

Joonas Seppänen

Hahmon mallintaminen 3ds Max - ohjelmalla ja sen tuonti XNA- pelimoottoriin

Opinnäytetyö
Tietojenkäsittely


Helmikuu 2013




MIKKELIN AMMATTIKORKEAKOULU

Mikkeli University of Applied Sciences

KUVAILULEHTI

 <p>MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences</p>	<p>Opinnäytetyön päivämäärä</p> <p>22.02.2013</p>				
<p>Tekijä(t) Joonas Seppänen</p>	<p>Koulutusohjelma ja suuntautuminen Tietojenkäsittely</p>				
<p>Nimeke</p> <p>Hahmon mallintaminen 3ds Max -ohjelmalla ja sen tuonti XNA-pelimoottoriin</p>					
<p>Tiivistelmä</p> <p>Opinnäytetyön aiheena oli kuvata 3ds Max Design 2011 -ohjelmalla tehdyn hahmon vieminen XNA Game Studio 4.0 -peliohjelmointiympäristöön. Mallinnusvaiheessa luotiin ihmismäinen hahmo, jonka sisään rakennettiin luuranko, jotta hahmolle saatiin tehtyä animaatio. Tämän jälkeen hahmo vietiin XNA:han ja testattiin, toimiko se samalla tavalla kuin mallinnusohjelmassa. Opinnäytetyö koostui siis kahdesta eri osasta, jotka molemmat olivat tärkeitä hyvän lopputuloksen saamiseksi.</p> <p>Opinnäytetyön ensimmäisessä osassa oli tarkoitus selvittää, millainen 3D-hahmosta tulisi tehdä, jotta sitä voitaisiin käyttää peliohjelmointiympäristössä. Mitä hahmolle pitäisi tehdä ennen kuin sen voi viedä XNA:han? Onko ohjelmointiympäristössä jonkinlaisia rajoituksia mallin suhteen? Työ toimi osittain tutoreaalina tai oppaana sellaisille henkilöille, jotka ovat kiinnostuneita peliohjelmoinnista tai mallintamisesta. Kaikki edelliset asiat selviävät opinnäytetyön edetessä.</p> <p>Käytännön toteutuksena tein muutamia versioita 3D-mallista, joille loin luurangon ja tein pienen animaation, jossa hahmo heilutti kättä sekä jalkaa. Tämän jälkeen XNA:ssa testasin, onnistuiko hahmon tuominen sinne. Opinnäytetyössä kävin läpi myös virhetilanteita, joita matkan varrella syntyi. Lisäominaisuutena kokeilin kääntää koodin suoraan Windows Phonelle. Näköjään se oli mahdollista, mutta koodiin joutui tekemään monia muutoksia.</p>					
<p>Asiasanat (avainsanat)</p> <p>3D-mallinnus, 3D Studio Max, peliohjelmointi</p>					
<p>Sivumäärä</p> <p>36</p>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Kieli</td> <td style="width: 50%;">URN</td> </tr> <tr> <td>Suomi</td> <td></td> </tr> </table>	Kieli	URN	Suomi	
Kieli	URN				
Suomi					
<p>Huomautus (huomautukset liitteistä)</p>					
<p>Ohjaavan opettajan nimi</p> <p>Jukka Selin</p>	<p>Opinnäytetyön toimeksiantaja</p>				

DESCRIPTION

 <p>MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences</p>		Date of the bachelor's thesis 22 February 2013
Author(s) Joonas Seppänen	Degree program and option Business Information Technology	
Name of the bachelor's thesis Character modelling with the 3ds Max program and importing to the XNA game engine		
Abstract The subject of this thesis was to present how a character made with the 3ds Max Design 2011 program would be exported to the Xna Game Studio 4.0 game programming environment. In the modelling phase the purpose was to make a humanoid character with an animated skeleton inside. After that the character was exported to the XNA and tested whether it worked as well as in the modelling program. This thesis consisted of two different parts which both were necessary to reach the final result. The first part of this thesis studied how to make a 3D character which could be used in the game programming environment. It also examined what the character should be like and what should be done before the character could be exported to the XNA and if there were some kinds of character limitations in the programming environment. This thesis was like a tutorial or guide for persons interested in game programming or modelling. All the issues listed were clarified as the study proceeded. In the practical part of this thesis I made some versions of 3D models for which I created skeletons and animations where the character moved his hand and foot. After that I tested how the characters should be imported to XNA. This thesis also introduced some error situations that came by. As an additional feature I tried to compile the code straightaway to a Windows Phone. It appeared that this was possible, but i needed to made many changes in the code.		
Subject headings, (keywords) 3D-modelling, 3D Studio Max, game programming		
Pages 36	Language Finnish	URN
Remarks, notes on appendices		
Tutor Jukka Selin	Bachelor's thesis assigned by	

SISÄLTÖ

1	JOHDANTO	1
2	TERMISTÖ.....	3
3	HAHMON MALLINTAMINEN.....	4
3.1	Mallin suunnitteleminen	4
3.2	Eri osien mallintaminen.....	5
3.3	Materiaalien tekeminen	6
3.4	Muut asiat	8
3.4.1	Polygonien määrä.....	8
3.4.2	Riggaus	9
4	HAHMON VIEMINEN XNA GAME STUDIOON	13
4.1	Content Pipeline.....	13
4.2	Muut asiat	16
5	ONGELMATILANTEITA	18
5.1	3ds Max	18
5.2	XNA Gamestudio	20
6	CASE.....	24
6.1	Hahmon suunnittelu.....	24
6.2	Hahmon mallintaminen	25
6.3	Hahmon teksturointi	28
6.4	Luurangon luominen.....	29
6.5	Skinning.....	29
6.6	Animaation luominen	30
6.7	Hahmon vieminen XNA:han	32
7	WINDOWS PHONE SOVELLUS	33
8	LOPETUS	35
	LÄHTEET.....	37

1 JOHDANTO

Opinnäytetyöni käsittelee kahta osa-aluetta, 3D-mallin tekemistä sekä sen viemistä Microsoft XNA Game Studio 4.0 -pelimoottoriin. Teen 3D-mallin Autodesk 3ds Max Design 2011 -ohjelmalla, jonka valitsin mallin tekemiseksi aikaisemman kokemuksen perusteella ja design-versio ei ole ihan niin raskas ohjelma kuin tavallinen versio, joten se pyörii paremmin koneellani. Lisäksi uudemmista versioista minulla ei ole kokemusta ja niiden hankinta olisi ollut paljon hankalampaa ja kalliimpaa. Toisena vaihtoehtona ohjelmaksi olisi ollut Autodeskin Maya -ohjelma, mutta minulla ei ole siitä lainkaan kokemusta, joten sen opettelu olisi vienyt paljon aikaa ja opinnäytetyö olisi ollut huomattavasti työläämpi.

Peliohjelmointiympäristöksi valitsin Microsoft XNA Game Studio 4.0:n, koska se on ainut peliohjelmointiympäristö, jota olen käyttänyt. Se on tällä hetkellä uusin versio kyseisestä ohjelmasta ja se on monipuolisempi kuin aikaisemmat versiot. XNA Game Studio on lisäosa Microsoftin Visual Studio -ohjelmaan ja se perustuu C#-ohjelmointikieleen. Visual Studiosta minulla on käytössä 2010-versio johtuen siitä, että koulussakin käytettiin samaa versiota. Eri versioilla tehdyt ohjelmat eivät välttämättä toimi ilman muutoksia toisissa versioissa, joten testaaminen on helpompaa samoilla välineillä, mitä koulussa käytettiin.

Opinnäytetyön 3D-mallinnus vaihe käsittelee hahmon mallintamista perinteisin menetelmin. Ennen mallin tekemistä suunnittelen ja piirrän hahmon paperille, ja sen jälkeen alan vasta mallintamaan sitä 3ds Maxilla. Luon hahmosta olion näköisen, ja teen sille vartalon, kädet, jalat sekä pään. Käytän eri osien mallintamiseen erilaisia menetelmiä ja testailen, miten kyseinen osa kannattaisi mallintaa. Teen hahmosta yksinkertaisen, koska sitä on helpompi testata, ja yksityiskohtaisen hahmon tekemiseen menisi liikaa aikaa. Hahmolle luodaan luuranko, joka liitetään kiinni hahmon raajoihin. Tämän avulla saan tehtyä hahmolle erilaisia liikkeitä animaation avulla.

Lisäksi testaan Microsoftilta saatua valmista hahmoa, kuinka se käyttäytyy ja miten sen liikkeet eroavat omatekemäni mallin liikkeistä. Kyseinen hahmo on ilmainen, testauskäyttöön tarkoitettu hahmo, ja se on viimeisen päälle aidon näköinen ihmishahmo. Mahdollisesti sitä voitaisiin käyttää uusimmissakin peleissä. Hahmon luurangossa on

huomattavasti enemmän luita kuin omassa hahmassani, joten odotan innolla testaustuloksia.

Mallintamisen jälkeen tuon hahmon peliohjelmointiympäristöön, jossa voin todentaa, että malli toimii siinä ja tekee tiettyjä liikkeitä. Tässä vaiheessa ohjelmoin hahmon tekemään animaatioissa tekemiä liikkeitä, säädän kameran sekä mallin paikkaa, ja piirrän hahmon ruudulle. Tähän osioon minulla on valmis koodi, jota käytin yhdellä kursilla, koska opinnäytetyön tarkoituksena ei ole tehdä vaativaa koodia tai "oikeaa" kolmiulotteista peliä. Koodia muuttamalla saan tehtyä tarpeellisia testejä, ja se riittää tässä opinnäytetyössä. Lisäksi tässä vaiheessa testataan, kuinka koodi kääntyy Windows Phonelle ja mitä eroja siinä on suhteessa Windowsiin.

Teen opinnäytetyöni Mikkelin ammattikorkeakoululle, koska kiireisellä aikataululla en saanut mukaan mitään yritystä, jolle olisin voinut tehdä kyseisen opinnäytetyön. Tarkemmin ajateltuna työ on eräänlainen opas 3D-mallintamisesta sekä peliohjelmoinnista kiinnostuneille henkilöille, ja hyvin onnistuessaan lukijat saattavat kiinnostua kyseisistä aiheista ja innostuvat tekemään omia malleja sekä pelejä.

Valitsin tämän aiheen, koska kiinnostuin kurssien myötä 3D-mallintamisesta ja peliohjelmoinnista. Lisäksi kyseiset aiheet ovat herättäneet paljon kiinnostusta nykypäivän Suomessa ja peliteollisuus kasvaa koko ajan. Esimerkiksi viime aikoina suomalaiset peliyhtiöt kuten Bugbear, Frozenbyte ja RedLynx ovat tehneet hyvinkin suosittuja pelejä, jotka myyvät maailmanlaajuisesti.

Opinnäytetyöni tutkimusongelmana on, kuinka tehdään sellainen malli, jota voidaan käyttää XNA:ssa sekä kuinka mallin saa tuotua peliohjelmointiympäristöön siten, että se toimii oikein ilman virheitä ja näyttää sellaiselta kuin sen pitää. Työn rakenne on seuraavanlainen: Tekstiosat koostuvat teoriaosuudesta sekä omista kokemuksistani mallia tehdessä. Teoriaosuudessa käsittelem muun muassa hahmon suunnittelua, eri osien mallintamista, materiaalien tekemistä sekä muita seikkoja, jotka ovat tärkeitä vaiheita mallinnuksessa. Toinen tärkeä osa teoriaosuutta on hahmon vieminen XNA-peliohjelmointiympäristöön, jossa käsitellään, miten se tapahtuu ja mitä ongelmia siinä voi tulla matkan aikana. Teoriaosuudet koostuvat siis kahdesta pääluvusta, hahmon mallintamisesta sekä hahmon viemisestä XNA Game Studioon. Omaa tekstiäni kuvaan casetyyppisesti, eli kirjoitan hahmon mallintamisen vaiheista ja sen tuomisesta

XNA Game Studioon. Tämä johdantoluku kertoo, mitä opinnäytetyö pitää sisällään, ja lopussa oleva lopetusluku pitää sisällään loppupäätelmät, ratkaisun tutkimusongelmaan sekä jatkokehityksen ja tulevaisuudennäkymien tarkastelun.

2 TERMISTÖ

Tässä luvussa käydään läpi opinnäytetyön tärkeimmät termit, sillä 3D-mallintamisessa ja ohjelmoinnissa oleva sanasto on välillä hankalaa ymmärtää. Termistön avulla selvitetään lukijalle tärkeät sanat, jotka täytyy tietää ennen kuin voi ymmärtää niiden merkityksen. Lisäksi termistö on melkein aina englanninkielistä, joten sanoja on jouduttu kääntämään suomalaisemmiksi. Jotkin sanat voivat olla siis suomen ja englannin kielen sekoitusta, joten siinä voi mennä sekaisin jos ei tiedä mitä kyseinen sana tarkoittaa.

Polygon tarkoittaa monikulmiota, josta 3D-malli koostuu. Polygonissa on vähintään kolme kulmaa, mutta yleensä polygonit ovat nelikulmioita. (Mallintaminen 2012.)

Vertex tarkoittaa polygonin kulmapisteessä olevaa kärkeä. Polygonissa on vähintään kolme verteksiä. (Mallintaminen 2012.)

Edge tarkoittaa vertex pisteiden välissä olevaa viivaa eli särmää. Jokaisessa polygonissa on vähintään kolme edgeä. (Mallintaminen 2012.)

Modifier tarkoittaa 3ds Max -ohjelmassa muunninta, jolla annetaan erilaisia lisäominaisuuksia 3D-objekteille (Orava 2011, 7).

*Riggau*s tarkoittaa 3D-hahmolle tehtävää luurankojärjestelmää (Orava 2011, 7).

Skinning on tekniikka, jossa luuranko liitetään 3D-hahmon osiin (Orava 2011, 7).

Animointi on prosessi, jossa 3D-ohjelmalla tehty objekti laitetaan esimerkiksi liikkumaan. Animointi tapahtuu aikajanalle tehtävien keyframejen avulla (Ahola 2011, 12).

Content Pipeline on XNA:ssa oleva luokkakirjasto, jonka avulla siihen tuodaan ulkopuolista sisältöä, kuten kuvia, musiikkia sekä 3D-malleja (What is the Content Pipeline? 2012).

3 HAHMON MALLINTAMINEN

Tässä luvussa kerrotaan, mistä mallintaminen lähtee käyntiin, eri osien mallintamisesta sekä muita seikkoja, joita kannattaa ottaa huomioon mallintamisessa. Myös materiaalien tehtävä selvitetään ja se, miksi niitä kannattaa käyttää. Mallinnus on hyvin pitkäjänteistä työtä, joten kaikki asiat kannattaa suunnitella valmiiksi, ennen kuin lähtee tekemään mallintamista. Kun seuraavat mallinnukseen liittyvät asiat on suunniteltu, niin mallintaminen helpottuu ja nopeutuu.

3.1 Mallin suunnitleminen

3D-hahmon mallintaminen lähtee liikkeelle aina mallin suunnittelusta. Helpoin tapa päästä alkuun on etsiä kuvia ja ideoita Internetistä. (Stoneham 2010, 15.) Hahmon ensimmäinen luonnos kannattaa piirtää aluksi paperille, jonka jälkeen sen voi siirtää tietokoneelle johonkin kuvankäsittelyohjelmaan, jota voi myöhemmissä vaiheissa muokata tarpeen tullen. Luonnoksia kannattaa piirtää useampiakin, joten niitä voi tarvittaessa yhdistellä keskenään. (Viitapohja 2010, 16.)

Hahmomallinnuksessa tulee tietää, kuinka ihmiskeho toimii ja miten luut liikkuvat, joten on tärkeää perehtyä hieman ihmisen anatomiaan ennen kuin aloittaa mallintamisen (Stoneham 2010, 26). Hahmon suunnittelussa on tärkeää miettiä hahmon tarkoitusta ja sitä, mihin käyttöön se tulee. Jos hahmoa käytetään tietokonepelissä, niin pelimoottoreilla on tietynlaisia rajoituksia koskien geometrisien elementtien määrää. Geometriset elementit tarkoittavat mallintamisessa polygoneja. Mikäli malli koostuu liian suuresta määrästä polygoneja, niin se ei toimi niin sujuvasti kuin pitäisi. (Franson & Thomas 2006, 2.)

Hahmomallintamisessa ihmisen mallintaminen on huomattavasti työläämpää kuin kuvitteellisen hahmon mallintaminen. Tämä johtunee siitä, että kuvitteellista mallia ei yleensä tehdä niin tarkasti ja hahmon eri osista voidaan tehdä hieman aidosta poikkeavia. (Viitapohja 2010, 17.) Tämän takia teen omasta hahmostani kuvitteellisem-

man näköisen, mutta käytän lisäksi Microsoftilta saamaani ihmishahmoa vertailukohteenä XNA:ssa.

Hahmomallinnuksessa kasvot kuvaavat eniten hahmon persoonaa, koska kasvojen ja silmien muodot ovat yksi tärkeimmistä osista hahmoa. Tämän takia kasvot herättävät katsojan kiinnostuksen. (Stoneham 2010, 32.) Myös tärkeä osa hahmoa on sen raajat, joiden avulla malli herättää ihmisten tunteita. Lisäksi hahmon ulkoinen olemus, eli se miltä hahmo näyttää katsojalle, kannattaa miettiä valmiiksi. Esimerkiksi vaatetus ja hahmon väritys kuuluvat mallin ulkoiseen olemukseen. (Viitapohja 2010, 18.)

Hahmon mallinnuksessa kannattaa käyttää apuna referenssikuvia, jotka ovat mittakaavaltaan samankokoisia kuin tuleva malli. Referenssikuvat on hyvä ottaa sekä edestä että sivusta, ja niissä on oltava myös sama mittasuhte, koska muuten hahmosta tulee erikokoinen eri kuvakulmista. Valmiit referenssikuvat viedään 3d-mallinnusohjelmaan siten, että edestä otettu kuva näkyy mallinnusohjelman etukuvakulmasta, ja sivusta otettu kuva näkyy sivukuvakulmassa. Referenssikuvien tarkoitus on, että mallintaja rupeaa mallintamaan hahmoa suoraan referenssikuvien päälle käyttäen niitä apunaan. Hankaluuksia voi esiintyä, mikäli referenssikuvat ovat erikokoisia. Silloin täytyy valita, kumpaa kuvaa käyttää apunaan. (Viitapohja 2010, 19.)

3.2 Eri osien mallintaminen

Eri osien mallintamisessa kannattaa miettiä, kannattaako osa rakentaa yksinkertaisesti vai mahdollisimman tarkasti. Asiaan vaikuttavat muun muassa se, animoidaanko malliin liikkeitä ja viedäänkö malli mahdollisesti johonkin 3D-peliin. Jos mallia käytetään näissä menetelmissä, silloin kannattaa tehdä yksinkertaisempia muotoja eri osiin tai lihaksiin, sillä animoidun hahmon liikkeet voivat vääristyä, jos mallintaa liian tarkasti aidon näköisen lihaksen. Mikäli hahmon käyttötarkoitus on tehdä mahdollisimman oikean näköinen, silloin kaikki osat ja lihakset kannattaa mallintaa mahdollisimman tarkasti, jotta hahmo näyttää niin realistiselta kuin mahdollista. Mitä yksityiskohtaisemman mallista tekee, sitä monimutkaisemmaksi ja polygoneja enemmän käyttäväksi mallin tekeminen menee. (Viitapohja 2010, 17–22.)

3D-mallintamisessa voidaan käyttää useita eri menetelmiä, kuten tavallisessa piirtämisessäkin. Melkein suosituin hahmomallinnus menetelmä on polygonimallinnus, jossa

on tarkoitus rakentaa mahdollisimman yksinkertaisista palasista osia, joita muokataan erilaisilla Modifier-työkaluilla. Kun jokin kappale on tehty, siihen aletaan lisätä polygoneja ja vertexejä, joiden avulla kappaleesta saadaan muokattua oikeannäköinen. Tämä tapahtuu valitsemalla objektin ja painamalla hiiren kakkosnappia ja valitsemalla Convert to Editable Poly -valinnan. Polygoneja lisäämällä saadaan aikaan näyttävämpiä muotoja. Edit Poly -tilassa on myös erilaisia veto-työkaluja, kuten Extrude sekä Bevel, joiden avulla voidaan vetää polygoneja ulospäin valmiista pinnasta. Tämä on yksi tapa saada lisää polygoneja kappaleeseen ja näillä keinoilla saadaan tehtyä esimerkiksi hahmon kädet, sormet sekä jalat. (Mallintaminen 2012.)

Hahmosta ei tarvitse tehdä kuin toinen puoli, koska erilaisilla työkaluilla, kuten Symmetry- sekä Mirror-Modifiereilla voidaan tehdä automaattisesti toinenkin puoli. Tämä helpottaa suuresti hahmon tekemistä. Perusmuodot onnistuvat näillä helposti, mutta mikäli hahmon rakenne ei ole symmetrinen, niin silloin näitä ei voi käyttää. (Character modeling tutorial - Sackboy 2012.)

Kasvojen ja pään mallintaminen on hankalin osa ihmismallin luomisessa, koska pään muodot ovat sellaisia, että siihen pitää käyttää paljon polygoneja, että se näyttäisi aidolta. Päänkin voi aluksi tehdä Mirror- tai Symmetry-työkaluilla, mutta silloin siitä ei tule niin aidonnäköistä. Tämän takia pään mallintamisessa kannattaa käyttää silmukkarakennetta, joka tapahtuu tekemällä edge looppeja siihen. *Edge loop* tarkoittaa vertexien välisten särmien eli edgejen jonoja, jotka menevät nelisivuisten vertexien läpi. (Lehtinen 2007.) Aluksi kannattaa aloittaa yksinkertaisista muodoista, jonka jälkeen ruvetaan lisäämään polygonien määrää, jolloin kasvoihin ja päähän saadaan oikeanlaisia muotoja. Pään mallintaminen on tunnetusti vaikeimpia osia hahmon mallintamisesta ja se vie myös eniten aikaa. (Viitapohja 2010, 23.)

3.3 Materiaalien tekeminen

Teksturoinnilla saadaan 3D-objekti näyttämään aidommalta. Yleensä teksturoinnissa liitetään kuvankäsittelyohjelmalla tehty kuva hahmon pinnalle. Tällä tavalla hahmo saadaan näyttämään yksityiskohtaisemmalta, ja sillä voidaan korostaa tiettyjä osia hahmosta. UV Mapping -tekniikka on 3D-objektin teksturointiin yleisin käytetty teksturointimenetelmä. Siinä on tarkoitus valita aluksi pinta, mihin UV Map liitetään. Sen

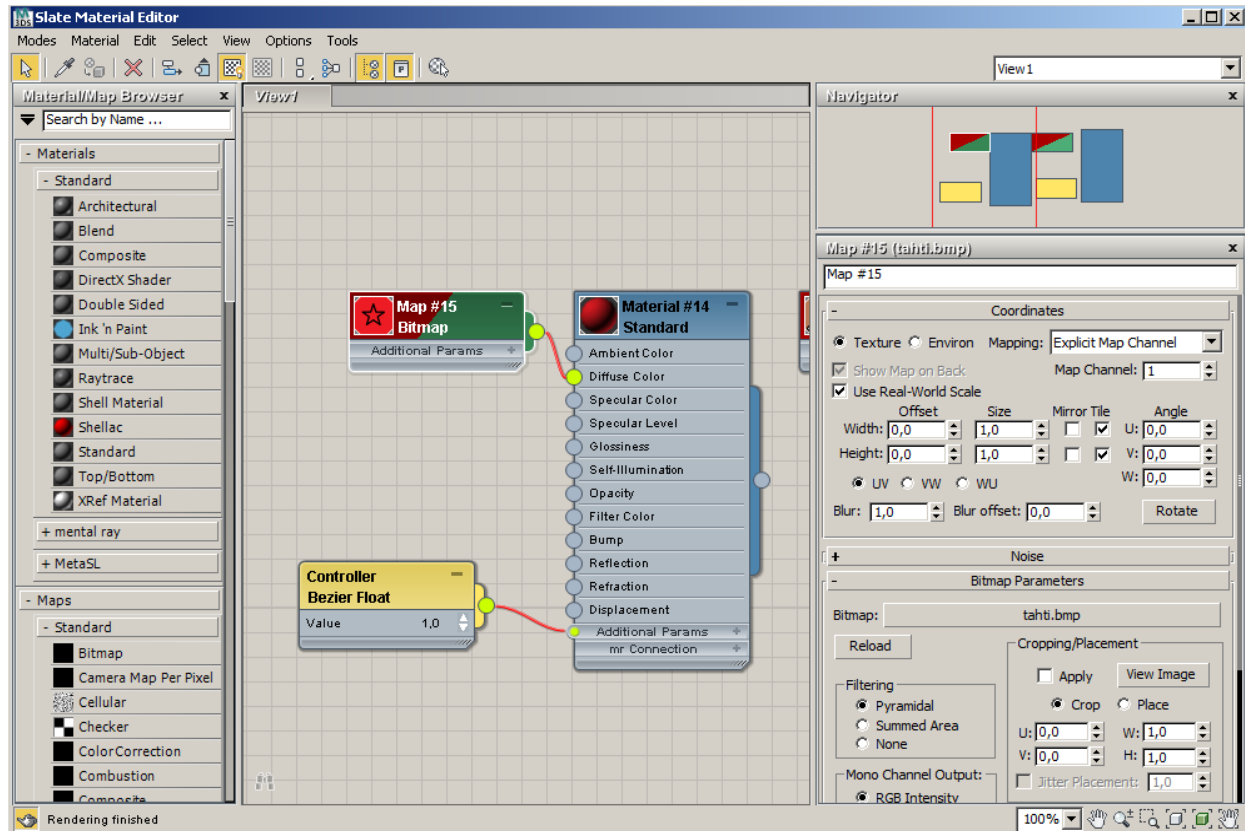
jälkeen kuvankäsittelyohjelmalla tehty kuva liitetään karttamaisesti pinnan päälle, johon se asettuu juuri oikealla tavalla. (Viitapohja 2010, 35–36.)

Ennen teksturointivaihetta on hyvä korjata hahmon eri osissa olevat virheet, koska teksturointi voi mennä pilalle, mikäli hahmon geometriassa on isoja virheitä. Geometrian virheet voi tarkastaa käyttämällä STL Check Modifieria. Se tarkastaa automaattisesti geometrian silmukoiden virheet ja ilmoittaa, onko testi mennyt läpi. Muut virheet on korjattava käsin, kuten päällekkäin menevät edget. Virheet on helppo havainnollistaa *wire frame* -näkyvässä, koska virhealueet ovat punaisen värisiä. Näitä alueita korjaamalla saadaan mallista virheetön, ja teksturointi helpottuu. STL Check Modifierin pitäisi näyttää korjauksen jälkeen, että virheitä ei enää ole. Hahmon kaikki osat ovat tässä vaiheessa vielä yhdistettyinä, mutta tämän jälkeen on hyvä erotella hahmon eri osat toisistaan, jolloin teksturointi ja UV-mappaus onnistuu helpommin pienissä erissä. Erottelu voi tapahtua esimerkiksi erottamalla ainakin jalat, kädet, vartalo sekä pää erillisiksi osiksi. Halutessaan voi lisätä eroteltavia osia, jolloin UV-mappauksesta tulee vieläkin helpompaa. (Franson & Thomas 2006, 63–66.)

Tekstuurikuvien bittimäärän tulisi olla jaollinen neljälle, sillä muuten XNA Gamestudio ei välttämättä osaa käsitellä tekstuuritiedostoa. Lisäksi kuvista voi tulla niin raskaita, että tietokone joutuu laskemaan turhan pitkään. (Ylimäki 2011.) Yleisesti kuvat ovat 24-bittisiä. Kuvan koon laskemiseen kannattaa käyttää kaavaa $(2)^n \times (2)^n$, joka toimii nelinkertaisesti.

Autodesk 3ds Max Design 2011 -ohjelmassa on materiaalien tekemiseen tarkoitettu kaksi erilaista materiaalieditoria, uudempi Slate Material Editor sekä vanhempi Compact Material Editor, joissa voi valita materiaaleja valmiista materiaalikirjastosta tai tuoda oman materiaalin bitmappina. Slate Material -editorissa on enemmän ominaisuuksia, kuin Compact Material Editorissa, ja se on tarkoitettu enimmäkseen uusille mallintajille. Compact Material Editor on taas helppokäyttöisempi vanhan koulukunnan mallintajille, jotka eivät halua opetella uusia menetelmiä. Editoreissa on myös monenlaisia asetuksia, joita materiaaleihin voi lisätä. Materiaalieditoriin pääsee esimerkiksi painamalla m-kirjainta ja Slate Material Editor aukeaa oletuksena siihen. Kuvassa 1 on Slate Material Editor. Materiaalien tarkoituksena on tuoda 3D-objektiin syvyyttä ja näkyvyyttä, koska ilman materiaaleja malli näyttäisi melko luonnottomalta. Materiaaleja voi siirrellä sekä käännellä 3D-objektin päällä miten haluaa, ja mate-

riaalin voi myös valita kohdistuvan tiettyyn osaan mallista. Modifier listassa on lisäksi erilaisia valintoja, joita materiaaleille voi tehdä, kuten materiaalien asettelu mallin päälle erilaisin keinoin. (Murdock 2010, 437–441.)



KUVA 1. Slate Material Editor

3.4 Muut asiat

Mallintamisvaiheen jälkeen hahmolle tehdään loppuviimeistely, jossa on tarkoituksena vähentää polygonien määrää. Lisäksi hahmolle luodaan luuranko, joka yhdistetään hahmoon. Tämän jälkeen luodaan Keyframe-animaatio.

3.4.1 Polygonien määrä

Parhaimmillaan hahmossa tulee olemaan jopa yli 10000 polygonia, joten sen vieminen peliohjelmointiympäristöön on raskasta ja se vaatii tietokoneesta paljon tehoja (Franson & Thomas 2006, 68). Esimerkiksi *Gears of War* -pelin Marcus-hahmoon on käytetty 15000 polygonia (Reed 2011, 474).

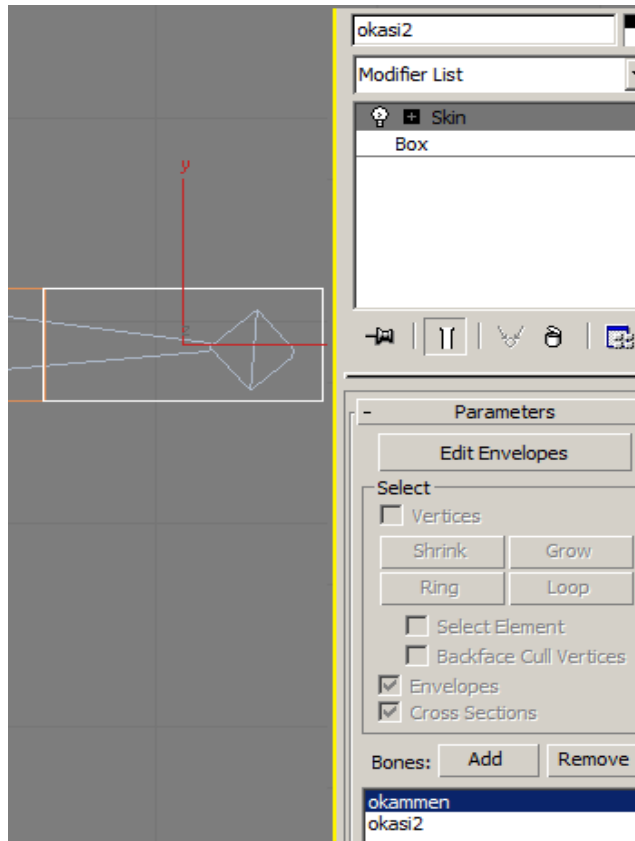
Polygonien määrää voi pienentää melkein puoleen ilman, että hahmon silmukkarakenne hajoaa suuresti. Tähän on olemassa erilaisia modifiereita, kuten Optimize Modifier ja MultiRes Modifier. Vaikka polygonien määrä tippuu puoleen, niin hahmon muoto säilyy vielä suurimmaksi osaksi samanlaisena kuin ennen modifierien käyttöä. (Franson & Thomas 2006, 68–69.) Painamalla 7-nappia, ohjelmaan tulee näkyviin kuvan 2 mukainen Statistics-ruutu, joka ilmoittaa muun muassa polygonien, vertexien ja edgejen määrän. Se myös ilmoittaa FPS:n eli Frames Per Seconds arvon. Viewport Configuration ikkunan Statistics välilehdeltä voi myös laittaa Statistics-ruudun näkyviin sekä vaihdella siinä näkyviä elementtejä. (Murdock 2012, 72–73.)

	Total
Polys:	892
Verts:	733
FPS:	5,071

KUVA 2. Statistics-ikkuna

3.4.2 Riggaus

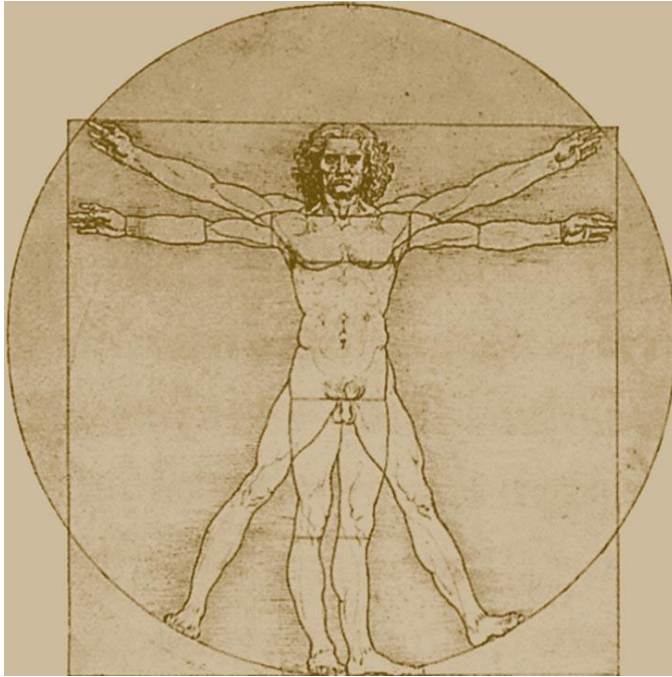
Riggaus on prosessi, jossa rakennetaan 3D-hahmolle luuranko, jota käytetään 3d-animaatioissa tai peleissä. Toisin sanoen yksinkertaisuudessaan riggaus tarkoittaa 3D-mallin sisään laitettavaa luurankoa. Hahmolle rakennetaan luuranko, joka vastaa aitoa luurankoa. Yksittäisten luusarjojen avulla saadaan muodostumaan rigejä, ja ne taas muodostavat kokonaisuudessaan luurangon. Tämän jälkeen luuranko liitetään 3D-hahmoon skinning-tekniikan avulla, jossa jokaisen vertexin tulee olla kiinnitettynä johonkin luuhun. Näistä muodostuu vertexryhmiä, joihin lisätään painoarvo, jonka mukaan säädetään yksittäisen luun vetovoima suhteessa vertexryhmään. (Viitapohja 2010, 38.) Luiden painoarvot tulee olla väliltä 0–1, mutta ei kuitenkaan tasan 0. Mitä pienempi arvo, sitä vähemmän vertex vaikuttaa kyseiseen luuhun. 1 arvo tarkoittaa, että luu vaikuttaa täysin vertexiin. Painoarvot näkyvät hahmossa erilaisilla väreillä, jonka avulla voi nähdä, mihin vertexeihin tietty luu vaikuttaa. (3ds Max Model Rigging. 2012.) Lisäksi skinning-tekniikassa on tarkoitus määrittää, mitkä vertexit liikuttavat mitäkin luuta (Nieminen 2009, 9). Kuvassa 3 näytetään, mihin luihin kappale okasi2 on liitetty.



KUVA 3. Kappaleeseen lisätty Skin Modifier

3D- hahmon luomisvaiheessa kannattaa ottaa huomioon hahmon asento, koska luiden liittäminen vertexeihin skinning-vaiheessa helpottuu. Optimaalinen hahmon asento on samanlainen kuin Da Vincin Vitruvialaisella miehellä. Asennossa jalat ovat levällään, ja polvet ovat hieman koukussa. Kädet ovat suorina sivuilla ja noin 90°:een kulmassa. Pää on suorana kroppaan nähden. (Nieminen 2009, 12.) Kuvassa 4 Da Vincin kuuluisa Vitruvialainen mies.

Luurankoa aletaan siis rakentaa yksittäisistä luokappaleista, jotka yhdistyvät toisiinsa automaattisesti. Luita voi myös kopioida kuten muitakin kappaleita Mirror-työkalulla, joten esimerkiksi toista kättä ja jalkaa ei tarvitse tehdä. Luita voi käännettä ja väännellä hahmon sisällä paremmin sopiviksi. (Lehtinen 2010.)



KUVA 4. Vitruvialainen mies (The Vitruvian Man 2012)

Kaikkien luiden ei kuitenkaan tarvitse tehdä mitään, vaikka ne ovat kiinnitettynä verteksiin, sillä jotkin luut avustavat vain toisten luiden toimintoja. Luiden liikuttamiseen on luotu kaksi erilaista tapaa, suora kinematiikka (Forward Kinematics) sekä käänteinen kinematiikka (Inverse Kinematics). Näillä kahdella tavalla on selvä eroavaisuus, suorassa kinematiikassa isä-kappale (parent) liikuttaa lapsikappaletta (child), kun taas käänteisessä kinematiikassa kappaleketjun viimeinen osa eli maali (goal) liikuttaa koko kappaleketjua. Näitä kahta tapaa voi myös yhdistellä, minkä tuloksena luiden liikuttaminen näyttää aidommalta. Isäkappale tarkoittaa sitä kappaletta, joka on ensimmäinen luu luotavassa osassa. Jos jalan luita aletaan luoda esimerkiksi lonkkaluusta, niin silloin lonkkaluu on isäkappale. Mikäli lonkkaluuta liikutetaan, niin kaikki muut siihen kiinnittyneet luut liikkuvat sen mukana. Luurankoon tulee myös juuriluu tai toiselta nimeltään kontrollikappale, joka liikuttaa koko luurankoa. Kaikki isäkappaleet ovat linkitettyinä kontrollikappaleeseen. (Nieminen 2009, 9–11.)

3D-malliin tulee isoja määriä luita, joten on hyvä nimetä kaikki luut selkeästi, että tietää myöhemmissä vaiheissa, mikä luu on kyseessä. 3ds Maxissa on valikko, josta näkee kaikki skenessä olevat objektit nimen perusteella. Tämän avulla on helppo katsoa, mitkä luut tulevat kiinni mihinkin vertexeihin. (Nieminen 2009, 12.) Tähän valikkoon pääsee painamalla Maxin työkalupalkissa olevaa select by name -nappia. Vali-

kosta voi suodattaa, mitä objekteja listaus näyttää ja sen avulla voi myös valita objekteja. (Murdock 2010, 22.)

3ds Maxissa on mahdollisuus luoda itse luuranko tai käyttää valmista bibe-luurankoa, mutta bibe-luurangon vieminen XNA:han tuottaa suuria ongelmia, koska se sisältää liian paljon luita, joten sitä ei voi tuoda XNA:han. Käyttämällä bones-luukappaleita saa paljon monipuolisemman luurangon, kuin luomalla pelkän bibe-luurangon. Itse tehtyyn luurankoon saa valita, kuinka luut liikkuvat siinä, mutta bibe-luurangossa luiden liikkeet ovat vakioita, eikä niitä voi muokata niin paljon kuin itse tekemässä. Luiden muokkaamiseen on tarkoitettu Edit Poly Modifier, jonka avulla luiden kokoa ja muotoa voi muokata. (Murdock 2010, 921–922.)

Sitten kun riggaus ja skinning vaiheet ovat tehty, niin on aika ruveta animoimaan hahmolle liikkeitä. Tämä onnistuu parhaiten erottamalla luuranko hahmon muista osista ja luoda animaatio pelkän luurangon avulla. Animointi helpottuu, koska muut osat eivät ole tiellä. Myös tietokoneen ei tarvitse käyttää kaikkia tehojaan pelkän luurangon animoinnissa. (Nieminen 2009, 12.) Animointia helpotetaan luomalla siihen käyttöliittymä. Tarkoituksena on kontrolloida hahmon liikkeitä luomalla eräänlaisia kontrollikappaleita siihen. Kontrollikappaleiden avulla voidaan liikuttaa tiettyä osaa hahmosta. Kontrollikappaleen luontivaiheessa sille luodaan uudet kontrollerit positiolle, rotaatiolle sekä skaalaukselle. Tämä tehdään Freeze Transform-komennon avulla, ja samalla kontrollikappaleen alkuperäiset asetukset nollaantuvat. Mikäli hahmon haluaa palauttaa takaisin oletusasentoon, täytyy kaikkiin transformaatioarvoihin laittaa arvoksi 0. (Nieminen 2009, 17.)

Hahmolle voidaan tehdä keyframe animaatio 3ds Maxissa olevan animaatiotyökalun avulla. Tarkoituksena on luoda aikajanelle keyframeja, joissa tapahtuu tietynlaisia liikkeitä. Keyframeja luodaan aikajanelle valitsemalla auto key tai set key. Auto key on helpompi käyttää, mutta set key on paljon monipuolisempi tapa luoda keyframeja. Kun jompikumpi on valittu, sen kuva muuttuu punaiseksi ja valinta lukittuu siihen. Auto keytä käytetään siten, että siirrytään haluamalle kohdalle aikajanelle, jolloin auto key luo siihen automaattisesti keyframen. Sitten voidaan esimerkiksi liikuttaa hahmon kättä johonkin asentoon. Sitten taas liikutaan aikajanelle jolloin siihen syntyy taas uusi keyframe, johon voidaan tehdä uusi liike. Tällä tavalla saadaan tehtyä perus animaatio helposti. Set key ei luo automaattisesti keyframeja vaan ne pitää laittaa itse haluamalle

kohdalle ja painaa sitten set key -nappia. Valikossa on suodatin, jossa määritellään, mitkä valinnat set key:llä voidaan tehdä. Aikajanalla olevat keyframet ovat jaettu neljään ryhmään, joissa jokaisessa on eri värikoodi. Punainen väri tarkoittaa sijaintia, vihreä kiertoa, sininen skaalaa sekä tummanharmaa parametrejä. Nämä keyframet näkyvät ainoastaan sille objektille, joka on valittuna. Jos mitään ei ole valittu, niin aikajanalla ei näy mitään. (3DS Max Tutorial – Basic Animation Techniques 2006.)

4 HAHMON VIEMINEN XNA GAME STUDIOON

Tässä luvussa käydään läpi, miten tiedoston pystyy viemään 3ds Max -ohjelmasta Microsoft XNA Game Studio -peliympäristöön, ja mitä kaikkea se pitää sisällään. Tässä käydään myös läpi, mitä XNA:n puolella pitää tehdä, että 3D- hahmon saa tuotua onnistuneesti sinne sekä kerrotaan mitä tiedostomuotoja XNA hyväksyy ottamaan vastaan.

4.1 Content Pipeline

XNA Gamestudioon viedään Content Pipelinen avulla ulkopuolista sisältöä kuten musiikkia, ääniä, tekstuureita, 3D-elementtejä sekä kaikkea muuta, mikä tulee pelin ulkopuolelta. Content Pipeline on siis eräänlainen luokkakirjasto, jota käyttämällä oman koodin kirjoittamista ei tarvitse niin paljoa ulkopuolisen sisällön tuomisessa, koska se osaa käsitellä kaiken tarpeellisen automaattisesti. (Heikkinen ym. 2008, 10.)

3D-mallit tuodaan luokkakirjastoon yleensä .fbx- tai .x-tiedostomuodossa. 3ds Maxissa on valittavana ainoastaan .fbx-tiedostomuoto, mutta joillakin muilla mallinnusohjelmilla saa tuotua myös .x-tiedostomuotoisia 3d-malleja. .X-tiedostomuoto tukee myös suoraan Microsoftin tekemää DirectX-rajapintaa. (Reed 2011, 202.) Mikäli haluaa, että tekstuurit tulevat .fbx-tiedoston mukana XNA:han täytyy muistaa laittaa exporttaus asetuksista Embed media kohtaan raksi, jolloin tekstuureja ei tarvitse kopioida enää uudestaan (Exporting FBX Files from 3DS Max 2012). Tällä hetkellä omassa 3ds Maxin versiossa exporttaukseen on käytössä FBX Export Version 2011.1.

Kun Content Pipelinen kautta tuodaan sisältöä XNA:han, niin se kääntää mallin ja tarkistaa, että se on .x-tiedostomuodossa. Vaikka malli olisikin .fbx-muodossa, niin

Content Pipeline kääntää sen automaattisesti .x-muotoon. XNA:ssa on BasicModel -luokka, jossa 3D-malleja voidaan hallita. BasicModel -luokassa tapahtuu 3D-mallin piirtäminen, muuntaminen sekä sen lataaminen. (Reed 2011, 205–206.)

On mahdollista käyttää myös muita tiedostomuotoja, mutta silloin on pakko käyttää erillistä mallien lataajaa apuna, koska XNA:lla ei ole tukea kuin kyseisiin kahteen tiedostomuotoon. Tekstuuritiedostoihin on käytössä enemmän vaihtoehtoisia tiedostomuotoja kuten .png, .jpg, .tga, sekä .bmp, mutta paras formaatti on kuitenkin .dds, joka on jo valmiiksi DirectX-formaatissa. (Heikkinen ym. 2008, 11.) Content Pipeline on tullut XNA Gamestudioon vuonna 2006, ja sen jälkeen ominaisuus on ollut käytössä kaikissa versioissa (Heikkinen ym. 2008, 2). Ilman Content Pipelineä en pystyisi toteuttamaan 3D-hahmon liikkeitä XNA Gamestudiolla paitsi koodaamalla kaikki käsin, joka vie aikaa monia tunteja.

Kun tarvittavat sisällöt on tuotu XNA:han, niin on hyvä buildata projekti. Se tapahtuu menemällä ylävalikossa olevaan build-valikkoon ja painamalla sieltä build solution -painiketta. Buildaamisen avulla saat tietää, onko projektissa virheitä. Mikäli virheitä ei tule, niin Content Pipeline on tunnistanut tuomasi sisällöt ja kääntää ne XNA:n ymmärtämään muotoon. (Reed 2011, 24.)

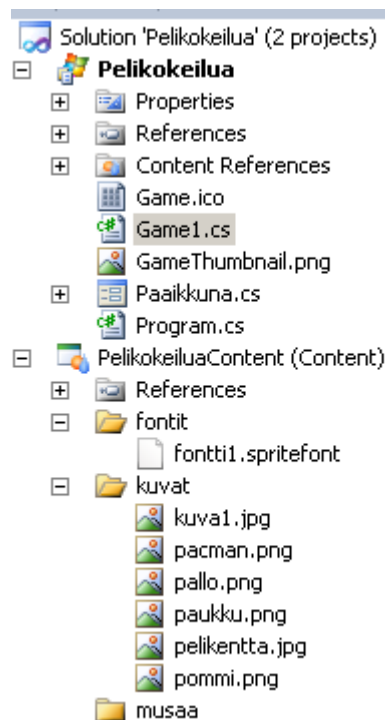
Kuvat sekä kaikki muu sisältö tuodaan peliin LoadContent-osassa. Tähän tarkoitukseen on tehty ContentManager-luokka, jonka avulla kaikki sisältö ladataan XNA:han Content Pipelinein kautta. Kuvassa 5 ja 6 näytetään esimerkki, kuinka 2D -kuva tuodaan peliin. Kuvassa 5 olevaan LoadContentin polkuun kirjoitetaan kuitenkin vain polku ja kuvan nimi ilman tiedostomuotoa, koska XNA osaa automaattisesti käyttää tiedostoa pelkän nimen perusteella. (Reed 2011, 24–25.) Esimerkiksi kuvalla pacman viitataan Pelikokeilua Contentin pacman.png kuvaan.

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // Ladataan kuvat
    this.kuva1 = Content.Load<Texture2D>("kuvat\\pelikentta");
    this.sankari = Content.Load<Texture2D>("kuvat\\pacman");
    this.vihu = Content.Load<Texture2D>("kuvat\\pommi");
    this.lahja = Content.Load<Texture2D>("kuvat\\pommi");
    this.paukku = Content.Load<Texture2D>("kuvat\\paukku");
    this.fontti1 = Content.Load<SpriteFont>("fontit\\fontti1");
}
```

KUVA 5. Kuvan lataus esimerkki

Sisällöt tuodaan kuvan 6 mukaisesti Pelikokeilua Contentiin. Sinne luodaan kansioita, johon tulee erilaisia sisältöjä. On hyvä eritellä kuvat, fontit, 3D-mallit sekä äänet eri kansioihin, koska Content-puu pysyy silloin selkeämpänä. Projektipuussa olevassa ”kuvat” kansiossa olevat kuvat ovat peliprojektin kansiossa Pelikokeilua/PelikokeiluaContent/kuvat.



KUVA 6. Pelikokeilua projektin puu

Microsoft XNA Gamestudiossa pitää hakea kaksi dll-tiedostoa peliprojektin References-puuhun, koska muuten ei saa vaihdettua ContentProcessoria SkinnedModelProcessoriksi. Tiedostot ovat SkinnedModel.dll ja SkinnedModelPipeline.dll. Aluksi kannattaa vain tuoda nämä tiedostot References-puuhun ja vasta sitten siirtää ne oikeille paikoilleen. Sitten kun tiedostot ovat jo peliprojektissa, lisätään SkinnedModel.dll-tiedosto References-puuhun uudestaan. SkinnedModelPipeline.dll-tiedosto puolestaan laitetaan projektin Contentin referensseihin, koska se käyttää ContentPipelineä. Nämä tiedostot tarvitaan ainoastaan silloin kun 3D-mallissa on käytössä luuranko. Tiedostot saa haettua Internetistä ilmaiseksi. (Brooks 2011.)

4.2 Muut asiat

XNA:ssa käytetään koordinaatistoa, jonka mukaan voidaan liikuttaa erilaisia objekteja ruudulla. X-koordinaatti tarkoittaa vaakasuuntaista akselia ja y-koordinaatti taas pystysuuntaista. 3D-mallit käyttävät XNA:ssa x,y sekä z-koordinaatteja, toisin kuin 2D-objektit, jotka käyttävät vain x ja y -koordinaatteja. Z-koordinaatti tarkoittaa x:n ja y:n lävistävää akselia. Koordinaatistojärjestelmässä on käytössä sekä vasen että oikeakätinen systeemi, mutta XNA käyttää oikeakätistä. (Reed 2011, 171–173.) Oikeakätisessä systeemissä x-koordinaatti on positiivinen mennessään oikealle ja y-koordinaatti on positiivinen mennessään ylöspäin. Z-koordinaatti puolestaan on positiivinen tullessaan suoraan kohti. (Reed 2011, 473.) Pisteiden keskikohtaa sanotaan origoksi (James 2010, 10). 3D-hahmo kannattaa rakentaa aina keskelle koordinaatistoa eli origoon 3ds Maxissa, koska tuodessasi hahmon XNA:han malli on oletusarvoisesti juuri sillä kohdalla, missä se oli 3ds Maxissakin (Nieminen 2009, 3).

XNA:ssa on mahdollista rakentaa 3D-malleja itsekin, mutta kaikki vertexit eli pisteet, joista malli koostuu pitää tehdä koodaamalla, ja se on hirveän vaivalloista. Tämän takia kannattaa käyttää jotain ulkopuolista mallinnusohjelmaa tuodakseen 3D-mallin XNA:han. Erilaisia mallinnusohjelmia löytyy paljonkin, joten valinnanvaraa löytyy ilmaisesta aina tosi kalliisiin. (Reed 2011, 201.) Tietokoneella tulee olla asennettuna DirectX 9 tai uudempi versio, jotta XNA pyörii siinä (Reed 2011, 8).

XNA:lla voi tehdä myös sovelluksia Windows Phonelle, mutta ennen kuin niitä voi ruveta tekemään, niin tarvitsee ladata Windows Phone Software Development Kit 7.1.

Tähän on myös tarjolla päivitys, 7.1.1. XNA 4.0:ssa voi tehdä sovelluksia tällä hetkellä Windows 7, 7,5 sekä Windows 8 puhelimille.

Windows 7 -käyttöjärjestelmää käyttäviin puhelimiin tarjottiin paljon paremmat mahdollisuudet tehdä sovelluksia kuin aikaisemmille Windows-puhelimille, ja tämän takia arveltiin, että Microsoftin kurssi lähtisi nousuun. Muun muassa parempi kosketusnäyttö ja isompi, jopa 480 x 800 tuuman näyttö useimmissa puhelimissa takaavat hyvät mahdollisuudet markkinoilla. Visual Studio tarjoaa virtuaalisen Windows-puhelimen emulaattorin, jonka avulla voi testata sovelluksia, ennen kuin siirtää ne oikeaan puhelimeen. Näin ollen sovellusten kehittäminen helpottuu ja nopeutuu. (Reed 2011, 365.)

Tavallisissa Windows-peleissä fps (frames per seconds), eli kuinka monta kuvaa näytetään sekunnissa, on 60. Windows Phone -peleissä taas fps on oletuksena 30, koska puhelimen laskentatehot eivät ole yhtä hyvät kuin tietokoneiden. Muita eroavaisuuksia ovat esimerkiksi, että Windows Phone -peli tukee erilaisia ääniasetuksia kuin Windows-peli. (Reed 2011, 372–373.) Windows Phone -pelissä on myös erilaiset kontrolit kuin Windows-pelissä. Esimerkiksi puhelinta kallistamalla kuva siirtyy ylösalaisin ja näppäimistöä ei ole, koska kaikki hahmon liikuttamiset ja muut toiminnot tapahtuvat kosketusnäyttöä koskettamalla. Koodissa siis tulee suurimmat eroavaisuudet juuri tässä asiassa. Joissakin peleissä hahmo liikkuu tiltaamalla (kääntelemällä) puhelinta, jolloin tarvitaan oliota nimeltä accelerometer (kiihdytinmittari), jonka avulla hahmon liikkeitä kontrolloidaan. Jotta accelerometer toimii, tarvitaan peliprojektin Referensseihin Microsoft.Devices.Sensors-kirjasto, joka löytyy .NET-välilehdeltä referenssien lisäyskohdassa. Windows puhelimen touchpanelia voidaan käyttää TouchPanel Classin avulla, josta voidaan hallita mitä näytöllä tapahtuu touchpanelia käyttäessä. (Reed 2011, 378–395.)

Kaikki Windows 7 -puhelimet tukevat 480 x 800 resoluutiota, joka tunnetaan paremmin Wide Video Graphics Array (WVGA) nimellä. Lisäksi Microsoftilta on tulossa tuki 320 x 480 resoluutiolle, joka on Half-size video graphics array (HVGA). (Williams & Clingerman 2011, 26.)

Windows 7 -puhelimet eivät tue custom shadereitä, koska puhelimien laitteistossa on tiettyjä rajoituksia. Custom shaderit tarkoittavat ohjelmaan laitettavia efektejä, joita

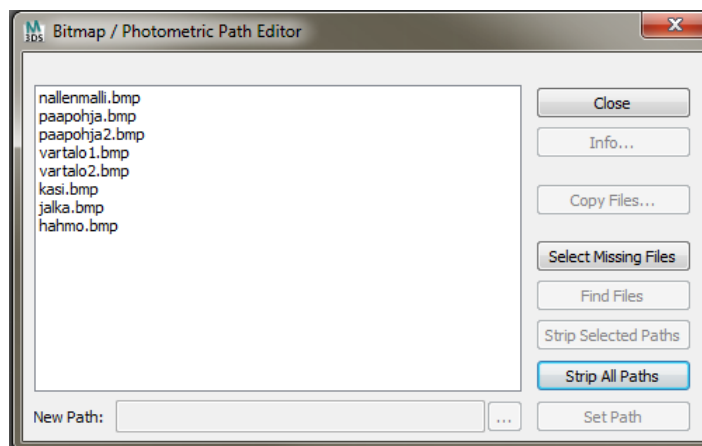
voi muokata itse. Näiden tilalle on laitettu 5 valmista efektiä, jotka saavat helposti kutsuttua ohjelmaan. (Williams & Clingerman 2011, 267.)

5 ONGELMATILANTEITA

Tässä luvussa käydään läpi ongelma- sekä virhetilanteita, joita matkan varrella on tullut. Virheitä on tapahtunut sekä 3ds Maxissa että XNA:ssa, mutta mielestäni enemmän kuitenkin XNA:n puolella. Nämä tilanteet perustuvat omakohtaisiin kokemuksiini.

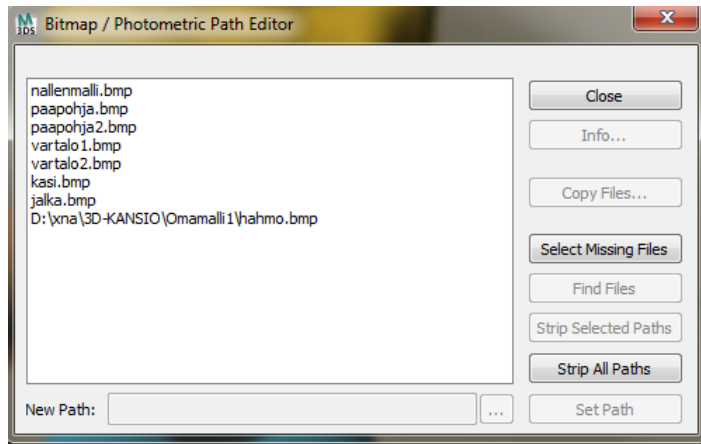
5.1 3ds Max

Valmiin 3D-mallin viemisessä XNA Gamestudioon tuli ongelmia moneen otteeseen. Esimerkiksi XNA Gamestudio ei osaa löytää 3ds Maxissa käytettyjen tekstuurikuvien polkuja oikein, mikäli niitä ei poista ennen exporttausta eli viemistä 3D-mallista. Tämä kyseinen ongelma korjaantuu kun valitsee 3ds Maxin oikealla reunalla olevasta valikosta utilities-työkalun ja painaa sen jälkeen more-painiketta. Valikosta valitaan tämän jälkeen Bitmap/Photometric Paths ja klikataan ok-nappia. Tämän jälkeen otetaan valittu merkki pois Include Material Editor -kohdasta, sillä muuten koko ohjelma kaatuu. Kukaan ei kuitenkaan tiedä tarkkaa syytä miksi näin tapahtuu. Ilmeisesti ohjelma käy liian raskaaksi, jos materiaalieditorin ottaa päälle. Ohjelma kuitenkin kysyy kaatuessaan, yritetäänkö scene ottaa talteen, mutta kun yrittää avata sitä tiedostoa, niin tekstuurit eivät tule mukaan. Mikäli ohjelma ei kaadu, niin painetaan Edit Resources -nappia, jonka jälkeen avautuu valikko, jossa painetaan Strip All Paths -nappia, jolloin kuvien polut lyhenevät siten, että jäljelle jää vain tiedoston nimi ja tiedostomuoto kuten kuvassa 7 näkyy, esimerkiksi jalka.bmp.



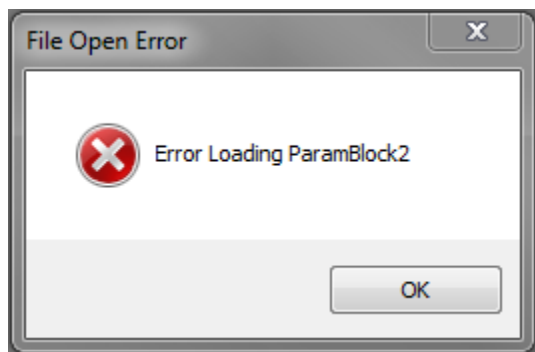
KUVA 7. Tekstuurien polut lyhykäisyydessään

Kuvassa 8 taas näkyy hahmo.bmp-tekstuuritiedoston koko polun. 3ds Max on huono ohjelma hakemaan oikein poluissa olevia tiedostoja. Polkujen lyhentämisen jälkeen tekstuurien pitäisi toimia XNA Gamestudiossa ilman ongelmia.



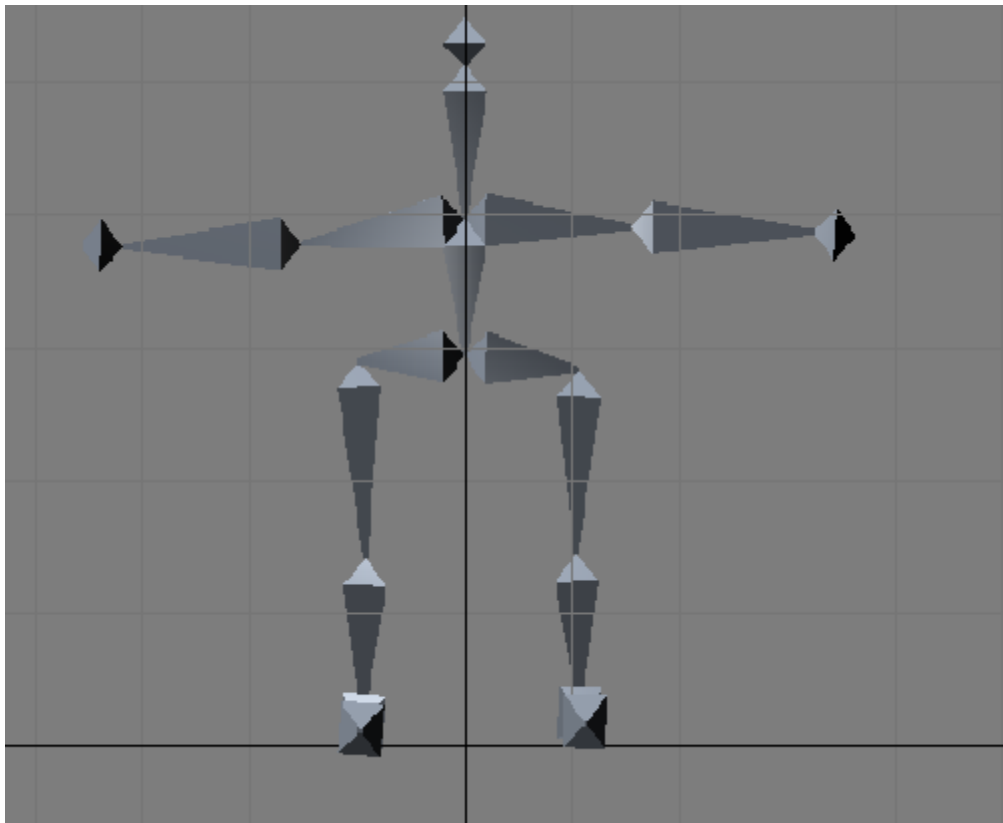
KUVA 8. Tekstuurien polku kokonaisuudessaan

Tein 3ds Maxissa harjoitusmallin ja tallensin sen normaaliin Maxin tallennusmuotoon. Seuraavalla kerralla kun avasin kyseisen tiedoston ohjelmaan, tuli kuvan 9 mukainen virhe. Virheen seurauksena melkein puolet hahmosta oli lähtenyt bittiavaruuteen eikä sitä saanut palautettua mitenkään. Minulla ei ollut hajuakaan, mitä virhe tarkoitti eikä googlettamallaakaan ongelma selvinnyt, joten jouduin tekemään koko mallin alusta uudelleen.



KUVA 9. 3ds Maxin latausvirhe

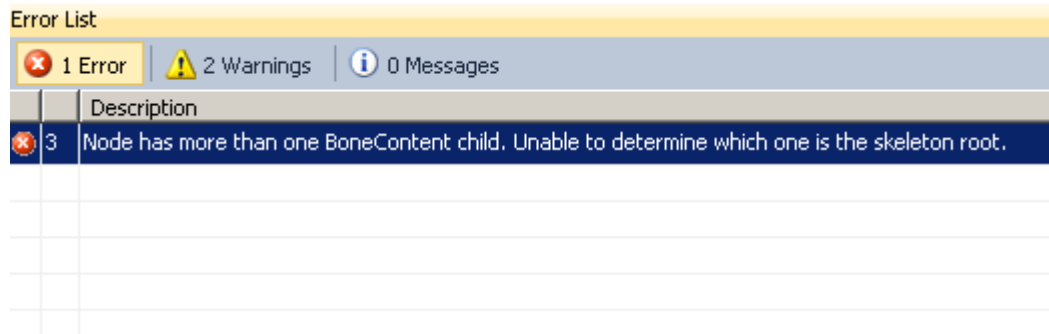
Ennen varsinaisen 3d-mallin tekemistä tein harjoitukseksi yksinkertaisemman mallin. Tähänkin malliin tein luurangon kuvan 10 mukaisesti, ja animoin erilaisia liikkeitä. Luurangon tekeminen lähti liikkeelle siten, että laitoin hahmon näkymään *Wireframe* valinnassa, joten siitä näkyi vaan ääriviivat. Tällä tavalla luurangon saa tehtyä oikeankokoiseksi ja sellaiseksi, että se mahtuu hahmon sisään. Sitten valitsin kuvakulmaksi etunäkymän ja rupesin tekemään luurankoa. Aloin luomaan bones-kappaleita päästä alaspäin, jolloin ylimmästä luusta tuli parent-luu, joka liikuttaa koko luurankoa. Kun luuranko oli valmis, niin siirsin sen kokonaisuena oikeaan kohtaan hahmoa.



KUVA 10. Valmis luuranko ensimmäisessä versiossa

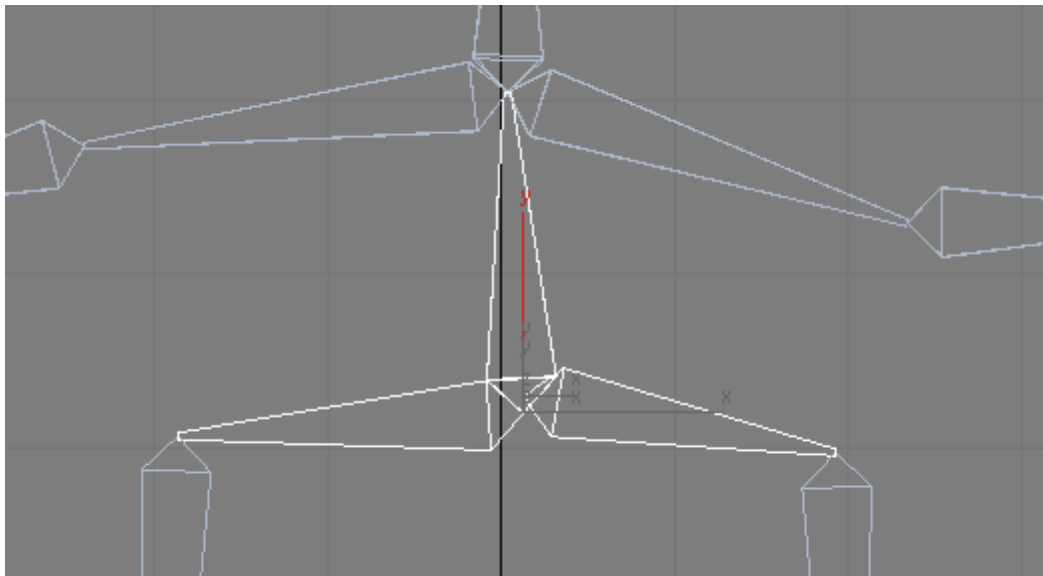
5.2 XNA Gamestudio

Tuotuaani XNA:han 3D-mallin, jossa oli sisällä luuranko, tapahtui kuvan 11 mukainen virhe useaan otteeseen. Virhe kertoo, että luurangon solmussa on enemmän kuin yksi child-luu, joten se ei osaa määrittellä mikä niistä on luurangon juuriluu. Kokeilin etsiä ratkaisua Internetistä, mutta en löytänyt mitään korjausvinkkejä.



KUVA 11. Virhetilanne XNA:ssa

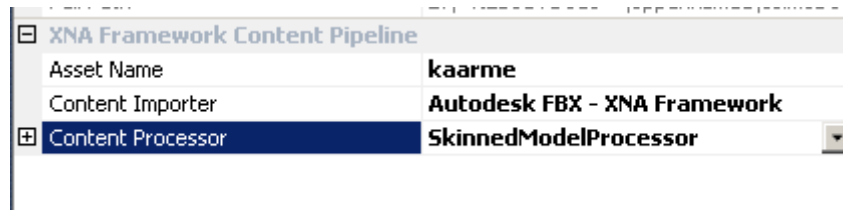
Kuvassa 12 on kuvattu hahmon luurangon luut, joiden takia virheilmoitus tuli. 3ds Maxissa ei vielä tullut mitään virhettä kun exporttasin tiedoston, mutta kyseinen virhe tuli XNA:ssa heti. Tämän jälkeen rupesin tutkimaan, minkä takia virhe tuli ja huomasin, että kaikki luut eivät olleet yhteydessä toisiinsa. Sitten tein koko luurangon uudelleen ja ongelma poistui.



KUVA 12. Luurangon luissa oleva vika

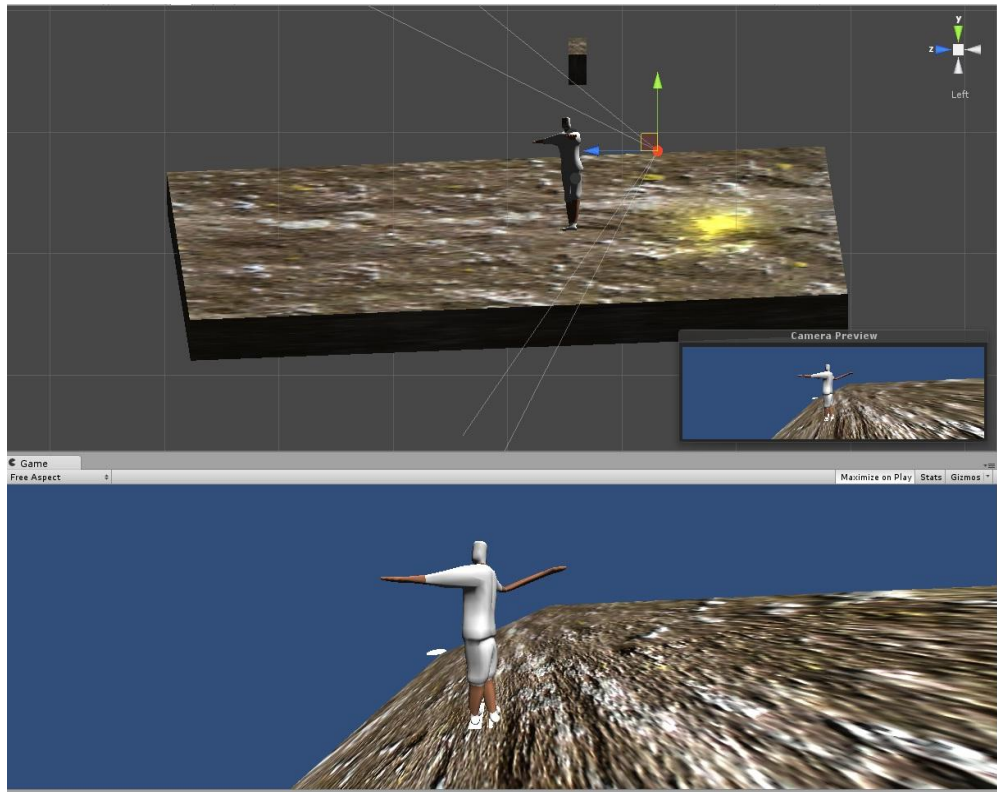
Mikäli 3D-mallin haluaa saada toimimaan XNA:ssa, niin .fbx-tiedoston Content Processor valinta pitää olla SkinnedModelProcessor, kuvan 13 mukaisesti. Oletuksena valinta on Model – XNA Framework. Tämä asia tuli esille tehdessäni 3D-mallia, koska en saanut aluksi mitenkään sitä toimimaan. Sitten katsoin toisesta mallista sen asetuksia ja huomasin, että siinä Content Processor valinta oli eri. Tämän jälkeen sain

mallin pyörimään heti ruudulle. Tämä valinta pitää tehdä ainoastaan, jos 3D-mallissa on käytössä luuranko.



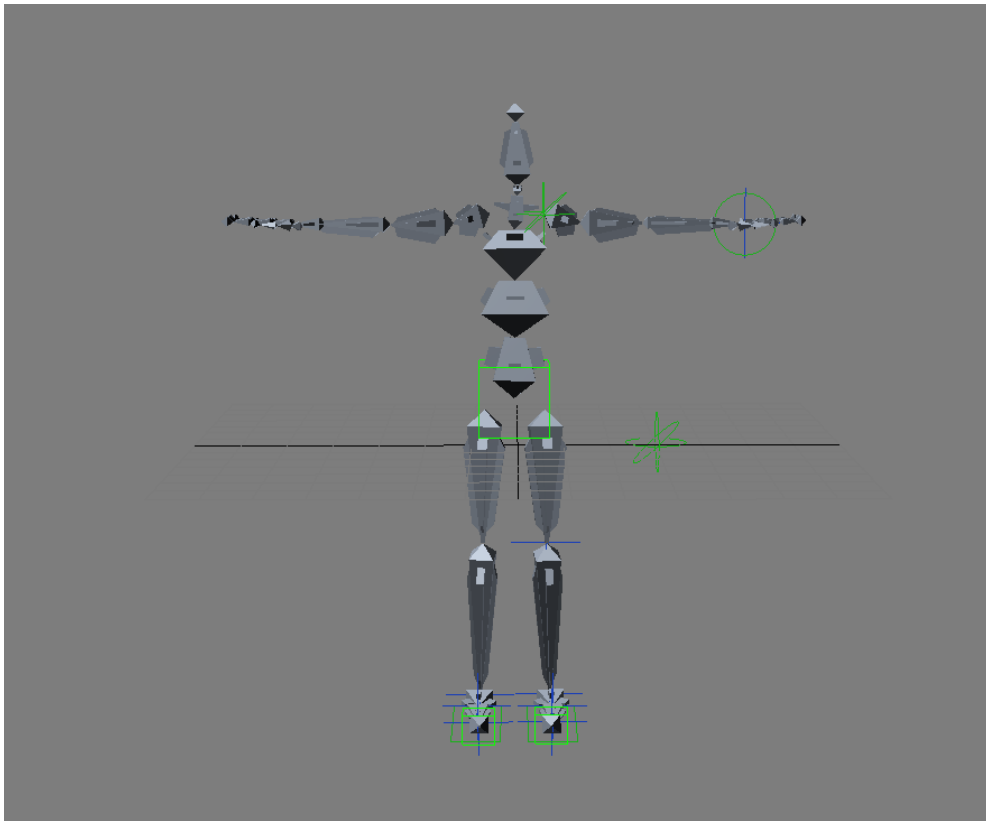
KUVA 13. Content Processor -valikko

XNA:ssa on melkoisen paljon erilaisia rajoituksia luurangon tuomisen suhteen. Esimerkiksi Bibed-luurankoa ei voi tuoda ollenkaan XNA:han koska siinä on reilu 100 luuta ja XNA:ssa luiden määrä rajoittuu hieman alle sataan luuhun. Kokeilin kuitenkin tuoda Bibed-luurankoa XNA:han, mutta eihän se toiminut siinä. Myöskään luurangon osat eivät saaneet olla irtonaisia vaan kaikkien luiden piti olla kiinni toisissaan. Kuvassa 11 oleva virhetilanne tuli tämän takia useaan kertaan. Tätä ongelmaa ei tullut Unity3D-peliohjelmointiympäristössä lainkaan. En tiedä tarkalleen, minkälaisia rajoituksia Unityssä on, mutta sama luuranko kuitenkin toimi siinä, mutta ei XNA:ssa. Tässä suhteessa XNA on melko huono pelimoottori. Kuvassa 14 on todiste siitä, että sama hahmo samalla luurangolla saatiin pyörimään Unityssä ilman minkäänlaisia ongelmia.



KUVA 14. Unity3D

3ds Maxilla pystyi tekemään minkälaisen luurangon tahansa, mutta ongelmat alkoivat kun toin hahmon XNA:han. Mistä minä olisin voinut tietää, minkälaisia rajoituksia pelimoottorissa on, koska missään ei lukenut tarkkoja tietoja niistä. Tein siis hienon luurangon täysin turhaan, koska sitä ei voinut käyttää. Kuvassa 15 on luuranko, joka on tehty viimeisen päälle hyvin, mutta XNA:n rajoitusten takia sitä ei voinut käyttää. Kuvan 15 mukainen luuranko on kuvan 14 mukaisessa Unity3D-pelimoottorissa kuitenkin toimiva.



KUVA 15. Luuranko joka ei toiminut XNA:ssa

6 CASE

Tässä luvussa kirjoitan mallintamisen vaiheita omasta kokemuksestani mallin suunnittelusta aina mallin valmiiksi tulemiseen asti. Lisäksi luon hahmolle luurangon ja animoin sille tietynlaisia liikkeitä, jotta voin todentaa XNA:ssa, että hahmo pyörii siellä oikein. Tämä on eräänlainen opas, kuinka saadaan tehtyä hahmo, jota voi käyttää XNA-pelimoottorissa.

6.1 Hahmon suunnittelu

Hahmon tekeminen lähti liikkeelle sen suunnittelusta. Tarkoituksena oli tehdä hahmo, joka näyttää ihmiseltä, mutta ei olisi niin tarkasti piirretty. Lisäksi Internetiä selailemalla sain selville, että 3D-peleihin tarkoitettut hahmot tulisi olla mahdollisimman vähän polygoneja käyttäviä, mutta niiden tulisi näyttää silti aidon näköisiltä. Näin ollen etsin tietoa low poly -mallintamisesta ja löysin Internetistä hyviä ohjeita sellaisen mallin tekemiseen. Tämän jälkeen aloitin mallintamisen ja heti kun rupesin tekemään mallia, huomasin että minulla ei ollut hajuakaan kuinka isoiksi teen hahmon eri

osat. Minulla ei ollut tässä vaiheessa tehty hahmosta minkäänlaisia piirustuksia, koska en ole mielestäni hyvä piirtämään.

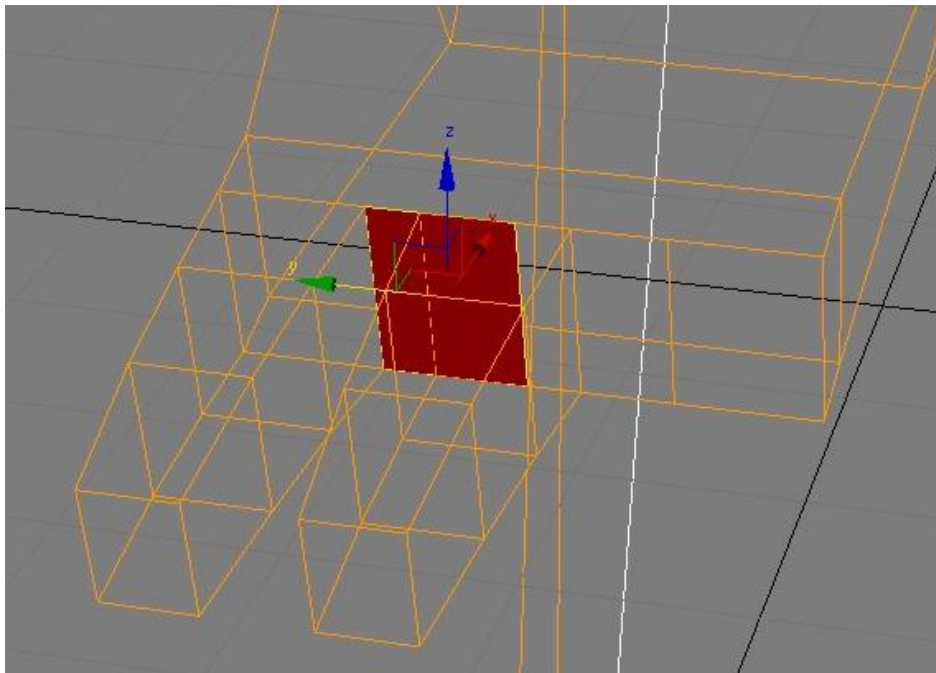
Sitten otin Internetistä referenssikuvaksi Da Vincin Vitruvialaisen miehen kuvan edestäpäin. Tein 3ds Max -ohjelmalla yhden plane-objektin ja liitin referenssikuvan sen pinnalle bittikarttana. Siirsin planen oikealle kohdalla gridin taakse siten, että se oli keskellä origoa. Tämän jälkeen pystyin mallintamaan hahmosta oikean kokoisen. Tärkeä huomio hahmon mallintamisessa oli myös, että hahmo kannattaa mallintaa keskelle origoa, koska siirtovaiheessa XNA:han malli on siinä kohdassa missä se on silloin kun malli exportataan 3ds Maxissa. Suunnittelu vaiheessa kannattaa myös päättää, mille pelialustalle ja millä peliohjelmointimoottorilla peliä on tekemässä, koska eri pelimoottoreiden välillä on erilaisia rajoituksia.

Sitten kun malli oli valmis ja siirsin sen XNA:han, huomasin, että mallin mittakaava oli silti virheellinen, koska hahmo näytti tosi suurelta. Onneksi 3ds Maxissa pystyy skaalaamaan mallia kokonaisuudessaan, joten selvisin pelkällä säikähdyksellä tästä ongelmasta. Myös XNA:ssa on mahdollista muokata hahmon skaalaa, mutta se on huomattavasti vaikeampaa kuin 3ds Maxissa, koska siinä tulee ottaa huomioon koordinaatiston kaikki akselit. Jos jotain akselia muuttaa, niin se vaikuttaa myös muihin, joten pienet muutokset voivat muuttaa hahmon kokoa radikaalisti. Tosin skaalaus pitää tehdä ennen luurangon luomista, sillä se ei enää mahdu hahmon sisälle, jos muokkaa hahmon kokoa silloin, kun luuranko on jo sen sisällä.

6.2 Hahmon mallintaminen

Kun suunnittelu oli valmis, niin rupesin mallintamaan hahmoa. Aloitin sen luomisen tekemällä yhden laatikon. Tarkoitus oli luoda hahmo yhdestä osasta erilaisten modifierien avulla sekä Extrude- ja Chamfer-työkalujen avulla. Lisäksi hahmolle luotiin muotoja yhdistelemällä vertexejä toisiinsa. Laatikon tekemisen jälkeen lisäsin leveys segmenttiluvun kahteen ja tein laatikosta editable polygonin. Sitten poistin toisen puoliskon laatikosta ja otin modifierlistasta Symmetry Modifierin, jonka avulla minun ei tarvitse tehdä kuin toinen puoli mallista, koska toinen puoli tulee sen avulla automaattisesti samanlaiseksi kuin toinen puoli. Uusia polygoneja loin valitsemalla tiettyjä edgejä ja yhdistämällä ne connect-toiminnolla. Seuraavaksi tein kädet valitsemalla laatikon ylimmän polygonin ja vetämällä Bevel -työkalulla sitä ulospäin laatikosta.

Tätä toistin pari kertaa, jonka jälkeen laatikolla oli kädet. Kämmenen loin Extrude-toimintoa käyttämällä, jonka jälkeen jaoin sen neljään osaan, joihin aloin rakentaa sormia. Sormet loin Bevel-toimintoa käyttämällä samalla tavalla kuin kädetkin. Tein vain yhden sormen, jonka jälkeen kopioin sen 4 kertaa, että sain kaikki sormet tehtyä. Sen jälkeen piti poistaa kämmenessä olevat ylimääräiset polygonit, koska sormet piti yhdistää oikeisiin kohtiin kämmenessä. Kuvassa 16 on esimerkki poistettavasta polygonista. Seuraavaksi valitsin toisen sormen vertexejä ja vedin ne Target Weld – työkalulla oikeisiin kohtiin. Nämä toimenpiteet tein kaikille sormille



KUVA 16. Sormien luominen

Jalkojen luominen tapahtui käyttämällä tuttuja Extrude- ja Bevel-työkaluja. Tämän jälkeen rupesin tekemään lihaksia hahmolle luomalla siihen lisää polygoneja ja muokkaamalla niiden kokoa ja muotoa. Myös yksittäisiä vertexejä liikuttamalla sain luotua hahmolle näyttävyyttä. Siinä meni melko kauan aikaa, koska hahmon lihaksiston näyttäminen aidolta oli hankalaa ja en ollut tyytyväinen niiden muotoon heti. 3ds Maxissa on mahdollisuus kumota tehty muutos niin kuin kaikissa muissakin ohjelmissa ja oletuksena siinä voi kumota 20 kertaa tehdyn muutoksen. Lisäsin asetuksista kumoamisen määrän 50:neen, koska tässä vaiheessa jouduin tekemään moneen kertaan uudestaan asioita. Tämän voi säätää menemällä päävalikon Customize-kohtaan ja valitsemalla Preferences-valikon. General-välilehden vasemmalla ylhäällä on valinta, missä tätä asetusta voi muuttaa.

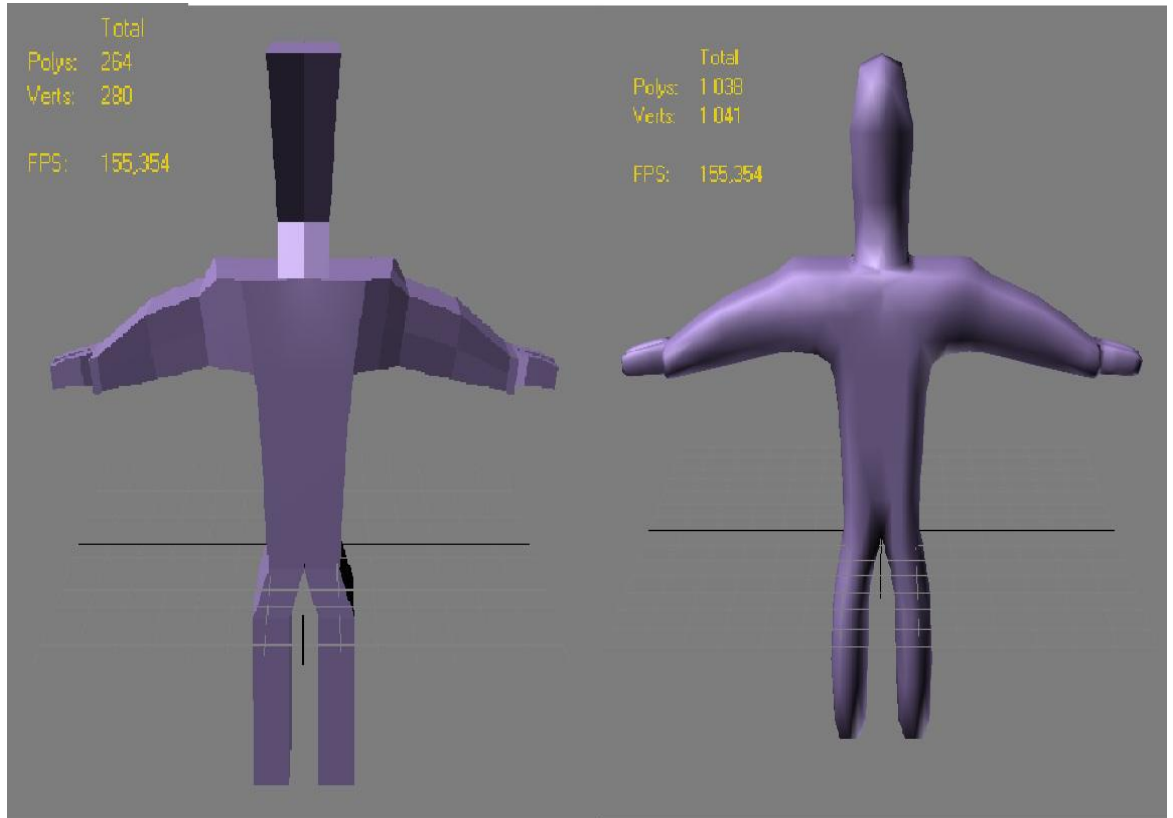
Hahmo näytti tässä vaiheessa vielä melko laatikkomaiselta, mutta sain sen näyttämään paremmalta Ring-työkalun avulla. Ring-työkalu toimii valitsemalla edgen, jonka jälkeen painetaan Ring-työkalua. Ring-työkalu valitsee automaattisesti samassa linjassa kulkevat edget, jonka jälkeen yhdistin ne Connect-toiminnolla. Tuloksena oli, että jokainen linjassa oleva polygon jakautui kahdeksi polygoniksi, jolloin myös vertexien määrä kasvoi. Tällä tavalla sain pyöristeltyä paremmin hahmon muotoja. Hahmon vaatetuskin on tehty siirtelemällä yksittäisiä vertexejä siten, että se näyttäisi siltä, että hahmolla on paita ja housut.

Hahmon pään luominen lähti käyntiin valitsemalla kaksi hartian oikean sivun reunimmaista polygonia. Sitten niiden keskelle luotiin Inset-työkalulla uusia polygoneja, jonka jälkeen poistettiin taas kaksi oikean reunimmaista polygonia. Sitten näistä kahdesta jäljelle jääneestä polygonista vedettiin Extrude-työkalulla hahmolle kaula. Sitten kaulasta vedettiin hahmolle pää. Päähen tehtiin muodot samalla tavalla kuin aikaisemminkin eli lisäämällä polygoneja ja siirtelemällä niitä.

Hahmon mallintaminen rupesi olemaan valmista muutamia korjauksia lukuun ottamatta. Tarkastin STL Check Modifierillä löytyykö hahmosta virheitä, kuten päällekkäin meneviä edgejä tai kohtia, missä polygon jää auki. Tarkastuksen tuloksena löytyi muutama virhe ja rupesin korjailemaan niitä *wire frame* -näkyvässä, koska virheet näkyvät siinä punaisella värillä. Siirtelin muutamia vertexejä eri paikkoihin, jolloin punaiset alueet poistuivat. Tämän jälkeen tein tarkastuksen uudestaan ja virheitä ei enää löytynyt.

Tässä vaiheessa hahmo oli melkein valmis, puuttui vaan hienosäätö. Lisäsin hahmolle Meshsmooth Modifierin, joka pyöristää kaikki hahmon muodot, jotta se näyttäisi vielä enemmän aidolta. Tämä Modifieri kuitenkin lisää melko paljon hahmon polygonien määrää, kuten kuvassa 17 näkyy. Tässä mallissa polygonien määrä nelinkertaistui käyttämällä Meshsmooth Modifieriä. Tämän jälkeen halusin vähentää polygonien määrää, jotta hahmo pyörisi parhaalla mahdollisella tavalla XNA:ssa. Lisäsin hahmoon MultiRes ja Optimize Modifierit ja rupesin säätämään niiden asetuksia. Aluksi vähensin MultiRes Modifierin asetuksista vertexien määrää 70:neen prosenttiin, jolloin vertexien, polygonien ja edgejen määrä tippui selvästi alemmaksi. Sitten käytin Optimize Modifieriä, jonka jälkeen niiden määrä tippui vielä lisää. Yhteensä näillä kahdella Modifierilla polygonien, vertexien sekä edgejen määrä tippui noin tuhannella

ilman, että hahmon rakenne olisi näyttänyt huonommalta ennen niiden käyttämistä. Hahmon muodot siis pysyivät suurin piirtein samana vaikka ”turhat” polygonit, verte-
xit ja edget poistettiin. Kuvasta 17 huomaa myös, että hahmossa on vieläkin paranta-
misen varaa, mutta päätin kuitenkin jättää hahmon mallintamisen tällaiselle tasolle.



KUVA 17. MeshSmooth Modifierin käyttäminen

6.3 Hahmon teksturointi

Hahmon teksturointiin käytin yksinkertaisia tekstuureja ja bitmappeja. Ihon tekstuu-
reihin loin Slate Material Editorilla standard-materiaalin, johon lisäsin bittikartan,
jossa oli vain ruskeaa väriä. Kuvan resoluutioksi otin 128x128 ja tiedostomuodoksi
valitsin .bmp:n, koska se on mielestäni paras tiedostomuoto bittikartoissa, koska tal-
lentaessa se pysyy täysin samanlaisena kuin se on tehty. Oikeisiin kohtiin sain teks-
tuurin valitsemalla tiettyjä polygoneja hahmosta ja liittämällä tekstuurin vain niihin,
mitkä olivat valittuina. Samalla tapaa tein hahmolle vaatteet sekä kypärän, mutta nii-
hin käytin valmiita materiaaleja editorista. Hahmon naaman teksturiin tein bittikartan
samoilla asetuksilla kuin ihoonkin. Valitsin puolet hahmon päästä ja liitin tekstuurin

siihen. Bittikartan suuntauksen valitsin z-akselin mukaan ja otin raksin pois Real-World Map size -valinnasta. Bittikartan saa tehtyä tietyn muotoiseksi hahmon pinnalle ja tähän valitsin Planar vaihtoehdon. Bittikarttaa voi liikutella haluamalleen kohdalle ja sitä voi suurentaa tai pienentää halutessaan. Näitä kaikkia valintoja ei kuitenkaan saa tehtyä jos liittää tekstuurin tavallisesti hahmoon vaan siihen täytyy lisätä Modifier-listasta UVW Mapping Modifier.

6.4 Luurangon luominen

Seuraavaksi oli aika luoda hahmolle luuranko, jotta saisin animoitua hahmolle liikkeitä. Valitsin hahmon ja klikkasin hiiren kakkosnappia ja valitsin Object Properties -valikon. Laitoin Display Properties kohdasta See-Through-valinnan päälle, jolloin hahmon läpi pystyi näkemään. Tällä tavalla luurangon luominen on helpompaa, koska saan tehtyä luista suoraan oikean kokoisia. Luurangon tekeminen lähti liikkeelle luomalla pään luun bones-luukappaleella, joka oli samalla luurangon juuriluu. Juuriluuta liikuttamalla koko muu luuranko liikkuu sen mukana. Sitten loin kaulan ja siitä lähtevät käsien luut. Käsissä oli hauisluu, käsivarsiluu sekä ranne. Sormien luiden tekeminen oli todella työlästä, koska luut tulivat moneen otteeseen ulos hahmon pinnasta. Selkärankaan tein kaksi luuta, minkä jälkeen tein lantion luun, joihin kiinnitin jalat. Molemmissa jaloissa oli reisiluu, sääriluu, nilkkaluu, jalkapöytä sekä jalkapöydänpää. Tässä vaiheessa nimesin kaikki luut oikeilla nimillä, koska skinning-vaiheessa se helpottaa melkoisen paljon luiden yhdistämistä vartalon osiin.

Kun luut olivat valmiita, rupesin hieman hienosäätämään niiden kokoa ja muotoja. Tarkoituksena oli tehdä luista niin isoja kuin mahdollista siten, etteivät ne kuitenkaan tule ulos hahmosta. Tällä tavalla luuranko toimii paremmin ja liikkeet näyttävät aidoimmilta. Tein yksittäisiin luihin eviä (fins) sivuille, eteen sekä taakse, jotka ovat eräänlaisia luiden ulokkeita. Lisäksi luista voi säätää, mistä kohdasta luu rupeaa kaaventumaan.

6.5 Skinning

Skinning-vaiheessa luuranko oli tarkoitus liittää hahmoon. Jokaisen yksittäisen luun piti olla kiinnitettyä johonkin hahmon osaan. Tässä vaiheessa nimesin hahmon osat ja tarkastin listauksesta, että kaikilla oli oikeat nimet. Tarkastin myös luiden nimet

kohdalleen. Skinnaus lähti käyntiin valitsemalla koko hahmon ja valitsemalla siihen Skin Modifierin. Tämän jälkeen luuranko oli liitettyä hahmoon, mutta en ollut vielä tyytyväinen raajojen liikkeisiin testatessani liikeratoja, koska luut lähtivät irtoilemaan kehosta ja ne eivät liikkuneet tarpeeksi aidonnäköisesti.

Tässä vaiheessa eri luiden painoarvoja piti muuttaa, koska jotkut luut vaikuttivat liikaa sinne, missä niiden ei tarvinnut vaikuttaa. Tämä tarkoittaa suomeksi, että kun liikuttaa esimerkiksi hahmon jalkaa niin lantio ja kroppa liikkuvat sen mukana liian paljon. Painoarvojen tulee olla 0:n ja 1:sen välillä, mutta tasan 0 ei kuitenkaan saa olla. Painoarvoja saa muuteltua valitsemalla koko hahmon, jonka jälkeen mennään Modify-välilehdelle ja valitaan Skin Modifier. Sitten valitaan listasta minkä luun painoarvoa halutaan muuttaa ja painetaan Edit Envelopes -nappia, joka on ihan asetuksien yläosassa. Sitten valitaan vertexejä ja laitetaan niille oikeat painoarvot Weight Properties -asetuksen Absolute Effects -kohdassa. Smooth + highlight -näkyvässä näkyy eri vertexien painoarvot kyseiseen luuhun. Mikäli painoarvo on 1, niin väri on punainen, jos väri on keltainen, niin arvo on noin 0,5 ja mikäli väri on sininen tai tavallisen värinen, niin arvo on 0. Tämä kuitenkin soti aikaisemmin sanomaani lausetta ”painoarvon pitää olla 0:n ja 1:n välillä” vastaan, mutta se johtuu siitä, että vertexit jotka olivat painoarvoiltaan 0 tähän luuhun nähden, on johonkin muuhun luuhun nähden arvoltaan 0–1. Painoarvojen laittaminen on hankalaa puuhaa ja mielestäni kokeilemalla eri arvoja tietyille vertexeille on paras vaihtoehto saada luiden liikkeet aidonnäköisiksi.

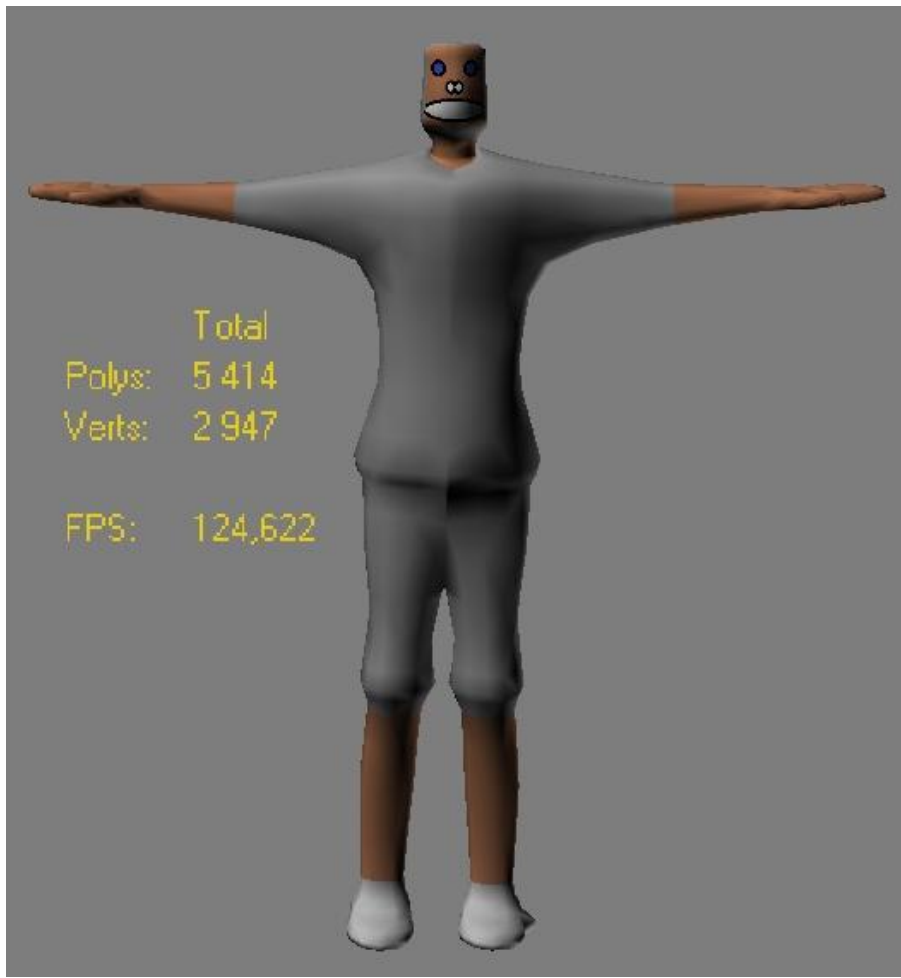
6.6 Animaation luominen

Animaation tekeminen oli aika pieni osa tätä projektia, sillä minulla ei ollut tarkoitus tehdä mitään monimutkaista animaatiota. Tarkoitus oli saada tehtyä vain jotain perus liikkeitä hahmolle, koska XNA:ssa tulee virhetilanne, mikäli hahmolla ei ole lainkaan animaatiota. Animaation tekeminen lähti liikkeelle piilottamalla kaikki muut osat paitsi luut hahmosta. Sen jälkeen menin timelinelle ja laitoin autokey-valinnan päälle. Seuraavaksi siirsin jalkoja ja käsiä tiettyyn asentoon, jonka jälkeen timelinelle syntyi keyframe tiettyyn kohtaan. Sitten siirsin aikapalkkia eteenpäin ja siirsin taas vähän jalkoja ja käsiä eri asentoon. Jälleen syntyi uusi keyframe aikajanalle. Tällä tavalla jatkoin siihen asti kun sain liikkeet kokonaisuudessaan valmiiksi. Animaatiota voi katsoa painamalla play-näppäintä ja aikajanalle pääsee myös liikkumaan framejen välillä.

Animaation tekeminen helpottuu, mikäli luiden liikuttamiseen tehdään erilaisia kontrollikappaleita joiden avulla liikutetaan tiettyjä luita. Näitä kontrollikappaleita liikuttamalla ei tarvitse valita erikseen pelkkiä luita, joten luiden vahingossa liikuttaminen vähenee huomattavasti. Kontrollikappaleiden avulla animaattorit tekevät animaatioita eikä heidän tarvitse ymmärtää luurangosta mitään. Kontrollikappaleet liitetään luihin käyttämällä IK Solvereita, joita löytyy neljä eri vaihtoehtoa. Loin jaloille yhden kontrollikappaleet, joiden avulla jalka liikkuu tietyllä tavalla. Kontrollikappaleet tein luomalla apupalikoita, joita käytettiin osien liikuttamiseen. Lisäksi tarvittiin IK Solvereita, joiden avulla saatiin yhdistettyä tiettyjä luita kontrollikappaleisiin. Kontrollikappaleiden tekeminen oli lähes hankalin vaihe koko projektissa.

Yhteen 3ds Maxin skeneen voi liittää vain yhden animaation, koska useammalle animaatiolle ei ole tukea. Mikäli haluaa luoda hahmolle useita animaatioita, niin vaihtoehtona on tehdä eri animaatiot eri tiedostoihin tai laittaa kaikki eri liikkeet samalle timelinelle. Autodeskin MotionBuilder -ohjelmalla saa kuitenkin tehtyä useita animaatioita samassa skenessä, mikäli siihen on tarvetta.

Kuvassa 18 on valmis hahmo, jolle on luotu luuranko. Kuvasta näkee myös polygonien määrän, joka ei ole hirveän suuri. Tämä johtuu siitä, koska päätä ei ole mallinnettu hirveän tarkasti. Päässä on yleensä enemmän polygoneja kuin muussa hahmossa yhteensä, joten se lisää polygonien määrää huomattavasti.



KUVA 18. Valmis hahmo

6.7 Hahmon vieminen XNA:han

Kun kaikki muut vaiheet olivat valmiina, oli viimeisen vaiheen vuoro eli hahmon vieminen XNA:han. Ennen viemistä tallensin hahmon tavalliseen .max-tiedostomuotoon vielä viimeisen kerran. Tein kuitenkin melko usein tallennuksia, koska 3ds Max kaatuu melko helposti ja syytä ei tiedä yleensä. Menin 3ds Maxin päävalikkoon ja painoin export-painiketta. Katsoin, että asetukset olivat oikeat ja exportasin tiedoston .fbx-muotoon.

Tämän jälkeen käynnistin Visual Studion. Loin projektin Contenttiin uuden kansion, johon laitoin kaikki tekstuuritiedostot sekä .fbx-tiedoston. Siirsin myös tiedostot tietokoneella olevaan kansioon, jonka olin juuri luonut. Sitten vaihdoin .fbx-tiedoston Content Processor -valinnan SkinnedModelProcessoriksi, koska tavallinen muoto ei kelpaa jos hahmossa on luuranko. Sitten otin aikaisemmista harjoitustöistä valmiiksi

tekemäni koodin ja liitin sen Game1.cs tiedostoon. Lisäsin koodin LoadContent-osaan .fbx-tiedoston polun, jossa ladataan peliin tulevat tiedostot. Sitten skaalasin mallia hieman ja siirsin sen oikealle paikalle. Seuraavaksi piti testata lähteekö peli käyntiin. Buildasin projektin ja annoin sen rullata loppuun saakka. Virheilmoituksia ei tullut joten oli aika testata peliä. Pelin saa pyörimään painamalla ctrl+F5 nappeja tai mene-mällä päävalikossa olevaan Debug-valikkoon ja painamalla start without debug -painiketta. Peli lähti pyörimään ja hahmoni teki siinä niitä liikkeitä, mitä olin animoinut sille. Testi siis onnistui vaikkakin matkan varrella oli tullut paljon virhetilanteita. Seuraavaksi oli tarkoitus tehdä malliin hieman parempi luuranko ja testata sitä.

Laitoin myös Microsoftin hahmon samaan aikaan pyörimään ruudulle ja vertasin kuinka paljon paremmin sen liikeradat toimivat verrattuna omaan hahmoon. Oman hahmoni liikuttaessa kättä sekä jalkaa huomasin heti, että kroppa lähti venymään liikkeen mukana, mutta Microsoftin hahmolla kaikki toimi tyylikkäästi ilman mitään repeämiä. Olisi ollut hyvä, jos Microsoftin hahmosta olisi ollut .fbx-tiedoston lisäksi .max-tiedosto, koska olisin saanut enemmän apuja muun muassa luurangon tekemiseen ja skinning-vaiheeseen.

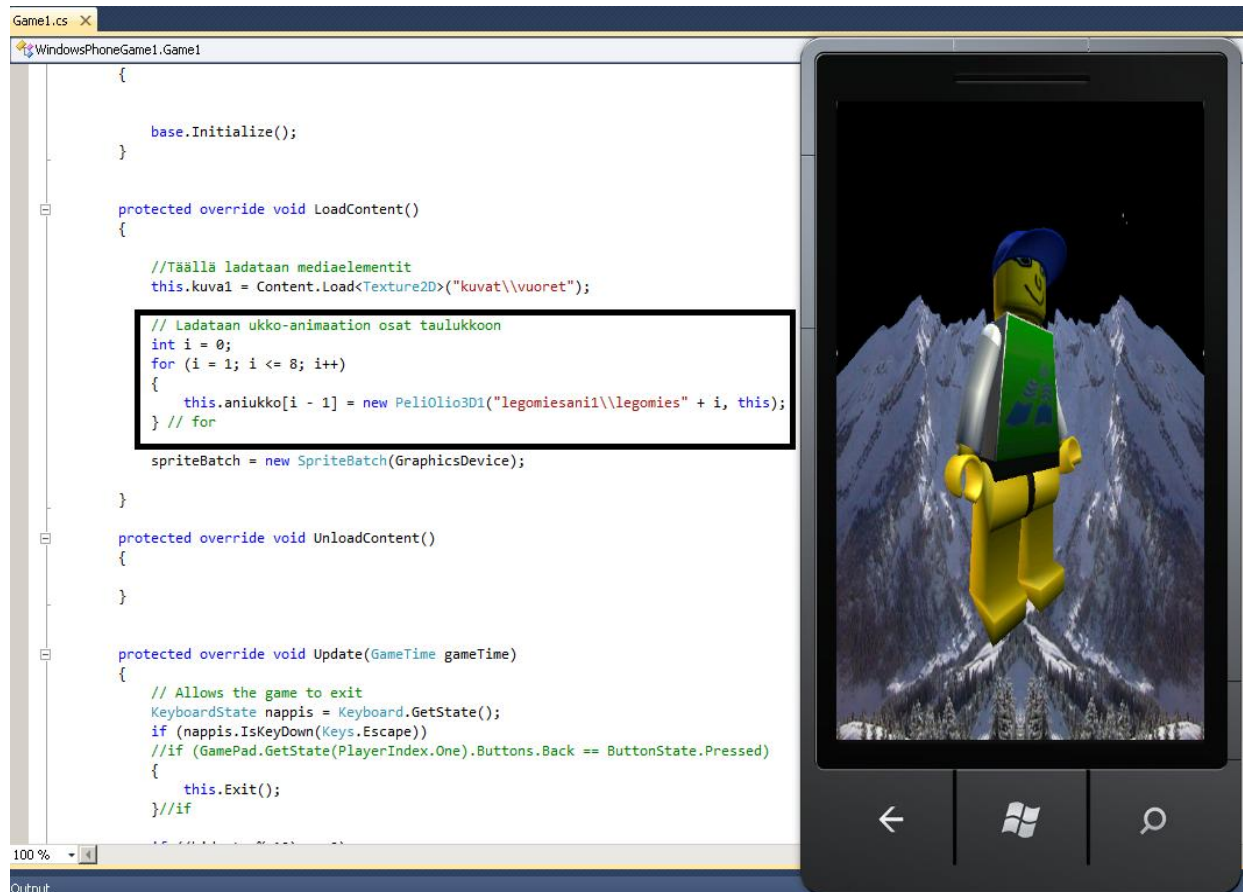
7 WINDOWS PHONE SOVELLUS

Luulin, että sama peli kääntyy suoraan Windows Phonelle, mutta asiahan ei mennyt niin. Virheilmoituksia tuli etenkin käytettävien kirjastojen kanssa, koska ohjelma ei osannut käyttää SkinnedModel-kirjastoa vaikka tiedostot oli laitettu peliprojektin puuhun. Sen takia muutamaa muuttujaa ei voinut käyttää, joten kokeilu ei edennyt. Seuraavaksi selailtuani Internetiä löysin yhden valmiin koodinpätkän, jota testasin ja siinä peli toimi. Tässä koodissa oli käytetty ulkopuolisia kirjastoja valmiiden kirjastojen lisäksi sekä siinä oli tehty muutama erillinen olioluokka joiden avulla pääohjelma toimi. Kuvassa 19 on Windows Phone -sovellus, joka käyttää GPU-skinning -kirjastoja.



KUVA 19. Windows Phone Game

En kuitenkaan lopettanut vielä tähän Windows Phonen tutkimuksiani vaan kokeilin erilaista toimintatapaa vielä lisäksi. Olin aikaisemmin testannut Windowsille tapaa, jossa hahmo ei sisällä lainkaan luurankoa, vaan sen liikkeet luodaan laittamalla monia kuvia, joissa on eri asento pyörimään sovelluksessa. Kokeilin kääntää sitä koodia suoraan Windows Phonelle, mutta siinäkin jouduin hieman kikkailemaan, jotta sain sen toimimaan oikein. Minulla oli pieni kirjoitusvirhe koodissa, joten sen paikantamiseen meni aikaa jonkin verran. Tämän jälkeen sain koodin toimimaan ja hahmo rupesi liikkumaan Windows Phone -emulaattorilla. 3ds Maxissa tein 8 erilaista .fbx-tiedostoa, joissa jokaisessa hahmo oli hieman eri asennossa. Sitten toin tiedostot XNA:ssa peliprojektiin Content Pipelinen kautta. Koodi muistuttaa hyvin pitkälti samanlaista kuin Windowsille tehdyssä pelissä, joka käytti luurankoa ja animaatiota, mutta suurimmat erot ovat siinä, että tähän ei tarvinnut laittaa animaatioitinta eikä Content Processorin tarvinnut olla Skinned model. Kuvassa 20 mustalla reunustetussa koodissa ladataan kaikki 8 .fbx – tiedostoa taulukkoon, josta niitä voi käyttää piirtovaiheessa. Koodi käy läpi kaikki 8 tiedostoa, jonka jälkeen se palaa takaisin ensimmäiseen, jolloin saadaan peli näyttämään siltä, että ukko kävelee siinä. Tämä on loppumaton luoppi, joka jatkuu niin kauan, kuin pelin sulkee.



KUVA 20. Windows Phone Game ilman luurankoa

8 LOPETUS

Tarkoituksena oli tutkia, millainen 3d-malli tulisi tehdä, jotta sen saisi vietyä XNA-peliohjelmointiympäristöön oikein ja ohjelman tulisi pyörittää 3d-hahmoa siinä.

Loppupäätelmäksi voidaan sanoa, että ilman hahmon suunnittelua sekä pelimoottorin rajoituksia selvittämättä projektista tulee vaikea ja aikaa kuluu turhanpäiväisiin testauksiin liikaa. Mielestäni XNA:ssa on liikaa rajoituksia ainakin luurangon tekemisen suhteen, koska jouduin tekemään monia luurankoja hahmolle, ennen kuin sen pystyi viemään XNA:han. Lisäksi koodin kääntäminen suoraan Windows Phonelle ei toiminnut niin kuin luulin vaan se tarvitsi ulkopuolisia kirjastoja huomattavasti enemmän, kuin tavallinen Windows-versio.

Ratkaisuna tutkimusongelmaan voidaan sanoa, että hahmosta pitää tehdä mahdollisimman vähän polygoneja käyttävä, jotta se pyörisi pelissä mahdollisimman sujuvasti ja siten, että tietokoneen laskentatehot riittävät pelin renderöimiseen. Lisäksi luuran-

gossa ei saa olla irtonaisia luita vaan kaikkien tulee olla kiinni jossain toisessa luussa. Luuranko pitää myös rakentaa siten, että luurangon päässä oleva luu on parent-luu, jota liikuttamalla koko luuranko liikkuu. XNA:n rajoituksena on myös hieman alle sadan luun luuranko, joten Biped-luurankoa ei voi tuoda lainkaan siihen, mikäli siitä ei poista joitakin luita, mutta siitä on melko hankala ruveta poistelemaan luita, koska silloin sen topologia muuttuu helposti liikaa. Lisäksi tulee ottaa huomioon tiedosto-
muodot, mitä XNA hyväksyy ottamaan vastaan.

3ds Max on mielestäni hyvä ohjelma 3D-mallien tekemiseen ja sillä saisi luotua tosi näyttäviä hahmoja sekä luurankoanimaatioita, mutta mallintamisen aikana huomasin muutaman kerran, että 3ds Max ilmoitteli sellaisia virheitä, joihin ei löytynyt syytä mistään sekä se kaatui muutaman kerran. XNA taas ei kaatuile itsestään ja koodaaminen C#-kielellä on mukavaa. Luurankorajoitusten takia XNA on hieman heikompi verrattuna esimerkiksi Unity3D-pelimoottoriin, johon sain vietyä luurangon, jossa kaikki luut eivät olleet kiinni toisissaan.

Jatkokehityksenä voisi tehdä esimerkiksi 3d-mallin high-poly-menetelmällä jossa on tarkoituksena tehdä hahmo niin tarkasti kuin mahdollista, jolloin sen polygoni määrä on vähintään kaksinkertainen low-poly-malliin nähden. Sitten voisi pikku hiljaa vähentää siitä polygoneja ja testata sitä XNA:ssa, kuinka paljon hahmossa voi olla polygoneja, että se pyörisi kunnolla pelissä.

Tulevaisuudennäkymät ovat tällä hetkellä hyvät, koska pelifirmoja tulee koko ajan lisää ja tietokoneiden tehot kasvavat, jolloin niiden laskentatehot paranevat ja nopeutuvat, joten pystytään tekemään vieläkin tarkempia 3d-malleja peleihin. Vaikka tehot paranevat, niin laitteista tehdään entistä pienempiä, esimerkiksi kännyköillä pystytään esittämään jo nykyään sellaista grafiikkaa, mitä ei olisi voinut ajatella muutamia vuosia sitten. Tilanne kehittyy sitä paitsi koko ajan jolloin tulevaisuudessa kännyköillä voi tehdä melkein mitä tahansa. Peliteollisuus tulee luultavasti tulevaisuudessa tekemään pelejä suoraan Internet-selaimella pelattaviksi OpenGL-ohjelmointirajapinnan avulla, joka on kokonaan laitteistoriippumaton. Silloin ei tarvitse enää miettiä mille alustalle peliä on tekemässä. Myös pilvipalvelut kehittyvät ja pelejä tullaan pelaamaan suoraan netin yli.

LÄHTEET

- Ahola, Tomi 2011. 3D-mallinnus ja animaatiot. Saimaan Ammattikorkeakoulu, Lappeenranta. Viestintätekniiikan koulutusohjelma. Opinnäytetyö.
- Character modeling tutorial - Sackboy. 2012. Tutorial-z.com. WWW-dokumentti. <http://tutorial-z.com/character-modeling-tutorial-sackboy/4/>. Päivitetty 25.03.2012. Luettu 16.12.2012.
- Exporting FBX Files from 3DS Max. 2012. Media and Entertainment Technologies Laboratory. WWW-dokumentti. <http://metlab.cse.msu.edu/exporting.html>. Ei päivitystietoja. Luettu 04.12.2012.
- Franson, David & Thomas, Eric 2006. Game Character Design Complete: using 3ds max 8 and Adobe Photoshop CS2. Boston, MA, USA: Course Tecnology.
- Heikkinen, Teemu; Kauhanen, Pavel & Pikkarainen, Antti 2008. XNA Pelikehitysympäristönä. Kajaanin ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma. Opinnäytetyö.
- James, Sean 2010. 3D graphics with XNA Game Studio 4.0. Olton Birmingham, GBR: Packet Publishing Ltd.
- Lehtinen Antti. 2007. Ihmisen pään 3D-mallinnus. Second Picture. WWW-dokumentti. http://www.secondpicture.com/tutoriaalit/3d/ihmisen_paan_3d-mallintaminen_3ds_max_01.html. Päivitetty 23.10.2007. Luettu 14.11.2012.
- Lehtinen Antti. 2010. Character Rigging. Polygon blog. WWW-dokumentti. <http://www.polygonblog.com/character-rigging/>. Päivitetty 25.10.2010. Luettu 11.11.2012
- Mallintaminen. 2012. 3D Raamattu. WWW-dokumentti. http://174.121.152.34/~asdderbi/3d%20raamattu/oikea_mallinnus.html. Ei päivitystietoja. Luettu 20.11.2012.

Murdock, Kelly L. 2010. 3ds Max 2011 Bible. Indianapolis Indiana: Wiley Publishing Inc.

Nieminen, Mikael 2009. Lowpoly-hahmon riggaus 3ds Max-ohjelmassa. Lahden ammattikorkeakoulu. Mediatekniikan koulutusohjelma. Opinnäytetyö.

Orava, Peetu 2011. 3D-hahmon luonti ja ohjaaminen Cave-virtuaaliympäristössä. Seinäjoen ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma. Opinnäytetyö.

Reed, Aaron 2011. Learning XNA 4.0 Game Development for the PC, Xbox 360, and Windows Phone 7. Gravenstein Highway North, Sebastopol CA: O'Reilly.

Brooks Mike. 2011. SkinnedCharacter. glos. WWW-dokumentti. <http://mmedia.glos.ac.uk/igd220/T3/SkinnedCharacter.html>. Päivitetty 2011. Luettu 20.11.2012.

Stoneham, Bill 2010. How to Create Fantasy Art for Computer Games. Huntingdon, GBR: A & C Black.

The Vitruvian Man. 2012. Rockland Community College. WWW-dokumentti. http://www.sunyrockland.edu/Members/jsansone/photo-gallery/Vitruvian%20Man%20%201.jpg/image_view_fullscreen. Päivitetty 2012. Luettu 15.10.2012.

Viitapohja, Liisa 2010. 3D-hahmon mallinnus: Ihmishahmon suunnittelu ja toteuttaminen Blender-ohjelmalla. Metropolia Ammattikorkeakoulu. Viestintä. Opinnäytetyö.

What is the Content Pipeline? 2012. Microsoft. WWW-dokumentti. <http://msdn.microsoft.com/en-us/library/bb447745.aspx>. Päivitetty 2012. Luettu 21.12.2012.

Williams, Chris G & Clingerman, George W 2011. Professional Windows Phone 7 Game Development: Creating Games Using XNA Game Studio 4. Hoboken, NJ, USA: Wrox.

Ylimäki, Juha 2011. Valaisukartat reaaliaikaisessa renderöinnissä. Metropolia Ammattikorkeakoulu. Viestintä. Opinnäytetyö.

3ds Max Model Rigging. 2012. Cygon's Blog. WWW-dokumentti. <http://blog.nuclex-games.com/tutorials/graphics/3ds-max-model-rigging/>. Ei päivitystietoja. Luettu 21.12.2012.

3DS Max Tutorial – Basic Animation Techniques. 2006. Expert Rating. WWW-dokumentti. <http://www.expertrating.com/courseware/3DCourse/3D-Animation-1.asp>. Päivitetty 2006. Luettu 4.1.2013.