jamk.fi

3D-pelin NPC-hahmon animaatio

Julianna Hautoniemi

Opinnäytetyö
Maaliskuu 2015

Mediatekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Hautoniemi, Julianna	Opinnäytetyö	31.03.2015
		Julkaisun kieli Suomi
	Sivumäärä 56	Verkkojulkaisulupa myönnetty: x

Työn nimi
3D-pelin NPC-hahmon animaatio

Koulutusohjelma
Mediatekniikan koulutusohjelma

Työn ohjaaja(t)
Niemi, Kari

Toimeksiantaja(t)
GameHunger projects

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi GameHunger projects, joka on aloitteleva pelinkehittäjätiimi. Tiimillä on työn alla heidän ensimmäinen 3D-videopeli.

Tavoitteena oli luoda yhdelle pelin lintumaisista NPC-hahmoista animaatiota erilaisia pelitilanteita varten. Hahmoanimaatio luotiin alusta loppuun 3ds Max -ohjelmalla. Animaatio-ohjelmistoa voidaan kuitenkin joutua vaihtamaan kesken pelinkehitysprosessin. Jatkokehityksen kannalta tavoitteena oli selvittää, kuinka 3ds Maxilla luotua animaatiota voidaan muokata myöhemmin Blenderillä.

Tietoperustassa käsitellään reaaliaikaista renderöintiä pääpiirteittäin ja yleistäen sen vaatimuksia animaation suhteen. Ennen hahmoanimaation toteutusta käydään läpi lyhyesti pelihahmon animaation työvaiheet, eli mallintaminen, riggaus, skinnaus ja animointi. Käytännön osiossa esitellään 3ds Max -ohjelman käyttöä ja työkaluja hahmoanimaation teossa ja käydään läpi lintuhahmon animaation suunnittelu ja toteutus. Loppuosa työstä keskittyy siihen, miten 3ds Maxilla luotu hahmoanimaatio saadaan järkevästi muokattavaan muotoon Blenderiä varten.

Opinnäytetyössä luotiin teknisesti toimivaa ja hyvännäköistä animaatiota ottamalla huomioon pelimoottorin ja reaaliaikaisen renderöinnin vaatimukset. Opinnäytetyön tuloksena syntyi myös kirjallinen selvitys siitä, mitä ja miten 3ds Maxilla luotua hahmoanimaatiota voidaan tallentaa Blenderin tukemaan muotoon.

Avainsanat (asiasanat)
3D-videopeli, NPC-hahmo, hahmoanimaatio, 3ds Max

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Hautoniemi, Julianna	Bachelor's thesis	31.03.2015
		Language of publication: Finnish
	Number of pages: 56	Permission for web publication: x

Title of publication
NPC animation in 3D video game

Degree programme
Media Engineering

Tutor(s)
Niemi, Kari

Assigned by
GameHunger projects

Abstract

This bachelor's thesis was assigned by GameHunger projects which is a small game development team. The team is currently working on their first 3D video game.

The purpose of the thesis was to create some animations to a non-player character of the 3D video game. The animation was created in 3ds Max animation software. However, the animation software may need to be replaced during the game development process. For future development the aim was to find out how animations created in 3ds Max can be edited in Blender.

The theoretical part of the thesis deals with real-time rendering in general and its requirements for animation. It also goes briefly through the character animation process starting with modeling and ending with animating. The practical part discusses the creation of some different animations for the NPC in 3ds Max with its special tools. The final part illustrates how animation created in 3ds Max can be edited particularly in Blender.

A technically functional and good-looking character animation was created during the thesis project. Creating animation, real-time rendering and its requirements were taken into consideration. The outcome of the thesis is a document that reveals what kind of animation created in 3ds Max can be saved in a format supported by Blender.

Keywords/tags (subjects)
3D video game, NPC, character animation, 3ds Max

Miscellaneous

Sisältö

Kuviot

Käsitteet

Low poly

3D-malli koostuu polygoneista, Low poly -mallissa polygoniverkon tiheys on pieni ja 3D-malli on tällöin laskennallisesti kevyt.

Modifier

Muokkain, joka lisätään 3D-mallinnusohjelmassa 3D-objektille. Useimmiten muokkaimet muokkaavat 3D-objektin geometriaa jollain tavoin ja yhdellä 3D-objektilla voi olla useita muokkaimia samaan aikaan käytössä.

NPC-hahmo

Non-player character on pelihahmo, joka ei ole pelattava. Esim. vihollinen tai hahmo, joka tarjoaa palautuspisteen tai ekstravoimia.

Renderöinti

Tietokone laskee 3D-grafiikan 2D-kuvaksi. Renderöidessään kone laskee 3D-mallin lisäksi tekstuurit, valot ja muut efektit ja muodostaa pikseleistä koostuvan kuvatiedoston.

Riggaus

Riggauksella tarkoitetaan hahmoanimaation sitä työvaihetta, jossa luodaan hahmolle luut, eli animointijärjestelmä.

Skinnaus

Skinnaus on työvaihe, jossa hahmon ”iho” kiinnitetään luihin ja jokaisen luun vaikutus-alue ”ihoon” säädetään sopivaksi.

1 Työn lähtökohdat

1.1 Taustaa ja toimeksiantaja

Kolmiulotteinen (Three-dimensional, 3D) videopelimaailma on pelaajalle kuin toinen todellisuus: siellä voi tutkia ympäristöä, suorittaa erilaisia tehtäviä, kokea elämyksiä ja viihtyä. Pelaajien vaatimustaso pelimaailman realistisuuden suhteen nostaa pelinkehittäjien työmäärää jatkuvasti. Pelisarjan uusin peli on aina edellistä osaa hieman näyttävämpi ja tekniikan kokoajan kehittyessä tehdään entistä realistisempaa jälkeä, mikä nostaa taas vuorostaan pelaajien odotuksia ja vaatimustasoa. Nykyään on onneksi kehittyneitä pelimoottoreita, joten pelinkehittäjät voivat keskittyä yksityiskohtiin tehdessään pelistä ainutlaatuista elämystä.

Pelinkehitys on prosessi, joka vaatii monen alan ammattilaisia ja asiantuntijoita. Ilman hyvää käsikirjoitusta pelissä ei ole juonta, ilman ohjelmoijia peli ei toimi, ilman musiikkia pelissä ei ole tunnelmaa ja ilman mallintajia ja animaattoreita pelissä ei ole oikeastaan mitään nähtävää. Pelinkehitysprosessissa on mukana monta tekijää, joilla jokaisella on oma erikoisosaamisensa.

Toimeksiantajana oli GameHunger projects, joka on omien sanojensa mukaan pieni tiimi, jonka jäsenet rakastavat videopelejä, niiden suunnittelua, tekemistä ja tietysti pelaamista. Tiimillä on suunnitteilla perustaa yritys lähiaikoina, ja tiimi etsii koko ajan uusia innokkaita tekijöitä ja myös muita mahdollisia yhteistyökumppaneita muualtakin kuin pelialalta. Tällä hetkellä GameHunger projects -tiimillä on työn alla pelattava demoversio heidän ensimmäisestä 3D-videopelistään. Pelin demoversiossa pelimoottorina on käytössä paljon käytetty Unity. Peli on ensimmäisen/kolmannen persoonan ammuntapeli, jossa on seikkailu- ja kauhuelementtejä. Pelin maailmassa jahdataan lintuja tai tullaan jahdatuksi laumojen tai yksittäisten lintujen hyökätessä. Peli alkaa, kun Maahan matkalla olleet avaruusmatkaajat tekevät pakkolaskun sumuiselle, oudolle ja vaaralliselle planeetalle. Pelaajan tehtävänä on löytää tiensä pelastuskapselille ja tärkeimpänä - selvitä matkasta hengissä.

1.2 Tavoitteet ja tehtävät

Työn tavoitteena oli suunnitella ja luoda valmiiksi mallinnetulle ihmismäiselle lintuhahmolle hyökkäysanimaatioita ja muutamia erilaisia liikkumisanimaatioita sekä muita tarvittavia animaatioita tekeillä olevaa peliä ja pelin esittelyvideota varten. Koska peliin tulee myöhemmin monenlaisia erilaisia lintuhahmoja, tavoitteena oli myös löytää toimiva ja melko mutkaton ratkaisu käyttää jo luotuja animaatioita edes osittain uudestaan fysiikaltaan erilaisten hahmojen kanssa.

Kesken pelinkehitysprosessia voidaan joutua vaihtamaan syystä tai toisesta animaatio-ohjelmistoa, mutta työtä olisi kuitenkin päästävä jatkamaan siitä pisteestä mihin se jäi. Jatkokehityksen kannalta tärkeä tutkimuskohde oli ohjelmien välillä tapahtuva animaation siirtomahdollisuus ja sen jälkeen muokkaus. Tutkimustyön tuloksena syntyi dokumentti, joka kertoo mitä 3ds Maxilla luotua animaatiota voidaan muokata myöhemmin Blenderillä.

Aluksi työssä tutustuttiin 3D-pelimaailmaan, NPC-hahmoihin, pelimoottoriin ja reaaliaikaiseen renderöintiin ja sen vaatimuksiin peligrafiikan ja animaation suhteen. Reaaliaikaisesta renderöinnistä kerrotaan luvussa 2.4. Seuraavaksi oli animaation suunnittelun ja toteutuksen vuoro. Lintuhahmoanimaation suunnitteluvaiheessa tutkittiin ja vertailtiin linnun ja ihmisen anatomiaa ja lintujen tapaa liikkua verrattuna ihmiseen. Toteutusvaihe sisälsi lintuhahmon riggauksen, skinnauksen ja animoinnin. Hahmoanimaation työvaiheet käydään yleisellä tasolla läpi luvussa 3 ja tarkemmin luvussa 4 tämän työn toteutuksen kautta. Kaikki hahmoanimaation työvaiheet riggauksesta animointiin tehtiin käyttäen 3ds Maxia. Perusteet 3ds Maxista ja sillä animoinnista oli hallussa, joten 3ds Max oli luonteva valinta. Hahmoanimaation ollessa jo melko valmis, tutkittiin vielä vaihtoehtoja, miten tässä työssä luotua animaatiota voi muokata muilla ohjelmilla ja käyttää hyväksi tulevissa projekteissa. Animaation muokattavuus ja uudelleenkäytettävyys on esitelty luvussa 5.

2 Hahmoanimaatio 3D-videopelissä

2.1 Mikä on 3D-videopeli?

Moni suurempi pelijulkaisu on nykyään 3D-peli, tai siinä on käytetty paljon 3D-elementtejä (Silverman 2013). Mikä sitten on 3D-videopeli? 3D-videopeliksi voidaan kutsua mitä tahansa videopeliä, jonka grafiikka on kolmiulotteista. Kolmiulotteinen pelimaailma vastaa paremmin oikeaa maailmaa verrattuna 2D-peleihin. Kolmiulotteisen pelimaailman kolmas ulottuvuus antaa pelaajalle mahdollisuuden katsella ympärilleen, tutkia ja vaeltaa pelimaailmassa, kuin reaalimaailmassa. Ennen 3D-pelien valtakautta 2D-pelit olivat aikanaan omanlaisiaan elämyksiä, mutta 3D-pelit ovat nostaneet pelikokemuksen aivan uudelle tasolle. 3D-pelissä pelitilanne ei ole koskaan täysin samanlainen johtuen pelaajan vapaudesta liikkua useampaan suuntaan esimerkiksi vihollisiin nähden.

2.2 Pelattavuus ja NPC-hahmojen interaktiivisuus

NPC-hahmo, eli englanniksi non-player character, on tietokoneen ohjaama sivuhahmo. NPC-hahmot voivat olla isokin osa tarinaa tai vain tarinaa tukevia hahmoja, ne voivat esimerkiksi antaa pelaajille tehtäviä, toimia palautuspisteinä tai antaa lisävoimia. NPC-hahmot ovat yleensä ystävällisiä, eivät ainakaan avoimen vihamielisiä. (NPCs 2014, Non-Player Character 2014). Tämän työn hahmo on pelin suunnittelijoiden mukaan NPC, vaikka se on avoimen vihamielinen ja hyökkäävä, poikkeuksia löytyy varmasti muistakin peleistä.

NPC-hahmot olivat automatisoituja ja melko rajoittuneita pelattaviin hahmoihin verrattuna varhaisissa tietokoneella pelattavissa roolipeleissä. Ne usein vain odottelivat paikallaan pelaajan saapumista niiden luo, ja mahdollinen keskustelu hoidettiin avainsanojen tai pelaajalle tarjottujen valmiiden repliikkien avulla. Nykyisin, ohjelmoinnin kehityttyä, NPC-hahmoista on tullut aktiivisempia ja niin sanotusti älykkäämpiä. Pelaajan hahmo voi saada esimerkiksi tietokoneen ohjaamia apureita, joille voi antaa yksinkertaisia käskyjä. NPC-hahmot voivat myös muuten reagoida ympäristöönsä. Pelaajan tai pelaajan ja NPC-

kumppanin saapuminen tiettyyn paikkaan voi laukaista jonkin NPC-hahmolle käsikirjoitetun kohtauksen tai piilossa olevan toisen NPC- hahmon liikkeellelähdön.

Peleissä, joissa on NPC-vihollisia, vihollisten tulee reagoida pelaajan sijaintiin, liikkeisiin ja usein myös aseisiin. Vihollinen ei ainakaan vahingossa saa kääntää katsettaan ja huomiotaan pois pelaajasta, kun pelaaja ampuu tai tekee jotain vastaavaa. Älykkäältä vaikuttavalla NPC-hahmolla on esimerkiksi taistelutilannetta varten laaja valikoima valmiita liikesarjoja, joista toistetaan jopa sattumanvaraisesti tietty liikesarja vasteena pelaajan toiminnalle ja sijainnille.

Vaikka pelin kenttäsuunnittelija pyrkii suunnittelemaan ja ennustamaan, mitä pelaaja mahdollisesti tekee missäkin pelitilanteessa, on mahdotonta ennustaa kaikkia mahdollisia pelaajan mielenliikkeitä. Pelitilanne on aina erilainen, koska pelaaja on vuorovaikutuksessa peliympäristön kanssa. Interaktiivisuutta pyritään luomaan kuitenkin pelimoottorin tekoälyn avulla ja luomalla NPC-hahmoille useita erilaisia animaatioita erilaisten pelitilanteiden varalle.

2.3 Pelimoottori

Pelimoottori on videopelejä varten suunniteltu tietynlainen ohjelmistokehys, jonka avulla rakennetaan pelejä konsoleille, PC-ympäristöön tai mobiililaitteille. Pelimoottori on se, joka saa pelin rullaamaan eteenpäin, vaikka toisinaan saattaa olla epäselvä raja, missä pelimoottori loppuu ja pelin sisältö alkaa. Pelimoottori suorittaa pelin ydintoiminnot, joita ovat mm. kuvan luonti, fysiikkamoottori ja muistinhallinta, jotta pelin kehittäjät voivat keskittyä yksityiskohtiin, jotka tekevät pelistä ainutlaatuisen. Pelimoottorit tarjoavat komponentteja, joita manipuloimalla toteutetaan mm. 3D-mallien lataus ja näyttö, animaatio, törmäyksen tunnistus, pelin graafinen käyttöliittymä ja jopa osa pelin tekoälystä. (Ward 2008.)

Unity

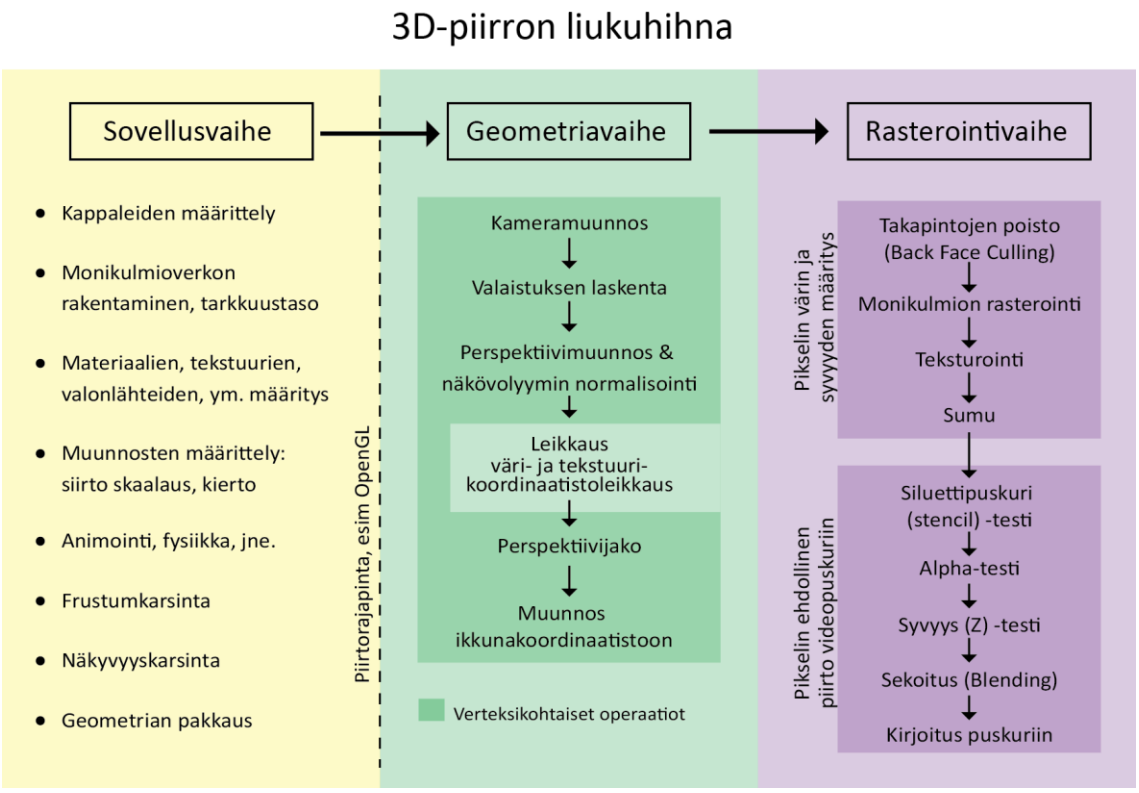
GameHunger projects käyttää pelin demoversiossa pelimoottorina Unitya, joka on Unity Technologiesin kehittämä pelimoottori. Unity on oikeastaan alustariippumaton pelinkehitysympäristö, joka sisältää pelimoottorin ja ohjelmointiympäristön. Unity on melko

helppokäyttöinen ja sillä voidaan kehittää kaksi- tai kolmiulotteisia pelejä selaimelle, konsoleille, PC:lle tai mobiililaitteille. Sillä voidaan julkaista pelejä useille alustoille, esimerkkeinä Android, Windows, iOS, Linux, PlayStation 4 ja Wii. Unitysta on saatavilla ilmainen Unity ja maksullinen versio Unity Pro, joka sisältää enemmän ominaisuuksia. (Unity 2014.)

2.4 Reaaliaikarenderöinti

Kolmiulotteisesta pelimaailmasta luodaan kuva (tietokoneen) näytölle, eli renderöidään kuva pelaajan näkemästä maisemasta ja tilanteesta reaaliajassa. Pelimoottori laskee 30–60 kuvaa sekunnissa. Reaaliaikarenderöinti tarkoittaa siis renderöintiä, joka tapahtuu reaaliajassa. Reaaliaikaisen renderöinnin vastakohta on renderöinti, joka tapahtuu etukäteen, esimerkiksi 3D-elokuvia ei renderöidä reaaliajassa, vaan niitä renderöidään jopa vuosia etukäteen ja renderöidyistä 2D-kuvista muodostetaan vasta elokuva.

Reaaliaikainen renderöinti on suoraviivainen prosessi ja sitä voidaan kutsua myös piirtoliukuhihnaksi (engl. The Graphics Rendering Pipeline). Piirtoliukuhihna tuottaa kaksiulotteisen kuvan virtuaalisen kameran näkemistä kolmiulotteisista objekteista, valonlähteistä, tekstuureista ja muista elementeistä. Piirtoliukuhihnalla on kolme vaihetta: sovellusvaihe (engl. the application stage), geometriavaihe (engl. the geometry stage) ja rasterointivaihe (engl. the rasterizer stage). Kahta viimeistä vaihetta voidaan myös pitää omina liukuhihnoinaan, koska ne on edelleen vaiheistettu. (Akenine-Möller, Haines & Hoffman 2008, 11.) Kuvio 1 havainnollistaa piirtoliukuhihnan vaiheistusta ja listaa kussakin vaiheessa suoritettavat operaatiot.



Kuvio 1. 3D-piirron liukuhihna (alkup. kuviot ks. Puhakka 2008, 164–170)

2.4.1 Sovellusvaihe

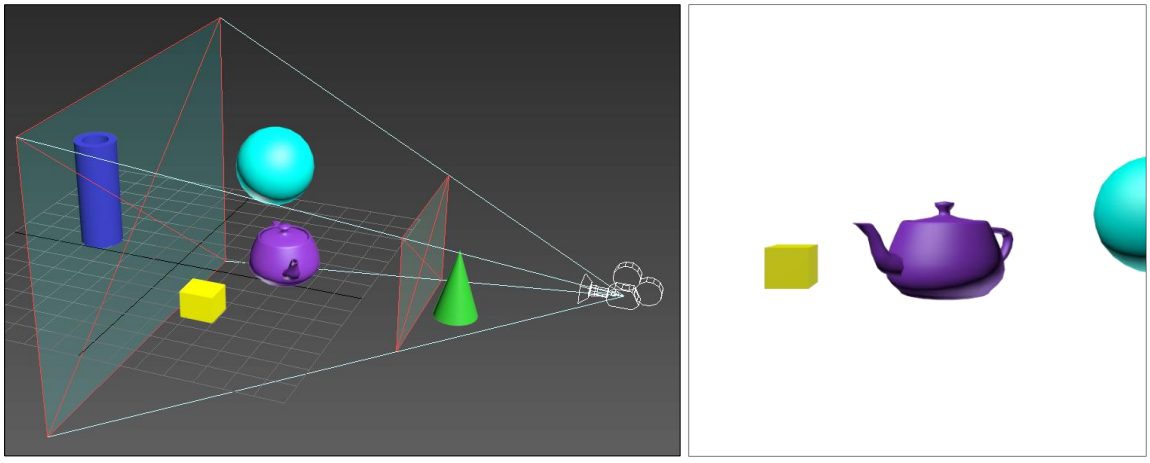
Piirtoliukuhihnan ensimmäinen vaihe on sovellusvaihe. Sen tärkein tehtävä on määrittää ja sitten syöttää liukuhihnan seuraavaan vaiheeseen renderöitävät primitiivit (pisteet, viivat ja kolmiot), jotka päätyvät lopulta näytettäväksi ruudulla tai näytöllä. Sovellusvaiheessa huolehditaan myös mm. näppäimistön ja hiiren syötteiden käsittelystä. Muita sovellusvaiheen prosesseja ovat lisäksi mm. tekstuurien animaatiot, geometrian muutosanimaatiot ja muut laskennalliset operaatiot, joita ei suoriteta muissa vaiheissa. (Akenine-Möller ym. 2008, 15.)

Sovellusvaiheen suorittaa tietokoneen prosessori. Sovellusvaihetta ei ole edelleen vaiheistettu, mutta suorituskyvyn lisäämiseksi tämän vaiheen tehtäviä suorittaa usein rinnakkain useampi prosessoriydin. Muutokset sovellusvaiheessa voivat vaikuttaa suorituskykyyn myös myöhemmissä vaiheissa, esimerkiksi jokin algoritmi tai asetus voi vähentää

renderöitävien polygonien määrää. (Akenine-Möller ym. 2008, 14–15.) Polygonien määrää vähennetään, koska geometriavaiheessa GPU:n (graphics processing unit) muisti on rajallinen. Raskaiden mallien polygoniverkkoja voidaan yksinkertaistaa suhteessa etäisyyteen kamerasta. Malli ei tarvitse kaikkia yksityiskohtia ollessaan kaukana kuvaruudusta, joten sitä voidaan karsia yksinkertaisemmaksi menettämättä kuitenkaan hyvää kuvanlaatua. Kaukana oleva maisema, usein ainakin puut ja muu kasvusto, kutistetaan jopa neliöpinnoiksi saakka. Geometrian vähentämistä tällä tavoin kutsutaan LOD:ksi eli Level Of Detail.(Puhakka 2008, 327–328.)

Frustumikarsinta (engl. frustum selection) liittyy kameraan. Pelimoottorin virtuaalinen kamera sijaitsee 3D-maailmassa jossain tietyssä pisteessä (x,y,z) ja sillä on frustumin eli katkaistun pyramidin mallinen näkökenttä, jota kutsutaan näköfrustumiksi. Vain kameran näkemät objektit renderöidään ja selvästi näkökentän ulkopuolelle jäävät objektit karsitaan jo sovellusvaiheessa pois. Tätä karsintaa kutsutaan frustumikarsinnaksi. Näköfrustumille on määritelty, kuinka korkea ja jyrkkä se on. Näköfrustumin koko ja jyrkkyys määräytyy kameran perspektiivin mukaan. Näköfrustumille määritelty takaleikkaustaso rajaa pois liian kaukana olevat objektit ja etuleikkaustaso rajaa pois liian lähellä ja pelaajan takana olevat objektit. (Akenine-Möller ym. 2008, 12; Puhakka 2008, 187.)

Vasemmalla puolella on kuva tilanteesta, jossa on kamera ja muutamia objekteja, oikealla puolella on kameran näkemä, renderöity, 2D-kuva (ks. kuvio 2).



Kuvio 2. Frustumikarsinta

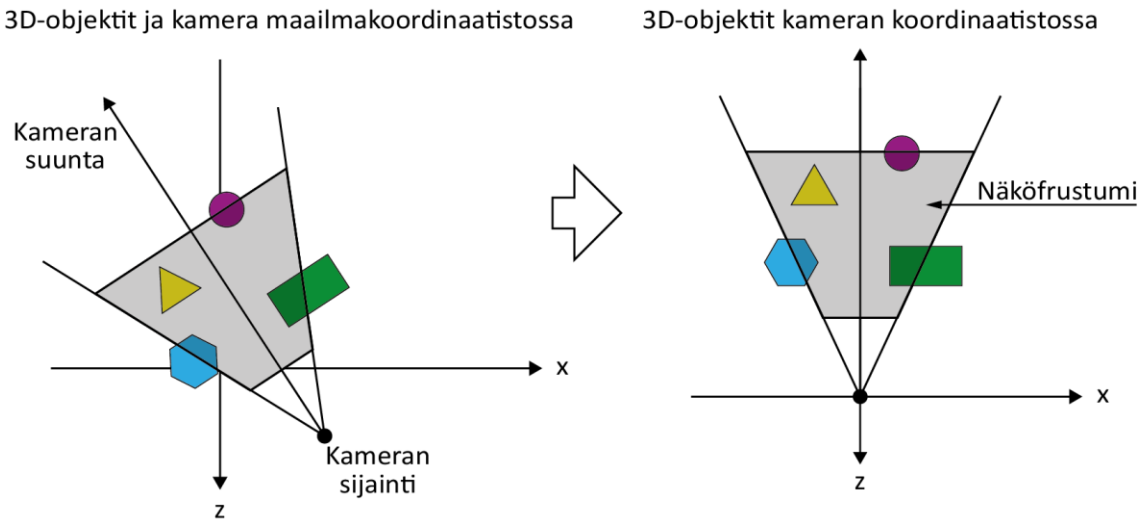
Frustumikarsinta on karkea karsinta, joten sen lisäksi suoritetaan vielä tarkempi näkyvyyskarsinta, joka rajaa toisten objektien takana olevat objektit kokonaan tai osittain pois. Näkyvyyskarsinta suoritetaan realistisen lopputuloksen (kuvan) ja tehokkuuden takia. On turhaa käsitellä liukuhihnalla kohteita, joita katsoja ei näe, joten on tehokkaampaa karsia ne pois heti liukuhihnan alkupuolella. (Puhakka 2008, 259–260.) Sovellusvaiheen lopulla geometria vielä pakataan mahdollisimman tiiviisti, ennen kuin se siirretään eteenpäin geometriavaiheeseen (Mt. 167).

2.4.2 Geometriavaihe

Piirtoliukuhihnan toinen vaihe, geometriavaihe, määrittää mitä piirretään, miten piirretään ja minne piirretään, eli se on vastuussa polygoni- ja verteksikohtaisista operaatioista. Geometriavaihe on piirtoliukuhihnan ensimmäinen varsinainen vaihe ja se on edelleen jaettu toiminnallisiin vaiheisiin.

Matkalla näytölle 3D-mallit käyvät läpi useita muunnoksia koordinaatistosysteemistä toiseen. 3D-malli sijaitsee aluksi omassa malliavaruudessa/koordinaatistossaan. 3D-mallin koordinaatit ovat mallin koordinaatistomuunnoksen jälkeen 3D-maailman koordinaatteja ja kaikki mallit sijaitsevat lopulta samassa maailmakoordinaatistossa. (Akenine-Möller ym. 2008, 16.)

Kameralla on myös oma koordinaatisto, ja mallit siirretäänkin seuraavaksi kameran eli katsojan koordinaatistoon, jotta mallien käsittely helpottuu geometriavaiheen seuraavissa vaiheissa. Kamera sijaitsee oman koordinaatistonsa origossa ja katsoo z-akselin suuntaan, x-akseli osoittaa oikealle ja y-akseli ylös. (Puhakka 2008, 173.) Alla oleva kuva (ks. kuvio 3kuvio 3) näyttää, kuinka maailmakoordinaatistossa olevat objektit siirretään kameran koordinaatistoon. Koordinaatistomuunnos on kuvattu ylhäältäpäin, y-akselin suuntaisesti.



Kuvio 3. Mallien koordinaatistomuunnos kameran koordinaatistoon

Koordinaatistomuunnoksen jälkeen lasketaan valaistuksen vaikutus objekteihin, valonlähteiden aiheuttamat heijastukset, valoisat kohdat ja varjot. Valaistuksen laskennan jälkeen objektit käyvät läpi perspektiivimuunnoksen. Perspektiivimuunnoksen tuloksena kauempana olevat kohteet näyttävät lähellä olevia kohteita pienemmiltä, eli näkymä muunnetaan entistä realistisemmaksi. Seuraavaksi käsiteltävää näkymää jälleen rajataan, nyt leikataan pois primitiivejä, eli kulma-/verteksipisteitä, jotka jäävät kameran näkemän näkymän ulkopuolelle. Leikkaus hävittää, mutta myös luo uusia kulmapisteitä, joten kulmapisteisiin liittyviä väri-, tekstuuri- ym. tietoja interpoloidaan leikkauksen jälkeen uusille kulmapisteille. (Puhakka 2008, 169.)

Leikkauksen jälkeinen perspektiivijako tarkoittaa, että palataan xyz-koordinaatistoon. Näkymä muunnettiin geometriavaiheen alussa neliulotteiseen homogeeniseen koordinaatistoon, jotta pystyttäisiin käsittelemään objektien etäisyyksien suhteita ja geometrisesti mahdottomia tilanteita, kuten katseluperspektiivin aiheuttamaa kahden samansuuntaisen viivan kohtaamista. (Puhakka 2008, 196.)

Geometriavaiheen lopussa suoritetaan vielä yksi koordinaatistomuunnos ennen rasterointivaihetta. Tämä viimeinen koordinaatistomuunnos muuntaa näkymän näyttölaitteen ikkunakoordinaatistoon, jotta näkymä voidaan rasteroida. Näyttölaitteen piirtoik-

kuna on px pikseliä leveä ja py pikseliä korkea ja origo on vasemmassa alakulmassa. Kaksiulotteisesta koordinaatistosta huolimatta näkymän pisteillä on edelleen myös syvyys, näkymä litistetään vasta rasterointivaiheessa. (Puhakka 2008, 169,198.)

2.4.3 Rasterointivaihe

Rasterointivaiheessa näkymä litistetään, eli rasteroidaan muuntamalla geometriavaiheesta saatu data pikseleiksi ja värittämällä pikselit. Rasterointivaihe on myös edelleen vaiheistettu.

Rasterointivaiheen alussa suoritetaan jälleen karsintaa, tällä kertaa poistetaan liukuhihnan käsittelystä pinnat (kolmiot), jotka osoittavat katsojasta poispäin. Tämä karsinta on nimeltään takapintojen poisto. Katsojaa kohti olevat pinnat litistetään kaksiulotteiseen koordinaatistoon ja pinnat väritetään tai teksturoidaan. Värityksen ja teksturoinnin jälkeen pikseleihin lisätään vielä sumun väriä, jos pelissä käytetään sumua. (Puhakka 2008, 169–171.)

Rasterointivaiheen lopulla pikseleille tehdään joitakin testejä, jotka määrittävät mitkä pikselit muodostavat lopullisen kuvan. Testeillä tutkitaan pikselien läpinäkyvyyttä ja niiden sijaintia syvyyssuunnassa. Jos jokin pikseli jää aikaisemmin piirretyn pikselin taa, sitä ei piirretä. Jos taas pikseli on aikaisemmin piirretyn pikselin edessä, se korvaa aiemmin piirretyn pikselin tai sen väri sekoitetaan aiemman pikselin väriin. Pikselitestien jälkeen lopulliset pikselit kirjoitetaan videopuskuriin/näyttöpuskuriin, josta pikselien muodostama kuva kopioidaan näytölle. (Puhakka 2008, 169–171.)

2.5 Animaation optimointi reaaliaikarenderöintiin

Edellisessä luvussa käytiin läpi pelimoottorin suorittaman reaaliaikaisen renderöintiprosessin kulku pääpiirteittäin. Nyt selvitetään, mitä tulee ottaa huomioon ja kuinka 3D-mallia tulee käsitellä, jotta se on mahdollisimman optimaalinen ja sopii käytettäväksi reaaliaikaisessa renderöinnissä. Mallit optimoidaan, koska pelimoottorit eivät pysty käsittelemään kuin rajatun määrän polygoneja hyväksyttävässä ajassa. 3D-objektit muodostuvat polygoneista, joilla on kulmapisteet 3D-koordinaatistossa. 3D-objektien optimointi

tapahtuu pääasiassa niiden mallinnusvaiheessa, ennen animointia ja vientiä pelimoottoriin.

2.5.1 CPU-optimointi

CPU-optimointi tarkoittaa piirtokutsujen vähentämistä niin, että yhdellä piirtokutsulla voidaan piirtää mahdollisimman paljon tavaraa näytölle. CPU, keskusyksikkö, lähettää piirtokutsun (engl. draw call) näytönohjaimen ajurin kautta näytönohjaimelle x kertaa sekunnissa. (Optimizing Graphics Performance 2014.) Scenellä on esimerkiksi kolmekymmentä renderöitävää 3D-mallia ja niillä kaikilla on käytössään kaksi erilaista materiaalia. Jos yksikään malli ei käytä toisen mallin kanssa samaa materiaalia, CPU lähettää tässä tapauksessa vähintään kuusikymmentä piirtokutsua jokaisen 2D-kuvan luomiseksi näytölle. Näytölle piirretään jopa 60 kuvaa sekunnissa, jotta katsoja mieltää näkemänsä liikkuvaksi kuvaksi, eli siihen tarvitaan tässä tapauksessa ainakin noin 3600 piirtokutsua sekunnissa.

Animoitujen objektien optimointiin pätevät samat säännöt kuin liikkumattomienkin objektien optimointiin. Valot, varjot, heijastukset ja käytettyjen materiaalien määrä vaikuttavat suorituskykyyn. Piirtokutsuja tarvitaan yhtä monta, kuin on valonlähteitä ja materiaaleja. (Optimizing Graphics Performance 2014.) Scenellä on esimerkiksi pensas, jossa on noin 3000 lehteä ja jokainen lehti on yksi kolmionmuotoinen taso/polygoni. Lehdet kuitenkin jakavat yhden ja saman materiaalin, joten ne on suorituskyvyn kannalta tehokkaampaa liittää yhdeksi mesh-objektiksi. Yhden mesh-objektin, pensaan lehvästön, piirtämiseksi piirtokutsuja tarvitaan vain jopa yksi ainut, koska enää scenellä on vain yksi piirrettävä objekti 3000 erillisen kolmion sijaan.

Hahmoanimaation kannalta objektien yhdistäminen tarkoittaa sitä, että hahmon iho ja vaatteet ovat samaa mesh-objektia, jos vaatteita ei ole tarkoitus riisua. Yksittäinen hahmo, jonka vaatteet ja iho eivät ole samaa mesh-objektia, ei ole vielä ”rikos”, mutta jo useampi hahmo samaan aikaan scenellä voi hyvinkin vaikuttaa suorituskykyyn, jos niillä kaikilla on eri vaatekappaleet erillisinä mesh-objekteina. Kun hahmon iho ja vaatteet ovat samaa objektia, tarvitaan parhaimmassa tapauksessa vain yksi ainut tekstuuri, joka kääritään hahmon ympärille.

2.5.2 GPU-optimointi

GPU-optimointi pyrkii vähentämään mallien geometriaa, jotta mallien prosessointi pelimoottorissa olisi nopeampaa ja siten tehokkaampaa. GPU (Graphics Processing Unit), eli tietokoneen grafiikkaprosessori prosessoi 3D-objektien geometriaa. Geometrian optimointiin liittyen on kaksi perussääntöä, joista ensimmäinen: ei yhtään ylimääräistä polygonia. (Optimizing Graphics Performance 2014.)

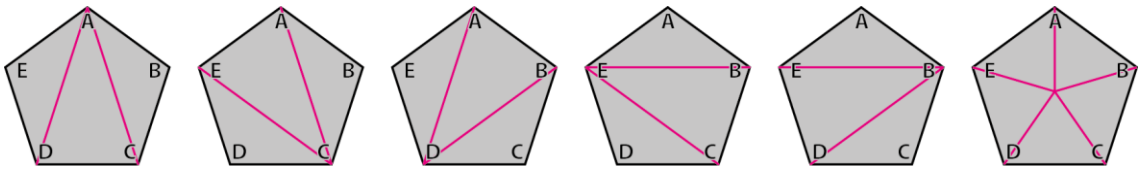
Useimmiten peleissä käytetäänkin hahmoja, joissa on vain vähän polygoneja verrattuna elokuvissa esiintyviin hahmoihin. Syy on yksinkertainen: pelimaailmasta renderöidään reaaliajassa se näkymä, jonka pelaaja näkee, kun taas elokuvia renderöidään etukäteen jopa vuosia käyttäen renderfarmeja, ja renderöidyistä still-kuvista muodostetaan vasta elokuva. Pelihahmot ovat low poly -hahmoja verrattuna hahmoihin 3D-elokuvissa. Elokuvia varten voidaan tehdä hahmoista niin yksityiskohtaisia, kuin halutaan. Pelejä varten täytyy tehdä kompromisseja tehokkuuden ja näyttävyyden välillä.

3D-mallien tulee olla low poly -objekteja, jotta reaaliaikaisen renderöinnin/piirron laskennallinen suorituskyky olisi mahdollisimman hyvä. Low poly on kuitenkin suhteellinen termi, sillä mikään tietty määrä kolmikulmaisia polygoneja ei määrittele low ja high poly -mallien välistä rajaa. Low poly voidaan määritellä seuraavasti: milloin ja mille laitteelle malli on suunniteltu, mallin yksityiskohtaisuus sekä mallin muoto ja ominaisuudet. (Gahan 2009, 161.) Jos esimerkiksi tänä vuonna (2015) tehdyssä PC-pelissä on scenellä hahmo ja yksinkertainen oksa, oksassa on noin pari tuhatta polygonia ja hahmolla saman verran, oksan voisi mieltää olevan high poly ja hahmon todennäköisesti kuitenkin low poly -objekti.

Polygonien määrään pystytään parhaiten vaikuttamaan 3D-objektien mallinnusvaiheessa. Polygonit on hyvä hajottaa jo mallinnusvaiheessa kolmioiksi tai pitää ne maksimissaan nelikulmaisina, jotta mallin lopullinen geometria pelimoottorissa vastaa mallinnusohjelmassa luotua geometriaa. Lisäksi lopullisesta polygonimäärästä on helpompi pitää lukua, jos polygonit ovat jo mallinnusvaiheessa kolmioita. Pelimoottoriin vieminen saattaa nostaa rajustikin polygonimäärää, jos malli ei koostu kolmioista tai maksimissaan nelikulmaisista polygoneista. Kun malli tuodaan mallinnusohjelmasta pelimoottoriin,

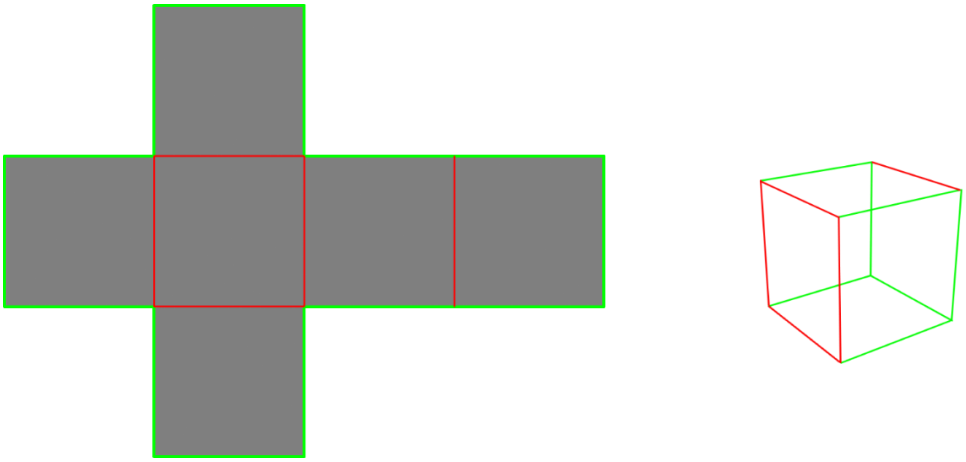
kaikki polygonit kolmioidaan (engl. triangulation) piirtoliukuhihnalla viimeistään sovellusvaiheen lopulla grafiikkakirjastorajapinnassa (esim. OpenGL, DirectX) (Silverman 2013).

Polygonit kolmioidaan, koska kolmio on yksinkertainen muoto. Kolmion kulmat eli verteksipisteet muodostavat aina yksiselitteisen tason 3D-koordinaatistossa ja sen sivut ovat aina samassa tasossa. Kolmiointi tarkoittaa käytännössä siis sitä, että jokainen ei-kolmionmuotoinen polygoni hajotetaan kolmioiksi. Kolmiointi on periaatteessa yksinkertainen prosessi, siinä luodaan uusia särmiä verteksipisteiden välille ja samalla muodostuu kolmioita. Mitä monikulmaisempi polygoni kuitenkin on, sitä enemmän on vaihtoehtoja sen kolmioimiseen. Nelikulmiot ovat siis yksinkertaisimpia kolmioitavia ja valmiiksi kolmion muotoisia tasoja ei tietenkään tarvitse kolmioida ollenkaan. Kuvio 4 havainnollistaa, kuinka monella tavalla pentagoni voidaan kolmioida. (Silverman 2013.)



Kuvio 4. Pentagonin kolmiointivaihtoehdot

Toinen geometrian optimointisääntö on, että UV-kartan saumojen lukumäärä tulee olla mahdollisimman pieni, koska UV-kartan saumat luovat teräviä reunoja objektin pintaan (Optimizing Graphics Performance 2014). UV-kartta on kaksiulotteinen kuva, joka esittää kolmiulotteisen objektin pintaa. 3D-objektin pinnasta tuotettu UV-kartta toimii pohjana mallin tekstuurille, joka tehdään kuvankäsittelyohjelmassa. Kuviosta 5 nähdään, millainen on optimaalinen kuution UV-kartta. Kuutiossa on 12 särmää, kuutiosta tehty UV-kartta sisältää 7 saumaa, eli särmien lukumäärä (12) vähennettynä UV-kartan sisällä olevat särmät (5). Kuvion 5 kuutio on myös geometrian optimoinnin ensimmäisen säännön kannalta erittäin optimaalinen; siinä on joka sivulla vain yksi polygoni.



Kuvio 5. Kuution UV-kartta

Toisinaan voi olla vaikea toteuttaa tätä UV-kartta sääntöä, koska kaikkien 3D-mallien pintaa ei voi levittää yhtenäiseksi tasoksi, kuten kuution pinnan. Pelihahmot ja mallit, joissa on erilaisia ulokkeita, ovat mahdottomia levittää yhtenäiseksi tasoksi. Saumoja tulee väkisin, jos halutaan tehdä kunnolla tasoksi levitetty UV-kartta. Ei reaalimaailmassakaan vaatekappaleita yleensä pystytä tekemään ilman yhtäkään saumaa, joten 3D-maailmassa ei pystytä levittämään virtuaalisen hahmon ihoa tai vaatteita yhtenäiseksi tasoksi.

Low poly -estetiikka

Low poly -objektit ovat usein varsin kulmikkaita, ja niistä puuttuu yksityiskohtia. Kuitenkin monet objektit voivat näyttää myös hyvältä varsin vähäisellä polygonimäärällä, kuten kuutio tai paperiarkki esitettynä vain muutamalla polygonilla. Monimutkainen objekti, esimerkiksi pelihahmo, tarvitsee useita satoja, jopa tuhansia polygoneja, vaikka kyseessä olisi low poly -objekti. Low poly -mallien karkeaa ja kulmikasta muotoa voidaan korjata bump- tai normaaalikartoilla. Kartoilla saadaan näkyviin uppoamia ja kohoumia sekä pimeämpiä ja valoisampia kohtia objektin pintaan renderöinnin jälkeen, mutta itse objekti pysyy muuttumattomana. Myös tekstuureilla (diffuse-kartat) saadaan low poly -malli näyttämään yksityiskohtaisemmalta.

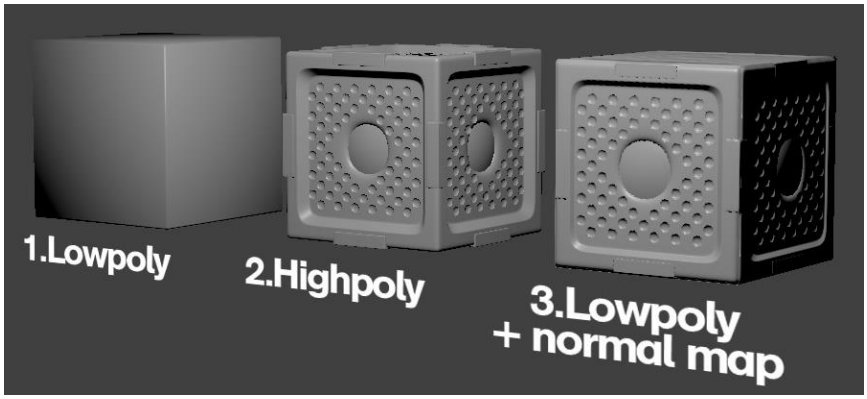
Toisaalta joitakin yksityiskohtia, kuten kynnet tai hampaat, voi olla vaikea luoda kartoilla ja niiden täytyy olla siis osa hahmon geometriaa. Monimutkainen hahmo, jolla on paljon

niveliä ja paljon luita, tarvitsee myös enemmän polygoneja, jotta nivelet ja muut yksityiskohdat ovat tarpeeksi selkeitä. Yksityiskohdat voivat lisätä harmittavasti polygonimäärää, mutta se on valinta joka täytyy tehdä tehokkuuden ja näyttävyyden välillä.

3 Pelihahmon animaation työvaiheet lyhyesti

3.1 Hahmon suunnittelu ja mallintaminen

Pelihahmon työstäminen alkaa hahmon suunnittelulla ja mallintamisella, joko erillisellä sculptaamiseen suunnitellulla ohjelmalla tai samalla ohjelmalla, jolla luodaan animaatio. Sculptaus on kuin virtuaalisen saven tai muovailuvahan muokkaamista erilaisilla ”pensseleillä” ja muilla työkaluilla. Sculptaus-vaiheessa pelihahmo on high poly -malli, josta luodaan lopuksi low poly -versio ja sille high poly -mallin avulla normaaalikartta (engl. normal map). Normaaalikartalla saadaan low poly -malli näyttämään yksityiskohtaisemmalta (ks. kuvio 6), eli normaalikartta siirtää high poly -mallin pinnanmuodot low poly -mallille. Normaalikartta ei ole pakollinen, mutta useimmille peleissä käytetyille hahmoille ja muille objekteille sellainen luodaan. Normaalikartan avulla toteutetaan GPU-optimointia; mallissa on hyväksyttävä määrä polygoneja, mutta siinä on kuitenkin haluttava määrä yksityiskohtia.



Kuvio 6. Low poly, high poly ja normaalikartan vaikutus (Normal map comparison 2014)

3.2 Riggaus

Riggaus (engl. rigging) tarkoittaa hahmon tai miksei muunkin monimutkaisemman objektin ensimmäistä esivalmisteluvaihetta ennen animointia. Hahmo, jonka on tarkoitus liikkua, tarvitsee animointijärjestelmän. Ilman animointijärjestelmää jouduttaisiin siirtämään manuaalisesti jokaista 3D-objektin verteksipistettä tai polygonipintaa jopa jokaisella framella haluttuun sijaintiin. Animaatioissa on yleensä n. 24–30 framea per sekunti. Animointi on todella paljon nopeampaa ja mallin geometria säilyttää paremmin muotonsa ja suhteensa, kun animoitavalle hahmolle on luotu rigi eli animointijärjestelmä.

Rigi on kuin luuranko, 3D-animointiohjelmissa rigin palaset onkin nimetty usein luiksi (engl. bone, bones). Pelkät luu-objektit eivät vielä välttämättä muodosta toimivaa ja animoitavaa luurankoa, vaan luille täytyy määritellä hierarkia ja luoda kontrollerit, jotka määräävät mihin suuntaan nivelet saavat taipua. Itse hahmo tai ”iho” eli rigattava malli tulee asettaa animointiohjelman koordinaatistoon niin, että se seisoo origossa ja maan pinnalla, eli (lähes) kokonaan z-akselin positiivisella puolella. Luut, tulee asetella tarkasti hahmon sisälle. Luut kannattaa luoda tai skaalata huolellisesti oikean kokoisiksi ja asettaa nivelet oikeille paikoilleen, koska se vähentää töitä skinnaus-vaiheessa.

Pelihahmon riggaus ei eroa teoriassa mitenkään animaatioelokuvien hahmojen riggauksesta, hahmolle luodaan joka tapauksessa animointijärjestelmä. Käytettävä pelimoottori ja sen vaatima tiedostomuoto kuitenkin asettavat joitakin vaatimuksia millaisen rigin hahmolle voi rakentaa, että animaatio näyttää samalta vielä tuotuna pelimoottoriin. Käytetyimpien (esim. 3ds Max, Maya, Blender) 3D-animointiohjelmien luujärjestelmät pystytään kuitenkin viemään sellaiseen tiedostomuotoon (esim. fbx), että käytetyimmät pelimoottorit (esim. Unity) niitä tukevat. Kuitenkin joitakin animaatioon ja myös riggaukseen käytettyjä työkaluja, esim. animointiohjelman uniikkeja apuobjekteja, on syytä välttää, koska niillä luotua animaatiota harvemmin pelimoottori ja sen vaatima tiedostomuoto tukevat. (What to Expect… 2015; Exchange File Formats 2015.) Tällaisia vältettäviä apuobjekteja ovat mm. 3ds Maxin helper-apuobjektit.

3.3 Skinnaus

Skinnaus (engl. skinning) on toinen esivalmisteluvaihe ennen animointia ja se sisältää mesh-objektin eli hahmon ihon kiinnittämisen luihin, jotka kontrolloivat mesh-objektia. Skinnaus-vaiheessa määritellään painotukset, jotka määräävät, mikä luu liikuttaa mitäkin ihon osaa ja sen seurauksena kuinka iho venyy ja muuttaa muotoaan luiden liikkuessa.

T-asento, testianimaatio ja täysin valmis geometria

Hahmon riggaus on helpompaa, jos se seisoo kädet suorana sivuille päin. Myös jalat voi olla vähän harallaan. Mahdollinen häntä, siivet ja muut osat tulisi pitää myös irti keskiruumiista, mikäli mahdollista. Raajojen ääriasennot näyttävät paremmilta, kun hahmo skinnataan T-asennossa, koska T-asennosta useimmiten ei ole pitkä matka ääriasentoihin, jolloin ihon ei tarvitse venyä niin paljoa. T-asento täytyy luoda animoitavalle hahmolle mallinnusvaiheessa, koska ennen skinnausta T-asento voi olla aika työläs luoda hahmolle, joka seisoo tai istuu sille luonnollisessa asennossa.

Skinnaus vai animointi ensin, onko järjestyksellä väliä? Kyllä on. Ennen skinnausta on hyvä olla luiden aikajanoilla jo edes jonkinlaista (testi)animaatiota, jotta näkee miten luiden liikkuminen saa ihon venymään tai supistumaan. Hyvä testianimaatio on sellainen, jossa luut käyvät niiden ääriasennoissa ja palaavat takaisin T-asentoon.

Hahmon tulee olla täysin valmis ennen skinnausta, koska jos hahmon geometria muuttuu vähänkään, jopa koko työläs skinnaustyö täytyy aloittaa alusta. Mallintaja ei saa enää työstää hahmoa siinä vaiheessa, kun aloitetaan skinnaus.

Low poly -hahmon skinnaus

Low poly -hahmoilla ei polygoniverkko ole kovin tiheä, joten skinnaus vaatii enemmän tarkkuutta ja huolellisuutta, varsinkin nivelten kohdilla. Toisaalta vähäinen polygonien ja siten verteksipisteiden määrä tuo mahdollisuuden säätää jopa jokaisen verteksin painotusta yksitellen, mutta skinnaukseen ei silti kulu aivan järjettömästi aikaa.

3.4 Animaation suunnittelu ja animointi

Osa hahmoanimaatiosta luodaan 3D-mallinnus-/animointiohjelmassa ja osa animaatioista vasta pelimoottorissa. Peliä varten animoitavan hahmon pivot, eli massakeskipiste tulee pysyä koko ajan paikallaan animointiohjelman koordinaatiston origossa, tai aivan sen tuntumassa, koska varsinainen liikkuminen hoidetaan pelimoottorissa. Hahmo kävelee, juoksee, ui ja lentää siis paikallaan animointiohjelmassa ikään kuin juoksumatolla. Massakeskipiste ei saa liikkua hahmon viemää tilaa kauemmas origosta, koska se voi aiheuttaa erikoisia tilanteita myöhemmin pelimoottorin puolella. Jos hahmo esim. kävellessä liikkuu jo mallinnusohjelmassa eteenpäin ja animaatio viedään pelimoottoriin, hahmo liikkuu kyllä mutta se saattaa liikkua tuplanopeutta ja sitten palata yhtäkkiä kymmenen askelta taaksepäin ja aloittaa alusta.

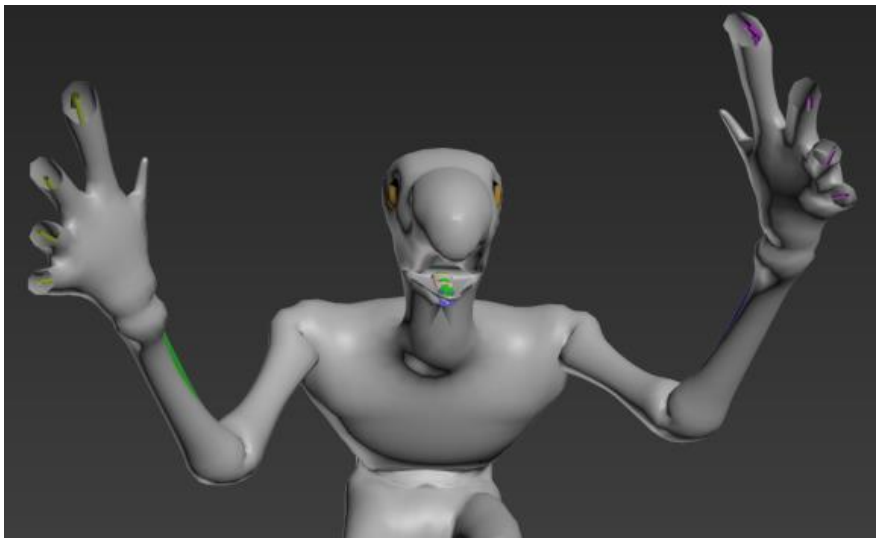
Toistettavat ja ”purkitetut” animaatiot

Muisti on rajallista, joten peleissä käytetään paljon toistoa. Toistettavat animaatiot (engl. looping animations), esim. kävelysykli, ovat lyhyitä animaatioita, joita toistetaan pelissä perä jälkeen pidemmän animaation tuottamiseksi. Toistettavat animaatiot ovat yleensä sellaisia, että niitä voidaan toistaa loputtomasti ilman että niistä huomaa missä animaatio alkaa uudestaan tai loppuu. Canned animation, eli suoraan suomennettuna purkitettu animaatio, on animaatio joka kuvaa jotain yksittäistä liikesarjaa, jota ei toisteta peräjälkeen kuten kävelysykliä. ”Purkitettuja” animaatioita voivat olla esimerkiksi aseen lataaminen tai hyppääminen. ”Purkitetut” animaatiot voivat olla myös samalla loppaavia. (Silverman 2013)

Pelimoottorin kameran vaatimukset

Pelimoottorin kamera ja sen piirtoetäisyys vaikuttavat hyvin paljon hahmon kokoon ja liikkeiden suunnitteluun ennen varsinaista animointivaihetta. Mallinnusohjelman ja pelimoottorin kameran perspektiivi voi olla hyvinkin erilainen, joten hahmo ei välttämättä enää vaikuta samalta kun se on viety pelimoottoriin. Pelimoottorin kamera on yhtä kuin pelaajan silmät ensimmäisen persoonan ammuntapeleissä. Kamera ei kuitenkaan pysty näkemään kaikkea, liian lähelle tulevat objektit katkaistaan aika luonnottomasti johtuen frustumikarsinnasta (ks. luku 2.4.1). Alla oleva kuva (ks. kuvio 7) on kuvakaappaus 3ds

Maxista (mallinnus-/animointiohjelma), mutta esimerkiksi Unity (pelimoottori) katkaisee mallit samaan tapaan, tosin Unityssa ei näy hahmon luita.



Kuvio 7. Pelimoottorin kameran frustumikarsinta simuloituna 3ds Maxilla

Frustumikarsinnasta johtuva mallien katkeaminen ei kaikissa tapauksissa haittaa, mutta luonnollisuus, aitous ja tunnelma saattavat kärsiä, jos aina lähikontaktissa viholliset leikkaantuvat. Ensimmäisen persoonan peleissä frustumikarsinta on häiritsevintä, jos se pelaajaa ylipäätään häiritsee. Mutta kolmannen persoonan näkökulmasta katsottuna esim. niin, että pelaajan hahmo näkyy lantiosta ylöspäin ja pelaaja katsoo pelitilannetta pelihahmon selän takaa, tilanne on aivan toinen. Kun pelitilannetta katsotaankin hieman kauempaa, viholliset eivät leikkaannu ja niille on mahdollista luoda myös entistä parempia hyökkäysanimaatioita lähihyökkäystilanteita varten.

4 Lintuhahmoanimaation suunnittelu ja toteutus

4.1 Autodesk 3ds Max

Työssä on käytetty Autodeskin 3ds Max -ohjelman versiota 2014. Kaikki työvaiheet rig-gauksesta animointiin on toteutettu Maxilla.

3ds Max on 3D-mallintamiseen ja animointiin suunnattu ammattilaistason ohjelma. 3ds Max tarjoaa mallinnus-, animointi-, simulointi- ja renderöintiratkaisun peli-, elokuva- ja liikegrafiikka-alan tekijöille (Autosdesk 3ds Max 2014). 3ds Max on trialware- tai sha-reware-lisenssillä julkaistu, ja lisäksi joidenkin oppilaitosten opiskelijat ja opettajat saa-vat koko ohjelmiston käyttöönsä täysin ilmaiseksi, toki tietyin ehdoin ja rajoitetuksi ajaksi. 3ds Max on julkaistu vain Windows-järjestelmälle, vaikka monista muista Auto-deskin ohjelmista on saatavilla Mac-yhteensopivat versiot. Alustarajoitteisuudesta riip-pumatta Maxia on käytetty ja käytetään paljon elokuvateollisuudessa ja videopelien tuo-tannossa (History of Autodesk 3ds Max 2014).

3ds Maxiin voidaan tuoda muokattavaksi monia muiden ohjelmien tuottamia tiedostoja, mm. Autodesk Inventor (.ipt, .iam, .ipj), Wavefont Object (.obj), Adobe Illustrator (.ai) ja 3D Studio (.3ds). Myös export-toiminto onnistuu edellä lueteltuihin formaatteihin ja vielä muutamiin muihin.

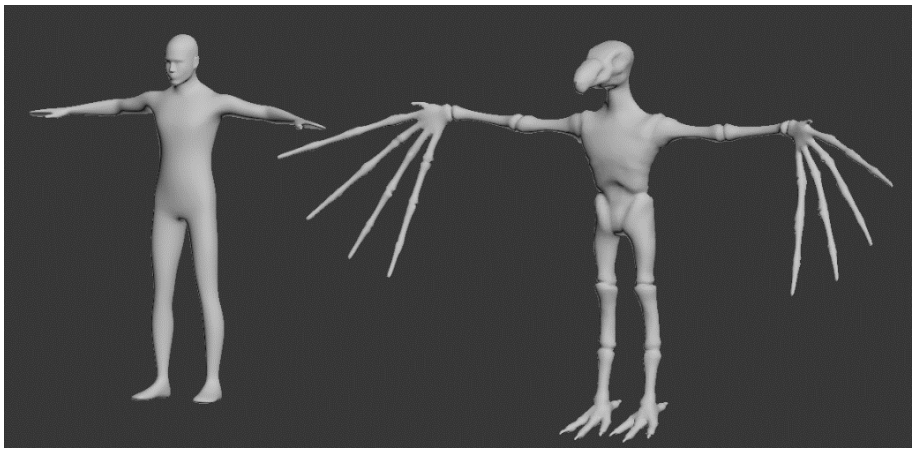
3ds Maxissa on sisäänrakennettuna Character Studio, joka sisältää työkalut hah-moanimaatiota varten. Character Studio oli aikaisemmin erilinen maksullinen lisäosa, sit-temmin se oli kokeiluversiona 3ds Maxin mukana, ja versiosta 3ds Max 7 ja vuodesta 2004 lähtien se on ollut kiinteä osa Maxia. (History of Autodesk 3ds Max 2014)

3ds Maxissa on myös oma sisäänrakennettu komentosarjakieli, MAXScript, joka on suun-niteltu täydentämään Maxin käyttöliittymästä löytyviä työkaluja. MAXScriptin syntaksi on yksinkertainen, se sisältää vain hyvin vähän välimerkki- ja muotoilusääntöjä, joten se soveltuu myös heille, jotka eivät pidä itseään ohjelmoijina. Syntaksin yksinkertaisuudesta

huolimatta scriptillä on valmiudet 3D vektoreihin, matriiseihin ja kvaternio algebraan, joten sitä voidaan käyttää ohjelmoitaessa ja simuloitaessa monimutkaisia reaalimaailman tilanteita. (MAXScript 2011.)

4.2 Lintuhahmon anatomia ja liikkeiden suunnittelu

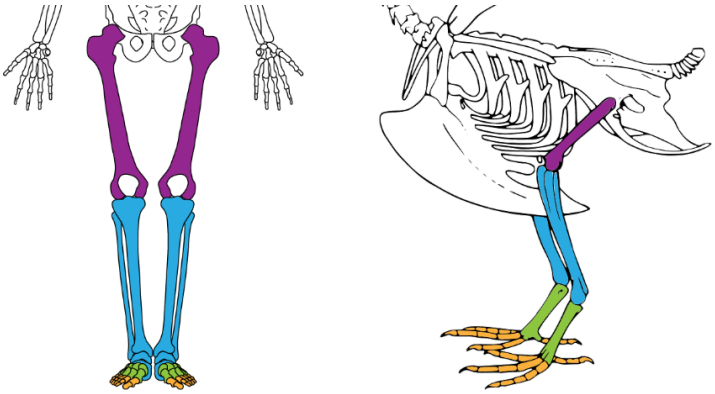
Pelin lintuhahmo on melko ihmismäinen, ainakin mittasuhteiltaan. Hahmon jalat ja selkä ovat suunnilleen samanmittaiset kuin vieressä seisovalla ihmishahmolla, kun ne on skaalattu samanpituisiksi (ks. kuvio 8). Haasteita animointiin tuovat pitkät sormet ja ylimääräinen nivel jalassa verrattuna ihmiseen.



Kuvio 8. Pelihahmo ja ihminen vierekkäin, skaalattuna samanpituisiksi

Kun verrataan linnun ja ihmisen luustoa jalkojen osalta, ehkä huomattavin ero on polven sijainti. Linnuilla polvi on paljon ylempänä ihmisiin verrattuna, useimmiten piilossa höyhenten seassa, jolloin näkyvissä olevaa nilkkaa luullaan usein linnun polveksi ja ihmetellään kuinka se lintujen ”polvi” on niin nurinkurinen. Monilla linnuilla nilkka on kuitenkin höyhenten seasta näkyvän jalan puolessa välissä, jossa ihmisellä on siis polvi. Nilkka on siinä kohtaa, koska linnuilla osa jalkapöydän luista on sulautunut nilkan luihin tai hävinnyt evoluution myötä kokonaan. Linnut ja myös monet nisäkkäät ovat varvasastujia, eli kävellessä vain varpaat osuvat maahan. Ihminen kuitenkin on puolestaan kanta-astuja. Useilla linnuilla on neljä varvasta ja ainakin yksi niistä osoittaa taaksepäin. (Bird feet and

legs 2014.) Kuvio 9 havainnollistaa ihmisen ja linnun jalan luiden erot, vastaavat luut ovat samanvärisiä.



Kuvio 9. Linnun ja ihmisen jalan luiden vertailu (Human bones 2008; Squelette oiseau 2009, muokattu)

Useimmiten linnut liikkuvat maalla hieman kömpelösti, vaikka lentävät tai uivat sujuvasti. Lintujen luut ovat ohuita ja keveitä, mutta lihakset, erityisesti lentolihakset, ovat voimakkaita. Linnuilla ei ole jalkaterissä juurikaan lihaksia, minkä takia käveleminen maalla vaikuttaa jäykältä ja nykivältä. Esimerkiksi kanat ja kyyhkyt liikuttavat päätään nykivästi eteen- ja taaksepäin kävellessä. Pään nykivä liike luultavasti tasapainottaa astuntaa ja auttaa lintua katselemaan ympärilleen. Linnut heilauttavat päätään hyvin nopeasti asennosta toiseen ja pitävät saman pään asennon hyvinkin pitkään muun ruumiin liikkeistä huolimatta. (Miksi linnut lentävät sulavasti mutta kävelevät nykivästi? 2008.)

Linnut ovat orgaanisia eliöitä, myös kyseisessä pelissä, joten niiden liikkeet eivät saa olla robottimaisia. Robottimaiset, nykivät liikkeet sopivat tietyn tyyliseen peliin ja sen maailmaan, mutta orgaaniseen ympäristöön ne eivät välttämättä sovi. Linnut toki liikkuvat hieman nykivästi, mutta liikesarjat vaativat sopivaa satunnaisuutta, jotta pelihahmot vaikuttavat mahdollisimman realistisilta ja aidoilta, vaikka ovatkin mielikuvituksen tuotteita.

Satunnaisuus ja sen tuoma luonnollisuus

Kun vihollishahmo hyökkää, hyökkäys ei saisi olla liian tunnistettava. Jos sama henkilö pelaa useamman kerran saman kohtauksen tai kentän peräkkäin, tilanne ei ehkä vaikuta enää niin uhkaavalta, jos pelaaja tietää, mistä hirviö ilmestyy ja miten se hyökkää. Hyökkäysanimaatioita olisi hyvä olla useampia erilaisia, jolloin niistä voisi arpoa randomilla, mikä niistä esitetään.

Uhkaava, pelottava hahmo

Koska kyseessä on peli, jossa on kauhuelementtejä, hahmon on tarkoitus olla pelottava. Pelottavuus on aina pelaajan ja katsojan silmissä, mutta hahmon liikkeet tulee suunnitella kuitenkin mahdollisimman uhkaaviksi. Pelottavuuteen vaikuttaa liikkeiden lisäksi kuitenkin myös hahmon ulkonäkö, vaaleanpunaiseen kumipukuun puettu hahmo ei todennäköisesti ole yhtä pelottava, kuin luurankomainen hahmo josta roikkuu mätäneviä nahan suikaleita. Sopivat liikkeet tuovat hahmon olemukseen uhkaavuutta ulkonäön ollessa vastenmielinen ja inhottava. Myös ympäristö ja äänet vaikuttavat siihen, miten pelottavan oloisia hahmot ovat. Hahmo, joka todennäköisesti vaikuttaa pelottavalta, on pelaajaa suurempi ja ilmestyy yhtäkkiä nurkan takaa päästäen ehkä jonkun äänen ja lähtee sen jälkeen seuraamaan pelaajaa.

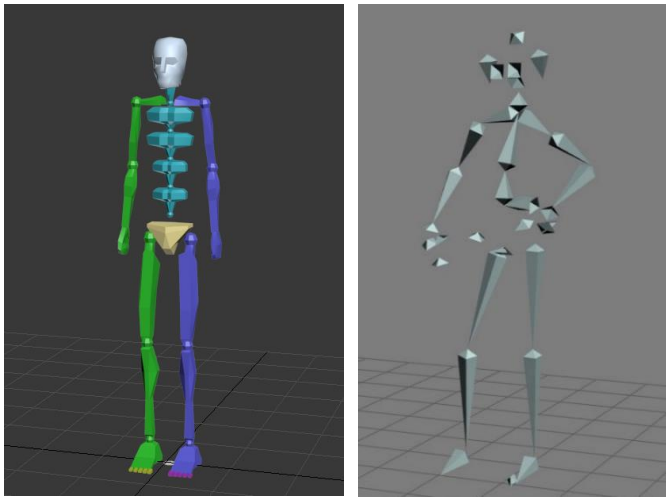
Vaikka liikkeet olisivat kuinka pelottavia, hahmo ei todennäköisesti ole pelottava, jos se on pienempi kuin pelaaja, joten hahmo on skaalattava viimeistään pelimoottorissa pelaajaa suuremmaksi. Onneksi hahmoja ja niiden liikkeitä voidaan vielä jonkin verran muokata sen jälkeen, kun hahmot on viety pelimoottoriin.

Animointivaiheessa on keskityttävä luomaan hahmolle liikesarjoja, jotka toivottavasti näyttävät uhkaavilta pelissä. Pelimaailma ei kuitenkaan vastaa todellisuutta, joten pelaajan näkökentän ulkopuolelta tulevat hyökkäykset ovat oikeastaan melko turhia. Ensimmäisen persoonan ammuntapeleissä esimerkiksi takaapäin tulevat hyökkäykset eivät tarvitse hienoja animaatioita, koska pelaaja ei huomaa takaa tulevaa hyökkäystä, jos siitä ei anneta tietoa tekstinä ja/tai äänenä.

4.3 Riggaus

Riggaus 3ds Maxissa

3ds Maxissa on monia vaihtoehtoja luoda animointijärjestelmiä, hahmojen kanssa ja erityisesti peleissä kaksi käytetyintä ovat kuitenkin Biped ja Maxin luujärjestelmä (Bones). Autodeskin ja Unityn keskustelupalstoilla ja muilla alan foorumeilla alan ammattilaiset ja harrastajat väittelevät kumpi edellisistä on parempi tapa tehdä hahmolle rigi. Kumpikin tapa on varmasti hyvä riippuen animaation käyttötarkoituksesta, mutta molemmissa on myös omat vähemmän hyvät puolensa. Suurin ero Bipedin ja luujärjestelmän välillä on työnkulku ja työkalut. Lopullisesta animaatiosta ei välttämättä osaa sanoa kumpaa riggaustapaa animaation teossa on käytetty. Rigiä ei useimmissa tapauksissa renderöidä ollenkaan, koska se on vain apuväline, jolla liikutetaan varsinaisia objekteja/hahmoja. Kuviosta 10 nähdään kuitenkin, kuinka Biped ja luujärjestelmää käyttäen tehty rigi eroavat ulkonäöltään. (Understanding Biped 2013; Bones System 2014)



Kuvio 10. Biped ja ihmishahmolle rakennettu luujärjestelmä (Bones 2012)

Luujärjestelmä

Luujärjestelmä on parempi vaihtoehto kustomoituja rigejä varten, kuin Biped. Luujärjestelmää käyttämällä rigi luodaan pala palalta, eli jokainen luu luodaan yksitellen. Luut linkittyvät toisiinsa, mutta rigi ei oletuksena sisällä mitään rajoituksia, mihin suuntaan nivelet saa taipua. Luujärjestelmän pystyttämiseen kuluu huomattavasti enemmän aikaa, koska sille täytyy rakentaa kontrollerit, joilla luita liikutellaan. (Bones System 2014)

Vaikka Bipedista saa muokkaamalla luurangon muillekin kuin kaksijalkaisille hahmoille, on ehkä kuitenkin jopa nopeampaa tehdä rigi käyttäen luujärjestelmää, jos jalkoja on enemmän kuin kaksi. Luujärjestelmää käyttäen saa juuri sellaisen rigin kuin hahmo tarvitsee: luiden lukumäärää ei ole rajoitettu, toisin kuin Bipedilla voi olla esimerkiksi vain viisi varvasta ja niissä vain kolme luuta. Bipedin ja luujärjestelmän yhdistelmä on myös mahdollinen, jos Bipedia käytettäessä tarvitaan lisää luita yksityiskohtien animointiin, kuten ilmeiden luontiin.

Biped

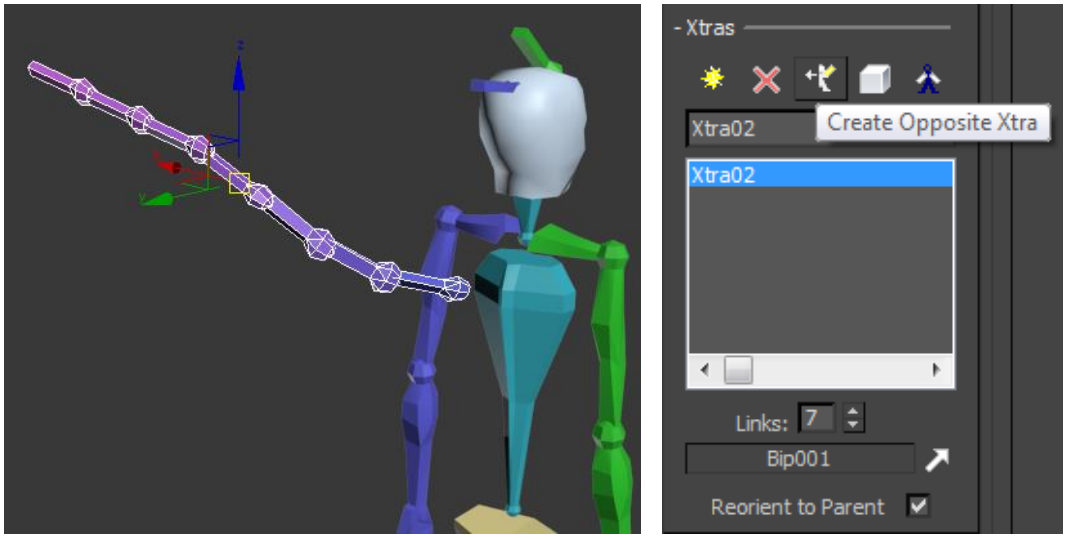
Biped on 3ds Maxin oma uniikki työkalu, se on helppo ja valmis nukkemainen rigi, ja sen saa luotua parilla klikkauksella scenelle. Sillä on monia ominaisuuksia, jotka helpottavat animaattorin työtä. Se on oletuksena kaksijalkainen ja muistuttaa ihmisen luurankoa. Bipedin nivelet on saranoitu seuraamaan ihmisen anatomiaa, ja luilla on vakaa käänteiskinematiikka, mikä tuottaa esimerkiksi kämmeniä liikutettaessa luonnollisia ihmismäisiä käden asentoja. (Understanding Biped 2013.) Bipedilla on helppo luoda kaksijalkaisia, ihmisen kaltaisia hahmoja, mutta Bipedin muokkaustyökalut mahdollistavat sen, että siitä voidaan muokata vaikkapa virtuaaliselle krokotiilille luuranko.

Bipedin kustomointi

Bipedin hierarkia poikkeaa Maxin muiden systeemien hierarkiasta siten, että Bipedilta ei voi poistaa yhtäkään luuta Delete-komennolla. Ainoat vaihtoehdot poistaa luita on piilottaa tai skaalata tarpeettomat luut näkymättömiin. Jos esimerkiksi päätä yrittää poistaa, poistuu samalla koko Biped sceneltä. Bipedilla on tietyt pakolliset osat: pää, yksi

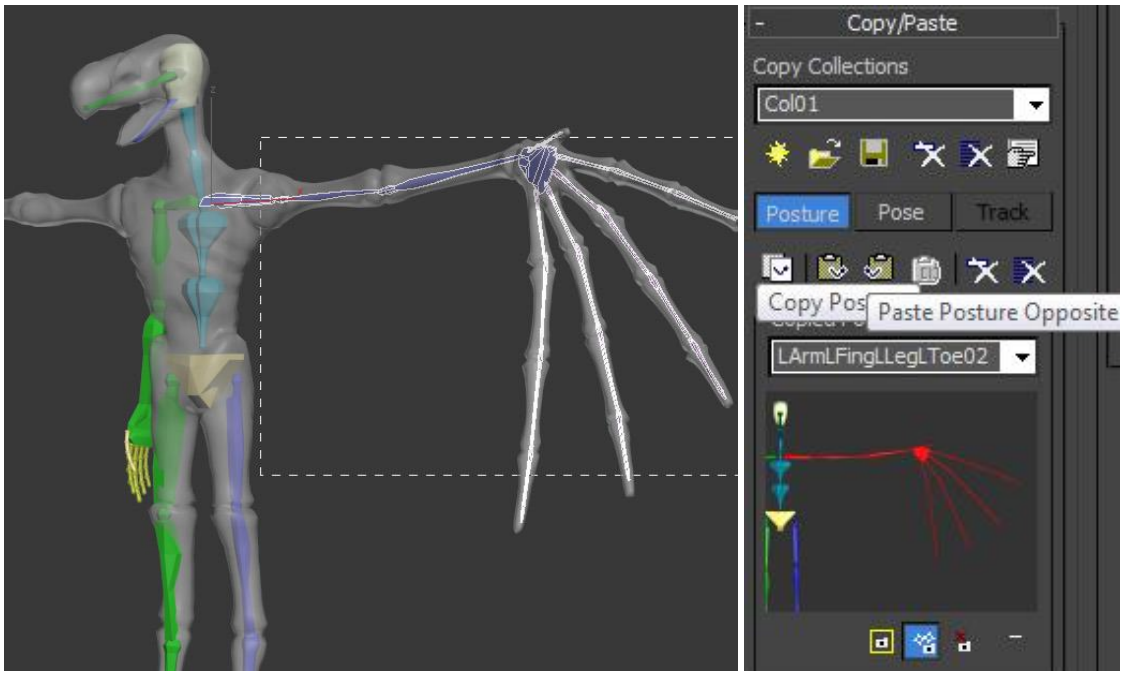
kaulanikama, yksi selkänikama, lantioluu, jalat ja yksi varvas kummassakin jalassa. Bipedia on vaikea saada rikki, mutta jos sen onnistuu hajottamaan, sitä on melkein mahdotonta korjata.

3ds Maxin Command-paneelin Motion-välilehdeltä löytyvät työkalut, joilla Bipedia muokataan ja animoidaan. Roll out -valikosta nimeltä Biped saa päälle Figure Mode -tilan, jossa luiden mittasuhteita ja lukumäärää sekä Bipedin rakennetta muokataan halutun Bipedin ollessa valittuna. Luita voi skaalata, kiertää ja siirtää rajallisesti. Lisää niveliä ja luita sormiin ja varpaisiin saa Structure-valikosta klikkailemalla. Bipedille on mahdollista lisätä häntä ja kaksi erillistä poninhäntää hiusten animointia varten. Häntä- ja poninhäntäketjuja voi siirtää juuri sinne missä niitä tarvitaan, joten niillä voi luoda luita myös vaikka sarvia ja muita erikoisia osia varten. Siivet, lisää häntiä, sarvia ja hiuslisäkkeitä voidaan luoda myös ekstrojen avulla. Structure roll out -valikossa viimeisenä ovat ekstrat (Xtras). Ekstroja voi skaalata, siirtää ja kääntää, kuten muitakin luita. Myös ekstroja saa peilattua, tai oikeastaan kopioitua peilikuvana vastakkeiselle puolelle Create Opposite Xtra -toiminnolla, kuten kuviosta 11kuvio 11 voidaan nähdä.



Kuvio 11. Ekstroilla voidaan luoda Bipedille esimerkiksi siivet.

Jos hahmo on symmetrinen, Bipedia käyttäessä ei tarvitse asetella ja mahdollisesti skaalata kuin toisen puolen raajat. Muotoilut saa peilattua toiselle puolelle, mikä nopeuttaa riggausta todella paljon. Vaikka hahmo ei olisikaan kuin osittain symmetrinen, nopeuttaa peilaus siinäkin tapauksessa työtä. Sormien luiden asetteluun voi mennä aikaa enemmän kuin tovi. Kuvio 12 havainnollistaa, kuinka peilataan käden asento toiselle puolelle. Ensin valitaan peilattavat luut, seuraavaksi ”Copy Posture” ja viimeisenä ”Paste Posture Opposite”.



Kuvio 12. Bipedin luiden koon ja asentojen peilaus Figure Mode -tilassa

Koska hahmo on lintumaisuudestaan huolimatta niin ihmisen kaltainen, sen luurangon sai helposti ja vaivattomasti muokattua Bipedista. Biped asetettiin 3D-maailman origossa olevan hahmon (hahmo T-asennossa) sisälle ja kaikki luut skaalattiin, käännettiin ja siirrettiin oikeille paikoilleen. Biped oli jo oletusasetuksineen aika lähellä sen lopullista muotoa. Bipedille lisättiin jalkaan yksi nivel lisää, yksi varvas käännettiin osoittamaan taaksepäin ja poninhäntäluut siirrettiin ja käännettiin osoittamaan eteenpäin nokaksi ja kie-

leksi. Kun kaikki luut olivat paikoillaan, niiden asento ja skaalaus peilattiin Bipedin toiselle puolelle, eli vain vasemman puolen raajojen luita siirrettiin ja skaalattiin manuaalisesti.

4.4 Skinnaus

Skinnaus 3ds Maxissa

Skinnaus tehdään Maxissa Skin- tai Physique-modifierin avulla, joka lisätään hahmon iho-objektin muokkainpinoon. Edellä mainitut modifierit eroavat toisistaan huomattavasti vain työnkulun suhteen. Ne tuottavat lähes samanlaisen tuloksen skinnaajan taidoista riippuen. Physique on melko yksinkertainen, kun sen käyttöön tutustuu, eikä Skin ole sekään kovin monimutkainen. Molemmat modifierit tarvitsevat toisinaan vielä kaverikseen Skin Morph -modifierin. Skin Morph -modifierilla saa toteutettua luonnolliset ääriasennot mm. polville ja kyynärpäille. (Skin Morph Modifier 2014.) Modifierien alustus osuu paremmin kohdalleen ja luiden vaikutusalueita (engl. envelopes) on helpompi säätää, kun hahmo on raajat levällään T-asennossa. Skin-modifier mahdollistaa myös (melko) symmetristen hahmojen verteksipisteiden painotuksien peilaamisen vastakkaiselle puolelle, mikä nopeuttaa työtä vielä huomattavasti lisää.

Pelimoottorin vaatimukset

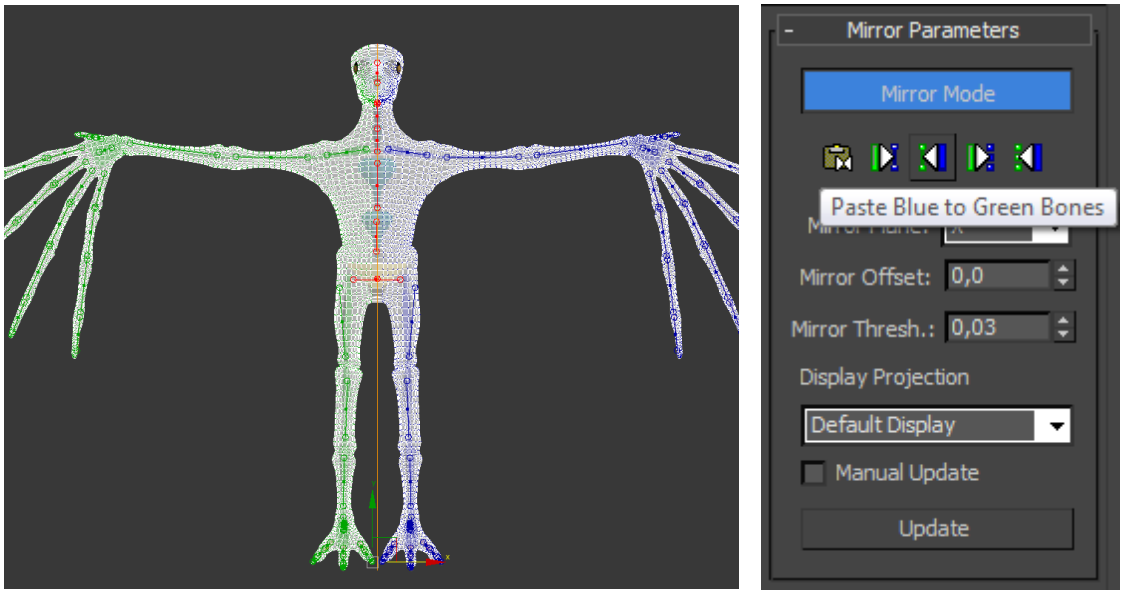
Tässä työssä skinnaus tehtiin Skin-modifierilla, koska Physique-modifierilla skinnattua hahmoa ei pysty viemään fbx-muotoon pelimoottoria varten. Vietäessä fbx-tiedostoa ulos Maxista, tulee ilmoitus, joka suosittelee käyttämään Skin-modifieria Physique-modifierin sijaan, jos Physique on sillä hetkellä käytössä. Ilmoituksesta voidaan päätellä, että hahmojen skinnaus ainakin pelejä varten täytyy tehdä Skin-modifierilla.

Skin-modifierin käyttö

Klikkaamalla Skin-modifierin Parameters-valikosta Add-painiketta avautuu ikkuna, josta valitaan kerralla kaikki tarvittavat luut ja vahvistetaan valinta. Luiden valinnan jälkeen päästään muokkaamaan luiden vaikutusalueita eli envelopeja. Envelopet ovat kapselin

muotoisia apuobjekteja, jotka kuvaavat mitkä ihon verteksipisteet kuuluvat minkäkin luun vaikutusalueeseen. Envelopeja saa skaalattua ja siirrettyä vapaasti.

Skin-modifierin käyttöä helpottaa myös peilaustoiminto. Peilaus voidaan tehdä luu- tai verteksikohtaisesti, eli peilataan joko verteksikohtaisesti tehdyt painotukset tai kaikki envelopet. Lantio, pää ja selkä- ja kaulanikamat eivät kuulu oletuksena peilattaviin envelopeihin. Myös keskiruumiin kohdalle asetellut ekstrat ja poninhännät saa pois peilauksen piiristä säätämällä Mirror Treshold -parametrin arvoa. Kuvio 13 valottaa miltä peilaustilanne näyttää.

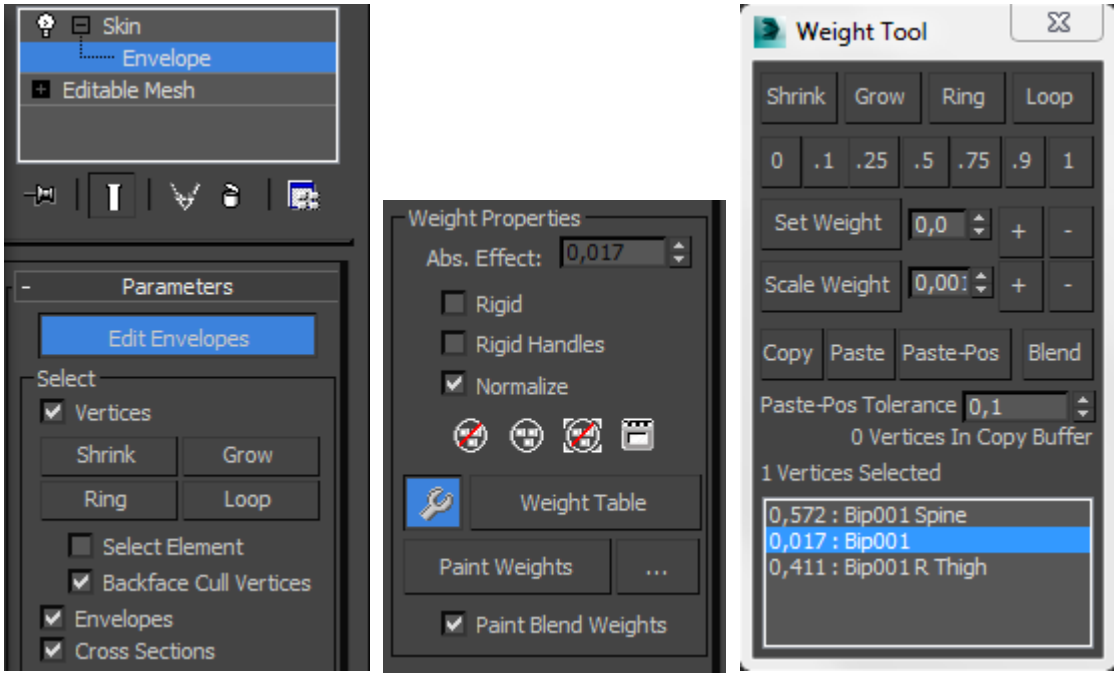


Kuvio 13. Skin-modifier ja peilaustoiminto

Raajojen ääriasennot

Polvet, kyynärpäät ja olkapäät ovat ainakin ihmishahmoilla ne hankalimmat ja eniten työtä vaativat kohdat skinnata. Lintuhahmolla on lisäksi vielä tuo ”ylimääräinen polvi”. Myös aukeava nokka vaatii omanlaisensa säädön, ennen kuin kaikki verteksit on painotettu oikein. Ääriasentoja varten voi käyttää Skin-modifierin lisänä Skin Morph -modifieria, mutta ääriasennot saadaan toimimaan myös ilman.

Hahmon jalkojen nivelien kohdalla säädettiin lähes jokaisen verteksin painotusta yksitellen, jotta ääriasennoissa iho ei mene liikaa päällekkäin. Jos nivelien verteksien painotuksia kuitenkin säätää niin paljon, että ääriasennoissa iho ei mene yhtään sisäkkäin, nivelien taitteet näyttävät samalta kuin wc-rullan hylsy, kun se taitetaan keskeltä. Lisäksi esim. reidet supistuvat olemattoman kokoisiksi ääriasennoissa, jos reiden ja säären verteksejä painotetaan liikaa. Verteksien painotusta säädetään yksitellen Skin-modifierin Weight tool -työkalulla, joka löytyy modifierin Parameters-valikosta. Painotusten muokkaamiseksi tulee olla valittuna Edit Envelopes ja Vertices (ks. kuvio 14, vasemmalla), sen jälkeen avataan itse työkalu Weight Properties -valikosta (ks. kuvio 14, keskellä). Painotustyökalulla (ks. kuvio 14, oikealla) voi asettaa valitun luun painon valitulle verteksille tai skaalata painoa, työkalun alareunassa näkyy luut, jotka vaikuttavat valittuun verteksiin.



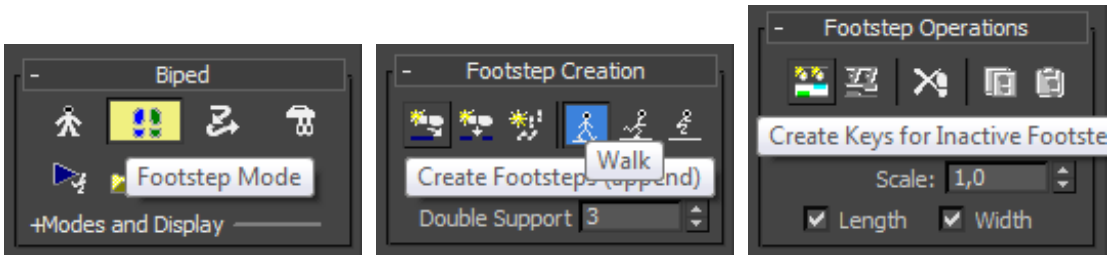
Kuvio 14. 3ds Maxin Weight Tool

4.5 Animointi

Pelimoottori ja fbx-tiedostomuoto asettavat vaatimuksia animaation toteutustavan suhteen. Kaikki animaatio ei tallennu fbx-tiedostoon, jos animaatio on toteutettu esim. käyttäen joitain Maxin omia uniikkeja animointityökaluja ja apuobjekteja. Elokuvia, tuoteesittelyvideoita ym. varten voidaan tehdä animaatio mitä mielikuvituksellisimmilla tavoilla, koska animaatio renderöidään suoraan ulos Maxista 2D-kuvina tai videona. 3ds Maxilla animoinnin perusteista on hyötyä, mutta aivan kaikkia edistyneempiä hienoja kikkoja ei voi hyödyntää luotaessa animaatiota pelihahmolle.

Bipedin automaattinen kävely

Bipedin saa luotua muutamalla klikkauksella, ja jotta sen saa myös kävelemään, ei vaadita kovin montaa klikkausta lisää. Biped roll out -valikosta saa päälle Footstep-moodin, jonka tarjoamilla työkaluilla saadaan Biped kävelemään automaattisesti. Bipedin saa kävelemään, juoksemaan tai hyppäämään, kun Footseps Creationin alta (ks. kuvio 15) on valittu haluttu liikkumistapa, esim. ”Walk”, ja sen jälkeen klikataan ”Create Footsteps”. Footsteps Operations -valikosta aktivoidaan vielä luodut askeleet ja sen jälkeen animaatio on periaatteessa valmis, jos niin haluaa. Askelia voi vielä luomisen jälkeen kääntää toiseen suuntaan tai skaalata.



Kuvio 15. Footstep moodin työkalut

Bipedille vapaasti tuotettu animaatio

Automaattisen kävelyn tuottamilla askeleilla ei saa Bipedia uimaan tai tekemään jotain muuta, mikä ei muistuta kävelyä. Luiden siirtely, kun Maxin aikajanan päässä oleva Auto

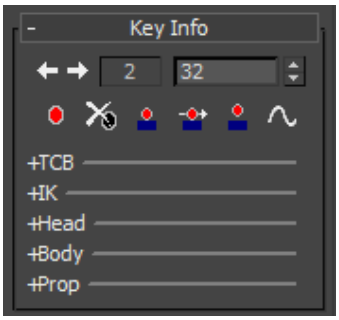
Key -nappula on aktiivinen, tuottaa aikajanalle keyframeja, joista muodostuu vapaasti tuotettu animaatio.

Bipedin lantioluun sisällä on oktaedrin muotoinen massakeskipiste, oletuksena nimeltään Bip. Massakeskipisteen avulla voidaan määrittää koko Bipedin sijainti ja asento. Bipin keyframeille tallentuu kolmenlaista tietoa liikkeistä: Body Vertical, Body Horizontal ja Body Rotation. Eli tietoa siitä, onko Bipiä siirretty xy- tai z-suunnassa vai onko sitä käännetty. (Understanding Biped 2013.)

Bipedin raajojen käänteiskinematiikka (engl. Inverse Kinematics), eli IK, käyttää kolmea parametria: IK Blend, Body/Object ja Join to Previous IK Key. IK Blend -parametri asettaa liikkeen interpolaation niin, että liike on sekoitus kinematiikkaa ja käänteiskinematiikkaa. Luku 0 tarkoittaa, että liike on täysin kinemaattinen ja 1, että liike on täysin käänteiskinemaattinen. Body/Object -parametri määrittelee IK-ketjun referenssikoordinaatin tilan: IK-ketju liikkuu Bipedin painopisteen (Bip) mukana, tai se kiinnittyy johonkin muuhun objektiin tai 3D-avaruuteen. Oletuksena referenssikoordinaatti on Body. Join to the Previous IK Key -parametri määrittelee, toimiiko edellinen keyframe seuraavan referenssinä. (Understanding Biped 2013.)

Bipedin raajojen luille voidaan luoda kolmenlaisia keyframeja (ks. kuvio 16), jotka ovat eri yhdistelmiä IK-parametreistä: istutettuja (planted keys), liukuvia (sliding keys) tai vapaita (free keys). (Understanding Biped 2013.) Keyframe-työkalut, eli Key Info (ks. kuvio 16, vasemmalla) löytyvät Command Panelin Motion-välilehdeltä Bipedin ollessa valittuna.

Prameter / Keyframe	IK Blend	Body/Object	Join to Previous IK Key
Planted keys	1	Object	On
Sliding keys	1	Object	Off
Free keys	0	Body	Off



Kuvio 16. Bipedin raajojen keyframetyypit ja niiden parametrit sekä Key Info

Kun käytetään istutettuja keyframeja, käsi tai jalka lukittuu maahan tai johonkin toiseen objektiin. Tämä johtuu Join to Previous IK Key -parametrista, joka sitoo kämmenen tai jalkaterän keyframen sen edelliseen keyframeen. Liukuvat keyframet eivät lukitse raajoja ilmaan tai maahan, vaan ne luovat viivasuoran liikeradan kahden liukuva-tyyppisen keyframen, eli raajan asennon välille. Liukuvilla keyframeilla saa esim. jalan liukumaan maata pitkin, kun luodaan hahmolle kävelysykliä. Kahden vapaan keyframen väliin muodostuu liikerata, joka voi olla jopa kurvikas, mikä johtuu IK Blend -parametrin arvosta 0.

Biped saattaa olla hieman jäykkä animoitava, johtuen juuri sen vakaasta kinematiikasta ja nivelten saranoinnista. Bipedin päätä ja lantiota ei saa lukittua paikoilleen samaan tapaan kuin sen jalat ja kädet pystyy lukitsemaan. Pää, lantio ja selkäranka eivät muodosta samanlaista IK-ketjua kuin käsien ja jalkojen luut. Lisäksi solisluu on animoitava aina erikseen, se ei kuulu käden luiden kanssa samaan IK-ketjuun.

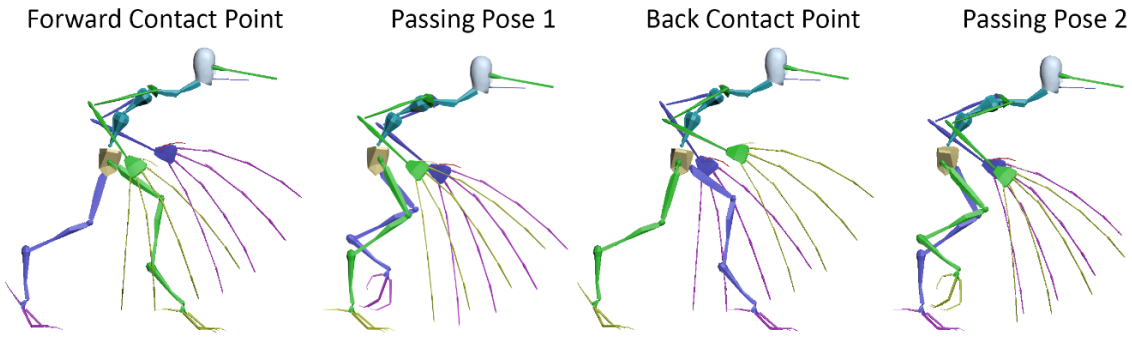
Lintuhahmon kävely ja juoksu

Jos animaatio halutaan tehdä niin, että mesh-objekti eli iho on näkyvissä, se on paras jäädyttää käyttäen Freeze-optiota ja säätää sen läpinäkyvyys noin puoleen. Iho on hyvä olla näkyvissä, jotta näkee kuinka luiden liikuttelu vaikuttaa siihen. Kun iho on jäädytetty, se ei ole valittavissa, eli siihen ei voi vahingossa koskea asetellessaan hahmon luita animaatiota varten.

Hahmon kävelysykli luotiin käyttämällä vapaata keyframe animointitekniikkaa. Bipedin automaattinen kävely ei sopinut pelihahmolle, koska se on lintu ja kävelee varpaillaan. Biped on oletuksena ihmismäiseen tapaan kanta-astuja, joten automaattinen kävely olisi vaatinut paljon muokkausta. Biped kävelee myöskin selkä suorassa, toisin kuin pelihahmo, jonka suunniteltiin kävelevän melko kyyryssä. Automaattisen kävelyn tuottamat keyfamet olisivat vaatineet useimpien luiden asentojen muokkausta jokaisella keyframella, joten kävelyanimaatio oli siis nopeampi luoda tyhjästä. Aivan tyhjästä sitä ei kuitenkaan luotu, sillä referenssinä käytettiin hahmon jalkojen osalta strutsin kävelysykliä, vaikka strutsin ja pelihahmon anatomia ja mittasuhteet eivät olekaan yhtäläiset.

Kävelysykli voidaan kaikessa yksinkertaisuudessaan esittää neljällä keyframella: Forward Contact Point, Passing Pose 1, Back Contact Point ja Passing Pose 2 (How to Create Walk

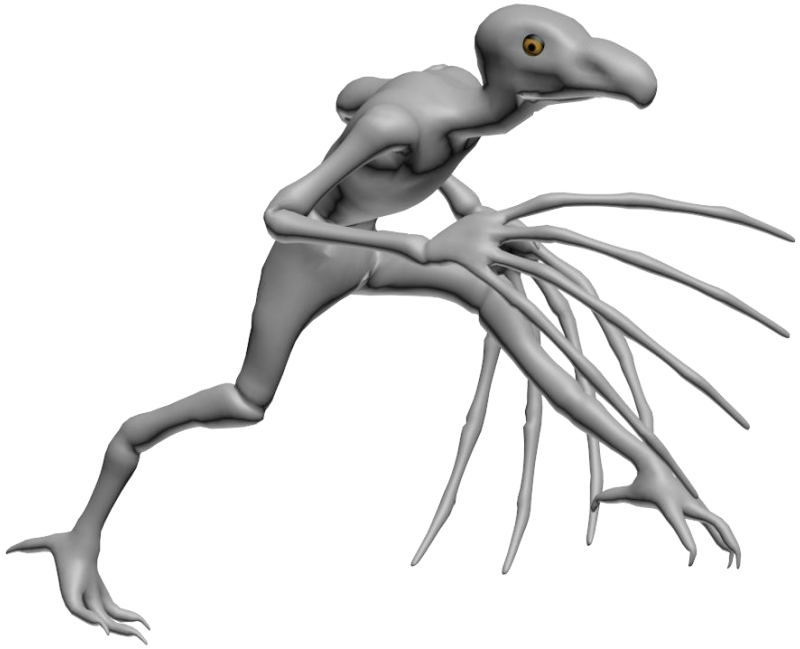
Cycle 2013). Kuten kuviosta 17 näkyy, Forward Contact Point ja Back Contact Point ovat toistensa peilikuvia, samoin Passing Pose 1 ja Passing Pose 2. Koska asennot ovat toistensa peilikuvia, Bipedin luiden peilaustoiminnolla sai kopioitua ensimmäisen/toisen keyframen asennon ja liitettyä sen peilikuvana kolmannelle/neljännelle keyframelle.



Kuvio 17. Pelihahmon kävelysykli esitettynä neljällä keyframella

Pelihahmon kävelyanimaatio on 40 framea pitkä, framella 0 on Forward Contact Point, framella 10 on Passing Pose 1, framella 20 on Back Contact Point ja framella 30 on Passing Pose 2. Jotta kävelyanimaatiosta tulee loopattava, ensimmäisen ja viimeisen keyframen tulee olla samat, eli framella 40 on myös Forward Contact Point. Näiden neljän keyframen väleissä Max interpoloi automaattisesti väliarvoja. Väliarvot, tässä tapauksessa jalkojen asennot, voi myös määritellä itse, jolloin saadaan aikaan todennäköisesti paremman näköiset liikeradat.

Juoksu on nopeampi versio kävelystä. Lisäksi hahmo ojentaa jalkansa paljon pidemmälle juostessaan ja lantio tekee laajempaa liikettä. Alla olevassa kuvassa (ks. kuvio 18) on yksi juoksun keyframeista, Back Contact Point.

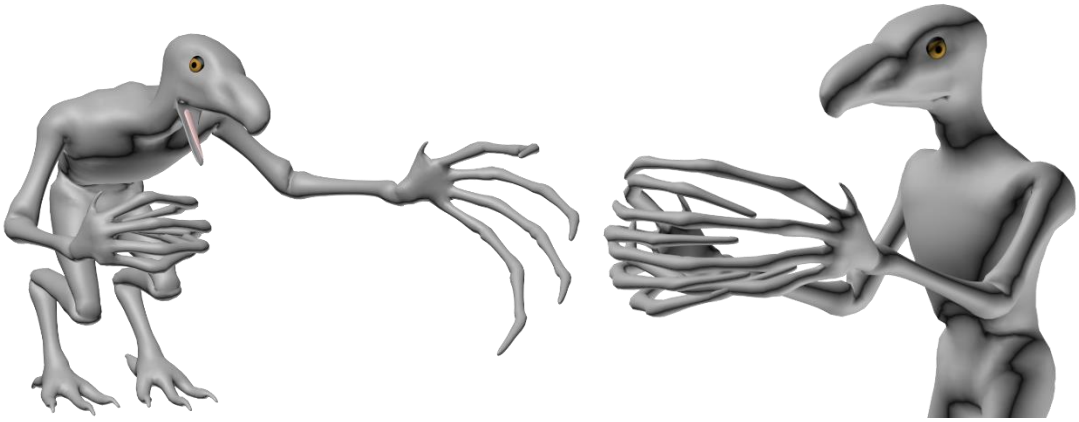


Kuvio 18. Hahmon juoksu

Hyökkäysanimaatiot

Hahmo on luonteeltaan väkivaltainen erakko ja vaeltaja, joka hyökkää kaikkien tunkeilijoiden kimppuun säälimättä. Se puolustaa omaa aluettaan raivokkaasti ja tappaa tunkeilijat salamannopeasti. Hahmo saattaa myös repiä puita juurineen irti maasta ja heitellä kiviä tunkeilijoita kohti, kuristaa pitkillä sormillaan tai potkia ja raapia jaloillaan.

Alun perin lintuhahmon nokka ei ollut suunniteltu aukeavaksi, mutta suunnitelmaa ja mallia muokattiin, jotta hahmoon saatiin enemmän eloa. Lähihyökkäystilanteessa hahmo avaa nokkansa ja huitaisee nopeasti kädellään. Alla olevassa kuvassa (ks. kuvio 19) on lähihyökkäystilannetta varten suunniteltu huitaisu ja kuristusote, jota voi käyttää myös hahmon repiessä puita.

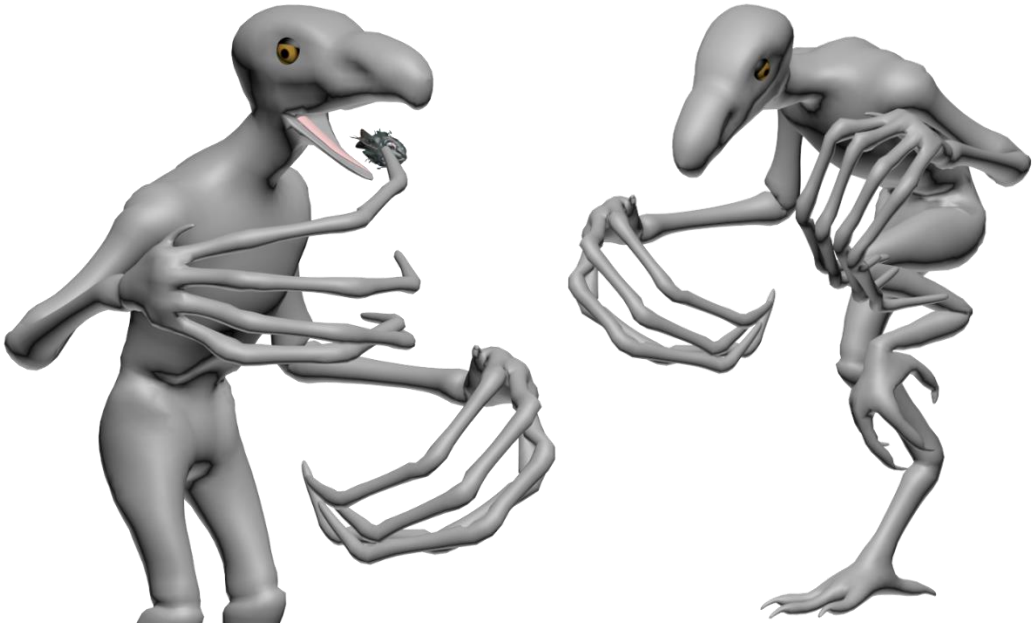


Kuvio 19. Hahmon lähihyökkäysanimaatio ja kuristusote

Hahmon muut liikkeet

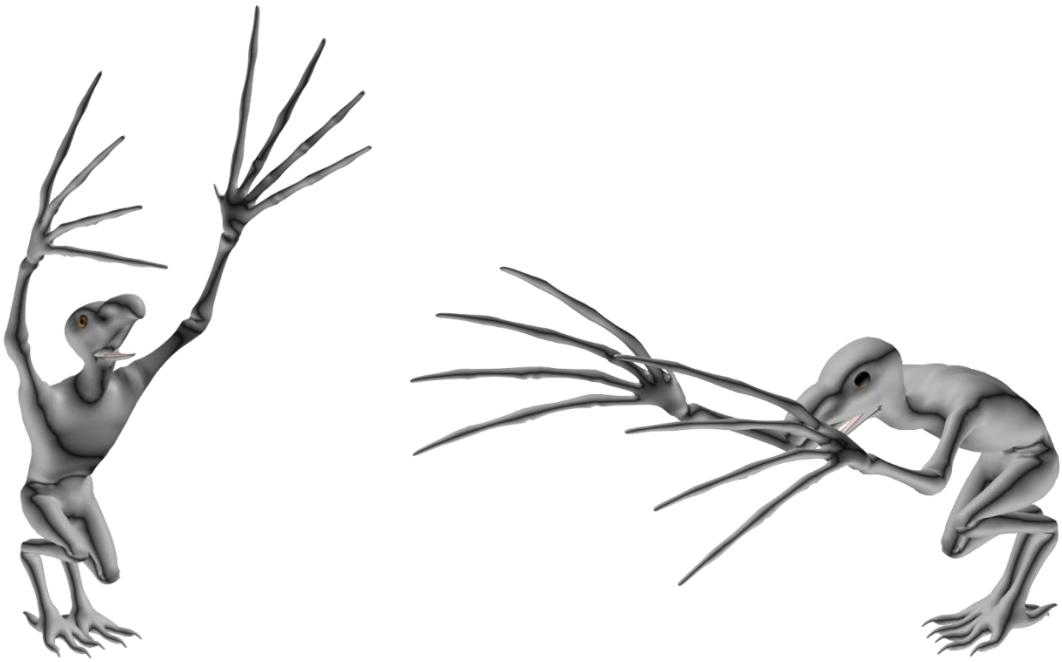
Usein pelihahmoilla on niin sanottu idle-tila, jolloin ne ovat joutilaita ja toimettomia. Idle-animaatio saa olla melko pitkä, jotta hahmon liikkeet vaikuttavat satunnaisilta. Hahmo ei voi seistä tai kyyhöttää koko ajan paikallaan täysin liikkumattomana samassa asennossa. Liikkumattomuus ei olisi kovin realistista, vaikka linnut saattavatkin liikkumatta tuijottaa samaa kohdetta hyvinkin pitkään. Vaikka pää ei liikkuisikaan, muu ruumis voi heilua edestakaisin tai lintu ojentelee jalkojaan ja siipiään.

Hahmon idle-animaatiot ovat lähinnä ympäristön tarkkailua, mutta yksi niistä on kalastaminen (ks. kuvio 20). Hahmo seisoo yhdellä jalalla vedessä ja tarkkailee kalaparvia, ja yksi epäonninen kala joutuu lintuhahmon saaliiksi. Hahmo iskee pitkän sormensa kalan läpi ja hotkaisee saaliin suihinsa. Pelimoottorissa toteutetaan kalaparvet, mutta animointiohjelmassa luotiin sen yksittäisen saaliiksi joutuvan kalan liikerata, jolloin kala on siis hahmon sormessa kiinni.



Kuvio 20. Kalastaminen

Kaikki hahmolle tehdyt liikesarjat ovat looppaavia ja idle-animaatioita voidaan kutsua myös ”purkitetuiksi” animaatioksi. Yksi esimerkki jälkimmäisestä on voitontassi (ks. kuvio 21).



Kuvio 21. Voitontanssi

Silmät ja kieli

Hahmon silmät tehtiin sisäkkäisistä palloista. Pienemmän pallon pinnassa on varsinaisen silmän tekstuuri (diffuse map) ja isompi pallo toimittaa silmäluomen virkaa. Isommat pallot ovat oikeastaan puolipalloja, joita käännetään itsensä ympäri luoden illuusio oikean silmäluomen sulkeutumisesta. Kaikki pallot on linkitetty hahmon päähän (Bipedin pää), jotta ne liikkuvat hahmon mukana.

Hahmolle tehtiin myös kieli, ja kieltä varten Bipedille lisättiin toinen poninhäntä ja siihen neljä niveltä. Kielelle lisättiin Skin-modifier ja modifierin parametreistä on määritelty, että Bipedin toinen poninhäntä toimii sen luina. Kielen animointi olisi voitu toteuttaa toisinkin, mutta fbx-tiedostomuoto ei tue kuin varsinaisia luu-objekteja (Biped tai Bones) Skin-modifierin kanssa, joten Bipedin poninhäntä oli ainut ratkaisu tehdä kielestä helposti animoitava.

Kasvonilmeet

Hahmo on aidompi, kun sillä on erilaisia ilmeitä. Tämän työn lintuhahmolle ei ollut tehty normaalikarttaa mallinnusvaiheen lopuksi, joten vain hahmon pää sculptattiin uudestaan ja luotiin sille samalla normaalikartta. Normaalikartan vaikutusta, tehoa ja toimintaa testattiin kun hahmolle luotiin muutama erilainen ilme Morpher-modifierin avulla.

Ilmeiden luomiseksi päästä luotiin muutama ”fyysinen” kopio ja kopioille luotiin erilaisia ilmeitä siirtelemällä verteksipisteitä. Verteksipisteiden lukumäärä täytyy pysyä samana kaikissa kopioissa, kuin myös alkuperäisessä päässä, muuten modifier ei toimi. Alkuperäistä päätä ei muokattu, vaan sen muokkainpinoon lisättiin Morpher. Modifierin parametreista, Channel Parameters ja Channel List -valikkojen alta valittiin mitkä ilmeet/kopiot ovat milläkin ”kanavalla”, eli määriteltiin Morph Targetit.

4.6 FBX-export pelimoottoria varten

Valmis animoitu hahmo viedään ulos Maxista Export-toiminnolla. Export-vaiheessa Max ehdottaa oletuksena tiedostomuotoa fbx. Varsinaisessa Export-ikkunassa valitaan vielä joitain asetuksia, mm. mitkä framet sisällytetään, sisällytetäänkö kamera ja valot sekä

huomioidaanko Skin-modifierin olemassaolo. 3D-mallin geometria voidaan myös jo tässä vaiheessa kolmioida. Tässä vaiheessa voidaan myös valita mitä yksikköä käytetään, eli käytetäänkö metrijärjestelmää vai brittiläistä yksikköjärjestelmää. Animoidun hahmon tai muun objektin kannalta tärkein valinta on, että sisällytetään animaatio. Tärkeä valinta on myös koordinaatiston akselit, eli mikä akseli osoittaa ylöspäin. Pelimoottorissa ei välttämättä osoita ylöspäin z-akseli, eli valitaan oikea akseli osoittamaan ylöspäin. FBX File Format -valinta riippuu jonkin verran pelimoottorin asetuksista, mutta pyritään yleisesti käyttämään uusinta FBX-versiota (FBX export guide 2014). Juuri luodun fbx-tiedoston voi vielä avata Maxissa ja katsoa, että se näyttää siltä kuin sen pitääkin näyttää. Seuraavaksi fbx-tiedosto voidaan tuoda pelimoottoriin jatkokäsittelyyn.

Bibed ja export

Bipedin luiden mukana on jonkin verran sellaisia turhia objekteja, jota ei tarvita fbx-tiedostossa. Näitä turhia objekteja ovat kaikki Bipedin osat, joiden oletusnimessä on Nub-pääte, sekä Footsteps-objekti. Turhat objektit saa jätettyä pois kun ne jäädyttää tai piilottaa ja valitsee sceneltä sen jälkeen vain ne objektit, jotka haluaa viedä fbx-tiedostoon. Kun halutaan viedä vain valitut objektit ulos fbx-tiedostona, valitaan Maxin Application Menu -valikosta Export -> Export Selected. Valittujen objektien vieminen ei eroa muuten mitenkään edellisessä kappaleessa kuvatusta Export-toiminnosta.

4.7 Animaation jatkokäsittely Unityssa

Unityyn tuotu animaatiota sisältävä fbx-tiedosto pitää pilkkoa klippeihin, koska Maxissa kaikki animaatio on yhtenä pötkönä samalla aikajanalla. Esimerkiksi looppaavat juoksu- ja kävelyanimaatiot pilkotaan omiksi klipeikseen. Animaatio ei itsestään maagisesti pilkkoonnu Maxin export- tai Unityn import-vaiheessa, koska Maxissa ei ole mahdollisuutta pilkkoa animaatiota tai antaa nimiä eri liikesarjoille aikajanalla. Tämän takia animaatiosta on hyvä olla jokin dokumentti, missä on kerrottu milloin mikäkin liikesarja aikajanalla alkaa ja loppuu. Lisäksi animoitu hahmo skaalataan Unityssa oikean kokoiseksi, jos se kaikesta suunnittelusta huolimatta on väärän kokoinen.

Hahmon kuoleminen toteutetaan myös usein vasta pelimoottorissa johtuen pelaajan mahdollisuudesta liikkua vihollishahmojen ympärillä kolmiulotteisessa pelimaailmassa. NPC-hahmoille ei siis välttämättä luoda animointivaiheessa minkäänlaista animaatiota tilannetta varten, jossa esimerkiksi pelaaja ampuu hahmoa ja luoti osuu. Animointivaiheessa luotu ”standardikuolema” ei välttämättä näyttäisi niin kovin luonnolliselta jokaisessa pelitilanteessa. Unityssa Ragdoll Wizard -työkalulla luodaan hahmolle fysiikka, eli törmäystilanteissa hahmo reagoi oikeasta kohtaa (Ragdoll Wizard 2014). Kun kuolettava isku osuu hahmoon, hahmon luut ikään kuin sulavat ja hahmo lyyhistyy sen jälkeen maahan.

5 Animaation uudelleenkäytettävyys ja muokkaus

5.1 Uudelleenkäytettävyys ja muokkaus 3ds Maxissa

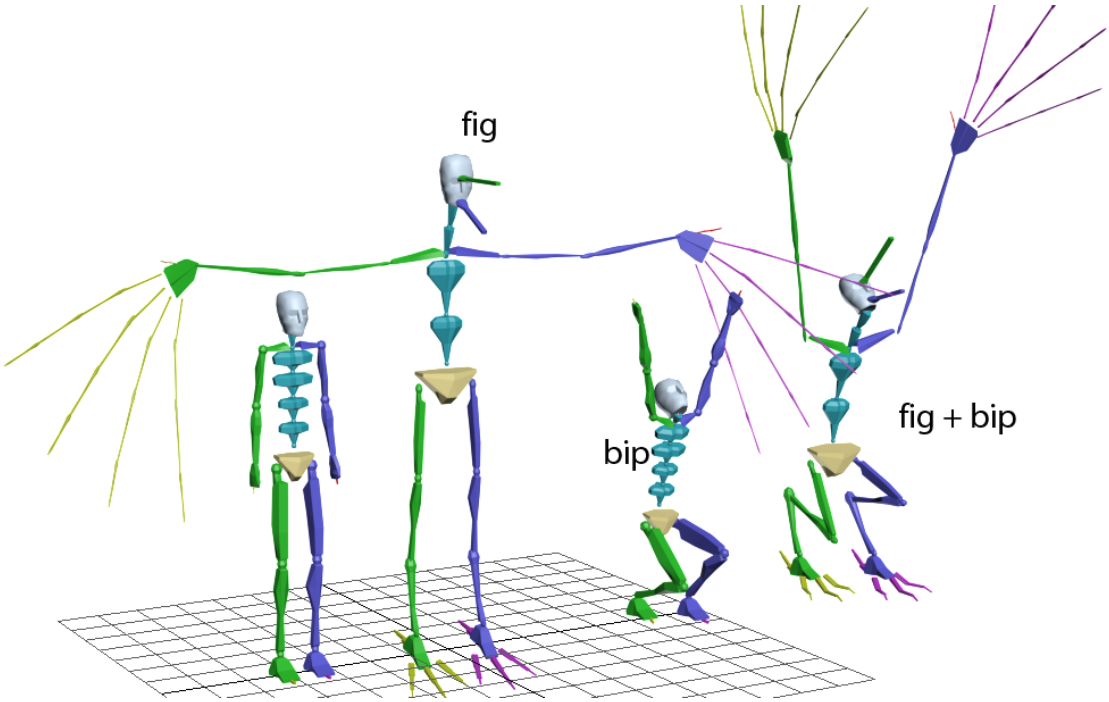
Kun kovalla työllä ja vaivalla tehty animaatio on valmis, sitä olisi mukava käyttää hyödyksi myös tulevissa projekteissa. Maxin max-tiedostoja voidaan avata teoriassa millä koneella tahansa, jolle Max on asennettuna, mutta Maxin versiointi kuitenkin estää avaamasta uudemmalla versiolla tehtyjä tiedostoja ohjelman vanhemmalla versiolla. Tämä voidaan kuitenkin ratkaista sillä, että tallennetaan tiedosto vanhemman Max-version mukaan, eli Max antaa tallentaa tiedostoja muutaman version taaksepäin. Esimerkiksi 3ds Max 2014 antaa tallentaa tiedostoja muotoon 3ds Max 2013 (*.max), 3ds Max 2012 (*.max) ja 3ds Max 2011 (*.max). Vanhemmalla versiolla luotu tiedosto avautuu kyllä uudemmilla versioilla lähes aina ongelmitta, riippuu kuitenkin kuinka vanhasta tiedostosta on kyse.

Biped ja animaation uudelleenkäyttö

Maxilla ja tarkemmin vielä Bipedilla tehtyä hahmoanimaatiota voidaan siirtää helposti Bipedilta toiselle. Samasta valikosta, josta saadaan Biped muokkaustilaan (Figure Mode), löytyy Load File ja Save File -nappulat. Bipedin rakenne ja animaatio voidaan tallentaa erilliseen fig/bip-muotoiseen tiedostoon, josta voidaan ladata uudelle Bipedille rakenne

ja/tai animaatio. Animaatio tallentuu bip-tiedostoon ja rakenne fig-tiedosoon. (Character studio File Formats 2014.) Fig-tiedosto voidaan tallentaa ja ladata vain valitun Bipedin ollessa sen muokkaustilassa ja bip-tiedosto ladataan kun Biped ei ole muokkaustilassaan.

Alla oleva kuva (ks. kuvio 22) valottaa bip- ja fig-tiedostojen eroa. Kuten kuvasta näkyy, fig-tiedosto skaalaa myös luut ja siten koko Bipedin. Kun bip-tiedosto ladataan uudelle Bipedille, sen mukana ei tule uusia luita, vaan se luo niille luille liikeradat, jotka ovat yhteisiä uuden Bipedin ja tiedostoon tallennetun Bipedin kanssa. Bip-tiedoston täydellinen lataaminen uudelle Bipedille vaatii siis sen, että sillä on samat luut, kuin tiedostoon tallennetulla Bipedilla oli tai sille ladataan ensin fig-tieodsto. Mikään ei estä kuitenkaan lataamasta vain osaa animaatiosta ja tarpeettoman animaation saa tietysti pyyhittyä pois kunkin luun aikajanalta. Tällä tavalla rakenteeltaan erilaisille Biped-hahmoille voidaan kopioida jonkun toisen hahmon liikkeet tai osa niistä.



Kuvio 22. Biped ja fig- sekä bip-tiedoston vaikutus

5.2 3ds Maxilla luodun animaation siirto muihin ohjelmiin

Pelin sisällön tuottamiseen on tähän mennessä käytetty 3ds Maxia, Mayaa ja Blenderiä. Jatkokehitystä ajatellen on hyvä olla ratkaisuja ja vaihtoehtoja, kuinka jatkaa mallien ja animaatioiden muokkausta järkevästi myös eri ohjelmilla, kuin ne on luotu. Esimerkiksi 3ds Maxilla tehdyt hahmoanimaatiot olisi hyvä saada tuotua Blenderiin muokattavaksi. Blender on alustariippumaton, täysin ilmainen ja vapaa 3D -mallinnus ja -animointiohjelmisto (Blender 2015). Tässä luvussa esitellään, mitä animaatiota voidaan viedä ja kuinka se viedään järkevästi muokattavassa muodossa Maxista Blenderiin ja toisinpäin. Lisäksi kerrotaan, miten animaatio valmistellaan vientiä varten ja mitä tulee huomioida Maxin export-vaiheessa. Testaus ja tutkimus tehtiin välillä 3ds Max 2014 ja Blender 2.73.

5.2.1 Tiedostomuodot

Mitään 3ds Maxilla tehtyä animaatiota ei voi avata suoraan Blenderissä sellaisenaan. Blender ei tue 3ds Maxin omia max-tiedostoja, niin kuin ei Max tue Blenderin blend-tiedostoja. 3D Studio (.3ds), Wavefront (.obj), Autodesk FBX (.fbx) ja Autodesk Collada (.dae) ovat kuitenkin sellaisia tiedostoformaatteja, jotka saadaan auki sekä Blenderillä, että Maxilla. Blender ja Max voivat myös suorittaa export-toiminnon edellä lueteltuihin formaatteihin. Autodeskin tiedostomuodot ovat animaation kannalta käyttökelpoisimpia, koska ne säilövät myös animaation. 3D Studio ja Wavefront ovat melko rajoittuneita formaatteja, ne pystyvät säilömään melko minimaalisen tiedon materiaaleista ja geometriasta, minkäänlaista animaatiota ei voi tallentaa. (What to Expect… 2015; Exchange File Formats 2015)

Tiivistetysti fbx tukee objektien siirtymistä koordinaatistossa ja niiden kääntymistä pivot-pisteidensä ympäri. Formaatti tukee Biped tai Bones riggausjärjestelmällä tehtyä animaatiota, missä hahmo/malli on kiinnitetty luihin Skin-modifierilla. Luihin linkitettyjen objektien liikeradat leivotaan, samoin kuin luidenkin, eli silmät ja muut erilliset objektit tallentuvat ongelmitta. Mukaan pystyy tallentamaan myös tekstuurit, mutta ei Autodeskin materiaalikirjaston materiaaleja kiiltoineen ja läpinäkyvyyksineen, kuten metalleja, lasia tai vettä.

Collada-tiedostoon voi tallentaa rigatun hahmon ja animaatiota, joka on tehty luita liikuttelemalla. Myös perus siirtymis- ja kääntymisanimaatiot tallentuu. Kaikki luihin linkitetyt objektit, esim. silmät, häviävät eli muuntuvat luu-objekteiksi, mikä ei ole käytännöllistä. Collada-tiedostoa varten kaikki objektit ja materiaalit on nimettävä niin, että niiden nimissä ei ole yhtään välejä vaan nimi on yhtenä pötkönä, esim. right_arm ei right arm.

Biped ja Maxin muut omat uniikit työkalut

Edellä mainitut yhteiset tiedostoformaatit eivät kuitenkaan säilö kaikkea tietoa oikein, kuten Collada ja luihin linkitetyt objektit. Biped ei tallennu niin, että se näyttäisi samalta edes avattuna uudestaan Maxiin sieltä tallennetusta dae- tai fbx-tiedostosta. Biped on Maxin oma riggaustyökalu, joten sitä ei saa sellaisenaan siirrettyä ja avattua muokattavaksi muihin ohjelmiin ja siksi se ei tallennu kuin max-tiedostoihin Bipedina. Biped muuttuu export-toiminnon yhteydessä ”tavallisiksi” luiksi, eli luiksi joita Maxin toinen riggausjärjestelmä, luujärjestelmä (Bones) luo. Bipedin muuntuminen on etu, koska siten sillä tehtyä hahmoanimaatiota voidaan muokata myös myöhemmin Blenderillä. Blender tunnistaa fbx- tai dae- tiedostossa olevan muuntuneen Biped-rigin luut luu-objekteiksi ja käsittelee niitä sen mukaisesti.

Helper-objekteja hyväksikäyttäen tehty animaatio ei tallennu fbx- tai dae-tiedostoon. Helper -objektit ovat nimensä mukaisesti apuobjekteja, joilla helpotetaan ja nopeutetaan jonkin verran animaation tekemistä. Tiettyjä parametreja tai verteksipisteitä ei tarvitse muuttaa/siirtää monen valikon kautta, vaan voidaan siirtää helperiä tai muuttaa parametrin arvoa slider-apuobjektin avulla suoraan viewport-ikkunassa, jossa animaatiota luodaan. Maxissa voidaan sitoa tai kytkeä toisiinsa Parameter Wiring -toiminnolla esim. apuobjekti ja jonkin objektin parametri. Apuobjektit voivat olla esim. pisteitä tai laatikoita, joita siirtämällä niihin linkitetty objekti liikkuu mukana, tai slider, jonka arvo muuttaa jonkin objektin yhtä tai useampaa parametria.

5.2.2 Import ja export

Export Maxista

Export-toiminto ei eroa oikeastaan millään tavalla pelimoottoria varten tehtävästä exportista, joka on esitelty luvussa 4.6. Valitaan sceneltä exportattavat objektit ja valitaan Export Selected tai exportataan kaikki tiedoston objektit valitsemalla Export. Formaatin valinnan jälkeen avautuu varsinainen Export-ikkuna, jossa on erilaisia asetuksia. Blenderiin vientiä varten geometriaa ei tarvitse välttämättä kolmioida. Animaatio täytyy kuitenkin leipoa, ainakin jos on käyttänyt Bipedia hahmon riggaamiseen, koska Max ei suorita exporttia loppuun, jos ei ole valinnut animaation leipomista. Animaation leipominen tarkoittaa käytännössä sitä, että jokainen objekti, jonka aikajanalla on animaatiota, saa aikajanalleen lisää keyframeja, jopa jokainen frame on siten keyframe. Keyframejen tiheyden saa haluamakseen Export-ikkunasta, mutta mitä harvemmassa niitä on, sitä vähemmän exportattu animaatio on alkuperäisen kaltainen.

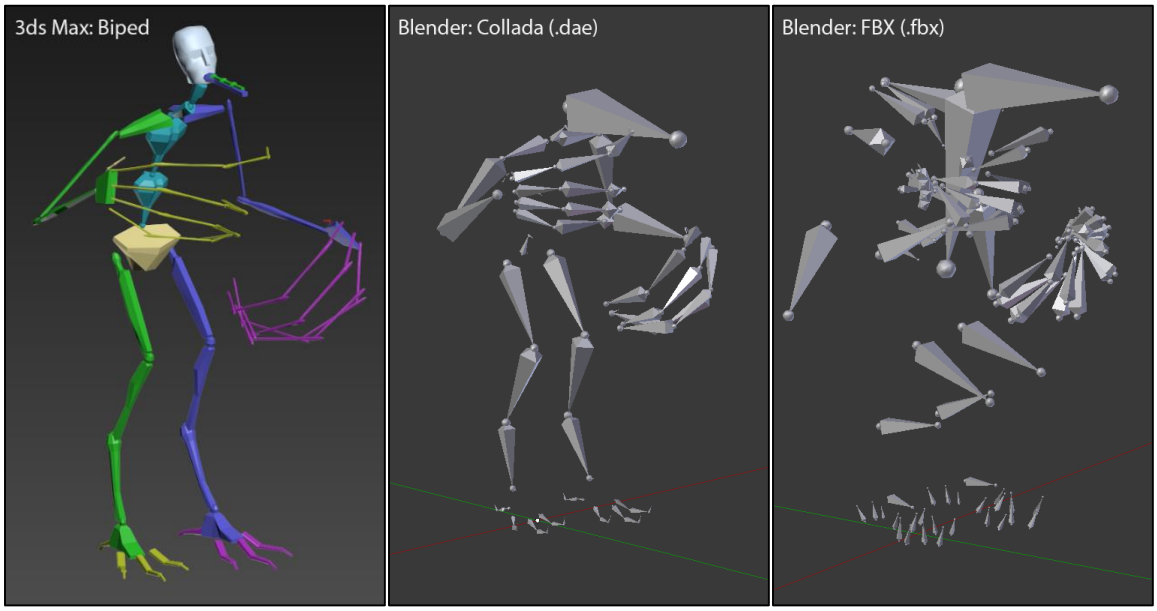
Blender Import

Tiedoston tuominen Blenderiin ei ole monimutkaista, valitaan File-valikosta Import ja haluttu tiedostomuoto, sen jälkeen etsitään avattava tiedosto ja avataan se. Jos tiedostossa on rigattu hahmo tai muu objekti joka liikkuu luiden mukaan, Blender tunnistaa luut. Molemmat tiedostomuodot (fbx, dae) tukevat Maxin Skin-modifieria, joten Blender tunnistaa, että hahmo liikkuu luiden mukaan ja hahmolla on Blenderissä myös eräänlainen skinnaus-modifier. Blenderiin tuotu animaatio on tietysti leivottu, mikä voi olla hankalaa muokkauksen kannalta, kun jokaisen objektin jokainen frame on keyframe.

Morpher-modifier kuuluu samaan kategoriaan importin/exportin kannalta, kuin Skin-modifier. Blender muuntaa Morpherin keyframet Shape Keyiksi. Blenderissä näkyy Maxissa luodut Morph Targetit, eli kopiot, mutta ne voi poistaa, koska ne ovat tarpeettomia. Kopiot voi toki poistaa jo Maxissakin tai exportata vain sen objektin, jolla on Morpher-modifier. Shape Keyt ovat objektin erityisiä keyframeja, joilla verteksit ovat eri paikoissa, joten fyysisiä kopioita mallista ei enää tarvita.

Collada (.dae) on aika rajoittunut formaatti materiaalien suhteen, esim. tässä työssä käytetty silmien tekstuuri aiheutti sen, että Maxista exportattu Collada ei aukea Blenderissä, myös tuominen uudestaan Maxiin aiheuttaa varoituksen. Colladan etu fbx-formaattiin verrattuna on kuitenkin se, että luut ovat paremmassa järjestyksessä, kun tiedosto tuodaan Blenderiin.

Alla olevasta kuvasta (ks. kuvio 23) näkyy, miltä näyttää luuranko tuotuna Blenderiin fbx- ja dae- tiedostoista verrattuna Bipediin. Ensimmäisessä kuvassa on Biped Maxissa, toisesssa ja kolmannessa kuvassa on Maxista viedyt ja Blenderiin tuodut dae ja fbx. Kaikki kuvat ovat samalta framelta, mutta kuten näkyy, viimeisessä kuvassa luut ovat hieman sekaisin. Fbx-tiedosto tuotuna Blenderiin ei näytä yhtä houkuttelevalta muokkauksen kannalta, kuin dae-tiedosto, jossa luut ovat paljon paremmassa järjestyksessä.



Kuvio 23. Maxin Biped ja Blenderiin tuodut dae ja fbx

fbx ja dae-tiedostojen tuominen Maxiin

Raahaamalla ja pudottamalla tiedostot Maxin käyttöliittymään on yksi tapa tuoda ne muokattaviksi, mutta Autodesk suosittelee käyttämään Maxin Application Menu -valikosta löytyvää import dialogia (To Import… 2015). Import dialogia käyttämällä saadaan

tiedostossa mahdollisesti olevat luut näkymään luina, eikä hankalasti muokattavina dummy-objekteina.

Blenderissä tehty Shape Keys -kasvoanimaatio voidaan exportata fbx-muotoon ja avata Maxissa. Kasvoanimaatio säilyy, koska Max tunnistaa Shape Keyt ja lisää automaattisesti pään/mallin muokkainpinoon Morpher-modifierin, jonka käyttö on hyvin samanlaista kuin Shape Keysien käyttö. Tiedoston mukana ei kuitenkaan ole fyysisiä kopioita päästä, mutta ne voi luoda Maxissa helposti modifierin Extract-toiminnolla, ne linkittyvät automaattisesti modifierin ”kavaniin”. Maxissa uusien Morph Targettien luominen tuo mahdollisuuden muokata niitä, kun ne ovat jälleen fyysisiä kopioita scenellä.

Pluginit

3ds Maxille ja Blenderille on molemmille saatavilla erilaisia lisäosia ja skriptejä, joiden avulla saadaan tallennettua tiedosto johonkin muuhun, kuin ohjelmien yleisesti tukemiin tiedostoformaatteihin. OpenCollada on yksi plugin, joka on saatavilla Maxille ja Mayalle. Se ei kuitenkaan tuonut ratkaisua luihin linkitettyjen objektien tunnistamiseen, vaan se muunsi ne luiksi Maxin Collada-exportin tapaan. Lisäksi kyseisellä pluginilla exportattu tiedosto kääntyi oudosti Blenderiin tuotuna, verrattuna Autodeskin Collada-formaattiin.

6 Tulokset

6.1 Yhteenveto

Hahmon liikuttamiseen tarvittavan animointijärjestelmän pystyttämiseen voi mennä 3D-ohjelmasta ja työkaluista riippuen muutama minuutti tai useita työpäiviä. 3ds Maxin Biped-rigi on käyttökelpoinen rigi, vaikka sitä ei ehkä arvosteta niin paljoa, kuin ”oikeata” rigiä joka on tehty pala palalta luu kerrallaan. Bipedilla saadaan nopeasti aikaan siedettävää tulosta, mutta jos haluasi ja olisi tarpeeksi aikaa, kannattaisi tehdä se niin sanottu oikea rigi, joka olisi ehkä pidemmän päälle ja jatkokehityksen kannalta kuitenkin käyttökelpoisempi.

Maxilla luodun animaation siirto muihin ohjelmiin oli työn varsinainen tutkimuskohde, mutta aihetta olisi voinut rajata tarkemmin. Bipedia ja sen animaatiota voidaan muokata ja käyttää hyväksi uusissa projekteissa erittäin helposti niin kauan kuin sitä ei tarvitse saada auki muilla ohjelmilla kuin Maxilla. Opinnäytetyön aiheena olisi voinut olla pelkästään animaatioiden siirto eri ohjelmien välillä ja muokkaus eri ohjelmassa kuin animaatio on luotu. Olisi saatu vielä parempia tuloksia ja olisi ollut aikaa esimerkiksi korjata ja sen jälkeen testata MD5 export -plugin, josta kerrotaan ongelmakohtien yhteydessä.

Olemassa olevia import/export plugineita, ainakaan kaikkia, ei päivitetä samaa tahtia kuin ohjelmia, joten niiden käyttö saattaa olla hankalaa, jollei jopa mahdotonta, jos niitä ei osaa itse päivittää. Toisaalta, mihin tarvitaan plugineja, jos (melkein) kaikki tarpeellinen tieto saadaan siirrettyä Maxin ja Blenderin välillä niiden tiedostojen avulla, joita molemmat ohjelmat tukevat. Pluginit voisivat tuoda kuitenkin yllättäviä ratkaisuja Autodeskin tiedostomuotojen vajavaisuuteen nähden.

6.2 Ongelmakohdat

Haasteita tämän työn aikana tuli vastaan, mutta ei onneksi yhtään ylitsepääsemätöntä ongelmaa. Kaikki pienemmät vastoinkäymiset ja ongelmanalut saatiin ratkaistua melko nopeasti ja päästiin jatkamaan työtä eteenpäin. Pelimoottorin ja fbx-tiedostomuodon asettamat vaatimukset tuottivat eniten niin sanotusti päänvaivaa, kun täytyi luopua ennemmin opituista animointitekniikoista tiedostomuodon vaatimusten takia.

Yhteensopimaton tiedostotyyppi

Ainut suurempi ongelma ilmaantui aivan animointiprojektin alussa. Lintuhahmo, joka oli mallinnettu Mudboxissa (Autodesk Mudbox), kaatoi 3ds Maxin, kun hahmoa koitettiin kiinnittää Maxissa sille luotuun luurankoon. Skin-modifierin lisääminen onnistui, mutta envelopejen säätäminen aiheutti virheilmoituksen ja Max kaatui. Virheilmoitus tuli myös silloin, jos skin-modifier oli aktiivinen ja laittoi animaation pyörimään. Tiedostomuodolla ei näyttänyt olevan eroa (fbx tai obj), vaan polygonien määrä (n. 7000–8000) näytti olevan Maxille liikaa. Väliaikainen ratkaisu ongelmaan löytyi, kun huomattiin, että sama lintuhahmo tuotuna Mudboxista erittäin vähäisellä polygonien määrällä (n. 2000–3000) ei

saanut Maxia kaatumaan. Lopullinen ratkaisu löytyi onneksi ajoissa. Ilmeisesti sama ongelma esiintyy muillakin Maxin käyttäjillä, sillä vastaus löytyi CGSocietyn keskustelufoorumilta. Mudboxista viety malli täytyy avata ensin Maxissa, viedä sitten Maxista 3ds-muotoon ja tuoda lopuksi 3ds siihen max-tiedostoon, jossa luut ovat. Maxiin tuotu 3ds näytti erittäin kulmikkaalta aluksi, mutta Smooth-modifier ja polygonimallinnustyökalut auttoivat pinnan silottelussa.

Pluginien yhteensopivuusongelma

Joitakin käyttökelpoisia exort/import -plugineita Maxille ja Blenderille saa ladattua katsbits.com -sivustolta. Hyödyllisimmät pluginit tämän työn kannalta olivat kuitenkin vanhentuneita, joten niitä ei voinut käyttää tai edes testata tässä työssä käytetyn 3ds Max 2014 -version kanssa. Se kaikista käyttökelpoisin plugin (3DS Max MD5 Export (+Skin)) tämän työn kannalta aiheutti Maxissa virheilmoituksen, johon ei löytynyt ratkaisua, joten pluginin käyttö jäi testaamattta. Pluginin korjailuun olisi mennyt todennäköisesti aikaa hukkaan, joten sitä ei lähdetty korjailemaan.

6.3 Jatkokehitys

Peliä varten luodun hahmoanimaation jatkokehitys tapahtuu todennäköisesti eri ohjelmalla, kuin se on luotu. Juuri jatkokehitystä varten selvitettiin mahdollisuuksia muokata animaatiota eri ohjelmalla, kuin se on luotu. Luvussa 5.2 käytiin läpi tutkimustulokset eli vaihtoehdot, miten 3ds Maxilla luotu animaatio saadaan järkevästi muokattavassa muodossa auki Blenderissä. Jotta muokkaus Blenderissä olisi järkevää, täytyy animaatiolle kuitenkin tehdä vielä joitain muutoksia Maxissa, jotta animaation saa exportattua kokonaan ulos dae-muodossa, joka avautuu Blenderissä kaikkein järkevimmin. Bipedin luihin linketetyt silmät ja kala täytyy irrottaa luista ja luoda niille omat liikeradat, tai luoda niille kaikille omat luut ja skinnata ne luihin kiinni, jotta ne eivät muutu luu-objekteiksi exportattaessa koko animaatiota dae-muotoon. Fbx-tiedoston export Maxista voi tarvita vielä lisätutkimusta, samoin sen import Blenderiin, eli miksi luut kääntyvät oudosti Blenderissä. Hahmolle täytyy myös luoda tekstuuri ja sitä ennen UV-kartta, mikä tapahtuu parhaiten Blenderissä, koska UV-kartan luominen siellä on paljon yksinkertaisempaa. On

myös mahdollista, että hahmo sculptataan vielä uudelleen, ja luodaan sille myös normaalikartta, nämäkin vaiheet tehdään Blenderissä.

Tämän työn hahmo on koko pelinkehitysprojektin aikana noussut yhdeksi tärkeimmistä hahmoista, ja sille käsikirjoitetaankin vielä lisää taustatarinaa sen lajin ja mahdollisen kulttuurin menneisyydestä, ja luodaan käsikirjoituksen pohjalta lisää animaatiota. Animaatiota luodaan lisää joko Maxissa tai Blenderissä.

7 Pohdinta

Kuten jo luvussa yksi todettiin, että kolmiulotteinen pelimaailma on kuin toinen todellisuus, se todella on sitä. Virtuaalinen, aito ja orgaaninen on äkkiseltään outo yhdistelmä adjektiiveja, mutta se on arkipäivää kiitos nykytekniikan. Mallinnus- ja aimointiohjelmat ja pelimoottorit kehittyvät kokoajan, mikä vaatii entistä enemmän tai vähemmän työtä pelin kehittäjiltä. Enemmän työtä siinä mielessä, että palaajat vaativat entistä realistisempaa tulosta, mutta toisaalta vähemmän työtä, koska tekniikan kehittyessä saadaan entistä helpommin realistisempaa jälkeä.

Orgaaniset eliöt ja nesteet ovat luultavasti niitä haastavimpia mallinnettavia ja animoitavia asioita. Kuka tahansa osaa mallintaa laatikon tai jalkapallon tutustuttuaan hetken 3d-mallinnusohjelman käyttöön, mutta orgaaniset eliöt vaativat kuitenkin paljon enemmän, kuin pelkkää mallinnustaitoa. Hahmon liikkeet ja hahmolle luotu tekstuuri vaikuttavat siihen, kuinka aidolta hahmo vaikuttaa, vaikka se olisi fantasiahahmo. Mallinnusvaiheen päätteeksi luodulla normaalikartallakin on suuri rooli pelihahmon yksityiskohtaisuuden ja aitouden kannalta. Pelihahmoille, aivan kuten elokuvienkin hahmoille, luodaan taustatarinaa ja persoonallisuus. Hahmoon täytyy eläytyä, jotta sille saa luotua persoonan ja persoonaa kuvaavia liikkeitä. Pelihahmoista voi tulla hyvinkin rakkaita ja läheisiä, varsinkin jos on luonut ne itse.

Tehtävänä oli luoda valmiiksi mallinnetulle ihmismäiselle lintuhahmolle animaatiota peliä ja sen esittelyvideota varten. Suunnittelu, suunnitelmallisuus ja suunnitelman mukaan etenemisen tärkeys tuli erittäin selväksi ja korostui tätä työtä tehdessä. On aivan

luonnollista ja inhimillistä, että suunnitelmat muuttuvat, kun projekti etenee, mutta joistakin asioista olisi silti hyvä pitää kiinni. Suunnitelmat muuttuivat työn edetessä aiheuttaen välillä pientä paniikkia, mutta alussa asetetut tavoitteet saavutettiin, ei välttämättä aivan odotetusti, mutta mitään alkuperäiseen suunnitelmaan kuulunutta ei jäänyt tekemättä. Animaatiota olisi voinut tehdä hahmolle enemmänkin, mutta päätettiin panostaa ennemmin laatuun, kuin määrään.

Opinnäytetyön tekeminen opetti paljon hahmojen animoinnista, pelinkehityksestä ja 3D-videopeleistä myös yleensä. Ongelmanratkaisutaitoja, luovuutta ja paineensietokykyä tarvittiin aika ajoin, luovuutta kaikista eniten. Animoinnin perusteet 3Ds Maxilla oli hallussa aloittaessa, mutta nyt voi sanoa, että taidot ja tietämys ovat nousseet uudelle tasolle. Enää ei tule tehtyä samoja virheitä, kuin projektin alussa tuli valitettavasti tehtyä. Nyt jos aloittaisi koko työn uudelleen, tulosta syntyisi paljon nopeammin ja tehokkaammin, kun voisi keskittyä tekemiseen eikä tekemisen opetteluun.

Lähteet

Akenine-Möller, T. Haines, E. Hoffman, N. 2008. Real-Time Rendering, Third Edition. Wellesley, MA: A K Peters.

Autodesk 3ds Max. 2014. Autodesk -sivusto. Viitattu 26.11.2014. http://www.autodesk.fi/products/3ds-max/overview

Bird feet and legs. 2014. Wikipedia-artikkeli. Viitattu 2.10.2014. http://en.wikipedia.org/wiki/Bird_feet_and_legs

Blender. 2015. Blender-sivusto. Viitattu 29.1.2015. http://www.blender.org/about/

Bones. 2012. Kuva osana dokumenttia linkAR-sivustolla. Viitattu 18.12.2014. http://arlab.com/doc/arlab/product/3d_engine/exporters/3ds_max/3.case_study/case_study_scene

Bones System. 2014. Autodesk Knowledge Network. Viitattu 27.10.2014. http://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/3DSMax/files/GUID-E6164716-CFA9-4DE9-9976-F8A58850461F-htm.html

Character studio File Formats. 2014. Autodesk Knowledge Network. Viitattu 12.3.2015. http://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/3DSMax/files/GUID-F77B03E5-268C-4DBF-B3B5-D1713336549A-htm.html

Exchange File Formats. 2015. Artikkeli TurboSquid-sivustolla. Viitattu 13.3.2015. http://support.turbosquid.com/entries/22104007

FBX export guide. 2014. Unity Manual -sivusto. Viitattu 10.2.2015. http://docs.unity3d.com/Manual/HOWTO-exportFBX.html

Features. 2015. Blender-sivusto. Viitattu 29.1.2015. http://www.blender.org/features/

Gahan, Andrew. 2009. 3Ds Max modeling for games, Insider´s guide to character, vehicle and environment modelling. Burlington: Elsevier

History of Autodesk 3ds Max. 2014. Dokumentti AREA Autodesk -sivustolla. Viitattu 27.11.2014. http://area.autodesk.com/maxturns20/history

How to Create Walk Cycle. 2013. Artikkeli CGMeetup-sivustolla. Viitattu 4.12.2014. http://www.cgmeetup.net/home/how-to-create-walk-cycle

Human bones. 2008. Kuvatiedosto Wikipediassa. 2.10.2014. http://commons.wikimedia.org/wiki/File:Human_bones.svg

MAXScript. 2011. MAXScript Help -sivusto. Viitattu 27.11.2014. http://docs.autodesk.com/3DSMAX/14/ENU/MAXScript%20Help%202012/

Miksi linnut lentävät sulavasti mutta kävelevät nykivästi?. 2008. Artikkeli Tieteen Kuvalehden verkkoarkistossa. Viitattu 26.10.2014. http://tieku.fi/luonto/miksi-linnut-lentavat-sulavasti-mutta-kavelevat-nykivasti

Normal map comparison. 2014. Kuva keskustelufoorumilla. Viitattu 12.3.2015. http://quakeone.com/forums/quake-mod-releases/finished-works/10492-simple-models-3.html

NPCs. 2014. Giant Bomb -sivusto. Viitattu 13.10.2014. http://www.giantbomb.com/non-player-character/3015-967/

Non-Player Character. 2014. Techopedia. Viitattu 13.10.2014. http://www.techopedia.com/definition/1920/non-player-character-npc

Optimizing Graphics Performance. 2014. Unity Manual -sivusto. Viitattu 3.12.2014. http://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html

Puhakka, A. 2008. 3D-grafiikka. Helsinki: Esa Print Oy.

Ragdoll Wizard. 2014. Unity Manual -sivusto . Viitattu 27.1.2015. http://docs.unity3d.com/Manual/wizard-RagdollWizard.html

Silverman, D. 2013. 3D Primer for Game Developers: An Overview of 3D Modeling in Games. Artikkeli Tuts+ -sivustolla. Viitattu 2.12.2014. http://gamedevelopment.tutsplus.com/articles/3d-primer-for-game-developers-an-overview-of-3d-modeling-in-games--gamedev-5704

Skin Morph Modifier. 2014. Autodesk Knowledge Network. Viitattu 18.12.2014. http://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/3DSMax/files/GUID-8E213537-FFD4-4B6D-A481-6E4B935448CF-htm.html

Squelette oiseau. 2009. Kuvatiedosto Wikipediassa. 2.10.2014. http://commons.wikimedia.org/wiki/File:Squelette_oiseau.svg

To Import your Customized Character in 3ds Max for Animation. 2015. Autodesk Knowledge Network. Viitattu 25.2.2015. http://knowledge.autodesk.com/support/character-generator/learn-explore/caas/CloudHelp/cloudhelp/ENU/CharacterGen-Help/files/GUID-9DDA7D10-AFFD-4FBD-9A45-96CE436DE0D8-htm.html

Understanding Biped. 2013. 3ds Max Help -sivusto. Viitattu 27.10.2014. http://docs.autodesk.com/3DSMAX/15/ENU/3ds-Max-Help/contents.html

Unity. 2014. Unity-sivusto. Viitattu 27.11.2014. http://unity3d.com/unity

Ward, J. 2008. What is a Game Engine?. Artikkeli GameCareerGuide.com -sivustolla. Viitattu 22.10.2014.
http://www.gamecareerguide.com/features/529/what_is_a_game_.php

What to Expect When You Import Exchange Formats. 2015. Viitattu 13.3.2015.
http://support.turbosquid.com/entries/22075033-What-to-Expect-When-You-Import-Exchange-Formats