisätä virtuaalisesti maailmaan. Onnistuneen lisäyksen ehdot ovat, että käyttäjä painaa sormella sovelluksen piirtämän alueen sisältä, eikä samalla painalluksella koske käyttöliittymää ohjaavaa nappia. Myös tarkasteltavan kohteen lukitus täytyy olla vapautettu.

```
private void Update() {
    //Jos osoitin osuu tunnistetun alueen sisälle, eikä osunut käyttöliittymän päälle, sekä kohteen lukitus on pois päältä,
    //tarkasteltava kohde siirtyy osoitettuun paikkaan
    if (Input.GetMouseButtonDown(0) && !IsClickOverUI() && !isLocked)
    {
        List<ARRaycastHit> hitPoints = new List<ARRaycastHit>();
        raycastManager.Raycast(Input.mousePosition, hitPoints, TrackableType.Planes);

        if (hitPoints.Count > 0) {
            Pose pose = hitPoints[0].pose;
            transform.rotation = pose.rotation;
            transform.position = pose.position;
        }
    }
    if(isLocked == true)
    {
        //Jos kohteen lukitus on päällä, sovellus ottaa käytöstä pois pintojen tunnistamisen ja poistaa näkyvistä jo tunnistetun maaston
        toggleButton.SetActive(false);
        toggleButton.GetComponent<ToggleAR>().OnValueChanged(isOn: false);
    }
    else
    {
        //Jos kohteen lukitus ei ole päällä, sovellus jatkaa pintojen tunnistamista.
        toggleButton.SetActive(true);
        toggleButton.GetComponent<ToggleAR>().OnValueChanged(isOn: true);
    }
}

bool IsClickOverUI() {
    //Ohjelma tarkistaa että osoitin ei ollut käyttöliittymän päällä
    PointerEventData data = new PointerEventData(EventSystem.current) {
        position = Input.mousePosition
    };
    List<RaycastResult> results = new List<RaycastResult>();
    raycaster.Raycast(data, results);
    return results.Count > 0;
}
```

Kuvio 5. Sovelluksessa käytetty ohjelmakoodi tarkasteltavan kohteen lisäämisestä maailmaan

Kun käyttäjä lukitsee työkoneen maailmaan, työkonetta pystytään kiertämään ja sovellus lopettaa pintojen tunnistamisen (Kuvio 6). Koska sovellus haluttiin myös

sisätiloissa käytettäväksi, sovelluksen sisätiloihin suunnitellussa versiossa käyttäjä pystyy itse muokkaamaan työkoneen kokoa vetämällä kahta sormea näytöllä joko sisäänpäin tai ulospäin toisistaan. Työkonetta pystytään myös kääntämään ruudulla liikuttamalla yhtä sormea näyttöä pitkin.


Kuvio 6. Työkone lukittuna maailmaan

Kuviossa 7 käydään läpi ohjelmakoodia, minkä avulla työkoneen kokoa voidaan säädellä. Kuviossa käytettävä ohjelmakoodi lukee jatkuvasti mahdollisia kosketuksia näytöllä, ja silloin kun sovellus tunnistaa kaksi kosketusta, se tallentaa molempien kosketuksien sijainnin. Kun kosketuksien sijainti näytöllä vaihtuu, muuttuu myös työkoneen koko.

```
void Update()
{
    int fingersOnScreen = 0;

    //Lukee näyttöä koskettavien asioiden määrää
    foreach (Touch touch in Input.touches)
    {
        fingersOnScreen++;

        //Jos sovellusta tunnistaa kaksi kosketusta näytöllä, alla oleva ohjelmakoodi suoritetaan
        if (fingersOnScreen == 2)
        {
            if (touch.phase == TouchPhase.Began)
            {
                //Ohjelma tallentaa molempien kosketuksien aloitussijainnin näytöllä
                initialFingersDistance = Vector2.Distance(Input.touches[0].position, Input.touches[1].position);
                initialScale = objectRotate.transform.localScale;
            }
            else
            {
                //Jos kosketuksien sijaintien välinen erotus muuttuu, muuttuu myös tarkasteltavan kohteen koko
                float currentFingersDistance = Vector2.Distance(Input.touches[0].position, Input.touches[1].position);
                float scaleFactor = currentFingersDistance / initialFingersDistance;
                objectRotate.transform.localScale = initialScale * scaleFactor;
            }
        }
    }
}
```

Kuvio 7. Tarkasteltavan kohteen kokoa ohjaava ohjelmakoodi

Kuviossa 8 käydään läpi työkoneen kääntämiseen käytettävää ohjelmakoodia. Myös tämä ohjelmakoodi lukee jatkuvasti näytön kosketuksien määrää. Kun ohjelmakoodi tunnistaa yhden kosketuksen näytöllä, se tallentaa sijainnin, ja jos kosketuksen sijainti muuttuu, kääntyy työkone näytöllä.

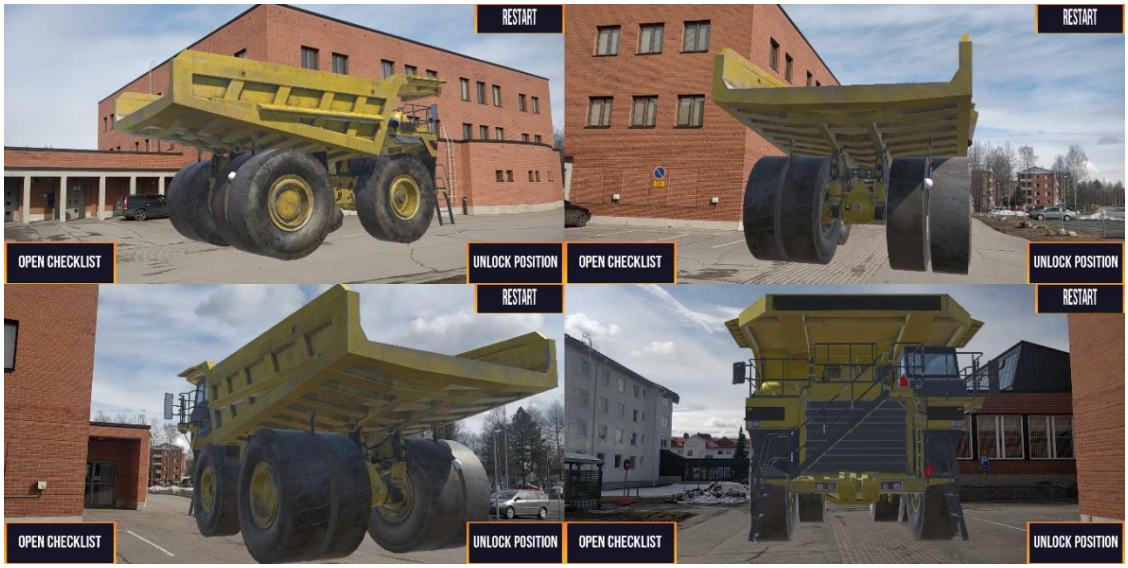
```
private void Update()
{
    int fingersOnScreen = 0;

    //Lukee näyttöä koskettavien asioiden määrää
    foreach (Touch touch in Input.touches)
    {
        fingersOnScreen++;

        //Jos sovellus tunnistaa kosketuksen näytöllä, ja kosketuksien määrä on alle 2, alla oleva ohjelmakoodi suoritetaan
        if (Input.touchCount == 1 && Input.touchCount < 2)
        {
            if (touch.phase == TouchPhase.Began)
            {
                //Tallentaa kosketuksen aloituspisteen ruudulla
                touchStartPos = touch.position.x;
            }
            else
            {
                //Jos kosketuksen sijainti näytöllä muuttuu x-vectorilla, myös tarkasteltava kohde kääntyy näytöllä
                if (touchStartPos > touch.position.x)
                {
                    objectRotate.transform.Rotate(Vector3.up, -rotateSpeed * Time.deltaTime);
                }
                else if (touchStartPos < touch.position.x)
                {
                    objectRotate.transform.Rotate(Vector3.up, rotateSpeed * Time.deltaTime);
                }
            }
        }
    }
}
```

Kuvio 8. Tarkasteltavan kohteen kääntämistä ohjaava ohjelmakoodi

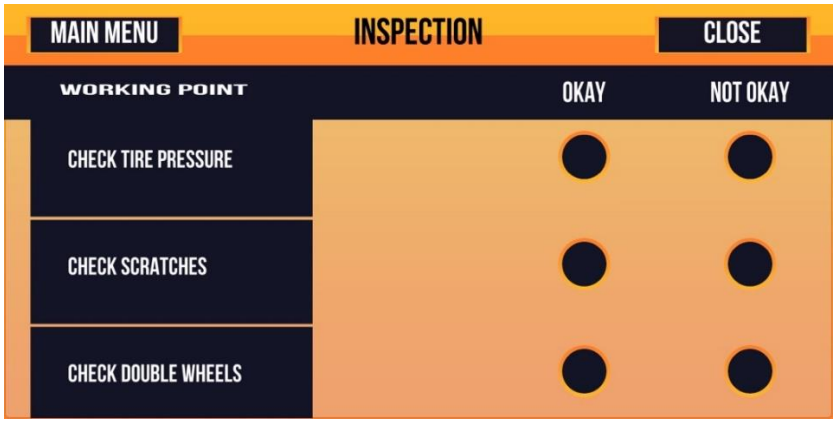
Sovelluksen ulkotiloihin kehitetyssä versiossa työkone pysyy alkuperäisessä mittakaavassa (Kuvio 9). Sovelluksen ulkotilan versiossa tarkoitus on saada käyttäjä itse kävellen kiertelemään ja tutkimaan konetta ja täten saada realistisen tuntuinen kokemus itse tarkastusprosessista.




Kuvio 9. Työkone alkuperäisessä mittakaavassa

## 3.6 Tarkistuslistan, ja sattumanvaraisten vikojen lisääminen

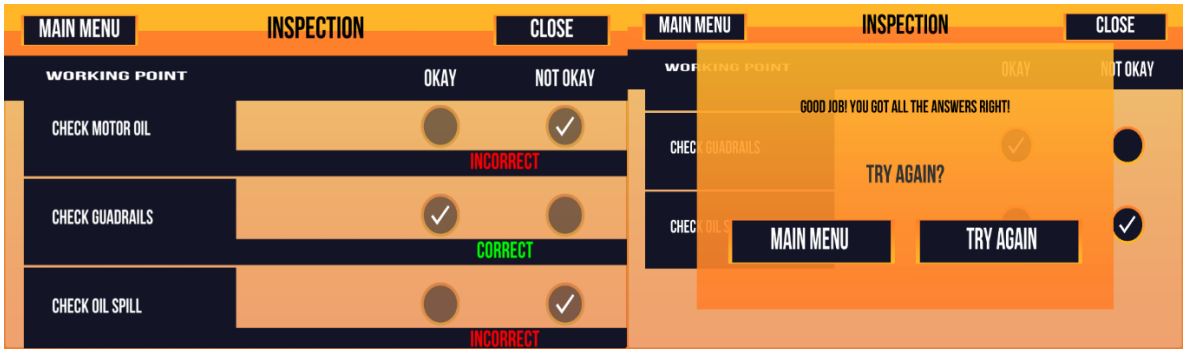
Koska sovelluksen tavoitteena on opettaa ja haastaa käyttäjän taidot, täytyy myös tarkastettavien kohteiden kunnon vaihdella. Tämän vuoksi sovellus ohjelmoitiin siten, että työkoneen kunto vaihtelee jokaisella käyttökerralla. Tämän sovelluksen versiossa tarkastettavia vikoja on yhteensä 12 kappaletta, jotka vaihtuvat jokaisella tarkastuskerralla. Lista työkoneen tarkastettavista kohteista sisältää jokaista tarkastuskohdetta koskevan valintataulukon (Kuvio 10) missä käyttäjällä on valittavana ”Okay” ja ”Not Okay” valintaruudut jokaiselle tarkistettavalle kohteelle.




Kuvio 10. Tarkistuslistan näkymä käyttäjälle

## 3.7 Tarkistuslistan tuloksen esittäminen

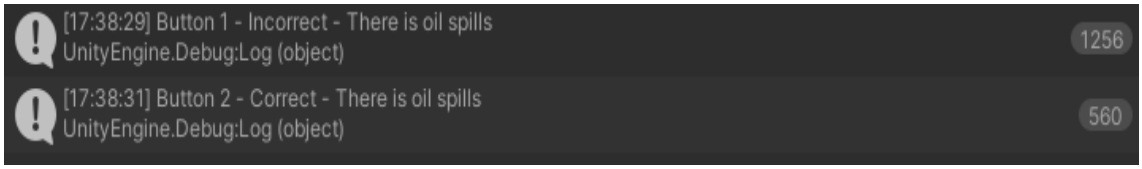
Kun käyttäjä haluaa saada oman suorituksen tuloksen näkyviin, tarkistus päätetään painamalla Check Results -näppäintä. Tarkistuksen päättämisen jälkeen sovellus näyttää tuloksesta riippuen käyttäjälle joko väärin vastatut vastaukset punaisella tekstillä korostettuna tai onnitteluruudun (Kuvio 11).




Kuvio 11. Tarkistuksen tulosruudut

## 3.8 Ohjelmakoodin logittaminen

Sovellusta kehitettäessä sen toiminnallisuuksia testattiin tiheästi, jotta mahdolliset ongelmat voitiin havaita ja korjata mahdollisimman varhaisessa vaiheessa sekä pystyttiin välttämään samoja virheitä jatkossa. Testauksessa hyödynnettiin toimeksiantajan tarjoamaa älypuhelinta, ja Unity3D:n ”Debug.Log”-ominaisuutta, joka tulostaa tekstiä Unity3D:n konsoliin, mistä pystyttiin tarkkailemaan tarkastettavien toimintojen toimivuutta.




Kuvio 12. Debug.Login avulla konsoliin tulostettua tekstiä

Kuviossa 12 näkyy Debug.Login avulla konsoliin tuotettuja ilmoituksia, jolla testattiin tarkistuslistan nappien toimivuutta öljyvalumien kohdalla. Virheilmoituksessa näkyvä Button 1 tarkoittaa tarkistuslistalla näkyvää Okay-nappia ja Button 2 Not Okay -nappia. Testaustarkoituksessa molempia nappeja painettiin, jotta nähtiin kuinka sovellus reagoi annettuihin vastauksiin. Tällä testauskerralla ohjelma oli asettanut yhdeksi turvallisuuspoikkeamaksi työkoneeseen aiheutuneen öljyvuodon (Kuvio 13), joten nappien voitiin todeta toimivan halutulla tavalla.

Kuvio 13. Öljyvalumia työkoneessa

# 4 POHDINTA

Opinnäytetyön tavoitteena oli tutkia, kuinka hyvin virtuaaliselle todellisuudelle suunniteltu sovellus toimii hyödyntämällä lisätyn todellisuuden ominaisuuksia. Opinnäytetyön valmistuttua voidaan todeta, että lisätyn todellisuuden avulla pystytään tuottamaan visuaalisesti autenttisen tuntuinen kokemus työkoneen ulkopuolisesta tarkastelusta.

Sovelluksen jatkokehitystä ajatellen sovellukseen voitaisiin kehitellä tarkistustuloksia seuraava tulostaulukko, jonka avulla käyttäjä pystyy seuraamaan omaa edistymistään ja tarkastelemaan mitkä asiat käyttäjille ovat tuottaneet eniten ongelmia. Tulostaulukon avulla esimerkiksi opettajalle mahdollistuisi seurata, mitkä asiat ovat tuottaneet vaikeuksia ja vaativat lisää perehdytystä. Myös iOS-käyttöjärjestelmää tukevan version rakentaminen olisi laajemman käyttäjäkunnan tavoittamiseksi tärkeä uudistus. Lisäksi sovelluksen käyttöliittymään voisi ohjelmoida useamman kielen.

Sovelluksen valmistuttua voidaan todeta, että Unity3D:n kehitysympäristön käyttö opinnäytetyön sovelluksen kehittämiseen oli hyvä valinta. Unity3D:n oma tuki Android-käyttöjärjestelmän kehitykseen vähensi kehitykseen käytettävää työmäärää huomattavasti. Android-käyttöjärjestelmälle kehitetty ARCore tarjosi AR-kehitykseen pohjan, jonka ympärille oli helppo lähteä rakentamaan omaa sovellusta.

LÄHTEET

AR Insider 2020. Mobile AR users approach 600 million. Viitattu 7.5.2021 https://arinsider.co/2020/09/10/mobile-ar-users-approach-600-million/.

Hurja 2020. WebAR – Verkkosisällön uusi ulottuvuus. Viitattu 12.5.2021 https://www.hurja.fi/blogi/webar-verkkosisallon-uusi-ulottuvuus/.

Kotimikro 2018. Mitä on lisätty todellisuus? Viitattu 3.5.2021 https://kotimikro.fi/yhteiskunta/uusi-tekniikka/mita-on-lisatty-todellisuus.

MiGaEL 2021. Hanke. Viitattu 26.4.2021 https://www.migael.fi/hanke/.

Projektihallinta 2021. Scrum. Viitattu 29.4.2021 https://projektinhallinta.info/scrum-ketterat-menetelmat-projektinhallinnassa/.

Sinha, S. 2021. Augmented reality in education: A staggering insight into the future. Viitattu 3.5.2021 https://elearningindustry.com/augmented-reality-in-education-staggering-insight-into-future.

Wareable 2021. The best smartglasses and headsets 2021. Viitattu 7.5.2021 https://www.wareable.com/ar/the-best-smartglasses-google-glass-and-the-rest.