VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Otto Sillanpää

3D-grafiikka ja pelimoottorit

Liiketalous ja matkailu
2014

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

TIIVISTELMÄ

Tekijä	Otto Sillanpää
Opinnäytetyön nimi	3D-grafiikka ja pelimoottorit
Vuosi	2014
Kieli	suomi
Sivumäärä	32
Ohjaaja	Päivi Rajala

Tässä opinnäytetyössä tutkitaan miten 3D-mallit saadaan sellaiseen muotoon, että ne olisivat käytettävissä eri pelimoottoreissa.

Tutkimuksen tarkoituksena on selvittää, miten luodaan 3D-malleja pelimoottoreihin, sekä miten 3D-mallinnusohjelmat ja pelimoottorit eroavat toisistaan, kun käsitellään 3D-malleja. Tässä työssä pelimoottoreina toimivat Valven Source sekä Epic Gamesin Unreal Engine 3. 3D-mallinnusohjelmista käytössä olivat Autodeskin 3ds Max 2014 ja Blender Foundationin Blender 2.71. Työssä luodaan molemmilla 3D-mallinnusohjelmilla kolmiulotteinen malli, joka tuodaan kumpaankin työssä käytettävään pelimoottoriin. Teoreettinen osa kertoo kolmiuloitteisen tietokonegrafiikan historiasta ja rakenteesta, sekä pelimoottorien perustoiminnoista ja 3D-mallinnustavoista. Taustatietoina ovat koulussa suoritettu 3D-mallinnuskurssi sekä kokemus Source ja Unreal Engine 3 pelimoottoreista.

Avainsanat 3D-mallinnus, pelimoottori, renderöinti, polygoni

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Otto Sillanpää
Title	3D Graphics and Game Engines
Year	2014
Language	Finnish
Pages	32
Name of Supervisor	Päivi Rajala

This thesis examined how 3D models can be converted into a format that is supported by game engines.

The way how the 3D models are made into game engines and how 3D modeling software and different game engines differ from each other was researched. The game engines that were used in this work were Source Engine made by Valve and Unreal Engine 3 made by Epic Games. Autodesk 3ds Max 2014 and Blender Foundations Blender 2.71 were the 3D modeling software used in this thesis.

In this thesis, a 3D model was created with both 3D modeling software and the models were exported into both game engines. The theory explained the history and structure of 3D-graphics. Also, the theory contains subjects like the basic features of game engines and different ways of 3D modeling. Background information for the work, was a 3D modeling course completed at Vaasa University of Applied Sciences, and personal experience from Source and Unreal Engine 3 game engines.

Keywords 3D modeling, game engine, rendering, polygon

SISÄLLYS

1 JOHDANTO

3D–grafiikan käyttö on yleistynyt hyvin monella eri alueella. Näitä alueita ovat elokuvateollisuus, opetuskäyttö, lääketiede, arkkitehtuuri sekä videopelit. 3D-grafiikan käyttäminen videopeleissä lisääntyi huimasti 90 - luvun puolivälissä ja nykyään peliteollisuuden liikevaihto on suurempi kuin elokuvateollisuudessa (Puhakka 2008, 25). Yritykset valmistavat pelimoottoreita, joiden pohjalle muut yritykset voivat tehdä videopelejä. Pelimoottoreilla voidaan saada aikaan muutakin kuin viihdyttämistä varten tehtyjä pelejä, niillä voidaan myös tehdä simulaatioita. Unreal Enginen pohjalle tehtyjä simulaatioita käyttävät muun muassa FBI ja muut Yhdysvaltain organisaatiot (BBC, 2012). Valven Source Engine sekä Epicin Unreal Engine 3 ovat markkinoiden tunnetuimpia pelimoottoreita. Tässä työssä selvitetään, miten 3D-mallit luodaan kyseisiin pelimoottoreihin. Opinnäytteen teoriaosuudessa kerrotaan yleisesti 3D-grafiikasta, pelimoottoreista ja 3D-mallien tuottamisesta pelimoottoreihin. Pelimoottoreina on käytössä Valven Source engine sekä Epicin Unreal Engine 3 pelimoottori. 3D-mallinnusta varten käytössä olivat Autodesk 3ds Max 2014 sekä avoimen lähdekoodin Blender 2.71, jonka kehittämisestä vastaa Blender Foundation. Työssä saadaan myös katsaus siihen, miten molempien pelimoottorien ja 3D-mallinnusohjelmien käytäntötavat eroavat.

Työn tavoitteena on selvittää, kuinka luodaan 3D-malleja ja kuinka niitä tuodaan pelimoottoreihin. Mallien toimimista pelimoottorin sisällä tutkitaan. Työssä perehdytään 3D-mallinnusohjelmien eri käyttötapoihin. Työn aiheen valitsin oman kiinnostukseni pohjalta videopelejä ja 3D-mallinnusta kohtaan. Aikaisemmin olin luonut 3D-malleja, joita pystyi vain katselemaan 3D-mallinnusohjelman sisällä. Tämän työn avulla halusin oppia luomaan 3D-malleja niin, että ne toimisivat sisältönä videopelissä. Työ on suunnattu 3D-mallinnuksesta ja videopeleistä kiinnostuneille, sekä niille jotka haluavat luoda 3D-mallin toimimaan myös muualla kuin pelkässä 3D-mallinnusohjelmassa.

Teoriaosuuteen lähteitä on löytynyt kirjastosta sekä internetistä. Käytännön osuuden lähteet ovat löytyneet internetistä videoiden ja tekstioppaiden muodossa. Käytännön osan tarkoitus on saada lukija ymmärtämään, kuinka malleja käsitellään pelimoottoreita varten. Ymmärrettyään tämän työn perustoimintoja lukija on valmis ymmärtämään yksityiskohtaisempia 3D-malleihin kuuluvia asetuksia, jotka vaikuttavat 3D-mallin käyttäytymiseen pelimoottorissa.

2 3D-GRAFIIKKA

3D-grafiikka on tietokoneella luotua grafiikkaa jossa on myös kolmas ulottuvuus. 3D-grafiikassa käytetään X, Y ja Z akseleita. Tietokoneen näytölle luotu kuva on kuitenkin kaksiulotteista, eli 3D malleissa on vain kolmiulotteinen vaikutelma. (Slick 2014 a)

3D-mallin ruudulle piirtoon on kolme erilaista vaihetta. Ensimmäinen vaihe eli sovellusvaihe (application stage) luo kolmiulotteisesta mallista sellaisen version, että se on valmis piirtämistä varten. Toinen vaihe on geometria, joka vastaanottaa tietoa ensimmäisestä vaiheesta ja luo mallille geometrisiä muunnoksia sekä leikkauksia, jotta malli on valmis rasteroitavaksi. Rasterointivaihe on viimeinen. Sen tehtävänä on luoda 3D-mallista ruudulle sopiva kaksiuloitteinen kuva, joka koostuu pikseleistä. Tällöin nähdään ruudulla kaksiuloitteinen kuva, josta katsoja saa kuitenkin kolmiuloitteisen vaikutelman. (Puhakka 2008, 164-171)

3D-grafiikka on yhdistettynä useimmiten vain 3D-elementeissä ja visuaalisissa tehosteissa, joita nähdään videopeleissä ja elokuvissa. Kuitenkin 3D-grafiikka on mullistanut tieteellisen kehityksen sen havainnollisuuden vuoksi. 3D:n avulla pystytään tekemään luonnoksia ja pienoismalleja erilaisista projekteista kuten rakennuksista. Kolmiulotteisen havainnollistamisen avulla voidaan helpommin saada käsitys siitä, millainen lopputulos tulisi saavuttaa. Kolmiulotteisia luonnoksia voidaan myös animoida, tätä tekniikkaa voidaan käyttää esim. jonkin tapahtuman rekonstruktioon. (Puhakka 2008, 24)

Tietokoneella on esitetty grafiikkaa jo 1940-luvulta lähtien. Ensimmäiset tietokonegrafiikkaa esittävät ohjelmistot luotiin lähinnä kylmää sotaa varten Yhdysvalloissa. Elokuvissa käytettyä tietokonegrafiikkaa nähtiin ensi kertaan Alfred Hitchcockin Vertigo-elokuvan avauksessa vuonna 1958 (Puhakka 2008, 25). Ensimmäisen modernia 3D- grafiikkaa vastaavan animaation loi Ed Catmull vuonna 1972. Animaatio on nimeltään A Computer Animated Hand. Animaatiossa näytettiin hänen luomaansa mallia vasemmasta kädestään eri kuvakulmista. Vuonna

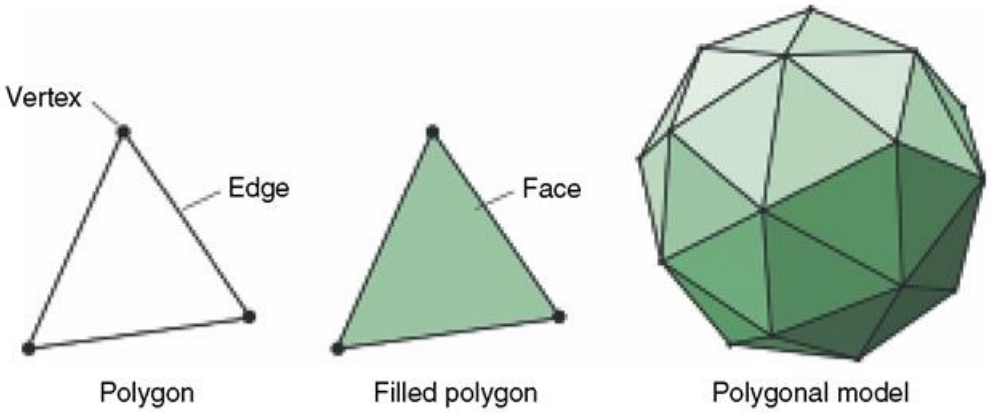
1976 julkaistussa elokuvassa Futureworld käytettiin otoksia kyseisestä animaatiosta. (Thornhill 2011)

Ed Catmull loi Z - puskuritekniikan periaatteen, tätä ratkaisua käytetään nykyisessä 3D-grafiikassa. Z - puskuriin tallentuu pikselin syvyyslukema. Näytönohjain valitsee lähimmän piirrettävän pikselin, ja luo näin kuvaan syvyysvaikutelman. (Puhakka 2008, 25)

2.1 3D-Mallin rakenne

3D-mallit koostuvat polygoneista. (Kuva 1) Polygonit taas koostuvat pienemmistä elementeistä kuten reunapisteistä (vertex). Reunapisteestä toiseen kulkeva suora on polygonin reuna (edge). Reunojen välillä oleva taso (face) on itse polygoni. (Slick 2014 b)

Kun polygonit ovat toistensa vieressä yhtenevästi, on kyseessä polygoniverkko eli mesh, joka antaa 3D–kappaleen muodon. Mitä korkeampi tiheys polygoniverkolla on, sitä tarkemmalta se näyttää. Korkealaatuisessa videopeliin kuuluvassa 3D-kappaleessa on harva polygoniverkko alueella, mikä ei vaadi kovin suurta tarkkuutta. Tälläinen alue voisi olla hahmon raaja. Polygoniverkko on tiheämpi hahmon naamassa tai kädessä mahdollisimman hyvän yksityiskohtaisuuden vuoksi. Liian isot polygonimäärät videopeleissä voi olla haitaksi, sillä suurempi polygonimäärä käyttää aina enemmän työaseman tehoja renderöintiä varten. (Slick 2014b)



Kuva 1. 3D-malli joka koostuu polygoneista (what-when-how 2014).

2.2 3D-grafiikan mallinnustapoja

Käyrämallinnus eli NURBS modeling, (Non-uniform rational basis spline) on tapa, missä luodaan malleja käyrien avulla. NURBS-malleja tehdessä 3D-mallintaja luo ensin käyriä, sen jälkeen käyrät yhdistetään toisiinsa. 3D-mallinnusohjelma laskee käyrien välimatkan ja luo pinnan niiden välille. NURBS-mallit ovat erittäin tarkkoja matemaattisesti, joten niitä käytetään hyvin usein teollisessa mallinnuksessa sekä autojen suunnittelussa. (Slick 2014 c)

Subdivision mallinnuksessa 3D-malli jaetaan aina yhä pienempiin osiin, jonka jälkeen saavutetaan haluttu lopputulos. Alkuun luotu laatikkomalli voi olla hyvinkin yksinkertaisen näköinen, mutta kun mallia ollaan käsitelty subdivision muuttujalla, niin mallin polygoonimäärä suurenee ja malli alkaa näyttämään paljon sileämmältä. (Slick 2014 c)

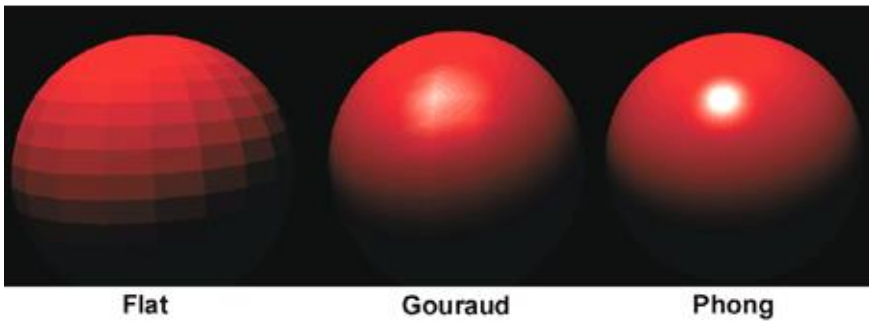
Veistäminen on melko uusi mallinnustekniikka. Veistämisessä voidaan manipuloida 3D-mallia niin kuin se olisi oikea veistos. Mallia voi hioa, työntää ja muovata eri kohdista. Tällä saadaan usein hyvin realistisen näköisiä malleja, joissa on erittäin korkea määrä polygoneja. (Slick 2014 c)

Polygonimallinnusta käytetään yleisimmiten videopeleissä, elokuvissa sekä animoinnissa. Polygonimallit koostuvat geometrisistä muodoista, malleissa on reunoja (edge), reunapisteitä (vertex) sekä tasoja (face). (Slick 2014 b)

2.3 3D-mallien pinnat

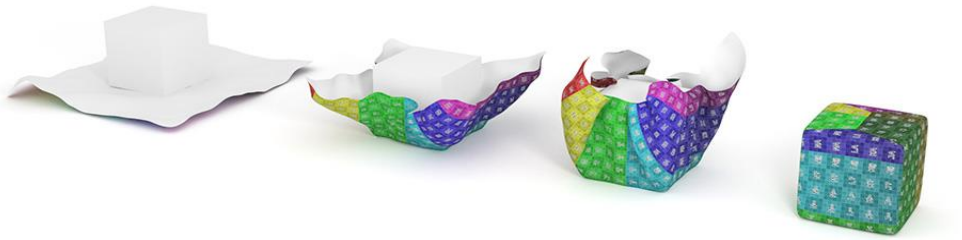
Kun 3D-malli on luotu, malli on useimmiten vain yksivärinen eikä siinä ole paljon eloa. Mallit saadaan elävemmiksi tekstuurien sekä varjostusten avulla. Tekstuurit ovat kaksiulotteisia kuvia, jotka liitetään 3D-mallin päälle. Kuvana voi olla esimerkiksi betonilattia. Kun tämä kuva liitetään vaikka suorakaiteen muotoiseen objektiin, mallin katsoja saa silloin heti käsitettyä, että kyseessä on betonilattia.

Jotta malli todella näyttäisi realistisemmalta, pelkkä teksturointi ei riitä. Varjostinalgoritmi (shader) kertoo mallinnusohjelmalle, kuinka paljon kiiltoa, heijastuvuutta sekä läpinäkyvyyttä tulisi olla mallissa (Kuva 2). Lisäksi varjostimet vaikuttavat siihen, miten malli reagoi valon kanssa. (Slick 2014 b)



Kuva 2. Kolme erilaista varjostustyyliä. (Intergraph Computer Systems)

UV Mapping (Kuva 3) on tapa, jolla saadaan liitettyä tekstuuri sopivan kokoiseksi 3D- mallin päälle. Ilman UV-kartoitusta, tekstuuri monistautuu vääriin paikkoihin tai näyttää muuten vääristyneeltä. (Griggs 2013)



Kuva 3. UV Mappingin toimintaperiaate. Kuutio esittää 3D-mallia, kun taas käärepaperi esittää tekstuuria (Sokolowski & Milewski 2014).

Bump Mapping (Kuva 4) on kartoitustyyli, jonka avulla malliin saadaan korkeuseroja muuttamatta itse mallin geometrista muotoa. Bump-Mapping muuttaa pin-

nan muotoa yksityiskohtaisemmaksi muuttamatta varsinaista 3d-mallin muotoa. Korkeuskartassa olevat kirkkaat sävyt ovat korkeampia sekä tummemmat sävyt ovat matalempia kohtia. (Birn 2006, 291)

Displacement Mapping (Kuva 4) taas on kartoitustyyli, jonka avulla saaadaan siirrettyä mallin pintaa eri muotoiseksi. Tämä on tavanomaista korkeuskartoitusta jyrkempi tekniikka. (Birn 2006, 290)

Kuva 4. Bump Mapping ja Displacement Mapping käytössä. (Nvidia 2014).

Normaalikarttojen avulla saadaan pinnoille korkeuseroja sekä valaistussefektejä. Normaalikarttoja käytetään usein videopeleissä, sillä normaalikarttojen käyttämisellä voidaan saada alhaisen polygonimäärän mallit näyttämään yksityiskohtaisemmilta. Pelimoottoreissa on yleensä rajoitukset sille, kuinka monta polygonia voidaan käsitellä tietyssä ajassa. Normaalikarttojen avulla voidaan nopeuttaa 3D-grafiikan piirtoa. (Slick 2014 d)

2.4 Piirtorajapinnat

Kaksi yleisintä piirtorajapintaa ovat OpenGL (Open Graphics Library) sekä Microsoftin DirectX. OpenGL sekä DirectX ovat ohjelmointirajapintoja jotka hyödyntävät työaseman grafiikkasuoritinta. Direct X on rajapinta myös muille laitteille kuten hiirille ja äänikorteille. Direct X:n varsinainen 3D-grafiikkarajapinta on nimeltään Direct 3D. (Sharma 2013)

Ohjelmointirajapinnat toimivat sovelluksen sekä laitteiston välillä. Rajapinnat myös helpottavat ohjelmoijien työtä. Ohjelmoijat voivat suoraan koodata sovelluksen tukemaan rajapintaa, ilman rajapintaa jouduttaisiin koodaamaan sovellus erikseen tiettyä laitteistoa varten. (Sharma 2013)

OpenGL on avoimen lähdekoodin grafiikkarajapinta. Se kehitettiin 1990-luvun alussa Silicon Graphicsin IRIS-piirtorajapinnan pohjalta. OpenGL on rajapinta, joka on tarkoitettu piirtämään perusprimitiivejä. OpenGL standardia käytetään laajasti opetuksessa, ohjelmistoissa, laitteissa ja peleissä. OpenGL ei ole sidoksissa vain yhteen käyttöjärjestelmään. (Puhakka 2008, 355)

Microsoftin Direct X käyttää suljettua lähdekoodia. Tämä tarkoittaa sitä, että Direct X on muokattavissa vain Microsoftin taholta. Direct X:n ensimmäinen versio julkaistiin vuonna 1995 ja viimeisin versio on Direct X 11. Direct X 11 tukee muun muassa tesselaatiota (Kuva 5), jonka avulla saadaan lisää tarkkuutta 3D-grafiikkaan.

Kuva 5. Direct X 10 (vasen) ja Direct X 11 (oikea) vertailussa (Megaleecher. 2013).

3 PELIMOOTTORIT

Pelimoottorit ovat ohjelmistokehyksiä, joita käytetään videopelin tekemisessä. Pelimoottoria voi sanoa siis pelin rungoksi, jonka päälle peli rakennetaan. Pelimoottoriin kuuluu aina jonkinlainen renderöintimoottori, joka määrittää millä tavalla grafiikkaa piirretään näytölle. Renderöintimoottori luo kaksi- tai kolmiulotteisen kuvan tiedostosta, jossa määritellään myös asioita kuten valaistus, tekstuuri ja näkökulma. Fysiikanmallinnusmoottorit saavat asiat ja esineet toimimaan peleissä fysiikan lakien mukaisesti, esimerkiksi kun pelaaja tönäisee kahvikuppia pöydällä, kahvikuppi tippuu maahan. Törmäyksen tunnistus varmistaa sen, että kun kaksi ruudulla renderöityä objektia törmää tai reagoi keskenään, tapahtuu pelissä jotakin. (WiseGeek 2014)

Muita yleisimpiä pelimoottoriin kuuluvia ominaisuuksia ovat äänien hallinta, tekoäly, sekä verkko-ominaisuudet.

Pelimoottoreista suurin osa on kaupallisia, mutta ilmaisia avoimen lähdekoodin pelimoottoreita myös löytyy.

Pelimoottoriin yleensä kuuluu renderöintimoottori, fysiikanmallinnus moottori, tekoäly, äänien käsittely, animaatiot ja verkko-ominaisuudet. Pelimoottorien keskuudessa lisensointi on yleistä. Uuden pelinkehittäjän ei tarvitse lisensoidulla pelimoottorilla tehdä peliään runkoa myöten aivan alusta asti. Tämä säästää erittäin paljon aikaa sekä kuluja. (WiseGeek 2014)

3.1 Source engine

Valve Corporation kehitti Source - pelimoottorin alun perin omaa Half-Life 2 - peliään varten, joka julkaistiin vuonna 2004. Half-Life 2:n sekä Counter-Strike Sourcen suuren menestyksen jälkeen pelimoottorista tuli suuri erittäin suosittu. Pelimoottori on vielä nykyään laajassa käytössä. Valve on julkaissut moottorilleen vuosien aikana useita päivityksiä, jotta se täyttäisi vielä kuluttajien vaatimukset pelien grafiikasta ja toiminnasta. Source - pelimoottori yllätti kuluttajat monipuolisella fysiikanmallinnuksellaan. (Valve Corporation 2014)

Sourcessa käytettävän fysiikkamoottorin on kehittänyt irlantilainen Havok. 3D-kappaleet alkoivat olla entistä elävämpiä virtuaalisissa ympäristöissä. Esineisiin kytkettävät fysiikan lait ovat vaikuttavat. Esineisiin vaikuttavat voima, kitka ja massa. Esineitä pystyy liikuttamaan ja tämä antaa uuden mahdollisuuden erilaisiin puzzle kohtauksiin. 3D-malleja pystyy myös tuhoamaan yksityiskohtaisesti Source pelimoottorissa. Karttojen tekstuureissa Source - pelimoottori pystyy näyttämään tekstuureita, joissa on mm. kohoumakartoitusta sekä valokartoitusta. (Valve Corporation 2014)

Pelihahmojen animointi monipuolisesti on mahdollista. Source - pelimoottori on myös tunnettu tarkasta kasvojen animoinnista. Valve on myös julkaissut Source Filmmaker - ohjelmiston, jolla voidaan tehdä animaatioelokuvia hyödyntäen Source pelimoottoria (Kuva 6). Source pelimoottorin työkalut ovat ilmaisia ja ne voidaan ladata Valven Steam - palvelusta.

Kuva 6. Source Filmmaker (Valve Corporation 2012).

3.2 Unreal engine

Unreal Engine on yksi käytetyimmistä pelimoottoreista. Pelimoottorin on kehittänyt Epic Games ja ensimmäinen versio julkaistiin vuonna 1998. Unreal Engine 4 on uusin versio pelimoottorista. Uusimman version työkalut saadaan käyttöön 19 euron hintaan. Tässä työssä on käytetty Unreal Engine 3 - versiota, jolla on toteutettu reilusti yli sata peliä. Unreal Engine 3 tukee muun muassa reaaliaikaisia varjostimia, monisäikeistä renderöintiä ja pikselikohtaista valaistusta. (Epic Games 2014)

Myös fysiikanmallinnus kuuluu pelimoottoriin. Tuotuihin 3D-kappaleihin voidaan määrittää asetuksia kuten massa, joka vaikuttaa kappaleen reagoimiseen pelaajan sekä ympäristön kanssa.

Unreal Enginen vahvuus on sen editori, jossa on hyvin monta ominaisuutta samassa. Pelimoottorin sisältöä muokataan Unreal Development kitillä. Samalla editorilla voidaan luoda karttoja, muokata 3D-kappaleiden toimintaa sekä vaikuttaa pelin perustoimintoihin. (Epic Games 2014)

4 3DS MAX JA MALLIN LUONTI

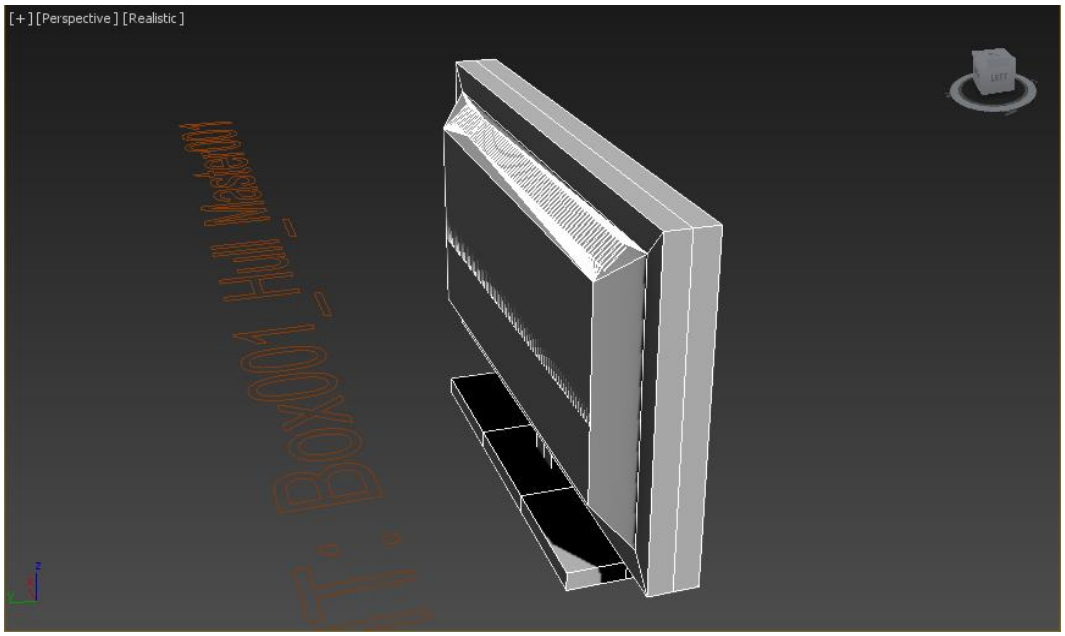
Käytin mallinnusten tekemiseen 3ds Maxin 2014 versiota. Ohjelmisto on erittäin laajassa käytössä, ja sopeutuu hyvin ammattitason mallinnuksiin mm. pelejä, elokuvia ja arkkitehtuuria varten. Ohjelmistosta tulee joka vuosi uusi versio. Uusina ominaisuuksina 3ds Maxiin on tullut 3D-stereokamera, Python - kielen komentosarjat, ShaderFX varjostintekniikka sekä pistepilvien tukeminen.

Aloitin mallin luonnin asentamalla ensin 3ds Maxin 2014 version. Sain opiskelijaversion käyttööni ilmaiseksi Autodeskin omilta sivuilta. Ensiksi ajattelin mallintaa television, jota voisin tutkia kahdessa eri pelimoottorissa. Käytin apunani koulussa käydyn 3D-kurssin materiaalia ja osaamista, jota olin kurssilta saanut.

Käytin mallina omaa televisioruutuani, joka helpotti muotojen keksimisessä. Televisioruudun aloitin aluksi Box - primitiivillä, johon lisäsin vielä yhden Height segmentin. Lisäsin laatikkoon Edit Poly - muuntajan, jotta saan muutettua laatikon muotoa yksityiskohtaisemmin. Tämän jälkeen jatkoin ruutuosan tekemisellä, Polygon alavalikon osasta valitsin Inset - toiminnon, jotta pystyn tekemään pienemmän polygonin laatikon sisälle. Ruudusta puuttui vielä syvennys, tämän sain tehtyä Bevel - toiminnolla. Nyt minulla oli valmiina jonkinlainen ruutu, mutta vielä malliin tarvittiin lisää elementtejä.

Television takaosan (Kuva 7) mallinnin samalla periaatteella kuin ruutuosan, mutta siirsin Bevel - toiminnolla polygonia toiseen suuntaan, jotta saisin sen ulospäin. Nyt sain televisioon takapaneelin. Takapaneeliin halusin hieman yksityiskohtia, kuten ilmastointiaukkoja. Tein yhden erittäin ohuen Box - primitiivin ja laitoin sen takapaneelin yläosaan. Seuraavaksi käytin Array-työkalua, jolla saan monistettua monta 3D- kappaletta helposti riviin. Kun olin mallintanut tarpeeksi monta laatikkoa vierekkäin, käytin Boolean toimintoa, jolloin laatikot leikkasivat takapaneeliin reiän. Kun television yläosa oli mallinnettuna, siirryin mallintamaan television jalkaa. Television jalan mallinnin kahdesta eri Box - primitiivistä. Jalan yläosaan lisäsin Edit Poly - muuntajan, jonka Vertex-pisteiden avulla muunsin

hieman yläosan muotoa. Jalan alaosa on pelkkä Box primitiivi ilman erikoisia muunnoksia. Jotta malli vaikuttaisi realistisemmalta, tuli malliin lisätä vielä tekstiä sekä painikkeita joilla televisiota voisi kuvittellisesti käyttää. Tekstin toteutin 3ds Maxin Text työkalulla joka sijaitsee Shapes -> Splines valikossa. Tekstin yhdistäminen television pinnalle onnistui Geometry -> Compound Objects valikon ShapeMerge toiminnolla. ShapeMerge työkalulla 3D-kappaleeseen saadaan halutun tekstin muotoiset polygonit. Näihin polygoneihin voidaan asettaa oma materiaalinsa, jolloin teksti saa oman värityksen.



Kuva 7. Televisiomalli takaa kuvattuna 3ds Maxin editointiruudussa.

4.1 3D-kappaleen teksturointi

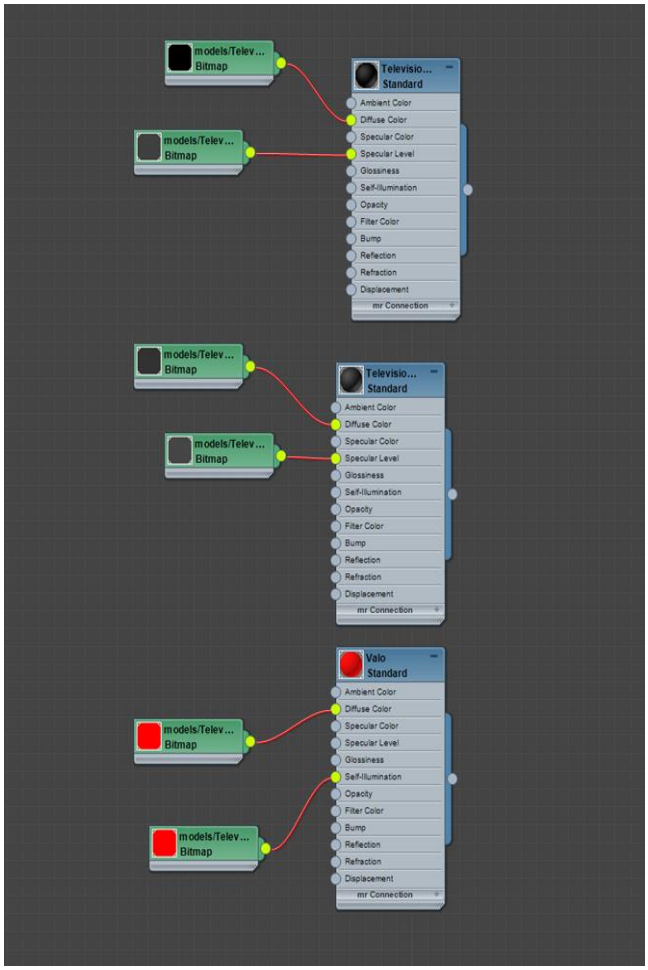
Useimmat modernit televisiot ovat mustia, joten päätin tehdä television runkoa varten mustan materiaalin (Kuva 8). Materiaali vaati myös kiiltävää ominaisuutta, jottei siitä tulisi liian pelkistetty. Ruudun väriksi sopii jokin vaaleampi väri, kuten

harmaa. Ruutuun voidaan liittää myös kiilto-ominaisuus. Source pelimoottoria varten tulee materiaalien pohjalla käyttää bittikarttoja, jotka ovat .tga - muodossa. Jos näin ei tehdä, Source engine löytää materiaalista vain diffuse - tekstuurin, jolloin kappale näyttää hyvin pelkistetyltä.

Kun bittikarttoja käytetään, voidaan asettaa materiaaliin muitakin kuin pelkkä diffuse ominaisuus. Diffuse mapin lisäksi voidaan laittaa materiaaliin myös esimerkiksi bump map ja specular map. Tämän kaltaisia materiaaleja voidaan luoda 3ds Maxin Material Editorissa (Kuva 9). Editorin vasemmasta palkista tulee ensiksi valita materiaalin tyyppi. Tässä työssä käytössä oli standard materiaalityyppi.

Bittikartat voidaan luoda kuvankäsittelyohjelmalla. Itse käytin bittikarttojen luomiseen GIMP - ohjelmaa, joka on ilmainen. Loin mustan bittikartan, jonka aion liittää materiaalin diffuse - ominaisuuteen. Bittikartan tulee olla kahdeksanbittisessä muodossa, muuten 3ds Max ei pysty avaamaan sitä. Jotta materiaali kiiltäisi Source - pelimoottorissa, tarvitaan vielä bittikartta, joka liitetään materiaalin specular ominaisuuteen. Specular - bittikartta on mustavalkoinen kuva, jossa valkoinen väri tarkoittaa valoa heijastavaa pintaa ja musta taas heijastamatonta pintaa. Jotta saadaan musta runko kiiltämään sopivasti Source - pelimoottorissa, käytin bittikarttana vain hieman vaaleampaa versiota diffuse kartasta. Ruutua varten käytin harmaata diffuse karttaa ja hieman tummempaa specular - karttaa, jotta ruutu ei heijastaisi yhtä paljon kuin runko.

Kuva 8. Renderöity malli 3ds Maxissa.



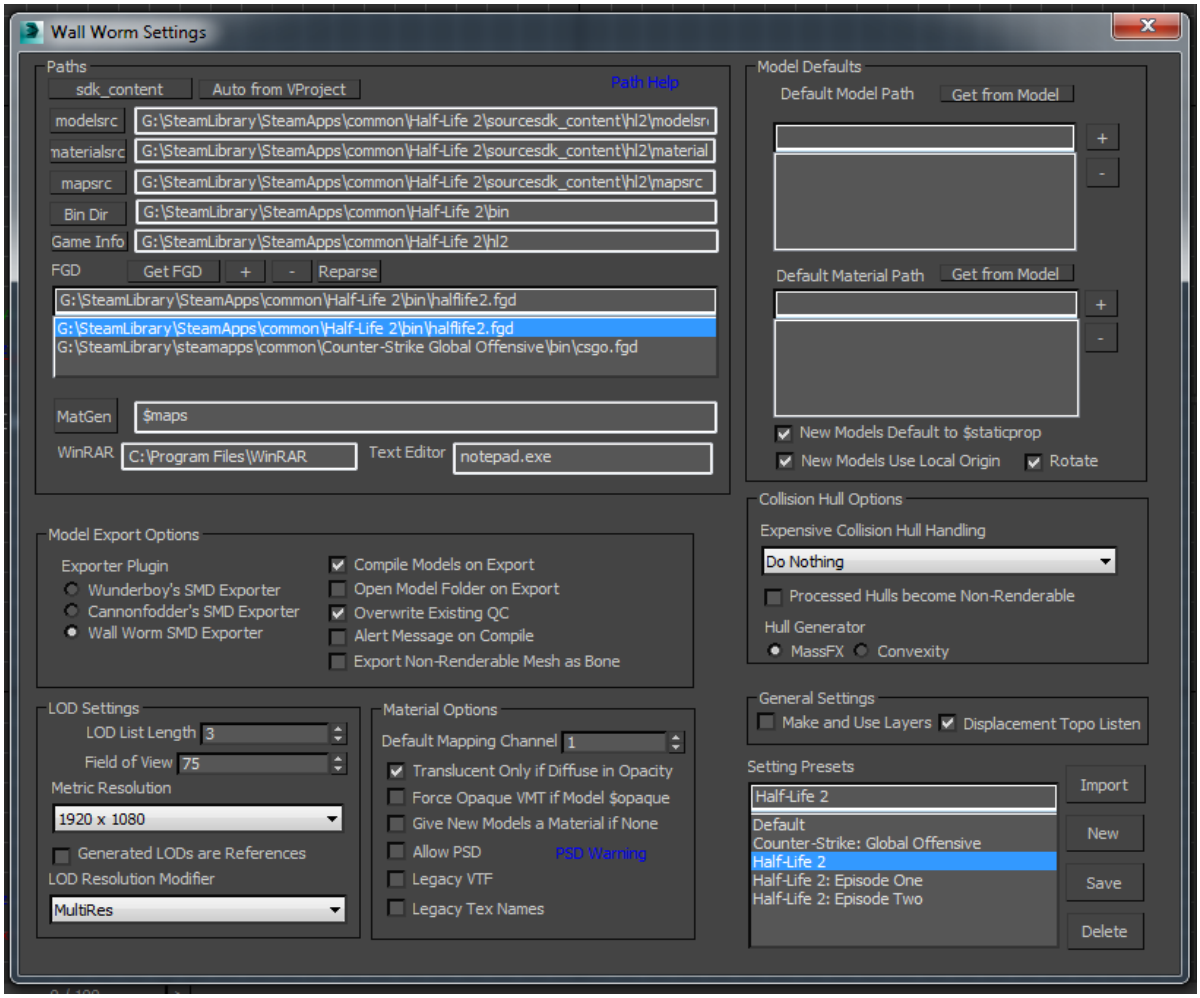
Kuva 9. Mallissa käytetyt materiaalit 3ds Maxin Slate material editorissa.

Materiaalit tulee vielä nimetä. Nimipalkki tulee näkyviin oikealle kun kaksoisklikataan haluttua materiaalia tai bittikarttaa. Kun materiaalit tuodaan Source - pelimoottoriin Wallworm skriptillä, materiaalista tulee .vmt - tiedosto. Bittikartoista taas muodostuu .vtf - tiedostoja. Bittikarttojen nimipalkin kohdalle laitetaan nimen sijasta polku, bittikarttojen nimet ovat jo tiedostoissa. Tässä työssä television diffuse kartan nimenä on DiffuseTV.tga ja se tulee tallentumaan Half-Life 2 - pelin hakemistoon materials/models/televisio/DiffuseTV.vtf (Kuva 10).

Kuva 10. Bittikartan tiedot 3ds Maxin Slate Material Editorissa.

4.2 Mallin vienti Source engineen

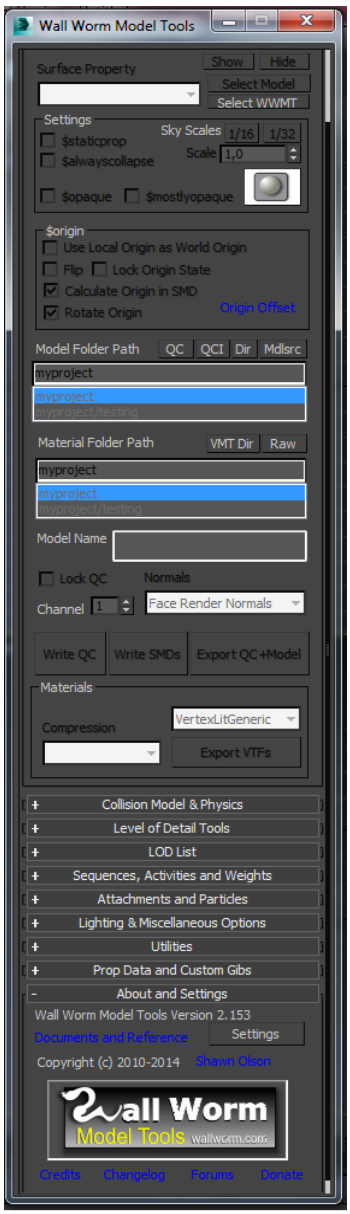
Parhaiten valmiin mallin tuominen 3ds Maxista Source pelimoottoriin onnistuu käyttämällä WallWorm Model tools skriptiä. Skriptin löytää sivulta dev.wallworm.com. Skriptin asennuksessa tulee siirtää .zip - tiedoston sisällä oleva WallWorm.com kansio 3ds Maxin omaan scripts hakemistoon. Scripts kansion siirtämisen jälkeen asennetaan skripti vielä 3ds Maxin kautta. Tämä onnistuu painamalla MAXScript - valikkoa ja etsimällä sieltä Run Script. Run Scriptin kautta etsitään Wallworm.com kansiosta install.ms - tiedosto, jonka avaamalla skripti asentuu käytettäväksi 3ds Maxiin. Asennuksen jälkeen pitää vielä konfiguroida Wallworm skriptiä niin, että 3D - kappaleet tallentuvat oikeisiin hakemistoihin. Asetukset riippuvat siitä, mihin Source pelimoottoria käyttävään peliin aikoo tuoda mallinsa. Tässä työssä käytin Half-Life 2 pelin asetuksia. Tietyn pelin asetukset saadaan nopeasti tuomalla pelin gameconfig.txt - tiedoston. Kyseinen tiedosto sijaitsee Source - pelimoottoria käyttävän pelin bin - kansiossa. Tiedosto tuodaan skriptiin painamalla Import - painiketta Setting Presets - kohdan vierestä (Kuva 11).



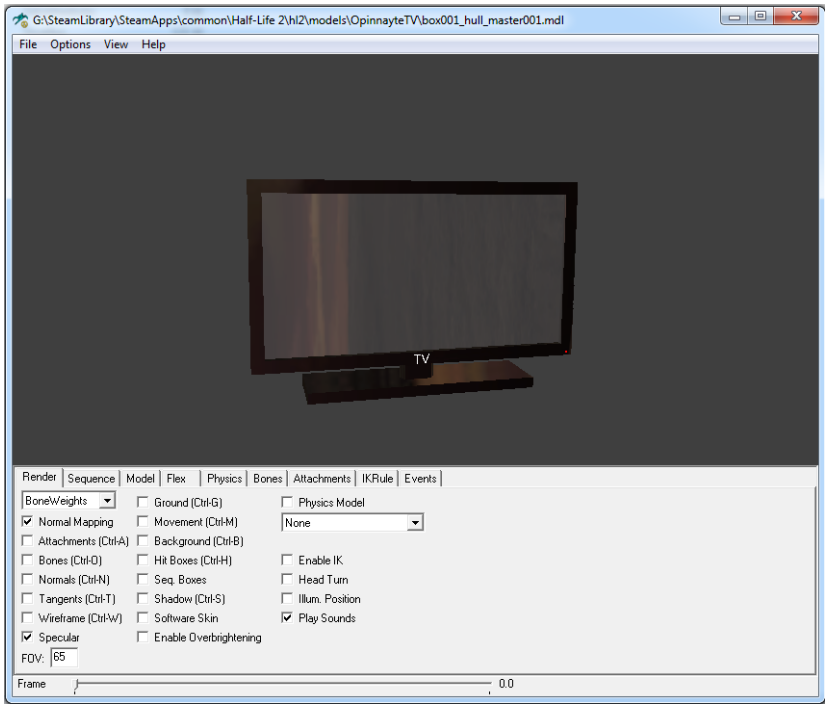
Kuva 11. WallWorm skriptin asetukset 3ds Maxissa.

Kun malli halutaan tuoda Source pelimoottoriin, avataan Wall Worm Model tools ikkuna valikosta WallWorm -> WallWorm Model Tools -> WallWorm Model Tools. Tämän jälkeen valitaan malli, joka halutaan tuoda painamalla Pick Model – painiketta, joka on aloitusvalikon yläosassa (Kuva 12). Model Folder path on polku, johon malli tuodaan. Material Folder path tarkoittaa polkua, johon mallin materiaalit menevät. Ennen mallin viemistä olisi hyvä että 3D-malliin saadaan laitettua collision model. Ilman collision modelia kaikki asiat menevät muuten pelissä 3D-mallin lävitse. Collision modelin saa nopeasti painamalla Auto hull - painikkeen päälle. Model name - kohtaan laitetaan mallin nimi. Export QC + Model painike käynnistää mallin tuomisprosessin.

Export VTF's painikkeella saadaan tuotua mallin materiaalit. Kun malli on viety, voidaan sitä tarkastella model viewerin kautta (Kuva 13). Model viewer löytyy pelin SDK – työkaluista, jotka voidaan käynnistää Steam - palvelun kautta tai pelin bin kansiosta.



Kuva 12. WallWorm skriptin aloitusvalikko.



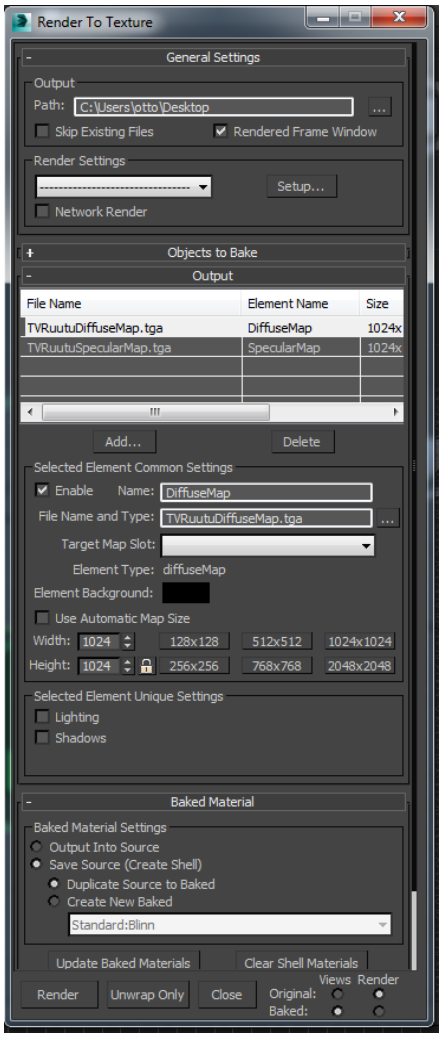
Kuva 13. 3ds Maxissa luotu malli vietynä Source pelimoottoriin

4.3 Mallin vienti Unreal Engineen

Mallin vieminen 3ds Maxista Unreal Engineen onnistuu ilman erikseen ladattavia skriptejä. 3ds Maxin malleja voidaan viedä .ASE formaatissa, mitä Unreal Engine tukee. Mallin saa tuotua käyttämällä Export toimintoa ja valitsemalla .ase formaatin. 3D-kappaleeseen tulee vielä lisätä Unwrap UVW - muokkain jolla tehdään UVW - kartoitus mallin pinnoille. Unwrap UVW - muokkaimesta valitaan alataso Polygon ja sen ollessa valittuna painetaan Open UV Editor - painiketta. UV Editorissa valitaan mapping - valikosta flatten mapping. Tämän jälkeen UV Editori on tehnyt UV – kartoituksen, jonka avulla saadaan mallin tekstuurit oikeaan kohtaan.

Unreal Development Kitissä käytettävä tekstuurikartta saadaan tuotua 3ds Maxista menemällä valikkoon Rendering -> Render to texture. Tässä valikossa tulee lisätä karttatyypit joita halutaan viedä painamalla Add - painiketta (Kuva 14). Tässä työssä vietiin Diffuse - kartta, sekä Specular - kartta. Tekstuureille tulee antaa ni-

met, polku jonne tekstuurit tallennetaan, sekä tekstuurien koko. Tämän työn tekstuureissa kokona oli käytössä 1024x1024 pikseliä. Lopuksi painetaan Render - painiketta, jotta saadaan tekstuurit tallennettua.

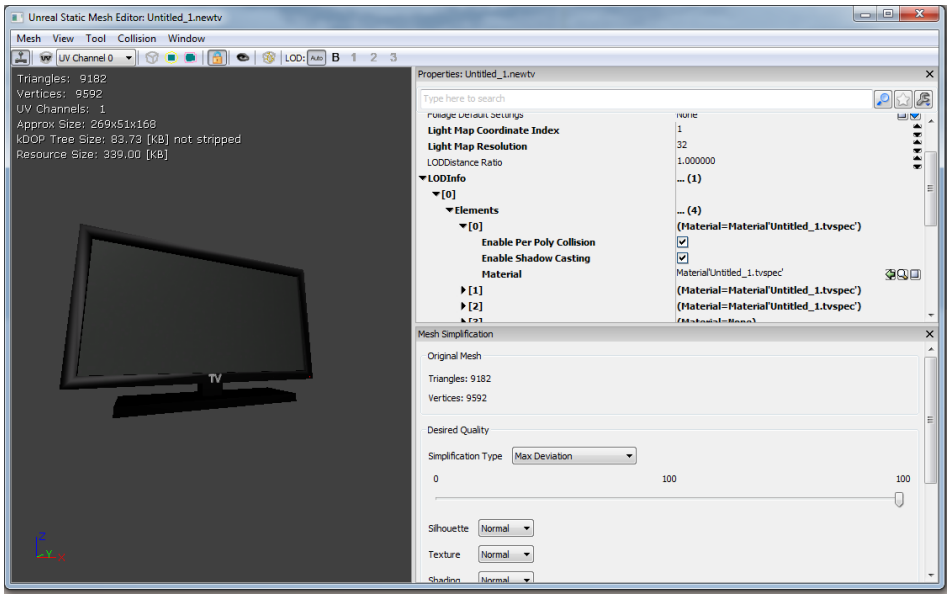


Kuva 14. 3ds Maxin Render To Texture ikkuna.

Mallin ja tekstuurikartan tallentamisen jälkeen tuodaan tiedostot Unreal Development kitiin. Content browserin vasemmassa alakulmassa on import - painike, jonka avulla tuodaan molemmat tiedostot. Jos Content browseria ei näy, saadaan se takaisin käyttöön valikosta View > Browser Windows > Content Browser. Juuri tuotua UV-karttaa ei voida vielä liittää malliin. Ensiksi täytyy luoda materiaali sivuklikkaamalla tyhjää aluetta Content Browserissa ja valitsemalla New material.

Name kohtaan tulee laittaa nimi, selkeyden vuoksi materiaalin nimi voi olla muuten sama kuin mallin, mutta perään laitetaan _mat pääte. Painamalla Ok - painiketta avautuu Unreal Material editori, joka ulkoasultaan muistuttaa 3ds Maxin Slate Material editoria. Tähän material editoriin voidaan nyt liittää tuotu UV kartta raahaamalla se ikkunaan. Kun tekstuuri näkyy editorissa, raahataan mustasta neliöstä viiva materiaalin Diffuse - kohtaan. Materiaali voidaan nyt tallentaa ja vielä tulee asettaa se 3D-kappaleeseen kaksoisklikkaamalla sitä Content browserissa. Klikkaamisen jälkeen avautuu Static mesh editor (Kuva 15), jossa voidaan muuntaa 3D-kappaleen asetuksia.

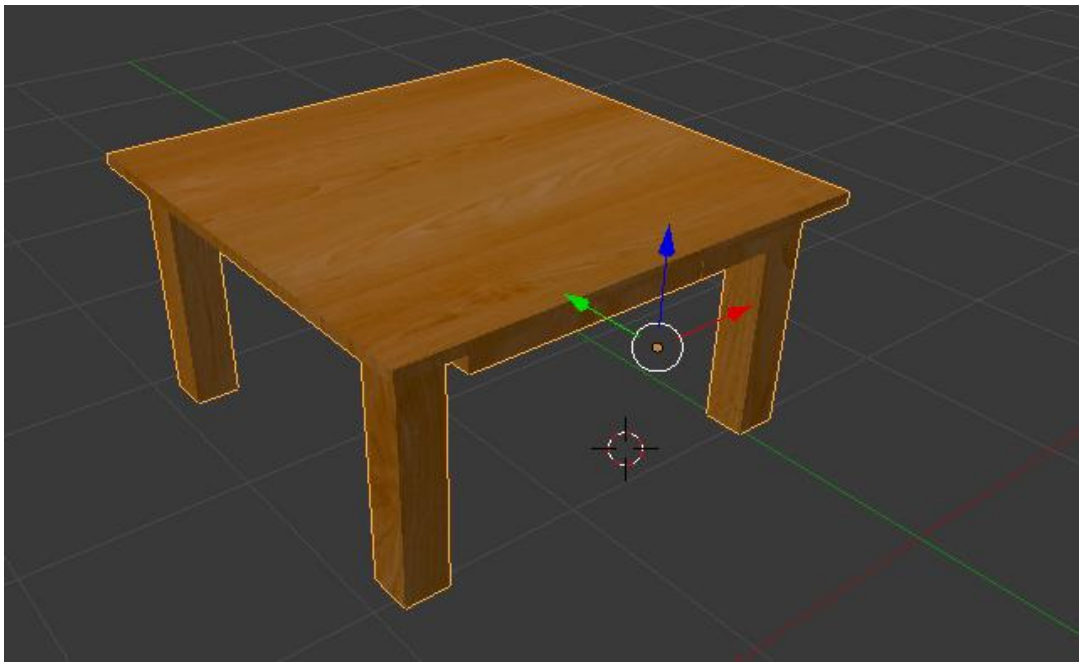
Materiaaliosuus löytyy LODInfo kohdan alapuolelta nimellä Material. Materiaalin saa nopeasti asetettua klikkaamalla ensin materiaalia content browserissa, ja sitten painamalla vihreätä nuolta, jossa lukee Use selected object in content browser. Tämän jälkeen 3D- kappaleessa on pinta.



Kuva 15. 3ds Maxista viety malli Unreal Development kitissä.

5 BLENDER JA MALLIN LUONTI

Blender on ilmainen avoimen lähdekoodin ohjelmisto 3D-mallien luomiseen. Työssä käytin Blenderin 2.71 versiota. Blenderillä loin puisen pöydän (Kuva 16). Aloitin pöydän levyosasta, josta vähitellen lisäilin Extrude - toiminnolla jalat sekä lipaston. Sain puisen materiaalin asettumaan hyvin käyttämällä Unwrap - komentoa, joka löytyi valikosta Uvs -> Unwrap.



Kuva 16. Blenderillä luotu pöytä 3D-malli.

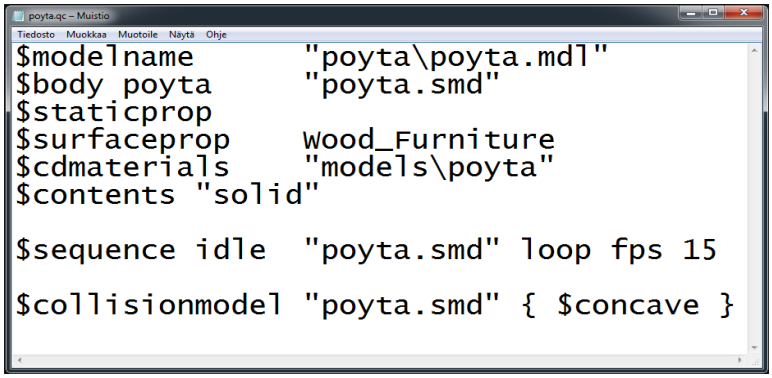
5.1 Blender ja mallin vieminen Source engineen

Blenderissä käytin mallin tuomista varten Blender Source Tools - nimistä työkalua. Työkalun asennus tapahtuu nopeasti. Blenderissä tulee valita File - valikosta User Preferences. Tämän jälkeen siirrytään addons - välilehdelle ja painetaan Install From File sekä valitaan ladattu .zip - tiedosto. Kun .zip - tiedosto on valittu, etsitään vielä Blender Source Tools - valikosta ja laitetaan ruksi valintalaatikkoon sekä odotetaan pieni hetki, että valinta on tapahtunut. Viimeinen varsinainen

asennustapahtuma on Save User Settings - painikkeen painaminen. Nyt Blender Source Tools on asennettuna Blenderiin.

Blender Source toolsin asetukset löytyvät scene välilehdeltä. Ainoa asetus jota skripti vaatii on polku johon mallit viedään. Vieminen Source - moottorin .smd - tiedostoksi onnistuu kulkemalla valikosta File > Export > Source Engine .smd

Malli ei ole kuitenkaan vielä täysin valmis toimimiseen Source pelimoottorin ympäristössä. Malli on nyt .smd - tiedostona, joka sisältää mallin tiedot ASCII - tekstinä. Malli tulee vielä kasata lopulliseen muotoonsa, .mdl - muotoon. Tätä varten tulee luoda .qc tiedosto (Kuva 17), joka sisältää erilaisia tietoja kuten sijainti, mihin lopullinen kasattu malli luodaan, mistä .smd - tiedostosta mallin muoto otetaan, animaatiot sekä käyttäytymismallit pelaajan vaikuttaessa malliin. Qc - tiedoston komennot alkavat $ - merkillä ja komentoja on jopa 100. Yksinkertaisessa mallissa qc-komentoja on noin 7 tai 8 kappaletta. Qc - tiedosto voidaan luoda Windowsin tekstieditorilla.



Kuva 17. Työssä käytetty .qc tiedosto.

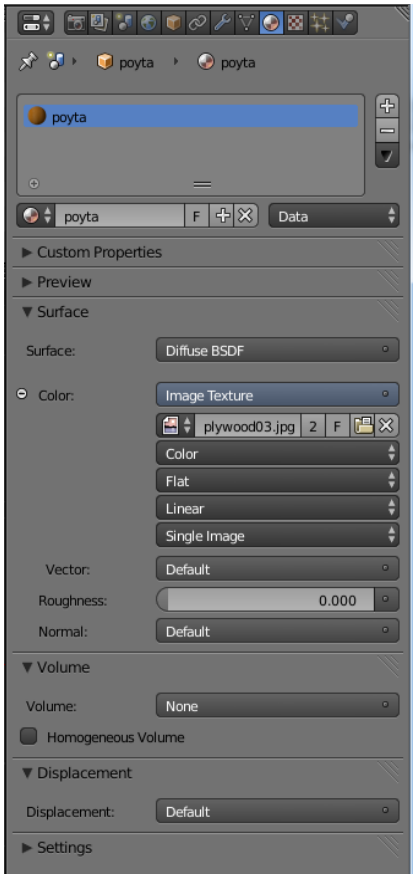
Modelname - komento määrittää mallin nimen ja sijainnin. Esimerkissä siis malli tulee tallentumaan models\poyta\ - polkuun nimellä poyta.mdl. Mallit sijaitsevat aina pelin models - kansiossa. Body komento hakee .smd – tiedoston, josta saadaan mallin muoto. Static prop komento kertoo sen, että mallissa ei ole liikkuvia

osia. Surfaceprop määrittää mallin tyypin ja käyttäytymisen. Eli onko kyseessä metallinen vai puinen objekti.

Cdmaterials etsii mallissa käytettävän materiaalin. Tässä tapauksessa kompiloin-tiohjelma etsii oikean tekstuurin materials\models\poyta -- hakemistosta. Tekstuu-rit ja materiaalit sijaitsevat aina materials - kansiossa. Contents - komennolla voi-daan määrittää, onko objekti kiinteä.

Qc tiedosto on hyvä sijoittaa samaan sijaintiin, mihin tuleva .mdl - tiedosto koo-taan. Tässä tapauksessa .qc tiedosto on laitettuna models/poyta - polkuun.

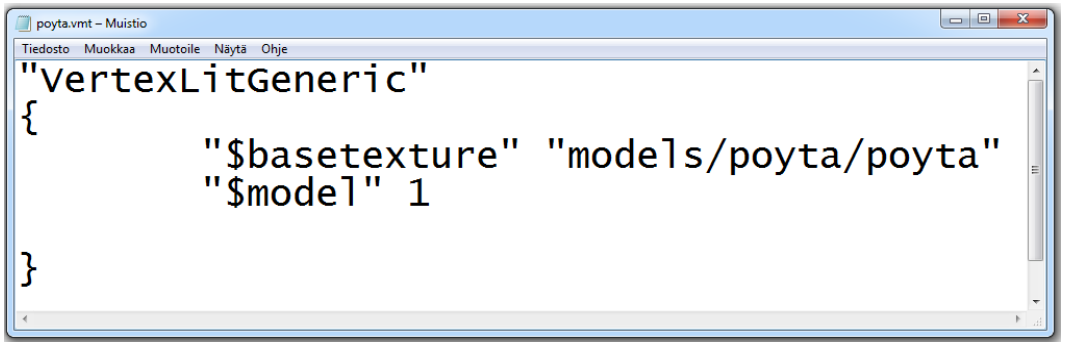
Source pelimoottori käyttää tekstuureissaan .vtf - tiedostomuotoa (Valve texture format). Kappaleen tekstuuri tulee muuntaa VTFEdit - ohjelmalla. VTFEdit ohjelmassa tekstuuri muutetaan tuomalla se File > Import - valikon kautta. Tämän jälkeen aukeaa ikkuna johon voidaan klikata OK. Kuva saadaan .vtf - muotoon painamalla Save as. Tekstuuritiedosto tulisi tallentaa pelin materials - kansioon. On parasta pitää tiedostonimi sekä kansioiden nimet samana, jottei tule sekaannuksia. Toinen huomioitava asia on se, että Blenderissä luodun materiaalin nimi on ollut sama kuin juuri luotu .vtf - tiedosto (Kuva 18).



Kuva 18. Blenderin materiaalinäkymä.

Vtf - tiedoston lisäksi tarvitaan .vmt - tiedosto (Valve material type). Tämä tiedosto sisältää komentoja mitä päätekstuuria (.vtf tiedostoa) malli käyttää, minkä tyyppinen pinta mallilla on, sekä mallin mahdolliset muut ominaisuudet kuten läpinäkyvyys, kohoumakartoitus tai normaalikartoitus. Vmt - tiedoston voi luoda

VTFEditillä painamalla Create VMT File ja asettamalla sen jälkeen oikeat tekstuurit sekä asetukset materiaalille. Kun painetaan Create - painiketta, VTFEdit generoi .vmt - tiedoston. Tiedosto voidaan myös luoda manuaalisesti tekstinkäsittelyohjelmalla.

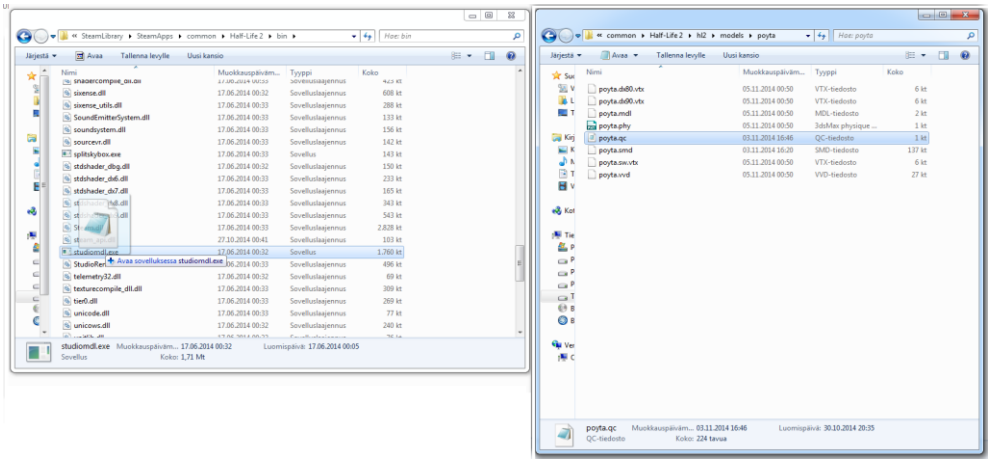


Kuva 19. Työssä käytetty .vmt tiedosto.

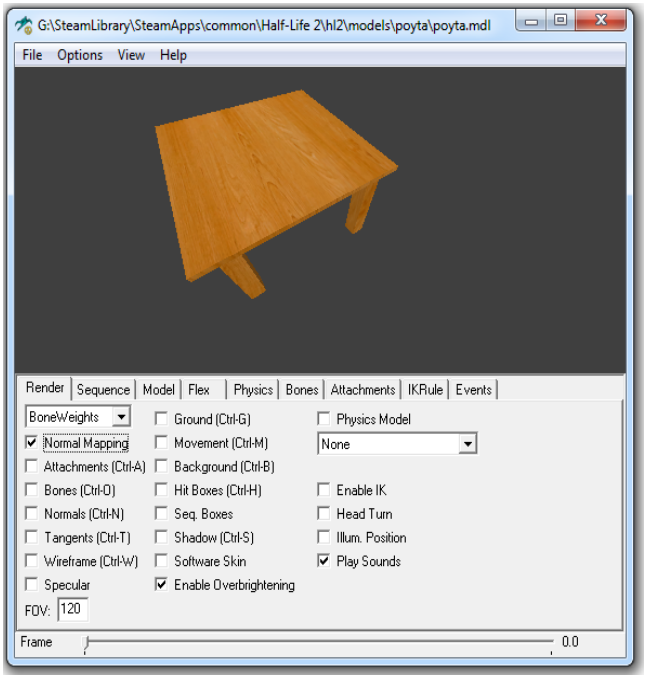
Pöytä 3D-mallin .vmt - tiedosto on yksinkertainen, koska mallissa on pelkkä diffuse - tekstuuri. VertexLitGeneric on yksi Source - pelimoottorin sävytintyyppi. $basetexture komento on mallin päätekstuuri ja se löytyy tässä työssä hakemistosta materials/models/poyta/poyta.vtf. $model 1 – komento ilmoittaa, että kyseinen materiaali kuuluu 3D-mallille, eikä Sourcen kenttäeditorin luomalle pelattavalle kartalle (Kuva 19). Selkeyden vuoksi poyta.vmt on tallennettu samaan hakemistoon poyta.vtf - tiedoston kanssa. Vmt ja .vtf - tiedostojen ei kuitenkaan ole pakko olla samassa hakemistossa. Tärkeintä on, että .vmt - tiedosto löytää .vtf - tiedoston oikean polun avulla.

Kun malli halutaan koota, voidaan se helposti raahata StudioMDL - mallin kompiloijaan (Kuva 20). StudioMDL aukeaa hetkeksi ja sulkeutuu sitten, kun malli on kompiloitu.

StudioMDL löytyy Source pelin bin - kansiosta muokkaamistyökalujen ollessa ladattuna (Software Development Kit). Kun kompilointi on valmis, on 3D-malli vietynä Source - pelimoottoriin. (Kuva 21) Mallia voidaan katsella Model viewerissä tai laittaa se pelin sisälle käyttämällä Sourcen kenttäeditoria.



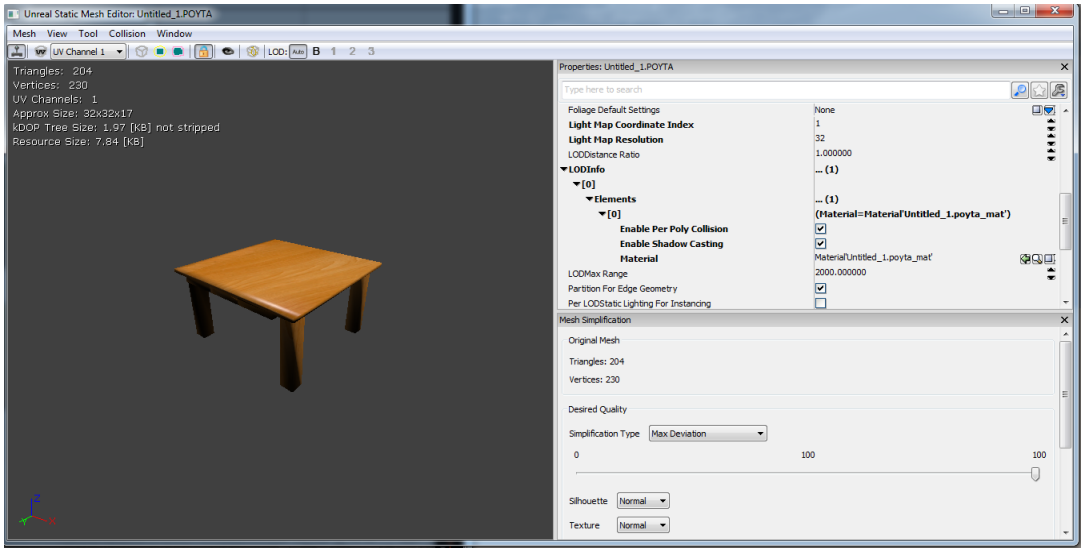
Kuva 20. Qc tiedoston vieminen kompilointiin.



Kuva 21. Pöytämalli Source pelimoottorissa.

5.2 Blender ja mallin vieminen Unreal Engineen

Blender ei tue automaattisesti .ASE - tiedostoformaattia, joten on mahdollista asentaa skripti, jolla se on mahdolllista. Hakukone Googlen avulla on mahdollista löytää lukuisia skriptejä, tässä työssä on käytetty versiota, joka löytyy osoitteesta https://code.google.com/p/ase-export-vmc/. Skripti asennetaan samalla tavalla kuin Blender Source Tools. Kun skripti on asennettuna, Blenderillä voidaan tallentaa .ASE - päätteisiä tiedostoja Export - valikon kautta. Tämän jälkeen voidaan käyttää menetelmää, mitä käytettiin mallin viemisessä 3ds Maxista Unreal Engineen, eli tuomalla malli ja tekstuuritiedosto Unreal Development Kitiin (Kuva 22).



Kuva 22. Pöytämalli Unreal Development Kitissä.

6 TULOKSET JA YHTEENVETO

Työlle asetetut tavoitteet saavutettiin, molempiin pelimoottoreihin saatiin luotua 3D-malli kummallakin 3D-mallinnusohjelmalla. Tämä oli mahdollista soveltamalla internetistä löytynyttä informaatiota. Kun saatiin selville, miten mallit tuodaan pelimoottoreihin, on mahdollisuus vertailla molempia pelimoottoreita ja 3D-mallinnusohjelmia.

Mallien tuomisessa oli selvästi eroavaisuuksia. Molemmilla ohjelmistoilla, 3ds Maxilla ja Blenderillä, mallien tuominen Unreal Engineen oli lähes samansuuruinen tehtävä. Blender tarvitsi vain skriptin .ase - mallien viemistä varten, kun taas 3ds Maxissa tämä toiminto oli jo valmiina. Unreal Enginessä mallien käsittely oli paljon helpompaa, sillä suuri osa työstä tehtiin Development Kitin puolella.

Source - pelimoottoriin mallien vieminen oli selvästi monimutkaisempaa. 3ds Maxilla Sourceen vieminen oli huomattavasti nopeampaa WallWorm - skriptin ansiosta. Skriptin takia sain tehtyä kaiken työn 3ds Maxissa. Skripti tekee asetuksieni perusteella itse .smd sekä .qc - tiedoston, muuttaa tekstuurit oikeaan muotoonsa ja kompiloi kaiken valmiiksi Sourceen. Sourcen toimintatapojen ymmärtämiseen kuluu huomattavasti enemmän aikaa.

Blenderillä mallien tuominen Source - pelimoottoriin oli hidasta. Blenderiin en löytänyt skriptiä, joka olisi tehnyt kaiken työn Blenderin sisällä. Tämän syystä minun täytyi tarkkaan opetella, miten malli kompiloidaan.

Pelkkä 3D-kappaleen luominen oli yhtä työlästä. Kuitenkin Blenderissä ja 3ds Maxissa oli paljon huomattavia eroavaisuuksia. Blenderissä käytetään huomattavasti vähemmän valikkojen takana olevia painikkeita. Hyvin moni toiminto oli käytettävissä pikanäppäimellä.

Kaikki ne, jotka aikovat kehittää peliä, voivat työssä saatujen tuloksien perusteella valita sopivimman vaihtoehdon 3D-mallinnusohjelmista ja pelimoottoreista omia käyttötarpeitaan varten.

LÄHTEET

BBC. 2012. Unreal games engine licensed to FBI and other US agencies. Viitattu 4.11.2014 http://www.bbc.com/news/technology-17535906

Birn, J. 2006. Digital Lightning & Rendering. Berkeley. New Riders.

Epic Games, 2014. Unreal Engine 3. Viitattu 26.11.2014 https://www.unrealengine.com/products/unreal-engine-3

Griggs, M. 2013. How to use UV maps. Viitattu 12.11.2014 http://www.creativebloq.com/how-use-uv-maps-8134077

Intergraph Computer Systems, 2001. Types of shading. Viitattu 1.11.2014 http://www.pcmag.com/encyclopedia/term/49185/phong-shading

Megaleecher. 2013. AMD Says There Could Be No DirectX 12 - Ever. Viitattu 25.9.2014 http://www.megaleecher.net/DirectX_12#axzz3Ld9Y1n7S

Nvidia Corporation. 2014. DirectX 11 Tessellation. Viitattu 8.9.2014 http://www.nvidia.com/object/tessellation.html

Puhakka, A. 2008. 3D-grafiikka. Helsinki. Talentum.

Sharma, K. 2013. Explained: What "DirectX" really is, How it works. http://www.softnuke.com/explained-directx-how-it-works/ Viitattu 19.10.2014

Slick, J. 2014 a. 3D Defined – What is 3D? Viitattu 23.8.2014 http://3d.about.com/od/3d-101-The-Basics/a/3d-Defined-What-Is-3d.htm

Slick, J. 2014 b. Anatomy of a 3D Model. Viitattu 29.8.2014 http://3d.about.com/od/3d-101-The-Basics/a/Anatomy-Of-A-3d-Model.htm

Slick, J. 2014 c. 7 Common Modeling Techniques for Film and Games. Viitattu 4.8.2014 http://3d.about.com/od/3d-101-The-Basics/a/Introduction-To-3d-Modeling-Techniques.htm

Slick, J. 2014 d. Current Gen Gameart Workflow – What Is Normal Mapping. Viitattu 5.9.2014 http://3d.about.com/od/3d-101-The-Basics/tp/Current-Gen-Gameart-Workflow-What-Is-Normal-Mapping.htm

Sokolowski, L. & Milewski, M. 2014. UV Mapping. Viitattu 7.10.2014 http://www.chocofur.com/5-uv-mapping.html

Thornhill, T. 2011. Nearly 40, but still looking good: How Pixar founders made the world's first 3D computer special effects in 1972. Daily Mail Online. Viitattu 30.7.2014. http://www.dailymail.co.uk/sciencetech/article-2034003/How-Pixar-founders-worlds-3D-graphics.html

Valve Corporation, 2014. Engine Licensing Information Sheet. Viitattu 4.11.2014 http://www.valvesoftware.com/SOURCE_InfoSheet.pdf

Valve Corporation, 2012. Introducing the Source Filmmaker. Viitattu 1.12.2014 http://www.sourcefilmmaker.com/post.php?id=7948&p=1

what-when-how. 2014. A 3D Primer (Introduction to Mudbox) (Digital Sculpting with Mudbox). Viitattu 2.11.2014 http://what-when-how.com/digital-sculpting-with-mudbox/a-3d-primer-introduction-to-mudbox-digital-sculpting-with-mudbox/

wiseGEEK. 2014. What Is A Game Engine. Viitattu 8.10.2014 http://www.wisegeek.com/what-is-a-game-engine.htm