LAB University of Applied Sciences

3D-malli Unreal Engine -pelimoottorissa

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja viestintätekniikka
2021
Pekka Huohvanainen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Huohvanainen Pekka	Opinnäytetyö, AMK	Syksy 2021
	Sivumäärä	
	38	

Työn nimi

3D-malli Unreal Engine pelimoottorissa

Tutkinto

Insinööri (AMK), Tieto- ja viestintätekniikka

Ohjaavan opettajan nimi, titteli ja organisaatio

Ismo Jakonen, Lehtori, Mediatekniikka

Toimeksiantajan nimi, titteli ja organisaatio

Henri Koskinen, Toimitusjohtaja & mediatuottaja, Pixtell Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli tutkia, kuinka luoda toimivia 3D-malleja Unreal Engine -pelimoottoriin. Tutkimus suoritetaan Unreal Engine 4:n näkökulmasta, mutta monet asiat pätevät myös muihin pelimoottoreihin ja 3D-projekteihin.

Pixtell Oy toimi työn toimeksiantajana. Yrityksen Last Drop seikkailupeliä varten täytyi luoda tehokas prosessi, minkä avulla on mahdollista luoda iso määrä yhtenevän näköisiä 3D-malleja.

Prosessi jakautui mallien suunnitteluun, mallinnukseen, teksturointiin ja exportaukseen. Lisäksi on ymmärrettävä kuinka 3D-mallit rakentuvat ja mitä tiedostoformaatteja Unreal Enginen tukee.

Johtopäätöksenä hyvä teksturointi ja materiaalien muokkaus Unreal Enginessä johtivat parhaaseen lopputulokseen. Runsaat mallikuvat ja toimiva mallin rakenne puolestaan tekevät itse mallinuksesta nopeaa.

Asiasanat

3D mallinnus, Unreal Engine, peli, teksturointi, exporttaus

Abstract

Author(s)	Type of Publication	Published
Huohvanainen Pekka	Thesis, UAS	Fall 2021
	Number of Pages	
	38	

Title of Publication

3D model in Unreal Engine

Name of Degree

Engineer (UAS), Information and Communications Technology

Name, title and organization of the supervising teacher

Ismo Jakonen, Lecturer, Media technology

Name, title and organization of the client

Henri Koskinen, CEO & Media Producer, Pixtell Oy

Abstract

Purpose of this thesis was to research how to create functional 3D models for Unreal Engine. Thesis is based around Unreal Engine 4 but many of its aspects apply to other game engines and 3D projects.

Client of this thesis was Pixtell Oy. Last Drop is an adventure game being made by the company which requires an efficient process for making consistent looking 3D models.

Process was split into planning, modeling, texturing, and exporting. In addition, it was essential to understand 3D model structure and file formats when importing game assets into Unreal Engine.

In conclusion well-made texturing and material editing in Unreal Engine led to the best result. In addition, sufficient reference pictures and good understanding of 3D model's structure makes the modeling process more efficient.

Keywords

3D modeling, Unreal Engine, game, texturing, export

Sisällys

1 Johdanto

Kolmiulotteiset mallit monien pelien ja animaatioiden perusta. 3D-mallien suunnittelu ja mallinnus ovat aikaa vieviä prosesseja, missä virheet voivat aiheuttaa myöhemmin paljon ongelmia esimerkiksi peliprojektissa. Peliin luodun mallin vaatii erityisesti suunnittelua, koska niiden täytyy olla tiedostokooltaan pieniä. Ne tarvitsevat myös törmäysobjektit, jotka estävät pelaajaa liikkumasta mallien läpi.

Tämän opinnäytetyön tarkoituksena on esitellä mihin eri asioihin on kiinnitettävä huomiota luotaessa 3D-malleja Unreal Engine pelimoottorille. Työssä ei ole tarkoitus käydä läpi kaikkia kolmiulotteisen mallinnuksen tekniikoita tai Unreal Enginen toimintaa, vaan kertoa mihin asioihin keskittymällä 3D-malleista saa helppokäyttöisiä ja toimivia, kun ne tuodaan Unreal Engine pelimoottoriin.

Teoriaosuudessa käydään ensin läpi mallinnuksen ja pelimoottoreiden perusteita, kuten mallin rakenne, esimerkkejä mallinnuskeinoista ja pelimoottoreista. Ennen mallinnusta malli on suunniteltava. Tämä tarkoittaa mallikuvien ja konseptitaiteen käyttöä. Toimivat mallikuvat nopeuttavat mallinnusta ja hyvä konseptitaide ohjaa mallintajaa kertomalla projektin teeman. Suunnittelun aikana on myös hyödyllistä luoda karkea malli, jonka avulla voi sommitella mallin koon ja yleisimmät piirteet. Mallinnusprosessi on tärkein osa mallin luontia, joten on tiedettävä, kuinka toimiva mallin rakenne luodaan ja, mitkä asiat nostavat pelimallin tiedostokokoa. Teoriaosuuden tekstuuri osiossa kerrotaan, kuinka tekstuurit asetetaan mallin pinnalle ja miten useiden tekstuurien käyttö tekee malleista paremman näköisiä.

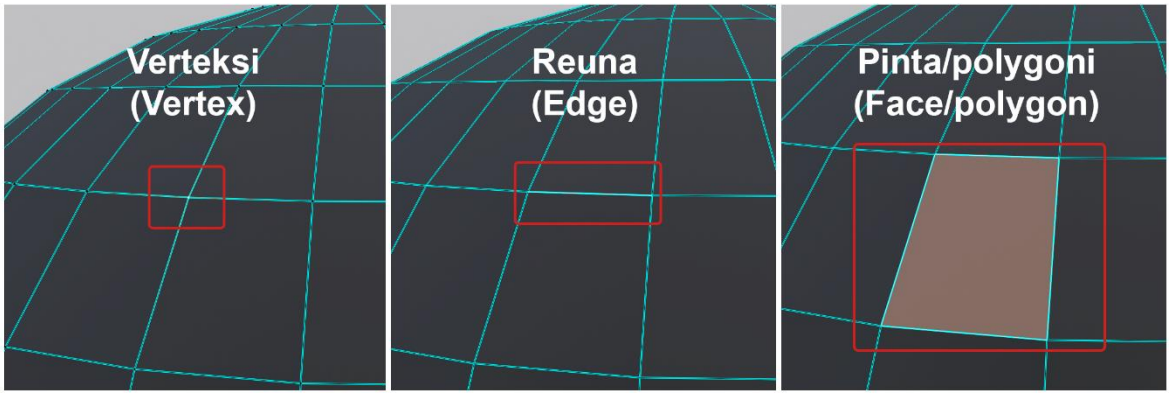
Opinnäytetyö tutkii prosessia Unreal Enginen näkökulmasta. Mallien kanssa työskennellessä on tiedettävä kuinka Unreal Engine ja muut pelimoottorit tulkitsevat mallien tekstuurit ja, mitä tiedostoformaatteja on käytettävä exporttaus vaiheessa. Unreal Enginessä pystyy myös jatkamaan mallien muokkausta. Tämä on tehokas keino luoda erilaisia versioita mallista muokkaamalla materiaalia ja lisäämällä tekstuureja mallin pinnalle.

Jyväskyläläinen Pixtell Oy on työn toimeksiantaja ja opinnäytetyön case osiossa esitellään Last Drop peliin mallinnettuja 3D-malleja. Huhtasuon seurakuntakeskuksesta luotu malli on esimerkki, jossa mallikuvien avulla suunniteltiin malli, joka teksturointi lisää malliin yksityiskohtia pitämällä tiedostokoon pienenä.

2 Pelimoottorit ja mallinnus

2.1 3D-mallinnus

3D-mallit muodostuvat vertekseistä, reunoista ja pinnoista. Kaksi verteksia muodostavat pinnan ja kolme tai useampi reuna muodostavat pinnan (Slick 2021.) Malli luodaan manipuloimalla ja lisäämällä näitä eri objektin osia. Nämä osat luovat mallille geometrian ja määrittävät tekstuurien lisäksi mallin tiedostokoon. Kuvassa yksi esitellään nämä osat. Pintoja kutsutaan myös polygoneiksi.

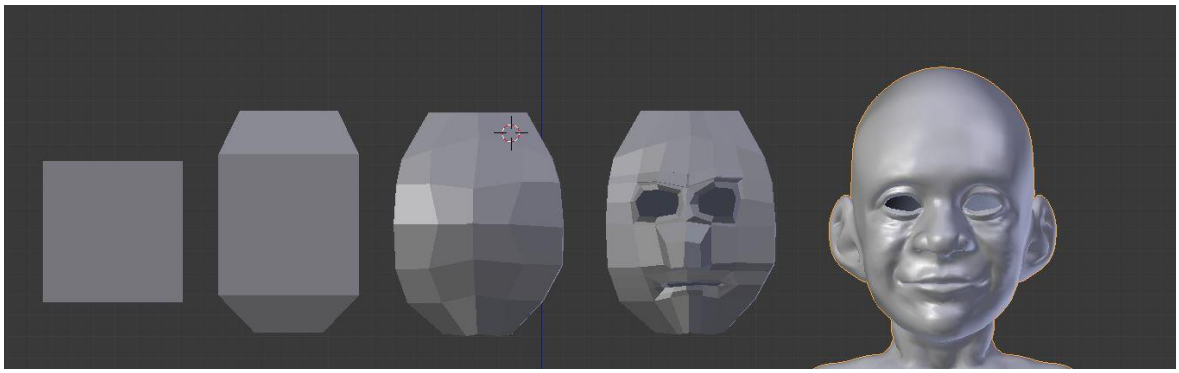


KUVA 1 Mallin eri osat

3D-malleja käytetään esimerkiksi peleissä, animaatioissa, elokuvateollisuudessa tai itsenäisissä rendereissä. Monet mallinnusohjelmat pystyvät luomaan 3D-animaatioita itsenäisesti ja Blenderin ominaisuuksiin kuuluu jopa video editointi työkalu. Tämä ei kuitenkaan ole aina vastine videoeditointi ohjelmille tai vastaaville animaatiotyökaluille. Jotkin mallinnusohjelmat pitävät sisällään myös hyvät työkalut renderöintiä varten. Renderöinti on prosessi, missä tietokone luo kuvan ohjelmassa luodusta kohtauksesta. Ohjelma laskee lopputuloksen ottaen huomioon valot, materiaalit sekä mallien geometrian, joten tässä prosessissa voi kestää erittäin pitkään. Renderöinti on usein viimeinen työvaihe, jos kyseistä kuvatiedostoa ei ole tarkoitus editoida jälkikäteen (Denham b). Blender mallinnusohjelmasta löytyy myös työkalut kuvan jälkikäsittelyyn, mikä toimii hyvin varsinkin animaatiota luodessa, koska kaikkiin renderöityihin kuviin saa helposti samat muutokset automaattisesti. Animoinnin ja renderöinnin pystyy myös suorittamaan erillisessä ohjelmassa, kuten Unreal Enginessä tai Unity -pelimoottoreissa. Lopputulos tulee olemaan erilainen ohjelmasta riippuen, koska eri ohjelmat käyttävät eri renderöinti moottoreita.

Itse 3D-mallinnukseen on monia eri keinoja. Mallinnuksen voi esimerkiksi aloittaa valmiista yksinkertaisesta objektista kuten kuutiosta tai sylinteristä. Kuutiosta aloitettu mallinnus tai

box modeling on helppoa ja nopeaa, joka toimii hyvin aloittelijalle, koska kuution kahdeksan verteksiä on helppo aloituspiste. Box modeling tapahtuu lisäämällä yksityiskohtia malliin, kunnes lopputulos on lähellä toivottua lopputulosta. Mallin voi viimeistely vaiheessa asettaa modifikaattorin, joka pehmentää mallin kulmat automaattisesti lisäämällä ison määrän pintoja. (Danan 2016 a). Reunojen pehmennystä voi säätää, jos tietyt alueet on tarkoitus pitää terävinä, Kuvassa kaksi on esimerkki tällaisesta mallinnuksesta. Box modeling auttaa aloittelijoita hahmottamaan mallinnuksen perusteita.



KUVA 2. Box modeling -mallinnuksen eri vaiheita (Danan 2016 a)

Veistäminen tai skulptaus, englanniksi sculpting, on mallinnusmenetelmä, missä malli luodaan esimerkiksi vetämällä, puristamalla ja tasoittamalla mallia kuin savea. Skulptaus tarvitsee ison määrän polygoneja eli pintoja, jotta päästään toivottuun muotoon. Mallinnuksen voi aloittaa esimerkiksi pallosta, jossa on paljon pintoja, joita siirtelemällä päästään toivottuun lopputulokseen. Skulptaus eroaa paljon monista muista mallinnustavoista, koska siinä käytetyt työkalut toimivat vain skulptauksessa. Skulptaus on monella tapaa box-modeling-tekniikan vastakohta, koska lähtökohdat ovat vastakkaiset. Box-modeling aloitetaan pienestä määrästä pintoja, kun taas skulptauksessa mallit sisältävät paljon geometriaa jo lähtötilanteessa. (Heginbotham.) Kuvassa kolme on esimerkki ZBrush -ohjelmalla skulptatusta 3D-mallista.

KUVA 3. Skulptattu 3D-hahmo (Gadea 2017)

Mallinukseen on olemassa monia eri maksullisia ja ilmaisia ohjelmia. Autodesk on alan johtaava 3D-mallinnus ohjelmistoja tarjoava yritys, joten monet yrityksen ohjelmat ovat alan standardi ohjelmistoja. Autodeskin 3ds Max on yksi alan vanhimmista ohjelmista, kun taas Maya on yksi käytetyimmistä. Autodeskin ohjelmat ovat kuitenkin maksullisia. Muita maksullisia ohjelmia ovat esimerkiksi ZBrush ja Houdini. ZBrush on suosittu skulptaus ohjelma ja Houdini on kattava työkalu VFX-suunnittelijalle. Blender on puolestaan ilmainen avoimen lähdekoodin mallinnusohjelma. Blenderin moniin ominaisuuksiin kuuluu mallinnuksen lisäksi riggaus, skulptaus, teksturointi ja animaatio. (Format 2019.)

2.2 Pelimoottorit

Pelimoottorit ovat tietokoneohjelmia, jotka ovat suunniteltu erityisesti pelien luontia varten. Näiden ohjelmien ominaisuuksiin kuuluu graafinen renderöinti, fysiikan simulointi, tekoäly, äänet ja animaatio. Näiden ansiosta pelimoottorit kykenevät luomaan uskottavia virtuaalisia maailmoja. (Fullscale 2021.) Pelimoottorit tukevat monia ohjelmointikieliä, joista suosituimpia ovat C#, C++, Java, Javascript sekä Lua (Bhattacharya 2021).

Unity ja Unreal Engine ovat suosittuja pelimoottoreita. Unity on ilmainen versio harrastelijoille ja yrityksille, joiden liikevaihto on alle 100 000 $ vuodessa. Maksullinen versio avaa lisää ominaisuuksia kehittäjän käyttöön. Unreal Engine on puolestaan ilmainen ohjelma, joka muuttuu maksulliseksi vasta, kun julkaistun pelin tuotto ylittää tietyn summan. (Fullscale 2021.)

3 3D-mallinnus

3.1 Mallin suunnittelu

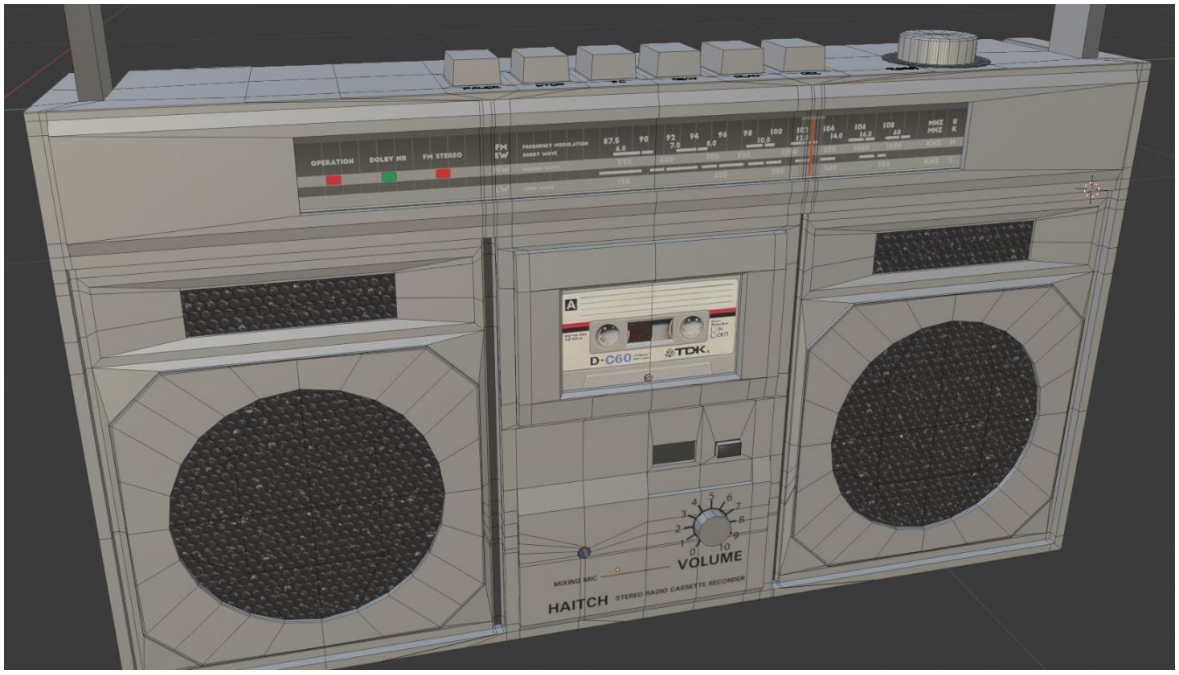
Ennen kolmiulotteista mallinnusta on tärkeä tietää mihin tarkoitukseen malli tehdään. Riippuen mallin tarkoituksesta tulee mallinuksessa keskittyä eri asioihin. Rendereissä käytetyt objektit nähdään vain tietyistä kuvakulmista, kun puolestaan peliobjektit usein nähdään joka puolelta. Pelien objektien täytyy myös olla tiedostokooltaan pieniä toisin kuin renderöitävien mallien. 3D-mallin kokoon vaikuttaa geometrian eli pintojen määrä, sekä tekstuurien eli mallin pinnalle projisoitujen kuvatiedostojen koko.

Jos projekti koostuu monista eri objekteista, kannattaa luoda yksinkertaiset mallit, joiden avulla on mahdollista sommitella lopullisten mallien paikat. Tärkeintä tällaisia malleja luodessa on pitää mielessä lopullisten mallien koko. Jos projekti simuloi fyysistä todellisuutta, malleja skaalatessa on hyvä perustaa mittakaava yhden objektin ympärille. Jos esimerkiksi ikkunan tai oven koko on standardi kaikissa tilanteissa, on tämän koon ympärille helppo skaalata muut mallit. Kaikkien mallien kokoa voi muuttaa myöhemmin, mutta suunnittelu vaiheessa on hyvä saada selville, minkälaiseen tilanteeseen malli luodaan. (Abid & Naghdi.)

Kamera ja sen käyttö on yksi tärkeimmistä elementeistä renderöintiä suunniteltaessa. Koko kohtaus luodaan kamerakulman ympärille valoista malleihin. Luonnollisesti suurin osa objekteista nähdään vain yhdestä kuvakulmasta. Näin ollen mallintaessa ei tarvitse kiinnittää huomiota miltä malli näyttää esimerkiksi takaa. Monissa objekteissa tämä ei kuitenkaan ole suotavaa, varsinkin jos kyseinen objekti on osa 3D-animaatiota. Kameran liike on yleistä animaatioissa, mikä saattaa paljastaa mallien epätäydellisiä osia. Kohtauksen kamera toimii kuin oikean maailman kamera. joten kameran syväterävyysaluetta tai polttoväliä on mahdollista säätää. Jotkin kameroiden asetukset kuitenkin vaihtelevat ohjelmien mukaan, mikä tarkoittaa tarkkojen kamerakulmien ja kamera asetusten suunnittelu ennen pelimoottoriin siirtämistä ei aina ole suotavaa. Pelimoottorit laskelmoivat myös valaistuksen eri tavalla verrattuna mallinnusohjelmiin. (Chang.) Blender mallinnusohjelmassa lamppu ei valaise samalla tavalla kuin Unreal Enginen vastaava valonlähde, koska ohjelmat käyttävät eri renderöintimoottoreita. Kaikki ohjelmat eivät myöskään käytä samoja yksiköitä valon intensiteettiä määritettäessä. Renderöinneissä mallien koolla ei ole suurta merkitystä. Mallit, joilla on paljon geometriaa voivat hidastaa mallinnusohjelmaa, mutta itse renderöinti aika ei nouse erityisesti. Tämän takia mallinnusmenetelmän voi valita vapaasti.

3.1.1 Peliobjektien suunnittelu

3D-mallit ovat monien pelien perusta. Pelien täytyy usein ladata useita malleja samalle alueelle, joten mallit täytyy suunnitella tarkasti. Mallintaessa on otettava huomioon kaikki asiat, mitkä nostavat lopullista tiedostokokoa, koska pelimoottorien täytyy ladata kaikki mallit aina, kun pelaaja käynnistää pelin. Tämä tarkoittaa, että mallien geometria sekä tekstuurien määrä ja koko on pidettävä mahdollisimman pienenä. Tästä syystä skulptaus tai vastaavat isoihin polygonimääriin johtavat mallinnusmenetelmät eivät toimi parhaiten kaikissa pelimalleissa. Geometriaa lisäävät modifikaattorit lisäävät pintoja eksponentiaalisesti, joten näiden käyttämistä kannattaa rajoittaa. Geometrian rajoittaminen on yleensä yksinkertaista, jos kyseessä on yksinkertainen kappale. Tämä ei kuitenkaan pidä paikkaansa, jos kyseessä on malli, jonka pelaaja näkee usein. Lisäksi koko pelinkehitys prosessia ei pysty suunnittelemaan mallinnettaessa ja on mahdollista, että mallinnusohjelmassa sommiteltuun pelialueeseen lisätään moninkertainen määrä malleja myöhemmin. Tämän takia malleista on turvallista tehdä mahdollisimman kevyitä, jotta työskentely mallien kanssa on sujuvaa ja lopullisen pelin latausajat pysyvät nopeina. (Jase 2020.) Kuvan neljä radio on esimerkki kevyestä mallista, josta on helppo luoda erilaisia versiota.



KUVA 4. Peliin luotu radio

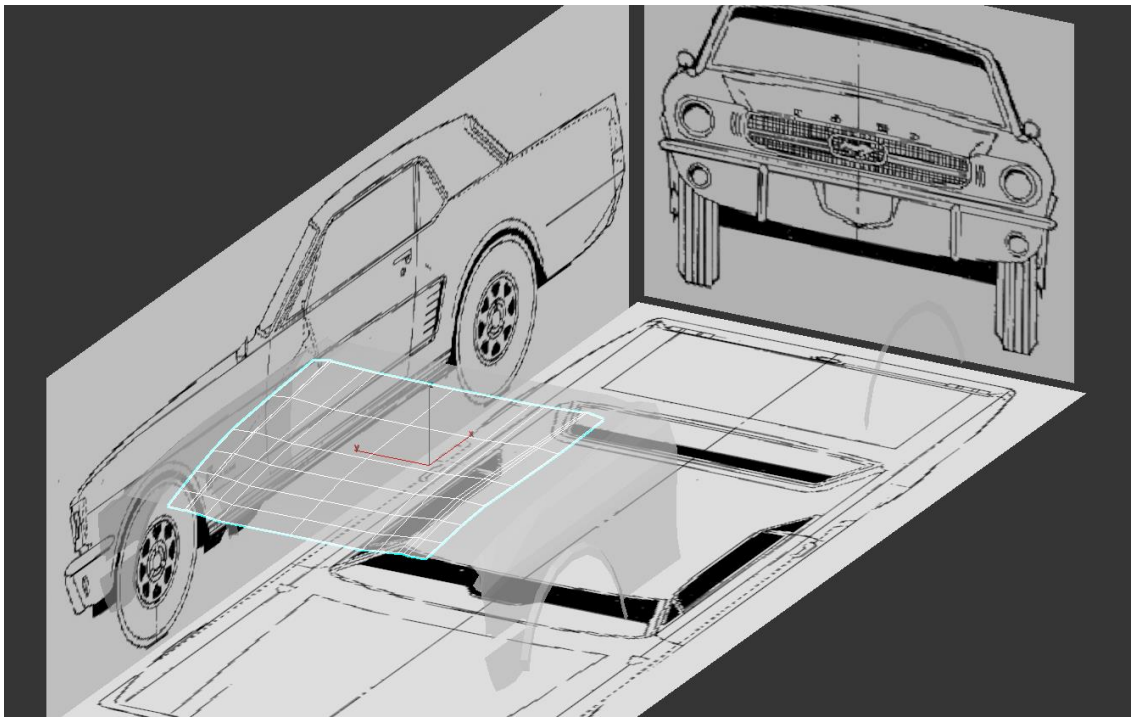
Ylimääräiset pinnat eivät pelkästään pienennä tiedostokokoa. 3D-mallit tarvitsevat värityksen, joka saadaan lisäämällä kuvatiedosto kappaleen ympärille. Näitä kutsutaan

tekstuureiksi. Tekstuurit ovat helpompia asetella kappaleen pinnalle myöhemmin, jos mallissa ei ole ylimääräisiä pintoja.

3.1.2 Mallikuvat

Mallikuvien käyttö on erittäin hyödyllistä mallinnettaessa. Monissa projekteissa ja erityisesti pelituotannossa mallikuvat tai konseptitaiteen luo erillinen artisti, joka työskentelee perinteisillä kaksiulotteisilla kuvilla, joita 3D-artisti käyttää pohjana malleille. Konseptitaide on termi kuville, jotka ohjaavat myöhempien pelitiimin jäsenten työskentelyä. Nämä voivat olla esimerkiksi yksinkertaisia luonnoksia tai tarkkoja piirroksia. Konseptitaide ei pelkästään näytä mallin muotoja tai värejä, mutta myös projektin yleistä teemaa ja tunnelmaa mihin mallin täytyy sopia. (Kutz 2019.) Mallikuvat voivat olla myös valokuvia. Mallikuvien on hyvä olla selkeissä kuvakulmissa, jotta mallinnus on helppoa.

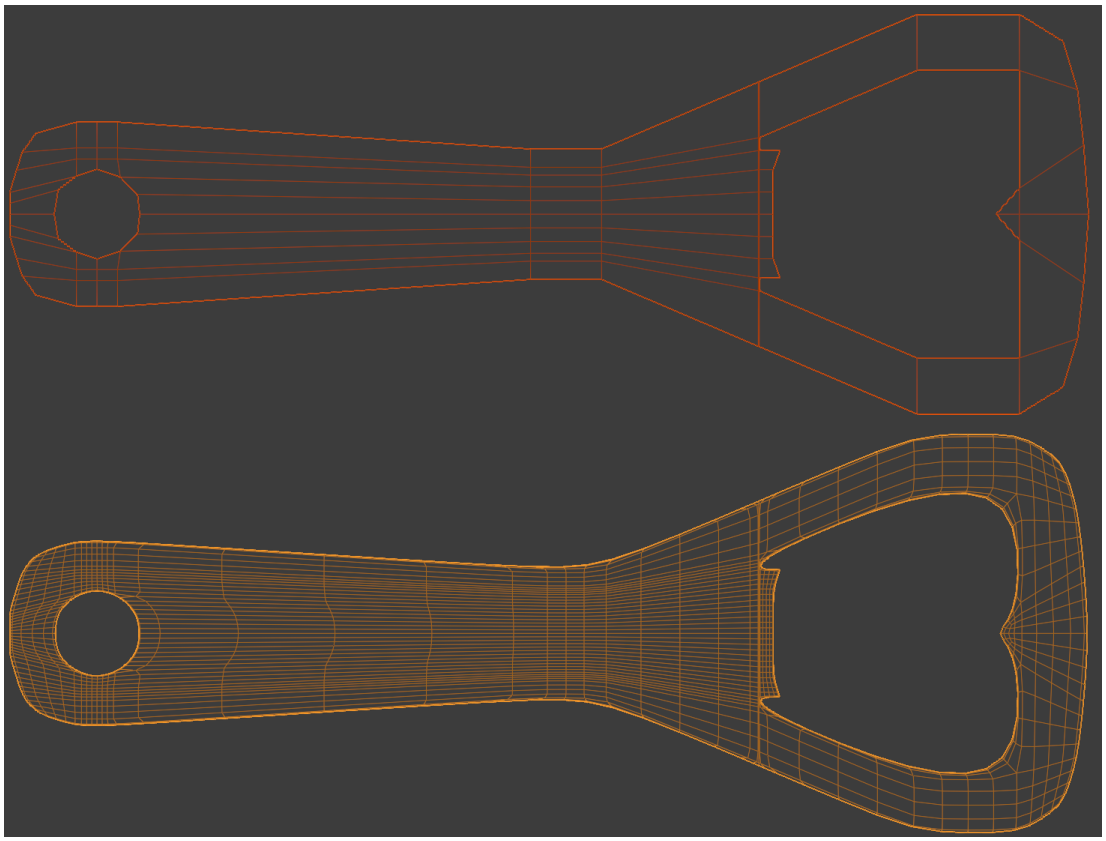
Kolmiulotteinen maailma jakautuu X, Y ja Z ulottuvuuksiin ja 3D-mallinnusohjelmissa objekteja voi tarkastella suoraan näiden akselien suuntaisesti. Työskennellessä malleja pystyy tarkastelemaan kaikista mahdollisista näkökulmista, mutta usein kappaleen hahmottaminen on helpompaa, kun sen näkee esimerkiksi suoraan edestä. Monet mallikuvat eri näkökulmista auttavat mallinnusprosessia erittäin paljon. 3D-mallinnusohjelmissa voi säätä näitä näkymiä ja jakaa ruudun, jotta mallikuvia pystytään tarkistelemaan eri kuvakulmista samanaikaisesti. Mallikuvat ovat hyödyllistä lukita paikoilleen ja piilottaa tarvittaessa, jotta ne eivät tule mallinnuksen tielle. (Selin.) Kuvassa viisi on esimerkki kolmen mallikuvan käytöstä 3ds Max -ohjelmassa. Järjestelemällä mallikuvat tällä tavoin tekee auton eri osien hahmottamisesta helppoa ja nopeaa.



KUVA 5. Esimerkki mallikuvien käytöstä

3.2 Low polygon -malli

Koska mallin koko määrittyy osaksi polygonien eli 3D-mallin pintojen määrästä, on suotavaa pitää mallien geometria mahdollisimman yksinkertaisena. Low polygon -malli on yksinkertainen 3D-objekti, jossa on suhteellisen pieni määrä pintoja. Monet projektit koostuvat useista objekteista ja pintojen määrä voi nopeasti kasvaa erittäin suureksi, jos jokainen objekti sisältää satoja tuhansia pintoja. Tämän takia suurimmasta osasta malleista kannattaa tehdä low polygon -malleja. Kevyet mallit ovat myös nopeampia mallintaa verrattuna objekteihin, jotka sisältävät suuren määrän pintoja. Ei ole olemassa tarkkaa määritelmää, missä vaiheessa malli ei enää ole low polygon -malli. Low polygon -mallit ovat erittäin yleisiä kaikkialla, missä 3D-malleja käytetään. Oli kyseessä renderöinti tai peliobjekti, suurin osa malleista ei tarvitse tarkkaa geometriaa varsinkin, kun hyvällä teksturoinnilla saa aikaan näyttävämpiä malleja helpommin. Kevyet mallit ovat nopeita mallintaa verrattuna malleihin, joissa on kymmeniä tuhansia pintoja. (Vendelskis 2021.) Kuvassa kuusi näytetään kahden eri pullon avaajan geometria. Ylempi malli on low polygon -versio, joka näyttää kaukaa hyvältä, kun taas alla oleva tarkempi versio näytetään pelissä läheltä. Low poly pullonavaaja koostuu 242 pinnasta ja korkea polygoninen 3768 pinnasta. Tarkempaan mallin on ainoastaan lisätty Blenderin subsurface modifikaattori, mikä lisää geometriaa automaattisesti.



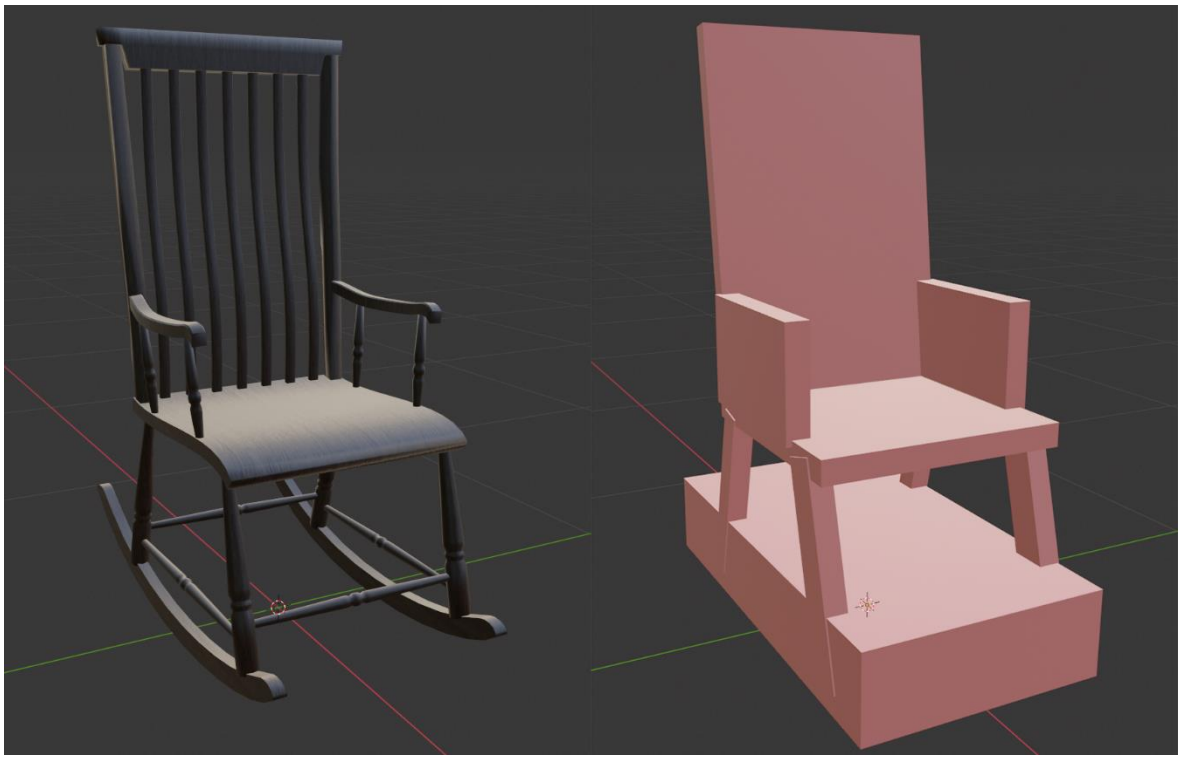
KUVA 6. Kaksi eri versiota pullonavaajasta

Pelikehityksessä monien mallien on suotavaa olla low polygon -malleja, mutta yksi tärkeä käyttö tällaisille objekteille ovat eri LOD-mallit. LOD (level of detail) tarkoittaa prosessia, missä mallista tehdään epätarkempi versio, jota käytetään, kun objekti nähdään kauempaa. Näiden entistä kevyempien mallien polygonien eli pintojen määrää, lasketaan ja tekstuurien kokoja pienennetään. Tiettyjä tekstuureja voidaan jopa poistaa malleista, pienentämällä tiedostokokoa entisestään. Tämä keventää kuormaa pelimoottorille ja pitää pelin kuvataajuuden tasaisena. LOD:it voidaan luokitella eri tasoihin. LOD0 on tarkin ja raskain versio mallista, mikä nähdään läheltä, kun taas LOD1 on kevyempi, mikä nähdään kauempaa. Jos pelin skaala sitä tarvitsee, on mahdollista luoda LOD2 ja LOD3 mallit, jotka ovat vielä epätarkempia ja tiedostokooltaan pienempiä. Tämä ei ole tarpeen peleissä, joissa pelaajan katseluetäisyys on rajattu. Jos malleja ei näe kaukaa, ylimääräisiä LOD:eja ei tarvita. Jos pelaaja katselee mallia sitä lähestyttäessä, on mahdollista huomata, kun mallin LOD vaihtuu. Tämän voi välttää suunnittelemalla pelialue uudelleen. Pitkien keskeyttämättömien näkölinjojen käyttäminen ratkaisee ongelman. LOD:in vaihtumisen voi myös kätkeä esimerkiksi sääefekteillä. Matalapolygonisista malleista voidaan myös luoda LOD:eja, koska low polygon mallille ei ole tarkkaa määritelmää. LOD:in luominen on usein helppoa. Mallinnettaessa on normaalia käyttää modifikaattoreita, jotka lisäävät automaattisesti pintoja malliin, kun mallinnus on loppusuoralla. Näillä modifikaattoreilla on usein omat asetuksensa, joilla

voi säätää lisättyjen pintojen määriä. Laskemalla näitä asetuksia tai jopa poistamalla modifikaattorin kokonaan voidaan luoda eri tasoisia LOD:eja nopeasti. Kaikissa tilanteissa LOD:eja ei kannata käyttää. Jos kyseessä on pienet tavarat tai huonekalut, on helpointa piilottaa nämä tavarat automaattisesti, kun pelaaja on riittävän kaukana peliobjekteista. Pelimoottori suorittaa tämän usein automaattisesti. (Denham a.) Kuvassa seitsämän esitellään eri LOD:eja. Numero kuvan alla viittaa malin pintojen lukumäärään.

KUVA 7. Auton kuusi erilaista LOD:ia (Game Dev Academy 2015)

Pelit tarvitsevat usein törmäysobjektit (collider) ja tähän tarkoitukseen low polygon mallit toimivat erinomaisesti. Törmäysobjekti on näkymätön elementti, jonka läpi pelaaja ei pysty liikkumaan. Pelimoottorit eivät luo törmäysobjekteja automaattisesti. Malliin liitetty törmäysobjekti on määritettävä erikseen, joko valmiin mallin mukaisesti tai erillisestä mallista. (Huston 2020.) Valmiista mallista määritettynä törmäysobjekti voi hidastaa peliä, jos malli on liian yksityiskohtainen. Pelimoottorilla on vaikeuksia laskelmoida kuinka tarkat törmäysobjektit vuorovaikuttavat toisiinsa, joten törmäysobjektit ovat hyödyllistä mallintaa erikseen. (The Open Augmented Reality Teaching Book.) Tällaisissa malleissa kannattaa käyttää mahdollisimman yksinkertaisia muotoja. Suorat yhdeksänkymmenen asteen kulmat toimivat usein parhaiten, koska silloin vältytään mahdollisilta yllätyksiltä, joita pelin fysiikkamoottori saattaa aiheuttaa. Kuvassa kahdeksan on esimerkki erikseen mallinnusohjelmassa luodusta törmäysobjektista. Keinutuolin ei kuulu keinua pelissä, joten mallin voi tehdä laatikoista. Törmäysobjektit ovat hyödyllistä värjätä selvästi, jotta ne erottuvat helposti pelimoottorissa.



KUVA 8. Esimerkki keinutuolin törmäysobjektista

Erillisten objektien lisäksi törmäysobjekteja tulee käyttää luodessa mahdollinen lattia tai maa, sekä rajattaessa pelaajan liikkeitä muuten, jotta he eivät putoa pelimaailmasta. Unreal enginen pystyy määrittelemään kappaleelle automaattisesti joko yksinkertaisen tai monimutkaisen törmäysobjektin. Yksinkertainen määritys toimii hyvin jo valmiiksi yksinkertaisille objekteille tai monille erikseen mallinnusohjelmassa luoduille törmäysobjekteille. (Epic Games Inc. c.)

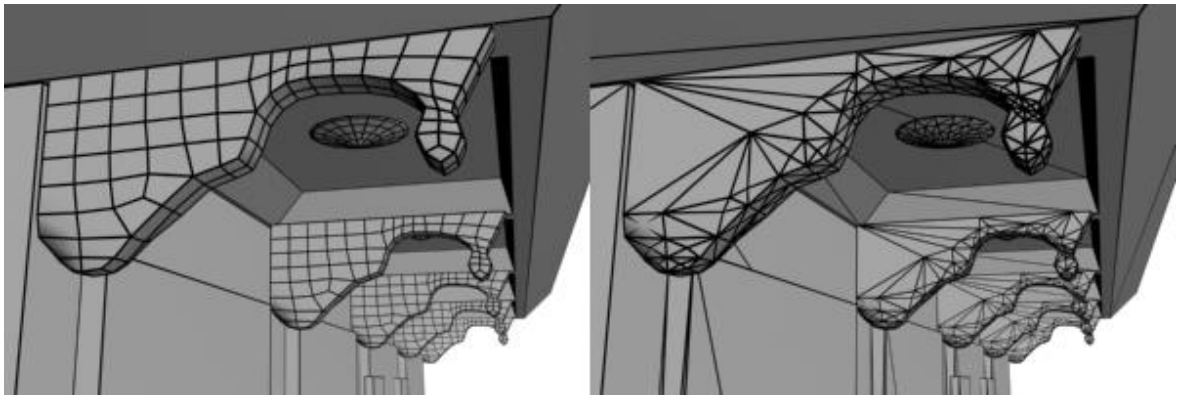
3.3 Topologia

Topologia tarkoittaa tapaa, miten kolmiulotteisenmallin verteksit ja reunat jakautuvat. Topologia siis kuvastaa mallin rakennetta. (Turbosquid.) Verteksit ja reunat muodostavat mallin pinnat, joten näiden elementtien asettelu on mallinnuksen ydin.

Toimiva topologia auttaa monin tavoin. Hyvä rakenne tekee työskentelystä helpompaa ja usein johtaa parempaan lopputulokseen. Animoidessa monet mallit muuttavat muotoaan joskus odottamattomilla tavoilla ja hyvä topologia auttaa tässäkin asiassa. Kun polygonit venyvät ja muuttavat muotoaan animaation aikana, yhtenevä ja toimiva rakenne pitää mallit yhdenmukaisen näköisinä liikkeestä huolimatta. Joskus mallit tarvitsevat muutoksia jälkikäteen tai niistä tarvitsee tehdä eri versioita. Pelimalleista tarvitaan erityisesti useita variaatioita, ja hyvä rakenne auttaa isoja ja pieniä muutoksia tehdessä. Malleihin lisätään monesti modifikaattoreita jotka muuttavat mallia automaattisesti. Esimerkiksi Blenderin subdivision

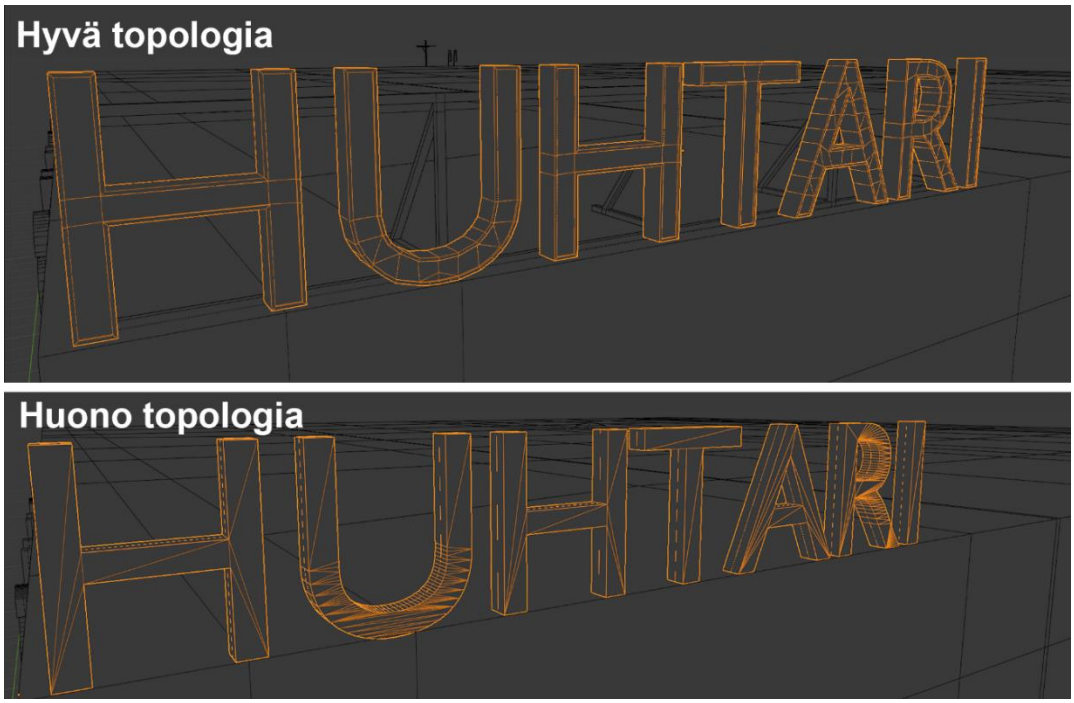
surface ja 3ds Maxin turbosmooth lisäävät geometriaa ja pehmentäen malleja ja tämä vaatii rakenteen olevan kunnossa, jotta lopputuloksesta saadaan toivottu. Monet tällaiset automaattiset työkalut olettavat, että mallin rakenne on yhtenevä. Hyvä rakenne ei pelkästään tarkoita reunojen ja verteksien paikan suunnittelua. Pintojen minimointi on myös iso osa hyvää topologiaa. Yhden suoran nelikulmaisen seinän ei tarvitse koostua kymmenistä pinnoista, kun yksi pinta johtaa samannäköiseen lopputulokseen. Pintojen minimointi tekee lopullisesta pelimoottoriin viedystä tiedostosta pienemmän, mikä on yksi keskeisimmistä asioista pelimalleissa. (Danan 2016 b.)

Mallinnettaessa on suotavaa pitää mahdollisimman monet pinnat nelikulmaisina (quads). Neljä sivua tekee polygonien muokkauksesta helpompaa, koska tällöin ohjelma pystyy esimerkiksi määrittämään kuinka reunat kiertävät objektin. On myös mahdollista pitää kaikki pinnat kolmikulmaisina (tris). Kolmesta verteksistä koostuvat polygonit voivat olla vaikeampia muokata, mutta ne toimivat hyvin esimerkiksi skulptauksessa. Lisäksi kaikki mallien pinnat muutetaan kolmioiksi automaattisesti, kun malli renderöidään joko pelimoottorissa tai mallinnusohjelmassa itsessään. Tämä onnistuu moitteettomasti, jos pinnat ovat nelikulmioita, koska tällöin kaikki pinnat jaetaan yksinkertaisesti kahtia. Mallin polygonit kannattaa myös pitää mahdollisimman samankokoisina. Samankokoiset pinnat tekevät erityisesti animaatioista paremman näköisiä, koska kaikki pinnat muuttavat muotoaan samalla tavalla. Joitain mallinnusmenetelmiä käytettäessä pintojen kokoon ei kuitenkaan tarvitse kiinnittää huomiota. Hyvä topologia ei pelkästään tee mallinnuksesta helpompaa. Kun mallin pinnoista on luotu kaksiulotteinen kuvaus teksturointia varten, hyvä topologia nopeuttaa työskentelyä. Kaikki ylimääräiset pinnat hidastavat prosessia ja tekevät mallin hahmottamisesta vaikeampaa, kun sen pintoja tarkastellaan kaksiulotteisena kuvana. (Danan 2016 b.) Kuvassa yhdeksän mallit ovat periaatteessa samanlaiset, mutta vasemmanpuoleisen mallin topologia on kaikin tavoin parempi, koska malli koostuu tasaisesti nelikulmaisista pinnoista.



KUVA 9. Esimerkki kahdesta erilaisesta topologiasta (Turbosquid)

Topologia ei ole asia, jonka pystyy aina korjaamaan jälkikäteen. Hyvän topologiaa on seurattava alusta asti ja laiminlyötynä aiheuttaa ongelmia projektin varrella. Blenderin tekstityökalulla luotu kyltti muodostuu pelkistä kolmioista, joiden kanssa on vaikea työskennellä. Lisäksi mallissa on monia turhia polygoneja. Onneksi Blenderin retopology -työkalu muuttaa suurimman osan pinnoista nelikulmioiksi. Vaikka tämä lisää ison määrän pintoja, on ylimääräisten pintojen poistaminen helppoa. (Blender.) Kun mallin rakenne on korjattu, pystyy mallia jälleen muokkaamaan. Kuvassa 10 topologia on korjattu Blenderin retopology työkalulla.

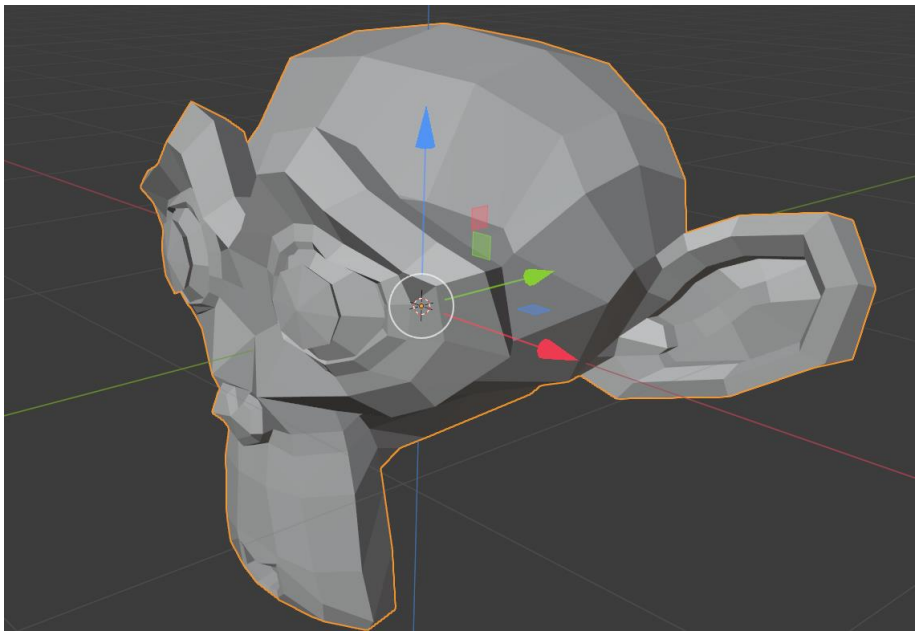


KUVA 10. Blenderin tekstityökalulla luodun kyltin topologian korjaus

3.4 Exporttaus ja tiedostoformaatit

Jotta malli saadaan käyttöön pelimoottorissa, on se tallennettava muotoon, jonka muut ohjelmat tunnistavat. Tätä exporttaus prosessia varten mallin tulee olla valmis käytettäväksi, koska monia asioita ei pysty muokkaamaan pelimoottorissa tai muutoksien tekeminen on hidasta. Teksturointiin pystyy vaikuttamaan pelimoottorissa mutta geometrian muokkaus on ongelmallisempaa. Exporttaessa on mahdollista määrittää, mitkä kappaleet kuuluvat lopulliseen malliin. Kappaleiden yhdistäminen voi olla hyödyllistä, jos kyseessä on yksi kokonaisuus, kun taas malleissa, jotka koostuvat monista mahdollisesti siirreltävistä asioista, kannattaa mallin osat pitää erillään. Jos tiedostossa on monia eri malleja, on näiden eri osien poistaminen mahdollista myöhemmin pelimoottorissa.

Eri objektien yhdistämisen lisäksi mallille on määritettävä skaala ja akselit, joille malli asettuu. Blender mallinnusohjelman mukaan Z-akselin on korkeus ja Y-akselin on mallin etu- ja takaosa. Unity pelimoottori puolestaan tulkitsee positiivisen Z-akselin kuvan etuosana ja Y-akselin korkeutena. (Polynook.) Unreal Engine tulkitsee koordinaatiston samoin kuin Blender (Emperore, Sherry). Näiden eroavaisuuksien takia kannattaa mallia exporttaessa määrittää, mille akseleille malli asettuu. Kuvassa 11 esitellään kuinka Blender määrittää akselit oletuksena. Y-akseli on vihreä, Z-akseli on sininen ja X-akseli on punainen.



KUVA 11. Koordinaatisto Blenderissä

Akseleiden lisäksi on määritettävä mallin skaala ja paikka maailmassa. Mallinnusohjelmalle pystyy määrittämään, mitä mittayksiköitä ohjelma käyttää. Tämän skaalan pystyy muuttamaan jälkikäteen, mutta jo työskentelyä aloittaessa on hyvä pitää mielessä, onko malli kaksi metriä vai 2 senttimetriä korkea. Mallinnuksen lopussa on myös määritettävä, mikä on mallin origin piste. Tämän on piste, jonka ympärille malli muodostuu ja jonka ympäri malli pyörii ja skaalautuu. (Polynook.) Mallin origin on verrattavissa esimerkiksi videoeditointiohjelmissa oleviin ankkuripisteisiin.

Exporttauksessa on määritettävä mallille tiedostoformaatti. Tiedostoformaatti määrittää, mikä mallin data tallennetaan eteenpäin. 3D-mallinnuksessa yleisimmät tiedostoformaatit FBX ja OBJ. OBJ on 80-luvulla kehitetty 3D-formaatti, mutta alkuperäinen ohjelmisto, joka käytti tätä tiedostomuotoa, on vanhentunut. OBJ formaatti on kevyt ja helppolukuinen, joten erittäin monet ohjelmat tukevat tätä tiedostomuotoa. OBJ on tekstipohjainen formaatti, mikä tekee datan pakkauksesta epätehokasta. Isot mallit tekevät OBJ tiedostoista erityisen

raskaita, joten niiden lataus ja prosessointi esimerkiksi Unreal Engine pelimoottorissa on hidasta. Lisäksi OBJ formaatti tallentaa vain mallin materiaali- ja geometriadatan. Näin ollen tätä formaattia ei pysty käyttämään animoiduissa objekteissa. (Houston 2019 a.)

FBX on puolestaan mallinnusohjelmien ja pelimoottorien käyttämä standardiformaatti monipuolisuutensa ansiosta. FBX-tiedosto tukee mallien lisäksi valoja, animaatioita ja materiaaleja. FBX on tehokas formaatti, jonka tiedosto koko ei kasva samalla tavalla kuin OBJ formaatin. Tämä tiedostotyyppi on kuitenkin Autodeskin omistuksessa, joten sen käyttö voi olla rajoitettua joissain rajatapauksissa. (Houston 2019 b.)

4 Teksturointi

4.1 Teksturoinnin perusteet

Yksityiskohtainen geometria ei yksinomaan tee mallista ammattimaisen näköistä. Vaikka mallinnusohjelmissa on monesti toimivat materiaalieditorit, joilla malleihin saa esimerkiksi kiiltoa sekä haluamansa värityksen, lopputulos ei usein ole yhtä näyttävä verrattuna teksturointiin. Mallinnusohjelmien materiaalieditorien avulla on mahdollista muuttaa esimerkiksi mallin väriä (base color), metallisuutta (metalness) tai karheutta (roughness). Teksturointi on kuitenkin tehokkaampi työkalu, jolla näitä elementtejä pystyy säätämään tarkemmin. Tekstuuri on kuvatiedosto, joka asetetaan mallin pinnalle, tehden mallista todenmukaisemman näköisen. (Petty.) Tekstuurit ovat yleensä JPG tai PNG formaattisia kuvia, jotka voivat olla kooltaan eri kokoisia tarpeesta riippuen. On olemassa monia eri tekstuureja, joiden väridatan avulla on mahdollista lisätä monia eri yksityiskohtia automaattisesti lisäämättä geometriaa. Monet ominaisuudet, mihin mallinnusohjelman materiaalieditorilla pystyy vaikuttamaan, voidaan määrittä tarkasti eri tekstuureilla. Tällä tavoin on mahdollista määrittää esimerkiksi, mitkä mallin alueet erityisesti heijastavat valoa. Kuvassa 12 näytetään miltä puinen väritekstuuri näyttää pianossa.

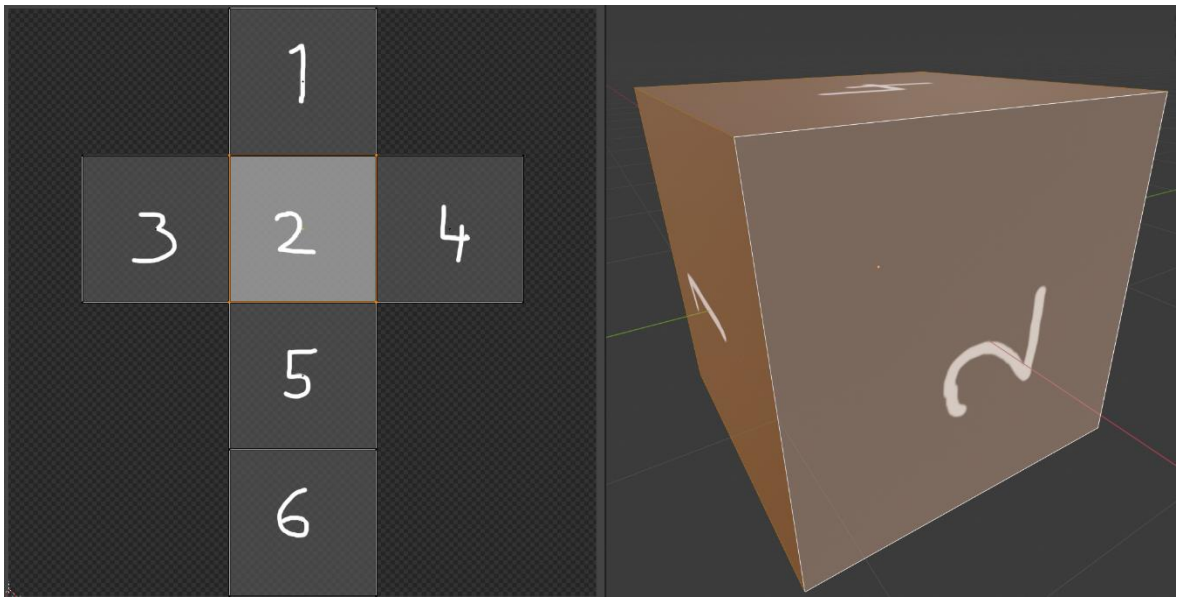
KUVA 12. Piano ja sen puu tekstuuri

Oli kyseessä animaatio tai peliobjekti, tekstuurien käyttö on todella yleistä. Tekstuurien koko on kuitenkin yksi olennainen ero. Tekstuurit ovat osa mallia, joten ne nostavat tiedostokokoa usein jopa enemmän kuin mallin geometria. Tämän takia on tiedettävä mihin tarkoitukseen mallia käytetään. Ei ole järkevää määrittää neljää isoa 4k-tekstuuria mallille, mihin pelaaja

ei kiinnitä huomiota tai kaikkiin malleihin, joita asetellaan ympäri pelimaailmaa. Toinen asia mikä on huomioitava pelimalleja teksturoidessa, on tekstuurin tarkka resoluutio. Kaikki pelimoottorit prosessoivat dataa tietyllä tavalla, mikä vaatii tekstuurien olevan kahdeksalla jaollisia. Tämä tarkoittaa, että tekstuurien on oltava kooltaan esimerkiksi 8, 16, 32, 64, 128, 512, 1024, 2048 pikseliä korkeita ja leveitä. Pelimoottorit tukevat isompiakin tekstuureja. Jos tekstuuri ei seuraa tätä sääntöä, joutuu pelimoottori korjaamaan tekstuurin itse, mikä ei välttämättä näyttäydy ongelmana, mutta vie kuitenkin prosessointi aikaa. Pelimoottorit korjaavat yksittäiset tekstuurit nopeasti, mutta jos kaikki pelin tekstuurit ovat väärän kokoisia, voi tämä hidastaa peliä huomattavasti. Pelimoottorin korjaama tekstuuri voi näyttää suttuiselta, jos huomattava osa tekstuuria on korjattava. (KatsBits.) Animaatioissa ja muissa rendereissä tekstuurien kokoa ei tarvitse aina rajoittaa, vaikka isot kuvatiedostot saattavatkin hidastaa ohjelman toimintaa.

4.2 UV-kartoitus

Tekstuuri on kaksiulotteinen kuvatiedosto ja mallit ovat kolmiulotteisia kappaleita, joten kuva tekstuurin projisointi mallin ympärille ei aina ole yksinkertaista. Tähän tarkoitukseen mallille on luotava UV-kartta. UV-kartta on kaksiulotteinen kuvaus kaikista mallin pinnoista, jonka mukaan tekstuuri asettuu mallin ympärille. U ja V viittaavat kuvan horisontaaliseen ja vertikaaliseen akseliin. UV-kartan luontia varten mallille on suoritettava UV-unwrap. Tämä on automaattinen prosessi, joka luo UV-kartan mallista toivottujen asetusten mukaan. Oletus asetukset toimivat ongelmitta erittäin yksinkertaisille malleille, mutta monille malleille täytyy määrittää saumakohdat ja jopa muuttaa UV-karttaa manuaalisesti. Yksi tekstuuri kattaa yleensä koko mallin värityksen, joten eri alueiden jäsentely UV-kartassa helpottaa tekstuurin luontia. Saumat määrittävät, mitkä mallin alueet ovat liitoksissa toisiinsa UV-kartassa. Tämän takia on hyödyllistä määrittää saumat alueiden ympärille, joiden on tarkoitus näyttää samalta. Esimerkiksi rakennuksen tiiliseinän rajat voidaan määrittää saumakohdiksi UV-kartassa. Blender ohjelmassa on mahdollista projisoida valitut alueet primitiivisten muotojen mukaan, kuten kuution tai sylinterin. Jos mallin muoto on lähellä näitä valmiita muotoja, on tämä nopea keino määrittää UV-kartta automaattisesti. (Denham c.) Kuvassa 13 on yksinkertaisen kuution UV-kartta. UV-kartassa kuution pinnat avautuvat loogisesti kaksiulotteiselle tasolle.



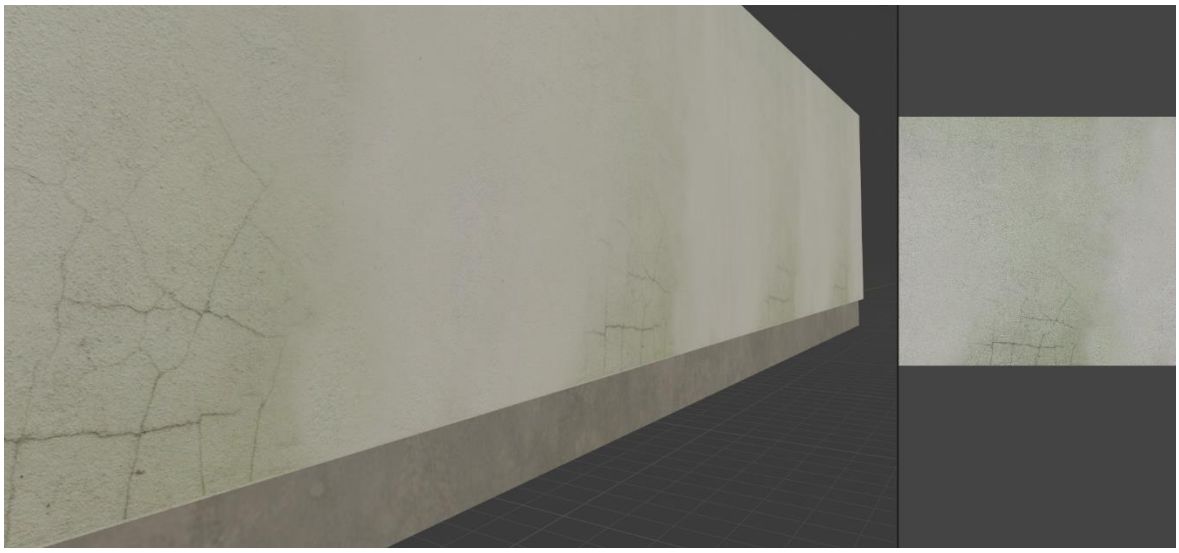
KUVA 13. Kuution UV-kartta

UV-kartassa eri pintojen ympärille on suotavaa usein jättää hieman tilaa, jos kyseisten pintojen on tarkoitus näyttää erilasilta. Tämä minimoi virheet kuvatiedostoa luodessa, koska eri alueiden väritykset eivät vuoda muille alueille helposti.

4.3 Värikartta

Värikartat (diffuse map) ovat tekstuureja, jotka lisäävät värin mallille. Näiden tekstuurien luominen ja käyttö on yksinkertaista verrattuna muihin tekstuureihin. Tekstuurin on oltava tasaisesti valottunut ja kuvan perspektiivin on oltava suoraan edestä. (Johnson 2014.)

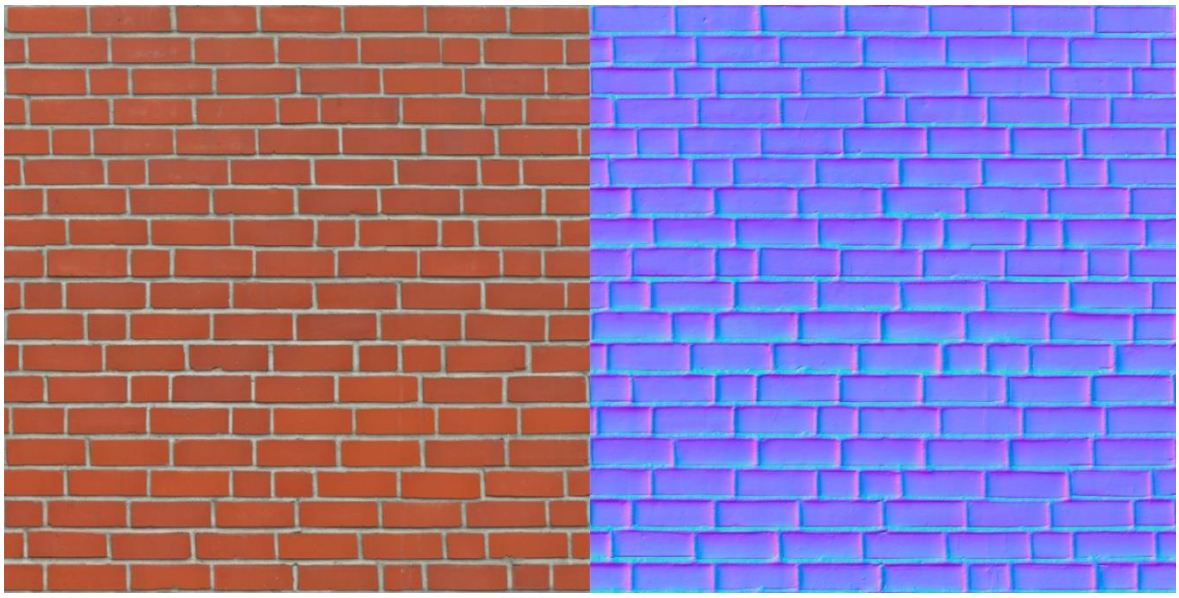
Tekstuurien uudelleen käyttö on helppo tapa pienentää tiedostokokoa, kun sama kuvio toistuu mallissa useaan kertaan. Tällöin kuvasta kannattaa poistaa tiettyjä yksityiskohtia, jotta katselija ei huomaa kuinka sama kuva toistuu mallissa. Esimerkiksi kiviseinää luodessa värikartasta on suotavaa poistaa silmäänpistävät halkeamat. Kuvassa 14 tekstuurin toistuvuuden huomaa heti.



KUVA 14. Epäonnistunut toistuva tekstuuri

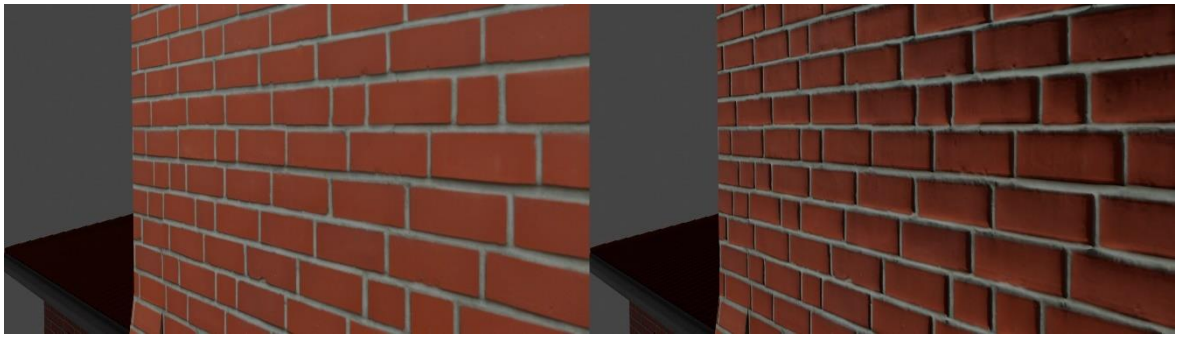
4.4 Normaalikartta

Värityksen lisäksi tekstuurit voivat muokata ulkonäköä tarkemmin. Normaalikartoitus on keino, millä lopputulokseen saadaan lisää yksityiskohtia kasvattamatta polygonien määrää tekemällä värikartan yksityiskohdista kolmiulotteisen näköisiä. Normaalikartta yhdistetään usein vastaavaan värikarttaan. Nämä tekstuurit koostuvat violetin ja sinisen sävyistä. (Pluralsight 2014.) Kuvassa 15 näytetään kuinka väri ja normaalikartat vastaavat toisiaan.



KUVA 15. Tiiliseinän väri ja -normaalikartta

Normaalikartta määrittää kuvan väridatan avulla, mihin suuntaan valo heijastuu mallin pinnasta. Jokaisella pikselillä on oma normaalinsa, eli suunta mihin ne osoittavat ja normaalikartta määrittää mihin suuntaan nämä normaalit osoittavat. Tämä tekee mallista realistisimman näköisiä lisäämättä polygonien määrää. Normaalikarttaa käytettäessä objektin siluetti pysyy samana, joten läheltä katsottuna malli voi näyttää epätarkalta. (Pluralsight 2014.) Kuvassa 16 verrataan mallia ilman normaalikarttaa ja normaalikartan kanssa. Valonlähde valaisee pelkää värikarttaa tasaisesti, kun puolestaan normaalikartta lisää tilleihin syvyyttä ja varjoja. Oikeanpuoleisessa kuvassa huomaa myös, kuinka rakennuksen kulma on tasainen. Normaalikartta ei siis muokkaa geometriaa ja pitää mallin siluetin samana.

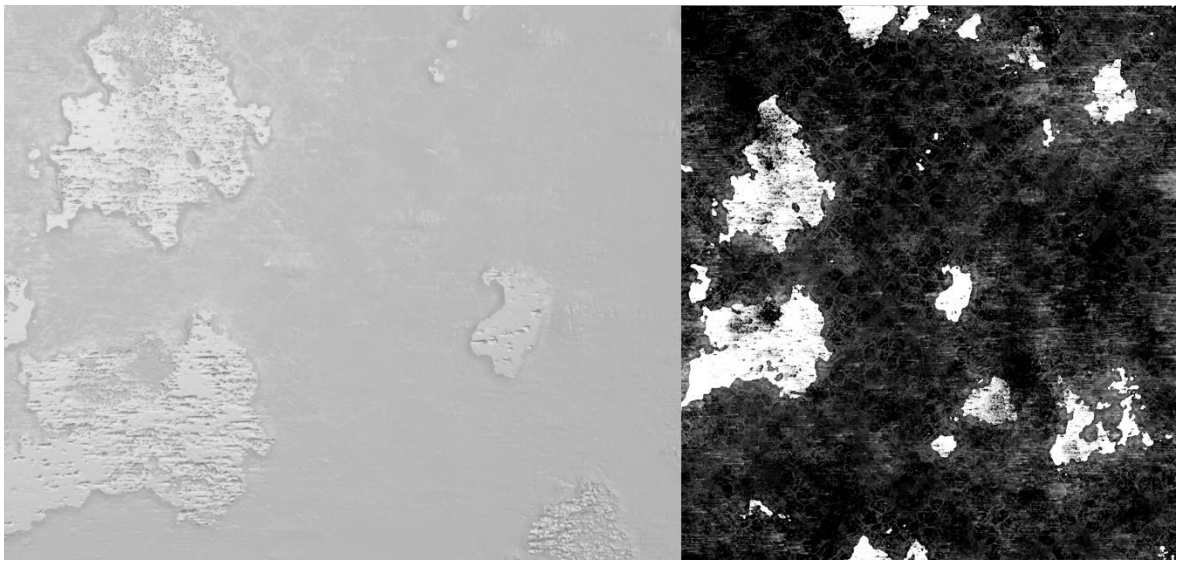


KUVA 16. Normaalikartan vaikutus malliin

4.5 Muita tekstuureita

On olemassa muita tekstuureja, joiden väridataa käytetään erilaisiin tarkoituksiin. Tämäntapaisilla tekstuureilla voidaan muokata esimerkiksi mallin metallisuutta tai jopa geometriaa liikuttamalla verteksien paikkaa automaattisesti.

Metallisuuskartta (metalness map) määrittää, mitkä mallin alueet ovat metallisia ja mitkä eivät. Nämä tekstuurit ovat mustavalkoisia kuvatiedostoja, joiden vaaleat alueet määrittävät mitkä alueet tulkitaan metalli pintoina. (Rose 2019.) Kuvassa 17 vasemmalla on renderöity kuva maalatusta metalli pinnasta ja oikealla on siihen käytetty metallisuuskartta. Metallisuustekstuurissa maalamattomat alueet ovat valkoisia, joten nämä alueet tulkitaan metalli pinnoiksi. Valo heijastuu renderöidyssä kuvassa näillä alueilla eri tavalla.



KUVA 17. Esimerkki metallisuuskartan vaikutuksesta

Siirtymäkartta (displacement map) on tekstuuri, joka muuttaa mallin geometriaa automaattisesti kuvatiedoston väridatan mukaisesti. Nämä tekstuurit ovat mustavalkoisia kuvia, minkä perusteella mallinnusohjelma korottaa valkoisia alueita ja laskee tummia alueita. Koska siirtymäkartta liikuttaa mallin verteksejä, on mallissa oltava paljon pintoja, jotta tekstuuri pystyy muokkaamaan mallin geometriaa tarkasti. (Pluralsight 2014.) Siirtymäkarttojen käyttöä kannattaa rajoittaa, koska malleista usein tulee erittäin suuria. Kuvassa 18 siirtymäkartta muokkaa mallia erittäin yksityiskohtaisesti. Tummimmat alueet ovat painautuneet eniten, kun puolestaan vaaleat alueet ovat nousseet.

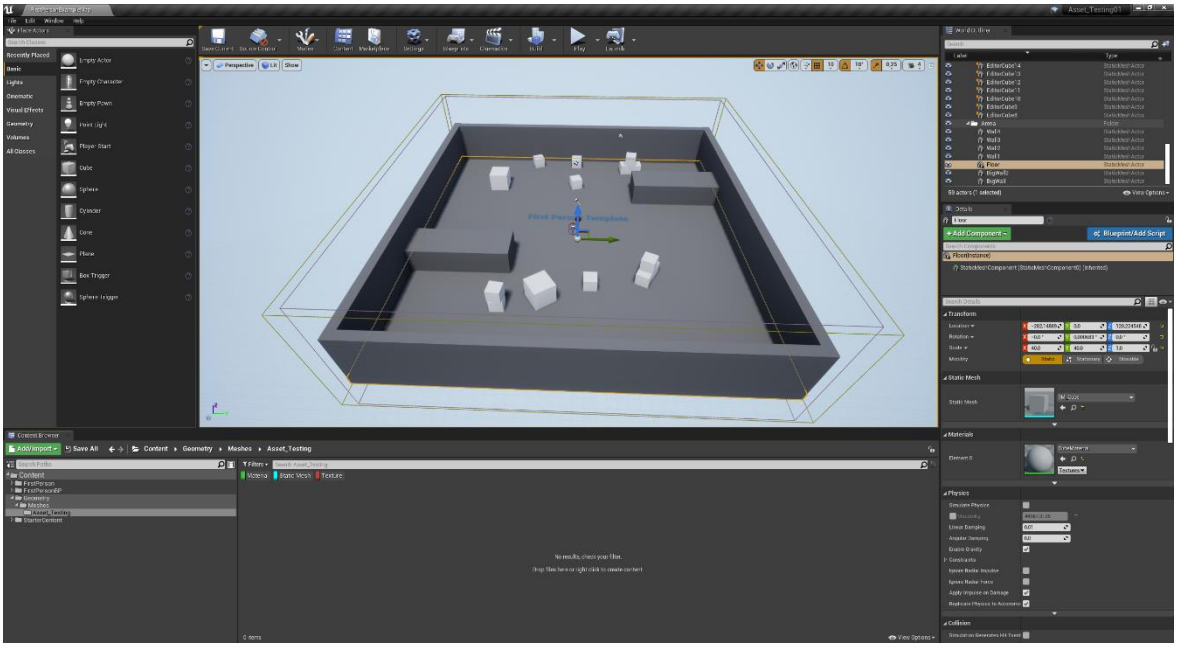
KUVA 18. Siirtymäkartan vaikutus malliin

Eri tekstuureja määrittäessä on mahdollista säätää kuinka paljon ne vaikuttavat malliin. Jos normaalitekstuurin luomat painaumat ovat liian äärimmäisiä tai siirtymäkartta ei kohota haluttuja alueita riittävästi, on näitä mahdollista muokata mallinnusprosessin aikana tai pelimoottorissa. Esimerkiksi normaalikartassa on voinut tapahtua sekaannus ja väärät alueet lasketaan painaumiksi. Tällöin normaalikartan pystyy muuttamaan käänteiseksi mallinnusohjelmassa tai Unreal Enginessä. (Erell 2020.)

5 Unreal engine

5.1 Tausta

Unreal Engine on Epic Gamesin kehittämä pelimoottori. Ensimmäinen versio julkaistiin vuonna 1998 ja tämänhetkinen versio, Unreal Engine 4, julkaistiin vuonna 2020. UE4 käyttää C++ ohjelmointikieltä, sekä visuaalista ohjelmointijärjestelmää nimeltä Blueprint. Blueprint on node pohjainen järjestelmä, joka auttaa pelin logiikan muodostuksessa. Blueprint on erityisen hyödyllinen kehittäjille ketkä eivät ole kokeneita ohjelmoijia. UE4 on tunnettu sen fotorealismista, ja tämän takia sitä käytetään usein myös peliteollisuuden ulkopuolella. (Dealessandri 2020.) Kuvassa 19 on esimerkki Unreal Engine 4 käyttöliittymästä. Tällainen valmis tyhjä projekti on hyvä ympäristö mallien tarkasteluun ja testaukseen.



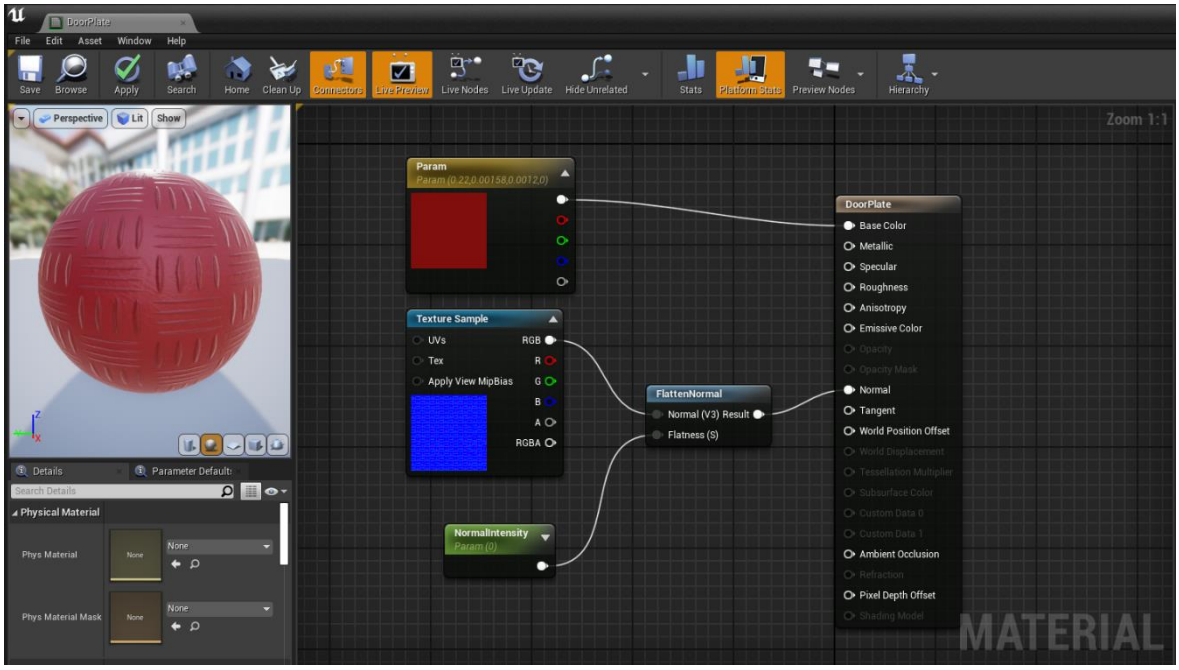
KUVA 19. Unreal Engine 4 käyttöliittymä

Pelin kehittäminen on ilmaista Unreal Enginellä, kunnes vuosineljänneksen tuotto ylittää 3000 dollaria. Tämän jälkeen Epic Games ottaa viiden prosentin rojalti maksun pelin tuotoista. (Dealessandri 2020.)

5.2 Materiaalit

Unreal Engine pystyy määrittämään peliobjekteille materiaalit. Näiden materiaalien ominaisuuksia, kuten metallisuutta ja heijastuneisuutta, pystyy muuttamaan samalla tavoin kuin mallinnusohjelmissa. Muutettavat ominaisuudet täytyy määrittää parametreiksi, jotta

muokkaus on nopeaa. Täten Unreal Engine pystyy muokkaamaan näitä materiaali instansseja (material instance) reaaliaikaisesti kokoamatta sitä uudelleen. (Epic Games Inc. b.) Kuvassa 20 on Unreal Engine 4 materiaalieditori, jossa normaalikartan vahvuus on määritetty parametriksi.



KUVA 20. Unreal Engine materiaalieditori

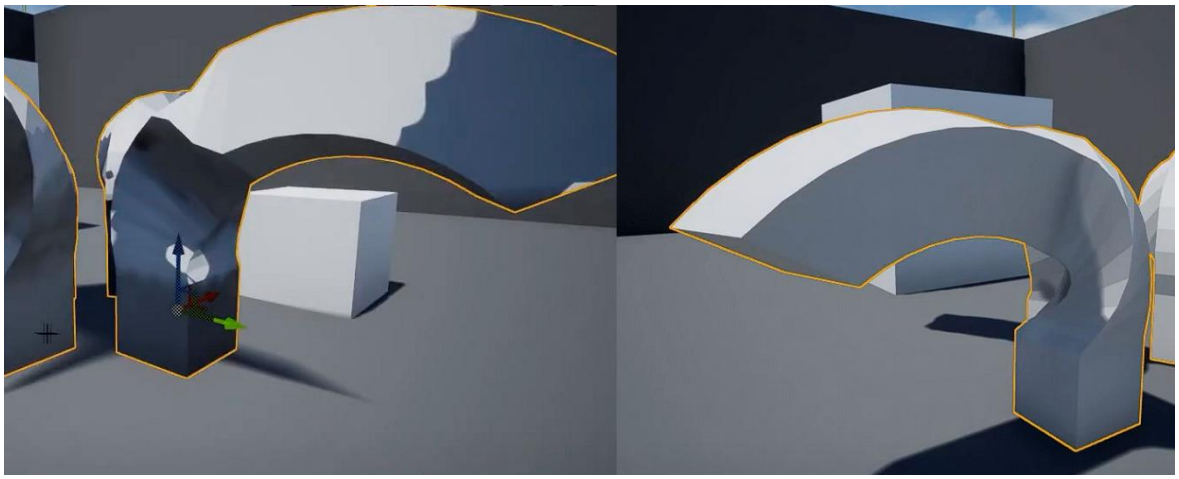
Tietyt materiaalit ovat mahdollista jättää vakioiksi. Näitä materiaaleja ei pysty muokkaamaan kesken pelin, mutta ovat silti hyödyllisiä. Esimerkiksi maalattu metalli voi olla autojen oletusmateriaali, jonka väriä tai karkeutta voi säätää tarpeesta riippuen. (Epic Games Inc. b.)

Malliin on myös mahdollista lisätä siirtokuvia (decal). Nämä ovat tarran omaisia kuvia, joilla voi lisätä yksityiskohtia haluttuihin kohtiin muuttamatta mallin tekstuureja tai UV-karttaa. Siirtokuvat ovat irrallisia tekstuureja, jotka toimivat hyvin esimerkiksi julisteina tai tahroina. (World of Level Design 2016.) Kuvassa 21 oksennus ja kulunut tapetti ovat luotu siirtokuvalla.

KUVA 21. Siirtokuvan käyttö Unreal Engine 4 pelimoottorissa

5.3 Valokartta

UV-kartta määrittä kuinka tekstuurit asettuvat malliin. Tämä tarkoittaa myös kuinka ulkopuoliset valot ja varjot vaikuttavat malliin. Jos mallissa on toistuvia tekstuureja, eli UV-kartan alueet ovat päällekkäin, varjot voivat olla väärissä paikoissa tai malli voi olla täysin varjojen peittämä. Näin ollen mallille on luotava valokartan UV unwrap. Tämä on verrattavissa tekstuurin unwrap prosessiin, mutta ei kuitenkaan vaikuta mallin tekstuureihin. Unreal Engine jättää automaattisesti UV-kartan alueille riittävät välit, jotta varjot eivät toistu mallissa. (Epic Games Inc. a.) Kuvassa 22 vasemmanpuoleiselle mallille ei ole tehty valokartan unwrap prosessia, joten mallin varjot ovat luonnottoman näköisiä.

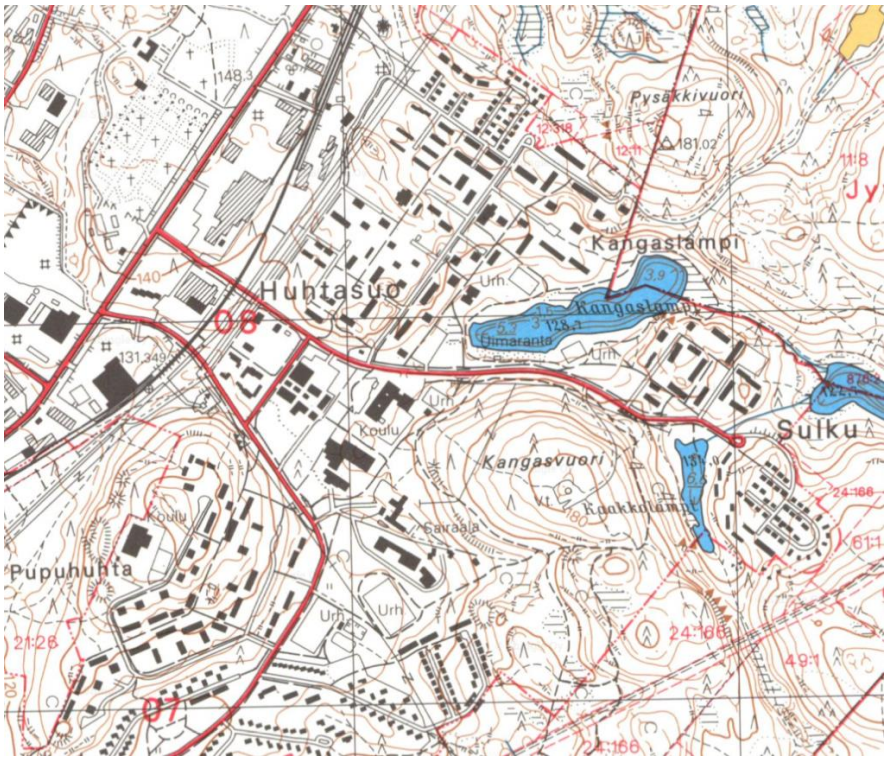


KUVA 22. Epäonnistunut ja toimiva valokartta 3D-mallissa (Markom3D 2019)

6 Case: Last Drop -peli

6.1 Esittely ja tavoitteet.

Last drop on alkoholismista kertova 3D-seikkailupeli, joka sijoittuu 90-luvun Jyväskylään. Peli luodaan Unreal Engine -pelimoottorissa ja pelinkehittäjänä toimii jyväskyläläinen video ja- 3D-tuotannon yritys Pixtell Oy. Pelikehityksen aikana keskeiset ideat ovat yksinkertaisuus, retrous ja realismi. Yksinkertaisuus tulee näkymään pelin erinäisiissä mekaniikoissa. Pelin järjestelmät ja maailman toiminta on oltava helposti ymmärrettävä kaikille pelaajille. Tämä yhdistyy myös realismiin. Mekaniikkojen ja pelimaailman on oltava yksinkertaisia ja realistisia, jotta pelin teema pysyy vakavana. Alkoholismi on aihe, joka näytetään mediassa usein koomisena, mutta tämä ei ole pelin tarkoitus. Pelin on tarkoitus olla moderni versio klassisista seikkailupeleistä. Näin ollen Last Drop pelissä kaikkia ongelmia ei ratkaista väkivallalla. Retrous näkyy myös 90-luvun Jyväskylässä, minne peli sijoittuu. Kuvassa 23 näytetään pelialue.



KUVA 23. Last Drop pelialue (Vanhatkartat 2021)

Pelin alueena toimii Jyväskylän kaupunginosa Kangaslampi. Peliympäristön tulee kuvastaa pelin teemaa, mikä tuo lisää haasteita mallintajalle. Teeman lisäksi alueet tukevat myös

pelin tarinaa esimerkiksi, kun elämä hymyilee pelin alussa, näkyy se myös ensimmäisessä alueessa, kun taas myöhemmät pelin ympäristöt saattavat olla huonommassa kunnossa.

3D-mallintajan tavoitteena on mallintaa suurin osa pelimaailmasta. Rakennukset, huonekalut, tavarat ja kulkuneuvot ovat luotava pelimaailmaan, mutta muut objektit kuten puut ja tiet luodaan suurimmaksi osin Unreal Enginen omilla työkaluilla. Monet animaatiot ja kaikki hahmot luodaan myös pelimoottorissa tai erillisellä ohjelmalla. Partikkelien simulointiin käytetään Houdini -ohjelmaa ja hahmot luodaan MetaHuman creator -ohjelmalla.

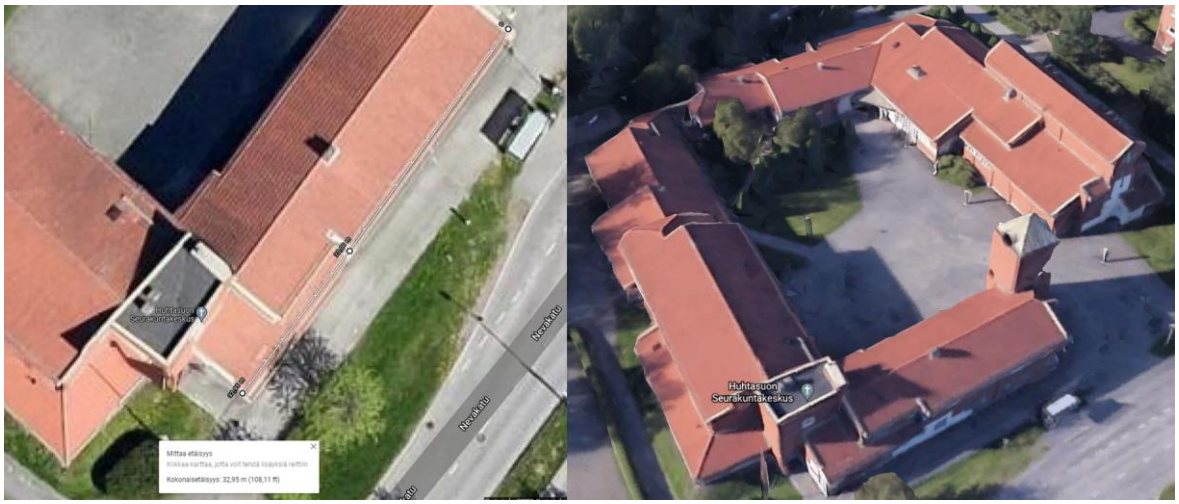
6.2 Suunnittelu

Realistinen 90-luvun Jyväskylä tuo mukanaan monia haasteita 3D-mallintajalle. Monet nykyiset Kangaslammen rakennukset eivät olleet olemassa 90-luvulla, joten referenssikuvien löytäminen osoittautui ongelmalliseksi. Jyväskylän karttapalvelut kuitenkin tarjoavat karttoja ja ilmakuvia 90-luvun Jyväskylästä. Nämä auttavat pelialueen suunnittelussa, mutta eivät auta itse mallien luonnissa. Onneksi alueella on silti 90-luvun aikaisia rakennuksia. Huonekalujen ja muiden tavaroiden mallinnus on yksinkertaista paitsi, jos kohteet ovat tekijänoikeus suojan alla. Tämä tarkoittaa, että malleja on muokattava erinäköisiksi. Malli täytyy silti olla tunnistettava ja uuden logon on oltava lähellä alkuperäistä, jotta maailma näyttää nostalgiselta.

Monimutkaisemmista malleista kuten rakennuksista on hyödyllistä tehdä yksinkertaiset versiot, jotta mallinnus on jatkossa helpompaa. Pelin mallit ovat usein low polygon -malleja mutta suunnittelu vaiheessa mallit voivat olla vielä yksinkertaisempia. Tämä suunnittelu versio voi olla raaka muoto koko mallista tai vain pieni osa sitä. Suunnitelmassa eri osien sommittelu ja skaalan määritys voivat nopeuttaa myöhempää mallinnusprosessia.

6.2.1 Ilma- ja satelliittikuvien käyttö

Jyväskylän karttapalveluiden tarjoamat resurssit, sekä Google Earth ja Google Maps Street View ovat arvokkaita suunniteltaessa ja mallinnettaessa. Nämä auttavat mallintajaa hahmottamaan rakennusten skaalan ja yleiset piirteet. Kuvassa 24 näytetään kuinka Google Mapsin avulla voi laskea rakennuksen mittoja ja miten Google Earth auttaa hahmottamaan rakennuksen eri perspektiiveistä. Earth auttaa erityisesti seurakuntakeskuksen harjakattojen mallinnuksessa.



KUVA 24. Huhtasuon seurakuntakeskus (Google)

Molemmat satelliittikuvat auttavat eri tavoin. Mapsin ylhäältä otettu kuva auttaa mittojen määrityksessä, vaikka kuvat eivät ole täydellisestä perspektiivistä tätä varten. Tämä on helppo kompensoida mallinnettaessa. Googlea Earth satelliitti kuvat ovat puolestaan epätarkempia mutta auttavat monimukaisempien rakennusten hahmottamisessa. Vapaasti liikuteltava satelliittikuva tukee hyvin staattisia Google Maps kuvia. Näiden resurssien avulla on mahdollista mallintaa kohteita etänä.

6.2.2 Mallikuvat

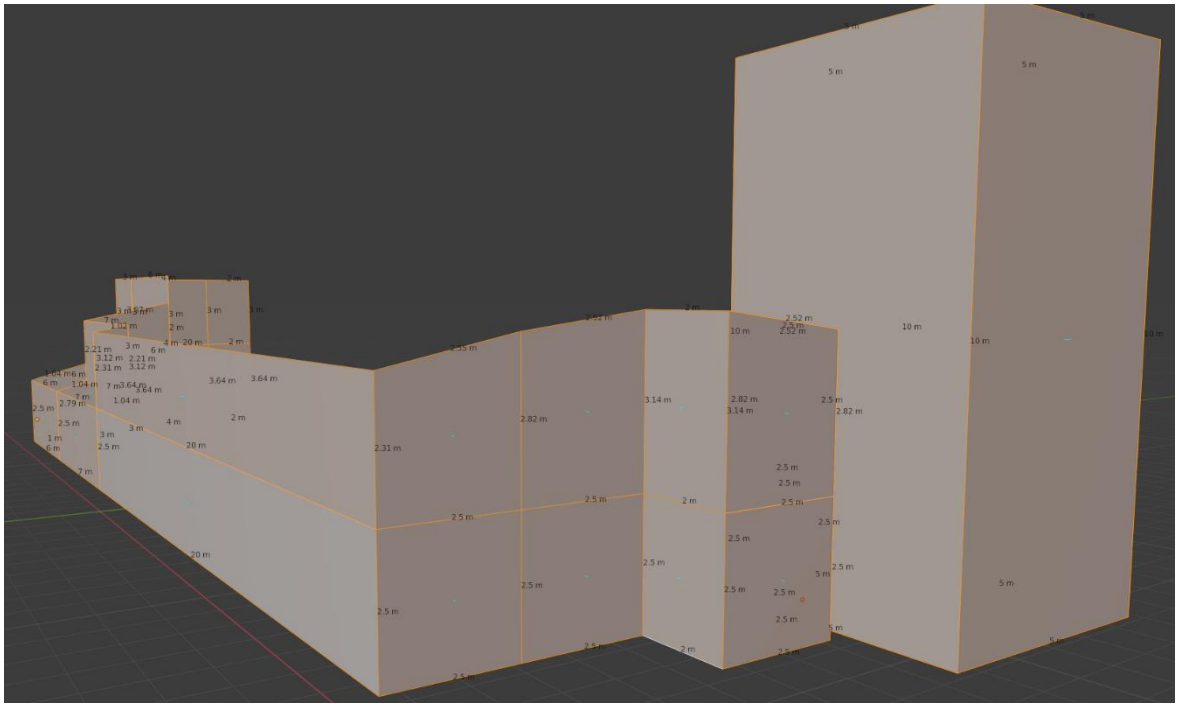
Googlen tarjoamat palvelut ovat hyödyllisiä mutta itse otetut mallikuvat näyttävät yksityiskohdat paremmin. Esimerkiksi rakennuksien ovet, ikkunat ja muut yksityiskohdat eivät näy satelliittikuvissa. Kuvassa 25 näytetään Huhtasuon kirkon mallikuva sekä 3D-malli. Ovien ja ikkunoiden määritykseen tarvitaan mallikuvia.

KUVA 25. Kirkon mallikuva sekä 3D-malli

Kyseinen kirkko näkyy suoraan edestä Google Mapsin Street View -näkymässä, joten sisäpihan kuvat olivat erityisen hyödyllisiä. Mallikuvia ei kuitenkaan aina kannata seurata liian orjallisesti. Peliobjekteja tehdessä täytyy tietää mihin käyttöön malli tulee. Esimerkiksi Huhtasuon seurakuntakeskuksen tornista on hyvä tehdä hieman suurempi, jotta rakennus toimii paremmin maamerkkinä Last Drop -pelissä.

6.3 Seurakuntakeskuksen mallinnus

Mallikuvien tarkastelun jälkeen mallinnus aloitettiin luomalla low polygon -suunnitelma. Rakennus on yksinkertaista mallintaa seinä kerrallaan, määrittämällä pituudet ja lisäämällä yksityiskohdat kuten ikkunat myöhemmin. Suunnitelman tarkoituksena on pääasiassa hahmottaa rakennuksen skaala ennalta määritettyjen mittojen avulla. Ovilla on standardimitat, joiden avulla voi määrittää suurin pirtein monia seinien pituuksia. Seurakuntakeskus on osittain kaksikerroksinen, joka määrittää rakennuksen korkeuden. Näitä standardimittoja on seurattava, kosta muut mallit mallinnetaan samojen mittojen mukaan. Mittavirheiltä ei voi kuitenkaan välttyä, ja seurakuntakeskuksen kohdalla oli yksinkertaista pyöristää mitat ylöspäin, koska rakennus on keskeisellä paikalla ja keskuksen tornin on tarkoitus olla nähtävissä kaukaa. Kuvassa 26 kirkon korkeus on määritetty huonekorkeuden mukaan.



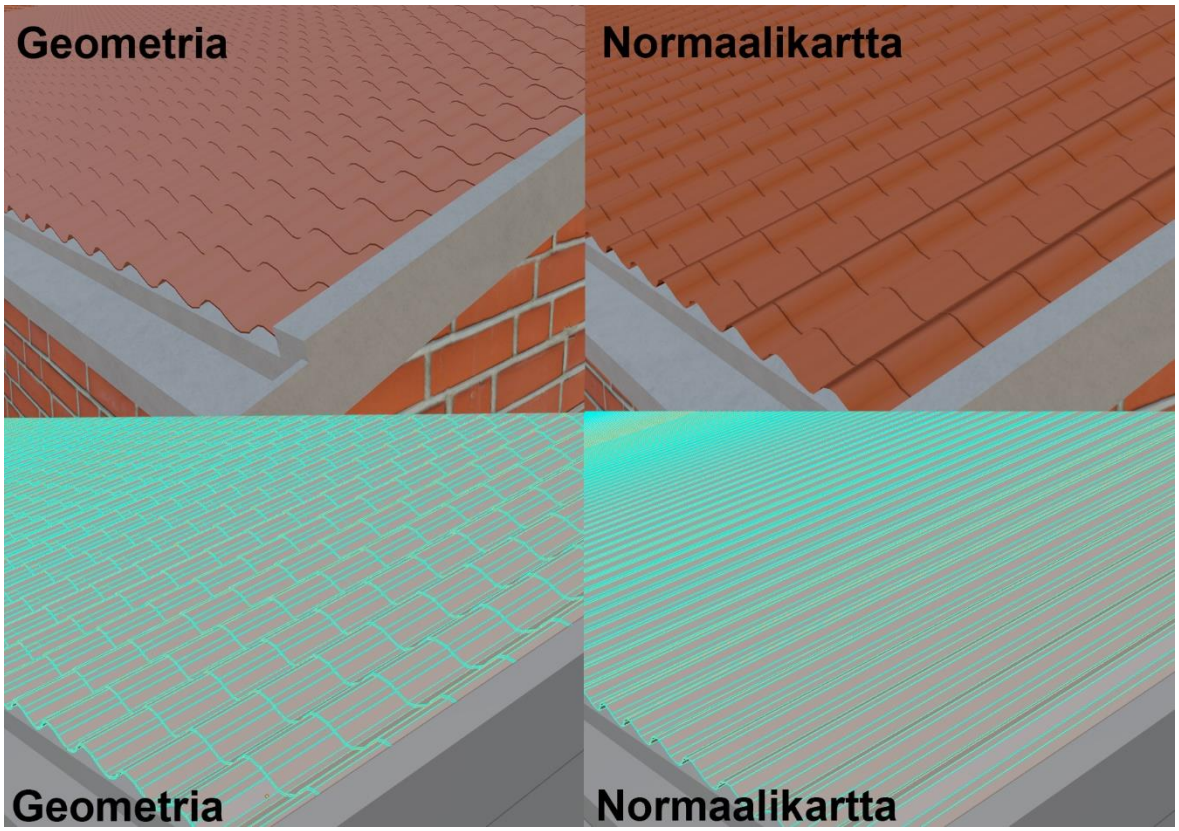
KUVA 26. Low polygon suunnitelma kirkon julkisivusta ja tornista.

Suunnitelman jälkeen mittojen määrittely on helppoa, koska monet rakennuksen elementit ovat samankokoisia. Ovien ja ikkunoiden ollessa samankokoisia on mallikuvan avulla

helppo määrittää seinien mittoja. Monet ikkunat ovat saman muotoisia, joten yksittäistä ikkunaa pystyy toistamaan läpi rakennuksen muuttamalla vain sen väriä ja kokoa. Tämä koskee myös rakennuksen kattoja ja tikkaita. Näitä elementtejä on myös helppo käyttää muissa malleissa.

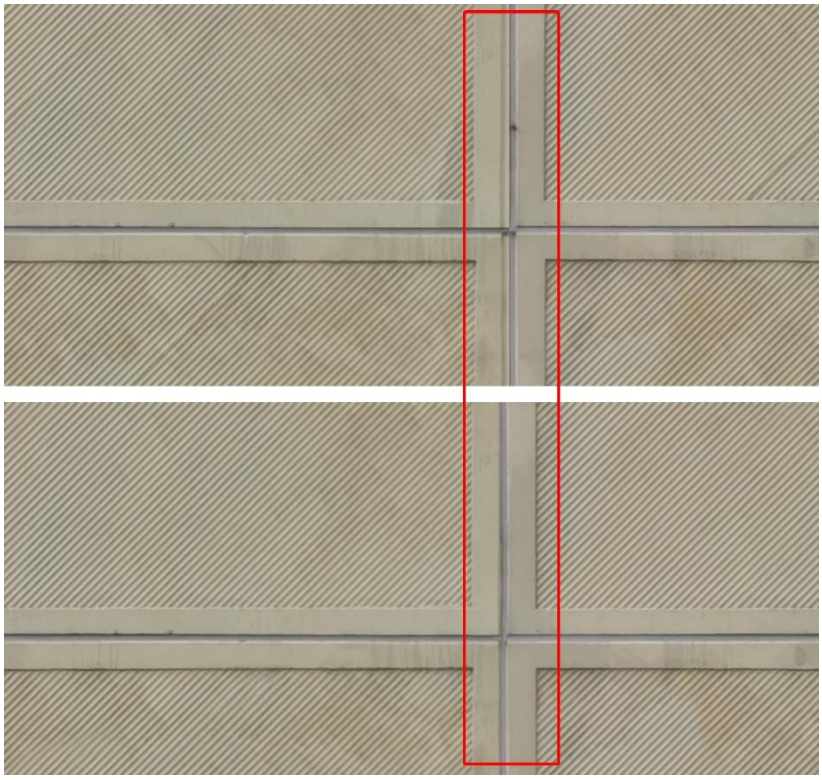
6.4 Teksturointi

Rakennuksissa on usein paljon samankaltaisia seiniä, joten näiden teksturointiin käytettiin toistuvia tekstuureja. Toistuvan tekstuurin on oltava hyvälaatuinen, jotta toistuvuus ei ole ilmiselvä pelaajalle. Kirkon tiiliseinää teksturoidessa ja UV-karttaa määrittäessä täytyi olla erityisen tarkka. Yksittäisen tiilen koko on oltava lähellä mallikuvaa, jotta skaala pysyy yhtenevänä. Rakennuksen kattojen yksityiskohdat luotiin normaalikartan avulla. Jos kattotiilien mallinnus tehdään geometrian avulla, yhteen kattoon tulee yli 75 tuhatta pintaa, kun taas normaalikarttaa käyttämällä pintoja on alle kaksituhatta. Kuvassa 27 näytetään näiden tekniikoiden erot. Vasemmanpuoleisessa kuvassa tiilen reunat mallinnettiin, kun puolestaan normaalikarttaa käytettäessä tekstuuri saa tiilet erottumaan toisistaan. Alemmista kuvista on poistettu värit, jotta geometria on helpommin tunnistettavissa.



KUVA 27. Geometrian ja normaalikartan ero kattoa mallinnettaessa

Jos valmiita tekstuureita ei ole käytettävissä, on tekstuuri luotava valmiista kuvasta. Rakennuksen yleisilmeen ja yksityiskohtien lisäksi, kannattaa mallikuvia kuvatessa kiinnittää huomiota seiniin mistä saa muokattua hyviä tekstuureja lopulliseen malliin. Kuvasta on mahdollista poistaa yksityiskohtia myöhemmin mutta hyvin otettu kuva nopeuttaa työskentelyä. Silmäänpistävien yksityiskohtien lisäksi tekstuurin reunojen on sulauduttava toisiinsa, jotta sitä voidaan toistaa huoletta mallissa. Kuvassa 28 on seinän tekstuuri ennen ja jälkeen editoinnin. Kuvaan on lisätty siirtymäefekti, joka näyttää kuinka alkuperäinen kuva ei tule toimimaan toistuvana tekstuurina ilman muokkausta. Korostetulla alueella huomaa kuinka käsitelty tekstuuri toistuu sulavasti. Muokatusta tekstuurista on myös poistettu yksityiskohtia.



KUVA 28. Muokkaamattoman ja muokatun tekstuurin ero

Värikartta on vain yksi tekstuuri, joka on luotava mallia varten. Suurin osa tekstuureista pohjautuvat väritekstuuriin, joten sen on oltava toimiva ennen muita tekstuureja. Normaalikartta on mahdollista luoda automaattisesti kuvankäsittelyohjelmassa. Värikartta on muutettava mustavalkoiseksi tätä varten. Mustavalkokuva auttaa ohjelmaa määrittämään, mitkä alueet tulkitaan painaumiksi, joten kontrastin lisääminen alueilla, joiden korkeuserot ovat isot on suotavaa. Tiiliseinän tiilien välit ovat hyvä esimerkki alueista missä kontrasti saa olla korkea. Kuvassa 29 oven väritekstuurista luotu normaalikartta tulkitsee kyltin korkeaksi,

vaikka kyseessä on tarra. Tällaisissa tilanteissa alkuperäistä kuvaa on muokattava ennen normaalikartan luomista.

KUVA 29. Oven normaalikartta on toimiva kylttiä lukuun ottamatta

6.5 Malli Unreal Enginessä

Ison tiedoston lisääminen Unreal Engine -projektiin voi olla hidasta, joten seurakuntakeskus jaettiin kolmeen eri osaan ennen exporttausta. Kirkon törmäysobjekti, sisätilat ja ulkotilat ovat erillään pienempiä, sekä helpompia muokata, jos mallista löytyy puutteita pelimoottorissa. Mallin ulkoseinät olivat silti isoin objekti, koska siihen yhdistettiin katot, ikkunat ja vastaavat ulkona näkyvät objektit. Törmäysobjekteissa yhdistyi sisätilat ja ulkotilat, koska mallin tiedostokoko on varsin pieni. Kaikkien kolmen FBX tiedoston origin tai lähtöpiste on sama, joten kaikki mallit asettuvat automaattisesti oikeille paikoilleen. Kuvassa 30 on aikainen versio kirkosta Unreal Engine -pelimoottorissa. Unreal Enginen renderöinti moottori saa mallin näyttämään erilaiselta verrattuna mallinnusohjelmaan.

KUVA 30. Seurakuntakeskus Unreal Engine 4 pelimoottorissa

Mallinnusohjelmassa on joskus vaikea hahmottaa, miltä tekstuurit näyttävät oikein valaistuilta. Seurakuntakeskuksen kohdalla tiiliseinän normaalikartassa oli huomattava virhe, joka oli vaikea huomata mallinnettaessa. Osa sisätilan huonekaluista olivat myös väärän kokoisia. Näistä syistä mallista tai sen osasta kannattaa luoda versio, jota on helppo tarkastella pelimoottorissa. Projektista riippuen malleja voi tarkastella tyhjässä projektissa, mikä on nopeaa, mutta ei anna mallista aina oikeaa kuvaa. Esimerkiksi valaistus voi olla erilainen lopullisessa peliympäristössä.

7 Yhteenveto

Opinnäytetyön tavoitteena oli tutkia, kuinka luoda toimivia 3D-malleja Unreal Engine pelimoottoriin. Jotta pelimoottoriin saatiin toimiva malli, täytyi mallin rakenteen olla toimiva ja tekstuurien oli lisättävä malliin yksityiskohtia. Lopullisen mallin oli oltava tiedostokooltaan mahdollisimman pieni ja tiedostoformaatin täytyi olla oikea.

Ennen mallinnusta malinnettavasta kohteesta oli oltava runsaasti hyviä mallikuvat, jotta mallin hahmottaminen on helppoa. Suunnittelu vaiheessa oli myös hyödyllistä käyttää kevyitä low polygon malleja skaalan määrityksessä. Kun suunnitelma oli hoidettu hyvin, olivat seuraavat työvaiheet helpompia.

Itse mallinnustekniikoiden lisäksi oli tiedettävä, miten mallin topologiasta tehdään toimiva. Hyvä topologia teki mallin myöhemmästä muokkauksesta helpompaa ja minimoi ongelmat teksturoidessa.

Teksturointi oli prosessi, jonka aikana mallin ympärille projisoidaan kuvatiedosto. Nämä kuvatiedostot muuttivat mallin väriä sekä lisäävät yksityiskohtia. Teksturoidessa oli ymmärrettävä, mitä eri tekstuurit tekevät ja kuinka pelimoottorit tulkitsevat tekstuureja. Ennen teksturointia mallille oli määritettävä UV-kartta. UV-kartta määrittää, miten tekstuuri asettuu mallin pinnalle. Jos mallin topologiaan ei keskitytty mallinnusvaiheessa, voi UV-kartoitus olla vaikeaa.

Jotta mallin voi viedä Unreal Engine -pelimoottoriin ilman ongelmia, mallista on luotava FBX-tiedosto. FBX on monipuolisin 3D-mallien tiedostoformaatti. Tässä export vaiheessa oli pidettävä mielessä kappaleen orientaatio ja skaala, jotta malli pysyi yhtenäisenä, kun se vietiin pelimoottoriin.

Kun malli oli tuotu Unreal Engineen, oli mallia vielä mahdollista muokata. Mallin materiaalia pystyi muokkaamaan tarkemmin pelimoottorissa ja siihen pystyi lisäämään lisää yksityiskohtia ylimääräisillä tekstuureilla. Unreal Enginessä oli myös tarkistettava mallin valokartta, joka määrittää kuinka valot ja varjot näyttäytyivät mallissa. Johtopäätöksenä Unreal Enginen materiaalieditori sekä hyvät ja monipuoliset tekstuurit tekivät malleista kevyitä ja toimivia.

Opinnäytetyötä pystyy jatkokehittämään monin tavoin. Opinnäytetyö ei keskittynyt Unreal Enginessä tehtäviin muutoksiin syvällisesti tai animoitaviin hahmoihin. Tehokas alusta mallien animoinnille tai Unreal Enginen eri ominaisuuksiin keskittyminen ovat mahdollisia tutkimusaiheita.

Lähteet

Abid P & Naghdi A. What is a 3D animation layout and why does it matter? Viitattu 22.11.2021. Saatavissa: https://dreamfarmstudios.com/blog/what-is-a-3d-animation-layout-and-why-does-it-matter/

Bhattacharya S. 2021. Love to play games? Look at these top 10 Programming Languages Used for game Development 2021. Viitattu 20.10.2021. Saatavissa: https://www.analyticsinsight.net/top-10-programming-languages-for-game-development-2021/

Blender. Retopology. Viitattu 1.11.2021. Saatavissa: https://docs.blender.org/manual/en/latest/modeling/meshes/retopology.html

Chang A. The Process of 3D Animation. Viitattu 17.11.2021. Saatavilla: https://www.media-freaks.com/the-process-of-3d-animation/

Danan. 2016. Box Modeling: The 3D Modeling Technique Viitattu 6.10.2021. Saatavissa: https://thilakanathanstudios.com/2016/10/box-modeling-the-3d-modeling-technique/

Danan. 2016. Why Do We Need Topology in 3D modeling. Viitattu 9.11.2021. Saatavissa: https://thilakanathanstudios.com/2016/09/why-do-we-need-topology-in-3d-modeling/

Denham T. What is LOD (Level of Detail) in 3D Modeling? Viitattu 1.11.2021. Saatavissa: https://conceptartempire.com/3d-lod-level-of-detail/

Denham T. What is Rendering? (For 3D & CG Work) Viitattu 8.10.2021. Saatavissa: https://conceptartempire.com/what-is-3d-rendering/

Denham T. What is UV Mapping & Unwrapping? Viitattu 5.11.2021. Saatavissa: https://conceptartempire.com/uv-mapping-unwrapping/

Dealessandri M. 2020. What is the best game engine: is Unreal Engine right for you? Viitattu. 19.11.2021. Saatavissa: https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unreal-engine-4-the-right-game-engine-for-you

Emperore K, Sherry D. Unreal Engine Physics Essentials Viitattu 3.11.2021 Saatavissa: https://subscription.packtpub.com/book/game-development/9781784394905/1/ch01lvl1sec18/the-2d-and-3d-coordinate-systems

Epic Games Inc. Generating Lightmap UVs Saatavissa: https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/Types/StaticMeshes/AutoGeneratedLightmaps/

Epic Games Inc. Instanced Materials. Viitattu 20.11.2021. Saatavissa: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/MaterialInstances/

Epic Games Inc. Simple versus Complex Collision. Viitattu 20.11.2021. Saatavissa: https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/SimpleVsComplex/

Erell O. 2020. UE4 – Quick fix for normal map encoding. Viitatu 15.10.2021. Saatavissa: https://odederell3d.blog/2020/07/06/ue4-quick-fix-for-normal-map-encoding/

Format. 2019. Our Top 19 3D Modeling Software Picks: Free and Paid. Viitattu 16.11.2021. Saatavissa: https://www.format.com/magazine/resources/design/3d-modeling-software

Full Scale. 2021. What is a game engine? Viitattu 20.10.2021. Saatavissa: https://fullscale.io/blog/what-is-game-engine/

Gadea W. Using ZBrush For Sculpting 3D Forms Viitattu 16.11.2021. Saatavissa: https://idearocketanimation.com/14252-zbrush-for-sculpting-3d-forms/?nab=1

Game Dev Academy. 2015. What is LOD? (Level of Detail & LOD Groups). viitattu 1.11.2021. Saatavissa: https://www.youtube.com/watch?v=mIkIMgEVnX0

Gonzalez O. 2021. In-Depth Analysis of the 3 Most Popular 3d Modeling Software. Viitattu Saatavissa https://www.goodfirms.co/blog/three-most-popular-3d-modeling-software

Google. 2021. Google Earth. Viitattu 13.11.2021. Saatavissa https://www.google.com/earth/

Google. 2021. Google Maps. Viitattu 13.11.2021. Saatavissa: https://www.google.fi/maps

Heginbotham C. What is 3D Digital Sculpting? Viitattu 8.10.2021. Saatavissa: https://conceptartempire.com/what-is-3d-sculpting/

Houston B. 2019. OBJ 3D file format: When should you use it? Viitattu 7.11.2021. Saatavissa: https://www.threekit.com/blog/when-should-you-use-fbx-3d-file-format?hsLang=en

Houston B. 2019. When should you use FBX 3D file format? Viitattu 7.11.2021. Saatavissa: https://www.threekit.com/blog/when-should-you-use-fbx-3d-file-format?hsLang=en

Huston S. 2020. Box Colliders Are Simply Magical. Viitattu 20.10.2021. Saatavissa: https://medium.com/@notaproblem/box-colliders-are-simply-magical-6a146fb3ba49

Jase A. 2020. Artists' best practices for mobile game development. Viitattu 18.11.2021. Saatavissa: https://blog.unity.com/technology/artists-best-practices-for-mobile-game-development

Johnson A. 2014. Diffuse Maps. Viitattu 14.10.2021. Saatavissa: https://docs.cryengine.com/display/SDKDOC2/Diffuse+Maps

KatsBits. Make Better Textures, The 'Power Of Two' Rule & Proper Image Dimensions. Viitattu 9.11.2021. Saatavilla: https://www.katsbits.com/tutorials/textures/make-better-textures-correct-size-and-power-of-two.php

Kutz S. 2019. Gaming Industry Explained: Concept Art for 3D Games. Viitattu 2.11.2021. Saatavilla: https://medium.com/imeshup/gaming-industry-explained-concept-art-for-3d-games-d88ad8492fb1

Markom3D. 2019. UV Lightmap are Overlapping - Resolve issue - UE4 Blueprints. Viitattu 21.11.2021. Saatavissa: https://www.youtube.com/watch?v=uCaaZF-Zo4c

Petty J. Textures vs Materials in 3D Graphics (A Complete Guide For Beginners). Viitattu 13.10.2021. Saatavissa: https://conceptartempire.com/3d-textures-vs-materials/

Polynook. How to Export Models from Blender to Unity. Viitattu 3.11.2021. Saatavissa: https://polynook.com/tutorial/how-to-export-models-from-blender-to-unity/

Pluralsight. 2014. Elliminate Texture Confusion: Bump, Normal and Displacement Maps. Viitattu 14.10.2020. Saatavissa: https://www.pluralsight.com/blog/film-games/bump-normal-and-displacement-maps

Rose D. 2019. Metallic Magic. Viitattu 18.11.2021. Saatavissa: https://medium.com/gametextures/metallic-magic-2dce9001fe15

Selin E. How to set up background reference images in Blender. Viitattu 10.10.2021. Saatavissa: https://artisticrender.com/how-to-set-up-background-reference-images-in-blender/

Slick. 2021. 3D Model Components—Vertices, Edges, Polygons & More. Viitattu 6.10.2021. Saatavissa: https://www.lifewire.com/3d-model-components-1952

The Open Augmented Reality Teaching Book. Unity (complete) / Unreal Engine. Viitattu 30.10.2021. Saatavissa: https://codereality.net/ar-for-eu-book/chapter/development/tools/unity/unityIntroduction/

Turbosquid. Topology. Viitattu 9.11.2021. Saatavissa: https://resources.turbosquid.com/checkmate/checkmate-product-presentation/topology/

Vanhat kartat. Viitattu 11.11.2021. Saatavissa: https://vanhatkartat.fi/#13.84/62.27051/25.80378

Vendelskis S. 2021. What is the difference between High Poly and Low Poly models in 3D modeling? Viitattu 18.10.2021. Saatavilla: https://www.arsenal.ai/blog/difference-between-high-poly-and-low-poly-models

World of Level Design. 2016. UE4: How to Create Your First Decal Material. Viitattu 20.11.2021. Saatavissa: https://www.worldofleveldesign.com/categories/ue4/ue4-decals01-your-first-decal.php