Pistepilvien visualisointi

Valmiit ratkaisut sekä Unity-prototyypin kehittäminen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
kevät, 2021

Heikki Luukkonen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Hämeenlinnan korkeakoulukeskus

Tekijä Heikki Luukkonen Vuosi 2021

Työn nimi Pistepilvien visualisointi – Valmiit ratkaisut sekä Unity-prototyypin kehittäminen

Ohjaajat Lasse Seppänen

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli selvittää tapoja pistepilvien nopeaan visualisointiin. Tavoitteena oli selvittää, minkälaisia valmiita ratkaisuja pistepilvien visualisointiin on olemassa ja olisiko tällainen työkalu mahdollista tuottaa itse esimerkiksi Unity-pelimoottoria hyödyntäen. Opinnäytetyön toimeksiantajana toimi Keskusrikospoliisin rikostekninen laboratorio. Visualisointityökalulta haluttuja ominaisuuksia olivat esimerkiksi mittauksien tekeminen ja pistepilven katselu virtuaalilasien avulla.

Työn tietoperustana ovat pistepilvistä tehdyt tuoreet opinnäytetyöt tietojenkäsittelyn ja rakennustekniikan alalta sekä aihealueen tutkimukset. Opinnäytetyön toiminnallisessa osuudessa kerrotaan Unityllä tuotetun pistepilvien visualisointityökalun prototyypin kehittämisestä ja kartoitetaan Faro Scene LT -ohjelmistoa mahdollisena ratkaisuna toimeksiantajan toiveisiin. Prototyyppi pohjautuu Unity-kaupasta saataviin pistepilvien visualisointiin tarkoitettuihin kirjastoihin.

Opinnäytetyössä havaittiin, että Faro Scene LT oli toimeksiantajan vaatimuksia vastaava työkalu sen sisältämien kattavien ominaisuuksien vuoksi. Työssä selvisi myös, että Unityllä on mahdollista luoda pistepilvien visualisointityökalu. Itseluodun työkalun etuina ovat ominaisuuksien räätälöinti juuri omiin tarkoituksiin, sekä ei ole pelkoa siitä että tuki tai päivitykset loppuisivat.

Avainsanat laserkeilaus, pistepilvi, Unity

Sivut 27 sivua ja liitteitä 1 sivu

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna University Centre

Abstract

Author Heikki Luukkonen
Year 2021
Subject Point cloud visualization – Readymade solutions and development of Unity-prototype
Supervisors Lasse Seppänen

ABSTRACT

The aim of this thesis was to find out ways for fast visualizing of point clouds. The purpose was to figure out what kind of readymade solutions were available for point cloud visualization and if it would be possible to develop such software inhouse using a game engine like Unity. The commissioner of this thesis was the Forensic Laboratory of the National Bureau of Investigations. The wanted features of the point cloud visualization software were for example a tool for measurements and the ability to view point clouds in virtual reality.

The theory of this thesis was based on recently published theses from the industry of information technology and construction engineering and also research papers and publications from related fields. The practical part of this thesis demonstrates the development of a Unity based prototype of a point cloud viewing software and also analyse Faro Scene LT as a feasible option for the commissioners use case. The Unity-prototype is based on a plugin available from the Unity Asset Store.

As a result of this thesis, it was found out that the Scene LT was what the commissioner was looking for in a point cloud viewing software. It was also found out that it is possible to develop a credible option for point cloud viewing software with Unity. The advantages of an inhouse software are that the features and UI can be tailored and there is no fear that support, or updates would stop.

Keywords laser scanning, point cloud, Unity

Pages 27 pages and appendices 1 page

Sanasto

ASCII	American Standard Code for Information Interchange. ASCII on standardi kirjainten näyttämisessä elektronisessa muodossa.
ASTM	Standardisoimisjärjestö
ASPRS	American Society for Photogrammetry and Remote Sensing.
Collider	Näkymätön komponentti, joka määrittää objektin reunat törmäyksen tunnistamiseksi.
Mesh	Polygonimalli tai polygoniverkko
Prefab	Unity-pelimoottorissa valmis elementti.
Scene	Unity-pelimoottorin valmis taso.
Json	JavaScript Object Notation. Kevyt tiedostomuoto tiedon tallentamiseen ja siirtämiseen.
SDK	Software Development Kit, kokoelma ohjelmistokehityksen työkaluja.

Sisällys

Kuvat, ohjelmakoodit ja taulukot

Liitteet

1 Johdanto

Laserkeilaus on etämittausmenetelmä, jolla voidaan tuottaa minuuteissa rakennetusta ympäristöstä miljoonia mittauksia ja kuvainformaatiota. Laserkeilauksella on jo mittaustapana monia etuja, mutta suurimmat hyödyt syntyvät laserkeilauksella tuotetun pistepilven hyödyntämisestä. Pistepilvi on numeerista tietoa, joka kertoo laserkeilauksella mitattujen pintojen koordinaatit ja heijastuksen voimakkuuden. Tietokonegrafiikalla esitettynä pistepilven tiheä ja värillinen pistejoukko luo visuaalisen kolmiulotteisen toisinnon reaalimaailman kohteesta. (ProDigiOUs, 2021)

Pistepilvien avulla on mahdollista tehdä tarkkoja etäisyysmittauksia, selvittää tilojen ja rakenteiden mittoja sekä selvittää talotekniikan reittejä. Laserkeilaus ei kuitenkaan rajoitu ainoastaan uudisrakennuksen ja entisöinnin pariin, vaan sitä voidaan käyttää myös rikos- ja onnettomuustutkinnassa. Laserkeilaus on nopea ja tehokas keino dokumentoida tapahtumapaikka luotettavasti sellaisena kuin se oli hetkenä, jolloin se keilattiin. Pistepilveä voidaan jälkeenpäin analysoida ja tutkia eri tavoin, esimerkiksi veriroiskeiden alkupisteen ja niiden suunnan selvittämiseksi. (Nordic Geo Center Oy, 2011; ProDigiOUs, 2021)

Opinnäytetyön aihe syntyi suoritettuani tutkintoon kuuluvan harjoittelun Keskusrikospoliisin rikosteknisessä laboratoriossa. Harjoittelussa tavoitteenani oli tuottaa tai löytää ratkaisu, jolla voi visualisoida sekä käsitellä pistepilviä. Ratkaisulla tarkoitetaan esimerkiksi Unity-pelimoottorilla tuotettua tai valmista ohjelmistoa. Käsittelyllä ei tarkoiteta itse pistepilven datan muuttamista, vaan esimerkiksi ihmismallin lisäämistä visualisointitarkoituksiin tai mittauksien tekemistä. Ratkaisua on tarkoitus pystyä hyödyntämään toimeksiantajan työtehtävissä. Opinnäytetyön keskeisimmiksi kysymyksiksi nousivat:

- Mitä olemassa olevia ratkaisuja pistepilvien visualisointiin on saatavilla ja kuinka hyvin ne täyttävät toimeksiantajan vaatimukset?
- Voiko tällaisen pistepilvien visualisointiohjelmiston tehdä itse esimerkiksi Unity -pelimoottoria hyödyntäen?

2 Pistepilvet

Opinnäytetyön tietoperusta koostuu pistepilvistä; pistepilvien ominaisuuksita, tuottamisesta, laadusta sekä niiden eri tiedostomuodoista. Lisäksi käsitellään myös pistepilven käyttökohteita. Tietoperusta tarjoaa kokonaisuutena kattavan pohjan opinnäytetyön toiminnalliselle osuudelle. Pistepilvien käsittelyn jälkeen siirrytään Unity-pelimoottorin kuvaukseen.

2.1 Pistepilvien ominaisuudet

Pistepilvi on laserkeilausprosessin tuloksena syntyvä tietorakenne, joka sisältää koordinaattitiedot ja sen lisäksi joskus myös väritiedot (kuva 1). Laserkeilain kerää skannaamalla pistetiedot jäljennettävästä tilasta, pistetiedot syntyvät lasersäteen kohdatessa jäljitettävän pinnan. Lasersäteen muodostamista pistetiedoista kootaan pistepilvi, eli muodostuva kolmiulotteinen kokonaisuus mallinnettavasta tilasta. (Miinalainen, 2019, s. 9)

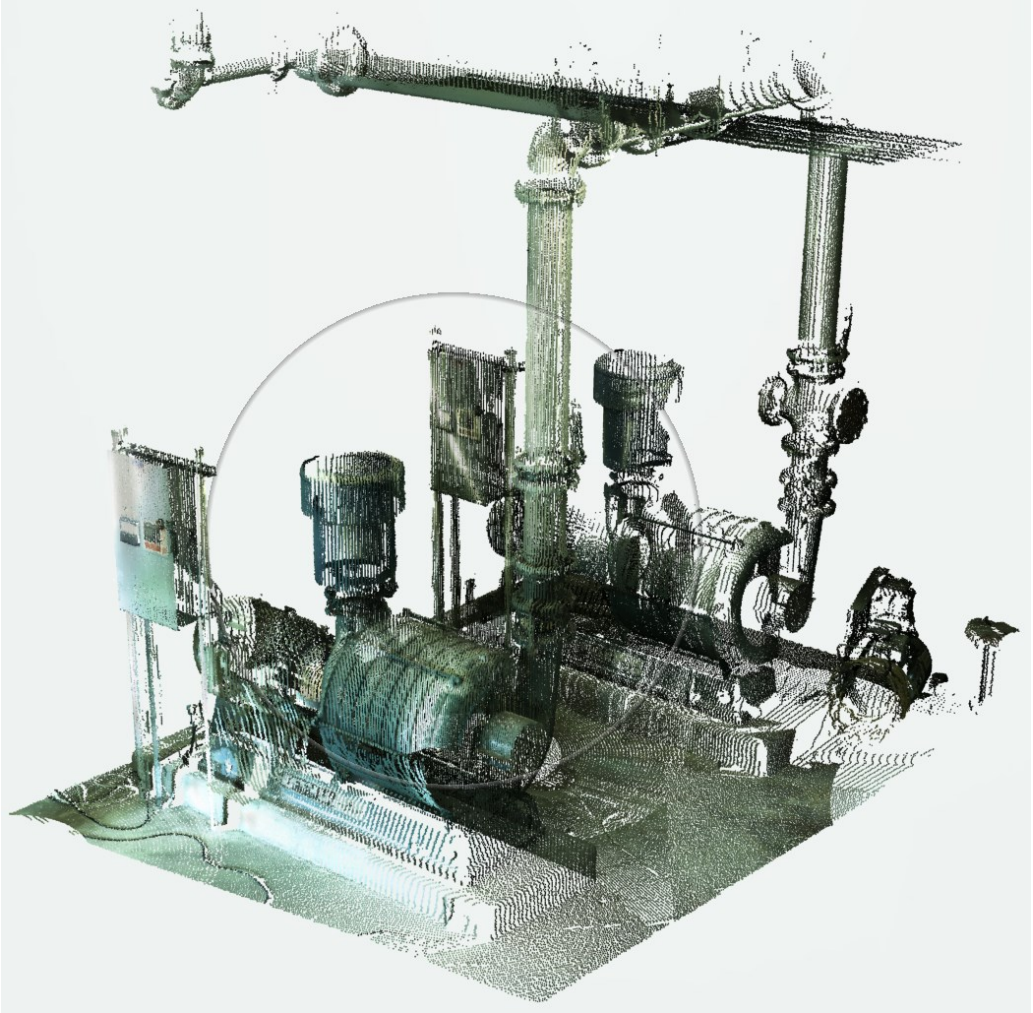
Pistepilviä on mahdollista hyödyntää rakennusten suunnittelussa ja erilaisten tilamallien muodostamisessa. Pistepilviä voi hyödyntää myös esimerkiksi maastonkartoituksessa ja kaavoituksessa sekä muissa vastaavissa analyyseissä ja seurannoissa. (Maanmittauslaitos, n.d.) Pistepilven tiedostomuotoja on useita erilaisia, jotka ovat niin avoimia kuin myös suojattuja tiedostomuotoja (Miinalainen, 2019, s. 15).

Avoimia tiedostomuotoja ovat esimerkiksi ASCII, LAS ja E57. Nämä tiedostomuodot eivät ole riippuvaisia laitevalmistajista. E57-tiedostomuoto on ASTM International standardisoimisjärjestön standardi pistepilvien tiedostomuodoille. Ohjelmistoja, jotka voivat käsitellä myös avoimia tiedostomuotoja ovat esimerkiksi ReCap ja CloudCompare. (Miinalainen, 2019, s. 15)

Suojattuja tiedostomuotoja ovat esimerkiksi RCP, PTX ja FLS. Nämä tiedostomuodot ovat riippuvaisia laitevalmistajista ja niiden käsittelyyn käytetään yleensä laitevalmistajan omia käsittelyohjelmia. Tällaisia käsittelyohjelmia on esimerkiksi Autodeskin, Faron Scene ja Leican Cyclone -ohjelmistot. Autodeskin uudempiin ohjelmistoihin soveltuu vain RCP

tiedostomuodossa olevat tiedostot. Scene on optimoitu Faron omiin tiedostomuotoihin kuten FLS, mutta se pystyy käsittelemään myös muita tiedostomuotoja. (Miinalainen, 2019, s. 15)

Kuva 1 E57 pistepilvi pumpusta.



2.1.1 Pistepilvien tuottaminen

Pistepilviaineistoa voi hankkia monilla eri tavoin kuten stereokameroilla, time-of-flight kameroilla, 3D-skannereilla, laserkeilaimella (kuva 2) tai aineistoa voidaan tuottaa keinotekoisesti tietokoneella. (Point Cloud Library, n.d.). Opinnäytetyön toimeksiantaja käyttää pistepilviaineiston tuottamiseen laserkeilainta. Laserkeilaus on menetelmä, jolla voidaan mitata esineen, kohteen- tai alueen muoto, koko ja sijainti. Menetelmän lopputuloksena saadaan kolmiulotteinen pistepilvi.

Laserkeilaimen toiminta perustuu valon kulkuaikaan, laserin vaihe-eroon tai kolmiomittaukseen. Laserkeilain lähettää laserpulsseja ympärilleen, ja kun lähetettyä sekä takaisinheijastunutta lasersädettä verrataan, voidaan laskea etäisyys kohteen ja keilaimen välillä. Lasersäteen lähtökulman perusteella saadaan selville kohteen suuntatieto keilaimen sijaintiin nähden. Laserkeilaimella on mahdollista skannata suuria kokonaisuuksia kerralla. Keilaimen voi tukea kolmijalkaan tai kiinnittää esimerkiksi lentokoneeseen. Laserkeilain itsessään muodostuu laserlähteestä, keilainosasta ja ilmaisinosasta. Keilainosa poikkeuttaa lasersäteen ja ilmaisinosa tulkitsee vastaanotetun signaalin. (Ahonen, 2015, ss. 7 – 8; Santaluoto, 2012, s. 8)

Kuva 2 Trimble X7-laserkeilain (Geospatial World, n.d.)

2.1.2 Pistepilven laatu

Välimatka pistepilven keskinäisten pisteiden välillä vaikuttaa merkittävästi mallin laatuun. Pistepilven pistetiheyden tulee olla mahdollisimman tiheä, jotta mallista tulisi tarpeeksi

yksityiskohtainen. Laserkeilain tallentaa keilatusta pisteestä palaavan signaalin voimakkuuden eli intensiteetin ja asettaa pisteelle sen intensiteetin mukaan sävyarvon. Pinnan materiaali on suuresti vaikuttava tekijä säteen vahvuudessa. (Ahonen, 2015, s. 14)

Säteeseen erityisesti vaikuttavia pintoja ovat absorboivat-, kiiltävät- ja läpikuultavat pinnat. Tällaisia ovat esimerkiksi väritykseltään mustat pinnat, kiillotetut metallit ja kirkas lasi. Lasin läpi kulkiessaan lasersäde aiheuttaa vääristymiä, jolloin mittauksen tarkkuus heikkenee. (Ahonen, 2015, ss. 13 – 15) Pinnan materiaalin lisäksi kohteen muoto vaikuttaa myös paluusignaalin voimakkuuteen. Mittauksen tarkkuus on riippuvainen myös etäisyydestä. Mitä suurempi mittausmatka on, sitä heikompi on paluusignaalin vahvuus. (Cronvall Timo et al., 2012, s. 19)

2.1.3 Pistepilvien tiedostomuodot

Pistepilvien tiedostomuotoja on useita. Faro, Leica ja Trimble ovat suuria laite- sekä ohjelmistovalmistajia pistepilvien tuottamiseen ja katselemiseen. Suurimmat erot eri pistepilvitiedostoformaattien välillä on ASCIIn ja binäärimuodon käytössä. ASCII muotoiset pistepilvitiedostot välittävät pistepilven tiedot luettavana tekstinä. ASCII muotoisessa pistepilvitiedostossa on rivittäin kerrottu laserin heijastuksen koordinaatit sekä tiedostomuodosta riippuen lisäinformaatiota kuten heijastuksen intensiteetti tai väriarvo.

ASCII tiedostomuodon etuna on, että tiedosto on helposti katseltavissa esimerkiksi normaalilla tekstieditorilla. ASCII muotoa suositaan esimerkiksi pistepilvitiedostojen arkistoinnissa. (Archaeology Data Service, n.d.; Vercator, n.d.) Kuvassa 3 on avattuna pts muotoinen ASCII pistepilvi tekstieditorissa. Ensimmäinen rivi kertoo pisteiden lukumäärän, ja sen jälkeen tulevat rivit käyvät yksittäisen pisteen arvot läpi. Kuvassa 3 esitellään myös pts-tiedoston sisältö; kolme ensimmäistä lukua kertovat pisteen XYZ koordinaattitiedot, sen jälkeen tuleva luku kertoo heijastuksen intensiteetin ja tämän jälkeen tulevat kolme lukua kertovat pisteen RGB väriarvot.

Kuva 3 ASCII muotoinen pts-tiedosto avattuna tekstieditorissa.

```
174479
-2.30590000 -3.76740000 1.37120000 -1871 13 16 7
-2.29720000 -3.76050000 1.38000000 -1871 12 15 8
-2.28880000 -3.75340000 1.40060000 -1839 16 19 8
-2.30920000 -3.76160000 1.32160000 -1967 6 8 3
-2.30540000 -3.75510000 1.32500000 -1935 8 11 4
```

Binäärimuodossa olevat pistepilvitiedostot tallentavat tiedot suoraan binäärikoodina. Binäärimuodossa olevat tiedostot ovat pienempikokoisia ja voivat sisältää enemmän informaatiota. Binäärimuotoisiin pistepilvitiedostoihin voi liittää esimerkiksi jokaisen pisteen tietoihin tiedoston allekirjoituksen, tietoa ohjelmistosta ja muuta metadataa. Binäärimuotoisten pistepilvien katselu ja käsittely on myös nopeampaa kuin ASCII muotoisten, sillä ne voidaan lukea osina rivi riviltä luvun sijaan. Useimmat pistepilviä käsittelevät ohjelmistot pystyvät tuomaan laajan skaalan eri pistepilvitiedostoja, ja pystyvät myös viemään tiedostoja monessa eri muodossa. (Archaeology Data Service, n.d.; Vercator, n.d.) Taulukossa 1 on vertailtu muutamia yleisimpiä tiedostomuotoja sekä niiden ominaisuuksia.

Taulukko 1 Pistepilvitiedostojen tyypit ja ominaisuudet. (Wiki, n.d.)

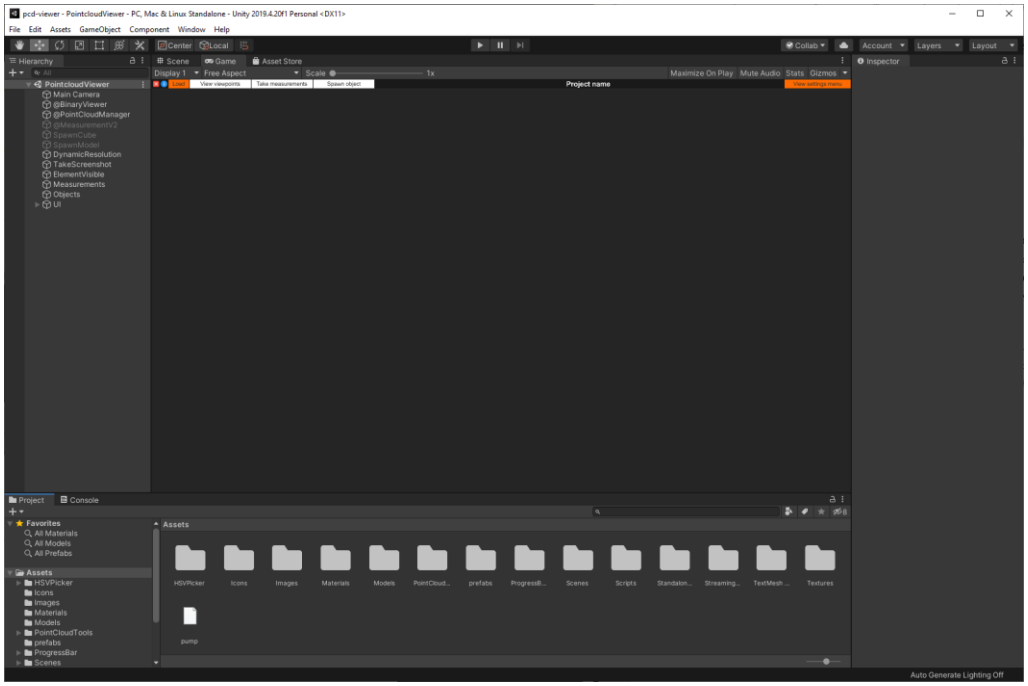
Tyyppi	Pääte	Kuvaus	ASCII/Binääri
ASCII	.asc, .txt, .xyz, .pts	ASCII muotoinen pistepilvitiedosto	ASCII
LAS	.las	ASPRS:n ylläpitämä tiedostomuoto	Binääri
E57	.e57	ASTM:n ylläpitämä tiedostomuoto	Sekoitus ASCIIta ja binääriä
PTX	.ptx	LEICAn pistepilvi tiedostomuoto	ASCII
FARO	.fls, *.fws	FAROn pistepilvi tiedostomuoto	Binääri
PLY	.ply	Stanfordin yliopiston 3D geometria formaatti.	Molemmat

3 Unity-pelimoottori

Termillä pelimoottori tarkoitetaan ohjelmistoa, jota hyödyntämällä saadaan luotua peliin tietty haluttu sisältö sekä säännöt. Pelin taustatyön hoitaa kuitenkin itse pelimoottori. Pelimoottori huolehtii fysiikkamallinnuksesta, grafiikoiden mallinnuksesta, ohjainlaitteista, tekoälystä ja äänistä. Pelimoottorit on mahdollista jakaa kolmeen pääryhmään, ylä-, keski- ja alatason pelimoottoreihin. (Petrell, 2017, ss. 9 – 10)

Tässä opinnäytetyössä hyödynnettävä Unity kuuluu ylätason pelimoottoreihin. Unity on Unity Technologiesin kehittämä työkalu (kuva 4), jonka avulla on mahdollista tuottaa kaksi- sekä kolmiulotteisia sovelluksia PC:lle, konsoleille ja mobiililaitteille. Unity on yksityishenkilöille ja pienyrityksille ilmainen. (Lamberg, 2020, s. 8)

Kuva 4 Kuvakaappaus Unityn käyttöliittymän ulkoasusta.



Unity on perusteiltaan nimensä mukaisesti pelimoottori, mutta sillä voi toteuttaa myös sovelluksia hyötykäyttöön. Unityä voi hyödyntää esimerkiksi virtuaalisissa asuntonäytöissä jo ennen kuin itse taloa on edes alettu rakentamaan. (Lamberg, 2020, s. 8) Muita mahdollisia käyttötapoja on esimerkiksi viranomaisten harjoittelu virtuaalimaailmassa. Unityllä on mahdollista luoda huokeasti tosielämän tilannetta vastaava harjoitteluympäristö esimerkiksi poliisille tai muille viranomaistahoille. (European Comission, 2018)

Unityn pelilogiikan skriptauskielenä toimii Microsoftin .NET-alustan C#-ohjelmointikieli. C# syntaksi on hyvin läheinen muiden C-kielien kuten C ja C++ kanssa, mutta muistuttaa myös paljon Java-ohjelmointikieltä. C# on hyvin monikäyttöinen ohjelmointikieli ja sitä käytetäänkin peliohjelmoinnin lisäksi myös web-kehityksessä sekä työpöytäohjelmistojen- ja mobiilisovellusten kehityksessä. C# yksi suurimmista vahvuuksista on, että se on helposti jaettavissa usealle eri alustalle. Tämä on erityisesti pelikehityksessä hyödyllistä, koska samaa koodia voidaan käyttää eri alustoille kehitettäessä uudelleen, koodin uudelleenkirjoituksen sijaan. (Lamberg, 2020, s. 9; W3Schools, n.d.)

4 Opinnäytetyön lähtökohdat

Opinnäytetyön lähtökohtana on löytää uusia toimivia ratkaisuja työelämän tarpeisiin sekä tuottaa arvokasta tietoa työn tilaajalle. Opinnäytetyö kehittää myös omaa ammatillista osaamista niin raportin tekemisestä kuin eri tietojenkäsittelyn aihealueista. Seuraavaksi kuvataan opinnäytetyön tavoite ja tarkoitus sekä esitellään työn tilaaja.

4.1 Työn tavoite ja tarkoitus

Opinnäytetyön tarkoituksena on esittää vertailua pistepilvien käsittelyyn ja katseluun tarkoitettujen ohjelmistojen välillä, sekä kuvata prototyypin kehittäminen ja sen vaiheet. Opinnäytetyön aihe nousi työelämän tarpeista löytää tehokas ja toimiva ratkaisu pistepilviaineistojen hyödyntämiseen. Tavoitteena on tuottaa selkeä ja kattava kuvaus selvitystyöstä ratkaisun löytämiseksi.

Opinnäytetyö on luonteeltaan toiminnallinen opinnäytetyö. Toiminnallinen opinnäytetyö nousee työelämän tarpeista ja tarkoituksena on kehittää käytännön toimintaa. Toiminnalliseen opinnäytetyöhön sisältyy teoreettinen ja toiminnallinen osuus. Tämä opinnäytetyö on luonteeltaan toiminnallinen, sillä itse työ on selvitys projektin toteutusprosessista (Ammattikorkeakoulu, n.d.)

4.2 Työn tilaajana Keskusrikospoliisi

Opinnäytetyön tilaajana toimii Keskusrikospoliisin rikostekninen laboratorio. Rikostekninen laboratorio tekee tutkimuksia poliisin sekä muiden viranomaisten toimeksiannoista. Tutkimuksien perusteella laboratorio antaa lausuntoja käytettäväksi esimerkiksi tutkinnassa ja oikeusprosesseissa. Rikosteknisessä laboratoriossa tutkitaan vuosittain yli 100 000 näytettä. Rikosteknisen laboratorion toiminnassa hyödynnetään kattavasti laboratorioautomatiikkaa sekä erilaisia tietojärjestelmiä. (Poliisi, 2020)

5 Projektin suunnittelu ja toteutus

Toiminnallisen opinnäytetyön olennaisena osana on kattava kuvaus työvaiheista, jotka tässä työssä ovat selvitystyön sekä prototyypin kehittämisen kuvaus. Opinnäytetyö on kehittämispainotteinen, joten projektiosuuden kuvaus sisältää suunnittelu- ja toteutusosuuden sekä valmiin tuotoksen kuvauksen. Seuraavaksi käydään läpi suunnittelun vaiheet sekä yksityiskohtaisemmin toteutus jaettuna eri aihepiireihin.

5.1 Suunnittelu

Tehtävänä oli siis kartoittaa keinoja pistepilvien helppoon ja nopeaan visualisointiin. Visualisointi olisi hyvä nähdä myös virtuaalitodellisuudessa. Lisäksi pistepilviin olisi hyvä pystyä lisäämään esimerkiksi henkilöhahmojen malleja. Projektin alkuvaiheessa pistepilvet olivat itselleni vieras käsite. Projekti aloitettiin näin ollen tutustumalla pistepilviin teoriaosuudessa mainittujen materiaalien avulla. Kuten teoriaosuudessa kerrottiin, pistepilvi on lyhykäisyydessään esimerkiksi laserkeilaamalla saatu koordinaattitietoja sekä heijastus ja/tai väriarvoja sisältävä tiedosto.

Syötettäessä tiedosto ohjelmaan, joka osaa tulkita sitä, piirtyy ruudulle kolmiulotteinen malli pistepilvestä. Yksi vaihtoehto pistepilven mahdollisesta jatkokäsittelystä sekä katselusta oli, että pistepilvestä tehtäisi mesh-malli. Tämä vaihtoehto suljettiin kuitenkin pois sillä mallinnus ei ole toimeksiantajan vaatimuksiin tarpeeksi tarkka. Kuitenkin on mahdollista, että yksitoikkoiset pinnat kuten valkoiset seinät tai katot voisi mallintaa meshinä suorituskyvyn parantamiseksi.

Kun pistepilvet olivat tulleet tutuiksi, selvitettiin mitkä ohjelmistot olisivat parhaita niiden visualisointiin. Ohjelmiston vaatimuksena oli, että se olisi mahdollisimman helppokäyttöinen ja sisältäisi työkaluja kuten mittauksen ja kuvakulman tallennuksen. Toivottuja ominaisuuksia olivat myös selkeä käyttöliittymä, esineiden- sekä objektien lisäys pistepilveen ja VR-lasien käyttömahdollisuus. Esille nousi Faro Scene LT, jonka ominaisuudet vastasivat toimeksiantajan toiveita.

Projektin loppuosan tarkoituksena oli selvittää olisiko mahdollista tehdä visualisointiohjelmisto talonsisäisesti esimerkiksi Unity-pelimoottoria hyödyntäen. Pelimoottoriksi valikoitui Unity Unreal Enginen sijaan, sillä siitä oli ennestään kokemusta. Talonsisäisen ohjelmiston etuna on, että ominaisuudet ja ulkoasu on täysin räätälöitävissä omien tarpeiden mukaisesti. Etuna on kehityksen kannalta myös se, että Unity-pelimoottorissa on laajat kirjastot, joiden avulla voi mahdollistaa esimerkiksi VR-lasien käytön projektissa.

Suunniteltaessa Unity-prototyyppiä tutkittiin onko vastaavia projekteja jo tehty ja olisiko mahdollista hyödyntää valmiita kirjastoja tai ohjelmistoja prototyypin teossa. Vastaavanlaisia varteenotettavia ohjelmistoja oli Ljungbergslaboratorietin tekemä ilmainen PointCloud XR sekä Unityn kaupasta löytyvä maksullinen Point Cloud Viewer and Tools -kirjasto. Projektissa päädyttiin hyödyntämään Point Cloud Viewer and Tools -kirjastoa sen kattavampien ominaisuuksien vuoksi. Prototyypin ulkoasuun sekä toimintaan otettiin mallia Faron Scene LT -ohjelmistosta.

5.2 Toteutus

Suunnitteluvaiheen jälkeen testattiin Faro Scene LT -ohjelmistoa eri tilanteissa ja pistepilvitiedostoja hyödyntäen ja aloitettiin prototyyppiprojektin toteuttaminen. Työskentelin projektin parissa yksin, joten toteutus oli hyvin vapaamuotoista. Työskentelytapa oli SCRUMia mukailevaa. Projektin vaatimukset selvennettiin ennen projektin aloittamista ja sitä tehdessä suoritettiin myös testausta toteutuksen yhteydessä. Projektin ominaisuuksien kehittäminen rakentui niiden tärkeyden mukaan. Niin kutsuttuja korkean prioriteetin ominaisuuksia olivat pistepilvessä liikkuminen ja sen tarkastelu, mittauksien tekeminen sekä kuvakulmien tallentaminen.

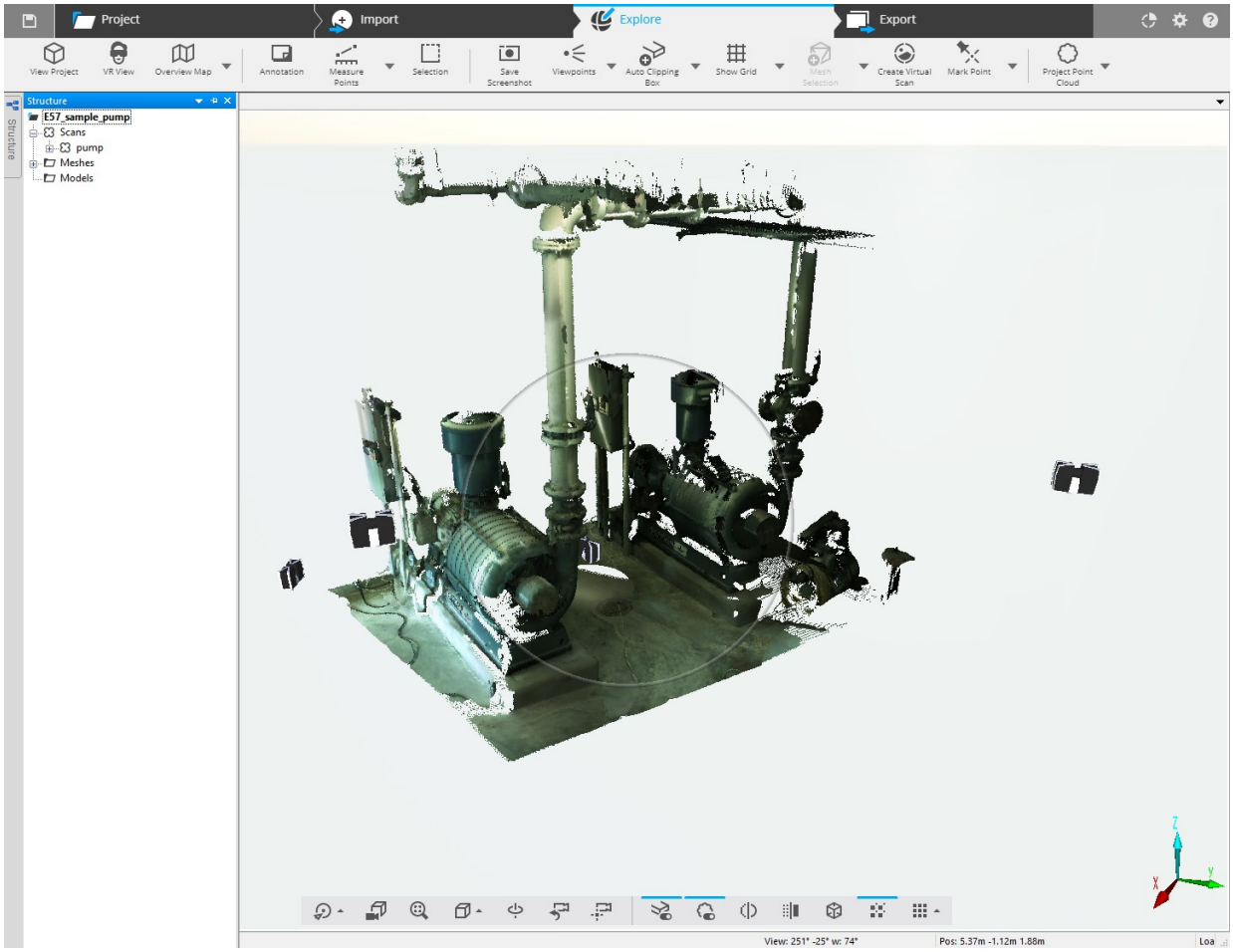
Prototyyppi päädyttiin toteuttamaan niin, että se ottaa vastaan ainoastaan pts-tiedostoja. Näin valikoitui sillä prototyypin ei tarvitse toimia itse pistepilven tuonnissa kovinkaan kattavasti, vaan kyseessä oli enemmänkin idean toteuttamiskelpoisuutta esittelevä vajavainen toteutus. Pistepilvitiedostomuodoksi valikoitui pts-tiedosto, koska se on ASCII muotoinen ja suhteellisen yleinen pistepilvitiedostomuoto.

5.2.1 Faro Scene LT

Ainoastaan Faro Scene LT täytti täysin toimeksiantajan vaatimukset pistepilven visualisointiohjelmistolta (kuva 5). Scene tukee kattavasti eri pistepilvitiedostomuotoja. Pistepilvien lisäksi projektiin voi tuoda 3D-malleja. Ohjelmisto tarjoaa valmiita työkaluja pistepilvien visualisointiin sekä julkaisumahdollisuuden SCENE Webshare Cloud -työkalulla. Pistepilvitiedosto on mahdollista viedä moneen eri tiedostoformaattiin käsittelyn jälkeen.

Scene sisältää kattavan määrän ominaisuuksia, joihin kuuluu esimerkiksi pistepilven katselu ja sen sisällä liikkuminen, mittauksien tekeminen ja -tallentaminen, kommenttien lisäys, kuvakulmien tallentaminen sekä VR-tila ja kartan automaattinen luominen pistepilvestä. Faro tarjoaa ohjelmistoon myös SDK:n, jonka avulla voi kirjoittaa omia kirjastoja C-ohjelmointikieltä käyttäen.

Kuva 5 Scene LT käyttöliittymä.



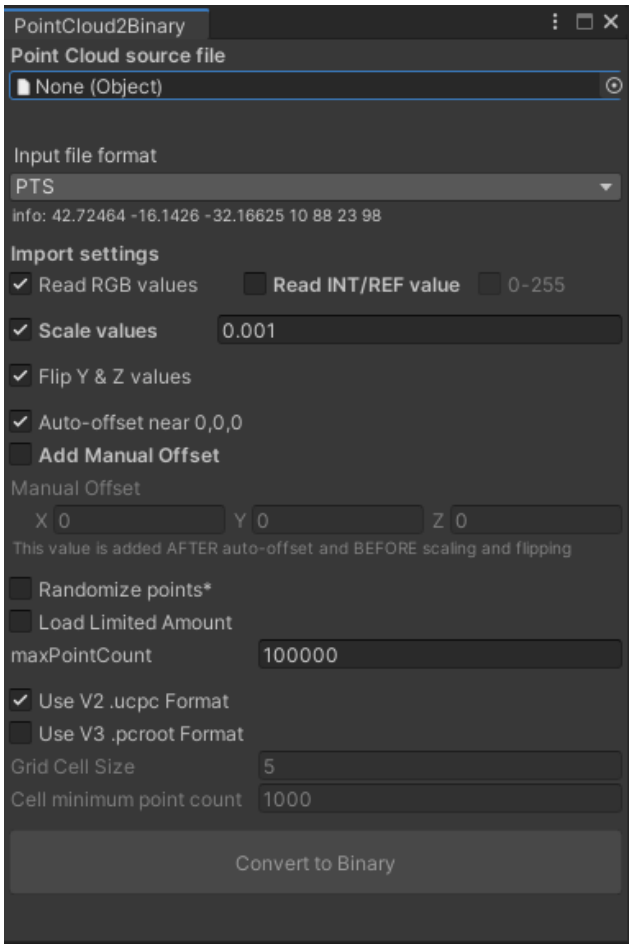
SCENEssä on mahdollista valita viiden eri visualisointiasetuksen välillä. ”Supersampling” renderöi pistepilviaineiston käytetyn näytön resoluutiota suuremmalla resoluutiolla ja pakottaa tämän jälkeen aineiston sopimaan takaisin näytön resoluutioon. Tämä tuottaa tasaisemman ja yksityiskohtaisemman ulkonäön aineistolle. ”Gap Filling” täyttää lähellä olevien pisteiden väliin jääneet aukot selkeämmän ulkonäön aikaansaamiseksi. ”Clear View” esittää harvan pistetiheyden omaavat alueet selkeämpinä ja tiheän pistetiheyden omaavat alueen pisteet esitetään kirkkaampina. Tämä lisää läpinäkyvyyttä aineistoon ja auttaa esimerkiksi seinäntakaisten rakenteiden havaittavuudessa. ”Point Sizes” asetuksella voi säätää pisteiden kokoa kolmen eri koon väliltä. ”Adaptive” näyttää pisteet erikokoisina suhteessa kameraan, kauempana olevat pisteet visualisoidaan pienempinä ja lähellä olevat suurempina. (Kokkonen, 2019, s. 39)

5.2.2 Point Cloud Viewer and Tools

Toteutusvaihe aloitettiin testaamalla Point Cloud Viewer and Tools -kirjastoa sekä sen sisältämiä esimerkkiscenejä. Esimerkit sisälsivät muun muassa mittaustyökalun esittelyn. Kirjasto muuttaa vastaanotetun pistepilven omaan binäärimuotoiseen tiedostomuotoon. Tiedostomuotoja on valittavissa kaksi, ucpc sekä uudempi pcroot. Muunnos tehdään koska, binäärimuotoiset pistepilvet ovat nopeampia lukea (Kuva 6). Muunnoksen yhteydessä valitaan tuotavan pistepilven ominaisuudet, kuten sisältääkö se väritietoja tai halutaanko vain osa pisteistä ladata. Tämä prosessi saattaa pistepilven koon mukaan kestää useita kymmeniä minuutteja.

Kirjastolla on myös mahdollista tuoda pistepilvi alkuperäisessä muodossaan suoraan ajonaikana. Tämä on kuitenkin raskaampaa kuin pistepilven lukeminen binäärimuotoisesta tiedostosta. Kirjaston perusominaisuuksiin kuuluu, että pistepilveä voi katsella ja sen sisällä voi liikkua näppäimistön sekä hiiren avulla. Pistepilvi ei näy Unityn editoritilassa vaan se piirtyy vasta kun ohjelma ajetaan. Kirjaston mukana tuli myös pistepilven visualisointiin suunniteltuja materiaaleja. Materiaali määrittää kuinka pistepilven pinta tulisi piirtää ja millainen sen tekstuuri on (Unity Technologies, n.d.-a). Mukana tulleiden materiaalien asetuksissa oli myös liukusäädin, jonka avulla pisteiden kokoa voi säätää.

Kuva 6 Point cloud viewer and Tools muuntoikkuna.



5.2.3 Point Cloud XR

Suunnitteluvaiheessa mainitsemaani Ljunbörslaboratorietin tuottamaa Point Cloud XR -ohjelmistoa testattiin toteutusvaiheessa. Tämä ei ollut Unityyn tuotava kirjasto vaan valmis visualisointiin tarkoitettu Unityllä tuotettu ohjelmisto. Testin tarkoituksena oli saada hieman mallia kuinka oman prototyypin tulisi toimia. Tämä ohjelmisto otti vastaan vain las-muotoisia tiedostoja ja se toimi ainoastaan virtuaalilasien avulla. Ohjelmiston ominaisuuksiin kuului mittauksien tekeminen, pistepilvien manipulointi ja -valinta sekä muokatun pistepilven vienti las-muodossa. Ohjelmistoa ei saatu kuitenkaan täysin näyttämään pistepilveä syystä, jota ei saatu selville. Tämän oudon ongelman vuoksi ohjelmaa ei kunnolla pystytty testaamaan. Ohjelmiston koodia kuitenkin tutkittiin, jotta ymmärrettäisiin kuinka se oli luotu ja voisiko sitä hyödyntää prototyypin teossa.

5.2.4 Unity-prototyypin kehitys

Unity-projektin kehittäminen aloitettiin suunnittelemalla ohjelmiston käyttöliittymää. Se suunniteltiin niin, että siirtymiä olisi mahdollisimman vähän, jos ollenkaan ja kaikki tapahtuisi yhden ikkunan kautta. Unity tarjoaa valmiin Unity UI-työkalun käyttöliittymän kehittämiseen. Tämä työkalu tarjosi kaiken tarvittavan yksinkertaisen käyttöliittymän tekemiseen. Käyttöliittymän tekeminen aloitettiin luomalla ruudun yläreunaan koko ruudun levyinen valintarivi, johon sijoitettiin kaikki tarvittavat nappulat. Valintariviin lisättiin valintapainikkeet poistumiselle, pistepilventuonnille, kuvakulman valinnalle, mittausten tekemiselle sekä objektien lisäykselle. Valintariviin tehtiin myös tekstikenttä, johon tuli avatun projektin nimi. Oikeaan laitaan lisättiin vielä painike, josta pääsee asetuksiin. Tämän painikkeen alapuolelle ei toistaiseksi tullut muita ominaisuuksia kuin pistepilven resoluution muuttaminen. Poistumispainikkeella ei ole varsinaista funktiota, ellei ohjelmistoa ole käännetty suoritettavaan EXE-tiedostomuotoon. Tuontinappulalla avataan valintaikkuna, josta voi valita tuotavan pistepilvitiedoston.

Pistepilvitiedoston polun etsimiseen käytettiin valmista kirjastoa. Tämän avulla pystyttiin helposti luomaan Windowsin omaa resurssienhallintaa muistuttavan valintaikkunan. Asetuksia muutettiin vielä niin, että se löytää ainoastaan pts-tiedostot. Tämä hakutoiminto lisättiin pistepilven konversion tekevään skriptiin. Tämä valikko tallentaa pistepilven polun suoraan binäärimuunnoksen tekevän skriptin muuttujaan olion kautta ja kutsuu sen jälkeen funktiota, joka tekee muunnoksen. Tämä on esiteltynä ohjelmakoodissa 7.

Ohjelmakoodi 7 Skripti, jolla haetaan pistepilvitiedosto muunnosta varten.

```
void Start()
{
    converter = new LoadPtsConvertBin();

    var button = GetComponent<Button>();
    button.onClick.AddListener(OnClick);
}

private void OnClick()
{
    var paths = StandaloneFileBrowser.OpenFilePanel("Open pointcloud file", "", "pts", false);
    if (paths.Length > 0)
    {
        converter.filepath = paths[0];
        converter.ConvertBinaryV2();
    }
}
```

Kuten aiemmin mainittiin, Point Cloud Viewer and Tools ei toimi aivan niin kuin alun perin oli toivottu, sillä pistepilveä ei ole mahdollista muuntaa ja ladata visualisoitavaksi silloin kun ohjelma on käynnissä. Seuraavaksi siirryttiin muokkaamaan kirjaston skriptiä, jolla konvertoidaan tuotu pistepilvi kirjaston omaan muotoon. Koska Point Cloud Viewer and Tools -kirjasto on maksullinen, ei tehtyjä muutoksia ole mahdollista näyttää. Skriptissä oli ehtolauseet usealle eri tuontimuodolle, mutta koska työhön riitti tuonti ainoastaan pts-muodossa, poistettiin kaikki ylimääräiset tuontivaihtoehdot. Kirjastosta poistettiin myös oman käyttöliittymän piirtämiseen liittyvät osat. Lopuksi skriptiä muutettiin niin, että se tallentaa käännetyn pistepilven ucpc-muotoon, ja tallentaa sen projekti kansioon. Vanhempi binäärimuoto valikoitui, koska esimerkiksi mittauksia tekevä esimerkkiscene toimi ainoastaan tällä muodolla. Muutosten tekeminen ei ollut kovinkaan nopea prosessi, sillä skripti ei ollut itselleni ennestään tuttu ja jotta olisi mahdollista tehdä vaaditut muutokset täytyi siihen tutustua yleisellä tasolla.

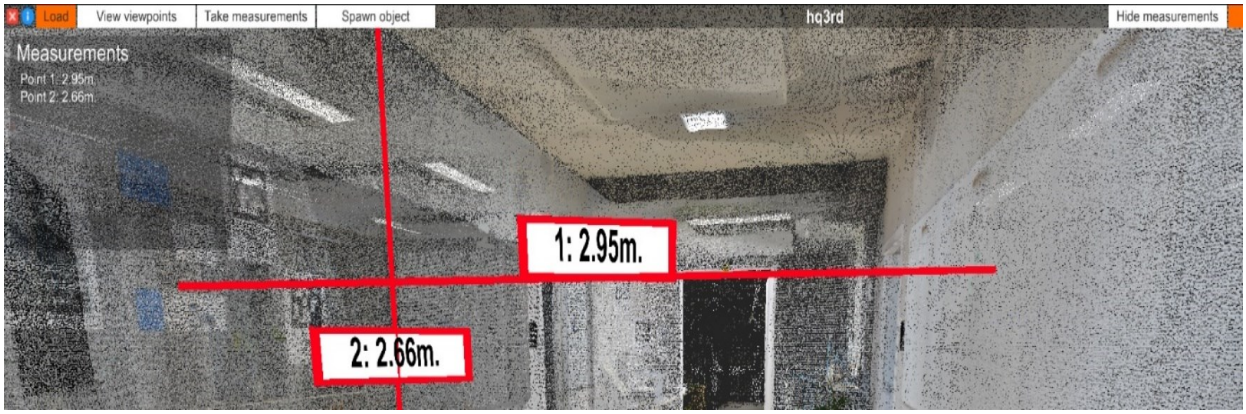
Projektin pohjana käytettiin kirjaston esimerkkisceneä mittaustyökalun käytöstä. Tähän esimerkkisceneen on valmiiksi sisällytetty peliobjekti, joka on vastuussa mittausten tekemisestä sekä peliobjekti, joka lukee pistepilven. Jälkimmäisessä peliobjektissa on skripti, jossa on valintaruutu muunnetun pistepilven valitsemiseen, sekä pistepilven materiaalin valitsemiseen. Skriptiä muokattiin niin, että se saa luettavan pistepilven nimen suoraan skriptiltä, joka tekee käännöksen binäärimuotoon. Työhön esivalittiin myös yleispätevä materiaali pistepilvelle. Tämän jälkeen suoritettiin testauksia eri pts-tiedostoilla ja todettiin ratkaisu toimivaksi.

5.2.5 Mittaustyökalu

Kun visualisointi oli saatu toimimaan, oli aika kehittää ominaisuuksia. Mittausominaisuus oli yksi prioriteettilistalla olevista ja se toimikin jo vähimmäisvaatimuksin pohjana käytetyssä esimerkkiscenessä. Valmiina oleva mittausfunktio toimi muuten hyvin, mutta se ei tallentanut mittaustuloksia. Näin ollen lähdettiin jatkokehittämään mittausta suorittavaa skriptiä. Skriptiin luotiin taulukko, johon tallennettiin mittauksen alkupiste, päättymispiste ja mittauksen tulos. Käyttöliittymään luotiin uusi ikkuna, johon listasin edellä mainitun

taulukon. Kun mittauksia suoritettiin, ne tallentuivat taulukkoon sekä ilmestyivät luotuun ikkunaan. Mittauksessa käytettävää prefabia muutettiin vielä niin, että mittaustuloksen ilmoittavan merkinnän tyyli saadaan hienostuneemmaksi. Mittaus toimii niin että käyttäjä painaa valikosta ”Take measurements”-painiketta, jonka jälkeen käyttäjä voi valita kaksi pistettä, joiden välisen etäisyyden käyttäjä haluaa tietää. Valittuaan mittauksen pisteet, ohjelma piirtää niiden välille punaisen viivan, jonka keskelle tulee mittauksen tulos. Mittaviivat sekä tulosikkunan voi myös piilottaa painamalla ”Hide measurements”-painiketta (kuva 8). Vasemmalla on ikkuna, jossa näkyy mittaustulokset. Keskellä menevät punaiset viivat esittävät mittapisteiden väliä.

Kuva 8 Mittausominaisuuden toiminta.



Mittauksen tekevä koodi käyttää kirjaston mukana tullutta pisteenvalitsintyökalua pisteenvalitsemiseen. Skripti tarkistaa valitsiko käyttäjä ensimmäisen- vai toisen pisteen. Jos piste oli ensimmäinen, yläpalkkiin ilmestyy teksti, jossa käyttäjää kehotetaan valitsemaan toinenkin piste. Mikäli valinta oli toinen, laskee skripti mittaetäisyyden Unityn Vector3.Distance-metodia käyttäen. Tämän jälkeen mittatulos lisätään viivanyläpuolelle erillisenä prefabina sekä tulosikkunaan.

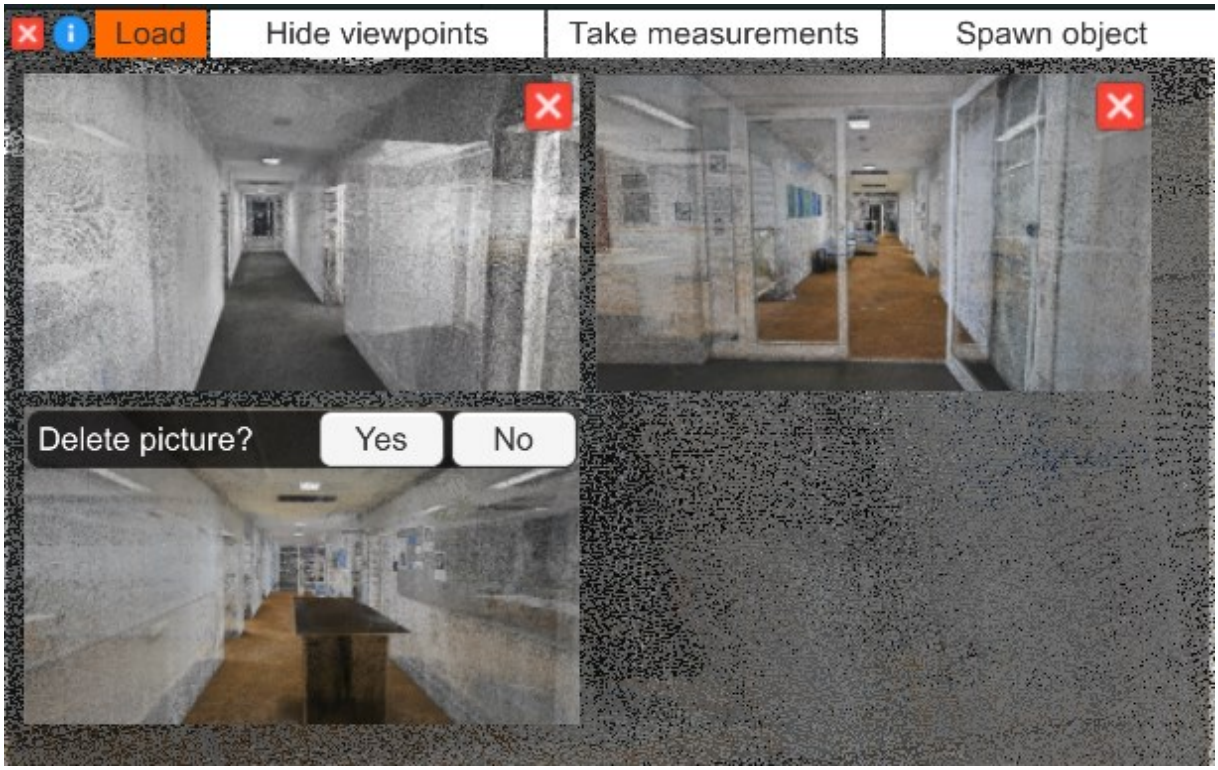
5.2.6 Kuvakulman tallennus

Seuraavaksi vuorossa oli niin kutsutun “Viewpointin” eli kuvakulman tallentaminen. Ominaisuuden ideana on, että kun ohjelmiston käyttäjä tutkii pistepilveä, käyttäjä saattaa haluta tallentaa tietyn kuvakulman palatakseen siihen. Jotta käyttäjän ei tarvitse etsiä kameraa liikuttamalla samaa kohtaa uudestaan, voi hän sen sijaan valita valintaikkunasta

tallennetun kuvakulman ja palata suoraan siihen. Ensin luotiin uusi ikkuna ohjelmiston käyttöliittymään (kuva 9). Ikkunassa on lista, johon otetut kuvat tallennetaan ruudukkona. Ruudukkoon tulevat kuvat ovat prefabeja, joihin on tallennettuna otettu näyttökuva tekstuurina. Prefab sisältää myös skriptin, joka suorittaa kameran siirron kyseisen kuvakulman kohdalle.

Valintaikkuna hyödyntää ”PopulateGrid”-skriptiä, joka hakee Unityn oman Update-funktion avulla projektikansiosta kuvat listaan. Lopuksi kutsutaan ”Populate”-funktiota (ohjelmakoodi 10), joka hakee kuvat, ja muuntaa ne 2D tekstuureiksi. Tämän jälkeen tekstuuri asetetaan prefabiin ja prefab lisätään listaan, joka tulostetaan valintaikkunaan. Valintaikkunasta voi myös poistaa valitun kuvakulman. Painamalla poisto painiketta, kysytään käyttäjältä varmistus kuvakulman poistamisesta. Mikäli käyttäjä haluaa poistaa kuvakulman, kutsuu painike yksinkertaista skriptiä (ohjelmakoodi 11), joka poistaa sekä json-tiedoston että kuvakaappauksen tallennustilasta.

Kuva 9 Ikkuna, josta valitaan kuvakulma.



Ohjelmakoodi 10 Populate-funktio.

```
void Populate(string[] filePaths)
{
    // Metodi muuttaa kuvatiedostot spriteiksi ja luo uuden kuvaobjektin.

    // Käy läpi jokainen löydetty tiedostopolku.
    foreach (string filePath in filePaths)
    {
        // Jos tiedostopolkua ei löydy listalta jatketaan.
        if (!screenshots.Contains(filePath))
        {
            // Kuvan nimi
            string name = Path.GetFileNameWithoutExtension(filePath);
            // Luetaan kuvan bitit ja tallennetaan ne taulukkoon.
            byte[] bytes = System.IO.File.ReadAllBytes(filePath);
            // Luodaan uusi 2D tekstuuri.
            Texture2D texture = new Texture2D(1, 1);
            // Ladataan kuva bitti taulukosta.
            texture.LoadImage(bytes);

            // Luodaan uusi sprite jossa käytetään luotua tekstuuria.
            Sprite sprite = Sprite.Create(texture, new Rect(0, 0, texture.width, texture.height), new Vector2(0.5f, 0.5f));

            // Luodaan uusi objekti pictureItem prefabista.
            GameObject newObj = (GameObject)Instantiate(picture, transform);

            // Lisätään objektiin luotu sprite
            newObj.GetComponentInChildren<Image>().sprite = sprite;

            // Vaihdetaan objektin nimi alkuperäiseen.
            newObj.name = name;

            // Lisätään tiedostopolku listaan.
            screenshots.Add(filePath);
        }
    }
}
```

Ohjelmakoodi 11 Kuvakulmanpoistoskripti.

```
// Poistaa valitun näyttökuvan ja siihen liittyvän metadatan.
// Liitettynä PictureItem prefabiin.
public class DestroyImage : MonoBehaviour
{
    // Poistettava tiedosto.
    public GameObject pictureItem;
    // Projektin nimi jolla löydetään projektikansio.
    GameObject projectNamePlaceholder;
    string projectFolder;

    void Start() {
        // Etsii objektin joka sisältää avatun projektin nimen.
        projectNamePlaceholder = GameObject.FindGameObjectWithTag("ProjectName");
    }

    public void DestroyPicture() {
        projectFolder = projectNamePlaceholder.GetComponent<Text>().text;
        // Polku valittuun tiedostoon.
        string pictureToDelete = Application.persistentDataPath + "/" + projectFolder + "/Screenshots/" + pictureItem.name;

        // Tuhoaa valitun objektin.
        Destroy(pictureItem);
        // Poistaa tiedostot levyltä.
        File.Delete(pictureToDelete + ".png");
        File.Delete(pictureToDelete + ".json");
    }
}
```

Seuraavaksi luotiin uusi ”Screenshot”-skripti (ohjelmakoodi 12), jonka tarkoituksena oli suorittaa kuvanotto sekä tallentaa kuvanottotiedot. Skriptiin luotiin muuttujia, joilla määritetään näppäin, jolla kuva otetaan sekä ohjelmiston käyttöliittymäelementit, kuvan tallennuskansion, ohjelmiston kameraobjekti, projektin nimi ja kameran sijaintitiedot. Skripti

käyttää Update-funktiota tarkastellakseen ehtolauseen avulla, milloin kuvakaappaus painiketta painetaan. Mikäli näppäintä painetaan, saa projektin nimen tallentava muuttuja itselleen arvon käyttöliittymän ylävalikon tekstiobjektista. Tämän perusteella määritellään arvo tallennuskansiolle. Skripti tarkistaa löytyykö tiedostopolku tallennuskansiolle. Jos tiedostopolkua ei löydy, skripti luo tarvittavat kansiot. Tarkistuksen jälkeen skripti kutsuu kuvankaappauksen suorittavaa funktiota.

Kuvakaappaus-funktio piilottaa ennen itse kaappausta kaikki käyttöliittymäelementit ja tallentaa tämän jälkeen kameran sijaintitiedot. Kuvakaappaus tallennetaan projektikansioon png-muodossa ja sen nimeämiskäytäntö on kuvanottohetki lisättynä ”Screenshot”-sanaan. Kameransijaintitiedot tallennetaan json-tiedostomuodossa kuvankaappauksen rinnalle projektin nimellä. Kun kuvalistasta painetaan kuvaa, se kutsuu skriptiä, joka hakee kameransijaintitiedot json-tiedostosta projektin nimen avulla ja muuttaa kameran sijainnin haetuiksi.

Ohjelmakoodi 12 Funktiot kuvakaappaukseen sekä json-tiedoston tallennukseen.

```
public IEnumerator CaptureScreen()
{
    yield return null;

    // Piilotetaan UI elementit
    ui.GetComponent<Canvas>().enabled = false;

    yield return new WaitForEndOfFrame();

    // Tallennetaan kameran sijaintitiedot kuvanotto hetkeltä.
    cameraPosition = cam.transform.position;
    cameraRotation = cam.transform.eulerAngles;

    // Kuvan tallennus
    string screenshotName = "Screenshot_" + System.DateTime.Now.ToString("dd-MM-yyyy-HH-mm-ss");
    ScreenCapture.CaptureScreenshot(System.IO.Path.Combine(folderPath, screenshotName + ".png"));

    // Kameratietojen tallennus olioon.
    data.cameraposition = cameraPosition;
    data.camerarotation = cameraRotation;

    // Luodaan tiedosto jossa kameratiedot.
    SaveIntoJson(screenshotName);

    ui.GetComponent<Canvas>().enabled = true;
}

public void SaveIntoJson(string name)
{
    // Luo json tiedoston, joka sisältää kameran sijaintitiedot.

    string jsonLocation = folderPath + "/" + name + ".json";

    string camerapositioninfo = JsonUtility.ToJson(data);
    System.IO.File.WriteAllText(jsonLocation, camerapositioninfo);
}
```

5.2.7 Objektin lisäys ja -hallinta

Viimeisenä ominaisuutena ohjelmistoon lisättiin vielä mahdollisuus objektien lisäykseen. Käyttöliittymään luotiin kuvakulmien listausta vastaava ikkuna. Kuvat listannutta prefabia käytettiin uudelleen, mutta siihen lisättiin tällä kertaa lisättävien objektien kuvat. Koska lisättävien objektien määrä oli vakio, tällä kertaa ei tarvinnut käyttää aiemmin mainittua ”PopulateGrid”-funktiota. Objektien lisäystä varten tehtiin ”SpawnObject”-skripti (ohjelmakoodi 13), jonka tarkoituksena oli lisätä objekti valittuun pisteeseen.

Skripti käyttää selvityksessä säteensuuntausta eli raycastingiä. Raycast lähettää ikään kuin näkymättömän säteen suoraan lähtökohdasta, kunnes se osuu collideriin. Tämä palauttaa tietoa osutusta pisteestä RaycastHit-objektissa (Unity Technologies, n.d.-b). Klikkauksen sijainti selvitetään käyttämällä kameraan liitettyä ScreenPointToRay-funktiota, jolle annetaan parametrina hiiren positio. Tästä saatu paikkatieto lähetetään Pointcloud Viewer and Tools -kirjaston funktiolle, joka tarkistaa osutun pisteen. Tämän jälkeen valittuun pisteeseen lisätään uusi listasta valittu peliobjekti.

Ohjelmakoodi 13 SpawnObject-skripti.

```
public GameObject prefab;
public PointCloudManager pointCloudManager;
public float rayOffsetFromNearClip = 0;
public GameObject folderForObjects;
Camera cam;
Vector3 mousePos;
Vector3 objectPos;

public void OnCall()
{
    cam = Camera.main;

    // Subscribe to event listener
    PointCloudManager.PointWasSelected -= InstantiateObject; // unsubscribe just in case
    PointCloudManager.PointWasSelected += InstantiateObject;
}


// Update is called once per frame
void Update()
{
    // dont do pick if over UI
    if (EventSystem.current != null && EventSystem.current.IsPointerOverGameObject()) return;

    if (Input.GetMouseButtonDown(0))
    {
        Ray ray = cam.ScreenPointToRay(Input.mousePosition);

        ray.origin = ray.GetPoint(cam.nearClipPlane + rayOffsetFromNearClip);

        pointCloudManager.RunPointPickingThread(ray);

    }
}

void InstantiateObject(Vector3 pos)
{
    var go = Instantiate(prefab, pos, Quaternion.identity) as GameObject;
    go.transform.parent = folderForObjects.transform;
}
public void OnDestroy()
{
    // unsubscribe
    PointCloudManager.PointWasSelected -= InstantiateObject;
}
```

Objekteihin lisättiin mahdollisuus muuttaa niiden väriä, kääntää niitä sekä muuttaa niiden kokoa. Objekti valitaan klikkaamalla, jonka jälkeen joko liukusäätimen arvoa vaihtamalla voi muuttaa sen asentoa. Pitämällä objektin kääntöön luotuja apuobjekteja painettuna ja liikuttamalla hiirtä samaan aikaan, voi sen asentoa myös muuttaa. Liukusäätimellä on mahdollista muuttaa myös objektin kokoa. Väriarvoja voi muuttaa värinvalintatyökalulla, joka tulee valinnan jälkeen näkyviin. Värinvalintatyökaluna hyödynnettiin ”HSVPicker”-nimistä kirjastoa, joka on saatavilla Unity Asset Storesta. Objektin valinta tehdään raycastiä käyttäen, ja vertaamalla osutun colliderin tag-arvoa.

Lisättyjen objektien kokoa muuttavien liukusäätimien toiminta perustuu yksinkertaiseen skriptiin. Yksi liukusäädin vastaa objektin yhtä sivua. Liukusäätimellä on skriptissä tarkkailija, joka muuttaa objektin kokoa sivulta, jonka liukusäätimen asentoa muutettiin. Skripti hakee objektin alkuperäisen koon ja muuttaa valitun sivun kokoa liukusäätimen arvon mukaan (ohjelmakoodi 14).

Ohjelmakoodi 14 Objektin koon muutos liukusäätimellä.

```
modelSizeSlider.onValueChanged.AddListener(value =>
{
    currentObject.transform.localScale = new Vector3(value, currentObject.transform.localScale.y, currentObject.transform.localScale.z);
    currentObject.transform.localScale = new Vector3(currentObject.transform.localScale.x, value, currentObject.transform.localScale.z);
    currentObject.transform.localScale = new Vector3(currentObject.transform.localScale.x, currentObject.transform.localScale.y, value);
});
```

Itse objektin kääntöskriptin toiminta perustuu hiiren painamisen tarkistukseen (ohjelmakoodi 15). Kun hiiren ensimmäinen painike on pohjassa, muuttujan ”isRotating”-arvo muuttuu todeksi, ilmoittaen että objektia halutaan kääntää. Tämän jälkeen hiiren nollapiste eli ensisijainti tallennetaan muuttujaan ja kun hiirtä liikutetaan, tallennetaan sen uusi sijainti ”Update”-funktion avulla toiseen muuttujaan. Näiden erotuksella saadaan laskettua hiiren liikkuma matka. Jotta objektin käännöstä voidaan tehdä sulavampi, kerrotaan erotus vielä hyväksi todetulla herkkyysarvolla. Tämän jälkeen objektin kiertoarvot korvataan erotuksesta saaduilla. Objektin kääntö lopetetaan, kun hiiren painike päästetään ylös. Objektia voi myös kääntää liukusäätimen avulla, sen toiminta on samankaltainen kuin objektin koon muuttamisessa (ohjelmakoodi 16).

Ohjelmakoodi 15 Objektin kääntämisen vaiheet ohjelmakoodissa.

```
void Update()
{
    colorpicker = GameObject.FindWithTag("colorpicker");
    if (_isRotating)
    {
        // hiiren liikkeiden erotus
        _mouseOffset = (Input.mousePosition - _mouseReference);

        // aseta kääntö
        _rotation.y = -(_mouseOffset.x + _mouseOffset.y) * _sensitivity;

        // käännä
        cube.transform.Rotate(_rotation);

        // hiiren sijainnin tallennus
        _mouseReference = Input.mousePosition;
    }
}

void OnMouseDown()
{
    if (colorpicker != null)
    {
        colorpicker.SetActive(false);
    }
    // objektia käännetään
    _isRotating = true;
    // hiiren alkusijainti
    _mouseReference = Input.mousePosition;
}
```

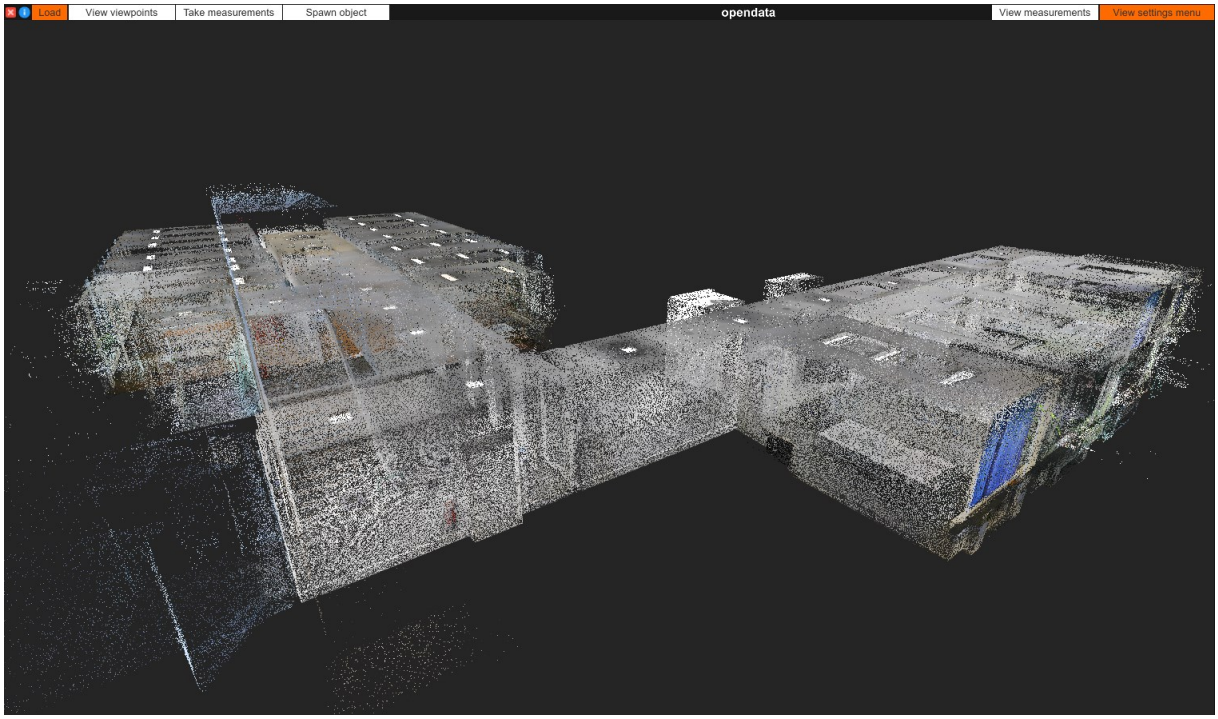
Ohjelmakoodi 16 Objektin kääntö liukusäätimellä.

```
rotateModelXSlider.onValueChanged.AddListener(value =>
{
    Debug.Log("Rotate slider value " + value);
    float delta = value - this.previousModelXValue;
    this.currentObject.transform.Rotate(Vector3.right * delta * 360);
    this.previousModelXValue = value;
});

rotateModelYSlider.onValueChanged.AddListener(value =>
{
    Debug.Log("Rotate slider value " + value);
    float delta = value - this.previousModelYValue;
    this.currentObject.transform.Rotate(Vector3.up * delta * 360);
    this.previousModelYValue = value;
});

rotateModelZSlider.onValueChanged.AddListener(value =>
{
    Debug.Log("Rotate slider value " + value);
    float delta = value - this.previousModelZValue;
    this.currentObject.transform.Rotate(0, 0, delta * 360);
    this.previousModelZValue = value;
});
```

5.3 Valmis prototyyppi

Valmis prototyyppi pystyy näyttämään pistepilvet sekä tekemään yksinkertaisia käsittelytehtäviä niiden parissa. Unityn tarjoamien työkalujen ansiosta VR-ominaisuus olisi ollut helppo lisätä prototyyppiin, mutta se todettiin liian raskaaksi ja jätettiin näin ollen pois valmiista prototyypistä. Prototyyppi on kuitenkin toimiva, mutta karsittu kokonaisuus ja osoittaa, että Unityä on mahdollista hyödyntää pistepilvityökaluna. Kuvassa 17 näkyy valmis prototyyppi, johon on tuotu NavVis-yrityksen sivuilta saatu avoin pistepilvitiedosto (NavVis, n.d.). Pistepilvitiedoston koko on 1.24Gt ja se sisältää 56 244 568 pistettä.

Kuva 17 Valmis prototyyppi.



6 Johtopäätökset ja pohdinta

Tutkimuskysymyksinä olivat, mitä olemassa olevia ratkaisuja pistepilvien visualisointiin on saatavilla ja kuinka hyvin ne täyttävät toimeksiantajan vaatimukset sekä voiko pistepilvien visualisointityökalun tehdä itse esimerkiksi Unity-pelimoottorilla. Työssä selvisi, että pistepilvien visualisointiin on saatavilla monia eri ohjelmistoja niin laitevalmistajien kuin myös kolmansien osapuolien toimesta. Toimeksiantajan minimivaatimuksena työkalulle oli, että se olisi helppokäyttöinen, siinä olisi mittaustyökalu ja sitä voisi katsella VR-lasien avulla. Nämä kaikki vaatimukset sekä enemmän täytti Faron Scene LT -ohjelmisto, se tukee Faron omien tiedostomuotojen lisäksi myös muita tiedostomuotoja ja siinä on kattavat ominaisuudet. Ohjelmistolla pystyy tekemään pistepilvestä jopa mesh-mallin.

Unity-prototyypin teko onnistui vaivattomasti ja suhteellisen nopeasti valmiin Pointcloud Viewer and Tools -kirjaston avulla. Mikäli kirjastoa ei olisi ollut, olisi prototyypin kehitysprosessiin kulunut paljon pidempi aika kuin mitä siihen nyt käytettiin. Prototyypin ominaisuudet ovat riittävät ja se osoittaa, että Unityllä on mahdollista tehdä varteenotettava työkalu. Etuna pelimoottorilla tehdyllä projektilla on, että sen voi räätälöidä juuri sellaiseksi kuin sen haluaa. Harmikseni kuitenkaan en saanut prototyyppiä täysin toimimaan VR-lasien avulla suorituskykyongelmien vuoksi.

Scene LT tarjoaa valmiin toimivan pistepilvienkäsittely ohjelmiston, joka toimii halutuilla ominaisuuksilla. Unity-prototyyppi on nimensä mukaisesti prototyyppi ja vaatii vielä paljon jatkokehitystä, jotta se voisi tuottaa yhtä laadukkaan kokemuksen pistepilvien katseluun kuin Scene. Prototyyppiä kehittämällä olisi mahdollista saada siitä konkreettisesti käyttöönotettava työkalu, mutta se ei kuitenkaan toistaiseksi vielä pärjää esimerkiksi Scenelle. Syynä on esimerkiksi se, että Scenestä voi käsitellyn aineiston viedä eri tiedostomuotoihin. Taloudellisesti ajatellen Unity-prototyypin hinta nousee Unity-lisenssimaksujen muodossa, sekä kehitykseen kuluvien palkkojen myötä. Toisaalta mikäli Sceneen haluaa luoda omia kirjastoja, nousee senkin hinta palkkojen takia.

Toimeksiantaja sai tehdyn työn avulla itselleen toimivan ratkaisun pistepilvien visualisointiin ja oli tyytyväinen työn tuloksiin. Lisäksi toimeksiantaja sai prototyypistä aihion, jota jatkokehittämällä pystyy varmasti tuottamaan vaatimuksia vastaavan työkalun.

7 Yhteenveto

Opinnäytetyön tekoprosessin aikana opin paljon hyödyllistä ja arvokasta tietoa niin pistepilvistä, niiden tuottamisesta kuin myös niiden visualisoinnista. Sain myös lisätietoa Unity-pelimoottorin toiminnasta sekä sitä hyödyntämällä tuottamaan itse toimivan prototyypin. Tutkimuskysymyksiin vastaaminen onnistui hyvin ja vaivattomasti.

Prototyyppiä kehittämällä siitä saisi varmasti varteenotettavan ratkaisun pistepilvien visualisointiin. Se ei kuitenkaan pärjää isojen yritysten tuottamille omille ohjelmistoille, sillä yritysten tuottamissa ohjelmistoissa on paljon enemmän ominaisuuksia kuten esimerkiksi käsitellyn aineiston vienti toiseen tiedostomuotoon. Prototyypin kehittäminen sai kiinnostukseni heräämään kehittämistyötä kohtaan ja opinnäytetyön taustalla oleva harjoittelu kasvatti osaltaan valmiuksia ja kiinnostusta projektityöskentelyyn.

Opinnäytetyö kasvatti myös omaa osaamista niin kirjoittamisesta kuin tieteellisen tekstin piirteistä ja vaatimuksista. Työskentely itsenäisesti uuden asian parissa vaati pitkäjänteisyyttä ja osaltaan lisäsi ammatillista itsevarmuutta. Kaiken kaikkiaan olen erittäin tyytyväinen opinnäytetyöhön niin prosessina kuin myös tuloksiin, jotka työn aikana saavutettiin. Opinnäytetyön aihe oli alusta asti itselleni mielenkiintoinen ja aihealueen parissa työskentely voisi olla varteenotettava vaihtoehto myös tulevaisuuden työelämässä.

Lähteet

Ahonen, P. (2015). *Saimaan ammattikorkeakoulu Tekniikka, Lappeenranta Rakennustekniikan koulutusohjelma Infratekniikka, maa-ja kalliotekniikka*. Saimaan ammattikorkeakoulu. http://urn.fi/URN:NBN:fi:amk-201503173287

Ammattikorkeakoulu, H. (n.d.). *Opinnäytetyöopas*. Hämeen ammattikorkeakoulu.

Archaeology Data Service. (n.d.). *Guides to Good Practice: LaserScan_3-1*. Retrieved January 29, 2021, from https://guides.archaeologydataservice.ac.uk/g2gp/LaserScan_3-1

Cronvall Timo, Kråknäs Pasi, & Turkka Tommi. (2012). *Laserkeilauksen käyttö liikennetunneleiden kunnossapidon hallinnassa* .

European Comission. (2018). *Police and first responder training enters mixed reality | Result In Brief | CORDIS | European Commission*. https://cordis.europa.eu/article/id/218536-police-and-first-responder-training-enters-mixed-reality

Geospatial World. (n.d.). *Trimble X7 3D laser scanner for simple, streamlined workflows*. Retrieved January 26, 2021, from https://www.geospatialworld.net/article/trimble-3d-laser-scanner/

Kokkonen, T. (2019). *Maalalaserkeilausaineiston visualisointi virtuaalitodellisuudessa*. https://aaltodoc.aalto.fi/bitstream/handle/123456789/40893/master_Kokkonen_Tia_2019.pdf?sequence=1&isAllowed=y

Lamberg, J. (2020). *Pelianalytiikan sisällyttäminen Unity-projektiin Unity Analyticsin avulla*.

Maanmittauslaitos. (n.d.). *Laserkeilausaineisto 0,5 p*. Retrieved January 26, 2021, from https://www.maanmittauslaitos.fi/kartat-ja-paikkatieto/asiantuntevalle-kayttajalle/tuotekuvaukset/laserkeilausaineisto-05-p

Miinalainen, K. (2019). *Pistepilven tehokas käsittely inventointimallinnusta varten*. http://urn.fi/URN:NBN:fi:amk-201904155046

NavVis. (n.d.). *Download Point Clouds*. Retrieved February 17, 2021, from https://www.navvis.com/m6-pointclouds

Nordic Geo Center Oy. (2011). *3D -laserskannaus forenssisessa rikospaikka/tapaturmatutkimuksessa | Todellisuutta tallentamassa*. http://www.geocenter.fi/blogi/3d-laserskannaus-forenssisessa-rikospaikkatapaturmatutkimuksessa/

Petrell, H. (2017). *Unity-pelimoottori ja peliprojekti*. Haaga-Helia ammattikorkeakoulu. http://urn.fi/URN:NBN:fi:amk-2017112718359

Point Cloud Library. (n.d.). *About*. Retrieved January 25, 2021, from

https://pointclouds.org/about/

Poliisi. (2020). *Rikoksen tutkinta*. https://poliisi.fi/rikoksen-tutkinta

ProDigiOUs. (2021). *Laserkeilauksen ja pistepilvien hyödyt*. https://prodigious.tamk.fi/laserkeilaus-ja-tietomallinnus-korjaushankkeissa/laserkeilauksen-ja-pistepilvien-hyodyt/

Santaluoto, O. (2012). *3D-skannaukseen perehtyminen*. Metropolia Ammattikorkeakoulu. http://urn.fi/URN:NBN:fi:amk-2012060111232

Unity Technologies. (n.d.-a). *Unity - Manual: Meshes, Materials, Shaders and Textures*. Retrieved February 4, 2021, from https://docs.unity3d.com/Manual/Shaders.html

Unity Technologies. (n.d.-b). *Unity - Manual: Rays from the Camera*. Retrieved February 15, 2021, from https://docs.unity3d.com/Manual/CameraRays.html

Vercator. (n.d.). *Common 3D point cloud file formats & solving interoperability issues*. Retrieved January 29, 2021, from https://info.vercator.com/blog/what-are-the-most-common-3d-point-cloud-file-formats-and-how-to-solve-interoperability-issues

W3Schools. (n.d.). *C# Tutorial (C Sharp)*. Retrieved February 17, 2021, from https://www.w3schools.com/cs/

Wiki, C. C. (n.d.). *FILE I/O - CloudCompareWiki*. Retrieved January 28, 2021, from https://www.cloudcompare.org/doc/wiki/index.php?title=FILE_I/O

Liite 1: Aineistonhallintasuunnitelma

Tämä opinnäytetyön tekemisessä ei ole ollut tarpeen käsitellä luottamuksellisia tietoja kuten esimerkiksi henkilötietoja tai muuta salassa pidettävää tietoa. Opinnäytetyö ei ole sisältänyt aineistoa, jota olisi tarpeen erikseen säilyttää tai tuhota. Opinnäytetyössä tuotettu prototyyppi on toimeksiantajan ja opinnäytetyön tekijän hallussa.

Opinnäytetyö sisältää kuvia prototyypistä ja sen ominaisuuksista. Kuvien hyödyntämiseen opinnäytetyössä on saatu toimeksiantajan suostumus. Työskentelyn tukena on käytetty myös harjoittelun aikana tuotettua päiväkirjaa, joka on sekin kirjoitettu ilman arkaluontoista tietoa ja sen säilytys on asianmukaista.