Riku Rönkä

IFC-MALLIN TUONNIN AUTOMATISOINTI UNITY-PELIMOOTTORIIN

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittely

2020

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä	Riku Rönkä
Työn nimi	IFC-mallin tuonnin automatisointi Unity-pelimoottoriin
Toimeksiantaja	Xamkin oppivan rakentamisen hanke
Vuosi	2020
Sivut	37 sivua
Työn ohjaaja	Jukka Selin

TIIVISTELMÄ

Opinnäytetyön tavoitteena on testata ja perehtyä erilaisiin IFC-tiedostojen tuontityökaluihin ja laajennuksiin koskien Unity-pelimoottoria, sekä saada ymmärrys rakennustietomallinnuksen hyödyistä. Opinnäytetyö tehdään Xamkin oppivan rakentamisen hankkeelle. Työn tulokset ovat merkityksellisiä, koska hankkeen kaikki mobiilisovellukset on tehty Unity-pelimoottorilla, joka ei tue natiivisti IFC-tiedostoja.

Opinnäytetyöraportin alussa on teoriaa, jossa kerrotaan rakennustietomallinnuksen perusperiaatteista ja sen tuottamista hyödyistä kokonaisuudessaan. Teoriaosuuden loppuosassa selostetaan, mikä on IFC ja mihin kaikkeen sitä pystytään käyttämään.

Teoriaosuuden jälkeen alkaa Tridify BIM Tools:in esittely. Luvun alkuosassa käydään läpi kokeiluversiot, tuotteen hinnoittelu, 3D-mallin optimointi, sekä tuotteen tuominen Unityyn. Luvun loppuosassa tehdään pienimuotoinen demo pelimaailmassa, joka sisältää toimistorakennuksen tietomallin, missä pystyy liikkumaan ja tarkastelemaan mallin osia.

Tridify BIM Tools:in jälkeen esitellään Pixyz Software. Tässä luvussa kerrotaan tuotteen asentamisesta Unityyn, lisenssin aktivoimisesta, sekä mallin optimoinnista käyttäen ToolBoxia ja Rule Engineä.

Opinnäytetyöraportin loppuosassa pelimaailmassa oleva malli käännetään verkkoselaimelle ja Androidille. Tässä luvussa vertaillaan molempien käännösversioiden plussia ja miinuksia.

Lopuksi on päätäntö, jossa kerrotaan opinnäytetyön tavoitteesta, onnistumisista, kehittymisestä, sekä sitä tehdessä ilmenevissä haasteissa. Päätäntöosassa osassa myös vertaillaan tuotteita keskenään.

Asiasanat: 3D-malli, Unity, IFC, BIM, pelimoottori

South-Eastern Finland University of Applied Sciences

Degree	Bachelor of Business Administration
Author	Riku Rönkä
Thesis title	Automation of importing IFC models into the Unity game engine
Commissioned by	Xamk's project of learning construction
Time	November 2020
Pages	37 pages
Supervisor	Jukka Selin

ABSTRACT

The purpose of this bachelor's thesis was to test and become familiar with various tools that could import IFC files and extensions into Unity game engine as well as gaining an understanding of the benefits of Building Information Modeling. The thesis was done for the Xamk's project of learning construction. The results of the work are relevant, because all the mobile applications of the project have been made with the Unity game engine, which does not natively support IFC files.

The thesis started with a theory section that explained the basic principles of building data modeling and the overall benefits produced. The rest of the theory section described what IFC was and what it could be used for.

After the theory part, the introduction of Tridify BIM Tools began. The first part of the chapter covered trial versions, product pricing, 3D model optimization, and importing the product to Unity. The rest of the chapter introduced a small-scale demo of the game world that included an office building information model where moving around and viewing the parts of the model was possible.

After Tridify BIM Tools, Pixyz Software was introduced. This chapter described how to install the product on Unity, activate the license and optimize the model using ToolBox and Rule Engine. In the end a model in the game world was compiled for a web browser and Android. This chapter compared the pros and cons of both compiled versions.

Finally, the conclusion discussed the goal of the thesis, its successes, its development and the challenges that came up. The concluding part also compared the products with each other.

Keywords: 3D-model, Unity, IFC, BIM, game engine

South-Eastern Finland University of Applied Sciences

SISÄLLYS

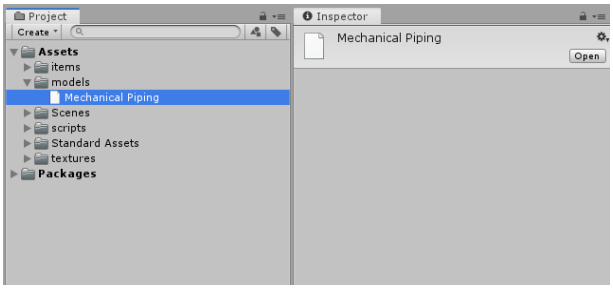
1 JOHDANTO

Tämän opinnäytetyön aiheena on testata ja perehtyä erilaisiin IFC-tiedostojen tuontityökaluihin ja laajennuksiin koskien Unity-pelimoottoria, sekä saada ymmärrys rakennustietomallinnuksen hyödyistä. Opinnäytetyö on tehty Xamkin oppivan rakentamisen hankkeelle. Hankkeen kaikki mobiilisovellukset on tehty Unity-pelimoottoria käyttäen. Unity ei tue natiivisti IFC-tiedostoja, joten työn tulokset ovat merkityksellisiä.

IFC-tiedostojen tuominen Unity-pelimoottoriin mahdollistaa rakennusmallien pelillistämisen. IFC-tiedostojen hyöty on suurimmillaan, kun rakennusmallista pystytään hakemaan tarkkoja tietoja eri kalusteista ja rakenteista.

Opinnäytetyöraportin toisessa luvussa käydään läpi rakennustietomallinnuksen perusperiaatteet ja kerrotaan IFC-tiedostojen hyödyistä. Luvussa vertaillaan rakennustietomallinnusta perinteisiin rakennusmenetelmiin.

Opinnäytetyöraportin kolmannessa luvussa käydään läpi kahden eri yrityksen IFC-tiedostojen tuontityökaluja Unity-pelimoottoriin. Luvussa perehdytään työkalujen antamiin 3D-mallien optimointimahdollisuuksiin ja toteutetaan pienimuotoinen demo, jossa pelaaja pysty tarkastelemaan mallia pelimaailmassa. Lopuksi käännetään 3D-rakennusmallista oleva demomaailma WebGL:lle, sekä Androidille ja arvioidaan hyötyjä, haittoja ja suorituskykyä.



Kuva 1. IFC-tiedosto Unityssä.

Unityn tuki IFC-tiedostoille ilman mitään laajennuksia on olematon. Ne näkyvät Unityssä vain pelkkinä tiedostoina (kuva 1), joille ei voi tehdä mitään eikä niissä näy mitään tietoja itse tiedostosta.

2 TIETOMALLINNUS

Rakennustietomallinnus on prosessien, tekniikan ja ihmisten yhdistämistä rakentamisen tulosten parantamiseksi. Rakennustietomallinnus on viimeisin rakennusteollisuuden kehitys, joka viittaa rakennuksen suunnittelu-, rakennus- ja käyttöprosessiin yhdessä käyttämällä yhtenäistä 3D-mallien järjestelmää erillisten suunnittelupiirustusten sijaan. Rakennustietomallinnus yhdistää ihmiset ja tekniikan tehostamaan ajankäyttöä ja kustannuksia, sekä parantamaan rakennusten esimerkiksi toimisto- ja asuinrakennuksien hyötysuhdetta. (Magicad s.a.)

Rakennustietomallinnus ei tarkoita vain ohjelmistosarjaa tai 3D-mallia, vaan se sisältää valtavan määrän tietoa projektista, sekä prosessin tietojen vaihtamiseksi muiden osapuolten kanssa. Aikaisemmat työnkulut perustuivat useisiin tiedostomuotoihin ja katkenneisiin prosesseihin, jotka jäivät heti jälkeen, kun muutoksia tehtiin. Rakennustietomallinnuksen työnkulut mahdollistavat paljon joustavamman lähestymistavan projektihallintaan. Rakennustietomallinnusta käytetään parantamaan rakennusprosessin tehokkuutta, vähentämään rakentamisen aikana syntyvää jätettä sekä parantamaan rakennusten laatua ja tehokkuutta. (Magicad s.a.)

Monet edistyneet teollisuudenalat ovat puolustaneet "Failing Fast" -periaatetta. Tämä tarkoittaa uusien asioiden kokeilemista sillä periaatteella, että ensimmäinen kerta merkitsisi epäonnistumista, joka johtaisi toiseen iterointiin ja lopulta menestykseen. Tämä konsepti toimii vain digitaalisessa maailmassa. Rakennuksen suunnittelu virtuaalisesti ennen fyysistä rakentamista on käytännössä on ainoa tapa "epäonnistua nopeasti". Ei ole mahdollista kokeilla rakentamisen eri vaihtoehtoja käytännössä. Tämä olisi aivan liiaan aikaavievää ja kallista. (Magicad s.a.)

Perinteistä rakennusmenetelmää käyttäen osa tiedoista menetetään edellisestä vaiheesta tiimien siirtyessä projektista toiseen. Rakennustietomallinnuksen avulla tiedot tallennetaan ja arkistoidaan digitaalisesti, jotta ne olisivat käytettävissä aina kun niitä tarvitaan. Tietomallin käyttöönotto perustuu jatkuvaan ja reaaliaikaiseen tiedonkulkuun. Jokainen rakennusprosessin vaihe on

toteutettu digitaalisesti. Tämä avaa uusia mahdollisuuksia parantaa tehokkuutta, tarkkuutta ja yhteistyötä rakennuksessa mukana olevien osapuolten välillä. Jokainen liiketoimintaprosessi digitalisoidaan ja jokainen päätös perustuu tietoon. Maailman talousfoorumi on julistanut tämän "neljänneksi teolliseksi vallankumoukseksi", jossa tietoja syöttävät koneet muuttavat eksponentiaalisesti prosesseja kaikilla teollisuudenaloilla. (Magicad s.a.)

2.1 Mikä on IFC

IFC-tietomallinnus on avoin standardisoitu rakennetusta ympäristöstä oleva digitaalinen kuvaus. IFC tulee sanoista Industry Foundation Classes. IFC-tiedostoja pystyy avaamaan erilaiset Building Information Modeling (BIM) ohjelmat, kuten esimerkiksi Adobe Acrobat DC, Autodesk AutoCAD Architecture ja GRAPHISOFT ArchiCAD. Sen tarkoituksena on olla käytettävissä monilla eri ohjelmistoalustoilla, sekä laitteilla. (Buildingsmart 2020.)

IFC:n metatietojen avulla pystytään kertomaan mallista monenlaisia eri ominaisuuksia. Esimerkkinä lattiasta voidaan kertoa siinä käytetty materiaali, väri ja maksimi lämmönsietokyky. Ilmanvaihtokoneesta voidaan kertoa sen valmistaja, sijainti rakennuksessa, sekä erilaiset toiminnot, joita sillä voi suorittaa. (Buildingsmart 2020.)

2.2 IFC:n käyttö

IFC:n käyttö perustuu osapuolten väliseen tietojen vaihtoon. Esimerkiksi arkkitehti voi toimittaa omistajalle mallin uudesta rakennussuunnitelmasta, kun taas omistaja pystyy laittamaan rakennusmallin urakoitsijalle ja sieltä takaisin omistajalle rakennusmalleineen, jossa on kuvattu valmistajan tiedot ja asennetut laitteet. IFC:n käyttö mahdollistaa myös rakennusvaiheen aikaisten tiedostojen arkistoimisen tietokokoelmana pitkäaikaiseen säilyttämiseen. (Buildingsmart 2020.)

Rakennusmalleissa olevat metatiedot pystytään viemään erilaisissa tiedonvaihto tiedostomuodoissa, kuten JSON ja XML. Tietoja pystytään viemään tai tuomaan verkkopalveluiden kautta tiedostoina. (Buildingsmart 2020.)

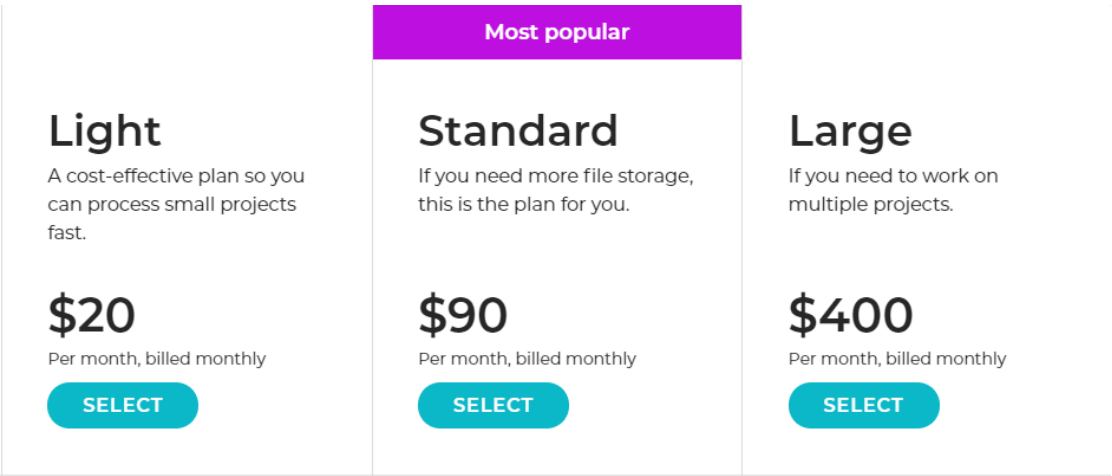
3 IFC-TIEDOSTOJEN TUONTI UNITY-PELIMOOTTORIIN

Pelimoottorit tukevat vain mesh-pohjaisia malleja eli polygon verkkoja. IFC-tiedosto on kurvipohjainen, ei polygon verkko. Syy siihen, että pelimoottorit tukevat lähtökohtaisesti vain mesh-pohjaisia malleja on teho kysymys. Jos 3D-malli olisi pelimoottorissa käyrämallina eli matemaattisina käyrinä, pelimoottori ei pystyisi piirtämään 60 kuvaa sekunnissa. IFC-malli pitää pystyä muuttamaan mesh-pohjaiseen malliin pelimoottoriin tuomisen yhteydessä.

3.1 Tridify Bim Tools

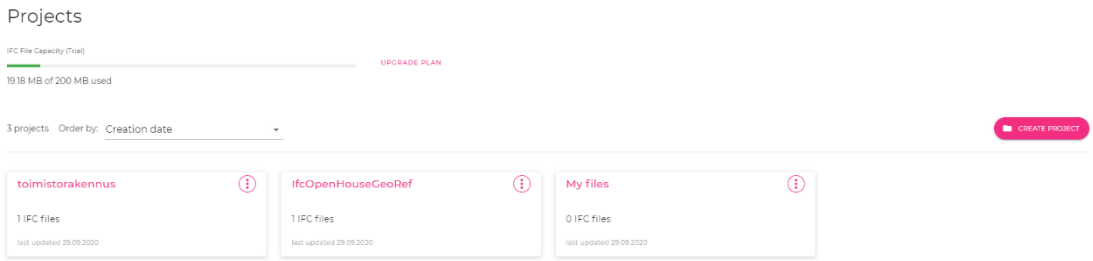
Tridify on helsinkiläinen ohjelmistoyritys, joka tuottaa rakennusarkkitehtuuriin liittyviä aputyökaluja, jotka helpottavat asiakkaiden töiden tuottavuutta ja luovuutta. Yritys on tehnyt Unityyn Tridify BIM Tools-nimisen työkalun, jonka avulla käyttäjät pystyvät tuomaan 3D-malleja Unityyn. (Tridify 2020.)

Tridify Bim Toolsin käyttö aloitetaan rekisteröitymällä Tridifyn sivustoon. Rekisteröitymisen jälkeen alkaa 14 päivän kokeiluaika. Kokeiluaikana tuotuja IFC-tiedostoja ei saa käyttää kaupallisiin tarkoituksiin.



Kuva 2. Tridifyn tarjoamat lisenssivaihtoehdot.

Kokeiluajan päätyttyä Tridify-tilillä olevia tiedostoja pystyy käyttämään kaupallisiin tarkoituksiin vasta sen jälkeen, kun on päivittänyt lisenssin (kuva 2). Kokeiluaika on sidottu käyttäjätiliin, joten luomalla uuden käyttäjätilin voi lisenssin päivitystarpeen ohittaa.

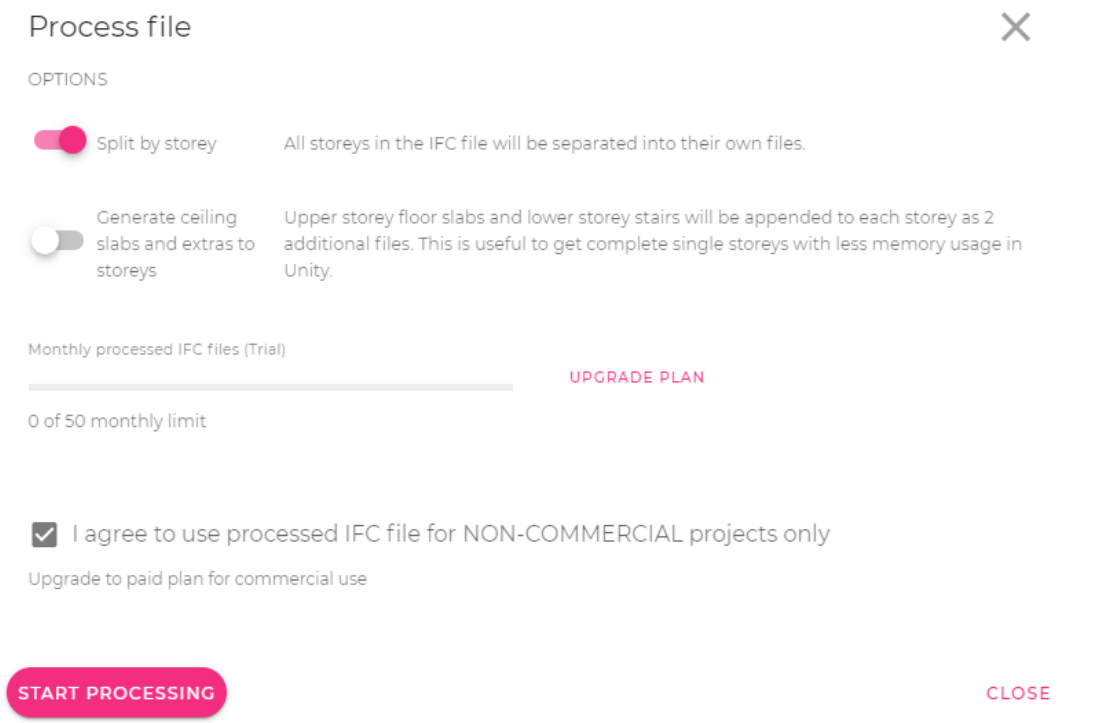


Kuva 3. Tridify-tilin projektit.

Oman Tridify-tilin pääsivulta löytyy ”Create Project”-nappi (kuva 3). Tätä painamalla projektille annetaan aluksi nimi. Tämän jälkeen projektiin pystyy lataamaan IFC-tiedoston koneelta drag’n’drop-tominnolla tai perinteisesti selaamalla tiedostoja.

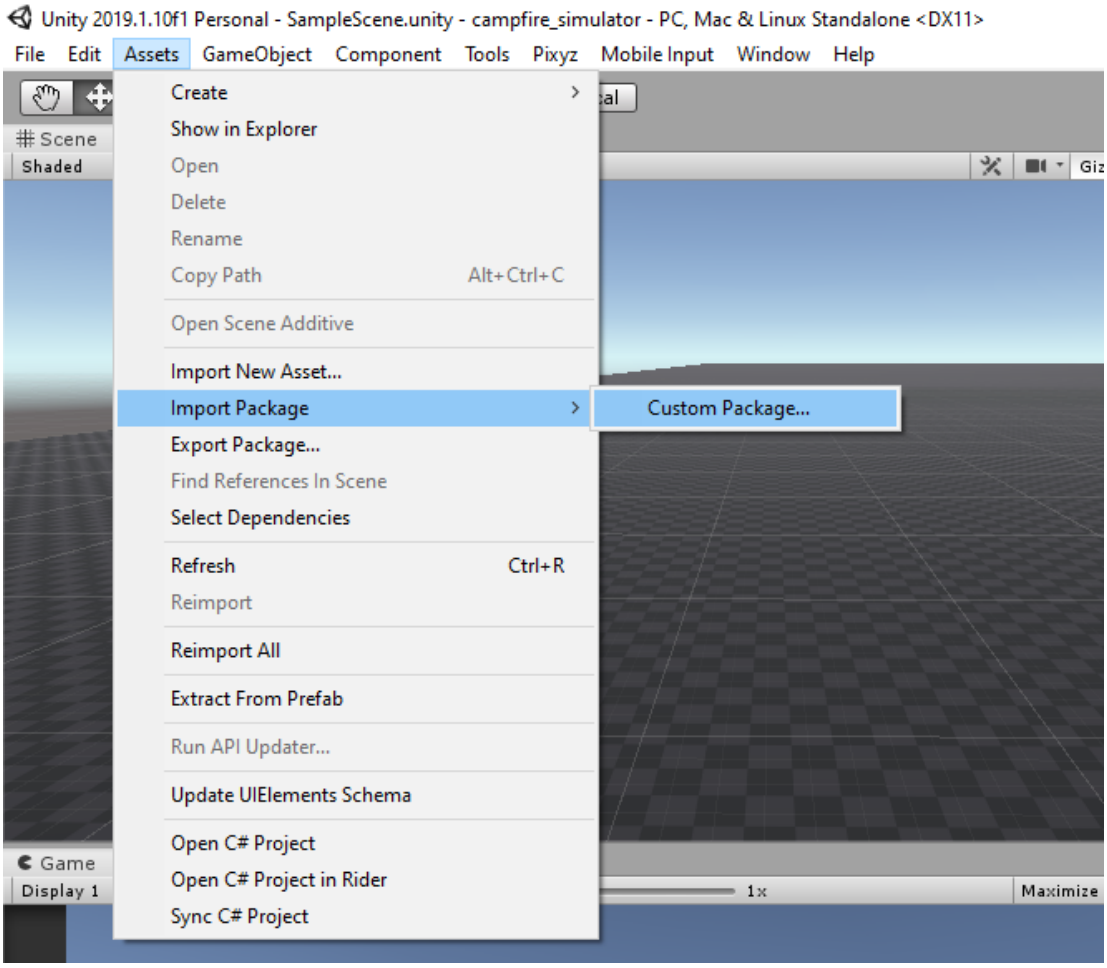
3.2 Tridify mallin optimointi

Tridify Bim Tools tarjoaa kaksi erilaista optimointivaihtoehtoa (kuva 4). Ensimmäinen vaihtoehto on ”Split by storey”, joka jakaa IFC-tiedostossa olevat kerrokset omiin tiedostoihinsa. Tämä toiminto selkeyttää projektissa olevien tiedostojen rakennetta.



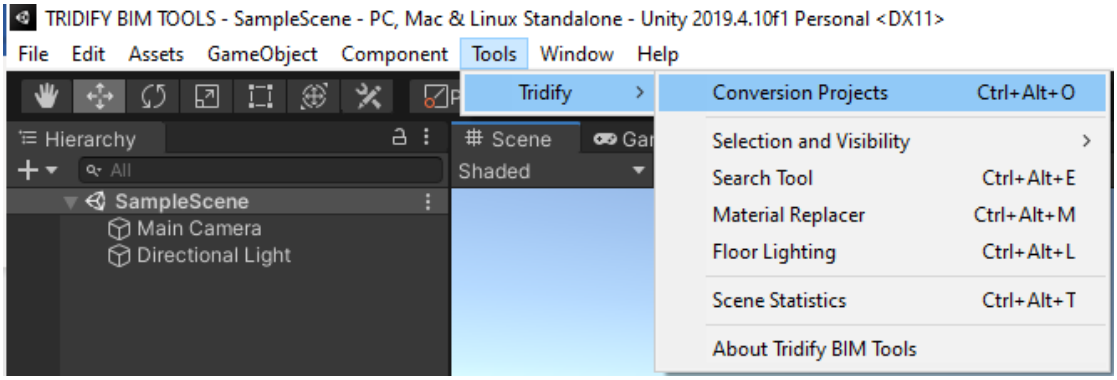
Kuva 4. Tridify Bim Tools optimointivaihtoehdot.

Toinen vaihtoehto on “Generate ceiling slabs and extras to storeys”. Tällä optimointivaihtoehdolla ylemmän kerroksen lattialaatat ja alakerroksen portaat liitetään jokaiseen kerrokseen kahtena ylimääräisenä tiedostona. Tässä hyötynä tulee se, että saadaan täydellisiä yksittäisiä kerroksia, joiden ansiosta Unityn muistinkäyttö vähenee huomattavasti, mutta haittapuolena tulee tiedostojen koon kasvaminen.



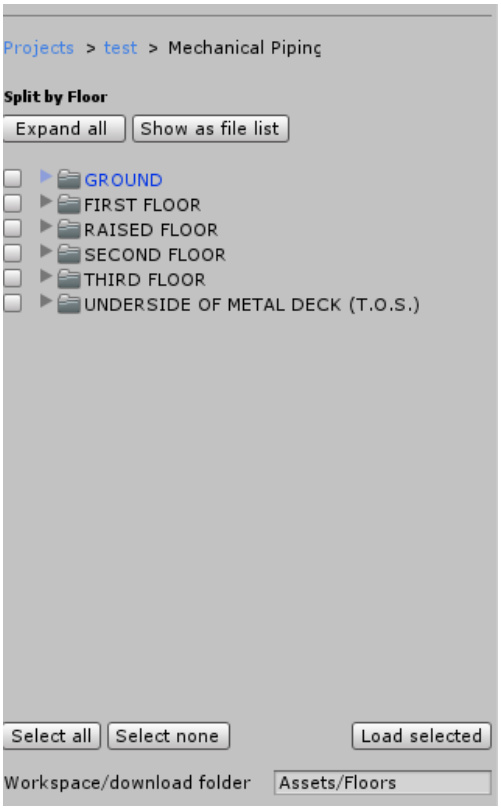
Kuva 5. Unitypackagen tuonti Unityyn

Tridify BIM Tools on unitypackage-tiedosto, joka tuodaan Unityyn Assets välilehden alta ja sieltä Import Package -> Custom Package (kuva 3).



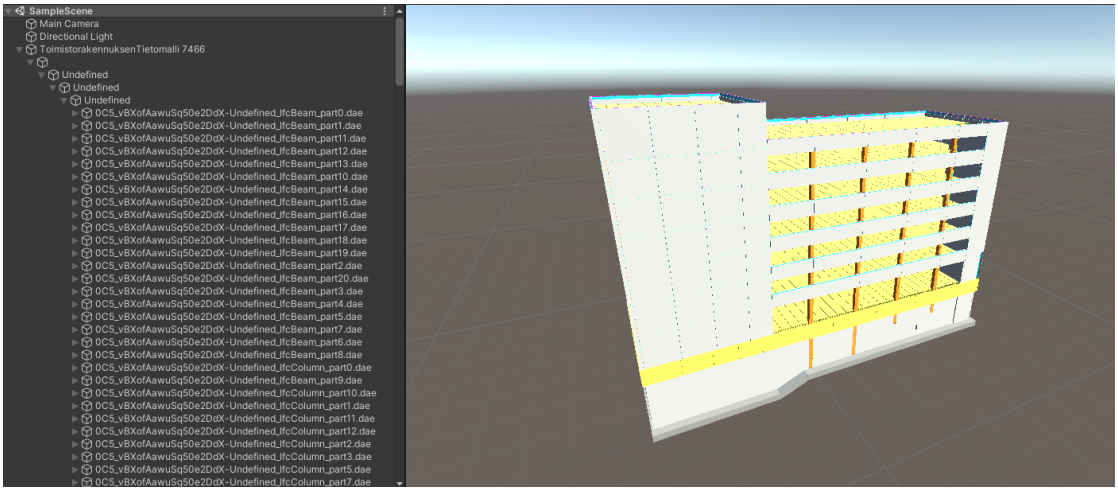
Kuva 6. Tridify-valikko

Tuonnin jälkeen yläpalkista Component- ja Window-valikkojen väliin ilmestyy Tools-valikko (kuva 4), jonka alta löytyy Tridify ja sieltä Conversion Projects. Tätä klikkaamalla avautuu Tridify-tilin kirjautumisikkuna. Kirjautumalla sisään näkee omaan tiliin ladatut ja konvertoidut 3D-mallit.



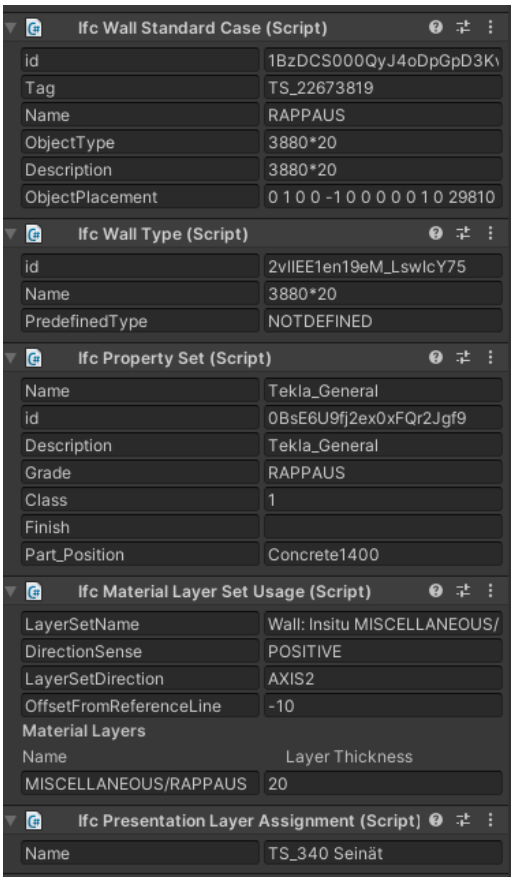
Kuva 7. IFC-tiedoston tasojen valinta.

Kaikki 3D-mallinnusohjelmassa malliin tehdyt tasot tulevat näkyviin ikkunaan (kuva 5). Mallista pystyy valitsemaan kaikki tasot tai vain osan mallissa olevista tasoista (kuva 5).



Kuva 8. IFC-malli Unityssä.

Malli ilmestyy tämän jälkeen skeneen (kuva 4) ja jokaisesta mallin osasta pystyy tarkastelemaan siihen kuuluvia metatietoja klikkaamalla sitä. Tuonnin jälkeen 19 megatavun IFC-mallista tuli 377 megatavun kokoinen (yhteenlasketuna DAE-tiedostojen koko). Tämän kokoisen tiedoston tuominen kesti Unityyn noin 10 minuuttia molemmat optimointivaihtoehdot valittuina (kuva 4).

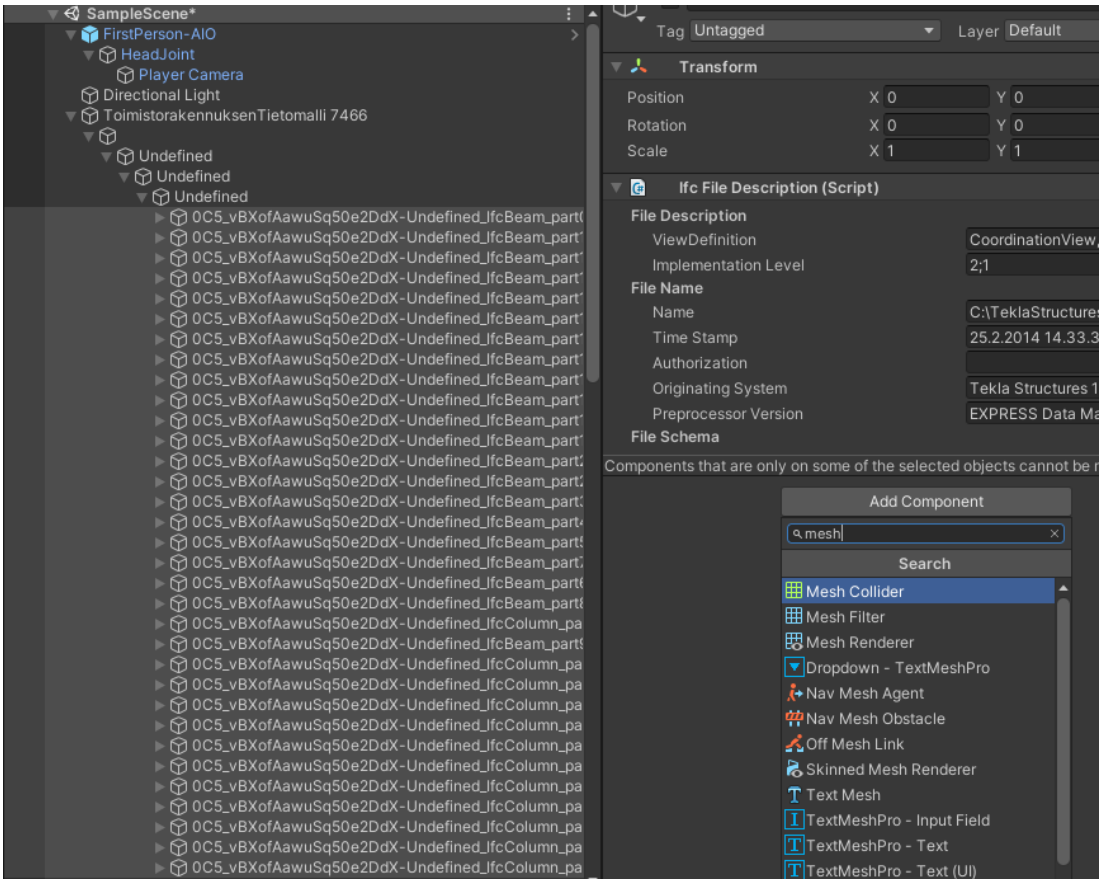


Kuva 9. Metatietoja seinästä.

Metatietoja ovat esim. nimi, tagi, id, objektityyppi (kuva 9). Tridify luo monta erilaista skriptiä perustuen siihen, minkä tason alla seinä on. Esimerkiksi seinä saa Ifc Material Layer Set Usage-skriptin, kun taas pylväs saa Ifc Column Type-skriptin, jossa kerrotaan tämän tyyppisistä materiaaleista enemmän.

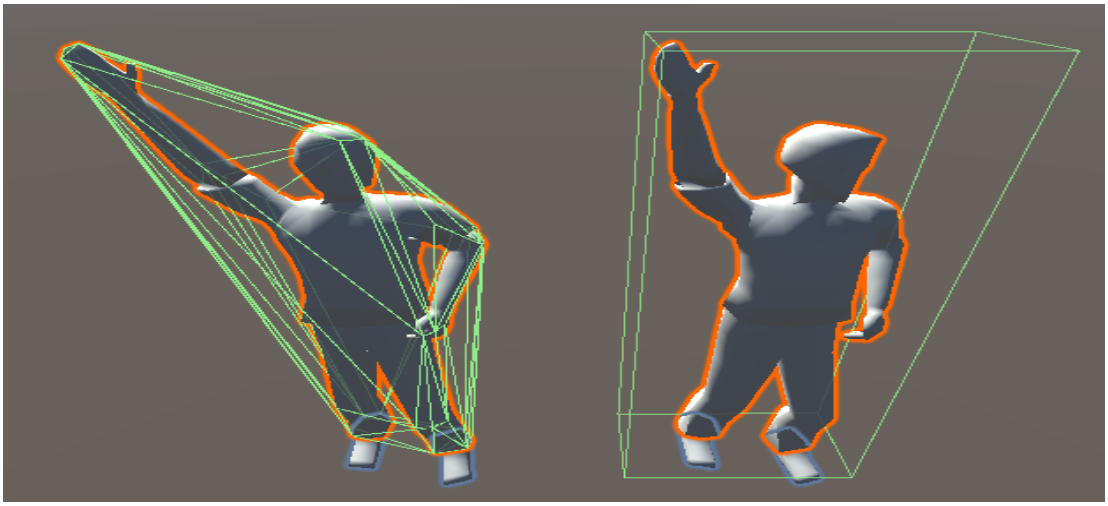
3.3 Törmäystunnistus mallissa

Lähdin testaamaan 3D-mallin törmäystunnistusta eri osien kanssa. Tulin siihen tulokseen, että Tridifyn bim tools ei tee 3D-mallien osille collidereita, joten ensimmäisenä mieleeni tuli, voisiko jokaiselle osalle laittaa koodin kautta colliderin. Colliderien lisääminen onnistui, mutta niiden sijainnit eivät olleet lähelläkään osien oikeita sijainteja. Seuraavana päätin ottaa jokaisen osan sijainnin talteen ja laittaa kyseisille collidereille nämä sijainnit. Unity ei pystynyt laittamaan 8565 eri osan sijainteja paikoilleen, joten se kaatui. Testasin seuraavaksi yhdellä 3D-mallin osalla, että onnistuisiko sen laittaminen kohdilleen. Ongelmaksi tuli, että en saanut colliderin sijaintia osumaan mallin osan kohdalleen, vaikka colliderin ja osan sijainnit ja koot olivat täsmälleen samat.



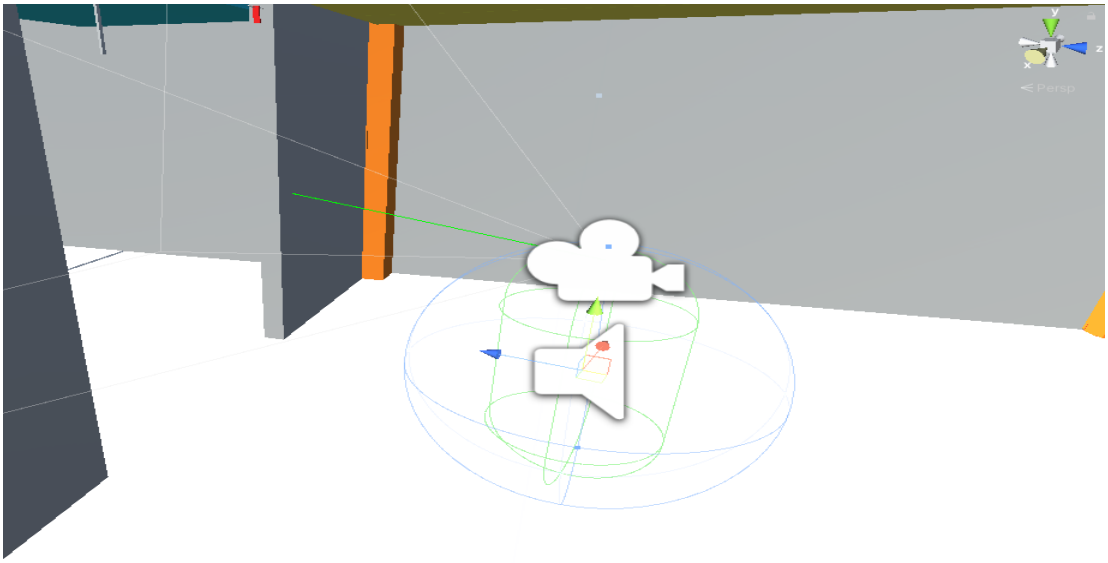
Kuva 10. Törmäystunnistuksen lisääminen malliin.

Seuraavana lähdin yrittämään toisenlaista ratkaisua valitsemalla kaikki malliin kuuluvat osat ja lisäämällä niille Mesh Collider-komponentin (kuva 10). Mesh Collider on järkevin valinta, jos tietää, että mallissa tulee olemaan muitakin kuin suorakaiteen muotoisia objekteja.



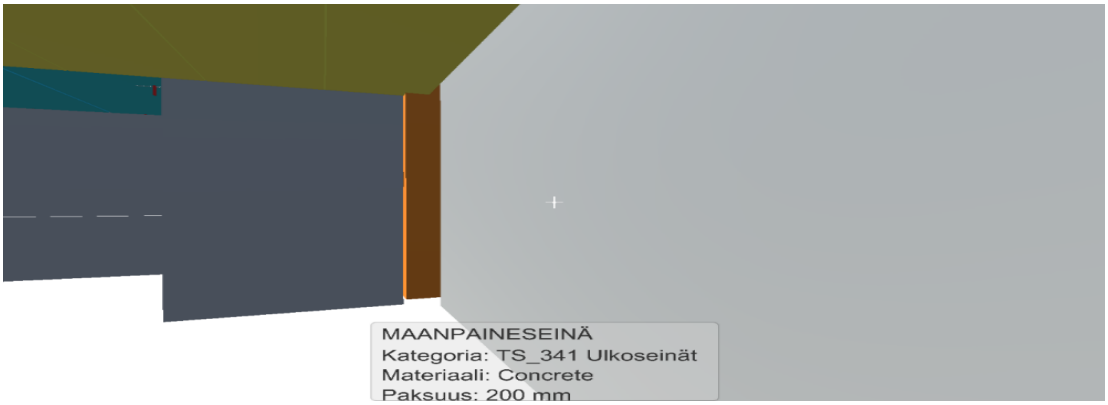
Kuva 11. Vasemmalla Mesh Collider ja oikealla Box Collider havainnollistettu vihreillä viivoilla.

Mesh Collider yrittää matkia mallin muotoja niin hyvin kuin se on suorituskyvyn kannalta mahdollista (kuva 11). Mesh Collider on Box Collideria paljon raskaampi pelimoottorille, koska siinä on enemmän laskemista törmäystunnistuksen osalta. Box Collider luo muotonsa vuoksi mallille suuremman törmäystunnistuksen, kun mitä itse malli on, mikäli malli ei ole suorakaiteen muotoinen.



Kuva 12. First Person Controller, jolla käyttäjä liikkuu mallissa.

Vihreän suoran viivan avulla (kuva 12) tapahtuu törmäystunnistus. Unityssä on monenlaisia törmäystunnisteita. Niistä yleisimmät ovat, box, sphere, capsule ja mesh. Pelaajan törmäystunnistus tapahtuu yleensä (”Capsule Collider”) -törmäystunnistuksella. Yksi hyöty kapselitörmäystunnistuksella verrattuna esimerkiksi laatikkotörmäystunnistukseen on se, että pelaajan noustessa portaita tai hyppäämällä tasanteelta toiselle pelaajan riski pysähtyä tai tai jäädä kiinni paikoilleen on paljon pienempi.



Kuva 13. Tietoja seinästä

Kun pelaaja osoittaa seinää tai lattiaa ja painaa hiiren vasenta painiketta, tulee tässä mallissa näkyviin nimi, kategoria, materiaali ja paksuus (kuva 13). Seinästä löytyy paljon muitakin tietoja (kuva 9), mutta nämä ovat oleellisemmat.

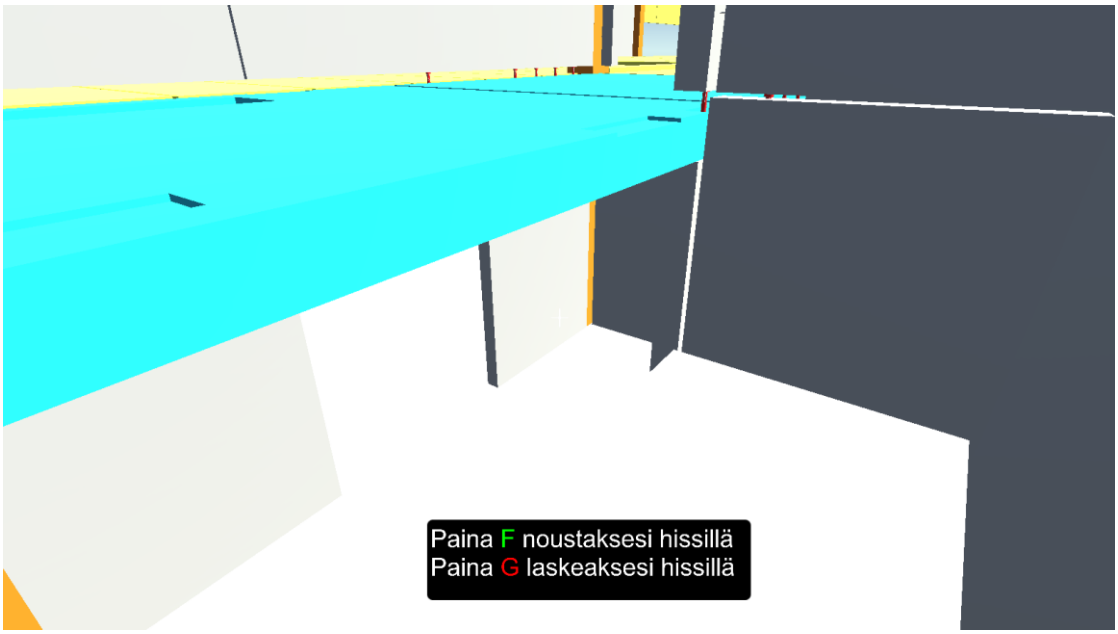
```
Vector3 fwd = fpscontroller.transform.TransformDirection(Vector3.forward);
Debug.DrawRay( start: fpscontroller.transform.position,  dir: fwd * 2, Color.green);
if (Physics.Raycast( origin: fpscontroller.transform.position,  direction: fwd, out RaycastHit col, maxDistance: 4 ))
{
    if (Input.GetMouseButtonDown(0))
    {
        tiedot_panel.SetActive(true);

        nimi.text = col.collider.name;
        materiaali_paksuus.text = "";
        kategoria.text = "";
        materiaali.text = "";

        if (col.collider.GetComponent<IfcMaterialLayerSetUsage>())
        {
            materiaali_paksuus.text = $"Paksuus: {col.collider.GetComponent<IfcMaterialLayerSetUsage>().MaterialLayers[0].LayerThickness} mm";
        }
        if (col.collider.GetComponent<IfcPresentationLayerAssignment>())
        {
            kategoria.text = $"Kategoria: {col.collider.GetComponent<IfcPresentationLayerAssignment>().Name}";
        }
        if (col.collider.GetComponent<IfcPropertySet>())
        {
            var str :string  = Regex.Replace( input: col.collider.GetComponent<IfcPropertySet>().Attributes[6].Value,  pattern: "[^A-Za-z]+",  replacement: "");
            materiaali.text = $"Materiaali: {str}";
        }
    }
}
```

Kuva 14. Törmäystunnistus ja tietojen näyttäminen ohjelmakoodina.

Tämä on toteutettu siten, että aluksi otetaan suunta talteen muuttujaan. Sen jälkeen testataan, osuuko suunnan raycast-säde mihinkään. Jos osuu ja käyttäjä painaa hiiren vasenta painiketta tiedot-paneeli näytetään. Kaikki tekstit nollataan aluksi ja tarkastetaan, löytyykö objektista, johon törmättiin Ifc-skripti. Jos löytyy, haetaan skriptistä siihen liittyvä tieto ja näytetään se paneelissa.



Kuva 15. Hissi mallissa.

Testaamassani mallissa oli monta kerrosta, koska se oli toimistorakennus. Tein tästä syystä malliin hissin, jonka avulla pystyy liikkumaan eri kerrosten välillä.

```
Event function
private void OnTriggerEnter(Collider other)
{
    if (other.name.Equals("FirstPerson-AIO"))
    {
        panel.SetActive(true);
        isColliding = true;
    }
}

Event function
private void OnTriggerExit(Collider other)
{
    if (other.name.Equals("FirstPerson-AIO"))
    {
        panel.SetActive(false);
        isColliding = false;
    }
}
```

Kuva 16. Hissiin menemisen ja hissistä poistumisen tunnistus ohjelmakoodilla.

Käytin törmäystunnistukseen Unityn sisäänrakennettuja metodeja: OnTriggerEnter ja OnTriggerExit. Molemmat metodit ottavat parametriksi törmäävän asian. Jos törmäävä henkilö on pelaaja, näkyy info teksti (kuva 15) ja isColliding-muuttuja vaihtuu trueksi.

```
void Update()
{
    if (isColliding)
    {
        if (Input.GetKey(KeyCode.F))
        {
            _hissi.position += new Vector3( x: 0, y: 1.25f * Time.deltaTime,  z: 0);
        }
        if (Input.GetKey(KeyCode.G))
        {
            _hissi.position -= new Vector3( x: 0, y: 1.25f * Time.deltaTime,  z: 0);
        }
    }
```

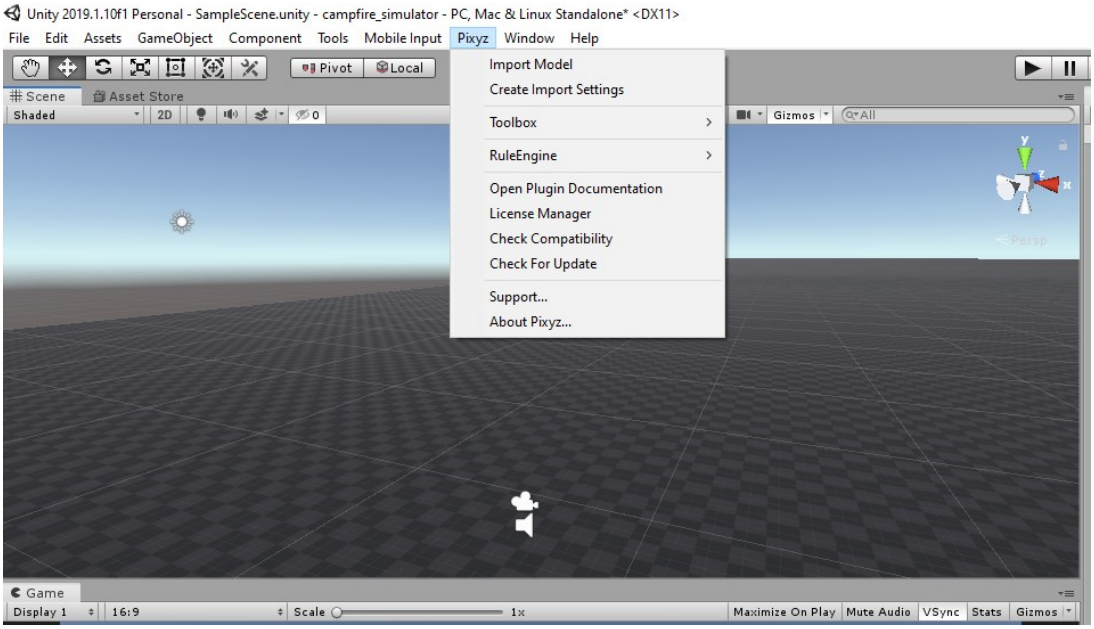
Kuva 17. Hissin ohjelmakoodi.

Kun isColliding-muuttuja on true, pelin jokaisella silmukalla tarkastetaan painaako pelaaja F-näppäintä. Jos painaa, niin lisätään hissin y-sijaintia. Jos pelaaja painaa G-näppäintä, vähennetään hissin y-sijaintia (kuva 17).

3.4 Pixyz software

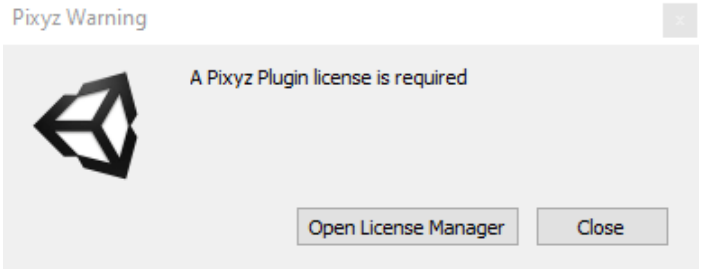
Pixyz on ranskalainen yritys, joka auttaa teollisuusyrityksiä ja 3D-kuluttajia tarjoamalla Pixyz studio, -plugin, -review ja -batch-työkalut. Näiden työkalujen tarkoituksena on säästää aikaa ja vaivaa, sekä maksimoida visualisoinnin suorituskyvyn. Tässä luvussa tullaan käsittelemään plugin-työkalua. (Pixyz s.a)

Pixyz tarjoaa (”Pixyz Plugin for Unity”) nimisen unitypackagen. Laajennuksessa on 7 päivän kokeiluversio. Pixyz Plugin for Unity tuodaan Unityyn samalla tavalla kuin muutkin unitypackaget (kuva 5, s. 10). Assets valikon alta löytyy Import Package ja sen alta Custom Package.



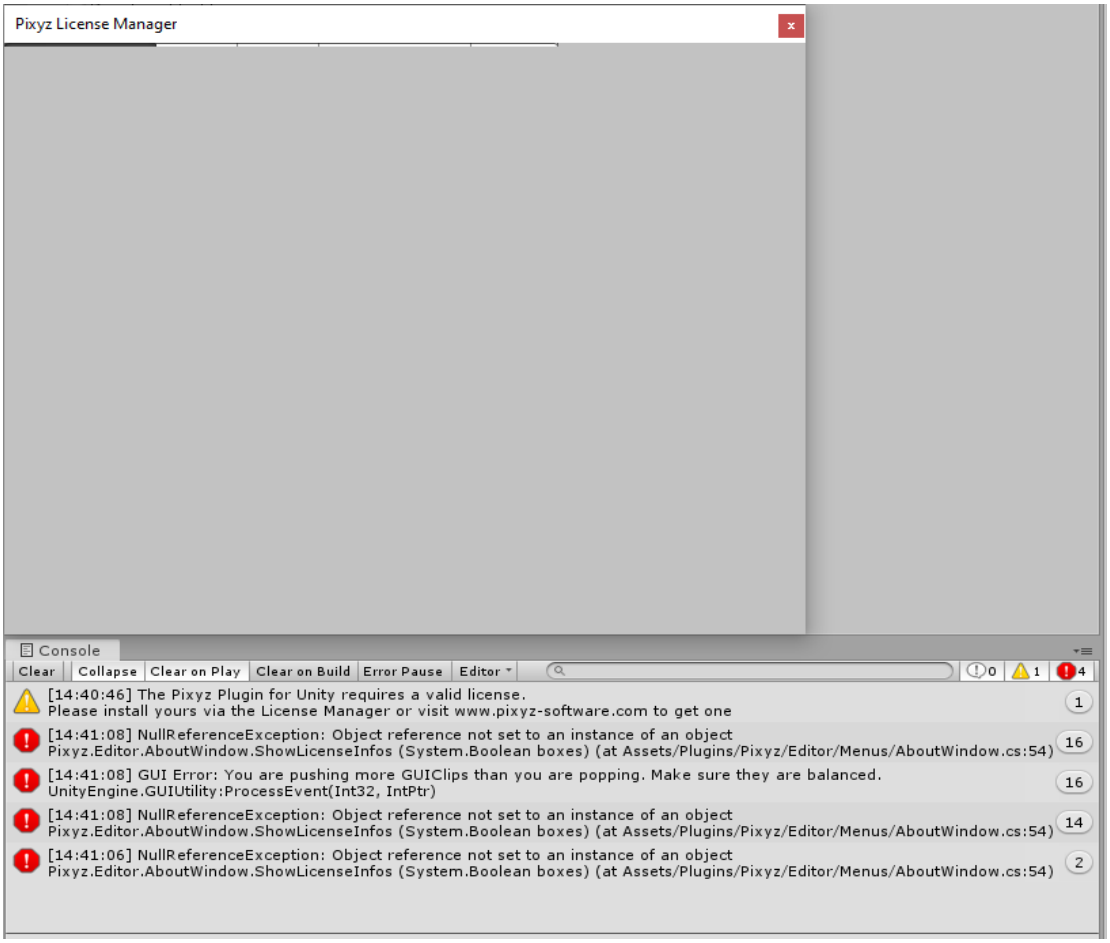
Kuva 18. Pixyz-valikko.

Pluginin asennuksen jälkeen Unityn yläpalkkiin ilmestyy ”Pixyz”-valikko (kuva 18), jonka alta löytyy ”Import Model”. Tätä klikkaamalla pystyy tuomaan mallin Unityyn.



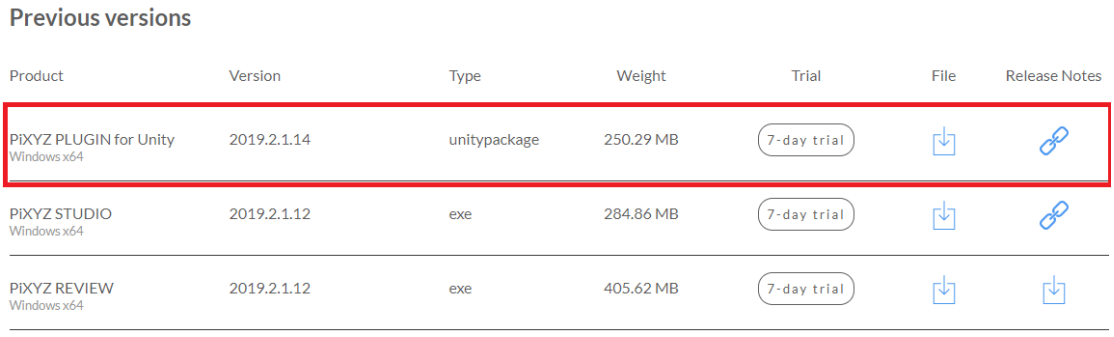
Kuva 19. Pixyz lisenssi-ilmoitus.

Tätä ennen kuitenkin Pixyz kysyy käyttäjältä lisenssiä tuotteen käyttöön (kuva 19). Pixyz License Manager-ikkuna ei näytä toimivan toivotulla tavalla. Ikkunan yläkulmassa näkyy erittäin ohuita nappeja, joiden painaminen ei tee mitään. Konsoliin tulee suuri määrä virheitä eikä tuotteen lisenssiä pysty aktivoimaan, koska Pixyz License Manager on tyhjä. Ikkunaa klikkaamalla konsoliin tulee lisää virheitä.



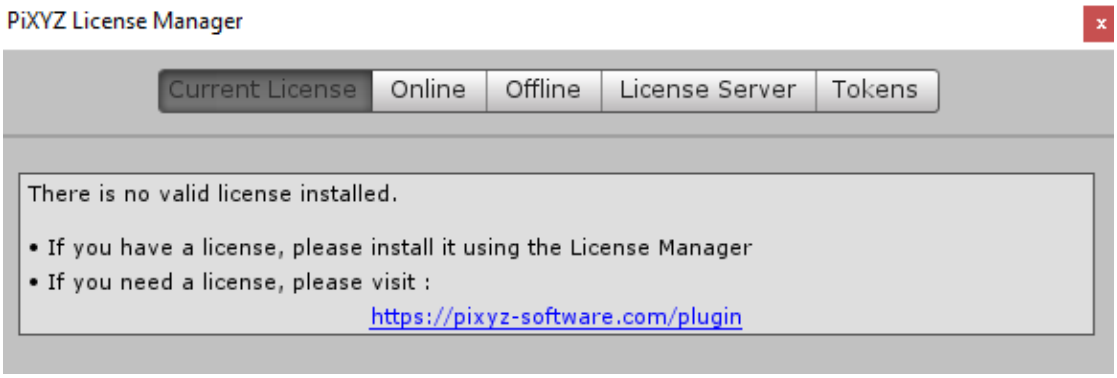
Kuva 20. Pixyz License Manager virheiden kanssa.

Kokeilin poistaa unitypackagen ja asentaa sen uudelleen. Sama lopputulos tuli näiden toimintojen jälkeen. Seuraavaksi tein aivan uuden Unity projektin ja yritin tuoda pluginin sinne ja aktivoida lisenssiä uudelleen. Pixyz License Manager-ikkuna oli vieläkin tyhjä ja konsoli täynnä virheitä (kuva 20). Menin Pixyzin verkkosivuille ja löysin sieltä vanhemman version pluginista.



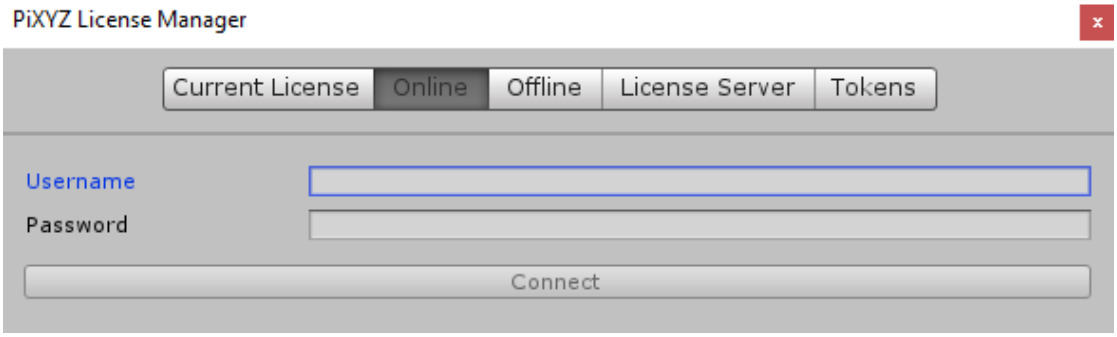
Kuva 21. Pixyz edelliset versiot.

Asensin vanhemman version (kuva 21) pluginista ja se näytti toimivan hyvin.



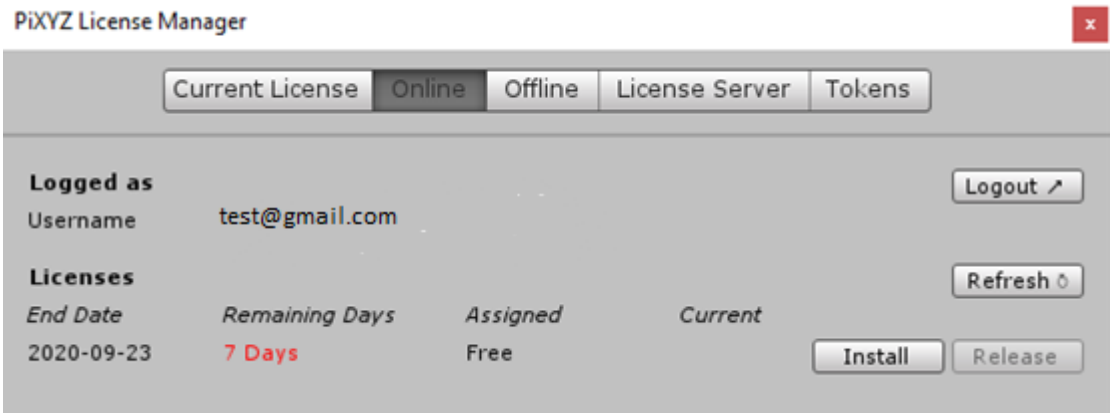
Kuva 22. Pixyz License Manager ikkuna.

PiXYZ License Manager-ikkunan aukaisun jälkeen se ilmoittaa, että lisenssiä ei ole asennettuna (kuva 22).



Kuva 23. Pixyz kirjautuminen.

Aktivoidakseen lisenssin täytyy kirjautua sisään Pixyz tiliin ”online”-välilehdeltä (kuva 23). Kokeiluversio kestää seitsemän peräkkäistä päivää. Kun klikkaa ”7-day trial”-nappia verkkosivulla, tulee ponnahdusikkuna, jossa kysytään vahvistusta lisenssin aloittamiseen. Jos hyväksyy, seitsemän päivän kokeilu alkaa tästä hetkestä, eikä vasta silloin, kun asentaa lisenssin tietokoneelle.



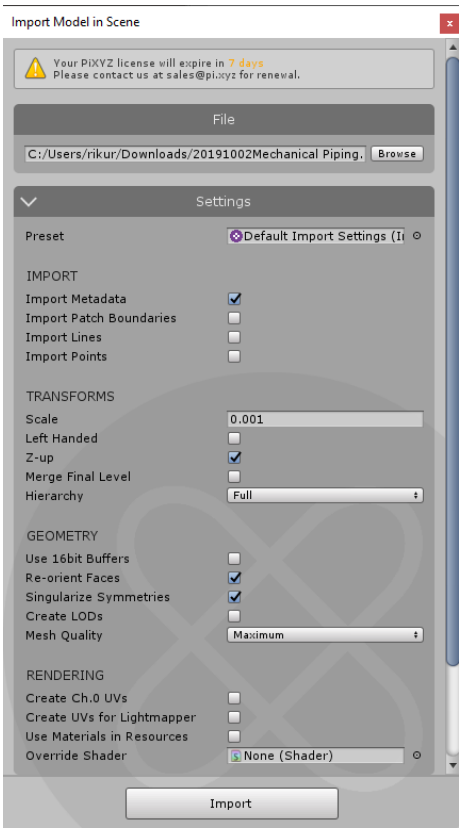
Kuva 24. Lisenssi aktivoitu.

Klikkaamalla ”Install” (kuva 24) ohjelma asentui Unityyn. Pixyzin kokeiluversion pystyy asentamaan yhdelle koneelle vain kerran. Lisenssi luodaan liittyen tietokoneen ID-tunnukseen. Jokaisella tietokoneella on uniikki Computer ID. Tämä tarkoittaa sitä, että Pixyziä ei pysty asentamaan samalle koneelle kahdesti, vaikka yrittäisi luoda toisen tilin ja asentaa.

Lisenssin aktivoimisen jälkeen Pixyz-valikon (kuva 5, s. 10) alta ”Import Model” pystyy tuomaan mallin Unityyn.

3.5 Pixyz mallin optimointi

3D-mallien tuonti Unity-pelimoottooriin käyttäen Pixyziä vaatii, että mallille suoritetaan optimointiprosessi. Prosessia pystyy säätämään käyttämällä tuomisasetuksia (kuva 25). Esimerkiksi mallissa olevien meshien laatua pystyy muuttamaan. Tämä asetus perustuu tesselaatio- tai desimaatio-algoritmeihin vähentämällä mallin polygonien määrää, kunnes vaadittu taso on saavutettu.



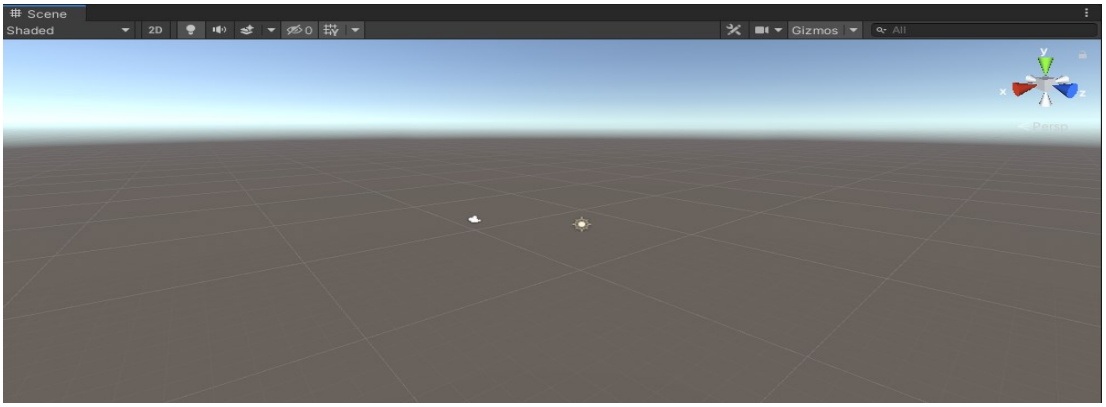
Kuva 25. Mallin tuomisikkuna.

Jos polygonien määrä on liian iso mallin tuomisen jälkeen ja Unity alkaa toimimaan hitaasti, on valittavana useita mahdollisia strategioita vaikuttaa tähän

riippuen käyttötilanteesta. Nämä toiminnot voidaan suorittaa manuaalisesti käyttäen Toolboxia tai automaattisesti käyttäen Rule Engineä.

3.5.1 ToolBox

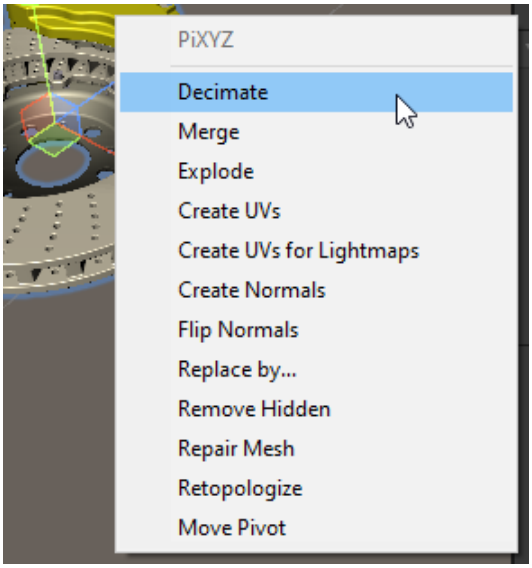
Toolbox on ominaisuus, joka otettiin käyttöön Pixyz Plugin for Unity versiossa 2018.3. Se on osa tämän laajennuksen käytettävissä olevia optimointi- ja vaiheistustyökaluja. (Pixyz s.a)



Kuva 26. Unityn näkymä (skene).

Toolbox mahdollistaa skenen (kuva 26) optimoinnin tai vaiheistamisen semi-automaattisesti. Toisin kuin Rule Enginessä, Toolboxin toiminnot kohdistetaan automaattiesti skenessä valittuun peliobjektiin. Tämä sallii tuonnin jälkeiset muutokset. Tästä voi olla hyötyä yksinkertaisissa muutoksissa, kätevän toimintaketjun löytämiselle säännön luomiseksi tai sellaisten toimintojen suorittamiseksi, joita Rule Engine ei voi tehdä. Toolboxin käyttö ei rajoitu Pixyzin avulla tuotuihin malleihin. Sen avulla voidaan työstää aivan kaikkia skenessä olevia asioita.

Toolboxissa on joukko oletustyökaluryhmän toimintoja, jotka sisältävät yleisiä työkaluja optimointiin tai vaiheistamiseen. Jotkut niistä käyttävät suoraan Pixyzin alkuperäisiä Core-algoritmeja.



Kuva 27. Toolboxin valikko.

Toolboxin valikko aukeaa klikkaamalla hiiren oikealla painikkeella peliobjektia skenessä, Pixyz-valikon kautta tai klikkaamalla hiiren oikealla painikkeella peliobjektia hierarkiassa. Toolboxissa on joukko erilaisia toimintoja. Esimerkiksi Decimate vähentää polygonien määrää meshissä. Repair Mesh on älykäs yhdistelmä algoritmeja, jotka korjaavat mallin automattisesti (kärkien hitsaus, pintojen suuntaus, reunojen romahtaminen). Remove Hidden poistaa automaattisesti sellaiset polygonit meshistä, joita ei pysty näkemään ulkoapäin.

3.5.2 Rule Engine

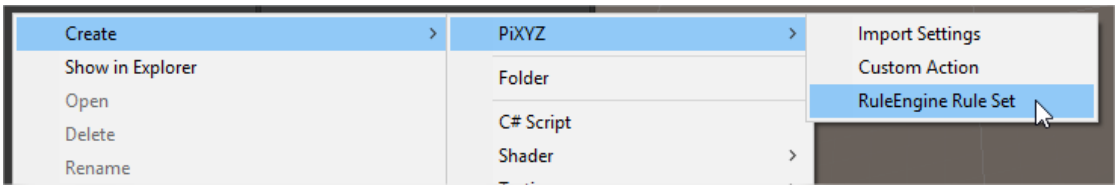
Rule Engine otettiin samaan aikaan käyttöön kuin Toolbox Pixyz Plugin for Unity versiossa 2018.3. Tämä on myös osa laajennuksen käytettävissä olevia optimointi- ja vaiheistustyökaluja.

Rule Enginen tärkein ominaisuus on skenen (kuva 26) optimointi tai vaiheistaminen täysin automaattisesti. Toisin kuin Toolboxissa, Rule Enginessä eri toiminnot suoritetaan automaattisesti kokonaisuutena peräkkäin. Tämä mahdollistaa yhdellä klikkauksella tapahtuvan tuonnin, optimoinnin ja vaihestamisen, mikä voi olla erittäin kätevää tuotaessa useita samanlaisia tiedostoja, jotka vaativat samanlaista käsittelyä.

Rule Enginessä on kokoelma oletustoimintoja, jotka sisältävät yleisiä työkaluja optimointiin ja vaiheistamiseen skenessä. Oletustoimintoja ovat Add, Custom, Debug, Filter, Get, Modify, Optimize, Scene, Set.

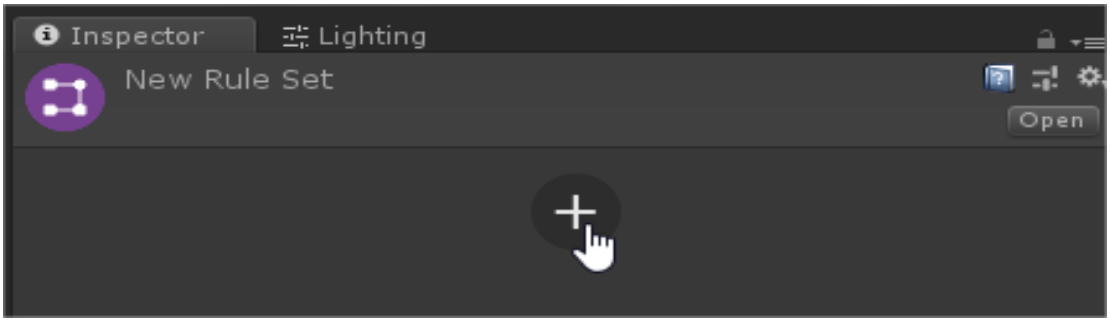
3.5.3 Rule Enginen käyttö

Rule Engine toimii sarjoitettujen sääntöjoukkojen kanssa, joten ensiksi on tehtävä joukko sääntöjä.



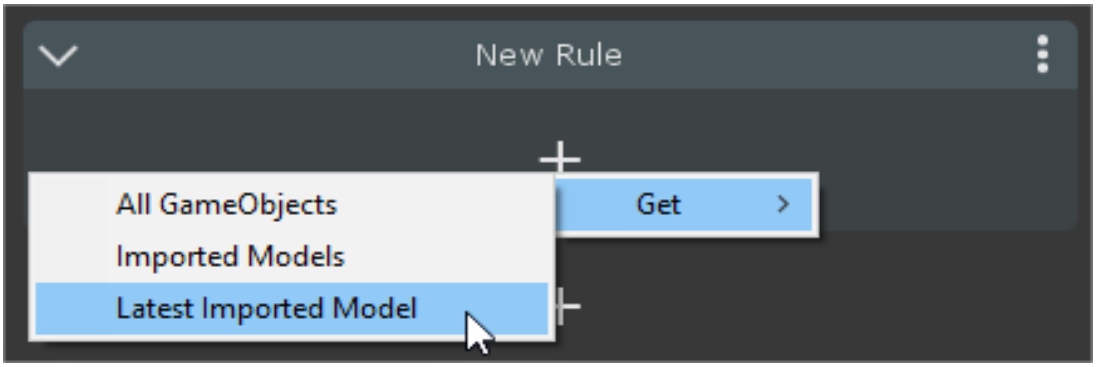
Kuva 28. Sääntöjoukon luominen.

Hiiren oikeaa näppintä klikkaamalla projektikansiossa löytyy valikko Create, josta löytyy PiXYZ ja sen alta RuleEngine Rule Set (kuva 27).



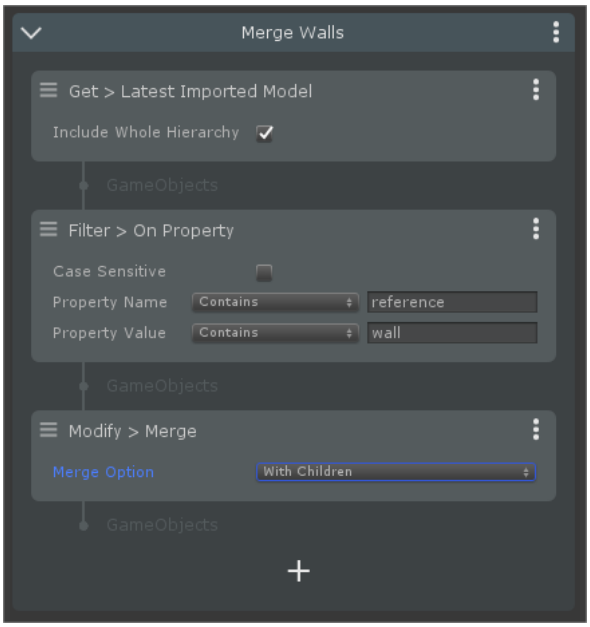
Kuva 29. Sääntöjoukon luominen.

Klikkaamalla juuri projektiin luotua sääntöjoukkoa, pääsee Inspectoriin, josta voi luoda säännön tälle sääntöjoukolle klikkaamalla plussaa (kuva 28).



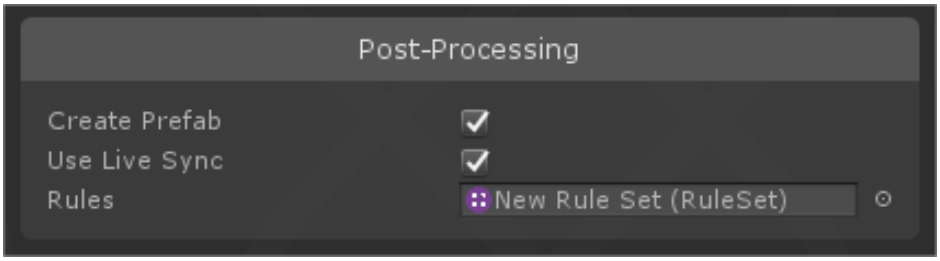
Kuva 30. Lähtökohdan lisääminen.

Esimerkiksi käyttämällä viimeisintä tuotua mallia lähtökohtana tiedetään, että seuraava tehty lohko sääntöjoukossa toimii viimeisimmällä Pixyzillä tuodulle mallille.



Kuva 31. Sääntöjen logiikka.

Tässä vaiheessa sääntöön luodaan oma logiikka. Esimerkiksi suodattimia, toimintoja, sekä sääntöjä voidaan luoda myös lisää. Säännöt suoritetaan järjestyksessä ylhäältä alas.



Kuva 32. Jälkikäsittely.

Lopuksi kun sääntöjoukko on tehty, sen voi suorittaa heti klikkaamalla ”Run” tai sen voi linkittää tuonti-ikkunaan.

3.6 Helpstertee Unity-IFCEngine

Unity-IFCEngine on saksalaisen filosofian tohtoriopiskelijan Helpsterteen suunnittelema IFC-tiedostojen tuontiskripti, jolla pystyy skaalaaman mallin

milli-, desi- ja senttimetreissä. Skriptillä pystyy määrittämään materiaalit materiaalimääritelmien mukaisesti Unity-komponentissa, sekä skriptin avulla saa IFC-mallista mallin pituusasteen, leveysasteen ja korkeuden. Skripti on ilmainen ja avoimen lähdekoodin alla.

IFCEngine käyttää CielaSpiken ThreadNinjaa, joka on yksinkertainen skripti. Skripti helpottaa kirjoittamaan taustalla pyöriviä monisäikeisiä rutiineja. Tämä skripti on luotu, koska Unityn API ei salli taustalla olevia säikeitä. ThreadNinjan pystyy lataamaan ja tuomaan projektiin Unityn Asset Storesta.

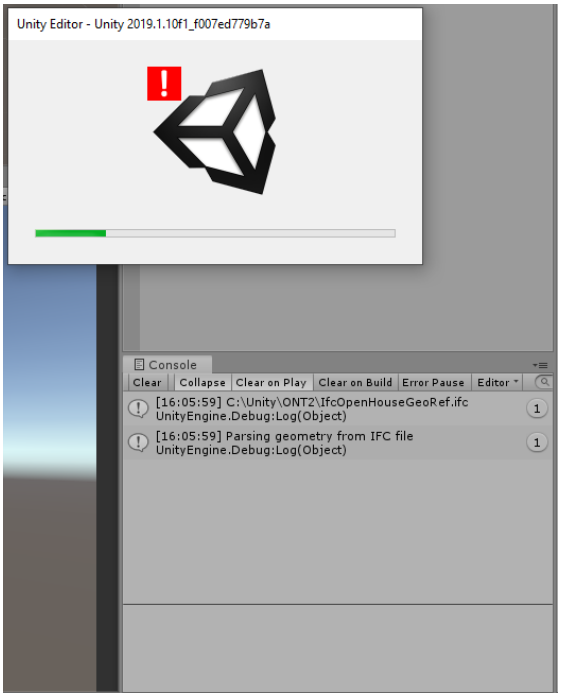
```
using System.IO;
using UnityEngine;

2 asset usages
public class test : MonoBehaviour
{
    Event function
    void Start()
    {
        ImportIFC import = GameObject.Find("Importer").GetComponent<ImportIFC>();
        import.Init();
        string path = Path.GetFullPath("IfcOpenHouseGeoRef.ifc");
        import.ImportFile(path, name: "IfcOpenHouseGeoRef", useNamesInsteadOfTypes: true);
    }
}
```

Kuva 33. IFC-tiedoston import koodi.

IFCEngineen kuuluvat tiedostot ladataan githubista zip-tiedostona ja puretaan projektin juureen. Skripti käyttää automaattisesti projektin juuressa olevia tiedostoja.

Yritin tuoda 44 megatavun kokoista 3D-mallia Unityyn käyttämällä Helpster-Teen IFCEnginen tuomisskriptiä. Koodi suoriutui muutaman sekunnin ajan pelin alkaessa, mutta sitten Unity kaatui tuntemattomaan ongelmaan (kuva 34).



Kuva 34. Unityn kaatuminen.

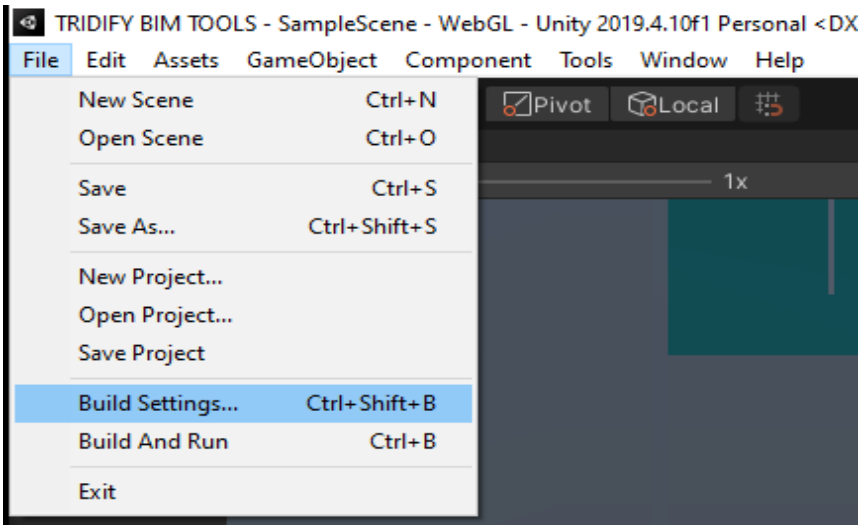
Ensimmäisenä tuli mieleen, että näin yksinkertainen skripti ei pysty käsittelemään suurehkoja malleja, joten päätin kokeilla uudelleen 170 kilotavun kokoisella (258 kertaa pienempi malli) mallilla. Unity kaatui tämän kokoisellakin mallilla. Tuntemattomasta virheestä ei oikein voinut lähtemään selvittämään, että mikä olisi ollut skriptissä vikana. Tulin siihen johtopäätökseen, että tämä skripti ei vain yksinkertaisesti toimi tai kaatuminen johtuu Unityn eri versioiden välisistä yhteensopivuusongelmista.

3.7 Kääntäminen WebGL

Unityn WebGL:än avulla pystyy renderöimään 2D- ja 3D-grafikkaa verkkoselaimessa käyttämättä erillisiä laajennuksia tai työkaluja. Julkaistuaan WebGL-koontiversion, Unity rakentaa verkkoselaimelle natiivin HTML5 javascript-ohjelman. (Unity Learn 2019.)

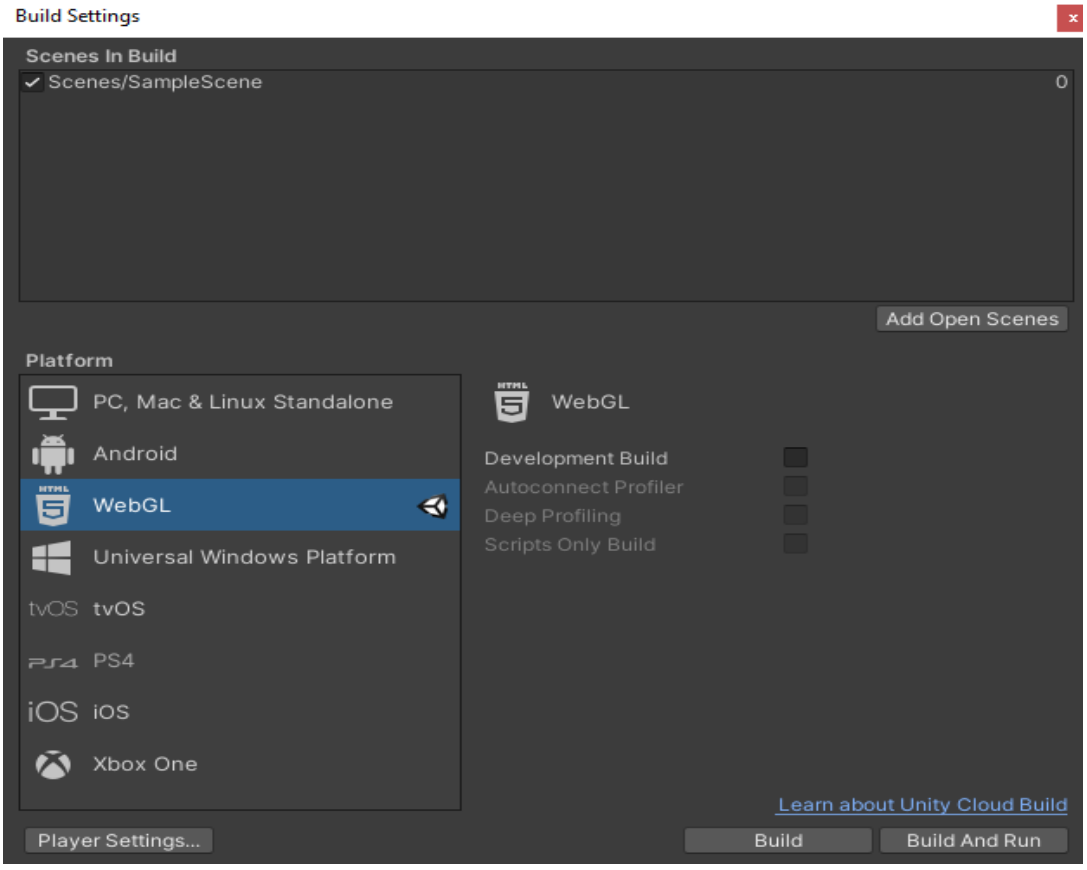
WebGL on tarkoitettu pienikokoisille ohjelmille, sillä käyttäjät joutuvat aina lataamaan ohjelman päästäkseen kokeilemaan sitä ja tämä voi olla erittäin hidasta, jos ohjelma on suuri ja käyttäjällä on alhainen latausnopeus. WebGL tukee vain tietokoneen näppäimistö- ja hiirisyötteitä. WebGL:lle julkaisemiselle on myös teknisiä rajoitteita. Esimerkiksi real-time Global Illumination, jonka on

3D-tietokonegrafiikassa tarkoitus lisätä realistisempaa valaistusta 3D-skeneihin ei ole tuettu WebGL:ssä. (Unity Learn 2019.)



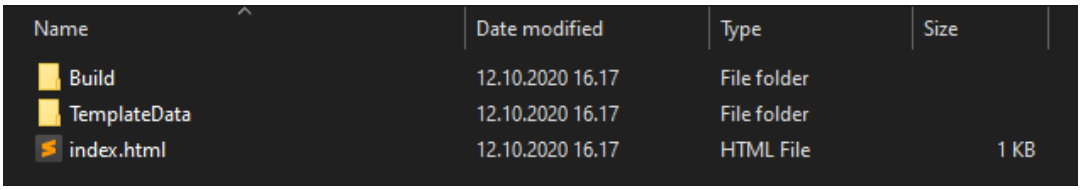
Kuva 35. Kääntämisvalikko.

Kääntäminen WebGL:lle tapahtuu samalla tavalla kuin muillekin alustoille, eli valitsemalla Unity-ikkunan vasemmasta yläreunasta File ja sen alta Build Settings (kuva 35).



Kuva 36. Ikkuna, josta pystyy kääntämään monelle eri alustalle.

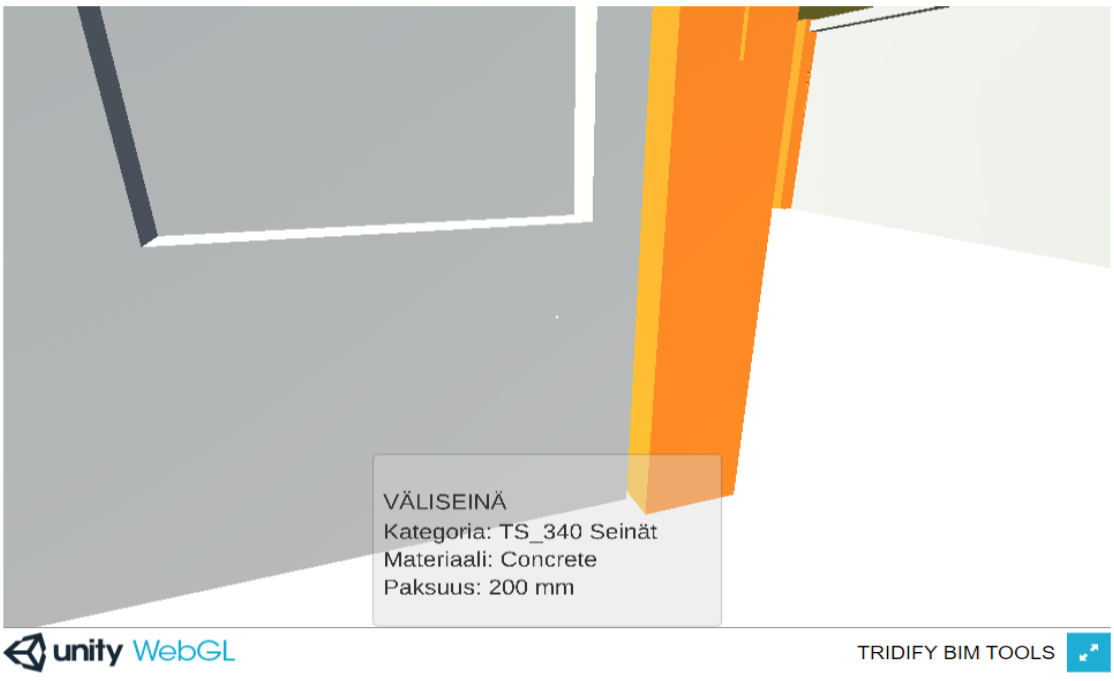
Build Settings-ikkunassa (kuva 36) on ennen millekään alustalle kääntämistä valittava ne skenet, jotka tulevat mukaan käännökseen. Tämä tapahtuu Add Open Scenes-nappia painamalla, jolloin Unity lisää kaikki avoinna olevat skenet käännökseen. Jokaisen skenen vieressä on myös täppä, jonka avulla voi ottaa skenen pois käännöksestä.



Kuva 37. WebGL-käännös valmis.

Unity pystyy kääntämään projekteja monelle eri alustalle, kuten Windows, Mac, Linux, Android, WebGL, tvOS, PS4, iOS ja Xbox One. WebGL:lle kääntäminen tapahtuu valikosta valitsemalla WebGL, jossa on HTML5-logo. Sen jälkeen oikeaan alakulmaan ilmestyy nappi (”Switch Platform”), joka vaihtaa Unityn ja Unityn kääntämisasetukset kyseiselle alustalle.

Tämän jälkeen painamalla (”Build”) tai (”Build And Run”) -nappia Unity kysyy mihin kansioon käännös tehdään. Käännöksen lopuksi Unity on luonut kaksi kansiota ja yhden index.html-tiedoston (kuva 37), jonka avaamalla palvelimelta selaimessa WebGL:lle käännetty projekti käynnistyy.

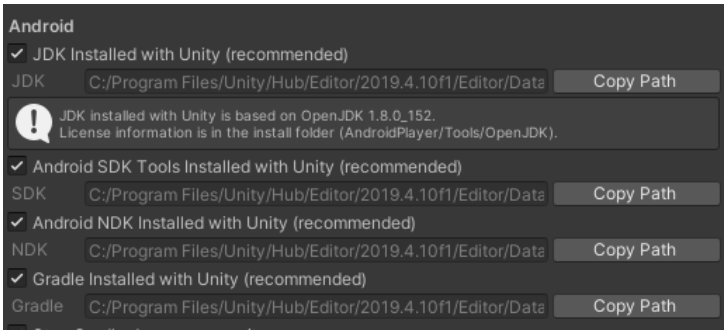


Kuva 38. 3D-rakennusmallin demo WebGL:llä.

Havaitsin WebGL:lle käännetyssä versiossa ulkopuolelta rakennusta katsoessa, että ruutunopeus laski todella paljon verrattuna Windowsille käännettyyn versioon. Rakennuksen sisällä ruutunopeus oli suunnilleen sama molemmilla alustoilla.

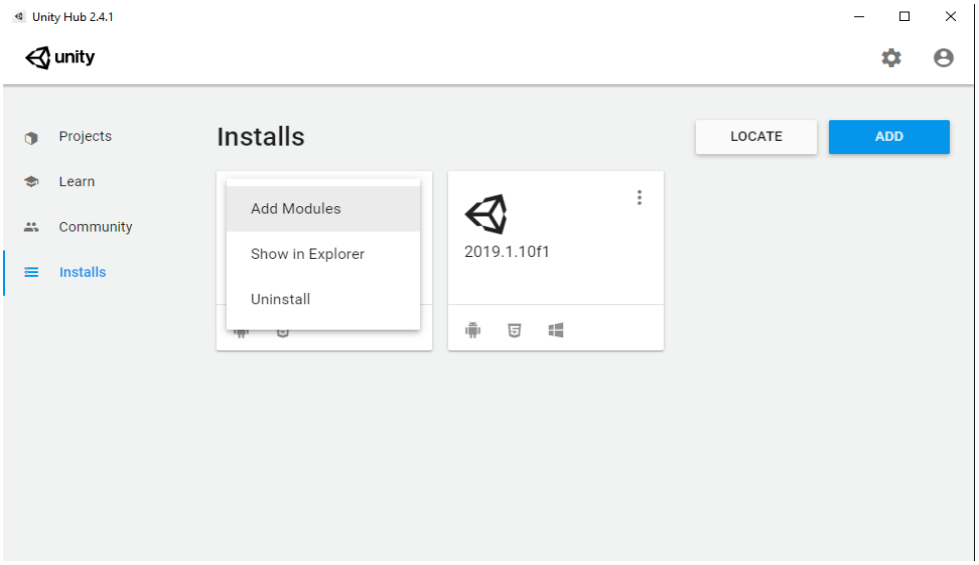
3.8 Kääntäminen mobiililaitteelle (Android)

Unity mahdollistaa pelien kääntämisen kahdelle eri mobiilialustalle; Androidille ja iOS:lle. Androidille kääntämiseen Unity tarvitsee Java Development Kitin, Android Software Development Kitin, Android Native Development Kitin, sekä Gradlen. Kaikki nämä pystyy asentamaan Unity Hubin kautta.



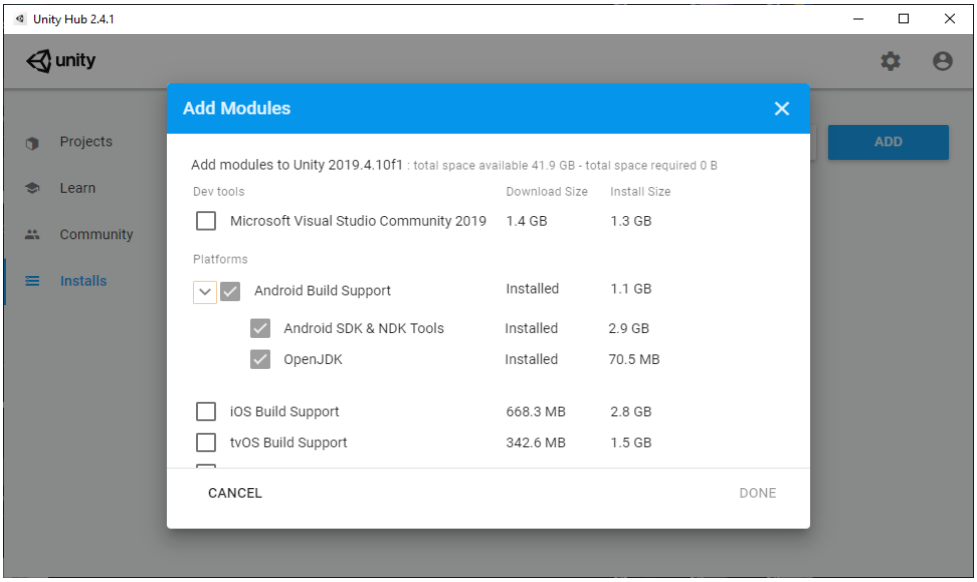
Kuva 39. Androidille kääntämiseen tarvittavat riippuvuudet.

Unity Hubin Installs-valikosta löytyy kaikki eri Unityn versiot, jotka ovat asennettuna tälle koneelle (kuva 40). Klikkaamalla kolmea pistettä halutun Unity version kohdalla tulee valikko (kuva 41), josta pystyy lisäämään moduuleja eli tuen eri alustoille kääntämiselle.



Kuva 40. Unity Hub moduulien lisäys.

Moduulivalikosta (kuva 41) näkyy ensimmäisenä, mitkä kehitystyökalut on asennettu Unityyn. Lisäksi näkyy, mitkä moduulit on jo asennettu tähän Unityn versioon. Moduulivalikosta pystyy myös asentamaan Unityn dokumentaation, sekä eri kielipakkauksia. Jokaisesta asennetusta osasta näkyy asennuksen koko.

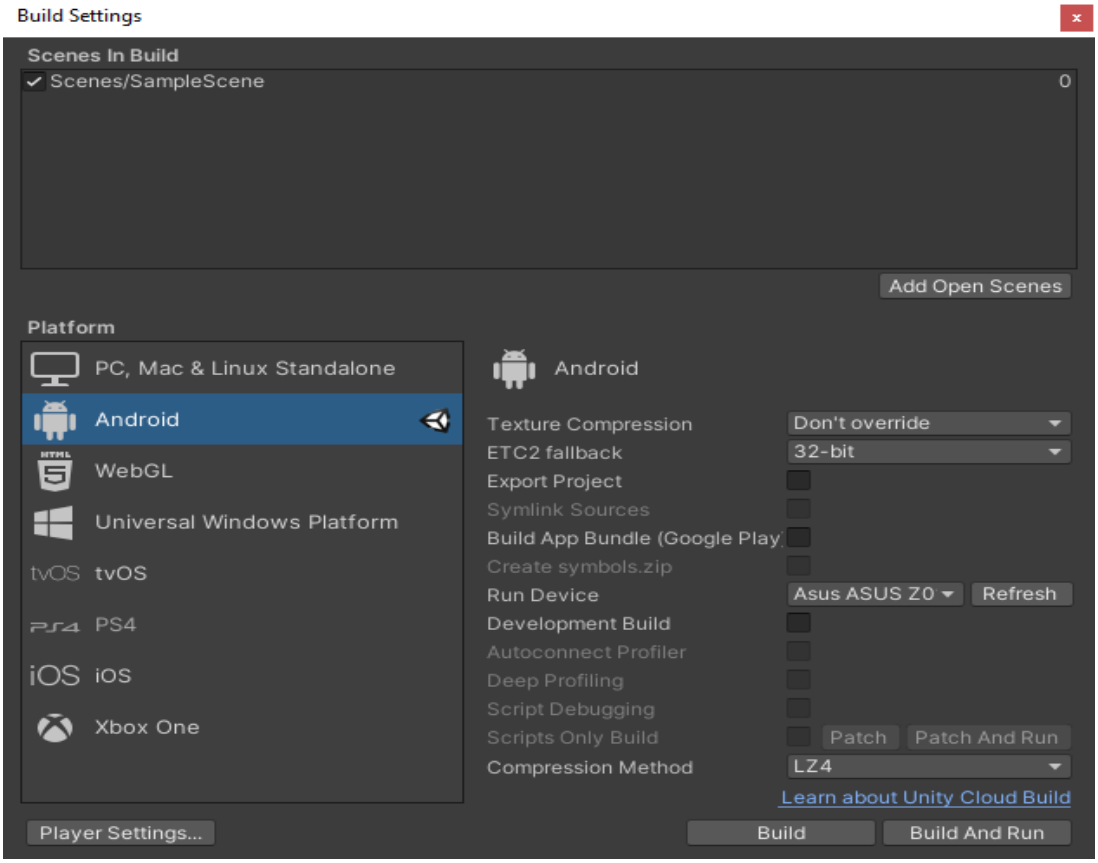


Kuva 41. Moduulivalikko.

Kun kaikki tarvittavat moduulit on asennettu, löytyy unityn käännösvalikosta kääntäminen Androidille (kuva 42). Mobiililaitteelle kääntäessä tulee aina huomioida se, että niiden suorituskyky ei ole lähellekään tietokoneiden suorituskykyä, joten Androidille kääntämisen yhteydessä valikosta (kuva 42) löytyy erilaisia valinta-asetuksia. Ensimmäisen on (”Texture Compression”). Tämän valikon alta löytyy erilaisia tekstuurin pakkausmenetelmiä. 3D-grafiikkalaitteisto edellyttää tekstuurien pakkaamista erikoistuneissa muodoissa, jotka on optimoitu nopeaan tekstuurin näyttämiseen.

ETC2 fallback-valikon alta löytyy tekstuurin dekompressiometodeita. Androidlaitteet, jotka eivät tue ETC2:ta, voivat ohittaa oletus ETC2 tekstuurin dekompressionin valitsemalla valikosta 32-bit, 16-bit tai 32-bit puolitetulla resoluutiolla. 32-bittinen vaihtoehto sisältää korkeimman laatuiset tekstuurit ja se vie tuplasti levytilaa verrattuna 16-bittiseen, mutta 16-bittinen tekstuuri menettää joitakin värejä. 32-bittinen vaihtoehto puolitetulla resoluutiolla vähentää muistinkäyttöä

huomattavasti, mutta tekstuurista tulee todennäköisemmin enemmän epäselvä. (Unity Documentation 2019.)



Kuva 42. Kääntäminen Androidille.

Viimeisimpänä löytyy Compression Method. Tämän valikon alta löytyy datan pakkausmenetelmiä (Default, LZ4, LZ4HC). Oletuksena Unity pakkaa ZIP-tiedostoon, mikä antaa jokseenkin paremmat pakkaustulokset kuin LZ4 tai LZ4HC, mutta data on hitaampaa dekompressoida.

LZ4 pakkausmenetelmää käytetään enimmäkseen kehitysversioissa (”Development Builds”). Tätä pakkausmenetelmää käyttämällä Unityllä tehtyjen pelien latausajat lyhenevät huomattavasti. Lopuksi LZ4HC pakkausmenetelmää, joka on muunnos LZ4 pakkausmenetelmästä, käytetään yleensä julkaisuversioissa (”Release Builds”). Tätä pakkausmenetelmää käyttämällä saa paremmat tulokset julkaisuversioon. (Unity Documentation 2019.)

Kuva 43. 3D-rakennusmallin demo mobiililaitteella.

Käytin Build Settings-valikossa (kuva 42) oletusasetuksia kaikkiin vaihtoehtoihin. Mobiililaitteella ongelmaksi ilmeni hahmon liikuttaminen. Hahmolla pystyi vain katsomaan ympärilleen ja sekin oli aika vaikeaa. Tuntui kuin kääntymisen herkkyys olisi aivan liian suurella. Ruutunopeudessa en huomannut rakennuksen sisällä mitään ongelmaa. Demo pyöri aika sulavasti.

4 PÄÄTÄNTÖ

Opinnäytetyön tavoitteena oli selvittää, minkälaisin eri keinoin on mahdollista tuoda IFC-rakennusmalleja Unity pelimoottoriin, sekä tuottaa pienimuotoinen demo, jossa pienessä pelimaailmassa käyttäjä pystyy tarkastelemaan 3D-rakennusmallia ja siihen liittyviä ominaisuuksia. Opinnäytetyön aikana käsitys 3D-mallien optimoinnista peliympäristöön sopeutuvaksi kasvoi huomattavasti. Perehdyin paljon erilaisiin pakkausmenetelmiin koskien, sekä tekstuureja, että 3D-malleja.

Suurimpana haasteena oli Pixyz-softwaren käyttö. Uusimmassa versiossa Pixyzistä oli semmoinen bugi, joka ei päästänyt ollenkaan kirjautumaan Pixyz-tilille ja sitä myöten tuomaan mallia Unityyn. Pixyzin ja Tridifyn dokumentaatioon olin tyytyväinen. Tietoa löytyi enemmän kuin tarpeeksi. HelpsterTeen UnityIFCEngine oli mielestäni surkea enkä suosittelisi sen käyttöä, koska en saanut sillä suoritettua yhtään demoa eikä se toiminut, vaikka onkin ilmainen.

Suosittelen Pixyz Plugin for Unityä Tridifyn ylitse. Syynä on paljon laaduk-kaammat ja monipuolisemmat mallin optimointivaihtoehdot. Pixyz tarjoaa yh-den ainoan lisenssin, joka on 1000 euroa vuodessa. Tridify tarjoaa kolme eri lisenssivaihtoehtoa, joista Light 20 dollaria kuukaudessa, Standard 90 dollaria kuukaudessa ja Large 400 dollaria kuukaudessa.

LÄHTEET

3D Repo. 2020. How to Import BIM models to Unity. Saatavissa: https://3drepo.com/how-to-import-bim-models-to-unity/ [viitattu 3.11.2020].

Aecmagazine. 2019. PiXYZ plug-in brings IFC files into Unity's real time engine. WWW-dokumentti. Saatavissa: https://aecmag.com/technology-main-menu-35/1777-pixyz-plug-in-brings-ifc-files-into-unity-s-real-time-engine [viitattu 27.8.2020].

Buildingsmart. 2020. Industry Foundation Classes (IFC) - An Introduction. WWW-dokumentti. Saatavissa: https://technical.buildingsmart.org/standards/ifc/ [viitattu 25.8.2020].

Graphisoft. 2020. Geometry Conversion for IFC Export. WWW-dokumentti. Saatavissa: https://helpcenter.graphisoft.com/user-guide/128918/ [viitattu 25.8.2020].

Magicad. Building Information Modelling. WWW-dokumentti. Saatavissa: https://www.magicad.com/en/bim/ [viitattu 21.9.2020].

Pixyz. GET YOUR 3D DATA READY FOR NEW EXPERIENCES. WWW-dokumentti. Saatavissa: https://www.pixyz-software.com/ [viitattu 21.9.2020].

Tridify. 2019. How to import BIM models into Unity. WWW-dokumentti. Saatavissa: https://go.tridify.com/support/how-to-import-bim-models-into-unity [viitattu 25.8.2020].

Unity Documentation. 2019. Building apps for Android. Saatavissa: https://docs.unity3d.com/Manual/android-BuildProcess.html [viitattu 12.10.2020].

Unity Learn. 2019. Päivitetty 2020. How to publish for WebGL. Saatavissa: https://learn.unity.com/tutorial/how-to-publish-for-webgl#5d925085edbc2a0c00bd4608 [viitattu 12.10.2020].

Zhong-Qi, W. & I-Chen, W. 2015. Design and Implementation of an IFC Data Model to Unity Data Model Transformation Mechanism. Pdf-dokumentti. Saatavissa: http://www.see.eng.osaka-u.ac.jp/seeit/icccbe2016/Proceedings/Full_Papers/203-064.pdf [viitattu 10.9.2020].

KUVALUETTELO