TAMPEREEN
AMMATTIKORKEAKOULU

DIGITAALINEN KUVANVEISTO PELIMOOTTORIIN

Mika Sarkkomaa

Opinnäytetyö
Lokakuu 2017
Tietojenkäsittely
Pelituotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Pelituotanto

SARKKOMAA, MIKA
Digitaalinen kuvanveisto pelimoottoriin

Opinnäytetyö 39 sivua
Lokakuu 2017

Tämän opinnäytetyön tarkoituksena on käydä läpi 3D-mallin työnkulku digitaalisesta veistoksesta pelimoottoriin ja siten määrittää selkeä työnkulku tälle prosessille ja selvittää miten 3D-malleja optimoidaan pelimoottoria varten.

Jokainen työvaihe on jaettu omaan lukuunsa. Jokaisessa luvussa tarkastellaan työvaiheiden keskeisiä käytäntöjä, mitä näissä vaiheissa tulee ottaa huomioon pelimoottorin kannalta ja miten vaiheet linkittyvät toisiinsa. Työvaiheiden visualisoinnin avuksi toteutettiin pelimoottoria varten optimoitu 3D-hahmo käyttäen opinnäytetyössä esiteltyjä tekniikoita.

Digitaaliseen kuvanveistämiseen käytetty ohjelma oli ZBrush, Muut vaiheet käydään läpi keskittymättä tiettyihin työkaluihin, jotta niitä voi soveltaa mahdollisimman moneen eri ohjelmaan. Tietolähteinä käytettiin alan kirjallisuutta sekä luotettaviksi todettuja internet-lähteitä.

Opinnäytetyöprojekti onnistui hyvin sekä teorian, että mallinnuksen osalta. Monivaiheiselle mallinnusprosessille saatiin määriteltyä työnkulku, joka on selkeä ja hyödyllinen pelimoottoreihin mallintajille. Opinnäytetyössä keskityttiin manuaalisiin tekniikkoihin, mutta työnkulkua on mahdollista kehittää tulevaisuudessa automatisoiduilla tekniikoilla.

Asiasanat: digitaalinen kuvanveisto, 3D-mallinnus, zBrush, peli

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Game Development

SARKKOMAA, MIKA:
Sculpting for Digital Games

Bachelor's thesis 39 pages
October 2017

The purpose of this thesis was to go through a 3D model's workflow from a digital sculpture to a working asset in a game engine and thus set a clear workflow for this process and find out how 3D models are optimized for the game engine.

Each step is divided into its own chapter. Each chapter looks at key practices of the work steps, what should be taken into account in these steps from the perspective of the game engine and how they are linked to each other. To help visualizing these steps, a 3D character was produced using the methods presented in this thesis.

The program used for the digital sculpting part was ZBrush, other steps are presented without focusing on certain tools so that they can be applied to as many programs as possible. Background information for this study was obtained through a review of related literature and a number of reliable internet sources.

The theory part and the modeling part of this study were successful. A clear workflow of the multi-step modelling process was defined, and it is likely to be useful for people who create 3D models for games. The focus in this thesis was on manual techniques, but is possible to extend the workflow to also cover automated techniques.

Key words: sculpting, 3D modeling, zBrush, game engine

SISÄLLYS

ERITYISSANASTO

Verteksi	Piste 3D-tilassa
Polygoni	3D-mallissa oleva pinta, jonka muodostaa kolme tai neljä verteksistä
N-gon	Viiden tai useamman verteksin muodostama pinta
Topologia	3D-mallin pinnan rakenne
Highpoly-malli	3D-malli joka koostuu suuresta määrästä polygoneja
Lowpoly-malli	Reaaliaikaiseen renderöintiin tarkoitettu 3D-malli joka koostuu pienestä määrästä polygoneja
Beikkaus	3D-mallin yksityiskohtien siirtäminen toiseen malliin
Normaalikartta	Tekstuurikartta joka sisältää normaalin eli suunnan jokaiselle pikselille.
Shader	Skripti, joka laskee 3D-mallin pinnan pikselien värin
Riggaus	3D-mallin luurangon rakentaminen ja painottaminen eri vertekseille

1 JOHDANTO

Digitaalisen kuvanveiston käyttö peleihin mallintamisessa yleistyy koko ajan. Sen myötä mallintamiseen käytettävät työskentelytavat ja tekniikat muuttuvat. Ennen oli yleistä, että 3D-hahmon luomiseen tarvittiin ainakin kaksi henkilöä. Suunnittelija konseptoi ja piirsi hahmon 2D-kuvina ja sitten 3D-mallintaja teki 3D-hahmon näiden kuvien pohjalta. Ero suunnittelijan ja mallintajan välillä alkaa kuitenkin olla häilyvä. Perinteiset mallinnustekniikat eivät sovellu konseptointiin, mutta digitaalinen kuvanveisto mahdollistaa todella nopean iteroinnin, minkä ansiosta 3D-mallintaja voi toteuttaa myös suunnitteluvaiheen. Kahteen eri asiaan erikoistuneen henkilön sijaan tarvitaan enää yksi. Tämän takia digitaalisen kuvanveiston oppiminen on tärkeää 3D-mallintajille. Opinnäytetyön tarkoituksena on käydä läpi 3D-mallin työnkulku digitaalisesta veistoksesta pelimoottoriin ja siten määrittää selkeä työnkulku tälle prosessille ja selvittää mitä asioita eri työvaiheissa tulee ottaa huomioon pelimoottorin kannalta.

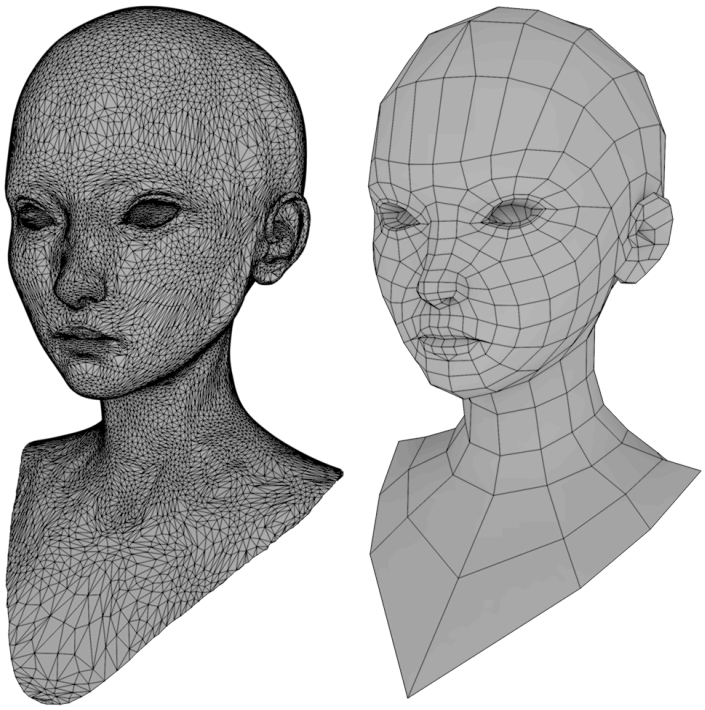
Pelit ovat uniikkeja siitä syystä, että ne ovat interaktiivisia. On mahdotonta ennustaa kaikkia mahdollisia asioita joita pelaaja saattaa tehdä, joten pelit täytyy renderöidä reaaliajassa. Reaaliaikainen renderöinti tarkoittaa kuvien generointia sitä mukaa kun peli etenee ja pelaaja suorittaa toimintoja. Peleissä renderöintiin kuluvalla ajalla on suuri merkitys, koska kuvia täytyy generoida tarpeeksi monta sekunnin aikana, jotta liike on sulavaa ja etteivät pelaajat joudu odottamaan toimintojensa tuloksia liian kauaa. Grafiikoiden monimutkaisuus on yksi vaikuttavimmista tekijöistä renderöintiajan nopeuteen. Elokuvissa ja still-kuvissa käytettävä grafiikka voi olla kuinka kompleksia tahansa, koska renderöinti tapahtuu etukäteen, eikä renderöintiin kuluneella ajalla ei ole juurikaan merkitystä. Peleihin valmistetun grafiikan pitää sen sijaan olla optimoitua nopean renderöimisen saavuttamiseksi. Tätä varten onkin kehitetty pelimoottoreita (Silverman 2013).

Pelimoottori on ohjelma, joka mahdollistaa pelien kehittämisen eri laitteille. Monet nykyisistä pelimoottoreista sisältävät 3D-renderöintimoottorin reaaliaikaiselle 3D-grafiikalle. Jotta reaaliaikainen renderöinti on mahdollista, täytyy pelimoottoreissa käytettävillä 3D-malleilla olla tiettyjä ominaisuuksia. Tärkein huomioon otettava asia on mallissa käytettävien polygonien määrä. Mitä enemmän polygoneja malli sisältää, sitä kauemmin sen renderointiin kuluu aikaa. Pelien 3D-mallinnuksessa ovat yleistyneet termit lowpoly-malli ja highpoly-malli (kuva 1). Highpoly-mallilla tarkoitetaan 3D-mallia, jossa on pal-

jon polygoneja. Highpoly-malleja ei ole tarkoitettu käytettäväksi pelimoottorissa sellaisenaan polygonimäärän ja huonon topologian takia, vaan niiden tarkoituksena on toteuttaa yksityiskohdat lowpoly-malliin. Lowpoly-malli on reaaliaikaista renderöintia varten optimoitu malli, jossa on alhainen polygonimäärä.

Opinnäytetyössä perehdytään tähän high- ja lowpoly-mallien työnkulkuun, käyttäen digitaalista kuvanveistoa highpoly-mallin toteuttamiseen. Työvaiheiden visualisoinnin avuksi toteutettiin pelimoottoria varten optimoitu 3D-hahmo, käyttäen opinnäytetyössä esiteltyjä tekniikoita. Vaikka prosessi aloitetaan veistoksesta, vaiheet UV-kartoituksesta eteenpäin pätevät myös muihinkin kuin vain veistoksesta aloitettuihin 3D-malleihin.

Opinnäytetyö on tarkoitettu kaikille, jotka ovat kiinnostuneita digitaalisesta kuvanveistosta sekä heille, jotka haluavat oppia 3D-mallien optimoinnista pelejä varten. On suositeltavaa, että lukija tietää 3D-grafiikan alkeet, sillä niitä ei selvitetä tässä opinnäytetyössä. Kuvanveistossa opastetaan Zbrushin käyttöä, koska se on alan standardiksi muodostunut sovellus digitaaliseen kuvanveistoon. Muut vaiheet käydään läpi keskittymättä tiettyihin työkaluihin, jotta niitä voi soveltaa mahdollisimman moneen eri ohjelmaan.



Kuva 1. Vasemmalla veistetty highpoly-malli, joka sisältää 32924 polygonia ja jossa on huono topologia. Oikealla lowpoly-malli, joka sisältää 565 polygonia (1054 kolmiota) ja jossa on hyvä topologia.

2 DIGITAALINEN KUVANVEISTO

Digitaalinen kuvanveisto (englanniksi sculpting) tarkoittaa 3D-mallien luomista tietokoneella siveltimien avulla. Se on hyvin samantapaista kuin perinteinen saven veistäminen, niin työkaluiltaan kuin tekniikoiltaan ja siinä käytetäänkin termiä digitaalinen savi, kun puhutaan digitaalisen veistoksen 3D-verkosta. Yleisimmin veistäminen tapahtuu piirtopöydän tai -näytön avulla niiden tarkkuuden ja paineentunnistus ominaisuuden ansiosta. (Keller 2012, 61.)

Kun kohteena on pelimoottori, digitaalisen kuvanveiston tarkoituksena on yleensä toteuttaa korkealaatuinen veistos eli highpoly-malli. Veistosta käytetään pohjana lowpoly-mallin pinnanmuotojen rakentamisessa ja lisäksi veistoksen pinnan yksityiskohdat siirretään tekstuurikarttojen muotoon, jotta niitä voidaan hyödyntää lowpoly-mallin pinnalla.

Digitaalinen kuvanveistäminen on nykyään mahdollista melkein kaikissa suosituimmissa 3D-mallinnusohjelmissa, mutta parhaimmat työkalut veistämiseen ovat kuvanveistoa varten kehitetyissä ohjelmissa, kuten Zbrush, Mudbox ja 3D-Coat. Näistä tällä hetkellä suosituin on Zbrush, minkä takia se valittiin tässä opinnäytetyössä käytettäväksi ohjelmaksi.

2.1 Digitaalinen kuvanveisto verrattuna perinteiseen 3D-mallinnukseen

Nykyään peleissä suositaan yhä enemmän digitaalista veistämistä. Sitä suositaan perinteisen 3D-mallintamisen sijaan etenkin silloin, kun malleihin halutaan paljon yksityiskohtia. Digitaalinen kuvanveisto tarjoaa intuitiivisemman tavan luoda 3D-malleja kuin perinteiset 3D-mallinnustekniikat, joissa polygoneja ja verteksejä liikutellaan yksitellen ja käytettävä topologia on lyötävä lukkoon heti alkuvaiheessa. Artistit voivat keskittyä enemmän mallintamisprosessiin 3D:n teknisten aspektien sijasta.

Perinteisesti 3D-mallinnus vaatii paljon teknistä osaamista. Ongelmana tässä on se, että graafikot keskittyvät mieluummin mallinnuksen taiteelliseen puoleen. Digitaalisessa kuvanveistossa ei tarvitse ottaa huomioon mallin topologiaa muutoksia tehdessä. Siinä ei oikeastaan muokata mallin pinnan rakennetta lainkaan, eikä lisätä sitä (joitain työkaluja

lukuun ottamatta), vaan todellisuudessa ainoastaan muutetaan sen muotoa. Koska topologiaan ei tarvitse kiinnittää huomiota, digitaalinen kuvanveisto mahdollistaa intuitiivisemman työskentelytavan kuin perinteinen 3D-mallinnus ja mahdollistaa helpomman iteroinnin myös myöhäisessä työvaiheessa.

Digitaalinen kuvanveisto soveltuu etenkin hahmojen mallintamiseen, sillä ne vaativat paljon orgaanisia, epätasaisia muotoja ja paljon yksityiskohtia, joita on perinteisin 3D-mallinnustekniikoin vaikea saavuttaa. Perinteinen 3D-mallintaminen puolestaan soveltuu paremmin keinotekoisten, geometrisesti täydellisen muotoisten, kovapintaisten ja suoralinjaisten objektien mallintamiseen.

2.2 ZBrush

ZBrush on vuonna 2000 julkaistu digitaalinen kuvanveistoon- ja maalausohjelma (Keller 2012, 61). ZBrushin käyttö on yleistynyt pelialalla viime vuosien aikana ja siitä onkin jo kehittynyt alan vakiotyökalu digitaaliseen kuvanveistoon. Suositun ZBrushista tekee se, että ZBrushin työkalut mahdollistavat tehokkaan ja epälineaarisen työskentelytavan. Suuri osa Zbrushin työkaluista jäljittelee perinteistä savenmuovaamista ja se tekeekin Zbrushista hyvin helposti lähestyttävissä ja luonnollisen kokemuksen (Pixologic 2017a).

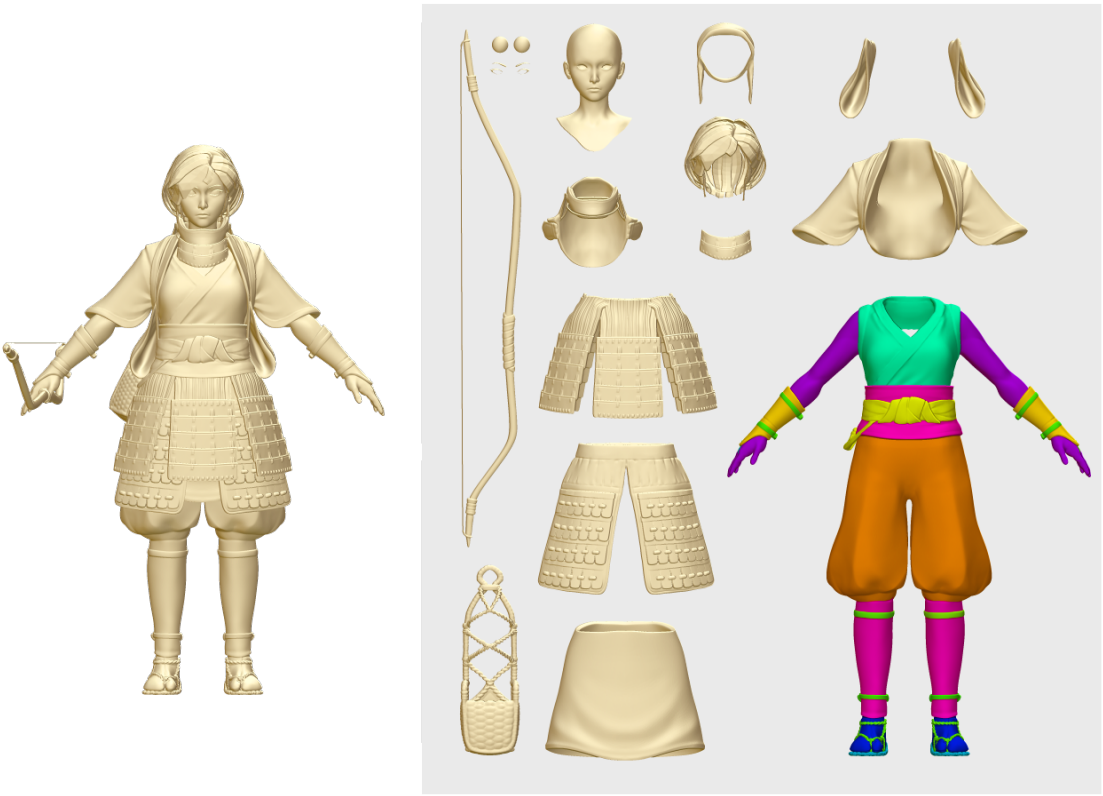
ZBrushia käytetään pelien lisäksi kaikilla muilla viihteenaloilla sekä monissa muissa kolmiulotteista grafiikkaa tarvitsevilla aloilla. Se soveltuu hyvin etenkin hahmojen ja kaikkien orgaanisten asioiden 3D-mallintamiseen.

2.2.1 Veistoksen rakenne

Digitaalinen veistos on tavallisten 3D-mallien tapaan polygonien muodostama verkko. ZBrushissa veistosta kutsutaan ZTooliksi. Veistosta ei yleensä rakenneta yhdestä monimutkaisesta polygoniverkosta, vaan veistämisessä voi käyttää apunaan Subtooleja. Subtoolit ovat pienempiä kokonaisuuksia, joista veistos muodostuu (kuva 2). Veistoksen luonnostaan erillään olevat osat on hyvä pitää erillisinä subtooleina, koska pienempien kokonaisuuksien käsittely on helpompaa ja vähemmän rajoittavaa, kuin yhden ison kokonaisuuden käsittely (Pixologic b). Subtooleja on mahdollista piilottaa näkyvistä, mikä

on hyödyllistä etenkin veistämisprosessin loppuvaiheessa, kun veistoksen polygonimäärä on todennäköisesti todella suuri. Subtoolien piilottaminen keventää ohjelman laskentakuormaa ja siten nopeuttaa navigointia ja työskentelyä. Jos yksi Subtool sisältää paljon polygoneja, voi Subtoolin osiakin piilottaa maskaamalla niitä.

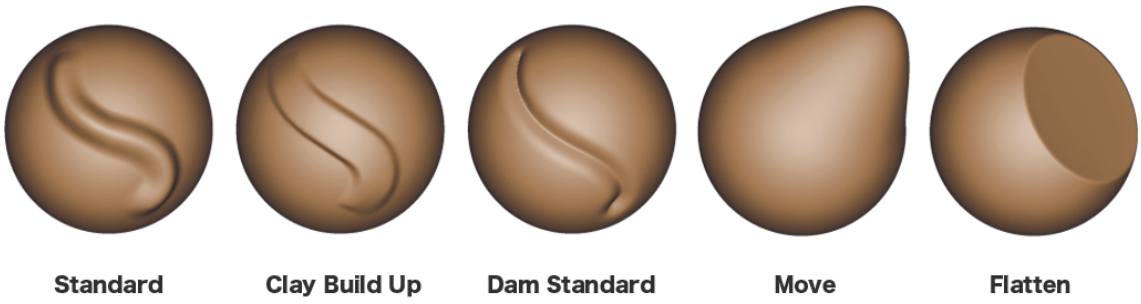
Subtoolit voivat myös muodostua pienemmistä osista (kuva 2). Näitä osia kutsutaan Zbrushissa Polygroupeiksi. Nimensä mukaisesti Polygroup on ryhmä polygoneja, joka on osa yhtä Subtoolia. Jokainen Subtool voi muodostua monesta eri Polygroupista. Polygroupeja ei välttämättä tarvitse huomioida, mutta niistä voi olla hyötyä monimutkaisissa veistoksissa. Niitä voidaan esimerkiksi käyttää automaattiseen polygonien valitsemiseen, mikä nopeuttaa työnkulkua.



Kuva 2. Veistoksen jokainen Subtool eriteltynä. Vartalon jokainen Polygroup merkittynä eri värillä.

2.2.2 Työkalut

Zbrush on täynnä hyödyllisiä työkaluja. Lähes kaikki sen työkaluista toimivat luontevasti siveltimien lailla. Jo muutamalla ZBrushin sisältämistä perussiveltimistä pääsee tosi pitkälle (kuva 3). Nämä siveltimet ovat: Standard Brush, Clay Build Up Brush saven lisäämistä varten, Dam Standard Brush viiltoja varten, Move Brush saven liikuttelu varten, Flatten Brush pintojen tasoittamiseen ja Smooth Brush saven siloittamista varten. Näillä siveltimillä on mahdollista tehdä lopullinen veistos valmiiksi, mutta muitakin siveltimiä voi käyttää oman mieltymyksen mukaan helpottamaan ja nopeuttamaan prosessia. Yksi ZBrushin hienouksista on, että se mahdollistaa myös omien siveltimien luomisen. Lisäksi siveltimillä on lukuisia eri parametreja, joita säätelemällä on mahdollista saada aikaan todella montaa erilaista jälkeä. Esimerkiksi siveltimien kokoa, tehokkuutta, muotoa, väriä ja materiaalia on mahdollista muuttaa (Pixologic c). Melkein jokaisessa siveltimessä on mahdollisuus käyttää joko Zadd tai Zsub-ominaisuuksia. Zadd lisää savea ja Zsub poistaa sitä.

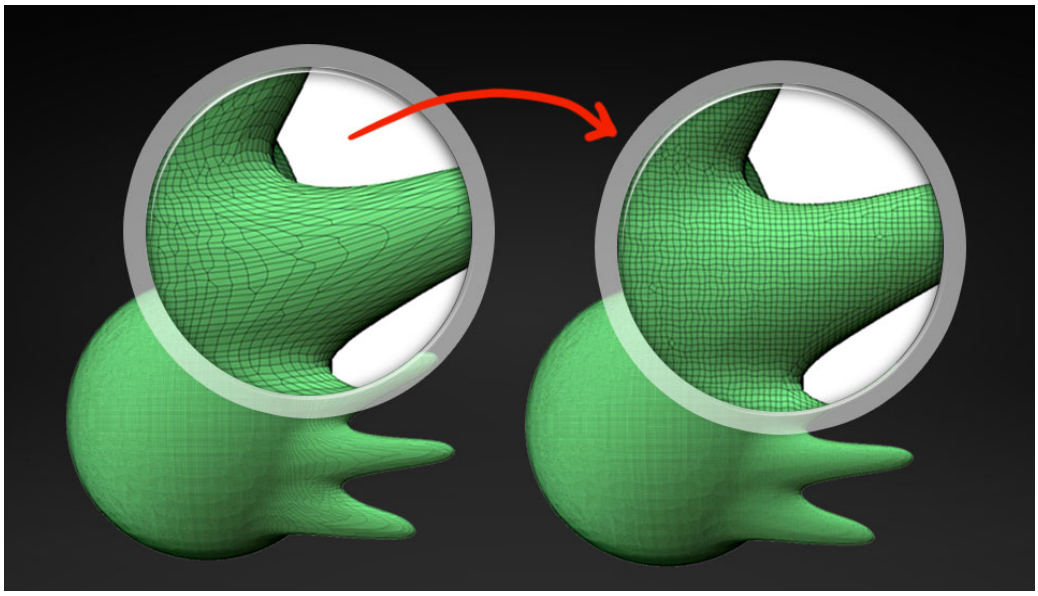


Kuva 3. ZBrushin keskeisten siveltimien tuottamat jäljet.

ZModeler on myös hyödyllinen “sivellin”, joka sisältää useita perinteisessä 3D-mallinnuksessa hyödynnettyjä työkaluja. Se mahdollistaa perinteisen 3D-mallintamisen ja siten samalla puhtaan topologian luomisen suoraan ZBrushissa. Koska muut siveltimet ovat erikoistuneet orgaanisten asioiden veistämiseen, on ZModeler todella hyödyllinen muiden, epäorgaanisten asioiden luomiseen.

Siveltimien lisäksi ZBrush sisältää monia muita työkaluja helpottamaan työskentelyä. Näistä merkittävin on DynaMesh. Se mahdollistaa veistoksen pinnan uudelleenrakentamisen automaattisesti sitä mukaan, kun veistokseen tehdään muutoksia (kuva 4). Suurin

hyöty siinä on, että veistoksen topologiaan ei tarvitse keskittyä ja suuria muutoksia on mahdollista tehdä vielä myöhemmissäkin työvaiheissa. Lisäksi DynaMesh mahdollistaa Subtoolien yhdistämisen yhtenäiseksi polygoniverkoksi. DynaMeshissä tulee ottaa huomioon, että sen resoluutio on riippuvainen mallin koosta. Jos mallin koko on liian pieni, ei DynaMesh todennäköisesti onnistu kovinkaan hyvin. Tällöin voidaan skaalata veistosta väliaikaisesti suuremmaksi, suorittaa DynaMesh ja skaalata veistos sitten takaisin normaaliin kokoon. Liian iso veistos puolestaan hidastaa DynaMeshin laskentaprosessia. Hyvän kokoisen veistoksen tulisi sopia suunnilleen ZBrushin sisältämään preview-ikkunaan. On hyvä ottaa huomioon, ettei DynaMeshia ei ole tarkoitettu käytettäväksi enää veistämisen loppuvaiheissa tarkkoja yksityiskohtia veistettäessä, koska DynaMeshilla saavutettavan polygoniverkon tiheys sekä topologia ovat rajallisia.



Kuva 4. Vasemmalla veistoksen topologia ennen Dynameshia ja oikealla Dynameshin jälkeen (Munoz, P 2015).

DynaMeshin lisäksi yksi hyödyllisimmistä ZBrushin työkaluista on ZRemesher. Se luo automaattisesti uuden verkon veistoksen pinnan muotojen mukaan, itse määritetyillä parametreilla. Tuloksena on paremmin käyttäytyvä topologia, joka on veistoksen muotojen mukainen. ZRemesher on hyödyllinen etenkin veistämisprosessin loppupuolella, kun ollaan siirrytty veistämään tarkempia yksityiskohtia ja puhdas topologia on tärkeämmässä asemassa.

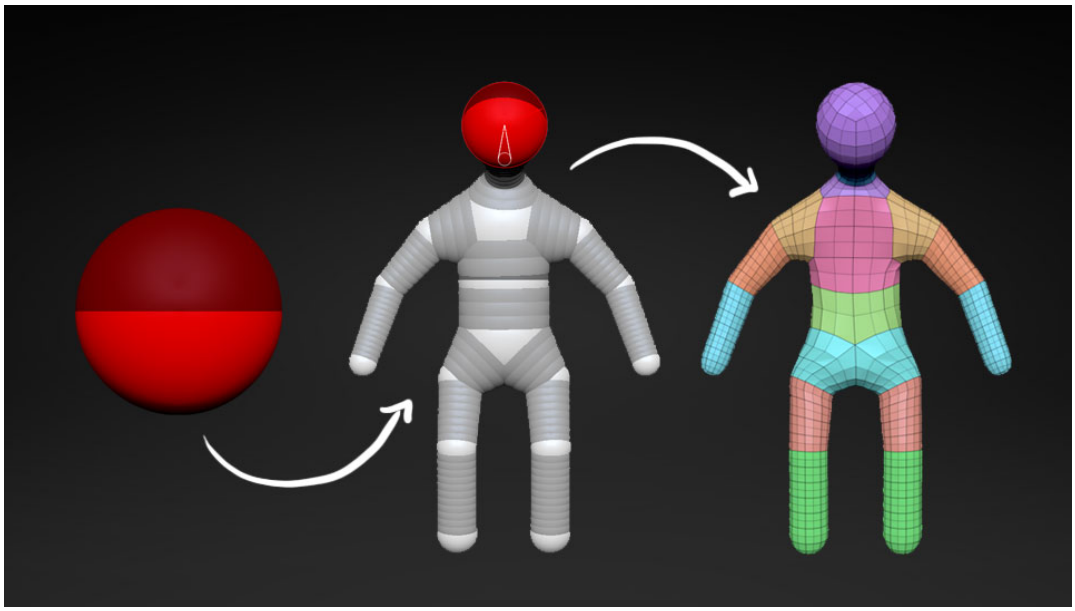
Kuva 5. Subdivisionit, DynaMesh ja ZRemesher löytyvät Työkaluvalikon Geometry-alavalikosta. Siveltimet löytyvät Brush-valikosta (Pixologic d).

2.2.3 Mistä aloittaa?

Hahmoja mallintaessa on otettava huomioon, että niiden on yleensä tarkoitus liikkua reaaliajassa moneen eri asentoon. Tämän takia hahmo mallinnetaan mahdollisimman neutraaliin ja symmetriseen asentoon, jotta sille on helppo rakentaa luuranko animaatiota varten ja että sen on mahdollista liikkua moniin eri asentoihin mallin muodon muuttumatta liian dramaattisesti. Pelihahmo mallinnetaan yleensä joko perinteiseen T-asentoon tai A-asentoon riippuen hahmon käyttötarkoituksesta. Jos hahmon on esimerkiksi tarkoitus nostaa käsiään ylös useissa tilanteissa, voi T-asento olla parempi vaihtoehto. A-asento on kuitenkin mielestäni parempi siitä syystä, että se on asentona ihmiselle luonnollisempi kuin T-asento. A-asennossa hahmo seisoo suorassa, kädet sivuille levitettynä noin 45-asteen kulmassa (kuva 2).

Veistäminen aloitetaan yleensä primitiiveistä. Primitiivit ovat yksinkertaisia 3D-muotoja, joita löytyy valmiina useita erilaisia melkein kaikista 3D-ohjelmista ja ne ovat varmasti tuttuja kaikille 3D-mallintajille. Primitiiveistä valitaan eniten veistoksen kohdetta muistuttava muoto, esimerkiksi pään veistäminen on helppo aloittaa pallosta. Primitiivien li-

säksi ZBursh mahdollistaa pohjamallin tekemisen Zspehrejen avulla. ZSpherejä asetellaan haluttuihin kohtiin ja niiden välit täyttyvät automaattisesti (kuva 6). Tämä mahdollistaa nopean luonnostelun etenkin sellaisiin osiin, joita on vaikea aloittaa primitiiveistä.



Kuva 6. Esimerkki ZSpherejen käytöstä (Munoz 2015).

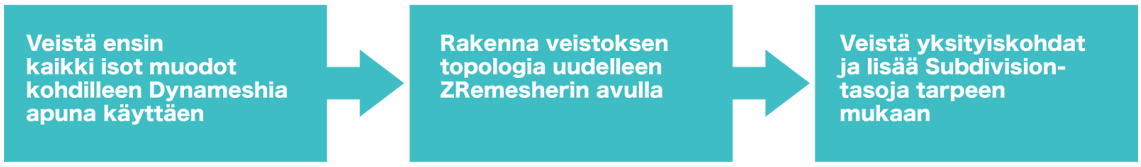
2.2.4 Veistämisprosessi

Veistämisen tavoitteena on 3D-malli, eli mallin on tarkoitus näyttää hyvältä joka suunnasta katsottuna. Alussa on hyvä keskittyä mittasuhteisiin ja hahmon suuriin linjoihin, sillä ne ovat todella kriittisiä onnistuneen veistoksen saavuttamiseksi. Veistoksen isot muodot pyritään saamaan kohdalleen ennen tarkkoihin yksityiskohtiin siirtymistä. Dynamesh on todella hyödyllinen työkalu veistoksen alkuvaiheessa, koska muotoja voi muuttaa vapaasti kiinnittämättä huomiota veistoksen topologiaan. Kun isot muodot on saatu kohdilleen, ei Dynameshia tarvitse enää käyttää, koska veistoksen muodot eivät tule muuttumaan enää niin radikaalisti.

Pienempiin yksityiskohtiin siirryttäessä todennäköisesti tarvitaan lisää polygoneja, jotta veistäminen onnistuisi ilman, että polygonit loppuvat kesken. Silloin veistos tai sen osa voidaan Subdividata. Subdividaus tarkoittaa jokaisen polygonin jakamista neljään osaan, jolloin veistoksen polygonimäärä lisääntyy ja veistoksen pinta muuttuu sileämmäksi. Ennen tätä veistokselle on hyvä luoda uusi verkko paremmalla topologialla Zremesherilla, etenkin jos apunaan käytti Dynameshia, sillä Dynameshin muodostama verkko ei sovellu

tarkkaan veistämiseen kovin hyvin. ZRemesherin muodostama tasainen pinnanmuotojen mukainen polygoniverkon rakenne on paljon mieluisampi veistää

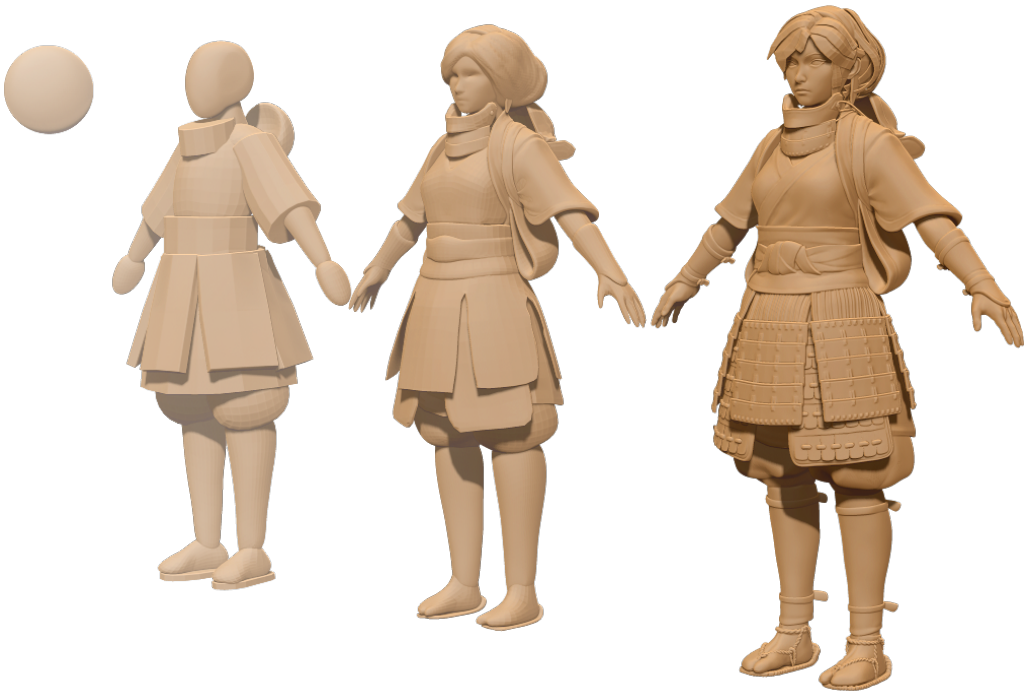
ZRemeshauksen jälkeen veistosta voi Subdividata sitä mukaan uudelleen, kun siirtyy veistetämään tarkempia yksityiskohtia ja polygonien määrä ei tunnu riittävän. Subdivision-tasojen välillä voidaan liikkua vapaasti riippuen siitä, kuinka isoja muutoksia haluaa veistokseen tehdä. Hyvä käytäntö on, että suuret muutokset veistokseen tehdään pienimmällä Subdivision-tasolla ja vain kaikista tarkimmat yksityiskohdat suurimmalla Subdivision-tasolla. Veistoksen jokainen osa ei todennäköisesti vaadi yhtä tarkkoja yksityiskohtia, eikä siten yhtä paljon Subdivisioneita ja polygoneja, joten veistos olisi hyvä muodostua useammasta eri osasta.



Kaavio 1. ZBrushin veistämisprosessi yksinkertaistettuna.

Veistoksen materiaalia ja valon suuntaa voi myös vaihtaa helpottamaan etenkin varjoon jäävien alueiden tarkempaan veistämiseen. Veistoksen materiaalin voi vaihtaa flat-materiaaliin, jolloin veistos ei ota vastaan valoa ja siten siluetti on helpompi hahmottaa.

Yksi digitaalisen kuvanveiston suurimmista eduista on automaattinen symmetrisyys. Eli kaikista veistoksen symmetrisistä osista tarvitsee veistää vain toinen puoli ja toinen syntyy automaattisesti sen pohjalta. Siitä kannattaa siis ottaa kaikki hyöty. Symmetrisyydessä pitää kuitenkin ottaa huomioon, että täydellinen symmetrisyys näyttää epäluonnolliselta, joten symmetrisyyttä on hyvä hieman rikkoa etenkin symmetristen puoliskojen liitoskohdassa. Tämä tehdään yleensä vasta veistoksen viimeistelyvaiheessa, sillä jos symmetrisyys on rikottu, ei symmetrinen veistäminen onnistu enää ongelmitta. Koska automaattinen symmetrisyys on niin tehokas työkalu, kannattaa sitä hyödyntää myös veistoksen osissa, jotka eivät sijaitse symmetrisesti. Ne voidaan veistää ensin symmetrisesti ja vasta myöhemmin asettaa oikeille, epäsymmetrisille paikoilleen.



Kuva 7. Veistämisprosessi pallosta valmiiksi veistokseksi.

2.2.5 Lopullisen veistoksen valmistelu seuraavia vaiheita varten

Mitä pienempi määrä polygoneja veistoksessa on, sitä nopeampaa pintanormaalien ja muiden yksityiskohtien beikkaaminen tulee olemaan. Nykyisin tällä ei ole niin suurta merkitystä beikkausaikojen lyhentyessä mitättömän pieniksi, mutta jos joudutaan tekemään lukuisia uudelleenbeikkauksia, on lyhyempi odotusaika toivottua. Alhainen polygonimäärä auttaa myös veistoksen käsittelynopeuteen myöhemmissä vaiheissa.

Jotta veistoksen polygonimäärä saataisiin mahdollisimman pieneksi, valmis veistos täytyy vielä Decimoida. Decimointi tarkoittaa yksinkertaisuudessaan 3D-mallin polygonimäärän vähentämistä muuttamatta mallin ulkonäköä. Zbrush osaa tehdä tämän automaattisesti ja se tapahtuu liitännäisellä nimeltä Decimation Master.

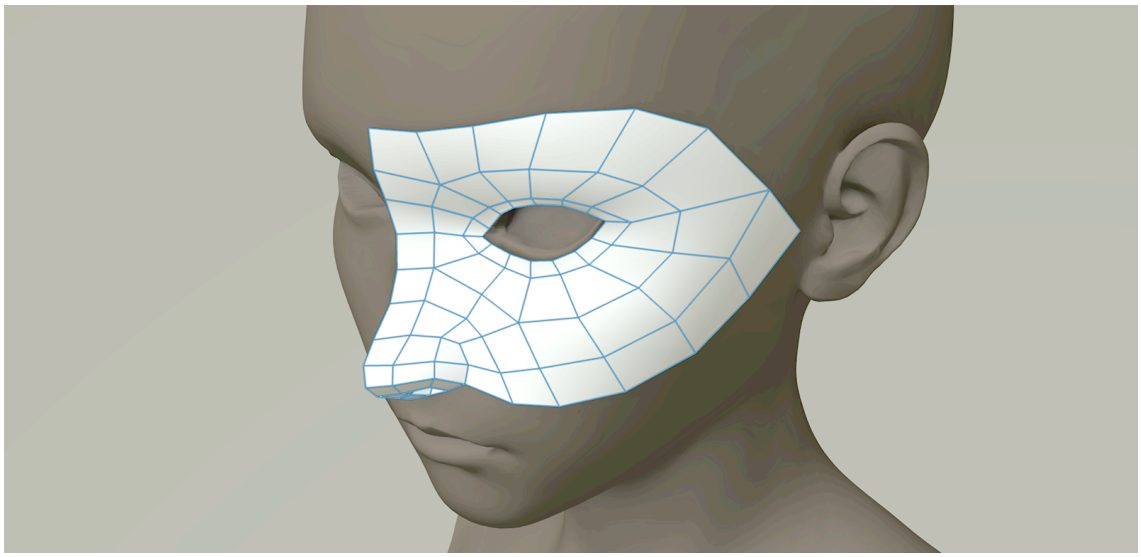
Lopullinen veistos tallennetaan yleensä obj-tiedostomuodossa, sillä se toimii lähes kaikissa 3D-ohjelmissa. Toinen vaihtoehto on fbx-tiedostomuoto, minkä tallentaminen ZBrushista onnistuu liitännäisellä.

Kuva 8. Fbx-tiedostomuodossa tallentaminen onnistuu Zplugin-valikon alavalikosta FBX Export/Import (Pixologic e).

3 TOPOLOGIAN UUDELLEEN RAKENTAMINEN

Seuraava vaihe on topologian uudelleen rakentaminen (englanniksi retopology). Sillä tarkoitetaan 3D-mallin geometrian rakentamista uudelleen optimaalisemmalla topologialla referenssimallin, eli veistetyn highpoly-mallin pohjalta. Veistoksen polygoniverkko on yleensä niin tiheä ja sotkuinen topologialtaan, ettei sitä voi käyttää sellaisenaan ainakaan pelimoottoreissa, joissa hyvä topologia ja alhainen polygonimäärä ovat tärkeässä asemassa. Topologian uudelleen rakentamisen tavoitteena on toteuttaa lowpoly-malli, joka on siluetiltaan lähestulkoon identtinen referenssinä käytetyn highpoly-mallin kanssa, mutta polygonien määrä ja mallin rakenne on optimoitu pelimoottoria ja usein animointia varten.

Topologian uudelleenrakentamiseen on monta eri tapaa ja käytettävät työkalut eroavat hieman toisistaan eri 3D-ohjelmien välillä, mutta periaate pysyy kuitenkin samana. Polygoneja ja/tai verteksejä kiinnitetään referenssimallin pintaan, kunnes pinta on kokonaan peitetty uudella polygoniverkolla (kuva 9).



Kuva 9. Polygoneja kiinnitetään highpoly-mallin pintaan.

Topologia on myös mahdollista luoda automaattisesti työkaluilla kuten Zbrushista löytyvällä ZRemesherilla. ZRemesherilla on mahdollista saada hyviäkin tuloksia, mutta manuaalisella topologian uudelleen rakentamisella saadaan yleensä hallitumpi topologia ja

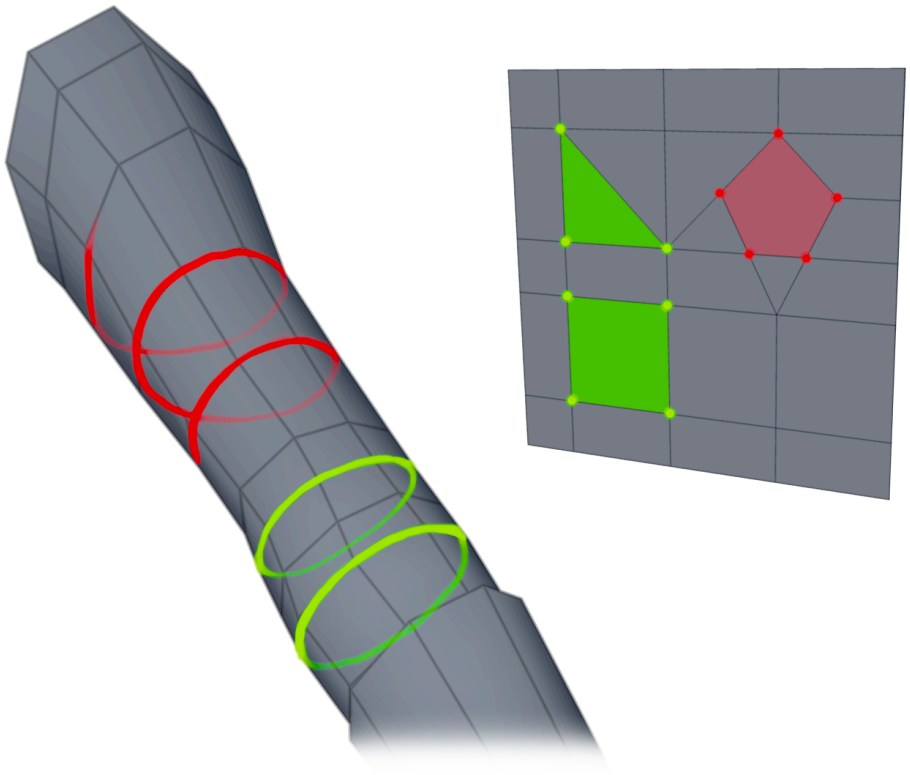
polygonien virta. Jos mallinnetun objektin on tarkoitus olla staattinen eikä topologialla ole niin paljon merkitystä, voi ZRemesher olla hyvä vaihtoehto nopeuttamaan työskentelyä. ZRemeshilla toteutettua lowpoly-mallia voi myös mahdollisesti käyttää pohjana omalle lowpoly-mallille, mikä nopeuttaa työskentelyä.

Koska 3D-mallin siluettiin ei voi vaikuttaa tekstuurien avulla, highpoly-mallin siluetin säilyttäminen lowpoly-mallissa mahdollisimman tarkkaan on yksi tärkeimmistä asioista johon tulisi kiinnittää huomiota tässä vaiheessa. Kaikista pienimpiä yksityiskohtia ei ole syytä mallintaa polygoneilla, vaan ne on hyvä sisällyttää pelkästään normaalikarttaan. Tekstuuriin säilötty pinnanmuoto vaatii yleisesti ottaen vähemmän laskentatehoa kuin sama muoto polygoneilla mallinnettuna.

Toinen seikka, johon kiinnittää huomiota lowpoly-mallia tehdessä on polygonien määrä. Nykyisin digitaalisen veistoksen polygonimäärä voi liikkua jopa kymmenissä miljoonissa, riippuen siitä kuinka tarkkoja yksityiskohtia halutaan. Peleissä yhdessä mallissa on maksimissaan satoja tuhansia polygoneja, yleensä paljon vähemmän. PC- ja konsolipelissä luku on yleensä kymmenissä tuhansissa ja mobiilipeleissä sadoissa tai tuhansissa. Polygoneja tulisi kuitenkin olla riittävästi joka kohdassa, ettei beikattava normaalikartta joudu kompensoimaan pinnanmuotoja liikaa.

Hyvässä topologiassa polygoneja on mahdollisimman tasaisesti ja ne kulkevat mallin muotojen mukaisesti, koska se helpottaa muotojen määrittämisessä. Polygonien reunat muodostavat silmukoita, eivätkä spiraaleita (kuva 10). Reunasilmukat ovat oleellisia useiden 3D-mallinnusohjelmista löytyvien työkalujen toiminnallisuuden kannalta. Lisäksi spiraalit aiheuttavat hämmennystä ja ovat haitallisia luurangon ja animaation tekemiseen. Animaation kannalta hyvässä topologiassa on myös oltava tarpeeksi reunasilmukoita nivelten kohdalla. joista hahmo venyy ja menee kasaan sen animatoitaessa, kuten polvet tai kyynärtaipeet.

On myös kiinnitettävä huomiota siihen, että minkälainen topologia on hyvää beikkauksen kannalta. Esimerkiksi yli 90 asteen kulmat pintojen välillä eivät ole suositeltavia beikkauksen onnistumiseksi. Sellaisiin kohtiin voidaan enemmän polygoneja kulman pienentämiseksi.



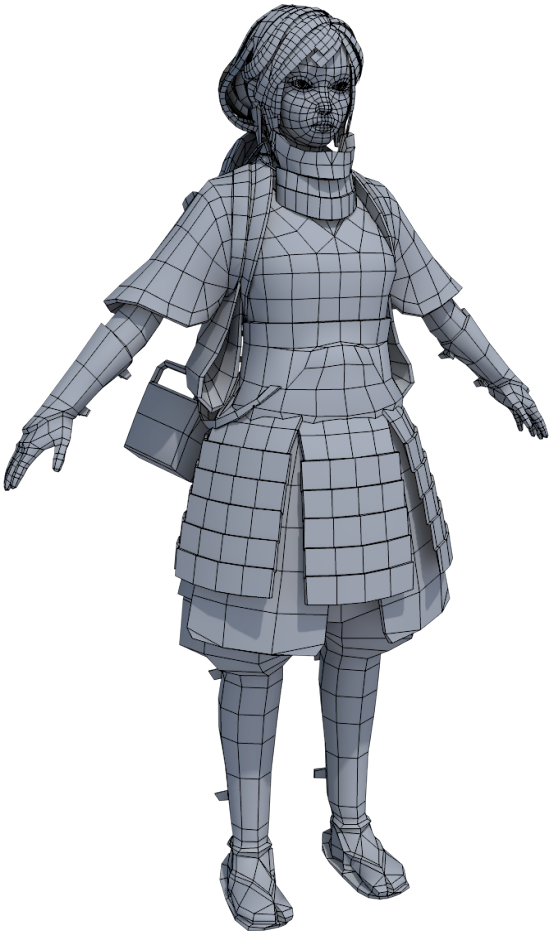
Kuva 10. Reunojen muodostamia spiraaleja tulisi välttää ja suosia reunasilmukoita. Malleissa tulisi käyttää vain nelikulmioita ja kolmioita, ei n-goneja.

3.1 Kolmiot, nelikulmiot ja monikulmiot

Yksi hyvän topologian ominaispiirteistä on, että se muodostuu melkein kokonaan nelikulmioista. Nelikulmiot ovat helpommin erotettavissa ja ymmärrettävissä kuin kolmiot. Nelikulmioiden muodostamat yhdensuuntaiset linjat on helppo hahmottaa ja monet 3D-ohjelmien työkaluista käyttävät samansuuntaisia reunaviivoja hyväkseen. 3D-malli ei myöskään taivu yhtä nätisti kolmioiden kohdalta kuin nelikulmioiden kohdalta, jos sille on tarkoitus tehdä animaatioita. Uutta topologiaa rakentaessa tulisi siis suosia nelikulmioita ja käyttää kolmioita vain polygonien määrän minimoimiseen ja reunojen sijoittamiseen tietyissä kohdissa (Taylor 2015).

N-goneja, eli viiden tai useamman verteksin muodostamia pintoja malleissa ei saa olla (kuva 10). Ne aiheuttavat valaisuvirheitä ja eivät käyttäydy kuten nelikulmiot ja kolmiot useissa tapauksissa. Paremman polygonivirran saavuttamiseksi myös viiden reunan kohtaamispisteitä tulisi olla mahdollisimman vähän, mutta niitä on vaikea välttää kokonaan.

Niitä ei kannata sijoittaa ainakaan muotoa muuttaville alueilla, eli esimerkiksi hahmon nivelkohtiin.

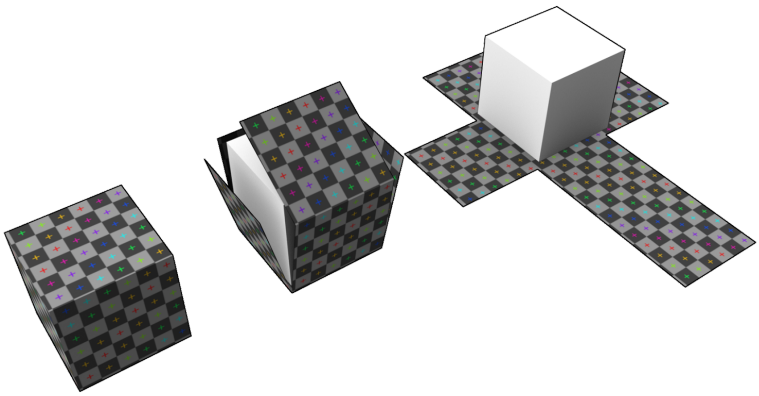


Kuva 11. Lowpoly-mallin polygoniverkko.

4 UV-KARTOITUS

Kun Lowpoly-malli on valmis, eikä siihen tehdä enää muutoksia, voidaan se UV-kartoittaa (englanniksi UV mapping). On syytä ottaa huomioon, että UV-kartoitusta ei tarvitse tehdä highpoly-mallille, sillä se ei tarvitse UV-koordinaatteja beikkausta varten.

UV:t ovat 2D-tekstuurikoordinaatteja, jotka vastaavat 3D-mallin verteksejä. UV-kartoituksella tarkoitetaan prosessia, jossa 3D-mallin pinta levitetään 2D-tilaan eli UV-koordinaatistoon (kuva 12). UV-koordinaatit määrittävät tekstuurikartan sijainnit 3D-mallin pinnalla, joten UV-kartoitus on tärkeä prosessi sen kannalta, että mallin pintaan saadaan lisättyä tekstuuri-informaatiota oikeisiin kohtiin. (Autodesk Maya 2014.)



Kuva 12. Kuution UV-kartoitus havainnollistettuna (Wikipedia, UV Mapping 2017)

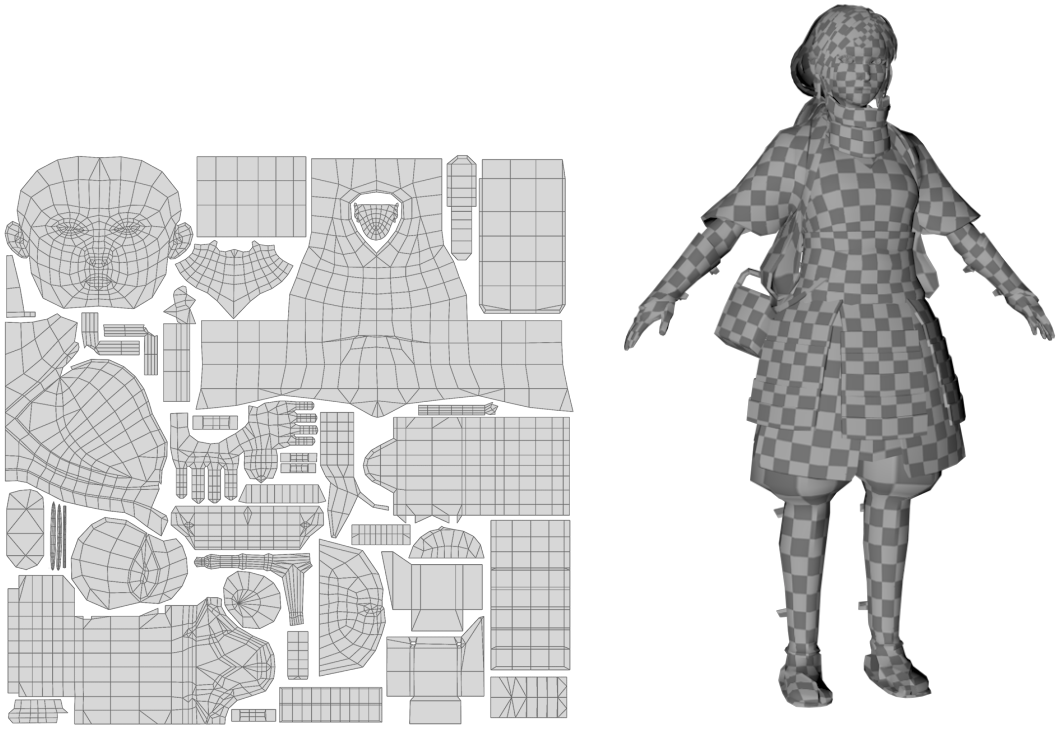
Monimutkaisen 3D-mallin levittäminen UV-kartaksi yhtenä palasena on usein mahdotonta ilman ongelmallista venymistä. Tämän takia 3D-malli on levitettävä UV-kartaksi paloina (kuva 13). Näitä paloja kutsutaan UV-saarekkeiksi. Saarekkeita tulisi olla mahdollisimman vähän, sillä saarekkeen reunojen kohdalle tulee aina sauma. Näiden saumojen kohdalla tekstuuri katkeaa ja jatkuu toisesta kohtaa. ja niitä on hankala saada pois näkyvistä. Saumat pyritään sijoittamaan sellaisiin kohtiin mallista, joista niitä on vaikea

havaita. Jokainen saumakohta lisäksi hidastaa pinnan valaisun laskentanopeutta. Saumojen kohdalle tulee pelimoottoreissa yhden verteksin sijaan kaksi päällekkäin. 3D-mallinnusohjelmat eivät yleensä näytä näitä “piilotettuja” verteksejä, sillä ne eivät ole niin oleellisia mallintajan kannalta, vaan pelimoottorit käyttävät niitä laskutoimituksissa.

3D-mallin symmetrisiä osia ei ole syytä UV-kartoittaa kahteen kertaan. Symmetriset osat voivat käyttää samoja UV-koordinaatteja, jotta samaa kohtaa ei tarvitse toistaa kahteen kertaan tekstuurissa ja siten pikselitiheyttä saadaan maksimoitua. Symmetristen osien toisen puoliskon voi poistaa UV-kartoituksen ajaksi ja peilata osat lopuksi uudelleen, jolloin myös samat UV-koordinaatit kopioituvat peilattuihin osiin. Samoja UV-koordinaatteja voi hyödyntää myös muissa osissa jotka käyttävät samaa tekstuuria keskenään.

UV-kartoitus aloitetaan UV:iden automaattisella levittämisellä. 3D-mallinnusohjelmat tarjoavat yleensä monta eri tapaa levittää UV.t. Yleensä on hyvä kokeilla useaa eri levitystapaa ennen kuin alkaa muokkaamaan UV-saarekkeita niiden lopulliseen muotoon. Automaattisella levityksellä saa harvoin täydellistä tulosta ja UV:ita pitääkin melkein aina muokata manuaalisesti. Jos levitettävä muoto on neliö tai sylinteri, polygonit on hyvä levittää yhdenmukaisiin linjoihin ja 90 asteen kulmiin. Esimerkiksi raajat ovat melkein sylinterin muotoisia, joten ne voidaan UV-kartoittaa tällä tavalla. Yhdenmukaisista linjoista on hyötyä etenkin saumojen peittämisessä ja niiden avulla on helpompi toteuttaa tasaisia tekstuureita. Jos teksturoinnin haluaa toteuttaa 2D-tilassa eli esimerkiksi Photoshopissa, on syytä kiinnittää huomiota myös siihen, mistä kuvakulmasta tekstuurit olisivat mieluisin toteuttaa.

UV-saarekkeiden kokojen kanssa on otettava huomioon tekselitiheys, eli toisin sanottuna kuinka paljon tekstuurin resoluutiota on varattu 3D-mallin pinnan eri kohtiin. Mallin eri osien välillä tulisi olla yhtäläinen tekselitiheys kuten myös pelimaailmassa käytettävien eri mallien välillä. Esimerkiksi jos yhden metrin korkuiseen ja levyiseen pintaan käytetään 1024 x 1024 resoluutioista tekstuuria, tulisi muihinkin saman kokoisiin pintoihin käyttää yhtä paljon resoluutiota. Yhtenäinen tekselitiheys varmistaa, ettei mallien välillä ole huomattavia eroja yksityiskohdissa. UV-kartoituksessa tekselitiheyttä on helppo tarkkailla ruutukuvioisen tekstuurin avulla (kuva 13). Jotta tekselitiheys on yhtäläinen mallin eri osien välillä, tulisi ruutukuvion näyttää mahdollisimman tasaiselta joka puolelta. Poikkeuksena tärkeät alueet, kuten hahmon naama, joille voi varata hieman enemmän tilaa.



Kuva 13. Vasemmalla esimerkki UV-kartoituksesta. Oikealla ruutukuvioinen tekstuuri 3D-mallin pinnalla eri alueiden resoluution sovittamiseksi yhtä tiheiksi ja venymisen tarkkailua varten.

Kun mallin kaikki polygonit on levitetty UV-saarekkeiksi, voidaan saarekkeet sijoittaa UV-tilaan. UV-saarekkeiden asettelu UV-tilaan aloitetaan suurimmasta saarekkeesta, sillä sille on haastavinta löytää sopiva tila. Pienet saarekkeet on helppo sommitella lopussa jäljelle jääneisiin rakoihin, joita muut saarekkeet eivät vielä peitä. 3D-mallissa vierekkäin sijaitsevien alueiden uv-saarekkeet on myös hyvä sijoittaa lähekkäin toisiaan, jotta teksturoitaessa on helpompi hahmottaa, missä mikäkin mallin osa sijaitsee UV-tilassa.

Saarekkeet pyritään levittämään siten, että ne ei eivät ole toistensa kanssa päällekkäin ja niiden väliin jäisi mahdollisimman vähän tyhjää tilaa (kuva 13). Tilaa pitää kuitenkin jättää tietty määrä riippuen käytettävän tekstuurin resoluutiosta. Mitä suurempi resoluutio tekstuurilla on, sitä vähemmän tilaa uv-saarien välille tarvitsee jättää. Väleihin jäävään tyhjään tilaan lisätään yleensä edge paddingiä, jotteivat saumakohdat tulisi näkyviin.

Hyvä määrä edge paddingiä riippuu tekstuurin koosta. Kehotuskoot ovat suunnilleen seuraavanlaiset (Polycount 2017):

- Tekstuurin koko 256 x 256 = 2px rako saarekkeiden välillä
- Tekstuurin koko 512 x 512 = 4px rako saarekkeiden välillä
- Tekstuurin koko 1024 x 1024 = 8px rako saarekkeiden välillä
- Tekstuurin koko 2048 x 2048 = 16px rako saarekkeiden välillä

5 BEIKKAUS

Beikkaus (englanniksi baking) tarkoittaa 3D-mallin pinnan yksityiskohtien siirtämistä toisen mallin pinnalle tekstuurikarttojen muodossa. Beikkausta varten tarvitaan highpoly-malli (veistos), josta halutaan kopioida pinnan yksityiskohdat, sekä UV-kartoitettu lowpoly-malli, johon nämä yksityiskohdat halutaan kopioida. Beikkauksessa lowpoly-mallin pinnalta ammutaan säteitä highpoly-mallin pinnalle. Säteen osuessa highpoly-mallin pinnalle, kyseisen kohdan pinnan yksityiskohdat tallentuvat tekstuurikarttaan lowpoly-mallin UV-koordinaattien mukaisesti (Polycount 2016a).

5.1 Beikkauksen valmistelu

Jotta beikkaus onnistuisi, on otettava huomioon muutama tärkeä seikka. Highpoly-mallin ja lowpoly-mallin on sijaittava samassa kohdassa 3D-tilassa, niillä pitää olla sama rotaatio ja niiden pitää olla samankokoisia. Ennen beikkausta on varmistettava, että kaikki symmetriset palaset ovat täydellisesti päällekkäin UV-kartassa. Tällaista geometriaa on myös hyvä liikuttaa UV-kartastossa sivulle 1 yksikön verran, ettei päällekkäin olevien UV-saarekkeiden kohdalle beikkaannu kahdesta eri kohdasta informaatiota, mikä saattaa aiheuttaa beikkausvirheitä (Polycount 2016a).

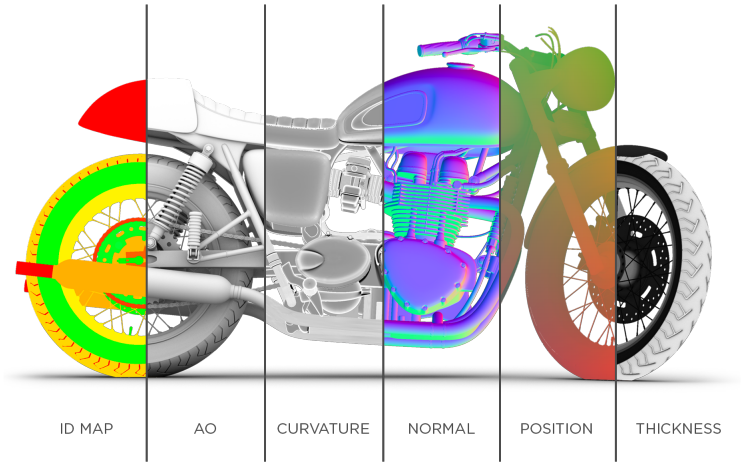
Vielä lopuksi ennen beikkauksen aloittamista on lowpoly-mallin kaikki nelikulmiot muutettava kolmioiksi. Kolmioiksi muuttaminen tapahtuu automaattisesti komennolla (yleensä nimellä Triangulate), joka löytyy melkein kaikista 3D-mallinnusohjelmista. Nelikulmioiden poistamisella varmistetaan, että beikatessa nelikulmot ovat jaettuna kolmioiksi samalla tavalla kuin pelimoottorissa. Jos nelikulmio jakautuu kolmioksi myöhemmin toisinpäin kuin beikatessa, tekstuurikartat todennäköisesti vääristyvät siltä kohdalta. Lisäksi jos malli sisältää peilattua geometriaa, jolla on samat UV-koordinaatit, pitää varmistaa, että nelikulmiot ovat jakautuneet samoin kummallakin puolella. Beikatessa on hyvä käyttää kopiota lowpoly-mallista, sillä lopullisesti kolmioista muodostuva malli on paljon vaikeammin käsiteltävä kuin nelikulmioista muodostuva malli, kuten jo aiemmin todettiin.

Jos malleissa on lähekkäin olevia erillisiä osia, jotka muodostavat ahtaita kohtia, saattavat ne aiheuttaa epätoivottuja artifakteja beikattuihin tekstuurikarttoihin. Tämän estämiseksi osat voidaan liikuttaa beikkausta varten erilleen toisistaan, jotta ne eivät ole toisistaan lähtevien säteiden tiellä. Esimerkiksi Substance Painter mahdollistaa eri osien nimeämisen tietyillä päätteillä, jolloin beikkaantuminen hoituu niiden perusteella. Toinen mahdollinen tapa on jakaa malli osiin beikkauksen ajaksi ja liikuttaa osat eri kohtiin siten, etteivät ne kosketa toisiaan.

5.2 Beikkaus

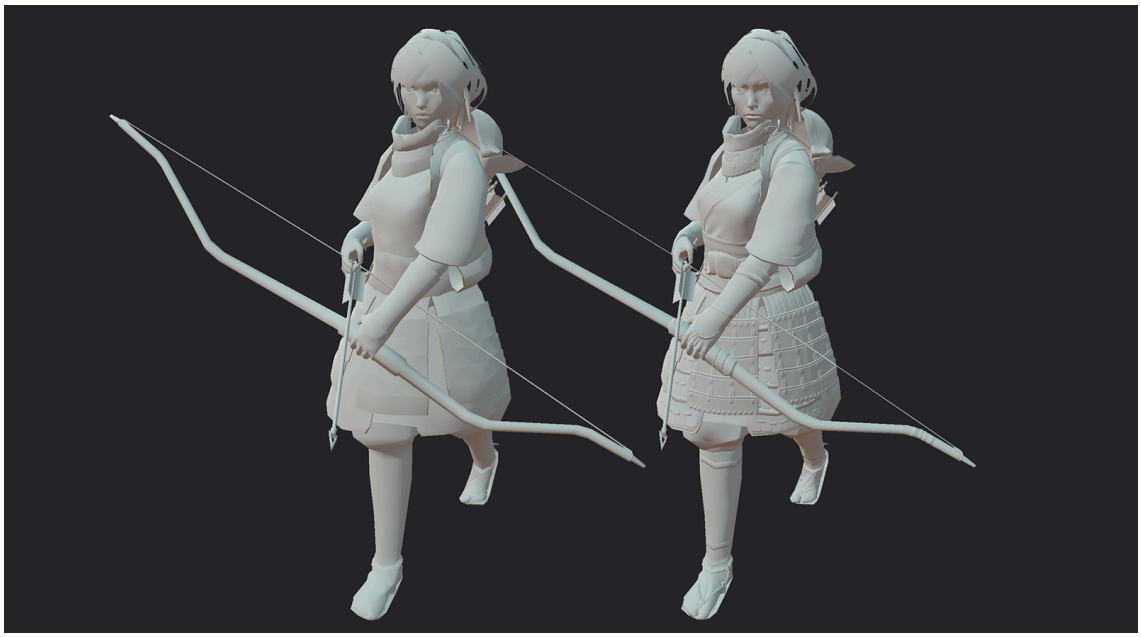
Monessa 3D-mallinnusohjelmassa on sisäänrakennettu beikkaaja. On myös beikkaukseen erikoistuneita ohjelmia kuten xNormal ja Knald. Kaikilla beikkausohjelmilla on mahdollista saada hyvin samankaltaisia tuloksia ja suurimmat eroavaisuudet ovatkin beikkaukseen kuluvissa ajoissa. Beikkausohjelmat mahdollistavat monien eri tekstuurikarttojen beikkaamisen (kuva 14). Suosituimmat beikattavista tekstuurikartoista ovat:

- Normaalikartta, joka sisältää oman normaalin jokaiselle pikselille
- Ambient occlusion kartta, joka sisältää pehmeän varjostuksen, mikä ilmenisi silloin kun malliin ei kohdistu suoraa valonlähdettä
- Curvature kartta, joka sisältää pinnan kurvit



Kuva 14. Beikattavat tekstuurikartat (Allegorithmic)

Kaikista hyödyllisin beikatuista kartoista on kuitenkin normaalikartta. 3D-mallissa jokaisella polygonilla ja verteksillä on oma normaalinsa. Normaali on kuin nuoli, joka osoittaa kohtisuoraan poispäin sen omistavasta komponentista, osoittaen suunnan josta valo osuu siihen kohtisuoraan. Ilman normaalikartta, normaalien määrä on hyvin rajallinen ja siten yksityiskohtaisten pintojen esittäminen vaatisi todella ison määrän polygoneja. Normaalikartat ovat RGB-kuvia, joissa RGB-värikanavat edustavat 3D-tilassa käytettäviä X, Y ja Z-koordinaatteja. Punainen on X, vihreä Y ja sininen Z. Eli jokaisella pikselillä normaalikartassa on oma suuntansa mihin se osoittaa (Ward, A 2008, 118). Pelimalleissa käytetään yleensä tangentti-normaalikarttoja, koska ne eivät ole riippuvaisia objektin rotaatiosta vaan toimivat suhteessa 3D-mallin pintaan jossa ne sijaitsevat. Lisäksi ne säästävät muistia enemmän verrattuna muihin normaalikarttatyyppeihin (Crytek 2013).



Kuva 15. Hahmon 3D-malli ilman normaalikarttaa sekä normaalikartan kanssa.

Beikattuja tekstuurikarttoja voidaan hyödyntää eri tavoin. Esimerkiksi Ambient occlusion-karttaa voidaan käyttää sellaisenaan pelimoottorissa tai sitä voidaan hyödyntää värikartan pohjana. Curvature-kartan avulla voidaan tuoda esiin esimerkiksi eri alueiden kuluneisuutta. Pelimoottorit tarjoavat mahdollisuuden käyttää erilaisia tektuurikarttoja materiaalista ja shaderista riippuen.

5.3 Virheiden korjaus

Täysin virheetöntä beikkiä on vaikea saada monimutkaisesta 3D-mallista. Yleensä joutuu tekemään pieniä muutoksia lowpoly-mallin topologiaan tai UV-karttaan ja beikkaamaan useamman kerran ennen kuin lopputulokseen voi olla tarpeeksi tyytyväinen.

Yleinen virhe beikkausvirhe tapahtuu, kun yritetään beikata vähän polygoneja sisältävä kaareva pinta. Tällaiseen kohtaan muodostuu usein aaltoileva tekstuuri. Se on mahdollista korjata lisäämällä polygoneja lowpoly-malliin kyseiseen kohtaan vastaamaan highpoly-mallin muotoa tarkemmin. Jos polygonibudjetti on jo täytetty, voi tällaiset kohdat yrittää suoristaa esimerkiksi Photoshopissa, mutta se ei kuitenkaan korjaa ongelmaa kokonaan.

Jos beikkaus onnistuu paremmin eri parametreilla yhdestä kohtaa kuin toisesta, voidaan beikattuja tekstuurikarttoja yhdistää esimerkiksi Photoshopissa leikkaa ja liimaa-periaattella. Kaikista onnistuneista beikeistä otetaan kaikki onnistuneet osat ja yhdistetään ne yhdeksi virheettömäksi kokonaisuudeksi. Pieniä virheitä on mahdollista maalata tai sumentaa pois, mutta tämä on vasta viimeinen vaihtoehto beikkauvirheiden korjaamiseen, sillä matemaattisesti lasketut tekstuurit ovat tarkempia kuin silmämääräisesti maalatut.

Beikkaaminen ei vaadi juurikaan manuaalista työskentelyä, ohjelma hoitaa melkein kaiken puolestasi, mutta se vaatii tietämystä normaaleista ja 3D-malleista ylipäätään, jotta beikkauksessa tapahtuneita virheitä ei tulisi ja jotta niitä osaisi korjata tarvittaessa. Jos kaikki onnistuu hyvin, beikkaaminen vie vain hetken, mutta virheiden korjaaminen ei välttämättä ole helppoa.

6 TEKSTUROINTI

Tekstuurit ovat 3D-mallien pinnoilla käytettäviä bittikarttakuvia, joita hyödynnetään moneen eri tarkoitukseen. Teksturoinnilla tarkoitetaan näiden kuvien tuottamista. Nykyään pelimoottoreissa on mahdollistaa käyttää lukuisia eri tekstuureita, näistä suosituimpia ovat seuraavat:

- Värikartat, joita käytetään objektin pohjavärin määrittämiseen
- Bump-kartat, jotka määrittävät pinnan epätasaisuuksia
- Spekulaarikartat, jotka määrittävät pinnan spekulaarisuuden, eli kuinka kiiltävä pinta on miltäkin kohdalta.

Tekstuurikartat voivat sisältää RBG-kanavien lisäksi Alpha-kanavan, johon on yleistä sisällyttää läpinäkyvyys-kartta, joka määrittää 3D-mallin läpinäkyvät alueet. Käytettävät tekstuurit riippuvat pelimoottorista, kohdelaitteesta, shaderista, materiaalista ja halutusta tyylistä.

Tekstuurien valmistamiseen voidaan käyttää lukuisia eri tekniikoita ja työkaluja. Tekstuureita voidaan tuottaa esimerkiksi maalaamalla, kuvamanipuloimalla tai proseduraalisella generoinnilla (Polycount 2016b). Tekstuurien pohjana voidaan hyödyntää monia beikkauksesta saatuja tekstuurikarttoja. Esimerkiksi ambient occlusion-karttaa tai curvature-karttaa voidaan hyödyntää värikartan pohjana.

Ennen aloittamista voi teksturointia helpottamaan generoida 3D-mallinnusohjelmassa ääriviivat UV-kartastoon levitetyistä polygoneista. Myös ID-kartan luominen voi olla hyödyllistä. Se merkitsee materiaalit tekstuurikarttaan eri väreillä, jotta jokaisen eri materiaalinen viemä alue on helppo valita teksturoidessa.

Useimmissa pelimoottoreissa tekstuurin resoluution on oltava suorituskyvyn maksimoimiseksi kahden potenssissa, tyypillisesti 256 x 256, 512 x 512, 1024 x 1024, 2048 x 2048 tai 4096 x 4096. Tekstuurien resoluutiossa on huomioitava myös, että minkä kokoinen mallin on tarkoitus olla muuhun pelimaailmaan verrattuna. Jos sen on tarkoitus olla pienikokoinen, on tekstuurin myös hyvä olla pienikokoinen, jotta pelimaailmassa olevien 3D-mallien välillä säilyy yhtenäinen resoluution tiheys. (Polycount 2016b.)

Yleinen tekstuurissa esiintyvä ongelma ovat saumakohdat, joita esiintyy UV-saarekkeiden reunakohdissa. Helpoin tapa päästä niistä eroon on maalata ne piiloon 3D-maalauksella. Esimerkiksi BodyPaint 3D ja 3D-Coat sisältävät työkalut 3D-tilassa maalaamiseen. Jos UV-kartoituksessa levitti polygonit suoriin linjoihin, on saumakohtien linjaus helpompaa koska saumakohdan kummatkin puolet ovat samassa kohdassa tietystä suunnasta katsottuna.

Tekstuureissa käytettävistä väreistä on syytä tietää sen verran, että täysin mustaa tai valkoista väriä ei kannata käyttää värikartassa, sillä ne eivät näytä luonnollisilta ja ne niiden valaisu ei onnistu kunnolla. Layerit ovat hyödyllisiä teksturointiprosessissa, joten .psd tai .tga tiedostomuodot ovat hyviä vaihtoehtoja tekstuurien valmistusvaiheessa.

Tekstuureissa on myös oltava tarpeeksi edge paddingia eli tekstuureihin on lisättävä ylimääräistä reunusta saumakohtiin, ettei niihin tule valaistus- tai tekstuuriongelmia. Tämä ongelma liittyy pelimoottoreiden tekemään mip-mappaukseen. Eli pelimoottorit luovat automaattisesti pienemmän resoluution tekstuureita, joita ne sitten käyttävät 3D-malli ollessa tarpeeksi kaukana katsojasta. Mip-mappaus aiheuttaa tekstuurien vuotamista UV-saarekkeiden ulkopuoliselta alueelta, jos ylimääräistä reunusta ei ole lisätty. Monet beikkausohjelmat osaavat tehdä reunusten lisäämisen automaattisesti, mutta se on myös mahdollista lisätä teksturointiohjelmassa.

Tekstuurista on vaikea tietää, että näyttääkö se sellaiselta kuin haluaisi sen näyttävän, jos sitä työstää vain 2D-ohjelmassa. Siksi malli kannattaa viedä pelimoottoriin ja asettaa sille materiaali, johon tarvittavat tekstuurit ovat liitettynä. Tämä on helpoin tapa tarkkailla tekstuurien onnistumista ja samalla saa kuvan, miltä valmis tuotos tulee näyttämään.

Kuva 16. Hahmon värikartta ja normaalikartta.

7 PELIMOOTTORIIN VIEMINEN

Tämä on viimeinen työvaihe, mutta 3D-mallin voi viedä pelimoottoriin tarkasteltavaksi jo mahdollisimman varhaisessa vaiheessa, jotta virheet voidaan havaita ja korjata.

Pelimoottoria varten 3D-malli pitää tallentaa tiettyyn tiedostomuotoon. Tällaisia tiedostomuotoja ovat muun muassa .fbx, .dae, .3ds, .dxf, .obj ja .skp (Unity 2017b). Näistä suositellaan käytettävän .fbx:ää (Unreal Engine). Pelimoottoriin vietäessä kaikki nelikulmiot muuttuvat kolmioksi, jotta renderöinnissä käytetyt laskutoimitukset olisivat helpompia ja siten nopeampia (Silverman 2013). Onkin syytä ottaa huomioon, että todellinen polygonien määrä kasvaa 3D-mallin vietäessä pelimoottoriin. Siksipä pelien kontekstissa olisi suositeltavaa puhua kolmioiden määrästä polygonien määrän sijasta.

Pelimoottoriin viedään 3D-mallin lisäksi tietysti myös sitä varten valmistetut tekstuurit. Koska useat nykyisistä pelimoottoreista osaavat automaattisesti kompressoida tekstuurit tiettyyn formaattiin, tekstuureita ei tarvitse tallentaa optimoituun tiedostomuotoon vaan ne voidaan tallentaa muutosten tekemistä varten johonkin layerit sisältävään tiedostomuotoon. Kompressoidut tekstuurit vähentävät tiedostokokoa, nopeuttavat latausaikoja ja ovat nopeampia renderöidä.

Jotta pelimoottorit voivat hyödyntää tekstuureita 3D-mallissa, täytyy tekstuureita varten tehdä materiaali. Materiaalit yhdistävät halutut tekstuurit ja niiden avulla määrittävät, miten pinta tulisi renderöidä (Unity 2017c). Shader-skriptit pystyvät sitten käyttämään näitä materiaaleja 3D-mallien pintojen renderöinnissä (Unity 2017c). Jos 3D-malli on jaettu eri objekteiksi, joilla on sama materiaali, voidaan osat yhdistää yhdeksi kokonaisuudeksi suorituskyvyn parantamiseksi. Materiaalien määrä kannattaa muutenkin pitää minimissään. Esimerkiksi metallisille osille ei välttämättä tarvitse tehdä omaa materiaaliaan, vaan erilaiset metallipinnat voidaan tuoda esiin spekulaarikartan avulla.

Kuva 17. Kuvankaappaus valmiista mallista Unity-pelimoottorissa.

8 POHDINTA

Digitaalisen kuvanveiston hallitseminen mahdollistaa rajan rikkomisen suunnittelijan ja mallintajan välillä. Jos kokee luontevammaksi työskennellä pelkästään 3D:nä, on suunnitteluvaihe mahdollista toteuttaa 3D:nä ZBrushin avulla ja koska digitaalinen kuvanveisto parhaimmillaan ei vaadi paljoa teknistä osaamista, myös 2D-grafiikan taitajat voivat tätä kautta perehtyä 3D-mallintamiseen. Suunnittelijan ja mallintajan roolin yhdistäminen on tärkeää etenkin Suomen mittakaavassa, koska pelifirmoilla on harvoin varaa palkata kovin montaa graafikkoa.

Moni 3D-mallintaja varmasti kokee digitaalisen kuvanveiston käyttämisen peleihin mallinnuksessa haasteellisena. Ennen mallintamisen pystyi hoitamaan kokonaan yhden ohjelman avulla, mutta normaalikarttojen ilmestymisen myötä prosessista on tullut paljon monivaiheisempi. Nykyisin melkein jokaista vaihetta varten on oma ohjelmansa opittavaksi ja vaadittavat tekniikat ovat lisääntyneet. Mielestäni tälle prosessille saatiin määriteltyä selkeisiin osiin jaoteltu työnkulku, josta on varmasti hyötyä aiheesta kiinnostuneille. Tähän työnkulkuun kuuluvat siis:

- Highpolymallin veistäminen
- Paremman topologian omaavan lowpoly-mallin rakentaminen highpoly-mallin pohjalta
- Lowpoly-mallin UV-kartoitus
- Highpoly-mallin yksityiskohtien beikkaaminen lowpoly-malliin
- Tekstuurien tuottaminen beikattuja tekstuurikarttoja apuna käyttäen.

Lopputuloksena pelimoottoria varten optimoitu 3D-malli.

Hyvä suorituskyky on kriittistä pelin onnistumisen kannalta. Pelin grafiikoiden renderöinti vie usein suurimman osan pelin suorituskyvystä, joten grafiikoiden optimointi on hyvin tärkeää. Optimoinnissa tärkeintä on keskittyä käytettävien polygonien määrään, mallin topologiaan, UV-saarekkeiden määrään ja tekstuurikarttojen kokoon.

Opinnäytetyössä esimerkkinä toimineen mallin toteuttamiseen käytetyt ohjelmat olivat: Pixologic ZBrush 4R7 (highpoly-mallin veistäminena), Topogun (topologian uudelleen rakentaminen), Cinema 4D (UV-kartoitus ja mallien yleinen käsittely), Substance Painter

2 (beikkaus) ja Photoshop CC (teksturointi). Samankaltaiset tulokset on mahdollista saavuttaa muissakin ohjelmissa, sillä monet 3D-ohjelmista sisältävät samankaltaisia työkaluja. Suosittuja ohjelmia eri työvaiheisiin ovat:

- Digitaalinen kuvanveisto:
    - ZBrush
    - Mudbox
    - 3D-Coat
- Topologian uudelleen rakentaminen:
    - Topogun
    - ZBrush
- UV-kartoitus:
    - Headus UVLayout
- Beikkaus:
    - Substance Painter/Designer
    - xNormal
    - Knald
    - Marmoset Toolbag 3
- Teksturointi:
    - Photoshop
    - Substance Painter/Designer
    - Quixel Suite
- Yleinen 3D-mallin käsittely (mahdollistavat myös kaikki yllä mainitut työvaiheet ainakin jollain tasolla):
    - Autodesk Maya
    - Autodesk 3ds Max
    - Cinema 4D
    - Modo
    - Blender

Kun asiat kirjoittaa konkreettiseen muotoon, huomaa kuinka abstraktilla tasolla oma ymmärrys asioista on. Näiden asioiden kirjoittaminen on auttanut ymmärtämään 3D-mallinnukseen liittyviä asioita selkeämmin ja sitä myötä parantamaan siihen liittyviä taitojani. Haasteellista oli koota koko työprosessi lineaariseen tekstimuotoon, sillä opinnäytetyössä

kuvattu työnkulku menee harvoin täysin suunnitellulla tavalla. Eri vaiheiden välillä joutuu liikkumaan joustavasti, joko unohdusten tai muiden ongelmien takia. On kuitenkin tärkeää olla olemassa jonkinlainen suunnitelma mitä seurata.

3D-ohjelmiin yritetään koko ajan kehittää uusia työkaluja, jotka pystyisivät automatisoimaan eri työvaiheita. Moni osa tässä opinnäytetyössä läpikäydyistä asioista on mahdollista toteuttaa automatisoiduilla tekniikoilla. Manuaalisilla tekniikoilla saavutetaan vielä kuitenkin optimaalisimmat lopputulokset etenkin pelimoottorin kannalta, joten halusin tässä opinnäytetyössä keskittyä näihin tekniikoihin. Esimerkkinä automatisoiduista tekniikoista on Substance Painter, jolla pystyy generoimaan proseduraalisia tekstuureita. Lisäksi uuden topologian ja UV-kartoituksen voi tehdä automaattisesti esimerkiksi ZBrushissa. Tulevaisuudessa nämä työvaiheet, jotka vaativat vähemmän artistista silmää ja enemmän teknistä osaamista, tulevat todennäköisesti olemaan täysi automatisoituja työkalujen parantuessa. Jatkotutkimusaiheet voisivat liittyä näihin automatisoituihin tekniikoihin, joiden avulla prosessista saataisiin karsittua työvaiheita.

LÄHTEET

Keller, E. 2012. Introducing ZBrush. John Wiley & Sons, Incorporated.

Scherer, M. 2011. ZBrush 4 Sculpting for Games Beginner's Guide : Beginner's Guide. Packt Publishing.

Spencer, S. 2008. ZBrush Character Creation : Advanced Digital Sculpting. Wiley.

Ward, A. 2008. Game Character Development: Digital Sculpting for the Realtime Artist. Course Technology.

Polycount. 2016a. Texture Baking. Luettu 5.4.2017. http://wiki.polycount.com/wiki/Texture_Baking

Polycount. 2017. Edge Padding. Luettu 5.4.2017. http://wiki.polycount.com/wiki/Edge_padding

Polycount. 2015. Base Mesh. Luettu 6.7.2017. http://wiki.polycount.com/wiki/BaseMesh

Polycount. 2016b. Texturing. Luettu 26.7.2017. http://wiki.polycount.com/wiki/Texturing

Unity. 2017a. Unity User Manual (2017.1). Modeling characters for optimal performance. Luettu 7.8.2017. https://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html

Unity. 2017b. Unity User Manual (2017.1). 3D Formats. Luettu 7.8.2017. https://docs.unity3d.com/Manual/3D-formats.html

Unity. 2017c. Unity User Manual (2017.1). Materials, Shaders & Textures. Luettu 7.8.2017. https://docs.unity3d.com/Manual/Shaders.html

Pixologic a. User Guide. DynaMesh. Luettu 6.4.2017. http://docs.pixologic.com/user-guide/3d-modeling/modeling-basics/creating-meshes/dynamesh/

Pixologic b. User Guide. DynaMesh. Luettu 6.4.2017. http://docs.pixologic.com/user-guide/3d-modeling/modeling-basics/subtools/

Pixologic c. Reference Guide. Draw. Luettu 6.4.2017. http://docs.pixologic.com/reference-guide/draw/

Pixologic. 2017a. Zbrush Overview. Luettu 7.4.2017. http://pixologic.com/zbrush/features/overview/

Pixologic. 2017b. Zbrush Features. Luettu 8.7.2017. http://pixologic.com/features/

Taylor, J. 2015. Why are triangles bad when modeling? Luettu 26.7.2017. https://www.methodj.com/why-are-triangles-bad-when-modeling/

Silverman, D. 2013. 3D Primer for Game Developers: An Overview of 3D Modeling in Games. Luettu 26.7.2017. https://gamedevelopment.tutsplus.com/articles/3d-primer-for-game-developers-an-overview-of-3d-modeling-in-games--gamedev-5704

Munoz, P. 2015. 3D Lingo. Luettu 4.7.2017.
http://www.zbrushguides.com/3d-lingo/

Autodesk Maya. 2014. Maya User Guide. Introduction to UV mapping. Luettu 4.7.2017.
https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/UV-mapping-overview-Introduction-to-UV-mapping-htm.html

Crytek. 2013. CryEngine Technical Documentation. Luettu 4.7.2017.
http://docs.cryengine.com/display/SDKDOC4/Tangent+Space+Normal+Mapping#TangentSpaceNormalMapping-BenefitsofTangentSpaceNormalMaps

Unreal Engine. Importing Static Meshes. Luettu 4.7.2017.
https://docs.unrealengine.com/latest/INT/Engine/Content/Types/StaticMeshes/HowTo/Importing/

Kuvalähteet:

Wikipedia. 2017. UV Mapping. Luettu 4.7.2017
https://en.wikipedia.org/wiki/UV_mapping

Munoz, P. 2015. 3D Lingo. Luettu 4.7.2017.
http://www.zbrushguides.com/3d-lingo/

Pixologic d. Reference Guide. Brush. Luettu 18.8.2017.
http://docs.pixologic.com/reference-guide/brush/

Allegorithmic. Substance Designer. Luettu 18.8.2017.
https://www.allegorithmic.com/products/substance-designer

Pixologic e. User Guide. FBX Settings. Luettu 4.9.2017.
http://docs.pixologic.com/user-guide/zbrush-plugins/fbx-exportimport/fbx-settings/