Pentti Liikanen

3D-MALLINNUKSEN MUOTOON TEHTÄVÄ RAKENNUKSEN KÄYTTÖ- JA HUOLTO-OHJE

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2020

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkintonimike	Aika
Pentti Liikanen	Tradenomi (AMK)	Lokakuu 2020

Opinnäytetyön nimi

3D-mallinnuksen muotoon tehtävä rakennuksen käyttö- ja huolto-ohje

48 sivua

Toimeksiantaja

ArkVisio Oy

Ohjaaja

Jukka Selin

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli tutkia mahdollisuuksia rakennuksen elinkaaren aikaisen tietomallin pelillistämiseksi 3D-muodossa. Tutkielman tarkoitus oli toteuttaa kevyt, monipuolinen ja helposti luotava ratkaisu metatietojen sisältämän tietomallin interaktiiviseen, kaksisuuntaiseen käyttöön rakennuksen elinkaaren ajaksi. Toiveena oli myös sovelluksen toimiminen verkkoselaimessa ja eri pelillistettyjen mallien liittäminen myöhemmin virtuaaliseen kaupunkiin.

Työssä selvitettiin teoreettiset perusteet tietomallille ja mallinnukselle (BIM). Lisäksi tutkittiin eroavaisuuksia toteutukselle eri ohjelmien välillä. Rakennuksen käyttö- ja huolto-ohjeesta toteutettiin pelkistetty prototyyppi.

Teoreettisen tiedon pohjalta todettuna pelimoottori toteuttaa muodot polygon-malleina ja CAD-ohjelma osin myös splini-käyrinä ja -pintoina. Splinistä polygoniksi muunnoksessa polygonien määrä kasvaa usein liikaa. Toteutuksessa muodostuvan ohjelman tiedostokoko muodostaa oman haasteen ylläpito- ja käyttöympäristölle. Tietomallin 3D-mallinnuksen renderöinti ja käyttö vaativat selaimelta yhteensopivuuden. Standardien puute tiedonsiirrossa vaikeuttaa tietojen yhteensovittamista eri ohjelmien välillä. IFC-tiedosto on yksi ratkaisu ongelmaan.

Empiirisen testauksen pohjalta ArchiCAD tuottaa BIMx hypermallin, joka soveltuu geometrian ja metatietojen esittelyyn, mutta ei oikein korjaus- ja huolto-ohjeen toteuttamiseen. Unityn ja Unreal Enginen välillä suurin ero on Unreal Enginen valmius käyttää IFC-tiedostonsiirtoa sisäänrakennettuna. Unity vaatii maksullisten lisäosien käyttöä IFC-tiedoston käyttöön tai geometrian ja metatietojen tuomisen erillisissä tiedostoissa.

Prototyyppivaiheessa tuotteesta ei ole suoranaista hyötyä toimeksiantajalle. Vaatimusmäärittelyn ja prototyypin pohjalta käyttökelpoinen ohjelma on mahdollista toteuttaa. Tulevaisuuden kuvana näkisin tuotteella olevan kaupallista potentiaalia oikeaan aikaan ja oikein toteutettuna.

Asiasanat

3D-mallinnus, rakennuksen tietomalli, BIM, pelillistäminen

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Pentti Liikanen	Bachelor of Business Administration	October 2020

Thesis title

3D user and maintenance manual of a building

48 pages

Commissioned by

ArkVisio Oy

Supervisor

Jukka Selin

Abstract

The objective of this thesis was to study the possibilities of a building's information model gamification in 3D. The application should be usable for the building's full life cycle. An interactive two-way application should be usable on a general internet browser. Also allowing adding new gamified information models to an existing virtual city.

The theoretical bases of a building information model (BIM) and modeling were studied. Also, the differences in developing between programs were under research. A 3D user and maintenance manual of a building was implemented as a prototype.

A game engine creates models as polygon meshes. A CAD program uses splines and spline surfaces too. In a spline to polygon conversions the polygon count may rise too much. The file size of a program causes a challenge for maintenance and user environment. The information model's 3D rendering, and general use requires certain interoperability from the browser. Lack of standards causes difficulties in file transfer between programs. An IFC file transfer is one solution for the problem.

In empirical tests, it was found that ArchiCAD created a BIMx Hyper-model which served better for a geometry and a metafile presentation rather than for creating user and maintenance manual. In this project, the biggest difference between the chosen game engines was Unreal Engine's built-in readiness to import the IFC file. Unity required a chargeable plugin to use IFC file directly or needed to import geometry and metafiles in separate files.

On a prototype stage, the product was not suitable for the thesis commissioner. With an existing requirement specification and a prototype, it is possible to implement a usable application. As a future prospect, I would see a possibility of a commercial success if the product is implemented in correct time and form.

Keywords

3D modeling, building data model, BIM, gamification

SISÄLLYS

1 JOHDANTO

ArkVisio Oy:n toimitusjohtaja Jukka Laamanen ilmaisi keskustelussamme tammikuun loppupuolella 2020 kaipaavansa eräänlaista 3D-mallinnettua digitaalista tiedostorakennetta rakennuksen koko elinkaaren ajaksi. Tästä keskustelusta virisi idea toteuttaa mallinnus kolmiulotteisesti IFC-tiedonsiirtoa hyväksikäyttäen ja metatiedot sisältäen, interaktiivisen kaksisuuntaisen käyttöliittymän kera.

Tämän opinnäytetyön tarkoituksena on tutkia eri mahdollisuuksia toteuttaa rakennuksen käyttö- ja huolto-ohje 3D-tietomallin pohjalta. ArkVisio Oy käyttää ArchiCAD BIM -ohjelmaa erilaisten rakennusten suunnitteluun ja mallintamiseen.

Tavoitteena on eräänlainen pelillistetty 3D-tietomallinnus, joka koostuu tietomallista sekä rakennuksen elinkaaren aikana muodostuvista muista tiedoista koskien esimerkiksi käyttöä ja huoltoa. Ohjelmaa tulisi voida käyttää web-selaimella ja sen sisältöä tulisi pystyä päivittämään ja muokkaamaan jopa vuosikymmenien ajan. Tarkoitus on myös tutkia ArchiCAD-ohjelman soveltuvuutta pelillistämiseen verrattuna oikeaan pelimoottoriin.

ArchiCAD itsessään osaa luoda BIMx-hypermallin tietomallin eri tietojen esittelyyn. Unreal Engineen saa tuotua tietomallin geometrian ja metatiedot IFC-muodossa suoraan natiivisti tai Datasmith-työkalulla. Unity vaatii ulkoisen lisäosan käyttöä tai IFC-tiedoston sisältämän geometrian vientiä omassa 3D-tiedostomuodossaan ja IFC-tiedoston sisältämien metatietojen vientiä erillään esimerkiksi erillisessä XML-tiedostomuodossa.

Haluan tässä yhteydessä kiittää ArkVisio Oy:tä ja erityisesti toimitusjohtaja Jukka Laamasta, joka tämän toimeksiannon mahdollisti, *ajattele isosti* -mielikuvalla. Lisäksi haluan kiittää Kaakkois-Suomen ammattikorkeakoulun yliopettajaa Jukka Seliniä, joka ohjasi kärsivällisesti opinnäytetyön kulkua ja it-asiantuntijaa Juha Ojalaa, joka neuvoi ja antoi vinkkejä Unreal Engineen liittyvissä

ongelmissa. Viimeisenä, mutta ei suinkaan vähäisimpänä, olen hyvin kiitollinen ArkVision toteuttaman suunnitelman tilaajien, Pirjo Tuomi sekä Jan Lehtinen, myötämielisyydestä käyttää tietomallia tässä opinnäytetyössä.

Tässä aineistossa kuvataan tietomallin ja -mallinnuksen teoreettiset perusteet ja joitain käytännön sovelluksia. 3D-geometrian, metatietojen sekä muiden rakennuksen käyttö- ja huolto-ohjeeseen liittyvien tietojen merkitys selitetään myös. Tietomalliin ja -mallinnukseen liittyvä IFC-tiedonsiirtostandardi ja sen merkitys avataan lukijalle.

Aineistossa tutkitaan empiirisesti eroja tietomallin pelillistämisessä yhden tietomallisuunnitteluohjelman ja kahden eri pelimoottorin kesken. Opinnäytteessä määritetään yleisluontoinen vaatimusmäärittely ja selostetaan yksinkertaisen prototyypin luominen. Ongelmat ja huomiot kuvataan niiltä osin, kun ne vaikuttivat työn kulkuun. Lopuksi arvioidaan pelillistetyn tietomallin merkitystä toimeksiantajalle sekä yleisesti. Käyttö- ja huolto-ohjeesta toteutetaan Windows-pohjainen natiiviohjelma.

2 TIETOMALLI JA MALLINNUS

Tietomalli tarkoittaa tässä työssä rakennuksen tietomallia (engl. *Building Information Model*), mistä käytetään myös lyhennettä **BIM** (M.A.D. 2013). Tietomalli sisältää rakennuksen 3D-geometrian ja muun rakennuksen datan. IFC-standardi mahdollistaa tiedon siirron eri BIM-ohjelmien välillä. Standardoidussa IFC-tiedonsiirtoformaatissa määritellään geometrian lisäksi eri elementtien ominaisuudet ja muut tiedot niin sanottuna metatietona. (buildingSMART 2020c.)

Tietomallin pelillistämisessä tulisi saada mallinnuksen geometria ja metatiedot esitettyä sellaisessa muodossa, että käyttäjä voi esimerkiksi osoittaa rakennuksen osaa tai kohdetta, joka tunnistuu halutusti. Pelillistämisessä tietomalliin lisätään mahdollisuus liikkua mallinnetussa ympäristössä ja toimia interaktiivisesti määriteltyjen toimintojen avulla. (Edwards ym. 2015, 1-2.)

Pelillistämisen yhdeksi ongelmaksi muodostuu splinit eli käyrät, joita usein käytetään viivojen ja pintojen mallintamiseen BIM- ja CAD-ohjelmissa. Pelimoottori muodostaa 3D-kuviot kaksitasoisena näytöllä tiettyyn paikkaan sijoittuvien kärkien ja niiden väliin tulevien reunojen kautta muodostuvilla polygoniverkoilla. Splinit muunnetaan pelillistämisprosessissa polygoniverkoiksi, jotta reaaliaikainen kuvan renderöinti olisi ylipäätänsä mahdollista. Sujuva reaaliaikainen renderöinti vaatii myös tietomallin polygonimäärän optimointia. Mitä vähemmän laskettavaa näytönohjaimella on, sen sujuvammin piirto toteutuu. (Luku 2.6.)

Käyttö- ja huolto-ohjeen liittäminen pelillistettyyn tietomalliin vaatii oman huomionsa. Käyttäjällä on oltava kaksisuuntainen yhteys ohjeiden käyttöön ja oman tiedon tallentamiseen. Käytännöllisintä olisi sijoittaa kaikki tieto sijainnista riippumattomaan paikkaan, esimerkiksi verkkopalvelimelle tai pilvipalveluun. (Edwards ym. 2015, 19.)

Rakennuksen tietomalli koostuu rakennuksen kaikista tiedoista digitaalisessa muodossa. Koko rakennuksen tietomalli koostuu eri suunnittelualojen osamalleista. Osamalleja ovat esimerkiksi arkkitehtuuri-, rakenne- ja talotekniikkamallit. (Nevalainen 2016, 1-5.)

Tietomalliin sisällytetään tarvittavat tiedot mm. geometriasta ja metatiedoista. Tietomalli mahdollistaa rakennuksen eri osien suunnittelijoiden pääsyn mallinnukseen ja metatietoihin reaaliaikaisesti. Suunnittelijat voivat simuloida ja testata erilaisia muuttujia ennen toteuttamista. Havaitut poikkeamat voidaan korjata tietomalliin ennen rakentamista. Näin säästetään aikaa ja materiaaleja. Metatietoina voidaan tallentaa kaikki relevantti rakennusta koskeva tieto esimerkiksi aikatauluista, kustannusarvioista, dokumentoinneista ja määristä. (Garber 2018, 14.)

Tietomallista voidaan tuottaa myös niin sanottu **Digital Twin**, mikä tarkoittaa rakennuksen digitaalista kaksosta vapaasti suomennettuna. Se muun muassa kokoaa staattisen ja dynaamisen tiedon eri lähteistä reaaliaikaisesti käyttäjän näkemäksi visuaaliseksi ilmentymäksi. (Granlund s.a.)

Mallinnus on tietomallin geometrinen ulkoasu. 3D-mallinnus toteutetaan mallinnusohjelmassa kolmiulotteiselle koordinaatistolle *x-, y- ja z*-akseleille. Eri 3D-mallinnusohjelmilla voi olla erilainen koordinaatisto, mikä määrittelee mallin positiivisen ja negatiivisen kiertosuunnan. Toteutus on kolmiulotteinen malli annetusta toimeksiannosta. (Tuhola & Viitanen 2008, luku 2.)

3D-mallinnus toteutetaan tarkoitukseen soveltuvalla ohjelmalla. 3D-malli piirretään erilaisilla splini-käyrillä -tai pinnoilla ja/tai polygoneilla hieman ohjelmasta ja käyttötarkoituksesta riippuen. Mallinnusohjelmassa mallia voidaan liikuttaa, pyörittää ja skaalata, sekä muokata monin eri tavoin. (Gumster 2009; Home 2015.)

3D-mallinnus tehostaa ja nopeuttaa varsinkin monimutkaisten mallien hallintaa verrattuna perinteiseen kaksiulotteiseen esitystapaan. Mallinnukselle on asetettava määrätty origo, jotta malli asettuu koordinaatistoon halutulla tavalla.

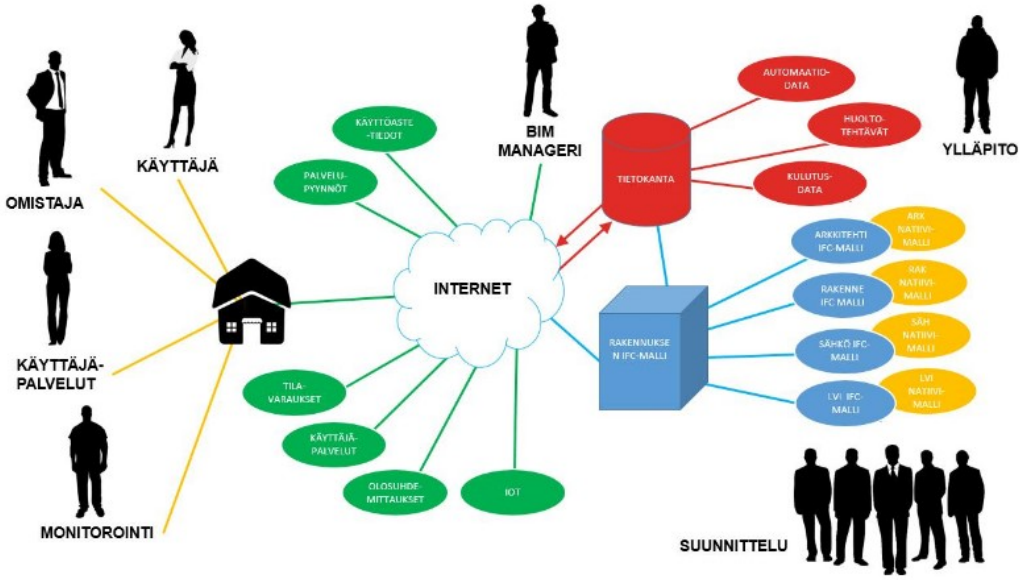
Origo voi olla esimerkiksi mallin keskipiste tai jokin avainsivun kulma. Muutoin malli voi esimerkiksi esiintyä väärinpäin tai väärässä kohdassa käytettäessä mallinnusta toisessa ohjelmassa. 3D-ohjelma ei itsessään tee juuri mitään ilman suunnittelijan ammattitaitoa. (Tuhola & Viitanen 2008, luku 2.)

Mallinnuksen on täytettävä tietyt ehdot toimiakseen sovelluksessa suunnitellulla tavalla. Liian monimutkainen malli ei välttämättä toimi riittävän sujuvasti kaikilla käyttöalustoilla. Tiedoston koko, sovelluksen muistinkäyttö ja päätelaitteen 3D-renderöintikyky vaikuttavat sovelluksen käyttökelpoisuuteen. Varsinkin vanhemmalla tai heikolla suorituskyvyllä varustetulla mobiililaitteella voi olla vaikea tai mahdoton pyörittää optimoimatonta sovellusta. Kun mallinnus on optimoitu lähtökohdiltaan riittävän hyvin, on jatkokäsittelykin helpompaa. (Botsch ym. 2008, luku 2.)

2.1 Tietomallin tarkoitus

Halmetojan (2016, 6) mukaan: ”Rakennuksen tietomalli on rakennuksen ja rakennusprosessin koko elinkaaren aikaisten tietojen kokonaisuus digitaalisessa muodossa.” Tietomalli sisältää rakennuksen geometriset määritteet kolmiulotteisen esittämisen havainnollistamisen ja simulointitarpeiden vuoksi.

Garberin (2014, 157) mukaan tietomalleilla voidaan toteuttaa myös niin sanottu törmäytys (engl. *clash detection*) rakennuksen eri alojen komponenttien ja osamallien välillä, jolloin selvitetään niiden yhteensopivuus keskenään.



Kuva 1. Esimerkki tietomallin käytöstä (Halmetoja 2016)

M.A.D:n (2013) mukaan: ”BIM ei kuitenkaan tarkoita pelkästään 3D-mallinnusta, vaan aitoa tietomallia hyödynnetään määrien ja energia-arvioiden laskentaan, aikataulutukseen ja muuhun ei-graafisen tiedon käyttöön.”

Halmetojan (2016, 8) mukaan tietomalli voi olla n-ulotteinen (n x D). Halmetoja määrittelee **yhdistelmämallin** useiden osamallien kokonaisuudeksi (mts. 5). Lisäksi Halmetoja (mts. 19) toteaa: ”**Ylläpitomalli** tarkoittaa rakennuksen yhdistelmämallia, jossa on esitetty käytön ja ylläpidon aikaista huoltoa, käyttöä ja kunnossapitoa vaativat rakenteet ja laitteet.” **Olosuhdemallin** tietosisältöä ei ole määritelty, mutta se voi esittää tietomallin avulla rakennuksen muuttuvia arvoja sensorien avulla, kuten lämpötiloja, ilman kosteutta tai vastaavia (mts. 22-23).

Tietomallien standardeja ovat:

- IFC – Industry Foundation Classes tarkoittaa tietomalliohjelmistojen yhteistä kuvaustapaa tai avointa tiedonsiirtomuotoa.
- DD – Data Dictionary tarkoittaa kansainvälistä nimikkeistöä tietomallikomponenteille.

- IDM – Information Delivery Manual on buildingSmartin (2020b) mukaan: ”Prosessikuvaus siitä, mitä tietoa tietomalleilla siirretään eri toimijoiden välillä eri käyttötilanteissa.”
- MVD – Model View Defination on buildingSmartin (mt.) mukaan: ” Tekninen kuvaus siitä, mitä IFC -muotoista tietoa eri toimijoiden välillä tietomalleilla siirretään eri käyttötilanteissa.”
- BCF – Building Collaboration Manual on tiedonvälitysmuoto, jolla voidaan siirtää älykkäitä viestejä eri tietomalliohjelmistojen välillä. (Mt.)

Tietomalli on nykyään olennainen osa rakennusten suunnittelua ja rakentamista. Se mahdollistaa tiedon ja datan helpomman käsittelyn, päivittämisen ja jakamisen eri osapuolten kesken. Monipuolisuutensa ansioista se käy moniin erilaisiin toteutuksiin ja on muokattavissa monin eri tavoin. 4D-suunnitteluksi kutsutaan tietomallia, johon on lisätty tehtävien aikataulutus. Tietomalliin voidaan sisällyttää pelimoottorin avulla kaksisuuntainen tietoyhteys, jolloin ei-ammattilainen loppukäyttäjä voi liikkua ja toimia virtuaalimallissa interaktiivisesti. (Edwards ym. 2015, 1-2.)

Tietomallia voidaan käyttää esimerkiksi erilaisiin olosuhdesimulointeihin, suunnitelmien testauksiin ja virtuaalisiin esittelyihin. Esimerkiksi **Virrake**-sovelluksen avulla voidaan lisätä interaktiivisuutta tietomalliin, kuten esimerkiksi aukeavia ovia, äänestysnappeja erilaisten skenaarioiden esittämiseen, pohjakarttoja, teleporttausta valittuun kohteeseen, erilaisia liikkumistapoja ynnä muita vastaavia toimintoja (Virrake s.a.). Granlund Managerin **Virtuaalinen kiinteistö** -työkalu käyttää yhdistelmätietomallia kiinteistön virtuaaliseen ylläpitoon ja hallintaan (Granlund s.a.).

buildingSMART Finlandin ja Suomen Sairaalatekniikan yhdistys ry:n toimeksiannoista vuonna 2016 tehtyjen kahden selvityksen mukaan tietomallintaminen alkaa yleistymään uudisrakennushankkeissa. Peruskorjaushankkeissa tietomallinnusta käytettiin harvemmin, lähinnä arkkitehtuurisuunnitteluun. (Kiviniemi 2017, 3-25.)

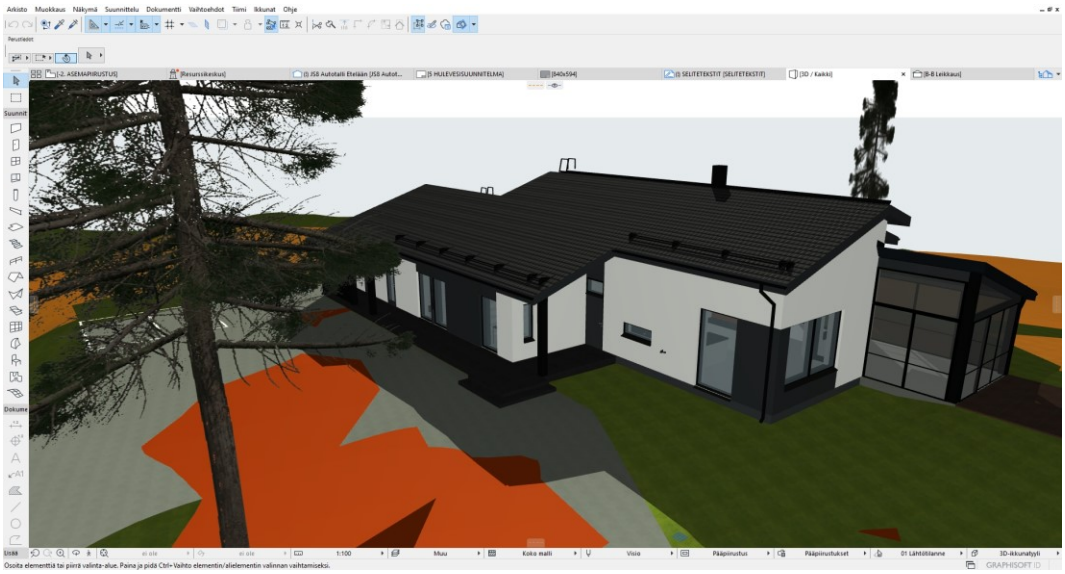
Tietomallien käytöstä on julkaistu ohjeistus **Yleiset tietomallivaatimukset YTV2012**. Hanke on perustunut tarpeelle määritellä ohjeet tietomallintamiseen eri tilanteissa ja kohteissa. Nopeasti kasvanut tietomallien käyttö rakennushankkeissa on lisännyt tarvetta ohjeistukselle. (buildingSMART 2020c; buildingSMART 2020c, osa 1).

Suomessa julkiset tai julkista rahoitusta saavat kotimaiset tai EU:n laajuiset rakennushankinnat edellyttävät usein tietomallin käyttöä (Hilma s.a.). Euroopan komission tukema EU:n BIM-työryhmä suosittelee jäsenmaitaan käyttämään tietomallintamista rakennushankkeissa esimerkiksi kustannustehokkuuden, toiminnan yhtenäistämisen, ympäristönsuojelun ja muiden vastaavien seikkojen myötä. (EU Bim Task Group 2016, 4.)

2.2 Mallinnus

Kiviniemen (2017, 14) mukaan eri suunnittelualojen mallinnuksien lisäksi voidaan mallintaa muitakin kohteen vaatimia komponentteja, kuten erilaisia teknisiä laitteita. Tietomallinnuksessa **ARK** tarkoittaa arkkitehtimallia ja **RAK** tarkoittaa rakennusmallia. **LVI**-, eli lämpö, vesi ja ilmanvaihto ja **S**, eli sähkösuunnittelu yhdistetään usein talotekniikan suunnitteluksi, eli **TATE**:ksi. (buildingSMART 2020c, osa 1.)

Mallinnuksen 3D-geometria voidaan toteuttaa esimerkiksi splini-käyrinä tai -pintoina ja polygoniverkkoina kohteen vaatimusten ja suunnittelijan mieltymysten mukaisesti.



Kuva 2. Tietomallin 3D-geometria ArchiCAD-ohjelmassa (ArkVisio 2019)

Yleisesti hieman suunnitteluohjelman mukaan origo tai object origin määrittää koko mallinnuksen tai tietomallin nollakohdan, eli kohdan missä mallipohjan x-, y- ja z-arvot ovat nolla. Origon sijainti voi olla määritelty niin kuin suunnittelija haluaa. Se voi olla esimerkiksi mallinnuksen keskellä tai jossain reunassa. Kussakin mallinnuksen tai tietomallin geometrisessa objektissa on pivot-piste (engl. *Pivot Point*). Se määrittää kunkin objektin määritellyn keskipisteen, minkä mukaan valittu objekti liikkuu, skaalautuu ja pyörii. (Gumster 2009; Blender s.a.)

ArkVisio Oy on tallentanut tietomalleihin tulevat komponentit, kuten esimerkiksi ikkunat ja ovet keskitettyyn pilvipalvelussa olevaan kirjastoon. Näin ollen jokaiselle suunnitteluun käytettävälle tietokoneelle ei tarvitse tallentaa komponentteja erikseen ja tieto pysyy ehyenä.

Mallinnukseen lisättävillä elementeillä voidaan kuvata tilan käyttötarkoitusta. Asuinhuoneiston mallinnuksessa olevat huonekalut ja kodinkoneet luovat konkreettisen esimerkin lisäksi visuaalisen assosiaation tilan käyttötarkoituksesta. Samoin esimerkiksi tehdassalin erilaiset koneet ja laitteet luovat virtuaalisen vaikutelman todellisuudesta. Erilaisilla sisällöllisillä mallinnuksilla voidaan havainnoida erilaisia tilaratkaisuja ja mahdollisia muutoksia tarpeen niin vaatiessa, ilman fyysistä mittaamista ja kokeilua. (Luku 2.)

Tekstuureilla ja heijastuksilla luodaan malliin visualisointi todellisista materiaaleista ja pinnoista. Valaistuksella voidaan esimerkiksi korostaa yksityiskohtia tai simuloida haluttua ajankohtaa tai muuta tilannetta. Simuloitua rakennusta on mahdollista muokata tavoin, mikä ei todellisuudessa olisi välttämättä edes mahdollista tai ainakaan kovin helppoa. Lisäksi kaikki voidaan tehdä rakennuksessa, jota ei ole vielä edes aloitettu rakentamaan tai jota ei välttämättä edes tulla rakentamaan. (Luku 2.)

Mallinnukseen soveltuvia ohjelmia on markkinoilla paljon. On avoimen lähdekoodin ilmaisia ja suljetun lähdekoodin kerta- tai kuukausimaksullisia. Osa ohjelmista on suunnattu enemmän tai vähemmän harrastajille, ollen yleiskäyttöisiä. Blender on ilmainen ohjelmisto, jolla on mahdollista toteuttaa hyvinkin monimutkaisia toteutuksia aina staattisesta 3D-mallinnuksesta renderöityyn 3D-animaatioon asti (Blender s.a.).

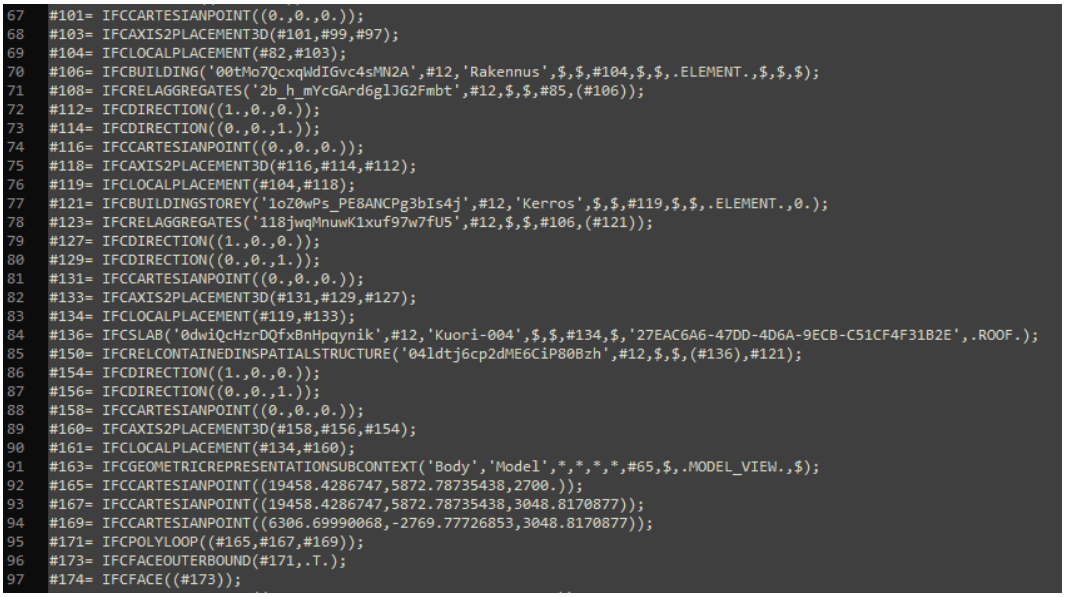
Osa ohjelmista on suunnattu suoraan ammattilaisille ja jopa erityiseen määriteltyyn käyttöön. Esimerkkinä ArchiCAD on suunniteltu tietomallipohjaiseen arkkitehtisuunnitteluun. Kukin suunnitteluala toki vaatii myös vähintään aiheen perusteiden osaamisen mallinnuksen toteuttamisen kannalta. Visuaalisen mallinnuksen toteuttaminen vaatii usein ainoastaan ulkoasun näyttävän halutulta, mutta esimerkiksi rakennustekniikan osamallin toteuttaminen vaatii tekijältään usein ko. alan koulutuksen tai vastaavan käytännön kokemuksen.

Ohjelmien perusteet on mielestäni usein helppo oppia, mutta vaativampi käyttö vaatii harjaannusta ja syvempää oppimista. 3D-suunnittelun ja -mallintamisen perusteiden ja matemaattisen teorian ymmärtäminen edes auttavalla tasolla helpottaa tajuamaan kokonaisuuden lisäksi 3D-mallinnusta käytännön toteutuksessa.

2.3 IFC-tiedosto

IFC-tiedosto (Industry Foundation Classes) on XML-pohjainen ISO-standardoitu (16739) tiedostomuoto, jonka alun perin kehitti IAI - the International

Alliance for Interoperability, joka nykyään tunnetaan buildingSMART-järjestönä. Tiedostomuoto mahdollistaa rakennusosapohjaisen tiedonsiirron CAD-ohjelmien välillä. IFC-tieto tallennetaan parametreina, geometriana tai molempina. IFC-metatiedon tunnistaminen ja käsittelykyky vaihtelee ohjelmistoittain. Jotkin ohjelmat ainoastaan lukevat tai tallentavat IFC-tietoa. 3D-mallin laskennan toteutus vaihtelee myös eri ohjelmittain. (M.A.D. 2013).



```
#101= IFCCARTESIANPOINT((0.,0.,0.));
#103= IFCAXIS2PLACEMENT3D(#101,#99,#97);
#104= IFCLOCALPLACEMENT(#82,#103);
#106= IFCBUILDING('00tMo7QcxqWdIGvc4sMN2A',#12,'Rakennus',$,$,#104,$,$,.ELEMENT.,$,$,$);
#108= IFCRELAGGREGATES('2b_h_mYcGArd6glJG2Fmbt',#12,$,$,#85,(#106));
#112= IFCDIRECTION((1.,0.,0.));
#114= IFCDIRECTION((0.,0.,1.));
#116= IFCCARTESIANPOINT((0.,0.,0.));
#118= IFCAXIS2PLACEMENT3D(#116,#114,#112);
#119= IFCLOCALPLACEMENT(#104,#118);
#121= IFCBUILDINGSTOREY('1oZ0wPs_PE8ANCPg3bIs4j',#12,'Kerros',$,$,#119,$,$,.ELEMENT.,0.);
#123= IFCRELAGGREGATES('118jwqMnuwK1xuf97w7fU5',#12,$,$,#106,(#121));
#127= IFCDIRECTION((1.,0.,0.));
#129= IFCDIRECTION((0.,0.,1.));
#131= IFCCARTESIANPOINT((0.,0.,0.));
#133= IFCAXIS2PLACEMENT3D(#131,#129,#127);
#134= IFCLOCALPLACEMENT(#119,#133);
#136= IFCSLAB('0dwiQcHzrDQfxBnHpqynik',#12,'Kuori-004',$,$,#134,$,'27EAC6A6-47DD-4D6A-9ECB-C51CF4F31B2E',.ROOF.);
#150= IFCRELCONTAINEDINSPATIALSTRUCTURE('04ldtj6cp2dME6CiP80Bzh',#12,$,$,(#136),#121);
#154= IFCDIRECTION((1.,0.,0.));
#156= IFCDIRECTION((0.,0.,1.));
#158= IFCCARTESIANPOINT((0.,0.,0.));
#160= IFCAXIS2PLACEMENT3D(#158,#156,#154);
#161= IFCLOCALPLACEMENT(#134,#160);
#163= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body','Model',*,*,*,*,#65,$,.MODEL_VIEW.,$);
#165= IFCCARTESIANPOINT((19458.4286747,5872.78735438,2700.));
#167= IFCCARTESIANPOINT((19458.4286747,5872.78735438,3048.8170877));
#169= IFCCARTESIANPOINT((6306.69990068,-2769.77726853,3048.8170877));
#171= IFCPOLYLOOP((#165,#167,#169));
#173= IFCFACEOUTERBOUND(#171,.T.);
#174= IFCFACE((#173));
```

Kuva 3. IFC-tiedoston sisältöä STEP-muodossa

IFC-tiedostosta on useita erilaisia formaatteja. Tässä yhteydessä esitellään kolme: .ifc, .ifcXML ja .ifcZIP. Ne eroavat toisistaan rakenteen suhteen. Esimerkiksi .ifc-rakenteessa tieto tallennetaan riveittäin, ns. STEP menetelmällä (kuva 3). #-merkki ilmaisee tunnistenumeron. Sen jälkeen esitetään luokan nimi. Attribuutit on rajattu sulkein ja erotettu toisistaan pilkuilla. IFC-standardissa on määritelty pakolliset luokat. XML-muodossa (Extensible Markup Language) tallennettava .ifcXML-tiedostomuoto muodostaa XML-standardin muotoisen tiedoston ja on kooltaan edellistä suurempi. ZIP-muoto on pakattu versio edellisistä. Pakkaus pienentää tiedoston kokoa huomattavasti. (buildingSMART 2020a.)

Periaatteessa IFC-tiedosto on ratkaisu tietomallin haluttujen tietojen siirtämiseen eri ohjelmien välillä. IFC-tiedosto sopiikin ennemmin tiedonsiirtoon kuin

itse tiedon tallentamiseen. Lisäksi alkuperäisen tietomallin tiedoston konversioissa IFC-muotoon voi osa informaatiosta hävitä, kuten ennalta määritelty pintamateriaali tai johonkin tiettyyn tietokenttään syötetty tieto. (Kiviniemi 2017, 9-13.)

2.4 Metatieto

Tietomalliin liittyvän metatiedon avulla tiedostoon voi tallentaa geometrian lisäksi muuta tietoa. Metatietojen avulla voidaan kuvata rakennettavan kohteen eri komponenttien ominaisuuksia. Esimerkiksi voidaan määritellä materiaali, mitat ja massa. (Nevalainen 2016, 7.)

Metatieto on sisällytetty tietomalliin, esimerkiksi IFC-tiedostossa tai erillisenä ulkoisena tiedostona muuta tiedostotyyppiä käytettäessä. Erillinen IFC-metatiedot sisältävä tiedosto voi olla XML-standardin tyyppinen, jolloin sitä voidaan hyödyntää esimerkiksi pelkän geometrisen tiedon sisältävän FBX-tiedostomuodossa olevan mallinnuksen yhteydessä. Metatiedon voi tallentaa käytettävään laitteeseen paikallisesti tai palvelimelle ulkoisesti niin tarvittaessa. Yleisesti metatiedolla tarkoitetaan tietoa tiedosta.

Nevalaisen (2016, 72) mukaan: ”Metadataa kuvaavat dokumentit tallennetaan *IfcMetadata*-kokoelmaan ja geometriaa kuvaavat dokumentit *IfcGeometryData*-kokoelmaan.” Vaikka geometria ja rakennetieto ovatkin eri kokoelmissa, voidaan ne yhdistää *GlobalId*-attribuutilla. Lisäksi Nevalainen (mts. 73) toteaa: ”Rakenneosan geometria kuvataan *Geometry*-luokan avulla. Luokka sisältää attribuutit, joiden avulla rakenneosan geometria voidaan kuvata kolmioituna verkkona.”

Metatiedon ongelmana voi olla hankaluus saada ohjattua tai yhdistettyä ajankohtainen ja oikea tieto mallinnuksen vastaavaan elementtiin. Metatiedon elementeillä tulee olla muuttumaton yksilöivä tunniste, jolla se määritellään kuuluvaksi tiettyyn yhteyteen. Puuttuva tai rikkoontunut tunniste voi aiheuttaa ongel-

mia. Esimerkiksi pelimoottoriin tuodusta mallinnuksesta voi puuttua komponentteja tai se tunnistuu väärin. Periaatteessa metatietona voidaan merkitä ja tallentaa mitä tahansa dataa.

Metatietojen käyttö voi olla hankalaa, jos pelimoottori ei tunnista suoraan IFC-tiedostoa. Unreal Engine tukee natiivisti, sekä Datasmith-lisäosalla IFC-tiedostomuotoa ja Unity tukee erilaisten lisäosien avulla tai tallentamalla metatiedot erilliseen XML-tiedostoon. Unityyn voi tuoda mallinnuksen geometrian esimerkiksi DAE- tai FBX-muodoissa.

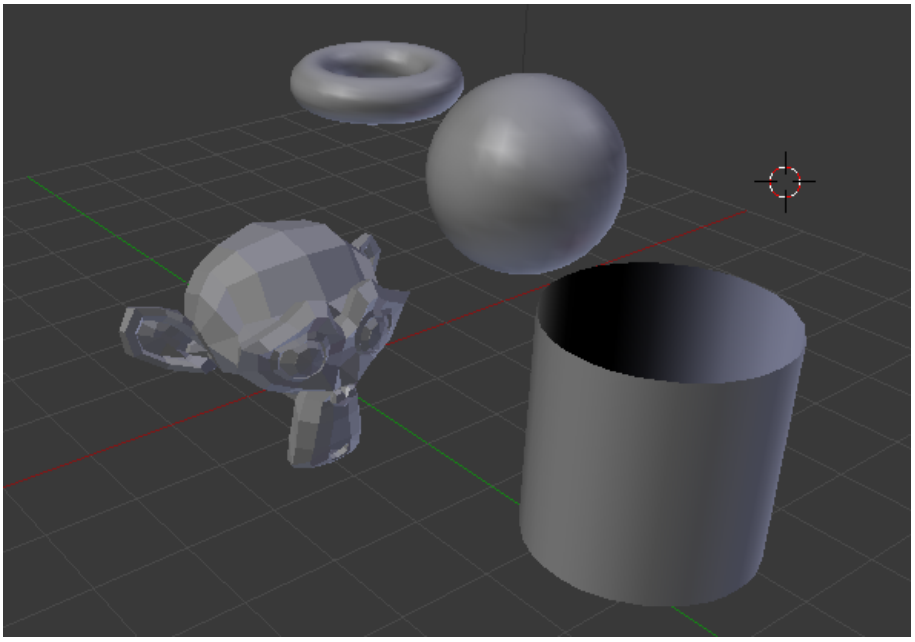
Nevalaisen (2016, 72-73) mukaan geometria- ja metatiedon kuvaamisessa käytetään hyväksi eri IFC-luokkia, joissa määritetyn rakenneosan ominaisuudet esitetään avain-arvo-pareina.

Tietomallin paketit ovat oliopohjaisia ja tarvittavia tietoja voi viedä ulkoisiin ohjelmiin. Erilaisia arviointeja voidaan suorittaa komponentin tarkkuudella. Esimerkiksi materiaali- ja energiakulutuksen arvoja voidaan laskea etukäteen. Hyperlinkeillä voidaan osoittaa määrättyihin www-osoitteisiin. (Garber 2014, 127.)

2.5 Splini-käyrät

Yleensä splini on **bézier**- tai **NURBS**-käyrä (engl. *spline*) tai -pinta (engl. *surface*), joka muodostetaan kolmen tai useamman pisteen kautta solmuvektorin avulla. Vektori voi sisältää tietoja eri arvoista, kuten esimerkiksi käyrän aloitus- ja lopetuskohdasta, pituudesta ja kaarevuuden asteesta. (Garber 2014, 93; Gumster 2009, 112.)

NURBS (engl. *Non-uniform rational basis spline*) -käyrä tai -pinta (kuva 4) muodostaa pintageometrian matemaattisilla funktioilla ja on yksi splini-käyrien erikoistyyppi (Botsch ym. 2010, 8). NURBS-käyrälle voidaan Garberin (2014, 150) mukaan asettaa painotettu kontrollipiste, jolla voidaan muokata käyrää tarkemmin kuin muilla splinityypeillä.



Kuva 4. Blenderin apinanpää-polygonimalli vasemmalla ja NURBS-pintaprimitiivejä vieressä

Gumster (2009, 121) kuvaa NURBS-käyrän järjestyksen vaikutuksen käyrän muotoihin. Järjestysluvultaan suurempi muodostaa pehmeämmän muotoisen käyrän kuin alempi.

Bézier-käyrä eroaa NURBS-käyrästä hieman rakenteensa vuoksi. Bézier-käyrällä muodostettu -ympyrä on arvio, kun taas NURBS-ympyrä on eksakti. **Poly spline** on käyrämuodoista yksinkertaisin. Kontrollipisteiden välillä ei tapahdu interpolointia. Poly spline luo tarkan mallin polygon meshin muunnoksessa käyräksi. (Blender s.a.)

Splini-käyrien ja -pintojen avulla CAD-ohjelman käyttäjän on helpompi luoda tasaisia muotoja. Alun perin autoteollisuuden tarpeisiin luotuna matemaattisena mallina splini on todenmukaisempi kuin monikulmioista koostuva polygonimalli. (Garber 2014, 93; Gumster 2009, 112.)

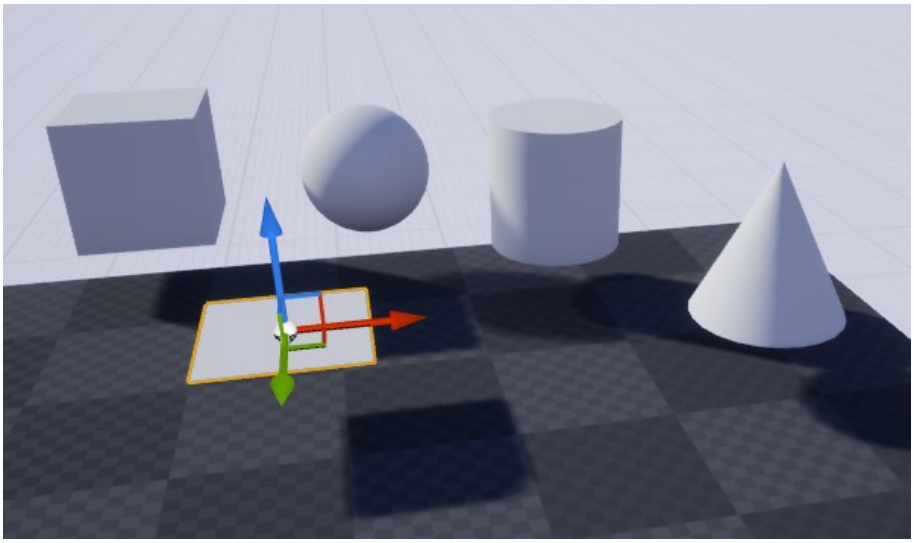
2.6 Polygonimalli

Polygonimalli (engl. *polygon mesh*) voi muodostua geometrisistä primitiiveistä (kuva 5), esimerkiksi pallo (engl. *sphere*), kuutio (engl. *cube*), kartio (engl. *cone*), sylinteri (engl. *cylinder*) ja taso (engl. *plane*). Monitahokas po-

lygonimalli puolestaan muodostuu monikulmioista (engl. *polygon*), jotka muodostuvat useammasta kärjestä (engl. yks. *vertex,* mon. *vertices*), sivusta (engl. *edge*) ja tahkosta (engl. *face*). Polygonimalli voi muodostua esimerkiksi kolmioista muodostuvasta verkosta (engl. *triangle mesh*). (Botsch ym. 2010, 86-87; Blender s.a.) Useamman kuin neljän kärjen polygonia voidaan kutsua myös käsitteellä **N-gon** (Blender s.a.).

Polyline tai **pline** on avoin polygoni, jota kutsutaan myös murtoviivaksi. Se voi sisältää suoria osuuksia sekä kaaria ja sen leveyden voi määritellä. (Home 2015, 53-58.)

Polygonimalli renderöityy reaaliaikaisesti nopeammin kuin splinimalli, mikä on tärkein syy polygonimallien käyttöön erityisesti peleissä ja muissa pelillistyksissä (Blender s.a.).



Kuva 5. Primitiivejä, Static Meshejä Unreal Engine 4

Static Mesh Unreal Enginessä ja **Mesh** Unityssä tarkoittavat staattista, ei-animoitua polygonimallia tai 3D-perusprimitiiviä. Edellä mainitut muodostavat suuren osan pelimaailman elementeistä ja niitä voi esimerkiksi liikuttaa, pyörittää ja skaalata. (Epic Games 2020a; Unity 2020a.)

Polygonimallin perustana oleva rautalankamalli muodostuu matriisiin tai vektoriin tallennettujen kärkien sijaintien yhdistämistä sivuista (kuva 6). Blenderissä apinanpää on erityinen testaukseen tarkoitettu malli (Blender s.a.)

Kuva 6. Blenderin Suzanne apinanpää rautalankamallina (engl. *wireframe model*)

Subdivision-toiminnolla saadaan tasoitettua polygonimallin ulkoasua. Mallin hienontaminen suurentaa sen polygonimäärää. Näytönohjaimen on raskaampi laskea monimutkaisia malleja kuin yksinkertaisia. (Botsch ym. 2010, 11-13.)

Tekstuureilla (engl. *textures*) muodostetaan mallin näkyvät pinnat halutun kaltaisiksi ja simuloidaan määriteltyä materiaalia (Epic Games 2020a). Näin luodaan malliin todellisuutta vastaava näkymä. Materiaalin heijastuksen, värin ja läpinäkyvyyden voi määritellä haluamakseen. Lisäksi käytetään **varjostinta** (engl. *shader*) kolmiulotteisen illuusion luomiseksi. Yksinkertaistettuna varjostin on tietokonealgoritmi, mikä määrittelee miten värit käyttäytyvät materiaalissa. (Gumster 2009, 144.)

Normals tarkoittaa polygonin (tai splinin) pinnasta suorassa kulmassa lähtevää suuntaa tai linjaa. Sillä määritetään pinnan etupuoli. **Render baking** -tekniikalla tarkoitetaan tapaa millä niin sanotusti paistetaan varjoja, valaistuksia ja tekstuureja eri tavoin, mikä osaltaan nopeuttaa lopullisen sovelluksen renderöintiä. (Blender s.a.)

Botschin ym. mukaan (2010, 64) esimerkiksi **normal-mapping** -tekniikassa käytettävään tekstuuriin on tallennettu korkearesoluutioinen kaksiulotteinen kuvainformaatio pakattuna. Erilaisilla algoritmeilla voidaan myös optimoida polygonimallia vähentämällä turhien polygonien määrää (mt.).

Polygonien määrä mesh-mallissa määrittelee pääsääntöisesti renderöinnin nopeuden. Näytönohjaimen laskentateho määrittelee sen kyvyn laskea ja piirtää reaaliaikaista 3D-mallinnusta. (Steenberg 2018.)

3 PELILLISTÄMINEN

Pelillistämisen tavoitteena on saada lisättyä tietomalliin kaksisuuntaista interaktiivista sisältöä. Mallinnuksessa luotaisiin liikkumisen lisäksi käyttäjän mahdollisuus vuorovaikutukseen mallin tietosisällön kanssa. Esimerkiksi ikkunaa klikkaamalla voitaisiin saada esille sitä vastaava IFC-metatieto. Lisäksi malliin olisi mahdollista dokumentoida yksityiskohtien muuttuneita arvoja, kuten esimerkiksi ikkunanpuitteiden huollon ajankohta, tekijä ja hinta.

Lisäksi olisi mahdollista lisätä tietomalliin ulkopuolisia tiedostoja. Esimerkiksi kuitti tai valokuva kuhunkin yksityiskohtaan kohdennettuna. Kunkin rakennuselementin, laitteen tai muun tarvikkeen toimittajan, asentajan, maahantuojan tai valmistajan verkkosivujen osoitteet löytyisivät myös tarvittaessa metatietojen kautta. Tarkan valmistuserän seurannan kautta mahdolliset laatupoikkeamat voitaisiin korjata ennen suurempaa vahinkoa. Jos esimerkiksi vaikkapa jonkin tuotantoerän teräspalkin lujuuksista löytyy epäkohtia, voitaisiin kriittiset rakenteet korjata ennen vahinkoa. (Luku 2.3.)

Optimoimattoman, splini-käyristä muokatun polygoneista muodostuvan mallin renderöinti on huomattavan raskasta jopa tehokkaalle tietokoneelle, koska muunnoksesta tulee usein monimutkainen. Jokainen lisääntyvä piste lisää kumulatiivisesti näytönohjaimen kuormaa. (Luku 2.6.)

Pelillistetyssä toteutuksessa voi olla realistinen tai sovellettu fysiikkamallinnus. Hahmo voi olla näkyvä ns. kolmannen persoonan näkökulman (engl. *3rd person view*) hahmo, jossa koko hahmo näkyy usein takaapäin kuvattuna tai ensimmäisen persoonan näkökulman (engl. *1st person view*) hahmo, jossa näkyvät esimerkiksi vain hahmon kädet. Hahmo voi olla myös oletettu näkökulma katsojan silmistä ilman mitään mallinnettuja fyysisiä kehon ulokkeita.

Pelkistetty taso voi kuvata maastoa, jossa tietomalli sijaitsee. On syytä määritellä pohjatason reunoille (näkymättömät) kulkuesteet, jotta käyttäjän hahmo ei putoa fysiikkamoottorin vaikutuksesta pois mallin tasalta. Maaston voi rakentaa myös luonnollista vastaavaksi niin halutessaan. Lisäksi erilaisia luonnonilmiöitä voi halutessaan liittää osaksi mallinnusta. Joskin ne todennäköisesti aiheuttaisivat ainoastaan turhaa kuormitusta ohjelmalle ja käyttöympäristölle ilman mainittavaa etua.

Lähiverkossa tai varsinkaan päätelaitteessa sijaitsevan tiedoston koko ei ole niin määrittävä tai rajoittava tekijä, kuin internetin yli ladattaessa. Toki tiedostokoon optimointi on aina tärkeä osa pelillistämisen toteutusta.

3.1 Pelillistettävä tietomalli

Tietomallin pelillistäminen toteutetaan joko samalla ohjelmalla kuin mallinnus tai vaihtoehtoisesti pelimoottorilla. Mallinnus täytyy muuntaa suunnitteluohjelmasta pelimoottorin ymmärtämään muotoon (Edwards ym. 2015).

Tietomallin tulee sisältää kaikki ne tarvittavat osa-alueet, joita lopputuloksessa tarvitaan. Mallinnuksen on syytä olla lopullisessa muodossa ennen pelillistämisen toteutusta, koska mahdolliset virheet voivat estää määritellyn toiminnan tai ulkoasun. Mallin korjaaminen voi olla mahdotonta pelillistämisvaiheessa.

Tietomallin geometria muunnetaan polygonimalliksi ja metatieto siirtyy joko tiedoston sisäisesti tai erillisessä tiedostossa.

Mallinnuksen käytössä on huomioitava eräitä seikkoja. Useampikerroksisessa rakennuksessa on päästävä toisiin kerroksiin. On päätettävä mitkä elementit ovat läpikuljettavia, kuten esimerkiksi ovet ja muut aukot. Mallinnus voi olla määritelty myös niin, että ovet aukeavat ohjelmoidusti hahmon niitä lähestyessä. Portaat on käytännössä oltava määritelty kiinteäksi niin, ettei hahmo putoa niiden läpi kuten myös sellaiset taso, joilla hahmon olisi syytä pystyä liikkumaan. Hahmon voi tarvittaessa määritellä liikkumaan myös ilman fysiikkamallinnusta.

DWG-, IFC- tai ohjelman oma natiivitiedosto siirretään suoraan pelimoottoriin, jossa interaktiiviset toiminnot toteutetaan vaatimusmäärittelyn tai käyttötapausten mukaisesti.

Lisäoptiona pelillistetyn tietomallin voisi sijoittaa esimerkiksi eräänlaiseen virtuaaliseen kaupunkiin, jonne suunnittelija tai arkkitehtitoimisto voisi sijoittaa preferenssirakennuksiaan esiteltäväksi. Asiakkaat voisivat omilta päätelaitteiltaan tutkia rakennuksia ja niiden ominaisuuksia monipuolisesti.

3.2 ArchiCAD

ArchiCAD on Graphisoftin tietomallisuunnitteluohjelma, josta tässä aineistossa on käytetty versiota 23 Edu, eli opiskelijalisenssiä. Opiskelijalisenssillä toteutettua tai muokattua tietomallia ei voi enää konvertoida kaupalliseksi. (Graphisoft 2020.)

BIMx-pelillistäminen ArchiCadillä onnistuu ohjelman omalla hypermallin julkaisu -toiminnolla. Hypermalli on tarkoitettu enemmänkin mm. tietomallin 3D-geometrian, metatietojen ja piirustusten esittelytarpeeseen, kuin interaktiiviseen käyttö- ja huolto-ohjeeseen soveltuvaksi.

Mallinnuksesta tulee melko massiivinen. Noin 20 Mt IFC-tiedostosta tulee melkein 320 Mt:n kokoinen BIMx-malli (kuva 7).

Kuva 7. ArchiCADin BIMx-pelillistys (ArkVisio 2019)

ArchiCADin BIMx-tietomallin näyttöohjelma toimii Windows- ja macOS-pohjaisissa tietokoneissa. Se toimii lisäksi sekä Android-, että iOS-pohjaisissa älypuhelimissa. Se sallii pelillistetyssä kohteessa liikkumisen kävellen tai lentämällä. Ohjelmassa voi muun muassa mitata etäisyyksiä ja valita kunkin IFC-elementin ääriviivoineen näkyväksi.

Mallinnuksessa ei esiinny samanlaisia geometria- tai tekstuurivirheitä kuin pelimoottoreihin viedyissä versioissa. Koska ArchiCADin hypermalli käyttää ohjelman natiivia tiedostomuotoa, kaikki elementit ja tekstuurit toistuvat alkuperäistä tietomallia vastaavasti. Tietomallinnuksen sijainti suhteessa pohjan origoon ei vaikuta lopullisen hypermallin toimivuuteen tai käyttöön.

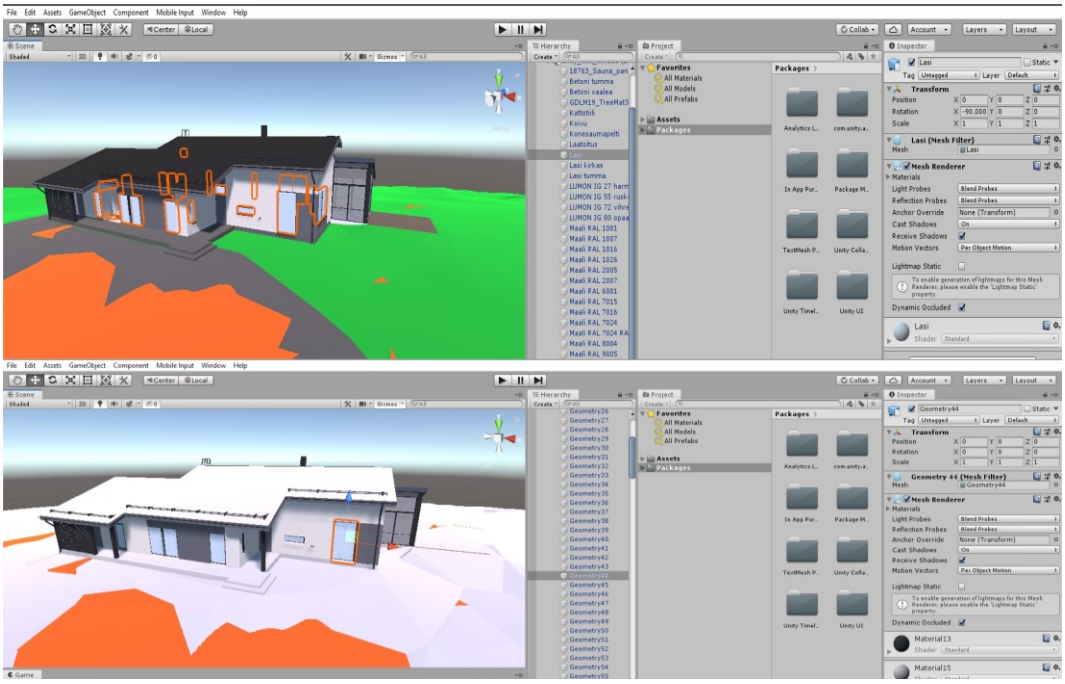
3.3 Unity

Unity-pelimoottorista on tässä aineistossa käytetty versiota 2019.1.8f1 (Unity 2020b).

Unity käyttää C#-kieltä ohjelmointiin. Unityssä käyttäjän peliobjektin törmäys voidaan laukaista hahmon osuessa esimerkiksi määriteltyyn törmäyslaatikkoon. Niin sanottuun meshiin, eli tässä työssä tarkoitettuun tietomallin 3D-geometriaan tai sen osaan voidaan luoda erilaisia törmäystunnistimia ja asettaa ne esimerkiksi kiinteiksi. Sen perusteella pelimoottori tunnistaa esimerkiksi oviaukot ja portaat määritellysti. Tällöin hahmo pystyy esimerkiksi liikkumaan eri tasoilla, eikä putoa kiinteäksi määrättyjen tasojen läpi. Seinien läpi kulkeminen ei onnistu myöskään, ellei niin ole jostain syystä määritelty.

ArchiCADista tuotu FBX-malli hävittää metatunnisteet ja kaikki samankaltaiset komponentit saavat yhteisen tunnisteen. Collada-muodossa tuotu DAE-malli erottelee komponentit Geometry + *juokseva numerointi* -tunnisteen alle (kuva 8). Käytännön eroa ei ilmennyt viennissä ArchiCADin natiivin PLN- tai standardoidun IFC-tiedostomuotojen välillä.

Unityyn tuotu mallinnus oli jossain vaiheessa menettänyt osan *IfcSite-ympäristökomponentin* maastoa kuvaavasta meshistä. Käytännössä tämä tarkoitti sitä, että mallinnusta ei sellaisenaan voisi välttämättä käyttää käyttö- ja huolto-ohjeen perustana, ainakaan ilman muokkausta.



Kuva 8. FBX- ja DAE-tiedostojen komponenttien erot (ArkVisio 2019)

Tekstuurit toistuvat eri 3D-tiedostotyypeissä hieman eri tavalla. FBX-tuonnissa ikkunoista saattaa puuttua läpinäkyvyys. Se on helppo korjata valitsemalla kaikki lasikomponentit ja vaihtamalla niiden materiaaliksi lasi.

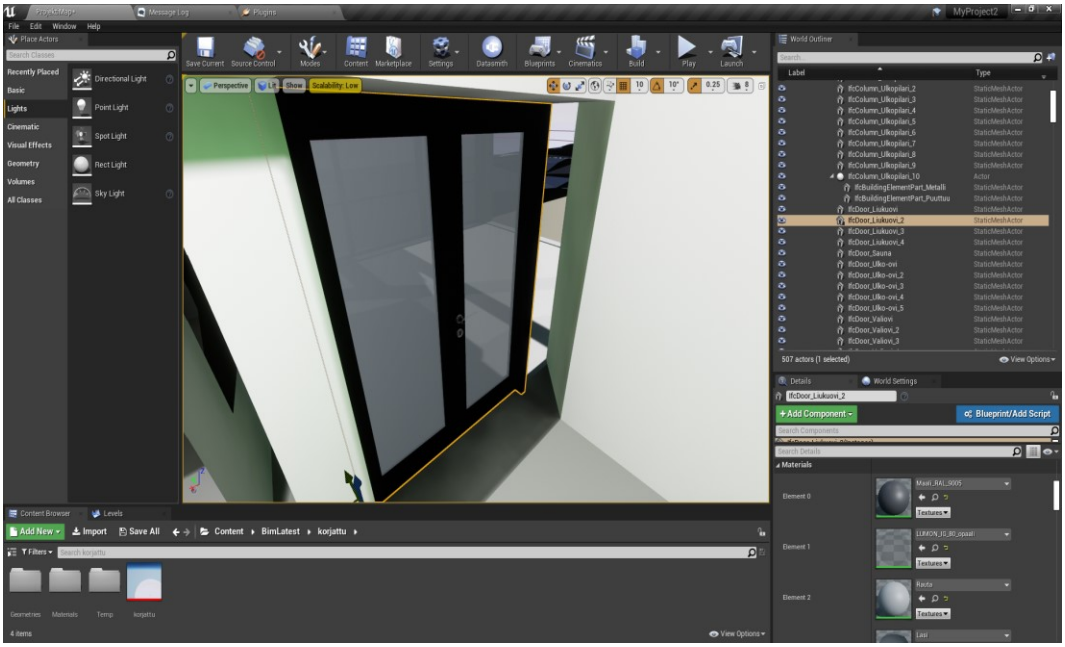
IFC-metatiedot sijoitetaan XML-tiedostoon, josta ne ovat tarvittaessa haettavissa. Kahden eri tiedoston yhteensovittamisessa on omat ongelmansa. Unityyn tuotu XML-tiedosto oli melko suuri ja sen operoiminen oli mielestäni haastavaa. Unitylla on kuitenkin helppo luoda www-selaimessa toimiva WebGL-toteutus. Jos IFC-metatietojen käsittely olisi käyttäjäystävällisempää, Unity olisi luonteva vaihtoehto käyttö- ja huolto-ohjeen www-palvelimella toimivan sovellusversion toteuttamiseen.

3.4 Unreal Engine

Epic Gamesin tarjoama pelimoottori on Unreal Engine, josta tässä aineistossa on käytetty versioita 4.25.1 sekä 4.25.3 (Epic Games 2020c).

Unreal Engine käyttää C++ ja Blueprints visual scripting -kieliä ohjelmointiin. Unreal Engine 4.24 -versiosta lähtien on poistettu ohjelmaan aiemmin sisällytetty HTML5-toteutus, mutta se on saatavissa uusiin versioihin ladattavana lisäosana.

IFC-tietomalli tuodaan Unreal Engineen natiivisti tai Datasmith-työkalu- ja lisäosakokoelman avulla. Unreal Enginen Datasmith-työkalu tukee monia eri tiedonsiirtoformaatteja ladattavien lisäosien avulla.



Kuva 9. UE4 IFC-tiedot ovesta, sekä hajonnut oven vierustan geometria (ArkVisio 2019)

Unreal Enginessä käyttäjän peliobjekti tunnistaa törmäyksen esimerkiksi niin sanottuna päällekkäisyytenä (engl. *overlap*) esimerkiksi laatikon muotoiseen törmäystunnistimeen (engl. *box collider*). Törmäyksentunniste voi olla esimerkiksi edellä mainittu laatikko (engl. *box*), kapseli (engl. *capsule*) tai pallo (engl. *sphere*). (Epic Games 2020a.)

Koska Unreal Engine on täysverinen pelimoottori, joka tunnistaa IFC-tiedostotyypin suoraan, on valintani käyttää Unreal Engineä käyttö- ja huolto-ohjeen toteuttamiseen melko selvä.

3.5 Muita ohjelmia

Erilaisia suunnittelu-, mallinnus- ja palvelinsovelluksia on runsaasti. Osa on ilmaisia, osa kertamaksullisia ja osa aikaperusteisella maksulla varustettuja.

Maksuttomia ohjelmistoja ovat mm:

- **IfcOpenShell** on avoimeen lähdekoodiin perustuva ohjelmakirjasto IFC-tiedostojen käsittelyyn eri ohjelmilla. Esimerkiksi Blenderiin löytyy IFC-tuonnin lisäosa **IfcBlender**. (IfcOpenShell 2020.)
- **BIMserver** on yhteisövetoisen niin ikään avoimeen lähdekoodiin perustuva tietomallipalvelin (BIMserver.org s.a.).

- **Blender** on avoimeen lähdekoodiin perustuva ilmainen, mutta erittäin kattava 3D-mallinnusohjelma moneen käyttöön, esimerkiksi 3D-sisällön luomiseen, mallinnukseen, renderöintiin ja animointiin. (Blender s.a.)

Autodeskin ohjelmistoja ovat mm:

- **Autodesk 3ds Ma**x on 3D-mallinnus- ja renderöintiohjelmisto.
- **Maya** on ohjelmisto 3D-animointiin, mallinnukseen, simulointiin ja renderöintiin.
- **Revit** 3D-tietomallinnusohjelmisto useiden ammattialojen koordinoituun suunnitteluun.
- **AutoCAD** on 2D- ja 3D-ohjelmisto CAD-suunnitteluun. (Autodesk 2020.)

Trimblen ohjelmistoja ovat mm:

- **Tekla Structures** on rakennesuunnitteluun ja tietomallintamiseen suunnattu ohjelmisto (Tekla s.a.).
- **SketchUp** on helppokäyttöinen ja käyttäjäystävällinen 3D-suunnitteluohjelmisto (Trimble s.a).

Muita ohjelmistoja ja lisäosia ovat mm:

- **Solibri** tarjoaa osin maksuttomia ja osin maksullisia sovelluksia eri alojen tietomallien hallintaan. Solibria käytetään esimerkiksi tietomallien yhdistämiseen, tarkistamiseen ja tiedonvälitykseen. (Solibri 2020.)
- **Twinmotion** on Epic Gamesin, esimerkiksi arkkitehtisuunnittelun 3D-visualisointiin suunnattu ohjelmisto (Epic Games 2020b).
- **Tridify BIM Tools for Unity** on lisäosa Unitylle 3D CAD ja BIM -mallien tuomiseen metatietoineen (Tridify 2020).
- **Pixyz Plugin for Unity** on toinen Unitylle tarkoitettu lisäosa 3D CAD ja BIM -mallien tuomiseen. Pixyz tarjoaa samoin toimivaa lisäosaa myös Unreal Enginelle. (Pixyz 2020.)
- **Virrake** on jo aineistossa mainittu Kaakkois-Suomen Ammattikorkeakoulun Virrake-hankkeessa kehitetty ohjelmisto (Luku 2.1).

Ohjelmia on siis joka lähtöön eri tasoille ja eri tarpeisiin. Ammattitason ohjelmistot on suunnattu jo hintansa perusteella kaupalliseen tai muuhun vaativaan käyttöön. Opiskelijoille on usein tarjolla ilmaisia tai vakiohintaa edullisempia opiskelijalisenssejä.

4 KÄYTTÖ- JA HUOLTO-OHJE

Tietomallin käyttö huoltokirjana edellyttää rakennuksen komponenttien tunnistamista tietomallissa. Tietokannasta luetaan komponentin huolto-ohje tai vastaava ja se voidaan tarvittaessa visualisoida. Tietomalli itsessään ei ole välttämättä oikea paikka säilyttää tietoa, vaan se tulisi sijoittaa erilliseen tietokantaan. (Halmetoja 2016.)

Rakennuksen 3D käyttö- ja huolto-ohjeen käyttäminen tulisi olla aika- ja paikkariippumatonta. Kukin rakennuksen kohde-elementti tunnistetaan määritellyllä tavalla. Kohteen ohje voi esimerkiksi avautua uuteen ikkunaan, jota voi siirtää tai sen läpinäkyvyyttä tai kokoa voi säätää. Erilaisen datan ja informaation, kuten kuvien tai tekstin lisääminen onnistuisi myös. Tietojen tulostus ja edelleen lähetys tarvittaessa on huomioitava. Käyttäjälle näkyvän datan määrä tulisi olla määriteltävissä ja sen tulisi skaalautua eri tarpeisiin.

Käyttöliittymän tulisi olla intuitiivinen ja selkeä. Tietomallin geometria, valitun komponentin infoikkuna ja tarvittavat painikkeet eri toiminnoille tulisi suunnitella palvelumuotoiluprosessina yksinkertaisiksi, mutta kattaviksi. Näkymässä voisi olla yksinkertaistettu käyttöliittymä ainoastaan perustoiminnallisuuksilla ja tarvittaessa erikseen valittuna laajempi, kaikkien työkalujen esitys.

Infoikkunan tulisi olla skaalautuva ja sen läpinäkyvyys olisi säädettävä. Infoikkunan sisältö koostuisi valitun komponentin metatiedoista ja käyttäjän valitsemista ja lisäämistä tiedoista. Infoikkunan käyttöliittymä koostuisi ainakin ikkunan sulkunapista (x), liukuvasta läpinäkyvyyden säädöstä, tietojen lisäyksestä, poistosta, päivityksestä, edelleen lähettämisestä laitteeseen asennettujen palveluiden avulla (esimerkiksi *One Drive, Google Drive, Facebook, WhatsApp*

tai muu vastaava) ja tulostuksesta. Toiminnot voisivat olla esimerkiksi hampurilaismenun tyylisen valikon alla ruutukoon ollessa pieni ja riittävällä ruudun leveydellä näkyä kokonaan.

Siinä missä talon elinkaari voi olla satoja vuosia, pitäisi käyttö- ja huolto-ohjeen olla käytettävissä yhtä lailla tulevaisuudessa. Käyttöjärjestelmäriippumattomuus, standardien hyväksi käyttäminen sekä avoimien tai päivityksiltään varmistettujen ohjelmistojen ja lisäosien käyttäminen tulisi olla perustana käytettäville ratkaisuille.

Jos ohjetta käytettäisiin virtuaalilaseilla, tulisi määritellä helppokäyttöiset eleet kunkin elementin tutkimiseen. Tarvittaessa voisi vaikka tarttua kiinni komponentista ja siirtää se erilleen rakennuksen muista osista ja purkaa palasiksi.

Vian haun apuna ohjelma voisi tallentaa tavanomaisesta poikkeavasti toimivan laitteen ääntä ja verrata sitä tietokannasta löytyvään verrokkiin. Jos ääni eroaa referenssistä huomattavasti, tulisi laite vaihtaa tai huoltaa, esimerkkinä ilmanvaihdon tuuletin tai lämpöpumppu.

Ohjelman kautta olisi mahdollisuus hakea tietoa keskitetysti rakenteista, laitteista, kojeista tai muista vastaavista suoraan valmistajan tai riippumaattoman referenssioperoijan hallinnoimasta lähteestä. Samalla toiminnolla ohjelman tekoäly kykenisi huomauttamaan käyttäjälle laitteen tai komponentin keskimääräisen huoltoajankohdan tai käyttöiän päättymisen ennakolta.

Tietoturva on tärkeä elementti ja on otettava huomioon jo suunnittelun alkuvaiheessa. On määriteltävä, kuinka suojataan tiedot ulkopuolisilta luotettavasti, ilman että menetetään tuotteen käytön helppous ja yleinen käyttökelpoisuus. Esimerkiksi mobiililaitteissa tunnistus tapahtuisi pin-koodilla, sormenjäljellä tai vastaavalla toiminnolla ja työasemalla esimerkiksi salasanatunnistuksella.

4.1 Vaatimusmäärittely

Jos tarvekartoitus puoltaa hankkeen suunnittelun aloittamista, tehdään tarkempi vaatimusmäärittely käyttö- ja huolto-ohjeelle. Vaatimusmäärittelyn tarkoitus on määritellä eri toiminnallisuudet, ei-toiminnallisuudet, rajoitteet ja niiden tärkeysjärjestykset. Etukäteen määrittely helpottaa prosessimallin valintaa ja projektin toteutusta. Tässä on kuvattu yleisluontoinen vaatimusmäärittely ilman priorisointia.

Taulukko 1. Toiminnalliset vaatimukset

Toiminnalliset vaatimukset
Tarvittava osamalli (ARK, RAK, S, LVI) on valittavissa yhdessä muiden kanssa tai erikseen.
Ohjelma toimii päätelaitteessa natiivina tai selaimessa WebGL/HTML5-toteutuksena.
Tietomalli on pelillistetty: liikkuminen tapahtuu kosketusnäytöllä, hiirellä, näppäimistöllä tai muulla tavalla ja interaktiiviset toiminnot ovat kaksisuuntaisia.
Tietomallin komponentit ovat tunnistettavissa hiiren klikkauksella tai näytön kosketuksella
IFC-tiedot ovat yhdessä tietomallin kanssa tai erillisessä tiedostossa www-palvelimella.
Tiedoston maksimikoko määritetään käyttötarpeen mukaan.
Tallennus toteutetaan päätelaitteeseen tai palvelimelle.
Toteutetaan back-end -ohjelma palvelimelle tiedon vientiin ja tuontiin tietokannasta, jos tietoja ei tallenneta paikallisesti ohjelman kanssa.
Ohjelma toimii vähintään 30 vuotta tai rakennuksen elinkaaren ajan ja on tarvittaessa siirrettävissä uuteen formaattiin sekä toimintaympäristöön.
Testaus on suunniteltava ja toteutettava virheiden löytämiseksi.

Taulukko 2. Ei-toiminnalliset vaatimukset

Ei-toiminnalliset vaatimukset
Tekstuurien ulkoasun on vastattava alkuperäistä mallia mahdollisimman tarkasti.

Materiaalit, esimerkiksi metalli, lasi, puu ym., tunnistuvat oikein.
Grafiikan taso on optimoitava, polygonien määrä on kuitenkin oltava riittävä tarvittavien yksityiskohtien erottamiseksi.
Vuodenaikojen, valaistuksen, ympäristön ja muiden vastaavien seikkojen olemassaolo voidaan määritellä optiona.
Äänet voidaan tarvittaessa määritellä.
3D-virtuaalikaupunki toteutetaan www-palvelimelle.

Taulukko 3. Rajoitteet

Rajoitteet
Käyttöoikeus malliin syntyy ainoastaan käyttäjän tunnistuksella.
Tulevaisuudessa käytettävät tekniikat voivat olla käytön este.

Etukäteen määrittely mahdollistaa projektin suunnitellun läpiviennin. Toteutetaan se sitten millä prosessimallilla ja vaihejaolla tahansa.

4.2 Toteutus

Päätimme toteuttaa käyttö- ja huolto-ohjeen Unreal Enginellä, sen valmiina olevien IFC-ominaisuuksien myötä. Toteutusprosessi on hieman samankaltainen kaikissa vaihtoehdoissa, joskin ArchiCADissa yksinkertaisin: tallennetaan tietomalli BIMx hypermalliksi. ArchiCAD tuottaa hypermallin automaattisesti napin painalluksella. Tuloksena on pelillistetty 3D-tietomalli metatietoineen.

Unity vaatii PLN- tai IFC-tiedoston muunnon FBX- tai DAE-tiedostomuotoon, sekä metatietojen viennin XML-tiedostoon. 3D-mallitiedosto ja XML-tiedosto tuodaan sen jälkeen Unityyn: *Assets -> Import New Assets*. Tuonnin jälkeen tehdään tarvittavat toimenpiteet pelillisyyden ja kaksisuuntaisen toiminnan lisäämiseksi, kuten Unreal Enginellä.

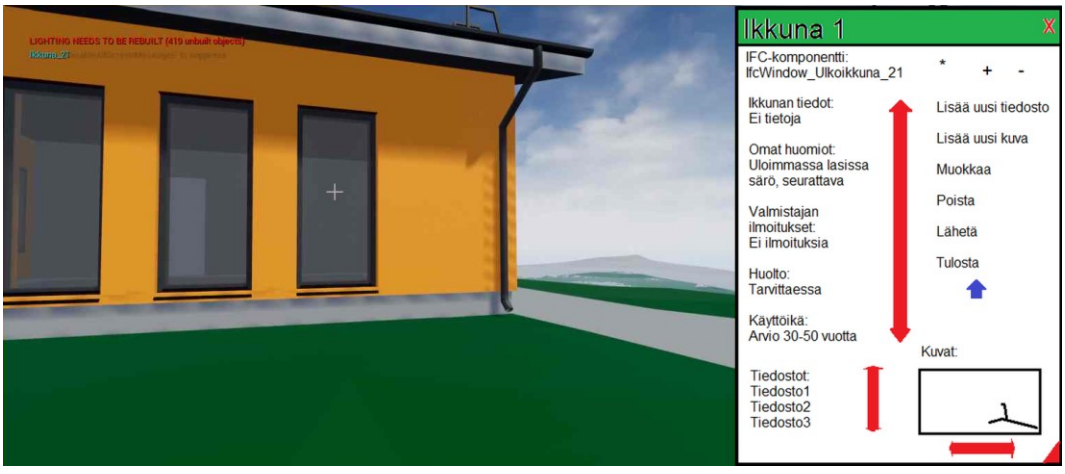
Unreal Engineen tietomallin natiivitiedosto viedään aluksi IFC-muotoon ja tuodaan natiivisti tai Datasmith-työkalulla pelimoottoriin. Sen jälkeen lisätään halutut interaktiiviset elementit valituilla työkaluilla.

Prototyypin luonti toteutettiin Unreal Engine -pelimoottorilla, koska sisäänrakennettu natiivi IFC-tuki ja Datasmith-työkalu helpottavat IFC-tiedostojen tuontia huomattavasti. Myös Unity olisi ollut luonteva vaihtoehto esimerkiksi maksullisilla Tridify BIM Tools for Unity tai Pixyz Plugin for Unity -lisäosilla varustettuna. Ilman näitä erillisen XML-tiedoston kanssa käyttö- ja huolto-ohjeen luominen vaikuttaa turhan työläältä.

Unreal Engineen voi tarvittaessa tuoda myös esimerkiksi AutoCADin tukemaa DXF/DWG-tiedostomuodossa olevaa aineistoa (Epic Games 2020a).

IFC-malli tuotiin Unreal Engineen Datasmith-työkalulla, sekä määriteltiin törmäystunnisteet ja käyttäjän interaktiivinen toiminta tietomallissa. Prototyyppivaiheessa määriteltiin yhden elementin törmäystunnistin, joka avaa pienen tietoikkunan pelinäkymään.

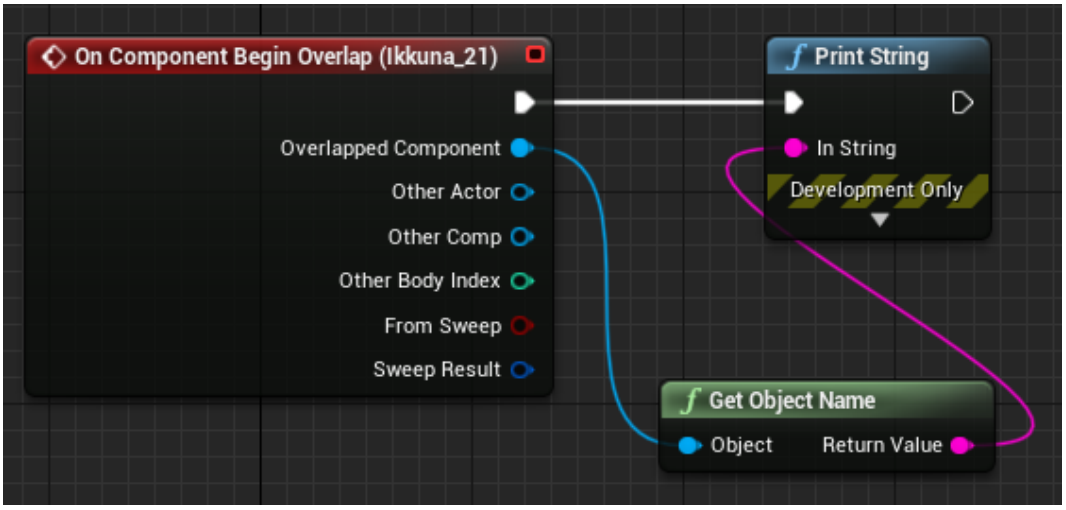
Punaisella värillä on merkitty vierityssuunta ja skaalauskohta. Sinisellä nuolella on merkitty sulkeutuva valikko, jonka saa avattua asteriskilla merkityllä näppäimellä. Vihreällä värillä merkitystä otsakkeesta ikkunaa voi liikuttaa pelinäkymässä. Plus- ja miinusnäppäimillä voidaan säätää ikkunan läpinäkyvyyttä. Punaisesta X-näppäimestä ikkuna sulkeutuu. IFC-komponentti näyttää tunnistetun komponentin mesh-nimen. Tietoikkunan elementit skaalautuvat kuvasuhteen tarpeiden mukaan. (Kuva 10.)



Kuva 10. Simuloitu elementin tunnistaminen ja tietoikkuna pelillistetyssä tietomallissa (ArkVisio 2019)

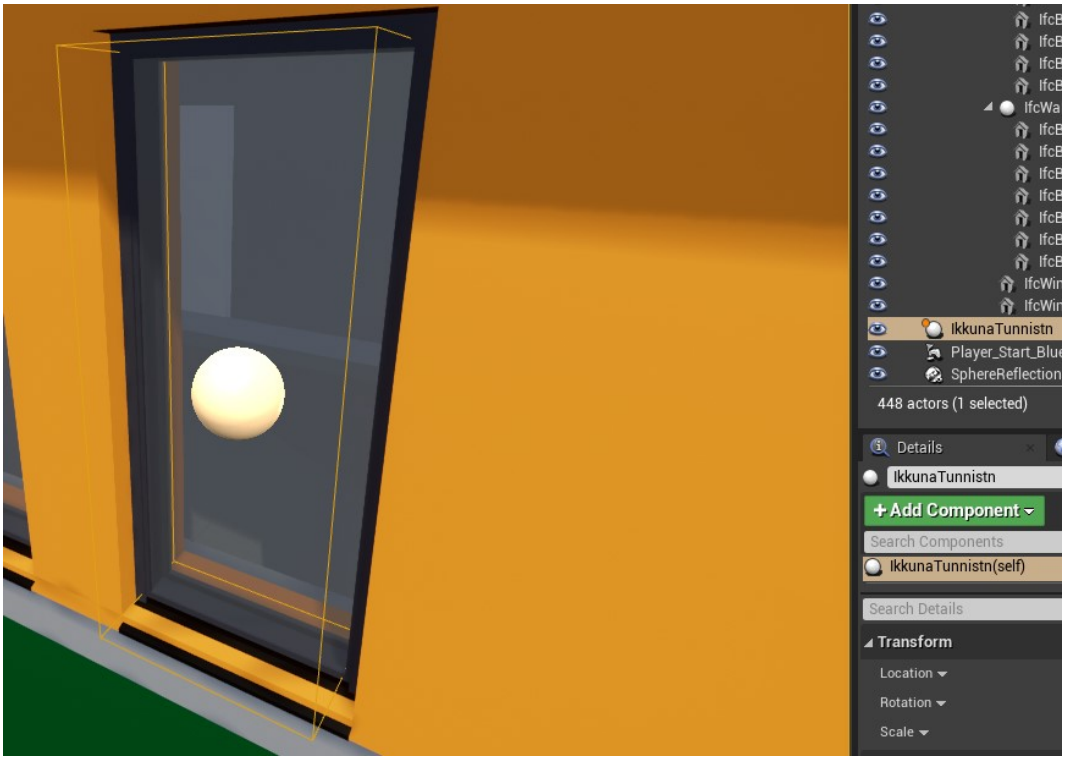
Mobiiliversiossa tietoikkuna voisi olla geometrianäytön alapuolella. Kokemukseni mukaan kädessä pidettävää mobiililaitetta pidetään yleensä pystysuunnassa. Ikkunan käyttöliittymäsisältö voisi olla sama kuin työpöytäversiossa, kunhan siitä toteutetaan responsiivinen, eli näyttöön skaalautuva. Käännettäessä mobiililaite vaakasuuntaan, ohjelma täyttää ruudun työpöydälle määritellysti.

Toteutin aluksi yhden törmäystunnistimen, joka niin sanotussa *overlap*-positiossa näyttää ruudulla törmäystunnistimen nimen (kuva 11).



Kuva 11. Yksinkertainen Blueprints-skripti törmäykselle

Unreal Enginessä hiiren kursorin suunnan voi määrittää ja käyttää sitä hyväksi halutun elementin kohdentamisessa valinnaksi. Hiiren kursorilla osoittamalla komponentin nimi näkyy ja klikatessa aukeaa tietoikkuna.



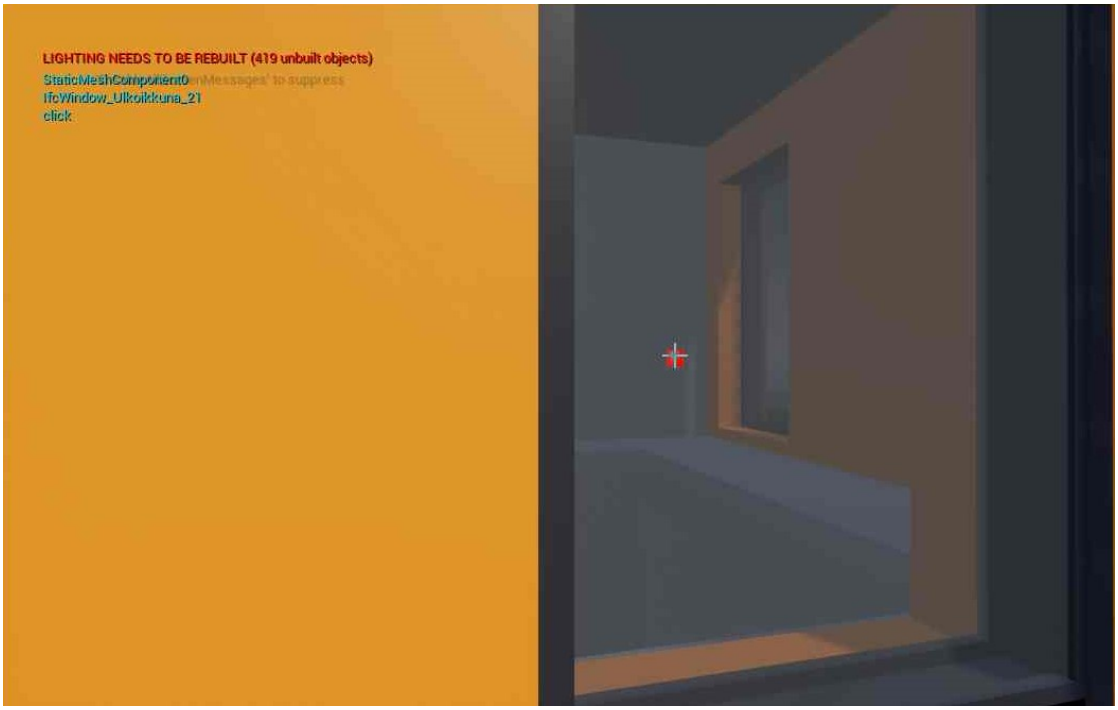
Kuva 12. Ikkunakomponenttiin yhdistetty törmäystunnistin (ArkVisio 2019)

Hiiren vasen nappi ei toiminut vakioasetuksilla. Loin *Default Modes*-valikossa uuden *GameModeBP*:n, ja poistin *Player_Start*-komponentista siihen lisätyn turhan Blueprints-skriptin. Player Controller Classin vaihdoin oma tekemäkseni Blueprints-pohjaiseksi kontrolleriksi ja hiiren napin painallus alkoi toimimaan. (Kuva 13.)

Kuva 13. Unreal Enginen Default Modes -valinnat

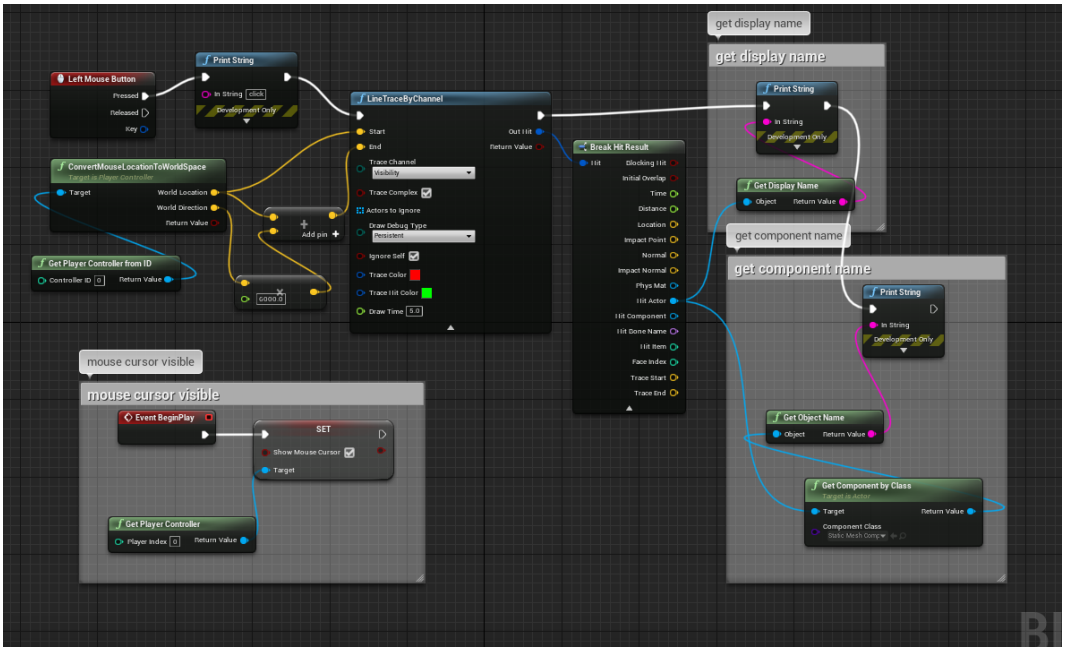
Unreal Enginen saa tarvittaessa tunnistamaan objektit hiiren klikkauksella (kuva 14). Testausvaiheessa käytettiin erilaisia toiminnallista havaittavuutta lisääviä ns. **debug**-ominaisuuksia, kuten esimerkiksi *Print String, Trace Chan-*

nel [Visibility] ja *Trace Complex [true].* Edellä mainitut näyttävät ruudulla esimerkiksi määritellyn tekstin ja vektorin suunnan ja kosketuspisteen. Prototyypissä liikkumiseen käytetään w-, a-, s- ja d-näppäimiä. Q-näppäin nostaa hahmoa ylöspäin ja e-näppäin laskee alaspäin. Vasen hiiren näppäin aktivoi kääntymisen ja palauttaa kursorin kohdalla olevan objektin tunnisteen. Oikea näppäin aktivoi pelkästään kääntymisen.



Kuva 14. Pelillistettyyn tietomalliin on lisätty objektien tunnistusskripti (ArkVisio 2019)

Tunnistettavat elementit voisi jakaa kokonaisuuksiksi: Katto, syöksyputket, ovet, ikkunat, ulkoseinät, sisäseinät, lattiat ja niin edelleen. Se helpottaisi kokonaisuuden hallintaa. Kokonaisuudet voisi jakaa edelleen yksilöllisemmiksi osiksi, kuten esimerkiksi jokainen ovi yksilöityisi omalle tunnisteelleen.



Kuva 15. Unreal Enginen Blueprints visuaalinen skripti klikattavan objektin tunnistukselle

Objektin tunnistukseen käytetään yksinkertaista skriptiä, mikä tunnistaa käyttäjän klikkaaman elementin ja näyttää sen luokan ja nimen (kuva 15). Tältä pohjalta on mahdollista jatkaa kohti valmista ohjelmaa.

Viimeisenä toimenpiteenä oli **lightmap UV:n** korjaaminen. Valitettavasti se ei onnistunut automaattisesti kovin hyvin. Valaistuksen voi korjata tai rakentaa myös käsin, mutta se on melko vaativa toimenpide. Lightmap UV on 2D-pohjainen kuva, jota käytetään etukäteen niin sanottuun paistettuun (engl. *baking*) valaistukseen. Dynaamisesti luotava valaistus vaatisi päätelaitteelta enemmän resursseja kuin staattinen etukäteen luotu valaistus. (Epic Games 2020a.)

Prototyypin Windows-versio oli käytettävissä projektin rakentamisen jälkeen: *File -> Package Project -> Windows (64 bit)*. Täysin optimoimattoman ohjelman koko on noin 754 Mt ja zip-pakattuna noin 423 Mt. Tarvittaessa ohjelman saa toteutettua myös muille Unreal Enginen tukemille käyttöympäristöille. (Epic Games 2020a.)

Mallinnuksena käytettiin oikeaa BIM-tietomallia. Tietomalli on ArkVisio Oy arkkitehtitoimiston vuonna 2019 suunnittelema.

4.3 Ongelmat ja huomiot

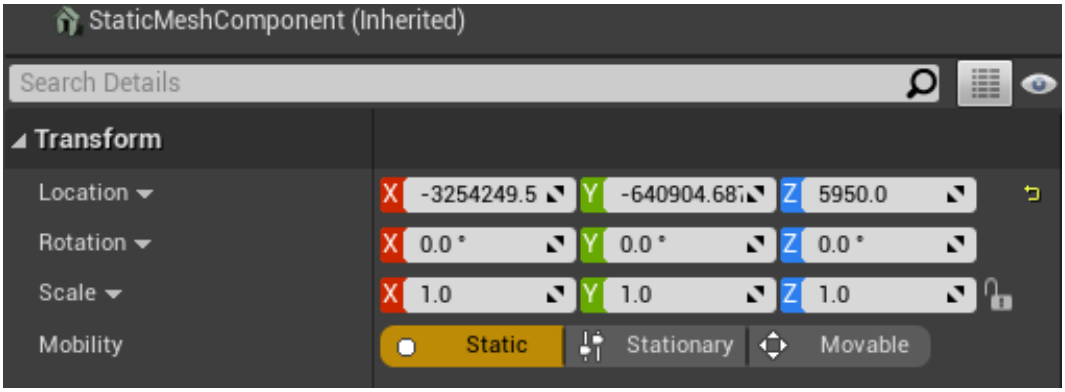
Manuaalisen työn osuus on melko suuri. Jonkinlainen törmäystunnistimien automaattinen luonti jokaiseen tai erikseen määriteltyyn IFC-elementtiin nopeuttaisi pelillistämistä.

Ohjelmien alati muuttuvat ominaisuudet voivat sotkea joitakin etukäteen määritettyjä toimintoja. Ohjelman version päivittyessä osa toiminnoista voi poistua tai muuttua maksullisiksi lisäominaisuuksiksi.

Ohjelmiston ylläpitäjä voi myös lopettaa kehitystyön, jolloin esimerkiksi turvallisuus-, versio- ja muut ominaisuuspäivitykset lakkaavat. Jos huolto- ja käyttöohjeen luonti on perustettu sille pohjalle, niin mitä sitten tehtäisiin?

Harmia aiheutti alkuperäisen mallinnuksen *lightmap UV overlapping* -ongelma. Alkuperäisen malliin toteutettu valaistus ei toimi Unreal Enginessä. Sisätilat muuttuvat mustiksi, koska kaksiulotteiset ns. UV-valaistuskartat menevät päällekkäin, eikä Unreal Enginen automaattinen valaistuksenkorjaus pysty tilannetta korjaaman. Muutoksen voisi tehdä alkuperäisessä ohjelmassa tai Unreal Enginen UV-työkalulla (Epic Games 2020a).

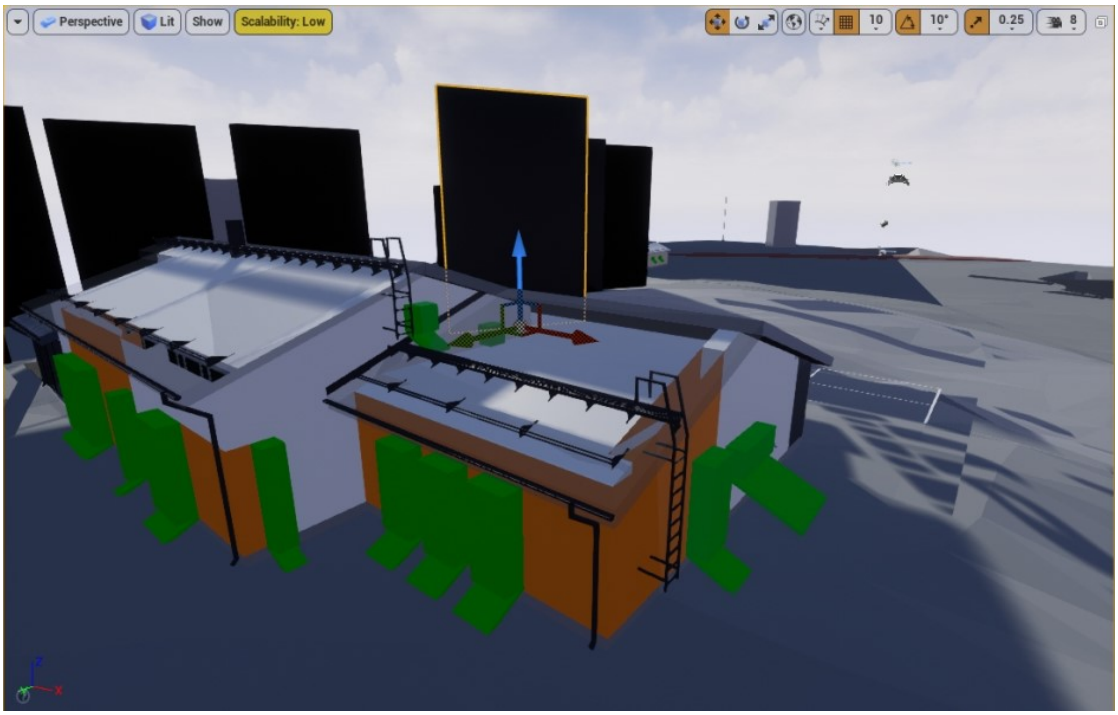
Mallin sijainti mallipohjassa suhteessa projektin origoon voi aiheuttaa pahoja ongelmia. Käyttämässäni tietomallissa mallinnus sijoittui hyvin kauas object originista, mikä aiheutti Unreal Engineen siirtäessä ongelmia (kuva 16). Mallinnusta oli mahdotonta saada toimimaan ilman korjausta. IFC-malli näkyi vain pienenä pisteenä Unreal Enginen editointiruudussa.



Kuva 16. Unreal Engineen tuodun IFC-mallin sijainti suhteessa pelimoottorin origoon

Mietin aluksi IFC-tiedoston muokkausta niin, että muuttaisin *IfcSite-Ympäristö-komponentin* koordinaattiarvot työpohjan origoa vastaaviksi. Onneksi minulla oli kuitenkin mahdollisuus muokata alkuperäistä ArchiCAD-tiedostoa. Siirsin projektin kokonaisuudessaan työpohjan nollapisteeseen, minkä jälkeen mallin hallinta ja käyttö onnistui UE-pelimoottorissa suunnitellusti.

Osa geometrista hajoaa jossain vaiheessa. Joitain komponenttien välisiä rakoja muodostuu ilmeisesti virheellisesti tallentuneen tai muunnoksen aikana muuttuneen geometrian takia.



Kuva 17. IFC-tuonnin virheitä Unreal Enginessä (ArkVisio 2019)

Osa IFC-tiedonsiirtona tuodusta geometriasta oli muuntunut ja tunnistui väärin. Kattolappeiden tekstuurien materiaali muuttui osin lasiksi. Puut näkyivät mustina laattoina ja tunnistuivat väärin, sekä rakennuksen kaikkiin aukkoihin ilmestyi vihreät elementit (kuva 17). Ikkunoiden materiaali piti vaihtaa kirkkaaksi lasiksi. Korjaus onnistuu onneksi helposti vaihtamalla kunkin ikkunaelementin materiaali oikeaa vastaavaksi. Vihreät elementit sai pois yksinkertaisesti valitsemalla ne kaikki ja painamalla delete-näppäintä.

Ohjelmien eri versiot eroavat toisistaan ja osa lisäosista ei välttämättä toimi kaikissa versioissa. Esimerkiksi Unityn *First Person Character* -asset ei toiminut suoraan, vaan piti Googletella vian syy. Netistä löytyi C#-koodiin korjaus, jolla asset alkoi toimimaan.

Suurimmaksi ongelmaksi omalla kohdallani huomasin projektin eteenpäin viennin sellaisissa tienhaaroissa, missä oli useampi kuin yksi tapa toteuttaa määritelty toimenpide. Vaihtoehtojen suunnaton määrä ja kokemattomuus tunnistaa oikea tapa hakea ongelman ratkaisu hidastavat toteuttamista. Googletus, Unreal Enginen dokumentoinnin läpikäynti ja avunpyyntö viimeisenä oljenkortena auttoivat lopulta eteenpäin.

Kysymykseen, *jos rakennus peruskorjataan tai siihen tehdään suuri rakenteellinen muutos, miten toteutetaan tietomallin päivitys*, en osaa tältä kädeltä suoraan vastata. Todennäköisesti uusi tietomalli korvaisi vanhan ja jo olemassa olevat tiedot ja toiminnallisuus liitettäisiin uuteen korjaus- ja huolto-ohjeeseen. Periaatteessa vanhaa IFC-tiedostoa voisi muokata vastaamaan uutta ulkoasua. Valmista pelillistämistä ei pysty enää jälkikäteen muokkaamaan pelimoottorissa geometrian suhteen.

Satunnaiset tietokoneen kaatumiset Unreal Enginen yhteydessä harmittivat. En tiedä mistä ne johtuivat, enkä alkanut sen kummemmin etsimään syytä tapahtuneeseen. Joskus sen on käsittääkseni aiheuttanut näytönohjaimen muistin ylivuoto tai vastaava ohjelmallinen seikka, eli ns. bugi.

5 PÄÄTÄNTÖ

Ideatasolla digitaalinen 3D-mallinnettu käyttö- ja huolto-ohje on mahtava. Lopullisen toteutuksen tiellä on kuitenkin monta mutkaa. Erilaiset ohjelmistojen ja tiedostojen yhteensopimattomuudet, toteutuksen työläys varsinkin alkuvaiheessa ja muut vastaavat aineistossa kuvatut negatiiviset seikat hidastavat prosessia.

En henkilökohtaisesti näe estettä tällaiselle tuotteelle, mutta manuaalisesti toteutettuna kaupallinen potentiaali jää vähäiseksi. Muunnoksen pitäisi olla automaattinen tai ainakin helposti ja nopeasti toteutettavissa käyttöliittymän integroituessa suoraan tietomalliin. Tarvittava käyttöliittymä voidaan suunnitella ja toteuttaa erillisenä komponenttina mikä liitetään tietomalliin pelimoottorissa. Näin tehden muunnokseen tarvittava aika voidaan minimoida.

Unreal Engine omilla ominaisuuksillaan olisi paras vaihtoehto toteuttaa halutun kaltainen IFC-tietomallia hyväksi käyttävä korjaus ja huolto-ohje. Sen valmiina olevat, sekä ladattavat lisäominaisuudet kattavat halutut toteutustyökalut. Virrake-työkalu helpottaisi tietomallin virtualisointia Unreal Enginellä, mutta ei ratkaisisi käyttö- ja huolto-ohjeen tuottamista koskevia ongelmakohtia ainakaan suoraan.

Unity itsessään toimii hieman kankeasti IFC-mallin käyttämiseen tällä hetkellä. Ulkoisen XML-tiedoston käyttö on hankalaa. Maksulliset lisäosat todennäköisesti helpottaisivat IFC-tiedoston kanssa toimimista.

ArchiCADin BIMx-hypermalli puolestaan on enemmänkin tietomallin erinäisten tietojen, kuten esimerkiksi geometrian, pohjapiirustusten ja metatiedon esittelytarpeen täyttävä.

Jos toteutus olisi vaatimusmäärittelyn kaltainen, siitä olisi todennäköisesti hyötyä tai jopa kaupallista potentiaalia toimeksiantajalle. Ohjelman voisi tarjota jälkimarkkinoille tai suunnitellun rakennuksen käyttäjälle. Toteutuksen jatko prototyypistä eteenpäin jää nähtäväksi. Pelimoottorissa rakennettu perusta olisi

kuitenkin käytettävissä uusien tietomallien pelillistämiseksi. Jos kuhunkin pelillistämiseen käytetty aika olisi muunnettavissa riittäväksi kompensaatioksi kulujen kattamisen ja riittävän voittomarginaalin saavuttamiseksi, toiminnalla voisi olla jopa voittoa tuottavaa liiketoiminnallista merkitystä.

Tulevaisuus on todennäköisesti virtuaalisempi kuin nyt. **VR**, eli virtuaalinen todellisuus (engl. *Virtual Reality*) ja **AR**, eli lisätty todellisuus (engl. *Augmented Reality*) sekä muut vastaavat teknologiat tulevat mahdollisesti lisääntymään jokapäiväisessä elämässämme. Uusiutuva teknologia syrjäyttää osin vanhenevaa tekniikkaa ajan myötä. Mutta ei pidä unohtaa vanhoja tekniikoita kokonaan. Paperille piirrettyjä rakennuspiirustuksia voi käyttää ilman sähköä ja tietoteknisiä laitteita.

Esineiden internet (engl. *Internet of Things*), eli **IoT**-ratkaisut lisääntynevät kodeissa ja konttoreissa. Kaikki kotien ja toimistojen pienetkin laitteet voisivat tulevaisuudessa mahdollisesti keskustella älykkään tietomallipohjaisen käyttö- ja huolto-ohjeen kanssa. Käyttäjä näkisi yhdellä silmäyksellä kaikki huoltoa, korjausta tai jotain muuta toimenpidettä tarvitsevan kohteen.

Eräänä toiminnallisena visiona automatisoitu korjausta tai ylläpitoa kaipaava laite lähettäisi huoltokutsun autonomisesti ja korjaaja kävisi tekemässä tarvittavan toimenpiteen kohteessa. Käyttäjän ei tarvitsisi kuin hyväksyä ja maksaa toimenpide.

Väärinkäytösten ja vastaavien estäminen on suunniteltava ennalta. Joitain muitakin riskejä sisältyy digitalisaatioon. Sähköisessä muodossa data ja informaatio ovat riippuvaisia monista ympäristövaikutteista. Data voi esimerkiksi korruptoitua, hävitä tai muuttua tahattomasti.

Ehkäpä tulevaisuudessa jonkinlaisella suoraan henkilön aivokuoreen syötettävällä sähköisellä impulssilla voitaisiin simuloida todellinen immersio jossain virtuaalikohteessa olemisesta. Tällä hetkellä tieteiskirjallisuudelta kuulostava asia voi olla joskus arkipäivää, muodossa tai toisessa.

Kärjistetysti sanoen toisen maailmansodan jälkeen piirrettiin Klubiaskin kanteen tyyppitalon piirustus. Nyt voidaan digitaalisesti simuloida kokonainen kaupunki kädessä pidettävään laitteeseen.

SANASTO

2D-malli	Kaksiulotteinen tasomalli
3D-malli	Tietokoneella tehty kolmiulotteinen mallinnus
4D-malli	3D-malli, johon on lisätty aikataulu
Asset	Pelimoottorin lisäpaketti tai lisättävä komponentti
BP Visual Scripting	UE:n Blueprints visuaalinen skriptaus
Baking	Tekniikka millä valaistuksia renderöidään ennakkoon
BIM	Building Information Model, rakennuksen tietomalli
BIMx	ArchiCADin interaktiivinen tietomalli, eli ns. hypermalli
CAD	Computer Aided Design, tietokoneavusteinen suunnittelu
Collider	Törmäystunnistin pelimoottorissa
Collision	Yleisnimitys erilaisille törmäyttimille pelimoottorissa
DAE	Collada, XML-pohjainen 3D-tiedostotyyppi
Digital Twin	Rakennuksen digitaalinen kaksonen
DWF/DWG	Drawing Interchange Format, Autodeskin tiedostotyyppi tiedonsiirtoon
Edge	Sivu, särmä, kahden kärjen määrittelemä jana
Face	Tahko, kolmen tai useamman sivun pinta
FBX	Autodeskin 3D-tiedostotyyppi
IFC	Industry Foundation Classes, avoin tiedostotyyppi tiedonsiirtoon
Natiivi	Esimerkiksi ohjelman oma tiedostotyyppi
Polygon mesh	Monikulmioverkko
Spline	Splini, matemaattisesti määritelty käyrä
Origo, object origin	Mallipohjaan määritetty nollapiste (x:0,y:0,z:0)
OBJ	Avoin 3D-tiedostotyyppi
Overlap	Unreal Enginen eräs törmäystyyppi
Pivot point	Objektiin määritetty keskipiste
PLN	ArchiCADin natiivi tiedostotyyppi
Renderöinti	Näytöllä näkyvä kuva muodostetaan tietokoneella
Trigger	Yleisnimitys törmäystunnistimen laukaisimelle
Vertex	Kärki, piste moniulotteisella koordinaatioasteikolla

LÄHTEET

Autodesk. 2020. Suunnittelun ja viihdeteollisuuden 3D-ohjelmistot. WWW-dokumentti. Saatavissa: https://www.autodesk.fi/ [viitattu 14.9.2020].

BIMserver.org. S.a. Open source building information server. WWW-dokumentti. Saatavissa: http://bimserver.org/ [viitattu 15.9.2020].

Blender. S.a. Blender manual. WWW-dokumentti. Saatavissa: https://docs.blender.org/manual/en/latest/index.html [viitattu 2.6.2020].

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P. & Lévy, B. 2010. Polygon Mesh Processing. E-kirja. CRC Press. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 24.4.2020].

buildingSMART. 2020a. IFC Formats. WWW-dokumentti. Saatavissa: https://technical.buildingsmart.org/standards/ifc/ifc-formats/ [viitattu 4.6.2020].

buildingSMART. 2020b. Standardit. WWW-dokumentti. Saatavissa: https://buildingsmart.fi/standardit/ [viitattu 24.7.2020].

buildingSMART. 2020c. Yleiset tietomallivaatimukset YTV2012. WWW-dokumentti. Saatavissa: https://buildingsmart.fi/yleiset-tietomallivaatimukset-ytv/ [viitattu 24.7.2020].

Edwards, G., Li, H., & Wang, B. 2015. BIM based collaborative and interactive design process using computer game engine for general end-users. *Visualization in Engineering* 3. Verkkolehti. DOI: 10.1186/s40327-015-0018-2. Saatavissa: https://viejournal.springeropen.com/track/pdf/10.1186/s40327-015-0018-2 [viitattu 29.6.2020].

Epic Games. 2020a. Unreal Editor Manual. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/en-US/Engine/Editor/index.html [viitattu 13.5.2020].

Epic Games. 2020b. Twinmotion. WWW-dokumentti. Saatavissa: https://www.unrealengine.com/en-US/twinmotion [viitattu 15.9.2020].

Epic Games. 2020c. Unreal Engine. WWW-dokumentti. Saatavissa: https://www.unrealengine.com/en-US/ [viitattu 15.9.2020].

EU Bim Task Group. 2016. Käsikirja tietomallintamisen käyttöön ottamisesta Euroopan julkisella sektorilla. PDF-dokumentti. Saatavissa: http://www.eubim.eu/wp-content/uploads/2018/10/GROW-2017-01356-00-00-FI-TRA-00.pdf [viitattu 27.8.2020].

Garber, R. 2014. BIM Design. E-kirja. Wiley. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 22.4.2020].

Granlund. S.a. Virtuaalinen kiinteistö. WWW-dokumentti Saatavissa: https://www.granlund.fi/ohjelmistot/tuotteet-ja-palvelut/virtuaalinen-kiinteisto/ [viitattu 9.6.2020].

Graphisoft. 2020. ArchiCAD. WWW-dokumentti Saatavissa: https://graphisoft.com/solutions/products/archicad [viitattu 17.9.2020].

Gumster, J. 2009. Blender for dummies. Indianapolis: Wiley.

Halmetoja, E. 2016. Tietomallit ylläpidossa. Senaatti-kiinteistöt. PDF-dokumentti. Päivitetty 21.9.2016. Saatavissa: https://www.senaatti.fi/app/uploads/2017/05/6099-Tietomallit_yllapidossa.pdf [viitattu 4.4.2020].

Hilma. S.a. Julkiset hankinnat. Työ- ja elinkeinoministeriö. WWW-dokumentti. Saatavissa: https://www.hankintailmoitukset.fi/fi/ [viitattu 27.8.2020].

Home, L. 2015. AutoCAD ja AutoCAD LT 2016 perusteet. Lulu.com.

IfcOpenShell. 2020. The open source ifc toolkit and geometry engine. WWW-dokumentti. Saatavissa: http://ifcopenshell.org/ [viitattu 15.9.2020].

Kiviniemi, M. 2017. Tietomallit ylläpitoon. VTT. PDF-dokumentti. Saatavissa: https://buildingsmart.fi/wp-content/uploads/2017/06/bSF_SSTY_Tietomallit-yll%C3%A4pitoon_31-05-2017.pdf [viitattu 22.4.2020].

M.A.D. 2013. IFC-tiedonsiirto. PDF-dokumentti. Saatavissa: https://www.mad.fi/tiedostot/pdf/kasikirja16/YS.IFC_web.pdf [viitattu 1.4.2020].

Nevalainen, J. 2016. Rakennuksen 3D-tietomallipalvelin. Tampereen teknillinen yliopisto. Tieto- ja sähkötekniikan tiedekunta. Diplomityö. PDF-dokumentti. Saatavissa: https://trepo.tuni.fi/bitstream/handle/123456789/23633/nevalainen.pdf?sequence=3 [viitattu 7.4.2020].

Pixyz. 2020. Products and solutions. WWW-dokumentti. Saatavissa: https://www.pixyz-software.com/ [viitattu 15.9.2020].

Solibri. 2020. Tuotteet. WWW-dokumentti. Saatavissa: https://www.solibri.com/fi/ [viitattu 15.9.2020].

Steenberg, E. 2018. Model for Real Time—Beyond Counting Polygons. Intel. WWW-dokumentti. Saatavissa: https://software.intel.com/content/www/us/en/develop/articles/model-for-real-time-beyond-counting-polygons.html [viitattu 13.5.2020].

Tekla. S.a. Tekla Structures. WWW-dokumentti. Saatavissa: https://www.tekla.com/fi/tuotteet/tekla-structures [viitattu 14.9.2020].

Tridify. 2020. Automated BIM to VR Workflow. Online IFC Viewer. WWW-dokumentti. Saatavissa: https://www.tridify.com/ [viitattu 15.9.2020].

Trimble Heavy Industry. S.a. SketchUp. WWW-dokumentti. Saatavissa: https://heavyindustry.trimble.com/en/products/sketchup [viitattu 14.9.2020].

Tuhola, E. & Viitanen, K. 2008. 3D-mallintaminen suunnittelun apuvälineenä. Tampere: Tammertekniikka.

Unity. 2020a. Unity Manual. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/Manual/ [viitattu 13.5.2020].

Unity. 2020b. Unity Real-Time 3D Development Platform. WWW-dokumentti. Saatavissa: https://unity.com/ [viitattu 17.9.2020].

Virrake. S.a. Virtuaalinen Rakentaminen. WWW-dokumentti. Saatavissa: https://virrake.fi/ [viitattu 20.5.2020].

KUVALUETTELO